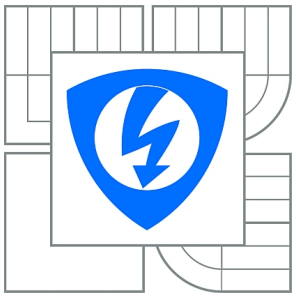
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

KNIHOVNY PRO MSP430G2 MIKROKONTROLERY

LIBRARIES FOR MSP430G2 MICROCONTROLLERS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

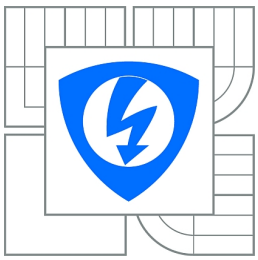
AUTOR PRÁCE — ADAM ŠTĚPÁNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. MILAN ŠTOHANZL

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student:	Adam Štěpánek	***ID:***	134418
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Knihovny pro MSP430G2 mikrokontrolery

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s mikrokontrolery řady MSP430G2XXX a aplikacemi, pro které je tato řada určena, seznamte se s pomocnými obvody pro mikrokontrolery. Po domluvě s vedoucím práce vyberte několik aplikací a vytvořte návrh zapojení se zvoleným mikrokontrolerem.

Realizujte zvolená zapojení a vytvořte programy pro mikrokontroler tak, aby sloužili jako přehledné demo-aplikace zobrazující zacházení s periferiemi.

DOPORUČENÁ LITERATURA:

[1] LUECKE, J. Analog and Digital Circuits for Electronic Control System Applications: Using the TI MSP 430 Microcontroller. Oxford: Elsevier Inc., 2005.

[2] MSP430 LaunchPad [online]. Dostupné na www: ti.com/launchpad.

Termín zadání:	10.2.2014	***Termín odevzdání:***	30.5.2014

Vedoucí práce: Ing. Milan Štohanzl
Konzultanti bakalářské práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se zabývá návrhem knihoven pro mikrokontrolery MSP430G2 ovládajících interní periferie mikrokontroleru a externí součástky. Mezi interní periferie se řadí sběrnice pro sériovou komunikaci a A/D převodník, pomocí nichž jsou ovládány externí součástky s podporou těchto komunikačních protokolů jako například EEPROM paměti, nebo expander. Dále obvod reálného času a 8x8 LED maticový displej.

KLÍČOVÁ SLOVA

Mikrokontroler, MSP430, RTC, 8x8 LED displej, SPI, UART, I^2C, EEPROM

ABSTRACT

This thesis solves a issue with libraries for microcontrollers MSP430 using them for interfacing internal peripheries such as serial buses or ADC convertor. And also external devices with support of this communication protocols, for example EEPROM memories and expander, or with other interfaces including real time circuit and 8x8 LED matrix display.

KEYWORDS

Microcontroller, MSP430, real-time-clock, 8x8 LED matrix display, serial peripherial interface, UART, I^2C, EEPROM

ŠTĚPÁNEK, A. *Knihovny pro MSP430G2 mikrokontrolery*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2014. 50 s., 6 s. příloh. Bakalářská práce. Vedoucí práce: ing. Milan Štohanzl.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Knihovny pro MSP430G2 mikrokontrolery jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a~jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 30. května 2014 .....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce ing. Milanu Štohanzlovi. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 31. května 2014 .....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem této práce je vytvořit knihovny pro ovládání interních periferií mikrořadiče MSP430 od firmy Texas Instruments řady G2xxx, a externích komponent běžně řízených jejich prostřednictvím. Knihovny budou ovládat zejména sériovou komunikaci rozhraní SPI, I^2C a UART, dále pak 10-bitový A/D převodník. Knihovny pro externí komponenty zajišťují řízení EEPROM a FLASH paměti, 16-bitového expandéru, 8x8 LED maticového displeje a obvodu reálného času RTC.

Dalším úkolem je takto vytvořené knihovny využít v demoaplikacích, které zprostředkovávají názorný příklad jejich použití v praxi. K vývoji samotných knihoven je určen nízkonákladový vývojový kit TI LaunchPad, vhodný k programování čipu především prostřednictví programu Code Composer Studio.

Fyzické zapojení řízených součástek je realizováno na DPS.

1 HARDWARE

V následující kapitole je představena základní charakteristika mikrořadiče, jeho struktura a vlastnosti. Dále je vysvětlena funkce a účelu užitých komponent, práce s nimi a popis komunikačních protokolů pro jejich ovládání.

1.1 Mikrokontrolery řady MSP430

Tato řada představuje MCU s velmi nízkou spotřebou. Zkratka Mixed Signal Processor odkazuje na praktické využití pro potřebu různých analogových vstupů. Jádro tvoří von Neumannova architektura s 16 bitovovou instrukční sadou RISC. Pro svoji nízkou spotřebu a jednoduchost ovládání úsporných režimů jsou vhodné pro aplikace napájené baterií, vyžadující dlouhou výdrž. Pro tyto aplikace je možné připojit externí krystal, který slouží jako oscilátor pro generování nižších frekvencí. Další využití nachází v automobilovém průmyslu, v ovládání prostřednictvím USB, snímání senzorů, bezpečnosti a kapacitního snímání dotyku.

1.2 MSP430G2xxx

V základní verzi představuje 14 – 20 vývodů, z toho 8 – 14 vstupně/výstupních pinů zahrnutých v 1 – 2 portech. Maximální frekvence 16 MHz je dostupná při napájení 3,3 V, přičemž vlastní napájení je v rozsahu 1,8 – 3,6 V. V tabulce 1.1 můžeme vidět rozdílné vlastnosti u několika vybraných mikrokontrolerů z této řady, zvláště velikost paměti a použitý A/D převodník.

Tab. 1.1 Vlastnosti různých G2xxx (převzato z [1])

MSP430	Flash (kB)	SRAM (kB)	Počet časovačů	A/D převodník
G2221	2	0,125	1	-
G2313	4	0,25	2	Slope
G2452	8	0,25	1	10-bit SAR
G2553	16	0,5	2	10-bit SAR
G2755	32	4	3	10-bit SAR

Možný omezující faktor pro použití náročnějších knihoven, které mohou zabírat více paměti, spočívá v omezeném prostoru paměti FLASH, kam se ukládá strojový kód programu i data. Může se stát, že všechny potřebné knihovny pro danou aplikaci se do paměti nevejdou. Je pro ně tedy nutné použít vhodný typ MCU a při programové části dbát zřetel na efektivitu a velikost kódu. SRAM je non-volatilní paměť, ztrácející všechna data po odpojení napájení, určená k uchovávání proměnných.

MCU je navržen pro práci v pěti možných režimech se sníženou s potřebou

prodlužujících životnost baterie, která slouží v mnohých aplikacích jako napájecí zdroj. Navíc vnitřní periferie se automaticky vypínají jakmile přestanou být využívány.

Komunikační rozhraní MSP430 se dělí na 2 základní moduly. Starší typ USI (Universal Serial Interface) poskytuje sériovou synchronní komunikaci s jedním hardwarovým modulem protokoly I^2C a SPI. USCI modul (Universal Serial Communication Interface), nabízí kromě komunikaci až dvou hardwarových modulů také snadnější ovládání než USI, větší možnosti nastavení a navíc UART rozhraní.

Vnitřní systém hodinových signálů umožňuje taktovat frekvenci pro CPU od 1 MHz do 16 MHz. Některé hodnoty jsou kalibrovány. Zdroj hlavního hodinového signálu je digitálně ovládaný oscilátor DCO. Z toho můžeme odebírat frekvenci i pro vysokorychlostní interní periferie, a jim sloužícím hodinový signál SMCLK. Vstupní piny MCU podporují připojení externího krystalu 32 kHz, nebo až 16 MHz. Běžný hodinkový krystal produkující signál pro pomocný ACLK může sloužit ke generování přerušení obsluhující udržování reálného času, nebo jako zdroj hodinového signálu pro periferie pracující během režimu nízké spotřeby s vypnutým hlavním oscilátorem.

Vstupně/výstupní piny mají obvykle vzhledem k vnitřnímu zapojení předem definovaný účel a je třeba při návrhu a zapojení brát na toto ohled, jak ukazuje následující tabulka.

Tab. 2: Vstupně/výstupní piny MSP430G2553 a jejich funkce (převzato z [1])

Označení pinu	Účel
P1.0	TA0CLK, ACLK
P1.1	UCA0RXD, UCA0SOMI, A1
P1.2	UCA0TXD, UCA0SIMO, A2
P1.3	A3
P1.4	USB0STE, UCA0CLK, A4, SMCLK
P1.5	UCB0CLK, UCA0STE, A5
P1.6	UCB0SOMI, UCB0SCL, A6
P1.7	USB0SIMO, UCB0SDA, A7
P2.0 - P2.5	Vstupy/výstupy pro komparační/zachytávací jednotky časovačů

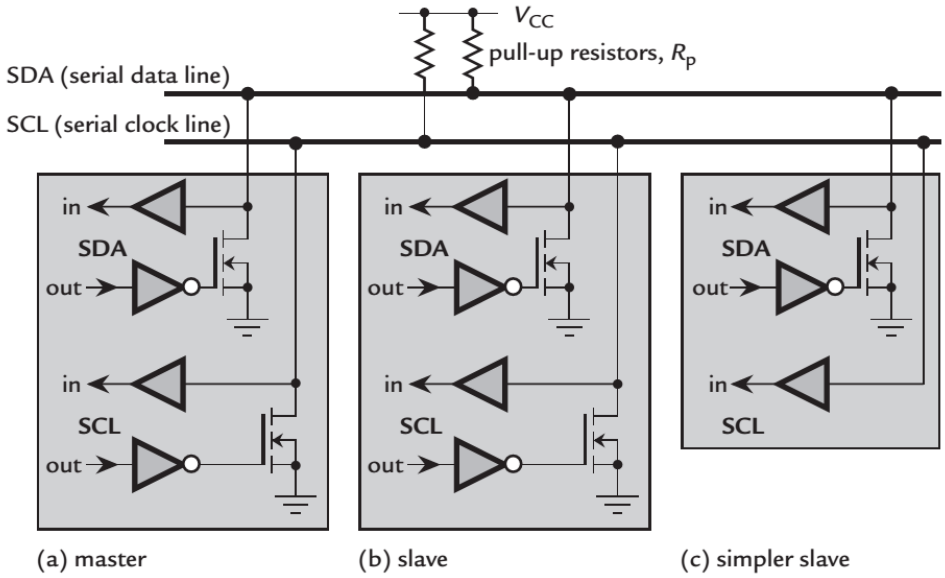
1.3 Napájení

Logické signály mají charakteristické hodnoty logických úrovní a v případě napájení s velkým rušením ze sítě by mohlo docházet k chybám. Filtrování nežádoucích rušení by vyžadovalo odrušovací filtr. Jednoduchým řešením je napájení z baterie. Protože potřebujeme napájení nejméně 3,3 V, použijeme stabilizátor napětí LM317. Vstupní napětí bude dodávat 9 V baterie. Druhou možností bude napájet DPS přímo z vývojového kitu LaunchPad, který má výstupní svorky pro napětí.

1.4 Protokol I^2C

I^2C je zkratka pro komunikační protokol, který byl poprvé představen firmou Philips a široce přijat po vypršení patentových smluv v roce 2006. Základ tvoří dvě linky, SDA - Serial Data a SCL - Serial Clock. (Společné uzemnění, ač je nezbytné, se nezahrnuje do počtu linek).

Výhodou je již zmíněný malý počet vývodů, protože nepotřebuje zvláštní linku pro výběr zařízení, jako je tomu u SPI. Místo toho výběr podřízeného zařízení Slave probíhá prostřednictvím unikátní adresy. Navíc SDA je vždy obousměrná.



Obr. 1.1: Schéma sběrnice a vnitřního zapojení komunikačních zařízení I^2C. (Převzato z [5])

Nevýhoda spočívá v omezené rychlosti - základní verze nabízí frekvenci do 100 kHz, přičemž většina novějších zařízení zvládne přenos do 400 kHz. Maximální teoretickou rychlost 3,4 MHz lze dosáhnout jen se značnou úpravou hardwaru. Toto omezení je způsobeno vnitřním zapojením součástek pro I^2C sběrnici. Linky SDA a SCL se v klidovém stavu nacházejí ve vysoké logické úrovni, ve které je drží pull-up rezistory připojené na napájení. Ke stažení linky na nízkou logická úroveň dochází prostřednictvím tranzistorů s otevřeným kolektorem, pro bipolární, a s otevřeným drainem v případě MOSFET, která je i na Obr. 1.1. Přes pull-up rezistor a přes tranzistor tedy teče proud k zemi a logická úroveň linky je 0. Z toho vyplývá další nevýhoda a tou je spotřeba proudu, který takto vznikne. Kvůli tomuto by měla být hodnota odporu rezistorů co největší, aby byl spotřebovaný proud co nejmenší. Jeho velikost je však ovlivněna ještě jiným faktorem. Mezi linkami sběrnice a stejně tak proti zemi vzniká parazitní vazební kapacita C_v. Když potom jedna z linek přejde z logické úrovně 1 do logické úrovně 0, vazební kapacita C_v se začne nabíjet kolektorovým proudem přes pull-up rezistor a následná sestupná hrana má menší strmost. Stejně tak nástupná. Specifikace I^2C sběrnice vyžaduje délku náběžné hrany pro 100kHz menší než 1 μs, což představuje limit pro RC konstantu vzniklé kapacitní vazby pro kapacitu C_v a odpor R_p. Předpokládejme vazební kapacitu C_v = 100 pF. Následujícím vztahem vypočítáme maximální hodnotu pull-up rezistoru:

$$R_p \leq \frac{\tau}{C_v} = \frac{1\,\mu s}{100\,pF} = \underline{\underline{10\,k\Omega}}, \tag{1.1}$$

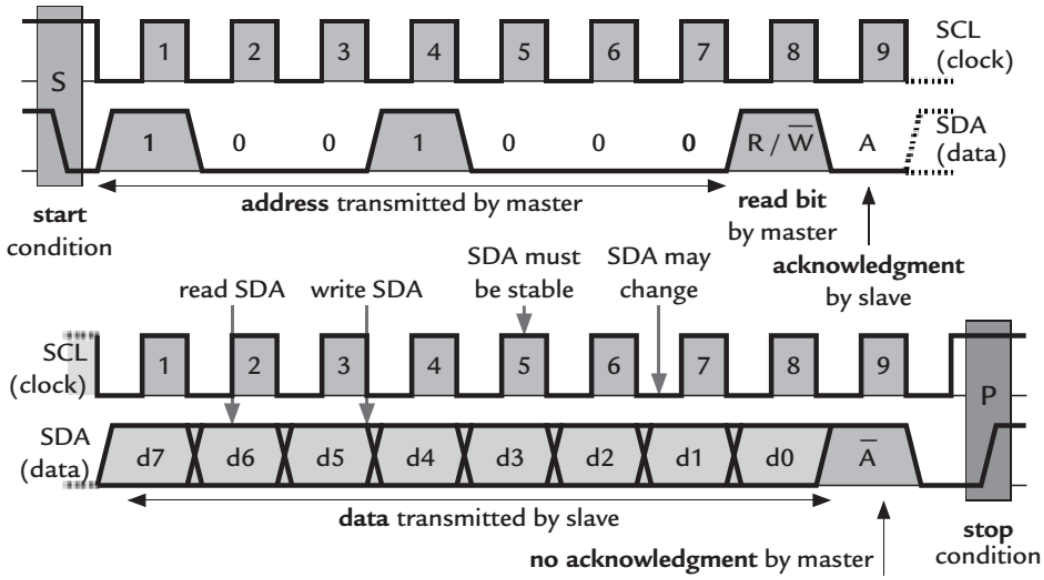
kde R_p je pull-up rezistor na lince, τ je časová konstanta RC článku pro 100 kHz a C_v vazební kapacita sběrnice. (Převzato z [5])

Pro naši aplikaci bude dostačovat rezistor 4,7kΩ, který zajistí bezproblémový chod sběrnice pro frekvence přesahující 200 kHz, v případě vhodným uspořádáním na DPS, kdy vazební kapacita nebude dosahovat takovéto hodnoty, a zároveň poskytne přijatelný ztrátový proud:

$$R_p \leq \frac{\tau}{C_v} = \frac{0{,}5\,\mu s}{100\,pF} = \underline{\underline{5\,k\Omega}} \tag{1.2}$$

kde R_p je pull-up rezistor na lince, τ je časová konstanta RC článku pro 200 kHz a C_v vazební kapacita sběrnice. (Převzato z [5])

Komunikace probíhá mezi zařízením Master. které je v uzlu obvykle jen jedno a pořízeným zařízením Slave. Master generuje hodinový signál nezbytný pro přenos, zahajuje a ukončuje komunikaci, a zprostředkovává adresaci.



Obr. 1.2: Příklad jednoduchého přenosu přes sběrnici I^2C. (Převzato z [5])

Při zahájení přenosu Master generuje tzv. počáteční podmínku, která spočívá ve změně stavu SDA z log. „1" do log. „0", zatímco SCL zůstává ve vysoké logické úrovni. Poté pošle adresu, která je v případě shody podřízeným zařízením potvrzena v devátém hodinovém cyklu (Obr. 1.2). Čtení následujících dat na SDA, která přichází v pořadí od nejvýznačnějšího bitu (MSB), probíhá při vysoké logické úrovni SCL a ke změně dochází při její nízké úrovni SCL. 7. bit adresy tvoří R/W bit určující směr přenosu následujících bytů. Každý přijatý byte potvrdí příjemce potvrzovacím ACK

bitem. Pokud se tak nestane, znamená to, že další data se již nemají odesílat. Následuje tzv. ukončovací podmínka, nebo opakovaná počáteční podmínka s jiným příznakem pro čtení, nebo zápis.

1.4.1 Expander MCP 23016

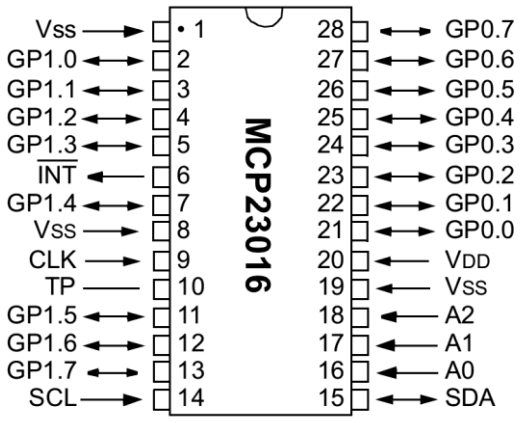
Značně omezený počet vstupně/výstupních portů MCU je znatelný při použití komponent, které mají velkou spotřebu těchto vývodů. Například maticová klávesnice nebo pole diod. Při řešení otázky obsazení portů může nastat situace, kdy nebude toto pole prvků možné připojit celé, nebo na ostatní komponenty již nezbyde volné místo. Potřebné rozšíření kapacity portů lze provést posuvným registrem, dalším mikrokontrolerem, který bude zajišťovat ovládání samostatně, nebo expandérem.

Expander MCP 23016 od firmy Microchip umožňuje zvýšit kapacitu ovládaných portů o 16 vstupně/výstupních pinů. 3 adresními bity A0 - A2 (Tab. 1.1), volené hardwarově pomocí napětí, lze celkový počet adresovaných zařízení navýšit až na 2^3 = 8. Což by teoreticky znamenalo 128 vývodů. Je řízen sběrnicí I^2C s frekvenčním rozsahem 0 – 400 kHz. K připojení na tuto sběrnici jsou nutné zvyšovací pull-up rezistory, které udrží při neaktivním stavu vysokou logickou úroveň a vysokou impedanci.

Tab. 3: Adresní byte MCP23016 (Převzato z [8])

0	1	0	0	A2	A1	A0	R/W

Samotný integrovaný obvod se vyznačuje relativně vysokou proudovou zatížitelností ± 25 mA na jeden výstup a pro potřeby této práce vyhovujícím rozsahem pracovního napětí 2 – 5,5 V. Jako zdroj vnitřní frekvence pro ovládací logiku expanderu slouží externí oscilátor tvořený RC členem s odporem 3,9 kΩ a kapacitou 33 pF, dodávající kmitočet 1 MHz, zapojený na vývodu CLK (Obr. 1.3). V novějších verzích je tento oscilátor zabudován přímo uvnitř IO.



Obr. 1.3: Pouzdro DIP expandéru MCP23016 (Převzato z [8])

Samotná manipulace se vstupně/výstupními porty probíhá skrze zápis do konkrétních registrů charakterizovanými svoji adresou a 8 bitovou délkou. Hodnoty zapsané v dvojici registrů GP0 a GP1 pak přímo reprezentuje aktuální stav portů, které lze pomocí párových registrů IODIR0 a IODIR1 jednotlivě nastavit jako vstupní, nebo výstupní. Piny definované jako vstupní lze číst v inverzním módu, nastaveným skrze registry IPOL1 a IPOL1.

Pokud jeho prostřednictvím ovládáme například klávesnici, není třeba neustále kontrolovat na všech portech sériovou komunikací, zda byla klávesa stisknuta. Slouží k tomu hardwarový příznak přerušení INT realizovaný přechodem z vysoké logické úrovně do nízké na samostatném pinu. Pin se nastaví do této úrovně, jakmile dojde ke změně stavu na portech (například stisknutou klávesou) a vnitřní logika přitom tuto skutečnost stihla zaznamenat.

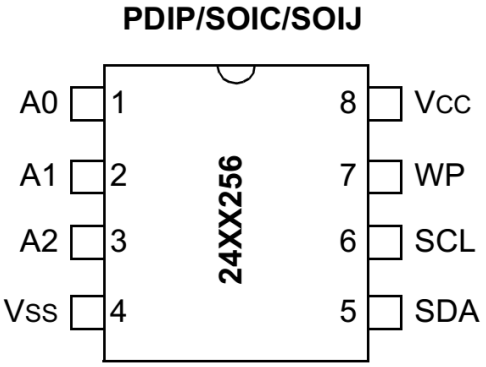
1.4.2 Paměť EEPROM

Význam zkratky Electrically Erasable Programmable Read-Only Memory, pochází z vývojově starších pamětí ROM, které sloužily k uchovávání neměnných dat a k jejich programování byly třeba speciální vypalovací programátory. Novější verze EPROM již umožňovala změnu dat, přičemž procedura vyžadovala smazání všech dat v paměti pomocí ultrafialového záření a teprve poté bylo možné zapisovat. Odtud význam Read-Only, opětovný přepis dat byl buď nemožný, nebo složitý. Novější verze EEPROM řeší mazání dat elektrickým polem a je tak mnohem přístupnější jako paměťové médium pro mikrokontrolery. Význam opětovného zapisování ale zůstal podobný. EEPROM má totiž omezený počet zapisovacích cyklů - asi 1 milion. Ačkoli je toto číslo relativně velké, liší se tímto od flash pamětí. Data se uchovávají jako náboj v NMOS tranzistorech, které využívají princip tunelování ke vložení elektrického náboje mezi vrstvy substrátu tranzistoru.

1.4.3 EEPROM 24LC256

Sériové EEPROM paměti se liší od paralelních menší kapacitou paměti, způsobem připojení k MCU skrze menší počet sériových portů, což umožňuje uložení v 8 vývodových pouzdrech zabírajících méně místa na DPS a jsou vhodné především k uchovávání parametrů a pro použití s mikrokontrolery s malým počtem vývodů.

24LC256 v praxi najde uplatnění především pro nízkopříkonové aplikace v osobní komunikaci a získávání dat. Velikost konkrétního typu je 32 kB a je ovládán pomocí sběrnice I^2C do frekvence 400 kHz. Adresa samotného zařízení obsahuje vrchní kontrolní bity "1010", následované 3 hardwarově volitelnými adresními bity A2 – A0, se kterými lze adresovat až 8 pamětí tohoto typu. Výsledná kapacita paměti by pak dosahovala velikosti 256 kB.



Obr. 1.3: 8 vývodové pouzdro 24LC256 (Převzato z [6])

Ovládání je z části popsáno v kapitole o I^2C protokolu. Při zápisu bytu do paměti se standardně odešle adresa zařízení slave, s přičteným Write bitem, a po jejím potvrzení přichází na řadu 2 bajtová adresa paměťové buňky, která je v rozsahu od 0000 – 3FFFh hexadecimálně. V jedné sekvenci je možné zapsat až 64 bytů dat, které se postupně přijímají, dokud není generována ukončovací podmínka. Pokud by se zapsalo více bytů, adresní ukazatel by se překlopil na první adresu stránky a předešlá zapsaná data by byla přepsána. Nelze však zapisovat celou stránku od libovolné adresy. Ohraničené adresní prostory o velikosti jedné stránky leží mezi celými násobky její velikosti. Například 0000 – 003Fh a 003F – 007Fh. Přes tyto fyzické adresy nelze zapisovat v jedné sekvenci a je nutné operaci rozdělit. Po odeslání dat a ukončovací podmínky potřebuje vnitřní ovládací logika 24LC256 nějaký čas na jejich trvalé zapsání. Po tuto dobu je zaneprázdněna. Tento stav můžeme snadno zjistit odesláním adresy zařízení, kterou ovšem nepotvrdí. Tento stav lze testovat v programové smyčce, dokud zařízení není připraveno k dalším operacím. Jakmile adresu potvrdí, můžeme zapisovat další data.

Operace čtení může probíhat i bez odeslání adresy paměťové buňky. Takto přijatá data pochází z prostoru, kam naposled ukazoval interní adresní čítač, navíc zvýšený o jedničku. Master musí vždy potvrdit přijaté data. Pokud by tak neučinil, znamená to, že nevyžaduje další data a sekvence se ukončí stop podmínkou. Na rozdíl od zápisu lze v jedné sekvenci přečíst celou paměť. Interní adresní čítač se vždy navýší o jedničku po každém čtení.

24LC256 nemá příkaz na smazání celé paměti. V případě potřeby proto musíme smazat data ručně, přepsáním nulovými hodnotami. Zápis do celé paměti lze chránit pomocí hardwarového pinu WP (Write Protection). My ale tuto funkci používat nebudeme a je tedy nutné ho uzemnit, nebo vůbec nezapojit.

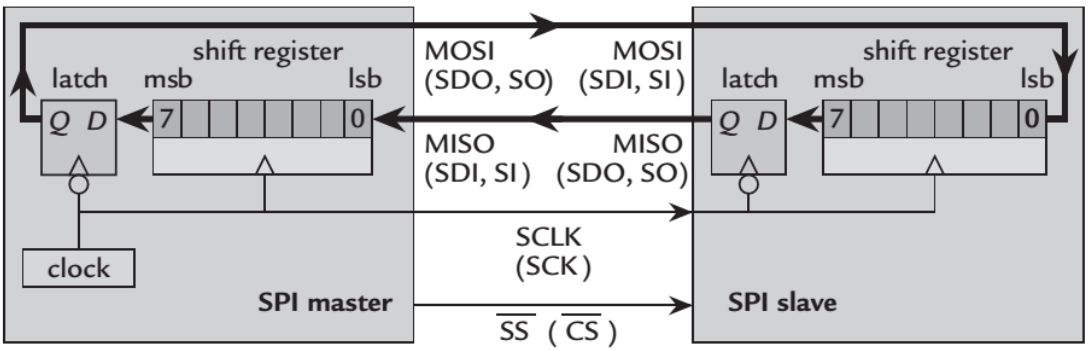
1.5 Protokol SPI

Koncept sériového protokolu Serial Peripherial Interface je založen na dvou posuvných registrech, každý na jedno zařízení, spojených ve smyčce, kdy jedno zařízení označené jako master zajišťuje hodinový signál a podnět k zahájení přenosu.

2 datové linky, z nichž jedna se nazývá MISO (Master In, Slave Out), jejímž

prostřednictvím posílá podřízené zařízení data. Druhá linka MOSI (Slave In, Master Out) slouží k opačnému směru komunikace, jak naznačují šipky na Obr. 1.4.

Nevýhoda SPI spočívá v nejednotné formě, která postrádá určitou normu jako u I^2C. V plném 4-žilovém módu máme 2 datové linky a jeden vodič slouží k výběru konkrétního zařízení Slave a nese označení CS, SS, CE a u MSP430 STE (Serial Transmit Enable). Používá se však také nekompletní rozhraní 3 vodičů, kdy data putují pouze po jednom z nich. Některé komponenty vůbec nemusí odesílat data k MCU, takovým příkladem je D/A převodník s jednosměrnou komunikací. Obvykle data opouští posuvný registr s prvním nejvíce významným bitem, u 3 vodičového rozhraní je tomu často naopak. Také názvy linek mohou být nadepsány rozdílně, jako je tomu u MSP430 - SIMO a MOSI. Dále u tohoto MCU nabývá nekompletní rozhraní 3 vodičů jiný význam a to sice úplnou absenci signálu STE, který však lze ovládat softwarově. Ten v některých případech skutečně není potřebný v případě pouze jednoho připojeného zařízení. Některé součástky ovšem vyžadují tento signál nejen k ohraničení komunikačního rámce, ale také k interním pokynům, zahájení nové konverze u A/D převodníku nebo začátek ukládání dat do paměti EEPROM.

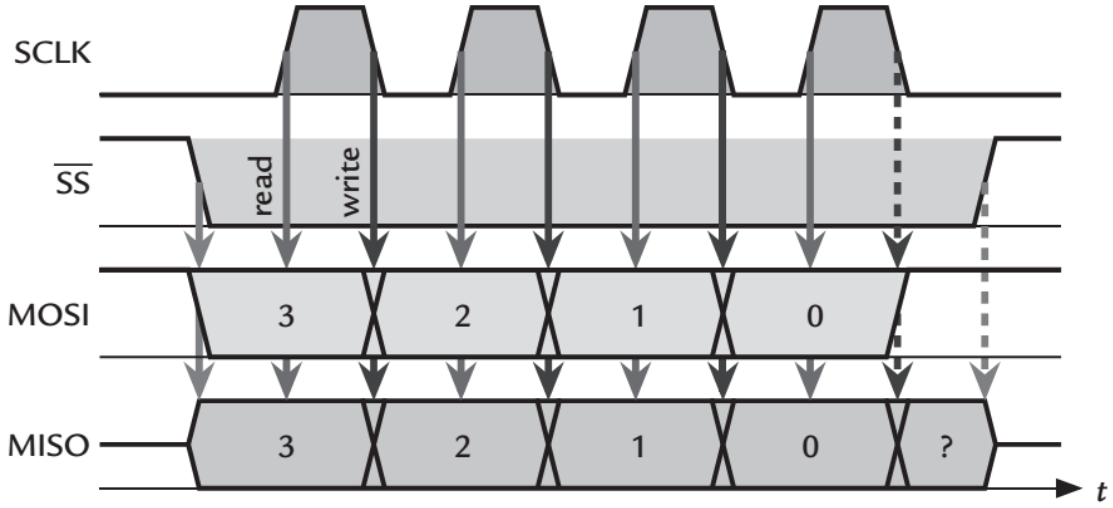


Obr. 1.4: Základní vnitřní struktura zařízení pro komunikaci SPI (Převzato z [5])

Samotný přenos začíná přechodem STE z vysoké logické úrovně do nízké. Na výstup datové linky se zapíše nová hodnota během nástupné/sestupné hrany hodinového signálu. Čtení dat proběhne s hranou opačného smyslu. Po dobu klidového stavu je signál CLK v log. „0" pro základní mód. Hodinový signál se nepotřebuje držet přesné frekvence a je možné, aby se jednotlivé hodinové cykly lišily v délce, pokud dodrží minimální časový interval pro kladnou a zápornou půlperiodu. Logická úroveň hodinového signálu, při které dochází ke čtení, se může lišit, což specifikuje bit CPOL, jehož prostřednictvím nastavíme požadovanou polaritu. Podobně pomocí bitu CPHA určujeme, která hrana bude sloužit pro příjem a která pro zápis. U MSP430 existuje bit CPHA, který má opačný smysl. Kombinací těchto nastavení dostáváme celkem 4 módy pro SPI.

Zapojení více zařízení Slave se může provést buď spojením všech datových linek stejného smyslu, kdy jednotlivé zařízení vybíráme pomocí dalších signálů CS0, CS1 atd., nebo spojením do řetězce, kdy posuvné registry tvoří uzavřený okruh, tzv. Daisy chain.

Výhodou SPI je rychlost až několik desítek MHz. MSP430 ale s ohledem na napájecí napětí zvládne frekvenci v řádu několika MHz. Použití SPI se preferuje zejména pro přenos velkých objemů dat pro velké pamětí, jako odnímatelné Flash a Secure Digital, a u grafických displejů. Další výhodou je jednoduchost samotného protokolu, umožňující ovládat periferie pomocí tzv. bit bangingu, kdy logické úrovně vytváříme na vstupně/výstupních pinech MCU přímo.



Obr. 1.5: Průběh odesílání 4 bitů v SPI módu 0 (převzato z [5])

1.5.1 EEPROM AT25256

Sériová paměť EEPROM od firmy Atmel nabízí kapacitu 64 kB. Je kompatibilní s SPI módy 0 a 3. 16 bitová adresuje prostor v rozmezí 0000 - 1FFF hexadecimálně. Ovládá se pomocí 8 bitového instrukčního setu 6 operačních kódů pro nastavení povolení/zákazu zápisu, přístupu do stavového registru a pro čtení a zápis. Má několik ochran před nechtěným zápisem. Za prvé hardwarovou ochranu pinem WP, na kterém nízká logická úroveň zamezí zápisu do celé paměti, a který nelze v případě, že ho nebudeme používat, nechat nezapojený. V tom našem na něj přivedeme napájecí napětí k zrušení této funkce. Dvojice bitů slouží k ochraně konkrétního adresního prostoru od jeho horní poloviny, spodní čtvrtiny, případně celé paměti. Dále je nutné před každým zápisem povolit zápis operačním kódem Write Enable Latch.

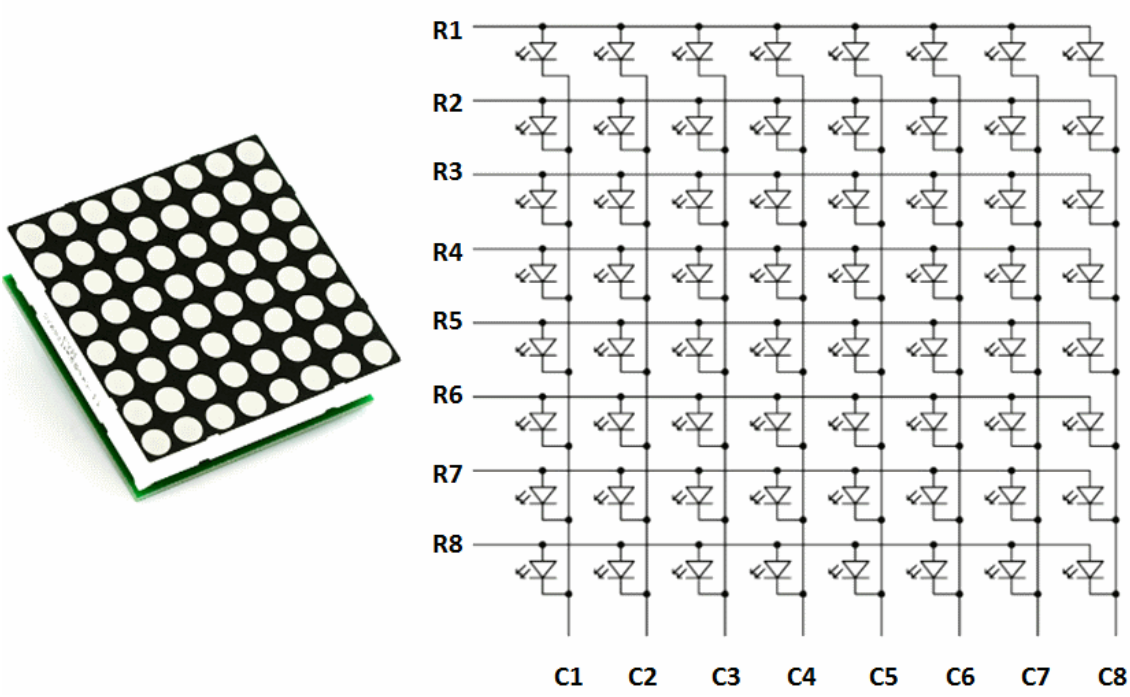
Operace pro čtení nebo zápis probíhá vysláním operačního kódu, následovaným adresními bajty. AT25256 umožňuje zápis až 32 bajtů v rámci stránkovacího módu. Po zapsání přejde vnitřní logika do zaneprázdněného stavu, aby zapsala data trvale. Čtením stavového registru, což je také jediná dovolená akce, můžeme tento stav indikovat a vytvořit programovou smyčku čekající na připravenost zařízení.

Instruction Name	Instruction Format	Operation
WREN	0000 X110	Set Write Enable Latch
WRDI	0000 X100	Reset Write Enable Latch
RDSR	0000 X101	Read Status Register
WRSR	0000 X001	Write Status Register
READ	0000 X011	Read Data from Memory Array
WRITE	0000 X010	Write Data to Memory Array

Obr. 1.6: Instrukční sada AT25256 (převzato z [9])

1.6 8x8 LED Maticový displej

Maticové led displeje se využívají jako zobrazovací jednotky textu, např. v MHD. Výhodou je jejich téměř neomezená možnost skládání vedle sebe, čímž vytvoří velkou zobrazovací plochu. Nevýhodou je pořizovací cena.



Obr. 1.7: Vnitřní zapojení LED diod u HD-M10EG88MD (převzato z [10])

Jednotka, se kterou budeme pracovat, HD-M10EG88MD, má 64 dvojbarevných led diod v matici. Protože bychom obecně potřebovali velký počet vývodů k rozsvěcování jednotlivých diod, jsou spojeny dohromady v řádcích a sloupcích. K zobrazování jedné barvy na displeji stačí 16 vývodů. Pokud chceme rozsvítit konkrétní diodu, přivedeme napětí na daný řádek a uzemníme sloupec, ve kterém se dioda nachází. Předpokládejme, že chceme rozsvítit diodu v 2. sloupci a 3. řádku na Obr. 1.6. Potom uzemníme vývod C2 a napětí přivedeme na vývod R3.

Abychom zobrazili celý znak, použijeme metodu multiplexování. Na řádky, které se mají rozsvítit v daném sloupci přivedeme napětí (logická úroveň „1") a katodu sloupce uzemníme (logická úroveň „0"). Takto postupně zobrazíme všechny sloupce. Tento způsob neumožňuje rozsvítit celý displej naráz, protože bychom nedokázali rozlišovat data pro jednotlivé sloupce. Navíc by to znamenalo velký odběr proudu. Pokud budeme celý displej obnovovat minimálně 25x za sekundu (25 Hz), lidské oko přepínání sloupců nepostřehne a bude vnímat celý vyobrazený znak.

MSP430 nemá kapacitu pro tolik vývodů, nezvládl by tudíž ovládat celý displej. V praxi se obvykle používá posuvný registr, který postupně "sepne" všechny sloupce a 8 řádků ovládá přímo MCU. Tyto posuvné registry ovšem většinou bývají navrženy pro +5 V TTL logiku. Kdežto MSP430 pracuje maximálně s 3,6 V. Práce s takovým posuvným registrem by vyžadovalo další napájecí napětí a úpravu vstupů na spínání tranzistory. Jako vhodná alternativa se nabízí 16 bitový expandér MCP23016, který již zvládne pracovat s napěťovými úrovněmi MCU.

Dalším problémem je odběr proudu samotnými LED diodami.Výrobce uvádí max. hodnotu proudu jednou diodou až 30 mA, kdežto maximální hodnota proudu jedním pinem expandéru je ± 25 mA. Je tedy zřejmé, že expandér by nezvládl dodávat dostatečný proud a displej by svítil jen malou intenzitou, nebo vůbec. Nejjednodušší řešení by byla metoda rozsvěcování každé diody zvlášť. Při obnovovací frekvenci 25 Hz celého displeje by potřebná frekvence zobrazování jednotlivých bodů dosáhla 25x64 = 1,6 kHz. Samotná I^2C sběrnice přitom zvládne komunikaci s rychlostí jen do 400 kHz.

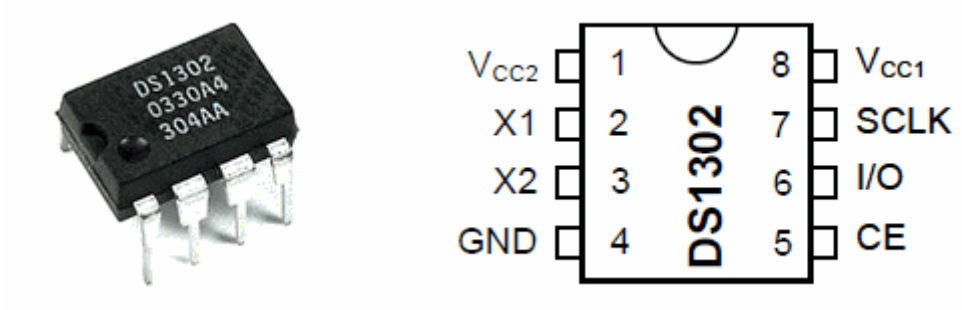
Musíme tedy vyřešit spínání diod pomocí tranzistorů. Anody diod v řádcích budeme spínat PNP tranzistory a katody sloupců tranzistory NPN. Sníží se tím potřebný proud expandéru ke spínání displeje. Navíc úbytek napětí na tranzistorech sníží napájecí napětí pro LED displej na vhodnou úroveň 2 V.

1.7 Obvod reálného času RTC

Obvod reálného času slouží k udržování aktuálního času. Uvolňuje tak výpočetní kapacitu MCU, který by jinak musel hodiny udržovat softwarově, navíc s menší přesností. Používá se ve velkém množství aplikací, kde je nutné po dlouhou dobu měřit čas a generovat datum. Například GPS navigace, které potřebují časový údaj k výpočtu polohy, servery, elektroměry a k časovému zaznamenávání veličin.

Obvod DS1302 je ze sortimentu firmy Dallas. Napájí se 2 – 5,5 V, přičemž samotná spotřeba je menší, než 1 μW. Mimo to má druhé napájení, které umožňuje připojit programovatelný záložní zdroj na pin Vcc_1. Záložní baterie může být nabíjena prostřednictvím obvodu, který se dá nastavit v příslušném registru. V případě, že hodnota hlavního napájení klesne pod úroveň napětí baterie, je napájen z tohoto zdroje.

V prvních 8 bytech volatilní paměti, která se při absenci napájení smaže, jsou uloženy sekundy, minuty, hodiny, dny v týdnu, rok a nastavení nabíjení. Hodiny je možné ukládat v 12 nebo 24 hodinovém formátu. Obsah těchto registrů je reprezentován BCD kódem. S procesorem komunikuje přes jednoduché třívodičové rozhraní synchronním sériovým přenosem přes linky I/O (pro vstup a výstup dat), CE a SCK.



Obr. 1.8: Integrovaný obvod reálného času DS1302 (převzato z [4])

Pro generování hodinového signálu, který je nezbytný pro udržování času a také pro vnitřní ovládací logiku, slouží externí krystalový oscilátor o frekvenci 32,768 kHz, zapojený na pinech X1 a X2. K připojení nejsou třeba žádné kondenzátory, pokud krystal splňuje potřebné specifikace. Novější obvody RTC mají již zabudovaný oscilátor i snímač teploty, který zprostředkovává korekci časování v závislosti na okolní teplotě.

Samotná komunikace je zahájena zvýšením hodnoty vstupu CE na logickou úroveň „1", které samo o sobě spíná ovládací logiku a povoluje přístup do posuvného registru. Nelze ho tedy zanedbat. Následuje příkazový byte, jehož 6. bit určuje výběr přístupu do paměti hodin nebo RAM a 0. bit pak určuje směr další komunikace. Následující data jsou přenášena od nejméně význačného bitu LSB. Po osmi hodinových cyklech jsou data zaznamenána do vyrovnávací paměti a poté synchronizována (v případě zápisu). 7. bit CH v registru sekund zastavuje čítání času a obvod se přepne do nízkopříkonového režimu. WP bit je třeba vynulovat před každým zahájením zápisu. Za zmínku ještě stojí paměť RAM o velikosti 31 x 8 bajtů, do které lze přistupovat v tzv. burst módu, kdy jsou odeslána nebo přijata data v jedné sekvenci. To platí i pro časové registry.

1.8 ADC10 MSP430

Účelem A/D převodníku obecně je převádět spojitý signál na analogovém vstupu na diskrétní hodnotu s binární reprezentací, se kterou může MCU dále pracovat. Řada MSP430 má buď 10 bitový nebo 12 bitový převodník. My budeme pracovat s ADC10. Výstupem 10 bitového převodníku je $2^{10} = 1024$ různých hodnot, které jsou produktem konverze v závislosti na referenčním napětí. Výsledkem konverze je celé číslo zaokrouhlené k nejbližší hodnotě podle vzorce (1.3) a nabývá hodnoty 0 – 1023.

$$N_{ADC} = 2^N \frac{V_{REF}}{V_{IN}}, \tag{1.3}$$

Kde N_{ADC} [-] je výsledek konverze, N [-] je počet bitů převodníku, V_{REF} [V] je referenční napětí a V_{IN} [V] je napětí analogového vstupu.

Konverze probíhá metodou postupné aproximace. Za referenční napětí lze zvolit hodnotu napájecího napětí, nebo interní referenci 1,5 a 2,5 V. Referenci 2,5 V lze použít jen při hodnotách napájení MCU většího, než 2,8 V. Defaultní volba je napájecího

napětí a interní reference se zapíná bitem REFON, vybírá pomocí bitu REF2_5V a potřebuje nějaký čas, aby se stabilizovalo napětí. Konkrétně 30µs. Někdy použití interní reference vyžaduje připojení externího kondenzátoru na vstup, i když je toto napětí používáno uvnitř IO.

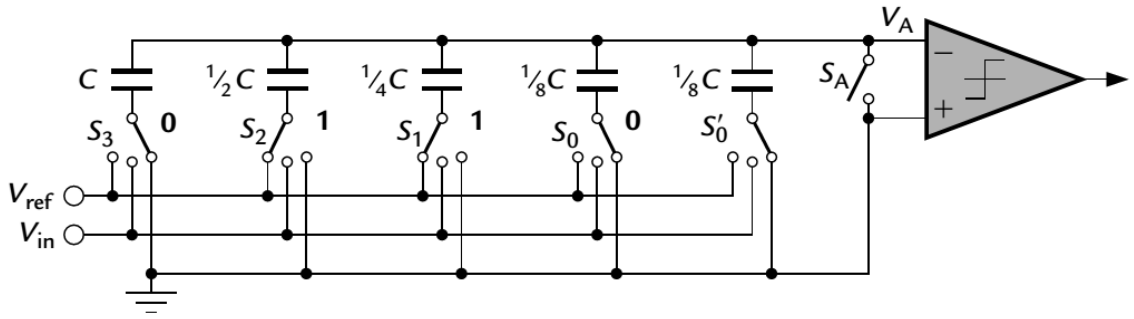
Volit zdroj analogového vstupu lze z 8 externích kanálů a 3 interních, zahrnující i zabudovaný senzor teploty. Všechny vstupy jsou multiplexovány k jednomu převodníku a v případě konverze všech těchto vstupů se přepínají v sekvenci. Výběr zdroje se nastavuje pomocí bitů INCHx.

Datasheet uvádí maximální celkovou chybu převodníku ±5 LSB, typově ±2 LSB. Charakteristika ADC10 je monotónní, bez chybějících kódů, takže výstup může nabývat hexadecimálních hodnot 0000 – 03FFh. Volitelný je i binární formát výsledku konverze. Kromě přímo binárního je možné zvolit dvojkový doplněk. 15. bit takového výsledku reprezentuje znaménko a nulová výsledná hodnota odpovídá polovičnímu napětí mezi referenčním napětím a zemí. V obou případech je výsledek zapsán do registru ADC10MEM.

Převod a konverze ADC10 může být automaticky řízena v několika módech. Buď jako jedno vzorkování a konverze jednoho kanálu (single channel), nebo jako vzorkování a konverze sekvence kanálů (single sequence of channels). Sekvence začíná vždy od analogového vstupu A7. Oba módy mají navíc volbu opakované konverze při nastavení bitu MSC a v určitém intervalu určeného pomocí časovače.

Ke správnému fungování a zvýšení přesnosti A/D převodníku lze přispět oddělením výkonových a signálových uzemění v celém obvodu. Rovněž blokovacími kondenzátory na napájecích vstupech, ty navíc snižují šum.

Jádro ADC10 A/D převodníku tvoří váhová kondenzátorová síť s komparátorem. Princip spočívá v přepínání kondenzátorové sítě k přivedení části vstupního napětí na komparátor a porovnáním zbylého napětí v určitém poměru. Síť je tvořena kondenzátory s číselným součtem kapacit 5C, kde 1. kondenzátor má hodnotu C a další vždy po sobě následující mají hodnotu o polovinu menší, než předchozí. Poslední kondenzátor doplňuje celkový číselný součet na 5C. Porovnávací kondenzátory potřebují nějaký čas na nabití. Při prvním přepnutí porovnáváme polovinu intervalu rozdílu napětí, tedy V_{REF} a 1/2 $V_{REF.}$ Pokud napětí leží v tomto intervalu, do nejvíce význačného bitu celkového výsledku konverze se zapíše log. „1". Takto půlíme interval, dokud nemáme výsledek o délce 10 bitů.



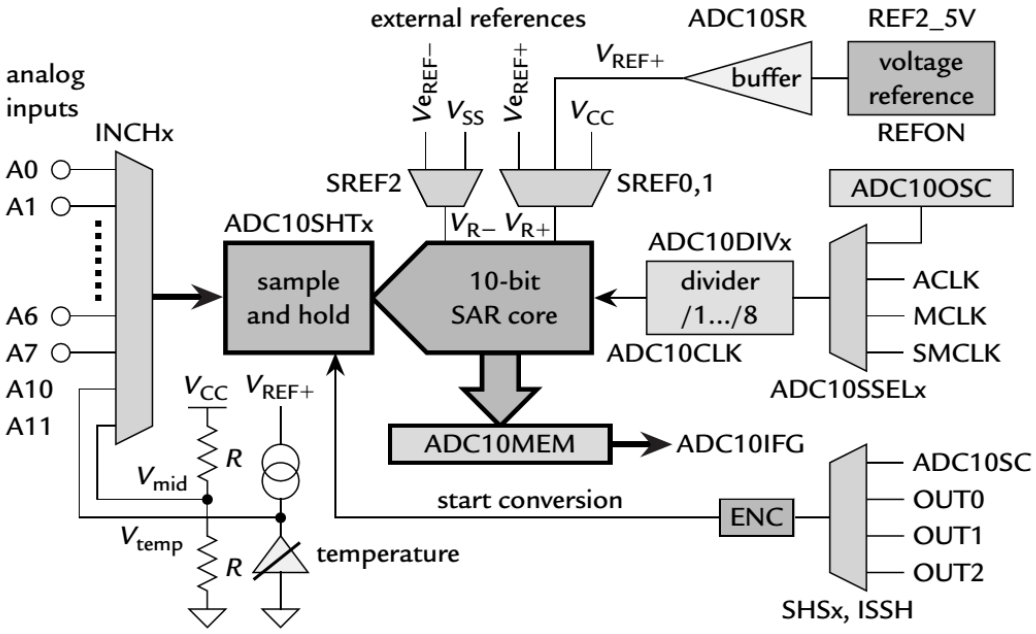
Obr. 1.9: Jádro 4 bitového ADC převodníku (převzato z [5])

Ve skutečnosti je síť kondenzátorů kombinována se sítí rezistorů a společně se podílejí na dělení referenčního napětí a celkové konverzi. Na Obr. 1.7 vidíme příklad kondenzátorové sítě pro 4 bitový převod se vstupním napětím na komparátoru rovným 0,4 krát referenčnímu napětí.

K ovládání funkce převodníku slouží registry ADC10CTL0 a ADC10CTL1. Při jejich nastavování musí být bit pro povolení konverze ENC v nízké logické úrovni. ADC10ON povoluje samotné jádro převodníku. Hodinový signál vybíráme pomocí bitů ADC10SSELx, který slouží jako zdroj pro interní oscilátor převodníku. Zdroj musí být ve frekvenčním rozmezí 0,45 – 6,3 MHz. Frekvence zdroje může být dělena předděličkou 1, 2, 3...8, podle nastavení bitu ADC10DIVx. Interní oscilátor má frekvenci typově 5 MHz, ale specifikace uvádí rozpětí 3,7 – 6,3 MHz.

Důležitý je také počet zvolených cyklů pro vzorkování, volený bity ADC10STx. Výběr je z 4, 8, 16 nebo 24 cyklů interního oscilátoru převodníku. Volí se podle předpokládané frekvence vstupního signálu a rezistivního a kapacitního charakteru analogového vstupu. Nesprávně zvolený počet cyklů může mít za následek, že se vstupní kapacity nestačí nabít na plnou úroveň, nebo aliasing. Blok Sample and Hold na Obr. 1.8 slouží k udržení úrovně měřeného signálu po čas vzorkování.

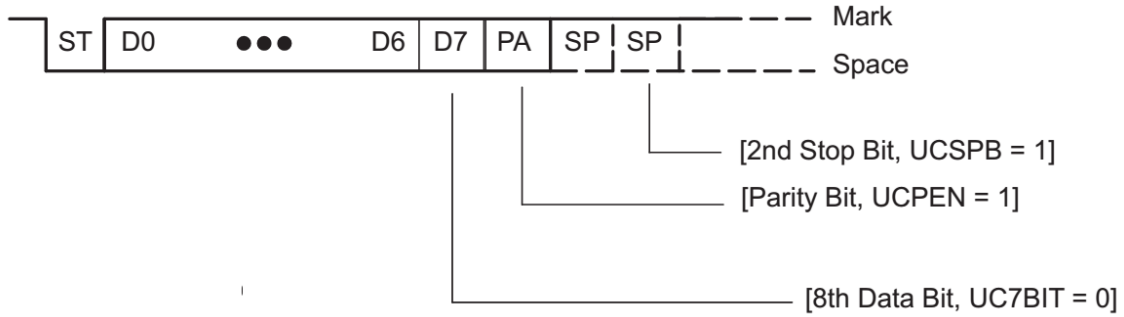
Zahájení konverze se děje nastavením bitu ENC a podnětem spouštěče, jehož zdrojem může být prosté nastavení bitu ADC10SC nebo výstup časovače. Za zmínku ještě stojí přímý datový přenos DTC, který umožňuje výsledky konverze uložit kdekoliv v paměti bez zásahu CPU.



Obr. 1.8: Vnitřní struktura A/D převodníku ADC10 (převzato z [5])

1.9 Protokol UART

Asynchronní přenos se liší od předešlých komunikačních protokolů možností ve stejný čas přijímat, i vysílat data. Je tedy plně duplexní. V základní verzi postrádá také linku pro hodinový signál. Každé zařízení má svůj vlastní generátor hodinového signálu. Kromě toho i samostatné posuvné registry pro příjem a vysílání. Pokud je linka neaktivní, spočívá její hodnota logické úrovně na log. „1". Stejně jako všechny protokoly, i UART potřebuje ohraničit jednotlivé komunikační rámce. Přenos se zahajuje START bitem stahujícím logickou úroveň na „0". Obvykle sedmi, ale i 5 – 9 bitová data se přenáší v pořadí od nejvíce význačného bitu LSB. Volitelně se přidává paritní bit označující sudou, nebo lichou paritu přenášeného znaku. Lichá nebo sudá parita odpovídá sudému, nebo lichému počtu jedniček v přenášeném znaku. Kompletní data následují 1 nebo 2 stop bity (výjimečně 1,5). Stop bity jsou realizované logickou úrovní „1" na lince po dobu trvání počtu stop bitů. Důležitým parametrem je symbolová rychlost SR v jednotkách Bd. Nejčastější používaná hodnota SR je 9600 Bd. Další odlišnost je v rovnocennosti komunikujících zařízení. Zatímco u SPI a I^2C bylo jedno zařízení Mater a další Slave, u UART toto neplatí. 2 zařízení spolu mohou komunikovat bez nutnosti multiprocesorového formátu.



Obr. 1.10: Formát přenášeného znaku (převzato z [2])

Modul USCI u MSP430 podporuje sběrnici UART. 2 piny UCA0RxD a UCA0TxD slouží pro vysílání a příjem na pinech P1.1 a P1.2 (viz. Tab 1.1). Podobu komunikačního rámce určíme nastavením kontrolního registru UCA0CTL1 pod softwarovým resetem modulu. Zvolíme asynchronní přenos směrem od nejvýznačnějšího bitu MSB. Pomocí předděličky sekundárního hodinového signálu zvolíme symbolovou rychlost SR [Bd]. Můžeme k tomu využít vzorce 1.4. Výsledek zaokrouhlený na celé číslo rozdělíme do dvou 8 bitových registrů UCA0BR0 a UCA0BR1.

$$N = \frac{f_{BRCLK}[MHz]}{SR[Bd]} \tag{1.4}$$

Kde f_{BRCLK} je frekvence zdroje hodinového signálu pro UART v MHz a SR symbolová rychlost v Bd.

Vysílání dat je zahájeno automaticky zápisem do vysílacího registru UCA0TXBUF. Jakmile jsou data odeslána do posuvného registru, nastaví se příznak

přerušení a procedura se může opakovat. Přijmutí znaku do UCA0RXBUF vyvolá samostatné přerušení.

Za zmínku stojí možnost automatické detekce symbolové rychlosti.

1.10 Čítač/časovač Timer_A

MCU má jeden nebo dva čítače/časovače v závislosti na typu. Oba obsahují čítací 16 bitový asynchronní registr. Mají 2 – 3 zachytávací/porovnávací registry a rozhraní pro PWM modulaci. Mají 4 módy čítání:

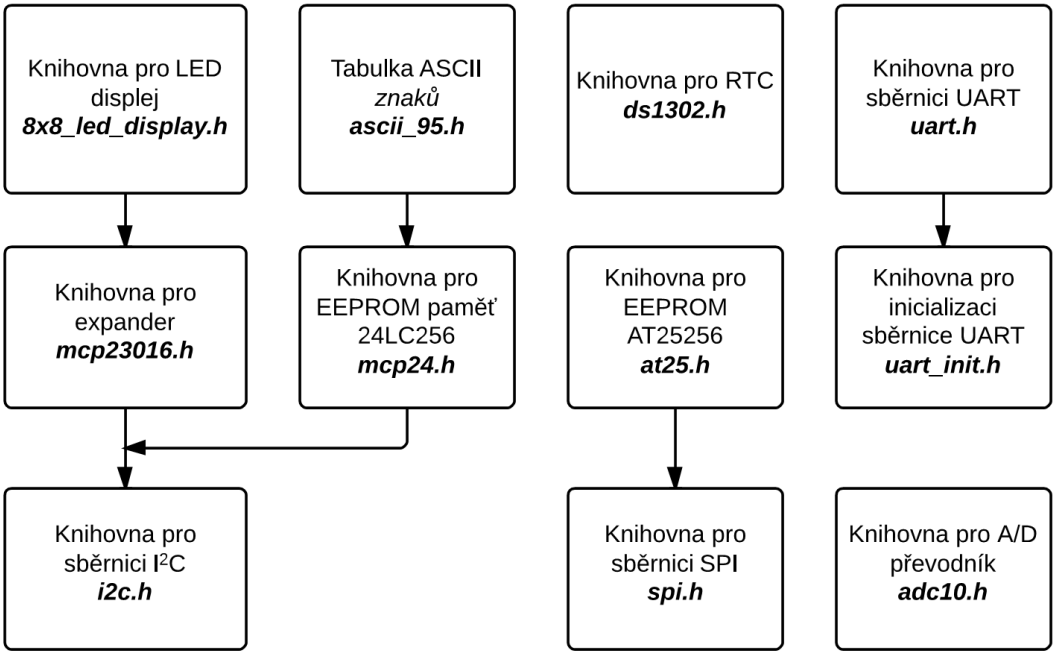
- STOP: čítač je zastaven
- UP: čítač počítá od 0 do hodnoty uložené v porovnávacím registru
- COUNTINUOUS: opakovaně počítá od 0 do FFFFh
- UP/DOWN: tento mód počítá od 0 do hodnoty uložené v porovnávacím registru a poté zase od této hodnoty k 0

Čítací registr má označení TAR. Při přetečení se nastaví příznak přerušení CCIFG. Dále má zachytávací a porovnávací mód. Porovnávací mód je vybrán, když bit CAP = 1. Používá se k zaznamenávání časových událostí. Například k měření frekvence. Zachytávací vstupy CCIxA jsou připojeny k externím pinům, nebo spojeny s interními signály. Konkrétní výběr uskutečňujeme CCISx bity. CMx vybírá sestupnou, nebo nástupnou zachytávací hranu. Pokud je zachycena vybraná hrana signálu, hodnota 16 bitového čítače TAR se zkopíruje do záchytného/porovnávacího registru TACCRx a vlajka přerušení CCIFG se nastaví. K zachytávání signálů, které jsou asynchronní k hodinovým cyklům MCU, lze použít synchronizaci.

Porovnávací mód (při CAP = 0) slouží ke generování výstupního signálu PWM, nebo ke generování přerušení ve specifických časových intervalech. Při přetečení časovače se nastaví výstupní jednotka v závislosti na použití jednoho z 8 operačních módů. Definují se bity OUTMODx.

2 SOFTWARE

Obsahem kapitoly je popis řešení softwarových částí knihoven pro periferie, s vysvětlením jejich funkcí.



Obr. 2.1: Závislost vytvořených knihoven

2.1 Code Composer Studio

Je integrované vývojové prostředí podporující mikroprocesory firmy Texas Instruments. Obsahuje editor kódu C++, kompilátor s optimalizací, debugger a mnoho dalších nástrojů. Účinným pomocníkem je také možnost krokování programu přímo v mikroprocesoru s vývojovým kitem Launchpad, přes který se také nahrává strojový kód do mikroprocesoru prostřednictvím rozhraní JTAG.

2.2 Knihovna pro I^2C

Knihovna zprostředkovává ovládání sběrnice I^2C jako samostatnou vrstvu pro další aplikace a knihovny. Obsahuje funkce pro inicializaci, přenos a příjem pole dat a je vhodná pro přenášení velkých objemů dat, například u EEPROM pamětí. O nastavení sběrnice se stará funkce *i2c_init*. Další funkce pro vysílání a příjem jsou: *i2c_write*, *i2c_write_array_to_mem* a *i2c_memory_receive*. Příjem je vždy spojen se zasláním adresy registru Slave zařízení, ze kterého budeme číst.

2.2.1 Funkce *i2c_init*

Spravuje potřebné počáteční nastavení pro správné fungování I^2C sběrnice.

K inicializaci sběrnice je třeba nastavit:

- softwarový reset
- přiřazení pinů k modulu USCI_B0
- nastavení MCU jako Master a synchronní mód
- nastavení adresy zařízení Slave
- zdroj hodinového signálu (SMCLK)
- předděličku hodinového signálu a tím rychlost sběrnice
- přerušení pro příjem a vysílání dat

Nastavení požadovaných hodnot provedeme zápisem do kontrolních registrů UCB0CTL0 a UCB0CTL1. Budeme používat 7 bitovou Slave adresu, což je také její nejčastější délka. Tu zapíšeme do UCB0I2CSA. 16 bitovou hodnotu předděličky rozdělíme do dvou registrů; LSB do UCB0BR0 a MSB do UCB0BR1. Celé nastavení se děje pod softwarovým resetem sběrnice, který po inicializaci vynulujeme. To vyvolá nastavení vlajky přerušení pro vysílání k indikaci připraveného vysílacího bufferu UCB0TXBUF. Zápis funkce vypadá následovně:

```
void i2c_init(unsigned char slave_address, unsigned int divider)
```

Při volání funkce se do parametru *slave_address* předá 7 bitová adresa podřízeného zařízení Slave a do divider hodnotu předděličky pro SMCLK. Frekvence SMCLK závisí na nastavení frekvence procesoru. Předpokládejme, že je nastavena na 1 MHz a hodnota předděličky je 10. Potom výsledná frekvence hodinového signálu pro sběrnici I^2C bude 100 kHz. Je možno zadat do proměnné předděličky i parametry pro přednastavenou volbu:

- I2C_FREQ_100KHZ
- I2C_FREQ_200KHZ
- I2C_FREQ_400KHZ

Velikost frekvence SMCLK je třeba definovat v hlavičkovém souboru, aby tyto parametry splnily svůj účel. V této funkci není povolení přerušení pro příjem a vysílání, nastavuje se v každé funkci zvlášť.

2.2.2 Funkce *i2c_write*

Funkce zahájí přenos, pošle určený počet bajtů a přenos ukončí. Zápis funkce s parametry:

```
void i2c_write(unsigned char byteCount, unsigned char *array_tx)
```

V parametru se předává počet bajtů, které se mají přenést. Počet se zkopíruje do softwarového čítače *byteCtr*. Ukazatel na první znak odesílaného pole se předává

ukazatelem. Je možné poslat i jediný znak a ukazatel může uchovávat adresu proměnné typu *unsigned char*. Jakmile je vyvoláno přerušení po odeslání počáteční podmínky, do vysílacího bufferu UCB0TXBUF se přenese první prvek odesílaného pole a ukazatel na něj se inkrementuje a čítač odeslaných bajtů se dekrementuje. To se opakuje do doby, kdy je přerušení vyvoláno po posledním odeslaném bajtu. V této chvíli nezbývá žádný byte k odeslání a následuje ukončovací podmínka se zakázáním dalšího přerušení.

2.2.3 Funkce *i2c_write_array_to_mem*

Funkce zapisuje pole nebo hodnotu proměnné na určitou adresu vnitřní struktury cílového zařízení. Adresa může být 1 nebo 2 bajtová, což je nutno předem definovat v samotné knihovně. Zápis je následující:

```
void i2c_write_array_to_mem(unsigned char byteCount, unsigned int
mem_address, unsigned char *array_tx)
```

Do parametru funkce se předává jako v předchozích funkcích počet odesílaných bajtů, ukazatel na pole nebo proměnnou a navíc počáteční adresa paměti zařízení, do které se bude zapisovat. V případě, že se jedná o registr, bude odeslaný pouze jeden byte. Funguje stejným způsobem jako předchozí funkce s tím rozdílem, že v jedné sekvenci je nejprve odeslána 1 – 2 bajtová adresa a poté data.

2.2.4 Funkce *i2c_current_receive*

Funkce přijme jediný znak následovaný NACK a STOP podmínkou bez použití přerušení.

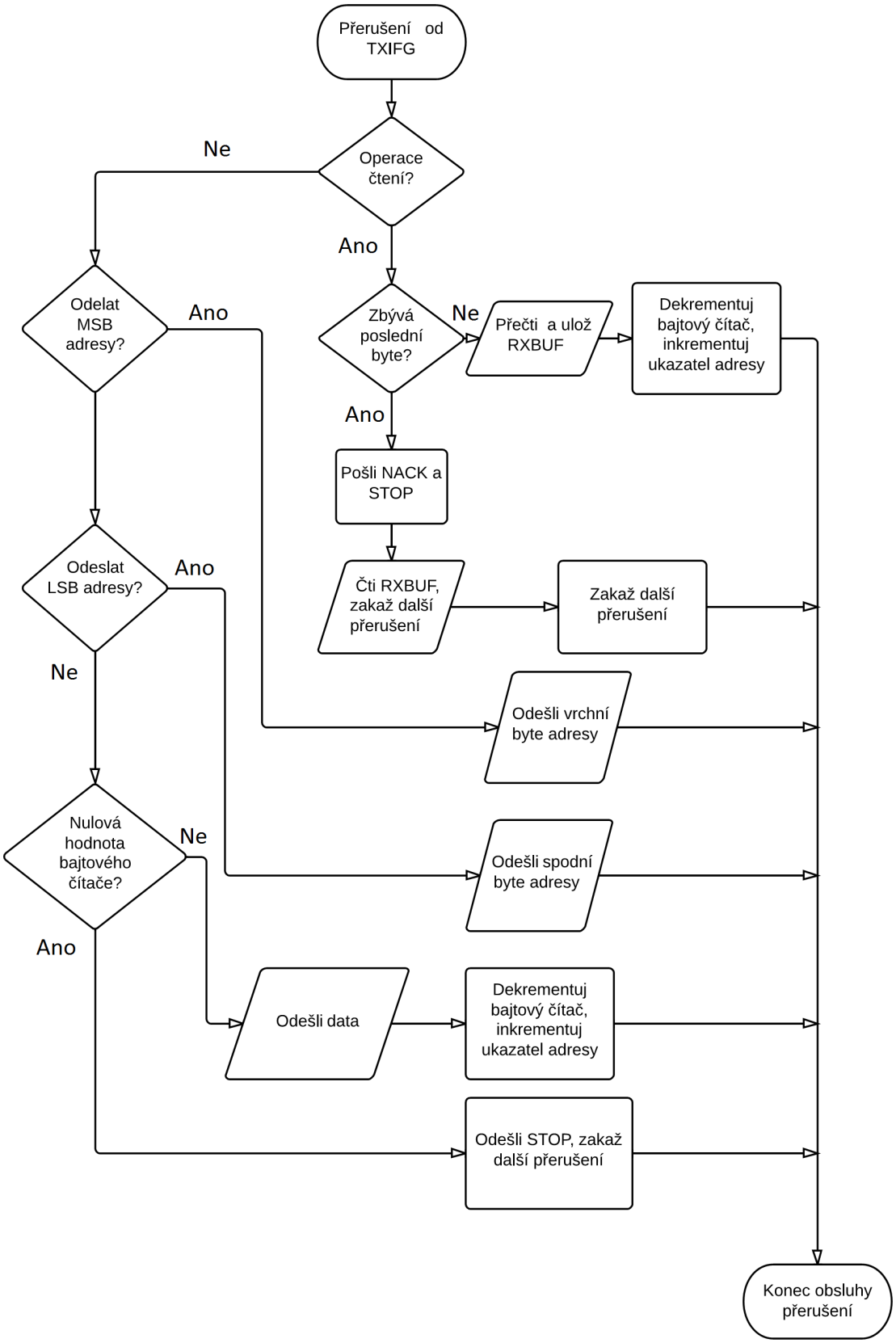
2.2.5 Funkce *i2c_memory_receive*

Přijme data, nebo pole dat od určité adresy vnitřní paměti zařízení. Opět je délka adresy volitelná 1 – 2 bajty. Vlastní fungování je rozděleno na dvě části. Za prvé vyslání počáteční podmínky s nastaveným bitem pro zápis, odeslání počáteční adresy paměti. Následuje opakovaná start podmínka s nastaveným bitem pro čtení. Po každém přijmutí bajtu do přijímacího bufferu UCB0RXBUF se vyvolá přerušení, ve kterém je jeho hodnota čtena a ukládána do pole, jehož adresa je uložena v ukazateli na pole, nebo proměnnou. To se opakuje do té doby, než je vyvoláno přerušení s posledním bytem ke čtení. Je třeba nejprve vyslat ukončovací podmínku a teprve potom buffer přečíst, jinak by zařízení Slave poslalo další byte. Zápis funkce je podobný jako pro čtení:

```
void i2c_memory_receive(unsigned char byteCount, unsigned int
mem_address, unsigned char *array_rx)
```

Parametry mají stejný význam jako ve funkci pro čtení až na to, že **array_rx* je ukazatel na první prvek pole, nebo proměnnou typu *unsigned char*.

Zvláštní případ nastává při očekávání příjmu pouze jediného bajtu. Start a podmínku je nutno poslat přímo za sebou a teprve poté číst přijímací buffer. Pokud bychom poslali stop podmínku až po čtení, přijaté bajty by byly ve skutečnosti dva. Jeden v cílové destinaci ukazatele na proměnou a druhý v přijímacím bufferu čekající na přečtení.



Obr. 2.2: Vývojový diagram obsluhy přerušení knihovny I^2C

2.3 Knihovna pro SPI

Narozdíl od I^2C a UART můžeme používat pro SPI oba moduly USCI_A0 a USCI_B0. K ovládání slouží oddělené knihovny pro každý modul zvlášť. Níže je popsána knihovna pro modul USCI_B0 a některé její funkce. MCU funguje jako Master, poskytuje hodinový signál pro Slave, začíná a ukončuje transakci. K posílání a přijímání dat slouží funkce *usci_b0_spi_init*, *usci_b0_spi_exchange*, *usci_b0_spi_write* *usci_b0_spi_receive* a *usci_b0_spi_single_write*. Může pracovat s módy SPI 0 až SPI 3. a má volitelnou softwarovou kontrolu STE.

Modul USCI u MSP430 dovoluje zvolit délku znaku na 7 nebo 8 bitů a jeho posílání od MSB nebo LSB. Budeme se držet nejběžnějšího rámce 8 bitů posílaných od nejvýznačnějšího bitu MSB. Vysílací i přijímací buffer je sdílený pro celý modul USCI_B0.

2.3.1 Funkce *usci_b0_spi_init*

Ke správnému fungování SPI je třeba nastavit následující parametry:

- nastavení výstupního pinu pro STE, SIMO, SOMI a SCLK
- nastavení SPI módu, Master módu a synchronní komunikace
- nastavení předděličky SMCLK
- povolení přerušení

Celé nastavení probíhá pod softwarovým resetem, neboli nastaveným bitem UCSWRST. Po tom co je nastavení dokončeno a reset vynulován, se automaticky nastaví vlajka přerušení indikující připravenost vysílacího registru. Funkce má následující zápis:

```
void usci_b0_spi_init(unsigned int divider, unsigned char spi_mode)
```

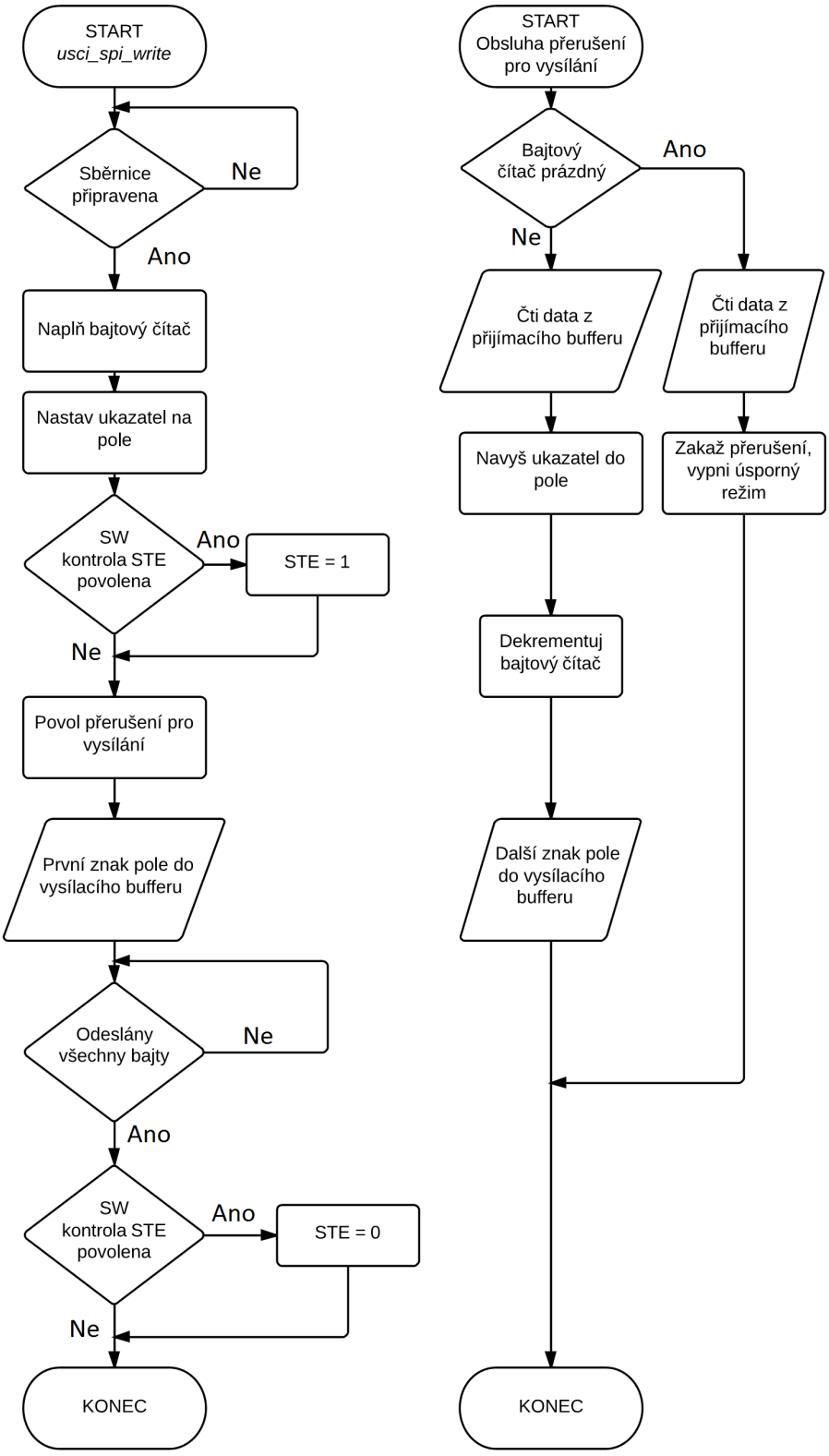
Parametr *divider* slouží k nastavení předděličky, jako v předchozí knihovně. S tím rozdílem, že nejsou předdefinovány žádné frekvence. Parametrem *spi_mode* se určuje mód, se kterým se má sběrnice inicializovat, například zápis: *SPI_MODE_0* pro mód 0.

2.3.2 Funkce *usci_b0_spi_exchange*

Ze základního principu fungování SPI sběrnice vyplývá, že abychom přijali znak, musíme nejdříve nějaký poslat. Nelze provést jedno bez druhého. Právě k tomu slouží tato funkce. Je prezentována následujícím zápisem:

```
unsigned char usci_b0_spi_exchange(unsigned char data)
```

Parametr data předává hodnotu, která se po zahájení přenosu stažením STE do nízké logické úrovně, přesune do vysílacího bufferu UCB0TXBUF. Po dobu trvání funkce je přerušení zakázáno a využívají se smyčky k čekání na nastavení přerušení. Po tom, co jsou data odvysílána, funkce vrátí hodnotu přijatou ve vysílacím bufferu ať už má nějaký význam či nikoli.



Obr. 2.3: Vývojový diagram funkce *spi_write_array* s obsluhou přerušení

2.3.3 Funkce *usci_b0_spi_write*

Tato funkce již využívá přerušení k poslání pole dat určité délky. Po tom co je předán ukazatel na pole do statické proměnné **TX_Array*, společný pro celou knihovnu a naplněn bajtový čítač *byteCtr*, se vyvolá přerušení zapsáním prvního znaku pole do vysílacího bufferu. V obsluze přerušení se posílají další prvky pole, dokud přerušení není vyvoláno posledním odeslaným bytem, kdy přečteme hodnotu vysílacího bufferu a zakážeme další přerušení. Po každém vyslaném bajtu přečteme přijímací buffer, i když data v něm nemají žádný význam, abychom vynulovali příznak přerušení pro příjem. Kdybychom ho vynulovali softwarově, nemusel by systém přerušení fungovat správně. A sběrnice by indikovala zaneprázdněný stav. Funkce je definována následovně:

```
void usci_b0_spi_write(unsigned char byteCount, unsigned char
*array_tx)
```

Parametry jsou stejné jako v knihovně I^2C. Ukazatel na první prvek pole **array_tx* a daný počet přenášených prvků *byteCount*.

2.3.4 Funkce *usci_b0_spi_receive*

Slouží k odeslání jedné proměnné. nebo celého pole o dané velikosti. K přijmutí jednoho bajtu vždy odešleme po sběrnici celý byte jedniček (hodnota 255). Software pro obsluhu přerušení je velmi podobný funkci *usci_b0_spi_write*. Smysl parametrů funkce je již znám z předchozí knihovny:

```
void usci_b0_spi_receive(unsigned char byteCount, unsigned char
*array_rx)
```

2.4 Knihovna pro UART

Dvě oddělené knihovny *uart.h* a *uart_init.h* společně dokáží komunikovat s PC přes USB převodník LaunchPad kitu. Umožňuje odeílání a přijímání dat a je uzpůsobena pro práci s terminálem PC. Samostatná knihovna pro inicializaci slouží k nastavení běžných hodnot symbolové rychlosti v závislosti na výběru kalibrovaného kmitočtu CPU, který se definuje v hlavičkovém souboru. Níže jsou popsány některé funkce.

2.4.1 Funkce *uart_send_array*

Pokud je sběrnice UART inicializována, pošle tato funkce z pole proměnných daný počet prvků. Princip je stejný jako u předchozích knihoven s tím rozdílem, že pro zahájení komunikace stačí prosté naplnění vysílacího bufferu daty.

2.4.2 Funkce *uart_send_string*

Jednoduchá funkce posílá textový řetězec na konzoli PC vložený jako argument funkce

2.4.3 Funkce *uart_send_crlf*

Funkce odešle dva ASCII znaky hodnoty 13 a 10. Na terminálu dojde k odřádkování a

další data budou zapisována na nový řádek.

2.4.4 Funkce *uart_receive_array_stop_cond*

Může se stát, že budeme potřebovat přijmout textový řetězec o neznámé délce, například větu. Funkce přijímá data, dokud jejich počet nedosáhne maximální délky řetězce, nebo dokud není přijat ukončovací znak. Ten je volitelný a může jím být například tečka. Poslední znak se do pole přijímaných znaků nezapíše. Zápis funkce:

```
char uart_receive_array_stop_cond(unsigned char maxByteCount,
unsigned char *array_rx, unsigned char stop_char)
```

Do parametru *stop_char* se zapíše ukončovací znak. Maximální možný počet přijatých bajtů určuje *maxByteCount* a ukazatel na první prvek pole, do kterého se bude zapisovat přijímaný řetězec se předá ukazatelem **array_rx*. Funkce vrací počet přijatých znaků.

2.4.5 Knihovna pro inicializaci UART

Pro správné fungování 2 zařízení je nastavení sběrnice velmi jednoduché. Stačí konfigurovat piny pro vysílání a příjem, a zvolit zdroj hodinového signálu s předděličkou. Jediná funkce knihovny má následující zápis:

```
void uart_init(unsigned long int BaudRate)
```

Do vstupního parametru *BaudRate* se předávají hodnoty symbolové rychlosti předem definované v hlavičkovém souboru. Frekvence mikroprocesoru se definuje ještě před vlastním překladem kódu a preprocesor vybere jen ty části, které potřebuje pro nastavení předděličky.

2.5 Knihovna pro EEPROM AT25xxx

Slouží pro samotné ovládání paměti, o komunikaci rozhraním SPI se stará vrstva z knihovny pro SPI. Knihovny jsou 2, každá zvlášť pro jeden modul USCI, kompatibilní s mikroprocesory řady MSP430Gxx3, které USCI modul mají. Níže bude popsána knihovna využívající modul USCI_B0. Jsou dvě možnosti připojení paměti, každá pro jeden modul:

Tab. 4: Možné zapojení paměti AT25xxx k MSP430

	USCI_A0	USCI_B0
SIMO	P1.2	P1.7
SOMI	P1.1	P1.6
CLK	P1.4	P1.5
STE	(P1.5)	(P1.4)

Protože používáme 3-wire mód modulu, STE je v knihovně ovládán softwarově. Můžeme volitelně pro tuto funkci zvolit i jiný pin a definovat ho v hlavičkovém

souboru. Jak naznačují x v názvu této kapitoly, většina EEPROM pamětí řady AT25 se liší jen jejich velikostí paměti. V hlavičkovém souboru lze zvolit její velikost a velikost čtecí stránky k tomu, aby knihovna byla kompatibilní s danou pamětí. Knihovna umožňuje zápis a čtení z paměti nebo stavového registru, mazání a hledání volného místa v paměti. Navíc umožňuje nastavit sektory chráněné před zápisem.

2.5.1 Funkce *at25_uscib0_write_cmd*

Úkolem funkce je odeslat příkaz WRDI nebo WREN. Oba příkazy manipulují se zámkem proti zápisu. Příkaz WREN pro povolení zápisu je třeba zadávat před každým zápisem do paměti, nebo stavového registru. Jinak se zápis neprovede. Zápis funkce:

```
void at25_uscib0_write_cmd(unsigned char op_cmd)
```

Parametr *op_cmd* slouží k zadání operačního kódu ve formátu: "WRDI" nebo "WREN". Funkce zkontroluje připravenost sběrnice SPI, připravenost zařízení k zápisu čtením stavového registru, zahájí přenos stažením STE do nízké logické úrovně, pošle operační příkaz a ukončí komunikaci.

2.5.2 Funkce *at25_uscib0_send_address*

Funkce slouží pouze k odeslání vrchního a poté spodního bajtu adresy paměťové buňky. Sama o sobě nezahajuje ani neukončuje komunikaci.

2.5.3 Funkce *at25_uscib0_read_status_register*

Návratovou hodnotou funkce je obsah stavového registru EEPROM paměti. Ten obvykle budeme číst z důvodu zjištění stavu zařízení, zda je připraveno k dalším operacím. Funkce má následující zápis:

```
unsigned char at25_uscib0_read_status_register(void)
```

Návratová hodnota typu unsigned char obsahuje hodnotu stavového registru. Čtení stavového registru je také jediná povolená operace po čas trvalého zápisu dat do paměti. Návratová funkce se bude v programu dále testovat, zda je nultý bit RDY v log. „1" a zápis tak neprobíhá. Stavový registr také uchovává informaci o oblastech paměti, ve kterých je zakázán zápis, jak vidíme na Obr. 2.3.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WPEN	X	X	X	BP1	BP0	WEN	$\overline{\text{RDY}}$

Obr. 2.4: Stavový registr paměti AT25xxx

2.5.4 Funkce *at25_uscib0_write_status_register*

Zapisování do stavového registru zprostředkovává tato funkce. Pomocí ní lze nastavit např. chráněné oblasti proti zápisu a povolení samotného zápisu. Ke své funkci nevyužívá přerušení a po odeslání operačních kódů pro povolení zápisu pošle novou

hodnotu, která se zapíše do registru.

2.5.5 Funkce *at25_uscib0_write_array*

Zapíše daný počet bajtů z pole předávaného ukazatelem od počáteční adresy paměti. Ve funkci je řešen zápis více bajtů, než umožňuje jedna stránkovací sekvence. V takovém případě se rozdělí data na několik stránek, v našem případě po 32 bajtech, a zapisují se v oddělených sekvencích. Také se testuje, zda nedojde k přetečení adresního čítače v průběhu zápisu. Tedy zda nejvyšší adresa paměťové buňky, do které budeme zapisovat neleží mimo rozsah adres paměti. V takovém případě by se interní adresní čítač překlopil na první adresu paměti a došlo by k přepsání dat. Zápis funkce:

```
char at25_uscib0_write_array(unsigned char byteCount, unsigned long
int mem_address, unsigned char *array_tx)
```

Do parametru se předá ukazatel na pole a počet bajtů, které budeme posílat a počáteční adresu. Funkce vrací hodnotu char, která ukazuje, zda došlo k úspěšnému zápisu, nebo se zápis neprovedl, protože by došlo k přetečení paměti.

2.5.6 Funkce *at25_uscib0_read_array*

Počet čtených bajtů není nijak omezen velikostí stránky a tudíž můžeme přečíst obsah paměti v jedné sekvenci. Po zkontrolování připravenosti zařízení a možného přetečení paměti se zahájí přenos, pošle se operační kód pro čtení a o vše ostatní se postará programová smyčka s odesíláním hodnoty FFh k přijmutí bajtu. Přijímaný byte se zapíše do daného prvku pole, ukazatel se inkrementuje a bajtový čítač dekrementuje. Po dokončení příjmu se hodnota na výstupní lince opět zvýší na vysokou logickou úroveň. Zápis funkce je následující:

```
char at25_uscib0_read_array(unsigned char byteCount, unsigned long
int mem_address, unsigned char *array_rx)
```

Význam parametrů je stejný jako v předchozí funkci. Přerušení USCI je zakázáno, neboť ji ke své správné funkci nepotřebuje.

2.5.7 Funkce *at25_uscib0_set_write_access*

Pomocí bitů BP0 a BP1 ve stavovém registru se nastavuje jedna ze čtyř úrovní ochrany. Chránit lze celou paměť, vrchní čtvrtinu, vrchní polovinu, nebo funkci vypnout, jak lze vidět v Tab.5 pro konkrétní paměť AT2526A. Funkce zprostředkuje zapsání těchto bitů do stavového registru a ověření, zda byly zapsány správně. Zápis funkce:

```
char at25_uscib0_set_write_access(char select_array)
```

Pokud přečtená data ze stavového registru souhlasí se zapsanými, návratová hodnota bude tento stav odrážet. Jinak vrátí informaci o nezdařilém zápisu. Parametr *select_array* je předdefinovaný v hlavičkovém souboru pro konkrétní hodnoty ochranných bitů BP0 a BP1.

Tab. 5: Nastavení bitů BP1 a BP0 pro chráněné oblasti AT25256A

Úroveň	BP1	BP0	Rozmezí adres	Parametr funkce
0	0	0	-	NONE
1 (1/4)	0	1	3000 – 3FFFh	QUARTER_TO_TOP
2 (1/2)	1	0	2000 – 3FFFh	MIDDLE_TO_TOP
3 (Celá paměť)	1	1	0 – 3FFFh	ALL

2.5.8 Funkce *at25_clear_array*

Protože AT25xxx nemá vestavěnou funkci mazání paměti, je toto nutno řešit jiným způsobem. Vybraný sektor paměti smažeme zapsáním nulových hodnot do paměťových buněk. K tomu slouží funkce zapsaná takto:

```
char at25_clear_array(unsigned int byteCount, unsigned long int
mem_address)
```

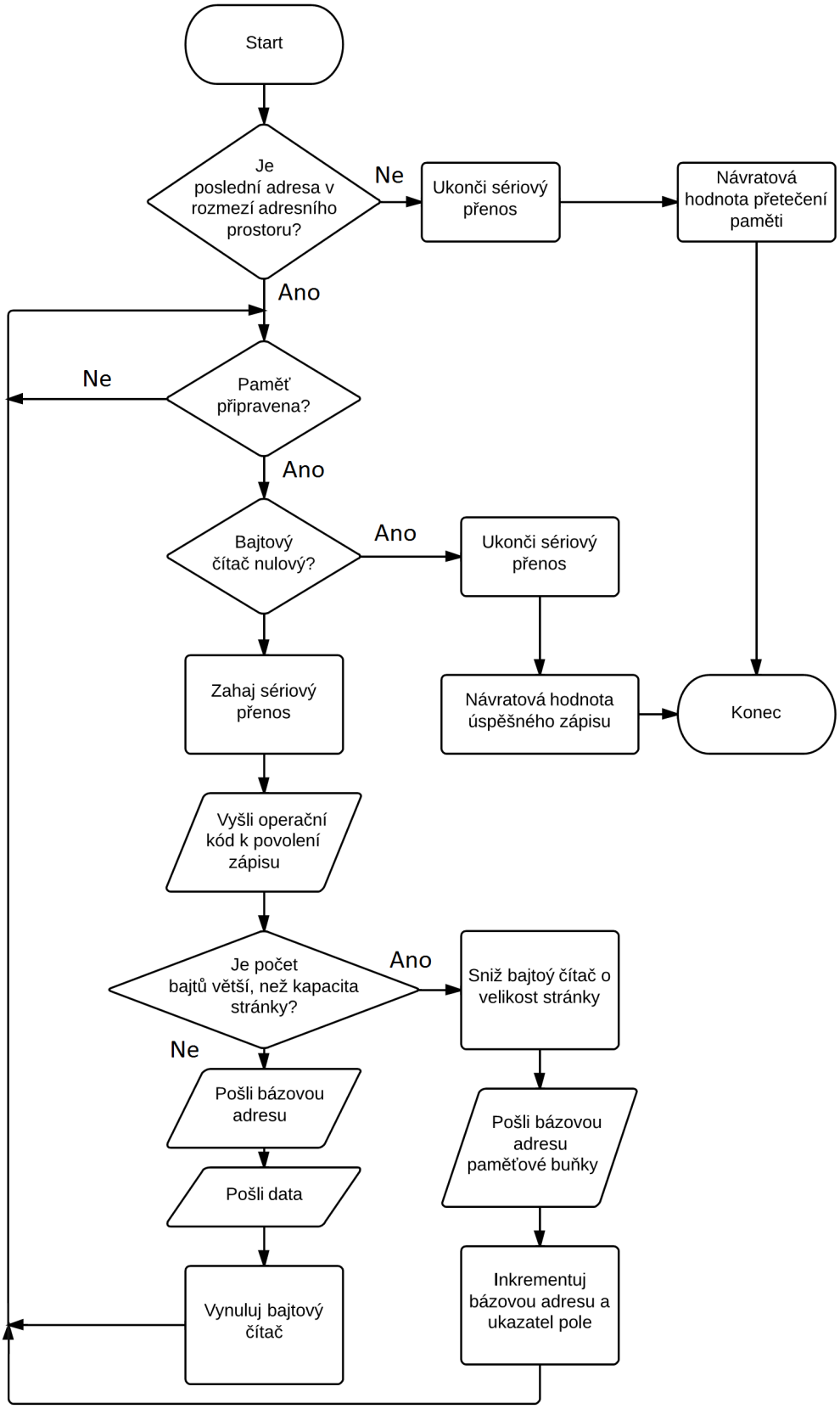
Parametrem předáme bázovou adresu sektoru paměti a počet požadovaných buněk k vymazání. Pokud počet bajtů přesáhne maximální kapacitu zapisovací stránky, bude mazání rozděleno do několika sekvencí. Funkce vrací v případě úspěšného smazání, pokud sektor nebyl chráněn proti zápisu, hodnotu OPERATION_COMPLETE. V případě že bázová adresa a počet bajtů přesáhne velikost paměti, vrátí hodnotu MEMORY_OVERFLOW.

2.5.9 Funkce *at25_find_empty_addr*

Pokud přistupujeme k paměti a není nám známo kam proběhl poslední zápis, můžeme zjistit volnou adresu pomocí této funkce. Za předpokladu, že volné místo buňky paměti uchovávají nulovou hodnotu. Funkce prohledává pole a když zjistí volnou paměťovou buňku, za kterou se nachází daný počet dalších volných buněk, vrátí adresu této buňky. Funkce je zapsána následovně:

```
unsigned long int at25_find_empty_addr(unsigned long int
mem_address, unsigned char num_of_empty_bytes)
```

kde *mem_address* je bázová adresa paměti, od které je hledání zahájeno a *num_of_empty_bytes* je počet požadovaných volných paměťových buněk za první nalezenou adresou.



Obr. 2.5: Vývojový diagram funkce *at_25_uscib0_write_array*

2.6 Knihovna pro expandér MCP 23016

Knihovna zprostředkovává základní funkce pro zápis a čtení do registrů expanderu. Změna výstupních logických úrovní portů se provádí zápisem do příslušného registru. Nejprve je nutno nastavit piny expanderu jako výstupní. To zajišťuje funkce *mcp23016_init*. K zápisu do libovolného registru slouží funkce:

```
void mcp23016_reg_write(unsigned char reg_address, unsigned char
reg_data)
```

Do parametru se předává název registru definovaný v hlavičkovém souboru, např. GP0, GP1, OLAT0 atd. Do registru budou zapsána data předané do parametru *reg_data*. Obdobně probíhá čtení z registru. Knihovna komunikuje s expanderem pomocí knihovny *i2c.h*. Adresa Slave zařízení s uzemněnými adresními piny má hexadecimální hodnotu 20h. Maximální frekvence, se kterou může komunikovat pomocí sběrnice je 400 kHz. To předpokládá správnou funkci externího oscilátoru expandéru.

2.7 Knihovna pro 8x8 LED maticový displej

Knihovna umožňuje zobrazit běžící text složený z textových znaků. Tyto znaky jsou uloženy v poli o známé délce. Rovněž zobrazení libovolného obrazce 8x8 pixelů. Jednotlivé sloupce jsou přenášeny přes sběrnici I^2C do expandéru MCP23016 a vrstva I^2C je již řešena v dříve uvedených knihovnách. Porty expandéru musí být nastaveny jako výstupní.

2.7.1 Funkce *show_screen*

Funkce zobrazí celý LED displej z 8 sloupců uložených ve statickém poli *screen*. Protože zapojení odpovídajících pinů expanderu a sloupců displeje by způsobilo zbytečné křížení na DPS, jsou zapojeny v opačném pořadí. Sloupce tedy musí být uloženy v poli *screen* v opačném pořadí od MSB. Binární hodnota sloupce odpovídá konkrétním diodám, které mají se mají rozsvítit v aktivním sloupci. Tu posíláme na port GP1 expanderu. Údaj o tom, který sloupec je aktivní posíláme na port GP0. Z důvodu prodlevy mezi odesláním jednotlivých bajtů, nedochází k přepínání aktivního sloupce zároveň s údaji dat pro jednotlivé sloupce. Je třeba poslat na oba porty expandéru log. „0" po každém zobrazeném sloupci. To zamezí nežádoucímu "přeblikávání" sloupců.

2.7.2 Funkce *scrolling_text*

Zobrazí textový řetězec uložený v poli formou běžícího textu na displeji. Zápis funkce:

```
void scrolling_text(unsigned char count, unsigned char
*string_array)
```

Do parametru **string_array* předáme ukazatel na pole, v němž je uložen textový řetězec. Číslo předané v proměnné *count* představuje počet znaků z pole, které se zobrazí. Když je konkrétní znak pole dalším v pořadí k zobrazení na displeji, převede se jeho ASCII hodnota na 8 sloupců odpovídající výslednému zobrazenému znaku na

displeji. Tyto sloupce se nahrají do softwarového vyrovnávacího bufferu *char_buff* o velikosti 16 bajtů. Použitím bufferu o velikosti 2 znaků usnadňuje přehlednost celé funkce. V Tab. 6 je znázorněn algoritmus zobrazení běžícího textu z tohoto bufferu.

Tab. 6: Algoritmus zobrazení běžícího textu

Sloupce 1. znaku								Sloupce 2. znaku								Cyklus:	Akce:
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8		
Sloupce displeje:																1.	
1	2	3	4	5	6	7	8									2.	Nahrání 2. znaku
	1	2	3	4	5	6	7	8								3.	
		1	2	3	4	5	6	7	8							4.	
			1	2	3	4	5	6	7	8						5.	
				1	2	3	4	5	6	7	8					6.	
					1	2	3	4	5	6	7	8				7.	
						1	2	3	4	5	6	7	8			8.	
							1	2	3	4	5	6	7	8		9.	
								1	2	3	4	5	6	7	8	10.	Nahrání 1. znaku
8									1	2	3	4	5	6	7	11.	
7	8									1	2	3	4	5	6	12.	
6	7	8									1	2	3	4	5	13.	
5	6	7	8									1	2	3	4	14.	
4	5	6	7	8									1	2	3	15.	
3	4	5	6	7	8									1	2	16.	
2	3	4	5	6	7	8									1	17.	

Znaky se střídavě nahrávají do první, nebo druhé poloviny vyrovnávacího bufferu. Sloupce jsou postupně nahrávány do samostatného pole *screen*, sloužícího k zobrazení konkrétních vybraných sloupců.

2.7.3 Funkce *load_scroll_char*

K nahrání konkrétních sloupců do softwarového bufferu slouží právě tato funkce. V hlavičkovém souboru *ascii-95.h* je uloženo 95 ASCII znaků v dvojrozměrném poli, počínající ASCII hodnotou 32 pro mezeru a končící hodnotou 126 pro vlnku. V rozmezí těchto hodnot se nacházejí velké a malé písmena, čísla, matematické operátory aj. První index dvojrozměrného pole odpovídá číslu znaku (tj. 0 – 94) a druhé číslu sloupci, který je třeba k zobrazení celého znaku. Jsou uloženy ve formátu 5x8 bodů. Výhoda toho formátu je ve volném prostoru 3 sloupců, které budou zobrazeny na displeji jako mezera před dalším znakem a také v úspoře paměťového prostoru. Nevýhodou je obtížnější rasterizace úhlopříček znaku. V Tab.7 vidíme příklad uložení znaku "A" v poli.

Tab. 7: Rozložení pixelů formátu 8x8 ve znaku "A"

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
Hexadecimální hodnota sloupců:				
7F	88	88	88	7F

2.7.4 Funkce *load_screen*

Tuto funkci lze použít k nahrání libovolného znaku 8x8 bodů do pole *screen*, které poté můžeme zobrazit funkcí *show_screen*. Má následující zápis:

```
void load_screen(unsigned char *image_array)
```

Pole, v němž je uloženo 8 sloupců se předává ukazatelem na první prvek pole. Neprobíhá zde převod z ASCII, ale překopírování 8 sloupců *image_char* do proměnné *screen*.

2.8 Knihovna pro 24LC256

Knihovna s názvem *mcp24.h* slouží k ovládání EEPROM paměti 24LC256 a jí podobné, které mají kompatibilní ovládání a liší se pouze velikostí paměti, případně velikostí stránky. O vrstvu sběrnice I^2C, přes kterou probíhá komunikace se zařízením, řeší již dříve popsaná knihovna *i2c.h*. Funkce slouží k zápisu a čtení pole hodnot, mazání paměti a hledání volné paměťové buňky. Před začátkem užití některé z těchto funkcí je třeba inicializovat sběrnici I^2C funkcí *i2c_init*.

2.8.1 Funkce *mcp24_write_array*

U EEPROM paměti AT25xxx bylo možné zapsat počet bajtů o maximální velikosti stránky kdekoliv v paměťovém prostoru. U 24LC256 není možné zapisovat přes fyzické hranice paměti, které jsou dány násobkem velikosti stránky. Takže například zápis dvou bajtů od bázové adresy paměti 3Fh, proběhne takto:

- byte se zapíše na adresu 3Fh
- interní adresní čítač se překlopí na spodní hranici stránky adresy, tj. 0
- 2. byte se zapíše právě na tuto adresu.

To však není žádoucí. Náš záměr byl 2. byte zapsat na adresu 40h. Funkce *mcp24_write_array* toto řeší následovně:

- nalezne vrchní hranici stránky, ke které se vztahuje naše bázová adresa, od které chceme zapisovat
- porovná, zda bázová adresa a počet zapisovaných bajtů nepřesáhne vrchní hranici stránky
- pokud hranici přesáhne, zapíše pouze bajty, které nepřesáhnou hranici stránky, počet zapsaných bajtů se přičte k bázové adrese a odečte od celkového počtu zapisovaných bajtů
- pro zbytek bajtů se toto celé opakuje

Funkce má tento zápis:

```
char mcp24_write_array(unsigned char byteCount, unsigned int
mem_address, unsigned char *array_tx)
```

Bázová adresa se předává parametrem *mem_address*, počet zapisovaných bajtů *byteCount* a první prvek pole **array_tx*. V případě úspěšného zápisu vrací hodnotu MCP24_OPERATION_COMPLETE, pokud nedošlo k pokusu zapsat poslední byte mimo paměť. Jinak vrací MCP24_MEMORY_OVERFLOW.

2.8.2 Funkce *mcp24_busy*

Po odeslání dat do vyrovnávacího bufferu paměti se vnitřní ovládací logika přepne do zaneprázdněného režimu a začne s trvalým zápisem obsahu bufferu do paměti. Po tuto dobu nelze s pamětí provádět žádné operace. Pokud se budeme snažit navázat spojení se zařízením, odpoví na svojí adresu nevysláním potvrzovacího bitu ACK. Toto využijeme k vytvoření zpoždění před další operací. Protože nepřijímáme žádná data, STOP podmínka je vyslána neprodlené po odeslání START podmínky. Do stavového registru UCB0STAT se zachytí případný NACK, tedy absence potvrzovacího bitu ACK. Zápis je následující:

```
char mcp24_busy(void)
```

Funkce pouze vrací hodnotu v závislosti na přijetí, nebo nepřijetí potvrzovacího bitu ACK. V případě, že ACK přijat nebyl, funkce vrátí hodnotu MCP24_BUSY, která

je reprezentována hodnotou 1. Pokud ACK přijme, vrátí MCP24_READY, tedy 0. Testovat stav zařízení lze v jednoduché programové smyčce while.

2.8.3 Funkce *mcp24_erase_sector*

Protože paměť nemá žádný příkaz pro mazání paměti, funkce *mcp_erase_sector* toto řeší softwarově. A to tak, že přepíše vybraný sektor, charakterizovaný bázovou adresou a počtem bajtů, nulovou logickou hodnotou.

2.8.4 Funkce *at25_uscia0_find_empty_addr*

Funkce nalezne volné místo v paměti a vrátí adresu, za kterou se nachází zadaný počet prázdných paměťových buněk. Pokud žádné místo nebylo nalezeno, vrátí hodnotu MCP24_MEMORY_FULL.

2.9 Knihovna pro RTC DS1302

Knihovna obsahuje sadu funkcí pro ovládání obvodu reálného času. K 3 vodičovému rozhraní by se dalo po menších úpravách přistupovat i pomocí SPI. My však použijeme bit banging. Tudíž budeme softwarově určovat logické hodnoty vstupních a výstupních portů. Hlavičkový soubor definuje strukturu k ukládání času v BCD, nebo decimálním formátu k větší přehlednosti.Také odděluje vrstvu řízení portů od zbytku programu direktivami preprocesoru. Jednotlivé funkce mají za úkol nastavovat čas, číst čas z registrů, přistupovat do registrů a převádět mezi BCD a dekadickou hodnotou časového formátu a psát/číst z interní paměti RAM. Příklad definice struktury pro uložení údaje o čase vypadá takto:

```
typedef struct{
  unsigned char seconds;
  unsigned char minutes;
  unsigned char hours;
  unsigned char days;
  unsigned char months;
  unsigned char day_of_week;
  unsigned char years;
}TTime_BCD;
```

2.9.1 Funkce *ds1302_single_write*

Tvoří základ pro ostatní funkce. Odešle 8 bitů následujícím způsobem:

- nastaví I/O pin jako výstupní
- vybere Slave, pin CS nastaví do vysoké logické úrovně
- testuje postupně bity vysílaného znaku od LSB a při souhlasné logické úrovni bitu nastaví I/O pin do logické úrovně „1", jinak do logické úrovně „0"
- vysílaný bit přijme zařízení při přepnutí hodinového signálu do log. „0"
- hodinový signál se po časovém zpoždění přepne zpět do log. úrovně „1"

- opakuje se pro zbývající bity

2.9.2 Funkce *ds1302_single_receive*

Funkce testuje hodnotu logické úrovně na vstupu I/O pinu a v případě souhlasné logické úrovně přičte jedničku na příslušnou pozici danou pořadím přijímaného bitu. Vrací hodnotu přijatého bajtu.

2.9.3 Funkce *ds1302_reg_write*

Adresy registrů jsou již definovány v hlavičkovém souboru. Do parametru funkce předáváme jen název konkrétního registru, který obsahuje i rozlišení pro zápis nebo čtení. Mějme na paměti, že jakémukoli zápisu musí předcházet příkaz pro povolení zápisu, přesněji řečeno zápis nulové hodnoty do registru 8Eh. Obdobně funguje i funkce *ds1302_reg_receive*.

2.9.4 Funkce *ds1302_read_time*

Funkce přijme aktuální časový údaj do struktury předávané odkazem. Přijaté údaje jsou ve formátu BCD. Jeho výhodou je snadná implementace zobrazení času například na LCD. Příjem probíhá v tzv. Burst módu. Zahájení a ukončení přenosu proběhne pouze jednou za celou sekvenci. Adresy jednotlivých časových registrů není potřeba zadávat. Příkaz pro Burst mód, 8Fh vyvolá sekvenci přijímání bezprostředně po sobě následujících bajtů z registrů hodin.

2.9.5 Funkce *convert_BCD_to_DEC* a *convert_DEC_to_BCD*

Pokud s časovým údaje, přijatým z RTC potřebujeme dále pracovat, více nám vyhovuje jeho dekadická reprezentace. Například při výpočtech. K převodu mezi formáty slouží tyto 2 funkce. Převod z BCD do DEC je poněkud jednodušší. Stačí sečíst horní půlku bajtu vynásobenou deseti se spodní částí.

Opačný převod z DEC do BCD je o něco složitější. Potřebujeme zjistit počet desítek v převáděné hodnotě a ty potom přesunout do vrchní půlky bajtu. Zbylé číslo, které je menší než 10, bude umístěno ve spodní půlce bajtu. K tomu lze použít funkci modulo, se kterou vydělíme převáděné číslo deseti. Tím dostaneme zbytek po celočíselném dělení, tedy spodní byte výsledku. Vrchní byte je výsledek celočíselného dělení deseti. Je známo, že operace modulo potřebuje k provedení velké množství cyklů hodinového signálu pro CPU. Ve funkci převádění je realizována operace zbytku po celočíselném dělení cyklem, ve kterém se postupně odečítá číslo 10 ukončeným návratovou hodnotou BCD převodu. Počet odečtení čísla 10 odpovídá počtu desítek v decimálním čísle. Příklad zápisu funkce pro převod:

```
void    convert_BCD_to_DEC(TTime_BCD    *ptr_TTime_BCD,    TTime_DEC
*ptr_TTime_DEC)
```

2.10 Knihovna pro generování zpoždění

Obsahuje funkce pro generování zpoždění v řádech milisekund, nebo sekund. Využívá časovače MCU. Pro každý časovač je určena zvláštní knihovna. Protože frekvence procesoru se pohybuje v rozmezí 1 – 16 MHz (podle nastavení), potřebujeme generovat zpoždění, které bude nezávislé na této frekvenci. V repertoáru zdrojů hodinových signálů pro MSP430 se nachází například externí krystalový oscilátor o typové frekvenci 32,768 kHz. Ten slouží jako zdroj pro pomocný hodinový signál ACLK. Nastavuje se pomocí tří kontrolních registrů pro zdroje hodinových signálů BCSCTLx. Funkce negenerují přesnou hodnotu zpoždění, protože samotné nastavení a obsluha přerušení potřebuje cykly navíc.

Pokud není připojen externí krystalový oscilátor, nabízí MSP430 ještě jednu alternativu nízkofrekvenčního zdroje pro zdroj hodinového signálu a sice napěťově ovládaný nízkopříkonový oscilátor VLO. Jeho frekvence je podle datasheetu 12 kHz, ale může být v rozmezí 4 – 20 kHz. Nepřesnost je značná, ale pro generování zpoždění nám bude stačit. Příklad zápisu funkce zpoždění pro volitelný počet sekund:

```
void timer_a0_delay_s(char sCount)
```

Počet sekund se předá prostřednictvím parametru *sCount* do programového čítače. Časovač se nastaví na 1 sekundu, což znamená hodnotu 32768 do čítacího registru pro externí krystal a 12000 pro VLO. Po přetečení časovače se vyvolá přerušení, ve kterém se buď znovu nastaví čítací registr časovače, nebo je generování přerušeno.

2.11 Knihovna pro ADC10

Knihovna zprostředkovává A/D konverzi externích a interních vstupů. Předpokládáme, že vstupní signál má téměř nulovou frekvenci. Funkce umožňují převod jednoho, nebo více kanálů v sekvenci, výběr referenčního napětí a funkci pro snímání teplotního senzoru.

2.11.1 Funkce *adc10_single_read*

Funkce vrací výsledek konverze jednoho kanálu. Ke celé konverzi je třeba:

- zakázat konverzi vynulováním bitu ENC
- vybrat počet cyklů pro vzorkování
- povolit přerušení
- povolit konverzi
- vybrat zdroj hodinového signálu
- zvolit hodnotu předděličky
- vybrat kanál
- povolit vstup kanálu

Po spuštění konverze a jejím dokončení se vyvolá obsluha přerušení, ve které je čten a uložen výsledek konverze. Funkce vrací výsledek konverze.

2.11.2 Funkce *adc10_single_read_ref*

Pracuje stejně jako předchozí funkce s tím rozdílem, že můžeme parametrem zvolit zdroj referenčního napětí. Funkce má zápis:

```
unsigned int adc10_single_read_ref(char channel, unsigned char reference)
```

Vrací výsledek 10 bitové konverze úrovně analogového signálu na vybraném kanálu.

2.11.3 Funkce *adc10_read_temp_sensor*

Funkce převede hodnotu napětí na integrovaném teplotním senzoru na stupně Celsia a tu přenese do návratové hodnoty funkce. Perioda vzorkování musí být větší, než 30 μs. Senzor má velkou offsetovou chybu, k přesné absolutní teplotě by byla třeba kalibrace. Pro výpočet teploty z 10 bitového výsledku konverze použijeme vztah (1.4) z uživatelské příručky[2]. Napěťový offset je 0,986 V a přírůstek napětí je 3,55mV/°C.

$$t = \frac{0.00355 * TEMP_C + 0{,}986}{1024} [°C] \tag{1.5}$$

kde $TEMP_C$ [-] je 10 bitový výsledek konverze a t [°C] je výsledná hodnota teploty. Aby vzorec obsahoval jen celá čísla, upravíme ho na vzorec (1.5).

$$t = \frac{(TEMP_C - 673) * 423}{1024} [°C] \tag{1.6}$$

kde $TEMP_C$ je 10 bitový výsledek konverze [-] a t [°C] je výsledná hodnota teploty.

2.12 Demoaplikace

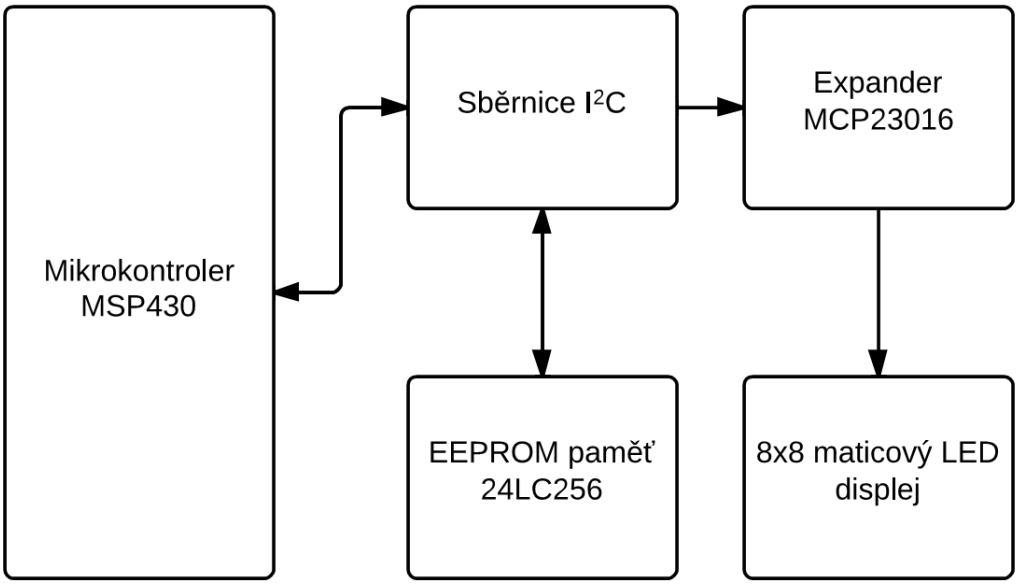
V následující kapitole budou představeny programy, které využívají výše popsané knihovny. S ohledem na velikost paměti MSP430 bylo k vytvoření demoaplikací nutno zvolit vhodné kombinace knihoven.

2.12.1 Zobrazení běžícího textu uloženého v EEPROM paměti

Demoaplikace využívá knihovny pro I^2C, LED displej, expander a EEPROM paměť 24LC256. Demoaplikace čte data uložená v EEPROM paměti 24LC256, které obsahují textový řetězec ukončený netisknutelným znakem "/n". K nahrání řetězce do paměti slouží program *LoadTextToMem*. Program text umístí na začátek paměti a jeho délka je omezena pouze velikostí pamětí. Teoreticky lze do paměti uložit 16383 znaků. Program

zapsané znaky znovu přečte z EEPROM paměti a porovná je, zda souhlasí. Pokud data v paměti odpovídají zapsaným datům, rozsvítí se červená LED dioda na LaunchPad kitu. V případě nesouladu dat začne dioda blikat.

Hlavní program postupně čte data z paměti a testuje, zda se nějaký znak nerovná "/0". V takovém případě vynuluje bázovou adresu k přístupu do paměti a pošle jen znaky umístěné před tímto znakem. Po každém čtení z paměti se zobrazí určený počet znaků na LED displeji. Použije k tomu funkci *scrolling_text* z knihovny pro LED displej. Na Obr 3.1 vidíme blokové schéma zapojení.



Obr. 2.6: Blokové schéma demoaplikace pro zobrazení textu z EEPROM paměti

2.12.2 Zobrazení animace uložené v EEPROM paměti

Demoaplikace má stejné blokové schéma jako v předchozím případě. Tentokrát ale místo běžícího textu zobrazuje postupně uložené obrazce 8x8 bodů uložené v EEPROM paměti. Do paměti lze uložit až 2047 takových obrazců. Do pole o velikosti 8 x 8 bitů se nahraje obrazec a zobrazí se na displeji. Toto se opakuje, dokud nedosáhne posledního znaku v paměti. Poté se bázová adresa vynuluje a vše začíná od začátku. K nahrání obrazců do paměti slouží program *LoadImagesToMem*.

2.12.3 Odesílání času a teploty do PC

Program využívá knihovnu pro ovládání 10 bitového A/D převodníku, sběrnici UART a obvodu reálného času DS1302. Funkce A/D převodníku *read_temp_sensor* konvertuje hodnotu integrovaného teplotního čidla na stupně Celsia. Ta je převedena na ASCII hodnotu vhodnou k zobrazení na terminálu PC. Hodnota se převede na BCD formát a přičte se číslo 48. Což je ASCII hodnota pro znak "0". Dále přečte časové registry RTC obvodu, které rovněž převede na ASCII hodnotu. Přibližně každou vteřinu provede výstup dat času a teploty. S ohledem na velikost paměti nejsou využívány standardní

funkce pro práce s řetězci. Místo toho využijeme funkci UART knihovny *uart_send_string*, která textový řetězec odešle. K odřádkování na obrazovce terminálu slouží funkce *uart_send_crlf*, která pošle znaky ASCII hodnoty "10" a "13".

Na začátku programu lze pomocí terminálu zadat datum, které se zapíše do obvodu reálného času. Tyto hodnoty se převedou z ASCII do BCD kódu a odešlou do RTC. Výstup terminálu potom vypadá takto:

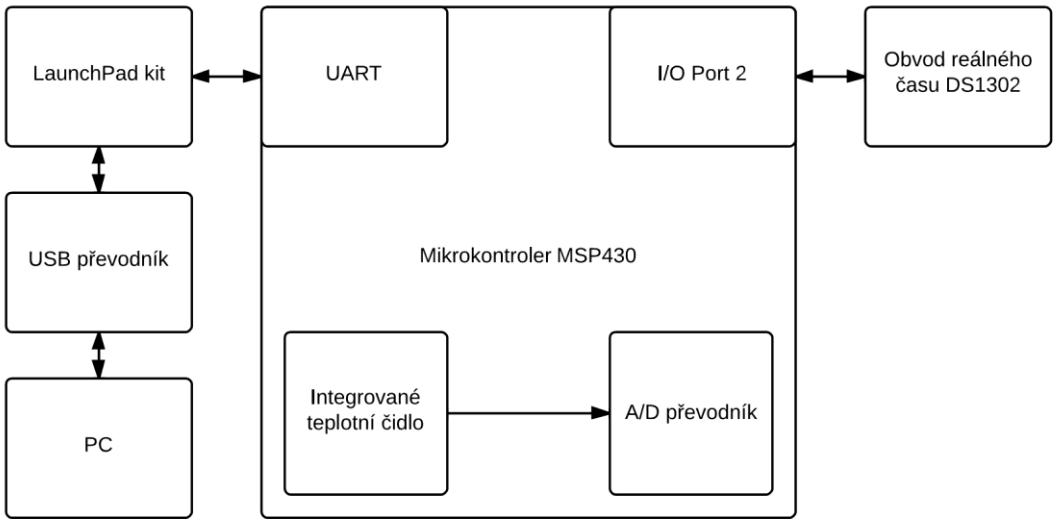
```
Demoaplikace pro posilani casu a teploty

----------------------------------------

Chcete zadat cas? (A/N): A

Zadejte vteriny (00-59):   12
Zadejte minuty (00-59):    30
Zadejte hodiny (00-23):    09
Zadejte den (01-31):       04
Zadejte mesic (01-12):     06
Zadejte den v tydnu (1-7): 03
Zadejte rok (00-99):       14

Cas:    09:30:13   Datum: 04.06.2014    Teplota: 31 oC
Cas:    09:30:14   Datum: 04.06.2014    Teplota: 31 oC
Cas:    09:30:15   Datum: 04.06.2014    Teplota: 30 oC
Cas:    09:30:15   Datum: 04.06.2014    Teplota: 30 oC
...
```

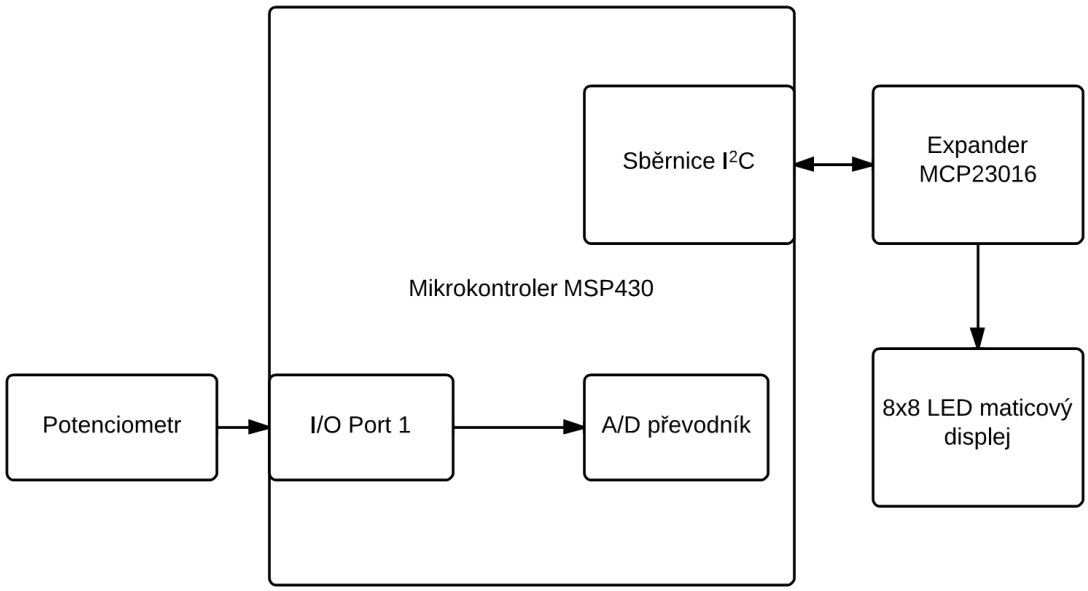


Obr. 2.7: Blokové schéma demoaplikace pro odesílání času a teploty přes sběrnici UART

2.12.4 Bargraf úrovně napětí na vstupním portu

Program provádí v čase A/D konverzi úrovní napětí na vstupním pinu MCU a tu prezentuje tyto hodnoty pak prezentuje jako bargraf na LED displeji. Úroveň napětí ovládáme pomocí potenciometru. Výsledek konverze porovnáváme s 8 intervaly rozsahu maximální hodnoty výsledku konverze. Pakliže je výsledek konverze větší, než spodní hranice intervalu, příslušný bit se nastaví do log. „1". Toto se provede pro všech 8 bitů. Výsledkem je hodnota prezentovaná výškou sloupce. Nový výsledek konverze se v tomto formátu nahraje na místo 1. sloupce LED displeje a ostatní sloupce se posunou. Takto získáme bargraf měnící se v čase.

Použitý potenciometr má hodnotu odporu 10 kΩ. K co nejpřesnějšímu měření hodnoty napětí na vstupu použijeme jako zdroj kladného napětí pro potenciometr logickou úroveň „1" na výstupu pinu P1.5. Potenciometr uzemníme na pin P1.0, který jsme definovali jako výstupní s výstupní logickou hodnotou „0". Vývod jezdce potenciometru připojíme na analogový vstup A/D převodníku kanálu 4, tedy P1.4. Mezi zem a analogový vstup připojíme navíc kondenzátor o hodnotě 10 nF. Kondenzátor sníží potřebný čas nabíjení vnitřní kondenzátorové sítě A/D převodníku a sníží kolísání napětí analogového vstupu při zahájení vzorkování.



Obr. 2.8: Blokové schéma demoaplikace pro zobrazení bargrafu úrovně napětí analogového vstupu

3 ZÁVĚR

V rámci práce bylo dosaženo podrobného seznámení s mikrokontrolery MSP430. Podařilo se realizovat knihovny pro širokou škálu periferií, ať už integrovaných v mikronotroleru, nebo pro externí součástky.

Realizovány byly knihovny pro SPI, I^2C, UART, EEPROM paměť 24LC256, EEPROM paměť AT25256, expander MCP23016, 8x8 LED maticový displej a interní A/D převodník.

Knihovny pro sběrnice fungují v režimu Master a jsou kompatibilní se všemi mikrokontrolery řady MSP430G2 obsahující USCI modul pro komunikaci. Mají volitelné parametry pro nastavení rychlosti přenosu. Tvoří komunikační vrstvu pro další knihovny.

Knihovny pro paměti zprostředkovávají čtení a zápis do paměti z jakékoli adresy a libovolným počtem bajtů, všechny možné nastavení a mazání paměti. Knihovna pro expander umožňuje ovládat vstupně/výstupní piny a přistupovat do kontrolních registrů. Knihovna pro RTC obvod nabízí funkce pro zápis a čtení kteréhokoli registru ve dvou módech.

Knihovna pro interní A/D převodník umožňuje převádět hodnotu napětí na kterémkoli externím kanálu a také na integrovaném teplotním čidlu. Úroveň napětí na tomto senzoru převede na stupně Celsia.

Dále je řešena knihovna pro LED maticový displej umožňující zobrazování volitelného obrazce, nebo textového řetězce jako běžící text. Lze nastavit dobu, po kterou bude 1 znak zobrazen.

Byly vytvořeny demoaplikace využívající tyto knihovny. Například pro posílání aktuálního času a teploty na terminál PC s možným nastavením časových registrů RTC obvodu přímo přes vstup textového terminálu. Dále demoaplikace zobrazující textový řetězec, který je uložen v EEPROM paměti jako běžící text na LED displeji, nebo animaci. V neposlední řadě demoaplikace převádějící hodnotu napětí analogového vstupu interního A/D převodníku na časově proměnný bargraf, zobrazený na LED displeji. Původní záměr využít všechny tyto knihovny zároveň v jedné demoaplikaci není realizovatelný. Mikrokontroler má omezenou paměť pro program a všechny se do ní nevejdou. Také není možné kombinovat některé náročnější knihovny, které ke svému fungování potřebují další. Například není možné zároveň využívat UART a knihovnu pro LED displej.

Dále byl navržen a sestrojen obvod pro zapojení všech periferií na desce plošných spojů. Výrobek kromě sdružovací funkce obsahuje také stabilizátor napětí a pole tranzistorů pro napájení LED maticového displeje. Některé piny jsou vyvedeny zvlášť pro připojení dalších periferií. Jumpery na desce plošných spojů slouží k připojení napájení pro periferie. Obsahuje také tlačítko RESET pro mikrokontroler. Vyvedené piny měly sloužit i k programování mikrokontroleru přímo na této desce, ale to se nepodařilo. Je tedy nutné externí programování.

LITERATURA

[1] TEXAS INSTRUMENTS. MSP430G2553 [online] 2012 Dostupný z: http://www.ti.com/lit/gpn/msp430g2553

[2] TEXAS INSTRUMENTS. MSP430 User Guide [online] 2012 Dostupný z: http://www.scribd.com/doc/59706391/MSP430-User-Guide

[3] LUECKE, Jerry. *Analog and Digital Circuits for Electronic Control System Aplications: Using the TI MSP430 Microcontroller*. 2004.

[4] Dallas semiconductor. DS1302 [online] 2004 Dostupný z: http://www.gme.cz/dokumentace/433/433-112/dsh.433-112.1.pdf

[5] DAVIES, John. *MSP430 Microcontroller Basics*. Oxford: Elsevier, 2008. ISBN 9780750682763.

[6] MICROCHIP TECHNOLOGY. *256K I2C CMOS SERIAL EEPROM* [pdf]. 1998-2013 [cit. 2.5. 2014]. ISBN 9781620776827. Dostupné z: http://ww1.microchip.com/downloads/en/devicedoc/21203N.pdf

[7] Připojení sériové paměti EEPROM. *www.dhservis.cz* [online]. 2002, 2014 [cit. 2014-05-01]. Dostupné z: http://www.dhservis.cz/eeprom.htm

[8] MICROCHIP. *MCP23016: 16-bit Expander* [pdf]. 2007 [cit. 2.5. 2014]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/20090C.pdf

[9] ATMEL. *AT25640b: Serial EEPROM* [pdf]. 2008, 24 s., 1.7. 2012 [cit. 4.5. 2014]. Dostupné z: http://www.atmel.com/Images/Atmel-8535-SEEPROM-AT25320B-640B_Datasheet.pdf

[10] Lab 12: Basics of LED dot matrix display. *Embedded lab* [online]. 2011 [cit. 2014-05-04]. Dostupné z: http://embedded-lab.com/blog/?p=2478&cpage=1

[11] BRIGH LED CORP. *BM-10EG88MD*. 3 s. Dostupné z: http://www.gme.cz/img/cache/doc/512/178/hd-m10eg88md-datasheet-1.pdf

[12] MANN, Burkhard. *C pro mikrokontroléry*. Praha: BEN, 2003. ISBN 80-7300-077-6.

[13] PINKER, Jiří. *Mikroprocesory a mikropočítače*. Praha: BEN, 2004. ISBN 80-7300-110-1.

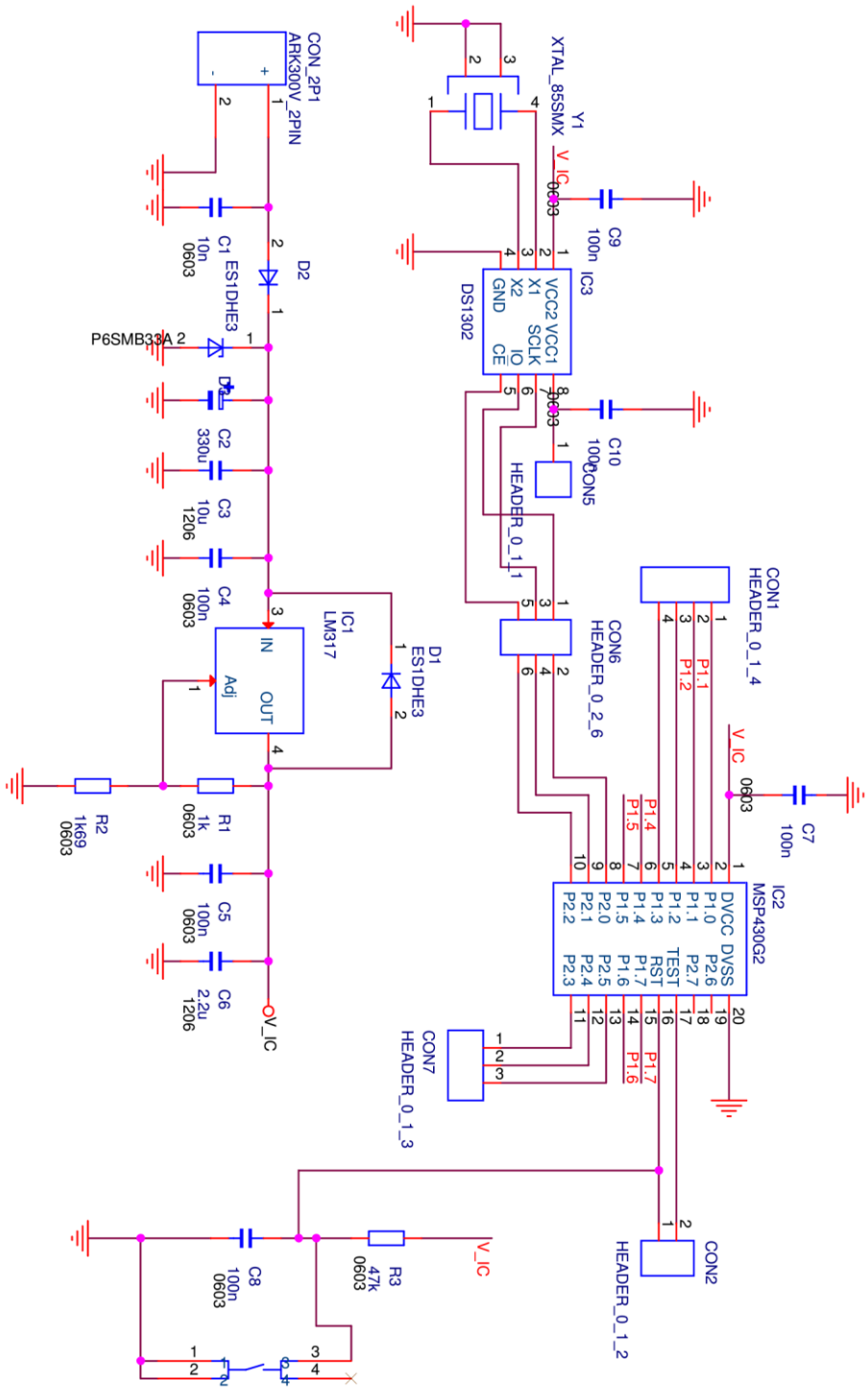
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADC	Analog to Digital converter
ASCII	Kódová tabulka pro textové znaky
CE	Linka pro výběr podřízeného zařízení
BCD	Binárně reprezentovaná dekadická hodnota
CPU	Vnitřní procesorová jednotka
DTC	Přímý přístup do paměti
DPS	Deska plošných spojů
EEPROM	Elektricky mazatelná paměť
FLASH	Elektricky programovatelná paměť typu RAM
IO	Integrovaný obvod
LSB	Nejméně význačný bit/byte
MISO/ MOSI	Master input, Slave output
MOSFET	Polem řízený tranzistor
MSB	Nejméně význačný bit/byte
PWM	Pulzní šířková modulace
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripherial Interface
STE	Linka pro výběr podřízeného zařízení
TTL	Standard pro implementaci logických obvodů
VLO	Napěťově řízený oscilátor

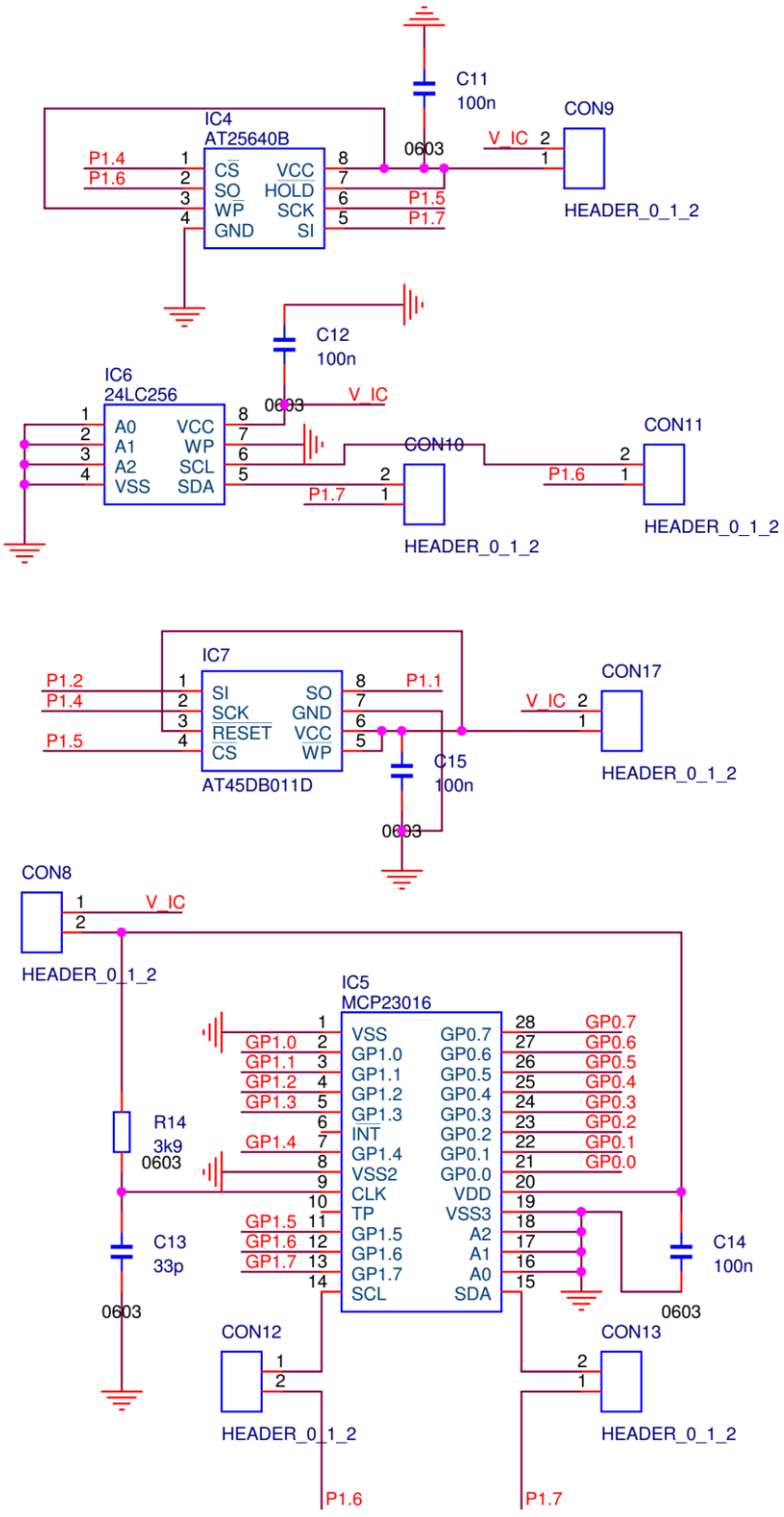
SEZNAM PŘÍLOH

A MODUL

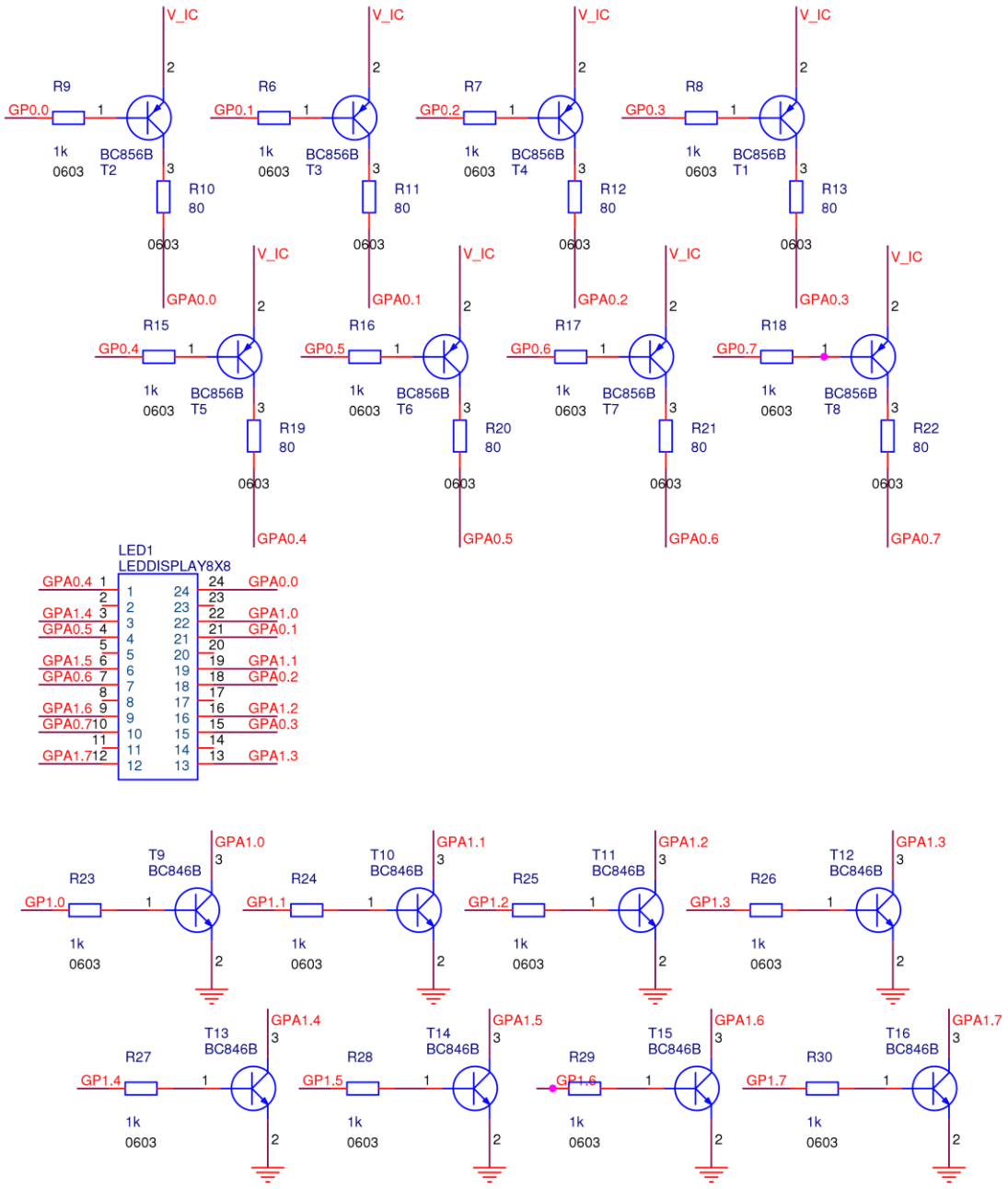
A.1 Obvodové zapojení 1. část - Napájení a MCU



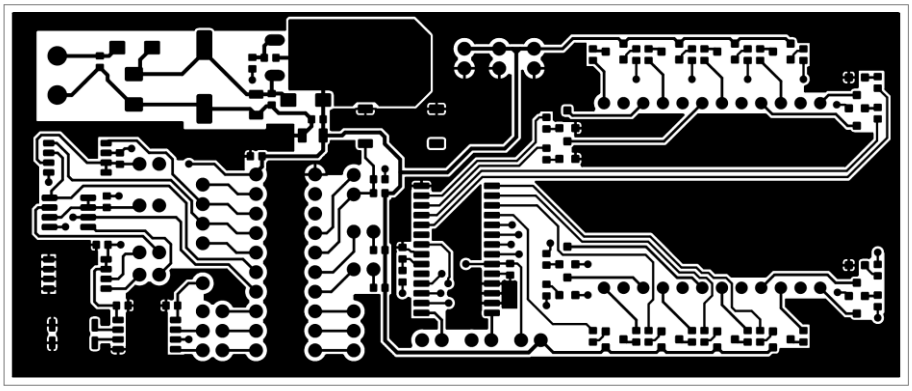
A.2 Obvodové zapojení 2. část - Integrované obvody



A.3 Obvodové zapojení 3. část - LED maticový displej

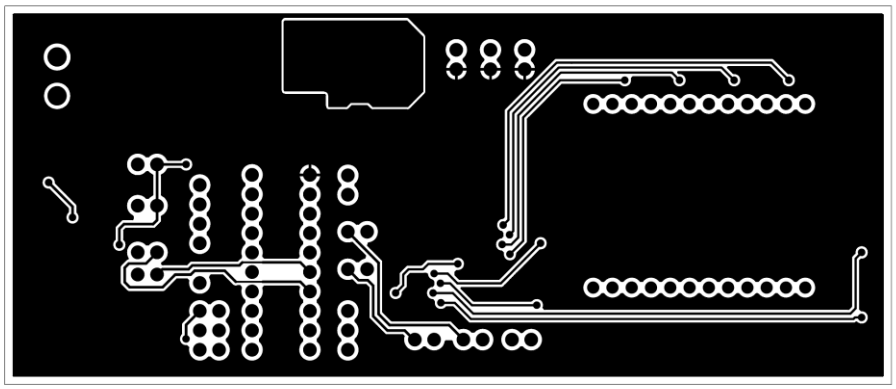


A.4 Deska plošného spoje - TOP



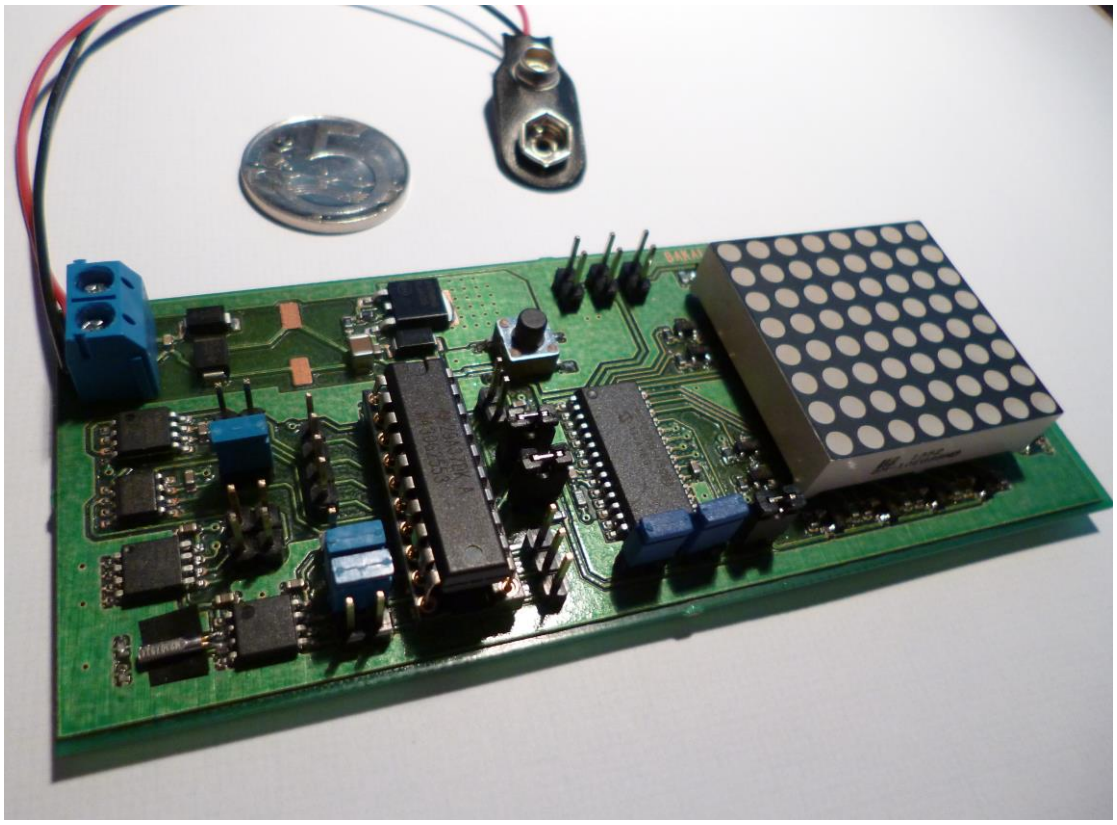
Rozměr desky 117 x 47 [mm], měřítko M1:1

A.5 Deska plošeného spoje - BOTTOM

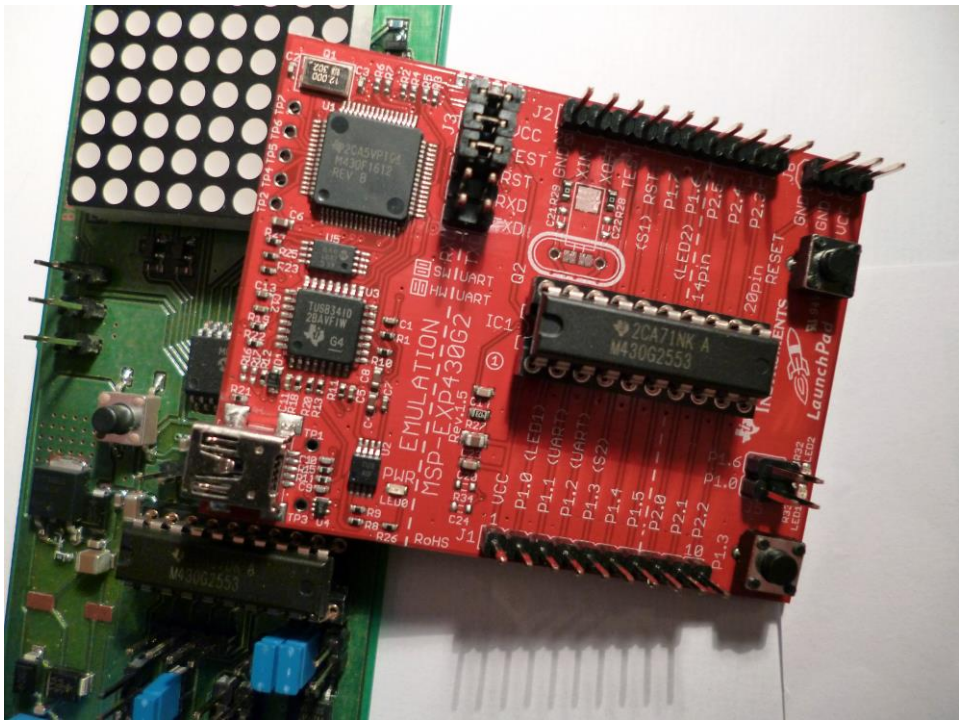


Rozměr desky 117 x 47 [mm], měřítko M1:1

A.6 Modul



A.7 Modul s LaunchPadem



B OBSAH DOPROVODNÉHO CD

Součástí práce je přiložené CD, které obsahuje následující adresáře:

/Demoaplikace	-	Obsahuje demoaplikace
/DPS	-	Podklady pro tvorbu DPS, schéma zapojení
/Knihovny	-	Zdrojové kódy knihoven
/Podpurne programy	-	Podpůrné programy pro demoaplikace