Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Lab Assignment 3: Memory Architecture**

1. **Program memory (Flash) and Data memory (RAM)**

In this lab, you will explore the program and data memory spaces of ATmega328P microcontroller used in the Arduino UNO.

The ATmega328P ([datasheet](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf)) microcontroller has 3 types of memory which are on chip and is designed along the lines of the Harvard architecture. The types of memory are,

* Flash memory (program space) is where Arduino sketch (program) is stored.
* SRAM (static random-access memory) is where sketch (program) creates and manipulates variables when it runs.
* EEPROM is memory space that programmers can use to store long-term information.

Diagram

Description automatically generated

E2END

Three types of memory spaces in the Arduino UNO are as shown in above Figure. Notice that addresses for each of the three memory spaces start with 0x0000 and go to a symbolic constant as the end address - FLASHEND, RAMEND, or E2END. These constants are defined in the IDE compiler.

The contents of RAM are initialized during the power on sequence in the program code. Note that the 32 general purpose registers and the I/O device registers are “memory mapped” into the data memory address space from 0x0000 to 0x00ff. The program memory is organized as 16-bit words while the data memory and EEPROM are organized as bytes.

1. Create a new sketch in Arduino IDE and then use a Serial.println() command in the setup function to print the hex values of the three constants which the end address of each of the memory spaces.
2. Upload the sketch and fill in the values you get from Serial Monitor below

|  |  |  |
| --- | --- | --- |
|  | HEX | DEC |
| FLASHEND | 7FFF | 32767 |
| RAMEND | 8FF | 2303 |
| E2END | 3FF | 1023 |

2. Check the size of each memory against the information from <https://store.arduino.cc/usa/arduino-uno-rev3> and discuss if there is any dispensary.

There is no dispensary. The specs written are exactly the same as recorded.

**Writing Data into Flash Memory**

The [PROGMEM](https://www.arduino.cc/reference/en/language/variables/utilities/progmem/) keyword is a variable modifier, it should be used only with the datatypes defined in pgmspace.h. It tells the compiler "put this information into flash memory", instead of into SRAM, where it would normally go.

**Syntax:** const dataType variableName[] PROGMEM = {data0, data1, data3…​};

Using PROGMEM is also a two-step procedure. After getting the data into Flash memory, it requires special methods (functions) pgm\_read\_byte and pgm\_read\_word, also defined in the <pgmspace.h> library, to read the data from program memory back into SRAM, so we can do something useful with it.

In C programming, pointers can only access a single address space. In the Harvard architecture, the C pointer space is mapped to the data memory space since that is the most common use for pointers. Although a function can be executed via dereferencing a function pointer, dereferencing the function pointer cannot be used to read the contents of program memory space.

 // correct usage of a function pointer in either architecture

void (\*pointer)() = &functionName; //defines and initializes the value of a function pointer

(\*pointer) (); // calls functionName via the function pointer

3. Download the program Memorylab1.ino form the Canvas and try to compile that program. If any error, debug the error.

4. Check the number of bytes used in both storages from Arduino’s sketch IDE.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Max capacity | Used in bytes | Used in % |
| Flash | 32566 | 3500 | 10 |
| RAM | 2048 | 988 | 48 |

5. Check Memory capacity left for local variables and fill below.

30095 Bytes (Including both flash and RAM)

6. Upload the program into Arduino board and open Serial monitor. Fill the values/content of all variables/pointers called in both setup and testfun functions from the Serial monitor

|  |  |  |  |
| --- | --- | --- | --- |
| Variable name | Function Name it uses | Value in HEX | Value in DEC |
| g (global) | setup | 100 | 256 |
| a | setup | 22 | 34 |
| ptr | setup | 8FB | 2299 |
| \*ptr | setup | 22 | 34 |
| g (global) | testfun | 100 | 256 |
| a | testfun | 30 | 48 |
| b | testfun | 20 | 32 |

6. Fill the location (address) of all variables/pointers from both setup and testfun functions from the Serial monitor. Determine if it is inside Flash or RAM. If it is in RAM, determine if it is in heap/global variables (RAM\_H) or stack (RAM\_S).

|  |  |  |  |
| --- | --- | --- | --- |
| Variable name | Function Name it uses | Address in HEX | Location (Flash, RAM\_H, RAM\_S) |
| g (global) | setup | 100 | RAM\_H |
| a | setup | 8E1 | RAM\_S |
| ptr | setup | 8FB | RAM\_S |
| \*ptr | setup | 8F9 | RAM\_S |
| string1 | setup | 68 | Flash |
| string2 | setup | 119 | RAM\_H |
| string3 | setup | 8D6 | RAM\_S |
| string4 | setup | 4DE | RAM\_S |
| testfun |  | 1C8 | Flash |
| fun\_ptr |  | 1C8 | Flash |
| g (global) | testfun | 100 | RAM\_H |
| a | testfun | 8E1 | RAM\_S |
| b | testfun | 8E2 | RAM\_S |

7. Download the program Memorylab2.ino form the Canvas and try to compile that program. If any error, explain about the error the program shows. And check the used capacity of both storages by compiling the program again.

Used too much memory so the code did not upload.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Max capacity | Used in bytes | Used in % |
| Flash | 32256 | 3888 | 12 |
| RAM | 2048 | 2492 | 121 |

8. Use PROGMEM to save all variables in the program memory (flash memory) instead of RAM and check the used capacity of both storages by compiling the program again. (<https://www.arduino.cc/reference/en/language/variables/utilities/progmem/>)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Max capacity | Used in bytes | Used in % |
| Flash | 32256 | 3888 | 12 |
| RAM | 2048 | 188 | 9% |

6. Run the program and check the output from Serial Monitor. What does the program display and why?

All the variables are now stored in the flash storage which means that the program does not fit in the RAM.

7. Modify the program to display the content from the program memory at Serial monitor using the snippet below. **Show the result to the instructor or TAs.**

strcpy\_P(buffer, (char \*)pgm\_read\_word/byte(&(string1\_table[i])));

Serial.println(buffer);

1. **EEPROM**

In the Arduino sketch IDE window, use the Help >> Reference menu, select the Libraries, and look at the EEPROM library. To access EEPROM, you need to include the header file <EEPROM.h>. <https://www.arduino.cc/en/Reference/EEPROM>

* The function EEPROM.read(int address) returns a char value from the indicated EEPROM address.
* The function EEPROM.write(int address, char value) stores the char value at the indicated EEPROM address.
* The function EEPROM.get(int address, float data or struct object) read any data type or object from the indicated EEPROM address.
* The function EEPROM.put(int address, float data or struct object) stores any data type or object to the indicated EEPROM address.

1. From Arduino code example, open eeprom\_read sketch which includes a loop to display the existing values in the EEPROM from the beginning to the end of its memory space. What values does the program show and where do they come from?

The program initially shows a defined value with different magnitude which means that there is something stored inside the given address (around 0-3) but after that, the program shows that the number “255” which means that the address had no value stored inside therefore, cant read it.

2. Create a new sketch that stores the product information: serial number (eg. TH01234), size, manufacture date (eg. Oct 7, 2020 and company’s name (eg. SIIE) into EEPROM using EEPROM.put() function.

3. Create another new sketch that read the product information from EEPROM using EEPROM.get () function and display on the Serial monitor.

4. Show your program to TAs and submit through online both codes from #2 and # 3.

References:

<https://www.cs.umb.edu/cs341/>

<http://www.weigu.lu/tutorials/microcontroller/04_memory/index.html\>

<https://playground.arduino.cc/Learning/Memory/>