# 8-Bit Microcontroller

# TM57 系列

# MCU 掉电记忆应用 AP note

# Application Note

# AMENDMENT HISTORY

| Version | Date | Description |
|---|---|---|
| V1.0 | Nov, 2011 | New release |

# CONTENTS

# 前言

注:本文件以 TM57PA40 为蓝本来说明 MCU 掉电记忆中 IC 特性、硬件、程序处理中应注意的问题。

目前在："烤面包机(吐司炉)"、"电饭煲"、"电压力锅"等小家电应用中,掉电记忆功能已成为基本要求。要求的记忆时间从 20 分钟到 2 个小时不等。该种功能可以保证产品在工作过程中市电意外停电再恢复后,可以自动恢复到原来的工作模式下继续工作,而不需要人为干预。这给用户的使用带来一定的便利。掉电记忆功能的实现,可以通过主控 IC 外置或内置 EEROM 的方式来实现,但这样成本较高,不经济。一种比较可行的方式是利用主控 IC SRAM 在电源有一点存电的情况下,内部数据不会丢失的特性来实现掉电记忆功能。记忆时间的长短,决定于两个方面:1.>电源断电后,IC 电源电压的保持时间。这取决于电源部分电解电容的大小和线路的放电电流。2.>IC SRAM 数据能正常保存而不丢失的最低工作电压,每家 IC 的特性不同,该最低电压存在差异。

针对掉电记忆功能的应用,IC 特性,硬件线路和程序处理分列如下:

## 一. IC 的特性：

TM57PA40/20 SRAM 数据最低安全保存电压点为 **0.8V**。在 0.8V 以上(Vcc>=0.8V),所有 SRAM 数据都能够正常保存；在 0.8V 以下 0.1V 以上(0.8V>Vcc>=0.1V),有些 SRAM 数据可以正常保存,而另一些 SRAM 数据则会丢失,这种现象有一定的随机性,并不能确定哪些地址区间的 SRAM 能正常保存,哪些不能；0.1V 以下(Vcc<0.1V)则所有 SRAM 数据都不能正常保存。编程时必须对这一现象加以考虑,以避免掉电后再上电程序运行出错。

## 二. 硬件处理：

线路处理上,为了保证足够的掉电记忆时间,IC 电源必须用二极管和其它外围线路进行隔离,IC 电源部分电解电容在允许的情况下尽可能选大一些。由于不同品质及型号的二极管反向漏电流不同,漏电流会缩短掉电记忆时间,因此选用二极管时,要选择反向漏电流小的产品。同理,电解电容也有漏电流问题,在选择时也应选择漏电流小的产品。此外,因电解电容反向漏电流大,所以在生产时,极性不可以插反。

## 三. 程序的处理：

程序原理是：定义几个变量("SRAM 保持变量")作为"SRAM 有记忆 & SRAM 无记忆"的判断条件,变量数目在程序资源允许的情况下尽可能多，此例程中定义 5 个：SRAMHOLD1、SRAMHOLD2、SRAMHOLD3、SRAMHOLD4、SRAMHOLD5。上电后判断"SRAM 保持变量"是否为"设定值",如果是,则表明"SRAM 有记忆",程序不对"被保护变量"进行初始化；如果不是,则表明"SRAM 无记忆",需对"被保护变量"进行初始化,并且给"SRAM 保持变量"赋"设定值"。

应用中,注意事项如下：

1. SRAMHOLD1、SRAMHOLD2、SRAMHOLD3、SRAMHOLD4、SRAMHOLD5 的"设定值"最好不要赋相同的值,尽可能赋值内容与其地址相同为好。地址要尽可能定义在 SRAM 区的不同区域,如 20H、45H、58H、6AH、7FH 等。
2. 对于 SAVE_DATA1、SAVE_DATA2、SAVE_DATA3、SAVE_DATA4、SAVE_DATA5 等"被保护变量"，需要做备份：SAVE_DATA1_BUF、SAVE_DATA2_BUF、SAVE_DATA3_BUF、SAVE_DATA4_BUF、SAVE_DATA5_BUF。"备份变量"和"被保护变量"地址尽量安排在 SRAM 区的不同区域。上电后需要 1.>判断"被保护变量"的值是否在允许的范围内；2.>判断"被保护变量"和"备份变量"的值是否相同。如果 1.>、2.>条件同时满足,则"

被保护变量"记忆完好,不需进行初始化；否则"被保护变量"记忆丢失,需对其重新进行初始化。

3. 任何情况下,都需要对程序中使用的"非保护变量"进行初始化；有些"非保护变量"(注*A,NON_SAVE_DATA1)如果不方便重新做初始化而又相对不重要,没有做"备份变量"时,必须要判断其值是否在有效范围之内,如果不在有效范围,则必须进行初始化。
4. 4."SRAM 保持变量"、"被保护变量"、"备份变量"的地址应避免和程序中需要经常写入数据的"非保护变量"地址交叉（请参考 SRAM 掉电丢失数据分析图）。如需要经常写入数据的"非保护变量"地址为 53H,那么"SRAM 保持变量"、"被保护变量"、"备份变量"的地址就不要选?3H 或 5?H 等。在 SRAM 有两个 BANK 的情况下,"SRAM 保持变量"、"被保护变量"、"备份变量"和"非保护变量"安排在不同 BANK 也是一个比较好的选择。
5. Vcc=3.6V 以上应用时,LVR 必须开 3.1V。
6. 如果程序选择使用 WDT,在上电后必须要判断是否为 WDT 复位。如是 WDT 复位，则需对所有变量重新进行初始化。

## 范例:

注：范例程序只是初始化的一部分。

```
;==============================
#define c 03h,0
#define z 03h,2
#define _to 03h,4
;----------------------
bank0 macro
  bcf  03h,5
  endm
bank1 macro
  bsf  03h,5
  endm
;;
w  equ  0
f  equ  1
;----------------------
;BANK1
SRAMHOLD1  equ  20H
SRAMHOLD2  equ  45H
SRAMHOLD3  equ  58H
SRAMHOLD4  equ  6AH
SRAMHOLD5  equ  7FH
;;
SAVE_DATA1 equ  23H
SAVE_DATA2 equ  47H
SAVE_DATA3 equ  5aH
SAVE_DATA4 equ  6eH
SAVE_DATA5 equ  70H
```

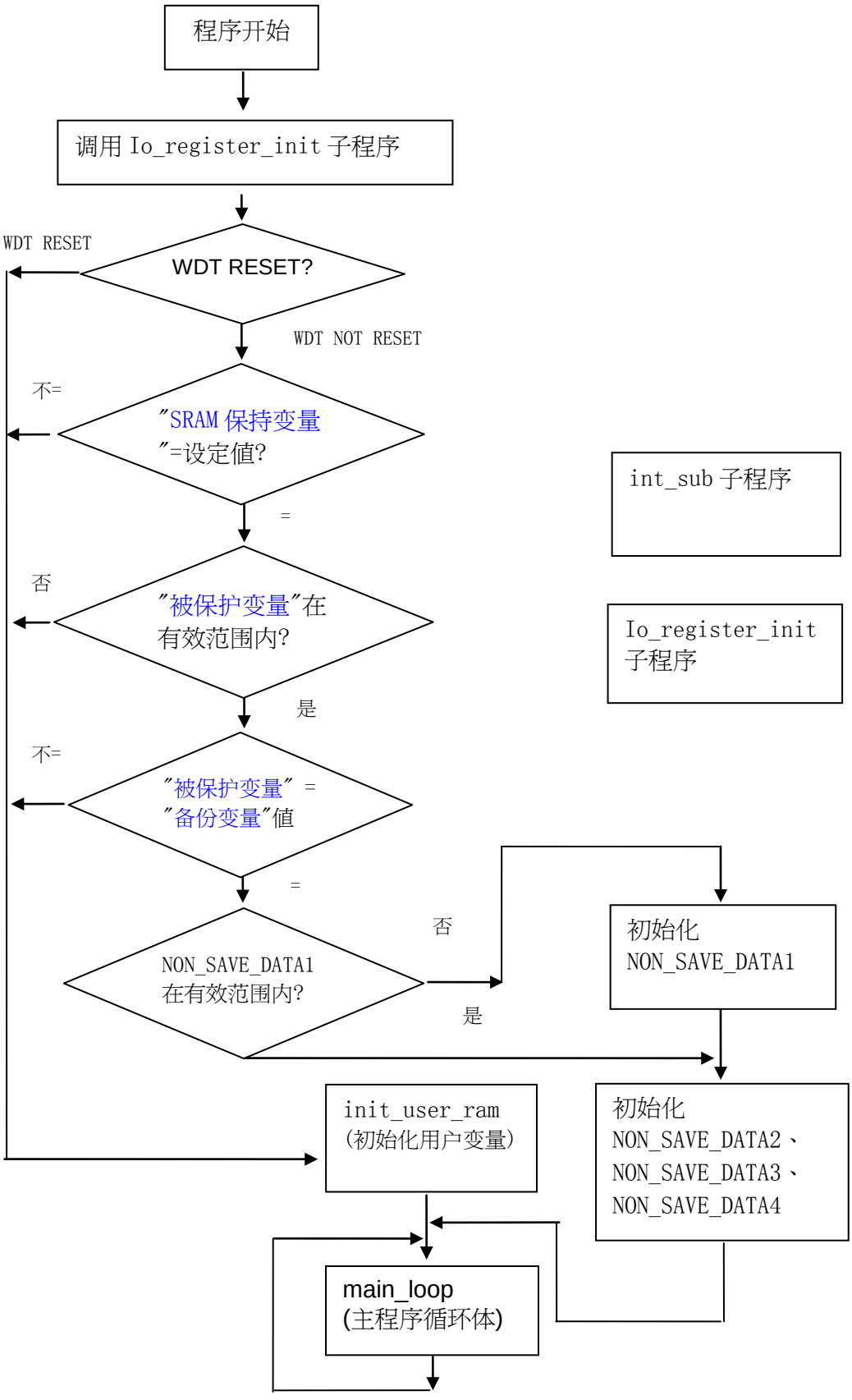
```
;;
SAVE_DATA1_BUF equ  28H
SAVE_DATA2_BUF equ  4aH
SAVE_DATA3_BUF equ  5dH
SAVE_DATA4_BUF equ  6fH
SAVE_DATA5_BUF equ  75H
;----------------------
;BANK0
NON_SAVE_DATA1 equ  20H
NON_SAVE_DATA2 equ  22H
NON_SAVE_DATA3 equ  40H
NON_SAVE_DATA4 equ  42H
;
;
;----------------------
注:".."代表略掉的程序
;----------------------
;程序区
   org    000H
   goto   main
   org    001H
   goto   int_sub
..
..
main:
call   Io_register_init                ;I/O 和寄存器初始化
;-------;判断是否是 WDT 复位
   btfsc  _to
goto   init_user_ram       ;WDT RESET,初始化用户变量
clrwdt                  ;WDT 没有 RESET
;-------;判断 SRAM 记忆是否丢失
bank1
movlw  SRAMHOLD1
xorwf   SRAMHOLD1,w
btfss   z
goto    init_user_ram      ;记忆丢失,初始化用户变量
movlw   SRAMHOLD2
xorwf   SRAMHOLD2,w
btfss   z
goto    init_user_ram      ;记忆丢失,初始化用户变量
movlw   SRAMHOLD3
xorwf   SRAMHOLD3,w
btfss   z
goto    init_user_ram      ;记忆丢失,初始化用户变量
movlw   SRAMHOLD4
xorwf   SRAMHOLD4,w
btfss   z
goto    init_user_ram      ;记忆丢失,初始化用户变量
```

```
movlw   SRAMHOLD5
xorwf   SRAMHOLD5,w
btfss   z
goto    init_user_ram     ;记忆丢失,初始化用户变量
;;判断"被保护变量"值是否在有效范围内
bank0
movlw   25H                       ;有效值范围依程序具体要求而定
subwf   SAVE_DATA1,w
btfss   c
goto    init_user_ram     ;不在有效范围,初始化用户变量
movlw   60H                       ;有效值范围依程序具体要求而定
subwf   SAVE_DATA1,w
btfsc   c
goto    init_user_ram     ;不在有效范围,初始化用户变量
;;
movlw   10H                       ;有效值范围依程序具体要求而定
subwf   SAVE_DATA2,w
btfss   c
goto    init_user_ram     ;不在有效范围,初始化用户变量
;;
movlw   30H                       ;有效值范围依程序具体要求而定
subwf   SAVE_DATA3,w
btfsc   c
goto    init_user_ram     ;不在有效范围,初始化用户变量
;;
movlw   50H                       ;有效值范围依程序具体要求而定
subwf   SAVE_DATA4,w
btfss   c
goto    init_user_ram     ;不在有效范围,初始化用户变量
;;
testz   SAVE_DATA5        ;有效值范围依程序具体要求而定
btfss   z
goto    init_user_ram     ;不在有效范围,初始化用户变量
    ;;判断"被保护变量"值是否和"备份变量"值相同
movfw   SAVE_DATA1
xorwf   SAVE_DATA1_BUF,w
btfss   z
goto    init_user_ram     ;"被保护变量"和"备份变量"值不相同,初始化用户变量
;;
movfw   SAVE_DATA2
xorwf   SAVE_DATA2_BUF,w
btfss   z
goto    init_user_ram     ;"被保护变量"和"备份变量"值不相同,初始化用户变量
;;
movfw   SAVE_DATA3
xorwf   SAVE_DATA3_BUF,w
btfss   z
```

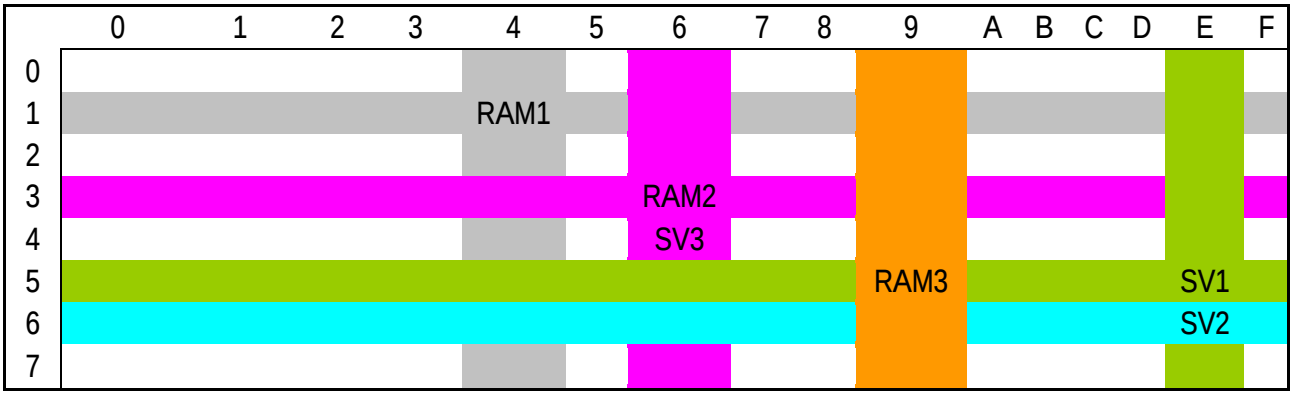
```
goto    init_user_ram     ;"被保护变量"和"备份变量"值不相同,初始化用户变量
;;
movfw   SAVE_DATA4
xorwf   SAVE_DATA4_BUF,w
btfss   z
goto    init_user_ram     ;"被保护变量"和"备份变量"值不相同,初始化用户变量
;;
movfw   SAVE_DATA5
xorwf   SAVE_DATA5_BUF,w
btfss   z
goto    init_user_ram     ;"被保护变量"和"备份变量"值不相同,初始化用户变量
;; "非保护变量"初始化
;;注*A
movlw   ??H               ;有效值范围依程序具体要求而定
subwf   NON_SAVE_DATA1,w
btfss   c
   goto    $+.3           ;值在有效范围,不需初始化
movlw   ??H                                 ;值不在有效范围,初始化 NON_SAVE_DATA1
movwf   NON_SAVE_DATA1
;;
movlw   ??H                                 ;初始化 NON_SAVE_DATA2
movwf   NON_SAVE_DATA2
movlw   ??H                                 ;初始化 NON_SAVE_DATA3
movwf   NON_SAVE_DATA3
movlw   ??H                                 ;初始化 NON_SAVE_DATA4
movwf   NON_SAVE_DATA4
;;
goto    main_loop
   ;-------
init_user_ram:    ;初始化用户变量
   bank1
movlw   SRAMHOLD1
movwf   SRAMHOLD1
movlw   SRAMHOLD2
movwf   SRAMHOLD2
movlw   SRAMHOLD3
movwf   SRAMHOLD3
movlw   SRAMHOLD4
movwf   SRAMHOLD4
movlw   SRAMHOLD5
movwf   SRAMHOLD5
;;
bank0
movlw   ??H
movwf   SAVE_DATA1
movwf   SAVE_DATA1_BUF
movlw   ??H
movwf   SAVE_DATA2
```

```
movwf   SAVE_DATA2_BUF
movlw   ??H
movwf   SAVE_DATA3
movwf   SAVE_DATA3_BUF
movlw   ??H
movwf   SAVE_DATA4
movwf   SAVE_DATA4_BUF
movlw   ??H
movwf   SAVE_DATA5
movwf   SAVE_DATA5_BUF
;;
movlw   ??H
movwf   NON_SAVE_DATA1
movlw   ??H
movwf   NON_SAVE_DATA2
movlw   ??H
movwf   NON_SAVE_DATA3
movlw   ??H
movwf   NON_SAVE_DATA4
;;
main_loop:     ;主程序循环
..
..
goto   main_loop
..
..
;
;
;-----------------------
;;中断子程序
int_sub:
..
..
reti
;
;
;-----------------------
;;I/O & 寄存器初始化子程序
;;I/O,定时器,中断等的初始化
Io_register_init:
..
..
ret
;
;
```

## 流程图


程序开始
调用 Io_register_init 子程序
WDT RESET
WDT RESET?
WDT NOT RESET
不=
″SRAM 保持变量″=设定值?
=
否
″被保护变量″在有效范围内?
是
不=
″被保护变量″ = ″备份变量″值
=
否
NON_SAVE_DATA1 在有效范围内?
是
int_sub 子程序
Io_register_init 子程序
初始化 NON_SAVE_DATA1
init_user_ram (初始化用户变量)
初始化 NON_SAVE_DATA2、NON_SAVE_DATA3、NON_SAVE_DATA4
main_loop (主程序循环体)

## SRAM 掉电丢失数据分析图




注:RAM1,RAM2,RAM3----LVR 动作期间有可能动到的变量

注:SV1,SV2,SV3 需要记忆的变量

理论上 **SV1,SV3** 都有机会被改变到,只有 **SV2** 才是安全的

如果把 **SV1,SV2,SV3** 按排在另一个 **BANK** 里面,则比较没有问题

| | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | SRAM address |
|---|---|---|---|---|---|---|---|---|---|
| **Normal good!** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 20H |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 21H |
| | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 22H |
| | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 70H |
| | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 71H |
| | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 72H |
| | | | | | | | | | |
| **Best!!** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 20H |
| | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5FH |
| | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 6DH |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 32H |
| | | | 類推 | | | | | | |

**Note: A7~A3: SRAM wordline**