'

Mikroprocesorové a vestavěné systémy

IMP Semestrální zkouška, ak. rok 2015/2016

Otázka dne: jsou někde zpracovány Fulltext otázky z let 2012/2013 a starší?

Odpověd dne: Nejsou.

Tento dokument vzniká jako skupinová práce studentů předmětu IMP na FIT VUT v Brně, v akademickém roce 2015/2016. Nestyďte se přispět svými vlastními poznatky a dotazy.

Staré otázky + wannabe survival guide (s.r.o.):

- https://fituska.eu/viewtopic.php?f=1205&t=23222

Nepřehledný originál zde prosím:

- https://docs.google.com/document/d/1CvR6zIG5NDlowCyWcLSk4IIf1yhFjqcWJio4TrgTstk/edit#

Teorie
Příklady (doplňte prosím)

Teorie

Harvardská architektura

- paměť pro data a program je oddělená
- ATMEL (AVR), PIC

Von Neumannova architektura

- společná paměť pro data i program
- MC9S08JM60 (HCS08 je Von Neumann)

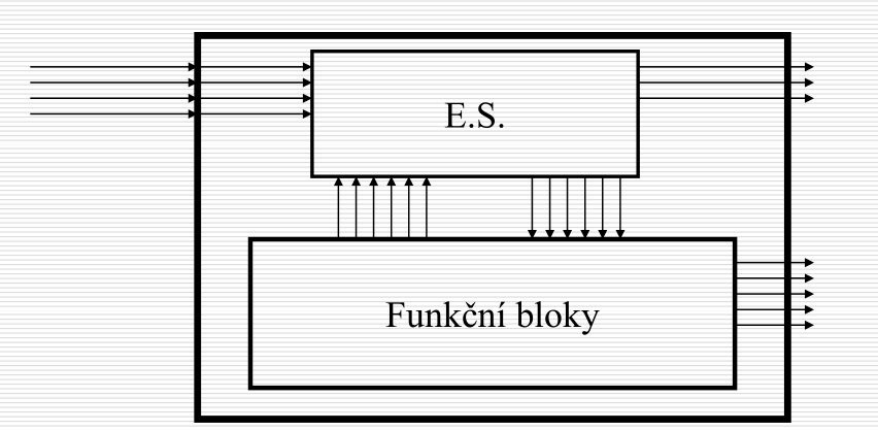
Vestavný systém (VS aka ES)

- kombinace hw a sw jejímž smyslem je řídit externí proces, zařízení či systém
- počítač zabudovaný do systému, které není pro uživatele viditelný
- deska s procesorem a ostatní elektronikou naprogramovanou pro řízení přístroje do něhož je zabudována
- Vlastnosti: Autonomní chování, reaktivnost, odezva na podněty z prostředí, kritičnost = vliv odchylek od normálního chování na bezpečné plnění úlohy

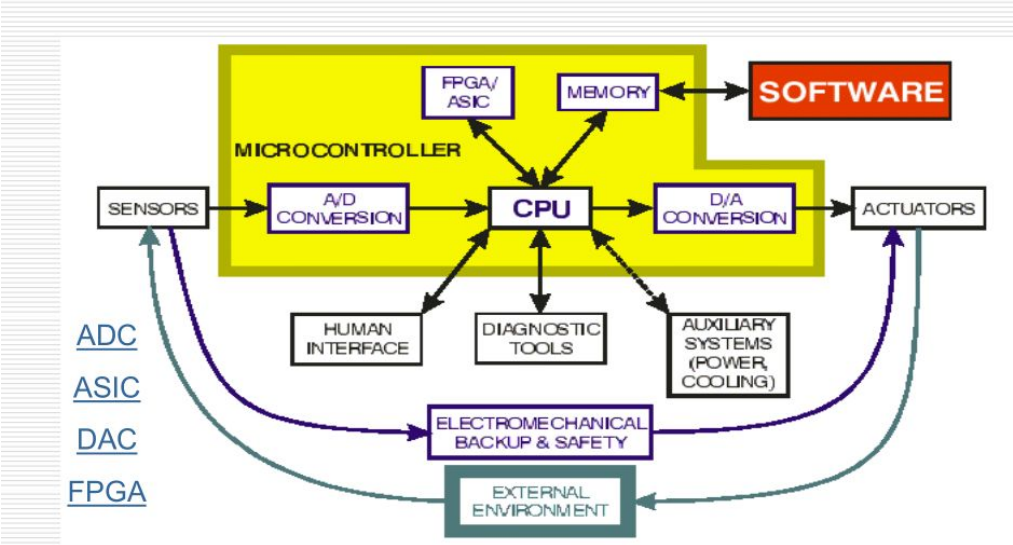
Rozdíly mezi VS a PC

VS	**PC**
jeden program po celý život	uživatel spouští různé programy
uživatel netuší že pracuje s počítačem, hlavní interakce nemusí být s člověkem	periferie hlavně pro komunikaci s uživatelem
startuje sám bez lidského zásahu	nutnost operačního systému
není programován koncovým uživatelem	je programován k.u.
pevná doba výpočtu (není třeba přídavný výkon)	rychlejší = lepší
kritéria: cena, příkon, předvídatelnost...	cena, střední příkon

Struktura VS



Funkční bloky vykonávají požadované fce, třeba měření apod.



X-by-wire

- technologie nahrazující mechanické a hydraulické systémy např. v automobilech pro ovládání řízení, brzd apod., elektronickými systémy
- v případě selhání nasazuje záložní moduly, bezpečný

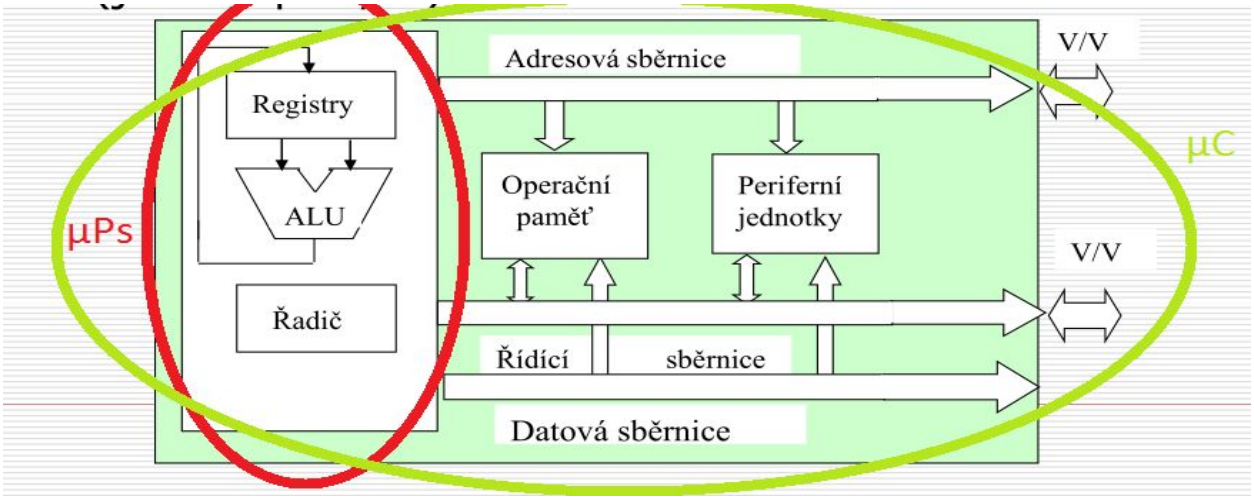
Vlastnosti VS

- potřebné I/O
- průchodnost - doba průchodu hlavní programovou smyčkou
- odezva - doba odezvy na vnější podměty
- cena
- rozměry
- životnost

- úsporný kód
- míra spolehlivosti - pravděpodobnost že systém bude pracovat spolehlivě v daném čase
- udržovatelnost - pst že systém bude pracovat správně po danou dobu po výskytu chyby
- funkčnost - pst že systém bude pracovat v daném čase
- bezpečnost - při poruše nedojde ke škodám
- zabezpečení - důvěryhodná komunikace

Vývoj/platforma VS

- s pevnou logikou - logické obvody poskytují omezený sortiment fcí, pro změnu fce třeba provést nový návrh obvodu
- mikroprocesory (μPs) - základní procesorová jednotka na čipu; vhodné ke zpracování dat v PC systémech; často vyžadují přídavné obvody pro I/O
- mikropočítače (μC) - μP + podpůrné obvody, periferie, paměť; při malém počtu I/O vývodů nazýváme mikrokontrolérem; vhodné k řízení vnějších zařízení v aplikacích požadujících minimum komponent; mají zabudované I/O obvody, podporu pro I/O operace, plánování a přerušení z venku



- instrukční sady:
  - **μP** instrukční sada je „processing intensive“ s výkonnými adresovými módy s instrukcemi pro operace nad velkým objemem dat.
  - **μCs**, na druhou stranu, mají instrukční sadu pro řízení vstupů a výstupů.

CISC

- Complex Instruction Set Computing
- jednoduché programování, efektivní využití paměti
- hodně instrukcí, adesovacích módů, instrukce o proměnné délce, instrukce pro operandy v paměti
- Intel 80x86, Motorola 68K

RISC

- Reduced Instruction Set Computing

- redukovaná sada instrukcí
- menší složitost
- v každém cyklu ukončí provedení instrukce
- 2 inst pro paměť LOAD a STORE
- mnoho symetrických registrů organizovaných do registrového souboru
- schopen pracovat na vyšší frekvenci než CISC
- složité inst trvají déle než na CISC (nutný rozklad na jednoduché inst), jejich počet je ale menší
- ARM

Metriky pro návrh VS

- NRE/JN náklady - jednorázové náklady na vývoj
- jednotkové náklady - náklady na výrobu jednoho kusu (bez NRE)
- velikost
- výkonnost - doba výpočtu nebo propustnost
- příkon (W, mW)
- flexibilita - schopnost měnit fci bez nutnosti nových NRE nákladů
- doba pro výrobu prototypu (time-to-prototype)
- doba do uvedení na trh (time-to-market)
- udržovatelnost - schopnost být modifikován při poruše
- správnost
- bezpečnost
- latence - doba mezi startem úlohy a ukončením
- průchodnost - počet úloh/sec

Klíčové technologie VS

Procesorová:

- procesorová - procesory pro obecné využití
    - nízké jednotkové náklady (NRE pokrývá velký počet jednotek)
    - pečlivý návrh, výkon, velikost, příkon
    - nízké NRE, krátká doba uvedení na trh, vysoká flexibilita
- procesorová - aplikačně-specifické proc….
    - programovatelný proc. optimalizován pro specifické aplikace
    - paměť programu, optimalizované datové cesty, speciální funkční jednotky
    - flexibilita, výkonnost, rozměry a výkon

Integrované čipy:

- IC (integrated chip) technologie - plnězákaznické VLSI
    - všechny vrstvy jsou optimalizovány (umístění tranzistorů, jejich rozměry, propojení)
    - výkonnost, rozměry, malý příkon. vysoké NRE
- IC (integrated chip) technologie - Polozákaznické
    - umístění a propojení některých bloků ponecháno návrhářům
    - výkonnost, rozměry, malý příkon, ale trochu menší NRE, dlouhá doba návrhu
- IC (integrated chip) technologie - PLD (programovatelné logické obvody)

    - všechny vrstvy existují, spoje jsou tvořeny/rušeny pro požadované fce
    - FPGA např.
    - nízké NRE, okamžitá dostupnost IC
    - dražší, větší, vyšší spotřeba, pomalejší

Režimy činnosti MC9S08JM60

- run - plný běh aplikace
- active background - ladění, bootloader
- wait - CPU zastaven, systémové hodiny běží, plná fce napěťového regulátoru
- stop - zastaveno CPU a BUSCLK
    - Stop 3 - omezení spotřeby vnitřních obvodů, možný rychlý přechod do run
        - uchován obsah RAM, USB RAM, registrů, I/O vývodů
        - napěťový regulátor nepřechází do nízkopříkonového režimu
    - Stop 2 - odpojení napájení vnitřních obvodů
        - uchován obsah RAM, USB RAM
        - nastavení I/O uloženo pro pozdější obnovení
        - stejně tak uloženo nastavení periferií
        - při návratu ze stop2 je nutné obnovit tato data, jinak budou v reset stavu

Programovací model HCS08

- CPU
    - Von Neumann (společná paměť pro kód i data)
    - instrukční sada CISC (hodně instrukcí, ...)
    - střadačová architektura
- Paměť
    - Big Endian (více významné B na na nižších adresách) (intel je Little Endian)
- Registry
    - 8bit Accumulator - **střadač**
    - 16bit Index registr H:X
    - 16bit Stack Pointer (SP)
    - 16bit Program Counter (PC) - **adresa aktuální instrukce**
    - 8bit Condition Code Register (CCR) - **příznaky** (CARRY, ZERO, NEGATIVE, INTERRUP MASK, …)

Zapis v programu

konstanta zacina ***#***

#64 - 64 v desítkove soustavě
#$40 - 40 v soustavě se základem 16
#%0100000 - binární soustava
#’@’ - znak

adresa (není vlevo #)
100 zápis v soustavě se základem 10, 8 bit

$F0 zápis v soustavě se základem 16, 8 bit
$B000 zápis v soustavě se základem 16, 16 bit

Adresovací režimy

- Inherent - INH - instrukce nepotřebuje zadat operandy
- Relative - REL - řeší překladač (skokové instrukce)
- Immediate - přímý operand instrukce př. *STA 10*
- Direct - DIR - adresa ulozen v 8bit operandu (*MOV 10, 20* = přesun z adresy 10 na adresu 20 = zleva doprava)
- Extended - EXT - jako Direct ale 16bit (lepsi 8bit pro optimalizaci, adresy 0-255 jsou rychlejsi)
- Indexed - př. *LDA ,X* *STA 40, X* (40 + indeX registr HX)

Adresový prostor

- RAM
- FLASH
- registry (v RAM i FLASH)

- **Registry**
  - direct page: hlavně příznaky
  - high page: obsluha resetu, nastavení konfiguraci (watchdog, ...)
  - non-volatile: nelze měnit za běhu aplikace
  - vektory přerušení:
    - vektor 16bit, uložen na 2 adresach (8+8bit)
    - mapování na libovolné místo v paměti (kódu), kde po přerušení se začně vykonávat program
    - vektor reset - uvedení mikrokontroléru do původniho stavu (0xFFFE:FFFF)

- **RAM**
  - statická paměť
  - po resetu beze zmen, pri power up nedefinovan
  - 0x0000 - 0x00FF
    - rychlá část paměti (1B adresa) => uložení často využívaných proměnných
    - instrukce pro práci s bity
  - 0x0100 a vyse
    - práce s daty a zásobníkem
    - pomalejší
  - USB RAM
    - CLK hodiny 2x rychlejší než u mikrokontroléru
    - primárně určena k alokaci prostoru pro uložení BDT (buffer descriptor table) a endpoint bufferů (buffer u komunikace)
- **Flash**
  - primárně pro uložení programu
  - nejmenší mazatelná jednotka je stránka
  - jako první je nutné nastavit pracovní frekvenci FCLK (v FCDIV) (150kHz - 200kHz)
  - po resetu lze do FCDIV zapsat max. 1
  - automatický přechod do úsporného režimu pri řídkých přístupech
  - lze ochránit blok paměti před čtením / zápisem

Ochrana bloku

- znemožněni vestavěné aplikaci provádět změny (programování, mazání) v chráněných blocích
- př. ochrana úseku s bootloaderem programu

Ochrana dat

- zamezení neautorizovanému přístupu k obsahu paměti RAM a FLASH

Obsluha události/periferií

- **Dotazováni (polling)**
  - cyklické testování příznaku procesorem
  - nevýhoda: procesor vytížen (spotřeba), zpoždění detekce
  - výhoda: jasné řízení programu (sekvence příkazů)
- **Přerušení**
  - činnost procesoru pozastavena v aktuální úloze => zpracování instrukcí od návěští dané ve vektoru přerušení
  - globální vs. lokální maska přerušení => tabulka přerušení
  - asynchronní v řízení programu

    - nevýhoda: frekvence událostí je častá => problém při výpočtech na hlavním vlákně
    - výhoda: u neperiodických událostí
    - přerušení je detekováno v posledním cyklu právě se provádějící instrukce a hned po ní obslouženo (u resetu okamžitě)

Přerušovací podsystém - flow

1. Detekce požadavku o přerušení
2. Kontext CPU (aktuální stav) uložen na zásobník
    a. uloženi registru H na zásobniku (v programu při přerušeni pak program může přepsat registr) - instrukce PSHH, PULH
3. Zakázání přeruseni nastavením I bit v CCR (pouze jedno přerušeni, nelze zanořovat ... explicitně lze zanořování povolit nastavením bitu v CCR - musí být ale omezeno - stackoverflow)
4. Vybere se vektor přerušení (s nejvyšší prioritou) a adresa začátku obsluhy přerušení se vloží do registru Program Counter
5. Po provedení obsluhy přerušení se volá instrukce rti, která zajistí obnovení kontextu procesoru ze zásobníku
6. Procesor pokračuje tam, kde byl přerušen.

Přerušení typy

- nemaskovatelné
    - mohou nastat i při vypnuté globální masky přerušení.
    - SW přerušení - instrukce swi
    - př. RESET
- maskovatelné
    - musí být povoleno lokální AND globální přerušení
    - do tabulky vektoru přerušení se přidá adresa pokračování při přerušení
    - po obsluze přerušení se musí nastavit bit acknowledge

Rozdíl mezi přerušením a podprogramem?

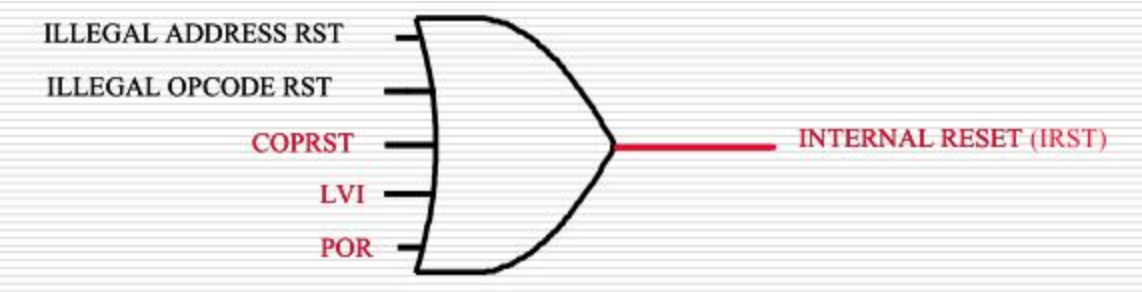
- rozdíl je v toku zpracovávaných dat
- podprogram zpracuje data v předvídatelném okamžiku, jeho volání je pevně dáno v kódu
- přerušení se děje asynchrónne - nezávisle na toku dat programu
- proto u přerušení ukládáme před obsluhou celý programovací model vyjma H registru a u podprogramu pouze návratovou adresu (proto na konci rozdílné návratové instrukce RTI a RTS)

Watchdog = COP (Computer operating properly)

- systém pro eliminaci zacyklení programu a špatné reakce mikrokontroléru
- po navršení čítače watchdog registru se vyvolá automaticky reset mikrokontroléru
- k oddálení resetu se musí čítač smazat
- explicitní vypnoutí watchdog modulu: COPT na 00 nebo vynulovat celý registr COP

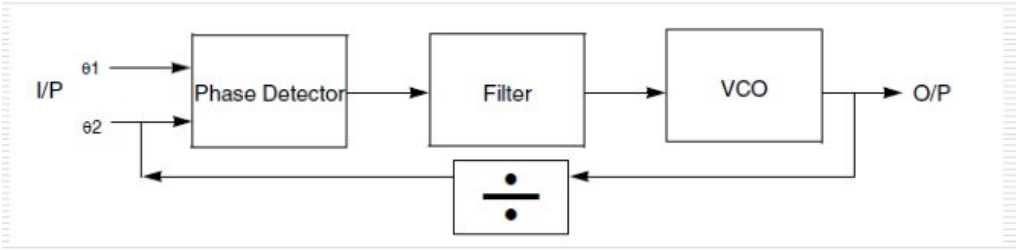
Zdroje resetu

- externí příčinou
  - při zapnutí mikrokontroléru (POR)
  - při přivedení nízké úrovně na vnější vývod RESET (nulovací tlačítko) POR
- interní příčinou
  - narušení správného běhu programu (COP)
  - pokles napájecího napětí (LVI)
  - detekce neplatného operačního znaku a adresy



PLL

- phase-locked loop aka fázový závěs
- jedná se o zapojení, které generuje výstupní signál jehož fáze je odvozena z fáze signálu vstupního (může generovat stejnou frekvenci či její násobek)
- Zpětnovazební regulační smyčka, dělička ve zpětné vazbě určuje poměr mezi vstupní a výstupní frekvencí
- Fázový detektor porovná fáze vstupního a výstupního signálu, rozdíl fází vyjádří napětím (kladným nebo záporným)
- Tímto napětím se dolaďuje frekvence oscilátoru.
- Dolnopropustný filtr stabilizuje chod zpětné vazby (odstraňuje šum a oscilace výstupu detektoru).
- používá se například k frekvenční modulaci



Odezvy u RT systémů

- soft:
    - doporučený čas odezvy, optimální pro dosažení uživatelem očekávané kvality služeb
    - nedodržení mezí odezvy vede k neškodnému snížení kvality systému
- hard:
    - časová odezva u tohoto systému musí být dodržena
    - nedodržení vede k vážným následkům (trvalé a nevratné)
- firm:
    - meze časové odezvy jsou definovány s určitou tolerancí
    - překročení tolerance může mít vážné dopady na okolí systému, je však snaha nastavit toleranci tak, aby k tomu nedošlo
    - typicky jsou následky vratné pokud neselžou záchranné mechanismy

Operační systém

- organizuje práci výpočetního systému
- zajišťuje pomocí služeb OS základní správu HW
- služby OS implementovány v jádru OS vytváří nad HW virtuální počítač
- dle struktury dělíme na:
    - monolitické
    - víceúrovňové
    - klient-server
    - distribuované

uC/OS-II vstup a opuštění kritické sekce

1. **disable/enable INTERRUPTS:**
    a. vstup do kritické sekce => maskování přerušení
    b. opuštění kritické sekce => odmaskování přerušení
    výhoda: jednoduchost
    nevýhoda: přerušení je vždy povoleno odchodem z kritické sekce (i když bylo zakázáno před vstupem do ks!)
2. **push/pop:**
    a. vstup do kritické sekce = uložení globální masky přerušení na zásobník a poté zákaz
    b. opuštění kritické sekce = načtení globální masky ze zásobníku
    výhoda: uchování masky přerušení (lokální zásobník funkce vstupující do ks)
    nevýhoda: větší režie než u 1., problém pokud je makro rozgenerováno před přístupem na lokální zásobník
3. **save/restore:**
    a. vstup do kritické sekce = globální maska přerušení uložena do lokální proměnné (třeba argument) fce vstupující do ks a zákaz
    b. opuštění kritické sekce = obnova masky
    výhoda: odstranění problému u 2.

Emulátor vs. simulátor

Simulátor

• Program, který běží v PC
• v paměti udržuje model µC
• interpretuje program pro µC
• zobrazuje stav µC.

Simulátor – výhody a nevýhody

+ Levné řešení,
+ lze ladit kdekoli,
+ není potřeba mít µC,
+ nehrozí poškození systému v případě fatální chyby,
+ není problém zobrazit veškeré informace o stavu µC.
– neběží zcela v reálném čase,
– problematická simulace periferií, zejména neobvyklých,
– nemusí být zcela věrné.

Obvodový emulátor

• Hardware, které se navenek chová jako µC, ale…
• uvnitř kromě obvodů zajišťujících všechny funkce µC jsou také obvody pro zobrazení stavu µC a jeho ovládání (breakpointy, změny stavu atd.)
• končí hlavicí, která se zasunuje do patice v zařízení místo mikrokontroléru,
• je připojen k PC, kde běží aplikace podobná simulátoru - debugger.

Emulátor – výhody a nevýhody

+ Přesná emulace v reálném čase,
+ možnost komfortního ladění přímo cílové aplikace včetně skutečných nestandardních periferií.
– drahý prostředek, vyplatí se jen při opakovaném komerčním vývoji vestavěných systémů,
– vazba na pracoviště
– obtížněji přenosný,
– zakoupení vede k nutnosti vyvíjet aplikace jen pro daný typ mikrokontroléru.

Zadání 2013/2014

1) Příklad na počítání obsahu registrů časovače v PWM režimu - napsat do kódu obsah TPM1MOD a TPM1C1V, zadáno BUSCLK, hodnota předděličky, F(PWM), střída (%). Takže klasické počítání podle vzorečků.

2) Doplnit kód pro záchyt hrany, resp. pro změření délky jednoho impulsu. Je to obslužná rutina přerušení. Prostě šlo o to ošéfovat, aby se při prvním přerušení uložil stav registru, ve kterém je "čas" hrany (to je ta nástupná), a při druhým volání/přerušení od nové hodnoty registru odečíst ten uloženej stav z minula a to je výsledek.

3) ADC převodník, nakreslené schéma připojení - MCU a k němu 3V přes potenciometr (tj. vstup převodníku 0 - 3V). Zadáno napájecí napětí MCU (a současně referenční - V(REFH)): 5V, 10-bit mód převodníku a číslicová hodnota, která je právě snímána (tuším 430). Spočítat v %, "kde je potenciometr (3V - 100%, 0V - 0%). Podle vzorečku se spočítá maximální číslicová hodnota: 3V/5V * (2^10 - 1), vyjde něco přes 600, to je 100%, pak už jen trojčlenka a vyjde, že těch 430 je asi 70%.

4) SPI: obrázek - schéma MCU a 2 PZ, dokreslit a popsat signály (dráty), tj. něco jako MISO, MOSI, SS (to by asi mělo být pro každé PZ zvlášť). Navíc ještě napsat, jaká stěžejní číslicová komponenta musí být v každém zařízení (snad to měl být posuvný registr).

5) Opravdu lehký příklad (takovej záchranej), podle konečného automatu dopsat mezery v kódu - jména daných stavů a událostí.

Zadání 2014/2015

Příklady si už moc nepamatuju ale myslím že tam bylo:
1) Něco s frenkvencí a periodou a zakreslit signál, vše se mělo vyčíst z kodu myslím. - podobné z minulých let
2) Doplnit do kody jak změříme délku pulzu - taky podobné jak minulé roky
3) ČA převodník ale měli jsme schéma a pro daná čísla v hexa tvaru zjistit co se sepne
4) Nepamatuju
5) Tabulka a doplnit křížky co je nezbytně nutné pro něco, to už si nepamatuju

z