

## 1. 組員

B10715029 陳彥瑋

B10730226 曾文彥

## 2. 執行環境

OS : Ubuntu 20.04

g++ : 9.3.0

## 3. 操作方式

使用 make 執行後，直接在命令列下 ./hw2 執行，並根據以下要執行的功能在執行階段打入對應的指令：

讀取： read [blockNo] [fromPage] [toPage]

寫入： write [blockNo] [fromPage] [toPage] [pattern]

清除： erase [blockNo]

## 4. 程式碼解說

```
16  ✓ /*
17      * mode           : Operation mode. Offered: read, write, erase.
18      * blockNo        : Target block number.
19      * fromPage [optional] : Read/Write begin at which page (included).
20      * toPage  [optional] : Read/Write end at which page (included).
21      * pattern  [optional] : Write pattern.
22      *
23      */
24  void flashControl(
25      std::string mode
26  );
27
28  ✓ int main(int argc, char **argv){
29      std::string mode = "read";
30      ✓ //uint32_t blockNo = 10;
31      //uint32_t fromPage = 4;
32      //uint32_t toPage = 9;
33
34      ✓ while(std::cin >> mode){
35          flashControl(mode);
36      }
37      return 0;
38  }
```

在 main 函式中，會讀入使用者要使用的 mode，並會呼叫 flashControl 來對 flash 做操作。

```

46     if(mode == "read")           // read
47     {
48         std::cin >> blockNo >> fromPage >> toPage;
49         // Check page's range
50         // No need to check < 0 because type is unsigned
51         if(fromPage >= PAGE_LIMIT)
52         {
53             fprintf(stderr, "Variable frompage overflow!!\n");
54             return;
55         }
56         if(toPage >= PAGE_LIMIT)
57         {
58             fprintf(stderr, "Variable toPage overflow!!\n");
59             return;
60         }
61         if(fromPage > toPage)
62         {
63             // swap
64             uint32_t temp = fromPage;
65             fromPage = toPage;
66             toPage = temp;
67         }
68
69         // read pages
70         for(uint32_t page=fromPage; page<=toPage; page++)
71         {
72             printf(" Read dwBlock %d(0x%04Xh)\n", blockNo, blockNo);
73             printf(" Read dwPage %d(0x%04Xh)\n", page, page);
74             vFTL_PageRead(page, blockNo); // read 1 page
75         }
76     }

```

在 flashControl 中，假如使用者要使用讀取功能，則會先檢查使用者輸入的 page 做範圍的檢查，檢查完後則根據 page 的範圍使用迴圈一一呼叫 vFTL\_PageRead 來讀取資料。

```

77  else if(mode == "write")    // write
78  {
79      int tmp_int;
80      std::cin >> blockNo >> fromPage >> toPage >> std::hex >> tmp_int >> std::dec;
81      //std::cout << tmp_int << std::endl;
82      pattern = tmp_int;
83      // Check page's range
84      // No need to check < 0 because type is unsigned
85      if(fromPage >= PAGE_LIMIT)
86      {
87          fprintf(stderr, "Variable frompage overflow!!\n");
88          return;
89      }
90      if(toPage >= PAGE_LIMIT)
91      {
92          fprintf(stderr, "Variable toPage overflow!!\n");
93          return;
94      }
95      if(fromPage > toPage)
96      {
97          // swap
98          uint32_t temp = fromPage;
99          fromPage = toPage;
100          toPage = temp;
101      }
102
103      //memset((byte *)(FLASH_MEM_BASE), bPATTERN, 128);
104      //pSRamBuffer[0] = 0;
105
106      // write pages
107      for(uint32_t page=fromPage; page<=toPage; page++)
108      {
109          printf(" dwPage %d(0x%04Xh)\n", page, page);
110          //pSRamBuffer[0] = page;
111          vFTL_PageProgram(page, blockNo, pattern); // write 1 page
112      }
113  }

```

而 write 的部分跟 read 很類似，一樣是先做範圍的檢查後使用迴圈來一一的使用 vFTL\_PageProgram 對 page 做寫入。

不過由於 flash 的限制，所以一旦資料被寫過之後就必需要先做清除才能再做寫入，否則資料是寫不進去的。

```

114  else if(mode == "erase")    // erase
115  {
116      std::cin >> blockNo;
117      bFTL_BlockErase(blockNo);
118      fprintf(stdout, "Erase block %d complete.\n", blockNo);
119  }

```

相較於前面的部分，erase 的部分就比較簡單。這個部分是直接讀入要清除的 block number 之後使用 bFTL\_BlockErase 來對整個 block 做清除。