



# Customer training workshop: Device Configurator ADC

TRAVEO™ T2G CYT4BF series Microcontroller Training  
V1.0.0 2023-07



## Scope of work

- This document helps application developers understand how to use the Device Configurator ADC as part of creating a ModusToolbox™ (MTB) application
  - The Device Configurator ADC is part of a collection of tools included with the MTB software. It shows all the analog resources, whether enabled or not, and how they connect.
- **ModusToolbox™ tools package version: 3.1.0**
- **Device Configurator version: 4.10**
- **Device**
  - The TRAVEO™ T2G CYT4BF8CE device is used in this code example.
- **Board**
  - The TRAVEO™ T2G KIT\_T2G-B-H\_LITE board is used for testing.

## Introduction – ADC features

- SAR ADC Core
  - 12-bit resolution with a maximum sample rate of 1 Msps
- 32 logical channels with the same capabilities
- Each logical channel can select input from:
  - 32 analog input pins
  - Diagnostic signals
  - Analog input pins of other ADC units
  - Support for external mux (three select bits)
  - AMUXBUSA/B
- Scans triggered by timer, software, continuous, pins, or system triggers
  - Multiple ADC units can be triggered by the same trigger to ensure lock-step operation
  - Triggers can be cleared by software
  - Optional debug pause
- Double buffering of output data
- Programmable sample time for each channel

## Introduction – ADC features (contd.)

- Programmable post processing options for each channel
  - Sign/zero extension to 16-bit
  - Left/right alignment
  - Averaging: First-order accumulate and dump, up to 256 samples
  - Programmable right shift
  - Range detection: below/above threshold, in/out-side range
  - Pulse detection: programmable positive and negative event counters
- Channels can be individual or grouped
  - Flexible grouping: from 32 groups with one channel to one group with 32 channels
- Group scans are dynamically scheduled by the hardware
  - Eight priorities, programmable per group
  - Four preemption types: resume, restart, cancel, or finish
  - Optional automatic idle power down

## Introduction – ADC features (contd.)

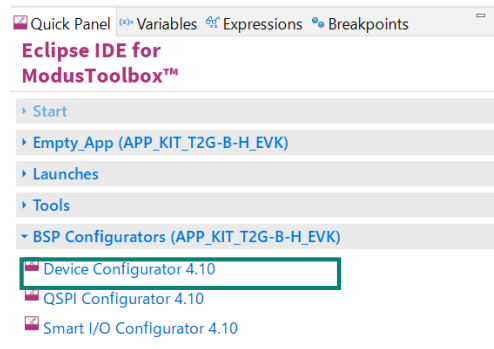
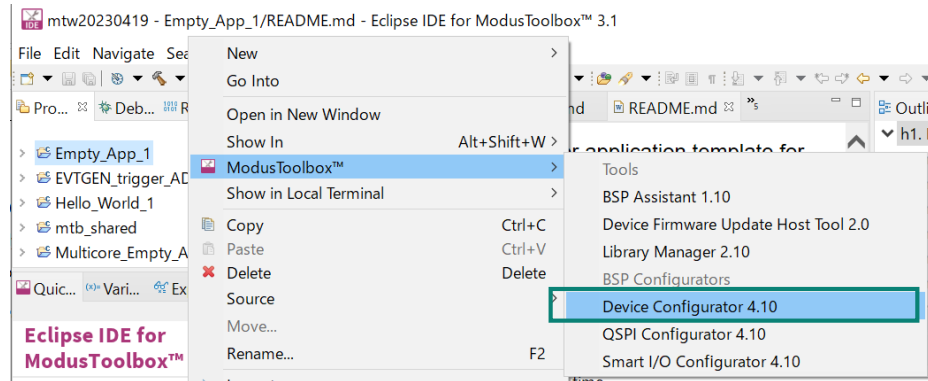
- Interrupt generation
  - Group scan done
  - Group scan done overflow detect
  - Group scan canceled
  - Per channel range detect
  - Per channel pulse detect
  - Per channel pulse/range overflow detect
- Output trigger generation per channel
  - Data ready/completion (each channel can trigger DW transfer)
  - Range violation detected
- Digital and analog calibration available
- Programmable offset and gain calibration
  - Non-intrusive background recalibration
  - Coherent calibration update

## Introduction – ADC features (contd.)

- Support for diagnostic measurements including broken wire detection. This includes:
  - ADC sampling capacitor preconditioning feature
  - Selectable current source or sink on selected ADC input while sampling
  - Support for LED diagnostics
- On-chip temperature sensor and power monitoring

## Launch the Device Configurator

- From Eclipse IDE
  - You can launch the Device Configurator by either of the following methods
- Right-click on the project in the Project Explorer and select ModusToolbox™ > Device Configurator <version>
  
- Click the Device Configurator link in the Quick Panel



# Device Configurator ADC config view

- Device Configurator ADC
  - On the Peripherals tab, you can select and configure each analog channel

The screenshot shows the Device Configurator interface for a CYT4BFP105 device. The 'Peripherals' tab is active, displaying an expandable tree of analog resources. A callout box explains that all available analog resources are shown in this tree and can be checked for enabling. The tree shows 'Programmable Analog' expanded to '12-bit SAR ADC 0', which is selected. Callouts point to the 'Name(s)' and 'Personality' columns in the resource list. A second callout explains that the 'Personality' column shows the selected personality for the resource. On the right, the 'Parameters' window for '12-bit SAR ADC 0 (ADC)' is open, showing various configuration options like 'Enable SAR Block', 'Clock Frequency', and 'Number Of Channels'. A callout points to this window, explaining that it is used to set parameters for the specified analog resource.

Peripherals tab

All available Analog are shown in an expandable tree. You can check the Analog channels that should be enabled setting.

This tab allows you to enter Names for the resource.

Personality shows the selected Personality, where applicable.

Parameter tab: You can set parameters for the specified Analog



## Quick start guide

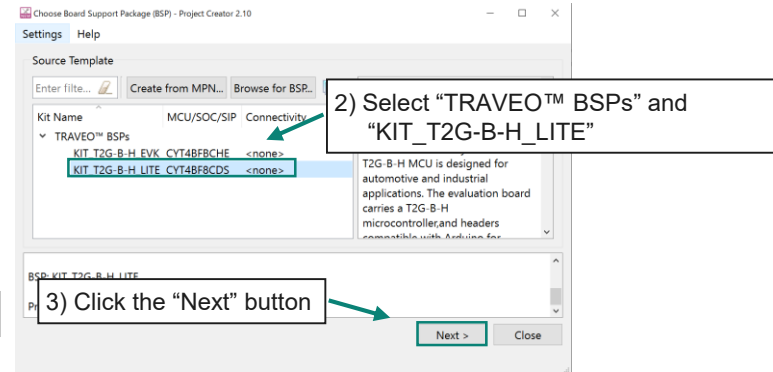
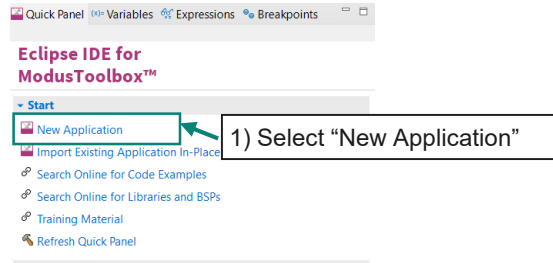
- To use the ADC Device Configurator for analog setting, do the following:
  - Launch the Device Configurator.
  - Check the analog channels to use
  - Select the parameter from the various pull-down menus to configure signals.
  - The ADC Device Configurator generates code into a "GeneratedSource" directory in your Eclipse IDE application, or in the same location you saved the \*.modus file for non-IDE applications. That directory contains the necessary source (.c) and header (.h) files for the generated firmware, which uses the relevant driver APIs to configure the hardware.
  - Use the generated structures as input parameters for ADC functions in your application.

## Use case

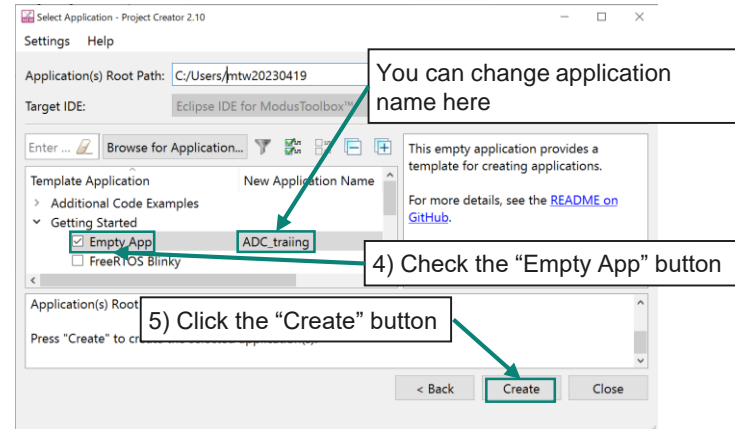
- This code example demonstrates how to use the TRAVEO™ T2G MCU event generator (EVTGEN) resource to trigger ADC conversion in the Active power mode
- Add “retarget-io” library using Library manager: A utility library to retarget the standard input/output (STDIO) messages to a UART port. You can directly print messages on a UART terminal using printf()
- The counter clock of the event generator block is configured to 1 MHz, and the active compare value is configured to 1000000 (1 sec)
- When the active counter is greater than or equal to the active compare value, it generates the active trigger event to trigger SAR ADC conversion and interrupt
- In the event generator interrupt handler, update the active comparator value corresponding comparator structure
- It generates events every second to trigger ADC conversions
- When SAR ADC conversion is complete, generate the ADC conversion “Group scan done” interrupt, and print the ADC result via UART
- See the EVTGEN\_trigger\_ADC application for operation

# ADC configuration

- Create project
- Click **New Application** in the Quick Panel and open the **Choose Board Support Package (BSP)** window

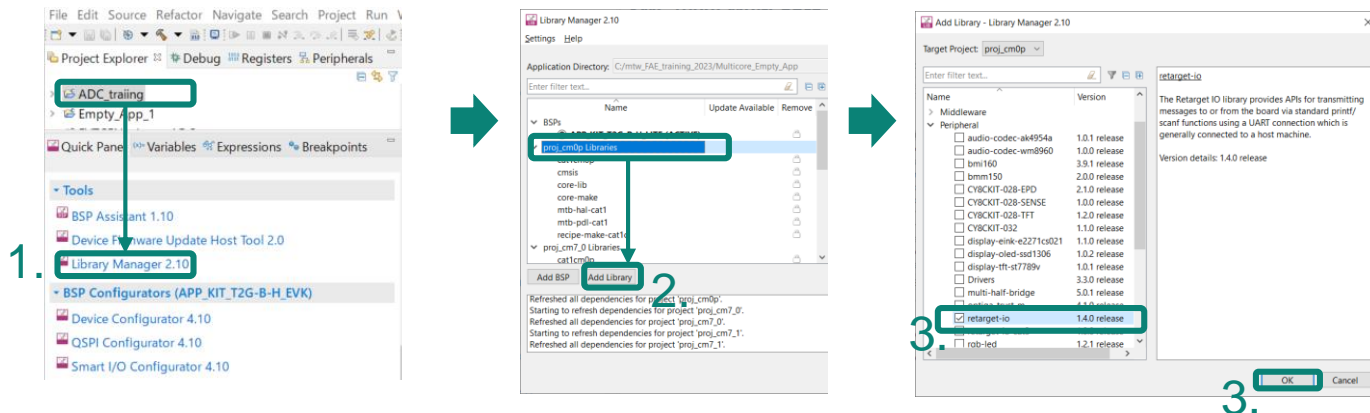


- Select TRAVEO™ BSPs and KIT\_T2G-B-H\_LITE
- Click **Next** and open the Application window
- Check the **Empty App** option.  
In this use case, change the application name to **ADC\_training**.
- Click **Create** to start application creation



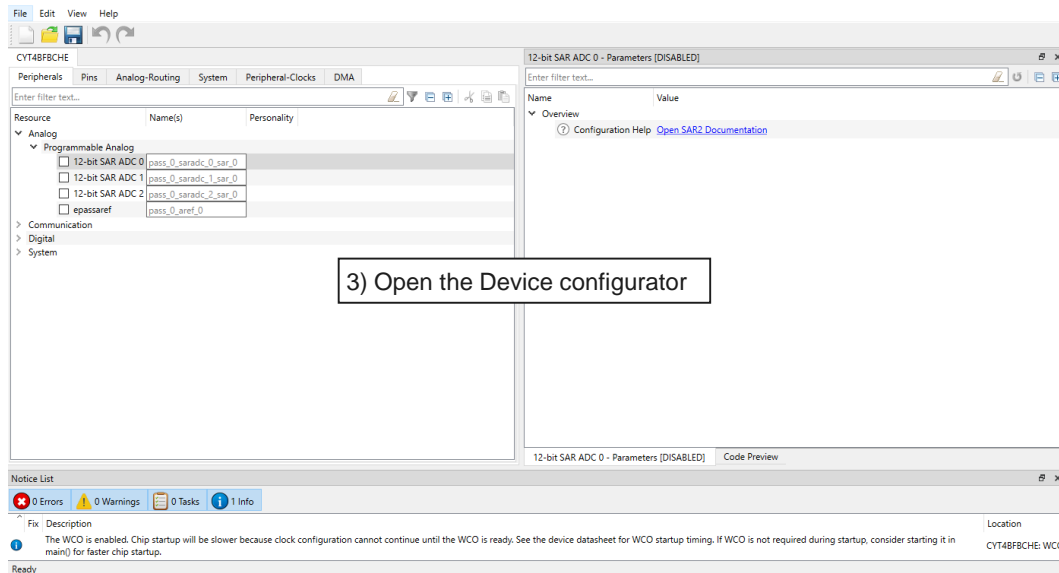
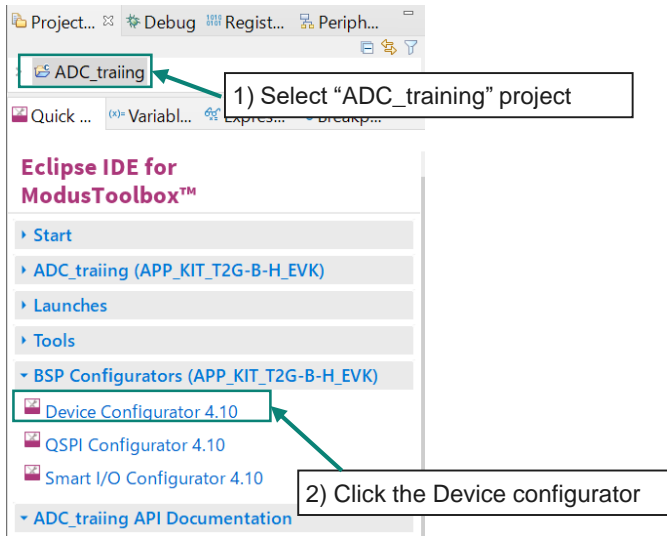
## ADC configuration (contd.)

- Add Library
  - Add “retarget-io” library to the ADC\_training application
    - You can specify the TX pin, RX pin, and the baud rate through the `cy_retarget_io_init()` function.
    - In this session, retarget-io library adds to proj\_cm0p Libraries
1. Select your application (click the ADC\_training), then click the “Library Manager 2.10” in the Quick Panel
  2. Select “proj\_cm0p Libraries” and click the “Add Library” button
  3. Select “retarget-io” in peripheral tab, then click the “OK” button



# ADC configuration (contd.)

- Launch the Device Configurator
  - Select the ADC\_training project.
  - Click the Device Configurator in the Quick Panel
  - Then, open the Device Configurator window



# ADC configuration (contd.)

## – Configure ADC

**Fill Analog name to ADC**

**Select the 12-bit SAR ADC 0**

**General**

- Enable SAR Block:
- Enable The SAR MUX:
- Enable The SAR ADC:
- Precondition Time: 0
- Power Up Time: 0
- Power Down If Idle:
- MSB Cycles: Use 1 clock cycles per conversion
- Half LSB:
- Number Of Channels: 1

**Connections**

- Clock: 16 bit Divider 0 clk [USED]
- Clock Frequency: 13.066667 MHz

**Channel 0**

- Enabled:
- HW Trigger: Generic 0
- Trigger Input: EVTGEN 0 tr\_out[12] [EVTGEN] [USED]
- Trigger Output: <unassigned>
- Trigger Chanel Input: <unassigned>
- Voltage Range Trigger Output: <unassigned>
- Channel Done Trigger Output: <unassigned>

Notice List:

- 0 Errors, 0 Warnings, 0 Tasks, 2 Infos
- The WCO is enabled. Chip startup will be slower because clock configuration cannot continue until the WCO is ready. See the device datasheet for WCO startup timing. If WCO is not required during startup, consider starting it in main() for faster chip startup.
- The file was last saved with a different version of the tools than will be used to perform code generation on save. Last saved with: 'Configurator Backend 3.0.0' from 'ModusToolbox 3.0.0'. Current: 'Configurator Backend 3.0.0' from 'ModusToolbox 3.0.0'.

- General**
- Enable SAR Block:
  - Enable The SAR MUX:
  - Enable The SAR ADC:
  - Precondition Time: 0
  - Power Up Time: 0
  - Power Down If Idle:
  - MSB Cycles: Use 1 clock cycles per conversion
  - Half LSB:
  - Number Of Channels: 1

- Connections**
- Clock: 16 bit Divider 0 clk [USED]
  - Clock Frequency: 13.066667 MHz

- Channel 0**
- Enabled:
  - HW Trigger: Generic 0
  - Trigger Input: EVTGEN 0 tr\_out[12] (EVTGEN) [USED]
  - Trigger Output: <unassigned>
  - Trigger Chanel Input: <unassigned>
  - Voltage Range Trigger Output: <unassigned>
  - Channel Done Trigger Output: <unassigned>

# ADC configuration (contd.)

## – Configure ADC

The screenshot shows the configuration tool interface for a 12-bit SAR ADC 0. The 'Parameters' window is open, displaying a list of settings. A red box highlights the 'Advanced' section, which includes the 'Store Config in Flash' checkbox. The 'Notice List' at the bottom shows 0 errors, 0 warnings, 0 tasks, and 2 infos.

Name	Value
Debug Freeze Input	<unassigned>
Priority	0
Preemption Type	Complete ongoing acquisition (including averaging), up on return Resume
Group End	<input checked="" type="checkbox"/>
Output Trigger Type	Pulse
Input	P6[0] analog (CYBSP_POT, CYBSP_A0) [USED]
Enable External Analog Mux	<input type="checkbox"/>
Precondition Mode	No Preconditioning
Overlap Diagnostic Mode	No Overlap or SARMUX Diagnostics
Sample Time (Aperture)	60
Selection Of Calibration Values	Regular
Result Data Alignment	The data is right aligned in Result[11:0]
Sign Extension	Unsigned
Post Processing Mode	No Post Processing
Averaging Count	1
Shift Right	0
Positive Reload	0
Negative Reload	0
Range Detection Mode	Inside Range (Low <= Result < High)
Range Detect Low Threshold	0
Range Detect High Threshold	0x0FFF
<b>Advanced</b>	
Store Config in Flash	<input checked="" type="checkbox"/>

- Channel 0 (contd.)
- Debug Freeze Input: <unassigned>
  - Priority: 0
  - Preemption Type: Complete ongoing acquisition (including averaging), up on return Resume
  - Group End:
  - Output Trigger Type: Pulse
  - Input: P6[0] analog (CYBSP\_POT, CYBSP\_A0) [USED]
  - Enable External Analog Mux:
  - Precondition Mode: No Preconditioning
  - Overlap Diagnostic Mode: No Overlap or SARMUX Diagnostics
  - Sample Time (Aperture): 60
  - Selection Of Calibration Values: Regular
  - Result Data Alignment: The data is right aligned in Result[11:0]
  - Sign Extension: <unassigned>
  - Post Processing Mode: No Post Processing
  - Averaging Count: 1
  - Shift Right: 0
  - Positive Reload: 0
  - Negative Reload: 0
  - Range Detection Mode: Inside Range (Low <= Result < High)
  - Range Detect Low Threshold: 0
  - Range Detect High Threshold: 0x0FFF

- Advanced
- Store Config in Flash:

# ADC configuration (contd.)

## – Configure Event generator (EVTGEN)

**Fill System name to EVTGEN**

**Select the EVTGEN0**

**Counter**

- Reference Clock: CLK\_HF3 (18 MHz ± 1%)
- Low Frequency Clock: CLK\_LF (32.768 ± 0.015% )
- Counter Tick: 1000000
- Ratio Control Mode: Hardware Control
- Ratio Value: 30.51757813
- Ratio Dynamic Mode: RatioDynamicMode\_0

**Comparator12**

- Enabled:
- Trigger Output: 12-bit SAR ADC 0 tr\_sar\_gen\_in[0] (ADC) [USED]
- Work Mode: Active
- Trigger Mode: Edge Sensitive
- Active Comparator Value: 1000000
- Period Active Event: 1.00000000



# ADC configuration (contd.)

- Confirm configuration result
  - You can check the configuration result in the “Code Preview” tab of the Device Configurator

```
Code Preview
Enter search text...
#include "cy_sar2.h"
#include "cy_sysclk.h"
#include "cyCfg_routing.h"

#define ADC_HW PASS0_SAR0
#define ADC_CH0_HW PASS0_SAR0_CH0
#define ADC_channel_0_HW ADC_CH0_HW
#define ADC_CH0_IRQ pass_0_interrupts_sar_0_IRQon
#define ADC_channel_0_IRQ ADC_CH0_IRQ
#define ADC_VDDA_RANGE CY_SAR2_VDDA_2_TV_TO_4_SV
#ifndef SARMUX0_CH0_PORT_ADDR
#define SARMUX0_CH0_PORT_ADDR 0
#endif
#define ADC_channel_0_IDX (OUL)

const cy_stc_sar2_channel_config_t ADC_channel_0_config =
{
    .channelHwEnable = true,
    .triggerSelection = CY_SAR2_TRIGGER_GENERIC0,
    .channelPriority = 0U,
    .preemptionType = CY_SAR2_PREEMPTION_FINISH_RESUME,
    .doneLevel = CY_SAR2_DONE_LEVEL_FULSE,
    .sigGroupEnd = true,
    .pinAddress = SARMUX0_CH0_PIN_ADDR,
    .portAddress = SARMUX0_CH0_PORT_ADDR,
    .extMaxEnable = false,
    .extMaxSelect = 0U,
    .preconditionMode = CY_SAR2_PRECONDITION_MODE_OFF,
    .overlapDiagMode = CY_SAR2_OVERLAP_DIAG_MODE_OFF,
    .sampleTime = 60U,
    .calibrationValueSelect = CY_SAR2_CALIBRATION_VALUE_REGULAR,
    .postProcessingMode = CY_SAR2_POST_PROCESSING_MODE_NONE,
    .resultAlignment = CY_SAR2_RESULT_ALIGNMENT_RIGHT,
    .signExtension = CY_SAR2_SIGN_EXTENSION_UNSIGNED,
    .averageCount = 1U,
    .rightShift = 0U,
    .positiveReload = 0U,
    .negativeReload = 0U,
    .rangeDetectionMode = CY_SAR2_RANGE_DETECTION_MODE_INSIDE_RANGE,
    .rangeDetectionLoThreshold = 0U,
    .rangeDetectionHiThreshold = 0x0FFFU,
};
const cy_stc_sar2_config_t ADC_config =
{

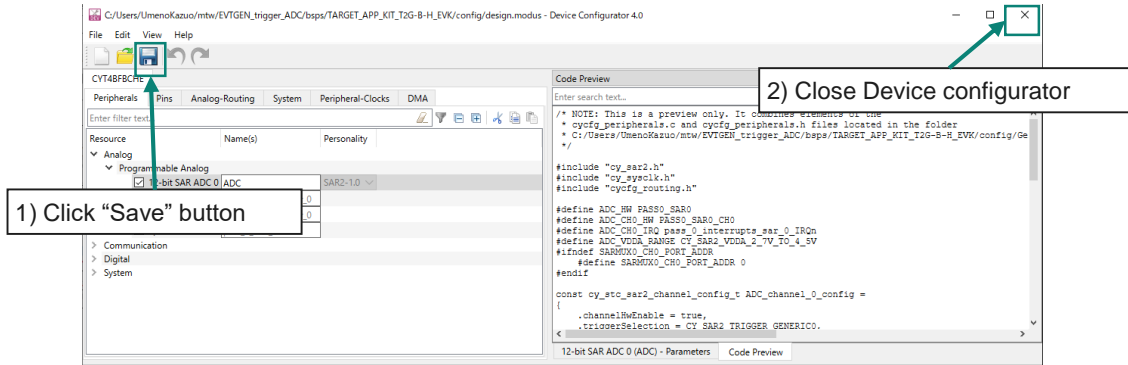
```

Note that it is regardless of the unit number, generated with such as sar2 name so that it can be used with the SAR2 driver.

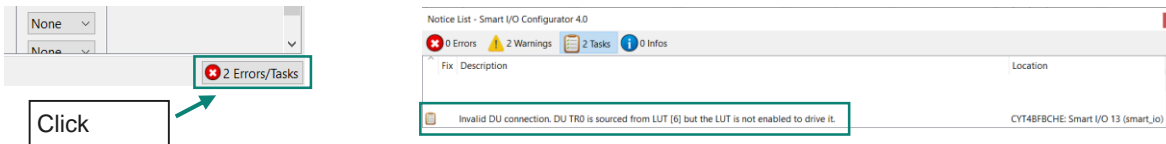
Code preview tab

# ADC configuration (contd.)

- Close Device configurator
  - Click the Save button after completing all the settings, then close the Device configurator

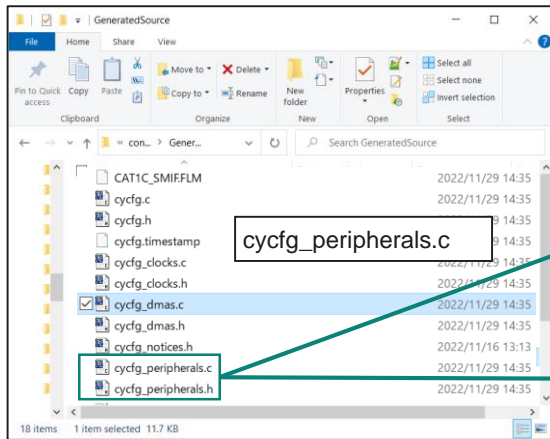


- If an Errors/Tasks message appears, it should be resolved according to the instructions



# ADC configuration (contd.)

- Configuration file
  - The Device Configurator generates code into a "GeneratedSource" directory in your Eclipse IDE application, or in the same location you saved the \*.modus file for non-IDE applications.
  - This example has the following code:



cycfg\_peripherals.c

cycfg\_peripherals.h

```

cy_stc_evtgen_config_t EVTGEN_config = {
    .frequencyRef = 1800000,
    .frequencyLf = 32768,
    .frequencyTick = 1000000,
    .ratioControlMode = CY_EVTGEN_RATIO_CONTROL_HW,
    .ratioValueDynamicMode = CY_EVTGEN_RATIO_DYNAMIC_MODED,
};

cy_stc_evtgen_struct_config_t EVTGEN_comp12_
{
    .functionalitySelection = CY_EVTGEN_ACTI
    .triggerOutEdge = CY_EVTGEN_EDGE_SENSITIV
    .valueActiveComparator = 100000,
    .valueDeepSleepComparator = 1,
};

const cy_stc_sar2_channel_config_t ADC_channe
{
    .channelHwEnable = true,
    .triggerSelection = CY_SAR2_TRIGGER_GENER
    .channelPriority = 0U,
    .preemptionType = CY_SAR2_PREEMPTION_FINI
    .doneLevel = CY_SAR2_DONE_LEVEL_PULSE,
    .isGroupEnd = true,
    .pinAddress = SARMUX0_CH0_PIN_ADDR,
    .portAddress = SARMUX0_CH0_PORT_ADDR,
    .extMuxEnable = false,
    .extMuxSelect = 0U,
    .preconditionTime = CY_SAR2_PRECONDITIONI
    .overlapMode = CY_SAR2_OVERLAP_DIAGN
    .sampleTime = 60U,
    .calibrationValueSelect = CY_SAR2_CALIBRA
    .postProcessingMode = CY_SAR2_POST_PROCE
    .resultAlignment = CY_SAR2_RESULT_ALIGNM
    .signExtension = CY_SAR2_STG_EXTENSION_I
    .averageCount = 1U,
    .rightShift = 0U,
    .positiveBeLoad = 0U,
    .negativeBeLoad = 0U,
    .rangeDetectionMode = CY_SAR2_RANGE_DETE
    .rangeDetectionLoThreshold = 0U,
    .rangeDetectionHiThreshold = 0xFFFFU,
};

const cy_stc_sar2_config_t ADC_config = {
    .preconditionTime = 0U,
    .powerUpTime = 0U,
    .enableIdlePowerDown = false,
    .msbStretchMode = CY_SAR2_MSB_STRETCH_MO
    .enableIFLsDown = false,
    .sarMuxEnable = true,
    .adcEnable = true,
    .sarIpEnable = true,
    .channelConfig = { (cy_stc_sar2_channel_c
    NULL, NULL, NULL, NULL, NULL, NULL, NU
};

#ifdef CYCFG_PERIPHERALS_H
#define CYCFG_PERIPHERALS_H

#include "cycfg_notices.h"
#include "cy_evtgen.h"
#include "cy_syslib.h"
#include "cy_sar2.h"
#include "cy_sysclk.h"
#include "cycfg_routing.h"

#if defined(__cplusplus)
extern "C" {
#endif

#define EVTGEN_ENABLED 1U
#define EVTGEN_HW EVTGEN0

#define EVTGEN_IRO evtgen_0_interrupt_IRQn
#define EVTGEN_DPSLP_IRO evtgen_0_interrupt_dpslp_IRQn

#define ADC_ENABLED 1U
#define ADC_HW PASSO_SAR0
#define ADC_CH0_HW PASSO_SAR0_CH0
#define ADC_channel_0_HW ADC_CH0_HW
#define ADC_CH0_IRO pass_0_interrupts_sar_0_IRQn
#define ADC_channel_0_IRO ADC_CH0_IRO
#define ADC_VDDA_RANGE CY_SAR2_VDDA_2_7V_TO_4_5V
#define ADC_channel_0_IDX (OUL)

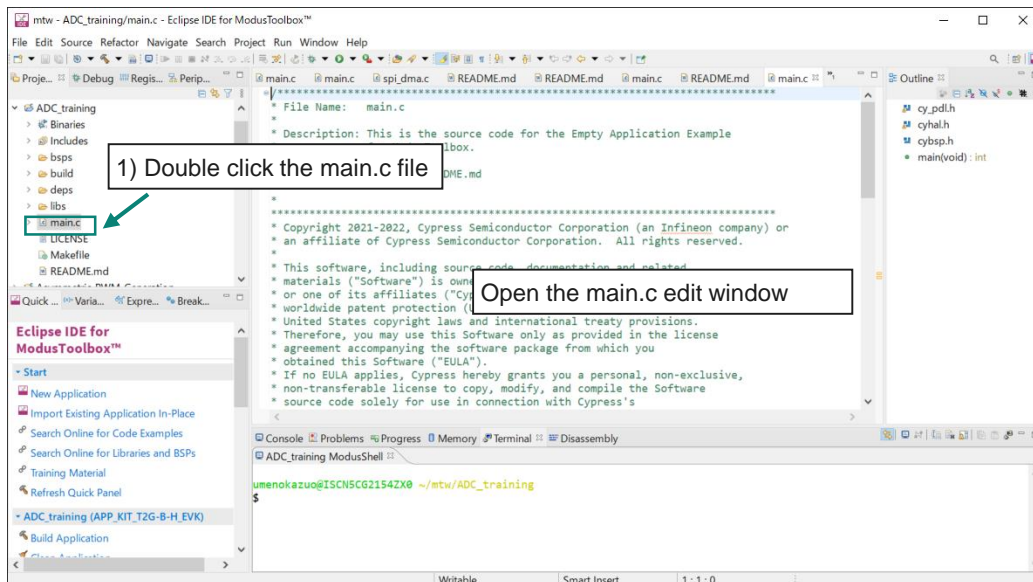
extern cy_stc_evtgen_config_t EVTGEN_config;
extern cy_stc_evtgen_struct_config_t EVTGEN_comp12_config;
extern const cy_stc_sar2_channel_config_t ADC_channel_0_config;
extern const cy_stc_sar2_config_t ADC_config;

void init_cycfg_peripherals(void);

#if defined(__cplusplus)
}
#endif
    
```

# Implementation

- The structure generated by the Device Configurator can be used by implementing the following function in your application code.



# Implementation (contd.)

– Add ADC, Event generator initialization and enable function

There is structure to configure ADC and Event generator in the cycfg\_peripherals.c file

```

main.c | README.md
/* Initialize ADC */
Cy_SAR2_Init(ADC_HW, &ADC_config);
/* Set ADC group done interrupt */
Cy_SAR2_Channel_SetInterruptMask(ADC_HW, ADC_LOGICAL_CHANNEL, CY_SAR2_INT_GRP_DONE);

/* Register ADC interrupt handler and enable interrupt */
Cy_SysInt_Init(&irq_cfg_sar, &adc_int_handler);
NVIC_ClearPendingIRQ(ADC_IRQ_NUM);
NVIC_EnableIRQ((IRQn_Type)ADC_IRQ_NUM);

/* Register event generator interrupt handler and enable interrupt */
Cy_SysInt_Init(&irq_cfg_evtgen, &evtgen_isr);
Cy_EvtGen_ClearInterrupt(EVTGEN_HW, 0xFFu);
NVIC_ClearPendingIRQ(EVTGEN_IRQ_NUM);
NVIC_EnableIRQ((IRQn_Type)EVTGEN_IRQ_NUM);

/* Initialize event generator */
Cy_EvtGen_Init(EVTGEN_HW, &EVTGEN_config);

Cy_EvtGen_Enable(EVTGEN_HW);

/* Delay a bit and wait for the counter completed initialization */
Cy_SysLib_DelayUs(625);
/* Check if the ratio status is valid during hardware control ratio */
if((CY_EVTGEN_RATIO_CONTROL_HW == EVTGEN_config.ratioControlMode) && (!Cy_EvtGen_GetRatioS
{
    CY_ASSERT(0);
}
/* Check if the event generator counter status is valid */
if(CY_EVTGEN_COUNTER_STATUS_VALID != Cy_EvtGen_GetCounterSt
{
    CY_ASSERT(0);
}

/* Initialize the event generator comparator structure */
Cy_EvtGen_InitStruct(EVTGEN_HW, EVTGEN_COMP_STRUCT_NUM, &EVTGEN_comp12_config);
    
```

Add Cy\_SAR2\_Init() initialization function for ADC

Add ADC int handler

Add Cy\_EvtGen\_Init() initialization function for event generator

Add Event generator enable

Add Cy\_EvtGen\_InitStruct() initialization function for comparator structure

```

cy_stc_evtgen_config_t EVTGEN_config = {
    .frequencyRef = 1800000u,
    .frequencyTick = 32768u,
    .frequencyDiv = 100000u,
    .ratioControlMode = CY_EVTGEN_RATIO_CONTROL_HW,
    .ratioValueDynamicMode = CY_EVTGEN_RATIO_DYNAMIC_MODED,
};
cy_stc_evtgen_struct_config_t EVTGEN_comp12_config = {
    .functionalitySelection = CY_EVTGEN_ACTIVE_FUNCTIONALITY,
    .triggerOutEdge = CY_EVTGEN_EDGE_SENSITIVE,
    .valueActiveComparator = 100000u,
    .valueDeepSleepComparator = 1,
};
const cy_stc_sar2_channel_config_t ADC_channel_0_config = {
    .channelHWEnable = true,
    .triggerSelection = CY_SAR2_TRIGGER_GENERICO,
    .channelPriority = 0U,
    .preemptionType = CY_SAR2_PREEMPTION_FINISH_RESUME,
    .doneLevel = CY_SAR2_DONE_LEVEL_PULSE,
    .isGroupEnd = true,
    .pinAddress = SARMUX0_CH0_PIN_ADDR,
    .portAddress = SARMUX0_CH0_PORT_ADDR,
    .muxEnable = false,
    .muxSelect = 0U,
    .preconditionMode = CY_SAR2_PRECONDITION_MODE_OFF,
    .overlapDiagMode = CY_SAR2_OVERLAP_DIAG_MODE_OFF,
    .sampleTime = 600,
    .calibrationValueSelect = CY_SAR2_CALIBRATION_VALUE_REGULAR,
    .postProcessingMode = CY_SAR2_POST_PROCESSING_MODE_NONE,
    .resultAlignment = CY_SAR2_RESULT_ALIGNMENT_RIGHT,
    .signExtension = CY_SAR2_STGN_EXTENSION_UNSTGNED,
    .averageCount = 1U,
    .rightShift = 0U,
    .positiveReload = 0U,
    .negativeReload = 0U,
    .rangeDetectionMode = CY_SAR2_RANGE_DETECTION_MODE_INSIDE_RANGE,
    .rangeDetectionThreshold = 0U,
    .rangeDetectionThreshold = 0x0FFFu,
};
const cy_stc_sar2_config_t ADC_config = {
    .preconditionTime = 0U,
    .powerupTime = 0U,
    .enableIdlePowerDown = false,
    .msbStretchMode = CY_SAR2_MSB_STRETCH_MODE_1CYCLE,
    .enableHalfLsbConv = false,
    .sarMuxEnable = true,
    .adcEnable = true,
    .sar1Enable = true,
    .channelConfig = {(cy_stc_sar2_channel_config_t *)&ADC_channel_0_config,
    NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
    };
    
```

## Implementation (contd.)

- Add ADC initialization and enable function
- See the EVTGEN\_trigger\_ADC application for others

```

main.c
*
***** ADC int handler *****
void adc_int_handler (void)
{
    uint32_t adc_status = 0;

    /* Get interrupt status */
    uint32_t intrSource = Cy_SAR2_Channel_GetInterruptStatusMasked(ADC_HW, ADC_LOGICAL_CHANNEL);
    if(CY_SAR2_INT_GRP_DONE == (intrSource & CY_SAR2_INT_GRP_DONE))
    {
        /* Get the ADC conversion result and status */
        adc_result = Cy_SAR2_Channel_GetResult(ADC_HW, ADC_LOGICAL_CHANNEL, &adc_status);
        if(CY_SAR2_STATUS_VALID == (adc_status & CY_SAR2_STATUS_VALID))
        {
            adc_done_flag = 1;
        }
        /* Clear interrupt source */
        Cy_SAR2_Channel_ClearInterrupt(ADC_HW, ADC_LOGICAL_CHANNEL, CY_SAR2_INT_GRP_DONE);
    }
}

```

ADC int handler

Add Cy\_SAR2\_Channel\_GetResult() function for ADC conversion result

## Implementation (contd.)

### ADC initialization

- Call the [CY\\_SAR2\\_Init\(\)](#) function to initialize the ADC channel
  - Initialize the ADC, connect to use pin port 6.0 as input

### ADC interrupt configuration

- Call the [Cy\\_SAR2\\_Channel\\_SetInterruptMask\(\)](#) function to configure the channel interrupt.
  - Specify [CY\\_SAR2\\_INT\\_GRP\\_DONE](#) as an argument to generate an interrupt when the conversion is completed
- Configure interrupt in the [CY\\_SysInt\\_Init\(\)](#) function.
  - Set the interrupt source (ADC channel 0), interrupt priority (7), interrupt vector, and the ISR ([adc\\_int\\_handler\(\)](#))

### Get ADC conversion result

- Call the [Cy\\_SAR2\\_Channel\\_GetResult\(\)](#) function to get the ADC conversion result and status.
- Call the [Cy\\_SAR2\\_Channel\\_ClearInterrupt\(\)](#) function to clear the interrupt factor.
  - Clear interrupt flag of specified ADC channel (channel 0 of ADC0)

## Implementation (contd.)

### Event Generator interrupt configuration

- Configure interrupt in the [CY\\_SysInt\\_Init\(\)](#) function
  - Set the interrupt source (EVTGEN0), interrupt priority (7), interrupt vector, and the ISR (evtgen\_isr())
  - Call the [Cy\\_EvtGen\\_ClearInterrupt\(\)](#) function
  - Set the EVTGEN0 bit to 0 to clear the interrupt

### Event Generator initialization

- Call the [Cy\\_EvtGen\\_Init\(\)](#) function to initialize the Event Generator
- Initializes Event Generator parameters (clock source frequency in Active/DeepSleep power mode, EVTGEN customized period, ratio control mode and specific dynamic mode)

### Event Generator comparator structure initialization

- Call the [Cy\\_EvtGen\\_InitStruct\(\)](#) function to initialize the Event Generator comparator structure
  - Initializes the Event Generator parameters (functionality comparator structure, condition for start trigger/interrupt, making period of interrupts/triggers and the making period of interrupts during DeepSleep)



## Implementation (contd.)

### Event Generator comparator structure initialization

- Call the [Cy EvtGen InitStruct\(\)](#) function to initialize the Event Generator comparator structure.
  - Initializes the Event Generator parameters (functionality comparator structure, condition for start trigger/interrupt, making period of interrupts/triggers and the making period of interrupts during DeepSleep)
  - Refer to [cy\\_stc\\_evtgen\\_struct\\_config\\_t](#) for parameter details

### Update active comparator value

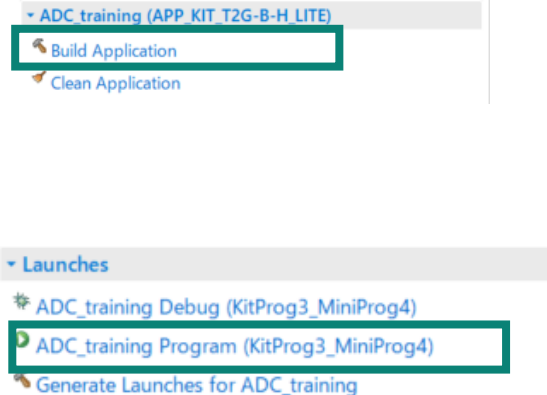
- Check that the interrupt source is EVTGEN0 with the return value of the [Cy EvtGen GetStructInterrupt\(\)](#) function.
  - Get interrupt flag of EVTGEN0
- Call the [Cy EvtGen ClearStructInterrupt\(\)](#) function to clear the interrupt factor.
  - Clear interrupt flag of EVTGEN0
- Call the [Cy EvtGen UpdateActiveCompValue\(\)](#) function to update the active comparator value.
  - Update active comparator to initial value (1000000 = 1 sec); this can be modified to change the ADC conversion cycle

# Compiling and programming

1. Connect to power and USB cable.
2. Use Eclipse IDE for ModusToolbox™ software for compiling and programming.
  
3. Compile
  - a. Select the target application project in the Project Explorer.
  - b. In the Quick Panel, scroll down and click “Build Application” in ADC\_training (APP\_KIT\_T2G-B-H\_LITE).
  
4. Open a terminal program and select the KitProg3 COM port. Set the serial port parameters to 8N1 and 115200 baud.
  
5. Programming
  - a. Select the target application project in the Project Explorer.
  - b. In the Quick Panel, scroll down and click “ADC\_training Program (KitProg3\_MiniProg4)” in Launches.



USB connector



## Run and test

1. After programming, the application starts automatically. Ensure that input voltages are provided at the analog input pin P6.0. This pin is connected to the potentiometer.
2. Rotate the potentiometer to change the ADC input voltage, and the result prints out every second in the terminal window.
3. Confirm that the input voltages are displayed on the UART terminal.

```

ADC conversion complete, result: 847
ADC conversion complete, result: 1305
ADC conversion complete, result: 1585
ADC conversion complete, result: 1927
ADC conversion complete, result: 2232
ADC conversion complete, result: 2442
ADC conversion complete, result: 2544
ADC conversion complete, result: 2543
ADC conversion complete, result: 2543
ADC conversion complete, result: 2544
ADC conversion complete, result: 2544
ADC conversion complete, result: 2542
ADC conversion complete, result: 2543
ADC conversion complete, result: 2543
ADC conversion complete, result: 2543

```



Potentiometer

# References

## Datasheet

- [CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family](#)

## Architecture Technical reference manual

- [TRAVEO™ T2G automotive body controller high family architecture technical reference manual](#)

## Registers Technical reference manual

- [TRAVEO™ T2G Automotive body controller high registers technical reference manual](#)

## PDL/HAL

- [PDL](#)
- [HAL](#)

## Training

- [TRAVEO™ T2G Training](#)

# Revision History

Revision	ECN	Submission Date	Description of Change
**	7942668	2023/07/25	Initial release

# Important notice and warnings

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

**Edition 2023-07**

**Published by  
Infineon Technologies AG  
81726 Munich, Germany**

**© 2023 Infineon Technologies  
AG.  
All Rights Reserved.**

**Do you have a question about  
this document?**

**Go to:**  
[www.infineon.com/support](http://www.infineon.com/support)

**Document reference  
002-38205 Rev. \*\***

## IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”).

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

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](http://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.

