PSoC 6 Analog Sub-system Design Using ModusToolBox

Vasanth R S
Staff Applications Engineer
WW35

Objective

- By the end of this training, you will
  - Learn about the analog peripherals available in PSoC 6
  - Understand the basic functionality of each of the peripherals
  - Learn how to use each of them using ModusToolBox

- Hardware
  - PSoC 6 WiFi-BT Pioneer Kit

- Software
  - ModusToolBox 2.1

Contents

Analog World!!!

- “***Every piece of information in the world has been copied. Backed up. Except the human mind, the last analog device in a digital world***”
  – WestWorld

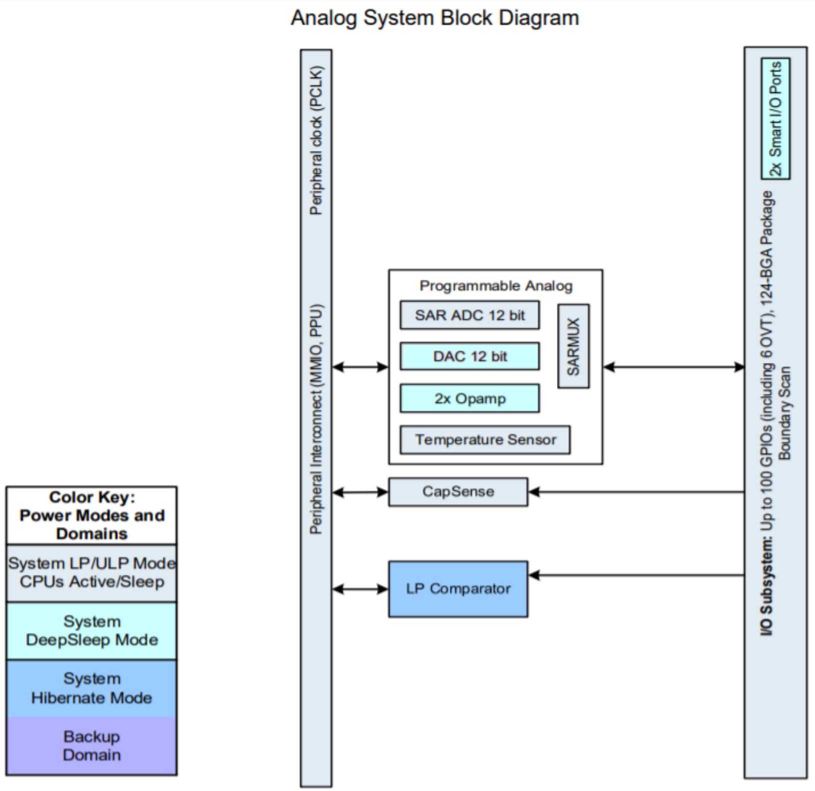
- World as we know is analog – we sense the world in its analog form.

- An analog interface to sense, measure the signals and work closely with digital system to interpret the signals

- The Internet Of Things(IoT) is all about massively gathering sensor information under strict power consumption constraints.

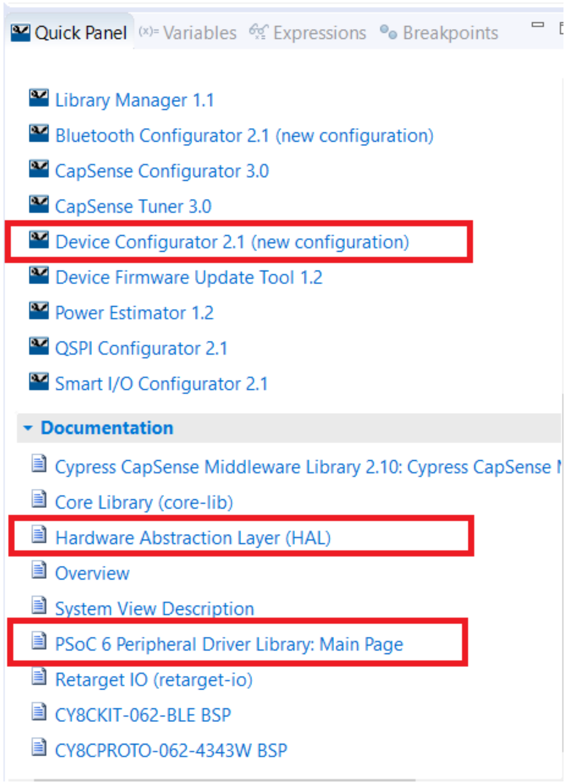
Programmable Analog Sub-System

- Consists of ,
  - 12 bit SAR ADC
  - 2 x Low-Power Comparators
  - 2 x Opamps
  - 12 bit DAC
  - Temperature sensor
  - Routing (SAR Mux, Analog Mux)


Analog System Block Diagram
Peripheral clock (PCLK)
Peripheral Interconnect (MMIO, PPU)
Programmable Analog
SAR ADC 12 bit
DAC 12 bit
2x Opamp
Temperature Sensor
SARMUX
CapSense
LP Comparator
I/O Subsystem: Up to 100 GPIOs (including 6 OVT), 124-BGA Package Boundary Scan
2x Smart I/O Ports
Color Key: Power Modes and Domains
System LP/ULP Mode CPUs Active/Sleep
System DeepSleep Mode
System Hibernate Mode
Backup Domain

Analog in ModusToolBox

- [HAL APIs]
  - Generic interface for multiple product families
  - Ease of use and portability
  - HAL Documentation
- [PDL APIs]
  - More granularity
  - PDL Documentation
- Device Configurator
  - Similar to Component Configurator
  - GUI for easy configuration



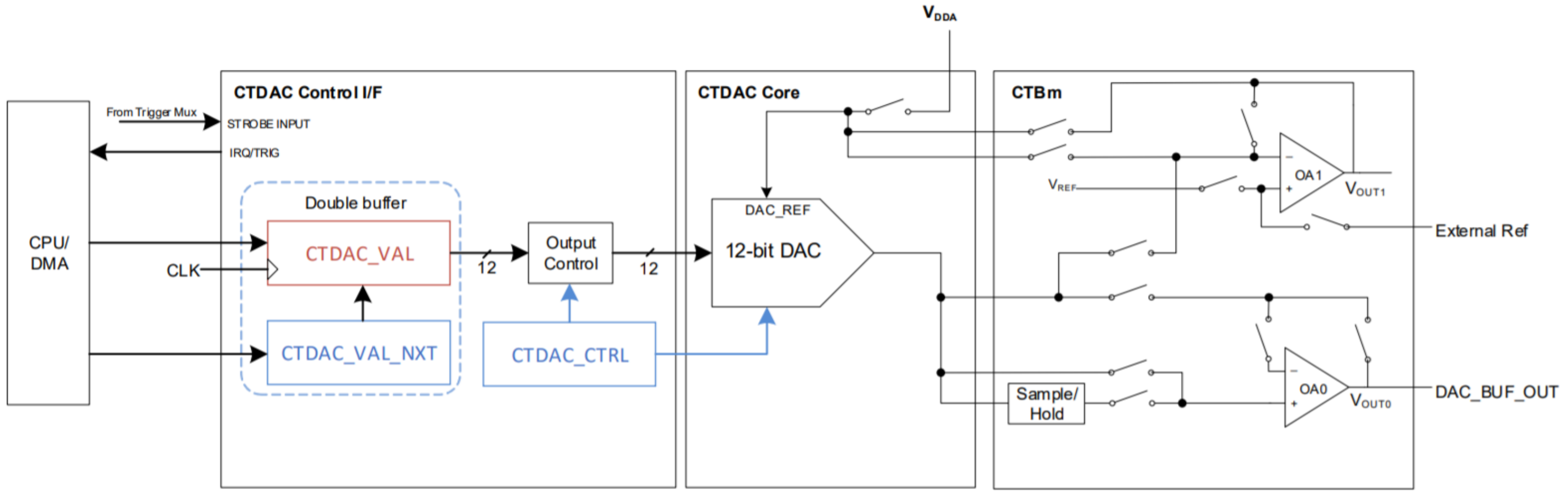
Library Manager 1.1
Bluetooth Configurator 2.1 (new configuration)
CapSense Configurator 3.0
CapSense Tuner 3.0
Device Configurator 2.1 (new configuration)
Device Firmware Update Tool 1.2
Power Estimator 1.2
QSPI Configurator 2.1
Smart I/O Configurator 2.1

Documentation

Cypress CapSense Middleware Library 2.10: Cypress CapSense
Core Library (core-lib)
Hardware Abstraction Layer (HAL)
Overview
System View Description
PSoC 6 Peripheral Driver Library: Main Page
Retarget IO (retarget-io)
CY8CKIT-062-BLE BSP
CY8CPROTO-062-4343W BSP

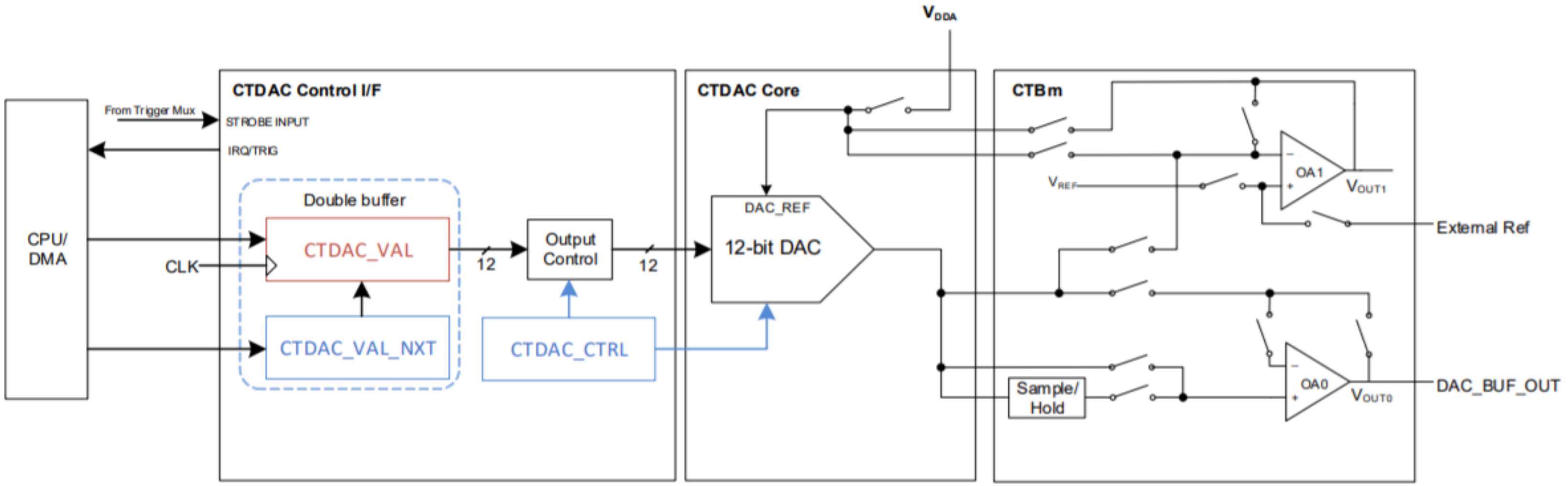
Digital To Analog Converter (CTDAC)

- 12 bit Continuous DAC
- 2us settling time(25pf load, buffered)
- Deep-Sleep Operation
- Selectable voltage reference
- Selectable Output Paths



Digital To Analog Converter (CTDAC)

- Selectable input modes
- Configurable update rate
- Double buffered DAC register
- Interrupt and DMA trigger on DAC buffer empty
- Configurable as PGA along with Opamp1 of CTB



Using CTDAC in your application - HAL

- To configure and start the DAC
  - `cyhal_dac_init(cyhal_dac_t * obj, cyhal_gpio_t pin);`

- To set the voltage reference
  - `cyhal_dac_set_reference(cyhal_dac_t * obj,cyhal_dac_ref_t ref);`

- To set DAC value
  - `cyhal_dac_write(const cyhal_dac_t * obj, uint16_t value );`
  - `cyhal_dac_write_mv(const cyhal_dac_t * obj,uint16_t value);`

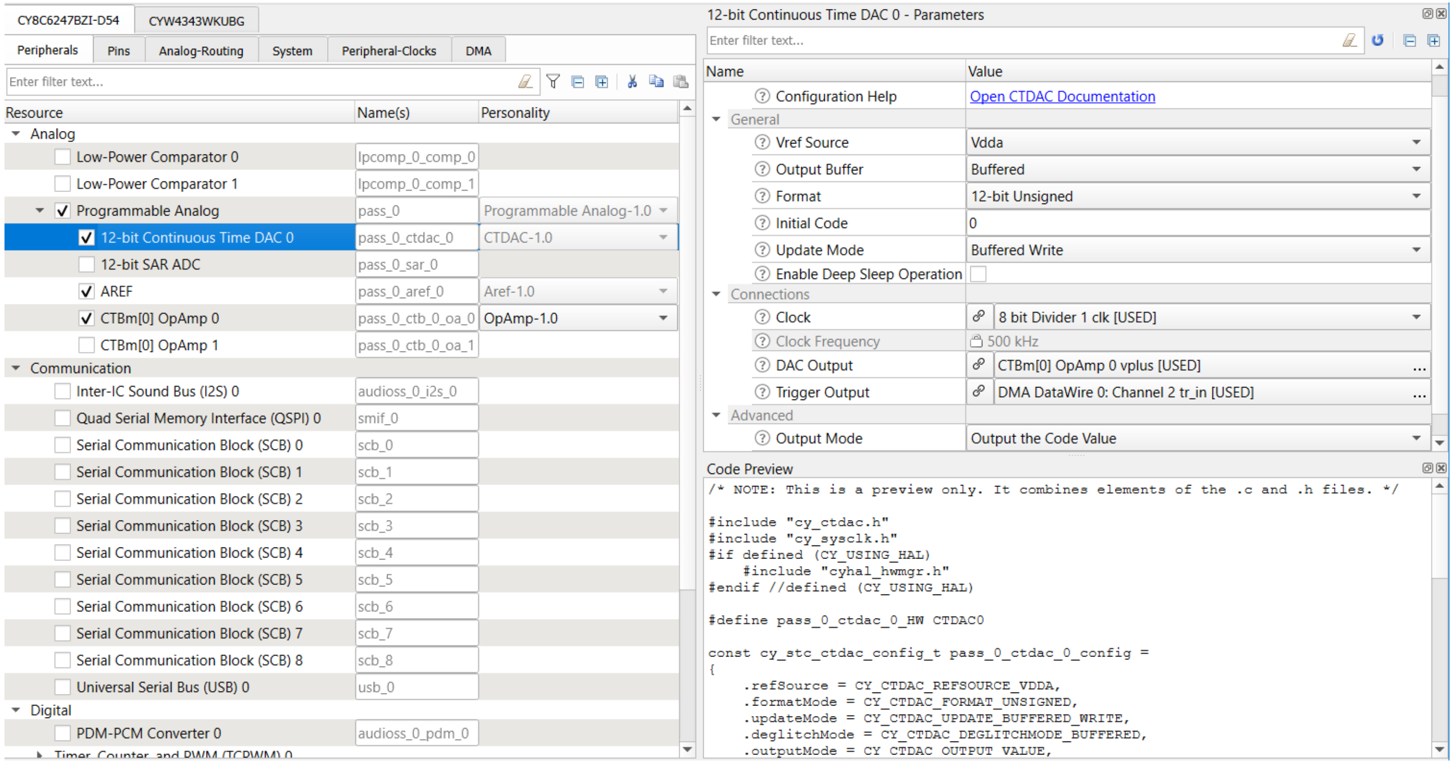
Using CTDAC in your application - PDL

- Initialize DAC
  - `Cy_CTDAC_Init(CTDAC_Type *base, const cy_stc_ctdac_config_t *config);`

- Enable the DAC
  - `Cy_CTDAC_Enable(CTDAC_Type *base);`

- Set DAC value
  - `Cy_CTDAC_SetValue(CTDAC_Type * base, int32_t value );`

Using CTDAC – Device Configurator



CY8C6247BZI-D54 | CYW4343WKUBG

Peripherals | Pins | Analog-Routing | System | Peripheral-Clocks | DMA

Enter filter text...

Resource	Name(s)	Personality
Analog		
☐ Low-Power Comparator 0	lpcomp_0_comp_0	
☐ Low-Power Comparator 1	lpcomp_0_comp_1	
✓ Programmable Analog	pass_0	Programmable Analog-1.0
✓ 12-bit Continuous Time DAC 0	pass_0_ctdac_0	CTDAC-1.0
☐ 12-bit SAR ADC	pass_0_sar_0	
✓ AREF	pass_0_aref_0	Aref-1.0
✓ CTBm[0] OpAmp 0	pass_0_ctb_0_oa_0	OpAmp-1.0
☐ CTBm[0] OpAmp 1	pass_0_ctb_0_oa_1	
Communication		
☐ Inter-IC Sound Bus (I2S) 0	audioss_0_i2s_0	
☐ Quad Serial Memory Interface (QSPI) 0	smif_0	
☐ Serial Communication Block (SCB) 0	scb_0	
☐ Serial Communication Block (SCB) 1	scb_1	
☐ Serial Communication Block (SCB) 2	scb_2	
☐ Serial Communication Block (SCB) 3	scb_3	
☐ Serial Communication Block (SCB) 4	scb_4	
☐ Serial Communication Block (SCB) 5	scb_5	
☐ Serial Communication Block (SCB) 6	scb_6	
☐ Serial Communication Block (SCB) 7	scb_7	
☐ Serial Communication Block (SCB) 8	scb_8	
☐ Universal Serial Bus (USB) 0	usb_0	
Digital		
☐ PDM-PCM Converter 0	audioss_0_pdm_0	
Timer, Counter, and PWM (TCPWM) 0		

12-bit Continuous Time DAC 0 - Parameters

Enter filter text...

Name	Value
Configuration Help	Open CTDAC Documentation
General	
Vref Source	Vdda
Output Buffer	Buffered
Format	12-bit Unsigned
Initial Code	0
Update Mode	Buffered Write
Enable Deep Sleep Operation	☐
Connections	
Clock	8 bit Divider 1 clk [USED]
Clock Frequency	500 kHz
DAC Output	CTBm[0] OpAmp 0 vplus [USED]
Trigger Output	DMA DataWire 0: Channel 2 tr_in [USED]
Advanced	
Output Mode	Output the Code Value

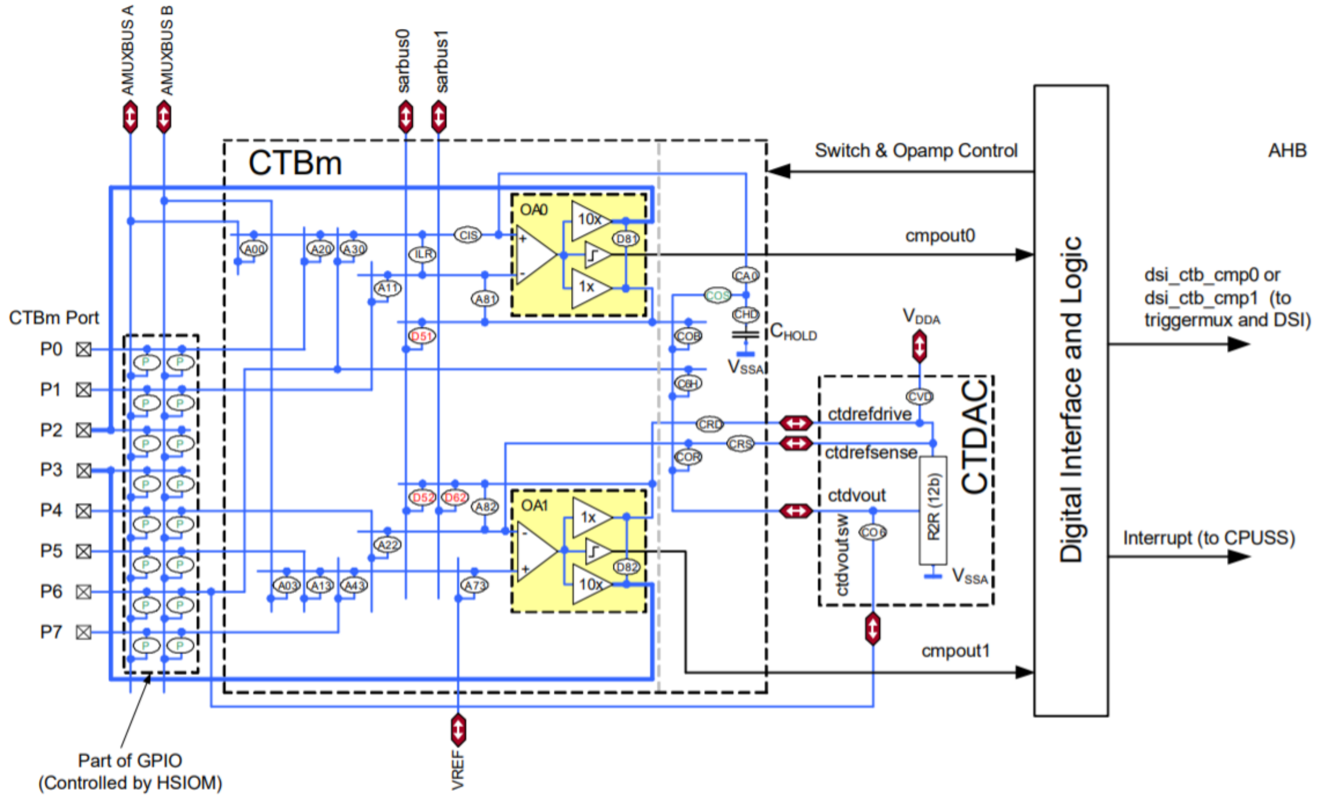
Code Preview

```
/* NOTE: This is a preview only. It combines elements of the .c and .h files. */

#include "cy_ctdac.h"
#include "cy_sysclk.h"
#if defined (CY_USING_HAL)
    #include "cyhal_hwmgr.h"
#endif //defined (CY_USING_HAL)

#define pass_0_ctdac_0_HW CTDAC0

const cy_stc_ctdac_config_t pass_0_ctdac_0_config =
{
    .refSource = CY_CTDAC_REFSOURCE_VDDA,
    .formatMode = CY_CTDAC_FORMAT_UNSIGNED,
    .updateMode = CY_CTDAC_UPDATE_BUFFERED_WRITE,
    .deglitchMode = CY_CTDAC_DEGLITCHMODE_BUFFERED,
    .outputMode = CY_CTDAC_OUTPUT_VALUE,
```

Continuous Time Block (CTBm)

- Two highly configurable Opamps
  - Configurable power and output drive strength
  - Can be configured as a voltage follower using internal routing
  - Can be configured as a comparator with optional 10 mV hysteresis
- Flexible input and output routing
- Works as a buffer or an amplifier for SAR ADC inputs
- Works as a buffer, amplifier, or sample and hold (SH) for the CTDAC output
- Can operate in Deep Sleep power mode

CTBm Architecture



Using CTBm Opamp in your application - HAL

- Initialize
  -
    ```
    cyhal_opamp_init(
        cyhal_opamp_t *  obj,
        cyhal_gpio_t     vin_p,
        cyhal_gpio_t     vin_m,
        cyhal_gpio_t     vout);
    ```

- Set power
  -
    ```
    cyhal_opamp_set_power(
        cyhal_opamp_t * obj,
        cyhal_power_level_t power);
    ```

Using CTBm Comparator in your application - HAL

› Configure Comparator
  – `cyhal_comp_config_t config = { .power = CYHAL_POWER_LEVEL_HIGH, .hysteresis = false };`

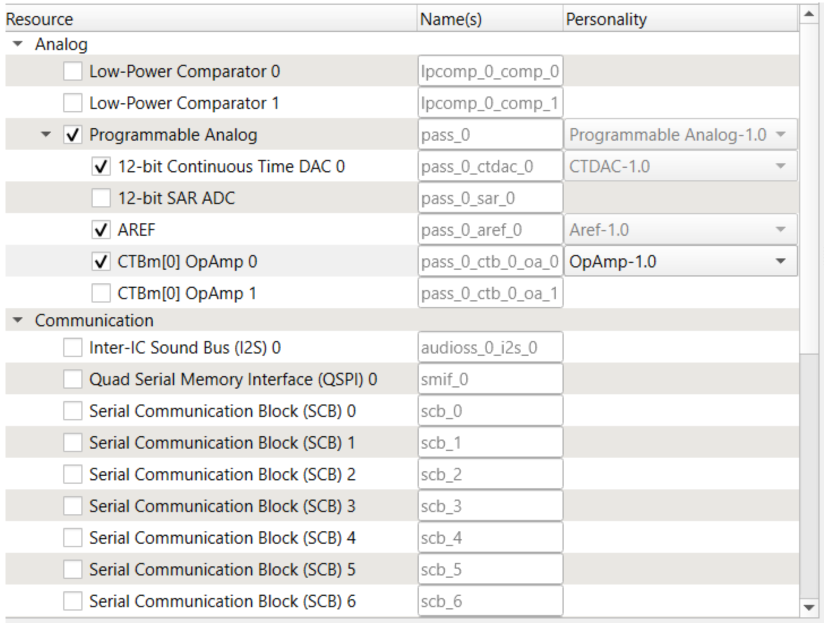
› Initialize comparator

```
cy_rslt_t cyhal_comp_init(cyhal_comp_t * obj,
cyhal_gpio_t         vin_p,
cyhal_gpio_t         vin_m,
cyhal_gpio_t         output,
cyhal_comp_config_t * cfg)
```

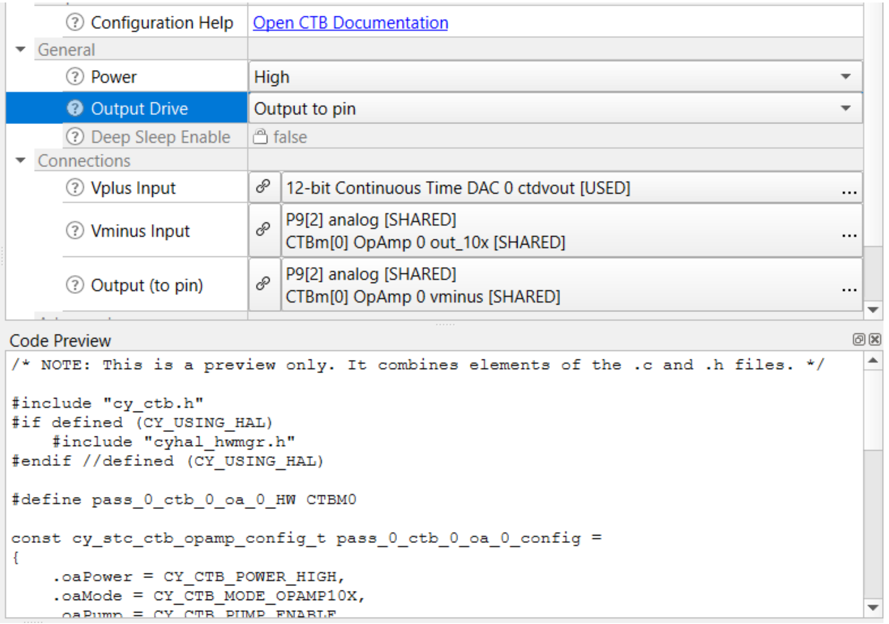
Using CTBm Opamp/Comparator in your application – PDL

› Using PDL APIs
  – Initialize the component
    – `cy_en_ctb_status_t Cy_CTB_Init(CTBM_Type *base, const cy_stc_ctb_config_t *config);`
    – `cy_en_ctb_status_t Cy_CTB_OpampInit(CTBM_Type * base, cy_en_ctb_opamp_sel_t opampNum, const cy_stc_ctb_opamp_config_t * config)`
  – Enable the component
    – `void Cy_CTB_Enable(CTBM_Type *base);`

Using CTB in your application - Opamp



Resource	Name(s)	Personality
Analog		
☐ Low-Power Comparator 0	lpcomp_0_comp_0	
☐ Low-Power Comparator 1	lpcomp_0_comp_1	
✓ Programmable Analog	pass_0	Programmable Analog-1.0
✓ 12-bit Continuous Time DAC 0	pass_0_ctdac_0	CTDAC-1.0
☐ 12-bit SAR ADC	pass_0_sar_0	
✓ AREF	pass_0_aref_0	Aref-1.0
✓ CTBm[0] OpAmp 0	pass_0_ctb_0_oa_0	OpAmp-1.0
☐ CTBm[0] OpAmp 1	pass_0_ctb_0_oa_1	
Communication		
☐ Inter-IC Sound Bus (I2S) 0	audioss_0_i2s_0	
☐ Quad Serial Memory Interface (QSPI) 0	smif_0	
☐ Serial Communication Block (SCB) 0	scb_0	
☐ Serial Communication Block (SCB) 1	scb_1	
☐ Serial Communication Block (SCB) 2	scb_2	
☐ Serial Communication Block (SCB) 3	scb_3	
☐ Serial Communication Block (SCB) 4	scb_4	
☐ Serial Communication Block (SCB) 5	scb_5	
☐ Serial Communication Block (SCB) 6	scb_6	

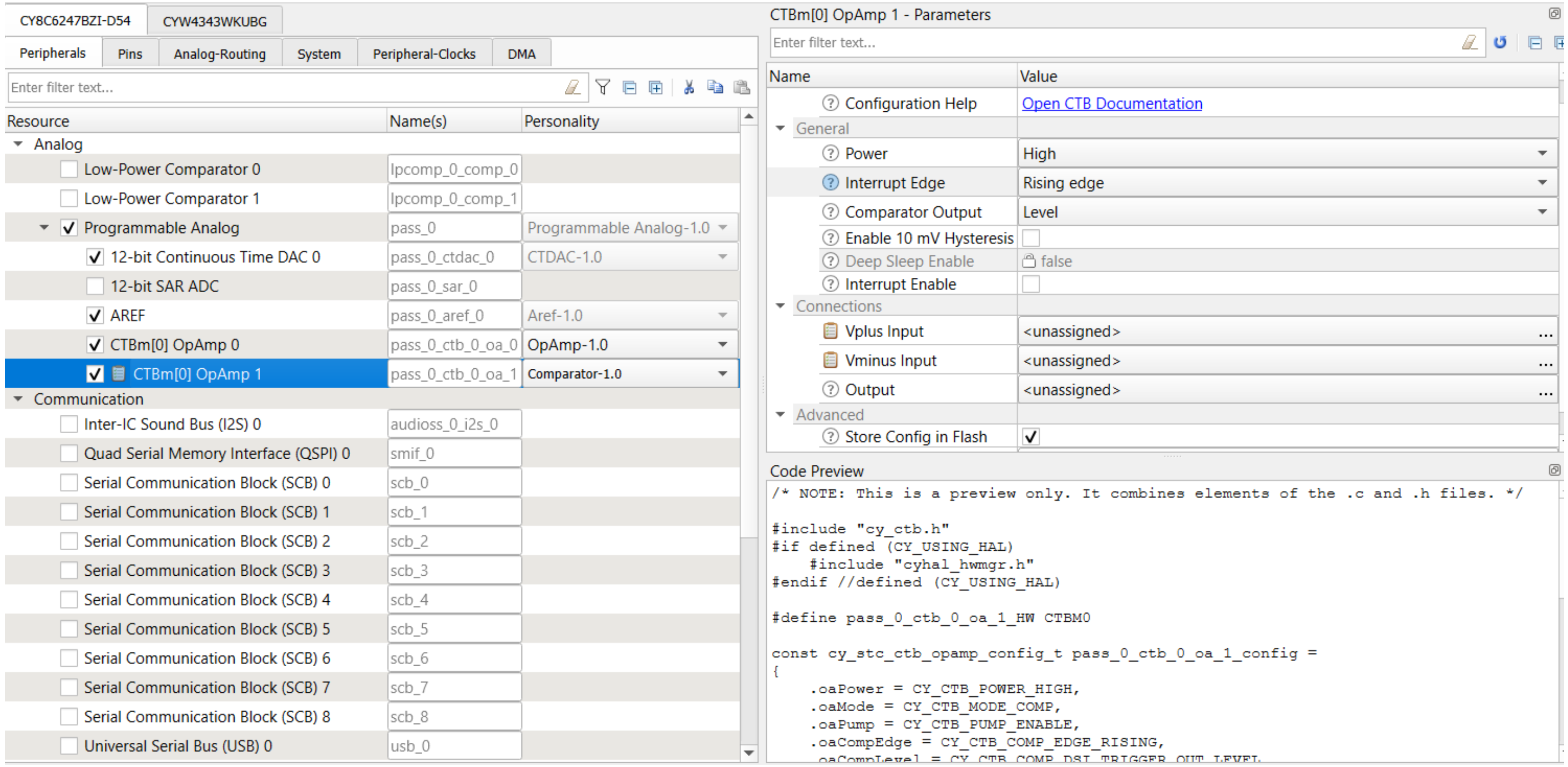


Configuration Help	Open CTB Documentation
General	
Power	High
Output Drive	Output to pin
Deep Sleep Enable	false
Connections	
Vplus Input	12-bit Continuous Time DAC 0 ctdvout [USED]
Vminus Input	P9[2] analog [SHARED] CTBm[0] OpAmp 0 out_10x [SHARED]
Output (to pin)	P9[2] analog [SHARED] CTBm[0] OpAmp 0 vminus [SHARED]

Code Preview

```
/* NOTE: This is a preview only. It combines elements of the .c and .h files. */

#include "cy_ctb.h"
#if defined (CY_USING_HAL)
    #include "cyhal_hwmgr.h"
#endif //defined (CY_USING_HAL)

#define pass_0_ctb_0_oa_0_HW CTBM0

const cy_stc_ctb_opamp_config_t pass_0_ctb_0_oa_0_config =
{
    .oaPower = CY_CTB_POWER_HIGH,
    .oaMode = CY_CTB_MODE_OPAMP10X,
    .oaPump = CY_CTB_PUMP_ENABLE
```

Using CTB in your application - Comparator



CY8C6247BZI-D54 | CYW4343WKUBG

Peripherals | Pins | Analog-Routing | System | Peripheral-Clocks | DMA

Enter filter text...

Resource	Name(s)	Personality
Analog		
☐ Low-Power Comparator 0	lpcomp_0_comp_0	
☐ Low-Power Comparator 1	lpcomp_0_comp_1	
✓ Programmable Analog	pass_0	Programmable Analog-1.0
✓ 12-bit Continuous Time DAC 0	pass_0_ctdac_0	CTDAC-1.0
☐ 12-bit SAR ADC	pass_0_sar_0	
✓ AREF	pass_0_aref_0	Aref-1.0
✓ CTBm[0] OpAmp 0	pass_0_ctb_0_oa_0	OpAmp-1.0
✓ CTBm[0] OpAmp 1	pass_0_ctb_0_oa_1	Comparator-1.0
Communication		
☐ Inter-IC Sound Bus (I2S) 0	audioss_0_i2s_0	
☐ Quad Serial Memory Interface (QSPI) 0	smif_0	
☐ Serial Communication Block (SCB) 0	scb_0	
☐ Serial Communication Block (SCB) 1	scb_1	
☐ Serial Communication Block (SCB) 2	scb_2	
☐ Serial Communication Block (SCB) 3	scb_3	
☐ Serial Communication Block (SCB) 4	scb_4	
☐ Serial Communication Block (SCB) 5	scb_5	
☐ Serial Communication Block (SCB) 6	scb_6	
☐ Serial Communication Block (SCB) 7	scb_7	
☐ Serial Communication Block (SCB) 8	scb_8	
☐ Universal Serial Bus (USB) 0	usb_0	

CTBm[0] OpAmp 1 - Parameters

Enter filter text...

Name	Value
Configuration Help	Open CTB Documentation
General	
Power	High
Interrupt Edge	Rising edge
Comparator Output	Level
Enable 10 mV Hysteresis	☐
Deep Sleep Enable	false
Interrupt Enable	☐
Connections	
Vplus Input	<unassigned>
Vminus Input	<unassigned>
Output	<unassigned>
Advanced	
Store Config in Flash	✓

Code Preview

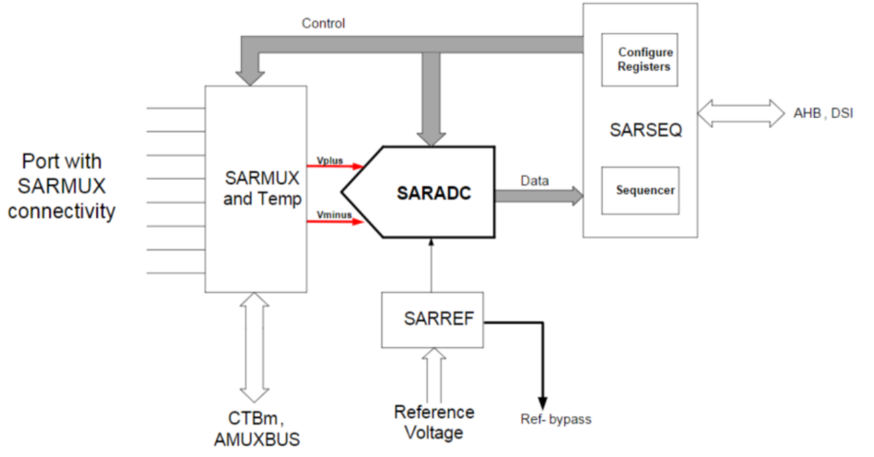
```
/* NOTE: This is a preview only. It combines elements of the .c and .h files. */

#include "cy_ctb.h"
#if defined (CY_USING_HAL)
    #include "cyhal_hwmgr.h"
#endif //defined (CY_USING_HAL)

#define pass_0_ctb_0_oa_1_HW CTBM0

const cy_stc_ctb_opamp_config_t pass_0_ctb_0_oa_1_config =
{
    .oaPower = CY_CTB_POWER_HIGH,
    .oaMode = CY_CTB_MODE_COMP,
    .oaPump = CY_CTB_PUMP_ENABLE,
    .oaCompEdge = CY_CTB_COMP_EDGE_RISING,
    .oaCompLevel = CY_CTB_COMP_DSI_TRIGGER_OUT_LEVEL,
```

Demo

- Demo 1: CTDAC Sinewave Generation
  - Using device configurator and PDL APIs
  - CTDAC and Opamp are used
  - Values stored in LUT transferred to CTDAC using DMA
  - Internal Bandgap reference is used as CTDAC source
  - Opamp used as reference buffer

Analog to Digital Converter

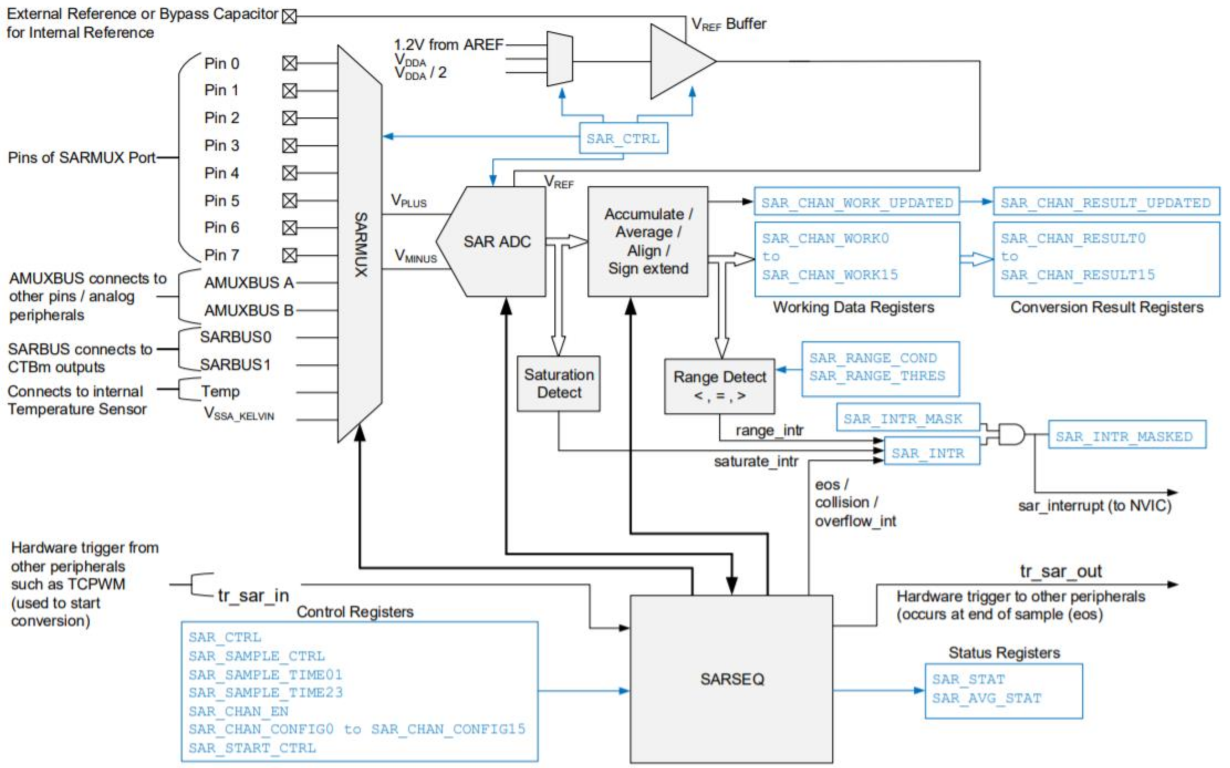
- SAR ADC Subsystem block
  - 12 bit SAR ADC converter
  - Embedded reference block
  - A Mux (SARMUX) at the input of the converter
  - A Sequence Controller (SARSEQ) which enables multi channel acquisition without CPU intervention



Analog to Digital Converter

- Maximum sample rate of 1Msps
- Sixteen individually configurable channels (depends on routing capabilities)
- Per channel selectable
  - Single ended or differential input mode
  - Input from external pin
  - One of four acquisition times
  - Averaging and accumulation
- Firmware/Hardware Trigger for Single Shot/ Continuous Mode
- Selectable voltage reference
- Interrupt Generation

Analog to Digital Converter



Using ADC in your application - HAL

- Configure ADC component
  - `cyhal_adc_init(`
    ```
    cyhal_adc_init(
    cyhal_adc_t * obj,
    cyhal_gpio_t pin,
    const cyhal_clock_t * clk)
    ```

- Configure ADC channel
  ```
  cyhal_adc_channel_init_diff  (cyhal_adc_channel_t * obj,
  cyhal_adc_t * adc,
  cyhal_gpio_t vplus,
  cyhal_gpio_t vminus,
  const cyhal_adc_channel_config_t * cfg )
  ```

- Read result
  - `int32_t cyhal_adc_read(const cyhal_adc_channel_t * obj)`

Using ADC in your application - PDL

- Initialize Component

  -
    ```
    cy_en_sar_status_t Cy_SAR_Init(
        SAR_Type * base,
        const cy_stc_sar_config_t * config)
    ```

- Enable Component

  -
    ```
    Cy_SAR_Enable(SAR_Type * base)
    ```

- Start Conversion

  -
    ```
    void Cy_SAR_StartConvert(
        SAR_Type * base,
        cy_en_sar_start_convert_sel_t startSelect )
    ```

Using ADC in your application - PDL

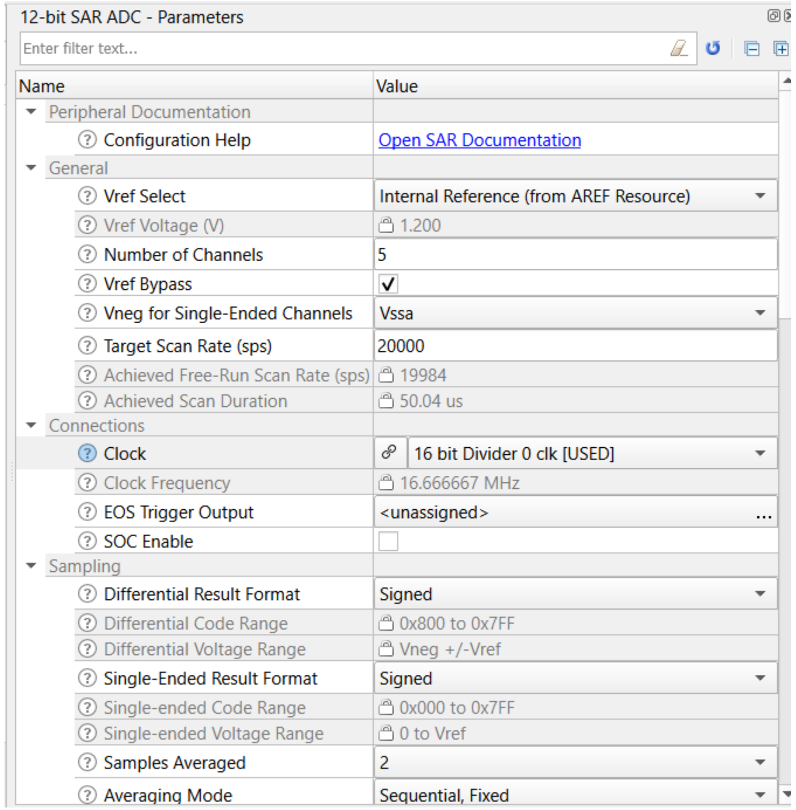
› Setting up interrupt

```
 const cy_stc_sysint_t ADC_IRQ_cfg = {
/* .intrSrc = */ pass_interrupt_sar_IRQn,/* Interrupt source is the SAR*/
 /* .intrPriority= */ 7UL    /* Interrupt priority is 7 */};

/* Configure the interrupt with vector at SAR_Interrupt(). */
(void)Cy_SysInt_Init(&ADC_IRQ_cfg, SAR_Interrupt);
/* Enable the interrupt. */
 NVIC_EnableIRQ(ADC_IRQ_cfg.intrSrc);
```

› Interrupt Status

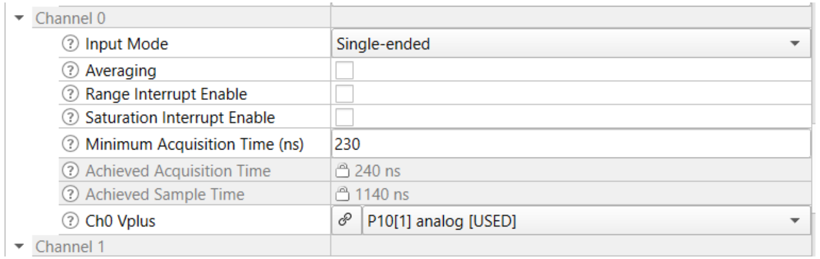
- `Cy_SAR_GetInterruptStatus(const SAR_Type * base)`
- `Cy_SAR_ClearInterrupt(SAR_Type * base,uint32_t intrMask);`

Using ADC in your application - Configurator



12-bit SAR ADC - Parameters

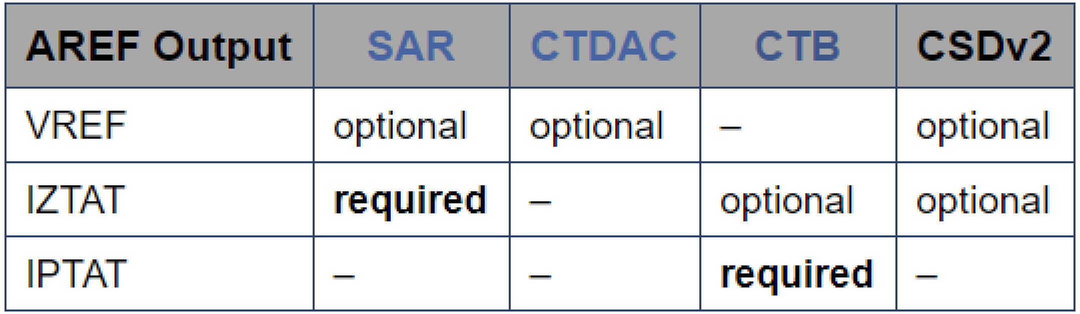
Name	Value
Peripheral Documentation	
Configuration Help	Open SAR Documentation
General	
Vref Select	Internal Reference (from AREF Resource)
Vref Voltage (V)	1.200
Number of Channels	5
Vref Bypass	✓
Vneg for Single-Ended Channels	Vssa
Target Scan Rate (sps)	20000
Achieved Free-Run Scan Rate (sps)	19984
Achieved Scan Duration	50.04 us
Connections	
Clock	16 bit Divider 0 clk [USED]
Clock Frequency	16.666667 MHz
EOS Trigger Output	<unassigned>
SOC Enable	
Sampling	
Differential Result Format	Signed
Differential Code Range	0x800 to 0x7FF
Differential Voltage Range	Vneg +/-Vref
Single-Ended Result Format	Signed
Single-ended Code Range	0x000 to 0x7FF
Single-ended Voltage Range	0 to Vref
Samples Averaged	2
Averaging Mode	Sequential, Fixed



Channel 0	
Input Mode	Single-ended
Averaging	
Range Interrupt Enable	
Saturation Interrupt Enable	
Minimum Acquisition Time (ns)	230
Achieved Acquisition Time	240 ns
Achieved Sample Time	1140 ns
Ch0 Vplus	P10[1] analog [USED]
Channel 1	

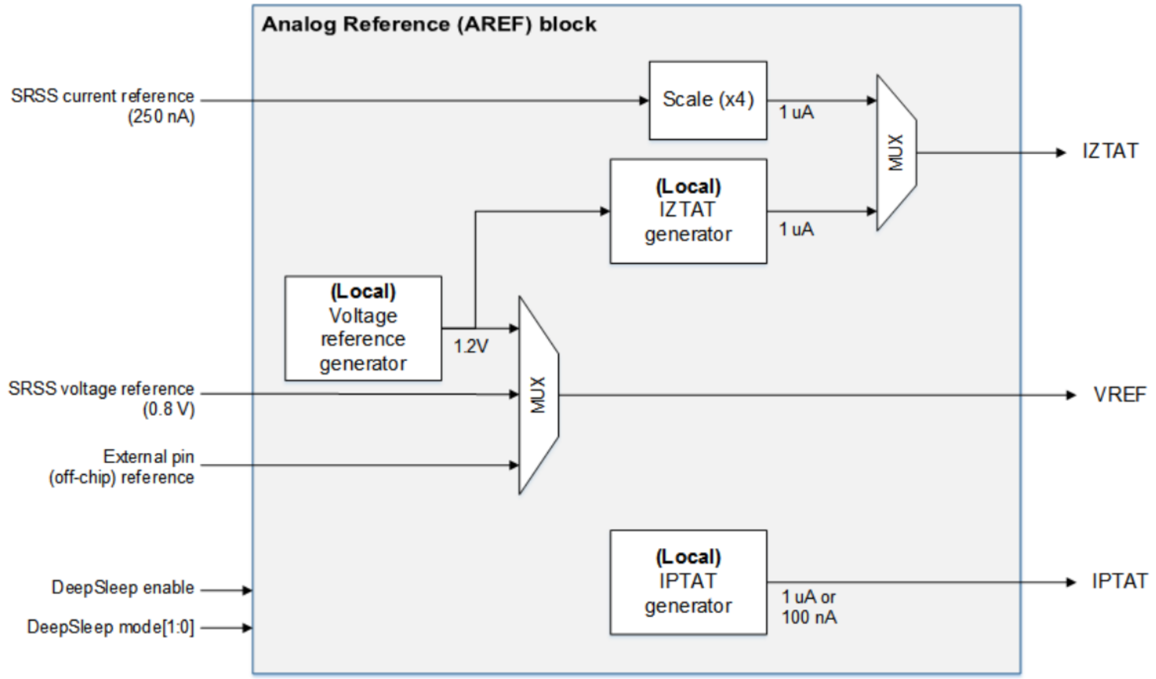
Analog Reference (AREF)

- Generates voltage reference Vref from one of the three sources
  - Local 1.2V reference (low noise, optimized for analog performance)
  - Reference from the SRSS(high noise, not recommended for analog)
  - An external pin
- Generates 1uA IZTAT independent of temperature variations(Local/SRSS)
- Generates IPTAT current reference
- Option to enable local reference in DeepSleep mode



AREF Output	SAR	CTDAC	CTB	CSDv2
VREF	optional	optional	–	optional
IZTAT	**required**	–	optional	optional
IPTAT	–	–	**required**	–

Analog Reference (AREF)



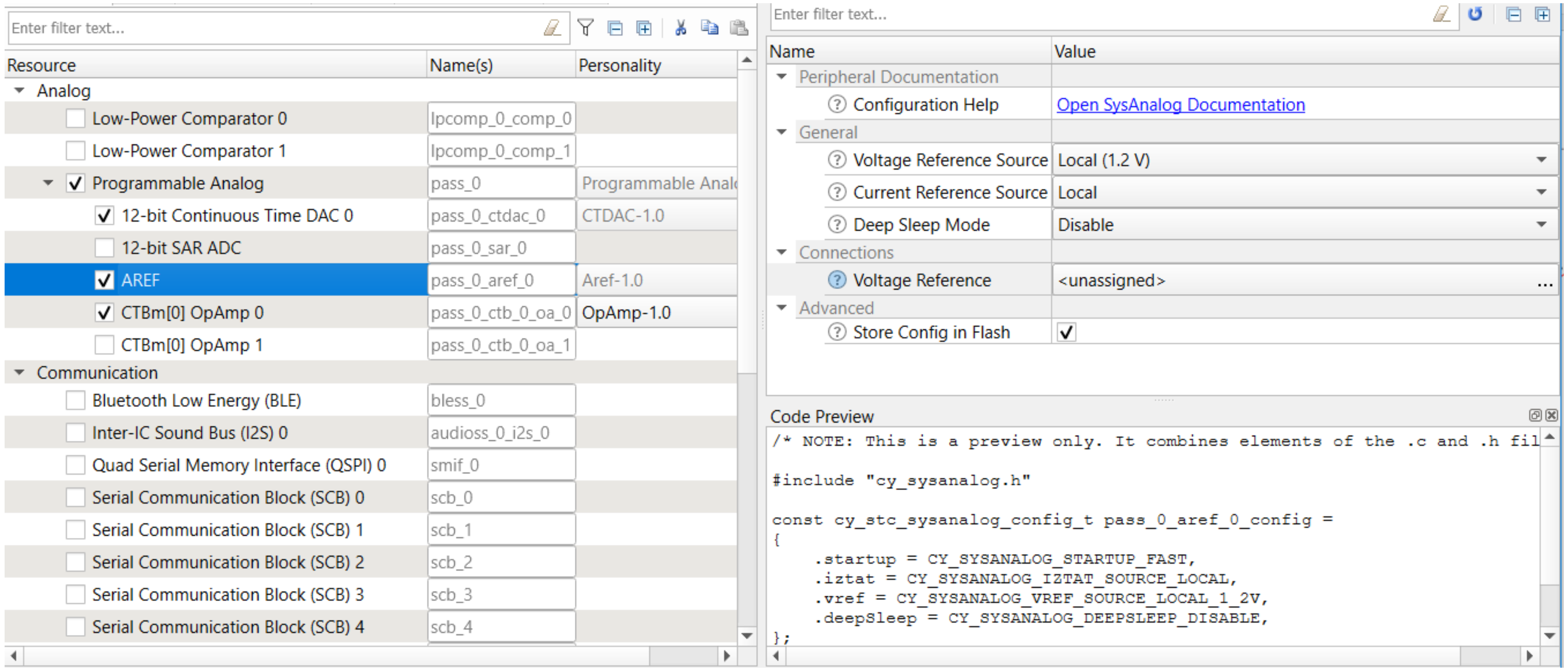
Analog Reference (AREF) block
SRSS current reference (250 nA)
Scale (x4)
1 uA
MUX
IZTAT
(Local) IZTAT generator
1 uA
(Local) Voltage reference generator
1.2V
MUX
SRSS voltage reference (0.8 V)
VREF
External pin (off-chip) reference
(Local) IPTAT generator
1 uA or 100 nA
IPTAT
DeepSleep enable
DeepSleep mode[1:0]

Using AREF in your application - PDL

- Initialize component
  - `cy_en_sysanalog_status_t Cy_SysAnalog_Init(const cy_stc_sysanalog_config_t * config)`

- Enable component
  - `void Cy_SysAnalog_Enable(void)`

Using AREF in your component – Device Configurator



Enter filter text...

Resource	Name(s)	Personality
Analog		
☐ Low-Power Comparator 0	lpcomp_0_comp_0	
☐ Low-Power Comparator 1	lpcomp_0_comp_1	
✓ Programmable Analog	pass_0	Programmable Analo
✓ 12-bit Continuous Time DAC 0	pass_0_ctdac_0	CTDAC-1.0
☐ 12-bit SAR ADC	pass_0_sar_0	
✓ AREF	pass_0_aref_0	Aref-1.0
✓ CTBm[0] OpAmp 0	pass_0_ctb_0_oa_0	OpAmp-1.0
☐ CTBm[0] OpAmp 1	pass_0_ctb_0_oa_1	
Communication		
☐ Bluetooth Low Energy (BLE)	bless_0	
☐ Inter-IC Sound Bus (I2S) 0	audioss_0_i2s_0	
☐ Quad Serial Memory Interface (QSPI) 0	smif_0	
☐ Serial Communication Block (SCB) 0	scb_0	
☐ Serial Communication Block (SCB) 1	scb_1	
☐ Serial Communication Block (SCB) 2	scb_2	
☐ Serial Communication Block (SCB) 3	scb_3	
☐ Serial Communication Block (SCB) 4	scb_4	

Enter filter text...

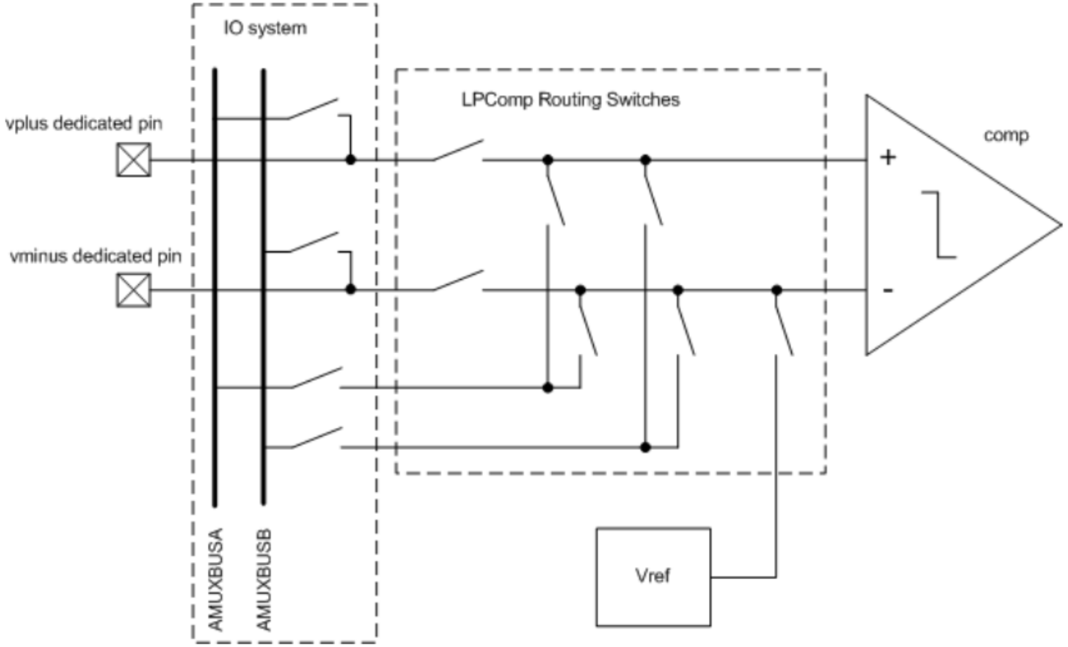
Name	Value
Peripheral Documentation	
Configuration Help	Open SysAnalog Documentation
General	
Voltage Reference Source	Local (1.2 V)
Current Reference Source	Local
Deep Sleep Mode	Disable
Connections	
Voltage Reference	<unassigned>
Advanced	
Store Config in Flash	✓

Code Preview

```
/* NOTE: This is a preview only. It combines elements of the .c and .h fil

#include "cy_sysanalog.h"

const cy_stc_sysanalog_config_t pass_0_aref_0_config =
{
    .startup = CY_SYSANALOG_STARTUP_FAST,
    .iztat = CY_SYSANALOG_IZTAT_SOURCE_LOCAL,
    .vref = CY_SYSANALOG_VREF_SOURCE_LOCAL_1_2V,
    .deepSleep = CY_SYSANALOG_DEEPSLEEP_DISABLE,
};
```

Low-Power Comparator

- Configurable input pins
- Configurable power and speed
- Ultra low-power mode support
- Hysteresis option
- Output Edge Detection
- Local reference voltage generation
- Wakeup source from low-power modes



Using LPComp in your application - HAL

- Configure Comparator
  - `cyhal_comp_config_t config = { .power = CYHAL_POWER_LEVEL_HIGH, .hysteresis = false };`

- Initialize comparator

```
cy_rslt_t cyhal_comp_init(cyhal_comp_t * obj,
cyhal_gpio_t         vin_p,
cyhal_gpio_t         vin_m,
cyhal_gpio_t         output,
cyhal_comp_config_t * cfg)
```

Using LPComp in your application - PDL

› Set the inputs

```
void Cy_LPComp_SetInputs(LPCOMP_Type * base,
cy_en_lpcomp_channel_t channel,
cy_en_lpcomp_inputs_t inputP,
cy_en_lpcomp_inputs_t inputN)
```

› LPComp configuration

```
Cy_LPComp_Init(LPCOMP_Type * base,
cy_en_lpcomp_channel_t       channel,
const cy_stc_lpcomp_config_t * config)
```

Using LPComp in your application - PDL

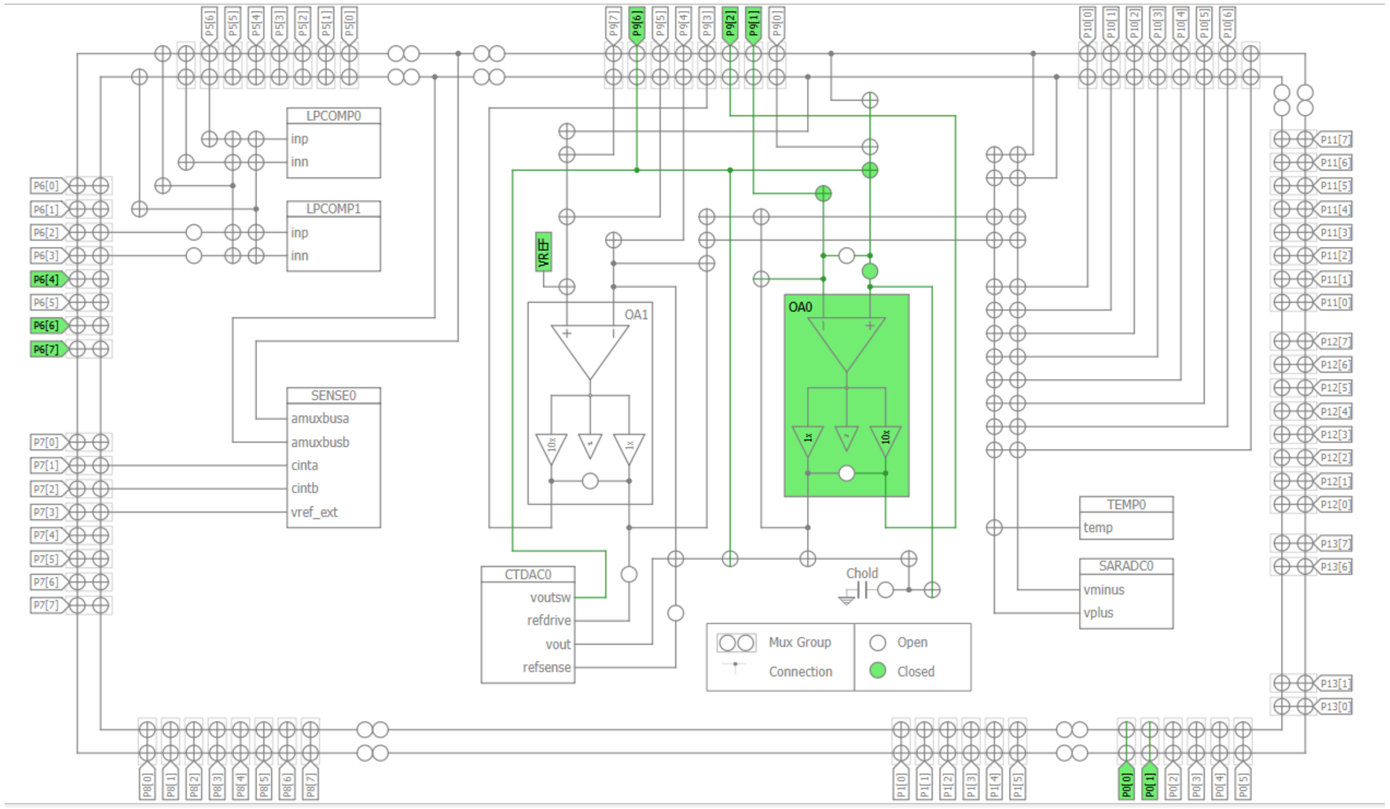
› Enable Local reference if needed

```
Cy_LPComp_ConnectULPReference(MYLPCOMP_HW, MYLPCOMP_CHANNEL);
 Cy_LPComp_UlpReferenceEnable(MYLPCOMP_HW);
```

› Enable the LPComp block

```
void Cy_LPComp_Enable
         (LPCOMP_Type * base,
         cy_en_lpcomp_channel_t channel);
```

Analog Routing



Switch Control Functions

- Switch control Functions in the PDL gives a set of API's to control the routing around an analog peripheral

- Peripheral specific APIs in controlling the switches around the peripheral

  -
    ```
    void Cy_CTDAC_SetAnalogSwitch(
        CTDAC_Type * base,
        uint32_t switchMask,
        cy_en_ctdac_switch_state_t state)
    ```

  -
    ```
    uint32_t Cy_SAR_GetAnalogSwitch(
        const SAR_Type * base,
        cy_en_sar_switch_register_sel_t switchSelect)
    ```

  -
    ```
    void Cy_CTB_OpenAllSwitches(CTBM_Type * base)
    ```

Demo

- Demo 2: ADC UART
  - Using HAL APIs
  - Post processing is done on data
  - Results are printed out through UART
- Demo 3: LPComp Hibernate
  - Using PDL APIs
  - Device is woken up from hibernate using LPComp

Overview

- Recap
  - Analog peripherals available in PSoC 6
  - Understood the basic functionality of each of the peripheral
  - Learned to use each block using ModusToolBox

- Resources
  - ModusToolBox User Guide
  - Cypress Github Landing Page
  - PSoC 6 Architecture TRM

Questions ?

https://community.cypress.com/welcome

Part of your life. Part of tomorrow.