Fault_Handling for KIT_T2G-B-H_LITE

Customer training workshop

Q3 2024

Scope of work

- This example demonstrates the HardFault functionality of TRAVEO™ T2G.
  It creates two types of hard faults on CM7_0:
  - UsageFault: divide by zeros
  - BusFault: Writes to read-only address

- **Device**
  - The TRAVEO™ T2G CYT4BF8CDS device is used in this code example

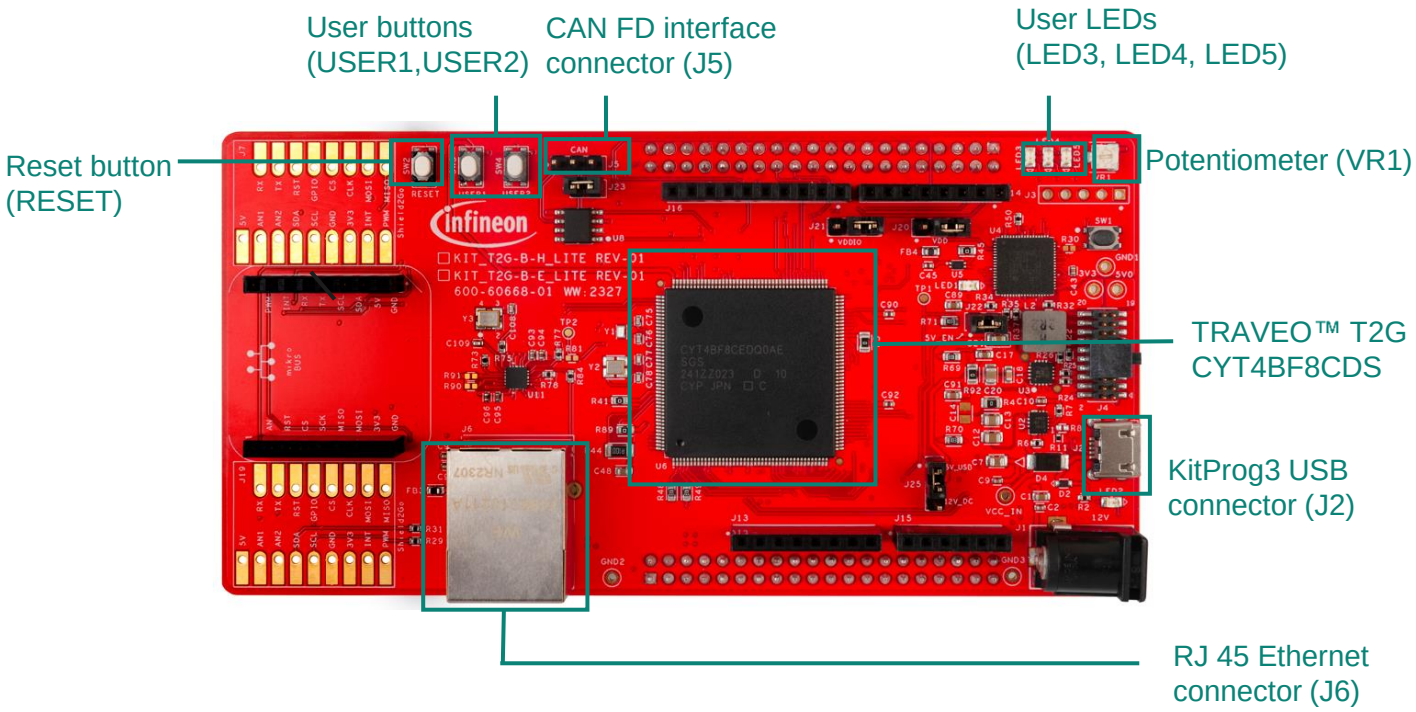
- **Board**
  - The TRAVEO™ T2G KIT_T2G-B-H_LITE board is used for testing

Introduction

- **TRAVEO™ T2G Arm® Cortex ® CPUs have the following features:**
  - Little-endian byte ordering for data, memory, and CPU registers
  - Each Cortex®-M7 has a 16 KB instruction cache and a 16 KB data cache with ECC support
  - Each Cortex®-M7 has 16-KB of instruction tightly coupled memory (ITCM) and 16-KB of data tightly coupled memory (DTCM) with ECC support TRAVEO™ T2G can access ITCM and DTCM with read and write wait ‘0’
  - Cortex®-M7 has a floating-point unit (FPU) with single and double precision that supports single-cycle digital signal processing (DSP) instructions and a memory protection unit (MPU)
  - Cortex®-M0+ has an MPU
  - Cortex®-M7 supports a subset of the Thumb instruction set
  - Cortex®-M0+ supports the Armv6-M Thumb instruction set
  - Both CPUs have nested vectored interrupt controllers (NVIC) for rapid and deterministic interrupt response
  - Both CPUs have extensive debug support
  - Inter-processor communication (IPC) hardware
  - Each CPU supports interrupts and exceptions on cores
    - Interrupts refer to events generated by peripherals external to the CPU
    - Exceptions refer to events generated by the CPU
  - Exception sources include Reset, Hard Fault, Non Maskable Interrupt (NMI), and so on
    - See the Interrupts section of the architecture reference manual for more details
  - Exceptions result in the current program flow being stopped and the exception handler being executed by the CPU. Upon completion of the exception handler, the CPU registers are restored to their original state using stack pop operations. Then, the CPU resumes the main code execution.

Hardware setup

- This code example has been developed for the KIT_T2G-B-H_LITE board
- Connect the PC to the board using the provided USB cable through the KitProg3 USB connector (J2)



Implementation

- This code example creates a BusFault or a UsageFault on CM7 according to the input from the terminal. The external reset is required to restart.

- **The following steps are used to configure the code example:**
  - STDOUT setting
  - System Handler Control and State Register (SHCSR) setting
  - Get the command from the terminal
  - Create BusFault or UsageFault
  - HardFault handling

Implementation (contd.)

- **STDOUT setting**
  - The ***cy_retarget_io_init ()*** function to use UART as STDOUT
    - Initialize P0.1 as UART TX, P0.0 as UART RX (these pins are connected to the KitProg3 COM port)
    - The serial port parameters change to 8N1 and 115200 baud

- **System Handler Control and State Register (SHCSR) setting**
  - Call the ***configure_fault_register()*** function to enable BusFault and UsageFault
    - BusFault is enabled if the BUSFAULTENA bit in SHCSR is set to ‘1’
    - UsageFault is enabled if the USGFAULTENA bit in SHCSR is set to’1’
    - The divide-by-zero UsageFault exception is enabled if the DIV_0_TRP bit in Configuration and Control Register (CCR) is set to ‘1’

- **Get the command from the terminal**
  - Use the ***cyhal_uart_getc()*** function to receive a character from the terminal
    - This is a blocking call, which waits till a character is received or till ***UART_TIMEOUT_MS*** has elapsed

Implementation (contd.)

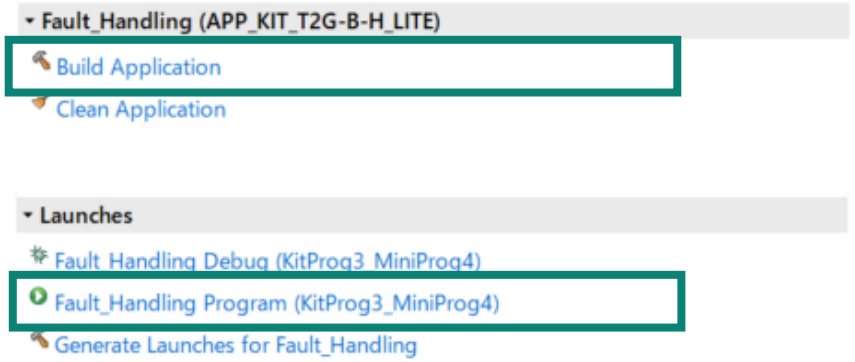
- **Create BusFault or UsageFault**
  - If the command is ‘b’ from the terminal, call the ***force_bus_fault()*** function to create BusFault
    - The ***force_bus_fault()*** function writes data to a read-only address and creates a BusFault
  - If the command is ‘u’ from the terminal, call the ***force_usage_fault()*** function to create UsageFault
    - The ***force_usage_fault()*** function divides by zeros and create a UsageFault

- **HardFault handling**
  - If BusFault or UsageFault are created, call the ***Cy_SysLib_ProcessingFault()*** function
    - This function is called from the HardFault handler function which is registered on the vector table
    - This function displays to terminal the Configurable Fault Status Register (CFSR), R0 to R3, R12, LR, PC, and PSR values
  - External reset is required to restart

Compiling and programming

1. Connect to power and USB cable
2. Use Eclipse IDE for ModusToolbox™ software for compiling and programming
3. For compilation:
   a. Select the target application project in the Project Explorer
   b. In the Quick Panel, scroll down, and click **“Build Application”** in Fault_Handling (KIT_T2G-B-H_LITE)
4. Open a terminal program (such as Tera Term) and select the KitProg3 COM port. Set the serial port parameters to **8N1** and **115200 baud**.
5. For programming:
   a. Select the target application project in the Project Explorer
   b. In the Quick Panel, scroll down, and click “**Fault Handling Program (KitProg3_MiniProg4)**” in the Launches

KitProg3 USB connector (J2)



Run and test

1. After programming, the application starts automatically. Confirm that the fault handling example title is displayed on the UART terminal as follows:
2. Press the ’u’ key to create a bus hard fault. The CM7 usage fault message appears as follows:
3. Press the reset button (RESET) to restart. The opening message appears in the terminal again as shown in the figure.

```
*****************  XMC7000 MCU: Fault Handling Basics *****************

Press 'u' key to create Usage Fault
Press 'b' key to create Bus Fault
Press the Reset button to start over after triggering a fault
```

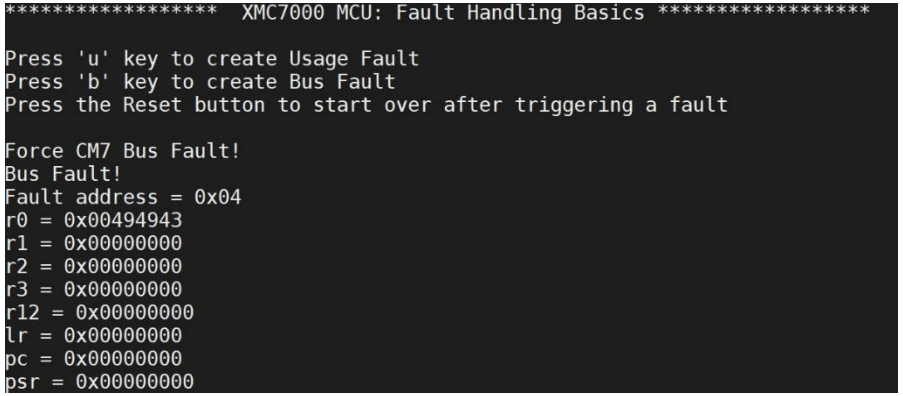

```
*****************  XMC7000 MCU: Fault Handling Basics *****************

Press 'u' key to create Usage Fault
Press 'b' key to create Bus Fault
Press the Reset button to start over after triggering a fault

Force CM7 Bus Fault!
Bus Fault!
Fault address = 0x04
r0 = 0x00494943
r1 = 0x00000000
r2 = 0x00000000
r3 = 0x00000000
r12 = 0x00000000
lr = 0x00000000
pc = 0x00000000
psr = 0x00000000
```

Run and test (contd.)

4. Press the ‘b’ key to create bus hard fault. The CM7 bus fault message appears as shown in the figure.
5. Press the reset button (RESET) to restart. The opening message appears in the terminal again as shown in the figure.



References

- **Datasheet**
  - CYT4BF TRAVEO™ T2G 32-bit Automotive MCU based on Arm® Cortex®- M7 dual
- **Architecture reference manual**
  - TRAVEO™ T2G Automotive MCU body controller high architecture reference manual
- **Registers reference manual**
  - TRAVEO™ T2G Automotive MCU: TVII-B-H-8M body controller high registers reference manual
- **PDL/HAL**
  - Peripheral driver library (PDL)
  - Hardware abstraction layer (HAL)
- **Training**
  - TRAVEO™ T2G training

Revision History

Revision	ECN	Submission Date	Description of Change
**	7840180	2022/11/24	Initial release
*A	8086105	2024/11/01	Replaced development board from KIT_T2G-B-H_EVK to KIT_T2G-B-H_LITE