Crash Handling

# Introduction

This document provides details on using the crash handler API to handle and debug error cases and interpret the crash printout.

# Crash Handler

To register a crash handler os\_set\_crash\_handler(void(\*crash\_cb)(void)) can be used. The function crash\_cb is called a crash. Note that the crash printout will occur before the registered functions are called.

In the following sample code, we register a function to the crash handler and then force an assert to demonstrate the functionality of the crash handler:

|  |
| --- |
| #include <kernel/os.h>  #include <assert.h>  /\* for print\_ver \*/  #include "utils.h"  static void \_\_irq  handle\_crash\_event()  {  os\_printf("Crash Handler...\n");  }  int  main(void)  {  print\_ver("Crash Handling Demo App", 1, 1);  //register crash handler  os\_set\_crash\_handler(handle\_crash\_event);  os\_printf("Assert in 5 seconds...\n");  os\_msleep(5000);  assert(0);  return 0;  } |



Figure : Crash printout with crash handler

As shown in Figure 1, the function registered using os\_set\_crash\_handler, gets called after the crash printout.

# Crash Output Debug

## Cortex M3 Crash

The main CPU is an ARM Cortex-M3 and this section illustrates the printout from a crash in the Cortex-M3.

Sample code is available at the following location of the SDK package: /examples/crash\_handling/crash\_debug.c.

Following is an example from a crash (due to a HardFault exception error in the application).

### Running the Application

Program crash\_debug.elf(sdk\_x.y\examples\crash\_handling\bin) using the Download tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down.
   2. ELF Input: Load the crash\_debug.elf by clicking on Select ELF File.
   3. Programming: Prog RAM or Prog Flash as per requirement.

For more details on using the Download tool, refer to the document: UG\_Download\_Tool.pdf (path: *sdk\_x.y/pc\_tools/Download\_Tool/doc*).

**Note**: x and y refer to the SDK release version. For example: sdk\_2.4/doc.

### Expected Output

|  |
| --- |
| UART:NWWWWWAE4 DWT comparators, range 0x8000  Build $Id: git-7e2fd6a94 $  app=gordon  flash: Gordon ready!  Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWAEBuild $Id: git-65f6c1f46 $  $App:git-46e2bea7  SDK Ver: sdk\_2.4  Crash Handling Demo App  Assert in 5 seconds...  ASSERTION FAILED: msg at 0x00104a30  OS\_UNEXPECTED\_EXCEPTION 0x6  R0=000b2ce0 R1=10000000 R2=e000e000 R3=00104a30  R4=04444444 R5=05555555 R6=06666666 R7=07777777  R8=08888888 R9=09999999 R10=0aaaaaaa R11=0bbbbbbb  R12=00104a30 SP=000b2c94 LR=00045129 PC=001049ec  xPSR=61000000 CONTROL=00000000 CFSR=00010000 BFAR=e000ed38  STACK:  0x000b2cd8: 03333333 000485d1 6e69616d 84be7200  0x000b2ce8: 001049c9 07f83301 00000002 000b2c94  0x000b2cf8: 000b24e8 a5631209 000b2d00 000b2d00  0x000b2d08: 000b2d08 000b2d08 0004000c 0004000c  0x000b2d18: 0004005c 000bed20 000b2d20 000b2d20  0x000b2d28: 00000000 00000000 dae34002 029b6eb4  0x000b2d38: d1b15013 8a789b3b 3f0f3230 94a5acbc  0x000b2d48: cf3cf34d 8e0ecab2 bedd6d63 7a29af77  0x000b2d58: fa528826 a9713fbe b3c784ab 56362dda  0x000b2d68: ecd9c852 0b30c634 f074edd3 eb42087e  0x000b2d78: e7ff7367 81482f15 d81e1b3a 6d0f25ae  0x000b2d88: f3d3f6e4 19c00255 58f9dc86 485cdbe7  0x000b2d98: bddbb93c 0c2c76ce f8792849 04c4aaba  0x000b2da8: f27d7027 41af8f33 f6a30800 eafac7c1  0x000b2db8: 565ee453 5b51121f 51a868d9 16edc158  0x000b2dc8: 73c78828 85d8eee5 52dd116d 522bd7de  Crash Handler... |

### Soft Faults

OS\_ERROR N indicates that there is a crash due to a fault detected by the OS, i.e. a soft fault. The following soft faults exist:

|  |  |
| --- | --- |
| **Soft Fault** | **Description** |
| 0x00 | Error in application |
| 0x01 | Heap is out of memory (actually printed explicitly as "OS\_ERROR: HEAP EXHAUSTED") |
| 0x02 | Failed to initialize virtual memory |
| 0xfa | Invalid argument in the os function call |
| 0xfb | An event is detected, for which there is no handler (callback) registered |
| 0xfc | OS internal error |
| 0xfd | Timer callback missing |
| 0xfe | Assertion failure (printed explicitly as "ASSERTION FAILED: …") |

Table : List of soft faults

### Exceptions

OS\_UNEXPECTED\_EXCEPTION M indicates that there is an exception that the OS cannot resolve.

For detailed information, please refer the following link: <https://developer.arm.com/documentation/dui0203/h/handling-cortex-m3-processor-exceptions/about-cortex-m3-processor-exceptions/exceptionnumbers>

Following are a list of valid exceptions:

|  |  |
| --- | --- |
| **Exceptions** | **Description** |
| 1 | Reset |
| 2 | NMI |
| 3 | HardFault |
| 4 | MemManage |
| 5 | BusFault |
| 6 | UsageFault |
| 11 | SVCall |
| 12 | Debug Monitor |
| 14 | PendSV |
| 15 | SysTick |
| 16 | External Interrupt(0) |

Table : List of exceptions

Registers in the crash dump are explained in the following link: <https://developer.arm.com/documentation/dui0552/a/the-cortex-m3-processor/programmers-model/core-registers>

## Co-processor Crash

There are three co-processors that handle Wi-Fi, Bluetooth, and Host Interface, and this section describes the printout from a crash in a co-processor.

Following is an example printout from a crash (due to a watchdog timeout in the Wi-Fi coprocessor):

|  |
| --- |
| COP0 EXCEPTION 0x8  COP0 REGDUMP:  000bbb38 00000008 00000004 00fc2a39  000bbb08 000bbb38 8000000c 00fc2a3b  00000000 00000000 00000000 00000000  00fc2a39 000a57d8 0005e152 0005dac0 |

COPx provides information on the coprocessor that crashed:

|  |  |
| --- | --- |
| **Co-processor** | **Description** |
| COP0 | Wi-Fi coprocessor |
| COP1 | Bluetooth coprocessor |
| COP2 | Host interface coprocessor |

Table : COPx descriptions

The exception code is a bitmask of the following bits:

|  |  |
| --- | --- |
| **Exception Code** | **Description** |
| bit0 | Idle (not an error) |
| bit1 | Invalid instruction |
| bit2 | Stopped via regwrite (not an error) |
| bit3 | Watchdog timeout |
| bit4 | Alignment fault |
| bit5 | Stack overflow |
| bit6 | Watchpoint |

Table : Exception code

**Note**: The REGDUMP for COPx uses an internal structure. For further debugging share the same with InnoPhase at the contact information provided in section 7.