SMART ARM-based Microcontrollers

AT10458: SAM L22 Tamper Detection using RTC Module

APPLICATION NOTE

Introduction

This application note demonstrates the Tamper Detection feature of the RTC (Real Time Counter) module in the Atmel® | SMART SAM L22 microcontrollers, with various configurations and code examples.

Tamper detection is the ability of a device to sense that an active attempt to compromise the device integrity or the data associated with the device is in progress; the detection of the threat may enable the device to initiate appropriate defensive actions. The main application of tamper detection is in utility metering system like energy meter, gas meter, water meter, etc. Tamper detection senses the unwanted activities that attempts to stop the operation of utility meters.

This application note describes the various methods of tamper detection. Here, the RTC is configured in clock/calendar mode, which keeps the track of the real time. On detection of a tamper attempt, an error message is displayed on the LCD and an alarm is set.

Table of Contents

1. Glossary

AC	Alternating Current
ADC	Analog to Digital Converter
ASF	Atmel Software Framework
DC	Direct Current
DMA	Direct Memory Access
EDBG	Embedded Debugger
EEPROM	Electrically Erasable Programmable Read Only Memory
IDE	Integrated Development Environment
PC	Personal Computer
PCB	Printed Circuit Board
RTC	Real Time Counter
RWW	Read While Write
SERCOM	Serial Communication interface
SLCD	Segmented Liquid Crystal Display
TCC	Timer/Counter for Control Application
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

.

2. Prerequisites

This application note requires the user to possess basic familiarity with the following skills and technologies:

- Atmel Studio 6.2 or later
- ASF version 3.26.10 or later
- SAM L22 Xplained Pro kit
- SLCD1 Xplained Pro kit
- Terminal window - Tera Term or Hyper terminal for Windows®

Note: Make sure that the SAM L22 Part Pack for Atmel Studio is installed before using the example application.

3. RTC - Real Time Counter

A real-time clock (RTC) is peripheral (most often in the form of an integrated circuit) that keeps track of current time. RTCs often have an alternate source of power, so they can continue to keep time while the primary source of power is off or unavailable. This alternate source of power is normally a battery.

3.1. Overview

The Real-Time Counter (RTC) module of the SAM L22 family is a 32-bit counter with a 10-bit programmable prescaler that typically runs continuously to keep track of the time. The RTC can wake up the device from sleep modes by using the alarm/compare wake up, periodic wake up, or overflow wake up mechanisms, or from the wake inputs.

The RTC can generate periodic peripheral events from the outputs of the prescaler, as well as alarm/compare interrupts and peripheral events, which can trigger at any counter value. Additionally, the timer can trigger an overflow interrupt and peripheral event, and can be reset on the occurrence of an alarm/compare match. This allows periodic interrupts and peripheral events at very long and accurate intervals.

The 10-bit programmable prescaler can scale down the clock source. By this, a wide range of resolutions and timeout periods can be configured. With a 32.768kHz clock source, the minimum counter tick interval is 30.5µs, and timeout periods can range up to 36 hours. For a counter tick interval of 1s, the maximum timeout period is more than 136 years.

3.2. Features

- 32-bit counter with 10-bit prescaler
- Multiple clock sources
- 32-bit or 16-bit counter mode
- Clock/Calendar mode
  - Time in seconds, minutes, and hours (12/24)
  - Date in day of month, month, and year
  - Leap year correction
- Digital prescaler correction/tuning for increased accuracy
- Overflow, alarm/compare match and prescaler interrupts and events
  - Optional clear on alarm/compare match
- Tamper Detection
  - Timestamp on event or up to five inputs with debouncing
  - Active layer protection
- Capable of running in Backup sleep mode

3.3. Principle of Operation

The RTC keeps track of time in the system and enables periodic events, as well as interrupts/events at a specified time. The RTC consists of a 10-bit prescaler that feeds a 32-bit counter. The actual format of the 32-bit counter depends on the RTC operating mode.

The RTC can function in one of the following modes:

- Mode 0 - COUNT32: RTC serves as 32-bit counter

- Mode 1 - COUNT16: RTC serves as 16-bit counter
- Mode 2 - CLOCK: RTC serves as clock/calendar with alarm functionality

3.3.1. 32-bit Counter (Mode0)

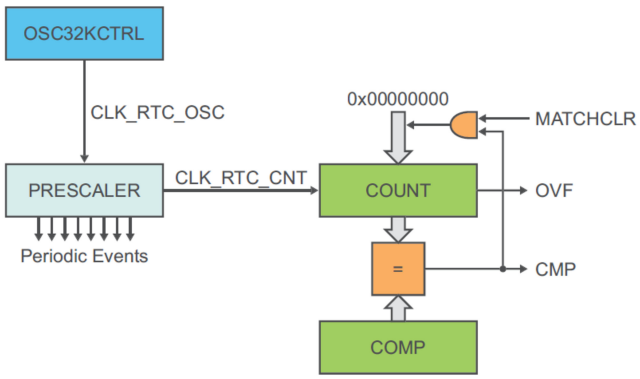
When the RTC Operating Mode bits in the Control A register are zero (CTRLA.MODE=00), the counter operates in 32-bit Counter mode. The block diagram of this mode is shown in the following figure, RTC Block Diagram (Mode 0 — 32-Bit Counter). When the RTC is enabled, the counter will increment on every 0-to-1 transition of `CLK_RTC_CNT`. The counter will increment until it reaches the top value of 0xFFFFFFFF, and then wrap to 0x00000000. This sets the Overflow Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF).

The RTC counter value can be read from or written to the Counter Value register (COUNT) in 32-bit format.

The counter value is continuously compared with the 32-bit Compare register (COMP0). When a compare match occurs, the Compare 0 Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMP0) is set on the next 0-to-1 transition of `CLK_RTC_CNT`.

If the Clear on Match bit in the Control A register (CTRLA.MATCHCLR) is '1', the counter is cleared on the next counter cycle when a compare match with COMP0 occurs. This allows the RTC to generate periodic interrupts or events with longer periods than the prescaler events. Note that when CTRLA.MATCHCLR is '1', INTFLAG.CMP0 and INTFLAG.OVF will both be set simultaneously on a compare match with COMP0.

Figure 3-1 RTC Block Diagram (Mode 0 - 32-bit Counter)



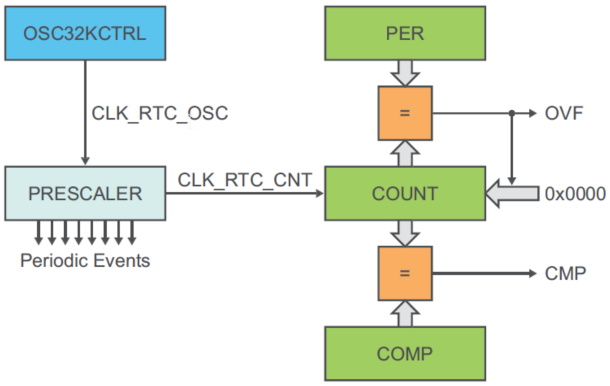
3.3.2. 16-bit Counter (Mode1)

When the RTC Operating Mode bits in the Control A register (CTRLA.MODE) are 1, the counter operates in 16-bit Counter mode as shown in the following figure, RTC Block Diagram (Mode 1 — 16-Bit Counter). When the RTC is enabled, the counter will increment on every 0-to-1 transition of `CLK_RTC_CNT`. In 16-bit Counter mode, the 16-bit Period register (PER) holds the maximum value of the counter. The counter will increment until it reaches the PER value, and then wrap to 0x0000. This sets the Overflow Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF).

The RTC counter value can be read from or written to the Counter Value register (COUNT) in 16-bit format.

The counter value is continuously compared with the 16-bit Compare registers (COMPn, n=0..1). When a compare match occurs, the Compare n Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn, n=0..1) is set on the next 0-to-1 transition of CLK_RTC_CNT.

Figure 3-2 RTC Block Diagram (Mode 1 - 16-bit Counter)



3.3.3. Clock/Calendar (Mode2)

When the RTC Operating Mode bit field in the Control A register (CTRLA.MODE) is '2', the counter operates in Clock/Calendar mode, as shown in the following figure, RTC Block Diagram (Mode 2 — Clock/Calendar). When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. The selected clock source and RTC prescaler must be configured to provide a 1Hz clock to the counter for correct operation in this mode.

The time and date can be read from or written to the Clock Value register (CLOCK) in a 32-bit time/date format. Time is represented as:

- Seconds
- Minutes
- Hours

Hours can be represented in either 12- or 24-hour format, selected by the Clock Representation bit in the Control A register (CTRLA.CLKREP). This bit can be changed only while the RTC is disabled.

Date is represented as:

- Day as the numeric day of the month (starting at 1)
- Month as the numeric month of the year (1 = January, 2 = February, etc.)
- Year as a value counting the offset from a reference value that must be defined in software

The date is automatically adjusted for leap years, assuming every year divisible by 4 is a leap year. Therefore, the reference value must be a leap year, e.g. 2000. The RTC will increment until it reaches the top value of 23:59:59 December 31 of year 63, and then wrap to 00:00:00 January 1 of year 0. This will set the Overflow Interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.OVF).

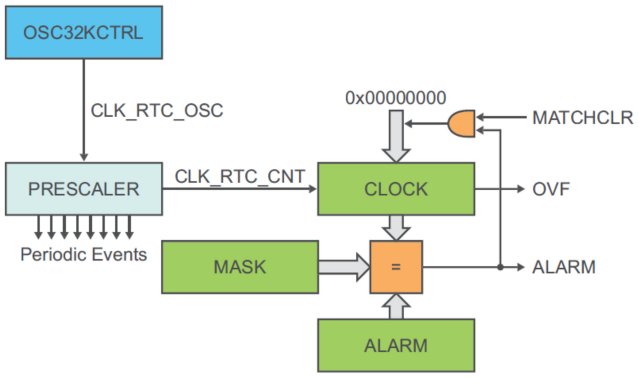
The clock value is continuously compared with the 32-bit Alarm register (ALARM0). When an alarm match occurs, the Alarm 0 Interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.ALARM0) is set on the next 0-to-1 transition of CLK_RTC_CNT.

A valid alarm match depends on the setting of the Alarm Mask Selection bits in the Alarm 0 Mask register (MASK0.SEL). These bits determine which time/date fields of the clock and alarm values are valid for comparison and which are ignored.

If the Clear on Match bit in the Control A register (CTRLA.MATCHCLR) is set, the counter is cleared on the next counter cycle when an alarm match with ALARM0 occurs. This allows the RTC to generate periodic interrupts or events with longer periods than it would be possible with the prescaler events only.

Note: When CTRLA.MATCHCLR is 1, INTFLAG.ALARM0 and INTFLAG.OVF will both be set simultaneously on an alarm match with ALARM0.

Figure 3-3 RTC Block Diagram (Mode 2 - Clock/Calendar)



3.4. Clocks

The RTC bus clock (CLK_RTC_APB) can be enabled and disabled in the Main Clock module MCLK, and the default state of CLK_RTC_APB can be found in the Peripheral Clock Masking section.

A 32KHz or 1KHz oscillator clock (CLK_RTC_OSC) is required to clock the RTC. This clock must be configured and enabled in the 32KHz oscillator controller (OSC32KCTRL) before using the RTC.

This oscillator clock is asynchronous to the bus clock (CLK_RTC_APB). Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains. Refer to Section Synchronization for further details.

3.5. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the Clock System Register Synchronization for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Changing the bit value under ongoing synchronization will not generate an error.

The following bits are synchronized when written:

- Software Reset bit in Control A register, CTRLA.SWRST
- Enable bit in Control A register, CTRLA.ENABLE
- Count Read Synchronization bit in Control A register (CTRLA.COUNTSYNC)
- Clock Read Synchronization bit in Control A register (CTRLA.COUNTSYNC)

The following registers are synchronized when written:

- Counter Value register, COUNT
- Clock Value register, CLOCK
- Counter Period register, PER
- Compare n Value registers, COMPn
- Alarm n Value registers, ALARMn
- Frequency Correction register, FREQCORR
- Alarm n Mask register, MASKn
- The General Purpose n registers (GPn)

The following registers are synchronized when read:

- The Counter Value register, COUNT, if the Counter Read Sync Enable bit in CTRLA (CTRLA.COUNTSYNC) is '1'
- The Clock Value register, CLOCK, if the Clock Read Sync Enable bit in CTRLA (CTRLA.CLOCKSYNC) is '1'
- The Timestamp Value register (TIMESTAMP)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read-synchronization is denoted by the "Read-Synchronized" property in the register description.

4. Tamper Detection

Tamper detection is the ability of a device to sense that an active attempt to compromise the device integrity or the data associated with the device is in progress; the detection of the threat may enable the device to initiate appropriate defensive actions.

4.1. Overview

The RTC provides up to five selectable polarity external inputs (INn) that can be used for tamper detection. The following table shows number of tamper input pins available for different series of SAM L22 devices. The RTC also supports an input event (TAMPEVT) for generating a tamper condition from within the event system. Refer to the following figure.

A single interrupt request (TAMPER) is available for all tamper sources. The polarity for each input is selected with the Tamper Level bits in the Tamper Control register (TAMPCTRL.TAMPLVLn). The tamper input event is enabled by the Tamper Input Event Enable in the Event Control register (EVCTRL.TAMPEVIE). The action of each input pin is configured by using the Input n Action bits in the Tamper Control register (TAMPCTRL.INnACT).

Figure 4-1 Tamper Detection Block Diagram

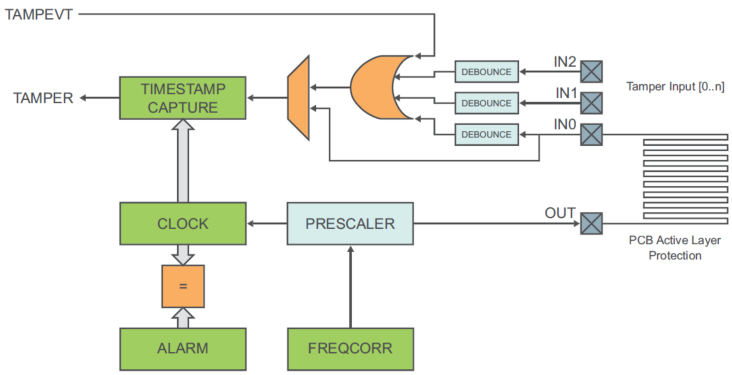


Table 4-1 Tamper Input Pins Configuration Summary

Device	SAM L22N	SAM L22J	SAM L22G
Tamper Input Pins	5	3	2

4.2. Modes of Operation

The tamper detection in this device can operate in one of the following modes.

• **Off**: Detection for INn is disabled

• **Wake**: A transition on INn matching TAMPCTRL.TAMPLVLn will be detected and the tamper interrupt flag (INTFLAG.TAMPER) will be set. The RTC value will not be captured in the TIMESTAMP register.

• **Capture**: A transition on INn matching TAMPCTRL.TAMPLVLn will be detected and the tamper interrupt flag (INTFLAG.TAMPER) will be set. The RTC value will be captured in the TIMESTAMP register.

• **Active Layer Protection**: A mismatch between INn and OUT will be detected and the tamper interrupt flag (INTFLAG.TAMPER) will be set. The RTC value will be captured in the TIMESTAMP register. See “Active Layer Protection on page 12” for more details.

To determine which tamper source caused a tamper event, the Tamper ID register (TAMPID) provides the detection status of each input pin and the input event. These bits remain active until cleared by software.

4.2.1. Active Layer Protection

The RTC provides a means of detecting broken traces on the PCB, also known as Active Layer Protection. Refer to Figure 4-1 Tamper Detection Block Diagram on page 11. In this mode an RTC output signal is routed over critical components on the board and fed back to one of the RTC inputs. The input and output signals are compared and a tamper condition is detected when they do not match.

Enabling active layer protection requires the following steps:

- Enable the RTC prescaler output by writing a one to the RTC Out bit in the Control B register (CTRLB.RTCOUT). The I/O pins must also be configured to correctly route the signal to the external pins.
- Select the frequency of the output signal by configuring the RTC Active Layer Frequency field in the Control B register (CTRLB.ACTF)

$$CLK\_RTC\_OUT = \frac{CLK\_RTC}{2^{CTRLB.ACTF+1}}$$

- Enable one of the tamper inputs (IN0n) in active layer mode by writing 3 to the corresponding Input Action field in the Tamper Control register (TAMPCTRL.IN0nACT). When active layer protection is enabled, the value of IN0n is sampled on the falling edge of CLK_RTC and compared to the expected value of OUT. Therefore up to one half of a CLK_RTC period is available for propagation delay through the trace.

4.3. Debouncing Action

Separate debouncers are embedded for each external input. The debouncer for each input is enabled/disabled with the Debounce Enable bits in the Tamper Control register (TAMPCTRL.DEBNCn). The debouncer configuration is fixed for all inputs as set by the Control B register (CTRLB).

The debouncing period duration is configurable using the Debounce Frequency field in the Control B register (CTRLB.DEBF). The period is set for all debouncers (i.e., the duration cannot be adjusted separately for each debouncer).

$$CLK\_RTC\_DEB = \frac{CLK\_RTC}{2^{CTRLB.DEBF}}$$

When TAMPCTRL.DEBNCn = 0, IN0n is detected asynchronously. See Figure 4-2 Edge Detection with Debouncer Disabled on page 13 for an example.

When TAMPCTRL.DEBNCn = 1, the detection time depends on whether the debouncer operates synchronously or asynchronously, and whether majority detection is enabled or not. Refer the following table for more details.

Synchronous versus asynchronous stability debouncing is configured by the Debounce Asynchronous Enable bit in the Control B register (CTRLB.DEBASYNC):

- Synchronous (CTRLB.DEBASYNC = 0): IN0n is synchronized in two CLK_RTC periods and then must remain stable for four CLK_RTC_DEB periods before a valid detection occurs. See Figure 4-3 Edge Detection with Synchronous Stability Debouncing on page 14 for an example.

- Asynchronous (CTRLB.DEBASYNC = 1): The first edge on IN0n is detected. Further detection is blanked until IN0n remains stable for four CLK_RTC_DEB periods. See Figure 4-4 Edge Detection with Asynchronous Stability Debouncing on page 14 for an example.

Majority debouncing is configured by the Debounce Majority Enable bit in the Control B register (CTRLB. CTRLB.DEBMAJ). IN0n must be valid for two out of three CLK_RTC_DEB periods. See Figure 4-5 Edge Detection with Majority Debouncing on page 15 for an example.

Table 4-2 Debouncer Configuration

TAMPCTRL.DEBNCn	CTRLB.DEBMAJ	CTRLB.DEBASYNC	Description
0	X	X	Detect edge on INn with no debouncing. Every edge detected is immediately triggered.
1	0	0	Detect edge on INn with synchronous stabilitydebouncing. Edge detected is only triggered when INn is stable for four consecutive CLK_RTC_DEB periods.
1	0	1	Detect edge on INn with asynchronous stabilitydebouncing. First detected edge is triggered immediately. All subsequent detected. Edges are ignored until INn is stable for four consecutive CLK_RTC_DEB periods.
1	1	X	Detect edge on INn with majority debouncing. Pin INn is sampled for three consecutive CLK_RTC_DEB periods. Signal level is determined by majority rule(LLL, LLH, LHL,HLL = “0” and LHH, HLH, HHL, HHH = “1”).

Figure 4-2 Edge Detection with Debouncer Disabled

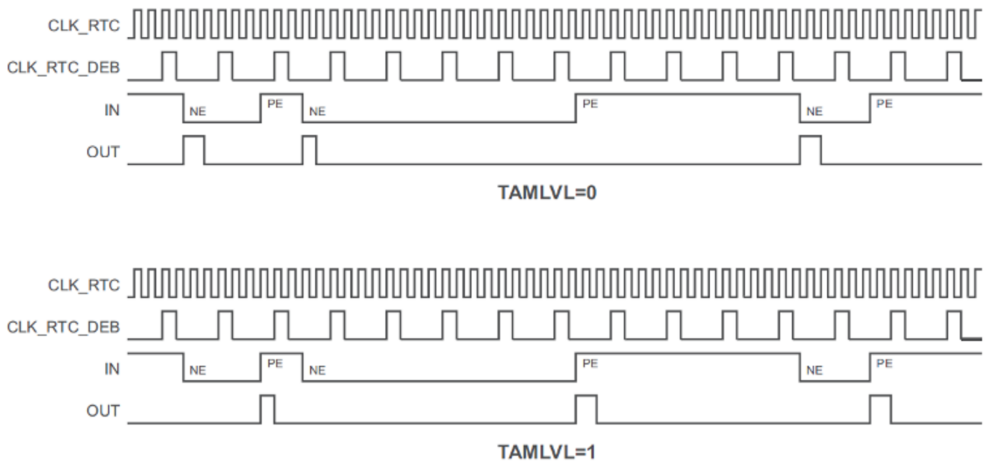


Figure 4-3 Edge Detection with Synchronous Stability Debouncing

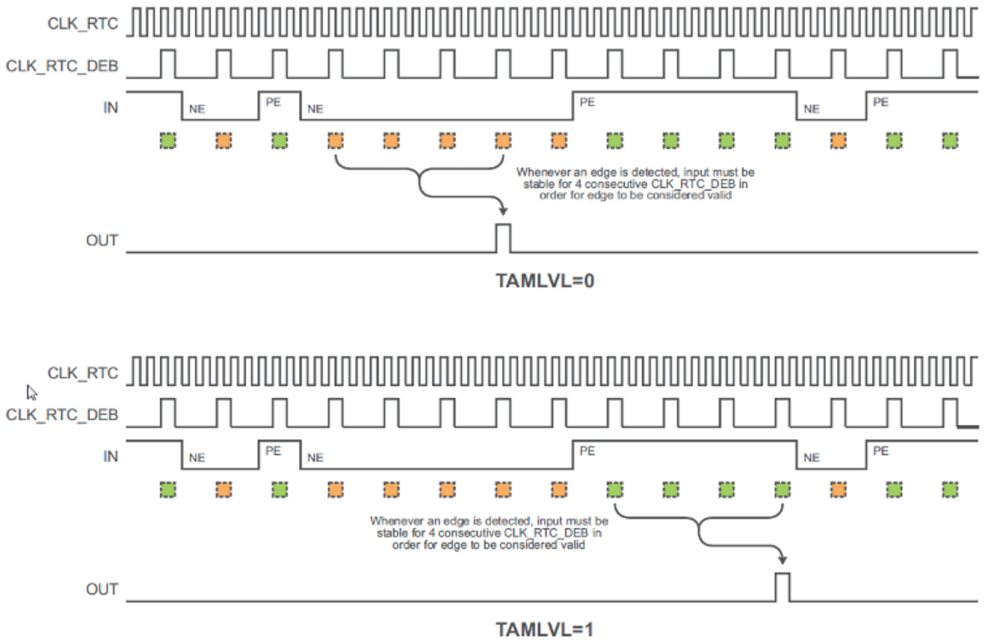


Figure 4-4 Edge Detection with Asynchronous Stability Debouncing

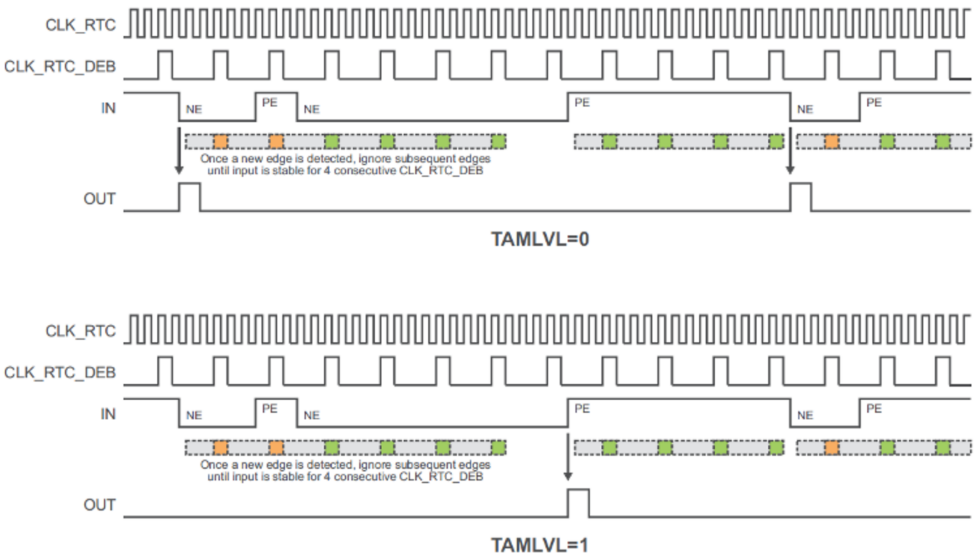
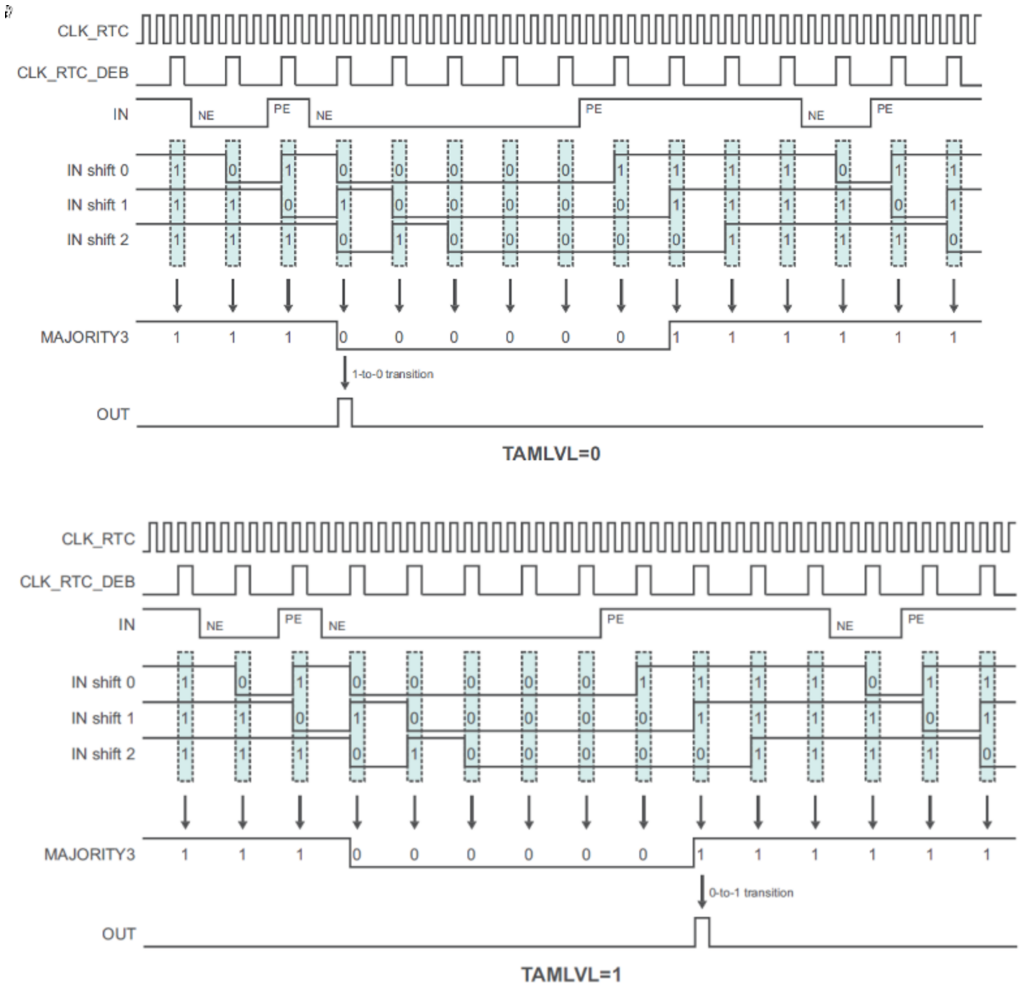


Figure 4-5 Edge Detection with Majority Debouncing



4.4. Timestamp

As part of tamper detection the RTC can capture the counter value (COUNT/CLOCK) into the 32-bit TIMESTAMP register.

Three CLK_RTC periods are required to detect the tampering condition and capture the value. The TIMESTAMP value can be read once the Tamper flag is set in the Interrupt Flag register (INTFLAG.TAMPER). If the DMA Enable bit in the Control B register is one (CTRLB.DMAEN), a DMA request will be triggered by the timestamp. In order to determine which tamper source caused a capture, the Tamper ID register (TAMPID) provides the detection status of each input pin and the input event. A DMA transfer can then read both TIMESTAMP and TAMPID in succession.

A new timestamp value cannot be captured until the Tamper flag is cleared, either by reading the timestamp or by writing a one to INTFLAG.TAMPER. If several tamper conditions occur in a short window before the flag is cleared, only the first timestamp may be logged. However, the detection of each tamper will still be recorded in TAMPID.

The Tamper Input Event (TAMPEVT) will always perform a timestamp capture. To capture on the external inputs (IN0n), the corresponding Input Action field in the Tamper Control register (TAMPCTRL.IN0nACT) must be written to one. If the input is set for wake functionality, they do not capture the timestamp; however, the Tamper flag and TAMPID will still be updated.

4.5. DMA Operation

The RTC generates the following DMA request:

• Tamper (TAMPER): The request is set on capture of the timestamp. The request is cleared when the Timestamp register is read.

If the CPU accesses the registers which are sources for the DMA request set/clear condition, the DMA request can be lost or the DMA transfer can be corrupted, if enabled.

4.6. Interrupts

The tamper detection of RTC has the following interrupt source:

• Tamper (TAMPER): Indicates detection of a valid signal on a tamper input pin or tamper event input.

The interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by setting the corresponding bit in the Interrupt Enable Set register (INTENSET=1), and disable by setting the corresponding bit in the Interrupt Enable Clear register (INTENCLR=1).

An interrupt request is generated when the interrupt flag is raised and the corresponding interrupt is enabled. The interrupt request remains active until either the interrupt flag is cleared, the interrupt is disabled, or the RTC is reset. See the description of the INTFLAG registers in the device datasheet for details on how to clear interrupt flags.

All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. Refer Nested Vector Interrupt Controller in the device datasheet for more details. The user must read the INTFLAG register to determine which interrupt condition is present.

Note: Interrupts must be globally enabled for interrupt requests to be generated. Refer Nested Vector Interrupt Controller in the device datasheet for more details.

4.7. Events

The tamper detection generate the following output event:

• Tamper (TAMPER): Generated on detection of valid signal on a tamper input pin or tamper event input.

Setting the Event Output bit in the Event Control Register (EVCTRL.TAMPEREO=1) enables the tamper output event. Writing a zero to this bit disables the tamper output event. Refer EVSYS -Event System in the device datasheet for details on configuring the event system.

The RTC can take the following actions on an input event:

• Tamper (TAMPEVT): Capture the RTC counter to the timestamp register.

Writing a one to an Event Input bit into the Event Control register (EVCTRL.TAMPEVEI) enables the tamper action on input event. Writing a zero to this bit disables the tamper action on input event.

4.8. Sleep Mode Operation

The tamper detection will continue to operate in any sleep mode where the source clock is active. The tamper interrupts can be used to wake up the device from a sleep mode. Tamper events can trigger other operations in the system without exiting the sleep mode.

An interrupt request will be generated after the wake-up if the Interrupt Controller is configured accordingly. Otherwise the CPU will wake up directly, without triggering any interrupt. In this case, the CPU will continue executing right from the first instruction that followed the entry into sleep.

Standby sleep mode is implemented in this application code. As RTC is capable of running in backup sleep mode, the user can also implement the backup sleep mode but with the following consideration. When backup sleep mode is enabled, the device goes to sleep and the RTC will be running internally. As tamper occurs the device wakes up and resets the controller. The tamper callback will not be executed in this condition. The user can handle this situation by writing a routine inside the main function, which initially checks for the cause of the backup reset. If the backup reset is because of tamper, then the user can execute required actions to be done during tamper detection.

5. Application Requirements

5.1. Hardware Requirements

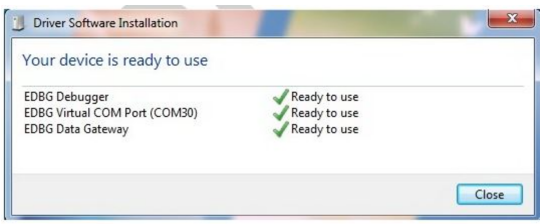
The application demonstration needs a SAM L22 Xplained Pro board as shown in the following figure.

Figure 5-1 SAM L22 Xplained Pro Board

There are two USB ports in the SAM L22 Xplained Pro board – DEBUG USB and TARGET USB. For debugging the target SAM L22 MCU using the Embedded debugger (EDBG), a Micro-B USB cable should be connected between a host PC running Atmel Studio and the DEBUG USB port on the SAM L22 Xplained Pro board.

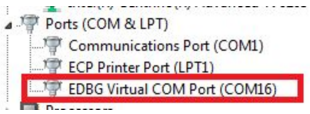
Once the kit is successfully connected, for the first time, the Windows Task bar will pop-up a message as shown in the following figure.

Figure 5-2 SAM L22 Xplained Pro Driver Installation



If the driver installation is proper, EDBG will be listed in the Device Manager as shown in the following figure

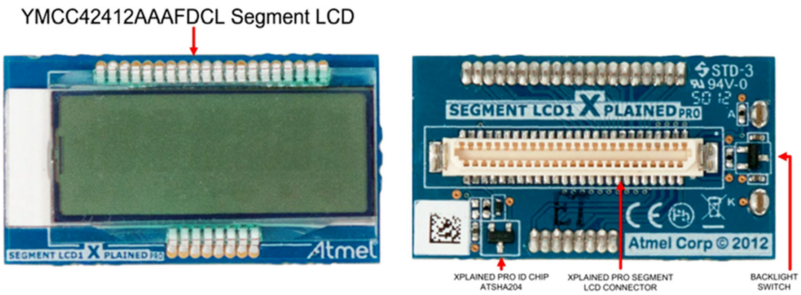
Figure 5-3 Successful EDBG Driver Installation



Application codes are tested in Atmel Studio 6.2 with ASF version 3.26.10 and later.

The SAM L22 Xplained Pro kit is the main board, which contains the MCU (ATSAML22N18A), used to run the example application and accessory board is Segment LCD1 Xplained Pro which contains segment LCD mounted on it, which is used to interface LCD to SAML22 Xplained Pro kit.

Figure 5-4 Top and bottom view of Segment LCD1 Xplained Pro



To interface Segment LCD (SLCD1 Xplained Pro) to SAM L22 Xplained Pro, connect SLCD1 Xplained Pro segment LCD connector (big, white connector on the bottom view of SLCD1 Xplained Pro) to segment LCD connector (EXT5) on the SAM L22 Xplained Pro kit.

5.2. Software Requirements

- Atmel Studio 6.2 or later
- ASF version 3.26.10 or later

Note: Make sure that the SAM L22 Part Pack for Atmel Studio is installed before using the example application.

5.3. Creating a Sample Project using SAM L22

Application codes are tested in Atmel Studio 6.2 with ASF version 3.26.10 and later.

GCC C ASF Board project from Atmel Studio is used for the implementation.

To create an ASF board project for SAM L22 Xplained Pro board, go to File menu -> New -> Project as shown in the figure below ("New Project in Atmel Studio") and select “GCC C ASF Board project” in the new project wizard as shown in the Figure 5-6 ASF Board Project on page 20.

Figure 5-5 New Project in Atmel Studio

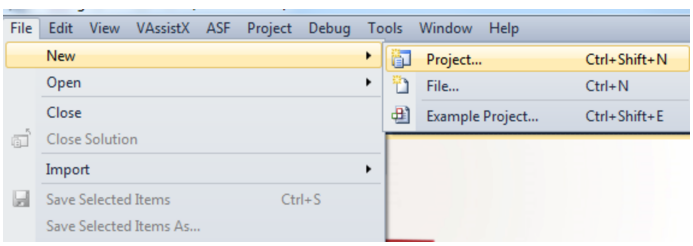
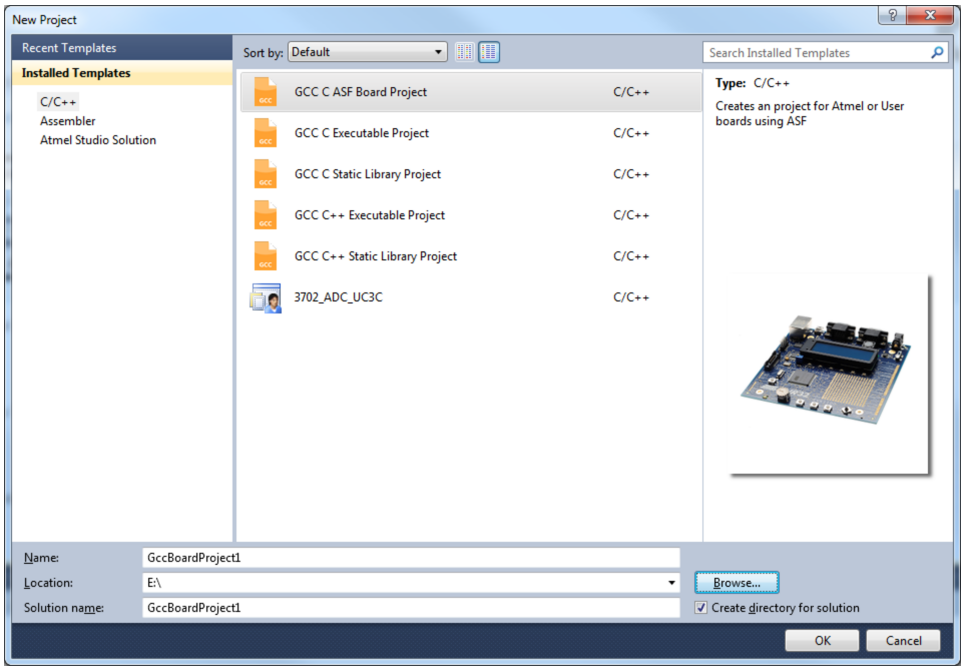
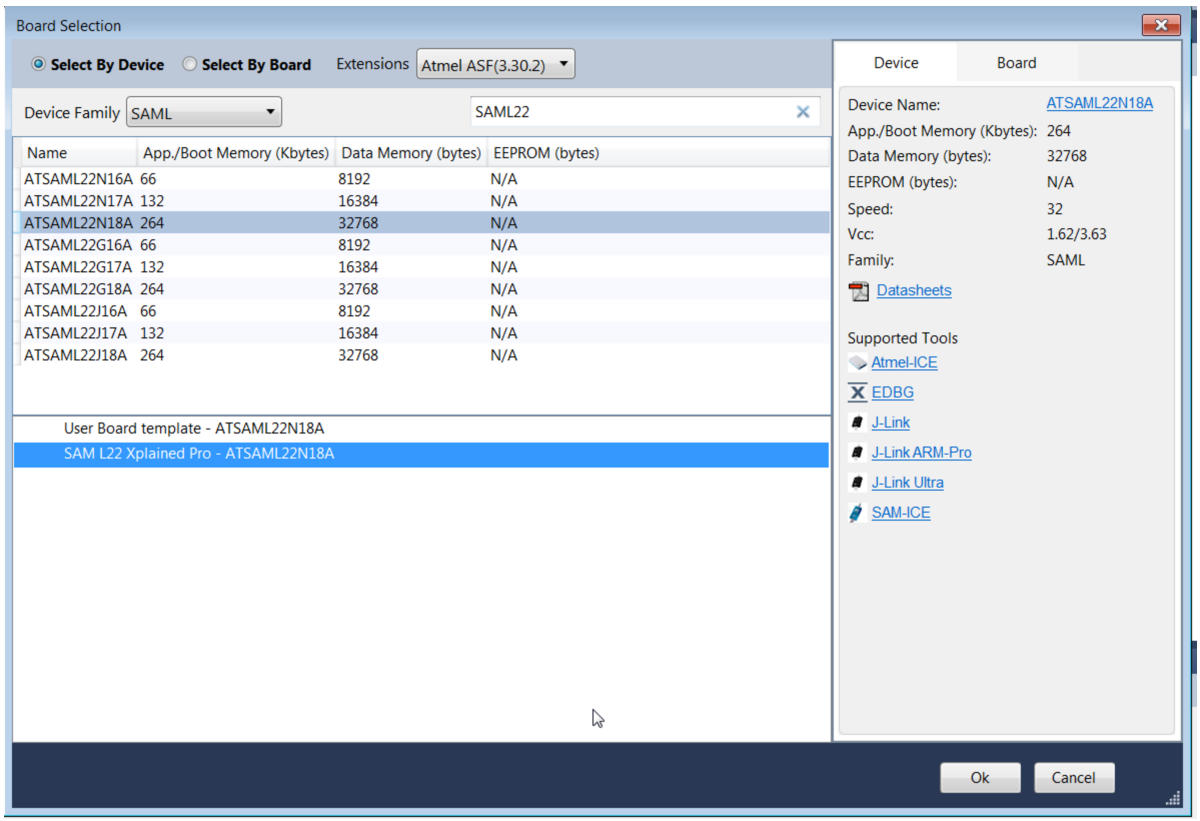


Figure 5-6 ASF Board Project



In the next window, select the device family as "SAM L", scroll down and select the device "ATSAML22N18A" and Board as "SAM L22 Xplained PRO - ATSAML22N18A" as shown in the following figure, and click on "OK" to create the new project.

Figure 5-7 Device and Board Selection

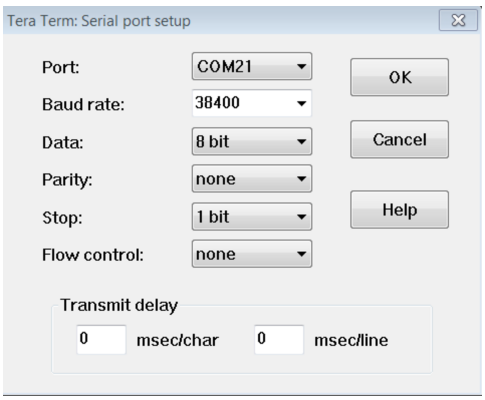


The new project by default has a minimal application that will turn ON or OFF the LED on SAM L22 Xplained Pro based on the state of the SW0 push button. Pressing the SW0 button will turn the LED ON and releasing the button will turn the LED OFF. To verify that the SAM L22 Xplained Pro is connected correctly this application can be run and checked whether it shows the expected output.

5.4. Terminal Window Setting

This application uses the terminal window of the PC to print various messages and status. The device communicates with the PC (terminal window) via EDBG. The following figure shows the serial port setting to be configured at the terminal window. Here Tera Term is used as the terminal window. The user can also use the "Terminal for Atmel Studio" available as an extension in Atmel Studio

Figure 5-8 Serial Port Settings



6. Application Demonstration

This chapter demonstrates the Tamper detection feature (an additional feature of RTC) in different modes and performs many other operations mentioned below.

6.1. Overview

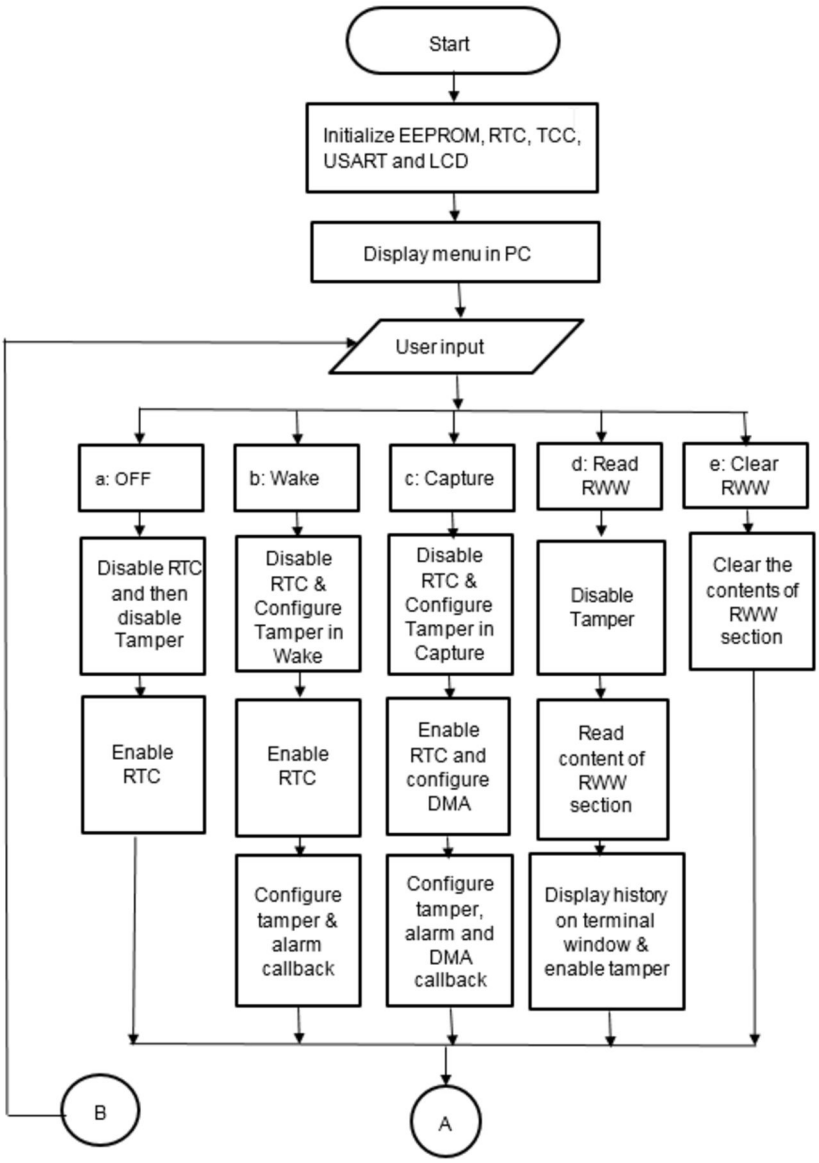
This application initializes the RTC module in clock/calendar mode (mode2) to keep track of the current time. Here, the user has the provision for setting the time in the clock register. Once the RTC initialization is done, the user will be requested for the mode in which the tamper detection has to occur. When the user select the mode of operation the tamper detection module will be configured according to the user input and RTC will be enabled. Now, the device is ready for tamper detection. When the tamper occurs, the application displays the error message on terminal window and SLCD, starts the timer for SLCD backlight toggling, stores the tamper history on RWW EEPROM, and gets ready for the next tamper. The detailed explanation of application is explained as follows.

1. Demonstrate various modes of tamper detection:
   - Tamper disable
   - Tamper enabled in wake mode
   - Tamper enable in capture mode
2. RTC in sleep mode and wake-up on Tamper.
3. Detect Tamper, starts the DMA transfer when the timestamp occurs and store the timestamp value in NVM of MCU.
4. Display error message on LCD and terminal window when tamper occurs.
5. Set alarm for 5 seconds.
6. Read the Tamper occurrence timestamp from NVM.
7. Erase the Tamper History.

6.2. Application Flowchart

The application note flow diagram is as follows:

Figure 6-1 Main Flowchart



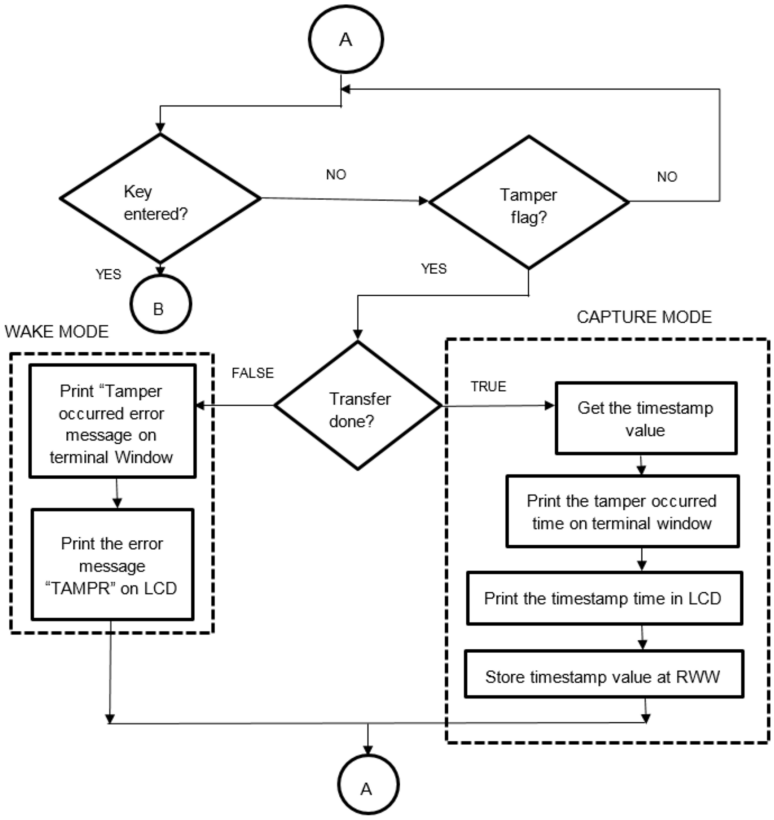


Figure 6-2 Tamper Callback Flowchart

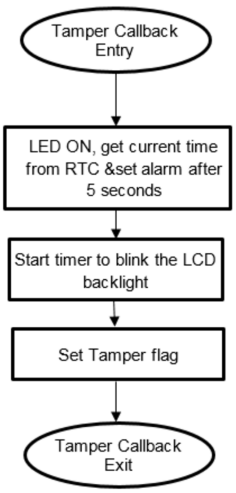
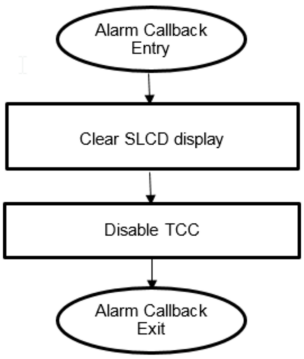


Figure 6-3 Alarm Callback Flowchart



6.3. Configuration

6.3.1. RTC Configuration

The application demonstration is explained as follows. This application demonstrates only three modes of tamper detection in SAM L22 Xplained Pro. The procedure for testing active layer protection is explained in section Active Layer Protection on page 12.

Before entering into the application demonstration, the basic condition in which the device SAM L22 is configured is explained as follows. If the user needs variations in the basic configuration files, which are to be changed, this is explained as follows.

1. The RTC is configured in **Clock/calendar mode (mode 2)** operating at 32.768kHz external oscillator. This mode is selected to keep track of the time and get the timestamp value easily.
2. The application allows user to set time; this includes date also. The application has current time and date set as 12:59:50 and 31/12/2012. The user can configure current time by changing the following macros in the header file `example.h`

```
#define YEAR    2012
#define MONTH    12
#define DAY      31
#define HOUR     11
#define MINUTE   59
#define SECOND   50
#define TIME_PM  false
```

Note: time.pm is not applicable if the RTC is configured for 24-hours clock mode

3. This application is configured in 12-hours clock mode. In order to configure for 24-hours clock mode, assign ‘true’ to the variable ‘config_rtc_calendar.clock_24h’ as shown below.

```
config_rtc_calendar.clock_24h = true;
```

Note: For 24-hours clock mode, the macro TIME_PM available in the header file example.h doesn't need to be defined and all the code statements related to the AM/PM settings can be removed.

4. The SAM L22 has five dedicated pins RTC/IN [0, 1…4] for tamper detection. User Switch SW0 available in the Xplained Pro board, connected to **RTC/IN [4]** pin, is used to provide tamper input.

The tamper input pin can be changed as per user's board requirements. The following macro should be changed corresponding to the tamper input pin:

```
#define TAMPER_INPUT_PIN   4
```

While configuring some other tamper input pin, ensure that the pull is enabled and the pin is configured as input.

This application is configured in such a way that the tamper is detected if there occurs a rising edge on the pin RTC/IN [4]. In order to configure for falling edge, change the level as shown below in both the functions, `configure_tamper_wake()` and `configure_tamper_capture()`.

```
config_rtc_tamper.in_cfg[4].level = RTC_TAMPER_LEVEL_FALLING;
```

5. As mentioned earlier, when the tamper is enabled in capture mode, the RTC takes the timestamp of the clock register. The clock register value is converted into time value and it is available in the global structure `struct tamper_time`.

6.3.2. Other Settings

USART Settings:

USART is initialized to communicate with the terminal window of the PC via the EDBG. The SERCOM USART is configured with the setting mentioned under the Terminal Window Setting on page 21, for communicating with the terminal window. If Baud rate change is required, the following macro in the header file `example.h` should be changed:

```
#define SET_BAUD_RATE  38400
```

SLCD Settings:

The SAM L22 Xplained Pro kit has the provision for Segment LCD Xplained Pro of Atmel. For more details regarding the SLCD configuration, refer the user guide which is specified in References on page 36.

On Tamper occurrence in WAKE mode, the error message **"TAMPR"** will be displayed on the LCD. If this error message needs to be changed, replace the error message in the following macro in the header file example.hp:

```
#define ERROR_STRING   "TAMPR"
```

Note: The SLCD in capable of displaying only five characters.

The TCC module is used for blinking the SLCD backlight while displaying the tamper error message. The Blinking time is controlled by the timer overflow value, which is defined as a macro as shown below. The user can change the overflow value if needed.

```
#define SLCD_BLINK_SPEED  2500
```

DMA Settings:

The RTC has the provision for enabling the DMA register to register transfer on Timestamp without CPU intervention. DMA transfers the 32-bit value of timestamp from TIMESTAMP register to a 32-bit buffer called `buffer_rtc_tamper` when the following functions are called.

```
void setup_transfer_descriptor(DmacDescriptor *descriptor)
void configure_dma_resource(struct dma_resource *resource)
dma_start_transfer_job(&example_resource)
```

EEPROM Configuration for Timestamp Storage

The firmware allows the user to store only 128 entries of tamper detection, which in turn occupies 898 bytes (896 (128*7) data memory and 2 bytes memory index) of RWW EEPROM memory. If the user needs to store more number of entries, change the maximum limit for writing the RWW EEPROM section, which is defined as a macro as shown below.

```
#define MAXIMUM_ENTRIES_STORED  128
```

In the above snippet, 128 is the number of time entries written to the EEPROM memory.

Note: The size of RWW EEPROM in SAML22N18A is 8KB out of which approximately 7KB is available for data storage. Hence user is requested to ensure that the memory index value is within the available memory.

The `erase_tamper_history()` function erases the full RWW EEPROM memory area, if the user intends to use the remaining RWW EEPROM memory section for some other data storage, care should be taken while using the `erase_tamper_history()` function. For more details regarding the RWW EEPROM emulator service, refer the application note "SAM EEPROM Emulator Service" mentioned under References on page 36.

Sleep Setting:

This application will enter into sleep mode once the configurations for tamper are done and wake up on tamper occurrence. The following statements will enable the STANDBY sleep mode and allow the device to go to sleep. This can be removed / modified as per user requirement.

```
system_set_sleepmode(SYSTEM_SLEEPMODE_STANDBY);
system_sleep();
```

6.4. Design Consideration

1. In SAM L22 Xplained Pro the GPIO PC27 is connected to both the LED and the SLCD's backlight. The LED is configured to be active Low and SLCD's backlight is configured to be active High. Turning ON the LED shall turn OFF the SLCD's backlight and vice versa.
2. There is no specific exit condition to return back to the normal state after the tamper occurrence. The user has to define the exit condition in their application as needed.
3. The TAMID flag and the INTFLAG are not cleared by the hardware and has to be taken care of by the firmware. Tamper condition will be detected even when the INTFLAG is set due to previous tamper, is not cleared. The flag has been cleared by the firmware in this application.
4. In the application demonstrated, the device enters into sleep mode after selecting the mode of tamper. The USART will be disabled in sleep mode. After entering a particular mode at least one tamper condition must occur before changing the mode to wake up the device. If tried to change the mode without waking up the device, the terminal window will not reflect the character entered.

5. The RTC time required has to be set by the user manually. Failing to do this shall load the default value of time into RTC clock register. The default time value in 12 hours clock mode:

```
time.year   = 2000;
time.month  = 1;
time.day    = 1;
time.hour   = 12;
time.minute = 00;
time.second = 00;
time.pm     = false;
```

6. In this application the RTC clock value will be lost when the device is powered off. On restarting the device, the user must ensure that the current time value is loaded to the device through the application code. The time value can be configured in the firmware as mentioned in the point 2, section RTC Configuration on page 25.
7. The NVM controller writes only 128 entries of tamper occurrence to the RWW EEPROM section. If it exceeds the limit the firmware will stop recording the timestamp value in the RWW EEPROM memory section until the user clears the memory.

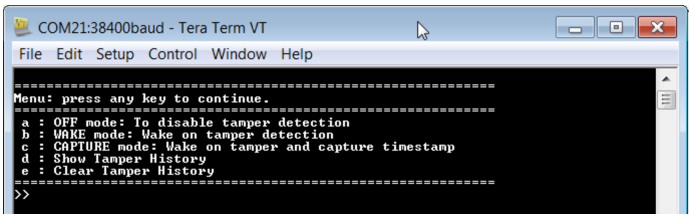
6.5. Test Procedure

The test procedure to be followed for demonstrating this application is explained as follows.

- Make sure that the SAM L22 Xplained Pro board is working fine by running the basic project as mentioned in section Creating a Sample Project using SAM L22 on page 19 of this application note
- Open the ASF project “TAMPER_DETECTION_EXAMPLE.atsln“
- This application includes the following drivers of ASF
  - General Board support (driver)
  - Sleep Manager (service)
  - Delay Routine (service) (cycle)
  - DMAC – Direct Memory Access Controller (driver)
  - PORT – GPIO pin function (driver)
  - RTC – Real Time Counter (driver) (calendar call back)
  - SERCOM – USART Serial Communication (driver) (polled)
  - SLCD – Segmented Liquid Crystal Display (driver) (polling)
  - Standard serial I/O (stdio) driver
  - TCC – Timer Counter for control application (driver) (call back)
  - Read While Write EEPROM emulator service (driver)
- Ensure that all the above drivers are present in the ASF wizard of the current project
- Compile the project and make sure that there are no errors and warnings
- Open the Terminal window “Tera-Term”, select the serial port in which the SAM L22 Xplained Pro is connected, and configure the serial port settings as mentioned under the section Terminal Window Setting on page 21
- Program the device and run the code
- The firmware initializes the following configurations
  - RWW EEPROM configuration - For enumerating RWW EEPROM section
  - RTC calendar configuration - For tracking the time easily from the clock register
  - TCC (timer) configuration - For blinking SLCD backlight
  - USART configuration - For communication with PC

- SLCD configuration - For display purpose
- Sleep configuration - For enabling the sleep modes
- DMA configuration - For initiating data transfer from register to register
- The display menu will be displayed on the terminal window as shown in the following figure

Figure 6-4 Display Menu



- Enter one of the following characters to enable the specific mode of tamper or to read the RWW EEPROM contents. The display menu consists of the following options:
  “a”: Disable tamper

  “b”: Enable tamper in wake mode

  “c”: Enable tamper in capture mode

  “d”: Read the contents of RWW memory location

  “e”: Clear the contents of RWW memory

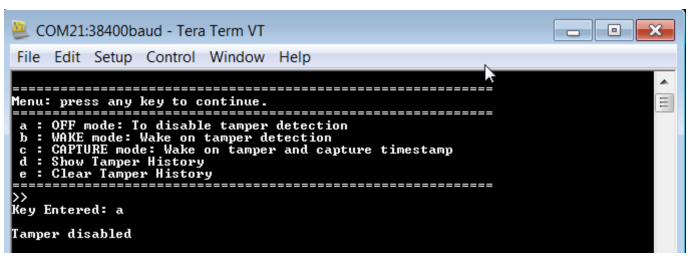
  When any of the above character is entered, it will be displayed on the terminal window.

  The operation under each and every mode is explained as follows.

6.5.1. OFF Mode

- When the key “a” is entered, the terminal window will appear as shown in the following figure
- Then, the device will disable the tamper operation and enable the RTC
- After enabling the RTC, the firmware will turn off the LCD backlight (turns on LED) and wait for the next input from the user
- Providing tamper by pressing SW0 shall not be detected by the device

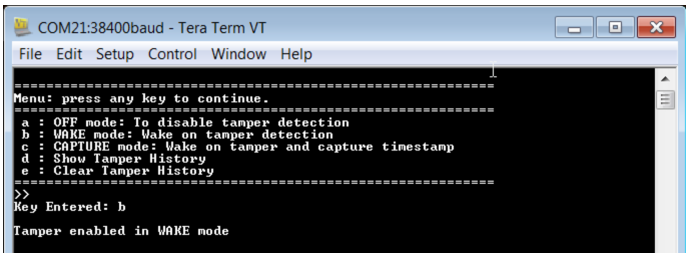
Figure 6-5 Terminal Window when Tamper Disabled



6.5.2. WAKE Mode

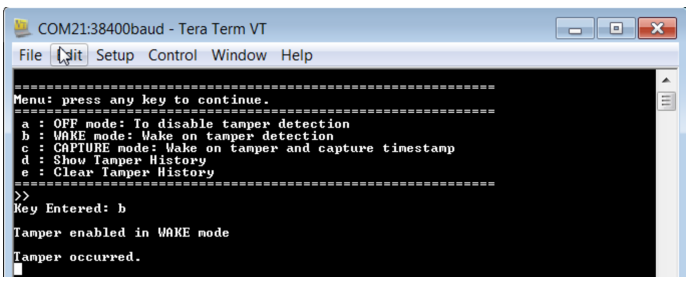
- When the key “b” is entered, the terminal window will appear as shown in the following figure

Figure 6-6 Terminal Window when Tamper Enabled in Wake Mode



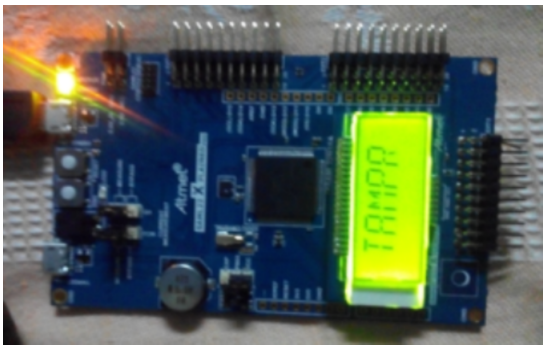
- The device will enable the tamper detection in wake mode (for more details, refer Modes of Operation on page 11) and enable the RTC
- After enabling the RTC, the clock starts counting at an interval of 1 second
- Set the mode flag and the previous mode flag as `SET_TO_WAKE`, to keep track of the current operating mode of tamper and previous mode of tamper
- The device goes to sleep. However, the RTC will still be running internally.
- Here, the Tamper condition can be generated by press and release of the user switch SW0 (which is connected to the tamper input pin)
- When a tamper is detected, the device wakes from sleep mode. The firmware will stay awake to detect future tamper inputs. The user also has the provision to change the input mode only after the first tamper condition is detected.
- Once the tamper occurs, the following action happens:
  - Displays the error message “Tamper occurred” to indicate that tamper has occurred as shown in the following figure

    Figure 6-7 Terminal Window when Tamper Callback Occured in Wake Mode

  - The device remains awake, detects tampers or key pressed by the user, and performs the actions accordingly
  - It turns OFF LED and turns ON LCD backlight
  - Get the current time from the clock register and sets alarm after five seconds
  - Set the tamper flag
  - Display the error message “TAMPR” on the LCD as shown in the following figure and enable the timer module with specified overflow value (user configurable) for blinking the SLCD backlight by toggling the port pin connected to it

Figure 6-8 LCD Screen when Tamper Occured in Wake Mode

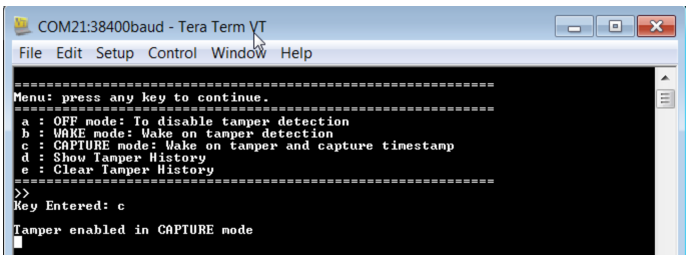


– Comes to alarm call back after five seconds where the firmware clears the LCD display and disables the TCC (timer module to stop LCD backlight blinking). Now, the device is ready for next the tamper condition or other operations.

6.5.3. CAPTURE Mode

- When the key “c” is entered, the terminal window will appear as shown in the following figure

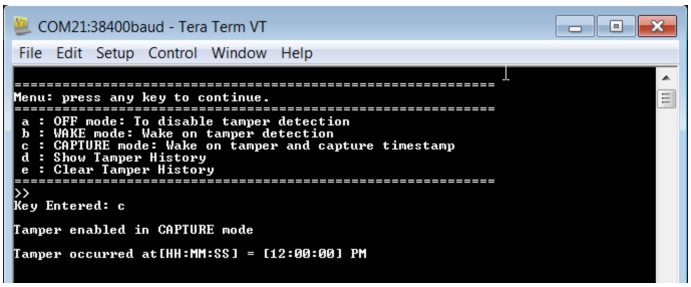
Figure 6-9 Terminal Window when Tamper Enabled in Capture Mode



- The device will enable tamper detection in capture mode (for more details, refer Modes of Operation on page 11) and enable the RTC
- After enabling the RTC, the firmware configures the DMA for getting the timestamp from the TIMESTAMP register without CPU intervention; the clock starts counting at an interval of 1 second
- Set the mode flag and the previous mode flag as SET_TO_CAPTURE, to keep track of the current operating mode of tamper and previous mode of tamper
- The device goes to sleep. However, the RTC will still be running internally.
- Now, the firmware will be waiting for the tamper input in the main function. In this condition the user also has the provision to change the input mode (only after the first tamper condition is detected).
- Here, the Tamper condition can be generated by pressing and leaving the user switch SW0 (which is connected to the tamper input pin)
- When the tamper occurs, the following action happens:
  – Get the current time from the clock register and set the alarm after five seconds and Set the tamper flag
  – Now the time value is available in the structure variable tamper_time. It prints the time on terminal window as shown in the following figure.
  – The format in which the time is printed on the terminal window, is given as follows

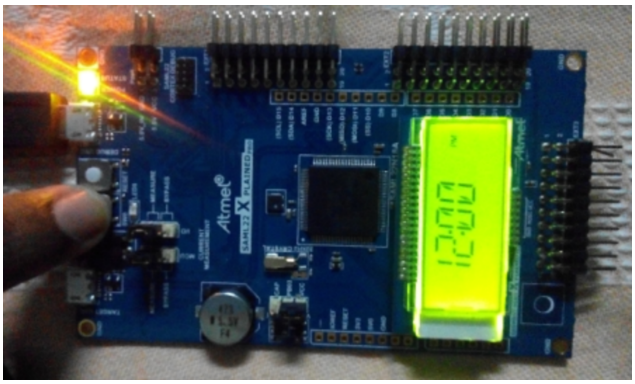
    [HH:MM:SS] = [12:00:00] PM

Figure 6-10 Terminal Window when Tamper Callback occured in Capture Mode



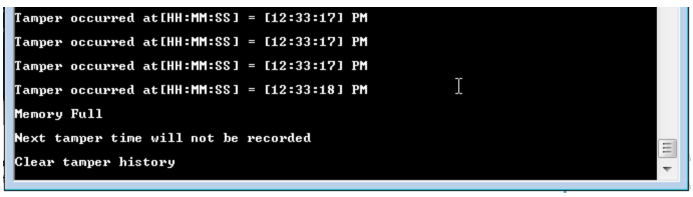
– Display the tamper occurred time on the LCD as shown in the following figure and enable the timer module with specified overflow value (user configurable) for blinking the SLCD backlight by toggling the port pin connected to it
Note: The LCD displays only Hours: Minutes as **12:00 PM** (for seconds, view the terminal window)

Figure 6-11 LCD Screen when Tamper Occurred in Capture Mode



– In addition to the above, it also writes the tamper Timestamp value to the RWW EEPROM section for keeping track of the tamper occurance
– The application is configured to store only 128 entries of time. When 128 entries are filled the firmware will display the error message shown in the following figure.

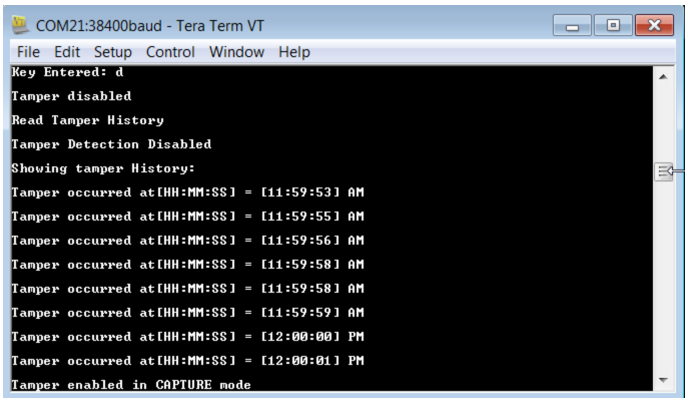
Figure 6-12 Memory Full Error Message



6.5.4. Read Tamper History

- When the key “d” is entered, the firmware turns off the SLCD, disables the tamer detection, and starts reading the RWW EEPROM section
- The terminal window will display the history of tamper occurred as shown in the following figure

Figure 6-13 Terminal Window when Tamper History Read



- When the tamper history is displayed, the firmware will reconfigure the tamper in the previous mode in which it was operating
- The tracking of previous mode is done by previous mode flag as mentioned in the sections WAKE Mode on page 29 and CAPTURE Mode on page 31
- As this memory is non-volatile, it will retain the value even after power reset. The first two bytes of the EEPROM emulated area contains the number of tampers occurred.
- The EEPROM emulated area of SAML22N18A is approx. 8KB. Here, the firmware is capable of writing 128 entries of tamper, that means it utilizes 898 bytes of RWW EEPROM emulated area for recording tamper occurrence. After writing the 128th entry, the firmware will provide a warning that this is the last memory of EEPROM area and clean the memory, as shown in the Figure 6-12 Memory Full Error Message on page 32. The next tamper entry will not be recorded by the firmware until the user erases the RWW EEPROM memory.
- However, the tamper will be detected and displayed in respective screens, but will not be stored in memory

  Note: As the tamper detection is disabled during the tamper history read operation the tamper conditions will not be detected until the tamper history read operation is completed.

6.5.5. Erase Tamper History

- When the input “e” is provided, the firmware will erase all the contents of the EEPROM emulated area

6.6. Testing Active Layer Protection

The active layer protection feature of the RTC can be tested as mentioned below. The main function of Active layer protection is detecting the broken traces on the PCB.

- Active layer protection involves two pins called RTC/IN[n] and RTC/OUT for detecting tamper condition
- Connect one end of the PCB to the RTC/IN [n] pin and the other end (which has continuity with IN[n] pin) to the RTC/OUT pin
- Then define a known pattern of waveform to the RTC/IN[n] pin and generate the same pattern at the RTC/OUT pin and send it across the PCB routing
- If there are any broken traces in the PCB or someone is attempting to remove the PCB, this will cause a mismatch in the wave pattern at IN and OUT pins, which will trigger a tamper interrupt

7. Use Case: Tamper Detection

The device SAM L22 is targeted for segment LCD and/or battery powered applications such as:

- Sport watches
- Personal healthcare devices
- Thermostat with user interface
- Access control panels and metering (gas, water, energy metering, and basic smart meter) applications

Among the above, energy meter is the typical application using this feature. Hence, the end user application regarding energy meter is discussed.

An energy meter is a device that measures the amount of electrical energy supplied to a residential or commercial building. The most common unit of measurement made by a meter is the kilowatt hour, which is equal to the amount of energy used by a load of one kilowatt in one hour.

A typical energy meter also requires a Real Time Clock (RTC) for tariff information. The RTC required for a metering application must be very accurate (< 5ppm) for Time of Day (TOD), which involves dividing the day, month, and year into tariff slots. Higher rates are applied at peak load periods and lower tariff rates at off-peak load periods. In addition to the tariff information the RTC in SAM L22 is also capable of detecting the tamper. SAM L22 has the tamper detection as an additional feature of RTC.

Meter tampering is an illegal method employed by consumers to gain entry, break into meter to deplete key functionalities, with the goal of reducing or completely eliminating the cost of energy usage. Traditional electricity meters have no ability to detect or deal with tampering because they only measure energy based on the voltage and current flowing between the inlet and outlet terminals. In such meters, tampering has become very easy and detection is harder.

All electronic meters have a microcontroller/microprocessor/ mixed-signal IC that performs energy measurement. These devices are very powerful and will directly contribute to the robustness of any meter. Energy is the instantaneous product of AC voltage and AC current averaged over time. Separate sensors for voltage and current will convert AC-mains voltage and current to a reduced and acceptable input for analog-to-digital converters (ADCs) (called as analog frontend) processing in the digital domain.

There are various tampering methods, which accumulates under following four categories:

1. Physical tampering.
2. Magnetic interference.
3. Removing wires.
4. Phase neutral Interchange.

7.1. Physical Tampering

Physical tampering is defined as the physical activities performed to prevent the energy meter measuring. This includes the following methods:

1. Top cover open.
2. Bottom cover open.
3. Inserting metal objects to prevent measurement, etc.

This type of physical tampering can be prevented by using the tamper detection feature of SAM L22.

It can be implemented as follows.

Connect one of the tamper input pins to a switch connected to the case cover of the energy meter. If someone attempt to open the case cover the switch will be pressed (edge detection happens), which in turn gives a tamper indication. Hence, if the user attempts to open the case cover of the energy meter tamper will be detected.

Inserting metal object can only be done when the case cover is opened.

7.2. Magnetic Interference

Meters use magnetic material in voltage and current measurement circuits and thus are affected by abnormal external magnetic influences, which in turn affect proper functioning of the meter.

This type of tampering can be prevented by placing magnetic sensors near the energy meter and configure the firmware in such a way that if any external magnetic field is detected, generate a rising edge on the tamper input pin. Hence, if the consumer brings an external magnetic field, tamper condition will be detected.

7.3. Removing Wires

The tampering action like missing neutral and missing Potential are caused because of the phase or neutral wire removal. There will be Analog frontend units which tracks the phase and neutral signals continuously. When someone is attempting to remove the phase or neutral line, the Analog Frontend will trigger an interrupt to the microcontroller connected to it. If the Analog frontend module triggers a rising/ falling edge on the RTC/IN pin of SAM L22, the tamper condition will be detected.

7.4. Phase Neutral Interchange

Some of the consumers will interchange the phase and the neutral wire to stop the energy meter reading the energy consumption. This type of tampering can be sensed by using Analog frontend or Active layer protection. As the consumer is attempting to change the phase and neutral wire, there will occur some mismatch in the PCB track of the energy meter. Hence, this can be detected as tamper action by the SAM L22 using the Active layer protection feature.

8. References

SAM L22 Device Datasheet

Web page: http://www.atmel.com/products/microcontrollers/arm/sam-l.aspx?tab=documents

Document: Atmel SAM L22 Datasheet.pdf

SAM L22 Xplained Pro User Guide and Schematics

Web Page: http://www.atmel.com/tools/ATSAML22-XPRO.aspx?tab=documents

Atmel Segment LCD1 Xplained Pro User Guide

Web link: http://www.atmel.com/Images/Atmel-42076-Segment-LCD1-Xplained-Pro_User-Guide.pdf

SAM EEPROM Emulator Service (EEPROM)

Web link: http://www.atmel.com/images/atmel-42125-sam-d20-eeprom-emulator-service-eeprom_application-note_at03265.pdf

9. Revision History

Doc Rev.	Date	Comments
42495A	08/2015	Initial document release.

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**