Application Report
SLAA665B–February 2015–Revised June 2015

Using the MSP430™ RTC_B Module With Battery Backup Supply

Katie Pier *MSP430 Applications*

ABSTRACT

Some applications need to retain an accurate real-time clock (RTC) through battery changes, power outages, and other events. For these applications, MSP430™ devices that contain the RTC_B Real-Time Clock module with a Battery Backup System can be a great fit. This application note demonstrates how to use RTC_B with Battery Backup Supply functionality to retain the time and keep the RTC counting through loss of main power supply, as well as handling correct reinitialization when the main power supply is restored.

The associated demo code can be downloaded from www.ti.com/lit/zip/slaa665.

Contents

List of Figures

List of Tables

1 Introduction: RTC_B Use With Battery Backup

Some MSP430 devices contain the RTC_B Real-Time Clock module with a Battery Backup System (for example, the MSP430F66xx, MSP430F56xx, MSP430F64xx, MSP430F53xx, and MSP430FG66xx devices). The Battery Backup System allows the RTC_B to keep running with the external crystal and keep time even when the main supply is lost, by running off of a backup battery. Note that this is a different mechanism than the RTC_C module with AUXPMM supply found on some other MSP430 devices (for example, the MSP430F67xx devices).

When the main supply is restored, the device resets. Therefore, the user application code needs to distinguish if a supply switching event occurred and RTC time data is retained, or if a reset was caused by full power loss (including loss of battery supply) and time data was lost, to handle these two different events accordingly. This application note describes how to detect if a switching event caused the reset or if a full brownout reset (BOR) caused it, and how to initialize the RTC in these different circumstances to retain time information when possible.

2 Detection Methodology

Three methods are used to determine whether the RTC calendar information might not be accurate or reliable on reset, or if time was retained during backup supply operation. Only a few parts of the MSP430 device are powered while running from the backup supply – the crystal oscillator, the backup memory registers, and parts of the RTC module. Therefore the backup supply switching detection method uses three different pieces of information that will be retained through the backup supply switching reset:

- a signature written into backup memory
- the RTC oscillator fault flag
- the RTC calibration register

The RTC calendar information is considered valid and useable only if all three criteria are met. If any of the tests is false, then the RTC calendar time information may not be accurate, and the RTC should be reinitialized.

2.1 Backup RAM Memory

MSP430 devices with a battery backup system, such as the MSP430F6638, contain a few words of backup RAM memory. These backup memory registers are typically used to store important variables or state information that need to be retained through a backup supply switching event, because normal RAM is not retained while running from the backup supply.

If the backup supply is lost, (for example, if the backup battery drains while running from backup supply or if the battery is removed and the caps drain) then backup RAM will not be retained. This property is used in the example code as one of the checks performed to detect backup switching events – the code writes a signature into BAKMEM0 at first startup. After a reset, the code checks to see if the signature is still present – if it is not, then backup memory was not retained, so any stored RTC calendar information cannot be considered reliable because backup supply was lost. This should be treated as a full BOR reset event rather than a reset from returning from successful backup supply operation.

2.2 RTC_B Oscillator Fault

The RTC Oscillator Fault Flag RTCOFIFG is checked to make sure that the RTC continued to count correctly during backup operation. In addition to issues if the device backup supply loses power, if something causes a fault on XT1 that is supplying the RTC during backup operation, the RTC calendar time also may not be accurate. An XT1 fault could mean that the crystal stops or oscillates at a wrong frequency. Either case would cause the RTC calendar time to be inaccurate or have lost some time by not ticking while the crystal was in a fault state. Therefore, the code must check RTCOFIFG to see if the RTC was able to run without fault during the entire time it was on the backup supply, to verify that the calendar time is accurate and useable.

2.3 RTC Calibration Register

According to the RTC_B module registers in the *MSP430x5xx and MSP430x6xx Family User's Guide* (

SLAU208), only a few registers within the RTC_B module are retained during backup operation or LPMx.5 operation. See the registers list in the RTC_B chapter of the user's guide to determine which registers are retained. Most retained registers are the RTC calendar information. However, this information is being updated by the RTC during backup operation, so it has no expected value that can be checked to determine correct backup supply switching. The RTCCTL2 and RTCCTL3 registers typically contain calibration information for the RTC, so they are also retained.

The example code uses the RTCCTL2 register as another check. Similar to the Backup RAM checking described in Section 2.1, a known value is set in this register at startup and then checked by the code after a reset to see if the value was retained. This is checked in addition to the backup RAM signature for added robustness. It is not defined in the data sheet specification whether on power loss backup RAM or the RTC registers will degrade and lose retention first. Checking both helps ensure that some edge condition (for example, a very brief glitch or power loss) does not occur where RTC registers may have lost retention or changed but backup RAM had not yet degraded, and have the software incorrectly assume RTC data is good in this case. It provides another layer of checking to make sure nothing unexpected occurred in the system during backup switching or operation.

The RTCCTL2 register is normally used to calibrate the RTC in applications. The setting of RTCCALx bits will cause the RTC to make an adjustment (± ppm) periodically in hardware, to compensate for error in the crystal or system. Typically in an application, this would be measured in the system at design or production and then code would set this register to the correct value for the system. So it is not advised to simply set an arbitrary value for the RTCCALx bits, as this will cause the RTC to make adjustments – so there are two options:

- Uncalibrated system – if there is no RTC calibration being used in the application or system, leave RTCCALx = 0, and set the RTCCALS bit = 1. (This is what is done in the code example provided). When RTCCALx = 0, no adjustment is made. RTCCALS simply sets the direction of the adjustment, so it can be used for the reset checking without impacting the RTC behavior as long as the RTCCALx = 0 for no adjustment. RTCCALS is set to 1 in this case so that it can be distinguished from the default startup value (0) to determine if RTC registers were retained or not.
- Calibrated system – if RTC calibration is being used in the application (RTCCALx is a nonzero value), then use this application-specific RTC calibration value for RTCCALx as the reset checking value. This value is different than the startup value of RTCCALx = 0, and so can be used to check if RTC registers were retained or not.

3 Software Implementation

Example software can be downloaded from www.ti.com/lit/zip/slaa665.

The following section describes key features of the software implementation in the RTC_B_with_Backup software project, to help demonstrate a way to implement the reset checking methodology described in chapter 2.

3.1 Power Management Module (PMM) and Supply Voltage Supervisor (SVS) Setting

The supply voltage supervisor (SVS) in the power management module (PMM) of the MSP430 devices controls the switching between DV_{CC} supply and the backup battery supply. In addition, using the correct PMM core voltage and correct SVS settings for this core voltage is important for correct operation of the MSP430 device core. Therefore, it is vital that the PMM and SVS are configured correctly for an application to work correctly with the battery backup system.

It is recommended to use driverlib in MSP430ware for configuring the V_{core} level of the device. The example code uses the driverlib API function PMM_setVcore() to configure the V_{core} level and correct SVS settings for proper supply switching. This function not only properly sets the V_{core} level (incrementing or decrementing one step at a time and doing all proper delays and checks as specified in the PMM section of the user's guide), but it also properly configures the SVS and SVM settings to the appropriate levels for the selected V_{core} level. Using this recommended driverlib function ensures that users have the correct SVS settings for proper supply switching so that the device correctly switches to and runs from the backup supply when DV_{CC} is removed. Additionally, the SVMH level must always be greater than the SVSH level for the SVS to work correctly (this is noted in the SVS section of the user's guide). The driverlib default settings from setVcore assure this, but users should be aware of this requirement in case they do any additional configuration of the SVS module.

Always check the data sheet *Recommended Operating Conditions* figure that shows frequency vs supply voltage to determine the minimum required core voltage setting (0, 1, 2, or 3) for the MCLK operating frequency of the application. Then use this setting when calling PMM_setVcore(). For example, if the system is running at ≥16 MHz, the function would be:

```
PMM_setVcore(PMM_CORE_LEVEL_3);
```

3.2 LOCKBAK and Crystal Oscillator

When the device switches to the backup supply, the backup-supplied subsystem retains its settings and the LOCKBAK bit is set. This means that the backup RAM is stored and that the RTC and crystal keep running at their current settings. While this is locked (LOCKBAK = 1), the data cannot be accessed and no new settings take effect. When DV_{CC} is restored and the device resets from this backup switching event, LOCKBAK is still 1, so the crystal and RTC are still running the settings from when the device switched to VBAT power. See the *Battery Backup System* chapter in the *MSP430x5xx and MSP430x6xx Family User's Guide* (SLAU208) for more information.

This means that before being able to check BAKMEM0 or the RTC settings to determine if a switching event occurred and RTC calendar time is still good, LOCKBAK must be cleared. However, clearing LOCKBAK causes the current RTC or XT1 settings to take effect. Therefore, the example code (see Example 1) reinitializes the crystal, reinitializes the RTCBCD, and clears the RTCHOLD bit before clearing LOCKBAK so that the RTC will continue to run when LOCKBAK is cleared.

Example 1. Settings to Reinitialize Before Clearing LOCKBAK

```
//We have to reinit some crystal and RTC settings before clearing LOCKBAK so that crystal/RTC won't
stop if we're coming out of VBat
//init crystal
UCSCTL6 = XT1DRIVE_3 | XCAP_3;    //set up the crystal load capacitors for 12pF
//init RTC
RTCCTL01 |= RTCBCD;        //use RTC in BCD mode
RTCCTL01 &= ~RTCHOLD; //keep RTC running when we clear LOCKBAK

//erase LOCKBAK bit for crystal + RTC settings to take effect
//you must erase LOCKBACK or you won't be able to check RTC flag etc
while(BAKCTL & LOCKBAK)
{
    BAKCTL &= ~(LOCKBAK);
}
```

3.3 Reset Checking Decision Logic

After LOCKBAK has been cleared, the BAKMEMx and RTC registers can be checked to determine if the RTC calendar data is valid or not (if the device is coming out of a successful backup switching event, or if this is a full reset). The code checks BAKMEM0 for the signature that is written into it by the later initialization code, the RTCOFIFG to determine if any oscillator faults occurred while running from the backup supply, and the RTCCTL2 calibration setting to see if it matches the setting from later initialization code. See Section 2 for more details on the checking methodology.

In Example 2, the calendar data is only deemed good if all three conditions evaluate true – no fault, signature present, and calibration present. If any of these conditions fails, then the calendar data is considered unreliable and the RTC is completely reinitialized.

Example 2. Reset Checking

```
//check if backup supply was drained or not by checking BAKMEM0, RTCOFIFG, and RTCCALS
//BAKMEM0 we are checking for backup memory retention
//RTCOFIFG we are checking if an oscillator fault occurred while we were on the backup supply
//RTCCALS will be retained through switching event but cleared on brownout, so also indicates if RTC
calendar registers retained
//NOTE: RTCCTL2 setting and checking should use whatever calibration values are required for the
specific device/application. RTCCALS is used as an example in this case.
if((BAKMEM0 == 0xDEAD) && !(RTCCTL01 & RTCOFIFG) && (RTCCTL2 & RTCCALS)) //backup not lost
{
```

3.4 Reset Reinitialization

If the checks all evaluate true, then the device has successfully returned from a backup supply switching event. The RTC calendar time data must be retained, and the RTC is still running (so there is no need to stop and reinitialize it). Therefore initialization in this case is minimal, simply reinitializing other system functions as seen in Example 3.

Example 3. Reinitialize After Successful Backup Switch

```
if((BAKMEM0 == 0xDEAD) && !(RTCCTL01 & RTCOFIFG) && (RTCCTL2 & RTCCALS)) //backup not lost
{
    timeLost = 0;     //we retained the RTC data

    P1DIR |= BIT0;    //RTC LED reinitialize to be output (GPIO are not retained)
}
```

If any of the checks evaluates false, then the device must completely reinitialize the RTC (see Example 4). First the crystal is reinitialized to restart it in case there was an oscillator fault or if this was first startup. Then, the RTC calendar is reinitialized to the default starting time and date.

Finally, and importantly, the signatures used for the reset-checking must be initialized so they can be checked after a backup switching event. This means writing the signature into BAKMEM0, and writing in the RTC calibration data in RTCCTL2 (this example uses an uncalibrated RTC and so simply sets RTCCALS as a check without any calibration impact). See Section 2 for more information.

Example 4. Reinitialize After Lost Backup RAM

```
else //either first startup or we lost backup RAM
{
    timeLost = 1;

    P1OUT &= ~BIT0;
    P1DIR |= BIT0;        //RTC LED

    //Clear faults
    clearOscFault();

    //reinitialize the RTC time
    resetTime();

    RTCCTL2 |= RTCCALS;   //this bit is retained so can be used as a check for VBAT switching
    BAKMEM0 = 0xDEAD;     //write in the signature
}
```

3.5 Oscillator Fault Handling

For robustness, it is important to catch RTC oscillator faults. If the crystal oscillator sourcing the RTC faults and is not properly running, then the RTC may not continue counting or may count at the wrong frequency – either case will result in an inaccurate time. Therefore, it is important to detect RTC oscillator faults during runtime, and reinitialize the crystal and RTC (and notify the system that the RTC time is no longer accurate). This is easily implemented by simply enabling RTC oscillator fault interrupts during RTC initialization. When a fault occurs, in the RTC ISR, the example code sets a software flag oscFault and wakes the device to handle the fault (see Example 5).

Example 5. Oscillator Fault Wakeup

```
case 12: //RTCOFIFG
    oscFault = 1;
    RTC_B_disableInterrupt(RTC_B_BASE, RTC_B_OSCILLATOR_FAULT_INTERRUPT); //disable further fault
interrupts
    __bic_SR_register_on_exit(LPM3_bits); //wake to handle fault
    __no_operation();
    break;
```

After a wake event, the main loop checks oscFault, and if it is set it clears the fault repeatedly until the fault stays clear (indicating the oscillator is now running correctly again). Then the code resets the RTC calendar time and sends a message over the UART to indicate that an oscillator fault occurred and time was reset. Finally, the RTC is restarted (see Example 6).

Example 6. Oscillator Fault Handling

```
if(oscFault)
{
    //check until fault clears
    clearOscFault();
    oscFault = 0;

    //reset time
    resetTime();

    //print message about osc fault and time resetting
    transmitString((unsigned char*)oscFaultString, OSC_FAULT_STRING_LENGTH);

    //restart RTC
    //Enable interrupt for RTC Ready Status, which asserts when the RTC
    //Calendar registers are ready to read.
    RTC_B_clearInterrupt(RTC_B_BASE, RTC_B_CLOCK_READ_READY_INTERRUPT |
RTC_B_OSCILLATOR_FAULT_INTERRUPT); //clear flag before enabling interrupt
    RTC_B_enableInterrupt(RTC_B_BASE, RTC_B_CLOCK_READ_READY_INTERRUPT |
RTC_B_OSCILLATOR_FAULT_INTERRUPT);    //enable ready interrupt and fault interrupt
    RTC_B_startClock(RTC_B_BASE);    //start clock
}
```

3.6 Other Functions

After the decision logic and RTC initialization is all handled, code should initialize all other needed modules as usual. The RTC_B_with_Backup example initializes the USCI_A0 module for sending data using the UART to a host device. This is so that the RTC time can be output and visible to the user, to show that time is correctly kept even while in backup mode. The UART output is primarily done with some helper functions provided in UART.c and UART.h, which make use of the driverlib UART functions for ease of use.

In a real application, all other code functions can be written as normal, with a couple of things to keep in mind for RTC with backup operation. Observe the RTC_B_ISR as shown in Example 7. At the beginning of this ISR, the LOCKBAK bit is cleared before anything else – this is necessary for LPM3 operation with the RTC_B module. See the user's guide for more information.

Example 7. RTC_B ISR

```
void RTC_B_ISR(void)
{
    while(BAKCTL & LOCKBAK)                    // Unlock backup system
         BAKCTL &= ~(LOCKBAK);

    switch (__even_in_range(RTCIV, 16)) {
```

One other thing to consider is the startup states of modules after a reset – most modules other than battery backup RAM and the RTC will be reset to default states by a backup supply switching event, and will need to be reinitialized for proper operation.

4 Running the Demo

This section discusses how to run and evaluate the software example that is associated with this application note. The software can be downloaded from www.ti.com/lit/zip/slaa665.

The example uses two hardware boards – one for the device under test (MSP-TS430PZ100USB with MSP430F6638) and another (MSP-EXP430F5529LP) that acts as a host that controls the DV_{CC} and VBAT rails and also sends data back to the PC for display on a terminal. This allows the user to see the time and date information on the PC to observe that time was retained while running off of the backup supply.

4.1 Required Hardware Tools

- MSP-TS430PZ100USB target board
  - MSP430F66xx, MSP430F56xx, MSP430F64xx, or MSP430F53xx device
- MSP-EXP430F5529LP + USB connection to a PC
- 5 jumper wires to connect the boards
- MSP-FET tool for programming the MSP430 device
  - MSP-EXP430F5529LP may alternately be used for programming the MSP430 device on the target board—see the section *Using the eZ-FET Lite Emulator with a Different Target* in the *MSP-EXP430F5529LP LaunchPad™ Development Kit User's Guide* (SLAU533).

4.2 Required Software Tools

- Code Composer Studio v 6.0 (or later) for MSP430
- Alternately, IAR Embedded Workbench for MSP430
- PC terminal software
- ZIP file: www.ti.com/lit/zip/slaa665

4.3 Code File Structure

The code is available for download here: www.ti.com/lit/zip/slaa665

The zip file contains a folder for IAR and a folder for CCS. Each folder contains the RTC_B_with_Backup and Demo_Host_F5529LP projects. RTC_B_with_Backup contains the target demo software for the MSP430F66xx device. Demo_Host_F5529LP contains the host code for the MSP-EXP430F559LP LaunchPad kit that interfaces to the MSP430F66xx device for testing and to display data on the PC for the user.

The code was built using CCSv6.0.1.00040 and MSP430ware/Driverlib version 1.95.00.32.

4.4 Setup

1. Build and load the project RTC_B_with_Backup into the MSP430F66xx device on the MSP-TS430PZ100USB target board. Disconnect from PC.
2. Build and load the project Demo_Host_F5529LP onto the MSP-EXP430F5529LP LaunchPad kit. Disconnect from PC.
3. Connect the two boards together with jumper wires according to Table 1.

Table 1. Board Connections

MSP-EXP430F5529LP	Signal	MSP-TS430PZ100USB
P2.0 (J5)	V_{CC}	VCC (JP3 pin 2)
GND (J10)	GND	GND
3V3 (J10)	VBAT	VBAT (JP11)
P3.4 (J1)	F6xx TXD	P2.0 (pin 17)
P3.3 (J1)	F6xx RXD	P2.1 (pin 18)

4. Connect the micro USB of MSP-EXP430F5529LP to the PC to power both boards. At power-up, the MSP-EXP430F5529LP red LED should be on and the green LED should be off.
5. In Device Manager on the PC, find the number of the COM Port labeled "MSP Application UART1". Using the terminal software, connect to this COM port with settings: 9600 baud, none parity, 8 data bits, and 1 stop bit.
6. If you reset the MSP-EXP430F5529LP using the RST button, you should see this message in the terminal: "Program start. Press button to enable Vcc."

4.5 Running and Evaluating the Demo

When the MSP-EXP430F5529LP connected to the MSP-TS430PZ100USB board is first powered up, it sends a message to the terminal program on the PC, and the red LED on the LaunchPad kit is lit to indicate that there is no power on V_{CC} to the MSP430 device.

4.5.1 Power Control

Press the button S1 on the MSP430F5529 LaunchPad kit to toggle power to DV_{CC} of the MSP430F6xx device.

- When power is being supplied to DV_{CC} on the MSP430F6xx from P2.0 on the MSP430F5529, the MSP430F5529 LaunchPad kit green LED is lit.
- When no power is supplied, the P2.0 GPIO on the MSP430F5529 is set as an input to simulate disconnecting the DV_{CC} on the MSP430F6xx, and the MSP430F5529 LaunchPad kit red LED is lit.

Power is always being supplied to the MSP430F6xx VBAT from the 3V3 pin on the MSP430F5529 LaunchPad kit.

4.5.2 Reset and Switching Events

Each time the button is pressed, there will be a message about whether time was retained or reset based on the detection method described in Section 2 – this message comes from the MSP430F6xx code for doing the check and is simply reported to the MSP430F5529 so the user can see this information in the terminal.

- When the detection logic determines that a reset event or RTC oscillator fault occurred (VBAT was lost, oscillator fault while running on backup supply, or first startup), the message displayed is: "Reset event. Time was reset." The RTC is set by code to an initial starting value of 05/04/2015 04:30:00.

NOTE: The first value sent over UART is 1 second later (04:30:01), because it is sent by the first RTC event after one second has passed from RTC start.

- When the detection logic determines that a switching event occurred (VBAT was maintained and the RTC kept running while DV_{CC} was removed), the message displayed is "Switching event. Time was retained." The RTC retained its calendar information and continued to keep time while DV_{CC} was removed – the next time message will reflect this.

To force a reset event, the user can simulate a loss of VBAT by briefly connecting the VBAT connection to GND while DV_{CC} is disabled (red LED is on). This drains the capacitors on VBAT so that the level drops and causes a reset as if a battery connected to VBAT had drained while the application was running on battery supply. When the LaunchPad kit button is pushed again to restore power, the report will show that a reset has occurred and time has been reset (just like the first startup).

While V_{CC} is off, the user can also force an RTC oscillator fault during backup operation, by touching a piece of wire or other conductive metal to briefly short the XT1 crystal pins on Q1 on the MSP-TS430PZ100USB target board. This causes RTCOFIFG to be set and the crystal to stop/be disturbed. The software will catch this state as well when DV_{CC} is restored, and again report a reset event because the time information cannot be considered reliable in this case - due to the crystal fault the RTC may have stopped counting or counted at a wrong frequency. If an RTC oscillator fault occurs while on the normal supply (instead of during backup operation), the software will catch this as well in the RTC ISR and send a UART message indicating that an oscillator fault occurred and time was reset.

Tests were performed on MSP-TS430PZ100USB revision 1.25 target board, with MSP430F6639 revision D. The host was an MSP-EXP430F5529LP LaunchPad kit revision 1.5 with MSP430F5529 revision I.

4.5.3 Demo UART Messages on the Terminal

The following message can be displayed on the PC. See Section 4.5.4 for examples.

- Program start. Press button to enable V_{CC}.

  Sent from MSP430F5529 when it is first powered up.
- V_{CC} On.

  P2.0 on MSP430F5529 was switched on to provide 3.3 V to the MSP430F66xx DVCC pin.
- V_{CC} Off.

  P2.0 on MSP430F5529 was set as an input to disconnect the MSP430F66xx DVCC pin.
- MM/DD/YYYY HH:MM:SS (for example, 05/04/2015 04:30:01)

  The current time reported by the MSP430F66xx device. This is reset on a reset event but is retained (counting continues) if proper VBAT switching occurred.
- Reset event. Time was reset.

  The MSP430F66xx code checking per Section 2 determined that the device reset or RTC information may not be reliable due to another cause (like oscillator fault). The calendar registers of the RTC are then reset to the initial time setting by the code.
- Switching event. Time was retained.

  The MSP430F66xx code checking per Section 2 determined that the device underwent a DV_{CC} switching event but VBAT was available and supply switching occurred. In addition, there was no oscillator fault during backup supply operation. The calendar registers of the RTC were retained, so the code does not overwrite them.

- Oscillator Fault. Time was reset.

  The crystal on the MSP430F66xx board encountered a fault condition. Because RTC time can no longer be considered correct, the RTC calendar registers are reset to the starting time. If an oscillator fault occurs while on the backup supply, this is handled by the reset event message above, but if an oscillator fault occurs during normal operation, this message is given.

4.5.4 Example Terminal Output

Figure 1 shows sample output on the PC.

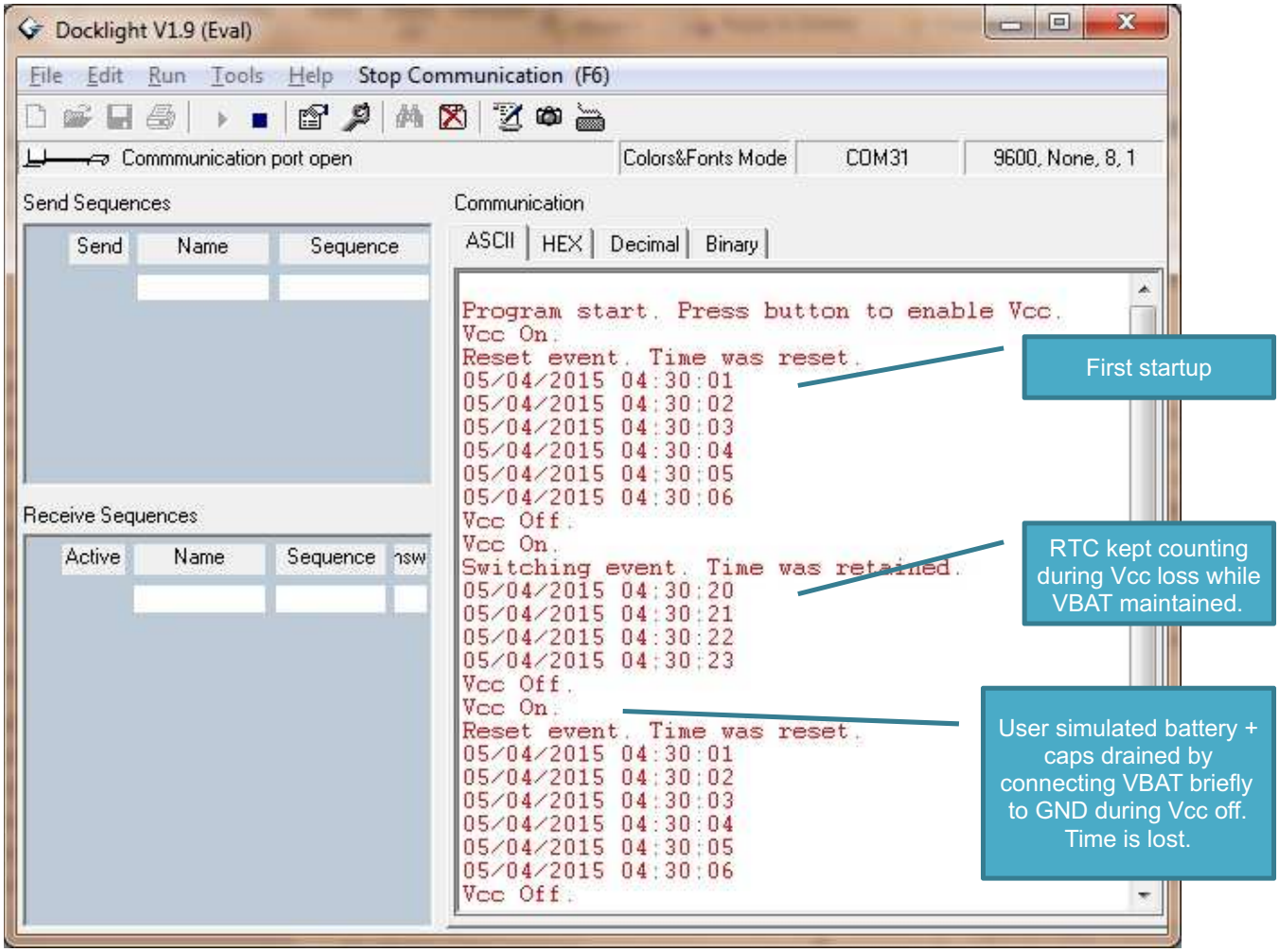

Figure 1. Example Terminal Output

5 Additional Resources

1. *MSP430x5xx and MSP430x6xx Family User's Guide* (SLAU208)
2. MSP430ware (MSP430WARE)
3. *MSP430F663x Mixed-Signal Microcontrollers* (SLAS566)
4. MSP430F6638 product folder
5. MSP-TS430PZ100USB Target Board tool folder (MSP-TS430PZ100USB)
6. MSP430F5529 USB LaunchPad Evaluation Kit tool folder (MSP-EXP430F5529LP)
7. Code Composer Studio (CCS) Integrated Development Environment (IDE) (CCSTUDIO)
8. MSP430 Flash Emulation Tool (MSP-FET)

Revision History

Changes from May 26, 2015 to June 4, 2015	Page
• Exchanged positions of "P2.0 (J5)" and "3V3 (J10)" in the MSP-EXP430F5529LP column of Table 1, *Board Connections*	8

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

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265