Memfault

How to Debug HardFaults on ARM Cortex-M MCUs

Chris Coleman, Founder & CTO, Memfault
Webinar | April 22, 2021

The Speaker

Chris Coleman

Co-Founder & CTO, Memfault

- Previously a Firmware Engineer @ Sun Microsystems, Pebble, & Fitbit
- Enjoys debugging a good fault (perhaps too much)
- Can find my thoughts and content on Memfault's Interrupt blog (interrupt.memfault.com)

pebble Memfault fitbit

Cortex-M Faults!

- Last Quarter Alone! - 4.4 billion Cortex M's shipped

- Faults lead to:
  - Devices getting bricked
  - RMAs
  - Security Exploits

- Set a KPI for faults: < 0.01% faults / device / day

Agenda

1. HardFault Overview
2. Manual Debug
3. Scripting the Analysis
4. Using Memfault
5. Faults at Scale

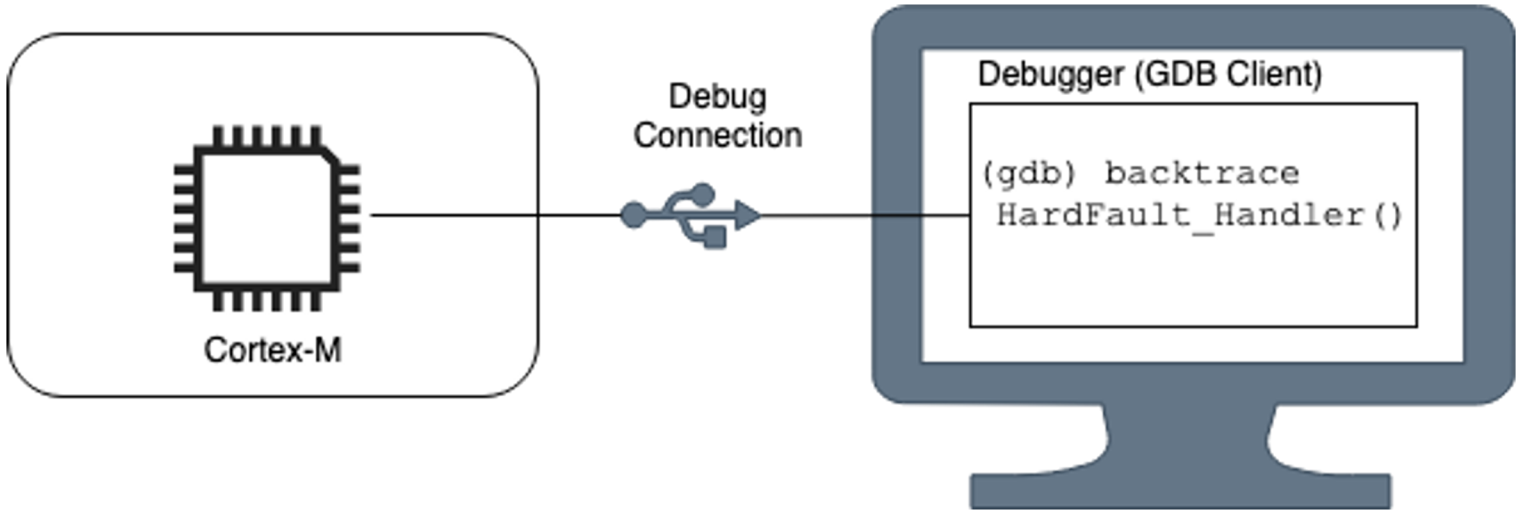
HardFault Overview

Fault

```
void buggy_function(void) {
  *(uint32_t *)0xbadcafe = 0x0;
}
```

```
void HardFault_Handler(void) {
    // … fault handling code ...
    NVIC_SystemReset();
}
```

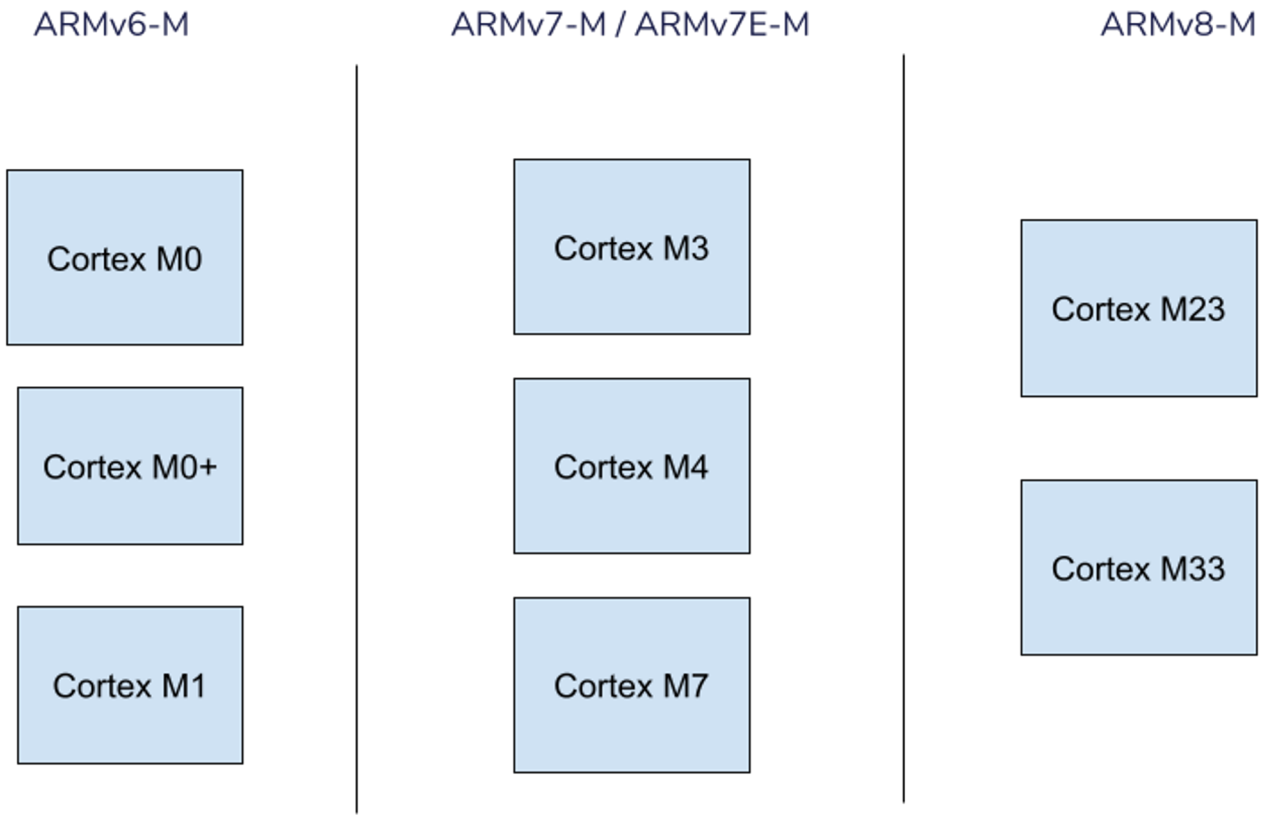
Attach Debugger



Local Debug Setup Recommendations

Reliable JTAG	i.e SEGGER JLink + JLinkGDBServer + GDB
Read/Write Registers/Memory	Debugger cable of updating register values
Scripting Support	Manually decoding registers is error prone!

Cortex-M Fault Architecture



- Principles apply to entire Cortex-M family!
- Faults cause an “Exception” handler to be invoked

Exception Causes

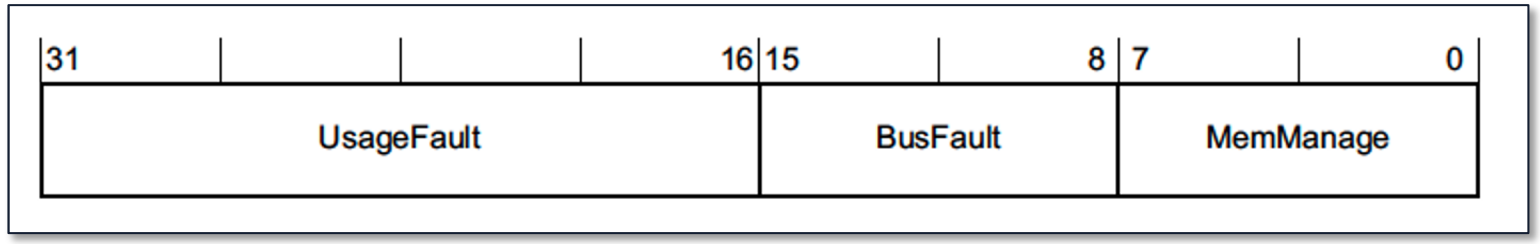
Type	Reason
UsageFault	Instruction Execution Errors Divide By Zero Unaligned Access
BusFault	Memory Access Errors Can be **Imprecise**!
MemManage	Memory Protection Unit Execute Never (XN) violation (0xE0000000 - 0xFFFFFFFF)

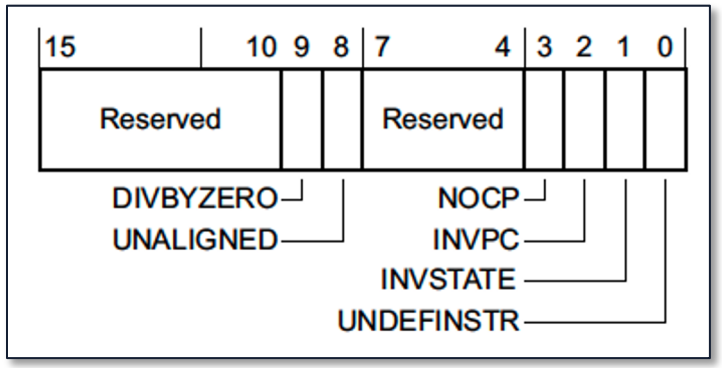
```c
void usagefault_example(void) {
  // udf 0xff
  __builtin_trap();
}
```

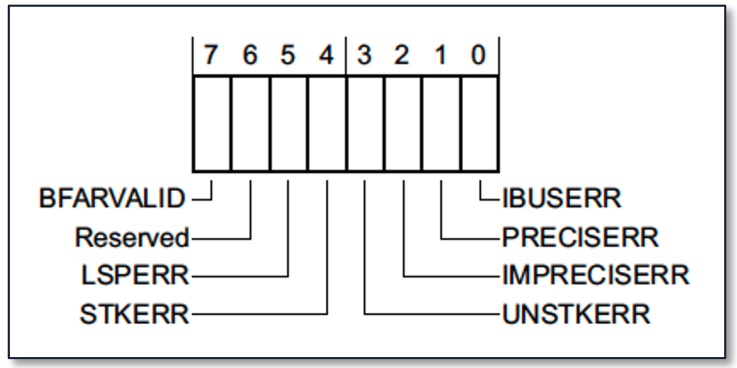
```c
void busfault_example(void) {
  *(uint32_t *)0xdead0000 = 0x20;
}
```

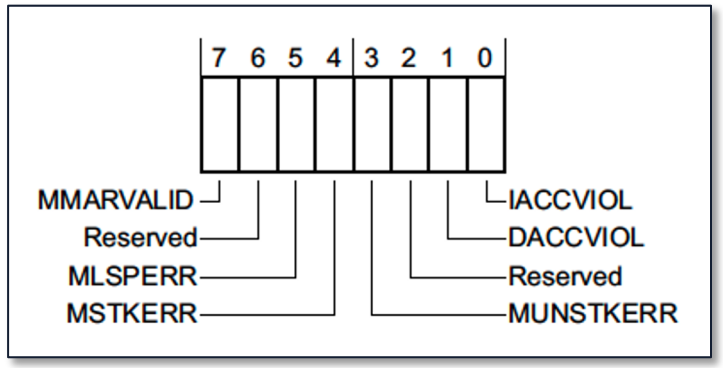
```c
void memmanage_example(void) {
  void (*func_in_xn_region)(void) = (void *)0xE0000000;

  func_in_xn_region();
}
```

Configurable Fault Status (0xE000ED28)









*** the CFSR is not present for ARMv6-M (Cortex-M0)**

CFSR Cheatsheet

Field	Mask	
BFARVALID	0x0000.8000	BusFault Address Register `p/x *(uint32_t)0xE000ED38`
MMARVALID	0x0000.0080	MemManage Fault Address Register `p/x *(uint32_t*)0xE000ED34`
STK	0x0000.3838	Stack Overflow / Corruption Likely
IMPRECISERR	0x0000.0400	pc doesn't point to faulting instruction
PRECISERR	0x0000.0200	pc points to faulting instruction

```
void buggy_function(void) {
  *(uint32_t *)0xbadcafe = 0x0;
}
```

```
(gdb) p/x *(uint32_t*)0xE000ED28
$7 = 0x400
```

Recovering Callstack

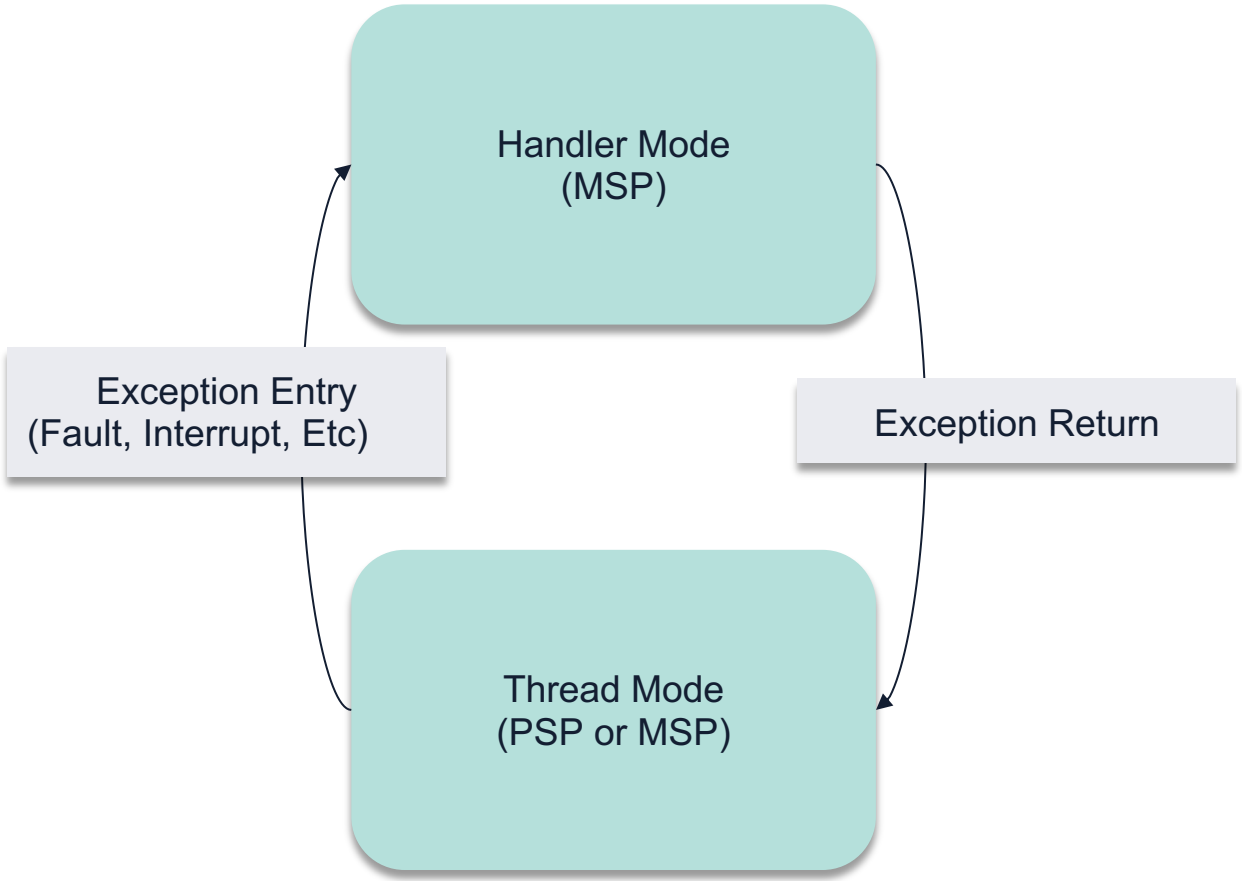
```
(gdb) backtrace
#0  HardFault_Handler () at ../src/main.c:96
#1  <signal handler called>
#2  0x00005a54 in prvPortStartFirstTask () at ./FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:270
#3  0x00005cfe in xPortStartScheduler () at ./FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:384
#4  0xf6103600 in ?? ()
```

HardFault Vector Catch

- Halts processor immediately at exception entry.
- Setting in “Debug Exception and Monitor Control Register”

```
(gdb) # DEMCR |= VC_HARDERR
(gdb) set *((uint32_t*)0xE000EDFC)|=0x0000400

...

Program received signal SIGTRAP, Trace/breakpoint trap.
HardFault_Handler () at ../src/main.c:96
(gdb) 96        __asm volatile(
```

Cortex-M Operation Modes



- Active Stack (SP)
  - Main Stack Pointer (MSP)
  - Process Stack Pointer (PSP)

```
(gdb) p/x $psp
$3 = 0x200009b8
(gdb) p/x $msp
$4 = 0x20004e20
(gdb) p/x $sp
$5 = 0x20004e20
```

Cortex-M Core Registers

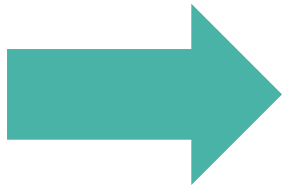
Caller	Registers	Role
“Caller Saved”	r0-r4	arguments
	r12	
“Callee Saved”	r4-r11	Temporaries
Special Registers	sp	Stack Pointer
	lr	Return Address
	pc	Program Counter

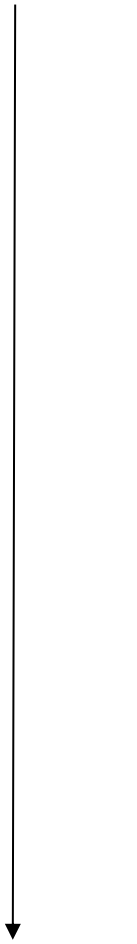
- 16 core registers
- Compilers follow “ABI”

```
void routine(void) {
  subroutine();
}
```

Exception Entry - Save Caller Registers

```
void buggy_function(void) {
  *(uint32_t *)0xbadcafe = 0x0;
}
```





		Present if original SP wasn't 8 byte aligned
		FPU Registers (if active) (18 registers, 72 bytes)
7	**0x1C**	**xPSR**
6	**0x18**	**ReturnAddress**
5	**0x14**	**LR**
4	0x10	R12
3	0x0C	R3
2	0x08	R2
1	0x04	R1
0	0x00	R0

Exception Entry - Update Registers

Hardware Register Updates

SP	MSP (if not already set)
LR	EXC_RETURN
PC	HardFault_Handler address

```
void HardFault_Handler(void) {



}
```

Exception Special Register Cheatsheet

Register	Bit	
xPSR	0..8	Exception Number or 0
	9*	Whether or not padding was used
EXC_RETURN (in lr)	2	1 = return to PSP 0 = return to MSP
	3	1 = thread mode 0 = handler mode
	4	0 = FPU registers saved
	28..31	Always 0xf

Exception number	Exception
1	Reset
2	NMI
3	HardFault
4	MemManage
5	BusFault
6	UsageFault

```
0x0800622a in
HardFault_Handler ()

(gdb) p/x $xpsr
$1 = 0x61000003

(gdb) p/x $lr
$3 = 0xfffffffd
```

*** Only present in xPSR pushed on stack**

Manual Debug

Recovering the Callstack

```
void buggy_function(void) {
   *(uint32_t *)0xbadcafe = 0x0;
}
```

```
(gdb) bt
#0  HardFault_Handler () at ../src/main.c:96
#1  <signal handler called>
#2  0x000059e4 in prvPortStartFirstTask () at ../third_party/freertos/FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:270
#3  0x00005c8e in xPortStartScheduler () at ../third_party/freertos/FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:384
#4  0xf6103610 in ?? ()
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
```

Recovering the callstack example, GDB

1 Recover location of exception frame

Register	Bit	
EXC_RETURN	2	1 = return to PSP 0 = return to MSP

```
(gdb) p/x $lr
$1 = 0xfffffffd

(gdb) set $exc_frame = ($lr & 0x4) ? $psp : $msp
```

Recovering the callstack example, GDB

2 Recover the length of exception frame

Register	Bit	
xPSR	9	Whether or not padding was used
EXC_RETURN	4	0 = FPU registers saved

```
set $stacked_xpsr = ((uint32_t *)$exc_frame)[7]
set $exc_frame_len = 32 +
    (($stacked_xpsr & (1 << 9)) ? 0x4 : 0x0) +
    (($lr & 0x10) ? 0 : 72)
```

Recovering the callstack example, GDB

3 Restore Original Register Values

Automatic Register Updates

SP	MSP (if not already set)
LR	EXC_RETURN
PC	HardFault_Handler address

```
(gdb) set $sp=($exc_frame + $exc_frame_len)
(gdb) set $lr=((uint32_t *)$exc_frame)[5]
(gdb) set $pc=((uint32_t *)$exc_frame)[6]
```

* sp, lr, & pc used for accurate unwind

Recovered Callstack

```c
void buggy_function(void) {
  *(uint32_t *)0xbadcafe = 0x0;
}
```

```
(gdb) backtrace
#0  HardFault_Handler () at ../src/main.c:96
#1  <signal handler called>
#2  0x00005a50 in prvPortStartFirstTask () at ../third_party/freertos/FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:270
#3  0x00005cfe in xPortStartScheduler () at ../third_party/freertos/FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:384
#4  0xf4103610 in ?? ()
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
(gdb) set $exc_frame = ($lr & 0x4) ? $psp : $msp
(gdb) set $stacked_xpsr = ((uint32_t *)$exc_frame)[7]
(gdb) set $exc_frame_len = 32 + (($stacked_xpsr & (1 << 9)) ? 0x4 : 0x0) + (($lr & 0x10) ? 0 : 72)
(gdb) set $sp=($exc_frame + $exc_frame_len)
(gdb) set $lr=((uint32_t *)$exc_frame)[5]
(gdb) set $pc=((uint32_t *)$exc_frame)[6]
(gdb) backtrace
#0  0x00003c82 in buggy_function () at ../third_party/memfault/memfault_platform_port.c:305
#1  0x00003c9a in test_buggy_function (argc=1, argv=0x200009f0 <ucHeap+2244>) at ../third_party/memfault/memfault_platform_port.c:310
#2  0x00003942 in prv_process () at ../third_party/memfault/memfault-firmware-sdk/components/demo/src/memfault_demo_shell.c:121
#3  0x000039c0 in memfault_demo_shell_receive_char (c=<optimized out>) at ../third_party/memfault/memfault-firmware-sdk/components/demo/
#4  0x000003b2 in prv_shell_thread (ctx=<optimized out>) at ../src/shell.c:36
#5  0x00005a6c in vPortEnableVFP () at ../third_party/freertos/FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:712
```

Scripting The Analysis

Scripting The Process, Enable Vector Catch

```
# ~/.gdbinit

define hook-continue
  # Enable automatic halt for HardFault Exception using
  # by setting DEMCR->VC_HARDERR
  set *((uint32_t*)0xE000EDFC)|=0x0000400
end
```

*Source: https://sourceware.org/gdb/onlinedocs/gdb/Extending-GDB.html#Extending-GDB

Scripting The Process, Analyze CFSR

```
# ~/.gdbinit

define fault_regs
  set $cfsr=*(uint32_t*)0xE000ED28
  printf "CFSR: 0x%08x\n", $cfsr
  if $cfsr & 0x8000
      printf "BusFault Address Valid: 0x%8x\n",
*(uint32_t)0xE000ED38
  end
end
```

*Source https://sourceware.org/gdb/onlinedocs/gdb/Extending-GDB.html#Extending-GDB

Scripting The Process, Recover Stack

```
# ~/.gdbinit

define unwind_stack
  # Set registers to their state prior to exception entry
  # to investigate origin of fault
  set $exc_frame = ($lr & 0x4) ? $psp : $msp
  set $stacked_xpsr = ((uint32_t *)$exc_frame)[7]
  set $exc_frame_len = 32 + ((($stacked_xpsr & (1 << 9)) ? 0x4 : 0x0) + (($lr & 0x10) ? 0 : 72)
  set $sp=($exc_frame + $exc_frame_len)
  set $lr=((uint32_t *)$exc_frame)[5]
  set $pc=((uint32_t *)$exc_frame)[6]
end
```

*Source https://sourceware.org/gdb/onlinedocs/gdb/Extending-GDB.html#Extending-GDB

Example

Memory Corruption

- FreeRTOS
- Cortex-M4 MCU (nRF52)

Memory Corruption Debug Strategy

1. Use BFAR & MFAR to locate access that failed

2. Check if address is near a memory boundary

1. Locate where the address references, i.e.
    - static or global
    - local variable
    - Heap

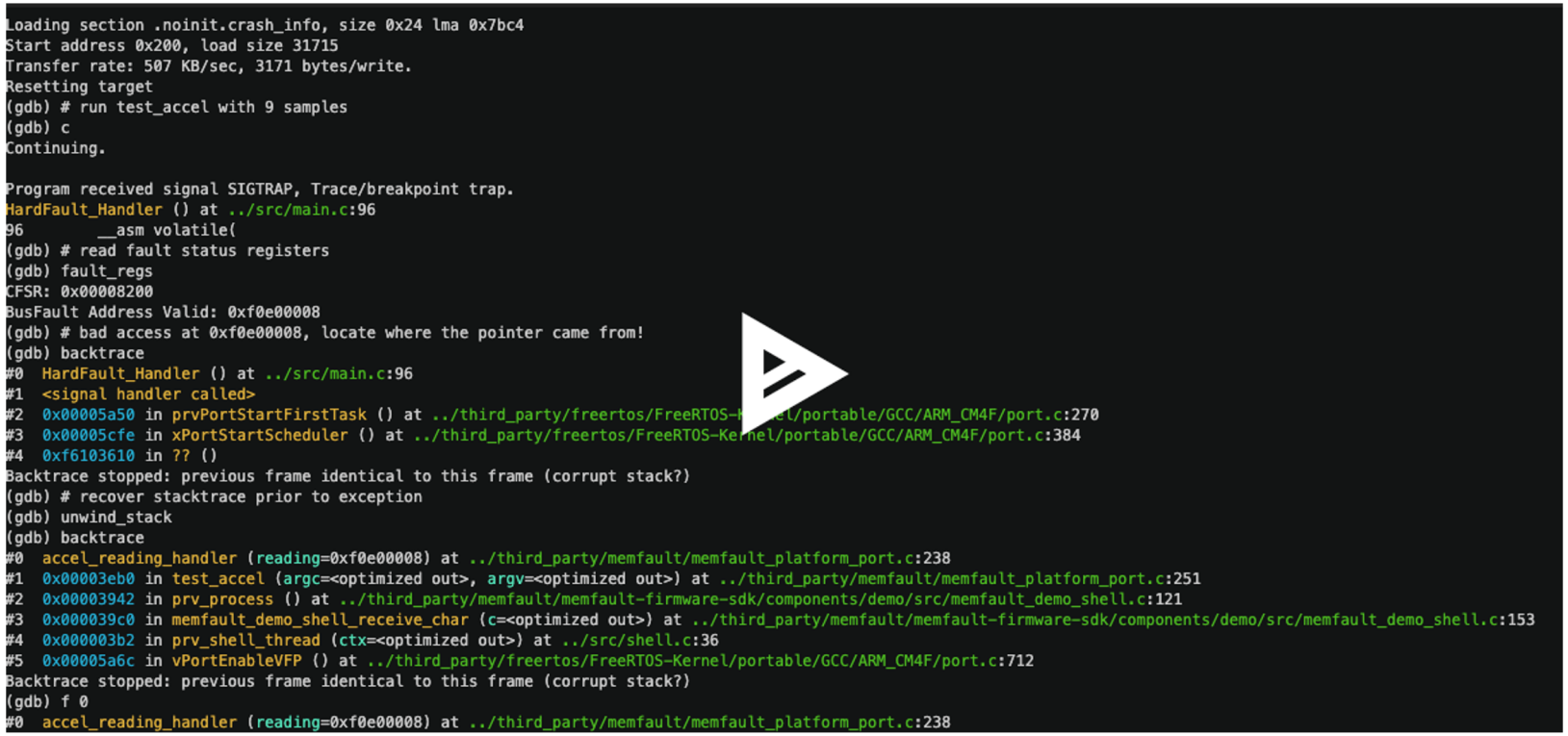
4. Locate allocations near address
    - `arm-none-eabi-nm  --numeric-sort <ELF>`

Example 1: Busfault

```c
static uint32_t s_accel_sample[8];

static struct {
  uint32_t *readings;
} s_accel_driver;

void accel_driver_init(void) {
  s_accel_driver.readings = &s_accel_sample[0];
}

void accel_reading_handler(uint32_t *reading) {
  debug_printf("Processing Accel Reading: 0x%x",
               *reading);
  // ... code processing data ...
}
```

```c
//! Generate fake accel samples and then process them
int test_accel(int argc, char *argv[]) {
  const int num_samples = atoi(argv[1]);

  for (int i = 0; i < num_samples; i++) {
    s_accel_driver.readings[i] = 0xF0E00000 | i;
  }

  for (int i = 0; i < num_samples; i++) {
    accel_reading_handler(&s_accel_driver.readings[i]);
  }

  return 0;
}
```

Example 1: Busfault



https://asciinema.org/a/exCWGkGIDVB3QmSQs03M87N2M

Debugging with Memfault

Example 1: Busfault

```
Program received signal SIGTRAP, Trace/breakpoint trap.
HardFault_Handler () at ../src/main.c:96
96          __asm volatile(
(gdb) memfault coredump -r 0x20000000 256000
One moment please, capturing memory...
Cortex-M4 detected
Collected MPU config
Capturing RAM @ 0x20000000 (256000 bytes)...
Captured RAM @ 0x20000000 (256000 bytes)
Uploading symbols...
Done!
Coredump uploaded successfully!
Once it has been processed, it will appear here:
https://app.memfault.com/organizations/memfault/projects/hardfault-example/issues?live
```

Example 2: Stack Overflow

```
int find_free_flash_page(int page) {
  if (page >= flash_max_pages()) {
    return -1;
  }

  uint8_t buf[256] = { 0x0 };
  flash_read_page(page, buf, sizeof(buf));

  if (!flash_page_erased(buf, sizeof(buf))) {
    find_free_flash_page(page + 1);
  }
  return page;
}
```

Example 2: Stackoverflow

```
(gdb) bt
#0  HardFault_Handler () at ../src/main.c:96
#1  <signal handler called>
#2  prv_fault_handling_assert (pc=pc@entry=0x20000258 <ucHeap+336>, lr=lr@entry=0x3bb8 <vApplicationStackOverflowHook+8>) at ../third_pa
y/memfault/memfault-firmware-sdk/components/panics/src/memfault_fault_handling_arm.c:537
#3  0x0000318c in memfault_fault_handling_assert (pc=0x20000258 <ucHeap+336>, lr=0x3bb8 <vApplicationStackOverflowHook+8>, extra=18213,
tra@entry=0) at ../third_party/memfault/memfault-firmware-sdk/components/panics/src/memfault_fault_handling_arm.c:545
#4  0x00003bbc in vApplicationStackOverflowHook (xTask=<optimized out>, pcTaskName=<optimized out>) at ../third_party/memfault/memfault-
rmware-sdk/ports/freertos/src/memfault_panics_freertos.c:24
#5  0x00004724 in vTaskSwitchContext () at ../third_party/freertos/FreeRTOS-Kernel/tasks.c:3052
#6  0x00005b44 in PendSV_Handler () at ../third_party/freertos/FreeRTOS-Kernel/portable/GCC/ARM_CM4F/port.c:441
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
(gdb) memfault coredump -r 0x20000000 256000
One moment please, capturing memory...
Cortex-M4 detected
Collected MPU config
Capturing RAM @ 0x20000000 (256000 bytes)...
Captured RAM @ 0x20000000 (256000 bytes)
Symbols have already been uploaded, skipping!
Coredump uploaded successfully!
Once it has been processed, it will appear here:
https://app.memfault.com/organizations/memfault/projects/hardfault-example/issues?live
(gdb)
```

```
~/dev/memfault (JLinkRTTClient)
mflt>
mflt>
mflt>
mflt>
mflt>
mflt> test_flash
```

Faults at Scale

Some Faults will only happen in the field!

“a third of all software faults take more than 5000 execution-years to manifest themselves.”

*Source: http://www.ganssle.com/tem/tem417.html

Remote Fault Recovery

1. Store Fault Info on Device
2. Push Collected Info to Cloud
3. Process and Analyze Data

Implement Fault Handler

```
void hardfault_handler_c(sContextStateFrame *exc_frame) {
    // TODO: Fill me in
}

__attribute__((naked))
void HardFault_Handler(void) {
  __asm volatile(
       "tst lr, #4 \n"
       "ite eq \n"
       "mrseq r0, msp \n"
       "mrsne r0, psp \n"
       "b hardfault_handler_c \n");
}
```

Allocate Storage Area

```
typedef struct __attribute__((packed)) ContextStateFrame {
  uint32_t r0;
  uint32_t r1;
  uint32_t r2;
  uint32_t r3;
  uint32_t r12;
  uint32_t lr;
  uint32_t return_address;
  uint32_t xpsr;
} sContextStateFrame;

#define CRASH_INFO_MAGIC 0x48535243

typedef struct {
  uint32_t magic;
  sContextStateFrame frame;
} sCrashInfo;

static sCrashInfo s_last_crash_info
    __attribute__((section(".noinit.crash_info")));
```

Save Exception Frame & Reset MCU

```
void hardfault_handler_c(sContextStateFrame *exc_frame) {
  s_last_crash_info.frame = *exc_frame;
  s_last_crash_info.magic = CRASH_INFO_MAGIC;

  NVIC_SystemReset();
}
```

Push Collected Info To Cloud

```
void check_crash_info(void) {
  if (s_last_crash_info.magic != CRASH_INFO_MAGIC) {
    return;
  }

  uint32_t pc = s_last_crash_info.frame.return_address;
  uint32_t lr = s_last_crash_info.frame.lr;

  MEMFAULT_LOG_INFO("Fault On Last Reset");
  MEMFAULT_LOG_INFO("PC=0x%x", pc);
  MEMFAULT_LOG_INFO("LR=0x%x", lr);

  s_last_crash_info.magic = 0x0;  // Clear Info

  platform_send_fault_data(pc, lr);
}
```

Process And Analyze Data

5 Decode with addr2line

```
arm-none-eabi-addr2line --pretty-print --functions --addresses 0x3c82 -e memfault.elf
0x00003c82: buggy_function at ./third_party/memfault/memfault_platform_port.c:306

arm-none-eabi-addr2line --pretty-print --functions --addresses 0x3c9a -e memfault.elf
0x00003c9a: test_buggy_function at ./third_party/memfault/memfault_platform_port.c:311
```

6 Store in Database for Analysis

Memfault for Fault Debugging

- ✓ C-SDK with Fault Handler Provided
- ✓ User configurable collection Regions
- ✓ CFSR, BFAR, MMFAR analyzed
- ✓ Full stacktrace & variable recovery
- ✓ Automated RTOS Analysis
- ✓ Issue De-duplication

… and more!

Trusted By

logitech

proxy

WHOOP®

GEOTAB®

PANIC

skip

Memfault

Connect with Chris

https://www.linkedin.com/in/christopher-coleman-812aa06b/

chris@memfault.com

https://interrupt.memfault.com

Interrupt Blog Resources

- How to debug a HardFault on an ARM Cortex-M MCU
- Fix Bugs and Secure Firmware with the MPU
- A Practical guide to ARM Cortex-M Exception Handling
- Automate Debugging with GDB Python API

Memfault