

Sistemas Baseados em Microprocessadores

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Faculdade de Engenharia



ATMega328P – EEPROM (datasheet: 12.4 e 12.6)



João Paulo de Sousa

EEPROM - Características

- Capacidade:
 - ATmega48/88/168/328 = 256/512/512/1024 bytes
- Resistência ao desgaste:
 - Apagamentos / Escritas com apagamento: $>10^5$
 - Leituras: ilimitadas
- Retenção de dados:
 - 100 anos a 25°C, 20 anos a 85°C
 - Conclusão: não é para sempre, mas quase...

EEPROM – Operações e Registos associados

- Leitura: CPU pára durante 4 ciclos de relógio
- Escrita: CPU pára durante 2 ciclos de relógio
 - Procedimento especial para evitar escritas acidentais
- Registos associados:
 - **EECR** - EEPROM Control Register
 - **EEARH, EEARL** - EEPROM Address Register (16bits)
 - **EEDR** - EEPROM Data Register


EEPROM – Registo de controlo

–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
---	---	-------	-------	-------	-------	------	------

EECR - EEPROM Control Register:

- EEPE, EERE: Iniciam as operações de escrita e leitura
- EEMPE: habilitação do bit EEPE (durante 4 impulsos de clk)
- EERIE: habilita geração de interrupção após uma escrita
- EEPM1, EEPM0: configuração do modo de escrita:
 - 0,0 = Apaga e escreve numa única operação (3,4ms)
 - 0,1 = Apaga apenas (1,8 ms); 1,0 = Escreve apenas (1,8 ms)
 - 1,1 = Reservado

EEPROM – Biblioteca avr-libc

- Atributo para variáveis: EEMEM
 - Exemplo: `unsigned char EEMEM best = 55;`
 - Inicialização via ficheiro separado a gravar no uC
 - Variáveis não inicializadas têm todos os bits a um
- } 
- Funções disponíveis (`#include avr/eeprom.h`):
 - `eeprom_read_byte`; `eeprom_update_byte`
 - `eeprom_read_word`; `eeprom_update_word`
 - `eeprom_read_block`; `eeprom_update_block`

EEPROM – Questões adicionais

- Inicializar sempre? Não!
- Como detetar se é a primeira vez que um programa está a usar a memória? Usar assinaturas.
- Assinatura = padrão de bits específico do programa.

Assinaturas em EEPROM

- Exemplos de assinaturas:

```
#define SIG08 0b01010101
```

```
#define SIG16 0b1011000101011010
```

- Algoritmo:
 - Ler endereço(s) onde deve estar a assinatura
 - Comparar com valor pré-definido
 - Se for igual assumir que a EEPROM já foi inicializada
 - Senão, assinar esse(s) endereço(s) e inicializar variáveis residentes em EEPROM

EEPROM – Exemplo

- Ler, processar e reescrever um byte na EEPROM

```
#include <avr/io.h>
#include <avr/eeprom.h>

#define LED PB5

uint8_t EEMEM best = 55;

void main(void) {
    uint8_t nvbest, average;
    average=143;
    DDRB = DDRB | (1<<LED);
    nvbest = eeprom_read_byte(&best);
    if (average > nvbest){
        eeprom_update_byte(&best,average);
        PORTB = PORTB | (1<<LED);
    }
    while(1);
}
```

AVR Memory Usage

Device: atmega328p

Program:	264 bytes (0.9% Full)
Data:	0 bytes (0.0% Full)
EEPROM:	1 bytes (0.1% Full)

- Experimentar em casa:
 - Variáveis globais/locais
 - Tipos 8/16 bits
 - Ver código gerado (*.lss)

EEPROM – Exemplo

- Assembly gerado pelo compilador (ficheiro *.lss)...

```
00000000 <__vectors>:
 0: 0c 94 34 00 jmp 0x68 ; 0x68 <__ctors_end>
 4: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
 8: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
 c: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
10: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
14: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
18: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
1c: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
20: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
24: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
28: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
2c: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
...

50: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
54: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
58: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
5c: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
60: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
64: 0c 94 3e 00 jmp 0x7c ; 0x7c <__bad_intrp>
```

```
00000068 <__ctors_end>:
68: 11 24 eor r1, r1
6a: 1f be out 0x3f, r1 ; 63
6c: cf ef ldi r28, 0xFF ; 255
6e: d8 e0 ldi r29, 0x08 ; 8
70: de bf out 0x3e, r29 ; 62
72: cd bf out 0x3d, r28 ; 61
74: 0e 94 40 00 call 0x80 ; 0x80 <main>
78: 0c 94 9f 00 jmp 0x13e ; 0x13e <_exit>

0000007c <__bad_intrp>:
7c: 0c 94 00 00 jmp 0 ; 0x0 <__vectors>

00000080 <main>:
80: cf 93 push r28
82: df 93 push r29
...

0000013e <_exit>:
13e: f8 94 cli

00000140 <__stop_program>:
140: ff cf rjmp .-2 ; 0x140 <__stop_program>
```

EEPROM – Exemplo

- O mesmo código, depois de uma limpeza...

```
;-----  
; Reset and other interrupt vectors:  
; - Redirect execution to init on reset;  
; - Redirect execution to a safe place  
;   on each undefined interrupt entry  
;-----  
reset:      jmp     init  
  
vectors:    jmp     bad_int      ; INT0 interrupt  
            jmp     bad_int      ; INT1  
            jmp     bad_int      ; PCINT0  
            jmp     bad_int      ; PCINT1  
            jmp     bad_int      ; PCINT2  
            jmp     bad_int      ; WDT  
            jmp     bad_int      ; TIMER2 COMPA  
            jmp     bad_int      ; TIMER2 COMPB  
            jmp     bad_int      ; TIMER2 OVF  
            ...  
            jmp     bad_int      ; TWI  
            jmp     bad_int      ; SPM READY
```

```
;-----  
; Startup code (C Run Time):  
; - SREG=0 (clear flags and I bit)  
; - Make SP pointing to the top of the available RAM  
; - Call main() and make sure the program never ends  
;-----  
init:       eor     R1, R1      ; R1 = 0  
            out     SREG, R1    ; Initialize SREG  
            ldi     R28, 0xFF    ; Initialize SP:  
            ldi     R29, 0x08    ; The ATmega328 has 0x800  
            out     SPH, R29     ; bytes of RAM starting  
            out     SPL, R28     ; at 0x100...  
            call    main        ; Run the user program  
            cli      ; Disable all interrupts  
            rjmp    $           ; Infinite loop  
  
bad_int:     jmp     reset  
  
main:        push   R28  
            push   R29  
            ...
```

Dúvidas

Para aprofundar:

- Datasheet: secções 12.4 e 12.6
- [Manual](#) de referência avr-libc: secção 6.13
- Application Notes:
 - [AVR100](#): Accessing the EEPROM
 - [AVR101](#): High Endurance EEPROM Storage
 - [AVR103](#): Using the EEPROM Programming Modes
 - [AVR104](#): Buffered Interrupt Controlled EEPROM Writes

