AN358

OPTIMIZING LOW-POWER OPERATION OF THE C8051F9XX

Relevant Devices

This application note applies to the following devices:
C8051F930, C8051F931, C8051F920, C8051F921, C8051F912, C8051F911, C8051F902, C8051F901, C8051F990, C8051F991, C8051F996, C8051F997, C8051F980, C8051F981, C8051F982, C8051F983, C8051F985, C8051F986, C8051F987, C8051F988, C8051F989

1. Introduction

The C8051F9xx family of low-voltage/low-power MCUs is an excellent choice for battery-powered embedded systems. Following are some of the key features of this product family:

- **Low Active and Inactive Mode Current**
  - 150 µA/MHz, 160 µA/MHz, or 170 µA/MHz Active Mode Current @ 25 MHz
  - < 1 µA Sleep Mode Current
- **Fast Wakeup Time and Fast Code Execution**
  - 400 ns Suspend Mode Wakeup (using the low power internal oscillator)
  - 2 µs Sleep Mode Wakeup (two-cell mode) or
    10 µs Sleep Mode Wakeup (one-cell mode)
  - Up to 25 MIPs Operation
- **Fast ADC Acquisition Time**
  - 1.5 µs VREF turn-on time, occurs while ADC is tracking
  - 3.3 µs back-to-back analog acquisition time
- **Support for 1 and 2 Cell Battery Configurations**
  - 0.9–1.8 V supply voltage range allows the system to be powered from a single alkaline or silver oxide battery.
  - 1.8–3.6 V supply voltage range allows the system to be powered from a single lithium battery or two alkaline batteries placed in series.

The C8051F9xx MCU family is very flexible and provides application software control over many factors affecting device power consumption. This application note describes how to achieve maximum efficiency in each power mode and how to optimize application code to take advantage of the low power features of the C8051F9xx family.

Included with this application note is example software that can be used to place the C8051F9xx MCU into each of its power modes for supply current measurements. Also included is a low-power software template that may be used as a starting point for new code development. The software can be found in the AN358SW.zip archive, which is distributed with this application note.

2. Key Points

- **Two Regions of Operation**—C8051F9xx devices have two distinct regions of operation. The Flash one-shot circuit must be disabled (bypassed) for high system clocks and enabled for system clocks in order to minimize supply current in the normal power mode. The optimum crossover frequency is 10 MHz (C8051F93x/2x) or 14 MHz (C8051F91x/0x/9x/8x).
- **Sleep Mode Supply Current**—The current in Sleep mode should always be < 1 µA at room temperature, even when the SmaRTClock is running. If the current meter is measuring a current higher than 1 uA, then the device is not configured properly; one or more GPIO pins are sourcing current to an external circuit, or a high-speed signal is being applied to a port pin (e.g., an external CMOS clock).
- **Software Considerations**—The proper entry and exit procedures should be used when entering and exiting a low-power mode. This ensures that the device will work in a reliable and predictable manner.
- **SmaRTClock Alarm Events**—The SmaRTClock ALRM flag is not persistent and is automatically cleared by hardware after 1 SmaRTClock cycle. The RTCAWK flag in the PMU0CF register is persistent and can be used to detect a SmaRTClock alarm event after the ALRM flag has been cleared.

- **Flash Memory Operations**—If erasing Flash memory, be sure to set the SmaRTClock alarm interval to a value longer than 36 ms to ensure that an alarm is not missed. When writing Flash memory, ensure that the alarm interval is longer than 71 µs.
- **Measuring Current**—The software supplied with this application note allows the digital supply current specification in the data sheet to be achieved. Any current flowing through the GPIO pins is in addition to the digital supply current required to operate the device. For example, driving a 24.5 MHz clock signal on a GPIO pin with a 3.3 V supply voltage can increase the supply current by 3 mA.

3. Power Modes Overview

The C8051F9xx family of MCUs support five power modes: Normal, Idle, Stop, Suspend and Sleep. A summary of the power modes can be found in Table 1. Detailed descriptions of each mode can be found in the Power Management chapter of the device data sheet.

Normal and Idle modes are classified as *Active Power Modes* because the system clock is active and power consumption scales with the clock frequency. Typical supply currents for each of the three different system clock sources (24.5 MHz Precision Oscillator, 20 MHz Low Power Oscillator, and 32.768 kHz SmaRTClock Oscillator) are provided in Table 1. Stop, Suspend, and Sleep modes are classified as *Inactive Power Modes* because the system clock is stopped.

Since the system clock in most low-power applications is not always present, the C8051F9xx MCUs have an ultra-low-power SmaRTClock that can be used for timekeeping. The SmaRTClock oscillator requires less than 1 uA of supply current and can remain functional even when the MCU goes into its lowest power Sleep mode.

Table 1. Power Mode Summary (Two-Cell Mode)

Power Mode	Functionality	Typical Supply Current (C8051F93x/2x)	Typical Supply Current (C8051F91x/0x)	Typical Supply Current (C8051F99x/8x)
Normal	Device fully functional	4.1 mA @ 24.5 MHz 3.5 mA @ 20.0 MHz 90 µA @ 32.768 kHz (±10 µA for supply voltage variations)	4.0 mA @ 24.5 MHz 3.4 mA @ 20.0 MHz 84 µA @ 32.768 kHz (±10 µA for supply voltage variations)	3.6 mA @ 24.5 MHz 3.1 mA @ 20.0 MHz 84 µA @ 32.768 kHz (±10 µA for supply voltage variations)
Idle	All clocks and peripherals fully functional. Code execution paused.	2.5 mA @ 24.5 MHz 1.9 mA @ 20.0 MHz 84 µA @ 32.768 kHz (±10 µA for supply voltage variations)	2.1 mA @ 24.5 MHz 1.6 mA @ 20.0 MHz 82 µA @ 32.768 kHz (±10 µA for supply voltage variations)	2.1 mA @ 24.5 MHz 1.6 mA @ 20.0 MHz 82 µA @ 32.768 kHz (±10 µA for supply voltage variations)
Stop	Legacy 8051 low power mode.	Greater than or equal to Suspend Mode		
Suspend	All clocks stopped. Code execution paused.	75 µA @ 1.8 V 90 µA @ 3.6 V (Low Power Osc.)	75 µA @ 1.8 V 90 µA @ 3.6 V (Low Power Osc.)	75 µA @ 1.8 V 90 µA @ 3.6 V (Low Power Osc.)
Sleep	Internal regulator disabled, memory preserved. Code execution paused. Comparator0 only functional in two-cell mode.	w/ SmaRTClock Crystal 0.600 µA w/o SmaRTClock 0.050 uA	w/ SmaRTClock Crystal 0.600 µA w/ SmaRTClock LFO 0.300 µA* w/o SmaRTClock 0.050 µA w/o VBAT Supply Monitor 0.010 uA*	w/ SmaRTClock Crystal 0.600 µA w/ SmaRTClock LFO 0.300 µA* w/o SmaRTClock 0.050 µA w/o VBAT Supply Monitor 0.010 uA*
***Note:** BLUE refers to power modes only available on C8051F912, C8051F902, C8051F99x, and C8051F98x devices.				

4. Minimizing Active Mode Current

The active modes in a low-power system typically require the most supply current; however, they are the modes in which the most critical system tasks are completed. Minimizing Active mode time is one of the best power saving strategies. This can be achieved by operating at the fastest possible system clock frequency. Since the MCU is most efficient at fast system clocks, minimizing active mode time results in greater overall benefit than reducing peak current.

The following figures show the typical supply current in Normal and Idle modes as a function of the system clock frequency. There are two observations to note about the Normal mode curves: 1) at the crossover point, the slope of the supply current vs. frequency curve changes. This divides the curve into two piece wise linear regions. 2) The absolute current per MHz decreases as the system clock frequency increases. At low frequencies, the CPU operates at a higher µA/MHz. As frequency increases, the µA/MHz drops and can be less than 150 µA/MHz on some devices.

The Active supply current can be influenced by a number of factors including supply voltage, temperature, system clock frequency, power mode, and other factors under the control of application software.

4.1. Effect of Supply Voltage

In most CMOS circuits, supply voltage has the greatest effect on supply current. However, since the C8051F9xx MCUs have an on-chip LDO for regulating the voltage supplied to the digital circuitry, supply voltage has a minimal effect on supply current. In fact, the supply current variation over the entire input voltage range (1.8–3.6 V) is typically less than ±10 µA from the midpoint voltage of 2.7 V.

4.2. Effect of Temperature

Changes in temperature can affect the active supply current. As temperatures rise, the supply current also increases, and as temperatures drop, the supply current decreases. The supply current variation over the entire operating temperature range (–40 to +85 °C) is typically less than ±5% from the supply current measured at 25 °C.

4.3. Effect of System Clock Frequency

The system clock frequency has the most significant effect on the active supply current. As the clock frequency increases, supply current and power efficiency increase, as shown in the following figures. When executing a task that requires a fixed number of instructions, the system clock should be set as fast as possible. The limiting factor in increasing the system clock should be the ability of the power supply to handle the increased peak currents.

For tasks that require a fixed amount of time to complete (e.g., waiting for a UART byte to be clocked in), increasing the system clock actually decreases power efficiency because the peak current increases while no additional work is being completed. In these situations, the system clock frequency should be minimized and the CPU placed in Idle mode.

4.4. Effect of Power Mode

The C8051F9xx MCUs have two Active Power Modes in which the system clock is running. Normal mode power consumption is shown in Figure 1, Figure 3, and Figure 5, and Idle mode power consumption is shown in Figure 4, Figure 6, and Figure 6. As a rule of thumb, placing the CPU in Idle mode will typically reduce the supply current by approximately 50%.

4.5. Optimizing Application Software

To achieve the supply current measurements listed in this application note and in the device data sheet, application software must properly configure the device into its optimum power setting. These low-power optimizations for *Active Power Modes* are as follows:

1. For system clock frequencies greater than the crossover frequency, disable (bypass) the one-shot circuit by setting the BYPASS bit (FLSCL.6) to logic 1. For system clock frequencies less than the crossover frequency, enable the one-shot circuit by clearing the BYPASS bit (FLSCL.6) to logic 0 and immediately following this operation with a write of a non-zero value to the FLWR register (only required for C8051F93x/2x devices). A detailed description of the one-shot circuit can be found in the Flash chapter of the device data sheet. Leaving the one-shot enabled for frequencies higher than the crossover frequency can result in 40% higher supply current. Leaving the one-shot bypassed for frequencies less than crossover frequency can result in greater than 500% increase in supply current.

   Note: The optimum one-shot crossover frequency is 14 MHz on C8051F91x/0x/9x/8x devices and 10 MHz on C8051F93x\2x devices.

2. If the Low Power Oscillator is not selected as the system clock source, clear all wake-up source flags by writing 0x20 to the PMU0CF register. Always use direct writes or reads when accessing this register. Clearing the wake-up source flags allows the Low Power Oscillator to be automatically disabled by hardware when it is not needed. This optimization reduces the supply current by 100 uA.
3. If the Precision Oscillator is not selected as the system clock source, disable the Precision Oscillator Bias by clearing the OSCBIAS bit (REG0CN.4) to logic 0. This optimization reduces the supply current by 85 uA.
4. When using one of the internal oscillators as the system clock source, disable the missing clock detector reset source in the RSTSRC register. Always use direct writes or reads when accessing this register and be careful not to disable the VDD Monitor as a reset source. This optimization reduces the supply current by 10 uA.
5. Disable the 1.8 V supply monitor in systems that use an external supply monitor or when a proper supply voltage is guaranteed (e.g., a system that uses a permanent battery where behavior at battery end-of-life is a don't care). This optimization reduces the supply current by 5 to 20 uA depending on supply voltage.
6. Whenever possible, try to execute code at the fastest possible clock frequency and use Idle mode to pause code execution when waiting for a specific event to occur (e.g., timer overflow flag in a software delay loop, GPIO pin changing state, UART transmission or ADC conversion to complete, etc.).
7. If software contains small loops, such as a *while(1)* statement, ensure that the loop does not straddle a flash row boundary. Devices with 1024 byte Flash pages have a row boundary of 128 bytes, and devices with 512 byte Flash pages have a row boundary of 64 bytes. Supply current can increase by up to 30% when a short loop straddles a Flash row boundary. See the Flash chapter of the MCU data sheet for more details about minimizing Flash read current.

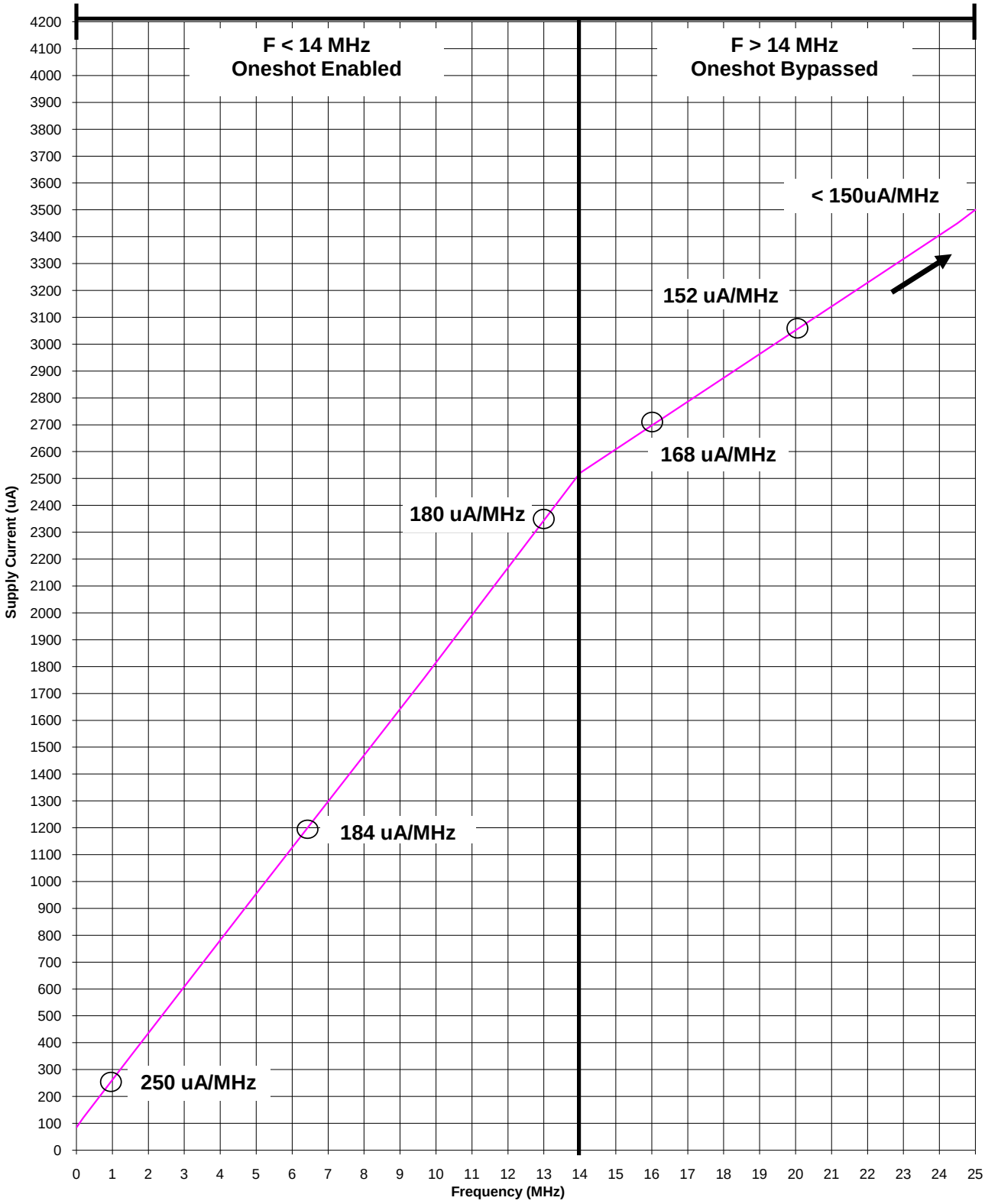

Figure 1. Typical Supply Current vs. Frequency (Normal Mode, External CMOS Clock, C8051F99x/8x)

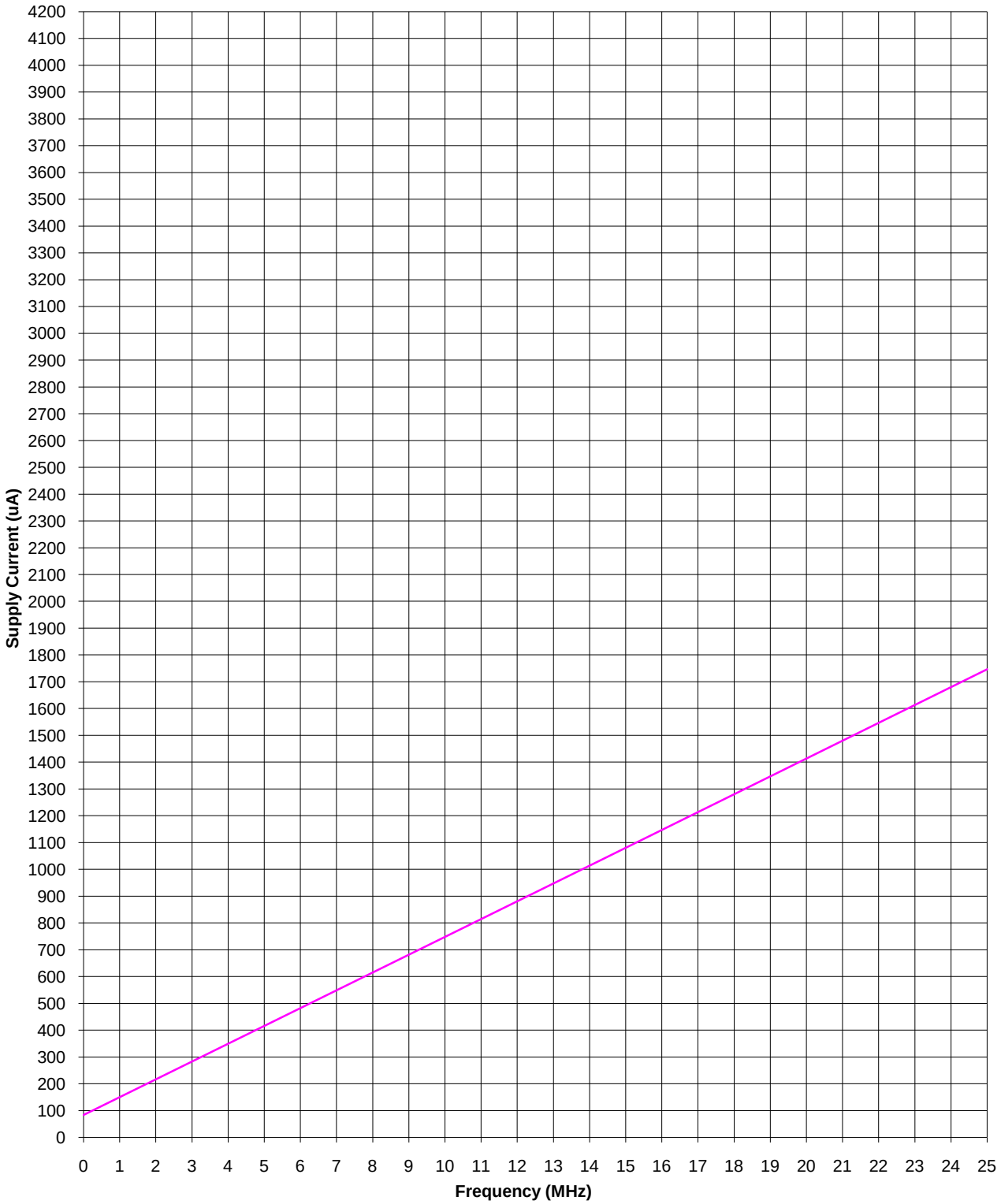

Figure 2. Typical Supply Current vs. Frequency (Idle Mode, External CMOS Clock, C8051F99x/8x)

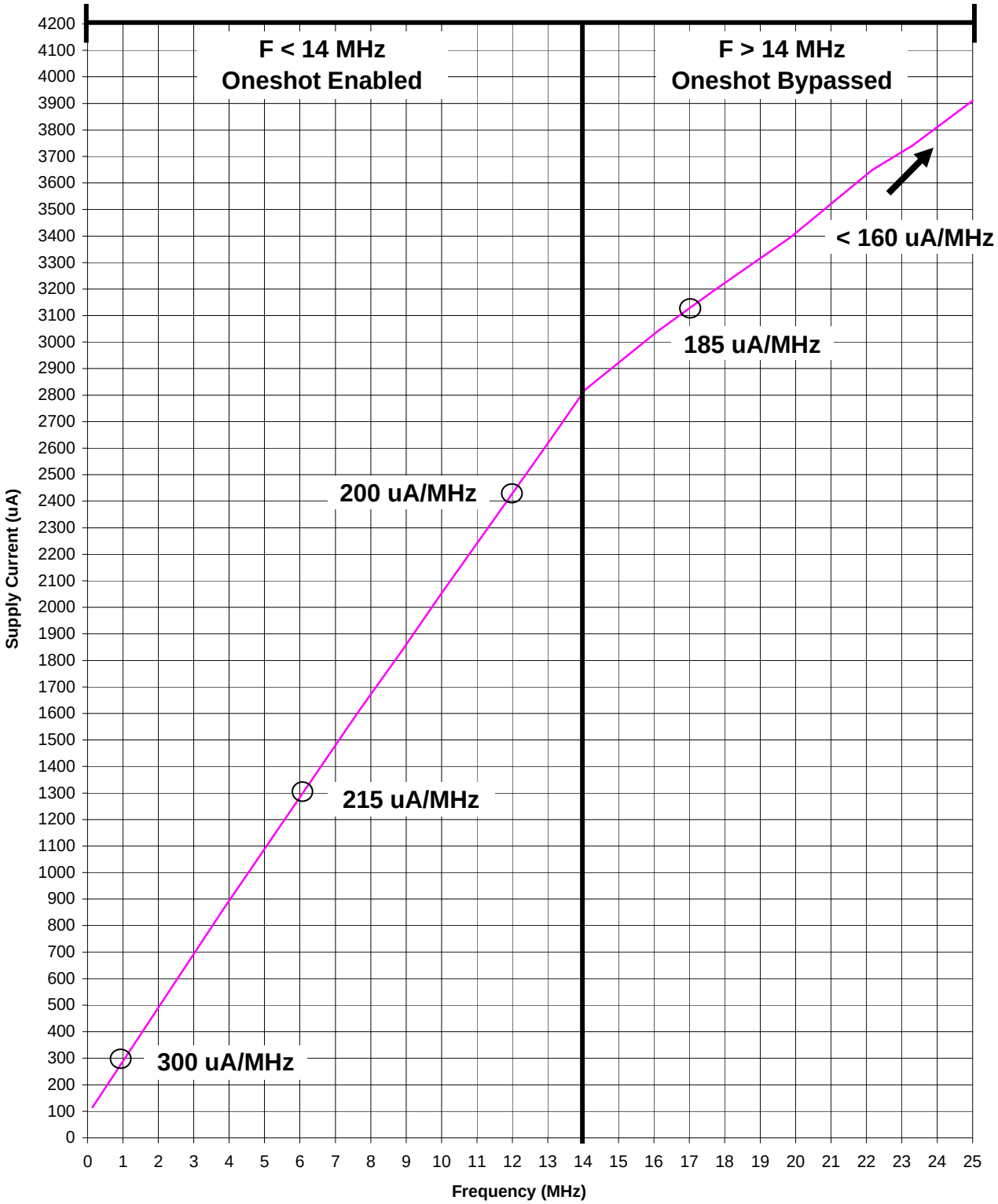

Figure 3. Typical Supply Current vs. Frequency (Normal Mode, External CMOS Clock, C8051F912/11/02/01)

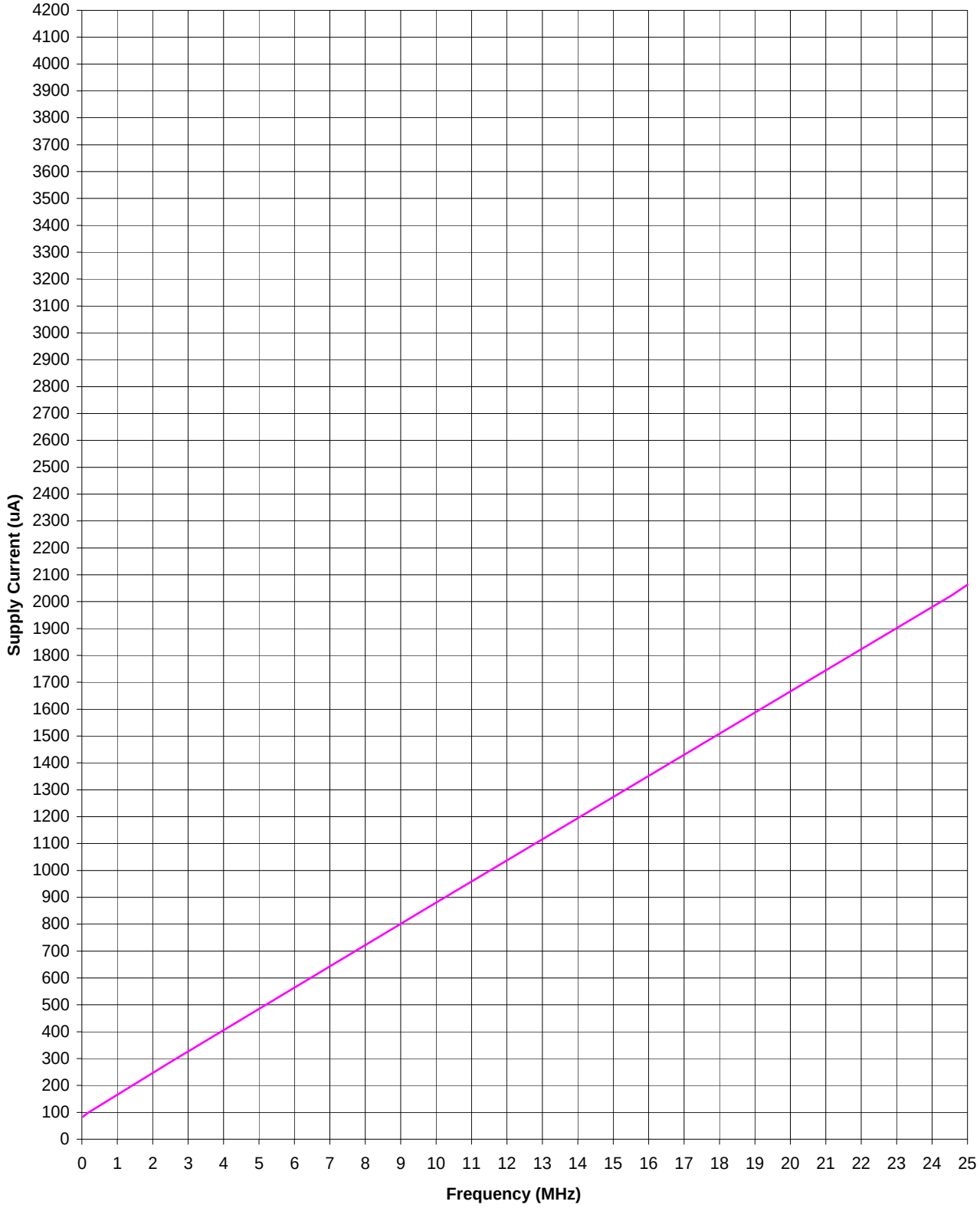

Figure 4. Typical Supply Current vs. Frequency (Idle Mode, External CMOS Clock, C8051F912/11/02/01)

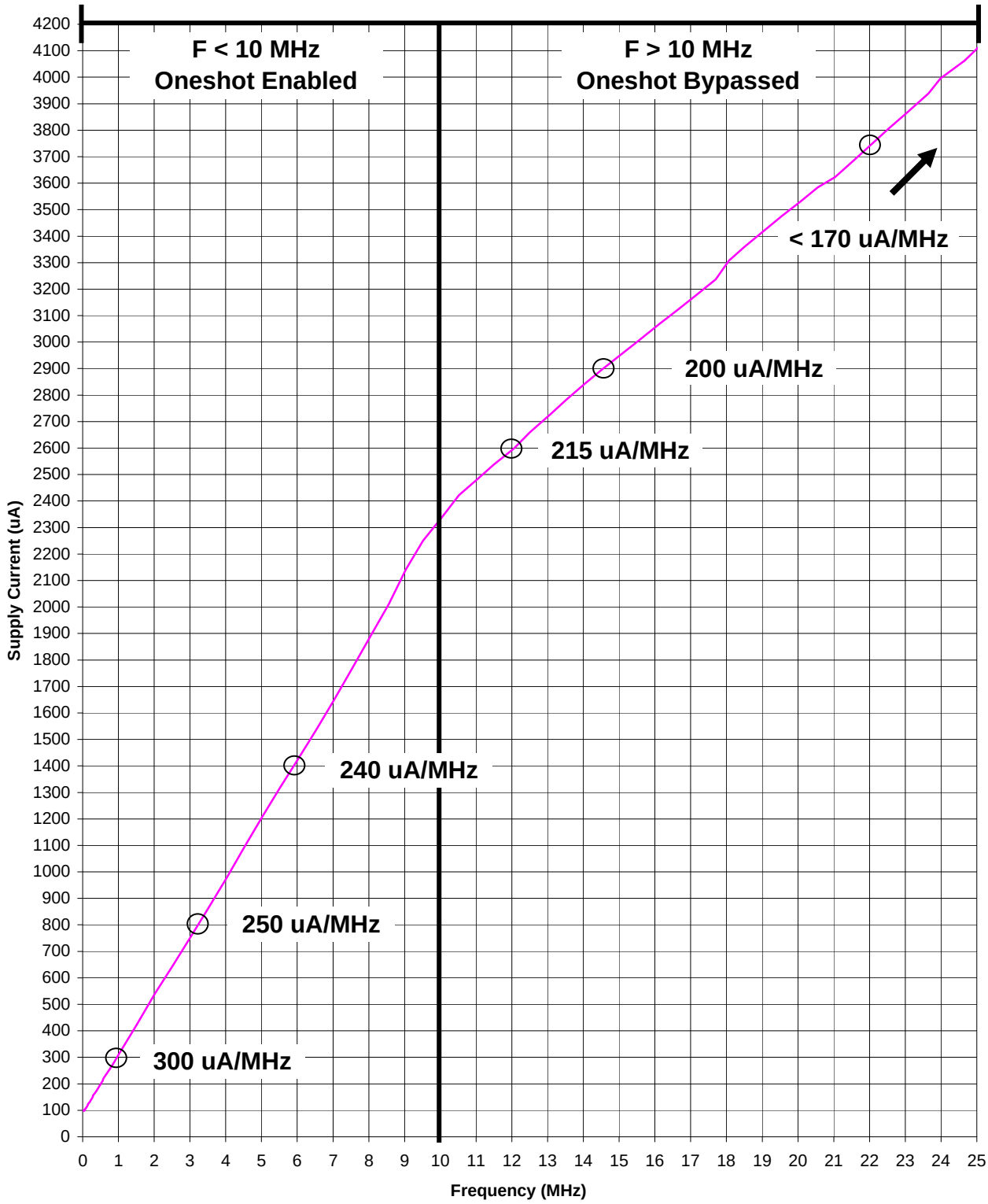

Figure 5. Typical Supply Current vs. Frequency (Normal Mode, External CMOS Clock, C8051F930/31/20/21)

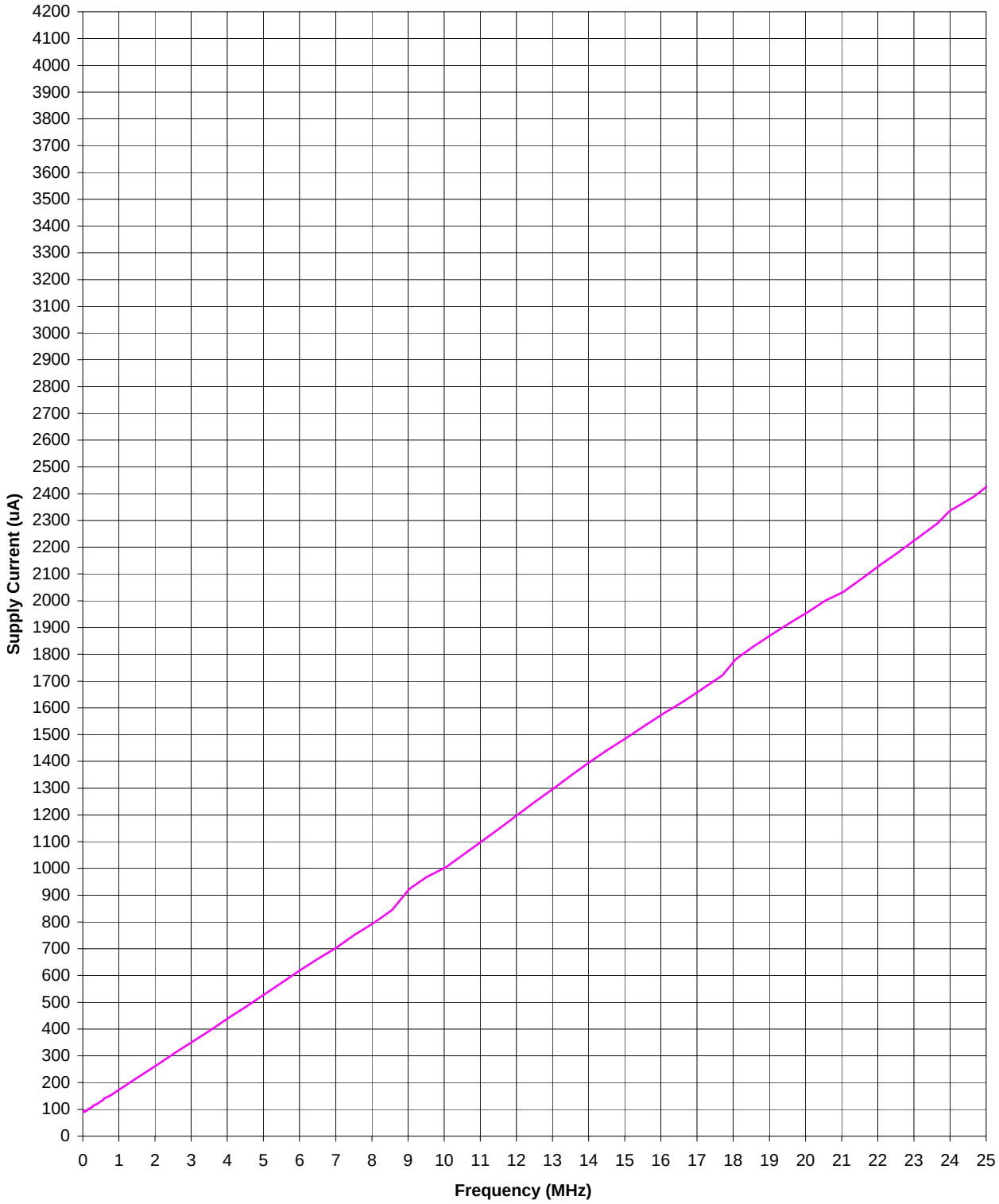

Figure 6. Typical Supply Current vs. Frequency (Idle Mode, External CMOS Clock, C8051F930/31/20/21)

5. Minimizing Inactive Mode Current

In most low-power applications, the MCU spends most of its time in the *Inactive Power Modes*. The C8051F9xx devices have an ultra-low-power Sleep mode in which the supply current drops below 1 µA. Two additional power modes, Suspend and Stop, are supported and allow the supply current to drop to 75–90 µA (two-cell mode) or 250–500 µA (one-cell mode). These power modes are discussed in the following sections.

5.1. Choosing an Inactive Power Mode

For two-cell applications operating from a supply voltage of 1.8–3.6 V, Sleep mode should be selected as the inactive power mode if the system can tolerate a typical wake-up time of 2 µs. This provides an inactive mode current less than 1 µA. If the application requires a faster wake-up, Suspend mode can provide a 400 ns wake-up time at the expense of increased inactive mode current (75–90 µA).

For one-cell applications operating from a supply voltage of 0.9–1.8 V, if the system can tolerate a typical wake-up time of 10 µs and the I/O voltage falling below 1.8 V, then Sleep mode should be selected as the inactive power mode. This provides an inactive mode current less than 1 µA. If the application requires a faster wake-up or cannot tolerate an I/O voltage below 1.8 V during the inactive state, then Suspend mode can provide a 400 ns wake-up time and a constant I/O voltage above 1.8 V at the expense of increased inactive mode current (75–90 µA plus any input current required by the dc-dc converter to maintain the VDD/DC+ supply rail at its programmed voltage).

When the dc-dc converter output voltage is programmed to 1.9 V, the Suspend mode current will range from 250–500 µA depending on the input voltage and the dc-dc converter settings. See "8. Minimizing One-Cell Mode Current" on page 20 for more information on how to configure the dc-dc converter to maximize power efficiency.

Stop mode is a legacy power mode that may be used in two-cell or one-cell applications. The inactive mode current in Stop mode is identical to that in Suspend mode; however, the MCU requires a reset in order to wake up from Stop mode. This makes the wake-up time from Stop mode very long when compared to the 400 ns required to wake up from Suspend mode. Under all circumstances, choosing Sleep or Suspend as the inactive power mode will provide more benefit to the system than Stop mode.

5.2. Achieving <1 µA Supply Current

Once the MCU is placed in Sleep mode, the current meter should display a value below 1 µA in one-cell or two-cell mode. If the current meter is capable of resolving currents smaller than 1 µA, it should display a value between 50 nA and 110 nA, depending on supply voltage (0.9–3.6 V) with the SmaRTClock disabled. If the SmaRTClock is enabled, the current meter should display a value between 420 and 815 nA depending on supply voltage (0.9–3.6 V).

If the measured current falls outside these ranges, then the MCU is not entering Sleep mode or is sourcing/sinking current through its GPIO pins. Following are some suggestions for achieving a <1 µA supply current:

1. Ensure that the USB Debug Adapter's ribbon cable is removed from the 10-pin debug socket on the target board. Leaving the USB Debug Adapter connected adds between 2–4 µA to the supply current measurement.
2. Ensure that all shorting blocks (with the exception of J11) are removed from the target board and that current is being measured across J17 or H2.

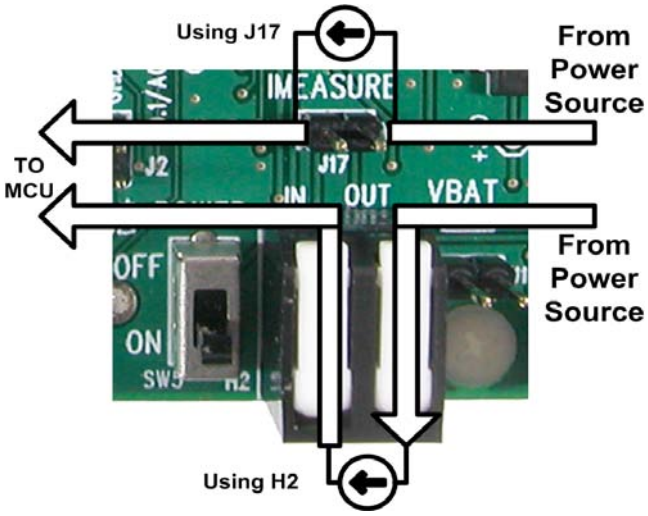

Figure 7. Current Measurement Setup on a C8051F930 Target Board

3. Ensure that digital inputs are driven or pulled to a HIGH logic state. Digital inputs in a LOW logic state can increase supply current by 0.2–20 µA depending on supply voltage due to the on-chip weak pull-up. If the logic state of the input cannot be controlled, the GPIO pin may be placed in Analog mode to disable the weak pull-up.
4. Ensure that analog signals are not applied to digital inputs. This can cause both the top and bottom transistors of CMOS logic gates to weakly turn on (i.e., Crowbar), causing the supply current to increase.
5. Ensure that all capacitors used in the system are low leakage ceramic capacitors. To demonstrate the effect of capacitor leakage, a 1 µF ceramic capacitor can increase the inactive supply current by approximately 3 nA at 3.0 V. An equal valued tantalum capacitor at the same voltage can increase the inactive supply current by up to 1 µA.
6. Ensure that application code is properly placing the device in Sleep mode. We recommend using the software supplied with this application note to place the device in its various power modes for measuring supply current.

5.3. Entering and Exiting the Sleep and Suspend Inactive Power Modes

In order to ensure proper entry and exit from the Sleep and Suspend inactive power modes, software should follow the recommendations in this section.

Sleep and Suspend Mode Entry Procedure:

1. Save the contents of the CLKSEL register; then, force the global clock divider to its divide-by-one setting and ensure that either the Low Power Oscillator or the Precision Oscillator is selected as the system clock source. The two valid values for CLKSEL at this point are: 0x00 and 0x04. Using C:

```
CLKSEL_save = CLKSEL;
CLKSEL = 0x04;
```

   Note: Per the C8051F930/31/20/21 errata, on Revision D and the earlier silicon, the value of CLKSEL should be 0x14 when entering Sleep or Suspend Mode. This errata item does not apply to C8051F91x/0x/9x/8x devices or to Revision E of C8051F930/31/20/21 devices.

2. Wait for the clock divider value to be applied by polling CLKSEL until the CLKRDY bit is set to 1. This can be achieved using the following C statement:

```
while((CLKSEL & 0x80) == 0);
```

   Note: Step 1 and Step 2 may be omitted if the system clock used in the system is already set to the value required for entering Suspend or Sleep mode or when using C8051F91x/0x/9x/8x devices.

3. Enable the Flash read one-shot timer if entering Suspend mode. This can be achieved with the following C statement:

```
FLSCL  &= ~0x40;
FLWR   =  0x01;
```

Notes:

1. Writing a dummy value to FLWR after clearing the BYPASS bit (FLSCL.6) is not required for C8051F91x/0x/9x/8x devices.

2. Enabling the Flash read one-shot timer is not required when entering Sleep mode. It is required for all other low-power modes.

4. Clear all wake-up source flags in PMU0CF. Be sure to enable interrupts for transient events before clearing the wake-up source flags. Using C:

```
PMU0CF = 0x20;
```

5. Place the device in the selected power mode and specify the desired wake-up sources.

   The power modes are:

```
#define SLEEP       0x80
#define SUSPEND     0x40
```

   The wake up sources are:

```
#define CP0         0x01
#define PORT_MATCH  0x02
#define RTC_ALRM    0x04
#define RTC_FAIL    0x08
#define RST         0x10

PMU0CF = (SLEEP + (PORT_MATCH | RTC_ALRM));
```

Sleep and Suspend Mode Exit Procedure:

1. Execute 4 NOP instructions. This can be done with the following C statement:

```
NOP(); NOP(); NOP(); NOP();
```

2. Restore the contents of the CLKSEL register. This can be done with the following C statement:

```
CLKSEL = CLKSEL_save;
```

3. Wait for the clock divider value to be applied by polling CLKSEL until the CLKRDY bit is set to 1. This can be achieved using the following C statement:

```
while((CLKSEL & 0x80) == 0);
```

   Note: Step 2 and Step 3 of the Exit Procedure may be omitted if Step 1 and Step 2 have been omitted upon entry into the low-power mode.

4. If the system clock is greater than the crossover frequency, bypass (disable) the Flash read one-shot timer. This can be achieved with the following C statement:

```
#if(SYSCLK > 10000000)
FLSCL |= 0x40;
#endif
```

   Note: The Flash read one-shot enabled/bypassed state is preserved. In most cases, this step can be omitted if Step 3 was omitted during entry into Sleep mode.

5. Decode the wake-up source flags. If the cause of wake-up is a falling edge on /RST, then the MCU must not be allowed to enter the low-power mode for a period of 15 µs. This provides the MCU sufficient time to respond to a pin reset event or synchronize with the debugger. Failing to insert a 15 µs delay before reentering the low-power mode can result in the MCU becoming non-responsive to the reset pin or disconnecting from the IDE.

```
wakeup_source = PMU0CF & 0x1F;
```

```
if (wakeup_source & RST)
{
   Wait_US(15);
}
if (wakeup_source & PORT_MATCH){}
if (wakeup_source & RTC_ALRM){}
if (wakeup_source & RTC_FAIL){}
if (wakeup_source & CP0){}
```

Additional Step for High-Current Applications:

Applications that have high-current requirements in their active mode may experience a power-fail reset upon waking up from Sleep mode. This occurs more frequently for one-cell applications. For these applications, we recommend adding the following step:

Immediately before entering Sleep mode:

```
RSTSRC = 0x00; // Disable VDD Monitor Reset
```

Immediately after waking from Sleep mode:

```
#define VDDOK 0x20 // Bit 5 of VDM0CN

// Poll the VDD Monitor Early Warning Bit
while((VDM0CN & VDDOK) == 0);
RSTSRC = 0x02; // Enable VDD Monitor Reset
```

Note: Constants written to RSTSRC used to enable and disable the VDD Monitor may vary depending on the enabled reset sources in the system.

6. An Event-Driven Architecture

In order to minimize average current and prolong battery life in portable embedded systems, attention must be given to how the application code is structured. An event-driven software architecture in which the MCU spends most of its time in the inactive state and only waking to handle specific events, has proven to be one of the most power-efficient ways to organize application code. Figure 8 shows a typical event-driven program flow.

```
// Device Initialization
MCU_Init();

// Main Application Loop
while(1)
{
   1. Perform Task A (Event Handler)
   2. Perform Task B (Event Handler)
   3. Perform Task C (Event Handler)
   ...
   Final Task:
   Enter Sleep Mode
   ~ MCU Sleeping ~
   ~ Event Occurs ~
   Exit Sleep Mode
}
```

Figure 8. Event Driven Program Flow

6.1. Periodic and Random Tasks

All tasks performed by the MCU can be classified as periodic or random. Periodic tasks, such as a real time clock function, generate events that periodically wake up the MCU from its inactive power mode. These tasks typically require the use of the SmaRTClock to generate periodic wake-up events. Random tasks, such as a switch press, can generate wake-up events using Port Match, which allows any rising or falling edge on a GPIO pin to wake the device from its inactive power mode. The inactive supply current for random tasks is lower than the inactive current for periodic tasks since the SmaRTClock oscillator can be turned off.

The C8051F9xx MCU family also supports one additional type of wake-up, Comparator 0. When the VDD/DC+ supply is present (two-cell Sleep mode or one-cell Suspend mode) Comparator 0 may be used to wake the device from Suspend or Sleep. Comparator 0 requires 0.4 uA in its lowest power setting and may be used to wake up the MCU upon crossing of an analog threshold.

6.2. Transient and Persistent Events

All wake-up events (whether random or periodic) will either be transient or persistent. Transient events are only present for a finite duration and persistent events remain present indefinitely until the event handler responds to the event.

In order to reliably and effectively handle transient events, the C8051F9xx MCUs have a power management unit (PMU0) which captures wake-up events and maintains wake-up source flags located in the PMU0CF register. All wake-up events captured by PMU0 are edge-triggered. This allows events to be captured without any clocks being active. The wake-up source flags in the PMU0CF register are persistent, and will remain asserted until cleared by software.

6.3. Handling SmaRTClock Events

The SmaRTClock can generate two events: SmaRTClock Alarm or SmaRTClock Oscillator Fail. SmaRTClock alarms are considered transient because the alarm event flag (ALRM) remains asserted for only one SmaRTClock oscillator cycle and SmaRTClock oscillator fail events are persistent because the oscillator fail event flag (OSCFAIL) can only be cleared by software. The wake-up source flags in PMU0CF that capture these SmaRTClock events are persistent and will remain asserted until cleared by software.

The recommended method of handling a SmaRTClock alarm event is to create a software flag that indicates that this event is pending. The software flag should be set any time the SmaRTClock alarm wake-up source flag in the PMU0CF register is set. This provides a 99.9% guarantee that all SmaRTClock events will be captured.

To achieve 100% coverage, we must check for SmaRTClock events that occur in the same instruction cycle in which the PMU0CF register is cleared. This can be done by enabling SmaRTClock interrupts before the clear operation begins and disabling them after the clear operation completes. If a SmaRTClock event happens during this brief period of time, the interrupt service routine should simply set the software <alarm_pending> flag to indicate that a SmaRTClock event has occurred.

The main application loop should use the software flag to detect when a SmaRTClock alarm has occurred. Upon entry into the event handler, the <alarm_pending> flag, the ALRM flag, and the PMU0CF register should be cleared to prevent the same event from being handled more than once. Figure 9 shows an example of how to handle SmaRTClock alarm events. Figure 10 shows an example of how to handle SmaRTClock oscillator fail events. Note the difference in implementation is due to the alarm being a transient event and the oscillator fail being a persistent event.

In these examples, there is only one transient event which is detected using the wake-up source flags in PMU0CF. If there was a second transient event (e.g., a transient port match event), then additional care should be taken when clearing PMU0CF. The clear operation would consist of the following steps:

1. Enable interrupts for all transient events.
2. Read the PMU0CF register to ensure no transient event has already occurred before enabling the interrupt.
3. Clear the PMU0CF register.
4. Disable interrupts for the transient events.

If multiple asynchronous events are enabled, be sure to test the firmware routines under worst case conditions to ensure that all events are properly captured and handled.

6.4. Handling Port Match Events

Depending on the application, changes in GPIO state can be transient or persistent. For example, the detection of a switch press is a transient phenomenon; however, once the switch is pressed, it is typically not released for a few hundred milliseconds. If the application code can guarantee that the event handler can be reached before the user removes their finger from the switch, then the event can be treated as persistent.

Port match events should be handled similar to SmaRTClock events—transient port match events should be detected using the PMU0CF register and persistent port match events can be detected by reading the GPIO pin state directly.

```
// Main Application Loop
while(1)
{
   //---------------------------------
   // SmaRTClock Alarm Task
   //---------------------------------
   if(PMU0CF & 0x04)
     alarm_pending = 1;

   // Alarm Handler
   if(alarm_pending)
   {
     // Clear the <alarm_pending> flag
     alarm_pending = 0;

     // Clear the ALRM flag by disabling
     // then re-enabling the alarm
     RTC_Write(RTC0CN, 0xD4); // Disable
     RTC_Write(RTC0CN, 0xDC); // Re-enable

     // Clear PMU0CF wakeup source flags
     EIE1 |= 0x02;  // Enable Alarm Int.
     PMU0CF = 0x20;  // Clear wake-up flags
     EIE1 &= ~0x02;  // Disable Alarm Int.

     // Additional Actions
     ...
   }
}
void ALARM_ISR (void) interrupt 8
{
   // Disable the Alarm interrupt
   EIE1 &= ~0x02;

   // Set the <alarm_pending> software flag
   alarm_pending = 1;
}
```

Figure 9. Handling a Transient Event

```
// Main Application Loop
while(1)
{
   //---------------------------------
   // SmaRTClock Osc. Fail Task
   //---------------------------------
   if(RTC_Read(RTC0CN) & 0x20)
     smaRTClock_fail = 1;

   // Osc. Fail Handler
   if(smaRTClock_fail)
   {
     // Clear the <smaRTClock_fail> flag
     smaRTClock_fail = 0;

     // Clear the OSCFAIL flag
     // by writing to the RTC0CN register
     RTC_Write(RTC0CN, 0xDC);

     // Additional Action
     ...
   }
}
```

Figure 10. Handling a Persistent Event

7. Low Power Software Template

To jump start the process of software development, a low power software template is distributed with this application note. The software template is a good starting point for any C8051F9xx firmware project that will be using Suspend or Sleep mode in an event driven application that is structured as shown in Figure 8.

The software template is divided into two primary modules, a “smartclock” module and a “power” module. The “smartclock” module handles all software interaction with the SmaRTClock peripherals and provides the following API functions:

- **RTC_Init()**—Starts the SmaRTClock oscillator in crystal or self-oscillate/LFO mode.
- **RTC_WriteAlarm()**—Writes a 32-bit value to the ALARM registers.
- **RTC_GetCurrentTime()**—Reads the 32-bit value representing the current time.
- **RTC_SetCurrentTime()**—Writes the passed 32-bit value to the main SmaRTClock counter.
- **RTC_SleepTicks()**—Places the device in Sleep mode for the specified number of SmaRTClock cycles. This function should only be used when the SmaRTClock is configured for Auto Reset mode.
- **RTC0CN_SetBits()**—Used to set bits in the SmaRTClock control register.
- **RTC0CN_ClearBits()**—Used to clear bits in the SmaRTClock control register.
- **RTC_Write()—**Used to write to an indirect SmaRTClock register.
- **RTC_Read()**—Used to read from an indirect SmaRTClock register.

The “power” module handles entry and exit of low power modes. It provides the following API functions:

- **LPM_Init()**—Initializes the low power mode API.
- **LPM_Enable_Wakeup()**—Enables the SmaRTClock, Port Match, or Comparator as a wake-up source from Sleep or Suspend Mode.
- **LPM_Disable_Wakeup()**—Disables the SmaRTClock, Port Match, or Comparator as a wake-up source from Sleep or Suspend Mode.
- **LPM()**—Called with an argument SLEEP or SUSPEND to place the device in a low power mode. Device will wake up once an enabled wake-up source event occurs.

7.1. Software Template Example

A software example that uses the software template is bundled with this application note. The example configures the SmaRTClock to generate an alarm every 100 ms. On every alarm, a 1 ms pulse is generated to be used as an oscilloscope trigger and an ADC conversion is initiated. The software template example can be easily modified to fulfill the requirements of the end application.

7.2. Configuring the Software Template

The software template has a number of compile-time configuration options that can be used to customize the software.

7.2.1. Configuration Options in C8051F930_lib.h

- **SYSCLK**—Defines the system clock frequency in Hertz. Used to calculate timer reload values and to determine if the Flash read one-shot should be enabled or disabled.
- **SMARTCLOCK_ENABLED**—Enables SmaRTClock functionality. When set to 0, the SmaRTClock routines will be excluded from the project build.

7.2.2. Configuration Options in SmaRTClock.h

- **RTC_CLKSRC**—Set to CRYSTAL to operate the SmaRTClock with a 32.768 kHz crystal or set to SELFOSC to operate the SmaRTClock in self-oscillate mode.
- **LOADCAP_VALUE**—Sets the programmed value of load capacitance for the SmaRTClock.
- **WAKE_INTERVAL**—The number of milliseconds between SmaRTClock alarms.

7.3. Additional Examples

Additional examples that use the software template can be found in the MCU examples folder for the device being used. The default path is as follows:

C:\Silabs\MCU\Examples\C8051F93x_92x\SleepMode\

or

C:\Silabs\MCU\Examples\C8051F91x_90x\SleepMode\

or

C:\Silabs\MCU\Examples\C8051F99x_98x\SleepMode\

8. Minimizing One-Cell Mode Current

In one-cell mode, the MCU may be powered from a 0.9 to 1.8 V supply. An on-chip dc-dc converter is used to boost the supply voltage up to a programmed value between 1.8 and 3.3 V. This voltage appears on the VDD/DC+ supply pin. The supply current taken from the VDD/DC+ supply pin to operate the MCU is equal to the supply current in two-cell mode.

From a power conservation standpoint, it is important to note that the input power (battery voltage x battery current) will always be equal to the output power (voltage x current at the VDD/DC+ pin) scaled by the efficiency factor. Since the battery voltage must be 0.2 V less than the output voltage, the battery current will always be higher than the two-cell mode supply current.

The battery current can be calculated from the two-cell mode supply current using Equation 1. Table 2 shows typical one-cell mode battery current as compared to two-cell mode supply current.

$$\text{Battery Current (one-cell mode)} = \frac{\text{Supply Voltage} \times \text{Supply Current (two-cell mode)}}{\text{DC-DC Converter Efficiency} \times \text{VBAT Voltage}}$$

Equation 1. Calculating One-Cell Battery Current

The following factors can help reduce battery current:

- Maximize the input battery voltage.
- Minimize the output supply voltage.
- Minimize the output supply current.
- Maximize the dc-dc converter efficiency.

8.1. Input Battery Voltage

Battery voltage has a large effect on the battery current. Since the dc-dc converter maintains a constant power output, the input current increases as the input voltage decreases. If the system is powered by a regulator, the ideal VBAT voltage is 1.7 V; however, when powered by a battery, the only control the system designer has over the VBAT voltage is in the selection of a battery chemistry.

To see the effect of battery chemistry on battery current, compare an Alkaline AAA to Lithium AAA at 50% remaining capacity. The Alkaline battery has a voltage of approximately 1.25 V while the Lithium battery has a voltage of approximately 1.5 V. When operating the MCU at 24.5 MHz, the Lithium battery needs to supply 20% less current than the Alkaline battery (6.3 vs. 7.8 mA) in order to maintain the same output power. This results in a significant boost in battery life. Other benefits of Lithium over Alkaline batteries are higher charge densities resulting in high capacity AA and AAA batteries, very low self-discharge currents resulting in prolonged shelf life, and the ability to support high drain applications without a significant reduction in battery capacity.

8.2. Output Supply Voltage

The output supply voltage (VDD/DC+) can be programmed to values between 1.8 and 3.3 V using the DC0CN Register. The recommended supply voltage that provides the most power efficient operation is 1.9 V. This is the default supply voltage at reset. Higher supply voltages should only be used when required (e.g., the system needs to turn on a blue or white LED).

The following example demonstrates the effect of output supply voltage on the battery current when operating the MCU at 24.5 MHz. A 60% increase in battery current (6.3 to 10 mA) is experienced when the supply voltage is increased from 1.9 to 2.7 V.

8.3. Output Supply Current

The output supply current (equivalent to the two-cell mode supply current) has a direct impact on battery current. The change in battery current as a result of an increase in supply current is typically 30–50% higher than the actual change in supply current. The supply current can be minimized using the same techniques described in this application note for two-cell mode.

8.4. DC-DC Converter Efficiency

The dc-dc converter efficiency depends on the input battery voltage, output supply current, component selection and software settings. Figure 11 through Figure 13 show the typical dc-dc converter efficiencies for various input battery voltages and output supply currents. When optimizing dc-dc converter efficiency, the system designer typically has little control over these parameters since they are usually determined by other constraints in the system.

The parameters which the system designer has the most control over are component selection, PCB layout, and software settings. The inductor should be chosen to have minimum dc resistance and a high current rating. The input and output capacitors should be low leakage ceramic capacitors. Recommended parameters for the required inductor and decoupling capacitors can be found in Table 4.14 of the MCU data sheet.

The PCB layout can have an effect on the dc-dc converter efficiency. In order to maximize efficiency, the inductor should be placed as close as possible to the DCEN pin and the capacitance of the trace connecting the inductor to DCEN should be minimized. The current loop consisting of the input capacitor, inductor, DCEN pin and the ground plane should be made as small as possible. On the output side, the traces connecting the VDD/DC+ pin to the output capacitor and the output capacitor to the GND/DC– pin should be as short and thick as possible in order to minimize parasitic inductance.

The software settings that affect dc-dc converter efficiency can be found in the DC0CN and DC0CF registers. The ideal setting for each of these parameters depends on the power requirements of the application. The system designer should experiment with the following settings until the optimum setting is determined for any particular application.

- **Output Voltage Select** - The output voltage select bits allow the output voltage to be programmed between 1.8 to 3.3 V. The lower the setting, the higher the efficiency. We recommend not setting the target value lower than 1.9 V to allow some margin above the VDD Monitor Threshold.
- **Switch Select** - The dc-dc converter provides two switches (large and small) for use under different load conditions. For small loads, the small switch should provide higher efficiency. For large loads, the large switch should provide higher efficiency. Under some conditions (e.g., when minimum pulse width is enabled), the large switch may provide higher efficiency at low currents.

Table 2. Power Mode Summary (One-Cell Mode)

Power Mode	Typical Two-Cell Supply Current (C8051F930/31/20/21)	Typical One-Cell Battery Current (C8051F930/31/20/21) VBAT = 1.5V	Typical Two-Cell Supply Current (C8051F912/11/02/01)	Typical One-Cell Battery Current (C8051F912/11/02/01) VBAT = 1.5V
Normal	4.1 mA @ 24.5 MHz 3.5 mA @ 20.0 MHz 90 µA @ 32.768 kHz (±10 µA for supply voltage variations)	6.3 mA @ 24.5 MHz 5.4 mA @ 20.0 MHz 360 µA @ 32.768 kHz	4.0 mA @ 24.5 MHz 3.4 mA @ 20.0 MHz 84 µA @ 32.768 kHz (±10 µA for supply voltage variations)	6.2 mA @ 24.5 MHz 5.2 mA @ 20.0 MHz 320 µA @ 32.768 kHz w/ LPM Enabled 285 µA @ 1.9 V
Idle	2.5 mA @ 24.5 MHz 1.9 mA @ 20.0 MHz 84 µA @ 32.768 kHz (±10 µA for supply voltage variations)	3.7 mA @ 24.5 MHz 2.9 mA @ 20.0 MHz 350 µA @ 32.768 kHz	2.1 mA @ 24.5 MHz 1.6 mA @ 20.0 MHz 82 µA @ 32.768 kHz (±10 µA for supply voltage variations)	3.3 mA @ 24.5 MHz 2.5 mA @ 20.0 MHz 316µA @ 32.768 kHz w/ LPM Enabled 280 µA @ 1.9 V
Suspend	75 µA @ 1.8 V 90 µA @ 3.6 V (Low Power Osc.)	330 µA @ 1.9 V	75 µA @ 1.8 V 90 µA @ 3.6 V (Low Power Osc.)	310 µA @ 1.9 V (Low Power Osc.) w/ LPM Enabled 275 µA @ 1.9 V
Sleep	w/ SmaRTClock Crystal 0.600 µA w/o SmaRTClock 0.050 µA	w/ SmaRTClock Crystal 0.600 µA w/o SmaRTClock 0.050 µA	w/ SmaRTClock Crystal 0.600 µA w/ SmaRTClock LFO 0.300 µA w/o SmaRTClock 0.050 µA w/o VBAT Supply Monitor 0.010 µA	w/ SmaRTClock Crystal 0.600 µA w/ SmaRTClock LFO 0.300 µA w/o SmaRTClock 0.050 µA w/o VBAT Supply Monitor 0.010 µA
Note: BLUE refers to power modes only available on C8051F912 and C8051F902 devices.				

- **Minimum Pulse Width**—The minimum pulse width forces the dc-dc converter to use a minimum duty cycle. For low current applications (e.g., MCU in suspend mode), a single pulse provides enough charge to keep the output above the target voltage for several clock pulses. The dc-dc converter remains idle during these clock cycles. This causes an improvement in efficiency due to a reduction in switching losses. Figure 14 shows the effect of minimum pulse width on suspend mode current.
- **Peak Current Limit Threshold—**The peak current through the inductor can be set to 125 or 500 mA. For low current applications, the lower setting should provide the highest efficiency. For high current applications, increasing the peak current should improve efficiency and reduce ripple.
- **VDD/DC+ Sleep Mode Connection**—In Sleep mode, the VDD/DC+ supply can be shorted to VBAT or allowed to float. If the application will be asleep for a short duration (not long enough to discharge the output capacitor), then the VDD/DC+ supply should be left floating. If the VDD/DC+ decoupling capacitor is expected to experience significant discharge while the MCU is asleep, then VDD/DC+ should be internally shorted to VBAT during sleep mode.
- **Low Power Mode**—On C8051F912 and C8051F902 devices, the dc-dc converter supports a low power mode that reduces bias currents. This mode will help improve efficiency when optimal transient response is not required.

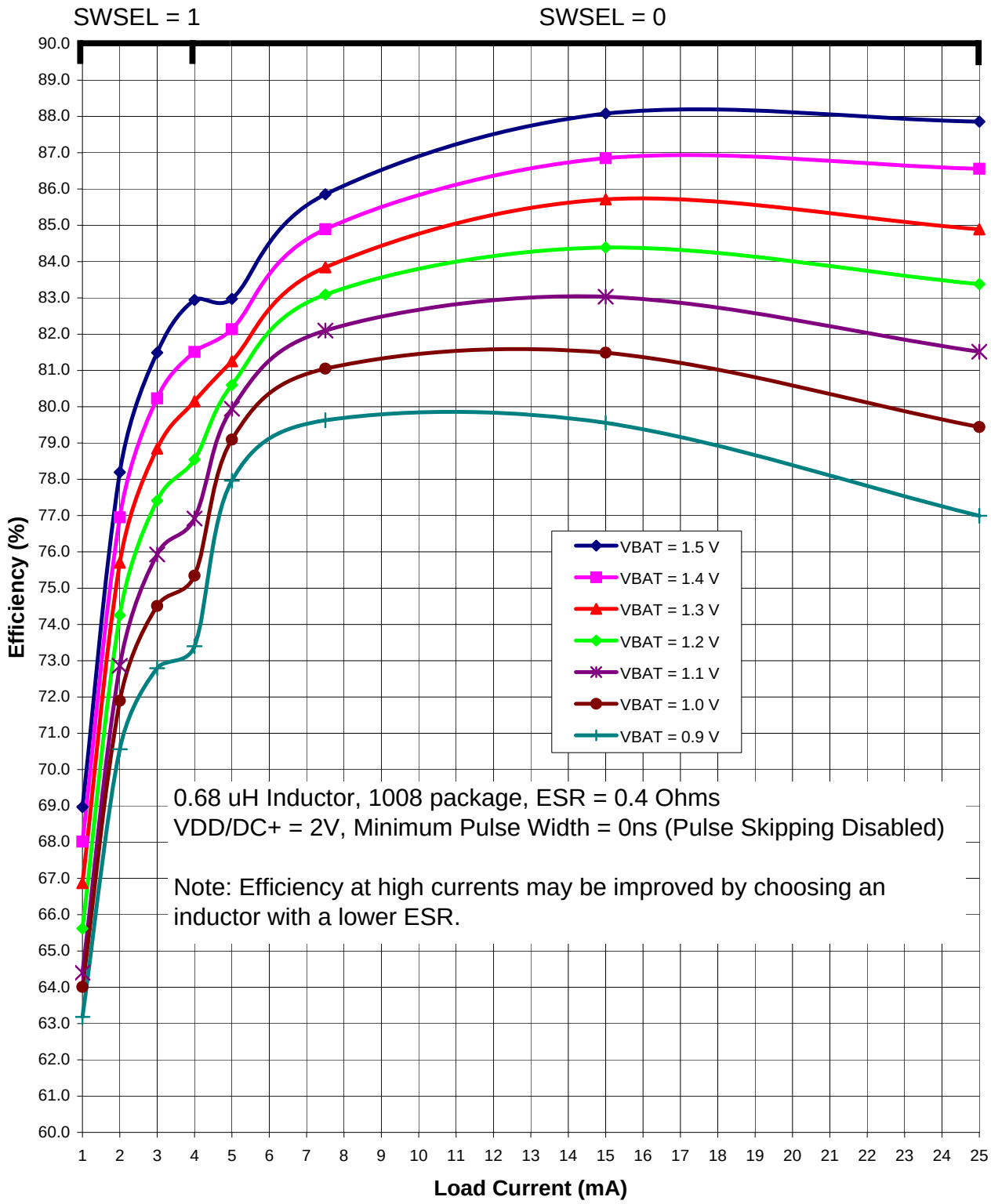

Figure 11. Typical DC-DC Converter Efficiency (High Current, VDD/DC+ = 2 V)

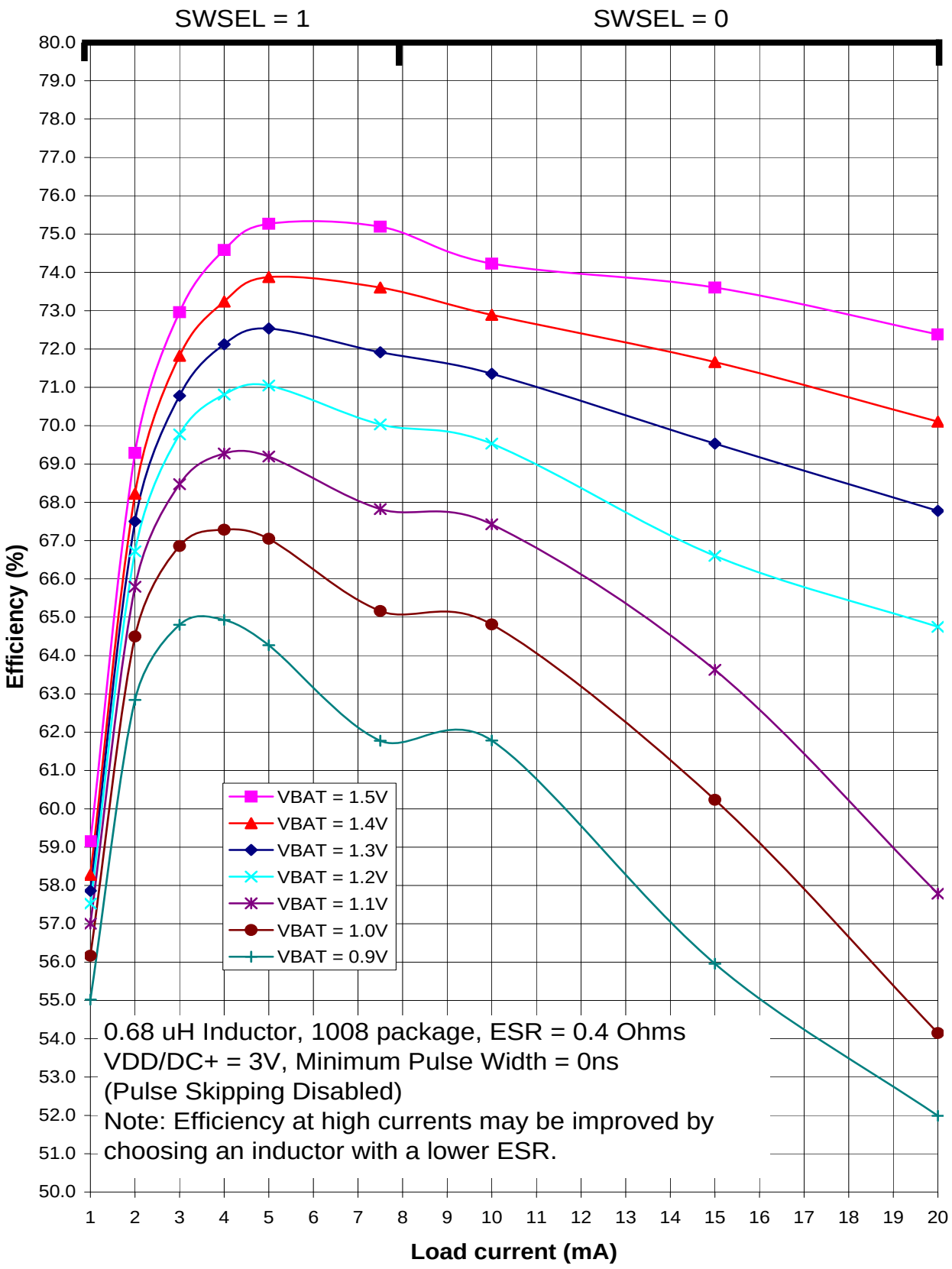

Figure 12. Typical DC-DC Converter Efficiency (High Current, VDD/DC+ = 3 V)

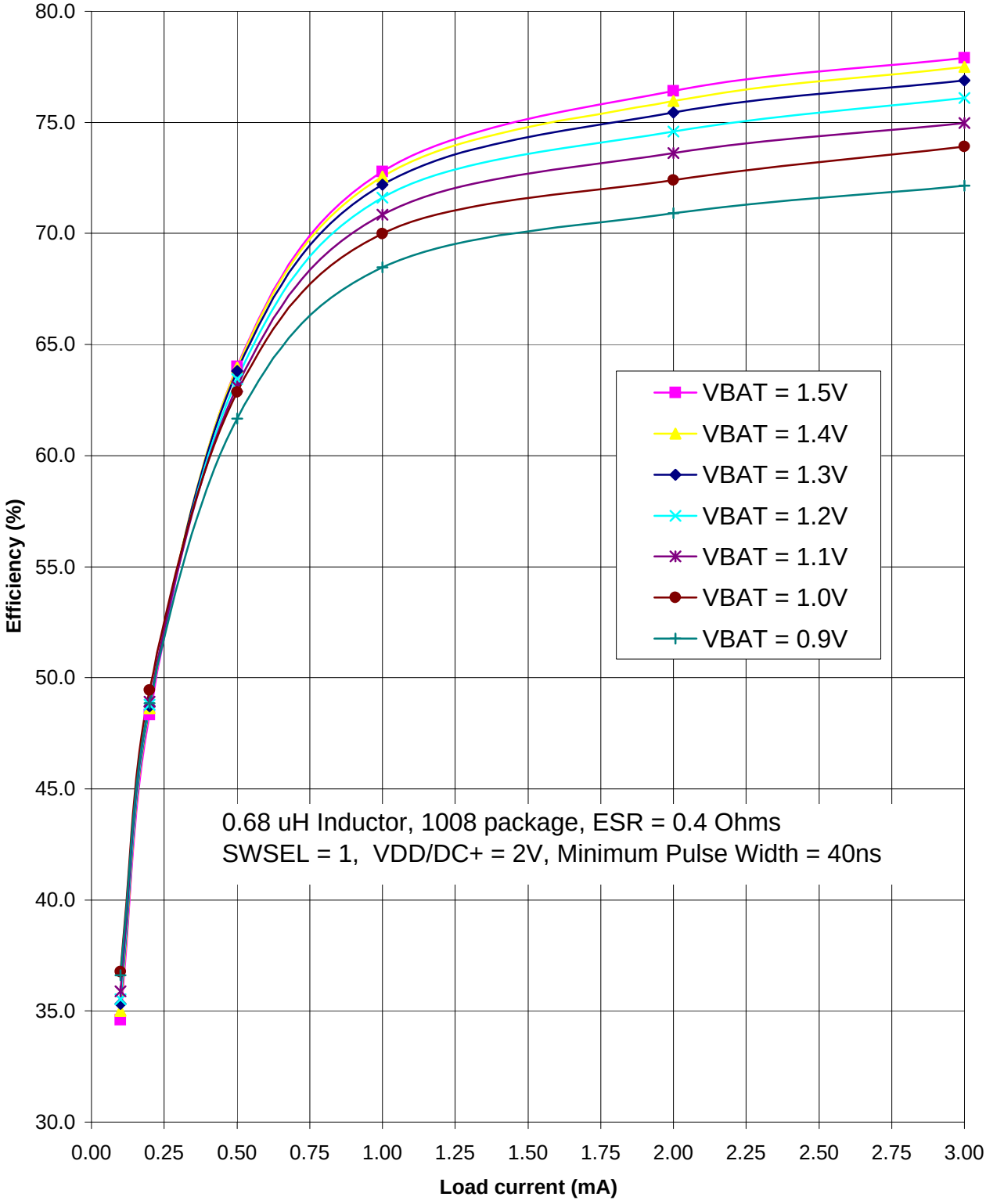

Figure 13. Typical DC-DC Converter Efficiency (Low Current, VDD/DC+ = 2 V)

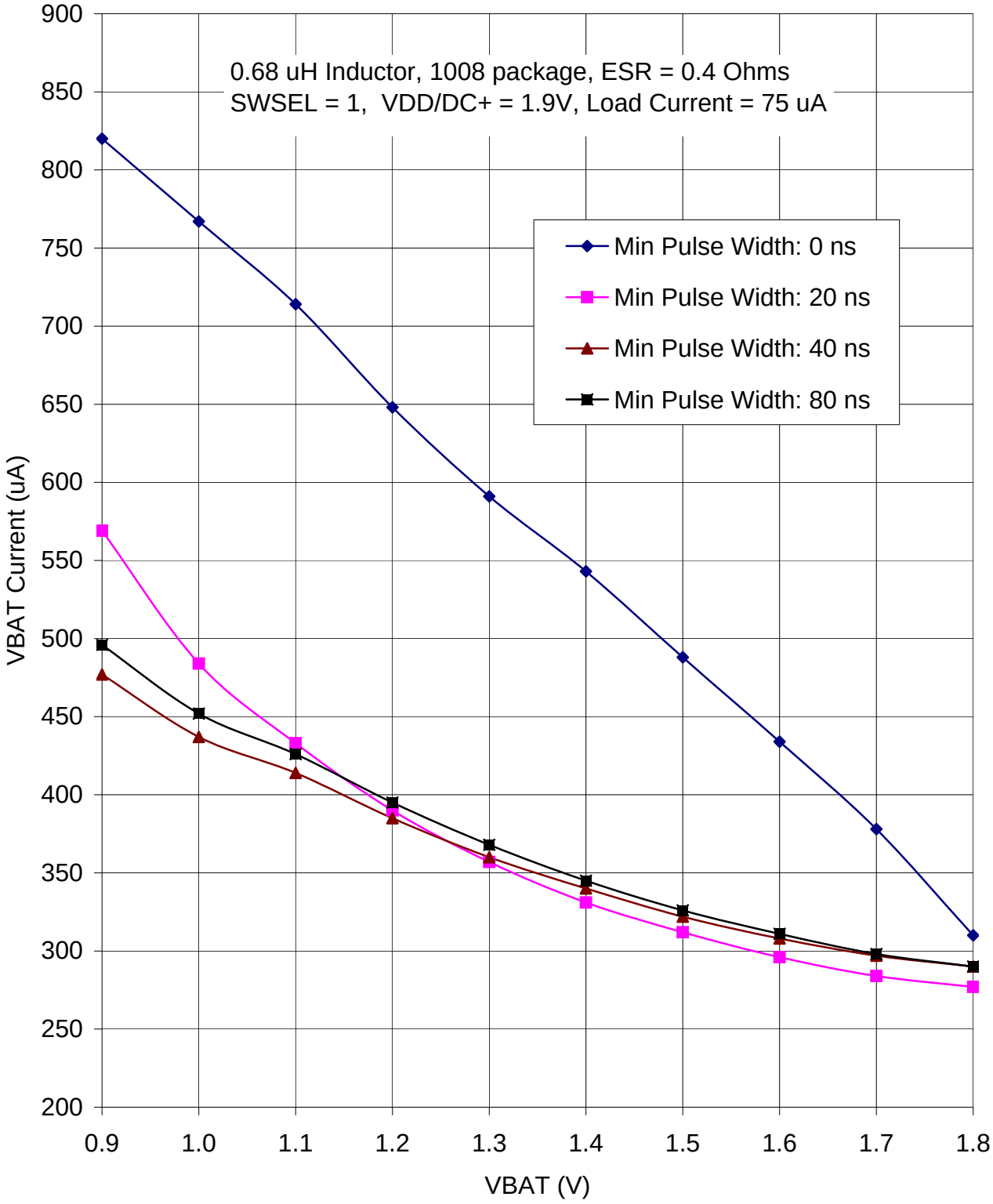

Figure 14. Typical Suspend Mode Supply Current

9. Analog Peripherals

Analog peripherals such as the ADC and Comparators can significantly increase supply current when enabled. Below are a few suggestions that can reduce supply current when using analog peripherals.

- Use the ADC at its maximum sampling rate. This allows the system to take the required samples quickly and turn off the ADC. When oversampling, use Burst Mode to automatically accumulate samples independently of the CPU.
- When using the temperature sensor, enable it immediately before starting a conversion and disable it immediately after the conversion is complete. The typical settling time for the temperature sensor is 1.7 µs, which is the same as the required ADC tracking time.
- Use the on-chip high speed voltage reference. This voltage reference has a settling time of 1.7 µs, which is the same as the required ADC tracking time. The high speed VREF is automatically enabled and disabled by hardware at the beginning and end of an ADC conversion.
- When comparing an analog voltage to 1/2 the supply voltage, use the on-chip resistors supplied with the comparator instead of an external voltage divider. The on-chip voltage divider can be disconnected once the comparison is complete, however, an off chip voltage divider continues to draw a static current as long as the device is powered.
- The comparators have four speed settings. Using the slowest speed setting results in longer response time, but can cause a significant decrease in supply current. This is particularly true when the device is in Sleep mode waiting on an analog signal to cross a certain threshold.
- Disable all analog peripherals (unless used to wake up) before going into Sleep mode.
- The current reference IREF0 in combination with an external resistor can be used to create an ultra low power reference voltage. The IREF0 output can be disabled or controlled to the nearest 1 uA up to 63 uA and to the nearest 8 uA up to 504 uA.

10. Using the Software Example

This application note is bundled with example software to place the device in its most efficient power state for each of its operating modes. The user configured constants are located in the “F9xx_Config.h” header file.

10.1. Configuring the Power Mode

The software defines two active power modes: NORMAL and IDLE, and four inactive power modes: STOP, SUSPEND, RTCSLEEP, and SLEEP. The difference between RTCSLEEP and SLEEP is whether the SmaRTClock oscillator is enabled or disabled in sleep mode. The POWER_MODE constant should be set to one of the power modes defined above.

10.2. Configuring the Clock Source

Any combination of the MCU’s four clock sources and eight clock divide values may be configured using the CLOCK_SELECTION constant. The constants for the clock sources are PRECISION_OSC, LOW_POWER_OSC, SMARTCLOCK_OSC, and EXTERNAL_OSC. There is also an option to enable the missing clock detector.

If the external oscillator is selected as the clock source, the software assumes that a CMOS clock is present at P0.3 and that the EXTERNAL_OSC_FREQ constant contains the proper CMOS clock frequency. When sweeping the system clock frequency, two sweeps need to be performed. The first one is for the range of frequencies between dc and 10 MHz, and the second one is for the range of frequencies between 10 MHz and 25 MHz. The EXTERNAL_OSC_FREQ constant must be updated between the two sweeps in order to result in the optimum operating current.

10.3. Location of the SJMP $ Instruction

When measuring current in normal mode, the most common practice is to insert an infinite loop (SJMP $) to hold CPU execution in one place. Since digital supply current is dependent on the Flash address, placing the infinite loop at the incorrect address can cause supply current to increase beyond its typical value.

The example project includes an “Imeasure.h” and “Imeasure.a51” header and source file that allow current to be properly measured. The Imeasure.h header file allows the user to locate the address of the SJMP $ instruction for minimum, typical, and maximum current. These two files can be added to any project to get an accurate supply current measurement. The function in the assembly file can be called using a function pointer as demonstrated in “F9xx_Main.c”.

DOCUMENT CHANGE LIST

Revision 0.3 to Revision 0.4

- Added support for 'F99x/8x devices.

NOTES:

Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Please visit the Silicon Labs Technical Support web page:
https://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.