IP2022 Internet Processor™

Features and Performance Optimized for Network Connectivity

1.0 Product Overview

The Ubicom IP2022 Internet Processor™ combines support for communication physical layer, Internet protocol stack, device-specific application and device-specific peripheral software modules in a single chip, and is reconfigurable over the Internet. It can be programmed, and reprogrammed, using pre-built software modules and configuration tools to create true single-chip solutions for a wide range of device-to-device and device-to-human communication applications. Fabricated in an advanced 0.25-micron process, its RISC-based deterministic architecture provides high-speed computation, flexible I/O control, efficient data manipulation, in-system programming, and in-system debugging. A hardware full-duplex serializer/deserializer (SerDes) function gives the IP2022 the ability to directly connect to a variety of common network interfaces. This function provides the ability to implement on-chip 10Base-T Ethernet (MAC and PHY), USB, and a variety of other fast serial protocols. The inclusion of two SerDes units facilitate translation from one format to another, allowing the IP2022 to be used as a protocol converter. The 100 MHz operating frequency, with most instructions executing in a single cycle, delivers the throughput needed for emerging network connectivity applications, and a flash-based program memory allows both in-system and on-the-fly reprogramming. The IP2022 implements peripheral, communications and control functions as software modules (ipModules™), replacing traditional hardware for maximum system design flexibility. This approach allows rapid, inexpensive product design and, when needed, quick and easy reconfiguration to accommodate changes in market needs or industry standards.

On-chip dedicated hardware also includes a PLL, an 8-channel 10-bit ADC, general-purpose timers, single-cycle multiplier, analog comparator, brown-out power voltage detector, watchdog timer, low-power support, multi-source wakeup capability, user-selectable clock modes, high-current outputs, and 52 general-purpose I/O pins.

A TCP/IP network protocol stack is available, and a variety of additional software that is necessary to form a complete end-to-end connectivity solution is being developed. Tools for developing with and using the IP2022, including the complete Red Hat GNUPro tools, are available from leading suppliers.

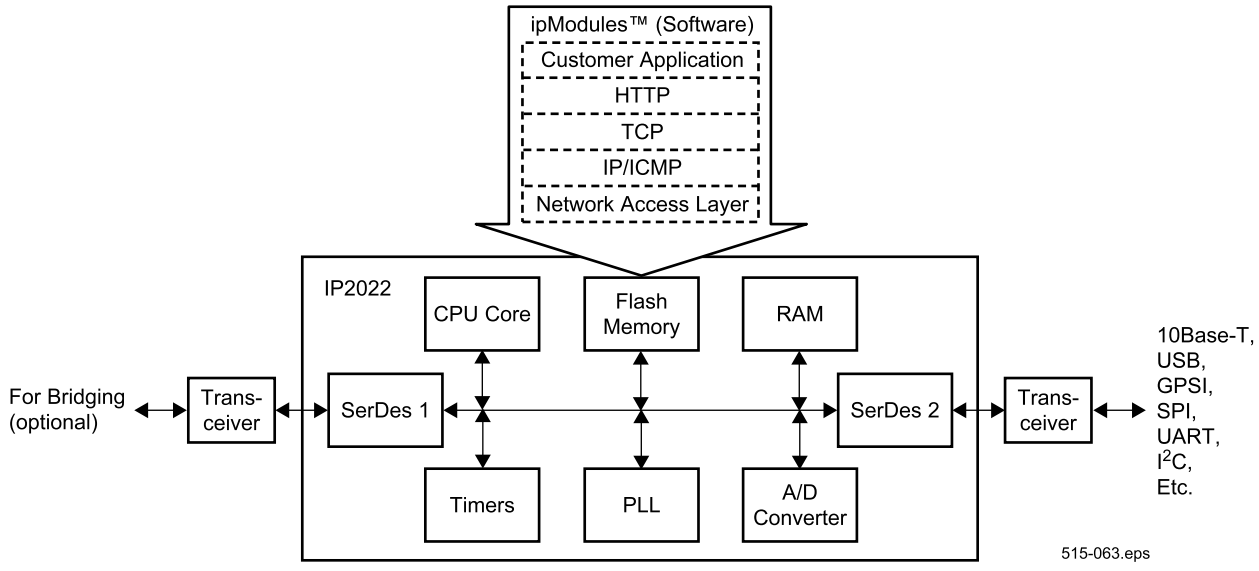


Figure 1-1 IP2022 Block Diagram

TABLE OF CONTENTS

1.1 Key Features

CPU Features

- RISC engine core with DC to 100 MHz operation
- 10 ns instruction cycle
- Compact 16-bit fixed-length instructions
- Single-cycle instruction execution on most instructions (3 cycles for jumps and calls)
- Sixteen-level hardware stack for high-performance subroutine linkage
- 8 × 8 signed/unsigned single-cycle multiply
- Pointers and stack operation optimized for C compiler
- Uniform, linear address space (no register banks)

On-Chip Memory

- 64 Kbyte (32K × 16) program flash memory
- 16 Kbyte (8K × 16) program/data RAM
- 4 Kbyte linear-addressed data RAM
- Self-programming with built-in charge pump: instructions to read, write, and erase flash memory

Fast and Deterministic Program Execution and Interrupt Response

- Predictable execution rate for real-time applications
- Fast and deterministic 3-cycle interrupt response
- 30 ns internal interrupt response at 100 MHz including context save
- Hardware save/restore of register context (PC, W, STATUS, MULH, SPDREG, IPH, IPL, DPH, DPL, SPH, SPL, ADDRH, ADDRL, DATAH, DATAL)

Multiple Networking Protocols and Physical Layer Support Hardware

- Two full-duplex serializer/deserializer (SerDes) channels for 10Base-T (MAC/PHY), USB, and other fast serial protocol support
  - Embedded connectivity nodes
  - Two channels for protocol bridging
  - I^2C, UART, SPI, Microwire/PLUS

General-Purpose Hardware Peripherals

- Two 16-bit timers with 8-bit prescalers supporting:
  - Timer mode
  - PWM mode
  - Capture/Compare mode
- Parallel host interface, 8/16-bit selectable for use as a communications coprocessor
- One 8-bit timer with programmable 15-bit prescaler
- One 8-bit real-time clock/counter with programmable 8-bit prescaler and 32 kHz crystal input
- Watchdog timer with prescaler
- On-chip PLL clock multiplier with pre- and post-divider
  - 2 MHz input produces 100 MHz internal operating frequency
- 10-bit, 8-channel ADC with 1/2 LSB accuracy
- Analog comparator with hysteresis enable/disable
- Brown-out minimum supply voltage detector
- External interrupt inputs on 8 pins (Port B)

Sophisticated Power and Frequency/Clock Management Support

- Operating voltage of 2.3V to 2.7V
- Switching the system clock frequencies between different clock sources
- Changing the core clock using a selectable divider
- Shutting down the PLL and/or the OSC input
- Dynamic CPU speed control with **speed** instruction
- Power-On-Reset (POR) logic

Flexible I/O

- 52 I/O Pins
- 2.3V to 3.3V symmetric CMOS output drive
- 5V-tolerant inputs
- Port A pins capable of sourcing/sinking 24 mA
- Optional I/O synchronization to CPU core clock

Programming and Debugging Support

- Updateable application program
  - Run-time self programming
- On-chip in-system programming interface
- On-chip in-system debugging support interface
- Debugging at full IP2022 operating speed
- Programming at device supply voltage level
- Real-time emulation, program debugging, and integrated software development environment offered by leading third-party tool vendors

Pre-Built Software Modules

- Selection of physical interfaces:
  - 10Base-T Ethernet
  - USB
  - UART
  - I^2C
  - SPI
  - Parallel host interface
- Complete TCP/IP stack implementation

Software Support

- Red Hat GNUPro tools
  - GCC ANSI C compiler and assembler, linker, utilities, GNU debugger, and IDE

- Nohau in-circuit debugger
  - Seehau Interface
  - USB-based debug hardware
  - Assembler
- Library of off-the-shelf ipModules (Ethernet, serial interfaces, USB interface, etc.)
- Evaluation kits for Internet and communication-intensive applications

1.2 Architecture

1.2.1 CPU

The IP2022 implements an enhanced Harvard architecture (i.e. separate instruction and data memories) with independent address and data buses. The 16-bit program memory and 8-bit dual-port data memory allow instruction fetch and data operations to occur in parallel. The advantage of this architecture is that instruction fetch and memory transfers can be overlapped by a multistage pipeline, so that the next instruction can be fetched from program memory while the current instruction is executed with data from the data memory.

Ubicom has developed a revolutionary RISC-based architecture that is deterministic, jitter free, and completely reprogrammable.

The IP2022 implements a four-stage pipeline (fetch, decode, execute, and write back). At the maximum operating frequency of 100 MHz, instructions are executed at the rate of one per 10 ns clock cycle.

1.2.2 Serializer/Deserializers

One of the key elements in optimizing the IP2022 for device-to-device and device-to-human communication is the inclusion of two on-chip serializer/deserializers. These units support popular communication protocols such as SPI, I^2C, UART, USB, and 10Base-T Ethernet, allowing the IP2022 to be used as a protocol converter in bridge and gateway applications.

By performing data serialization and deserialization in hardware, the CPU bandwidth needed to support serial communications is greatly reduced, especially at high baud rates. Providing two units allows easy implementation of protocol conversion or bridging functions, such as a USB-to-I^2C bridge.

1.2.3 Low-Power Support

Particular attention has been paid to minimizing power consumption. For example, an on-chip PLL allows use of a lower-frequency external source (e.g., an inexpensive 2 MHz crystal oscillator can be used to produce a 100 MHz internal operating frequency), which reduces both power consumption and EMI. In addition, software can change the execution speed of the CPU to reduce power consumption, and a mechanism is provided for automatically changing the speed on entry and return from an interrupt service routine. The **speed** instruction specifies power-saving modes that include a clock divisor between 1 and 128. This divisor only affects the clock to the CPU core, not the timers. The **speed** instruction also specifies the clock source (OSC1 clock, RTCLK oscillator, or PLL clock multiplier), and whether to disable the OSC1 clock oscillator or the PLL. The **speed** instruction executes using the current clock divisor.

1.2.4 Memory

The IP2022 CPU executes from a 32K × 16 flash program memory and an 8K × 16 RAM program/data memory. The instruction RAM can alternatively be used for data storage. In addition, the ability to write into the program flash memory allows flexible non-volatile RAM implementation. The maximum execution rate is 40 MIPS from flash memory and 100 MIPS from RAM. Speed-critical routines can be copied from the flash memory to the RAM for faster execution. The IP2022 has a mechanism for in-system programming of its flash and RAM program memories through a four-wire SPI interface, and software has the ability to reprogram the program memories at run time. This allows the functionality of a device to be changed in the field over the Internet.

1.2.5 Instruction Set

The IP2022 instruction set, using 16-bit words, implements a rich set of arithmetic and logical operations, including signed and unsigned 8-bit × 8-bit integer multiply with a 16-bit product.

1.2.6 The ipModule Concept

The ipModule concept enables the “software system-on-a-chip” approach. An ipModule is a software implementation of an interface, protocol, or other function that replaces traditional hardware. This takes advantage

of the Ubicom architecture's high performance and deterministic nature to produce the same results as hardware, but with much greater system design flexibility. Having functionality implemented as pre-built software modules allows the IP2022 to be programmed and reprogrammed at any time in the design and manufacturing cycle, and even in the field over the Internet.

The speed and flexibility of the Ubicom architecture, together with the availability of Internet connectivity software modules, simultaneously address a wide range of engineering and product development concerns. They decrease the product development cycle dramatically, shortening time to production to as little as a few weeks.

Ubicom's timesaving ipModules give system designers a choice of ready-made solutions, or a head start on developing their own peripherals. With ipModules handling established functions, design engineers can concentrate on adding value to other areas of the application.

Overall, the ipModule concept provides such benefits as simpler hardware architecture, reduced component count, fast time to market, increased flexibility in design, application customization, and overall system cost reduction.

Some examples of ipModules are:

- Ethernet and USB network interfaces
- Communication interfaces such as I^2C™, Microwire™, SPI, and UART
- Internet connectivity protocols, such as UDP, TCP/IP, ARP, DHCP, HTTP, SMTP, and POP3

1.2.7 Programming and Debugging Support

The IP2022 is supported by leading third-party tool vendors. On-chip in-system debug capabilities allow these tools to provide an integrated software development environment that includes editor, assembler, debugger, simulator, and programmer tools. For example, the complete Red Hat GNUPro tools, including C compiler, assembler, linker, utilities and GNU debugger, supports the IP2022. Likewise, the Seehau interface, high-end debugger, assembler, and USB debug hardware from Nohau can be used with the IP2022.

Unobtrusive in-system programming is provided through the ISP interface. There is no need for a bond-out chip for software development. This eliminates concerns about differences in electrical characteristics between a bond-out chip and the actual chip used in the target application. Designers can test and revise code on the same part used in the actual application.

1.2.8 Applications

The IP2022 Internet Processor™ is optimized for network connectivity applications, and is ideally suited for use in the node and bridge/gateway portions of the Internet infrastructure.

Node device applications are those that are commonly associated with the "embedded Internet," such as home appliances, medical devices, vending machines, and remote monitoring and control systems. These nodes are frequently interconnected by local-area networks (LANs). Bridge/gateway devices provide the functions that are required to connect the nodes, and their related LANs, to the Internet, such as protocol conversion, IP address routing, and firewall functions. The IP2022 enables true single-chip device and bridge/gateway connectivity implementations at a consumer price point. The library of ipModules, including the Internet protocol stack and communication interfaces, allows design engineers to embed Internet connectivity in all of their products at low cost with very fast time-to-market.

2.0 Pin Assignments

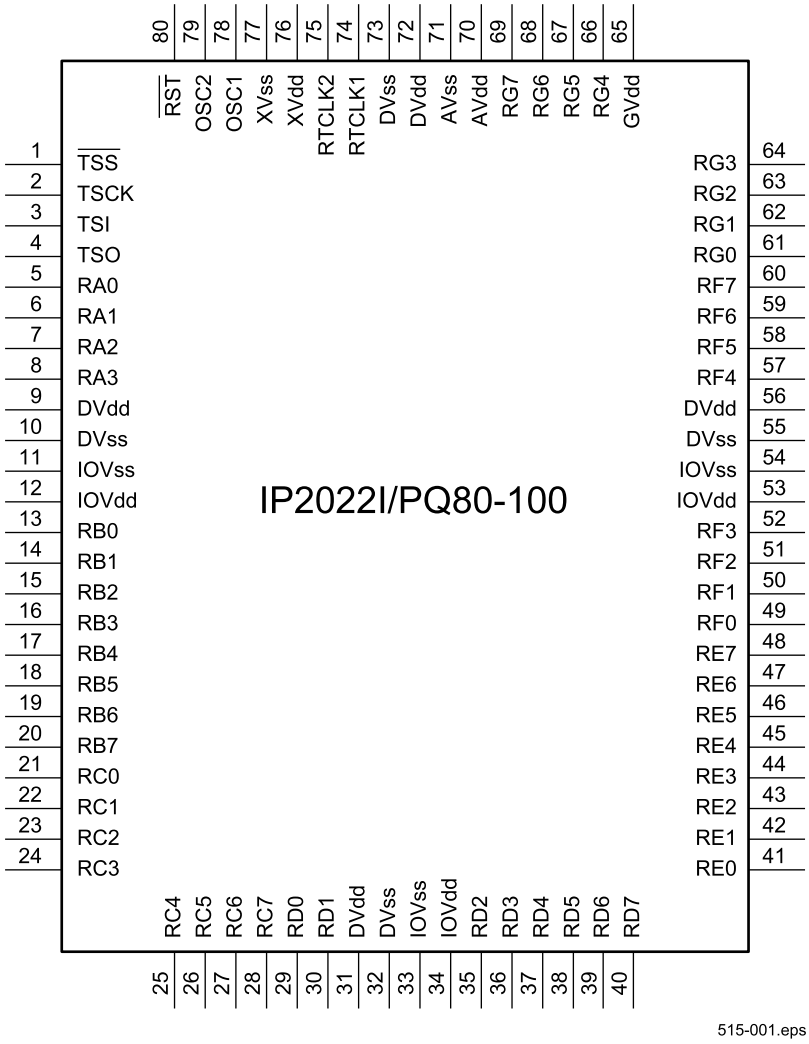


Figure 2-1 Pin Assignments (Top View)

2.1 Signal Descriptions

I = Digital Input, AI = Analog Input, O/DO = Digital Output, HiZ = High Impedance, OD = Open Drain, P = Power, PLP = On-Chip Pullup, ST = Schmitt Trigger

Table 2-1 Signal Descriptions

Name	Pin	Type	Sink @ 3.3V	Source @ 3.3V	Function
AVDD	70	P			Analog Supply
AVSS	71	P			Analog Ground
DVDD	9, 31, 56, 72	P			Logic Supply
DVSS	10, 32, 55, 73	P			Logic Ground
GVDD	65	P			I/O Port G supply
IOVDD	12, 34, 53	P			I/O Supply (except Port G)
IOVSS	11, 33, 54	P			I/O Ground (all ports)
OSC1	78	I			Clock/Crystal/Resonator Input
OSC2	79	O			Crystal/Resonator Output
$\overline{\text{RST}}$	80	I/ST			Reset Input
RTCLK1	74	I			Real-Time Clock/Crystal Input
RTCLK2	75	O			Real-Time Crystal Output
TSCK	2	I/ST/PLP			Target SPI Clock
TSI	3	I/ST/PLP			Target SPI Serial Data Input
TSO	4	O/HiZ			Target SPI Serial Data output
$\overline{\text{TSS}}$	1	I/ST/PLP			Target SPI Slave Select
XVDD	76	P			Analog Supply (crystal/resonator oscillator and PLL)
XVSS	77	P			Analog Ground (crystal/resonator oscillator and PLL)
RA0	5	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Capture 1 Input
RA1	6	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Capture 2 Input
RA2	7	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Clock Input
RA3	8	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Output
RB0	13	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Capture 1 Input
RB1	14	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Capture 2 Input
RB2	15	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Clock Input
RB3	16	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Output
RB4	17	I/O	8 mA	8 mA	I/O Port, External Interrupt
RB5	18	I/O	8 mA	8 mA	I/O Port, External Interrupt, Parallel Slave Peripheral $\overline{\text{HOLD}}$

Table 2-1 Signal Descriptions (continued)

Name	Pin	Type	Sink @ 3.3V	Source @ 3.3V	Function
RB6	19	I/O	8 mA	8 mA	I/O Port, External Interrupt, Parallel Slave Peripheral R/W
RB7	20	I/O	8 mA	8 mA	I/O Port, External Interrupt, Parallel Slave Peripheral $\overline{CS}$
RC0	21	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC1	22	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC2	23	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC3	24	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC4	25	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC5	26	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC6	27	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RC7	28	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD0	29	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD1	30	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD2	35	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD3	36	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD4	37	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD5	38	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD6	39	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RD7	40	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data
RE0	41	I/O	8 mA	8 mA	I/O Port, Serial 1 CLK
RE1	42	I/O	8 mA	8 mA	I/O Port, Serial 1 RXP
RE2	43	I/O	8 mA	8 mA	I/O Port, Serial 1 RXM
RE3	44	I/O	8 mA	8 mA	I/O Port, Serial 1 RXD
RE4	45	I/O	8 mA	8 mA	I/O Port, Serial 1 TXPE
RE5	46	I/O	24 mA	24 mA	I/O Port, High Power Output, Serial 1 TXP
RE6	47	I/O	24 mA	24 mA	I/O Port, High Power Output, Serial 1 TXM
RE7	48	I/O	8 mA	8 mA	I/O Port, Serial 1 TXME
RF0	49	I/O	8 mA	8 mA	I/O Port, Serial 2 TXPE
RF1	50	I/O	24 mA	24 mA	I/O Port, High Power Output, Serial 2 TXP
RF2	51	I/O	24 mA	24 mA	I/O Port, High Power Output, Serial 2 TXM
RF3	52	I/O	8 mA	8 mA	I/O Port, Serial 2 TXME
RF4	57	I/O	8 mA	8 mA	I/O Port, Serial 2 CLK
RF5	58	I/O	8 mA	8 mA	I/O Port, Serial 2 RXP
RF6	59	I/O	8 mA	8 mA	I/O Port, Serial 2 RXM
RF7	60	I/O	8 mA	8 mA	I/O Port, Serial 2 RXD

Table 2-1 Signal Descriptions (continued)

Name	Pin	Type	Sink @ 3.3V	Source @ 3.3V	Function
RG0	61	AI/DO	8 mA	8 mA	Output Port, ADC0 Input, Comparator Output
RG1	62	AI/DO	8 mA	8 mA	Output Port, ADC1 Input, Comparator + Input
RG2	63	AI/DO	8 mA	8 mA	Output Port, ADC2 Input, Comparator – Input
RG3	64	AI/DO	8 mA	8 mA	Output Port, ADC3 Input, ADC reference Input
RG4	66	AI/DO	8 mA	8 mA	Output Port, ADC4 Input
RG5	67	AI/DO	8 mA	8 mA	Output Port, ADC5 Input
RG6	68	AI/DO	8 mA	8 mA	Output Port, ADC6 Input
RG7	69	AI/DO	8 mA	8 mA	Output Port, ADC7 Input

3.0 System Architecture

The IP2022 CPU executes from a 32K × 16 flash program memory and an 8K × 16 RAM program memory. Figure 3-1 shows the IP2022 detailed block diagram. The maximum execution rate is 40 MIPS from flash and 100 MIPS from RAM. Speed-critical routines can be copied from the flash memory to the RAM for faster execution.

The CPU operates on 8-bit data in 128 special-purpose registers, 128 global registers, and 3840 bytes of data memory. The special-purpose registers hold control and status bits used for CPU control and for interface with hardware peripherals (timers, I/O ports, A/D converter, etc.).

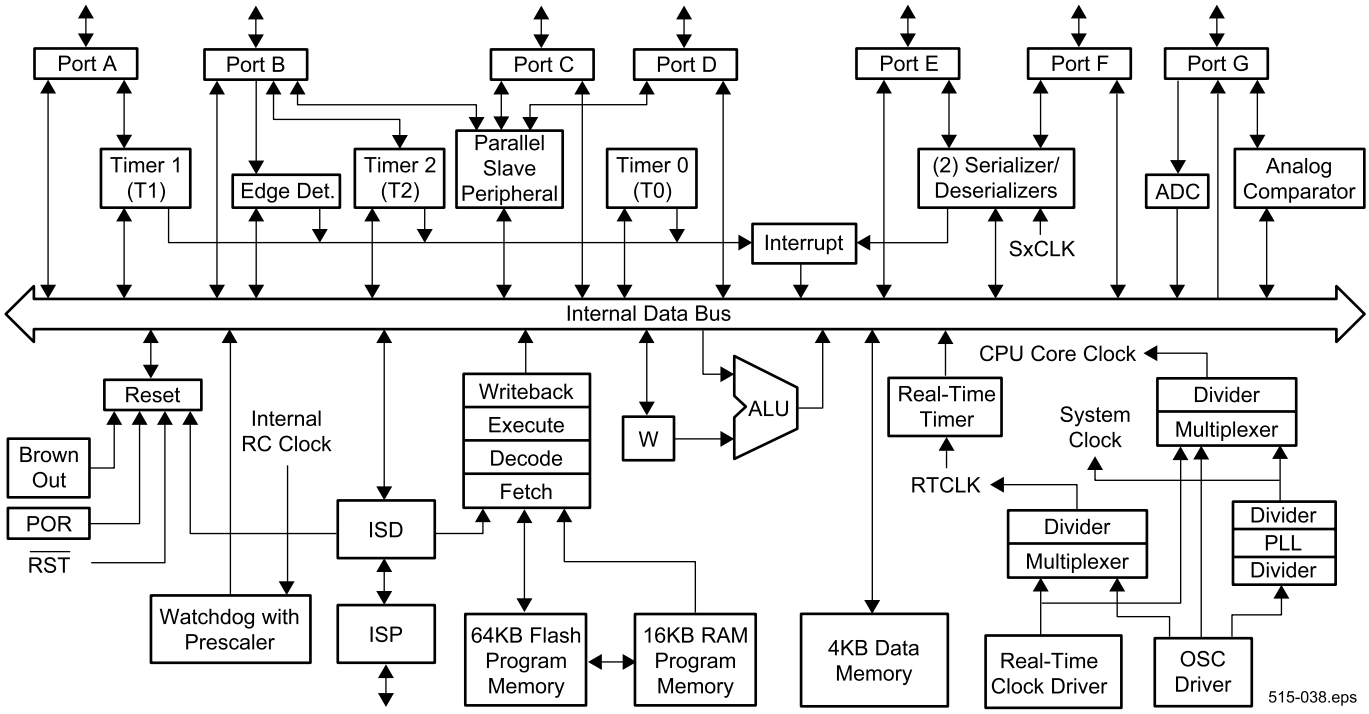


Figure 3-1 IP2022 Detailed Block Diagram

Although the philosophy followed in the design of Ubicom products emphasizes the use of fast RISC CPUs with predictable execution times to emulate peripheral devices in software (called ipModules), there are a few hardware peripherals which are difficult to emulate in software alone (e.g. an A/D converter) or consume an excessive number of instruction cycles when operating at high speed (e.g. data serialization/deserialization). The design of the IP2022 incorporates only those hardware peripherals which can greatly accelerate or extend the reach of the ipModule concept. The hardware peripherals included on-chip are:

- 52 I/O port pins
- Watchdog Timer
- Real-Time Timer
- 2 multifunction 16-bit timers with compare and capture registers
- 2 real-time 8-bit timers
- 2 Serializer/Deserializer channels
- 10-bit, 8-channel A/D converter
- Analog comparator
- Parallel slave peripheral interface

There is a single interrupt vector which can be reprogrammed by software. On-chip peripherals and up to 8 external inputs can raise interrupts.

There are five sources of reset:

- $\overline{\text{RST}}$ external reset input
- Power-On Reset (POR) logic
- Brown-Out Reset (BOR) logic (detects low Vdd condition)
- Watchdog Timer
- Tool Reset from SPI port programming interface

An on-chip PLL clock multiplier enables high-speed operation (up to 100 MHz) from a slow-speed external clock input, crystal, or ceramic resonator. A CPU clock-throttling mechanism allows fine control over power consumption in modes that do not require maximum speed, such as waiting for an interrupt.

The IP2022 has a mechanism for in-system programming of its flash and RAM program memories through a four-wire SPI interface. This provides easy programming and reprogramming of devices on assembled circuit boards. In addition, the flash memory can be programmed by software at run time, for example to store user-specific data such as phone numbers and to receive software upgrades downloaded over the Internet. The IP2022 also has an on-chip debugging facility which makes the internal operation of the chip visible to third-party debugging tools.

3.1 CPU Registers

Figure 3-2 shows the CPU registers, which consist of six 8-bit registers and eight 16-bit registers. The 16-bit registers are formed from pairs of 8-bit registers.

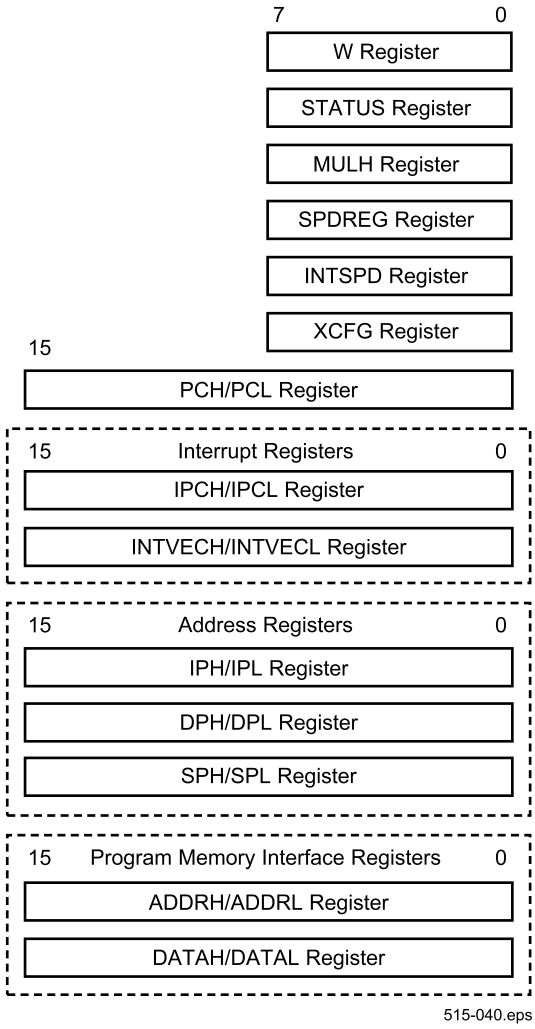


Figure 3-2 CPU Registers

The W or working register is used as the source or destination for most arithmetic and logical instructions.

The STATUS register holds the condition flags for the results of arithmetic and logical operations, the page bits (used for jumps and subroutine calls), and bits which indicate the cause of the last reset (watchdog timer overflow or brown-out voltage detector). Figure 3-3 shows the assignment of the bits in the STATUS register.

7 5	4	3	2	1	0
PA2:0	WD	BO	Z	DC	C

Figure 3-3 STATUS Register

- *PA2:PA0*—Program memory page select bits. Used to extend the 13-bit address encoded in jump and call instructions. Modified using the **page** instruction.
- *WD*—Watchdog time-out bit. Set at reset, if reset was triggered by Watchdog Timer overflow, otherwise cleared.
- *BO*—Brown-out reset bit. Set at reset, if reset was triggered by brown-out voltage level detection, otherwise cleared.
- *Z*—Zero bit. Affected by most logical, arithmetic, and data movement instructions. Set if the result was zero, otherwise cleared.
- *DC*—Digit Carry bit. After addition, set if carry from bit 3 occurred, otherwise cleared. After subtraction, cleared if borrow from bit 3 occurred, otherwise set.
- *C*—Carry bit. After addition, set if carry from bit 7 of the result occurred, otherwise cleared. After subtraction, cleared if borrow from bit 7 of the result occurred, otherwise set. After rotate (**rr** or **rl**) instructions, loaded with the LSB or MSB of the operand, respectively.

The MULH register receives the upper 8 bits of the 16-bit product from signed or unsigned multiplication. The lower 8 bits are loaded into the W register.

The SPDREG register holds bits that indicate the CPU speed and clock source settings loaded by the **speed** instruction, as shown in Figure 3-4. For more information about the **speed** instruction and the clock throttling mechanism, see Section 3.4.

7 6	5 4	3 0
PWRD1:0	CLK1:0	CDIV3:0

Figure 3-4 SPDREG Register

- *PWRD1:0*—controls operation of clock logic.
  - PWRD1—disable PLL clock multiplier; 1 = disabled.
  - PWRD0—disable OSC oscillator; 1 = disabled (stops OSC oscillator and blocks propagation of OSC1 external clock input).
- *CLK1:0*—selects the system clock source, as shown in Table 3-1. See Figure 3-17 for the clock logic.

Table 3-1 CLK1:0 Field Encoding

CLK1:0	System Clock Source
00	PLL Clock Multiplier
01	OSC Oscillator/External OSC1 Input
10	RTCLK Oscillator/External RTCLK1 Input
11	Clock Off

- *CDIV3:0*—selects the clock divisor used to generate the CPU core clock from the system clock, as shown in Table 3-2.

Table 3-2 System Clock Divisor

CDIV3:0	System Clock Divisor	CPU Core Frequency (from 100 MHz System Clock before division)
0000	1	100 MHz
0001	2	50 MHz
0010	3	33.3 MHz
0011	4	25 MHz
0100	5	20 MHz
0101	6	16.7 MHz
0110	8	12.5 MHz
0111	10	10 MHz
1000	12	8.33 MHz
1001	16	6.25 MHz
1010	24	4.17 MHz
1011	32	3.13 MHz
1100	48	2.08 MHz
1101	64	1.56 MHz
1110	128	0.78 MHz
1111	Clock Off	0 MHz

The INTSPD register holds bits that control the CPU speed and clock source during interrupt service routines. It has the same format as the SPDREG register.

The XCFG register holds additional control and status bits, as shown in Figure 3-5.

7	6	5	4 … 1	0
GIE	$\overline{\text{FWP}}$	$\overline{\text{RTEOS}}$	Reserved	FBUSY

Figure 3-5 XCFG Register

- *GIE*—global interrupt enable bit. When set, interrupts are enabled. When clear, interrupts are disabled. For more information about interrupt processing, see Section 3.5.
- *$\overline{\text{FWP}}$*—flash write protect bit. When clear, writes to flash memory are ignored. For more information about programming the flash memory, see Section 4.7.
- *$\overline{\text{RTEOS}}$*—real-time timer oversampling enable bit. When clear, oversampling is used. For more information, see Section 5.3.
- *FBUSY*—read-only flash memory busy bit. Set while executing out of flash memory or while busy processing an **`fread`**, **`fwrite`**, or **`ferase`** instruction. For more information about programming the flash memory, see Section 4.7.

The PCH and PCL register pair form a 16-bit program counter.

The IPCH and IPCL register pair specifies the return address when a **`reti`** instruction is executed.

The INTVECH and INTVECL register pair specifies the interrupt vector. It has a default value of 0 following reset. On a return from interrupt, an option of the **`reti`** instruction allows software to save the incremented value of the program counter in the INTVECH and INTVECL registers.

The IPH and IPL register pair is used as a pointer for indirect addressing. For more information about indirect addressing, see Section 4.1.2.

The DPH and DPL register pair and the SPH and SPL register pair are used as pointer registers for indirect-with-offset addressing. For more information about indirect-with-offset addressing, see Section 4.1.3. The SPH and SPL registers are automatically post-decremented when storing to memory with a **`push`** instruction, and they are automatically pre-incremented when reading from memory with a **`pop`** instruction.

The ADDRH and ADDRL register pair is used to address program memory. On reads, the data from program memory is loaded in the DATAH and DATAL register pair. On writes, the contents of the DATAH and DATAL register pair are loaded into the program memory.

3.2 Data Memory

Figure 3-6 is a map of the data memory. The special-purpose registers and the first 128 data memory locations (between addresses 0x080 and 0x0FF) can be accessed with a direct addressing mode in which the absolute address of the operand is encoded within the instruction. The remaining 3840 bytes of data memory (between addresses 0x100 and 0xFFF) must be accessed using indirect or indirect-with-offset addressing modes. There is one 16-bit register for the indirect address pointer, and two 16-bit registers for indirect-with-offset address pointers. The offset is a 7-bit value encoded within the instruction. For more information about the addressing modes, see Section 4.1.

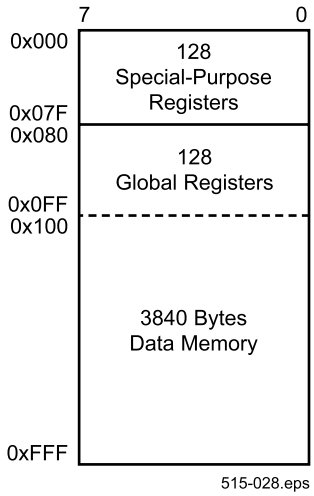


Figure 3-6 Data Memory Map

3.3 Program Memory

Figure 3-7 is a map of the program memory. Architecturally, the IP2022 handles all references to locations in program memory as word addresses. However, the GNU software tools require byte addresses when referring to locations in program memory. As a result, a program memory address interpreted by a tool such as the assembler or debugger must be specified as a byte address. A program memory address interpreted by the IP2022 such as a literal value loaded in the INTVECH/INVECL or ADDRH/ADDRL registers must be specified as a word address.

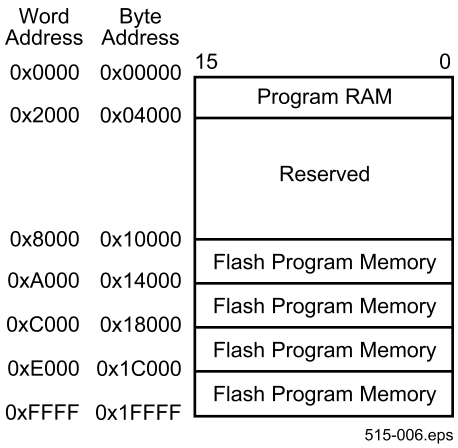


Figure 3-7 Program Memory Map

The program memory is organized as 8K-word pages (16K bytes). Single-instruction jumps and subroutine calls are restricted to be within the same page. Longer jumps and calls require using a **page** instruction to load the upper address bits into the PA2:0 bits of the STATUS register. The **page** instruction must immediately precede the jump or call instruction. The PA2:0 bits should not be modified by writing directly to the STATUS register, because this may cause a mismatch between the PA2:0 bits in the STATUS register and the current program counter (see Section 3.3.2).

3.3.1 Loading the Program RAM

Software loads the program RAM using the **fread** and **iwrite** instructions. The **fread** instruction reads the location in the program flash memory specified by the word address held in the ADDRH and ADDRL registers. The lower eight bits of the address are held in the ADDRL register, and the upper eight bits are held in the ADDRH register. The upper eight bits of the 16-bit word in program memory are then loaded into the DATAH register, and the lower eight bits are loaded into the DATAL register.

The **iwrite** instruction writes the data held in the DATAH and DATAL registers to the word address held in the ADDRH and ADDRL registers. The address must be in program RAM, or no operation will be performed.

3.3.2 Program Counter

The program counter holds the 16-bit address of the instruction to be executed. The lower eight bits of the program counter are held in the PCL register, and the upper eight bits are held in the PCH register. A write to the PCL register will cause a jump to the 16-bit address specified by the PCH and PCL registers. If the PCL register is written as the destination of an **add** or **addc** instruction and carry occurs, the PCH register is automatically incremented. (This may cause a mismatch between the PA2:0 bits in the STATUS register and the current program counter, therefore it is strongly recommended that direct modification of the PCL register is only used for jumps within a page.) The PCH register is read-only. The following code example shows making a jump by loading the PCL register.

```
.org   0x1000
mov    w,#0x22  ;load W with 0x22
mov    pcl,w    ;move W to PCL (i.e. jump to
                ;byte address 0x1044)
```

The **.org** statement is not an instruction. It is an assembler directive that specifies the address for the object code. In this case, the two **mov** instructions are placed at byte addresses 0x1000 and 0x1002 (i.e. word addresses 0x800 and 0x801).

The PA2:0 bits in the STATUS register are not used for address generation, except when a jump or subroutine call instruction is executed. However, when an interrupt is taken, the PA2:0 bits are automatically updated with the upper three bits of the interrupt vector. These bits are restored from the STATUS shadow register when the interrupt service routine returns (i.e. executes a **reti** instruction).

3.4 Low Power Support

Software can change the execution speed of the CPU to reduce power consumption. A mechanism is provided for automatically changing the speed on entry and return from the interrupt service routine. The **speed** instruction specifies power-saving modes that include a clock divisor between 1 and 128. This divisor only affects the clock to the CPU core, not the timers. The **speed** instruction also specifies the clock source (OSC1 clock, RTCLK oscillator, or PLL clock multiplier) and whether to disable the OSC1 clock oscillator or the PLL.

The **speed** instruction executes using the current clock divisor. The new clock divisor takes effect with the following instruction, as shown in the following code example.

```
nop             ;assume divisor is 4, so this
                ;instruction takes 4 cycles
speed  div8     ;change the divisor to 8,
                ;instruction takes 4 cycles
nop             ;instruction takes 8 cycles
speed  div1     ;change the divisor to 1,
                ;instruction takes 8 cycles
nop             ;instruction takes 1 cycle
```

In this example, **div1** and **div8** are assumed to be constants defined with appropriate bit settings for the SPDREG register (see Figure 3-4 for details about the SPDREG encoding).

The SPDREG register holds the current settings for the clock divisor, clock source, and disable bits. These settings can be explicitly changed by executing a **speed** instruction, and they change automatically on interrupts. The SPDREG register is read-only, and its contents may only be changed by executing a **speed** instruction, taking an interrupt, or returning from an interrupt. The INTSPD register specifies the settings used during execution of the interrupt service routine. The INTSPD register is both readable and writeable.

On return from interrupts, the **reti** instruction includes a bit that specifies whether the pre-interrupt speed is restored or the current speed is maintained.

The actual speed of the CPU is indicated by the SPDREG register unless the specified speed is faster than the flash access time (25 ns) and the program is executing out of flash. When program execution moves from program RAM to program flash memory, the new clock divisor will be the greater (slower) of the clock divisor indicated by the SPDREG register and the clock divisor required to avoid violating the flash memory access time. The SPDREG register does not indicate if the flash clock divisor is being used. The value indicated by the SPDREG will be overridden only if the speed is too fast for the flash memory.

The FCFG register holds bits that specify the minimum number of system clock cycles for each flash memory cycle (see Section 4.7.1).

3.4.1 Speed Change Delay

The automatic speed changes require a certain amount of delay to take effect:

- *Changing the Clock Divisor*–there is no delay when the clock divisor is changed.
- *Changing the Clock Source*–the delay is up to one cycle of the slower clock. For example, changing between 32 kHz and 100 MHz could require up to 31.25 microseconds.
- *Powering Up the OSC1 Clock Oscillator*–the speed change delay is specified in the WUDX2:0 bits in the FUSE0 register.
- *Powering Up the PLL Clock Multiplier*–the speed change delay is specified in the WUDP2:0 bits in the FUSE0 register.

If both the OSC oscillator and PLL are re-enabled simultaneously, the delay is controlled by only the WUDX2:0 bits. Bits in the FUSE0 register are flash memory cells which cannot be changed dynamically during program execution.

3.4.2 Instruction Timing

All instructions that perform branches take 3 cycles to complete, consisting of 1 cycle to execute and 2 cycles to load the pipeline.

Table 3-3 Branch Timing

Instruction	Execution Time	Pipeline Load Time
jmp	1	2
call	1	2
ret	1	2
reti	1	2

In the case of an automatic speed change, the execution time will be with respect to the original speed and the pipeline load time will be with respect to the new speed.

Conditional branching is implemented in the IP2022 by using conditional skip instructions to branch over an unconditional jump instruction. To support conditional branching to other pages, the conditional skip instructions will skip over two instructions if the first instruction is a **`page`** instruction. The **`loadh`** and **`loadl`** instructions also cause an additional instruction to be skipped. When any of these conditions occur, it is called an *extended skip instruction*.

Skip instructions take 1 cycle if they do not skip, or 2 cycles if they skip over one instruction. An extended skip instruction may skip over more than one **`loadh`**, **`loadl`**, or **`page`** instruction, however this operation is interruptible and does not affect interrupt latency.

The **`iread`** and **`iwrite`** instructions take 4 cycles. The multiply instructions take 1 cycle.

3.5 Interrupt Support

There are three types of interrupt sources:

- *On-Chip Peripherals*–all of the hardware peripherals except the A/D converter and the analog comparator are capable of generating interrupts. The parallel slave peripheral does not generate interrupts on its own, however it requires programming one of the Port B external interrupt inputs to generate interrupts on its behalf.
- *External Interrupts*–the eight pins on Port B can be programmed to generate interrupts on either rising or falling edges.
- ***`int`*** *Instruction*–the **`int`** instruction can be executed by software to generate an interrupt.

Figure 3-8 shows the system interrupt logic. Each interrupt source (except the **`int`** instruction) has an interrupt enable bit. To be capable of generating an interrupt, the interrupt enable bit and the global interrupt enable (GIE) bit must be set.

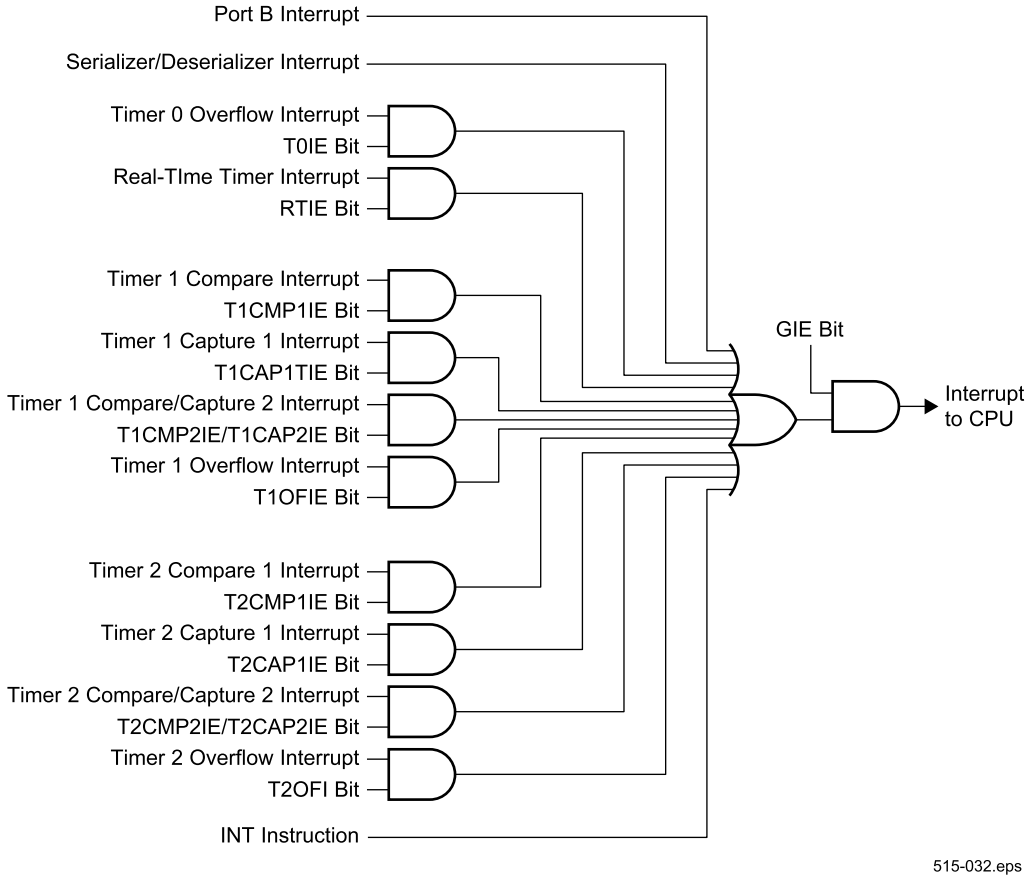


Figure 3-8 System Interrupt Logic

3.5.1 Port B Interrupts

Any of the 8 Port B pins can be configured as an external interrupt input. Logic on these inputs can be programmed to sense rising or falling edges. When an edge is detected, the interrupt flag for the port pin is set.

The recommended initialization sequence is:

1. Configure the port pins used for interrupts as inputs by programming the RBDIR register.
2. Select the desired edge for triggering the interrupt by programming the RBINTED register.
3. Clear the interrupt flags in the RBINTF register.
4. Enable the interrupt input(s) by setting the corresponding bit(s) in the RBINTE register.
5. Set the GIE bit.

Figure 3-9 shows the Port B interrupt logic. Port B has three registers for supporting external interrupts, the RBINTED (Section 5.1.5), RBINTF (Section 5.1.6), and RBINTE (Section 5.1.7) registers. The RBINTED register controls the logic which selects the edge sensitivity (i.e. rising or falling edge) of the Port B pins. When an edge of the selected type occurs, the corresponding flag in the RBINTF register is set. The interrupt signal passed to the system interrupt logic is the OR function of the AND of each interrupt flag in the RBINTF register with its corresponding enable bit in the RBINTE register.

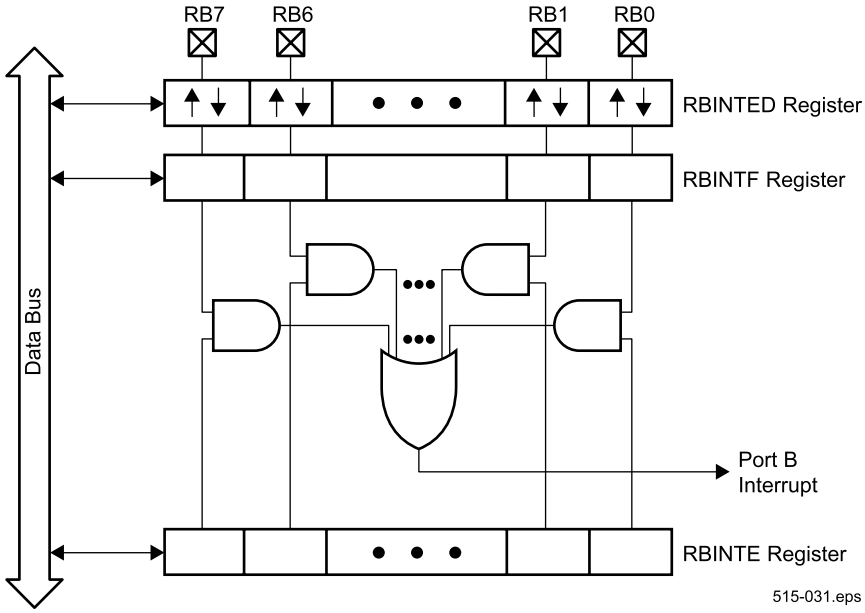


Figure 3-9 Port B Interrupt Logic

3.5.2 Interrupt Processing

There is one interrupt vector held in the INTVECH and INTVECL registers, which is reprogrammable by software. When an interrupt is taken, the current PC is saved in the IPCH and IPCL registers. On return from interrupt (i.e. execution of the **`reti`** instruction), the PC is restored from the IPCH and IPCL registers. Optionally, the **`reti`** instruction may also copy the incremented PC to the INTVECH and INTVECL registers before returning. This has the effect of loading the INTVECH and INTVECL registers with the address of the next instruction following the **`reti`** instruction. This option can be used to directly implement a state machine, such a simple round-robin scheduling mechanism for a series of interrupt service routines (ISRs) in consecutive memory locations.

If multiple sources of interrupts have been enabled, the ISR must check the interrupt flags of each source to determine the cause of the interrupt. The ISR must clear the interrupt flag for the source of the interrupt to prevent retriggering of the interrupt on completion of the ISR (i.e. execution of the **`reti`** instruction). Because the interrupt logic adds a 2-cycle delay between clearing an interrupt flag and deasserting the interrupt request to the CPU, the

flag must be cleared at least 2 cycles before the **`reti`** instruction is taken.

When an interrupt is taken, the registers shown in Figure 3-10 are copied to a shadow register set. Each shadow register is actually a 2-level push-down stack, so one level of interrupt nesting is supported in hardware. The interrupt processing mechanism is completely independent of the 16-level call/return stack used for subroutines.

The contents of the DATAH and DATAL registers are pushed to their shadow registers 4 cycles after the interrupt occurs, to protect the result of any pending **`fread`** instruction. Therefore, software should not access the DATAH or DATAL registers during the first instruction of an ISR.

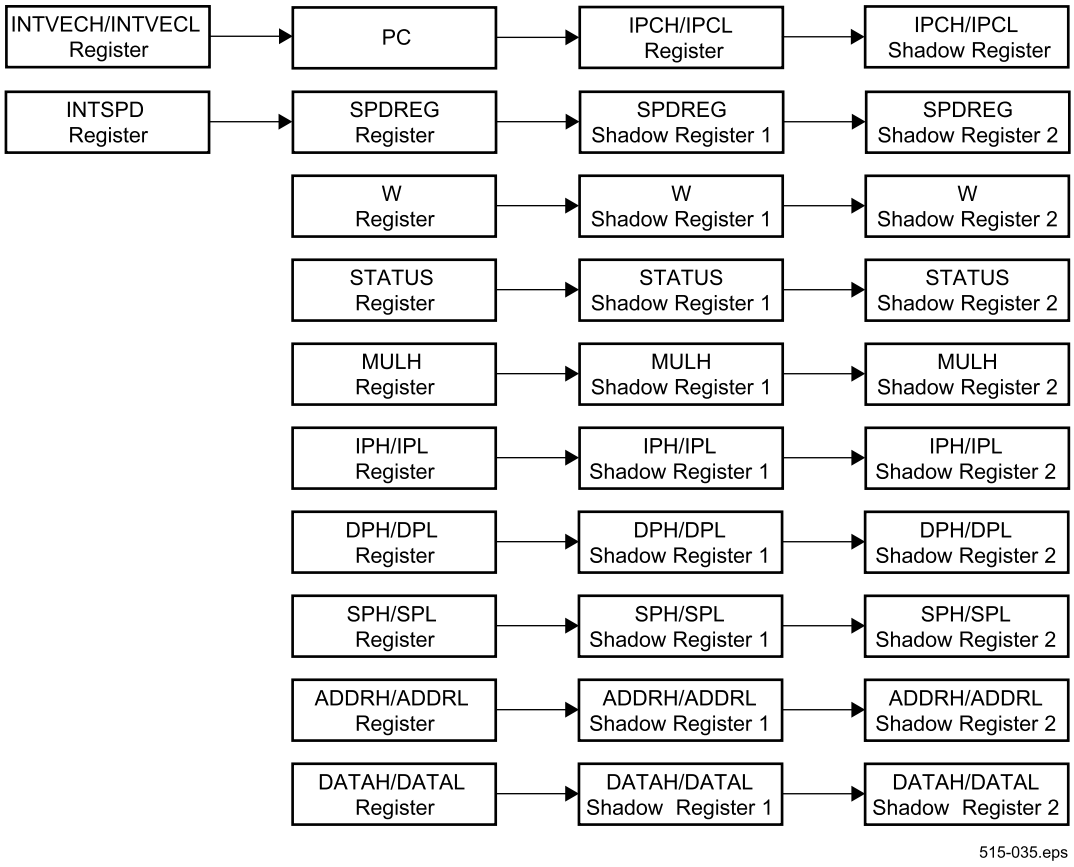


Figure 3-10 Interrupt Processing (On Entry to the ISR)

On return from the ISR, these registers are restored from the shadow registers, as shown in Figure 3-11.

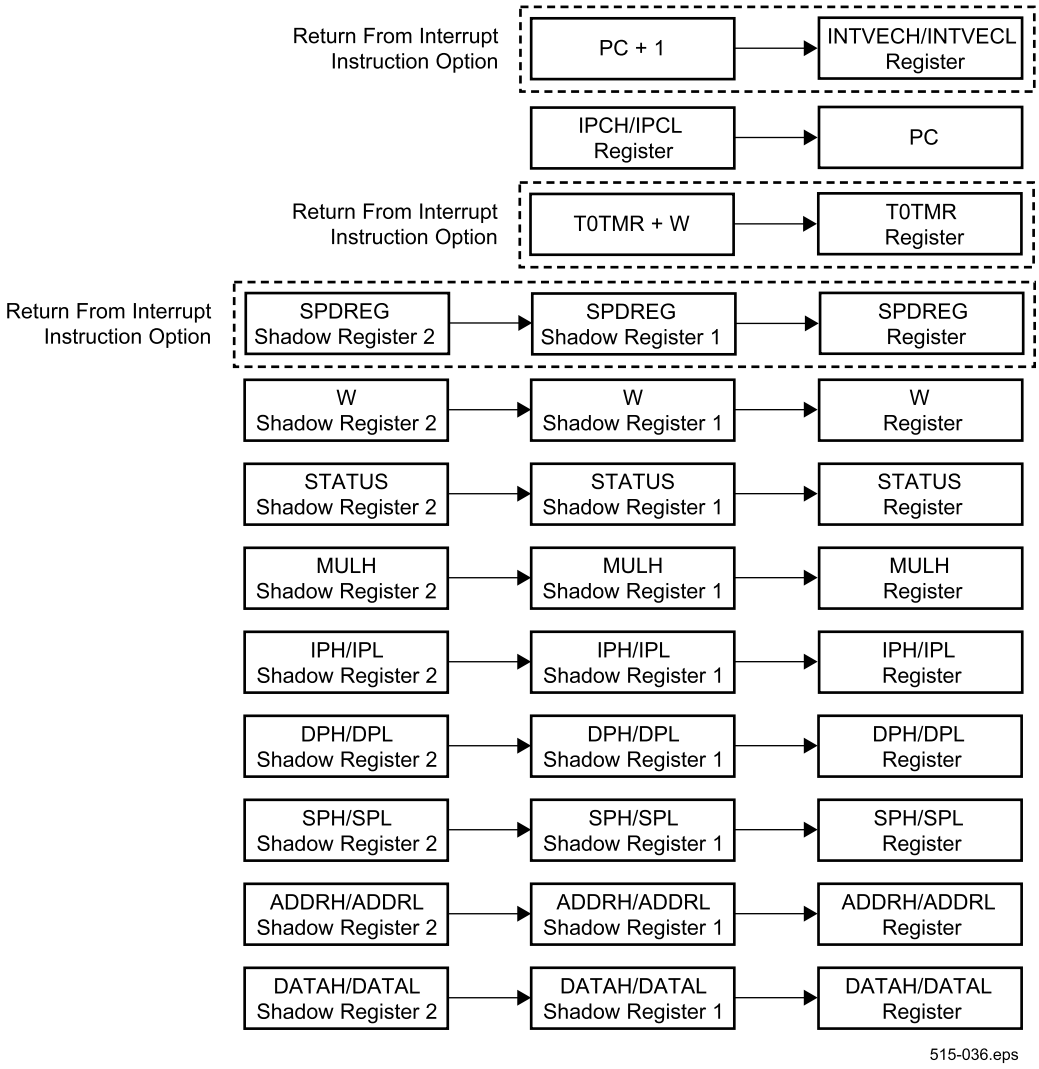


Figure 3-11 Interrupt Processing (On Return from the ISR)

3.5.3 Global Interrupt Enable Bit

The GIE bit serves two purposes:

- Preventing an interrupt in a critical section of mainline code
- Supporting nested interrupts

The GIE bit is automatically cleared when an interrupt occurs to disable interrupts while the ISR is executing. The GIE bit is automatically set by the **`reti`** instruction to re-enable interrupts when the ISR returns.

Table 3-4 GIE Bit Handling

Event	Effect
Enter ISR (interrupt)	GIE bit is cleared
Exit ISR (**`reti`** instruction)	GIE bit is set
`setb gie` instruction (inside ISR)	Enable interrupts for nested interrupt support

Table 3-4 GIE Bit Handling (continued)

Event	Effect
`clrb gie` instruction (inside ISR)	Nothing, the GIE bit is already clear
`setb gie` instruction (mainline code)	Enable interrupts
`clrb gie` instruction (mainline code)	Disable interrupts

To re-enable interrupts during ISR execution, the ISR code must first clear the source of the first interrupt. It may also be desirable to disable specific interrupts before setting the GIE bit to provide interrupt prioritization. Caution must be taken not to exceed the interrupt shadow register stack depth of 2.

Clearing the GIE bit in the ISR cannot be used to globally disable interrupts so that they remain disabled when the ISR returns, because the **`reti`** instruction automatically sets the GIE bit. To disable interrupts in the ISR so that they remain disabled after the ISR returns, the individual interrupt enable bits for each source of interrupts must be cleared.

3.5.4 Interrupt Latency During Speed Change

The interrupt latency is the time from the interrupt event occurring to first ISR instruction being latched from the decode to the execute stage. If the interrupt comes from a Port B input and the SYNC bit in the FUSE1 register is 0, an additional two cycles of synchronization delay are added to the interrupt latency.

The **`iread`** or **`iwrite`** instructions are blocking (i.e. prevent other instructions from being executed) for 4 cycles. If these instructions are used in mainline code, interrupt latency may be increased by an additional 3 cycles.

When an interrupt event is triggered, the CPU speed is changed to the speed specified by the INTSPD register. The SPDREG register is copied to a shadow register, then loaded with the value from the INTSPD register.

If the clock source for the system clock before the interrupt is the same as after the interrupt (i.e. only the core divider is modified), then the interrupt latency is deterministic with respect to the post-interrupt CPU clock. The interrupt latency is 3 cycles for synchronous interrupts.

For example, if the clock divisor is changed from 128 to 1 due to an interrupt then the interrupt latency is 3 cycles with respect to the system clock.

As another example, if the INTSPD register is configured such that the system clock is the PLL and the clock divisor is 1 (100 MIPS from 2 MHz) then the mainline code can reduce the clock divisor down to a slower speed (e.g. a clock divisor of 128) without affecting interrupt latency.

If the clock source is changed, then the delay to change the system clock will be up to one cycle of the slower of the pre- and post-interrupt clocks. The total interrupt latency will be this delay plus the normal interrupt latency (with respect to the new core clock).

If the interrupt speed change requires re-enabling the clock multiplier PLL or crystal oscillator, then the interrupt latency will be extended by the PLL or oscillator startup delay. These delay times are programmed in the WUDP2:0 and WUDX2:0 fields of the FUSE0 register, respectively.

3.5.5 Return From Interrupt

The **`reti`** instruction word includes three bits which control its operation, as shown in Table 3-5. The three bits are specified from assembly language in a literal (e.g. **`reti #0x7`** to specify all bits as 1).

Table 3-5 `reti` Instruction Options

Bit	Function
2	Reinstate the pre-interrupt speed 1 = enable, 0 = disable
1	Store the PC+1 value in the INTVECH and INTVECL registers 1 = enable, 0 = disable
0	Add W to the T0TMR register 1 = enable, 0 = disable

Updating the interrupt vector allows the programmer to implement a sequential state machine. The next interrupt will resume the code directly after the previous **`reti`** instruction.

The **`reti`** instruction takes 1 cycle to execute, and there is a further delay of 2 cycles at the mainline code speed to load the pipeline before the mainline code is resumed.

3.5.6 Disabled Resources

If a peripheral is disabled, it does not have the ability to set an interrupt flag. The interrupt flag, however, is still a valid source of interrupt.

If software sets an interrupt flag, the corresponding interrupt enable bit is set, and the GIE bit is set, then the CPU will be interrupted whether or not the peripheral is enabled or disabled.

If a peripheral is disabled inside the ISR, then its interrupt flag must be cleared to prevent a spurious interrupt from being taken when the ISR completes.

3.5.7 Clock Stop Mode

When a speed change occurs, it is possible for the CPU clock source to be disabled. The clock to the CPU core may be disabled while the system clock is left running, or the system clock may be disabled which also disables the CPU core clock. When the system clock is disabled, the interrupt logic continues to function, and the watchdog timer and real-time timer may be enabled to keep running. (For minimum power consumption in clock stop mode, disable these timers if they are not needed.)

Recovery from clock stop mode to normal execution is possible from these sources:

- External interrupts (i.e. Port B interrupts)
- Real-time timer interrupts
- Watchdog timer overflow reset
- Brown-out voltage reset
- $\overline{\text{RST}}$ external reset

The first two sources listed above do not reset the chip, so program execution continues from where it was stopped. The last three sources reset the chip, so software must perform all of its reset initialization tasks to recover. This usually requires additional time, as compared to recovery through an external interrupt.

3.6 Reset

Power-on reset, brown-out reset, watchdog reset, external reset (from the $\overline{\text{RST}}$ pin), and Target Reset (from the debugging interface) can reset the CPU. Each of these reset conditions causes the program counter to branch to the top of the program memory (word address 0xFFF0 or byte address 0x1FFE0).

The IP2022 incorporates a Power-On Reset (POR) detector that generates an internal reset as Vdd rises during power-up. Figure 3-12 is a block diagram of the reset logic. It includes a 10-bit startup timer and a reset latch. The startup timer controls the reset time-put delay. The reset latch controls the internal reset signal. On power-up, the reset latch is set (CPU held in reset), and the startup timer starts counting once it detects a valid logic high signal on the $\overline{\text{RST}}$ pin. Once the startup timer reaches the end of the timeout period, the reset latch is cleared, releasing the CPU from reset.

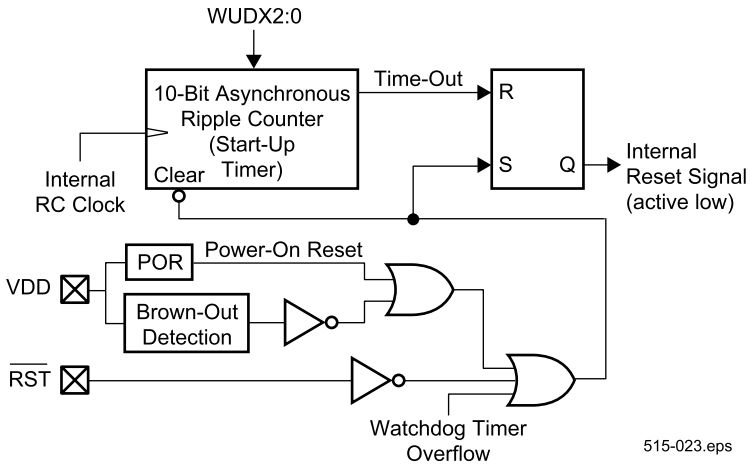


Figure 3-12 On-Chip Reset Circuit Block Diagram

The status register contains two bits to indicate the source of the reset, WD and BO. The WD bit is cleared on reset unless the reset was caused by the watchdog timer, in which case the WD bit is set. The BO bit is cleared on reset unless the reset was caused by the brown-out logic, in which case, the BO bit is set.

Figure 3-13 shows a power-up sequence in which $\overline{RST}$ is not tied to the Vdd pin and the Vdd signal is allowed to rise and stabilize before $\overline{RST}$ pin is brought high. The device will actually come out of reset after the startup stabilization period (Tstartup) from $\overline{RST}$ going high. The WUDX2:0 bits of the FUSE0 register specify the length of the stabilization period.

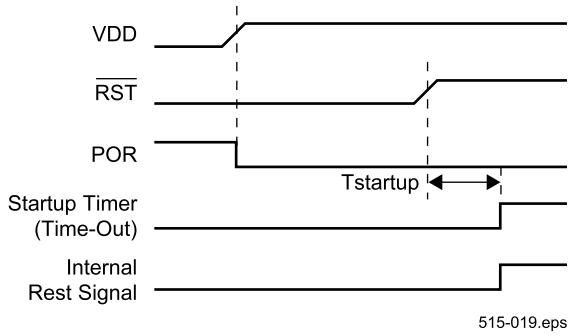


Figure 3-13 Power-On Reset, Separate $\overline{RST}$ Signal

The brown-out circuitry resets the chip when device power (Vdd) dips below its minimum allowed value, but not to zero, and then recovers to the normal value.

Figure 3-14 shows the on-chip Power-On Reset sequence in which the $\overline{RST}$ and Vdd pins are tied together. The Vdd signal is stable before the startup timer expires. In this case, the CPU receives a reliable reset.

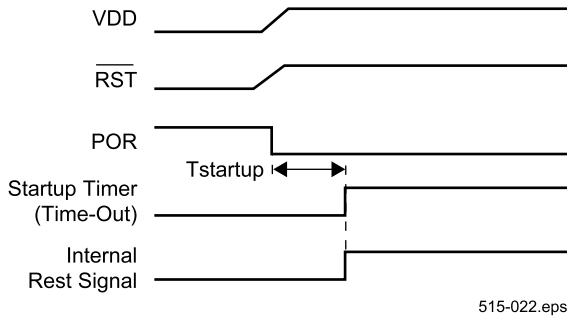


Figure 3-14 Power-On Reset, $\overline{RST}$ Tied To Vdd

However, Figure 3-15 depicts a situation in which Vdd rises too slowly. In this scenario, the startup timer will time out prior to Vdd reaching a valid operating voltage level (Vdd min). This means the CPU will come out of reset and start operating with the supply voltage below the level required for reliable performance. In this situation, an external RC circuit is recommended for driving $\overline{RST}$. The RC delay should exceed the time period required for Vdd to reach a valid operating voltage.

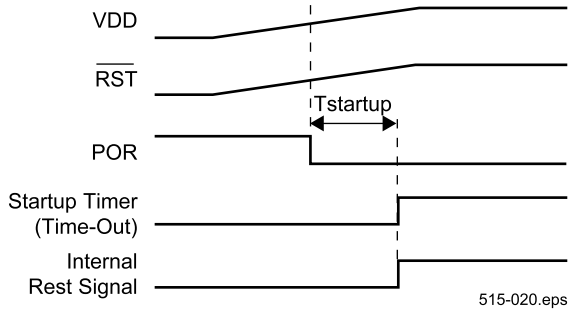


Figure 3-15 Vdd Rise Time Exceeds Tdrt

Figure 3-16 shows the recommended external reset circuit. The external reset circuit is required only if the Vdd rise time has the possibility of being too slow.

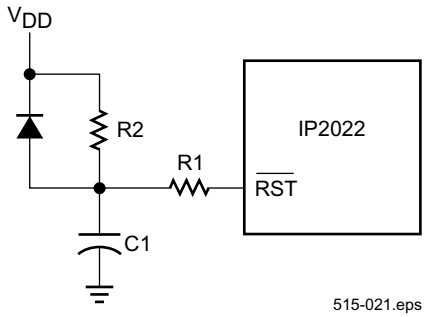


Figure 3-16 External Reset Circuit

The diode D discharges the capacitor when Vdd is powered down.

R1 = 100 Ω to 1 kΩ will limit any current flowing into $\overline{RST}$ from external capacitor C1. This protects the $\overline{RST}$ pin from breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

R2 < 40 kΩ is recommended to make sure that voltage drop across R2 does not violate the device electrical specifications.

3.6.1 Brown-Out Detector

The on-chip brown-out detection circuitry resets the CPU when Vdd dips below the brown-out voltage level programmed in the BOR2:0 bits of the FUSE1 register. Bits in the FUSE1 register are flash memory cells which cannot be changed dynamically during program execution.

The device is held in reset as long as Vdd stays below the brown-out voltage. The CPU will come out of reset when Vdd rises above the brown-out voltage. The brown-out level can be programmed using the BOR2:0 bits in the FUSE register, as shown in Table 3-6.

Table 3-6 Brown-Out Voltage Levels

BOR2:0	Voltage
000	2.30V
001	2.25V
010	2.20V
011	2.15V
100	2.10V
101	2.05V
110	2.00V
111	Disabled

3.6.2 Reset and Interrupt Vectors

After reset, the PC is loaded with 0xFFF0, which is near the top of the program memory space. Typical activities for the reset initialization code include:

- Setting up the FCFG register with appropriate values for flash timing compensation.
- Issuing a **`speed`** instruction to initialize the CPU core clock speed.
- Checking for the cause of reset (brown-out voltage, watchdog timer overflow, or other cause). In some applications, a “warm” reset allows some data initialization procedures to be skipped.
- Copying speed-critical sections of code from flash memory to program RAM.
- Setting up data memory structures (stacks, tables, etc.).
- Initializing peripherals for operation (timers, etc.).
- Initializing the dynamic interrupt vector and enabling interrupts.

Because the default interrupt vector location is 0, which is in program RAM, interrupts cannot be enabled until the ISR is loaded in shadow RAM or the dynamic interrupt vector is loaded with the address of an ISR in flash memory. There is a single dynamic interrupt vector shared by all interrupts. The interrupt vector can be changed by loading the INTVECH and INTVECL registers, or by issuing a **`reti`** instruction with an option specifying that the interrupt vector should be updated with the current PC value.

3.6.3 Register States Following Reset

The effect of different reset sources on a register depends on the register and the type of reset operation. Some registers are initialized to specific values, some are left unchanged, and some are undefined.

A register that starts with an unknown value should be initialized by the software to a known value. Do not simply test the initial state and rely on it starting in that state consistently. Table 3-7 lists the IP2022 registers and shows the state of each register upon reset, with a different column for each type of reset.

Table 3-7 Register States Following Reset

Register	Power-On	RST	Brown-Out Voltage	Watchdog Timer Overflow
PCH (holds word address)	0xFF	0xFF	0xFF	0xFF
PCL (holds word address)	0xF0	0xF0	0xF0	0xF0
CALLH (holds word address)	0xFF	0xFF	0xFF	0xFF
CALLL (holds word address)	0xFF	0xFF	0xFF	0xFF
STATUS	Bits 7:5 - 111 Bits 4:3 - 00 Bits 2:0 - 000	Bits 7:5 - 111 Bits 4:3 - 00 Bits 2:0 - 000	Bits 7:5 - 111 Bits 4:3 - 01 Bits 2:0 - 000	Bits 7:5 - 111 Bits 4:3 - 10 Bits 2:0 - 000
SPDREG	0x93	0x93	0x93	0x93

Table 3-7 Register States Following Reset (continued)

Register	Power-On	RST	Brown-Out Voltage	Watchdog Timer Overflow
XCFG	0x01	0x01	0x01	0x01
Global registers	Undefined	Unchanged	Undefined	Unchanged
Data memory	Undefined	Unchanged	Undefined	Unchanged
All other CPU registers	0x00	0x00	0x00	0x00
All I/O port registers except RxDIR	0x00	0x00	0x00	0x00
RxDIR registers	0xFF	0xFF	0xFF	0xFF
All other peripheral registers	0x00	0x00	0x00	0x00

3.7 Clock Oscillator

There are two clock oscillators, the OSC oscillator and the RTCLK oscillator. The OSC oscillator is capable of operating at 1 to 6 MHz using an external crystal or ceramic resonator. Using the PLL clock multiplier, the OSC clock is intended to provide the time base for running the CPU core at speeds up to 100 MHz. The RTCLK oscillator operates at 32.768 kHz using an external crystal. This oscillator is intended for running the real-time timer when the OSC oscillator and PLL clock multiplier are turned off. Either clock source can be driven by an external clock signal, up to 150 MHz for the OSC1 input and up to 100 MHz for the RTCLK1 input.

Figure 3-17 shows the clock logic. The PLL clock multiplier has a fixed multiplication factor of 50. The PLL is preceded by a divider capable of any integer divisor between 1 and 8, as controlled by the PIN2:0 bits of the FUSE0 register. The PLL is followed by a second divider capable of any integer divisor between 1 and 4, as controlled by the POUT1:0 bits of the FUSE0 register. A third divider which only affects the clock to the CPU core is controlled by the speed change mechanism described in Section 3.4. If both the OSC oscillator and PLL are re-enabled simultaneously, the delay is controlled by only the WUDX2:0 bits. Bits in the FUSE0 register are flash memory cells which cannot be changed dynamically during program execution.

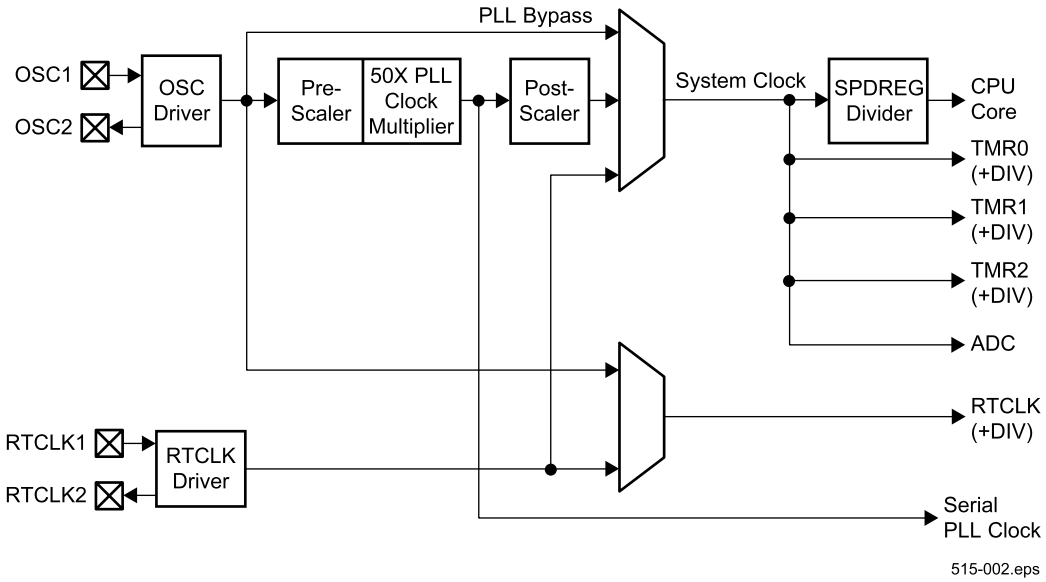


Figure 3-17 Clock Logic

3.7.1 External Connections

Figure 3-18 shows the connections for driving the OSC or RTCLK clock sources with an external signal. To drive the OSC clock source, the external clock signal is driven on the OSC1 pin and the OSC2 pin is left open. The external clock signal driven on the OSC1 pin may be any frequency up to 150 MHz. To drive the RTCLK clock source, the external clock signal is driven on the RTCLK1 input and the RTCLK2 output is left open. The external clock signal driven on the RTCLK1 pin may be any frequency up to 100 MHz.

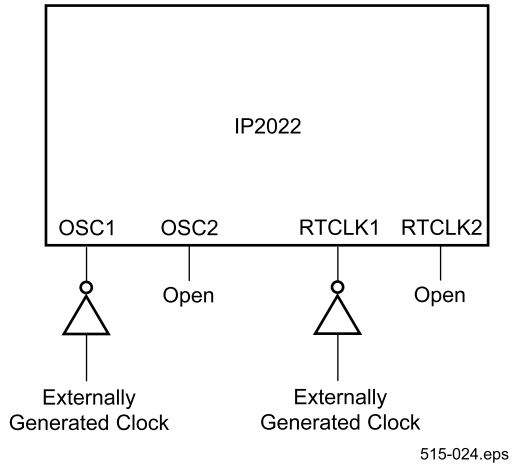


Figure 3-18 External Clock Input

Figure 3-19 shows the connections for attaching an external crystal to the OSC or RTCLK oscillator. For the OSC oscillator, a crystal between 1 MHz and 6 MHz is connected across the OSC1 and OSC2 pins. For the RTCLK oscillator, a 32.768 kHz crystal is connected across the RTCLK1 and RTCLK2 pins. No external capacitors are required.

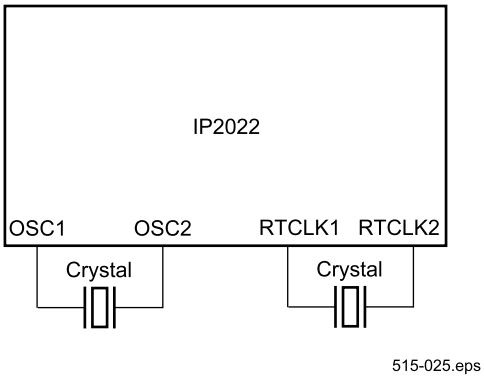


Figure 3-19 Crystal Connection

Figure 3-20 shows the connections for attaching an external ceramic resonator to the OSC oscillator. The frequency of the resonator must be between 1 MHz and 6 MHz. The value of the external capacitors C1 and C2 is 5 pF. The RTCLK oscillator may not be used with an external ceramic resonator.

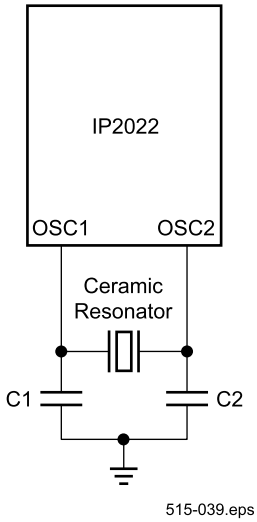


Figure 3-20 Ceramic Resonator Connection

3.8 Configuration Block

The configuration block is a set of flash memory registers outside of both program memory and data memory. These registers are not readable or writeable at run time.

The FUSE0, FUSE1, and TRIM0 registers hold settings that must be specified by system designers. The other configuration block registers are used by software tools. See the IP2022 User's Manual for details about programming the configuration block registers.

3.8.1 FUSE0 Register

15	14	13 12	11 9	8 7 6	5 3	2 0
$\overline{\text{XTAL}}$	$\overline{\text{RTCLK}}$	POUT1:0	PIN2:0	Reserved	WUDP2:0	WUDX2:0

Figure 3-21 FUSE0 Register

$\overline{\text{XTAL}}$ Disables OSC2 crystal drive output
- 0 = enabled
- 1 = disabled

$\overline{\text{RTCLK}}$ Disables RTCLK2 crystal drive output
- 0 = enabled
- 1 = disabled

POUT1:0 Specifies PLL clock multiplier postscaler divisor
- 00 = 1
- 01 = 2
- 10 = 3
- 11 = 4

PIN2:0 Specifies PLL clock multiplier prescaler divisor
- 000 = 1
- 001 = 2
- 010 = 3
- 011 = 4
- 100 = 5
- 101 = 6
- 110 = 7
- 111 = 8

WUDP2:0 Specifies PLL clock multiplier start-up stabilization period
- 000 = 128 µs
- 001 = 192 µs
- 010 = 320 µs
- 011 = 576 µs
- 100 = 1.088 ms
- 101 = 2.112 ms
- 110 = 4.160 ms
- 111 = 8.256 ms

WUDX2:0 Specifies OSC oscillator start-up stabilization period
- 000 = 320 µs
- 001 = 1.088 ms
- 010 = 4.160 ms
- 011 = 8.256 ms
- 100 = 16.448 ms
- 101 = 65.600 ms
- 110 = 524.352 ms
- 111 = 1048.64 ms

3.8.2 FUSE1 Register

15	14	13 … 7	6 5 4	3	2 1 0
$\overline{CP}$	$\overline{SYNC}$	Reserved	BOR2:0	WDTE	WDPS2:0

Figure 3-22 FUSE1 Register

$\overline{CP}$ Clear to enable code protection. Once cleared, this bit cannot be set until the entire device is erased. When code protection is enabled, program memory reads as all 0 to an external device programmer. This bit does not affect access to program flash memory made by software, using the **`fread`** instruction.

0 = enabled

1 = disabled

$\overline{SYNC}$ Set to read directly from the port pins through the RxIN register, clear to read through a synchronization register. This bit should be clear if any external devices that can be read from I/O port pins are running asynchronously to the CPU core clock. See Section 5.1.

0 = enabled

1 = disabled

BOR2:0 Specifies brown-out voltage level. If Vdd goes below this level, the IP2022 is reset.

000 = 2.40V

001 = 2.35V

010 = 2.30V

011 = 2.25V

100 = 2.20V

101 = 2.10V

110 = 2.00V

111 = Disabled, no brown-out reset can occur.

WDTE Enables Watchdog Timer.

0 = disabled

1 = enabled

WDPS2:0 Specifies the Watchdog Timer prescaler divisor. This controls the time period before the Watchdog Timer expires. If the Watchdog Timer is enabled, software must clear the Watchdog Timer periodically within this time period to prevent a reset of the IP2022 from occurring. The **`cwdt`** instruction clears both the Watchdog Timer and its prescaler.

000 = 16.4 ms

001 = 32.8 ms

010 = 65.6 ms

011 = 131 ms

100 = 262 ms

101 = 524 ms

110 = 1049 ms

111 = 2099 ms

3.8.3 TRIM0 Register

15 – 11	10	9 – 7	6 – 5	4 – 3	2 – 1 – 0
Reserved	FPERT	CMPT2:0	WDTT1:0	BORT1:0	VCOT2:0

Figure 3-23 TRIM0 Register

FPERT	Controls flash block pulse erase time. 0 = 20 ms 1 = 10 ms
CMPT2:0	Comparator response time trim bits.
WDTT1:0	Watchdog Timer trim bits.
BORT2:0	Brown-out detector trim bits.
VCOT2:0	PLL VCO frequency trim bits.

3.9 Special-Purpose Register Map

Table 3-8 shows the addresses and reset values of all special-purpose registers.

Table 3-8 Register Addresses and Reset State

Address	Name	Description	Register Status Following Reset (Power-On, RST, BOR, Watchdog)
0x0001	Reserved	Reserved	Reserved
0x0002	Reserved	Reserved	Reserved
0x0003	Reserved	Reserved	Reserved
0x0004	IPH	Indirect Pointer (high byte)	0000 0000
0x0005	IPL	Indirect Pointer (low byte, see Section 4.1)	0000 0000
0x0006	SPH	Stack Pointer (high byte)	0000 0000
0x0007	SPL	Stack Pointer (low byte, see Section 4.1)	0000 0000
0x0008	PCH	Current PC bits 15:8 (read-only)	1111 1111
0x0009	PCL	Virtual register for direct PC modification	1111 0000
0x000A	WREG	W register	0000 0000
0x000B	STATUS	STATUS register	See Table 3-7
0x000C	DPH	Data Pointer (high byte)	0000 0000
0x000D	DPL	Data Pointer (low byte, see Section 4.1)	0000 0000
0x000E	SPDREG	Current speed (read-only)	1001 0011
0x000F	MULH	Multiply result (high byte)	0000 0000
0x0010	ADDRH	Program memory address (high byte)	0000 0000
0x0011	ADDRL	Program memory address (low byte, see Section 4.1)	0000 0000
0x0012	DATAH	Program memory data (high byte)	0000 0000
0x0013	DATAL	Program memory data (low byte)	0000 0000
0x0014	INTVECH	Interrupt vector (high byte)	0000 0000
0x0015	INTVECL	Interrupt vector (low byte)	0000 0000
0x0016	INTSPD	Interrupt speed register	0000 0000
0x0017	RBINTF	Port B interrupt flags	0000 0000
0x0018	RBINTE	Port B interrupt enable bits	0000 0000
0x0019	RBINTED	Port B interrupt edge select bits	0000 0000
0x001A	FCFG	Flash configuration register	0000 0000
0x001B	TCTRL	Timer 1/2 common control register	0000 0000
0x001C	XCFG	Extended configuration (bit 0 is read-only)	0000 0001
0x001D	Reserved	Reserved	Reserved
0x001E	IPCH	Interrupt return address (high byte)	0000 0000

Table 3-8 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, BOR, Watchdog)
0x001F	IPCL	Interrupt return address (low byte)	0000 0000
0x0020	RAIN	Data on Port A pins	N/A
0x0021	RAOUT	Port A output latch	0000 0000
0x0022	RADIR	Port A direction register	1111 1111
0x0023	Reserved	Reserved	Reserved
0x0024	RBIN	Data on Port B pins	N/A
0x0025	RBOUT	Port B output latch	0000 0000
0x0026	RBDIR	Port B direction register	1111 1111
0x0027	Reserved	Reserved	Reserved
0x0028	RCIN	Data on Port C pins	N/A
0x0029	RCOUT	Port C output latch	0000 0000
0x002A	RCDIR	Port C direction register	1111 1111
0x002B	Reserved	Reserved	Reserved
0x002C	RDIN	Data on Port D pins	N/A
0x002D	RDOUT	Port D output latch	0000 0000
0x002E	RDDIR	Port D direction register	1111 1111
0x002F	Reserved	Reserved	Reserved
0x0030	REIN	Data on Port E pins	N/A
0x0031	REOUT	Port E output latch	0000 0000
0x0032	REDIR	Port E direction register	1111 1111
0x0033	Reserved	Reserved	Reserved
0x0034	RFIN	Data on Port F pins	N/A
0x0035	RFOUT	Port F output latch	0000 0000
0x0036	RFDIR	Port F direction register	1111 1111
0x0037	Reserved	Reserved	Reserved
0x0038	Reserved	Reserved	Reserved
0x0039	RGOUT	Port G output latch	0000 0000
0x003A	RGDIR	Port G direction register	1111 1111
0x003B	Reserved	Reserved	Reserved
0x003C	Reserved	Reserved	Reserved
0x003D	Reserved	Reserved	Reserved
0x003E	Reserved	Reserved	Reserved
0x003F	Reserved	Reserved	Reserved
0x0040	RTTMR	Real-time timer value	0000 0000

Table 3-8 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, BOR, Watchdog)
0x0041	RTCFG	Real-time timer configuration register	0000 0000
0x0042	T0TMR	Timer 0 value	0000 0000
0x0043	T0CFG	Timer 0 configuration register	0000 0000
0x0044	T1CNTH	Timer 1 counter register high (read only)	0000 0000
0x0045	T1CNTL	Timer 1 counter register low (read only)	0000 0000
0x0046	T1CAP1H	Timer 1 Capture 1 register high (read only)	0000 0000
0x0047	T1CAP1L	Timer 1 Capture 1 register low (read only)	0000 0000
0x0048	T1CAP2H/T1CMP2H	Timer 1 Capture 2/Compare 2 register high	0000 0000
0x0049	T1CAP2L/T1CMP2L	Timer 1 Capture 2/Compare 2 register low	0000 0000
0x004A	T1CMP1H	Timer 1 Compare 1 register high	0000 0000
0x004B	T1CMP1L	Timer 1 Compare 1 register low	0000 0000
0x004C	T1CFG1H	Timer 1 configuration register 1 high	0000 0000
0x004D	T1CFG1L	Timer 1 configuration register 1 low	0000 0000
0x004E	T1CFG2H	Timer 1 configuration register 2 high	0000 0000
0x004F	T1CFG2L	Timer 1 configuration register 2 low	0000 0000
0x0050	ADCH	ADC value (high)	0000 0000
0x0051	ADCL	ADC value (low)	0000 0000
0x0052	ADCCFG	ADC configuration register	0000 0000
0x0053	ADCTMR	ADC timer register	0000 0000
0x0054	T2CNTH	Timer 2 counter register high (read only)	0000 0000
0x0055	T2CNTL	Timer 2 counter register low (read only)	0000 0000
0x0056	T2CAP1H	Timer 2 Capture 1 register high (read only)	0000 0000
0x0057	T2CAP1L	Timer 2 Capture 1 register low (read only)	0000 0000
0x0058	T2CAP2H/T2CMP2H	Timer 2 Capture 2/Compare 2 register high	0000 0000
0x0059	T2CAP2L/T2CMP2L	Timer 2 Capture 2/Compare 2 register low	0000 0000
0x005A	T2CMP1H	Timer 2 Compare 1 register high	0000 0000
0x005B	T2CMP1L	Timer 2 Compare 1 register low	0000 0000
0x005C	T2CFG1H	Timer 2 configuration register 1 high	0000 0000
0x005D	T2CFG1L	Timer 2 configuration register 1 low	0000 0000
0x005E	T2CFG2H	Timer 2 configuration register 2 high	0000 0000
0x005F	T2CFG2L	Timer 2 configuration register 2 low	0000 0000
0x0060	S1TMRH	Serializer 1 clock timer register (high bits)	0000 0000

Table 3-8 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, BOR, Watchdog)
0x0061	S1TMRL	Serializer 1 clock timer register (low bits)	0000 0000
0x0062	S1TBUFH	Serializer 1 transmit buffer (high bits)	0000 0000
0x0063	S1TBUFL	Serializer 1 transmit buffer (low bits)	0000 0000
0x0064	S1TCFG	Serializer 1 transmit configuration	0000 0000
0x0065	S1RCNT	Serializer 1 received bit count (actual) (read-only)	0000 0000
0x0066	S1RBUFH	Serializer 1 receive buffer (high bits)	0000 0000
0x0067	S1RBUFL	Serializer 1 receive buffer (low bits)	0000 0000
0x0068	S1RCFG	Serializer 1 receive configuration	0000 0000
0x0069	S1RSYNC	Serializer 1 receive bit sync pattern	0000 0000
0x006A	S1INTF	Serializer 1 status/Interrupt flags	0000 0000
0x006B	S1INTE	Serializer 1 Interrupt enable bits	0000 0000
0x006C	S1MODE	Serializer 1 serial mode/clock select register	0000 0000
0x006D	S1SMASK	Serializer 1 receive sync mask	0000 0000
0x006E	PSPCFG	Serializer 1 parallel slave peripheral config register	0000 0000
0x006F	CMPCFG	Serializer 1 comparator configuration register	0000 0000
0x0070	S2TMRH	Serializer 2 clock timer register (high bits)	0000 0000
0x0071	S2TMRL	Serializer 2 clock timer register (low bits)	0000 0000
0x0072	S2TBUFH	Serializer 2 transmit buffer (high bits)	0000 0000
0x0073	S2TBUFL	Serializer 2 transmit buffer (low bits)	0000 0000
0x0074	S2TCFG	Serializer 2 transmit configuration	0000 0000
0x0075	S2RCNT	Serializer 2 received bit count (actual) (read-only)	0000 0000
0x0076	S2RBUFH	Serializer 2 receive buffer (high bits)	0000 0000
0x0077	S2RBUFL	Serializer 2 receive buffer (low bits)	0000 0000
0x0078	S2RCFG	Serializer 2 receive configuration	0000 0000
0x0079	S2RSYNC	Serializer 2 receive bit sync pattern	0000 0000
0x007A	S2INTF	Serializer 2 status/Interrupt flags	0000 0000
0x007B	S2INTE	Serializer 2 interrupt enable bits	0000 0000
0x007C	S2MODE	Serializer 2 serial mode/clock select register	0000 0000
0x007D	S2SMASK	Serializer 2 receive sync mask	0000 0000
0x007E	CALLH	Top of call stack (high 8 bits)	1111 1111
0x007F	CALLL	Top of call stack (low 8 bits)	1111 1111

4.0 Instruction Set Architecture

The IP2022 implements a powerful load-store RISC architecture with a rich set of arithmetic and logical operations, including signed and unsigned 8-bit × 8-bit integer multiply with a 16-bit product.

The CPU operates on data held in 128 special-purpose registers, 128 global registers, and 3840 bytes of data memory. The special-purpose registers are dedicated to control and status functions for the CPU and peripherals. The global registers and data memory may be used for any functions required by software, the only distinction among them being that the 128 global registers (addresses 0x080 to 0x0FF) can be accessed using a direct addressing mode. The remaining 3840 bytes of data memory (between addresses 0x100 and 0xFFF) must be accessed using indirect or indirect-with-offset addressing modes. The IPH/IPL register is the pointer for the indirect addressing mode, and the DPH/DPL and SPH/SPL registers are the pointers for the indirect-with-offset addressing modes.

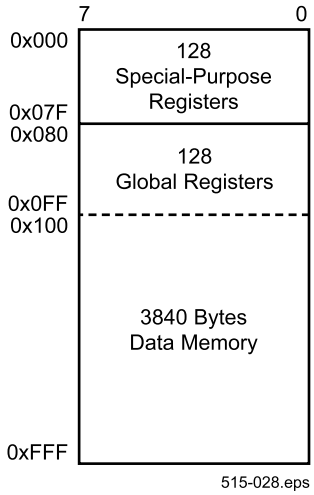


Figure 4-1 Data Memory Map

4.1 Addressing Modes

The arithmetic and logical instructions have one or two operands. The two-operand instructions implicitly use the W register as one of the operands. The single operand used by one-operand instructions and one of the two operands used by two-operand instructions are accessed in the data memory using addressing modes. A 9-bit field within the instruction, called the “fr” field, specifies the addressing mode and the address (in the case of direct addressing) or the address offset (in the case of indirect-with-offset addressing), as shown in Table 4-1.

Table 4-1 Addressing Mode Summary

“fr” Field	Mode	Syntax	Effective Address (EA)	Restrictions
000000000	Indirect	`mov w,(ip)` `mov (ip),w`	IPH || IPL	EA ≥ 0x020
00nnnnnnn	Direct, special-purpose registers	`mov w,fr` `mov fr,w`	nnnnnnn	0x001 ≤ EA ≤ 0x07F
01nnnnnnn	Direct, global registers	`mov w,fr` `mov fr,w`	0x080 + nnnnnnn	0x080 ≤ EA ≤ 0x0FF
10nnnnnnn	Indirect with offset, data pointer	`mov w,offset(dp)` `mov offset(dp),w`	DPH || DPL + nnnnnnn	0x000 ≤ nnnnnnn ≤ 0x0FF EA ≥ 0x020
11nnnnnnn	Indirect with offset, stack pointer	`mov w,offset(sp)` `mov offset(sp),w`	SPH || SPL + nnnnnnn	0x000 ≤ nnnnnnn ≤ 0x0FF EA ≥ 0x020

When an addition or increment instruction (i.e. **`add`**, **`addc`**, **`inc`**, **`incsz`**, or **`incsnz`**) on the low half of a 16-bit pointer register (i.e. IPL, DPL, SPL, or ADDRL) results in a carry, the high half of the register is incremented. For example, if the IP register holds 0x00FF and an **`inc ipl`** instruction is executed, the register will hold 0x0100 after the instruction. When a subtraction or decrement instruction (i.e. **`sub`**, **`subc`**, **`dec`**, **`decsz`**, or **`decsnz`**) results in a borrow, the high half of the register is decremented. Because carry and borrow are automatically handled, the **`addc`** and **`subc`** instructions are not needed for 16-bit arithmetic on pointer registers.

4.1.1 Direct Addressing Mode

Figure 4-2 shows the direct addressing mode used to reference the special-purpose registers. Seven bits from the “fr” field allow addressing up to 128 special-purpose registers. (Not all 128 locations in this space are implemented in the IP2022; several locations are reserved for future expansion.)

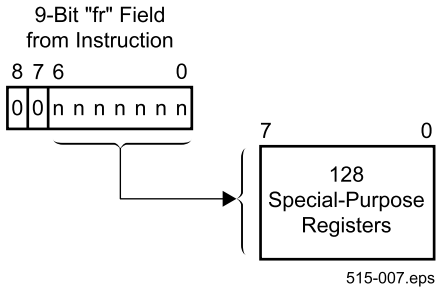


Figure 4-2 Direct Mode, Special-Purpose Registers

The following code example uses direct mode.

```
mov   w,0x0012  ;load W with the contents of
                ;the memory location at 0x0012
                ;(the DATAL register)
```

Figure 4-3 shows the direct addressing mode used to reference the global registers. This mode is distinguished from the mode used to access the special-purpose registers in bit 7 of the “fr” field. Because these registers have this additional addressing mode not available for the other data memory locations, they are especially useful for holding global variables and frequently accessed data.

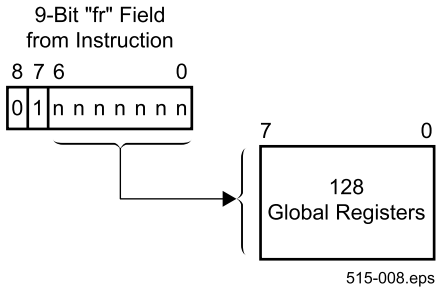


Figure 4-3 Direct Mode, Global Registers

4.1.2 Indirect Addressing Mode

The indirect addressing mode is used when all of the bits in the “fr” field are clear. The location of the operand is specified by a 12-bit pointer in the IPH and IPL registers. The upper four bits of the IPH register are not used. Addresses from 0x000 to 0x01F cannot be accessed reliably with this addressing mode, therefore it must not be used for this purpose. (Direct mode should be used instead.) Figure 4-4 shows indirect mode.

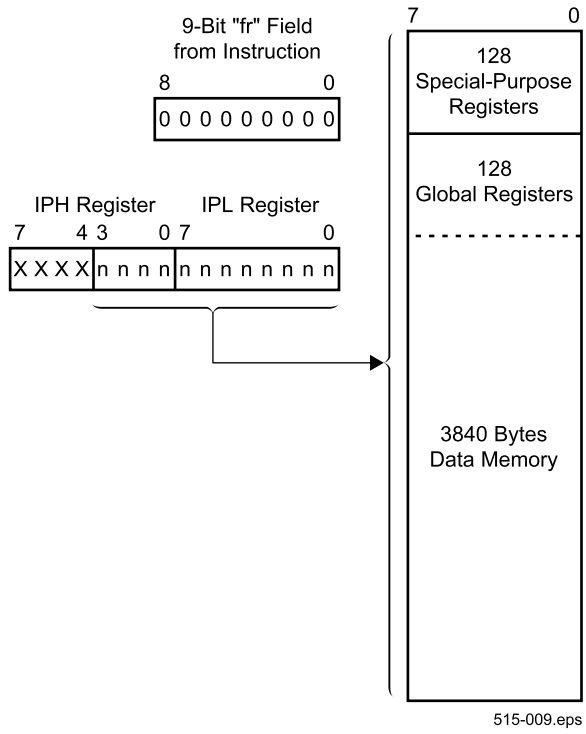


Figure 4-4 Indirect Mode

The following code example uses indirect mode.

```
mov   w,#0x03   ;load W with 0x03
mov   iph,w     ;load the high byte of the
                ;indirect pointer from W
mov   w,#0x85   ;load W with 0x85
mov   ipl,w     ;load the low byte of the
                ;indirect pointer from W
mov   w,(ip)    ;load W with the contents of
                ;the memory location at
                ;effective address 0x0385
```

4.1.3 Indirect-with-Offset Modes

The indirect-with-offset addressing mode is used when bit 8 of the “fr” field is set. The location of the operand is specified by a 7-bit unsigned immediate from the “fr” field added to a 12-bit base address in a pointer register. Addresses from 0x000 to 0x01F cannot be accessed reliably with this addressing mode, therefore it must not be used for this purpose. (Direct mode should be used instead.)

When bit 7 of the “fr” field is clear, the DPH/DPL register is selected as the pointer register. This register is accessed using the **loadh** and **loadl** instructions, which load its high and low bytes, respectively. The upper four bits of the DPH register are not used. Figure 4-5 shows indirect-with-offset addressing using the DPH/DPL register as the pointer register.

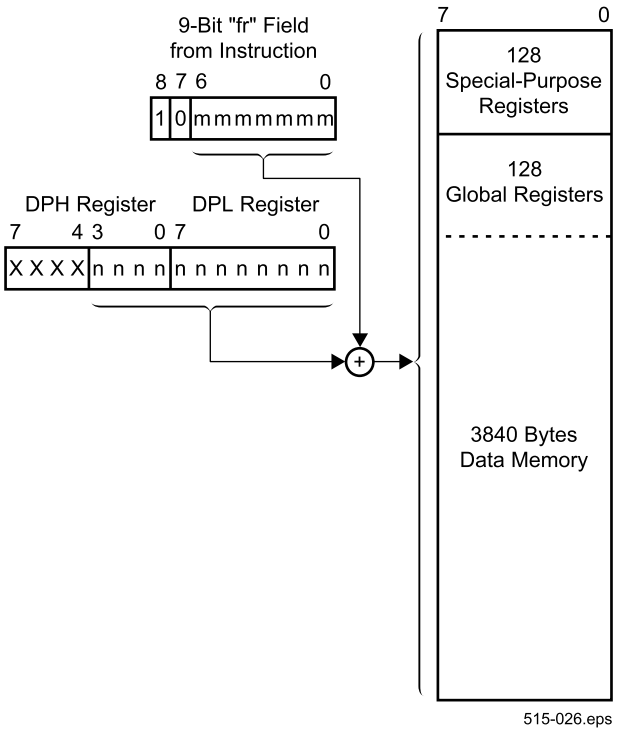


Figure 4-5 Indirect-with-Offset Mode, Data Pointer

The following code example uses indirect-with-offset mode.

```
MyStuff= 0x038D   ;define address MyStuff
loadh    MyStuff  ;load the high byte of the
                  ;DPH/DPL pointer register
                  ;with 0x03
loadl    MyStuff  ;load the low byte of the
                  ;DPH/DPL pointer register
                  ;with 0x8D
mov      w,8(dp)  ;load W with the contents of
                  ;the memory location at
                  ;effective address 0x038D
                  ;(i.e. 0x0385 + 0x0008)
```

When bit 7 of the “fr” field is set, the SPH/SPL register is selected as the pointer register. The upper four bits of the SPH register are not used. Figure 4-6 shows indirect-with-offset mode using the SPH/SPL register. In addition to this indirect-with-offset addressing mode, there are also **push** and **pop** instructions which automatically increment and decrement the SPH/SPL register while performing a data transfer between the top of stack and a data memory location specified by the “fr” field. Stacks grow down from higher addresses to lower addresses. This stack addressing mechanism is completely independent from the hardware stack used for subroutine call and return.

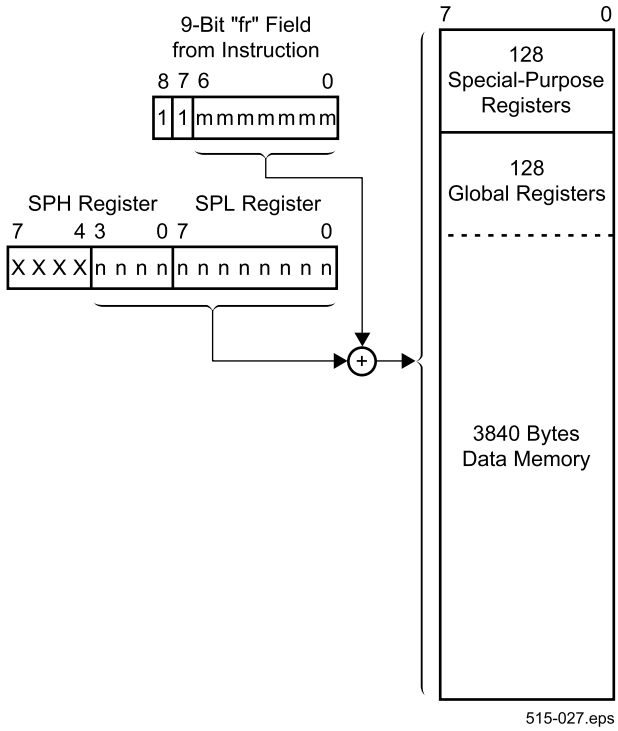


Figure 4-6 Indirect-with-Offset Mode, Stack Pointer

4.2 Instruction Set

The instruction set consists entirely of single-word (16-bit) instructions, most of which can be executed at a rate of one instruction per clock cycle, for a throughput of up to 100 MIPS when executing out of program RAM.

Assemblers may implement additional instruction mnemonics for the convenience of programmers, such as a long jump instruction which compiles to multiple IP2022 instructions for handling the page structure of program memory. Refer to the assembler documentation for more information about any instruction mnemonics implemented in the assembler.

4.2.1 Instruction Formats

There are five instruction formats:

- Two-operand arithmetic and logical instructions
- Immediate-operand arithmetic and logical instructions
- Jumps and subroutine calls
- Bit operations
- Miscellaneous instructions

Figure 4-7 shows the two-operand instruction format. The two-operand instructions perform an arithmetic or logical operation between the W register and a data memory location specified by the “fr” field. The D bit indicates the destination operand. When the D bit is clear, the destination operand is the W register. When the D bit is set, the destination operand is specified by the “fr” field.

There are some exceptions to this behavior. The multiply instructions always load the 16-bit product into the MULH and W registers. The MULH register receives the upper 8 bits, and the W register receives the lower 8 bits.

Traditionally single-operand instructions, such as increment, are available in two forms distinguished by the D bit. When the D bit is clear, the source operand is specified by the “fr” field and the destination operand is the W register. When the D bit is set, the data memory location specified by the “fr” field is both the source and destination operand.

Also, there are a few cases of unrelated instructions, such as **`clr`** and **`cmp`**, which are distinguished by the D bit.

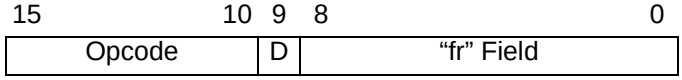


Figure 4-7 Two-Operand Instruction Format

Figure 4-8 shows the immediate operand instruction format. In this format, an 8-bit literal value is encoded in the instruction field. Usually the W register is the destination operand, however this format also includes instructions that use the top of the stack or a special-purpose register as the destination operand.

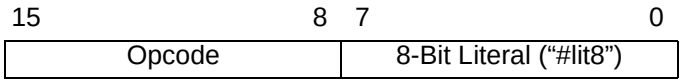


Figure 4-8 Immediate-Operand Instruction Format

Figure 4-9 shows the format of the jump and subroutine call instructions. 13 bits of the entry point address are encoded in the instruction. The remaining three bits come from the PA2:0 bits of the STATUS register.

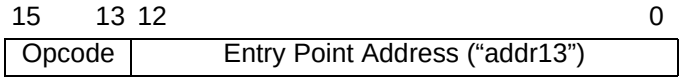


Figure 4-9 Jump and Call Instruction Format

Figure 4-10 shows the format of the instructions that clear, set, and test individual bits within registers. The register is specified by the “fr” field, and a 3-bit field in the instruction selects one of the eight bits in the register.

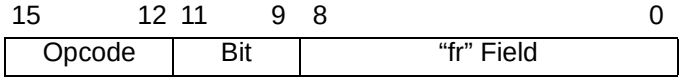


Figure 4-10 Bit Operation Instruction Format

Figure 4-11 shows the format of the remaining instructions.

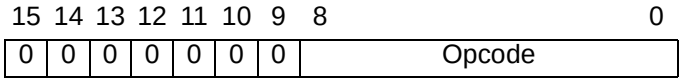


Figure 4-11 Miscellaneous Instruction Format

4.2.2 Instruction Types

The instructions are grouped into the following functional categories:

- Logical instructions
- Arithmetic and shift instructions
- Bit operation instructions
- Data movement instructions
- Program control instructions
- System control instructions

Logical Instructions

Each logic instruction performs a standard logical operation (AND, OR, exclusive OR, or logical complement) on the respective bits of the 8-bit operands. The result of the logic operation is written to W or to the data memory location specified by the “fr” field.

All of these instructions take one clock cycle for execution.

Arithmetic and Shift Instructions

Each arithmetic or shift instruction performs an operation such as add, subtract, add with carry, subtract with carry, rotate left or right through carry, increment, decrement, clear to zero, or swap high/low nibbles. The compare (**cmp**) instruction performs the same operation as subtract, but it only updates the C, DC, and Z flags of the STATUS register; the result of the subtraction is discarded.

There are instructions available that increment or decrement a register and simultaneously test the result. If the 8-bit result is zero, the next instruction in the program is skipped. These instructions can be used to make program loops. There are also compare-and-skip instructions which perform the same operation as subtract, then perform a conditional skip without affecting either operand or the condition flags in the STATUS register.

All of the arithmetic and shift instructions take one clock cycle for execution, except in the case of the test-and-skip instructions when the tested condition is true and a skip occurs, in which case the instruction takes at least two cycles. If a skip instruction is immediately followed by a **loadh**, **loadl**, or **page** instruction (and the tested condition is true) then two instructions are skipped and the operation consumes three cycles. This is useful for skipping over a conditional branch to another page, in which a **page** instruction precedes a **jmp** instruction. If several **page** or **loadh**/**loadl** instructions immediately follow a skip instruction, then they are all skipped plus the next instruction and a cycle is consumed for each. These “extended skip instructions” are interruptible, so they do not affect interrupt latency.

Bit Operation Instructions

There are four bit operation instructions:

- **setb**—sets a single bit in a data register without affecting other bits
- **clrb**—clears a single bit in a data register without affecting other bits
- **sb**—tests a single bit in a data register and skips the next instruction if the bit is set
- **snb**—tests a single bit in a data register and skips the next instruction if the bit is clear

All of the bit operation instructions take one clock cycle for execution, except for test-and-skip instructions when the tested condition is true and a skip occurs.

Data Movement Instructions

A data movement instruction moves a byte of data from a data memory location to either the W register or the top of stack, or it moves the byte from either the W register or the top of stack to a data memory location. The location is specified by the “fr” field. The SPH/SPL register pair points to the top of stack. This stack is independent of the hardware stack used for subroutine call and return.

Program Control Instructions

A program control instruction alters the flow of the program by changing the contents of the program counter. Included in this category are the jump, call, return-from-subroutine, and interrupt instructions.

The **jmp** instruction has a single operand that specifies the entry point at which to continue execution. The entry point is typically specified in assembly language with a label, as in the following code example:

```
snb    status.0  ;test the carry bit
jmp    do_carry  ;jump to do_carry routine
                 ;if C = 1


...
do_carry:        ;jump destination label
...              ;execution continues here
```

If the carry bit is set to 1, the **jmp** instruction is executed and program execution continues where the **do_carry** label appears in the program.

The **call** instruction works in a similar manner, except that it saves the contents of the program counter before jumping to the new address. It calls a subroutine that is terminated by a **ret** instruction, as shown in the following code example:

```
call    add_2bytes  ;call subroutine
                    ;add_2bytes
nop                 ;execution returns to
                    ;here after the
                    ;subroutine is finished
...
add_2bytes:         ;subroutine label
...                 ;subroutine code goes
                    ;here
ret                 ;return from subroutine
```

Returning from a subroutine restores the saved program counter contents, which causes program to resume execution with the instruction immediately following the

call instruction (a **nop** instruction, in the above example)

A program memory address contains 16 bits. The **jmp** and **call** instructions specify only the lowest thirteen bits of the jump/call address. The upper 3 bits come from the PA2:0 bits of the STATUS register. An indirect relative jump can be accomplished by adding the contents of the W register to the PCL register (i.e. an **add pcl,w** instruction).

Program control instructions such as **jmp**, **call**, and **ret** alter the normal program sequence. When one of these instructions is executed, the execution pipeline is automatically cleared of pending instructions and refilled with new instructions, starting at the new program address. Because the pipeline must be cleared, three clock cycles are required for execution, one to execute the instruction and two to reload the pipeline.

System Control Instructions

A system control instruction performs a special-purpose operation that sets the operating mode of the device or reads data from the program memory. Included in this category are the following types of instructions:

- **speed**—changes the CPU core speed (for saving power)
- **break**—enters debug mode
- **page**—writes the PA2:0 bits in the STATUS register
- **loadh**/**loadl**—loads a 16-bit pointer into the DPH and DPL registers
- **iread**—reads a word from the program memory (flash memory or RAM)
- **iwrite**—writes a word to the program RAM
- **fread**—reads the flash program memory
- **fwrite**—writes the flash program memory
- **ferase**—erases the flash program memory
- **cwdt**—clears the Watchdog Timer

4.3 Instruction Pipeline

An instruction goes through a four-stage pipeline to be executed, as shown in Figure 4-12. The first instruction is fetched from the program memory on the first clock cycle. On the second clock cycle, the first instruction is decoded and a second instruction is fetched. On the third clock cycle, the first instruction is executed, the second instruction is decoded, and a third instruction is fetched. On the fourth clock cycle, the first instruction's results are written to its destination, the second instruction is executed, the third instruction is decoded, and a fourth instruction is fetched. Once the pipeline is full, instructions are executed at the rate of one per clock cycle.

Stage	Cycle 1	Cycle 2	Cycle 3	Cycle 4
Fetch	Instruction 1	Instruction 2	Instruction 3	Instruction 4
Decode		Instruction 1	Instruction 2	Instruction 3
Execute			Instruction 1	Instruction 2
Write				Instruction 1

Figure 4-12 Pipeline Execution

Instructions that directly affect the contents of the program counter (such as jumps and calls) require that the pipeline be cleared and subsequently refilled. Therefore, these instructions take two additional clock cycles.

4.4 Subroutine Call/Return Stack

A 16-level hardware call/return stack is provided for saving the program counter on a subroutine call and restoring the program counter on subroutine return. The stack is not mapped into the data memory address space except for the top level, which is accessible as the CALLH and CALLL registers. Software can read and write these registers to implement a deeper stack, in those cases which require nesting subroutines more than 16 levels deep. This stack is completely independent of the stack used with the **push** and **pop** instructions and the SPH/SPL register pair.

When a subroutine is called, the return address is pushed onto the subroutine stack, as shown in Figure 4-13. Specifically, each saved address in the stack is moved to the next lower level to make room for the new address to be saved. Stack 1 receives the contents of the program counter. Stack 16 is overwritten with what was in Stack 15. The contents of stack 16 are lost.

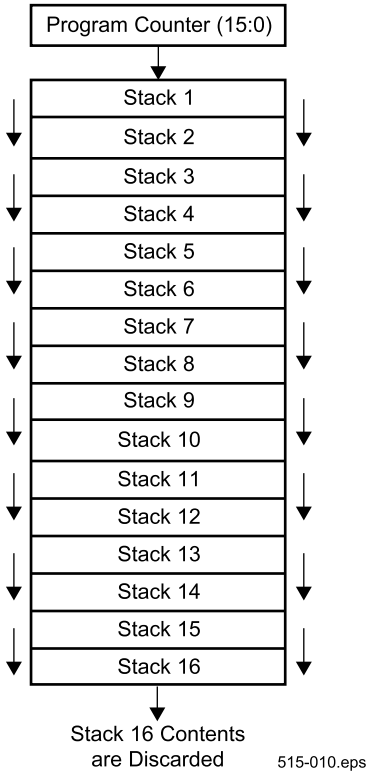


Figure 4-13 Stack Operation on Subroutine Call

When a return instruction is executed the subroutine stack is popped, as shown in Figure 4-14. Specifically, the contents of Stack 1 are copied into the program counter and the contents of each stack level are moved to the next higher level. When a value is popped off the stack, the bottom entry is initialized to 0xFFFF. For example, Stack 1 receives the contents of Stack 2, etc., until Stack 15 is overwritten with the contents of Stack 16. Stack 16 is initialized to 0xFFFF.

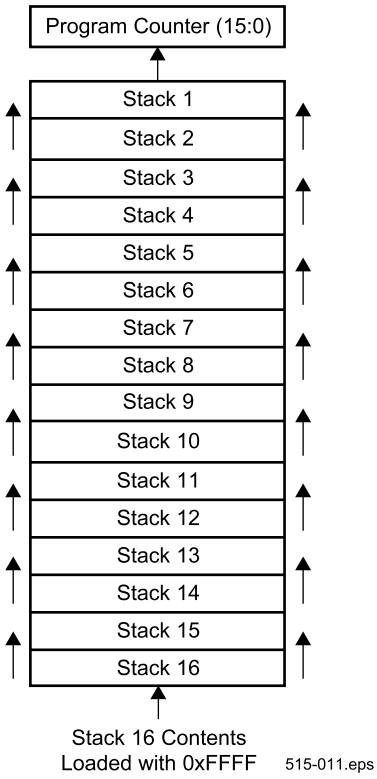


Figure 4-14 Stack Operation on Subroutine Return

For program bugs involving stack underflow, the instruction at byte address 0x1FFFE (word address 0xFFFF) can be used to jump to an appropriate handler. For example, system recovery may be possible by jumping to the reset vector at byte address 0x1FFE0 (word address 0xFFF0).

4.5 Key to Abbreviations and Symbols

Symbol	Description
W	Working register
fr	File register field (a 9-bit file register address specified in the instruction)
PCL	Virtual register for direct PC modification (global file register 0x009)
STATUS	STATUS register (global file register 0x00B)
IPH	Indirect Pointer High - Upper half of pointer for indirect addressing (global file register 0x004)
IPL	Indirect Pointer Low - Lower half of pointer for indirect addressing (global file register 0x005)
DPH	Upper half of data pointer for indirect-with-offset addressing (global file register 0x00C)
DPL	Lower half of data pointer for indirect-with-offset addressing (global file register 0x00D)
SPH	Upper half of stack pointer for indirect-with-offset addressing (global file register 0x006)
SPL	Lower half of stack pointer for indirect-with-offset addressing (global file register 0x007)
C	Carry bit in the STATUS register (bit 0)
DC	Digit Carry bit in the STATUS register (bit 1)
Z	Zero bit in the STATUS register (bit 2
BO	Brown-out bit in the STATUS register (bit 3)
WD	Watchdog Timeout bit in the STATUS register (bit 4)
PA2:PA0	Page bits in the STATUS register (bits 7:5)
WDT	Watchdog Timer counter and prescaler
f	File register address bit in opcode
k	Constant value bit in opcode
n	Numerical value bit in opcode
bit	Bit position selector bit in opcode
.	File register/bit selector separator (e.g. **`clrb status.z`**)
#	Immediate literal designator in assembly language instruction (e.g. **`mov w,#0xff`**)
#lit8	8-bit literal value in assembly language instruction
addr13	13-bit address in assembly language instruction
addr16	16-bit address in assembly language instruction

Symbol	Description
(address)	Contents of memory referenced by address
|	Logical OR
||	Concatenation
^	Logical exclusive OR
&	Logical AND
!=	inequality

4.6 Instruction Set Summary Tables

Table 4-2 through Table 4-7 list all of the IP2022 instructions, organized by category. For each instruction, the table shows the instruction mnemonic (as written in assembly language), a brief description of what the instruction does, the number of instruction cycles required for execution, the binary opcode, and the flags in the STATUS register affected by the instruction.

Although the number of clock cycles for execution is typically 1, for the skip instructions the exact number of cycles depends whether the skip is taken or not taken. Taking the skip adds 1 cycle. The effect of extended skip instructions (i.e. a skip followed by a **`loadh`**, **`loadl`**, or **`page`** instruction) is not shown.

Table 4-2 Logical Instructions

Assembler Syntax	Pseudocode Definition	Description	Cycles	Opcode	Flags Affected
`and fr,w`	fr = fr & W	AND fr,W into fr	1	`0001 011f ffff ffff`	Z
`and w,fr`	fr = W & fr	AND W,fr into W	1	`0001 010f ffff ffff`	Z
`and w,#lit8`	W = W & lit8	AND W,literal into W	1	`0111 1110 kkkk kkkk`	Z
`not fr`	fr = $\overline{fr}$	Complement fr into fr	1	`0010 011f ffff ffff`	Z
`not w,fr`	W = $\overline{fr}$	Complement fr into W	1	`0010 010f ffff ffff`	Z
`or fr,w`	fr = fr | W	OR fr,W into fr	1	`0001 001f ffff ffff`	Z
`or w,fr`	W = W | fr	OR W,fr into W	1	`0001 000f ffff ffff`	Z
`or w,#lit8`	W = W | lit8	OR W,literal into W	1	`0111 1101 kkkk kkkk`	Z
`xor fr,w`	fr = fr ^ W	XOR fr,W into fr	1	`0001 101f ffff ffff`	Z
`xor w,fr`	W = W ^ fr	XOR W,fr into W	1	`0001 100f ffff ffff`	Z
`xor w,#lit8`	W = W ^ lit8	XOR W,literal into W	1	`0111 1111 kkkk kkkk`	Z

Table 4-3 Arithmetic and Shift Instructions

Assembler Syntax	Pseudocode Definition	Description	Cycles	Opcode	Flags Affected
`add fr,w`	fr = fr + W	Add fr,W into fr	1	`0001 111f ffff ffff`	C, DC, Z
`add w,fr`	W = W + fr	Add W,fr into W	1	`0001 110f ffff ffff`	C, DC, Z
`add w,#lit8`	W = W + lit8	Add W,literal into W	1	`0111 1011 kkkk kkkk`	C, DC, Z
`addc fr,w`	fr = C + fr + W	Add carry,fr,W into fr	1	`0101 111f ffff ffff`	C, DC, Z
`addc w,fr`	W = C + W + fr	Add carry,W,fr into W	1	`0101 110f ffff ffff`	C, DC, Z
`clr fr`	fr = 0	Clear fr	1	`0000 011f ffff ffff`	Z
`cmp w,fr`	fr - W	Compare W,fr then update STATUS	1	`0000 010f ffff ffff`	C, DC, Z
`cmp w,#lit8`	lit8 - W	Compare W,literal then update STATUS	1	`0111 1001 kkkk kkkk`	C, DC, Z
`cse w,fr`	if (fr - W) = 0 then skip	Compare W,fr then skip if equal	1 or 2 (skip)	`0100 001f ffff ffff`	None
`cse w,#lit8`	if (lit8 - W) = 0 then skip	Compare W,literal then skip if equal	1 or 2 (skip)	`0111 0111 kkkk kkkk`	None
`csne w,fr`	if (fr - W) != 0 then skip	Compare W,fr then skip if not equal	1 or 2 (skip)	`0100 000f ffff ffff`	None
`csne w,#lit8`	if (lit8 - W) != 0 then skip	Compare W,literal then skip if not equal	1 or 2 (skip)	`0111 0110 kkkk kkkk`	None
`cwdt`	WDT = 0	Clear Watchdog Timer	1	`0000 0000 0000 0100`	None
`dec fr`	fr = fr - 1	Decrement fr into fr	1	`0000 111f ffff ffff`	Z

Table 4-3 Arithmetic and Shift Instructions (continued)

Assembler Syntax	Pseudocode Definition	Description	Cycles	Opcode	Flags Affected
dec w,fr	W = fr -1	Decrement fr into W	1	**0000 110f ffff ffff**	Z
decsnz fr	fr = fr - 1 if fr != 0 then skip	Decrement fr into fr then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0100 111f ffff ffff**	None
decsnz w,fr	W = fr - 1 if fr != 0 then skip	Decrement fr into W then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0100 110f ffff ffff**	None
decsz fr	fr = fr - 1 if fr = 0 then skip	Decrement fr into fr then skip if zero (STATUS not updated)	1 or 2 (skip)	**0010 111f ffff ffff**	None
decsz w,fr	W = fr - 1 if fr = 0 then skip	Decrement fr into W then skip if zero (STATUS not updated)	1 or 2 (skip)	**0010 110f ffff ffff**	None
inc fr	fr = fr + 1	Increment fr into fr	1	**0010 101f ffff ffff**	Z
inc w,fr	W = fr + 1	Increment fr into W	1	**0010 100f ffff ffff**	Z
incsnz fr	fr = fr + 1 if fr != 0 then skip	Increment fr into fr then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0101 101f ffff ffff**	None
incsnz w,fr	W = fr + 1 if fr != 0 then skip	Increment fr into W then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0101 100f ffff ffff**	None
incsz fr	fr = fr + 1 if fr = 0 then skip	Increment fr into fr then skip if zero (STATUS not updated)	1 or 2 (skip)	**0011 111f ffff ffff**	None
incsz w,fr	W = fr + 1 if fr = 0 then skip	Increment fr into W then skip if zero (STATUS not updated)	1 or 2 (skip)	**0011 110f ffff ffff**	None
muls w,fr	MULH || W = W × fr	Signed 8 × 8 multiply (bit 7 = sign) W,fr into MULH || W	1	**0101 010f ffff ffff**	None
muls w,#lit8	MULH || W = W × lit8	Signed 8 × 8 multiply (bit 7 = sign) W,literal into MULH || W	1	**0111 0011 kkkk kkkk**	None
mulu w,fr	MULH || W = W × fr	Unsigned 8 × 8 multiply W,fr into MULH || W	1	**0101 000f ffff ffff**	None
mulu w,#lit8	MULH || W = W × lit8	Unsigned 8 × 8 multiply W,literal into MULH || W	1	**0111 0010 kkkk kkkk**	None
rl fr	fr || C = C || fr	Rotate fr left through carry into fr	1	**0011 011f ffff ffff**	C
rl w,fr	W || C = C || fr	Rotate fr left through carry into W	1	**0011 010f ffff ffff**	C
rr fr	C || fr = fr || C	Rotate fr right through carry into fr	1	**0011 001f ffff ffff**	C
rr w,fr	C || W = fr || C	Rotate fr right through carry into W	1	**0011 000f ffff ffff**	C
sub fr,w	fr = fr - W	Subtract W from fr into fr	1	**0000 101f ffff ffff**	C, DC, Z
sub w,fr	W = fr - W	Subtract W from fr into W	1	**0000 100f ffff ffff**	C, DC, Z
sub w,#lit8	W = lit8 - W	Subtract W from literal into W	1	**0111 1010 kkkk kkkk**	C, DC, Z
subc fr,w	fr = fr - $\overline{C}$ - W	Subtract $\overline{carry}$,W from fr into fr	1	**0100 101f ffff ffff**	C, DC, Z
subc w,fr	W = fr - $\overline{C}$ - W	Subtract $\overline{carry}$,W from fr into W	1	**0100 100f ffff ffff**	C, DC, Z

Table 4-3 Arithmetic and Shift Instructions (continued)

Assembler Syntax	Pseudocode Definition	Description	Cycles	Opcode	Flags Affected
`swap fr`	fr = fr3:0 || fr7:4	Swap high,low nibbles of fr into fr	1	`0011 101f ffff ffff`	None
`swap w,fr`	W = fr3:0 || fr7:4	Swap high,low nibbles of fr into W	1	`0011 100f ffff ffff`	None
`test fr`	if fr = 0 then Z = 1 else Z = 0	Test fr for zero and update Z	1	`0010 001f ffff ffff`	Z

Table 4-4 Bit Operation Instructions

Assembler Syntax	Pseudocode Definition	Description	Cycles	Opcode	Flags Affected
`clrb fr.bit`	fr.bit = 0	Clear bit in fr	1	`1000 bbbf ffff ffff`	None
`sb fr.bit`	if fr.bit = 1 then skip	Test bit in fr then skip if set	1 or 2 (skip)	`1011 bbbf ffff ffff`	None
`setb fr.bit`	fr.bit = 1	Set bit in fr	1	`1001 bbbf ffff ffff`	None
`snb fr.bit`	if fr.bit = 0 then skip	Test bit in fr then skip if clear	1 or 2 (skip)	`1010 bbbf ffff ffff`	None

Table 4-5 Data Movement Instructions

Assembler Syntax	Pseudocode Definition	Description	Cycles	Opcode	Flags Affected
`mov fr,w`	fr = W	Move W into fr	1	`0000 001f ffff ffff`	None
`mov w,fr`	W = fr	Move fr into W	1	`0010 000f ffff ffff`	Z
`mov w,#lit8`	W = lit8	Move literal into W	1	`0111 1100 kkkk kkkk`	None
`push fr`	(SP) = fr SP = SP - 1	Move fr onto top of stack	1	`0100 010f ffff ffff`	None
`push #lit8`	(SP) = lit8 SP = SP - 1	Move literal onto top of stack	1	`0111 0100 kkkk kkkk`	None
`pop fr`	SP = SP + 1 fr = (SP)	Move top of stack into fr	1	`0100 011f ffff ffff`	None

Table 4-6 Program Control Instructions

Assembler Syntax	Description	Cycles	Opcode	Flags Affected
call addr13	Call subroutine	3	**110k kkkk kkkk kkkk**	None
jmp addr13	Jump	3	**111k kkkk kkkk kkkk**	None
int	Software interrupt	3	**0000 0000 0000 0110**	None
nop	No operation	1	**0000 0000 0000 0000**	None
ret	Return from subroutine	3	**0000 0000 0000 0111**	PA2:0
reti #lit3	Return from interrupt	3	**0000 0000 0000 1nnn**	All
retw #lit8	Return from subroutine with literal into W	3	**0111 1000 kkkk kkkk**	PA2:0

Table 4-7 System Control Instructions

Assembler Syntax	Description	Cycles	Opcode	Flags Affected
break	Software breakpoint	1	**0000 0000 0000 0001**	None
ferase	Erase flash block	1	**0000 0000 0000 0011**	None
fread	Read from flash memory	1	**0000 0000 0001 1011**	None
fwrite	Write into flash memory	1	**0000 0000 0001 1010**	None
iread	Read from program memory	4	**0000 0000 0001 1001**	None
iwrite	Write into program RAM	4	**0000 0000 0001 1000**	None
loadh addr16	Load high data address into DPH	1	**0111 0000 kkkk kkkk**	None
loadl addr16	Load low data address into DPH	1	**0111 0001 kkkk kkkk**	None
page addr16	Load page bits from program address	1	**0000 0000 0001 0nnn**	PA2:0
speed #lit8	Change CPU speed from literal	1	**0000 0001 nnnn nnnn**	None

4.7 Self-Programming Instructions

The IP2022 has several instructions used to read and write the program RAM and the program flash memory. These instructions allow the program flash memory to be read and written through special-purpose registers in the data memory space, which allows the flash memory to be used to store both program code and data.

Because no special programming voltage is required to write to the flash memory, any application may take advantage of this feature at run-time. Typical uses include saving phone numbers and passwords, downloading new or updated software, and logging infrequent events such as errors and Watchdog Timer overflow.

The self-programming instructions are not affected by the code-protection flag (the $\overline{CP}$ bit of the FUSE1 register), so the entire program memory is visible to any software running on the IP2022.

There are five instructions used for self-programming, as shown in Table 4-8. Certain uses of the instructions are not valid, and the instruction is executed as though it were a **nop** instruction (i.e. the program counter is incremented, but no other registers or bits are affected).

Table 4-8 Instructions Used for Self-Programming

Instruction	Stops Other Instructions From Executing?	Execution Time (Cycles)	Executed From RAM to Operate On Program RAM	Executed From RAM to Operate On Flash Memory	Executed From Flash to Operate On Program RAM	Executed From Flash to Operate On Flash Memory	Executed While FWP Bit in XCFG Register is Clear (FWP = 0)
iread	Yes	4	Valid	Valid if CPU core speed ≤ 40 MHz	Valid	Valid	Valid
iwrite	Yes	4	Valid	**nop**	Valid	**nop**	Valid
fread	No	1	**nop**	Valid	**nop**	**nop**	Valid
fwrite	No	1	**nop**	Valid	**nop**	**nop**	**nop**
ferase	No	1	**nop**	Valid	**nop**	**nop**	**nop**

The **iread** and **iwrite** instructions are blocking (i.e. no other instructions execute until they complete), and they require 4 cycles to execute. The **fread**, **fwrite**, and **ferase** instructions are non-blocking and require only 1 cycle, however these instructions only initiate an operation which is not complete until indicated by the FBUSY bit in the XCFG register being cleared.

Read and write operations that involve program memory use two registers. The DATAH/DATAL register is a 16-bit data buffer used for loading or unloading data in program memory. The ADDRH/ADDRL register provides a word address used to specify a location in program memory. Unlike the address pointer used for in-system programming, the ADDRH/ADDRL register is not automatically incremented by memory read or write operations. Like the other pointer registers (i.e. IPH/IPL, DPH/DPL, and SPH/SPL), an increment or addition instruction to the low byte of the register will cause the high byte to be incremented if carry occurs. A decrement or subtraction instruction to the low byte of the register will cause the high byte to be decremented if borrow occurs.

While executing from flash memory, the CPU speed is automatically throttled down (if necessary) so that instruction fetches do not exceed the flash memory access time. As a side effect, this ensures that an **iread** instruction executed from flash memory will not exceed the flash memory access time.

Software can use the FBUSY bit to check that a previous flash memory operation has completed before executing another **fread**, **fwrite**, or **ferase** instruction, before jumping or calling program code in flash memory, and before changing the CPU core speed. Software must not attempt to execute from flash memory while the FBUSY bit is set, because the flash memory is unreadable during that time. Code which uses the **fread**, **fwrite**, or **ferase** instructions must execute from program RAM, not flash memory. The CPU core speed must not change while a **fread**, **fwrite**, or **ferase** operation is in progress.

It is not necessary to check the FBUSY bit if the flash operation is known to complete in time. For example, while executing from program RAM at 100 MHz, the **fread** instruction will require 3 cycles to complete, as shown below.

```
fread           ;start the read
nop             ;1 cycle
nop             ;2 cycles
nop             ;3 cycles
mov    w,datal  ;data is known to be available
```

Unlike RAM, flash memory requires an explicit erase operation before being written. The **ferase** instruction is used to erase a 512-byte block of flash memory. After the block has been erased, individual words can be written with the **fwrite** instruction. The self-programming instructions have no access to the flash memory bits used to implement the configuration block.

4.7.1 Flash Timing Control

The FCFG register controls the timing of flash memory operations. Figure 4-15 shows the FCFG register.

7	6	5	4	3	0
FRDTS1:0		FRDTC1:0		FWRT3:0	

Figure 4-15 FCFG Register

- *FRDTS1:0*—the CPU core clock while executing from flash program memory must not exceed 40 MHz, otherwise unreliable operation may result. The IP2022 automatically increases the number of system clock cycles for each CPU core clock cycle when executing from flash, but it is the responsibility of software to load the FRDTS1:0 bits appropriately for the operating frequency, as shown in Table 4-9. The actual speed will be the slower of the speed indicated in the SPDREG register and the speed specified by the FRDTS1:0 bits.

Table 4-9 Flash Execution Timing

System Clock Frequency	System Clock Cycles For Each CPU Core Clock Cycle	FRDTS1:0
0–40 MHz	1	00
40–80 MHz	2	01
80–120 MHz	3	10
120–160 MHz	4	11

- *FRDTC1:0*—the number of CPU core cycles for reading the flash memory with an **`fread`** instruction must be specified to prevent the flash memory access time from being exceeded. Because the CPU core speed may change, the value programmed in these bits should be appropriate for the fastest speed that might be used (typically, the faster of the mainline code and the interrupt service routine). The FRDTC1:0 bits specify the number of CPU core clock cycles required for read access, as shown in Table 4-10.

Table 4-10 Flash Read Timing

CPU Core Clock Frequency	Clock Cycles Per Flash Access	FRDTC1:0
0–40 MHz	1	00
40–80 MHz	2	01
80–120 MHz	3	10
120–160 MHz	4	11

- *FWRT3:0*—the flash memory erase and write timing is derived from the CPU core clock through a programmable divider, which is controlled by the FWRT3:0 bits. The time base must be 1 to 2 microseconds. Below 1 microsecond, the flash memory will be underprogrammed, and data retention is not guaranteed. Above 2 microseconds, the flash memory will be overprogrammed, and reliability is not guaranteed. Because the minimum flash write clock divisor is 2, the minimum clock frequency for self-programming is 1 MHz. The range of values for FWRT3:0 is shown in Table 4-16.

Figure 4-16 Flash Write Timing

CPU Core Clock Frequency	Flash Write Clock Divisor	FWRT3:0
1–2 MHz	2	0000
2–3 MHz	3	0001
3–4 MHz	4	0010
4–6 MHz	6	0011
6–8 MHz	8	0100
8–12 MHz	12	0101
12–16 MHz	16	0110
16–24 MHz	24	0111
24–32 MHz	32	1000
32–48 MHz	48	1001
48–64 MHz	64	1010
64–96 MHz	96	1011
96–128 MHz	128	1100
128–192 MHz	192	1101
	256	1110
	384	1111

4.7.2 Interrupts During Flash Operations

Before starting a flash write or erase operation, the flash write timing compensation must be set up properly for the current speed. The CPU core clock is the time base for the flash write timing compensation, so it is critical that the CPU core clock speed is not changed during a flash write or erase operation. Interrupts may be taken during a flash write or erase operation, if the INTSPD register is set up so the speed does not change when an interrupt occurs.

If the flash read timing compensation is set up for a clock divisor of 1 (i.e. fastest speed), interrupts will not cause

fread instructions to fail, so no special precautions need to be taken to avoid violating the flash read access time.

5.0 Peripherals

The IP2022 provides an array of on-chip peripherals needed to support a broad range of embedded Internet applications:

- Watchdog timer
- Real-time timer
- 2 General-purpose timers with compare and capture Registers
- 2 Serializer/Deserializer channels
- 10-bit, 8-channel A/D converter
- Analog comparator
- Parallel slave peripheral interface

All of the peripherals except the Watchdog Timer and the Real-Time Timer use alternate functions of the I/O port pins to interface with external signals.

5.1 I/O Ports

The IP2022 contains one 4-bit I/O port (Port A) and six 8-bit I/O ports (Port B through Port G) and. The four Port A pins have 24 mA current drive capability. All the ports have symmetrical drive. Inputs are 5V-tolerant CMOS levels. Outputs can use the same 2.3–2.7V power supply used for the CPU core and peripheral logic, or they can use a higher voltage (up to 3.6V). The IOVdd pins are provided for those applications in which a separate power supply is used for the I/O port pin output drivers. Port G has a separate GVdd pin which can be used to run the Port G output drivers at a voltage different from that used for the other ports, since Port G must run from a 2.3–2.7V power supply.

Each port has separate input (RxIN), output (RxOUT), and direction (RxDIR) registers, which are memory mapped. The numbers in the pin names correspond to the bit positions in these registers. These registers allow each port bit to be individually configured as a general-purpose input or output under software control.

Port B has three additional registers for supporting external interrupts, the RBINTED (Section 5.1.5), RBINTF (Section 5.1.6), and RBINTE (Section 5.1.7) registers. For more information about the Port B interrupts, see Section 2.5.1.

In addition, each port pin has an alternate function used to support the on-chip hardware peripherals, as listed in Table 5-1. Port A and Port B support the multi-function timers Timer 1 and Timer 2. Port B, Port C, and Port D support the Parallel Slave Peripheral (PSP) functions. Port E and Port F support the Serializer/Deserializers. Port G supports the Analog to Digital Converter (ADC) and the Analog Comparator. Before enabling a hardware peripheral, configure the port pins for input or output as required by the peripheral.

Table 5-1 I/O Port Pin Alternate Functions

Name	Pin	Alternate Function
RA0	5	High Power Output, Timer 1 Capture 1 Input (T1CPI1 pin)
RA1	6	High Power Output, Timer 1 Capture 2 Input (T1CPI2 pin)
RA2	7	High Power Output, Timer 1 Clock Input (T1CLK pin)
RA3	8	High Power Output, Timer 1 Output (T1OUT pin)
RB0	13	External Interrupt, Timer 2 Capture 1 Input (T2CPI1 pin)
RB1	14	External Interrupt, Timer 2 Capture 2 Input (T2CPI2 pin)
RB2	15	External Interrupt, Timer 2 Clock Input (T2CLK pin)
RB3	16	External Interrupt, Timer 2 Output (T2OUT pin)
RB4	17	External Interrupt
RB5	18	External Interrupt, Parallel Slave Peripheral $\overline{\text{HOLD}}$ output
RB6	19	External Interrupt, Parallel Slave Peripheral R/W input
RB7	20	External Interrupt, Parallel Slave Peripheral $\overline{\text{CS}}$ input
RC0	21	Parallel Slave Peripheral Data
RC1	22	Parallel Slave Peripheral Data
RC2	23	Parallel Slave Peripheral Data
RC3	24	Parallel Slave Peripheral Data
RC4	25	Parallel Slave Peripheral Data
RC5	26	Parallel Slave Peripheral Data
RC6	27	Parallel Slave Peripheral Data
RC7	28	Parallel Slave Peripheral Data
RD0	29	Parallel Slave Peripheral Data

Table 5-1 I/O Port Pin Alternate Functions (continued)

Name	Pin	Alternate Function
RD1	30	Parallel Slave Peripheral Data
RD2	35	Parallel Slave Peripheral Data
RD3	36	Parallel Slave Peripheral Data
RD4	37	Parallel Slave Peripheral Data
RD5	38	Parallel Slave Peripheral Data
RD6	39	Parallel Slave Peripheral Data
RD7	40	Parallel Slave Peripheral Data
RE0	41	Serial 1 Clock (S1CLK pin)
RE1	42	Serial 1 Receive Plus-Side Data (S1RXP pin)
RE2	43	Serial 1 Receive Minus-Side Data (S1RXM pin)
RE3	44	Serial 1 Receive Data (S1RXD pin)
RE4	45	Serial 1 Transmit Plus-Side Pre-Emphasis Data/Output Enable (S1TXPE/S1OE pin)
RE5	46	High Power Output, Serial 1 Transmit Plus-Side Data (S1TXP pin)
RE6	47	High Power Output, Serial 1 Transmit Minus-Side Data (S1TXM pin)
RE7	48	Serial 1 Transmit Minus-Side Pre-Emphasis Data (S1TXME pin)
RF0	49	Serial 2 Transmit Plus-Side Pre-Emphasis Data/Output Enable (S2TXPE/S2OE pin)
RF1	50	High Power Output, Serial 2 Transmit Plus-Side Data (S2TXP pin)
RF2	51	High Power Output, Serial 2 Transmit Minus-Side Data (S2TXM pin)
RF3	52	Serial 2 Transmit Minus-Side Pre-Emphasis Data (S2TXME pin)
RF4	57	Serial 2 Clock (S2CLK pin)
RF5	58	Serial 2 Receive Plus-Side Data (S2RXP pin)
RF6	59	Serial 2 Receive Minus-Side Data (S2RXM pin)
RF7	60	Serial 2 Receive Data (S2RXD pin)
RG0	61	ADC0 Input, Analog Comparator Output
RG1	62	ADC1 Input, Analog Comparator + Input
RG2	63	ADC2 Input, Analog Comparator – Input
RG3	64	ADC3 Input, ADC reference Input
RG4	66	ADC4 Input
RG5	67	ADC5 Input
RG6	68	ADC6 Input
RG7	69	ADC7 Input

Figure 5-1 shows the internal hardware structure and configuration registers for each pin of a port.

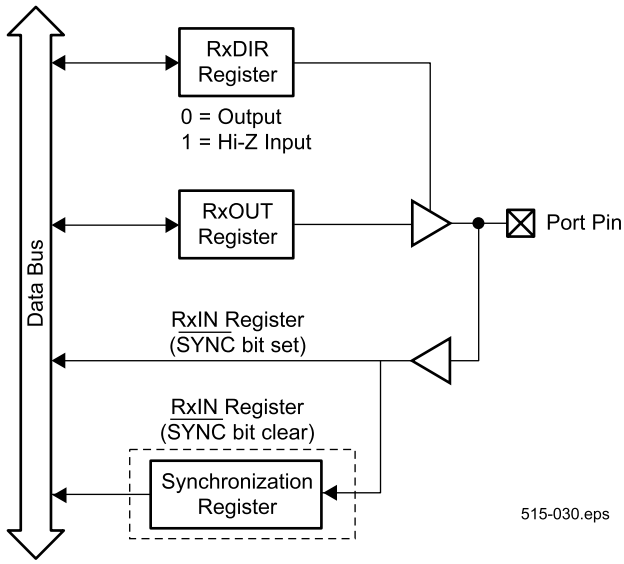


Figure 5-1 Port Pin Block Diagram

5.1.1 Reading and Writing the Ports

The port registers are memory-mapped into the data memory address space between 0x020 and 0x03A. In addition, Port B has three extra registers located between 0x017 and 0x019 which are used to support external interrupt inputs.

When two successive read-modify-write instructions read and write the same I/O port with a very high clock rate, the “write” part of one instruction might not occur soon enough before the “read” part of the very next instruction, resulting in the second instruction getting “old” data. To ensure predictable results, avoid using two successive read-modify-write instructions that access the same register if

the clock rate is high, or insert 2 **nop** instructions between successive read-modify-write instructions (if the SYNC bit in the FUSE1 register is clear, 3 **nop** instructions are required). When reading from the RxOUT register rather than the RxIN register, the CPU reads data from a register instead of the port pins. In this case, the **nop** instructions are not required.

5.1.2 RxIN Registers

The RxIN registers are virtual registers that provide read-only access to the physical I/O pins. Reading these registers returns the states on the pins, which may be driven either by the IP2022 or an external device. If the SYNC bit in the FUSE1 register is clear, the states are read from a synchronization register. If an application reads data from a device running asynchronously to the IP2022, the SYNC bit should be cleared to avoid the occurrence of metastable states (i.e. corrupt data caused by an input which fails to meet the setup time before the sampling clock edge, which theoretically could interfere with the operation of the CPU).

5.1.3 RxOUT Registers

The RxOUT registers are data output buffer registers. The data in these registers is driven on any I/O pins that are configured as outputs. On reads, the RxOUT registers return the data previously written to the data output buffer registers, which might not correspond to the states actually present on pins configured as inputs or pins forced to another state by an external device.

5.1.4 RxDIR Registers

The RxDIR registers select the direction of the port pins. For each output port pin, clear the corresponding RxDIR bit. For each input port pin, set the corresponding RxDIR bit. Unused pins that are left open-circuit should be configured as outputs, to keep them from floating.

For example, to configure Port A pins RA3 and RA2 as outputs and RA1 and RA0 as inputs, the following code could be used:

```
mov   w,#0x03   ;load W with the value 0x03
                ;(bits 3:2 low, and bits 1:0
                ;high)
mov   0x022,w   ;write 0x03 to RADIR
                ;register
```

The second move instruction in this example writes the RADIR register, located at address 0x022. Because Port A has only four I/O pins, only the four least significant bits of this register are used.

To drive the RA1 pin low and the RA0 pin high, the following code then could be executed:

```
mov   w,#0x01   ;load W with the value 0x01
                ;(bits 3:1 low, and bit 0
                ;high)
mov   0x021,w   ;write 0x01 to RAOUT
                ;register
```

The second move instruction shown above writes the RAOUT register, located at address 0x021. When reading the Port A pins through the RAIN register (0x020), the upper four bits always read as zero.

When a write is performed to the RxOUT register of a port pin that has been configured as an input, the write is performed but it has no immediate effect on the pin. If that pin is later configured as an output, the pin will be driven with the data that had been previously written to the RxOUT register.

5.1.5 RBINTED Register

The RBINTED register consists of 8 edge detection bits that correspond to the 8 pins of Port B. A set bit in the RBINTED register makes the corresponding port pin trigger on falling edges, while a clear bit makes the pin trigger on rising edges.

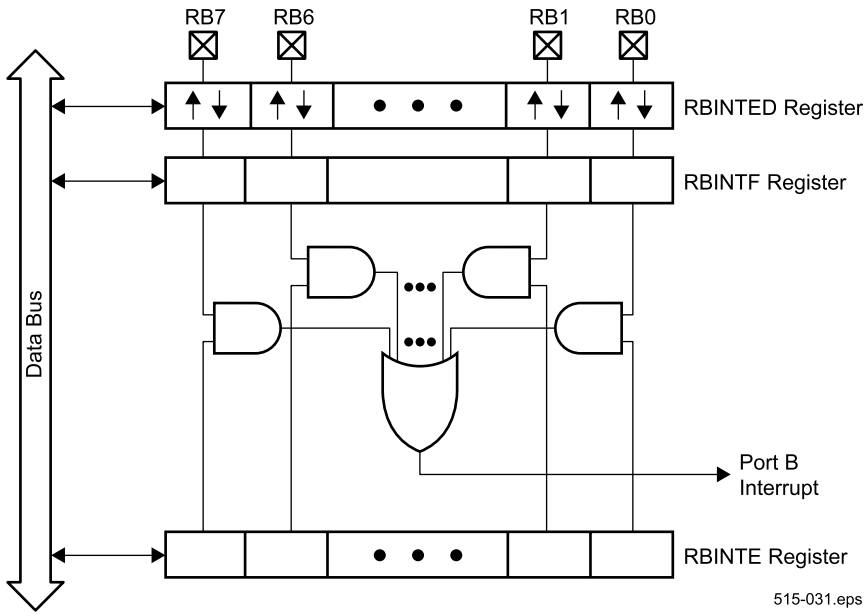


Figure 5-2 Port B Interrupt Registers

5.1.6 RBINTF Register

The RBINTF register consists of 8 interrupt flags that correspond to the 8 pins of Port B. If the trigger condition for a Port B pin occurs, the corresponding bit in the RBINTF register is set. The bit is set even if the port pin is not enabled as a source of interrupts.

The interrupt service routine (ISR) can check this register to determine the source of an external interrupt. If a Port B pin enabled for generating interrupts has a set bit in the RBINTF register, software must clear the bit prior to exiting to prevent repeated calls to the ISR.

The Port B interrupt logic is asynchronous (e.g. functions without a clock in clock-stop mode). A side effect is that there is a 2-cycle delay between the instruction that clears a RBINTF bit and the bit being cleared. This means that software must clear the bit at least 2 cycles before executing a return from interrupt (**`reti`**) instruction.

5.1.7 RBINTE Register

The RBINTE register consists of 8 interrupt enable bits that correspond to the 8 pins of Port B. A Port B pin is enabled as a source of interrupts by setting the corresponding bit in the RBINTE register. The pin is disabled as an interrupt source by clearing the corresponding RBINTE bit.

5.1.8 Port Configuration Upon Power-Up

On power-up, all the port control registers (RxDIR) are initialized to 0xFF. Therefore, each port pin is configured as a high-impedance input. This prevents any false signalling to external components which could occur if the ports were allowed to assume a random configuration at power-up.

5.2 Timer 0

Timer 0 is an 8-bit timer intended to generate periodic interrupts for ipModules that require being called at a constant rate, such as UART and DTMF functions. This use is supported in the instruction set by an option for the **`reti`** instruction which adds the W register to the T0TMR register when returning from an interrupt. Figure 5-3 shows the Timer 0 logic.

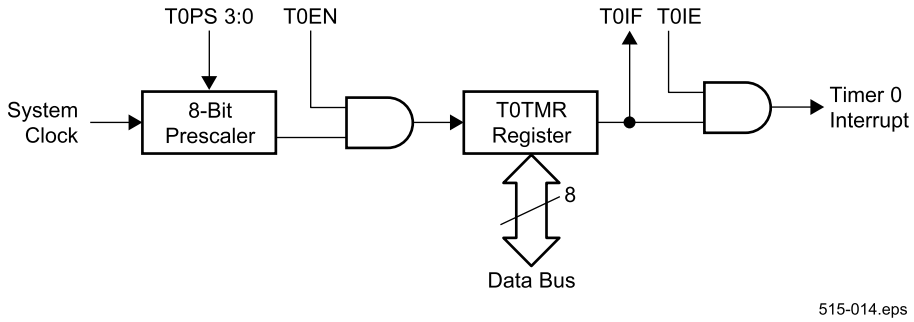


Figure 5-3 Timer 0 Block Diagram

The prescaler divisor is controlled by the T0PS3:0 bits of the T0TMR register, as shown in Table 5-2.

Table 5-2 Prescaler Divisor

TOPS3:0	Divisor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Timer 0 is readable and writable as the T0TMR register. The control and status register for Timer 0 is the T0CFG register, as shown in Figure 5-4.

7	6 … 3	2	1	0
T0EN	T0PS3:0	Reserved	T0IE	T0IF

Figure 5-4 T0CFG Register

- *T0EN*—set to enable Timer 0, clear to disable. When Timer 0 is disabled, clocking is inhibited to save power.
- *T0PS3:0*—prescaler divisor, as described in Table 5-2.
- *T0IE*—set to enable Timer 0 overflow interrupts, clear to disable interrupts.
- *T0IF*—set on the occurrence of a Timer 0 overflow.

5.3 Real-Time Timer

The Real-Time Timer is an 8-bit timer intended to provide a periodic system wake-up interrupt. Unlike the other peripherals (except the Watchdog Timer), the Real-Time Timer continues to function when the system clock is disabled. For those applications which spend much of their time with the OSC clock oscillator turned off to conserve power, there are only two available mechanisms for a periodic wake-up: reset from the Watchdog Timer and interrupt from the Real-Time Timer. By using an interrupt rather than reset, more of the CPU state is preserved and some reset procedures such as initializing the port direction registers can be skipped.

Figure 5-5 shows the Real-Time Timer logic.

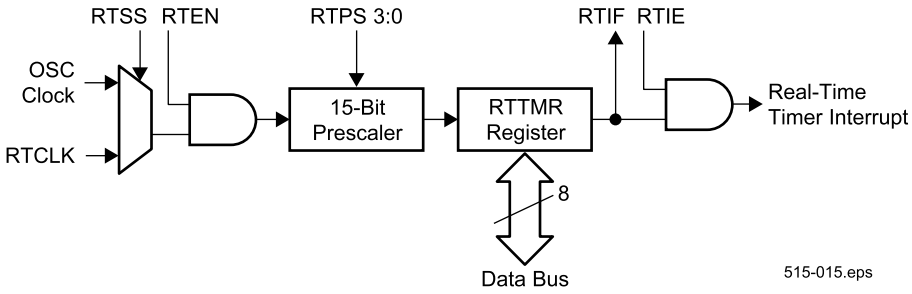


Figure 5-5 Real-Time Timer Block Diagram

The real-time timer is readable and writable as the RTTMR register. The control and status register for the timer is the RTCFG register, as shown in Figure 5-6.

7	6	3	2	1	0
RTEN	RTPS3:0		RTSS	RTIE	RTIF

Figure 5-6 RTCFG Register

- *RTEN*—set to enable the Real-Time Timer, clear to disable. When disabled, clocking is inhibited to save power.
- *RTPS3:0*—prescaler divisor, as shown in Table 5-3.
- *RTSS*—selects the clock source. Set for RTCLK, clear for the pre-PLL clock.
- *RTIE*—set to enable Real-Time Timer overflow interrupts, clear to disable interrupts.
- *RTIF*—set on Real-Time Timer overflow.

Table 5-3 Real-Time Timer Prescaler Divisor

RTPS3:0	Divisor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

The $\overline{\text{RTOES}}$ bit in the XCFG register selects the sampling mode for the external input.

If the $\overline{\text{RTOES}}$ bit is clear, the external input is oversampled with the system clock. The CPU can always read the value in the RTTMR register, however, the system clock must be at least twice the frequency of the external input. If the system clock source is changed to RTCLK or turned off, then the $\overline{\text{RTOES}}$ bit must be set for the Real-Time Timer to function.

If the $\overline{\text{RTOES}}$ bit is set then the external input directly clocks the Real-Time Timer (i.e. RTCLK is not oversampled). The Real-Time Timer will always function whether the clock input is synchronous or asynchronous. However, the CPU cannot reliably read the value in the RTTMR register unless the RTCLK clock is synchronous to the system clock.

If the value in the RTTMR register does not need to be used by the CPU (i.e. only the interrupt flag is of interest) then the $\overline{\text{RTOES}}$ bit can be set (i.e. RTCLK not oversampled) which allows the Real-Time Timer to function for any configuration of the system clock.

If the value in the RTTMR register needs to be used by the CPU, but the Real-Time Timer is not required to function when the system clock is set to RTCLK or turned off, then the $\overline{\text{RTOES}}$ bit should be cleared to ensure the CPU can reliably read the RTTMR register.

If the value in the RTTMR register needs to be used by the CPU and the Real-Time Timer is required to function

when the system clock is set to RTCLK or off, then software must change the RTOES bit when changing the system clock source. To read the RTTMR register when the system clock is not synchronous to the RTCLK, the RTOES bit must be clear to ensure reliable operation. Before the system clock is changed to RTCLK or turned off, the RTOES bit must be set (i.e. RTCLK not oversampled) for the Real-Time Timer to continue to function.

5.4 Multi-Function Timers

The IP2022 contains two independent 16-bit multi-function timers, called T1 and T2. These versatile, programmable timers reduce the software burden on the CPU in real-time control applications such as PWM generation, motor control, triac control, variable-brightness display control, sine-wave generation, and data acquisition.

Each timer consists of a 16-bit counter register supported by a dedicated 16-bit capture register and two 16-bit compare registers. The second compare register can also serve as capture register. Each timer may use up to four external pins: TxCPI1 (Capture Input), TxCPI2 (Capture Input), TxCLK (Clock Input), TxOUT (Output). These pins are multiplexed with general-purpose I/O port pins. The port direction register has priority over the timer configuration, so the port direction register must be programmed appropriately for each of these four signals if their associated timer functions are used.

Figure 5-7 is a block diagram showing the registers and I/O pins of one timer. Each timer is based on a 16-bit counter/timer driven by a 15-bit prescaler. The input of the prescaler can be either the system clock or an external clock signal which is internally synchronized to the system clock. The counter cannot be directly written by software, but it may be cleared by writing to the TxRST bit in the TxCTRL register.

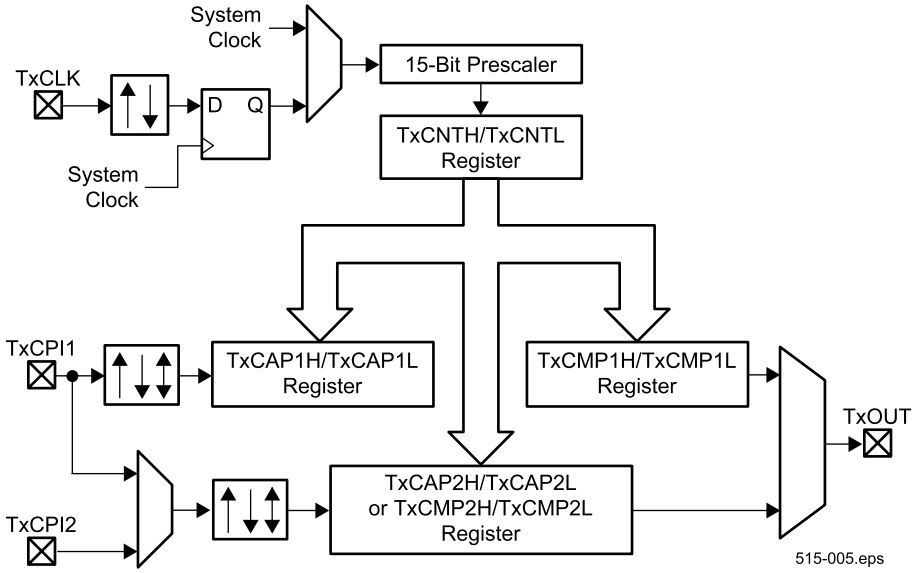


Figure 5-7 Multifunction Timer Block Diagram

5.4.1 Operating Modes

Each timer can be configured to operate in one of the following modes:

- Pulse-Width Modulation (PWM)
- Timer
- Capture/Compare

PWM Mode

In PWM Mode, the timer can generate a pulse-width modulated signal on its output pin, TxOUT. The period of the PWM cycle (high + low) is specified by the value in the TxCMP2 register. The high time of the pulse is specified by the value in the TxCMP1 register.

PWM mode can be used to generate an external clock signal that is synchronous to the IP2022 system clock. For

example, by loading TxCMP1 with 1 and TxCMP2 with 2, a symmetric 50-MHz external clock can be generated from a 100 MHz system clock. In some applications, this can eliminate crystals or oscillators required to produce clock signals for other components in the system.

The 16-bit counter/timer counts upward. After reaching the value stored in the TxCMP1 register minus one, at the next clock edge the TxOUT pin is driven low. The counter/timer is unaffected by this event and continues to increment. After reaching the value stored in the TxCMP2 register minus one, at the next clock edge the timer is cleared. When the counter is cleared, the TxOUT output is driven high, *unless* the TxCMP1 register is clear, in which case the TxOUT pin is driven low.

There are two special cases. When the TxCMP1 register is clear, the TxOUT pin is driven with a continuous low, corresponding to a duty-cycle of 0%. When the value in the TxCMP1 register pair is equal to the value in the TxCMP2 register pair, the TxOUT output is driven with a continuous high, corresponding to a duty-cycle of 100%.

The behavior of the timers when the value in the TxCMP1 register are greater than the value in the TxCMP2 register is undefined.

The timer is glitch-free no matter when the TxCMP1H/TxCMP1L register pair or the TxCMP2H/TxCMP2L register pair are changed relative to the value of the internal counter/timer. The new duty cycle or period values do not take effect until the current PWM cycle is completed (the counter/timer is reset).

Interrupts, if enabled through the TxCFG1 register, can be generated whenever the timer output is set or cleared. If the TxCMP1 register is clear, or if the value in the TxCMP1 register is equal to the value in the TxCMP2 register, an interrupt is generated each time the counter/timer is reset to zero.

In PWM mode, the Capture 1 input remains active (if enabled by the CPI1EN bit in the TxCFG1 register) and, when triggered, captures the current counter/timer value into the TxCAP1 register.

The multifunction timers can be configured to interrupt on a Capture 1 event and reset the counter/timer on the event. For PWM operation without Capture 1, software must disable the Capture 1 input by clearing the CPI1EN bit in the TxCFG1 register.

Timer Mode

This is not a separate timer mode (from the hardware point of view), but is a conceptual mode for programmers. It is the PWM mode, except that software disables the timer output by clearing the OEN bit in the TxCFG register.

Capture/Compare Mode

In Capture/Compare mode, one or both of the timer capture inputs (TxCPI1 and TxCPI2) may be used. Their pin functions must be enabled in the TxCFG1 register. Each capture input can be programmed in the TxCFG2 register to trigger on a rising edge, falling edge, or both rising and falling edges.

When a trigger event occurs on either capture pin, the current value of the counter/timer is captured into the TxCAP1H/TxCAP1L register pair or the TxCAP2H/TxCAP2L register pair for that input pin.

The counter/timers can also be configured to reset on a TxCP1 input event, in which case the value of the counter/timer before it was reset is captured in the TxCAP1H/TxCAP1L register pair and the counter/timer is reset to zero. This mode is useful for measuring the frequency (or width) of external signals. By using both capture inputs and configuring them for opposite edges, the duty cycle of a signal can also be measured. To avoid wasting I/O port pins in this configuration, the CPI2CPI1 bit in the TxCFG1 register is provided to internally tie the TxCPI1 and TxCPI2 inputs together, which frees the TxCPI2 pin to be used as a general-purpose I/O port pin.

An interrupt can be generated for any capture event and for counter/timer overflows.

This mode also features an output-compare function. The TxCMP1H/TCMP1L register pair is constantly compared against the internal counter/timer. When the counter/timer reaches the value of the TxCMP1H/TxCMP1L register pair minus one, at the next counter clock the TxOUT output is toggled. The TxOUT output, if enabled via the OEN bit, can be driven high or low by writing to the TOUTSET and TOUTCLR bits in the TxCFG2 register. An interrupt can be enabled for this event.

5.4.2 Timer Pin Assignments

The following table lists the I/O port pins associated with the Timer T1 and Timer T2 I/O functions.

Table 5-4 Timer T1/T2 Pin Assignments

I/O Pin	Timer T1/T2 Function
RA0	Timer T1 Capture 1 Input
RA1	Timer T1 Capture 2 Input
RA2	Timer T1 External Event Clock Source
RA3	Timer T1 Output
RB0	Timer T2 Capture 1 Input
RB1	Timer T2 Capture 2 Input
RB2	Timer T2 External Event Clock Source
RB3	Timer T2 Output

5.4.3 Timer Registers

Each timer has six 16-bit register pairs, which are accessed as 8-bit registers in the special-purpose register space. There is also one 8-bit register shared by both timers.

TxCNTH/TxCNTL Register

The TxCNTH/TxCNTL register pair indicates the value of the counter/timer and increments synchronously with the rising edge of the system clock. This register is read-only. The timer counter may be cleared by writing to the TxRST bit in the TCTRL register.

Reading the TxCNTL register returns the least-significant 8 bits of the internal TxCNT counter and causes the most-significant 8 bits of the counter to be latched into the TxCNTH register. This allows software to read the TxCNTH register later and still be assured of atomicity.

TxCAP1H/TxCAP1L Register

The TxCAP1H/TxCAP1L register pair captures the value of the counter/timer when the TxCPI1 input is triggered. This register is read-only.

Reading the TxCAP1L register returns the least-significant 8 bits of an internal capture register and causes the most-significant 8-bits of the register to be latched into the TxCAP1H register. This allows software to read the TxCAP1H register later and still be assured of atomicity.

TxCMP1H/TxCMP1L Register

In Capture/Compare mode, the TxOUT output pin is toggled (if enabled by the OEN bit in the TxCFG1 register) when the counter/timer increments to the value in the TxCMP1 register. In this mode, the value written to the TxCMP1 register takes effect immediately.

Writing to the TxCMP1L register causes the value to be stored in the TxCMP1L register with no other effect. Writing to the TxCMP1H register causes an internal compare register to be loaded with a 16-bit value in which the low 8 bits come from the TxCMP1L register and high 8 bits come from the value being written to the TxCMP1H register. Software should write the TxCMP1L register before writing the TxCMP1H register, because writing to the TxCMP1H register is used as an indication that a new compare value has been written. Writing to the TxCMP1H register is required for the new compare value to take effect. In PWM mode, the 16-bit number latched into the internal compare register by writing to the TxCMP1H register does not take effect until the end of the current PWM cycle.

Reading the TxCMP1H or TxCMP1L registers returns the previously written value whether or not the value stored in these registers has been transferred to the internal compare register by writing to the TxCMP1H register.

TxCAP2H/TxCAP2L or TxCMP2H/TxCMP2L Register

This register pair is called the TxCAP2H/TxCAP2L registers in Capture/Compare mode and the TxCMP2H/TxCMP2L registers in PWM mode.

In PWM mode, this register determines the period of the PWM signal. In this mode, this register is both readable and writeable. However, on writes the value is not applied until the end of the current PWM cycle.

Writing to the TxCMP2L register causes the value to be stored in the TxCMP2L register with no other effect. Writing to the TxCMP2H register causes an internal compare register to be loaded with a 16-bit value in which the low 8 bits come from the TxCMP2L register and the high 8 bits come from the value being written to the TxCMP2H register. Software should write the TxCMP2L register before writing the TxCMP2H register, because writing to the TxCMP2H register is used as an indication that a new compare value has been written. Writing to the TxCMP2H register is required for the new compare value to take effect. In PWM mode, the 16-bit number latched into the internal compare register by writing to the TxCMP2H register does not take effect until the end of the current PWM cycle.

Reading the TxCMP2H or TxCMP2L registers returns the previously written value regardless of whether the value stored in these registers has been transferred to the

internal compare register by writing to the TxCMP2H register.

In Capture/Compare mode, this register captures the value of the counter/timer when the TxCPI2 input is triggered. In this mode, this register is read-only.

Reading the TxCAP2L register returns the least-significant 8 bits of an internal capture register and causes the most-significant 8-bits to be latched into the TxCAP2H register. This allows software to read the TxCAP2H register later and still be assured of atomicity.

TxCFG1H/TxCFG1L Register

Selects timer operation mode, pin functions, interrupts and other configuration settings.

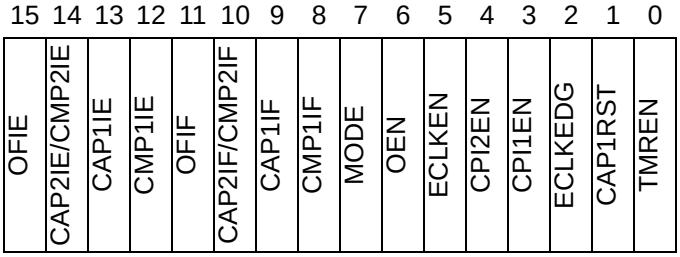


Figure 5-8 TxCFG1H/TxCFG1L Register

- *OFIE*—Timer overflow interrupt enable. Set to enable timer overflow interrupts, clear to disable.
- *CAP2IE/CMP2IE*—in PWM mode, compare 2 interrupt enable. In Compare/Capture mode, capture 2 interrupt enable.
- *CAP1IE*—capture 1 interrupt enable.
- *CMP1IE*—compare 1 interrupt enable.
- *OFIF*—Timer overflow interrupt flag. Set on timer overflow from 0xFFFF to 0x0000.
- *CAP2IF*/*CMP2IF*—in PWM mode, set when the counter/timer increments to the same value held in the TxCMP2 register. In Compare/Capture mode, set when the TxCPI2 input is triggered.
- *CAP1IF*—set when the TxCPI1 input is triggered.
- *CMP1IF*—in PWM mode, set when the counter/timer is reset. In Compare/Capture mode, set when the counter/timer is incremented to the same value held in the TxCMP1 register.
- *MODE*—set to select Compare/Capture mode, cleared to select PWM or Timer mode.
- *OEN*—TxOUT enable. Set to enable, clear to disable. The port direction register bit for the corresponding port pin must be programmed for output to enable this output.
- *ECLKEN*—enables the TxCLK input as a clock for the counter/timer. Set to enable, clear to disable. Setting this bit does not affect the use of the corresponding port pin as a general-purpose input or output.
- *CPI2EN*—enables the TxCAP2 register to be loaded on a capture event. Set to enable, clear to disable. Setting this bit does not affect the use of the corresponding port pin as a general-purpose input or output.
- *CPI1EN*—enables the TxCAP1 register to be loaded on a capture event. Set to enable, clear to disable. Setting this bit does not affect the use of the corresponding port pin as a general-purpose input or output.
- *ECLKEDG*—selects the sensitive edge for clocking the counter/timer. Set for falling edges, clear for rising edges.
- *CAP1RST*—set to enable counter/timer reset on a capture 1 input event, clear to disable reset.
- *TMREN*—set to enable the counter, clear to disable. When the counter/timer is disabled, clocking is inhibited to minimize power consumption.

TxCFG2H/TxCFG2L Register

Selects capture input trigger edges, prescaler setting, and other configuration settings.

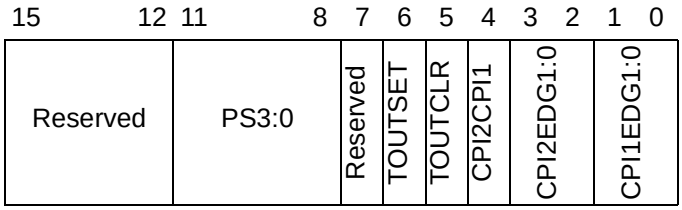


Figure 5-9 TxCFG2H/TxCFG2L Register

- *PS3:0*—Timer prescaler divisor, as shown in Table 5-5.

Table 5-5 Prescaler Divisor

PS3:0	Prescaler Divisor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512

Table 5-5 Prescaler Divisor (continued)

PS3:0	Prescaler Divisor
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

- *TOUTSET*—set this bit to force TxOUT high. Clearing this bit has no effect. This bit always reads as 0.
- *TOUTCLR*—set this bit to force TxOUT low. Clearing this bit has no effect. This bit always reads as 0.
- *CPI2CPI1*—set this bit to tie internally the TxCPI2 input to the TxCPI1 input. When this bit is set, the external TxCPI1 input is used to trigger both capture functions, for measuring both the duty cycle and the period of an external signal. This leaves the pin assigned to the TxCPI2 input free to be used as a general-purpose I/O port pin.
- *CPI2EDG1:0*—these bits select the sensitive edge for the TxCPI2 input. 00 selects falling edges, 01 selects rising edges, 10 and 11 select any edge.
- *CPI1EDG1:0*—these bits select the sensitive edge for the TxCPI1 input. 00 selects falling edges, 01 selects rising edges, 10 and 11 select any edge.

TCTRL Register

Unlike the other timer control registers, there is only one TCTRL register which is used for both timers. Setting the TxRST bit clears the TxCNTH/TxCNTL register pair and the prescaler counter, which allows global synchronization of all timers on the device. There are also individual timer interrupt-enable bits.

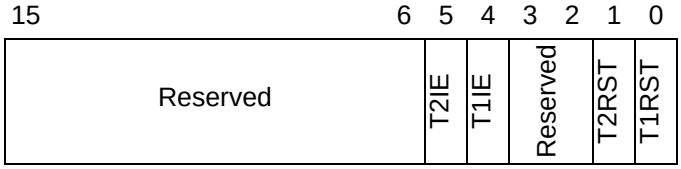


Figure 5-10 TCTRL Register

- *T2IE*—Timer 2 global interrupt enable. Set to enable timer interrupts, clear to disable.
- *T1IE*—Timer 1 global interrupt enable. Set to enable timer interrupts, clear to disable.
- *T2RST*—Timer 2 reset. Set this bit to clear Timer 2 and its prescaler.
- *T1RST*—Timer 1 reset. Set this bit to clear Timer 1 and its prescaler.

5.5 Watchdog Timer

A Watchdog Timer is available for recovering from unexpected system hangups. When the Watchdog Timer is enabled, software must periodically clear the timer by executing a **`cwdt`** instruction. Otherwise, the timer will overflow, which resets the IP2022 and sets the WD bit in the STATUS register. Any other source of reset clears the WD bit, so software can use this bit to identify a reset caused by the Watchdog Timer. The Watchdog Timer is shown in Figure 5-11.

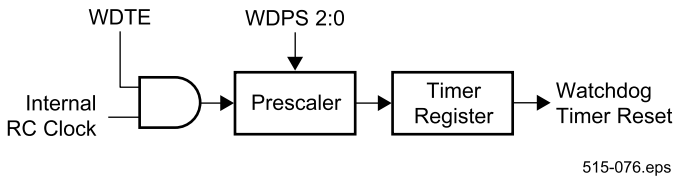


Figure 5-11 Watchdog Timer

The Watchdog Timer is enabled by setting the WDTE bit in the FUSE1 register. The time period between reset or clearing the timer and timer overflow is controlled by the WDPS2:0 bits in the FUSE1 register, as shown in Table 5-6.

Table 5-6 Watchdog Timer Period

WDPS2:0	Period (ms)
000	16.4
001	32.8
010	65.6
011	131
100	262
101	524
110	1049
111	2099

The Watchdog Timer register is not visible to software. The only feature of the Watchdog Timer visible to software is the WD bit in the STATUS register.

5.6 Serializer/Deserializer

There are two serializer/deserializer units in the IP2022, which support a variety of protocols, including SPI interface, I^2C interface, UART, USB bus, and 10Base-T Ethernet. By performing data serialization/deserialization in hardware, the CPU bandwidth needed to support serial communications is greatly reduced, especially at high baud rates. Providing two units allows easy implementation of protocol conversion or bridging functions, such as a USB to 10Base-T Ethernet bridge.

Each serializer/deserializer uses up to 8 external pins: SxRXD, SxRXP, SxRXM, SxTXP, SxTXM, SxTXPE, SxTXME, and SxCLK. The pins for the Serial 1 unit are multiplexed with the Port E pins, and the pins for the Serial 2 unit are multiplexed with the Port F pins. The port direction bits must be set appropriately for each pin that is used. The SxOE signal is multiplexed onto the pin assigned to the SxTXPE signal.

Figure 5-12 shows the clock/data separation and EOP detection logic of a serializer/deserializer unit. The SxRXD input carries the data received from either a transceiver (USB) or a differential line receiver (Ethernet). The SxRXP and SxRXM pins correspond to the differential inputs of the USB bus. Providing both inputs allows sensing of an End-of-Packet (EOP) condition. For protocols with single-ended input, only the SxRXD pin is used and the clock/data separation circuit is bypassed.

The SxRXP and SxRXM pins should not both be grounded or left floating, otherwise a false EOP condition may be detected. At least one of these inputs should be driven high to prevent spurious EOP events.

The synchronization pattern register (SxRSYNC) is used for USB and 10Base-T protocols for detecting bit patterns that signal the start of a frame. For USB, this register is loaded with 00000001, while for 10Base-T, it is 10101011 (called also SFD, start of frame delimiter). The incoming data stream, after passing through the polarity inversion logic (which can be turned on or off under software control) is compared to the synchronization pattern. Once a match is found, an internal counter is set to zero and data is shifted into a shift register. The synchronization matching operation is then disabled until an EOP condition is detected, because the synchronization pattern potentially could be embedded in the data stream as valid data.

The clock/data separation circuit performs Manchester decoding and NRZI decoding. In addition, bit unstuffing is performed after the NRZI decoding for the USB bus. This means every bit after a series of six consecutive ones is dropped.

For UART operation, two internal divide-by-16 circuits are used. Based on the clock source (either internal or external), the receive section and the transmit section use two divided-by-16 clock that potentially can be out of phase. This is due to the nature of the UART bus transfers. The receive logic, based on the 16x bit clock (the clock source chosen by user), will sample the incoming data for an falling edge. Once the edge is detected, the receive logic counts 8 clock cycles and samples the number of bits specified in the SxRCNT register using the bit clock (which is obtained by dividing the clock source by 16).

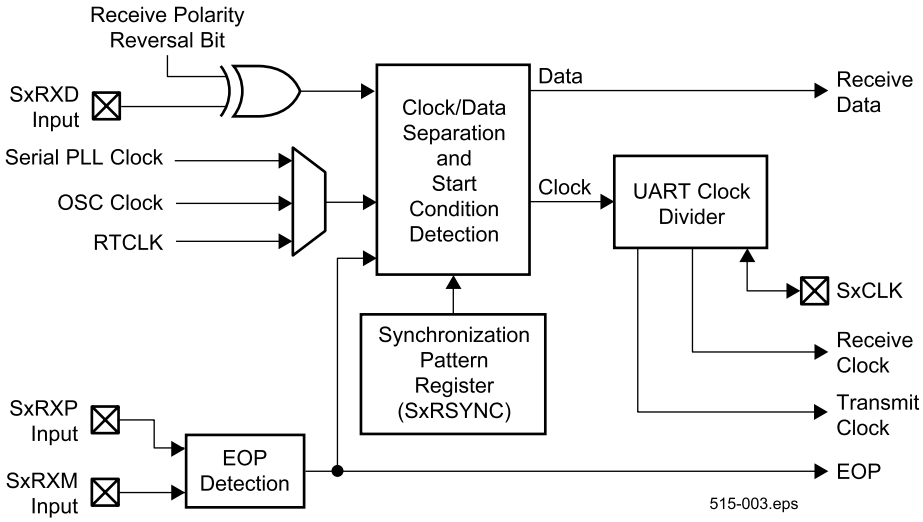


Figure 5-12 Clock/Data Separation and EOP Detection Block Diagram

Figure 5-13 shows the receive data paths. Software prepares the serializer/deserializer to receive data by programming the receive shift count register (SxRCNT) and the clock select bits in the SxMODE register appropriately for the selected protocol. The SxRCNT register is copied to an internal counter, and when that number of bits of data has been received, the received data is loaded into the SxRBUF register.

In the case of the USB bus or Ethernet, when an EOP is detected the SxRCNT register is loaded with the number of bits actually received, the EOP bit of the SxINTF register is set, and the data bits are loaded into the SxRBUF register. The corresponding bit in the SxINTE register can be set to enable an interrupt on this event.

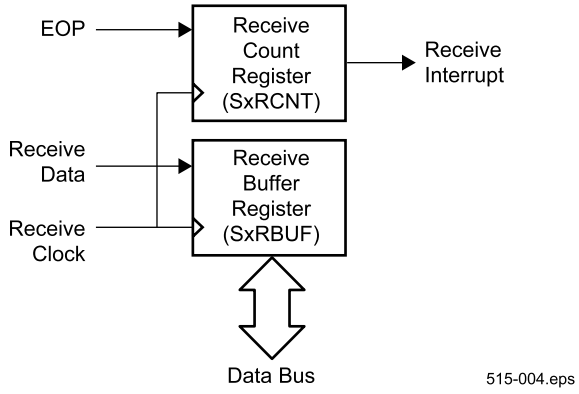


Figure 5-13 Receive Data Paths

Figure 5-14 shows the transmit data paths. The SxTXP and SxTXM pins correspond to the differential outputs of the USB or Ethernet bus. Other serial protocols require only one output pin, which is SxTXP by default.

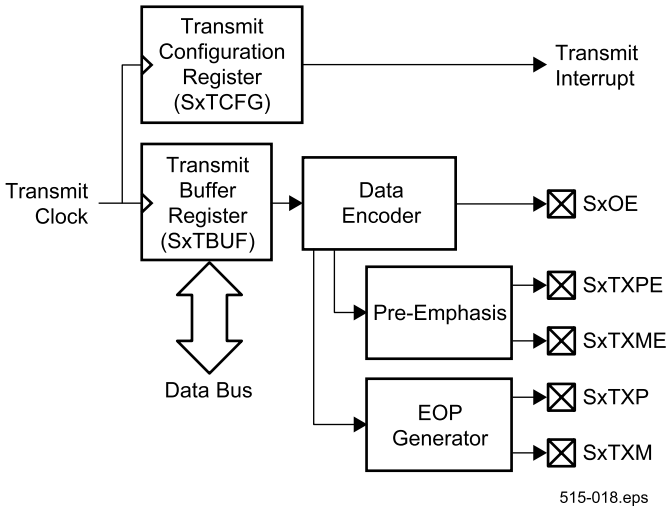


Figure 5-14 Transmit Data Paths

The SxTXP and SxTXM pins have high current outputs for driving Ethernet magnetics directly without the use of transceivers.

When the clock select register is programmed with the value for 10Base-T, the transmit pre-emphasis requirement enables the SxTXPE and SxTXME outputs, which have a 50ns-delayed version of the transmit output that is resistively combined outside the chip before driving the magnetics.

The data encode block performs polarity inversion, if necessary, then in 10Base-T mode it performs Manchester encoding. In USB bus mode, it performs bit stuffing and then NRZI encoding. Bit stuffing means that after six consecutive ones, a zero bit is inserted. The active low SxOE pin is used to enable the USB transceiver for transmission. Otherwise, this pin is held high. For 10Base-T, the output pins of the serializer are driven low when not transmitting. The encode block is bypassed for all other protocols.

For transmitting, software must specify the number of bits to transmit and load the data in the SxTBUF register. When there is a transmit buffer underrun event (i.e. the data has been shifted out, but the SxTBUF register has not been reloaded), an EOP condition is generated on the SxTXP and SxTXM outputs when an internal counter decrements to zero. Also, the EOP flag in the SxINTF register is set. After the EOP event, the TXIDLE flag in the SxINTF register is set.

For protocols other than USB and Ethernet, the EOP generator is bypassed.

Table 5-7 Required Clock Frequencies From PLL

Protocol	Receive	Transmit
USB 1.1 High Speed	48 MHz	12 MHz
USB 1.1 Low Speed	6 MHz	1.5 MHz
10Base-T Ethernet	80 MHz	20 MHz

5.6.1 Serializer/Deserializer Registers

SxMODE Register

Selects between internal, external, or recovered clock and positive or negative clock edges. An optional prescaler allows division of the clock rate, so that one clock source can serve the two serializer/deserializer units which may be using different protocols and speeds.

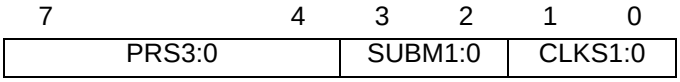


Figure 5-15 SxMODE Register

- *PRS3:0*—selects the protocol mode to enable various functions, as shown in Table 5-9. Unused encodings are reserved for future expansion.
- *SUBM1:0*—selects submodes specific to the enabled protocol, as shown in Table 5-8.

Table 5-8 Submode Field Encoding

SUBM1:0	Protocol	Description
01	USB Bus	Low-Speed USB
10	USB Bus	High-Speed USB
00	Microwire, SPI	Receive on falling edge, transmit on rising edge.
01	Microwire, SPI	Receive on rising edge, transmit on rising edge.
10	Microwire, SPI	Receive on falling edge, transmit on falling edge.
11	Microwire, SPI	Receive on rising edge, transmit on falling edge.

Table 5-9 Protocol Selection

PRS3:0	Mode	Encoding Method	Differential or Single-Ended?	Synchroniza-tion Register Enabled?	EOP Generation/ Detection?	Bit Stuffing/ Unstuffing?	Pre-Emphasis Outputs Enabled?
0000	Disabled	None	None	No	No	No	No
0001	10Base-T	Manchester	Single-Ended	Yes	Yes	No	Yes
0010	USB Bus	NRZI	Differential	Yes	Yes	Yes	No
0011	UART	None	Single-Ended	No	No	No	No
0100	I^2C	None	Single-Ended	No	No	No	No
0101	SPI/ Microwire	None	Single-Ended	No	No	No	No

- *CLKS1:0*—selects the clock source, as shown in Table 5-10.

Table 5-10 Clock Selection

CLKS1:0	Clock Source
00	None, clock disabled
01	Reserved
10	OSC clock oscillator
11	PLL clock multiplier

SxRSYNC Register

Used to store the 8-bit synchronization pattern used for frame start detection with the USB and 10Base-T protocols.

SxRBUFH/SxRBUFL Register

16-bit register pair for unloading received data. The RXBF bit in the SxINTF register indicates when new data has been loaded into this register. If the corresponding bit in the SxINTE register is set, an interrupt is generated.

SxRCFG Register

Selects receiver options.

7	6	5	4 … 0
Reserved	$\overline{\text{SYNCMP}}$	RPOREV	RXSCNT4:0

Figure 5-16 SxRCFG Register

- *$\overline{\text{SYNCMP}}$*—set to disable synchronization pattern detection.
- *RPOREV*—set to reverse received data polarity, otherwise clear.
- *RXSCNT4:0*—specifies the number of bits to receive.

SxRCNT Register

Read-only register which indicates link pulse duration and number of bits actually received.

7	6	5	4 … 0
Rsrv.	RxCRSCTRL		RXACNT4:0

Figure 5-17 SxRCNT Register

- *RxCRSCTRL*—carrier sense status and interrupt control, as shown in Table 5-11.

Table 5-11 RxCRSCtrl Encoding

RxCRSCTRL1:0	Description
0x	No interrupt. Software can poll the RXXCRS bit in the SxINTF register for carrier status.
10	Interrupt on carrier detection.
11	Interrupt on loss of carrier.

- *RXANCNT4:0*—number of bits actually received.

SxTBUFH/SxTBUFL Register

16-bit register pair for loading transmit data. The TXBE bit in the SxINTF register indicates when the data has been transmitted and the register is ready to be loaded with new data. If the corresponding bit in the SxINTE register is set, an interrupt is generated.

SxTCFG Register

Selects transmitter options.

7	6	5	4 … 0
SEREN	Reserved	TPOREV	TXSCNT4:0

Figure 5-18 SxTCFG Register

- *SEREN*—set to enable the serializer, clear to disable.
- *TPOREV*—set to reverse transmit data polarity, otherwise clear.
- *TXSCNT4:0*—specifies the number of bits to transmit.

SxINTF Register

Indicates conditions that may be enabled as interrupts.

7	6	5	4	3	2	1	0
RXERROR	EOP	SYND	TXBE	TXEOP	SXLINKPULSE	RXBF	RXXCRS

Figure 5-19 SxINTF Register

- *RXERROR*—set on double zero detection in Manchester encoding during the preamble. (Not affected by double zero occurring in the data.)
- *EOP*—set on packet detection in USB and 10Base-T modes.
- *SYND*—set when received data matches the pattern held in the SxRSYNC register.
- *TXBE*—set on transmit buffer register (SxTBUF) being unloaded, indicating it is empty and ready to receive new data.
- *TXEOP*—set when the previous data in the transmit buffer register (SxTBUF) has been transmitted and no new data has been loaded into the register. This is a transmit underrun condition, and in the USB or 10Base-T modes it causes an EOP condition to be generated.
- *SXLINKPULSE*—set after a link pulse of 75 to 250 ns duration is detected.
- *RXBF*—set when new data is loaded into the receive buffer register (SxRBUF), indicating it is full and unable to receive new data.
- *RXXCRS*—set when new data is being shifted into the receive buffer register (SxRBUF). May be used as a carrier sense indication, or to check whether data is being received before entering a power-down mode.

SxINTE Register

The SxINTE register has the same format as the SxINTF register. For each condition indicated by a flag in the SxINTF register, setting the corresponding bit in the SxINTE register enables the interrupt for that condition. Figure 5-20 shows the interrupt logic for the two serializer/deserializer units.

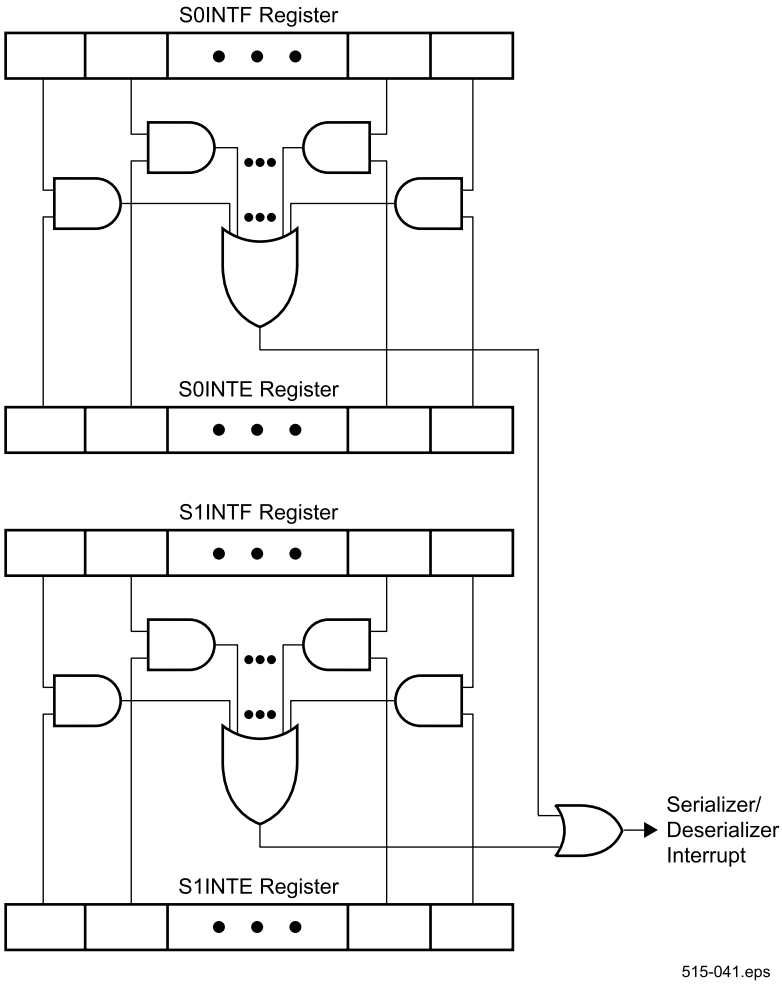


Figure 5-20 Serializer/Deserializer Interrupt Logic

5.6.2 Protocol-Specific Considerations

UART Interface

To set up the serializer/deserializer for UART mode, select UART mode in the PRS3:0 bits of the SxMODE register. This causes the data to be clocked in after a valid start bit is detected. Make sure that the polarity selected by the RPOREV bit in the SxRCFG register matches the polarity provided by the RS-232 transceiver. (Most of them are inverted.) Then, select a clock divisor that is 16 times the desired baud rate.

To operate in UART mode, depending on the application, either transmit or receive can be performed first. In both cases, the configuration register needs to be programmed with a bit count that is appropriate for the format. The bit count depends on the number of data bits, stop bits, and parity bits. The start bit is included in the bit count.

I^2C Protocol

To set up the serializer/deserializer as an I^2C slave, select I^2C in the PRS3:0 bits of the SxMODE register. This causes the data to be clocked in after a valid start condition (i.e. falling edge on the SxRXP input, while the SxCLK signal is high) is detected. This shifting of data continues until a stop condition (i.e. a rising edge on the SxRXP input while the SxCLK signal is high) is detected or another start condition without a stop is detected.

To set up the serializer/deserializer as an I^2C master, select an internal clock with rising edge in the CLKS1:0 and SUBM1:0 bits of the SxMODE register. An

appropriate divisor must also be programmed. Start and stop conditions, arbitration, and synchronization must be performed by software.

To operate in I^2C mode, software is responsible for generating the acknowledge signal when the data is received. At the standard I^2C data rate, there are 2,000 cycles to respond, and in the fast mode, 500 cycles. This involves directly writing to the SxTXP port pin data register and setting it to zero.

The shift count register must be programmed with a value that is appropriate for the format, whether it is 7-bit or 10-bit.

As an I^2C slave, software must hold the SCL line (which is mapped to SxCLK) low, if it cannot respond immediately. For an I^2C master, software must handle the acknowledgement and wait state request.

SPI/Microwire Protocol

To set up the serializer/deserializer for SPI or Microwire protocols, set up clock with opposing phases for transmit and receive by programming the SUBM1:0 field of the SxMODE register. If in slave mode, specify an external clock. If in master mode, specify an internal clock. In slave mode, software must check if the designated chip select (or slave select for SPI) line is activated first before responding. In master mode, software must first set the proper designated chip select (or slave select for SPI) pin to the active level, before enabling the serializer/deserializer.

To operate in SPI or Microwire mode, once the transmit and receive bit counts in the configuration registers are programmed with non-zero values, the serializer/deserializer begins shifting operations on the programmed clock edges. Caution must be exercised to program them quickly to avoid losing any data.

USB 1.1 Protocol

To set up the serializer/deserializer for USB mode, the received data output of the USB transceiver should be connected to SxRXD. The VP and VM pins of the transceiver are connected to the SxRXP and SxRXM pins to allow detection of the EOP condition. Figure 5-21 shows the connections required between an external USB transceiver and the IP2022.

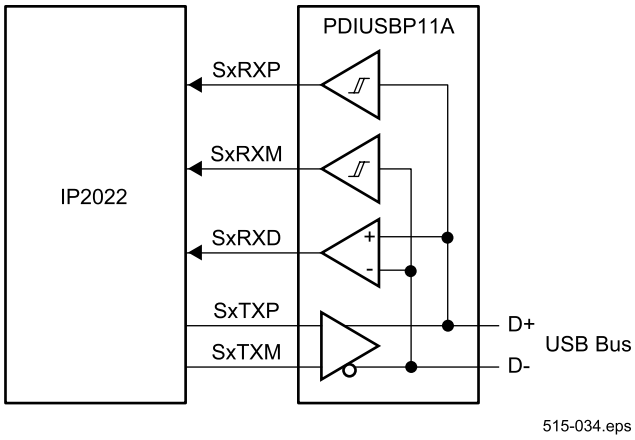


Figure 5-21 USB Interface Example

The SxMODE register must be programmed with values for a recovered clock, and the PLL clock multiplier must be programmed to generate the appropriate frequency. For example, it can be programmed at 48 MHz for full speed with a divisor of zero (=1). A divisor of 8 will make it suitable for low-speed operation. The synchronization pattern must be programmed into the SxRSYNC register to trigger an interrupt when a packet is received.

To operate in USB mode, software must perform the following functions:

- CRC generation and checking.
- Detecting reset of the device function, which is indicated by 10 milliseconds of a single-ended zero (SE0) condition on the bus.
- Detecting the suspend state, which is indicated by more than 3 milliseconds of idle. Software must make sure that the suspend current of 500 µA will be drawn after 10 milliseconds of bus inactivity.
- Formation of the USB packet by putting the sync, pid, and data into the transmit register and setting the proper count.

10Base-T Ethernet Protocol

To set up the serializer/deserializer for 10Base-T Ethernet, the input data from a differential line receiver or on-board comparator is connected to the SxRXD input. The signals designated Tx+, Tx-, TxD+, and TxD- correspond to the SxTXP, SxTXM, SxTXPE, and SxTXME pins of the corresponding serializers/deserializers. These pins are connected to an RJ45 jack through a transformer with terminations.

The SxMODE register must be programmed for a recovered clock, and the PLL clock multiplier must be

programmed for an appropriate speed. For example, it can be programmed to be 80 MHz for 8× oversampling. The received data stream is used, together with the clock recovery circuit, to recover the original transmit clock and data.

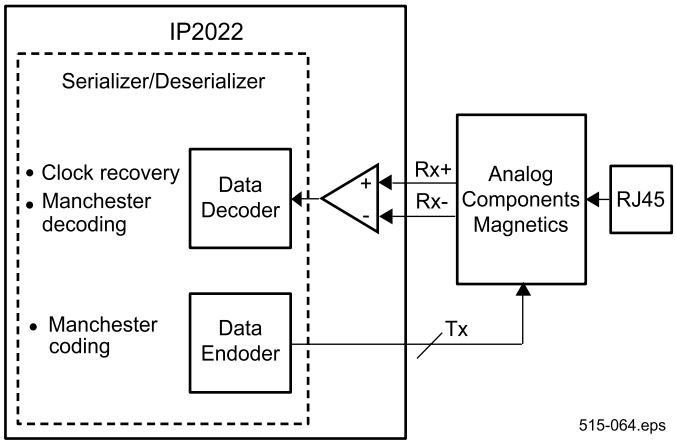


Figure 5-22 Ethernet Interface Example

Software must perform the following functions:

- Polarity detection and reversal.
- Carrier sense.
- Jabber detection.
- Link integrity test and link pulse generation.
- Random back off in case of collision.
- When a collision is detected, sending a 32-bit jam sequence. Collisions can be detected by either receiving an RXXCRS interrupt or by setting the bit count to 7 and then received a RXBF interrupt while transmitting.
- Formation of Ethernet packet by putting the preamble, sfd, destination address, source address, length/type, MAC client data into the transmit buffer. Frame check computation also must be done in software.
- MAC layer functions.

5.7 Analog to Digital Converter (ADC)

The on-chip A/D converter has the following features:

- 10-bit ADC, ½ LSB accuracy
- 8 input channels
- 48 kHz maximum sampling rate
- One-shot conversion.
- Optional external reference voltage
- Vmax = AVdd (max 2.7V)
- Result returned in the ADCH and ADCL registers

Figure 5-23 shows the A/D converter circuitry. The ADC input pins use alternate functions of the Port G pins. The result of an ADC sample is the analog value measured on the selected pin. To correctly read an external voltage, the pin being sampled must be configured as an input in the port direction register (i.e. the RGDIR register). If the pin is configured as an output, then the result will indicate the voltage level being driven by the output buffer. The RG1 and RG2 port pins are also used as the analog comparator input pins. The result of sampling the RG1 or RG2 pins will be correct whether or not the comparator is operating. The RG0 pin is also used as the comparator output pin. If the comparator is enabled, then sampling the RG0 pin will indicate the voltage level being driven by the comparator. The RG3 pin is multiplexed with the external reference voltage.

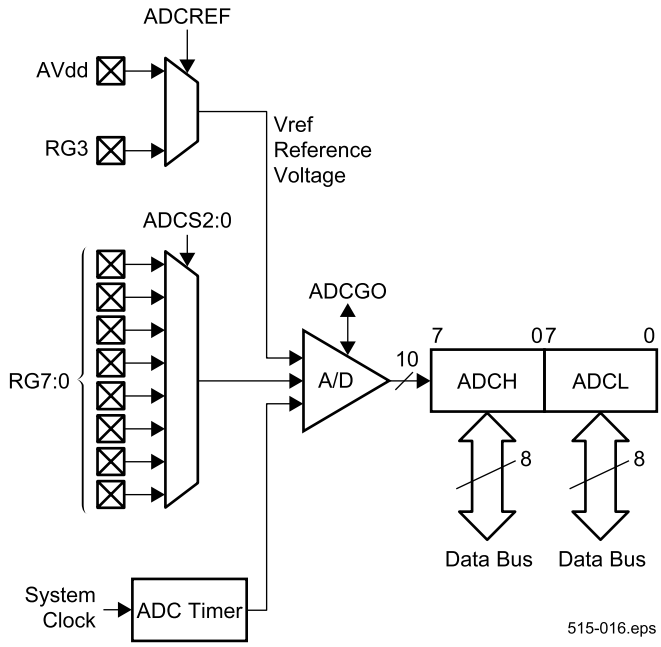


Figure 5-23 A/D Converter Block Diagram

5.7.1 Reference Voltage

The reference voltage (Vref) can come from either the RG3 port pin or from the AVdd supply voltage. If AVdd is used, the RG3 port pin may be used as a channel of analog input or as a general-purpose port pin.

Vref defines a voltage level which reads as one increment of resolution below the full-scale voltage. The full-scale voltage reads as 0x3FF, so the Vref voltage reads as 0x3FE and the A/D converter resolution is Vref/0x3FE.

Table 5-12 shows the values reported at the upper and lower limits of the ADC input voltage range.

Table 5-12 ADC Values

Vin Voltage	ADC Value
0V	0x000
Vref/0x3FE	0x001
Vref	0x3FE
Vref + (Vref/0x3FE)	0x3FF

5.7.2 ADC Result Justification

The 10 bits of the ADC value can be mapped to the 16 bits of the ADCH/ADCL register pair in three different ways, as shown in Table 5-13. In this table, the numbers in the cells represent bit positions in the 10-bit ADC value, Z represents zero (as opposed to bit position 0), and -9 represents the inversion of bit position 9.

Table 5-13 Justification of the ADC Value

Mode	ADCH Register Bits								ADCL Register Bits							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Left Justified	9	8	7	6	5	4	3	2	1	0	Z	Z	Z	Z	Z	Z
Right Justified	Z	Z	Z	Z	Z	Z	9	8	7	6	5	4	3	2	1	0
Signed	-9	-9	-9	-9	-9	-9	-9	8	7	6	5	4	3	2	1	0

5.7.3 Using the A/D Converter

The following sequence is recommended:

1. Set the ADCTMR register to the correct value for the system clock speed.
2. Load the ADCCFG register to specify the channel and set the ADCGO bit. Setting the ADCGO bit enables and resets the ADC timer.
3. After a period of time (12 timer overflows = 20.8 µs) the conversion will complete, the ADCGO bit will be cleared, and the ADC timer will be disabled.
4. A timer-based interrupt service routine detects (or assumes) the ADCGO bit has been cleared and reads the ADC value.
5. Another load to the ADCCFG register can then be used start another conversion.

5.7.4 A/D Converter Registers

ADCTMR Register

The ADCTMR register is used to specify the number of system clock cycles required for a delay of 1736 ns, which is used to provide the 576 kHz (48 kHz × 12) clock period reference clock for the A/D converter.

At a clock frequency of 100 MHz, the timer register should be set to 174 (100 MHz/0.576 MHz). The minimum value that may be loaded into the ADCTMR register is 2, so the system clock must be at least 24 times the ADC sampling frequency for the ADC to function.

ADCCFG Register

The A/D converter configuration register (ADCCFG) provides the control and status bits for the A/D converter, as shown in Figure 5-24.

7	6	5	4	3	2	0
ADCREF	ADCJST		Reserved	ADCGO	ADCS2:0	

Figure 5-24 ADCCFG Register

- *ADCREF*—set to use an external reference voltage, clear to use AVdd as the reference voltage.
- *ADCJST*—00 selects right justified, 01 selects signed, and 10 selects left justified. 11 is reserved.
- *ADCGO*—set to start a conversion. When the bit becomes clear, the conversion is complete.
- *ADCS2:0*—specifies the bit number of the Port G pin used as the analog input.

5.8 Analog Comparator

The IP2022 has an on-chip analog comparator which uses alternate functions of the RG0, RG1, and RG2 port pins. The RG1 and RG2 pins are the comparator negative and positive inputs, respectively, while the RG0 pin is the comparator output pin. To use the comparator, software must program the port direction register (RGDIR) so that

RG1 and RG2 are inputs. RG0 may be set up as a comparator output pin.

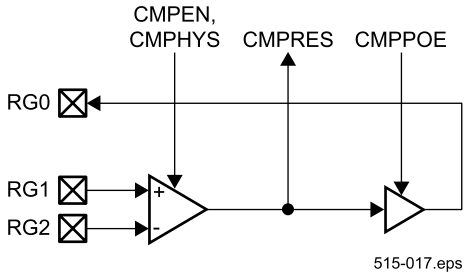


Figure 5-25 Analog Comparator

The comparator enable bits are cleared on reset, which disables the comparator. To avoid drawing additional current during power-down mode, the comparator should be disabled before entering power-down mode. A 50 mV hysteresis is applied between the inputs, when the CMPHYS bit is set in the CMPCFG register.

5.8.1 Analog Comparator Register

The CMPCFG register is used to enable the comparator, to read the output of the comparator internally, to enable the output of the comparator to the comparator output pin, and to enable the hysteresis. Figure 5-26 shows the bits in this register.

7	6	5	4	1	0
CMPEN	CMPOE	CMPHYS	Reserved		CMPRES

Figure 5-26 CMPCFG Register

- *CMPEN*—set to enable the comparator, clear to disable.
- *CMPOE*—set to enable the comparator output on the RG0 pin, clear to disable. (If the comparator is enabled and this bit is set, the RG0 pin will be configured as an output, overriding bit 0 in the RGDIR register.)
- *CMPHYS*—set to enable hysteresis, clear to disable.
- *CMPRES*—comparator result (read-only).

5.9 Parallel Slave Peripheral

The parallel slave peripheral allows the IP2022 to operate as an 8- or 16-bit slave to an external device, much like a memory chip. Alternate functions of Port C and Port D are used for transferring data, and alternate functions of Port B are used for control signals. Figure 5-27 shows the connections between an external master and the parallel slave peripheral interface.

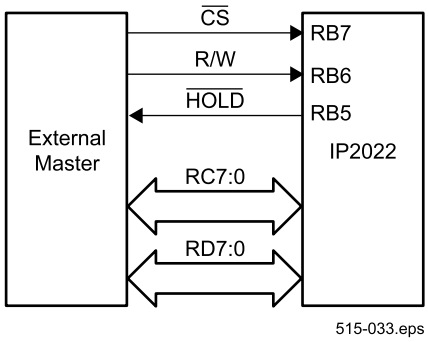


Figure 5-27 Parallel Slave Peripheral

To read or write through the Parallel Slave Peripheral interface, the external master asserts the chip select ($\overline{CS}$) signal low. This signal is an alternate function of port pin RB7. The direction of transfer is indicated by the R/W signal, which is an alternate function of port pin RB6. When the R/W signal is high, the master is reading from the slave. When the R/W signal is low, the master is writing to the slave.

Optionally, a $\overline{HOLD}$ signal may be enabled as an alternate function of port pin RB5. Assertion of $\overline{HOLD}$ indicates to the external master that the Parallel Slave Peripheral interface is not ready to allow the data transfer to complete. The $\overline{HOLD}$ signal is driven like an open-collector signal, i.e. low when asserted and high-impedance when not asserted. When the $\overline{CS}$ signal is not asserted (i.e. the IP2022 is not selected), the $\overline{HOLD}$ signal is in high-impedance mode.

When $\overline{CS}$ is asserted, an interrupt is generated and $\overline{HOLD}$ (if enabled) is automatically asserted. If the data transfer is a write from the external master, software reads the Port C, Port D, or both. If the data transfer is a read, software writes the data to the port or ports. Finally, if $\overline{HOLD}$ is asserted, software releases assertion of $\overline{HOLD}$ by writing to the PSPRDY bit in the PSPCFG register.

The Parallel Slave Peripheral does not generate interrupts by itself. Software is required to enable port pin RB7 as a falling-edge interrupt input for the Parallel Slave Peripheral to function.

The PSPCFG register is used to enable the Parallel Slave Peripheral, select which ports are used for data transfer, enable the $\overline{HOLD}$ output, and release the $\overline{HOLD}$ output when the data transfer is ready to complete.

7	6	5	4	3	0
PSPEN2	PSPEN1	PSPHEN	PSPRDY	Reserved	

Table 5-14 PSPCFG Register

- *PSPEN2*—set to enable Port D for data transfer, clear to disable. (If this bit is set, the Parallel Slave Peripheral overrides the RDDIR register.)
- *PSPEN1*—set to enable Port C for data transfer, clear to disable. (If this bit is set, the Parallel Slave Peripheral overrides the RCDIR register.)
- *PSPHEN*—set to enable $\overline{\text{HOLD}}$ output, clear to disable. (If this bit is set, the Parallel Slave Peripheral overrides bit 5 of the RBDIR register.)
- *PSPRDY*—set to release $\overline{\text{HOLD}}$. This bit always reads as 0.

6.0 In-System Programming

The interface used for in-system programming (ISP) and in-system debugging (ISD) is compatible with the SPI serial interface protocol. Whenever possible, a standard connector should be incorporated in the system design for in-system debugging and programming. The recommended connector layout for the ISD/ISP interface is shown in Figure 6-1. The recommended connector is a male 10-pin connector with 100-mil pin spacing, whose pin assignments are listed in Table 6-1. The connector is keyed to prevent backward insertion.

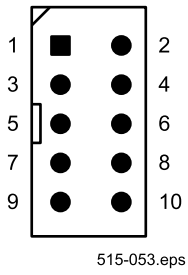


Figure 6-1 Recommended ISD/ISP Connector

Signal levels on the connector are LVTTL-compatible. The target system provides the TSCK, TSI, $\overline{\text{TRST}}$, and $\overline{\text{TSS}}$ signals with 10K ohm pullup resistors.

For more information about the ISD/ISP interface and the interaction between the debugger/programmer and the target system, see the *IP2022 User's Manual*.

Table 6-1 Connector Pin Assignments

Pin	Name	Description
1	KEY	Key (not a signal)
2	$\overline{\text{TSS}}$	*Target Slave Select*—Active-low signal which enables the IP2022 to communicate on the SPI bus. Connect to pin 1 on the IP2022.
3	GND	Ground
4	TSCK	*Target Data Clock*—Serial clock. Connect to pin 2 on the IP2022.
5	OSC	*Target Clock Oscillator*—If the debugger/programmer is capable of supplying an OSC clock for the target system, then this clock must be configurable so that it can be disabled to prevent it from interfering with the target system (i.e. the OSC clock output is placed in a high-impedance state).
6	Reserved	Reserved
7	$\overline{\text{TRST}}$	*Target Reset*—The target system may use the $\overline{\text{TRST}}$ signal to reset the entire system, to reset only the IP2022, or it may ignore the $\overline{\text{TRST}}$ signal. The debugger/programmer may provide a 100-ms system reset signal ($\overline{\text{TRST}}$) to the target system. If supported, the $\overline{\text{TRST}}$ output must be an open-collector driver to accommodate other sources of reset in the target system. The minimum source requirement for this driver is 6 mA. The debugger/programmer should not detect or be reset by the $\overline{\text{TRST}}$ signal being driven low by the target system. There is no requirement that the IP2022 is connected to the $\overline{\text{TRST}}$ signal, so the debugger/programmer cannot assume that the IP2022 has been reset if the target system pulls the $\overline{\text{TRST}}$ pin low.
8	TSI	*Target Serial Input*—Sampled on the rising edge of TSCK. Connect to pin 3 on the IP2022.
9	Vdd	Power
10	TSO	*Target Serial Output*—Driven after the falling edge of TSCK. Connect to pin 4 on the IP2022.

7.0 Register Quick Reference

7.1 Registers (sorted by address)

Table 7-1 shows the addresses and reset values of all special-purpose registers in data memory, sorted by their address.

Table 7-1 Register Addresses and Reset State

Address	Name	Description	Reset Value
0x0001	Reserved	Reserved	Reserved
0x0002	Reserved	Reserved	Reserved
0x0003	Reserved	Reserved	Reserved
0x0004	IPH	Indirect Pointer (high byte)	00000000
0x0005	IPL	Indirect Pointer (low byte)	00000000
0x0006	SPH	Stack Pointer (high byte)	00000000
0x0007	SPL	Stack Pointer (low byte)	00000000
0x0008	PCH	Current PC bits 15:8 (read-only)	11111111
0x0009	PCL	Virtual register for direct PC modification	11110000
0x000A	W	W register	00000000
0x000B	STATUS	STATUS register	On POR or RST Reset: 11100000 On Brown-out Reset: 11101000 On WDT Overflow: 11110000
0x000C	DPH	Data Pointer (high byte)	00000000
0x000D	DPL	Data Pointer (low byte)	00000000
0x000E	SPDREG	Current speed (read-only)	10010011
0x000F	MULH	Multiply result (high byte)	00000000
0x0010	ADDRH	Program memory address (high byte)	00000000
0x0011	ADDRL	Program memory address (low byte)	00000000
0x0012	DATAH	Program memory data (high byte)	00000000
0x0013	DATAL	Program memory data (low byte)	00000000
0x0014	INTVECH	Interrupt vector (high byte)	00000000
0x0015	INTVECL	Interrupt vector (low byte)	00000000
0x0016	INTSPD	Interrupt speed register	00000000
0x0017	RBINTF	Port B interrupt flags	00000000
0x0018	RBINTE	Port B interrupt enable bits	00000000
0x0019	RBINTED	Port B interrupt edge select bits	00000000
0x001A	FCFG	Flash configuration register	00000000
0x001B	TCTRL	Timer 1/2 common control register	00000000
0x001C	XCFG	Extended configuration	00000001
0x001D	Reserved	Reserved	Reserved
0x001E	IPCH	Interrupt return address (high byte)	00000000
0x001F	IPCL	Interrupt return address (low byte)	00000000
0x0020	RAIN	Data on Port A pins	N/A
0x0021	RAOUT	Port A output latch	00000000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x0022	RADIR	Port A direction register	11111111
0x0023	Reserved	Reserved	Reserved
0x0024	RBIN	Data on Port B pins	N/A
0x0025	RBOUT	Port B output latch	00000000
0x0026	RBDIR	Port B direction register	11111111
0x0027	Reserved	Reserved	Reserved
0x0028	RCIN	Data on Port C pins	N/A
0x0029	RCOUT	Port C output latch	00000000
0x002A	RCDIR	Port C direction register	11111111
0x002B	Reserved	Reserved	Reserved
0x002C	RDIN	Data on Port D pins	N/A
0x002D	RDOUT	Port D output latch	00000000
0x002E	RDDIR	Port D direction register	11111111
0x002F	Reserved	Reserved	Reserved
0x0030	REIN	Data on Port E pins	N/A
0x0031	REOUT	Port E output latch	00000000
0x0032	REDIR	Port E direction register	11111111
0x0033	Reserved	Reserved	Reserved
0x0034	RFIN	Data on Port F pins	N/A
0x0035	RFOUT	Port F output latch	00000000
0x0036	RFDIR	Port F direction register	11111111
0x0037	Reserved	Reserved	Reserved
0x0038	Reserved	Reserved	Reserved
0x0039	RGOUT	Port G output latch	00000000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x003A	RGDIR	Port G direction register	11111111
0x003B	Reserved	Reserved	Reserved
0x003C	Reserved	Reserved	Reserved
0x003D	Reserved	Reserved	Reserved
0x003E	Reserved	Reserved	Reserved
0x003F	Reserved	Reserved	Reserved
0x0040	RTTMR	Real-time timer value	00000000
0x0041	RTCFG	Real-time timer configuration register	00000000
0x0042	T0TMR	Timer 0 value	00000000
0x0043	T0CFG	Timer 0 configuration register	00000000
0x0044	T1CNTH	Timer 1 counter register (high byte, read-only)	00000000
0x0045	T1CNTL	Timer 1 counter register (low byte, read-only)	00000000
0x0046	T1CAP1H	Timer 1 Capture 1 register (high byte, read-only)	00000000
0x0047	T1CAP1L	Timer 1 Capture 1 register (low byte, read-only)	00000000
0x0048	T1CAP2H T1CMP2H	Timer 1 Capture 2 (high byte) Timer 1 Compare 2 (high byte)	00000000
0x0049	T1CAP2L T1CMP2L	Timer 1 Capture 2 (low byte) Timer 1 Compare 2 (low byte)	00000000
0x004A	T1CMP1H	Timer 1 Compare 1 register (high byte)	00000000
0x004B	T1CMP1L	Timer 1 Compare 1 register (low byte)	00000000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x004C	T1CFG1H	Timer 1 configuration register 1 (high byte)	00000000
0x004D	T1CFG1L	Timer 1 configuration register 1 (low byte)	00000000
0x004E	T1CFG2H	Timer 1 configuration register 2 (high byte)	00000000
0x004F	T1CFG2L	Timer 1 configuration register 2 (low byte)	00000000
0x0050	ADCH	ADC value (high byte)	00000000
0x0051	ADCL	ADC value (low byte)	00000000
0x0052	ADCCFG	ADC configuration register	00000000
0x0053	ADCTMR	ADC timer register	00000000
0x0054	T2CNTH	Timer 2 counter register (high byte, read-only)	00000000
0x0055	T2CNTL	Timer 2 counter register (low byte, read-only)	00000000
0x0056	T2CAP1H	Timer 2 Capture 1 register (high byte, read-only)	00000000
0x0057	T2CAP1L	Timer 2 Capture 1 register (low byte, read-only)	00000000
0x0058	T2CAP2H T2CMP2H	Timer 2 Capture 2 (high byte) Timer 2 Compare 2 (high byte)	00000000
0x0059	T2CAP2L T2CMP2L	Timer 2 Capture 2 (low byte) Timer 2 Compare 2 (low byte)	00000000
0x005A	T2CMP1H	Timer 2 Compare 1 register (high byte)	00000000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x005B	T2CMP1L	Timer 2 Compare 1 register (low byte)	00000000
0x005C	T2CFG1H	Timer 2 configuration register 1 (high byte)	00000000
0x005D	T2CFG1L	Timer 2 configuration register 1 (low byte)	00000000
0x005E	T2CFG2H	Timer 2 configuration register 2 (high byte)	00000000
0x005F	T2CFG2L	Timer 2 configuration register 2 (low byte)	00000000
0x0060	S1TMRH	Serializer 1 clock timer register (high byte)	00000000
0x0061	S1TMRL	Serializer 1 clock timer register (low byte)	00000000
0x0062	S1TBUFH	Serializer 1 transmit buffer (high byte)	00000000
0x0063	S1TBUFL	Serializer 1 transmit buffer (low byte)	00000000
0x0064	S1TCFG	Serializer 1 transmit configuration	00000000
0x0065	S1RCNT	Serializer 1 received bit count (actual) (read-only)	00000000
0x0066	S1RBUFH	Serializer 1 receive buffer (high byte)	00000000
0x0067	S1RBUFL	Serializer 1 receive buffer (low byte)	00000000
0x0068	S1RCFG	Serializer 1 receive configuration	00000000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x0069	S1RSYNC	Serializer 1 receive bit sync pattern	00000000
0x006A	S1INTF	Serializer 1 status/interrupt flags	00000000
0x006B	S1INTE	Serializer 1 interrupt enable bits	00000000
0x006C	S1MODE	Serializer 1 serial mode/clock select register	00000000
0x006D	S1SMASK	Serializer 1 receive sync mask	00000000
0x006E	PSPCFG	Parallel slave peripheral configuration register	00000000
0x006F	CMPCFG	Comparator configuration register	00000000
0x0070	S2TMRH	Serializer 2 clock timer register (high byte)	00000000
0x0071	S2TMRL	Serializer 2 clock timer register (low byte)	00000000
0x0072	S2TBUFH	Serializer 2 transmit buffer (high byte)	00000000
0x0073	S2TBUFL	Serializer 2 transmit buffer (low byte)	00000000
0x0074	S2TCFG	Serializer 2 transmit configuration	00000000
0x0075	S2RCNT	Serializer 2 received bit count (actual) (read-only)	00000000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x0076	S2RBUFH	Serializer 2 receive buffer (high byte)	00000000
0x0077	S2RBUFL	Serializer 2 receive buffer (low byte)	00000000
0x0078	S2RCFG	Serializer 2 receive configuration	00000000
0x0079	S2RSYNC	Serializer 2 receive bit sync pattern	00000000
0x007A	S2INTF	Serializer 2 status/interrupt flags	00000000
0x007B	S2INTE	Serializer 2 interrupt enable bits	00000000
0x007C	S2MODE	Serializer 2 serial mode/clock select register	00000000
0x007D	S2SMASK	Serializer 2 receive sync mask	00000000
0x007E	CALLH	Top of call stack (high byte)	11111111
0x007F	CALLL	Top of call stack (low byte)	11111111

7.2 Registers (sorted alphabetically)

Table 7-2 shows the addresses and reset values of all special-purpose registers in data memory, sorted alphabetically by name.

Table 7-2 Register Addresses and Reset State

Address	Name	Description	Reset Value
0x0052	ADCCFG	ADC configuration register	00000000
0x0050	ADCH	ADC value (high byte)	00000000
0x0051	ADCL	ADC value (low byte)	00000000
0x0053	ADCTMR	ADC timer register	00000000
0x0010	ADDRH	Program memory address (high byte)	00000000
0x0011	ADDRL	Program memory address (low byte)	00000000
0x007E	CALLH	Top of call stack (high byte)	11111111
0x007F	CALLL	Top of call stack (low byte)	11111111
0x006F	CMPCFG	Comparator configuration register	00000000
0x0012	DATAH	Program memory data (high byte)	00000000
0x0013	DATAL	Program memory data (low byte)	00000000
0x000C	DPH	Data Pointer (high byte)	00000000
0x000D	DPL	Data Pointer (low byte)	00000000
0x001A	FCFG	Flash configuration register	00000000
0x0016	INTSPD	Interrupt speed register	00000000
0x0014	INTVECH	Interrupt vector (high byte)	00000000
0x0015	INTVECL	Interrupt vector (low byte)	00000000
0x001E	IPCH	Interrupt return address (high byte)	00000000
0x001F	IPCL	Interrupt return address (low byte)	00000000
0x0004	IPH	Indirect Pointer (high byte)	00000000
0x0005	IPL	Indirect Pointer (low byte)	00000000
0x000F	MULH	Multiply result (high byte)	00000000
0x0008	PCH	Current PC bits 15:8 (read-only)	11111111
0x0009	PCL	Virtual register for direct PC modification	11110000
0x006E	PSPCFG	Parallel slave peripheral configuration register	00000000
0x0022	RADIR	Port A direction register	11111111
0x0020	RAIN	Data on Port A pins	N/A
0x0021	RAOUT	Port A output latch	00000000
0x0026	RBDIR	Port B direction register	11111111
0x0024	RBIN	Data on Port B pins	N/A
0x0018	RBINTE	Port B interrupt enable bits	00000000
0x0019	RBINTED	Port B interrupt edge select bits	00000000
0x0017	RBINTF	Port B interrupt flags	00000000
0x0025	RBOUT	Port B output latch	00000000

Table 7-2 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x002A	RCDIR	Port C direction register	11111111
0x0028	RCIN	Data on Port C pins	N/A
0x0029	RCOUT	Port C output latch	00000000
0x002E	RDDIR	Port D direction register	11111111
0x002C	RDIN	Data on Port D pins	N/A
0x002D	RDOUT	Port D output latch	00000000
0x0032	REDIR	Port E direction register	11111111
0x0030	REIN	Data on Port E pins	N/A
0x0031	REOUT	Port E output latch	00000000
0x0036	RFDIR	Port F direction register	11111111
0x0034	RFIN	Data on Port F pins	N/A
0x0035	RFOUT	Port F output latch	00000000
0x003A	RGDIR	Port G direction register	11111111
0x0039	RGOUT	Port G output latch	00000000
0x0041	RTCFG	Real-time timer configuration register	00000000
0x0040	RTTMR	Real-time timer value	00000000
0x006B	S1INTE	Serializer 1 interrupt enable bits	00000000
0x006A	S1INTF	Serializer 1 status/interrupt flags	00000000

Table 7-2 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x006C	S1MODE	Serializer 1 serial mode/clock select register	00000000
0x0066	S1RBUFH	Serializer 1 receive buffer (high byte)	00000000
0x0067	S1RBUFL	Serializer 1 receive buffer (low byte)	00000000
0x0068	S1RCFG	Serializer 1 receive configuration	00000000
0x0065	S1RCNT	Serializer 1 received bit count (actual) (read-only)	00000000
0x0069	S1RSYNC	Serializer 1 receive bit sync pattern	00000000
0x006D	S1SMASK	Serializer 1 receive sync mask	00000000
0x0062	S1TBUFH	Serializer 1 transmit buffer (high byte)	00000000
0x0063	S1TBUFL	Serializer 1 transmit buffer (low byte)	00000000
0x0064	S1TCFG	Serializer 1 transmit configuration	00000000
0x0060	S1TMRH	Serializer 1 clock timer register (high byte)	00000000
0x0061	S1TMRL	Serializer 1 clock timer register (low byte)	00000000
0x007B	S2INTE	Serializer 2 interrupt enable bits	00000000
0x007A	S2INTF	Serializer 2 status/interrupt flags	00000000

Table 7-2 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x007C	S2MODE	Serializer 2 serial mode/clock select register	00000000
0x0076	S2RBUFH	Serializer 2 receive buffer (high byte)	00000000
0x0077	S2RBUFL	Serializer 2 receive buffer (low byte)	00000000
0x0078	S2RCFG	Serializer 2 receive configuration	00000000
0x0075	S2RCNT	Serializer 2 received bit count (actual) (read-only)	00000000
0x0079	S2RSYNC	Serializer 2 receive bit sync pattern	00000000
0x007D	S2SMASK	Serializer 2 receive sync mask	00000000
0x0072	S2TBUFH	Serializer 2 transmit buffer (high byte)	00000000
0x0073	S2TBUFL	Serializer 2 transmit buffer (low byte)	00000000
0x0074	S2TCFG	Serializer 2 transmit configuration	00000000
0x0070	S2TMRH	Serializer 2 clock timer register (high byte)	00000000
0x0071	S2TMRL	Serializer 2 clock timer register (low byte)	00000000
0x000E	SPDREG	Current speed (read-only)	10010011
0x0006	SPH	Stack Pointer (high byte)	00000000
0x0007	SPL	Stack Pointer (low byte)	00000000

Table 7-2 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x000B	STATUS	STATUS register	On POR or RST Reset: 11100000 On Brown-out Reset: 11101000 On WDT Overflow: 11110000
0x0043	T0CFG	Timer 0 configuration register	00000000
0x0042	T0TMR	Timer 0 value	00000000
0x0046	T1CAP1H	Timer 1 Capture 1 register (high byte, read-only)	00000000
0x0047	T1CAP1L	Timer 1 Capture 1 register (low byte, read-only)	00000000
0x0048	T1CAP2H T1CMP2H	Timer 1 Capture 2 (high byte) Timer 1 Compare 2 (high byte)	00000000
0x0049	T1CAP2L T1CMP2L	Timer 1 Capture 2 (low byte) Timer 1 Compare 2 (low byte)	00000000
0x004C	T1CFG1H	Timer 1 configuration register 1 (high byte)	00000000
0x004D	T1CFG1L	Timer 1 configuration register 1 (low byte)	00000000
0x004E	T1CFG2H	Timer 1 configuration register 2 (high byte)	00000000
0x004F	T1CFG2L	Timer 1 configuration register 2 (low byte)	00000000
0x004A	T1CMP1H	Timer 1 Compare 1 register (high byte)	00000000

Table 7-2 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x004B	T1CMP1L	Timer 1 Compare 1 register (low byte)	00000000
0x0044	T1CNTH	Timer 1 counter register (high byte, read-only)	00000000
0x0045	T1CNTL	Timer 1 counter register (low byte, read-only)	00000000
0x0056	T2CAP1H	Timer 2 Capture 1 register (high byte, read-only)	00000000
0x0057	T2CAP1L	Timer 2 Capture 1 register (low byte, read-only)	00000000
0x0058	T2CAP2H T2CMP2H	Timer 2 Capture 2 (high byte) Timer 2 Compare 2 (high byte)	00000000
0x0059	T2CAP2L T2CMP2L	Timer 2 Capture 2 (low byte) Timer 2 Compare 2 (low byte)	00000000
0x005C	T2CFG1H	Timer 2 configuration register 1 (high byte)	00000000
0x005D	T2CFG1L	Timer 2 configuration register 1 (low byte)	00000000
0x005E	T2CFG2H	Timer 2 configuration register 2 (high byte)	00000000

Table 7-2 Register Addresses and Reset State (continued)

Address	Name	Description	Reset Value
0x005F	T2CFG2L	Timer 2 configuration register 2 (low byte)	00000000
0x005A	T2CMP1H	Timer 2 Compare 1 register (high byte)	00000000
0x005B	T2CMP1L	Timer 2 Compare 1 register (low byte)	00000000
0x0054	T2CNTH	Timer 2 counter register (high byte, read-only)	00000000
0x0055	T2CNTL	Timer 2 counter register (low byte, read-only)	00000000
0x001B	TCTRL	Timer 1/2 common control register	00000000
0x000A	W	W register	00000000
0x001C	XCFG	Extended configuration	00000001

7.3 Register Bit Definitions

For those registers which have special functions assigned to bits or fields within the register, the definition of those bits and fields is described below. The registers are presented alphabetically.

7.3.1 ADCCFG Register (A/D Converter Configuration)

7	6	5	4	3	2	0
ADCREF	ADCJST		Rsrvd.	ADCGO	ADCS2:0	

Name	Description
ADCREF	A/D converter reference voltage select 0 = AVDD is the reference voltage 1 = RG3 port pin is used to receive an external reference voltage
ADCJST	A/D converter result justification mode select 00 = Right justified 01 = Signed 10 = Left justified 11 = Reserved
ADCGO	A/D converter GO/DONE bit 0 = When the last conversion has completed, this bit reads as 0. 1 = Write 1 to begin a new conversion. While the conversion is in progress, this bit reads as 1.
ADCS2:0	A/D converter input channel select 000 = Port pin RG0 001 = Port pin RG1 010 = Port pin RG2 011 = Port pin RG3 100 = Port pin RG4 101 = Port pin RG5 110 = Port pin RG6 111 = Port pin RG7

7.3.2 CMPCFG Register (Comparator Configuration)

7	6	5	4	1	0
CMPEN	CMPOE	CMPHYS	Reserved		CMPRES

Name	Description
CMPEN	Comparator enable bit 0 = Comparator disabled 1 = Comparator enabled
CMPOE	Comparator output enable bit 0 = Comparator output disabled. 1 = Comparator output enabled on port pin RG0.
CMPHYS	Comparator hysteresis enable bit 0 = Hysteresis disabled 1 = Hysteresis enabled
CMPRES	Comparator result (read-only) 0 = RG2 voltage > RG1 1 = RG1 voltage > RG2

7.3.3 FCFG Register (Flash Configuration)

7	6	5	4	3	2	1	0
FRDTS1:0		FRDTC1:0		FWRT3:0			

Name	Description
FRDTS1:0	The system clock frequency is automatically reduced (if necessary) when executing out of flash memory to prevent the flash memory access time from being exceeded. The FRDTS1:0 bits specify the minimum number of system clock cycles required for instruction execution from flash memory. The actual execution speed from flash memory will be the slower of the speed indicated in the SPDREG register and the speed specified by the FRDTS1:0 bits. 00 = 1 cycle per flash access, for 0–40 MHz system clock frequency 01 = 2 cycles per flash access, for 40–80 MHz system clock frequency 10 = 3 cycles per flash access, for 80–120 MHz system clock frequency 11 = 4 cycles per flash access, for 120–160 MHz system clock frequency
FRDTC1:0	The number of CPU core cycles for reading the flash memory using an **`fread`** instruction must be specified to prevent the flash memory access time from being exceeded. Because the CPU core is subject to changes in speed, the value programmed in these bits should be appropriate for the fastest speed that might be used (typically, the faster of the main line code and the interrupt service routine). The FRDTC1:0 bits specify the number of CPU core clock cycles required for read access. 00 = 1 cycle per flash access, for 0–40 MHz CPU core clock frequency 01 = 2 cycles per flash access, for 40–80 MHz CPU core clock frequency 10 = 3 cycles per flash access, for 80–120 MHz CPU core clock frequency 11 = 4 cycles per flash access, for 120–160 MHz CPU core clock frequency
FWRT3:0	The flash memory erase and write timing is derived from the CPU core clock through a programmable divider, which is controlled by the FWRT3:0 bits. The time base must be 1 to 2 microseconds. Below 1 microsecond, the flash memory will be underprogrammed, and data retention is not guaranteed. Above 2 microseconds, the flash memory will be overprogrammed, and reliability is not guaranteed. Because the minimum flash write clock divisor is 2, the minimum clock frequency for self-programming is 1 MHz. 0000 = Clock divisor is 2, for 1–2 MHz CPU core clock frequency 0001 = Clock divisor is 3, for 2–3 MHz CPU core clock frequency 0010 = Clock divisor is 4, for 3–4 MHz CPU core clock frequency 0011 = Clock divisor is 6, for 4–6 MHz CPU core clock frequency 0100 = Clock divisor is 8, for 6–8 MHz CPU core clock frequency 0101 = Clock divisor is 12, for 8–12 MHz CPU core clock frequency 0110 = Clock divisor is 16, for 12–16 MHz CPU core clock frequency 0111 = Clock divisor is 24, for 16–24 MHz CPU core clock frequency 1000 = Clock divisor is 32, for 24–32 MHz CPU core clock frequency 1001 = Clock divisor is 48, for 32–48 MHz CPU core clock frequency 1010 = Clock divisor is 64, for 48–64 MHz CPU core clock frequency

Name	Description
	1011 = Clock divisor is 96, for 64–96 MHz CPU core clock frequency 1100 = Clock divisor is 128, for 96–128 MHz CPU core clock frequency 1101 = Clock divisor is 192, for 128–192 MHz CPU core clock frequency 1110 = Clock divisor is 256 1111 = Clock divisor is 384

7.3.4 INTSPD Register

7	6	5	4	3	0
PWRD1:0		CLK1:0		CDIV3:0	

Name	Description
PWRD1	Controls PLL clock multiplier operation. If the PLL is not required, power consumption can be reduced by disabling it. 0 = PLL clock multiplier enabled 1 = PLL clock multiplier disabled
PWRD0	Controls OSC oscillator operation. If the oscillator is not required, power consumption can be reduced by disabling it. 0 = OSC oscillator enabled 1 = OSC oscillator disabled
CLK1:0	Selects the system clock source. 00 = PLL clock multiplier 01 = OSC oscillator/external clock on OSC1 input 10 = RTCLK oscillator/external clock on RTCLK1 input 11 = System clock disabled (off)
CDIV3:0	Selects the system clock divisor. 0000 = 1 0001 = 2 0010 = 3 0011 = 4 0100 = 5 0101 = 6 0110 = 8 0111 = 10 1000 = 12 1001 = 16 1010 = 24 1011 = 32 1100 = 48 1101 = 64 1110 = 128 1111 = System clock disabled (off)

7.3.5 PSPCFG Register (Parallel Slave Peripheral Configuration)

7	6	5	4	3	0
PSPEN2	PSPEN1	PSPHEN	PSPRDY	Reserved	

Name	Description
PSPEN2	Port D enable bit 0 = Port D is available for general-purpose I/O 1 = Port D is configured for the parallel slave peripheral interface
PSPEN1	Port C enable bit 0 = Port C is available for general-purpose I/O 1 = Port C is configured for the parallel slave peripheral interface
PSPHEN	$\overline{\text{HOLD}}$ output enable bit 0 = $\overline{\text{HOLD}}$ output disabled. Port pin RB5 available for general-purpose I/O. 1 = $\overline{\text{HOLD}}$ output enabled on port pin RB5.
PSPRDY	Ready bit 0 = This bit always reads as zero. 1 = Write 1 to release $\overline{\text{HOLD}}$ when the IP2022 is ready to allow the data transfer to complete.

7.3.6 RTCFG Register (Real-Time Timer Configuration)

7	6	3	2	1	0
RTEN	RTPS3:0		RTSS	RTIE	RTIF

Name	Description
RTEN	Real-Time Timer enable bit 0 = Real-Time Timer disabled 1 = Real-Time Timer enabled
RTPS3:0	Real-Time Timer prescaler divisor 0000 = 1 0001 = 2 0010 = 4 0011 = 8 0100 = 16 0101 = 32 0110 = 64 0111 = 128 1000 = 256 1001 = 512 1010 = 1024 1011 = 2048 1100 = 4096 1101 = 8192 1110 = 16384 1111 = 32768
RTSS	Real-Time Timer clock source select 0 = pre-PLL clock 1 = external RTCLK
RTIE	Real-Time Timer interrupt enable bit 0 = Real-Time Timer interrupt disabled 1 = Real-Time Timer interrupt enabled
RTIF	Real-Time Timer interrupt flag 0 = No timer overflow has occurred since this bit was last cleared 1 = Timer overflow has occurred

7.3.7 SxINTF Register

7	6	5	4	3	2	1	0
RXERROR	EOP	SYND	TXBE	TXEOP	SXLINKPULSE	RXBF	RXBUSY

Name	Description
RXERROR	Receive error in preamble (Manchester encoding only) 0 = Receive error has not been detected since this bit was last cleared 1 = Double zero has been detected in preamble (not affected by double zero in data)
EOP	End-of-Packet detection interrupt flag (USB and 10Base-T modes only) 0 = End-of-Packet has not been detected since this bit was last cleared 1 = End-of-Packet has been detected
SYND	Synchronization pattern detection interrupt flag (USB and 10Base-T modes only) 0 = Synchronization pattern has not been detected since this bit was last cleared 1 = Synchronization pattern has been detected
TXBE	Transmit buffer empty interrupt flag 0 = Transmit buffer has not been empty since this bit was last cleared 1 = Transmit buffer has been empty

Name	Description
TXEOP	Transmit underrun. This bit is set when the previous data in the transmit buffer register (SxTXBUF) has been transmitted and no new data has been loaded in the register. In USB and 10Base-T modes, this causes an EOP condition to be generated. 0 = Transmit underrun has not occurred since this bit was last cleared 1 = Transmit underrun has occurred
SXLINKPULSE	Set after a link pulse of 75 to 250 ns duration is detected. 0 = No link pulse has been detected since this bit was last cleared 1 = Link pulse detected
RXBF	Receive buffer full interrupt flag 0 = Receive buffer has not been full since this bit was last cleared 1 = Receive buffer has been full
RXBUSY	Receive buffer busy interrupt flag. This bit can be used to check whether data is being received before entering a low-power mode. 0 = No new data has been received since this bit was last cleared 1 = New data has been (or is being) received

7.3.8 SxMODE Register

7	6	5	4	3	2	1	0
PRS3:0				SUBM1:0		CLKS1:0	

Name	Description
PRS3:0	Protocol select. Unassigned encodings are reserved. 0000 = Disabled 0001 = 10Base-T 0010 = USB Bus 0011 = UART 0100 = I^2C 0101 = SPI/Microwire
SUBM1:0	Submode select (in USB mode): 01 = Low-speed USB interface 10 = High-speed USB interface Submode select (in Microwire and SPI modes): 00 = Receive on falling edge, transmit on rising edge 01 = Receive on rising edge, transmit on rising edge 10 = Receive on falling edge, transmit on falling edge 11 = Receive on rising edge, transmit on falling edge
CLKS1:0	Clock source select 00 = Clock disabled 01 = Reserved 10 = OSC clock oscillator 11 = PLL clock multiplier

7.3.9 SxRCFG Register

7	6	5	4	0
Reserved	$\overline{\text{SYNCMP}}$	RPOREV	RXSCNT4:0	

Name	Description
SYNCMP	Synchronization pattern detection disable bit 0 = Synchronization pattern detection enabled 1 = Synchronization pattern detection disabled
RPOREV	Receive data polarity reversal select 0 = Data polarity uninverted 1 = Data polarity inverted
RXSCNT4:0	Receive shift count, specifies number of bits to receive

7.3.10 SxRCNT Register

7	6	5	4	0
Reserved	RxCRSCTRL1:0	RXACNT4:0		

Name	Description
RxCRSCTRL1:0	Carrier sense status and interrupt control 0x = No interrupt. Software can poll the RXXCRS bit in the SxINTF register for carrier status. 10 = Intterupt on carrier detection 11 = Interrupt on loss of carrier
RXACNT4:0	Receive shift count, actual number of bits received (read-only)

7.3.11 SxTCFG Register

7	6	5	4	0
SEREN	Reserved	TPOREV	TXSCNT4:0	

Name	Description
SEREN	Serializer enable bit 0 = Serializer disabled 1 = Serializer enabled
TPOREV	Transmit data polarity reversal select 0 = Data polarity uninverted 1 = Data polarity inverted
TXSCNT4:0	Transmit shift count, specifies number of bits to transmit

7.3.12 SPDREG Register

7	6	5	4	3	0
PWRD1:0		CLK1:0		CDIV3:0	

Name	Description
PWRD1	Controls PLL clock multiplier operation. If the PLL is not required, power consumption can be reduced by disabling it. 0 = PLL clock multiplier enabled 1 = PLL clock multiplier disabled
PWRD0	Controls OSC oscillator operation. If the oscillator is not required, power consumption can be reduced by disabling it. 0 = OSC oscillator enabled 1 = OSC oscillator disabled
CLK1:0	Selects the system clock source. 00 = PLL clock multiplier 01 = OSC oscillator/external clock on OSC1 input 10 = RTCLK oscillator/external clock on RTCLK1 input 11 = System clock disabled (off)

Name	Description
CDIV3:0	Selects the system clock divisor. 0000 = 1 0001 = 2 0010 = 3 0011 = 4 0100 = 5 0101 = 6 0110 = 8 0111 = 10 1000 = 12 1001 = 16 1010 = 24 1011 = 32 1100 = 48 1101 = 64 1110 = 128 1111 = System clock disabled (off)

7.3.13 STATUS Register

7 5	4	3	2	1	0
PA2:0	WD	BO	Z	DC	C

Name	Description
PA2:0	Program memory page select bits. Used to extend the 13-bit address encoded in jump and call instructions. Modified using the **page** instruction.
WD	Watchdog time-out bit. Set at reset, if reset was triggered by Watchdog Timer overflow, otherwise cleared. 0 = Last reset was not caused by a Watchdog Timer overflow. 1 = Last reset was caused by a Watchdog Timer overflow.
BO	Brown-out reset bit. Set at reset, if reset was triggered by brown-out voltage level detection, otherwise cleared. 0 = Last reset was not caused by the brown-out voltage detector sensing a low-voltage condition. 1 = Last reset was caused by the brown-out voltage detector sensing a low-voltage condition.
Z	Zero bit. Affected by most logical, arithmetic, and data movement instructions. Set if the result was zero, otherwise cleared. 0 = Result of last ALU operation was non-zero. 1 = Result of last ALU operation was zero.
DC	Digit Carry bit. After addition, set if carry from bit 3 occurred, otherwise cleared. After subtraction, cleared if borrow from bit 3 occurred, otherwise set. 0 = Last addition did not generate carry out of bit 3, or last subtraction generated borrow out of bit 3. 1 = Last addition generated carry out of bit 3, or last subtraction did not generate borrow out of bit 3.
C	Carry bit. After addition, set if carry from bit 7 of the result occurred, otherwise cleared. After subtraction, cleared if borrow from bit 7 of the result occurred, otherwise set. After rotate (**rr** or **rl**) instructions, loaded with the LSB or MSB of the operand, respectively. 0 = Last addition did not generate carry out of bit 7, last subtraction generated borrow out of bit 7, or last rotate loaded a 0. 1 = Last addition generated carry out of bit 7, last subtraction did not generate borrow out of bit 7, or last rotate loaded a 1.

7.3.14 T0CFG Register (Timer 0 Configuration)

7	6	3	2	1	0
T0EN	T0PS3:0		Rsrvd.	T0IE	T0IF

Name	Description
T0EN	Enables Timer 0 0 = Timer 0 disabled 1 = Timer 0 enabled
T0PS3:0	Specifies Timer 0 prescaler divisor 0000 = 1 0001 = 2 0010 = 4 0011 = 8 0100 = 16 0101 = 32 0110 = 64 0111 = 128 1000 = 256 1001 to 1111 = Reserved
T0IE	Timer 0 interrupt enable bit 0 = Timer 0 interrupt disabled 1 = Timer 0 interrupt enabled
T0IF	Timer 0 interrupt flag 0 = No timer overflow has occurred since this bit was last cleared 1 = Timer overflow has occurred

7.3.15 TxCFG1H/TxCFG1L Register

15	14	13	12	11	10	9	8
OFIE	CAP2IE CMP2IE	CAP1IE	CMP1IE	OFIF	CAP2IF CMP2IF	CAP1IF	CMP1IF

7	6	5	4	3	2	1	0
MODE	OEN	ECLKEN	CPI2EN	CPI1EN	ECLKEDG	CAP1RST	TMREN

Name	Description
OFIE	Timer overflow interrupt enable bit 0 = Overflow interrupt disabled 1 = Overflow interrupt enabled
CAP2IE CMP2IE	PWM mode: Compare 2 interrupt enable bit Capture/Compare mode: Capture 2 interrupt enable bit 0 = Capture/Compare 2 interrupt disabled 1 = Capture/Compare 2 interrupt enabled
CAP1IE	Capture 1 interrupt enable bit 0 = Capture 1 interrupt disabled 1 = Capture 1 interrupt enabled
CMP1IE	Compare 1 interrupt enable bit 0 = Compare 1 interrupt disabled 1 = Compare 1 interrupt enabled
OFIF	Timer overflow interrupt flag 0 = No timer overflow has occurred since this bit was last cleared 1 = Timer overflow has occurred

Name	Description
CAP2IF CMP2IF	PWM mode: Compare 2 interrupt flag (i.e. timer value matched TxCMP2 value) Capture/Compare mode: Capture 2 flag (i.e. TxCPI2 input triggered) 0 = No capture/compare 2 event has occurred since this bit was last cleared 1 = Capture/compare 2 event has occurred
CAP1IF	Capture 1 interrupt flag 0 = No capture 1 event has occurred since this bit was last cleared 1 = Capture 1 event has occurred
CMP1IF	Compare 1 interrupt flag 0 = No compare 1 event has occurred since this bit was last cleared 1 = Compare 1 event has occurred
MODE	Timer mode select 0 = PWM/timer mode 1 = Capture/compare mode
OEN	TxOUT enable bit 0 = TxOUT disabled. Port pin available for general-purpose I/O. 1 = TxOUT enabled. Port pin must be configured for output in corresponding RxDIR register bit.
ECLKEN	TxCLK enable bit 0 = TxCLK disabled. Port pin available for general-purpose I/O. 1 = TxCLK enabled as clock source for timer. Enabling this bit does not make any other restrictions on the use of the TxCLK port pin for general-purpose I/O.

Name	Description
CPI2EN	TxCPI2 enable bit 0 = System clock enabled as clock source for timer. TxCPI2 port pin available for general-purpose I/O. 1 = TxCLK enabled as clock source for timer. Enabling this bit does not make any other restrictions on the use of the port pin for general-purpose I/O.
CPI1EN	TxCPI1 enable bit 0 = Capture 1 input disabled. TxCPI1 port pin available for general-purpose I/O. 1 = TxCPI1 enabled as capture 1 input. Enabling this bit does not make any other restrictions on the use of the port pin for general-purpose I/O.
ECLKEDG	TxCLK edge sensitivity select. (This bit is ignored if the ECLKEN bit is clear.) 0 = TxCLK increments timer on rising edge 1 = TxCLK increments timer on falling edge
CAP1RST	Reset timer on capture 1 event enable bit 0 = Timer value unchanged by occurrence of a capture 1 event 1 = Timer value cleared by occurrence of a capture 1 event
TMREN	Timer enable bit 0 = Timer disabled. Timer clock source shut off to reduce power consumption. 1 = Timer enabled

7.3.16 TxCFG2H/TxCFG2L Register

15	14	13	12	11	8
0	0	0	0	PS3:0	

7	6	5	4	3	2	1	0
Reserved	TOUTSET	TOUTCLR	CPI2CPI1	CPI2EDG1:0		CPI1EDG1:0	

Name	Description
PS3:0	Timer prescaler divisor 0000 = 1 0001 = 2 0010 = 4 0011 = 8 0100 = 16 0101 = 32
	0110 = 64 0111 = 128 1000 = 256 1001 = 512 1010 = 1024 1011 = 2048 1100 = 4096 1101 = 8192 1110 = 16384 1111 = 32768
TOUTSET	Override bit to set the TxOUT output. This bit always reads as zero. 0 = Writing 0 to this bit has no effect 1 = Writing 1 to this bit forces the TxOUT signal high
TOUTCLR	Override bit to clear the TxOUT output. This bit always reads as zero. 0 = Writing 0 to this bit has no effect 1 = Writing 1 to this bit forces the TxOUT signal low
CPI2CPI1	Internally connect the TxCPI2 input to the TxCPI1 input. This makes the TxCPI2 port pin available for general-purpose I/O. 0 = No internal connection between TxCPI1 and TxCPI2 1 = TxCPI1 and TxCPI2 internally connected
CPI2EDG1:0	TxCPI2 edge sensitivity select 00 = Falling edge on TXCPI2 recognized as capture 2 event 01 = Rising edge on TXCPI2 recognized as capture 2 event 10 = Any falling or rising edge on TXCPI2 recognized as capture 2 event 11 = Any falling or rising edge on TXCPI2 recognized as capture 2 event
CPI1EDG1:0	TxCPI1 edge sensitivity select 00 = Falling edge on TXCPI1 recognized as capture 1 event 01 = Rising edge on TXCPI1 recognized as capture 1 event 10 = Any falling or rising edge on TXCPI1 recognized as capture 1 event 11 = Any falling or rising edge on TXCPI1 recognized as capture 1 event

7.3.17 TCTRL Register

7	6	5	4	3	2	1	0
0	0	T2IE	T1IE	0	0	T2RST	T1RST

Name	Description
T2IE	Timer 2 interrupt enable 0 = Timer 2 interrupt disabled 1 = Timer 2 interrupt enabled
T1IE	Timer 1 interrupt enable 0 = Timer 1 interrupt disabled 1 = Timer 1 interrupt enabled
T2RST	Timer 2 reset bit. This bit always reads as zero. 0 = Writing 0 to this bit has no effect. 1 = Writing 1 to this bit clears Timer 2.
T1RST	Timer 1 reset bit. This bit always reads as zero. 0 = Writing 0 to this bit has no effect. 1 = Writing 1 to this bit clears Timer 1.

7.3.18 XCFG Register

7	6	5	4	1	0
GIE	FWP	RTEOS	Reserved		FBUSY

Name	Description
GIE	Global interrupt enable bit 0 = Interrupts disabled 1 = Interrupts enabled
FWP	Flash write protect bit. This bit only affects operation of the self-programming instructions, not programming through the ISD/ISP interface. 0 = Writes to flash memory disabled 1 = Writes to flash memory enabled
RTEOS	Real-time timer oversampling bit 0 = Oversampling enabled 1 = Oversampling disabled
FBUSY	Flash memory busy bit (read-only) 0 = Flash memory is idle 1 = Flash memory is busy with a previous operation

8.0 Electrical Characteristics

8.1 Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units
Ambient temperature under bias (commercial temp. range)	0	70	°C
Ambient temperature under bias (industrial temp. range, excluding Flash programming)	-40	85	°C
Ambient temperature under bias (Flash programming)	0	70	°C
Storage temperature			
Voltage on DVdd with respect to Vss	-0.4	4.5	V
Voltage on AVdd with respect to Vss	-0.4	4.5	V
Voltage on GVdd with respect to Vss	-0.4	4.5	V
Voltage on XVdd with respect to Vss	-0.4	4.5	V
Voltage on IOVdd with respect to Vss	-0.4	5.7	V
Voltage on OSC1/RTCLK1 with respect to Vss	-0.4	5.7	V
Voltage on $\overline{TSS}$, TSCK, TSI with respect to Vss	-0.4	5.7	V
Voltage on all other digital inputs with respect to Vss	-0.4	5.7	V
Voltage on all other analog inputs with respect to Vss	-0.4	4.5	V
Total power dissipation			
Maximum current out of Vss pins			
Maximum current into Vdd pins			
Maximum DC current into an input pin (with internal protection diode forward biased)			
Input clamp current, I_{ik} ($V_i < 0$ or $V_i > Vdd$)			
Output clamp current, I_{ok} ($V_O < 0$ or $V_O > Vdd$)			
Maximum allowable sink current per I/O pin			
Maximum allowable source current per I/O pin			
Maximum allowable sink current per group of I/O pins between Vdd pins			
Maximum allowable source current per group of I/O pins between Vdd pins			
Latchup			
θ_{JA}, 80-pin PQFP Package			
Flash erase cycle lifetime			Cycles
Flash write cycle lifetime			Cycles
ESD Human Body Model - all pins			
ESD Machine Model - all pins			

8.2 DC Characteristics

Operating Temperature 0°C ≤ Ta ≤ +70°C (Commercial) or -40°C ≤ Ta ≤ +85°C (Industrial)

Symbol	Parameter	Min	Typ	Max	Units	Conditions
DVdd	Digital supply voltage	2.3	2.5	2.7	V	
AVdd	Analog supply voltage	2.3	2.5	2.7	V	≤ DVdd
GVdd	Port G supply voltage	2.3	2.5	2.7	V	≤ DVdd
XVdd	Crystal oscillator supply voltage	2.3	2.5	2.7	V	≤ DVdd
IOVdd	I/O supply voltage (except Port G)	2.3	2.5/3.3	3.6	V	
Idd	Supply current, full operation				mA	
Ics	Supply current, sleep				µA	PLL and oscillators off
DVih, DVil	Input voltage, digital inputs	Vss		5.5	V	
XVih, XVil	Input voltage, OSC1 and RTCLK1 inputs	Vss		5.5	V	
TVih, TVil	Input voltage, $\overline{TSS}$, TSCK, and TSI inputs	Vss		5.5	V	
Iil	Input low current				µA	
Iih	Input high current				µA	
Voh, Vol	Output high voltage				V	Voh = 0.7 × IOVdd Vol = 0.3 × IOVdd

1. Vdd must start rising from Vss to ensure proper Power-On-Reset when relying on the internal Power-On-Reset circuitry.
2. Data in the Typical (“Typ”) column is at 5V, 25°C unless otherwise stated.

8.3 AC Characteristics

Operating Temperature 0°C ≤ Ta ≤ +70°C (Commercial) or -40°C ≤ Ta ≤ +85°C (Industrial)

Symbol	Parameter	Min	Typ	Max	Units	Conditions
Fosc	External OSC1 clock frequency	0		150	MHz	
	Oscillator frequency (OSC1)	1		6	MHz	Crystal/resonator
TosL, TosH	Clock in (OSC1) low or high Time					
Fosc	External RTCLK1 clock frequency	0		100	MHz	
	Oscillator frequency (RTCLK1)	32.768		32.768	kHz	Crystal
TosL, TosH	Clock in (RTCLK1) low or high time					

8.4 Analog Comparator DC and AC Specifications

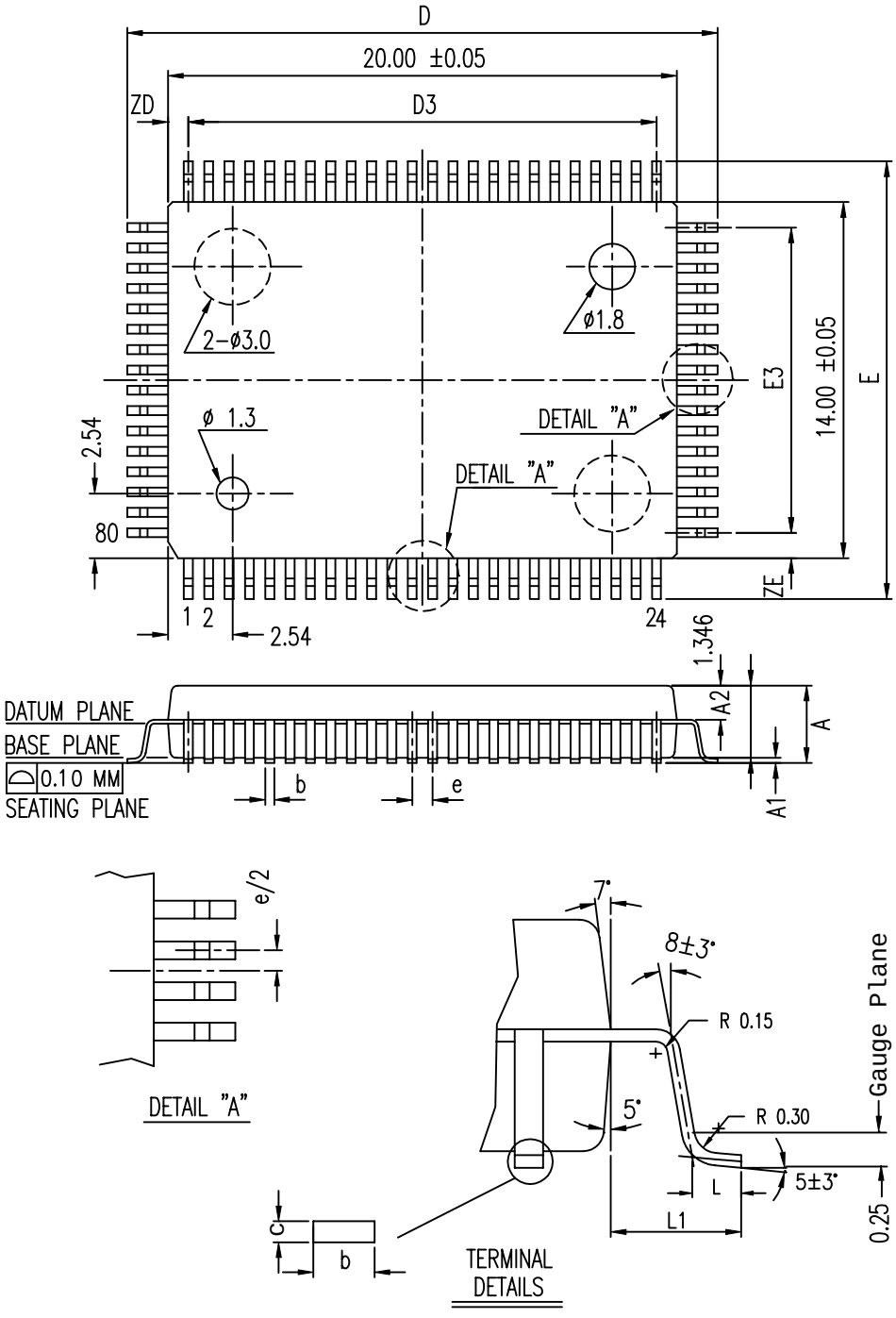
Parameter	Min	Typ	Max	Units	Conditions
Input Offset Voltage				V	
Input Common Mode Voltage Range				V	
Voltage Gain					
Response Time				µs	

8.5 A/D Converter DC and AC Specifications

Parameter	Min	Typ	Max	Units	Conditions
Conversion Time				µs	
Delay Time Between Conversions				µs	
Maximum Sampling Rate				ksps	
Total Unadjusted Error			½	LSB	
Vref				V	
Vin				V	
Vin Leakage Current				µA	
Total Harmonic Distortion (THD)				%	

9.0 Package Dimensions (in millimeters)

80-pin, 14 mm × 20 mm × 2.8 mm body, 0.8 mm pitch, 17.9 × 23.9 tip-to-tip

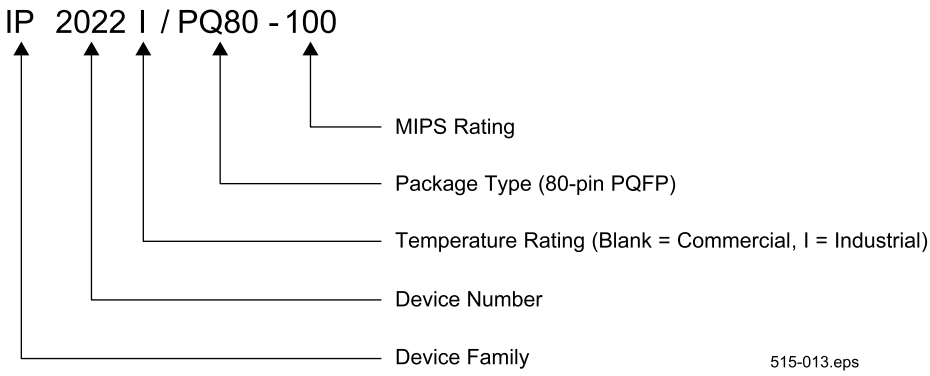


b (TYP.)	0.375
c (TYP.)	0.175
e (TYP.)	0.80
L1 ±0.17	1.95
L ±0.15	0.80
ZE (TYP.)	1.00
E3 (TYP.)	12.00
E ±0.25	17.90
ZD (TYP.)	0.80
D3 (TYP.)	18.40
D ±0.25	23.90
A1 ±0.08	0.178
A ±0.1	3.023
A2 ±0.05	2.845
N	80 L
JEDEC	MO-112 CB-2

10.0 Part Numbering

Table 10-1 Ordering Information

Device	Pins	I/O	Program Flash (Bytes)	Program RAM (Bytes)	Data RAM (Bytes)
IP2022I/PQ80-100	80	52	64K (32K x 16)	16K (8K x 16)	4K



Lit. #:: SXL-DS05-01

Sales and Tech Support Contact Information

For the latest contact and support information on IP devices, please visit the Ubicom website at www.ubicom.com. The site contains technical literature, local sales contacts, tech support, and many other features.

Ubicom, Inc.
1330 Charleston Road
Mountain View, CA 94043

Tel.: (650) 210-1500
Fax: (650) 210-8715
E-Mail: sales@ubicom.com
Web Site: www.ubicom.com