CPE 325: Intro to Embedded Computer System

Lab 10

Reverse Engineering

Submitted by: Hannah Kelley

Date of Experiment: November 10, 2025

Report Deadline: November 17, 2025

Demonstration Deadline: November 17, 2025

Theory

Topic 1: ELF File Components

The Executable and Linkable File (ELF) format is a standardized way of creating executable files, object files, shared libraries, and core dumps. The ELF file always starts with a header that details the architecture of the device used and the entry point immediately followed by a program header mapping sections to their corresponding memory addresses. Key sections in the ELF file including ".text" for executable code, ".data" for initialized variables, and ".bss" for uninitialized variables. Other important features include ".rodata" sections for read-only data and ".stack" and ".heap" for runtime memory management. The symbol table in the ELF file provides addresses for every function and variable used in the program, which can be very beneficial for debugging purposes. Overall, this structure helps optimize memory usage and provides a structured format that is easy for the programmer to understand and use.

Topic 2: Naken Utility

The Naken Utility is an open-source tool designed for working with embedded assembly code on microcontrollers such as the MSP430. It includes features for assembling source code into machine code, disassembling binaries back into readable assembly, and simulating code execution. The built-in simulator allows users to observe register values, memory contents, and instruction flow in real time, which is especially useful for testing and debugging programs without needing physical hardware. In addition to the MSP430, the tool supports several other architectures, making it a flexible option for low-level embedded development. Overall, the Naken Utility provides an efficient environment for writing, analyzing, and debugging assembly code in educational and professional embedded systems projects.

Topic 3: MSP430 Flasher

The MSP430 Flasher is a command-line tool developed by Texas Instruments for programming MSP430 microcontrollers. It enables users to flash firmware, erase memory, and verify program integrity directly from a computer terminal. The tool communicates with MSP430 devices via interfaces such as JTAG and Spy-Bi-Wire, providing a reliable connection for uploading and managing firmware. Because it supports automation through scripting, the MSP430 Flasher is particularly useful for production environments and batch programming. Its lightweight design and compatibility with multiple operating systems make it an essential utility for embedded developers who need to efficiently program and maintain MSP430-based systems.

Results & Observation

Problem 1:

Report Questions:

The following questions were embedded in the tutorial for lab 10.

1. **What insights can you glean from your analysis?**
   ELF commands from the TI compiler have subtle differences when compared to those generated from the GNU compiler. The differences result in variations in the .out files. The resultant files were similar, yet differed in the section headers, flags, number of section headers, section header table index, and the magic number. The commands used in the tutorial gave similar results no matter which compiler was used.

2. **Can you find what symbol is associated with the address 0x3272?**
   The symbol associated with memory address 0x3272 (in decimal 12,914) is “memset”. This answer can be found thanks to the symbol table found on page 16 of the tutorial.

```
217: 00003272    20 FUNC    GLOBAL DEFAULT   11 memset
```

Figure 1: Screenshot of Symbol Table Entry 217

Problem 2:

Program Description:

The PWM code I submitted for Lab 6 was turned into a hex file using the Hex Utility Linker in Code Composer Studio.

```
@4400
0F 12 0E 12 E2 B3 01 02 2C 20 E2 B3 00 02 29 20
82 93 04 24 06 20 92 43 04 24 82 43 06 24 92 43
08 24 92 53 06 24 B2 90 5E 00 06 24 29 38 0F 43
0E 43 82 93 08 24 01 24 1E 43 0E 93 01 20 1F 43
82 4F 08 24 82 93 08 24 07 20 1F 42 00 24 0F 5F
92 4F 42 47 56 03 02 3C 82 43 56 03 82 43 06 24
0F 3C 92 93 04 24 0C 20 82 43 04 24 82 43 06 24
92 43 08 24 1F 42 00 24 0F 5F 92 4F 42 47 56 03
3E 41 3F 41 00 13 0A 12 09 12 08 12 0A 4C 78 4A
09 43 11 3C 0E 4D 0E 8B 1E 83 1D 53 FD 4E FF FF
1F 83 FB 23 03 3C 1D 53 FD 4A FF FF 12 C3 08 10
19 53 39 92 EC 37 18 B3 F6 23 7B 4A 7F 4A 0C 4B
B0 12 CE 46 0B 4C 0C 4F B0 12 50 46 3C F0 0F 00
0B DC 3F F0 0F 00 3F 50 03 00 3F 90 12 00 0C 20
7E 4A 3E B0 80 00 07 24 7C 4A 4C 4C B0 12 C8 46
3E F0 7F 00 0E DC 0F 5E 3B 90 FF 0F CB 23 30 40
20 47 B2 40 80 5A 5C 01 F2 D2 0A 02 F2 D2 04 02
E2 C3 05 02 E2 D3 07 02 E2 D3 03 02 E2 D3 1B 02
E2 D3 19 02 E2 C3 1D 02 E2 C3 04 02 E2 D3 06 02
E2 D3 02 02 E2 D3 1A 02 E2 D3 18 02 E2 C3 1C 02
B2 40 1C 5A 5C 01 92 D3 00 01 B2 40 E7 03 52 03
B2 40 E0 00 46 03 1F 42 00 24 0F 5F 92 4F 42 47
56 03 B2 40 10 02 40 03 32 D2 32 D0 18 00 0C 43
30 41 0A 12 09 12 08 12 19 42 5C 01 B2 40 80 5A
5C 01 3F 40 62 47 3F 90 66 47 16 24 3F 40 66 47
3F 90 6A 47 11 24 3A 40 6A 47 3A 80 66 47 0A 11
0A 11 38 40 66 47 3C 48 7F 4C 0F 5F 1F 4F 62 47
3D 48 8F 12 1A 83 F7 23 79 C2 39 D0 08 5A 82 49
5C 01 B0 12 40 47 30 40 20 47 0F 12 0E 12 0D 12
0C 12 0B 12 E2 B3 1D 02 16 24 B2 40 20 4E 02 24
82 93 02 24 05 24 92 83 02 24 82 93 02 24 FB 23
E2 B3 1D 02 08 24 B2 90 06 00 00 24 04 2C 92 53
00 24 B0 12 28 47 E2 C3 1D 02 3B 41 3C 41 3D 41
3E 41 3F 41 00 13 3D F0 0F 00 3D E0 0F 00 0D 5D
0D 5D 00 5D 12 C3 0C 10 12 C3 0C 10 12 C3 0C 10
12 C3 0C 10 12 C3 0C 10 12 C3 0C 10 12 C3 0C 10
12 C3 0C 10 12 C3 0C 10 12 C3 0C 10 12 C3 0C 10
12 C3 0C 10 12 C3 0C 10 12 C3 0C 10 12 C3 0C 10
30 41 0F 12 0E 12 0D 12 0C 12 0B 12 E2 B3 1C 02
15 24 B2 40 20 4E 02 24 82 93 02 24 05 24 92 83
02 24 82 93 02 24 FB 23 E2 B3 1C 02 07 24 82 93
00 24 04 24 92 83 00 24 B0 12 28 47 E2 C3 1C 02
3B 41 3C 41 3D 41 3E 41 3F 41 00 13 3D F0 0F 00
3D E0 0F 00 0D 5D 00 5D 0C 5C 0C 5C 0C 5C 0C 5C
0C 5C 0C 5C 0C 5C 0C 5C 0C 5C 0C 5C 0C 5C 0C 5C
0C 5C 0C 5C 0C 5C 30 41 31 40 00 44 B0 12 3C 47
0C 93 02 24 B0 12 72 45 0C 43 B0 12 02 45 1C 43
B0 12 36 47 0F 4C 0C 4D 3D 40 03 00 0D 5F 1E 4F
01 00 30 40 06 47 0E 93 06 24 0F 4C 1F 53 FF 4D
FF FF 1E 83 FB 23 30 41 34 41 35 41 36 41 37 41
38 41 39 41 3A 41 30 41 1F 42 00 24 0F 5F 92 4F
42 47 56 03 30 41 03 43 FF 3F 03 43 1C 43 30 41
30 41 00 00 A7 00 4D 01 F4 01 9B 02 41 03 E8 03
32 D0 10 00 FD 3F 03 43 00 1B 03 00 00 03 01 00
FF F0 86 44 F4 46 58 47 00 24
@ffd2
50 47 CA 45 50 47 50 47 50 47 50 47 62 46 50 47
50 47 50 47 50 47 50 47 50 47 50 47 50 47 50 47
00 44 50 47 50 47 50 47 50 47 50 47 D8 46
q
```

Figure 2: PWM Hex File Screenshot

Problem 3:

Program Description:

The .out file generated from my Lab 6 PWM Code and the msp430-elf-readelf utility were used to answer the following questions:

a. **What is the magic number used?**
   The magic number used was 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00.
b. **What is the class of this .out file?**
   The class of the .out file is ELF32.
c. **What machine was this file built for?**
   The file was built for a Texas Instruments msp430 microcontroller.
d. **What is the size of the header?**
   The size of the header file is 52 bytes.
e. **How many section headers are there?**
   There are 99 section headers.

Program Output:

```
C:\Users\hgk0004\Documents\hgk0004_CPE325\Lab10\Debug>msp430-elf-readelf -h Lab10.out
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Texas Instruments msp430 microcontroller
  Version:                           0x1
  Entry point address:               0x46d8
  Start of program headers:          59568 (bytes into file)
  Start of section headers:          59696 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         4
  Size of section headers:           40 (bytes)
  Number of section headers:         99
  Section header string table index: 98
```

Figure 3: Screenshot of the msp-430-elf-readelf -h Lab10.out Command Output

Program 4:

Program Description:

Using naken utility and the hex file from part 2 the following questions were answered:

a. **Program the given hex file to your microcontroller using the MSP430Flasher tool and paste the output in your report.**

```
C:\Users\hgk0004\Documents\hgk0004_CPE325\Lab10\Debug>C:\ti\MSPFlasher_1.3.20\MSP430Flasher -n MSP430F5529 -w Lab10.txt
-v -z [Vcc]

* Unable to access log file. Creating folder...done
* -----/|---------------------------------------------------------------- *
*     / |__                                                               *
*    /_   /   MSP Flasher v1.3.20                                         *
*      | /                                                                *
* -----|/---------------------------------------------------------------- *
*
* Evaluating triggers...done
* Checking for available FET debuggers:
* Found USB FET @ COM9 <- Selected
* Initializing interface @ COM9...done
* Checking firmware compatibility:
* FET firmware is up to date.
* Reading FW version...
* Debugger does not support target voltages other than 3000 mV!
* Setting VCC to 3000 mV...done
* Accessing device...done
* Reading device information...done
* Loading file into device...done
* Verifying memory (Lab10.txt)...done
*
* ------------------------------------------------------------------------
* Arguments   : -n MSP430F5529 -w Lab10.txt -v -z [Vcc]
* ------------------------------------------------------------------------
* Driver      : loaded
* Dll Version : 31400000
* FwVersion   : 31200000
* Interface   : TIUSB
* HwVersion   : E 3.0
* JTAG Mode   : AUTO
* Device      : MSP430F5529
* EEM         : Level 7, ClockCntrl 2
* Erase Mode  : ERASE_ALL
* Prog.File   : Lab10.txt
* Verified    : TRUE
* BSL Unlock  : FALSE
* InfoA Access: FALSE
* VCC ON      : 3000 mV
* ------------------------------------------------------------------------
* Starting target code execution...done
* Disconnecting from device...done
*
* ------------------------------------------------------------------------
* Driver      : closed (No error)
* ------------------------------------------------------------------------
*/
```

Figure 4: MSPFlasher Command Line Output

b. Show that using the Flasher, you can change the Brightness Level as you could in the CCS environment.

After connecting LED1 (P1.0) to P1.3 and flashing the program, the program works as it did during Lab 6 using the normal CCS compiler. This was shown in my demo upon request.

c. Using the naken utility and the steps shown in Section 5.2 of the tutorial, reverse engineer the hex file to assembly code.

```
naken_util - by Michael Kohn
                Joe Davisson
    Web: https://www.mikekohn.net/
  Email: mike@mikekohn.net

Version:

Loaded RetrievedHex.txt of type ti_txt / msp430 from 0x4400 to 0x243ff
Type help for a list of commands.

Addr    Opcode Instruction                          Cycles
------- ------ -----------------------------------  ------
0x4400: 0x120f push.w r15                           3
0x4402: 0x120e push.w r14                           3
0x4404: 0xb3e2 bit.b #2, &0x0201                    4
0x4406: 0x0201
0x4408: 0x202c jne 0x4462  (offset: 88)             2
0x440a: 0xb3e2 bit.b #2, &0x0200                    4
0x440c: 0x0200
0x440e: 0x2029 jne 0x4462  (offset: 82)             2
0x4410: 0x9382 cmp.w #0, &0x2404                    4
0x4412: 0x2404
0x4414: 0x2006 jne 0x4422  (offset: 12)             2
0x4416: 0x4392 mov.w #1, &0x2404                    4
0x4418: 0x2404
0x441a: 0x4382 mov.w #0, &0x2406                    4
0x441c: 0x2406
```

Figure 5: Screenshot of a Portion of the Generated Assembly Code

d. Comment on each line of the assembly code generated from Q4c above to describe what each line is doing.

In the following multi-page assembly code my comments (C-style) are in bold.

```
Addr    Opcode Instruction                          Cycles
------- ------ -----------------------------------  ------
0x4400: 0x120f push.w r15    // push R15 onto the stack 3
0x4402: 0x120e push.w r14    // push R14 onto the stack 3
0x4404: 0xb3e2 bit.b #2, &0x0201                    4
0x4406: 0x0201 // logical and 2 and the value at address 0x0201
0x4408: 0x202c jne 0x4462 // jump to address 0x4462     2
0x440a: 0xb3e2 bit.b #2, &0x0200                    4
0x440c: 0x0200 // logical and 2 and the value at address 0x0200
0x440e: 0x2029 jne 0x4462  // jump to address 0x4462    2
0x4410: 0x9382 cmp.w #0, &0x2404                    4
0x4412: 0x2404 // compare the value at address 0x2404 to 0
0x4414: 0x2006 jne 0x4422  // jump to address 0x4422    2
0x4416: 0x4392 mov.w #1, &0x2404                    4
0x4418: 0x2404 // move the value 1 into memory at address 0x2404
0x441a: 0x4382 mov.w #0, &0x2406                    4
0x441c: 0x2406 // move the value 0 into memory at address 0x2406
0x441e: 0x4392 mov.w #1, &0x2408                    4
0x4420: 0x2408 // move the value 1 into memory at address 0x2408
0x4422: 0x5392 add.w #1, &0x2406                    4
0x4424: 0x2406 // move the value 1 into memory at address 0x2406
0x4426: 0x90b2 cmp.w #0x005e, &0x2406               5
0x4428: 0x005e // compare the value 0x5E to the value at address 0x2406
```

```
0x442a: 0x2406 // here until the bottom of page 15 is pure code
0x442c: 0x3829 jl 0x4480  (offset: 82)                    2
0x442e: 0x430f mov.w #0, r15                              1
0x4430: 0x430e mov.w #0, r14                              1
0x4432: 0x9382 cmp.w #0, &0x2408                          4
0x4434: 0x2408
0x4436: 0x2401 jeq 0x443a  (offset: 2)                    2
0x4438: 0x431e mov.w #1, r14                              1
0x443a: 0x930e cmp.w #0, r14                              1
0x443c: 0x2001 jne 0x4440  (offset: 2)                    2
0x443e: 0x431f mov.w #1, r15                              1
0x4440: 0x4f82 mov.w r15, &0x2408                         4
0x4442: 0x2408
0x4444: 0x9382 cmp.w #0, &0x2408                          4
0x4446: 0x2408
0x4448: 0x2007 jne 0x4458  (offset: 14)                   2
0x444a: 0x421f mov.w &0x2400, r15                         3
0x444c: 0x2400
0x444e: 0x5f0f add.w r15, r15                             1
0x4450: 0x4f92 mov.w 18242(r15), &0x0356                  6
0x4452: 0x4742
0x4454: 0x0356
0x4456: 0x3c02 jmp 0x445c  (offset: 4)                    2
0x4458: 0x4382 mov.w #0, &0x0356                          4
0x445a: 0x0356
0x445c: 0x4382 mov.w #0, &0x2406                          4
0x445e: 0x2406
0x4460: 0x3c0f jmp 0x4480  (offset: 30)                   2
0x4462: 0x9392 cmp.w #1, &0x2404                          4
0x4464: 0x2404
0x4466: 0x200c jne 0x4480  (offset: 24)                   2
0x4468: 0x4382 mov.w #0, &0x2404                          4
0x446a: 0x2404
0x446c: 0x4382 mov.w #0, &0x2406                          4
0x446e: 0x2406
0x4470: 0x4392 mov.w #1, &0x2408                          4
0x4472: 0x2408
0x4474: 0x421f mov.w &0x2400, r15                         3
0x4476: 0x2400
0x4478: 0x5f0f add.w r15, r15                             1
0x447a: 0x4f92 mov.w 18242(r15), &0x0356                  6
0x447c: 0x4742
0x447e: 0x0356
0x4480: 0x413e pop.w r14   --  mov.w @SP+, r14           2
0x4482: 0x413f pop.w r15   --  mov.w @SP+, r15           2
0x4484: 0x1300 reti                                       5
0x4486: 0x120a push.w r10                                 3
0x4488: 0x1209 push.w r9                                  3
0x448a: 0x1208 push.w r8                                  3
0x448c: 0x4c0a mov.w r12, r10                             1
0x448e: 0x4a78 mov.b @r10+, r8                            2
0x4490: 0x4309 mov.w #0, r9                               1
0x4492: 0x3c11 jmp 0x44b6  (offset: 34)                   2
0x4494: 0x4d0e mov.w r13, r14                             1
```

```
0x4496: 0x8b0e sub.w r11, r14                          1
0x4498: 0x831e sub.w #1, r14                           1
0x449a: 0x531d add.w #1, r13                           1
0x449c: 0x4efd mov.b @r14+, -1(r13)                    5
0x449e: 0xffff
0x44a0: 0x831f sub.w #1, r15                           1
0x44a2: 0x23fb jne 0x449a  (offset: -10)               2
0x44a4: 0x3c03 jmp 0x44ac  (offset: 6)                 2
0x44a6: 0x531d add.w #1, r13                           1
0x44a8: 0x4afd mov.b @r10+, -1(r13)                    5
0x44aa: 0xffff
0x44ac: 0xc312 clrc  --  bic.w #1, SR                  1
0x44ae: 0x1008 rrc.w r8                                1
0x44b0: 0x5319 add.w #1, r9                            1
0x44b2: 0x9239 cmp.w #8, r9                            1
0x44b4: 0x37ec jge 0x448e  (offset: -40)               2
0x44b6: 0xb318 bit.w #1, r8                            1
0x44b8: 0x23f6 jne 0x44a6  (offset: -20)               2
0x44ba: 0x4a7b mov.b @r10+, r11                        2
0x44bc: 0x4a7f mov.b @r10+, r15                        2
0x44be: 0x4b0c mov.w r11, r12                          1
0x44c0: 0x12b0 call #0x46ce                            5
0x44c2: 0x46ce
0x44c4: 0x4c0b mov.w r12, r11                          1
0x44c6: 0x4f0c mov.w r15, r12                          1
0x44c8: 0x12b0 call #0x4650                            5
0x44ca: 0x4650
0x44cc: 0xf03c and.w #0x000f, r12                      2
0x44ce: 0x000f
0x44d0: 0xdc0b bis.w r12, r11                          1
0x44d2: 0xf03f and.w #0x000f, r15                      2
0x44d4: 0x000f
0x44d6: 0x503f add.w #0x0003, r15                      2
0x44d8: 0x0003
0x44da: 0x903f cmp.w #0x0012, r15                      2
0x44dc: 0x0012
0x44de: 0x200c jne 0x44f8  (offset: 24)                2
0x44e0: 0x4a7e mov.b @r10+, r14                        2
0x44e2: 0xb03e bit.w #0x0080, r14                      2
0x44e4: 0x0080
0x44e6: 0x2407 jeq 0x44f6  (offset: 14)                2
0x44e8: 0x4a7c mov.b @r10+, r12                        2
0x44ea: 0x4c4c mov.b r12, r12                          1
0x44ec: 0x12b0 call #0x46c8                            5
0x44ee: 0x46c8
0x44f0: 0xf03e and.w #0x007f, r14                      2
0x44f2: 0x007f
0x44f4: 0xdc0e bis.w r12, r14                          1
0x44f6: 0x5e0f add.w r14, r15                          1
0x44f8: 0x903b cmp.w #0x0fff, r11                      2
0x44fa: 0x0fff
0x44fc: 0x23cb jne 0x4494  (offset: -106)              2
0x44fe: 0x4030 mov.w #0x4720, PC                       3
0x4500: 0x4720
```

```
0x4502: 0x40b2 mov.w #0x5a80, &0x015c                  5
0x4504: 0x5a80
0x4506: 0x015c
0x4508: 0xd2f2 bis.b #8, &0x020a                       4
0x450a: 0x020a
0x450c: 0xd2f2 bis.b #8, &0x0204                       4
0x450e: 0x0204
0x4510: 0xc3e2 bic.b #2, &0x0205                       4
0x4512: 0x0205
0x4514: 0xd3e2 bis.b #2, &0x0207                       4
0x4516: 0x0207
0x4518: 0xd3e2 bis.b #2, &0x0203                       4
0x451a: 0x0203
0x451c: 0xd3e2 bis.b #2, &0x021b                       4
0x451e: 0x021b
0x4520: 0xd3e2 bis.b #2, &0x0219                       4
0x4522: 0x0219
0x4524: 0xc3e2 bic.b #2, &0x021d                       4
0x4526: 0x021d
0x4528: 0xc3e2 bic.b #2, &0x0204                       4
0x452a: 0x0204
0x452c: 0xd3e2 bis.b #2, &0x0206                       4
0x452e: 0x0206
0x4530: 0xd3e2 bis.b #2, &0x0202                       4
0x4532: 0x0202
0x4534: 0xd3e2 bis.b #2, &0x021a                       4
0x4536: 0x021a
0x4538: 0xd3e2 bis.b #2, &0x0218                       4
0x453a: 0x0218
0x453c: 0xc3e2 bic.b #2, &0x021c                       4
0x453e: 0x021c
0x4540: 0x40b2 mov.w #0x5a1c, &0x015c                  5
0x4542: 0x5a1c
0x4544: 0x015c
0x4546: 0xd392 bis.w #1, &0x0100                       4
0x4548: 0x0100
0x454a: 0x40b2 mov.w #0x03e7, &0x0352                  5
0x454c: 0x03e7
0x454e: 0x0352
0x4550: 0x40b2 mov.w #0x00e0, &0x0346                  5
0x4552: 0x00e0
0x4554: 0x0346
0x4556: 0x421f mov.w &0x2400, r15                      3
0x4558: 0x2400
0x455a: 0x5f0f add.w r15, r15                          1
0x455c: 0x4f92 mov.w 18242(r15), &0x0356               6
0x455e: 0x4742
0x4560: 0x0356
0x4562: 0x40b2 mov.w #0x0210, &0x0340                  5
0x4564: 0x0210
0x4566: 0x0340
0x4568: 0xd232 eint  --  bis.w #8, SR                  1
0x456a: 0xd032 bis.w #0x0018, SR                       2
0x456c: 0x0018
```

```
0x456e: 0x430c mov.w #0, r12                          1
0x4570: 0x4130 ret   --  mov.w @SP+, PC               3
0x4572: 0x120a push.w r10                             3
0x4574: 0x1209 push.w r9                              3
0x4576: 0x1208 push.w r8                              3
0x4578: 0x4219 mov.w &0x015c, r9                      3
0x457a: 0x015c
0x457c: 0x40b2 mov.w #0x5a80, &0x015c                 5
0x457e: 0x5a80
0x4580: 0x015c
0x4582: 0x403f mov.w #0x4762, r15                     2
0x4584: 0x4762
0x4586: 0x903f cmp.w #0x4766, r15                     2
0x4588: 0x4766
0x458a: 0x2416 jeq 0x45b8  (offset: 44)               2
0x458c: 0x403f mov.w #0x4766, r15                     2
0x458e: 0x4766
0x4590: 0x903f cmp.w #0x476a, r15                     2
0x4592: 0x476a
0x4594: 0x2411 jeq 0x45b8  (offset: 34)               2
0x4596: 0x403a mov.w #0x476a, r10                     2
0x4598: 0x476a
0x459a: 0x803a sub.w #0x4766, r10                     2
0x459c: 0x4766
0x459e: 0x110a rra.w r10                              1
0x45a0: 0x110a rra.w r10                              1
0x45a2: 0x4038 mov.w #0x4766, r8                      2
0x45a4: 0x4766
0x45a6: 0x483c mov.w @r8+, r12                        2
0x45a8: 0x4c7f mov.b @r12+, r15                       2
0x45aa: 0x5f0f add.w r15, r15                         1
0x45ac: 0x4f1f mov.w 18274(r15), r15                  3
0x45ae: 0x4762
0x45b0: 0x483d mov.w @r8+, r13                        2
0x45b2: 0x128f call r15                               4
0x45b4: 0x831a sub.w #1, r10                          1
0x45b6: 0x23f7 jne 0x45a6  (offset: -18)              2
0x45b8: 0xc279 bic.b #8, r9                           1
0x45ba: 0xd039 bis.w #0x5a08, r9                      2
0x45bc: 0x5a08
0x45be: 0x4982 mov.w r9, &0x015c                      4
0x45c0: 0x015c
0x45c2: 0x12b0 call #0x4740                           5
0x45c4: 0x4740
0x45c6: 0x4030 mov.w #0x4720, PC                      3
0x45c8: 0x4720
0x45ca: 0x120f push.w r15                             3
0x45cc: 0x120e push.w r14                             3
0x45ce: 0x120d push.w r13                             3
0x45d0: 0x120c push.w r12                             3
0x45d2: 0x120b push.w r11                             3
0x45d4: 0xb3e2 bit.b #2, &0x021d                      4
0x45d6: 0x021d
0x45d8: 0x2416 jeq 0x4606  (offset: 44)               2
```

```
0x45da: 0x40b2 mov.w #0x4e20, &0x2402                    5
0x45dc: 0x4e20
0x45de: 0x2402
0x45e0: 0x9382 cmp.w #0, &0x2402                         4
0x45e2: 0x2402
0x45e4: 0x2405 jeq 0x45f0  (offset: 10)                  2
0x45e6: 0x8392 sub.w #1, &0x2402                         4
0x45e8: 0x2402
0x45ea: 0x9382 cmp.w #0, &0x2402                         4
0x45ec: 0x2402
0x45ee: 0x23fb jne 0x45e6  (offset: -10)                 2
0x45f0: 0xb3e2 bit.b #2, &0x021d                         4
0x45f2: 0x021d
0x45f4: 0x2408 jeq 0x4606  (offset: 16)                  2
0x45f6: 0x90b2 cmp.w #0x0006, &0x2400                    5
0x45f8: 0x0006
0x45fa: 0x2400
0x45fc: 0x2c04 jhs 0x4606  (offset: 8)                   2
0x45fe: 0x5392 add.w #1, &0x2400                         4
0x4600: 0x2400
0x4602: 0x12b0 call #0x4728                              5
0x4604: 0x4728
0x4606: 0xc3e2 bic.b #2, &0x021d                         4
0x4608: 0x021d
0x460a: 0x413b pop.w r11   --  mov.w @SP+, r11          2
0x460c: 0x413c pop.w r12   --  mov.w @SP+, r12          2
0x460e: 0x413d pop.w r13   --  mov.w @SP+, r13          2
0x4610: 0x413e pop.w r14   --  mov.w @SP+, r14          2
0x4612: 0x413f pop.w r15   --  mov.w @SP+, r15          2
0x4614: 0x1300 reti                                      5
0x4616: 0xf03d and.w #0x000f, r13                        2
0x4618: 0x000f
0x461a: 0xe03d xor.w #0x000f, r13                        2
0x461c: 0x000f
0x461e: 0x5d0d add.w r13, r13                            1
0x4620: 0x5d0d add.w r13, r13                            1
0x4622: 0x5d00 add.w r13, PC                             2
0x4624: 0xc312 clrc  --  bic.w #1, SR                    1
0x4626: 0x100c rrc.w r12                                 1
0x4628: 0xc312 clrc  --  bic.w #1, SR                    1
0x462a: 0x100c rrc.w r12                                 1
0x462c: 0xc312 clrc  --  bic.w #1, SR                    1
0x462e: 0x100c rrc.w r12                                 1
0x4630: 0xc312 clrc  --  bic.w #1, SR                    1
0x4632: 0x100c rrc.w r12                                 1
0x4634: 0xc312 clrc  --  bic.w #1, SR                    1
0x4636: 0x100c rrc.w r12                                 1
0x4638: 0xc312 clrc  --  bic.w #1, SR                    1
0x463a: 0x100c rrc.w r12                                 1
0x463c: 0xc312 clrc  --  bic.w #1, SR                    1
0x463e: 0x100c rrc.w r12                                 1
0x4640: 0xc312 clrc  --  bic.w #1, SR                    1
0x4642: 0x100c rrc.w r12                                 1
0x4644: 0xc312 clrc  --  bic.w #1, SR                    1
```

```
0x4646: 0x100c rrc.w r12                                1
0x4648: 0xc312 clrc  --  bic.w #1, SR                   1
0x464a: 0x100c rrc.w r12                                1
0x464c: 0xc312 clrc  --  bic.w #1, SR                   1
0x464e: 0x100c rrc.w r12                                1
0x4650: 0xc312 clrc  --  bic.w #1, SR                   1
0x4652: 0x100c rrc.w r12                                1
0x4654: 0xc312 clrc  --  bic.w #1, SR                   1
0x4656: 0x100c rrc.w r12                                1
0x4658: 0xc312 clrc  --  bic.w #1, SR                   1
0x465a: 0x100c rrc.w r12                                1
0x465c: 0xc312 clrc  --  bic.w #1, SR                   1
0x465e: 0x100c rrc.w r12                                1
0x4660: 0x4130 ret   --  mov.w @SP+, PC                 3
0x4662: 0x120f push.w r15                               3
0x4664: 0x120e push.w r14                               3
0x4666: 0x120d push.w r13                               3
0x4668: 0x120c push.w r12                               3
0x466a: 0x120b push.w r11                               3
0x466c: 0xb3e2 bit.b #2, &0x021c                        4
0x466e: 0x021c
0x4670: 0x2415 jeq 0x469c  (offset: 42)                 2
0x4672: 0x40b2 mov.w #0x4e20, &0x2402                   5
0x4674: 0x4e20
0x4676: 0x2402
0x4678: 0x9382 cmp.w #0, &0x2402                        4
0x467a: 0x2402
0x467c: 0x2405 jeq 0x4688  (offset: 10)                 2
0x467e: 0x8392 sub.w #1, &0x2402                        4
0x4680: 0x2402
0x4682: 0x9382 cmp.w #0, &0x2402                        4
0x4684: 0x2402
0x4686: 0x23fb jne 0x467e  (offset: -10)                2
0x4688: 0xb3e2 bit.b #2, &0x021c                        4
0x468a: 0x021c
0x468c: 0x2407 jeq 0x469c  (offset: 14)                 2
0x468e: 0x9382 cmp.w #0, &0x2400                        4
0x4690: 0x2400
0x4692: 0x2404 jeq 0x469c  (offset: 8)                  2
0x4694: 0x8392 sub.w #1, &0x2400                        4
0x4696: 0x2400
0x4698: 0x12b0 call #0x4728                             5
0x469a: 0x4728
0x469c: 0xc3e2 bic.b #2, &0x021c                        4
0x469e: 0x021c
0x46a0: 0x413b pop.w r11   --  mov.w @SP+, r11         2
0x46a2: 0x413c pop.w r12   --  mov.w @SP+, r12         2
0x46a4: 0x413d pop.w r13   --  mov.w @SP+, r13         2
0x46a6: 0x413e pop.w r14   --  mov.w @SP+, r14         2
0x46a8: 0x413f pop.w r15   --  mov.w @SP+, r15         2
0x46aa: 0x1300 reti                                     5
0x46ac: 0xf03d and.w #0x000f, r13                       2
0x46ae: 0x000f
0x46b0: 0xe03d xor.w #0x000f, r13                       2
```

```
0x46b2: 0x000f
0x46b4: 0x5d0d add.w r13, r13                          1
0x46b6: 0x5d00 add.w r13, PC                           2
0x46b8: 0x5c0c add.w r12, r12                          1
0x46ba: 0x5c0c add.w r12, r12                          1
0x46bc: 0x5c0c add.w r12, r12                          1
0x46be: 0x5c0c add.w r12, r12                          1
0x46c0: 0x5c0c add.w r12, r12                          1
0x46c2: 0x5c0c add.w r12, r12                          1
0x46c4: 0x5c0c add.w r12, r12                          1
0x46c6: 0x5c0c add.w r12, r12                          1
0x46c8: 0x5c0c add.w r12, r12                          1
0x46ca: 0x5c0c add.w r12, r12                          1
0x46cc: 0x5c0c add.w r12, r12                          1
0x46ce: 0x5c0c add.w r12, r12                          1
0x46d0: 0x5c0c add.w r12, r12                          1
0x46d2: 0x5c0c add.w r12, r12                          1
0x46d4: 0x5c0c add.w r12, r12                          1
0x46d6: 0x4130 ret   --  mov.w @SP+, PC               3
0x46d8: 0x4031 mov.w #0x4400, SP                       2
0x46da: 0x4400
0x46dc: 0x12b0 call #0x473c                            5
0x46de: 0x473c
0x46e0: 0x930c cmp.w #0, r12                           1
0x46e2: 0x2402 jeq 0x46e8  (offset: 4)                 2
0x46e4: 0x12b0 call #0x4572                            5
0x46e6: 0x4572
0x46e8: 0x430c mov.w #0, r12                           1
0x46ea: 0x12b0 call #0x4502                            5
0x46ec: 0x4502
0x46ee: 0x431c mov.w #1, r12                           1
0x46f0: 0x12b0 call #0x4736                            5
0x46f2: 0x4736
0x46f4: 0x4c0f mov.w r12, r15                          1
0x46f6: 0x4d0c mov.w r13, r12                          1
0x46f8: 0x403d mov.w #0x0003, r13                      2
0x46fa: 0x0003
0x46fc: 0x5f0d add.w r15, r13                          1
0x46fe: 0x4f1e mov.w 1(r15), r14                       3
0x4700: 0x0001
0x4702: 0x4030 mov.w #0x4706, PC                       3
0x4704: 0x4706
0x4706: 0x930e cmp.w #0, r14                           1
0x4708: 0x2406 jeq 0x4716  (offset: 12)                2
0x470a: 0x4c0f mov.w r12, r15                          1
0x470c: 0x531f add.w #1, r15                           1
0x470e: 0x4dff mov.b @r13+, -1(r15)                    5
0x4710: 0xffff
0x4712: 0x831e sub.w #1, r14                           1
0x4714: 0x23fb jne 0x470c  (offset: -10)               2
0x4716: 0x4130 ret   --  mov.w @SP+, PC               3
0x4718: 0x4134 pop.w r4   --  mov.w @SP+, r4          2
0x471a: 0x4135 pop.w r5   --  mov.w @SP+, r5          2
0x471c: 0x4136 pop.w r6   --  mov.w @SP+, r6          2
```

```
0x471e: 0x4137 pop.w r7   --  mov.w @SP+, r7           2
0x4720: 0x4138 pop.w r8   --  mov.w @SP+, r8           2
0x4722: 0x4139 pop.w r9   --  mov.w @SP+, r9           2
0x4724: 0x413a pop.w r10   --  mov.w @SP+, r10         2
0x4726: 0x4130 ret   --  mov.w @SP+, PC               3
0x4728: 0x421f mov.w &0x2400, r15                     3
0x472a: 0x2400
0x472c: 0x5f0f add.w r15, r15                         1
0x472e: 0x4f92 mov.w 18242(r15), &0x0356              6
0x4730: 0x4742
0x4732: 0x0356
0x4734: 0x4130 ret   --  mov.w @SP+, PC               3
0x4736: 0x4303 nop   --  mov.w #0, CG                 1
0x4738: 0x3fff jmp 0x4738  (offset: -2)               2
0x473a: 0x4303 nop   --  mov.w #0, CG                 1
0x473c: 0x431c mov.w #1, r12                          1
0x473e: 0x4130 ret   --  mov.w @SP+, PC               3
0x4740: 0x4130 ret   --  mov.w @SP+, PC               3
0x4742: 0x0000 mova @PC, PC                           ?
0x4744: 0x00a7 adda #0x14d, r7                        ?
0x4746: 0x014d
0x4748: 0x01f4 suba SP, r4                            ?
0x474a: 0x029b cmpa #0x20341, r11                     ?
0x474c: 0x0341
0x474e: 0x03e8 adda CG, r8                            ?
0x4750: 0xd032 bis.w #0x0010, SR                      2
0x4752: 0x0010
0x4754: 0x3ffd jmp 0x4750  (offset: -6)               2
0x4756: 0x4303 nop   --  mov.w #0, CG                 1
0x4758: 0x1b00 mova @PC, CG                           ?
0x475a: 0x0003
0x475c: 0x0300 mova @CG, PC                           ?
0x475e: 0x0001 mova @PC, SP                           ?
0x4760: 0xf0ff and.b #0x4486, 18164(r15)              5
0x4762: 0x4486
0x4764: 0x46f4
0x4766: 0x4758 mov.b 9216(r7), r8                     3
0x4768: 0x2400
0x476a: 0xffff and.b @r15+, -1(r15)                   5
```

e. Describe what the program is doing in a neat flowchart. You can also write a paragraph to describe in addition to the flowchart.

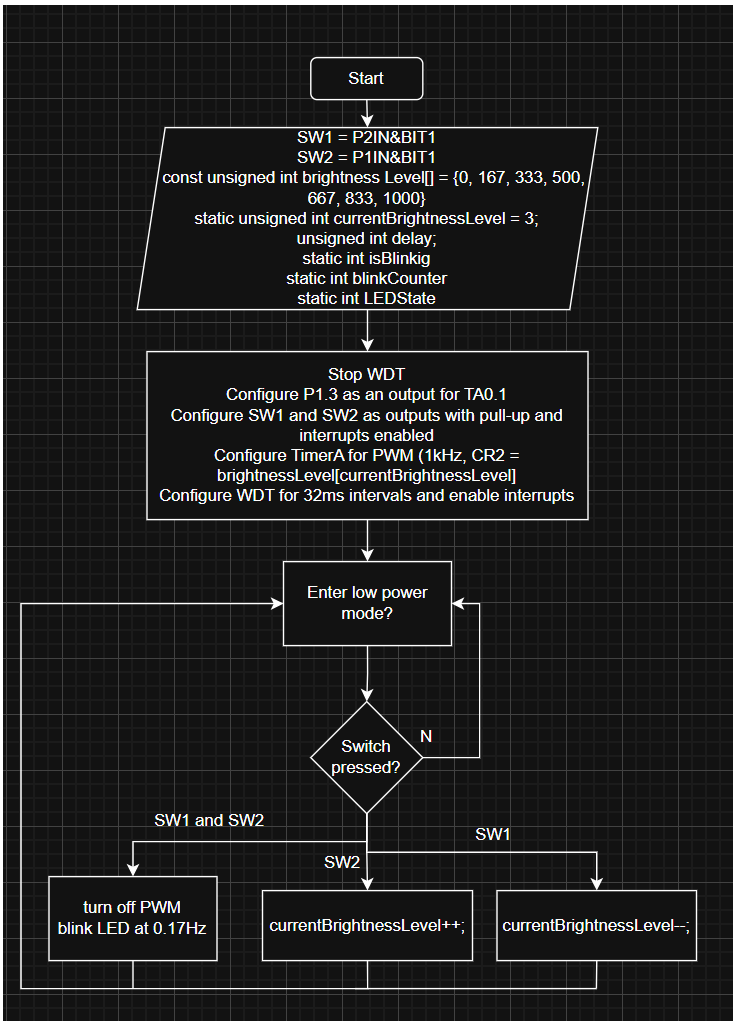

Figure 6: Program 4 Flowchart

Conclusion

In this lab, we explored several important tools and file structures fundamental to embedded systems development using the MSP430 platform. The ELF file format provides a well-organized structure that defines how program code, data, and memory are managed during execution, offering valuable insights into how compilers and linkers prepare firmware for microcontrollers. The Naken Utility demonstrated how low-level assembly code can be assembled, disassembled, and simulated, giving developers precise control over program behavior and facilitating debugging without the need for hardware. Finally, the MSP430 Flasher showcased an efficient method for programming and verifying firmware through a command-line interface, supporting automation and streamlined production workflows. Together, these tools and concepts build a comprehensive understanding of how embedded software is created, analyzed, and deployed, reinforcing key principles of low-level programming and system-level development.

The biggest challenge for me was getting the software to work on the lab computers. The programs and command line were very unintuitive to me. The tutorial helped, but I found I needed to do much more trial and error than I expected to while using the tutorial as a reference. Ultimately I learned valuable skills in debugging and troubleshooting.

Appendix

Table 1: Lab 6 PWM Source Code (C)

```
/*-------------------------------------------------------------------------
 * File:        Lab03_P1.c
 * Board:       5529
 * Input:       SW1 and SW2
 * Output:      LED1
 * Author:      Hannah Kelley
 * Date:        October 6, 2025
 *-------------------------------------------------------------------------*/
#include <msp430.h>
#include <stdio.h>
#define SW1 P2IN&BIT1
#define SW2 P1IN&BIT1
const unsigned int brightnessLevel[] = {0, 167, 333, 500, 667, 833, 1000};
static unsigned int currentBrightnessLevel = 3; // init brightness = 50%
unsigned int delay = 0;
static int isBlinking = 0;
static int blinkCounter = 0;
static int LEDState = 1;
void updateBrightness(); // function prototype

int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Stop WDT
    // config PWM output
    P1SEL |= BIT3; // select TA0.1 for PWM output
    P1DIR |= BIT3; // set P1.3 to output
    // config SW1 (increase)
    P2DIR &= ~BIT1; // set SW2 as input
    P2REN |= BIT1;  // enable pullup reg
    P2OUT |= BIT1;  // set as output
    P2IE |= BIT1;   // enable interupt for SW2
    P2IES |= BIT1;  // set interrupt on falling edge
    P2IFG &= ~BIT1; // clear interrupt flag
    // config SW2 (decrease)
    P1DIR &= ~BIT1; // set SW1 as input
    P1REN |= BIT1;  // enable pullup reg
    P1OUT |= BIT1;  // set as output
    P1IE |= BIT1;   // enable interupt for SW1
    P1IES |= BIT1;  // set interrupt on falling edge
    P1IFG &= ~BIT1; // clear interrupt flag
    WDTCTL = WDT_MDLY_32; // config WDT for 32ms
    SFRIE1 |= WDTIE;      // enable WDT interrupt
    // config timer A for PWM
    TA0CCR0 = 1000 - 1;         // set period to 1kHz
    TA0CCTL2 = OUTMOD_7;        // set/reset mode
    TA0CCR2 = brightnessLevel[currentBrightnessLevel];
    TA0CTL = TASSEL_2 | MC_1;   // use SMCLK in up mode
    _EINT(); // Enable global interrupts
    __bis_SR_register(LPM0_bits + GIE);
    return 0; // end of program
}
void updateBrightness() {
    TA0CCR2 = brightnessLevel[currentBrightnessLevel];
    return;
```

```
}
// SW1 ISR (increase brightness level)
#pragma vector = PORT2_VECTOR
__interrupt void PORT2_ISR(void) {
    if (P2IFG & BIT1) {                             // if SW1 pressed
        for (delay = 20000; delay > 0; delay--);    // debounce
        if (P2IFG & BIT1) {         // if still pressed it is a valid press
            if (currentBrightnessLevel < 6) {// if level can be increased
                currentBrightnessLevel++;           // increase level
                updateBrightness();                 // update LED
            }
        }
    }
    P2IFG &= ~BIT1;                           // clear interrupt flag
}
// SW2 ISR (increase brightness level)
#pragma vector = PORT1_VECTOR
__interrupt void PORT1_ISR(void) {
    if (P1IFG & BIT1) {                             // if SW2 pressed
        for (delay = 20000; delay > 0; delay--);    // debounce
        if (P1IFG & BIT1) {         // if still pressed it is a valid press
            if (currentBrightnessLevel > 0) { // if level can be increased
                currentBrightnessLevel--;           // decrease level
                updateBrightness();                 // update LED
            }
        }
    }
    P1IFG &= ~BIT1;                                 // clear interrupt flag
}
// WDT ISR
#pragma vector = WDT_VECTOR
__interrupt void WATCHDOG_TIMER(void) {
    if(((SW1) == 0) && ((SW2) == 0)) {
        if (isBlinking == 0) {
            isBlinking = 1;
            blinkCounter = 0;
            LEDState = 1;
        }
        blinkCounter++;
        if (blinkCounter >= 94) {
            LEDState = !LEDState;
            if (!LEDState) {
                TA0CCR2 = brightnessLevel[currentBrightnessLevel];
            }
            else {
                TA0CCR2 = 0;
            }
            blinkCounter = 0;
        }
    }
    else {
        if (isBlinking == 1) {
            isBlinking = 0;
            blinkCounter = 0;
            LEDState = 1;
            TA0CCR2 = brightnessLevel[currentBrightnessLevel];
        }
```

```
        }
}
```