| Ousmane Toure & Jianning Chen  EECE2160 | Embedded Design: Enabling Robotics  Prelab Assignment 7 |
| --- | --- |

Prelab Assignment 7

Ousmane Toure & Jianning Chen

[Toure.o@northeastern.edu](mailto:Toure.o@northeastern.edu)

[chen.jiann@northeastern.edu](mailto:chen.jiann@northeastern.edu)

Submit date: 7/14/2022

Due Date: 7/14/2022

**7.0.1 Hexadecimal digits to display on the 7-segment displays**

Table 7.1

| **Character** | **OUTPUTS - Segment 0/1** | | | | | | |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **Decimal** | **Hex** |
| **0** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 63 | 0x3F |
| **1** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 6 | 0x06 |
| **2** | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 91 | 0x5B |
| **3** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 79 | 0x4F |
| **4** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 102 | 0x66 |
| **5** | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 109 | 0x6D |
| **6** | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 125 | 0x7D |
| **7** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 7 | 0x07 |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 127 | 0x7F |
| **9** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 111 | 0x6F |
| **A** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 119 | 0x77 |
| **B** | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 124 | 0x7C |
| **C** | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 57 | 0x39 |
| **D** | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 94 | 0x5E |
| **E** | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 121 | 0x79 |
| **F** | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 113 | 0x71 |

**7.0.2 Where is figure 1 found?**

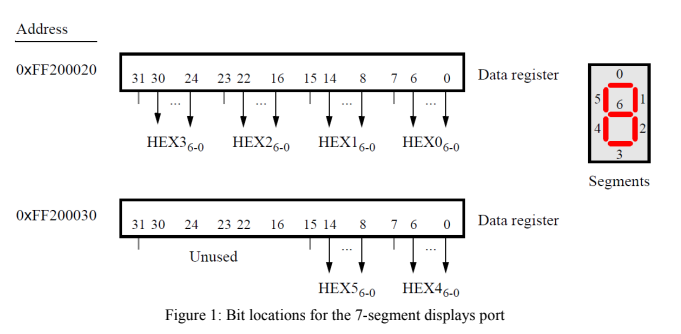
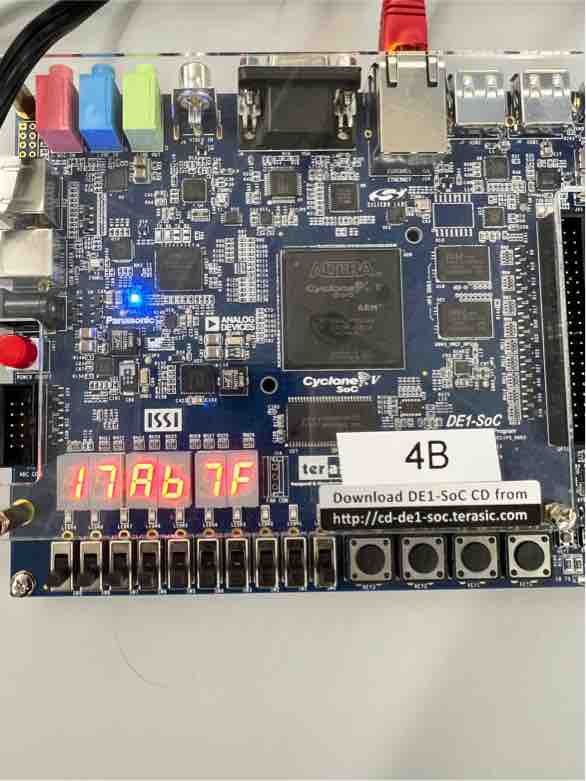
****

Figure 1 is found on page 8 of *DE1-SoC\_Computer\_System\_with\_ARM\_Cortex\_A9.pdf*.

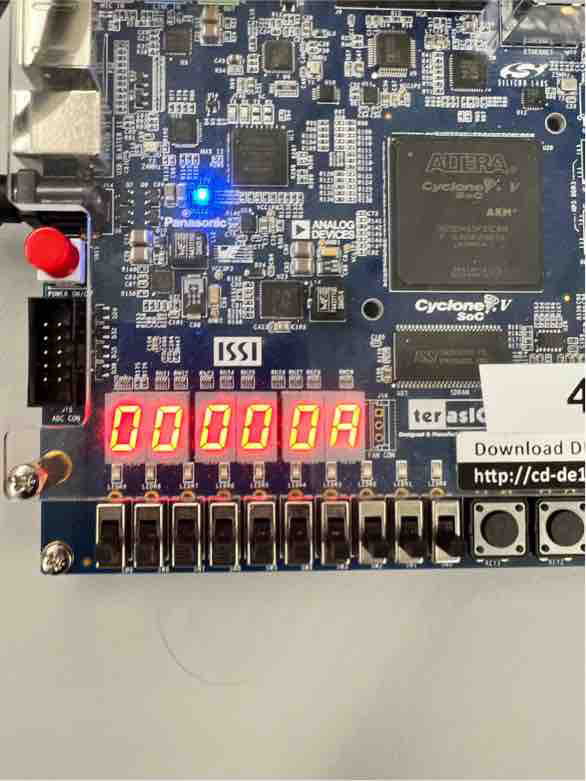
**7.0.3**

Picture of number input 1551231:

Picture of number input 255:



Picture of number 10:



**Code:**

**#include <iostream>**

**#include "SevenSegment.h"**

**#include "DE1SoCfpga.h"**

**SevenSegment::SevenSegment(){**

**reg0\_hexValue = (bit\_values[0] << 24) | (bit\_values[0] << 16) | (bit\_values[0] << 8) | bit\_values[0];**

**reg1\_hexValue = (bit\_values[0] << 8) | bit\_values[0];**

**RegisterWrite(HEX3\_0\_OFFSET,reg0\_hexValue);**

**RegisterWrite(HEX5\_4\_OFFSET,reg1\_hexValue);**

**}**

**SevenSegment::~SevenSegment() {**

**Hex\_ClearAll();**

**}**

**void SevenSegment::Hex\_ClearAll() {**

**reg0\_hexValue = 0x3F3F3F3F;**

**reg1\_hexValue = 0x3F3F3F3F;**

**RegisterWrite(HEX3\_0\_OFFSET,reg0\_hexValue);**

**RegisterWrite(HEX5\_4\_OFFSET,reg1\_hexValue);**

**}**

**void SevenSegment::Hex\_ClearSpecific(int index) {**

**if(index == 0){reg0\_hexValue = reg0\_hexValue & 0x00F;}**

**else if(index == 1){reg0\_hexValue = reg0\_hexValue & 0x0F0;}**

**else if(index == 2){reg0\_hexValue = reg0\_hexValue & 0xF00;}**

**else if(index == 3){reg0\_hexValue = reg0\_hexValue & 0xF000;}**

**else if(index == 4){reg1\_hexValue = reg1\_hexValue & 0x00F;}**

**else if(index == 5){reg1\_hexValue = reg1\_hexValue & 0x0F0;}**

**else{**

**std::cerr << "Index exceeding 5 or less than 0!" << std::endl;**

**exit(1);**

**}**

**}**

**void SevenSegment::Hex\_WriteNumber(int number) {**

**int temp;**

**temp = number;**

**int store[6] = {0,0,0,0,0,0};**

**int remainder;**

**int count = 0;**

**while(temp > 0){**

**remainder = temp%16;**

**store[count] = remainder;**

**std::cout << remainder;**

**++count;**

**temp = temp/16;**

**}**

**reg0\_hexValue = 0x00;**

**reg1\_hexValue = 0x00;**

**reg0\_hexValue = (bit\_values[store[3]] << 24) | (bit\_values[store[2]] << 16) | (bit\_values[store[1]] << 8) | bit\_values[store[0]];**

**reg1\_hexValue = (bit\_values[store[5]] << 8) | bit\_values[store[4]];**

**RegisterWrite(HEX3\_0\_OFFSET,reg0\_hexValue);**

**RegisterWrite(HEX5\_4\_OFFSET,reg1\_hexValue);**

**}**

**void SevenSegment::Hex\_WriteSpecific(int index, int value) {**

**int temp;**

**if(value > 15 || value < -16) {**

**std::cerr << "Value exceeding boundary!" << std::endl;**

**exit(1);**

**}**

**else if(value >= 0) {temp = value;}**

**else{temp = 16+value;}**

**temp = value;**

**if(index < 4){reg0\_hexValue = reg0\_hexValue | (bit\_values[temp] << 8\*index);}**

**else{reg1\_hexValue = reg1\_hexValue | (bit\_values[temp] << 8\*(index-4));}**

**RegisterWrite(HEX3\_0\_OFFSET,reg0\_hexValue);**

**RegisterWrite(HEX5\_4\_OFFSET,reg1\_hexValue);**

**}**

**/\***

**int SevenSegment::PushButtonGet() {**

**int value = this->RegisterRead(pBase,KEY\_OFFSET);**

**return value;**

**}**

**\*/**

**DE1SoCfpga::DE1SoCfpga(){**

**fd = open("/dev/mem",(O\_RDWR | O\_SYNC));**

**if(fd == -1){**

**std::cout << "ERROR: could not open /dev/mem..." << std::endl;**

**exit(1);**

**}**

**char \*virtual\_base = (char \*)mmap(NULL, LW\_BRIDGE\_SPAN,(PROT\_READ | PROT\_WRITE), MAP\_SHARED, fd, LW\_BRIDGE\_BASE);**

**if(virtual\_base == MAP\_FAILED){**

**std::cout << "ERROR: mmap() failed..." << std::endl;**

**close(fd);**

**exit(1);**

**}**

**pBase = virtual\_base;**

**}**

**DE1SoCfpga::~DE1SoCfpga() {**

**if(munmap(pBase,LW\_BRIDGE\_SPAN) != 0){**

**std::cout << "ERROR: munmap() failed..." << std::endl;**

**exit(1);**

**}**

**close(fd);**

**}**

**//int DE1SoCfpga::RegisterRead(unsigned int offset) {**

**//return \*(volatile unsigned int \*)(pBase + offset);**

**//}**

**void DE1SoCfpga::RegisterWrite(unsigned int offset, unsigned int value) {**

**\*(volatile unsigned int \*)(pBase + offset) = value;**

**}**

**int main()**

**{**

**SevenSegment \*display = new SevenSegment;**

**std::cout << "Program Starting...!" << std::endl;**

**bool condition = true;**

**int hex\_value;**

**while(condition){**

**std::cout << "Please enter a number for display (in the range -16777216 to 16777215):";**

**std::cin >> hex\_value;**

**display->Hex\_WriteNumber(hex\_value);**

**std::cout << "Do you want to stop? ";**

**char v;**

**std::cin >> v;**

**if(v == 'y'){condition = false;}**

**}**

**std::cout << "Terminating...!" << std::endl;**

**delete display;**

**return 0;**

**}**

# References

1. Prof. Julius Marpaung, “*Lab Report Guide*”, Northeastern University, January 6 2020.