Customer training workshop: TRAVEO™ T2G Sample driver library

ATV MC TM CES
November 2021

Agenda

1. What is SDL? What does the SDL include?
2. Supported toolchains
3. Folder structure, drivers, middleware
4. Startup sequence
5. Sample blinky main
6. Workspaces – Open, build, download, debug, run, pause, reset, and stop
7. External links
8. Support

Target products

› Target product list for this training material

Family category	Series	Code flash memory size
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B6	Up to 576 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B7	Up to 1088 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B9	Up to 2112 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2BL	Up to 4160 KB
TRAVEO™ T2G Automotive Body Controller High	CYT3BB/4BB	Up to 4160 KB
TRAVEO™ T2G Automotive Body Controller High	CYT4BF	Up to 8384 KB
TRAVEO™ T2G Automotive Cluster Entry	CYT2CL	Up to 4160 KB
TRAVEO™ T2G Automotive Cluster 2D	CYT3DL	Up to 4160 KB
TRAVEO™ T2G Automotive Cluster 2D	CYT4DN	Up to 6336 KB

What is SDL

- Infineon's sample driver library (SDL) simplifies software development for TRAVEO™ T2G devices
  - Drivers for an extensive set of peripherals
  - Arm® Cortex® Microcontroller Software Interface Standard (CMSIS) core header files directly from the CMSIS 5.7 release
  - CMSIS compliant device header files, startup code (platform initialization), and device configuration header files
  - Application programming interface reference manual
  - Examples to evaluate various peripherals
- SDL is provided as an executable, tested on Windows 10 with a minimum installation size requirement of around 400 MB.

Disclaimer

SDL sample software is provided “as-is” and for **evaluation purposes only**

It does not adhere to any industry standard and is **not a production software**

Infineon may, but is **not required to**, provide technical **support** for the SDL

In **no** event shall Infineon be **liable** for incidental or consequential **damages** arising from **use** of the **SDL**.

What does the SDL include?

The SDL contains

- Device-specific header files that provide a complete definition of all peripheral registers and bits in the respective device
- CMSIS-compliant startup code to initialize the system after device reset, and transfer the code execution to main()
- Template workspaces, linker files for each supported device, and toolchain (IAR and GHS)
- Peripheral drivers and middleware
- Code examples covering all basic functions of all peripherals supported by the device
- Tool-specific patches to support download and debug, and SVD (system view description) files

Supported toolchains

- Green Hills MULTI: 7.1.4, Compiler: 2017.1.4, Probe Version: 6.4.4 recommended
  - Development Autobuild 5.6 634260/AB as of patch #12996, or higher

- IAR Embedded Workbench for Arm® 8.42.1 (EWARM-CD-8421-xxxxx.exe), IAR I-Jet debugger
  - Flash loaders are available at the location
    - "\misc\tools\iar\IAR_EWARM_8421_FlashLoader_Patch_TraveoII"
  - Refer to "\misc\tools\iar\Readme_Patch.txt" to update the TRAVEO™ T2G patch for IAR

SDL patch for the tools

- Green Hills toolchain
  - Install GHS MULTI, set license, and set probe firmware in advance
  - Using the scripts tvii_detect.py and multi.irc, in \misc\tools\ghs\debugging\AppData_GHS
    - Copy the files to the <%APPDATA%\GHS> folder (C:\Users\<LOGIN_NAME>\AppData\Roaming\GHS) on the PC

- IAR patch: /misc/tools/iar/IAR_EWARM_8421_FlashLoader_Patch_TraveoII.7z
  - Copy files of \misc\tools\iar\IAR_EWARM_8421_FlashLoader_Patch_TraveoII.7z to the IAR install folder (do not copy anything at folder level)
  - Restart IAR EWARM
  - Rebuild All
  - Download & debug

SDL folder structure (1/2)

Path/Folder	Description
common/ hdr/cmsis	CMSIS core access headers
common/ src/drivers	Drivers common across all the devices
common/ src/mw	Middleware common across all the devices
common/ src/startup	Tool specific startup code for all the devices
docs	SDL API documentation, release notes, known issues
misc/tools	GHS/IAR specific flash loaders, SVD files
TRAVEO™ T2G Body and Cluster Entry devices (tviibe1m/2m/4m/512k, tviice4m)	
hdr	Device-specific header files, BSP for T2G Base Board/CPU boards, GPIO assignments
hdr/ip	Device IP specific headers
hdr/mcureg	IP Specific Register Addresses
src/drivers	Driver source and respective headers specific to TVIIBE1M/2M/4M/512K, TVIICE4M device
src/examples	Code examples in accordance to TVIIBE1M/2M/4M/512K, TVIICE4M device
src/system	TVIIBE1M/2M/4M/512K, TVIICE4M system specific code and system header for clock configurations
src/interrupts/cy_interrupt_map_cm0plus.h	User interrupt mapping file
src/interrupts/cy_interrupt_map_cm4.h	User interrupt mapping file
src/main_cm0plus.c	Sample main source file for CM0+ core
src/main_cm4.c	Sample main source file for CM4 core
tools/ghs	GHS MULTI workspaces for SRAM/Flash for CM0+/CM4 cores, linker specific files, and GRD files
tools/iar	IAR workspaces for SRAM/Flash for CM0+/CM4 cores, linker specific files

SDL folder structure (2/2)

Path/Folder	Description
TRAVEO™ T2G Body High and Cluster 2D devices (tviibh4m/8m, tviic2d4m/6m/6mddr)	
hdr	Device-specific header files, BSP for T2G Base Board/CPU boards, GPIO assignments
hdr/ip	Device IP specific headers
hdr/mcureg	IP Specific Register Addresses
src/drivers	Driver source and respective headers specific to TVIIBH4M/8M, TVIIC2D4M/6M/6MDDR devices
src/mw/	Middleware support
src/examples	Code examples in accordance device specific
src/system	Device system specific code and system header for clock configurations
src/interrupts/cy_interrupt_map_cm0plus.h	User interrupt mapping file
src/interrupts/cy_interrupt_map_cm7_0.h	User interrupt mapping file
src/interrupts/cy_interrupt_map_cm7_1.h	User interrupt mapping file
src/main_cm0plus.c	Sample main source file for CM0+ core
src/main_cm7_0.c	Sample main source file for CM7_0 core
src/main_cm7_1.c	Sample main source file for CM7_1 core
tools/ghs	GHS MULTI workspaces for SRAM/Flash for CM0+/CM7_0/CM7_1 cores, linker specific files, and GRD files
tools/iar	IAR workspaces for SRAM/Flash for CM0+/CM7_0/CM7_1 cores, linker specific files

SDL drivers (1/3)

Driver	Description	API functionality
ADC	Analog to digital converter	Manage ADC operations
Audioss	Sound Subsystem for I^2S, DAC, Mixer, PWM, SG, TDM	Manages I2S, Audio DAC, Mixer, PCM-PWM, Sound Generator, TDM as part of sound subsystem
AXIDMA	M-DMA on AXI bus	Memory to memory transfer over AXI bus
CAN FD	Controller Area Network Flexible Data-Rate	Manages Classic and FD operations
CPU	CPU driver	Enables core of CPU specific features
CRYPTO	Cryptographic Operations	Perform cryptographic operations on user-designated data. Available as libraries
CXPI	Clock eXtension Peripheral Interface	Manages communication over CXPI interface
DMA	Direct Access Memory	Perform memory-to-memory (M-DMA) and peripheral-to-memory (P-DMA) (and vice versa) operations
ETHERNET	Ethernet	Basic ethernet driver supporting automotive and gigabit ethernet PHYs
EVTGEN	Event Generator	Performs event generation for interrupts and triggers in active power mode
FLASH	Flash Memory	Manage code/work flash memory operations
FLEXRAY	FlexRay Interface	Manages FlexRay communication
FPDLINK	FDP-Link or LVDS	FPD-link or LVDS video driver
GPIO	General purpose I/O ports	Configure and access device input/output pins
I^2S	Inter-IC Sound (TVII-B-H devices Only)	Manage Inter-IC Sound. I2S is used to send digital audio streaming data to external I2S devices, such as audio codecs or simple DACs. It can also receive digital audio streaming data

SDL drivers (2/3)

Driver	Description	API functionality
IPC	Inter process communication	Manage data transfer between CPUs or processes in a device
LIN	Local interconnect network	Provides master and slave data transfer capabilities
LVD	Low voltage detection	Provides LVD capabilities
MCWDT	Multi-counter watchdog timer	Provides control and status capabilities
MIPICSI2	Video input	Manage and control serial camera inputs
MPU	Memory protection unit	Manages the configuration of core specific MPU
PROT	Memory and peripheral protection	Manages the MPU, Shared MPU (SMPU), and Peripheral Protection Unit (PPU) structures for memory and peripheral secured access
SCB	Serial communication block	Manage serial communication as I2C, SPI, or UART
SD_HOST	Secure digital host controller	Manages SD and eMMC devices
SMART IO	Smart I/O	Configure and access the Smart I/O hardware present between the GPIOs (pins) and HSIOMs (pin muxes) on select device ports. It can be used to perform simple logic operations on peripheral and GPIO signals at the GPIO port
SMIF	Serial memory interface	SPI-based communication interface for interfacing external memory devices to T2G. The SMIF supports Octal-SPI, Dual Quad-SPI, Quad-SPI, DSPI, and SPI. This interface also supports xSPI protocol devices like HyperRAM and HyperFlash devices.
SROM	Internal SROM driver	APIs to support some basic access to SROM System calls
SYSCLK	System clock	Provides APIs to control and read status of various clocking capabilities of the device
SYSFLT	System fault	Controls CPUs fault processing Subsystem
SYSINT	System interrupt	Manage interrupts and exceptions, in conjunction with the CMSIS core NVIC API

SDL drivers (3/3)

Driver	Description	API functionality
SYSLIB	System library	Utility functions to handle delays, register read/write, asserts, silicon unique ID, and more
SYSPM	System power modes	Controls device power modes
SYSREGHC/ SYSPMIC	REGHC/PMIC control and status	Controls High Current Regulator or the PMIC module
SYSRESET	System reset	Provides APIs for reading reset reason and clearing them
SYSRTC	System real time clock	Provides capabilities to handle RTC, Alarms etc.
SYSTICK	Systick timer	Manage a 24-bit down-counter timer
SYSWDT	Free running watchdog timer	Provides control and status capabilities
TCPWM	Timer counter PWM	Manage a 16- or 32-bit periodic Counter, PWM, Quadrature decoder, Shift register
TRIGMUX	Trigger multiplexer	Manage the multiplexing of trigger outputs to specific trigger inputs across multiple peripherals

SDL middleware

› Device-specific

Middleware	Description	API functionality
GFX_ENV	Graphics environment setup	Supported only for TVIIC2D6M/TVIIC2D4M (Graphics environment setup support)
MIPI_SENSOR	MIPI CSI2 controller	Support top level MIPI CSI2 APIs for camera access map to capture interface of VIDEOSS IP
POWER	Reghc or PMIC based power control	REGHC or PMIC controller middleware (REGHC/TVIIBH4M/TVIIBH8M, PMIC/TVIIC2D6M/TVIIC2D4M)
SMIF_MEM	SMIF SPI/Hyper Access Control	SPI or HyperBus specific device support

› Common

Middleware	Description	API functionality
Button	Button middle layer	APIs to support buttons
Semihosting	SCB/UART middle layer	Supports “printf” via UART for debugging
SW_Timer	Software timer	Enables multiple software timers
Flash	Code and work flash	User level APIs for ease of use

› Hardware-specific middleware such as CS42488, DP83867, AIC26, TJA110, etc.

Device startup sequence

› System reset (@0x0000 0000)

› CM0+ executes ROM boot (@0x0000 0004[1])
- Applies trims
- Applies debug access port (DAP) access restrictions and system protection from eFuse and supervisory flash
- Authenticates flash boot (only in SECURE life-cycle stage) and transfers control to it

› CM0+ executes flash boot (@0x1700 2000)
- Configures debug pins as per the SWD/JTAG
- Sets CM0+ vector offset register to the beginning of flash (@0x1000 0000) (Simple case without TOC2)
- Flash Boot saves the user's vector table address in a register, then triggers CM0+ soft reset, ROM is executed again, but this time a short cut is taken and the shortcut will then use the information from the register to set SP and PC according to users configuration

› CM0+ starts its application execution
- Moves CM0+ vector table to SRAM (updates CM0+ vector table base)
- Sets CM4 / CM7_VECTOR_TABLE _BASE to the location of CM4/7 vector table mentioned in flash
- Releases CM4/CM7 from reset
- Continues execution of CM0+ user application

› CM4/CM7 executes directly from either code-flash or SRAM
- CM4/7 branches to its Reset handler
- Continues execution of CM4/7 user application

1. ROM boot code is located at the address pointed by the value @0x0000 0004

Application startup sequence

Dual core application

- System reset (@0x1000 0000)

- Enables CM4/7 application core

- CM0+ executes application main and calls “SystemInit”:
  - Disables WDT
  - Applies flash wait states
  - Sets clock configuration
  - Enables generic system IRQ
  - Continues execution of CM0+ user application

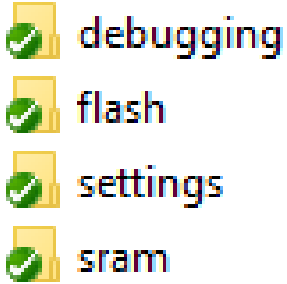
SDL sample blinky example

```c
#include "cy_project.h"
#include "cy_device_headers.h"

#if (CY_USE_PSVP == 1)
    #define USER_LED_PORT           CY_LED0_PORT
    #define USER_LED_PIN            CY_LED0_PIN
    #define USER_LED_PIN_MUX        CY_LED0_PIN_MUX
#else
    #define USER_LED_PORT           CY_CB_LED_PORT
    #define USER_LED_PIN            CY_CB_LED_PIN
    #define USER_LED_PIN_MUX        CY_CB_LED_PIN_MUX
#endif


cy_stc_gpio_pin_config_t user_led_port_pin_cfg =
{
    .outVal = 0x00,
    .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
    .hsiom = USER_LED_PIN_MUX,
    .intEdge = 0,
    .intMask = 0,
    .vtrip = 0,
    .slewRate = 0,
    .driveSel = 0,
    .vregEn = 0,
    .ibufMode = 0,
    .vtripSel = 0,
    .vrefSel = 0,
    .vohSel = 0,
};
```

```c
int main(void)
{
    __enable_irq();

    SystemInit();

    /* Enable CM4. CY_CORTEX_M4_APPL_ADDR is calculated in linker script,
       check it in case of problems. */
    Cy_SysEnableApplCore(CY_CORTEX_M4_APPL_ADDR);

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    Cy_GPIO_Pin_Init(USER_LED_PORT, USER_LED_PIN, &user_led_port_pin_cfg);

    for(;;)
    {
        // Wait 0.05 [s]
        Cy_SysTick_DelayInUs(50000);
        Cy_GPIO_Inv(USER_LED_PORT, USER_LED_PIN);
    }
}
```

SDL supported workspaces

› Workspaces supported
  – Flash
  – SRAM

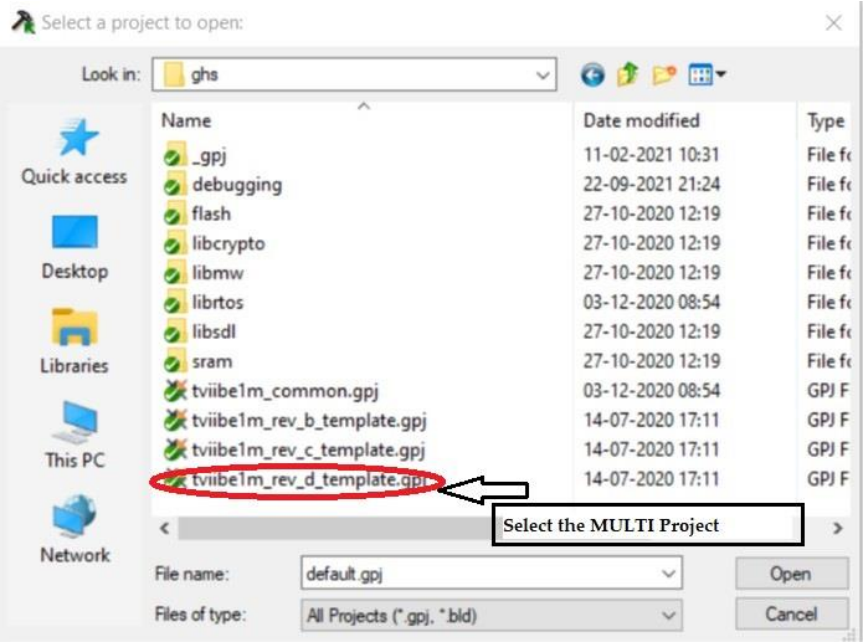


› Flash workspace downloads the example application code on to the device code flash area and is permanent; the program will stay in between resets

› SRAM workspace downloads the example application code on to the device's SRAM area; a power on and off will wipe out the program

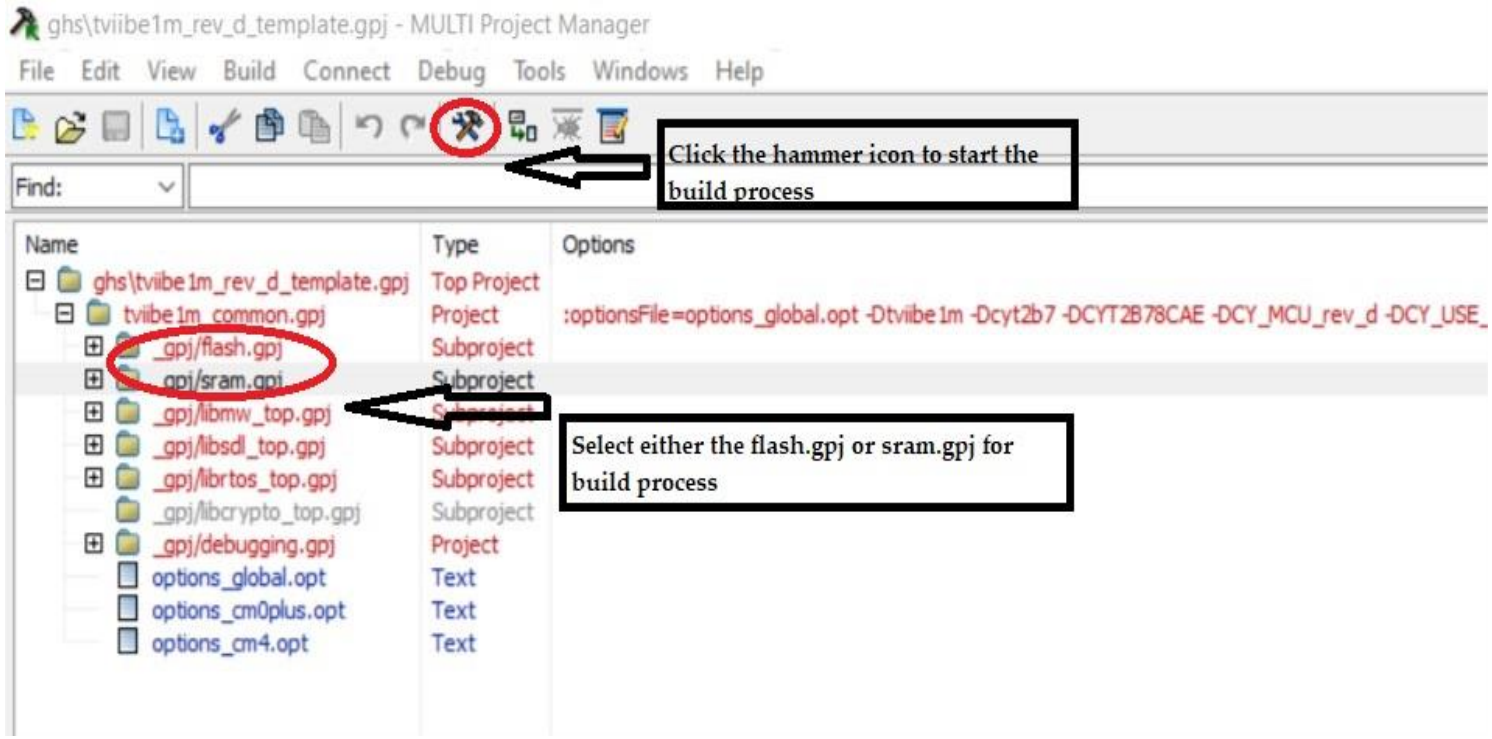
SDL GHS toolchain

- Green Hills toolchain
  - Start the MULTI Project Manager and open the SDL project file
    - T2G_Sample_Driver_Library_rev*device*\tools\ghs*device*_*rev*_template.gpj
  - Select the template



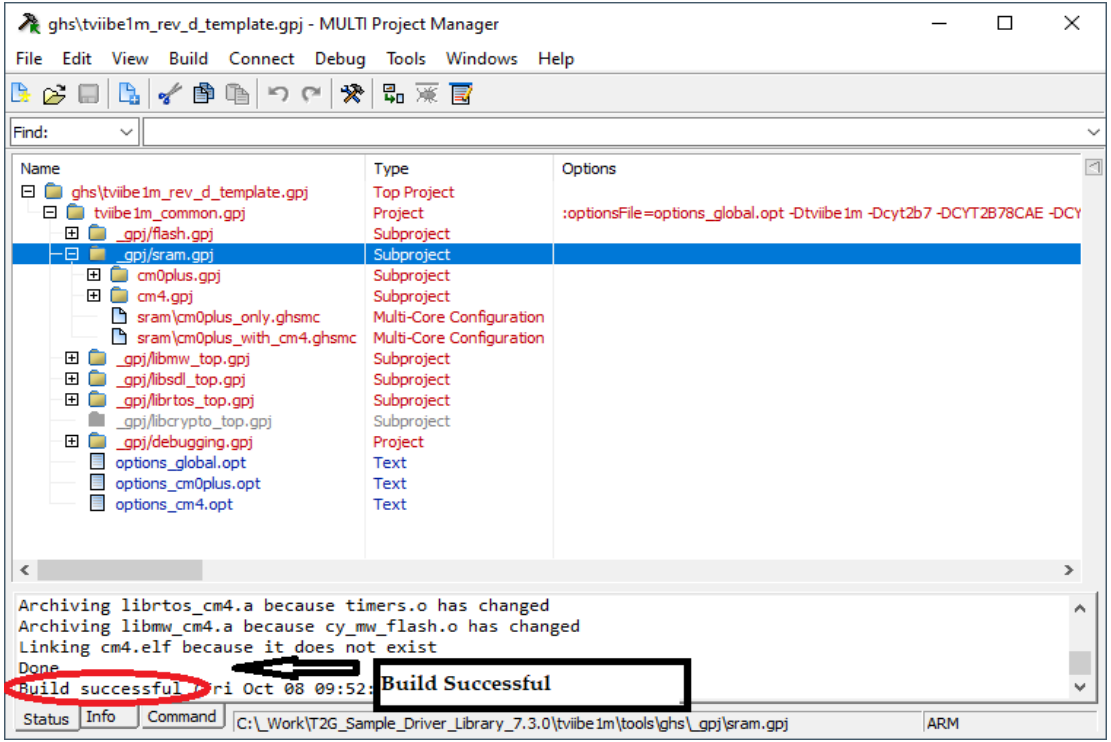
SDL GHS toolchain – Build (1/2)

› Building a project



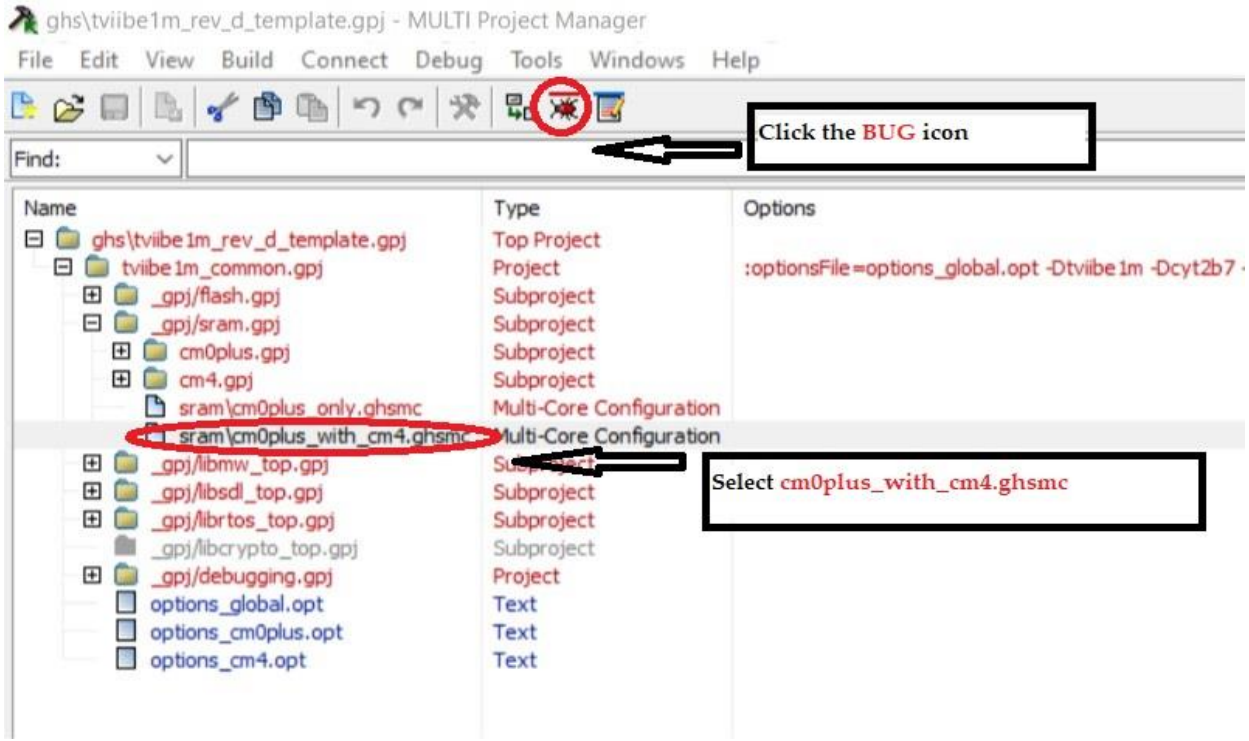
SDL GHS toolchain – Build (2/2)

› Build successful



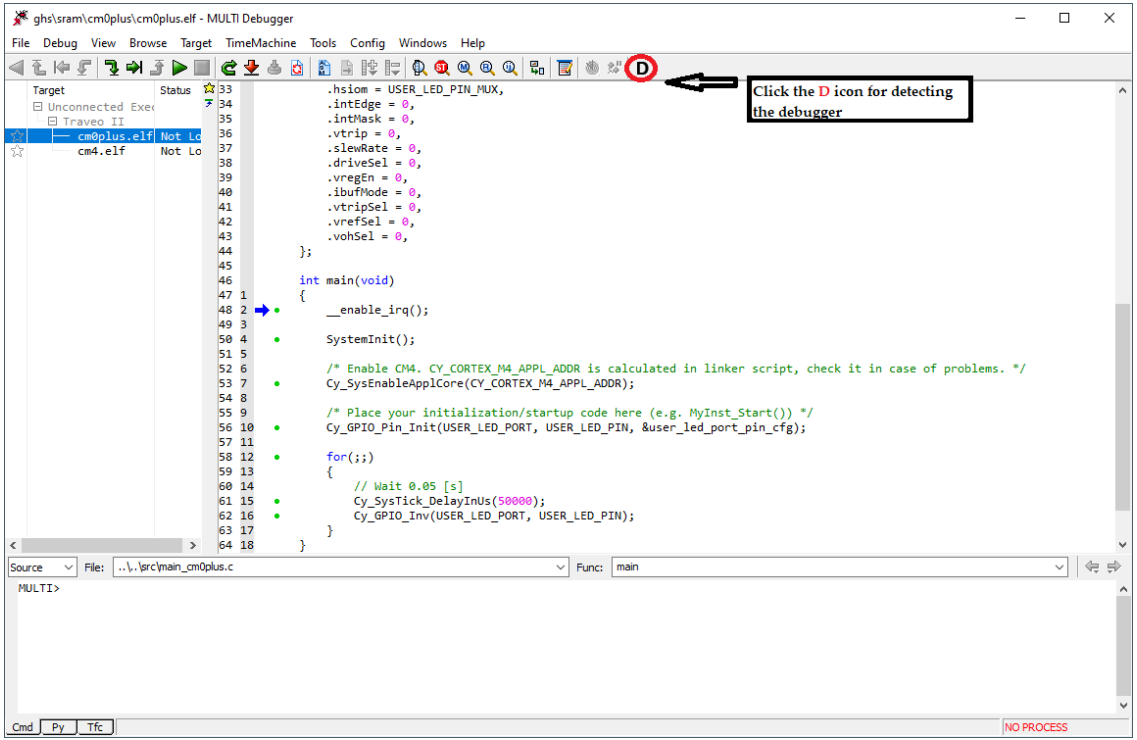
SDL GHS toolchain – Debug (1/2)

› Start the debugger



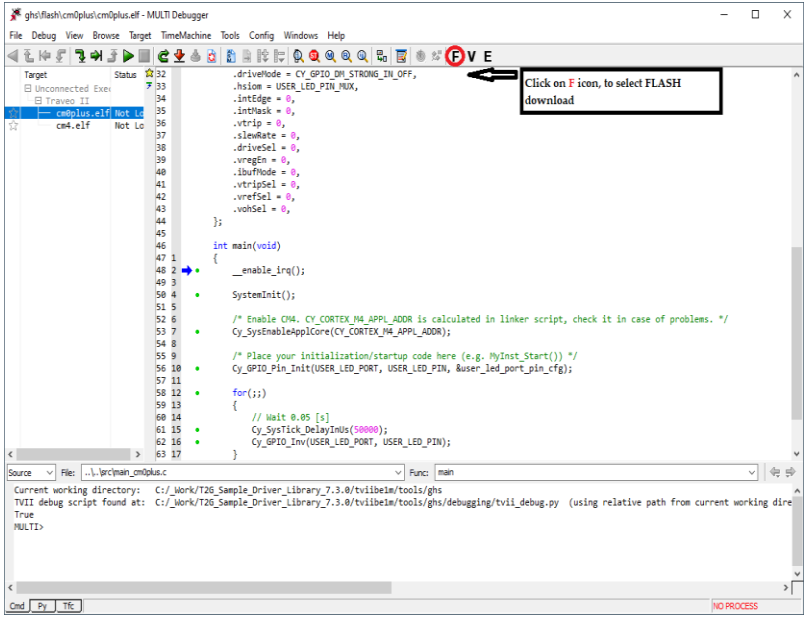
SDL GHS toolchain – Debug (2/2)

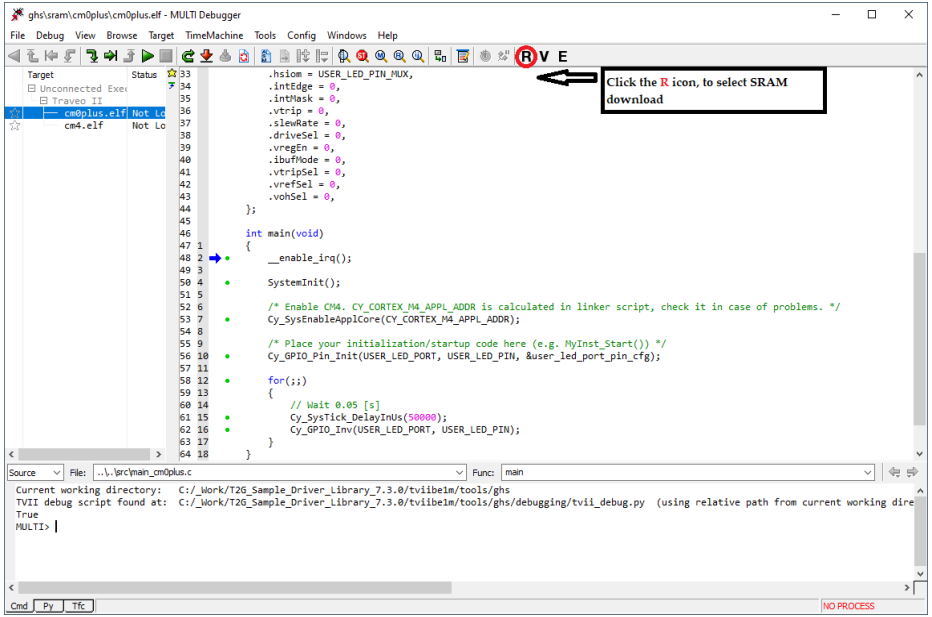
› Detect the debugger



SDL GHS toolchain - Load

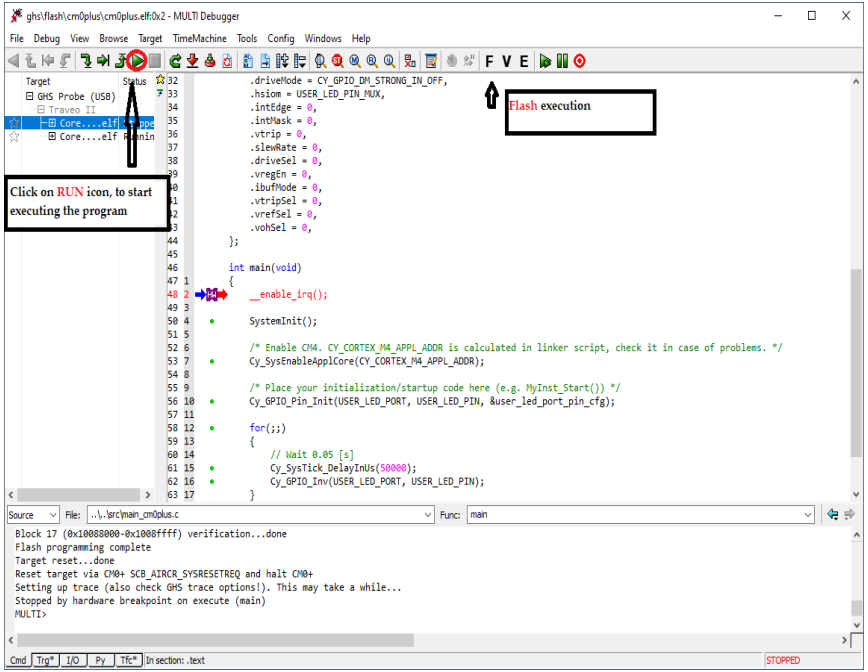
› Load the program (either to flash or SRAM memory)

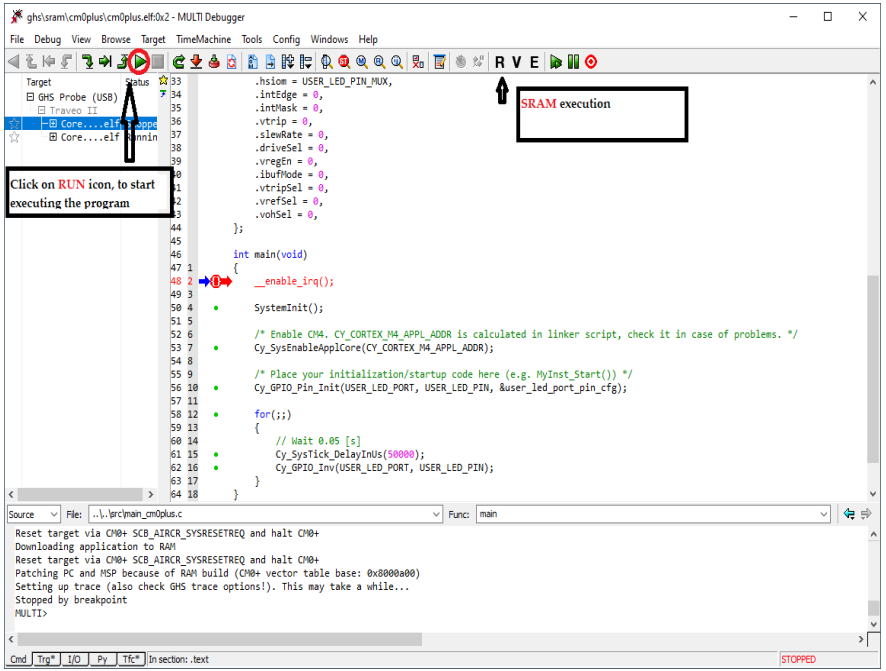




SDL GHS toolchain - Run

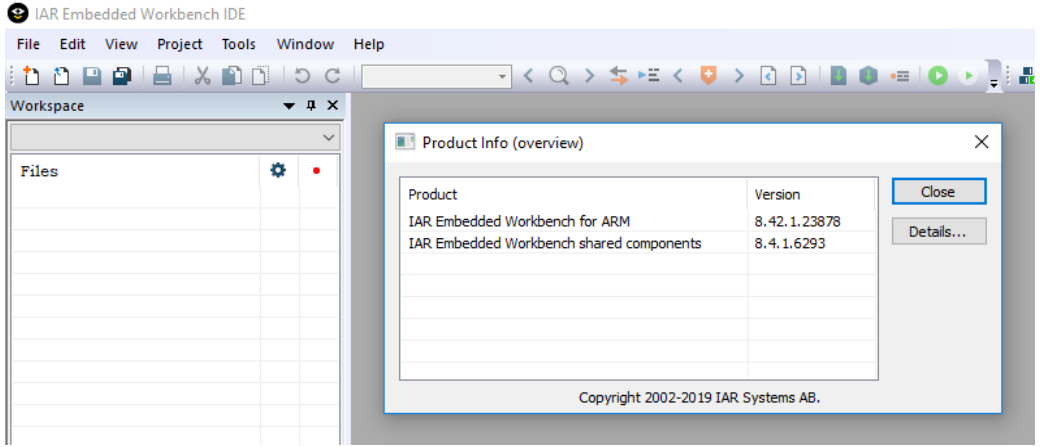
› Run/execute the program (either from flash or SRAM)





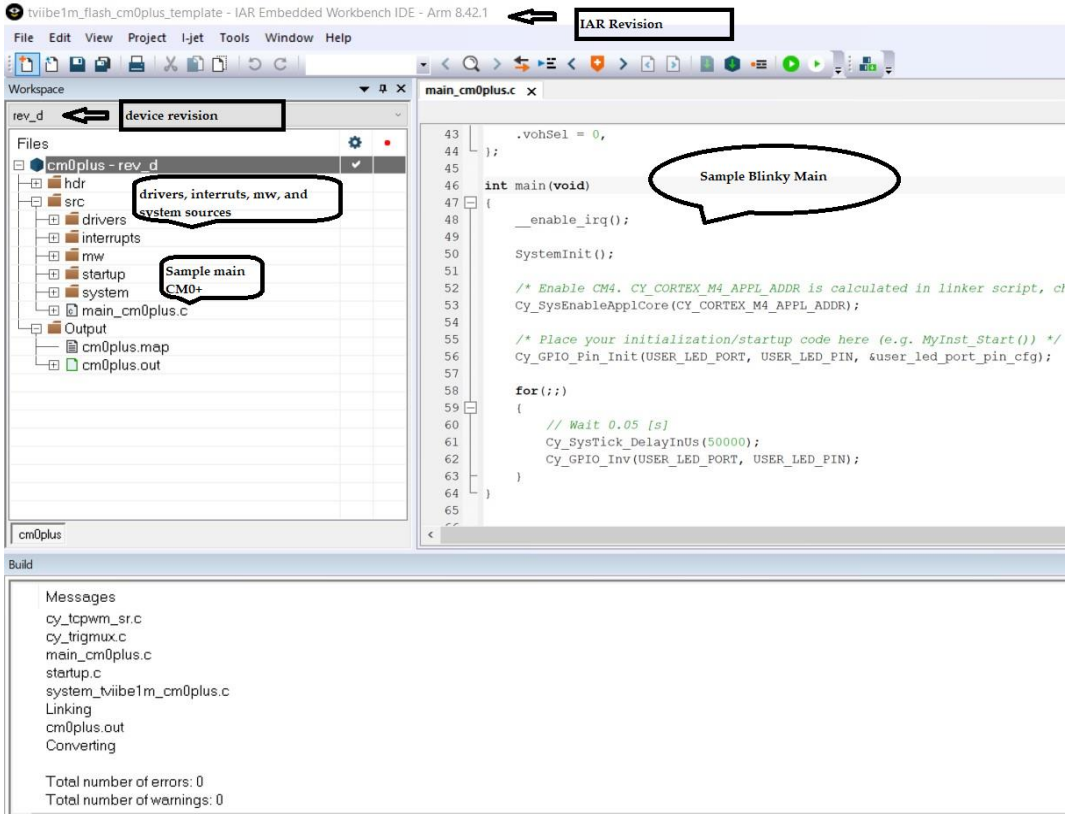
SDL IAR toolchain

- IAR Embedded Workbench for Arm®
  - Refer to SDL Readme file for the installer
  - Open the IDE
    - (Typical path: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.4\common\bin\IarIdePm.exe)
  - Ensure the licenses are properly set for the build process



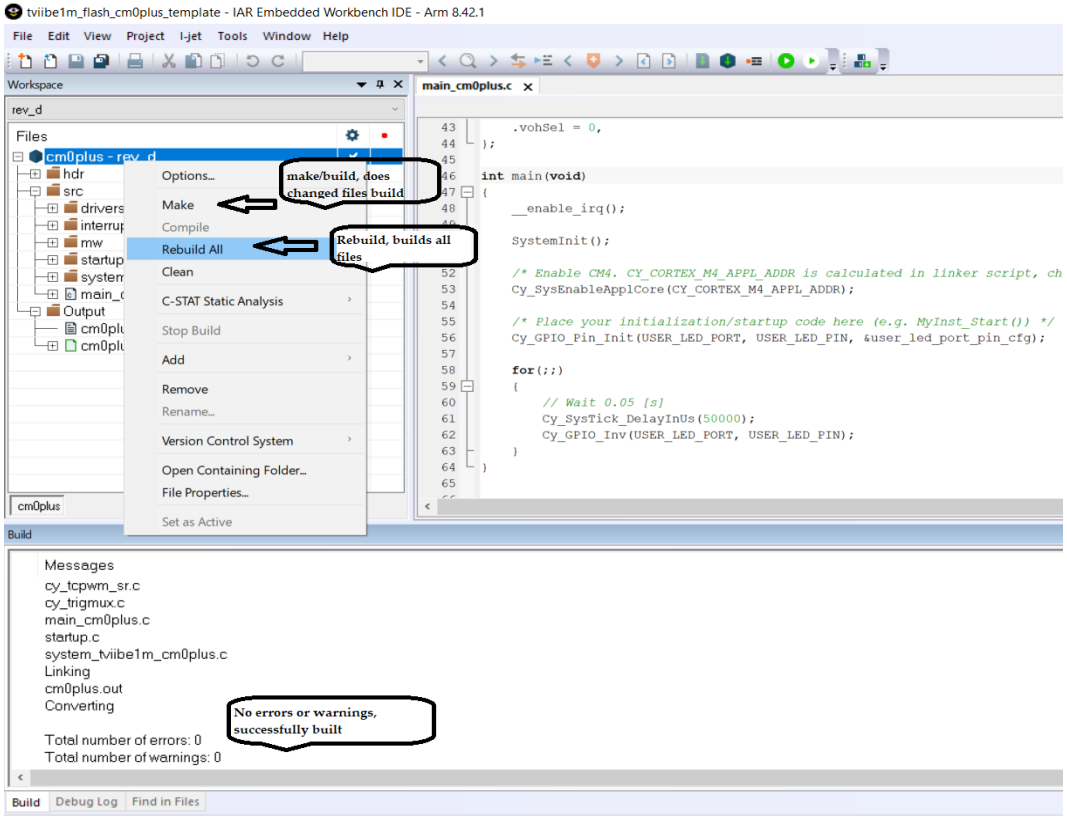
SDL IAR Toolchain – Flash workspace example

› Open workspace from the IDE



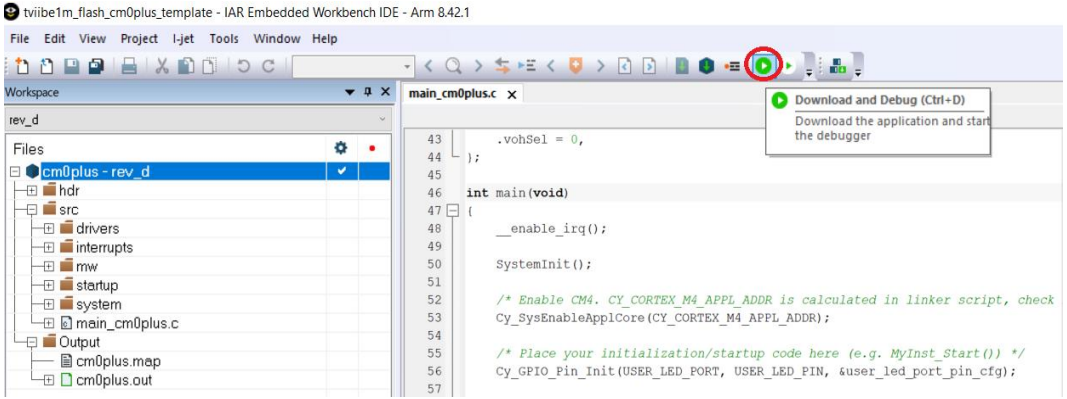
SDL IAR toolchain - Build

› Build or rebuild

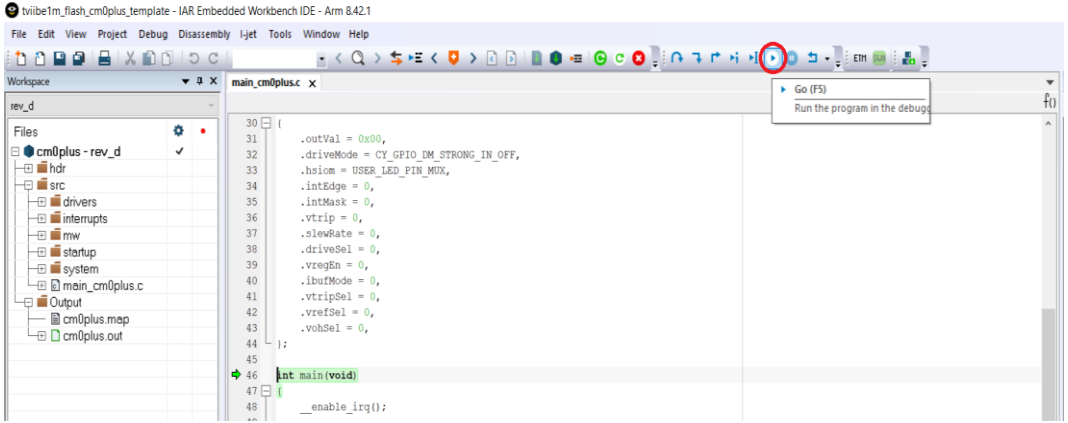


SDL IAR toolchain – Download, Run

› Download
  – Ensure power to the device is enabled

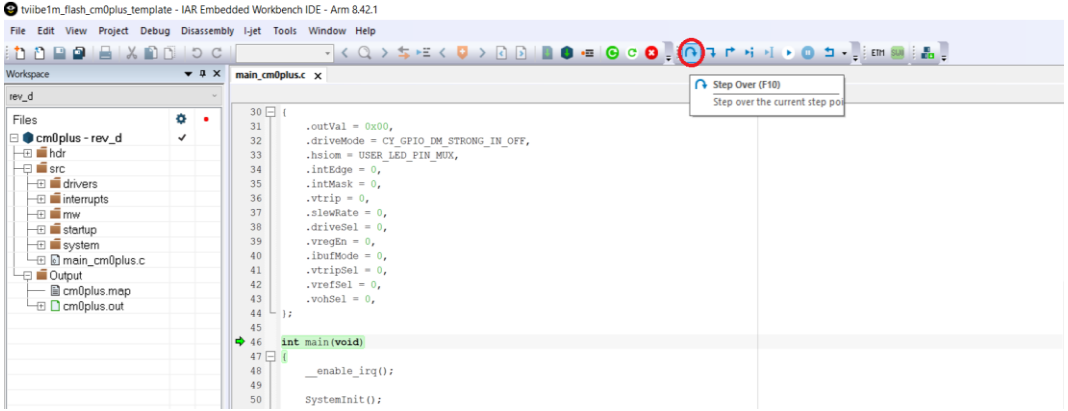


› Run
  – Observe the LED blink

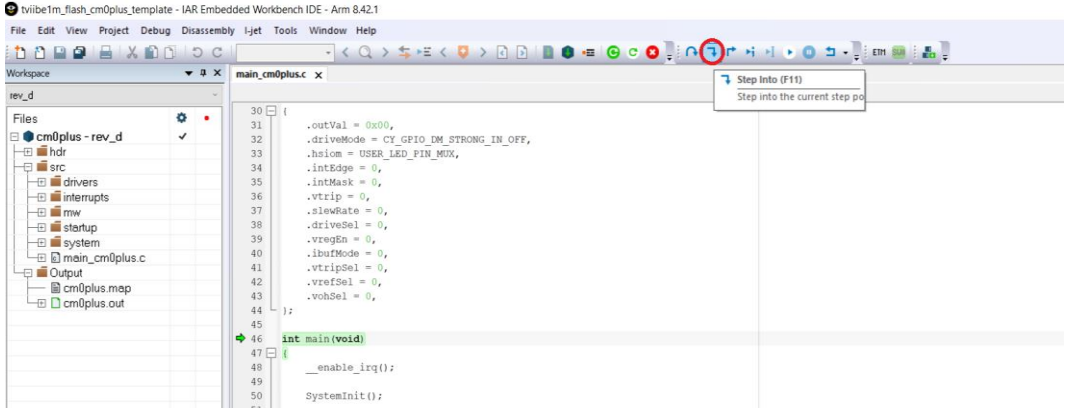


SDL IAR toolchain – Debug

› Debug
  – F10 – step over a function

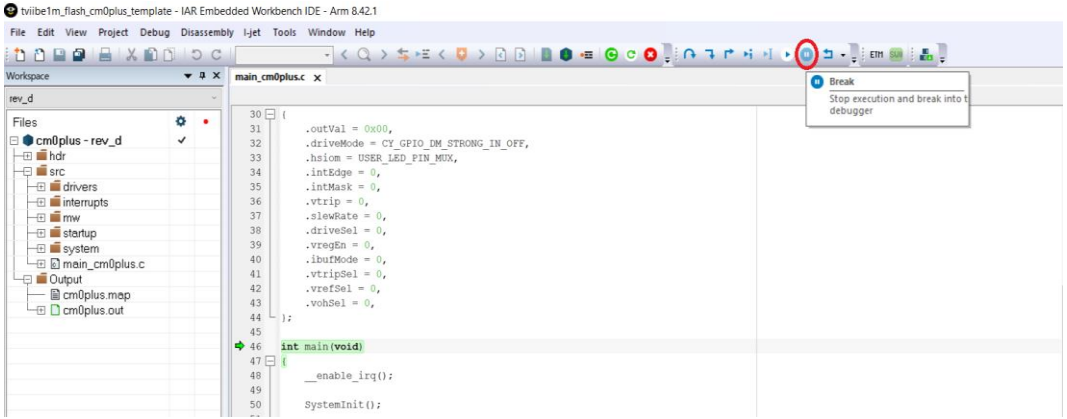


  – F11 – step into a function or a statement

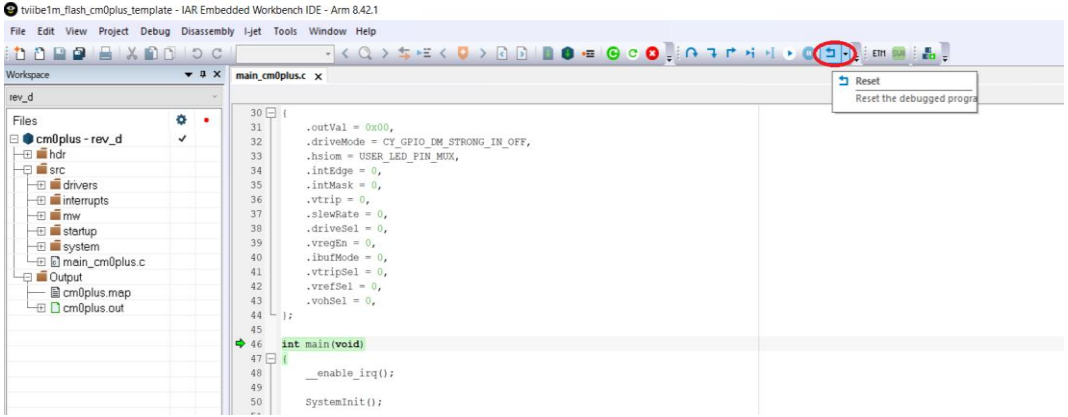


SDL IAR toolchain – Pause, Reset

› Pause
  – Random pause halts at any random instruction
  – Use breakpoints to methodically halt at a needed statement

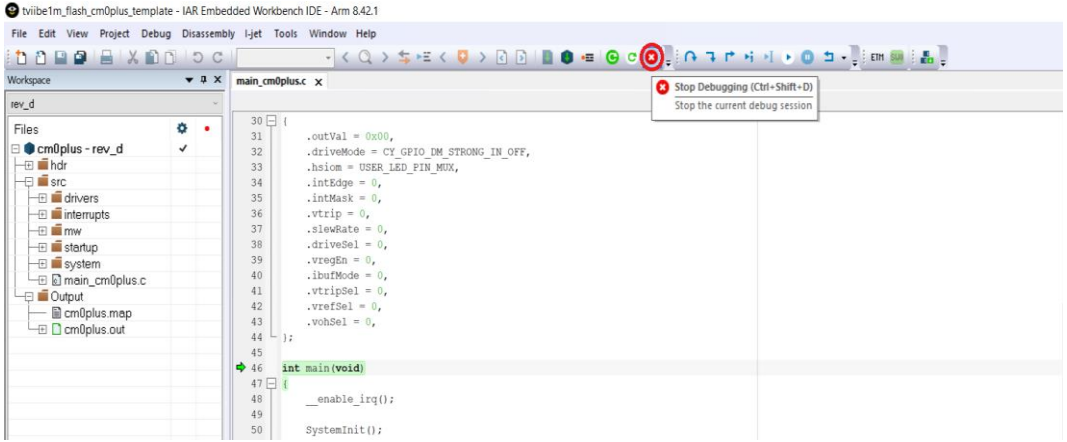


› Reset
  – Brings the control to the beginning of main



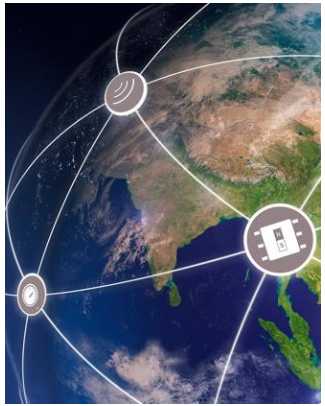
SDL IAR toolchain – Stop

› Stop the debugger



› Make sure you do this before the power to the CPU board is removed

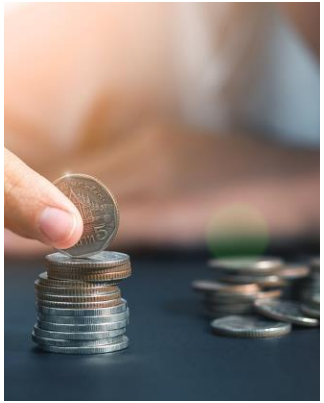
SDL examples usage



› Copy any example “main_**core**.c”[1] into the “\device\src” location

Note **1**: **core** may indicate cm0plus or cm4 or cm7_0 or cm7_1

› Rebuild the project in the IDE

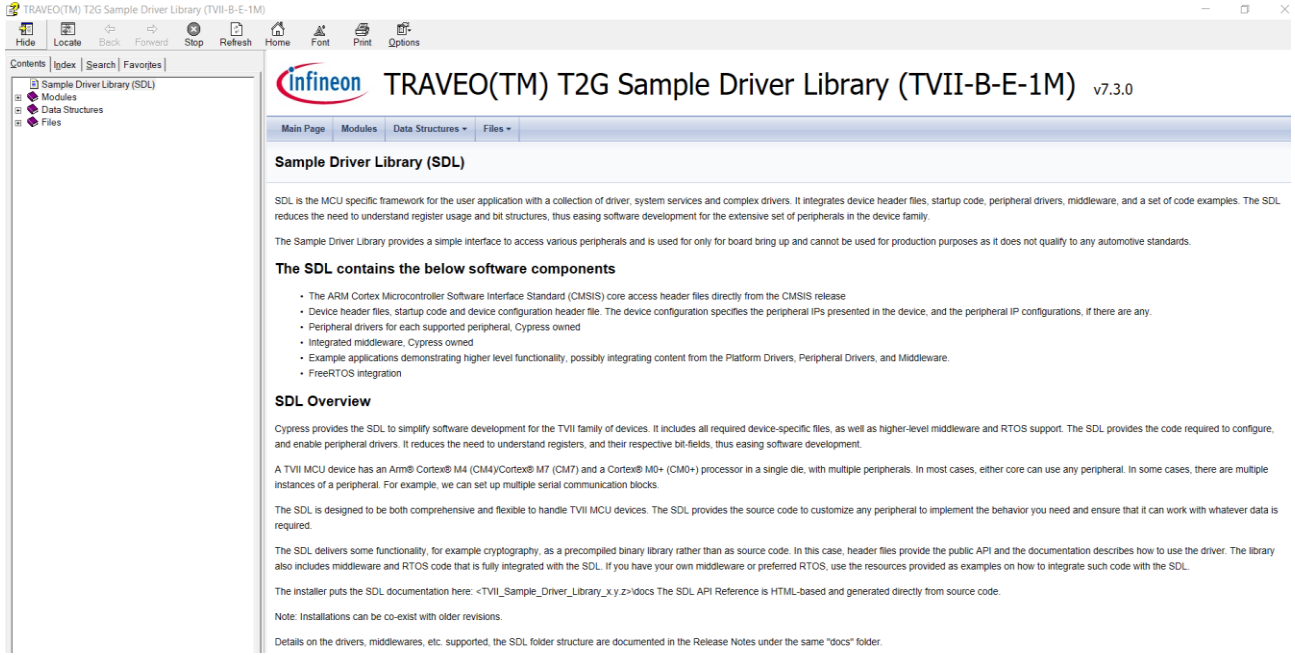


› Readme inside the example helps, understand the description, do necessary hardware connections if required

› Download and debug

SDL API documentation

› API documentation (T2G_Sample_Driver_Library_rev\docs\SDL_**device**.chm)



› Modules supported are listed on the left pane

External links for the tools

- IAR
  - Overview EWARM
    - https://www.youtube.com/watch?v=sMLS4S3-htI
  - EWARM Download
    - http://files.iar.com/ftp/pub/box/EWARM-CD-8421-23878.exe
  - Debuggers
    - https://www.iar.com/iar-embedded-workbench/add-ons-and-integrations/in-circuit-debugging-probes/
  - Licensing
    - https://www.iar.com/iar-embedded-workbench/#!?architecture=Arm&currentTab=editions-and-licensing
- Greenhills
  - Training
    - https://www.ghs.com/training.html
  - GHS Multi/Patch
    - https://www.ghs.com/products/MULTI_IDE.html
    - https://support.ghs.com/downloads/
  - Debuggers
    - https://www.ghs.com/products/probe.html
  - Licensing
    - https://support.ghs.com/licensing/

Technical support

- If you have any questions, our technical support team is happy to assist you
  - Create a support request on the Infineon Technical Support page

- You can also use the following support resources if you need quick assistance
  - Local Sales Office Locations

Part of your life. Part of tomorrow.

Revision History

Revision	ECN	Submission Date	Description of Change
**	6971522	09/25/2020	Initial release
*A	7450682	11/16/2021	Corrections and updates