GPIO Usage Setup in Traveo II Family

About this document

Scope and purpose

AN220193 explains how to use GPIO pins effectively in Traveo™ II family MCUs. This application note also explains GPIO basics, configuration options, interrupts, and low-power behavior.

Associated Part Family

Traveo™ II Family CYT2/CYT3/CYT4 Series

Table of contents

1 Introduction

Traveo II family MCUs have a flexible general-purpose I/O (GPIO) architecture that provides additional features than traditional MCUs. Traveo II GPIOs are controlled not only by configuring the registers in firmware, similar to traditional MCUs, but are also driven by custom digital logic and analog block signals. This application note explains the basics of Traveo II GPIO pins and shows techniques for using them effectively for different functions. To understand more details about the functionality and terminology used in this application note, see the “I/O System” chapter of the **Traveo II Architecture Technical Reference Manual (TRM)**.

2 GPIO Pin Basics

Traveo II GPIO pins offer the following features:

- Analog and digital input and output capabilities
- Various drive modes
- Separate port read and write registers
- Slew rate control
- High-speed I/O matrix (HSIOM)
- Edge-triggered interrupts on rising edge, falling edge, or on both the edges, on all GPIO
- Hold mode for latching previous state (used to retain the I/O state in DeepSleep mode)
- Selectable CMOS, TTL, and automotive input buffer mode

The GPIO functionality depends on the peripherals available in Traveo II family MCUs.

HSIOM is a set of high-speed multiplexers that route internal CPU and peripheral signals to and from GPIOs. HSIOM allows GPIOs to be shared with multiple functions and multiplexes the pin connection to a peripheral that you select. For details on HSIOM connection, see the **Architecture TRM**.

2.1 Physical Structure of GPIO Pins

Figure 1 shows the pin connections with the resources in Traveo II device.

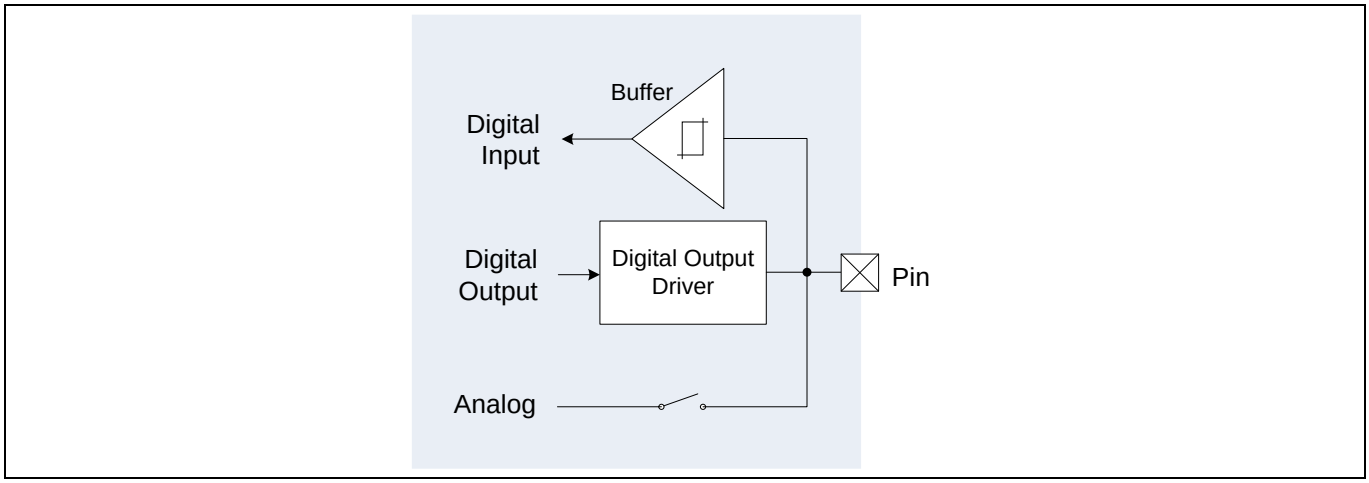

Figure 1 Simplified GPIO Block Diagram

A detailed block diagram of the GPIO structure is available in the I/O System chapter of the **Architecture TRM**. Each pin can act as an input or an output to the digital peripheral such as TCPWM, SCB. Some of the device-specific GPIO can also act as an analog pin for use with ADC. For more details on the analog pins, see the **device datasheet**. TCPWM provides timers, counters, pulse width modulators, and SCB provides serial communication functions. See the **Architecture TRM** for details of peripheral functions.

At any given time, you can use a pin for digital input, digital output, analog pin, or even combinations of these three. For example, if you enable both digital output and input, it provides a digital bidirectional pin. The input buffer provides high impedance to the external input. It is configurable to CMOS, TTL, and Automotive levels. Automotive has a higher threshold level of tripping point than CMOS. See the **Architecture TRM** for different trip levels between CMOS and Automotive.

GPIO Pin Basics

Figure 2 details the Digital Output Driver in **Figure 1**. The digital output from each peripheral drives the pin with a digital output driver. The digital output driver supports different drive modes and slew rate control.

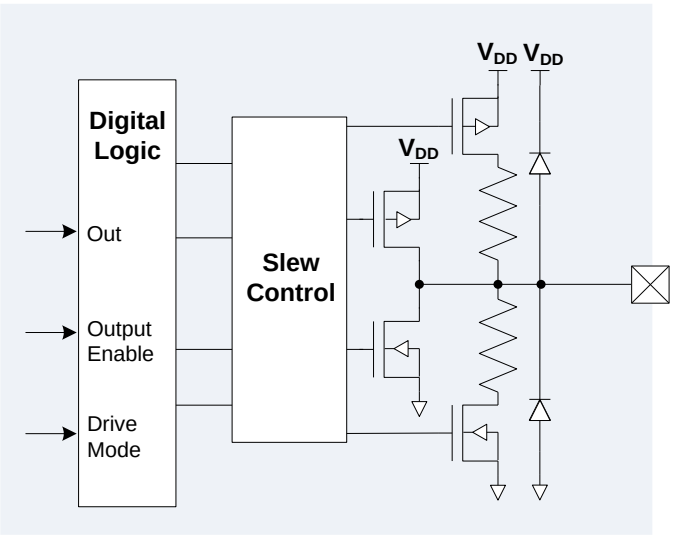

Figure 2 Digital Output Driver

Slew rate control is provided to reduce EMI and crosstalk and is configured using the SLOW bit of the port output configuration register (GPIO_PRTx_CFG_OUT). There are two options: Fast and Slow. Slew rate is set to Fast by default. Use the Slow option when the signals are not speed critical.

Traveo II device supports various drive modes. Drive mode is configured by the DRIVE_MODE field in the Port Configuration (GPIO_PRTx_CFG) register. **Table 1** lists the supported drive modes, and DRIVE_MODE field setting value. See the **Architecture TRM** for output driver block diagram corresponding to drive modes.

Table 1 Drive Modes and Applications

DRIVE_MODE [2:0] Field[1]	Drive Mode	Application Examples
0	High impedance	Interface to analog and digital input. For digital signals, the input buffer is enabled. For analog signals, the input buffer is typically disabled to reduce crowbar current and leakage in low-power designs.
2	Resistive Pull-Up	Interface to open-drain LOW input, such as the tachometer output from motors or a switch connected to ground. Pins can be used for either digital input or digital output.
3	Resistive Pull-Down	Interface to an open-drain HIGH input or a switch connected to VDD. Pins can be used for either digital input or digital output.
4	Open Drain, Drives Low	Provides high impedance in the HIGH state and a strong drive in the LOW state; this configuration is used for I^2C pins. This mode works in conjunction with an external pull-up resistor.
5	Open Drain, Drives High	Provides strong drive in the HIGH state and high impedance in the LOW state. This mode works in conjunction with an external pull-down resistor.
6	Strong	CMOS output drives in both LOW and HIGH states

[1] Setting DRIVE_MODE to 1 is prohibited.

DRIVE_MODE [2:0] Field[1]	Drive Mode	Application Examples
7	Resistive Pull-Up and Down	Adds a series resistor in both HIGH and LOW states

2.2 Startup and Low-Power Behavior

On reset/power-up, all GPIO pins start up in the high-impedance analog state, that is, with the input buffer and output driver disabled. These GPIO pins remain in this mode until the reset is released. During runtime, GPIOs can be configured by writing to the associated registers.

In Sleep mode, GPIO pins are active and can be actively driven by the peripherals, TCPWM, and SCB; only the CPU is inactive in this mode. In DeepSleep mode, the GPIO pins connected to DeepSleep domain peripherals are functional.

Traveo II MCUs have an additional feature that freezes the GPIOs in DeepSleep and Hibernate modes. The freezing of the GPIO is automatically released, when the device comes out of the low power mode. However, note that the GPIOs driven by DeepSleep peripherals are active in DeepSleep mode and are not frozen.

In the case of Hibernate mode, a device reset wakes up the device. This clears the GPIO configuration and pin state and initializes the GPIO pins to high-impedance analog state. Therefore, GPIO reconfiguration is required.

2.3 Interrupt

All port pins have the capability to generate GPIO interrupts. **Figure 3** shows GPIO interrupt input block diagram.

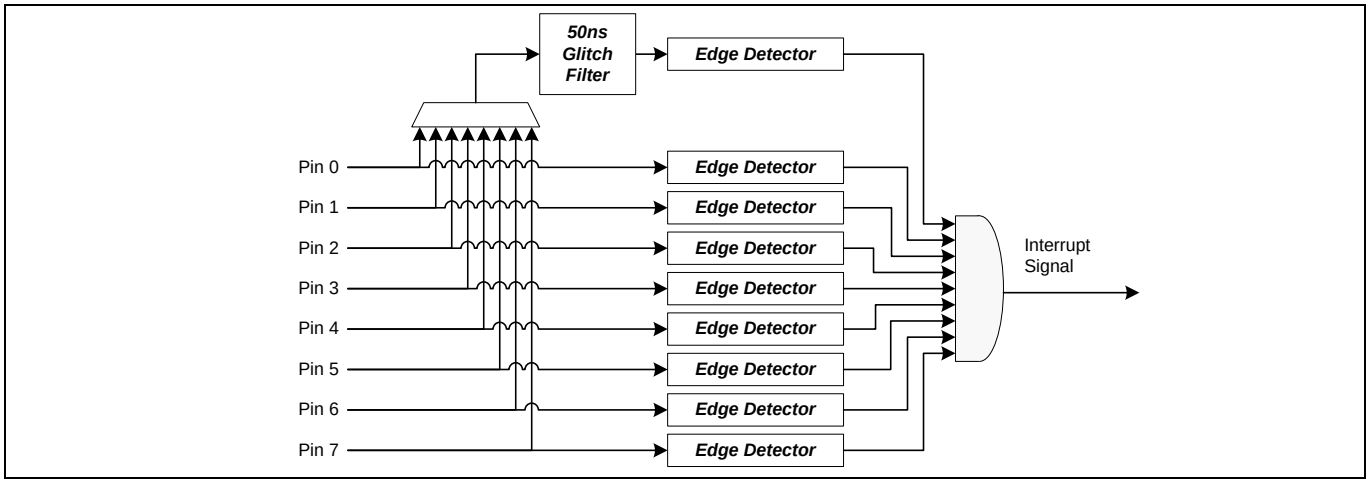

Figure 3 GPIO Edge Detect Block Architecture

One GPIO interrupt signal is generated per port. An edge detector is present at each input pin and glitch filter output. The edge detector can detect any of rising-edge, falling-edge, and both edges in the incoming GPIO signal. The glitch filter can be used by one of the pins of a port at a time. Edge detection type is configured by the EDGEx_SEL field in the Port Interrupt Configuration (`GPIO_PRTx_INTR_CFG`) register. **Table 2** lists the supported edge detection types and EDGEx_SEL field values. See the **Architecture TRM** for the output driver block diagram corresponding to drive modes.

Table 2 Edge Detection Setting

EDGEx_SEL Field	EDGE Type
0	Disable
1	RISING Edge
2	FALLING Edge
3	BOTH Edges

Individual GPIO interrupt signals within a port are ORed together to generate a single interrupt request. Thus, there is one interrupt vector for each port. To determine the port that triggered the interrupt, the GPIO_INTR_CAUSEx registers can be read. The software can read this register to determine the pin(s) or glitch filter signal that caused interrupt activation. The software needs to clear the interrupt cause flags to deactivate the interrupt in the interrupt service routine (ISR).

All I/O pins can be used as wakeup interrupts in Sleep and DeepSleep power mode.

2.4 Configuration of GPIO Output Data

Two types of configurations are available for changing GPIO output data, when GPIO is used as output port:

- GPIO_PRTx_OUT
  The write register changes the output data to the written data value, and the read reflects the output data setting. Note that the register read does not reflect the current input of IO pins. You need to use read-modify-write to retain the output data of other pins.
- GPIO_PRTx_OUT_CLR and GPIO_PRTx_OUT_SET
  These registers can change the output data in the corresponding IO pins without affecting the output data of other IO pins. Writing “1” to the GPIO_PRTx_OUT_CLR register clears the corresponding IO pin to “0”, and writing “0” does not affect the register. Writing “1” to the GPIO_PRTx_OUT_SET register sets the corresponding IO pin to “1”, and writing “0” does not affect the register.

3 GPIO Settings

This section provides practical examples on the GPIO pins usage and sample register bit settings based on the use case with Sample Driver Library (SDL) provided by Cypress. The use cases portrayed in this document are based on CYT2B series. For details on the settings of each series, see the datasheets mentioned in **Related Documents**.

SDL basically has a configuration part and a driver part. The configuration part mainly configures the parameter values for the desired operation. The driver part configures each register based on the parameter values in the configuration part.

You can configure the configuration part according to your system.

3.1 Initializing GPIO

Figure 4 shows the flow to initialize the GPIO pin. In this flow, (1) is performed in the configuration part, and (2) to (10) are performed in the driver part. The suffix “y” indicates the pin bit-field within the port register. The detailed settings are described in the use case.

(0) Configure the parameter values according to the system.

(1) Set the initial output state of the GPIO pin to port the output data set register (OUT_SET.OUTy). This is not required if the GPIO pin is set to input.

(2) Configure the HSIOM connection to the pin to the HSIOM port selection register (PORT_SEL0.IOy_SEL). Port becomes active (Port output) by the HSIOM register configuration.

(3) Select the input buffer to port input buffer configuration register (CFG_IN.VTRIP_SELy_0).

(4) Configure the pin output buffer slew rate to the port output buffer configuration (CFG_OUT.SLOWy).

(5) Configure the output drive strength to the port output buffer configuration (CFG_OUT.DRIVE_SELy).

(6) Configure the drive mode to the port configuration register (CFG.DRIVE_MODE3).

(7) Select the disable or enable interrupt to the port interrupt configuration register (INTR_CFG.EDGEy_SEL).

(8) Clear the pin interrupt to the port interrupt status register (INTR.EDGEy).

(9) Configure the interrupt mask to the port interrupt mask register (INTR_MASK.EDGEy).

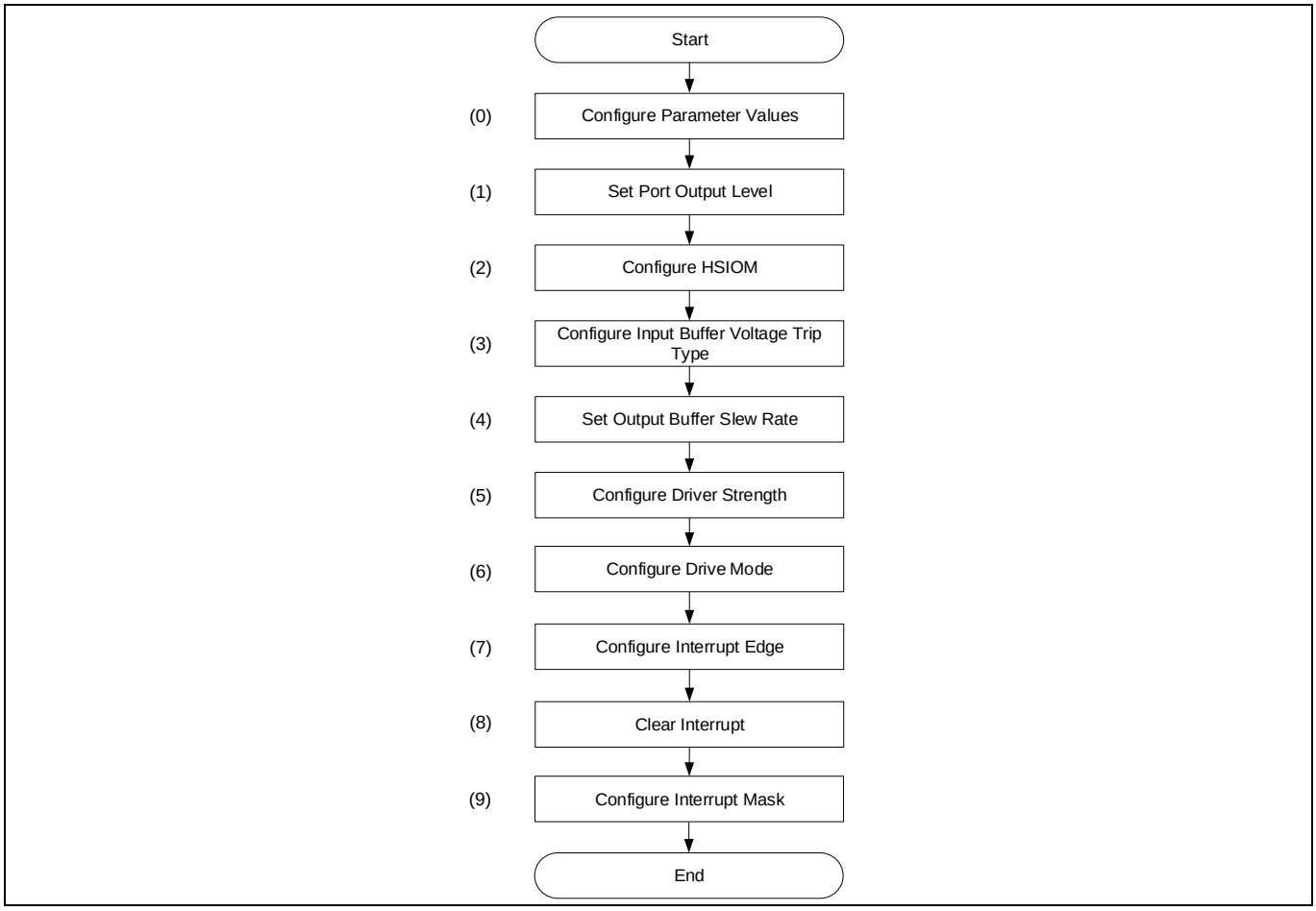

Figure 4 GPIO Pin Initialization Procedure

3.1.1 Example Code to Initialize GPIO in Driver Part

Code Listing 1 demonstrates an example program to initialize GPIO for writing port level in driver part.

The following description will help you understand the register notation of the driver part of SDL:

- `base` signifies the pointer to the port register base address. `pinNum` indicates the pin number within the port register.
- base->unOUT_CLR.u32Register is the GPIO_PRTx_OUT_CLR register mentioned in the **Registers TRM**. “x” signifies the GPIO pin’s port number.
- portAddr**HSIOM**->un**PORT_SEL0**.u32Register is the **HSIOM**_PRTx_**PORT_SEL0** register mentioned in the **Registers TRM**. Other registers are also described in the same manner.
- To improve the register setting performance, the SDL writes a complete 32-bit data to the register. Each bit field is generated and written to the register as the final 32-bit data.

```
tempReg = base->unCFG_IN.u32Register &
~(CY_GPIO_CFG_IN_VTRIP_SEL_MASK << pinNum);
base->unCFG_IN.u32Register = tempReg |
((value & CY_GPIO_CFG_IN_VTRIP_SEL_MASK) << pinNum);
```

See *cyip_gpio_v2.h* under *hdr/rev_x/ip* for more information on the union and structure representation of registers.

GPIO Settings

Code Listing 1 Example for Initializing GPIO in Driver Part

```
cy_en_gpio_status_t Cy_GPIO_Pin_Init(volatile stc_GPIO_PRT_t *base, uint32_t pinNum, const cy_stc_gpio_pin_config_t
*config)
{
    cy_en_gpio_status_t status = CY_GPIO_SUCCESS;

    if((NULL != base) && (NULL != config))
    {
        Cy_GPIO_Write(base, pinNum, config->outVal);
        Cy_GPIO_SetHSIOM(base, pinNum, config->hsiom);
        Cy_GPIO_SetVtrip(base, pinNum, config->vtrip);
        Cy_GPIO_SetSlewRate(base, pinNum, config->slewRate);
        Cy_GPIO_SetDriveSel(base, pinNum, config->driveSel);
        Cy_GPIO_SetDrivemode(base, pinNum, config->driveMode);
        Cy_GPIO_SetInterruptEdge(base, pinNum, config->intEdge);
        Cy_GPIO_ClearInterrupt(base, pinNum);
        Cy_GPIO_SetInterruptMask(base, pinNum, config->intMask);
    }
    else
    {
        status = CY_GPIO_BAD_PARAM;
    }

    return(status);
}
```

(1) Set Port Output Level
(2) Configure HSIOM
(3) Configure Input Buffer Voltage Trip Type
(4) Set Output Buffer Slew Rate
(5) Configure Drive Strength
(6) Configure Drive Mode
(7) Configure Interrupt Edge
(8) Clear Pin Interrupt
(9) Configure Interrupt Mask

Following steps break down the configurations of **Code Listing 1**.

1. Set Port Output Level.

Code Listing 2 Setting Port Output Level

```
__STATIC_INLINE void Cy_GPIO_Write(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    /* Thread-safe: Directly access the pin registers instead of base->OUT */
    if(CY_GPIO_ZERO == value)
    {
        base->unOUT_CLR.u32Register = CY_GPIO_OUT_MASK << pinNum;
    }
    else
    {
        base->unOUT_SET.u32Register = CY_GPIO_OUT_MASK << pinNum;
    }
}
```

If `value` is '0', output state at OUT_CLR.OUT is set to '0'.

If `value` is '1', output state at OUT_SET.OUT is set to '1'.

GPIO Settings

2. Configure HSIOM.

Code Listing 3 Configuring HSIOM

```
__STATIC_INLINE void Cy_GPIO_SetHSIOM(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, en_hsiom_sel_t value)
{
    uint8_t  pinNumForHsiom;
    uint32_t portNum;
    uint32_t tempReg;
    stc_HSIOM_PRT_t* portAddrHSIOM;

    portNum = ((uint32_t)(base) - GPIO_BASE) / GPIO_PRT_SECTION_SIZE;
    portAddrHSIOM = (stc_HSIOM_PRT_t*)(HSIOM_BASE + (HSIOM_PRT_SECTION_SIZE * portNum));

    if(pinNum < CY_GPIO_PRT_HALF)
    {
        pinNumForHsiom = pinNum;
        tempReg = portAddrHSIOM->unPORT_SEL0.u32Register & ~(CY_GPIO_HSIOM_MASK << (pinNumForHsiom <<
CY_GPIO_HSIOM_OFFSET));
        portAddrHSIOM->unPORT_SEL0.u32Register = tempReg | ((value & CY_GPIO_HSIOM_MASK) << (pinNumForHsiom <<
CY_GPIO_HSIOM_OFFSET));
    }
    else
    {
        pinNumForHsiom = pinNum - CY_GPIO_PRT_HALF;
        tempReg = portAddrHSIOM->unPORT_SEL1.u32Register & ~(CY_GPIO_HSIOM_MASK << (pinNumForHsiom <<
CY_GPIO_HSIOM_OFFSET));
        portAddrHSIOM->unPORT_SEL1.u32Register = tempReg | ((value & CY_GPIO_HSIOM_MASK) << (pinNumForHsiom <<
CY_GPIO_HSIOM_OFFSET));
    }
}
```

If `pinNum` is less than four, select HSIOM on port selection 0 PORT_SEL0.IO_SEL.

If `pinNum` is four or greater, select HSIOM on port selection 1 PORT_SEL1.IO_SEL

3. Configure Input Buffer Voltage Trip Type.

Code Listing 4 Configuring Input Buffer Voltage Trip Type

```
__STATIC_INLINE void Cy_GPIO_SetVtrip(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    uint32_t tempReg;

    tempReg = base->unCFG_IN.u32Register & ~(CY_GPIO_CFG_IN_VTRIP_SEL_MASK << pinNum);
    base->unCFG_IN.u32Register = tempReg | ((value & CY_GPIO_CFG_IN_VTRIP_SEL_MASK) << pinNum);
}
```

Select the pin input buffer mode for the IO pin at CFG_IN.VTRIP_SEL

4. Set Output Buffer Slew Rate.

Code Listing 5 Setting Output Buffer Slew Rate

```
__STATIC_INLINE void Cy_GPIO_SetSlewRate(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    uint32_t tempReg;

    tempReg = base->unCFG_OUT.u32Register & ~(CY_GPIO_CFG_OUT_SLOW_MASK << pinNum);
    base->unCFG_OUT.u32Register = tempReg | ((value & CY_GPIO_CFG_OUT_SLOW_MASK) << pinNum);
}
```

Select between fast or slow slew rate for the IO pin at CFG_OUT.SLOW

5. Configure Drive Strength.

Code Listing 6 Configuring Drive Strength

```
_STATIC_INLINE void Cy_GPIO_SetDriveSel(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    uint32_t tempReg;
    uint32_t pinLoc;

    pinLoc = (uint32_t)(pinNum << CY_GPIO_CFG_OUT_DRIVE_OFFSET) + CY_GPIO_CFG_OUT_DRIVE_REG_OFFSET;
    tempReg = base->unCFG_OUT.u32Register & ~(CY_GPIO_CFG_OUT_DRIVE_SEL_MASK << pinLoc);
    base->unCFG_OUT.u32Register = tempReg | ((value & CY_GPIO_CFG_OUT_DRIVE_SEL_MASK) << pinLoc);
}
```

Select drive strength for the IO pin at CFG_OUT.DRIVE_SEL

6. Configure Drive Mode.

Code Listing 7 Configuring Drive Mode

```
__STATIC_INLINE void Cy_GPIO_SetDrivemode(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    uint32_t tempReg;
    uint32_t pinLoc;

    pinLoc = pinNum << CY_GPIO_DRIVE_MODE_OFFSET;
    tempReg = (base->unCFG.u32Register & ~(CY_GPIO_CFG_DM_MASK << pinLoc));
    base->unCFG.u32Register = tempReg | ((value & CY_GPIO_CFG_DM_MASK) << pinLoc);
}
```

Select the drive mode for the IO pin at CFG.DRIVE_MODE

Set CFG.IN_EN to '1' to enable input buffer

7. Configure Interrupt Edge.

Code Listing 8 Configuring Interrupt Edge

```
__STATIC_INLINE void Cy_GPIO_SetInterruptEdge(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    uint32_t tempReg;
    uint32_t pinLoc;

    pinLoc = pinNum << CY_GPIO_INTR_CFG_OFFSET;
    tempReg = base->unINTR_CFG.u32Register & ~(CY_GPIO_INTR_EDGE_MASK << pinLoc);
    base->unINTR_CFG.u32Register = tempReg | ((value & CY_GPIO_INTR_EDGE_MASK) << pinLoc);
}
```

Select interrupt edge for IO pin at INTR_CFG.EDGE_SEL.

8. Clear Pin Interrupt.

Code Listing 9 Clearing Pin Interrupt

```
__STATIC_INLINE void Cy_GPIO_ClearInterrupt(volatile stc_GPIO_PRT_t* base, uint32_t pinNum)
{
    /* Any INTR MMIO registers AHB clearing must be preceded with an AHB read access */
    (void)base->unINTR.u32Register;

    base->unINTR.u32Register = CY_GPIO_INTR_STATUS_MASK << pinNum;

    /* This read ensures that the initial write has been flushed out to the hardware */
    (void)base->unINTR.u32Register;
}
```

Clear the triggered pin interrupt at INTR.EDGE.

9. Configure Interrupt Mask

Code Listing 10 Configuring Interrupt Mask

```
__STATIC_INLINE void Cy_GPIO_SetInterruptMask(volatile stc_GPIO_PRT_t* base, uint32_t pinNum, uint32_t value)
{
    uint32_t tempReg;

    tempReg= base->unINTR_MASK.u32Register & ~(CY_GPIO_INTR_EN_MASK << pinNum);
    base->unINTR_MASK.u32Register = tempReg | ((value & CY_GPIO_INTR_EN_MASK) << pinNum);
}
```

Configure the pin interrupt to be forwarded to the CPU NVIC at INTR_MASK.EDGE

3.2 Toggling a Port Level

The simple use case of a GPIO is to set the output of a pin to HIGH or LOW in firmware. This example demonstrates the use case of configuring a GPIO pin as an output and toggling the port level in CYT2B7 series. For writing a port level, the drive mode of the GPIO pin must be set to Strong. For details, see **Table 1**.

CPU alternately sets the pin to “0” or “1” periodically.

Example configuration:

- Port number: P19.3
- Initial state: Low
- Input/Output: Output
- Drive mode: Strong
- Functionality configuration (HSIOM): GPIO
- Interrupt function: Unused
- Slew rate: Fast
- Output driver strength: Strong (Full Drive)

3.2.1 Configuration

Table 3 lists the parameters of the configuration part in SDL for writing a port level.

Table 3 List of GPIO Pin Parameters

Parameters	Description	Value
.outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
.driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down 8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives LOW. Input buffer ON. 12: Open Drain, Drives HIGH. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	CY_GPIO_DM_STRONG_IN_OFF = 6ul See point **1** of the SDL description.
.hsiom	Sets connection for IO pin 0 route.	P19_3_GPIO See point **3** of the SDL description.
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	0ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	0ul
.vtrip	Selects the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Selects slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

The following description will help you understand the configuration of GPIO port, pin, and HSIOM of this example in SDL:

1. The constants to be used for setting the drive mode of the pin are defined in the GPIO driver header *cy_gpio.h* in the *common\src\drivers\gpio* folder

```
#define CY_GPIO_DM_ANALOG               (0x00ul)
/*Analog High-Z. Input buffer off */
#define CY_GPIO_DM_PULLUP_IN_OFF        (0x02ul)
/*Resistive Pull-Up. Input buffer off */
#define CY_GPIO_DM_PULLDOWN_IN_OFF      (0x03ul)
/*Resistive Pull-Down. Input buffer off */
#define CY_GPIO_DM_OD_DRIVESLOW_IN_OFF (0x04ul)
/*Open Drain, Drives Low. Input buffer off */
#define CY_GPIO_DM_OD_DRIVESHIGH_IN_OFF(0x05ul)
/*Open Drain, Drives High. Input buffer off */
#define CY_GPIO_DM_STRONG_IN_OFF        (0x06ul)
/*Strong Drive. Input buffer off */
#define CY_GPIO_DM_PULLUP_DOWN_IN_OFF   (0x07ul)
/*Resistive Pull-Up/Down. Input buffer off */
#define CY_GPIO_DM_HIGHZ                (0x08ul)
/*Digital High-Z. Input buffer on */
#define CY_GPIO_DM_PULLUP               (0x0Aul)
/*Resistive Pull-Up. Input buffer on */
#define CY_GPIO_DM_PULLDOWN             (0x0Bul)
/*Resistive Pull-Down. Input buffer on */
#define CY_GPIO_DM_OD_DRIVESLOW         (0x0Cul)
/*Open Drain, Drives Low. Input buffer on */
#define CY_GPIO_DM_OD_DRIVESHIGH        (0x0Dul)
/*Open Drain, Drives High. Input buffer on */
#define CY_GPIO_DM_STRONG               (0x0Eul)
/*Strong Drive. Input buffer on */
#define CY_GPIO_DM_PULLUP_DOWN          (0x0Ful)
/*Resistive Pull-Up/Down. Input buffer on */
```

2. Each port number is defined in the device header in the *header* folder.

   For example, if the revision of the MCU silicon is Revision B, the device header is in *hdr/rev_b*.

```
#define GPIO_PRT19      ((volatile stc_GPIO_PRT_t*) &GPIO->PRT[19]
```

   The port number and pin number are defined together in the project-specific header in the same folder.

   For example, if the MCU is CYT2B7 series on a Revision C CPU board, you can find the following definition in *bb_bsp_tviibe1m_revc.h.*

```
#define CY_LED0_PORT          GPIO_PRT19
#define CY_LED0_PIN           3
#define CY_LED0_PIN_MUX       P19_3_GPIO
```

3. Each GPIO pin has its dedicated HSIOM selection. The HSIOM of the specific pin and its parameter value can be found in the device GPIO header.

GPIO Settings

For example, if the MCU is 176-pin CYT2B7 series, the header file is *gpio_cyt2b7_176_lqfp.h*.

For 176-pin CYT2B7 series MCU, the HSIOM connections for P19.3 are as follows:

```
/* P19.3 */
P19_3_GPIO                          =  0,     /* GPIO controls 'out' */
P19_3_AMUXA                         =  4,     /* Analog mux bus A */
P19_3_AMUXB                         =  5,     /* Analog mux bus B */
P19_3_AMUXA_DSI                     =  6,
/* Analog mux bus A, DSI control */
P19_3_AMUXB_DSI                     =  7,
/* Analog mux bus B, DSI control */
P19_3_TCPWM0_LINE28                 =  8,
/* Digital Active - tcpwm[0].line[28]:2 */
P19_3_TCPWM0_LINE_COMPL27           =  9,
/* Digital Active - tcpwm[0].line_compl[27]:2 */
P19_3_TCPWM0_TR_ONE_CNT_IN84     = 10,
/* Digital Active - tcpwm[0].tr_one_cnt_in[84]:2 */
P19_3_TCPWM0_TR_ONE_CNT_IN82     = 11,
/* Digital Active - tcpwm[0].tr_one_cnt_in[82]:2 */
P19_3_TCPWM0_TR_ONE_CNT_IN1540   = 16,
/* Digital Active - tcpwm[0].tr_one_cnt_in[1540]:0 */
P19_3_SCB2_UART_CTS                 = 17,
/* Digital Active - scb[2].uart_cts:1 */
P19_3_SCB2_SPI_SELECT0              = 19,
/* Digital Active - scb[2].spi_select0:1 */
P19_3_PERI_TR_IO_INPUT29            = 26,
/* Digital Active - peri.tr_io_input[29]:0 */
```

See the device datasheet for the specific HSIOM functional connections for each pin.

Code Listing 11 demonstrates an example program to initialize GPIO and toggle port level in the configuration part.

Code Listing 11 Example for Initializing GPIO and Toggling Port Level in Configuration Part

```
cy_stc_gpio_pin_config_t user_led_port_pin_cfg =
{
    .outVal    = 0ul,       /* Pin output state */
    .driveMode = CY_GPIO_DM_STRONG_IN_OFF,    /* Drive mode */
    .hsiom     = CY_LED0_PIN_MUX,             /* HSIOM selection */
    .intEdge   = 0ul,       /* Interrupt Edge type */
    .intMask   = 0ul,       /* Interrupt enable mask */
    .vtrip     = 0ul,       /* Input buffer voltage trip type */
    .slewRate  = 0ul,       /* Output buffer slew rate */
    .driveSel  = 0ul,       /* Drive strength */
};

int main(void)
{
```

(0) Configure Parameter Values

Code Listing 11 Example for Initializing GPIO and Toggling Port Level in Configuration Part

```
    SystemInit();

    __enable_irq();

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    user_led_port_pin_cfg.hsiom = CY_LED0_PIN_MUX;
    Cy_GPIO_Pin_Init(CY_LED0_PORT, CY_LED0_PIN, &user_led_port_pin_cfg);

    for(;;)
    {
        // Wait 0.05 [s]
        Cy_SysTick_DelayInUs(50000ul);

        Cy_GPIO_Inv(CY_LED0_PORT, CY_LED0_PIN);
    }
}
```

Port number. Pin number. GPIO pin structure.

Output state is set to HIGH and LOW, successively.

3.2.2 Example Program of Toggling Port Level

Code Listing 12 demonstrates an example program to toggle port level in the driver part.

Code Listing 12 Example for Toggling Port Level in Driver Part

```
__STATIC_INLINE void Cy_GPIO_Inv(volatile stc_GPIO_PRT_t* base, uint32_t pinNum)
{
    base->unOUT_INV.u32Register = CY_GPIO_OUT_MASK << pinNum;
}
```

Output state is set to the inverse of the current output state at OUT_INV.OUT3

3.3 Reading an Input

This example demonstrates a use case for reading from a GPIO pin in CYT2B7 series. Enable the digital input buffer for external digital input and read the port level. If the GPIO pin is set to read, the drive mode of the pin must be set to Digital High Impedance. For details, see **Table 1**.

Example configuration:

- Port number: P6.5
- Initial state: Low
- Input/Output: Input
- Drive mode: Digital High impedance
- Functionality configuration (HSIOM): 0 (GPIO)
- Interrupt function: Unused

3.3.1 Configuration

Table 4 lists the parameters of the configuration part in SDL for reading an input.

Table 4 List of GPIO Pin Parameters

Parameters	Description	Value
.outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
.driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down 8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives Low. Input buffer ON. 12: Open Drain, Drives High. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	CY_GPIO_DM_HIGHZ = 8ul
.hsiom	Sets the connection for IO pin 0 route.	P6_5_GPIO See point 2 of the SDL description.
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	0ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	0ul
.vtrip	Selects the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Selects slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

The following description will help you understand the configuration of GPIO port, pin, and HSIOM of this example in SDL:

1. Each port number is defined in the device header in the *header* folder.

   For example, if the revision of the MCU silicon is Revision B, the device header can be found in *hdr/rev_b*.

```
#define GPIO_PRT6       ((volatile stc_GPIO_PRT_t*) &GPIO->PRT[6]
```

   The port number and pin number are defined together in the project-specific header in the same folder.

   For example, if the MCU is CYT2B7 series on a Revision C CPU board, you can find the following definition in *bb_bsp_tviibe1m_revc.h*.

```
#define CY_CB_BUTTON_PORT       GPIO_PRT6
#define CY_CB_BUTTON_PIN        5
#define CY_CB_BUTTON_PIN_MUX    P6_5_GPIO
```

2. Each GPIO pin has its dedicated HSIOM selection. The HSIOM of a specific pin and its parameter value is in the device GPIO header.

   For example, if the MCU is 176-pin CYT2B7 series, the header file is *gpio_cyt2b7_176_lqfp.h*.

   For 176-pin CYT2B7 series MCU, the HSIOM connections for P6.5 are as follows:

```
/* P6.5 */
P6_5_GPIO                       =  0,     /* GPIO controls 'out' */
P6_5_AMUXA                      =  4,     /* Analog mux bus A */
P6_5_AMUXB                      =  5,     /* Analog mux bus B */
P6_5_AMUXA_DSI                  =  6,
/* Analog mux bus A, DSI control */
P6_5_AMUXB_DSI                  =  7,
/* Analog mux bus B, DSI control */
P6_5_TCPWM0_LINE2               =  8,
/* Digital Active - tcpwm[0].line[2]:0 */
P6_5_TCPWM0_LINE_COMPL258       =  9,
/* Digital Active - tcpwm[0].line_compl[258]:0 */
P6_5_TCPWM0_TR_ONE_CNT_IN6      = 10,
/* Digital Active - tcpwm[0].tr_one_cnt_in[6]:0 */
P6_5_TCPWM0_TR_ONE_CNT_IN775    = 11,
/* Digital Active - tcpwm[0].tr_one_cnt_in[775]:0 */
P6_5_SCB4_SPI_SELECT2           = 19,
/* Digital Active - scb[4].spi_select2:0 */
P6_5_LIN0_LIN_EN4               = 20,
/* Digital Active - lin[0].lin_en[4]:0 */
```

See the device datasheet for the specific HSIOM functional connections for each pin.

GPIO Settings

Code Listing 13 demonstrates an example program for reading an input in the configuration part.

Code Listing 13 Example for Reading an Input in Configuration Part

```
#define USER_BUTTON_PORT        CY_CB_BUTTON_PORT
#define USER_BUTTON_PIN         CY_CB_BUTTON_PIN
#define USER_BUTTON_PIN_MUX     CY_CB_BUTTON_PIN_MUX

cy_stc_gpio_pin_config_t user_button_port_pin_cfg =
{
    .outVal    = 0ul,       /* Pin output state */
    .driveMode = CY_GPIO_DM_HIGHZ,    /* Drive mode */
    .hsiom     = USER_BUTTON_PIN_MUX, /* HSIOM selection */
    .intEdge   = 0ul,       /* Interrupt Edge type */
    .intMask   = 0ul,       /* Interrupt enable mask */
    .vtrip     = 0ul,       /* Input buffer voltage trip type */
    .slewRate  = 0ul,       /* Output buffer slew rate */
    .driveSel  = 0ul,       /* Drive strength */
};

int main(void)
{
    SystemInit();

    __enable_irq(); /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    Cy_GPIO_Pin_Init(USER_BUTTON_PORT, USER_BUTTON_PIN, &user_button_port_pin_cfg);

    // Detect falling edge of the button GPIO
    uint32_t curLevel      = 0ul;

    for(;;)
    {
       // Get the current button level
       curLevel = Cy_GPIO_Read(USER_BUTTON_PORT, USER_BUTTON_PIN);
       :
}
```

(0) Configure Parameter Values

Port number.

Pin number.

GPIO pin structure.

The current logic level on the input buffer of P6.5 is read

3.3.2 Example Program of Reading a Port Level

Code Listing 14 demonstrates an example program to read an input in the driver part.

Code Listing 14 Example of Reading an Input in Driver Part

```
__STATIC_INLINE uint32_t Cy_GPIO_Read(volatile stc_GPIO_PRT_t* base, uint32_t pinNum)
{
    return (base->unIN.u32Register >> (pinNum)) & CY_GPIO_IN_MASK;
}
```

Reads the current pin status on the input buffer of the pin from IN.IN5

3.4 Interrupt

This example demonstrates a use case for interrupt generation from a pin in CYT2B7 series. This pin uses one IRQ terminal. Thus, the interrupt source must be identified in the ISR. For the input port, enable the interrupt edge and interrupt mask.

When P6.5 detects a falling edge, the interrupt occurs. For more information, see the Initial Setting Procedure section in AN219842 listed in **Related Documents**.

Example configuration:

- Port number: P6.5
- Initial state: Low
- Input/Output: Input
- Drive mode: Digital High impedance
- Functionality configuration (HSIOM): GPIO
- Interrupt edge type: Falling edge
- Interrupt function: Used
- Interrupt number: 3
- Interrupt priority: 0

3.4.1 Configuration

Table 5 lists the parameters of the configuration part in SDL for interrupt on pin P6.5.

Table 5 List of Parameters for Interrupt on Pin P6.5

Parameters	Description	Value
.outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
.driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down	CY_GPIO_DM_HIGHZ = 8ul

GPIO Settings

Parameters	Description	Value
	8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives Low. Input buffer ON. 12: Open Drain, Drives High. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	
.hsiom	Sets the connection for IO pin 0 route.	P6_5_GPIO See point **2** of the SDL description.
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	CY_GPIO_INTR_FALLING = 2ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	1ul
.vtrip	Selects the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Selects slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

Figure 5 shows the flow of the pin interrupt configuration example. Initialize the port with GPIO input setting using the interrupt.

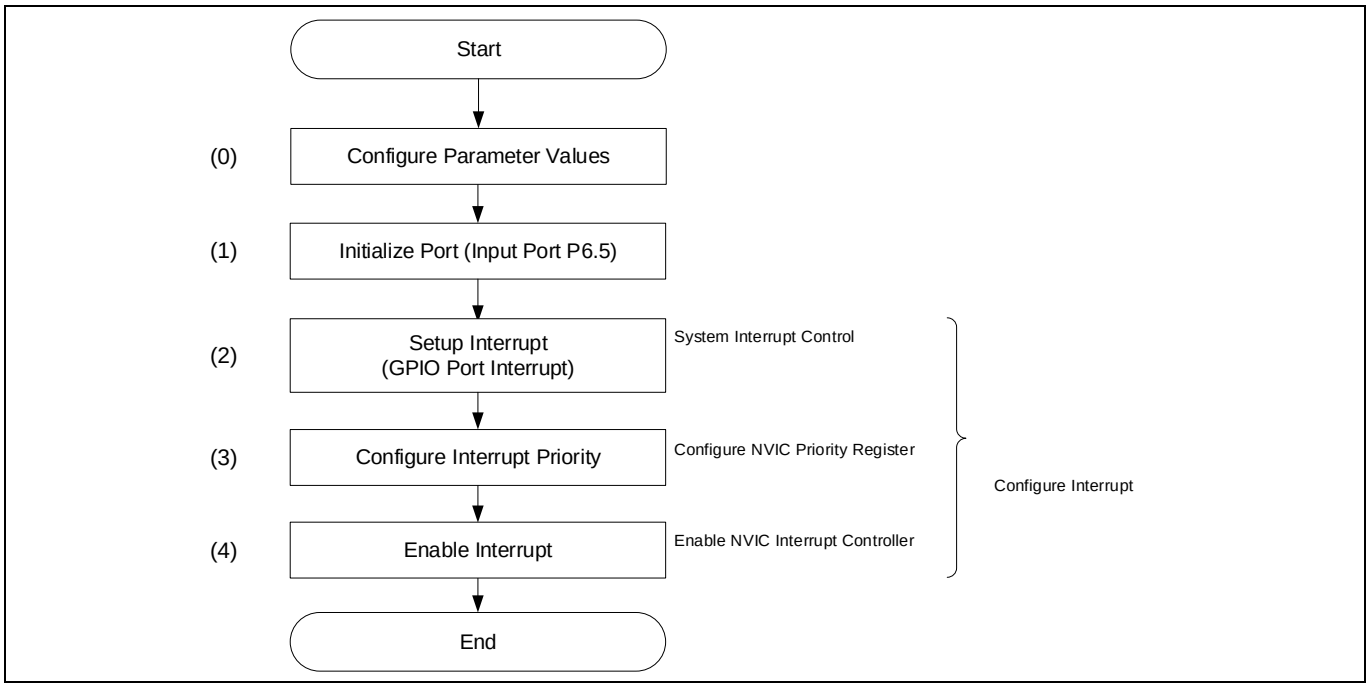

Figure 5 Pin Interrupt Generation Procedure

The following description will help you understand the configuration of GPIO port, pin, and HSIOM of this example in SDL:

1. Each port number is defined in the device header in the *header* folder.

   For example, if the revision of the MCU silicon is Revision B, the device header is in *hdr/rev_b.*

   ```
   #define GPIO_PRT6       ((volatile stc_GPIO_PRT_t*) &GPIO->PRT[6]
   ```

   The port number, pin number, and interrupt number are defined together in the project-specific header in the same folder.

   For example, if the MCU is CYT2B7 series on a Revision C CPU board, you can find the the following definition in *bb_bsp_tviibe1m_revc.h.*

   ```
   #define CY_CB_BUTTON_PORT       GPIO_PRT6
   #define CY_CB_BUTTON_PIN        5
   #define CY_CB_BUTTON_PIN_MUX    P6_5_GPIO
   #define CY_CB_BUTTON_IRQN       ioss_interrupts_gpio_6_IRQn
   ```

2. The HSIOM of the specific pin and its parameter value is in the device GPIO header.

   For example, if the MCU is 176-pin CYT2B7 series, the header file is *gpio_cyt2b7_176_lqfp.h*.

   For 176-pin CYT2B7 series MCU, the HSIOM connections for P6.5 are as follows:

   ```
   /* P6.5 */
   P6_5_GPIO                       =  0,     /* GPIO controls 'out' */
   P6_5_AMUXA                      =  4,     /* Analog mux bus A */
   P6_5_AMUXB                      =  5,     /* Analog mux bus B */
   ```

GPIO Settings

```
P6_5_AMUXA_DSI                  =  6,
/* Analog mux bus A, DSI control */
P6_5_AMUXB_DSI                  =  7,
/* Analog mux bus B, DSI control */
P6_5_TCPWM0_LINE2               =  8,
/* Digital Active - tcpwm[0].line[2]:0 */
P6_5_TCPWM0_LINE_COMPL258       =  9,
/* Digital Active - tcpwm[0].line_compl[258]:0 */
P6_5_TCPWM0_TR_ONE_CNT_IN6      = 10,
/* Digital Active - tcpwm[0].tr_one_cnt_in[6]:0 */
P6_5_TCPWM0_TR_ONE_CNT_IN775    = 11,
/* Digital Active - tcpwm[0].tr_one_cnt_in[775]:0 */
P6_5_SCB4_SPI_SELECT2           = 19,
/* Digital Active - scb[4].spi_select2:0 */
P6_5_LIN0_LIN_EN4               = 20,
/* Digital Active - lin[0].lin_en[4]:0 */
```

3. The constants to be used for setting the interrupt trigger type on the pin are defined in the GPIO driver header *cy_gpio.h* in the*common\src\drivers\gpio* folder.

```
#define CY_GPIO_INTR_DISABLE     (0x00ul)
/**< \brief Disable the pin interrupt generation */
#define CY_GPIO_INTR_RISING      (0x01ul)
/**< \brief Rising-Edge interrupt */
#define CY_GPIO_INTR_FALLING     (0x02ul)
/**< \brief Falling-Edge interrupt */
#define CY_GPIO_INTR_BOTH        (0x03ul)
/**< \brief Both-Edge interrupt */
```

4. The HSIOM of the specific pin and its parameter value is in the device header.

```
typedef enum {
:
  ioss_interrupts_gpio_6_IRQn       =  27,
  /*!<  27 [DeepSleep] GPIO Port Interrupt #6 */
:
} cy_en_intr_t;
```

See the device datasheet for the specific HSIOM functional connections and the interrupts assignments.

GPIO Settings

Code Listing 15 demonstrates an example program to GPIO interrupt generation in the configuration part.

Code Listing 15 Example of GPIO Interrupt in Configuration Part

```
#define USER_BUTTON_PORT        CY_CB_USER_BUTTON_PORT
#define USER_BUTTON_PIN         CY_CB_USER_BUTTON_PIN
#define USER_BUTTON_PIN_MUX     CY_CB_USER_BUTTON_PIN_MUX
#define USER_BUTTON_IRQ         CY_CB_USER_BUTTON_IRQN

cy_stc_gpio_pin_config_t user_button_port_pin_cfg =
{
    .outVal    = 0ul,       /* Pin output state */
    .driveMode = CY_GPIO_DM_HIGHZ,      /* Drive mode */
    .hsiom     = USER_BUTTON_PIN_MUX,   /* HSIOM selection */
    .intEdge   = CY_GPIO_INTR_FALLING,  /* Interrupt Edge type */
    .intMask   = 1ul,       /* Interrupt enable mask */
    .vtrip     = 0ul,       /* Input buffer voltage trip type */
    .slewRate  = 0ul,       /* Output buffer slew rate */
    .driveSel  = 0ul,       /* Drive strength */
};

/* Setup GPIO for BUTTON1 interrupt */
const cy_stc_sysint_irq_t irq_cfg =
{
    .sysIntSrc  = USER_BUTTON_IRQ,
    .intIdx     = CPUIntIdx3_IRQn,
    .isEnabled  = true,
};

int main(void)
{
    SystemInit();

    __enable_irq(); /* Enable global interrupts. */
    :
    Cy_GPIO_Pin_Init(USER_BUTTON_PORT, USER_BUTTON_PIN, &user_button_port_pin_cfg);

    Cy_SysInt_InitIRQ(&irq_cfg);
    Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, ButtonIntHandler);

    NVIC_SetPriority(irq_cfg.intIdx, 0);
    NVIC_EnableIRQ(irq_cfg.intIdx);

    for(;;);
}
```

(0) Configure Parameter Values

(1) Initialize Port (Input port P6.5)

(2) Setup Interrupt (GPIO Port Interrupt)

(3) Configure Interrupt Priority

(4) Enable Interrupt

GPIO Settings

Figure 6 shows an interrupt handler. When the input port detects an interrupt edge, the interrupt handler is activated. First, read the interrupt status and identify which port pin is being interrupted. Next, clear the interrupt status to detect the next interrupt. **Code Listing 16** shows the code example for the interrupt handler.

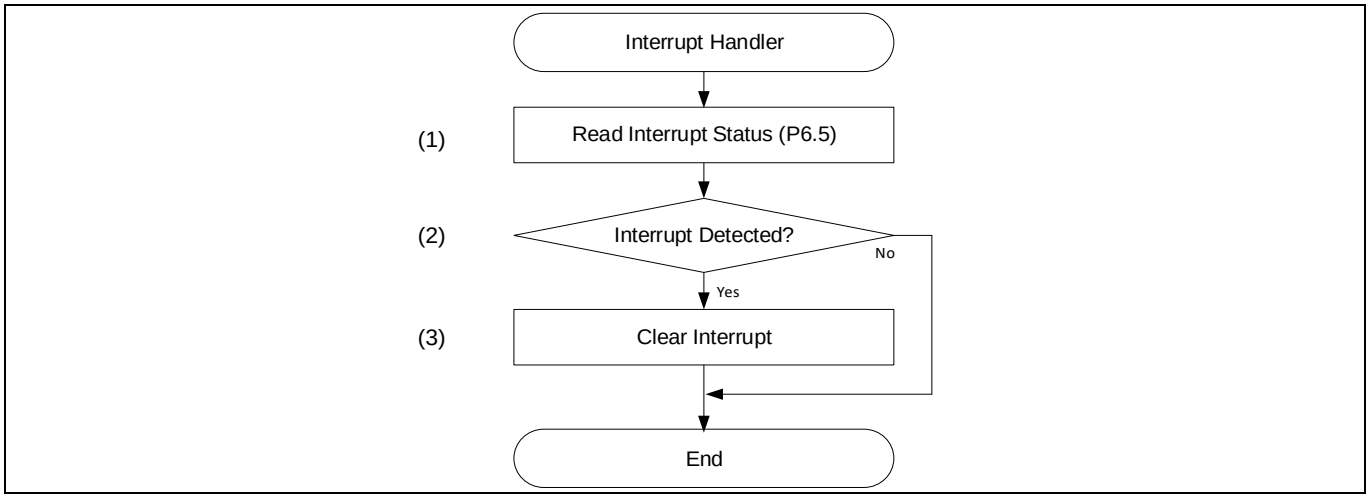

Figure 6 Example of Interrupt Handler Flow

Code Listing 16 Example of Interrupt Handler

```
void IntHandler(void)
{
    uint32_t intStatus;

    /* If falling edge detected */
    intStatus = Cy_GPIO_GetInterruptStatusMasked(USER_BUTTON_PORT, USER_BUTTON_PIN);
    if (intStatus != 0ul)

    {
        Cy_GPIO_ClearInterrupt(USER_BUTTON_PORT, USER_BUTTON_PIN);
    }
}
```

(1) Read Interrupt Status (P6.5)

(2) Interrupt Detected?

(3) Clear Interrupt

Code Listing 17 shows an example program for step **(1) Read Interrupt Status** of the driver part.

Code Listing 17 Example Program for Reading Interrupt Status in Driver Part

```
__STATIC_INLINE uint32_t Cy_GPIO_GetInterruptStatusMasked(volatile stc_GPIO_PRT_t* base, uint32_t pinNum)
{
    return (base->unINTR_MASKED.u32Register >> pinNum) & CY_GPIO_INTR_MASKED_MASK;
}
```

Return the state of pin interrupt mask to be forwarded to CPU interrupt controller at INTR_MASKED.EDGE5

3.5 Peripheral Function

This section explains allocating the peripheral functions to the IO pin. Peripheral function is selected by HSIOM, default setting is GPIO. The pin has several specific functions and can be selected.

See the **device datasheet** for the specific connections available for each pin.

An example of analog port, SCB port, TCPWM port configuration is shown below.

3.5.1 SCB Port Configuration

This example demonstrates the configuration of a port in CYT2B7 series associated with Serial Communication Block (SCB), which is configured as UART. In this example, SCB3 (UART) port (P13.0/P13.1) is used, and it is assigned to ACT_5 of the HSIOM pin connection. Configure the UART function after setting the Rx/Tx port.

For details on selecting an appropriate port for each function, see the Alternate Function Pin Assignments section in the **datasheet**. For details on UART settings, see AN220119 listed in **Related Documents**.

Figure 7 shows the signal path as SCB port of P13.0 and P13.1. To use SCB ch3, configure HSIOM to SCB3_RX/TX.

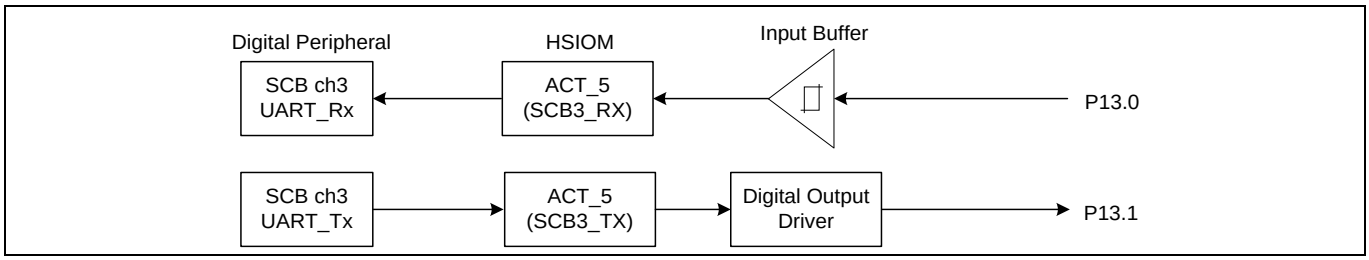

Figure 7 SCB Port Configuration

Example configuration:

RX port

- Port number: P13.0
- Initial state: Low
- Input/Output: Input
- Drive mode: Digital High impedance
- Functionality configuration (HSIOM): SCB (UART Rx)
- Interrupt function: Unused

TX port

- Port number: P13.1
- Initial state: Low
- Input/Output: Output
- Drive mode: Strong
- Functionality configuration (HSIOM): SCB (UART Tx)
- Interrupt function: Unused

GPIO Settings

Table 6 and Table 7 list the parameters of the configuration part in SDL for SCB RX port and TX port, respectively.

Table 6 List of GPIO Pin Parameters of SCB RX port

Parameters	Description	Value
outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down 8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives Low. Input buffer ON. 12: Open Drain, Drives High. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	CY_GPIO_DM_HIGHZ = 8ul
.hsiom	Sets connection for IO pin 0 route.	P13_0_SCB3_UART_RX See point 2 of the SDL description.
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	0ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	0ul
.vtrip	Selects the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Selects slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

Table 7 List of GPIO Pin Parameters of SCB TX Port

Parameters	Description	Value
.outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
.driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down 8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives Low. Input buffer ON. 12: Open Drain, Drives High. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	CY_GPIO_DM_STRONG_IN_OFF = 6ul
.hsiom	Sets the connection for IO pin 0 route.	P13_1_SCB3_UART_TX See point **2** of the SDL description.
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	0ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	0ul
.vtrip	Selects the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Selects slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

The following description will help you understand the configuration of GPIO port, pin, and HSIOM of this example in SDL:

1. Each port number is defined in the device header in the *header* folder.

   For example, if the revision of the MCU silicon is Revision B, the device header is in *hdr/rev_b*.

   ```
   #define GPIO_PRT13      ((volatile stc_GPIO_PRT_t*) &GPIO->PRT[13]
   ```

   The port number and pin number are defined together in the project-specific header in the same folder.

   For example, if the MCU is CYT2B7 series on a Revision C CPU board, you can find the following definition in *bb_bsp_tviibe1m_revc.h*.

   ```
   #define CY_USB_SCB_UART_RX_PORT         GPIO_PRT13
   #define CY_USB_SCB_UART_RX_PIN          0
   #define CY_USB_SCB_UART_RX_MUX          P13_0_SCB3_UART_RX
   #define CY_USB_SCB_UART_TX_PORT         GPIO_PRT13
   #define CY_USB_SCB_UART_TX_PIN          1
   #define CY_USB_SCB_UART_TX_MUX          P13_1_SCB3_UART_TX
   ```

2. The HSIOM of the specific pin and its parameter value is in the device GPIO header.

   For example, if the MCU is 176-pin CYT2B7 series, the header file is *gpio_**cyt2b7_176**_lqfp.h*.

   For 176-pin CYT2B7 series MCU, the HSIOM connections for P13.0 and P13.1 are as follows:

   ```
   /* P13.1 */
   P13_0_GPIO                      =  0,    /* GPIO controls 'out' */
   P13_0_AMUXA                     =  4,    /* Analog mux bus A */
   P13_0_AMUXB                     =  5,    /* Analog mux bus B */
   P13_0_AMUXA_DSI                 =  6,
   /* Analog mux bus A, DSI control */
   P13_0_AMUXB_DSI                 =  7,
   /* Analog mux bus B, DSI control */
   P13_0_TCPWM0_LINE264            =  8,
   /* Digital Active - tcpwm[0].line[264]:0 */
   P13_0_TCPWM0_LINE_COMPL43       =  9,
   /* Digital Active - tcpwm[0].line_compl[43]:0 */
   P13_0_TCPWM0_TR_ONE_CNT_IN792   = 10,
   /* Digital Active - tcpwm[0].tr_one_cnt_in[792]:0 */
   P13_0_TCPWM0_TR_ONE_CNT_IN130   = 11,
   /* Digital Active - tcpwm[0].tr_one_cnt_in[130]:0 */
   P13_0_PASS0_SAR_EXT_MUX_SEL6    = 16,
   /* Digital Active - pass[0].sar_ext_mux_sel[6] */
   P13_0_SCB3_UART_RX              = 17,
   /* Digital Active - scb[3].uart_rx:0 */
   P13_0_SCB3_SPI_MISO             = 19,
   /* Digital Active - scb[3].spi_miso:0 */
   ```

GPIO Settings

```
    /* P13.1 */
    P13_1_GPIO                      =  0,    /* GPIO controls 'out' */
    P13_1_AMUXA                     =  4,    /* Analog mux bus A */
    P13_1_AMUXB                     =  5,    /* Analog mux bus B */
    P13_1_AMUXA_DSI                 =  6,
    /* Analog mux bus A, DSI control */
    P13_1_AMUXB_DSI                 =  7,
    /* Analog mux bus B, DSI control */
    P13_1_TCPWM0_LINE44             =  8,
    /* Digital Active - tcpwm[0].line[44]:0 */
    P13_1_TCPWM0_LINE_COMPL264      =  9,
    /* Digital Active - tcpwm[0].line_compl[264]:0 */
    P13_1_TCPWM0_TR_ONE_CNT_IN132   = 10,
    /* Digital Active - tcpwm[0].tr_one_cnt_in[132]:0 */
    P13_1_TCPWM0_TR_ONE_CNT_IN793   = 11,
    /* Digital Active - tcpwm[0].tr_one_cnt_in[793]:0 */
    P13_1_PASS0_SAR_EXT_MUX_SEL7    = 16,
    /* Digital Active - pass[0].sar_ext_mux_sel[7] */
    P13_1_SCB3_UART_TX              = 17,
    /* Digital Active - scb[3].uart_tx:0 */
    P13_1_SCB3_I2C_SDA              = 18,
    /* Digital Active - scb[3].i2c_sda:0 */
    P13_1_SCB3_SPI_MOSI             = 19,
    /* Digital Active - scb[3].spi_mosi:0 */
```

See the device datasheet for the specific HSIOM functional connections for each pin.

Code Listing 18 demonstrates an example program to initialize GPIO for SCB RX/ TX in the configuration part. The example program for the driver part is the same as **Code Listing 1**.

Code Listing 18 Example for Initializing GPIO for SCB RX/ TX in Configuration Part

```
int main(void)
{
    SystemInit();
   :
      __enable_irq(); /* Enable global interrupts. */

   /* Initialize Port and Clock */
   Peripheral_Initialization();
   :
}

void Peripheral_Initialization(void)
{
   cy_stc_gpio_pin_config_t    stc_port_pin_cfg_uart = {0};
   :
```

Contains port configurations

Code Listing 18 Example for Initializing GPIO for SCB RX/ TX in Configuration Part

```
    /*-----------------------------*/
    /* Port Configuration for UART */
    /*-----------------------------*/

    /* P13.0 -> scb[3].uart_rx */
    stc_port_pin_cfg_uart.driveMode = CY_GPIO_DM_HIGHZ;
    stc_port_pin_cfg_uart.hsiom     = CY_USB_SCB_UART_RX_MUX;
    Cy_GPIO_Pin_Init(CY_USB_SCB_UART_RX_PORT, CY_USB_SCB_UART_RX_PIN, &stc_port_pin_cfg_uart);

    /* P13.1 -> scb[3].uart_tx */
    stc_port_pin_cfg_uart.driveMode = CY_GPIO_DM_STRONG_IN_OFF;
    stc_port_pin_cfg_uart.hsiom     = CY_USB_SCB_UART_TX_MUX;
    Cy_GPIO_Pin_Init(CY_USB_SCB_UART_TX_PORT, CY_USB_SCB_UART_TX_PIN, &stc_port_pin_cfg_uart);
    :
}
```

Set drive mode for P13.0

Set HSIOM to SCB UART RX

Set drive mode for P13.1

Set HSIOM to SCB UART TX

See Initializing GPIO

3.5.2 TCPWM Port Configuration

This example demonstrates the configuration of a port for TCPWM output in CYT2B7 series. In this example, TCPWM ch0 port (P6.1) is used, and it is assigned to ACT_0 of HSIOM. Configure the TCPWM function after setting the port.

For details on selecting an appropriate port for each function, see the Alternate Function Pin Assignments section in the **datasheet**. For details on TCPWM settings, see AN220224 listed in **Related Documents**.

Figure 8 shows the signal path as a TCPWM port on P6.1. To use PWM ch0, configure HSIOM to PWM_0.

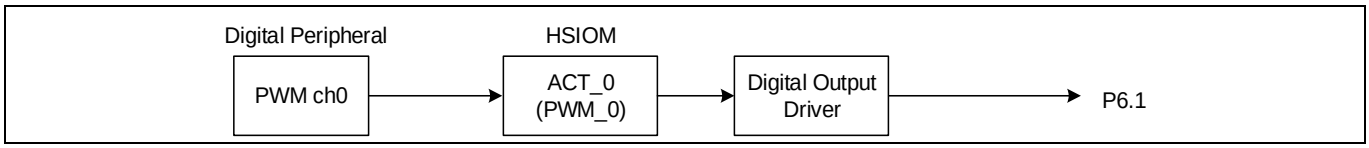

Figure 8 TCPWM Port Configuration

Example configuration:

- Port number: P6.1
- Initial state: Low
- Input/Output: Output
- Drive mode: Strong
- Functionality configuration (HSIOM): TCPWM
- Interrupt function: Unused

GPIO Settings

Table 8 lists the parameters of the configuration part in SDL for TCPWM port.

Table 8 List of GPIO Pin Parameters of TCPWM Port

Parameters	Description	Value
.outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
.driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down 8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives Low. Input buffer ON. 12: Open Drain, Drives High. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	CY_GPIO_DM_STRONG_IN_OFF = 6ul
.hsiom	Sets the connection for IO pin 0 route.	P6_1_TCPWM0_LINE0 See point 2 of the SDL description.
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	0ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	0ul
.vtrip	Select the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Select slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

The following description will help you understand the configuration of GPIO port, pin, and HSIOM of this example in SDL:

1. Each port number is defined in the device header in the *header* folder.

   For example, if the revision of the MCU silicon is Revision B, the device header is in *hdr/rev_b.*

   ```
   #define GPIO_PRT6       ((volatile stc_GPIO_PRT_t*) &GPIO->PRT[6]
   ```

   The port number and pin number are defined together in the device GPIO header in the same folder.

   For example, if the MCU is 176-pin CYT2B7 series, you can find the following definition in *gpio_cyt2b7_176_lqfp.h.*

   ```
   #define P6_1_PORT      GPIO_PRT6
   #define P6_1_PIN       1u
   ```

2. Each GPIO pin has its dedicated HSIOM selection. See the device GPIO header for the HSIOM of the specific pin and its parameter value. Set the HSIOM of the pin according to the device series.

   For 176-pin CYT2B7 series MCU, the HSIOM connections for P6.1 are as follows:

   ```
   /* P6.1 */
   P6_1_GPIO                       =  0,    /* GPIO controls 'out' */
   P6_1_AMUXA                      =  4,    /* Analog mux bus A */
   P6_1_AMUXB                      =  5,    /* Analog mux bus B */
   P6_1_AMUXA_DSI                  =  6,
   /* Analog mux bus A, DSI control */
   P6_1_AMUXB_DSI                  =  7,
   /* Analog mux bus B, DSI control */
   P6_1_TCPWM0_LINE0               =  8,
   /* Digital Active - tcpwm[0].line[0]:0 */
   P6_1_TCPWM0_LINE_COMPL256       =  9,
   /* Digital Active - tcpwm[0].line_compl[256]:0 */
   P6_1_TCPWM0_TR_ONE_CNT_IN0      = 10,
   /* Digital Active - tcpwm[0].tr_one_cnt_in[0]:0 */
   P6_1_TCPWM0_TR_ONE_CNT_IN769    = 11,
   /* Digital Active - tcpwm[0].tr_one_cnt_in[769]:0 */
   P6_1_SCB4_UART_TX               = 17,
   /* Digital Active - scb[4].uart_tx:0 */
   P6_1_SCB4_I2C_SDA               = 18,
   /* Digital Active - scb[4].i2c_sda:0 */
   P6_1_SCB4_SPI_MOSI              = 19,
   /* Digital Active - scb[4].spi_mosi:0 */
   P6_1_LIN0_LIN_TX3               = 20,
   /* Digital Active - lin[0].lin_tx[3]:0 */
   ```

GPIO Settings

Figure 9 shows the pin assignment with alternate functions of P6.1. “LINE0” in P6_1_TCPWM0_LINE0 represents “PWM_0” which indicates TCPWM counter number 0.

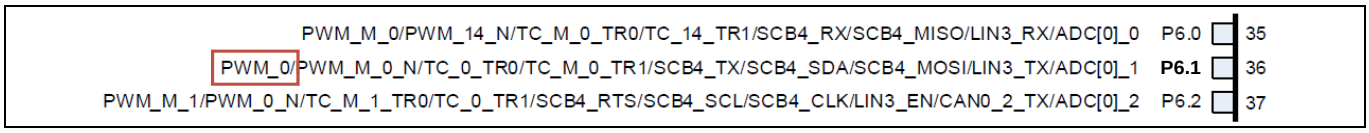

Figure 9 P6.1 Pin Assignment with Alternate Functions in CYT2B7 Series

See the device datasheet for the specific HSIOM functional connections for each pin.

Code Listing 19 demonstrates an example program to initialize GPIO for TCPWM in the configuration part. The example program for the driver part is the same as Code Listing 1.

Code Listing 19 Example for Initializing GPIO for TCPWM in Configuration Part

```
/* TCPWM_TR_ONE_CNT_IN0 */
#define TCPWM_LINEx_PORT                GPIO_PRT6
#define TCPWM_LINEx_PIN                 1u
#define TCPWM_LINEx_MUX                 P6_1_TCPWM0_LINE0

cy_stc_gpio_pin_config_t pin_cfg1 =
{
    .outVal    = 0ul,       /* Pin output state */
    .driveMode = CY_GPIO_DM_STRONG_IN_OFF, /* Drive mode */
    .hsiom     = TCPWM_LINEx_MUX,         /* HSIOM selection */
    .intEdge   = 0ul,       /* Interrupt Edge type */
    .intMask   = 0ul,       /* Interrupt enable mask */
    .vtrip     = 0ul,       /* Input buffer voltage trip type */
    .slewRate  = 0ul,       /* Output buffer slew rate */
    .driveSel  = 0ul,       /* Drive strength */
};

int main(void)
{
   :
   /*-------------------------------*/
   /* Port Configuration for TCPWM */
   /*-------------------------------*/
   Cy_GPIO_Pin_Init(TCPWM_LINEx_PORT, TCPWM_LINEx_PIN, &pin_cfg1);
   :
}
```

3.5.3 Analog Port Configuration

This example demonstrates the configuration of an analog port in CYT4BF series. In this example, ADC[0]_0 is assigned to an analog input. The GPIO drive mode of the port is set to High-Z and the input buffer is disabled to avoid crowbar current. For details on SAR ADC setting, see AN219755 listed in **Related Documents**.

Example configuration:

- Port number: P12.6
- Initial state: Low
- Input/Output: Input
- Drive mode: High impedance
- Functionality configuration (HSIOM): GPIO
- Interrupt function: Unused

GPIO Settings

Table 9 lists the parameters of the configuration part in SDL for analog port.

Table 9 List of GPIO Pin Parameters

Parameters	Description	Value
.outVal	Selects pin output state. 0: Output state not affected, 1: Output state set to '0'	0ul
.driveMode	Selects GPIO drive mode for IO pin. 0: Analog High Impedance 1: Reserved and should not be used 2: Resistive pull-up 3: Resistive pull-down 4: Open drain, drives LOW 5: Open drain, drives HIGH 6: Strong drive 7: Resistive pull-up/down 8: Digital High-Z. Input buffer ON. 9: Resistive Pull-Up. Input buffer ON. 10: Resistive Pull-Down. Input buffer ON. 11: Open Drain, Drives Low. Input buffer ON. 12: Open Drain, Drives High. Input buffer ON. 13: Strong Drive. Input buffer ON. 14: Resistive Pull-Up/Down. Input buffer ON.	CY_GPIO_DM_ANALOG = 0ul
.hsiom	Sets connection for IO pin 0 route.	P12_6_GPIO (Refer to the second point of SDL descriptions below.)
.intEdge	Sets the edge which will trigger an IRQ for IO pin 0. 0: Disabled, 1: Rising edge, 2: Falling edge, 3: Both	0ul
.intMask	Masks edge interrupt on IO pin. 0: Pin interrupt forwarding disabled 1: Pin interrupt forwarding enabled	0ul
.vtrip	Selects the pin 0 input buffer mode. 0: CMOS, 1: TTL	0ul
.slewRate	Selects slew rate for IO pin. 0: Fast slew rate, 1: Slow slew rate	0ul
.driveSel	Sets the GPIO drive strength for IO pin. 0: Full drive strength 1: Full drive strength 2: 1/2 drive strength 3: 1/4 drive strength	0ul

GPIO Settings

The following description will help you understand the configuration of GPIO port, pin, and HSIOM of this example in SDL:

1. Each port number is defined in the device header is in the header folder.

   For example, if the revision of the MCU silicon is Revision D, the device header is in *hdr/rev_d*.

   ```
   #define GPIO_PRT12      ((volatile stc_GPIO_PRT_t*) &GPIO->PRT[12]
   ```

   The port number and pin number are defined together in the project-specific header in the same folder.

   For example, if the MCU is CYT4BF series, you can find the following definition in *bb_bsp_tviibh8m.h*.

   ```
   #define CY_ADC_POT_PORT                 GPIO_PRT12
   #define CY_ADC_POT_PIN                  6
   #define CY_ADC_POT_PIN_MUX              P12_6_GPIO
   ```

2. The HSIOM of the specific pin and its parameter value can be found in the device GPIO header.

   For example, if the MCU is 176-pin CYT4BF series, the header file is *gpio_cyt4bf_176_teqfp.h*.

   For 176-pin CYT4BF series MCU, the HSIOM connections for P12.6 are as follows:

   ```
   /* P12.6 */
   P12_6_GPIO                      =  0,    /* N/A */
   P12_6_AMUXA                     =  4,    /* AMUXBUS A */
   P12_6_AMUXB                     =  5,    /* AMUXBUS B */
   P12_6_AMUXA_DSI                 =  6,    /* N/A */
   P12_6_AMUXB_DSI                 =  7,    /* N/A */
   P12_6_TCPWM1_LINE42             =  8,
   /* Digital Active - tcpwm[1].line[42]:0 */
   P12_6_TCPWM1_LINE_COMPL41       =  9,
   /* Digital Active - tcpwm[1].line_compl[41]:0 */
   P12_6_TCPWM1_TR_ONE_CNT_IN126   = 10,
   /* Digital Active - tcpwm[1].tr_one_cnt_in[126]:0 */
   P12_6_TCPWM1_TR_ONE_CNT_IN124   = 11,
   /* Digital Active - tcpwm[1].tr_one_cnt_in[124]:0 */
   ```

   See the device datasheet for the specific HSIOM functional connections for each pin.

GPIO Settings

Code Listing 20 demonstrates an example program to initialize GPIO for analog port in the configuration part. The example program for the driver part is the same as **Code Listing 1**.

Code Listing 20 Example for Initializing GPIO for Analog Port in Configuration Part

```
int main(void)
{
    :
    /* ADC port setting (Note default port setting after reset is just fine) */
    {
       cy_stc_gpio_pin_config_t adcPinConfig =
       {
           .outVal    = 0ul,       /* Pin output state */
           .driveMode = CY_GPIO_DM_ANALOG, /* Drive mode */
           .hsiom     = CY_ADC_POT_PIN_MUX, /* HSIOM selection */
           .intEdge   = 0ul,       /* Interrupt Edge type */
           .intMask   = 0ul,       /* Interrupt enable mask */
           .vtrip     = 0ul,       /* Input buffer voltage trip type */
           .slewRate  = 0ul,       /* Output buffer slew rate */
           .driveSel  = 0ul,       /* Drive strength */
       };
       Cy_GPIO_Pin_Init(CY_ADC_POT_PORT, CY_ADC_POT_PIN, &adcPinConfig);
    }
:
}
```

4 Glossary

Table 10 Glossary

Terms	Description
ADC	Analog-to-digital converter. See the SAR ADC chapter of the **Architecture TRM** for details.
GPIO	General-purpose I/O
HSIOM	High-speed I/O matrix. See the High-Speed I/O Matrix section in the I/O System chapter of the **Architecture TRM** for details.
SCB	Serial Communications Block. See the Serial Communications Block (SCB) chapter of the **Architecture TRM** for details.
Slew rate control	The change of voltage per unit of time. See the Slew Rate Control” section in the I/O System chapter of the **Architecture TRM** for details.
TCPWM	Timer, Counter, and Pulse Width Modulator. See the Timer, Counter, and PWM chapter of the **Architecture TRM** for details.

5 Related Documents

The following are the Traveo II family series datasheets, Technical Reference Manuals and Application notes. Contact **Technical Support** to obtain these documents.

- Device datasheet
  - CYT2B7 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family
  - CYT2B9 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family
  - CYT4BF Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT4DN Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT3BB/4BB Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
- Body Controller Entry Family
  - Traveo™ II Automotive Body Controller Entry Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B7
  - Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B9
- Body Controller High Family
  - Traveo™ II Automotive Body Controller High Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT4BF
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT3BB/4BB
- Cluster 2D Family
  - Traveo™ II Automotive Cluster 2D Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Cluster 2D Registers Technical Reference Manual (TRM)
- Application Notes
  - AN219755 – Using a SAR ADC in Traveo II Family
  - AN219842 – How to Use Interrupt in Traveo II
  - AN220119 – How to Use Serial Communications Block (SCB) in Traveo II Family
  - AN220224 – How to Use TCPWM In Traveo II Family

Revision history

Document version	Date of release	Description of changes
**	2019-03-06	New application note.
*A	2020-02-14	Updated Associated Part Family as “Traveo™ II Family CYT2/CYT3/CYT4 Series”. Changed target part numbers from “CYT2B/CYT4B Series” to “CYT2/CYT4 Series” in all instances across the document. Added target part numbers “CYT3 Series” related information in all instances across the document.
*B	2020-06-11	Updated GPIO Settings: Added flowchart and example codes in all instances.
*C	2021-04-08	Updated to Infineon template.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-04-08
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
002-20193 Rev. *C

IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.