Embedded Systems Reverse Engineering

// WEEK 07

Constants in Embedded Systems:
Debugging and Hacking Constants
w/ 1602 LCD I2C Basics

George Mason University

RP2350 // ARM Cortex-M33

#define vs const

Preprocessor Macros vs Constant Variables

`#define FAV_NUM 42`

Preprocessor text replacement
Happens BEFORE compilation
No memory allocated
Cannot take address (&)

In Binary:

`movs r1, #42 @ 0x2a`

16-bit Thumb instruction
Value embedded as immediate
Compiler sees only "42"

`const int OTHER_FAV_NUM=1337`

Creates real variable
Theoretically in .rodata
Has an address (if needed)
Type-checked by compiler

In Binary:

`movw r1, #1337 @ 0x539`

32-bit Thumb-2 instruction
Also embedded as immediate!
Compiler optimized it away

KEY INSIGHT: Both ended up as instruction immediates!

The compiler saw &OTHER_FAV_NUM is never used, so it optimized const the same way as #define -- no memory load needed.

Lesson: const is a source-level concept -- not guaranteed in binary

I2C Protocol

Two-Wire Serial Communication

What is I2C?

Two-wire serial protocol

SDA = Serial Data

SCL = Serial Clock

Open-drain with pull-up resistors

I2C Bus

GPIO 2 = SDA, GPIO 3 = SCL

Pull-ups hold lines HIGH

Common I2C Addresses (7-bit)

0x27	LCD
0x3F	LCD Alt
0x48	Sensor
0x50	EEPROM

I2C Transaction Flow

START --> **Address** --> **ACK** --> **Data** --> **ACK** --> **STOP**

Master sends START, then 7-bit address + R/W bit

Slave responds with ACK, then data bytes follow

C Structs & typedef

Grouping Related Data in C

Struct Definition

```
typedef struct {
  i2c_hw_t *hw;
  bool restart_on_next;
} i2c_inst_t;
```

typedef creates an alias
so we can write: i2c_inst_t var;
instead of: struct { ... } var;

Memory Layout

i2c_inst_t at 0x2000062C

Offset 0x00 **hw** **= 0x40098000**
i2c_hw_t* (4 bytes)

Offset 0x04 **restart_on_next**
= 0x00 (false) bool (1 byte)

Total struct size: 8 bytes

hw points to I2C1 registers

Forward Declaration

```
struct i2c_inst; // tells compiler: this type exists, define later
```

Why Structs Matter in RE

GDB shows raw memory -- you must recognize struct layouts
x/2wx 0x2000062c shows: 0x40098000 0x00000000

Pico SDK Macro Chain

From I2C_PORT to Hardware Registers

Macro Expansion Chain

I2C_PORT --> **i2c1** --> **&i2c1_inst**

`#define I2C_PORT i2c1` — `#define i2c1 (&i2c1_inst)` — Address of global struct

Struct Contents at 0x2000062C

```
i2c_inst_t i2c1_inst = {
    .hw = (i2c_hw_t *)0x40098000,
    .restart_on_next = false
};
```

Hardware Register Access

`i2c1_inst.hw` --> **i2c1_hw** --> **(i2c_hw_t*)0x40098000**

I2C1_BASE = 0x40098000 I2C0_BASE = 0x40090000

Direct memory-mapped I/O to RP2350 peripheral

FULL CHAIN: **I2C_PORT** --> **i2c1** --> **&i2c1_inst** --> **0x40098000**

Macro --> Macro --> Struct pointer --> HW register base

Source Code

0x0017_constants.c

```c
//--- Defines and Constants ---
#define FAV_NUM 42
#define I2C_PORT i2c1
#define I2C_SDA_PIN 2
#define I2C_SCL_PIN 3
const int OTHER_FAV_NUM = 1337;

//--- Main Loop ---
lcd_set_cursor(0, 0);
lcd_puts("Reverse");
lcd_set_cursor(1, 0);
lcd_puts("Engineering");

//--- Serial Output Loop ---
printf("FAV_NUM: %d\r\n", FAV_NUM);
printf("OTHER_FAV_NUM: %d\r\n", OTHER_FAV_NUM);
```

LCD Output

Line 0: "Reverse"
Line 1: "Engineering"

Serial Output

```
FAV_NUM: 42
OTHER_FAV_NUM: 1337
```

GDB Analysis

Disassembly of main() at 0x10000234

Key Instructions from x/54i 0x10000234

```
push {r3, lr}                   // save return addr
bl stdio_init_all               // init serial
ldr r1, [pc, #104]              // r1 = 100000 (baud)
ldr r0, [pc, #104]              // r0 = &i2c1_inst
bl i2c_init                     // init I2C at 100kHz
movs r0, #2                     // GPIO 2 (SDA)
bl gpio_set_function            // set pin to I2C
movs r1, #39                    // 0x27 = LCD addr
bl lcd_i2c_init                 // init LCD device
b.n 0x1000028e                  // infinite loop start
... AAPCS: r0-r3 = first 4 args, r0 = return value
```

Literal Pool at 0x100002A4

0x000186A0	I2C baudrate (100000)	0x10003EFC	"FAV_NUM: %d\r\n"
0x2000062C	&i2c1_inst struct in RAM	0x10003F0C	"OTHER_FAV_NUM: %d\r\n"
0x10003EE8	"Reverse" string in flash		
0x10003EF0	"Engineering" string in flash		

Instruction Encoding

movs (16-bit Thumb) vs movw (32-bit Thumb-2)

movs r1, #42 (FAV_NUM)

At address 0x1000028E

Bytes: 2A 21

2A = immediate value (42)

21 = opcode (movs r1)

16-bit Thumb instruction

Fits values 0-255 in 8 bits

File offset: 0x28E

movw r1, #1337 (OTHER_FAV)

At address 0x10000296

Bytes: 40 F2 39 51

40 F2 = opcode (first halfword)

39 = imm8 (lower 8 bits)

51 = dest reg + upper imm

32-bit Thumb-2 instruction

File offset: 0x296

movw Byte Layout (40 F2 39 51)

40 F2	39	51	imm16 = 0x539
Opcode + upper imm	imm8 (lower 8 bits)	Dest reg (r1) + bits	= 1337 decimal

Why movw instead of movs?

1337 > 255 -- does not fit in 8-bit movs immediate

movw encodes 0-65535 in 32-bit instruction

I2C Struct in Memory

Examining i2c1_inst at 0x2000062C

GDB Memory Dump

```
x/2wx 0x2000062c:        0x40098000        0x00000000
```

i2c_inst_t Struct Layout

```
Offset 0x00 | 4 bytes
i2c_hw_t *hw      = 0x40098000
```

--> **I2C1 HW Registers**
Base: 0x40098000 (MMIO)

```
Offset 0x04 | 1 byte
bool restart_on_next    = false
```

I2C0 base = 0x40090000
I2C1 base = 0x40098000

String Literals in Flash (.rodata)

```
x/s 0x10003ee8:    "Reverse"
```

```
x/s 0x10003ef0:    "Engineering"
```

Stored consecutively in .rodata (flash)
These addresses are targets for patching

Hacking the Binary

Patching LCD Text: "Reverse" --> "Exploit"

File Offset Formula

file_offset = address - 0x10000000 Binary loaded at 0x10000000

Hack: Change LCD String

Address 0x10003EE8 --> File offset 0x3EE8

Original:

52 65 76 65 72 73 65 00 "Reverse"

Patched:

45 78 70 6C 6F 69 74 00 "Exploit"

Same length (7 chars) -- null terminator stays

Flash the Hacked Binary

```
python uf2conv.py build\patched.bin
```

1. Save patched .bin file
2. Convert to .uf2 format
3. Hold BOOTSEL, plug in Pico
4. Drag hacked.uf2 to drive

LCD now shows: "Exploit" instead of "Reverse" **No source code needed!**

I2C & Macro Exploitation

Constants, I2C, Structs, and Hacking

Key Concepts

#define	Text replacement, no memory
const	Variable in .rodata (maybe)
I2C	Two-wire: SDA + SCL
struct	Groups related data fields
typedef	Creates type alias
AAPCS	r0-r3 args, r0 return
movs	16-bit, imm 0-255
movw	32-bit, imm 0-65535
Literal Pool	Large consts after code

Key Addresses

0x10000234	main() entry
0x1000028E	FAV_NUM (movs)
0x10000296	OTHER_FAV_NUM (movw)
0x10003EE8	"Reverse" string
0x10003EF0	"Engineering" string
0x40098000	I2C1 HW base
0x2000062C	i2c1_inst struct

file_offset = addr - 0x10000000
String patches must be same length

Macro Chain

I2C_PORT --> **i2c1** --> **&i2c1_inst** --> **0x40098000**

Binary Hack Result

LCD: "Reverse" --> **"Exploit"** Patched at 0x3EE8

Compiler may optimize const same as #define

TAKEAWAY: const is a source-level concept. In binary, everything can change!