|  |  |
| --- | --- |
| etf_logo_skew_shapes copy.tif | **ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ**  **УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ** |

Пројектни задатак из предмета „Микропроцесорски системи 2“ ДИГИТАЛНИ ВОЛТМЕТАР

Студент: Небојша Марковић

Број индекса: 1241/11

Предметни наставник: др Златко Бундало

Предметни асистент: др Младен Кнежић

|  |  |
| --- | --- |
| Број бодова: |  |

# САДРЖАЈ :

# ДЕФИНИСАЊЕ ПРОЈЕКТА

Пројектовати дигитални волтметар коришћењем A/D конвертора MCP3204 и микроконтролера AT89C51/52. Вриједност измјереног напона потребно је приказивати сваке секунде на LCD дисплеју који ради у четверобитном моду.

Потребно је држати се праксе о прегледном писању кода, раздвајања дефиниције и имплементације функција, као и писања одговарајућих коментара. У извјештају потребно је приложити поступак пројектовања, блок шему, електричну шему, изворни програм, те резултате симулације и експерименталног тестирања. Такође, приложити списак коришћених компоненти и њихове основне функционалне спецификације. За симулацију је на располагању *Proteus*, а за тестирње развојно окружење Easy8051.

# ЗАХТЈЕВИ ЗА РЕАЛИЗАЦИЈУ ПРОЈЕКТА

## Увод у израду пројекта

Модули захтјевани задатком за реализацију дигиталног волтметра су:

* AT89C51/52 микроконтролер
* MCP3204 A/D конвертор
* LCD дисплеј у 4-битном моду

Но, међутим, да бисмо дошли до закључка шта све треба да има наш систем да бисмо на крају успјешно реализовали дигитални волтметар кренимо од најједноставнијих „**корисничких захтијева**“ који ће нам рећи ста је то што треба да реализујемо и шта све треба да посједује, те какве функционалности треба наш производ да омогући кориснику.

Затим, када се утврди шта тачно треба да реализујемо на ред долази следеће питање – „ Како то да реализујемо?“ - када састављамо „**системске захтјеве**“, на основу којих пројектујемо хардверски дио захтјеваног производа. На основу корисничких захтјева, утврђених у претходном кораку, закључујемо које компоненте треба да има систем који реализујемо, те интеракцију која треба да постоји између њих у циљу остваривања жељених функционалности коначног производа. У овом кораку треба преиспитати корисничке захтјеве и префрмулисати их ако за то постоји потреба. С обзиром да у току реализације и даљег развоја производа може доћи до непредвиђених компликација, као и промјене или допуњавања захтјева од стране корисника, препоручљиво је развој радити уз помоћ развојне плоче, уколико је могуће.

Након што је хардверско окружење познато прелази се на дефинисање „**софтверских захтјева**“, који у случају система који се састоји и од хардверског и од софтверског дијела треба јасно да дефинишу:

* које софтверске модуле треба развити
* какву интеракцију ти модули имају са хардвером
* какву интеракцију имају ти модули медјусобно
* слојевитост софтверског програма, као и евентуално (ако је познато)
* библиотеке које ће бити развијене

Тек сада на ред долази пописивање функција које ће бити потребно развити за реализацију пројекта.

Упоредо са утврђивањем и писањем захтјева на сваком нивоу пројектовања, потребно је описати и тестове, такође на одговарајућем нивоу, који ће потврдити да су жељени захтјеви испуњени. Сем „**софтверских тестова**“, који служе да би се потврдило да су сви захтјеви наручиоца испуњени, у овом извјештају тестирање и тестови неће бити описани јер није тражено у дефиницији задатка, а одузело би много времена.

## Кориснички захтјеви

Када помислимо на дигитални волтметар прво што закључујемо јесте да је то дигитални уређај који мјери напон, али из само тога не можемо извући све функционалности које он треба да има, него се морамо поставити у улогу корисника и размислити шта све треба да има један волтметар.

На овај начин долазимо до корисничких захтјева, који гласе:

|  |  |
| --- | --- |
| **Ознака** | **Опис захтјева** |
| **К.З.1** | Волтметар има сонде за мјерење напона (или прикључке за њих) |
| **К.З.2** | Измјерена вриједнст напона се приказује на дисплеју |
| **К.З.3** | Вриједност која се приказује се update-ује сваке секунде и поново исписује |
| **К.З.4** | Након што се волтметар упали пали се и сигнална ламица која показује да је упаљен и спреман за кориштење |
| **К.З.5** | Аутоматски подешава скалу у којој се приказује мјерена вриједност у зависности од висине измјерене вриједности напона |
| **К.З.6** | Има могућност ручног избора скале |

*Таб. 2.1. Кориснички захтјеви*

Тестови којипокривају ове захтјеве су интуитивно јасни – потребно је тестирати готов производ (на овом нивоу никако не хрдвер или софтвер посебно, или неки од модула, него директно захтјев онакав какав је постављен).

Из *табеле 2.1.* коју смо добили анализом како захтјева корисника, тако и анализом функционалности које волтметар треба да има и ако нису директно наведене кроз захтјеве корисника, добијамо детаљнију слику шта све производ који правимо треба да има/задовољава, али не и детаље реализације производа. Управо следећим кораком се приближавамо почетку реализације производа. Попишимо шта нам све од хардверских компонената/модула треба и какве везе треба да постоје међу њима да би се могле постићи горе описане функционалности. Приступ писању системских захтјева је следећи корак на путу до циља и биће мало детаљније урађен и образложен у односу на писање корисничких захтјева.

## Системски захтјеви

У овом кораку анализираћемо хрдверске компоненте које наш систем треба да има (на нивоу модула за потребе овог рада – у право пројектовање хардвера се нећемо упуштати због обима посла који превазилази обим једног пројектног задатка), те везе и интеракцију које треба да постоје између њих.

Ако треба да пројектујемо дигитални волтметар, а знамо да је напон аналогни сигнал. Да би рад дигиталног система био могућ овдје треба да улазну величину прилагодимо за даљу обраду у систему. На овом мјесту се мжемо одлучити како за А/Д конвертор, тако и за нпр. напонски сензор који као излазну величину даје сигнал чија фреквенција зависи од висине улазног напона. Како је задатком директно дефинисано да се користи А/Д конвертор MCP3204 то ћемо узети и као прву хардверску компоненту која је позната.

Даље, знамо да мјерене величине мора да буде могуће приказати на дисплеју и задатком је дефинисано да то буде LCD дисплеј који ради у четверобитном моду. Дисплеј који може да задовољи овај захтјев је нпр. Hitachi-јев HD44780U, али није наведено да мора овај дисплеј да се користи, па тако може да се изабере и било који други који може да ради у 4-битном моду.

Прије самог исписа на дисплеј информација која долази са А/Д конввертора мора да се обради и прилагоди за испис на дислеју. Због, такође, саме поставке задатка гдје је речено да на располагању имамо микроконтролер AT89C51/52, изабрати микроконтролер за ову обраду – нека то буде нпр. баш АТ89S8253.

Да би кориснички захтеви били испуњени потрени су нам још тастери за ручну промјену скале и сигнална диода од хардверских компонената, док се аутоматско подешавање скале и освјежавање дисплеја сваке секунде може ријешити софтверски.

Да би ово могло бити функционално потребно је још да знамо везе између изабраних/предложених компонената, испланирамо размјену података, интеракцију и контролу у систему.

Не упуштајући се дубље у анализу шта како и зашто смо тако поставили (и ако то треба да се уради када се приступа пројектовању испочетка – овдје ћемо рећи да се до предложене шеме дошло консултујући datasheet-ове одаабраних компонената), даћемо предлог како наш склоп треба да изгледа (*Сл.2.1. Предложена хардверса архитектура система*).

Оно што је сад већ очигледно да смо заборавили напоменути (мада је интуитивно јасно ипак мора да буде наведено као захтјев) јесте да нам за исправан рад склопа треба како стабилан такт сигнал, те напон напајање Vcc и референтни напон(потребан за рад А/Д конвертора).

Ако је већ јасно да је неке корисничке захтјеве немогуће испунити или ако видимо да смо направили пропусте при пројектовању система, сада је право вријеме да се то поправи. До ријешења се ријетко када долази из првог покушаја. Уобичајно је да се грешке и пропусти уоче тек на следећим корацима израде проекта, па се тако сви захтјеви и ријешења надопуњују и мијењају све до краја пројекта(мада је наравно добро када постоји што мање потребе за тим).



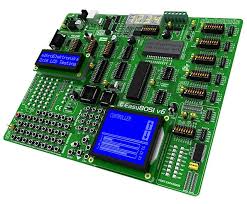
*Сл. 2.1. Предложена хардверса архитектура система*

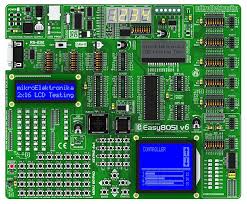
Оваквом анализом долазимо од системских захтјева које треба да наш систем испуњава (*Табела 2.2*) до предлога изгледа система приказаног на слици изнад.

|  |  |
| --- | --- |
| **Ознака** | **Опис захтјева** |
| **С.З.1** | Претварање аналогног напона у дигиталну величину погодну за обраду |
| **С.З.2** | Слање вриједности микроконтролеру на обраду (предложен SPI) протокол |
| **С.З.3** | Могућност управљања сигналном LED диодом |
| **С.З.4** | Могућност конфигурације и управљања дисплејом који ради у 4-битном моду |
| **С.З.5** | Тастери за избор скале и веза са системом за прилагођење података – микроконтролером |
| **С.З.6** | Стабилан такт сигнал |
| **С.З.7** | Напон напајања Vcc |
| **С.З.8** | Стабилан референтни напон |

*Таб. 2.1. Кориснички захтјеви*

Поређењем предложеног система са расположивим развојним плочама закључујемо да се овакав систем може добити кориштењем развојне плоче „Easy8051 v6“ (*Сл.2.2. Easy8051 v6*), произвођача „Микроелектроника“, па је дата плоча и изабрана за даљи развој ројекта.





*Сл. 2.2.* *Easy8051 v6*

### *Спецификација хардвера*

* **Mикроконтролер AT89S8253**

AT89S8253 је 8-битни микроконтролер високих перформанси, који је израђен у CMOS технологији, а има следеће карактеристике:

* Припада фамилији микроконтролера 8051
* 12 kB флеш меморије
* 2 kB EEPROM меморије
* 256 B интерног е-а за смијештање промјенљивих
* Напон напајања 4-6 [V]
* Радна фреквенција до 24 [MHz]
* 32 улазно-излазне линије распоређене у 4 порта са по 8 пинова
* Три 16-битна тајмера/бројача
* 9 извора прекида
* Прграмабилну UART сеијску комуникацију
* Програмабилни Watch Dog тајмер
* 2 додатна мода рада за смањење потрошње (Idle и Power-down мод)
* Тростепену заштиту уписаног програма

Приказаћемо пинове микроконтролра и њихов распоред у циљу описивања самог контролера и функције његових пинова (*Сл. 2.3. Распоред пинова микроконтролера АТ89Ѕ8253*) .



*Сл. 2.3 Распоред пинова микроконтролера АТ89Ѕ8253*

* Vcc – напон напајања (4-6V).
* GND – маса
* Порт 0 (P0.0 – P0.7)– Када се дефинишу као излази, на сваки од ових пинова може се прикључити до 8 ТТЛ улазних кола. Када се одреди да раде као улази, тада су то улази бесконачне унутрашње отпорности - односно такви улази се понашају као да „висе у ваздуху“. Ако се користи додатна меморија, на овим изводима се наизмјенично појављују подаци и адресе (А0-А7) намјењени за приступ овом додатном меморијском чипу. Сигнал на ALE пину, притом, одређује шта ће и у ком тренутку бити прослијеђено на порт
* Порт 2 (P2.0 – P2.7) – Када се ови изводи користе као улази или излази, њихове особине су по свему сличне порту P1. Ако се користи спољна меморија, овде ће се појавити виши адресни бајт (А8-А15) за адресирање додатног чипа.
* Порт 3 (P3.0 – P3.7) – Као и на порту P1, изводи се и овде користе као универзални улази или излази али имају и неку додатну улогу
* RST – позитиван напонски импулс на овом пину ресетује микроконтролер.
* ALE/PROG – У нормалном раду, овај пин емитује поворку импулса чија је фреквенција једнака 1/6 фреквенције главног осцилатора. Ако се користи додатна меморија, сигнал на овом пину управља додатним регистром у којем се привремено смјешта нижи адресни бајт (А0-А7). За време уписивања програма у микроконтролер, овај пин такође служи као контролни улаз.
* PSEN – Сигнал са овог пина користи се за читање спољне програмске ROM меморије.
* EA/VPP – Када се овај пин споји на потенцијал масе, микроконтролер ће узимати програмске инструкције из спољне програмске меморије.
* XTAL1 – Улаз интерног осцилатора.
* XTAL2 – Излаз интерног осцилатора.

**ROM меморија** је капацитета 12 кB и изведена је у FLASH технологији, што омогућава велики број уписа/брисања садржаја меморије. Програмирање се врши преко уграђеног SPI модула (Serial Peripheral Interface). Могуће је да се ROM „догради“ спољним чипом.

**RAM меморија** се састоји од 3 блока са по 128 регистара чија структура одговара 8051 стандарду:

* 128 регистара опште намјене
* 128 меморијских локација резервисаних за SFR регистре. И ако су само неке од њих стварно искориштене, празне локације не треба да се користе за смјештање промјенљивих.
* 128 додатних регистара који су слободни за употребу (немају неку посебну намјену). Пошто имају исте адресе као и SFR регистри њима се приступа индиректним путем.

**EEPROM меморија** - Ово је посебна врста меморије која има осбине и RAM и ROM меморије - подаци се уписују и бришу у току рада али остају сачувани и после нестанка напона напајања. Овај микроконтролер има укупно 2 кB (2048 локација) EEPROM меморије.

* **LCD дисплеј - Hitachi HD44780**

LCD дисплеј служи за исписивање порука. Садржи контролер HD44780 фирме Hitachi. Састоји се из 2 реда са по 16 линија у којима се исписују карактери док се свако од ових поља састоји од матрице величине 5х8 пиксела као што је приказано на слици (*Сл.2.4. LCD дисплеј*)



*Сл. 2.4 LCD дисплеј*

Могучности дисплеја су: приказује сва слова абецеде, грчка слова, знакове интерпукције,

математичке симболе итд. Поред тога могуће је приказати и знакове које корисник сам дефинише. Такође посједује и аутоматско помјерање порука преко екрана (шифтовање у лијево и удесно), појављивање курсора, позадинско освјетљење и слично. Контраст на екрану зависи од напона напајања и од тога да ли се поруке исписују у једном или два реда. Због тога се на извод означен са Vee прикључује променљиви напон од 0 – Vdd (обично се за ово користи тример потенциометар).

Унутар дисплеја се налазе 3 меморијска блока:

* DDRAM – Display Data RAM,
* CGROM – Цхарацтер Генератор ROM,
* CGRAM – Цхарацтер Генератор RAM.

**DDRAM меморија** - У ову меморију смјештају се карактери који треба да буду приказани на дисплеју. Величина ове меморије је довољна за смјештање 80 знакова, а један дио ових локација има директне везе са пољима на екрану.

**CGROM меморија** - У ову меморију фабрички је уписана мапа са изгледом свих карактера које дисплеј може да прикаже - сваком карактеру одговара једна меморијска локација. Адресе меморијских локација ЦГРОМ-а се поклапају са стандардним ASCII вриједностима карактера. То значи да, уколико се у програму који микроконтролер извршава захтјева исписивање неког карактера, на излазу ће се појавити бинарна вриједност карактера. Када се бинарни број учита у LCD дисплеј, приказаће се симбол који се налази на меморијској локацији која одговара бинарном броју у CGROMU-у. Ово важи за сва слова абецеде (велика и мала) али не и за бројеве.

**CGRAM меморија** - Поред тога што приказује све стандардне карактере, LCD дисплеј може да прикаже и ознаке које корисник сâм дефинише. Тиме је омогућено исписивање ћириличних фонтова као и других симбола који стају у оквир величине 5х8 пиксела. Све то омогућава мала RAM меморија (CGRAM) величине 64 бајта. Величина регистра у овој меморији је 8 бита али се користи само нижих 5. Логичка јединица у сваком регистру представља затамњен пиксел, док осам локација узетих

заједно предтављају један знак. Симболи се дефинишу на почетку програма а њихово приказивање се врши позивањем адресе (прва колона у CGROM мапи).

* **Команде, повезивање и иницијализација LCD дисплеја**

Сви подаци који се преносе на LCD дисплеј преко извода D0 – D7 биће тумачени као команде или подаци у зависности од логичког стања на пину RS - RS = 1 – Битови D0 – D7 су адресе карактера који треба да се прикажу на дисплеју. Уграђени процесор адресира уграђену мапу карактера и приказује одговарајући знак. Место појављивања је одређено DDRAM адресом. RS = 0 – Битови D0 – D7 су команде које одређују начин рада дисплеја. Списак свих команди које LCD дисплеј подржава су наведене у табели:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда | RS | R/W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Обриши дисплеј | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Курсор на почетак | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Х |
| Унос карактера | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S |
| On/off | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | U | B |
| Помјерање | 0 | 0 | 0 | 0 | 0 | 1 | D/C | R/L | X | X |
| Начин рада | 0 | 0 | 0 | 0 | 1 | DL | N | F | X | X |
| CGRAM адреса | 0 | 0 | 0 | 1 | CGRAM адреса | | | | | |
| DDRAM адреса | 0 | 0 | 1 | DDRAM адреса | | | | | | |
| Читање BUSY fleg-а (BF) | 0 | 1 | BF | DDRAM адреса | | | | | | |
| Упис у CGRAM или DDRAM | 1 | 0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Читање из CGRAM /DDRAM-a | 1 | 1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

*Таб. 2.2. Команде за LCD дисплеј*

I/D 1 = Инкремент

0 = Декремент

S 1 = Укључен шифт регистра

0 = Искључен шифт регистра

D 1 = Укључен дисплеј

0 = Искључен дисплеј

U 1 = Видљива линија курсора

0 = Без линије курсора

B 1 = Курсор трепери

0 = Курсор не трепери

R/L 1 = Помjерање удесно

0 = Померање улијево

DL 1 = 8-битно повезивање

0 = 4-битно повезивање

N 1 = Испис у две линије

0 = Испис у једној линији

F 1 = Формат 5x10

0 = Формат 5x8

D/C 1 = Померање дисплеја

0 = Померање курсора

У 4-битном моду рада дисплеја за комуникацију се користе само 4 виша бита (D4 – D7) док се осталимогу оставити неповезани. У том случају, сваки податак се LCD-у шаље у два корака: прво се шаљу 4 виша бита а затим се шаљу 4 нижабита.

Одмах након довоñења напона напајања LCD дисплеј се аутоматски ресетује. Након тога дисплеј је спреман за рад и налази се у фабрички подешеном начину рада што значи:

1. Дисплеј је обрисан;

2. Начин рада:

DL = 1, комуникација се изводи 8-битном везом;

N = 0, поруке се исписују у једном реду;

F = 0, користе се фонтови формата 5џ8 тачака;

3. Укључено/искључено:

D = 0, дисплеј је искључен;

U = 0, не види се линија курсора;

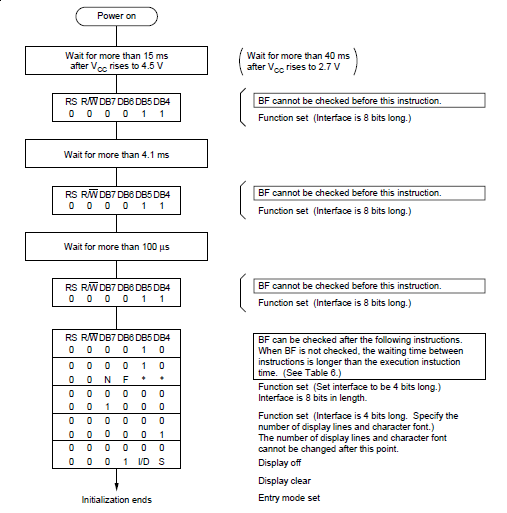
B = 0, искључено је блинкање курсора;

4. Унос карактера:

ID = 1, адресе на екрану се аутоматски увећавају за 1

S = 0 ,искључено померање (shift)

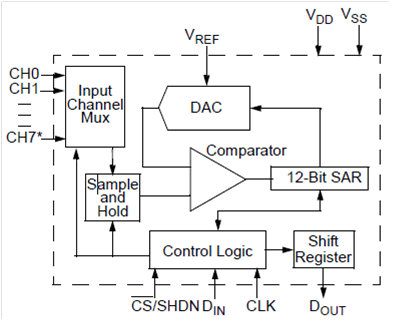
Алгоритам по коме се обавља иницијализација у случају 4-битног повезивања приказан је на слици:



*Сл. 2.5 Алгоритам иницијализације LCD-а*

* **MCP3204 А/Д конвертор**

MCP3204 је 12-битни А/Д конвертор са сукцесивном апроксимацијом (*Сл.2.3. Блок шема MCP3204 А/Д конвертора*).

****

*Сл. 2.6 Блок шема MCP3204 А/Д конвертора*

Приказани А/Д конвертор чине:

* Мултиплексер за улазне канале
* Sample and Hold коло
* Д/А конвертор
* Компаратор
* 12-битни SAR (Successive Aproximatio Register)
* Помјерачки регистар
* Контролна логика

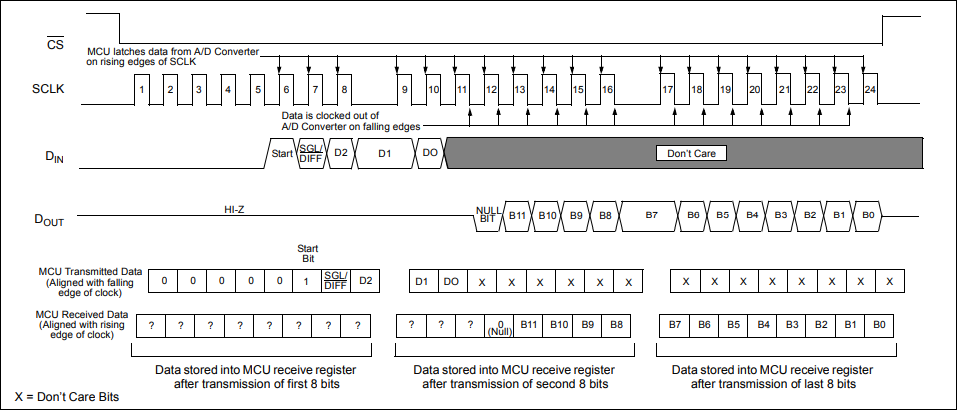
Основне карактеристике овог конвертора су:

* MCP3204 посједује четири канала који се могу користити као четири независна канала или као два диференцијална пара.
* Комуникација са А/Д конверртором остварује се кориштењем серијског интерфејса а са SPI протоколом.
* Референтни напон одређује аналогни улазни опсег. Вриједност бита најмање вриједности (LSB) директно је пропорционална са вриједношћу референтонг напона.
* Дигитални излазни код добија се као функција референтног напона и мјерене вриједности напона као:

Комуникација са MCP3204 А/Д конвертором се обавља помоћу стандардног SPI компатибилног серијског интерфејса. Иницијализација комуникације са било којим уређајем се врши довођењем логичке нуле на пин CS. Први примљени такт импулс, заједно са сигналом CS на нули и сигналом Din на јединици, генерисаће стартни бит за серијску комуникацију. Затим, иза стартног бита следи SGL/DIFF бит и њиме се одређује да ли ће се за конверзију користити један канал у независном или два канала у диференцијалном моду рада. Наредна три бита (D2, D1 и D0) одређују одговарајућу конфигурацију улазних канала при чему вредност бита D2није битна за овај А/Д конвертор. Са семпловањем аналогног сигнала почиње се на четвртој растућој ивици тактног сигнала након проласка стартног бита а завршетак семплавања је на петој опадајућој ивици такта. За потребе овог задатка користи се канал CH0 у независном моду рада (за детаље о раду овог А/Д конвертра консултујте [1])

Након проласка бита D0 потребно је да проñе још један такт да би завршио sample and hold. На опадајућој ивици наредног такта на Dout излазу се јавља NULL бит. Наредних 12 битa на излазу Dout означавају резултат конверзије при чему се први јавља MSB бит (Most Significant Bit). Подаци се на излаз увijек шаљу на опадајућој ивици такта.

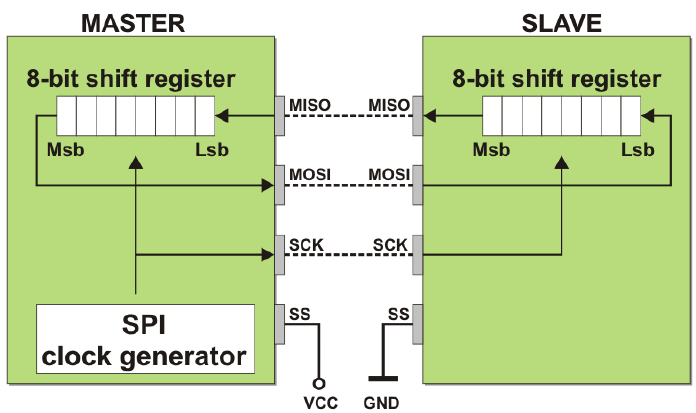
У комуникацији са микроконтролерима обично се шаљу и примају 3 бајта. Такође, потребно је да SPI интерфејс микроконтролера буде подешен тако да тактује податке на опадајућој ивици а да их лечује на растућој. На слици *Сл. 2.4*  приказана је комуникација између микроконтролера и А/Д конвертора у SPI моду 0,0 - режиму рада за који је потребно да сигнал SCLK буде на логичкој нули, те да се пребаци на логичку јединицу при старту комуникације.



*Сл. 2.7 SPI комуникација кориштењем 8-битних сегмената (mode = 0,0 => SCLK на ниском логичком нивоу)*

* **SPI интерфејс**

SPI je систем који омогућава брзу синхрону везу измеñу контролера и једног или више периферијских уређаја који подржавају SPI. Код овакве везе увијек је један уређај главни (мастер). Он одређује брзину, смјер преноса података (да ли се они шаљу или примају) и формат података. На другом крају налази се “slave”, који је у подређеном положају, што значи да не може покренути размјену података и мора да се прилагођава мастер страни. На слици *Сл. 2.6*  приказан је SPI систем.



*Сл. 2.8 SPI модул*

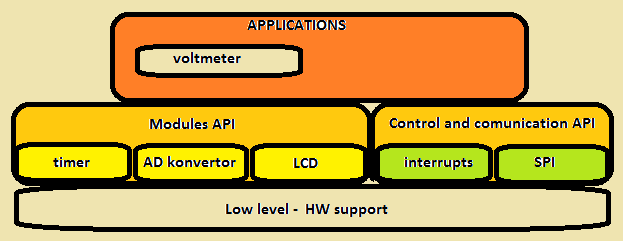
Подаци се преносе full-duplex везом помоћу 3 проводника који се повезују са изводима MISO, MOSI и SCK (P1.6 , P1.5 и P1.7 пинови микроконтролера у нашем случају). Четврти конторлни пин SS се на страни мастера не користи и може се бити искоришћен као улаз/излаз док на страни слејва мора бити на ниском логичком нивоу. Ако се на SS извод слејва доведе сигнал логичке јединице његов SPI систем се деактивира и MOSI пин може да се искористи као улаз за неку другу намjену.

## Софтверсо рјешење

Када смо задовољили системске захтјеве и на основу њих испројектовали хардверски дио, а с обзиром да наш систем садржи микроконтролер, на ред долази да се обезбједи софтверска подршка.

Вратимо се опет на корисничке захтјеве и погледајмо како се они односе на софтвер који треба развити. Речено је да треба мјерени резултат да се исписује на дисплеју, да скала може ручно да се мијења, да се садржај дисплеја освјежава сваке секунде. На основу овога можемо закључити да требамо обезбједити иницијализацију, управљање свим модулима, те комуникацију између одоварајућих модула. Потребно је осмислити софтверску архитектуру – tj. софтверске слојеве, модуле, односе и комуникацију између њих, па на крају и функције које нам требају да би све функционисало.

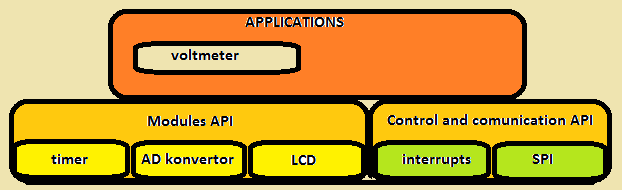
С обзиром да се за реализацију волтметра користе модули који се могу користити и за реализацију много других апликација, а не само за волтметар, циљ је да се обезбједи добра софтверска подршка за рад са било којим од модула, а онда да се за било коју нову апликацију развијају функције које су специфичне баш за ту апликацију. Тако је првобитно планирано да се софтвер развије у три слоја (*Сл. Првобитно предложжена софтверска архитектура*).



*Сл. 2.9 Првобитно предложжена софтверска архитектура*

Међутим, и ако гледано са стране даљег развоја и надоградње овако организованог софтвера, кроз фазу тестирања показало се као непрактично рјешење за изабрану хардверску архитектуру. Наиме, добре стране овакве организације су те да ако додамо нови дисплеј, тимер или А/Д конвертор довољно би било да обезбједимо основне функције за рад са њима (чији је прототип већ установљен) те би се оне само позивале у Модул АПИ функцијама (које већ постоје), те би врло лако било замјенити било који од модула које се користи за нашу апликацију и сама замјена не би уопште утицала на саме апликације или на апликативни слој за рад са модулима, као ни на комуникационе библиотеке. Тек када би се додавао потпуно нови модул, тада би било неопходно развити функције и на апликативном и на хардверском нивоу, док би за сваку промјену модула или оптимизацију постојећих функција за рад са њим само функције HW support слоја требало мијењати, а за развој апликација имали би стандардизоване Modul API функције на које смо навикли, те би развој апликација уз кориштење истих био знатно олакшан.

Међутим, показало се да је при раду са одабраним микроконтролером непрактично развијати софтвер који има три нивоа. При раду са датим контролером имамо могућност избора три различита меморијска модела: small, compact и large. Small модел је модел који све промјенљиве, као и све остале податке, аутоматски смјешта у интеру меморију којој се има директан приступ (користи се директно адресирање). Small модел и јесте најбржи модел који користи интерну меморију микроконтролера и то је модел који желимо користити (како због брзине, тако и због тога што екстерну меморију нисмо предвидјели при хардверском планирању ). Критична тачка овдје јесте позив функције из функције гдје се RAM меморија брзо „троши“, па да би заобишли проблеме који се ту јављају морали смо избацити слој за хардверску подршку и у склопу Module API функција реализовати и дио који пружа хардверску подршку (*Сл.2.10.* *Коначна софтверска архитектура*)



*Сл. 2.10 Коначна софтверска архитектура*

Базираћемо се само на развијену апликацију, тј. волтметар. Волтметар треба да може да измјери напон и прикаже измјерену вриједност, што му је примарна функција. Поред најосновнинјих функционалности потребно омогућити још неке, као што су промјена слале у којој се мјерена вриједност приказује или сигнална лампица која означава да је волтметар укључен и спреман за рад, у циљу да уређај буде једноставан за кориштење и да пружи кориснику могућности које нису директно захтијеване, али су примамљиве за корисника.

Тако би APPLICATIONS слој садржао фајлове са декларацијама и дефиницијама функција везаних за нашу апликацију, те филе у коме је main програм који позива наведене функције. Добра пракса јесте да се у главном програму апликације користе само функције дефинисане на APPLICATIONS нивоу.

Главни програм за волтметар апликацију имао би следећи изглед:

int main()

{

/\* mesuredVal is variable used to accept mesured val sent from ADC \*/

float idata mesuredVal;

static unsigned long temp;

static unsigned long idata count;

/\* Initialise all modules used for voltmeter \*/

initVoltmeter();

while(1)

{

/\* do the mesuring \*/

/\* if REFRESH\_TIME is passed since last display refres - refresh the disp\*/

mesuredVal = mesProccess(&scale);

temp = milis();

if (1000 <= (temp - count))

{

count = temp;

dispVoltage(mesuredVal, scale);

}

}

return 0;

}

Видимо да је прво потребно извршити иницијализацију волтметра, а затим се у while обезбјеђује да се сво вријеме за које је инструмент укључен врши мјерење напона и освјежавање приказа на дисплеју сваке секунде.

Функција *mesProccess(&scale)* омогућавада се на основу измјерене вриједности аутоматски одреди и скала у којој се измјерена вриједност напона приказује. Као што је речено, скала у којој се приказује мјерење може и ручно да се бира притиском једног од два тастера која су планирана пројектом. Притиском на тастер дешава се интерапт којим се увећава или смањује вриједност глобалне промјенљиве scale. Одабрано је да се притиском на тастер покреће прекидна рутина из разлога што је промјена скале нешто што корисник захтјева при кориштењу инструмента и требало би да одзив инструмента буде брз при таквом захтјеву.

Функције специфичне за волтметар убачене су на ниво апликације кроз фајлове *voltmeter.h* и *voltmeter.c.* Функције специфичне за кориштене модуле, комуникацију или прекидне рутине дефинисане су и имплементиране кроз Modue API слој. Све развијене функције су при самој декларацији у коду детаљно описане, те неће бити детаљније објашњаване у овом раду.

# ЛИТЕРАТУРА

[2] MCP3204 datasheet: “ <http://pdf1.alldatasheet.com/datasheet-pdf/view/74936/MICROCHIP/MCP3204.html>”

* Бранко.Л.Докић.:“Енергетска електроника“,ЕТФ. БЛ. Мај 2007.