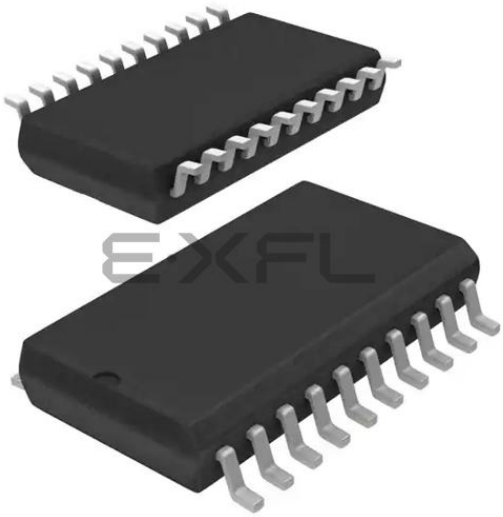

**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | M8C |
| Core Size | 8-Bit |
| Speed | 24MHz |
| Connectivity | SPI, UART/USART |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 16 |
| Program Memory Size | 8KB (8K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 256 x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 5.25V |
| Data Converters | A/D 1x8b, 1x11b, 1x12b; D/A 1x9b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-SOIC (0.295", 7.50mm Width) |
| Supplier Device Package | 20-SOIC |
| Purchase URL | https://www.e-xfl.com/product-detail/infineon-technologies/cy8c26233-24sxi |

This page has intentionally been left blank.

## List of Figures


Figure 1: Block Diagram ..... 13
Figure 2: CY8C25122 ..... 15
Figure 3: CY8C26233 ..... 15
Figure 4: 26443 PDIP/SOIC/SSOP ..... 16
Figure 5: 26643 TQFP ..... 17
Figure 6: 26643 PDIP/SSOP ..... 18
Figure 7: General Purpose I/O Pins ..... 30
Figure 8: External Crystal Oscillator Connections ..... 37
Figure 9: PSoC MCU Clock Tree of Signals ..... 39
Figure 10: Interrupts Overview ..... 43
Figure 11: GPIO Interrupt Enable Diagram ..... 47
Figure 12: Digital Basic and Digital Communications PSoC Blocks ..... 49
Figure 13: Polynomial LFSR ..... 65
Figure 14: Polynomial PRS ..... 65
Figure 15: SPI Waveforms ..... 68
Figure 16: Array of Analog PSoC Blocks ..... 72
Figure 17: Analog Reference Control Schematic ..... 73
Figure 18: NMux Connections ..... 78
Figure 19: PMux Connections ..... 79
Figure 20: RBotMux Connections ..... 79
Figure 21: Analog Continuous Time PSoC Blocks ..... 81
Figure 22: Analog Switch Cap Type A PSoC Blocks ..... 86
Figure 23: AMux Connections ..... 87
Figure 24: CMux Connections ..... 87
Figure 25: BMuxSCA/SCB Connections ..... 88
Figure 26: Analog Switch Cap Type B PSoC Blocks ..... 95
Figure 27: Analog Input Muxing ..... 103
Figure 28: Analog Output Buffers ..... 105
Figure 29: Multiply/Accumulate Block Diagram ..... 110
Figure 30: Decimator Coefficients ..... 112
Figure 31: Execution Reset ..... 115
Figure 32: Three Sleep States ..... 117
Figure 33: Switch Mode Pump ..... 119
Figure 34: Programming Wave Forms ..... 124
Figure 35: PSoC Designer Functional Flow ..... 125
Figure 36: CY8C25xxx/CY8C26xxx Voltage Frequency Graph ..... 127
Figure 37: 44-Lead Thin Plastic Quad Flat Pack A44 ..... 143
Figure 38: 20-Pin Shrunk Small Outline Package O20 ..... 144
Figure 39: 28-Lead (210-Mil) Shrunk Small Outline Package O28 ..... 145
Figure 40: 48-Lead Shrunk Small Outline Package O48 ..... 145
Figure 41: 20-Lead (300-Mil) Molded DIP P5 ..... 146
Figure 42: 28-Lead (300-Mil) Molded DIP P21 ..... 146
Figure 43: 48-Lead (600-Mil) Molded DIP P25 ..... 147
Figure 44: 20-Lead (300-Mil) Molded SOIC S5 ..... 147
Figure 45: 28-Lead (300-Mil) Molded SOIC S21 ..... 148

**Examples**:

```
                        ;In this case, the
                        ;value in the memory
                        ;location at address
                        ;X+7 is added with
ADD   [X+7],    5       ;the immediate value
                        ;of 5, and the result
                        ;is placed in the
                        ;memory location at
                        ;address X+7.
                        ;In this case, the
                        ;immediate value of 6
MOV   REG[X+8], 6       ;is moved into the
                        ;location in the
                        ;register space at
                        ;address X+8.
```

### 2.3.8 Destination Direct Direct

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

**Table 20: Destination Direct Direct**

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

**Example**:

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

### 2.3.9 Source Indirect Post Increment

The result of an instruction using this addressing mode is placed in the Accumulator. Operand 1 is an address pointing to a location within the memory space, which contains an address (the indirect address) for the source of the instruction. The indirect address is incremented as part of the instruction execution. This addressing mode is only valid on the MVI instruction. The instruction using this addressing mode is two bytes in length. See **Section 7. Instruction Set** in *PSoC Designer: Assembly Language User Guide* for further details on MVI instruction.

**Table 21: Source Indirect Post Increment**

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

**Example**:

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

### 2.3.10 Destination Indirect Post Increment

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

**Table 22: Destination Indirect Post Increment**

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

**Example**:

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

## 2.4 Instruction Set Summary

**Table 23: Instruction Set Summary (Sorted by Mnemonic)**

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

**Note**: Interrupt acknowledge to Interrupt Vector table = 13 cycles.

### 6.3.2 Port Drive Mode 1 Registers

**Table 32: Port Drive Mode 1 Registers**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |
| Bit Name | DM1 [7] | DM1 [6] | DM1 [5] | DM1 [4] | DM1 [3] | DM1 [2] | DM1 [1] | DM1 [0] |
| **Bit [7:0]**: **DM1 [7:0]** See truth table for Port Drive Mode 0 Registers, above | | | | | | | | |

Port 0 Drive Mode 1 Register (PRT0DM1, Address = Bank 1, 01h)
Port 1 Drive Mode 1 Register (PRT1DM1, Address = Bank 1, 05h)
Port 2 Drive Mode 1 Register (PRT2DM1, Address = Bank 1, 09h)
Port 3 Drive Mode 1 Register (PRT3DM1, Address = Bank 1, 0Dh)
Port 4 Drive Mode 1 Register (PRT4DM1, Address = Bank 1, 11h)
Port 5 Drive Mode 1 Register (PRT5DM1, Address = Bank 1, 15h) **Note**: Port 5 is 4-bits wide

### 6.3.3 Port Interrupt Control 0 Registers

**Table 33: Port Interrupt Control 0 Registers**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |
| Bit Name | IC0 [7] | IC0 [6] | IC0 [5] | IC0 [4] | IC0 [3] | IC0 [2] | IC0 [1] | IC0 [0] |
| **Bit [7:0]**: **IC0 [7:0]** The two Interrupt Control bits that control a particular port pin are treated as a pair and are decoded as follows:<br>IC1 [x], IC0 [x] = 0 0 = Disabled (Default)<br>IC1 [x], IC0 [x] = 0 1 = Falling Edge (-)<br>IC1 [x], IC0 [x] = 1 0 = Rising Edge (+)<br>IC1 [x], IC0 [x] = 1 1 = Change from Last Direct Read | | | | | | | | |

Port 0 Interrupt Control 0 Register (PRT0IC0, Address = Bank 1, 02h)
Port 1 Interrupt Control 0 Register (PRT1IC0, Address = Bank 1, 06h)
Port 2 Interrupt Control 0 Register (PRT2IC0, Address = Bank 1, 0Ah)
Port 3 Interrupt Control 0 Register (PRT3IC0, Address = Bank 1, 0Eh)
Port 4 Interrupt Control 0 Register (PRT4IC0, Address = Bank 1, 12h)
Port 5 Interrupt Control 0 Register (PRT5IC0, Address = Bank 1, 16h) **Note**: Port 5 is 4-bits wide

### 7.1.5 Phase-Locked Loop (PLL) Operation

The Phase-Locked Loop (PLL) function generates the system clock with crystal accuracy. It is designed to provide a 23.986 MHz oscillator when utilized with an external 32.768 kHz crystal. Although the PLL provides crystal accuracy it requires time to lock onto the reference frequency when first starting. After the External Crystal Oscillator has been selected and enabled, the following procedure should be followed to enable the PLL and allow for proper frequency lock:

1. Select a CPU frequency of 3 MHz or less.
2. Enable the PLL.
3. Wait at least 10 ms.
4. Set CPU to a faster frequency, if desired. To do this, write the bits CPU[2:0] in the OSC_CR0 register.

   The CPU frequency will immediately change when these bits are set.

If the proper settings are selected in PSoC Designer, the above steps are automatically done in *boot.asm*.

## 7.2 System Clocking Signals

There are twelve system-clocking signals that are used throughout the device. Referenced frequencies are based on use of 32.768 kHz crystal. The names of these signals and their definitions are as follows:

**Table 39: System Clocking Signals and Definitions**

| Signal | Definition |
|---|---|
| **48M** | The direct 48 MHz output from the Internal Main Oscillator. |
| **24M** | The direct 24 MHz output from the Internal Main Oscillator. |
| **24V1** | The 24 MHz output from the Internal Main Oscillator that has been passed through a user-selectable 1 to 16 divider {F = 24 MHz / (1 to 16) = 24 MHz to 1.5 MHz}. The divider value is found in the Oscillator Control 1 Register (OSC_CR1). Note that the divider will be N+1, based on a value of N written into the register bits. |
| **24V2** | The 24V1 signal that has been passed through an additional user-selectable 1 to 16 divider {F = 24 MHz / ((1 to 16) * (1 to 16)) = 24 MHz to 93.7 kHz}. The divider value is found in the Oscillator Control 1 Register (OSC_CR1). Note that the divider will be N+1, based on a value of N written into the register bits. |
| **32K** | The multiplexed output of either the Internal Low Speed Oscillator or the External Crystal Oscillator. |
| **CPU** | The output from the Internal Main Oscillator that has been passed through a divider that has 8 user selectable ratios ranging from 1:1 to 1:256, yielding frequencies ranging from 24 MHz to 93.7 kHz. |
| **SLP** | The **32K** system-clocking signal that has been passed through a divider that has 4 user selectable ratios ranging from $1:2^6$ to $1:2^{15}$, yielding frequencies ranging from 512 Hz to 1 Hz. This signal is used to clock the sleep timer period. |

The following diagram shows the PSoC MCU Clock Tree of signals 48M through SLP:

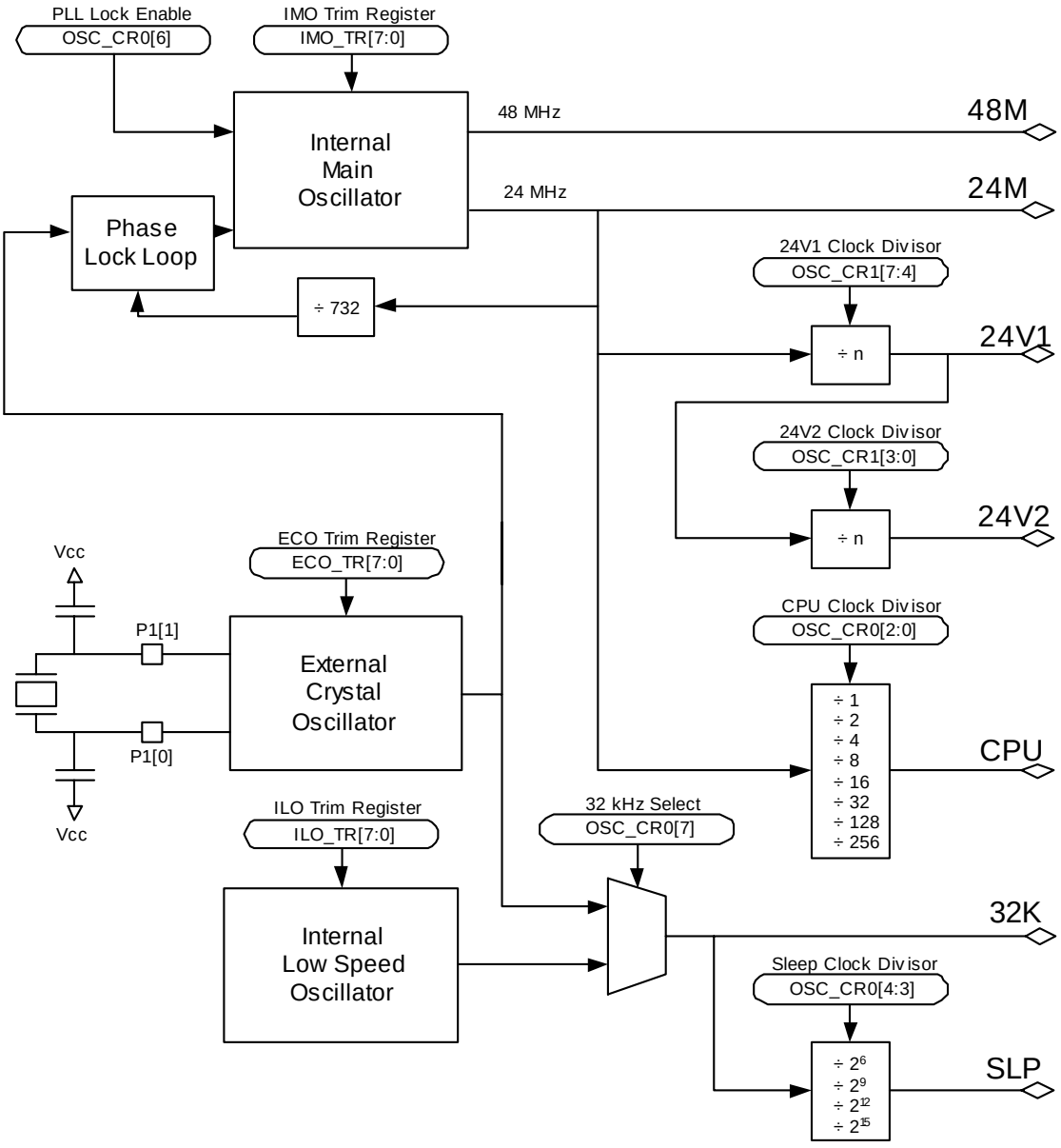


**Figure 9: PSoC MCU Clock Tree of Signals**

### 7.2.1 CPU and Sleep Timer Clock Options

The CPU is clocked off the **CPU** system-clocking signal, which can be configured to run at one of eight rates. This selection is independent from all other clock selection functions. It is completely safe for the CPU to change its clock rate without a timing hazard. The CPU clock period is determined by setting the CPU[2:0] bits in the Oscillator Control 0 Register (OSC_CR0).

The sleep timer is clocked off the **SLP** system-clocking signal. The SLEEP[1] and SLEEP[0] bits in the Oscillator Control 0 Register (OSC_CR0) allow the user to select from the four available periods.

## 8.2 Interrupt Control Architecture

The interrupt controller contains a separate flip-flop for each interrupt. When an interrupt is generated, it is registered as a pending interrupt. It will stay pending until it is serviced, a reset occurs, or there is a write to the INT_VC Register. A pending interrupt will only generate an interrupt request when enabled by the appropriate mask bit in the Digital PSoC Block Interrupt Mask Register (INT_MSK1) or General Interrupt Mask Register (INT_MSK0), and the Global IE bit in the CPU_F register is set.

Additionally, for GPIO Interrupts, the appropriate enable and interrupt-type bits for each I/O pin must be set (see section 6.0, Table 29 on page 31, Table 33 on page 33, and Table 34 on page 34). For Analog Column Interrupts, the interrupt source must be set (see section 10.10 and Table 77 on page 101).

During the servicing of any interrupt, the MSB and LSB of Program Counter and Flag registers (CPU_PC and CPU_F) are stored onto the program stack by an automatic `CALL` instruction (13 cycles) generated during the interrupt acknowledge process. The user firmware may preserve and restore processor state during an interrupt using the `PUSH` and `POP` instructions. The memory oriented CPU architecture requires minimal state saving during interrupts, providing very fast interrupt context switching. The Program Counter and Flag registers (CPU_PC and CPU_F) are restored when the `RETI` instruction is executed. If two or more interrupts are pending at the same time, the higher priority interrupt (lower priority number) will be serviced first.

After a copy of the Flag Register is stored on the stack, the Flag Register is automatically cleared. This disables all interrupts, since the Global IE flag bit is now cleared. Executing a `RETI` instruction restores the Flag register, and re-enables the Global Interrupt bit.

Nested interrupts can be accomplished by re-enabling interrupts inside an interrupt service routine. To do this, set the IE bit in the Flag Register. The user must store sufficient information to maintain machine state if this is done.

Each digital PSoC block has its own unique Interrupt Vector and Interrupt Enable bit. There are also individual interrupt vectors for each of the Analog columns, Supply Voltage Monitor, Sleep Timer and General Purpose I/Os.

## 8.3 Interrupt Vectors

**Table 43: Interrupt Vector Table**

| Address | Interrupt Priority Number | Description |
|---|---|---|
| 0x0004 | 1 | Supply Monitor Interrupt Vector |
| 0x0008 | 2 | DBA00 PSoC Block Interrupt Vector |
| 0x000C | 3 | DBA01 PSoC Block Interrupt Vector |
| 0x0010 | 4 | DBA02 PSoC Block Interrupt Vector |
| 0x0014 | 5 | DBA03 PSoC Block Interrupt Vector |
| 0x0018 | 6 | DCA04 PSoC Block Interrupt Vector |
| 0x001C | 7 | DCA05 PSoC Block Interrupt Vector |
| 0x0020 | 8 | DCA06 PSoC Block Interrupt Vector |
| 0x0024 | 9 | DCA07 PSoC Block Interrupt Vector |
| 0x0028 | 10 | Acolumn 0 Interrupt Vector |
| 0x002C | 11 | Acolumn 1 Interrupt Vector |
| 0x0030 | 12 | Acolumn 2 Interrupt Vector |
| 0x0034 | 13 | Acolumn 3 Interrupt Vector |
| 0x0038 | 14 | GPIO Interrupt Vector |
| 0x003C | 15 | Sleep Timer Interrupt Vector |
| 0x0040 | | On-Chip Program Memory Starts |

The interrupt process vectors the Program Counter to the appropriate address in the Interrupt Vector Table. Typically, these addresses contain `JMP` instructions to the start of the interrupt handling routine for the interrupt.

**Table 47: Digital Basic Type A/ Communications Type A Block xx Function Register**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | RW | RW | RW | RW | RW | RW | RW | RW |
| Bit Name | Reserved | Reserved | End | Mode 1 | Mode 0 | Function [2] | Function [1] | Function [0] |

**Bit 7**: **Reserved**
**Bit 6**: **Reserved**

**Bit 5**: **End**
0 = PSoC block is not the end of a chained function (End should not be set to 0 in block DCA07)
1 = PSoC block is the end of a chained function, or is an unchained PSoC block

**Bit 4**: **Mode 1** The definition of the Mode [1] bit depends on the block function selected
Timer: The Mode [1] bit signifies the Compare Type
0 = Less Than or Equal
1 = Less Than
Counter: The Mode [1] bit signifies the Compare Type
0 = Less Than or Equal
1 = Less Than
CRC/PRS: The Mode [1] bit is unused in this function
Deadband: The Mode [1] bit is unused in this function
UART: The Mode[1] bit signifies the Interrupt Type (Transmitter only)
0 = Transmit: Interrupt on TX_Reg Empty
1 = Transmit: Interrupt on TX Complete
SPI: The Mode[1] bit signifies the Interrupt Type
0 = Master: Interrupt on TX Reg Empty, Slave: Interrupt on RX Reg Full
1 = Master: Interrupt on SPI Complete, Slave: Interrupt on SPI Complete

**Bit 3**: **Mode 0** The definition of the Mode [0] bit depends on the block function selected
Timer: The Mode [0] bit signifies Interrupt Type
0 = Terminal Count
1 = Compare True
Counter: The Mode [0] bit signifies Interrupt Type
0 = Terminal Count
1 = Compare True
CRC/PRS: The Mode [0] bit is unused in this function
Deadband: The Mode [0] bit is unused in this function
UART: The Mode [0] bit signifies the Direction
0 = Receive
1 = Transmit
SPI: The Mode [0] bit signifies the Type
0 = Master
1 = Slave

**Bit [2:0]**: **Function [2:0]** The Function [2:0] bits select the block function which determines the basic hardware configuration
0 0 0 = Timer (chainable)
0 0 1 = Counter (chainable)
0 1 0 = CRC/PRS (Cyclical Redundancy Checker or Pseudo Random Sequencer) (chainable)
0 1 1 = Reserved
1 0 0 = Deadband for Pulse Width Modulator
1 0 1 = UART (function only available on DCA type blocks)
1 1 0 = SPI (function only available on DCA type blocks)
1 1 1 = Reserved

Digital Basic Type A Block 00 Function Register (DBA00FN, Address = Bank 1, 20h)
Digital Basic Type A Block 01 Function Register (DBA01FN, Address = Bank 1, 24h)
Digital Basic Type A Block 02 Function Register (DBA02FN, Address = Bank 1, 28h)
Digital Basic Type A Block 03 Function Register (DBA03FN, Address = Bank 1, 2Ch)
Digital Communications Type A Block 04 Function Register (DCA04FN, Address = Bank 1, 30h)

## 10.2 Analog System Clocking Signals

**Table 61: Analog System Clocking Signals**

| Signal | Definition |
|---|---|
| **ACLK0** | A system-clocking signal that is driven by the clock output of a digital PSoC block and can be selected by the user to drive the clocking signal to an analog column. Any of the 8 digital PSoC blocks can be muxed into this line using the ACLK0[2:0] bits in the Analog Clock Select Register (CLK_CR1). |
| **ACLK1** | A system-clocking signal that is driven by the clock output of a digital PSoC block and can be selected by the user to drive the clocking signal to an analog column. Any of the 8 digital PSoC blocks can be muxed into this line using the ACLK1[2:0] bits in the Analog Clock Select Register (CLK_CR1). |
| **Acolumn0** | A system-clocking signal that can drive all analog PSoC blocks in Analog Column 0. This signal is derived from the muxed input of the **24V1**, **24V2**, **ACLK0**, and **ACLK1** system clock signals. The output of this mux is then passed through a 1:4 divider to reduce the frequency by a factor of 4. The Acolumn0[1:0] bits in the CLK_CR0 Register determine the selected Column Clock. |
| **Acolumn1** | A system-clocking signal that can drive all analog PSoC blocks in Analog Column 1. This signal is derived from the muxed input of the **24V1**, **24V2**, **ACLK0**, and **ACLK1** system clock signals. The output of this mux is then passed through a 1:4 divider to reduce the frequency by a factor of 4.The Acolumn1[1:0] bits in the CLK_CR0 Register determine the selected Column Clock. |
| **Acolumn2** | A system-clocking signal that can drive all analog PSoC blocks in Analog Column 2. This signal is derived from the muxed input of the **24V1**, **24V2**, **ACLK0**, and **ACLK1** system clock signals. The output of this mux is then passed through a 1:4 divider to reduce the frequency by a factor of 4. The Acolumn2[1:0] bits in the CLK_CR0 Register determine the selected Column Clock. |
| **Acolumn3** | A system-clocking signal that can drive all analog PSoC blocks in Analog Column 3. This signal is derived from the muxed input of the **24V1**, **24V2**, **ACLK0**, and **ACLK1** system clock signals. The output of this mux is then passed through a 1:4 divider to reduce the frequency by a factor of 4. The Acolumn3[1:0] bits in the CLK_CR0 Register determine the selected Column Clock. |

## 10.3 Array of Analog PSoC Blocks

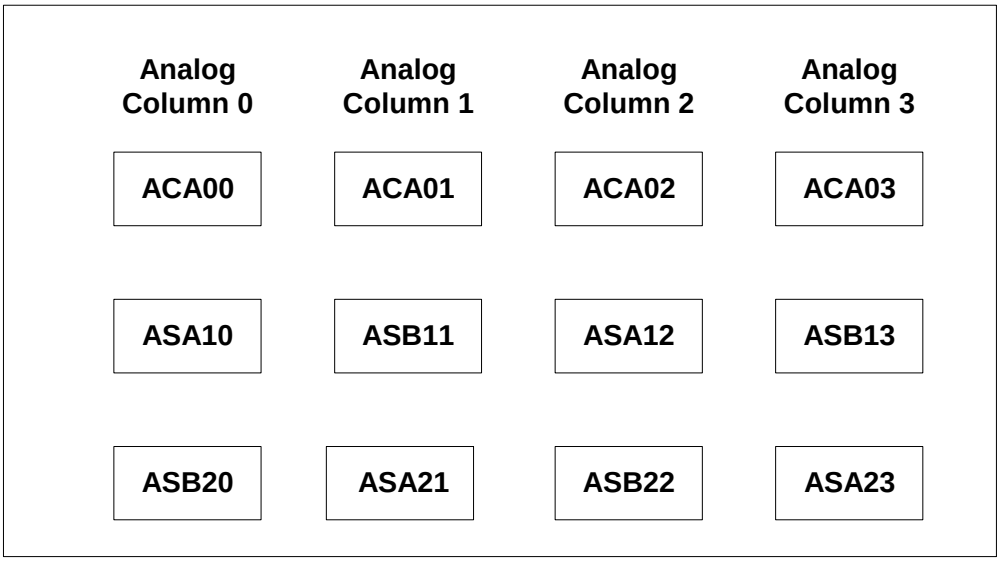


**Figure 16: Array of Analog PSoC Blocks**

inverting op-amp input) or for loss (center tap to output of the block). Note that setting Gain alone does not guarantee a gain or loss block. Routing of the other ends of the resistor determine this.

Note that connections between GIN and GOUT, and LIN and LOUT are automatically resolved by PSoC Designer when they are set in a differential configuration with an adjacent CT block.

**Table 66: Analog Continuous Time Block xx Control 0 Register**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/ Write | RW | RW | RW | RW | RW | RW | RW | RW |
| Bit Name | RTap-Mux[3] | RTap-Mux[2] | RTap-Mux[1] | RTap-Mux[0] | Gain | RTopMux | RBotMux[1] | RBotMux[0] |

**Bit [7:4]**: **RTapMux [3:0]** Encoding for selecting 1 of 16 resistor taps
0 0 0 0 = Rf 15 = Ri 01 = Loss .0625 / Gain 16.00
0 0 0 1 = Rf 14 = Ri 02 = Loss .1250 / Gain 8.000
0 0 1 0 = Rf 13 = Ri 03 = Loss .1875 / Gain 5.333
0 0 1 1 = Rf 12 = Ri 04 = Loss .2500 / Gain 4.000
0 1 0 0 = Rf 11 = Ri 05 = Loss .3125 / Gain 3.200
0 1 0 1 = Rf 10 = Ri 06 = Loss .3750 / Gain 2.667
0 1 1 0 = Rf 09 = Ri 07 = Loss .4375 / Gain 2.286
0 1 1 1 = Rf 08 = Ri 08 = Loss .5000 / Gain 2.000
1 0 0 0 = Rf 07 = Ri 09 = Loss .5625 / Gain 1.778
1 0 0 1 = Rf 06 = Ri 10 = Loss .6250 / Gain 1.600
1 0 1 0 = Rf 05 = Ri 11 = Loss .6875 / Gain 1.455
1 0 1 1 = Rf 04 = Ri 12 = Loss .7500 / Gain 1.333
1 1 0 0 = Rf 03 = Ri 13 = Loss .8125 / Gain 1.231
1 1 0 1 = Rf 02 = Ri 14 = Loss .8750 / Gain 1.143
1 1 1 0 = Rf 01 = Ri 15 = Loss .9375 / Gain 1.067
1 1 1 1 = Rf 00 = Ri 16 = Loss 1.000 / Gain 1.000

**Bit 3**: **Gain** Select gain or loss configuration for output tap
0 = Loss
1 = Gain

**Bit 2**: **RTopMux** Encoding for feedback resistor select
0 = Rtop to Vcc
1 = Rtop to op-amp's output

**Bit [1:0]**: **RBotMux [1:0]** Encoding for feedback resistor select

| | ACA00 | ACA01 | ACA02 | ACA03 |
|---|---|---|---|---|
| 0 0 = | ACA01 | ACA00 | ACA03 | ACA02 |
| 0 1 = | AGND | AGND | AGND | AGND |
| 1 0 = | Vss | Vss | Vss | Vss |
| 1 1 = | ASA10 | ASB11 | ASA12 | ASB13 |

Analog Continuous Time Block 00 Control 0 Register (ACA00CR0, Address = Bank 0/1, 71h)
Analog Continuous Time Block 01 Control 0 Register (ACA01CR0, Address = Bank 0/1, 75h)
Analog Continuous Time Block 02 Control 0 Register (ACA02CR0, Address = Bank 0/1, 79h)
Analog Continuous Time Block 03 Control 0 Register (ACA03CR0, Address = Bank 0/1, 7Dh)

## 10.9 Analog Switch Cap Type B PSoC Blocks

### 10.9.1 Introduction

The Analog Switch Cap Type B PSoC blocks are built around an operational amplifier. There are several analog muxes that are controlled by register-bit settings in the control registers that determine the signal topology inside the block. There are also four arrays of unit value capacitors that are located in the feedback path for the op-amp, and are switched by two phase clocks, PHI1 and PHI2. These four capacitor arrays are labeled A Cap Array, B Cap Array, C Cap Array, and F Cap Array. There is also an analog comparator connected to the output OUT, which converts analog comparisons into digital signals.

There are three discrete outputs from this block. These outputs are:

1. The analog output bus (ABUS), which is an analog bus resource that is shared by all of the analog blocks in the analog column for that block.

2. The comparator bus (CBUS), which is a digital bus that is a resource that is shared by all of the analog blocks in a column for that block.

3. The output bus (OUT), which is an analog bus resource that is shared by all of the analog blocks in a column and connects to one of the analog output buffers, to send a signal externally to the device.

The SCB block also supports Delta-Sigma, Successive Approximation and Incremental A/D Conversion, Capacitor DACs, and SC filters. It has two input arrays of switched capacitors, and a Non-Switched capacitor feedback array from the output. When preceded by an SC Block A Integrator, the combination can be used to provide a full Switched Capacitor Biquad.

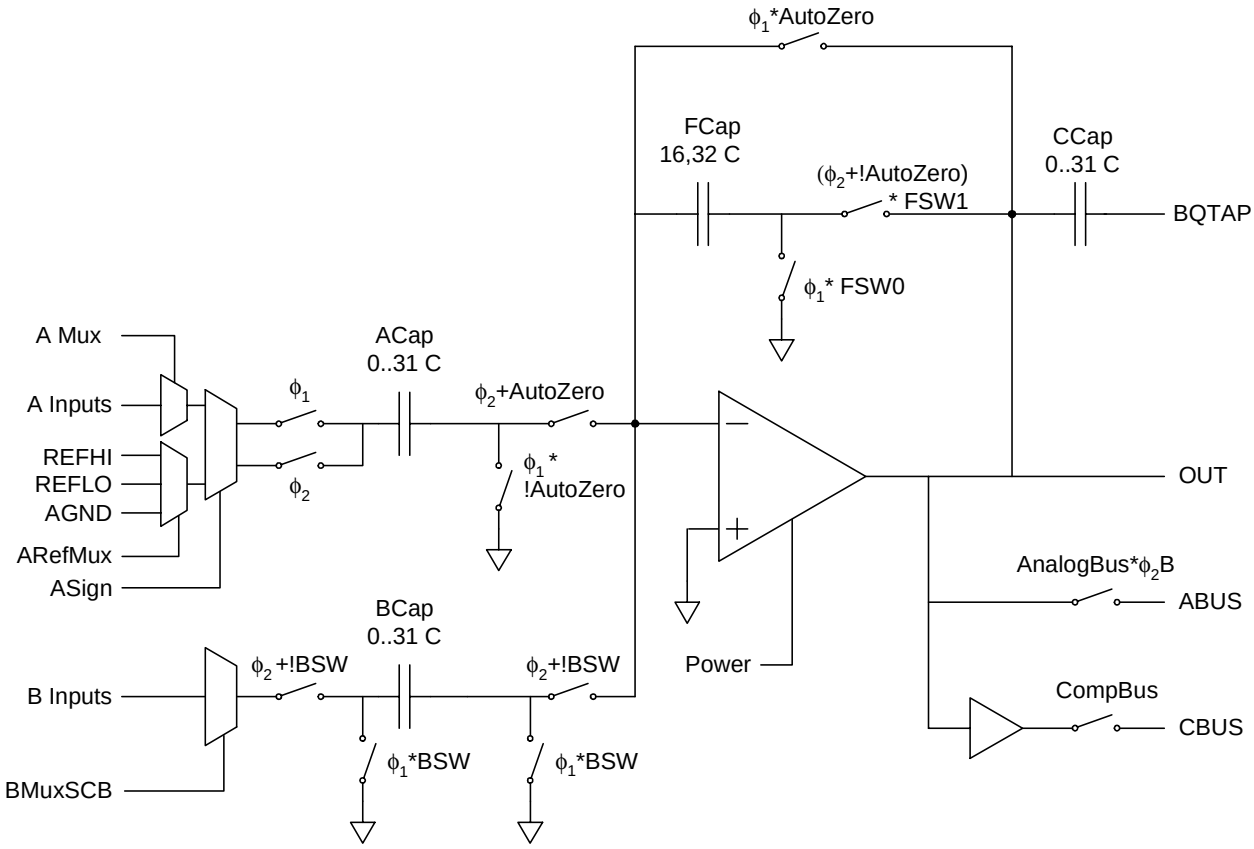


**Figure 26: Analog Switch Cap Type B PSoC Blocks**

### 10.9.2 Registers

#### 10.9.2.1 Analog Switch Cap Type B Block xx Control 0 Register

FCap controls the size of the switched feedback capacitor in the integrator.

ClockPhase controls the internal clock phasing relative to the input clock phasing. ClockPhase affects the output of the analog column bus which is controlled by the AnalogBus bit in Control 2 Register (ASB11CR2, ASB13CR2, ASB20CR2, ASB22CR2).

ASign controls the switch phasing of the switches on the bottom plate of the A capacitor. The bottom plate samples the input or the reference.

The ACap bits set the value of the capacitor in the A path.

**Table 73: Analog Switch Cap Type B Block xx Control 0 Register**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/ Write | RW | RW | RW | RW | RW | RW | RW | RW |
| Bit Name | FCap | ClockPhase | ASign | ACap[4] | ACap[3] | ACap[2] | ACap[1] | ACap[0] |

**Table 75: Analog Switch Cap Type B Block xx Control 2 Register**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/ Write | RW | RW | RW | RW | RW | RW | RW | RW |
| Bit Name | AnalogBus | CompBus | AutoZero | CCap[4] | CCap[3] | CCap[2] | CCap[1] | CCap[0] |

**Bit 7**: **AnalogBus** Enable output to the analog bus
0 = Disable output to analog column bus
1 = Enable output to analog column bus
(The output on the analog column bus is affected by the state of the ClockPhase bit in Control 0 Register (ASB11CR0, ASB13CR0, ASB20CR0, ASB22CR0). If AnalogBus is set to 0, the output to the analog column bus is tri-stated. If AnalogBus is set to 1, the ClockPhase bit selects the signal that is output to the analog column bus. If the ClockPhase bit is 0, the block output is gated by sampling clock on last part of PHI2. If the ClockPhase bit is 1, the block output continuously drives the analog column bus)

**Bit 6**: **CompBus** Enable output to the comparator bus
0 = Disable output to comparator bus
1 = Enable output to comparator bus

**Bit 5**: **AutoZero** Bit for controlling gated switches
0 = Shorting switch is not active. Input cap branches shorted to op-amp input
1 = Shorting switch is enabled during internal PHI1. Input cap branches shorted to analog ground during internal PHI1 and to op-amp input during internal PHI2.

**Bit [4:0]**: **CCap [4:0]** Binary encoding for 32 possible capacitor sizes for C Capacitor:

| | |
|---|---|
| 0 0 0 0 0 = 0 Capacitor units in array | 1 0 0 0 0 = 16 Capacitor units in array |
| 0 0 0 0 1 = 1 Capacitor units in array | 1 0 0 0 1 = 17 Capacitor units in array |
| 0 0 0 1 0 = 2 Capacitor units in array | 1 0 0 1 0 = 18 Capacitor units in array |
| 0 0 0 1 1 = 3 Capacitor units in array | 1 0 0 1 1 = 19 Capacitor units in array |
| 0 0 1 0 0 = 4 Capacitor units in array | 1 0 1 0 0 = 20 Capacitor units in array |
| 0 0 1 0 1 = 5 Capacitor units in array | 1 0 1 0 1 = 21 Capacitor units in array |
| 0 0 1 1 0 = 6 Capacitor units in array | 1 0 1 1 0 = 22 Capacitor units in array |
| 0 0 1 1 1 = 7 Capacitor units in array | 1 0 1 1 1 = 23 Capacitor units in array |
| 0 1 0 0 0 = 8 Capacitor units in array | 1 1 0 0 0 = 24 Capacitor units in array |
| 0 1 0 0 1 = 9 Capacitor units in array | 1 1 0 0 1 = 25 Capacitor units in array |
| 0 1 0 1 0 = 10 Capacitor units in array | 1 1 0 1 0 = 26 Capacitor units in array |
| 0 1 0 1 1 = 11 Capacitor units in array | 1 1 0 1 1 = 27 Capacitor units in array |
| 0 1 1 0 0 = 12 Capacitor units in array | 1 1 1 0 0 = 28 Capacitor units in array |
| 0 1 1 0 1 = 13 Capacitor units in array | 1 1 1 0 1 = 29 Capacitor units in array |
| 0 1 1 1 0 = 14 Capacitor units in array | 1 1 1 1 0 = 30 Capacitor units in array |
| 0 1 1 1 1 = 15 Capacitor units in array | 1 1 1 1 1 = 31 Capacitor units in array |

Analog Switch Cap Type B Block 11 Control 2 Register (ASB11CR2, Address = Bank 0/1, 86h)
Analog Switch Cap Type B Block 13 Control 2 Register (ASB13CR2, Address = Bank 0/1, 8Eh)
Analog Switch Cap Type B Block 20 Control 2 Register (ASB20CR2, Address = Bank 0/1, 92h)
Analog Switch Cap Type B Block 22 Control 2 Register (ASB22CR2, Address = Bank 0/1, 9Ah)

## 10.15 Temperature Sensing Capability

A temperature-sensitive voltage derived from the Band Gap sensing on the die is buffered and available as an analog input into the Analog Switch Cap Type A Block ASA21. Temperature sensing allows protection of device operating ranges for fail-safe applications. Temperature sensing combined with a long sleep timer interval (to allow the die to approximate ambient temperature) can give an approximate ambient temperature for data acquisition and battery charging applications. The user may also calibrate the internal temperature rise based on a known current consumption.

The temperature sensor input to the ASA21 block is labeled VTemp, and its associated ground reference is labeled TRefGND (see Figure 22:, Figure 24:).

## 11.5 Supply Voltage Monitor

The Supply Voltage Monitor detector generates an interrupt whenever Vcc drops below a pre-programmed value. There are eight voltage trip points that are selectable by setting the VM [2:0] bit in the Voltage Monitor Control Register (VLT_CR). These bits also select the Switch Mode Pump trip points. The Supply Voltage Monitor will remain active when the device enters sleep mode.

**Table 96: Voltage Monitor Control Register**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/ Write | W | -- | -- | -- | -- | W | W | W |
| Bit Name | SMP | Reserved | Reserved | Reserved | Reserved | VM [2] | VM [1] | VM [0] |

**Bit 7**: **SMP** Disables SMP function
0 = Switch Mode Pump enabled, default
1 = Switch Mode Pump disabled

**Bit 6**: **Reserved**
**Bit 5**: **Reserved**
**Bit 4**: **Reserved**
**Bit 3**: **Reserved**

**Bit [2:0]**: **VM [2:0]**

| Low Voltage Detection | Switch Mode Pump |
|---|---|
| 0 0 0 = 2.95 Trip Voltage[1] | 0 0 0 = 3.17 Trip Voltage |
| 0 0 1 = 3.02 Trip Voltage | 0 0 1 = 3.25 Trip Voltage |
| 0 1 0 = 3.17 Trip Voltage | 0 1 0 = 3.42 Trip Voltage |
| 0 1 1 = 3.71 Trip Voltage | 0 1 1 = 3.94 Trip Voltage |
| 1 0 0 = 4.00 Trip Voltage | 1 0 0 = 4.19 Trip Voltage |
| 1 0 1 = 4.48 Trip Voltage | 1 0 1 = 4.64 Trip Voltage |
| 1 1 0 = 4.56 Trip Voltage | 1 1 0 = 4.82 Trip Voltage |
| 1 1 1 = 4.64 Trip Voltage | 1 1 1 = 5.00 Trip Voltage |

1. Voltages are ideal typical values. Tolerances are in Table 104 on page 129.

Voltage Monitor Control Register (VLT_CR, Address = Bank 1, E3h)

### 11.10.1 Data File Read

The user's data file should be read into the programmer. The checksum should be calculated by the programmer for each record and compared to the record checksum stored in the file for each record. If there is an error, a message should be sent to the user explaining that the file has a checksum error and the programming should not be allowed to continue.

### 11.10.2 Programmer Flow

The following sequence (with descriptions) is the main flow used to program the devices: (Note that failure at any step will result in termination of the flow and an error message to the device programmer's operator.)

#### 11.10.2.1 Verify Silicon ID

The silicon ID is read and verified against the expected value. If it is not the expected value, then the device is failed and an error message is sent to the device programmer's operator.

This test will detect a bad connection to the programmer or an incorrect device selection on the programmer.

The silicon ID test is required to be first in the flow and cannot be bypassed. The sequence is as follows:

```
Set Vcc=0V
Set SDATA=HighZ
Set SCLK=VILP
Set Vcc=Vccp
Start the programmer's SCLK driver
"free running"
WAIT-AND-POLL
ID-SETUP
WAIT-AND-POLL
READ-ID-WORD
```

**Notes**: See "DC Specifications" table in section 13 for value of Vccp and VILP. See "AC Specifications" table in section 13 for value of frequency for the SCLK driver (Fsclk).

#### 11.10.2.2 Erase

The Flash memory is erased. This is accomplished by the following sequence:

```
SET-CLK-FREQ(num_MHz_times_5)
Erase All
WAIT-AND-POLL
```

#### 11.10.2.3 Program

The Flash is programmed with the contents of the user's programming file. This is accomplished by the following sequence:

```
For num_block = 0 to max_data_block
For address =0 to 63
WRITE-BYTE(address,data):
End for address loop
SET-CLK-FREQ(num_MHz_times_5)
SET-BLOCK-NUM(num_block)
PROGRAM-BLOCK
WAIT-AND-POLL
End for num_block loop
```

#### 11.10.2.4 Verify (at Low Vcc and High Vcc)

The device data is read out to compare to the data in the user's programming file. This is accomplished by the following sequence:

```
For num_block = 0 to max_data_block
SET-BLOCK-NUM (num_block)
VERIFY-SETUP
Wait & POLL the SDATA for a high to
low transition
For address =0 to max_byte_per_block
READ-BYTE(address,data)
End for address loop
End for num_block loop
```

**Note**: This should be done 2 times; once at Vcc=Vcclv and once at Vcc=Vcchv.

#### 11.10.2.5 Set Security

The security operation protects certain blocks from being read or changed. This is done at the end of the flow so that the security does not interfere with the verify step. Security is set with the following sequence:

```
For address =0 to 63
WRITE-SECURITY-BYTE(address,data):
End for address loop
SET-CLK-FREQ(num_MHz_times_5)
SECURE
WAIT-AND-POLL
```

**Note**: This sequence is done at Vcc=Vccp.

## 12.0 Development Tools

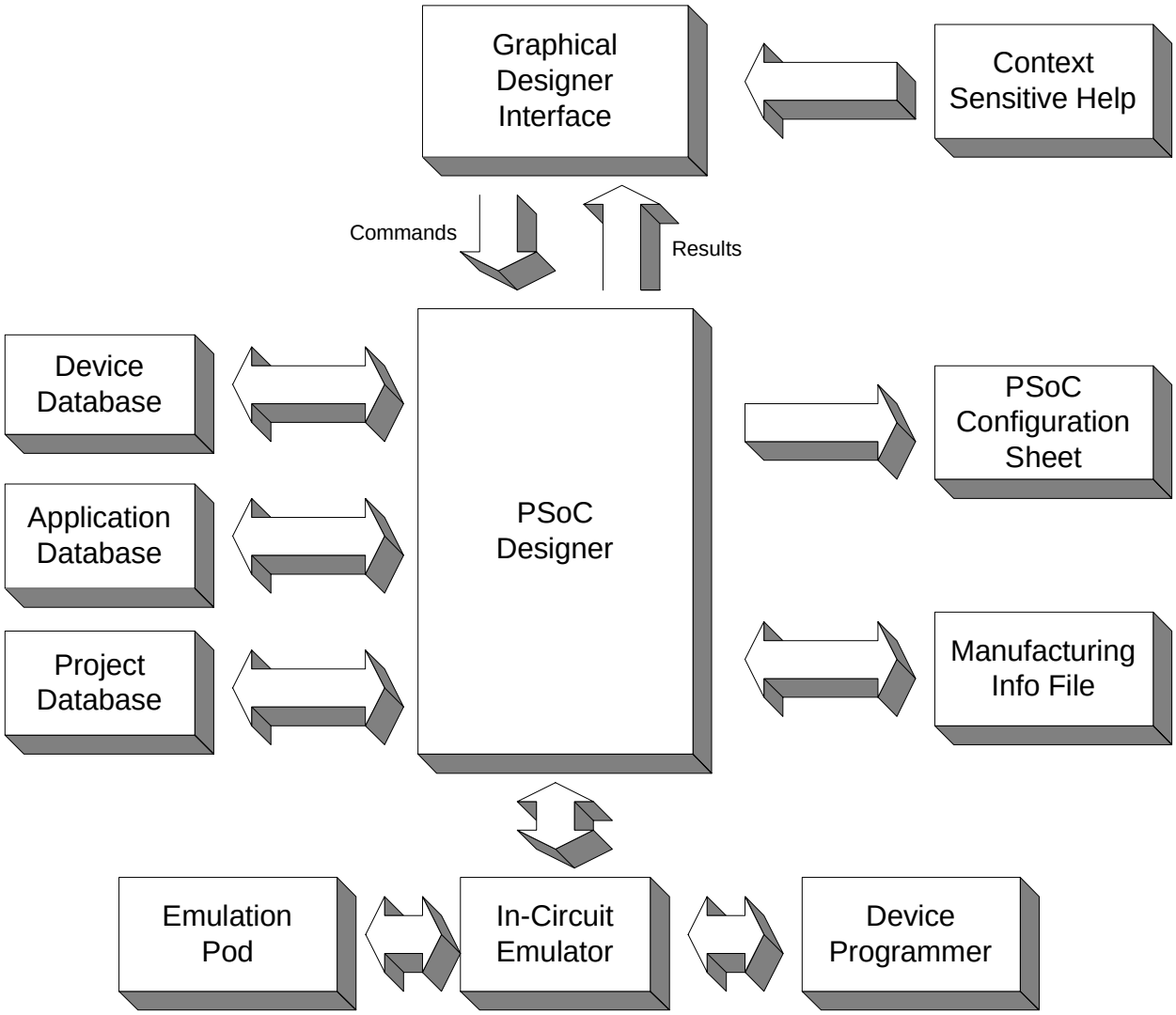


**Figure 35: PSoC Designer Functional Flow**

## 12.1 Overview

The Cypress MicroSystems PSoC Designer is a Microsoft® Windows-based, integrated development environment for the Programmable System-on-Chip (PSoC) devices. The PSoC Designer runs on Windows 98, Windows NT 4.0, Windows 2000, Windows Millennium (Me), or Windows XP.

PSoC Designer helps the customer to select an operating configuration for the microcontroller, write application code that uses the microcontroller, and debug the application. This system provides design database management by project, an integrated debugger with In-Circuit Emulator, in-system programming support, and the CYASM macro assembler for the CPUs.

PSoC Designer also supports a high-level C language compiler developed specifically for the devices in the family.

### 13.2.1.2 3.3V Specifications

The following table lists guaranteed maximum and minimum specifications for the voltage and temperature ranges, 3.3V +/- 10% and -40°C <= $T_A$ <= 85°C. The Operational Amplifier is a component of both the Analog Continuous Time PSoC blocks and the Analog Switch Cap PSoC blocks. The guaranteed specifications are measured in the Analog Continuous Time PSoC block. Typical parameters apply to 5V at 25°C and are for design guidance only. For 5V operation, see Table 105 on page 130.

**Table 106: 3.3V DC Operational Amplifier Specifications**

| Symbol | 3.3V DC Operational Amplifier Specifications | Minimum | Typical | Maximum | Unit |
|---|---|---|---|---|---|
| | Input Offset Voltage (Absolute Value) | - | 7 | 30 | mV |
| | Average Input Offset Voltage Drift | - | +24 | - | µV/°C |
| | Input Leakage Current[1] | - | 2 | 700 | nA |
| | Input Capacitance[2] | .32 | .36 | .42 | pF |
| | Common Mode Voltage Range[3] | .5 | - | $V_{cc}$ - 1.0 | VDC |
| | Common Mode Rejection Ratio | 80 | - | - | dB |
| | Open Loop Gain | 80 | - | - | dB |
| | High Output Voltage Swing (Worst Case Internal Load)<br>Bias = Low<br>Bias = Medium<br>Bias = High | <br>$V_{cc}$ - .4<br>$V_{cc}$ - .4<br>$V_{cc}$ - .4 | <br>-<br>-<br>- | <br>-<br>-<br>- | <br>V<br>V<br>V |
| | Low Output Voltage Swing (Worst Case Internal Load)<br>Bias = Low<br>Bias = Medium<br>Bias = High | <br>-<br>-<br>- | <br>-<br>-<br>- | <br>0.1<br>0.1<br>0.1 | <br>V<br>V<br>V |
| | Supply Current (Including Associated AGND Buffer)<br>Bias = Low<br>Bias = Medium<br>Bias = High | <br>-<br>-<br>- | <br>80<br>112<br>320 | <br>200<br>300<br>800 | <br>µA<br>µA<br>µA |
| | Supply Voltage Rejection Ratio | 60 | - | - | dB |

1. The leakage current includes the Analog Continuous Time PSoC block mux and the analog input mux. The leakage related to the General Purpose I/O pins is not included here.
2. The Input Capacitance includes the Analog Continuous Time PSoC block mux and the analog input mux. The capacitance of the General Purpose I/O pins is not included here.
3. The common-mode input voltage range is measured through an analog output buffer. The specification includes the limitations imposed by the characteristics of the analog output buffer

# 14.0 Packaging Information

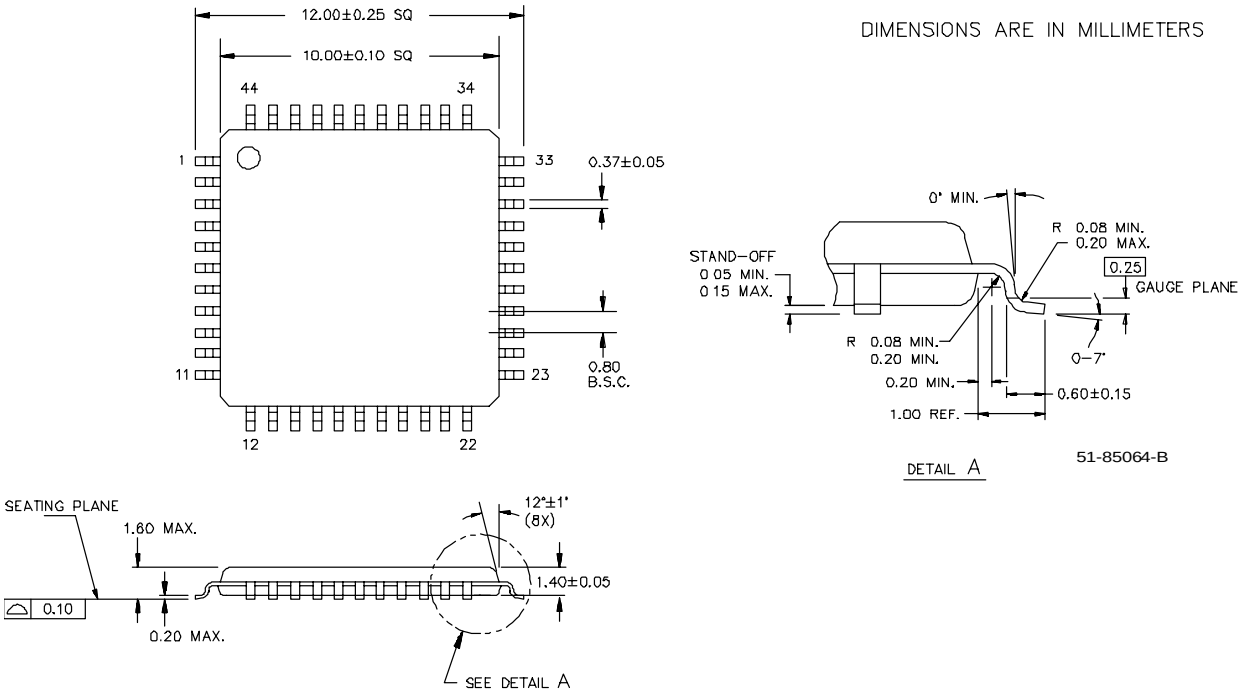


**Figure 37: 44-Lead Thin Plastic Quad Flat Pack A44**