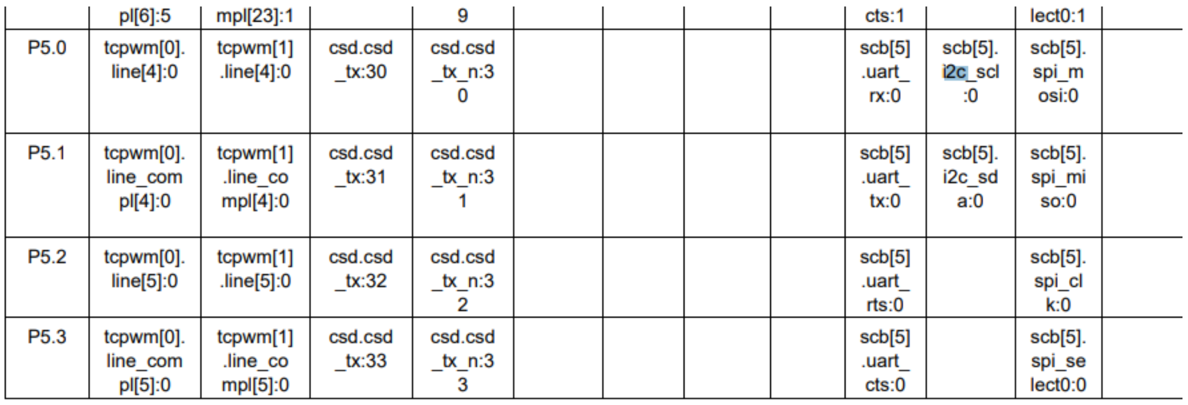
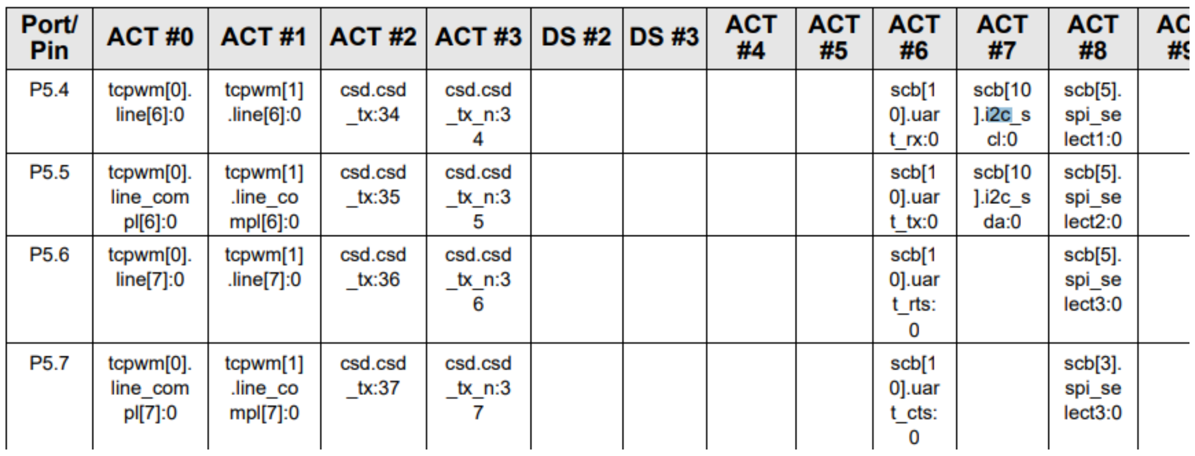
Serial Communication Blocks (SCB)
This product line has 13 SCBs:
■ Eight can implement either I2C, UART, or SPI.
■ Four can implement either I2C or UART.
■ One SCB (SCB #8) can operate in system Deep Sleep mode with an external clock; this SCB can be either SPI slave or I2C slave.
I2C Mode: The SCB can implement a full multi-master and slave interface (it is capable of multimaster arbitration). This block can operate at speeds of up to 1 Mbps (Fast Mode Plus). It also supports EZI2C, which creates a mailbox address range and effectively reduces I2C communication to reading from and writing to an array in memory. The SCB supports a 256-byte FIFO for receive and transmit.
The I2C peripheral is compatible with I2C standard-mode, Fast Mode, and Fast Mode Plus devices as defined in the NXP I 2C-bus specification and user manual (UM10204). The I2C bus I/O is implemented with GPIO in open-drain modes.
The port 1 pins are capable of overvoltage-tolerant (OVT) operation, where the input voltage may be higher than VDDD. OVT pins are commonly used with I2C, to allow powering the chip OFF while maintaining a physical connection to an operating I2C bus without affecting its functionality.



	pl[6]:5	mpl[23]:1		9					cts:1		lect0:1	
P5.0	tcpwm[0].line[4]:0	tcpwm[1].line[4]:0	csd.csd_tx:30	csd.csd_tx_n:30					scb[5].uart_rx:0	scb[5].i2c_scl:0	scb[5].spi_mosi:0	
P5.1	tcpwm[0].line_compl[4]:0	tcpwm[1].line_compl[4]:0	csd.csd_tx:31	csd.csd_tx_n:31					scb[5].uart_tx:0	scb[5].i2c_sda:0	scb[5].spi_miso:0	
P5.2	tcpwm[0].line[5]:0	tcpwm[1].line[5]:0	csd.csd_tx:32	csd.csd_tx_n:32					scb[5].uart_rts:0		scb[5].spi_clk:0	
P5.3	tcpwm[0].line_compl[5]:0	tcpwm[1].line_compl[5]:0	csd.csd_tx:33	csd.csd_tx_n:33					scb[5].uart_cts:0		scb[5].spi_select0:0	

Table 8. Multiple Alternate Functions *(continued)*



Port/ Pin	ACT #0	ACT #1	ACT #2	ACT #3	DS #2	DS #3	ACT #4	ACT #5	ACT #6	ACT #7	ACT #8	AC #9
P5.4	tcpwm[0].line[6]:0	tcpwm[1].line[6]:0	csd.csd_tx:34	csd.csd_tx_n:34					scb[10].uart_rx:0	scb[10].i2c_scl:0	scb[5].spi_select1:0	
P5.5	tcpwm[0].line_compl[6]:0	tcpwm[1].line_compl[6]:0	csd.csd_tx:35	csd.csd_tx_n:35					scb[10].uart_tx:0	scb[10].i2c_sda:0	scb[5].spi_select2:0	
P5.6	tcpwm[0].line[7]:0	tcpwm[1].line[7]:0	csd.csd_tx:36	csd.csd_tx_n:36					scb[10].uart_rts:0		scb[5].spi_select3:0	
P5.7	tcpwm[0].line_compl[7]:0	tcpwm[1].line_compl[7]:0	csd.csd_tx:37	csd.csd_tx_n:37					scb[10].uart_cts:0		scb[3].spi_select3:0	

									0			
P6.0	tcpwm[0].line[0]:1	tcpwm[1].line[8]:0	csd.csd_tx:38	csd.csd_tx_n:38	scb[8].i2c_scl:0				scb[3].uart_rx:0	scb[3].i2c_scl:0	scb[3].spi_mosi:0	
P6.1	tcpwm[0].line_compl[0]:1	tcpwm[1].line_compl[8]:0	csd.csd_tx:39	csd.csd_tx_n:39	scb[8].i2c_sda:0				scb[3].uart_tx:0	scb[3].i2c_sda:0	scb[3].spi_miso:0	
P6.2	tcpwm[0].line[1]:1	tcpwm[1].line[9]:0	csd.csd_tx:40	csd.csd_tx_n:4					scb[3].uart_		scb[3].spi_cl	

P6.4	tcpwm[0].line[2]:1	tcpwm[1].line[10]:0	csd.csd_tx:42	csd.csd_tx_n:42	scb[8].i2c_scl:1				scb[6].uart_rx:2	scb[6].i2c_scl:2	scb[6].spi_mosi:2	
P6.5	tcpwm[0].line_compl[2]:1	tcpwm[1].line_compl[10]:0	csd.csd_tx:43	csd.csd_tx_n:43	scb[8].i2c_sda:1				scb[6].uart_tx:2	scb[6].i2c_sda:2	scb[6].spi_miso:2	

Table 8. Multiple Alternate Functions *(continued)*

Port/Pin	ACT #0	ACT #1	ACT #2	ACT #3	DS #2	DS #3	ACT #4	ACT #5	ACT #6	ACT #7	ACT #8	AC #
P7.0	tcpwm[0].line[4]:1	tcpwm[1].line[12]:0	csd.csd_tx:46	csd.csd_tx_n:46					scb[4].uart_rx:1	scb[4].i2c_scl:1	scb[4].spi_mosi:1	
P7.1	tcpwm[0].line_compl[4]:1	tcpwm[1].line_compl[12]:0	csd.csd_tx:47	csd.csd_tx_n:47					scb[4].uart_tx:1	scb[4].i2c_sda:1	scb[4].spi_miso:1	

P8.0	tcpwm[0].line[0]:2	tcpwm[1].line[16]:0	csd.csd_tx:54	csd.csd_tx_n:54					scb[4].uart_rx:0	scb[4].i2c_scl:0	scb[4].spi_mosi:0	
P8.1	tcpwm[0].line_compl[0]:2	tcpwm[1].line_compl[16]:0	csd.csd_tx:55	csd.csd_tx_n:55					scb[4].uart_tx:0	scb[4].i2c_sda:0	scb[4].spi_miso:0	
P8.2	tcpwm[0].	tcpwm[1]	csd.csd	csd.csd		lpcom			scb[4]		scb[4].	

Table 8. Multiple Alternate Functions *(continued)*

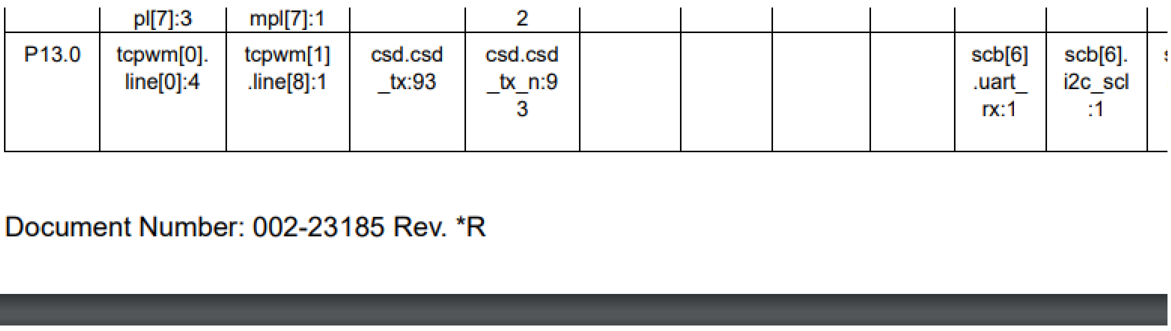
Port/Pin	ACT #0	ACT #1	ACT #2	ACT #3	DS #2	DS #3	ACT #4	ACT #5	ACT #6	ACT #7	ACT #8	ACT #9
P8.4	tcpwm[0].line[2]:2	tcpwm[1].line[18]:0	csd.csd_tx:58	csd.csd_tx_n:58					scb[11].uart_rx:0	scb[11].i2c_scl:0	scb[4].spi_select1:0	
P8.5	tcpwm[0].line_compl[2]:2	tcpwm[1].line_compl[18]:0	csd.csd_tx:59	csd.csd_tx_n:59					scb[11].uart_tx:0	scb[11].i2c_sda:0	scb[4].spi_select2:0	

									0			
P9.0	tcpwm[0].line[4]:2	tcpwm[1].line[20]:0	csd.csd_tx:62	csd.csd_tx_n:62					scb[2].uart_rx:0	scb[2].i2c_scl:0	scb[2].spi_mosi:0	
P9.1	tcpwm[0].line_compl[4]:2	tcpwm[1].line_compl[20]:0	csd.csd_tx:63	csd.csd_tx_n:63					scb[2].uart_tx:0	scb[2].i2c_sda:0	scb[2].spi_miso:0	

	line_compl[0]:6	.line_compl[1]:2	_tx:69	_tx_n:69								
P10.0	tcpwm[0].line[6]:2	tcpwm[1].line[22]:0	csd.csd_tx:70	csd.csd_tx_n:70					scb[1].uart_rx:1	scb[1].i2c_scl:1	scb[1].spi_mosi:1	

Table 8. Multiple Alternate Functions *(continued)*

Port/ Pin	ACT #0	ACT #1	ACT #2	ACT #3	DS #2	DS #3	ACT #4	ACT #5	ACT #6	ACT #7	ACT #8	A
P10.1	tcpwm[0].line_compl[6]:2	tcpwm[1].line_compl[22]:0	csd.csd_tx:71	csd.csd_tx_n:71					scb[1].uart_tx:1	scb[1].i2c_sda:1	scb[1].spi_miso:1	
	pl[1]:0	mpl[2]:2										
P11.0	tcpwm[0].line[1]:3	tcpwm[1].line[1]:1	csd.csd_tx:78	csd.csd_tx_n:78				smif.spi_select2	scb[5].uart_rx:1	scb[5].i2c_scl:1	scb[5].spi_mosi:1	
P11.1	tcpwm[0].line_compl[1]:3	tcpwm[1].line_compl[1]:1	csd.csd_tx:79	csd.csd_tx_n:79				smif.spi_select1	scb[5].uart_tx:1	scb[5].i2c_sda:1	scb[5].spi_miso:1	
P12.0	tcpwm[0].line[4]:3	tcpwm[1].line[4]:1	csd.csd_tx:85	csd.csd_tx_n:85				smif.spi_data4	scb[6].uart_rx:0	scb[6].i2c_scl:0	scb[6].spi_mosi:0	
P12.1	tcpwm[0].line_compl[4]:3	tcpwm[1].line_compl[4]:1	csd.csd_tx:86	csd.csd_tx_n:86				smif.spi_data5	scb[6].uart_tx:0	scb[6].i2c_sda:0	scb[6].spi_miso:0	



	pl[7]:3	mpl[7]:1		2							
P13.0	tcpwm[0].line[0]:4	tcpwm[1].line[8]:1	csd.csd_tx:93	csd.csd_tx_n:93					scb[6].uart_rx:1	scb[6].i2c_scl:1	

Document Number: 002-23185 Rev. *R

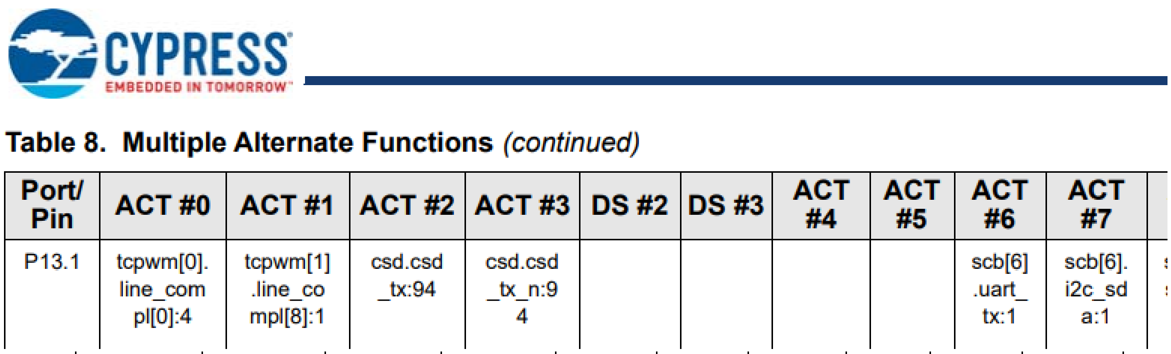
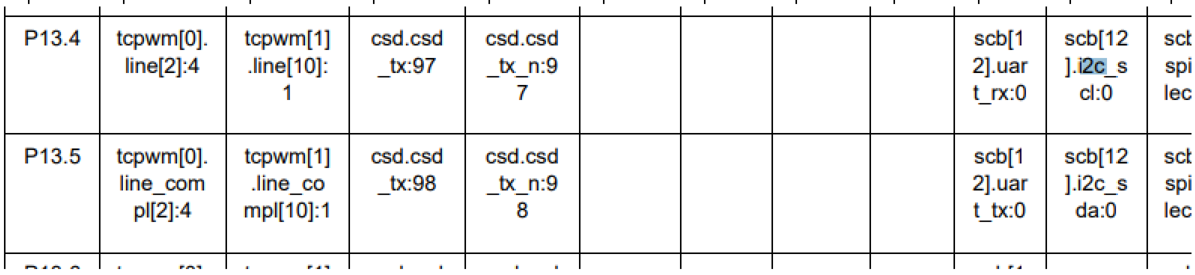


Table 8. Multiple Alternate Functions *(continued)*

Port/ Pin	ACT #0	ACT #1	ACT #2	ACT #3	DS #2	DS #3	ACT #4	ACT #5	ACT #6	ACT #7	
P13.1	tcpwm[0].line_compl[0]:4	tcpwm[1].line_compl[8]:1	csd.csd_tx:94	csd.csd_tx_n:94					scb[6].uart_tx:1	scb[6].i2c_sda:1	



P13.4	tcpwm[0].line[2]:4	tcpwm[1].line[10]:1	csd.csd_tx:97	csd.csd_tx_n:97					scb[12].uart_rx:0	scb[12].i2c_scl:0	sc spi lec
P13.5	tcpwm[0].line_compl[2]:4	tcpwm[1].line_compl[10]:1	csd.csd_tx:98	csd.csd_tx_n:98					scb[12].uart_tx:0	scb[12].i2c_sda:0	sc spi lec

Architectural TRM: -

28. Serial Communications Block (SCB)

The Serial Communications Block (SCB) supports three serial communication protocols: Serial Peripheral Interface (SPI), Universal Asynchronous Receiver Transmitter (UART), and Inter Integrated Circuit (I2C or IIC). Only one of the protocols is supported by an SCB at any given time. The number of SCBs in a PSoC 6 MCU varies by part number; consult the PSoC 61 datasheet/PSoC 62 datasheet to determine number of SCBs and the SCB pin locations. Not all SCBs support all three modes (SPI, UART, and I2C); consult the PSoC 61 datasheet/PSoC 62 datasheet to determine which modes are supported by which SCBs. Not all SCBs operate in deep sleep, consult the PSoC 61 datasheet/PSoC 62 datasheet to determine which SCBs operate in deep sleep.

28.1 Features

The SCB supports the following features:

- ■ Standard SPI master and slave functionality with Motorola, Texas Instruments, and National Semiconductor protocols
- ■ Standard UART functionality with SmartCard reader, Local Interconnect Network (LIN), and IrDA protocols
    - ❐ Standard LIN slave functionality with LIN v1.3 and LIN v2.1/2.2 specification compliance
- ■ Standard I2C master and slave functionality

■ Trigger outputs for connection to DMA
■ Multiple interrupt sources to indicate status of FIFOs and transfers
■ Features available only on Deep Sleep-capable SCB:

❒ EZ mode for SPI and I2C slaves; allows for operation without CPU intervention
❒ CMD_RESP mode for SPI and I2C slaves; allows for operation without CPU intervention
❒ Low-power (Deep Sleep) mode of operation for SPI and I2C slaves (using external clocking)
❒ Deep Sleep wakeup on I2C slave address match or SPI slave selection

28.2.1.1 FIFO Mode
In this mode the RAM is split into two 128-byte FIFOs, one for transmit (TX) and one for receive (RX). The FIFOs can be configured to be 8 bits x 128 elements or 16 bits x 64 elements; this is done by setting the BYTE_MODE bit in the SCB control register.
FIFO mode of operation is available only in Active and Sleep power modes. However, the I2C address or SPI slave select can be used to wake the device from Deep Sleep on the Deep Sleep-capable SCB. Statuses are provided for both the RX and TX FIFOs. There are multiple interrupt sources available, which indicate the status of the FIFOs, such as full or empty; see “SCB Interrupts” on page 368.
28.2.1.2 EZ Mode
In easy (EZ) mode the RAM is used as a single 256-byte buffer. The external master sets a base address and reads and writes start from that base address.
EZ Mode is available only for SPI slave and I2C slave. It is available only on the Deep Sleep capable SCB.
EZ mode is available in Active, Sleep, and Deep Sleep power modes.
Note: This document discusses hardware implementation of the EZ mode; for the firmware implementation, see the PDL.
28.2.1.3 CMD_RESP Mode
Command Response (CMD_RESP) mode is similar to EZ mode except that the base address is provided by the CPU not the external master.
MD_RESP mode is available only for SPI slave and I2C slave. It is available only on the Deep Sleep-capable SCB.
CMD_RESP mode operation is available in Active, Sleep, and Deep Sleep power modes.
28.2.2 Clocking Modes
The SCB can be clocked either by an internal clock provided by the peripheral clock dividers (referred to as clk_scb in this document), or it can be clocked by the external master.

■ UART, SPI master, and I2C master modes must use clk_scb.
■ Only SPI slave and I2C slave can use the clock from an external master, and only the Deep Sleep capable SCB supports this.

Internally- and externally-clocked slave functionality is determined by two register fields of the SCB CTRL register:
■ EC_AM_MODE indicates whether SPI slave selection or I 2C address matching is internally (‘0’) or externally (‘1’) clocked.
■ EC_OP_MODE indicates whether the rest of the protocol operation (besides SPI slave selection and I2C address matching) is internally (‘0’) or externally (‘1’) clocked.

Notes:
■ FIFO mode supports an internally- or externally-clocked address match (EC_AM_MODE is ‘0’ or

‘1’); however, data transfer must be done with internal clocking. (EC_OP_MODE is ‘1’).

■ EZ and CMD_RESP modes are supported with externally clocked operation (EC_OP_MODE is ‘1’).

Table 28-1 provides an overview of the clocking and buffer modes supported for each communication mode.

Table 28-1. Clock Mode Compatibility

	Internally clocked (IC)			**Externally clocked (EC) (Deep Sleep SCB only)**		
	FIFO	**EZ**	**CMD_RESP**	**FIFO**	**EZ**	**CMD_RESP**
I^2C master	Yes	No	No	No	No	No
I^2C slave	Yes	Yes	No	Yes[a]	Yes	Yes
I^2C master-slave	Yes	No	No	No	No	No
SPI master	Yes	No	No	No	No	No
SPI slave	Yes	Yes	No	Yes[b]	Yes	Yes
UART transmitter	Yes	No	No	No	No	No
UART receiver	Yes	No	No	No	No	No

a. In Deep Sleep mode the external-clocked logic can handle slave address matching, it then triggers an interrupt to wake up the CPU. The slave can be programmed to stretch the clock, or NACK until internal logic takes over. This applies only to the Deep Sleep-capable SCB.

b. In Deep Sleep mode the external-clocked logic can handle slave selection detection, it then triggers an interrupt to wake up the CPU. Writes will be ignored and reads will return 0xFF until internal logic takes over. This applies only to the Deep Sleep-capable SCB.

Table 28-2. Clock Configuration and Mode support

Mode	**EC_AM_MODE is '0'; EC_OP_MODE is ‘0’**	**'EC_AM_MODE is '1'; EC_OP_MODE is '0'**	**'EC_AM_MODE is '1'; EC_OP_MODE is '1'**
FIFO mode	Yes	Yes	No
EZ mode	Yes	Yes	Yes
CMD_RESP mode	No	No	Yes