Žilinská univerzita v Žiline

Fakulta elektrotechniky a informačných technológií

Katedra mechatroniky a elektroniky

28260620202002

Programovanie vnorených systémov na báze mikroprocesorov TI C2000 v prostredí MATLAB pre autotronické aplikácie

5.5.2020

Bc. Jakub Baroniak

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA ELEKTROTECHNIKY A INFORMAČNÝCH TECHNOLÓGIÍ
KATEDRA MECHATRONIKY A ELEKTRONIKY

Programovanie vnorených systémov na báze mikroprocesorov TI C2000 v prostredí MATLAB pre autotronické aplikácie

DIPLOMOVÁ PRÁCA

Študijný program:	Výkonové elektronické systémy
Študijný odbor:	Elektrotechnika
Školiace pracovisko:	Žilinská univerzita v Žiline, Fakulta elektrotechniky a informačných technológií, Katedra mechatroniky a elektroniky
Školiteľ:	doc. RNDr. Juraj Pančík, CSc.
Konzultant:	

5.5.2020
Bc. Jakub Baroniak

ŽILINSKÁ UNIVERZITA V ŽILINE

ACERT ISO 9001

FAKULTA ELEKTROTECHNIKY A INFORMAČNÝCH TECHNOLÓGIÍ

KATEDRA MECHATRONIKY A ELEKTRONIKY

Univerzitná 8215/1, 010 26 Žilina
tel.: 041/5131601 fax:041/5131524 e-mail: kme@fel.uniza.sk

ZADANIE DIPLOMOVEJ PRÁCE

Meno: **Bc. Jakub BARONIAK**

Študijný program: Výkonové elektronické systémy

Názov diplomovej práce:

Programovanie vnorených systémov na báze mikroprocesorov TI C2000 v prostredí MATLAB pre autotronické aplikácie

Pokyny pre vypracovanie diplomovej práce:

1. Vymedzenie definícií a opis pojmov (generátory kódu, MATLAB a SIMULINK, procesory TI C2000, MATLAB Coder, Simulink Coder a Embedded Coder)
2. Analýza možností generátorov kódu na báze programovacieho prostredia MATLAB pre procesory TI C2000, analýza možností programovania komunikácií na báze CAN
3. Analýza príkladov opísaných v dokumentácii k MATLAB Embedded Coder-u
4. Návrh a realizácia vybraných príkladov, vhodných pre študijný program autotronika
5. Príprava výučbových materiálov pre učiteľa a študenta

Predpokladaný rozsah práce - počet strán textu: max. 50
počet strán grafických príloh: max.15

Školiteľ diplomovej práce: doc. RNDr. Juraj Pančík, CSc.

Konzultant diplomovej práce :

Dátum zadania diplomovej práce: **31. 10. 2019**

Dátum odovzdania diplomovej práce: **5. 5. 2020**

doc. Ing. Michal Frivaldský, PhD.
vedúci katedry

Abstrakt

Diplomová práca sa zaoberá problematikou vývoja založenom na modeli (MBD) a automatickému generovaniu kódu z prostredia MATLAB pre konkrétnu architektúru mikrokontrolérov Texas Instruments C2000. Nástroje pre vývoj sú voľne dostupné a vygenerovaný kód v jazyku C zodpovedá štandardu MISRA. V teoretickej časti sú definované pojmy vnorené systémy, automatické generátory kódu, MATLAB, mikrokontroléri Texas Instruments C2000, študijný program autotronika, štandard MISRA a funkčná bezpečnostná norma pre automobilový priemysel ISO 26262. Analytická časť sa zaoberá generovaním kódu pre procesory Texas Instruments C2000 a možnosťami programovania komunikácií. V praktickej časti sú uvedené príklady vhodné pre študijný program autotronika.

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA ELEKTROTECHNIKY A INFORMAČNÝCH TECHNOLÓGIÍ

KATEDRA MECHATRONIKY A ELEKTRONIKY

ANOTAČNÝ ZÁZNAM –DIPLOMOVÁ PRÁCA

Meno a priezvisko: Bc. Jakub Baroniak **Akademický rok:** 2019/2020

Názov práce: Programovanie vnorených systémov na báze mikroprocesorov TI C2000 v prostredí MATLAB pre autotronické aplikácie

Počet strán:	**53**	**Počet obrázkov:**	**18**	**Počet tabuliek:**	**7**
Počet grafov:	**0**	**Počet príloh:**	**17**	**Počet použ. lit.:**	**40**

Anotácia v slovenskom jazyku:

Diplomová práca sa zaoberá programovaním mikroprocesorov TI C2000 v prostredí MATLAB.

Anotácia v anglickom jazyku:

The Diploma Thesis deal with programming microprocessors TI C2000 in software MATLAB.

Kľúčové slová: Automatické generátory kódu, MATLAB, Procesory TI C2000, Vnorené systémy,

Vedúci Diplomovej práce: doc. RNDr. Juraj Pančík, CSc.

Konzultant:

Recenzent: _____

Dátum odovzdania práce: 5.5.2020

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

Skratka	Anglický význam	Slovenský význam
UART	Universal asynchronous receiver-tranmiter	Univerzálny asynchrónny prijímač alebo vysielač
SPI	Serial Peripheral Interface	Synchrónne sériové komunikačné rozhranie
I2C	Inter-Integrated Circuit	Dvojvodičová obojsmerná zbernica
CAN	Controller Area Network	Sériová zbernica využívaná hlavne v automobiloch
LIN	Local Interconnect Network	Trojvodičová sériová zbernica
USB	Universal Serial Bus	Univerzálna sériová zbernica
PWM	Pulse Width Modulation	Impulzová šírková modulácia
HRPWM	High Resolution Pulse Width Modulation	Impulzová šírková modulácia s vysokým rozlíšením
A/D	Analog to Digital Converter	Analógovo-digitálny prevodník
D/A	Digital to Analog Converter	Digitálne-Analógový prevodník
CPU	Central Processing Unit	Centrálna procesorová jednotka
GPU	Graphics Processing Unit	Grafický procesor
FPGA	Field Programmable Gate Array	Programovateľné pole logických členov
I/O	Input/Output	Vstupné/Výstupné zariadenia
RAM	Random Access Memory	Pamäť s náhodným prístupom
ROM	Read Only Memory	Pamäť len na čítanie
Flash	is a non-volatile memory chip used for storage and for transfering data	Energeticky nezávislá pamäť na ukladanie a prenos údajov
MCU	Microcontroller	Mikroradič, mikrokontrolér
FPU	Floating Point Unit	Koprocesor určený na vykonávanie matematických operácií s číslami s pohyblivou desatinnou čiarkou
VCU	Viterbi Complex math Unit	Koprocesor pre zložité matematické operácie
CLA	Control Law Accelerator	Koprocesor pre urýchlenie matematických operácií
DMA	Direct Memory Access	Priamy prístup do pamäte
MBD	Model Based Development	Vývoj založený na modeli

MIL	Model in the loop	Simulácia modelu v slučke
SIL	Software in the Loop	Simulácia softvéru v slučke
PIL	Processor in the Loop	Simulácia procesora v slučke
HIL	Hardware in the Loop	Testovanie softvéru na cieľovom hardvéri
DES	Data Encryption Standard	Algoritmus na šifrovanie digitálnych údajov
AES	Advanced Encryption Standard	Algoritmus používaný k šifrovaniu dát
MD5	Message Digest Algorithm 5	Kryptografická funkcia
VHDL	Very High Speed Integrated Circuit Hardware Description Language	Programovací jazyk pre popis hardvéru
ASIC	Application Specific Integrated Circuit	Integrovaný obvod pre špecifickú aplikáciu
DSP	Digital Signal Processor	Procesor na spracovanie signálov
PLC	Programmable Logic Controller	Programovateľný logický automat
ABS	Anti-lock Braking System	Protiblokovací systém
OTP	One Time Programmable Memory	Jednorazová programovateľná pamäť
PIE	Peripheral Interrupt Expansion	Rozširujúci blok pre prerušenia
HRCAP	High Resolution Capture	Meranie šírky impulzov vo vysokom rozlíšení
QEP	Quadrature Encoder Pulse	Počítadlo impulzov z enkodéra
McBSP	Multichannel Buffered Serial Port	Viackanálový sériový port s vyrovnávacou pamäťou
OPAMP	Operational Amplifier	Operačný zosilňovač
POR	Power on Reset	Generuje reset pri zapnutí
BOR	Brown-out Reset	Generuje reset ak napájacie napätie klesne pod povolenú hodnotu
VDD	Supply Voltage	Napájacie napätie
VDDIO	Supply Voltage for digital interfaces	Napájacie napätie pre digitálne rozhrania
ASIL	Automotive Safety Integrity Level	Úroveň integrity automobilovej bezpečnosti
AUTOSAR		
MISRA	Motor Industry Software Reliability Association	Združenie spoľahlivosti softvéru pre automobilový priemysel

kB	Kilobyte	Kilobajt
MB	Megabyte	Megabajt
Napr.	For example	Napríklad
SCI	Serial Communication Interface	Sériové komunikačné rozhranie
FOC	Field oriented control	Vektorové riadenie
Mbit/s	Megabit per second	Megabit za sekundu
ISR	Interrupt service routine	obslužná rutina prerušenia
PMSM	Permanent Magnet Synchronous motor	Synchrónny motor s permanentnými magnetmi
EEPROM	Electrically Erasable Programable Read Only Memory	Elektronicky zmazateľná pamäť ROM
DC	Direct Current	Jednosmerný prúd
AC	Alternating current	Striedavý prúd
PFC	Power Factor Correction	Kompenzácia účinníka
FIFO	First In First Out	Prvý dnu prvý von
FEIT	Faculty of electrical engineering and information technology	Fakulta elektrotechniky a informačných technológii
UNIZA	University of Žilina	Žilinská univerzita
TI	Texas Instruments	Texas Instruments
GPIO	General Purpose Input/ Output	Univerzálny Vstupný/ Výstupný pin

Zoznam symbolov

Symbol	Jednotka	Význam symbolu
U	[V]	Napätie
I	[A]	Prúd
T	[s]	Perióda
f	[Hz]	Frekvencia
t	[s]	Čas

Poďakovanie

Týmto by som chcel poďakovať konzultantovi mojej diplomovej práce doc. RNDr. Juraj Pančík, CSc., za jeho odborné vedenie a cenné rady ktoré mi pomohli pri vypracovávaní diplomovej práce.

ÚVOD

V diplomovej práci sa venujeme programovaniu vnorených systémov na báze mikroprocesorov Texas Instruments C2000 v prostredí MATLAB pre aplikácie v autotronike. Cieľom práce je overiť integráciu platformy Texas Insruments C2000 pomocou MATLAB/ SIMULINK do prostredia výučby na katedre mechatroniky a elektroniky FEIT UNIZA, navrhnúť a realizovať vhodné príklady pre študijný program autotronika a nájsť obmedzenia tejto metódy programovania oproti jazyku C.

Diplomová práca sa skladá z troch častí. Prvá časť je teoretická a sú v nej definované základné pojmy, vnorené systémy, proces návrhu softvéru pre automobily, automatické generátory kódu, prostredie MATLAB a SIMULINK, mikroprocesory Texas Instruments C2000 a študijný program autotronika. V teoretickej časti je uvedený súbor odporúčaní pre vývoj softvéru MISRA a funkčná bezpečnostná norma pre automobily ISO 26262. Druhá časť je analytická a zaoberá sa analýzou generovania kódu z prostredia MATLAB/ SIMULINK pre mikroprocesory Texas Instruments C2000 a programovaním komunikácií. Ďalej sú tejto časti popísané príklady z dokumentácie Embedded Coder-a. V tretej praktickej časti sú realizované vybrané príklady a sú zdokumentované ako výučbové materiály pre učiteľa a študenta.

Prínosom práce je návrh a realizácia príkladov vhodných pre študijný program autotronika a ich aplikácia vo výučbe.

1 DEFINÍCIE A POPIS POJMOV

1.1 VNORENÉ SYSTÉMY A ICH PROGRAMOVANIE

Vnorené systémy sú špecializované počítačové systémy, ktoré slúžia na jednu alebo viac funkcií. Tieto systémy sa často nachádzajú v spotrebnej elektronike, automatizácií, priemyselných zariadeniach, automobiloch. Základnou konštrukčnou jednotkou je mikroprocesor, ktorý je doplnený o operačnú pamäť a množinu potrebných periférií (Volný, 2011), (J.S.Freudenberg, 2019). Medzi periférie patria:

- Čítače, časovače
- Obvody typu watchdog pre kontrolu správnej funkcie systému
- Sériové rozhrania typu UART, SPI, I2C
- PWM generátory
- A/D a D/A prevodníky
- USB radiče
- Ethernet radiče
- DMA radiče pre urýchlenie prenosu medzi operačnou pamäťou a periférnym zariadením
- Šifrovacie jednotky DES, AES, MD5
- Detektory poklesu napájania

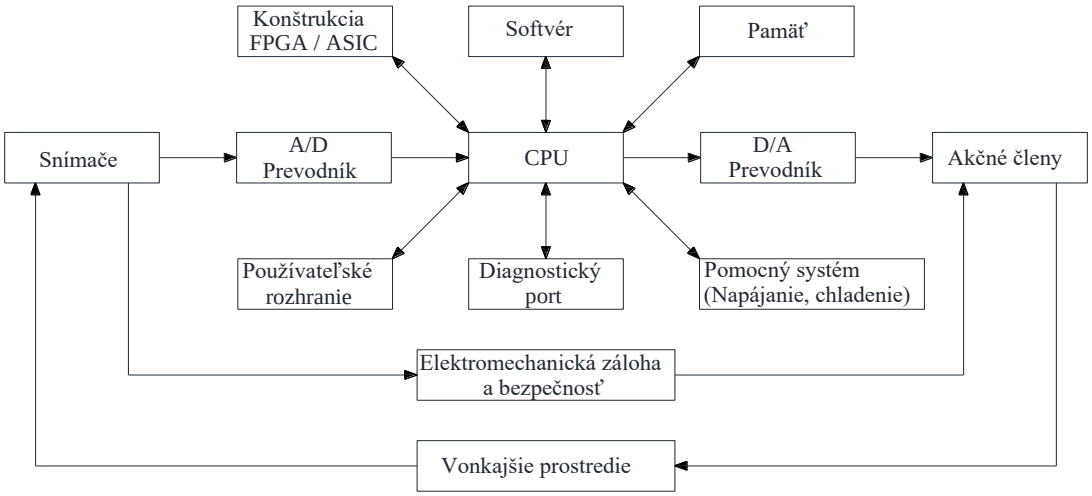
Vnorené systémy sú v dnešnej dobe z veľkej časti plnohodnotné počítačové systémy obsahujúce jeden alebo viac procesorov s 8 až 64 bitovou architektúrou, operačnú pamäť a veľa periférnych zariadení. Vnorené systémy sa od počítačových systémov všeobecného určenia líšia (Volný, 2011), (SWATHI, 2019):

- Menej výkonné procesory z hľadiska ich taktovacej frekvencie
- Málo operačnej pamäte, veľkosť sa pohybuje od jednotiek KB po stovky MB
- Veľké množstvo periférnych obvodov rôznych typov
- Snaha o minimalizáciu spotreby elektrickej energie

Pri hardvérovom návrhu konkrétneho systému je potrebné myslieť najmä na hardvérové doplnky, ktoré budú podporovať špecifické funkcie daného zariadenia. Častou chybou býva predpoklad, že mnohé funkcie, hlavne časovo závislé budú realizované pomocou programu. Toto je možné, má to však niekoľko nevýhod (Volný, 2011), (SWATHI, 2019):

- Tento prístup vyčerpáva zdroje systému, hlavne čas procesora. Ten sa potom nemôže venovať ďalším činnostiam
- Nie je možné zaručiť správny a očakávaný chod takto realizovaných funkcií, napríklad rýchle časovače, meranie času, presné taktovanie sériovej zbernice na vyšších rýchlostiach alebo vzorkovanie analógových dát.

Hlavným programovacím nástrojom sa stáva jazyk C a C++ doplnený o časti kódu, napísané v jazyku assembler konkrétneho procesora. Hlavným nástrojom programátora je prekladač jazyka C / C++ a assembler, doplnený o spojovací program (linker) a knihovník. Naviac je mnohokrát k dispozícii integrované vývojové prostredie, ktoré obsahuje editor a projektový manažér, uľahčujúce vytváranie projektov a ich správu. Tieto programové balíky bývajú tiež doplnené o podporu odladťovania, prípadne simuláciu procesora v slučke (Volný, 2011), (SWATHI, 2019).



Obrázok 1.1: Bloková schéma vnoreného systému

Zdroj: (SWATHI, 2019)

1.2 ARCHITEKTÚRY VNORENÝCH SYSTÉMOV PRE AUTOMOBILY

Softvérová architektúra je základom pre návrh softvéru pre automobily. Je to návrh dizajnu na vysokej úrovni, ktorý kombinuje viacnásobné pohľady na softvérový systém a poskytuje projektovým tímom možnosť komunikovať a robiť technické rozhodnutia o organizácií a funkčnosti softvérového systému. Umožňuje nám to pochopiť a predpovedať výkon systému ešte predtým, ako bude systém navrhnutý (Staron, 2017), (Sommervile, 2011).

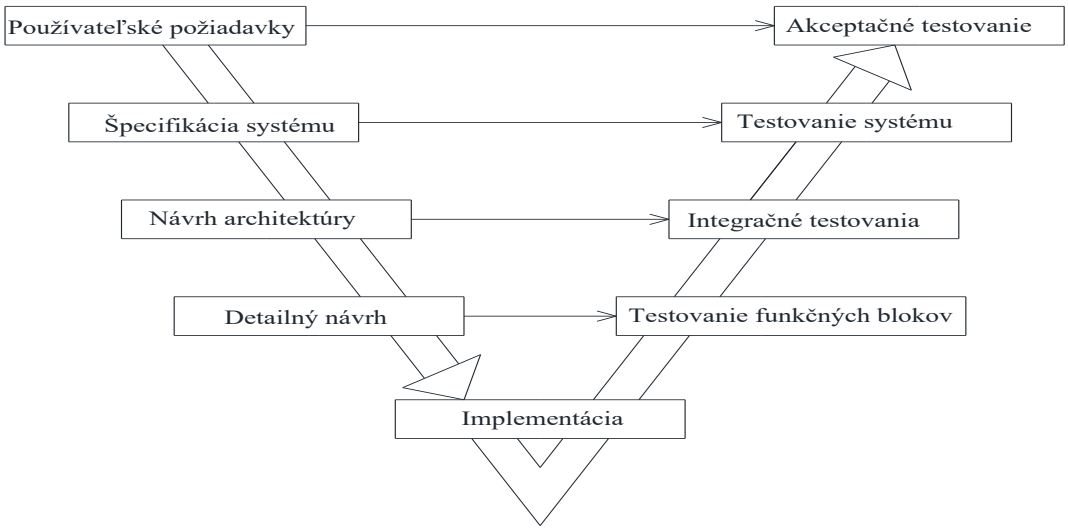
Proces vývoja softvéru je usporiadaný do fáz, ktoré sú zamerané na špecifickú časť softvéru. Tieto fázy sa často uskutočňujú súbežne s vývojom softvéru (Staron, 2017), (Sommervile, 2011).
Tieto fázy obsahujú:

- Inžinierske požiadavky v tejto fáze vytvoríme predstavy o funkciách systému.
- Softvérová analýza , vykonáva sa analýza systému a prijímajú sa rozhodnutia o pridelení funkcionality logickej časti systému
- Softvérová architektúra , softvérový vývojári opisujú návrh systému na vysokej úrovni , vrátane jeho súčastí
- Návrh softvéru v tejto fáze sa podrobne navrhne každá časť systému
- Implementácia, každá časť systému sa implementuje do programovacieho jazyka
- Testovanie, softvér sa testuje rôznymi spôsobmi, napríklad testy jednotlivých časti systému

Prevládajúcim modelom vývoja softvéru v automobiloch je model nazývaný V model, kde sú tieto fázy zarovnané krivkou tvaru V. Fázy návrhu sa nachádzajú na ľavej strane V modelu a testovacie fázy na pravej strane. Vodorovné šípky smerujúce z ľavej vetvy modelu do pravej ukazujú, ktorá vývojová aktivita je podkladom pre ktorú testovaciu. Tento model je predpísaný medzinárodnými priemyselnými normami pre vývoj kritických systémov z hľadiska bezpečnosti, ako je ISO/IEC 26262 (Staron, 2017), (Nagy, 2011), (Beine, a iní, 2004), (Sommervile, 2011).

Proces vývoja softvéru v ľavej vetve začína definovaním používateľských požiadaviek. Nasleduje špecifikácia systému, na základe ktorej sa vykoná návrh architektúry. Ďalej nasleduje detailný návrh systému a ako posledná vývojová aktivita je implementácia. Testovací proces je v tomto modeli rozdelený do štyroch základných úrovní (Nagy, 2011), (Beine, a iní, 2004), (Sommervile, 2011). Partia sem:

- Testovanie funkčných blokov, ide o testovanie elementárnych jednotiek programu, napríklad funkcia, procedúra alebo trieda. Je to testovanie softvéru na najnižšej úrovni.
- Integračné testovanie, testuje schopnosť vzájomnej integrácie už otestovaných elementárnych jednotiek.
- Testovanie systému, ide o testovanie na úrovni celého systému. Na tvorbu testovacích scenárov sa využívajú skutočné scenáre, ktoré od vyvíjaného systému očakávame.
- Akceptačné testovanie, pri tomto testovaní sa pracuje už s vyvinutým systémom a to spôsobom, pre ktorý bol systém vyvinutý. Testovanie prebieha u dodávateľa aj u odberateľa softvéru.



Obrázok 1.2: V model pre návrh softvéru

Zdroj: (Nagy, 2011)

Druhým modelom je X model, tento model sa nezaoberá aktivitami životného cyklu softvéru. Definuje základné testovacie aktivity, ktoré pri testovaní softvérového produktu treba vykonať, ale nestanovuje presnú časovú súvislosť medzi týmito aktivitami a aktivitami životného cyklu softvéru. X model rozdeľuje testovacie aktivity do dvoch základných kategórií. Prvou kategóriou je tetovanie komponentov, pri ktorom sa testuje funkcionalita základných programových jednotiek a ich spolupráca na úrovni komunikačných rozhraní. Druhou kategóriou je testovanie systému, čo predstavuje testovanie integrácie komponentov a ich skupín na základe očakávaného fungovania celého vyvíjaného systému (Nagy, 2011), (Sommervile, 2011).

Tretím modelom W model. W model sa skladá z dvoch vetiev na ľavej strane modelu a dvoch na pravej strane. V prvej vetve na ľavej strane sú znázornené aktivity procesu vývoja softvéru, podobne ako pri V modeli. K tejto vetve je priradená vetva s konštruktívnymi aktivitami testovania. Tieto aktivity sa zaoberajú plánovaním testovania, návrhom a tvorbou testovacích scenárov. V pravej časti W modelu sa nachádzajú taktiež dve vetvy. Prvá zahŕňa činnosti na vykonávanie testov. Druhá vetva zobrazuje aktivity zamerané na ladenie testovaného softvéru a odstraňovanie chýb. Prínosom W modelu je to, že oddeľuje plánovacie a vykonávacie aktivity testovacieho procesu (Nagy, 2011), (Sommervile, 2011).

1.3 VÝVOJ VNORENÉHO SOFTVÉRU NA BÁZE MODELOVANIA

1.3.1 Vývoj založený na modeli (MBD) pre autotronické aplikácie

Vývoj založený na modeli umožňuje rýchly a ekonomicky efektívny vývoj dynamických systémov, systémov pre spracovanie signálu a komunikačných systémov. Používa sa v mnohých aplikáciách na riadenie pohybu, priemysel, letectvo a automobilový priemysel. Vývoj založený na modeli je metodika používaná pri

navrhovaní softvéru pre vnorené systémy. Pri vývoji založenom na modeli sa používa dvojstranný model a kód beží na cieľovom hardvéri. Na jednej strane je model riadenia, ktorý predstavuje softvér vnoreného systému. Architektúra softvéru vnoreného systému je modelovaná blokmi obsahujúcimi algoritmy, funkcie a logické členy. Softvér je automaticky generovaný z tohto modelu. Druhá strana modelu reprezentuje fyzické aspekty systému. Každý blok obsahuje matematické funkcie, ktoré mu umožňujú napodobňovať správanie sa daného fyzického bloku systému. Modelovanie je spôsob ako vytvoriť model systému, ktorý bude simulovať správanie sa reálneho systému. Tento model môžeme simulovať za rôznych podmienok a zistiť ako sa správa. Modelovanie a simulácie sú dôležité pre testovanie podmienok, ktoré je ťažké reprodukovať pomocou hardvérových prototypov. To platí najmä v počiatočnej fáze návrhu, keď ešte nie je k dispozícii hardvér. Modelovanie a simulácie môžu zlepšiť kvalitu návrhu systému, znížením počtu chýb zistených neskôr v procese návrhu (MathWorks, 2019), (Jackson, 2019), (DENNEY, a iní, 2008), (Liang, 2015).

Z modelu môžeme automaticky generovať kód a ak vieme aký hardvér bude použitý, môžeme vytvoriť testovacie prostredie na overenie systému. Generovanie kódu šetrí čas a znižuje počet chýb (MathWorks, 2019), (Jackson, 2019), (DENNEY, a iní, 2008), (Liang, 2015), (Beine, a iní, 2004).

Fázy návrhu:

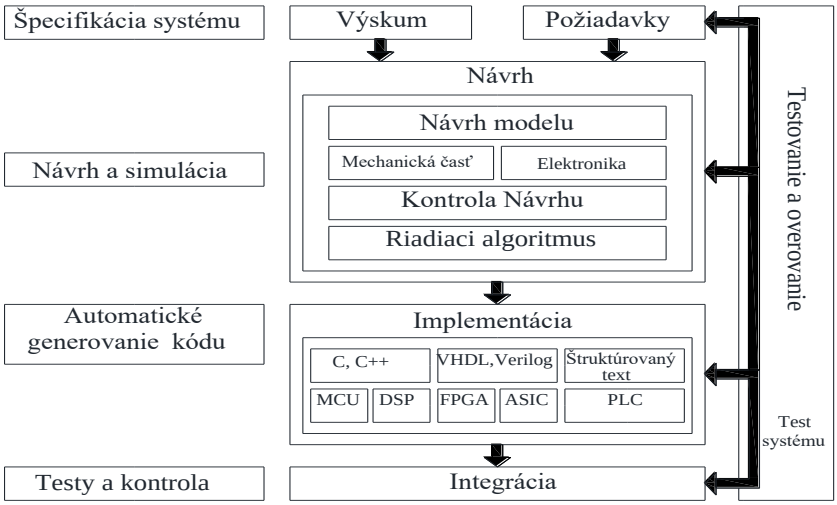
- Model in the Loop (MIL), v tejto fáze je softvér spustený ako model založený na schéme.
- Software in the Loop (SIL), softvér je automaticky generovaný z modelu a beží v simulovanom prostredí. Toto testovanie odhalí chyby softvérovej architektúre.
- Processor in the Loop (PIL), softvér je stále generovaný z modelu, spustený na cieľovom procesore. Odhaľujú sa všetky problémy so zostaveným softvérom a cieľovom procesore.
- Hardware in the Loop (HIL), softvér je podľa modelu spustený na úplnom cieľovom hardvéri, napríklad na riadiacej jednotke.

Vývoj založený na modeli proces návrhu:

- Definovanie požiadaviek
- Návrh softvérovej architektúry, model architektúry
- Návrh softvéru, funkčný model
- Implementácia softvéru
- Automatické generovanie kódu
- Simulácia jednotlivých častí systému SIL alebo PIL
- Integračné testovanie, HIL test
- Overenie funkčnosti softvéru pomocou HIL

Výhody vývoja založeného na modeli:

- Postupné overovanie navrhovaného softvéru.
- Simulácia modelu
- Spoločné návrhové prostredie, ktoré uľahčuje analýzu a overenie systému medzi vývojovými skupinami.
- Môžeme nájsť a opraviť chyby na začiatku návrhu systému, tým znížime čas potrebný na vývoj softvéru a finančné náklady.
- Jednoduchšia úprava modelu, pri použití v podobných aplikáciách.



Obrázok 1.3: Vývoj systému s použitím MBD

Zdroj: (Friedman, 2014)

1.3.2 Automatické generátory kódu

Automatický generátor kódu spočíva v použití vyššieho programovacieho jazyka (napr. MATLAB) spolu s príslušnou knižnicou (napr. MATLAB Coder + Simulink Coder), ktorý dokáže automaticky vygenerovať kód v nižšom programovacom jazyku pre konkrétny mikropočítač alebo iný hardvér. Bez použitia generátora kódu by programátor musel kód napísať manuálne. Hlavnou výhodou automatického generátora kódu je, že sa programátor môže viac sústrediť na algoritmizáciu riešenia problému vo vyššom programovacom jazyku, ďalšou výhodou je jednoduché nastavenie blokov, vizualizácie a spracovanie nameraných dát v analytických a simulačných programoch (KYSLAN, a iní, 2019), (J.S.Freudenberg, 2019), (Liang, 2015).

1.3.3 Matlab a Simulink

MATLAB je vysoko výkonný jazyk pre technické výpočty. MATLAB je inžiniersky nástroj a interaktívne prostredie pre vedecké a technické výpočty, analýzu dát, vizualizáciu a vývoj algoritmov. S programom MATLAB môžeme riešiť úlohy v rôznych odboroch (HUMUSOFT, 2019), (DENNEY, a iní, 2008).

Poskytuje riešenie v oblastiach, ako je napríklad aplikovaná matematika, spracovanie signálov a komunikácia, spracovanie obrazu a počítačové videnie, finančná analýza a modelovanie, návrh riadiacich systémov, robotika a veľa ďalších (HUMUSOFT, 2019), (DENNEY, a iní, 2008).

V súčasnosti môže byť MATLAB použitý na vývoj systémov aktívnej bezpečnosti automobilov, robotické sondy vesmírneho programu, monitorovacie systémy v zdravotníctve, inteligentné energetické siete (HUMUSOFT, 2019).

Použitie:

- Numerické výpočty
- Modelovanie
- Počítačové simulácie
- Vývoj algoritmov

- Meranie a spracovanie signálov
- Návrh riadiacich a komunikačných systémov
- Vývoj aplikácií

Ponuka MATLAB-u:

- Výkonný programovací jazyk pre numerické výpočty a vývoj algoritmov
- Interaktívne prostredie s grafickými nástrojmi pre skúmanie, návrh a riešenie úloh
- Tisíce funkcií z oblasti matematiky, štatistiky, optimalizácie technických výpočtov
- Nástroje pre matematické modelovanie, simuláciu, analýzu a prezentáciu dát, meranie a testovanie
- Grafické funkcie pre vizualizáciu dát a možnosť vytvárania vlastných grafov
- Vývojové nástroje pre zlepšenie kvality kódu a maximalizáciu výpočtového výkonu
- Nástroje pre tvorbu aplikácií s grafickým používateľským rozhraním
- Funkcie pre integráciu algoritmov v jazyku MATLAB s externými aplikáciami a jazykmi ako sú C, Java a Microsoft Excel
- Paralelné a distribuované výpočty, GPU výpočty, práca s rozsiahlymi dátami

Systém MATLAB ponúka viac ako 80 nadstavbových aplikačných knižníc funkcii, vrátane simulačnej platformy Simulink, ktoré rozširujú použitie programu v mnohých aplikačných oblastiach. Kombinácia výpočtového jadra a vývojového prostredia s interaktívnymi nástrojmi vytvorila z MATLAB-u jazyk pre technické výpočty (HUMUSOFT, 2019), (J.S.Freudenberg, 2019).

Dáta môžeme do MATLAB-u importovať zo súborov, iných aplikácií alebo externých zariadení. Keď sú naše dáta v MATLAB-e môžeme ich analyzovať pomocou vstavaných inžinierskych a matematických funkcií, grafov, a vizualizácií (HUMUSOFT, 2019), (J.S.Freudenberg, 2019).

Jazyk MATLAB-u podporuje vektorové a maticové operácie, ktoré sú dôležité pre riešenie technických a vedeckých úloh. Príkazy môžu byť spustené jeden po druhom a

okamžite vracať výsledky. To nám umožní skúmať viac možností a dosiahnuť optimálne riešenie. Pre automatizáciu a opakované využitie našej práce je možné v MATLAB-e vytvárať skripty a funkcie. Môžeme tak tvoriť aj komplexné programy a aplikácie. Vývojové nástroje zjednodušia efektívnu implementáciu našich algoritmov a optimalizáciu ich výpočtového výkonu. MATLAB poskytuje nielen prvky klasického programovacieho jazyka, ale aj nástroje pre návrh vlastných grafických používateľských rozhraní (HUMUSOFT, 2019).

Aplikačné knižnice, rozširujú prostredie MATLAB-u pre riešenie úloh z najrôznejších oblastí:

- matematické výpočty, štatistika a optimalizácia
- návrh a analýza riadiacich systémov
- spracovanie signálov a komunikácia
- spracovanie obrazu a videa
- meranie a testovanie
- finančná analýza a modelovanie
- modelovanie fyzikálnych sústav

Simulink je nadstavba MATLAB-u určená na simuláciu a modelovanie dynamických systémov. Umožňuje používateľovi rýchlo a jednoducho vytvárať modely dynamických sústav vo forme blokových schém (HUMUSOFT, 2019).

Modely môžu byť popísané rovnicami alebo môžu byť zostavené z blokov ktoré reprezentujú prvky reálnych systémov. Okrem modelov fyzikálnych sústav je možné modelovať tiež algoritmy riadiacich systémov, systémy pre spracovanie signálov, komunikácia a spracovanie obrazu (HUMUSOFT, 2019), (DENNEY, a iní, 2008).

Modelovanie systémov:

- Grafický editor pre tvorbu hierarchických blokových schém
- Knižnice preddefinovaných blokov určených k modelovaniu spojitých a diskrétnych systémov
- Simulácia modelov a vizualizácia simulačných výsledkov
- Priebežné testovanie a kontrola modelovaných systémov a algoritmov

Správa modelov

- Nástroje pre správu modelov, dát a projektov
- Analýza modelov
- Simulácia v reálnom čase a prepojenie s hardvérom
- Automatické generovanie kódu (C, C++, HDL) a implementácia na vnorené systémy

1.3.3.1 Matlab Coder

MATLAB Coder generuje kód C a C++ z kódu MATLAB-u pre rôzne hardvérové platformy, od stolových systémov až po zabudovaný hardvér. Vygenerovaný kód je čitateľný a prenosný. Môžeme začleniť existujúci C kód a knižnice, aby sme dosiahli maximálnu efektívnosť pre kľúčové časti nášho algoritmu alebo aby sme znova použili kód ktorému dôverujeme. Generovaný kód môžeme zabaliť aj ako funkciu MEX na použitie v prostredí MATLAB na overenie alebo zrýchlenie (MathWorks, 2019), (J.S.Freudenberg, 2019).

1.3.3.2 Simulink a Embedded Coder

Simulink Coder generuje a vykonáva C a C++ kód z modelov Simulink-u, grafov a funkcií MATLAB-u. Generovaný zdrojový kód sa môže použiť pre aplikácie v reálnom čase a testovanie hardvéru v slučke (HIL). Môžeme vyladiť a sledovať kód pomocou Simulnk-u alebo spustiť a pracovať s kódom mimo MATLAB-u a Simulink-u (MathWorks, 2019), (J.S.Freudenberg, 2019).

Embedded Coder generuje čitateľný, kompaktný a rýchly kód C a C++ pre vnorené systémy používané v hromadnej výrobe. Rozširuje MATLAB Coder a Simulink Coder s pokročilými optimalizáciami pre presnú kontrolu generovaných funkcií, súborov a dát. Tieto optimalizácie zlepšujú efektívnosť kódu a uľahčujú integráciu so starým kódom, dátovými typmi a kalibračnými parametrami (MathWorks, 2019).

Embedded Coder ponúka vstavanú podporu pre softvérové štandardy AUTOSAR a MISRA C. Poskytuje tiež automatizované overenie softvéru na podporu vývoja

softvéru DO-178, IEC 61508 a ISO 26262. Kód Embedded Coder-a je prenosný a dá sa skompilovať a vykonať na ľubovoľnom procesore. Embedded Coder tiež ponúka podporné balíčky s pokročilými optimalizáciami a ovládačmi pre konkrétny hardvér (MathWorks, 2019).

1.3.4 Príklady riešenia generátorov kódu

1.3.4.1 TargetLink (dSPACE)

TargetLink je softvér na automatické generovanie kódu založený na modeloch simulink, ktorý vyrába spoločnosť dSPACE GmbH. TargetLink vyžaduje existujúci model MATLAB/ Simulink. Generuje ANSI-C a strojový kód optimalizovaný pre konkrétne procesory. Podporuje tiež generovanie kódu kompatibilného s AUTOSAR pre softvérové komponenty pre automobilový priemysel. Testovanie vygenerovaného kódu je implementované v Simulink-u. TargetLink podporuje tri simulačné režimy na testovanie vygenerovaného kódu (dspace, 2019) :

- Simulácia modelu v slučke (MIL), tento režim umožňuje kontrolu návrhu modelu, premenné sú väčšinou premenné s pohyblivou desatinnou čiarkou.
- Simulácia softvéru v slučke (SIL), simulácia je založená na vykonaní generovaného kódu, ktorý beží na PC.
- Simulácia procesora v slučke (PIL), vygenerovaný kód beží na cieľovom hardvéri alebo na testovacej doske.

1.3.4.2 Texas Instruments C2000

Rozširujúci balík Embedded Coder pre procesory TI C2000, umožňuje vygenerovať spustiteľný kód a nahrať ho do vývojovej dosky TI C2000. Embedded Coder automaticky generuje kód C a vkladá ovládače I/O zariadení do blokovej schémy. Tieto ovládače sa vkladajú do vygenerovaného kódu C (MathWorks, 2019).

Hlavné funkcie:

- Automatické generovanie a vykonávanie kódu

- Blokové knižnice pre periférie ako sú A/D prevodník, digitálne a analógové I/O piny, ePWM, SPI, I2C, CAN a ďalšie.
- Úprava a zaznamenávanie parametrov v reálnom čase pomocou externého režimu
- Kód optimalizovaný pre procesory vrátane knižníc DMC a IQMath
- Schopnosť simulácie procesora v slučke (PIL)

1.3.4.3 NXP

Matlab Coder, Simulink Coder a Embedded Coder generujú kód ANSI/ISO C/C++, ktorý je možné vygenerovať a spustiť na procesoroch NXP. Rozširujúci balík Embedded Coder pre procesory NXP umožňuje vygenerovať kód z modelov Simulink a implementovať ho do podporovaných procesorov (MathWorks, 2019).

Hlavné funkcie:

- Automatické generovanie a vykonávanie kódu
- Schopnosť simulácie procesora v slučke (PIL)
- Optimalizované bloky riadenia motorov vrátane Park a Clarke transformácií, digitálnych filtrov a všeobecných funkcií
- Blokové knižnice pre CAN, SPI, I2C, UART, PWM, A/D prevodník a ďalšie
- Podpora externého režimu

1.3.5 Kvalita automaticky generovaného kódu, štandard MISRA C

Dôležitosť Programovacieho jazyka C rastie a využíva pre vnorené systémy v automobilovom priemysle. Medzi dôvody jeho použitia patrí (MISRA, 2016):

- V mnohých prípadoch nie sú pre hardvér k dispozícií iné jazyky
- Jazyk C poskytuje dobrú podporu pre vysokorýchlostné, nízkoúrovňové, vstupné a výstupné operácie, ktoré sú nevyhnutné pre mnoho vnorených systémov používaných v automobilovom priemysle.

- Zložitosť aplikácií spôsobuje, že je vhodnejšie použitie jazyka na vyššej úrovni
- Jazyk C dokáže generovať kód, ktorý je menej náročný na pamäť RAM
- Používanie automaticky generovaného kódu C
- Zvyšujúci sa záujem o otvorené prostredia

Žiadny programovací jazyk nemôže zaručiť, že sa vygenerovaný kód bude správať presne tak, ako očakával programátor. V každom programovacom jazyku sa môže vyskytnúť množstvo problémov, napríklad (MISRA, 2016), (MISRA, 2004):

- Programátor spraví chybu
- Programátor nerozumie programovaciemu jazyku
- Kompilátor nerobí to, čo od neho programátor očakáva
- Kompilátor obsahuje chyby
- Chyby keď program beží

MISRA C je súbor odporúčaní pre vývoj softvéru v jazykoch C a C++ vyvinutý spoločnosťou MISRA (Motor Industry Software Reliability Association). Jeho ciele sú zlepšiť zabezpečenie, bezpečnosť, presnosť a spoľahlivosť kódu pre vnorené systémy. Prvá verzia štandardu MISRA C pochádza z roku 1998 a obsahuje redukovanú verziu jazyka C, pomocou ktorej je možné dosiahnuť kód akceptovateľný pre úroveň bezpečnosti SIL. Využíva sa pri vývoji softvéru pre vnorené systémy, kde musia byť dodržané normy. Tieto usmernenia uznávajú, že v niektorých situáciách nie je možné dodržať tieto usmernenia a je potrebné odchýliť sa. Odchýlenie sa nevyhnutne neohrozuje bezpečnosť kódu ale nesie so sebou veľkú zodpovednosť. V najlepšom prípade to oslabí dôveryhodnosť tvrdení o dodržiavaní MISRA, v najhoršom prípade to ohrozí kvalitu kódu, bezpečnosť alebo zabezpečenie (MISRA, 2016), (MISRA, 2004), (Banks, 2016).

Odporúčania MISRA sú určené na použitie v rámci zdokumentovaného procesu vývoja softvéru. Najlepšie výsledky majú keď je zavedený proces ktorý zabezpečí že (MISRA, 2016), (Banks, 2016):

- Softvérové požiadavky vrátane bezpečnostných požiadaviek sú úplné, jednoznačné a správne.
- Vygenerovaný kód sa správa tak ako, je špecifikované v návrhu
- Časti kódu boli testované jednotlivo a spoločne, aby sa identifikovali a odstránili chyby

MISRA C obsahuje 142 pravidiel. Dodržaných musí byť 122 pravidiel a 20 nie je povinných. Všetky pravidlá sa vzťahujú na zdrojový kód a nie na vygenerovaný kód. Pravidlá sú usporiadané podľa rôznych kategórii (MISRA, 2016), (Banks, 2016):

- Prostredie
- Jazykové rozšírenia
- Dokumentácia
- Znakové sady
- Identifikátory
- Dátové typy
- Konštanty
- Deklarácie a definície
- Inicializácia
- Ukazovatele a polia
- Štruktúry

1.4 BEZPEČNOSTNÁ NORMA ISO 26262, ASIL A AUTOSAR

Norma ISO 26262 je funkčná bezpečnostná norma pre automobilový priemysel, ktorá vychádza z normy IEC 61508. Takmer všetci výrobcovia automobilov požadujú, aby ich dodávatelia dodávali hardvérové a softvérové komponenty spĺňajúce požiadavky normy ISO 26262. Norma ISO 26262 je orientovaná na splnenie cieľov funkčnej bezpečnosti. Norma ISO 26262 obsahuje nariadenia a odporúčania počas celého procesu vývoja produktu, od koncepčného vývoja až po vyradenie z prevádzky. Podrobne popisuje ako priradiť prijateľnú úroveň rizika systému alebo

komponentu a zdokumentovať celý proces testovania. Táto norma môže byť použitá len pre čisto hardvérové, softvérové komponenty alebo ich kombináciu. Norma pozostáva z desiatich dokumentov. Osem z nich definuje požiadavky v nasledujúcich oblastiach (NI, 2019), (Els, 2019), (ISO_26262, 2018):

- Manažment
- Fáza konceptu
- Produktový dizajn
- Výroba
- Prevádzka a podpora

Tabuľka 1.1: Alokačná tabuľka ASIL Zdroj: (Embitel, 2018)

		Ovládateľnosť (C)			
Závažnosť (S)	Pravdepodobnosť (E)	C0 Plne ovládateľné	C1 Jednoducho ovládateľné	C2 Ťažšie ovládateľné	C3 Ťažko ovládateľné až neovládateľné
S0 Bez zranení		QM	QM	QM	QM
S1 Ľahké zranenia	E0- Žiadna pravdepodobnosť	QM	QM	QM	QM
	E1- Veľmi nízka pravdepodobnosť	QM	QM	QM	QM
	E2- Nízka pravdepodobnosť	QM	QM	QM	QM
	E3- Stredná pravdepodobnosť	QM	QM	QM	ASIL A
	E4- Vysoká pravdepodobnosť	QM	QM	ASIL A	ASIL B
S2 Ťažké zranenia	E0- Žiadna pravdepodobnosť	QM	QM	QM	QM
	E1- Veľmi nízka pravdepodobnosť	QM	QM	QM	QM
	E2- Nízka pravdepodobnosť	QM	QM	QM	ASIL A
	E3- Stredná pravdepodobnosť	QM	QM	ASIL A	ASIL B
	E4- Vysoká pravdepodobnosť	QM	ASIL A	ASIL B	ASIL C
S3 Život ohozujúce alebo smrteľné zranenia	E0- Žiadna pravdepodobnosť	QM	QM	QM	QM
	E1- Veľmi nízka pravdepodobnosť	QM	QM	QM	ASIL A
	E2- Nízka pravdepodobnosť	QM	QM	ASIL A	ASIL B
	E3- Stredná pravdepodobnosť	QM	ASIL A	ASIL B	ASIL C
	E4- Vysoká pravdepodobnosť	QM	ASIL B	ASIL C	ASIL D

ASIL (Automotive Safety Integrity Level) je kľúčovým komponentom pre súlad s normou ISO 26262. ASIL sa určuje na začiatku procesu vývoja. Funkcie systému sa analyzujú s ohľadom na možné riziká. ASIL nerieši technológie používané v systéme, zameriava sa výlučne na poškodenie vodiča a ostatných účastníkov cestnej premávky. Každej bezpečnostnej požiadavke je priradený ASIL QM, A, B, C alebo D. ASIL QM predstavuje nebezpečenstvá ktoré nevyžadujú žiadne bezpečnostné prvky. ASIL A má najmenší vplyv na ľudské zdravie, napríklad autorádio. ASIL D vyžaduje rozsiahle opatrenia na zníženie rizika, tak aby v zlyhanie systému nespôsobilo vážne ohrozenie

vodiča, a ostatných účastníkov cestnej premávky (NI, 2019). Úrovne ASIL sú priradené podľa alokačnej tabuľky definovanej v norme ISO 26262 (Tabuľka 1.1).

AUTOSAR je štandardizačná iniciatíva výrobcov automobilov a ich dodávateľov s cieľom vytvoriť spoločnú architektúru softvéru pre riadiace jednotky v automobiloch (Kindel, a iní, 2009).

Ciele AUTOSAR vychádzajú z potreby zjednodušiť vývoj softvéru a tým znížiť náklady, na riadiacu elektroniku v automobiloch. Architektúry štandardu AUTOSAR, stoja na nasledujúcich princípoch (Kindel, a iní, 2009):

- Modularita, vhodné rozčlenenie funkcií softvéru do modulov
- Škálovateľnosť, musí byť možné jednoducho meniť rozsah funkcie a počet paralelných modulov v systéme.
- Prenositeľnosť, moduly musia byť jednoducho prenositeľné medzi hardvérovými platformami.
- Opakované použitie, funkcionalita modulu musí byť ohraničená tak, aby modul bolo možné znovu použiť, ideálne bez zmien a aby zároveň nebránil zdokonaľovaniu aplikácie, ktorú konečný užívateľ vidí v podobe ponúkaných funkcií a služieb zariadenia.
- Štandardné rozhrania, moduly by sa mali dať ľahko používať aj medzi rôznymi dodávateľmi, je potrebné, aby boli rozhrania jasne, efektívne a univerzálne definované.

Štandardizácia sa týka softvéru pre nasledujúce skupiny riadiacich jednotiek (Kindel, a iní, 2009):

- Šasi elektronika, elektronické otváranie dverí, stierače.
- Motor a riadenie prenosu výkonu vozidla, riadenie vstrekovania, riadenie ventilov, elektronické riadenie automatickej prevodovky.
- Bezpečnostné prvky, ABS, airbag.
- Multimédia, rádio, telefón.
- Elektronické prvky, tvoriace komunikačné rozhranie s vodičom, displeje, mikrofóny, reproduktory.

1.5 FUNKČNÁ BEZPEČNOSŤ A PROCESORY TI C2000

Mikrokontroléry (MCU) TI C2000 sú vysokovýkonné mikrokontroléry, ktoré sú určené na riadenie výkonovej elektroniky a poskytujú pokročilé spracovanie digitálneho signálu v priemyselných a automobilových aplikáciách. Za viac ako 20 rokov sa v oblasti analógového a digitálneho riadenia jednotky MCU C2000 vyvinuli s cieľom poskytovať presné snímanie, výkonné spracovanie aby inžinieri mohli vytvárať čo najúčinnejšie systémy riadenia (TI, 2013).

Všetky MCU C2000 používajú 32 bitový procesor C28x s pohyblivou rádovou čiarkou (FPU) alebo bez nej. Vyššie rady MCU C2000 podporujú zložitejšie matematické operácie pomocou VCU jednotky. Bolo dokázané, že sady funkcií MCU Piccolo (TMS320F280 6x /5x /3x/ 2x) a Delfino (TMS320F280 23x/ 33x) poskytujú vysoký výkon v mnohých aplikačných oblastiach a stále sa zlepšujú (TI, 2013).

Architektúra mikrokontrolérov C2000 sa delí na štyri typy na základe výkonu procesora a subsystémov (TI, 2019), (TI, 2013):

- Typ A, C28x CPU + FPU + VCU + CLA s riadiacim a analógovým subsystémom, Piccolo 6x, ASIL B
- Typ B, C28x CPU + CLA s riadiacim a analógovým subsystémom, Piccolo 3x/5x, ASIL B
- Typ C, C28x CPU + FPU s riadiacim a analógovým subsystémom, Delfino F2833x
- Typ D, C28x CPU s riadiacim a analógovým subsystémom, Piccolo2x a Delfino F2823x

Typ A , C28x CPU + FPU + VCU + CLA s riadiacim a analógovým subsystémom, 32 bitový mikroprocesor s pohyblivou rádovou čiarkou, jednotkou pre urýchlenie matematických operácií (CLA) a špeciálne I/O optimalizované na vykonávanie riadiacich algoritmov v reálnom čase, dostupné na MCU Piccolo6x (F2806x). Tento systém obsahuje niekoľko riadiacich periférií (PWM, enkodér, rozhrania pre sériovú komunikáciu). CLA obsahuje CPU s pohyblivou rádovou čiarkou a optimalizovanými I/O pre analógový subsystém. CLA môže tiež obsahovať bezpečnostné funkcie

a slúžiť ako dozorný mikroprocesor. MCU Piccolo6x (2806x) majú ďalšiu jednotku pre matematické výpočty VCU, ktorá slúži na vykonávanie pokročilých funkcií spracovania signálu pre aplikácie na riadenie a komunikáciu po elektrickej sieti (TI, 2013), (TI, 2018).

Typ B, C28x CPU + CLA s riadiacim a analógovým subsystémom. 32 bitový mikroprocesor s pevnou rádovou čiarkou, s CLA a I/O optimalizované na vykonávanie riadiacich algoritmov v reálnom čase dostupné na MCU Piccolo3x/5x (F2803x/ F2805x). Tento systém obsahuje niekoľko riadiacich periférií (PWM, enkodér, rozhrania pre sériovú komunikáciu) pre pokročilé riadenie a sofistikované aplikácie na spracovanie signálu. CLA obsahuje CPU s pohyblivou rádovou čiarkou a optimalizovanými I/O pre analógový subsystém. CLA môže tiež obsahovať bezpečnostné funkcie a slúžiť ako dozorný mikroprocesor (TI, 2013), (TI, 2018).

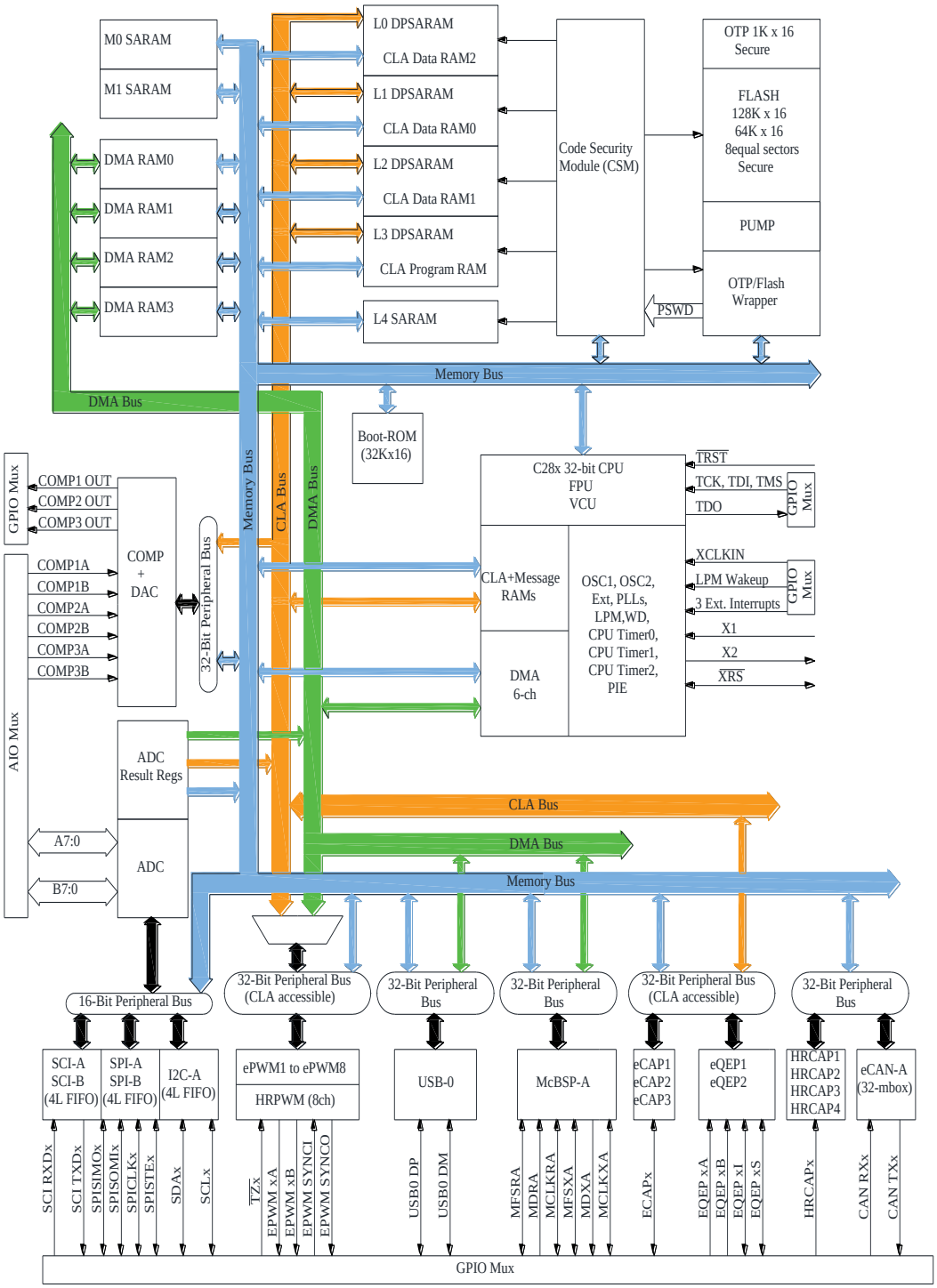
Typ C, C28x CPU + FPU s riadiacim a analógovým subsystémom. 32 bitový mikroprocesor s pohyblivou rádovou čiarkou a I/O optimalizované na vykonávanie riadiacich algoritmov v reálnom čase, dostupné na MCU Delfino (F2833x). Tento systém obsahuje niekoľko riadiacich periférií (PWM, enkodér, rozhrania pre sériovú komunikáciu) pre pokročilé riadenie a sofistikované aplikácie na spracovanie signálu (TI, 2013), (TI, 2018).

Typ D, C28x CPU s riadiacim a analógovým subsystémom. 32 bitový mikroprocesor s pevnou rádovou čiarkou, s CLA a I/O optimalizované na vykonávanie riadiacich algoritmov v reálnom čase dostupné na MCU Piccolo 2x (F2802x) a Delfino (F2823x). Tento systém obsahuje niekoľko riadiacich periférií (PWM, enkodér, rozhrania pre sériovú komunikáciu) pre pokročilé riadenie a sofistikované aplikácie na spracovanie signálu (TI, 2013), (TI, 2018).

Analógový subsystém, MCU Piccolo obsahujú sadu integrovaných analógových komponentov, napríklad vysoko výkonný 12 bitový A/D prevodník, analógové komparátory, oscilátory a regulátory napätia. MCU Delfino obsahujú menej integrovaných analógových komponentov, majú 12 bitový A/D prevodník. V nasledujúcej tabuľke (Tabuľka 1.2) sú uvedené vlastnosti mikrokontrolérov C2000 (TI, 2013), (TI, 2018).

Tabuľka 1.2: Vlastnosti mikrokontrolérov C2000 Zdroj: (TI, 2013)

	C2000 32 Bitové mikropočítače					
	Mikropočítače Picollo				Mikropočítače Delfino	
	TMS320 F2806x	TMS320 F2805x	TMS320 F2803x	TMS320 F2802x	TMS320 F2833x	TMS320 F2823x
32 Bitový procesor	90 MHz	80 MHz	60 MHz	60 MHz	150 MHz	150 MHz
C28x CPU	X	X	X	X	X	X
FPU	X				X	
Matematické funkcie	X	X	X	X	X	X
VCU	X					
CLA	X	X	X			
DMA	X				X	X
Pamäť						
Bootovanie z ROM	X	X	X	X	X	X
RAM	X	X	X	X	X	X
FLASH	X	X	X	X	X	X
ROM	X	X	X	X		
OTP	X	X	X	X	X	X
PIE	X	X	X	X	X	X
Riadiace periférie						
32 Bitové CPU časovače	X	X	X	X	X	X
PWM	X	X	X	X	X	X
HRPWM	X		X	X	X	X
HRCAP	X		X			
QEP	X	X	X	X	X	X
Komunikačné periférie						
USB	X					
SPI	X	X	X		X	X
SCI	X	X	X	X	X	X
LIN	X		X			
CAN	X	X	X		X	X
I2C	X	X	X	X	X	X
McBSP	X				X	X
Externá pamäťová zbernica	X				X	X
Analógové periférie						
Oscilátory s 2 pinmi	X		X	X	X	X
Oscilátory s 0 pinmi	X	X	X	X		
POR a BOR	X	X	X	X		
Regulátor napätia	X	X	X	X		
Snímač teploty	X	X	X	X		
ADC	X	X	X	X	X	X
OPAMP		X				
DAC	X	X	X	X		
Komparátor	X	X	X	X		
Puzdro	80 pinov 100 pinov	64 pinov 80 pinov	56 pinov 64 pinov 80 pinov	38 pinov 48 pinov	176 pinov 179 pinov	176 pinov 179 pinov
Aplikačné oblasti	Riadenie motorov	Riadenie motorov	Riadenie motorov	Riadenie motorov	Riadenie motorov	Riadenie motorov
	Premena energie		Premena energie	Premena energie	Premena energie	Premena energie
	Komunikácia po elektrickej sieti				Automobilový radar	Automobilový radar



Obrázok 1.4: Bloková schéma MCU C2000

Zdroj: (TI, 2018)

MCU C2000 sú zamerané primárne pre priemyselné aplikácie a môžu sa zaoberať bezpečnostnými aplikáciami. Je očakávané, že tie isté mikrokontroléry, ktoré sa používajú na všeobecné funkcie zariadení, sa budú používať aj na bezpečnostné funkcie zariadenia v priemysle alebo pri premene energie. Napríklad pri pohonoch je reakčný čas pri poruche je niekoľko desiatok milisekúnd. Celková odozva systému založeného na MCU C2000 môže dosiahnuť časový interval aj menej ako 10ms. MCU C2000 sú navrhnuté tak, aby spĺňali väčšinu bezpečnostných prvkov uvedených v normách IEC 60730 alebo v rovnocenných normách. Aj keď tieto zariadenia majú niekoľko hardvérových bezpečnostných funkcií, softvér na aplikačnej úrovni je rozhodujúcou súčasťou, ktorá zlepšuje hardvérové bezpečnostné funkcie a spĺňa bezpečnostné požiadavky. Správne navrhnutý systém využívajúci MCU C2000 by mal byť schopný splniť bezpečnostné požiadavky, ak dizajnéri používajú zariadenie v rámci stanovených špecifikácií a implementujú požadovaný diagnostický hardvér a softvér (TI, 2019), (TI, 2013), (TI, 2018).

Pre vývoj systémov z hľadiska bezpečnosti je potrebné zvládnuť systematické aj náhodné poruchy. Spoločnosť Texas Instruments (TI) vytvorila jedinečný vývojový proces pre systémy z hľadiska bezpečnosti, ktoré výrazne znižujú pravdepodobnosť systematického zlyhania. Vývojové tými TI využívajú tento proces vo všetkých svojich radoch MCU, ktoré sa zaoberajú aplikáciami v priemysle a automobiloch. Priemyselné a automobilové trhy majú vysoké požiadavky na kvalitu a spoľahlivosť produktov. Tento vývojový proces je možné považovať za kvalitný ale nespĺňa všetky požiadavky normy IEC 61058 a ASIL. Proces vývoja štandardu TI je certifikovaný v súlade s normou ISO TS 16949 podľa hodnotenia spoločnosti Det Norske Veritas Certification, Inc. (Katy, Texas). Vývoj je tiež certifikovaný v súlade s normou ISO 9001:2008 podľa hodnotenia DNV Certification B.V. Tieto normy sa vzťahujú na MCU TI (TI, 2019), (TI, 2013).

Proces vývoja MCU Piccolo a Delfino prebieha podľa vývojových metód TI a je v súlade s normou ISO TS 16949. Cieľom procesu vývoja je vziať najlepšie aspekty z každého toku vývoja a spolupracovať, aby sa dosiahli najlepšie schopnosti v triede na zníženie výskytu systematických porúch. Proces vývoja je tiež zameraný, aby bol

v súlade s normou IEC 61508 a neustále sa zlepšuje s cieľom začleniť nové funkcie. Série MCU Piccolo a Delfino sú vyvinuté tak, aby ponúkali zariadenia vo viacerých triedach pre spotrebiteľské, priemyselné a automobilové aplikácie (TI, 2013).

Kľúčovými prvkami procesu vývoja mikropočítačov TI sú:

- Predpoklady týkajúce sa projektovania na úrovni systému, koncepcie bezpečnosti a požiadaviek založených na odborných znalostiach spoločnosti TI v oblasti vývoja systémov z hľadiska bezpečnosti.
- Kombinované kvalitatívne a kvantitatívne alebo podobné techniky technickej analýzy, ktoré uvažujú režimy zlyhania a diagnostické techniky spoločnosti TI.
- Odhad poruchy založený na viacerých priemyselných štandardoch, ako aj na výrobných údajoch firmy TI.
- Proces vývoja MCU C2000 prijíma nové procesy vývoja produktov QRASAP00160

Architektúra MCU C2000 obsahuje bezpečnostné mechanizmy, ktoré pri správnom použití môžu pomôcť odhaliť a reagovať na náhodné poruchy. MCU Piccolo a Delfino sa zameriavajú na funkčnú bezpečnosť s obmedzenou asistenciou hardvéru a s knižnicami softvérového dozoru na detekciu zlyhania systému a zariadenia a na urýchlenie nápravných opatrení. MCU Piccolo ponúka dva odlišné varianty implementácie bezpečnosti na základe svojich hardvérových možností. Väčšina MCU Piccolo2x využíva jednoduché konfigurácie procesora C28x a preto sú bezpečnostnými prvkami hlavne softvérové funkcie spolu s hlavným aplikačným firmvérom. MCU Piccolo3x a 6x majú dve možnosti na zvýšenie bezpečnosti pomocou procesora C28x a CLA (TI, 2019), (TI, 2013).

Prevádzkové stavy MCU C2000:

- Zariadenie je vypnuté, ide o počiatočný prevádzkový stav MCU C2000. Zariadenie nie je pripojené na napájanie a nie je funkčné. Tento stav môže prejsť len do bezpečného stavu a dá sa dosiahnuť iba z bezpečného stavu.

Napájanie jadra býva (VDD= (1,2 až 1,8)V), napájanie I/O (VDDIO=3,3V) (TI, 2013), (TI, 2018).

- Bezpečný stav, v bezpečnom stave je napájaný procesor C28x a jeho subsystémy, ale nie sú funkčné. Reset pri zapnutí je presadzovaný systémom, ale nie je uvoľnený pokiaľ napájacie zdroje neprejdú do stabilného stavu. Zariadenie obsahuje monitor vnútorného napätia (POR a BOR), ktorý zisťuje či sú napájacie zdroje v minimálnom prevádzkovom rozsahu. Keď je zariadenie v bezpečnom stave, procesor a periférie nie sú funkčné (TI, 2013), (TI, 2018).
- Zapínanie za studena, v tomto stave sú hlavné analógové prvky, logika digitálneho riadenia a logika ladenia inicializované pre budúce použitie. Procesor a periférie zostávajú napájané, ale nie sú funkčné.
- Zapínanie za tepla, resetuje sa digitálna logika a spúšťa sa procesor C28x. Procesor C28x začne spúšťať softvér z ROM, inicializuje sa systém a čaká na spúšťací režim. Neexistuje žiadne hardvérové blokovanie, ktoré by potvrdilo že zapínanie za tepla je ukončené. Toto je softvérové rozhodnutie v spúšťacom kóde procesora (TI, 2018), (TI, 2013).
- Prevádzková a spúšťacia fáza, v tomto stave vstupuje procesor C28x do spúšťacej fázy, kontroluje spúšťacie režimy a vstupuje do spúšťacej funkcie. Procesor C28x vykonáva inicializáciu softvéru z flash pamäte a spúšťa aplikačný kód určený pre koncovú aplikáciu (TI, 2013), (TI, 2018).

Štandardné bezpečnostné diagnostické funkcie MCU C2000 a jeho subsystémov. MCU C2000 majú niekoľko funkčných bezpečnostných prvkov, ktoré môžu v prípade poruchy vyvolať prerušenie. Nie všetky hardvérové moduly dokážu vyvolať prerušenie v nečinnom stave. V nečinnom stave dokáže vyvolať prerušenia len niekoľko špeciálnych hardvérových modulov. Knižnice bezpečnostného softvéru podľa normy IEC 60730 pomáhajú monitorovať všetky regióny mikropočítača, procesor, pamäť a periférne zariadenia a generovať chybové hlásenia, ak vykazujú akékoľvek odchýlky. Poruchy CPU, registrov alebo subsystémov väčšinou zostávajú

nečinné alebo neaktívne, pokiaľ ich softvérový alebo hardvérový nástroj pravidelne nesleduje (TI, 2019).

Opis bezpečnostných modulov MCU C2000:

- Napájanie, MCU Piccolo potrebujú externý regulátor, ktorý dodáva požadované napätie a prúd. Zariadenia Piccolo majú nezávislé napájanie pre analógovú a digitálnu logiku. Zariadenie pracuje s externým zdrojom 3,3 V, ak nie sú aktivované interné regulátory. Väčšina systémov používa interné regulátory, ktoré znižujú napätie na 1,8 V. MCU Piccolo tiež podporujú interné POR a BOR ktoré monitorujú správnu činnosť interných regulátorov. Pomáhajú monitorovať napätie a vyvolať prerušenie alebo reset. MCU Delfino potrebujú externé napájanie, pretože nemajú regulátory na čipe (TI, 2013), (TI, 2018).
- Externé monitory napätia. Pre MCU Piccolo sa odporúča použitie externých monitorov napätia na monitorovanie ochrany proti prepätiu na vstupoch. Prepätia sú bežnejšie pre priemyselné aplikácie a preto topológiu MCU dopĺňa externý monitor napätia. Väčšina obvodov na detekciu prepätia môže rýchlo varovať subsystémy MCU Piccolo, aby nedošlo k poškodeniu riadiacich systémov. Toto platí aj pre MCU Delfino (TI, 2013), (TI, 2018).
- Reset, MCU Piccolo a Delfino vyžadujú externý reset pri zapínaní za tepla a za studena, aby sa všetka asynchrónna a synchrónna logika dostala do známeho stavu. Reset pri zapnutí generuje interný signál na resetovanie väčšiny blokov subsystému predtým, ako začne spúšťací proces (TI, 2013), (TI, 2018).
- Hodiny, MCU Piccolo sú primárne synchrónne logické zariadenia a pre správnu funkciu potrebujú hodinový signál. Logika hodín je prepracovaná tak, aby poskytovala synchrónny hodinový signál vo všetkých subsystémoch a poskytovala bezpečná hodiny aj keď dôjde k výpadku hlavných hodín. Zariadenia Piccolo majú dva interné oscilátory, OSC1 a OSC2. MCU Delfino nepodporujú oscilátory na čipe, podporujú iba kryštálový oscilátor a externé hodiny (TI, 2013), (TI, 2018).

- CPU Prerušenia pre nedovolené inštrukcie. Procesory C28x majú neoddeliteľnú funkciu na detekciu nepovolených inštrukcií a na vyvolanie prerušenia kvôli bezpečnému zotaveniu (TI, 2013), (TI, 2018).
- Delenie nulou. Inštrukcie procesorov C28x neobsahujú hardvérovú funkciu na detekciu delenia nulou. Ak programátor zostavuje funkciu delenia pomocou C28x inštrukcií, mal by do kódu pridať funkciu na kontrolu delenia nulou. Jednotka FPU v subsystémoch s procesorom C28x je schopná detegovať delenie nulou. Algoritmy by mali obsahovať kontrolu delenia nulou pomocou príznaku, ktorý vyvolá jednotka FPU pri delení nulou (TI, 2013), (TI, 2018).
- Kontrola prebudenia v režime nízkej spotreby. MCU Piccolo a Delfino podporujú niekoľko režimov nízkej spotreby, aby sa šetrila energia. Systémový dizajnér by mal navrhnúť vhodnú metódu prebudenia so vstupnými spúšťacími signálmi a tiež pridať záložný režim obnovy v prípade, ak vstupy prebudenia zariadenia nefungujú (TI, 2013), (TI, 2018).

Odporúčania pre návrh softvéru MCU Texas Instruments (TI, 2013):

1. Názvy premenných
    - Všetky premenné začínajú s malým písmenom. Globálne premenné musia začínať s písmenom ‘g‘. Napríklad: uint16_t loopCounter; uint16_t gDataBuffer
    - Symbolické konštanty musia byť napísané veľkými písmenami a musia začínať názvom modulu.

      Napríklad: #define MODULE_REGISTER_BIT (1<<4)
2. Názvy Funkcií
    - Všetky funkcie musia začínať názvom modulu a musia byť napísané veľkými písmenami, za názvom modulu musí byť nasledujúci znak ‘_‘. Napríklad: PWM_setPeriod();

3. Komentáre
    - Všetky komentáre musia používať oddeľovač '//' a musia byť umiestnené v rovnakom riadku ako je komentovaná časť kódu.
4. Prototypy funkcií
    - Všetky funkcie musia byť definované v zodpovedajúcom hlavičkovom súbore.
5. Odsadenie
    - Každá úroveň odsadenia sú štyri medzery.
    - Riadky ktoré pokračujú musia byť odsadené ôsmimi medzerami.
6. Výrazy v zátvorkách
    - Výrazy musia byť napísané jedným z nasledujúcich príkladov:

```
if (expr)        do          for(a;b;c)      else if (a) {
{                {           {               stmt;
stmt;            stmt;       stmt;           }
}                }           }               else{
                                             stmt;
switch (expr)                                }
{
case n;
 break;
default;
 break;
}
```

7. Štruktúra súborov
    - Celý softvér musí byť umiestnený v jednom súbore. Časti súboru musia byť v nasledujúcom poradí: hlavičkové súbory, definície, globálne premenné, funkčné prototypy.

8. Dátové typy
    - _Bool pre booleovské typy
    - (u)int_leastX_t (portable)
    - (u)int_leastX_t (architecture), kde X je 8, 16,32 a 64
9. Definície makier
    - Definícia makra musí byť jednoriadková.
10. Statická analýza kódu
    - Statická analýza podľa pravidiel MISRA a LDRA.
    - Nástroje LDRA sa kontrolu kódu.
11. Proces archivácie
    - Archív je cez centralizovaný systém úložísk.
    - Chyby a problémy sú hlásené prostredníctvom webového sledovacieho systému Bugzilla.

Tabuľka 1.3: MISRA C výnimky pre MCU s C28x architektúrou

Zdroj: (TI, 2013)

1.	Povinné pravidlá ktoré sú porušené			
		a.	Jednoriadkové komentáre	MISRA-C:2004 2.2
		b.	Ukazovateľe použité mimo poľa	MISRA-C:2004 17.1, 17.4
		c.	# pragma, použitie pragmy	MISRA-C:2004 3.4
2.	Odporúčané pravidlá ktoré sú porušené			
		a.	Použitý základný typ deklarácie	MISRA-C:2004 6.3
		b.	Výraz nie je typu Boolean	MISRA-C:11.1, 11.4
		c.	Ukazovateľ na funkciu	MISRA-C:2004 12.6, 13.2
		d.	Ukazovateľ na celé čislo	MISRA-C:2004 11.3

1.6 ŠTUDIJNÝ PROGRAM AUTOTRONIKA

Bakalársky študijný program autotronika je orientovaný predovšetkým na zvládnutie základných a všeobecných znalostí potrebných v širokom spektre elektrotechnických odborností najmä z oblasti automobilovej elektroniky, hybridných vozidiel a elektromobilov, potrebných na štúdium študijných programov druhého stupňa uskutočňovaného priamo v tomto, ale aj v príbuzných študijných odboroch. Pokiaľ absolvent nepokračuje v štúdiu na 2. stupni vysokoškolského štúdia, nadobudne požadovaný široký odborný profil a je schopný sa adaptovať v rôznych

technických, ako aj iných prevádzkach. Absolventi štúdia autotroniky by mali byť odborní pracovníci schopní identifikovať akékoľvek elektronické závady vo vozidlách. Ich uplatnenie sa predpokladá najmä: v servisoch a opravárenských dielňach, v predajniach moderných automobilov a vo vzdelávacích inštitúciách. (PortalVS, 2019)

Študijný program autotronika bol navrhnutý v úzkej spolupráci so subjektami pôsobiacimi v automobilovom priemysle. Výučba v laboratóriu je zameraná na výučbu predmetov zameraných na elektrotechniku cestných vozidiel, diagnostiku elektrických systémov automobilov a senzory a akčné členy motorových vozidiel. V laboratóriu sa nachádza automobil KIA, kde sa vykonávajú online merania. V rámci meracej techniky je laboratórium vybavené zdrojmi striedavého a jednosmerného prúdu, číslicovými osciloskopmi spolu s prúdovými a napäťovými sondami, číslicovými multimetrami a základnou automobilovou diagnostickou technikou, ako je systém tester KTS 52 s modulom FSA a motortestery FSA 720 a 450 od firmy BOSCH. Primárne je laboratórium určené na výučbu predmetov autotronika a vnorené procesorové systémy pre automobilové aplikácie. Pracovisko je vybavené riadiacimi jednotkami rôznych výrobcov ako aj najmodernejšími vývojovými doskami TRK-MPC 5xxx od spoločnosti Freescale na programovanie procesorov určených na riadenie výkonu motorových vozidiel. Za účelom praktických ukážok a overenia riadiacich algoritmov sa v laboratóriu nachádzajú moderné CAN analyzátory KVASER pre skúmanie vlastností elektrického, dieselového a benzínového pohonu.

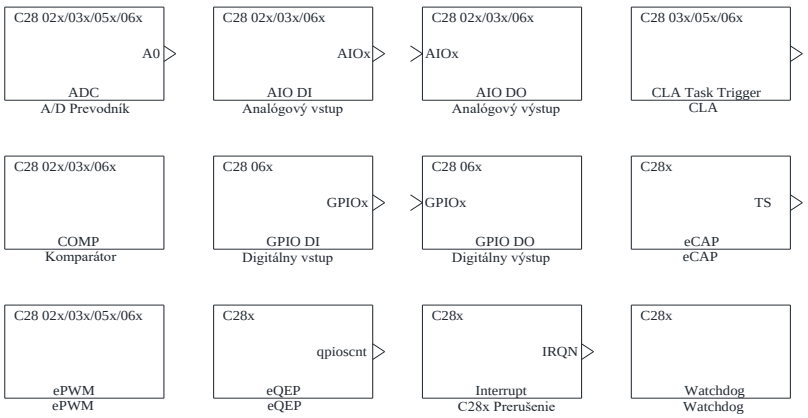
Softvérové zručnosti: Jazyk C, C++, MATLAB, Simulink, CodeWarrior, CodeComposer, Asembler, AVR Studio, Vissim, PLECS . (KME, 2019)

2 ANALÝZA MOŽNOSTÍ GENERÁTOROV KÓDU NA BÁZE PROGRAMOVACIEHO PROSTREDIA MATLAB

2.1 GENERÁTORY KÓDU NA BÁZE PROGRAMOVACIEHO PROSTREDIA MATLAB PRE MIKROKONTROLÉRY TI C2000

MATLAB Coder generuje ANSI C alebo C++ kód z prostredia MATLAB a je nevyhnutný pre Simulink Coder a Embedded Coder. Simulink Coder taktiež generuje kód C alebo C++ ale z prostredia Simulink. Embedded Coder umožňuje optimalizovať kód vygenerovaný z MATLAB Coder-a a Simulink Coder-a pre cieľovú platformu.

Pre mikrokontroléry TI C2000 môžeme vygenerovať kód pomocou aplikácie MATLAB Coder alebo z prostredia simulink. Embedded Coder pre prostredie Simulink obsahuje blokové knižnice pre jednotlivé periférie (Obrázok 2.1) MCU TI C2000 a bloky pre matematické funkcie (Obrázok 2.2).



Obrázok 2.1: Bloky pre nastavenie periférií v prostredí simulink

Zdroj: (MathWorks, 2019)

Na obrázku (obrázok 2.1) sú uvedené vybrané bloky z knižnice Embedded Coder pre mikrokontroléry TI C2000 v prostredí Simulink.

V bloku A/D prevodníka môžeme nastaviť nasledujúce parametre:

- Vstupný kanál na ktorom budeme merať
- Režim vzorkovania
- Prioritu začiatku prevodu
- Šírku vzorkovacieho okna
- Zdroj ktorý spustí prevod
- Spustenie prevodu od prerušenia A/D prevodníka
- Čas vzorkovania
- Dátový typ
- Vyvolanie prerušenia po skončení prevodu

V blokoch analógového vstupu a výstupu nastavíme ktorý pin má byť analógový vstup alebo výstup. Bloky digitálneho vstupu a výstupu slúžia na nastavenie pinu ktorý bude digitálny vstup alebo výstup. Pri výstupných pinoch môžeme nastaviť aby sa preklápala hodnota na pine a pomocou vzorkovacieho času nastavíme v akom časovom intervale sa má táto hodnota preklopiť (MathWorks, 2019).

CLA je koprocesor, ktorý umožňuje paralelné spracovanie dát. V bloku sa nastavuje číslo úlohy, zdroj ktorý spúšťa CLA funkciu a frekvenciu s akou je funkcia volaná (MathWorks, 2019).

Komparátor porovnáva dve vstupné napätia. Zvolíme si ktorý modul komparátora chceme použiť, každý modul má dva vstupné piny a jeden výstupný. Môžeme si zvoliť druhý vstup komparátora, synchronizáciu a čas vzorkovania (MathWorks, 2019).

Blok eCAP zachytáva načasovanie externých udalostí. V bloku watchdog nastavíme zdroj ktorý zresetuje modul watchdog na procesore. Blok eQEP sa používa spolu s enkodérom na získanie informácií o polohe smere a rýchlosti točivého stroja. Pri hardvérovom prerušení nastavíme číslo prerušenia CPU a PIE. Čísla prerušenia do

CPU a PIE určujú od ktorej periférie chceme vyvolať prerušenie. V tabuľke 2.1 sú uvedené čísla prerušení CPU a PIE pre MCU TI C2000 (MathWorks, 2019).

Tabuľka 2.1: Čísla prerušení CPU a PIE *Zdroj: (MathWorks, 2019)*

PIE→	1	2	3	....	7	8
CPU↓						
1	ADCINT1	ADCINT2	Reserved	....	TINT0	WAKEINT
2	EPWM1_TZINT	EPWM2_TZINT	EPWM3_TZINT	....	EPWM7_TZINT	EPWM8_TZINT
3	EPWM1_INT	EPWM2_INT	EPWM3_INT	....	EPWM7_INT	EPWM8_INT
4	ECAP1_INT	ECAP2_INT	ECAP3_INT	....	HRCAP1_INT	HRCAP2_INT
5	EQEP1_INT	EQEP2_INT	EQEP3_INT	....	EPWM10_INT	EPWM9_INT
⋮	⋮	⋮	⋮		⋮	⋮
12	XINT3	XINT4	XINT5	....	LVF	LUF

V bloku PWM môžeme nastaviť:

- Modul PWM, ktorý chceme použiť
- V akých jednotkách chceme zadať periódu
- Periódu
- Režim počítania
- Preddeličku
- Fázový posun
- Porovnávacie hodnoty CPMA a CMPB
- Výber akcie keď dosiahne porovnávaciu hodnotu
- Nastavenie bezpečnostnej doby
- Spustenie prevodu AD prevodníka pri dosiahnutí požadovanej hodnoty
- Prerušenie od PWM
- PWM s vysokým rozlíšením

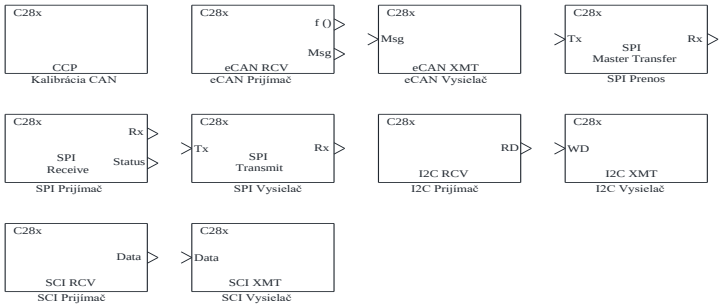
Na obrázku (Obrázok 2.2) je kód vygenerovaný z prostredia Simulink pre hardvér TI C2000 MCU F28069M Piccolo. Po vygenerovaní kódu sa otvorí správa o generovaní kódu v ktorej sú uvedené vygenerované súbory v jazyku C, takto vygenerovaný kód už nie je možné upraviť Vygenerovaný kód je čitateľný a obsahuje komentáre. Vygenerovaný kód v jazyku C alebo C++ je možné upraviť v programe od TI Code Composer Studio.

```
20  #include "GP_T.h"
21  #include "rtwtypes.h"
22
23  volatile int IsrOverrun = 0;
24  static boolean_T OverrunFlag = 0;
25  void rt_OneStep(void)
26  {
27    /* Check for overrun. Protect OverrunFlag against preemption */
28    if (OverrunFlag++) {
29      IsrOverrun = 1;
30      OverrunFlag--;
31      return;
32    }
33
34    enableTimer0Interrupt();
35    GP_T_step();
36
37    /* Get model outputs here */
38    disableTimer0Interrupt();
39    OverrunFlag--;
40  }
41
42  volatile boolean_T stopRequested = false;
43  volatile boolean_T runModel = false;
44  int main(void)
45  {
46    float modelBaseRate = 0.5;
47    float systemClock = 90;
48
49    /* Initialize variables */
50    stopRequested = false;
51    runModel = false;
52    c2000_flash_init();
53    init_board();
54
55  #ifdef MW_EXEC_PROFILER_ON
56
57    config_profilerTimer();
58
59  #endif
60
61    ;
62    bootloaderInit();
63    rtmSetErrorStatus(GP_T_M, 0);
64    GP_T_initialize();
65    configureTimer0(modelBaseRate, systemClock);
66    runModel =
67      rtmGetErrorStatus(GP_T_M) == (NULL);
68    enableTimer0Interrupt();
69    globalInterruptEnable();
70    while (runModel) {
71      stopRequested = !(
72                          rtmGetErrorStatus(GP_T_M) == (NULL));
73    }
74
75    /* Disable rt_OneStep() here */
76
77    /* Terminate model */
78    GP_T_terminate();
79    globalInterruptDisable();
80    return 0;
81  }
```

Obrázok 2.2: Príklad vygenerovaného kódu

2.2 ANALÝZA MOŽNOSTÍ PROGRAMOVANIA KOMUNIKÁCIÍ

Rozšírenie Embedded Coder-a pre prostredie Simulink obsahuje aj bloky pre programovanie komunikácií CAN, SPI, I2C, a SCI.



Obrázok 2.3: Bloky pre programovanie komunikácií Zdroj: (MathWorks, 2019)

2.2.1 Sériová komunikácia

Na obrázku (Obrázok 2.3) sú uvedené bloky z prostredia simulink na programovanie komunikácií pre MCU TI C2000. Pre každý typ komunikácie je prijímací a vysielací blok. Bloky obsahujú vstup na ktorý privedieme správu, ktorú chceme poslať a výstup na ktorom je prijatá správa .

Nastavenie SPI komunikácie:

- SPI modul
- Polarita hodín
- Fázový posun hodín
- Adresa zariadenia slave
- Počet prenášaných bitov
- Metóda výberu zariadenia slave

Nastavenie I2C komunikácie:

- I2C modul
- Formát adresovania

- Adresa zariadenia slave
- Počet bitov
- NACK bit
- Frekvenciu s akou sú dáta čítané
- Dátový typ

Nastavenie SCI komunikácie:

- Výber SCI modulu
- Začiatok dát
- Koniec dát
- Dátový typ
- Dĺžka dát
- Frekvenciu s akou sú dáta čítané

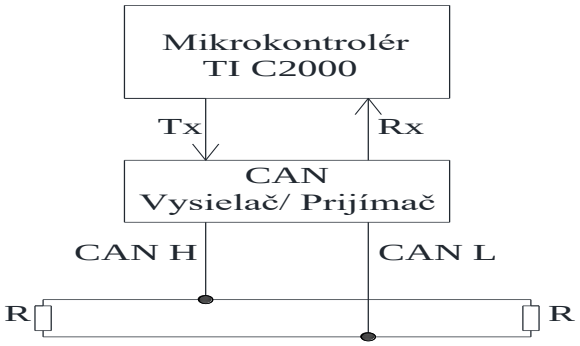
2.2.2 Komunikácia na báze CAN

CAN je sériový komunikačný protokol, ktorý bol vyvinutý firmou Bosch pre automobilový priemysel a je definovaný normou ISO 11898. Medzi dôvody použitia zbernice CAN patrí nízka cena, spoľahlivosť a vysoká prenosová rýchlosť. CAN umožňuje distribuované riadenie systémov v reálnom čase s vysokým zabezpečením proti chybám. Každý uzol na zbernici môže byť zariadenie master, preto nie je potrebné riadiť celú sieť z jedného nadradeného zariadenia. Správy vysielané po zbernici CAN neobsahujú informácie o cieľovom zariadení, pre ktoré sú určené. Správy sú prijímané všetkými zariadeniami, každá správa má svoj identifikátor, ktorý udáva význam správy a jej prioritu. Identifikátor určuje pre ktoré zariadenie je správa určená. CAN zbernica je rozdelená do troch vrstiev (Freescale, 1998), (Polák, 2003):

- Vrstva objektov
- Transportná vrstva
- Fyzická vrstva

Vrstva objektov slúži na filtrovanie správ. Úlohou transportnej vrstvy je prenosový protokol, riadenie rámcov, kontrola chyb. V transportnej vrstve sa rozhoduje či je

zbernica voľná pre prenos alebo príjem dát. Fyzická vrstva slúži k prenosu jednotlivých bitov medzi zariadeniami. Norma protokolu CAN definuje dve hodnoty bitu na zbernici dominantný a recesívny. Zbernicu tvoria dva vodiče označované ako CAN H a CAN L. Pre elimináciu odrazov na vedení sú na oboch koncoch ukončovacie odpory s hodnotou 120 Ω. Príklad fyzickej realizácie siete CAN je na nasledujúcom obrázku (Obrázok 2.4). Maximálna prenosová rýchlosť je 1 Mbit/s (Freescale, 1998), (Polák, 2003).



Obrázok 2.4: Fyzická realizácia siete CAN Zdroj: (Freescale, 1998)

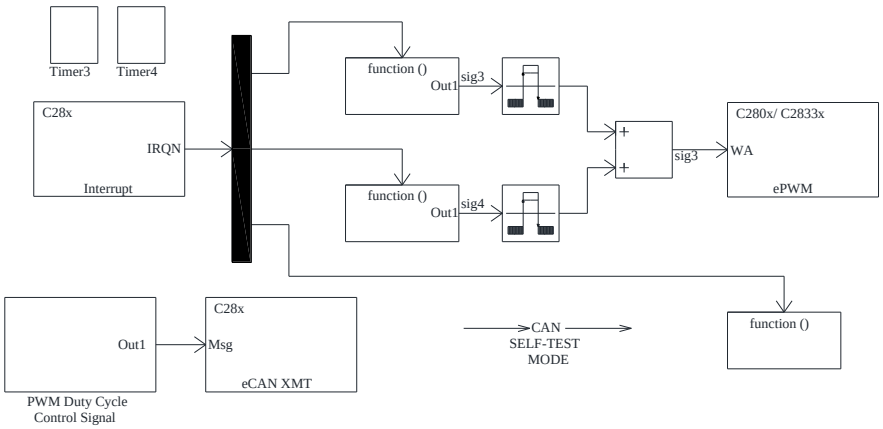
V prostredí Simulink sú pre nastavenie komunikácie CAN tri bloky, kalibračný protokol CAN, blok pre príjem a vysielanie. V týchto blokoch môžeme nastaviť nasledujúce parametre:

- Adresu cieľového zariadenia
- CAN modul
- Typ identifikátora
- Identifikátor správy
- Dátový typ
- Číslo mailboxu
- Vyvolanie prerušení

2.3 ANALÝZA PRÍKLADOV POPÍSANÝCH V DOKUMENTÁCII K NÁSTROJU EMBEDDED CODER

1. Príklad ADC PWM synchronizácia pomocou prerušenia od A/D prevodníka. Tento príklad ukazuje, ako použiť A/D prevodník na vzorkovanie analógového signálu a ako použiť blok PWM na vytvorenie PWM signálu. Taktiež ukazuje ako použiť blok hardvérového prerušenia na zmenu šírky impulzu PWM pomocou hodnoty z A/D prevodníka. Príklad je uvedený v elektronickej prílohe č.1 (MathWorks, 2020).

2. Príklad má názov Asynchrónne riadenie a ukazuje, ako používať periférne zariadenia TI C28x a bloky hardvérového prerušenia na asynchrónne volanie funkcií subsystémov Simulink (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.2.



Obrázok 2.5: Asynchrónne riadenie Zdroj: (MathWorks, 2020)

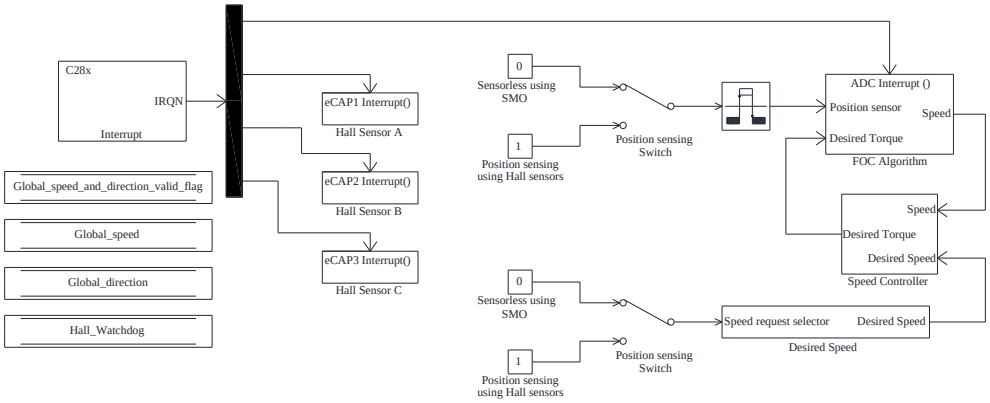
Bloky časovača alebo PWM sa používajú na konfiguráciu prerušenia od časovača. Prerušenia od časovača sa spúšťajú na základe periódy časovača a prerušenie od CAN sa spustí, keď je správa prijatá. Blok hardvérového prerušenia spúšťa obslužné rutiny prerušenia (ISR) pre prerušenia od časovača a aj pri prerušení od CAN. ISR volajú subsystémy pripojené k výstupu z bloku hardvérového prerušenia. Výstup z prvých dvoch subsystémov sú voľno bežiace počítadlá a ich súčet riadi zmenu šírky impulzu

ePWM2, šírka impulzu sa lineárne mení z 0% po 100%. Tretí subsystém obsahuje prijímací blok CAN, ktorého výstup riadi zmenu šírky impulzu bloku PWM1. Šírka impulzu sa pohybuje od 0% do 100%, pretože správy sú prijímané z vysielacieho bloku CAN. Tretí subsystém slúži k automatickému testu CAN (MathWorks, 2020).

3. Príklad je použitie kalibračného protokolu CAN na monitorovanie a ladenie. V tomto príklade je použitý kalibračný protokol CAN na monitorovanie signálov z modelu a vyladenie parametrov v aplikačnom kóde spustenom na cieľovom hardvéri. Parametre je možné načítať do flash pamäte a skopírovať do pamäte RAM počas inicializácie (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.3.

4. Príklad je simulácia vektorového riadenia (FOC) pre synchrónny motor s permanentnými magnetmi (PMSM). Príklad je uvedený v elektronickej prílohe č.4.

5. Príklad je riadenie trojfázového synchrónneho motora s permanentnými magnetmi v uzavretej slučke pomocou FOC (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.5.



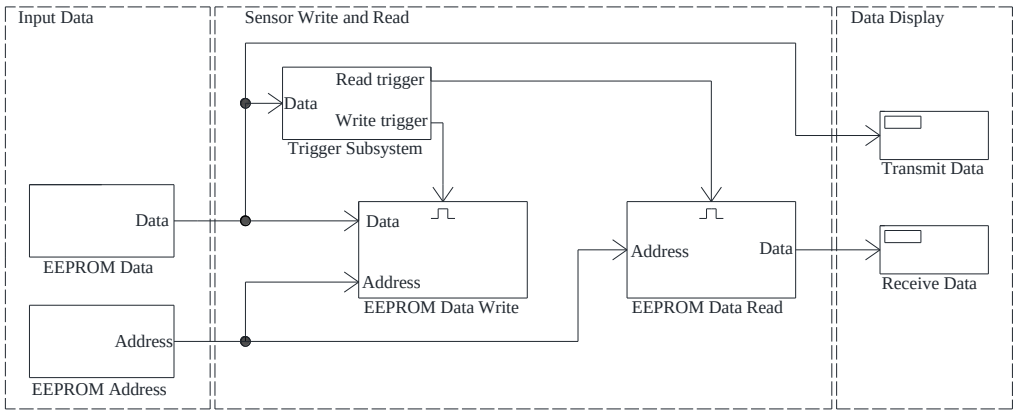
Obrázok 2.6: Riadenie PMSM Zdroj: (MathWorks, 2020)

6. Príklad využíva CLA blok. CLA je koprocesor ktorý umožňuje paralelné spracovanie dát. CLA uvoľní hlavný procesor aby mohol súčasne vykonávať ďalšie úlohy (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.6.

7. Príklad ukazuje ako nakonfigurovať a používať bloky SPI komunikácie pre zápis a čítanie dát. Príklad je uvedený v elektronickej prílohe č.7.

8. Príklad využíva I2C zbernicu ku komunikácií so senzormi. Príklad je uvedený v elektronickej prílohe č.8. Tento príklad je rozdelený na dve časti. Prvá časť pristupuje pomocou I2C zbernice k pamäti EEPROM, číta a zapisuje do nej dáta. Model sa skladá z troch častí:

- Vstupné dáta
- Zápis a čítanie
- Zobrazenie výstupných dát



Obrázok 2.7:Použitie zbernice I2C pre prístup k EEPROM

Zdroj: (MathWorks, 2020)

V druhej časti sa pomocou I2C zbernice čítajú dáta z akcelerometra a gyroskopu. Model pozostáva z nasledujúcich subsystémov:

- Inicializačný subsystém
- Subsystém pre čítanie zo snímačov
- Subsystém pre spracovanie údajov

9. Príklad je modelovanie regulátora napätia pre DC/DC znižujúci menič. V tomto príklade je vymodelovaný regulátor napätia pre DC/DC znižujúci menič s použitím knižníc Embedded Coder-a pre TI C2000 procesory. Model v tomto príklade je určený pre TI F28377S alebo F28379D (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.9. Spustením tohto modelu môžeme:

- Simulovať regulátor pre DC/DC znižujúci menič

- Vygenerovať kód a nahrať ho do MCU TI C2000
- Sledovať a upravovať parametre regulátora, záťaže a výstupného napätia

10. Príklad využíva k zmene šírky impulzu PWM priameho prístupu do pamäte (DMA). Použitím DMA je skopírovaný sínusový signál z vyhľadávacej tabuľky do porovnávacieho registra PWM (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.10.

11. Príklad je kompenzácia účinníka (PFC) s použitím zvyšujúceho meniča. Tento príklad je určený pre mikrokontrolér F28035 a v tomto príklade môžeme (MathWorks, 2020):

- Simulovať PFC
- Vygenerovať kód a nahrať ho do cieľového hardvéru

Príklad je uvedený v elektronickej prílohe č.11.

V 12. Príklade je vysvetlené ako nakonfigurovať a použiť LCD displej na zobrazenie obrázka pomocou periférií C28x pre procesory TI C2000. Príklad je uvedený v elektronickej prílohe č.12 (MathWorks, 2020).

13. Príklad je fotovoltický menič. Tento príklad využíva mikrokontrolér TI F28035 a TI Solar Explorer Kit. S použitím tohto príkladu môžeme (MathWorks, 2020):

- Simulovať model pre fotovoltický systém
- Testovať výkon a vyladiť regulátor
- Vygenerovať kód pre cieľový hardvér
- Monitorovať signály a upravovať parametre

Príklad je uvedený v elektronickej prílohe č.13.

14. Príklad je overenie kódu pomocou PIL. V tomto príklade je vysvetlené ako nakonfigurovať model zo Simulink-u aby sme mohli spustiť simuláciu procesora v slučke (PIL). Pri PIL simulácií kód beží na cieľovom hardvéri (MathWorks, 2020). Príklad je uvedený v elektronickej prílohe č.14.

3 NÁVRH A REALIZÁCIA VYBRANÝCH PRÍKLADOV V PODOBE NÁVRHU VÝUČBOVÝCH MATERIÁLOV

3.1 VŠEOBECNÉ INFORMÁCIE K NÁVRHU A REALIZÁCII VYBRANÝCH PRÍKLADOV

V nasledujúcej časti sú navrhnuté a realizované príklady pre MCU TI Piccolo F28069M LaunchPad v prostredí MATLAB/ Simulink. Príklady sú realizované v podobe výučbových materiálov vhodných pre výučbu. Pri príkladoch sú uvedené modely z prostredia Simulink, vygenerované zdrojové kódy a inštruktážne videá, ktoré sú uvedené v elektronických prílohách.

Požadovaný softvér:

- MATLAB 9.7
- Simulink 10.0
- Embedded Coder 7.3
- MATLAB Coder 4.3
- Simulink Coder 9.2
- Embedded Coder Support Package for Texas Instruments C2000 processors
- Texas Instruments Control Suite 3.4.9
- Texas Instrument Code Composer Studio 8.3.0
- Texas Instrument C2000 Ware 2_00_00_02
- Texas Instrument F28044 Headers 130

Požiadavky k príkladom: Znalosť prostredia MATLAB/ Simulink. Znalosť prostredia Code composer Studio a programovania TI C2000.

3.2 PRÍKLAD NASTAVENIA GPIO PRE MIKROKONTROLÉR TI C2000 POMOCOU MATLAB/ SIMULINK

Názov laboratórneho cvičenia : Nastavenie GPIO pre mikrokontrolér TI C2000 pomocou MATLAB/ Simulink

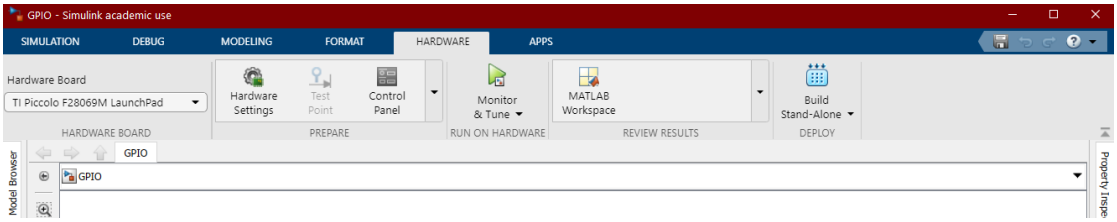
Požadovaný hardvér: TI Piccolo F28069M LaunchPad, LED diódy, Tlačidlo.

Kontrolné otázky pred cvičením:

- Čo je to GPIO pin ?
- Čo pripojíme na vstupný pin ?
- Čo pripojíme na výstupný pin ?

Zadanie: Vytvorte program v ktorom bude jedno tlačidlo a dve LED diódy, ak tlačidlo nebude stlačené bude svietiť LED1 a LED2 nebude svietiť. Ak stlačíme tlačidlo LED1 nebude svietiť a rozsvieti sa LED2.

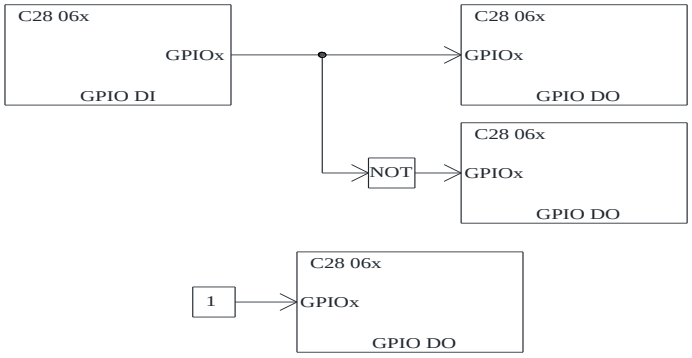
Riešenie: V prostredí Simulink najskôr nastavíme hardvér. V záložke hardvér klikneme na nastavenie hardvéru, otvorí sa okno v ktorom vyberieme hardvér TI Piccolo F28069M LaunchPad (Obrázok 3.1).



Obrázok 3.1: Prostredie Simulink Zdroj: MATLAB 2019b

V programe použijeme blok digitálny vstup. Signál z digitálneho vstupu je pripojený na dva bloky digitálneho výstupu na ktoré sú pripojené LED diódy. Pred jedným blokom digitálneho výstupu je použitý logický člen NOT ktorý neguje hodnotu ktorá ide na výstup. Vstup je na GPIO 12, dátový typ je booleovský a vzorkovací čas 0,1s. Výstupy sú na GPIO 18 a GPIO 22. Na výstup GPIO 34 je

privedená konštanta ktorá každých 0,5 s mení hodnotu na pine a slúži len na kontrolu. Model je na obrázku (obrázok 3.2). Po zostavení modelu môžeme vygenerovať a nahrať kód do cieľového hardvéru alebo len vygenerovať kód. Po vygenerovaní kódu sa otvorí správa o generovanom kóde, ktorá obsahuje obsah a vygenerovaný kód.



Obrázok 3.2: Program pre príklad GPIO Zdroj: MATLAB 2019b

Vygenerovaný kód je rozdelený do súborov uvedených v nasledujúcej tabuľke (Tabuľka 3.1). V súbore ert_main.c sú volané funkcie pre inicializáciu GPIO pinov (GPIO_initialize), stlačenie tlačidla (GPIO_step0), kontrolnú LED diódu (GPIO_step1) a funkcie pre nastavenie hardvéru. V súbore GPIO.c sú definované funkcie GPIO_initialize, GPIO_step0 a GPIO_initialize.

V súbore GPIO.h sú definované dátové typy konštánt a premenných. V súbore GPIO_data.c je definovaná hodnota konštanty. V ostatných súboroch je nastavenie periférií MCU a nastavenie MCU.

Kontrolne otázky po cvičení:

- Aký typ rezistora je použitý pre vstupný pin a ako by ste ho zmenili ?
- Vo vygenerovanom kóde vyhľadajte funkciu pre stlačenie tlačidla
- Zmeňte vzorkovací čas tlačidla na 1 sekundu
- Ako ste vyriešili aby LED1 svietila a LED2 nesvietila ?
- Vo vygenerovanom kóde vyhľadajte funkciu pre inicializáciu GPIO pinov

Výstupy: Model v prostredí Simulink, inštruktážne video a vygenerovaný kód je uvedený v elektronickej prílohe č.15.

Tabuľka 3.1: Vygenerované súbory pre príklad GPIO Zdroj: MATLAB 2019b

Hlavný súbor	ert_main.c		Ostatné súbory	F2806x_CodeStartBranch.asm
Súbory modelu	GPIO.c			F2806x_CpuTimers.c
	GPIO.h			F2806x_DefaultIsr.c
	GPIO_private.h			F2806x_GlobalVariableDefs.c
	GPIO_types.h			F2806x_usDelay.asm
Dátové súbory	GPIO_data.c			MW_c28xx_board.c
Pomocné súbory	rtwtypes.h			MW_c28xx_csl.c
Súbory rozhrania	GPIO.a2l			MW_c28xx_pie.h
	rtmodel.h			MW_target_hardware_resources.h
Ostatné súbory	F2806x_Adc.c			blapp_support.c
	F2806x_PieCtrl.c			c2806xBoard_Realtime_Support.c
	F2806x_PieVect.c			c2806xSchedulerTimer0.c
	F2806x_SysCtrl.c			profiler_Support.c
	F2806x_dma.c			

3.3 PRÍKLAD ZMENY ŠÍRKY IMPULZU PWM POMOCOU MIKROKONTROLÉRA TI C2000 V PROSTREDÍ MATLAB/ SIMULINK

Názov laboratórneho cvičenia : Zmena šírky impulzu PWM pomocou MCU TI C2000 v prostredí MATLAB/ Simulink.

Požadovaný hardvér: TI Piccolo F28069M LaunchPad, Osciloskop.

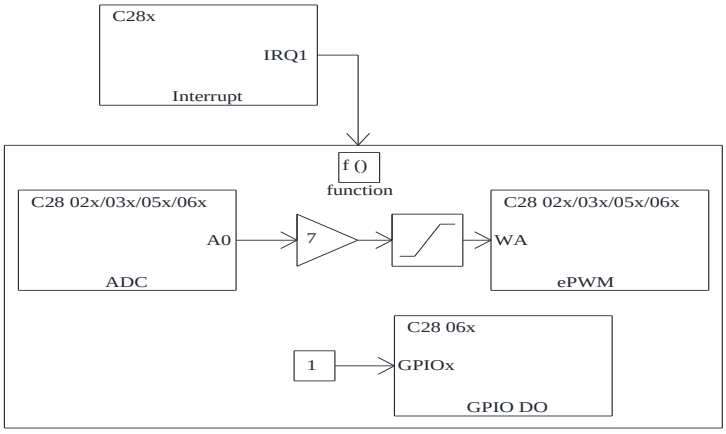
Kontrolné otázky pred cvičením:

- Koľko Bitov má časovač PWM ?
- Koľko Bitov má A/D prevodník ?
- Ako vypočítame periódu časovača PWM ?
- Čo musíme nastaviť v prerušení ?
- Aké sú režimy počítania časovača PWM ?
- K čomu slúži blok saturácie ?

Zadanie: Vytvorte program v ktorom budete meniť šírku impulzu PWM pomocou potenciometra. V programe využite prerušenie. Frekvenciu PWM nastavte na 50 Hz a šírka impulzu sa bude meniť od 10% do 90%.

Riešenie: V programe použijeme blok hardvérového prerušenia v ktorom nastavíme prerušenie od A/D prevodníka. Prerušenie bude vyvolané po skončení prevodu A/D prevodníka a spustí sa podprogram. V podprograme sa nachádza blok A/D prevodníka ktorého výstup sa vynásobí pretože výstup z A/D prevodníka je maximálne 4095 a perióda časovača je nastavená na 28125. Z bloku násobenia ide hodnota do saturačného bloku ktorý zabezpečuje aby sa šírka impulzu pohybovala od 10% do 90%. Hodnota zo saturačného bloku vchádza do bloku PWM kde mení hodnotu v CMPA registri, ktorá mení šírku impulzu. V bloku A/D prevodníka nastavíme parametre prevodníka. Začiatok prevodu bude keď časovač PWM dosiahne hodnotu periódy PWM. V bloku PWM nastavíme modul PWM, periódu časovača pre použitý režim počítania vypočítame pomocou vzorca (3.1), kde SYSCLK je frekvencia jadra, TBCLK a HSPCLKDIV sú registre preddeličky a f je frekvencia. Režim počítania je nastavený na up-down.

$$T = \frac{SYSCLK}{2 * TBCLK * HSPCLKDIV} * \frac{1}{f} \qquad (3.1)$$



Obrázok 3.3: Program pre zmenu šírky impulzu Zdroj: MATLAB 2019b

Vstupný kanál A/D prevodníka je ADCINA0, spustenie prevodu je od PWM1A a po skončení prevodu sa vygeneruje prerušenie ADCINT1. Čísla prerušení pre ADCINT1, číslo CPU je 1 a číslo PIE je 1. Hodnota z A/D prevodníka je násobená číslom 7. Blok saturácie je nastavený od 1406 do 26718. Hodnota z bloku saturácie vstupuje do bloku PWM a mení šírku impulzu. V bloku PWM je nastavená hodnota

CMPA pomocou vstupného portu. Po vygenerovaní kódu sa otvorí správa o generovanom kóde, ktorá obsahuje súbory uvedené v tabuľke (Tabuľka 3.2).

Tabuľka 3.2: Vygenerované súbory pre zmenu šírky impulzu Zdroj: MATLAB 2019b

Hlavný súbor	ert_main.c		Ostatné súbory	F2806x_CodeStartBranch.asm
Súbory modelu	ADCPWM.c			F2806x_CpuTimers.c
	ADCPWM.h			F2806x_GlobalVariableDefs.c
	ADCPWM_private.h			F2806x_usDelay.asm
	ADCPWM_types.h			MW_c28xx_adc.c
Dátové súbory	ADCPWM_data.c			MW_c28xx_board.c
Pomocné súbory	rtwtypes.h			MW_c28xx_csl.c
Súbory rozhrania	rtmodel.h			MW_c28xx_pie.h
Ostatné súbory	F2806x_Adc.c			MW_c28xx_pwm.c
	F2806x_PieCtrl.c			MW_target_hardware_resources.h
	F2806x_PieVect.c			blapp_support.c
	F2806x_SysCtrl.c			c2806xBoard_Realtime_Support.c
	F2806x_dma.c			c2806xSchedulerTimer0.c
	F2806x_DefaultIsr.c			profiler_Support.c

V súbore ert_main.c sú volané funkcie pre jednotlivé bloky. V ADCPWM.c sú definované funkcie prerušenia a nastavenia PWM modulu. Funkcia podprogramu prerušenia má názov isr_int1pie1_task_fcn a obsahuje zápis výsledku z A/D prevodníka do premennej, násobenie hodnoty z A/D prevodníka, funkciu saturácie a zápis hodnoty do registra CMPA. Druhá funkcia má názov ADCPWM_initialize. V tejto funkcií je volaná funkcia pre nastavenie A/D prevodníka a nastavenie PWM modulu. V súbore ADCPWM.h sú definované dátové typy premenných a konštánt v bloku násobenia, saturácie a v bloku konštanty. V súbore ADCPWM_data.c sú uvedené hodnoty konštánt. V MW_c28xx_adc.c je funkcia pre nastavenie bloku A/D prevodníka. Súbor MW_c28xx_board.c slúži na inicializáciu hardvéru. V súbore MW_c28xx_csl.c je definovaná funkcia pre prerušenie od A/D prevodníka a povolenie prerušení. V MW_c28xx_pwm.c je nastavený pin GPIO0 ako výstup PWM1A.

Kontrolne otázky po cvičení:

- Vo vygenerovanom kóde nájdite funkcie pre bloky saturácie, násobenia a zápis hodnoty do registra PWM
- Ktorý register PWM slúži k zmene šírky impulzu ?
- K čomu slúžia registre CAD a CAU ?

- V ktorom vygenerovanom súbore sú uložené hodnoty bloku saturácie a násobenia ?

Výstupy: Model v prostredí Simulink, inštruktážne video a vygenerovaný kód je uvedený v elektronickej prílohe č.16.

3.4 PRÍKLAD MERANIA VNÚTORNEJ TEPLOTY MIKROKONTROLÉRA TI C2000 POMOCOU MATLAB/ SIMULINK

Názov laboratórneho cvičenia : Meranie vnútornej teploty mikrokontroléra TI C2000 pomocou MATLAB/ Simulink.

Požadovaný hardvér: TI Piccolo F28069M LaunchPad.

Kontrolné otázky pred cvičením:

- Čo je CLA ?
- Ako je pripojený tepelný senzor ?
- Aký je vzorec pre prepočet na stupne Celzia ?

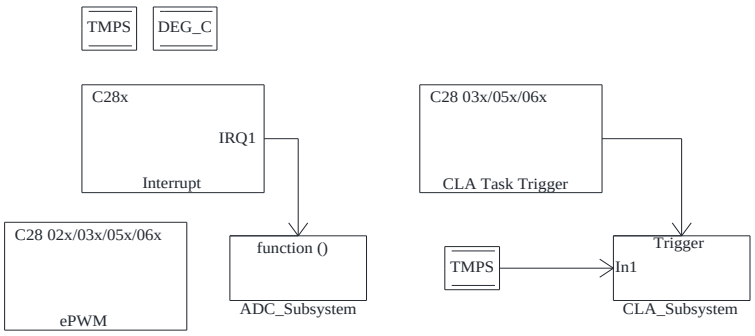
Zadanie: Zmerajte vnútornú teplotu mikrokontroléra TI Piccolo F28069M LaunchPad pomocou vnútorného tepelného senzora. Teplotu zobrazte v stupňoch Celzia a pre výpočet využite blok CLA. Vzorový príklad je uvedený v technickej referenčnej príručke daného mikrokontroléra.

Riešenie: Výstup zo senzora je pripojený na vstup A/D prevodníka ADCINA5. Výstup zo senzora pripojíme pomocou bitu ADCCTL1.TEMPCONV. Pre prepočet hodnoty z A/D prevodníka na stupne Celzia použijeme rovnicu (3.2). Vo vzorci 3.2 premenná SensorSample je výstup zo senzora, konštanta Offset je uložená na adrese 0x3D7E85 a jej hodnota je 1678. Konštanta Slope je na adrese 0x3D7E82 a jej hodnota je 5598. Hodnota KELVIN je 273, KELVIN_OFF= FP_SCALE*KELVIN. Konštanta FP_SCALE je mierka pre čísla s pevnou desatinnou čiarkou Q15 a jej hodnota je 32768 a konštanta FP_ROUND= FP_SCALE/2

$$\text{DEG\_C} = \frac{(\text{SensorSample} - \text{Offset}) * \text{Slope} + \text{FP\_ROUND} + \text{KELVIN\_OFF}}{\text{FP\_SCALE}} - \text{KELVIN} \qquad (3.2)$$

Po úprave rovnice (3.2) získame jednoduchšiu rovnicu (3.3), ktorú je výhodnejšie použiť pre výpočet.

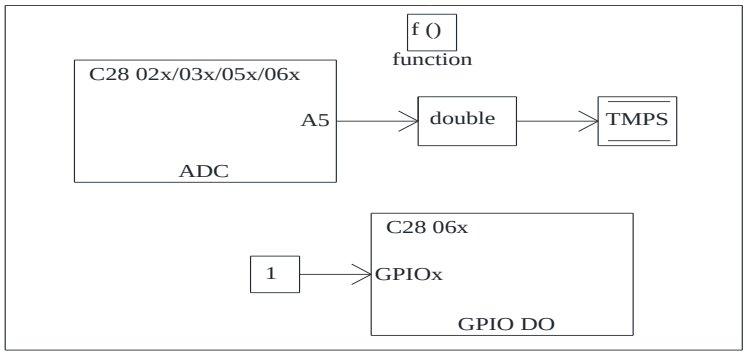
$$\text{DEG\_C} = (\text{SensorSample} - \text{Offset}) * 0{,}1708 + 0{,}5 \qquad (3.3)$$



Obrázok 3.4: Model pre tepelný senzor Zdroj: MATLAB 2019b

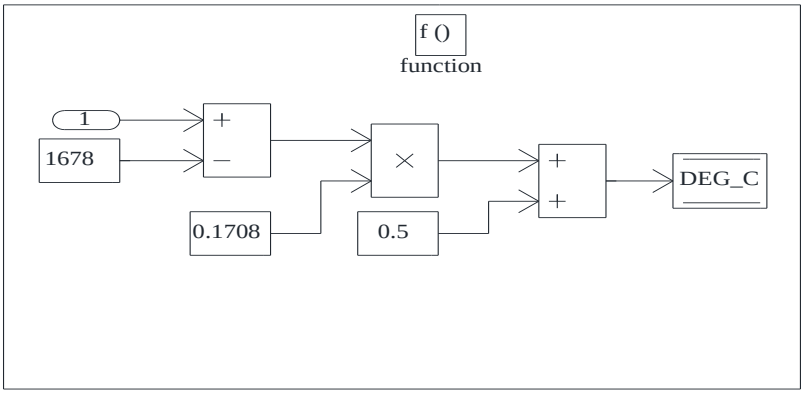
Na obrázku (Obrázok 3.4) je model pre príklad tepelný senzor. Model obsahuje viacero blokov. Blok PWM má nastavenú periódu na 2 sekundy a slúži na spustenie prevodu A/D prevodníka. Začiatok prevodu A/D prevodníka bude každé 2 sekundy. Prerušenie bude vyvolané po skončení prevodu A/D prevodníka a spustí podprogram ADC_Subsystem. Bloky pre uloženie dát slúžia na uloženie premenných. Do premennej TMPS je uložený výsledok z A/D prevodníka a táto hodnota vstupuje do podprogramu CLA_Subsystem. V premennej DEG_C je uložená hodnota z A/D prevodníka prepočítaná na stupne Celzia. V bloku CLA je natavená CLA úloha 1, spúšťanie je softvérové a je spúšťaná každú sekundu. CLA úloha spustí podprogram CLA_Subsystem. Dátový typ je vo všetkých blokoch nastavený na double.

Na obrázku (Obrázok 3.5) je podprogram ktorý sa spustí keď nastane prerušenie od A/D prevodníka ADCINT1. Vstup A/D prevodníka je nastavený na ADCINA5 pretože k nemu je pripojený tepelný senzor. V bloku A/D prevodníka nemáme možnosť sa pripojiť k tepelnému senzoru preto budeme musieť pripojiť tepelný senzor k ADCINA5 až vo vygenerovanom kóde. Výstup z A/D prevodníka prevedieme na dátový typ double a hodnotu zapíšeme do premennej TMPS.



Obrázok 3.5: ADC_Subsystem Zdroj: MATLAB 2019b

Na obrázku (Obrázok 3.6) je podprogram CLA. V tomto podprograme je prepočet hodnoty z A/D prevodníka na stupne Celzia pomocou rovnice (3.3) v ktorej namiesto premennej SensorSample dosadíme premennú TMPS. Výsledok zapíšeme do premennej DEG_C. Pri nastavovaní podprogramu klikneme na vlastnosti signálu vychádzajúceho z bloku In1, zadáme názov signálu. V záložke generovanie kódu triedu signálu nastavíme na tic2000demospkg.Signal a pamäťovú triedu na CPUToCla1MsgRAM. V bloku pre uloženie dát DEG_C v záložke vlastnosti signálu nastavíme triedu signálu na tic2000demospkg.Signal a pamäťovú triedu na ClaToCpuMgsRAM. Po zostavení a nastavení modelu zvolíme vygenerovať kód. Kód nemôžeme nahrať a spustiť pretože nie je pripojený tepelný senzor.



Obrázok 3.6: CLA_Subsystem Zdroj: MATLAB 2019b

Tabuľka 3.3: Vygenerované súbory pre tepelný senzor Zdroj: MATLAB 2019b

Hlavný súbor	ert_main.c		Ostatné súbory	F2806x_CodeStartBranch.asm
Súbory modelu	TEMP_SENS.c			F2806x_CpuTimers.c
	TEMP_SENS.h			F2806x_DefaultIsr.c
	TEMP_SENS_private.h			F2806x_GlobalVariableDefs.c
	TEMP_SENS_types.h			F2806x_usDelay.asm
	cla_task.cla			MW_c28xx_adc.c
Dátové súbory	cla_header.h			MW_c28xx_board.c
Pomocné súbory	rtwtypes.h			MW_c28xx_csl.c
Súbory rozhrania	TEMP_SENS.a2l			MW_c28xx_pie.h
	rtmodel.h			MW_c28xx_pwm.c
Ostatné súbory	F2806x_Adc.c			MW_target_hardware_resources.h
	F2806x_PieCtrl.c			blapp_support.c
	F2806x_PieVect.c			c2806xBoard_Realtime_Support.c
	F2806x_SysCtrl.c			c2806xSchedulerTimer0.c
	F2806x_dma.c			profiler_Support.c

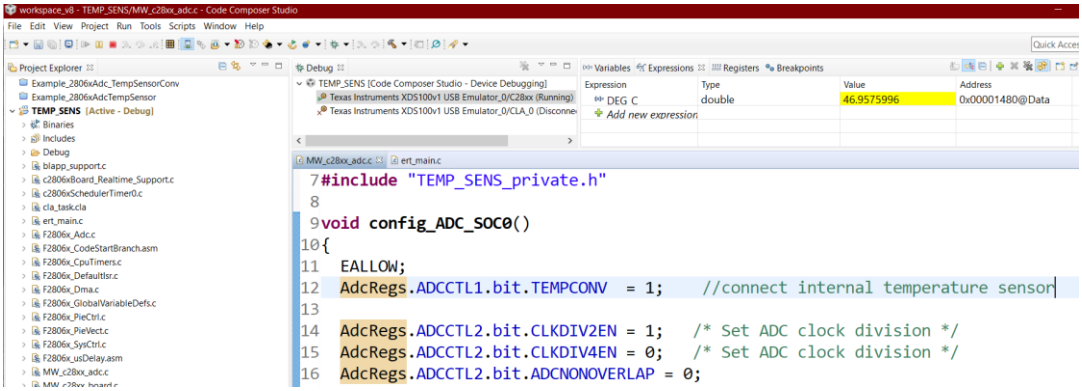
Vygenerovaný kód bude obsahovať súbory uvedené v tabuľke (Tabuľka 3.3). V súbore ert_main.c sú funkcie pre inicializáciu blokov, povolenie prerušení a nastavenie hardvéru.

V súbore TEMP_SENS.c je definovaná funkcia podprogramu prerušenia isr_int1pie1_task_fcn, v ktorej je zápis výsledku z A/D prevodníka do premennej TMPS. Funkcia CLA úlohy TEMP_SENS_step. Funkcia pre nastavenie CLA, PWM a A/D prevodníka ktorá má názov TEMP_SENS_initialize. V cla_task.cla je výpočet v podprograme CLA Subsystem.

Pre pripojenie tepelného senzora na ADCINA musíme vygenerovaný kód upraviť v programe Code Composer Studio. Otvoríme Code Composer Studio v záložke súbor zvolíme importovať CCS projekt. Vyberieme priečinok v ktorom je vygenerovaný kód. V nájdených projektoch zvolíme kód vygenerovaný pre tepelný senzor a klikneme na dokončiť. Otvoríme súbor MW_c28xx_adc.c a do funkcie config_ADC_SOC0() pridáme riadok kódu (3.4) pre pripojenie tepelného senzora (Obrázok 3.7).

```
AdcRegs.ADCCTL1.bit.TEMPCONV = 1;
```
(3.4)

Po pripojení senzora môžeme nahrať kód do MCU a po nahratí dáme spustiť. Zmenu teploty môžeme sledovať v premennej DEG_C ako je uvedené na obrázku (Obrázok 3.7).



Obrázok 3.7: Prostredie Code Composer Studio

Zdroj: Code Composer Studio 8.3.0

Kontrolne otázky po cvičení:

- Ako ste pripojili tepelný senzor ?
- Upravte vzorec aby bola výsledná hodnota v stupňoch Kelvina
- Vo vygenerovanom kóde nájdite kde je výpočet v podprograme CLA
- V ktorom súbore je funkcia pre nastavenie A/D prevodníka ?

Výstupy: Model v prostredí Simulink, inštruktážne video a vygenerovaný kód je uvedený v elektronickej prílohe č.17.

3.5 ROZDIELY PROGRAMOVANIA POMOCOU MODELU OPROTI JAZYKU C

Pri programovaní pomocou modelu sa model skladá z blokov. Tieto bloky obsahujú funkcie, napríklad pre matematické operácie, nastavenie periférií, ukladanie dát. Pri programovaní pomocou jazyka C by sme museli tieto funkcie napísať. Programovanie pomocou modelu je rýchlejšie a prehľadnejšie ale bloky neposkytujú niektoré funkcie ako napríklad v príklade merania vnútornej teploty mikrokontroléra kde sme museli pripojiť tepelný senzor až po vygenerovaní kódu alebo ak by sme chceli pre vstupný pin nastaviť pull-down rezistor.

ZÁVER

V diplomovej práci sa venujeme programovaniu vnorených systémov na báze mikroprocesorov Texas Instruments C2000 v prostredí MATLAB.

Prvý cieľ práce definície a popis pojmov sa nachádza v kapitole 1. V podkapitolách kapitoly 1 sú definované pojmy vnorené systémy, architektúry vnorených systémov, vývoj založený na modeli (MBD), automatické generátory kódu, MATLAB, Simulink, Simulink Coder, Embedded Coder, MATLAB Coder, štandard MISRA, norma ISO 26262, ASIL, mikrokontroléry TI C2000 a študijný program autotronika.

Druhý cieľ analýza možností generátorov kódu na báze programovacieho prostredia MATLAB pre procesory TI C2000 je v kapitole 2. V kapitole 2.1 sú popísané bloky z prostredia Simulink pre mikrokontroléry TI C2000. V kapitole 2.2 sú analyzované bloky pre programovanie komunikácií. V kapitole 2.3 sú analyzované príklady, ktoré sa nachádzajú v dokumentácií k Embedded Coder. Ku každému príkladu je v elektronickej prílohe uvedený model a webová stránka na ktorej sa príklad nachádza.

Tretí cieľ navrhnúť a realizovať vybrané príklady vhodné pre študijný program autotronika je uvedený v kapitole 3. V tejto kapitole sa nachádzajú tri príklady ktoré sú spracované ako výučbové materiály. K príkladom sú v elektronických prílohách uvedené modely z prostredia Simulink, vygenerované kódy a inštruktážne videá. V kapitole 3.5 sú uvedené rozdiely medzi programovaním pomocou modelu a jazyka C.

Prínosom práce je návrh a realizácia príkladov vhodných pre študijný program autotronika a ich aplikácia vo výučbe. Myslím si že som splnil zadanie diplomovej práce.

Zoznam použitej literatúry

Banks, Andrew. 2016. MISRA C -. 2016.

Beine, Michael a Jungmann, Michael. 2004. Code Generation for Safety-Critical Systems. *pdfs.semanticscholar*. 2004.

DENNEY, E. a TRAC, S. 2008. *A Software Safety Certification Tool for Automatically Generated Guidance.* MT, USA : Big Sky, 2008. LOCAL: ...DP BARONIAK__ZDROJE\Denney-BigSky-08.pdf.

dspace. 2019. TargetLink. 01 2019.

Els, Peter. 2019. www.automotive-iq.com. *www.automotive-iq.com.* [Online] 23. 4 2019. [Dátum: 13. 2 2020.] https://www.automotive-iq.com/electrics-electronics/articles/what-is-iso-26262.

Embitel. 2018. www.embitel.com. *www.embitel.com.* [Online] 4 2018. [Dátum: 20. 2 2020.] https://www.embitel.com/blog/embedded-blog/understanding-how-iso-26262-asil-is-determined-for-automotive-applications.

Freescale. 1998. nxp. [Online] 1998. [Dátum: 26. 2 2020.] https://www.nxp.com/docs/en/reference-manual/BCANPSV2.pdf?fbclid=IwAR0-JpwffuvckFgcWEljGBLxvR2qlx_scXBf4cbNA1fEGtggAKMbm8WYtas.

Friedman, Jonathan. 2014. slideplayer.com. [Online] 2014. [Dátum: 20. 2 2020.] https://slideplayer.com/slide/3244628/.

HUMUSOFT. 2019. *HUMUSOFT/ Simulink real time.* [Online] 2019. [Dátum:] https://www.humusoft.cz/matlab/simulink-real-time/.

—. 2019. *HUMUSOFT/Matlab.* [Online] 2019. [Dátum: 8. 12 2019.] https://www.humusoft.cz/matlab/.

—. 2019. *HUMUSOFT/Simulink.* [Online] 2019. [Dátum: 8. 12 2019.] https://www.humusoft.cz/matlab/simulink/.

ISO_26262. 2018. *Road vehicles — Functional safety — Parts 1- 12.* 2018.

J.S.Freudenberg. 2019. www.ethz.ch. [Online] 31. 5 2019. [Dátum: 3. 11 2019.] https://www.ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Lectures/Embedded-Control-Systems/OtherNotes/Analyzing Generated Code.pdf.

Jackson, Chad. 2019. WHAT IS MODEL-BASED SOFTWARE DEVELOPMENT? *lifecycleinsights.* 2019.

Kindel, Olaf a Friedrich, Mario. 2009. *Softwareentwicklung mit AUTOSAR.* Heidelberg : dpunkt.verlag, 2009. s. 292. ISBN 978-3-89864-563-8.

KME. 2019. Študijný program AUTOTRONIKA. [Online] 2019. [Dátum: 4. 12 2019.] http://kmae.uniza.sk/images/Autotronika.pdf.

KYSLAN, K., a iní. 2019. Automatické generovanie kódu z prostredia MATLAB/Simulink a porovnanie vybraných prostriedkov pre jeho realizáciu. *Elektrorevue.* JUNE 2019, Zv. 21, 3, s. 68-75. LOCAL: ...\DP BARONIAK__ZDROJE\clanek_11_30.06.2019.pdf.

Liang, Tiffany. 2015. mathworks. [Online] 2015. [Dátum: 2. 12 2019.] https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/company/events/conferences/matlab-conference-australia/2015/proceedings/automatic-code-generation-for-embedded-control-systems.pdf.

MathWorks. 2019. *NXP support.* [Online] 2019. [Dátum: 8. 12 2019.] https://www.mathworks.com/hardware-support/nxp.html.

—. 2019. *TI C2000 support.* [Online] 2019. [Dátum: 8. 12 2019.] https://www.mathworks.com/hardware-support/ti-c2000.html?s_tid=AO_HS_info.

—. 2019. Embedded Coder. 2019.

—. 2019. mathworks.com. [Online] 2019. [Dátum: 23. 2 2020.] https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/referencelist.html?type=block&category=index&s_tid=CRUX_lftnav_block_c28x-generic-blocks.

—. 2019. MATLAB Coder. 2019.

—. 2019. Model-Based Design with simulink. 2019.

—. 2019. Simulink Coder. 2019.

—. 2020. uk.mathworks.com. [Online] 2020. [Dátum: 26. 2 2020.] https://uk.mathworks.com/help/supportpkg/texasinstrumentsc2000/examples.html.

MISRA. 2016. *MISRA Compliance.* Nuneaton : HORIBA MIRA Limited, 2016. s. 25. ISBN 978-1-906-400-13-2.

—. 2004. *MISRA-C:2004.* Nuneaton : MIRA Limited, 2004. 978-0-9524156-4-0.

Nagy, Roman. 2011. Procesné modely testovania softvéru. *ATP Journal.* 31. 5 2011, s. 3.

NI. 2019. www.ni.com/. *www.ni.com/.* [Online] 5 2019. [Dátum: 13. 2 2020.] https://www.ni.com/cs-cz/innovations/white-papers/11/what-is-the-iso-26262-functional-safety-standard-.html.

Polák, Karel. 2003. elektrorevue.cz. [Online] 2003. [Dátum: 26. 2 2020.] http://www.elektrorevue.cz/clanky/03021/index.html.

PortalVS. 2019. PortalVS. *PortalVS.sk.* [Online] 12. 9 2019. [Dátum: 24. 11 2019.] https://www.portalvs.sk/sk/studijny-program/autotronika.

Sommervile, Ian. 2011. *Software Engineering.* s.l. : Pearson Education, Inc., 2011. ISBN-13: 978-0-13-703515-1.

Staron, Miroslaw. 2017. *Automotive Software Architectures.* Gothenburg : Springer International Publishing AG, 2017. s. 237. ISBN 978-3-319-58610-6.

SWATHI, P. 2019. swathiprabhala. [Online] 21. 2 2019. [Dátum: 20. 2 2020.] https://swathiprabhala.blogspot.com/2019/02/embedded-systems.html.

TI. 2019. http://www.ti.com/. *http://www.ti.com/.* [Online] 8 2019. [Dátum: 12. 2 2020.] http://www.ti.com/lit/an/spracn0/spracn0.pdf.

—. 2018. http://www.ti.com/. *http://www.ti.com/.* [Online] 5 2018. [Dátum: 20. 1 2020.] http://www.ti.com/lit/ds/symlink/tms320f28069m.pdf.

—. 2013. ti.com. [Online] 2013. [Dátum: 14. 2 2020.] http://www.ti.com/lit/ug/spruhi3a/spruhi3a.pdf.

—. 2018. TMS320F2806x Piccolo. *ti.com.* [Online] 5 2018. [Dátum: 20. 2 2020.] http://www.ti.com/lit/ds/symlink/tms320f28069m.pdf.

Volný, Petr. 2011. Úvod do embedded systémů. 3 2011.

ČESTNÉ VYHLÁSENIE

Vyhlasujem, že som zadanú diplomovú prácu vypracoval samostatne, pod odborným vedením vedúceho diplomovej práce doc. RNDr. Juraj Pančík, CSc. a používal som len literatúru uvedenú v práci.

Súhlasím so zapožičiavaním diplomovej práce.

V Žiline dňa _____

podpis

Zoznam elektronických príloh

Všetky elektronické prílohy sú uvedené v samostatných priečinkoch.

Elektronická Príloha č.1..... DVD
Elektronická Príloha č.2..... DVD
Elektronická Príloha č.3..... DVD
Elektronická Príloha č.4..... DVD
Elektronická Príloha č.5..... DVD
Elektronická Príloha č.6..... DVD
Elektronická Príloha č.7..... DVD
Elektronická Príloha č.8..... DVD
Elektronická Príloha č.9..... DVD
Elektronická Príloha č10..... DVD
Elektronická Príloha č.11..... DVD
Elektronická Príloha č.12..... DVD
Elektronická Príloha č.13..... DVD
Elektronická Príloha č.14..... DVD
Elektronická Príloha č.15..... DVD
Elektronická Príloha č.16..... DVD
Elektronická Príloha č.17..... DVD