Факултет инжењерских наука
Крагујевац

Архитектура рачунарских система

Семинарски рад

Имплементација семафора на MSP430FG439

Студенти:
Ковачевић Јован
Обрадовић Данило

Професор:
др Александар Пеулић

12. фебруар 2017.

Садржај

1 Увод

У овом раду описаћемо како се преко MSP430FG439-а може конструисати семафор, архитектуру поменутог микроконтролера, начин функционисања уређаја као и његову примену у системима саобраћаја. За реализацију овог пројекта, користили смо MSP430FG439, три диоде, MSP-FET Emulation tool помоћу ког смо микроконтролер MSP430FG439 повезали са рачунаром ради програмирања. Највише ћемо се усресредити на описивање архитектуре MSP430FG439, јер, било би превише за један рад описивати и архитектуру MSP-FET Emulation tool-а.

2 Архитектура MSP430FG439 микроконтролера

Микроконтролер MSP430FG439 базиран је на 16-bit RISC процесору. Инкорпорира флексибилни систем унутрашњих сатова који су међусобно повезани фон Нојмановим заједничким меморијским адресним bus-ом и меморијским data bus-ом. Опремљен модуларно меморијски-мапираним аналогно/дигиталним периферијама, MSP430FG439 представља одличан избор у разним захтевним апликацијама.

– Главне карактеристике MSP430xx4xx фамилије су :

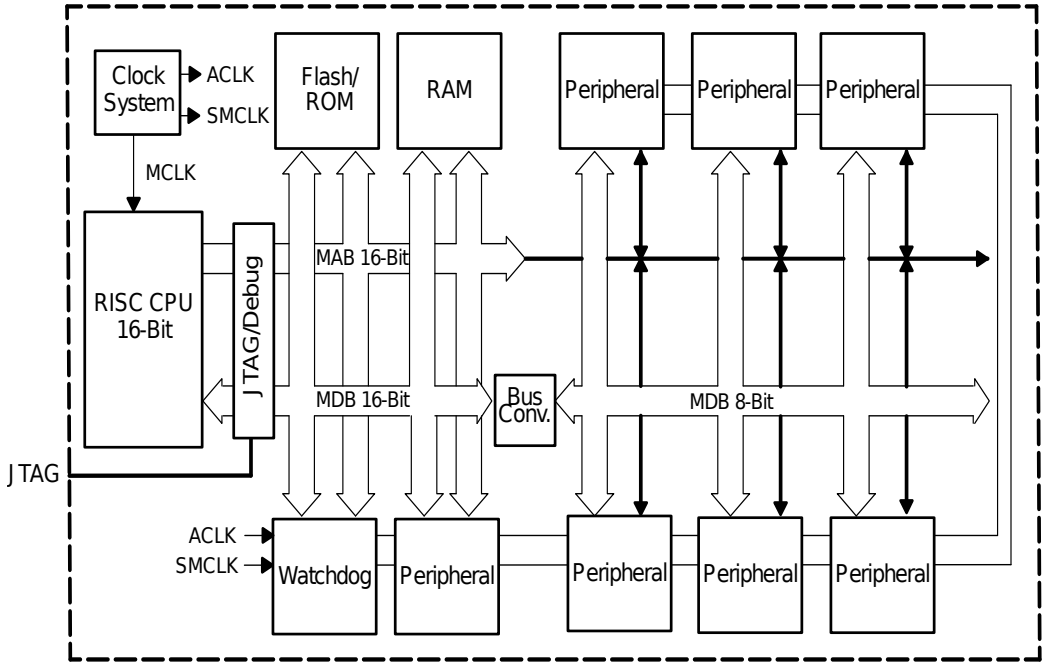
- Веома мала потрожња која продужава животни век батерије
  - $0.1\mu A$ оптерећење RAM-а
  - $0.8\mu A$ реално-временски clock мод
  - $250\mu A$ MIPS
- Фисоко перформантни аналогни интерфејс
  - 12-bit 10-bit ADC – 200 ksps, температурни сензор, V_{Ref}
  - 12-bit дуални DAC
  - Comparator-gated тајмери за мерење отпорничких елемената
  - Контролори напона
- 16-bit RISC CPU омогућава нове апликације коришћењен краћег кода
  - Велики регистри који елиминишу сметње при раду
  - Компактан дизајн језгра који смањује потрошњу енергије
  - Само 27 инстукција језгра и 7 адресних модова
  - Екстензивне могућности векторских прекида
- Системски програмабилни FLASH дозвољава флексибилну промену кода као и data logging

2.1 Систем флексибилних сатова

Сатни систем дизајниран је специјално за рад са уређајима за чији рад је неопходна батерија. Ниско фрекфентни помоћни сат (ACLK) директно се покреће преко 32kHz кристала. ACLK се може користити за позадинске реално-временске програме, тојест за функције самопобуде. Интегрисани брзи дигитално контролисани осцилатор (DCO) може да користи главни сат (MCLK) који користи CPU и периферије које раде на великим брзинама. По дизајну (DCU) је активан и стабилан за мање од $6\mu s$. Решења базирана на MSP430 системима ефективно користе 16-bit RISC процесор, тако да га оптимално оптерећују.

– Ниско фрекфентни помоћни сат = Standby mode

– Брзи главни сат = Високо перформантни мод за обраду сигнала



2.2 Уграђена емулација

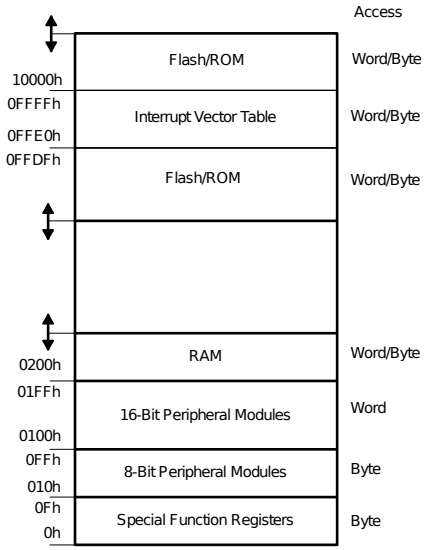
Посебно намењена уграђена емулациона логика налази се на самом уређају и приступа јој се путем JTAG без потребе коришћења додатних ресурса.

– Бенефиције уграђеног емулатора

- Пристојан развој, отклањање грешака са извршавањем у пуној брзини, тачке прекида, корак по корак
- Развој програма је унутар-системски и има исте карактериситике као финални програм
- Могућност да се очува интегритет више-сигналних апликација тако да не зависе од интерфејса каблова

2.3 Адресни простор

MSP430 је базиран на фон Најмановој архитектури тако да постоји један заједнички адресни простор који деле: регистри са специјалним функцијама (SFRs), периферије, RAM, као и FLASH/ROM меморија. Кодом је могуће приступити само парним адресама. Подацима се може приступити као бајтовима или речима. величина адресне меморије је 128 KB са планом да се њен капацитет у будућности прошити



2.4 FLASH/ROM

Стартна адреса FLASH/ROM меморије зависи од количине присутне FLASH/ROM меморије на уређају. Последња адреса за уређаје са мање од 60kB FLASH/ROM је 0FFFFh; у супротном зависи од уређаја. FLASH се може користити како за код тако и за податке. Речи или бајтови могу се сторнирати и користити директно из FLASH/ROM без претходног копирања у RAM. Вектор прекида заузима последњих 16 речи FLASH/ROM адресног простора, где се вектор највеће тежине налази на највишем нивоу, на адреси 0FFFFh.

2.5 RAM

RAM меморија почиње на 0200h. Крајња адреса RAM-а зависи од количине RAM меморије присутне на уређају. У меморији се могу чувати подаци и програми.

2.6 Периферијски модули

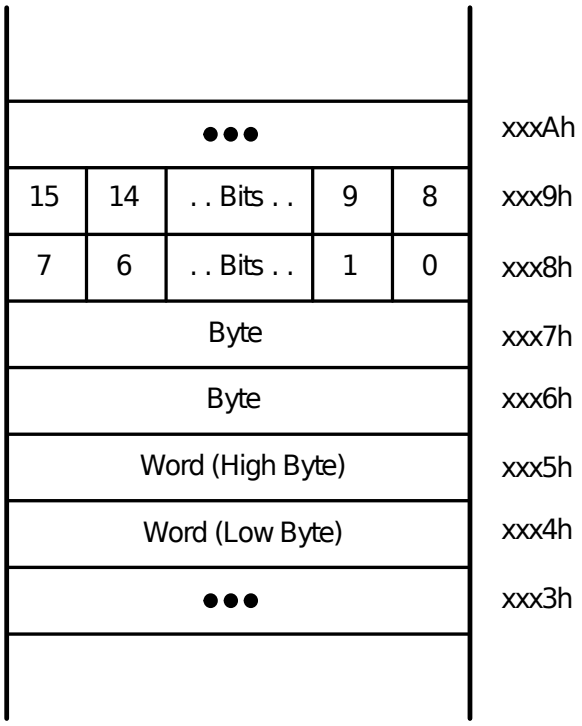
Периферни 16-bit модули у адресном простору налазе се од 0100h до 01FFh. Овим модулима се искључиво приступа са инструкцијама дужине речи. Ако се користе инструкције дужине бајта, могуће је приступити само парним адресама, а виши бит је увек "0. Адресни простор од 010h до 0FFh намењен је 8-bit периферним модулима. Овим модулима приступа се инструкцијама дужине бајта. Ако се приступа користећи инструкције дужине речи, долази до грешака у битовима највеће тежине. Ако се инструкција дужине речи уписује у бајт модул, тада се само уписују доњи битови, без битова највеће тежине.

2.7 Регистар са специјалним функцијама SFRs

Неке периферне функције можемо пронаћи у SFRs регистрима. Ови регистри се налазе у доњих 16 битова адресног простора и оргранизовани су у бајтове. Приступамо им инструкцијама дужине бајта.

2.8 Организација меморије

Бајтови се налазе на парним или на непарним адресним локацијама. Инструкције дужине речи налазе се само на парним локацијама. Када се користимо инструкцијама дужине речи можемо користити само парне адресне локације. Доњи бајт речи, увек се налази на парној локацији. Виши бајт се налази на следећој непарној адреси. На пример ако се неки податак дужине речи налази на локацији XXX4h, онда се виши бајт налази на локацији XXX5h.



2.9 Дигитални улазно/излазни портови

MSP430FG439 има 10 улазно/излазних портова, од P1 до P10. Сваки од портова има осам пинова. Сваки пин, могуће конфигурисати као улаз или као излаз, или се са сваке улазно/излазне линије могу читати подаци. Портови P1 и P2 имају могућност реализације прекида. Прекиди се могу програмирати да се активирају на падајућу или пењајућу ивицу улазног сигнала, и тако за оба P1,P2 порта појединачно. Сви пинови P1 и P2 улазно/излазних портова могу да подрже један вектор прекида.

– Карактеристике дигитално улазно/излазних портова су :
  - Независно програмабилни портови
  - Неограничене комбиновање излаза или улаза
  - Независно програмибилни прекиди на P1 и P2 портовима
  - Неависни регистри за улазе и излазе

2.10 Начин функционисања дигиталних улазно/излазних портова

Конфигурација дигиталних улазно/излазних портова врши се софтверски. Регистар сваког порта има дужину 8-bit и можемо им приступити инструкцијама дужине бајта.

2.10.1 Улазни регистар PXIN

Сваки бит у сваком регистру PXIN представља вредност улазног сигнала на кореснодентном порту, када је тај пин конфигурисан да врши улазно/излазну функцију.

– Бит=0; Улаз је на нули

– Бит=1; Улаз је на јединици

2.10.2 Излазни регистри PXOUT

Сваки бит у сваком појединачном регистру PXOUT-а представља вредност која ће бити излаз на одговарајућем улазно/излазном порту, када је тај пин конфигурисан да врши улазно/излазну функцију.

– Бит=0; Улаз је на нули

– Бит=1; Улаз је на јединици

2.10.3 Регистри правца PXDIR

Сваки бит у сваком појединачном PXDIR селектује правац одговарајућег улазно/излазног порта, не узимајући у обзир функцију која је додељена том пину.

– Бит=0; Улаз је на нули

– Бит=1; Улаз је на јединици

2.10.4 Регистар за одабир функције PXSEL

Пинови портова често су мултиплексовани са осталим периферијским модуларним функцијама. Сваки PXSEL користимо да бисмо поставили функцију пина, улазно/излазног порта или модула периферије.

– Бит=0; Улазно/излазна функција је додељена пину

– Бит=1; Постављена је функција за модул периферије

Постављање PXSELx на "1" не поставља аутоматски и правац пина. Зато је у неким случајевима потребно подесити и PXDIRx бит, тако да одговара функцији коју тај модул периферије врши.

2.10.5 Табела са улазно/излазним регистарским командама

У табели се могу видети карактеристике сваког појединачног порта. Тип регистра,адреса и иницијално стање.

Port	Register	Short Form	Address	Register Type	Initial State
P1	Input	P1IN	020h	Read only	–
	Output	P1OUT	021h	Read/write	Unchanged
	Direction	P1DIR	022h	Read/write	Reset with PUC
	Interrupt Flag	P1IFG	023h	Read/write	Reset with PUC
	Interrupt Edge Select	P1IES	024h	Read/write	Unchanged
	Interrupt Enable	P1IE	025h	Read/write	Reset with PUC
	Port Select	P1SEL	026h	Read/write	Reset with PUC
	Resistor Enable	P1REN	027h	Read/write	Reset with PUC
P2	Input	P2IN	028h	Read only	–
	Output	P2OUT	029h	Read/write	Unchanged
	Direction	P2DIR	02Ah	Read/write	Reset with PUC
	Interrupt Flag	P2IFG	02Bh	Read/write	Reset with PUC
	Interrupt Edge Select	P2IES	02Ch	Read/write	Unchanged
	Interrupt Enable	P2IE	02Dh	Read/write	Reset with PUC
	Port Select	P2SEL	02Eh	Read/write	0C0h with PUC
	Resistor Enable	P2REN	02Fh	Read/write	Reset with PUC
P3	Input	P3IN	018h	Read only	–
	Output	P3OUT	019h	Read/write	Unchanged
	Direction	P3DIR	01Ah	Read/write	Reset with PUC
	Port Select	P3SEL	01Bh	Read/write	Reset with PUC
	Resistor Enable	P3REN	010h	Read/write	Reset with PUC
P4	Input	P4IN	01Ch	Read only	–
	Output	P4OUT	01Dh	Read/write	Unchanged
	Direction	P4DIR	01Eh	Read/write	Reset with PUC
	Port Select	P4SEL	01Fh	Read/write	Reset with PUC
	Resistor Enable	P4REN	011h	Read/write	Reset with PUC
P5	Input	P5IN	030h	Read only	–
	Output	P5OUT	031h	Read/write	Unchanged
	Direction	P5DIR	032h	Read/write	Reset with PUC
	Port Select	P5SEL	033h	Read/write	Reset with PUC
	Resistor Enable	P5REN	012h	Read/write	Reset with PUC
P6	Input	P6IN	034h	Read only	–
	Output	P6OUT	035h	Read/write	Unchanged
	Direction	P6DIR	036h	Read/write	Reset with PUC
	Port Select	P6SEL	037h	Read/write	Reset with PUC
	Resistor Enable	P6REN	013h	Read/write	Reset with PUC
P7 PA	Input	P7IN	038h	Read only	–
	Output	P7OUT	03Ah	Read/write	Unchanged
	Direction	P7DIR	03Ch	Read/write	Reset with PUC
	Port Select	P7SEL	03Eh	Read/write	Reset with PUC
	Resistor Enable	P7REN	014h	Read/write	Reset with PUC
P8	Input	P8IN	039h	Read only	–
	Output	P8OUT	03Bh	Read/write	Unchanged
	Direction	P8DIR	03Dh	Read/write	Reset with PUC
	Port Select	P8SEL	03Fh	Read/write	Reset with PUC
	Resistor Enable	P8REN	015h	Read/write	Reset with PUC
P9 PB	Input	P9IN	008h	Read only	–
	Output	P9OUT	00Ah	Read/write	Unchanged
	Direction	P9DIR	00Ch	Read/write	Reset with PUC
	Port Select	P9SEL	00Eh	Read/write	Reset with PUC
	Resistor Enable	P9REN	016h	Read/write	Reset with PUC
P10	Input	P10IN	009h	Read only	–
	Output	P10OUT	00Bh	Read/write	Unchanged
	Direction	P10DIR	00Dh	Read/write	Reset with PUC
	Port Select	P10SEL	00Fh	Read/write	Reset with PUC
	Resistor Enable	P10REN	017h	Read/write	Reset with PUC

3 Покретање FET debugger-a

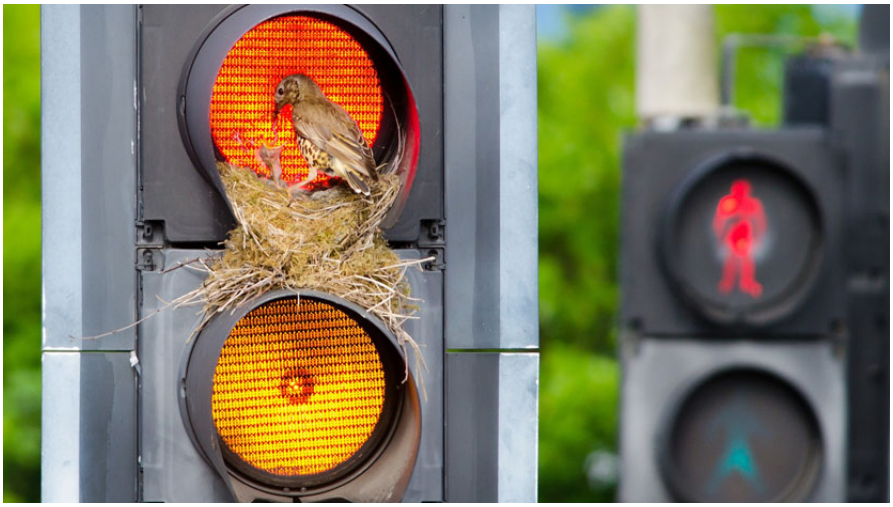
У оквиру IAR Embedded Workbench-a, програмер може да бира да ли ће симулирати свој програм или ће га послати директно на уређај где ће се његове инструкције извршавати. Ово убацивање кода на MSP430 микроконтролере реализује се помоћу MSP FET-Flash Emulation Tool.

MSP FET-Flash Emulation Tool се преко 12-pin конектора повезује са MSP430 микроконтролерима, а са рачунаром се повезује путем Micro-USB кабла. Пре повезивања Micro-USB кабла са рачунаром, потребно је инсталирати адекватне драјвере, који ће омогућути нормално функционисање уређаја. Приликом инсталације IAR Embedded Workbench-a (ово важи само за најновију верзију софтвера, тренутно IAR Embedded Workbench 6.5, НЕЋЕ РАДИТИ НА ЗАСТАРЕЛОМ СОФТВЕРУ) биће нам понуђено да инсталирамо додатне драјвере што треба да прихватимо.

Након инсталације софтвера, креирамо пројекат, испишемо код па кликнемо на таб у горњој траци под називом Project. Излази нам падајући мени. Сада је потребно ући у Options. У оквиру менија Options, потребно је да кликнемо на ставку Debugger, а затим уместо Simulator у одељку Driver, треба поставити FET Debugger и на крају кликнемо OK. Сада можемо да извршимо debbug нашег кода кликом на зелену стрелицу. Када кликнемо на стрелицу FET Debugger прво брише заостали код у меморији микроконтролера а затим почиње са пребацивањем новог кода. Debugg је могуће вршити у реалном времену, где ће debugger притиском тастера F11, на тастатури ићи кроз код линију по линију, тако да можемо да пратимо како се наш код извршава у реалном времену, што нам омогућава да имамо потпуни преглед и контролу над током програма.

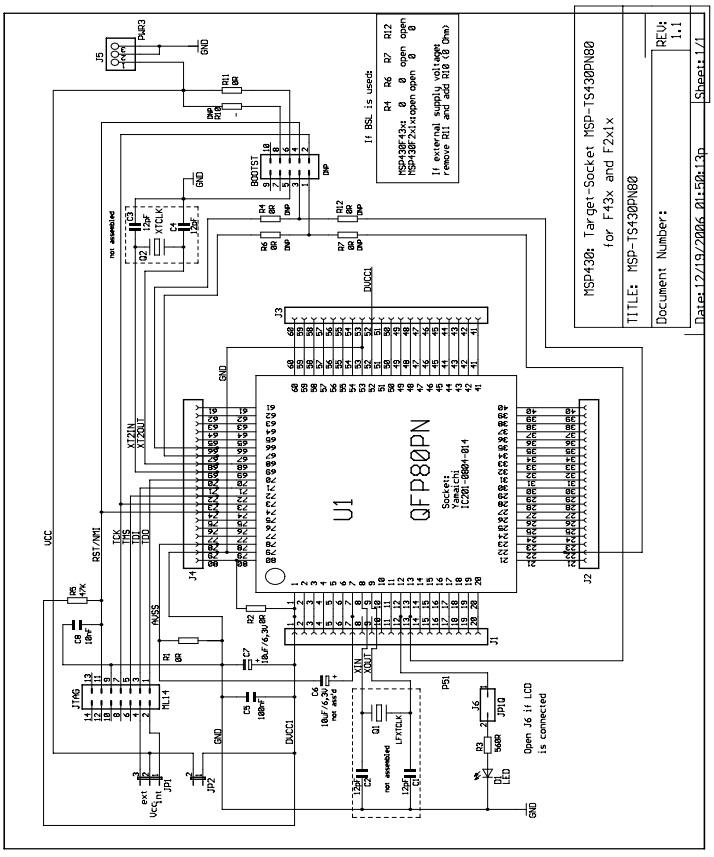
4 Пројектни задатак

Циљ овог пројекта је да направимо уређај који ће симулирати рад семафора у системима саобраћаја. Уређај поседује три диоде црвену, жуту и зелену које имитирају рад светла на семафору, која се наизменично пале. Прво се пали црвена диода, а после одређеног времена се пали и жута диода. Затим се ове две диоде гасе и пали се зелена диода која такође светли одређено време а потом се гаси и опет се пали црвена диода и тако у круг.



5 Реализација пројектног задатка

За реализацију овог пројекта користили смо MSP430FG439, диоде, IAR Embedded Workbench и MSP FET-Flash Emulation Tool.На пинове микроконтролера (P1.0,P2.0 ,P2.1) смо залемили три диоде: црвену,жуту и зелену. Што се тиче софтвера користили смо IAR Embedded Workbench, и у њему смо написали код који пали ове три диоде наизменично, прво црвену па жуту и на крају зелену. Затим је програматор (MSP FET-Flash Emulation Tool) испрограмирао микроконтролер и тако смо добили уређај који симулира рад семафора на некој ракрсници. Тако испрограмирани микроконтролер сада може да се смести у одговарајуће кућиште где би се само уместо диода на исти начин залемила светла семафора. За напајање уређаја можемо користити CR2032 батерију која даје 3V што је довољно за несметан рад уређаја, или да микроконтролер прикачимо на екстерни извор напајања



6 Закључак

Циљ овог пројекта био је да направимо уређај који ће симулирати рад семафора у саобраћају. Дакле овакав уређај је веома користан у системима саобраћаја,а његова имплементација и није толико сложена. Потребан је један микроконтролер рецимо, као у нашем случају, MSP430FG439, софтвер написан у програмском језику C и програматор који ће тај код после сместити на уређај.

7 Прилог код

```
 #include <msp430xG43x.h>
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;  // Заустављамо Watchdog timer
  P1DIR |= 0x01;  // пин за црвено светло на семафору
  P2DIR |= 0x01;  // пин за жуто светло на семафору
  P2DIR |= 0x02 ; // пин за зелено светло на семафору
  for (;;){
    unsigned int i;
    unsigned int k;
 P1OUT ^= 0x01; // уапли црвено светло
    i = 5000000;   // кашњење
    do (i--);    // декрементирање бројача
   while (i != 0);
    P2OUT ^= 0x01; // упали жуто светло
P1OUT &= 0x00; //  гаси се црвено и жуто светло
    P2OUT &= 0x00;
   P2OUT ^= 0x02; //  пали се зелено светло
    k=1200000;    ;   // кашњење

    do(k--);  // декрементирање бројача
    while(k!=0);
    P2OUT &= 0x00; // гаси се зелено светло
   // програм поново улази у главну бесконачну петљу
```