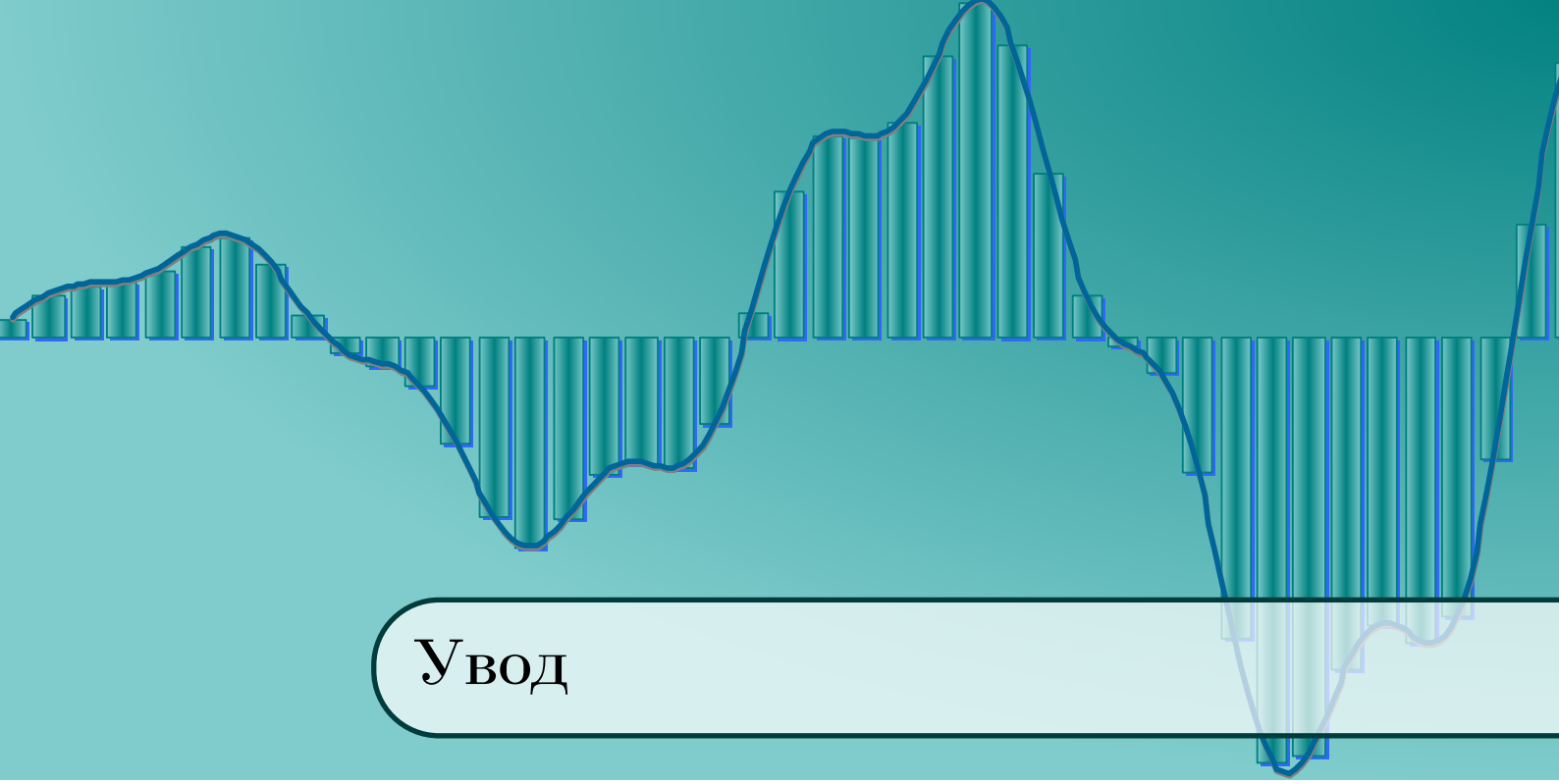
Микропроцесорски системи
– ауторизована предавања –

Срђан Т. Митровић

Младен Антонић

Садржај



Увод

Ауторизована предавања "Микропроцесорски системи"су настала као резултат вишегодишњег педагошког рада аутора у релазицији наставе из истоименог предмета. Ауторизована предавања су намењена како студентима и кадетима[1] који слушају наставу из предмета "Микропроцесорски системи", тако и инжењерима који желе да се подсете или стекну нова знања из области.

Иако познавање садржаја из области дигиталне електронике и основна знања програмирања у програмском језику *C* умногоме олакшавају праћење садржаја књиге, ауторизована предавања се могу користити и као незавнисна литература, јер је садрже и кратак сажетак из области које су неопходне за рад са микропроцесорима и микроконтролерима.

Садржај књиге је више оријентисан на практично разумевање потешкоћа и начина њиховог превазилажења, него на архитектуру и теорију примене микроконтролера. Мада је књига прилагођена за употребу у електронском облику (велики број линкова омогућује брже сналажање читаоцу), њена папирна верзија нема ништа мању употребну вредност.

Предавања су груписана у седам поглавља. Аутор првих шест поглавља је Срђан Митровић, а поглавље о меморији је написао и припремио Младен Антонић.

Основи архитектуре микроконтролера су обрађени у првом поглављу "О микроконтролерима", након чега је представљено коришћење и управљање улазно излазним пиновима микроконтролера, прво за дигиталне а затим и аналогне сигнале. Следе поглавља о прекидима, раду са бројачима и тајмерима, затим су детаљно представљени комуникациони модули и интерфејси који се користе у микроконтролерима. Књига се завршава поглављем о меморијима које су интегрисане у типичне микроконтролере.

Аутори
Београд, 10. новембар 2017. године

[1] Предавања су писана сажето и **прескакање** садржаја при припреми колоквијума и испита крајње **је непрепоручљиво** .

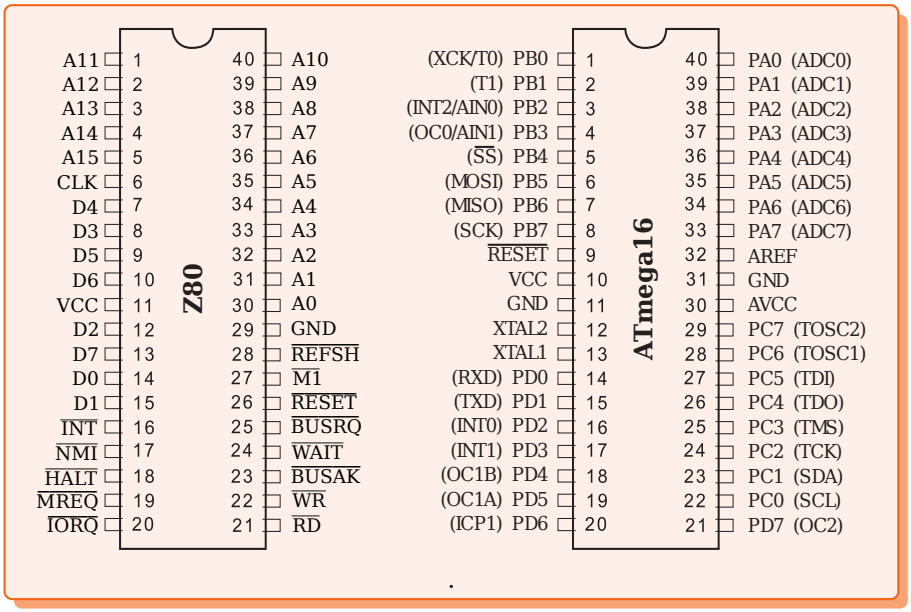
1 О микроконтролерима

У овом делу ћемо покушати да одговоримо на следећа питања:

1. Шта је микроконтролер?
2. Која је разлика између микроконтролера и микропроцесора?
3. Чему микроконтролер служи?

1.1 Микроконтролер-микропроцесор

Претпоставићемо да нам је појам микропроцесора познат, и погледати слику 1.1 на којој су приказани распореди пинова једног микропроцесора и једног микроконтролера. Број пинова је исти код обе компоненте, а у њиховим ознакама се крије



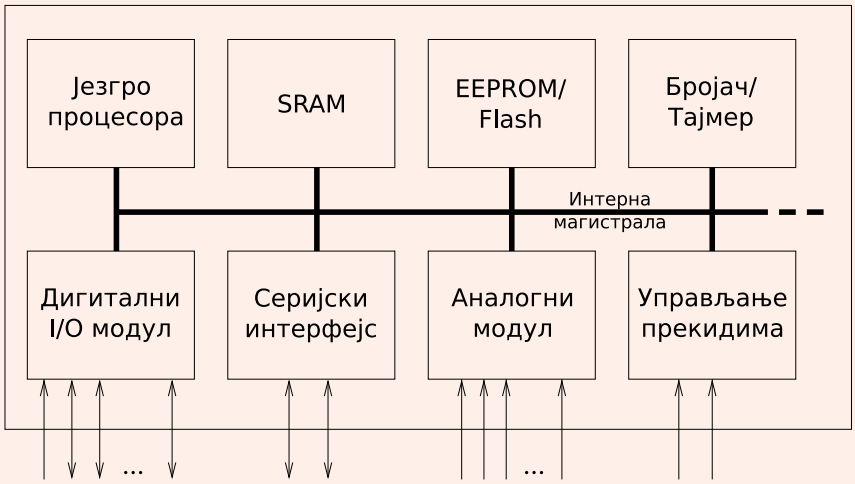
Слика 1.1: Распоред пинова микропроцесора Z80 (лево) и микроконтролера ATmega16 (десно)

прва битна разлика. Примећујемо да многи пинови микроконтролера имају по две ознаке, за разлику од микропроцесора, код кога је сваки пин означен тачно једном. То у ствари значи да су сви пинови микропроцесора једнозначно одређени, док се пиновима микроконтролера може додељивати намена из предефинисаног скупа. На

пример пин 40 микропроцесора Z80 има ознаку `A10`, што значи да тај пин припада адресној магистрали, док код Atmega16 стоји `PA0`(ADC0). Ознака `PA0` значи да овај пин припада порту `A`, при чему се може дефинисати као улазни или излазни пин [1], док `ADC0` означава да је овај пин могуће користити као аналогни улаз за аналогно дигитални конвертор. За разлику од микропроцесора, пинови микроконтролера се могу користити на више различитих начина, у наведеном примеру пин 40 може имати троструку намену:

1. дигитални улаз
2. дигитални излаз
3. аналогни улаз

Поменули смо аналогно-дигитални (АД) конвертор, што је још једна разлика у односу на микропроцесор. Микроконтролери осим процесора садрже и друге компоненте (слика 1.2) као што су: различите меморије, бројачи, тајмери (поглавље 5), дигитални улазни и излазни модули (поглавље 2), комуникациони модули (поглавље 6), механизам за управљање прекидима (поглавље 4), аналогни интерфејс (поглавље 3) итд.



Језгро процесора – *Processor Core*, садржи аритметичко логичку јединицу, управљачку јединицу, регистре, ...

SRAM и EEPROM су типови меморија. Меморије служе за складиштење података и програма.

Бројач/Тајмер – *Counter/Timer* служе за бројање догађаја, мерење временских интервала, имулсно ширинску модулацију ...

Дигитални I/O модул је дигитална улазно излазна јединица, чији се број пинова може кретати у распону од 3-4 па до 90

Серијски интерфејс садржи комуникационе модуле као што су: SPI, SCI, I2C, CAN, PCI, USB, Ethernet

Аналогни модул је аналогна улазно излазна јединица, и карактеристика је већих микроконтролера (АД конвертор, ДА конвертор, ...)

Управљање прекидима – *Interrupt Controller*. Нормалан ток програма се прекида у случају важних спољашњих или генерисаних догађаја

Слика 1.2: Основне компоненте микрокотролера

.

Микроконтролер-микропроцесорски ситем – појам

То је микро процесор (огољен), који је опремљен: меморијом, тајмерима, I/O пиновима (паралелно) и другим уграђеним периферијама.

1.2 Практична примена микропроцесорских система

Због чега се у неким апликацијама користе микроконтролери као интегрисани микропроцесорски систем, уместо микропроцесора са додатим периферијама према захтевима те апликације? Разлози су вишеструки, а први је цена, јер интегрисање свих елемената у један чип умањује цену, како производње, тако и пројектовања. Други разлози су:

- могућност надоградње система,
- мања потрошња електричне енергије,
- већа поузданост,
- могућност програмирања, ...

Области у којима се микроконтролери користе су многобројне, навешћемо неке:

- Војска (борбени системи),
- телекомуникације,
- управљање - аутоматика,
- мерни инструменти,
- медицина,
- производи широке потрошње (веш машине, електрични шпорети, машине за прање суђа, телевизори, ...)

На тржишту су доступни микроконтролери различитих произвођача. Проблем одабира одговарајућег контролера се усложњава када имамо у виду да сваки од произвођача нуди више типова микроконтролера (табела 1.1).

Табела 1.1: Део понуде из скупа од 218 осмобитних AVR микроконтрлера [2].

Controller	Flash (KB)	SRAM (Byte)	EEPROM (Byte)	I/O-Pins	A/D (Channels)	Interfaces
AT90C8534	8	288	512	7	8	
AT90LS2323	2	128	128	3		
AT90LS2343	2	160	128	5		
AT90LS8535	8	512	512	32	8	UART, SPI
AT90S1200	1	64		15		
AT90S2313	2	160	128	15		
ATmega128	128	4096	4096	53	8	JTAG, SPI, IIC
ATmega162	16	1024	512	35		JTAG, SPI
ATmega169	16	1024	512	53	8	JTAG, SPI, IIC
ATmega16	16	1024	512	32	8	JTAG, SPI, IIC
ATtiny11	1		64	5+1 In		
ATtiny12	1		64	6		SPI
ATtiny15L	1		64	6	4	SPI
ATtiny26	2	128	128		16	SPI
ATtiny28L	2	128		11+8 In		

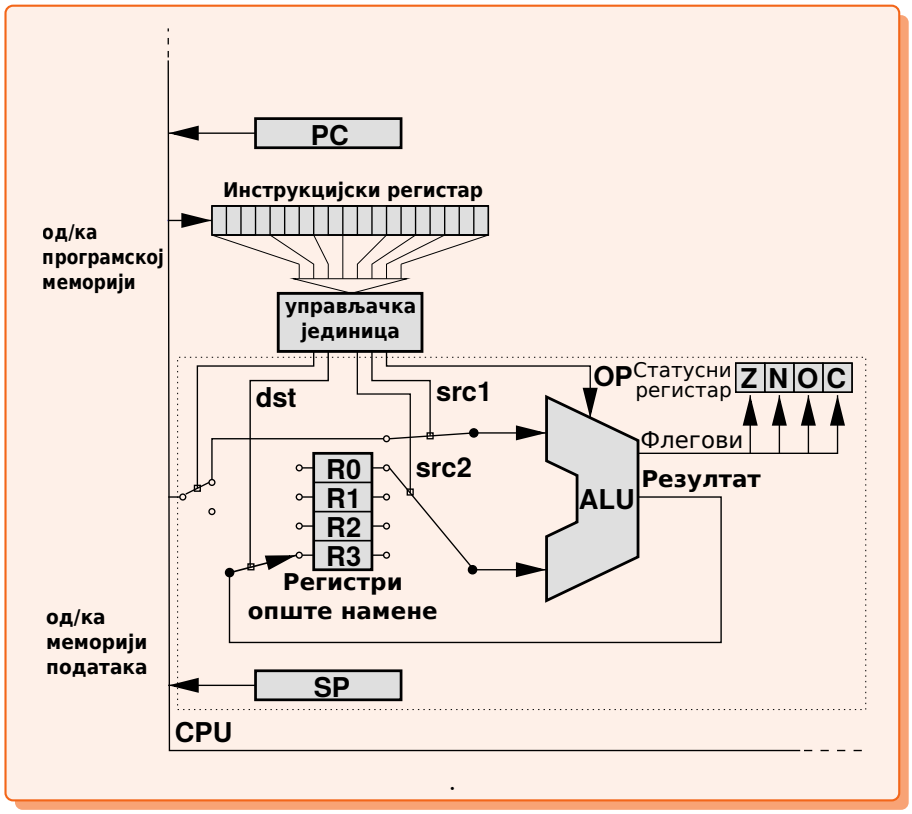
Дефинисање једнозначних правила како одабрати микроконтролер за апликацију није једноставно, али би требало имати у виду следеће:

1. Бирамо микроконтролере произвођача са којим већ имамо искуство.
2. Бирамо микроконтролере за које имамо развојно окружење.

3. Размишљамо о надоградњи система и пре његовог завршетка, не бирамо микроконтролер чије ћемо ресурсе искористити максимално, остављамо резерву за надоградњу.
4. Потрошња енергије може утицати на избор.

1.3 Језгро процесора

Подразумева се да је читаоц већ упознат са основном архитектуром процесора, те ово поглавље није намењено као помоћ у детаљном изучавању архитектуре микропроцесора, него уочавању специфичности микропроцесора у микропроцесорским системима–микроконтролерима. На слици 1.3 је приказана основна архитектура централне процесорске јединице (енгл. CPU - central processor unit) једног микропроцесора.



Слика 1.3: Основна архитектура микропроцесора

1.3.1 Аритметичко логичка јединица - Arithmetic Logic Unit–ALU

Као што се може закључити из назива ALU је намењена за извршавање аритметичких и логичких операција. У зависности од операције ALU узима један или оба операнда и генерише резултат. Операнди се могу учитати из регистра или меморије, а резултат се такође може ускладиштити у регистар или у меморију. Управљачка јединица (енгл. Control Unit) одређује ток података - одакле стижу операнди и где се смешта резултат, као и која се операција извршава. Осим резултата ALU генерише и друге информације о извршеној операцији, тј. резултату и уписује их у статус регистар (енгл. status register, види поглавље 1.3.2).

Представљање негативних бројева

Рачунар за представљање негативних бројева користи 0 и 1. Како представити негативан цео број?

Идеја 1: Инвертовати све битове позитивног броја. Ово се зове комплемент јединице.

Недостатак Постоје две интерпретације нуле.

Идеја 2: Инвертовати све битове позитивног броја и додати 1. Ово се зове комплемент двојке – други комплемент.

Пример

За четворобитну репрезентацију се добија:

$$1 = 0001 \rightarrow -1 = 1110 + 1 = 1111$$

$$0 = 0000 \rightarrow -0 = 1111 + 1 = 0000$$

1.3.2 Регистри

Registar File садржи радне регистре CPU. Регистри могу бити опште намене (енгл. general purpose) у које се смештају операнди или резултат. Постоје и специјални регистри (енгл. dedicated register):

Accumulator - акумулатор. Користи се за логичке и аритметичке операције.
Index register - индексни регистар. Користи се за неке модове адресирања.
Status register - статусни регистар. Детаљније је описан у даљем тексту.
. . .

Статусни регистар

Статусни регистар чине такозвани флегови, који су у ствари једнобитне информације. Најчешће коришћени су:

Z (Zero) Сетује се ако је резултат операције нула.
N (Negative) Резултат операције је негативан.
O (Overflow) Операција је генерисала прекорачење.
C (Carry) У операцији постоји пренос.

Статусни регистар је регистар попут осталих и очекивано је да се код осмобитних контролера искористи свих осам битова, као што је то случај са AVR микроконтролерима, чији је статусни регистар приказан у табели 1.2.

Значење ових флегова је следеће:

Bit 7 I - Global Interrupt Enable. Сетовањем овог флега (бита) микроконтролеру се омогућује регистровање и реаговање на прекиде (детаљније описано у поглављу 4).

Bit 6 T - Bit Copy Storage. Инстуркције за копирање бита BLD (Bit LoaD) и BST (Bit STore) користе T-бит, наиме бит из једног регистра се може копирати у T-бит помоћу инструкције BST, а вредност T-бита се може учитати инструкцијом BLD.

Bit 5 H - Half Carry Flag Означава да се десио "Half Carry"(полу-пренос) и користи се у BCD аритметици.

Bit 4 S - Sign Bit. Он увек представља резултат логичке операције ексклузивно

Bit	7	6	5	4	3	2	1	0
Flag	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
Initial Value	0	0	0	0	0	0	0	0

Бит	7	6	5	4	3	2	1	0
Флег	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
Почетна вредност	0	0	0	0	0	0	0	0

Табела 1.2: Статусни регистар AVR микроконтролера–SREG

или између N и V флегова, тј. $S = N \oplus V$.

Bit 3 V - Two's Complement Overflow Flag Прекорачење у другом комплементу (комплементу двојке). Посебан флег за прекорачење када се користи аритметика у комплементу двојке.

Bit 2 N - Negative Flag. Резултат операције је негативан.

Bit 1 Z - Zero Flag. Сетује се ако је резултат операције нула.

Bit 0 C - Carry Flag. У операцији постоји пренос.

Као што се може видети из табеле 1.2, почетна вредност свих флегова (Initial Value) у статусном регистру SREG је једнака нули. Статусни регистар се може користити када се сабирају или одузимају бројеви чија је дужина већа од процесорске речи, те CPU нуди инструкције које на пример могу да користе *carry flag*, као што је ADDC – сабери са преносом.

Пример употребе инструкције `ADDC`: `0x01E0+0x0230`

```
CLC             ;[a] брише carry флег
LD R0,   #0xE0 ;  учитава први нижи бајт у  R0
ADDC R0, #0x30 ;  сабира други нижи бајт са преносом (carry <- 1)
LD R1,   #0x01 ;  учитава први виши бајт у R1
ADDC R1, #0x02 ;  сабира други виши бајт, пренос
               ;  из претходног ADDC је додат [b]
```

[a] коментар се у асемблеру означава ;

[b] када би користили наредбу `ADD` уместо `ADDC`, уместо тачног резултата `0x0410` добили би резултат `0x0310`.

1.3.3 Стек – Stack

Стек (*stack*) је део меморије у простору података који CPU користи за смештај адресе повратка и евентуално садржаја регистара у случају:

- позива процедуре или
- дешавања прекида.

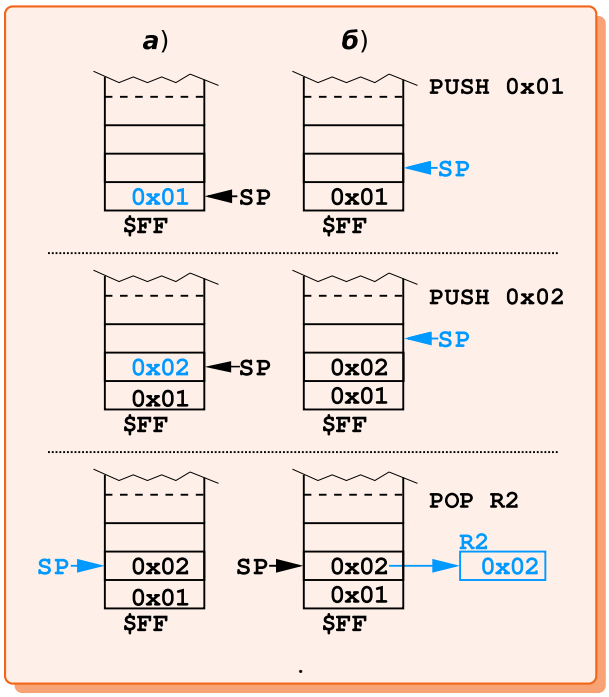
Стек може користити и програмер. Команде за приступ стеку су:

- `PUSH` (стави нешто у стек) и
- `POP` (узми нешто из стека).

О томе са које меморијске локације у стеку се узима, или где се складишти податак, говори нам вредност показивача који се назива Stack Pointer (SP)–показивач стека.

SP може на два начина показивати на меморисјку локацију у стеку (слика 1.4):

- прва слободна адреса (Atmel AVR) или
- последња коришћена адреса.



Слика 1.4: Показивач стека [3]
а) Motorola HCS12 SP
б) Atmel AVR SP

Микроконтролери Atmel AVR прво ускладиште податак а затим декрементирају показивач стека, док се на пример код микроконтролера HCS12 наредбом PUSH прво декрементује показивач стека, након чега се податак складишти на прву слободну адресу.

Напомена

SP иницијализује програмер, морате се упознати са начином на који га ваш микроконтролер користи.

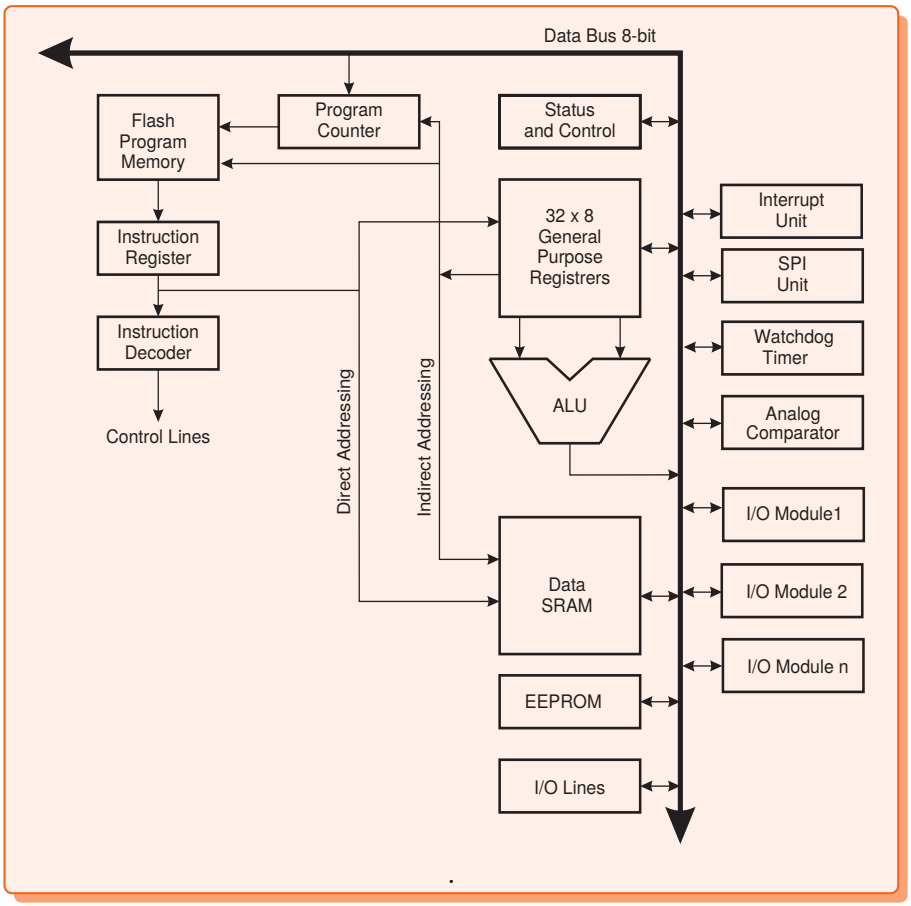
1.3.4 Управљачка јединица – Control Unit

Централна процесорска јединица непрекидно извршава инструкције (осим при ресету и HALT). Задатак CPU је да одреди која се следећа инструкција извршава и да у зависности од тога обезбеди одговарајући проток података. Адреса наредне инструкције се чува у специјалном регистру познатом као Program Counter-PC[1]. Управљачка јединица учитава инструкцију у регистар инструкција (енгл. Instruction Register-IR), декодира је и поставља ток података са циљем њеног извршавања, што значи да се за ALU обезбеђују одговарајући улази, операција која се извршава, као и локација за складиштење података. Након извршавања инструкције вредност

[1]Слободан превод на спрски језик би био бројач програма, али није уобичајено да се ова кованица преводи.

PC се инкрементира на адресу следеће инструкције у програму, или се учитава нова адреса у случају гранања програма или позива процедуре. Уобичајено је да PC након ресета показује на адресу $0000.

На почетку овог подпоглавља је илустрована основна архитектура микропроцесора, (слика 1.3), а на слици 1.5 је дата основна архитектура конкретног микропроцесора, преузета из документације произвођача [1].



Слика 1.5: Основна архитектура AVR микропроцесора [1]

1.4 Скуп инструкција

Основно обележје сваког CPU представљају број и врсте расположивих инструкција. Инструкције утичу на величину кода, што значи и на меморисјки простор неопходан за његово складиштење. При избору микроконтролера увек треба имати у виду компатибилност његових инструкција са потребама конкретне апликације. На избор могу утицати следећи чиниоци:

- величина инструкција,
- брзина извршавања,
- расположивост инструкција и
- расположиви модови адресирања.

1.4.1 Величина инструкција

Opcode дефинише операцију која треба да се изврши и њене операнде. Већи *opcode* процесора не подразумева нужно да програм заузима више меморије

10 линија кôда ATmega16 RISC процесора захтева 20 бајтова (свака инструкција се декодује у 16 битова). 10 линија кôда 68030 процесора захтева 4 бајта .

Архитектуре према врсти инструкција можемо сврстати у следеће групе:

Stack Architecture: У овој архитектури, познатој и као *0-address format architecture*, експлицитно не постоје операнди. Они су организовани у стеку.

Accumulator Architecture: (*1-address format architecture*), један операнд се увек налази у акумулатору, где се смешта и резултат.

2-address Format Architecture: Оба операнда су задата, али се резултат смешта на локацију једног од њих (ATmega16 користи први регистар)

3-address Format Architecture: Локације операнада и резултата се експлицитно задају. Ово је најфлексибилнија архитектура, али су и инструкције највеће (тј. њихов opcode).

У табели 1.3 је у асемблеру представљено решење једноставног проблема: $(A + B) \cdot C$ за све наведене типове адресних архитектура. Осим што се решења разликују по броју операнада у инструкцијама, најважнија разлика је у томе где се налази резултат након извршавања овог дела програма, што је наглашено у последњем реду табеле.

Табела 1.3: Поређење адресних архитектура кроз пример: $(A + B) \cdot C$

стек	акумулатор	2-адресно	3-адресно
`PUSH A`	`LD A`	`LD R1,A`	`LD R1,A`
`PUSH B`	`ADD B`	`LD R2,B`	`LD R2,B`
`ADD`	`MUL C`	`ADD R1,R2`	`ADD R1,R1,R2`
`PUSH C`		`LD R2,C`	`LD R2,C`
`MUL`		`MUL R1,R2`	`MUL R2,R1,R2`
стек	акумулатор	R1	R2

1.4.2 Брзина извршавања

На брзину извршавања инструкција утиче више чинилаца:

- сложеност архитектуре,
- величина речи и
- фреквенција осцилатора.

Према сложености архитектуре у односу на инструкције микропроцесоре најчешће можемо сврстати у једну од две групе: CISC и RISC.

RISC – *Reduced Instruction Set Computer*. RICS архитектура располаже једноставним инструкцијама, чије извршавање по правилу траје свега неколико циклуса такта. RICS процесоре одликује мали и фиксан код, са релативно малим бројем инструкција и модова адресирања, али се инструкције извршавају веома брзо, мада је скуп расположивих интрукција релативно мали [3].

CISC – *Complex Instruction Set Computer*. CISC архитектуру одликују инструкције чији је микрокод знатно сложенији, а извршавања захтева релативно

велики број циклуса такта. CISC архитектура поседује велики број различитих инструкција и адресних модова [3].

Поређење архитектура кроз пример

Размотримо сложени CISC адресни мод упоредо са његовом имплементацијом на RISC архитектури[a]. Моторолин процесор 68030 нуди мод "меморијско индиректно адресирање са преиндексним скалирањем"[b]:

```
MOVE D1, ([24,A0,4*D0])
```

Ова операцији складишти садржај регистра D1 у меморију на адресу:

```
24 + [A0] + 4 * [D0]
```

где средња заграда означава садржај регистра или меморисјку локацију. Да би се овакво адресирање користило на RISC процесору произвођача Atmel, неопходан је следећи код:

```
;-----------------------------------------------------------
LD   R1, X   ; индиректно учитава податак (из [X] у R1)
LSL  R1      ; померање улево -> множење са 2
LSL  R1      ; урађено је 4*[D0]
MOV  X,  R0  ; постављање показивача (учитавање A0)
LD   R0, X   ; индиректно учитавање (завршено [A0] )
ADD  R0, R1  ; сабирање показивача ([A0]+4*[D0])
LDI  R1, $24 ; учитавање константе ($ = hex)
ADD  R0, R1  ; и сабирање (24+[A0]+4*[D0])
MOV  X,  R0  ; постављање показивача за упис
ST   X,  R2  ; уписивање вредности ([24+[A0]+4*[D0]] <- R2)
;-----------------------------------------------------------
```

У овом коду се продразумева да је R0 коришћено као A0, X замењује D0, а R2 садржи вредност D1. Иако RISC архитектура затева 10 линија кода у поређењу са једном код 68030, она није спорија. Ова инструкција процесора 68030 се извршава у 14 циклкуса такта, док се кôд за RISC процесор извршава у 13 циклуса такта.

[a] Пример је преузет из [3], преведен и прилагођен.
[b] memory indirect preindexed

Поређење аритектура: CISC–RISC

CISC

Complex Instruction Set Computer

- сложене микроинструкције
- велики број наредби
- дуже време извршавања инструкције

RISC

Reduced Instruction Set Computer

- једноставна архитектура
- редукован број наредби
- краће време извршавања инструкције

1.4.3 Расположивост инструкција

Уобичајено је да се инструкције деле на следеће класе:

Аритметичко-логичке [2]: ADD, SUB, MUL, ... AND, OR, XOR, ... BSET (*сетуј бит*), BCLR (*обриши бит*), и BTST (*да ли је бит сетован*).

Управљање подацима [3]: LD (*load*) и ST (*store*), PUSH и POP.

Регулисање тока програма [4]: скокови, условна гранања, позиви процедура, повратак из процедуре или прекида: BNE, RET или RETI, ...

Управљачке инструкције [5]: утичу на рад контролера: NOP (не ради ништа), ресет, *sleep*, управљање дебаговањем, ...

Размотримо инструкције за померање битова у регистру. Померање битова се може извести:

- у леву страну или
- у десну страну.

Инструкције за померање битова могу бити и аритметичке и логичке операције. Разлика је у третирању бита највеће тежине–MSB (*Most Significant Bit*) када се померање изводи у десну страну, што представља дељење бројем 2. За померање у леву страну, тј. множење бројем 2, MSB нема значај.

Померање у лево:

4-битни контролер, комплемент двојке[a]
број -3 је бинарно 1101
померањем на лево се добија $1010 = (-6)_{10}$

[a] Објашњење представљања негативних бројева у другом комплементу погледати на страни 34, табела 3.1.

Померање у десно:

број $-4 = 1100$
аритметичким померањем на десно се добија $1110 = (-2)_{10}$ (уз сетован MSB), док логичко померање на десно даје: $0110 = (6)_{10}$.

1.4.4 Адресни модови

При употреби аритмечких операција мора се експлицитно нагласити где се операнди налазе! Операнди могу бити константе, садржај регистра или садржај меморијске локације. Према начину како се операнд прослеђује до аритметичко логичке јединице, разликујемо следеће адресне модове, тј. начине адресирања.

1. **Непосредно** – Immediate/literal. Код овог типа адресирања операнд је константа. Код миктроконтролера Atmega16 ова константа се може унети асемблерском наредбом LDI (LoaD Immediate), али и је и програмер може директно дефинисати помоћу префикса # (префикс # означава константу).
2. **Регистарско** – Register. Адреса операнда се налази у регистру, или се у регистар смешта резултат операције.
3. **Директно** – Direct/Absolute. Операнд се налази на меморијској локацији која је директно уписана.

[2] Arithmetic-Logic Instructions
[3] Data Transfer
[4] Program Flow
[5] Control Instructions

4. **Регистарско индиректно** – Register Indirect. У регистру се налази адреса меморијске локације операнда.
5. **Са аутоинкрементирањем** – Autoincrement. Варијанта индиректног адресирања са инкрементирањем вредности регистра.
6. **Са аутодекрементирањем** – Autodecrement. Варијанта индиректног адресирања са декрементирањем вредности регистра.
7. **Са померајем** – Displacement/based. Задају се константа и регистар: константа + регистар → меморијска локација операнда.
8. **Индексирано** – Indexed. Слично претходном моду, али се уместо константе и регистра задају два регистра, а меморијска локација операнда је резултат њиховог збира: R0 + R1 → меморијска локација операнда. Овај мод је погодан за адресирање низова, где на пример регистар R0 указује на први елемент у низу, а R1 на n-ти елемент.
9. **Меморијско индиректно** – Memory indirect. Задаје се регистар у коме се налази адреса меморијске локације која садржи адресу операнда: регистар → мем. лок. са адресом → мем. лок. операнда.

У табели 1.4 су наведени примери употребе наведених адресних модова у двоадресној архитектури за операцију сабирања-`ADD`. Приметимо да се резултат операције уписује у регистар `R1`.

Табела 1.4: Поређење адресних модова у асемблеру на примеру операције сабирања – ADD

адресирање	**наредба**	**резултат извршавања**
непосредно	`ADD R1,#4`	`R1 ← R1 + 4`
регистарско	`ADD R1,R2`	`R1 ← R1 + R2`
директно	`ADD R1,100`	`R1 ← R1 + M[100]`
регистарско индиректно	`ADD R1,(R2)`	`R1 ← R1 + M[R2]`
пост-инкрементирање	`ADD R1,(R2)+`	`R1 ← R1 + M[R2]`
		`R2 ← R2 + d`
пре-декреметирање	`ADD R1,-(R2)`	`R2 ← R2 - d`
		`R1 ← R1 + M[R2]`
са померајем	`ADD R1,100(R2)`	`R1 ← R1 + M[100+R2]`
индексирано	`ADD R1,(R2+R3)`	`R1 ← R1 + M[R2+R3]`
меморисјко индиректно	`ADD R1,@(R2)`	`R1 ← R1 + M[M[R2]]`

d одговара величини једног елемента

2 Дигитални улази и излази

Управљање и надгледање хардвера је најважнија особина микроконтролера, и због тога практично сви микроконтролери имају најмање $1-2$ (уобичајено $8-32$) улазна или излазна пина који се могу повезати са хардвером. Дигитални улазни (енг. *Input*–I) и излазни (*Output*–O) пинови микроконтролера се могу директно повезати са спољашњим хардвером, уз уважавање ограничења контролера.

Чему служе?
Омогућавају управљање хардвером.
Омогућавају надгледање *(monitoring)* хардвера.

Да ли их имају сви микроконтролери?
ДА! 1–2 *I/O* пина најмање.
Уобичајено 8–32 пина.
Motorola HCS12 преко 90 I/O пинова.

Улазно излазни (I/O) пинови се најчешће групишу у портове од по 8 пинова, којима се може приступити као једном бајту. Пинови могу бити само улазни, само излазни, али је уобичајено да буду двосмерни (бидирекционални), што значи да могу бити и улазни и излазни. Често пинови могу имати и додатне функције, а све са циљем одржавања малог броја пинова и компактног дизајна микроконтролера. Остали модули микроконтролера који користе I/O пинове (нпр. тајмери или аналогни модули), у ствари користе њихове додатне (алтернативне) функције, тако да је задатак програмера да одабере како ће се неки пин користити, али и да постави одговарајућа подешавања у складу са тим избором.

Како су организовани?
- Најчешће као портови са 8 пинова. Због чега баш 8?
- Могу бити улазни или излазни.
- Најчешће су бидирекционални (двосмерни) или вишенаменски.

Ако неки пин користимо као део неког аналогног модула, он се у тој апликацији не може користи као дигитални I/O, и обратно.

2.1 Управљање пиновима

Три регистра управљају пиновима за дигиталне улазе/излазе:

- DDR – *Data Direction Register*
- PORT – *Port Register* и
- PIN – *Port Input Register*.

DDR – *Data Direction Register*. Сваки бидирекционални порт има свој DDR регистар са по једним битом за сваки пин порта. Намена пина (улаз или излаз) је одређена вредношћу бита у регистру. Пинови једног порта се могу конфигурисати различито, као на пример, да 2 буду улазна, а осталих 6 излазни. Уобичајено да се после ресета пинови иницијализују као улазни. Читањем DDR регистра се може установити како су пинови постављени, тј. који су улазни, а који излазни.

PORT – *Port Register*. Овим регистром се управља напонским нивоом на пину. Ако је порт конфигурисан као излазни, сетовање битова (bitset) у PORT регистру значи њихово постављање на стање „high“, док брисање бита (bitclear) значи његово постављање на стање „low“. При промени вредности једног бита, треба обратити пажњу да се не промене остали битови регистра PORT, па се препоручује употреба наредби за операције над једним битом. Уколико се користе операције над целим бајтом, мора се поштовати такозвани *read-modify-write* (учитај-узмени-упиши) приступ, при чему се његово извршавање не сме прекинути. Када су пинови постављени као излазни, читањем регистра PORT добијамо вредности које смо поставили на излаз. Уколико су пинови постављени као улазни, њихова функционалност зависи од типа микроконтролера. Неки контролери омогућавају читање вредности улазних пинова помоћу PORT регистра, док на пример ATmega16 користи PORT регистар у друге сврхе ако су пинови сетовани као улазни, што намеће потребу за упознавањем са особинама микроконтролера на основу документације произвођача.

Read-Modify-Write приступ

- Обратити пажњу код мултитаскинг система!
- Један процес не сме приступити регистру који користи други процес!
- Уколико желимо изменити само један бит у регистру, остали битови се не смеју мењати!
- Користити инструкције за операцију над битом!

PIN – *Port Input Register*. Информоција о напонском нивоу („high“ или „low“) свих пинова једног порта је садржана у његовом PIN регистру. Најчешће је омогућено само читање овог регистра (*read-only*) и користи се као провера стања улазних или излазних пинова.

Atmega16 – конфигурација пинова једног порта [1]

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

2.2 Дигитални улази

Дигитални улази се користе за надгледање сигнала који може узети две вредности, тј. има само два стања: ниско (логичка нула) и високо (логичка јединица).

Када се користе?

Када улазни сигнал може имати само две вредности.

Наравно, у једном тренутку времена. Сигнал се у општем случају мења са временом.

Логички нивои*:

- логичка јединица – *"high"*
- логичка нула – *"low"*

*Позитивна логика.

Од напонског нивоа сигнала у тренутку његовог очитавања зависи да ли ће очитана вредност бити протумачена као нула или као јединица, што зависи од контролера и његовог напона напајања, који се означава са V_{CC}. На пример, напајање контролера ATmega16 мора бити у границама $[4.5, 5.5]$ V и напонски ниво логичке нуле је $[-0.5, 0.2V_{CC}]$ V, а логичке јединице $[0.6V_{CC}, V_{CC} + 0.5]$ V. Када се напон улазног сигнала нађе у опсегу $(0.2V_{CC}, 0.6V_{CC})$, његова бинарна вредност није дефинисана.

Atmega16

- $V_{CC} = [4.5, 5.5]$V
- 1: $[0.6V_{CC}, V_{CC} + 0.5]$ V
- 0: $[-0.5, 0.2V_{CC}]$ V
- интервал $(0.2V_{CC}, 0.6V_{CC})$ није дефинисан

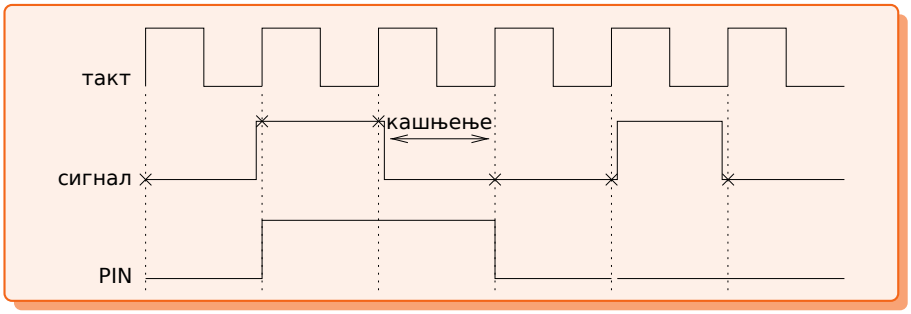
2.2.1 Узорковање сигнала

Једноставно решење

- "лечујемо" вредност PIN регистра
- са сваким доласком такта читамо "леч"
- промене читамо са закашњењем
- **спора промена сигнала?** (у односу на фреквенцију такта)

Пошто је реалан сигнал представљен само својим напонским нивоом, поставља се питање како се тај напон може трансформисати у бинарну вредност у регистру. Најједноставније решење је читање вредности PIN регистра преко његовог кола задршке–лече (енгл. *latch*). Ако се коло задршке окида системским тактом, читаћемо тренутно стање на почетку сваког циклуса такта, што значи да је учестаност одабирања улазног сигнала једнака фреквенцији системског такта. Промене улазног сигнала које су краће од периоде такта могу бити пропуштене, слика 2.1.

Кашњење узоркованог сигнала у односи на оригинални–d_{latch} се може наћи у опсегу $d_{latch} \in (0, 1]$ циклуса такта. Нула је изостављена из опсега јер није дефинисано шта ће се догодити ако се промена стања сигнала деси истовремено са променом такта. Одавде следи да импулси садржани у улазном сигналу морају трајати дуже



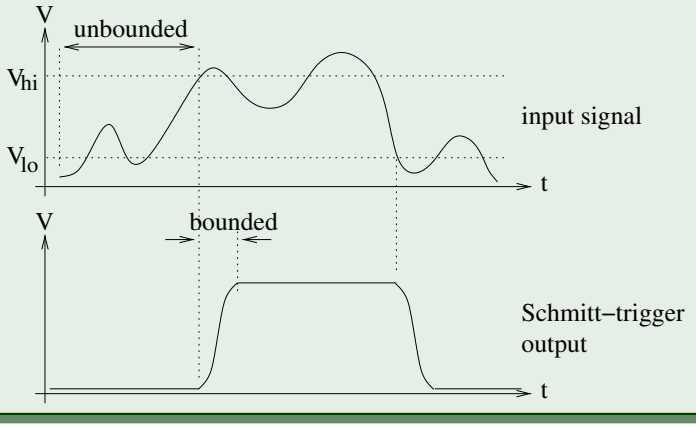
Слика 2.1: Илустрација кашњења (најнеповољнији случај) и пропуштеног имулса када се сигнал узоркује директно са PIN регистра

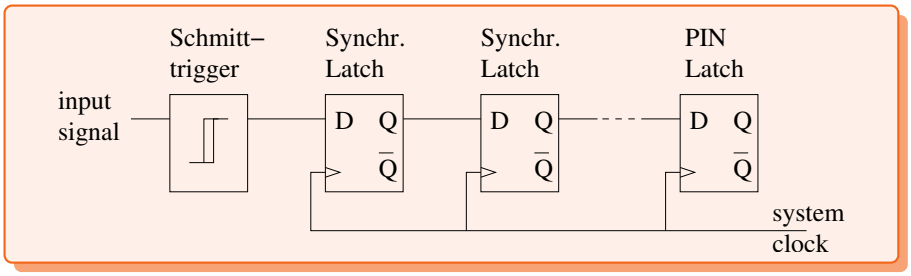
од једне периоде системског такта да би са сигурношћу били препознати (учитани). Опсег кашњења улаза означићемо са $\bar{d}_in \in (d_{in}^{min}, d_{in}^{max}]$, где је d_{in}^{min} доња граница–минимално кашњење улаза, а d_{in}^{max} је горња граница - максимално могуће кашњење улаза.

Ова техника је наизглед врло примамљива, јер користи основне могућности микроконтролера, али је непогодна за ситуацију која се често може срести у реалној примени када су промене стања улазног сигнала споре у односу на фреквенцију такта. Сигнал генерише хардвер и није могуће са сигурношћу гарантовати да ће се промене стања сигнала одвијати брзо. Спора промена стања сигнала води ка проблему који је познат као **метастабилност** . Једно коло задршке које одабира недефинисан напонски ниво улазног сигнала има вероватноћу p да уђе и остане у метастабилном стању, у коме улаз може бити протумачен као низак, као висок, недефинисан или чак може и осциловати. Последње две опције су крајње непожељне, како за микроконтролер тако и за апликацију и морају се избећи.

Смањење вероватноће уласка у метастабилност се може постићи коришћењем Шмитовог кола (енгл. *Schmitt–trigger*) које решава проблем трајања узлазних и силазних ивица сигнала. Постављањем једног или више лечева између Шмитовог кола и леча PIN регистра може се додатно смањити вероватноћа појаве метастабилног стања p. Оваква конструкција се назива синхронизатор (енгл. *synchronizer*) и приказана је на слици 2.2. Ако са k означимо број додатних кола задршке (лечева) вероватноћа појаве метастабилног стања је p^k, што је мање или једнако p.

Шмитово коло–принцип рада





Слика 2.2: Смањење појаве метастабилности коришћењем два кола задршке

Метастабилност (*meta-stability*)

- Вероватноћа одабирања недефинисаног стања је p по лечу
- Коришћењем k лечева смањује се на $p^k \leq p$
- **Недостатак?** Време пропагације сигнала (кашњење)
- Пример ATmega16: $d_{in}^{min} = 0.5clk$, $d_{in}^{max} = 1.5clk$

2.2.2 Избегавање шума

У случајевима када је улазни сигнал јако зашумљен или долази до појаве електромагнетних сметњи (укључење или искључење неког потрошача веће снаге) може се десити да стање PIN регистра не одговара стању реалног сигнала.

Неки контролери имају уграђен механизам за избегавање шума ($\neq$ филтрирање). Уколико је ова опција укључена, контролер више пута очитава вредност, нпр. k пута, и прихвата нову вредност ако је свих k одбирака идентично. **Недостатак** је утицај на укупно кашњење које је сада бар k пута веће.

2.2.3 Заштитни отпорници (Pull Resistors)

Чему служе?

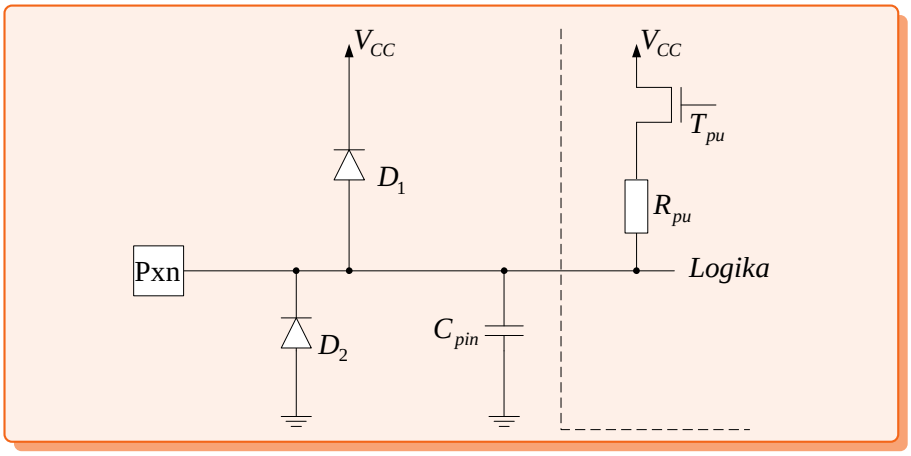
Пинови контролера не смеју да „висе“, јер то узрокује шум („у великим количинама“).

Многи микроконтролери имају интегрисане заштитне отпорнике на својим улазним пиновима. Њихов задатак је да повежу улазни пин на дефинисани напон када њиме не управља спољашњи хардвер. На пример, када је улазни пин повезан механичким прекидачем са спољашњим хардвером, напон на пину је дефинисан када је прикидач затворен, али у случају када је прекидач искључен напон на пину је недефинисан (пин „виси“).

У контролерима се користе на два начина, као:

- pull-up (R_{pu} на слици 2.3),
- pull-down

Pull-up отпорник повезују пин на високи ниво – напајање V_{CC} и чешће се среће у микроконтролерима, док pull-down отпорник повезује пин на ниски ниви –GND. Овим отпорницима се управља помоћу регистра и могуће је одабрати да ли ће они бити укључени или не. Микроконтролери ATmega16 користе PORT регистар за управљање заштитним отпорницима, док неки контролери имају специјалне регистре за ову функцију. На слици 2.3 је приказан pull-up отпорника пина Pxn микроконтро-



Слика 2.3: Pull Up отпорник R_{pu} код ATmega микроконтролера

лера ATmega. Отпорник R_{pu} се укључује-искључује преко транзистора T_{pu}, и када је укључен пин Pxn је повезана на V_{CC}, тј. његов напонки ниво је висок (логичка јединица у позитивној логици).

2.3 Дигитални излази

На излазном дигиталном пину је могуће поставити две дигиталне вредности, логичку јединицу (висок напонски ниво) и логичку нулу (низак напонски ниво). Код ATmega16 микроконтролера чији је напон напајања $V_{CC} = 5\,\mathrm{V}$, максимална вредност ниског напонског нивоа је $0,7\,\mathrm{V}$, а минимална вредност високог напонског нивоа је $4,2\,\mathrm{V}$.

Све док је вредност за одређени пин у DDR регистру подешена као излазна, контролер поставља напонске нивое на том пину у складу са вредошћу у PORT регистру која одговара пину. Излазни пин у општем случају може бити струјни извор или струјни понор (потрошач). Ефективна вредност струје која може протицати кроз дигитални пин се обично креће у опсегу $4 - 20\,\mathrm{mA}$.

Карактеристике:

- Омогућавају постављање напона пина на жељени ниво.
- Atmega16 „low“= max 0.7V; „high“= min 4.2 V
- Све док је DDR подешен за излаз, пинови се сетују према PORT регистру.
- Излазни пин може да „даје“ или „вуче“ струју
- Вредности струје за излазне пинове миктроконтролера су $[4 - 20]\mathrm{mA}$[a], највише.

[a] 20mA је довољно за напајање стандардне LED диоде

При пројектовању хардвера и софтвера, дигиталним излазима треба посветити посебну пажњу због могућности изазивања кратког споја. У пракси се дешава да је пин дигиталног порта преко спољашњег хардвера физички спојен на GND. Ако се тај пин користи као излазни, кад год се он постави на висок напонски ниво изазива се кратак спој (КС), што је крајње непожељно и може трајно оштетити микроконтролер.

Излазни пинови су критичнији од улазних.
Уколико излазни пин физички спојимо на GND и сетујемо га на 1 долази до **кратког споја!**

Произвођачи микроконтролера често дозвољавају краткотрајан кратак спој. Могуће је и програмски проверити да ли је изазван кратак спој на излазном пину дигиталног порта:

- Очитава се PIN регистар и упоређује са вредношћу PORT регистра.
- Кратак спој је могуће детектовати тек пошто је настао (кашњење).
- Ако се детектује КС, пин се сетује као излаз и обавештава се корисник.

Поставља се питање који регистар прво поставити:
- DDR или
- PORT?

Ако контролер не користи битове PORT регистра за друге намене (као Atmega16 где се помоћу PORT регистра управља „pull-up“ отпорницима), прво сетујемо PORT, а затим DDR регистар. Код Atmega16 (и осталих контролера фамилије Atmel AVR) је мало компликованије, јер PORT управља „pull-up“ отпорницима. Ако је прва вредност која се поставља на излаз „1“, и сетује се PORT а затим DDR, краткотрајно ће се укључити „pull-up“ отпорници, пре него што порт постане излазни. У највећем броју случајева ово неће имати последице на микроконтролер и спољашњи хардвер, али је увек потребно пажљиво проверити да ли ће краткотрајно укључивање „pull-up“ отпорника имати последице на спољашњи харвер.

Закључак
Ако контролер дозвољава, а хардверу не смета, сетујемо прво PORT па DDR. Увек треба проверити могуће нежељене ефекте.

2.4 Илустративни пример

Задатак 1. Потребно је подесити пинове порта B микроконтролера на следећи начин:

- пинови 0 и 1 „high“
- пинови 2 и 3 „low“
- пинови 4–7 улазни
- пинови 6 и 7 „pull-up“

Пиновима 0–3 додељујемо одређене вредности, што значи да ће они бити излазни. Решење које се односи на контролер Atmega16, у асемблеру је:

Asembler Atmega16

```
...
1  ; definisati pull-up i postaviti izlaze na "high"
2  ; definisati smer za pinove porta
3  ldi r16,0xC3
4  ldi r17,0x0F
5  out PORTB,r16
6  out DDRB,r17
7  ; Ubaciti nop za sinhronizaciju
8  nop
9  ; Očitati vrednost pinova
10 in r16,PINB
...
```

Линије 1, 2, 7 и 9 су коментари (почињу тачка зарезом „ ; “).

У линији 3, се у регистар `r16` уписује хексадецимални број C3 (префикс 0x означава да је у питању хексадецимални запис). 0xC3 је у бинарном запису једнако 11000011. Јединице на битовима $b0$ и $b1$ одговарају захтеву задатка да се пинови 0 и 1 поставе на „high“. Јединице на битовима $b6$ и $b7$ одговарају последњем захтеву задатка да се на пиновима 6 и 7 укључе „pull-up“ отпорници.

У линији 4 се у регистар `r17` уписује хексадецимални број 0F, што је једнако бинарном броју 00001111, а ово одговара постављању огворајућих пинова као улазних или излазних.

Према препоруци, прво постављамо PORT регистар порта B, а то је `PORTB`, а затим одговарајући DDR регистар–`DDRB` наредбом `out`. У десетој линији помоћу наредбе `in` учитамо садржај одговарајућег PIN регистра `PINB` у регистар `r16`, који можемо користити у даљем току програма.

Инструкција `nop` (*No Operation*) је наредба која ништа не ради, али је време њеног извршавања једнако времену извршавања стандардне инструкције, те се често користи за синхронизацију, на пример за чекање да се напон на излазним пиновима постави на задати ниво.

Једно решење задатка у програмском језику `C` је:

C кôд

```
unsigned char i;
... /* definisati pull-ups i postaviti izlaze na "high" */
/* definisati smer za pinove porta */
PORTB = (1«PB7)|(1«PB6)|(1«PB1)|(1«PB0);
DDRB = (1«DDB3)|(1«DDB2)|(1«DDB1)|(1«DDB0);
/* Ubaciti nop za sinhronizaciju*/
_NOP()
/* Očitati vrednost pinova*/
i = PINB;
...
```

3 Аналогни улази и излази

Миктроконтролер обрађује дигиталне информације и резултат те обраде је такође у дигиталном облику. Није увек довољно представити вредност неког напона са два нивоа, некада је потребно познавати тачну вредност напона. На пример ако користимо фототранзистор као детектор осветљености за укључивање-искључивање светла у условима дан/мрак, довољне су две вредности, али када желимо да аутоматски подешавамо осветљеност просторије дневним светлом променом угла ролетни у „паметним кућама“ неопходан је и податак о измереној осветљености, тј. вредност напона са фото транзистора.

Дигиталне информације са аналогних сензора:
- фототранзистор,
- потенциометар,
- селсин,
- ...

Аналогно–дигитални (АД) конвертор конвертује аналогни напон у дигиталну вредност (број). Након прорачуна угла ролетни из нашег примера, потребно је тај резултат доставити актуатору задуженом за физичко постављање тог угла. Извршни елементи (актуатори) су такође најчешће аналогни (мотори, грејачи, пумпе,...) што намеће потребу и за коришћењем дигитално–аналогних (ДА) конвертора.

Не поседују сви контролери АД и ДА конверторе.
(**цена**)

3.1 Дигитално аналогна конверзија

Дигитални сигнал је број, који се у бинарном облику може представити са r битова:

$$B = (b_{r-1}, b_{r-2}, \ldots, b_0)_2 \tag{3.1}$$

где је $r \geq 1$, и број B се може наћи у опсегу $[0, 2^r - 1]$. Задатак дигитално-аналогног конвертора је да на основу вредности B генерише њему пропорционалан напон V_o. Иако је ова особина често пожељна при пројектовању апликација, микроконтролери најчешће имају мале или чак никакве могућности за генерисање аналогних напона, тако да се овај задатак најчешће[1] мора решавати изван микроконтролера.

Микроконтролери имају* мале могућности генерисања аналогних излаза

* чита се ***немају*** , али . . .

Уколико апликација захтева употребу ДА конвертора, конверзија се може извести употребом спољашњег ДА конверотра као посебне компоненте, или чак без њега.

3.1.1 Аналогни излаз без ДА конвертора

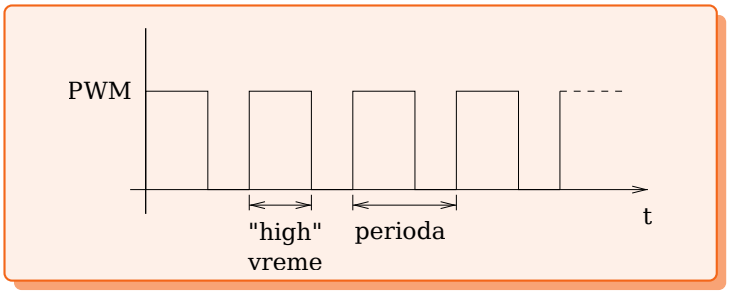
То што микроконтролери не поседују уграђене ДА конверторе, не значи да је немогуће генерисати аналогни напон, уз мало основних знања из електронике и електротехнике. Анализираћемо употребу лествичастих мрежа отпорника са једне стране и имуплсно ширинску модулацију и RC филтар са друге стране, као помоћ микроконтролеру при генерисању аналогних напона.

3.1.2 Импулсно ширинска модулација (PWM) и RC филтар

PWM је сктраћеница од енглеског назива *Pulse-Width Modulation*–импулсно ширинска модулација. Аналогни напон је могуће генерисати:

- употребом само једног пина микроконртолера,
- користећи PWM и
- RC филтар.

Типичан PWM сигнал са једног пина микроконтролера је приказан на слици 3.1.



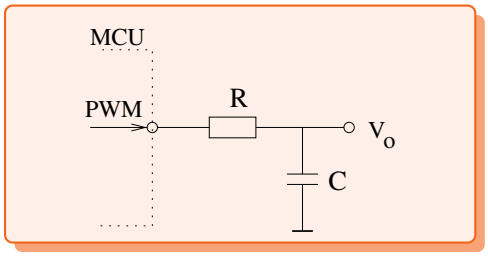
Слика 3.1: Типичан PWM сигнал, где је *high* време трајања јединице

Потребно је генерисати PWM сигнал код кога су однос трајања високог напонског нивоа и периоде пропорционални дигиталној вредности B. PWM сигнал се затим ради уобличавања филтрира RC филтром пропусником ниских учестаности (слика 3.2).

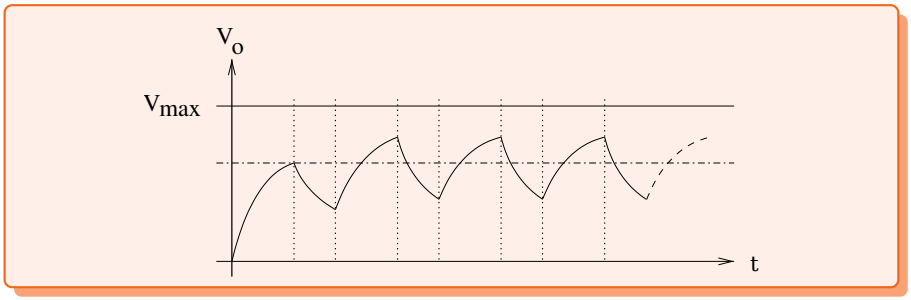
Пример аналогног напона на излазу RC филтра је приказан на слици 3.3.

Као што се може приметити са слике 3.3 излазни напон има осцилаторни карактер, али је његова средња вредност пропорционална дигиталној вредности B. Амплитуда осцилација зависи од избора отпорника и кондензатора, као и од периоде

[1] може се рећи и увек, али се оставља могућност за долазеће генерације микроконтролера



Слика 3.2: Илустраиција *RC* филтра пропусника ниских учестаности..



Слика 3.3: Аналогни напон на излазу *RC* филтра.

импулсно ширинске модулације. Смањење осцилација се може постићи одабиром великих отпорности и капацитивности елемената у филтру, али се тиме продужава време стабилизације, тј. време од тренутка задавања нове вредности напона до тренутка када је средња вредност напона на излазу филтра пропорционална задатој дигиталној вредности. Без обзира на осцилаторна својства овакав аналогни напон се може успешно користити у многим апликацијама, на пример када је извршни орган мотор једносмерне струје (намотаји у мотору се понашају као додатни филтер) или за регулацију температуре (грејач). Треба имати у виду да су излази микроконтролера мале снаге у односу на наведене актуаторе, те је потребно користити и појачавач снаге.

Особине PWM sa *RC* филтром:

- Осцилаторност аналогног напона.
- Осцилације зависе од избора R и C, као и од периоде PWM.
- Велики R и $C \rightarrow$ { мање осцилације; дуже време стабилизације }
- Потребан је прецизан тајмер за генерисање PWM сигнала.
- Користи се само један пин контролера

3.1.3 Мреже отпорника у АД конверизији

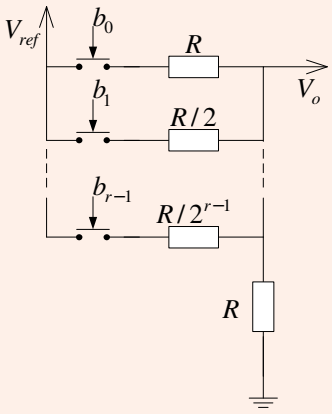
Бинарна мрежа отпорника

Други начин генерисања аналогног напона је помоћу бинарне мреже отпорника приказа не слици 3.4.

Аналогни напон V_o се израчунава према:

$$V_o = V_{ref} \sum_{i=1}^{r} \frac{1}{2^i} b_{r-i} = V_{ref} \frac{B}{2^r} \tag{3.2}$$

што значи да зависи од дигиталне вредности B и референтног напона V_{ref}. Недостаци овог решења су употреба различитих отпорника мале толеранције, те се ретко користи у пракси.

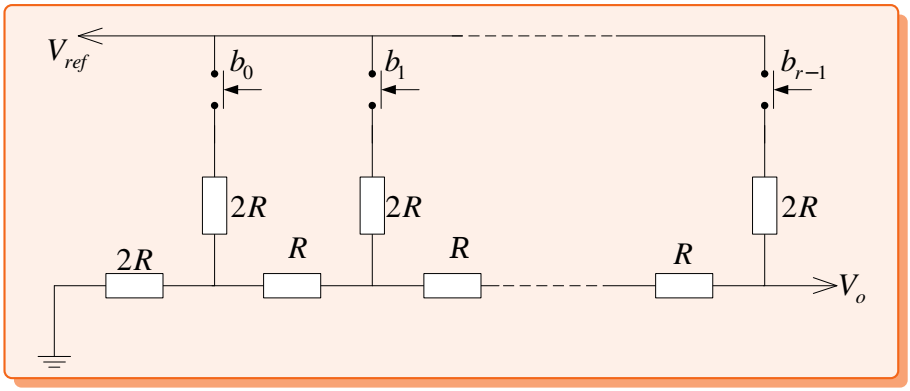


Слика 3.4: Бинарна мрежа отпорника за генерисање аналогног напона.

Неке особине:
- r битни дигитални улаз,
- потребни прецизни отпорници,
- **тешко остварљив у пракси** .

$R-2R$ лествичаста мрежа

Решењем приказаним на слици 3.5 се превализе недостаци претходног решења. Излазни напон V_o се рачуна према једначини (3.2).



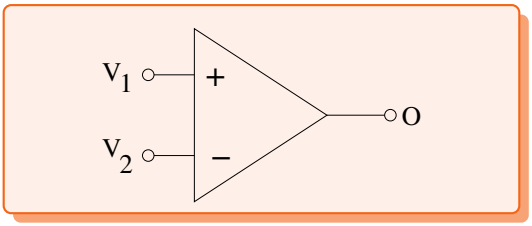
Слика 3.5: $R-2R$ лествичаста мрежа отпорника за генерисање аналогног напона.

Особине $R-2R$ мреже
- r битни дигитални улаз.
- Потребна само два типа отпорника.
- Лакше остварљив у пракси.

3.2 Аналогни компаратор

У неким случајевима није потребно познавати тачну вредност напона, већ само податак о томе да ли је напон већи од унапред задатог прага, или не. Неки микроконтролери због тога имају уграђен аналогни компаратор (слика 3.6). Његов задатак да је да пореди два аналогна напона и генерише дигитални резултат–бит који даје информацију о међусобном односу тих напона. У раније разматраном примеру са фототранзистором, аналогни компаратор би се могао користити за укључење

вештачког осветљења када интензитет природног осветљења опадне испод унапред задате вредности.



Слика 3.6: Аналогни компаратор. Аналогни улази: V_1, V_2 . Дигитални излаз: O.

Оба улаза аналогног компаратора могу потицати од спољашњих извора, или један може бити спољашњи а други референца генерисана у микроконтролеру. У аналогном компаратору је такође могућа појава метастабилности када су улазни напони приближно једнаки ($V_1 \approx V_2$), јер за разлику од дигиталних улаза, овде се не користи Шмитово коло.

Карактеристике аналогног компаратора:

- поређење напона са референтним напоном
- када је $V_1 > V_2$ излаз $O = 1$
- када је $V_1 \leq V_2$ излаз $O = 0$
- **метастабилност** када је $V_1 \approx V_2$ (не користи се Шмитово коло)
  - нарочито у комбинацији са прекидима

3.3 Аналогно дигитална конверзија

Вратимо се на пример са фототранзистором са почетка поглавља, где нам је потребна вредност напона као информација о осветљености просторије, као један од улазних података за алгоритам (подешавање нивоа дневног светла у „паметној“ кући). Пошто је излаз фототранзистора–напон аналогна величина, а алгоритам микроконтролера може обрађивати само дигиталне податке, уочава се потреба за уређајем који аналогне величине претвара у дигиталне, тј. аналогно-дигиталним конвертором.

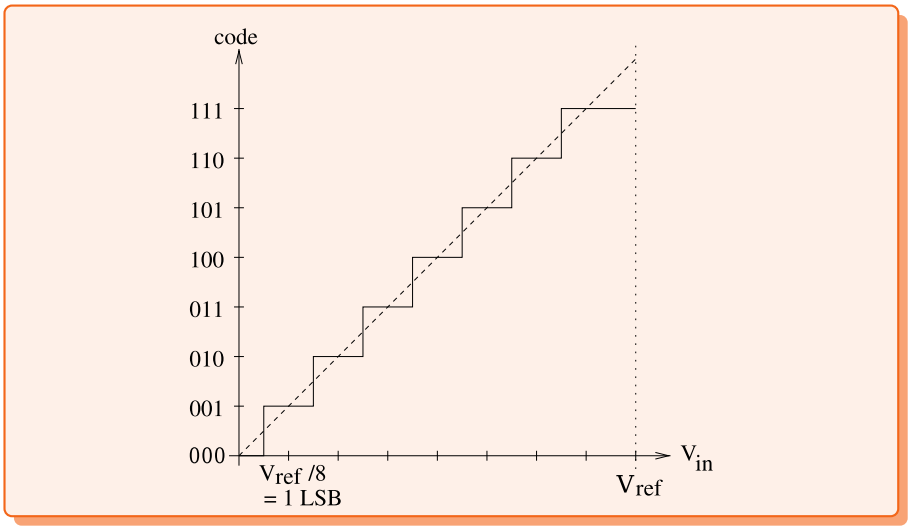
АД конвертор се користи:

када је потребна информација о вредности напона.

3.3.1 Принцип рада

Улазно-излазна карактеристика једног АД конвертора је приказана на слици 3.7.

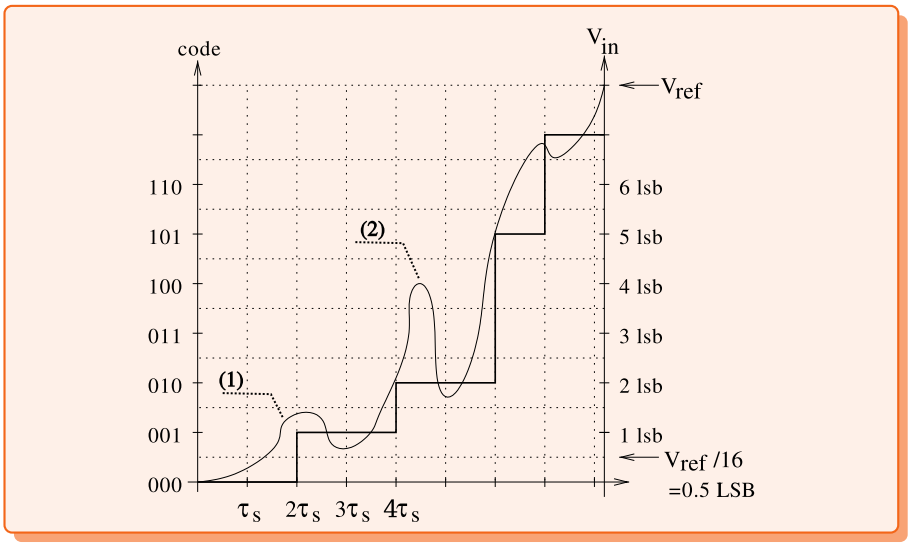
Аналогни улазни напон се налази у опсегу $[\,\mathrm{GND}, V_{ref}]$ и подељен је на 2^r делова–класа, где r представља број бита који се користи за представљање дигиталне вредности у бинарном запису. Свака класа аналогног напона одговара једном бинарном броју из опсега $[0, 2^r - 1]$. Аналогне вредности се пресликавају у одговарајуће представнике своје класе. Број r се назива резолуцијом АД конвертора и његове типичне вредности у пракси су 8 или 10, мада се у неким применама користе и АД конвертори веће резолуције (нпр. 12 или више). Бит најмање тежине–*LSB* (Least Significant Bit) представља најмању промену улазног напона која се може детектовати, што је



Слика 3.7: Улазно-излазна карактеристика АД конвертора ($r = 3$, GND $= 0$).

једнако $V_{ref}/2^r$. Ова вредност се у литератури назива грануларност АД конвертора, али се често означава и као резолуција. Ширина већине класа одговара вредности једног LSB, са изузетком прве чија је $0,5$LSB и последње класе чија је ширина $1,5$LSB. Ова асиметрија потиче од потребе да се аналогна вредност од 0 V представи бинарним бројем 0, тако да прва класа има само половину ширине осталих. Да би се избегла асиметрија могуће је најнижу вредност класе одабрати као њеног представника, али је у том случају грешка дигитализације 1LSB уместо $\pm 0,5$LSB као за пример на слици 3.7, где је репрезентативни представник средина класе.

На слици 3.8 је приказан принцип рада АД конвертора резолуције $r = 3$, чији је напон GND $= 0$ V.



Слика 3.8: Принцип рада АД конвертора ($r = 3$, GND $= 0$).

Са слике 3.8 се може уочити да АД конверзија аналогни улазни напон V_{in} не пресликава у потпуности верно у дигитални еквивалент. Пресликавања аналогног напона у класе као последицу има губитак информација. Осцилације унутар опсега једне класе се не могу детектовати, на пример све осцилације близу тачке (1) на слици 3.8 се пресликавају у исти дигитални напон-број $(001)_2$, као последица мале резолуције АД конвертора. Резолуцију по напону АД конвертора можемо повећати на два начина:

- смањењем напона V_{ref}, што као последицу има мањи дозвољени опсег улазног напона, или
- повећањем r, што изискује и већи број бита за представљање дигиталне вредности.

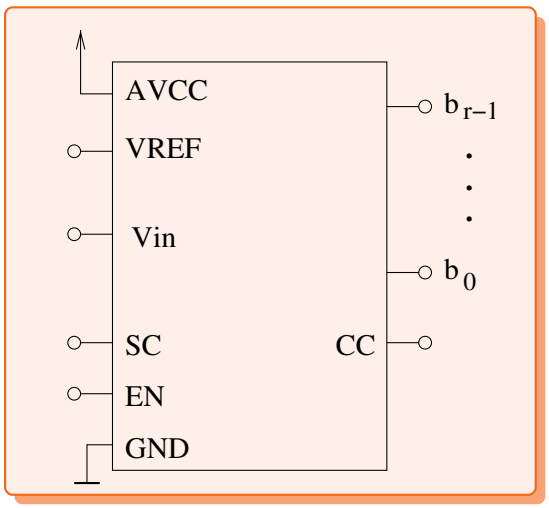
Такође треба узети у обзир и време неопходно за АД конверзију, јер оно није једнаки нули - од почетка АД конверзије па до тренутка када се на излазу АД конвертора појави њен резултат протекне коначно време, које је један од основа за дефинисање минималне периоде одабирања τ_s у једној апликацији. Промене улазног напона у току периоде одабирања се такође не могу детектовати (тачка (2) на слици 3.8). Највећа фреквенција $f_{\max}$ у улазном аналогном напону која се може реконструисати из дигиталног еквивалента, је теоријски одређена теоремом одабирања, познатом и као Шенонова теорема, или Никвистов критеријум (једначина 3.3).

Шенонова теорема:

$$f_{max} < \frac{f_s}{2} = \frac{1}{2\tau_s} \tag{3.3}$$

На основу теореме о одабирању (3.3) највећа фреквенција улазног сигнала $f_{\max}$ треба да буде мања од половине фреквенције одабирања f_s.

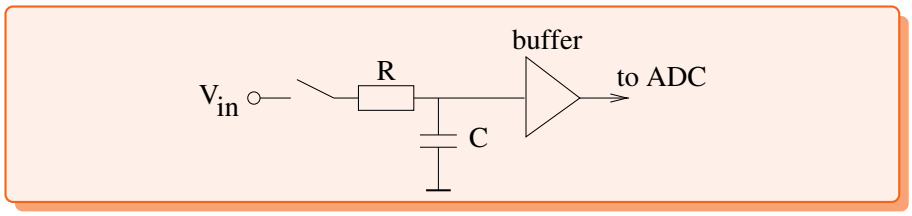
АД конвертор је као блок приказан на слици 3.9. Напајање се прикључује између пинова $AVCC$ и GND. Референти напон V_{REF} је уједно и максимални напон који се може дигитализовати. Прикључак за улаз који се дигитализује је означен са V_{in}. Помоћу EN бита (*ENable bit*) се омогућава конверзија, а она почиње одмах након сетовања SC (*Start Conversion bit*) бита. На излазној страни АД конвертора су пинови са којих се очитавају битови дигитализованог напона $[b_0, \ldots, b_{r-1}]$ и CC пин (*Conversion Completed*) који обавештава о завршетку коверзије.



Слика 3.9: Блок АД конвертора:
- $AVCC$ – напајање
- V_{REF} – референтни напон
- V_{in} – улазни напон
- SC – почетак конверзије
- EN – омогући конверзију
- CC – конверзија завршена
- $[b_0, \ldots, b_{r-1}]$ Битови дигитализованог напона

Референтни напон V_{REF} (слика 3.9) се уобичајено доводи на екстерни пин, али неки миктроконтролери (нпр. ATmega16) обезбеђују референцу и унутар микроконтролера, тако да корисник може да изабере да ли ће користити уграђену или спољашњу референцу. Треба обратити пажњу на случајеве када улазни напон излази из интервала $[GND, V_{REF}]$, јер када је улазни напон V_{in} већи од V_{REF} он се конвертује у $2^r - 1$, а када је мањи од GND конвертује се у број 0. Савременији АД конвертори у овим случајевима генеришу специјалан флег о прекорачењу опсега конверзије.

Осцилације сигнала током процеса конверзије могу утицати на тачност конверзије, те је уобичајено да се на улазу АД конвертора користи коло задршке (нултог реда), које обезбеђује стабилан напонски ниво сигнала током трајања процеса АД конверзије (слика 3.10). На почетку конверзије кондензатор је напуњен и напон на њему је једнак улазном напону V_{in}, након чега се отвара прекидач и напон на улазу у АД конвертора је константан за време трајања конверзије (једнак је напону на кондензатору).



Слика 3.10: Коло задршке нултог реда

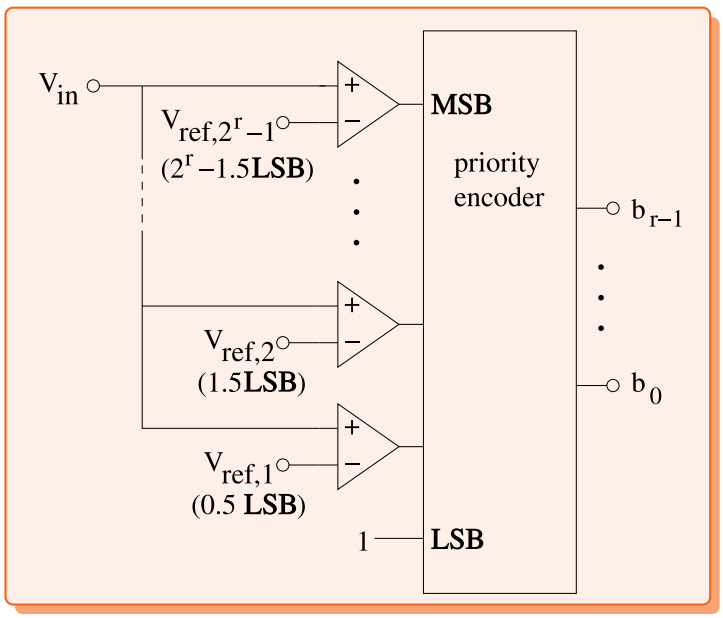
3.3.2 Технике АД конверзије

Аналогно дигитални конвертори се могу реализовати на различите начине:

- Конвертор са директном конверзијом (страна 30),
- Инкрементални АД конвертор (страна 31),
- АД конвертор са сукцесивним апроксимацијама (страна 32),
- ...

Конвертор са директном конверзијом

Функционално најједноставнији АД конвертор је конвертор са директном конверзијом, који се среће и под називом флеш (*flash*) АД конвертор [4], чији је принцип рада илустрован на слици 3.11.



Слика 3.11: Конвертор са директном конверзијом– *Flash* АД конвертор

Улазни напон се пореди са неколико референтних напона $V_{ref,i}$, који се могу одредити на следећи начин:

$$V_{ref,i} = \frac{V_{ref}(2i-1)}{2^{r+1}}, \qquad 1 \leq i \leq 2^r - 1 \tag{3.4}$$

Ако је улазни напон већи од референтног напона на компаратору $V_{ref,i}$, излаз i-тог компаратора ће бити 1. Сви компаратори су повезани на енкодер који као излаз генерише бинарни број који одговара компаратору највеће тежине који има ненулти излаз. Када је излаз свих комапратора једнак нули резултат АД конверзије је 0.

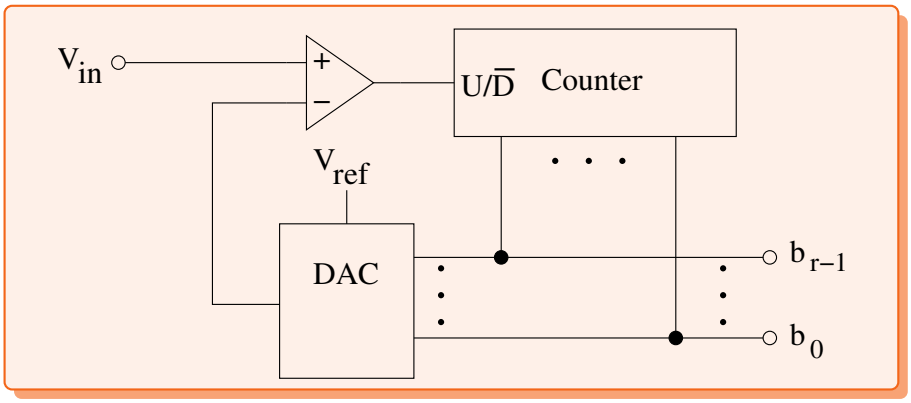
На основу једначине (3.4) можемо закључити да су ширине класа (види део 3.3.1, на стр. 27) једнаке 1 LSB, осим за минимални напон 0 V чија је ширина 0,5 LSB и максимални напон коме одговара опсег ширине 1,5 LSB.

АД конвертор са директном конверзијом одликује велика брзина конверзије [2], јер се конверзија изводи у само једном кораку. Са друге стране недостатак је велика сложеност хардвера: потребно је $2^r - 1$ компаратора, а повећање резолуције за један бит захтева двоструки број компаратора, због тога је овај тип АД конвертора скуп.

Флеш АД конвертор
- брзина (flash) √
- цена ×
- димензије ×

Инкрементални АД конвертор

Блок шема инкременталног АД конвертора (*енг. Tracking Converter*) је приказана на слици 3.12. Инкрементални АД конвертор користи ДА конвертор (DAC). Принцип рада је једноставан: Излаз бројача се конвертује у аналогни напон и на компаратору пореди са улазним напоном, и у зависности од резултата поређења, бројач се инкрементира или декрементира (ако је улазни напон већи од тренутне вредности бројача, бројач се инкрементира, иначе се декрементира).

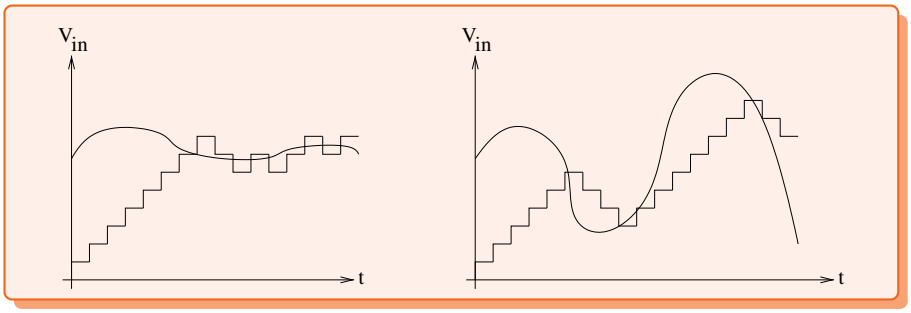


Слика 3.12: Инкрементални АД конвертор– *Tracking Converter*

Овај АД конвертор је спор за већину апликација. Време потребно за извршење АД конверзије није константно, зависи од садржаја бројача у почетку конверзије (слика 3.13)

Инкрементални АД конвертор
(*Tracking Converter*)
- користи ДА конвертор за АД конверзију,
- исувише спор за већину апликација.

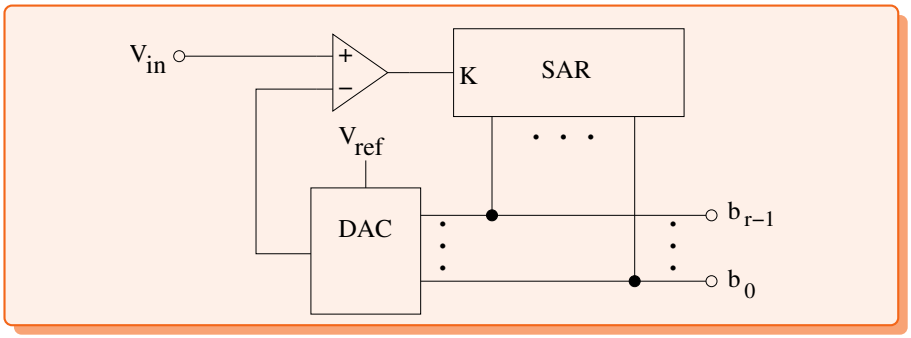
[2] flash–муња



Слика 3.13: Принцип рада. Лево: споре промене улазног напона V_{in}. Десно: брзе промене V_{in}.

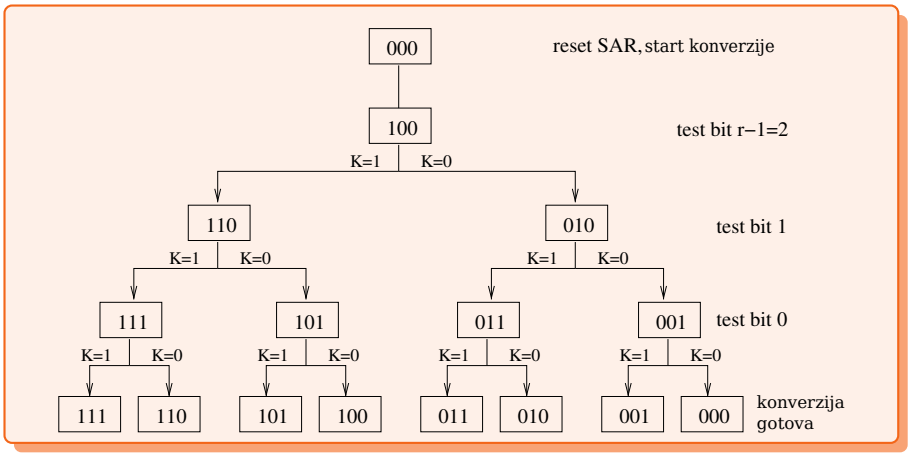
АД конвертор са сукцесивним апроксимацијама

Блок шема АД конвертора са сукцесивним апроксимацијама је приказана на слици 3.14, и може се приметити да се од инкременталног АД конвертора (слика 3.13) разликује само по блоку који је означен као *SAR*. Блок *SAR* (енг. *Successive Approximation Register*) је регистар сукцесивних апроксимација, који омогућава бинарно претраживање уместо инкрементирања и декрементирања.



Слика 3.14: АД конвертор са сукцесивним апроксимацијама.

Принцип бинарног претраживања (познатог и као метода половљења интервала) је илустрован на слици 3.15. На почетку конверзије MSB ($b_{2^r}-1$) регистра сукцесивних апроксимација се поставља на 1 и добијена аналогна вредност се пореди са улазним аналогним напоном V_{in}. У зависности од резултата поређења одређује се коначна вредност бита $b_{2^r}-1$: Ако је V_{in} већи онда вредност бита $b_{2^r}-1$ остаје један, иначе се он поставља на нулу, па се процедура понавља за бит $b_{2^r}-2$, $b_{2^r}-3$, све до b_0, тј. до краја АД конверзије. После r поређења вредност регистра сукцесивних апроксимација одговара вредности аналогног улазног напона V_{in}.



Слика 3.15: Бинарно претраживање–половљење интервала.

АД конвертор са сукцесивним апроксимацијама

- Користи SAR уместо бројача
- Бинарно претраживање уместо инкрементирања и декрементирања
- Константно време конверзије
- Често се користи у микроконтролерима

3.3.3 Практична примена ADC

Микроконтролери са аналогним улазима обично имају 4 до 16 аналогних улазних канала, који су преко мултиплексера повезани на један АД конвертор, што значи да није могуће истовремено конвертовати аналогне напоне повезане на различите канале, већ се резултати конверзије морају очитавати један за другим. Одабере се канал и започне конверзија. По завршетку конверзије читамо резултат и иницијализујемо следећи канал. Неки контролери нуде могућност аутоматизације процеса конверзије са више канала.

- Уобичајено 4–16 аналогних улазних канала.
- Више канала се **не може истовремено** очитавати.

Старт конверзије може задати корисник, али микроконтролери уобичајено омогућавају задавање почетка конверзије преко аутоматског уграђеног механизма. Осим специјалног пина у статус регистру АД конвертора, којим корисник може задати старт конверзије (single conversion mode), већина микроконтролера нуди непрекидан мôд, што значи да нова конверзија започиње одмах након завршетка претходне. Старт конверзије се може задати и преко тајмера, неког спољашњег сигнала или неког дефинисаног догађаја. Конверзија има коначно трајање, а пошто микроконтролери најчешће користе АД конвертор са сукцесивним апроксимацијама, оно је константно.

- Старт конверзије може задати корисник, али и други извори trigger-а.
- Често постоји непрекидни мод (i-конв. готова, почиње $i+1$).
- Конверзија има своје трајање.

За АД конверзију је потребан сигнал такта који одговара спецификацијама АД конвертора, који може бити и спољашњи. Некада је потребно конфигурисати и прескалер да би се ускладиле брзине такта и АД конвертора. Ако је такт спорији од спецификација АД конвертора, конверзија ће трајати дуже него што је потребно, али ако је такт бржи од потребног осим утицаја на време конверзије могући су и додатни ефекти: нетачна конверзија (погрешан LSB). Са порастом брзине такта (изнад дозвољене), расте и број погрешно одређених нижих битова, све до бита највеће тежине, што значи потпуно нетачну АД конверзију.

- За конверзију је потребан сигнал такта (властити или спољашњи)
- Скалирање такта – може бити потребно
- Сувише брз такт, нетачна конверзија

Након завршетка конверзије, сетује се одгварајући флег у статус регистру АД конвертора (*CC–Conversion Completed*), а могуће је и генерисати прекид као информацију о завршетку конверзије. Резултат конверзије се складишти у регистар.

Пошто резолуција АД конвертора може бити већа од речи у микроконтролеру, на пример када се користи 10-битни АД конвертор на 8-битном миктроконтролеру, неопходно је *атомско* читање регистра. Уобичајено да контролери имају уграђени механизам за *атомско* читање, на пример ATmega16 чува садржај регистра након читања нижег бајта, и тек када се прочита и виши бајт дозвољава упис резултата нове конверзије у регистар.

Када улазни напон излази ван дозвољених граница, он ће се конвертовати у вредност која одговара доњој или горњој граници (на пример 0 или $2^r - 1$). Да би се избегла евентуална оштећења миктроконтролера потребно је да напон V_{in} увек буде у оквиру граница дозвољених спецификацијама миктроконтролера.

- Крај конверзије: резултат → регистар
- 10-битни конвертор, 8-битна архитектура?
- Напон ван опсега: LSB, MSB или **трајно оштећење**

3.3.4 Диференцијална/Биполарна конверзија

До сада је разматрана једнострана конверзија, где се аналогни улазни напон пореди са GND. Некада је потребна информација о разлици два аналогна сигнала. Да би се упоредила два аналогна улазна сигнала V^+ и V^- неки АД конвертори нуде диференцијални улаз, где је улаз у АД конвертор разлика сигнала V^+ и V^-.

Поставља се питање колики је дозвољени опсег разлике улазних сигнала. Код једностране конверзије, улазни напон припада опсегу $[\mathrm{GND}, V_{ref}]$ а дигитални излаз је у опсегу $[0, 2^r - 1]$. Напон изван овог опсега се пресликава у вредност ближе границе. Са диференцијалним АД конвертором погодно је да улазни опсег буде на пример $[-V_{ref}/2, V_{ref}/2]$ и да се дозволе негативне вредности.

За представљање резултата конверзије негативних напона из опсега $[-V_{ref}/2, V_{ref}/2]$ микроконтролери користе други комплемент или *excess-K* кôд. *Excess-K* или „померено“ представљање бројева, где K представља позитиван цео број, користи се тако што K представља вредност напона од 0 V, а $-K$ се кодира свим нулама у бинарном запису. Поређење представљања негативних бројева у другом комплементу и $Excess - 4$ је дато у табели 3.1.

Вредност	Други комплемент	$Excess - 4$
3	011	111
2	010	110
1	001	101
0	000	100
-1	111	011
-2	110	010
-3	101	001
-4	100	000

Табела 3.1: Представљање негативних бројева

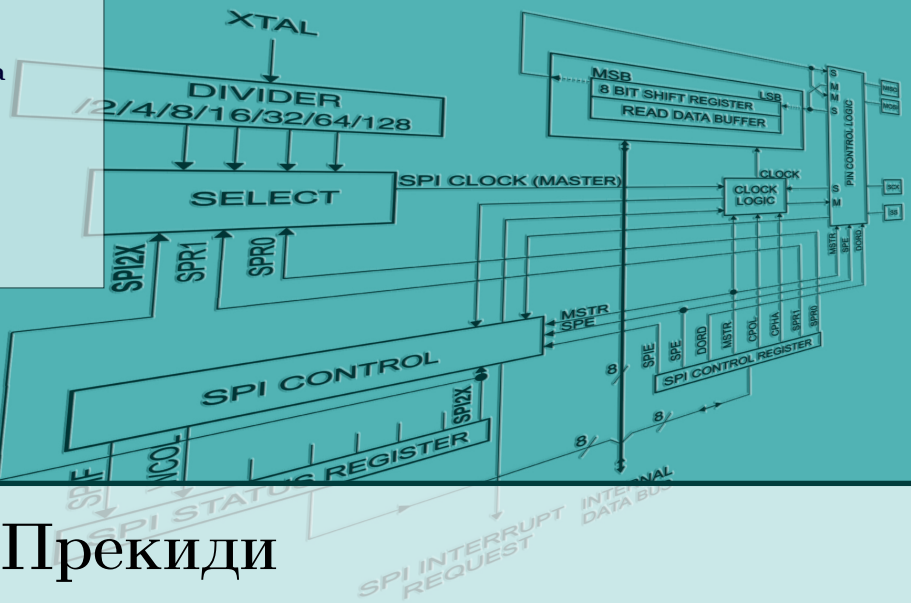
У пракси се често дешава да диференцијални улаз користи само део опсега АД

конвертора: $[-V_{ref}/2,\ V_{ref}/2]$. Да би се резолуција АД конвертора искористила у потпуности аналогни напон се на АД конвертор доводи преко појачавача са подесивим појачањем. АД конвертори у општем случају имају један или више појачавача, али ретко за све канале. Појачање G се код двостраног АД конвертора бира у складу са следећом релацијом:

$$G \cdot (V^+ - V^-) \in [-V_{ref}/2, V_{ref}/2] \tag{3.5}$$

Двострана конверзија

- Директно упоређивање два сигнала
- Неки контролери нуде диференцијални улаз
- Улазни опсег $[-V_{ref}/2,\ V_{ref}/2]$ → негативне вредности
- За представљане нег. бројева → { други комплемент, excess representation
- Подесиво појачање



4 Прекиди

Микроконтролери се примењују у системима код којих се очекује да реагују на одређене спољашње догађаје. Тип реакције може бити једноставан, нпр. инкрементирање бројача, али и захтеван, као што је искључивање напајања када човек уђе у опасан радни простор неког уређаја или машине. Ако препоставимо да микроконтролер може надгледати догађај, поставља се питање на који начин треба то надгледање спровести. Узмимо као пример део алармног система који се односи на отварање једног прозора. Претпоставимо да стање региструјемо помоћу прекидача који је повезан на дигитални улаз микроконтролера: „0“–прозор затворен, „1“–прозор отворен. Дигитални улаз можемо очитавати континуирано, у сваком проласку кроз петљу главног програма, а затим проверавати његово стање. Након детектције стања „1“ позива се одговарајућа процедура. Овај механизам континуираног надгледања је у литератури познат као *polling*. За догађаје који се не могу занемарити и на њих се мора реаговати, а не дешавају регуларно (дешавају се ретко) овакав механизам обраде није најпогоднији.

Периодично (континуирано) проверавање стања (*polling*)
- непотребно трошење процесорског времена (ретки догађаји),
- контролер има “друга посла”,
- тешко је модификовати кôд.

Овакав догађај, који захтева да се на њега (хитно) реагује се назива прекид (енг. *interrupt*). Микроконтролери имају механизам који омогућава извршавање главног програма независно од надгледања прекида. Прекиде надгледа посебан механизам, који прекида извршавање главног програма тек када се прекид деси и при томе позива одговарајућу процедуру за обраду прекида (ISR-*Interrupt Service Routine*). Проблем дешавања више прекида у исто време се решава одређивањем њиховог приоритета.

Механизам прекида
- главни програм се прекида само када дође до промене и реагује се на њу,
- иначе се главни програм извршава без утицаја прекида

4.1 Управљање прекидима

Механизмом прекида се управља помоћу два бита: *interrupt enable*–IE (прекид дозвољен) бит поставља програмер чиме одлучује да ли ће омогућити да микроконтролер реагује на прекид и позове одговарајућу процедуру за обраду прекида као реакцију на догађај. Други бит *interrupt flag*–IF поставља микроконтролер када се догађај деси, и он се брише или аутоматски након уласка у процедуру за обраду прекида или га програмер може обрисати у кôду.

Интерфејс прекида

У микроконтролеру се интерфејс прекида формира помоћу два бита:
IE (*Interrupt Enable*) Одређује да ли контролер треба да реагује на прекиде.
IF (*Interrupt Flag*) Обезбеђује информацију да ли се прекид десио.

IE и **IF** битови у општем случају постоје за сваки извор прекида којим процесор управља, али се некад, да би се уштедели битови, они мапирају у један бит. На пример, код моторолиног контролера HCS12 сваки улазни сигнал дигиталног I/O порта може генерисати прекид. Али постоји само један IE и само једна ISR за цео порт. Међутим, сваки пин на порту има свој IF, што ипак омогућава кораткно обрађивање прекида.

Додатне могућности

- прекид на предњу ивицу сигнала
- прекид на задњу ивицу сигнала
- без промене ивице (на промену нивоа – *level interrupt*)
- *global interrupt enable*

У случајевима када главни програм не сме прекинути извршавање због прекида, временски захтевно и непрактично је забранити све дозвољене прекиде, па микроконтролери нуде могућност глобалног укључивања или искључивања прекида (енгл. *global interrupt enable*).

- Процедура за обраду прекида (ISR) ← IE и IF укључени (*enabled*)
- IE *set* не значи увек да су прекиди дозвољени (*set* / *clear*)
- искључивање прекида не значи сигурно да ће прекид бити пропуштен
  - IE искључен (глобални или локални)
  - прекид се десио: IF сетован, без обзира на IE (глобални или локални)
  - IE укључен → позива се одговарајућа ISR
  - изгубљено време, али не и догађај.
- NMI – NonMaskable Interrupt, не може се искључити глобалним IE
  - NMI поседује властито управљање (TMSP430 фамилија)
- После ресета је уобичајено да су прекиди искључени!

4.1.1 Табела прекида

Само регистровање дешавања прекида није довољно, поставља се питање: **Коју ISR треба позвати?** Мапирање прекида се изводи помоћу табеле прекида (енгл. *interrupt vector table*), која садржи податке за сваки вектор прекида:

- Вектор прекида је једноставан: 1 прекид → 1 број

- Сваки вектор има фиксну адресу, која опет има фиксну адресу у програмској меморији.
- Програмер уноси почетну адресу ISR, или инструкцију скока (JMP) на ISR
- **прекид** $\rightarrow \begin{cases} \text{JMP на локацију у табели} \\ \text{JMP на одговарајући вектор} \end{cases} \Rightarrow \text{ISR}$

У табели 4.1 је представљен део табеле прекида за микроконтролере ATmega16. Табела прекида почиње са адресе $0x0000$. Адреса вектора k је $2(k-1)$ и очекује

Табела 4.1: Део табеле прекида за микроконтрлере ATmega16 [1].

Р.бр. вектора	Адреса у програму	Извор	Опис прекида
Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
⋮	⋮	⋮	⋮
21	$028	SPM_RDY	Store Program Memory Ready

се да контролер „скочи“ на одговарујућу ISR. Приоритет прекида је код ATmega16 фиксан: мањи редни број вектора → већи приоритет.

4.1.2 Приоритет прекида

Прекиде могу иницијализовати различити извори: тајмери, бројачи, спољашњи сигнали, ... и поставља се питање шта се дешава када се више прекида деси истовремено? Код контролера који имају табелу прекида овај проблем је најчешће решен редним бројем прекида, који уједно означава и приоритет прекида. На пример ATmega16 статички додељује највиши приоритет прекиду са најмањим редним бројем. Уобичајено је да се немаскираним прекидима (NMI– NonMaskable Interrupt) додељује највећи приоритет, уколико их микроконтролер подржава.

Такође треба размотрити могућност да ли прекид са вишим приоритетом може прекинути процедуру за обраду прекида нижег приоритета (угњеждени прекид– *nested interrupt*). Код ATmega16 је дозвољено сваком прекиду да прекине ISR која се извршава (без обзира на приоритет), све док је његов IE бит укључен. Како ово често није пожељна ситуација многи контролери искључују глобални IE пре уласка у ISR, или нуде неки други механизам који омогућује избор да ли ће се дозволити прекидање прцедуре за обраду прекида.

Даље, статички приоритет прекида није увек најпогоднији решење за све апликације, па је зато код неких контролера могуће мењати приоритет појединих прекида

у току извршавања програма (динамичка додела приоритета).

- Више прекида истовремено?
- ATmega16:мањи редни број → већи приоритет.
- Да ли се сме прекинути ISR?
- Неки контролери нуде динамичку доделу приоритета.

4.2 Обрађивање прекида

Када контролер поседује механизам прекида, подразумева се да он нуди и начин за њихово обрађивање. То значи да има хардвер намењен за детекцију догађаја као и уграђени механизам за позивање процедуре за обраду прекида. Да би детектовао спољашњи догађај контролер узоркује сигнал са улазне линије на почетку сваког циклуса такта (видети поглавље 2.2 на страни 17) и добијену вредност пореди са претходно узоркованом вредношћу. Када се детектује услов за прекид, постави се IF бит. Логика задужена за прекиде тада проверава да ли је IE бит укључен, и ако јесте, а на чекању нема прекида са већим приоритетом, дешава се прекид, тј. позива се процедура за обраду прекида. Приметимо да је детекција закашњена због кола за одабирање, а сигнал мора бити стабилан дуже од једног циклуса такта да би са сигурношћу био детектован. Сигнали краћег трајања могу, али не морају бити детектовани.

Спољашње догађаје генерише хардвер који је са микроконтролером често повезан везама које нису отпорне на сметње. Ако је на пример прекид дефинисан као узлазна ивица спољашњег сигнала, краткотрајне електричне сметње на линији могу генерисати услове за прекид без обзира на спољашњи хардвер. Овакав шум може генерисати лажне прекиде који су изузетно непогодан извор грешака и необичног понашања програма, јер се ретко дешавају и практично је немогуће пратити њихов утицај. Да би се избегао овакав утицај спољашњег хардвера на извршавање програма на микроконтролеру неки произвођачи у контролере уграђују механизам за избегавање шума (енгл. *noise cancellation*) на пиновима који су повезани са механизмом прекида. Када је механизам за избегавање шума укључен он узоркује сигнал, на пример, $2k$ пута и генерише догађај ако је првих k узорака једнако нули, а осталих k једнако јединици. На овај начин детекција услова за прекид касни за $k-1$ циклуса такта, али се успешно избегава утицај краткотрајних сметњи.

Детекција услова за прекид

- интерни
- екстерни:
  - избегавање шума (*noise cancellation*)
  - лажни прекиди (*spurious–ghost interrupts*)

За интерне догађаји (тајмер и сл.) механизам прекида поседује хардвер који поставља IF бит када је услов задовољен, а остатак механизма за обраду прекида је идентичан претходно описаном. Шум и сметње не утичу на интерне догађаје, те се ни не користе мере за заштиту од њих.

4.2.1 Позивање процедуре за обраду прекида

Позивање процедуре за обраду прекида је више од једноставног скока на одговарајућу адресу. Прво, контролер мора да сними адресу повратка у програм. Неки контролери такође снимају и садржаје регистара. Ако постоји само један прекид у табели контролер може да обрише IF, јер је извор прекида очигледан. У општем случају контролер забрањује прекиде искључивањем глобалног IE бита. Ово омогућава процедури за обраду прекида да се изврши без прекидања од стране другог прекида. Ако ипак треба омогућити неком другом прекиду да се изврши у току њеног извршавања, глобални IE се може укључити у самој процедури за обраду прекида. Овакви угњеждени прекиди су често извор грешака које је изузетно тешко пронаћи и отклонити, па их треба избегавати осим ако нису апсолутно неопходни.

Након завршетка свих припремних радњи, микроконтролер извршава прву инструкцију процедуре за обраду прекида. Процедура за обраду прекида (ISR) је слична свакој другој процедури, осим што се мора завршити наредбом RETI (*return from interrupt*), након чега се контролер враћа на део програма одакле је прекинут и наставља његово регуларно извршавање. Код неких контролера (ATmega16) се након повратка из процедуре за обраду прекида мора извршити бар једна инструкција главног програма пре поновног уласка у процедуру за обраду прекида (ако се прекид десио). Ово омогућава да се главни програм извршава без обзира на учестаност прекида, мада значајно спорије.

Позивање процедуре за обраду прекида:

1. адреса повратка се уписује на стек
2. неки контролери снимају садржаје регистара
3. 1 прекид → бришемо IF
4. прекиди се најчешће искључују помоћу глобалног IE
   - ISR се може извршити без прекида
   - ако мора неки други прекид да се обради у току ISR, IE се може укључути у самој ISR
   - овакав поступак компликује налажење грешака у кôду.
5. извршава се прва инструкција ISR
6. ISR је слична обичној процедури, али се завршава специјалном наредбом – RETI (*return from interrupt*).

Од детекције прекида извршавају се следећи кораци:

Поставља се IF (*Set interrupt flag*). Информацију да се десио прекид контролер уписује у IF.

Завршава се текућа инструкција (*Finish current instruction*). Прекидање већ започете инструкције компликује хардверску изведбу микроконтролера, па је у општем случају једноставније завршити текућу инструкцију пре реаговања на прекид. Време реаговања на прекид је наравно продужено за један или више циклуса такта.

Ако је контролер био у *sleep* моду, иако ниједна инструкција није у току, неопходно је одређено време да се контролер „пробуди“. Ово време може бити и неколико милисекунди уколико се чека стабилизација фреквенције осцилатора.

Одређује се ISR (*Identify ISR*). Појава догађаја који је услов за прекид не значи да ће се аутоматски позвати и процедура за обраду прекида. Уколико одговарајући IE није укључен, то значи да корисник не жели прекиде. Даље, контролер

има више могућих извора прекида и могуће је да се више од једног прекида деси истовремено (у истом циклусу такта). Микроконтролер мора да пронађе који од прекида са укљученим IE и IF битовима има највећи приоритет.

Позива се ISR (*Call ISR*). Након одређивања почетне адресе, контролер снима PC, регистре итд и извршава процедуру за обраду прекида.

Низ акција које се морају предузети пре самог реаговања на прекид као последицу има кашњење од тренутка детекције прекида до реакције на њега. Ово кашњење се креће у интервалу од 2-20 циклуса[1], и његова највећа вредност (под претпоставком да је IE укључен и да нема симултаних прекида) је најчешће наведена у техничкој документацији контролера. Уколико ово кашњење угрожава рад апликације у реалном времену, препоручује се контролер са већом фреквенцијом такта, бржим инструкцијама или контролер који не снима садржај регистара на стек.

4.3 Процедура за обраду прекида – Interrupt Service Routine

ISR садржи кôд неопходан за реакцију на прекид. У њему се може искључити IF (ако већ није искључен) или IE (ако је то неопходно). Процедура за обраду прекида такође може садржати и кôд за обраду догађаја који је узроковао прекид. Одлуку о томе шта треба урадити у самој ISR а шта у главном програму није увек лако донети и умногоме зависи од намене апликације, али и од искуства инжењера. У даљем тексту су дате неке препоруке које могу бити од помоћи у решавању ових недоумица.

ISR мора садржати сав кôд неопходан за реаговање на прекид:

- [брисање IF]
- [искључивање IE]
- [кôд акције]

4.3.1 Избор: Механизам прекида или надгледање?

Прва одлука коју треба донети при обради догађаја је да ли уопште треба користити механизам прекида уместо надгледања у главном програму. Неколико фактора утиче на њу. Да ли је програм мали или велики? Да ли се у главном програму обрађује још нешто осим догађаја који разматрамо? Колико брзо морамо да реагујемо? Да ли је битна промена стања или само ниво сигнала?

Размотримо као пример спуштање/подизање ролетни притиском на дугме помоћу једносмерног мотора. Док је дугме притиснуто мотор треба да се покреће. Надгледање је на први поглед погоднији решење, јер нас занима стање (дугме притиснуто или није) а не тренутак промене стања, али треба размотрити и шта још апликација треба да ради. Ако је овај задатак једини у програму, требало би размотрити коришћење механизма прекида и „sleep“ мода. Кашњење од неколико милисекунди због повратка из sleep мода у овој апликацији неће бити никакав проблем, нити ће га корисник приметити. Ако се у програму обрађују још неки задаци, али не превише њих, надгледање је боље решење (због једноставности). У наведеном примеру не постоји погрешно решење, јер рад у реалном времену није критичан.

[1] термин *latency* се за ову појаву чешће користи на енглеском језику од термина *delay*

Индикације када треба користити прекиде:

- догађај се не дешава често
- дуг интервал између догађаја
- промена стања је битна
- кратки импулси
- догађај генерише хардвер, без бочних ефеката
- ако се ништа друго не ради, прелазимо у *sleep* мод

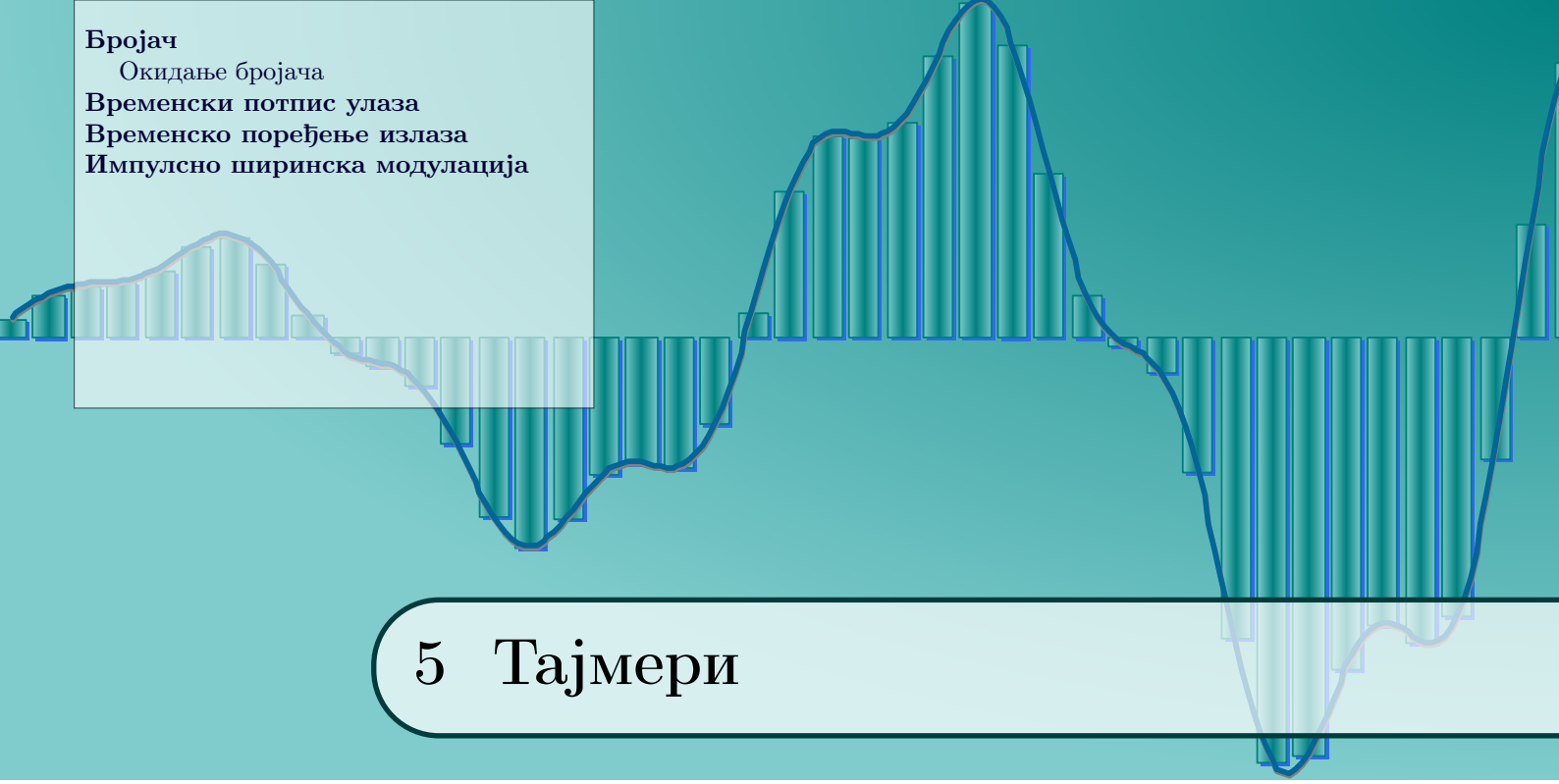
Ситуације када је надгледање вероватно погодније:

- оператор је човек
- не захтева се прецизан „тајминг“
- стање је важно
- импусли су дуги
- сигнал је зашумљен
- контролер „има посла“, али не превише

4.3.2 Реакција на догађај

Следећа недоумица коју треба решити је где написати део програма који реагује на догађај: у процедури за обраду прекида или у главном програму. Иако се чини да је процедура за обраду прекида очигледан избор, јер је она намењена обради прекида, то није увек и боље решење. Ако ISR траје дуго, то значи да ће главни програм правити велику паузу у извршавању. Кратке паузе у општем случају не ометају ток главног програма, али велика пауза у току главног програма због прекида може узроковати нежељене последице. Увек треба размотрити како ће кашњења због обраде прекида утицати на ток главног програма.

Ако са друге стране постоји више извора прекида, ISR чије извршење траје дуго утиче и на време реаговања на друге прекиде који чекају. Иако се прекиди обично не дешавају често, то никако не значи да је време реакције на њих мање важно. Како је показано у пракси погодније је да процедура за обраду прекида садржи што мање кода за обраду догађаја који је иницијализовао прекид, а да се већи део (уколико постоји потреба за тим) извршава у главном програму.



5 Тајмери

Тајмер је у ствари бројач, и уобичајено је да микроконтролери поседују један или више тајмера са 8–битном или 16–битном резолуцијом. Користе се за једноставне задататке као што је мерење различитих кашњења, али и за сложене као што је на пример генерисање таласних облика. Најчешће се користе у функцији бројача, али се помоћу тајмера може доделити и временски потпис екстерном догађају, иницирати прекид након одређеног времена. Такође се помоћу тајмера може генерисати и сигнал са импулсно ширинском модулацијом, који се даље може користити за, на пример, управљање једосмерним мотором.

5.1 Бројач

Сваки тајмер је бројач који може да броји унапред (инкрементирање) или уназад (декрементирање) у складу са циклусом такта. Смер бројања може (али не мора) да буде конфигурациона опција. Тренутна вредност бројача (тајмера) се може очитати из његовог регистра (енг. *count register*). Резолуција тајмера n значи да он може да броји у интервалу од 0 до $2^n - 1$. Када достигне границу, тајмер може да иницира прекид. Ова особина се може искористити за генерисање периодичних сигнала, али са нестабилном периодом, јер када тајмер одброји до краја потребна је његова иницијализација (враћање на почетак), чије трајање није дефнисано. Неки тајмери поседују неприкидни мод (*modulus mode*) бројања, у ком се бројач аутоматкси иницијализује на почетну вредност када изброји до краја. За генерисање периодичних сигнала са прецизно дефинисаном периодом осим непрекидног мода често се користи PWM.

Треба обратити пажњу код микроконтролера који имају тајмере веће резолуције од дужине речи микроконтролера, на пример када се користи 16–битни тајмер у 8–битној архитектури. У овом случају се читање вредности тајмера мора извести у две итерације, што може узроковати грешку.

Нека је тренутна вредност бројача 0x00FF, што значи да ће у следећем циклусу вредност бити 0x0100. Ако читамо прво виши, а затим нижи бајт (у следећем циклусу), очитаћемо вредност 0x0000. Ако пак читамо прво нижи а затим виши бајт, очитана вредност би била 0x01FF.

За превазилажење овог проблема миктроконтролер ATmega16 виши бајт тајмера складишти у регистар, тако да када се год очитава нижи бајт тајмера, виши бајт се аутоматски складишти у регистар и може се очитати у следећем циклусу такта. Такође за уписивање нове почетне вредности у тајмер, прво се уписује виши бајт у регистар и чим се упише и нижи бајт оба се истовремено уписују у тајмер (*count register*).

Карактеристике бројача:

- Инкрементира се или декрементира на сваки циклус такта.
- [Смер бројања може да се мења.]
- Резолуција $n \Rightarrow$ може да броји у опсегу $[0, 2^n - 1]$.
- **Обратити пажњу** када изброји до 2^n!
- **16-битни тајмер у 8-битној архитектури** ?
- Може да се генерише прекид када дође до прекорачења.
- Прекид може да се користи као периода сигнала, али **непрецизна** .

5.1.1 Окидање бројача

Иако је уобичајено да тајмер и микроконтролер коритсте исти такт, то није правило. Бројач могу да окидају следећи извори:

- системски такт (*System Clock–Internal Clock*),
- прескалер (*Prescaler*),
- спољашњи импулс (*External Pulse–Pulse Accumulator*)
- спољашњи кристал (*External Crystal–Asynchronous Mode*).

Системски такт

У овом моду рада тајмер се окида (инкрементира или декрементира) са сваким циклусом такта. Ово је подразумевани мод рада тајмера. Осцилатор који генерише системски такт може бити и екстерни.

Прескалер

У овом моду се такође користи системски такт, али он не окида бројач, него прескалер који затим окида бројач. Прескалер је у ствари и сâм бројач резолуције типично 8 или 10 бита, који се активира системским тактом, а тајмер се активира преко прескалера. На пример, прескалер за ATmega16 микроконтролере се може поставити на вредност из скупа $P \in [8, 64, 256, 1024]$, што у ствари значи да се тајмер окида на сваку осму, шездесет четврту, $\ldots, 2^n$ ивицу такта. Опсег тајмера се може проширити употребом прескалера, али се мора имати у виду и негативна последица, а то је смањење његове прецизности. Због тога се препоручује коришћење најмање вредности прескалера која задовољава потребе апликације.

Прескалером се може проширити опсег тајмера, али се истовремено смањује његова прецизност.

8-битни тајмер, на 1 MHz

- опсег без прескалера је 255 μs, а прецизност 1 μs.
- са прескалером 1024 опсег тајмера је 260 ms, а прецизност 1 ms.

Спољашњи импулс

У овом моду тајмер окида спољашни импулс сигнала који је повезан на одговарајући пин микроконтролера. Тајмер мења своју вредност у складу са променама сигнала, то јест на сваку узлазну ивицу сигнала. Пошто се екстерни сигнал узоркује као и остали дигитални улази, размак између две његове узлазне ивице мора бити већи од периоде системског такта.

Асинхрони мод

У овом моду сигнал такта за тајмер се генерише помоћу екстерног осцилатора који је повезан на два пина микроконтролера. Овај мод се може користити за имплементацију сата реалног времена (RTC-*Real-Time Clock*). Тајмер се инкрементира у складу са независним сигналом такта и ради асинхроно у односу на остатак микроконтролера.

5.2 Временски потпис улаза

Времеснки потпис улаза (енг. *Input Capture*) омогућава додавање временског потписа догађајима (најчешће спољашњим). Може се активирати на улазну ↑, или силазну ивицу сигнала ↓. Кад се догађај деси, тајмер аутоматски копира тренутну вредност у *input capture register*, одакле се даље може користити у програму. Истовремено се сетује *input capture flag*, што омогућава иницијализацију прекида који би обавестио апликацију да је улаз регистрован. Микроконтролери имају један или више пинова који омогућавају ову функционалност.

Input capture се може користити и за интерне догађаје. На пример, ATmega16 може окинути *input capture* са модула аналогног компаратора, што омогућава апликацији читање временског потписа промене стања аналогног компаратора.

Напоменимо да сетовање *input capture* не значи да је аутоматски одговарајући пин подешен као излазни. (Урадити програмски). ATmega16 дозвољава да се пин који је сетован као излазни може користити као *input capture*, ако програм генерише потребне улазе. Ово се може користити за мерење кашњења између излазног догађаја и реакције на њега.

Временски потпис мора бити што је могуће тачнији, а тачност временског потписа се може израчунати према релацији:

$$t_{ev} - t_{cap} \in \left(-d_{in}^{max}, P - 1 - d_{in}^{min}\right] \tag{5.1}$$

где су: t_{ev} време (реално) када се догађај десио, t_{cap} је време временског потписа, d_{in}^{max} и d_{in}^{min} одговарају најгорем случају регистровања, а P је вредност прескалера. На

основу једначине (5.1) можемо закључити да прескалер треба поставити на најмању задовољавајућу вредност. У најгорем случају, грешка мерења између два догађаја у циклусима такта износи:

$$d_{in}^{max} + P - 1 - d_{in}^{min} \tag{5.2}$$

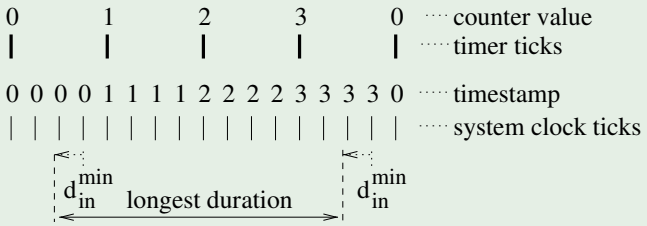
Највећи интервал између два догађаја који неће узроковати прекорачење, у циклусима такта:

$$(2^r - 1) \cdot P \tag{5.3}$$

где је r резолуција тајмера.

Пример

Вредност прескалера P је 4, а резолуција тајмера r је 2;



При читању вредности *input capture* регистра могуће су неочекиване појаве, на које треба обратити пажњу. Под претпоставиком да догађај иницијализује прекид (што је у пракси најчешће случај), а неопходан је и његов временски потпис, размотрићемо две ситуације:

- догађај (1.) "стигао" на улазни пин
- време доласка (1.) уписано у *input capture register*
- ISR (1.) покушава да очита вредност тог регистра
- ↑ траје неколико циклуса такта
- догађај (2.) "стигао" на улазни пин
- време доласка (2.) уписано у *input capture register*
- ISR (1.) очита вредност регистра која одговара догађају (2.)

Пошто иницијализација ISR траје неколико циклуса такта, у тренутку када је очитана вреност *input capture* регистра у њој се налазио временски потпис догађаја који се десио у међувремену, а не догађаја који је покренуо ISR. У пракси ово није проблем, јер је реаговано на други догађај, а први је само пропуштен.

Прави проблем се може десити код микроконтрлера који аутоматски бришу IF пре завршетка ISR:

- догађај (1.) "стигао" на улазни пин
- IF обрисан, позива се ISR
- ISR (1.) се извршава
- догађај (2.) "стигао" на улазни пин
- IF постављен, време доласка (2.) уписано у *input capture* регистар
- ISR (1.) очита вредност регистра која одговара догађају (2.)
- ISR (2.) се изврши, са истим временским потписом

У претходном примеру је реаговано на оба догађаја, али је први обрађен са временским потписом другог, што је крајње непожељна ситуација [3].

5.3 Временско поређење излаза

Ова могућност ради слично као и временски потпис излаза, али уместо да када се нешто деси на улазној линији запамтимо временски потпис, код *output compare* на излазној линији се дефинише промена у одређеном тренутку. Због имплементације ове функције тајмер поседује *output compare* регистар, где програмер уписује тренутак у коме жели промену на излазној линији. Увек када тајмер достигне вредност уписану у *output compare* регистар иницијализује се дефинисани догађај. На овај начин можемо поставити жељени ниво на линију (нула или један) или чак комплементирати претходну вредност нивоа. Такође је ову функцију могуће користити и за иницијализацију прекида.

Опција *output compare* често има уграђени ресет који аутоматски ресетује бројач кад је вредност у *output compare* регистру достигнута. Ово омогућава врло једноставно генерисање периодичног прекида или излазног сигнала.

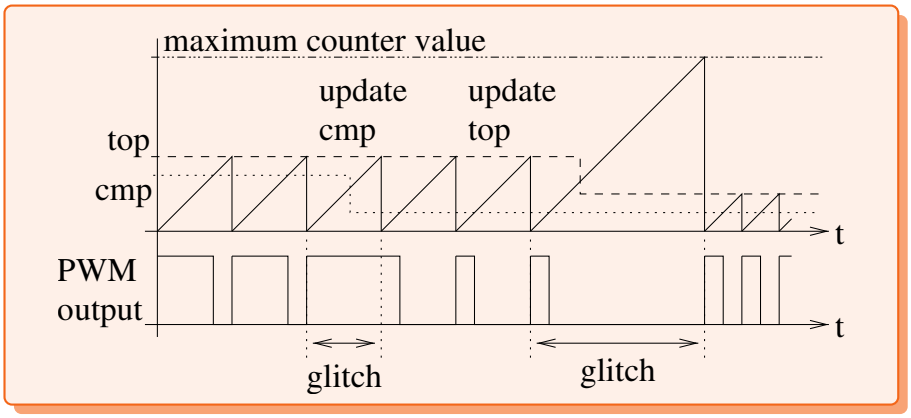
5.4 Импулсно ширинска модулација

Импулсно ширинска модулација (*Pulse Width Modulation*–PWM) је специјалан случај *output compare* у коме тајмер генерише периодичан излазни сигнал са програмабилном периодом и трајањем импулса (паузе). Интерна реализација PWM је једноставна, користи бројач (са бројањем унапред или уназад) и два регистра за поређење. Користе се две имплементације, прва са једносмерним бројачем (унапред или уназад) и друга са двосмерним бројачем. У наредним примерима претпоставићемо да корисник задаје време трајања јединице, за које се у литератури користи израз *compare value*, и периоду сигнала (*top value*).

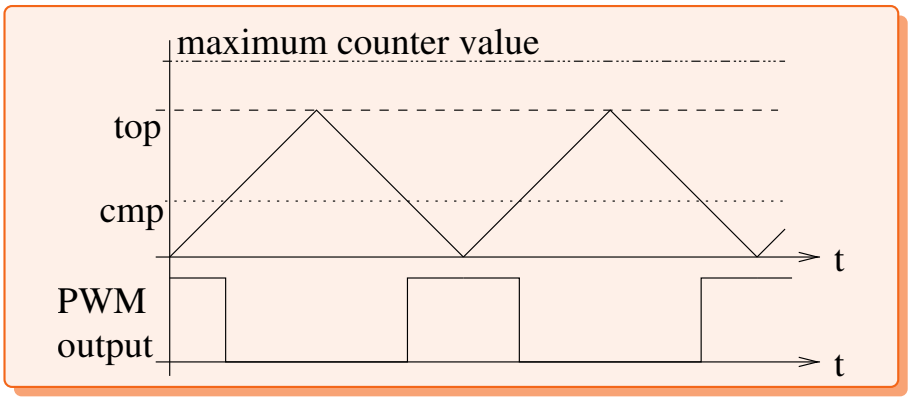
На слици 5.1 је илустрован пример генерисања импулсно ширински модулисаног сигнала употребом једностраног бројача са инкрементирањем (бројањем унапред). PWM сигнал се поставља на јединицу када год је вредност бројача нула, а бројач се ресетује на нулу када достигне вредност уписану у *top* регистар. PWM сигнал се поставља на нулу када вредност бројача достигне вредност уписану у *compare* регистар (*cmp*). Са слике се види да помоћу регистра *top* подешавамо периоду PWM сигнала, док нам *cmp* регистар служи за подешавање времена трајања високог нивоа у сигналу. На слици 5.1 су такође илустроване две ситуације промене параметара PWM. У првој ситуацији је промењена вредност *cmp* регистра на нижу, пре него што ју је бројач достигао, па долази до појаве сметње (енгл. glitch), а у другој ситуацији је на исти начин промењена вредност *top* регистра, што такође узрокује нерегуларни излазни сигнал. Пошто је у другој ситуацији нова вредност *top* регистра у тренутку постављања била испод тренутне вредности у бројачу, бројач је пре сетовања на нулу одбројао до своје максималне вредности, што у зависности од претходне вредности *top* регистра може као последицу имати значајно већу сметњу од промене *cmp* регистра.

У реализацији са двостраним бројачем промене у PWM сигналу се дешавају када вредност бројача достигне вреднност *cmp* регистра, као што је илустровано на слици 5.2. Када бројач при инкрементирању достигне вредност *cmp* PWM сигнал постаје нула, и обрнуто, када бројач при декрементирању дође на вредност *cmp* PWM сигнал се поставља на висок ниво. И у овој реализацији асинхроне промене вреднсоти регистра као последицу могу имати нежељени облик излазног сигнала.

У обе реализације остварљива периода је одређена резолуцијом бројача. Ако се



Слика 5.1: PWM сигнал генерисан помоћу једностраног бројача са инкрементирањем и последице промене параметара у произвољном тренутку.



Слика 5.2: PWM signal generated by an up-down-counter.

вредност *cmp* регистра постави на нулу или изнад вредности *top* регистра, излазни сигнал ће бити идентично једнак нули [3].

Параметри PWM сигнала:

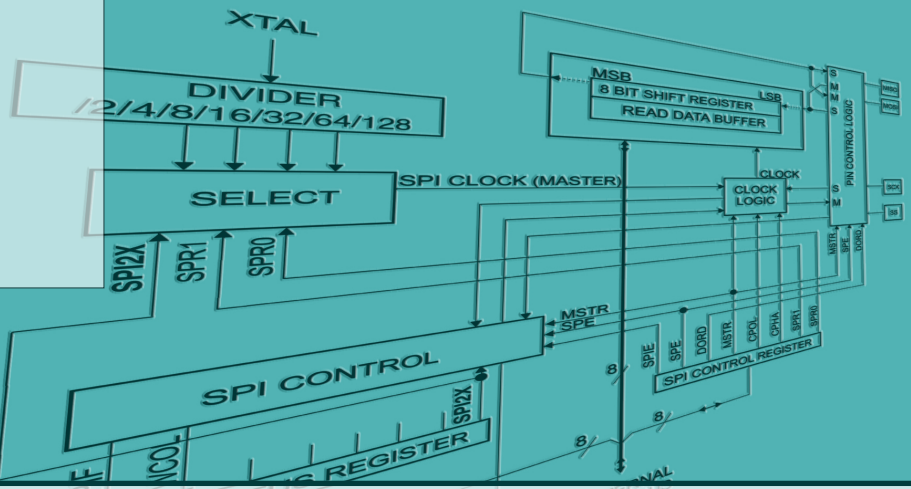
- *top* регистар → периода
- *cmp* регистар → трајање „јединице“
- асинхрона промена *cmp* или *top* регистра → „glitch“

SCI–UART
- RS-232
- RS-422

USART

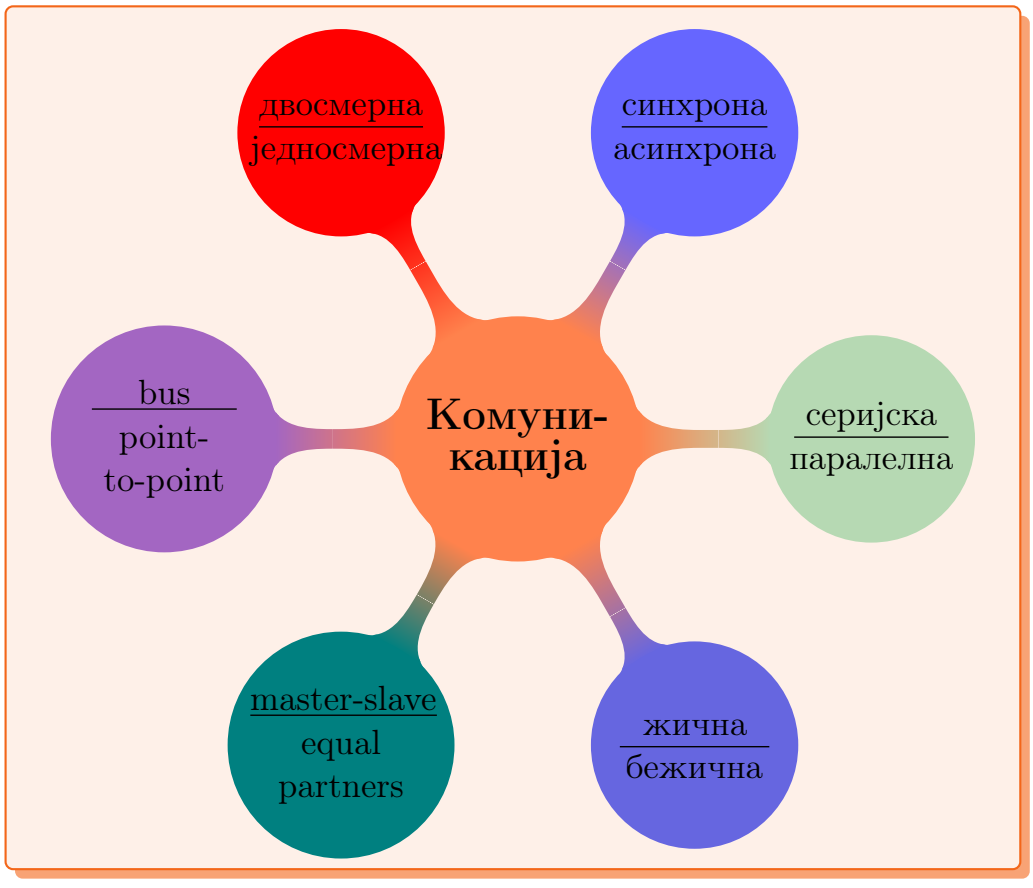
SPI

IIC-I2C
- Пренос података
- Синхронизација такта
- Арбитрација линије



6 Комуникација

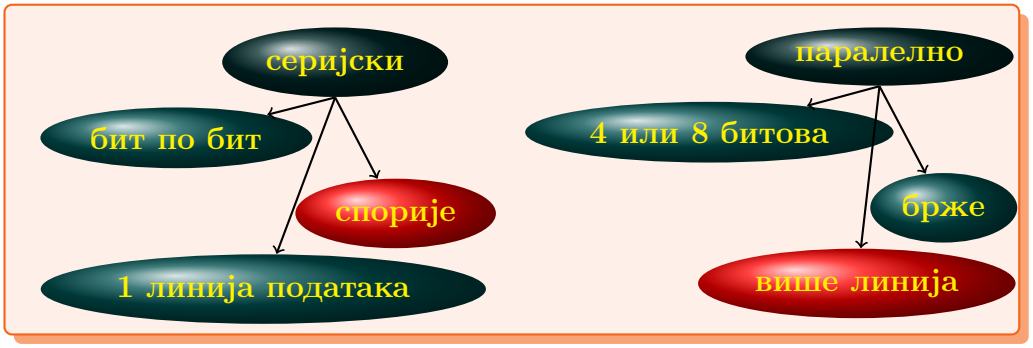
У реалним применама микроконтролера често постоји потреба за комуникацијом са другим уређајима, као што су други микроконтролери, периферије или персонални рачунари. Микроконтролери у општем случају имају више различитих комуникационих интерфејса, па и вишеструке истоветне интерфејсе. Комуникациони интерфејси се могу класификовати на различите начине, на пример као на слици 6.1



Слика 6.1: Класификација комуникације према различитим критеријумима

Серијски интерфејс шаље податке секвенцијално – бит по бит, па овај метод захтева само једну линију за пренос података, што га чини ефикасним у погледу хардверских ресурса. Негативна последица је мања брзина слања (пријема) података.

Са друге стране паралелни интерфејс користи више линија за истовремени пренос више битова. Број битова који се истовремено шаљу може бити различит. За примену у микроконтролерима су нарочито погодне ширине од 4 или 8 битова, зато што одговарају половини или читавом бајту. Паралелни интерфејси оваквих ширина се користе, на пример, за комуникацију са LCD дисплејом. Неке карактеристике серијског и параленог интерфејса су дате на слици 6.2.



Слика 6.2: Поређење паралелног и серијског интерфејса.

Према усклађености такта предајне и пријемне стране, комуникација може бити синхрона или асинхрона. Синхрона комуникација подразумева да су тактови на обе стране комуникационог канала усклађени, тј. синхронизовани. Синхронизација се може извести помоћу додатне линије такта, или рекострунцијом такта на пријемној страни, што сам формат поруке мора омугућавати. Пријемник не мора имати генератор такта, што су уз избегавање грешака насталих због лоше синхронизације предности овог начина комуникације.

Код асинхроних комуникационих модула тактови предајне и пријемне стране нису повезани. Пријемник мора унапред познавати параметре комуникације и подесити своју фрекевенцију према фреквенцији предајника. Пошто тактови нису усклађени узорковање сигнала на пријемној страни мора бити брже од фреквенције предаје (енгл. *oversampling*). Порука такође мора садржати специјалне старт и стоп битове који омогућавају пријемнику да препозна почетак (или крај) поруке. Асинхрона комуникација је у општем случају спорија од синхроне, како због *oversampling*-а, тако и због мање ефикасности (преносе се битови који нису подаци - старт, стоп). Најважније карактеристике синхроне и асинхроне комуникације су издвојене у табели 6.1.

Табела 6.1: Поређење неких карактеристика синхроне и асинхроне комуникације.

синхрона комуникација	**асинхрона комуникација**
• усклађен такт пријем-предаја – додатна линија такта – реконструкција такта	• тактови нису усклађени • неопходан *oversampling* • *START* и *STOP* бит • **спорије**

Трећа подела са слике 6.1 се односи на то да ли је комуникација једмосмерна–*half-duplex* или двосмерна*full-duplex* . Двосмерна комуникација је уобичајена код микроконтролера, што значи да он може у исто време и примати и слати податке ка спољашњем интерфејсу са којим комуницира.

Према начину повезвања учесника у комуникацији, они могу користити заједничку магистралу (енгл. bus) или могу бити директно повезани међусобно (енгл. point-

to-point).

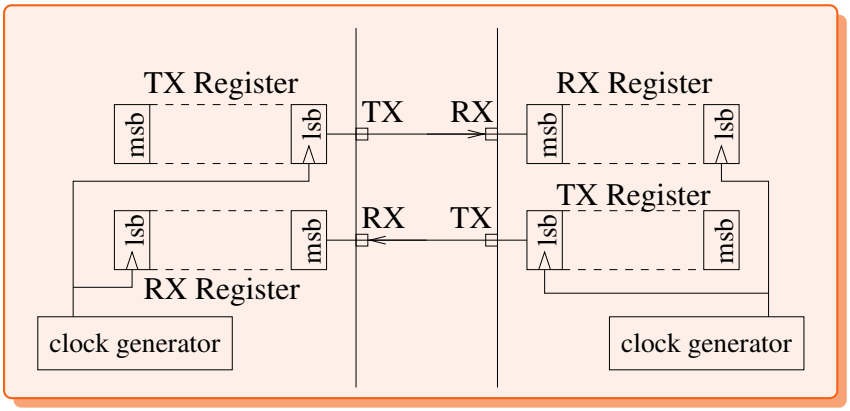
Следећа подела са слике 6.1 master-slave–equal partners се односи на то ко може да започне предају података кроз комуникациони канал. Код master-slave архитектуре само master може започети предају података, док остали учесници (slave) морају чекати дозволу да започну предају. У систему где су сви учесници једнаки (енгл. equal partners) сваки учесник може започети предају, уколико је комуникациони канал слободан. Овакав приступ са друге стране намеће потребу за неким механизмом арбитрације, који ће решити проблем истовременог захтева за предајом од више учесника.

Микроконтролери не комуницирају бежично са другим уређајима, тако да тај тип комуникације нећемо проучавати током овог курса.

6.1 Серијски комуникациони интерфејс - SCI (UART)

Серијски комуникациони интерфејс (SCI–Serial Communication Interface) се назива још и асинхрони комуникациони интерфејс (Universal Asynchronous Receiver Transmitter–UART). UART модул користи две линије, једну за предају–TX и једну за пријем–RX података за једностране или двостране комуникацију (full-duplex, half-duplex).

Структура два повезана UART модула је приказана на слици 6.3. Модул у основи чине два регистра, предајни и пријемни. Пошто је модул асинхрони, оба учесника у комуникацији имају властити генератор такта.



Слика 6.3: Основна структура UART модула

- Користи две линије:
  - TX – предаја
  - RX – пријем
- UART није протокол
- UART је модул

UART није комуникациони протокол, већ модул који се користи за асинхрону комуникацију, а њиме се управља преко микроконтролера. Конфигурациони параметри UART модула су:

Број битова за податке (енгл. *Number of Data Bits*) Број битова за подате се обично може одабрати из широког дујапазона. ATmega дозвољава број из опсега $D \in [5, 9]$.

Бит парности (енгл. *Parity Bit*) Корисник одлучује да ли ће, или неће користити бит парности. Када се користи може бити паран или непаран. Ако је

бит парности сетован као паран, он је једнак 0 када је збир јединица битова података паран. Непаран *Parity Bit* се користи супротно.

Стоп бит(ови) (енгл. *Stop Bits*) У општем случају корисник може одабрати да ли ће користити један или два стоп бита.

Брзина преноса (енгл. *Baud Rate*) UART модул садржи регистар који омогућава кориснику да одабере одговарајућу брзину преноса у битовима у секунди (енгл. *bit per second – bps*). Брзина преноса се најчешће налази у интервалу од 9 600 до 115 200 bps. **Фреквенција такта** утиче на bps.

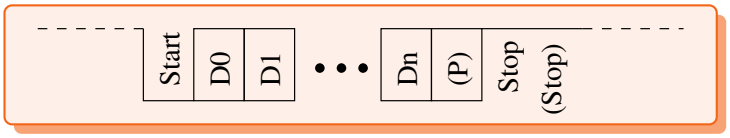
Номенклатура UART поруке

$$D\,\{E|O|N\}\,S$$

где је D број битова података,
S је број стоп битова,
$E|O|N$ значе: *Even* (паран), *Odd* (непаран), *No parity* (без бита парности).
Пример: $8E1$
Број старт битова је увек 1.

Пренос података

Поруке се преносе коришћењем енкодинга „без враћања на нулу“[1] (енгл. Non Return to Zero –NRZ). У позитивној логици у NRZ “1” одговара вишем напону, а “0” одговара нижем напону (у негативној логици је обрнуто). Подаци се увек налазе између бар једног старт бита и стоп бита. На слици 6.4 је илустрован формат UART поруке.



Слика 6.4: UART формат поруке

Када се комуникационом каналом не преносе подаци линија је неактивна (енгл. idle) и постављена је на виши напонски ниво. Порука почиње старт битом, који линију поставља на “0”. Ова силазна ивица је знак пријемнику да нова поруке пристиже. После старт бита следе битови података, од бита најмање тежине ка биту највеће тежине. Њихов број је конфигурациони параметар и мора бити постављен на исту вреност код оба учесника у комуникацији. После битова података у поруци се може (али и не мора) поставити бит парности, који је такође конфигурациони параметар. Порука се завршава једним (или са два) стоп битом. Стоп бит одговара вишем напонском нивоу.

[1] Назив је формиран на основу старијег енкодинга Return to Zero, у коме се у другој половини сваког бита напон поставља на нулу.

- принцип *Non Return to Zero (NRZ)*
- "1" одговара вишем напону[a]
- "0" одговара нижем напону[a]
- подаци су између бар једног старт и стоп бита
- када нема података линија је "*high*"
- пријемник и предајник се морају подесити на исти начин

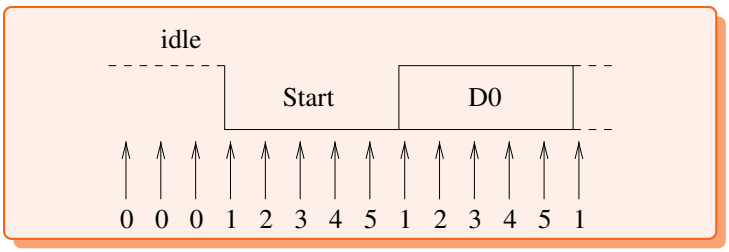
[a] у позитивној логици

Синхронизација

Пошто је комуникација асинхрона, треба имати у виду да су тактови на пријемној и предајној страни комуникационог канала међусобно независни, а и:

- Одабиром *baud rate* пријемник "зна" фреквенцију, али не и тренутак доласка поруке.
- Пријемник мора да се синхронизује са силазном ивицом старт бита.
- *Drift* осцилатора је различит од нуле.
- Након синхронизације на почетку, могуће је да се пријемник раздеси до краја пријема поруке.
- Примењује се ***oversampling*** .
- Пријемна страна узима узорке s пута по биту.
- Типичан вредност за s је 16.
- Код **ATmega16** s је 16, а на основу вредности узорака 8, 9 и 10 се доноси одлука о вредности бита поруке[2]. Због начина доношења одлуке, овакав начин је познат и као гласање (енг. *voting*)

На слици 6.5 је илустровано узорковање поруке на пријемној страни за $s = 5$.



Слика 6.5: Узорковање поруке на пријемној страни за $s = 5$

Брзина преноса

На предајној страни брзина преноса се генерише на основу системског такта. Коришћењем *baud rate* регистра (попут *output compare* регистра) генерише се периодични сигнал такта, који се затим помоћу prescaler-а скалира (s пута) да би се добила одговарајућа брзина преноса.

На пријемној страни ситуација је врло слична, осим што се сигнал такта узима пре прескалера. То значи да пријемник узоркује s пута брже него предајник. Напоменимо да није могуће прецизно постићи сваку жељену брзину преноса, јер скуп могућих брзина преноса зависи од фреквенције такта.

[2] Микроконтролер ATmega16 има два мода рада U(S)ART модула, нормални и двоструки. Код нормалног мода рада се за гласање користе узорци 8, 9 и 10, док се у моду двоструке брзине у ствари користе узорци 4, 5 и 6 (а s је 8).

Потребно је одредити вредност прескалера C, ако је фреквенција осцилатора који генерише системски такт $f = 8\,\text{MHz}$, а *oversampling* $s = 8$, за две брзине преноса: $B_1 = 0.5\,\text{Mbps}$ и $B_2 = 115.2\,\text{kbps}$.

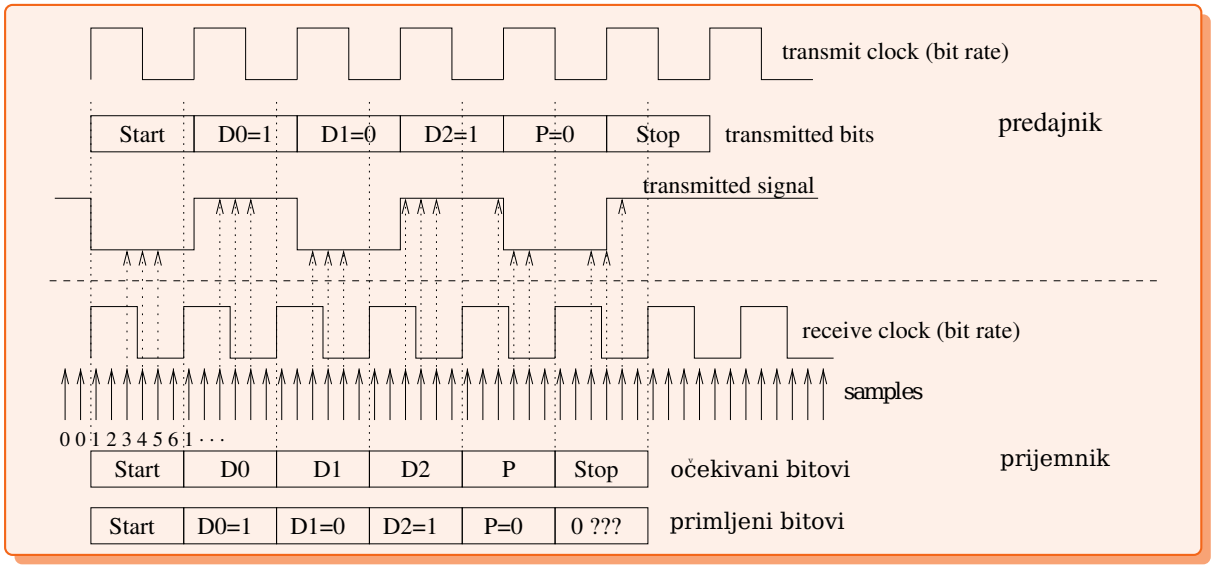
$B_1 = 0.5\,\text{Mbps}$ можемо остварити тачно, уз $C_1 = 2$.
$B_2 = 115.2\,\text{kbps}$ не можемо остварити тачно, јер је:

$$C_2 = \frac{f}{s \cdot B_2} = \frac{8MHz}{8 \cdot 115.2KHz} = 8.68,$$

а прескалер мора бити цео број.

Грешка синхронизације

Раздешеност такта на пријемној и предајној страни може као последицу имати грешке у комуникацији, чак до губитка комуникације, што је илустровано на слици 6.6, за формат поруке $\{3E1\}$, уз $s = 6$, а помоћу узорака 3, 4, и 5 се доноси одлука (јединица или нула).



Слика 6.6: Грешка синхронизације

У илустрованом примеру пријемник није примио стоп бит као последицу раздешавања такта у току пријема, што значи да читава порука није примљена. Ово се у пракси манифестује као немогућност успостављања комуникације (пријемник не прима никакве податке), а учесници не пријављују грешку, јер је веза остварена (али неуспешно).

Релативна грешка синхронизације се може израчунати према:

$$\Delta[\%] = \frac{B' - B}{B} \cdot 100\% \tag{6.1}$$

где је B жељена, а B' остварена брзина израчуната према:

$$B' = \frac{f_{osc}}{s \cdot C} \tag{6.2}$$

У претходној једначини C представља вредност прескалера (цео број), док је s *over-sampling* (такође цео број).

Да би израчунали релативну грешку синхронизације за пример са стране 56, прво ћемо израчунати реално остварену фреквенцију такта према једначини (6.2):

$$B' = \frac{8MHz}{8 \cdot 9} = 111111$$

а затим према једначини (6.1) и релативну грешку синхронизације:

$$\Delta[\%] = \frac{111111 - 115200}{115200} \cdot 100\% = -3.5\%$$

Пошто се у реалним применама користе фреквенције осцилатора из предефинисаног скупа, уобичајено да произвођачи у документацији микроконтролера дају вредност релативних грешака за све остваривe комбинације фреквенција осцилатора и прескалера, као на пример у табели 6.2.

Табела 6.2: Релативна грешка за фреквенцију осцилатора 8 MHz и различите вредности прескалера за микроконтролере ATmega16 [1], где је:

- UBBR – USART Baud Rate Register
- Asynchronous Normal Mode → $U2X = 0$
- Asynchronous Double Speed Mode $U2X = 1$

Baud Rate (bps)	f_{osc} = 8.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%
4800	103	0.2%	207	0.2%
9600	51	0.2%	103	0.2%
14.4k	34	-0.8%	68	0.6%
19.2k	25	0.2%	51	0.2%
28.8k	16	2.1%	34	-0.8%
38.4k	12	0.2%	25	0.2%
57.6k	8	-3.5%	16	2.1%
76.8k	6	-7.0%	12	0.2%
115.2k	3	8.5%	8	-3.5%
230.4k	1	8.5%	3	8.5%
250k	1	0.0%	3	0.0%
0.5M	0	0.0%	1	0.0%
1M	–	–	0	0.0%

Жељена брзина преноса (Baud Rate) се код контролера Atmega16 поставља помоћу два регистра: UBRR (Baud Rate Register) и $U2X$. Регистар $U2X$ нам говори да ли је одабрана регуларна или двострука брзина.

Максимална дужина поруке

Увидом у колоне табеле 6.2 примећујемо да је релативна грешка синхронизације за већину одабраних комбинација UBRR и $U2X$ различита од нуле, што није случај

само код Atmega16 контролера. Како је реално очекивати грешку синхронизације при асинхроној комуникацији, од интереса је израчунати максималну дужину поруке која ће бити успешно пренешена, упркос раздешености фреквенција такта пријемне и предајне стране.

Број битова k поруке који ће бити успешно примљени можемо израчунати на основу следеће релације [3]:

$$k < \frac{v_n \cdot f_{TX}}{s \cdot (f_{RX} - f_{TX})}, \qquad f_{RX} > f_{TX} \tag{6.3}$$

где су f_{TX} реална фреквенција предаје, f_{RX} реална фреквенција пријема, s број *oversampling* и v_n редни број бита за гласање. У случају када је $f_{TX} > f_{RX}$ важи релација:

$$k < \frac{s \cdot f_{RX} - (v_n + 1) \cdot f_{TX}}{s \cdot (f_{TX} - f_{RX})}, \qquad f_{RX} < f_{TX} \tag{6.4}$$

Релације (6.3) и (6.4) важе само у условима када не постоји шум.

Грешке у преносу могу бити:

Грешка због шума :
- Ако се користи бит парности може се детектовати грешка у 1 биту,
- ***parity error*** се сетује у UART регистру статуса.

Грешка због раздешене синхронизације :
- не препознаје се стоп бит
- генерише се ***frame error***

Data overrun :
- Нови пакет пре читања старог из бафера.

6.1.1 RS-232

UART је асинхрони серијски комуникациони модул, али не дефинише никакве физичке карактеристике интерфејса, као што су на пример напонски нивои. У микроконтролеру се битови података једноставно пресликавају у напонске нивое који одговарају логичкој нули или јединици. Међутим, уз одговарајуће коло за прилагођење нивоа, UART се може користи за комуникацију са пуно различитих физичких интерфејса. Један од стандардних који се користи у персоналним рачунарима, а и шире је RS-232.

RS-232 је намењен за такозвану комуникацију *point-to-point*, и потпуни стандард дефинише 25 линија, мада се код персоналних рачунара много чешће среће само 9 линија (D-SUB9 конектор). Од ових девет битова UART користи само два (RXD и TXD), мада се у пракси захтева и трећи (GND). Остали битови су контролни и управљачки и користе их напреднији протоколи.

По RS-232 спецификацији радни напонски нивои треба да се налазе у опсегу $\pm 3 - 15\,\mathrm{V}$, а уређај мора дозволити максималних $\pm 25\,\mathrm{V}$. Управљачке линије користе позитивну логику, док линије података користе негативну логику.

Пошто микроконтролер не може да подржи напонске нивое које захтева RS-232 стандард, неопходно је користити коло за прилагођење напонских нивоа, као на пример MAX232.

6.1.2 RS-422

Сандард RS-422 се, као и RS-232, користи за *point-to-point* комуникацију, али користи диференцијалне линије (и RXD и TXD су изведени као упредене парице) тако да не захтева заједнички GND. RS-422 се користи уместо стандарда RS-232 када се мора проширити домет комуникација, мада је RS-422 бољи избор у зашумљеном окружењу, због чега је овај стандард прихваћен као индустријски.

6.2 USART[3]

Универзални синхроно–аисхрони примопредајни модул – USART допуњава функционалност UART модула тако да он постаје синхрони. Због тога USART користи једну линију више, којом преноси сигнал такта. У синхроном моду сигнал такта генерише један од учесника у комуникацији, а користе га оба, како за пријем тако и за слање података. Пошто је такт на обе стране усклађен, *oversampling* механизам је непотребан, па је синхрона комуникација s пута бржа од асинхроне. У USART модул је уграђена логика која подржава и синхрони и асинхрони мôд рада. Када се користи као асинхрони, линија такта је слободна и одговарајући пин се може користити као стандардни I/O дигитални пин.

Иницијализација USART модула код ATmega16 [1] – C кôд

```
#define FOSC 1843200// Frekvencija oscilatora
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
...
USART_Init ( MYUBRR );
...
}
void USART_Init( unsigned int ubrr)
{
/* Postavljanje brzine prenosa (baud rate) */
UBRRH = (unsigned char)(ubrr>>8);
UBRRL = (unsigned char)ubrr;
/* Omogućiti prijem i slanje */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Format poruke: 8 data, 2 stop bita - 8N2 */
UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Ова реализација је за асинхрону комуникацију и то у случају када се не користи механизам прекида. Брзина преноса (baud rate) је параметар функције.

[3] USART–Universal Synchronous Asynchronous Receiver Transmitter

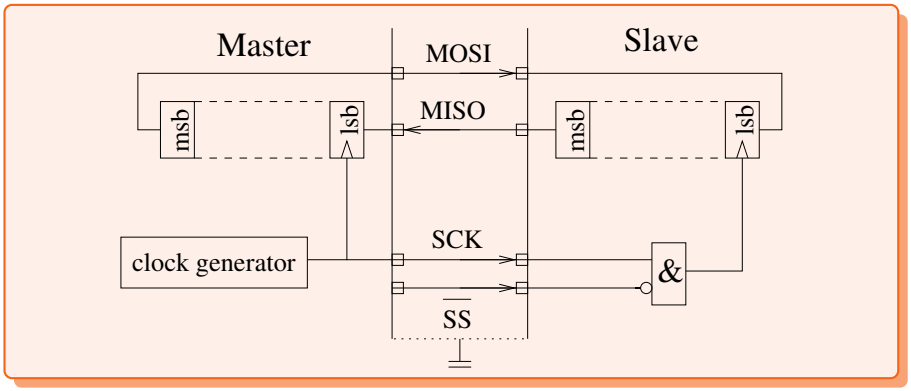
6.3 Серијски периферијски интерфејс–SPI

Серијски периферијски интерфејс (енгл. Serial Peripheral Interface-SPI) је једноставан серијски *point-to-point* интерфејс заснован на *master-slave* принципу. Он омогућава *full-duplex* комуникацију између мастера (најчешће микроконтролер) и једног или више периферног уређаја (*slaves*).

Интерфејс чине четири *singleended* линије:

1. **MOSI** – *Master Out, Slave In.* Ову линију користи мастер за предају података ка слејву.
2. **MISO** – *Master In, Slave Out.* Овом линијом слејв шаље податке мастеру.
3. **SCK** – *System Clock.* Мастер шаље слејву сигнал такта
4. **SS** – *Slave Select.* Овом линијом мастер селектује слејв.

Основни принцип рада SPI модула је илустрован на слици 6.7. Мастер и слејв имају по један померачки регистар који су подржани сигналом такта SCK. Са сваким циклусом такта MSB (или LSB-зависи како је конфигурисано) из померачког регистра са стране мастера се преко MOSI линије уписује у померачки регистар слејва као LSB. Истовремено MSB се са стране слејва преко MISO линије уписује као LSB у померачки регистар мастера. Након 8 цилкуса такта мастер и слејв размењују свих 8 битова из својих регистара.



Слика 6.7: SPI–Серјски периферијски интерфејс

Размена садржаја регистара:

- интерни померачки регистар
- на сваки импулс такта MSB мастера преко MOSI долази у регистар слејва као LSB
- ↑ на сваки импулс такта MSB слејва преко MISO долази у регистар мастера као LSB
- после 8 циклуса, мастер и слејв су разменили поруке

Мастер мора експлицитно селектовати слејв постављајући $\overline{SS}$ на "low". Могуће је поставити два слејва на SPI ако један директно реагује на $\overline{SS}$, а други прво инвертује линију. Уколико мастер може да користи више I/O пинова, број слејвова се може повећати на 2^n употребом спољашњег декодера [3].

6.4 IIC–I²C–I2C

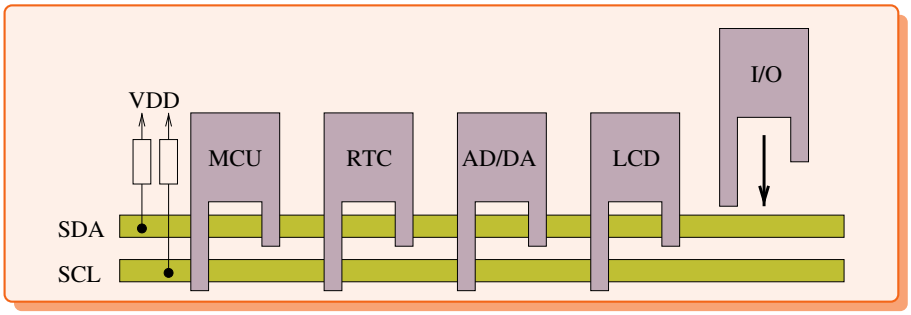
Стандард је развила и патентирала компанија Philips Semiconductors (сада NXP Semiconductors) [5] као двожичну-двосмерну магистралу за ефикасну синхрону ко-

муникацију између микрочипова. У литератури се среће под називима IIC, I^2C, или I2C, када се користи патентирани стандард. Неке компаније обезбеђују хардверску компатибилност својих производа са I2C стандардом, али уместо патентираног назива користе *Two Wire Interface*–двожична магистрала.

Комуникација се одвија по master – slave принципу и користи се на кратким растојањима (IIC–Inter-Integrated Circuit). Може да фукционише у пет различитих модова [5]:

1. *Standard mode* – 100 kbit/s
2. *Fast mode* – 400 kbit/s
3. *Fast mode Plus* – 1 Mbit/s
4. *High-speed mode* – 3.4 Mbit/s
5. *Ultra Fast-mode* – 5 Mbit/s (али је једносмеран)

Прва четири мода (стандардни, Fm, Fm+ и Hs) су међусобно компатибилна (виши са нижим), што не важи за *Ultra Fast-mode* у коме је комуникација једносмерна. На слици 6.8 је илустрован пример повезивања уређаја преко IIC магистрале. За пренос података се користе две линије: линија података – SDA (Serial Data Line) и линија такта – SCL (Serial Clock Line). Сваки уређај повезан на ову магистралу има своју јединствену адресу. На магистралу је могуће повезати више од једног мастера (мора постојати бар један) захваљући интегрисаним механизмима арбитрације и детекције колизије. Максималан број уређаја који се могу повезати на исту магистралу је ограничен само њеном капацитивношћу.



Слика 6.8: Пример повезивања уређаја на IIC магистралу

- једноставно додавање нових уређаја на магистралу
- подржава 7-битно и 10-битно адресирање
- највиши битови (3 или 4) су хардверски резервисани (произвођач)
- адресе $(0000XXX)_2$ и $(1111XXX)_2$ су резервисане
- преостаје 112 могућих адреса за слејвове

6.4.1 Пренос података

IIC је *single-ended* магистрала код које су напонски нивои су дефинисани у односу на масу:

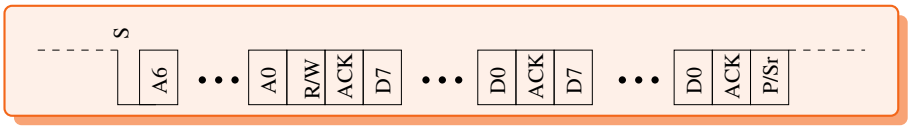
- *low level* је у опсегу $[-0,5 \quad 0,3 \cdot V_{DD}]$ V (позитивна логика)
- *high level* је у опсегу $[0,7V_{DD} \quad V_{DD} + 0,5]$ V (позитивна логика)

Пошто су физички проводници повезани преко спољашних *pull-up* отпорника, *high level* се никада не успоставља. Ова асимтеричност између *low level* и *high level* узрокује то да линија има доминатно и рецесивно стање (*dominant and recessive state*). Уколико уређај да на излазу 0 и линија прелази у „ниско“ стање, оно остаје ниско

чак и ако један или више уређаја да на излазу „1“. Пошто „0“ увек побеђује, ова особина се зове и жичано И коло (*wired-AND*), а користи се и за арбитрацију.

I²C пакет

Пакет података је илустрован на слици 6.9.

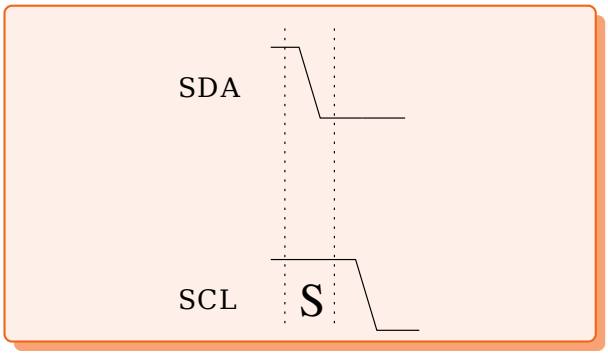


Слика 6.9: I²C пакет података

- Неактивно стање је високо
- Предају иницира мастер, стављањем сигнала такта на SCL и услова старта (S) на SDA линију.
- Мастер ставља адресу (A6A5...A0) на магистралу и врсту приступа R/W.
- Након R/W бита *slave* шаље потврду (ACK) да је препознао адресу.
- Шаљу се подаци (D7D6...D0), са потврдом (ACK) након сваког бајта.
- На крају преноса, мастер генерише STOP (P) и магистрала је слободна,
- или понавља START (Sr – Start repeated) чиме се завршава текући пренос и почиње нови.

СТАРТ и Поновљени СТАРТ

START услов (S) је приказан на слици 6.10. Можемо приметити да њега карактерише силазна ивица на SDA линији, док је на SCL линији стање високо.



Слика 6.10: START услов за I²C пакет

Услов поновљеног старта (Sr) се не разликује од START услова. У примени, он замењује STOP/START услове у случајевима када мастер жели да настави коришћење магистрале. Када постоји један мастер у конфигурацији, овако се штеди један циклус такта, али када је више мастера на магистрали (Sr) спречава арбитрацију, чиме обезбеђује да тренутна комуникација задржава право на коришћење магистрале.

Напомена

Само START и STOP мењају ниво на SDA док је SCL на “high”. Нормалан ток података, укључујући и потврду – ACK, мења ниво када је стање на SCL “low”.

Адресе и управљање смером (Direction Control)

Код 7-битних а дреса прво се шаље msb. SDA ниво се мења када је SCL на ниском нивоу, а чита се када је SCL на високом нивоу.

После 7 адресних битова, мастер завршава бајт слањем (R/W) бита који означава смер следеће трансмисије. Ако је R/W висок, мастер жели да чита податке, а ако је R/W низак мастер жели да шаље податке. Сваку трансмисију на магистрали започиње мастер, слањем адресе слејва, ако је R/W висок, слејв шаље потврду и мења се смер – слејв почиње са слањем података.

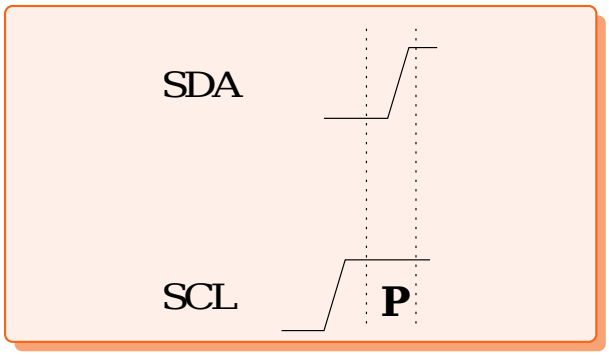
Подаци и потврда пријема (Acknowledgement)

Битови података се шаљу као и остали битови, и сваки бајт се мора потврдити од стране пријемника. Прво се шаље msb бит. Не постоји ограничење броја битова података у једном фрејму предаје.

После сваких 8 битова пријемник шаље потврду – ($\overline{ACK}$) да је примио податке. Потврда се шаље постављањем SDA на "low". Једини изузетак је последња потврда када је мастер пријемник: тада мастер не потврђује последњи бајт и SDA остаје "high", што сигнализира слејву који предаје податке да је пренос завршен. Слејв ослобађа линију и чека поновљени старт или стоп.

STOP

Сигнал STOP (P) је приказан на слици 6.11 и он је супротан услову START, сада се SDA поставља са "low" на "high" када је SCL "high". Стоп шаље мастер када жели да ослободи магистралу. Чим је STOP послат, магистрала је неактивна (idle) и може је захтевати неки други мастер.



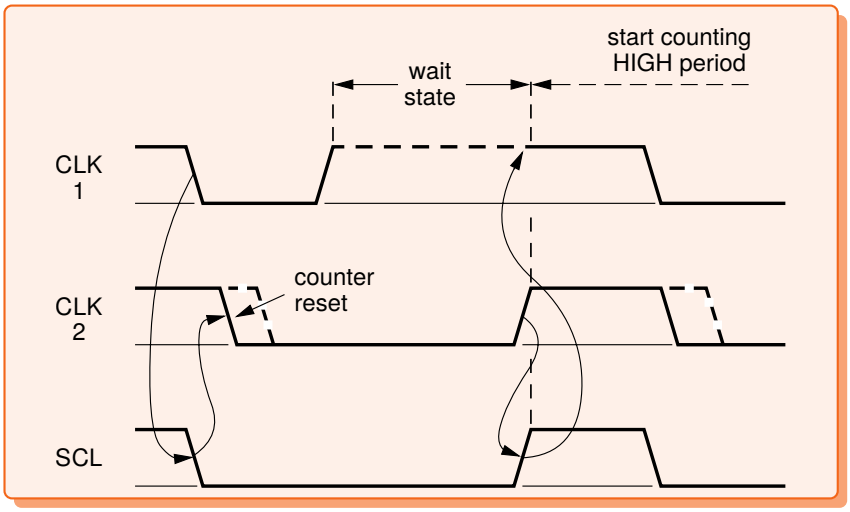
Слика 6.11: STOP услов за I^2C пакет

6.4.2 Синхронизација такта

Поставља се питање: како синхронизовати сигнал такта када у систему постоји више од једног мастера, пошто сваки мастер независно генерише властити сигнал такта на SCL линији?

- Сваки мастер генерише свој сигнал такта.
- Како синхронизовати сигнал такта на SCL линији?

Подсетимо се да IIC има уграђени механизаним који се назива жичано *И*, који важи за обе линије. Ово у пракси значи да мастер са најдужим нивоом "low" генерише "low" на SCL, док мастер са најкраћим нивоом "high" генерише "high" на SCL, као што је илустровано на слици 6.12.



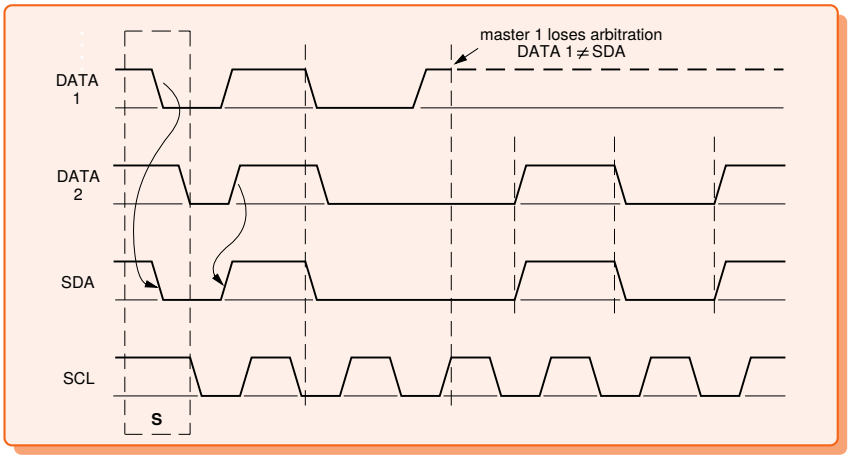
Слика 6.12: Пример синхронизације такта за систем са два мастера [5]

- Мастер са најдужим нивоом “low” генерише “low” на SCL.
- Мастер са најкраћим нивоом “high” генерише “high” на SCL.
- **wired-AND**

6.4.3 Арбитрација линије

Ако у комуникацији учествује више од једног мастера, поставља се питање: **Како одредити ком мастеру доделити SDA линију?**

И у овом случају се користи *wired-AND* особина, али сада на SDA линији, као што је приказано на слици 6.13.



Слика 6.13: Пример арбитраже у систему са два мастера [5]

У случају ако је магистрала неактивна (idle), било који мастер може послати старт и почети трансмисију. Мастер континуирано проверава стање SDA линије, тј. проверава да ли је послати бит постављен на линији. Пошто је низак ниво доминантан, мастер који уписује 1 а чита 0 препознаје да је магистрала заузета.

- Када је магистрала неактивна (idle), било који мастер може послати старт и почети трансмисију.
- Мастер проверава стање SDA линије (да ли је послати бит постављен на линији).
- Низак ниво је доминантан, мастер који уписује 1 и чита 0 препознаје да је магистрала заузета.

7 Меморија

У претходним поглављима смо више пута помињали појам меморије, од одређивања појма микроконтролера, преко манипулације операндима у аритметичко логичкој јединици, до употребе меморије за чување података и програма. Већ нам је познатно да је меморија намењена за складиштење – чување података и програма, и да се информације могу уписати и прочитати из меморије.

Са аспекта приступа меморијској ћелији разликују се меморијски модули са:

- секвецијалним (серијским) приступом,
- цикличним (периодичним) приступом,
- случајним – произвољним приступом,
- асоцијативним приступом.

Са аспекта могућности промене садржаја меморијске локације меморије се могу поделити на:

- променљиве меморије (нема ограничења у виду промене садржаја меморијске локације)
- полупроменљиве (садржај се не може мењати нормалним поступком, већ само специјалним методама)
- сталне меморије (садржај се формира у току процеса производње и ни под којим условима се не може мењати)

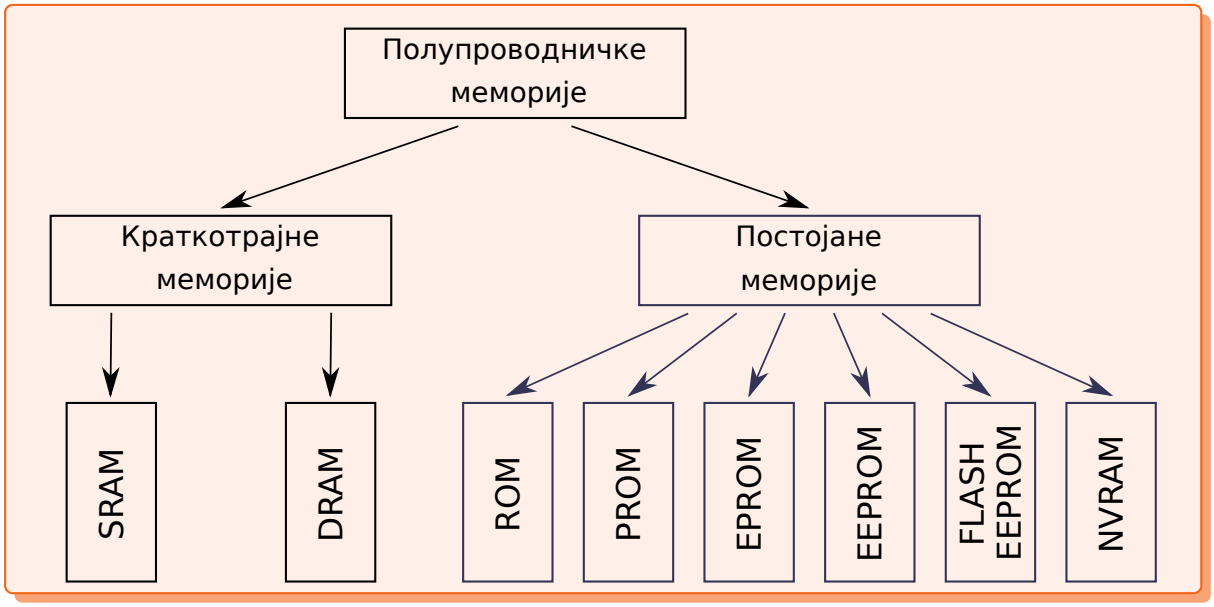
Латенца је мера кашњења од тренутка задавања задатка до тренутка када је податак читљив. То је функција највећег опсега, времена почињавања (lead off time) и мешања између више уређаја.

Пропусност меморије је мера њене брзине по којој податак може да се преноси са, и на меморију и обично се изражава у мегабајтима по секунди (MB/sec)

Зависно од медијума на коју се информација памти најчешће се користе полупроводничке (flash, SSD), меморије са магнетном површином (flopy, HDD) и меморије које користе оптичку технологију (CD, DVD).

7.1 Трајност складиштења података

Познато је да микропроцесори користе полупроводничке меморије, па ћемо им посцетити посебну пажњу. Једна од подела полупроводничких меморија је дата на слици 7.1.



Слика 7.1: Подела полупроводничких меморија према сталности ускладиштења података

Краткотрајна меморија (енг. Volatile Memory) чува садржај само док процесор ради. Време приступа је реда неколико наносекунди. Постојане меморије (енг. Non-volatile Memory) чувају садржај независно од рада процесора.

7.2 Краткотрајне полупроводничке меморије

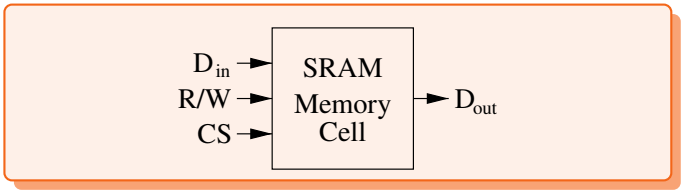
Познате су још и као меморије са произвољним приступом - RAM (Random Access Memory) и њихов садржај се губи са искључењем напајања. Намењене су за привремено меморисање података у току рада микропроцесора. Произвољан присуп значи да се може приступити било којој меморијској локацији. Краткотрајне полупроводничке меморије могу бити статичке (SRAM - Static RAM) или динамичке (DRAM - Dynamic RAM). Статичка меморија чува садржај све док има напајање, док динамичка захтева повремено освежавање садржаја, без кога би време чувања података било реда неколико милисекунди.

7.2.1 SRAM

SRAM је мерорија чија се основна меморијска ћелија састоји из флип-флопа (слика 7.2) који је реализован помоћу 6 транзистора у MOSFET или CMOS технологији. У односу на DRAM је карактерише велика брзина рада, али истовремено и мали капацитет у поређењу са површином коју заузима на чипу. Данас се најчешће употребљава као спрега између процесора и радне меморије.

SRAM

- прва краткотрајна меморија широке примене
- чини је низ ћелија, од којих свака меморише 1 бит
- ћелију чини један флип–флоп

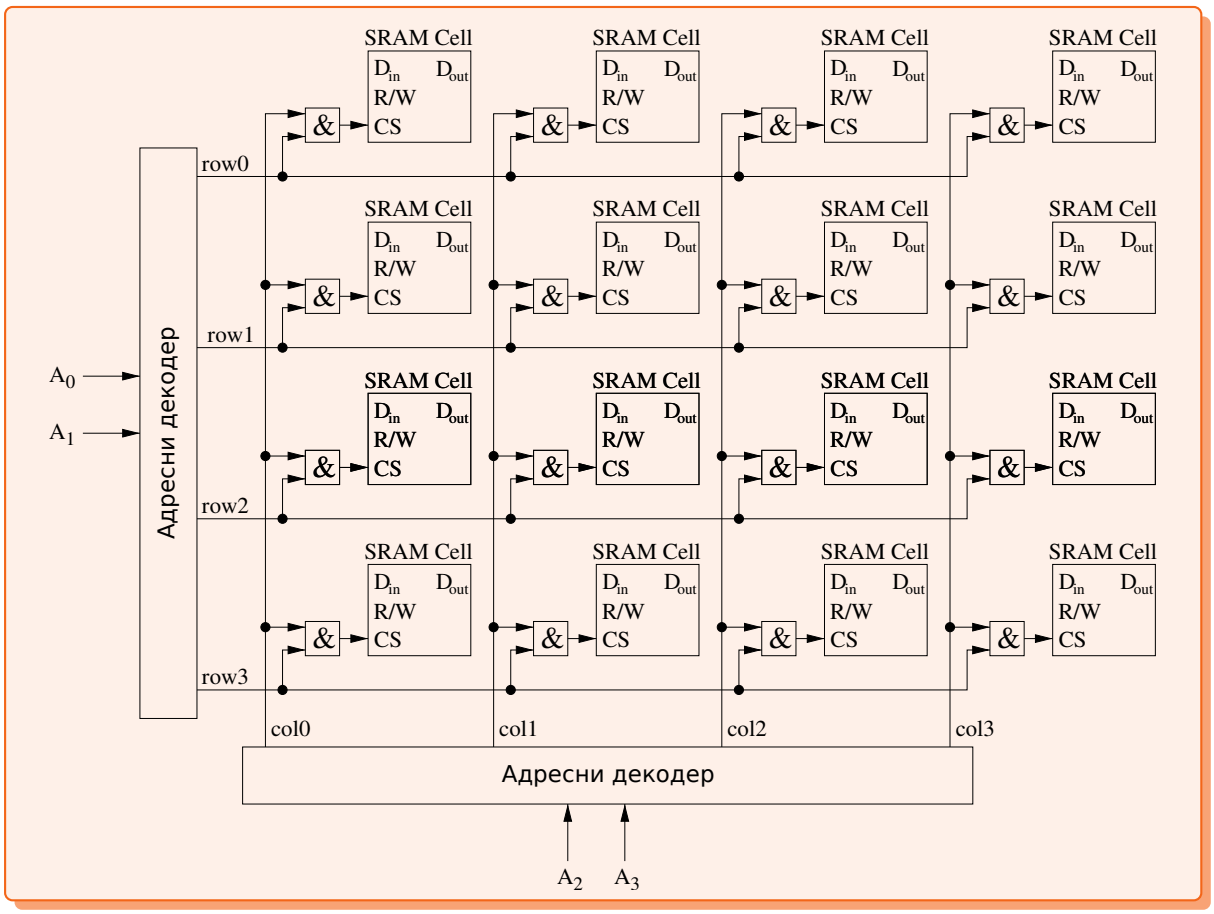


Слика 7.2: Једна меморијска ћелија SRAM [3]

Објашњења сигнала меморијске ћелије са слике 7.2:

Data In На улазу D_{in} ћелија прихвата 1 бит за складиштење.
Data Out D_{out} Заузима стање запамћеног бита.
Read/Write R/W Логичка 0 → дозвољен упис. Логичка 1 → омогућено читање.
Cell Select $CS = 0$ → онемогућен приступ ћелији, $CS = 1$ → омогућен приступ.

Меморијске ћелије се организују у матрицу (слика 7.2), а појединачној ћелији се приступа избором њене врсте и колоне.



Слика 7.3: Пример шеснаестобитне статичке меморије са адресним декодерима

Осим адреса на слици 7.3, SRAM има и додатне спољашње линије које служе за манипулацију подацима. Намена ових линија је слична као и код појединачне ћелије, а најчешће означавање је (слика 7.4):

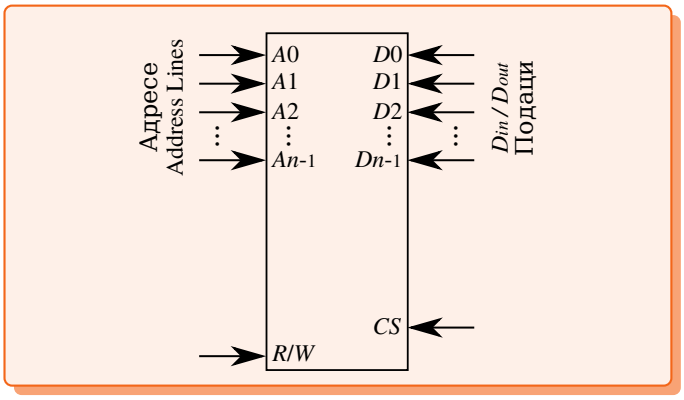
Address Lines $A_0 \ldots A_{n-1}$ Помоћу њих одабирамо једну од 2^n меморијских ћелија.

Data In (D_{in}) Има исту функцију како и код појединачне ћелије. За RAM са $n \geq 2$ има n линија.

Data Out (D_{out}) Има исту функцију како и код појединачне ћелије. За RAM са $n \geq 2$ је то магистрала.

Chip Select (CS) или *Chip Enable* (CE) Представља оно што је *Cell Select* код меморисјке ћелије.

Read/Write (R/W) Исто као и код меморисјке ћелије.



Слика 7.4: Спољашње везе SRAM

7.2.2 DRAM

DRAM (Dinamic RAM) је основна меморија сваког рачунара направљена да задовољи потребе за капацитетом (велика густина паковања) и нижом ценом. Меморијска ћелија се састоји од једног транзистора и једног кондензатора, за разлику од SRAM меморије која има шест или више транзистора. Један бит информације одговара стању напуњености једног кондензатора. Пошто је капацитет кондензатора мали (реда 10-13pF), а MOSFET и у непроводном стању има коначну отпорност, кондензатор се постепено празни и информација би била изгубљена после неколико милисекунди. Да се то не би десило DRAM се циклично "освежава", односно поново се уписује исти саджај у ћелију. Читање DRAM меморије је деструктивно, па се после читања једног реда мора извршити поновни упис, а тек онда прећи на следећи ред. Због своје густине паковања и цене коштања ова врста меморије се данас користи као основна радна меморија персоналних рачунара.

Постоји више врста ових меморија од којих су најзастуљенији SDRAM и DDR. SDRAM (Sinchronous DRAM) меморија се користила као основна радна меморија рачунара са брзинама такта до 133MHz. Њен наследник је DDR (Double Data Rate) меморија са бзином до 266MHz. Основна разлика је у томе што SDRAM обавља једну операцију током једног такта, док DDR обавља две (на растућу и опадајућу ивицу сигнала такта) чиме се теоретски постиже дупло већа брзина. Даље повећваање брзине такта, смањивање потрошње електричне енергије и повећавање капацитета меморијских модула доводи до конструкције нових меморијских елемената, као што су DDR2, DDR3 ...

7.3 Постојане полупроводничке меморије

7.3.1 ROM

ROM (Read Only Memory) је меморија са константним садржајем код које се специјалним поступком жељена информација може уписати само једанпут, након чега се ROM може само читати. Ово су сталне меморије општег типа, за смештање системских програма које пружају одређени ниво сигурности од случајних промена садржаја. Због високе цене припреме за производњу, овај тип меморије је погодан само за уградњу у уређаје који се производе у великим серијама.

7.3.2 PROM

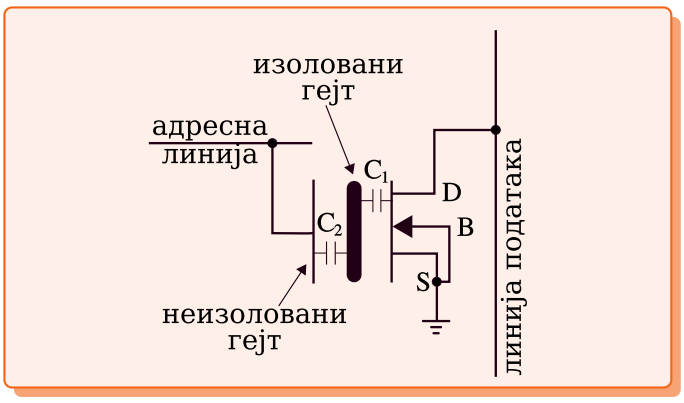
Коришћење ROM меморије је изузетно скупо у фазама пројектовања, тестирања или прављења прототипова. Програмирање сваког меморијског чипа засебно од стране самог програмера, коју нуди PROM (Programmabile Read Only Memory) је велики корак ка јефтинијем и бржем пројектовању и синтези нових уређаја.

PROM меморија је у ствари матрица меморијксих ћелија где свака ћелија има полупроводнички осигурач. Иницијално су сви осигурачи исправни и у све ћелије је уписана логичка јединица. Одабиром ћелије и применом кратког високонапонског импулса осигурач се уништава и ћелија сада као информацију садржи логичку нулу. Упис у целокупан меморијски елемент се изводи помоћу специјалног PROM програматора. Недостатак је што се једном уписани програм више не може мењати.

На тржишту су доступни OTP микроконтролери (One Time Programmable), који имају PROM као програмску меморију. Иако су микроконтролери са PROM или OTP меморијама погодни за производњу, нису исплативи за фазу пројектовања где је често потребно мењати садржај меморије и/или програма.

7.3.3 EPROM

За разлику од PROM елемента, код EPROM (Erasable PROM) елемената је могуће вишеструко брисање и уписивање садржаја. EPROM за свој рад користи MOSFET транзисторе са изолованим гејтом (слика 7.5). Сви транзистори у меморијској матрици имају по два гејта. Изоловани гејт је окружен савршеном изолацијом и са неизолованим гејтом преставља капацитивни разделник што је приказано на слици 7.5.



Слика 7.5: Једна EPROM ћелија меморијске матрица

Када меморијска ћелија није програмирана, напон логичке јединице на адресној линији је довољан да преко капацитивног разделника формира канал у MOS тра-

нзистору па је садржај локације нула. Програмирање ћелије се врши тако што се на адресну линију и линију података доведе висок напон (око 25V) који изазива велику струју дрејна. Ова струја ствара велико убрзање електрона који пробијају изолацију и акумулирају се на изолованом гејту. Сада је изоловани гејт на негативном потенцијалу (око -5V) па напон логичке јединице на адресној линији није довољан да формира канал у MOSFET транзистору и излаз ћелије је логичка јединица. Овако програмирана меморија не мења садржај бар 10 година. Међутим, ако се ћелија изложи дејству ултраљубичасте светлости око 20 минута, садржај се губи јер SiO_2 постаје слабо проводан и електрони напуштају изоловани гејт. Сваки EPROM чип има мали провидан прозор на врху кућишта који служи за брисање меморије, а кроз њега се може видети и унутрашњост чипа.

Меморија је много погоднија за употребу од обичне PROM али и даље захтева специјано светло (EPROM брисач) за брисање.

7.3.4 EEPROM

EEPROM (Electrically EPROM) такође користи MOSFET транзисторе са изолованим гејтом, али је отпорност изолације између електрода знатно мања. Садржај се уписује на исти начин као код EPROM меморије, али је напон програмирања мањи (око 10V). Меморија се брише електричним путем довођењем негативног напона на гејт. Најчешће се користе као програмске меморије у малим микропроцесорима или као меморијски модули за стално чување података о конфигурационим параметрима система. Производе се са серијским и паралелним интерфејсом.

7.3.5 FLASH EEPROM

Иако нам се може учинити да је EEPROM савршен избор када се тражи постојана меморија, њена висока цена је главни недостатак. FLASH меморија је конструисана као компромис. Она је у суштини надограња EEPROM-а где брисање података није омогућено за сваку појединачну меморисјку ћелију, већ се подаци могу брисати у великим блоковима или је чак могуће обрисати читав елемент одједном.

FLASH меморија се најчешће користи као програмска меморија. Број гарантованих уписа и брисања је много мањи у односу на класичне EEPROM, па се због тога не користи за складиштење података.

Напоменимо и да су FLASH меморије знатно јефтиније од класичних EEPROM-а.

7.3.6 NVRAM

Non-Volatile RAM (NVRAM), тј. постојана RAM меморија је осмишљена тако да комбинује предности RAM-а са једне стране и класичних постојаних меморија са друге стране. NVRAM се може конструисати на више начина, први је додавање батеријског напајања SRAM елементу, помоћу којег се садржај чува након искључења спољашњег напајања.

Друго решење је комбинација SRAM и EEPROM чипова у истом кућишту. После укључења подаци се из EEPROM-а копирају у SRAM и током рада микроконтролера подаци се и уписују и читају из SRAM-а. Када се напајање искључи, подаци се из EEPROM преписују у SRAM.

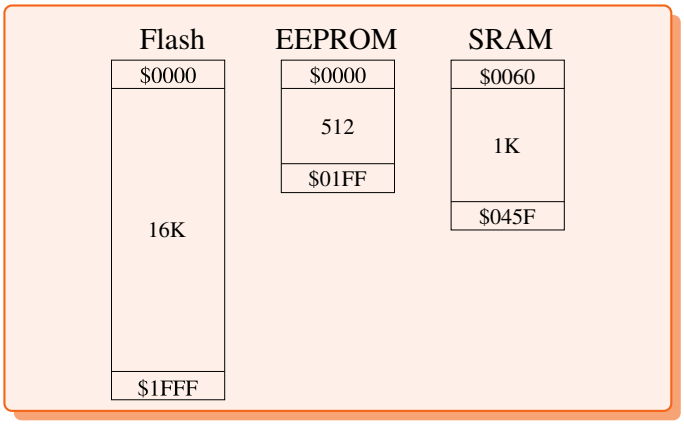
7.4 Организација меморије у микроконтролерима

Постоје два типа архитектуре меморије:

- HARVARD (RISC) и
- PRINCETON (von Neumann, CISC).

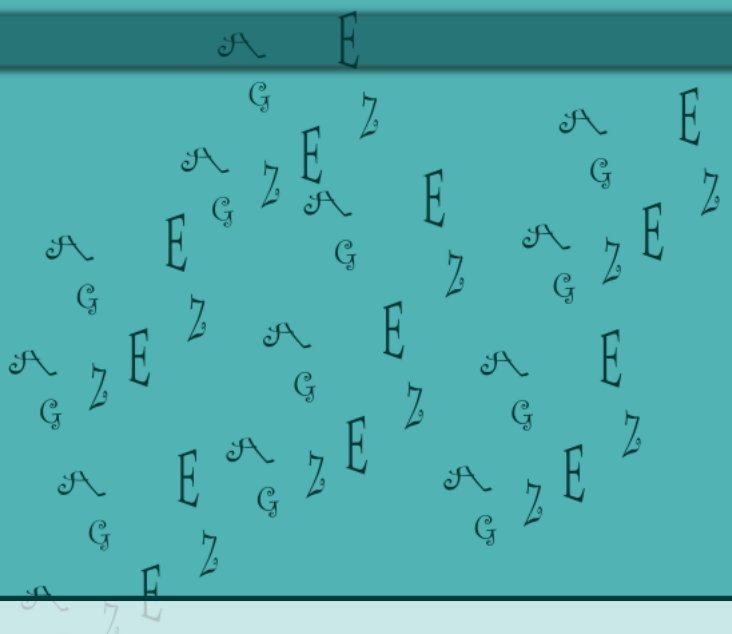
Код прве архитектуре су меморија за инструкције и меморија за податке физички раздвојене, док код друге један меморијски простор деле и инструкције и подаци. У циљу максимизације перформанси, AVR користи харвардску архитектуру са одвојеним меморијама и магистралама за програм и податке. Инструкције у програмској меморији се извршавају у различитим линијама. Док се једна инструкција извршава, следећа инструкција је долази из програмске меморије. Меморију ATмега16 контролера чине следеће целине (слика 7.6):

1. Програмска меморија FLASH, капацитета 16 килобајта.
2. Меморија података типа SRAM, капацитета 1 килобајт и
3. 512 бајтова EEPROM.



Слика 7.6: Организација меморије код Atmega16 микроконтролера.

Напоменимо да FLASH меморија садржи 8192 речи, што је једнако 16 килобајта меморијског простора. Адресе из опсега $0000 до $005F у SRAM чипу су резервисане за улазно-излазне и регистре опште намене.



Литература

[1] Atmel Corporation. Korisničko uputstvo za mikrokontrolere ATmega16 i ATmega16L, 2010. In English.

[2] Atmel Corporation. Atmel AVR 8- and 32-bit microcontrollers, Avgust 2014. http://www.atmel.com/products/microcontrollers/avr/.

[3] G. Gridling and B. Weiss. Introduction to microcontrollers. *Vienna University of Technology, Institute of Computer Engineering, Embedded Computing Systems Group, Version*, 1.4, 2007.

[4] Y. Koike. Flash A/D converter, Dec. 27 1988. US Patent 4,794,374.

[5] NXP Semiconductors . UM10204 I2C-bus specification and user manual. *User Manual*, Rev. 6, April 2014.