

## Features

- Radio System-on-Chip with built-in 8-bit MCU in a single device.
- Operates in the unlicensed worldwide Industrial, Scientific, and Medical (ISM) band (2.400 GHz to 2.483 GHz).
- On Air compatible with second generation radio WirelessUSB™ LP and PRoC LP.
- Pin-to-pin compatible with PRoC LP except the Pin31 and Pin37.

## Intelligent

- M8C based 8-bit CPU, optimized for human interface devices (HID) applications
- 256 bytes of SRAM
- 8 Kbytes of flash memory with EEPROM emulation
- In-system reprogrammable through D+/D- pins
- CPU speed up to 12 MHz
- 16-bit free running timer
- Low power wakeup timer
- 12-bit programmable interval timer with interrupts
- Watchdog timer

## Low Power

- 21 mA operating current (Transmit at -5 dBm)
- Sleep current less than 1  $\mu$ A
- Operating voltage from 2.7 V to 3.6 V DC
- Fast startup and fast channel changes
- Supports coin cell operated applications

## Reliable & Robust

- Receive sensitivity typical -90 dBm
- AutoRate™ - Dynamic Data Rate Reception
  - Enables data reception for any of the supported bit rates automatically.
  - DSSS (250 Kbps), GFSK (1 Mbps)

- Operating temperature from 0 °C to 70 °C
- Closed-loop frequency synthesis for minimal frequency drift

## Simple Development

- Auto transaction sequencer (ATS): MCU can remain in sleep state longer to save power
- Framing, length, CRC16, and Auto ACK
- Separate 16 byte transmit and receive FIFOs
- Receive signal strength indication (RSSI)
- Built-in serial peripheral interface (SPI) control while in Sleep Mode
- Advanced development tools based on Cypress's PSoC® tools
- Flexible I/O
- 2 mA source current on all GPIO pins. Configurable 8 mA or 50 mA/pin current sink on designated pins
- Each GPIO pin supports high impedance inputs, configurable pull up, open drain output, CMOS/TTL inputs, and CMOS output
- Maskable interrupts on all I/O pins

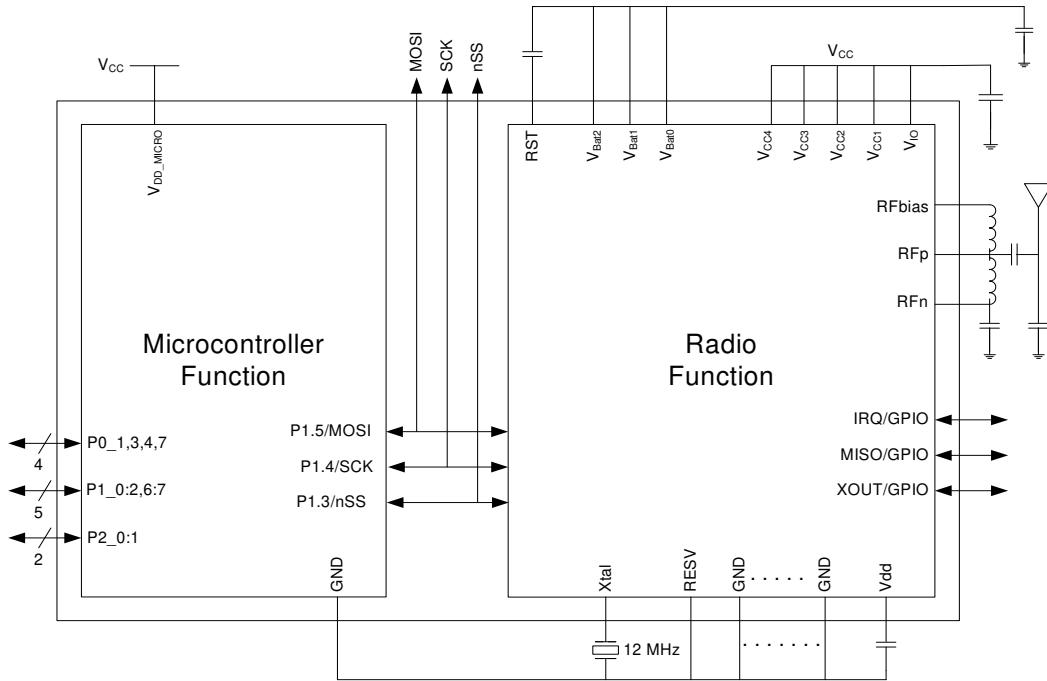
## BOM Savings

- Low external component count
- Small footprint 40-pin QFN (6 mm × 6 mm)
- GPIOs that require no external components
- Operates off a single crystal

## Applications

- Wireless keyboards and mice
- Presentation tools
- Wireless gamepads
- Remote controls
- Toys
- Fitness

### Logic Block Diagram



**Contents**

|  |           |  |           |
|--|-----------|--|-----------|
| <b>Functional Description</b> .....                    | <b>4</b>  | Flash .....  | 18        |
| <b>Functional Overview</b> .....                       | <b>4</b>  | SROM .....   | 18        |
| 2.4 GHz Radio Function .....                           | 4         | SROM Function Descriptions .....                     | 19        |
| Data Transmission Modes .....                          | 4         | <b>Clocking</b> .....                                | <b>22</b> |
| Microcontroller Function .....                         | 4         | SROM Table Read Description .....                    | 23        |
| Backward Compatibility .....                           | 4         | Clock Architecture Description .....                 | 24        |
| <b>Pinouts</b> .....                                   | <b>5</b>  | CPU Clock During Sleep Mode .....                    | 28        |
| <b>Functional Block Overview</b> .....                 | <b>6</b>  | <b>Reset</b> .....                                   | <b>29</b> |
| 2.4 GHz Radio .....                                    | 6         | Power-on Reset .....                                 | 30        |
| Frequency Synthesizer .....                            | 6         | Watchdog Timer Reset .....                           | 30        |
| Baseband and Framer .....                              | 6         | <b>Sleep Mode</b> .....                              | <b>30</b> |
| Packet Buffers and Radio Configuration Registers ..... | 7         | Sleep Sequence .....                                 | 30        |
| Auto Transaction Sequencer (ATS) .....                 | 7         | Low Power in Sleep Mode .....                        | 31        |
| Interrupts .....                                       | 7         | Wakeup Sequence .....                                | 31        |
| Clocks .....   | 8         | <b>Power-on Reset Control</b> .....                  | <b>32</b> |
| GPIO Interface .....                                   | 8         | POR Compare State .....                              | 33        |
| Power-on Reset .....                                   | 8         | ECO Trim Register .....                              | 33        |
| Timers .....   | 8         | <b>General-Purpose I/O Ports</b> .....               | <b>33</b> |
| Power Management .....                                 | 8         | Port Data Registers .....                            | 33        |
| Low Noise Amplifier (LNA) and Received                 |           | GPIO Port Configuration .....                        | 34        |
| Signal Strength Indication (RSSI) .....                | 9         | GPIO Configurations for Low Power Mode .....         | 40        |
| <b>SPI Interface</b> .....                             | <b>9</b>  | Serial Peripheral Interface (SPI) .....              | 41        |
| Three-Wire SPI Interface .....                         | 9         | SPI Data Register .....                              | 42        |
| 4-Wire SPI Interface .....                             | 9         | SPI Configure Register .....                         | 42        |
| SPI Communication and Transactions .....               | 10        | SPI Interface Pins .....                             | 44        |
| SPI I/O Voltage References .....                       | 10        | <b>Timer Registers</b> .....                         | <b>44</b> |
| SPI Connects to External Devices .....                 | 10        | Registers .....                                      | 44        |
| <b>CPU Architecture</b> .....                          | <b>11</b> | <b>Interrupt Controller</b> .....                    | <b>47</b> |
| <b>CPU Registers</b> .....                             | <b>11</b> | Architectural Description .....                      | 47        |
| Flags Register .....                                   | 11        | Interrupt Processing .....                           | 48        |
| Accumulator Register .....                             | 12        | Interrupt Latency .....                              | 48        |
| Index Register .....                                   | 12        | Interrupt Registers .....                            | 48        |
| Stack Pointer Register .....                           | 12        | Microcontroller Function Register Summary .....      | 51        |
| CPU Program Counter High Register .....                | 12        | <b>Radio Function Register Summary</b> .....         | <b>53</b> |
| CPU Program Counter Low Register .....                 | 12        | <b>Absolute Maximum Ratings</b> .....                | <b>54</b> |
| <b>Addressing Modes</b> .....                          | <b>13</b> | <b>DC Characteristics (T = 25 °C)</b> .....          | <b>54</b> |
| Source Immediate .....                                 | 13        | <b>AC Characteristics</b> .....                      | <b>56</b> |
| Source Direct .....                                    | 13        | <b>RF Characteristics</b> .....                      | <b>60</b> |
| Source Indexed .....                                   | 13        | <b>Ordering Information</b> .....                    | <b>62</b> |
| Destination Direct .....                               | 13        | Ordering Code Definitions .....                      | 62        |
| Destination Indexed .....                              | 14        | <b>Package Handling</b> .....                        | <b>63</b> |
| Destination Direct Source Immediate .....              | 14        | <b>Package Diagram</b> .....                         | <b>63</b> |
| Destination Indexed Source Immediate .....             | 14        | <b>Acronyms</b> .....                                | <b>65</b> |
| Destination Direct Source Direct .....                 | 14        | <b>Document Conventions</b> .....                    | <b>65</b> |
| Source Indirect Post Increment .....                   | 15        | <b>Document History Page</b> .....                   | <b>66</b> |
| Destination Indirect Post Increment .....              | 15        | <b>Sales, Solutions, and Legal Information</b> ..... | <b>66</b> |
| <b>Instruction Set Summary</b> .....                   | <b>16</b> | Worldwide Sales and Design Support .....             | 66        |
| <b>Memory Organization</b> .....                       | <b>17</b> | Products .....                                       | 66        |
| Flash Program Memory Organization .....                | 17        | PSoC Solutions .....                                 | 66        |
| Data Memory Organization .....                         | 18        |  |           |

## Functional Description

PRoC LPstar devices are integrated radio and microcontroller functions in the same package to provide a dual-role single-chip solution.

Communication between the microcontroller and the radio is through the radio's SPI interface.

## Functional Overview

The CYRF69303 is a complete Radio System-on-Chip device, providing a complete RF system solution with a single device and a few discrete components. The CYRF69303 is designed to implement low-cost wireless systems operating in the worldwide 2.4 GHz Industrial, Scientific, and Medical (ISM) frequency band (2.400 GHz to 2.4835 GHz).

### 2.4 GHz Radio Function

The SoC contains a 2.4 GHz, 1 Mbps GFSK radio transceiver, packet data buffering, packet framer, DSSS baseband controller, received signal strength indication (RSSI), and SPI interface for data transfer and device configuration.

The radio supports 98 discrete 1 MHz channels (regulations may limit the use of some of these channels in certain jurisdictions).

The baseband performs DSSS spreading/despreading, Start of Packet (SOP), End of Packet (EOP) detection, and CRC16 generation and checking. The baseband may also be configured to automatically transmit Acknowledge (ACK) handshake packets whenever a valid packet is received.

When in receive mode, with packet framing enabled, the device is always ready to receive data transmitted at any of the supported bit rates. This enables the implementation of mixed-rate systems in which different devices use different data rates. This also enables the implementation of dynamic data rate systems that use high data rates at shorter distances or in a low-moderate interference environment or both. It changes to lower data rates at longer distances or in high interference environments or both.

## Data Transmission Modes

The radio supports two different data transmission modes:

- In GFSK mode, data is transmitted at 1 Mbps, without any DSSS
- In DSSS mode eight bits (8DR, 32 chip) are encoded in each derived code symbol transmitted, resulting in effective 250 kbps data rate.

32 chip Pseudo Noise (PN) codes are supported. The two data transmission modes apply to the data after the SOP. In particular the length, data, and CRC16 are all sent in the same mode. In general, DSSS reduce packet error rate in any environment.

## Microcontroller Function

The MCU function is an 8-bit Flash-programmable microcontroller. The instruction set is optimized specifically for HID and a variety of other embedded applications.

The MCU function has up to 8 Kbytes of Flash for user's code and up to 256 bytes of RAM for stack space and user variables.

In addition, the MCU function includes a Watchdog timer, a vectored interrupt controller, a 16-bit Free Running Timer, and 12-bit Programmable Interrupt Timer.

The microcontroller has 15 GPIO pins grouped into multiple ports. With the exception of the four radio function GPIOs, each GPIO port supports high impedance inputs, configurable pull-up, open drain output, CMOS/TTL inputs and CMOS output. Up to two pins support programmable drive strength of up to 50 mA. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Each GPIO port has its own GPIO interrupt vector with the exception of GPIO Port 0. GPIO Port 0 has two dedicated pins that have independent interrupt vectors (P0.3 - P0.4).

The microcontroller features an internal oscillator.

## Backward Compatibility

The CYRF69303 IC is fully interoperable with the main modes of the second generation Cypress radio SoC namely the CYRF6936, CYRF69103 and CYRF69213.

CYRF69303 IC device may transmit data to or receive data from a second generation device, or both.

Pinouts

Figure 1. Pin Diagram

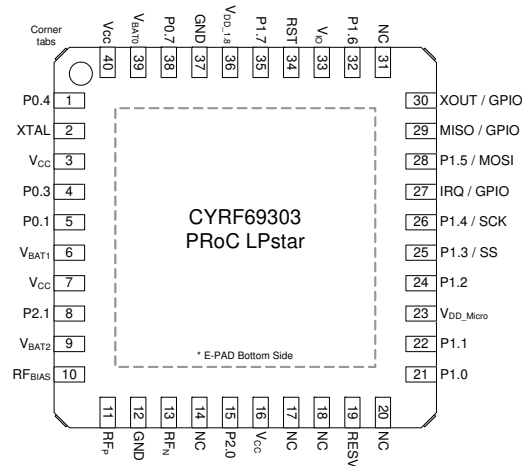


Table 1. Pin Definitions

| Pin            | Name                  | Description   |
|----------------|-----------------------|---|
| 1              | P0.4                  | Individually configured GPIO  |
| 2              | XTAL                  | 12 MHz crystal  |
| 3, 7, 16, 40   | V <sub>CC</sub>       | Connected to 2.7 V to 3.6 V supply, through 0.047 μF bypass C.                          |
| 4              | P0.3                  | Individually configured GPIO  |
| 5              | P0.1                  | Individually configured GPIO  |
| 6              | V <sub>bat1</sub>     | Connect to 2.7 V to 3.6 V power supply, through 47 ohm series/1 μF shunt C              |
| 8              | P2.1                  | GPIO. Port 2 Bit 1  |
| 9              | V <sub>bat2</sub>     | Connected to 2.7 V to 3.6 V main power supply, through 0.047 μF bypass C                |
| 10             | RF <sub>bias</sub>    | RF pin voltage reference  |
| 11             | RF <sub>p</sub>       | Differential RF to or from antenna  |
| 12             | GND                   | GND   |
| 13             | RF <sub>n</sub>       | Differential RF to or from antenna  |
| 14, 17, 18, 20 | NC                    |   |
| 15             | P2.0                  | GPIO  |
| 19             | RESV                  | Reserved. Must connect to GND   |
| 21             | P1.0                  | GPIO  |
| 22             | P1.1                  | GPIO  |
| 23             | V <sub>DD_micro</sub> | MCU supply connected to V <sub>CC</sub> , max CPU 12 MHz                                |
| 24             | P1.2                  | GPIO  |
| 25             | P1.3 / nSS            | Slave Select  |
| 26             | P1.4 / SCK            | SPI Clock   |
| 27             | IRQ                   | Radio Function Interrupt output, configure High, Low or as Radio GPIO                   |
| 28             | P1.5 / MOSI           | MOSI pin from microcontroller function to radio function                                |
| 29             | MISO                  | 3-wire SPI mode configured as Radio GPIO. In 4-wire SPI mode sends data to MCU function |
| 30             | XOUT                  | Buffered CLK or Radio GPIO  |
| 31             | NC                    | Must be floating  |

**Table 1. Pin Definitions** (continued)

| Pin | Name               | Description  |
|-----|--------------------|--|
| 32  | P1.6               | GPIO   |
| 33  | V <sub>IO</sub>    | 2.7 V to 3.6 V to main power supply rail for Radio I/O   |
| 34  | RST                | Radio Reset. Connected to V <sub>CC</sub> with 0.47 μF. Must have a RST=HIGH event the very first time power is applied to the radio otherwise the state of the radio control registers is unknown |
| 35  | P1.7               | GPIO   |
| 36  | V <sub>DD1.8</sub> | Regulated logic bypass. Connected to 0.47 μF to GND  |
| 37  | GND                | Must be connected to ground  |
| 38  | P0.7               | GPIO   |
| 39  | V <sub>bat0</sub>  | Connected to 2.7 V to 3.6 V main power supply, through 0.047 μF bypass C   |
| 41  | E-pad              | Must be connected to ground  |
| 42  | Corner Tabs        | Do Not connect corner tabs   |

## Functional Block Overview

All the blocks that make up the PProC LPstar are presented in this section.

### 2.4 GHz Radio

The radio transceiver is a dual conversion low IF architecture optimized for power and range/robustness. The radio employs channel matched filters to achieve high performance in the presence of interference. An integrated Power Amplifier (PA) provides up to 0 dBm transmit power, with an output power control range of 30 dB in six steps. The supply current of the device is reduced as the RF output power is reduced.

**Table 2. Internal PA Output Power Step Table**

| PA Setting | Typical Output Power (dBm) |
|------------|----------------------------|
| 6          | 0                          |
| 5          | -5                         |
| 4          | -10                        |
| 3          | -15                        |
| 2          | -20                        |
| 1          | -25                        |
| 0          | -30                        |

### Frequency Synthesizer

Before transmission or reception may commence, it is necessary for the frequency synthesizer to settle. The settling time varies depending on channel; 25 fast channels are provided with a maximum settling time of 100 μs.

The “fast channels” (<100 μs settling time) are every third frequency, starting at 2400 MHz up to and including 2472 MHz (that is, 0,3,6,9.....69 and 72).

### Baseband and Framing

The baseband and framing blocks provide the DSSS encoding and decoding, SOP generation and reception and CRC16 generation and checking, and EOP detection and length field.

#### Data Transmission Modes and Data Rates

The SoC supports two different data transmission modes:

- In GFSK mode, data is transmitted at 1 Mbps, without any DSSS.
- In DSSS mode eight bits (8DR, 32 chip) are encoded in each derived code symbol transmitted, resulting in effective 250 kbps data rate.

32 chip Pseudo Noise (PN) codes are supported. The two data transmission modes apply to the data after the SOP. In particular the length, data, and CRC16 are all sent in the same mode. In general, DSSS reduce packet error rate in any environment.

#### Link Layer Modes

#### SOP

Packets begin with a two-symbol SoP marker. If framing is disabled then an SOP event is inferred whenever two successive correlations are detected. The SOP\_CODE\_ADR code used for the SOP is different from that used for the “body” of the packet, and if desired may be a different length. SOP must be configured to be the same length on both sides of the link.

#### Length

Length field is the first eight bits after the SOP symbol, and is transmitted at the payload data rate. An EoP condition is inferred after reception of the number of bytes defined in the length field, plus two bytes for the CRC16.

**CRC16**

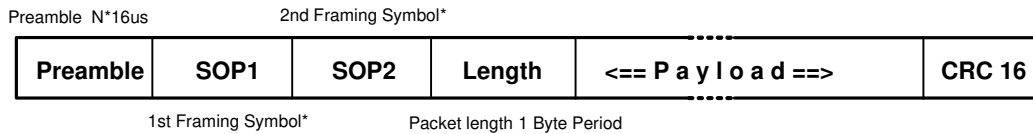
The device may be configured to append a 16-bit CRC16 to each packet. The CRC16 uses the USB CRC polynomial with the added programmability of the seed. If enabled, the receiver verifies the calculated CRC16 for the payload data against the received value in the CRC16 field. The starting value for the CRC16 calculation is configurable, and the CRC16 transmitted may be calculated using either the loaded seed value or a zero seed; the received data CRC16 is checked against both the configured and zero CRC16 seeds.

CRC16 detects the following errors:

- Any one bit in error
- Any two bits in error (no matter how far apart, which column, and so on)
- Any odd number of bits in error (no matter where they are)
- An error burst as wide as the checksum itself

Figure 2 shows an example packet with SOP, CRC16 and lengths fields enabled.

**Figure 2. Example Default Packet Format**



\*Note: 32 us

**Packet Buffers and Radio Configuration Registers**

Packet data and configuration registers are accessed through the SPI interface. All configuration registers are directly addressed through the address field in the SPI packet. Configuration registers are provided to allow configuration of DSSS PN codes, data rate, operating mode, interrupt masks, interrupt status, and others.

*Packet Buffers*

All data transmission and reception use the 16-byte packet buffers: one for transmission and one for reception.

The transmit buffer allows a complete packet of up to 16 bytes of payload data to be loaded in one burst SPI transaction, and then transmitted with no further MCU intervention. Similarly, the receive buffer allows an entire packet of payload data up to 16 bytes to be received with no firmware intervention required until packet reception is complete.

The CYRF69303 IC supports packet length of up to 40 bytes; interrupts are provided to allow an MCU to use the transmit and receive buffers as FIFOs. When transmitting a packet longer than 16 bytes, the MCU can load 16 bytes initially, and add further bytes to the transmit buffer as transmission of data creates space in the buffer. Similarly, when receiving packets longer than 16 bytes, the MCU must fetch received data from the FIFO periodically during packet reception to prevent it from overflowing.

**Auto Transaction Sequencer (ATS)**

The CYRF69303 IC provides automated support for transmission and reception of acknowledged data packets.

When transmitting a data packet, the device automatically starts the crystal and synthesizer, enters transmit mode, transmits the packet in the transmit buffer, and then automatically switches to receive mode and waits for a handshake packet—and then

automatically reverts to sleep mode or idle mode when either an ACK packet is received, or a time out period expires.

Similarly, when receiving in transaction mode, the device waits in receive mode for a valid packet to be received, then automatically transitions to transmit mode, transmits an ACK packet, and then switches back to receive mode to await the next packet. The contents of the packet buffers are not affected by the transmission or reception of ACK packets.

In each case, the entire packet transaction takes place without any need for MCU firmware action; to transmit data the MCU simply needs to load the data packet to be transmitted, set the length, and set the TX GO bit. Similarly, when receiving packets in transaction mode, firmware simply needs to retrieve the fully received packet in response to an interrupt request indicating reception of a packet.

**Interrupts**

The radio function provides an interrupt (IRQ) output, which is configurable to indicate the occurrence of various different events. The IRQ pin may be programmed to be either active high or active low, and be either a CMOS or open drain output.

The radio function features three sets of interrupts: transmit, receive, and system interrupts. These interrupts all share a single pin (IRQ), but can be independently enabled/disabled. In transmit mode, all receive interrupts are automatically disabled, and in receive mode all transmit interrupts are automatically disabled. However, the contents of the enable registers are preserved when switching between transmit and receive modes.

If more than one radio interrupt is enabled at any time, it is necessary to read the relevant status register to determine which event caused the IRQ pin to assert. Even when an interrupt source is disabled, the status of the condition that would otherwise cause an interrupt can be determined by reading the appropriate status register. It is therefore possible to use the devices without making use of the IRQ pin by polling the status register(s) to wait for an event, rather than using the IRQ pin.



**Clocks**

A 12 MHz crystal (30 ppm or better) is directly connected between XTAL and GND without the need for external capacitors. A digital clock out function is provided, with selectable output frequencies of 0.75, 1.5, 3, 6, or 12 MHz. This output may be used to clock an external microcontroller (MCU) or ASIC. This output is enabled by default, but may be disabled.

The requirements for the crystal to be directly connected to XTAL pin and GND are:

- Nominal Frequency: 12 MHz
- Operating Mode: Fundamental Mode
- Resonance Mode: Parallel Resonant
- Frequency Initial Stability:  $\pm 30$  ppm
- Series Resistance:  $\leq 60$  ohms
- Load Capacitance: 10 pF
- Drive Level: 100  $\mu$ W

The MCU function features an internal oscillator. The clock generator provides the 12 MHz and 24 MHz clocks that remain internal to the microcontroller.

**GPIO Interface**

The MCU function features up to 15 general-purpose I/O (GPIO) pins. The I/O pins are grouped into three ports (Port 0 to 2). The pins on Port 0 and Port 1 may each be configured individually while the pins on Port 2 may only be configured as a group. Each GPIO port supports high-impedance inputs, configurable pull-up, open drain output, CMOS/TTL inputs, and CMOS output with up to two pins that support programmable drive strength of up to 50 mA sink current. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Each GPIO port has its own GPIO interrupt vector with the exception of GPIO Port 0. GPIO Port 0 has three dedicated pins that have independent interrupt vectors (P0.1, P0.3–P0.4).

**Power-on Reset**

The power-on reset (POR) circuit detects logic when power is applied to the device, resets the logic to a known state, and begins executing instructions at Flash address 0x0000. When power falls below a programmable trip voltage, it generates reset or may be configured to generate interrupt.

**Timers**

The free-running 16-bit timer provides two interrupt sources: the programmable interval timer with 1  $\mu$ s resolution and the 1.024 ms outputs. The timer can be used to measure the duration of an event under firmware control by reading the timer

at the start and at the end of an event, then calculating the difference between the two values.

**Power Management**

The operating voltage of the device is 2.7 V to 3.6 V DC, which is applied to  $V_{CC}$  and  $V_{BAT}$  pins. The device can be shut down to a fully static sleep mode by writing to the FRC END = 1 and END STATE = 000 bits in the XACT\_CFG\_ADR register over the SPI interface. The device enters sleep mode within 35  $\mu$ s after the last SCK positive edge at the end of this SPI transaction. Alternatively, the device may be configured to automatically enter sleep mode after completing the packet transmission or reception. When in sleep mode, the on-chip oscillator is stopped, but the SPI interface remains functional. The device wakes from sleep mode automatically when the device is commanded to enter transmit or receive mode. When resuming from sleep mode, there is a short delay while the oscillator restarts. The device can be configured to assert the IRQ pin when the oscillator has stabilized.

The following Figure 3 is an example of the circuit used when the supply voltage is always above 2.7 V. This could be three 1.5 V battery cells in series along with a linear regulator, or some similar power source. Figure 4 shows an example of using an external boost to supply power to the device.

**Figure 3. Example Circuit - Linear Regulator**

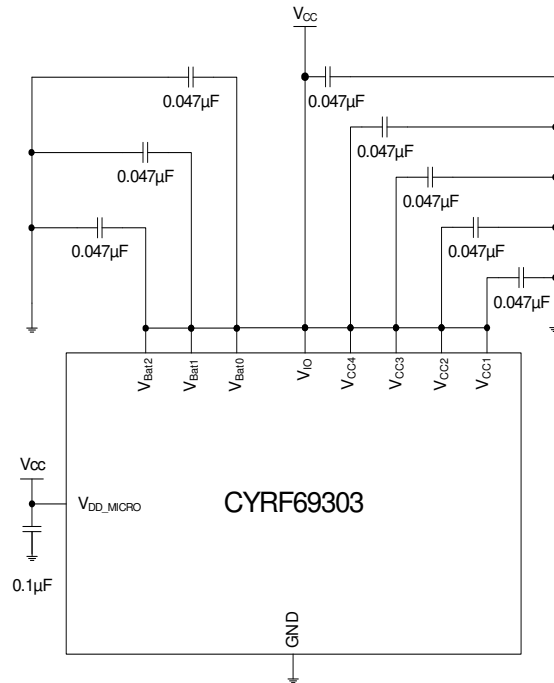
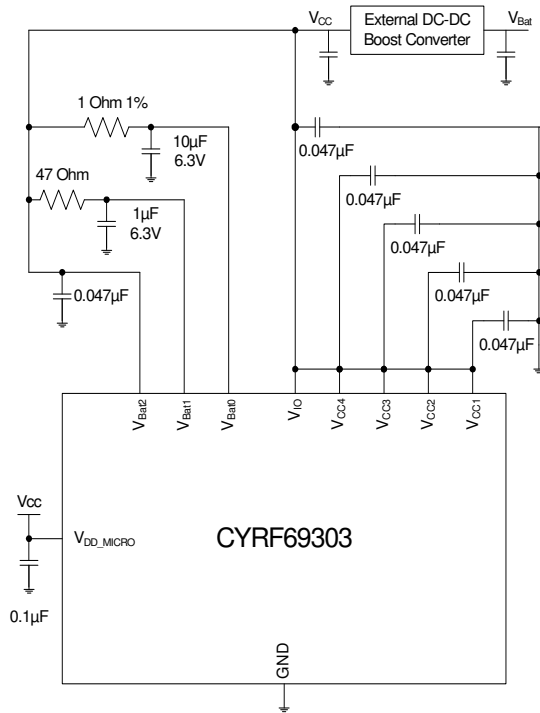




Figure 4. Example Circuit - External Boost Converter



**Low Noise Amplifier (LNA) and Received Signal Strength Indication (RSSI)**

The gain of the receiver may be controlled directly by clearing the AGC EN bit and writing to the low noise amplifier (LNA) bit of the RX\_CFG\_ADR register. When the LNA bit is cleared, the receiver gain is reduced by approximately 20 dB, allowing accurate reception of very strong received signals (for example when operating a receiver very close to the transmitter). An additional 20 dB of receiver attenuation can be added by setting the Attenuation (ATT) bit; this allows data reception to be limited to devices at very short ranges. Disabling AGC and enabling LNA is recommended unless receiving from a device using external PA.

The RSSI register returns the relative signal strength of the on-channel signal power.

When receiving, the device may be configured to automatically measure and store the relative strength of the signal being received as a 5-bit value. When enabled, an RSSI reading is taken and may be read through the SPI interface. An RSSI reading is taken automatically when the start of a packet is detected. In addition, a new RSSI reading is taken every time the previous reading is read from the RSSI register, allowing the background RF energy level on any channel to be easily measured when RSSI is read when no signal is being received. A new reading can occur as fast as once every 12 µs.

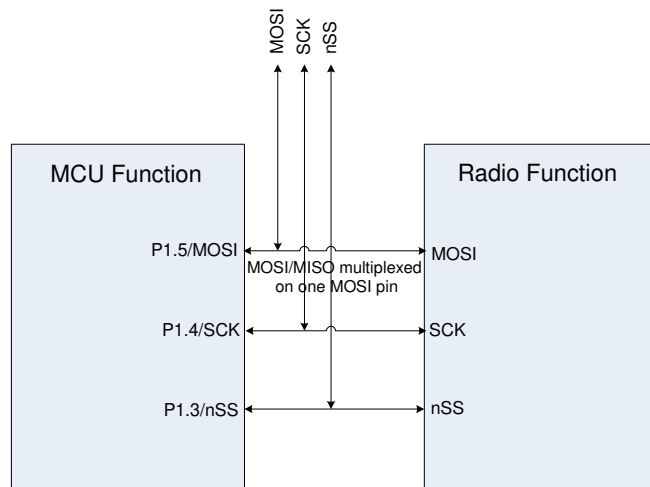
**SPI Interface**

The SPI interface between the MCU function and the radio function is a 3-wire SPI Interface. The three pins are Master Out Slave In (MOSI), Serial Clock (SCK), and Slave Select (SS). There is an alternate 4-wire MISO Interface that requires the connection of two external pins. The SPI interface is controlled by configuring the SPI Configure Register. (SPICR Addr: 0x3D).

**Three-Wire SPI Interface**

The radio function receives a clock from the MCU function on the SCK pin. The MOSI pin is multiplexed with the MISO pin. Bidirectional data transfer takes place between the MCU function and the radio function through this multiplexed MOSI pin. When using this mode the user firmware must ensure that the MOSI pin on the MCU function is in a high impedance state, except when the MCU is actively transmitting data. Firmware must also control the direction of data flow and switch directions between MCU function and radio function by setting the SWAP bit [Bit 7] of the SPI Configure Register. The SS pin is asserted before initiating a data transfer between the MCU function and the radio function. The IRQ function may be optionally multiplexed with the MOSI pin; when this option is enabled the IRQ function is not available while the SS pin is low. When using this configuration, user firmware must ensure that the MOSI function on MCU function is in a high-impedance state whenever SS is high.

Figure 5. 3-Wire SPI Mode

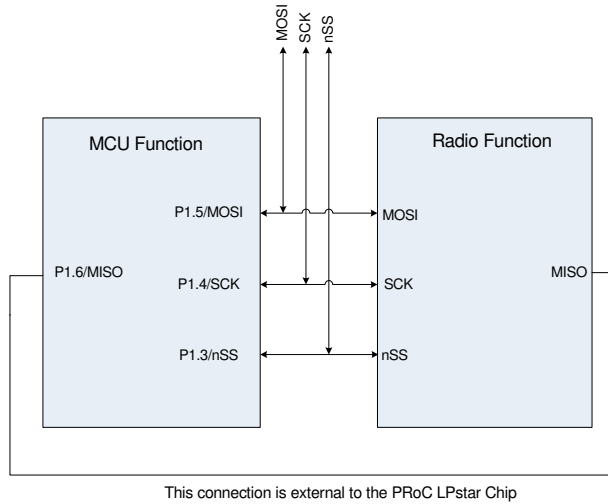


**4-Wire SPI Interface**

The 4-wire SPI communications interface consists of MOSI, MISO, SCK, and SS.

The device receives SCK from the MCU function on the SCK pin. Data from the MCU function is shifted in on the MOSI pin. Data to the MCU function is shifted out on the MISO pin. The active low SS pin must be asserted for the two functions to communicate. The IRQ function may be optionally multiplexed with the MOSI pin; when this option is enabled the IRQ function is not available while the SS pin is low. When using this configuration, user firmware must ensure that the MOSI function on MCU function is in a high-impedance state whenever SS is high.

Figure 6. 4-Wire SPI Mode



The INC bit helps to read or write consecutive bytes from contiguous memory locations in a single burst mode operation.

If Slave Select is asserted and INC = 1, then the master MCU function reads a byte from the radio, the address is incremented by a byte location, and then the byte at that location is read, and so on.

If Slave Select is asserted and INC = 0, then the MCU function reads/writes the bytes in the same register in burst mode, but if it is a register file then it reads/writes the bytes in that register file.

The SPI interface between the radio function and the MCU is not dependent on the internal 12 MHz oscillator of the radio. Therefore, radio function registers can be read from or written into while the radio is in sleep mode.

**SPI I/O Voltage References**

The SPI interfaces between MCU function and the radio and the IRQ and RST have a separate voltage reference  $V_{IO}$ . For CYRF69303  $V_{IO}$  is normally set to  $V_{CC}$ .

**SPI Connects to External Devices**

The three SPI wires, MOSI, SCK, and SS are also drawn out of the package as external pins to allow the user to interface their own external devices (such as optical sensors and others) through SPI. The radio function also has its own SPI wires MISO and IRQ, which can be used to send data back to the MCU function or send an interrupt request to the MCU function. They can also be configured as GPIO pins.

**SPI Communication and Transactions**

The SPI transactions can be single byte or multi-byte. The MCU function initiates a data transfer through a Command/Address byte. The following bytes are data bytes. The SPI transaction format is shown in Figure 5.

The DIR bit specifies the direction of data transfer. 0 = Master reads from slave. 1 = Master writes to slave.

Table 3. SPI Transaction Format

|          | Byte 1 |     |         | Byte 1+N |
|----------|--------|-----|---------|----------|
| Bit #    | 7      | 6   | [5:0]   | [7:0]    |
| Bit Name | DIR    | INC | Address | Data     |

## CPU Architecture

This family of microcontrollers is based on a high-performance, 8-bit, Harvard architecture microprocessor. Five registers control the primary operation of the CPU core. These registers are affected by various instructions, but are not directly accessible through the register space by the user.

**Table 4. CPU Registers and Register Name**

| Register        | Register Name |
|-----------------|---------------|
| Flags           | CPU_F         |
| Program Counter | CPU_PC        |
| Accumulator     | CPU_A         |
| Stack Pointer   | CPU_SP        |
| Index           | CPU_X         |

The 16-bit Program Counter Register (CPU\_PC) allows for direct addressing of the full eight Kbytes of program memory space.

## CPU Registers

### Flags Register

The Flags Register can only be set or reset with logical instruction.

**Table 5. CPU Flags Register (CPU\_F) [R/W]**

| Bit #      | 7        | 6 | 5 | 4   | 3     | 2     | 1    | 0         |
|------------|----------|---|---|-----|-------|-------|------|-----------|
| Field      | Reserved |   |   | XIO | Super | Carry | Zero | Global IE |
| Read/Write | –        | – | – | R/W | R     | RW    | RW   | RW        |
| Default    | 0        | 0 | 0 | 0   | 0     | 0     | 1    | 0         |

**Bits 7:5** Reserved

**Bit 4** XIO  
Set by the user to select between the register banks.  
0 = Bank 0  
1 = Bank 1

**Bit 3** Super  
Indicates whether the CPU is executing user code or Supervisor Code (This code cannot be accessed directly by the user).  
0 = User Code  
1 = Supervisor Code

**Bit 2** Carry  
Set by CPU to indicate whether there has been a carry in the previous logical/arithmetic operation.  
0 = No Carry  
1 = Carry

**Bit 1** Zero  
Set by CPU to indicate whether there has been a zero result in the previous logical/arithmetic operation.  
0 = Not Equal to Zero  
1 = Equal to Zero

**Bit 0** Global IE  
Determines whether all interrupts are enabled or disabled.  
0 = Disabled  
1 = Enabled

**Note** This register is readable with explicit address 0xF7. The *OR F, expr* and *AND F, expr* must be used to set and clear the CPU\_F bits.

The Accumulator Register (CPU\_A) is the general-purpose register that holds the results of instructions that specify any of the source addressing modes.

The Index Register (CPU\_X) holds an offset value that is used in the indexed addressing modes. Typically, this is used to address a block of data within the data memory space.

The Stack Pointer Register (CPU\_SP) holds the address of the current top-of-stack in the data memory space. It is affected by the PUSH, POP, LCALL, CALL, RETI, and RET instructions, which manage the software stack. It can also be affected by the SWAP and ADD instructions.

The Flag Register (CPU\_F) has three status bits: Zero Flag bit [1]; Carry Flag bit [2]; Supervisory State bit [3]. The Global Interrupt Enable bit [0] is used to globally enable or disable interrupts. The user cannot manipulate the Supervisory State status bit [3]. The flags are affected by arithmetic, logic, and shift operations. The manner in which each flag is changed is dependent upon the instruction being executed (for example, AND, OR, XOR). See [Table 21 on page 16](#).

**Accumulator Register**
**Table 6. CPU Accumulator Register (CPU\_A)**

| Bit #      | 7                     | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|-----------------------|---|---|---|---|---|---|---|
| Field      | CPU Accumulator [7:0] |   |   |   |   |   |   |   |
| Read/Write | –                     | – | – | – | – | – | – | – |
| Default    | 0                     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 CPU Accumulator [7:0]**

8-bit data value holds the result of any logical/arithmetic instruction that uses a source addressing mode.

**Index Register**
**Table 7. CPU X Register (CPU\_X)**

| Bit #      | 7       | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|---------|---|---|---|---|---|---|---|
| Field      | X [7:0] |   |   |   |   |   |   |   |
| Read/Write | –       | – | – | – | – | – | – | – |
| Default    | 0       | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 X [7:0]**

8-bit data value holds an index for any instruction that uses an indexed addressing mode.

**Stack Pointer Register**
**Table 8. CPU Stack Pointer Register (CPU\_SP)**

| Bit #      | 7                   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|---------------------|---|---|---|---|---|---|---|
| Field      | Stack Pointer [7:0] |   |   |   |   |   |   |   |
| Read/Write | –                   | – | – | – | – | – | – | – |
| Default    | 0                   | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 Stack Pointer [7:0]**

8-bit data value holds a pointer to the current top-of-stack.

**CPU Program Counter High Register**
**Table 9. CPU Program Counter High Register (CPU\_PCH)**

| Bit #      | 7                      | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|------------------------|---|---|---|---|---|---|---|
| Field      | Program Counter [15:8] |   |   |   |   |   |   |   |
| Read/Write | –                      | – | – | – | – | – | – | – |
| Default    | 0                      | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 Program Counter [15:8]**

8-bit data value holds the higher byte of the program counter.

**CPU Program Counter Low Register**
**Table 10. CPU Program Counter Low Register (CPU\_PCL)**

| Bit #      | 7                     | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|-----------------------|---|---|---|---|---|---|---|
| Field      | Program Counter [7:0] |   |   |   |   |   |   |   |
| Read/Write | –                     | – | – | – | – | – | – | – |
| Default    | 0                     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7:0 Program Counter [7:0]**

8-bit data value holds the lower byte of the program counter.

## Addressing Modes

Examples of the different addressing modes are discussed in this section and example code is given.

### Source Immediate

The result of an instruction using this addressing mode is placed in the A register, the F register, the SP register, or the X register, which is specified as part of the instruction opcode. Operand 1 is an immediate value that serves as a source for the instruction. Arithmetic instructions require two sources. Instructions using this addressing mode are two bytes in length.

**Table 11. Source Immediate**

| Opcode      | Operand 1       |
|-------------|-----------------|
| Instruction | Immediate Value |

#### Examples

- ADD A, 7 In this case, the immediate value of 7 is added with the Accumulator, and the result is placed in the Accumulator.
- MOV X, 8 In this case, the immediate value of 8 is moved to the X register.
- AND F, 9 In this case, the immediate value of 9 is logically ANDed with the F register and the result is placed in the F register.

### Source Direct

The result of an instruction using this addressing mode is placed in either the A register or the X register, which is specified as part of the instruction opcode. Operand 1 is an address that points to a location in either the RAM memory space or the register space that is the source for the instruction. Arithmetic instructions require two sources; the second source is the A register or X register specified in the opcode. Instructions using this addressing mode are two bytes in length.

**Table 12. Source Direct**

| Opcode      | Operand 1      |
|-------------|----------------|
| Instruction | Source Address |

#### Examples

- ADD A, [7] In this case, the value in the RAM memory location at address 7 is added with the Accumulator, and the result is placed in the Accumulator.
- MOV X, REG[8] In this case, the value in the register space at address 8 is moved to the X register.

### Source Indexed

The result of an instruction using this addressing mode is placed in either the A register or the X register, which is specified as part of the instruction opcode. Operand 1 is added to the X register forming an address that points to a location in either the RAM memory space or the register space that is the source for the instruction. Arithmetic instructions require two sources; the second source is the A register or X register specified in the opcode. Instructions using this addressing mode are two bytes in length.

**Table 13. Source Indexed**

| Opcode      | Operand 1    |
|-------------|--------------|
| Instruction | Source Index |

#### Examples

- ADD A, [X+7] In this case, the value in the memory location at address X + 7 is added with the Accumulator, and the result is placed in the Accumulator.
- MOV X, REG[X+8] In this case, the value in the register space at address X + 8 is moved to the X register.

### Destination Direct

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is an address that points to the location of the result. The source for the instruction is either the A register or the X register, which is specified as part of the instruction opcode. Arithmetic instructions require two sources; the second source is the location specified by Operand 1. Instructions using this addressing mode are two bytes in length.

**Table 14. Destination Direct**

| Opcode      | Operand 1           |
|-------------|---------------------|
| Instruction | Destination Address |

#### Examples

- ADD [7], A In this case, the value in the memory location at address 7 is added with the Accumulator, and the result is placed in the memory location at address 7. The Accumulator is unchanged.
- MOV REG[8], A In this case, the Accumulator is moved to the register space location at address 8. The Accumulator is unchanged.

### Destination Indexed

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is added to the X register forming the address that points to the location of the result. The source for the instruction is the A register. Arithmetic instructions require two sources; the second source is the location specified by Operand 1 added with the X register. Instructions using this addressing mode are two bytes in length.

**Table 15. Destination Indexed**

| Opcode      | Operand 1         |
|-------------|-------------------|
| Instruction | Destination Index |

#### Example

ADD [X+7], A In this case, the value in the memory location at address X+7 is added with the Accumulator, and the result is placed in the memory location at address x+7. The Accumulator is unchanged.

### Destination Direct Source Immediate

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is the address of the result. The source for the instruction is Operand 2, which is an immediate value. Arithmetic instructions require two sources; the second source is the location specified by Operand 1. Instructions using this addressing mode are three bytes in length.

**Table 16. Destination Direct Source Immediate**

| Opcode      | Operand 1           | Operand 2       |
|-------------|---------------------|-----------------|
| Instruction | Destination Address | Immediate Value |

#### Examples

ADD [7], 5 In this case, value in the memory location at address 7 is added to the immediate value of 5, and the result is placed in the memory location at address 7.

MOV REG[8], 6 In this case, the immediate value of 6 is moved into the register space location at address 8.

### Destination Indexed Source Immediate

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is added to the X register to form the address of the result. The source for the instruction is Operand 2, which is an immediate value. Arithmetic instructions require two sources; the second source is the location specified by Operand 1 added with the X register. Instructions using this addressing mode are three bytes in length.

**Table 17. Destination Indexed Source Immediate**

| Opcode      | Operand 1         | Operand 2       |
|-------------|-------------------|-----------------|
| Instruction | Destination Index | Immediate Value |

#### Examples

ADD [X+7], 5 In this case, the value in the memory location at address X+7 is added with the immediate value of 5 and the result is placed in the memory location at address X+7.

MOV REG[X+8], 6 In this case, the immediate value of 6 is moved into the location in the register space at address X+8.

### Destination Direct Source Direct

The result of an instruction using this addressing mode is placed within the RAM memory. Operand 1 is the address of the result. Operand 2 is an address that points to a location in the RAM memory that is the source for the instruction. This addressing mode is only valid on the MOV instruction. The instruction using this addressing mode is three bytes in length.

**Table 18. Destination Direct Source Direct**

| Opcode      | Operand 1           | Operand 2      |
|-------------|---------------------|----------------|
| Instruction | Destination Address | Source Address |

#### Example

MOV [7], [8] In this case, the value in the memory location at address 8 is moved to the memory location at address 7.



**Source Indirect Post Increment**

The result of an instruction using this addressing mode is placed in the Accumulator. Operand 1 is an address pointing to a location within the memory space, which contains an address (the indirect address) for the source of the instruction. The indirect address is incremented as part of the instruction execution. This addressing mode is only valid on the MVI instruction. The instruction using this addressing mode is two bytes in length. Refer to the *PSoC Designer: Assembly Language User Guide* for further details on MVI instruction.

**Table 19. Source Indirect Post Increment**

| Opcode      | Operand 1              |
|-------------|------------------------|
| Instruction | Source Address Address |

**Example**

MVI A, [8] In this case, the value in the memory location at address 8 is an indirect address. The memory location pointed to by the indirect address is moved into the Accumulator. The indirect address is then incremented.

**Destination Indirect Post Increment**

The result of an instruction using this addressing mode is placed within the memory space. Operand 1 is an address pointing to a location within the memory space, which contains an address (the indirect address) for the destination of the instruction. The indirect address is incremented as part of the instruction execution. The source for the instruction is the Accumulator. This addressing mode is only valid on the MVI instruction. The instruction using this addressing mode is two bytes in length.

**Table 20. Destination Indirect Post Increment**

| Opcode      | Operand 1                   |
|-------------|-----------------------------|
| Instruction | Destination Address Address |

**Example**

MVI [8], A In this case, the value in the memory location at address 8 is an indirect address. The Accumulator is moved into the memory location pointed to by the indirect address. The indirect address is then incremented.

## Instruction Set Summary

The instruction set is summarized in Table 21 numerically and serves as a quick reference. If more information is needed, the Instruction Set Summary tables are described in detail in the *PSoC Designer Assembly Language User Guide* (available on [www.cypress.com](http://www.cypress.com)).

**Table 21. Instruction Set Summary Sorted Numerically by Opcode Order<sup>[1, 2]</sup>**

| Opcode Hex | Cycles | Bytes | Instruction Format | Flags | Opcode Hex | Cycles | Bytes | Instruction Format    | Flags                        | Opcode Hex | Cycles | Bytes | Instruction Format    | Flags |
|------------|--------|-------|--------------------|-------|------------|--------|-------|-----------------------|------------------------------|------------|--------|-------|-----------------------|-------|
| 00         | 15     | 1     | SSC                |       | 2D         | 8      | 2     | OR [X+expr], A        | Z                            | 5A         | 5      | 2     | MOV [expr], X         |       |
| 01         | 4      | 2     | ADD A, expr        | C, Z  | 2E         | 9      | 3     | OR [expr], expr       | Z                            | 5B         | 4      | 1     | MOV A, X              | Z     |
| 02         | 6      | 2     | ADD A, [expr]      | C, Z  | 2F         | 10     | 3     | OR [X+expr], expr     | Z                            | 5C         | 4      | 1     | MOV X, A              |       |
| 03         | 7      | 2     | ADD A, [X+expr]    | C, Z  | 30         | 9      | 1     | HALT                  |                              | 5D         | 6      | 2     | MOV A, reg[expr]      | Z     |
| 04         | 7      | 2     | ADD [expr], A      | C, Z  | 31         | 4      | 2     | XOR A, expr           | Z                            | 5E         | 7      | 2     | MOV A, reg[X+expr]    | Z     |
| 05         | 8      | 2     | ADD [X+expr], A    | C, Z  | 32         | 6      | 2     | XOR A, [expr]         | Z                            | 5F         | 10     | 3     | MOV [expr], [expr]    |       |
| 06         | 9      | 3     | ADD [expr], expr   | C, Z  | 33         | 7      | 2     | XOR A, [X+expr]       | Z                            | 60         | 5      | 2     | MOV reg[expr], A      |       |
| 07         | 10     | 3     | ADD [X+expr], expr | C, Z  | 34         | 7      | 2     | XOR [expr], A         | Z                            | 61         | 6      | 2     | MOV reg[X+expr], A    |       |
| 08         | 4      | 1     | PUSH A             |       | 35         | 8      | 2     | XOR [X+expr], A       | Z                            | 62         | 8      | 3     | MOV reg[expr], expr   |       |
| 09         | 4      | 2     | ADC A, expr        | C, Z  | 36         | 9      | 3     | XOR [expr], expr      | Z                            | 63         | 9      | 3     | MOV reg[X+expr], expr |       |
| 0A         | 6      | 2     | ADC A, [expr]      | C, Z  | 37         | 10     | 3     | XOR [X+expr], expr    | Z                            | 64         | 4      | 1     | ASL A                 | C, Z  |
| 0B         | 7      | 2     | ADC A, [X+expr]    | C, Z  | 38         | 5      | 2     | ADD SP, expr          |                              | 65         | 7      | 2     | ASL [expr]            | C, Z  |
| 0C         | 7      | 2     | ADC [expr], A      | C, Z  | 39         | 5      | 2     | CMP A, expr           | if (A=B) Z=1<br>if (A<B) C=1 | 66         | 8      | 2     | ASL [X+expr]          | C, Z  |
| 0D         | 8      | 2     | ADC [X+expr], A    | C, Z  | 3A         | 7      | 2     | CMP A, [expr]         |                              | 67         | 4      | 1     | ASR A                 | C, Z  |
| 0E         | 9      | 3     | ADC [expr], expr   | C, Z  | 3B         | 8      | 2     | CMP A, [X+expr]       |                              | 68         | 7      | 2     | ASR [expr]            | C, Z  |
| 0F         | 10     | 3     | ADC [X+expr], expr | C, Z  | 3C         | 8      | 3     | CMP [expr], expr      |                              | 69         | 8      | 2     | ASR [X+expr]          | C, Z  |
| 10         | 4      | 1     | PUSH X             |       | 3D         | 9      | 3     | CMP [X+expr], expr    |                              | 6A         | 4      | 1     | RLC A                 | C, Z  |
| 11         | 4      | 2     | SUB A, expr        | C, Z  | 3E         | 10     | 2     | MVI A, [ [expr]++ ]   | Z                            | 6B         | 7      | 2     | RLC [expr]            | C, Z  |
| 12         | 6      | 2     | SUB A, [expr]      | C, Z  | 3F         | 10     | 2     | MVI [ [expr]++ ], A   |                              | 6C         | 8      | 2     | RLC [X+expr]          | C, Z  |
| 13         | 7      | 2     | SUB A, [X+expr]    | C, Z  | 40         | 4      | 1     | NOP                   |                              | 6D         | 4      | 1     | RRC A                 | C, Z  |
| 14         | 7      | 2     | SUB [expr], A      | C, Z  | 41         | 9      | 3     | AND reg[expr], expr   | Z                            | 6E         | 7      | 2     | RRC [expr]            | C, Z  |
| 15         | 8      | 2     | SUB [X+expr], A    | C, Z  | 42         | 10     | 3     | AND reg[X+expr], expr | Z                            | 6F         | 8      | 2     | RRC [X+expr]          | C, Z  |
| 16         | 9      | 3     | SUB [expr], expr   | C, Z  | 43         | 9      | 3     | OR reg[expr], expr    | Z                            | 70         | 4      | 2     | AND F, expr           | C, Z  |
| 17         | 10     | 3     | SUB [X+expr], expr | C, Z  | 44         | 10     | 3     | OR reg[X+expr], expr  | Z                            | 71         | 4      | 2     | OR F, expr            | C, Z  |
| 18         | 5      | 1     | POP A              | Z     | 45         | 9      | 3     | XOR reg[expr], expr   | Z                            | 72         | 4      | 2     | XOR F, expr           | C, Z  |
| 19         | 4      | 2     | SBB A, expr        | C, Z  | 46         | 10     | 3     | XOR reg[X+expr], expr | Z                            | 73         | 4      | 1     | CPL A                 | Z     |
| 1A         | 6      | 2     | SBB A, [expr]      | C, Z  | 47         | 8      | 3     | TST [expr], expr      | Z                            | 74         | 4      | 1     | INC A                 | C, Z  |
| 1B         | 7      | 2     | SBB A, [X+expr]    | C, Z  | 48         | 9      | 3     | TST [X+expr], expr    | Z                            | 75         | 4      | 1     | INC X                 | C, Z  |
| 1C         | 7      | 2     | SBB [expr], A      | C, Z  | 49         | 9      | 3     | TST reg[expr], expr   | Z                            | 76         | 7      | 2     | INC [expr]            | C, Z  |
| 1D         | 8      | 2     | SBB [X+expr], A    | C, Z  | 4A         | 10     | 3     | TST reg[X+expr], expr | Z                            | 77         | 8      | 2     | INC [X+expr]          | C, Z  |
| 1E         | 9      | 3     | SBB [expr], expr   | C, Z  | 4B         | 5      | 1     | SWAP A, X             | Z                            | 78         | 4      | 1     | DEC A                 | C, Z  |
| 1F         | 10     | 3     | SBB [X+expr], expr | C, Z  | 4C         | 7      | 2     | SWAP A, [expr]        | Z                            | 79         | 4      | 1     | DEC X                 | C, Z  |
| 20         | 5      | 1     | POP X              |       | 4D         | 7      | 2     | SWAP X, [expr]        |                              | 7A         | 7      | 2     | DEC [expr]            | C, Z  |
| 21         | 4      | 2     | AND A, expr        | Z     | 4E         | 5      | 1     | SWAP A, SP            | Z                            | 7B         | 8      | 2     | DEC [X+expr]          | C, Z  |
| 22         | 6      | 2     | AND A, [expr]      | Z     | 4F         | 4      | 1     | MOV X, SP             |                              | 7C         | 13     | 3     | LCALL                 |       |
| 23         | 7      | 2     | AND A, [X+expr]    | Z     | 50         | 4      | 2     | MOV A, expr           | Z                            | 7D         | 7      | 3     | LJMP                  |       |
| 24         | 7      | 2     | AND [expr], A      | Z     | 51         | 5      | 2     | MOV A, [expr]         | Z                            | 7E         | 10     | 1     | RETI                  | C, Z  |
| 25         | 8      | 2     | AND [X+expr], A    | Z     | 52         | 6      | 2     | MOV A, [X+expr]       | Z                            | 7F         | 8      | 1     | RET                   |       |
| 26         | 9      | 3     | AND [expr], expr   | Z     | 53         | 5      | 2     | MOV [expr], A         |                              | 8x         | 5      | 2     | JMP                   |       |
| 27         | 10     | 3     | AND [X+expr], expr | Z     | 54         | 6      | 2     | MOV [X+expr], A       |                              | 9x         | 11     | 2     | CALL                  |       |
| 28         | 11     | 1     | ROMX               | Z     | 55         | 8      | 3     | MOV [expr], expr      |                              | Ax         | 5      | 2     | JZ                    |       |
| 29         | 4      | 2     | OR A, expr         | Z     | 56         | 9      | 3     | MOV [X+expr], expr    |                              | Bx         | 5      | 2     | JNZ                   |       |
| 2A         | 6      | 2     | OR A, [expr]       | Z     | 57         | 4      | 2     | MOV X, expr           |                              | Cx         | 5      | 2     | JC                    |       |
| 2B         | 7      | 2     | OR A, [X+expr]     | Z     | 58         | 6      | 2     | MOV X, [expr]         |                              | Dx         | 5      | 2     | JNC                   |       |
| 2C         | 7      | 2     | OR [expr], A       | Z     | 59         | 7      | 2     | MOV X, [X+expr]       |                              | Ex         | 7      | 2     | JACC                  |       |
|            |        |       |                    |       |            |        |       |                       |                              | Fx         | 13     | 2     | INDEX                 | Z     |

**Notes**

1. Interrupt routines take 13 cycles before execution resumes at interrupt vector table.
2. The number of cycles required by an instruction is increased by one for instructions that span 256-byte boundaries in the Flash memory space.

## Memory Organization

### Flash Program Memory Organization

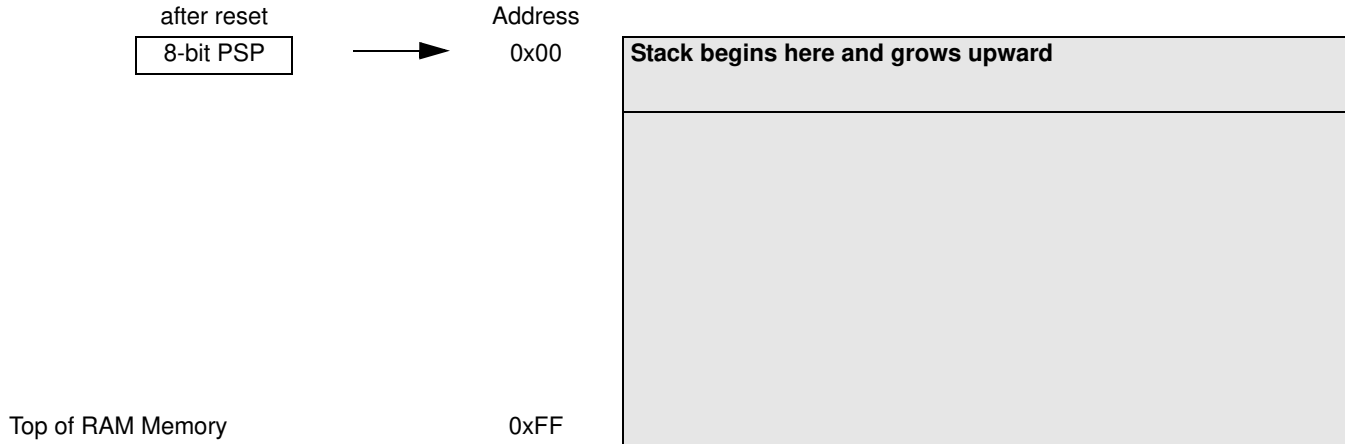
Table 22. Program Memory Space with Interrupt Vector Table

|                          |   |         |  |
|--------------------------|---|---------|--|
| after reset<br>16-bit PC | → | Address |  |
|                          |   | 0x0000  | Program execution begins here after a reset  |
|                          |   | 0x0004  | POR  |
|                          |   | 0x0008  | Reserved   |
|                          |   | 0x000C  | SPI Transmitter Empty  |
|                          |   | 0x0010  | SPI Receiver Full  |
|                          |   | 0x0014  | GPIO Port 0  |
|                          |   | 0x0018  | GPIO Port 1  |
|                          |   | 0x001C  | INT1   |
|                          |   | 0x0020  | Reserved   |
|                          |   | 0x0024  | Reserved   |
|                          |   | 0x0028  | Reserved   |
|                          |   | 0x002C  | Reserved   |
|                          |   | 0x0030  | Reserved   |
|                          |   | 0x0034  | 1 ms Interval Timer  |
|                          |   | 0x0038  | Programmable Interval Timer  |
|                          |   | 0x003C  | Reserved   |
|                          |   | 0x0040  | Reserved   |
|                          |   | 0x0044  | 16-bit Free Running Timer Wrap   |
|                          |   | 0x0048  | INT2   |
|                          |   | 0x004C  | Reserved   |
|                          |   | 0x0050  | GPIO Port 2  |
|                          |   | 0x0054  | Reserved   |
|                          |   | 0x0058  | Reserved   |
|                          |   | 0x005C  | Reserved   |
|                          |   | 0x0060  | Reserved   |
|                          |   | 0x0064  | Sleep Timer  |
|                          |   | 0x0068  | <b>Program Memory begins here (if below interrupts not used, program memory can start lower)</b> |
|                          |   |         |  |
|                          |   | 0x1FFF  |  |

### Data Memory Organization

The MCU function provides up to 256 bytes of data RAM.

**Table 23. Data Memory Organization**



### Flash

This section describes the Flash block of the CYRF69303. Much of the user visible Flash functionality, including programming and security, are implemented in the M8C Supervisory Read Only Memory (SROM). CYRF69303 Flash has an endurance of 1000 cycles and 10-year data retention.

#### Flash Programming and Security

All Flash programming is performed by code in the SROM. The registers that control the Flash programming are only visible to the M8C CPU when it is executing out of SROM. This makes it impossible to read, write, or erase the Flash by bypassing the security mechanisms implemented in the SROM.

Customer firmware can only program the Flash through SROM calls. The data or code images can be sourced by way of any interface with the appropriate support firmware. This type of programming requires a 'bootloader'—a piece of firmware resident on the Flash. For safety reasons, this bootloader must not be over written during firmware rewrites.

The Flash provides four auxiliary rows that are used to hold Flash block protection flags, boot time calibration values, configuration tables, and any device values. The routines for accessing these auxiliary rows are documented in the SROM section. The auxiliary rows are not affected by the device erase function.

#### In-System Programming

CYRF69303 enables this type of in-system programming by using the P1.0 and P1.1 pins as the serial programming mode interface. This allows an external controller to cause the CYRF69303 to enter serial programming mode and then to use the test queue to issue Flash access functions in the SROM.

### SROM

The SROM holds code that is used to boot the part, calibrate circuitry, and perform Flash operations (Table 24 lists the SROM functions). The functions of the SROM may be accessed in normal user code or operating from Flash. The SROM exists in a separate memory space from user code. The SROM functions are accessed by executing the Supervisory System Call instruction (SSC), which has an opcode of 00h. Before executing the SSC, the M8C's accumulator needs to be loaded with the desired SROM function code from Table 24. Undefined functions causes a HALT if called from user code. The SROM functions are executing code with calls; therefore, the functions require stack space. With the exception of Reset, all of the SROM functions have a *parameter block* in SRAM that must be configured before executing the SSC. Table 25 lists all possible parameter block variables. The meaning of each parameter, with regards to a specific SROM function, is described later in this section.

**Table 24. SROM Function Codes**

| Function Code | Function Name | Stack Space |
|---------------|---------------|-------------|
| 00h           | SWBootReset   | 0           |
| 01h           | ReadBlock     | 7           |
| 02h           | WriteBlock    | 10          |
| 03h           | EraseBlock    | 9           |
| 05h           | EraseAll      | 11          |
| 06h           | TableRead     | 3           |
| 07h           | Checksum      | 3           |

Two important variables that are used for all functions are KEY1 and KEY2. These variables are used to help discriminate between valid SSCs and inadvertent SSCs. KEY1 must always have a value of 3Ah, while KEY2 must have the same value as the stack pointer when the SROM function begins execution. This is the Stack Pointer value when the SSC opcode is executed, plus three. If either of the keys do not match the expected values, the M8C halts (with the exception of the SWBootReset function). The following code puts the correct value in KEY1 and KEY2. The code starts with a halt, to force the program to jump directly into the setup code and not run into it.

```
halt
SSCOP: mov [KEY1], 3ah
mov X, SP
mov A, X
add A, 3
mov [KEY2], A
```

**Table 25. SROM Function Parameters**

| Variable Name            | SRAM Address |
|--------------------------|--------------|
| Key1/Counter/Return Code | 0,F8h        |
| Key2/TMP                 | 0,F9h        |
| BlockID                  | 0,FAh        |
| Pointer                  | 0,FBh        |
| Clock                    | 0,FC h       |
| Mode                     | 0,FDh        |
| Delay                    | 0,FEh        |
| PCL                      | 0,FFh        |

The SROM also features Return Codes and Lockouts.

#### Return Codes

Return codes aid in the determination of success or failure of a particular function. The return code is stored in KEY1's position in the parameter block. The CheckSum and TableRead functions do not have return codes because KEY1's position in the parameter block is used to return other data.

**Table 26. SROM Return Codes**

| Return Code | Description  |
|-------------|--|
| 00h         | Success  |
| 01h         | Function not allowed due to level of protection on block |
| 02h         | Software reset without hardware reset                    |
| 03h         | Fatal error, SROM halted                                 |

Read, write, and erase operations may fail if the target block is read or write protected. Block protection levels are set during device programming.

The EraseAll function overwrites data in addition to leaving the entire user Flash in the erase state. The EraseAll function loops through the number of Flash macros in the product, executing the following sequence: erase, bulk program all zeros, erase. After all the user space in all the Flash macros are erased, a second loop erases and then programs each protection block with zeros.

## SROM Function Descriptions

All SROM functions are described in the following sections.

### SWBootReset Function

The SROM function, SWBootReset, is the function that is responsible for transitioning the device from a reset state to running user code. The SWBootReset function is executed whenever the SROM is entered with an M8C accumulator value of 00h; the SRAM parameter block is not used as an input to the function. This happens, by design, after a hardware reset, because the M8C's accumulator is reset to 00h or when user code executes the SSC instruction with an accumulator value of 00h. The SWBootReset function does not execute when the SSC instruction is executed with a bad key value and a nonzero function code. A CYRF69303 device executes the HALT instruction if a bad value is given for either KEY1 or KEY2.

The SWBootReset function verifies the integrity of the calibration data by way of a 16-bit checksum, before releasing the M8C to run user code.

### ReadBlock Function

The ReadBlock function is used to read 64 contiguous bytes from Flash—a block.

The first thing this function does is to check the protection bits and determine if the desired BLOCKID is readable. If read protection is turned on, the ReadBlock function exits setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h, indicating a read failure. If read protection is not enabled, the function reads 64 bytes from the Flash using a ROMX instruction and store the results in SRAM using an MVI instruction. The first of the 64 bytes are stored in SRAM at the address indicated by the value of the POINTER parameter. When the ReadBlock completes successfully, the accumulator, KEY1 and KEY2, all have a value of 00h.

**Table 27. ReadBlock Parameters**

| Name    | Address | Description  |
|---------|---------|--|
| KEY1    | 0,F8h   | 3Ah  |
| KEY2    | 0,F9h   | Stack Pointer value, when SSC is executed                        |
| BLOCKID | 0,FAh   | Flash block number   |
| POINTER | 0,FBh   | First of 64 addresses in SRAM where returned data must be stored |

### WriteBlock Function

The WriteBlock function is used to store data in the Flash. Data is moved 64 bytes at a time from SRAM to Flash using this function. The first thing the WriteBlock function does is to check the protection bits and determine if the desired BLOCKID is writable. If write protection is turned on, the WriteBlock function exits, setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h, indicating a write failure. The configuration of the WriteBlock function is straightforward. The BLOCKID of the Flash block, where the data is stored, must be determined and stored at SRAM address FAh.

The SRAM address of the first of the 64 bytes to be stored in Flash must be indicated using the POINTER variable in the parameter block (SRAM address FBh). Finally, the CLOCK and DELAY values must be set correctly. The CLOCK value determines the length of the write pulse that is used to store the data in the Flash. The CLOCK and DELAY values are dependent on the CPU. Refer to 'Clocking' Section for additional information.

**Table 28. WriteBlock Parameters**

| Name     | Address | Description   |
|----------|---------|---|
| KEY1     | 0,F8h   | 3Ah   |
| KEY2     | 0,F9h   | Stack Pointer value, when SSC is executing  |
| BLOCK ID | 0,FAh   | 8 KB Flash block number (00h–7Fh)<br>4 KB Flash block number (00h–3Fh)<br>3 KB Flash block number (00h–2Fh) |
| POINTER  | 0,FBh   | First 64 addresses in SRAM where the data to be stored in Flash is located before calling WriteBlock        |
| CLOCK    | 0,FCh   | Clock Divider used to set the write Pulse width   |
| DELAY    | 0,FEh   | For a CPU speed of 12 MHz set to 56h  |

*EraseBlock Function*

The EraseBlock function is used to erase a block of 64 contiguous bytes in Flash. The first thing the EraseBlock function does is to check the protection bits and determine if the desired BLOCKID is writable. If write protection is turned on, the EraseBlock function exits, setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h, indicating a write failure. The EraseBlock function is only useful as the first step in programming. Erasing a block does not cause data in a block to be one hundred percent unreadable. If the objective is to obliterate data in a block, the best method is to perform an EraseBlock followed by a WriteBlock of all zeros.

To setup the parameter block for the EraseBlock function, correct key values must be stored in KEY1 and KEY2. The block number to be erased must be stored in the BLOCKID variable and the CLOCK and DELAY values must be set based on the current CPU speed.

**Table 29. EraseBlock Parameters**

| Name    | Address | Description                                     |
|---------|---------|---|
| KEY1    | 0,F8h   | 3Ah   |
| KEY2    | 0,F9h   | Stack Pointer value when SSC is executed        |
| BLOCKID | 0,FAh   | Flash block number (00h–7Fh)                    |
| CLOCK   | 0,FCh   | Clock Divider used to set the erase pulse width |
| DELAY   | 0,FEh   | For a CPU speed of 12 MHz set to 56h            |

*ProtectBlock Function*

The CYRF69303 device offers Flash protection on a block-by-block basis. Table 30 lists the protection modes available. In the table, ER and EW are used to indicate the ability to perform external reads and writes. For internal writes, IW is used. Internal reading is always permitted by way of the ROMX instruction. The ability to read by way of the SROM ReadBlock function is indicated by SR. The protection level is stored in two bits, according to Table 30. These bits are bit packed into the 64 bytes of the protection block. Therefore, each protection block byte stores the protection level for four Flash blocks. The bits are packed into a byte, with the lowest numbered block's protection level stored in the lowest numbered bits.

The first address of the protection block contains the protection level for blocks 0 through 3; the second address is for blocks 4 through 7. The 64th byte stores the protection level for blocks 252 through 255.

**Table 30. Protection Modes**

| Mode | Settings    | Description            | Marketing       |
|------|-------------|------------------------|-----------------|
| 00b  | SR ER EW IW | Unprotected            | Unprotected     |
| 01b  | SR ER EW IW | Read protect           | Factory upgrade |
| 10b  | SR ER EW IW | Disable external write | Field upgrade   |
| 11b  | SR ER EW IW | Disable internal write | Full protection |

| 7         | 6 | 5         | 4 | 3         | 2 | 1       | 0 |
|-----------|---|-----------|---|-----------|---|---------|---|
| Block n+3 |   | Block n+2 |   | Block n+1 |   | Block n |   |

The level of protection is only decreased by an EraseAll, which places zeros in all locations of the protection block. To set the level of protection, the ProtectBlock function is used. This function takes data from SRAM, starting at address 80h, and ORs it with the current values in the protection block. The result of the OR operation is then stored in the protection block. The EraseBlock function does not change the protection level for a block. Because the SRAM location for the protection data is fixed and there is only one protection block per Flash macro, the ProtectBlock function expects very few variables in the parameter block to be set before calling the function. The parameter block values that must be set, besides the keys, are the CLOCK and DELAY values.

**Table 31. ProtectBlock Parameters**

| Name  | Address | Description                                     |
|-------|---------|---|
| KEY1  | 0,F8h   | 3Ah   |
| KEY2  | 0,F9h   | Stack Pointer value when SSC is executed        |
| CLOCK | 0,FCh   | Clock Divider used to set the write pulse width |
| DELAY | 0,FEh   | For a CPU speed of 12 MHz set to 56h            |



### EraseAll Function

The EraseAll function performs a series of steps that destroy the user data in the Flash macros and resets the protection block in each Flash macro to all zeros (the unprotected state). The EraseAll function does not affect the three hidden blocks above the protection block in each Flash macro. The first of these four hidden blocks is used to store the protection table for its eight Kbytes of user data.

The EraseAll function begins by erasing the user space of the Flash macro with the highest address range. A bulk program of all zeros is then performed on the same Flash macro, to destroy all traces of the previous contents. The bulk program is followed by a second erase that leaves the Flash macro in a state ready for writing. The erase, program, erase sequence is then performed on the next lowest Flash macro in the address space if it exists. Following the erase of the user space, the protection block for the Flash macro with the highest address range is erased. Following the erase of the protection block, zeros are written into every bit of the protection table. The next lowest Flash macro in the address space then has its protection block erased and filled with zeros.

The end result of the EraseAll function is that all user data in the Flash is destroyed and the Flash is left in an unprogrammed state, ready to accept one of the various write commands. The protection bits for all user data are also reset to the zero state.

The parameter block values that must be set, besides the keys, are the CLOCK and DELAY values.

**Table 32. EraseAll Parameters**

| Name  | Address | Description                                     |
|-------|---------|---|
| KEY1  | 0,F8h   | 3Ah   |
| KEY2  | 0,F9h   | Stack Pointer value when SSC is executed        |
| CLOCK | 0,FCh   | Clock Divider used to set the write pulse width |
| DELAY | 0,FEh   | For a CPU speed of 12 MHz set to 56h            |

### TableRead Function

The TableRead function gives the user access to part specific data stored in the Flash during manufacturing. It also returns a Revision ID for the die (not to be confused with the Silicon ID).

**Table 33. Table Read Parameters**

| Name    | Address | Description                              |
|---------|---------|--|
| KEY1    | 0,F8h   | 3Ah                                      |
| KEY2    | 0,F9h   | Stack Pointer value when SSC is executed |
| BLOCKID | 0,FAh   | Table number to read                     |

The table space for the CYRF69303 is simply a 64 byte row broken up into eight tables of eight bytes. The tables are numbered zero through seven. All user and hidden blocks in the CYRF69303 consist of 64 bytes.

An internal table holds the Silicon ID and returns the Revision ID. The Silicon ID is returned in SRAM, while the Revision ID is returned in the CPU\_A and CPU\_X registers. The Silicon ID is a value placed in the table by programming the Flash and is controlled by Cypress Semiconductor Product Engineering. The Revision ID is hard coded into the SROM. The Revision ID is discussed in more detail later in this section.

An internal table holds alternate trim values for the device and returns a one-byte internal revision counter. The internal revision counter starts out with a value of zero and is incremented each time one of the other revision numbers is not incremented. It is reset to zero each time one of the other revision numbers is incremented. The internal revision count is returned in the CPU\_A register. The CPU\_X register is always set to FFh when trim values are read. The BLOCKID value, in the parameter block, is used to indicate which table must be returned to the user. Only the three least significant bits of the BLOCKID parameter are used by the TableRead function for the CYRF69303. The upper five bits are ignored. When the function is called, it transfers bytes from the table to SRAM addresses F8h–FFh.

The M8C's A and X registers are used by the TableRead function to return the die's Revision ID. The Revision ID is a 16-bit value hard coded into the SROM that uniquely identifies the die's design.

### Checksum Function

The Checksum function calculates a 16-bit checksum over a user specifiable number of blocks, within a single Flash macro (Bank) starting from block zero. The BLOCKID parameter is used to pass in the number of blocks to calculate the checksum over. A BLOCKID value of 1 calculates the checksum of only block 0, while a BLOCKID value of 0 calculates the checksum of all 256 user blocks. The 16-bit checksum is returned in KEY1 and KEY2. The parameter KEY1 holds the lower eight bits of the checksum and the parameter KEY2 holds the upper eight bits of the checksum.

The checksum algorithm executes the following sequence of three instructions over the number of blocks times 64 to be checksummed.

```
romx
add [KEY1], A
adc [KEY2], 0
```

**Table 34. Checksum Parameters**

| Name    | Address | Description                                     |
|---------|---------|---|
| KEY1    | 0,F8h   | 3Ah   |
| KEY2    | 0,F9h   | Stack Pointer value when SSC is executed        |
| BLOCKID | 0,FAh   | Number of Flash blocks to calculate checksum on |

### Clocking

The CYRF69303 internal oscillator outputs two frequencies, the Internal 24 MHz Oscillator and the 32 kHz Low power Oscillator.

The Internal 24 MHz Oscillator is designed such that it may be trimmed to an output frequency of 24 MHz over temperature and voltage variation. The Internal 24 MHz Oscillator accuracy is 24 MHz -22% to +10% (between 0 °C -70 °C). No external components are required to achieve this level of accuracy.

Firmware is responsible for selecting the correct trim values from the User row to match the power supply voltage in the end application and writing the values to the trim registers IOSCTR and LPOSCTR.

The internal low speed oscillator of nominally 32 kHz provides a slow clock source for the CYRF69303 in suspend mode. This is used to generate a periodic wakeup interrupt and provide a clock to sequential logic during power-up and power-down events when the main clock is stopped. In addition, this oscillator can

also be used as a clocking source for the Interval Timer clock (ITMRCLK) and Capture Timer clock (TCAPCLK). The 32 kHz Low power Oscillator can operate in low power mode or can provide a more accurate clock in normal mode. The Internal 32 kHz Low power Oscillator accuracy ranges from -53.12% to +56.25%. The 32 kHz low power oscillator can be calibrated against the internal 24 MHz oscillator or another timing source if desired.

CYRF69303 provides the ability to load new trim values for the 24 MHz oscillator based on voltage. This allows  $V_{DD}$  to be monitored and have firmware trim the oscillator based on voltage present. The IOSCTR register is used to set trim values for the 24 MHz oscillator. CYRF69303 is initialized with 3.30 V trim values at power on, then firmware is responsible for transferring the correct set of trim values to the trim registers to match the application's actual  $V_{DD}$ . The 32 kHz oscillator generally does not require trim adjustments versus voltage but trim values for the 32 kHz are also stored in Supervisory ROM.

Figure 7. SROM Table

|         | F8h                     | F9h                     | FAh                     | FBh                     | FCh                    | FDh                    | FEh                    | FFh                    |
|---------|-------------------------|-------------------------|-------------------------|-------------------------|------------------------|------------------------|------------------------|------------------------|
| Table 0 | Silicon ID [15-8]       | Silicon ID [7-0]        |                         |                         |                        |                        |                        |                        |
| Table 1 |                         |                         |                         |                         |                        |                        |                        |                        |
| Table 2 |                         |                         |                         |                         | 24 MHz IOSCTR at 3.30V | 24 MHz IOSCTR at 3.00V | 24 MHz IOSCTR at 2.85V | 24 MHz IOSCTR at 2.70V |
| Table 3 | 32 kHz LPOSCTR at 3.30V | 32 kHz LPOSCTR at 3.00V | 32 kHz LPOSCTR at 2.85V | 32 kHz LPOSCTR at 2.70V |                        |                        |                        |                        |
| Table 4 |                         |                         |                         |                         |                        |                        |                        |                        |
| Table 5 |                         |                         |                         |                         |                        |                        |                        |                        |
| Table 6 |                         |                         |                         |                         |                        |                        |                        |                        |
| Table 7 |                         |                         |                         |                         |                        |                        |                        |                        |

To improve the accuracy of the IMO, new trim values are loaded based on supply voltage to the part. For this, firmware needs to make modifications to two registers:

1. The internal oscillator trim register at location 0x34.
2. The gain register at location 0x38.

#### Trim values for the IOSCTR register:

The trim values are stored in SROM tables in the part as shown in [Figure 7 on page 22](#). The trim values are read out from the part based on voltage settings and written to the IOSCTR register at location 0x34. The following pseudo code shows how this is done.

```
_main:
    mov     A, 2
    mov     [SSC_BLOCKID], A
Call SROM operation to read the SROM table (Refer to Table 33 on page 21 in the section SROM on page 18)
//After this command is executed, the trim values for 3.3, 3.0, 2.85 and 2.7 are stored at locations
FC through FF in the RAM. SROM calls are explained in the previous section of this datasheet
;         mov     A, [FCh]                // trim values for 3.3 V
         mov     A, [FDh]                // trim values for 3.0 V
;         mov     A, [FEh]                // trim values for 2.85 V
;         mov     A, [FFh]                // trim values for 2.70 V
         mov     reg[IOSCTR],A // Loading IOSCTR with trim values for 3.0 V
.terminate:
    jmp .terminate
```

#### SROM Table Read Description

The Silicon IDs for CYRF69303 devices are stored in SROM tables in the part, as shown in [Figure 7](#).

The Silicon ID can be read out from the part using SROM Table reads. This is demonstrated in the following pseudo code. As mentioned in the section [SROM on page 18](#), the SROM variables occupy address F8h through FFh in the SRAM. Each of the variables and their definition is given in section [SROM on page 18](#).

```
AREA SSCParmBlkA(RAM,ABS)

    org F8h // Variables are defined starting at address F8h

SSC_KEY1:        ; F8h supervisory key
SSC_RETURNCODE: blk 1 ; F8h result code
SSC_KEY2 :       blk 1 ; F9h supervisory stack ptr key
SSC_BLOCKID:    blk 1 ; FAh block ID
SSC_POINTER:    blk 1 ; FBh pointer to data buffer
SSC_CLOCK:      blk 1 ; FCh Clock
SSC_MODE:       blk 1 ; FDh ClockW ClockE multiplier
SSC_DELAY:      blk 1 ; FEh flash macro sequence delay count
SSC_WRITE_ResultCode: blk 1 ; FFh temporary result code

_main:
    mov     A, 0
    mov     [SSC_BLOCKID], A // To read from Table 0 - Silicon ID is stored in Table 0
//Call SROM operation to read the SROM table
    mov     X, SP           ; copy SP into X
    mov     A, X           ; A temp stored in X
    add     A, 3           ; create 3 byte stack frame (2 + pushed A)
    mov     [SSC_KEY2], A   ; save stack frame for supervisory code

    ; load the supervisory code for flash operations
    mov     [SSC_KEY1], 3Ah ;FLASH_OPER_KEY - 3Ah

    mov     A,6           ; load A with specific operation. 06h is the code for Table read Table 24 on page 19
    SSC                    ; SSC call the supervisory ROM

// At the end of the SSC command the silicon ID is stored in F8 (MSB) and F9(LSB) of the SRAM

.terminate:
```

```
jmp .terminate
```

**Gain value for the register at location [0x38]:**

3.3 V = 0x40

3.0 V = 0x40

2.85 V = 0xFF

2.70 V = 0xFF

Load register [0x38] with the gain values corresponding to the appropriate voltage.

**Table 35. Oscillator Trim Values vs. Voltage Settings**

| Supervisory ROM Table | Function                 |
|-----------------------|--------------------------|
| Table2 FCh            | 24 MHz IOSCTR at 3.30 V  |
| Table2 FDh            | 24 MHz IOSCTR at 3.00 V  |
| Table2 FEh            | 24 MHz IOSCTR at 2.85 V  |
| Table2 FFh            | 24 MHz IOSCTR at 2.70 V  |
| Table3 F8h            | 32 kHz LPOSCTR at 3.30 V |
| Table3 F9h            | 32 kHz LPOSCTR at 3.00 V |
| Table3 FAh            | 32 kHz LPOSCTR at 2.85 V |
| Table3 FBh            | 32 kHz LPOSCTR at 2.70 V |

When using the 32 kHz oscillator the PITMRL/H must be read until two consecutive readings match before sending/receiving data. The following firmware example assumes the developer is interested in the lower byte of the PIT.

```
Read_PIT_counter:
mov A, reg[PITMRL]
mov [57h], A
```

**Table 36. CPU Clock Config (CPUCLKCR) [0x30] [R/W]**

| Bit #      | 7        | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----------|---|---|---|---|---|---|---|
| Field      | Reserved |   |   |   |   |   |   |   |
| Read/Write | -        | - | - | - | - | - | - | - |
| Default    | 0        | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0** Reserved

**Note** The CPU speed selection is configured using the OSC\_CR0 Register (Figure 8 on page 27).

```
mov A, reg[PITMRL]
mov [58h], A
mov [59h], A
mov A, reg[PITMRL]
mov [60h], A
;;;Start comparison
mov A, [60h]
mov X, [59h]
sub A, [59h]
jz done
mov A, [59h]
mov X, [58h]
sub A, [58h]
jz done
mov X, [57h]
;;;correct data is in memory location 57h
done:
mov [57h], X
ret
```

**Clock Architecture Description**

The CYRF69303 clock selection circuitry allows the selection of independent clocks for the CPU, Interval Timers, and Capture Timers.

*CPU Clock*

The CPU clock, CPUCLK, can be sourced from the Internal 24 MHz oscillator. The selected clock source can optionally be divided by  $2^{n-1}$  where  $n$  is 0–7 (see Table 37 on page 25).

**Table 37. OSC Control 0 (OSC\_CR0) [0x1E0] [R/W]**

| Bit #      | 7        | 6 | 5       | 4                 | 3   | 2               | 1   | 0   |
|------------|----------|---|---------|-------------------|-----|-----------------|-----|-----|
| Field      | Reserved |   | No Buzz | Sleep Timer [1:0] |     | CPU Speed [2:0] |     |     |
| Read/Write | –        | – | R/W     | R/W               | R/W | R/W             | R/W | R/W |
| Default    | 0        | 0 | 0       | 0                 | 1   | 0               | 0   | 0   |

**Bits 7:6** Reserved

**Bit 5** No Buzz

During sleep (the Sleep bit is set in the CPU\_SCR Register—[Table 41 on page 29](#)), the POR detection circuit is turned on periodically to detect any POR events on the V<sub>CC</sub> pin (the Sleep Duty Cycle bits in the ECO\_TR are used to control the duty cycle—[Table 45 on page 33](#)). To facilitate the detection of POR events, the No Buzz bit is used to force the POR detection circuit to be continuously enabled during sleep. This results in a faster response to a POR event during sleep at the expense of a slightly higher than average sleep current. Obtaining the absolute lowest power usage in sleep mode requires the No Buzz bit be clear

0 = The POR detection circuit is turned on periodically as configured in the Sleep Duty Cycle.

1 = The Sleep Duty Cycle value is overridden. The POR detection circuit is always enabled.

**Note** The periodic Sleep Duty Cycle enabling is independent with the sleep interval shown in the following Sleep [1:0] bits.

**Bits 4:3** Sleep Timer [1:0]

| Sleep Timer [1:0] | Sleep Timer Clock Frequency (Nominal) | Sleep Period (Nominal) | Watchdog Period (Nominal) |
|-------------------|---------------------------------------|------------------------|---------------------------|
| 00                | 512 Hz                                | 1.95 ms                | 6 ms                      |
| 01                | 64 Hz                                 | 15.6 ms                | 47 ms                     |
| 10                | 8 Hz                                  | 125 ms                 | 375 ms                    |
| 11                | 1 Hz                                  | 1 sec                  | 3 sec                     |

**Note** Sleep intervals are approximate

**Bits 2:0** CPU Speed [2:0]

The CYRF69303 may operate over a range of CPU clock speeds. The reset value for the CPU Speed bits is zero. Therefore, the default CPU speed is 3 MHz.

| CPU Speed [2:0] | CPU when Internal Oscillator is selected |
|-----------------|--|
| 000             | 3 MHz (Default)                          |
| 001             | 6 MHz                                    |
| 010             | 12 MHz                                   |
| 011             | Reserved                                 |
| 100             | 1.5 MHz                                  |
| 101             | 750 kHz                                  |
| 110             | 187 kHz                                  |
| 111             | Reserved                                 |

**Table 38. Timer Clock Config (TMRCLKCR) [0x31] [R/W]**

| Bit #      | 7               | 6   | 5              | 4   | 3               | 2   | 1              | 0   |
|------------|-----------------|-----|----------------|-----|-----------------|-----|----------------|-----|
| Field      | TCAPCLK Divider |     | TCAPCLK Select |     | ITMRCLK Divider |     | ITMRCLK Select |     |
| Read/Write | R/W             | R/W | R/W            | R/W | R/W             | R/W | R/W            | R/W |
| Default    | 1               | 0   | 0              | 0   | 1               | 1   | 1              | 1   |

**Bits 7:6**TCAPCLK Divider [1:0]

TCAPCLK Divider controls the TCAPCLK divisor.

- 0 0 = Divider Value 2
- 0 1 = Divider Value 4
- 1 0 = Divider Value 6
- 1 1 = Divider Value 8

**Bits 5:4** TCAPCLK Select

The TCAPCLK Select field controls the source of the TCAPCLK.

- 0 0 = Internal 24 MHz Oscillator
- 0 1 =Reserved
- 1 0 = Internal 32 kHz Low power Oscillator
- 1 1 = TCAPCLK Disabled

**Note** The 1024  $\mu$ s interval timer is based on the assumption that TCAPCLK is running at 4 MHz. Changes in TCAPCLK frequency cause a corresponding change in the 1024  $\mu$ s interval timer frequency.

**Bits 3:2** ITMRCLK Divider

ITMRCLK Divider controls the ITMRCLK divisor

- 0 0 = Divider value of 1
- 0 1 = Divider value of 2
- 1 0 = Divider value of 3
- 1 1 = Divider value of 4

**Bits 1:0** ITMRCLK Select

- 0 0 = Internal 24 MHz Oscillator
- 0 1 = Reserved
- 1 0 = Internal 32 kHz Low power Oscillator
- 1 1 = TCAPCLK

**Note** Changing the source of TMRCLK requires that both the source and destination clocks be running. Attempting to change the clock source away from TCAPCLK after that clock has been stopped is not successful.

*Interval Timer Clock (ITMRCLK)*

The Interval Timer clock (ITMRCLK) can be sourced from the internal 24 MHz oscillator, internal 32 kHz low power oscillator, or timer capture clock. A programmable prescaler of 1, 2, 3, or 4 then divides the selected source. The 12-bit Programmable Interval Timer is a simple down counter with a programmable reload value. It provides a 1  $\mu$ s resolution by default. When the down counter reaches zero, the next clock is spent reloading. The reload value can be read and written while the counter is running, but care must be taken to ensure that the counter does not unintentionally reload while the 12-bit reload value is only partially stored—for example, between the two writes of the 12-bit value. The programmable interval timer generates interrupt to the CPU on each reload.

The parameters to be set appears on the device editor view of PSoC Designer after you place the CYRF69303 timer user module. The parameters are PITIMER\_Source and PITIMER\_Divider. The PITIMER\_Source is the clock to the timer and the PITIMER\_Divider is the value the clock is divided by.

The interval register (PITMR) holds the value that is loaded into the PIT counter on terminal count. The PIT counter is a down counter.

The Programmable Interval Timer resolution is configurable. For example:

TCAPCLK divide by x of CPU clock (for example TCAPCLK divide by 2 of a 24 MHz CPU clock gives a frequency of 12 MHz)

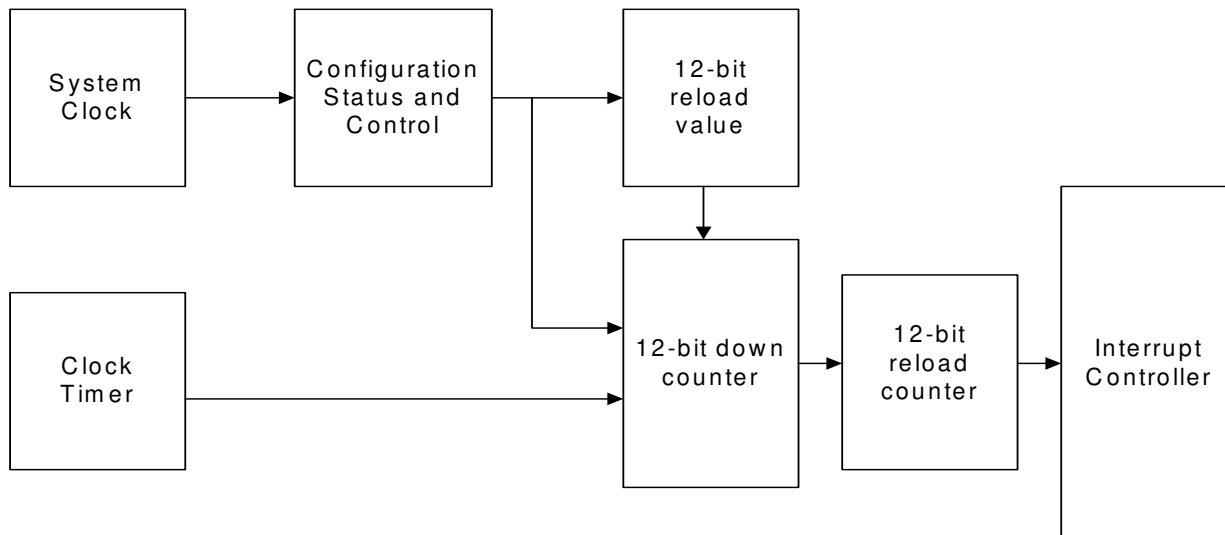
ITMRCLK divide by x of TCAPCLK (for example, ITMRCLK divide by 3 of TCAPCLK is 4 MHz so resolution is 0.25  $\mu$ s).

*Timer Capture Clock (TCAPCLK)*

The Timer Capture clock (TCAPCLK) can be sourced from the internal 24 MHz oscillator or the internal 32 kHz low power oscillator. A programmable prescaler of 2, 4, 6, or 8 then divides the selected source.



Figure 8. Programmable Interval Timer Block Diagram



Internal Clock Trim

Table 39. IOSC Trim (IOSCTR) [0x34] [R/W]

| Bit #      | 7          | 6   | 5   | 4         | 3   | 2   | 1   | 0   |
|------------|------------|-----|-----|-----------|-----|-----|-----|-----|
| Field      | ffset[2:0] |     |     | Gain[4:0] |     |     |     |     |
| Read/Write | R/W        | R/W | R/W | R/W       | R/W | R/W | R/W | R/W |
| Default    | 0          | 0   | 0   | D         | D   | D   | D   | D   |

The IOSC Calibrate register is used to calibrate the internal oscillator. The reset value is undefined but during boot the SROM writes a calibration value that is determined during manufacturing test. The 'D' indicates that the default value is trimmed to 24 MHz at 3.30 V at power on.

**Bits 7:5** fffset [2:0]

This value is used to trim the frequency of the internal oscillator. These bits are not used in factory calibration and are zero. Setting each of these bits causes the appropriate fine offset in oscillator frequency:  
 fffset bit 0 = 7.5 kHz  
 fffset bit 1 = 15 kHz  
 fffset bit 2 = 30 kHz

**Bits 4:0** Gain [4:0]

The effective frequency change of the offset input is controlled through the gain input. A lower value of the gain setting increases the gain of the offset input. This value sets the size of each offset step for the internal oscillator. Nominal gain change (kHz/offsetStep) at each bit, typical conditions (24 MHz operation):  
 Gain bit 0 = -1.5 kHz  
 Gain bit 1 = -3.0 kHz  
 Gain bit 2 = -6 kHz  
 Gain bit 4 = -24 kHz

*LPOSC Trim*
**Table 40. LPOSC Trim (LPOSCTR) [0x36] [R/W]**

| Bit #      | 7                | 6        | 5                      | 4   | 3                      | 2   | 1   | 0   |
|------------|------------------|----------|------------------------|-----|------------------------|-----|-----|-----|
| Field      | 32 kHz Low Power | Reserved | 32 kHz Bias Trim [1:0] |     | 32 kHz Freq Trim [3:0] |     |     |     |
| Read/Write | R/W              | –        | R/W                    | R/W | R/W                    | R/W | R/W | R/W |
| Default    | 0                | –        | D                      | D   | D                      | D   | D   | D   |

This register is used to calibrate the 32 kHz Low speed Oscillator. The reset value is undefined but during boot the SROM writes a calibration value that is determined during manufacturing test. This is the meaning of 'D' in the Default field. The trim value can be adjusted vs. voltage as noted in [Table 36 on page 24](#).

**Bit 7** 32 kHz Low Power  
 0 = The 32 kHz Low speed Oscillator operates in normal mode.  
 1 = The 32 kHz Low speed Oscillator operates in a low power mode. The oscillator continues to function normally but with reduced accuracy.

**Bit 6** Reserved

**Bits [5:4]** 32 kHz Bias Trim [1:0]  
 These bits control the bias current of the low power oscillator.  
 0 0 = Mid bias  
 0 1 = High bias  
 1 0 = Reserved  
 1 1 = Reserved

**Important Note** Do not program the 32 kHz Bias Trim [1:0] field with the reserved 10b value as the oscillator does not oscillate at all corner conditions with this setting.

**Bits 3:0** 32 kHz Freq Trim [3:0]  
 These bits are used to trim the frequency of the low power oscillator.

**CPU Clock During Sleep Mode**

When the CPU enters sleep mode, the oscillator is stopped. When the CPU comes out of sleep mode it is running on the internal oscillator. The internal oscillator recovery time is three clock cycles of the Internal 32 kHz Low power Oscillator.

## Reset

The microcontroller supports two types of resets: power-on reset (POR) and watchdog reset (WDR). When reset is initiated, all registers are restored to their default states and all interrupts are disabled.

The occurrence of a reset is recorded in the System Status and Control Register (CPU\_SCR). Bits within this register record the occurrence of POR and WDR Reset respectively. The firmware can interrogate these bits to determine the cause of a reset.

The microcontroller resumes execution from Flash address 0x0000 after a reset. The internal clocking mode is active after a reset, until changed by user firmware.

**Note** The CPU clock defaults to 3 MHz (Internal 24 MHz Oscillator divide-by-8 mode) at POR to guarantee operation at the low  $V_{CC}$  that might be present during the supply ramp.

**Table 41. System Status and Control Register (CPU\_SCR) [0xFF] [R/W]**

| Bit #      | 7    | 6        | 5                  | 4                  | 3     | 2        | 1        | 0    |
|------------|------|----------|--------------------|--------------------|-------|----------|----------|------|
| Field      | GIES | Reserved | WDRS               | PORS               | Sleep | Reserved | Reserved | Stop |
| Read/Write | R    | –        | R/C <sup>[3]</sup> | R/C <sup>[3]</sup> | R/W   | –        | –        | R/W  |
| Default    | 0    | 0        | 0                  | 1                  | 0     | 1        | 0        | 0    |

The bits of the CPU\_SCR register are used to convey status and control of events for various functions of a CYRF69303 device.

### Bit 7 GIES

The Global Interrupt Enable Status bit is a read-only status bit and its use is discouraged. The GIES bit is a legacy bit, which was used to provide the ability to read the GIE bit of the CPU\_F register. However, the CPU\_F register is now readable. When this bit is set, it indicates that the GIE bit in the CPU\_F register is also set which, in turn, indicates that the microprocessor services interrupts:

- 0 = Global interrupts disabled
- 1 = Global interrupt enabled

### Bit 6 Reserved

### Bit 5 WDRS

The WDRS bit is set by the CPU to indicate that a WDR event has occurred. The user can read this bit to determine the type of reset that has occurred. The user can clear but not set this bit:

- 0 = No WDR
- 1 = A WDR event has occurred

### Bit 4 PORS

The PORS bit is set by the CPU to indicate that a POR event has occurred. The user can read this bit to determine the type of reset that has occurred. The user can clear but not set this bit:

- 0 = No POR
- 1 = A POR event has occurred (Note that WDR events do not occur until this bit is cleared).

### Bit 3 SLEEP

Set by the user to enable CPU sleep state. CPU remains in sleep mode until any interrupt is pending. The Sleep bit is covered in more detail in the section [Sleep Mode on page 30](#).

- 0 = Normal operation
- 1 = Sleep

### Bits 2:1 Reserved

### Bit 0 STOP

This bit is set by the user to halt the CPU. The CPU remains halted until a reset (WDR, POR, or external reset) has taken place. If an application wants to stop code execution until a reset, the preferred method is to use the HALT instruction rather than writing to this bit.

- 0 = Normal CPU operation
- 1 = CPU is halted (not recommended)

**Note**

- 3. C = Clear. This bit can only be cleared by the user and cannot be set by firmware.

### Power-on Reset

POR occurs every time the power to the device is switched on. POR is released when the supply is typically 2.6 V for the upward supply transition, with typically 50 mV of hysteresis during the power on transient. Bit 4 of the System Status and Control Register (CPU\_SCR) is set to record this event (the register contents are set to 00010000 by the POR). After a POR, the microprocessor is held off for approximately 20 ms for the V<sub>CC</sub> supply to stabilize before executing the first instruction at address 0x00 in the Flash. If the V<sub>CC</sub> voltage drops below the POR downward supply trip point, POR is reasserted. The V<sub>CC</sub> supply needs to ramp linearly from 0 to V<sub>CC</sub> in 0 to 200 ms.

**Important** The PORS status bit is set at POR and can only be cleared by the user, and cannot be set by firmware.

### Watchdog Timer Reset

The user has the option to enable the WDT. The WDT is enabled by clearing the PORS bit. When the PORS bit is cleared, the

WDT cannot be disabled. The only exception to this is if a POR event takes place, which disables the WDT.

The sleep timer is used to generate the sleep time period and the Watchdog time period. The sleep timer uses the Internal 32 kHz Low power Oscillator system clock to produce the sleep time period. The user can program the sleep time period using the Sleep Timer bits of the OSC\_CR0 Register (Table 37 on page 25). When the sleep time elapses (sleep timer overflows), an interrupt to the Sleep Timer Interrupt Vector is generated.

The Watchdog Timer period is automatically set to be three counts of the Sleep Timer overflows. This represents between two and three sleep intervals depending on the count in the Sleep Timer at the previous WDT clear. When this timer reaches three, a WDR is generated.

The user can either clear the WDT, or the WDT and the Sleep Timer. Whenever the user writes to the Reset WDT Register (RES\_WDT), the WDT is cleared. If the data that is written is the hex value 0x38, the Sleep Timer is also cleared at the same time.

**Table 42. Reset Watchdog Timer (RESWDT) [0xE3] [W]**

| Bit #      | 7                          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----------------------------|---|---|---|---|---|---|---|
| Field      | Reset Watchdog Timer [7:0] |   |   |   |   |   |   |   |
| Read/Write | W                          | W | W | W | W | W | W | W |
| Default    | 0                          | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Any write to this register clears the Watchdog Timer, a write of 0x38 also clears the Sleep Timer.

**Bits 7:0**Reset Watchdog Timer [7:0]

### Sleep Mode

The CPU can only be put to sleep by the firmware. This is accomplished by setting the Sleep bit in the System Status and Control Register (CPU\_SCR). This stops the CPU from executing instructions, and the CPU remains asleep until an interrupt comes pending, or there is a reset event (either a Power on Reset, or a Watchdog Timer Reset).

The Internal 32 kHz low speed oscillator remains running. Before entering suspend mode, firmware can optionally configure the 32 kHz Low speed Oscillator to operate in a low power mode to help reduce the overall power consumption (using the 32 kHz Low Power bit, Table 40). This helps save approximately 5 μA; however, the trade off is that the 32 kHz Low speed Oscillator be less accurate (-53.12% to +56.25% deviation).

All interrupts remain active. Only the occurrence of an interrupt wakes the part from sleep. The Stop bit in the System Status and Control Register (CPU\_SCR) must be cleared for a part to resume out of sleep. The Global Interrupt Enable bit of the CPU Flags Register (CPU\_F) does not have any effect. Any unmasked interrupt wakes the system up. As a result, any interrupts not intended for waking must be disabled through the Interrupt Mask Registers.

When the CPU enters sleep mode, the internal oscillator is stopped. When the CPU comes out of sleep mode, it is running on the internal oscillator. The internal oscillator recovery time is three clock cycles of the Internal 32 kHz Low power Oscillator.

On exiting sleep mode, when the clock is stable and the delay time has expired, the instruction immediately following the sleep instruction is executed before the interrupt service routine (if enabled).

The Sleep interrupt allows the microcontroller to wake up periodically and poll system components while maintaining very low average power consumption. The Sleep interrupt may also be used to provide periodic interrupts during non sleep modes.

### Sleep Sequence

The Sleep bit is an input into the sleep logic circuit. This circuit is designed to sequence the device into and out of the hardware sleep state. The hardware sequence to put the device to sleep is shown in Figure 9 on page 31 and is defined as follows.

1. Firmware sets the SLEEP bit in the CPU\_SCR0 register. The Bus Request (BRQ) signal to the CPU is immediately asserted. This is a request by the system to halt CPU operation at an instruction boundary. The CPU samples BRQ on the positive edge of CPUCLK.
2. Due to the specific timing of the register write, the CPU issues a Bus Request Acknowledge (BRA) on the following positive edge of the CPU clock. The sleep logic waits for the following negative edge of the CPU clock and then asserts a system-wide Power-down (PD) signal. In Figure 9 on page 31 the CPU is halted and the system-wide power-down signal is asserted.

3. The system-wide PD (power-down) signal controls several major circuit blocks: The Flash memory module, the internal 24 MHz oscillator, the EFTB filter and the bandgap voltage reference. These circuits transition into a zero power state. The only operational circuits on chip are the Low Power oscillator, the bandgap refresh circuit, and the supply voltage monitor (POR) circuit.

**Low Power in Sleep Mode**

To achieve the lowest possible power consumption during suspend or sleep, the following conditions are observed in addition to considerations for the sleep timer:

- All GPIOs are set to outputs and driven low
- Clear P11CR[0], P10CR[0]

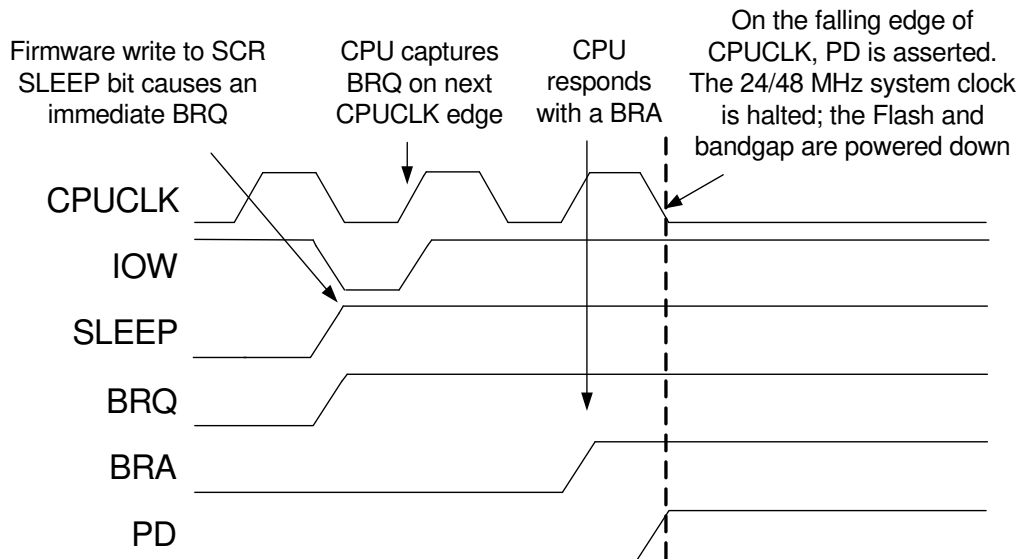
- Set P10CR[1]
- To avoid current consumption make sure ITMRCLK and TCPCLK are not sourced by either low power 32 kHz oscillator or 24 MHz crystal-less oscillator.

All the other blocks go to the power-down mode automatically on suspend.

The following steps are user configurable and help in reducing the average suspend mode power consumption:

1. Configure the power supply monitor at a large regular intervals, control register bits are 1,EB[7:6] (Power system sleep duty cycle PSSDC[1:0]).
2. Configure the Low power oscillator into low power mode, control register bit is LOPSCTR[7].

**Figure 9. Sleep Timing**

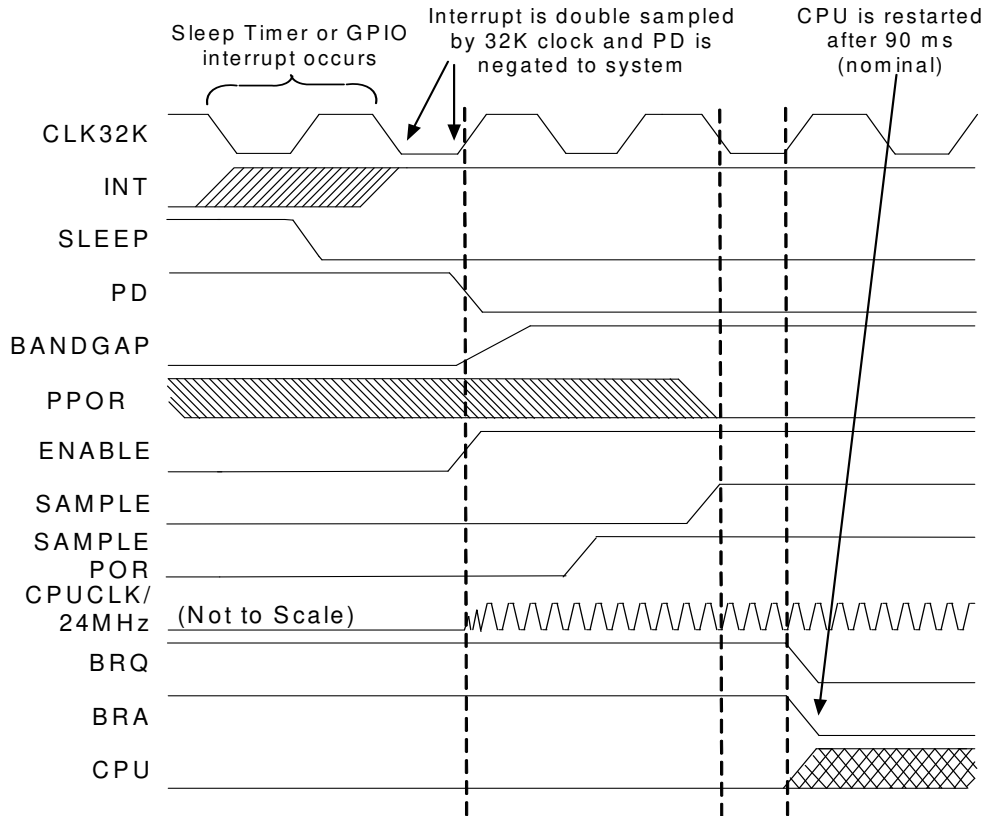


**Wakeup Sequence**

When asleep, the only event that can wake the system up is an interrupt. The global interrupt enable of the CPU flag register does not need to be set. Any unmasked interrupt wakes the system up. It is optional for the CPU to actually take the interrupt after the wakeup sequence. The wakeup sequence is synchronized to the 32 kHz clock. This is done to sequence a startup delay and enable the Flash memory module enough time to power-up before the CPU asserts the first read access. Another reason for the delay is to enable the oscillator, Bandgap, and POR circuits time to settle before actually being used in the system. As shown in Figure 10, the wakeup sequence is as follows:

1. The wakeup interrupt occurs and is synchronized by the negative edge of the 32 kHz clock.
2. At the following positive edge of the 32 kHz clock, the system-wide PD signal is negated. The Flash memory module, internal oscillator, EFTB, and bandgap circuit are all powered up to a normal operating state.
3. At the following positive edge of the 32 kHz clock, the current values for the precision POR have settled and are sampled.
4. At the following negative edge of the 32 kHz clock (after about 15 μs nominal), the BRQ signal is negated by the sleep logic circuit. On the following CPUCLK, BRA is negated by the CPU and instruction execution resumes. Note that in Figure 10 fixed function blocks, such as Flash, internal oscillator, EFTB, and bandgap, have about 15 μs start up. The wakeup times (interrupt to CPU operational) ranges from 75 μs to 105 μs.

**Figure 10. Wakeup Timing**



**Power-on Reset Control**

**Table 43. Power-on Reset Control Register (POR CR) [0x1E3] [R/W]**

| Bit #      | 7        | 6 | 5           | 4   | 3        | 2 | 1 | 0 |
|------------|----------|---|-------------|-----|----------|---|---|---|
| Field      | Reserved |   | PORLEV[1:0] |     | Reserved |   |   |   |
| Read/Write | -        | - | R/W         | R/W | -        | - | - | - |
| Default    | 0        | 0 | 0           | 0   | 0        | 0 | 0 | 0 |

This register controls the configuration of the Power on Reset circuit. This register can only be accessed in the second bank of I/O space. This requires setting the XIO bit in the CPU flags register.

**Bits 7:6** Reserved

**Bits 5:4** PORLEV[1:0]

This field controls the level below which the precision power on reset (PPOR) detector generates a reset

0 0 = 2.7 V Range (trip near 2.6 V)

0 1 = 3 V Range (trip near 2.9 V)

1 0 = Reserved

1 1 = PPOR does not generate a reset, but values read from the Voltage Monitor Comparators Register (Table 44) give the internal PPOR comparator state with trip point set to the 3 V range setting.

**Bits 3:0** Reserved



### POR Compare State

**Table 44. Voltage Monitor Comparators Register (VLTCMP) [0x1E4] [R]**

| Bit #      | 7        | 6 | 5 | 4 | 3 | 2 | 1 | 0    |
|------------|----------|---|---|---|---|---|---|------|
| Field      | Reserved |   |   |   |   |   |   | PPOR |
| Read/Write | –        | – | – | – | – | – | – | R    |
| Default    | 0        | 0 | 0 | 0 | 0 | 0 | 0 | 0    |

This read-only register allows reading the current state of the Precision-Power-On-Reset comparators:

**Bits 7:1** Reserved

**Bit 0** PPOR

This bit is set to indicate that the precision-power-on-reset comparator has tripped, indicating that the supply voltage is below the trip point set by PORLEV[1:0]:

0 = No precision-power-on-reset event

1 = A precision-power-on-reset event has tripped

**Note** This register can only be accessed in the second bank of I/O space. This requires setting the XIO bit in the CPU flags register

### ECO Trim Register

**Table 45. ECO (ECO\_TR) [0x1EB] [R/W]**

| Bit #      | 7                      | 6   | 5        | 4 | 3 | 2 | 1 | 0 |
|------------|------------------------|-----|----------|---|---|---|---|---|
| Field      | Sleep Duty Cycle [1:0] |     | Reserved |   |   |   |   |   |
| Read/Write | R/W                    | R/W | –        | – | – | – | – | – |
| Default    | 0                      | 0   | 0        | 0 | 0 | 0 | 0 | 0 |

This register controls the ratios (in numbers of 32 kHz clock periods) of ‘on’ time versus ‘off’ time for POR detection circuit.

**Bits 7:6** Sleep Duty Cycle [1:0]

0 0 = 1/128 periods of the Internal 32 kHz low-speed Oscillator

0 1 = 1/512 periods of the Internal 32 kHz low-speed Oscillator

1 0 = 1/32 periods of the Internal 32 kHz low-speed Oscillator

1 1 = 1/8 periods of the Internal 32 kHz low speed Oscillator

**Note** This register can only be accessed in the second bank of I/O space. This requires setting the XIO bit in the CPU flags register

### General-Purpose I/O Ports

The general-purpose I/O ports are discussed in the following sections.

#### Port Data Registers

**Table 46. P0 Data Register (P0DATA)[0x00] [R/W]**

| Bit #      | 7    | 6        | 5 | 4         | 3         | 2        | 1    | 0        |
|------------|------|----------|---|-----------|-----------|----------|------|----------|
| Field      | P0.7 | Reserved |   | P0.4/INT2 | P0.3/INT1 | Reserved | P0.1 | Reserved |
| Read/Write | R/W  | -        |   | R/W       | R/W       | -        | R/W  | -        |
| Default    | 0    | -        | - | 0         | 0         | 0        | 0    | -        |

This register contains the data for Port 0. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 0 pins.

**Bit 7** P0.7 Data

**Bits 6:5** Reserved

**Bits 4:3** P0.4–P0.3Data/INT2–INT1

In addition to their use as the P0.4–P0.3 GPIOs, these pins can also be used for the alternative functions as the Interrupt pins (INT1–INT2). To configure the P0.4–P0.3 pins, refer to the P0.3/INT1–P0.4/INT2 Configuration Register (Table 50 on page 37).

**Bit 2** Reserved

**Bit 1** P0.1 Data

**Bit 0** Reserved

**Table 47. P1 Data Register (P1DATA) [0x01] [R/W]**

| Bit #      | 7    | 6    | 5          | 4         | 3         | 2    | 1    | 0    |
|------------|------|------|------------|-----------|-----------|------|------|------|
| Field      | P1.7 | P1.6 | P1.5/SMOSI | P1.4/SCLK | P1.3/SSEL | P1.2 | P1.1 | P1.0 |
| Read/Write | R/W  | R/W  | R/W        | R/W       | R/W       | R/W  | R/W  | R/W  |
| Default    | 0    | 0    | 0          | 0         | 0         | 0    | 0    | -    |

This register contains the data for Port 1. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 1 pins.

**Bits 7** P1.7

**Bits 6** P1.6 or alternate function of SMOSI in a 4-wire SPI

**Bits 5:3** P1.5–P1.3 Data/3-wire SPI Pins (SMISO/SMOSI, SCLK, SSEL)

In addition to their use as the P1.6–P1.3 GPIOs, these pins can also be used for the alternative function as the SPI interface pins. To configure the P1.6–P1.3 pins, refer to the P1.3–P1.6 Configuration Register (Table 50 on page 35)

**Bits 2:1** P1.2–P1.1

**Bit 0** P1.0

**Table 48. P2 Data Register (P2DATA) [0x02] [R/W]**

| Bit #      | 7        | 6 | 5 | 4 | 3 | 2 | 1         | 0   |
|------------|----------|---|---|---|---|---|-----------|-----|
| Field      | Reserved |   |   |   |   |   | P2.1–P2.0 |     |
| Read/Write | -        |   |   |   |   |   | R/W       | R/W |
| Default    | -        |   |   |   |   |   | 0         | 0   |

This register contains the data for Port 2. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 2 pins.

**Bits 7:2** P2 Data [7:2]

**Bits 1:0** P2 Data [1:0]

### GPIO Port Configuration

All the GPIO configuration registers have common configuration controls. The following are the bit definitions of the GPIO configuration registers. By default all GPIOs are configured as inputs. To prevent the inputs from floating, the pull-up resistors are enabled. Firmware needs to configure each of the GPIOs before use.

#### Int Enable

When set, the Int Enable bit allows the GPIO to generate interrupts. Interrupt generate can occur regardless of whether the pin is configured for input or output. All interrupts are edge sensitive, however for any interrupt that is shared by multiple sources (that is, Ports 2, 3, and 4) all inputs must be deasserted before a new interrupt can occur.

When clear, the corresponding interrupt is disabled on the pin.

It is possible to configure GPIOs as outputs, enable the interrupt on the pin and then to generate the interrupt by driving the appropriate pin state. This is useful in test and may have value in applications.

#### Int Act Low

When clear, the corresponding interrupt is active HIGH. When set, the interrupt is active LOW. For P0.3–P0.4 Int act Low clear causes interrupts to be active on the rising edge. Int act Low set causes interrupts to be active on the falling edge.

#### TTL Thresh

When set, the input has TTL threshold. When clear, the input has standard CMOS threshold.

**Important Note** The GPIOs default to CMOS threshold. User's firmware needs to configure the threshold to TTL mode if necessary.

#### High Sink

When set, the output can sink up to 50 mA.

When clear, the output can sink up to 8 mA.

On the CY7C601xx, only the P3.7, P2.7, P0.1, and P0.0 have 50 mA sink drive capability. Other pins have 8 mA sink drive capability.

On the CY7C602xx, only the P1.7–P1.3 have 50 mA sink drive capability. Other pins have 8 mA sink drive capability.

#### Open Drain

When set, the output on the pin is determined by the Port Data Register. If the corresponding bit in the Port Data Register is set, the pin is in high impedance state. If the corresponding bit in the Port Data Register is clear, the pin is driven LOW.

When clear, the output is driven LOW or HIGH.

#### Pull-up Enable

When set the pin has a 7K pull-up to V<sub>DD</sub>.

When clear, the pull-up is disabled.

### Output Enable

When set, the output driver of the pin is enabled.

When clear, the output driver of the pin is disabled.

For pins with shared functions there are some special cases.

P0.0 (CLKIN) and P0.1 (CLKOUT) can not be output enabled when the crystal oscillator is enabled. Output enables for these pins are overridden by XOSC Enable.

P1.3 (SSEL), P1.4 (SCLK), P1.5 (SMOSI) and P1.6 (SMISO) can be used for their dedicated functions or for GPIO. To enable

the pin for GPIO use, clear the corresponding SPI Use bit or the Output Enable has no effect.

### SPI Use

The P1.3 (SSEL), P1.4 (SCLK), P1.5 (SMOSI) and P1.6 (SMISO) pins can be used for their dedicated functions or for GPIO. To enable the pin for GPIO, clear the corresponding SPI Use bit. The SPI function controls the output enable for its dedicated function pins when their GPIO enable bit is clear.

**Table 49. P0.1 Configuration (P01CR) [0x06] R/W**

| Bit #      | 7        | 6          | 5           | 4          | 3         | 2          | 1              | 0             |
|------------|----------|------------|-------------|------------|-----------|------------|----------------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | TTL Thresh | High Sink | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | R/W      | R/W        | R/W         | R/W        | R/W       | R/W        | R/W            | R/W           |
| Default    | 0        | 0          | 0           | 0          | 0         | 0          | 0              | 0             |

This register is used to configure P0.1. In the CYRF69303, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.

If this pin is used as a general purpose output it draws current. This pin must be configured as an input to reduce current draw.

- Bit 7** Reserved
- Bit 6** see Section *Int Enable*
- Bit 5** see Section *Int Act Low*/*Int Act Low*
- Bit 4** see Section *TTL Thresh*
- Bit 3** see Section *High Sink*
- Bit 2** see Section *Open Drain*/*Open Drain*
- Bit 1** see Section *Pull-up Enable*
- Bit 0** see Section *Output Enable*

**Table 50. P0.3–P0.4 Configuration (P03CR–P04CR) [0x08–0x09] [R/W]**

| Bit #      | 7        | 6 | 5           | 4          | 3        | 2          | 1              | 0             |
|------------|----------|---|-------------|------------|----------|------------|----------------|---------------|
| Field      | Reserved |   | Int Act Low | TTL Thresh | Reserved | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | –        | – | R/W         | R/W        | –        | R/W        | R/W            | R/W           |
| Default    | 0        | 0 | 0           | 0          | 0        | 0          | 0              | 0             |

These registers control the operation of pins P0.3–P0.4 respectively. These pins are shared between the P0.3–P0.4 GPIOs and the INT1–INT2. The INT1–INT2 interrupts are different than all the other GPIO interrupts. These pins are connected directly to the interrupt controller to provide three edge-sensitive interrupts with independent interrupt vectors. These interrupts occur on a rising edge when Int act Low is clear and on a falling edge when Int act Low is set. These pins are enabled as interrupt sources in the interrupt controller registers ([Table 76](#) on page 53 and [Table 74](#) on page 52).

To use these pins as interrupt inputs, configure them as inputs by clearing the corresponding Output Enable. If the INT1–INT2 pins are configured as outputs with interrupts enabled, firmware can generate an interrupt by writing the appropriate value to the P0.3, and P0.4 data bits in the P0 Data Register.

Regardless of whether the pins are used as Interrupt or GPIO pins the Int Enable, Int act Low, TTL Threshold, Open Drain, and Pull-up Enable bits control the behavior of the pin.

The P0.3/INT1–P0.4/INT2 pins are individually configured with the P03CR (0x08), and P04CR (0x09) respectively.

**Note** Changing the state of the Int Act Low bit can cause an unintentional interrupt to be generated. When configuring these interrupt sources, it is best to follow the following procedure:

1. Disable interrupt source
2. Configure interrupt source
3. Clear any pending interrupts from the source
4. Enable interrupt source

**Table 51. P0.7 Configuration (P07CR) [0x0C] [R/W]**

| Bit #      | 7        | 6          | 5           | 4          | 3        | 2          | 1              | 0             |
|------------|----------|------------|-------------|------------|----------|------------|----------------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | TTL Thresh | Reserved | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | –        | R/W        | R/W         | R/W        | –        | R/W        | R/W            | R/W           |
| Default    | 0        | 0          | 0           | 0          | 0        | 0          | 0              | 0             |

This register controls the operation of pin P0.7.

- Bit 7** Reserved
- Bit 6** see Section *Int Enable*
- Bit 5** see Section *Int Act Low*
- Bit 4** see Section *TTL ThreshHigh Sink*
- Bit 3** Reserved
- Bit 2** see Section *Open Drain*
- Bit 1** see Section *Pull-up Enable*
- Bit 0** see Section *Output Enable*

**Table 52. P1.0 Configuration (P10CR) [0x0D] [R/W]**

| Bit #      | 7        | 6          | 5           | 4        | 3        | 2        | 1                | 0             |
|------------|----------|------------|-------------|----------|----------|----------|------------------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | Reserved | Reserved | Reserved | 5K pullup Enable | Output enable |
| Read/Write | R/W      | R/W        | R/W         | -        | -        | -        | R/W              | R/W           |
| Default    | 0        | 0          | 0           | 0        | 0        | 0        | 0                | 0             |

This register controls the operation of the P1.0 pin.

**Note** The P1.0 is an open drain only output. It can actively drive a signal low, but cannot actively drive a signal high.

**Bit 0** This bit enables the output on P1.0. This bit must be cleared in sleep mode.

- Bit 7** Reserved
- Bit 6** see Section *Int Enable*
- Bit 5** see Section *Int Act Low*
- Bit 4** Reserved
- Bit 3** Reserved
- Bit 2** Reserved
- Bit 1** 0 = disables the 5K ohm pull-up resistors  
1 = enables 5K ohm pull-up resistors for both P1.0 and P1.1 (this is not compatible with USB)

**Table 53. P1.1 Configuration (P11CR) [0x0E] [R/W]**

| Bit #      | 7        | 6          | 5           | 4        | 3 | 2          | 1        | 0             |
|------------|----------|------------|-------------|----------|---|------------|----------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | Reserved |   | Open Drain | Reserved | Output Enable |
| Read/Write | –        | R/W        | R/W         | –        | – | R/W        | –        | R/W           |
| Default    | 0        | 0          | 0           | 0        | 0 | 0          | 0        | 0             |

This register controls the operation of the P1.1 pin.

The pull-up resistor on this pin is enabled by the P10CR Register.

**Note** There is no 2 mA sourcing capability on this pin. The pin can only sink 5 mA at V<sub>OL3</sub> section.

- Bit 7**      Reserved
- Bit 6**      see *Section Int Enable*
- Bit 5**      see *Section Int Act Low*
- Bit 4**      Reserved
- Bit 3**      Reserved
- Bit 2**      see *Section Open Drain*
- Bit 1**      Reserved
- Bit 0**      see *Section Output Enable*

**Table 54. P1.2 Configuration (P12CR) [0x0F] [R/W]**

| Bit #      | 7          | 6          | 5           | 4             | 3        | 2          | 1              | 0             |
|------------|------------|------------|-------------|---------------|----------|------------|----------------|---------------|
| Field      | CLK Output | Int Enable | Int Act Low | TTL Threshold | Reserved | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | R/W        | R/W        | R/W         | R/W           | –        | R/W        | R/W            | R/W           |
| Default    | 0          | 0          | 0           | 0             | 0        | 0          | 0              | 0             |

This register controls the operation of the P1.2.

- Bit 7**      CLK Output  
0 = The internally selected clock is not sent out onto P1.2 pin.  
1 = When CLK Output is set, the internally selected clock is sent out onto P1.2 pin.
- Bit 6**      see *Section Int Enable*
- Bit 5**      see *Section Int Act Low*
- Bit 4**      Reserved
- Bit 3**      see *Section High Sink*
- Bit 2**      see *Section Open Drain*
- Bit 1**      see *Section Pull-up Enable*
- Bit 0**      see *Section Output Enable*

**Table 55. P1.3 Configuration (P13CR) [0x10] [R/W]**

| Bit #      | 7        | 6          | 5           | 4        | 3         | 2          | 1              | 0             |
|------------|----------|------------|-------------|----------|-----------|------------|----------------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | Reserved | High Sink | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | –        | R/W        | R/W         | –        | R/W       | R/W        | R/W            | R/W           |
| Default    | 0        | 0          | 0           | 0        | 0         | 0          | 0              | 0             |

This register controls the operation of the P1.3 pin.

The P1.3 GPIO's threshold is always set to TTL.

When the SPI hardware is enabled, the output enable and output state of the pin is controlled by the SPI circuitry. When the SPI hardware is disabled, the pin is controlled by the Output Enable bit and the corresponding bit in the P1 data register.

Regardless of whether the pin is used as an SPI or GPIO pin the Int Enable, Int act Low, High Sink, Open Drain, and Pull-up Enable control the behavior of the pin.

50 mA sink drive capability is available.

- Bit 7** Reserved
- Bit 6** see *Section Int Enable*
- Bit 5** see *Section Int Act Low*
- Bit 4** Reserved
- Bit 3** see *Section High Sink*
- Bit 2** see *Section Open Drain*
- Bit 1** see *Section Pull-up Enable*
- Bit 0** see *Section Output Enable*



**Table 56. P1.4–P1.6 Configuration (P14CR–P16CR) [0x11–0x13] [R/W]**

| Bit #      | 7       | 6          | 5           | 4        | 3         | 2          | 1              | 0             |
|------------|---------|------------|-------------|----------|-----------|------------|----------------|---------------|
| Field      | SPI Use | Int Enable | Int Act Low | Reserved | High Sink | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | R/W     | R/W        | R/W         | –        | R/W       | R/W        | R/W            | R/W           |
| Default    | 0       | 0          | 0           | 0        | 0         | 0          | 0              | 0             |

These registers control the operation of pins P1.4–P1.6, respectively.

The P1.4–P1.6 GPIO's threshold is always set to TTL.

When the SPI hardware is enabled, pins that are configured as SPI Use have their output enable and output state controlled by the SPI circuitry. When the SPI hardware is disabled or a pin has its SPI Use bit clear, the pin is controlled by the Output Enable bit and the corresponding bit in the P1 data register.

Regardless of whether any pin is used as an SPI or GPIO pin the Int Enable, Int act Low, High Sink, Open Drain, and Pull-up Enable control the behavior of the pin.

The 50 mA sink drive capability is only available in the CY7C602xx. In the CY7C601xx, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.

- Bit 7** SPI Use  
0 = Disable the SPI alternate function. The pin is used as a GPIO  
1 = Enable the SPI function. The SPI circuitry controls the output of the pin
- Bit 6** see *Section Int Enable*
- Bit 5** see *Section Int Act Low*
- Bit 4** Reserved
- Bit 3** see *Section High Sink*
- Bit 2** see *Section Open Drain*
- Bit 1** see *Section Pull-up Enable*
- Bit 0** see *Section Output Enable*

**Note** For Comm Modes 01 or 10 (SPI Master or SPI Slave, see [Table 60](#) on page 44)

When configured for SPI (SPI Use = 1 and Comm Modes [1:0] = SPI Master or SPI Slave mode), the input/output direction of pins P1.3, P1.5, and P1.6 is set automatically by the SPI logic. However, pin P1.4's input/output direction is NOT automatically set; it must be explicitly set by firmware. For SPI Master mode, pin P1.4 must be configured as an output; for SPI Slave mode, pin P1.4 must be configured as an input.

**Table 57. P1.7 Configuration (P17CR) [0x14] [R/W]**

| Bit #      | 7        | 6          | 5           | 4        | 3         | 2          | 1              | 0             |
|------------|----------|------------|-------------|----------|-----------|------------|----------------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | Reserved | High Sink | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | –        | R/W        | R/W         | –        | R/W       | R/W        | R/W            | R/W           |
| Default    | 0        | 0          | 0           | 0        | 0         | 0          | 0              | 0             |

This register controls the operation of pin P1.7.

50 mA sink drive capability is available. The P1.7 GPIO's threshold is always set to TTL.

- Bit 7** Reserved
- Bit 6** see *Section Int Enable*
- Bit 5** see *Section Int Act Low*
- Bit 4** Reserved
- Bit 3** see *Section High Sink*
- Bit 2** see *Section Open Drain*
- Bit 1** see *Section Pull-up Enable*
- Bit 0** see *Section Output Enable*

**Table 58. P2 Configuration (P2CR) [0x15] [R/W]**

| Bit #      | 7        | 6          | 5           | 4          | 3         | 2          | 1              | 0             |
|------------|----------|------------|-------------|------------|-----------|------------|----------------|---------------|
| Field      | Reserved | Int Enable | Int Act Low | TTL Thresh | High Sink | Open Drain | Pull-up Enable | Output Enable |
| Read/Write | –        | R/W        | R/W         | R/W        | R/W       | R/W        | R/W            | R/W           |
| Default    | 0        | 0          | 0           | 0          | 0         | 0          | 0              | 0             |

This register controls the operation of pins P2.0–P2.1.

- Bit 7** Reserved
- Bit 6** see *Section Int Enable*
- Bit 5** see *Section Int Act LowTTL Thresh*
- Bit 4** see *Section TTL Thresh*
- Bit 3** see *Section High Sink*
- Bit 2** see *Section Open DrainPull-up Enable*
- Bit 1** see *Section Pull-up Enable*
- Bit 0** see *Section Output Enable*

### GPIO Configurations for Low Power Mode

To ensure low power mode, unbonded GPIO pins in CYRF69303 must be placed in a non-floating state. The following assembly code snippet shows how this is achieved. This snippet can be added as a part of the initialization routine.

```

//Code Snippet for addressing unbonded GPIOs

mov A, 01h
mov reg[1Fh],A
mov A, 01h
mov reg[16h],A // Port3 Configuration register - Enable output
mov A, 00h
mov reg[03h],A // Asserting P3.0 to P3.7 outputs to '0'
//Port 2 configurations
mov A, 01h
mov reg[15h],A //Port 2 Configuration register -Enable output
mov A, 00h
mov reg[02h],A //Asserting P2.0 to P2.7 outputs to '0'
mov A, 01h
mov reg[05h],A // Port0.0 Configuration register - Enable output
mov reg[07h],A // Port0.2 Configuration register - Enable output
mov reg[0Ah],A // Port0.5 Configuration register - Enable output
mov reg[0Bh],A // Port0.6 Configuration register - Enable output
mov A,reg[00h]
mov A,00h
and A,9Ah
mov reg[00h], A // Asserting outputs '0' to pins in port 1
// NOTE: The code fragment in italics is to be used only if your application configures P2.0 and
P2.1 as push-pull outputs.

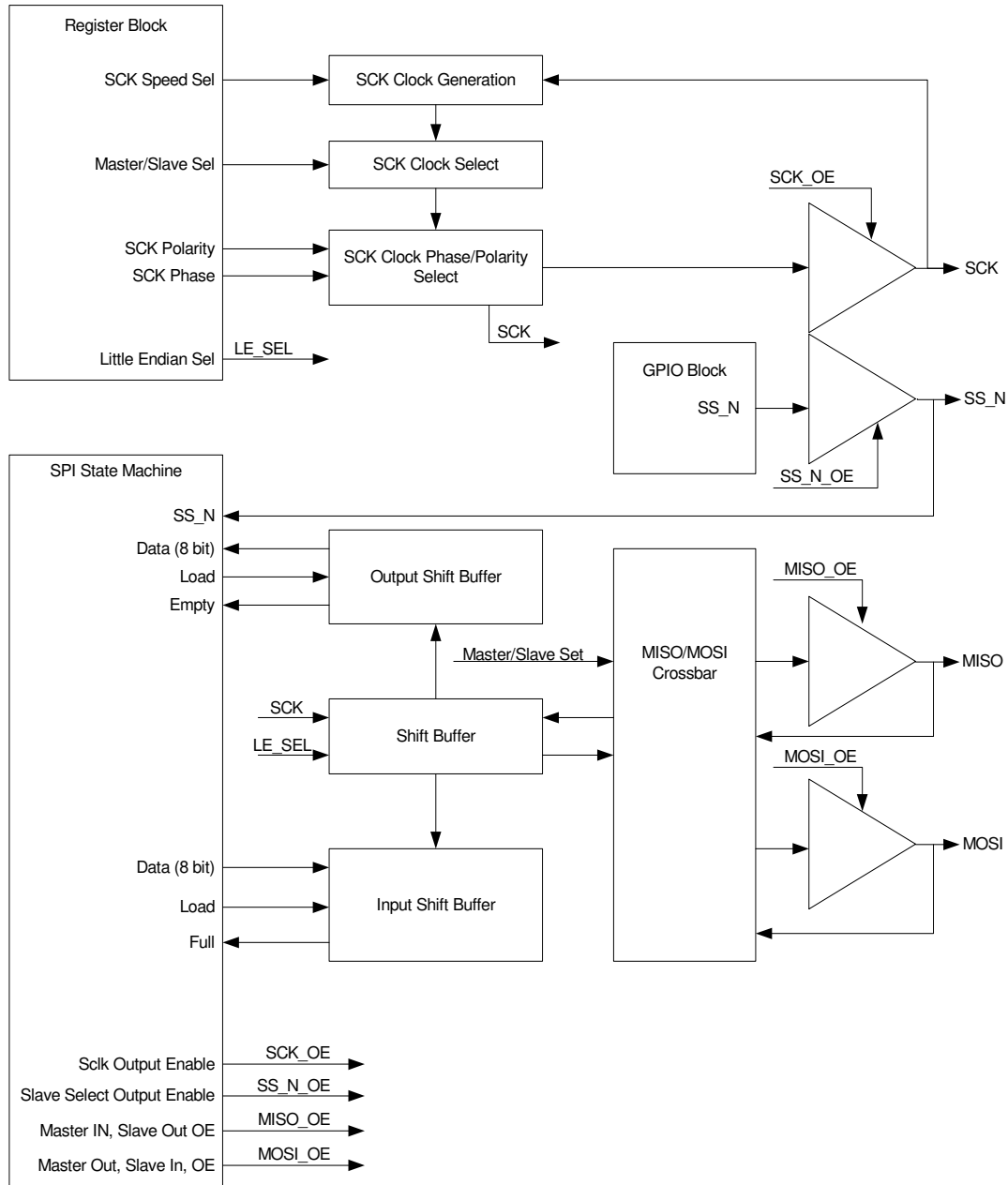
```

When writing to port 0, to access GPIOs P0.1,3,4,7, mask bits 0,2,5,6. Failing to do so voids the low power.

### Serial Peripheral Interface (SPI)

The SPI Master/Slave Interface core logic runs on the SPI clock domain. The SPI clock is a divider off of the CPUCLK when in Master Mode. SPI is a four-pin serial interface comprised of a clock, an enable, and two data pins.

Figure 11. SPI Block Diagram



**SPI Data Register**

**Table 59. SPI Data Register (SPIDATA) [0x3C] [R/W]**

| Bit #      | 7            | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------------|--------------|-----|-----|-----|-----|-----|-----|-----|
| Field      | SPIData[7:0] |     |     |     |     |     |     |     |
| Read/Write | R/W          | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Default    | 0            | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

When read, this register returns the contents of the receive buffer. When written, it loads the transmit holding register.

**Bits 7:0** SPI Data [7:0]

When an interrupt occurs to indicate to firmware that a byte of receive data is available, or the transmitter holding register is empty, firmware has 7 SPI clocks to manage the buffers—to empty the receiver buffer, or to refill the transmit holding register. Failure to meet this timing requirement results in incorrect data transfer.

**SPI Configure Register**

**Table 60. SPI Configure Register (SPICR) [0x3D] [R/W]**

| Bit #      | 7    | 6         | 5         | 4   | 3    | 2    | 1           | 0   |
|------------|------|-----------|-----------|-----|------|------|-------------|-----|
| Field      | Swap | LSB First | Comm Mode |     | CPOL | CPHA | SCLK Select |     |
| Read/Write | R/W  | R/W       | R/W       | R/W | R/W  | R/W  | R/W         | R/W |
| Default    | 0    | 0         | 0         | 0   | 0    | 0    | 0           | 0   |

**Bit 7** Swap

0 = Swap function disabled.

1 = The SPI block swaps its use of SMOSI and SMISO. Among other things, this can be useful in implementing single wire SPI-like communications.

**Bit 6** LSB First

0 = The SPI transmits and receives the MSB (Most Significant Bit) first.

1 = The SPI transmits and receives the LSB (Least Significant Bit) first.

**Bits 5:4** Comm Mode [1:0]

0 0: All SPI communication disabled.

0 1: SPI master mode

1 0: SPI slave mode

1 1: Reserved

**Bit 3** CPOL

This bit controls the SPI clock (SCLK) idle polarity.

0 = SCLK idles low

1 = SCLK idles high

**Bit 2** CPHA

The Clock Phase bit controls the phase of the clock on which data is sampled. [Table 61](#) shows the timing for the various combinations of LSB First, CPOL, and CPHA.

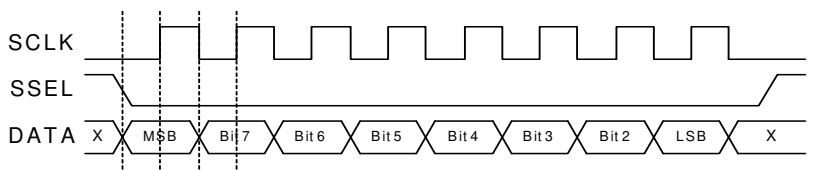
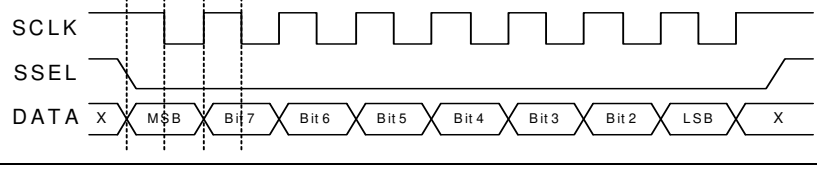
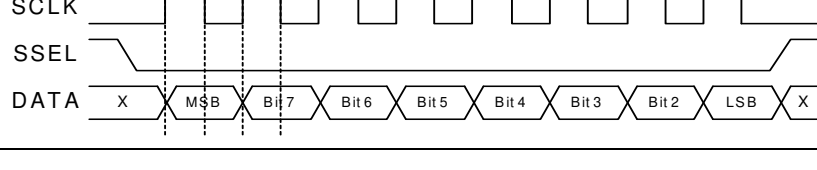
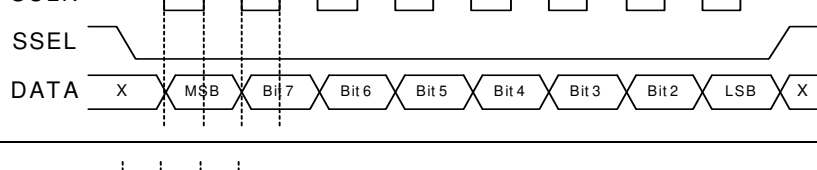
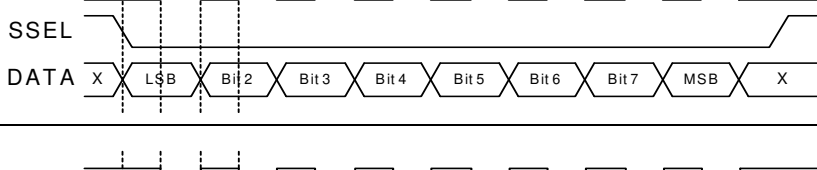
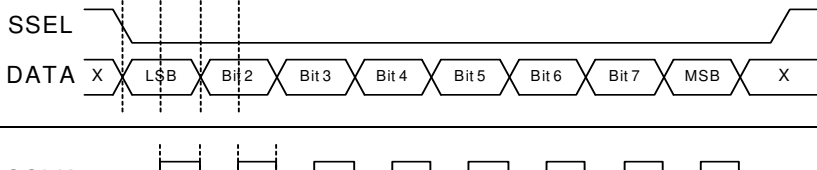
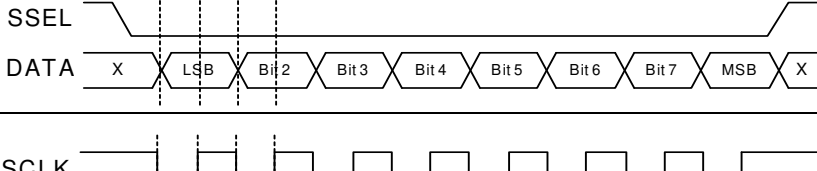
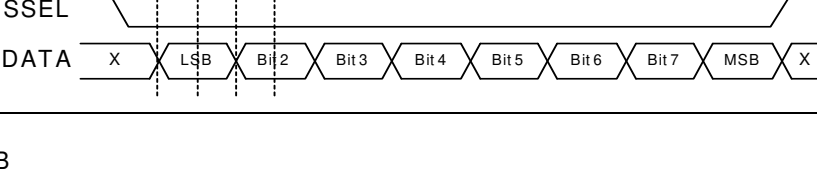
**Bits 1:0** SCLK Select

This field selects the speed of the master SCLK. When in master mode, SCLK is generated by dividing the base CPUCLK.

**Important Note for Comm Modes 01b or 10b (SPI Master or SPI Slave):**

When configured for SPI, (SPI Use = 1—[Table 56 on page 39](#)), the input/output direction of pins P1.3, P1.5, and P1.6 is set automatically by the SPI logic. However, pin P1.4's input/output direction is NOT automatically set; it must be explicitly set by firmware. For SPI Master mode, pin P1.4 must be configured as an output; for SPI Slave mode, pin P1.4 must be configured as an input.

Table 61. SPI Mode Timing vs. LSB First, CPOL and CPHA

| LSB First | CPHA | CPOL | Diagram  |
|-----------|------|------|--|
| 0         | 0    | 0    |    |
| 0         | 0    | 1    |    |
| 0         | 1    | 0    |    |
| 0         | 1    | 1    |   |
| 1         | 0    | 0    |  |
| 1         | 0    | 1    |  |
| 1         | 1    | 0    |  |
| 1         | 1    | 1    |  |

**Table 62. SPI SCLK Frequency**

| SCLK Select | CPUCLK Divisor | SCLK Frequency when CPUCLK = 12 MHz |
|-------------|----------------|-------------------------------------|
| 00          | 6              | 2 MHz                               |
| 01          | 12             | 1 MHz                               |
| 10          | 48             | 250 kHz                             |
| 11          | 96             | 125 kHz                             |

**SPI Interface Pins**

The SPI interface between the radio function and MCU function uses pins P1.3–P1.5 and optionally P1.6. These pins are configured using the P1.3 and P1.4–P1.6 Configuration.

**Timer Registers**

All timer functions of the CYRF69303 are provided by a single timer block. The timer block is asynchronous from the CPU clock.

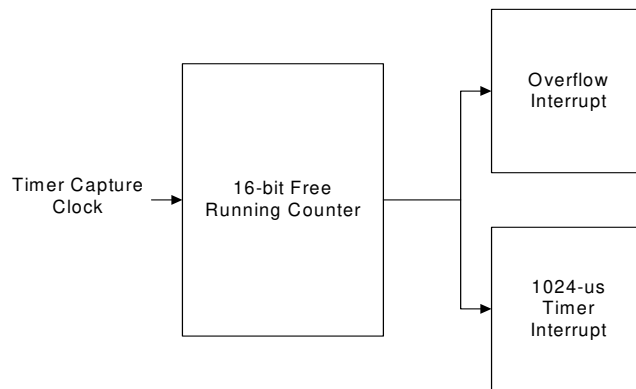
The 16-bit free running counter is used as the time-base for timer captures and can also be used as a general time-base by software.

**Registers**

*Free Running Counter*

The 16-bit free running counter is clocked by a 4 or 6 MHz source. It can be read in software for use as a general purpose time base. When the low order byte is read, the high order byte is registered. Reading the high order byte reads this register allowing the CPU to read the 16-bit value atomically (loads all bits at one time). The free running timer generates an interrupt at 1024  $\mu$ s rate. It can also generate an interrupt when the free running counter overflow occurs—every 16.384 ms. This allows extending the length of the timer in software.

**Figure 12. 16-bit Free Running Counter Block Diagram**



**Table 63. Free Running Timer Low Order Byte (FRTMRL) [0x20] [R/W]**

| Bit #      | 7                        | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------------|--------------------------|-----|-----|-----|-----|-----|-----|-----|
| Field      | Free Running Timer [7:0] |     |     |     |     |     |     |     |
| Read/Write | R/W                      | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Default    | 0                        | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

**Bits 7:0 Free running Timer [7:0]**

This register holds the low order byte of the 16-bit free running timer. Reading this register causes the high order byte to be moved into a holding register allowing an automatic read of all 16 bits simultaneously.

For reads, the actual read occurs in the cycle when the low order is read. For writes the actual time the write occurs is the cycle when the high order is written.

When reading the free running timer, the low order byte must be read first and the high order second. When writing, the low order byte must be written first then the high order byte.



**Table 64. Free Running Timer High-Order Byte (FRTMRH) [0x21] [R/W]**

| Bit #      | 7                         | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------------|---------------------------|-----|-----|-----|-----|-----|-----|-----|
| Field      | Free Running Timer [15:8] |     |     |     |     |     |     |     |
| Read/Write | R/W                       | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Default    | 0                         | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

**Bits 7:0** Free Running Timer [15:8]

When reading the free running timer, the low order byte must be read first and the high order second. When writing, the low order byte must be written first then the high order byte.

**Table 65. Programmable Interval Timer Low (PITMRL) [0x26] [R]**

| Bit #      | 7                         | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|---------------------------|---|---|---|---|---|---|---|
| Field      | Prog Interval Timer [7:0] |   |   |   |   |   |   |   |
| Read/Write | R                         | R | R | R | R | R | R | R |
| Default    | 0                         | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0** Prog Interval Timer [7:0]

This register holds the low order byte of the 12-bit programmable interval timer. Reading this register causes the high order byte to be moved into a holding register allowing an automatic read of all 12 bits simultaneously.

**Table 66. Programmable Interval Timer High (PITMRH) [0x27] [R]**

| Bit #      | 7        | 6  | 5  | 4  | 3                          | 2 | 1 | 0 |
|------------|----------|----|----|----|----------------------------|---|---|---|
| Field      | Reserved |    |    |    | Prog Interval Timer [11:8] |   |   |   |
| Read/Write | --       | -- | -- | -- | R                          | R | R | R |
| Default    | 0        | 0  | 0  | 0  | 0                          | 0 | 0 | 0 |

**Bits 7:4** Reserved

**Bits 3:0** Prog Internal Timer [11:8]

This register holds the high order nibble of the 12-bit programmable interval timer. Reading this register returns the high order nibble of the 12-bit timer at the instant that the low order byte was last read.

**Table 67. Programmable Interval Reload Low (PIRL) [0x28] [R/W]**

| Bit #      | 7                   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------------|---------------------|-----|-----|-----|-----|-----|-----|-----|
| Field      | Prog Interval [7:0] |     |     |     |     |     |     |     |
| Read/Write | R/W                 | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Default    | 0                   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

**Bits 7:0** Prog Interval [7:0]

This register holds the lower 8 bits of the timer. While writing into the 12-bit reload register, write lower byte first then the higher nibble.

**Table 68. Programmable Interval Reload High (PIRH) [0x29] [R/W]**

| Bit #      | 7        | 6  | 5  | 4  | 3                   | 2   | 1   | 0   |
|------------|----------|----|----|----|---------------------|-----|-----|-----|
| Field      | Reserved |    |    |    | Prog Interval[11:8] |     |     |     |
| Read/Write | --       | -- | -- | -- | R/W                 | R/W | R/W | R/W |
| Default    | 0        | 0  | 0  | 0  | 0                   | 0   | 0   | 0   |

**Bits [7:4]** Reserved

**Bits 3:0** Prog Interval [11:8]

This register holds the higher 4 bits of the timer. While writing into the 12-bit reload register, write lower byte first then the higher nibble.

**Figure 13. 16-Bit Free Running Counter Loading Timing Diagram**

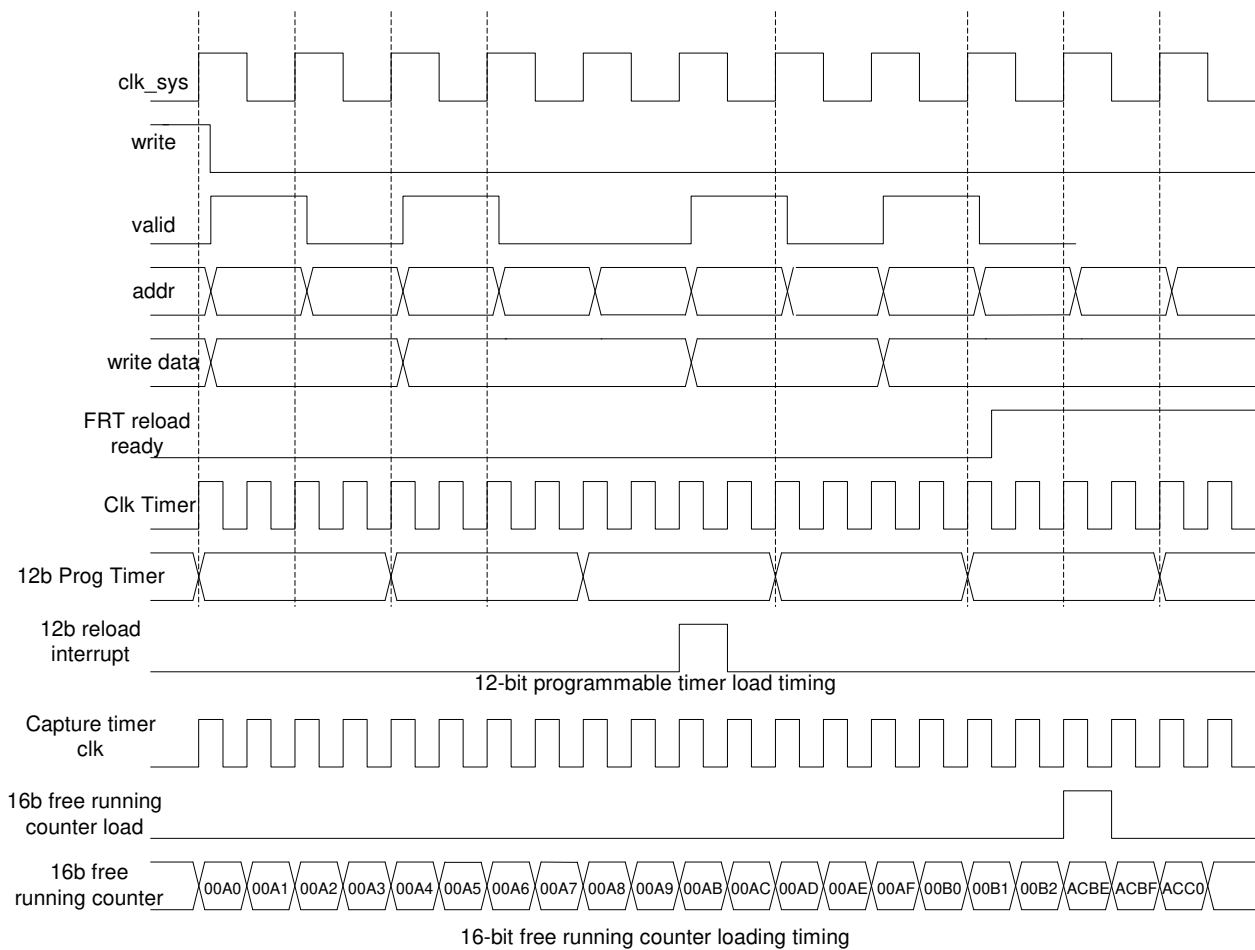
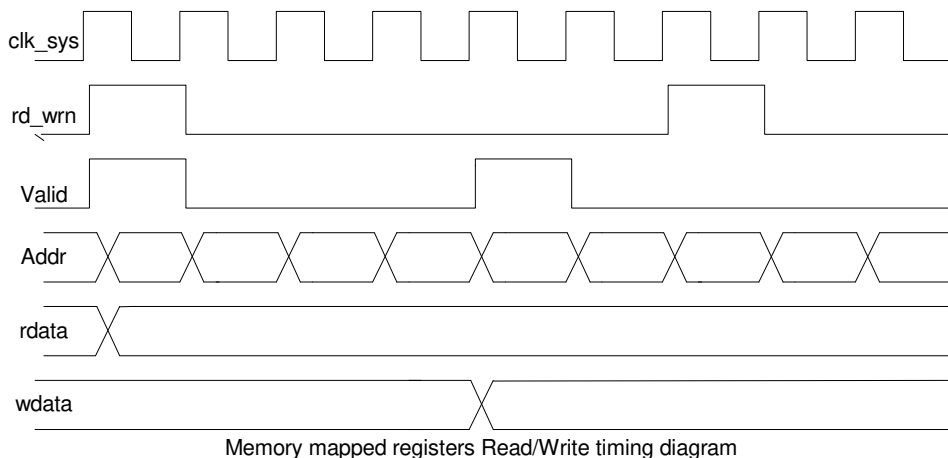


Figure 14. Memory Mapped Registers Read/Write Timing Diagram



## Interrupt Controller

The interrupt controller and its associated registers allow the user’s code to respond to an interrupt from almost every functional block in the CYRF69303 devices. The registers associated with the interrupt controller allow interrupts to be disabled either globally or individually. The registers also provide a mechanism by which a user may clear all pending and posted interrupts, or clear individual posted or pending interrupts.

The following table lists all interrupts and the priorities that are available in the CYRF69303.

Table 69. Interrupt Priorities, Address, Name

| Interrupt Priority | Interrupt Address | Name                        |
|--------------------|-------------------|-----------------------------|
| 0                  | 0000h             | Reset                       |
| 1                  | 0004h             | POR                         |
| 2                  | 0008h             | Reserved                    |
| 3                  | 000Ch             | SPI Transmitter Empty       |
| 4                  | 0010h             | SPI Receiver Full           |
| 5                  | 0014h             | GPIO Port 0                 |
| 6                  | 0018h             | GPIO Port 1                 |
| 7                  | 001Ch             | INT1                        |
| 8                  | 0020h             | Reserved                    |
| 9                  | 0024h             | Reserved                    |
| 10                 | 0028h             | Reserved                    |
| 11                 | 002Ch             | Reserved                    |
| 12                 | 0030h             | Reserved                    |
| 13                 | 0034h             | 1 ms Interval timer         |
| 14                 | 0038h             | Programmable Interval Timer |
| 15                 | 003Ch             | Reserved                    |

Table 69. Interrupt Priorities, Address, Name (continued)

| Interrupt Priority | Interrupt Address | Name                           |
|--------------------|-------------------|--------------------------------|
| 16                 | 0040h             | Reserved                       |
| 17                 | 0044h             | 16-bit Free Running Timer Wrap |
| 18                 | 0048h             | INT2                           |
| 19                 | 004Ch             | Reserved                       |
| 20                 | 0050h             | GPIO Port 2                    |
| 21                 | 0054h             | Reserved                       |
| 22                 | 0058h             | Reserved                       |
| 23                 | 005Ch             | Reserved                       |
| 24                 | 0060h             | Reserved                       |
| 25                 | 0064h             | Sleep Timer                    |

## Architectural Description

An interrupt is posted when its interrupt conditions occur. This results in the flip-flop in Figure 15 on page 48 clocking in a ‘1’. The interrupt remains posted until the interrupt is taken or until it is cleared by writing to the appropriate INT\_CLRx register.

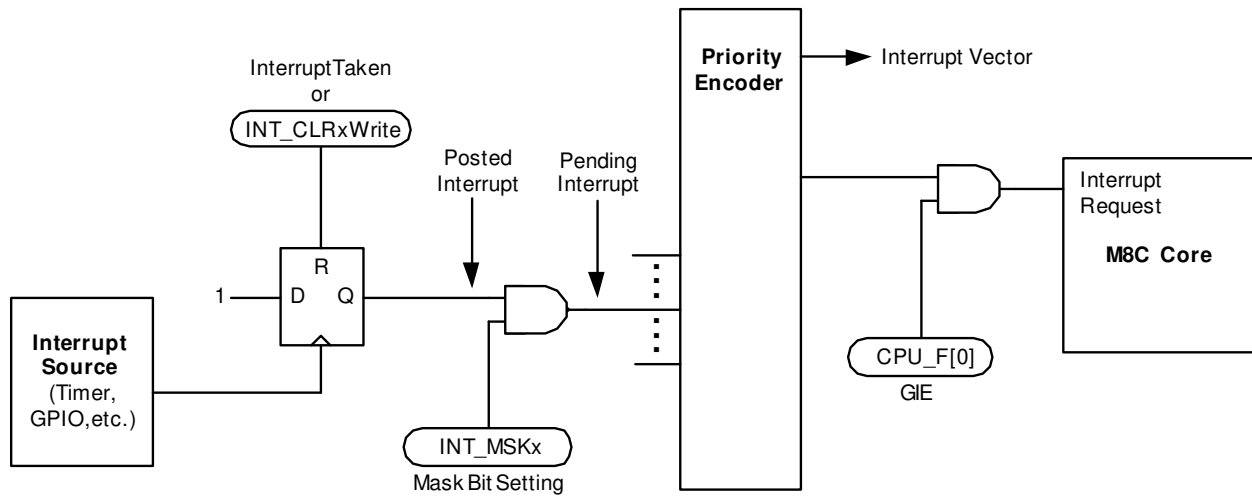
A posted interrupt is not pending unless it is enabled by setting its interrupt mask bit (in the appropriate INT\_MSKx register). All pending interrupts are processed by the Priority Encoder to determine the highest priority interrupt which is taken by the M8C if the Global Interrupt Enable bit is set in the CPU\_F register.

Disabling an interrupt by clearing its interrupt mask bit (in the INT\_MSKx register) does not clear a posted interrupt, nor does it prevent an interrupt from being posted. It simply prevents a posted interrupt from becoming pending.

Nested interrupts can be accomplished by reenabling interrupts inside an interrupt service routine. To do this, set the IE bit in the Flag Register.

A block diagram of the CYRF69303 Interrupt Controller is shown in Figure 15.

Figure 15. Interrupt Controller Block Diagram



### Interrupt Processing

The sequence of events that occur during interrupt processing is as follows:

1. An interrupt becomes active, either because:
  - a. The interrupt condition occurs (for example, a timer expires).
  - b. A previously posted interrupt is enabled through an update of an interrupt mask register.
  - c. An interrupt is pending and GIE is set from 0 to 1 in the CPU Flag register.

1. The current executing instruction finishes.
2. The internal interrupt is dispatched, taking 13 cycles. During this time, the following actions occur:
  - a. The MSB and LSB of Program Counter and Flag registers (CPU\_PC and CPU\_F) are stored onto the program stack by an automatic CALL instruction (13 cycles) generated during the interrupt acknowledge process.
  - b. The PCH, PCL, and Flag register (CPU\_F) are stored onto the program stack (in that order) by an automatic CALL instruction (13 cycles) generated during the interrupt acknowledge process.
  - c. The CPU\_F register is then cleared. Because this clears the GIE bit to 0, additional interrupts are temporarily disabled.
  - d. The PCH (PC[15:8]) is cleared to zero.
  - e. The interrupt vector is read from the interrupt controller and its value placed into PCL (PC[7:0]). This sets the program counter to point to the appropriate address in the interrupt table (for example, 0004h for the POR interrupt).

1. Program execution vectors to the interrupt table. Typically, a LJMP instruction in the interrupt table sends execution to the user's Interrupt Service Routine (ISR) for this interrupt.
2. The ISR executes. Note that interrupts are disabled because GIE = 0. In the ISR, interrupts can be re-enabled if desired by setting GIE = 1 (care must be taken to avoid stack overflow).

3. The ISR ends with a RETI instruction which restores the Program Counter and Flag registers (CPU\_PC and CPU\_F). The restored Flag register re-enables interrupts because GIE = 1 again.
4. Execution resumes at the next instruction, after the one that occurred before the interrupt. However, if there are more pending interrupts, the subsequent interrupts are processed before the next normal program instruction.

### Interrupt Latency

The time between the assertion of an enabled interrupt and the start of its ISR can be calculated from the following equation.

Latency = Time for current instruction to finish + Time for internal interrupt routine to execute + Time for LJMP instruction in interrupt table to execute.

For example, if the 5-cycle JMP instruction is executing when an interrupt becomes active, the total number of CPU clock cycles before the ISR begins is as follows:

$$(1 \text{ to } 5 \text{ cycles for JMP to finish}) + (13 \text{ cycles for interrupt routine}) + (7 \text{ cycles for LJMP}) = 21 \text{ to } 25 \text{ cycles.}$$

In the following example, at 12 MHz, 25 clock cycles take 2.08 μs.

### Interrupt Registers

The Interrupt Registers are discussed in the following sections.

#### Interrupt Clear Register

The Interrupt Clear Registers (INT\_CLRx) are used to enable the individual interrupt sources' ability to clear posted interrupts.

When an INT\_CLRx register is read, any bits that are set indicates an interrupt has been posted for that hardware resource. Therefore, reading these registers gives the user the ability to determine all posted interrupts.

**Table 70. Interrupt Clear 0 (INT\_CLR0) [0xDA] [R/W]**

| Bit #      | 7           | 6           | 5    | 4           | 3           | 2            | 1        | 0   |
|------------|-------------|-------------|------|-------------|-------------|--------------|----------|-----|
| Field      | GPIO Port 1 | Sleep Timer | INT1 | GPIO Port 0 | SPI Receive | SPI Transmit | Reserved | POR |
| Read/Write | R/W         | R/W         | R/W  | R/W         | R/W         | R/W          | R/W      | R/W |
| Default    | 0           | 0           | 0    | 0           | 0           | 0            | 0        | 0   |

When reading this register:

0 = There is no posted interrupt for the corresponding hardware.

1 = Posted interrupt for the corresponding hardware present.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits and to the ENSWINT (Bit 7 of the INT\_MSK3 Register) posts the corresponding hardware interrupt.

The GPIO interrupts are edge-triggered.

**Table 71. Interrupt Clear 1 (INT\_CLR1) [0xDB] [R/W]**

| Bit #      | 7        | 6                   | 5                           | 4        | 3 | 2 | 1 | 0 |
|------------|----------|---------------------|-----------------------------|----------|---|---|---|---|
| Field      | Reserved | Prog Interval Timer | 1 ms Programmable Interrupt | Reserved |   |   |   |   |
| Read/Write | -        | R/W                 | R/W                         | -        | - | - | - | - |
| Default    | 0        | 0                   | 0                           | 0        | 0 | 0 | 0 | 0 |

When reading this register:

0 = There is no posted interrupt for the corresponding hardware.

1 = Posted interrupt for the corresponding hardware present.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT.

Bit 7 Reserved

**Table 72. Interrupt Clear 2 (INT\_CLR2) [0xDC] [R/W]**

| Bit #      | 7        | 6        | 5        | 4          | 3        | 2    | 1                   | 0        |
|------------|----------|----------|----------|------------|----------|------|---------------------|----------|
| Field      | Reserved | Reserved | Reserved | GPIO Port2 | Reserved | INT2 | 16-bit Counter Wrap | Reserved |
| Read/Write | -        | -        | -        | R/W        | -        | R/W  | R/W                 | -        |
| Default    | 0        | 0        | 0        | 0          | 0        | 0    | 0                   | 0        |

When reading this register:

0 = There is no posted interrupt for the corresponding hardware

1 = Posted interrupt for the corresponding hardware present.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT (Bit 7 of the INT\_MSK3 Register) posts the corresponding hardware interrupt.

Bits 7,6,5,3,0]Reserved

*Interrupt Mask Registers*

The Interrupt Mask Registers (INT\_MSKx) are used to enable the individual interrupt sources' ability to create pending interrupts.

There are four Interrupt Mask Registers (INT\_MSK0, INT\_MSK1, INT\_MSK2, and INT\_MSK3) that may be referred to in general as INT\_MSKx. If cleared, each bit in an INT\_MSKx register prevents a posted interrupt from becoming a pending interrupt (input to the priority encoder). However, an interrupt can still post even if its mask bit is zero. All INT\_MSKx bits are independent of all other INT\_MSKx bits.

If an INT\_MSKx bit is set, the interrupt source associated with that mask bit may generate an interrupt that becomes a pending interrupt.

The Enable Software Interrupt (ENSWINT) bit in INT\_MSK3[7] determines the way an individual bit value written to an INT\_CLRx register is interpreted. When is cleared, writing 1's to an INT\_CLRx register has no effect. However, writing 0's to an INT\_CLRx register, when ENSWINT is cleared causes the corresponding interrupt to clear. If the ENSWINT bit is set, any 0's written to the INT\_CLRx registers are ignored. However, 1's written to an INT\_CLRx register, while ENSWINT is set, cause an interrupt to post for the corresponding interrupt.

Software interrupts can aid in debugging interrupt service routines by eliminating the need to create system level interactions that are sometimes necessary to create a hardware-only interrupt.

**Table 73. Interrupt Mask 3 (INT\_MSK3) [0xDE] [R/W]**

| Bit #      | 7       | 6        | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|---------|----------|---|---|---|---|---|---|
| Field      | ENSWINT | Reserved |   |   |   |   |   |   |
| Read/Write | R       | –        | – | – | – | – | – | – |
| Default    | 0       | 0        | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7** Enable Software Interrupt (ENSWINT)

0 = Disable. Writing 0's to an INT\_CLRx register, when ENSWINT is cleared, cause the corresponding interrupt to clear

1 = Enable. Writing 1's to an INT\_CLRx register, when ENSWINT is set, cause the corresponding interrupt to post

**Bits 6:0** Reserved

**Table 74. Interrupt Mask 2 (INT\_MSK2) [0xDF] [R/W]**

| Bit #      | 7        | 6        | 5        | 4                      | 3        | 2               | 1                              | 0        |
|------------|----------|----------|----------|------------------------|----------|-----------------|--------------------------------|----------|
| Field      | Reserved | Reserved | Reserved | GPIO Port 2 Int Enable | Reserved | INT2 Int Enable | 16-bit Counter Wrap Int Enable | Reserved |
| Read/Write | –        | –        | –        | R/W                    | –        | R/W             | R/W                            | –        |
| Default    | 0        | 0        | 0        | 0                      | 0        | 0               | 0                              | 0        |

**Bit 7:** Reserved

**Bit 6:** Reserved

**Bit 5:** Reserved

**Bit 4:** GPIO Port 2 Interrupt Enable

0 = Mask GPIO Port 2 interrupt

1 = Unmask GPIO Port 2 interrupt

**Bit 3:** Reserved

**Bit 2:** INT2 Interrupt Enable

0 = Mask INT2 interrupt

1 = Unmask INT2 interrupt

**Bit 1:** 16-bit Counter Wrap Interrupt Enable

0 = Mask 16-bit Counter Wrap interrupt

1 = Unmask 16-bit Counter Wrap interrupt

**Bit 0:** Reserved

The GPIO interrupts are edge-triggered.

**Table 75. Interrupt Mask 1 (INT\_MSK1) [0xE1] [R/W]**

| Bit #      | 7        | 6                              | 5                     | 4        | 3 | 2 | 1 | 0 |
|------------|----------|--------------------------------|-----------------------|----------|---|---|---|---|
| Field      | Reserved | Prog Interval Timer Int Enable | 1 ms Timer Int Enable | Reserved |   |   |   |   |
| Read/Write | R/W      | R/W                            | R/W                   | –        | – | – | – | – |
| Default    | 0        | 0                              | 0                     | 0        | 0 | 0 | 0 | 0 |

**Bit 7** Reserved

**Bit 6** Prog Interval Timer Interrupt Enable

0 = Mask Prog Interval Timer interrupt

1 = Unmask Prog Interval Timer interrupt

**Bit 5** 1 ms Timer Interrupt Enable

0 = Mask 1 ms interrupt

1 = Unmask 1 ms interrupt

**Bit 4:0** Reserved



**Table 76. Interrupt Mask 0 (INT\_MSK0) [0xE0] [R/W]**

| Bit #      | 7                      | 6                      | 5               | 4                      | 3                      | 2                       | 1        | 0              |
|------------|------------------------|------------------------|-----------------|------------------------|------------------------|-------------------------|----------|----------------|
| Field      | GPIO Port 1 Int Enable | Sleep Timer Int Enable | INT1 Int Enable | GPIO Port 0 Int Enable | SPI Receive Int Enable | SPI Transmit Int Enable | Reserved | POR Int Enable |
| Read/Write | R/W                    | R/W                    | R/W             | R/W                    | R/W                    | R/W                     | R/W      | R/W            |
| Default    | 0                      | 0                      | 0               | 0                      | 0                      | 0                       | 0        | 0              |

- Bit 7** GPIO Port 1 Interrupt Enable  
0 = Mask GPIO Port 1 interrupt  
1 = Unmask GPIO Port 1 interrupt
- Bit 6** Sleep Timer Interrupt Enable  
0 = Mask Sleep Timer interrupt  
1 = Unmask Sleep Timer interrupt
- Bit 5** INT1 Interrupt Enable  
0 = Mask INT1 interrupt  
1 = Unmask INT1 interrupt
- Bit 4** GPIO Port 0 Interrupt Enable  
0 = Mask GPIO Port 0 interrupt  
1 = Unmask GPIO Port 0 interrupt
- Bit 3** SPI Receive Interrupt Enable  
0 = Mask SPI Receive interrupt  
1 = Unmask SPI Receive interrupt
- Bit 2** SPI Transmit Enable  
0 = Mask SPI Transmit interrupt  
1 = Unmask SPI Transmit interrupt
- Bit 1** Reserved
- Bit 0** POR Interrupt Enable  
0 = Mask POR interrupt  
1 = Unmask POR interrupt

*Interrupt Vector Clear Register*

**Table 77. Interrupt Vector Clear Register (INT\_VC) [0xE2] [R/W]**

| Bit #      | 7                       | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------------|-------------------------|-----|-----|-----|-----|-----|-----|-----|
| Field      | Pending Interrupt [7:0] |     |     |     |     |     |     |     |
| Read/Write | R/W                     | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Default    | 0                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

The Interrupt Vector Clear Register (INT\_VC) holds the interrupt vector for the highest priority pending interrupt when read, and when written clears all pending interrupts.

**Bits 7:0** Pending Interrupt [7:0]

8-bit data value holds the interrupt vector for the highest priority pending interrupt. Writing to this register clears all pending interrupts.

**Microcontroller Function Register Summary**

| Addr  | Name        | 7        | 6          | 5           | 4          | 3         | 2          | 1              | 0             | R/W      | Default  |
|-------|-------------|----------|------------|-------------|------------|-----------|------------|----------------|---------------|----------|----------|
| 00    | P0DATA      | P0.7     | Reserved   | Reserved    | P0.4/INT2  | P0.3/INT1 | Reserved   | P0.1           | Reserved      | b--bb-b- | 00000000 |
| 01    | P1DATA      | P1.7     | P1.6/SMISO | P1.5/SMOSI  | P1.4/SCLK  | P1.3/SSEL | P1.2       | P1.1           | P1.0          | bbbbbbb- | 00000000 |
| 02    | P2DATA      | Reserved |            |             |            |           |            | P2.1–P2.0      |               | -----bb  | 00000000 |
| 06    | P01CR       | Reserved | Int Enable | Int Act Low | TTL Thresh | High Sink | Open Drain | Pull-up Enable | Output Enable | bbbbbbbb | 00000000 |
| 08–09 | P03CR–P04CR | Reserved |            | Int Act Low | TTL Thresh | Reserved  | Open Drain | Pull-up Enable | Output Enable | --bb-bbb | 00000000 |
| 0C    | P07CR       | Reserved | Int Enable | Int Act Low | TTL Thresh | Reserved  | Open Drain | Pull-up Enable | Output Enable | -bbb-bbb | 00000000 |

Microcontroller Function Register Summary (continued)

| Addr  | Name        | 7                           | 6                              | 5                      | 4                      | 3                          | 2                       | 1                              | 0              | R/W       | Default   |          |
|-------|-------------|-----------------------------|--------------------------------|------------------------|------------------------|----------------------------|-------------------------|--------------------------------|----------------|-----------|-----------|----------|
| 0D    | P10CR       | Reserved                    | Int Enable                     | Int Act Low            | Reserved               |                            |                         | 5K pullup Enable               | Output enable  | bbb---b   | 00000000  |          |
| 0E    | P11CR       | Reserved                    | Int Enable                     | Int Act Low            | Reserved               |                            | Open Drain              | Reserved                       | Output Enable  | -bb-b-b   | 00000000  |          |
| 0F    | P12CR       | CLK Output                  | Int Enable                     | Int Act Low            | TTL Threshold          | Reserved                   | Open Drain              | Pull-up Enable                 | Output Enable  | bbbb-bbb  | 00000000  |          |
| 10    | P13CR       | Reserved                    | Int Enable                     | Int Act Low            | Reserved               | High Sink                  | Open Drain              | Pull-up Enable                 | Output Enable  | -bb-bbbb  | 00000000  |          |
| 11-13 | P14CR-P16CR | SPI Use                     | Int Enable                     | Int Act Low            | Reserved               | High Sink                  | Open Drain              | Pull-up Enable                 | Output Enable  | bbb-bbbb  | 00000000  |          |
| 14    | P17CR       | Reserved                    | Int Enable                     | Int Act Low            | Reserved               | High Sink                  | Open Drain              | Pull-up Enable                 | Output Enable  | -bb-bbbb  | 00000000  |          |
| 15    | P2CR        | Reserved                    | Int Enable                     | Int Act Low            | TTL Thresh             | High Sink                  | Open Drain              | Pull-up Enable                 | Output Enable  | -bbbbbbb  | 00000000  |          |
| 20    | FRTMRL      | Free Running Timer [7:0]    |                                |                        |                        |                            |                         |                                |                | bbbbbbbbb | 00000000  |          |
| 21    | FRTMRH      | Free Running Timer [15:8]   |                                |                        |                        |                            |                         |                                |                | bbbbbbbbb | 00000000  |          |
| 26    | PITMRL      | Prog Interval Timer [7:0]   |                                |                        |                        |                            |                         |                                |                | rrrrrrrr  | 00000000  |          |
| 27    | PITMRH      | Reserved                    |                                |                        |                        | Prog Interval Timer [11:8] |                         |                                |                | ----rrrr  | 00000000  |          |
| 28    | PIRL        | Prog Interval [7:0]         |                                |                        |                        |                            |                         |                                |                | bbbbbbbbb | 00000000  |          |
| 29    | PIRH        | Reserved                    |                                |                        |                        | Prog Interval [11:8]       |                         |                                |                | ----rrrr  | 00000000  |          |
| 30    | CPUCLKCR    | Reserved                    |                                |                        |                        |                            |                         |                                |                | -----     | 00000000  |          |
| 31    | TMRCLKCR    | TCAPCLK Divider             |                                | TCAPCLK Select         |                        | ITMRCLK Divider            |                         | ITMRCLK Select                 |                | bbbbbbbbb | 10001111  |          |
| 34    | IOSCTR      | foffset[2:0]                |                                |                        | Gain[4:0]              |                            |                         |                                | bbbbbbbbb      |           | 000dddd   |          |
| 36    | LPOSCTR     | 32 kHz Low Power            | Reserved                       | 32 kHz Bias Trim [1:0] |                        | 32 kHz Freq Trim [3:0]     |                         |                                | 0-bbbbbb       |           | d-dddddd  |          |
| 3C    | SPIDATA     | SPIDATA[7:0]                |                                |                        |                        |                            |                         |                                |                | bbbbbbbbb | 00000000  |          |
| 3D    | SPICR       | Swap                        | LSB First                      | Comm Mode              |                        | CPOL                       | CPHA                    | SCLK Select                    |                |           | bbbbbbbbb | 00000000 |
| DA    | INT_CLR0    | GPIO Port 1                 | Sleep Timer                    | INT1                   | GPIO Port 0            | SPI Receive                | SPI Transmit            | Reserved                       | POR            | bbbbbbb-b | 00000000  |          |
| DB    | INT_CLR1    | Reserved                    | Prog Interval Timer            | 1 ms Timer             | Reserved               |                            |                         |                                |                |           | -bb-----  | 00000000 |
| DC    | INT_CLR2    | Reserved                    | Reserved                       | Reserved               | GPIO Port 2            | Reserved                   | INT2                    | 16-bit Counter Wrap            | Reserved       | ---b-bb-  | 00000000  |          |
| DE    | INT_MSK3    | ENSWINT                     | Reserved                       |                        |                        |                            |                         |                                |                |           | r-----    | 00000000 |
| DF    | INT_MSK2    | Reserved                    | Reserved                       | Reserved               | GPIO Port 2 Int Enable | Reserved                   | INT2 Int Enable         | 16-bit Counter Wrap Int Enable | Reserved       | ---b-bb-  | 00000000  |          |
| E0    | INT_MSK0    | GPIO Port 1 Int Enable      | Sleep Timer Int Enable         | INT1 Int Enable        | GPIO Port 0 Int Enable | SPI Receive Int Enable     | SPI Transmit Int Enable | Reserved                       | POR Int Enable | bbbbbbb-b | 00000000  |          |
| E1    | INT_MSK1    | Reserved                    | Prog Interval Timer Int Enable | 1 ms Timer Int Enable  | Reserved               |                            |                         |                                |                |           | -bb-----  | 00000000 |
| E2    | INT_VC      | Pending Interrupt [7:0]     |                                |                        |                        |                            |                         |                                |                | bbbbbbbbb | 00000000  |          |
| E3    | RESWDT      | Reset Watchdog Timer [7:0]  |                                |                        |                        |                            |                         |                                |                | wwwwwwwww | 00000000  |          |
| --    | CPU_A       | Temporary Register T1 [7:0] |                                |                        |                        |                            |                         |                                |                | -----     | 00000000  |          |
| --    | CPU_X       | X[7:0]                      |                                |                        |                        |                            |                         |                                |                | -----     | 00000000  |          |
| --    | CPU_PCL     | Program Counter [7:0]       |                                |                        |                        |                            |                         |                                |                | -----     | 00000000  |          |
| --    | CPU_PCH     | Program Counter [15:8]      |                                |                        |                        |                            |                         |                                |                | -----     | 00000000  |          |
| --    | CPU_SP      | Stack Pointer [7:0]         |                                |                        |                        |                            |                         |                                |                | -----     | 00000000  |          |
| F7    | CPU_F       | Reserved                    |                                |                        | XIO                    | Super                      | Carry                   | Zero                           | Global IE      | ---brbbb  | 00000010  |          |
| FF    | CPU_SCR     | GIES                        | Reserved                       | WDRS                   | PORS                   | Sleep                      | Reserved                | Reserved                       | Stop           | r-ccb-b   | 00010100  |          |
| 1E0   | OSC_CR0     | Reserved                    |                                | No Buzz                | Sleep Timer [1:0]      |                            | CPU Speed [2:0]         |                                |                | --bbbbbb  | 00001000  |          |
| 1E3   | PORCR       | Reserved                    |                                | PORLEV[1:0]            |                        | Reserved                   |                         |                                |                | --bb-bbb  | 00000000  |          |
| 1E4   | VLTCMP      | Reserved                    |                                |                        |                        |                            |                         |                                | PPOR           | -----rr   | 00000000  |          |
| 1EB   | ECO_TR      | Sleep Duty Cycle [1:0]      |                                |                        | Reserved               |                            |                         |                                |                |           | bb-----   | 00000000 |

## Radio Function Register Summary

| Address        | Mnemonic                    | b7  | b6         | b5               | b4         | b3          | b2           | b1           | b0        | Default <sup>(4)</sup> | Access <sup>(4)</sup> |             |          |
|----------------|-----------------------------|---|------------|------------------|------------|-------------|--------------|--------------|-----------|------------------------|-----------------------|-------------|----------|
| 0x00           | CHANNEL_ADR                 | Channel   |            |                  |            |             |              |              |           |                        | -1001000              | -bbbbbbb    |          |
| 0x01           | TX_LENGTH_ADR               | TX Length   |            |                  |            |             |              |              |           |                        | 00000000              | bbbbbbb     |          |
| 0x02           | TX_CTRL_ADR                 | TX GO   | TX CLR     | TXB15 IRQEN      | TXB8 IRQEN | TXB0 IRQEN  | TXBERR IRQEN | TXC IRQEN    | TXE IRQEN | 00000011               | bbbbbbb               |             |          |
| 0x03           | TX_CFG_ADR                  | Not Used  | Not Used   | DATA CODE LENGTH | RSVD       | Data mode   | PA SETTING   |              |           |                        | --000101              | --bbbbbb    |          |
| 0x04           | TX_IRQ_STATUS_ADR           | OS IRQ  | RSVD       | TXB15 IRQ        | TXB8 IRQ   | TXB0 IRQ    | TXBERR IRQ   | TXC IRQ      | TXE IRQ   | -----                  | rrrrrrr               |             |          |
| 0x05           | RX_CTRL_ADR                 | RX GO   | RSVD       | RXB16 IRQEN      | RXB8 IRQEN | RXB1 IRQEN  | RXBERR IRQEN | RXC IRQEN    | RXE IRQEN | 00000111               | bbbbbbb               |             |          |
| 0x06           | RX_CFG_ADR                  | AGC EN  | LNA        | ATT              | HILO       | FASTTURN EN | Not Used     | RXOW EN      | VLD EN    | 10010-10               | bbbb-bb               |             |          |
| 0x07           | RX_IRQ_STATUS_ADR           | RXOW IRQ  | SOPDET IRQ | RXB16 IRQ        | RXB8 IRQ   | RXB1 IRQ    | RXBERR IRQ   | RXC IRQ      | RXE IRQ   | -----                  | brrrrrr               |             |          |
| 0x08           | RX_STATUS_ADR               | RX ACK  | PKT ERR    | EOP ERR          | CRC0       | Bad CRC     | RX Code      | RX Data Mode |           |                        | -----                 | rrrrrrr     |          |
| 0x09           | RX_COUNT_ADR                | RX Count  |            |                  |            |             |              |              |           |                        | 00000000              | rrrrrrr     |          |
| 0x0A           | RX_LENGTH_ADR               | RX Length   |            |                  |            |             |              |              |           |                        | 00000000              | rrrrrrr     |          |
| 0x0B           | PWR_CTRL_ADR                | <b>The firmware should set "00010000" to this register while initiating</b> |            |                  |            |             |              |              |           |                        | 10100000              | bbb-bbbb    |          |
| 0x0C           | XTAL_CTRL_ADR               | XOUT FN   |            |                  | XSIRQ EN   | Not Used    | Not Used     | FREQ         |           |                        |                       | 000-100     | bbb-bbb  |
| 0x0D           | IO_CFG_ADR                  | IRQ OD  | IRQ POL    | MISO OD          | XOUT OD    | RSVD        | HSVD         | SPT 3PIN     | IRQ GPIO  | 00000000               | bbbbbbb               |             |          |
| 0x0E           | GPIO_CTRL_ADR               | XOUT OP   | MISO OP    | RSVD             | IRQ OP     | XOUT IP     | MISO IP      | RSVD         | IRQ IP    | 0000----               | bbbrrrr               |             |          |
| 0x0F           | XACT_CFG_ADR                | ACK EN  | Not Used   | FRC END          | END STATE  |             |              | ACK TO       |           |                        | 1-000000              | b-bbbbbb    |          |
| 0x10           | FRAMING_CFG_ADR             | SOP EN  | SOP LEN    | LEN EN           | SOP TH     |             |              |              |           |                        | 10100101              | bbbbbbb     |          |
| 0x11           | DATA32_THOLD_ADR            | Not Used  | Not Used   | Not Used         | Not Used   | TH32        |              |              |           |                        | ---0100               | ---bbb      |          |
| 0x12           | DATA64_THOLD_ADR            | Not Used  | Not Used   | Not Used         | TH64       |             |              |              |           |                        | ---01010              | ---bbbb     |          |
| 0x13           | RSSI_ADR                    | SOP   | Not Used   | LNA              | RSSI       |             |              |              |           | 0-100000               | r-rrrrr               |             |          |
| 0x14           | EOP_CTRL_ADR <sup>(9)</sup> | HEN   | HINT       |                  |            | EOP         |              |              |           |                        | 10100100              | bbbbbbb     |          |
| 0x15           | CRC_SEED_LSB_ADR            | CRC SEED LSB  |            |                  |            |             |              |              |           |                        | 00000000              | bbbbbbb     |          |
| 0x16           | CRC_SEED_MSB_ADR            | CRC SEED MSB  |            |                  |            |             |              |              |           |                        | 00000000              | bbbbbbb     |          |
| 0x17           | TX_CRC_LSB_ADR              | CRC LSB   |            |                  |            |             |              |              |           |                        | -----                 | rrrrrrr     |          |
| 0x18           | TX_CRC_MSB_ADR              | CRC MSB   |            |                  |            |             |              |              |           |                        | -----                 | rrrrrrr     |          |
| 0x19           | RX_CRC_LSB_ADR              | CRC LSB   |            |                  |            |             |              |              |           |                        | 11111111              | rrrrrrr     |          |
| 0x1A           | RX_CRC_MSB_ADR              | CRC MSB   |            |                  |            |             |              |              |           |                        | 11111111              | rrrrrrr     |          |
| 0x1B           | TX_OFFSET_LSB_ADR           | STRIM LSB   |            |                  |            |             |              |              |           |                        | 00000000              | bbbbbbb     |          |
| 0x1C           | TX_OFFSET_MSB_ADR           | Not Used  | Not Used   | Not Used         | Not Used   | STRIM MSB   |              |              |           |                        |                       | ----0000    | ----bbbb |
| 0x1D           | MODE_OVERRIDE_ADR           | RSVD  | RSVD       | FRC SEN          | FRC AWAKE  |             | Not Used     | Not Used     | RST       | 00000-0                | wwwwww-w              |             |          |
| 0x1E           | RX_OVERRIDE_ADR             | ACK RX  | RXTX DLY   | MAN RXACK        | FRC RXDR   | DIS CRC0    | DIS RXCRC    | ACE          | Not Used  | 00000000               | bbbbbbb               |             |          |
| 0x1F           | TX_OVERRIDE_ADR             | ACK TX  | FRC PRE    | RSVD             | MAN TXACK  | OVRD ACK    | DIS TXCRC    | RSVD         | TX INV    | 00000000               | bbbbbbb               |             |          |
| 0x26           | XTAL_CFG_ADR                | RSVD  | RSVD       | RSVD             | RSVD       | START DLY   | RSVD         | RSVD         | RSVD      | 00000000               | wwwwww<br>w           |             |          |
| 0x27           | CLK_OVERRIDE_ADR            | RSVD  | RSVD       | RSVD             | RSVD       | RSVD        | RSVD         | RXF          | RSVD      | 00000000               | wwwwww<br>w           |             |          |
| 0x28           | CLK_EN_ADR                  | RSVD  | RSVD       | RSVD             | RSVD       | RSVD        | RSVD         | RXF          | RSVD      | 00000000               | wwwwww<br>w           |             |          |
| 0x29           | RX_ABORT_ADR                | RSVD  | RSVD       | ABORT EN         | RSVD       | RSVD        | RSVD         | RSVD         | RSVD      | 00000000               | wwwwww<br>w           |             |          |
| 0x32           | AUTO_CAL_TIME_ADR           | AUTO_CAL_TIME   |            |                  |            |             |              |              |           |                        | 00000011              | wwwwww<br>w |          |
| 0x35           | AUTO_CAL_OFFSET_ADR         | AUTO_CAL_OFFSET   |            |                  |            |             |              |              |           |                        | 00000000              | wwwwww<br>w |          |
| 0x39           | ANALOG_CTRL_ADR             | RSVD  | RSVD       | RSVD             | RSVD       | RSVD        | RSVD         | RX INV       | ALL SLOW  | 00000000               | wwwwww<br>w           |             |          |
| Register Files |                             |   |            |                  |            |             |              |              |           |                        |                       |             |          |
| 0x20           | TX_BUFFER_ADR               | TX Buffer File  |            |                  |            |             |              |              |           |                        | -----                 | wwwwww<br>w |          |
| 0x21           | RX_BUFFER_ADR               | RX Buffer File  |            |                  |            |             |              |              |           |                        | -----                 | rrrrrrr     |          |
| 0x22           | SOP_CODE_ADR                | SOP Code File   |            |                  |            |             |              |              |           |                        | Note [5]              | bbbbbbb     |          |
| 0x23           | DATA_CODE_ADR               | Data Code File  |            |                  |            |             |              |              |           |                        | Note [6]              | bbbbbbb     |          |
| 0x24           | PREAMBLE_ADR                | Preamble File   |            |                  |            |             |              |              |           |                        | Note [7]              | bbbbbbb     |          |
| 0x25           | MFG_ID_ADR                  | MFG ID File   |            |                  |            |             |              |              |           |                        | NA                    | rrrrrrr     |          |

All registers are read and writable, except where noted. Registers may be written to or read from either individually or in sequential groups. A single-byte read or write reads or writes from the addressed register. Incrementing burst read and write is a sequence that begins with an address, and then reads or writes to/from each register in address order for as long as clocking continues. It is possible to repeatedly read (poll) a single register using a nonincrementing burst read.

**Notes**

- b = read/write; r = read only; w = write only; '-' = not used, default value is undefined.
- SOP\_CODE\_ADR default = 0x17FF9E213690C782.
- DATA\_CODE\_ADR default = 0x02F9939702FA5CE3012BF1DB0132BE6F.
- PREAMBLE\_ADR default = 0x333302
- Registers must be configured or accessed only when the radio is in IDLE or SLEEP mode. The GPIOs, RSSI registers can be accessed in Active Tx and Rx mode.
- EOP\_CTRL\_ADR[6:4] must never have the value of "000" i.e. EOP Hint Symbol count must never be "0"

### Absolute Maximum Ratings

Exceeding maximum ratings may shorten the useful life of the device. User guidelines are not tested.

Storage temperature ..... -40 °C to +90 °C  
 Ambient temperature with power applied ..... 0 °C to +70 °C  
 Supply voltage on any power supply pin relative to V<sub>SS</sub> ..... -0.3 V to +3.9 V  
 DC voltage to logic inputs<sup>[10]</sup> ..... -0.3 V to V<sub>IO</sub> +0.3 V  
 DC voltage applied to outputs in High-Z state ..... -0.3 V to V<sub>IO</sub> +0.3 V

Static discharge voltage (Digital)<sup>[11]</sup> ..... >2000 V  
 Static discharge voltage (RF)<sup>[11]</sup> ..... 1100 V  
 Latch up current ..... +200 mA, -200 mA  
 Ground voltage ..... 0 V  
 F<sub>OSC</sub> (Crystal frequency) ..... 12 MHz ±30 ppm

### DC Characteristics (T = 25 °C)

| Parameter  | Description                                      | Conditions                                     | Min | Typ  | Max | Unit |
|--|--|--|-----|------|-----|------|
| V <sub>BAT</sub>   | Battery voltage                                  | 0-70 °C  | 2.7 | -    | 3.6 | V    |
| V <sub>IO</sub>  | V <sub>IO</sub> voltage                          |  | 2.7 | -    | 3.6 | V    |
| V <sub>CC</sub>  | V <sub>CC</sub> voltage                          | 0-70 °C  | 2.7 | -    | 3.6 | V    |
| <b>Device Current</b> (For total current consumption in different modes, for example Radio, active, MCU, and sleep, add Radio Function Current and MCU Function Current) |  |  |     |      |     |      |
| I <sub>CC</sub> (GFSK) <sup>[12]</sup>   | Average I <sub>CC</sub> , 1 Mbps, slow channel   | PA = 5, 2-way, 4 bytes/10 ms CPU speed = 6 MHz | -   | 9.87 | -   | mA   |
| I <sub>CC</sub> (32-8DR) <sup>[12]</sup>   | Average I <sub>CC</sub> , 250 kbps, fast channel | PA = 5, 2-way, 4 bytes/10 ms CPU speed = 6 MHz | -   | 10.2 | -   | mA   |
| I <sub>SB</sub>  | Sleep Mode I <sub>CC</sub>                       | V <sub>CC</sub> = 3.0 V, MCU sleep             | -   | 2.72 | -   | µA   |

**Notes**

- 10. It is permissible to connect voltages above V<sub>IO</sub> to inputs through a series resistor limiting input current to 1 mA. AC timing not guaranteed.
- 11. Human Body Model (HBM).
- 12. Includes current drawn while starting crystal, starting synthesizer, transmitting packet (including SOP and CRC16), changing to receive mode, and receiving ACK handshake. Device is in sleep except during this transaction.

**DC Characteristics** (T = 25 °C) (continued)

| Parameter   | Description                                    | Conditions  | Min                   | Typ             | Max                  | Unit            |
|---|--|---|-----------------------|-----------------|----------------------|-----------------|
| <b>Radio Function Currents</b> (V <sub>CC</sub> = 3.0 V, MCU Sleep) |  |   |                       |                 |                      |                 |
| IDLE I <sub>CC</sub>  | Radio off, XTAL Active                         | XOUT disabled   | –                     | 1.1             | –                    | mA              |
| I <sub>synth</sub>  | I <sub>CC</sub> during Synth Start             |   | –                     | 8.6             | –                    | mA              |
| TX I <sub>CC</sub>  | I <sub>CC</sub> during transmit                | PA = 5 (–5 dBm)   | –                     | 21.2            | –                    | mA              |
| TX I <sub>CC</sub>  | I <sub>CC</sub> during transmit                | PA = 6 (0 dBm)  | –                     | 28.5            | –                    | mA              |
| RX I <sub>CC</sub>  | I <sub>CC</sub> during receive                 | LNA off, ATT on.  | –                     | 18.9            | –                    | mA              |
| RX I <sub>CC</sub>  | I <sub>CC</sub> during receive                 | LNA on, ATT off.  | –                     | 21.9            | –                    | mA              |
| <b>MCU Function Currents</b> (V <sub>DD</sub> = 3.0 V)              |  |   |                       |                 |                      |                 |
| I <sub>DD1</sub>  | V <sub>DD</sub> operating supply current       | CPU speed = 6 MHz   | –                     | 5.0             | –                    | mA              |
| I <sub>DD1</sub>  | V <sub>DD</sub> operating supply current       | CPU speed = 3 MHz   | –                     | 4.4             | –                    | mA              |
| <b>Radio Function GPIO Interface</b>                                |  |   |                       |                 |                      |                 |
| V <sub>OH1</sub>  | Output High voltage condition 1                | At I <sub>OH</sub> = –100.0 μA                                      | V <sub>IO</sub> – 0.1 | V <sub>IO</sub> | –                    | V               |
| V <sub>OH2</sub>  | Output High voltage condition 2                | At I <sub>OH</sub> = –2.0 mA  | V <sub>IO</sub> – 0.4 | V <sub>IO</sub> | –                    | V               |
| V <sub>OL</sub>   | Output Low voltage                             | At I <sub>OL</sub> = 2.0 mA   | –                     | 0               | 0.4                  | V               |
| V <sub>IH</sub>   | Input High voltage                             |   | 0.76 V <sub>IO</sub>  | –               | V <sub>IO</sub>      | V               |
| V <sub>IL</sub>   | Input Low voltage                              |   | 0                     | –               | 0.24 V <sub>IO</sub> | V               |
| I <sub>IL</sub>   | Input leakage current                          | 0 < V <sub>IN</sub> < V <sub>IO</sub>                               | –1                    | 0.26            | +1                   | μA              |
| C <sub>IN</sub>   | Pin input capacitance                          | except XTAL, RF <sub>N</sub> , RF <sub>P</sub> , RF <sub>BIAS</sub> | –                     | 3.5             | 10                   | pF              |
| <b>MCU Function GPIO Interface</b>                                  |  |   |                       |                 |                      |                 |
| R <sub>UP</sub>   | Pull-up resistance                             |   | 4                     | –               | 12                   | KΩ              |
| V <sub>ICR</sub>  | Input threshold voltage low, CMOS mode         | Low to High edge  | 40%                   | –               | 65%                  | V <sub>CC</sub> |
| V <sub>ICF</sub>  | Input threshold voltage low, CMOS mode         | High to Low edge  | 30%                   | –               | 55%                  | V <sub>CC</sub> |
| V <sub>HC</sub>   | Input hysteresis voltage, CMOS Mode            | High to low edge  | 3%                    | –               | 10%                  | V <sub>CC</sub> |
| V <sub>ILTTL</sub>  | Input Low voltage, TTL Mode                    |   | –                     | –               | 0.72                 | V               |
| V <sub>IHTTL</sub>  | Input HIGH voltage, TTL Mode                   |   | 1.6                   | –               |                      | V               |
| V <sub>OL1</sub>  | Output Low voltage, High Drive <sup>[13]</sup> | I <sub>OL1</sub> = 50 mA  | –                     | –               | 1.4                  | V               |
| V <sub>OL2</sub>  | Output Low voltage, High Drive <sup>[13]</sup> | I <sub>OL1</sub> = 25 mA  | –                     | –               | 0.4                  | V               |
| V <sub>OL3</sub>  | Output Low voltage, Low Drive                  | I <sub>OL2</sub> = 8 mA   | –                     | –               | 0.8                  | V               |
| V <sub>OH</sub>   | Output High voltage <sup>[14]</sup>            | I <sub>OH</sub> = 2 mA  | V <sub>CC</sub> – 0.5 | –               |                      | V               |

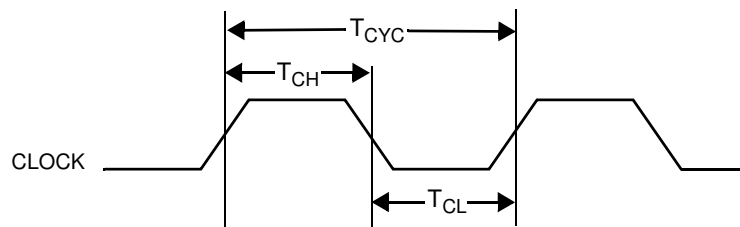
**Notes**

13. Available only on P1.3,P1.4,P1.5,P1.6,P1.7.  
 14. Except for pins P1.0, P1.1 in GPIO mode.

### AC Characteristics

| Parameter           | Description                                      | Conditions  | Min     | Typ | Max  | Unit |
|---------------------|--|---|---------|-----|------|------|
| <b>GPIO Timing</b>  |  |   |         |     |      |      |
| T <sub>R_GPIO</sub> | Output rise time                                 | Measured between 10 and 90% V <sub>dd</sub> with 50 pF load | –       | –   | 50   | ns   |
| T <sub>F_GPIO</sub> | Output fall time                                 | Measured between 10 and 90% V <sub>dd</sub> with 50 pF load | –       | –   | 15   | ns   |
| F <sub>IMO</sub>    | Internal main oscillator frequency               | With proper trim values loaded                              | 18.72   | –   | 26.4 | MHz  |
| F <sub>ILO</sub>    | Internal low power oscillator                    | With proper trim values loaded                              | 15.0001 | –   | 50.0 | kHz  |
| <b>SPI Timing</b>   |  |   |         |     |      |      |
| T <sub>SMCK</sub>   | SPI master clock rate                            | F <sub>CPUCLK</sub> /6                                      | –       | –   | 2    | MHz  |
| T <sub>SSCK</sub>   | SPI slave clock rate                             |   | –       | –   | 2.2  | MHz  |
| T <sub>SCKH</sub>   | SPI clock high time                              | High for CPOL = 0, Low for CPOL = 1                         | 125     | –   | –    | ns   |
| T <sub>SCKL</sub>   | SPI clock low time                               | Low for CPOL = 0, High for CPOL = 1                         | 125     | –   | –    | ns   |
| T <sub>MDO</sub>    | Master data output time <sup>[15]</sup>          | SCK to data valid   | –25     | –   | 50   | ns   |
| T <sub>MDO1</sub>   | Master data output time, First bit with CPHA = 0 | Time before leading SCK edge                                | 100     | –   | –    | ns   |
| T <sub>MSU</sub>    | Master input data setup time                     |   | 50      | –   | –    | ns   |
| T <sub>MHD</sub>    | Master input data hold time                      |   | 50      | –   | –    | ns   |
| T <sub>SSU</sub>    | Slave input data setup time                      |   | 50      | –   | –    | ns   |
| T <sub>SHD</sub>    | Slave input data hold time                       |   | 50      | –   | –    | ns   |
| T <sub>SDO</sub>    | Slave data output time                           | SCK to data valid   | –       | –   | 100  | ns   |
| T <sub>SDO1</sub>   | Slave data output time, First bit with CPHA = 0  | Time after SS LOW to data valid                             | –       | –   | 100  | ns   |
| T <sub>SSS</sub>    | Slave select setup time                          | Before first SCK edge                                       | 150     | –   | –    | ns   |
| T <sub>SSH</sub>    | Slave select hold time                           | After last SCK edge   | 150     | –   | –    | ns   |

Figure 16. Clock Timing



**Note**

15. In Master mode first bit is available 0.5 SPICLK cycle before Master clock edge available on the SCLK pin.

Figure 17. GPIO Timing Diagram

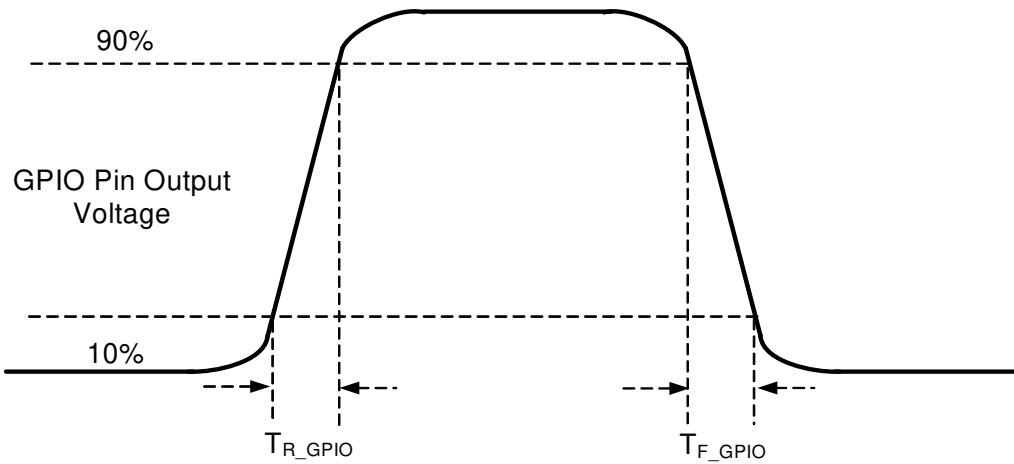


Figure 18. SPI Master Timing, CPHA = 1

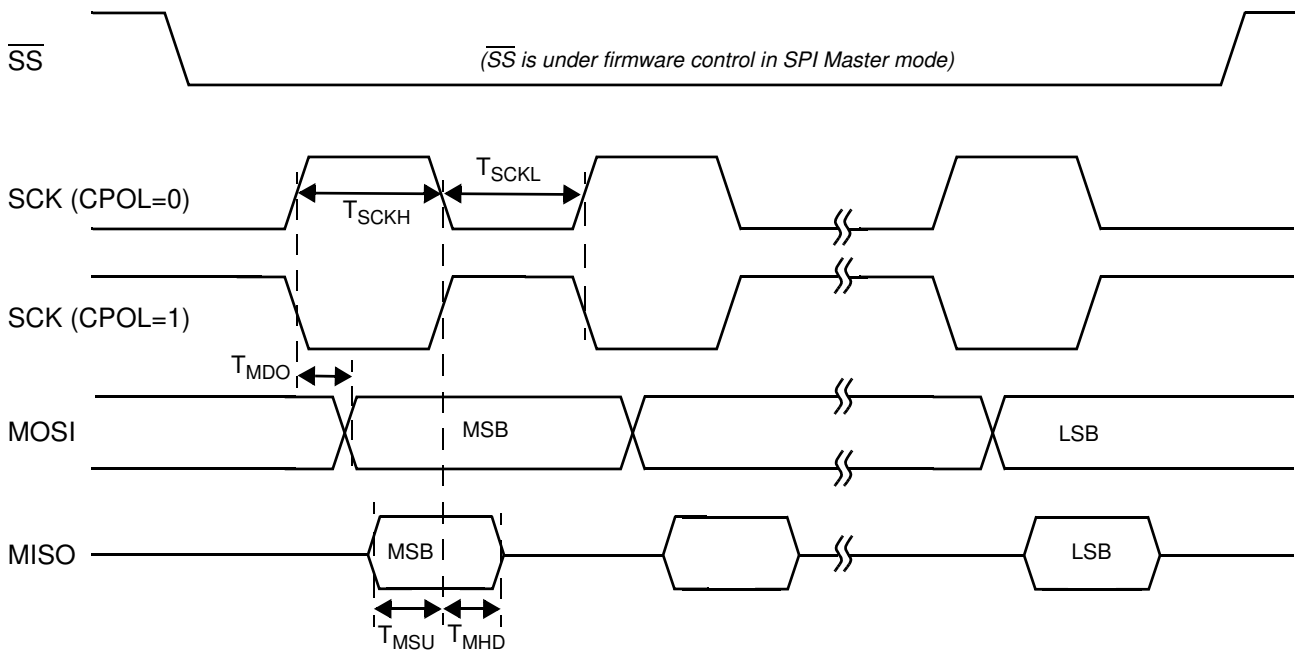




Figure 19. SPI Slave Timing, CPHA = 1

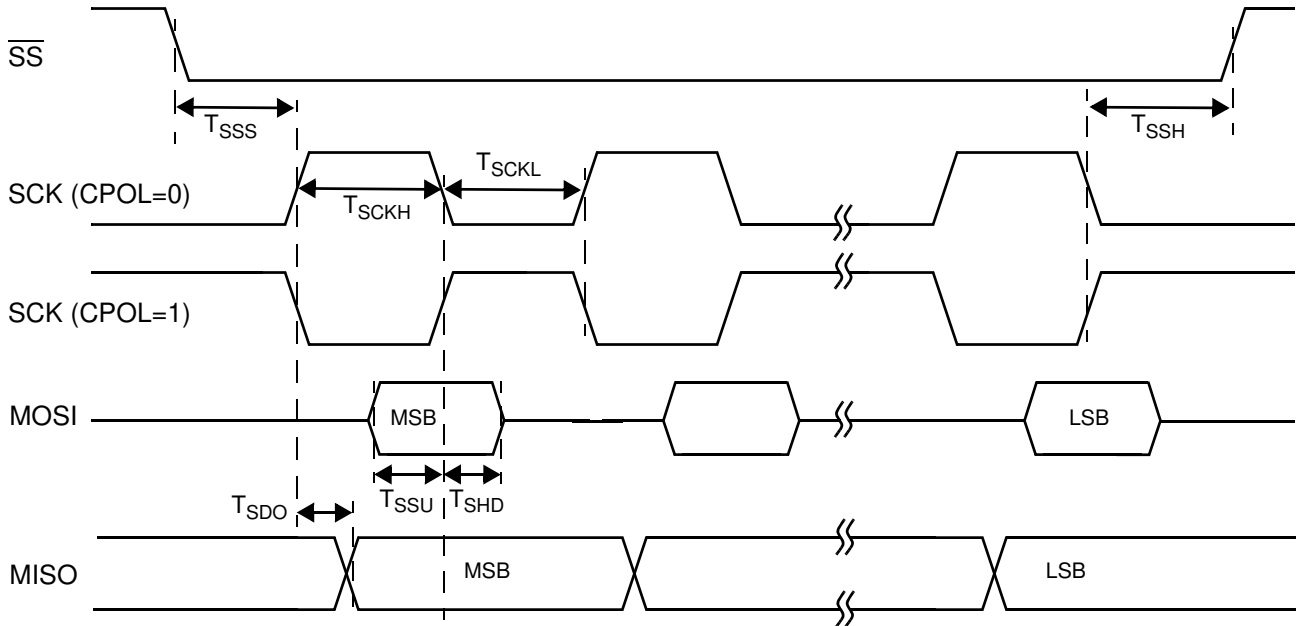


Figure 20. SPI Master Timing, CPHA = 0

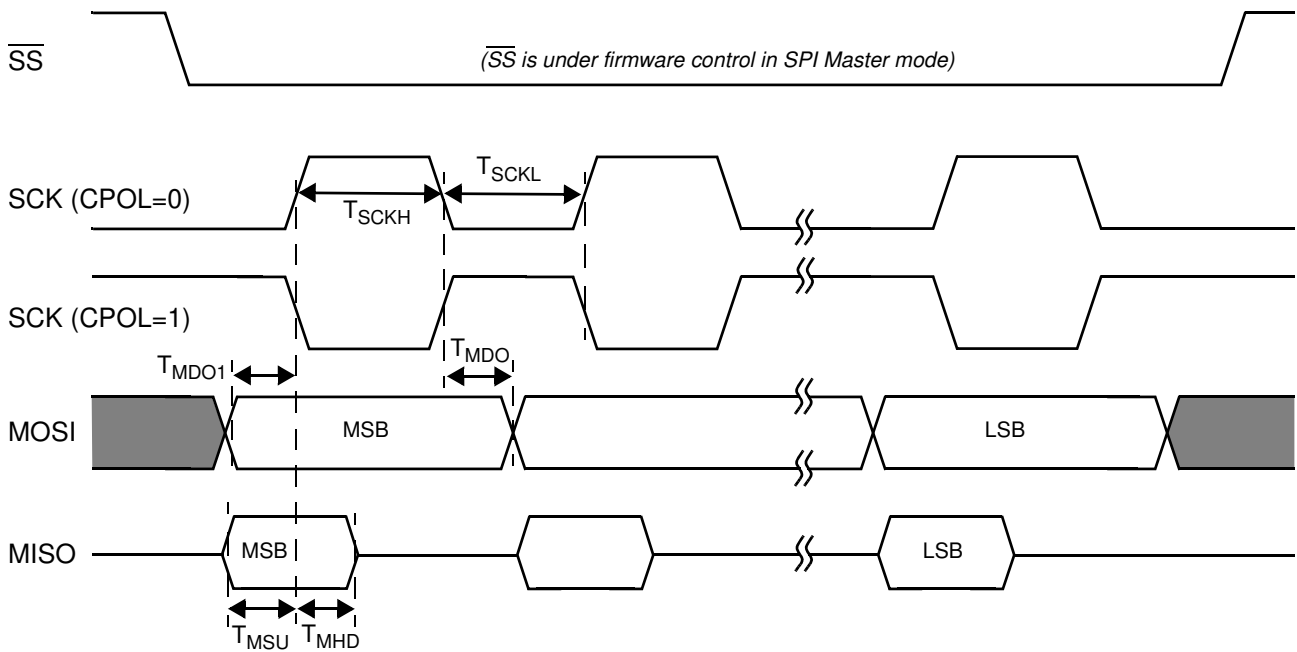
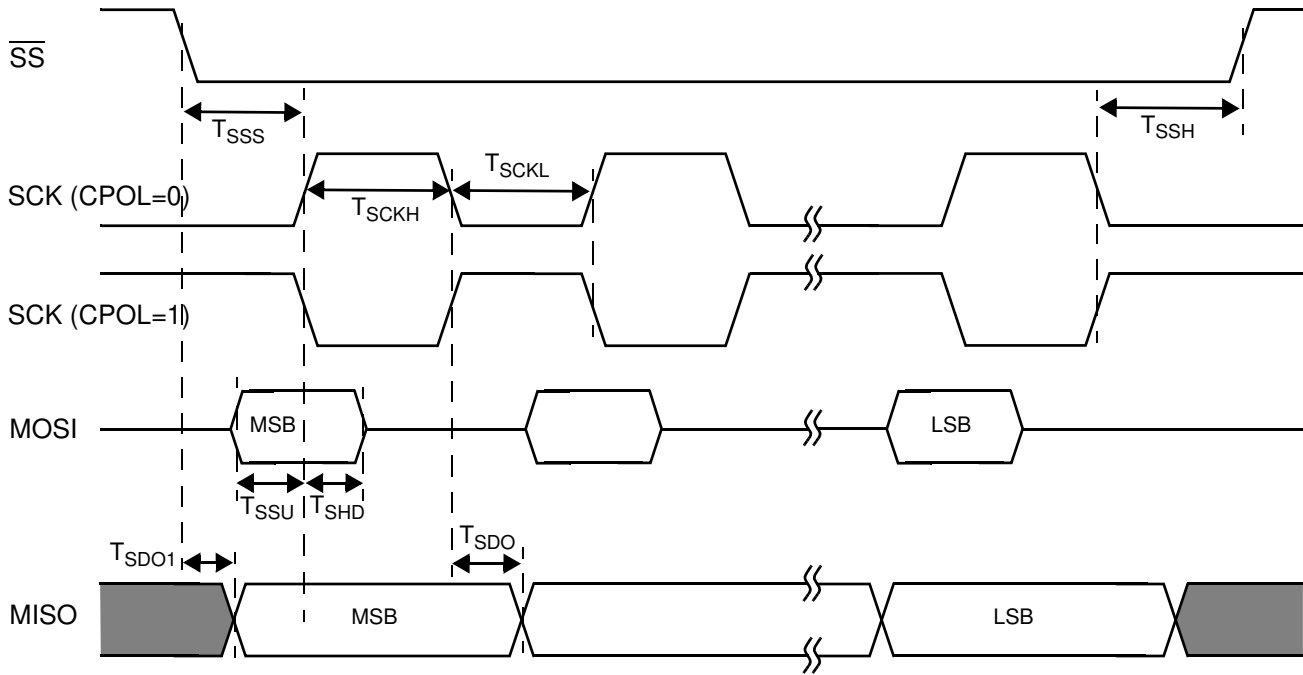


Figure 21. SPI Slave Timing, CPHA = 0



**RF Characteristics**
**Table 78. Radio Parameters**

| Parameter Description  | Conditions                 | Min   | Typ   | Max   | Unit     |
|--|----------------------------|-------|-------|-------|----------|
| RF frequency range   | Subject to regulation      | 2.400 | –     | 2.497 | GHz      |
| <b>Receiver</b> (T = 25 °C, V <sub>CC</sub> = 3.0 V, f <sub>OSC</sub> = 12.000 MHz, BER < 10 <sup>-3</sup> ) |                            |       |       |       |          |
| Sensitivity 250 kbps 32-8DR  | BER 1E-3                   | –     | –90   | –     | dBm      |
| Sensitivity GFSK   | BER 1E-3, ALL SLOW = 1     | –     | –84   | –     | dBm      |
| LNA Gain   |                            | –     | 22.8  | –     | dB       |
| ATT Gain   |                            | –     | –31.7 | –     | dB       |
| Maximum Received Signal  | LNA On                     | –15   | –6    | –     | dBm      |
| RSSI Value for PWR <sub>in</sub> –60 dBm   | LNA On                     | –     | 21    | –     | Count    |
| RSSI Slope   |                            | –     | 1.9   | –     | dB/Count |
| <b>Interference Performance</b> (CER 1E-3)   |                            |       |       |       |          |
| Co-channel Interference rejection<br>Carrier-to-Interference (C/I)   | C = –60 dBm                | –     | 9     | –     | dB       |
| Adjacent (±1 MHz) Channel Selectivity C/I 1 MHz  | C = –60 dBm                | –     | 3     | –     | dB       |
| Adjacent (±2 MHz) Channel Selectivity C/I 2 MHz  | C = –60 dBm                | –     | –30   | –     | dB       |
| Adjacent (≥ 3 MHz) Channel Selectivity C/I ≥ 3 MHz   | C = –67 dBm                | –     | –38   | –     | dB       |
| Out-of-Band Blocking 30 MHz–12.75 MHz <sup>[16]</sup>  | C = –67 dBm                | –     | –30   | –     | dBm      |
| Intermodulation  | C = –64 dBm, Δf = 5,10 MHz | –     | –36   | –     | dBm      |
| <b>Receive Spurious Emission</b>   |                            |       |       |       |          |
| 800 MHz  | 100 kHz ResBW              | –     | –79   | –     | dBm      |
| 1.6 GHz  | 100 kHz ResBW              | –     | –71   | –     | dBm      |
| 3.2 GHz  | 100 kHz ResBW              | –     | –65   | –     | dBm      |
| <b>Transmitter</b> (T = 25°C, V <sub>CC</sub> = 3.0 V, f <sub>OSC</sub> = 12.000 MHz)                        |                            |       |       |       |          |
| Maximum RF transmit power  | PA = 6                     | –2    | 0     | +2    | dBm      |
| Maximum RF transmit power  | PA = 5                     | –7    | –5    | –3    | dBm      |
| Maximum RF transmit power  | PA = 0                     | –     | –35   | –     | dBm      |
| RF power control range   |                            | –     | 35    | –     | dB       |
| RF power range control step size   | Six steps, monotonic       | –     | 5.6   | –     | dB       |
| Frequency deviation Min  | PN Code Pattern 10101010   | –     | 270   | –     | kHz      |
| Frequency deviation Max  | PN Code Pattern 11110000   | –     | 323   | –     | kHz      |
| Error vector magnitude (FSK error)   | >0 dBm                     | –     | 10    | –     | %rms     |
| Occupied bandwidth   | –6 dBc, 100 kHz ResBW      | 500   | 876   | –     | kHz      |

**Notes**

16. Exceptions F/3 and 5C/3.

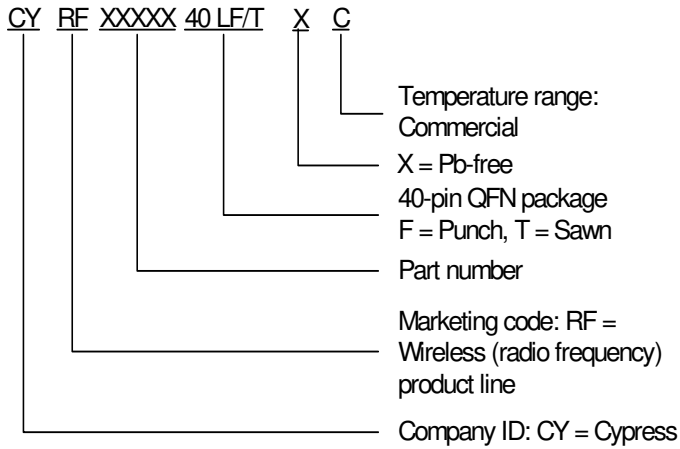
Table 78. Radio Parameters (continued)

| Parameter Description                                  | Conditions                  | Min | Typ | Max | Unit    |
|--|-----------------------------|-----|-----|-----|---------|
| <b>Transmit Spurious Emission</b> (PA = 6)             |                             |     |     |     |         |
| In-band Spurious Second Channel Power ( $\pm 2$ MHz)   |                             | –   | –38 | –   | dBm     |
| In-band Spurious Third Channel Power ( $\geq 3$ MHz)   |                             | –   | –44 | –   | dBm     |
| Non-Harmonically Related Spurs (8.000 GHz)             |                             | –   | –38 | –   | dBm     |
| Non-Harmonically Related Spurs (1.6 GHz)               |                             | –   | –34 | –   | dBm     |
| Non-Harmonically Related Spurs (3.2 GHz)               |                             | –   | –47 | –   | dBm     |
| Harmonic Spurs (Second Harmonic)                       |                             | –   | –43 | –   | dBm     |
| Harmonic Spurs (Third Harmonic)                        |                             | –   | –48 | –   | dBm     |
| Fourth and Greater Harmonics                           |                             | –   | –59 | –   | dBm     |
| <b>Power Management</b> (Crystal PN# eCERA GF-1200008) |                             |     |     |     |         |
| Crystal Start to 10ppm                                 |                             | –   | 0.7 | 1.3 | ms      |
| Crystal Start to IRQ                                   | XSIRQ EN = 1                | –   | 0.6 | –   | ms      |
| Synth Settle   | Slow channels               | –   | –   | 270 | $\mu$ s |
| Synth Settle   | Medium channels             | –   | –   | 180 | $\mu$ s |
| Synth Settle   | Fast channels               | –   | –   | 100 | $\mu$ s |
| Link Turnaround Time                                   | GFSK                        | –   | –   | 30  | $\mu$ s |
| Link Turnaround Time                                   | 250 kbps                    | –   | –   | 62  | $\mu$ s |
| Max. packet length                                     | < 60 ppm crystal-to-crystal | –   | –   | 40  | bytes   |

**Ordering Information**

| Package                           | Ordering Part Number |
|-----------------------------------|----------------------|
| 40-pin Pb-free Punch-QFN 6 × 6 mm | CYRF69303-40LFXC     |
| 40-pin Pb-free Sawn-QFN 6 × 6 mm  | CYRF69303-40LTXC     |

**Ordering Code Definitions**



### Package Handling

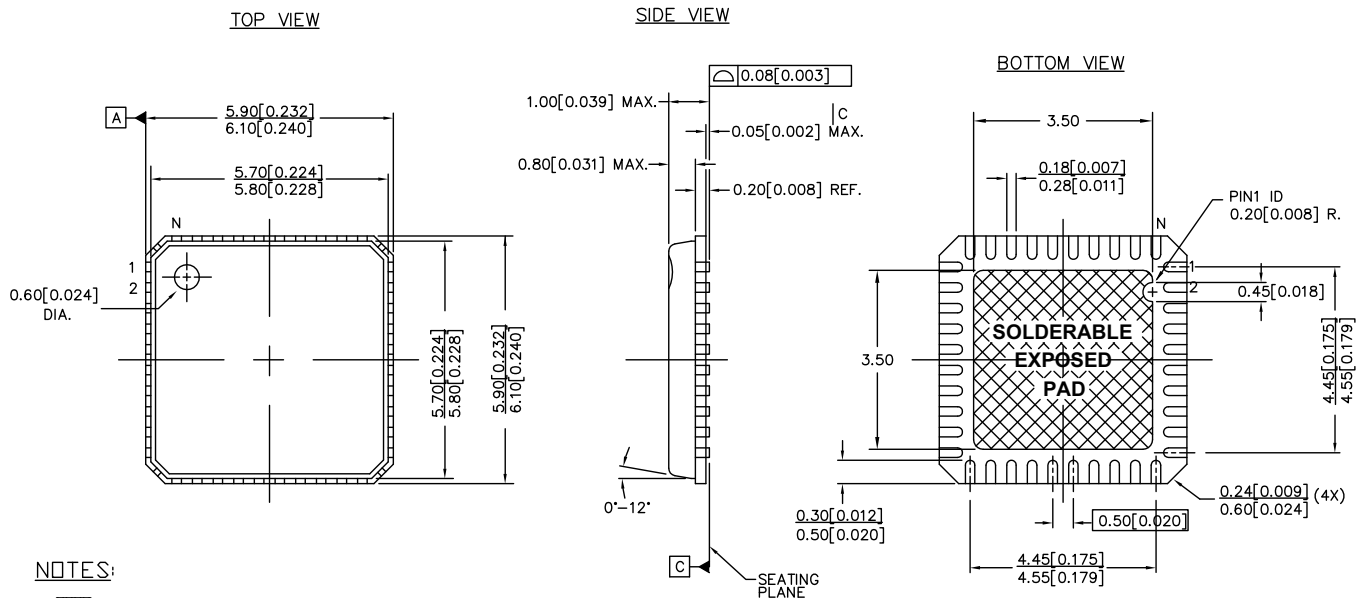
Some IC packages require baking before they are soldered onto a PCB to remove moisture that may have been absorbed after leaving the factory. A label on the packaging has details about actual bake temperature and the minimum bake time to remove this moisture. The maximum bake time is the aggregate time that the parts are exposed to the bake temperature. Exceeding this exposure time may degrade device reliability.

**Table 79. Package Handling**

| Parameter             | Description      | Min               | Typ | Max               | Unit  |
|-----------------------|------------------|-------------------|-----|-------------------|-------|
| T <sub>BAKETEMP</sub> | Bake Temperature |                   | 125 | see package label | °C    |
| t <sub>BAKETIME</sub> | Bake Time        | see package label |     | 24                | hours |

### Package Diagram

**Figure 22. 40-Pin Pb-free Punch-QFN 6 × 6 mm**



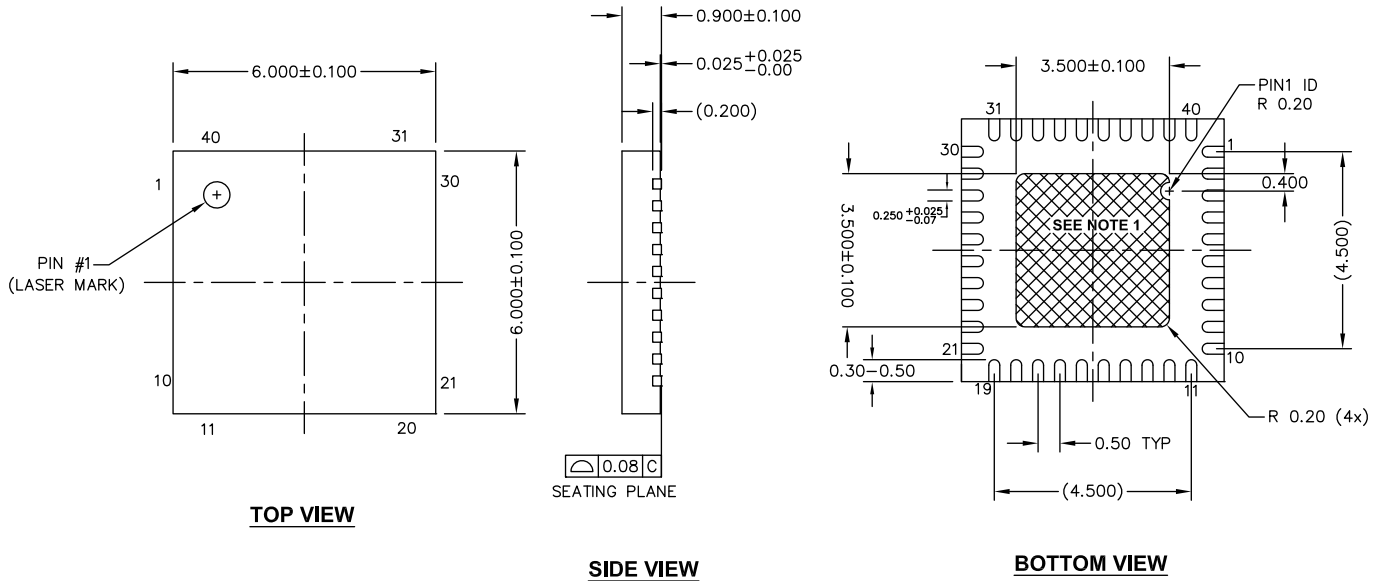
**NOTES:**

1. [Hatched Area] HATCH IS SOLDERABLE EXPOSED AREA
2. REFERENCE JEDEC#: MO-220
3. PACKAGE WEIGHT: 0.086g
4. ALL DIMENSIONS ARE IN MM [MIN/MAX]
5. PACKAGE CODE


| PART # | DESCRIPTION |
|--------|-------------|
| LF40A  | STANDARD    |
| LY40A  | PB-FREE     |

001-12917 \*C

Figure 23. 40-Pin Pb-free Sawn-QFN 6 x 6 mm



NOTES:

1.  HATCH IS SOLDERABLE EXPOSED AREA.
2. REFERENCE JEDEC #: MO-220
3. PACKAGE WEIGHT: 0.086g
4. ALL DIMENSIONS ARE IN MILLIMETERS

001-44328 \*F



## Acronyms

**Table 80. Acronyms Used in this Document**

| Acronym | Description                              |
|---------|--|
| ACK     | Acknowledge (packet received, no errors) |
| BER     | Bit error rate                           |
| BOM     | Bill of materials                        |
| CMOS    | complementary metal oxide semiconductor  |
| CRC     | cyclic redundancy check                  |
| FEC     | forward error correction                 |
| FER     | frame error rate                         |
| GFSK    | Gaussian frequency-shift keying          |
| HBM     | Human body model                         |
| ISM     | Industrial, scientific, and medical      |
| IRQ     | interrupt request                        |
| MCU     | microcontroller unit                     |
| NRZ     | non return to zero                       |
| PLL     | phase-locked loop                        |
| QFN     | Quad flat no-leads                       |
| RSSI    | received signal strength indication      |
| RF      | radio frequency                          |
| Rx      | receive                                  |
| Tx      | transmit                                 |

## Document Conventions

### Units of Measure

**Table 81. Units of Measure**

| Symbol | Unit of Measure             |
|--------|-----------------------------|
| °C     | degree Celsius              |
| dB     | decibels                    |
| dBc    | decibel relative to carrier |
| dBm    | decibel-milliwatt           |
| Hz     | hertz                       |
| KB     | 1024 bytes                  |
| Kbit   | 1024 bits                   |
| kHz    | kilohertz                   |
| kΩ     | kilohm                      |
| MHz    | megahertz                   |
| MΩ     | megaohm                     |
| μA     | microampere                 |
| μs     | microsecond                 |
| μV     | microvolt                   |
| μVrms  | microvolts root-mean-square |
| μW     | microwatt                   |
| mA     | milliampere                 |
| ms     | millisecond                 |
| mV     | millivolt                   |
| nA     | nanoampere                  |
| ns     | nanosecond                  |
| nV     | nanovolt                    |
| Ω      | ohm                         |
| pp     | peak-to-peak                |
| ppm    | parts per million           |
| ps     | picosecond                  |
| sps    | samples per second          |
| V      | volt                        |

## Document History Page

| Document Title: CYRF69303 Programmable Radio-on-Chip LPstar<br>Document #: 001-66502 |         |                 |                 |   |
|--|---------|-----------------|-----------------|---|
| Revision   | ECN     | Orig. of Change | Submission Date | Description of Change   |
| **   | 3188093 | NXZ/KKCN        | 04/05/11        | New advance datasheet.  |
| *A   | 3333406 | KPMD            | 08/01/2011      | Removed Advance status from the datasheet. Post to external web.  |
| *B   | 3532316 | KKCN            | 02/28/2012      | Updated <a href="#">Ordering Information</a> (CYRF69303-40LTXC) and <a href="#">Ordering Code Definitions</a> .<br>Added <a href="#">Package Diagram</a> (001-44328). |

## Sales, Solutions, and Legal Information

### Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer’s representatives, and distributors. To find the office closest to you, visit us at [Cypress Locations](#).

### Products

|                            |  |
|----------------------------|--|
| Automotive                 | <a href="http://cypress.com/go/automotive">cypress.com/go/automotive</a>   |
| Clocks and Buffers         | <a href="http://cypress.com/go/clocks">cypress.com/go/clocks</a>   |
| Interface                  | <a href="http://cypress.com/go/interface">cypress.com/go/interface</a>   |
| Lighting and Power Control | <a href="http://cypress.com/go/powerpsoc">cypress.com/go/powerpsoc</a><br><a href="http://cypress.com/go/plc">cypress.com/go/plc</a> |
| Memory                     | <a href="http://cypress.com/go/memory">cypress.com/go/memory</a>   |
| Optical and Image Sensing  | <a href="http://cypress.com/go/image">cypress.com/go/image</a>   |
| PSoC                       | <a href="http://cypress.com/go/psoc">cypress.com/go/psoc</a>   |
| Touch Sensing              | <a href="http://cypress.com/go/touch">cypress.com/go/touch</a>   |
| USB Controllers            | <a href="http://cypress.com/go/USB">cypress.com/go/USB</a>   |
| Wireless/RF                | <a href="http://cypress.com/go/wireless">cypress.com/go/wireless</a>   |

### PSoC Solutions

[psoc.cypress.com/solutions](http://psoc.cypress.com/solutions)  
PSoC 1 | PSoC 3 | PSoC 5

© Cypress Semiconductor Corporation, 2011-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress’ product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.