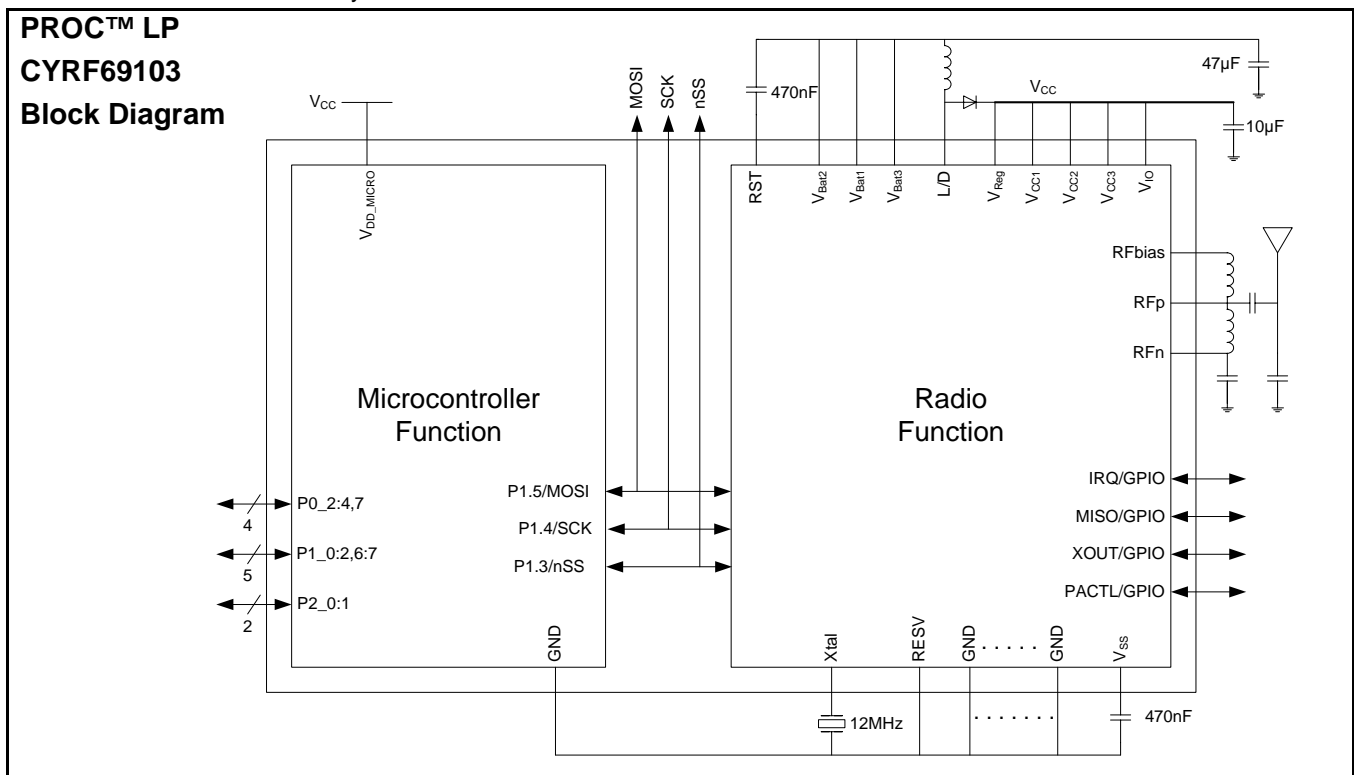


# Programmable Radio on Chip Low Power

## PRoC™ LP Features

- Single Device, Two Functions
  - 8-bit, Flash based MCU function and 2.4 GHz radio transceiver function in a single device.
- Flash-Based Microcontroller Function
  - 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
  - CPU speed up to 12 MHz
  - 16-bit free-running timer
  - Low power wakeup timer
  - 12-bit Programmable Interval Timer with interrupts
  - Watchdog timer
- Industry-Leading 2.4 GHz Radio Transceiver Function
  - Operates in the unlicensed worldwide Industrial, Scientific, and Medical (ISM) band (2.4 GHz–2.483 GHz)
  - DSSS data rates of up to 250 Kbps
  - GFSK data rate of 1 Mbps
  - -97 dBm receive sensitivity
- Programmable output power up to +4 dBm
- Auto Transaction Sequencer (ATS)
- Framing CRC and Auto ACK
- Received Signal Strength Indication (RSSI)
- Automatic Gain Control (AGC)
- Component Reduction
  - Integrated 1.8V boost converter
  - GPIOs that require no external components
  - Operates off a single crystal
- Flexible I/O
  - High current drive on 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
- Operating voltage from 1.8V to 3.6V DC
- Operating temperature from 0 to 70°C
- Lead-free 40-lead QFN package
- Advanced development tools based on Cypress's PSoC® tools



## Applications

The CYRF69103 PRoC LP is targeted for the following applications:

- Wireless HID devices
  - Mice
  - Remote Controls
  - Presenter tools
  - Barcode scanners
  - POS terminal
- General purpose wireless applications:
  - Industrial applications
  - Home automation
  - White goods
  - Consumer electronics
  - Toys

## Functional Description

PRoC LP 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 via the radio's SPI interface.

## Functional Overview

The CYRF69103 is a complete Radio System-on-Chip device, providing a complete RF system solution with a single device and a few discrete components. The CYRF69103 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–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). In DSSS modes the baseband performs DSSS spreading/despreading, while in GFSK Mode (1 Mb/s - GFSK) the baseband performs Start of Frame (SOF), End of Frame (EOF) 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, except SDR, enabling the implementation of mixed-rate systems in which different devices use different data rates. This also enables the implementation of dynamic data rate systems, which use high data rates at shorter distances and/or in a low-moderate interference environment, and change to lower data rates at longer distances and/or in high interference environments.

The radio meets the following world-wide regulatory requirements:

- Europe
  - ETSI EN 301 489-1 V1.4.1
  - ETSI EN 300 328-1 V1.3.1
- North America
  - FCC CFR 47 Part 15
- Japan
  - ARIB STD-T66

### Data Transmission Modes

The radio supports four different data transmission modes:

- In GFSK mode, data is transmitted at 1 Mbps, without any DSSS
- In 8DR mode, 1 byte is encoded in each PN code symbol transmitted
- In DDR mode, 2 bits are encoded in each PN code symbol transmitted
- In SDR mode, a single bit is encoded in each PN code symbol transmitted

Both 64-chip and 32-chip data PN codes are supported. The four data transmission modes apply to the data after the Start of Packet (SOP). In particular, the packet length, data and CRC are all sent in the same mode.

### Microcontroller Function

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

The MCU function has up to eight 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 three dedicated pins that have independent interrupt vectors (P0.2 - P0.4).

The microcontroller features an internal oscillator.

The PRoC LP includes a Watchdog timer, a vectored interrupt controller, a 12-bit programmable interval timer with configurable 1-ms interrupt and a 16-bit free-running timer with capture registers.

In addition, the CYRF69103 IC has a Power Management Unit (PMU), which allows direct connection of the device to any battery voltage in the range 1.8V to 3.6V. The PMU conditions the battery voltage to provide the supply voltages required by the device, and may supply external devices.

### Backward Compatibility

The CYRF69103 IC is fully interoperable with the main modes of other Cypress radios CYWUSB6934 and CYRF69103. The 62.5-kbps mode is supported by selecting 32-chip DATA\_CODE\_ADR codes, DDR mode, and disabling the SOP, length, and CRC16 fields. Similarly, the 15.675-kHz

mode is supported by selecting 64-chip DATA\_CODE\_ADR codes and SDR mode.

In this way, a suitably configured CYRF69103 IC device may transmit data to and/or receive data from a first generation device.

### Pinout

Pin	Name	Function/Description
1	P0.4	Individually configured GPIO
2	XTAL	12 MHz crystal
3, 7, 16	V <sub>CC</sub>	2.4V to 3.6V supply. Connected to pin 40 (0.047- $\mu$ F bypass)
4	P0.3	Individually configured GPIO
5	P0.1	Individually configured GPIO
6	V <sub>bat1</sub>	Connect to 1.8V to 3.6V power supply, through 47-ohm series/1- $\mu$ F shunt C
8	P2.1	GPIO. Port 2 Bit 1
9	V <sub>bat2</sub>	Connected to 1.8V to 3.6V main power supply, through 0.047- $\mu$ F bypass C
10	RF <sub>bias</sub>	RF pin voltage reference
11	RF <sub>p</sub>	Differential RF to/from antenna
12	GND	GND
13	RF <sub>n</sub>	Differential RF to/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 pin 40, 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, PACTL <sub>n</sub> or Radio GPIO
31	PACTL	Control for external PA or Radio GPIO
32	P1.6	GPIO
33	V <sub>IO</sub>	1.8V to 3.6V to main power supply rail for Radio IO
34	RST	Radio Reset. Connected to pin 40 with 0.47 $\mu$ F. 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 $\mu$ F to GND
37	L/D	Inductor/Diode connection for Boost. When Internal PMU is not being used connect L/D to GND.
38	P0.7	GPIO
39	V <sub>bat0</sub>	Connected to 1.8V to 3.6V main power supply, through 0.047- $\mu$ F bypass C
40	V <sub>REG</sub>	Boost regulator output voltage feedback

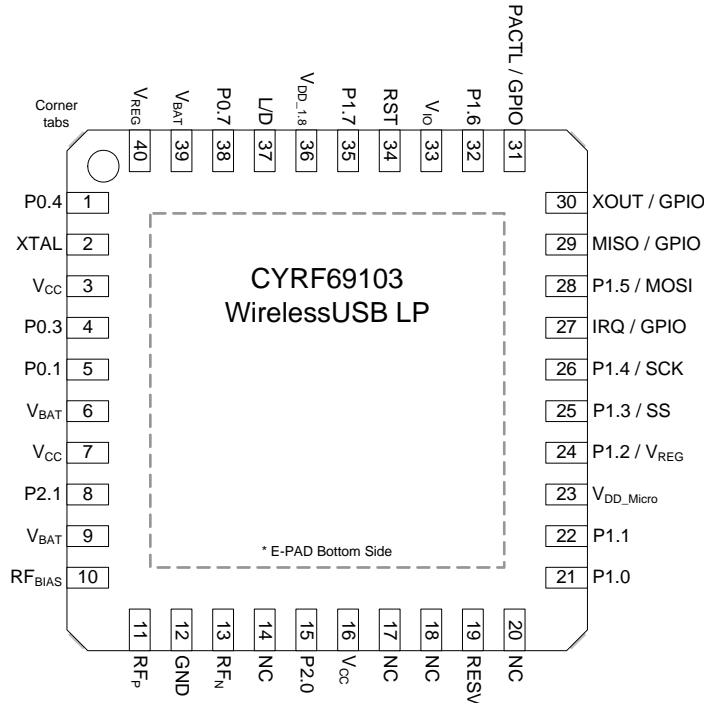
**Pinout**

Pin	Name	Function/Description
41	E-pad	GND

**Pinout Diagram**

Figure 1. Pinout

40-pin Lead-Free QFN 6x6 mm LY40



**Functional Block Overview**

All the blocks that make up the PRoC LP are presented here.

**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 +4 dBm transmit power, with an output power control range of 34 dB in 7 steps. The supply current of the device is reduced as the RF output power is reduced.

Table 1. Internal PA Output Power Step Table

PA Setting	Typical Output Power (dBm)
7	+4
6	0
5	-5
4	-10
3	-15

Table 1. Internal PA Output Power Step Table

PA Setting	Typical Output Power (dBm)
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 3<sup>rd</sup> frequency, starting at 2400 MHz up to and including 2472 MHz (i.e., 0,3,6,9.....69 & 72).

**Baseband and Frammer**

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

**Data Transmission Modes and Data Rates**

The SoC supports four different data transmission modes:

- In GFSK mode, data is transmitted at 1 Mbps, without any DSSS.
- In 8DR mode, 8 bits are encoded in each DATA\_CODE\_ADR derived code symbol transmitted.
- In DDR mode, 2-bits are encoded in each DATA\_CODE\_ADR derived code symbol transmitted. (As in the CYWUSB6934 DDR mode).
- In SDR mode, 1 bit is encoded in each DATA\_CODE\_ADR derived code symbol transmitted. (As in the CYWUSB6934 standard modes.)

Both 64-chip and 32-chip DATA\_CODE\_ADR codes are supported. The four 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, lower data rates reduces packet error rate in any given environment.

The CYRF69103 IC supports the following data rates:

- 1000-kbps (GFSK)
- 250-kbps (32-chip 8DR)
- 125-kbps (64-chip 8DR)
- 62.5-kbps (32-chip DDR)
- 31.25-kbps (64-chip DDR)
- 15.625-kbps (64-chip SDR)

Lower data rates typically provide longer range and/or a more robust link.

**Link Layer Modes**

The CYRF69103 IC device supports the following data packet framing features:

**SOP** – Packets begin with a 2-symbol Start of Packet (SOP) marker. This is required in GFSK and 8DR modes, but is optional in DDR mode and is not supported in SDR mode; 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.

**EOP** – There are two options for detecting the end of a packet. If SOP is enabled, then a packet length field may be enabled. GFSK and 8DR must enable the length field. This is the first 8 bits after the SOP symbol, and is transmitted at the payload data rate. If the length field is enabled, an End of Packet (EOP) condition is inferred after reception of the number of bytes defined in the length field, plus two bytes for the CRC16 (if enabled—see below). The alternative to using the length field is to infer an EOP condition from a configurable number of successive non-correlations; this option is not available in GFSK mode and is only recommended when using SDR mode.

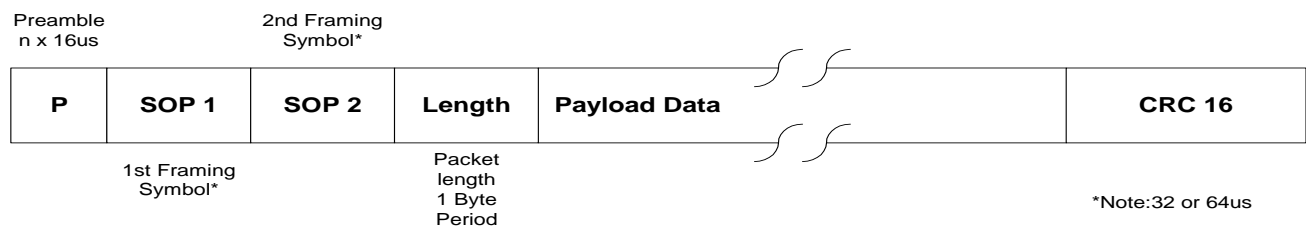
**CRC16** – The device may be configured to append a 16-bit CRC16 to each packet. The CRC16 uses the USB CRC polynomial with the added programmability of the seed. If enabled, the receiver will verify the calculated CRC16 for the payload data against the received value in the CRC16 field. The starting 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 will be 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**



**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 (as in the CYWUSB6934). 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 uses 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 CYRF69103 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 CYRF69103 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 a given 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 capac-

itors. 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.

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

- 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: 10  $\mu$ W–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/Low-Voltage Detect

The power-on reset 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. There is a low-voltage detect circuit that detects when  $V_{CC}$  drops below a programmable trip voltage. It may be configurable to generate an LVD interrupt to inform the processor about the low-voltage event. POR and LVD share the same interrupt. There is not a separate interrupt for each. The Watchdog timer can be used to ensure the firmware never gets stalled in an infinite loop.

### 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 1.8V to 3.6V DC, which is applied to the  $V_{BAT}$  pin. The device can be shut down to a fully static sleep mode by writing the FRC END = 1 and END STATE = 000 bits in the XACT\_CFG\_ADR register over the

SPI interface. The device will enter 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 packet transmission or reception. When in sleep mode, the on-chip oscillator is stopped, but the SPI interface remains functional. The device will wake 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 may be configured to assert the IRQ pin when the oscillator has stabilized.

The output voltage ( $V_{REG}$ ) of the Power Management Unit (PMU) is configurable to several minimum values between 2.4V and 2.7V.  $V_{REG}$  may be used to provide up to 15 mA (average load) to external devices. It is possible to disable the PMU, and to provide an externally regulated DC supply voltage to the device in the range 2.4V to 3.6V. The PMU also provides a regulated 1.8V supply to the logic.

The PMU has been designed to provide high boost efficiency (74–85% depending on input voltage, output voltage and load) when using a Schottky diode and power inductor, eliminating the need for an external boost converter in many systems where other components require a boosted voltage. However, reasonable efficiencies (69–82% depending on input voltage, output voltage and load) may be achieved when using low-cost components such as SOT23 diodes and 0805 inductors.

The PMU also provides a configurable low battery detection function which may be read over the SPI interface. One of seven thresholds between 1.8V and 2.7V may be selected. The interrupt pin may be configured to assert when the voltage on the  $V_{BAT}$  pin falls below the configured threshold. LV IRQ is not a latched event. Battery monitoring is disabled when the device is in sleep mode.

The following three figures show different examples of how to use PRoC LP with and without the PMU. Figure 3. shows the most common circuit making use of the PMU to boost battery voltage up to 2.7v. Figure 4. is an example of the circuit used when the supply voltage will always be above 2.7V. This could be three 1.5v battery cells in series along with a linear regulator, or some similar power source. Figure 5. shows an example of using the PRoC LP with its PMU disabled and an external boost to supply power to the device. This might be required when the load is much greater than the 15 mA average load that PRoC can support.

Figure 3. PMU Enabled

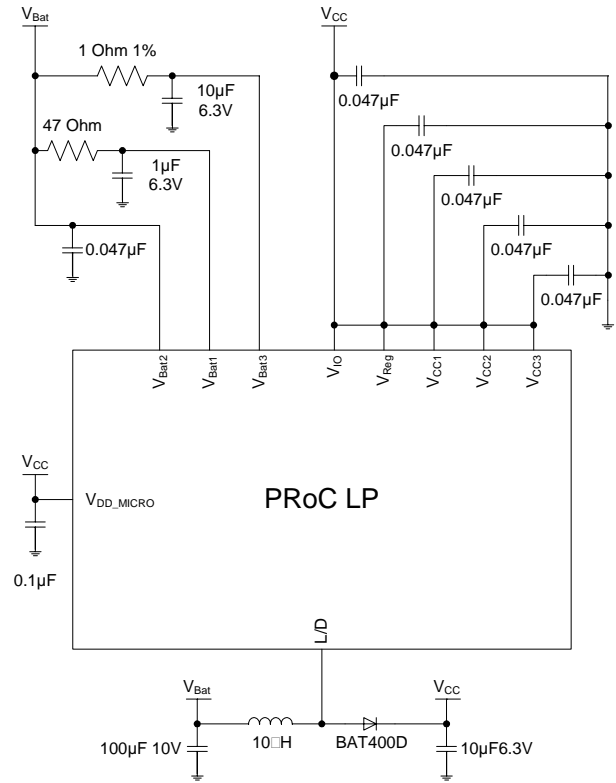


Figure 4. PMU Disabled - Linear Regulator

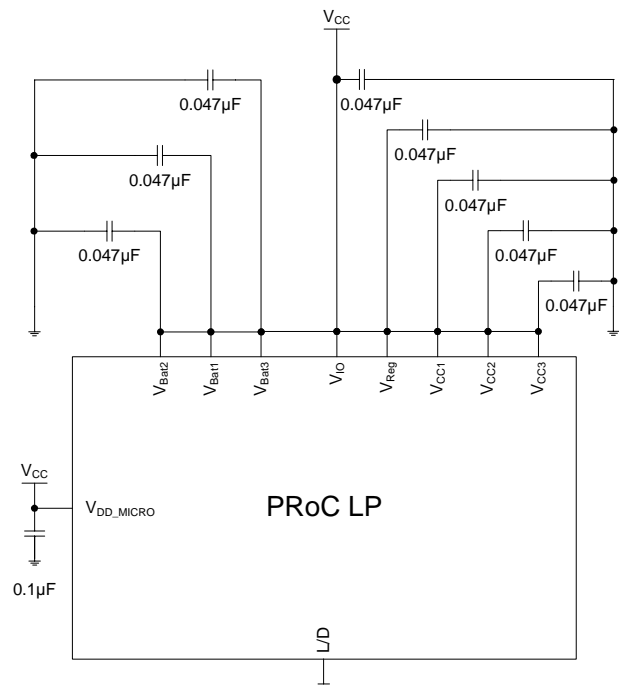
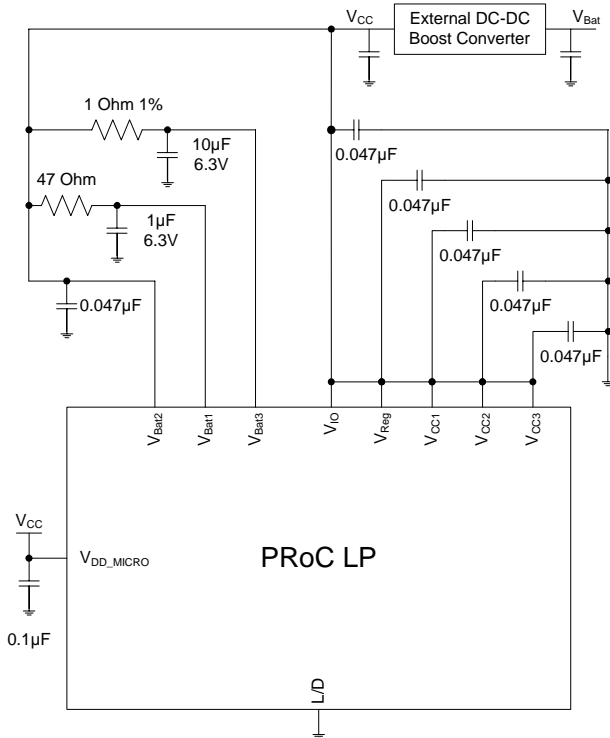


Figure 5. PMU Disabled - External Boost Converter



**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 given 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**

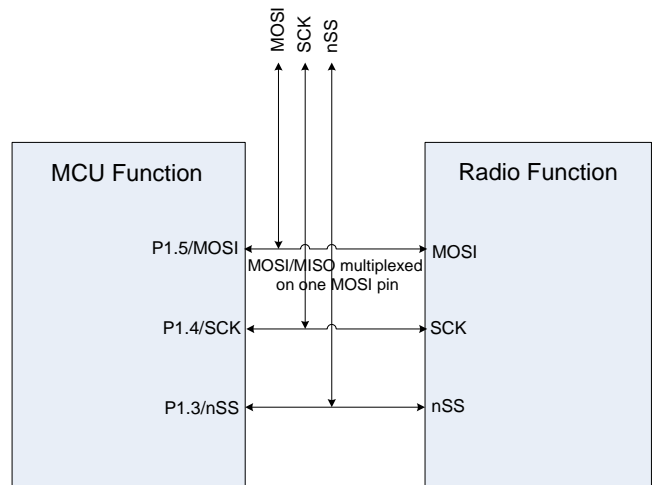
The SPI interface between the MCU function and the radio function is a 3-wire SPI Interface. The three pins are MOSI

(Master Out Slave In), SCK (Serial Clock), SS (Slave Select). 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).

**3-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 should 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 prior to 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 should ensure that the MOSI function on MCU function is in a high-impedance state whenever SS is high.

Figure 6. 3-Wire SPI Mode



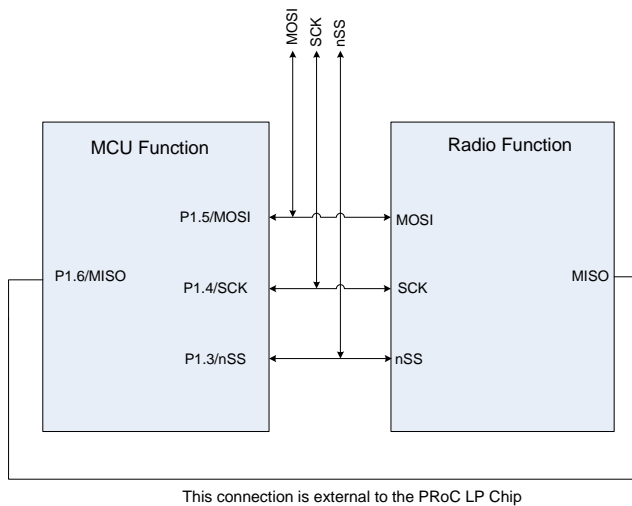
**4-Wire SPI Interface**

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 should ensure that the MOSI function on MCU function is in a high-impedance state whenever SS is high.



Figure 7. 4-Wire SPI Mode



**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 8.

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

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 IO Voltage References**

The SPI interfaces between MCU function and the radio and the IRQ and RST have a separate voltage reference  $V_{IO}$ . For CYRF69103  $V_{IO}$  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.

Figure 8. 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 2. 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.

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 19.

## CPU Registers

### Flags Register

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

**Table 3. 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							
<b>Note:</b> This register is readable with explicit address 0xF7. The <i>OR F, expr</i> and <i>AND F, expr</i> must be used to set and clear the CPU_F bits								

### Accumulator Register

**Table 4. 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 5. 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 6. 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 7. 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 8. 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 9. 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 10.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 11.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 12.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 13.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 14.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 15. 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 16. 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 17. 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 18. 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**

Instruction Set Summary tables are described in detail in the *PSoC Designer Assembly Language User Guide* (available on the [www.cypress.com](http://www.cypress.com) web site).

The instruction set is summarized in [Table 19](#) numerically and serves as a quick reference. If more information is needed, the

**Table 19. 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		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]	if (A=B) Z=1	68	7	2	ASR [expr]	C, Z
0F	10	3	ADC [X+expr], expr	C, Z	3C	8	3	CMP [expr], expr	if (A<B) C=1	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**

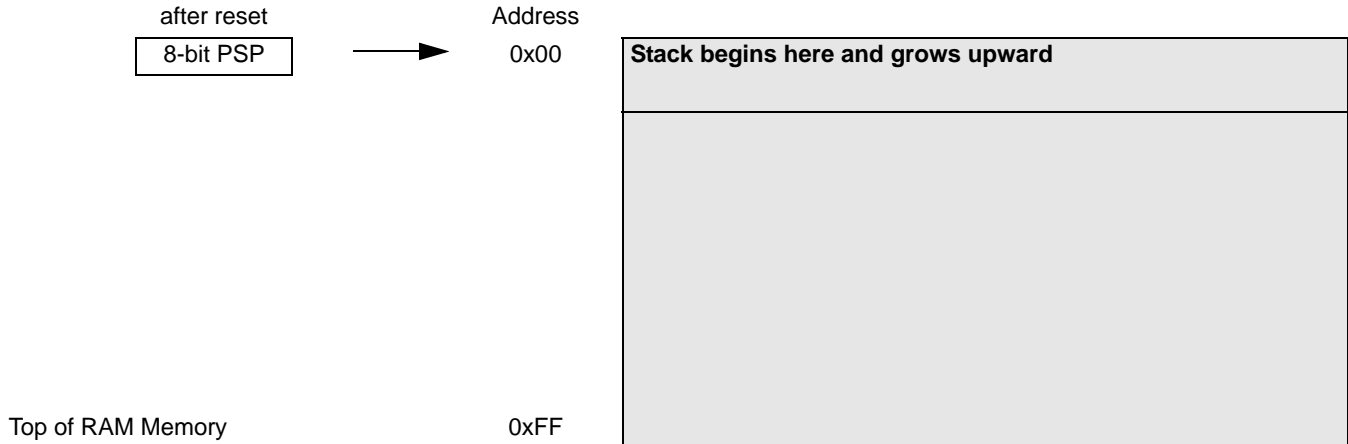
1. Interrupt routines take 13 cycles before execution resumes at interrupt vector table.
2. The number of cycles required by an instruction is increased by one for instructions that span 256-byte boundaries in the Flash memory space.



**Data Memory Organization**

The MCU function provides up to 256 bytes of data RAM

**Figure 10. Data Memory Organization**



**Flash**

This section describes the Flash block of the CYRF69103. Much of the user-visible Flash functionality, including programming and security, are implemented in the M8C Supervisory Read Only Memory (SROM). CYRF69103 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 via 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 'boot-loader'—a piece of firmware resident on the Flash. For safety reasons this boot-loader should 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*

CYRF69103 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 CYRF69103 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 20 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. Prior to executing the SSC, the M8C's accumulator needs to be loaded with the desired SROM function code from Table 20. Undefined functions will cause 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 21 lists all possible parameter block variables. The meaning of each parameter, with regards to a specific SROM function, is described later in this chapter

**Table 20.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 would be 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 will halt (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 21.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,FCCh
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 22.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 will happen, 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 will not execute when the SSC instruction is executed with a bad key value and a nonzero function code. A CYRF69103 device will execute 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 will exit setting the accumulator and KEY2 back to 00h. KEY1 will have a value of 01h, indicating a read failure. If read protection is not enabled, the function will read 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 will be stored in SRAM at the address indicated by the value of the POINTER parameter. When the ReadBlock completes successfully, the accumulator, KEY1 and KEY2, will all have a value of 00h.

**Table 23.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 should 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 will exit, setting the accumulator and KEY2 back to 00h. KEY1 will have 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 will be 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 24. 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) 4-KB Flash block number (00h–3Fh) 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 prior to 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 will exit, setting the accumulator and KEY2 back to 00h. KEY1 will have a value of 01h, indicating a write failure. The EraseBlock function is only useful as the first step in programming. Erasing a block will 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 set up 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 25. 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 CYRF69103 device offers Flash protection on a block-by-block basis. Table 26 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 26. 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 will store the protection level for blocks 252 through 255.

**Table 26. 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 prior to calling the function. The parameter block values that must be set, besides the keys, are the CLOCK and DELAY values.

**Table 27. 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 28. 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 29. 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 CYRF69103 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 CYRF69103 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 will always be set to FFh when trim values are read. The BLOCKID value, in the parameter block, is used to indicate which table should be returned to the user. Only the three least significant bits of the BLOCKID parameter are used by the TableRead function for the CYRF69103. 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 will calculate the checksum of only block 0, while a BLOCKID value of 0 will calculate 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 30. 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 CYRF69103 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°–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 CYRF69103 in suspend mode, particularly to generate a periodic wake-up interrupt and also to 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.

CYRF69103 provides the ability to load new trim values for the 24-MHz oscillator based on voltage. This allows V<sub>dd</sub> 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. CYRF69103 is initialized with 3.30V 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<sub>dd</sub>. The 32-KHz oscillator generally does not require trim adjustments vs. voltage but trim values for the 32-KHz are also stored in Supervisory ROM.

**Table 31. Oscillator Trim Values vs. Voltage Settings**

Supervisory FLASH User Row Address	Function
0xC094	24-MHz IOSCTR @ 3.30V
0xC095	24-MHz IOSCTR @ 3.00V
0xC096	24-MHz IOSCTR @ 2.85V
0xC097	24-MHz IOSCTR @ 2.70V
0xC098	32-kHz LPOSCTR@3.30V
0xC099	32-kHz LPOSCTR@3.00V
0xC09A	32-kHz LPOSCTR@2.85V
0xC09B	32-kHz LPOSCTR@2.70V

When using the 32-KHz oscillator the PITMRL/H should be read until two consecutive readings match before

**Table 32. CPU Clock Config (CPUCLKCR) [0x30] [R/W]**

Bit #	7	6	5	4	3	2	1	0
Field	Reserved							Reserved
Read/Write	-	-	-	-	-	-	-	-
Default	0	0	0	0	0	0	0	0
Bits 7:1	Reserved							
Bit 0	CPU CLK Select 0 = Internal 24-MHz Oscillator 1 = Reserved to 0							
<b>Note</b>	The CPU speed selection is configured using the OSC_CR0 Register (Figure 11)							

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
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
    
```

The CYRF69103 can optionally be sourced from an external crystal oscillator. The external clock driving on CLKIN range is from 187 KHz to 24 MHz.

**Clock Architecture Description**

The CYRF69103 clock selection circuitry allows the selection of independent clocks for the CPU, Interval Timers, and Capture Timers.

On the CYRF69103, the external oscillator can be sourced by the crystal oscillator, or when the crystal oscillator is disabled it is sourced directly from the CLKIN pin.

*CPU Clock*

The CPU clock, CPUCLK, can be sourced from the external crystal oscillator, the Internal 24-MHz Oscillator, or the Internal 32-KHz Low-power Oscillator. The selected clock source can optionally be divided by 2<sup>n-1</sup> where n is 0-7 (see Table 33).

**Table 33.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	0	0	0	0
Bits 7:6	Reserved							
Bit 5	No Buzz During sleep (the Sleep bit is set in the CPU_SCR Register— <a href="#">Table 37</a> ), the LVD and POR detection circuit is turned on periodically to detect any POR and LVD events on the V <sub>CC</sub> pin (the Sleep Duty Cycle bits in the ECO_TR are used to control the duty cycle— <a href="#">Table 41</a> ). To facilitate the detection of POR and LVD events, the No Buzz bit is used to force the LVD and POR detection circuit to be continuously enabled during sleep. This results in a faster response to an LVD or 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 LVD and POR detection circuit is turned on periodically as configured in the Sleep Duty Cycle 1 = The Sleep Duty Cycle value is overridden. The LVD and POR detection circuit is always enabled							
<b>Note</b> The periodic Sleep Duty Cycle enabling is independent with the sleep interval shown in the Sleep [1:0] bits below								
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					
<b>Note</b> Sleep intervals are approximate								
Bits 2:0	CPU Speed [2:0] The CYRF69103 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	External Clock						
000	3 MHz (Default)	Clock In/8						
001	6 MHz	Clock In/4						
010	12 MHz	Clock In/2						
011	Reserved	Reserved						
100	1.5 MHz	Clock In/16						
101	750 KHz	Clock In/32						
110	187 KHz	Clock In/128						
111	Reserved	Reserved						

**Table 34. Timer Clock Config (TMRCLKCR) [0x31] [R/W]**

Bit #	7	6	5	4	3	2	1	0
<b>Field</b>	TCAPCLK Divider		TCAPCLK Select		ITMRCLK Divider		ITMRCLK Select	
<b>Read/Write</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>Default</b>	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							
<b>Note</b> The 1024- $\mu$ s interval timer is based on the assumption that TCAPCLK is running at 4 MHz. Changes in TCAPCLK frequency will 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							
<b>Note</b> 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 will not be successful								

*Interval Timer Clock (ITMRCLK)*

The Interval Timer clock (ITMRCLK) can be sourced from the internal 24-MHz oscillator, the internal 32-kHz low-power oscillator, or the 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 should 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 will appear on the device editor view of PSoC Designer once you place the enCoRe II 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:

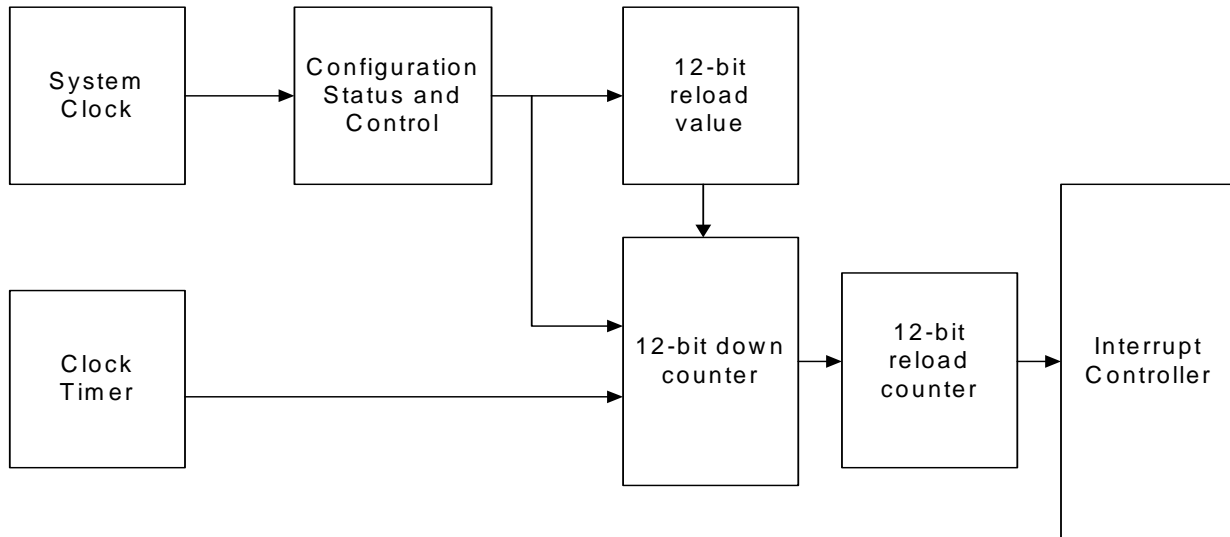
TCAPCLK divide by x of CPU clock (for example TCAPCLK divide by 2 of a 24-MHz CPU clock will give a frequency of 12 MHz)

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 external crystal oscillator, 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 11. Programmable Interval Timer Block Diagram



Internal Clock Trim

Table 35. 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 @ 3.30V 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 will be 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 36. LPOSC Trim (LPOSCTR) [0x36] [R/W]**

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

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 [-: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
<b>Important Note</b> 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 CPUCLK Select (Bit [0], [Table 32](#)) is forced to the internal oscillator, and 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.

If the system requires the CPU to run off the external clock after awakening from sleep mode, firmware will need to switch the clock source for the CPU. If the external clock source is the external oscillator and the oscillator is disabled, firmware will need to enable the external oscillator, wait for it to stabilize, and then change the clock source.

**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 37. 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 CYRF69103 device.

Bit 7	<p><b>GIES</b></p> <p>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 will service interrupts</p> <p>0 = Global interrupts disabled 1 = Global interrupt enabled</p>
Bit 6	Reserved
Bit 5	<p><b>WDRS</b></p> <p>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</p> <p>0 = No WDR 1 = A WDR event has occurred</p>
Bit 4	<p><b>PORS</b></p> <p>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</p> <p>0 = No POR 1 = A POR event has occurred. (Note that WDR events will not occur until this bit is cleared.)</p>
Bit 3	<p><b>SLEEP</b></p> <p>Set by the user to enable CPU sleep state. CPU will remain in sleep mode until any interrupt is pending. The Sleep bit is covered in more detail in the Sleep Mode section</p> <p>0 = Normal operation 1 = Sleep</p>
Bits 2:1	Reserved
Bit 0	<p><b>STOP</b></p> <p>This bit is set by the user to halt the CPU. The CPU will remain 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 would be to use the HALT instruction rather than writing to this bit</p> <p>0 = Normal CPU operation 1 = CPU is halted (not recommended)</p>

### Power-on Reset

POR occurs every time the power to the device is switched on. POR is released when the supply is typically 2.6V 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. Once the PORS bit is

cleared, the WDT cannot be disabled. The only exception to this is if a POR event takes place, which will disable 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 33). When the sleep time elapses (sleep timer overflows), an interrupt to the Sleep Timer Interrupt Vector will be 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 will be cleared. If the data that is written

#### Note

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

is the hex value 0x38, the Sleep Timer will also be cleared at the same time.

**Table 38. 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 will clear the Watchdog Timer, a write of 0x38 will also clear 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 will remain asleep until an interrupt comes pending, or there is a reset event (either a Power-on Reset, or a Watchdog Timer Reset).

The Low-voltage Detection circuit (LVD) drops into fully functional power-reduced states, and the latency for the LVD is increased. The actual latency can be traded against power consumption by changing the Sleep Duty Cycle field of the ECO\_TR Register.

The Internal 32-KHz Low-speed Oscillator remains running. Prior to 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 36). This will help save approximately 5 μA; however, the trade off is that the 32-KHz Low-speed Oscillator will be less accurate (-53.12% to +56.25% deviation).

All interrupts remain active. Only the occurrence of an interrupt will wake 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 will wake the system up. As a result, any interrupts not intended for waking should be disabled through the Interrupt Mask Registers.

When the CPU enters sleep mode the CPUCLK Select (Bit 1, Table 32) is forced to the Internal Oscillator. The internal oscillator recovery time is three clock cycles of the Internal 32-kHz Low-power Oscillator. The Internal 24-MHz Oscillator restarts immediately on exiting Sleep mode. If the external crystal oscillator is used, firmware will need to switch the clock source for the CPU.

Unlike the Internal 24-MHz Oscillator, the external oscillator is not automatically shut down during sleep. Systems that need the external oscillator disabled in sleep mode will need to disable the external oscillator prior to entering sleep mode. In systems where the CPU runs off the external oscillator, firmware will need to switch the CPU to the internal oscillator prior to disabling the external oscillator.

On exiting sleep mode, once 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 12 and is defined as follows.

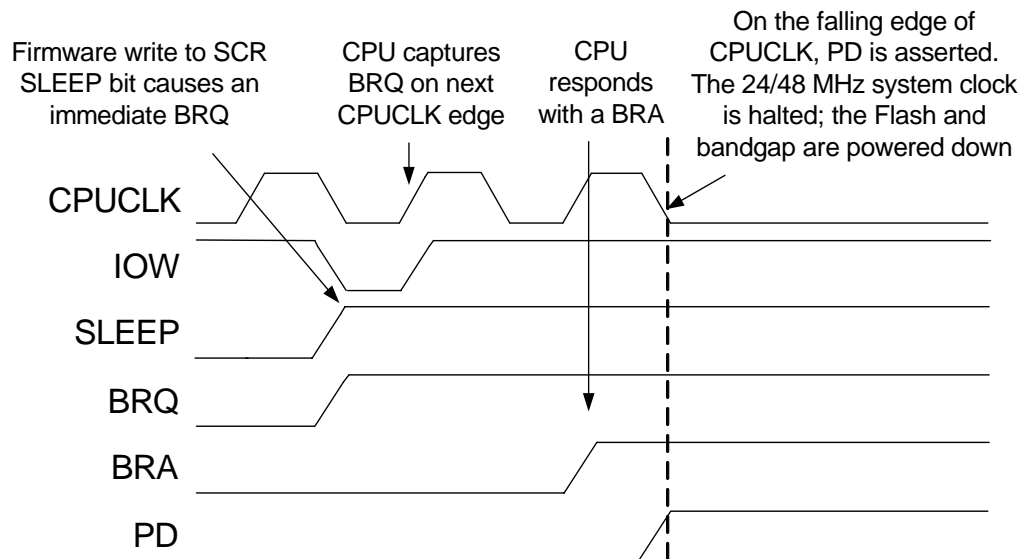
1. Firmware sets the SLEEP bit in the CPU\_SCR0 register. The Bus Request (BRQ) signal to the CPU is immediately asserted. This is a request by the system to halt CPU operation at an instruction boundary. The CPU samples BRQ on the positive edge of CPUCLK.
2. Due to the specific timing of the register 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 12 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/LVD) circuit.

The external crystal oscillator on CYRF69103 devices is not automatically powered down when the CPU enters the sleep state. Firmware must explicitly disable the external crystal oscillator in order to reduce power to levels specified.

**Note** To achieve the lowest possible power consumption during suspend/sleep, the following conditions must be observed in addition to considerations for the sleep timer and external crystal oscillator:

- All GPIOs must be set to outputs and driven low
- The port pins P1.0 and P1.1 should be configured as inputs with their pull ups enabled.

Figure 12. Sleep Timing



**Wakeup Sequence**

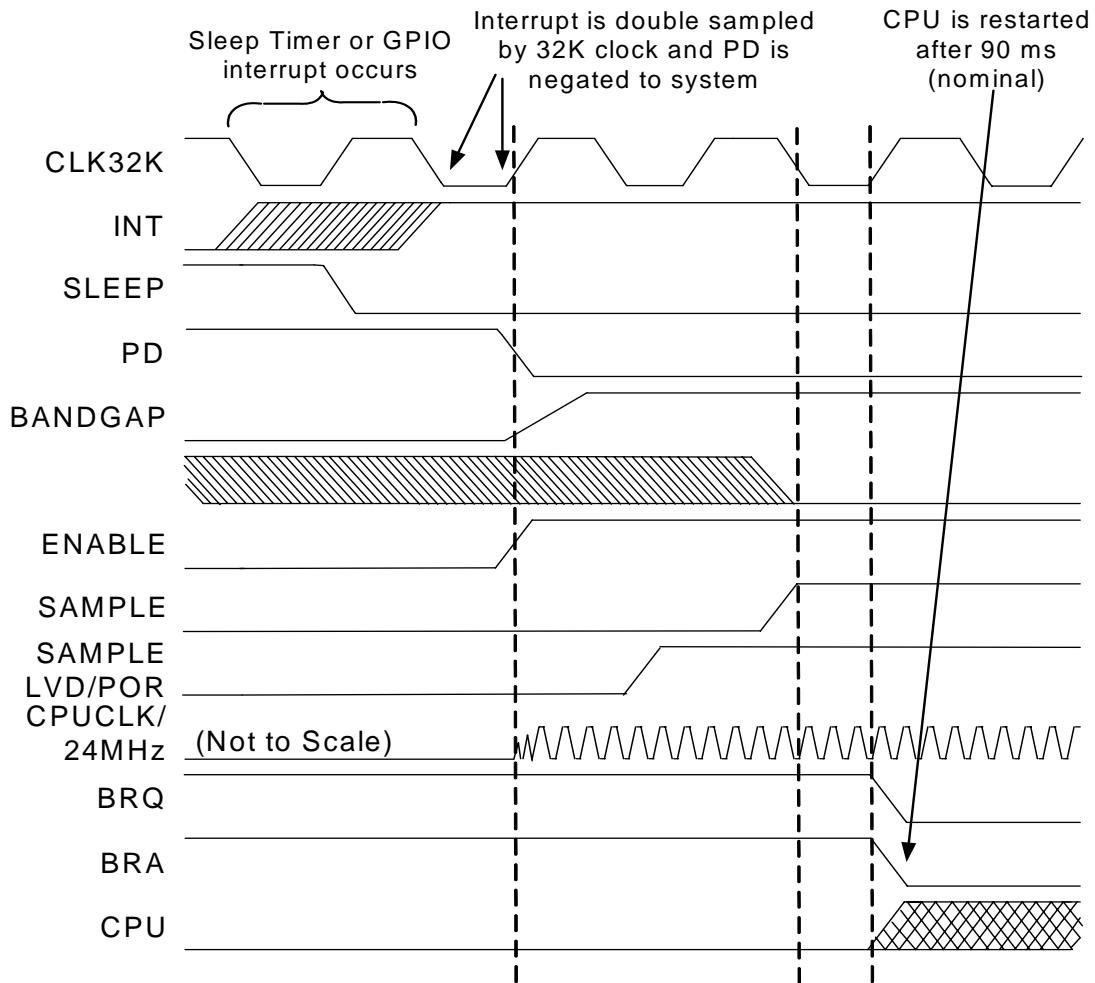
Once asleep, the only event that can wake the system up is an interrupt. The global interrupt enable of the CPU flag register does not need to be set. Any unmasked interrupt will wake the system up. It is optional for the CPU to actually take the interrupt after the wakeup sequence. The wakeup sequence is synchronized to the 32-KHz clock for purposes of sequencing a startup delay, to allow the Flash memory module enough time to power up before the CPU asserts the first read access. Another reason for the delay is to allow the oscillator, Bandgap, and LVD/POR circuits time to settle before actually being used in the system. As shown in Figure 13, the wake p 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 and LVD 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 13 fixed function blocks, such as Flash, internal oscillator, EFTB, and bandgap, have about 15 μs start up. The wakeup times (interrupt to CPU operational) will range from 75 μs to 105 μs.

Figure 13. Wakeup Timing





**Low-Voltage Detect Control**

**Table 39. Low-voltage Control Register (LVDCR) [0x1E3] [R/W]**

Bit #	7	6	5	4	3	2	1	0
Field	Reserved		PORLEV[1:0]		Reserved	VM[2:0]		
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 configuration of the Power-on Reset/Low-voltage Detection 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 = 2.7V Range (trip near 2.6V)

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

1 0 = Reserved

1 1 = PPOR will not generate a reset, but values read from the Voltage Monitor Comparators Register (Table 40) give the internal PPOR comparator state with trip point set to the 3V range setting.

Bit 3 Reserved

Bits 2:0 VM[2:0]

This field controls the level below which the low-voltage-detect trips—possibly generating an interrupt and the level at which the Flash is enabled for operation.

VM[2:0]	LVD Trip Point (V)		
	Min.	Max.	Typical
000	2.69	2.72	2.7
001	2.90	2.94	2.92
010	3.00	3.04	3.02
011	3.10	3.15	3.13
100	Reserved		
101	Reserved		
110	Reserved		
111	Reserved		

**POR Compare State**

**Table 40. Voltage Monitor Comparators Register (VLTCMP) [0x1E4] [R]**

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

This read-only register allows reading the current state of the Low-voltage Detection and Precision-Power-On-Reset comparators

Bits 7:2 Reserved

Bit 1 LVD

This bit is set to indicate that the low-voltage-detect comparator has tripped, indicating that the supply voltage has gone below the trip point set by VM[2:0] (See Table 39.)

0 = No low-voltage-detect event

1 = A low-voltage-detect has tripped

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 41. 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 LVD and POR detection circuit

Bits 7:6 Sleep Duty Cycle [1:0]  
 0 0 = 128 periods of the Internal 32-KHz Low-speed Oscillator  
 0 1 = 512 periods of the Internal 32-KHz Low-speed Oscillator  
 1 0 = 32 periods of the Internal 32-KHz Low-speed Oscillator  
 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 42. 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/CLKOUT	Reserved
Read/Write	R/W	-		R/W	R/W	R/W	-	
Default	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–INT0  
 In addition to their use as the P0.4–P0.2 GPIOs, these pins can also be used for the alternative functions as the Interrupt pins (INT0–INT2). To configure the P0.4–P0.2 pins, refer to the P0.2/INT0–P0.4/INT2 Configuration Register (Table 46)

Bit 2 Reserved  
 Bit 1 P0.1 Data/Clock-output.  
 Bit 0 Reserved

**Table 43. 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 7:6 P1.7- P1.6  
 Bits 5:3 P1.5–P1.3 Data/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)

Bits 2:1 P1.2–P1.1  
 Bit 0 P1.0

**Table 44.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. In order to prevent the inputs from floating, the pull-up resistors are enabled. Firmware will need to configure each of the GPIOs prior to 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 as well.

*Int Act Low*

When clear, the corresponding interrupt is active HIGH. When set, the interrupt is active LOW. For P0.2–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 will have 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 45.P0.1/CLKOUT Configuration (P01CR) [0x06] R/W**

Bit #	7	6	5	4	3	2	1	0
Field	CLK Output	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 pin is shared between P0.1 GPIO use and the CLKOUT pin for the external crystal oscillator. When the external oscillator is enabled the settings of this register are ignored. When CLK output is set, the internally selected clock is sent out onto P0.1CLKOUT pin

The alternate function of the pin as the CLKOUT is only available in the CY7C601xx. When the external oscillator is enabled (the XOSC Enable bit of the CLKIOCR Register is set—[Table 73](#)), the GPIO function of the pin is disabled

The 50-mA sink drive capability is only available in the CY7C601xx. In the CY7C602xx, 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 will draw current. This pin should be configured as an input to reduce current draw

Bit 7            CLK Output  
0 = The clock output is disabled  
1 = The clock selected by the CLK Select field (Bit [1:0] of the CLKIOCR Register—[Table 73](#)) is driven out to the pin

**Table 46.P0.3–P0.4 Configuration (P03CR–P04CR) [0x07–0x09] [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 P0.2–P0.4 respectively. These pins are shared between the P0.2–P0.4 GPIOs and the INTO–INT2. The INTO–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 71](#) and [Table 69](#))

To use these pins as interrupt inputs, configure them as inputs by clearing the corresponding Output Enable. If the INTO–INT2 pins are configured as outputs with interrupts enabled, firmware can generate an interrupt by writing the appropriate value to the P0.2, 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 P0.2/INT0–P0.4/INT2 pins are individually configured with the P02CR (0x07), 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 47.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

**Table 48.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

**Table 49.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.

**Table 50.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

**Table 51.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  
**Important Note for Comm Modes 01 or 10 (SPI Master or SPI Slave, see Table 55)**  
 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 52. 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

**Table 53.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

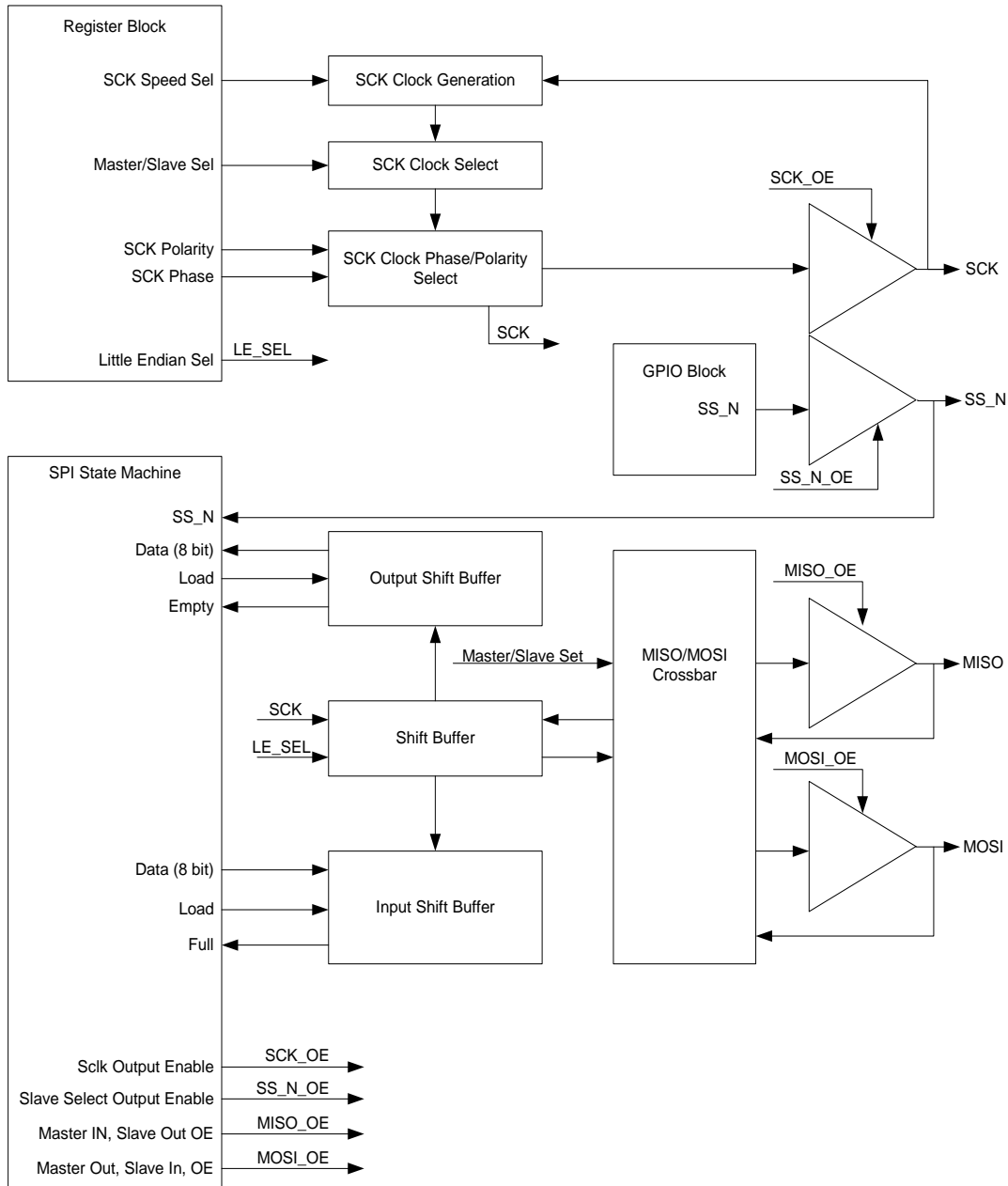
This register controls the operation of pins P2.0–P2.1



### 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 14. SPI Block Diagram



**SPI Data Register**

**Table 54. 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 an 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 will result in incorrect data transfer.

**SPI Configure Register**

**Table 55. 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

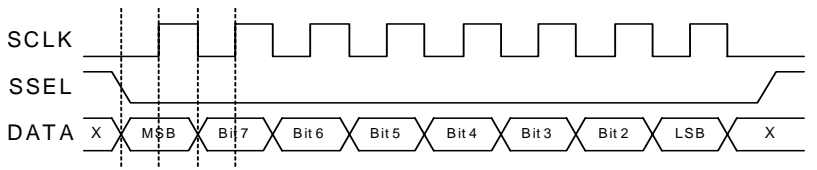
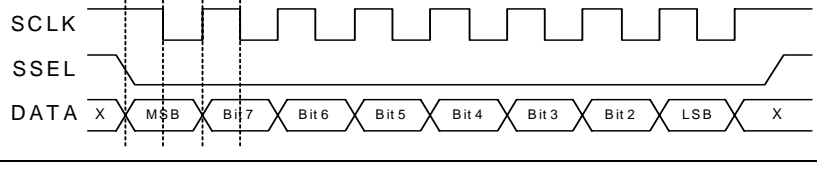
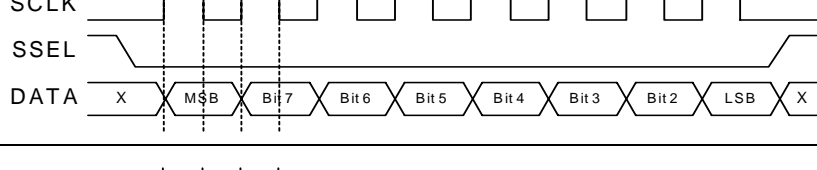
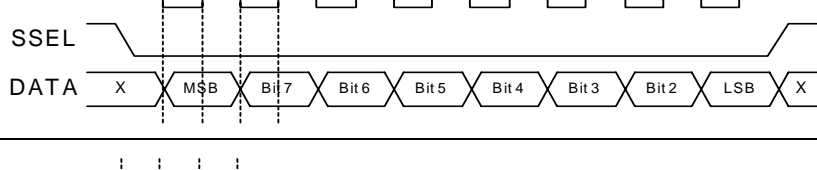
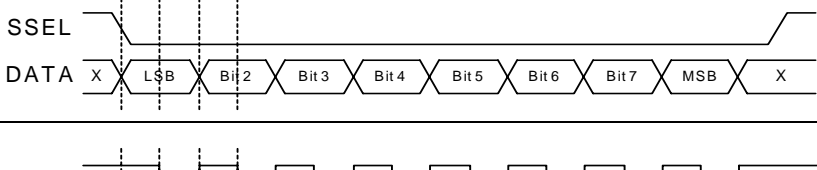
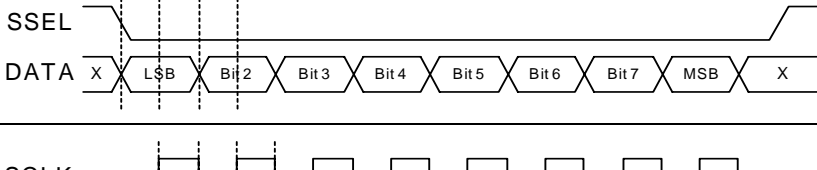
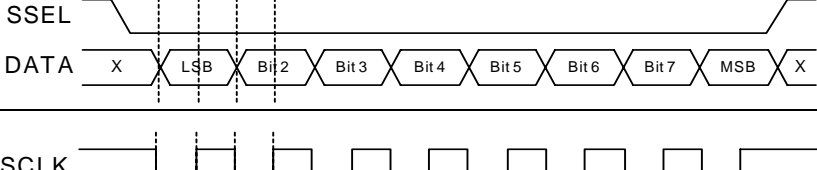
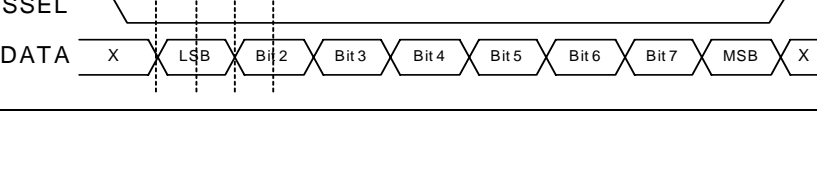
Bit 3 CPOL  
 This bit controls the SPI clock (SCLK) idle polarity  
 0 = SCLK idles low  
 1 = SCLK idles high

Bit 2 CPHA  
 The Clock Phase bit controls the phase of the clock on which data is sampled. [Table 56](#) 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 51](#)), 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 56.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 57. 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 CYRF69103 are provided by a single timer block. The timer block is asynchronous from the CPU

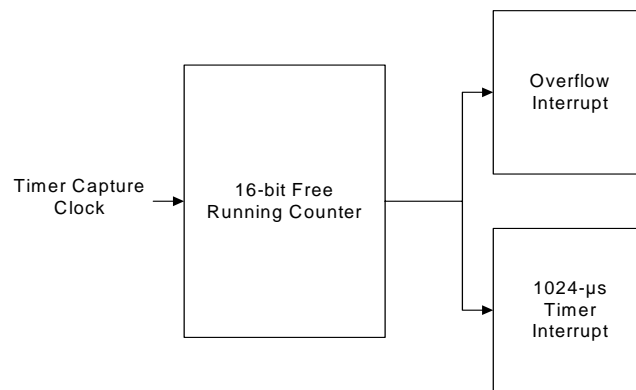
clock. The 16-bit free-running counter is used as the time-base for timer captures and can also be used as a general time-base by software.

**Registers**

*Free-Running Counter*

The 16-bit free-running counter is clocked by a 4-/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 15. 16-bit Free Running Counter Block Diagram**



**Table 58. 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 should be read first and the high-order second. When writing, the low-order byte should be written first then the high-order byte

**Table 59.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 should be read first and the high-order second. When writing, the low-order byte should be written first then the high-order byte.								

**Table 60.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 61.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 62.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 63.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 16. 16-Bit Free-Running Counter Loading Timing Diagram

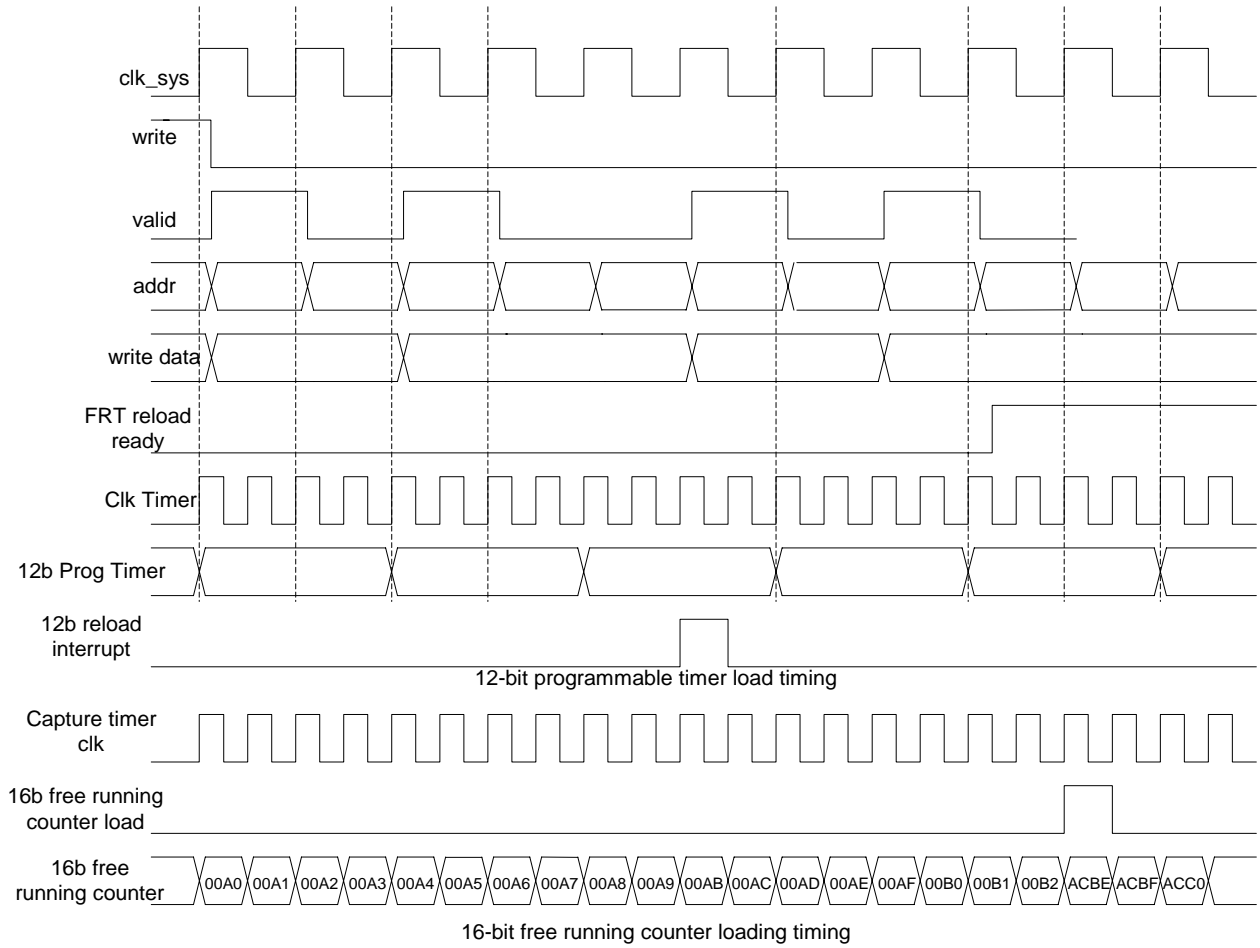
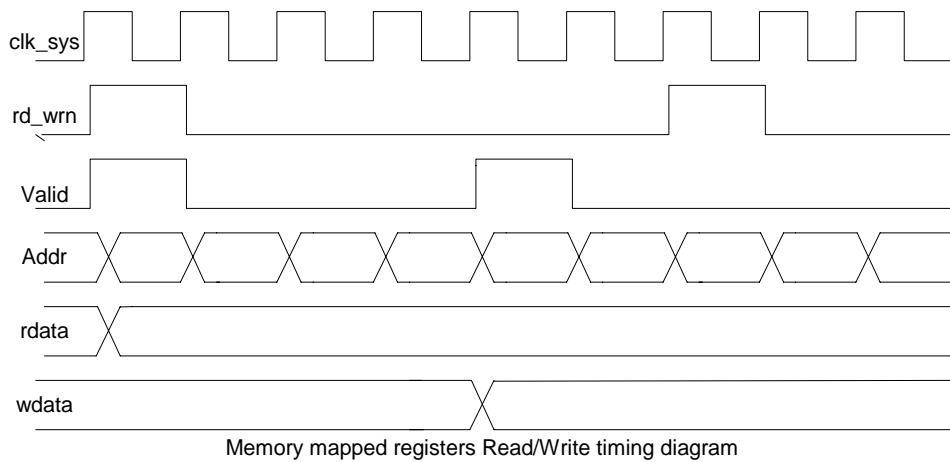


Figure 17. Memory Mapped Registers Read/Write Timing Diagram





### 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 CYRF69103 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 CYRF69103.

**Table 64. Interrupt Priorities, Address, Name**

Interrupt Priority	Interrupt Address	Name
0	0000h	Reset
1	0004h	POR/LVD
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
16	0040h	Reserved

**Table 64. Interrupt Priorities, Address, Name (continued)**

Interrupt Priority	Interrupt Address	Name
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 18](#) clocking in a '1'. The interrupt will remain posted until the interrupt is taken or until it is cleared by writing to the appropriate INT\_CLRx register.

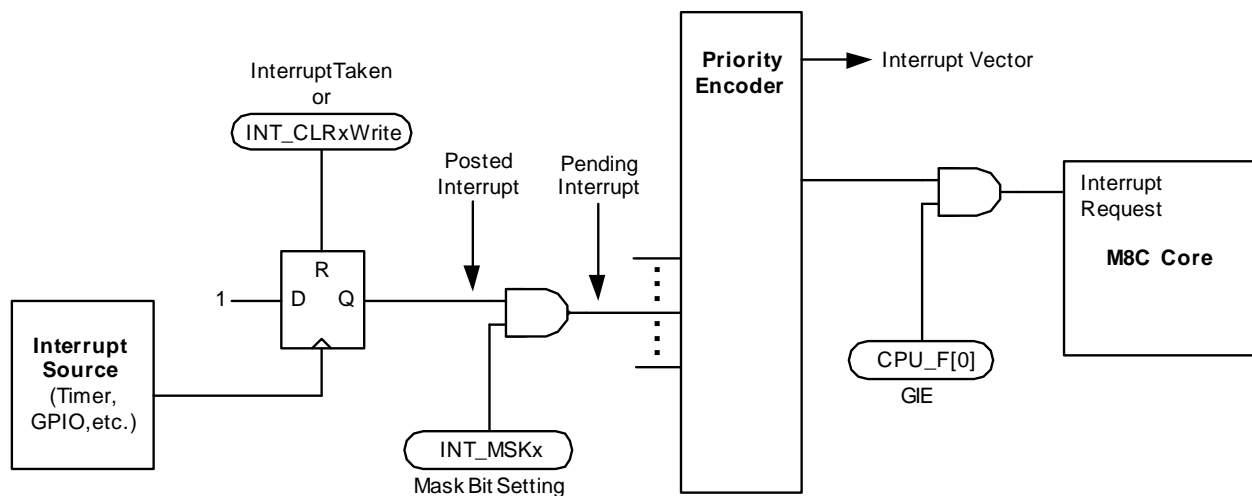
A posted interrupt is not pending unless it is enabled by setting its interrupt mask bit (in the appropriate INT\_MSKx register). All pending interrupts are processed by the Priority Encoder to determine the highest priority interrupt which will be taken by the M8C if the Global Interrupt Enable bit 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 CYRF69103 Interrupt Controller is shown in [Figure 18](#).

**Figure 18. 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.
2. The current executing instruction finishes.
3. 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. Since 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/LVD interrupt).
4. 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.
5. The ISR executes. Note that interrupts are disabled since GIE = 0. In the ISR, interrupts can be re-enabled if desired

by setting GIE = 1 (care must be taken to avoid stack overflow).

6. 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, since GIE = 1 again.
7. Execution resumes at the next instruction, after the one that occurred before the interrupt. However, if there are more pending interrupts, the subsequent interrupts will be 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 would be as follows:

(1 to 5 cycles for JMP to finish) + (13 cycles for interrupt routine) + (7 cycles for LJMP) = 21 to 25 cycles.

In the example above, 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 65. 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/LVD
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's no posted interrupt for the corresponding hardware  
 1 = Posted interrupt for the corresponding hardware present  
 Writing a '0' to the bits will clear the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT (Bit 7 of the INT\_MSK3 Register) will post the corresponding hardware interrupt.

**Table 66. 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's no posted interrupt for the corresponding hardware  
 1 = Posted interrupt for the corresponding hardware present  
 Writing a '0' to the bits will clear the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT  
 Bit 7           Reserved

**Table 67. 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's no posted interrupt for the corresponding hardware  
 1 = Posted interrupt for the corresponding hardware present  
 Writing a '0' to the bits will clear the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT (Bit 7 of the INT\_MSK3 Register) will post 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), which 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 will become 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, will cause 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, will 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 68. 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, will cause the corresponding interrupt to clear  
 1 = Enable. Writing 1's to an INT\_CLRx register, when ENSWINT is set, will cause the corresponding interrupt to post  
 Bits 6:0       Reserved

**Table 69. Interrupt Mask 2 (INT\_MSK2) [0xDF] [R/W]**

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

**Table 70. Interrupt Mask 1 (INT\_MSK1) [0xE1] [R/W]**

Bit #	7	6	5	4	3	2	1	0
<b>Field</b>	Reserved	Prog Interval Timer Int Enable	1-ms Timer Int Enable	Reserved				
<b>Read/Write</b>	R/W	R/W	R/W	–	–	–	–	–
<b>Default</b>	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 71. Interrupt Mask 0 (INT\_MSK0) [0xE0] [R/W]**

Bit #	7	6	5	4	3	2	1	0
<b>Field</b>	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/LVD Int Enable
<b>Read/Write</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>Default</b>	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/LVD Interrupt Enable 0 = Mask POR/LVD interrupt 1 = Unmask POR/LVD interrupt							

*Interrupt Vector Clear Register*
**Table 72. Interrupt Vector Clear Register (INT\_VC) [0xE2] [R/W]**

Bit #	7	6	5	4	3	2	1	0
<b>Field</b>	Pending Interrupt [7:0]							
<b>Read/Write</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>Default</b>	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 will clear 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 will clear 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/CLKOUT	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	CLK Output	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable	bbbbbbbbb	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		
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							CPU CLK Select Reserved to 0.	-----b	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-ddddd		
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/LVD	bbbbbb-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		
E1	INT_MSK1	Reserved	Prog Interval Timer Int Enable	1-ms Timer Int Enable	Reserved						-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/LVD Int Enable	bbbbbb-b	00000000		
E2	INT_VC	Pending Interrupt [7:0]								bbbbbbbbb	00000000		
E3	RESWDT	Reset Watchdog Timer [7:0]								wwwwwwww	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	00000000		
1E3	LVDCR	Reserved		PORLEV[1:0]		Reserved	VM[2:0]			--bb-bbb	00000000		
1EB	ECO_TR	Sleep Duty Cycle [1:0]			Reserved							bb-----	00000000
1E4	VLTCOMP	Reserved						LVD	PPOR	-----rr	00000000		



### Radio Function Register Summary

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 non-incrementing burst read.

**Table 73. Register Map Summary**

Address	Mnemonic	b7	b6	b5	b4	b3	b2	b1	b0	Default <sup>(4)</sup>	Access <sup>(4)</sup>		
0x00	CHANNEL_ADR	Not Used		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	DATA MODE		PA SETTING			--000101	--bbbbbb		
0x04	TX_IRQ_STATUS_ADR	OS IRQ	LV IRQ	TXB15 IRQ	TXB8 IRQ	TXB0 IRQ	TXBERR IRQ	TXC IRQ	TXE IRQ	10111000	rrrrrrrr		
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	SOFDDET IRQ	RXB16 IRQ	RXB8 IRQ	RXB1 IRQ	RXBERR IRQ	RXC IRQ	RXE IRQ	00000000	brrrrrrr		
0x08	RX_STATUS_ADR	RX ACK	PKT ERR	EOP ERR	CRC0	Bad CRC	RX Code	RX Data Mode		00001---	rrrrrrrr		
0x09	RX_COUNT_ADR	RX Count								00000000	rrrrrrrr		
0x0A	RX_LENGTH_ADR	RX Length								00000000	rrrrrrrr		
0x0B	PWR_CTRL_ADR	PMU EN	LVIRQ EN	PMU Mode Force	Not Used	LVI TH		PMU OUTV		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	PACTL OD	PACTL GPIO	SPI 3PIN	IRQ GPIO	00000000	bbbbbbb		
0x0E	GPIO_CTRL_ADR	XOUT OP	MISO OP	PACTL OP	IRQ OP	XOUT IP	MISO IP	PACTL IP	IRQ IP	0000----	bbbbrrrr		
0x0F	XACT_CFG_ADR	ACK EN	Not Used	FRC END	END STATE			ACK TO		1-000000	b-bbbbb		
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	---bbbb			
0x12	DATA64_THOLD_ADR	Not Used	Not Used	Not Used	TH64					---01010	---bbbb		
0x13	RSSI_ADR	SOP	Not Used	LNA	RSSI					0-100000	r-rrrrrr		
0x14	EOP_CTRL_ADR	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								-----	rrrrrrrr		
0x18	TX_CRC_MSB_ADR	CRC MSB								-----	rrrrrrrr		
0x19	RX_CRC_LSB_ADR	CRC LSB								11111111	rrrrrrrr		
0x1A	RX_CRC_MSB_ADR	CRC MSB								11111111	rrrrrrrr		
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	www-w-w		
0x1E	RX_OVERRIDE_ADR	ACK RX	RXTX DLY	MAN RXACK	FRC RXDR	DIS CRC0	DIS RXCRC	ACE	Not Used	0000000-	bbbbbbb-		
0x1F	TX_OVERRIDE_ADR	ACK TX	FRC PRE	RSVD	MAN TXACK	OVRD ACK	DIS TXCRC	RSVD	TX INV	00000000	bbbbbbb		
0x27	CLK_OVERRIDE_ADR	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RXF	RSVD	00000000	wwwwww		
0x28	CLK_EN_ADR	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RXF	RSVD	00000000	wwwwww		
0x29	RX_ABORT_ADR	RSVD	RSVD	ABORT EN	RSVD	RSVD	RSVD	RSVD	RSVD	00000000	wwwwww		
0x32	AUTO_CAL_TIME_ADR	AUTO_CAL_TIME_MAX								00000011	wwwwww		
0x35	AUTO_CAL_OFFSET_ADR	AUTO_CAL_OFFSET_MINUS_4								00000000	wwwwww		
0x39	ANALOG_CTRL_ADR	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	ALL SLOW	00000000	wwwwww		
Register Files													
0x20	TX_BUFFER_ADR	TX Buffer File								-----	wwwwww		
0x21	RX_BUFFER_ADR	RX Buffer File								-----	rrrrrrrr		
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	rrrrrrrr		

**Notes**

- 4. b = read/write, r = read only, w = write only, - = not used, default value is undefined.
- 5. SOP\_CODE\_ADR default = 0x17FF9E213690C782.
- 6. DATA\_CODE\_ADR default = 0x02F9939702FA5CE3012BF1DB0132BE6F.
- 7. PREAMBLE\_ADR default = 0x333302.

Mnemonic	CHANNEL_ADR						Address	0x00
Bit	7	6	5	4	3	2	1	0
Default	-	1	0	0	1	0	0	0
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	Not Used Channel							
Bits 6:0	<p>This field selects the channel. 0x00 sets 2400 MHz; 0x62 sets 2498 MHz. Values above 0x62 are not valid. The default channel is a fast channel above the frequency typically used in non-overlapping WiFi systems. Any write to this register will impact the time it takes the synthesizer to settle.</p> <p>fast (100-μs) - 0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 96            medium (180-μs) - 2 4 8 10 14 16 20 22 26 28 32 34 38 40 44 46 50 52 56 58 62 64 68 70 74 76 78 80 82 84 86 88 90 92 94            slow (270-μs) - 1 5 7 11 13 17 19 23 25 29 31 35 37 41 43 47 49 53 55 59 61 65 67 71 73 75 77 79 81 83 85 87 89 91 93 95 97</p> <p>Usable channels subject to regulation.            Do not access or modify the register during a transmit or receive.</p>							

Mnemonic	TX_LENGTH_ADR						Address	0x01
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	TX Length							
Bits 7:0	<p>This register sets the length of the packet to be transmitted. A length of zero is valid, and will transmit a packet with SOP, length and CRC16 fields (if enabled), but no data field. Packet lengths of more than 16 bytes will require that some data bytes be written after transmission of the packet has begun. Typically, length is updated prior to setting TX GO. The maximum packet length for all packets is 40 bytes except for framed 64-chip DDR where the maximum packet length is 16 bytes.</p> <p>Maximum packet length is limited by the delta between the transmitter and receiver crystals of 60-ppm or better.</p>							

Mnemonic	TX_CTRL_ADR						Address	0x02
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	TX GO	TX CLR	TXB15 IRQEN	TXB8 IRQEN	TXB0 IRQEN	TXBERR IRQEN	TXC IRQEN	TXE IRQEN
Bit 7	<p>Start Transmission. Setting this bit triggers the transmission of a packet. Writing a 0 to this flag has no effect. This bit is cleared automatically at the end of packet transmission. The transmit buffer may be loaded either before or after setting this bit. If data is loaded after setting this bit, the length of time available to load the buffer depends on the starting state (sleep, idle or synth), the length of the SOP code, the length of preamble, and the packet data rate. For example, if starting from idle mode on a fast channel in 8DR mode with 32 chip SOP codes the time available is 100 μs (synth start) + 32 μs (preamble) + 64 μs (SOP length) + 32 μs (length byte) = 228 μs. If there are no bytes in the TX buffer at the end of transmission of the length field, a TXBERR IRQ will occur.</p>							
Bit 6	<p>Clear TX Buffer. Writing a 1 to this register clears the transmit buffer. Writing a 0 to this bit has no effect. The previous packet may be retransmitted by setting TX GO and not setting this bit. A new transmit packet may be loaded and transmitted without setting this bit if TX GO is set after the new packet is loaded to the buffer. If the TX_BUFFER_ADR is to be loaded after the TX GO bit has been set, then this bit should be set before loading a new transmit packet to the buffer and before TX GO is set.</p>							
Bit 5	<p>Buffer Not Full Interrupt Enable. See TX_IRQ_STATUS_ADR for description.</p>							
Bit 4	<p>Buffer Half Empty Interrupt Enable. See TX_IRQ_STATUS_ADR for description.</p>							
Bit 3	<p>Buffer Empty Interrupt Enable. See TX_IRQ_STATUS_ADR for description.</p>							
Bit 2	<p>Buffer Error Interrupt Enable. See TX_IRQ_STATUS_ADR for description.</p>							
Bit 1	<p>Transmission Complete Interrupt Enable. See TX_IRQ_STATUS_ADR for description. TXC IRQEN and TXE IRQEN must be set together.</p>							
Bit 0	<p>Transmit Error Interrupt Enable. See TX_IRQ_STATUS_ADR for description. TXC IRQEN and TXE IRQEN must be set together.</p>							

Mnemonic	TX_CFG_ADR						Address	0x03
Bit	7	6	5	4	3	2	1	0
Default	-	-	0	0	0	1	0	1
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Function	Not Used	Not Used	Data Code Length	Data Mode		PA Setting		
Bit 5	Data Code Length. This bit selects the length of the DATA_CODE_ADR code for the data portion of the packet. This bit is ignored when the data mode is set to GFSK. 1 = 64 chip codes. 0 = 32 chip codes.							
Bits 4:3	Data Mode. This field sets the data transmission mode. 00 = 1-Mbps GFSK, 01 = 8DR Mode, 10 = DDR Mode, 11 = SDR Mode. It is recommended that firmware sets the ALL SLOW bit in register ANALOG_CTRL_ADR when using GFSK data rate mode.							
Bits 2:0	PA Setting. This field sets the transmit signal strength. 0 = -30 dBm, 1 = -25 dBm, 2 = -20 dBm, 3 = -15 dBm, 4 = -10 dBm, 5 = -5 dBm, 6 = 0 dBm, 7 = +4 dBm.							

Mnemonic	TX_IRQ_STATUS_ADR						Address	0x04
Bit	7	6	5	4	3	2	1	0
Default	1	0	1	1	1	0	0	0
Read/Write	R	R	R	R	R	R	R	R
Function	OS IRQ	LV IRQ	TXB15 IRQ	TXB8 IRQ	TXB0 IRQ	TXBERR IRQ	TXC IRQ	TXE IRQ
The state of all IRQ status bits is valid regardless of whether or not the IRQ is enabled. The IRQ output of the device is in its active state whenever one or more bits in this register is set and the corresponding IRQ enable bit is also set. Status bits are non-atomic (different flags may change value at different times in response to a single event). In particular, standard error handling is only effective if the premature termination of a transmission due to an exception does not leave the device in an inconsistent state.								
Bit 7	Oscillator Stable IRQ Status. This bit is set when the internal crystal oscillator has settled (synthesizer sequence starts).							
Bit 6	Low Voltage Interrupt Status. This bit is set when the voltage on V <sub>BAT</sub> is below the LVI threshold (see PWR_CTL_ADR). This interrupt is automatically disabled whenever the PMU is disabled. When enabled, this bit reflects the voltage on V <sub>BAT</sub> .							
Bit 5	Buffer Not Full Interrupt Status. This bit is set whenever there are 15 or fewer bytes remaining in the transmit buffer.							
Bit 4	Buffer Half Empty Interrupt Status. This bit is set whenever there are 8 or fewer bytes remaining in the transmit buffer.							
Bit 3	Buffer Empty Interrupt Status. This bit is set at any time that the transmit buffer is empty.							
Bit 2	Buffer Error Interrupt Status. This IRQ is triggered by either of two events: (1) When the transmit buffer (TX_BUFFER_ADR) is empty and the number of bytes remaining to be transmitted is greater than zero; (2) When a byte is written to the transmit buffer and the buffer is already full. This IRQ is cleared by setting bit TX CLR in TX_CTRL_ADR.							
Bit 1	Transmission Complete Interrupt Status. This IRQ is triggered when transmission is complete. If transaction mode is not enabled then this interrupt is triggered immediately after transmission of the last bit of the CRC16. If transaction mode is enabled, this interrupt is triggered at the end of a transaction. Reading this register clears this bit. TXC IRQ and TXE IRQ flags may change value at different times in response to a single event. If transaction mode is enabled and the first read of this register returns TXC IRQ = 1 and TXE IRQ = 0 then firmware must execute a second read to this register to determine if an error occurred by examining the status of TXE. There can be a case when this bit is not triggered when ACK EN = 1 and there is an error in transmission. If the first read of this register returns TXC IRQ = 1 and TXE IRQ = 1 then the firmware must not execute a second read to this register for a given transaction. If an ACK is received RXC IRQ and RXE IRQ may be asserted instead of TXC IRQ and TXE IRQ.							
Bit 0	Transmit Error Interrupt Status. This IRQ is triggered when there is an error in transmission. This interrupt is only applicable to transaction mode. It is triggered whenever no valid ACK packet is received within the ACK timeout period. Reading this register clears this bit.							

Mnemonic	RX_CTRL_ADR						Address	0x05
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	RX GO	RSVD	RXB16 IRQEN	RXB8 IRQEN	RXB1 IRQEN	RXBERR IRQEN	RXC IRQEN	RXE IRQEN

Status bits are non-atomic (different flags may change value at different times in response to a single event).

Bit 7 Start Receive. Setting this bit causes the device to transition to receive mode. If necessary, the crystal oscillator and synthesizer will start automatically after this bit is set. Firmware must never clear this bit. This bit must not be set until after it self clears. The recommended method to exit receive mode when an error has occurred is to force END STATE and then dummy read all RX\_COUNT\_ADR bytes from RX\_BUFFER\_ADR or poll RSSI\_ADR.SOP (bit 7) until set. See XACT\_CFG\_ADR and RX\_ABORT\_ADR for description.

Bit 6 Start of Packet Detect Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Bit 5 Buffer Full Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Bit 4 Buffer Half Empty Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Bit 3 Buffer Not Empty Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Bit 2 Buffer Error Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Bit 1 Packet Reception Complete Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Bit 0 Receive Error Interrupt Enable. See RX\_IRQ\_STATUS\_ADR for description.

Mnemonic	RX_CFG_ADR						Address	0x06
Bit	7	6	5	4	3	2	1	0
Default	1	0	0	1	0	-	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
Function	AGC EN	LNA	ATT	HILO	FAST TURN EN	Not Used	RXOW EN	VLD EN

Status bits are non-atomic (different flags may change value at different times in response to a single event).

Bit 7 Automatic Gain Control (AGC) Enable. When this bit is set, AGC is enabled, and the LNA is controlled by the AGC circuit. When this bit is cleared the LNA is controlled manually using the LNA bit. Typical applications will clear this bit during initialization. It is recommended that this bit be disabled and bit 6 (LNA) be enabled unless the device will be used in a system where it may receive data from a device using an external PA to transmit signals at >+4 dBm.

Bit 6 Low Noise Amplifier (LNA) Manual Control. When AGC EN (Bit 7) is cleared, this bit controls the state of the receiver LNA; when AGC EN is set, this bit has no effect. Setting this bit enables the LNA; clearing this bit disables the LNA. Device current in receive mode is slightly lower when the LNA is disabled. Typical applications will set this bit during initialization.

Bit 5 Receive Attenuator Enable. Setting this bit enables the receiver attenuator. The receiver attenuator may be used to desensitize the receiver so that only very strong signals may be received. This bit should only be set when the AGC EN is disabled and the LNA is manually disabled.

Bit 4 HILO. When FAST TURN EN is set, this bit is used to select whether the device will use the high frequency for the channel selected, or the low frequency. 1 = hi; 0 = lo. When FAST TURN EN is not enabled this also controls the highlow bit to the receiver and should be left at the default value of 1 for high side receive injection. Typical applications will clear this bit during initialization.

Bit 3 Fast Turn Mode Enable. When this bit is set, the HILO bit determines whether the device receives data transmitted 1MHz above the RX Synthesizer frequency or 1 MHz below the receiver synthesizer frequency. Use of this mode allows for very fast turnaround, because the same synthesizer frequency may be used for both transmit and receive, thus eliminating the synthesizer resettle period between transmit and receive. Note that when this bit is set, and the HILO bit is cleared, received data bits are automatically inverted to compensate for the inversion of data received on the "image" frequency. Typical applications will set this bit during initialization.

Bit 1 Overwrite Enable. When this bit is set, if an SOP is detected while the receive buffer is not empty, then the existing contents of receive buffer are lost, and the new packet is loaded into the receive buffer. When this bit is set, the RXOW IRQ is enabled. If this bit is cleared, then the receive buffer may not be overwritten by a new packet, and whenever the receive buffer is not empty SOP conditions are ignored, and it is not possible to receive data until the previously received packet has been completely read from the receive buffer.

Bit 0 Valid Flag Enable. When this bit is set, the receive buffer can store only 8 bytes of data. The other half of the buffer is used to store valid flags. See RX\_BUFFER\_ADR for more detail.

Mnemonic	RX_IRQ_STATUS_ADR						Address	0x07
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	R/W	R	R	R	R	R	R	R
Function	RXOW IRQ	RSVD	RXB16 IRQ	RXB8 IRQ	RXB1 IRQ	RXBERR IRQ	RXC IRQ	RXE IRQ

The state of all IRQ Status bits is valid regardless of whether or not the IRQ is enabled. The IRQ output of the device is in its active state whenever one or more bits in this register is set and the corresponding IRQ enable bit is also set. Status bits are non-atomic (different flags may change value at different times in response to a single event). In particular, standard error handling is only effective if the premature termination of a transmission due to an exception does not leave the device in an inconsistent state.

- Bit 7 Receive Overwrite Interrupt Status. This IRQ is triggered when the receive buffer is overwritten by a packet being received before the previous packet has been read from the buffer. This bit is cleared by writing any value to this register. This condition is only possible when the RXOW EN bit in RX\_CFG\_ADR is set. This bit must be written '1' by firmware before the new packet may be read from the receive buffer.
- Bit 6 Reserved. Must not be set.
- Bit 5 Receive Buffer Full Interrupt Status. This bit is set whenever the receive buffer is full, and cleared otherwise.
- Bit 4 Receive Buffer Half Full Interrupt Status. This bit is set whenever there are 8 or more bytes remaining in the receive buffer. Firmware must read exactly eight bytes when reading RXB8 IRQ. It is possible, in rare cases, that the last byte of a packet may remain in the buffer even though the RXB1\_IRQ flag has cleared. This can ONLY happen on the last byte of a packet and only if the packet data is being read out of the buffer while the packet is still being received. The flag is trustworthy under all other conditions, and for all bytes prior to the last. When using RXB1\_IRQ and unloading the packet data during reception, the user should be sure to check the RX\_COUNT\_ADR value after the RXC/RXE is set and unload the last remaining byte if the number of bytes unloaded is less than the reported count, even though the RXB1\_IRQ is not set.
- Bit 3 Receive Buffer Not Empty Interrupt Status. This bit is set at any time that there are 1 or more bytes in the receive buffer, and cleared when the receive buffer is empty. RXB1 IRQ must not be set when RXB8 IRQ is set and vice versa.
- Bit 2 Receive Buffer Error Interrupt Status. This IRQ is triggered in one of two ways: (1) When the receive buffer is empty and there is an attempt to read data. (2) When the receive buffer is full and more data is received. This flag is cleared when RX GO is set and a SOP is received.
- Bit 1 Packet Receive Complete Interrupt Status. This IRQ is triggered when a packet has been received. If transaction mode is enabled, then this bit is not set until after transmission of the ACK. If transaction mode is not enabled then this bit is set as soon as a valid packet is received. This bit is cleared when this register is read. RXC IRQ and RXE IRQ flags may change value at different times in response to a single event. There are cases when this bit is not triggered when ACK EN = 1 and there is an error in reception. Therefore, firmware should examine RXC IRQ, RXE IRQ, and CRC 0 to determine receive status. If the first read of this register returns RXC IRQ = 1 and RXE IRQ = 0 then firmware must execute a second read to this register to determine if an error occurred by examining the status of RXE IRQ. If the first read of this register returns RXC IRQ = 1 and RXE IRQ = 1 then the firmware must not execute a second read to this register for a given transaction.
- Bit 0 Receive Error Interrupt Status. This IRQ is triggered when there is an error in reception. It is triggered whenever a packet is received with a bad CRC16, an unexpected EOP is detected, a packet type (data or ACK) mismatch, or a packet is dropped because the receive buffer is still not empty when the next packet starts. The exact cause of the error may be determined by reading RX\_STATUS\_ADR. This bit is cleared when this register is read.

Mnemonic	RX_STATUS_ADR						Address	0x08
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	1	-	-	-
Read/Write	R	R	R	R	R	R	R	R
Function	RX ACK	PKT ERR	EOP ERR	CRC0	Bad CRC	RX Code	RX Data Mode	
<p>It is expected that firmware does not read this register until after TX GO self clears. Status bits are non-atomic (different flags may change value at different times in response to a single event).</p> <p>Bit 7      RX Packet Type. This bit is set when the received packet is an ACK packet, and cleared when the received packet is a standard packet.</p> <p>Bit 6      Receive Packet Type Error. This bit is set when the packet type received is what not was expected and cleared when the packet type received was as expected. For example, if a data packet is expected and an ACK is received, this bit will be set.</p> <p>Bit 5      Unexpected EOP. This bit is set when an EOP is detected before the expected data length and CRC16 fields have been received. This bit is cleared when SOP pattern for the next packet has been received. This includes the case where there are invalid bits detected in the length field and the length field is forced to 0.</p> <p>Bit 4      Zero-seed CRC16. This bit is set whenever the CRC16 of the last received packet has a zero seed.</p> <p>Bit 3      Bad CRC16. This bit is set when the CRC16 of the last received packet is incorrect.</p> <p>Bit 2      Receive Code Length. This bit indicates the DATA_CODE_ADR code length used in the last correctly received packet. 1 = 64-chip code, 0 = 32-chip code.</p> <p>Bits 1:0    Receive Data Mode. These bits indicate the data mode of the last correctly received packet. 00 = 1-Mbps GFSK, 01 = 8DR, 10 = DDR, 11 = Not Valid. These bits do not apply to unframed packets.</p>								

Mnemonic	RX_COUNT_ADR						Address	0x09
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
Function	RX Count							
<p>Count bits are non-atomic (updated at different times).</p> <p>Bits 7:0    This register contains the total number of payload bytes received during reception of the current packet. After packet reception is complete, this register will match the value in RX_LENGTH_ADR unless there was a packet error. This register is reset to 0x00 when RX_LENGTH_ADR is loaded. Count should not be read when RX_GO = 1 during a transaction.</p>								

Mnemonic	RX_LENGTH_ADR						Address	0x0A
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
Function	RX Length							
<p>Length bits are non-atomic (different flags may change value at different times in response to a single event).</p> <p>Bits 7:0    This register contains the length field, which is updated with the reception of a new length field (shortly after start of packet detected). If there is an error in the received length field, 0x00 is loaded instead, except when using GFSK datarate, and an error is flagged.</p>								



Mnemonic	PWR_CTRL_ADR					Address	0x0B	
Bit	7	6	5	4	3	2	1	0
Default	1	0	1	-	0	0	0	0
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
Function	PMU EN	LVIRQ EN	PMU Mode Force	Not Used	LVI TH		PMU OUTV	
Bit 7	Power Management Unit (PMU) Enable. Setting this bit enables the PMU if PMU Mode Force bit (bit-5) is set, otherwise it has no effect. See PMU Mode Force bit description.							
Bit 6	Low Voltage Interrupt Enable. Setting this bit enables the LV IRQ interrupt. When this interrupt is enabled, if the $V_{BAT}$ voltage falls below the threshold set by LVI TH, then a low voltage interrupt will be generated. The LVI is not available when the device is in sleep mode. The LVI event on IRQ pin is automatically disabled whenever the PMU is disabled.							
Bit 5	PMU Mode Force. If this bit is set the PMU operation will be based on the value of the PMU EN bit (bit 7). If this bit is not set the PMU is disabled during sleep. Otherwise it is enabled when not in Sleep mode.							
Bits 3:2	Low Voltage Interrupt Threshold. This field sets the voltage on $V_{BAT}$ at which the LVI is triggered. 11 = 1.8V, 10 = 2.0V, 01 = 2.2V, 00 = PMU OUTV voltage.							
Bits 1:0	PMU Output Voltage. This field sets the minimum output voltage of the PMU. 11 = 2.4V, 10 = 2.5V, 01 = 2.6V, 00 = 2.7V. When the PMU is active, the voltage output by the PMU on $V_{REG}$ will never be less than this voltage provided that the total load on the $V_{REG}$ pin is less than the specified maximum value, and the voltage in $V_{BAT}$ is greater than the specified minimum value.							
To force the chip to always enable the PMU (even during sleep), set bits 5 and 7. To force the chip to always disable the PMU, set bit 5 and clear bit 7. To allow the chip to disable the PMU during sleep, clear bit 5.								
The sequence of writing bits to this register impacts the sleep current $I_{SB}$ .								

Mnemonic	XTAL_CTRL_ADR					Address	0x0C	
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	-	-	1	0	0
Read/Write	R/W	R/W	R/W	-	-	R/W	R/W	R/W
Function	XOUT FN		XSIRQ EN	Not Used	Not Used	FREQ		
Bits 7:6	XOUT Pin Function. This field selects between the different functions of the XOUT pin. 00 = Clock frequency set by XOUT FREQ; 01 = Active LOW PA Control; 10 = Radio data serial bit stream. If this option is selected and SPI is configured for 3-wire mode then the MISO pin will output a serial clock associated with this data stream; 11 = GPIO. To disable this output, set to GPIO mode, and set the GPIO state in IO_CFG_ADR.							
Bit 5	Crystal Stable Interrupt Enable. This bit enables the OS IRQ interrupt. When enabled, this interrupt generates an IRQ event when the crystal has stabilized after the device has woken from sleep mode. This event is cleared by writing zero to this bit.							
Bits 2:0	XOUT Frequency. This field sets the frequency output on the XOUT pin when XOUT FN is set to 00. 0 = 12 MHz, 1 = 6 MHz, 2 = 3 MHz, 3 = 1.5 MHz, 4 = 0.75 MHz; other values are not defined.							

Mnemonic	IO_CFG_ADR						Address	0x0D
	7	6	5	4	3	2		
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	IRQ OD	IRQ POL	MISO OD	XOUT OD	PACTL OD	PACTL GPIO	SPI 3PIN	IRQ GPIO
<p>To use a GPIO pin as an input, the output mode must be set to open drain, and a '1' written to the corresponding output register bit.</p> <p>Bit 7      IRQ Pin Drive Strength. Setting this bit configures the IRQ pin as an open drain output. Clearing this bit configures the IRQ pin as a standard CMOS output, with the output '1' drive voltage being equal to the <math>V_{IO}</math> pin voltage.</p> <p>Bit 6      IRQ Polarity. Setting this bit configures the IRQ signal polarity to be active HIGH. Clearing this bit configures the IRQ signal polarity to be active low.</p> <p>Bit 5      MISO Pin Drive Strength. Setting this bit configures the MISO pin as an open drain output. Clearing this bit configures the MISO pin as a standard CMOS output, with the output '1' drive voltage being equal to the <math>V_{IO}</math> pin voltage.</p> <p>Bit 4      XOUT Pin Drive Strength. Setting this bit configures the XOUT pin as an open drain output. Clearing this bit configures the XOUT pin as a standard CMOS output, with the output '1' drive voltage being equal to the <math>V_{IO}</math> pin voltage.</p> <p>Bit 3      PACTL Pin Drive Strength. Setting this bit configures the PACTL pin as an open drain output. Clearing this bit configures the PACTL pin as a standard CMOS output, with the output '1' drive voltage being equal to the <math>V_{IO}</math> pin voltage.</p> <p>Bit 2      PACTL Pin Function. When this bit is set the PACTL pin is available for use as a GPIO.</p> <p>Bit 1      SPI Mode. When this bit is cleared, the SPI interface acts as a standard 4-wire SPI Slave interface. When this bit is set, the SPI interface operates in '3-Wire Mode' combining MISO and MOSI on the same pin (SDAT), and the MISO pin is available as a GPIO pin.</p> <p>Bit 0      IRQ Pin Function. When this bit is cleared, the IRQ pin is asserted when an IRQ is active; the polarity of this IRQ signal is configurable in IRQ POL. When this bit is set, the IRQ pin is available for use as a GPIO pin, and the IRQ function is multiplexed onto the MOSI pin. In this case the IRQ signal state is presented on the MOSI pin whenever the <math>\overline{SS}</math> signal is inactive (HIGH).</p>								

Mnemonic	GPIO_CTRL_ADR						Address	0x0E
	7	6	5	4	3	2		
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	-	-	-	-
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Function	XOUT OP	MISO OP	PACTL OP	IRQ OP	XOUT IP	MISO IP	PACTL IP	IRQ IP
<p>To use a GPIO pin as an input, the output mode must be set to open drain, and a '1' written to the corresponding output register bit.</p> <p>Bit 7      XOUT Output. When the XOUT pin is configured to be a GPIO, the state of this bit sets the output state of the XOUT pin.</p> <p>Bit 6      MISO Output. When the MISO pin is configured to be a GPIO, the state of this bit sets the output state of the MISO pin.</p> <p>Bit 5      PACTL Output. When the PACTL pin is configured to be a GPIO, the state of this bit sets the output state of the PACTL pin.</p> <p>Bit 4      IRQ Output. When the IRQ pin is configured to be a GPIO, the state of this bit sets the output state of the IRQ pin.</p> <p>Bit 3      XOUT Input. When the XOUT pin is configured to be a GPIO, the state of this bit reflects the voltage on the XOUT pin.</p> <p>Bit 2      MISO Input. When the MISO pin is configured to be a GPIO, the state of this bit reflects the voltage on the MISO pin.</p> <p>Bit 1      PACTL Input. When the PACTL pin is configured to be a GPIO, the state of this bit reflects the voltage on the PACTL pin.</p> <p>Bit 0      IRQ Input. When the IRQ pin is configured to be a GPIO, the state of this bit reflects the voltage on the IRQ pin.</p>								

Mnemonic	XACT_CFG_ADR						Address	0x0F
Bit	7	6	5	4	3	2	1	0
Default	1	-	0	0	0	0	0	0
Read/Write	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
Function	ACK EN	Not Used	FRC END	END STATE			ACK TO	
Bit 7	Acknowledge Enable. When this bit is set, an ACK packet is automatically transmitted whenever a valid packet is received; in this case the device is considered to be in transaction mode. After transmission of the ACK packet, the device automatically transitions to the END STATE. When this bit is cleared, the device transitions directly to the END STATE immediately after the end of packet transmission.							
Bit 5	Force End State. Setting this bit forces a transition to the state set in END STATE. By setting the desired END STATE at the same time as setting this bit the device may be forced to immediately transition from its current state to any other state. This bit is automatically cleared upon completion.							
Bits 4:2	Transaction End State. This field defines the mode to which the device transitions after receiving or transmitting a packet. 000 = Sleep Mode, 001 = Idle Mode, 010 = Synth Mode (TX), 011 = Synth Mode (RX), 100 = RX Mode. In normal use, this field will typically be set to 000 or 001 when the device is transmitting packets, and 100 when the device is receiving packets. Note that when the device transitions to receive mode as an END STATE, the receiver must still be armed by setting RX GO before the device can begin receiving data. If the system only support packets <=16 bytes then firmware should examine RXC IRQ and RXE IRQ to determine the status of the packet. If the system supports packets > 16 bytes ensure that END STATE is not sleep, force RXF = 1, perform receive operation, force RXF = 0, and if necessary set END STATE back to sleep.							
Bits 1:0	ACK Timeout. When the device is configured for transaction mode, this field sets the timeout period after transmission of a packet during which an ACK must be correctly received in order to prevent a transmit error condition from being detected. This timeout period is expressed in terms of a number of SOP_CODE_ADR code lengths; if SOP LEN is set, then the timeout period is this value multiplied by 64 μs and if SOP LEN is cleared then the timeout is this value multiplied by 32 μs. 00 = 4x, 01 = 8x, 10 = 12x, 11 = 15x the SOP_CODE_ADR code length. ACK_TO must be set to greater than 30 + Data Code Length (only for 8DR) + Preamble Length + SOP Code Length (x2).							

Mnemonic	FRAMING_CFG_ADR						Address	0x10
Bit	7	6	5	4	3	2	1	0
Default	1	0	1	0	0	1	0	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	SOP EN	SOP LEN	LEN EN	SOP TH				
Bit 7	SOP Enable. When this bit is set, each transmitted packet begins with a SOP field, and only packets beginning with a valid SOP field will be received. If this bit is cleared, no SOP field will be generated when a packet is transmitted, and packet reception will begin whenever two successive correlations against the DATA_CODE_ADR code are detected.							
Bit 6	SOP PN Code Length. When this bit is set the SOP_CODE_ADR code length is 64 chips. When this bit is cleared the SOP_CODE_ADR code length is 32 chips.							
Bit 5	Packet Length Enable. When this bit is set the 8-bit value contained in TX_LENGTH_ADR is transmitted immediately after the SOP field. In receive mode, the 8 bits immediately following the SOP field are interpreted as the length of the packet. When this bit is cleared no packet length field is transmitted. 8DR always sends the packet length field (forces LEN EN = 1). GFSK requires user set LEN EN = 1.							
Bits 4:0	SOP Correlator Threshold. This is the receive data correlator threshold used when attempting to detect a SOP symbol. There is a threshold for the SOP_CODE_ADR code. This (single) threshold is applied independently to each of SOP1 and SOP2 fields. There are then two thresholds for each of the 64-chip DATA_CODE_ADR codes and 32 chip DATA_CODE_ADR codes. When SOP LEN is set, all 5 bits of this field are used. When SOP LEN is cleared, the most significant bit is disregarded. Typical applications configure SOP TH = 04h for SOP32 and SOP TH = 0Eh for SOP64.							

Mnemonic	DATA32_THOLD_ADR				Address			0x11
Bit	7	6	5	4	3	2	1	0
Default	-	-	-	-	0	1	0	0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
Function	Not Used	Not Used	Not Used	Not Used	TH32			
Bits 3:0: 32 Chip Data PN Code Correlator Threshold. This register sets the correlator threshold used in DSSS modes when DATA CODE LENGTH (see TX_CFG_ADR) is set to 32. Typical applications configure TH32 = 05h.								

Mnemonic	DATA64_THOLD_ADR				Address			0x12
Bit	7	6	5	4	3	2	1	0
Default	-	-	-	0	1	0	1	0
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
Function	Not Used	Not Used	Not Used	TH64				
Bits 4:0: 64 Chip Data PN Code Correlator Threshold. This register sets the correlator threshold used in DSSS modes when the DATA CODE LENGTH (see TX_CFG_ADR) is set to 64. Typical applications configure TH64 = 0Eh.								

Mnemonic	RSSI_ADR				Address			0x13
Bit	7	6	5	4	3	2	1	0
Default	0	-	1	0	0	0	0	0
Read/Write	R	-	R	R	R	R	R	R
Function	SOP	Not Used	LNA	RSSI				
<p>A Received Signal Strength Indicator (RSSI) reading is taken automatically when an SOP symbol is detected. In addition, an RSSI reading is taken whenever RSSI_ADR is read. The contents of this register are not valid after the device is configured for receive mode until either a SOP symbol is detected, or the register is read. The conversion can occur as often as once every 12 μs.</p> <p>If it is desired to measure the background RF signal strength on a channel before a packet has been received then the MCU should perform a "dummy" read of this register, the results of which should be discarded. This "dummy" read will cause an RSSI measurement to be taken, and therefore subsequent readings of the register will yield valid data.</p> <p>Bit 7: SOP RSSI Reading. When set, this bit indicates that the reading in the RSSI field was taken when a SOP symbol was detected. When cleared, this bit indicates that the reading stored in the RSSI field was triggered by a previous SPI read of this register.</p> <p>Bit 5: LNA State. This bit indicates the LNA state when the RSSI reading was taken. When cleared, this bit indicates that the LNA was disabled when the RSSI reading was taken; if set this bit indicates that the LNA was enabled when the RSSI reading was taken.</p> <p>Bits 4:0: RSSI Reading. This field indicates the instantaneous strength of the RF signal being received at the time that the RSSI reading was taken. A larger value indicates a stronger signal. The signal strength measured is for the RF signal on the configured channel, and is measured after the LNA stage.</p>								

Mnemonic	EOP_CTRL_ADR							Address	0x14
Bit	7	6	5	4	3	2	1	0	
Default	1	0	1	0	0	1	0	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Function	HEN		HINT			EOP			

If the LEN EN bit is set, then the contents of this register have no effect. If the LEN EN bit is cleared, then this register is used to configure how an EOP (end of packet) condition is detected.

Bit 7      EOP Hint Enable. When set, this bit will cause an EOP to be detected if no correlations have been detected for the number of symbol periods set by the HINT field and the last two received bytes match the calculated CRC16 for all previously received bytes. Use of this mode reduces the chance of non-correlations in the middle of a packet from being detected as an EOP condition.

Bits 6:4      EOP Hint Symbol Count. The minimum number of symbols of consecutive non-correlations at which the last two bytes are checked against the calculated CRC16 to detect an EOP condition.

Bits 4:0      EOP Symbol Count. An EOP condition is deemed to exist when the number of consecutive non-correlations is detected.

Mnemonic	CRC_SEED_LSB_ADR							Address	0x15
Bit	7	6	5	4	3	2	1	0	
Default	0	0	0	0	0	0	0	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Function	CRC SEED LSB								

The CRC16 seed allows different devices to generate or recognize different CRC16s for the same payload data. If a transmitter and receiver use a randomly selected CRC16 seed, the probability of correctly receiving data intended for a different receiver is 1/65535, even if the other transmitter/receiver are using the same SOP\_CODE\_ADR codes and channel.

Bits 7:0      CRC16 Seed Least Significant Byte. The LSB of the starting value of the CRC16 calculation.

Mnemonic	CRC_SEED_MSB_ADR							Address	0x16
Bit	7	6	5	4	3	2	1	0	
Default	0	0	0	0	0	0	0	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Function	CRC SEED MSB								

Bits 7:0      CRC16 Seed Most Significant Byte. The MSB of the starting value of the CRC16 calculation.

Mnemonic	TX_CRC_LSB_ADR							Address	0x17
Bit	7	6	5	4	3	2	1	0	
Default	-	-	-	-	-	-	-	-	
Read/Write	R	R	R	R	R	R	R	R	
Function	TX CRC LSB								

Bits 7:0      Calculated CRC16 LSB. The LSB of the CRC16 that was calculated for the last transmitted packet. This value is only valid after packet transmission is complete.

Mnemonic	TX_CRC_MSB_ADR							Address	0x18
Bit	7	6	5	4	3	2	1	0	
Default	-	-	-	-	-	-	-	-	
Read/Write	R	R	R	R	R	R	R	R	
Function	TX CRC MSB								

Bits 7:0      Calculated CRC16 MSB. The MSB of the CRC16 that was calculated for the last transmitted packet. This value is only valid after packet transmission is complete.

Mnemonic	RX_CRC_LSB_ADR							Address	0x19
Bit	7	6	5	4	3	2	1	0	
Default	1	1	1	1	1	1	1	1	
Read/Write	R	R	R	R	R	R	R	R	
Function	RX CRC LSB								
Bits 7:0	Received CRC16 LSB. The LSB of the CRC16 field from the last received packet. This value is valid whether or not the CRC16 field matched the calculated CRC16 of the received packet.								

Mnemonic	RX_CRC_MSB_ADR							Address	0x1A
Bit	7	6	5	4	3	2	1	0	
Default	1	1	1	1	1	1	1	1	
Read/Write	R	R	R	R	R	R	R	R	
Function	RX CRC MSB								
Bits 7:0	Received CRC16 MSB. The MSB of the CRC16 field from the last received packet. This value is valid whether or not the CRC16 field matched the calculated CRC16 of the received packet.								

Mnemonic	TX_OFFSET_LSB_ADR							Address	0x1B
Bit	7	6	5	4	3	2	1	0	
Default	0	0	0	0	0	0	0	0	
Read/Write	R/W	R	R	R	R	R	R	R	
Function	STRIM LSB								
Bits 7:0	<p>The least significant 8 bits of the synthesizer offset value. This is a 12-bit 2's complement signed number which may be used to offset the transmit frequency of the device by up to <math>\pm 1.5</math> MHz. A positive value increases the transmit frequency, and a negative value reduces the transmit frequency. A value of +1 increases the transmit frequency by 732.6 Hz; a value of -1 decreases the transmit frequency by 732.6 Hz. A value of 0x0555 increases the transmit frequency by 1 MHz; a value of 0xAAB decreases the transmit frequency by 1 MHz. Typically, this register is loaded with 0x55 during initialization. Typically this feature is used to avoid the need to change the synthesizer frequency when switching between TX and RX. As the IF = 1 MHz the RX frequency is offset 1 MHz from the synthesizer frequency; therefore, transmitting with a 1 MHz offset allows the same synthesizer frequency to be used for both transmit and receive.</p> <p>Synthesizer offset has no effect on receive frequency.</p>								

Mnemonic	TX_OFFSET_MSB_ADR				Address				0x1C
Bit	7	6	5	4	3	2	1	0	
Default	-	-	-	-	0	0	0	0	
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W	
Function	Not Used	Not Used	Not Used	Not Used	STRIM MSB				
Bits 3:0	The most significant 4 bits of the synthesizer trim value. Typically, this register is loaded with 0x05 during initialization.								

Mnemonic	MODE_OVERRIDE_ADR						Address	0x1D
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	-	-	0
Read/Write	W	W	W	W	W	-	-	W
Function	RSVD	RSVD	FRC SEN	FRC AWAKE		Not Used	Not Used	RST
Bits 7	Reserved. Do not write a 1 to these bits.							
Bits 5	Manually Initiate Synthesizer. Setting this bit forces the synthesizer to start. Clearing this bit has no effect. For this bit to operate correctly, the oscillator must be running before this bit is set.							
Bits 4:3	Force Awake. Force the device out of sleep mode. Setting both bits of this field forces the oscillator to keep running at all times regardless of the END STATE setting. Clearing both of these bits disables this function.							
Bits 0	Reset. Setting this bit forces a full reset of the device. Clearing this bit has no effect.							

Mnemonic	RX_OVERRIDE_ADR						Address	0x1E
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
Function	ACK RX	RXTX DLY	MAN RXACK	FRC RXDR	DIS CRC0	DIS RXCRC	ACE	Not Used
This register provides the ability to override some automatic features of the device.								
Bits 7	When this bit is set, the device uses the transmit synthesizer frequency rather than the receive synthesizer frequency for the given channel when automatically entering receive mode.							
Bits 6	When this bit is set and ACK EN is enabled, the transmission of the ACK packet is delayed by 20 μs.							
Bits 5	Force Expected Packet Type. When this bit is set, and the device is in receive mode, the device is configured to receive an ACK packet at the data rate defined in TX_CFG_ADR.							
Bits 4	Force Receive Data Rate. When this bit is set, the receiver will ignore the data rate encoded in the SOP symbol, and will receive data at the data rate defined in TX_CFG_ADR.							
Bits 3	Reject packets with a zero-seed CRC16. Setting this bit causes the receiver to reject packets with a zero-seed, and accept only packets with a CRC16 that matches the seed in CRC_SEED_LSB_ADR and CRC_SEED_MSB_ADR.							
Bits 2	The RX CRC16 checker is disabled. If packets with CRC16 enabled are received, the CRC16 will be treated as payload data and stored in the receive buffer.							
Bits 1	Accept Bad CRC16. Setting this bit causes the receiver to accept packets with a CRC16 that do not match the seed in CRC_SEED_LSB_ADR and CRC_SEED_MSB_ADR. An ACK is to be sent regardless of the condition of the received CRC16.							

Mnemonic	TX_OVERRIDE_ADR						Address	0x1F
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	ACK TX	FRC PRE	RSVD	MAN TXACK	OVRD ACK	DIS TXCRC	RSVD	TX INV
This register provides the ability to override some automatic features of the device.								
Bits 7	When this bit is set, the device uses the receive synthesizer frequency rather than the transmit synthesizer frequency for the given channel when automatically entering transmit mode.							
Bits 6	Force Preamble. When this bit is set, the device will transmit a continuous repetition of the preamble pattern (see PREAMBLE_ADR) after TX GO is set. This mode is useful for some regulatory approval procedures.							
Bits 5	Reserved. Do not write a 1 to this bit.							
Bits 4	Transmit ACK Packet. When this bit is set, the device sends an ACK packet when TX GO is set.							
Bits 3	ACK Override. Use TX_CFG_ADR to determine the data rate and the CRC16 used when transmitting an ACK packet.							
Bits 2	Disable Transmit CRC16. When set, no CRC16 field is present at the end of transmitted packets.							
Bits 1	Reserved. Do not write a 1 to this bit.							
Bits 0	TX Data Invert. When this bit is set the transmit bitstream is inverted.							



Mnemonic	CLK_OFFSET_ADR						Address	0x27
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Function	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RXF	RSVD

This register provides the ability to override some automatic features of the device.

Bits 7:2 Reserved. Do not write a 1 to these bits.

Bits 1 Force Receive Clock

Bits 0 Reserved. Do not write a 1 to this bit.

Mnemonic	CLK_EN_ADR						Address	0x28
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Function	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RXF	RSVD

This register provides the ability to override some automatic features of the device.

Bits 7:2 Reserved. Do not write a 1 to these bits.

Bits 1 Force Receive Clock Enable. Typical application will set this bit during initialization.

Bits 0 Reserved. Do not write a 1 to this bit.

Mnemonic	RX_ABORT_ADR						Address	0x29
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Function	RSVD	RSVD	ABORT EN	RSVD	RSVD	RSVD	RSVD	RSVD

This register provides the ability to override some automatic features of the device.

Bits 7:6 Reserved. Do not write a 1 to these bits.

Bits 5 Receive Abort Enable.

Bits 4:0 Reserved. Do not write a 1 to these bits.

Mnemonic	AUTO_CAL_TIME_ADR						Address	0x32
Bit	7	6	5	4	3	2	1	0
Default	0	0	0	0	0	0	1	1
Read/Write	W	W	W	W	W	W	W	W
Function	AUTO_CAL_TIME_MAX							

This register provides the ability to override some automatic features of the device.

Bits 7:0 Auto Cal Time Max. Firmware must write 3Ch to this register during initialization.

Mnemonic	AUTO_CAL_OFFSET_ADR							Address	0x35
Bit	7	6	5	4	3	2	1	0	
Default	0	0	0	0	0	0	0	0	
Read/Write	W	W	W	W	W	W	W	W	
Function	AUTO_CAL_OFFSET_MINUS_4								
This register provides the ability to override some automatic features of the device.									
Bits 7:0 Auto Cal Time Max. Firmware must write 14h to this register during initialization.									

Mnemonic	ANALOG_CTRL_ADR							Address	0x39
Bit	7	6	5	4	3	2	1	0	
Default	0	0	0	0	0	0	0	0	
Read/Write	W	W	W	W	W	W	W	W	
Function	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	ALL SLOW	
This register provides the ability to override some automatic features of the device.									
Bits 7:1 Reserved. Do not write a 1 to these bits.									
Bits 0 All Slow. When set, the synth settling time for all channels is the same as for slow channels. It is recommended that firmware set this bit when using GFSK data rate mode.									

## Register Files

Files are written to or read from using nonincrementing burst read or write transactions. In most cases reading a file may be destructive; the file must be completely read, otherwise the contents may be altered.

Mnemonic	TX_BUFFER_ADR	Address	0x20
Length	16 Bytes	R/W	W
Default	0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF		

The transmit buffer is a FIFO. Writing to this file adds a byte to the packet being sent. Writing more bytes to this file than the packet length in TX\_LENGTH\_ADR will have no effect, and these bytes will be lost after successful packet transmission. It is **NOT** possible to load two 8-byte packets into this register, and then transmit them sequentially by enabling the TX GO bit twice; this would have the effect of sending the first eight bytes twice.

Mnemonic	RX_BUFFER_ADR	Address	0x21
Length	16 Bytes	R/W	R
Default	0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF		

The receive buffer is a FIFO. Received bytes may be read from this file register at any time that it is not empty, but when reading from this file register before a packet has been completely received care must be taken to ensure that error packets (for example with bad CRC16) are handled correctly.

When the receive buffer is configured to be overwritten by new packets (the alternative is for new packets to be discarded if the receive buffer is not empty), similar care must be taken to verify after the packet has been read from the buffer that no part of it was overwritten by a newly received packet while this file register is being read.

When the VLD EN bit in RX\_CFG\_ADR is set, the bytes in this file register alternate—the first byte read is data, the second byte is a valid flags for each bit in the first byte, the third byte is data, the fourth byte valid flags, etc. In SDR and DDR modes the valid flag for a bit is set if the correlation coefficient for the bit exceeded the correlator threshold, and is cleared if it did not. In 8DR mode, the MSB of a valid flags byte indicates whether or not the correlation coefficient of the corresponding received symbol exceeded the threshold. The seven LSBs contain the number of erroneous chips received for the data.

Mnemonic	SOP_CODE_ADR	Address	0x22
Length	8 Bytes	R/W	R/W
Default	0x17FF9E213690C782		

When using 32-chip SOP\_CODE\_ADR codes, only the first four bytes of this register are used; in order to complete the file write process, these four bytes must be followed by four bytes of “dummy” data. However, a class of codes known as “multiplicative codes” may be used; there are 64-chip codes with good auto-correlation and cross-correlation properties where the least significant 32 chips themselves have good auto-correlation and cross-correlation properties when used as 32-chip codes. In this case the same eight-byte value may be loaded into this file and used for both 32-chip and 64-chip SOP symbols.

When reading this file, all eight bytes must be read; if fewer than eight bytes are read from the file, the contents of the file will have been rotated by the number of bytes read. This applies to writes, as well.

Recommended SOP Codes:

```
0x91CCF8E291CC373C
0x0FA239AD0FA1C59B
0x2AB18FD22AB064EF
0x507C26DD507CCD66
0x44F616AD44F6E15C
0x46AE31B646AEC5A
0x3CDC829E3CDC78A1
0x7418656F74198EB9
0x49C1DF6249C0B1DF
0x72141A7F7214E597
```

Do not access or modify the register during a transmit or receive.

Mnemonic	DATA_CODE_ADR	Address	0x23
Length	16 Bytes	R/W	R/W
Default	0x02F9939702FA5CE3012BF1DB0132BE6F		

This file is ignored when using the device in 1-Mbps GFSK mode. In 64-SDR mode, only the first eight bytes are used; in order to complete the file write process, these eight bytes must be followed by eight bytes of “dummy” data. In 32-SDR mode, only four bytes are used, and in 32-DDR mode only eight bytes are used. In 64-DDR and 8DR modes, all sixteen bytes are used. Certain sixteen-byte sequences have been calculated that provide excellent auto-correlation and cross-correlation properties, and it is recommended that such sequences be used; the default value of this register is one such sequence. In typical applications, all devices use the same DATA\_CODE\_ADR codes, and devices and systems are addressed by using different SOP\_CODE\_ADR codes; in such cases it may never be necessary to change the contents of this register from the default value.

When reading this file, all sixteen bytes must be read; if fewer than sixteen bytes are read from the file, the contents of the file will have been rotated by the number of bytes read. This applies to writes, as well.

Typical applications should use the default code.  
Do not access or modify the register during a transmit or receive.

Mnemonic	PREAMBLE_ADR	Address	0x24
Length	3 Bytes	R/W	R/W
Default	0x333302		

Byte 1 – The number of repetitions of the preamble sequence that are to be transmitted. The preamble may be disabled by writing 0x00 to this byte.

Byte 2 – Least significant eight chips of the preamble sequence

Byte 3– Most significant eight chips of the preamble sequence

If using 64-SDR to communicate with CYWUSB69xx devices, set number of repetitions to four for optimum performance

When reading this file, all three bytes must be read; if fewer than three bytes are read from the file, the contents of the file will have been rotated by the number of bytes read. This applies to writes, as well.

Do not access or modify this register during Transmit or Receive.

Mnemonic	MFG_ID_ADR	Address	0x25
Length	6 Bytes	R	R
Default	NA		

Byte 1 – 4 bits version + 2 bits vendor ID + high 2 bits of Year

Byte 2 through Byte 6: Manufacturing ID for the device.

To minimize ~190  $\mu$ A of current consumption (default), execute a ‘dummy’ single-byte SPI write to this address with a zero data stage after the contents have been read. Non-zero to enable reading of fuses. Zero to disable reading fuses.

**Absolute Maximum Ratings**

Storage Temperature ..... –65°C to +150°C  
 Ambient Temperature with Power Applied ..... 0°C to +70°C  
 Supply Voltage on any power supply pin relative to V<sub>SS</sub> –0.3V to +3.9V  
 DC Voltage to Logic Inputs<sup>[8]</sup> ..... –0.3V to V<sub>IO</sub> +0.3V  
 DC Voltage applied to Outputs in High-Z State .. –0.3V to V<sub>IO</sub> +0.3V

Static Discharge Voltage (Digital)<sup>[9]</sup> ..... >2000V  
 Static Discharge Voltage (RF)<sup>[9]</sup> ..... 1100V  
 Latch-up Current..... +200 mA, –200 mA  
 Ground Voltage..... 0V  
 F<sub>OSC</sub> (Crystal Frequency) ..... 12 MHz  $\pm$ 30 ppm

**DC Characteristics** (T = 25°C)

Parameter	Description	Conditions	Min.	Typ.	Max.	Unit
V <sub>BAT</sub>	Battery Voltage	0–70°C	1.8		3.6	V
V <sub>REG</sub> <sup>[10]</sup>	PMU Output Voltage	2.7V mode	2.7	2.73		V
V <sub>LVD</sub>	Low Voltage Detect	LVDCR [2:0] set to 000				
V <sub>IO</sub>	V <sub>IO</sub> Voltage		1.8		3.6	V
V <sub>CC</sub>	V <sub>CC</sub> Voltage	0–70°C	2.4		3.6	V
<b>Device Current</b> (For total current consumption in different modes, for example Radio, active, MCU, sleep, etc., add Radio Function Current and MCU Function Current)						
I <sub>CC</sub> (GFSK) <sup>[11]</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>[11]</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>SB1</sub>	Sleep Mode I <sub>CC</sub>	V <sub>CC</sub> = 3.0V, MCU sleep, PMU disabled		2.72		μA
I <sub>SB2</sub>	Sleep Mode I <sub>CC</sub>	V <sub>CC</sub> = 3.0V, MCU sleep, PMU enabled		30.4		μA

**Notes**

- 8. 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.
- 9. Human Body Model (HBM).
- 10. V<sub>REG</sub> depends on battery input voltage.
- 11. 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)

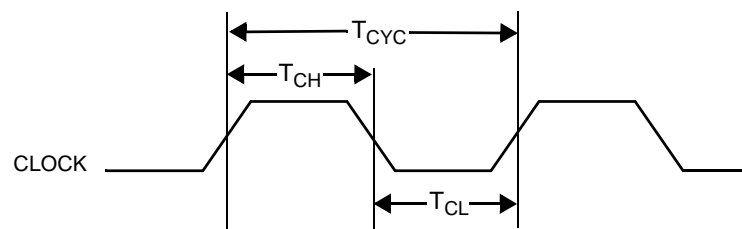
Parameter	Description	Conditions	Min.	Typ.	Max.	Unit
<b>Radio Function Currents</b> ( $V_{CC} = 3.0V$ , MCU Sleep)						
IDLE $I_{CC}$	Radio off, XTAL Active	XOUT disabled		1.1		mA
$I_{synth}$	$I_{CC}$ during Synth Start			8.6		mA
TX $I_{CC}$	$I_{CC}$ during Transmit	PA = 5 (-5 dBm)		21.2		mA
TX $I_{CC}$	$I_{CC}$ during Transmit	PA = 6 (0 dBm)		28.5		mA
TX $I_{CC}$	$I_{CC}$ during Transmit	PA = 7 (+4 dBm)		39.9		mA
RX $I_{CC}$	$I_{CC}$ during Receive	LNA off, ATT on.		18.9		mA
RX $I_{CC}$	$I_{CC}$ during Receive	LNA on, ATT off.		21.9		mA
Boost Eff	PMU Boost Converter Efficiency	$V_{BAT} = 2.5V$ , $V_{REG} = 2.73V$ , $I_{LOAD} = 20$ mA		83		%
$I_{LOAD\_EXT}$	Average PMU External Load current	$V_{BAT} = 1.8V$ , $V_{REG} = 2.73V$ , RX Mode			15	mA
<b>MCU Function Currents</b> ( $V_{DD} = 3.0V$ )						
$I_{DD1}$	$V_{DD}$ Operating Supply Current	CPU speed = 6 MHz		5.0		mA
$I_{DD1}$	$V_{DD}$ Operating Supply Current	CPU speed = 3 MHz		4.4		mA
<b>Radio Function GPIO Interface</b>						
$V_{OH1}$	Output High Voltage Condition 1	At $I_{OH} = -100.0$ $\mu A$	$V_{IO} - 0.1$	$V_{IO}$		V
$V_{OH2}$	Output High Voltage Condition 2	At $I_{OH} = -2.0$ mA	$V_{IO} - 0.4$	$V_{IO}$		V
$V_{OL}$	Output Low Voltage	At $I_{OL} = 2.0$ mA		0	0.4	V
$V_{IH}$	Input High Voltage		$0.76V_{IO}$		$V_{IO}$	V
$V_{IL}$	Input Low Voltage		0		$0.24V_{IO}$	V
$I_{IL}$	Input Leakage Current	$0 < V_{IN} < V_{IO}$	-1	0.26	+1	$\mu A$
$C_{IN}$	Pin Input Capacitance	except XTAL, $RF_N$ , $RF_P$ , $RF_{BIAS}$		3.5	10	pF
<b>MCU Function GPIO Interface</b>						
$R_{UP}$	Pull-up Resistance		4		12	K $\Omega$
$V_{ICR}$	Input Threshold Voltage Low, CMOS mode	Low to High edge	40%		65%	$V_{CC}$
$V_{ICF}$	Input Threshold Voltage Low, CMOS mode	High to Low edge	30%		55%	$V_{CC}$
$V_{HC}$	Input Hysteresis Voltage, CMOS Mode	High to low edge	3%		10%	$V_{CC}$
$V_{ILTTL}$	Input Low Voltage, TTL Mode				0.72	V
$V_{IHTTL}$	Input HIGH Voltage, TTL Mode		1.6			V
$V_{OL1}$	Output Low Voltage, High Drive <sup>[12]</sup>	$I_{OL1} = 50$ mA			1.4	V
$V_{OL2}$	Output Low Voltage, High Drive <sup>[12]</sup>	$I_{OL1} = 25$ mA			0.4	V
$V_{OL3}$	Output Low Voltage, Low Drive	$I_{OL2} = 8$ mA			0.8	V
$V_{OH}$	Output High Voltage <sup>[13]</sup>	$I_{OH} = 2$ mA	$V_{CC} - 0.5$			V

**Notes**

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

**AC Characteristics**

Parameter	Description	Conditions	Min.	Typical	Max.	Unit
<b>GPIO Timing</b>						
$T_{R\_GPIO}$	Output Rise Time	Measured between 10 and 90% Vdd/Vreg with 50 pF load			50	ns
$T_{F\_GPIO}$	Output Fall Time	Measured between 10 and 90% Vdd/Vreg with 50 pF load			15	ns
<b>SPI Timing</b>						
$T_{SMCK}$	SPI Master Clock Rate	$F_{CPUCLK}/6$			2	MHz
$T_{SSCK}$	SPI Slave Clock Rate				2.2	MHz
$T_{SCKH}$	SPI Clock High Time	High for CPOL = 0, Low for CPOL = 1	125			ns
$T_{SCKL}$	SPI Clock Low Time	Low for CPOL = 0, High for CPOL = 1	125			ns
$T_{MDO}$	Master Data Output Time <sup>[14]</sup>	SCK to data valid	-25		50	ns
$T_{MDO1}$	Master Data Output Time, First bit with CPHA = 0	Time before leading SCK edge	100			ns
$T_{MSU}$	Master Input Data Setup time		50			ns
$T_{MHD}$	Master Input Data Hold time		50			ns
$T_{SSU}$	Slave Input Data Setup Time		50			ns
$T_{SHD}$	Slave Input Data Hold Time		50			ns
$T_{SDO}$	Slave Data Output Time	SCK to data valid			100	ns
$T_{SDO1}$	Slave Data Output Time, First bit with CPHA = 0	Time after SS LOW to data valid			100	ns
$T_{SSS}$	Slave Select Setup Time	Before first SCK edge	150			ns
$T_{SSH}$	Slave Select Hold Time	After last SCK edge	150			ns

**Figure 19. Clock Timing**

**Note**

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



Figure 20. GPIO Timing Diagram

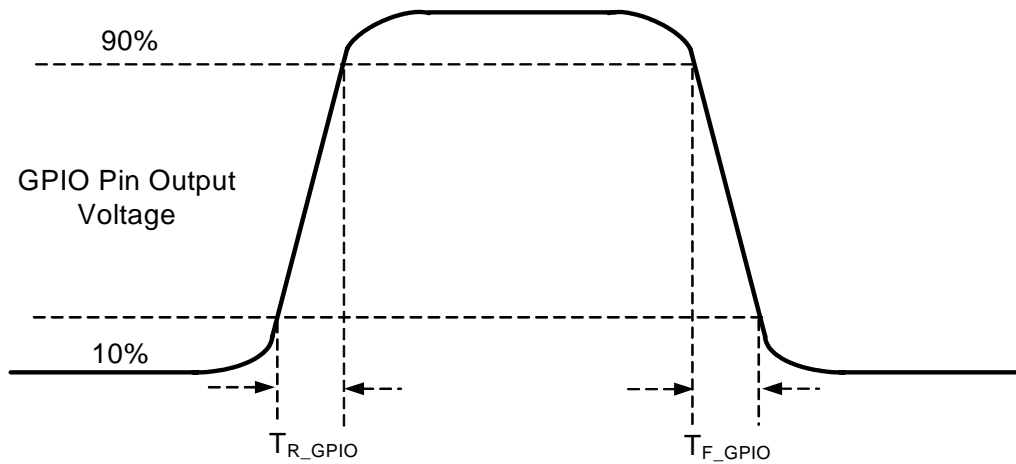


Figure 21. SPI Master Timing, CPHA = 1

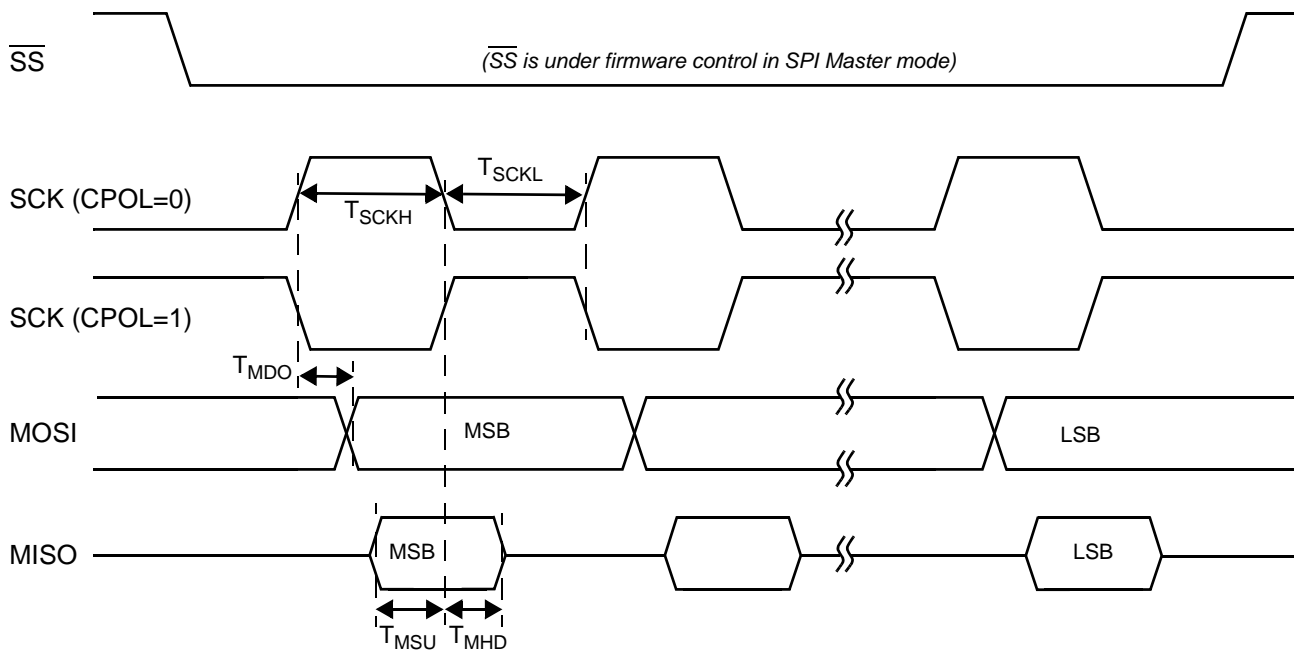


Figure 22. SPI Slave Timing, CPHA = 1

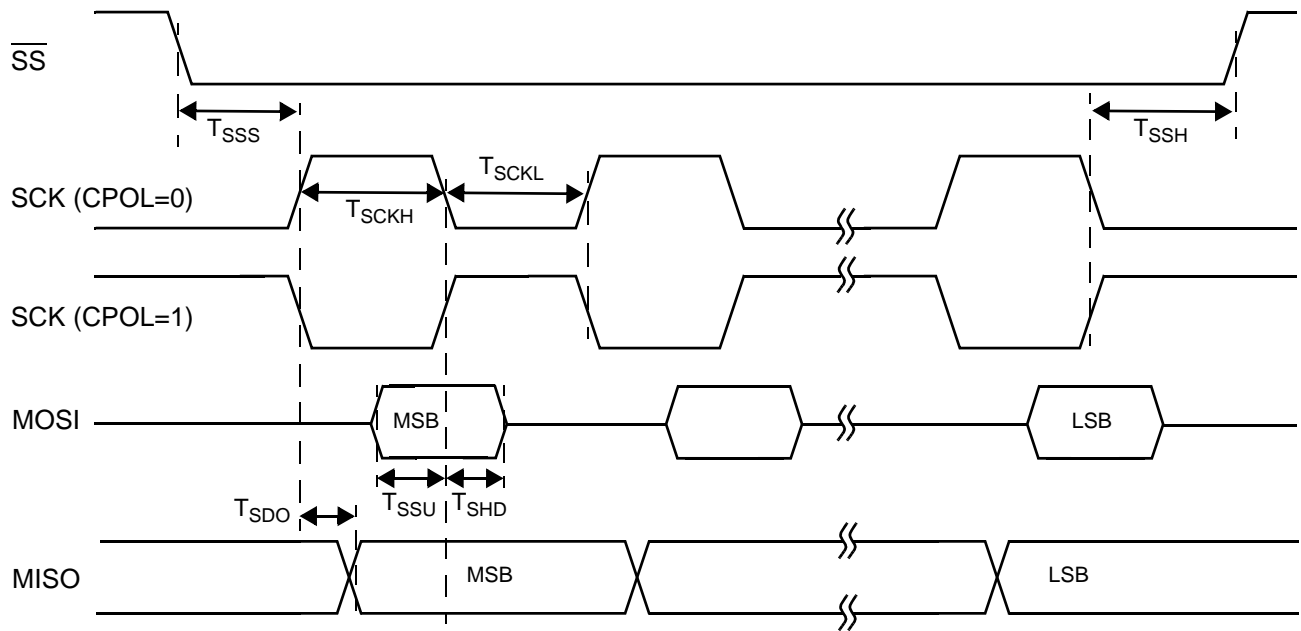


Figure 23. SPI Master Timing, CPHA = 0

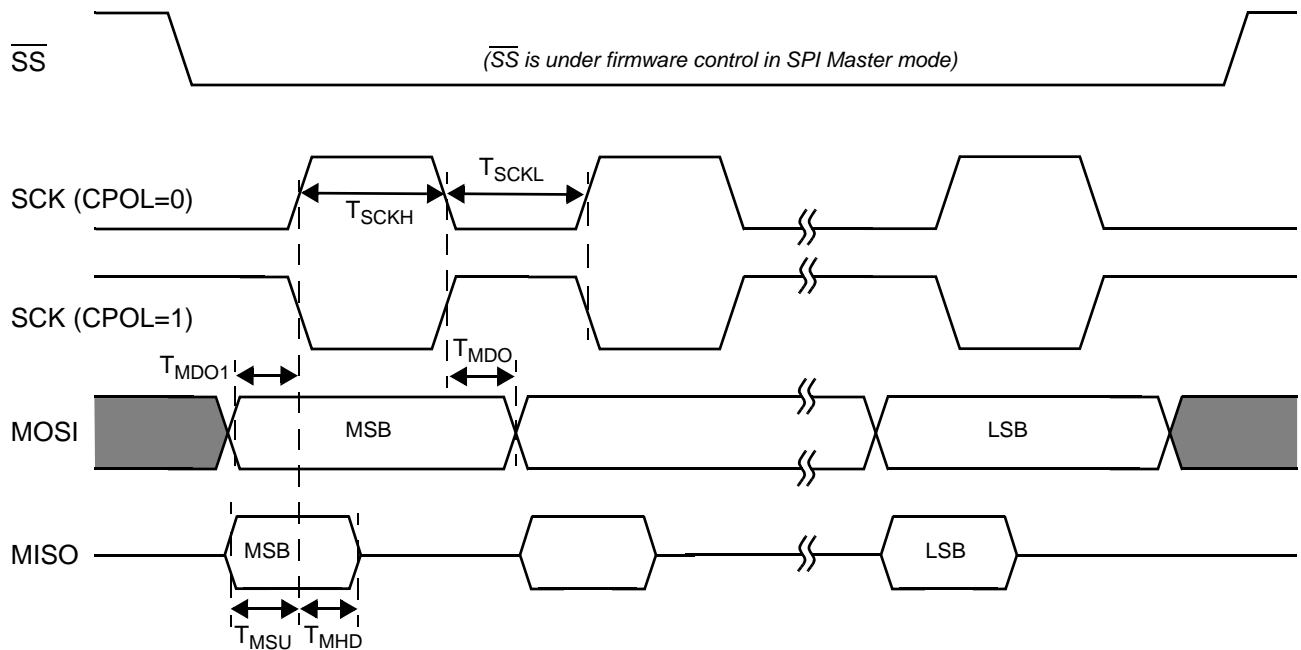
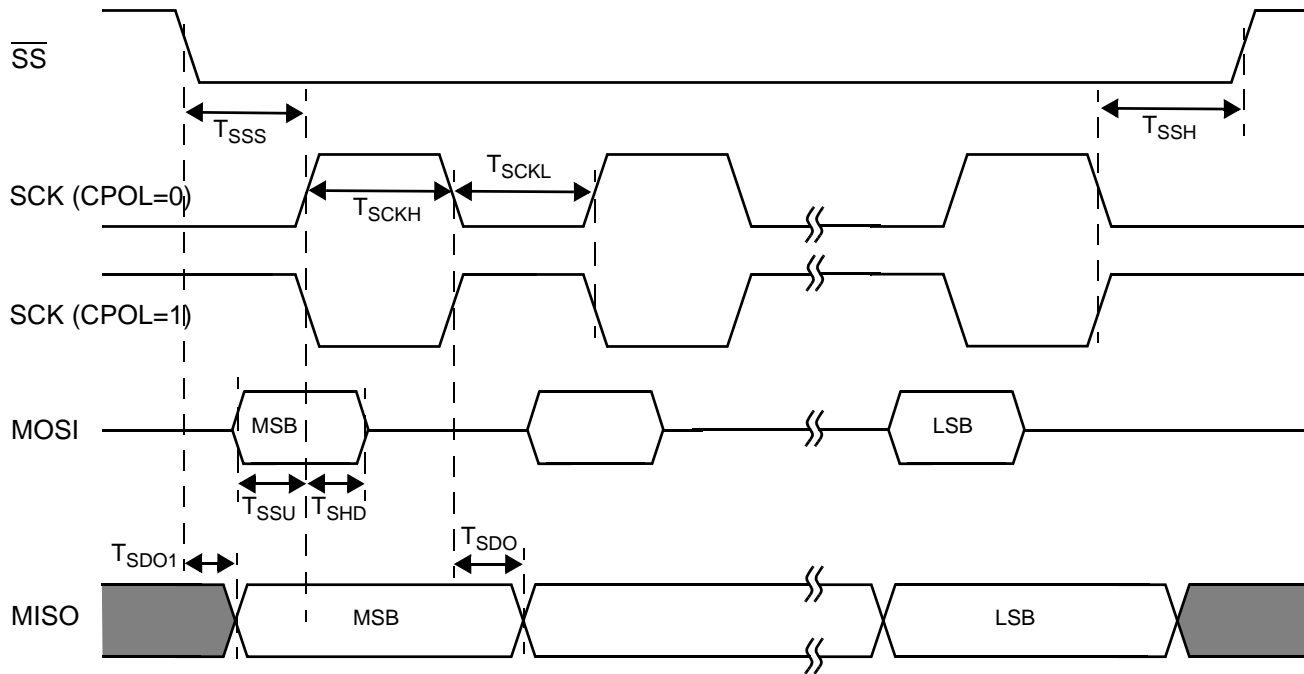


Figure 24. SPI Slave Timing, CPHA = 0



**RF Characteristics**
**Table 74. 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> = 3.0V, f <sub>OSC</sub> = 12.000 MHz, BER < 10 <sup>-3</sup> )					
Sensitivity 125kbps 64-8DR	BER 1E-3		-97		dBm
Sensitivity 250-kbps 32-8DR	BER 1E-3		-93		dBm
Sensitivity	CER 1E-3	-80	-87		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 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>[15]</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> = 3.0V, f <sub>OSC</sub> = 12.000 MHz)					
Maximum RF Transmit Power	PA = 7	+2	4	+6	dBm
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			39		dB
RF Power Range Control Step Size	seven 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**

15. Exceptions F/3 &amp; 5C/3.

16. When using an external switching regulator to power the radio, care must be taken to keep the switching frequency well away from the IF frequency of 1MHz.

Table 74. Radio Parameters (continued)

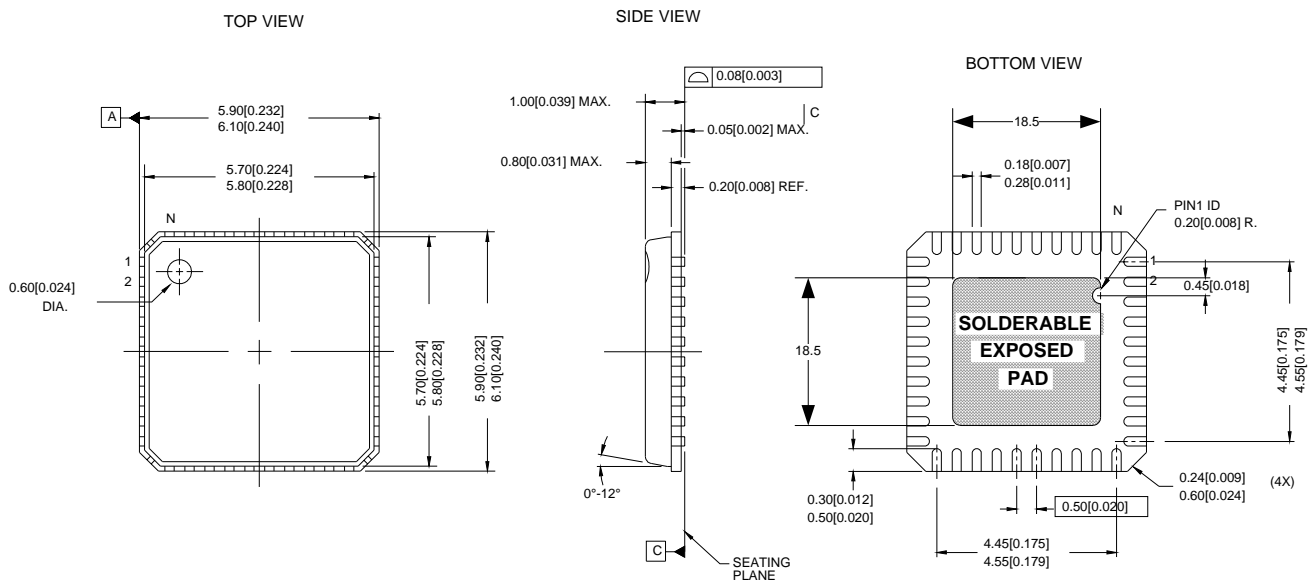
Parameter Description	Conditions	Min.	Typ.	Max.	Unit
<b>Transmit Spurious Emission</b> (PA = 7)					
In-band Spurious Second Channel Power ( $\pm 2$ MHz)			-38		dBm
In-band Spurious Third Channel Power ( $\geq 3$ MHz)			-44		dBm
Non-Harmonically Related Spurs (8.000 GHz)			-38		dBm
Non-Harmonically Related 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
Link Turnaround Time	125 kbps			94	$\mu$ s
Link Turnaround Time	<125 kbps			31	$\mu$ s
Max. packet length	< 60 ppm crystal to crystal, all modes except 64-DDR			40	bytes
Max. packet length	< 60 ppm crystal to crystal, 64-DDR			16	bytes

Table 75. Ordering Information


Package	Ordering Part Number
40-pin Lead-Free QFN 6x6 mm	CYRF69103-40LFXC

Package Diagram

Figure 25. 40-pin Lead-Free QFN 6x6 mm



NOTES:

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

51-85190-<sup>\*</sup>A

PSoC is a registered trademark and WirelessUSB, enCoRe, and PRoC are trademarks of Cypress Semiconductor Corporation. All products and company names mentioned in this document may be the trademarks of their respective holders.

**Document History Page**

Document Title: CYRF69103 Programmable Radio on Chip Low Power Document #: 001-07611				
REV.	ECN No.	Issue Date	Orig. of Change	Description of Change
**	479801	See ECN	OYR	New advance data sheet.
*A	501282	See ECN	OYR	Preliminary data sheet. Created Preliminary data sheet from Advance Information.
*B	631696	See ECN	BOO	Final data sheet. Updated DC Characteristics table with characterization data. Minor text changes GPIO capacitance and timing diagram included Sleep and Wakeup sequence documented PIT Timer registers' R/W capability corrected to read only Updated radio function register descriptions Changed L/D pin description Changed RST Capacitor from 0.1uF to 0.47uF Added example PMU configuration circuits