

ELEC5620M | Assignment 1: 7-Segment Display Driver

Mathew Fuller

Github Username: el18mf

201289335



DE1-SoC F65

Property of University of Leeds
School of Electronic & Electrical Engineering

ABSTRACT

This report details the steps taken to ensure the creation of a well optimised, functioning a 7-segment display driver tailored specifically for the DE1-SoC board. To facilitate this, the use of GitHub to track and safeguard change to the code, Visual Studio Code to facilitate the coding interface and committing to GitHub, and the Arm Development Studio (ADS) which allows the error checking and debugging of the driver.

By meticulously unpacking the debugging process and detailing the outcomes of our hardware testing undertaken, this document offers invaluable insights into the multifaceted challenges we encountered and the innovative solutions we devised to surmount them.

This approach underscores the significance of a holistic perspective during development regarding embedded systems. An approach that seamlessly integrates rigorous debugging practices, comprehensive testing methodologies, and strategic, forward-thinking project management strategies to maximise the end efficacy of software being developed, whether a broad metal approach or an operating system (OS) approach, are important lessons and skills applicable across the board.

INTRODUCTION:

The project focuses on the application of hardware programming and debugging learnt to gauge and increase proficiency in said areas, through the implementation of a 7-segment display driver for the DE1-SoC board.

By employing low-level C programming and ADS, a plethora of coding and debugging was able to be undertaken, which is outlined in this report alongside the steps involved in configuring the project, debugging code errors, validating hardware functionality, and gauging the efficacy of the developed driver.

This report will detail the resultant working driver, the debugging to back-up this assertion about said driver, and the processes employed throughout that facilitated this outcome.

DEBUGGING

CHALLENGES & SOLUTIONS

From the outset setbacks were aplenty, with files disappearing from the root, incorrect naming instructions preventing the required files combination, such as with error L6218E which was solved due to comprehensive preparatory approach to the assignment despite the short time scale. This was exacerbated by issues with internet connectivity, IDE configuration issues among several others, all of which being detrimental due to the already strict and short timeline due to previous mitigating circumstances hindering the undertaking of this assignment.

The configuration of the settings, while rough at points, was able to be quickly tackled through a meticulous, individual approach to each issue as it cropped up. All issues, small and large, required diligent scrutiny of execution flow and strategic placement of breakpoints, techniques commonly employed and advocated for by those who abide by





DE1-SoC F65

Property of University of Leeds
School of Electronic & Electrical Engineering

industry-standard debugging methodologies. Real-time monitoring of register values emerged as a critical tool in our debugging arsenal during these issues, enabling precise identification of code faults and errors at a faster pace than usual, such as infinite loops or incorrectly named files, while simultaneously facilitating targeted interventions and strategies frequently employed during debugging in embedded systems development.

This enabled the quick adaption of the code to errors and errors, which ended up being paramount to keep on track with regards to time. A sort of collaborative approach with those who came before me, through the adaption of vastly differing resources solutions were derived. Debugging also enables intuitive learning, as you are able to gauge the functioning of certain aspects of the hardware such as the buttons activation and how the software would only register said activation in the following loop and not the loop it was activate in, i.e. if `keys_pressed` was prompted by `KEY1`, it is not instantaneous as the DE1-SoC will enter the initial conditional branch, causing the `getPressedKeys()` function to increment the `single_hex_display_value` by 1, disproving my initial assumption before inspecting the code of behaviour more akin to flags and/or interrupts. This highlighted an added benefit not normally attributed to the act of debugging, which is garnering of important information during the debugging process tangentially or totally unrelated to the goals linked to the debugging process.

HARDWARE TESTING

There were several minimal hardware tests, but the two most notable were the physical testing and documentation of the `KEY0` during Step 4. Done through the iterative activation of the button (`KEY0`) until the range limit, `F`, was reached. After said range, as intended, the display showed solely the middle segment with indicator no. 6 being activated which signals an out-of-range value.

The other was when file errors, indicating missing files preventing the USB Booster to work, as well as intermittent connection issues via the USB-A cable, warranted a look-over with a multimeter before further use to ensure no components were damaged while ensuring no further damage in the case there was.

DISCUSSIONS AND CONCLUSIONS

In conclusion, through the information garnered above, one can surmise that polling is used to implement the driver as a package. It is able to control display 0, 2, 3, 4, and 5 through variables (unsigned int's). It is also aided by modular nature of display 0 and it's display function, which not only aids the driver with display 2 and 3, but also highlights positive attributes of the driver such as its cohesion efficacy.

Moving forward, the insights gained from this project will inform future endeavours in embedded system development, having taught valuable lessons about the importance of prep and boon debugging can be.



LEEDS

terasic

Designed & Manufactured by Terasic

Download DE1-SoC CD from

APPENDIX CONTENTS

Abstract	1
Introduction:.....	1
Debugging	1
Challenges & Solutions.....	1
Hardware Testing	1
Discussions and Conclusions.....	1
Appendix	0

SOURCE CODE

```
/*
 * DE1SoC_SevenSeg.c
 *
 * Created on: 12 Feb 2021 | Author: Harry Clegg
 *
 * Updated on: 24 Feb 2024 | Author: David Cowell
 * Updated on: 16 Mar 2024 | Author: Mathew Fuller
 */

#include "DE1SoC_SevenSeg.h"

#define SEVENSEG_N_DISPLAYS_LO 4 // There are four HEX displays attached to
the low (first) address.
#define SEVENSEG_N_DISPLAYS_HI 2 // There are two HEX displays attached to the
high (second) address.

#define SEVENSEG_N_SINGLE_RANGE 15 // The maximum value that a single picture
tube can achieve

#define SEVENSEG_N_DEFAULT_NUM 16 // As the name says, it assigns a default
value of 16
#define SEVENSEG_N_DEFAULT_DISPLAY 0x40 // The stock display, resulting in the
middle bar (no.6) being active

#define SEVENSEG_N_DOUBLEHEX_RANGE 0xff // Used for Step 6: DoubleHex's max
Value
#define SEVENSEG_N_DOUBLEDEC_RANGE 0x63 // Used for Step 7: DoubleDec's max
value

// Step 4: Add the base addresses of the seven segment display peripherals.
```


DE1-SoC F65

Property of University of Leeds
School of Electronic & Electrical Engineering

```
volatile unsigned char *sevensseg_base_lo_ptr = (unsigned char *)0xFF200020; //
Step 4 Solution: change 0 to (unsigned char *)0xFF200020, which is comprised
of display 0-3
volatile unsigned char *sevensseg_base_hi_ptr = (unsigned char *)0xFF200030; //
Step 4 Solution: change 0 to (unsigned char *)0xFF200030, which is comprised
of display 4 & 5

// Using the excel document I was able to find all the hex values for 0-9 and
A-F, as well as providing a frame work to quickly figure out how to implement
more alphanumericals.
// table[16] = { 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7
, 8 , 9 , A , B , C , D , E , F };
int t_hex[16] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F,
0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};

// Select between the two addresses so that the higher level functions have a
seamless interface. If we are targeting a low address, use byte addressing to
access directly.
void DE1SoC_SevenSeg_Write(unsigned int display, unsigned char value)

    {if (display < SEVENSEG_N_DISPLAYS_LO) {sevensseg_base_lo_ptr[display] =
value;}}

    else {display = display - SEVENSEG_N_DISPLAYS_LO;
sevensseg_base_hi_ptr[display] = value;}} // If we are targeting a high
address, shift down so byte addressing works.

// Step 5: Write the code for driving a single seven segment display here.
Your function should turn a real value 0-F into the correctly encoded
//      bits to enable the correct segments on the seven segment display to
illuminate. Use the DE1SoC_SevenSeg_Write function you created earlier to set
the bits of the display.*/
// Step 5: Solution - Checks if variable value is <16. If true, the unsigned
int value translates to the relevant bit-mapping value.
void DE1SoC_SevenSeg_SetSingle(unsigned int display, unsigned int value)

    {if(value <= SEVENSEG_N_SINGLE_RANGE) {DE1SoC_SevenSeg_Write(display,
t_hex[value]);}}

    else {DE1SoC_SevenSeg_Write(display, SEVENSEG_N_DEFAULT_DISPLAY);}}
//When value exits accepted range middle segment illuminates. Sets the middle
segment active
```



Download DE1-SoC CD from

DE1-SoC F65

Property of University of Leeds
School of Electronic & Electrical Engineering

```
// Step 6: Write the code for setting a pair of seven segment displays here.
Good coding practice suggests your solution should call
DE1SoC_SevenSeg_SetSingle() twice.
// This function should show the first digit of a HEXADECIMAL number on the
specified 'display', and the second digit on the display to the left of
the specified display.
void DE1SoC_SevenSeg_SetDoubleHex(unsigned int display, unsigned int value)

    {if(value >
SEVENSEG_N_DOUBLEHEX_RANGE){DE1SoC_SevenSeg_SetSingle(display,SEVENSEG_N_DEFAU
LT_NUM); DE1SoC_SevenSeg_SetSingle(display+1,SEVENSEG_N_DEFAULT_NUM);}

// Both above and below follow similar structure. If the if statement is
true, the range of the first bit is roughly 0 to 0xF
// while the second bit, if the if statment is false, with the variable value
being altered using & and a shift of 4 bits to the left

        else{DE1SoC_SevenSeg_SetSingle(display,value & 0xf);
DE1SoC_SevenSeg_SetSingle(display+1,(value & 0xf0)>>4);}} // Feeds back
data into DE1SoC_SevenSeg_Write (apologies for such succinct commenting from
here, time is a factor)

// Step 7: Write the code for setting a pair of seven segment displays here.
Good coding practice suggests your solution should call
DE1SoC_SevenSeg_SetSingle() twice.
// This function should show the first digit of a DECIMAL number on the
specified 'display', and the second digit on the display to the left of the
specified display.
void DE1SoC_SevenSeg_SetDoubleDec(unsigned int display, unsigned int value)
//A default trigger limit is set at 99 (Dec)/ 0x63 (Hex), which if the value
goes over it shall trigger a default condition, while
    {if(value > SEVENSEG_N_DOUBLEDEC_RANGE ){
DE1SoC_SevenSeg_SetSingle(display,SEVENSEG_N_DEFAULT_NUM);
DE1SoC_SevenSeg_SetSingle(display+1,SEVENSEG_N_DEFAULT_NUM);}

// Similar to DoubleHex, both bits have a similar standing, with the
first of the two having a 0 to 9 range, causing only a manipulation of value
via % 10 to be sufficient
// While the second bit, as stated previously sharing a similar standing,
needs to undergo the same modification but first divided by 10. Ergo
(value/10)%10
```

LEEDS

terasic

Designed & Manufactured by Terasic

Download DE1-SoC CD from



```
else{DE1SoC_SevenSeg_SetSingle(display,value % 10);  
DE1SoC_SevenSeg_SetSingle(display+1,(value / 10) % 10);}}
```

IMAGES – STEP 1

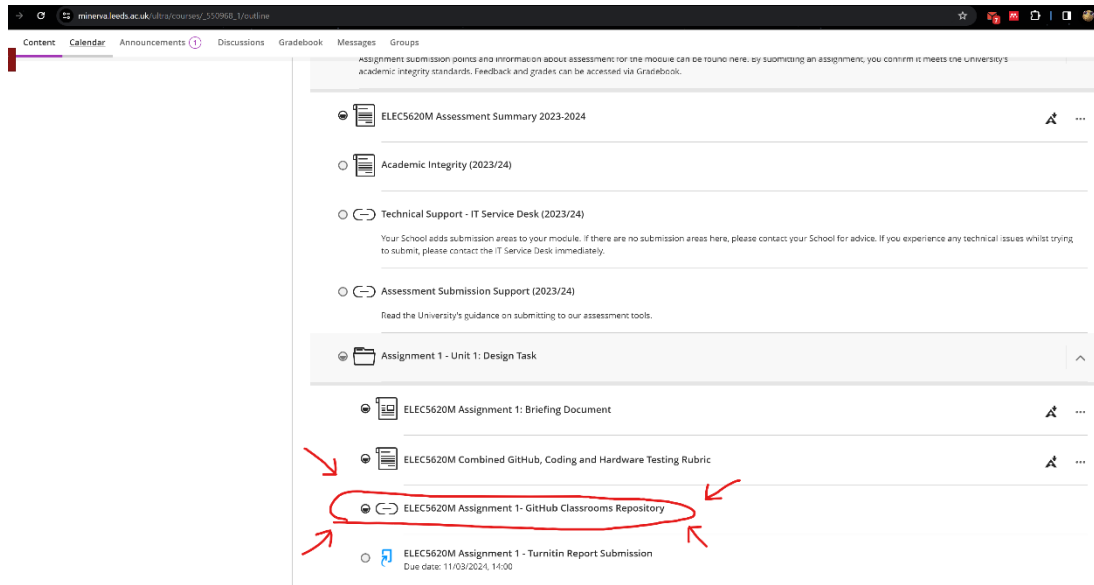


Figure 2 - Step 1.1: Cloning the Project Repository

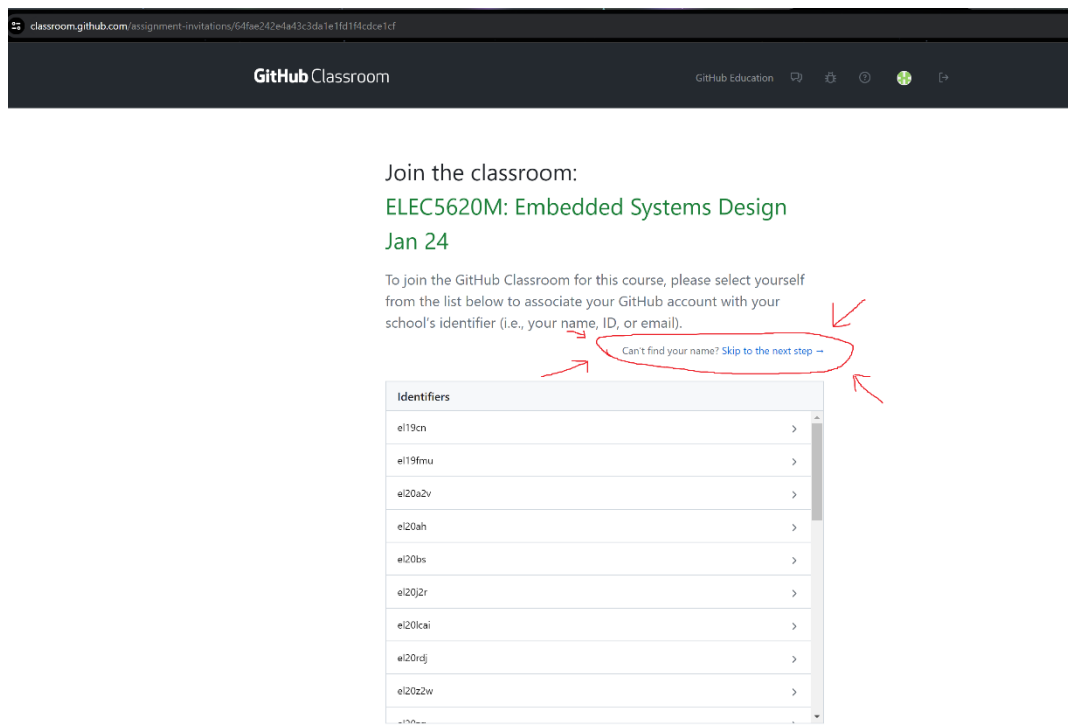


Figure 1 - Step 1.2: Cloning the Project Repository





IMAGES – STEP 2

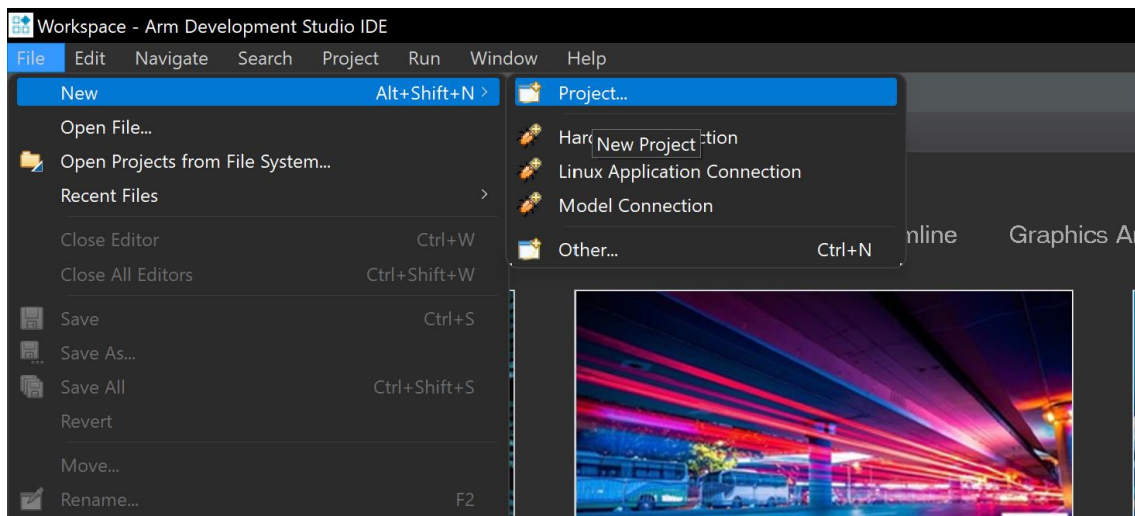


Figure 3 - Step 2.1: Creation of Arm Development Studio Project

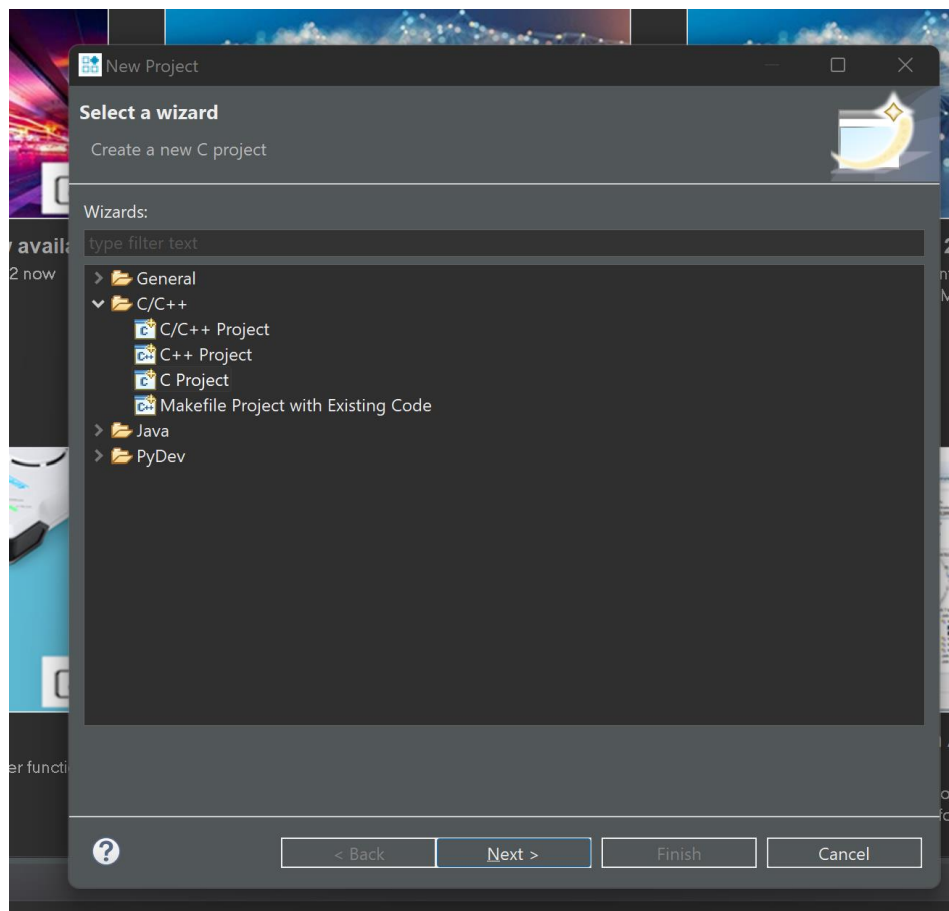


Figure 4 - Step 2.2: Selection of C Project



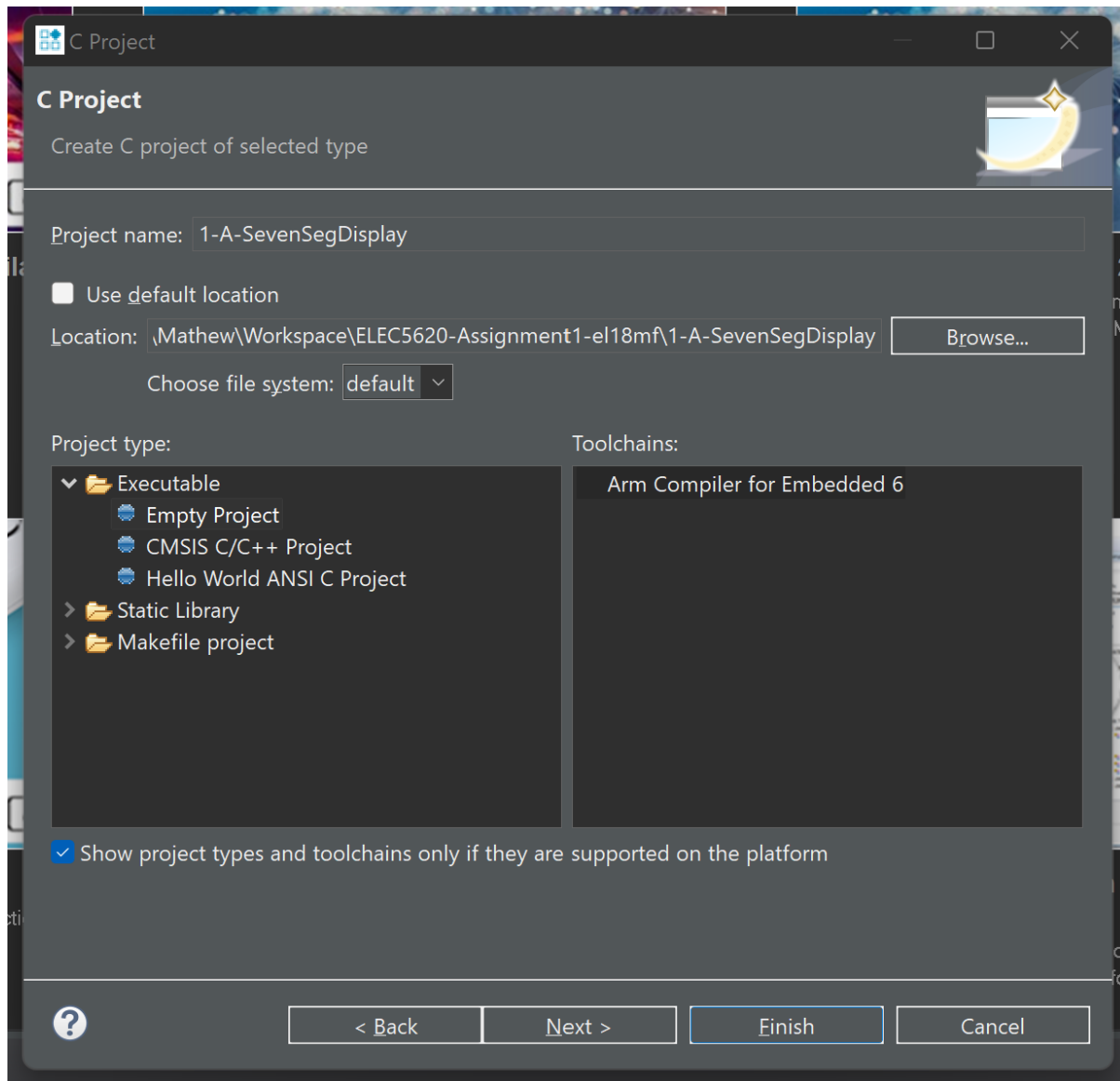


Figure 5 – Step 2.3: Project Name and Location Settings



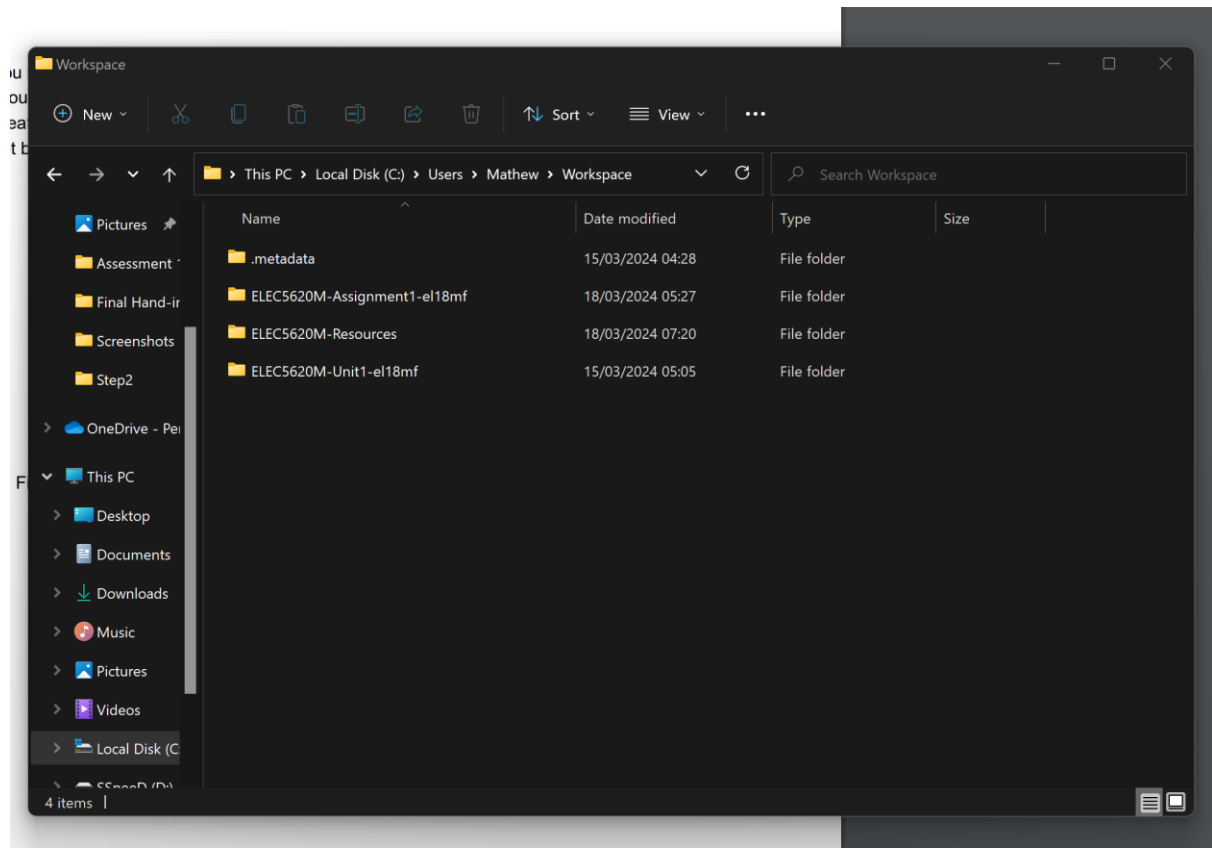


Figure 6 - Step 2.4: Inspecting the location to ensure it is correct



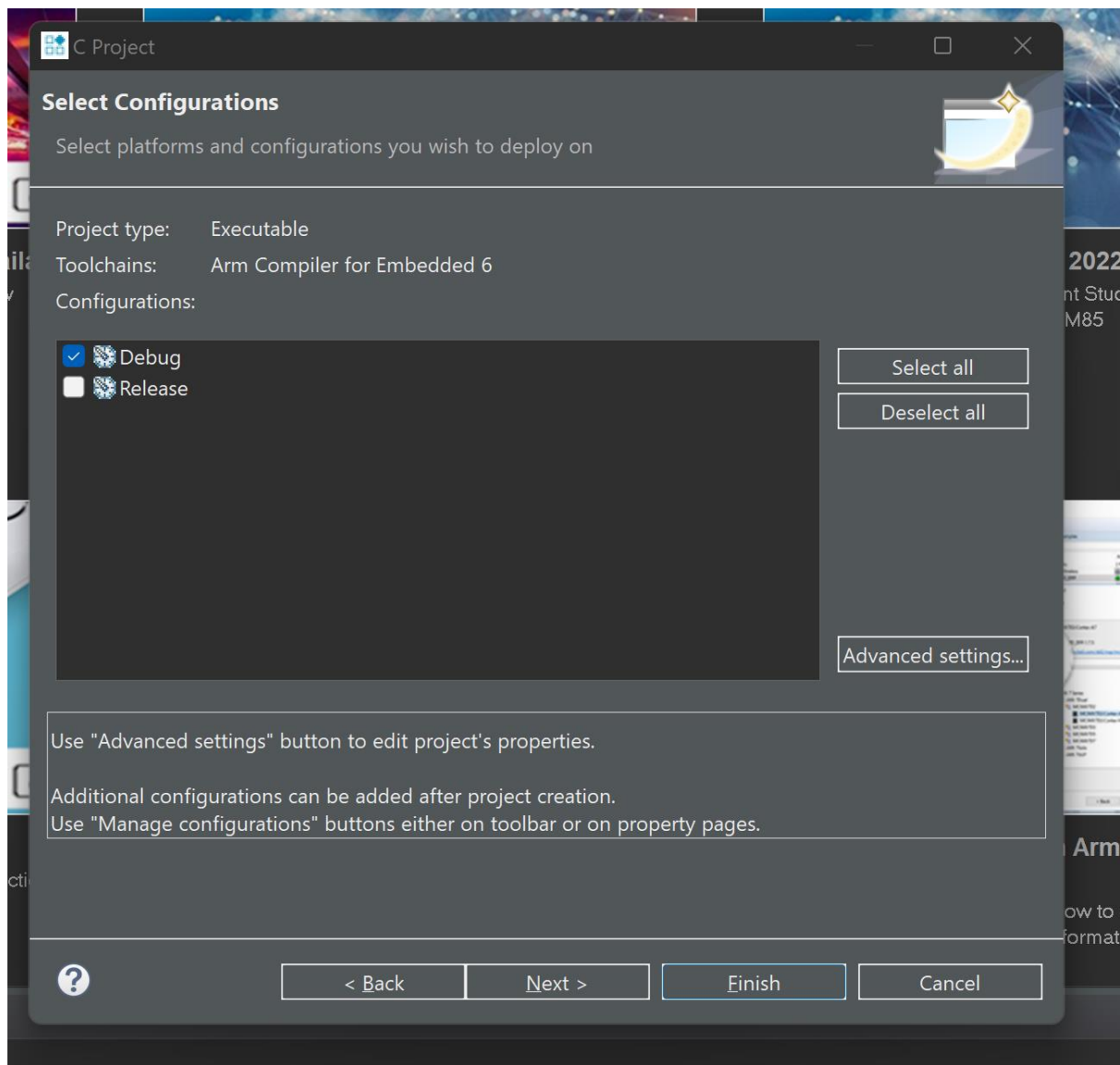


Figure 7 - 2.5: Configuration Selection – Debug



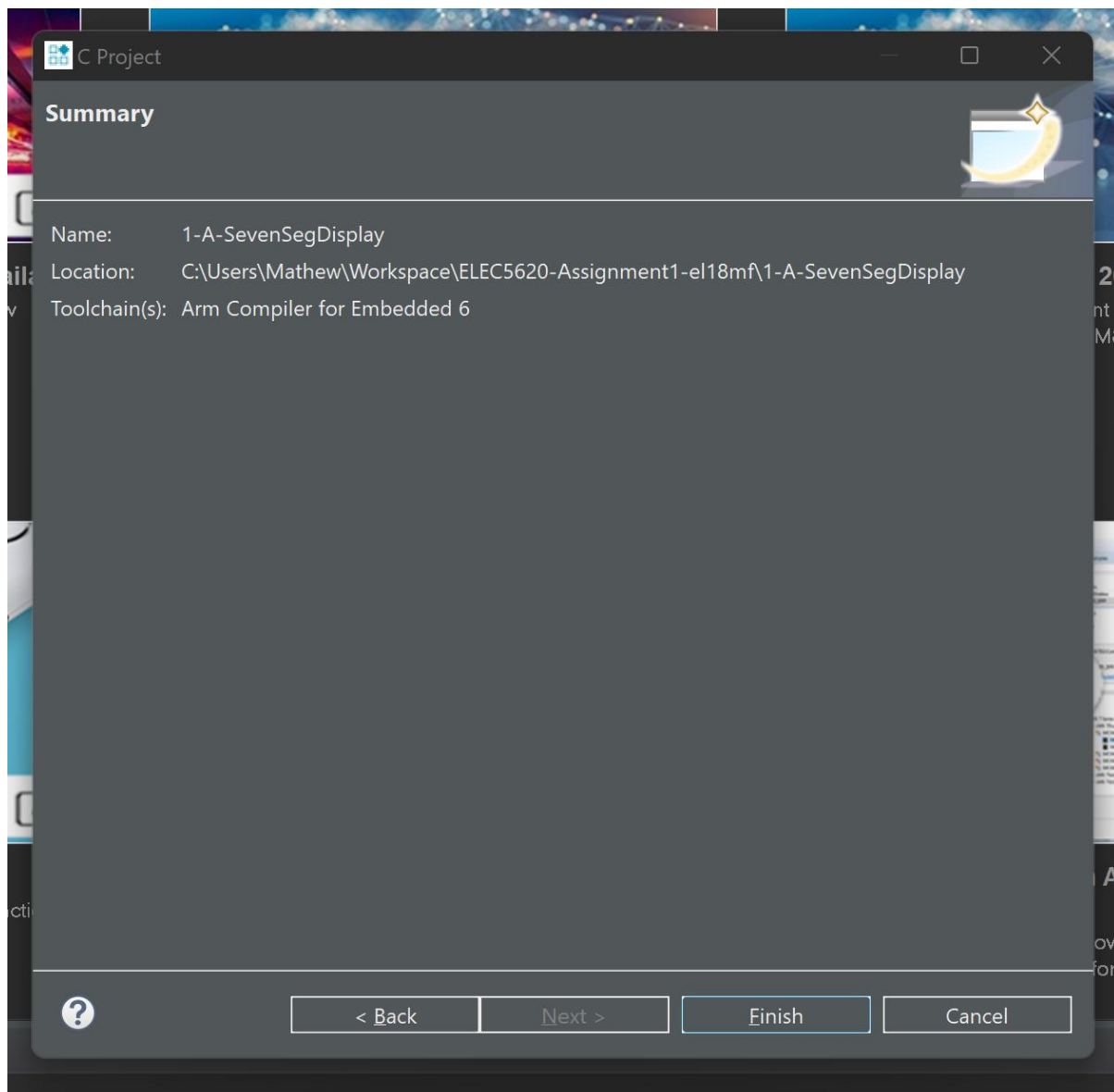


Figure 8 - 2.6: Project Summary

IMAGES – STEP 3



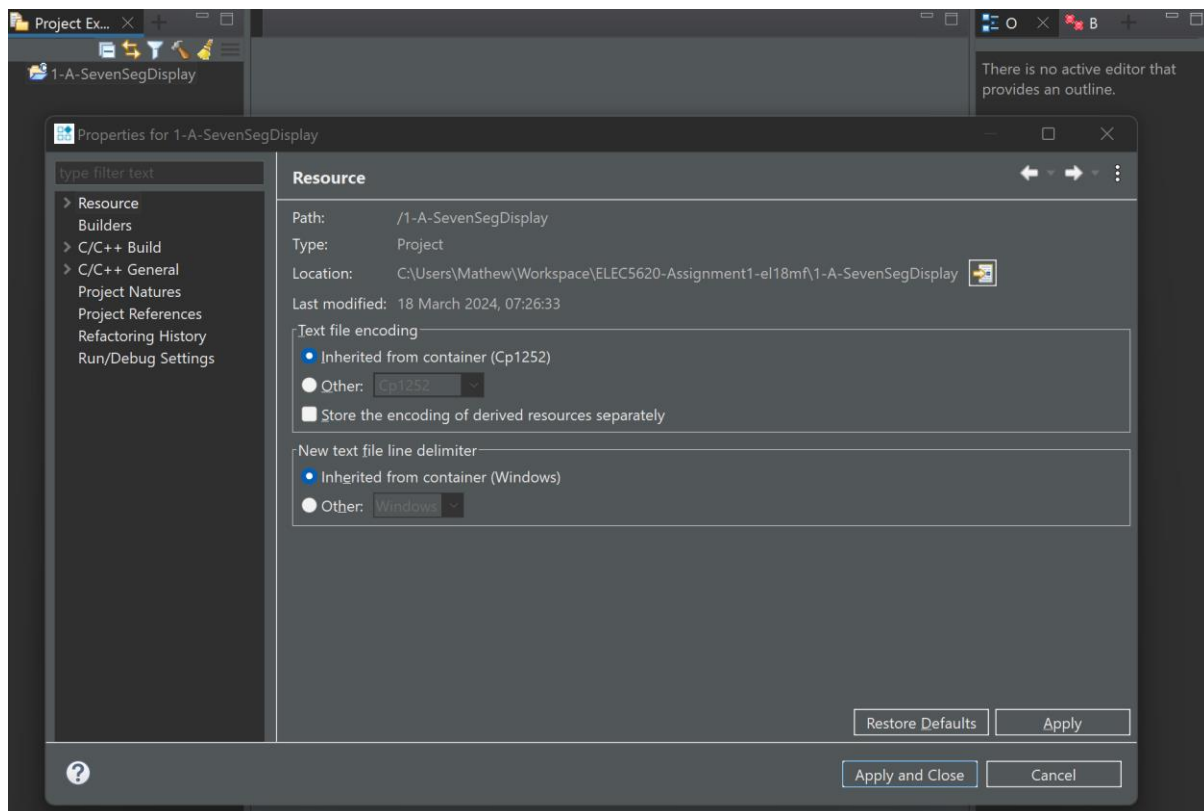


Figure 9 - 3.1: Arm Development Studio Configuration

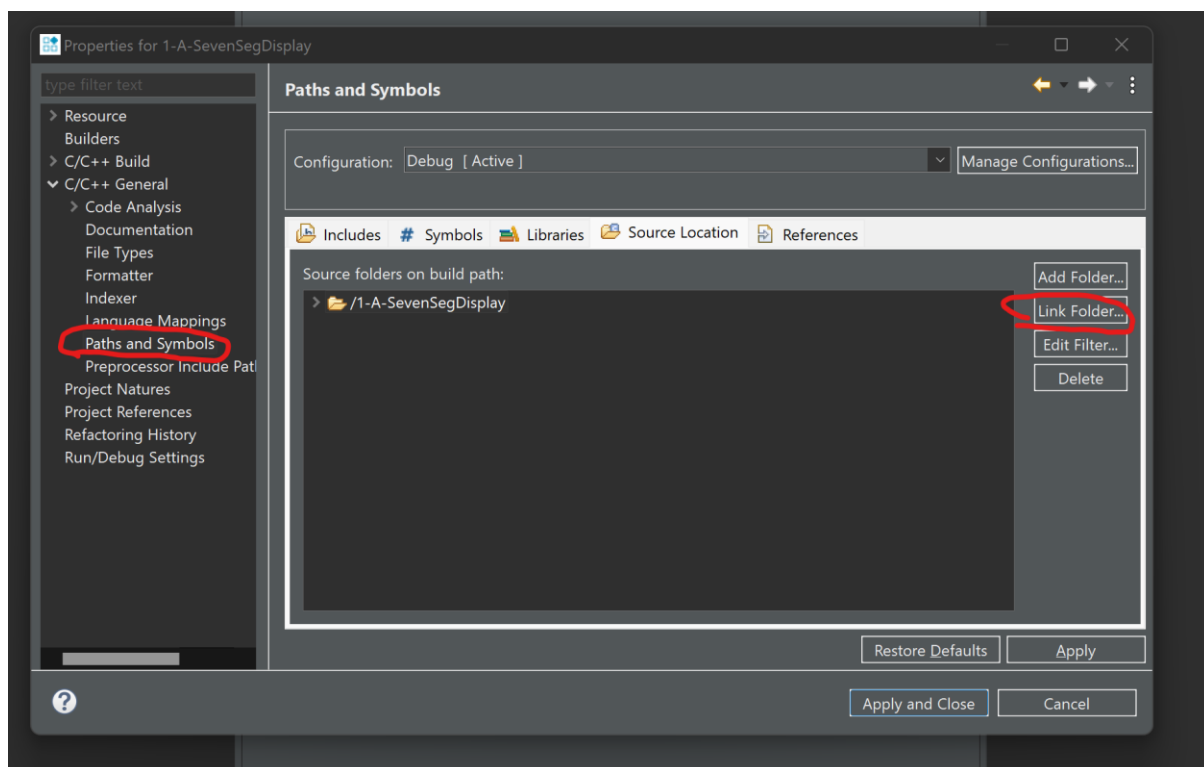


Figure 10 - 3.2: Locating Paths and Symbols folder linking



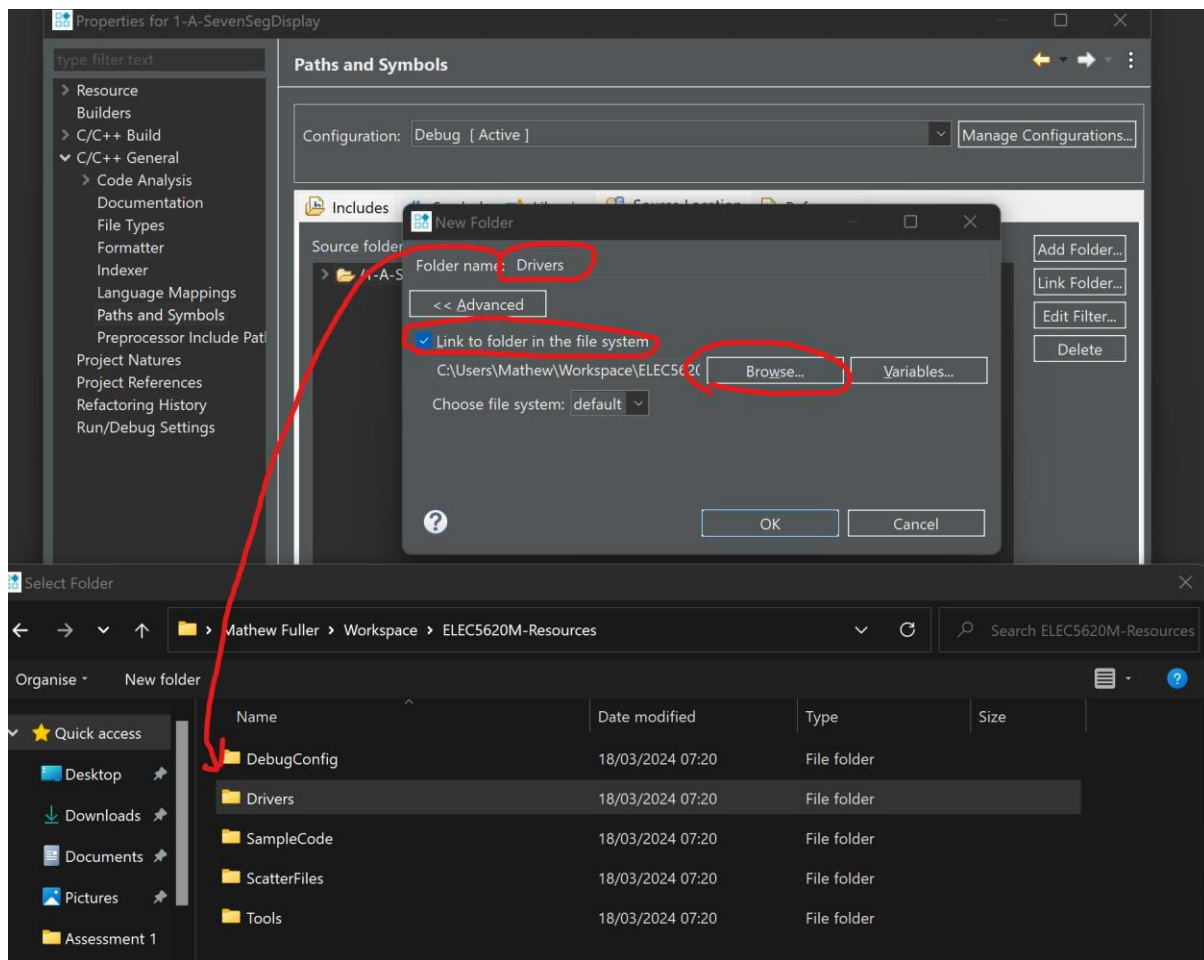


Figure 11 - 3.3: Linking Drivers Folder to Arm Dev Project



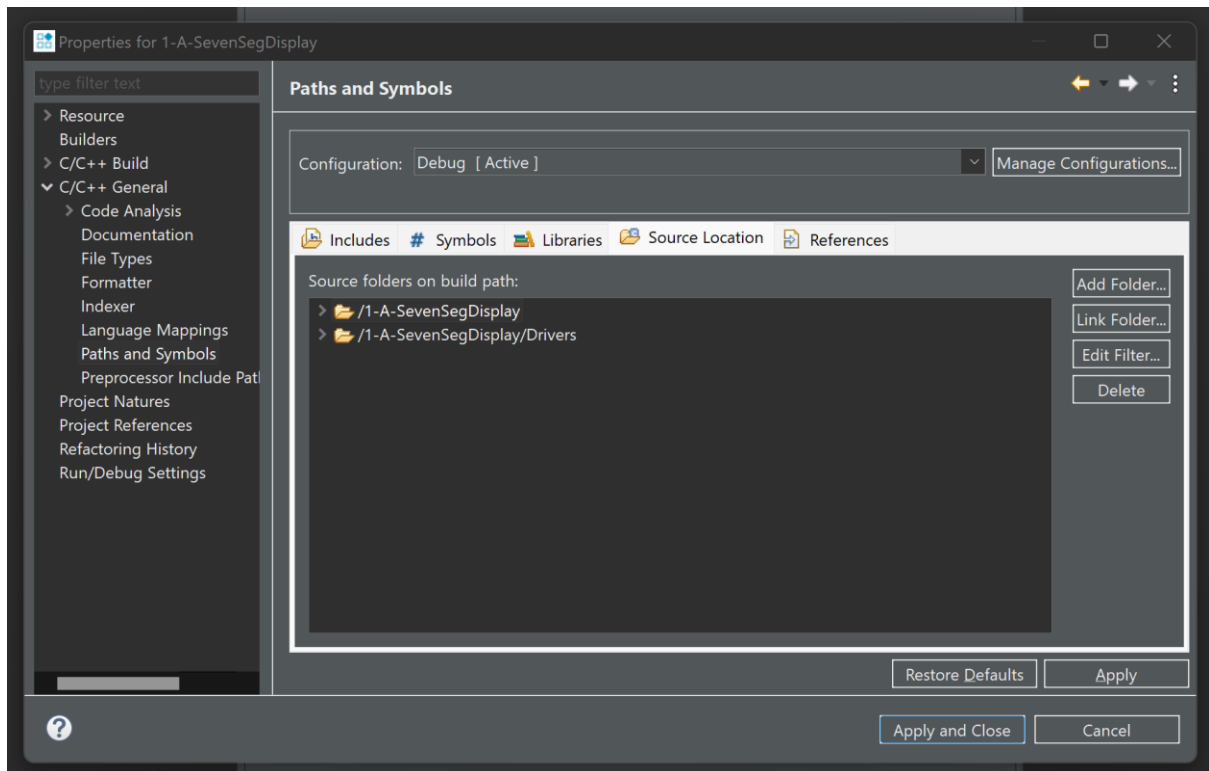


Figure 12 - 3.4: Drivers Folder Successfully Linked

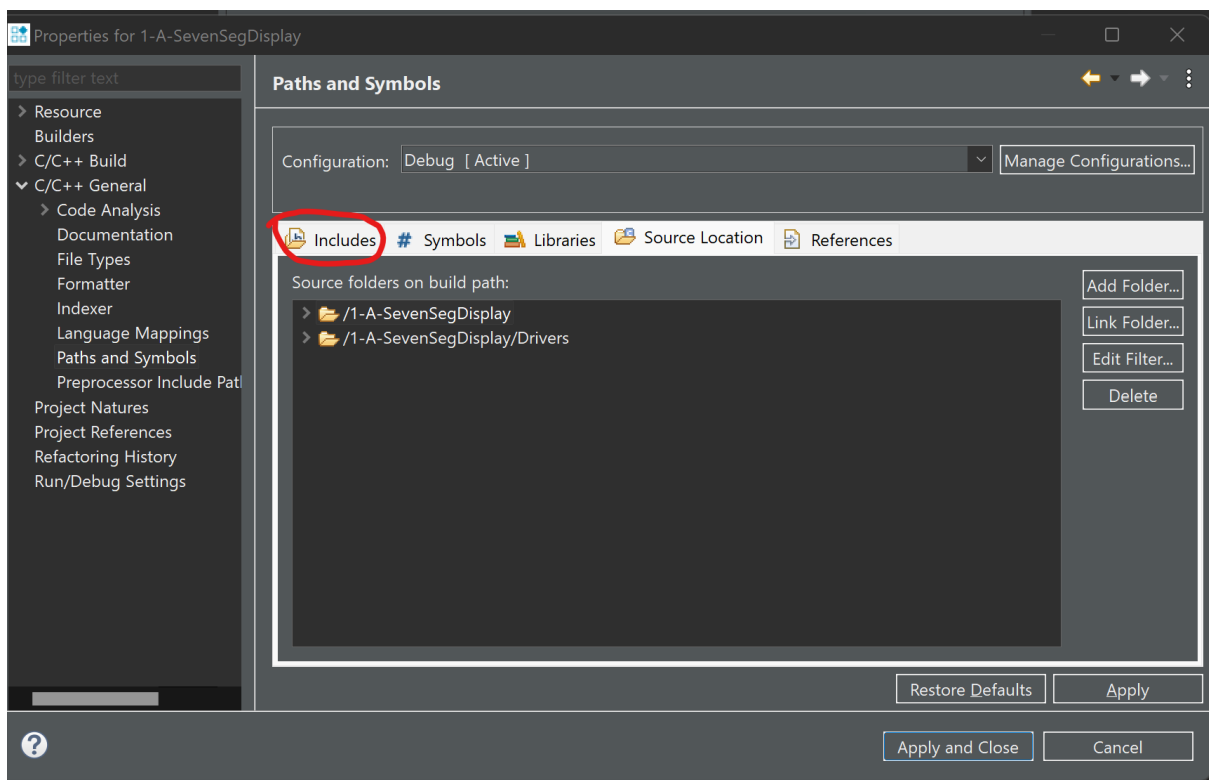


Figure 13 - 3.5: Locate the Includes area of Paths and Symbols



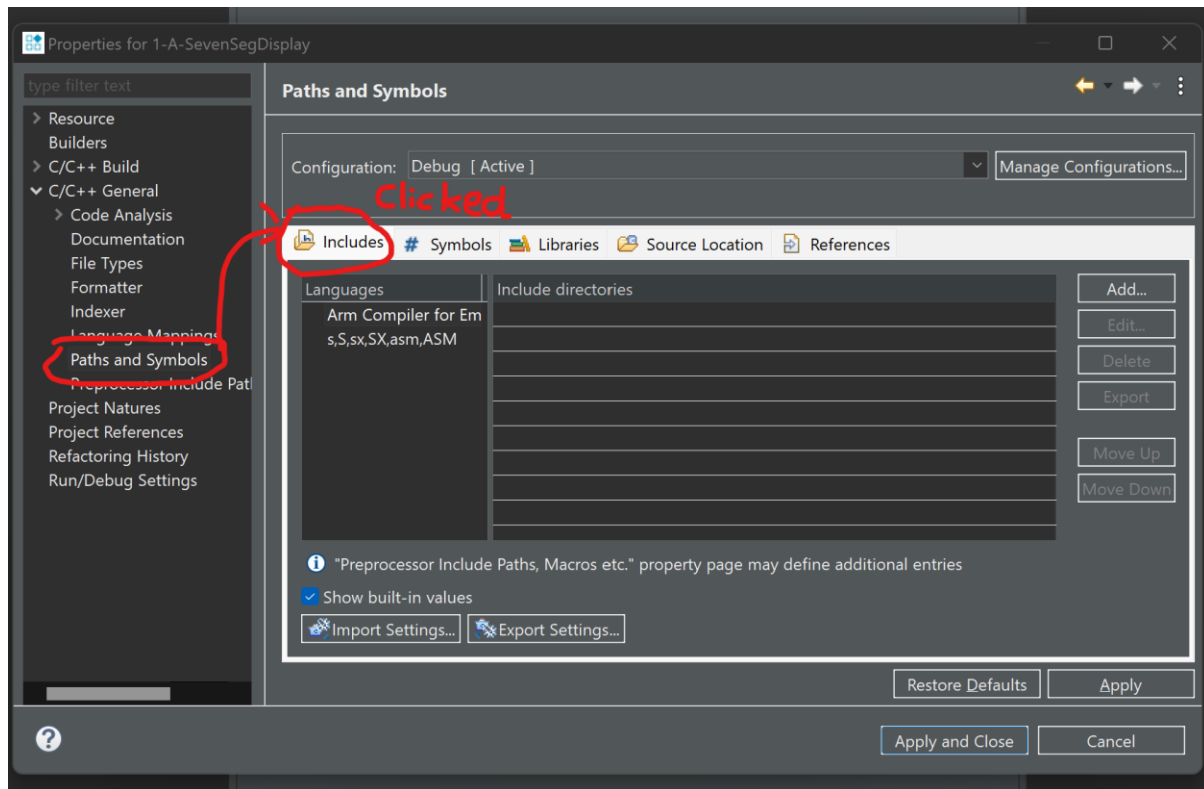


Figure 14 - 3.6: Enter the Includes section in Paths and Symbols



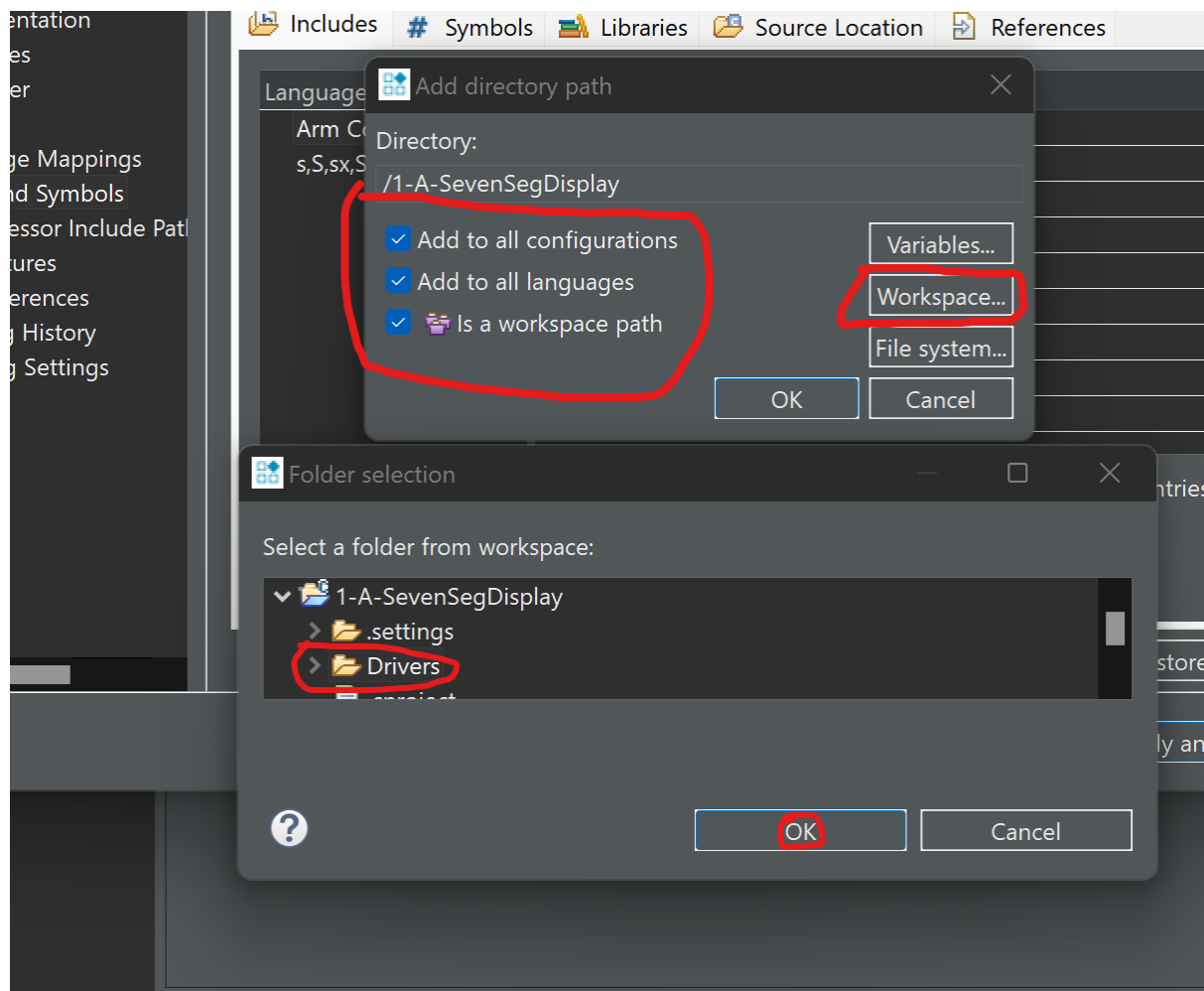


Figure 15 - 3.7: Inclusion of the drivers folder for use for all project within workspace



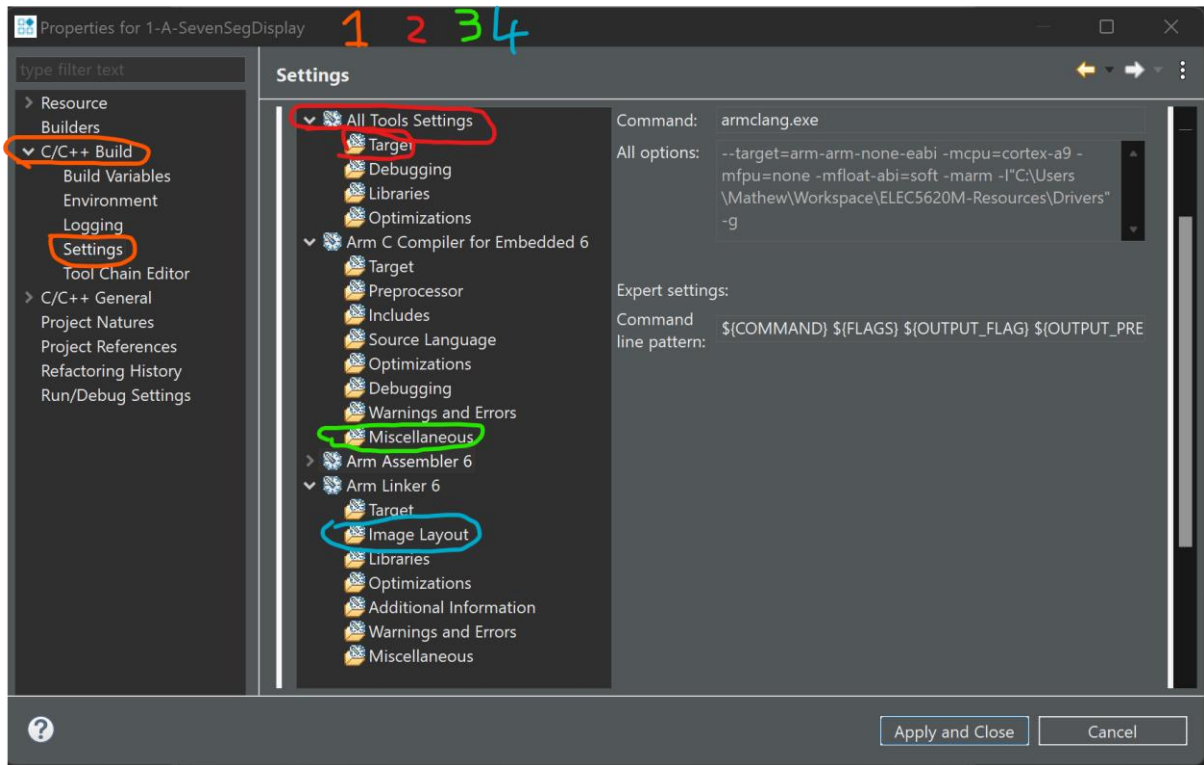


Figure 16 - 3.8: Compiler Configuration


All Tool Settings	Parameter	Value
» Target	Target CPU	Cortex-A9
» Target	Target FPU	No FPU
Arm Compiler for Embedded 6	Parameter	Value
» Miscellaneous	Other Flags	Click  then type -mno-unaligned-access
Arm Linker 6	Parameter	Value
» Image Layout	Image entry Point	<u><u>vector</u></u> _table <i>Note the double underscore</i>
» Image Layout	Scatter file	See below.

Table 5.1: Tool Settings for Cyclone V SoC Device - Leave Other Settings Unchanged

Scatter file: `${workspace_loc}\ELEC5620M-Resources\ScatterFiles\FPGARomRam.scat`.

Figure 17 - 3.9: Compiler configuration settings



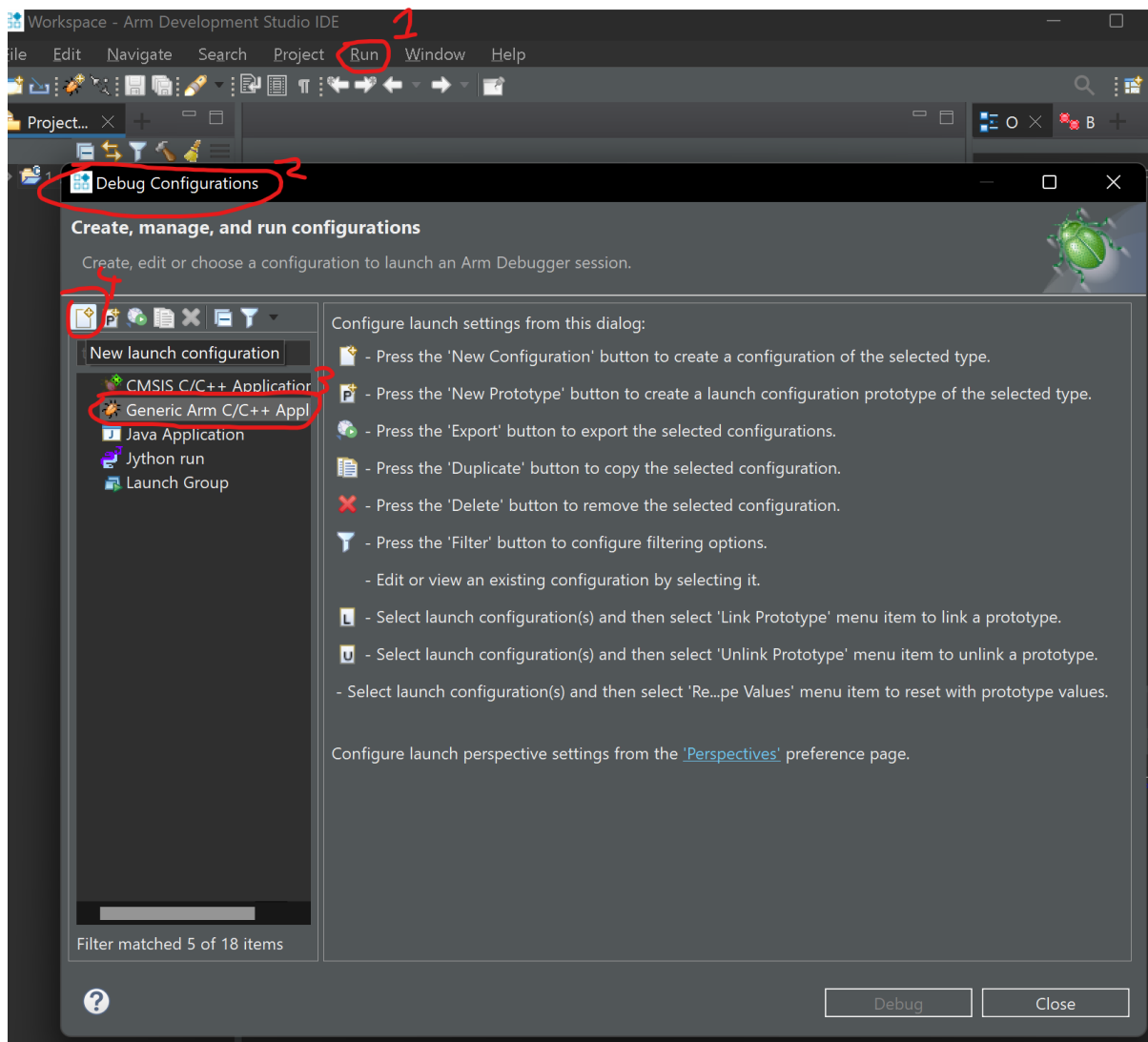


Figure 18 - 3.10: Debug Configurations





Connections Tab	Figure 6.2
Select Target	Intel SoC FPGA » Cyclone V SoC (Single Core) » Bare Metal Debug » Debug Cortex-A9_0 You can search in for Intel SoC to quickly filter the list of devices.
Target Connection	USB-Blaster
DSTL Options	Do Not Modify These
Connection	Click Browse... , choose DE-SoC on localhost [USB-1] and then click Select . Note: The DE1-SoCs USB Blaster port must be connected to the computer and DE1-SoC powered on else no connection will be found.
Files Tab	Figure 6.3
Target Connection	Click Workspace... , find <ProjectName>/ , then Debug/ then <ProjectName>.axf
Load Symbols	Ensure box is selected (checked)
Files	From the dropdown menu, select <i>Add peripheral description files from directory</i> . Then click Workspace... , find DebugConfig/ , select synthesis/ , then OK .
Debugger Tab	Figure 6.4
Run Control	Select <i>Debug from symbol</i> , and enter in to the box: main
Debug Init. Script	Select the <i>Run target initialisation debugger script (.ds/.py)</i> option. Then click Workspace... , find DebugConfig/ , select debugload.ds , then OK .

Figure 19 – Step 3.11: Debug Configuration Settings

```

CDT Build Console [1-A-SevenSegDisplay]
09:02:51 **** Build of configuration Debug for project 1-A-SevenSegDisplay ****
make all
'Building file: C:/Users/Mathew/Workspace/ELEC5620M-Resources/Drivers/Util/driver_crc.c'
'Invoking: Arm C Compiler for Embedded 6'
armclang.exe --target=arm-arm-none-eabi -mcpu=cortex-a9 -mfpu=none -mfloat-abi=soft -marm -I"C:\Use
armclang: error: Failed to check out a license.
Unable to connect to the license server. Check that ARMLMD_LICENSE_FILE is set correctly, and the l
armclang: note:
Information about this error is available at: http://ds.arm.com/support/lic56/m15
General licensing information is available at: http://ds.arm.com/support/licensing/

```

Figure 20 - Step 3.12: Debugging and Testing

IMAGES – STEP 4



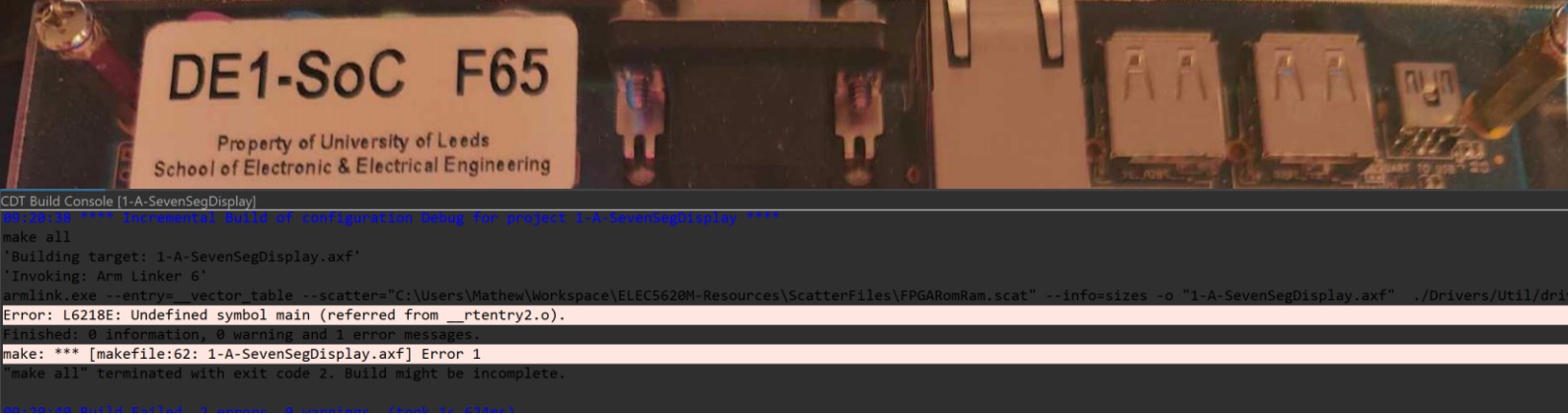


Figure 21 - 4.1: Compile Error

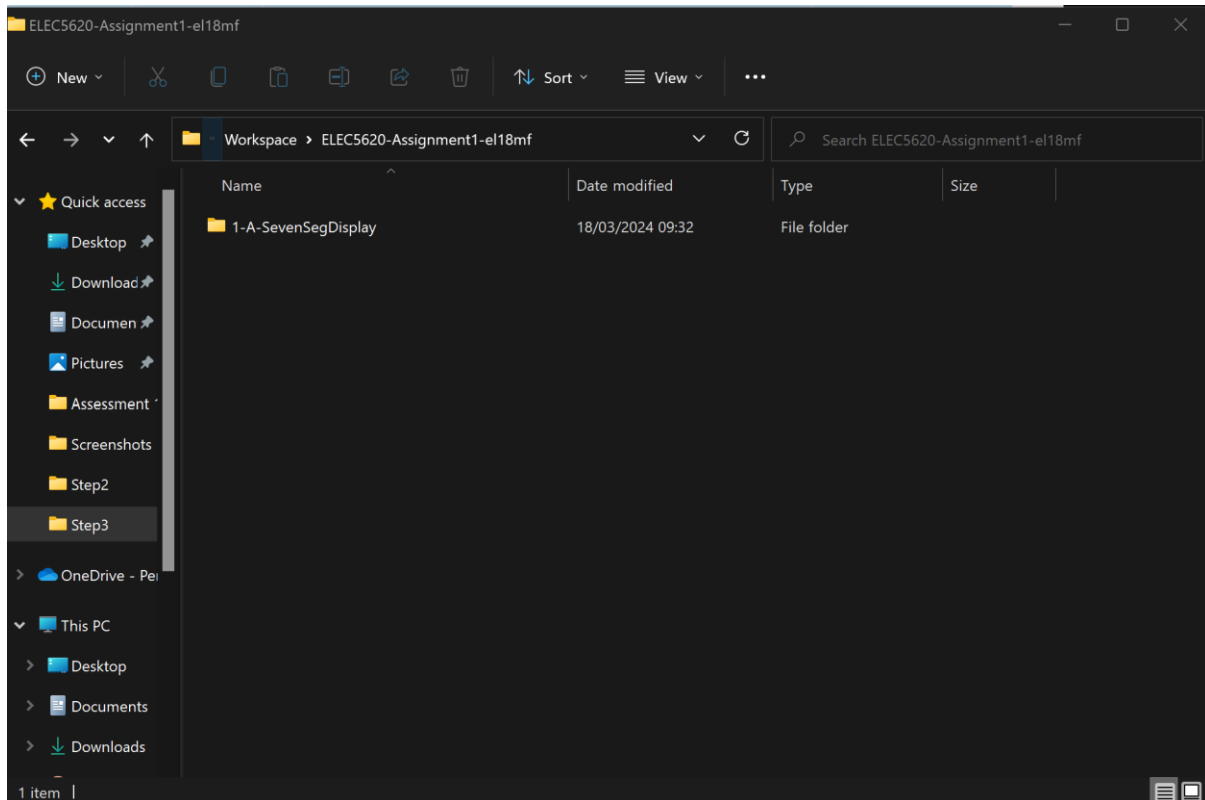


Figure 212 - 4.2: Location of Error - Pre-solution

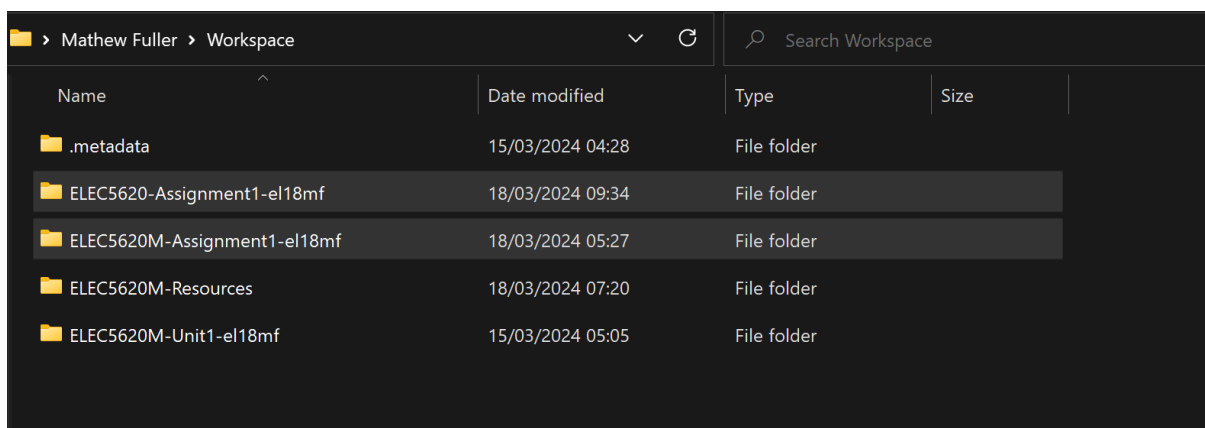


Figure 223 - 4.3: Solved Error - Copied bottom highlighted folder into the top





Figure 234 - 4.4: Hardware Testing of Buttons at limit



Figure 245 - 4.5: Displays correct functioning of the out-of-range values on the display (it lights up the middle bar solely)

IMAGES – STEP 5

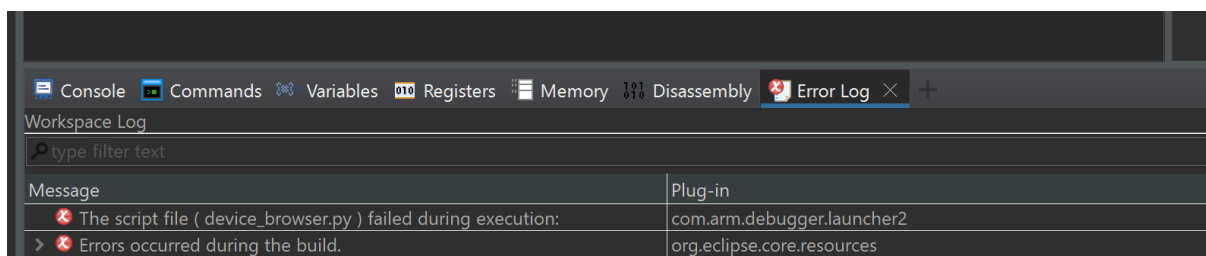


Figure 26 - 5.1: Initial occurrence of repeating with missing files error





```
=====

Code (inc. data)  RO Data  RW Data  ZI Data  Debug

11848      554      672      100      65904      17586  Grand Totals
11848      554      672      100      65904      17586  ELF Image Totals
11848      554      672      100           0           0  ROM Totals

=====

Total RO  Size (Code + RO Data)           12520 ( 12.23kB)
Total RW  Size (RW Data + ZI Data)        66004 ( 64.46kB)
Total ROM Size (Code + RO Data + RW Data)  12620 ( 12.32kB)

=====

'Finished building target: 1-A-SevenSegDisplay.axf'
''

11:47:23 Build Finished. 0 errors, 0 warnings. (took 2s.815ms)
```

Figure 257 - Step 5.2: Successful compilation sans errors - pre-mistake corrected

```
=====

Code (inc. data)  RO Data  RW Data  ZI Data  Debug

11848      554      672      100      65904      17586  Grand Totals
11848      554      672      100      65904      17586  ELF Image Totals
11848      554      672      100           0           0  ROM Totals

=====

Total RO  Size (Code + RO Data)           12520 ( 12.23kB)
Total RW  Size (RW Data + ZI Data)        66004 ( 64.46kB)
Total ROM Size (Code + RO Data + RW Data)  12620 ( 12.32kB)

=====

'Finished building target: 1-A-SevenSegDisplay.axf'
''

11:32:46 Build Finished. 0 errors, 0 warnings. (took 2s.860ms)
```

Figure 268 – Step 5.3: Successful, error free compilation post mistake fix

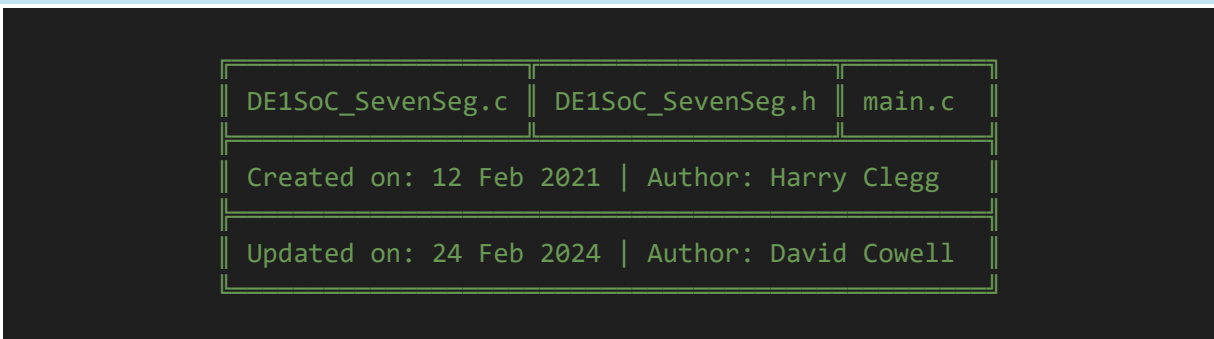
IMAGES – STEP 6 & 7





Figure 279 - Step 7.1: Successful compilation and testing of both Step 6 & 7 (did both simultaneously as time was too short to do it individually)

REFERENCES & ACKNOWLEDGEMENT



Citations for references used in code: [1] [2] [3]

REFERENCES

- [1] F4NT0, "stackoverflow," Stack Exchange Inc., 5 September 2022. [Online]. Available:
] <https://stackoverflow.com/questions/11509830/how-to-add-color-to-githubs-readme-md-file#:~:text=You%20can%20however%20add%20color,or%20%22code%22%20tags%20needed..> [Accessed 16 3 2024].
- [2] A. Pritchard, "GitHub," GitHub Inc., 27 May 2022. [Online]. Available:
] <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>. [Accessed 17 March 2024].
- [3] uupaa, "GitHub," GitHub Inc., 26 July 2016. [Online]. Available:
] <https://gist.github.com/uupaa/f77d2bcf4dc7a294d109#file-image-resize-in-github-flavored-markdown-md>. [Accessed 17 March 2024].

