Programowanie mikrokontrolerów

Pamięci nieulotne

Marcin Engel Marcin Peczarski

Instytut Informatyki Uniwersytetu Warszawskiego

30 listopada 2012

Dostępne pamięci nieulotne

- ATmega16 posiada wewnętrzną pamięć danych EEPROM.
- ATmega16 posiada wewnętrzną pamieć programu FLASH.
- W zestawie uruchomieniowym jest zainstalowana zewnętrzna pamięć danych FLASH.

Wewnętrzna pamięć danych EEPROM

- W ATmega16 ma 512 bajtów.
- Producent gwarantuje minimum 100000 cykli zapisu.
- Odczyt wstrzymuje procesor na 4 cykle zegara.
- Zapis wstrzymuje procesor na 2 cykle zegara.
- Zapis po zainicjowaniu trwa ok. 8,5 ms.
- Pamięć jest widziana jako układ peryferyjny przez rejestry wejścia-wyjścia:
  - `EEARH`, `EEARL` – zawierają adres do odczytu lub zapisu,
  - `EEDR` – zawiera odczytany bajt lub bajt przeznaczony do zapisania,
  - `EECR` – steruje pracą pamięci.

Rejestr EECR

EEPROM Control Register

7	6	5	4	3	2	1	0
–	–	–	–	EERIE	EEMWE	EEWE	EERE

- EERIE – uaktywnia przerwanie gotowości do zapisu. Przerwanie jest zgłaszane w sposób ciągły, gdy bit EEWE ma wartość 0.
- EEMWE – rozpoczyna procedurę zapisu. W celu wykonania zapisu należy w ciągu 4 cykli zegara od ustawienia bitu EEMWE ustawić bit EEWE.
- EEWE – startuje zapis. Bit pozostaje ustawiony aż do zakończenia procedury zapisu.
- EERE – startuje odczyt.

Procedura zapisu

- Zaczynamy jak zwykle.

```
EEPROM_write:
        push r19
```

- Czekamy na zakończenie poprzedniego zapisu.

```
EEPROM_write_wait:
        sbic EECR, EEWE
        rjmp EEPROM_write_wait
```

- Opcjonalnie czekamy na zakończenie operacji na pamięci programu.

```
EEPROM_write_FLASH_wait:
        in r19, SPMCR
        sbrc r19, SPMEN
        rjmp EEPROM_write_FLASH_wait
```

- Zapisujemy adres i dane.

```
        out EEARH, r17
        out EEARL, r16
        out EEDR, r18
```

Procedura zapisu, cd.

- Blokujemy przerwania.

```
in  r19, SREG
cli
```

- Inicjujemy zapis.

```
sbi EECR, EEMWE
sbi EECR, EEWE
```

- Przywracamy poprzednią wartość znacznika przerwań.

```
out SREG, r19
```

- Wracamy.

```
pop r19
ret
```

Procedura odczytu

- Czekamy na ewentualne zakończenie zapisu.

```
EEPROM_read:
        sbic EECR, EEWE
        rjmp EEPROM_read
```

- Ustawiamy adres.

```
        out EEARH, r17
        out EEARL, r16
```

- Czytamy.

```
        sbi EECR, EERE
        in r18, EEDR
```

- Wracamy.

```
        ret
```

Przykład użycia

- Deklarujemy zmienną nieulotną i ją inicjujemy.

```
.eseg
.org $30
zmienna: .db 4
```

- Zapisujemy nową wartość.

```
ldi r16, LOW(zmienna)
ldi r17, HIGH(zmienna)
ldi r18, $17
rcall EEPROM_write
```

- Czytamy wartość zmiennej.

```
ldi r16, LOW(zmienna)
ldi r17, HIGH(zmienna)
rcall EEPROM_read
```

Przydatne makro

- Pierwszym argumentem jest nazwa rejestru górnego, a drugim adres w EEPROM.

```
.macro lde
        sbic EECR, EEWE
        rjmp PC - 1
        ldi @0, HIGH(@1)
        out EEARH, @0
        ldi @0, LOW(@1)
        out EEARL, @0
        sbi EECR, EERE
        in @0, EEDR
.endmacro
```

- Przykład użycia

```
        lde r16, zmienna
```

Uwagi dotyczące używania EEPROM

- EEPROM współdzieli mechanizm zapisu z pamięcią programu FLASH.
- Jeśli program wykonuje również zapisy do FLASH, to w procedurach obsługi EEPROM należy również czekać na zakończenie operacji na pamięci FLASH.

- Trwający zapis blokuje wyłączenie zegara i uniemożliwia wejście w niektóre tryby uśpienia, np. Power-down.
- Przed wejściem w tryb uśpienia należy poczekać na zakończenie zapisu.

- Zbyt niskie napięcie zasilania może spowodować błąd zapisu.
- Należy używać układu BOD.

- Reset nie przerywa operacji zapisu o ile dostępne jest zasilanie.

Pamięć programu FLASH

- W ATmega16 ma 16 kB (8192 słów).
- Może być modyfikowana programowo.
- Jest podzielona na strony – w ATmega16 mają rozmiar 128 B.
- Strony można kasować i zapisywać tylko w całości.
- Producent gwarantuje minimum 10000 cykli zapisu.

Pamięć programu FLASH, cd.

- Jest podzielona fizycznie na dwie sekcje:
  - RWW (Read While Write), w ATmega16 adresy \$0000–\$1BFF,
  - NRWW (No Read While Write), w ATmega16 adresy \$1C00–\$1FFF.
- Program wykonujący się w sekcji NRWW może za pomocą instrukcji `SPM` modyfikować sekcję RWW bez wstrzymywania CPU.
- Program wykonujący się w sekcji NRWW może też modyfikować sekcję NRWW, ale wstrzymuje to pracę CPU.
- Instrukcja `SPM` nie może być wykonana z sekcji RWW.
- Jest podzielona logicznie na dwie sekcje:
  - aplikacji,
  - BLS (Boot Loader Section).

Logiczna organizacja pamięci programu

- Jest określona przez bity konfiguracyjne (ang. fuse bits) i bit IVSEL w rejestrze GICR.
- Podział na sekcje

BOOTSZ1	BOOTSZ0	sekcja aplikacji	boot loader
1	1	\$0000–\$1E7F	\$1F80–\$1FFF
1	0	\$0000–\$1EFF	\$1F00–\$1FFF
0	1	\$0000–\$1DFF	\$1E00–\$1FFF
0	0	\$0000–\$1BFF	\$1C00–\$1FFF

- Położenie wektora przerwań

BOOTRST	IVSEL	reset	pozostałe przerwania
1	0	\$0000	\$0002
1	1	\$0000	boot loader + \$0002
0	0	boot loader	\$0002
0	1	boot loader	boot loader + \$0002

Organizacja pamięci programu, cd.

- Jest opisana w pliku nagłówkowym (`m16def.inc` dla ATmega16).

```
;($1F80) smallest boot block is 256B
.equ   SMALLBOOTSTART  = 0b1111110000000

;($1F00) second boot block size is 512B
.equ   SECONDBOOTSTART = 0b1111100000000

;($1E00) third boot block size is 1kB
.equ   THIRDBOOTSTART  = 0b1111000000000

;($1C00) largest boot block is 2kB
.equ   LARGEBOOTSTART  = 0b1110000000000

.equ   PAGESIZE = 64   ;number of WORDS in a page

.equ   FLASHEND = $1fff
```

Zmiana bitu IVSEL

- Należy ustawić bit IVCE.

```
in r16, GICR
sbr r16, 1 << IVCE
out GICR, r16
```

- A następnie w ciągu 4 taktów zegara ustawić właściwą wartość bitu IVSEL, zerując bit IVCE.

```
sbr r16, 1 << IVSEL
cbr r16, 1 << IVCE
out GICR, r16
```

- Przerwania są automatycznie blokowane w trakcie powyższej procedury.

Instrukcja SPM

- Umożliwia:
  - skasowanie strony,
  - przepisanie danych z SRAM do bufora zapisu FLASH,
  - zaprogramowanie strony z bufora zapisu,
  - odblokowanie odczytu sekcji RWW,
  - ustawienie bitów zapezpieczających BLB12, BLB11, BLB10, BLB01 (ang. boot loader lock bits).
- Operacja jest wybierana za pomocą rejestru SPMCR.
- W rejestrze `Z` umieszcza się adres strony lub słowa w buforze.
- W parze rejestrów `R1:R0` umieszcza się dane do zapisania.
- Kasowanie i zapisywanie strony oraz ustawianie bitów zabezpieczających trwa ok. 4 ms.

Rejestr SPMCR

Store Program Memory Control Register

7	6	5	4	3	2	1	0
SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN

- SPMIE – uaktywnia przerwanie gotowości SPM. Przerwanie jest zgłaszane w sposób ciągły, dopóki bit SPMEN jest wyzerowany.
- Znaczenie pozostałych bitów najlepiej jest poznać, oglądając przykład użycia w dokumentacji.

Interfejs SPI

- Serial Peripheral Interface
- Interfejs szeregowy, synchroniczny, dwukierunkowy (ang. *full-duplex*)
- Trzy linie:
  - linia zegarowa (SCK, CLK) – jedno urządzenie (master) generuje na niej sygnał zegarowy,
  - linia Master Out Slave In, po której master przesyła swoje dane do urządzenia slave,
  - linia Master In Slave Out, po której slave przesyła swoje dane do urządzenia master.
- Do magistrali może być przyłączonych wiele urządzeń.
- Konkretny pakiet jest przesyłany zawsze między dwoma urządzeniami.

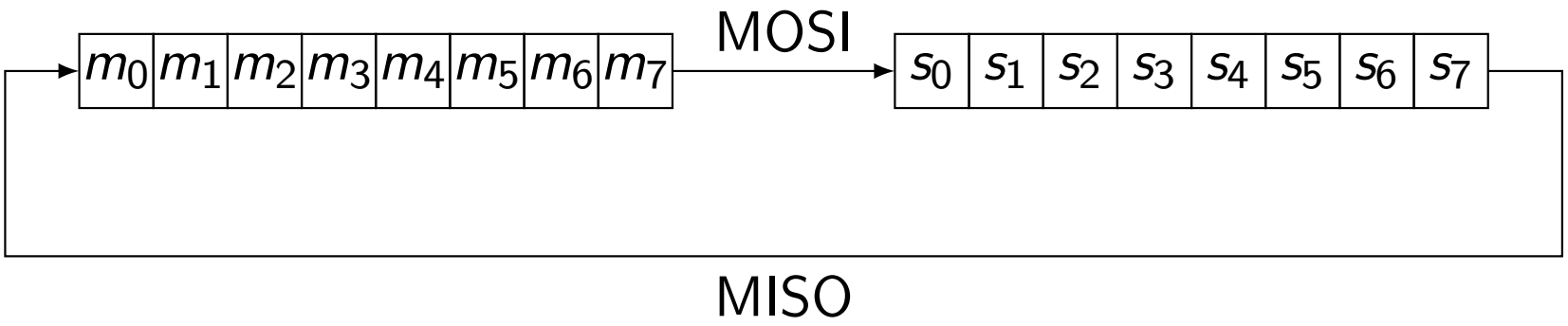
Warstwa fizyczna

- Urządzenia obsługujące protokół SPI są wyposażone w:
  - wewnętrzne rejestry przesuwne – tu umieszcza się przesyłane i odbierane dane,
  - wyprowaddzenia komunikacyjne MISO i MOSI,
  - wejście SS (Slave Select), oznaczane też CS (Chip Select),
  - wyprowadzenie zegara (SCK, CLK).
- Wyprowadzenia MISO, MOSI i SCK są przyłączone do odpowiednich linii.
- Master może wybrać adresata pakietu, ustawiając jego wejście SS w stan niski (za pomocą osobnego połączenia).

Protokół komunikacyjny

- Master inicjuje komunikację, podając na wejście SS wybranego urządzenia slave stan niski.
- Master i slave umieszczają dane do przesłania w swoich rejestrach przesuwnych.
- Master generuje sygnał zegarowy na linii SCK.
- W każdym cyklu zegara:
  - po linii MOSI jest przesyłany jeden bit z rejestru urządzenia master do rejestru urządzenia slave,
  - po linii MISO, jest przesyłany jeden bit z rejestru urządzenia slave do rejestru urządzenia master,
  - zawartość rejestrów jest przesuwana o jeden bit.

Protokół komunikacyjny, cd.



- Po ośmiu cyklach zegara wartość z rejestru urządzenia slave trafia do rejestru urządzenia master i na odwrót.
- Kolejność przesyłania bitów (najpierw LSB lub najpierw MSB) jest konfigurowalna.
- Po przesłaniu dowolnie długiego pakietu master podaje na wejście SS urządzenia slave stan wysoki.

SPI w ATmega16

- Mikrokontroler może zostać skonfigurowany jako master lub jako slave.
- Port MOSI jest wyprowadzony jako PB5, MISO jako PB6, SCK jako PB7, a SS jako PB4.
- Po włączeniu interfejsu SPI wyprowadzenia te są odłączane od portu PORTB mikrokontrolera.
- Uwaga! Do tych samych wyprowadzeń jest podłączony programator ISP.

SPI w trybie master

- Generowanie sygnału zegarowego rozpoczyna się automatycznie po załadowaniu wartości do rejestru SPDR.
- Po przesłaniu 8 bitów mikrokontroler kończy generowanie sygnału zegarowego, ustawiając znacznik SPIF.
- Może to wyzwolić przerwanie o adresie symbolicznym `SPIaddr`.
- Aktywowanie urządzeń slave, czyli podanie stanu niskiego na ich wejścia SS, nie odbywa się automatycznie.

SPI w trybie slave

- Dopóki wejście SS jest w stanie wysokim, wyjście MISO jest w stanie wysokiej rezystancji i urządzenie nie przesyła danych z rejestru SPDR.
- Gdy SS jest w stanie niskim, odbywa się transmisja poszczególnych bitów w kolejnych cyklach zegara.
- Po przesłaniu 8 bitów jest ustawiany znacznik SPIF.
- Może to wyzwolić przerwanie o adresie symbolicznym `SPIaddr`.

Kierunek portów w poszczególnych trybach

- W trybie slave:
  - MISO jest wejściem lub wyjściem, zależnie od ustawienia DDRB.
  - MOSI jest wejściem, niezależnie od ustawienia DDRB,
  - SCK jest wejściem, niezależnie od ustawienia DDRB,
  - SS jest wejściem, niezależnie od ustawienia DDRB.
- W trybie master:
  - MISO jest wejściem, niezależnie od ustawienia DDRB,
  - MOSI jest wejściem lub wyjściem, zależnie od ustawienia DDRB,
  - SCK jest wejściem lub wyjściem, zależnie od ustawienia DDRB,
  - SS jest wejściem lub wyjściem, zależnie od ustawienia DDRB.

Noga SS w trybie master

- Jeśli w trybie master SS skonfigurowano jako wyjście, to pracuje ono niezależnie od modułu SPI (jest przyłączone do PORTB).
- Jeśli w trybie master SS skonfigurowano jako wejście, to pojawienie się na nim stanu niskiego:
  - przerywa rozpoczętą transmisję,
  - przełącza urządzenie w tryb slave,
  - zeruje bit MSTR w rejestrze SPCR,
  - ustawia znacznik SPIF,
  - może wyzwolić przerwanie.

Rejestr SPCR

7	6	5	4	3	2	1	0
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

- bit 7 – uaktywnia przerwania po ustawieniu znacznika SPIF,
- bit 6 – włącza interfejs SPI,
- bit 5 – jeśli ustawiony, to najpierw jest przesyłany LSB,
- bit 4 – jeśli ustawiony, to urządzenie pracuje w trybie master,
- bity 3, 2 – ustalają tryb pracy interfejsu SPI,
- bity 1, 0 – preskaler zegara SPI w trybie master – dzielnik zegara systemowego, odpowiednio 4, 16, 64 lub 128 – szczegóły w dokumentacji mikrokontrolera.

Rejestr SPSR

7	6	5	4	3	2	1	0
SPIF	WCOL	–	–	–	–	–	SPI2X

- bit 7 – ustawiany po zakończonej transmisji lub na skutek pojawienia się stanu niskiego na nodze SS w trybie master; zerowany automatycznie przez sprzęt w trakcie obsługi przerwania lub za pomocą sekwencji: odczyt SPSR, zapis SPDR,
- bit 6 – ustawiany przy próbie zapisu do SPDR, gdy poprzednia transmisja nie zakończyła się,
- bit 0 – gdy ustawiony, podwaja prędkość transmisji.

Rejestr SPDR

- Zawiera dane do przesłania lub dane odebrane.
- Zapis do rejestru inicjuje transmisję danych.
- Nie można zapisać do SPDR przed zakończeniem poprzedniej transmisji.
- SPDR jest podwójnie buforowany przy odczycie – można najpierw wpisać do SPDR nową wartość, a potem odczytać odebraną.
- Wartość z SPDR musi jednak zostać odczytana przed zakończeniem kolejnej transmisji.

Tryby pracy SPI

- Jeśli bit CPOL=0, to SCK jest w stanie niskim, gdy zegar nieaktywny.
- Jeśli bit CPOL=1, to SCK jest w stanie wysokim, gdy zegar nieaktywny.
- Jeśli bit CPHA=0, to dane są przesyłane przy pierwszym (i każdym nieparzystym) zboczu zegara.
- Jeśli bit CPHA=1, to dane są przesyłane przy drugim (i każdym parzystym) zboczu zegara.

tryb	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Przykładowy kod, inicjacja w trybie master

```
SPI_MasterInit:
  ; MOSI i SCK jako wyjścia
  ldi r17, 1<< PB5 | 1 << PB7
  out DDRB, r17

  ; Włączamy SPI w trybie Master, preskaler 16.
  ldi r17, 1 << SPE | 1 << MSTR | 1 << SPR0
  out SPCR, r17
  ret
```

Przykładowy kod, transmisja w trybie master

```
SPI_MasterTransmit:
  ; Rozpoczynamy transmisję danych z r16.
  out SPDR, r16

  ; Czekamy na zakończenie transmisji.
Wait:
  sbis SPSR, SPIF
  rjmp Wait
  ret
```

Przykładowy kod, inicjacja w trybie slave

```
SPI_SlaveInit:
  ; Ustawiamy MISO jako wyjście.
  sbi DDRB, PB6

  ; Włączamy SPI.
  ldi r17, 1 << SPE
  out SPCR, r17
  ret
```

Przykładowy kod, odbiór w trybie slave

```
SPI_SlaveReceive:
  ; Czekamy na koniec transmisji.
  sbis SPSR, SPIF
  rjmp SPI_SlaveReceive

  ; Odczytujemy odebrane dane.
  in r16, SPDR
  ret
```

Pamięć AT45DB041D

- W (większości) zestawów znajduje się pamięć FLASH AT45DB041D.
- Nogi są wyprowadzone w grupie FLASH.
- Pamięć pracuje w trybach 0 i 3 SPI jako slave.
- Pojemność pamięci wynosi 4 megabity.
- Częstotliwość zegara do 66 MHz.
- Szczegóły obsługi pamięci trzeba wyszukać w jej dokumentacji.