RFM-001

使用手冊

V1.00

2007/01/31

聯暘電子股份有限公司

Sunion Electronics Corp.

241 台北縣三重市興德路 123-7 號 11 樓

TEL：+886-02-85121456　FAX：+886-2-85121457

www.sunion.com.tw

目　錄

合法版權聲明

1. 模組簡介

1.1 特性簡介

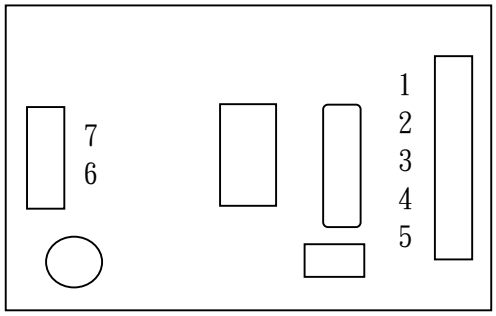
TIRIS (**T**exas **I**nstruments **R**egistration and **I**dentification **S**ystem) 是美國德州儀器公司所開發出之無線感應辨識系統其系統架構內包含有感應式讀卡機(Reader)、感應天線 (Antenna) 和感應器 (Transponder) 等設備。
其工作原理是：先由讀卡機透過天線發出 134.2 kHz 的電磁波對感應器 (Transponder) 充電，此時若有感應器在此磁場範圍內，就會利用磁能轉換的原理，將接收到的電磁波轉換為其所需之電能，並將感應器內所存有的密碼 (ID Code) 以 FM 的調變方式傳送回去，此時讀卡機停止發射電磁波，並經由天線接收感應器傳回的密碼，並進行解碼的動作，如此便完成讀卡的動作，故使用 TIRIS 感應辨識系統有以下優點：

1. 感應器 (Transponder) 本身不需使用電池，且不受灰塵，水氣所影響，故障率低。
2. 使用 FM 調變方式傳送數位信號，不易受雜訊所干擾。
3. 感應器充電程序與資料回傳程序分別進行，不易受環境電場所干擾。(此為 TI 專利技術)

TIRIS 在汽機車防盜器、門禁系統、工業自動化控制等領域中受到廣泛的應用，而 **RFM-001** 便是聯暘為 **TIRIS** 系統所開發出的 **RF 模組**，它具備了以下幾點特色：

1. 可直接驅動天線，不需外加驅動元件。
2. 具備天線短路保護。
3. 具備自動進入睡眠模式，耗電僅 0.2mA。
4. 小體積之 RF Module。

1.2 接腳說明

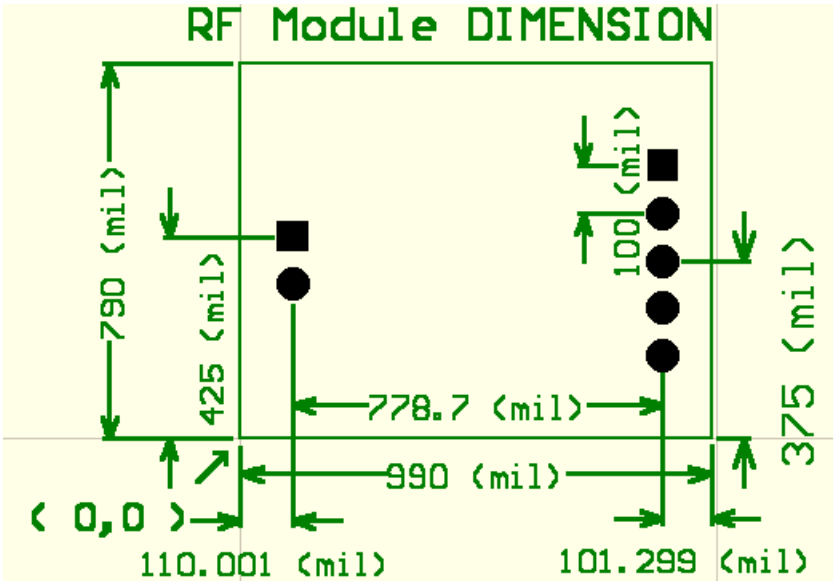


RFM-001 (零件面)

腳位	名稱	功能
1	TXCT-	控制輸入 (連接到微控制器 (MCU)，為 LOW 動作)
2	SCIO	數位資料輸出 (連接到微控制器 (MCU))
3	GND	邏輯電源接地
4	GND	邏輯電源接地
5	VDD	邏輯電源輸入
6	ANT2	天線輸出 2
7	ANT1	天線輸出 1

1.3 尺寸規格

長 25mm　　寬 20mm　　高 11mm (含排針座)



1.4 電氣規格

Temperature：Tamb = -40℃ to +85℃
Supply Voltage：Vdd = 4.5V to 5.5V

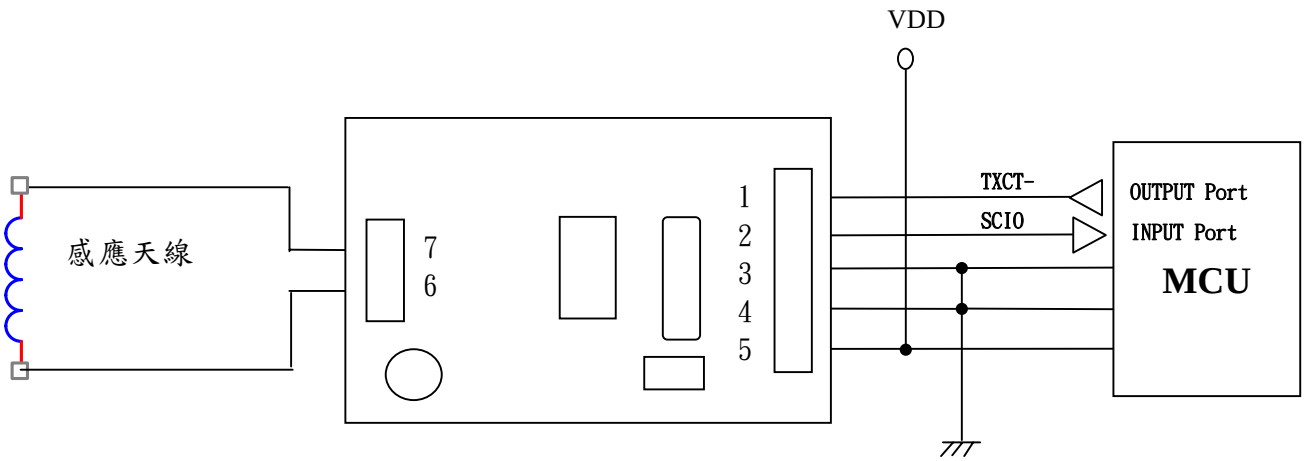
PARAMETERS AND CONDITIONS	SYMBOL	NOTE	MIN.	TYP.	MAX.	UNIT
Power Supply						
(VDD) Supply Voltage	Vdd		4.5	5	5.5	V
Sum of Supply Current in Charge Phase, without antenna load	Idd			8	20	mA
Sum of Supply Current in Sleep Mode, without I/O currents	Isleep			0.015	0.2	mA
Logic Outputs (SCIO)						
Output High Voltage	Voh		0.8Vdd			V
Output Current at Voh	Ioh		-1			mA
Output Low Voltage	Vol				0.2Vdd	V
Output Current at Vol	Iol				1	mA

NAME	SYMBOL	COMMENT	MIN.	MAX.	UNIT
Full-Bridge Outputs (ANT1，ANT2)	Protection against voltage peaks and short-circuits (in accordance to application)				
Output Voltage	Vant		-0.3	Vdd+0.3	V
Output Peak Current	Iant		-1.1	1.1	A

1.5 天線規格

天線電感值 = 430uH ～ 460uH
標準天線 14 * 10.5 cm (電感值 = 430uH)

2. 電路範例



3. 讀取控制流程

3.1 讀取 Read Only Type

(1) 將 TXCT- 設為 Low，Delay 50ms 後，再將 TXCT- 恢復成 High(參考 Figure1)
(2) 此時約過 3ms，SCIO 開始輸出資料，第一個 Byte 即為 START Byte，總共輸出 14Bytes 資料 (參考 Figure2)。
(3) 每個 Byte 的格式如 Figure2，由 10 bits 組成，第一個 bit 是 START bit 固定為 HI，最後一個 bit 是 Stop bit 固定為 LOW，第 2～9 bit 為資料 (最先收到的 bit 為 LSB)，資料需要反相處理 (LOW=1、HI=0)。

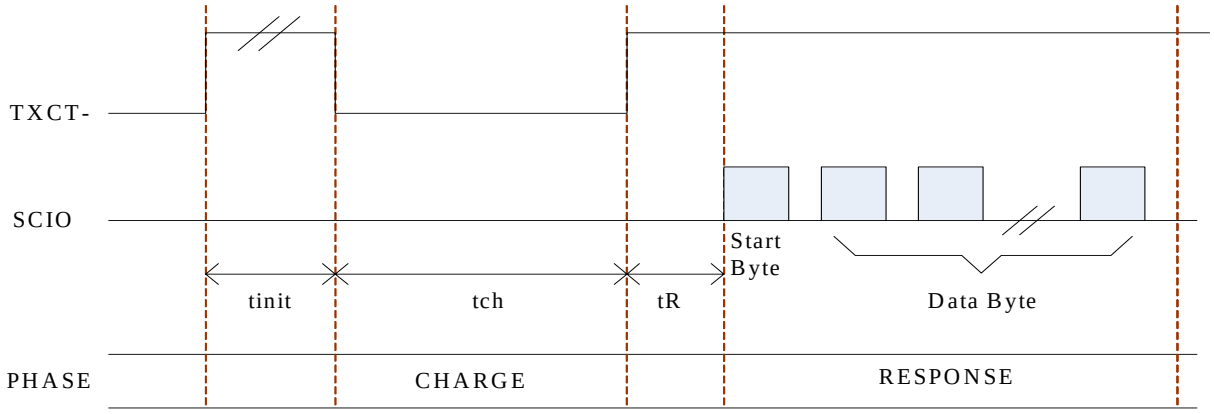


Figure 1

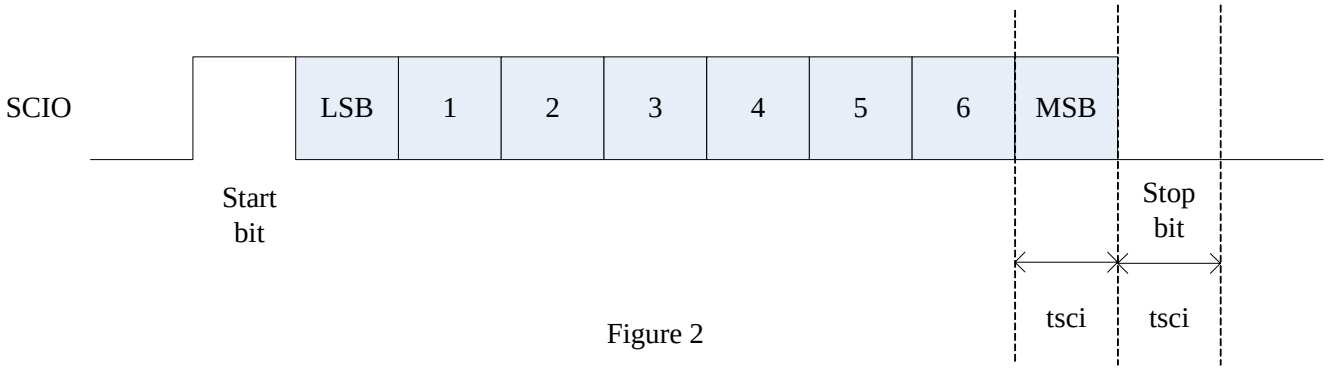


Figure 2

PARAMETERS AND CONDITIONS	SYMBOL	MIN.	TYP.	MAX.	UNIT
Time for TXCT high to initialize a new Transmission, from start of the oscillator after power-on or waking up until reaching the idle mode	t init min	2	2.05	2.2	ms
Charge phase duration	tch		50		ms
Delay between end of charge or end of program and start of transponder data Tx on SCIO	tR		3		ms
NRZ bit duration on SCIO in asynchronous mode	tsci	63	64	65	us

註：當 TXCT 為 High 時，當時間超過 Max Time 2.2ms 時，將會自動進入睡眠模式。

附表 1

（4）Read Only type 回應格式：包含 Start byte 所得到的 **14** 個 **Bytes**，即為我們所要的資料，分析如下：

第 1 個 byte	Start byte - R/O=7EH
第 2～9 個 byte	ID 號碼(8 個 bytes)
第 10、11 個 byte	ID 號碼的 CRC 檢查碼
第 12 個 byte	Stop byte – R/O=7EH
第 13、14 個 byte	Read Only 固定為 0000H

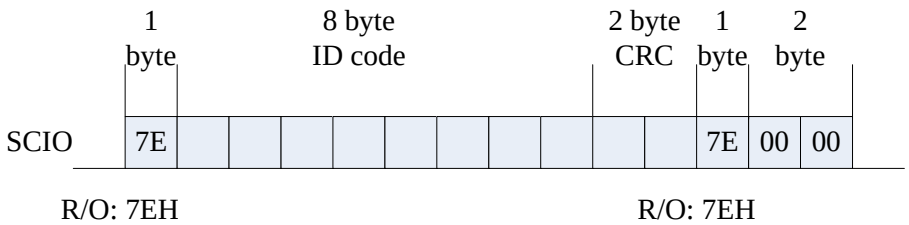


Figure 3

【注意】Tiris transponder的回應之中，最後一個byte的有效bit數為 7 bits。在此是指所有Command的回應，包含讀、寫、鎖。

3.2 讀取 Read/Write Type

(1) 時序同 Read Only，請參考 Figure1, Figure2。
(2) Read/Write Type 回應格式，分析如下：

第 1 個 byte	Start byte - R/W=FEH
第 2～9 個 byte	ID 號碼(8 個 bytes)
第 10、11 個 byte	ID 號碼的 CRC 檢查碼
第 12 個 byte	Stop byte – R/W=FEH
第 13、14 個 byte	Read Write 格式為第 13 個 byte 等於第 2 個 byte，第 14 個 byte 等於第 3 個 byte

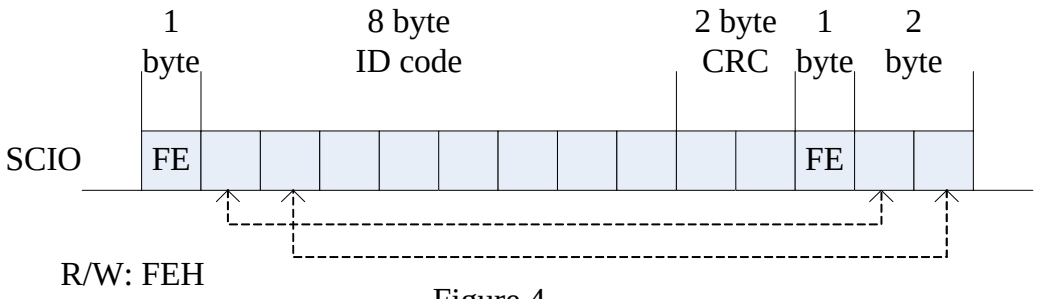


Figure 4

3.3 讀取 Multi-Page Type

(1) 時序和 Read Only 略有不同，在 Charge 50ms 後要先送 Read Address，才做讀取動作，請參考 Figure5。

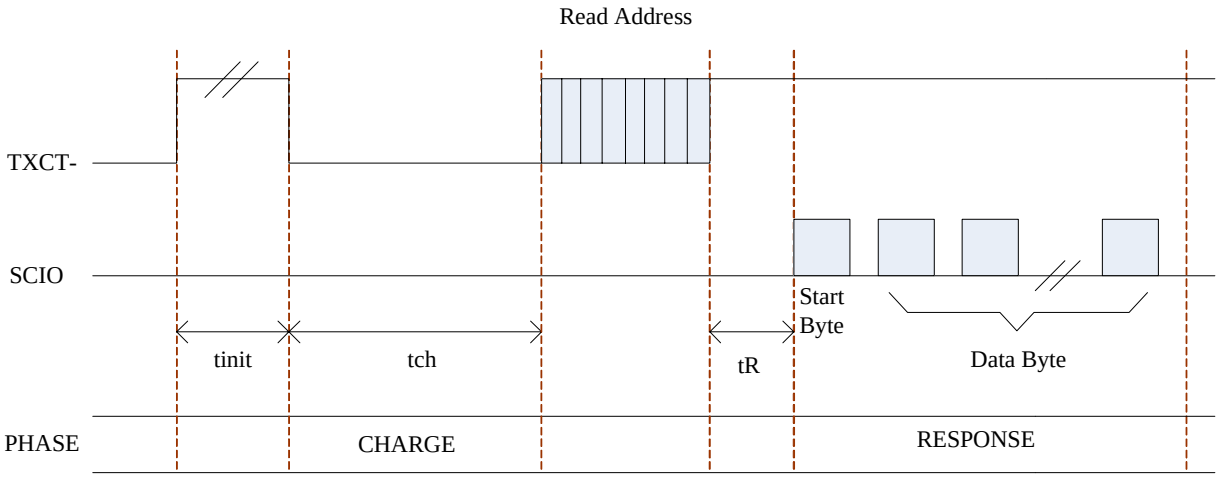


Figure 5

(2) Read Address 為 1 byte，用來指定頁數與命令，格式如下：

PAGE						Command	
7	6	5	4	3	2	1	0

MSB LSB

[PAGE]

PAGE1: 000001
PAGE2: 000010
: :
PAGE17: 010001

[Command]

Read: 00
Write: 01
Lock: 10

(3) Reader 對 Transponder 寫資料時的位元編碼：

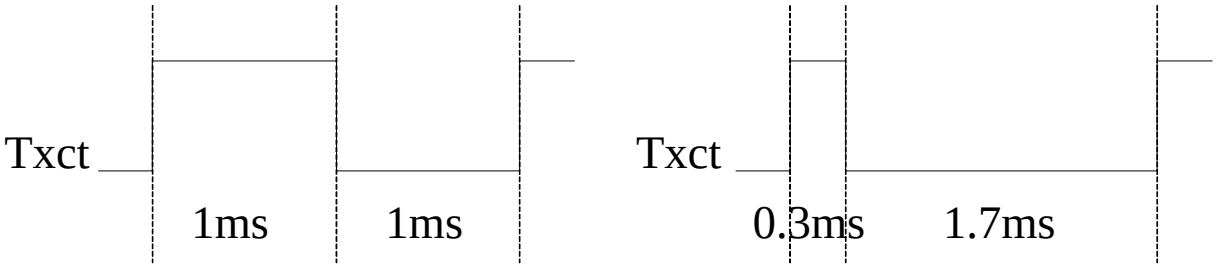


Figure6

(4) Multi-Page 回應的格式如下：

第 1 個 byte	Start byte – Multi-Page = FEH
第 2～9 個 byte	Page 資料 (8 個 bytes)
第 10、11 個 byte	Page 資料的 CRC 檢查碼
第 12 個 byte	Read Address
第 13、14 個 byte	CRC-1，此二個 byte 為第 2～12 個 byte 的 CRC 檢查碼

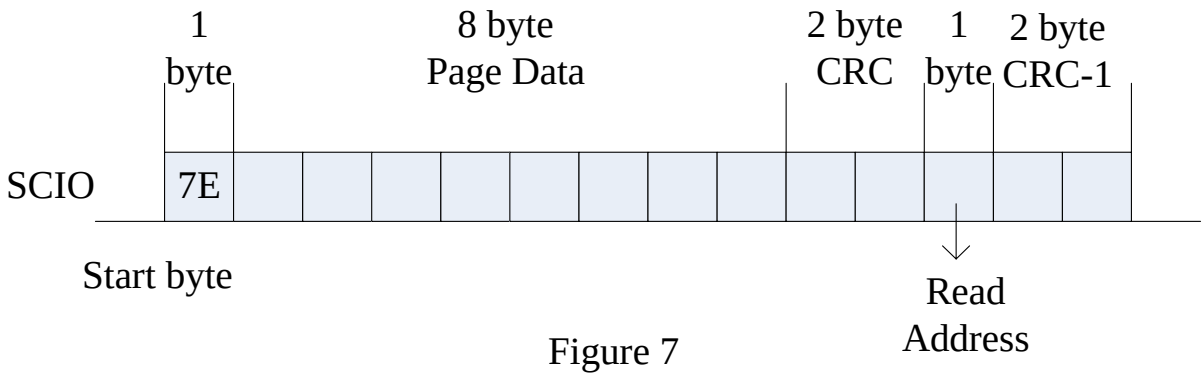


Figure 7

(5) 回應資料中的 Read Address 用來表示頁數與狀態，格式如下：

PAGE						Status	
7	6	5	4	3	2	1	0

MSB LSB

[PAGE]

PAGE1: 000001
PAGE2: 000010
:
:
PAGE17: 010001

[Status]

00: Read Unlocked Page
01: Programming Done
10: Read Locked Page

4. 寫入控制流程

4.1 寫入 Read/Write Type

(1) 寫入時序請參閱 Figure 8，共送出 14 bytes，格式如下：

第 1、2 個 byte	固定為 BBH, EBH
第 3～10 個 byte	要寫入的 ID 號碼(8 bytes)
第 11、12 個 byte	ID 號碼的 CRC 檢查碼
第 13、14 個 byte	固定為 00H, 03H

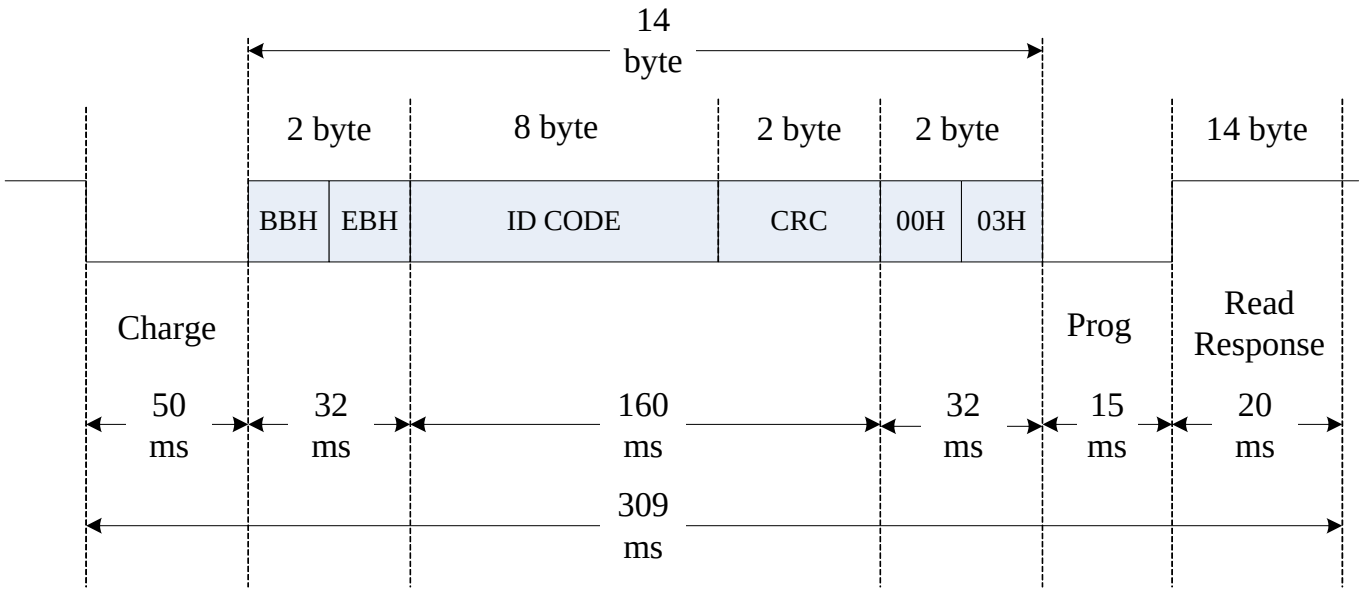


Figure 8

(2) 寫入資料的編碼格式請參閱 Figure 6，為 LSB 先送。
(3) 回應資料格式與 Figure 4 相同。

4.2 寫入 Multi-Page Type

(1) 寫入時序請參閱 Figure 9，共送出 13 bytes，格式如下：

第 1 個 byte	Write Address
第 2～9 個 byte	要寫入的 ID 號碼(8 bytes)
第 10、11 個 byte	ID 號碼的 CRC 檢查碼
第 12、13 個 byte	CRC-1 為前 11 碼運算出來的 CRC 檢查碼

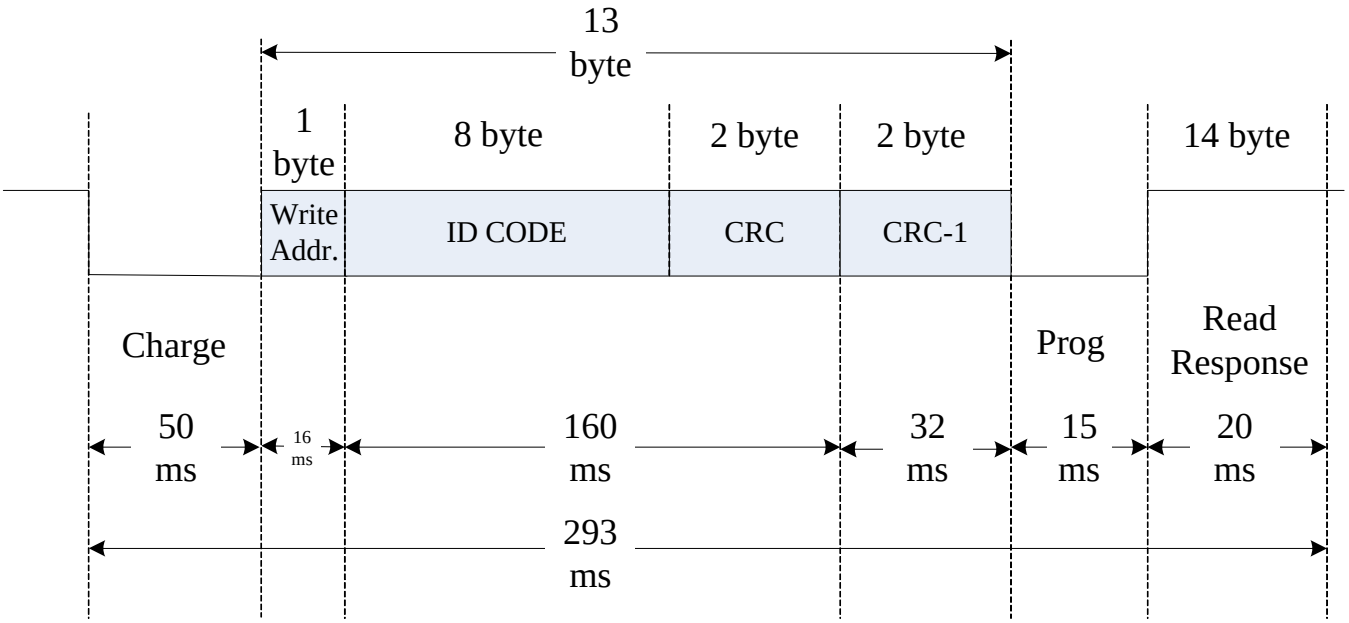


Figure 9

(2) 寫入資料的編碼格式請參閱 Figure 6，為 LSB 先送。
(3) Write Address 為 1 byte，用來指定頁數與命令，格式如下：

PAGE						Command	
7	6	5	4	3	2	1	0

MSB LSB

[PAGE]

PAGE1: 000001
PAGE2: 000010
:
:
PAGE17: 010001

[Command]

Read: 00
Write: **01**
Lock: 10

(4) 回應資料格式與 Figure 7 相同。

5. 鎖定控制流程

5.1 鎖定 Multi-Page Type

(1) 寫入時序請參閱 Figure 10，共送出 3 bytes，格式如下：

第 1 個 byte	Write Address
第 2、3 個 byte	第 1 個 byte 運算的 CRC 檢查碼

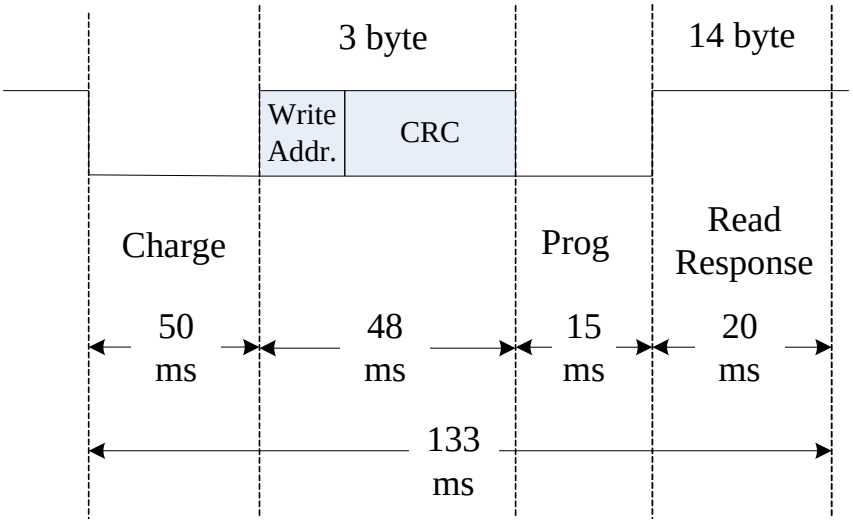


Figure 10

(2) 寫入資料的編碼格式請參閱 Figure 6，為 LSB 先送。

(3) Write Address 為 1 byte，用來指定頁數與命令，格式如下：

PAGE						Command	
7	6	5	4	3	2	1	0

MSB LSB

[PAGE]

PAGE1: 000001
PAGE2: 000010
:
:
PAGE17: 010001

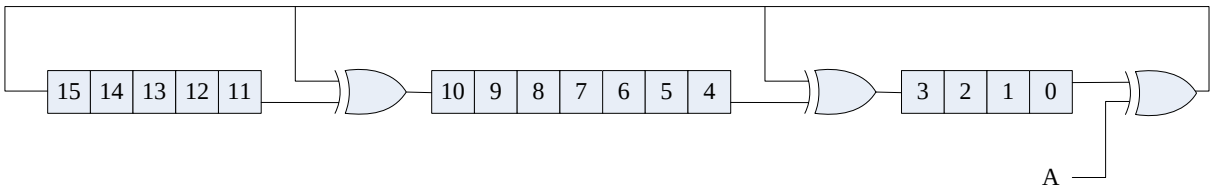
[Command]

Read: 00
Write: 01
Lock: **10**

(4) 回應資料格式與 Figure 7 相同。

6. CRC 演算法

6.1 CRC 的運算方法



(1) 上圖中 Bit15 – Bit0 為 CRC 16 位元的值。A 表示要計算 CRC 的資料。
(2) 先設定 CRC 的初始值，為 0000H。
(3) CRC 全部右旋一位元，右方移出的 LSB 移至 MSB，將新的 MSB 與 A 的最低位元做 XOR 計算。
(4) 若(3)的計算結果為 1，則 CRC 的 Bit15、Bit10 與 Bit3 反相。若此值為 0，則直接跳至(5)。
(5) 資料 A 亦右移一位元。
(6) 上述動作重覆至 A 資料的每一位元皆運算過為止。
(7) 計算完畢的 CRC 值即為 CRC 檢查碼。

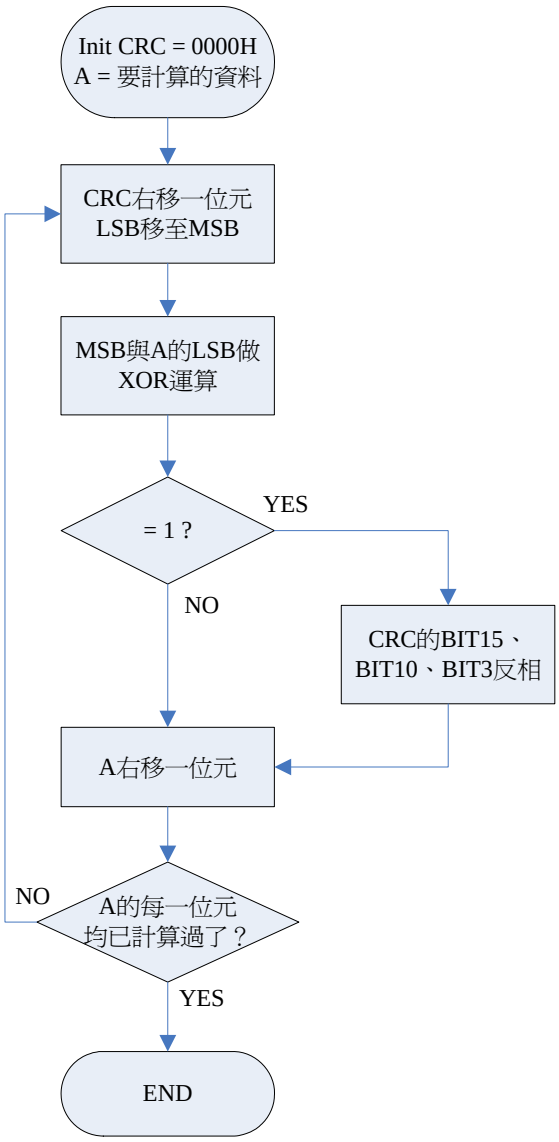
6.2 CRC 的檢查

D3	D2	D1	D0	CRC_H	CRC_L

(1) 以上圖為例，假設資料為 4 Bytes，D3 ~ D0，將此 4 Byte 計算出來的 CRC 值為 CRC_H, CRC_L。
(2) 要檢查 CRC_H, CRC_L 是否正確，則將 D3 ~ D0, CRC_H, CRC_L 當做一 6 Bytes 的資料，並對此做 CRC 運算，運算的結果，以 CRC1_H, CRC1_L 稱之。
(3) 若 CRC_H, CRC_L 是正確的，則 CRC1_H, CRC1_L 必定為 0000H。

D3	D2	D1	D0	CRC_H	CRC_L	CRC1_H	CRC1_L

6.3 CRC 流程圖


Init CRC = 0000H
A = 要計算的資料
CRC右移一位元
LSB移至MSB
MSB與A的LSB做
XOR運算
= 1 ?
YES
NO
CRC的BIT15、
BIT10、BIT3反相
A右移一位元
NO
A的每一位元
均已計算過了？
YES
END

7. 軟體範例

此段程式碼僅為程式片段，以供韌體開發人員參考之用。使用者仍需依實際電路圖、MCU 等環境，自行定義變數等相關資料，本段程式碼無法組譯之後逕行使用。

```
;===============================================================================
;===============================================================================
; RFM-001 sample code
;===============================================================================
; MCU: 8051    X'TAL = 11.0592MHz
;===============================================================================
; REGISTER
;===============================================================================
DR0       equ  00H
DR1       equ  01H
DR2       equ  02H
DR3       equ  03H
DR4       equ  04H
DR5       equ  05H
DR6       equ  06H
DR7       equ  07H
;===============================================================================
; Variables
;===============================================================================
T0CountH  equ  08H            ; SYSTEM COUNTER, +1/50 ms
T0CountL  equ  09H            ; SYSTEM COUNTER, -1/195 us
;===============================================================================
; Buffer
;===============================================================================
RFIDBuf   equ  28H            ; 24 bytes.
StackBuf  equ  70H            ; 16 bytes.
;-------------------------------------------------------------------------------
          ORG  00H            ; RESET
          ljmp Reset
;-------------------------------------------------------------------------------
          ORG  0bH            ; TIMER0
          djnz T0CountL,sys_tm01
          inc  T0CountH
sys_tm01: reti
;-------------------------------------------------------------------------------
          ORG  40H
Reset:
          mov  P1,#11111111B
          mov  P3,#11111111B
          mov  SP,#StackBuf        ; Initial Stack Point.
          mov  IE,#00000000B       ; Disable Interrupt.
          mov  IP,#00000000B       ; Set Interrupt Priority
          mov  TMOD,#00100010B     ; Timer0, Timer1 Mode1
          mov  TCON,#00000101B     ; INT0, INT1 Falling Edge Triggered.
          mov  TL0,#256 - 180      ; 12*180*256/11059200 = 50ms.
          mov  TH0,#256 - 180      ;
          setb TR0                 ; TIME0 Start.
          setb ET0                 ; TIME0 Interrupt Enable.
          setb EA                  ; Enable Interrupt.

          lcall     RFM001Init
Main:
          mov       RFIDBuf,#02H
          mov       RFIDBuf,#01H
          lcall     RFM001Read     ; Read Multi-Page type, Page01
          jnc       read_fail
read_success:
          ; excute success function here.
          sjmp Main
```

```
read_fail:
          ; excute failure function here.
          sjmp Main
;=========================================================================
        $INCLUDE (RFM001.asm)
;=========================================================================
          END


;=============================================================================================
; RFM001.ASM
;=============================================================================================
; Definition.
;=============================================================================================
DTRSROnly      equ    07eH
DTRSRWrite     equ    0feH
;=============================================================================================
; Macro
;=============================================================================================
mTxctSet       macro                  ; Set Txct pin
               setb P3.2
               endm

mTxctClr       macro                  ; Clr Txct pin
               clr  P3.2
               endm

mScioSet       macro                  ; Set Scio pin
               setb P3.3
               endm

mScioClr       macro                  ; Clr Scio pin
               clr  P3.3
               endm

mScio_JSet     macro     Address      ; If Scio = Hi, jmp to Address
               jb   P3.3,Address
               endm

mScio_JClr     macro     Address      ; If Scio = Lo, jmp to Address
               jnb  P3.3,Address
               endm

mScio_Rd       macro                  ; C <-- Scio
               mov  C,P3.3
               cpl  C
               endm
;=============================================================================================
; Functions.
;=============================================================================================
; Initial Pins to RFM-001
;=============================================================================================
RFM001Init:
               mTxctSet
               mScioSet
               ret
;=============================================================================================
; Read Page    Return  C = 1, Success.
;              C = 0, Fail.
;         Pointer       0    1    2    3    4    5    6    7    8    9   10   11   12
;
;    Send RFIDBuf = [ CT + PN ]
;
;    Rece RFIDBuf = [ XX + ST + PN + XX + CT + D7 + D6 + D5 + D4 + D3 + D2 + D1 + D0 ]
;
; CT: Card Type
; PN: Page Number
; XX: RFU. Don't care
; ST: Status
; D7 ~ D0: Page Data, 8 bytes.
```

```
;=============================================================================================
RFM001Read:
               lcall     RFM001_Charge       ; RF charge start.
               mov       A,RFIDBuf
               cjne      A,#2,r001_rd1       ; Card Type = Multi-Page ?
               lcall     RFM001_MPage
               lcall     RFM001_WByte        ; Only Multi page needs to send this byte.
r001_rd1:      mov       A,T0CountL
               add       A,#150
               mov       B,A                 ; B = Time Out value. (20ms)
r001_rd2:      mov       R1,#RFIDBuf+13      ; R1 = data pointer.
               lcall     RFM001_RByte
               jnc       r001_rd5            ; jmp if Time Out.

               ;--- Check Start Byte ------------------------------------------------
               cjne      @R1,#DTRSROnly,r001_rd3
               sjmp      r001_rd4            ; jmp if READ_ONLY or MULTI_PAGE.
r001_rd3:      cjne      @R1,#DTRSRWrite,r001_rd2 ; Restart receiving.
r001_rd4:      dec       R1
               mov       R4,#13
               lcall     RFM001_RStr
               jnc       r001_rd5            ; jmp if Time Out.
               lcall     RFM001_Check        ; Check CRC, Type...
               jnc       r001_rd5            ; Read fail.
               ret                           ; Check ok, success.
r001_rd5:      clr    C
               ret
;=============================================================================================
; Write Page   Return  C = 1, Success.
;              C = 0, Fail.
;         Pointer       0    1    2    3    4    5    6    7    8    9   10   11   12
;
;    Send RFIDBuf = [ CT + PN + XX + XX + XX + D7 + D6 + D5 + D4 + D3 + D2 + D1 + D0 ]
;
;    Rece RFIDBuf = [ XX + ST + PN + XX + CT + D7 + D6 + D5 + D4 + D3 + D2 + D1 + D0 ]
;
; CT: Card Type
; PN: Page Number
; XX: RFU. Don't care
; ST: Status
; D7 ~ D0: Page Data, 8 bytes.
;=============================================================================================
RFM001Write:
               mov       A,RFIDBuf
               cjne      A,#1,r001_wt1       ; Card Type = Read/Write ?
               mov       RFIDBuf+14,#0bbH    ; save KEYWORD.
               mov       RFIDBuf+13,#0ebH    ; save PASSWORD.
               mov       R4,#8
               mov       R1,#RFIDBuf+12
               lcall     RFM001_CRCStr       ; Calculate CRC.
               mov       RFIDBuf+4,R6        ; save CRC LOW.
               mov       RFIDBuf+3,R7        ; save CRC HIGH.
               mov       RFIDBuf+2,#0        ; save TERMINATE WORD.
               mov       RFIDBuf+1,#3
               mov       R4,#14
               mov       R1,#RFIDBuf+14
               sjmp      r001_wt2

r001_wt1:      cjne      A,#2,r001_wt3
               lcall     RFM001_MPage
               orl       A,#01H
               mov       RFIDBuf+13,A        ; save PAGE.
               mov       R4,#8
               mov       R1,#RFIDBuf+12
               lcall     RFM001_CRCStr       ; Calculate CRC.
               mov       RFIDBuf+4,R6        ; save CRC LOW.
               mov       RFIDBuf+3,R7        ; save CRC HIGH.
               mov       R4,#11
               mov       R1,#RFIDBuf+13
               lcall     RFM001_CRCStr       ; Calculate CRC.
```

```
                mov       RFIDBuf+2,R6        ; save CRC LOW.
                mov       RFIDBuf+1,R7        ; save CRC HIGH.
                mov       R4,#13
                mov       R1,#RFIDBuf+13
r001_wt2:       lcall     RFM001_Charge
                lcall     RFM001_WStr
                lcall     RFM001_Prog
                ljmp      r001_rd1
r001_wt3:       clr       C
                ret
;=============================================================================================
; Lock Page     Return  C = 1, Success.
;               C = 0, Fail.
;          Pointer       0    1    2    3    4    5    6    7    8    9   10   11   12
;
;     Send RFIDBuf = [ CT + PN ]
;
;     Rece RFIDBuf = [ XX + ST + PN + XX + CT + D7 + D6 + D5 + D4 + D3 + D2 + D1 + D0 ]
;
; CT: Card Type
; PN: Page Number
; XX: RFU. Don't care
; ST: Status
; D7 ~ D0: Page Data, 8 bytes.
;=============================================================================================
RFM001Lock:
                mov       A,RFIDBuf
                cjne      A,#2,r001_wt3       ; Card Type = Multi-Page ?
                lcall     RFM001_MPage
                orl       A,#02H              ; 02 = Lock Command.
                mov       RFIDBuf+4,A         ; save PAGE.
                mov       R4,#1
                mov       R1,#RFIDBuf+4
                lcall     RFM001_CRCStr       ; Calculate CRC.
                mov       RFIDBuf+3,R6        ; save CRC LOW.
                mov       RFIDBuf+2,R7        ; save CRC HIGH.
                mov       R4,#3
                mov       R1,#RFIDBuf+4
                sjmp      r001_wt2
;=============================================================================================
; other routines
;=============================================================================================
RFM001_Charge:
                mov       A,#(256-255)        ; Delay 50 ms. (255*195us = 50ms)
                sjmp      r001_prog1
;---------------------------------------------------------------------------------------------
RFM001_Prog:
                mov       A,#(256-78)         ; Delay 15 ms. (78*195us = 15ms)
r001_prog1:     mTxctClr                      ; RF On.
                add       A,T0CountL
                cjne      A,T0CountL,$
                mTxctSet                      ; RF Off.
                ret
;---------------------------------------------------------------------------------------------
RFM001_MPage:
                mov       A,RFIDBuf+1
                rl        A
                rl        A
                anl       A,#0fcH
                ret
;=============================================================================================
; Read, Write Function.
;=============================================================================================
RFM001_RStr:
                lcall     RFM001_RByte
                jnc       rfid_rstr1          ; jmp if Time Out.
                dec       R1
                djnz      R4,RFM001_RStr
rfid_rstr1:     ret
;---------------------------------------------------------------------------------------------
```

```
RFM001_WStr:
               mov       A,@R1
               lcall     RFM001_WByte
               dec       R1
               djnz      R4,RFM001_WStr
               ret
;===============================================================================
RFM001_RByte:
               mov       R6,#8               ; R6 = Number of Bit read.
RFM001_RBit:
               mov       A,B                 ; B = Time Out value.
               cjne      A,T0CountL,r001_rbit1   ; jmp if No Time Out.
               clr       C
               ret
r001_rbit1:    mScio_JClr RFM001_Rbit        ; jmp if no data in.
               clr       EA                  ; Disable interrupt
               mov       R7,#32
               djnz      R7,$
r001_rbit2:    mScio_Rd                      ; Get RF data.
               mov       A,@R1
               rrc       A
               mov       @R1,A
               mov       R7,#27
               djnz      R7,$                ; Delay a while.
               djnz      R6,r001_rbit2
               setb      EA                  ; Enable interrupt
               setb      C                   ; Return TRUE
               ret
;-------------------------------------------------------------------------------
RFM001_WByte:
               mov       R6,#8               ; R6 = Number of bit written.
RFM001_WBit:
               jb        ACC.0,r001_wbit2    ; jmp if High.
               mov       R7,#133
               djnz      R7,$                ; delay 0.3 ms.
               mov       R7,#223
               mTxctClr                      ; Txct = 0.
r001_wbit1:    mov       DR7,R7              ; delay
               mov       DR7,R7              ; delay
               nop
               djnz      R7,r001_wbit1       ; delay 1.7 ms. (1566.72)
               sjmp      r001_wbit5

r001_wbit2:    mov       R7,#228
r001_wbit3:    mov       DR7,R7              ; delay.
               djnz      R7,r001_wbit3       ; delay 1 ms. (921.6)
               mov       R7,#230
               mTxctClr                      ; Txct = 0.
r001_wbit4:    mov       DR7,R7              ; delay
               djnz      R7,r001_wbit4       ; delay 1 ms. (921.6)
r001_wbit5:    mTxctSet                      ; Txct = 1.
               rr        A
               djnz      R6,RFM001_WBit
               ret
;===============================================================================
; Check Function.
;===============================================================================
RFM001_Check:
               mov       R4,#10                  ; R4 = 10 Bytes.
               mov       R1,#RFIDBuf+12          ; R1 = 2nd Byte.
               lcall     RFM001_CRCStr           ; Calculate CRC of 2nd - 11th Bytes.
               jnz       r001_chk3               ; jmp if CRC Error.
               mov       A,#DTRSROnly            ; RFIDBuf+13 = RFIDBuf+2 = DTRSROnly.
               cjne      A,RFIDBuf+13,r001_chk1  ; jmp if not Read Only or Multi-Page Type
               cjne      A,RFIDBuf+2,r001_chk2   ; jmp if not Read Only Type
               ;--- Check if Read Only Type ------------------------------------------
               mov       A,#7fH
               anl       A,RFIDBuf               ; Last byte just only 7 bits effective.
               orl       A,RFIDBuf+1
               jnz       r001_chk3               ; jmp if last 2 bytes are not 00, fail.
```

```
                mov     RFIDBuf+4,#0            ; Return Card Type = READ_ONLY.
                mov     RFIDBuf+1,#0            ; Status = 0
                mov     RFIDBuf+2,#0            ; PN = 0
                setb    C
                ret

                ;--- Check if Read/Write Type ----------------------------------------
r001_chk1:      mov     A,#DTRSRWrite           ; RFIDBuf+13 = RFIDBuf+2 = DTRSRWrite.
                cjne    A,RFIDBuf+13,r001_chk3  ; jmp if Byte1  != FEH
                cjne    A,RFIDBuf+2,r001_chk3   ; jmp if Byte12 != FEH
                mov     A,RFIDBuf+12
                cjne    A,RFIDBuf+1,r001_chk3   ; jmp if Byte2 != Byte13
                mov     A,#7fH
                anl     RFIDBuf,A               ; Last byte just only 7 bits effective.
                anl     A,RFIDBuf+11
                cjne    A,RFIDBuf,r001_chk3     ; jmp if Byte3 != Byte14
                mov     RFIDBuf+4,#1            ; Return Card Type = Read/Write
                mov     RFIDBuf+1,#0            ; Status = 0
                mov     RFIDBuf+2,#0            ; PN = 0
                setb    C
                ret

                ;--- Check if Multi-Page Type
r001_chk2:      mov     R4,#3                   ; R4 = 3 Bytes.
                mov     R1,#RFIDBuf+2           ; R1 = 12th Byte.
                lcall   RFM001_CRCStr           ; Calculate CRC of 12th~14th Byte.
                ; (Because CRC of 2nd~11th Byte is 0000, ignore it.)
                jnz     r001_chk3               ; jmp if CRC-1 error, fail.
                mov     A,RFIDBuf+2             ; Get [Read Address]
                mov     R4,A
                anl     A,#03H                  ; Get Status
                mov     RFIDBuf+1,A             ; save STATUS.
                mov     A,R4
                anl     A,#0fcH                 ; Get Page Number
                jz      r001_chk3               ; If Page Number = 0, Error.
                rr      A
                rr      A
                mov     RFIDBuf+2,A             ; save PAGE.
                mov     RFIDBuf+4,#2            ; Return Card Type = Multi-Page
                setb    C
                ret
r001_chk3:      clr     C
                ret
;=================================================================================================
; Calculate CRC string.
;=================================================================================================
RFM001_CRCStr:
                mov     R6,#0                   ; Initial CRC.
                mov     R7,#0                   ; R6 = CRC LOW, R7 = CRC HIGH.
r001_crcstr1:
                mov     A,@R1                   ; Get data.
                lcall   RFM001_CRCByte          ; Calculate
                dec     R1
                djnz    R4,r001_crcstr1
                mov     A,R7                    ; Check if CRC is 0, if 0 then OK, else Error.
                orl     A,R6
                ret
;-------------------------------------------------------------------------------------------------
RFM001_CRCByte:
                mov     R5,#8                   ; Calcualte 8 Bits.
                xrl     A,R6
                mov     R6,A
RFM001_CRCBit:
                clr     C
                mov     A,R7
                rrc     A                       ; Shift rigth CRC HIGH , C <- LSB.
                mov     R7,A
                mov     A,R6
                rrc     A                       ; Shift right CRC LOW  , C <- LSB.
                mov     R6,A
```

```
                jnc       r001_crcbit1            ; If C=0, do nothing.
                mov       A,R7
                xrl       A,#84H                  ; Reverse Bit-15, Bit-10.
                mov       R7,A
                mov       A,R6
                xrl       A,#08H                  ; Reverse Bit-03.
                mov       R6,A
r001_crcbit1:   djnz      R5,RFM001_CRCBit
                ret
```