

RFID_LAB2_GROUP2_書面說明

組員：

512411003 洪健航

512411010 曾偉桓

512411014 劉俊麟

512411016 鄭竣元

512411024 劉瑋湏

512411029 李元銘

程式作法與重點描述：

1. Data struct

因為會員的四個屬性存放的位置都不相同，所以有先設定一個 struct 來存放四個屬性的 sector 和 block 和 keyAB(第 12 行-15 行)

縮寫說明：

No=會員編號、Nm=Name=會員名稱、Ad=ApplyDate=申請日期、Ct=Credit=會員點數

st=Sector、bk=block、ab=keyAB

```
12  AddressStruct adrNo = new AddressStruct(0, 1, "A");
13  AddressStruct adrNm = new AddressStruct(0, 2, "A");
14  AddressStruct adrAd = new AddressStruct(1, 0, "A");
15  AddressStruct adrCt = new AddressStruct(1, 1, "A");
```

```
233  public struct AddressStruct
234  {
235      public UInt16 st;
236      public UInt16 bk;
237      public String ab;
238      4 個參考
239      public AddressStruct(UInt16 Sector, UInt16 Block, String KeyAB)
240      {
241          this.st = Sector;
242          this.bk = Block;
243          this.ab = KeyAB;
244      }
```

2. write_rfid_value () & build_write_cmd()

這兩個 function 主要負責組出 Write Command 和寫入資料到 Card，與 read_rfid_value() 的主要差異就是 LEN 和 CMD 這兩個欄位差異(見下二圖，左圖是 read、右圖是 write)。

| | |
|---|---|
| <pre>// build up read command byte[] WriteBuffer = new byte[] { 0x2, // STX 0xA, // LEN 0x15, // CMD (byte)((keyAB == "A")? 0x60: 0x61), key_bytes[0], // KEY most left key_bytes[1], // KEY key_bytes[2], // KEY key_bytes[3], // KEY key_bytes[4], // KEY key_bytes[5], // KEY most right (byte)sector, // Sector (byte)block // Block };</pre> | <pre>// build up write command byte[] WriteBuffer = new byte[] { 0x2, // STX 0x1A, // LEN 0x16, // CMD (byte)((keyAB == "A")? 0x60: 0x61), key_bytes[0], // KEY most left key_bytes[2], // KEY key_bytes[1], // KEY key_bytes[3], // KEY key_bytes[4], // KEY key_bytes[5], // KEY most right (byte)sector, // Sector (byte)block // Block };</pre> |
|---|---|

另外，為了彈性使用 write_rfid_value ()，我多了一個參數 byte[] val，方便未來可以寫入任何資料【卡片製作】、【清空卡片】、【儲值】、【消費】都會用到，並把 build_write_cmd 返回的 byte[]，Concat 起來後(第 95、96 行)，再發送給 Rfid card。

```
90 unsafe public String write_rfid_value(UInt16 sector, UInt16 block, String keyAB, String key, byte[] val)
91 {
92     UInt32 dwResult, Index;
93     UInt32 uiLength, uiRead, uiResult, uiWritten;
94     byte[] ReadBuffer = new byte[0x40];
95     byte[] WriteBuffer = build_write_cmd(sector, block, keyAB, key);
96     WriteBuffer = WriteBuffer.Concat((val)).ToArray();
97     byte[] sResponse = null;
98     sResponse = new byte[21];
```

3. Create_Card() & Clear_Card()

這兩個 function 分別對應按鍵：【卡片製作】、【清空卡片】。

這兩個 function 呼叫 write_rfid_value (), 並把會員的四個屬性都先轉換成 string ,

透過 ConvertStringToByteArray () (第 187 行) 把 string 轉換成 byte[] , 然後再透過 write_rfid_value ()把資料寫入到 CARD .

另外四個屬性的寫入並沒有像 SQL 一樣的 transaction , 所以沒有判斷是否有完整寫入, 例如 NO 有寫入, 但是 NAME 寫入失敗需要 ROLL BACK.

ConvertByteArrayToString() (第 200 行)和 ConvertStringToByteArray () (第 187 行) 是一對, 一個用來 string 轉換成 byte[], 一個用來把 byte[]轉成 string

```
22 public void Create_Card(string no, string name, DateTime applydate, int credit)
23 {
24     var sNo = write_rfid_value(adrNo.st, adrNo.bk, adrNo.ab, loadKey, ConvertStringToByteArray(no));
25     var sNm = write_rfid_value(adrNm.st, adrNm.bk, adrNm.ab, loadKey, ConvertStringToByteArray(name));
26     var sAd = write_rfid_value(adrAd.st, adrAd.bk, adrAd.ab, loadKey, ConvertStringToByteArray(applydate.ToShortDateString()));
27     var sCt = write_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey, ConvertStringToByteArray(credit.ToString()));
28 }
29 public void Clear_Card()
30 {
31     var sNo = write_rfid_value(adrNo.st, adrNo.bk, adrNo.ab, loadKey, new byte[16]);
32     var sNm = write_rfid_value(adrNm.st, adrNm.bk, adrNm.ab, loadKey, new byte[16]);
33     var sAd = write_rfid_value(adrAd.st, adrAd.bk, adrAd.ab, loadKey, new byte[16]);
34     var sCt = write_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey, new byte[16]);
35 }
```

```
187 public byte[] ConvertStringToByteArray(string text)
188 {
189     // 使用 UTF8 編碼將字符串轉換成 byte[]
190     byte[] byteArray = Encoding.UTF8.GetBytes(text);
191     // 如果長度小於16, 則補零
192     if (byteArray.Length < 16)
193     {
194         byte[] paddedArray = new byte[16];
195         Array.Copy(byteArray, paddedArray, byteArray.Length);
196         return paddedArray;
197     }
198     return byteArray;
199 }
```

```
200 public string ConvertByteArrayToString(byte[] byteArray)
201 {
202     // 使用 UTF8 解碼將 byte[] 轉換成字符串
203     string resultString = Encoding.UTF8.GetString(byteArray);
204     // 去除尾部的零
205     resultString = resultString.TrimEnd('\0');
206     return resultString;
207 }
```

4. Read_Card() & read_rfid_value()

Read_Card() function 主要是讀取 Card 的四個資料，並轉換成相對應的資料型態後回傳，會用在【讀取卡片】與【儲值】的加分題中。
read_rfid_value()則是 Lab1 作業裡面的，就不再重述。

ConvertByteArrayToString() (第 200 行)和 ConvertStringToByteArray() (第 187 行) 是一對，一個用來 string 轉換成 byte[]，一個用來把 byte[] 轉成 string

```
36 public (string no, string name, DateTime applydate, int credit) Read_Card()
37 {
38     var byteNo = ConvertHexStringToByteArray(read_rfid_value(adrNo.st, adrNo.bk, adrNo.ab, loadKey));
39     var byteNm = ConvertHexStringToByteArray(read_rfid_value(adrNm.st, adrNm.bk, adrNm.ab, loadKey));
40     var byteAd = ConvertHexStringToByteArray(read_rfid_value(adrAd.st, adrAd.bk, adrAd.ab, loadKey));
41     var byteCt = ConvertHexStringToByteArray(read_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey));
42     return (ConvertByteArrayToString(byteNo), ConvertByteArrayToString(byteNm), Convert.ToDateTime(ConvertByteArrayToString(byteAd
43 }
```

```
187 public byte[] ConvertStringToByteArray(string text)
188 {
189     // 使用 UTF8 編碼將字符串轉換成 byte[]
190     byte[] byteArray = Encoding.UTF8.GetBytes(text);
191     // 如果長度小於16，則補零
192     if (byteArray.Length < 16)
193     {
194         byte[] paddedArray = new byte[16];
195         Array.Copy(byteArray, paddedArray, byteArray.Length);
196         return paddedArray;
197     }
198     return byteArray;
199 }
```

```
200 public string ConvertByteArrayToString(byte[] byteArray)
201 {
202     // 使用 UTF8 解碼將 byte[] 轉換成字符串
203     string resultString = Encoding.UTF8.GetString(byteArray);
204     // 去除尾部的零
205     resultString = resultString.TrimEnd('\0');
206     return resultString;
207 }
```

5. Charge_Card()

這個主要是用在【儲值】

先利用 `read_rfid_data()` 取出卡片原有的點數為多少，然後再加上 `credit_plus` 這個數字之後
再透過 `write_rfid_value()` 來進行寫入點數。

```
44 public (int credit_after, int credit_plus, string msg) Charge_Card(int credit_plus)
45 {
46     var byteCt = ConvertHexStringToByteArray(read_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey));
47     int credit_before = int.Parse(ConvertByteArrayToString(byteCt));
48     int credit_after = credit_before + credit_plus;
49     var sCt = write_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey, ConvertStringToByteArray(credit_after.ToString()));
50     string msg = @"儲值:" + credit_plus + "; 可用餘額:" + credit_after;
51     return (credit_after, credit_plus, msg);
52 }
```

6. Consume_Card () 【消費點數】加分題說明

這邊利用了 while() 迴圈，先檢查消費點數是否高於卡片中的點數，如果沒有則進行一次加值，如果還是不夠，會再進行第二次加值，一直到卡片的點數高於消費點數。當高於消費點數後，才會真正進行【消費點數】的行為。

並且如果有自動加值，則會把相關訊息秀給使用者看。

```
53 public (int credit_after, int credit_plus, string msg) Consume_Card(int credit_plus)
54 {
55     string msg = "";
56     #region//加分題，自動儲值
57     int autoChargeCredit = 2000;//自動加值金額
58     int autoChargeRounds = 0;//自動加值次數
59     while (Read_Card().credit - credit_plus <= 0)
60     {
61         autoChargeRounds++;
62         var autoChargeResult = Charge_Card(autoChargeCredit);
63     }
64     if (autoChargeRounds > 0)
65     {
66         msg += "由於您的紅利點數不足，系統幫您自動加值!" + Environment.NewLine +
67             "自動加值:" + autoChargeCredit +
68             " 次數:" + autoChargeRounds + " (共" + (autoChargeCredit * autoChargeRounds) + ")" + Environment.NewLine;
69     }
70     #endregion
71     var byteCt = ConvertHexStringToByteArray(read_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey));
72     int credit_before = int.Parse(ConvertByteArrayToString(byteCt));
73     int credit_after = credit_before - credit_plus;
74     var sCt = write_rfid_value(adrCt.st, adrCt.bk, adrCt.ab, loadKey, ConvertStringToByteArray(credit_after.ToString()));
75     msg += @"消費:" + credit_plus + "; 可用餘額:" + credit_after;
76     return (credit_after, credit_plus, msg);
77 }
```

遇到困難點, 與解決方法

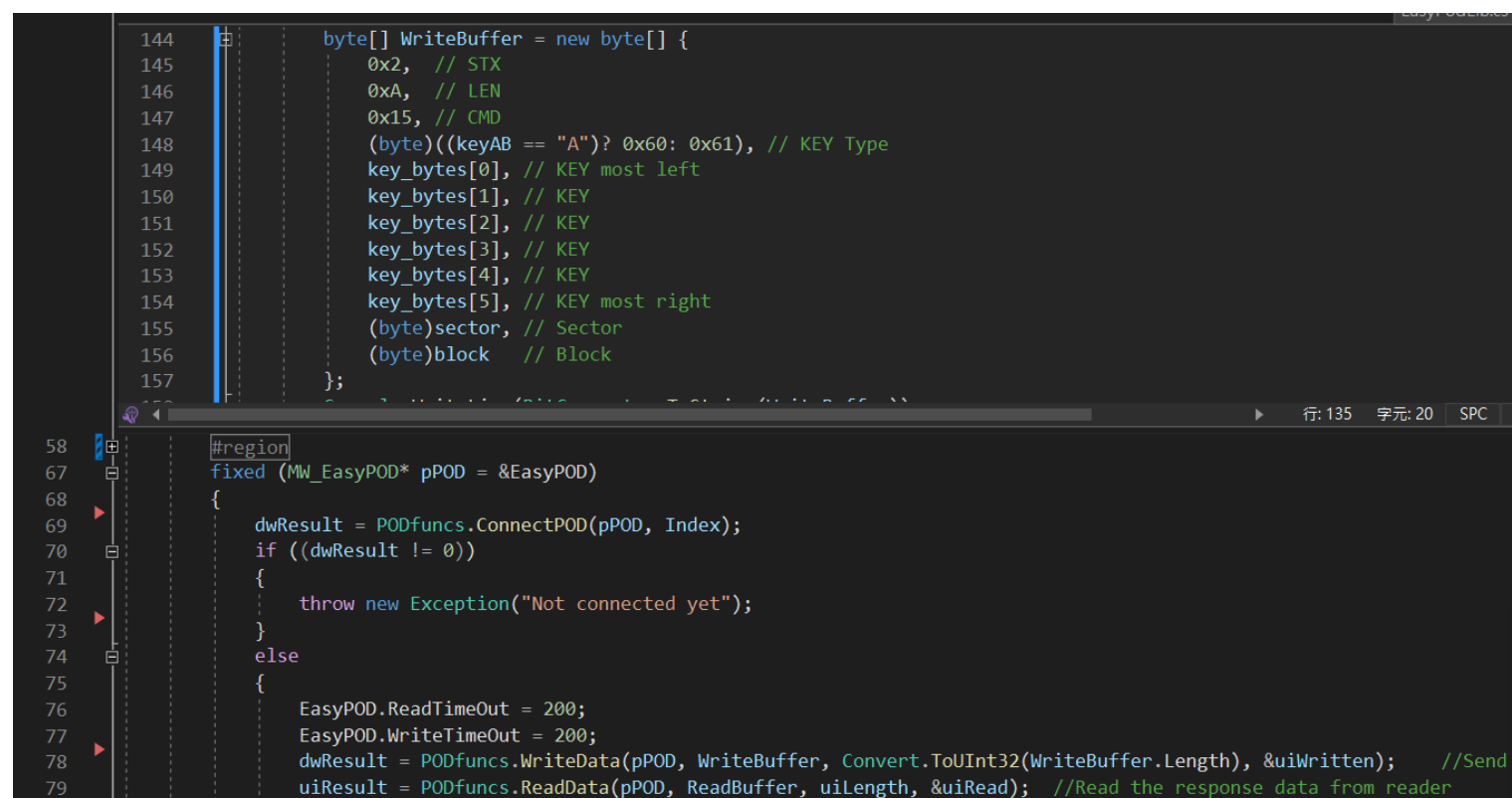
1. 我人在美國出差，機器在台灣組員瑋湏身上，且有時差問題：

因為 LAB1 作業時是由瑋湏為主要開發，讀取器放在瑋湏家中，LAB2 作業時，因為程式邏輯比較困難，所以由我來進行主要開發．
首先碰到的問題就是硬體問題，如何在美國使用台灣的機器，經過初次測試發現卡片可以一直放在讀卡器上，不需要人為移出／移入，
因此瑋湏早上出門把卡片放到機器上，美國晚上時間請瑋湏開啟遠端連線，讓我可以連線連到瑋湏電腦進行開發並測試卡片．

2. 不知道如何寫 WRITE RFID 程式碼：

因為一開始只有 LAB1 的作業原始碼，而 LAB1 作業只有讀取 CARD，沒有 WRITE 的實作程式碼，所以花了很多時間找 GOOGLE，PDF 說明檔也看不太懂，這情況一直拖了兩個多禮拜都沒有進展．

後來在操作 RD200/300 程式的時候，發現下方有個 TX 與 RX，再與 LAB1 作業比較的時候，發現 TX 的位元組就是 C#程式中需要組成的 byte[] (144~157 行)，而 RX 的位元組就是 PODfuncs.ReadData 返回的 ReadBuffer(第 79 行) ，



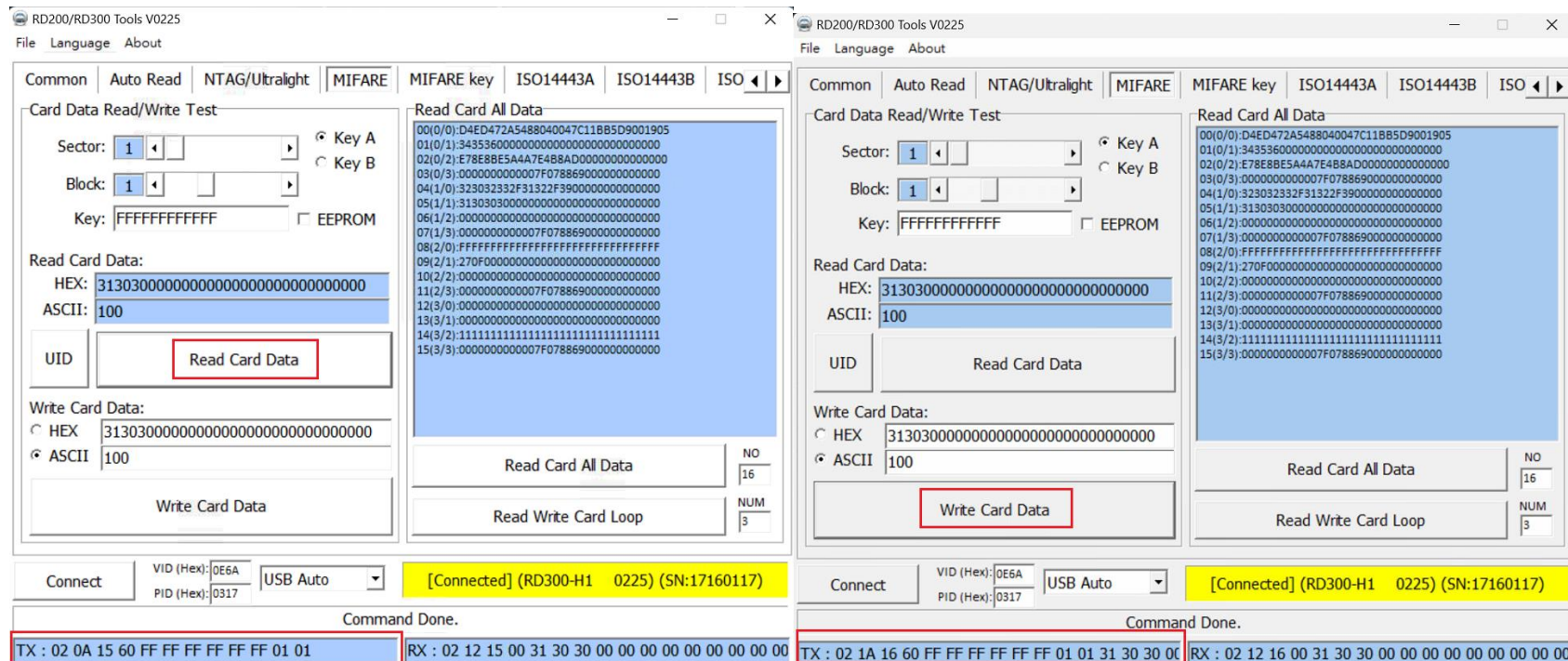
```
144     byte[] WriteBuffer = new byte[] {
145         0x2, // STX
146         0xA, // LEN
147         0x15, // CMD
148         (byte)((keyAB == "A")? 0x60: 0x61), // KEY Type
149         key_bytes[0], // KEY most left
150         key_bytes[1], // KEY
151         key_bytes[2], // KEY
152         key_bytes[3], // KEY
153         key_bytes[4], // KEY
154         key_bytes[5], // KEY most right
155         (byte)sector, // Sector
156         (byte)block // Block
157     };

58     #region
67     fixed (MW_EasyPOD* pPOD = &EasyPOD)
68     {
69         dwResult = PODfuncs.ConnectPOD(pPOD, Index);
70         if ((dwResult != 0))
71         {
72             throw new Exception("Not connected yet");
73         }
74         else
75         {
76             EasyPOD.ReadTimeOut = 200;
77             EasyPOD.WriteTimeOut = 200;
78             dwResult = PODfuncs.WriteData(pPOD, WriteBuffer, Convert.ToUInt32(WriteBuffer.Length), &uiWritten); //Send
79             uiResult = PODfuncs.ReadData(pPOD, ReadBuffer, uiLength, &uiRead); //Read the response data from reader
```


接著再操作 RD200，觀察 read card data 和 write card data 這兩者 TX 的差異，才順利寫出 write_rfid_value ()和 build_write_cmd()

read data TX : 02 0A 15 60 FF FF FF FF FF FF 01 01

write data TX : 02 1A 16 60 FF FF FF FF FF FF 01 01 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00



3. 把字串轉換成 hex：這部分先前經驗比較少做轉換，所以是直接問 ChatGPT，修改多次之後才符合此次作業使用。

```
187 public byte[] ConvertStringToByteArray(string text)
188 {
189     // 使用 UTF8 編碼將字串轉換成 byte[]
190     byte[] byteArray = Encoding.UTF8.GetBytes(text);
191     // 如果長度小於16，則補零
192     if (byteArray.Length < 16)
193     {
194         byte[] paddedArray = new byte[16];
195         Array.Copy(byteArray, paddedArray, byteArray.Length);
196         return paddedArray;
197     }
198     return byteArray;
199 }

200 public string ConvertByteArrayToString(byte[] byteArray)
201 {
202     // 使用 UTF8 解碼將 byte[] 轉換成字串
203     string resultString = Encoding.UTF8.GetString(byteArray);
204     // 去除尾部的零
205     resultString = resultString.TrimEnd('\0');
206     return resultString;
207 }
```

4. 程式架構的問題：一開始寫了很多類似 write_rfid_data()的 function，分別對應不同的【卡片製作】、【清空卡片】、【儲值】、【消費】，後來把參數抽離出來，才可以一個 function 重複使用。

5. 寫錯到保留區塊／控制區：一開始沒認真讀 PDF 檔，所以我把資料寫入 0/0 0/3 1/3，導致卡片壞掉，呈現 No Data，無法再重新寫入資料。後來經由組員瑋湏提醒才發現這幾個是保留區，因此跳過這幾個保留區就正常了。

心得感想：

這次的作業遇到了不少挑戰，但透過團隊合作和解決問題的努力，成功完成了 RFID 卡片的讀寫功能。以下是一些心得感想：

- **團隊合作與跨時區協作：** 在團隊合作中，跨時區的協作確實是一大挑戰。然而，透過合理分工和有效的溝通，我們成功地克服了時差問題，確保了作業的順利進行。使用遠端連線的方式，讓卡片可以在台灣的讀卡器上被遠程讀取。這種解決方案確實需要團隊成員之間的彈性和密切的合作。
- **學習新技術與解決問題：** 在開發過程中，遇到了寫 RFID 卡片的挑戰。由於一開始缺少相應的範例程式碼，花了一些時間在研究和試驗中找到解決方案。透過觀察硬體操作和不斷的實驗，最終成功地撰寫了相應的寫卡程式碼。
- **字串轉換與程式優化：** 處理字串轉換的挑戰也是一個學習過程。透過不斷的嘗試和諮詢，我們成功地將字串轉換成 hex 格式，以符合作業的需求。此外，對程式的架構進行優化，使得一個通用的 function 可以應對不同的操作，這也是提升程式可維護性的好方法。
- **學會從錯誤中學習：** 在作業的過程中，我們也經歷了一些錯誤。例如，一開始未仔細閱讀 PDF 文件，導致在錯誤的保留區寫入資料，最終導致卡片無法正常運作。然而，透過及時的提醒和學習，我們學會了從錯誤中吸取教訓，並成功地修正了這些問題。

這次的作業讓我深刻體會到了在團隊中解決問題和克服困難的重要性。從技術層面到團隊協作，都是一次難得的學習經驗。希望未來能夠繼續保持這種合作的精神，面對各種挑戰，不斷學習和進步。