SmartFusion cSoC: System Power Optimization Using Low Power Modes

Table of Contents

Introduction

The SmartFusion® customizable system-on-chip (cSoC) FPGA devices integrate FPGA technology with hardened ARM® Cortex™-M3 processor based microcontroller subsystem (MSS) and programmable high-performance analog blocks built on a low power flash semiconductor process. The MSS consists of hardened blocks, such as a 100 MHz ARM Cortex-M3 processor, peripheral DMA (PDMA), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), embedded FlashROM (eFROM), external memory controller (EMC), watchdog timer, the Philips Inter-Integrated Circuit (I^2C), serial peripheral interface (SPI), 10/100 Ethernet controller, real-time counter (RTC), GPIO block, fabric interface controller (FIC), in-application programming (IAP), and system registers. The programmable analog block contains the analog compute engine (ACE) and analog front-end (AFE) consisting of ADCs, DACs, active bipolar prescalers (ABPS), comparators, current monitors, and temperature monitors.

Power consumption is one of the most significant concerns apart from the design complexity in FPGA-based design today. The concept of achieving the system-on-chip (SoC) functionality in an FPGA cuts down the design complexity, yet power consumption is quickly becoming the most critical issue for the design community. The SmartFusion cSoC reduces the design complexity and exhibits low power characteristics similar to that of an ASIC, making it an ideal choice for power sensitive applications. In the SmartFusion cSoC devices there is no power-on current surge and no high current transition, both of which are common with other types of FPGAs. The SmartFusion cSoC devices also have low dynamic power consumption, provide various methods to control power consumption, and support low power Standby mode, offering further power savings.

This application note explains the various methods for controlling the power consumption and low power modes supported by the SmartFusion FPGAs. Refer to the 'SmartFusion DC and Switching Characteristics' chapter in the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet for information on different components contributing to the power consumption in the SmartFusion cSoC devices.

The design example provided with this application note demonstrates the implementation of various SmartFusion-based system power optimization methods.

FPGA Power Consumption Overview

The power profile of FPGA devices varies depending on the underlying technology used to create the programming element. The power profiles of Microsemi's flash-based nonvolatile FPGAs more closely resemble those of ASICs and ASSPs than SRAM-based FPGAs.

There are four basic power components that contribute to the total system power. They are power-up or in-rush power, configuration power, static power, and dynamic power.

In-Rush Power

During the start-up stage, the device requires a substantial amount of logic array current for a specific duration in order to ramp up VCC to the correct voltage. This initial high current is called in-rush current, often seen as a power spike. The duration of VCC ramp-up depends on the amount of current available from the power supply. When the VCC supply reaches 90 percent of its value, this initial high current is not required any longer.

Configuration Power

The configuration power is the power required to configure the device. This phase is typically required only for SRAM-based FPGAs where the configuration data for the device is stored in an external nonvolatile memory such as an EPROM or flash device. During the configuration, the device draws the current to read the routing and look-up table (LUT) configurations from memory into the device. During the initialization, the device draws the current to reset the registers, enable I/O pins, and enter operating mode.

Static Power

The static power is proportional to the static current that flows when the device is powered up, configured, and is doing nothing, that is, there is no activity at I/Os and clock inputs.

Dynamic Power

Dynamic power is very sensitive to switched capacitances and primarily routing capacitances. Also, it is a function of operating frequency and switching of capacitive loads such as I/Os, internal gates, registers, clock lines, buffers, and internal memory accesses.

The SmartFusion cSoC devices are true flash-based FPGAs that offer low power due to their architecture and interconnect features. The Flash-based FPGAs have only two major power components—static and dynamic; whereas SRAM-based FPGAs have two additional power components—the in-rush power and configuration power.

Total Power Consumption of SRAM-based FPGAs = $P_{Static} + P_{Dynamic} + P_{Inrush} + P_{Config}$

Total Power Consumption of SmartFusion FPGAs = $P_{Static} + P_{Dynamic}$

The total power consumption of the SmartFusion cSoC devices can be further reduced by entering into low power modes such as standby mode and time keeping mode.

The following sections describe the SmartFusion power modes and how to optimize the system power in the SmartFusion-based systems.

SmartFusion Power Modes

The major contributors to the power consumption in the SmartFusion FPGAs are MSS peripherals, analog peripherals, FPGA tiles, internal clock frequencies, and the number of I/O pins and their standards used in the design. The SmartFusion cSoC devices provide a various methods for controlling the power consumption of the device. These methods include the following:

- The sleep mode feature of the Cortex-M3 processor is used to conserve the power. When the Cortex-M3 processor is in sleep mode, it effectively uses the clock gating for the entire processor with the exception of interrupt controller sections. Sleep mode is implemented through the wait for interrupt (WFI) and wait for event (WFE) instructions.
- Firmware controlled clock frequency switching by configuring the *MSS_CCC* and no-glitch *MUX (NGMUX)*.
- Firmware controlled reset of MSS peripherals that allows these peripherals to be held in reset mode if they are not in use.
- Firmware controlled power-down or power-up of programmable analog blocks.
- Power-on control of the 1.5 V internal voltage regulator by the RTC or external signal (PU_N).
- Power-off control of the 1.5 V internal voltage regulator by firmware or FPGA logic.

You can implement the following power modes based on the above mentioned methods.

- SoC mode
- Standby mode
- Time keeping mode

SoC Mode

This is the normal mode of operation where both the MSS and the FPGA fabric are operational with the required peripherals.

Standby Mode

This mode is for the applications that intend to put the device into a low power state. However, the device should be ready to respond to an interrupt that is sourced from the MSS, the FPGA fabric, or the AFE.

In Standby mode, the SmartFusion cSoC device is active but the Cortex-M3 processor is in sleep mode, eNVM is held in reset, the main crystal oscillator is disabled and the system is running at a lower frequency clock than what is used to clock the MSS. For example, the low power 32 KHz crystal oscillator can be used to clock the MSS. The peripherals not being used in Standby mode are put into a low power state or power off.

In this mode, the embedded SRAM, I/Os, and registers retain their values so that the device can enter and exit this mode without any penalty.

Time Keeping Mode

In Time keeping mode, the only supply to the SmartFusion cSoC device, which is enabled in the VBAT rail. You can transit to Time keeping mode by having all the supplies turned off other than VBAT. Typically, a lithium ion coin cell is connected to VBAT. The RTC in this mode keeps track of time while the lithium coin cell is still charged. When the VCC33A and the VCCIO supplies are turned on to the SmartFusion cSoC device, the device wakes up on an *RTC_MATCH* or assertion of *PU_N*. This mode is used to keep the track of elapsed time in the event of a power outage or in portable devices when you swap out the main battery. Table 1 summarizes the status of the SmartFusion resources in various low power modes.

Table 1 • **SmartFusion Power Modes**

SmartFusion Resource	SoC	Standby	Time Keeping
MSS			
Cortex-M3 Processor	Active	Sleeping	Off
Peripherals	Active	Active	Off
Memory Controllers			
eSRAM	Active	Active	Off
eNVM	Active	Reset	Off
RTC	Active	Active	Active
Oscillators			
LPXTAL	Active	Enable	Enable
MAINXTAL	Active	Disable	Off
FPGA Fabric	Active	Active	Off
Programmable Analog			
Analog Front-End	Active	Active	Off

Note:

- On-chip RC oscillator is always ON.
- Active means that proper voltage and clocks are applied.
- Reset means that peripheral is held in reset.
- Enable/Disable means that block is powered up but enabled/disabled through firmware control.
- Off means that voltage supply is disconnected.

SmartFusion Power System

The SmartFusion power system consists of separate analog, digital, and I/O bank supply pins, as shown in Figure 2 on page 8. It also includes an internal voltage regulator that can provide a 1.5 V regulated supply for the MSS and FPGA fabric. The SmartFusion cSoC has two voltage domains: 1.5 V domain and 3.3 V domain.

The SmartFusion analog circuitry requires clean analog voltages and the remaining digital circuitry requires normal DC voltages. The SmartFusion MSS, fabric and some of the programmable analog, such as the ADCs circuitry, operates on 1.5 V and rest of the analog operates on 3.3 V.

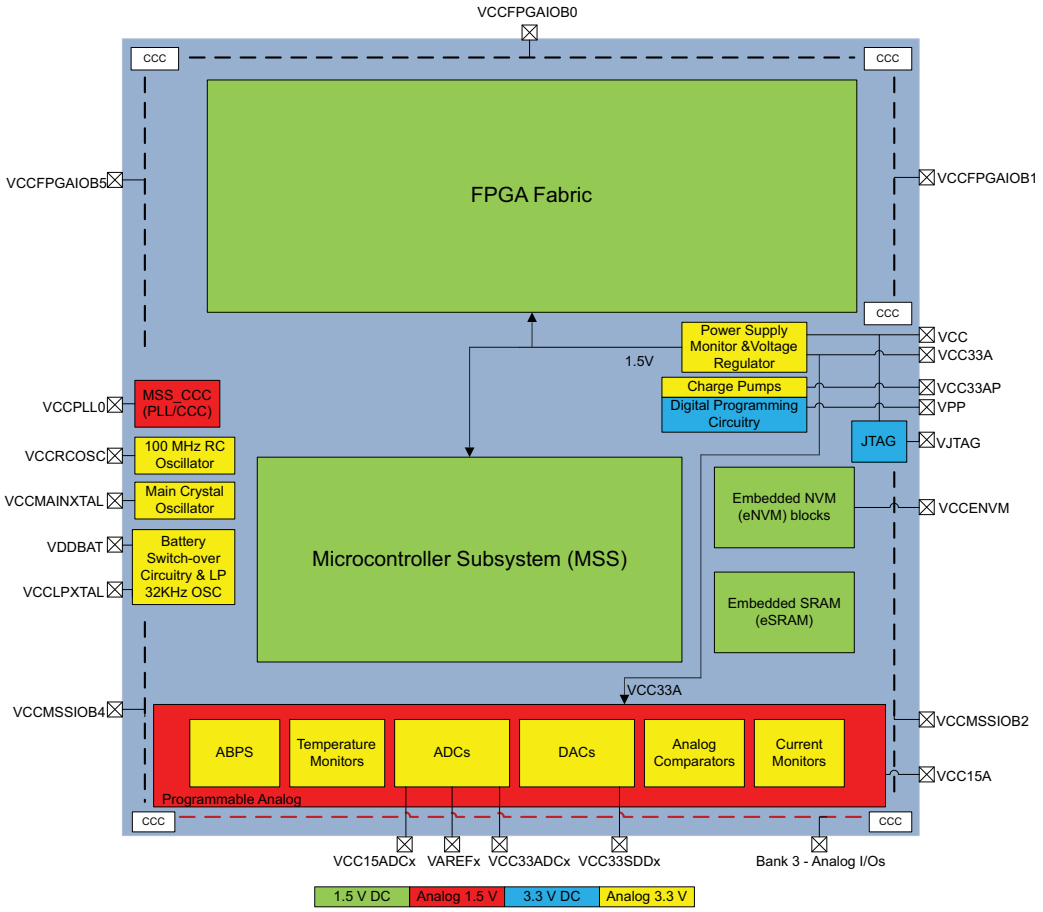

Figure 1 • **SmartFusion Power System**

Table 2 on page 6 shows the power supply configuration in three power modes. Refer to the 'SmartFusion DC and Switching Characteristics' chapter in the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet for recommended voltage ranges specific to each power supply.

Table 2 • **SmartFusion Power Supplies Configuration**

Power Supplies	SoC Mode	Standby Mode	Time Keeping Mode
VCCFPGAIOBx/VCCMSSIOBx	On[1]	On	0 V
VCC33A/VCC33ADCx/VCC33AP/VCC33SDDx/ VCCMAINXTAL/VCCLPXTAL	3.3 V	3.3 V	0 V
VCC/VCCENVM	1.5 V	1.5 V	0 V
VDDBAT	N/A	N/A	3.3 V
VCCPLLx/VCC15A/VCC15ADCx	1.5 V	1.5 V	0 V
VCCRCOSC	3.3 V	3.3 V	0 V
VJTAG[2]	N/A	N/A	0 V
VPP[3]	N/A	N/A	0 V
VAREFx[4]	N/A	N/A	0 V

Notes:

1. *On indicates proper voltage is applied to I/O banks.*
2. *If the JTAG interface is neither used nor planned for use, the VJTAG pin together with the TRSTB pin is tied to GND. VCC is required to be powered for JTAG operation; VJTAG alone is insufficient. The VJTAG supply runs at any voltage from 1.5 V to 3.3 V (nominal).*
3. *The VPP can be left floating during normal operation (not Programming mode). For programming, VPP should be in the 3.3 V ± 5% range.*
4. *The external voltage reference for ADC. The SmartFusion cSoC devices can be configured to generate a 2.56 V internal reference that can be used by the ADC. When VAREF0 is internally generated, a bypass capacitor must be connected from this pin to ground. The valid range of values that can be supplied to the ADC is 1.0 V to 3.3 V.*

Design Description

The example application uses *MSS_UART_0*, *MSS_CCC*, Cortex-M3 processor, RTC, on-chip RC oscillator, eSRAM, and eNVM memory controllers. The MSS is configured to run at a clock frequency of 100 MHz.

Most of the SmartFusion MSS functional peripherals can be put in a low power state by using the SmartDesign MSS configurator graphical interface. You can overwrite this configuration using the Cortex-M3 firmware control.

The example software shows the real-time clock application implementation and provides the UART-based user interface. The UART-based user interface can be used to enable or disable the SmartFusion cSoC peripherals and to switch between different SmartFusion cSoC low power modes. For example, unused MSS peripherals and analog blocks can be put into a low power state for power saving. You can observe the change in the currents sourced from the 3.3 V and 1.5 V supplies by enabling or disabling the SmartFusion cSoC peripherals or by entering into the low power states.

In this application, you can put the system in a low power state by entering into Standby mode using UART-based communication. In Standby mode, the Cortex-M3 processor is in Sleep mode and the system is clocked at 32 KHz frequency from the 32 KHz low power crystal oscillator for power saving. However, the eSRAM and eNVM block should retain their values so the device can exit in Standby mode without any penalty.

In Standby mode, an interrupt that is sourced from the MSS peripherals, the fabric, or the analog front-end can switch the system into normal operation. The RTC match from the real-time counter (RTC_MATCH = 1) can wake up the Cortex-M3 processor and other peripherals and the transition to SoC mode begins. The peripherals that are not used in the Standby mode are put into a low power state or powered off.

With reference to the example application, the *MSS_UART_0* is not required during the Standby mode. Therefore, *MSS_UART_0* can be held in reset before the processor goes into Sleep mode. In some applications where the ACE and AFE are used for voltage, current, or temperature monitoring, you can keep them running continuously to monitor the analog inputs and put the other unused peripherals into reset state. Whenever analog inputs exceed the threshold values, the AFE generates an interrupt that switches the system into SoC mode for normal operation. In the same way, you can implement the system controller in the fabric. The system controller remains active at all times to manage the system critical events (system alerts or alarms, user input such as switch presses, etc.) while the high power consuming blocks in the device such as the processor, MSS peripherals, and analog blocks are placed in Sleep or Shutdown mode. An interrupt generated from the fabric logic can switch the system to normal operation.

The following sections explain the implementation of the power modes discussed above and methods of optimizing the power consumption.

System Power Optimization

You can configure and control the major functional blocks according to design requirements to reduce the power consumption, for example, PLLs/CCCs, the Cortex-M3 processor, MSS memory controllers, MSS peripherals, AFE, power supply monitor (PSM), and voltage regulator (VR).

To save the power, unused functional blocks are put into reset (low power state) or power off state under firmware control by the Cortex-M3 processor using control registers. The control registers for the SmartFusion cSoC functional blocks are located in the system registers address space starting at 0xE0042000 and extend to address 0xE0042FFF in the Cortex-M3 processor memory map. The peripherals that are not used are put into low power state by asserting their individual resets in the *SOFT_RST_CR* register. Analog blocks that are not used can be powered off using registers *ANA_COMM_CTRL* and *ADCx_MISC_CTRL* where x equals 0, 1, or 2, indicating which ADC to power down. Refer to the *SmartFusion Microcontroller Subsystem (MSS) User's Guide* and *SmartFusion Programmable Analog User's Guide* for available system control registers and corresponding bit definitions.

If the Cortex-M3 processor wake-up time is not a restriction then you can switch the MSS clock input from the on-chip RC oscillator or main crystal oscillator to a slow clock source such as low power 32 kHz oscillator to save the power. The main crystal oscillator can be disabled by the Cortex-M3 processor via the *MSS_CCC_MUX_CR*, bit 29 MAINOSCEN. When the main crystal oscillator is not in use, the MAINXIN and MAINXOUT pins can be left floating. It is important to note that the on-chip RC oscillator is always turned on. For safe clock switching methods, refer to the PLLs, Clock Conditioning Circuitry, and On-chip Crystal Oscillators chapter of the *SmartFusion Microcontroller Subsystem (MSS) User's Guide*.

The total power consumption in the SmartFusion cSoC device comes from 3.3 V and 1.5 V power rails. The MSS block and fabric blocks are powered from the 1.5 V power rail and all the analog circuitry and eNVM blocks are powered from the 3.3 V power rail.

$$\text{Total power consumption} = (1.5 \times I_{1.5\,V} + 3.3 \times I_{3.3\,V}) \text{ Watts}$$

Here $I_{1.5\,V}$ and $I_{3.3\,V}$ are currents drawn from 1.5 V and 3.3 V sources.

On the SmartFusion Development Kit Board, the total power consumed by the system can be measured by measuring the current drawn from 3.3 V supply and 1.5 V internal or external regulator using JP2 and JP6 jumpers respectively.

The SmartFusion cSoC devices have an internal voltage regulator which is powered from the 3.3 V supply and produces 1.5 V regulated output, which can be used as a supply to the MSS and FPGA fabric. If an existing external 1.5 V voltage regulator rail is already available in the system, you can use that to power the 1.5 V rail and switch off the internal voltage regulator to save the power. The PSM provides reference voltages for the analog-to-digital converter (ADC) and the eNVM.

The RTC runs on a 3.3 V analog supply. The RTC is an *APB_0* slave that provides a counter as well as a MATCH output signal that can be used to interrupt the Cortex-M3 processor and power up the on-chip 1.5 V voltage regulator. An on-chip 32 kHz oscillator provides the clock source for the RTC. Apart from the turn-on instruction from the RTC, the 1.5 V internal voltage regulator can be turned on by a low signal on the external pin *PU_N* or by JTAG being active; that is TRSTB = 1 or the PUPO signal from the VR Initialization (VR Init) block.

The SmartFusion cSoC devices have an input for an external battery source that allows both the RTC and the low power crystal oscillator to function when the 3.3 V rail supply has been removed. The battery switching circuit automatically powers the RTC and the low power crystal oscillator from the battery whenever the battery voltage is approximately 0.4 V or more than the VCCLPXTL pin voltage.

Hardware Implementation

This section explains the hardware configuration required to run the example software application. Create a new project using Microsemi Libero® System-on-Chip (SoC) and configure the MSS peripherals using the SmartDesign MSS configurator. In this design example, *MSS_UART_0* is used for UART communication and the on-chip RC oscillator is used to source the 100 MHz MSS frequency. The external 1.5 V voltage regulator is used to supply power to the 1.5 V power rail. You can also configure the MSS peripherals using the example software application that is supplied.

Note: The internal 1.5 V voltage regulator output at power-up time can be set to ON using the MSS configurator, as shown in Figure 2. If the voltage regulator output at power-up time is set to OFF, power up the internal voltage regulator for system operation using external pin *PU_N*, which is available as a switch (SW7) on the SmartFusion Development Kit Board.

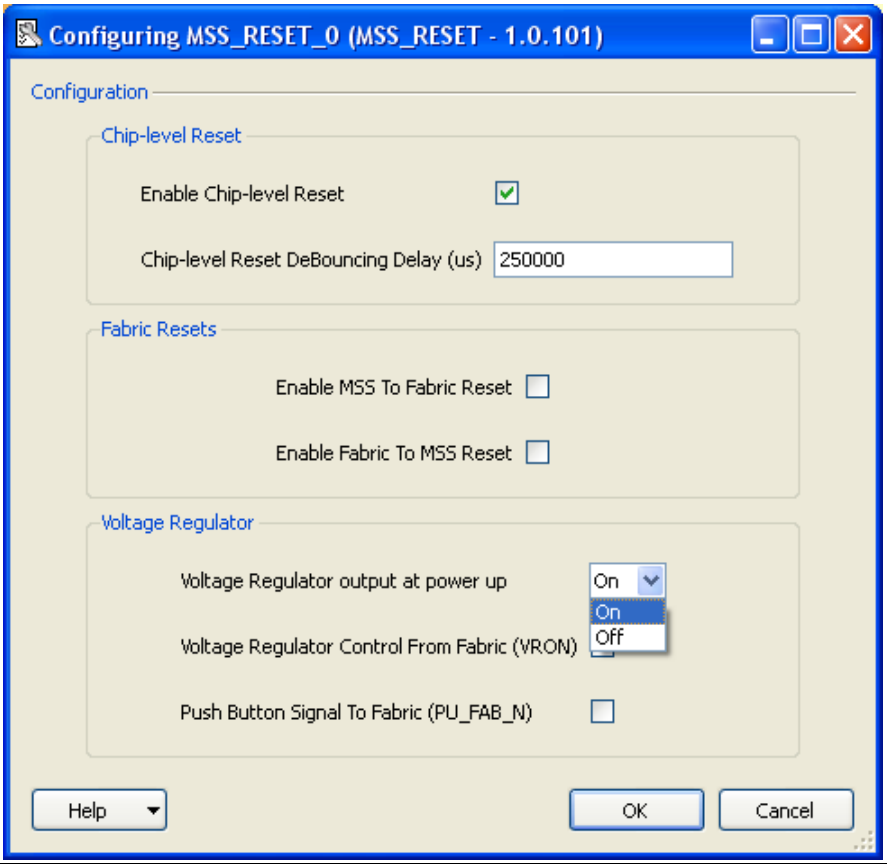

Figure 2 • **Voltage Regulator Output Configuration**

In the example hardware project, all the MSS peripherals are enabled and the internal voltage regulator output at power-up is set to ON. These peripherals are controlled through the example software application for power consumption demonstration purposes.

Refer to the *Using UART with SmartFusion: Libero SoC Flow Tutorial* to understand the SmartFusion hardware and software design flow.

Software Implementation

The example software design includes a real-time clock application program that establishes the UART-based serial communication with serial terminal emulation programs such as HyperTerminal for clock time display and system power management. This application provides a UART-based user interface to set the time, enable or disable functional blocks, and to switch between different power modes.

In the example software code, RTC is configured in Free running mode (no reset on RTC MATCH) to interrupt the Cortex-M3 processor and enable the internal voltage regulator on RTC MATCH. The application displays the RTC count in real-time counter format (Hr:Min:Sec) through the UART. You can set the RTC time through the UART interface. While the clock is running in the background, you can use the firmware to put the unused SmartFusion cSoC peripherals in a reset or powered down state and the system in low power modes such as Standby or Time keeping. In the example application, a top-level menu function (top_level_menu()) is implemented for the following tasks:

1. To set the new clock time
2. To configure the SmartFusion cSoC internal peripherals
3. To put the system into Standby mode
4. To put the system into Timekeeping mode

The function *config_smartfusion()* enables or disables the SmartFusion cSoC internal functional blocks by setting appropriate bits in the system control registers, such as *SOFT_RST_CR*, *ANA_COMM_CTRL*, and *ADCx_MISC_CTRL*. The following snippet shows the functions used for the MSS and analog peripherals control.

```
/* SmartFusion MSS peripherals enable/disable control function */
void peripheral_control(
    uint8_t User_Input_This_Selection,
    uint32_t Peripheral_Enable,
    uint32_t Peripheral_Disable
    )
{
    uint32_t i;
    const uint32_t RETAIN_MSS_CONFIG = 0xFFFFFFFF;
         if(User_Input_This_Selection == '1') {
         SYSREG->SOFT_RST_CR&= Peripheral_Enable;
         }
         else if(User_Input_This_Selection == '0') {
         SYSREG->SOFT_RST_CR|= Peripheral_Disable;
         }
         else{
         SYSREG->SOFT_RST_CR&= RETAIN_MSS_CONFIG;
      }
}
/* SmartFusion Analog Front-End control function */
void AnalogPowerDown(uint8_t Analog_Pwr_Selection)
{
    uint32_t i;
    const uint32_t RETAIN_MSS_CONFIG = 0xFFFFFFFF;
         if(Analog_Pwr_Selection == '1') {
         ACE->ANA_COMM_CTRL&= ANALOG_PWRDN;
         }
         else if(Analog_Pwr_Selection == '0') {
         ACE->ANA_COMM_CTRL|= ANALOG_PWRUP;
         }
         else{
         ACE->ANA_COMM_CTRL&= RETAIN_MSS_CONFIG;
         }
}
```

The example software takes the user input through the UART interface for controlling the SmartFusion cSoC peripherals.

Before putting the system in Standby mode, the given application calls the function *config_smartfusion()* to disable the unused peripherals for power saving. The function *standbymode_menu()* sets the RTC MATCH count based on user input and puts the Cortex-M3 processor in Sleep mode by executing the wait for interrupt (WFI) assembly instruction. This assembly instruction is implemented in the *PutCM3ToSleep()* function. While the Cortex-M3 processor is in Sleep mode, the system frequency is reduced to 32 KHz by switching the clock source to 32 KHz low-power crystal oscillator from 100MHz on-chip oscillator. This clock switching is performed by the *SwitchMSSClockto32Khz()* function. This switching saves the power consumption on both 1.5 V and 3.3 V power rails.

```
/* Cortex-M3 sleep instruction */
void PutCM3ToSleep()
{
    __ASM volatile ("WFI");
}
```

The RTC MATCH event from the RTC interrupts the Cortex-M3 processor NVIC module, which in turn wakes up the Cortex-M3 processor from the Sleep mode. If you want to enable the disabled blocks right away after a Cortex-M3 processor wake-up call, then you must write the functions after the *PutCM3ToSleep()* function.

After the Cortex-M3 processor comes out of Sleep mode, the system frequency is switched back to the original frequency 100 MHz using the S*witchMSSClocktoGLA()* function. Now the system resumes the application execution and displays the RTC current time on UART terminal.

The function *timekeeping_mode()* prompts you to ensure that the battery is connected on the VBAT rail to enter into Time keeping mode.

The example software project provided in the design files is in Debug mode. Refer to the *SmartFusion: Building Executable Image in Release Mode and Loading into eNVM Tutorial* for generating the application in release mode.

A description of the APIs used in the design is as follows:

1. The *top_level_menu()* function displays the main menu for setting clock time, enabling or disabling the SmartFusion cSoC blocks, and entering the Standby mode or Time keeping mode
2. The *display_time()* function displays the RTC time on HyperTerminal
3. The *set_time_menu()* function sets the clock time
4. The *config_smartfusion()* function enables or disables the SmartFusion cSoC peripherals
5. The *enable_disable_peripherals()* function accepts the user inputs for enabling or disabling the peripherals
6. The *manage_cm3_sleep()* function configures the RTC with required RTC MATCH value and puts the Cortex- M3 processor into Sleep mode
7. The *analogpowerdown()* function accepts the user inputs for enabling or disabling the entire SmartFusion analog block
8. The *ADC_control()* function accepts the user inputs for enabling or disabling the ADC blocks individually
9. The *SCBReset()* function puts the signal conditioning block (SCB) in low power state (reset)
10. The *SwitchMSSClockto32Khz()* function switches the system clock frequency to 32 KHz to save the power.
11. The *SwitchMSSClocktoGLA()* function switches the system clock frequency back to the frequency configured in the *MSS_CCC* block (100 MHz) of the MSS configurator.
12. The *standby_mode()* function puts the system in the Standby mode
13. The *timekeeping_mode()* function enables entry to the Time keeping mode

Running the Design

Connect one digital ammeter or multimeter in series with the 3.3 V supply source (across the JP2 jumper), and another one in series with the external voltage regulator supply (across pins 2 and 3 of the JP6 jumper) on the SmartFusion Development Kit Board.

Start a HyperTerminal session with a baud rate of 57,600, 8 data bits, 1 stop bit, no parity, and no flow control. If your PC does not have the HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, or PuTTY.

Program the SmartFusion Development Kit Board (A2F500-DEV-KIT) with the provided STP file (refer to the "Appendix A – Design Files" on page 13) using the FlashPro. Disconnect the low cost programmer USB cable and remove jumpers JP11 and JP12. Power cycle the board.

The serial terminal emulation program window shows the user interface as shown in Figure 3.

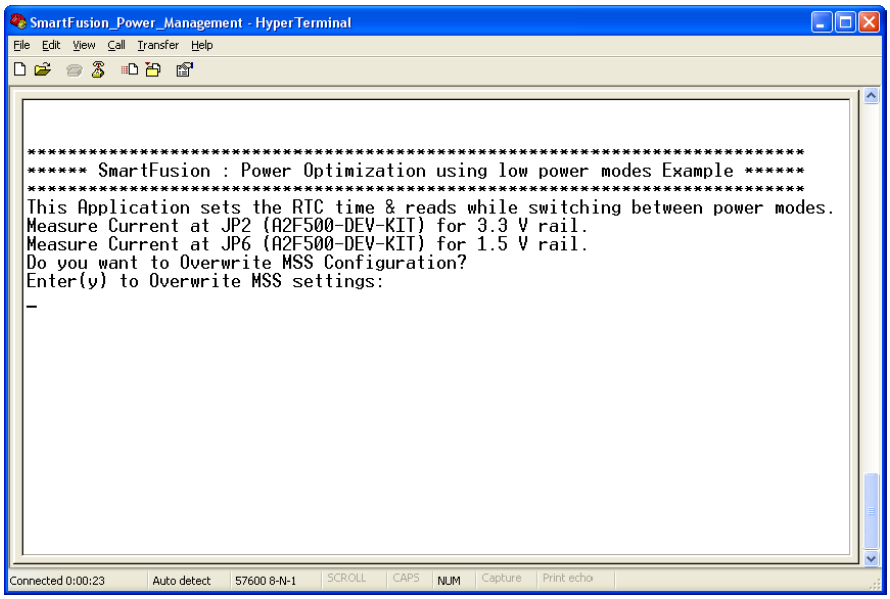

Figure 3 • **User Interface in HyperTerminal Window**

Follow the instructions displayed in the serial terminal emulation program. Press any key on the keyboard to display the top-level menu for setting the new clock time, to enable or disable the peripherals, and to enter Standby or Time keeping mode. Observe the ammeter current readings and total system power consumption can be calculated with the following equation:

$$\text{Total power consumption} = (1.5 \times I_{1.5\ V} + 3.3 \times I_{3.3\ V}) \text{ Watts}$$

The example software implements a real-time clock application and provides the firmware control of the Cortex-M3 processor, MSS peripherals, programmable analog peripherals, and an internal voltage regulator for system power management.

At application startup, the Cortex-M3 processor is active and all the MSS peripherals, memory controllers, programmable analog section and internal voltage regulator are enabled. In the example application, data activity is not happening with any of the MSS and analog peripherals except *MSS_UART0*. Set the clock time to current time without changing the MSS configuration, keeping the programmable analog section and voltage regulator enabled. Observe the current readings on the 1.5 V rail and 3.3 V rail while the clock time is being displayed on the HyperTerminal.

Using the UART-based user interface, put the unused MSS peripherals in reset and put the system into Standby mode. Standby mode puts the Cortex-M3 processor in Sleep mode and reduces the system clock frequency to 32 KHz. You can define the Cortex-M3 procesor's sleep time through the UART interface. After the Cortex-M3 processor comes out of Sleep mode, the system clock frequency switches back to original frequency for normal operation of the application. Observe the current consumption while the system is on Standby mode. Observe the current consumption by disabling the analog blocks. Table 3 on page 12 shows the current consumption on the 1.5 V rail and 3.3 V rail of a SmartFusion cSoC device with the different configurations.

Note: The given readings are observed while the clock time is being displayed on the HyperTerminal except for Standby mode.

These measurements are made using an A2F500-DEV-KIT board with an external regulated power supply (Agilent E3631A) to power the 1.5 V rail and an onboard voltage source to power the 3.3 V rail. Two digital multimeters (Agilent 34401A) are used to measure the current consumption. You can use the SmartPower tool available with Libero SoC for power analysis.

Note: The power numbers given are to show the relative power saving in different states (custom power modes) for the given example design running on A2F500-DEV-KIT at room temperature. The given values are based on measurements made on a single A2F500-DEV-KIT board. Refer to the 'SmartFusion DC and Switching Characteristics' chapter in the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet for quiescent supply current values.

Table 3 • **Power Consumption of the Example Design**

Design State	Cortex-M3	MSS Peripherals	Analog	Voltage Regulator (VR)	$I_{1.5\ V}$ (mA)	$I_{3.3\ V}$ (mA)	Power (mW)
Startup - SoC mode with VR ON	Active	Active	Enable	Enable	57.41	16.40	140.23
MSS peripherals except ACE held in reset and analog section is ON	Active	Reset except ACE[1]	Enable	Disable	56.50	14.81	133.62
All MSS peripherals except UART_0 are held in reset and the analog section is disabled	Active	Reset	Disable	Disable	56.47	13.97	130.80
Cortex-M3 is in sleep mode with MSS peripherals held in reset and the analog section is disabled	Sleep	Reset	Disable	Disable	43.60	13.98	111.53
Standby mode[2] - Cortex-M3 is in sleep mode, MSS peripherals held in reset, analog section disabled and system frequency reduced to 32 KHz	Sleep	Reset	Disable	Disable	8.2	2.6	20.88

Notes:

1. *Putting the ACE in reset disables the complete analog front-end.*
2. *By default, the main crystal oscillator is disabled and memory controllers are active since the application is running from eNVM.*

Conclusion

This application note explains various methods for controlling the power consumption and different low power modes supported by the SmartFusion cSoC devices.

Appendix A – Design Files

You can download the design files from the Microsemi SoC Products Group website:

www.microsemi.com/soc/download/rsc/?f=A2F_AC364_DF.

The design file consists of Libero SoC Verilog project, a SoftConsole software project in debug mode, and a programming file (*.STP) for A2F500 device. Refer to the Readme.txt file included in the design file for the directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC364_PF.

The programming zip file consists of STAPL programming file (*.stp) for A2F500-DEV-KIT and a Readme.txt file.

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 3 (February 2012)	Removed ".zip" extension in the links (SAR 36763).	13
Revision 2 (January 2012)	Modified the section "Running the Design" (SAR 35625).	
	Modified the section "Design Description" (SAR 35625).	10
	Modified the section "System Power Optimization" (SAR 35625).	6
	Modified the section "Hardware Implementation" for Libero v10.0 (SAR 35794).	7
	Modified the section "Software Implementation" (SAR 35625).	9
	Modified the section "Appendix A – Design Files" (SAR 35794).	13
Revision 1 (March 2011)	Modified the section "SmartFusion Power System" (SAR 35974).	4
	Modified the section "Design Description" (SAR 35974).	6
	Modified the section "Software Implementation" (SAR 35974).	9
	Modified the section "Running the Design" (SAR 35974).	10

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900232-3/02.12