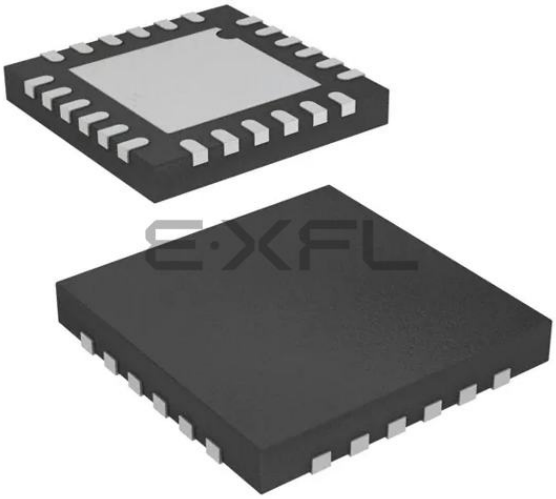


Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	Touchscreen Controller
Core Processor	M8C
Program Memory Type	FLASH (32kB)
Controller Series	CY8CT
RAM Size	2K x 8
Interface	I²C, SPI, UART/USART, USB
Number of I/O	20
Voltage - Supply	1.8V
Operating Temperature	-40°C ~ 85°C
Mounting Type	Surface Mount
Package / Case	24-UFQFN Exposed Pad
Supplier Device Package	24-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy8ctst200a-24lqxi

Core Register Summary

This table lists all the PSoC registers for the CPU core in ***address*** order within their system resource configuration. The grayed out bits are reserved bits. If you write these bits always write them with a value of '0'. For the core registers, the first 'x' in some ***register*** addresses represents either bank 0 or bank 1. These registers are listed throughout this manual in bank 0, even though they are also available in bank 1.

Summary Table of the Core Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
M8C REGISTER (page 27)										
x,F7h	CPU_F	PgMode[1:0]		XIO_1	XIO		Carry	Zero	GIE	RL : 02
RAM PAGING (SRAM) REGISTERS (page 39)										
x,6Ch	TMP_DR0	Data[7:0]								RW : 00
x,6Dh	TMP_DR1	Data[7:0]								RW : 00
x,6Eh	TMP_DR2	Data[7:0]								RW : 00
x,6Fh	TMP_DR3	Data[7:0]								RW : 00
0,D0h	CUR_PP						Page Bits[2:0]			RW : 0
0,D1h	STK_PP						Page Bits[2:0]			RW : 0
0,D3h	IDX_PP						Page Bits[2:0]			RW : 0
0,D4h	MVR_PP						Page Bits[2:0]			RW : 0
0,D5h	MVW_PP						Page Bits[2:0]			RW : 0
INTERRUPT CONTROLLER REGISTERS (page 45)										
0,DAh	INT_CLR0	I2C	Sleep	SPI	GPIO	Timer0	TrueTouch	Analog	V Monitor	RW : 00
0,DBh	INT_CLR1	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Rst	Timer2	Timer1	RW : 00
0,DCh	INT_CLR2			USB_WAKE	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4	RW : 00
0,DEh	INT_MSK2			USB Wakeup	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4	RW : 00
0,DFh	INT_MSK1	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Reset	Timer2	Timer1	RW : 00
0,E0h	INT_MSK0	I2C	Sleep	SPI	GPIO	Timer0	TrueTouch	Analog	V Monitor	RW : 00
0,E1h	INT_SW_EN								ENSWINT	RW : 0
0,E2h	INT_VC	Pending Interrupt[7:0]								RC : 00
GENERAL PURPOSE I/O (GPIO) REGISTERS (page 59)										
0,00h	PRT0DR	Data[7:0]								RW : 00
0,01h	PRT0IE	Interrupt Enables[7:0]								RW : 00
0,04h	PRT1DR	Data[7:0]								RW : 00
0,05h	PRT1IE	Interrupt Enables[7:0]								RW : 00
0,08h	PRT2DR	Data[7:0]								RW : 00
0,09h	PRT2IE	Interrupt Enables[7:0]								RW : 00
0,0Ch	PRT3DR	Data[7:0]								RW : 00
0,0Dh	PRT3IE	Interrupt Enables[7:0]								RW : 00
1,00h	PRT0DM0	Drive Mode 0[7:0]								RW : 00
1,01h	PRT0DM1	Drive Mode 1[7:0]								RW : FF
1,04h	PRT1DM0	Drive Mode 0[7:0]								RW : 00
1,05h	PRT1DM1	Drive Mode 1[7:0]								RW : FF
1,08h	PRT2DM0	Drive Mode 0[7:0]								RW : 00
1,09h	PRT2DM1	Drive Mode 1[7:0]								RW : FF
1,0Ch	PRT3DM0	Drive Mode 0[7:0]								RW : 00
1,0Dh	PRT3DM1	Drive Mode 1[7:0]								RW : FF
0,10h	PRTxDR	Data[7:0]								RW : 00
0,11h	PRTxIE	Interrupt Enables[7:0]								RW : 00
1,10h	PRTxDM0	Drive Mode 0[7:0]								RW : 00
1,11h	PRTxDM1	Drive Mode 0[7:0]								RW : 00

Table 2-2. Instruction Set Summary Sorted Alphabetically 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 if (A<B) C=1	3E	10	2	MVI A, [[expr]++]	Z	47	8	3	TST [expr], expr	Z
3A	7	2	CMP A, [expr]		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 1 Interrupt acknowledge to Interrupt Vector table = 13 cycles.

Note 2 The number of cycles required by an instruction is increased by one for instructions that span 128 byte page boundaries in the Flash memory space.

4. Program execution vectors to the interrupt table. Typically an LJMP instruction in the interrupt table sends execution to the user's interrupt service routine for this interrupt. (See Instruction Set Summary on page 28.)
5. The ISR executes. Interrupts are disabled since GIE = 0. In the ISR, interrupts can be re-enabled if necessaty by setting GIE = 1 (take care to avoid stack overflow in this case).
6. The ISR ends with an RETI instruction. This pops the Flag register, PCL, and PCH from the stack, restoring those registers. The restored Flag register re-enables interrupts since GIE = 1 again.
7. Execution resumes at the next instruction, after the instruction that occurred before the interrupt. However, if there are more pending interrupts, the subsequent interrupts are processed before the next normal program instruction.

Interrupt Latency. The time between the assertion of an enabled interrupt and the start of its ISR is calculated using this equation:

Latency =

Time for current instruction to finish +

Time for M8C to change program counter to interrupt address +

Time for LJMP instruction in interrupt table to execute.

Equation 1

$$\begin{aligned} &Latency = \\ &Time\ for\ current\ instruction\ to\ finish + \\ &Time\ for\ M8C\ to\ change\ program\ counter\ to\ interrupt\ address + \\ &Time\ for\ LJMP\ instruction\ in\ interrupt\ table\ to\ execute. \end{aligned}$$

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

Equation 2

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

In this example, at 24 MHz, 25 clock cycles take 1.042 µs.

Interrupt Priority. Interrupt priorities come into consideration when more than one interrupt is pending during the same instruction cycle. In this case, the Priority Encoder (see Figure 5-1) generates an interrupt vector for the highest priority pending interrupt.

5.1.1 Posted versus Pending Interrupts

An interrupt is posted when its interrupt conditions occur. This results in the flip-flop in Figure 5-1 clocking in a 1. The interrupt remains posted until the interrupt is taken or until it is cleared by writing to the appropriate INT_CLRx register.

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

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

It is especially important to understand the functionality of clearing posted interrupts, if the configuration of the PSoC device is changed by the application.

For example, if a block has a posted interrupt when it is enabled and then disabled, the posted interrupt remains. Therefore, it is good practice to use the INT_CLR register to clear posted interrupts before enabling or re-enabling a block.

5.2 Application Overview

The interrupt controller and its associated registers allow the user’s code to respond to an interrupt from almost every functional block in PSoC devices. Interrupts for all the digital blocks and each of the analog columns are available, as well as interrupts for supply voltage, sleep, variable clocks, and a general GPIO (pin) interrupt.

The registers associated with the interrupt controller allow for the disabling of interrupts either globally or individually. The registers also provide a mechanism by which a user can ***clear*** all pending and posted interrupts or clear individual posted or pending interrupts. A ***software*** mechanism is provided to set individual interrupts. Setting an interrupt by way of software is very useful during code development, when one may not have the complete hardware system necessary to generate a real interrupt.

5.3 Register Definitions

The following registers are associated with the Interrupt Controller and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of Interrupt Controller registers, refer to the Summary Table of the Core Registers on page 24.

5.3.1 INT_CLR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,DAh	INT_CLR0	I2C	Sleep	SPI	GPIO	Timer0	TrueTouch	Analog	V Monitor	RW : 00

The Interrupt Clear Register 0 (INT_CLR0) enables the individual interrupt sources' ability to clear posted interrupts.

The INT_CLR0 register is similar to the INT_MSK0 register in that it holds a bit for each interrupt source. Functionally the INT_CLR0 register is similar to the INT_VC register, although its operation is completely independent. When the INT_CLR0 register is read, any bits that are set indicate an interrupt was posted for that hardware resource. Reading this register gives the user the ability to determine all posted interrupts.

The Enable Software Interrupt (ENSWINT) bit in the INT_SW_EN register determines how an individual bit value, written to an INT_CLR0 register, is interpreted. When ENSWINT is cleared (the default state), writing 1's to the INT_CLR0 register has no effect. However, writing 0's to the INT_CLR0 register, when ENSWINT is cleared, causes the corresponding interrupt to clear. If the ENSWINT bit is set, any 0's written to the INT_CLR0 register are ignored. However, 1's written to the INT_CLR0 register, while ENSWINT is set, cause an interrupt to post for the corresponding interrupt.

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

Bit 7: I2C. This bit allows posted I2C interrupts to be read, cleared, or set.

Bit 6: Sleep. This bit allows posted sleep interrupts to be read, cleared, or set.

Bit 5: SPI. This bit allows posted SPI interrupts to be read, cleared, or set.

Bit 4: GPIO. This bit allows posted GPIO interrupts to be read, cleared, or set.

Bit 3: Timer0. This bit allows posted timer interrupts to be read, cleared, or set.

Bit 2: TrueTouch. This bit allows posted TrueTouch interrupts to be read, cleared, or set.

Bit 1: Analog. This bit allows posted analog interrupts to be read, cleared, or set.

Bit 0: V Monitor. This bit allows posted voltage monitor interrupts to be read, cleared, or set.

For additional information, refer to the INT_CLR0 register on page 242.

7.3 Register Definitions

The following registers are associated with the Internal Main Oscillator (IMO). The register descriptions have associated register tables showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table showing all oscillator registers, refer to the Summary Table of the Core Registers on page 24.

7.3.1 IMO_TR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E8h	IMO_TR	Trim[7:0]								RW : 00

The Internal Main Oscillator Trim Register (IMO_TR) manually centers the oscillator's output to a target frequency.

This register is loaded with a factory trim value at boot. When changing frequency ranges, the matching frequency trim value must be loaded into this register.

A TableRead command to the Supervisory ROM returns the trim values to the SRAM. EraseAll Parameters (05h), on page 36 has information on the location of various trim settings stored in Flash tables. Firmware needs to read the right trim value for desired frequency and update the IMO_TR register. The IMO_TR register must be changed at the lower frequency range setting.

For additional information, refer to the IMO_TR register on page 281

7.3.2 IMO_TR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,FAh	IMO_TR1						Fine Trim[2:0]			RW : 0

The Internal Main Oscillator Trim Register 1 (IMO_TR1) adjusts the IMO frequency .

Bits 2 to 0: Fine Trim[2:0]. These bits provide a fine tuning capability to the IMO trim. These three bits are the 3 LSB of the IMO trim with the IMO_TR register supplying the 8 MSB. A larger value in this register will increase the speed of the oscillator. The value in these bits varies the IMO frequency: approximately 7.5 kHz/step. When the EnableLock bit is set in the USB_CR1 register, firmware writes to this register are disabled.

For additional information, refer to the IMO_TR1 register on page 286.

TrueTouch Register Summary

The table below lists all the PSoC registers for the TrueTouch system in address order within their system resource configuration. The bits that are grayed out are reserved bits. If these bits are written, always write them with a value of '0'.

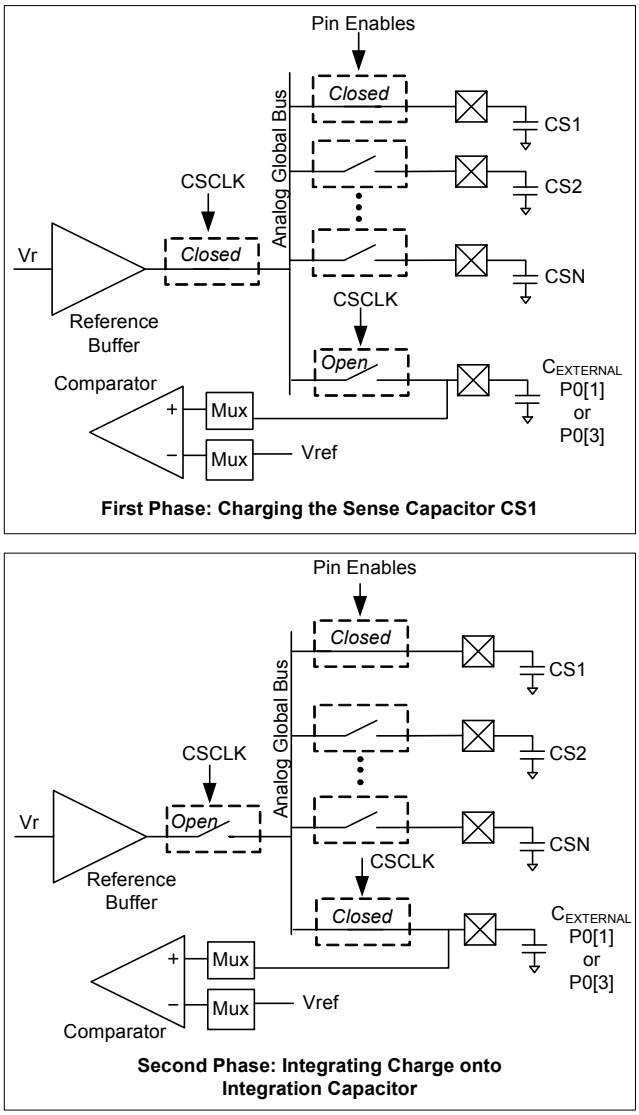
Summary Table of the TrueTouch Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
TRUETOUCH MODULE REGISTERS (page 92)										
0,A0h	CS_CR0	CSOUT[1:0]		CSD_ PRSCLK	CSD_CS_ CLK	CSD_ MODE	MODE[1:0]		EN	RW : 00
0,A1h	CS_CR1	CHAIN	CLKSEL[1:0]		RLOCLK	INV	INSEL[2:0]			RW : 00
0,A2h	CS_CR2	IRANGE		IDACDIR	IDAC_EN	CIN_EN	PXD_EN	CIP_EN	RO_EN	RW : 00
0,A3h	CS_CR3		REFMUX	REFMODE	REF_EN	LPFilt[1:0]		LPF_EN[1:0]		RW : 00
0,A4h	CS_CNTL	Data[7:0]								RO : 00
0,A5h	CS_CNTH	Data[7:0]								RO : 00
0,A6h	CS_STAT	INS	COLS	COHS	PPS	INM	COLM	COHM	PPM	# : 00
0,A7h	CS_TIMER		Timer Count Value[6:0]							RW : 00
0,A8h	CS_SLEW	FastSlew[6:0]							FS_EN	RW : 00
0,A9h	PRS_CR	CS_CLK_OUT	CS_CLK_INV	PRS_12BIT	PRS_EN	PRESCALE_ BYP	PRESCALE_CLK_DIV[2:0]			RW : 00
0,FDh	IDAC_D	DACDATA[7:0]								RW : 00
IO ANALOG MULTIPLEXER REGISTERS (page 100)										
0,61h	AMUX_CFG			PRX_ MODE		ICAPEN[1:0]		INTCAP[1:0]		RW : 00
1,D8h	MUX_CR0	ENABLE[7:0]								RW : 00
1,D9h	MUX_CR1	ENABLE[7:0]								RW : 00
1,DAh	MUX_CR2	ENABLE[7:0]								RW : 00
1,DBh	MUX_CR3	ENABLE[7:0]								RW : 00
1,DFh	MUX_CR4					ENABLE[3:0]				RW : 0
COMPARATOR REGISTERS (page 103)										
0,78h	CMP_RDC			CMP1D	CMP0D			CMP1L	CMP0L	# : 0
0,79h	CMP_MUX	INP1[1:0]		INN1[1:0]		INP0[1:0]		INN0[1:0]		RW : 00
0,7Ah	CMP_CR0				CMP1EN				CMP0EN	RW : 0
0,7Bh	CMP_CR1	CINT1	CPIN1	CRST1	CDS1	CINT0	CPIN0	CRST0	CDS0	RW : 00
0,7Ch	CMP_LUT	LUT1[3:0]				LUT0[3:0]				RW : 00

LEGEND
Access is bit specific. Refer to the Register Reference chapter on page 187 for additional information.
R Read register or bit(s).
W Write register or bit(s).

The circuit operates by alternately charging the sense capacitance to the internal voltage buffer level (first phase), then on the opposite phase of the clock (second phase), the analog global bus is connected to the integration capacitor while the voltage buffer is disconnected. (See Figure 11-3.) This builds up voltage on the integration capacitor, and eventually it trips the monitoring comparator. The comparator is configured to capture the number of counts of the internal oscillator during the charging interval, yielding a capacitance measurement of the sense pin.

Figure 11-3. Charging the TrueTouch and Connection



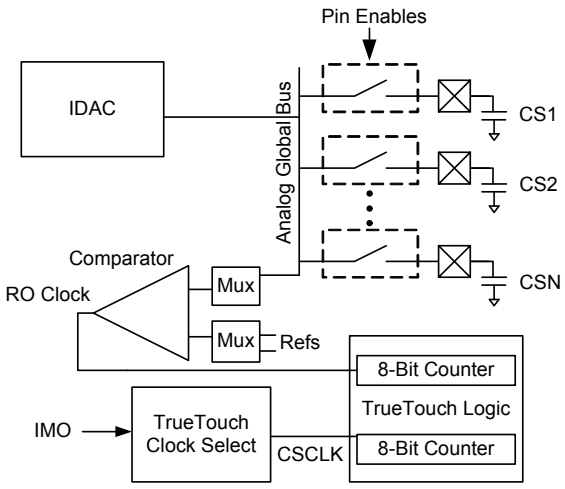
11.1.1.2 *Relaxation Oscillator*

The relaxation oscillator (RO) method operates by forming an oscillator using the sense capacitance. The IDAC, sense capacitance, and comparator (switching between two references) form the RO. There are two RO methods of capsensing supported.

In the first method, the RO is compared to the frequency of a similar internal oscillator. Normally, the relaxation oscillator is initially tuned (through firmware) to match the internal oscillator or to match a submultiple of the internal oscillator. The frequency difference is measured by defining an interval of internal oscillator clocks and counting the number of RO periods that occur during this interval. (See Figure 11-4.) Two 8-bit counters are used: one clocked by the internal oscillator and one clocked by the relaxation oscillator.

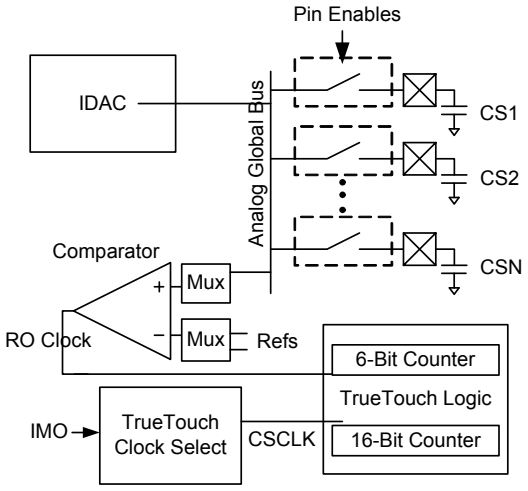
With the RO frequency closely matched to the internal oscillator and the interval set to 256 clocks, the 8-bit RO count result is the difference between the two oscillators. An overflow bit is available to test whether the number of capture counts is greater than or less than 256. Other count intervals are available. (See Figure 11-13.)

Figure 11-4. Relaxation Oscillator #1 Block Diagram



In the second RO method, the interval is set by a number of cycles of the RO using a 6-bit counter. During this interval, the IMO clocks a 16-bit counter and the final count gives a measure of capacitance. (See Figure 11-5.)

Figure 11-5. Relaxation Oscillator #2 Block Diagram

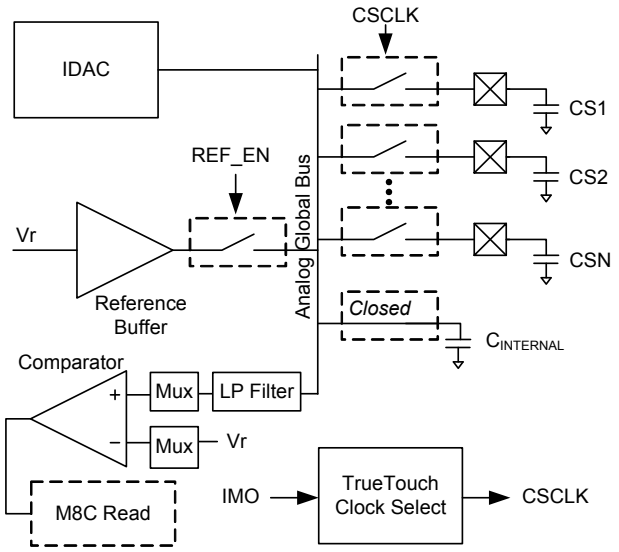


11.1.1.3 Successive Approximation

The successive approximation method provides a fast algorithm for capacitance measurement for applications such as detecting button presses. After a baseline is established, a set of capacitive sensors are very quickly scanned. High sensitivity can be achieved to enable scanning through a large dielectric. Figure 11-6 shows the hardware used in this method. The successive approximation method is used for proximity detection, fast button detection, and high resolution capacitance change measurement. The TrueTouch Successive Approximation (CSA) User Module in PSoC Designer™ uses this method.

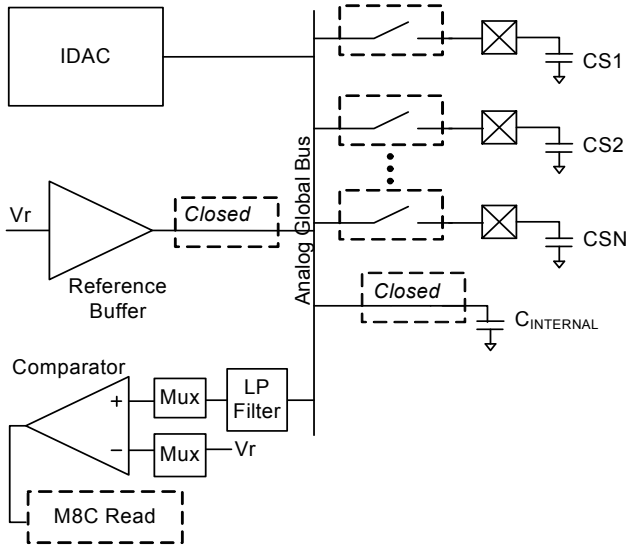
In this method, an internal capacitor is connected to the analog global bus. This bus contains ripple at the clock frequency, which is filtered with a low pass filter leading into the comparator. In addition, the IDAC current is set to the desired value as explained later in this section.

Figure 11-6. Successive Approximation Block Diagram



The reference bus is connected to the analog global line to initialize it to the Vr voltage (see Figure 11-7). After that, the reference buffer is disconnected and not used. The sense capacitance pin is now alternately grounded (see Figure 11-8) and connected to the analog global (see Figure 11-9). With the IDAC's current driving onto this net, the global net alternately charges and discharges a small amount.

Figure 11-7. Initialization Phase for Successive Approximation



11.2 Register Definitions

The following registers are associated with the TrueTouch Module and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of TrueTouch Module registers, refer to the TrueTouch Register Summary on page 84.

11.2.1 CS_CR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,A0h	CS_CR0	CSOUT[1:0]		CSD_ PRSCLK	CSD_CS_ CLK	CSD_ MODE	MODE[1:0]		EN	RW : 00

The TrueTouch Control Register 0 (CS_CR0) controls the operation of the TrueTouch counters. Do not write bits [7:1] while the block is enabled.

Bits 7 and 6: CSOUT[1:0]. These bits select between a number of TrueTouch signals that can be driven to an output pin. Refer to Figure 11-13 on page 91 and to the OUT_P1 register on page 273.

CSOUT[1:0]	Description
00	IN
01	CS_INT
10	COL
11	COH

Bit 5: CSD_PRSCLK. This bit selects between IMO-P or the PRS output as a clock source to drive the main capacitor switch. '0' selects IMO-P. '1' selects PRS output.

Bit 4: CSD_CS_CLK. This bit selects between IMO or IMO-P for the TrueTouch counters to work. Depending on this bit selection, either IMO or IMO-P is sent as the source clock to the clock dividers, which generate CS_CLK as shown in Figure 11-13 on page 91. '0' selects IMO. '1' selects IMO-P.

Bit 3: CSD_MODE. This bit enables the CSD mode. When this bit is enabled, the TIMER1 block works on the IMO-P (pre-scaled IMO) clock. This is also an enable for TrueTouch counters to toggle.

'0' disables CSD mode. Programmable Timer1 works on either CPUCLK/CLK32, (depending on CLKSEL bit selection in the PT1_CFG (0, B3h) register). '1' enables CSD mode.

When this bit is set to '1', the Programmable Timer1 works on IMO-P.

Note: Once the CSD_MODE bit is enabled, the IMO-P clock is a free running divider clock that cannot be stopped and restarted. The IMO-P and the CPU clock are both derived from the IMO clock but the phase relationship between them is nondeterministic.

Bits 2 and 1: MODE[1:0]. These bits specify the operating mode of the counter logic. The modes are shown in the following table.

MODE[1:0]	Description
00	**Stop On Event** In this mode, the block starts counting when the EN bit is set, and stops counting upon the selected interrupt event. This mode allows the user to read the counter results in firmware. Counting is restarted again by disabling and re-enabling the block using the EN bit.
01	**Pulse Width** In this mode, after the EN bit is set, the block waits for a positive edge upon the data input selection to start the counter, and then stops the counter upon the following negative edge of the data input. Polarity is adjusted with the INV bit (CS_CR1). Counting is restarted by disabling and re-enabling the block using the EN bit.
10	**Period** In this mode, after the EN bit is set, the block waits for a positive edge upon the data input selection to start the counter, and then stops the counter upon the following positive edge of the data input. Polarity is adjusted with the INV bit (CS_CR1). Counting is restarted by disabling and re-enabling the block using the EN bit.
11	**Continuous** In this mode, the counter is used to generate a periodic interrupt. The period is set by the input clock selection in conjunction with using one 8-bit counter (period=100h) or the chained 16-bit counter (period = 10000h).

Bit 0: EN. When this bit is written to '1', the counters are enabled for counting. When this bit is written to '0', counting is stopped and all counter values are reset to zero. If the counting mode is stopped in conjunction with an event (see MODE[1:0]), the current count is held and read from the counter registers. Toggle the EN bit to '0' and then back to '1' to start a new count.

For additional information, refer to the CS_CR0 register on page 211.

13.2.3 CMP_CR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,7Ah	CMP_CR0				CMP1EN				CMP0EN	RW : 00

The Comparator Control Register 0 (CMP_CR0) enables and configures the input range of the comparators.

Bit 4: CMP1EN. This bit enables comparator 1.

Bit 0: CMP0EN. This bit enables comparator 0.

For additional information, refer to the CMP_CR0 register on page 207.

13.2.4 CMP_CR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,7Bh	CMP_CR1	CINT1	CPIN1	CRST1	CDS1	CINT0	CPIN0	CRST0	CDS0	RW : 00

The Comparator Control Register 1 (CMP_CR1) configures the comparator output options.

Bit 7: CINT1. This bit connects the comparator 1 output to the analog output.

Bit 6: CPIN1. This bit selects whether the comparator 1 LUT output or the latched output is routed to a GPIO pin.

Bit 5: CRST1. This bit selects whether the comparator 1 latch is reset upon a register write or by a rising edge from the comparator 0 LUT output.

Bit 4: CDS1. This bit selects between the comparator 1 LUT and the latched output for the main comparator output that drives to the capacitive sense and interrupt logic.

Bit 3: CINT0. This bit connects the comparator 0 output to the analog output.

Bit 2: CPIN0. This bit selects whether the comparator 0 LUT output or the latched output is routed to a GPIO pin.

Bit 1: CRST0. This bit selects whether the comparator 0 latch is reset upon a register write or by a rising edge from the comparator 1 LUT output.

Bit 0: CDS0. This bit selects between the comparator 0 LUT and the latched output for the main comparator output that drives to the capacitive sense and interrupt logic.

For additional information, refer to the CMP_CR1 register on page 208.

13.2.5 CMP_LUT Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,7Ch	CMP_LUT	LUT1[3:0]				LUT0[3:0]				RW : 00

The Comparator LUT Control Register (CMP_LUT) selects the logic function.

Bits 7 to 4: LUT1[3:0]. These bits control the selection of the LUT 1 logic functions that may be selected for the comparator channel 1.

Bits 3 to 0: LUT0[3:0]. These bits control the selection of LUT 0 logic functions that may be selected for the comparator channel 0. The selections are shown in the following table:

LUTx[3:0]	0h: 0000: FALSE 1h: 0001: A .AND. $\overline{B}$ 2h: 0010: A .AND. $\overline{B}$ 3h: 0011: $\overline{A}$ 4h: 0100: $\overline{A}$ .AND. B 5h: 0101: B 6h: 0110: A .XOR. B 7h: 0111: A .OR. B 8h: 1000: A .NOR. B 9h: 1001: $\overline{A}$ .XNOR. B Ah: 1010: $\overline{B}$ Bh: 1011: $\overline{A}$ .OR. $\overline{B}$ Ch: 1100: $\overline{A}$ Dh: 1101: $\overline{A}$ .OR. B Eh: 1110: A. NAND. B Fh: 1111: TRUE

For additional information, refer to the CMP_LUT register on page 210.

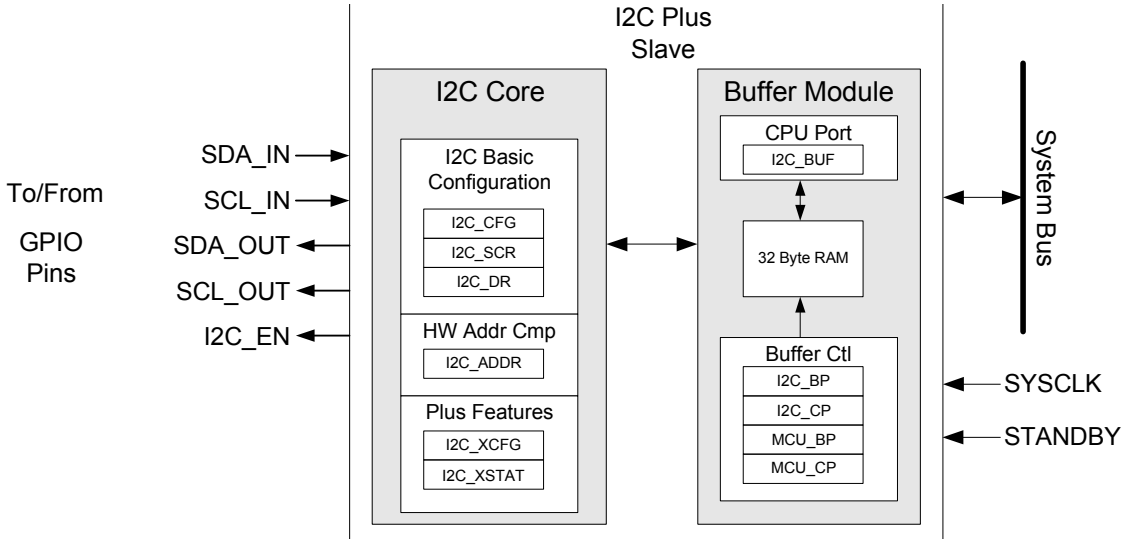
15. I²C Slave

This chapter explains the I²C Slave block and its associated registers. The I²C communications block is a serial processor designed to implement a complete I²C slave. For a complete table of the I²C registers, refer to the Summary Table of the System Resource Registers on page 106. For a quick reference of all PSoC registers in address order, refer to the Register Reference chapter on page 187.

15.1 Architectural Description

The I²C slave enhanced communications block is a serial-to-parallel processor, designed to interface the PSoC device to a two-wire I²C serial communications bus. To eliminate the need for excessive CPU intervention and overhead, the block provides I²C-specific support for status detection and generation of framing bits. By default, the I²C Slave Enhanced module is firmware compatible with the previous generation of I²C slave functionality. However, this module provides new features that are configurable to implement significant flexibility for both internal and external interfacing.

Figure 15-1. I²C Block Diagram



Basic I²C features include:

- Slave, transmitter, and receiver operation
- Byte processing for low CPU overhead
- Interrupt or polling CPU interface
- Support for clock rates of up to 400 kHz
- 7- or 10-bit addressing (through firmware support)
- SMBus operation (through firmware support)

Enhanced features of the I²C Slave Enhanced module include:

- Support for 7-bit hardware address compare
- Flexible data buffering schemes
- A "no bus stalling" operating mode
- A low power bus monitoring mode

18.2.2 SPI_RXR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,2Ah	SPI_RXR	Data[7:0]								R : 00

The SPI Receive Data Register (SPI_RXR) is the SPI's receive data register. A write to this register clears the RX Reg Full status bit in the Control register (SPI_CR).

Bits 7 to 0: Data[7:0]. These bits encompass the SPI Receive register. They are discussed by function type in Table 18-2 and Table 18-3.

For additional information, refer to the SPI_RXR register on page 191.

18.2.2.1 SPI Master Data Register Definitions

There are two 8-bit Data registers and one 8-bit Control/Status register. Table 18-2 explains the meaning of the Transmit and Receive registers in the context of SPIM operation.

Table 18-2. SPIM Data Register Descriptions

Name	Function	Description
SPI_TXR	TX Buffer	Write only register. If no transmission is in progress and this register is written to, the data from this register is loaded into the Shift register on the following clock edge, and a transmission is initiated. If a transmission is currently in progress, this register serves as a buffer for TX data. This register must only be written to when TX Reg Empty status is set and the write clears the TX Reg Empty status bit in the Control register. When the data is transferred from this register to the Shift register, then TX Reg Empty status is set.
SPI_RXR	RX Buffer	Read only register. When a byte transmission/reception is complete, the data in the shifter is transferred into the RX Buffer register and RX Reg Full status is set in the Control register. A read from this register clears the RX Reg Full status bit in the Control register.

18.2.2.2 SPI Slave Data Register Definitions

There are two 8-bit Data registers and one 8-bit Control/Status register. Table 18-3 explains the meaning of the Transmit and Receive registers in the context of SPIS operation.

Table 18-3. SPIS Data Register Descriptions

Name	Function	Description
SPI_TXR	TX Buffer	Write only register. This register must only be written to when TX Reg Empty status is set and the write clears the TX Reg Empty status bit in the Control register. When the data is transferred from this register to the Shift register, then TX Reg Empty status is set.
SPI_RXR	RX Buffer	Read only register. When a byte transmission/reception is complete, the data in the shifter is transferred into the RX Buffer register and RX Reg Full status is set in the Control register. A read from this register clears the RX Reg Full status bit in the Control register.

21.3.14 EPx_CNT0

Endpoint Count 0 Registers

Individual Register Names and Addresses:

EP1_CNT0 : 0,40h
EP2_CNT0 : 0,42h
EP3_CNT0 : 0,44h
EP4_CNT0 : 0,46h
EP5_CNT0 : 0,48h
EP6_CNT0 : 0,4Ah
EP7_CNT0 : 0,4Ch
EP8_CNT0 : 0,4Eh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RC : 0						# : 0
Bit Name	Data Toggle	Data Valid						Count MSB

These registers are endpoint count 0 registers.

In the table above, note that reserved bits are grayed table cells and are not described in the bit description section below. Reserved bits must always be written with a value of ‘0’. For additional information, refer to the Register Definitions on page 171 in the Full-Speed USB chapter.

Bit	Name	Description
7	**Data Toggle**	This bit selects the data packet's toggle state.
6	**Data Valid**	This bit is used for OUT transactions only and is read only.
0	**Count MSB**	This bit is the 1 MSb of a 9-bit counter.

21.3.19 CMP_MUX

Comparator Multiplexer Register

Individual Register Names and Addresses:

CMP_MUX : 0,79h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0		RW : 0		RW : 0		RW : 0	
Bit Name	INP1[1:0]		INN1[1:0]		INP0[1:0]		INN0[1:0]	

This register contains control bits for input selection of comparators 0 and 1.

For additional information, refer to the Register Definitions on page 103 in the Comparators chapter.

Bit	Name	Description
7:6	**INP1[1:0]**	Comparator 1 Positive Input Select 00b Analog Global Mux Bus 01b Reserved 10b P0[1] pin 11b P0[3] pin
5:4	**INN1[1:0]**	Comparator 1 Negative Input Select 00b VREF (1.0V) 01b Ref Lo (approximately 0.6V) 10b Ref Hi (approximately 1.2V) 11b Reserved
3:2	**INP0[1:0]**	Comparator 0 Positive Input Select 00b Analog Global Mux Bus 01b Reserved 10b P0[1] pin 11b P0[3] pin
1:0	**INN0[1:0]**	Comparator 0 Negative Input Select 00b VREF (1.0V) 01b Ref Lo (approximately 0.6V) 10b Ref Hi (approximately 1.2V) 11b Reserved

21.3.64 IDAC_D

Current DAC Data Register

Individual Register Names and Addresses:

IDAC_D : 0,FDh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	IDACDATA[7:0]							

This register specifies the 8-bit multiplying factor that determines the output IDAC current.

For additional information, refer to the Register Definitions on page 92 in the TrueTouch Module chapter.

Bits	Name	Description
7:0	**IDACDATA[7:0]**	This is an 8-bit value that selects the number of current units that combine to form the IDAC current. This current then drives the analog mux bus when IDAC mode is enabled. For example, a setting of 80h means that the charging current is 128 current units. The current size also depends on the IRANGE setting in the CS_CR2 register. This setting supplies the charging current for the relaxation oscillator. This current and the external capacitance connected to the analog global bus determines the RO frequency. This register is also used to set the charging current in the proximity detect mode. Step size is approximately 330 nA/bit for default IRANGE state 00b. 00h Smallest current. . . . FFh Largest current.

21.4.16 IO_CFG2

Input/Output Configuration Register 2

Individual Register Names and Addresses:

IO_CFG2 : 1,DEh

	7	6	5	4	3	2	1	0
Access : POR			RW : 0				RW : 0	
Bit Name			REG_LEVEL[2:0]				REG_CLOCK[1:0]	

The Input/Output Configuration 2 Register (IO_CFG2) selects output regulated supply and clock rates.

In the table above, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits should always be written with a value of ‘0’. For additional information, refer to the IO_CFG2 Register on page 61 in the GPIO chapter.

Bits	Name	Description
5:3	REG_LEVEL[2:0]	These bits select output regulated supply

REG_LEVEL[2:0]	Approx. Regulated Supply (V)		
000	3	2.5	1.8
001	3.1	2.6	1.9
010	3.2	2.7	2.0
011	3.3	2.8	2.1
100	3.4	2.9	2.2
101	3.5	3.0	2.3
110	3.6	3.1	2.4
111	3.7	3.2	2.5

Bits	Name	Description
1:0	REG_CLOCK[1:0]	The Regulated I/O charge pump can operate with a maximum clock speed of 12 MHZ. The REG_CLOCK[1:0] bits select clocking options for the regulator. Setting REG_CLOCK[1:0] to ‘10’ should be used with 24 MHz SYSCLK and ‘01’ should be used with 6/12 MHz SYSCLK.

REG_CLOCK[1:0]	SYSCLK Clock Rate
10	24 MHz
01	6/12 MHz

21.4.21 VLT_CMP

Voltage Monitor Comparators Register

Individual Register Names and Addresses:

VLT_CMP : 1,E4h

	7	6	5	4	3	2	1	0
Access : POR							R : 0	
Bit Name							LVD	

This register reads the state of the internal supply voltage monitors.

In the table above, note that reserved bits are grayed table cells and are not described in the bit description section below. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 144 in the POR chapter.

Bit	Name	Description
1	LVD	This bit reads the state of the LVD comparator. 0 Vdd is above LVD trip point. 1 Vdd is below LVD trip point.

phase	The relationship between two signals, usually the same frequency, that determines the delay between them. This delay between signals is either measured by time or angle (degrees).
Phase-Locked Loop (PLL)	An electronic circuit that controls an ***oscillator*** so that it maintains a constant phase angle relative to a reference signal.
pin	A terminal on a hardware component. Also called lead.
pinouts	The pin number assignment: the relation between the logical inputs and outputs of the PSoC device and their physical counterparts in the printed circuit board (PCB) package. Pinouts involves pin numbers as a link between schematic and PCB design (both being computer generated files) and may also involve pin names.
port	A group of pins, usually eight.
positive edge	A transition from a logic 0 to a logic 1. Also known as a rising edge.
posted interrupts	An interrupt that has been detected by the hardware but may or may not be enabled by its mask bit. Posted interrupts that are not masked become pending interrupts.
Power On Reset (POR)	A circuit that forces the PSoC device to reset when the voltage is below a pre-set level. This is one type of ***hardware reset***.
program counter	The instruction pointer (also called the program counter) is a register in a computer processor that indicates where in memory the CPU is executing instructions. Depending on the details of the particular machine, it holds either the address of the instruction being executed or the address of the next instruction to be executed.
protocol	A set of rules. Particularly the rules that govern networked communications.
PSoC®	Cypress Semiconductor's Programmable System-on-Chip (PSoC) mixed-signal array. PSoC® is a trademark of Cypress.
PSoC blocks	*See* ***analog blocks*** and ***digital blocks***.
PSoC Designer™	The software for Cypress' Programmable System-on-Chip technology.
pulse	A rapid change in some characteristic of a signal (for example, phase or frequency) from a baseline value to a higher or lower value, followed by a rapid return to the baseline value.
pulse width modulator (PWM)	An output in the form of duty cycle which varies as a function of the applied measure.

R

RAM	An acronym for random access memory. A data-storage device from which data can be read out and new data can be written in.
register	A storage device with a specific capacity, such as a bit or byte.
reset	A means of bringing a system back to a known state. See ***hardware reset*** and ***software reset***.
resistance	The resistance to the flow of electric current measured in ohms for a conductor.

T

U

V