UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Tomaž Lovrenčič

Sistem inteligentne hiše

Diplomska naloga

Maribor, avgust 2009

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO
2000 Maribor, Smetanova ul. 17

Diplomska naloga univerzitetnega študijskega programa

Študent: Tomaž LOVRENČIČ
Študijski program: univerzitetni
Smer: telekomunikacije

Mentor: red. prof. dr. Kačič Zdravko
Komentor: doc. dr. Žgank Andrej

Maribor, avgust 2009

ZAHVALA

Zahvaljujem se mentorju red. prof. dr. Kačič Zdravku za pomoč in vodenje pri opravljanju diplomske naloge. Zahvaljujem se tudi dr. Rotovnik Tomažu za ves posvečen čas in pomoč pri izdelavi diplomske naloge ter doc. dr. Žgank Andreju za nesebično pomoč in svetovanje. Zahvala velja tudi staršem, da so mi omogočili študij, in vsem ostalim, ki so me podpirali.

SISTEM INTELIGENTNE HIŠE

Ključne besede: inteligentna hiša, pametni dom, PIC mikrokrmilnik, sistem varovanja, umetna inteligenca, infrardeči detektor gibanja, LCD prikazovalnik, AT komande

UDK: **681.518.5:004.8(0432)**

Povzetek:

Diplomsko delo predstavi pojem inteligentne hiše, njene prednosti pred klasičnim modelom hiše, opis delovanja in povzetek že znanih protokolov in izdelkov iz kategorije inteligentnih produktov. Nadaljnje je prikazan in izdelan vzorčni primer sistema inteligentne hiše, kjer vsak od šestih modulov opravlja svojevrstno specifično nalogo. Diplomska naloga podrobno opiše vsak kos strojne opreme ter kasneje še primer programa v zbirniku, ki je naložen na mikrokrmilnik in komunicira s to strojno opremo. Interakcija z uporabnikom se vrši neposredno preko LCD zaslona ali na daljavo s pomočjo GSM modula in AT komand. Avtomatiziranost sistema inteligentne hiše zagotavlja samostojno delovanje, s poudarkom na nadzorom na daljavo preko SMS sporočil pa uporabnika realnočasovno obvešča o morebitnih alarmih, ki se pripetijo v času njegove fizične odsotnosti.

INTELLIGENT HOUSE SYSTEM

Keywords: intelligent house, smart home, PIC microcontroller, system security, artificial intelligence, infrared motion detector, LCD display, AT commands

UDK: **681.518.5:004.8(0432)**

Abstract:

In this thesis I focus on the concept of the intelligent house: how does it work and what are its advantages over the classical model of a house? First I give a summary description of existing recognized protocols and products in the category of intelligent systems. Then I present a model example of an intelligent house, where each of 6 modules performs its own specific task. A detailed description of each piece of hardware is included, as well as an example of a program written in an assembler, which is loaded in a microcontroller and which communicates with the hardware. Interaction with the user is performed directly through an LCD display or remotely through a GSM module and AT commands. Automation of the intelligent house enables it to operate on its own, yet there is great emphasis on remote control via real-time SMS messaging, through which the user is informed of any alarms that occur during his or her physical absence.

VSEBINA

Seznam slik

Seznam primerov

Seznam diagramov

Seznam tabel

Uporabljene kratice

PIN	Personal Identification Number
RFID	Radio-Frequency IDentification
LCD	Liquid Crystal Display
GSM	Global System for Mobile communication
PIC	Peripheral Interface Controller
PIR	Passive InfraRed sensor
PSTN	Public Switched Telephone Network
UTP	Unshielded Twisted Pair
STP	Shielded Twisted Pair
IR	InfraRed
RF	Radio Frequency
WLAN	Wireless Local Area Network
ISM	Industrial, Scientific and Medical
UHF	Ultra High Frequency
SRD	Short Range Device
ITU	International Telecommunication Union
BERR	Bit ERror Rate
IEEE	Institute of Electrical and Electronics Engineers
WPAN	Wireless Personal Area Network
CPU	Central Processing Unit
ANSI	American National Standards Institute
OSI	Open Systems Interconnection
EHS	European Home Systems protocol
EIB	European Installation Bus
SMS	Short Message Service
USART	Universal Synchronous/Asynchronous Receiver Transmitter
PWM	Pulse Width Modulation
USB	Universal Serial Bus
CAN	Controller Area Network
EEPROM	Electrically Erasable Programmable Read-Only Memory
CCP	Capture, Compare, PWM
PDIP	Plastic Dual In-line Package
PAL	Programmed Aray Logic
ICD	In-Circuit Debugger
ICSP	In-Circuit Serial Programmer
RAM	Random Access Memory

IDE	Integrated Development Environment
CGRAM	Character Generator Random Access Memory
DDRAM	Display Data Random Access Memory
ASCII	American Standard Code for Information Interchange
PDU	Protocol Data Unit
TTL	Transistor-Transistor Logic
DTE	Data Terminal Equipment
DCE	Data Circuit-terminating Equipment
PC	Personal Computer
EIA	Electronics Industries Alliance
ITU-T	International Telecommunication Union: Telecommunication standardisation sector
ROM	Read-Only Memory
CRC	Cyclic Redundancy Check
LSB	Least Significant Bit
MSB	Most Significant Bit
RISC	Reduced Instruction Set Computer
BASIC	Beginner's All-purpose Symbolic Instruction Code
IC	Integrated Circuit
GND	GrouND
SRAM	Static Random Access Memory
LED	Light-Emitting Diode
PCCP437	PC Code Page 437
PCDN	People’s Content Delivery Network
IRA	International Reference Alphabet
SMSC	Short Message Service Center

1 UVOD

Pojem inteligentne hiše oziroma inteligentnega okolja je poznan že dolgo. Od nekdaj je človek želel razna opravila narediti hitro, učinkovito, zanesljivo in s čim manj truda. Rešitev je bila uporaba avtomatiziranih sistemov, ki bi opravila opravljali sami. Nekoč so avtomatizirani sistemi in roboti veljali za pojem industrijske tehnologije in se jih ni uporabljalo doma, predvsem zaradi cenovne nedosegljivosti, premajhne ponudbe, nekompatibilnosti, fizične velikosti in neustreznosti. Z razvojem sodobne tehnologije zadnjih desetletij, prenosom izkušenj in spoznanj, masovno izdelavo elektronskih komponent in integriranih vezij ter veliko uporabnostjo silicija in ostalih polprevodnikov prihaja do nenehnih pocenitev tovrstnih tehnologij, ki zato postajajo venomer lažje dosegljive tudi domačemu uporabniku. Trg je preplavila množica elektronskih naprav: od preprostega termostata za vodo, ki brez človeške pomoči ohranja vodo toplo in ugasne grelec, ko temperatura doseže kritično mejo, pa vse do kompletnih sistemov za popoln nadzor celotnega doma. S stopnjevanjem zahtev uporabnikov so se tako razvila naprednejša oziroma inteligentna okolja z namenom »inteligentnega« odzivanja na okolje in zagotavljanja maksimalnega udobja domačemu uporabniku. Danes je v svetu kar nekaj ponudnikov sistemov inteligentnih hiš, vendar se to področje še dodatno razvija, tudi na račun novih tehnoloških napredkov (na dotik občutljivi LCD zasloni, IR senzorji, brezžična komunikacija ...).

Cilji diplomske naloge so predstavitev sistema inteligentne hiše, izbira primernih rešitev in načinov za njeno implementacijo, prikaz sorodnih, že obstoječih shem in modelov na tržišču ter končno načrtovanje in izdelava prototipa sistema inteligentne hiše na nivoju mikrokrmilnikov. Namen je izdelati ločene module, ki bi vsak zase opravljali svojo specifično nalogo.

Drugo poglavje tako vsebuje koncept inteligentne hiše, njene prednosti, rešitve implementacije in opis že obstoječih, konkurenčnih izdelkov na tržišču.

Tretje poglavje nas seznani s strojno opremo, potrebno pri implementaciji inteligentne hiše. Pri tem se osredotočamo na lastnosti posameznega kosa strojne opreme, ki so ključnega

pomena za model inteligentne hiše. Dodatno je predstavljeno podpoglavje o programatorju, ki je nujna oprema za programiranje mikrokrmilnikov.

Četrto poglavje opisuje programsko opremo. Najprej predstavimo razvojno okolje na računalniku, nadaljnje so nanizani odseki programske kode v zbirniku za vsak posamezen modul inteligentne hiše. Zraven vsakega primera je kratek opis delovanja zadevnega dela programske kode, za podrobnejše razumevanje pa so dodatno navedeni komentarji ob vsaki vrstici kode. Izdelali in sprogramirali smo naslednje module: *TEMP modul* za nadzor in izvajanje temperaturnih nalog, *POPLAVA modul*, ki alarmira poplavo v kopalnici, *LCD modul* kot uporabniški vmesnik, *PIR modul* s PIR senzorjem za detekcijo vlomilcev in alarmiranje, *SMS* modul s priključenim GSM aparatom za interakcijo uporabnika na daljavo, in *MASTER modul*, ki vrši vmesne funkcije in predstavlja vlogo gospodarja.

V sklepu je podan povzetek opravljenega dela in analiza rezultatov, opisane pa so tudi možne smeri nadaljnjega dela.

2 KONCEPT INTELIGENTNE HIŠE

2.1 Pojem inteligentne hiše, njene lastnosti in prednosti

V današnjem času je zelo opazen trend hišne avtomatizacije, saj so sistemi, ki so trenutno na tržišču, zaradi vsesplošne pocenitve elektronskih komponent in integracije (in s tem tudi velikoserijske proizvodnje) postali dostopni tudi majhnim uporabnikom. Tako je danes sodoben bivanjski prostor z vidika uporabnika predvsem integracija številnih samostojnih sistemov in različnih tehnologij, brez katerih si življenja sploh ne moremo več predstavljati. Korak naprej k takšni ideji je povezava vseh teh ločenih sistemov (npr. sistemov za ogrevanje, klimatizacijo, alarmnih sistemov itd.) v celoto, kjer se le-ti med seboj razumejo in vsak opravlja določeno specifično nalogo. Takšen skupek predstavlja inteligentno okolje, kar pomeni zmožnost »inteligentnega« oziroma »pametnega« odzivanja. »Inteligentni« so torej tisti objekti, ki se ponašajo z določeno stopnjo aktivne inteligence, kar pomeni, da dinamično reagirajo na zaznane parametre ter se primerno odzovejo na podlagi ustreznih algoritmov. Ker je umetno okolje prirejeno za bivanjski prostor (hiša, stanovanje, poslovno poslopje), so tudi elementi tega umetnega okolja usmerjeni h kakovostnemu bivanju v takšni zgradbi ter izpolnjevanju temu primernih nalog. Skupni izraz tega je inteligentna hiša.

Inteligentna hiša tako predstavlja samostojen, zaključen sistem, ki povečuje uporabnikovo udobje, skrbi za naloge, ki bi jih drugače moral opravljati uporabnik sam ali ki jih, zaradi same zahtevnosti opravila ali hendikepiranosti, ta sploh ne bi bil zmožen opraviti. Tako lahko funkcije, kot jih vidi uporabnik, in pripadajoče module oziroma elemente inteligentne hiše razporedimo v naslednje kategorije:

- **funkcija varovanja**: npr. infrardeči senzorji, laserske prepreke, identifikacija ob vstopu s PIN kodo ali RFID kartico, detekcija šuma v varovanem prostoru, vklop sirene, izklop naprav, ki so požarno ali poplavno kritične, elektronska zapora za vrata in okna itd.,

- **funkcija alarmiranja**: npr. senzorji za poplavo, dim, svetlost, padavine, hitrost vetra itd.,
- **funkcije interakcije in komunikacije**: npr. doseg vseh funkcij preko LCD prikazovalnika ali daljinca ali zaslona, občutljivega na dotik, obveščanje uporabnika na daljavo, recimo preko GSM omrežja, interneta, intraneta, daljinsko upravljanje za garažna vrata itd.,
- **funkcije zagotavljanja udobja**: nastavitve ogrevanja ali hlajenja, videofon, električna ključavnica, kontrola garažnih/vhodnih vrat (avtomatsko ali z daljincem), spust in dvig žaluzij, upravljanje notranje in zunanje razsvetljave, možnost ročne regulacije po želji, avtomatsko zalivanje zelenice, kontrola hišnega kina itd.,
- **funkcija spremljanja stanja naprav znotraj inteligentne hiše**: evidenca porabe električne energije, obveščanje o morebitnih okvarah naprav itd.

Splošen model inteligentne hiše prikazuje slika 2.1., pri tem so elementi inteligentne hiše naslednji:

1) interakcija z uporabnikom preko grafičnega zaslona, vklop sistema,
2) elektronska ključavnica (in kontrola pristopa), videodomofon,
3) pasivni infrardeči senzor ali PIR senzor (angl. »Passive InfraRed sensor«),
4) sirena za odganjanje vlomilcev v primeru vloma,
5) senzor poplave v kopalnici,
6) uravnavanje termostata za temperaturo vode,
7) vklop in utripanje zunanjih luči v primeru vloma,
8) obveščanje uporabnika preko GSM ali stacionarnega PSTN omrežja o stanju znotraj hiše,
9) protivlomni senzorji in zaščita na oknih,
10) upravljanje žaluzij/rolet (centralno ali po prostorih),
11) avtomatska vhodna/garažna vrata z daljinsko kontrolo,
12) video nadzor okolice,

13) senzor vetra, zunanje temperature, padavin, svetlosti itd.,
14) avtomatsko zalivanje zelenice,
15) izklop/kontrola električnih in požarno kritičnih porabnikov/elementov v času, ko nas ni doma,
16) upravljanje z multimedijskimi aparati, recimo avdio in video napravami (zabavna elektronika ...).

Slika 2.1 Splošen model inteligentne hiše [1]

Omenjena zasnova je le splošna in ji lahko poljubno spreminjamo kompleksnost, od najpreprostejših senzorjev za dež do avtomatske prepoznave govora ali robotskega pomočnika, ki opravlja hišna opravila. Prav tako je s številom modulov, ki lahko obsega le nekaj med seboj povezanih senzorjev ali pa kompletno zasnovo in popolni nadzor nad celotno bivalno zgradbo s po več sto moduli z zelo specifičnimi nalogami.

Vpeljava inteligence v takšno hišo omogoča vrsto prednosti, med katerimi lahko izpostavimo ekonomske prednosti in prednosti za uporabnika (prilagodljivost uporabniku in njegovemu okolju). Ekonomske prednosti inteligentne hiše se kažejo kot prihranek energije, in s tem kot znižanje stroškov za ogrevanje, hlajenje, osvetlitev prostorov. Tako se visoki začetni investicijski vložek hitro povrne z učinkovito izrabo sredstev (energije) z minimalnimi življenjskimi stroški.

Za stanovalca inteligentne hiše je zelo pomembno tudi varovanje. Z nadzorom hišnih objektov in protivlomnimi zaščitami zagotovimo udobno bivanje ter zaščito lastnine uporabnika sistema. Ker takšni sistemi varovanja/alarmiranja delujejo suvereno in vseh 24 ur na dan, so več kot primerljivi z ostalimi oblikami varovanja, kjer je vključen na primer človeški (torej nepopoln) faktor.

Ključnega pomena v pametni hiši je dobro izgrajena komunikacijska mreža. S tem izolirane naprave povežemo in tvorimo zaključeno celoto. Za prenos podatkov med posameznimi segmenti sistema se najpogosteje uporabljajo naslednji načini:

- **namensko izgrajeno žično omrežje**: samostojna napeljava, uporabljana izključno za namene pošiljanja podatkov med elementi sistema inteligentne hiše. Ponavadi je komunikacijsko vodilo sestavljeno iz UTP kabla (zaradi vsesplošne razširjenosti, primernih karakteristik in cenovne ugodnosti). Alternative so še STP, optični vod, USB, koaksialni kabel ...
- **domače električno omrežje**: z uporabo posebnih modemov je možno doseči komunikacijo po standardnem 220 V električnem omrežju (v evropskem prostoru). Prednost takšnega pristopa je že izgrajena mreža, slabost pa modulacija velikih amplitud el. signalov (šum, odstopanja nosilne frekvence in amplitude),

- **IR komunikacija**: brezžični prenos s pomočjo infrardeče svetlobe (podobno kot pri TV upravljavcih). Načeloma potrebujemo za tak prenos vidno linijo med sprejemno in oddajno stranjo komunikacije. IR komunikacija se uporablja za nadzor naprav znotraj istega prostora. Prednosti IR prenosa so brezžičen način ter poceni IR sprejemne in oddajne diode, slabost pa predvsem majhna simbolna hitrost (klasična nosilna frekvenca je 36 do 40 KHz, bitna hitrost v realnih aplikacijah pa med 100 in 2000 bps) in kratek doseg (odvisno od korekcijskih filtrov, oddajne moči, navadno pod 10 m),
- **RF komunikacija**: prenos podatkov s pomočjo elektromagnetnega valovanja. Nelicenčni ISM pas UHF signala, kjer lahko potekajo podatkovne komunikacije, so na primer: 2450 MHz pas za »modri zob« (angl. »Bluetooth«), 2450 MHz in 5800 MHz pas za WLAN (angl. »Wireless Local Area Network«), 433 MHz za SRD (angl. »Short Range Device«) komunikacijo v regiji 1 (Evropa, Afrika, Bližnji in Srednji Vzhod) po ITU specifikaciji oziroma 915 MHz za regiji 2 (Severna in Južna Amerika, Grenlandija, Pacifik) in 3 (Avstralija, Izrael, Oceanija), 868 MHz za ZigBee. RF komunikacija danes predstavlja napredno tehnologijo in lahko konkurira žičnim povezavam v količini prenosa podatkov, v zadnjem času tudi v BERR. Še vedno pa predstavljata problem višja cena modulov in večja električna poraba (še posebej pri »močnejših« prenosih).

Način komunikacije je večinoma tipa »gospodar-suženj« na vodilu (angl. »bus«) ali popolnoma dvosmerni promet (angl. »full duplex«) pri uporabi zvezdaste ali delno zvezdaste topologije, kjer so posamični moduli povezani tudi med seboj in ne potrebujejo logično centralnega modula – »gospodarja«.

Proizvajalci že obstoječih sistemov inteligentnih hiš so uvedli razne protokole za pošiljanje podatkov z namenom boljše preglednosti in zanesljivosti delovanja teh sistemov. Med bolj znanimi in zanimivimi so naslednji:

- **LonTalk[2]**: komunikacijski protokol za podporo omrežni platformi LonWorks (razvita pri ameriški korporaciji Echelon). LonTalk je bil leta 1999 odobren s strani

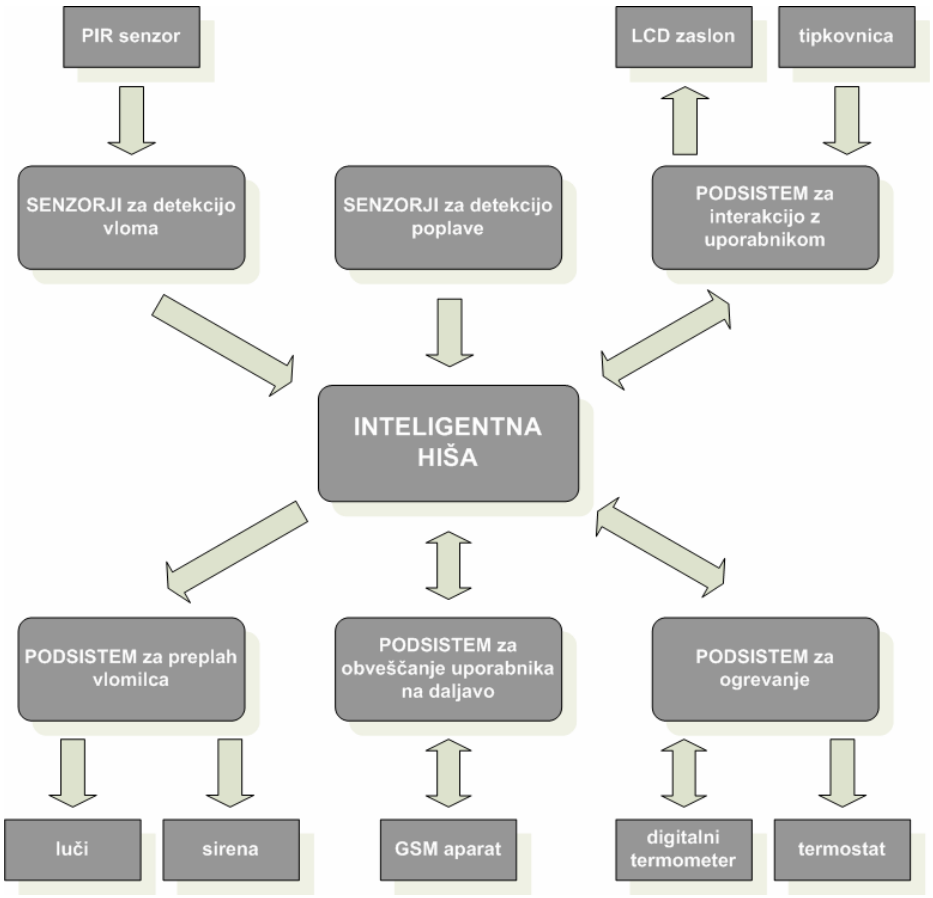
ANSI in sprejet kot omrežni standard za nadzorne aplikacije (angl. »control networking«) *ANSI/CEA-709.1-B*. Prenos je možen po katerem koli mediju (RF, električno omrežje, sukana parica, optično vlakno ...),

- **KNX[3]**: je protokol, baziran na OSI modelu in specializiran posebej za inteligentne zgradbe. Predstavlja združenje in nadaljevanje treh različnih, prej ločenih standardov, in sicer: »European Home Systems Protocol« (EHS), »BatiBUS« in »European Installation Bus« (EIB). Tudi ta pozna širok spekter medijev za prenos (sukana parica, el. omrežje, RF, IR, ethernet). KNX podpira več kot 100 proizvajalcev te opreme,
- **X10[4]**: prvi splošno namenski protokol za komunikacijo med elektronskimi napravami, zasnovan že leta 1975 (Pico Electronics, Škotska). Zaradi dolgotrajnega obstoja na tržišču je dandanes v uporabi na milijone naprav, ki koristijo ta standard,
- **ZigBee[5]**: brezžični (868 MHz in 2.4 GHz pas ISM), cenovno ugoden komunikacijski protokol za naprave majhnih moči izdan s strani ZigBee Alliance. Baziran na IEEE 802.15.4-2003 standardu za WPAN (angl. »Wireless Personal Area Network«) omrežja. V zadnjem času močno razširjen tudi za namene hišne avtomatizacije, predvsem zaradi poudarka na preprostosti, majhne električne porabe, zasedanja licenčno prostih RF pasov, poceni modulov, možnega »mesh« povezovanja (vsak element omrežja se torej lahko ali pa tudi ne poveže z vsakim drugim).

Glede na te protokole so se na tržišču razvile razne ponudbe hišne avtomatizacije in pametnih sistemov. Med bolj znanimi ponudniki inteligentnih hišnih sistemov za domače uporabnike pri nas so: LonWorks, ComfortClick, Marmitek, APH, Dometra, Electres, Conrad …

2.2 Zasnova modela inteligentne hiše z mikrokontrolerjem PIC

V diplomski nalogi smo zasnovali sistem inteligentne hiše s poudarkom na daljinskem nadzoru in alarmiranju uporabnika na mobilni telefon. Sistem nas preko SMS sporočil obvešča o trenutnem stanju v zgradbi in javi morebiten alarm v primeru vloma (sprožijo se tudi luči in sirena za preplah) ali poplave v kopalnici. Dodatno skrbi za ekonomsko varčnost hiše s samostojno kontrolo ogrevanja (vklop in izklop termostata). Shemo sistema inteligentne hiše vidimo na sliki 2.2.



Slika 2.2 Praktična shema sistema inteligentne hiše

Podrobnejše zahteve, ki smo si jih zadali v diplomski nalogi, so naslednje:

- **uporabljen CPU**: mikrokontroler Microchip PIC družina 16F,

- **komunikacija med moduli**: tip komunikacije »gospodar-suženj« z žičnim prenosom podatkov po parici,
- **obveščanje uporabnika na daljavo**: s pomočjo GSM aparata in SMS sporočil,
- **neposredna in enostavna interakcija na LCD zaslonu**: 4 tipke, in sicer »GOR«, »DOL«, »MENI/PREKLIČI« ter »OK/POTRDI« za brskanje po menijih na zaslonu,
- **programski jezik za CPU**: zbirnik,
- **dobra funkcionalnost**,
- **zanesljivo delovanje**: če se sproži alarm je nujno potrebno o tem takoj obvestiti uporabnika, ne glede na potek trenutnega izvajanja programa (različne rutine imajo različno prioriteto),
- **ekonomična zasnova**: izbira strojne opreme s čim manj redundantne periferije in gospodarno določanje lokacij pomnilnika programskim rutinam.

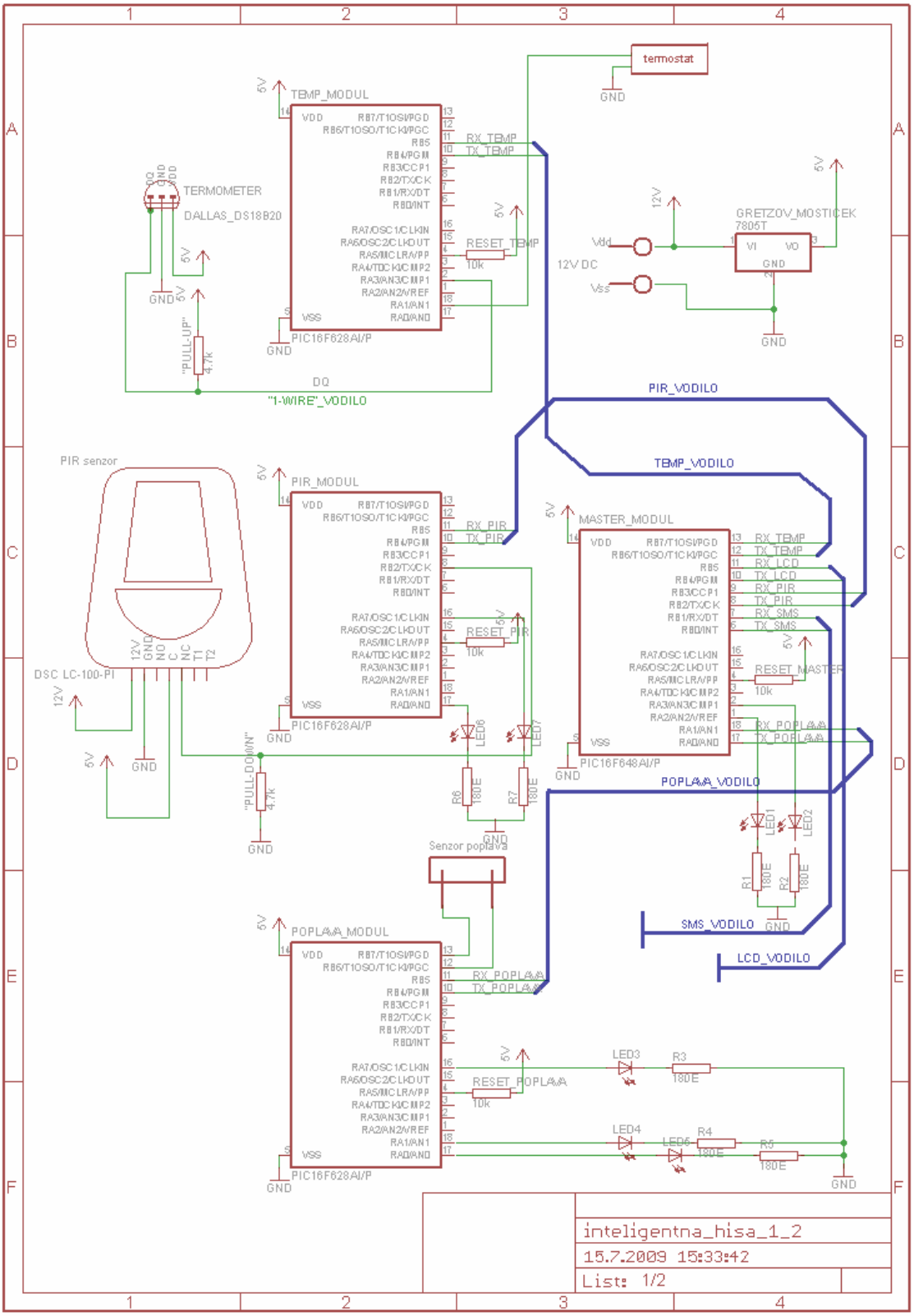
V nadaljevanju (poglavje 3) smo nazorno predstavili vso strojno opremo, ki je bila potrebna za praktično izvedbo sistema inteligentne hiše. Opisane oziroma poudarjene so tiste lastnosti vsakega kosa strojne opreme, ki so poglavitne za namene diplomske naloge. Poglavje 4 se posveti programskemu delu (to je programska koda, ki smo jo naložili na mikrokrmilnike PIC), ki naredi strojno opremo funkcionalno koristno in povezljivo. Rutine, programi in podprogrami vsebujejo tudi komentarje za lažje razumevanje delovanja algoritmov. Pri izbiri strojne opreme in tudi programiranju smo upoštevali podrobnejše zahteve, ki smo si jih zadali in opisali zgoraj.

3 STROJNA OPREMA

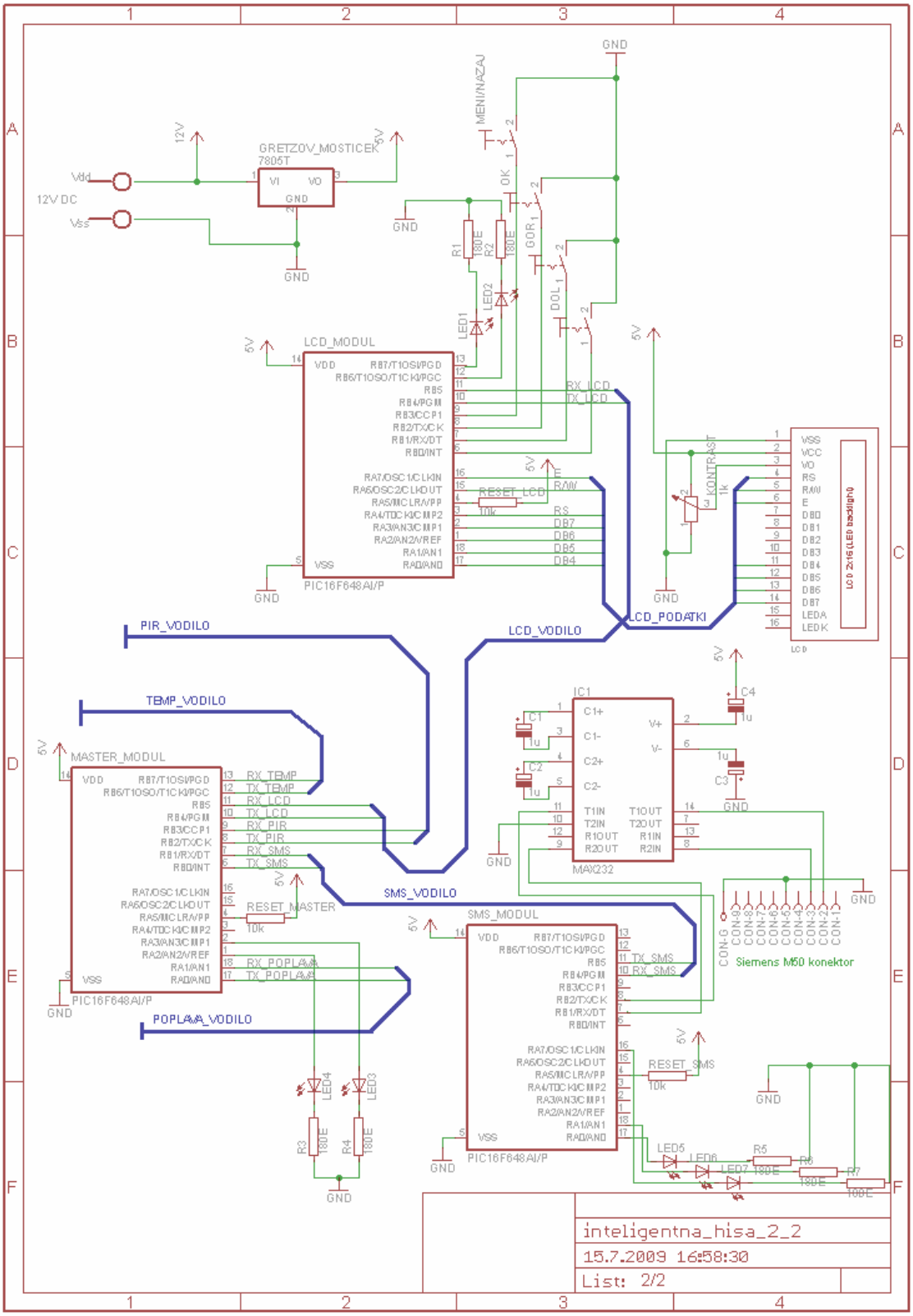
3.1 Shema

Sistem inteligentne hiše smo sestavili modularno, pri čemer vsak od šestih modulov opravlja svojo nalogo. Shema je prikazana na slikah 3.1 in 3.2. Shemo smo narisali v programu Eagle 5.0.0 (razvit pri podjetju CadSoft).

Moduli so povezani centralno na *MASTER modul*, in sicer preko dveh linij, preko katerih serijsko pošiljamo podatke (SER_IN, SER_OUT). Sistem vsebuje temperaturni modul (*TEMP modul*) s funkcijo termostata in termometra, poplavni modul (*POPLAVA modul*) za detekcijo morebitne poplave, modul z zaslonom (*LCD modul*) za neposredno kontrolo sistema preko tipk in zaslona, modul z infrardečim senzorjem (*PIR modul*) za detekcijo gibanja in morebitnega vloma, modul daljinske kontrole (*SMS modul*) za sprejemanje in oddajanje SMS sporočil in glavni modul (*MASTER modul*), ki, kot že omenjeno, povezuje sistem v celoto z »gospodar-suženj« komunikacijo. Pripadajoča tiskanina prototipa sistema inteligentne hiše je v prilogi, kosovnico prikazuje tabela 7.2.



Slika 3.1 Shema inteligentne hiše 1/2



Slika 3.2 Shema inteligentne hiše 2/2

3.2 Mikrokontroler PIC

PIC je družina mikrokontrolerjev proizvajalca Microchip. Med njih uvrščamo veliko število različnih tipov z različnimi zmogljivosti – od najenostavnejših z osmimi nožicami (»pini«), 8-bitnim podatkovnim vodilom ter omejeno podporo raznim vhodno/izhodnim napravam, npr. USART (angl. »Universal Synchronous Asynchronous Receiver Transmitter«), PWM (angl. »Pulse Width Modulation«) in podporo I^2C protokolu; pa vse tja do 100 pinskih 32-bitnih PIC-ev in dsPIC-ev (z integriranim digitalnim signalnim procesorjem) z dobro podporo tudi zahtevnejši strojni opremi, recimo USB, CAN, RF oddajnemu/sprejemnemu vezju itd. Kateri PIC izberemo, je odvisno od danih zahtev aplikacije in kasneje od produkcije izdelka. Če izberemo takšnega s preveč redundantne opreme, s tem zvišujemo stroške samega čipa, tvegamo večjo naložbo v dodatno potrebno opremo, recimo dražji programator, dražja je podprta programska oprema, zahtevnejši programski jezik. Če izberemo takšnega z manj zahtevnimi specifikacijami, pa po drugi strani tvegamo, da nam v času izdelave prototipa ali kasneje pri nadgradnji mikrokrmilnik ne more zagotoviti potrebnih lastnosti.

Za naše namene smo uporabili dva tipa, in sicer PIC16F628A in PIC16F648A. Razlika med njima je edino v količini programskega spomina FLASH in podatkovnega spomina RAM (angl. »Random Access Memory«) in EEPROM (angl. »Electrically Erasable Programmable Read-Only Memory«). Lastnosti obeh lahko strnemo v naslednje značilnosti:

Pomnilnik

- 2048 x 14 FLASH pomnilnika za PIC16F628A in 4096 x 14 za PIC16F648A, pri tem 14 predstavlja velikost naslovnega vodila (angl. »program bus«) v bitih, 2048 oziroma 4096 pa označuje količino naslovnih polj, to je število besed (angl. »word«),
- 224 x 8 RAM podatkovnega prostora (256 x 8 za model 648A),
- 128 x 8 EEPROM podatkovnega prostora (256 x 8 za model 648A).

Oscilator

- Do 20 MHz zunanji oscilator, možnosti so:

- LP kristal male moči (do 200 kHz),
- XT kristal/resonator (do 4 MHz),
- HS kristal/resonator večjih hitrosti (do 20 MHz),
- RC zunanji RC člen,
- INTOSC interni oscilator,
- EC zunanji vhodni takt.

Periferija

- 16 V/I pinov,
- analogni komparator,
- napetostno referenčno vezje,
- 2 komparatorja,
- časovniki (2 x 8-bitni, 1 x 16-bitni),
- CCP modul (angl. »Capture, Compare, PWM«),
- USART.

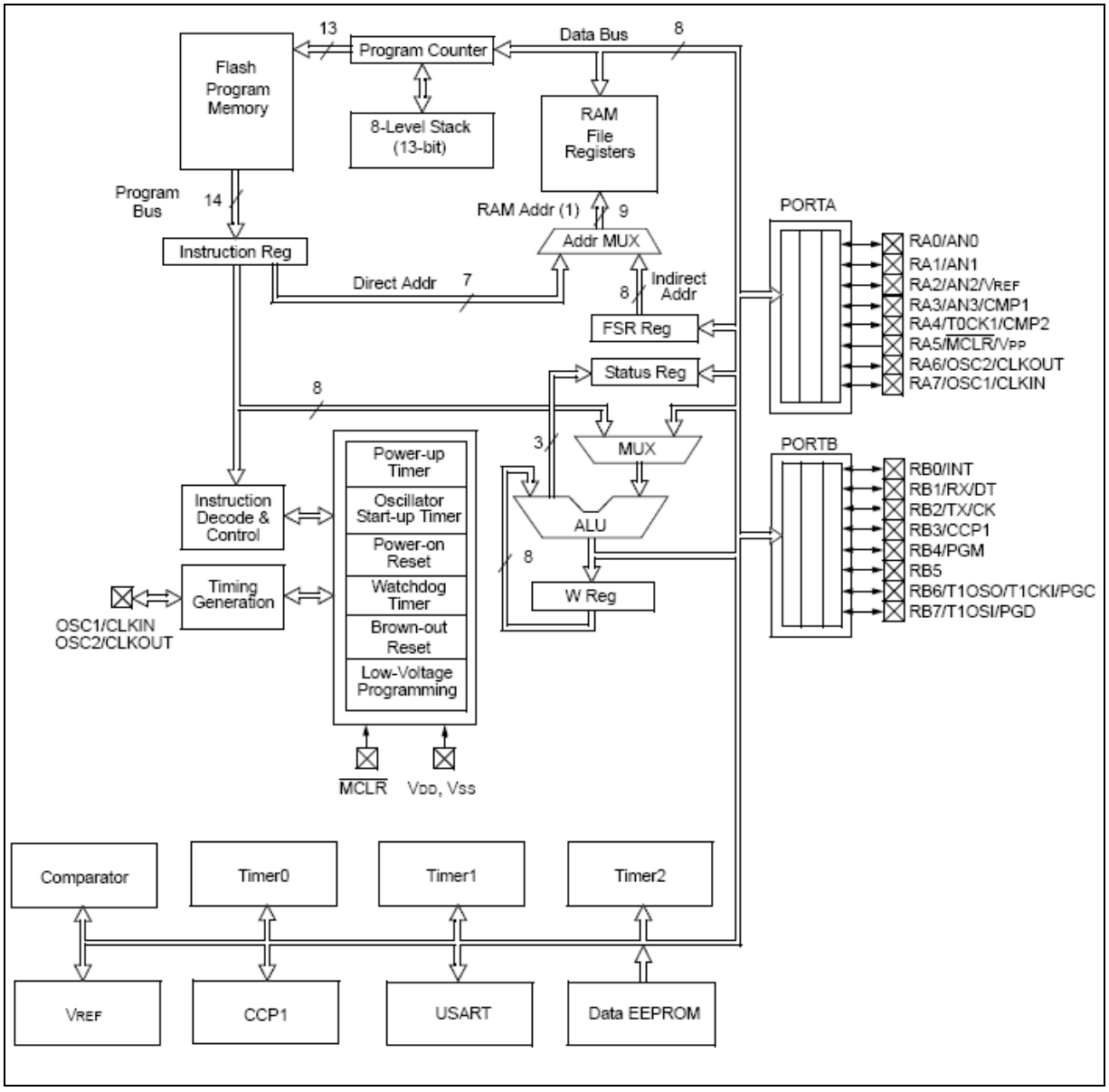
Procesor

- Možnost prekinitev (angl. »interrupt«),
- 8-nivojski sklad,
- 35 asemblerskih inštrukcij,
- direktno, indirektno in relativno naslavljanje,
- 1 inštrukcijski cikel sestavljajo 4 urini takti, pri tem je vsaka inštrukcija sestavljena iz enega cikla, razen tistih, ki vejijo kodo, npr. GOTO, CALL, RETURN.

Ohišje

- 18-pinski PDIP (angl. »Plastic Dual In-line Package«)

Celotna zgradba je razvidna na sliki 3.3.

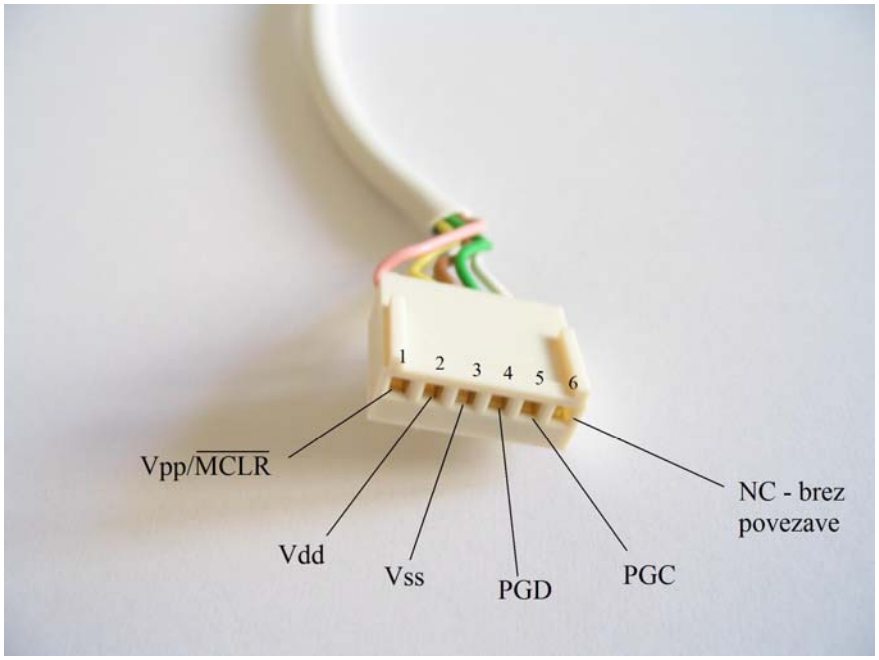


Slika 3.3 Blokovni diagram mikrokrmilnika PIC16F627A/628A/648A

3.3 Programator

Programator je vezje, namenjeno programiranju mikrokrmilnikov (isto ime se uporablja tudi za napravo, ki programira ostale oblike programirljivih vezij, npr. EEPROM-e, PAL-e, procesorje itd.). Uporabili smo razvojno orodje MPLAB ICD 2, razvito pod okriljem Microchipa. Predstavlja poceni izvedbo serijskega programatorja v vezju (angl. »In-Circuit Serial Programmer« ali ICSP). Obenem nudi tudi možnost razhroščevalnika (angl. »In-Circuit Debugger«). Podpira mikrokrmilnike serij PIC10F, PIC12F, PIC16F, PIC18 in dcPIC30F. Za možnost razhroščevanja moramo seveda uporabiti PIC-a, ki to možnost omogoča, pri tem pa se za ta namen uporabi del pomnilnika, torej FLASH-a, RAM-a ter 0 ali 1 nivo sklada.

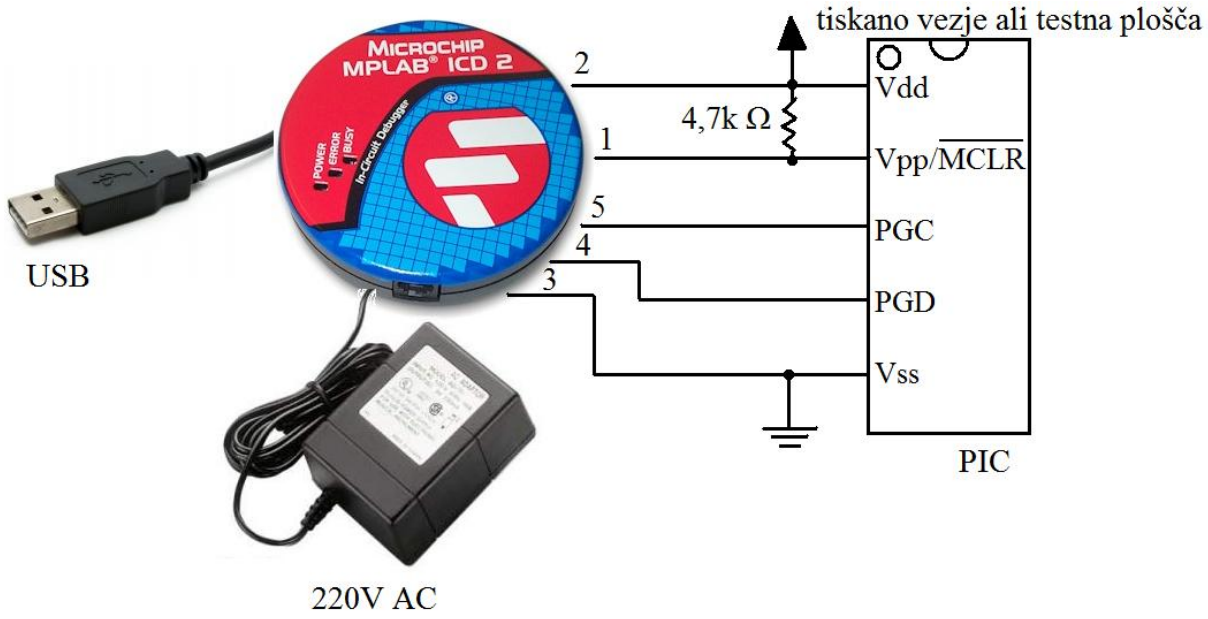
Programator in mikrokrmilnik med seboj povežemo s serijskim konektorjem, prikazanem na sliki 3.4.



Slika 3.4 Serijski konektor za programator

Za povezavo potrebujemo najmanj tri linije: Vpp/MCLR (angl. »programming voltage input/master clear«), PGC (angl. »programming clock«) in PGD (angl. »programming data«). Vpp/MCLR pripravi krmilnik na sprejem podatkov za preprogramiranje čipa (s tem da dovedemo približno 13 Vpp na to nožico) ter po končanem programiranju držimo

mikrokrmilnik v resetu (s povezavo pina na mikrokrmilniku na logično nizek nivo) za namen razhroščevanja. PGC opravlja vlogo urinega takta, da mikrokrmilnik ve, kdaj lahko zajame podatke, ki so na podatkovni liniji PGD. Opcijsko lahko priključimo še pina 2 (Vdd) in 3 (Vss), vendar se je potrebno zavedati, da lahko tako preko USB vodila (od koder se napaja tudi sam programator) pridobimo le omejeno količino moči, za večje zahteve moramo dodatno priključiti eksterni vir napetosti. Primer izvedbe je prikazan na sliki 3.3.



Slika 3.5 Povezava mikrokrmilnika in programatorja

Algoritem za razhroščevanje in programiranje, izbiro pravih knjižnic (.LKR datoteke), določitev datoteke z mnemoniki (.INC datoteke) ter kreiranje strojne kode (.HEX datoteke) izvedemo v MPLAB IDE okolju (uporabili smo MPLAB verzija 7.40). Za izbiro razhroščevalnika izberemo Debugger>Select Tool>MPLAB ICD 2, samo za programator izberemo Programmer>Select Programmer>MPLAB ICD 2 in nato Programmer>Connect, da vzpostavimo zvezo z ICD 2[6].

3.4 LCD s Hitachi HD44780 gonilnikom

Alfanumeričen zaslon s tekočimi kristali (angl. »LCD, Liquid Crystal Display«) na osnovi HD44780 je danes *de facto* industrijski standard za interakcijo z vgrajenimi sistemi (angl. »embedded systems«). Zasloni imajo tipične »velikosti« 1 × 8, 2 × 8, 4 × 20, 1 × 16 …

Upravljanje LCD-ja poteka preko 8- ali 4-bitnega vodila za podatke, ter 3-bitnega za signalizacijo.

Slika 3.6 LCD zaslon 2 x 16 znakov

HD44780 nam nudi podporo industrijskemu standardu za nadzor karakternih LCD modulov. Nožice nad zaslonom (glej slika 3.6) imajo naslednjo vlogo:

Tabela 3.1 LCD pini

Pin	Simbol	Vhod/Izhod	Funkcija, opis
1	Vss	-	Potencial 0V (referenca zemlje)
2	Vcc	-	Potencial +5 V
3	Vee	-	Nastavitev kontrasta
4	RS	V	0 = vnos inštrukcije 1 = vnos podatkov
5	R/W	V	0 = pišemo na LCD 1 = beremo iz LCD-ja

6	E	V	0 = omogočimo komunikacijo z LCD-jem 1 = onemogočimo komunikacijo z LCD-jem
7	DB0	V/I	Podatkovna linija bit 0, najnižji bit (angl. »LSB – Least Significant Bit«)
8	DB1	V/I	Podatkovna linija bit 1
9	DB2	V/I	Podatkovna linija bit 2
10	DB3	V/I	Podatkovna linija bit 3
11	DB4	V/I	Podatkovna linija bit 4
12	DB5	V/I	Podatkovna linija bit 5
13	DB6	V/I	Podatkovna linija bit 6
14	DB7	V/I	Podatkovna linija bit 7, najvišji bit (angl. »MSB – Most Significant Bit«)
17	Anoda	-	Pozitivni priključek za vgrajeno LED za osvetlitev (+5 V)
18	Katoda	-	Negativni priključek za vgrajeno LED za osvetlitev (0 V)

S kombinacijo različnih logičnih nivojev na omenjenih pinih lahko dosežemo vse potrebne operacije nad LCD-jem. Podrobneje nabor inštrukcij prikazuje tabela 3.2.

Tabela 3.2 LCD nabor inštrukcij[7]

E	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Opis
Počisti zaslon:											
1	0	0	0	0	0	0	0	0	0	1	Zbrišemo vse na zaslonu, kurzor se premakne na začetno pozicijo (naslov 0).
Kurzor na začetek:											
1	0	0	0	0	0	0	0	0	1	*	Vrnitev kurzorja na začetno pozicijo (naslov 0).
Nastavitev načina vnosa:											
1	0	0	0	0	0	0	0	1	I/D	S	*I/D* določimo smer premikanja kurzorja: 0 – levo 1 – desno *S* določimo premik zaslona: 0 – zaslon se ne zamakne (angl. »shift«) 1 – zaslon se zamakne
Nastavitev prikaza zaslona in kurzorja:											

<table>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>D</td><td>C</td><td>B</td><td>D vklop/izklop zaslona:
0 – izklop zaslona
1 – vklop zaslona
C vklop/izklop kurzorja:
0 – izklop kurzorja
1 – vklop kurzorja
B utripanje kurzorja:
0 – ne utripa
1 - utripa</td></tr>
<tr><td colspan="12">Zamikanje zaslona in kurzorja:</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>S/C</td><td>R/L</td><td>*</td><td>*</td><td>S/C določimo, ali naj se ob prehodu na naslednji znak zamakne zaslon ali kurzor:
0 – premakni kurzor
1 – premakni zaslon
R/L določimo smer zamika:
0 – levo
1 - desno</td></tr>
<tr><td colspan="12">Nastavitev podatkovnega vodila, števila vrstic, nabora znakov:</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>DL</td><td>N</td><td>F</td><td>*</td><td>*</td><td>DL določimo širino vodila:
0 – 4-bitni vmesnik
1 – 8-bitni vmesnik
N število vrstic:
0 – 1 vrstica
1 – 2 vrstici
F pisava:
0 – 5 × 7 pik/znak
1 – 5 × 10 pik/znak</td></tr>
<tr><td colspan="12">Nastavitev CGRAM naslova:</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td colspan="6">CGRAM naslov (angl. »Character Generator RAM«)</td><td>Naslov, kamor želimo shraniti svoj poljubno definiran znak. Možnih je 8 znakov velikosti 5 × 7 pik. Za tem ukazom sledi kodni zapis uporabniškega znaka.</td></tr>
<tr><td colspan="12">Nastavitev DDRAM naslova:</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td colspan="7">DDRAM naslov (angl. »Display Data RAM«)</td><td>Naslov oziroma pozicija na LCD zaslonu, kjer želimo prikazati znak (prva vrstica od naslova 0x00 do 0x0F, druga od 0x40 do 0x4F). Za tem ukazom sledi kodni zapis znaka.</td></tr>
<tr><td colspan="12">Branje zastavice zasedenosti in števca naslova:</td></tr>
</table>

1	0	1	BF	CGRAM ali DDRAM naslov	*BF* zastavice zasedenosti: 0 – modul ni zaseden in lahko sprejema podatke 1 – modul izvaja interno operacijo
Piši v CGRAM ali DDRAM:					
1	1	0	Vpiši 1 bajt podatkov		Vpišemo podatke v CGRAM oziroma DDRAM..
Beri iz CGRAM-a ali DDRAM-a:					
1	1	1	Preberi 1 bajt podatkov		Preberemo podatke iz CGRAM-a oziroma DDRAM-a.

Za namene naše diplomske naloge smo uporabili 4-bitno širino vodila, tako da smo vsak bajt podatkov poslali v dveh polbajtih (angl. »nibble«). S tem smo »privarčevali« 4 nožice na samem mikrokontrolerju (LCD_modul) na račun daljše časovne zakasnitve, ki je minimalna. Na LCD smo prikazovali standardne ASCII znake s strojnim USART modulom na PIC-u, torej preko TX in RX pinov. Prikaz je bil pričakovan, le pri znaku za stopinjo (°) smo uporabili kodni zapis iz nabora znakov za LCD-je (binarno: 11011111), ki se razlikuje od tistega v ASCII tabeli (binarno: 10100111).

3.5 GSM modem s podporo AT komand

Uporabnik lahko sistem inteligentne hiše upravlja tudi daljinsko, z GSM aparatom. V ta namen smo uporabili Siemensov mobilni telefon tipa M50. Pri iskanju primernega telefona smo se osredotočili na tiste, ki so imeli vgrajen vmesnik za komunikacijo z zunanjim svetom ter možnost upravljanja z SMS sporočili (kratka tekstovna sporočila, angl. »Short Message Service«). Najprimernejša se nam je zdela uporaba modema s podporo AT komand. Za sistem inteligentne hiše je tako mogoče uporabiti katerikoli modem, da le tolmači AT komande po GSM standardu 07.05 (vsebuje specifikacije za SMS). Uporabljene AT ukaze prikazuje tabela 3.3.

Tabela 3.3 AT komande[8]

Funkcija	Ukaz[1]	Opis
Testna komanda	**AT**	***AT*** (angl. »ATtention«) – pozor. S tem ukazom preverimo, če se določena naprava/modem odziva na nabor AT komand.
Odmev komande	KOMANDA TEST: / KOMANDA BERI: / KOMANDA VPIŠI: **ATE*vrednost***	***vrednost*** 0 – odmev (angl. »echo«) izključen 1 – odmev vključen
Oblika SMS sporočila	KOMANDA TEST: **AT+CMGF=?** KOMANDA BERI: **AT+CMGF?** KOMANDA VPIŠI: **AT+CMGF=*nacin***	***nacin*** 0 – PDU oblika sporočila 1 – tekstovna oblika sporočila
Pošlji SMS sporočilo	KOMANDA TEST: **AT+CMGS=?** KOMANDA BERI: / KOMANDA VPIŠI: PDU način: **AT+CMGS=*dolzina*<CR>** ***PDU*<ctrl-Z>**	***dolzina*** dolžina PDU-ja **<*CR*>** angl. »Carriage Return«: ASCII kontrolni znak za vrnitev kurzorja na prvo mesto v vrstici ***PDU*** angl. »Protocol Description Unit«: protokolno-opisna enota; podatkovni paket, ki vsebuje podatke o telefonski številki, servisnem centru za pošiljanje SMS-ov, dolžini številke, časovni žig, uporabniške podatke. PDU niz vsebuje heksadecimalne oktete in decimalne semioktete.
Izpiši SMS sporočila iz spomina	KOMANDA TEST: **AT+CMGL=?** KOMANDA BERI: **AT+CMGL[=*stat*]** KOMANDA VPIŠI: /	***stat*** 0 sprejeta neprebrana sporočila (privzeto) 1 sprejeta prebrana sporočila 2 shranjena neprebrana sporočila 3 shranjena poslana sporočila 4 vsa sporočila

[1] KOMANDA TEST predstavlja ukaz, ki preveri možne parametre za dani ukaz.
KOMANDA BERI predstavlja ukaz, ki prebere trenutno nastavljene parametre ali druge podatke.
KOMANDA VPIŠI predstavlja ukaz, ki spremeni dani parameter.

Preberi posamezno SMS sporočilo iz spomina	KOMANDA TEST: **AT+CMGR=?** KOMANDA BERI: **AT+CMGR=*indeks*** KOMANDA VPIŠI: /	***indeks*** iz spomina preberemo SMS sporočilo, ki se nahaja na tej indeksni številki
Vpiši SMS sporočilo v spomin	KOMANDA TEST: **AT+CMGW=?** KOMANDA BERI: / KOMANDA VPIŠI: PDU način: **AT+CMGW=*dolzina[,stat]* <CR>** ***PDU*<ctrl-Z>**	***dolzina*** dolžina PDU-ja ***stat*** določa kakšen status naj ima sporočilo, ki ga shranjujemo 0 sprejeta neprebrana sporočila (privzeto) 1 sprejeta prebrana sporočila 2 shranjena neprebrana sporočila 3 shranjena poslana sporočila 4 vsa sporočila
Pošlji prednastavljeno SMS sporočilo iz spomina	KOMANDA TEST: **AT+CMSS=?** KOMANDA BERI: / KOMANDA VPIŠI: **AT+CMSS=*indeks*[da, toda]**	***indeks*** pošljemo SMS sporočilo, ki se nahaja na tej indeksni številki v spominu ***da*** telefonska številka kamor pošiljamo (v obliki niza) ***toda*** format telefonske številke
Zbriši SMS sporočilo iz spomina	KOMANDA TEST: **AT+CMGD=?** KOMANDA BERI: / KOMANDA VPIŠI: **AT+CMGD=*indeks***	***indeks*** iz spomina zbrišemo SMS sporočilo, ki se nahaja na tej indeksni številki.

Primer komunikacije med modemom in osebnim računalnikom prikazuje primer 3.1. Računalnik ima prednaložen OS Windows XP Pro, komunicira preko vgrajenega terminalskega okna (angl. »HyperTerminal«), ki ga najdemo v Start> Programi> Pripomočki> Communications> HyperTerminal. Uporabili smo serijski kabel (D-sub 9), simbolna hitrost 9600 baud/sekundo, 8 podatkovnih bitov, brez paritete, 1 »stop« bit ter brez kontrole pretoka. S krepko pisavo smo označili vnos uporabnika na PC-ju, ostalo je odziv modema.

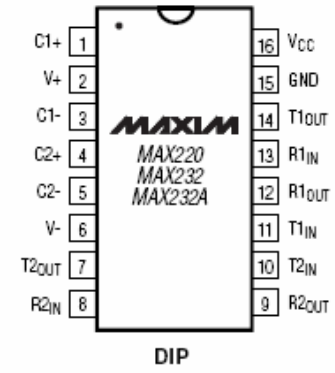
Primer 3.1 Komunikacija preko AT komand

```
AT
AT
OK
ATE0
ATE0
OK
AT+CMGF?
+CMGF: 0
OK
AT+CMGL=4
+CMGL: 1,3,,23
07918346401400F011380B818346507696F20000FF0A417658DE064193D210
OK
AT+CMGS=27
>
07918346401400F011000B818346507696F20000FF0E417658DE06C1DF7076D81E0E01
+CMGS: 59

OK
```

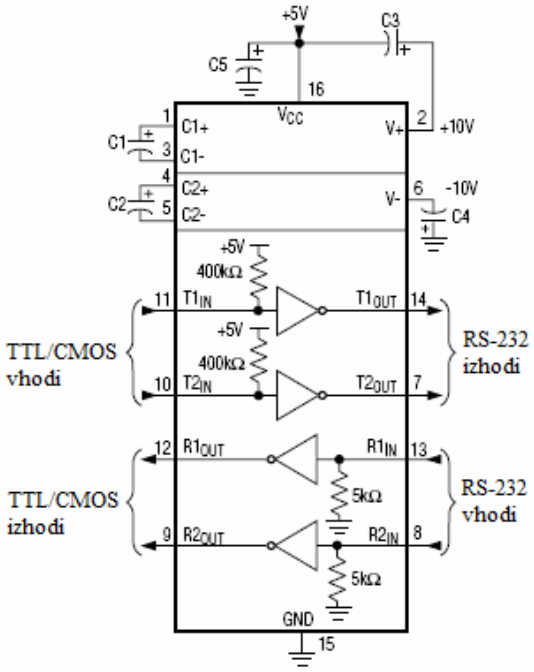
3.6 MAX232

MAX232 je linijski gonilnik/sprejemnik za EIA-232E signale (včasih standard RS-232). Čip se napaja iz enojne +5 V napajalne linije, za vhod v sprejemnik uporablja TTL nivoje (+5 V in 0 V), izhod pa je balansirane napetosti, tipično ±8 V (minimalno ±5 V).

Čeprav se nam pri površnem pogledu ta del vezja zdi nesmiseln (pretvorba napetostnih nivojev naprej iz TTL na RS-232, nato spet, v samem GSM aparatu, nazaj na TTL +3.3 V), smo to naredili na račun večje kompatibilnosti. Sistem inteligentne hiše je tako moč povezati z vsakim GSM aparatom ter njegovim pripadajočim originalnim podatkovnim kablom (za povezavo na *D-sub 9* konektor), pogoj je le delovanje aparata kot modem, torej komunikacija preko AT komand. Notranjo zgradbo MAX232 vezja prikazuje slika 3.7.



Kapacitivnost (µF)					
TIP	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



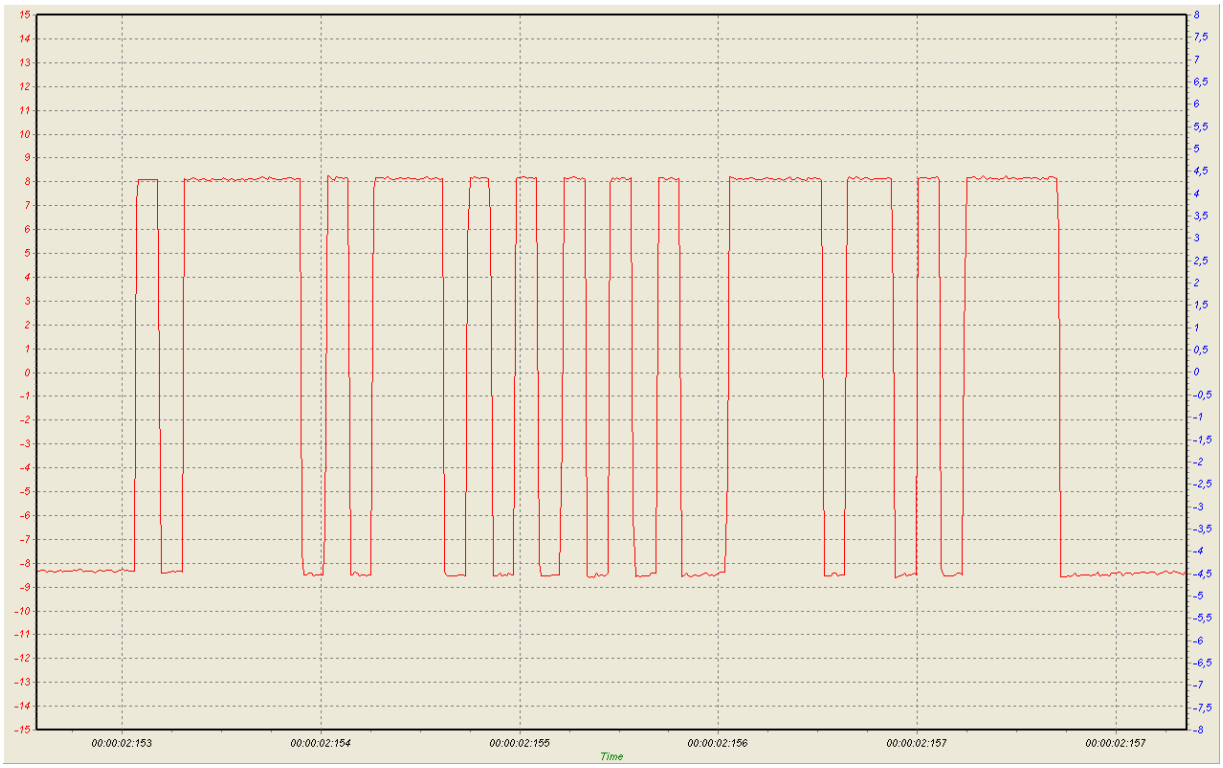
Slika 3.7 MAX232: notranja zgradba[9]

Standard RS-232[2] definira serijsko komunikacijo na fizičnem nivoju med DTE (angl. »Data Terminal Equipment«) in DCE (angl. »Data Circuit-terminating Equipment«). Pri tem vlogo DTE predstavlja mikrokrmilnik ali PC, DCE pa modem.

Logične enice in ničle predstavljajo napetostni nivoji, ki zavzemajo vrednosti med ±3 V in ±15 V. Logična enica (stanje tudi poimenovano kot »OFF«) zavzema negativno napetost, ničla (oziroma »ON«) pa pozitivno. Splošna shema podatkovnega paketa sestoji iz »START« bita, 5 do 8 podatkovnih bitov ter 1, 1.5 ali 2 »STOP« bita. Podatkovni (uporabniški) biti se pošiljajo LSB najprej. Za namene diplomske naloge smo rabili naslednjo obliko RS-232 komunikacije:

- 1 »START« bit,
- 1 bajt uporabniških podatkov,
- 1 »STOP« bit.

[2] Standard je predstavila EIA (angl. »Electronics Industries Alliance«), kasneje ga je v podobni obliki standardiziral še sektor za standardizacije v telekomunikacijah ITU-T (angl. »Telecommunication Standardization Sector«).



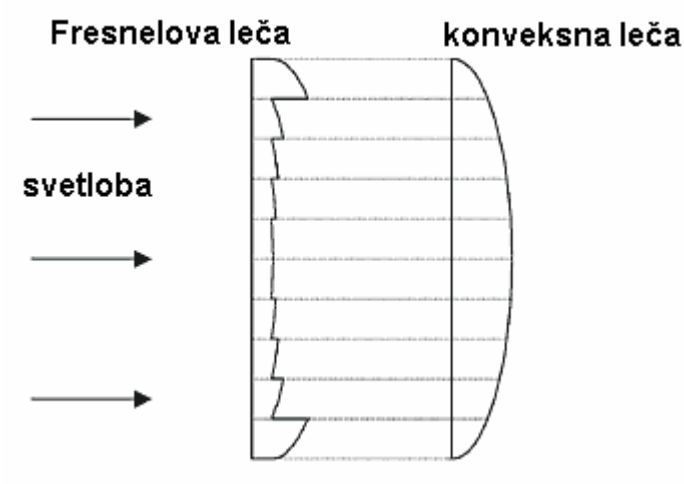
Slika 3.8 RS-232 podatkovni paket ('A', 'T', CR in LF)

3.7 PIR senzor

Pasivni infrardeči senzor (nadaljnje PIR senzor) je elektronska naprava, ki meri IR radiacijo predmetov znotraj svoje dosega. Gibanje se zazna, ko vir IR svetlobe z eno temperaturo, recimo človek, preide območje pred drugim virom IR svetlobe z drugačno temperaturo, recimo zidom. Vsi predmeti imajo, tako rečeno, črno sevanje telesa, ki je odvisno od temperature predmeta. Pasivni v tem primeru pomeni, da senzor ne oddaja lastnega IR snopa, temveč le pasivno sprejema dohodno IR sevanje[10].

PIR senzor, ki smo ga uporabili za naš sistem inteligentne hiše, se nahaja v ohišju, ki poleg elektronike (tiskano vezje, senzor) vsebuje še plastično steklo v obliki majhnih Fresnelovih leč, ki fokusirajo širše območje detekcije na sam senzor. Fresnelova leča je optični ekvivalent (poti optičnih žarkov so teoretično enake) konveksni leči, le da je tanjša. S

tem se zaradi same debeline materiala pridobi na manjši absorpciji žarkov. Koristi so s tem tudi ekonomske (manjša količina materiala, in s tem nižja cena).



Slika 3.9 Fresnelova leča

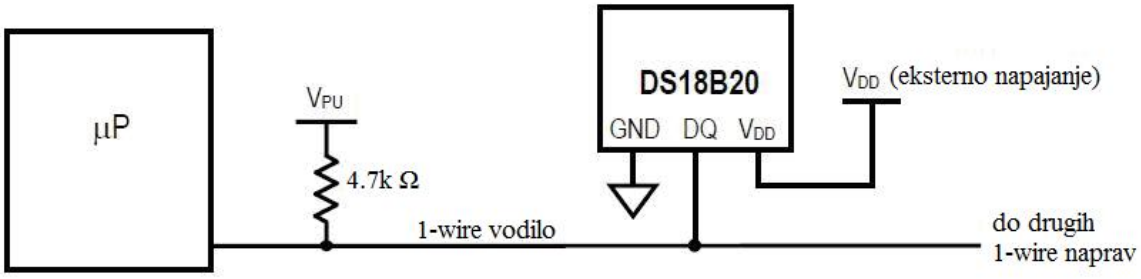
Možno je nastaviti senzibilnost samega senzorja, če na primer v območju delovanja PIR-a dovoljujemo gibanje domačih živali (do določene velikosti oziroma teže).

Z mikrokrmilnikom smo senzor povezali na način NC (angl. »Normally-Closed«). Vgrajen rele nam, ko je aktiviran, izključi priklopljeno vezje in obratno. Na izhodu senzorja tako odčitavamo logično visok nivo (+5 V) v stanju neaktivnosti in logično nizek nivo (0 V) v primeru detekcije predmeta. Uporabljen senzor je upodobljen na sliki 3.10.

Slika 3.10 DSC PIR senzor[11]

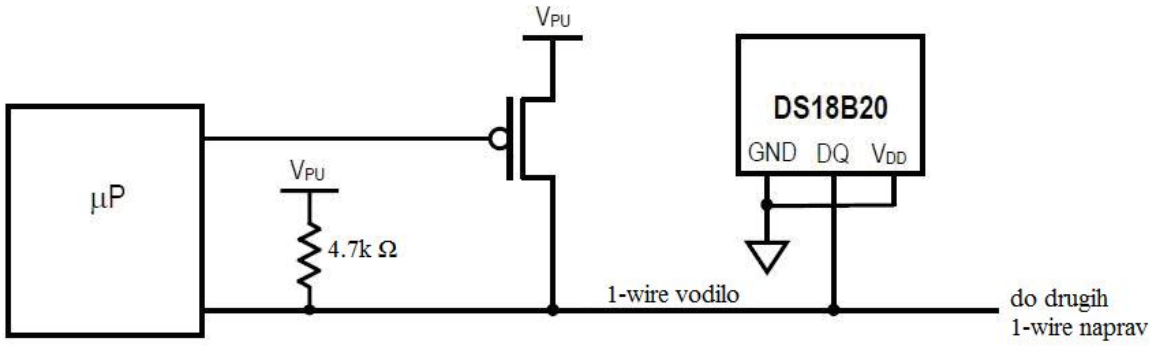
3.8 Digitalni termometer

Za merjenje temperature smo uporabili digitalni termometer DS18B20 [12] podjetja Maxim[3]. DS18B20 je digitalni termometer z možnostjo nastavljanja resolucije odčitavanja (9- do 12-bitna). Za komunikacijo se uporablja napredna »1-wire« komunikacija. Komunicira se preko ene samcate linije, oznaka DQ. Naprava se nahaja v TO-92 ohišju (3 nožice), napajanje je lahko izvedeno eksterno (enosmerni tok, napetost +3 do +5.5 V) ali parazitno (direktno iz DQ linije). Oba načina sta prikazana na sliki 3.11 in sliki 3.12.



Slika 3.11 DS18B20: eksterno napajanje

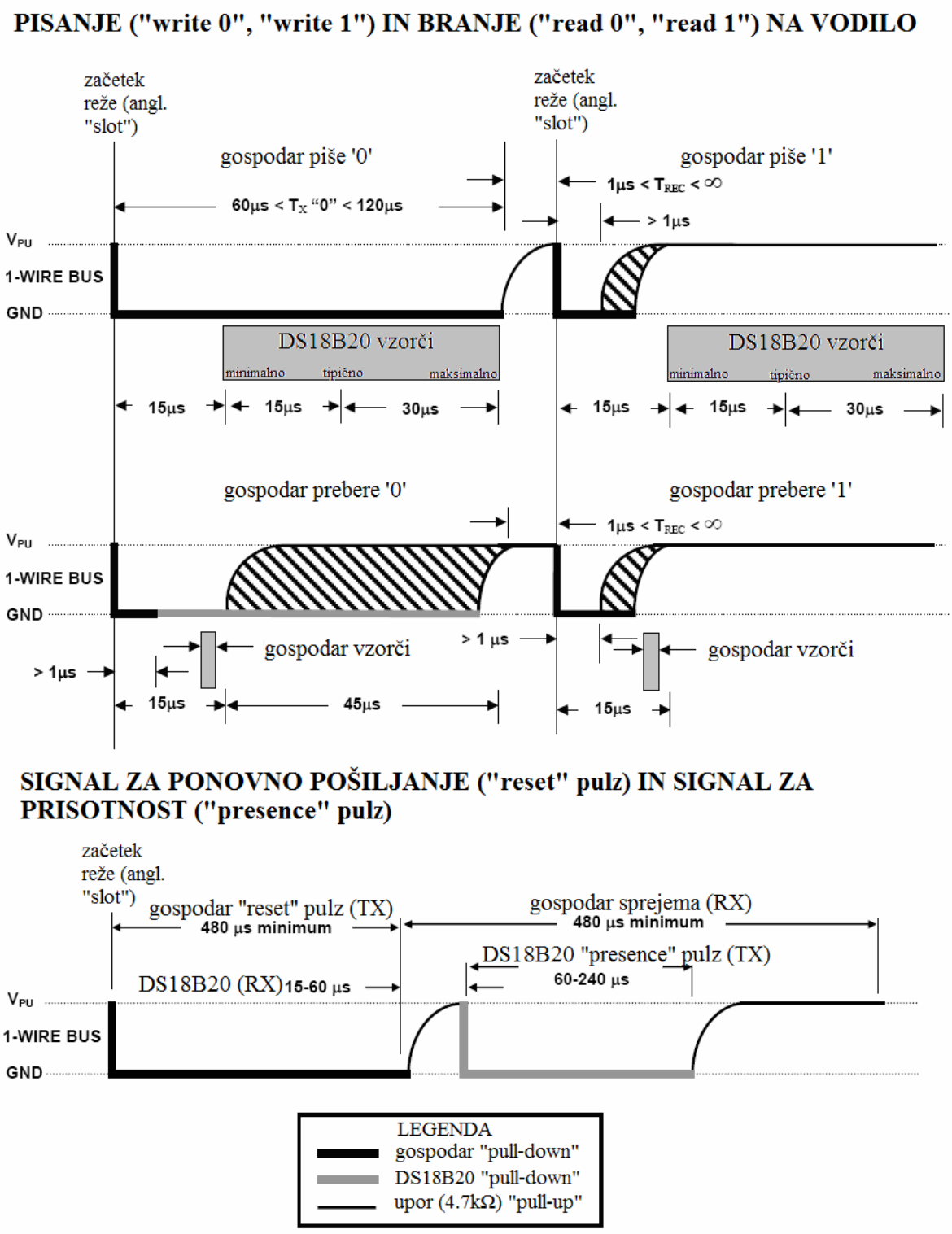
[3] http://www.maxim-ic.com



Slika 3.12 DS18B20: parazitno napajanje

Na podatkovni liniji DQ (»1-wire vodilo«) je (teoretično) možnih 2^{64} naprav, saj vsebuje vsaka od naprav unikatno 64-bitno lasersko vžgano kodo v ROM-u (ID naprave). Protokol definira 6 tipov signala na fizični ravni, in sicer:

- »reset« pulz (signal za ponovno pošiljanje),
- »presence« pulz (signal za prisotnost),
- »write 0« (pisanje bita 0),
- »write 1« (pisanje bita 1),
- »read 0« (branje bita 0) in
- »read 1« (branje bita 1).



Slika 3.13 1-wire komunikacija

Iz omenjenih signalov tvorimo bajt z najmanj obteženim bitom najprej, ki jih pošljemo na vodilo. Protokol nadaljnje definira 5 komand za naslavljanje naprav na vodilu, in sicer:

- **SEARCH ROM [0xF0];** iskanje vseh dosegljivih »1-wire« naprav in včitavanje njihovih ID naprav,
- **READ ROM [0x33];** mikrokontroler lahko brez uporabe SEARCH ROM ukaza prebere ID naprave. Pogoj je, da imamo na vodilu le eno napravo,
- **MATCH ROM [0x55];** po izvršitvi tega ukaza sledi 64-bitna koda ID naprave. Na ukaz se bo odzvala le naprava, katere ID ustreza njenemu lastnemu, medtem bodo vse ostale naprave čakale na »reset« pulz,
- **SKIP ROM [0xCC];** s tem ukazom mikrokontroler določi vse »1-wire« naprave na vodilu istočasno. Primer uporabe: vsi termometri naj izmerijo temperaturo,
- **ALARM SEARCH [0xEC];** odzovejo se naprave, ki imajo setirano zastavico za alarm, torej je temperatura previsoka oziroma prenizka.

Možnih je 6 funkcij za upravljanje z »1-wire« napravami:

- **CONVERT T [0x44];** naprava izmeri temperaturo (enkrat) in jo shrani v temperaturni register (velikosti dveh bajtov). Mikrokontroler lahko nato izvrši funkcijo branja,
- **WRITE SCRATCHPAD [0x4E];** pisanje v pomnilnik »1-wire« naprave. Vpišemo lahko 3 bajte podatkov: T_L – alarm termostata za prenizko temperaturo, T_H – alarm termostata za previsoko temperaturo ter prepis konfiguracijskega registra,
- **READ SCRATCHPAD [0xBE**]; preberemo 9 bajtov iz pomnilnika, najmanj obtežen bit prvega bajta najprej. Mikrokontroler lahko prekine sprejemanje vseh 9 bajtov, če pošlje »reset« pulz,
- **COPY SCRATCHPAD [0x48];** kopira T_L, T_H in konfiguracijski register iz pomnilnika v EEPROM. S tem se izognemo ponovnemu konfiguriranju v primeru izpada napetosti na termometru, ponovnemu vklopu itd.

- **RECALL E^2 [0xB8];** kopira T_L, T_H in konfiguracijski register iz EEPROM-a v pomnilnik naprave,
- **READ POWER SUPPLY [0xB4**]; mikrokontroler lahko s tem ukazom preveri, kako se napaja določena naprava. Parazitno napajan termometer bo v času sekvence branja vodilo postavil na logično nizek nivo, eksterno napajan pa na logično visok.

Temperaturno območje je med +125 °C ($00000111\ 11010000_2$) ter -55 °C ($11111100\ 10010000_2$). Največji čas konverzije znaša 750 ms za 12-bitno resolucijo, kar je treba upoštevati tudi pri odčitavanju temperature z mikrokontrolerjem. DS18B20 ima tudi možnost nastavljanja alarma nizke ter alarma visoke temperature (deluje kot termostat). Primer uporabe termometra DS18B20 in mikrokontrolerja PIC prikazuje tabela 3.4.

Tabela 3.4 Odčitavanje temperature, 1 senzor DS18B20 na vodilu (podatki si sledijo v časovnem sosledju)

Signal	Vhod/izhod glede na PIC	Opis
»reset«	I	PIC izvrši »reset« pulz, inicializacija novega pogovora.
»presence«	V	DS18B20 se odzove.
0xCC	I	PIC ukaže preskok iskanja naprave z določeno unikatno ROM kodo.
0x4E	I	PIC ukaže pisanje v pomnilnik (angl. »scratchpad«)
3 bajti (T_L, T_H, konfiguracijski register)	I	PIC pošlje 3 bajte za nastavitev termostata in konfiguracijskega registra (00011111_2 za 9-bitno resolucijo)
»reset«	I	PIC izvrši »reset« pulz, inicializacija novega pogovora.
»presence«	V	DS18B20 se odzove.
0xCC	I	PIC ukaže preskok iskanja naprave z določeno unikatno ROM kodo.
0x44	I	PIC ukaže konverzijo temperature.
»reset«	I	PIC izvrši »reset« pulz, inicializacija novega pogovora.
»presence«	V	DS18B20 se odzove.
0xCC	I	PIC ukaže preskok iskanja naprave z določeno unikatno ROM kodo.
0xBE	I	PIC ukaže branje pomnilnika.

9 bajtov	V	bajt 0: temperatura LSB bajt 1: temperatura MSB bajt 2: T_H bajt 3: T_L bajt 4: konfiguracijski register bajt 5: rezervirano (0xFF) bajt 6: rezervirano bajt 7: rezervirano (0x10) bajt 8: CRC

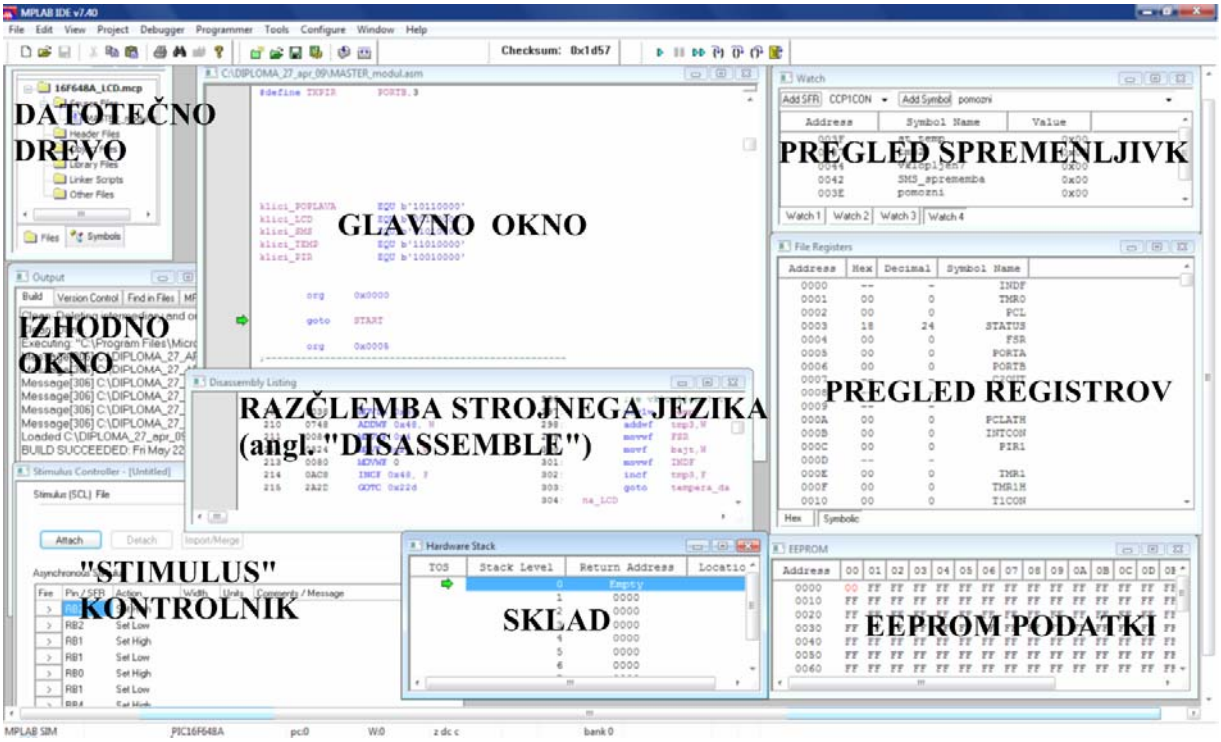
4 PROGRAMSKA OPREMA

4.1 Programsko okolje MPLAB

MPLAB integrirano razvojno okolje (angl. »IDE – Integrated Development Environment«) je prosto dostopen integriran nabor orodij za razvoj namenskih aplikacij (angl. »embedded applications«), ki vključuje Microchip PIC® in dsPIC® mikrokontrolerje. MPLAB IDE teče kot 32-bitna aplikacija na operacijskem sistemu MS Windows®, je lahka za uporabo ter vključuje kopico programskih komponent za hiter razvoj aplikacij in razhroščevanje. Uporablja enostaven uporabniški grafični vmesnik (angl. »GUI – Graphical User Interface«), kjer je možna tudi podpora zunanje programske opreme (angl. »third party software«)[13].

Grafično podobo orodja MPLAB verzija 7.40 prikazuje slika 4.1. *Datotečno drevo* prikazuje vse datoteke, vključitvene knjižnice, objektne ter druge datoteke, ki so vključene v trenutno odprti projekt. V *Glavnem oknu* imamo možnost urejanja programske kode v projektni datoteki. MPLAB podpira napreden simulator (Debugger>Select Tool>MPLAB SIM), kjer lahko virtualno simuliramo delovanje spisanega programa. *Funkcije simulatorja* vsebujejo upravljanje simulacije: »RESET« za skok programskega kurzorja na začetno pozicijo, »RUN« za zagon programa, »STEP INTO«, »STEP OVER« in »STEP OUT« za nadzor skokov. Stanja spremenljivk, ki jih definiramo v RAM pomnilniku PIC-a, lahko spremljamo v oknu *Pregled spremenljivk*. Posebne strojno določene registre, kot so na primer časovniki, USART, komparatorji, registri za prekinitev, registri za PORTA, PORTB, lahko spremljamo v oknu *Pregled registrov*. Kaj se dogaja v trajnem pomnilniku EEPROM, opazujemo v oknu *EEPROM podatki*. Ob preskokih in zankah v programu, na primer z ukazom CALL, si PIC zapomni mesto v spominu, kamor se ob koncu CALL procedure more vrniti. Ta naslov se shrani v sklad, viden v oknu *Sklad*. Zbirnik (angl. »assembler«) nam ponuja uporabo določenih mnemonikov in makrov, ki naredijo programiranje človeku lažje razumljivo. Še lažje razumevanje programa dosežemo z višjenivojskimi jeziki, kot je na primer ANSI C. Ker včasih potrebujemo natančno kontrolo nad izvajanjem strojnih ukazov, recimo pri časovnih relevantnih funkcijah, lahko natančno zaporedje strojnih ukazov vidimo v »razčlenjeni« obliki (angl. »disassembly list«), okno *Razčlemba strojnega jezika*. V simulaciji

lahko simuliramo še dogajanje zunanjega sveta. Sem spadajo: spremembe napetostnih nivojev na vratih PORTA in PORTB, sprejemanje USART podatkov itd. To določimo v oknu *»Stimulus« kontrolnik*, kjer poljubno nastavljamo stanje na pinih. Ko program spišemo, ga je treba spremeniti/prevesti v strojno kodo, da ga lahko naložimo na PIC mikrokrmilnik. Stanje prevajanja, morebitne sintaktične napake ter izhodno datoteko (.HEX) nam prikazuje okno *Izhodno okno*.



Slika 4.1 MPLAB 7.40

Tipe vhodnih in izhodnih datotek pri projektu nam prikazuje tabela 4.1.

Tabela 4.1 Tipi datotek v MPLAB 7.40

Končnica datoteke	Vhodna/izhodna datoteka[4]	Opis
.MCW	V	Datoteka z informacijami o delovnem prostoru. V en delovni prostor lahko vključimo večje število projektov (ter pripadajočih vključitvenih datotek, knjižnic itd.).
.MCP	V	Datoteka z informacijami o projektu.
.ASM	V	Izvirna datoteka; datoteka z informacijami o programu, napisanem v MPLAB zbirniku.
.INC	V	Dodatna, vključitvena datoteka. Ponavadi za dodajanje mnemonikov za posamezno vrsto mikrokrmilnika (npr. »P16F648A.INC«).
.LKR	V	Datoteka s podatki o povezovalniku (angl. »linker file«).
.STC	V	Datoteka s »stimulus« informacijami.
.LST	I	Absolutna koda, celoten pregled nad vsako vrstico, ki predstavlja en ukaz oziroma eno zasedeno pomnilniško lokacijo v spominu določenega mikrokrmilnika.
.O	I	Objektna datoteka. Vmesna koda, ki se zgenerira, ko prevedemo (angl. »compile«) izvirno datoteko.
.COD	I	Datoteka, ki vsebuje informacije o simbolih ter objektno kodo za MPLAB zbirnik.
.HEX	I	Strojna koda v heksadecimalnem zapisu. Primeren zapis za programator.
.ERR	I	Datoteka o napakah in opozorilih pri prevajanju programa.

4.2 Programska koda

4.2.1 Zbirnik

Programe za vsak PIC smo spisali v zbirniku (angl. »assembler«). Zbirnik deluje na najosnovnejšem nivoju programske logike. Na kratko: zbirnik je človeku berljiva tekstovna

[4] Vhodna datoteka je tista, ki jo samostojno vnesemo v projekt oziroma ki že obstaja ob kreiranju novega projekta, izhodno datoteko pa MPLAB zgenerira sam (npr. ob kliku opcije »Build«).

oblika strojne binarne kode, ki jo razume procesor. Za razliko od višjenivojskih programskih jezikov (npr. C, BASIC) ima vrsto prednosti, kot so na primer:

- neposredna in hitra kontrola strojnih registrov,
- hitro izvajanje programov zaradi manj redundantnih podatkov (optimizacija),
- majhna zasedenost pomnilnika s programom,
- majhno število inštrukcij za RISC (angl. »Reduced Instruction Set Computer«) procesorje,
- visoka primernost za časovno relevantne operacije, recimo realnočasovne funkcije.

Poleg zgoraj omenjenih pozitivnih lastnosti pa moramo upoštevati tudi tiste negativne, na primer:

- težavno programiranje, slaba podpora strukturnemu programiranju ter pomanjkanje objektnega programiranja,
- slabo berljiva, na pogled neurejena koda,
- dolgotrajen razvoj.

Zbirnik za serijo mikrokontrolerjev PIC16F pozna 35 inštrukcij, ki so predstavljene v tabeli 4.2.

Tabela 4.2 Inštrukcije zbirnika[14]

Inštrukcija, operand	Argumenti[5]	Primer	Opis
ADDLW	k	addlw b'10000001'	K splošnemu registru W prištej binarno vrednost '10000001'.

[5] f – naslovno polje registra; d – destinacija operacije ('0' shrani v splošni register W, '1' shrani rezultat v register f); b – številka bita v 8-bitnem registru; k – število, konstanta ali označba (angl. »label«).

ADDWF	f, d	addwf REG1,1	Dodaj vrednost iz splošnega registra W k vrednosti v REG1. Shrani v REG1.
ANDLW	k	andlw 0xFF	Opravi operacijo AND med splošnim registrom W in vrednostjo hFF (»W AND FF«).
ANDWF	f, d	andwf REG1,0	Opravi operacijo AND med splošnim registrom W in podatkovnim registrom REG1. Rezultat shrani v W.
BCF	f, b	bcf REG1,2	Pobriši 2. bit registra REG1 (vrednost '0').
BSF	f, b	bcf REG1,0	Nastavi 0. bit registra REG1 (vrednost '1').
BTFSC	f, b	btfsc REG1,3 andlw 0xFF bcf REG2,0 …	Inštrukcija odločitve. Če je 3. bit registra REG1 pobrisan, potem preskoči naslednjo inštrukcijo (to je »andlw 0xFF«). Namesto nje se izvede »prazna« operacija NOP (saj je inštrukcija že predčasno včitana), nadaljujemo naprej (izvajanje inštrukcije »bcf REG2,0«). Če 3. bit REG1 nastavljen, izvedemo »andlw 0xFF«, »bcf REG2,0« itd.
BTFSS	f, b	btfss REG1,3 andlw 0xFF bcf REG2,0 …	Inštrukcija odločitve. Če je 3. bit registra REG1 nastavljen, potem preskoči naslednjo inštrukcijo (to je »andlw 0xFF«). Namesto nje se izvede »prazna« operacija NOP (saj je inštrukcija že predčasno včitana), nadaljujemo naprej (izvajanje inštrukcije »bcf REG2,0«). Če 3. bit REG1 pobrisan, nadaljujemo s programom: »andlw 0xFF«, »bcf REG2,0« itd.
CALL	k	call termostat	Klic funkcije »termostat«. Naslednji naslovni prostor se vpiše v sklad (vrnitev iz funkcije z »return«). »Termostat« zdaj predstavlja 11-bitni naslov (med 0 in 2047), ki se naloži v programski števec (angl. »PC – Program Counter«) ter PCLATH.
CLRF	f	clrf REG1	Register REG1 je pobrisan (privzame vrednost '0').
CLRW		clrw	Pobrišemo splošni register W.
CLRWDT			Pobrišemo časovnik za samodejni reset (angl. »Watch Dog Timer«).
COMF	f, d	comf REG1,0	Register REG1 komplementiramo ter shranimo v splošni register W.
DECF	f, d	decf REG1,1	Registru REG1 odštejemo vrednost 1 ter shranimo nazaj v REG1.

DECFSZ	f, d	decfsz REG1,1 goto NAZAJ goto NAPREJ …	Inštrukcija odločitve. Od registra REG1 odštejemo 1. Če je vrednost, ki jo dobimo, enaka 0, preskočimo naslednjo vrednost (»goto NAZAJ«) ter nadaljujemo s programom: »goto NAPREJ« itd.
GOTO	k	goto NAPREJ	Skočimo na fizični naslov »NAPREJ« (= označba). Spodnjih 8 bitov naslova se naloži v PC, ostali trije pa v PCLATH. Možnih 2048 različnih naslovnih mest za skok.
INCF	f, d	incf REG1,1	Registru REG1 prištejemo vrednost 1 ter shranimo nazaj v REG1.
INCFSZ	f, d	incfsz REG1,1 goto NAZAJ goto NAPREJ …	Inštrukcija odločitve. K registru REG1 prištejemo vrednost 1. Če je vrednost, ki jo dobimo, v REG1 enaka 0, preskočimo naslednjo vrednost (»goto NAZAJ«) ter nadaljujemo s programom: »goto NAPREJ« itd.
IORLW	k	iorlw .12	Operacija inkluzivnega OR med vrednostjo, shranjeno v splošnem registru W, ter decimalno vrednostjo 12.
IORWF	f, d	iorwf REG1,0	Operacija inkluzivnega OR med vrednostjo, shranjeno v splošnem registru W, ter vrednostjo, shranjeno v registru REG1. Rezultat shranimo v W.
MOVLW	k	movlw 0xFF	V splošni register W naloži število hFF.
MOVF	f, d	movf REG1,W	Premakni vrednost iz registra REG1 v splošni register W.
MOVWF	f	movwf REG1	Vrednost iz W kopiramo v register REG1.
NOP		nop	»Prazna« operacija (angl. »No OPeration«).
RETFIE		retfie	Vrnitev iz prekinitvene rutine.
RETLW	k	retlw 'A'	Vrnitev od klica funkcije s konstantno vrednostjo, ki jo ima črka A v ASCII tabeli (to je decimalno 65) .
RETURN		return	Vrnitev iz funkcije/rutine. Iz sklada se prebere naslov, kamor se vračamo s PC.
RLF	f, d	rlf REG1,1	Vrednost na naslovu REG1 rotiramo »levo« za 1 bit skozi vrednost »CARRY«. Rezultat shranimo nazaj v register REG1. 

RRF	f, d	rrf REG1,1	Vrednost na naslovu REG1 rotiramo »desno« za 1 bit skozi vrednost »CARRY«. Rezultat shranimo nazaj v register REG1. 
SLEEP		sleep	Procesor mikrokrmilnika postavimo v stanje mirovanja. Oscilator se ustavi. Procesor zbudimo z enim od naslednjih dogodkov: - zunanji rest na MCLR pinu, - časovnik za samodejni reset (WDT) se aktivira, - prekinitev na PORTB (če je omogočena).
SUBLW	k	sublw .20	Od splošnega registra W odštej decimalno vrednost 20. Vrednost shrani v W.
SUBWF	f, d	subwf REG1,0	Odštej register W od podatkovnega registra REG1 (binarna operacija). Rezultat shrani v splošni register W.
SWAPF	f, d	swapf REG1,1	Zamenjaj spodnji in zgornji polbajt (angl. »nibble«) ter shrani v REG1.
XORLW	k	xorlw b'10110010'	Izvedemo operacijo ekskluzivnega OR med W in binarnim številom '10110010' ter shranimo v splošni register W.
XORWF	f, d	xorwf REG1,1	Izvedemo operacijo ekskluzivnega OR med W in podatkovnim registrom REG1 ter shranimo v REG1.

4.2.2 Serijska komunikacija

Za medsebojno komunikacijo med PIC-i smo uporabili samostojno definiran serijski protokol z naslednjimi lastnostmi:

- v času povezave, vendar brez uporabniških podatkov, je na fizičnem nivoju visok logični nivo »1« (Vdd, tipično +5 V),
- komunikacija poteka po sistemu »najmanj obtežen bit najprej«, podatkovni paket sestavlja:

START bit	ID_naprave0	ID_naprave1	ID_naprave2	M/S	Ukaz0	Ukaz1	Ukaz2	Ukaz3	STOP bit
»0«	*0. bit*	*1. bit*	*2. bit*	*3. bit*	*4. bit*	*5. bit*	*6. bit*	*7. bit*	*»1«*

- komunikacija je »gospodar-suženj«, gospodarja predstavlja *MASTER modul*, ostali so sužnji. Suženj lahko komunicira le v primeru, da ga gospodar pozove,
- biti ID_naprave*X* unikatno označujejo vsakega od modulov – »SLAVE«:

Tabela 4.3 Serijska komunikacija: klic posameznih modulov

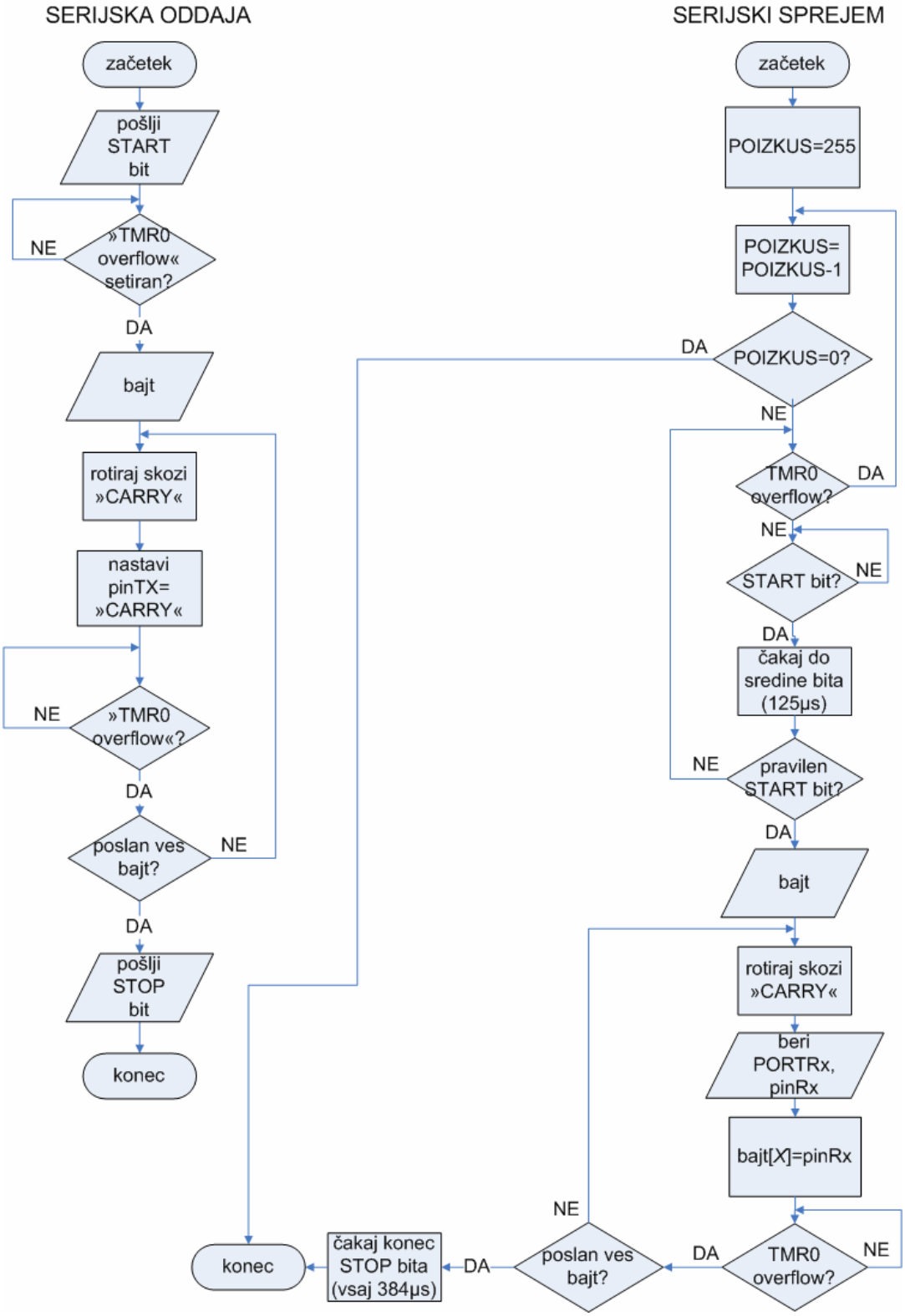
ID_naprave			Labela	Opis
0	1	2		
0	0	1	LCD	Naslovimo PIC, ki upravlja z zaslonom iz tekočih kristalov.
0	1	0	SMS	Naslovimo PIC, ki komunicira z GSM aparatom.
1	0	0	PIR	Naslovimo PIC, ki nadzoruje prostor s PIR senzorjem.
1	0	1	POPLAVA	Naslovimo PIC, ki spremlja stanje poplave.
1	1	0	TEMP	Naslovimo PIC, ki uravnava temperaturo in/ali kotel s toplo vodo.

- M/S: »MASTER/SLAVE« označuje, kdo je pošiljatelj paketa (za primer, ko imamo več sužnjev na eni poljubni podatkovni liniji do gospodarja). »1« za gospodarja, »0« za sužnja,
- Ukaz*X*: ukaz ali potrditev,
- dolžino enega bita določa čas, potreben za pretek prosto tekočega časovnika TIMER0 iz vrednosti 0->0. TIMER0 je širok 8 bitov, kar predstavlja čas 256 inštrukcijskih ciklov med dvema »TIMER0 OVERFLOW« spremembama (register INTCON, 2. bit). Pri hitrosti kristala 4 MHz (npr. interni oscilator) je bitna hitrost 256 μs/bit (približno 3,9 kbit/s), simbolna hitrost pa 10*256 = 2.56 ms/simbol.

Potek za serijski sprejem in oddajo je prikazan v diagramu 4.1[6].

[6] Legenda simbolov je prikazana v tabeli 7.1.

Diagram 4.1 Serijska komunikacija



Koda za oddajo in sprejem bajta na strani gospodarja (*MASTER modul*) je vsebovana v dveh makrih: *sprejem_serijski* (glej primer 4.1) in *oddaja_serijski* (glej primer 4.2).

Primer 4.1 Sprejemna procedura za serijsko komunikacijo

```
sprejem_serijski macro bajt,portRX,pinRX
;argumenti so:
;           -bajt:      8-bitni register, v katerega shranimo prejeti bajt
;                       podatkov
;           -portRX:    port, na katerem sprejemamo (npr. PORTB)
;           -pinRX:     pin na PORT-u, kjer sprejemamo podatke

;lokalne spremenljivke in oznake
Local sprejem_start_bit
Local naslednji_bit
Local konec_stop?
Local konec_stop2?
Local pravilen_start_bit?
Local stej
Local preveri
Local stej_dalje
Local zakljuci_sprejem

            clrf  bajt
            movlw .8                        ;resetiramo spremenljivko count
                                            ;za sprejem novih 8 bitov
            movwf count
            clrf  poskus
            movlw HIGH stej
            movwf PCLATH                    ;da lahko uporabljamo
                                            ;kratke »goto« in »call« stavke
stej
            decfsz      poskus,F
            goto  stej_dalje
            goto  zakljuci_sprejem          ;čakali smo 65 mS in ni bilo
                                            ;odgovora, zato zaključimo sprejem
stej_dalje
            clrf  TMR0                      ;štejemo čas, ki ga porabimo,
                                            ;da oddajnik na drugi strani
                                            ;odda svoj paket
            bcf   INTCON,T0IF               ;časa ima ~256 x 256 ciklov (pri
                                            ;4 MHz internem oscilatorju
                                            ;je to pribljižno 65 mS)
sprejem_start_bit                           ;... če v tem času ni odgovora,
                                            ;potem skočimo na konec sprejema
            btfsc INTCON,T0IF
            goto  stej
            btfsc portRX,pinRX
            goto  sprejem_start_bit
preveri
            movlw .125                      ;počakamo do sredine »start« bita
                                            ;in še enkrat preverimo, če je na
```

```
            movwf TMR0                  ;vhodu res logična »0«(=start bit)
            bcf   INTCON,T0IF
pravilen_start_bit?
            btfss INTCON,T0IF
            goto  pravilen_start_bit?
            btfsc portRX,pinRX          ;"0?"
            goto  sprejem_start_bit     ;ne
            clrf  TMR0                  ;da
            bcf   INTCON,T0IF
naslednji_bit
            btfss INTCON,T0IF
            goto  naslednji_bit
            btfsc portRX,pinRX
            bsf   STATUS,C
            btfss portRX,pinRX
            bcf   STATUS,C              ;če je vrednost na vhodu »0«,
                                        ;potem samo zarotiramo »bajt«,
                                        ;saj že imamo v »bajt-u«
                                        ;vpisane same ničle
            rlf   bajt,F
            movlw .5                    ;5 mikrosekund odmika, saj smo na
                                        ;sredino bita prišli takoj za
                                        ;oznako »naslednji_bit«
            movwf TMR0
            bcf   INTCON,T0IF
            decfsz      count,F
            goto  naslednji_bit

;-------------------------------------------------
;en bajt je bil sprejet!
;-------------------------------------------------

            clrf  TMR0                  ;to naredimo zato, da ima oddajna
                                        ;stran še čas poslati zadnji
                                        ;»stop« bit (v bistvu še vzamemo
                                        ;za polovico bita rezerve - torej
                                        ;čakamo 2 cela bita - za boljšo
                                        ;sinhronizacijo in morebitne
                                        ;zakasnitve)
            bcf   INTCON,T0IF
konec_stop?
            btfss INTCON,T0IF
            goto  konec_stop?
            clrf  TMR0
            bcf   INTCON,T0IF
konec_stop2?                            ;... še drugič (za boljšo
                                        ;sinhronizacijo)
            btfss INTCON,T0IF
            goto  konec_stop2?
zakljuci_sprejem
            endm                        ;»end of macro«
```

Primer 4.2 Oddajna procedura za serijsko komunikacijo

```
oddaja_serijski macro bajt,portTX,pinTX
;argumenti so:
;           -bajt:      8-bitni register, ki ga pošiljamo
;           -portTX:    port, na katerem pošiljamo (npr. PORTB)
;           -pinTX:     pin na PORT-u, kjer pošiljamo

;lokalne spremenljivke in oznake
Local poslji_paket
Local start_bit
Local poslji_bajt
Local je_bajt_poslan
Local stop_bit

poslji_paket                                ;pošljemo komunikacijski bajt
                                            ;zapakiran med "start" in "stop"
                                            ;bit
            clrf  TMR0
            bcf   INTCON,T0IF               ;zbrišemo zastavico za prekinitev
                                            ;časovnika TIMER0
            bcf   portTX,pinTX
start_bit                                   ;"0"
            btfss INTCON,T0IF
            goto  start_bit                 ;"start" bit še ni odposlan
                                            ;"start" bit je bil poslan (oz.
                                            ;TMR0 je setiral zastavico
                                            ;"TMR0 overflow")
            movlw .8
            movwf count                     ;nam odšteva število rotacij
                                            ;"bajt-a"
poslji_bajt
            rlf   bajt,F                    ;spremenljivko "bajt" rotiramo
                                            ;levo skozi "CARRY" bit (se
                                            ;nahaja v STATUS registru, bit
                                            ;0)
            btfsc STATUS,C
            bsf   portTX,pinTX
            btfss STATUS,C
            bcf   portTX,pinTX
            bcf   INTCON,T0IF
            movlw .11                       ;v TMR0 naložimo vrednost 11
                                            ;(saj želimo hitrost 256 uS/bit,
                                            ;nekaj časa smo že porabili za
                                            ;ta odsek kode)
            movwf TMR0
je_bajt_poslan
            btfss INTCON,T0IF
            goto  je_bajt_poslan
            decfsz      count,F             ;smo že poslali vseh 8 bitov?
            goto  poslji_bajt               ;ne
            bsf   portTX,pinTX              ;da, pripravi se na »stop« bit
            bcf   INTCON,T0IF

stop_bit                                    ;"1"
```

```
        btfss INTCON,T0IF
        goto  stop_bit
        endm                            ;»end of macro«
```

4.2.3 MASTER modul

Ob vklopu sistema inteligentne hiše se vsi moduli nahajajo v stanju pripravljenosti. Sistem se aktivira, ko uporabnik vklopi sistem ročno, z vnosom ustrezne šifre (PIN kode). Aktivacija se izvrši na *LCD modulu* s pomočjo tipk. *MASTER modul* čaka na potrditev vklopa (ukaz »LCD_VKLOPI_VSE«). V stanju pripravljenosti sistema sta na razpolago le dva ukaza, in sicer »LCD_termostat_temperatura«, ki prečita temperaturo iz DS18B20 za prikaz trenutne temperature v meniju za nastavitev funkcij termostata. Drugi ukaz je »LCD_Alarmi_PIR_sirena«, s katerim nudimo uporabniku izbiro frekvence sirene iz prednastavljenih vrednosti, v korakih po 700 Hz.

Po vklopu *LCD modul* pošlje uporabnikove nastavitve: alarme, telefonsko številko za obveščanje uporabnika na daljavo, izbrano temperaturo za termostat, PIN številko za preverjanje pristnosti uporabnika (recimo preko SMS sporočil). *MASTER* vklopi module glede na izbrane lastnosti, ki jih od sistema želimo. Program nato »pade« v neskončno zanko klicanja vsakega od vklopljenih modulov. Za ponovni prehod sistema inteligentne hiše v stanje pripravljenosti je treba vpisati pravilno šifro (s tem se tudi izklopijo vsi moduli). Ta potek prikazujejo diagrami od 4.2 do 4.4.

Diagram 4.2 MASTER modul 1/3

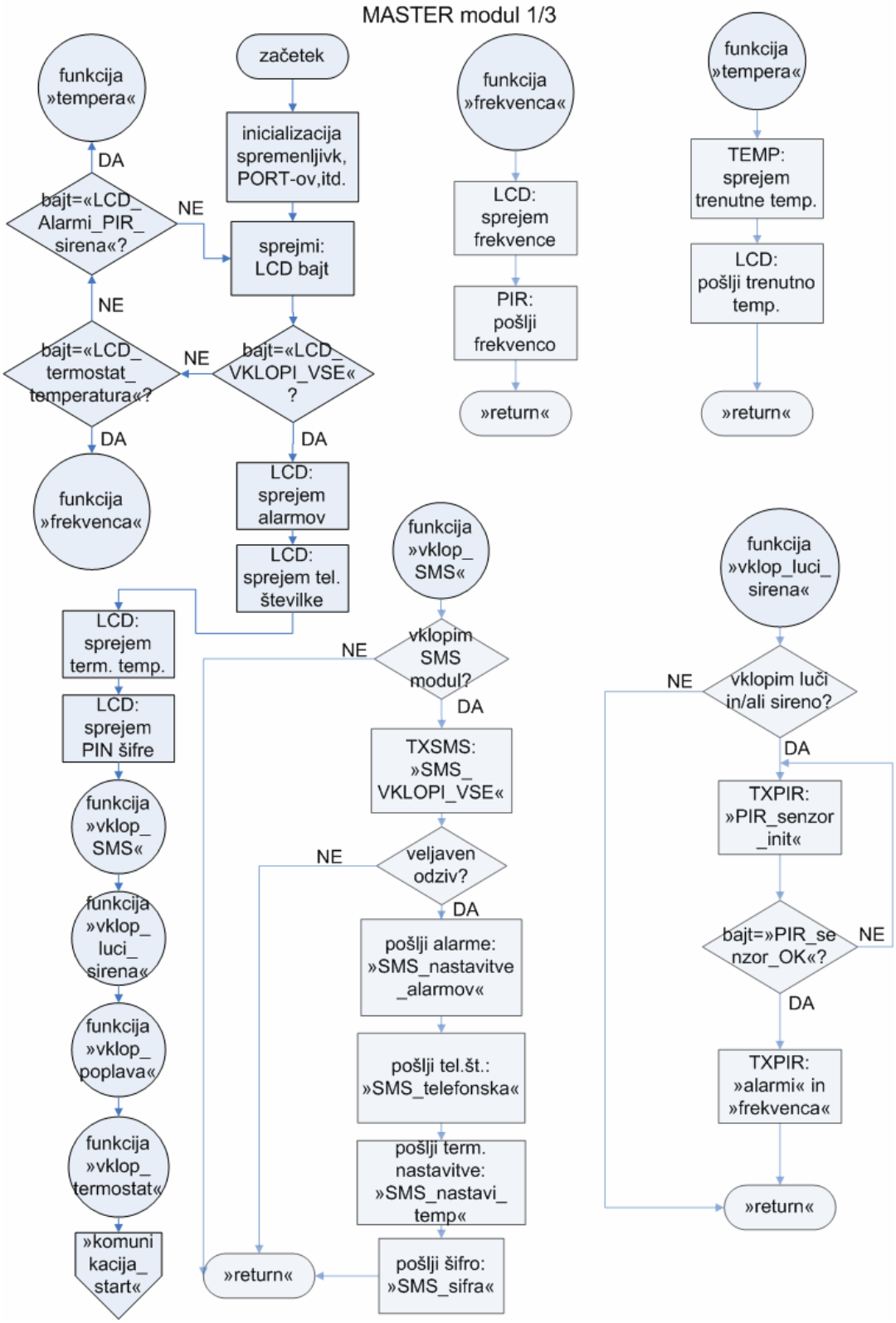


Diagram 4.3 Master modul 2/3

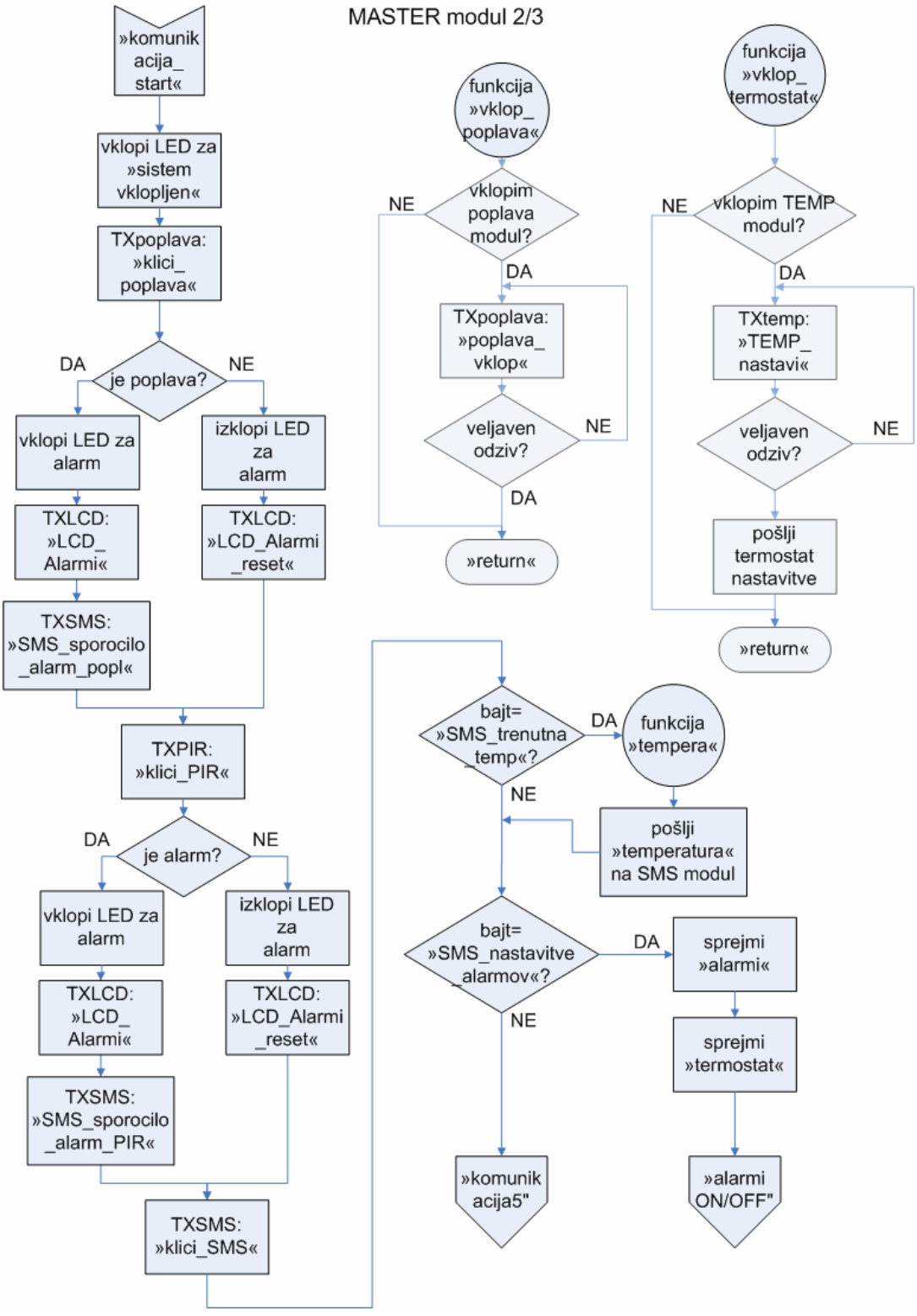
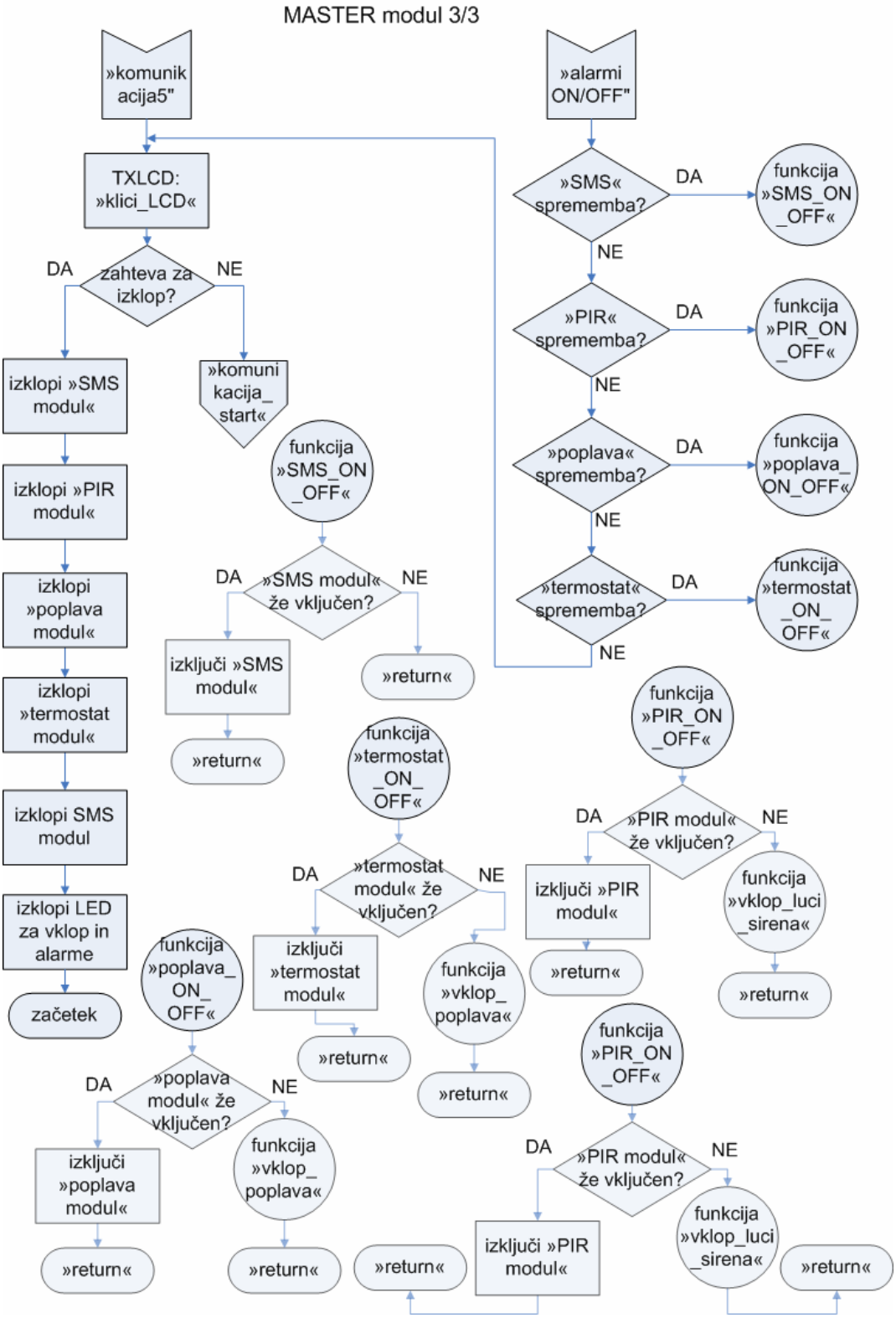


Diagram 4.4 MASTER modul 3/3



Program se začne z direktivo za tip procesorja, vključitvijo zunanjih datotek, nastavitvijo varnostnih opozoril, ki se pojavijo pri prevajanju programa, ter definicijo konfiguracijskega registra.

Primer 4.3 MASTER modul: začetek programa

```
      LIST    p=16F648a                    ;pove zbirniku/prevajalniku,
                                           ;kateri čip bomo uporabili
      include »P16F648a.inc«               ;»include« datoteka z mnemoniki za
                                           ;registre
      include »serijska_komunikacija.inc« ;datoteka z makri za serijsko
                                           ;komunikacijo
      include »gotom_macro.inc«            ;makro za »dolge« skoke
      include »callm_macro.inc«            ;makro za »dolge« klice
      include »mnemoniki_master.inc«       ;vključitvena datoteka

      ERRORLEVEL  0, -302, -205, -207      ;zaduši varnostna opozorila v
                                           ;»IZHODNEM OKNU«
                                           ;0    prikaži vse napake in
                                           ;     opozorila
                                           ;-302 ne prikaži opozoril o
                                           ;     operaciji nad registrom, ki
                                           ;     ni v »bank0«
                                           ;-205 ne prikaži opozoril o
                                           ;     napakah direktiv v 1.
                                           ;stolpcu
                                           ;-207 ne prikaže opozoril o
                                           ;     označbah, ki niso v
                                           ;1.stolpcu

      __CONFIG _CP_OFF & _WDT_OFF &
      _PWRTE_ON & _BODEN_ON & _LVP_OFF &
      _MCLRE_ON & _INTRC_OSC_NOCLKOUT      ;konfiguracijski register
```

V vključitvenih datotekah »gotom_macro.inc« in »callm_macro.inc« imamo definirana makra za skok na oznako oziroma klic funkcije, ki je v drugem 2048-mestnem bloku. Inštrukciji CALL in GOTO (glej tabelo 4.1) lahko naslovita le 2048 pomnilniških lokacij, za doseg drugih lokacij (pri PIC16F648A jih imamo 4096) moramo urediti še vrednost v PCLATH[15]. Za PIC16F628A ta ukrep ni potreben. Makra prikazujeta primera 4.4 in 4.5.

Primer 4.4 Dolg skok (»gotom_macro.inc«)

```
gotom macro jump
      movlw HIGH jump
```

```
      movwf PCLATH
      goto  jump
      endm
```

Primer 4.5 Dolg klic (»callm_macro.inc«)

```
callm macro rutina

Local address
address
      movlw HIGH address
      movwf PCLATH_HIGH       ;naložimo PCLATH trenutnega naslova v
                              ;spremenljivko PCLATH_HIGH
      movlw HIGH rutina
      movwf PCLATH            ;v PCLATH naložimo višji bajt od naslova
                              ;"rutina"
      call  rutina
      movf  PCLATH_HIGH,W
      movwf PCLATH            ;za klicem rutine ponovno naložimo PCLATH s
                              ;trenutnega naslova
      endm
```

Programu moramo definirati spremenljivke, ki jih bomo uporabili v času izvajanja. Shranjene so v SRAM-u. V primeru prekinitve napajanja so vrednosti nedefinirane.

Primer 4.6 MASTER modul: definicija spremenljivk

```
            cblock      0x20        ;začetek RAM lokacij
                  count             ;uporabljeno v zankah
                  count1            ;za časovno zakasnitev
                  counta            ;za časovno zakasnitev
                  countb            ;za časovno zakasnitev
                  bajt              ;serijska komunikacija
                  alarm             ;pomožni register za alarme
                  alarmPIR          ;trenutni alarmi na PIR senzorju
                  telefonska:.12    ;12 lokacij za telefonsko številko;
                                    ;maksimalno 11 (8-bitnih)znakov, na
                                    ;koncu terminator 0
                  termostat:.4      ;termostat (znaki)
                  temp:.4           ;trenutna temperatura (znaki)
                  st_znakov_tel     ;št. znakov za telefonsko številko
                  st_znakov_temp    ;število znakov za temperaturo
                  pomozni           ;pomožni register
                  st_temp           ;pomožni register za temperaturo
                  ALARMI            ;vrednosti alarmov ('0' – izkloljen, '1'
                                    ;- vklopljen)
                                    ;bit 0: SMS
                                    ;bit 1: luči
```

```
                                ;bit 2: sirena
                                ;bit 3: poplava
                                ;bit 4: termostat
                                ;bit 5: frekvenca sirene 1/3
                                ;bit 6: frekvenca sirene 2/3
                                ;bit 7: frekvenca sirene 3/3
                ALARMI_SMS      ;alarmi ki jih sprejmemo od SMS_modul-a
                SMS_sprememba   ;spremembe alarmov na SMS modulu
                poskus          ;serijska komunikacija, sprejem
                vklopljen?      ;označuje vklop/izklop sistema
                alarm_na_LCD    ;vodi evidenco, kateri alarm/reset
                                ;alarma smo že izpisali na LCD ekranu
                                ;--------TRENUTNO STANJE ALARMOV:------
                                ;0. bit: poplava SET,
                                ;1. bit: poplava RESET,
                                ;2. bit: PIR SET,
                                ;3. bit: PIR RESET
                                ;----------STANJE NA LCD-ju:-----------
                                ;4. bit: poplava SET,
                                ;5. bit: poplava RESET,
                                ;6. bit: PIR SET,
                                ;7. bit: PIR RESET
                tmp2            ;pomožni register
                tmp3            ;pomožni register
                tmp4            ;pomožni register
                sifra:.4        ;PIN koda (znaki)
                PCLATH_HIGH     ;callm_macro
        endc
```

Sopomenke (mnemoniki) služijo za lažjo berljivost programske kode. Ob prevajanju programa jih prevajalnik avtomatsko nadomesti z definiranimi vrednostmi. Za vlogo gospodarja se nahajajo v datoteki »mnemoniki_master.inc« (primer 4.7).

Primer 4.7 MASTER modul: definicije mnemonikov, datoteka »mnemoniki_master.inc«

```
;definicije pinov
#define     RXtrisPOPLAVA     TRISA,1
#define     TXtrisPOPLAVA     TRISA,0
#define     RXpoplava         PORTA,1
#define     TXpoplava         PORTA,0
#define     RXtrisLCD         TRISB,5
#define     TXtrisLCD         TRISB,4
#define     RXLCD             PORTB,5
#define     TXLCD             PORTB,4
#define     RXtrisSMS         TRISB,0
#define     TXtrisSMS         TRISB,1
#define     RXSMS             PORTB,0
#define     TXSMS             PORTB,1
#define     RXtrisTEMP        TRISA,7
```

```
#define     TXtrisTEMP        TRISA,6
#define     RXtemp            PORTA,7
#define     TXtemp            PORTA,6
#define     RXtrisPIR         TRISB,2
#define     TXtrisPIR         TRISB,3
#define     RXPIR             PORTB,2
#define     TXPIR             PORTB,3

;serijska komunikacija
klici_POPLAVA           EQU   b'10110000'
klici_LCD               EQU   b'00110000'
klici_SMS               EQU   b'01010000'
klici_TEMP              EQU   b'11010000'
klici_PIR               EQU   b'10010000'
;-----------
;LCD
;-----------
LCD_nastavitve_tel_1_TX             EQU   b'00110001'
LCD_nastavitve_tel_1_RX             EQU   b'00100001'
LCD_nastavitve_cas_sprozitve_TX     EQU   b'00110011'
LCD_nastavitve_cas_sprozitve_RX     EQU   b'00100011'
LCD_Alarmi_PIR_poslji_SMS_TX        EQU   b'00110100'
LCD_Alarmi_PIR_poslji_SMS_RX        EQU   b'00100100'
LCD_Alarmi_PIR_sirena_TX            EQU   b'00110110'
LCD_Alarmi_PIR_sirena_RX            EQU   b'00100110'
LCD_Alarmi_TX                       EQU   b'00110111'
LCD_Alarmi_RX                       EQU   b'00100111'
LCD_termostat_temperatura_TX        EQU   b'00111000'
LCD_termostat_temperatura_RX        EQU   b'00101000'
LCD_termostat_vklopi_TX             EQU   b'00111001'
LCD_termostat_vklopi_RX             EQU   b'00101001'
LCD_VKLOPI_VSE_TX                   EQU   b'00111111'
LCD_VKLOPI_VSE_RX                   EQU   b'00101111'
LCD_Alarmi_reset_TX                 EQU   b'00111010'
LCD_Alarmi_reset_RX                 EQU   b'00101010'
LCD_sifra_TX                        EQU   b'00111101'
LCD_sifra_RX                        EQU   b'00101101'
LCD_IZKLOPI_VSE_TX                  EQU   b'00111011'
LCD_IZKLOPI_VSE_RX                  EQU   b'00101011'
;-----------
;SMS
;-----------
SMS_telefonska_TX                   EQU   b'01010001'
SMS_telefonska_RX                   EQU   b'01000001'
SMS_nastavitve_alarmov_TX           EQU   b'01010010'
SMS_nastavitve_alarmov_RX           EQU   b'01000010'
SMS_sporocilo_alarm_popl_TX         EQU   b'01010011'
SMS_sporocilo_alarm_popl_RX         EQU   b'01000011'
SMS_sporocilo_alarm_PIR_TX          EQU   b'01011001'
SMS_sporocilo_alarm_PIR_RX          EQU   b'01001001'
SMS_trenutna_temp_TX                EQU   b'01010101'
SMS_trenutna_temp_RX                EQU   b'01000101'
SMS_nastavi_temp_TX                 EQU   b'01010110'
SMS_nastavi_temp_RX                 EQU   b'01000110'
```

```
SMS_VKLOPI_VSE_TX                   EQU   b'01010111'
SMS_VKLOPI_VSE_RX                   EQU   b'01000111'
SMS_napaka_modul_TX                 EQU   b'01011000'
SMS_napaka_modul_RX                 EQU   b'01001000'
SMS_sifra_TX                        EQU   b'01011100'
SMS_sifra_RX                        EQU   b'01001100'
SMS_IZKLOPI_VSE_TX                  EQU   b'01011101'
SMS_IZKLOPI_VSE_RX                  EQU   b'01001101'
;-----------
;PIR
;-----------
PIR_senzor_alarm_TX                 EQU   b'10010001'
PIR_senzor_alarm_RX                 EQU   b'10000001'
PIR_senzor_OK_TX                    EQU   b'10010010'
PIR_senzor_OK_RX                    EQU   b'10000010'
PIR_senzor_init_TX                  EQU   b'10010100'
PIR_senzor_init_RX                  EQU   b'10000100'
PIR_senzor_IZKLOPI_VSE_TX           EQU   b'10010111'
PIR_senzor_IZKLOPI_VSE_RX           EQU   b'10000111'
PIR_sirena_frekvenca_TX             EQU   b'10011000'
PIR_sirena_frekvenca_RX             EQU   b'10001000'
;-----------
;poplava
;-----------
poplava_NE_TX                       EQU   b'10110010'
poplava_NE_RX                       EQU   b'10100010'
poplava_DA_TX                       EQU   b'10110011'
poplava_DA_RX                       EQU   b'10100011'
poplava_vklop_TX                    EQU   b'10110111'
poplava_vklop_RX                    EQU   b'10100111'
poplava_izklop_TX                   EQU   b'10111011'
poplava_izklop_RX                   EQU   b'10101011'
;-----------
;TEMP
;-----------
TEMP_temperatura_TX                 EQU   b'11010001'
TEMP_temperatura_RX                 EQU   b'11000001'
TEMP_vklopi_TX                      EQU   b'11010010'
TEMP_vklopi_RX                      EQU   b'11000010'
TEMP_nastavi_TX                     EQU   b'11010100'
TEMP_nastavi_RX                     EQU   b'11000100'
TEMP_IZKLOPI_VSE_TX                 EQU   b'11010110'
TEMP_IZKLOPI_VSE_RX                 EQU   b'11000110'
;-----------
;EOF
;-----------
```

Izvajanje strojne kode se začne na reset vektorju (naslov 0x000). Na lokaciji 0x004 skočimo v prekinitveno rutino (če jo uporabljamo). Potem mikrokontroler korak za korakom

izvršuje inštrukcije. Najprej je potrebno inicializirati uporabljene spremenljivke in registre (primer 4.8).

Primer 4.8 MASTER modul: inicializacija

```
            org   0x0000
            goto  START

            org   0x0005
START
            movlw 0x07
            movwf CMCON             ;izklop komparatorjev
            bsf   STATUS,RP0
            movlw b'00000000'
            movwf TRISB
            movwf TRISA             ;vsi so izhodi
            movlw b'10001000'
            movwf OPTION_REG        ;interni urin takt za TMR0,
                                    ;predmnožilnik nastavljen na WDT,«pull-
                                    ;up« upori nastavljeni na PORTB-u
            movlw b'00000000'
            movwf INTCON
            bcf   STATUS,RP0

;SISTEM JE IZKLOPLJEN!
            bcf   PORTA,2           ;vse izklopljeno!
;inicializacija pinov
            init_serijski     RXtrisPOPLAVA,TXtrisPOPLAVA
            init_serijski     RXtrisLCD,TXtrisLCD
            init_serijski     RXtrisSMS,TXtrisSMS
            init_serijski     RXtrisTEMP,TXtrisTEMP
            init_serijski     RXtrisPIR,TXtrisPIR

            clrf  vklopljen?        ;na začetku predpostavimo izklopljen
                                    ;sistem
            clrf  alarm
            clrf  alarmPIR
            clrf  ALARMI
```

Po uspešni inicializaciji program pade v neskončno zanko preverjanja odziva *LCD modula*. Program čaka na vklop sistema inteligentne hiše. Preveri še, če je potrebno posredovati informacije o trenutni temperaturi (*TEMP modul*) in/ali frekvenci sirene (*PIR modul*).

Primer 4.9 MASTER modul: inicializacija

```
pocakam_na_VKLOPI_VSE
            movlw klici_LCD         ;komunikacijski bajt za povezavo z LCD
                                    ;modul-om
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD


            movlw b'00101111'       ;VKLOPI_VSE?
            subwf bajt,W
            btfsc STATUS,Z          ;ali smo sprejeli pravi paket?
            goto  sprejmi_alarme    ;da
                                    ;ne
                                    ;preverimo, če mogoče LCD modul
                                    ;potrebuje podatke o trenutni
                                    ;temperaturi
            movlw b'00101000'       ;trenutna temperatura?
            subwf bajt,W
            btfsc STATUS,Z
            call  tempera           ;da
                                    ;ne
                                    ;;preverimo, če mogoče LCD modul
                                    ;potrebuje podatke o frekvenci zvočnika
                                    ;(PIR modul)
            movlw b'00100110'
            subwf bajt,W
            btfsc STATUS,Z
            call  frekvenca         ;da
                                    ;ne
            goto  pocakam_na_VKLOPI_VSE   ;zankaj na začetek
```

Funkcija »tempera« pokliče *TEMP modul* in povpraša o trenutni temperaturi, ter informacijo pošlje do *LCD modula* (če se nahajamo v stanju pripravljenosti sistema inteligentne hiše) ali shrani na 4 pomnilniške lokacije pod imenom »temp«. Temperatura je shranjena kot ASCII vrednost 4 znakov-številk. Za primer trenutne temperature *23,5°C* shranimo vrednosti: 0x30 (ASCII koda za znak »0«), 0x32 (ASCII koda za znak »2«), 0x33 (ASCII koda za znak »0«) in 0x35 (ASCII koda za znak »0«). Funkcijo »tempera« prikazuje primer 4.10.

Primer 4.10 MASTER modul: funkcija »tempera«

```
tempera
            movlw TEMP_temperatura_TX
            movwf bajt
```

```
            oddaja_serijski   bajt,TXtemp
            sprejem_serijski  bajt,RXtemp
                                        ;čakamo na potrditev...
            movlw TEMP_temperatura_RX   ;temperatura?
            subwf bajt,W
            btfss STATUS,Z
            goto  tempera               ;ponovi postopek
                                        ;ok, lahko nadaljujemo, sprejem 4
                                        ;bajtov
            clrf  tmp3
            movlw .4
            movwf st_temp
sprejem_temperature
            movlw TEMP_temperatura_TX   ;kontrolni bajt
            movwf bajt
            oddaja_serijski   bajt,TXtemp
            sprejem_serijski  bajt,RXtemp
            btfss vklopljen?,0          ;smo v »stanju prirpavljenosti«
                                        ;ali v »stanju delovanja«?
            goto  na_LCD                ;»stanje pripravljenosti«
                                        ;»stanje delovanja« (vklopljeno!)
                                        ;temperaturo bomo shranili na
                                        ;lokacijo »temp«
            movlw temp
            addwf tmp3,W
            movwf FSR                   ;kazalec za indirektno
                                        ;naslavljanje
            movf  bajt,W
            movwf INDF                  ;ni fizični register, označuje
                                        ;indirektno naslovljen register
            incf  tmp3,F
            goto  tempera_dalje
na_LCD
            oddaja_serijski   bajt,TXLCD
tempera_dalje
            decfsz      st_temp,F
            goto  sprejem_temperature

            return
```

Funkcija »frekvenca« sprejme bajt z alarmi, v katerem so tudi podatki o frekvenci (najvišji 3 biti). Prednastavljene vrednosti prikazuje tabela 4.7 (glej poglavje »4.2.6 PIR modul«).

Uporabnik pri nastavitvi frekvence sliši ton z izbrano frekvenco, da lahko določi najprimernejšo s slušno percepcijo. Programsko kodo funkcije »frekvenca« prikazuje primer 4.11.

Primer 4.11 MASTER modul: funkcija »frekvenca«

```
frekvenca
            movlw LCD_Alarmi_PIR_sirena_TX;OK, smo sprejeli prošnjo za
                                          ;spremembo frekvence na zvočniku
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD  ;dobimo podatke o alarmih (3
                                          ;najvišji biti so podatki o
                                          ;frekvenci)
            movf  bajt,W
            movwf alarmPIR
            movlw PIR_sirena_frekvenca_TX
            movwf bajt
            oddaja_serijski   bajt,TXPIR  ;želimo poslati podatke o
                                          ;frekvenci zvočnika
            sprejem_serijski  bajt,RXPIR
            movf  alarmPIR,W
            movwf bajt
            oddaja_serijski   bajt,TXPIR  ;pošljemo frekvenco
            sprejem_serijski  bajt,RXPIR

            return
```

Ko sistem vklopimo, ga je najprej potrebno inicializirati. Vse nastavitve se iz *LCD modula* pošljejo na *MASTER modul*, ki jih logično sprocesira ter opravi postopke zagona za vsak posamezen modul v sistemu.

Primer 4.12 MASTER modul: inicializacija

```
sprejmi_alarme                            ;VKLOPI_VSE! Najprej sprejmemo
                                          ;nastavitve za vse alarme
            movlw klici_LCD
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movf  bajt,W
            movwf ALARMI
                                          ;sprejmemo telefonsko številko
            movlw .11
            movwf st_znakov_tel
            clrf  pomozni                 ;štejemo znake
sprejem_telefonske
            movlw klici_LCD
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw telefonska
            addwf pomozni,W
            movwf FSR
```

```
            movf  bajt,W
            movwf INDF
            incf  pomozni,F
            decfsz      st_znakov_tel,F
            goto  sprejem_telefonske
                                        ;sprejmemo temperaturo termostata
            movlw .4
            movwf st_znakov_temp
            clrf  pomozni
sprejem_temp
            movlw klici_LCD
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw termostat
            addwf pomozni,W
            movwf FSR
            movf  bajt,W
            movwf INDF
            incf  pomozni,F
            decfsz      st_znakov_temp,F
            goto  sprejem_temp
                                        ;sprejmemo šifro
            movlw .4
            movwf st_znakov_temp
            clrf  pomozni
sprejem_sifra
            movlw LCD_sifra_TX
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw sifra
            addwf pomozni,W
            movwf FSR
            movf  bajt,W
            movwf INDF
            incf  pomozni,F
            decfsz      st_znakov_temp,F
            goto  sprejem_sifra
            call  vklop_SMS             ;vklop modulov
            call  vklop_luci_sirena
            call  vklop_poplava
            call  vklop_termostat

            gotom komunikacija_start
```

Vsakega od modulov-»sužnjev« vklopimo s klicem rutine. Imamo 4 funkcije, in sicer »vklop_SMS« za vklop *SMS modula*, »vklop_luci_sirena« za vklop luči in zvočnika na *PIR modulu*, »vklop_poplava« za *POPLAVA modul* in »vklop_termostat« za *TEMP modul*. Rutine so prikazane v primerih 4.13 do 4.16.

Primer 4.13 MASTER modul: funkcija »vklop_SMS«

```
vklop_SMS
            btfss ALARMI,0
            return                          ;SMS modula ni treba vklopiti

            movlw SMS_VKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS

            movlw SMS_VKLOPI_VSE_RX         ;SMS_VKLOPI_VSE?
            subwf bajt,W
            btfss STATUS,Z
            return                          ;ni pravega odziva, vrni se nazaj
                                            ;odziv je OK, sedaj počakamo da
                                            ;SMS_modul inicializira GSM aparat
sprejem_SMS_init
            sprejem_serijski  bajt,RXSMS
            movlw SMS_nastavitve_alarmov_RX     ;pošlji nastavitve alarmov?
            subwf bajt,W
            btfsc STATUS,Z
            goto  SMS_init_OK

            movlw SMS_napaka_modul_RX       ;SMS_napaka?
            subwf bajt,W
            btfss STATUS,Z
            goto  sprejem_SMS_init          ;zankamo tako dolgo, dokler ne
                                            ;dobimo pravega odziva
            movlw LCD_napaka_modul_TX       ;NAPAKA na modulu
            movwf bajt                      ;pošlji podatek na LCD modul
            oddaja_serijski   bajt,TXLCD
            return
SMS_init_OK                                 ;init GSM = OK, nadaljujemo...
            movf  ALARMI,W                  ;»ALARMI«
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS

            movlw SMS_nastavi_temp_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
                                            ;»termostat« nastavitve
            clrf  tmp3
            movlw .4
            movwf tmp2
 SMS_termostat
            movlw termostat
            addwf tmp3,W
            movwf FSR
            movf  INDF,W
            movwf bajt
```

```
            incf  tmp3,F
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            decfsz      tmp2,F
            goto  SMS_termostat
SMS_tel                                 ;»telefonska številka«
            movlw SMS_telefonska_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS

            movlw SMS_telefonska_RX     ;SMS telefonska OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  SMS_tel

            clrf  tmp3
            movlw .11
            movwf tmp2
SMS_telefonska                          ;pošljemo 11 znakov za mednarodno
                                        ;telefonsko številko (primer:
                                        ;«38640567692«)
            movlw telefonska
            addwf tmp3,W
            movwf FSR
            movf  INDF,W
            movwf bajt
            incf  tmp3,F
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            decfsz      tmp2,F
            goto  SMS_telefonska
SMS_sifra                               ;šifra
            movlw SMS_sifra_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS

            movlw SMS_sifra_RX          ;SMS šifra OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  SMS_sifra
            clrf  tmp3
            movlw .4                    ;pošljemo 4 znake (default:
                                        ;"1234")
            movwf tmp2
SMS_sifra_poslji
            movlw sifra
            addwf tmp3,W
            movwf FSR
            movf  INDF,W
            movwf bajt
            incf  tmp3,F
            oddaja_serijski   bajt,TXSMS
```

```
            sprejem_serijski  bajt,RXSMS
            decfsz      tmp2,F
            goto  SMS_sifra_poslji
            return                          ;konec rutine »vklop_SMS«
```

»Vklop_SMS« vklopi *SMS modul* in čaka da le-ta nastavi vse parametre na GSM telefonu. Pri neuspešni inicializaciji (npr.: ni priključenega GSM aparata, GSM se ne odziva itd.) se vrne negativna potrditev na LCD zaslon (izpis »NAPAKA modul«) ter izhod iz funkcije. Pozitivni potrditvi sledi izmenjava nastavitev (alarmi, temperatura termostata, telefonska številka, šifra).

Primer 4.14 MASTER modul: funkcija »vklop_luci_sirena«

```
vklop_luci_sirena                       ;1. in/ali 2. bit v »ALARMI«
            btfsc ALARMI,1
            goto  vklop_luci_sirena2
            btfss ALARMI,2
            return                      ;»goto vklop_poplava«

vklop_luci_sirena2
            movlw PIR_senzor_init_TX    ;zahteva za vklop/inicializacijo!
            movwf bajt
            oddaja_serijski   bajt,TXPIR
            sprejem_serijski  bajt,RXPIR
            movlw PIR_senzor_init_RX    ;inicializacija potrjena?
            subwf bajt,W
            btfsc STATUS,Z
            goto  vklop_luci_sirena2    ;ne
                                        ;da
            movlw PIR_senzor_OK_RX      ;PIR senzor OK!
            subwf bajt,W
            btfss STATUS,Z
            return                      ;»goto vklop_poplava«
            movf  ALARMI,W
            movwf bajt
            oddaja_serijski   bajt,TXPIR
            return
```

PIR modul sprejme zahtevo za inicializacijo. Sledi podatkovni paket z alarmi, v katerem je zapisano, kaj se bo v primeru alarma sprožilo (sirena, luč ali oboje) ter podatki za frekvenco sirene.

Primer 4.15 MASTER modul: funkcija »vklop_poplava«

```
vklop_poplava
            btfss ALARMI,3                ;3. bit »ALARMI«: »poplava«
            return                        ;»goto vklop_termostat«
                                          ;vklopi POPLAVA_modul
            movlw poplava_vklop_TX
            movwf bajt
            oddaja_serijski   bajt,TXpoplava
            sprejem_serijski  bajt,RXpoplava
            movlw poplava_vklop_RX        ;poplava_vklop OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  vklop_poplava           ;ne, ponovno poskušamo
                                          ;ok, modul se bo inicializiral!
            return
```

Primer 4.16 MASTER modul: funkcija »vklop_termostat«

```
vklop_termostat
            btfsc ALARMI,4
            goto  nastavi_termostat
            return                        ;»gotom komunikacija_start«
                                          ;vklopi TEMP_modul
nastavi_termostat
            movlw TEMP_nastavi_TX
            movwf bajt
            oddaja_serijski   bajt,TXtemp
            sprejem_serijski  bajt,RXtemp
            movlw TEMP_nastavi_RX         ;potrditev za "TEMP_nastavi"?
            subwf bajt,W
            btfss STATUS,Z
            goto  nastavi_termostat
                                          ;imamo OK signal, da lahko
                                          ;pošljemo temperaturo termostata
            movlw .4
            movwf st_znakov_temp
            clrf  pomozni
oddaja_temperature
            movlw termostat
            addwf pomozni,W
            movwf FSR
            movf  INDF,W
            movwf bajt
            oddaja_serijski   bajt,TXtemp
            sprejem_serijski  bajt,RXtemp
            movlw TEMP_nastavi_RX         ;TEMP_nastavi OK?
            subwf bajt,W
            btfss STATUS,Z
            nastavi_termostat             ;ne, odziv ni OK,poskusimo ponovno
                                          ;imamo OK signal, da lahko
                                          ;pošljemo temperaturo termostata
            incf  pomozni,F
```

```
            decfsz      st_znakov_temp,F
            goto  oddaja_temperature
            return
```

Po vklopu potrebnih modulov *MASTER modul* preide v proces klicanja vsakega od njih, enega za drugim, ter posredovanja informacij med njimi. Drugi bit PORTA tudi povežemo na logično visok nivo, priključena LED zasveti in označuje aktiviranost sistema inteligentne hiše.

Proces klicanja modulov začnemo s *POPLAVA modulom*. Preverimo, ali smo zaznali poplavo. Če DA, to sporočimo *LCD modulu*, ki izpiše obvestilo na ekranu. Alarm pošljemo še na *SMS modul*, ki pošlje SMS sporočilo s vsebino »Alarm poplava!« na uporabnikovo telefonsko številko. Modul se lahko z majhnimi spremembami programske kode in strojne opreme uporabi tudi kot senzor za dež. Ta bi v inteligentni hiši služil na primer za spuščanje rolet, zaprtje strešnih oken itd. Programsko kodo prikazuje primer 4.17.

Primer 4.17 MASTER modul: klicanje modulov – POPLAVA modul

```
komunikacija_start
            movlw .1
            movwf vklopljen?              ;označuje aktiviran sistem!
            bsf   PORTA,2                 ;LED za »sistem je aktiviran«
      movlw HIGH komunikacija1
      movwf PCLATH                        ;da bomo lahko izvajali »kratke«
                                          ;skoke z »goto« (namesto »gotom«)
komunikacija1                             ;poplava
            btfss ALARMI,3
            goto  komunikacija2           ;dalje na naslednji modul
            movlw klici_POPLAVA           ;komunikacijski bajt za povezavo s
                                          ;POPLAVA_modul-om
            movwf bajt
            oddaja_serijski   bajt,TXpoplava
            sprejem_serijski  bajt,RXpoplava
                                          ;zdaj smo sprejeli en bajt od
                                          ;sužnja (shranjeno v »bajt«)
            movlw poplava_DA_RX           ;poplava_DA?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija1_2         ;preveri za drugi ukaz ...
                                          ;paket je res za nas, sprožena je
                                          ;bila poplava, na LCD pošljemo:
                                          ;»LCD_Alarmi_poplava«
            movlw b'11110000'
            andwf alarm_na_LCD,F
```

```
            bsf   alarm_na_LCD,0          ;v nižjem nibble-u setiramo
                                          ;0. bit: "poplava SET"
            btfsc alarm_na_LCD,4          ;smo na LCD že poslali podatek, da
                                          ;je prišlo do poplave?
            goto  komunikacija2           ;da
                                          ;ne, nismo še
            bsf         PORTA,3           ;LED indikator za alarme
poplava_na_LCD                            ;pošljemo na LCD ekran
            movlw LCD_Alarmi_TX
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw LCD_Alarmi_RX           ;izpis alarma na LCDju?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija2           ;ni potrditve, mogoče je LCD modul
                                          ;zaposlen, bomo isto poskušali v
                                          ;naslednjem ciklu klicanja
            movlw b'00001111'             ;ok, LCD nam je potrdil sprejem
                                          ;ukaza, setiramo bit, ki to določa
            andwf alarm_na_LCD,F
            bsf   alarm_na_LCD,4          ;v višjem pol-bajtu setiramo
                                          ;4. bit: "poplava SET"(na LCDju
                                          ;sprejeto!)
komunikacija1_SMS                         ;pošljemo še na SMS modul
            movlw SMS_sporocilo_alarm_popl_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            movlw SMS_sporocilo_alarm_popl_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija1_SMS       ;zankamo, dokler ni sprejema s
                                          ;strani SMS modula
            goto  komunikacija2           ;na naslednji modul
komunikacija1_2                           ;drugi ukaz
            movlw poplava_NE_RX           ;poplava_NE?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija2
                                          ;paket je res za nas, sprožena je
                                          ;bila poplava, na LCD pošljemo:
                                          ;»LCD_Alarmi_poplava_reset« (če še
                                          ;nismo)
            movlw b'11110000'
            andwf alarm_na_LCD,F
            bsf   alarm_na_LCD,1          ;v nižjem polbajtu setiramo 1.
                                          ;bit: »poplava RESET«
            btfsc alarm_na_LCD,5          ;smo na LCD že poslali podatek, da
                                          ;je prišlo do »poplava RESET«?
            goto  komunikacija2           ;da
                                          ;ne
            bcf   PORTA,3                 ;LED za alarme ... izklopi
            movlw LCD_Alarmi_reset_TX     ;alarmi_reset
```

```
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw LCD_Alarmi_reset_RX     ;potrditev reseta alarma?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija2           ;ne
                                          ;da
            movlw b'00001111'
            andwf alarm_na_LCD,F
            bsf   alarm_na_LCD,5          ;v višjem polbajtu setiramo 5.
                                          ;bit: »poplava RESET«(na LCD-ju
                                          ;sprejeto!)
            goto  komunikacija2
```

Podobno kodo ima tudi komunikacija s *PIR modulom*: preverimo za morebitno sproženje alarma IR senzorja. Če le-ta zazna gibanje, spremeni relejni izhod na nizek nivo, mikrokontroler na *PIR modulu* pa pošlje ukaz »PIR_senzor_alarm_RX« do gospodarja. Sledi izpis na LCD zaslonu ter obvestilo SMS (pod pogojem, da je SMS modul aktiviran). Zagori tudi signalna luč ter sirena na samem *PIR modulu*, če je uporabnik to določil. Interakcijo s *PIR modulom* prikazuje primer 4.18.

Primer 4.18 MASTER modul: klicanje modulov – PIR modul

```
komunikacija2
            btfss ALARMI,1
            goto  komunikacija3
            movlw klici_PIR         ;komunikacijski bajt za povezavo s PIR
                                    ;modulom
            movwf bajt
            oddaja_serijski   bajt,TXPIR
            sprejem_serijski  bajt,RXPIR
            movlw PIR_senzor_alarm_RX     ;alarm PIR?
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_dalje               ;ne
                                          ;da
            movlw PIR_senzor_alarm_TX     ;potrdimo alarm
            movwf bajt
            oddaja_serijski   bajt,TXPIR
PIR_na_LCD
            movlw LCD_Alarm_PIR_TX        ;alarm PIR!
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw b'11110000'
            andwf alarm_na_LCD,F
```

```
            bsf   alarm_na_LCD,2          ;v nižjem polbajtu setiramo 2.
                                          ;bit: »PIR SET«
            movlw LCD_Alarm_PIR_RX        ;»alarm PIR« OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_na_LCD
            bsf   PORTA,3                 ;LED za alarme
                                          ;ok, LCD nam je potrdil sprejem
                                          ;ukaza
            movlw b'00001111'
            andwf alarm_na_LCD,F
            bsf   alarm_na_LCD,6          ;v višjem polbajtu setiramo 6.
                                          ;bit: »PIR SET«(na LCD-ju
                                          ;sprejeto!)
komunikacija2_SMS                         ;pošljemo še na SMS_modul
            movlw SMS_sporocilo_alarm_PIR_TX    ;»alarm PIR«!
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            movlw SMS_sporocilo_alarm_PIR_RX    ;je sprejel »alarm PIR«?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija2_SMS       ;zankamo, dokler ni sprejema s
                                          ;strani SMS modula (obvezen
                                          ;postopek!)
            goto  komunikacija3
PIR_dalje
            movlw PIR_senzor_OK_RX        ;ni alarma PIR
            subwf bajt,W
            btfsc STATUS,Z
            goto  komunikacija3
```

TEMP modul nam ne vrača specifičnih informacij. Vsa avtomatizacija termostata ter prilagajanje izhodnih pinov glede na spremembo temperature (npr. vklop/izklop grelnikov, peči itd.) opravi *TEMP modul* sam. Možno je sicer preveriti trenutno stanje temperature v inteligentni hiši, ter spremeniti temperaturo termostata preko SMS sporočila z uporabnikovega GSM aparata, vendar se to opravi v odseku programa, kjer komuniciramo z *SMS modulom*. Klic *SMS modula* prikazuje primer 4.19.

Primer 4.19 MASTER modul: klicanje modulov – SMS modul

```
komunikacija4
            btfss ALARMI,0
            goto  komunikacija5
            movlw klici_SMS               ;komunikacijski bajt za povezavo z
                                          ;SMS modul-om
            movwf bajt
            oddaja_serijski   bajt,TXSMS
```

```
            sprejem_serijski  bajt,RXSMS
            movlw SMS_trenutna_temp_RX    ;SMS modul potrebuje trenutno
                                          ;temperaturo v inteligentni hiši?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija4_1         ;ne
                                          ;da, najprej potrdimo sprejem
                                          ;ukaza in njegovo potrditev
            movlw SMS_trenutna_temp_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            callm tempera                 ;kličemo funkcijo za poizvedbo
                                          ;trenutne temperature
                                          ;sedaj smo sprejeli podatke,
                                          ;trenutno temperaturo imamo na
                                          ;lokaciji »temp«+[offset]
                                          ;pošljemo do SMS modula
            movlw SMS_trenutna_temp_TX    ;druga potrditev
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            movlw SMS_trenutna_temp_RX    ;je OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija5           ;ne
                                          ;da, POŠLJI! (4 bajti)
            clrf  tmp3
            movlw .4
            movwf st_temp
stiri_digiti_SMS
            movlw temp
            addwf tmp3,W
            movwf FSR
            movf  INDF,W
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            incf  tmp3,F
            decfsz      st_temp,F
            goto  stiri_digiti_SMS
            goto  komunikacija5           ;konec (gremo na naslednji modul)
komunikacija4_1
            movlw SMS_nastavitve_alarmov_RX     ;SMS_nastavitve_alarmov?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija5           ;ne
                                          ;da, sprejem novih nastavitev, ki
                                          ;smo jih sprejeli preko SMS
                                          ;sporočila
            clrf  SMS_sprememba           ;označuje, če so podatki, ki smo
                                          ;jih sprejeli od SMS modula
                                          ;drugačni od naših (tj. trenutnih
                                          ;nastavitev)
            movlw SMS_nastavitve_alarmov_TX
```

```
                                        ;SMS_nastavitve_alarmov OK! Pošlji
            movwf bajt
            oddaja_serijski   bajt,TXSMS
                                        ;dobili bomo najprej alarme
                                        ;(1. bajt), potem še nastavitve za
                                        ;termostat (2. ,3. ,4. bajt)
            sprejem_serijski  bajt,RXSMS
            movf  bajt,W
            movwf ALARMI_SMS
            movlw SMS_nastavitve_alarmov_TX     ;POTRDITEV
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            clrf  tmp3                  ;TERMOSTAT
            movlw .4
            movwf st_temp
novi_termostat
            sprejem_serijski  bajt,RXSMS
            movlw termostat             ;primerjamo s trenutno »termostat«
                                        ;nastavitvijo
            addwf tmp3,W
            movwf FSR
            movf  INDF,W
            subwf bajt,W
            btfsc STATUS,Z
            goto  naslednji_term
            bsf   SMS_sprememba,0       ;drugačna nova nastavitev za
                                        ;»termostat«
            movf  bajt,W
            movwf INDF
naslednji_term
            movlw SMS_nastavi_temp_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            incf  tmp3,F
            decfsz      st_temp,F
            goto  novi_termostat
;---------------------------------------
;NOVE NASTAVITVE ALARMOV IN TERMOSTATA
;---------------------------------------
            movf  ALARMI,W              ;sedaj še razpošljemo vse alarme
                                        ;naokrog na posamezne module
            xorwf ALARMI_SMS,W
            movwf tmp4                  ;sedaj imamo v spremenljivki
                                        ;»tmp4« spremembe alarmov (tam,
                                        ;kjer je prišlo do spremembe - na
                                        ;določenem bitu - spremenimo
                                        ;stanje modula iz »OFF->ON« ali
                                        ;obratno)
            movf  ALARMI_SMS,W
            movwf ALARMI                ;zdaj še spremenimo vrednosti v
                                        ;»ALARMI«(to so zdaj naše nove
                                        ;nastavitve)
            ;SMS-----------------------
            btfsc tmp4,0
```

```
            call  SMS_ON_OFF
            ;PIR-----------------------
            btfsc tmp4,1
            call  PIR_ON_OFF
            ;poplava-------------------
            btfsc tmp4,3
            call  poplava_ON_OFF
            ;termostat-----------------
            btfsc tmp4,4
            call  termostat_ON_OFF
            goto  komunikacija5
SMS_ON_OFF
            btfsc ALARMI,0                ;narediti moramo ravno obratno kot
                                          ;je zdaj (vrednost v registru
                                          ;"ALARMI" je že spremenjena!)
            goto  SMS_ON                  ;če je alarm nastavljen na OFF,
                                          ;damo na ON (ali obratno)
SMS_OFF
            movlw SMS_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS

            movlw SMS_IZKLOPI_VSE_RX      ;potrditev?
            subwf bajt,W
            btfss STATUS,Z
            goto  SMS_OFF                 ;pošljemo znova
            return
SMS_ON                                    ;nedefinirano stanje! SMS modul ne
                                          ;more (preko SMS sporočila) priti
                                          ;iz nedelujočega v delujoče
                                          ;stanje, saj v primeru, da ni
                                          ;vklopljen, sploh ne sprejema SMS
                                          ;sporočil
            return
PIR_ON_OFF
            movlw b'00000110'             ;1. bit - luči, 2. bit - sirena
            andwf ALARMI,W
            sublw .0
            btfss STATUS,Z
            goto  PIR_ON
PIR_OFF
            movlw PIR_senzor_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXPIR
            sprejem_serijski  bajt,RXPIR
            movlw PIR_senzor_IZKLOPI_VSE_RX     ;izklop PIR OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_OFF
            bcf   ALARMI_SMS,1
            bcf   ALARMI_SMS,1
            return
PIR_ON
```

```
            callm vklop_luci_sirena2
            bsf   ALARMI_SMS,1
            bsf   ALARMI_SMS,1
            return
poplava_ON_OFF
            btfsc ALARMI,3
            goto  poplava_ON
poplava_OFF
            movlw TEMP_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXtemp
            sprejem_serijski  bajt,RXtemp

            movlw TEMP_IZKLOPI_VSE_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  poplava_OFF
            return
poplava_ON
            callm vklop_poplava
            return
termostat_ON_OFF
            btfsc ALARMI,4
            goto  termostat_ON
termostat_OFF
            movlw TEMP_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXtemp
            sprejem_serijski  bajt,RXtemp

            movlw TEMP_IZKLOPI_VSE_RX     ;izklop termostata OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  termostat_OFF
            return
termostat_ON
            callm vklop_termostat
            return
```

SMS modul je zmožen sprejemanja in oddajanja uporabniških podatkov. Možne izmenjave informacij prikazuje tabela 4.6 v poglavju »4.2.5 SMS modul«.

Čeprav je komunikacija z *LCD modulom* vzpostavljena že prej (recimo v primeru prikaza alarma na LCD ekranu), je potrebno še preveriti, če je uporabnik izključil sistem neposredno na *LCD modulu* s pomočjo tipk in grafičnega vmesnika. Če je šifra, ki jo vnese uporabnik, pravilna, *LCD modul* počaka na poziv gospodarja, nakar pošlje ukaz za izklop sistema (»LCD_IZKLOPI_VSE_RX«). *MASTER modul* izklopi vsakega od modulov (če je

le-ta bil sploh vklopljen) ter preide v začetno, neaktivirano stanje. Komunikacijo z *LCD modulom* prikazuje primer 4.20.

Primer 4.20 MASTER modul: klicanje modulov – LCD modul

```
komunikacija5
            movlw klici_LCD               ;komunikacijski bajt za povezavo z
                                          ;LCD modulom
            movwf bajt
            oddaja_serijski   bajt,TXLCD
            sprejem_serijski  bajt,RXLCD
            movlw LCD_IZKLOPI_VSE_RX      ;če uporabnik vpiše pravilno šifro
                                          ;na LCD modulu, potem izključi
                                          ;celoten sistem inteligentne hiše
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija1           ;od znova, »klicanje POPLAVA
                                          ;modula«
            bcf   PORTA,2                 ;uporabnik je izključil sistem
            clrf  vklopljen?
            bcf   PORTA,3                 ;LED za alarme; če mogoče gori, jo
                                          ;zdaj izključimo
konec_poplava                             ;obvestimo še druge module o
                                          ;izklopu sistema (najprej POPLAVA
                                          ;modul itn.)
            btfss ALARMI,3
            goto  konec_PIR
            movlw poplava_izklop_TX
            movwf bajt
            oddaja_serijski   bajt,TXpoplava
            sprejem_serijski  bajt,RXpoplava
            movlw poplava_izklop_RX       ;izklop OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  konec_poplava           ;ne, napaka v prenosu
konec_PIR
            btfss ALARMI,1
            goto  konec_temp
            movlw PIR_senzor_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXPIR
            sprejem_serijski  bajt,RXPIR
            movlw PIR_senzor_IZKLOPI_VSE_RX     ;izklop OK?
            subwf bajt,W
            btfss STATUS,Z
            goto  konec_PIR               ;ne, napaka v prenosu
konec_temp
            btfss ALARMI,4
            goto  konec_SMS
            movlw TEMP_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXtemp
```

```
            sprejem_serijski  bajt,RXtemp
            movlw TEMP_IZKLOPI_VSE_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  konec_temp
konec_SMS
            btfss ALARMI,0
            goto  IZKLOPI_VSE             ;zaključi proces klicanja modulov
            movlw SMS_IZKLOPI_VSE_TX
            movwf bajt
            oddaja_serijski   bajt,TXSMS
            sprejem_serijski  bajt,RXSMS
            movlw SMS_IZKLOPI_VSE_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  konec_SMS
IZKLOPI_VSE
            gotom START                   ;vrnemo se na začetek
      end                                 ;EOF
```

4.2.4 LCD modul

Komunikacija uporabnika s sistemom inteligentne hiše lahko poteka na dva načina:

- preko uporabniškega vmesnika na LCD zaslonu,
- preko SMS sporočil.

Prvi način predstavlja LCD zaslon s Hitachi gonilnikom ter 4 tipke: »MENI/NAZAJ«, »OK/NAPREJ«, »GOR« in »DOL«. Diagram poteka celotnega modula lahko vidimo na diagramih od **Napaka! Vira sklicevanja ni bilo mogoče najti.** do 4.9.

Diagram 4.5 LCD modul 1/5

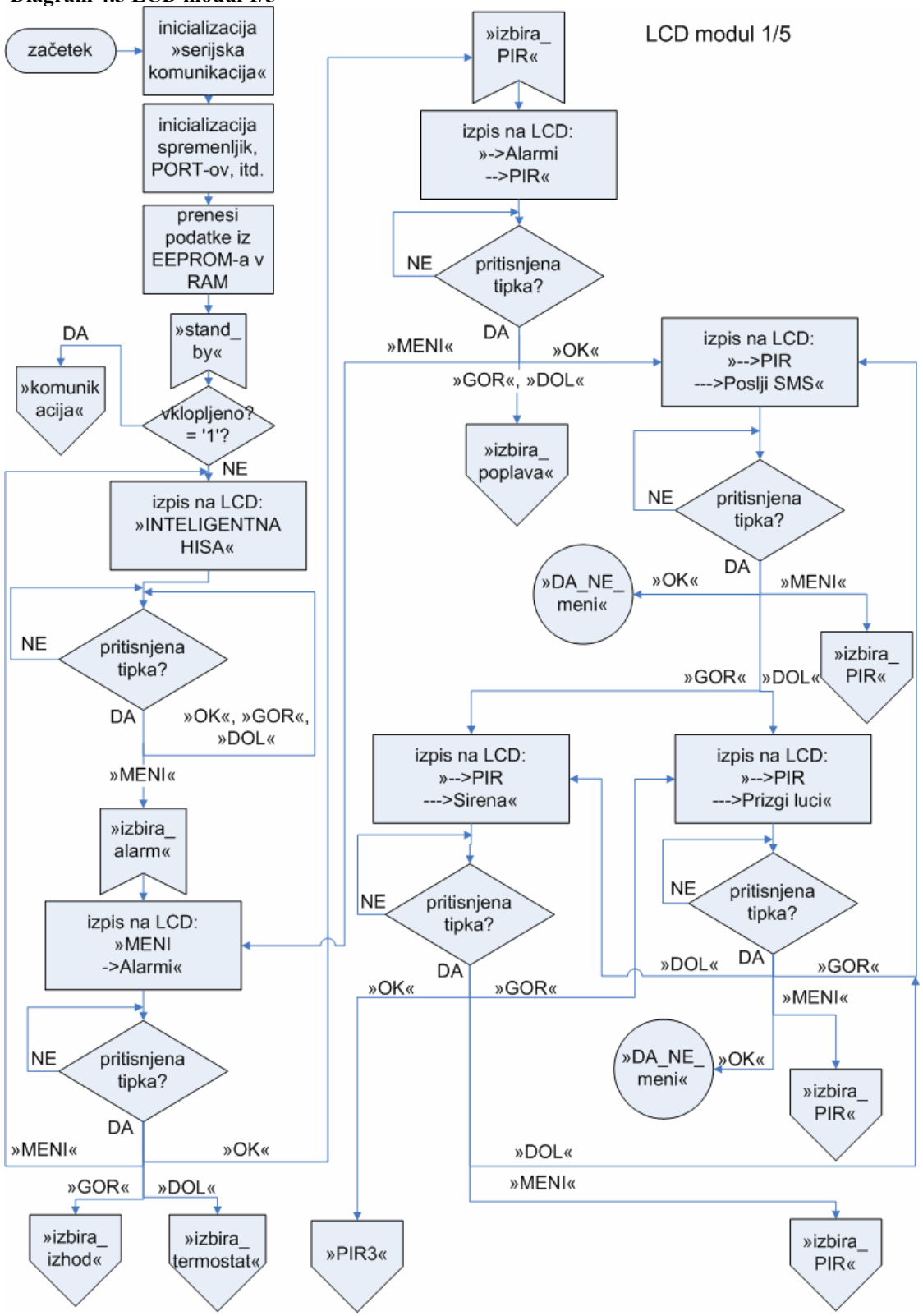


Diagram 4.6 LCD modul 2/5

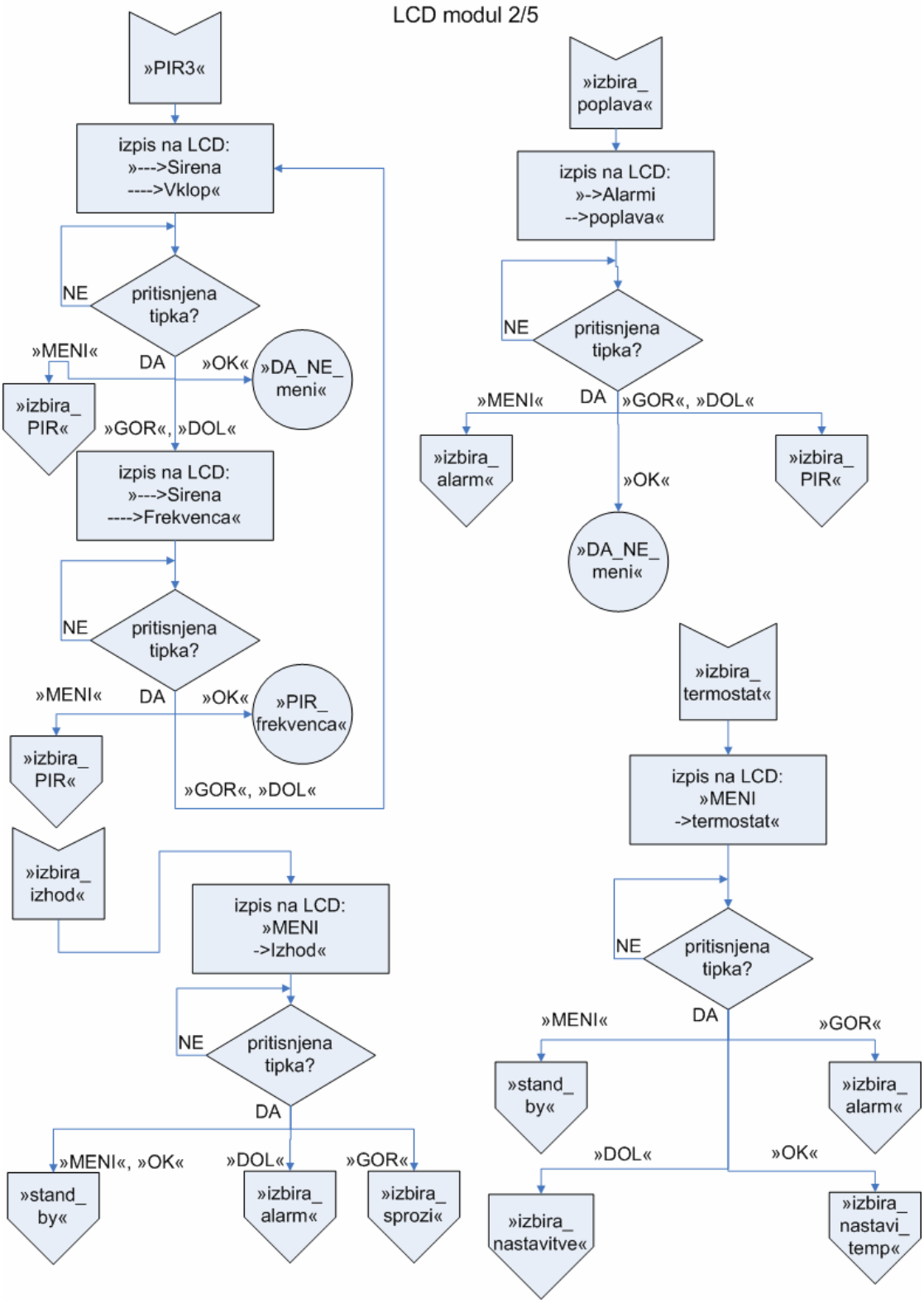


Diagram 4.7 LCD modul 3/5

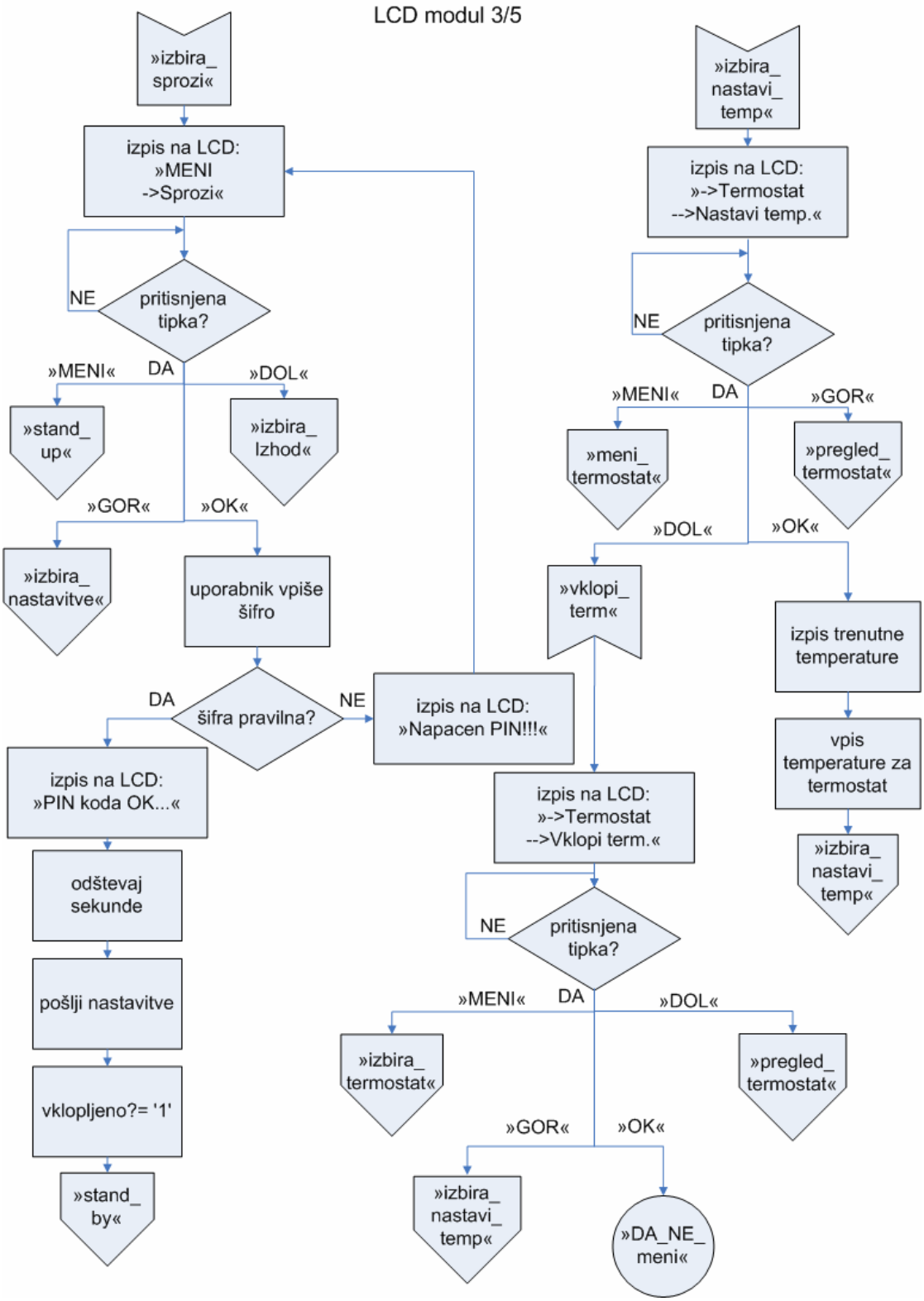


Diagram 4.8 LCD modul 4/5

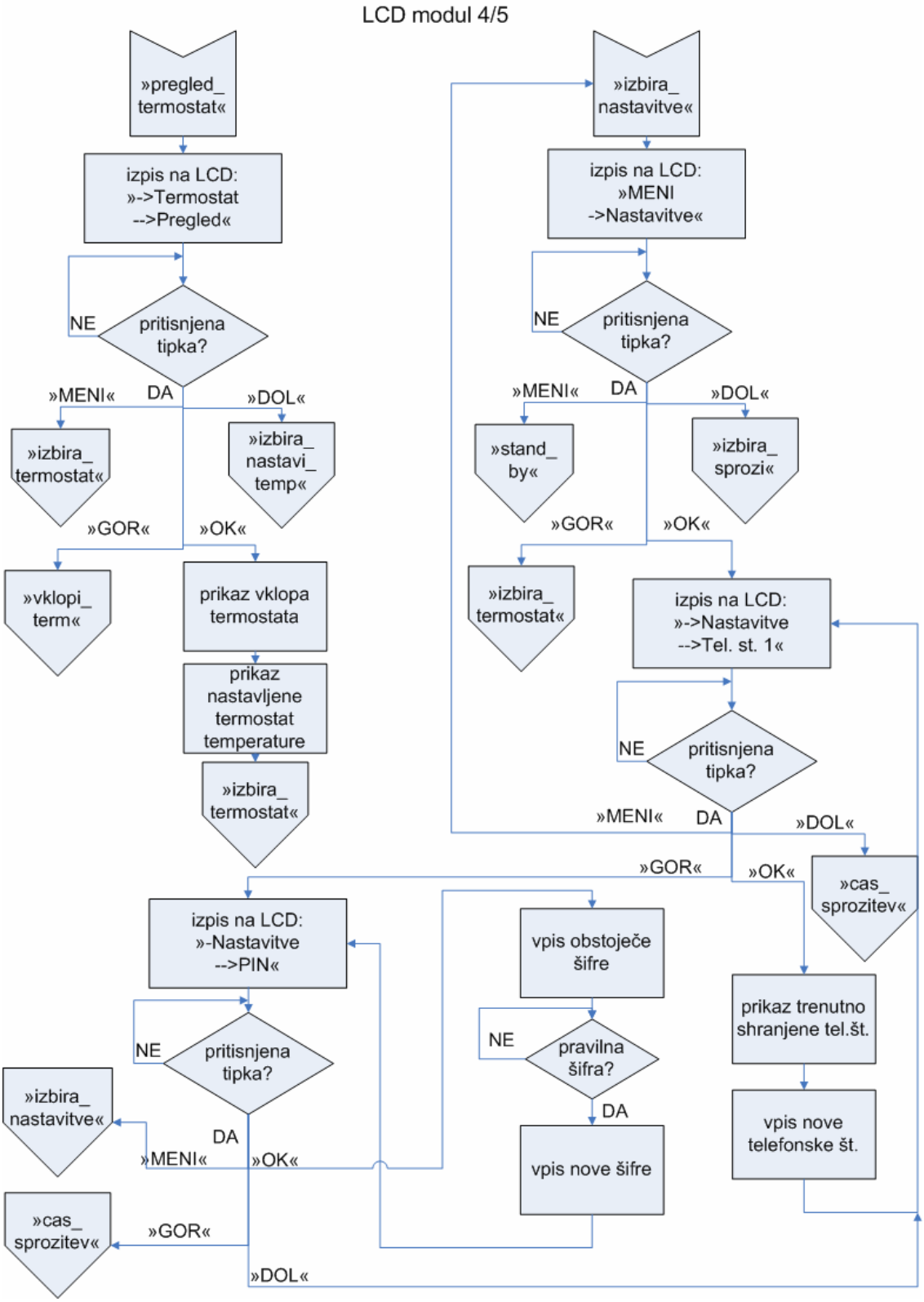
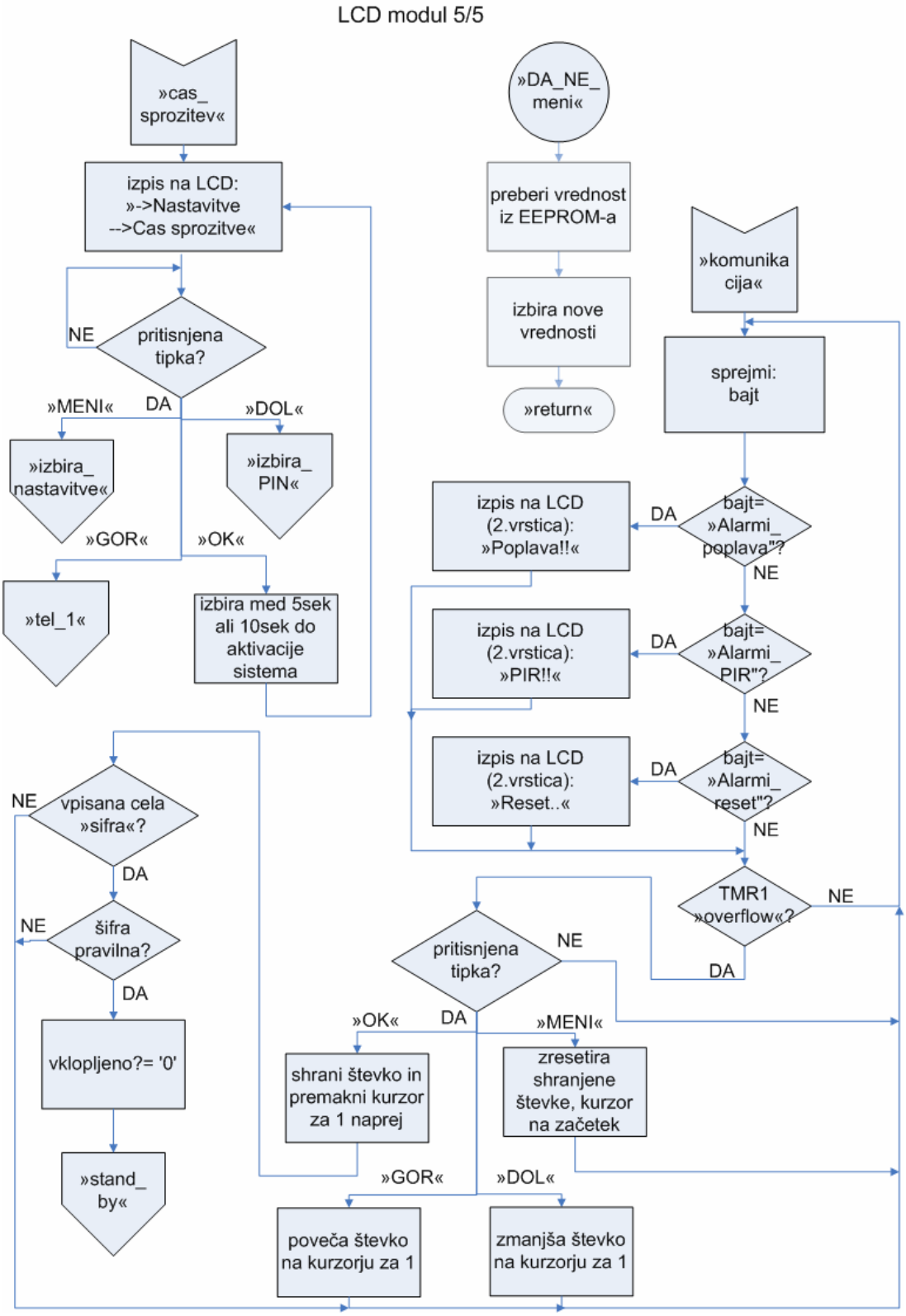


Diagram 4.9 LCD modul 5/5



S tipkami se premikamo po menijih ter vpisujemo uporabniške podatke: šifra, telefonska številka, temperatura za termostat itd. Nastavitve, ki jih uporabnik vpiše, si sistem zapomni (shranjeno v EEPROM pomnilniku *LCD modula*). V uporabljen EEPROM prostor nekatere vrednosti definiramo že v času nalaganja ».HEX« datoteke na čip, na primer PIN kodo (privzeta vrednost: »1234«). EEPROM drži lokacije od naslova 0x2100 dalje (pomembno za prevajalnik). To vidimo na primeru 4.21.

Primer 4.21 LCD modul: začetek programa

```
LIST  p=16F648a
include "P16F648a.inc"
include "zacetni_meniji_macro.inc"  ;uporabniški meniji
include "DA_NE_meni_macro.inc"      ;»DA/NE« izbirni meni
include "izbirni_meni_macro.inc"    ;makro za uporabniške menije
include "serijska_komunikacija_macro.inc" ;inicializacijski postopek
                                          ;za nastavitev pinov na
                                          ;portu (za namene serijske
                                          ;komunikacije)
include "callm_macro.inc"           ;makro za »dolge« klice
include "gotom_macro.inc"           ;makro za »dolge« skoke
include "izpis_na_LCD_macro.inc"    ;makro za izpis znakov na LCD
                                    ;ekranu

ERRORLEVEL  0,    -302, -205, -207

__CONFIG _CP_OFF & _WDT_OFF &
_PWRTE_ON & _BODEN_ON & _LVP_OFF &
_MCLRE_ON & _INTRC_OSC_NOCLKOUT

      cblock      0x20        ;začetek registrov
            count             ;uporabljeno v zankah
            count1            ;uporabljeno v zankah
            count2            ;uporabljeno v zankah
            counta            ;za časovno zakasnitev
            countb            ;za časovno zakasnitev
            tmp1              ;pomožni register 1
            tmp2              ;pomožni register 2
            tmp3              ;pomožni register 3
            templcd           ;začasni register za 4-bitni način
                              ;prenosa do LCD enkrana
            templcd2          ;pomožni register za LCD ekran
            znak              ;znak na LCD
            stolpec           ;označuje stolpec trenutnega znaka na
                              ;LCD-ju
            stolpec2          ;pomožni register za »stolpec«
            lokacija          ;lokacija v EEPROM pomnilniku (za prenos
                              ;bajtov v/iz EEPROM-a)
            podatki           ;1 bajt podatkov namenjenih za
                              ;vpis/branje iz EEPROM-a
```

```
            pomozni             ;pomožni register
            vklopljeno?         ;«vklopljeno?« =  1: sistem vklopljen
                                ;                 0: sistem izklopljen
            bajt                ;serijska komunikacija
            poizkus             ;serijska komunikacija
            pet_ali_deset_sekund    ;čas odštevanja do vklopa sistema
            PCLATH_HIGH         ;register za shranjevanje višjih bitov
                                ;PCLATH
            PCLATH_LOW          ;register za shranjevanje nižjih bitov
                                ;PCLATH
            text_zacetni        ;izpis teksta na ekranu, začenši s
                                ;tekstom na naslovu »text_zacetni«
            izhod?              ;»DA_NE_meni_macro.inc«
            crke:.16            ;splošni register za eno vrstico znakov
                                ;na LCD ekranu
            st_znakov_tel       ;binarno: število znakov telefonske
                                ;številke
            st_znakov_temp      ;pomožni register
            st_vnosov_LCD       ;določa število znakov (format), ki jih
                                ;včitujemo, glede na podatek, ki ga
                                ;potrebujemo:
                                ;           -'0'  šifra (to je PIN koda)
                                ;           -'1'  telefonska številka
                                ;           -'2'  temperatura
            text_tabela         ;določa tekstovno polje, iz katerega
                                ;beremo znake; '0' = text, '1' = text2
            text_tabela_temp
            temperatura:.4      ;register za temperaturo

                                ;format temperature:
                                ;      ---------------------------
                                ;     |1. bajt|2. bajt|3. bajt|4. bajt|
                                ;     |   x   |   y   |   z   |   q   |
                                ;      ---------------------------;
                                ;Primer: »Trenutna temperatura:xyz,q °C«
            PIN                 ; '0'=napačen, '1'=pravilen
            frekv_pomozni
            PIN_skrita          ;bit 0: '0' = ni skrito, '1' = je skrito
                                ;bit 1: '0' = stari PIN, '1' = novi PIN
            stevec              ;števec znakov
      endc

      cblock      0x60          ;EE-PROM podatki
            ee_podatki          ;sem shranimo vse nastavitve za alarme
                                ;(tudi frekvenco zvočnika - 7., 6., 5.
                                ;bit)
            telefonska:.11      ;telefonska številka v ASCII obliki
            sifra:.4            ;avtentikacija pravilnega SMS sporočila
      endc

      org   0x2100              ;pripravim željeno vrednost v EEPROM
      de .0                     ;prvo mesto EEPROM-a, »ALARMI«
                                ;telefonska številka
      de '3'
```

```
            de '8'
            de '6'
            de '4'
            de '0'
            de '5'
            de '6'
            de '7'
            de '6'
            de '9'
            de '2'
                                    ;šifra
            de '1'
            de '2'
            de '3'
            de '4'
                                    ;termostat, privzeto: "020,0" °C
            de '0'
            de '2'
            de '0'
            de '0'
```

Mnemoniki ukazov za serijsko komunikacijo so enaki kot v primeru gospodarja, le da se tukaj zamenjata končnici »_RX« in »_TX«, ki označujeta dohodni oziroma odhodni ukaz.

LCD zaslon pred uporabo inicializiramo (`pripravi_LCD`), sosledje ukazov in rutin kaže primer 4.22.

Primer 4.22 LCD modul: LCD komande

```
;sopomenke
LCD_PORT    EQU   PORTA             ;PORT za LCD
LCD_TRIS    EQU   TRISA
LCD_RS      EQU   0x04              ;»Register Select«
LCD_RW      EQU   0x06              ;»Read/Write«
LCD_E       EQU   0x07              ;»Enable«

;rutine
pripravi_LCD
            call  Delay20           ;počakamo, da se LCD stabilizira
            call  LCD_Init          ;namestitev LCD-ja
            call  LCD_CurOff        ;izključi kurzor
            return
LCD_Init                            ;INICIALIZACIJA LCD
            movlw 0x20              ;nastavi »4 bit mode«
            call  LCD_Cmd
            movlw 0x28              ;nastavi »display shift«
            call  LCD_Cmd
            movlw 0x06              ;nastavi »display character mode«
            call  LCD_Cmd
```

```
            movlw 0x0d              ;nastavi vklop zaslona in kurzor
            call  LCD_Cmd
            call  LCD_Clr           ;pobriši ekran
            retlw 0x00              ;vrni se z vrednostjo '0' v W
LCD_Cmd                             ;POŠLJI KOMANDO
            movwf templcd
            swapf templcd,w         ;pošlji zgornji polbajt
            andlw 0x0f              ;pobriši zgodnji polbajt
            movwf LCD_PORT
            bcf   LCD_PORT, LCD_RS  ;RS linija na '0'
            call  Pulse_e           ;onemogoči komunikacijo (E = visok nivo)
            movf  templcd,w         ;pošlji spodnji polbajt
            andlw 0x0f              ;pobriši zgornji polbajt
            movwf LCD_PORT
            bcf   LCD_PORT, LCD_RS  ;RS linija na '0'
            call  Pulse_e
            call  Delay5            ;zakasnitev 5 ms (pri urinem taktu
                                    ;4 MHz)
            retlw 0x00
LCD_Char                            ;POŠLJI ZNAK NA ZASLON
            swapf templcd, w
            andlw 0x0f
            movwf LCD_PORT
            bsf   LCD_PORT, LCD_RS
            call  Pulse_e
            movf  templcd, w
            andlw 0x0f
            movwf LCD_PORT
            bsf   LCD_PORT, LCD_RS
            call  Pulse_e
            call  Delay5
            retlw 0x00
LCD_Cursor_ON
            movlw 0x0f              ;kurzor vklopi!
            call  LCD_Cmd
            retlw 0x00
LCD_Line1
            movlw 0x80              ;premakni se v prvo vrstico, prvi znak
            call  LCD_Cmd
            retlw 0x00
LCD_Line2
            movlw 0xc0              ;premakni se v prvo vrstico, prvi znak
            call  LCD_Cmd
            retlw 0x00
LCD_Line1W
            movf  templcd,W
            addlw 0x80              ;premakni se v prvo vrstico, W-ti znak
            call  LCD_Cmd
            retlw 0x00
LCD_Line2W
            movf  templcd,W
            addlw 0xc0              ;premakni se v drugo vrstico, W-ti znak
            call  LCD_Cmd
            retlw 0x00
```

```
LCD_Clr_Line1                       ;pobriši prvo vrstico
            call  LCD_Line1
            movlw ' '
            movwf templcd
            movlw .15               ;pobrišemo 16 znakov (oziroma
                                    ;nadomestimo s SPACE-i)
            movwf count2
ponovitev1
            call  LCD_Char
            decfsz      count2,F
            goto  ponovitev1
            call  LCD_Line1         ;s kurzorjem nazaj v prvi stolpec
            retlw 0x00
LCD_Clr_Line2
            call  LCD_Line2
            movlw ' '
            movwf templcd
            movlw .15
            movwf count2
ponovitev2
            call  LCD_Char
            decfsz      count2,F
            goto  ponovitev2
            call  LCD_Line2
            retlw 0x00
LCD_CurOn
            movlw 0x0d              ;ekran vklop/izklop & kurzor
            call  LCD_Cmd
            retlw 0x00
LCD_CurOff
            movlw 0x0c              ;ekran vklop/izklop & kurzor
            call  LCD_Cmd
            retlw 0x00
LCD_Clr
            movlw 0x01              ;pobriši ekran
            call  LCD_Cmd
            retlw 0x00
```

Po dvigu na delovno napetost LCD ekran potrebuje določen čas za nastavitev. Tudi za prikaz znaka, sprejem ali oddajo inštrukcije se kaže ta zahteva. Zakasnitve so definirane v dokumentaciji za vsak tip LCD ekrana posebej, za naš primer imamo naslednje omejitve[16]:

- čakaj vsaj 15 ms od trenutka, ko Vcc = 4.5 V,
- čakaj vsaj 4.1 ms med včitavanjem zgornjega in spodnjega polbajta, če želimo zgolj 4-bitni vmesnik,

- čakaj vsaj 100 μs po izbiri vmesnika (kasneje lahko »zasedenost« gonilnika preverimo z BF – »Busy Flag«).

Programsko smo zakasnitve rešili kot prikazuje primer 4.23. Uporabili smo jih še pri drugih rutinah (merjenje časa držanja tipke, prikaza znaka na zaslonu, odštevanje do sprožitve sistema inteligentne hiše).

Primer 4.23 LCD modul: časovne zakasnitve

```
Delay1s                             ;zakasnitev: 1 s
            call  Delay255
            call  Delay255
            call  Delay255
            call  Delay255
            retlw 0x00
Delay255
            movlw 0xff              ;zakasnitev: 255 ms
            goto  d0
Delay100
            movlw d'100'            ;zakasnitev: 100 ms
            goto  d0
Delay50                             ;zakasnitev: 50 ms
            movlw d'50'
            goto  d0
Delay20                             ;zakasnitev: 20 ms
            movlw d'20'
            goto  d0
Delay5                              ;zakasnitev: 5 ms
            movlw 0x05
d0
            movwf count1
d1
            movlw 0xC7              ;zakasnitev: 1 ms
            movwf counta
            movlw 0x01
            movwf countb
Delay_0
            decfsz      counta, f
            goto  $+2               ;skok na naslov PC+2
            decfsz      countb, f
            goto  Delay_0

            decfsz      count1, f
            goto  d1
            retlw 0x00
Pulse_e                             ;»ENABLE« pulz
            bsf   LCD_PORT, LCD_E
            nop
            bcf   LCD_PORT, LCD_E
```

```
        retlw 0x00
```

Uporabniški meniji zavzemajo naslednjo hierarhijo:

```
MENU
     ->Alarmi
          -->PIR
               --->Poslji SMS
               --->Prizgi luci
               --->Sirena
                    ---->Vklop
                    ---->Frekvenca
          -->Poplava
               --->Vklop
     ->Termostat
          -->Nastavi temp.
          -->Vklopi term.
          -->Pregled
     ->Nastavitve
          -->Tel.st. 1
          -->Cas sprozitve
          -->PIN
     ->Sprozi
     ->Izhod
```

Na zaslonu se lahko hkrati izpišeta 2 vrstici po 16 znakov, zato smo temu prilagodili tudi programsko kodo izpisa. Za izpis menija kot na sliki 4.2 smo v zbirniku spisali makro, prikazan v primeru 4.24 z vstopnimi argumenti `izbirni_meni Alarm_text, PIR_text, izbira_poplava, izbira_poplava, PIR, izbira_alarm`.

Slika 4.2 LCD modul: Primer menija

Primer 4.24 LCD modul: makro za menije, datoteka »izbirni_meni_makro.inc«

```
izbirni_meni      macro prvi_text,drugi_text,naprej,nazaj,enter,cancel

                                    ;meni je sestavljen po nalsednjem
                                    ;sistemu:
                                    ;     - prvi_text: besedilo napisano v
                                    ;     zgornji vrstici,
                                    ;     - drugi_text: besedilo napisano v
                                    ;     spodnji vrstici,
                                    ;     - naprej: premik na naslednji
                                    ;     »izbirni_meni« ob pritisku na
                                    ;     tipko »DOL«,
                                    ;     - nazaj: premik na prejšnji
                                    ;     »izbirni_meni« ob pritisku na
                                    ;     tipko »GOR«
                                    ;     - enter: premik na nižji
                                    ;     »izbirni_meni« ob pritisku na
                                    ;     tipko »OK«
                                    ;     - cancel: premik na višji
                                    ;     »izbirni_meni« ob pritisku na
                                    ;     tipko »MENI«
Local krozi1                        ;definicija lokalnih oznak (angl.
                                    ;»label«)
Local naprej1
Local nazaj1
Local enter1
Local cancel1

            callm LCD_Clr           ;pobriši ekran
            movf  text_tabela,W
            movwf text_tabela_temp
            clrf  text_tabela       ;»text_tabela« označuje branje znakov iz
                                    ;prvega odseka besedil
            izpis_na_LCD      sklop_1,prvi_text,1
            izpis_na_LCD      sklop_1,drugi_text,2
            movf  text_tabela_temp,W
            movwf text_tabela
```

```
            callm Delay255          ;0.5 s zakasnitve, da ne pride do
                                    ;nehotenega pritiska na tipko
            callm Delay255
krozi1
            btfss PORTB,0           ;«DOL«
            goto  naprej
            btfss PORTB,1           ;«GOR«
            goto  nazaj
            btfss PORTB,2           ;«OK«
            goto  enter
            btfss PORTB,3           ;«MENI«
            goto  cancel
            goto  krozi1            ;ponavljaj v nedogled, dokler ni
                                    ;pritisnjena ena od tipk
            endm
```

Primer 4.25 LCD modul: makro za izpis na LCD, datoteka »izpis_na_LCD_makro.inc«

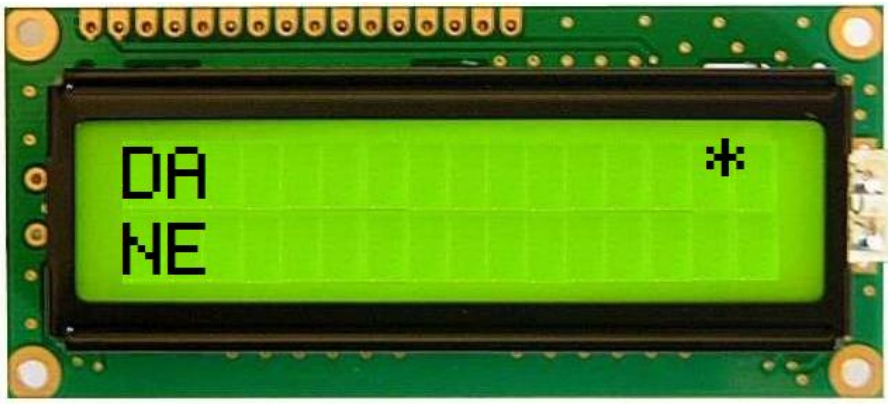
```
izpis_na_LCD      macro predloga,zacetek,vrstica
                                    ;Argumenti:
                                    ;     - predloga: predloga za branje
                                    ;     besedil
                                    ;     - zacetek: začetek besedila v
                                    ;     predlogi
                                    ;     - vrstica: prva ali druga vrstica
            if          vrstica==1  ;»if« preprocesorska direktiva
                  callm LCD_Line1
            else
            if          vrstica==2
                  callm LCD_Line2
            endif
            endif

            movlw zacetek
            movwf text_zacetni
            callm predloga          ;pokličemo predlogo za izpis niza
            endm
```

Nizi besedil so shranjeni na začetnih lokacijah FLASH pomnilnika, naslovi od 0x005 do 0x1F8. Dosežemo jih z odmikom (angl. »offset«) s pomočjo programskega števnika. Ker je programski števnik le 8-biten, lahko naslovi največ 256 lokacij (vsaka lokacija premore 1 ASCII znak), zato smo nize segmentirali na 2 bloka: `text` in `text2`. Iz katerega bloka bomo brali znake, določa `text_tabela` (vrednost '0' za `text` in '1' za `text2`).

Primer 4.26 LCD modul: branje znakov iz bloka pomnilnika

```
sklop_1
            clrf  count                   ;dostop do posameznega znaka v
                                          ;nizu
            clrw                          ;pobriši splošni register W
zanka
            incf  count,F
            movf  count,W
            addwf text_zacetni,W
            btfsc text_tabela,0           ;beremo iz prvega bloka nizov?
            goto  naslednja_text          ;ne
            addlw text                    ;da
            movwf tmp3
            movlw HIGH text
            movwf PCLATH
            movf  tmp3,W
            call  text                    ;iz funkcije se vrnemo z (ASCII)
                                          ;znakom v splošnem registru W
            goto  preveri_tabelo
naslednja_text
            addlw LOW text2               ;Inštrukcijski register poskuša
                                          ;vpisati ves naslov »text2«
                                          ;(0x0100), vendar je le 8-bitni
                                          ;register, zato vpišemo le 8
                                          ;najmanj obteženih bitov
            movwf tmp3
            movlw HIGH text2              ;v PCLATH še vpišemo višje bite
                                          ;naslova registra
            movwf PCLATH
            movf  tmp3,W
            call  text2
preveri_tabelo
            xorlw 0x00                    ;smo prišli do 0?
            btfsc STATUS, Z
            goto  konec                   ;da
            movwf templcd                 ;ne, pošljemo trenutni znak na LCD
            call  LCD_Char
            goto  zanka                   ;zankaj
konec
            return
```

Najgloblji meniji vedno posredujejo ali sprejmejo informacijo uporabnika. Primer tega je vklop/izklop želenih modulov. Izberemo »DA« za vklop in »NE« za izklop. Makro prikazuje primer 4.27.



Slika 4.3 LCD modul: »DA/NE« meni

Primer 4.27 LCD modul: vklop/izklop modulov, datoteka »DA_NE_meni_macro.inc«

```
DA_NE_meni macro  ID_alarma,vrni_se
                                    ;Argumenti:
                                    ;     - ID_alarma: bit v registru
                                    ;     »ALARMI« (za vsak modul posebej)
                                    ;     0. bit: SMS modul,
                                    ;     1. bit: PIR modul – luči,
                                    ;     2. bit: PIR modul – sirena,
                                    ;     3. bit: POPLAVA modul,
                                    ;     4. bit: TERMSOTAT modul.
                                    ;     - vrni_se: skok na to oznako po
                                    ;     izbiri
Local abc                           ;lokalne oznake
Local abc2
Local abc3
Local abc4
Local naprej
            callm izpis_DA_NE       ;izpiši »DA« in »NE« na zaslon
            btfss ee_podatki,ID_alarma
                                    ;pogledamo trenutno stanje alarma, ki je
                                    ;vpisano v EEPROM-u PIC-a
            goto  naprej            ;»NE«
            callm kvadrat_gor       ;»DA«
naprej
            btfsc ee_podatki,ID_alarma
            goto  abc
            callm kvadrat_dol       ;»NE«









abc
```

```
            btfsc PORTB,0
            goto  abc2
            callm kvadrat_dol
            bcf   tmp1,ID_alarma    ;v »tmp1« začasno shranimo vrednosti
                                    ;izbire alarma (kasneje jo vpišemo v
                                    ;EEPROM, če uporabnik izbere »OK«)
abc2
            btfsc PORTB,1
            goto  abc3
            callm kvadrat_gor       ;"DA"
            bsf   podatki,ID_alarma
abc3
            btfsc PORTB,2           ;uporabnik pritisnil »OK«?
            goto  abc4              ;ne
            clrf  lokacija          ;da, vpišemo v EEPROM
                                    ;»lokacija« predstavlja lokacijo v
                                    ;EEPROM-u (na poziciji 0. so »ALARMI«)
            callm VPIS_EE
            movf  podatki,W
            movwf ee_podatki
DA_NE_meni
            goto  vrni_se
abc4
            btfsc PORTB,3           ;uporabnik pritisnil »MENI«?
            goto  abc               ;ne, preverjamo dalje za morebitni
                                    ;pritisk tipke
            goto  vrni_se           ;da
            endm
```

Trajno shranjevanje podatkov v EEPROM izvede procedura `VPIS_EE`. Za vsak nov vnos je potrebna posebna sekvenca v programu (določena v Microchipovih navodilih[17]) ter 10 ms zakasnitve.

Primer 4.28 LCD modul: vpis v EEPROM

```
VPIS_EE
      movf  podatki,W
      bsf   STATUS,RP0              ;treba je preklopiti na »bank1«, saj
                                    ;se registri za upravljanje z EEPROM-om
                                    ;nahajajo tukaj
      ;   bcf     INTCON,GIE        ;izklop vseh prekinitev, če bi jih
                                    ;uporabljali
      ;   btfsc   INTCON,GIE
      ;   goto    $-2               ;so res izključene?
      movwf EEDATA                  ;EEPROM podatek
      bcf   STATUS,RP0
      movf  lokacija,W
      bsf   STATUS,RP0
      movwf EEADR                   ;EEPROM lokacija
```

```
      bsf   EECON1,WREN             ;omogočim vpis
      movlw 0x55                    ;začetek »POSEBNA SEKVENCA«
      movwf EECON2
      movlw 0xAA
      movwf EECON2                  ;konec »POSEBNA SEKVENCA«
      bsf   EECON1,WR               ;izvedem vpis
      btfsc EECON1,WR               ;je vpis končan?
      goto  $-1                     ;ne, preveri ponovno
      ;   bsf     INTCON,GIE        ;da, vključim prekinitve
      bcf   STATUS,RP0              ;nazaj v »bank0«
      return
```

Branje EEPROM-a ne zahteva časovne zakasnitve. Proizvajalec priporoča proceduro osveževanja (angl. »refresh procedure«) v primeru mnogokratnega branja EEPROM-a (z branjem se izgubi določen delež elektronov, na dolgi rok pomeni to nedoločljivo stanje registra). Ker podatke beremo le ob vklopu sistema, aktivaciji sistema in preveritvi v menijih, le-tega nismo uporabili. Branje enega bajta podatkov iz EEPROM lokacije `lokacija` prikazuje primer 4.29.

Primer 4.29 LCD modul: branje iz EEPROM-a

```
BERI_EE
      movf  lokacija,W
      bsf   STATUS,RP0              ;»bank1«
      movwf EEADR                   ;lokacija v EEPROM-u
      bsf   EECON1,RD               ;izvedem branje
      movf  EEDATA,W
      movwf podatki                 ;shranimo v register »podatki«
      bcf   STATUS,RP0              ;»bank0«
      return
```

Šifro, telefonsko številko in temperaturo termostata vpišemo z `vpis_na_LCD`. Šifra dovoljuje 4 številke (maskirane z *), telefonska 11 ter termostat 4 (formatirano v »*xyz*,`q`°C«). Tip vnosa, ki ga obravnavamo, označuje register `st_vnosov_LCD`.

S tipkama »GOR« (vstop v proceduro `naprej`) in »DOL« (vstop v proceduro `nazaj`) inkrementiramo/dekrementiramo vrednost na ekranu, od '0' do '9' ali obratno. Z »OK« (vstop v proceduro `naslednji`) sledi skok na naslednjo številko, torej shranimo znak ter

premaknemo kurzor na LCD-ju za ena naprej. Tipka »MENI« prekine vpis in vrača v izbirne menije.

Primer 4.30 LCD modul: funkcija »vpis_na_LCD«

```
vpis_na_LCD
            clrf  count             ;nastavi števni register na 0
            clrf  stolpec           ;nastavi trenutni stolpec na LCD ekranu
                                    ;na 0
            btfss PORTB,2
            goto  $-1               ;če imamo še vedno pritisnjeno tipko za
                                    ;povečanje kurzorja (»OK«), počakamo, da
                                    ;se sprosti
Message2
            movf  st_vnosov_LCD,W   ;šifra?
            sublw .0
            btfsc STATUS,Z
            goto  pokazi_sifra
            movf  st_vnosov_LCD,W   ;telefonska številka?
            sublw .1
            btfsc STATUS,Z
            goto  pokazi_telefonska
            movf  st_vnosov_LCD,W   ;temperatura?
            sublw .2
            btfsc STATUS,Z
            goto  pokazi_temp
;---------------------
;vpis šifre (PIN koda)
;---------------------
pokazi_sifra
            movlw .4                ;preverimo, če smo že vpisali 4 znake v
                                    ;register »sifra«
            subwf stolpec,W
            btfsc STATUS,Z
            goto  finish
            goto  pokazi_znake
;---------------------
;vpis telefonske št.
;---------------------
pokazi_telefonska
            movlw .11               ;preverimo, če smo že vpisali 11 znakov
                                    ;v register »telefonska« (kot recimo
                                    ;»38640567692«)
            subwf stolpec,W
            btfsc STATUS,Z
            goto  finish
            goto  pokazi_znake
;---------------------
;vpis temperature
;---------------------
pokazi_temp
```

```
            movlw .4                ;preverimo, če smo že vpisali vse 4
                                    ;znake (recimo: »020,0«°C)
            subwf stolpec,W
            btfsc STATUS,Z
            goto  finish
            goto  pokazi_znake
;--------------------
pokazi_znake
            movf  st_vnosov_LCD,W
            sublw .2                ;preverimo za temperaturo
            btfss STATUS,Z
            goto  ni_temp           ;ni temperatura
                                    ;je temperatura
                                    ;preverimo, če smo prišli do tretje
                                    ;številke za temperaturni vnos. Če DA,
                                    ;vrinemo vejice
            movlw .3
            subwf stolpec,W
            btfss STATUS,Z
            goto  ni_temp
            movlw ','
            movwf templcd
            call  LCD_Char
ni_temp
            movlw '0'               ;začnemo z '0'
            movwf znak
            movwf templcd
            call  LCD_Char
zanka0
            btfsc PORTB,1
            goto  zanka1
            call  naprej
zanka1
            btfsc PORTB,0
            goto  zanka2
            call  nazaj
zanka2
            btfsc PORTB,2
            goto  sistem_ON_prispel_alarm?
            goto  naslednji
sistem_ON_prispel_alarm?            ;neprestano preverjamo za morebitnimi
                                    ;alarmi od MASTER modul-a
            movlw .1
            subwf vklopljeno?,W     ;je sistem vklopljen?
            btfss STATUS,Z
            goto  zanka0            ;ne
                                    ;da, sistem je vključen, preverimo
                                    ;MASTER_modul, če potrebuje našo
                                    ;pozornost
            call  sprejem_serijski
            movlw javi_LCD          ;splošni paket sužnja »LCD modul«
            movwf bajt
            call  oddaja_serijski
            goto  zanka0
```

```
;-------------------------------------------------------------------------
;rutina,ki premakne kurzor na LCD-ju za ena naprej ter shrani znak v »crke«
;-------------------------------------------------------------------------
naslednji
            movf  stolpec,W         ;shranimo vrednost v splošni register W
            movwf FSR               ;register za indirektno naslavlanje
            movlw crke
            addwf FSR,F             ;dodamo še pot do začetnega naslova za
                                    ;»crke«
            movf  znak,W
            movwf INDF              ;na naslov »crke« vpišemo prvo črko, na
                                    ;naslov »crke«+1 drugo itd.
                        ;če da, potem še "znak" vpišemo na lokacijo
"telefonska"
            movf  st_vnosov_LCD,W   ;preverimo, če mogoče vpisujemo
                                    ;telefonsko številko, če da, potem še
                                    ;»znak« vpišemo na pomnilniško lokacijo
                                    ;»telefonska« (za kasnejšo obdelavo)
            sublw .1                ;telefonska?
            btfss STATUS,Z
            goto  ni_telefonska     ;ni telefonska št.
            movf  stolpec,W         ;je telefonska št.
            movwf FSR
            movlw telefonska
            addwf FSR,F
            movf  znak,W
            movwf INDF
ni_telefonska                       ;preverimo, če mogoče vpisujemo
                                    ;temperaturo, če da, potem še »znak«
                                    ;vpišemo na lokacijo »temperatura«
            movf  st_vnosov_LCD,W   ;temperatura?
            sublw .2
            btfss STATUS,Z
            goto  ni_temperatura    ;ne
            movf  stolpec,W         ;da
            addlw temperatura
            movwf FSR
            movf  znak,W
            movwf INDF
ni_temperatura                      ;preverimo, če je potrebno zadnji znak
                                    ;skriti (recimo pri šifri)
            btfss PIN_skrita,0
            goto  ni_skrito
            movf  vklopljeno?,W
            xorlw 0x00
            btfsc STATUS, Z
            goto  ni_temperatura2   ;sistem ni vklopljen. skok na
                                    ;»ni_temperatura2«
            movf  stolpec,W         ;sistem je vklopljen!
            addlw .4                ;odmik zaradi besedila pred tem (»PIN:«)
            movwf templcd
            call  LCD_Line1W
            movlw '*'
            movwf templcd
```

```
            call  LCD_Char
            incf  stolpec,F         ;gremo v naslednji stolpec na LCD
                                    ;zaslonu
            return
ni_temperatura2
            movf  stolpec,W
            movwf templcd
            call  LCD_Line2W
            movlw '*'
            movwf templcd
            call  LCD_Char
ni_skrito                           ;trenutne številke ne rabimo skriti
            incf  stolpec,F         ;gremo v naslednji stolpec na LCD
                                    ;zaslonu
            call  Delay1s
            goto  Message2          ;gremo na naslednjo črko v nizu
finish                              ;zaključimo z vnosom, uporabnik je
                                    ;vpisal dovoljšnjo količino števk
            call  LCD_CurOff
            movf  st_vnosov_LCD,W   ;telefonska št.?
            sublw .1
            btfss STATUS,Z
            goto  finish_termostat
finish_telefonska
            call  LCD_Clr
            movf  text_tabela,W     ;začasno shranimo vrednost v
                                    ;«text_tabela_temp« (saj bomo izpisali
                                    ;niz »Nast. shranjena!«)
            movwf text_tabela_temp
            movlw .1
            movwf text_tabela
            movlw Nast_shranjena_text
            movwf text_zacetni
            call  sklop_1
            call  Delay1s           ;počakamo 1 sekundo z izpisom na zaslonu
            movf  text_tabela_temp,W      ;nazaj postavimo vrednost iz
                                    ;»text_tabela«
            movwf text_tabela
            goto  finish2
finish_termostat                    ;termostat, vpišemo na EEPROM lokacije
            movf  st_vnosov_LCD,W
            sublw .2
            btfss STATUS,Z
            goto  finish2
            movlw .16               ;odmik za lokacijo »termostat« v
                                    ;EEPROM-u
            movwf tmp2
            movlw .4
            movwf count
            movlw temperatura
            movwf FSR
termostat_zanka
            movf  INDF,W
            movwf tmp1
```

```
            call  v_eeprom
            incf  tmp2,F
            incf  FSR,F
            decfsz      count,F
            goto  termostat_zanka
finish2
            return
;-------------------------------------------------------------------------
;rutina,ki pobriše številko na trenutnem kurzorju ter vpiše za ena višjo
številko
;-------------------------------------------------------------------------
naprej                              ;izpišemo naslednji znak
                                    ;(1, 2, 3, ... 9, 0, 1 ...), če pridemo
                                    ;do znaka '9', potem zarotiramo naprej
                                    ;na znak '0', saj želimo samo številke
            incf  znak,F
            movlw .58               ;':' znak, ki je v ASCII tabeli za '9'
                                    ;ko ga dosežemo, začnemo spet pri '0'
            subwf znak,W
            btfss STATUS,Z
            goto  stevke_naprej
            movlw '0'
            movwf znak
stevke_naprej
            movf  stolpec,W         ;vrtimo se na istem kurzorju
            movwf templcd
            movf  st_vnosov_LCD,W   ;telefonska št.?
            sublw .1
            btfss STATUS,Z
            goto  ni_telefonska2
            movlw .5                ;dodamo dodatni odmik k poziciji
                                    ;kurzorja na LCD-ju, zaradi napisa
                                    ;»Novo:« (5 znakov)
            addwf templcd,F
ni_telefonska2
            movf  st_vnosov_LCD,W   ;temperatura?
            sublw .2
            btfss STATUS,Z
            goto  ni_temp2
            movlw .9                ;dodamo dodatni odmik k poziciji
                                    ;kurzorja na LCD-ju, zaradi napisa
                                    ;»Nastavi :« (9 znakov)
            addwf templcd,F
            movlw .3
            subwf stolpec,W
            btfsc STATUS,Z
            incf  templcd,F
ni_temp2
            movf  vklopljeno?,W     ;sistem vklopljen?
            xorlw 0x00
            btfsc STATUS, Z
            goto  ni_temp_izklopljen      ;ne
            movf  stolpec,W               ;da
            addlw .4
```

```
            movwf templcd
            call  LCD_Line1W
            movf  znak,W
            movwf templcd
            call  LCD_Char
            return
ni_temp_izklopljen                  ;sistem je še izklopljen (zato drugačna
                                    ;procedura)
            call  LCD_Line2W
            movf  znak,W
            movwf templcd
            call  LCD_Char
            call  Delay255          ;zakasnitev 0.5 s
            call  Delay255
            return
;-------------------------------------------------------------------------
;rutina,ki pobriše številko na trenutnem kurzorju ter vpiše za ena manjšo
številko
;-------------------------------------------------------------------------
nazaj                               ;če pridemo do znaka '0', potem
                                    ;zarotiramo nazaj na znak '9', ker
                                    ;želimo samo številke
            decf  znak,F
            movlw .47               ;'/' (ASCII, 47 decimalno)
            subwf znak,W
            btfss STATUS,Z
            goto  stevke_nazaj
            movlw '9'               ;zarotirajmo nazaj na znak '9'
            movwf znak
stevke_nazaj
            movf  stolpec,W         ;vrtimo se na istem kurzorju
            movwf templcd
            movf  st_vnosov_LCD,W   ;telefonska št.?
            sublw .1
            btfss STATUS,Z
            goto  ni_telefonska3
            movlw .5                ;dodamo dodatni offset k poziciji
                                    ;kurzorja na LCD-ju, zaradi napisa
                                    ;»Novo:« (5 znakov)
            addwf templcd,F
ni_telefonska3
            movf  st_vnosov_LCD,W
            sublw .2
            btfss STATUS,Z
            goto  ni_temp3
            movlw .9                ;»Nastavi :« (= 9 znakov)
            addwf templcd,F
            movlw .3
            subwf stolpec,W
            btfsc STATUS,Z
            incf  templcd,F
ni_temp3
            movf  vklopljeno?,W     ;Sistem inteligetne hiše vklopljen?
            xorlw 0x00
```

```
            btfsc STATUS, Z
            goto  ni_temp_izklopljen3     ;izklopljeno!
                                          ;vklopljeno!
            movf  stolpec,W
            addlw .4
            movwf templcd
            call  LCD_Line1W
            movf  znak,W
            movwf templcd
            call  LCD_Char
            return
ni_temp_izklopljen3
            call  LCD_Line2W
            movf  znak,W
            movwf templcd
            call  LCD_Char
            call  Delay255
            call  Delay255
            return
```

Dokler je sistem neaktiven, ima uporabnik možnost vpogleda v stanje trenutne temperature na zaslonu (ko se sistem enkrat aktivira, je le-to možno zgolj preko SMS sporočila, saj predvidevamo, da uporabnika ni v inteligentni hiši). Nastavitve temperature uporabnik doseže v podmeniju `MENU-> Termostat--> Nastavi temp.`. *LCD modul* sprejme 4 bajte podatkov (primer 4.32) in jih izpiše na zaslonu (primer 4.33).

Primer 4.31 LCD modul: podmeni nastavitve temperature

```
nastavi_temp_podmeni
            call  LCD_Clr
            call  LCD_Line1
            bsf   text_tabela,0
            izpis_na_LCD      sklop_1,Trenutna_temp_text,1
            bcf   text_tabela,0
            call  sprejem_serijski
            movlw klici_LCD         ;splošni paket od MASTER za LCD modul?
            subwf bajt,W
            btfsc STATUS,Z
            goto  vprasaj_za_temp
            call  LCD_Clr           ;zgodila se je napaka! (nepravilen ukaz
                                    ;od MASTER modul-a)
;-----------
;NAPAKA!
;-----------
            bsf   text_tabela,0
            izpis_na_LCD      sklop_1,Napaka_modul_text,1
            bcf   text_tabela,0
```

```
            call  Delay1s
            call  Delay1s
            goto  konec_podmeni_temp      ;zaključi
;-----------
;OK!
;-----------

vprasaj_za_temp                     ;če ukaz od MASTER modula OK, bomo
                                    ;vprašali za trenutno temperaturo

            call  temp_master       ;sprejeli smo vse 4 bajte »temperatura«
                                    ;zdaj še jih izpišemo na zaslon
            call  temp_master_izpis
            bsf   text_tabela,0
            izpis_na_LCD      sklop_1,Nastavi_temp_text,2
            bcf   text_tabela,0
            movlw .2
            movwf st_vnosov_LCD     ;označimo, da gre za temperaturo
            call  vpis_na_LCD       ;uporabnik lahko nastavi TERMOSTAT
                                    ;temperaturo!
konec_podmeni_temp
            movlw HIGH izbira_nastavi_temp
            movwf PCLATH_HIGH
            return
```

Slika 4.4 LCD modul: pogled temperature

Primer 4.32 LCD modul: sprejem temperature

```
temp_master
            movlw .4
            movwf tmp3
            clrf  pomozni
temp_zanka
            movlw LCD_termostat_temperatura_TX
            movwf bajt
            call  oddaja_serijski
```

```
            call  sprejem_serijski
            movlw temperatura
            addwf pomozni,W
            movwf FSR
            movf  bajt,W
            movwf INDF
            incf  pomozni,F
            decfsz      tmp3,F
            goto  temp_zanka
            return
```

Primer 4.33 LCD modul: izpis temperature

```
temp_master_izpis                   ;izpis temperature na ekranu LCD
                                    ;format: »XYZ,Q °C«
            movlw .4
            movwf tmp3
            clrf  pomozni
temp_na_LCD
            movlw temperatura
            addwf pomozni,W
            movwf FSR
            movf  INDF,W
            movwf templcd
            call  LCD_Char
            incf  pomozni,F
            movf  pomozni,W
            sublw .3                ;preverimo do tam, kjer moramo vstaviti
                                    ;vejico (za 3. znakom)
            btfss STATUS,Z
            goto  ni_vejice
            movlw ','               ;prišli smo do vejice
            movwf templcd
            call  LCD_Char
ni_vejice
            decfsz      tmp3,F
            goto  temp_na_LCD
            movlw b'11011111'       ;binarni znak za stopinjo (°) iz nabora
                                    ;znakov, ki jih uporablja gonilnik
                                    ;Hitachi
            movwf templcd
            call  LCD_Char
            movlw 'C'
            movwf templcd
            call  LCD_Char
            return
```

PIR modul je neposredno povezan s sireno za plašenje vlomilcev. Frekvenco sirene moduliramo s PWM, primerno izbere uporabnik v `MENU-> Alarmi--> PIR---> Sirena----> Frekvenca`. Izbira med 5 vrednostmi (glej tabelo 4.7). Ob pritisku tipke

»GOR« ali «DOL« se pošlje ukaz do *PIR modula*, ki za trenutek akustično ponazori zvok z izbrano frekvenco. Če komunikacija ni mogoča, se javi napaka. Ob pritisku »OK« se nastavitev shrani v EEPROM, »MENI« opusti spremembe in se vrne v predhoden meni. Programsko rešitev prikazuje primer 4.34.

Primer 4.34 LCD modul: podmeni nastavitve frekvence sirene

```
PIR_frekvenca
            call  LCD_Clr
            movlw .1
            movwf text_tabela
            movlw Trenutna_frekv_text
            movwf text_zacetni
            call  sklop_1
            call  Delay255
            call  Delay255
            call  LCD_Line2
PIR_frekvenca_sprejem
            call  sprejem_serijski
            movlw klici_LCD
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_frekvenca_napaka    ;napaka, ni stika z MASTER ali
                                          ;drugim modulom
            movlw LCD_Alarmi_PIR_sirena_TX      ;ok MASTER identifikacija
            movwf bajt
            call  oddaja_serijski
            call  sprejem_serijski
            movlw LCD_Alarmi_PIR_sirena_RX      ;je MASTER pripravljen?
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_frekvenca_napaka    ;ne! Napaka, ni stika z MASTER ali
                                          ;drugim modulom
            ;ok, lahko pošljemo naše trenutne nastavitve (iz ee_proma) za
alarme (v katerih tudi piše podatek o frekvenci)

            movlw b'11100000'       ;pogledamo, kakšna je vrednost v
                                    ;»ee_podatki« za frekvenco
            andwf ee_podatki,W      ;začasno shranimo »podatki« v »tmp2«
            movwf tmp2              ;v »tmp2« imamo zdaj samo 3 najvišje
                                    ;bite (= frekvenca)
            movwf bajt
            call  oddaja_serijski
PIR_frekv300
            clrw
            subwf tmp2,W
            btfss STATUS,Z
            goto  PIR_frekv1000
            movlw hz_300            ;Izpis na zaslonu(2. vrstica): »300 Hz«
            movwf text_zacetni
```

```
            call  sklop_1
            goto  izbira_frekv
PIR_frekv1000
            movlw b'00100000'
            subwf tmp2,W
            btfss STATUS,Z
            goto  PIR_frekv1700
            movlw hz_1000           ;»1000 Hz«
            movwf text_zacetni
            call  sklop_1
            goto  izbira_frekv
PIR_frekv1700
            movlw b'01000000'
            subwf tmp2,W
            btfss STATUS,Z
            goto  PIR_frekv2400
            movlw hz_1700           ;»1700 Hz«
            movwf text_zacetni
            call  sklop_1
            goto  izbira_frekv
PIR_frekv2400
            movlw b'01100000'
            subwf tmp2,W
            btfss STATUS,Z
            goto  PIR_frekv3100
            movlw hz_2400           ;»2400 Hz««
            movwf text_zacetni
            call  sklop_1
            goto  izbira_frekv
PIR_frekv3100
            ;default
            movlw hz_3100           ;»3100 Hz«
            movwf text_zacetni
            call  sklop_1
;---------------------------------
;meni za frekvenco
;---------------------------------
izbira_frekv                        ;"GOR"
            btfsc PORTB,1
            goto  izbira_frekv1
            call  frekv_naprej
izbira_frekv1                       ;"DOL"
            btfsc PORTB,0
            goto  izbira_frekv2
            call  frekv_nazaj
izbira_frekv2                       ;"OK"
            btfsc PORTB,2
            goto  izbira_frekv3
            goto  frekv_OK
izbira_frekv3                       ;"MENI"
            btfsc PORTB,3
            goto  izbira_frekv
            movlw .0
            movwf text_tabela
```

```
            return
;-----------------------------------------------
;funkcija, ki pošlje izbrano frekvenco do MASTER
;-----------------------------------------------
LCD_oddaj_frekvenco
            call  sprejem_serijski
            movlw LCD_Alarmi_PIR_sirena_TX
            movwf bajt
            call  oddaja_serijski
            call  sprejem_serijski
            movlw LCD_Alarmi_PIR_sirena_RX      ;je ok, lahko pošljem
                                                ;frekvenco sirene?
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_frekvenca_napaka
            movf  tmp2,W
            movwf bajt
            call  oddaja_serijski
            return
frekv_naprej
            movlw .242              ;če je trenutni kazalec na nizu
                                    ;»hz_3100«, potem zankamo naprej do
                                    ;»hz_300«
            subwf text_zacetni,W
            btfsc STATUS,Z
            goto  frekv_obrni_nap
            call  LCD_Line2
            movlw .5
            addwf text_zacetni,F
            call  sklop_1
            movlw .32
            addwf tmp2,F
            nop
            call  LCD_oddaj_frekvenco     ;ok, oddaj frekvenco!
            call  Delay255
            return
frekv_obrni_nap
            call  LCD_Line2
            movlw .222
            movwf text_zacetni
            call  sklop_1
            clrf  tmp2
            call  LCD_oddaj_frekvenco     ;ok, oddaj frekvenco!
            call  Delay255
            return
frekv_nazaj
            movlw .222              ;če je trenutni kazalec nastavljen na
                                    ;niz »hz_300«, potem zankamo nazaj do
                                    ;»hz_3100« (da dobimo krožni meni)
            subwf text_zacetni,W
            btfsc STATUS,Z
            goto  frekv_obrni_naz
            call  LCD_Line2
            movlw .5
```

```
            subwf text_zacetni,F
            call  sklop_1
            movlw .32
            subwf tmp2,F
            call  LCD_oddaj_frekvenco
            call  Delay255
            return
frekv_obrni_naz
            call  LCD_Line2
            movlw .242
            movwf text_zacetni
            call  sklop_1
            movlw b'10000000'
            movwf tmp2
            call  LCD_oddaj_frekvenco
            call  Delay255
            return
frekv_OK                          ;izbrali smo frekvenco, to potrdimo in
                                  ;shranimo v EEPROM
            bcf   ee_podatki,5
            bcf   ee_podatki,6
            bcf   ee_podatki,7
            movf  tmp2,W
            addwf ee_podatki,F
            movf  ee_podatki,W
            movwf tmp1            ;podatki za EEPROM
            clrf  tmp2            ;naslov v EEPROM-u
            call  v_eeprom
            movlw .0
            movwf text_tabela
            return
;--------------------------
;NAPAKA na modulu
;--------------------------
PIR_frekvenca_napaka
            movlw Napaka_modul_text
            movwf text_zacetni
            call  sklop_1         ;izpis »NAPAKA!« na LCD-ju
            clrf  text_tabela
            call  Delay1s
            return
```

Edini možen način, da aktiviramo sistem pametne hiše, je vpis pravilne PIN kode (šifre). Možno jo je tudi zamenjati v `MENU-> Nastavitve--> PIN`. Prednastavljena vrednost je »1234« (vedno je dolžine 4 števk).

Po pravilnem vpisu PIN kode (funkcija »vpis_na_LCD«, primer 4.30) se na ekranu odštevajo sekunde do vklopa sprožitve. V tem času mora uporabnik zapustiti prostor, sicer se sproži alarm (*PIR modul*).

Primer 4.35 LCD modul: aktivacija sistema inteligentne hiše

```
;------------------
;MENU             |
;->Sprozi         |
;------------------
izbira_sprozi
            izbirni_meni      Menu_text, Sprozi_text, izbira_izhod,
                              izbira_nastavitve, ODSTEVAJ, stand_by
ODSTEVAJ                            ;uporabnik da zahtevo za aktivacijo
                                    ;sistema
            bsf   PIN_skrita,0      ;skrijemo znake (recimo: »***4«, pri
                                    ;vpisu šifre: »1234«)
            callm PIN_meni          ;glej spodaj
            bcf   PIN_skrita,0
            btfss PIN,0             ;je šifra pravilna?
            goto  izbira_ sprozi    ;ne, gremo nazaj
            callm odstevaj_sekunde  ;da, odštevaj 5 ali 10 sekund (odvisno
                                    ;od nastavitve uporabnika)
            callm vklopi_poslji_nastavitve      ;pošljemo še vse nastavitve
            gotom stand_by          ;skok nazaj v zanko preverjanja odziva
                                    ;MASTER modula
;---------
;PIN meni
;---------
PIN_meni
            call  LCD_Clr
            clrf  st_vnosov_LCD     ;označuje, da gre za šifro
            bsf   text_tabela,0
            call  LCD_Line1         ;prva vrstica na LCD-ju
            btfss PIN_skrita,0      ;PIN_skrita:
                                    ;     - bit 0: '0' = ni skrito, '1' = je
                                    ;     skrito (znak *)
                                    ;     - bit 1: '0' = stara šifra, '1' =
                                    ;     nova šifra
            goto  PIN_meni_naprej
            btfss PIN_skrita,1
            movlw '1'               ;»PIN_meni« kličemo na 2 možna načina:
                                    ;     - prvič za šifro, ki jo primerjamo
                                    ;     s tisto že obstoječo v pomnilniku
                                    ;     PIC-a (avtentikacija uporabnika),
                                    ;     - drugič za novo šifro, ki jo
                                    ;     vpišemo v pomnilnik PIC-a (pri
                                    ;     spremembi gesla/šifre)
            btfsc PIN_skrita,1
            movlw '2'
            movwf templcd
            call  LCD_Char
            movlw ')'
            movwf templcd
            call  LCD_Char
PIN_meni_naprej
            movlw Vpisi_PIN_text    ;»Vpisi kodo PIN:«
```

```
            movwf text_zacetni
            call  sklop_1
            bcf   text_tabela,0
            call  LCD_Line2
            call  vpis_na_LCD       ;v drugi vrstici sledi uporabnikov vpis
PIN_preverba                        ;preverimo pravilnost šifre oz. naredimo
                                    ;nov vpis
            clrf  count
            movlw .4
            movwf pomozni
preverba
            movlw crke
            addwf count,W
            movwf FSR
            movf  INDF,W
            movwf tmp1              ;prvo primerjavo shranimo v »tmp1«
            btfss PIN_skrita,1
            goto  preverba_naprej
            movwf podatki           ;nov vpis šifre v EEPROM
            movlw .12
            addwf count,W
            movwf lokacija
            call  VPIS_EE
            movlw sifra             ;nato sledi še vpis na RAM lokacije (za
                                    ;hitrejši doseg vrednosti v času
                                    ;neprekinjenega napajanja sistema)
            addwf count,W
            movwf FSR
            movf  tmp1,W
            movwf INDF
            goto  sifra_OK
preverba_naprej
            movlw .12               ;odmik za »šifra« (glej primer 4.18)
            addwf count,W
            movwf lokacija
            call  BERI_EE
            subwf tmp1,W            ;primerjava
            btfsc STATUS,Z          ;je šifra pravilna?
            goto  sifra_OK          ;da
            call  LCD_Clr           ;ne, šifra ni OK! Izpis ter nazaj v
                                    ;meni
            bsf   text_tabela,0
            call  LCD_Line1
            movlw Napacen_PIN_text
            movwf text_zacetni
            call  sklop_1           ;»Napacen PIN!!!«
            bcf   text_tabela,0
            call  Delay1s           ;pustimo napis 2 sekundi na zaslonu, da
                                    ;ga uporabnik lahko prebere
            call  Delay1s
            clrf  PIN               ;'0' = nepravilen PIN
            return
sifra_OK                            ;šifra je pravilna, nadaljujemo s
                                    ;preverjanjem
```

```
            incf  count,F
            decfsz      pomozni,F
            goto  preverba          ;gremo preverit naslednji znak, dokler
                                    ;ne obdelamo vseh znakov (4)
pravilen_PIN_dokoncaj               ;izpišemo, da smo sprejeli »pravilen
                                    ;PIN«
            call  LCD_Clr
            bsf   text_tabela,0
            call  LCD_Line1
            movlw Pravilen_PIN_text
            movwf text_zacetni
            call  sklop_1
            bcf   text_tabela,0
            call  Delay1s
            call  Delay1s
            movlw .1
            movwf PIN               ;'1' = pravilen PIN
            return
```

Za uspešnost vklopa sistema je potrebno še pravilno poslati vse uporabniške podatke do *MASTER modula* (in od tam naprej). Če je uspešno tudi to, na *MASTER modulu* zasveti LED za »sistem vklopljen«. *LCD modul* začne izvajati določeno obliko večopravilnosti (angl. »multitasking«), istočasno spremlja stanje linij serijske komunikacije (če se je sprožil kateri od alarmov, se to izpiše na zaslonu) ter stanje tipk (vnos šifre). Če je vpisana koda pravilna, se sistem deaktivira. Programski odsek tega prikazuje primer 4.36.

Primer 4.36 LCD modul: večopravilnost

```
;-------------VKLOPLJENO!-----------------
komunikacija
            clrf  PIN               ;PIN 0. bit: '0' = napačna šifra,
                                    ;            '1' = pravilna šifra
                                    ;privzeto je »napačen« PIN
            callm LCD_Clr
            movlw Vpisi_PIN_text    ;v prvi vrstici
            addlw .11               ;zamik zaradi besedila »PIN:«
            movwf text_zacetni
            bsf   text_tabela,0     ;beremo iz »text2« sklopa besedil
            callm sklop_1
                                    ;     ====================
                                    ;     //PIN:[šifra]     //
                                    ;     //    [alarmi]    //
                                    ;     ====================
                                    ;TIMER1 - ob »TMR1 overflow« preverimo
                                    ;stanje tipk ("GOR", "DOL" itd.)
            clrf  stolpec           ;označuje, na kateri offset poziciji
                                    ;PIN šifre smo
```

```
            clrf  st_vnosov_LCD     ;oznaka označuje »sifra«
            bsf   PIN_skrita,0      ;označuje, da bomo PIN številko
                                    ;sproti skrivali z znakom '*'
            bcf   PIN_skrita,1      ;označimo, da gre za »staro PIN«
                                    ;številko
init_TMR1                           ;inicializacija TMR1 časovnika
            bcf   PIR1,TMR1IF       ;pobrišemo »TIMER1 overflow« bit
            clrf  TMR1L             ;pobrišemo nižji bajt registra TIMER1
            clrf  TMR1H             ;pobrišemo višji bajt registra TIMER1
            movlw b'00110001'
            movwf T1CON             ;T1CON je register za upravljanje s
                                    ;časovnikom TIMER1, nastavitve:
                                    ;     - TIMER1 prescaler 1:8,
                                    ;     - no SYNC,
                                    ;     - internal oscilator,
                                    ;     - enable TIMER1
                                    ;prekoračitev (angl. »overflow«) po
                                    ;8*2^16 = cca. 0,52 sekunde
            movlw '0'               ;začnemo z '0'
            movwf znak
            movwf templcd
            callm LCD_Char
preveri
            callm ukazi             ;preverimo stanje serijske povezave z
                                    ;MASTER modulom (morebitni alarmi?)
;--------------
;OPRAVILO 0
;--------------
vklopljen_naprej
            btfsc PORTB,1           ;"GOR"
            goto  vklopljen_nazaj
            callm naprej
            goto  preveri
vklopljen_nazaj
            btfsc PORTB,0           ;"DOL"
            goto  vklopljen_naslednji
            callm nazaj
            goto  preveri
vklopljen_naslednji                 ;"OK"
            btfsc PORTB,2
            goto  vklopljen_preklici
            callm naslednji
            movlw .4
            subwf stolpec,W         ;smo že vpisali 4 znake?
            btfss STATUS,Z
            goto  init_TMR1         ;ne
            callm PIN_preverba      ;da, uporabnik je vpisal 4 števila
            btfss PIN,0             ;je bil pravilen PIN?
            goto  komunikacija      ;ne
obvesti_MASTER                      ;da, pravilen PIN! Obvesti o tem MASTER
                                    ;modul, da izključi sistem
            callm sprejem_serijski
            movlw klici_LCD         ;"blank check"?
            subwf bajt,W
```

```
            btfss STATUS,Z
            goto  obvesti_MASTER    ;čakamo na klic gospodarja v nedogled
            movlw LCD_IZKLOPI_VSE_TX      ;da, smo dobili odziv
            movwf bajt
            callm oddaja_serijski
            goto  START             ;gremo na začetek ...
vklopljen_preklici                  ;"MENI"
            btfsc PORTB,3
            goto  preveri
brisi_PIN                           ;pobrišemo vse nazaj do "PIN:"
            incf  stolpec,W
            movf  stolpec,W
            addlw .4                ;odmik za "PIN:" (4 znaki)
            movwf templcd
            callm LCD_Line1W
            movlw ' '
            movwf templcd
            callm LCD_Char
            decfsz      stolpec,F
            goto  brisi_PIN
            goto  init_TMR1         ;ponovi (zanka)
;--------------
;OPRAVILO 1
;--------------
ukazi
            call  sprejem_serijski
poplava_alarm
            movlw LCD_Alarmi_RX     ;prišlo do alarma?
            subwf bajt,W
            btfss STATUS,Z
            goto  alarmi_reset      ;ne
            movlw LCD_Alarmi_TX     ;da, potrdi sprejem
            movwf bajt
            call  oddaja_serijski
            call  LCD_Clr_Line2
            call  LCD_Line2
            movlw Alarm_poplava
            movwf text_zacetni
            call  sklop_1
            call  kurzor_prvotni    ;postavimo kurzor nazaj na svoje prvotno
                                    ;mesto (v prvo vrstico)
            goto  uporabnik
alarmi_reset
            movlw LCD_Alarmi_reset_RX     ;reset alarma?
            subwf bajt,W
            btfss STATUS,Z
            goto  PIR_alarm               ;ne, preveri za naslednji ukaz
            movlw LCD_Alarmi_reset_TX     ;ok, potrdi
            movwf bajt
            call  oddaja_serijski
            clrw
            subwf ze_napisano,W     ;od vklopa sistema še ni prišlo
                                    ;do nobenega alarma?
            btfss STATUS,Z
```

```
            goto  ekran_je_popisan  ;ekran je že popisan
            incf  ze_napisano,F     ;prvi ukaz od MASTER_modula po
                                    ;tistem, ko se sistem aktivira
            goto  uporabnik
ekran_je_popisan
            call  LCD_Clr_Line2
            call  LCD_Line2
            movlw Alarm_poplava_reset
            movwf text_zacetni
            call  sklop_1
            call  kurzor_prvotni    ;postavimo kurzor nazaj na svoje
                                    ;prvotno mesto (v prvo vrstico)
            goto  uporabnik
PIR_alarm
            movlw LCD_Alarm_PIR_RX  ;PIR alarm?
            subwf bajt,W
            btfss STATUS,Z
            goto  uporabnik         ;ne
            movlw LCD_Alarm_PIR_TX  ;da, potrdi sprejem
            movwf bajt
            call  oddaja_serijski
            call  LCD_Clr_Line2     ;počisti drugo vrstico
            call  LCD_Line2
            movlw 'P'               ;2 odseka besedil sta že popolnoma
                                    ;zasedena, zato izpišemo kar s klicanjem
                                    ;posameznih znakov (manj prog. kode, kot
                                    ;če bi ustvarili nov »odsek besedil«,
                                    ;npr.text3)
            movwf templcd
            call  LCD_Char
            movlw 'I'
            movwf templcd
            call  LCD_Char
            movlw 'R'
            movwf templcd
            call  LCD_Char
            movlw '!'
            movwf templcd
            call  LCD_Char
            call  kurzor_prvotni
            goto  uporabnik
uporabnik
            btfss PIR1,TMR1IF       ;je »TMR1 overflow« bit setiran?
            goto  ukazi             ;če je uporabnik kaj pritisnil,
                                    ;preverimo samo vsake pol sekunde
            bcf   PIR1,TMR1IF       ;pobrišemo »TMR1 overflow« bit, TMR1
                                    ;pustimo, da teče dalje
            return

kurzor_prvotni                      ;postavimo kurzor na LCD-ju nazaj, kjer
                                    ;je bil
            movf  stolpec,W
            addlw .4
            movwf templcd
```

```
            call  LCD_Line1W
            return
```

4.2.5 SMS modul

SMS modul skrbi za obveščanje uporabnika ter nadzor sistema inteligentne hiše na daljavo (angl. »remote control«). GSM omrežje, ki ga uporablja, ima vrsto prednosti, predvsem dobro pokritost signala, torej dostop od skoraj kjerkoli do kamorkoli[7], zanesljivo in kakovostno omrežje, cenovno ugodno uporabo. PIC se z mobilnim telefonom (Siemens M50) poveže s strojno implementiranim USART vmesnikom. Simbolna hitrost prenosa je 9600 znakov/sekundo, 8 bitov v paketu, brez CRC bita ter z 1 dolžino »stop« bita na koncu vsakega poslanega paketa.

[7] Pokritost prebivalstva z GSM signalom na območju Slovenije je za največja mobilna operaterja, Simobil in Mobitel, za junij 2009 znašala 99.6%, vir: http://www.mobitel.si/pomoc-in-nastavitve/pokritost.aspx
vir: http://www.simobil.si/sl/inside.cp2?cid=B0850770-48AD-C0FA-B6B6-931CADEDC3C7&linkid=article

Diagram 4.10 SMS modul 1/3

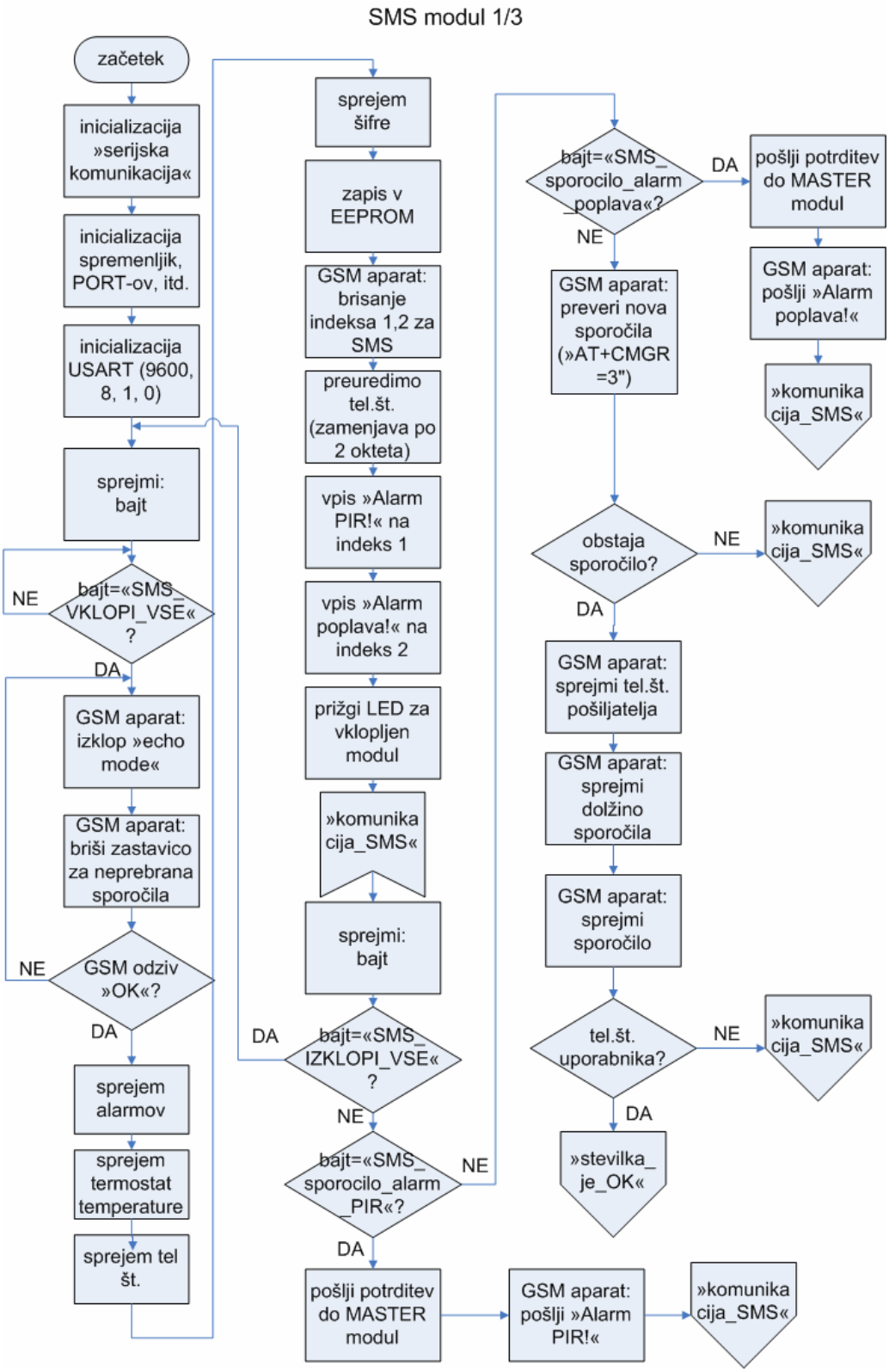


Diagram 4.11 SMS modul 2/3

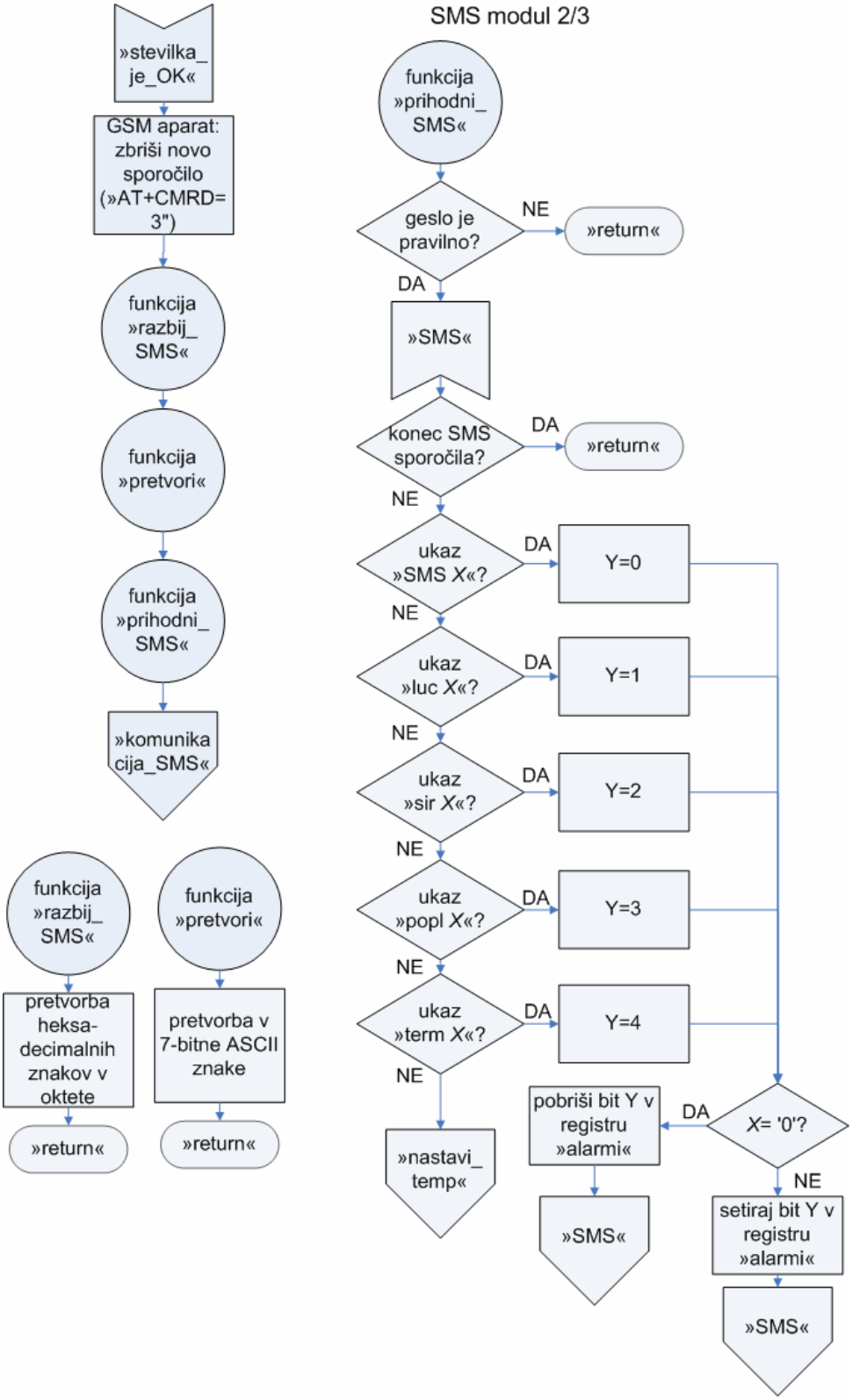
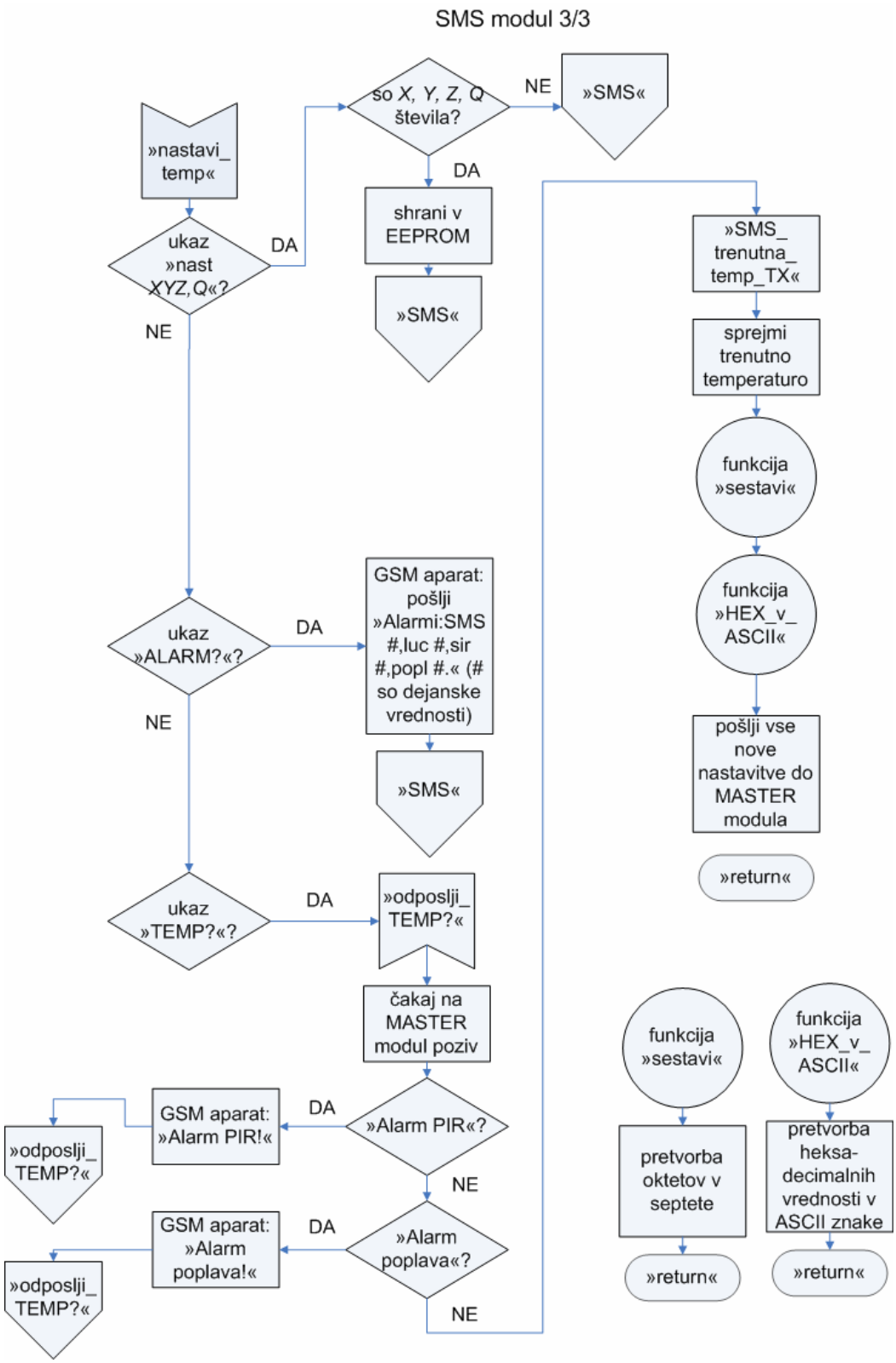


Diagram 4.12 SMS modul 3/3



Če dobimo veljaven ukaz za vklop modula (»SMS_VKLOPI_VSE_RX«), pogledamo, če je mobilni telefon sploh priključen in nas razume (AT komande). Izključimo »echo mode« (to je brez ponavljanja poslanih ukazov s strani GSM telefona), da je pogovor hitrejši. Nize znakov na USART pošilja funkcija »sklop_1« (primer 4.37). Potrdilo, prejeto z GSM aparata, preverjamo samo po ključnih črkah (s funkcijo »sprejem_OK«, primer 4.39) z znakom 'K', če je pozitivna potrditev (iz niza »OK«) oziroma ostalo kot negativno (to nikoli ne vsebuje 'K', ponavadi sprejmemo »ERROR«). S tem dodatno privarčujemo na dolžini zasedenega pomnilniškega prostora.

Primer 4.37 SMS modul: serijsko pošiljanje znakov na USART, funkcija »sklop_1«

```
sklop_1
            movlw PDU_sporocilo     ;začeli bomo pisati na lokaciji
                                    ;»PDU_sporocilo« (sem shranjujemo
                                    ;začasna sporočila z GSM aparata ali
                                    ;tista, ki jih bomo tja poslali)
            movwf FSR               ;register za posredno naslavljanje
            clrf  count
            clrw
zanka
            incf  count,F
            movf  count,W
            addwf text_zacetni,W    ;»text_zacetni« označuje prvo črko niza
                                    ;iz bloka nizov v pomnilniku
            btfsc text_tabela,0
            goto  naslednja_text
            addlw text
            movwf tmp3
            movlw HIGH text
            movwf PCLATH
            movf  tmp3,W
            call  text              ;beremo znak iz niza
            xorlw 0x00              ;je znak '0'(= konec niza)?
            btfsc STATUS, Z
            goto  konec             ;da
            movwf TXREG             ;ne, pošljem na USART
            call  CAKAJ             ;čakam, da je pošiljanja konec, nato
                                    ;nadaljujem (glej spodaj)
            goto  zanka
naslednja_text
            addlw LOW text2         ;Inštrukcijski register poskuša vpisati
                                    ;ves naslov »text2« (0x0100), vendar je
                                    ;le 8-bitni register, zato vpišemo le 8
                                    ;najmanj obteženih bitov
            movwf tmp3
            movlw HIGH text2        ;v PCLATH še vpišemo višje bite naslova
                                    ;registra
```

```
            movwf PCLATH
            movf  tmp3,W
            call  text2
            xorlw 0x00              ;konec niza?
            btfsc STATUS, Z
            goto  konec             ;da
            movwf INDF              ;ne
            incf  FSR,F
            goto  zanka
konec
            btfsc text_tabela,0
            movwf INDF              ;vpišemo tja, kamor kaže FSR
            return
;-----------------------------------------------------------------------
;»CAKAJ« funkcija se vrne šele takrat, ko so vsi biti znaka odposlani
;-----------------------------------------------------------------------
CAKAJ
            bsf   STATUS,RP0
            btfss TXSTA,TRMT
            goto  $-1
            bcf   STATUS,RP0
            return
```

Primer 4.38 SMS modul: inicializacija GSM

```
stand_by
            callm sprejem_serijski
            movlw SMS_VKLOPI_VSE_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  stand_by
            movlw SMS_VKLOPI_VSE_TX ;vrnemo odziv
            movwf bajt
            callm oddaja_serijski
            movf  RCREG,W           ;izbrišemo morebitne začasno shranjene
                                    ;znake sprejete preko USART-a (globina
                                    ;3)
            movf  RCREG,W
            movf  RCREG,W
            movlw AT_text           ;"AT", .13, .10
            movwf text_zacetni
            callm sklop_1           ;»sklop_1« zdaj, podobno kot na LCD
                                    ;modulu, deluje kot bralnik in
                                    ;izpisovalnik nizov, shranjenih v
                                    ;pomnilniku PIC-a; za razliko od LCD
                                    ;modula se tukaj znaki pošiljajo na
                                    ;USART
            callm sprejem_OK        ;pogledamo, če dobimo veljaven odziv
            btfsc napakaGSM,0
            goto  KOMUNIKACIJA      ;napaka! To čimprej javimo MASTER modulu
            movf  RCREG,W           ;ok, imamo modem, ki podpira AT komande
            movlw echo_off_text     ;izklopimo »Echo mode«
```

```
            movwf text_zacetni
            callm sklop_1           ;"ATE0", .13, .10
            callm sprejem_OK        ;imamo odziv?
            btfsc napakaGSM,0
            goto  KOMUNIKACIJA      ;ne, javi napako gospodarju
            movf  RCREG,W           ;pobriši sprejemni register USART
            movlw neprebrana_SMS_text     ;"AT+CMGL=0", .13, .10
            movwf text_zacetni
            callm sklop_1
            callm sprejem_OK
            btfsc napakaGSM,0
            goto  KOMUNIKACIJA      ;napaka, javi čimprej MASTER modulu
```

Primer 4.39 SMS modul: sprejem »OK« pozitivne potrditve, funkcija »sprejem_OK«

```
sprejem_OK
            bcf   RCSTA,CREN        ;reset USART sprejemni vmesnik
            bsf   RCSTA,CREN
prisotnost_ok?
            btfss PIR1,RCIF         ;je »USART receive buffer« poln?
            goto  prisotnost_ok?    ;ne
sprejemam                           ;da
            movf  RCREG,W
            sublw 'K'               ;v nizu, ki ga prejmemo, iščemo samo ta
                                    ;znak
            btfss STATUS, Z
            goto  prisotnost_ok?
            movlw .2                ;prišli smo do znaka 'K', na slepo
                                    ;sprejmemo še 2 znaka, in sicer
                                    ;»Carriage Return« in »New Line«
                                    ;(časovni interval 2/9600 s, cca. 208 us
                                    ;ali več)
            movwf stevec
se_dva_znaka
            btfss PIR1,RCIF
            goto  se_dva_znaka
            movf  RCREG,W
            decfsz      stevec,F
            goto  se_dva_znaka
            bcf   napakaGSM,0       ;ni napake!
konec_ok
            call  RS232_pavza       ;majhna pavza, da GSM aparat sprocesira
                                    ;sprejemne ukaze
            return
```

Eksperimentalno smo ugotovili, da GSM aparat ne more slediti hitremu tempu procesiranja mikrokontrolerja, zato za ukazi (recimo po prejemu »OK« potrditve) določen čas

še ne pošiljamo novih AT komand. Definirali smo precej veliko časovno pavzo za RS-232 podatke (primer 4.40).

Primer 4.40 SMS modul: časovna zakasnitev na RS-232 vodilu, funkciji »RS232_pavza« in »RS232_velika_pavza«

```
RS232_velika_pavza                  ;večja pavza, uporabimo za ukazom
                                    ;pošiljanja ali SMS besedila (komandi
                                    ;»AT+CMGS«, »AT+CMGW«)
            movlw .20
            movwf timer5            ;pomožni »časovni« register
zanka_velika                        ;približno 4 sekunde
            call  RS232_pavza
            decfsz      timer5,F
            goto  zanka_velika
            return                  ;izhod iz funkcije »RS232_velika_pavza«
RS232_pavza                         ;manjša pavza, uporabimo za potrditvijo
                                    ;poslane komande
            clrf  timer3            ;pomožni »časovni« register
            clrf  timer4            ;pomožni »časovni« register
zanka232                            ;približno 200 mS
            decfsz      timer3,F
            goto  zanka232
            decfsz      timer4,F
            goto  zanka232
            return                  ;izhod iz funkcije »RS232_pavza«
```

Nadaljnje sprejmemo vse potrebne parametre za pravilno delovanje *SMS modula*, vrednosti šifre (za avtentikacijo), telefonske številke (za pošiljanje k uporabniku), nastavitve alarmov (za možnost vklopa in izklopa modulov na daljavo) ter temperaturo termostata (vklop grelca ob izbrani temperaturi, na daljavo). Iz privzetega pomnilnika telefona zbrišemo prvi dve indeksirani sporočili (če obstajata) in naložimo naši predefinirani sporočili za alarme (»Alarm poplava!« in »Alarm PIR«). To prikazuje primer 4.41.

Primer 4.41 SMS modul: nastavitev GSM aparata

```
;1:                                 ;izbris prvih 2 indeksiranih mest
            movf  RCREG,W
            movlw zbrisi_SMS_1_text ;"AT+CMGD=1", .13, .10
            movwf text_zacetni
            callm sklop_1
            callm sprejem_OK        ;je ok?
;2:
            movf  RCREG,W
```

```
            movlw zbrisi_SMS_2_text ;"AT+CMGD=2", .13, .10
            movwf text_zacetni
            callm sklop_1
            callm sprejem_OK
            movlw .12               ;preuredimo telefonsko številko
                                    ;»SMS_stevilka1« (maksimalno število
                                    ;znakov bo 12)
            movwf dolzina_senderja  ;dolžina telefonske številke
            movlw SMS_sporocilo     ;tukaj začasno shranjena tel. številka
                                    ;za pošiljanje SMS sporočil
            movwf pomozni           ;tukaj shranjena lokacija, na kateri
                                    ;želimo preurejati tel. številko
                                    ;(fizični naslov)
            callm tel_v_PDU         ;»tel_v_PDU« uredi tel. št. v format,
                                    ;primeren za SMS sporočilo
            movlw SMS_sporocilo     ;na koncu še vstavimo 'F' namesto 0
            addlw .10
            movwf FSR
            movlw 'F'
            movwf INDF

            clrf  stevec            ;zdaj ko smo številko preuredili, jo
                                    ;vnesemo v EEPROM (kasneje lahko to
                                    ;vrednost primerjamo brez pretvarjanja,
                                    ;recimo ob sprejemu sporočila)
            movlw .12
            movwf count
tel_v_EEPROM
            movlw SMS_sporocilo     ;shranimo na lokacijo »SMS_sporocilo«
            addwf stevec,W          ;... plus odmik (velikosti »stevec«)
            movwf FSR
            movf  INDF,W
            movwf podatki           ;za v EEPROM
            movlw .1
            addwf stevec,W
            movwf lokacija          ;lokacija v EEPROM pomnilniku
            callm VPIS_EE           ;vpis v EEPROM
            incf  stevec,F
            decfsz      count,F
            goto  tel_v_EEPROM
;VPIS NOVIH SMS-jev
            clrf  vrsta_SMS         ;»vrsta_SMS« določa, katero sporočilo
                                    ;trenutno obdelujemo, in sicer:
                                    ;     - »vrsta_SMS«='0': alarm PIR
                                    ;     - »vrsta_SMS«='1': alarm poplava
                                    ;     - »vrsta_SMS«='2': nastavitve
                                    ;     alarmov
                                    ;     - »vrsta_SMS«='3': temperatura in
                                    ;     termostat
cakamo_ponovno
            callm cakaj_AT_odziv    ;čakamo odziv »AT«
            movf  RCREG,W           ;odplaknemo podatke iz RCREG registra
                                    ;(angl. »flush RCREG«)
            movf  RCREG,W
```

```
            movf  RCREG,W
            clrw
            subwf vrsta_SMS,W
            btfsc STATUS,Z
            goto  SMS_PIR
            movlw shrani_SMS_2_text ;vrsta_SMS='1'
            movwf text_zacetni      ;"AT+CMGW=27", .13
            callm sklop_1
            goto  lahko_pisem?
SMS_PIR                             ;vrsta_SMS='0'
            movlw shrani_SMS_1_text ;"AT+CMGW=23", .13
            movwf text_zacetni
            callm sklop_1
lahko_pisem?
            btfsc PIR1,RCIF
            goto  sprejemam2
            goto  lahko_pisem?
sprejemam2                          ;zdaj pogledamo, če imamo odziv (to je
                                    ;znak ">", ki mu sledi znak " ")
            movf  RCREG,W
            sublw ' '
            btfss STATUS, Z         ;smo sprejeli znak za presledek?
            goto  lahko_pisem?      ;ne
            callm RS232_pavza       ;da
            movlw PDU_header_text   ;... najprej pošljemo glavo sporočila
            movwf text_zacetni      ;"07918346401400F011000B81"
            callm sklop_1
            movlw .12               ;nato sledi telefonska številka
            movwf tmp2
            clrf  tmp3
vrini_telefonsko
            movlw .1
            addwf tmp3,W
            movwf lokacija
            callm BERI_EE           ;prebere preoblikovano telefonsko
                                    ;številko
            movf  podatki,W
            movwf TXREG             ;pošlje na USART
            callm CAKAJ             ;čaka do konca pošiljanja znaka
            incf  tmp3,F
            decfsz      tmp2,F
            goto  vrini_telefonsko
            clrw                    ;OK, smo vrinili telefonsko, pošlji še
                                    ;ostali del PDU-ja
            subwf vrsta_SMS,W
            btfsc STATUS,Z
            movlw Alarm_PIR_text    ;"0000FF0A417658DE064193D210", .26
            btfss STATUS,Z
            movlw Alarm_poplava_text;"0000FF0E417658DE06C1DF7076D81E0E01",
                                    ;.26
            movwf text_zacetni
            callm sklop_1
            callm sprejem_OK
            incf  vrsta_SMS,F
```

```
            movlw .2
            subwf vrsta_SMS,W
            btfss STATUS,Z
            goto  cakamo_ponovno
            bsf   PORTA,2           ;koncali smo z vpisom v GSM aparat,
                                    ;vključimo LED za »modul pripravljen«
```

Sprejemanje in oddajanje SMS sporočil je mogoče na 2 načina: tekstovno in PDU. Tekstovni način je uporabniško lažje berljiv (uporablja se kodna tabela ASCII) in predstavlja zgolj kodiranje toka podatkov (bitov) po določenem standardu. Abecede različnih standardov se lahko razlikujejo (isti znaki so kodirani drugače), najbolj znani standardi so »PCCP437«, »PCDN«, »8859-1«, »IRA« in »GSM«[18]. Tukaj pride na račun uporaba PDU oblike, ki do GSM aparata privede univerzalni nabor znakov (definiranih v GSM specifikaciji za GSM omrežja), ki se šele nato v samem aparatu oblikujejo v za uporabnika berljivo besedilo (lahko je uporabljen poljuben kodirni algoritem, za slovenščino tudi tak s šumniki). Nekateri GSM moduli podpirajo samo PDU format.

PDU ne vsebuje samo sporočila, temveč tudi metainformacije o pošiljatelju, časovnem žigu, servisnem centru itd. Podatki se nahajajo v heksa-decimalnih oktetih ali decimalnih semi-oktetih (to je nepravih oktetih, razloženo kasneje). Podroben opis niza PDU za sporočilo »Alarm poplava!« prikazuje tabela 4.4. Celoten poslan/sprejet niz je »07918346401400F011000B818346507696F20000FF0E417658DE06C1DF7076D81E0E01« (glej primer 3.1).

Tabela 4.4 SMS modul: PDU

Oktet(i)	Opis
07	Dolžina SMSC (angl. »Short Message Service Center«) informacije (7 oktetov).
91	Tip naslova SMSC; 91 pomeni mednarodni format telefonske številke.
83 46 40 14 00 F0	Telefonska številka servisnega centra, zapisana v decimalnih semioktetih[8].
11	Prvi oktet »SMS SUBMIT« sporočila.

[8] Tvorba semioktetov pojasnjena v nadaljevanju.

00	TP-MR (angl. »TP Message Reference«). Določa referenčni tip sporočila (npr. uspešno poslano sporočilo, sporočilo z napako itd.). Vrednost 0x00 pomeni, da GSM aparat sam izbere TP-MR.
0B	Dolžina telefonske številke.
81	Tip telefonske številke. 0x81 označuje neznan format telefonske številke (lahko bi uporabili tudi 0x91 za mednarodni format z 11 številkami).
83 46 50 76 96 F2	Telefonska številka prejemnika, napisana v semioktetih[9].
00	TP-PID (angl. »TP Protocol Identifier«). Določa protokol sporočila (npr. faks, e-mail, videotex, X.400, PSTN, ISDN ...). Vrednost 0x00 je privzeta vrednost za neznane ali s strani servisnega centra določene protokole.
00	TP-DCS (angl. »TP Data Coding Sheme«). Shema kodiranja uporabniških podatkov (sporočila). 0x00 je privzet GSM nabor znakov, nekompresirano besedilo, interpretirano v 7-bitnih znakih. Druge možnosti so: kompresija, 16-bitni Unicode znaki, 8-bitni znaki (za »smart messaging«) ...
FF	TP-VP (angl. »TP Validity Period«). Določa čas, kako dolgo lahko SMS sporočilo »živi«. Če v tem času sporočilo ni dostavljeno, ga SC uniči. 0xFF določa najdaljši čas (odvisen od SC).
0E	TP-UDL (angl. »TP User Data Length«). Dolžina uporabniških podatkov (sporočila). Ker TP-DCS določa 7-bitne znake, kodirane v oktetih, 0x0E potemtakem pomeni 14 septetov-znakov (sporočilo »Alarm poplava!«).
417658DE06C1DF7076D81E0E01	TP-UD (angl. »TP User Data«). Uporabniški podatki, torej besedilo, ki ga uporabnik lahko neposredno vidi na mobilnem telefonu[10].

Telefonska številka servisnega centra ter prejemnika sporočila je prirejena tako, da po dve številki zamenjamo med seboj. Če je število številk liho, potem se na koncu doda 'F'. Primer za telefonsko številko »38640567692« je rezultat: »8346507696F2«.

»Alarm poplava!« sestavlja 14 znakov, vsak od njih je kodiran s 7 biti oz. predstavlja 1 septet. Za SMS prenos sporočila je potrebno te septete vnesti v 8-bitno masko na naslednji način: prvi septet (znak 'A') je preoblikovan v oktet tako, da mu za najvišji bit vstavimo

[9] Pojasnjeno v nadaljevanju.

[10] Algoritmi in postopki za sestavljanje berljivega tekstovnega sporočila v PDU obliko in obratno, prikazano v nadaljevanju.

najnižji bit naslednjega septeta. Naslednji septet ima sedaj še samo 6 bitov, zato mu je potrebno dodati 2 najnižja bita tretjega septeta na konec okteta itn. Postopek nazorno prikazuje tabela 4.5 .

Tabela 4.5 SMS modul: pretvorba besedila v PDU obliko

črka	'A'	'l'	'a'	'r'	'm'	' '	'p'
decimalni zapis(ASCII)	65	108	97	114	109	32	112
binarni zapis (7-bit)	1000001	1101100	1100001	1110010	1101101	0100000	1110000
rezani biti	1000001	1101100	1100001	1110010	1101101	0100000	1110000
dodani biti (oktet)	01000001	01110110	01011000	11011110	00000110	11000001	11011111
nastali okteti (hex)	41	76	58	DE	06	C1	DF
črka	'o'	'p'	'l'	'a'	'v'	'a'	'!'
decimalni zapis(ASCII)	111	112	108	97	118	97	33
binarni zapis (7-bit)	1101111	1110000	1101100	1100001	1110110	1100001	0100001
rezani biti	1101111	1110000	1101100	1100001	1110110	1100001	0100001
dodani biti (oktet)	/	01110000	01110110	11011000	00011110	00001110	00000001
nastali okteti (hex)		70	76	D8	1E	0E	01

Sporočilo z GSM aparata v PIC prenesemo z AT komando za branje SMS sporočil (»AT+CMGR«), pri tem shranimo samo telefonsko številko pošiljatelja, dolžino telefonske številke pošiljatelja, dolžino sporočila ter samo sporočilo. Številko pošiljatelja primerjamo s tisto shranjeno v EEPROM-u (primer 4.42). Če se ujema, nadaljujemo z obdelavo sporočila, v nasprotnem primeru se javi napaka.

V RAM pomnilniku smo rezervirali 140 naslovnih polj kot začasni prostor za izvajanje operacij in shranjevanje uporabniških podatkov (označba lokacij: »SMS_sporocilo«[50 lokacij] in »PDU_sporocilo«[90 lokacij]). Sporočilo, ki ga lahko sprocesiramo, je tako odvisno od količine tega naslovnega polja.

Primer 4.42 SMS modul: avtentikacija telefonske številke

```
            clrf  stevec
preveri_telefonsko                  ;preverimo pravilnost tel. št.
                                    ;(shranjena na lokaciji 1 do 12 v
                                    ;EEPROM-u)
```

```
            movlw SMS_sporocilo     ;telefonska številka začasno shranjena
                                    ;na lokacije »SMS_sporocilo« (v času
                                    ;branja z mobilnega telefona)
            addwf stevec,W
            movwf FSR               ;kazalec na posredni register
            movf  INDF,W            ;posredni register
            xorlw 0x00              ;konec besedila vedno označimo z 0
            btfsc STATUS,Z          ;je 0?
            goto  stevilka_je_OK    ;da, številka je pravilna
            movwf T1                ;ne, preveri naslednji prejeti znak
                                    ;tel. št. Shrani v začasni splošni
                                    ;register »T1«
            movlw .1
            addwf stevec,W
            movwf lokacija
            callm BERI_EE           ;»pravilna« številka iz EEPROM-a (ki je
                                    ;tudi že prirejena na SMS format)
            movf  podatki,W
            subwf T1,W              ;primerjaj shranjeno številko in prejeto
                                    ;številko
            btfss STATUS,Z
            goto  napaka_sporocilo_tel    ;NAPAKA! Sporočilo ni iz prave
                                          ;številke!
            incf  stevec,F          ;nadaljujemo na naslednji znak
            goto  preveri_telefonsko      ;zanka
```

Sporočilo po končanem branju pobrišemo iz pomnilnika telefona (»AT+CMGD«). Prejeto sporočilo v PDU-ju najprej razbijemo s heksadecimalne oblike v oktete (v tabeli 4.5 vrstica »dodani biti (oktet)«) s klicem funkcije »razbij_SMS«.

Primer 4.43 SMS modul: pretvorba heksadecimalnih števil v oktete, funkcija »razbij_SMS«

```
razbij_SMS
            clrf  stevec            ;n-ti znak v PDU-ju
            clrf  trenutni_stevec   ;števec, ki meri, kam bomo že prirejen
                                    ;oktet vpisali
            clrf  tmp1              ;trenutni znak
            clrf  tmp2              ;trenutni odmik znaka
            clrf  tmp3              ;binarna vrednost obeh znakov
            clrf  pomozni           ;določimo, ali smo pri prvem ali drugem
                                    ;znaku
prvi_drugi_znak
            movf  stevec,W
            addlw PDU_sporocilo     ;prejeto sporočilo se nahaja na lokaciji
                                    ;»PDU_sporocilo«
            movwf FSR
            movf  INDF,W
            movwf tmp1              ;trenutni prebran znak shranimo v
                                    ;začasni register »tmp1«
```

```
            xorlw 0x00              ;konec prejetih znakov smo označili z 0
            btfsc STATUS,Z          ;konec sporočila?
            goto  zakljuci          ;da
            movlw 0x30              ;ne
                                    ;če znaku »0« odštejemo 0x30, dobimo
                                    ;vrednost 0, če znaku »1«, dobimo 1 itd.
            subwf tmp1,F            ;ta vrednost se nato shrani v »tmp1«
kk
            btfss STATUS,Z
            goto  naslednji_znak    ;naslednji znak
            movlw .0
            xorwf pomozni,W         ;smo pri pravem znaku?
            btfsc STATUS,Z
            goto  poisci_prvi       ;iskanje binarne vrednosti prvega znaka
            goto  poisci_drugi      ;iskanje binarne vrednosti drugega znaka
naslednji_znak
            incf  tmp2,F
            movlw .1
            subwf tmp1,F
            goto  kk
poisci_prvi                         ;iskanje šesnajstic
            movlw HIGH ASCII_HEX
            movwf PCLATH
            movf  tmp2,W
            call  ASCII_HEX         ;iščemo iz tabele »ASCII_HEX«
            movwf tmp3              ;v »tmp3« shranimo po dva znaka skupaj
                                    ;pretvorjena v binarno obliko, recimo iz
                                    ;znaka »1« in »4« dobimo vrednost '14' v
                                    ;»tmp3«
            incf  stevec,F          ;gremo na naslednji znak
            clrf  tmp2              ;pobrišemo začasni register
            incf  pomozni,F
            goto  prvi_drugi_znak
poisci_drugi                        ;iskanje enic
            movlw HIGH ASCII_HEX
            movwf PCLATH
            movf  tmp2,W
            call  ASCII_HEX
            movwf tmp2
            swapf tmp2,W
            xorwf tmp3,F
            incf  stevec,F
            clrf  tmp2
            clrf  pomozni
            movf  trenutni_stevec,W
            addlw PDU_sporocilo     ;vpišemo nazaj na lokacije
                                    ;»PDU_sporocilo«
            movwf FSR
            movf  tmp3,W
            movwf INDF
            incf  trenutni_stevec,F ;«trenutni_stevec« šteje sformatirane
                                    ;znake
            goto  prvi_drugi_znak
zakljuci
```

```
            movf  trenutni_stevec,W ;na koncu sformatiranega niza vpišemo 0
                                    ;(da bomo kasneje pri obdelavi vedeli
                                    ;kdaj je konec niza)
            addlw PDU_sporocilo
            movwf FSR
            movlw 0x00
            movwf INDF
            return
;-------------------------------------------
;ASCII_HEX vrača binarne vrednosti šestnajstic
;-------------------------------------------
ASCII_HEX
            ADDWF PCL,F             ;k PC dodamo vrednost iz splošnega
                                    registra W (vrne se n-ta vrstica, če je
                                    ;v W vrednost n)
            RETLW 0x00
            RETLW 0x10
            RETLW 0x20
            RETLW 0x30
            RETLW 0x40
            RETLW 0x50
            RETLW 0x60
            RETLW 0x70
            RETLW 0x80
            RETLW 0x90
            ;neveljavnih 7 znakov (v ASCII tabeli med »9« in »A«)
            RETLW 0x00
            RETLW 0x00
            RETLW 0x00
            RETLW 0x00
            RETLW 0x00
            RETLW 0x00
            RETLW 0x00
            RETLW 0xA0
            RETLW 0xB0
            RETLW 0xC0
            RETLW 0xD0
            RETLW 0xE0
            RETLW 0xF0
```

Nato je treba oktete razčleniti na 7-bitne ASCII vrednosti. Besedilo shranimo na lokacije »SMS_sporocilo«. Algoritem razčlembe prikazuje primer 4.44. Spisan je tudi obraten algoritem, prenos tekstovnih podatkov v PDU obliko (primer 4.48).

Primer 4.44 SMS modul: izpis 7-bitnih ASCII znakov, funkcija »pretvori«

```
pretvori
            movlw .1
            movwf st_shranjenih_bitov_tmp ;količina shranjenih bitov
```

```
            movwf st_shranjenih_bitov     ;količina shranjenih bitov
            clrf  tmp1              ;trenutna črka
            clrf  tmp2              ;shranimo ostanek do 7-ih bitov
            clrf  tmp3
            clrf  trenutni_bit
            clrf  stevec            ;za klicanje posameznih znakov iz tabele
                                    ;znakov
            clrf  znak_SMS
START2                              ;začetek
            movf  stevec,W
            addlw PDU_sporocilo
            movwf FSR
            movf  INDF,W
            movwf tmp1
            xorlw 0x00
            btfsc STATUS,Z          ;konec niza?
            goto  konec_pretvori    ;da
            clrf  tmp3              ;ne
            incf  stevec,F
            movf  stevec,W
            movwf trenutni_stevec
zankaj                              ;je »stevec« že obšel vseh 8 bitov?
            movlw .8
            subwf trenutni_stevec,W
            btfss STATUS,C
            goto  go                ;ne
            btfsc STATUS,Z          ;da, prišli smo do prestopnega znaka
            call  prestopni
            movlw .7
            subwf trenutni_stevec,F
            goto  zankaj
;-----------------------------------------------
;»prestopni« funkcija shrani ostanek
;-----------------------------------------------
prestopni                           ;za »prestopne« znake
            bcf   STATUS,C
            rrf   tmp2,F            ;shranimo ostanek kot znak (saj se ga je
                                    ;nabralo za vseh 7 bitov = nov znak)
            movlw SMS_sporocilo
            addwf znak_SMS,W
            movwf FSR
            movf  tmp2,W
            movwf INDF
            incf  znak_SMS,F        ;»znak_SMS« šteje število shranjenih
                                    ;znakov
            clrf  tmp2              ;ostanek = 0
            return
;-----------------------------------------
go
            movf  trenutni_stevec,W
            movwf preostanek
            sublw .8
            movwf stevilo_nicel
            movwf stevilo_nicel_tmp
```

```
            incf  stevilo_nicel_tmp,F
ponovimo
            movf  tmp1,W
            movwf tmp3              ;trenutni znak premaknemo v »tmp3«
ponovimo4
            bcf   STATUS,C
            rrf   tmp3,F            ;zamaknemo (angl. »shift«) v desno (k
                                    ;manj obteženim bitom)
            decfsz      stevilo_nicel_tmp,F
            goto  ponovimo4
            movf  stevilo_nicel,W
ponovimo5
            rrf   tmp3,F
            decfsz      preostanek,F
            goto  ponovimo5

            movf  trenutni_stevec,W
            movwf preostanek
            goto  ponovimo6_1
ponovimo6
            rlf   tmp1,F
            bcf   STATUS,C          ;pobrišemo »CARRY« bit, da se ne vrine
                                    ;z druge strani (glej tabela 4.1, »RRF«
                                    ;in »RLF«)
ponovimo6_1
            decfsz      preostanek,F
            goto  ponovimo6
            bcf   tmp1,7
            movf  trenutni_stevec,W
            movwf preostanek
            goto  ponovimo7_1
ponovimo7
            rrf   tmp1,F
ponovimo7_1
            decfsz      preostanek,F
            goto  ponovimo7
            movf  trenutni_stevec,W
            movwf pomozni
ponovimo8
            decfsz      pomozni,F
            goto  ponovimo8_1
            goto  izpisi7bit        ;shrani 7-bitni ASCII znak
ponovimo8_1
            rlf   tmp1,F
            rlf   tmp2,F
            btfsc STATUS,C
            bsf   tmp1,0
            btfss STATUS,C
            bcf   tmp1,0
            goto  ponovimo8
izpisi7bit
            movlw SMS_sporocilo     ;shranimo na lokacije »SMS_sporocilo« +
                                    ;odmik
            addwf znak_SMS,W
```

```
            movwf FSR
            movf  tmp1,W
            movwf INDF
            incf  znak_SMS,F
            movf  tmp3,W
            movwf tmp2
            clrf  tmp3
            goto  START2            ;ponovno
konec_pretvori
            movlw .7
            subwf trenutni_stevec,W ;če je slučajno »trenutni_stevec« prišel
                                    ;že do 7, potem moramo še enkrat klicati
                                    ;funkcijo »prestopni« (saj je
                                    ;neshranjen še 1 znak)
            btfss STATUS,Z
            goto  ni_vec_prestopni  ;ni prestopnega znaka
            call  prestopni         ;je prestopni znak
            incf  FSR,F
            clrw
            movwf INDF
            return
ni_vec_prestopni
            movlw SMS_sporocilo
            addwf znak_SMS,W
            movwf FSR
            clrw
            movwf INDF              ;na koncu vpišemo še 0
            return
```

Sporočilo je zdaj v obliki, kot ga je videl uporabnik, ki ga je poslal. Ukazi na daljavo, ki so pri tem možni, so prikazani v tabeli 4.6.

Tabela 4.6 SMS modul: Vhodno/izhodne informacije v SMS sporočilu

ukaz	vhodno/izhodna informacija (V/I)[11]	zgled implementacije	opis
obvestilo o poplavi	V – MASTER modul I – GSM aparat	primer 4.14	V primeru poplave na POPLAVA modulu MASTER modul pošlje alarm, SMS modul pa dalje na SMS številko sporočilo »Alarm poplava!«

[11] Vhodna informacija je tista, ki jo *SMS modul* prejme (od *MASTER modula* ali GSM aparata).

Izhodna informacija je ta, ki se ustvari znotraj *SMS modula* ter se nato posreduje naprej (*MASTER modulu* ali GSM aparatu).

obvestilo o detekciji na PIR modulu	V – MASTER modul I – GSM aparat	primer 4.15	V primeru detekcije gibanja na PIR modulu MASTER modul pošlje alarm, SMS modul pa dalje na SMS številko sporočilo »Alarm PIR!«
vklop/izklop SMS modula	V – GSM aparat I – MASTER modul	primer 4.16	Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »SMS *X*« (*X* je '0' – izklop SMS modula ali '1' – vklop SMS modula).
vklop/izklop luci na PIR modulu	V – GSM aparat I – MASTER modul	primer 4.16	Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »luc *X*« (*X* je '0' – izklop luči ali '1' – vklop luči).
vklop/izklop sirene na PIR modulu	V – GSM aparat I – MASTER modul	primer 4.16	Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »sir *X*« (*X* je '0' – izklop sirene ali '1' – vklop sirene).
vklop/izklop POPLAVA modula	V – GSM aparat I – MASTER modul	primer 4.16	Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »popl *X*« (*X* je '0' – izklop POPLAVA modula ali '1' – vklop POPLAVA modula).
vklop/izklop TEMP modula	V – GSM aparat I – MASTER modul	primer 4.16	Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »term *X*« (*X* je '0' – izklop TEMP modula ali '1' – vklop TEMP modula).
nastavi temperaturo termostata na TEMP modulu	V – GSM aparat I – MASTER modul		Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »nast *XYZ*,*Q*« (*XYZ* predstavljajo stopinje, *Q* polovico stopinje). Primer: »nast 020,5« bo nastavilo termostat na 20,5 °C.
obvestilo o trenutni temperaturi	V – GSM aparat I – MASTER modul, GSM aparat		Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »TEMP?« Kot odgovor se na uporabnikovo številko pošlje trenutna temperatura v SMS sporočilu.
obvestilo o trenutnih nastavitvah alarmov	V – GSM aparat I – MASTER modul, GSM aparat		Prejeto je bilo SMS sporočilo z uporabnikove telefonske številke s ključno besedo »ALARMI?« Kot odgovor se na uporabnikovo številko pošljejo informacije o modulih (vklopljeno/izklopljeno stanje).

Programska implementacija problema preveri PIN številke (prvi štirje znaki znotraj SMS besedila) in izvede iskanje ključnih nizov oz. ukazov. Funkcijo smo poimenovali

»prihodni_SMS«. Vključitev ali izključitev *SMS modula* prikazuje primer 4.45, proces je pri ostalih modulih (»luc *X*«, »sir *X*«, »popl *X*« in »term *X*«) enak, le da se maskira drugi bit v registru »alarmi«.

Primer 4.45 SMS modul: vklop/izklop modula, funkcija »prihodni_SMS«

```
prihodni_SMS
            clrf  lokacija          ;preberemo alarme z EEPROM-a
            call  BERI_EE
            movf  podatki,W
            movwf alarmi            ;»alarmi« uporabimo kot začasni
                                    ;register, nad katerim bomo uporabljali
                                    ;spremembe s prejetega SMS-ja
;uporabnik mora poslati SMS v obliki za prepoznavo, in sicer:
;1)GESLO (prvi 4 znaki v SMS)
;2)UKAZI
;                 SMS 0       izklopi SMS modul
;                 SMS 1       vklopi SMS modul
;                 luc 0       izklopi alarmiranje z lučmi (PIR modul)
;                 luc 1       vklopi alarmiranje z lučmi (PIR modul)
;                 sir 0       izklopi alarmiranje s sireno (PIR modul)
;                 sir 1       vklopi alarmiranje s sireno (PIR modul)
;                 popl 0      izklopi alarm za poplavo (POPLAVA modul)
;                 popl 1      vklopi alarm za poplavo (POPLAVA modul)
;                 term 0      izklopi termostat (TEMP modul)
;                 term 1      vklopi termostat (TEMP modul)
;                 nast xyz,q  nastavi temperaturo termostata na »xyz,q« °C ;
                              (TEMP modul)
;                 ALARM?      uporabnik: "pošlji mi alarme"
;                 TEMP?       uporabnik: "pošlji mi nastavitve za termostat
;                             in temperaturo"
            clrf  stevec
            movlw .4
            movwf count
preveri_prihodni_geslo          ;1)GESLO
            movlw SMS_sporocilo
            addwf stevec,W
            movwf FSR
            movlw .0            ;je terminator v »SMS_sporocilo« (=konec
                                ;SMS sporočila)?
            subwf INDF,W
            btfsc STATUS,Z
            return              ;da
            movf  INDF,W        ;ne
            movwf tmp1
            movlw .13           ;odmik v EEPROM-u, kjer se nahaja šifra
            addwf stevec,W
            movwf lokacija
            call  BERI_EE
            movf  podatki,W
            subwf tmp1,W
```

```
            btfss STATUS,Z          ;številka registra »sifra« ustreza tisti v
                                    ;sporočilu?
            return                  ;ne
            incf  stevec,F          ;da
            decfsz      count,F
            goto  preveri_prihodni_geslo
preveri_prihodni                    ;2)UKAZI
SMS                                 ;najprej za vklop/izklop SMS modula
            movlw SMS_sporocilo
            addwf stevec,W
            movwf FSR
            movlw .0                ;terminator v »SMS_sporocilo«?
            subwf INDF,W
            btfsc STATUS,Z
            goto  konec_prihodni_SMS      ;da, končamo s preverjanjem
                                          ;(»konec_prihodni_SMS« prikazan v
                                          ;nadaljevanju)
            movlw 'S'               ;ne, poglej za prvo ustrezno črko ('S')
            subwf INDF,W
            btfss STATUS,Z
            goto  luc               ;ni ukaz »SMS«, poglej naprej za
                                    ;naslednji ukaz (»luc« za vklop/izklop
                                    ;luči na PIR modulu)
;-------------------------------------
            incf  stevec,F          ;'M'
            incf  FSR,F
            movlw 'M'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'S'
            incf  FSR,F
            movlw 'S'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;' '
            incf  FSR,F
            movlw ' '
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F
            incf  FSR,F
            movlw .0                ;preverimo, da slučajno ni terminator
                                    ;(0)!
            subwf INDF,W
            btfsc STATUS,Z
            goto  konec_prihodni_SMS      ;da, končamo s preverjanjem
                                          ;(»konec_prihodni_SMS« prikazan v
                                          ;nadaljevanju)
```

```
            movlw '0'               ;ok, ni terminator, torej je to ta
                                    ;vrednost, ki jo želimo; pogledamo še,
                                    ;če je res cifra (ASCII '0' ali ASCII
                                    ;'1')
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS1              ;'1'
            bcf   alarmi,0          ;'0'
            goto  SMS
SMS1
            movlw '1'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS               ;nazaj na začetek
            bsf   alarmi,0
            goto  SMS               ;nazaj na začetek
```

Nastavitev temperature termostata shrani novo prebrano vrednost (če je pravilna, torej so primerni oblika in znaki) v EEPROM. Kasneje do gospodarja pošljemo vse vrednosti, ki so se spremenile.

Primer 4.46 SMS modul: nova temperatura termostata, funkcija »prihodni_SMS«

```
nastavi_temp
            movlw 'n'               ;najprej ukaz
            subwf INDF,W            ;'n'
            btfss STATUS,Z
            goto  ALARM?
;-------------------------------------
            incf  stevec,F          ;'a'
            incf  FSR,F
            movlw 'a'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'s'
            incf  FSR,F
            movlw 's'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'t'
            incf  FSR,F
            movlw 't'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
```

```
;-------------------------------------
            incf  stevec,F          ;' '
            incf  FSR,F
            movlw ' '
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            movlw .3
            movwf count             ;pogledamo prve tri znake
            clrf  counta            ;štetje posameznih znakov od 0 do 3
sprejmi_prve_tri
            incf  stevec,F
            incf  FSR,F
            movlw .0                ;preverimo za terminator
            subwf INDF,W
            btfsc STATUS,Z
            goto  konec_prihodni_SMS      ;konec SMS sporočila
            movlw 0x30              ;je res številka (torej more imeti znak
                                    ;vrednost med 0x30 za '0' in 0x39 za
                                    ;'9')?

            subwf INDF,W
            btfss STATUS,C
            goto  SMS               ;ni številka!
            movlw 0x40              ;zdaj more »carry« bit (C) označevati
                                    ;negativno število (C = 0)
            subwf INDF,W
            btfsc STATUS,C
            goto  SMS               ;ni številka!
            movf  INDF,W            ;je številka, začasno uporabimo
                                    ;spremenljivke »tmp1«, »tmp2« in »tmp3«
            movwf pomozni           ;začasno shranimo številko
            movlw tmp1
            addwf counta,W
            movwf FSR
            movf  pomozni,W
            movwf INDF
            incf  counta,F
            decfsz      count,F
            goto  sprejmi_prve_tri
            incf  stevec,F          ;sledi decimalna vejica
            movf  stevec,W
            addwf SMS_sporocilo,W
            movwf FSR
            movlw ','
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
            incf  stevec,F          ;ter še en znak (= številka)
            incf  FSR,F
            movlw .0
            subwf INDF,W
            btfsc STATUS,Z
```

```
            goto  konec_prihodni_SMS      ;konec SMS sporočila
            movlw 0x30
            subwf INDF,W
            btfss STATUS,C
            goto  SMS               ;ni številka!
            movlw 0x40
            subwf INDF,W
            btfsc STATUS,C
            goto  SMS               ;ni številka!
            movf  INDF,W            ;je številka, lahko opravimo vpis v EEPROM
            movwf podatki
            movlw .20
            call  VPIS_EE
            clrf  count
            movlw .3
            movwf counta            ;uporabimo začasno spr.
stevke_termostat                    ;vpis v EEPROM
            movlw .17
            addwf count,W
            movwf lokacija          ;17 je odmik lokacije, kjer se nahaja
                                    ;»termostat temperatura«
            movlw tmp1
            addwf count,W
            movwf FSR
            movf  INDF,W
            movwf podatki
            call  VPIS_EE
            incf  count,F
            decfsz      counta,F
            goto  stevke_termostat
            goto  SMS               ;vpis temperature termostata v EEPROM je
                                    ;končan
```

Ključna beseda »ALARM?« vrne uporabniku stanje modulov (vklop/izklop).

Slika 4.5 SMS modul: odziv ukaza »ALARM?«

Primer 4.47 SMS modul: sporočilo o stanju alarmov, funkcija »prihodni_SMS«

```
ALARM?
            movlw 'A'               ;'A'
            subwf INDF,W
            btfss STATUS,Z
            goto  TEMP?
;-------------------------------------
            incf  stevec,F          ;'L'
            incf  FSR,F
            movlw 'L'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'A'
            incf  FSR,F
            movlw 'A'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'R'
            incf  FSR,F
            movlw 'R'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'M'
```

```
            incf  FSR,F
            movlw 'M'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'?'
            incf  FSR,F
            movlw '?'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
odposlji_ALARMI                     ;pošljemo alarme v SMS sporočilu
            call  cakaj_AT_odziv
            movf  RCREG,W           ;»flush« RCREG
            movf  RCREG,W
            movf  RCREG,W
            movlw poslji_SMS_3_text ;"AT+CMGS=42", .13
            movwf text_zacetni
            call  sklop_1
lahko_pisem2?
            btfsc PIR1,RCIF
            goto  sprejemam4
            goto  lahko_pisem2?
sprejemam4
            movf  RCREG,W
            sublw ' '               ;pravilen odziv GSM aparata («> »)?
            btfss STATUS, Z
            goto  lahko_pisem2?
pisanje_v_memory2
            call  RS232_pavza
            movlw PDU_header_text   ;najprej pošljemo glavo PDU-ja
            movwf text_zacetni      ;"07918346401400F011000B81"
            call  sklop_1
            movlw .12               ;sledi telefonska številka
                                    ;število znakov pri telefonski (11 +
                                    ;redundantni 'F')
            movwf tmp2
            clrf  tmp3
vrini_telefonsko2
            movlw .1
            addwf tmp3,W
            movwf lokacija
            call  BERI_EE
            movf  podatki,W
            movwf TXREG
            call  CAKAJ
            incf  tmp3,F
            decfsz      tmp2,F
            goto  vrini_telefonsko2
            movlw delni_PDU_text    ;pošljemo še ostali del PDU-ja (o
                                    ;alarmih)
            movwf text_zacetni
```

```
            call  sklop_1           ;prepis iz data table (DT) v RAM
                                    ;lokacije
            bsf   text_tabela,0
            movlw alarmi_text       ;"Alarmi:SMS #,luc #,sir #,popl #."
            movwf text_zacetni
            call  sklop_1
            bcf   text_tabela,0
;ALARMI 0.bit: SMS                  ;zamenjamo znak '#' z '0' (izklopljeno)
                                    ;ali '1' (vklopljeno)
            movlw PDU_sporocilo
            addlw alarm_SMS
            movwf FSR
            btfsc alarmi,0
            movlw '1'
            btfss alarmi,0
            movlw '0'
            movwf INDF
;ALARMI 1.bit: luci
            movlw PDU_sporocilo
            addlw alarm_luci
            movwf FSR
            btfsc alarmi,1
            movlw '1'
            btfss alarmi,1
            movlw '0'
            movwf INDF
;ALARMI 2.bit: sirena
            movlw PDU_sporocilo
            addlw alarm_sirena
            movwf FSR
            btfsc alarmi,2
            movlw '1'
            btfss alarmi,2
            movlw '0'
            movwf INDF
;ALARMI 3.bit: poplava
            movlw PDU_sporocilo
            addlw alarm_popl
            movwf FSR
            btfsc alarmi,3
            movlw '1'
            btfss alarmi,3
            movlw '0'
            movwf INDF
            bcf   STATUS,RP1        ;nastavitve za izbiro »bank«, v katerem
                                    ;se nahaja RAM spremenljivka
                                    ;»SMS_sporocilo« v »bank0«,
                                    ;»PDU_sporocilo« v »bank1« (zato bomo
                                    ;spreminjali samo RP0 (preklop med tema
                                    ;dvema bankama)
            bcf   STATUS,RP0
            call  sestavi           ;funkcija sestavi septete
            call  HEX_v_ASCII       ;funkcija tvori ASCII znake iz
                                    ;heksadecimalnih vrednosti
```

```
            clrf  stevec
odposlji_konec_SMSja                ;pošljemo PDU
            movlw PDU_sporocilo
            addwf stevec,W
            movwf FSR
            movf  INDF,W
            xorlw 0x00
            btfsc STATUS, Z
            goto  konec_ALARMI?
            movwf TXREG
            call  CAKAJ
            incf  stevec,F
            goto  odposlji_konec_SMSja
konec_ALARMI?
            bcf   RCSTA,CREN        ;resetiramo »buffer« za prihajajoče
                                    ;podatke na USART
            bsf   RCSTA,CREN
ERROR_alarmi?                       ;je GSM aparat sprejel naše podatke?
            btfss PIR1,RCIF
            goto  ERROR_alarmi?
            movf  RCREG,W
            movwf tmp1
            sublw 'R'               ;»ERROR«?
            btfss STATUS, Z
            goto  ok_alarmi?
            call  RS232_velika_pavza;napaka, ni bilo poslano, gremo na
                                    ;ponovno pošiljanje
            goto  odposlji_ALARMI
ok_alarmi?
            movf  tmp1,W
            sublw 'K'                     ;»OK«?
            btfss STATUS, Z
            goto  ERROR_alarmi?
            call  RS232_pavza       ;sprejeli smo pozitivno potrdilo - »OK«
            bsf   PORTA,1           ;vizualno potrdilo (2 s gori LED)
            call  Delay1s
            call  Delay1s
            bcf   PORTA,1
            goto  konec_prihodni_SMS_2
```

»Sestavi« procedura je nasprotna tisti, poimenovani »razbij_SMS«. Ta bajte-oktete maskira v septete. Heksadecimalne vrednosti teh septetov potem spremenimo v ASCII znake (funkcija »HEX_v_ASCII«) in dobimo končno obliko PDU-ja za pošiljanje na GSM modul (primer 4.48).

Primer 4.48 SMS modul: pretvorba tekstovnega besedila v PDU obliko

```
sestavi                             ;sestavi - setavimo oktete iz septetov
```

```
            movlw .32               ;dolžina sporočila (velikost
                                    ;prednastavljenega sporočila poznamo že
                                    ;vnaprej)
            movwf stevec            ;štejemo znake do konca sporočila
                                    ;(okteti)
            clrf  tmp2              ;trenutni znak
            clrf  offset
            clrf  znak_PDU          ;število znakov v PDU-ju
            clrf  znak_SMS          ;število znakov v tekstovni obliki
            clrf  tmp3              ;(trenutni znak) + 1
            clrf  biti              ;označuje odmik v trenutnem bajtu
sestavi_start
            movlw PDU_sporocilo
            addwf znak_PDU,W
            movwf FSR
            movf  INDF,W
            movwf tmp2              ;shranimo trenutno prebran znak
            incf  FSR,F
            movf  INDF,W
            movwf tmp3              ;shranimo naslednji znak
            incf  offset,F
            movf  offset,W
            movwf biti
            btfss biti,3            ;3 bit označuje 2^3 = 8 (8 bitov)
            goto  vrti              ;rotacijski števec »biti« je dosegel
                                    ;vrednost 8 (= 1 bajt)
            clrf  offset
            incf  znak_PDU,F
            decf  stevec,F
            clrw
            subwf stevec,W
            btfsc STATUS,Z          ;števec prišel na 0?
            goto  sestavi_konec     ;ne
            goto  sestavi_start     ;da, preskočimo 1 izrabljen bajt
vrti                                ;zavrtimo trenutno prebran znak v levo
            rlf   tmp2,F
            decfsz      biti,F
            goto  vrti
            movf  offset,W
            movwf biti
vrti2
            rrf   tmp3,F
            rrf   tmp2,F
            decfsz      biti,F
            goto  vrti2
            movf  tmp3,W
            movwf INDF              ;»tmp3« shranimo nazaj na
                                    ;»PDU_sporocilo« (FSR še vedno kaže na
                                    ;to lokacijo)
            movlw SMS_sporocilo
            addwf znak_SMS,W
            movwf FSR
            movf  tmp2,W            ;»tmp2« na »SMS_sporocilo«
            movwf INDF
```

```
            incf  znak_PDU,F        ;naslednji znak v PDU-ju
            incf  znak_SMS,F        ;naslednji znak v SMS-ju
            decfsz      stevec,F
            goto  sestavi_start     ;naslednji znak
            movlw SMS_sporocilo     ;konec preobrazbe v septete, na koncu
                                    ;vpišemo še 0 (= konec s podatki)
sestavi_konec
            addwf znak_SMS,W
            movwf FSR
            clrf  INDF
            return
;-----------------------------------
;-----------------------------------
HEX_v_ASCII                         ;pretvorba HEX v ASCII
            clrf  stevec            ;zapisujemo nazaj iz »SMS_sporocilo« v
                                    ;»PDU_sporocilo«
            clrf  znak_PDU
            clrf  znak_SMS
            movlw .2
            movwf tmp3              ;s »tmp3« bomo označili, ali gre za prvi
                                    ;ali drugi »pol«-bajt
                                    ;primer: vrednost 0xA1 želimo pretvoriti
                                    ;v znaka »A« in »1«, pri tem je »A« prvi
                                    ;»pol-bajt«, »1« pa drugi
HEX_v_ASCII_2
            movlw SMS_sporocilo
            addwf znak_SMS,W
            movwf FSR
            clrw
            subwf INDF,W
            btfsc STATUS,Z          ;vrednost 0 (= konec niza)?
            goto  pretvorba_koncana ;da
            movlw HIGH HEX_Table    ;ne, »HEX_table« vsebuje šestnajstiške
                                    ;vrednosti:
                                    ;HEX_Table
                                    ;     movwf PCL
                                    ;     DT "0123456789ABCDEF"
            movwf PCLATH
            swapf INDF,F            ;najprej višji »polbajt«
            movlw b'00001111'       ;maskiranje bajta
            andwf INDF,W
            addlw HEX_Table
            addlw .1                ;v HEX_Table prva vrstica porabljena za
                                    ;inštrukcijo »movwf PCL«
            call  HEX_Table
            movwf tmp2
            movlw PDU_sporocilo
            addwf znak_PDU,W
            movwf FSR
            movf  tmp2,W
            movwf INDF
            incf  znak_PDU,F
            decfsz      tmp3,F
            goto  HEX_v_ASCII_2
```

```
            movlw .2
            movwf tmp3              ;drugi »polbajt«
            incf  znak_SMS,F
            goto  HEX_v_ASCII_2
pretvorba_koncana
            movlw PDU_sporocilo
            addwf znak_PDU,W
            movwf FSR
            movlw .26               ;znak »ESCAPE« (s tem se vedno zaključi
                                    ;oddaja SMS sporočila do GSM aparata, po
                                    ;GSM specifikaciji)
            movwf INDF
            incf  FSR,F
            clrw
            movwf INDF              ;0 (= konec sporočila)
            return
```

Nazadnje modul preveri še morebitno poizvedbo za temperaturne specifikacije (»TEMP?«). Ker se to vedno dogaja v času aktiviranega sistema, vmes preverja še ukaze gospodarja, recimo ukaza za poplavo in IR detekcijo, ki imata višjo prioriteto. Če v prenosu naleti na napačen odziv, se do gospodarja javi napaka (primer 4.49). Na koncu gospodarju dostavimo vse spremenjene vrednosti.

Primer 4.49 SMS modul: pošiljanje nastavitev termostata

```
TEMP?
            movlw 'T'               ;'T'
            subwf INDF,W
            btfss STATUS,Z
            goto  naslednji_znak_SMS
;-------------------------------------
            incf  stevec,F          ;'E'
            incf  FSR,F
            movlw 'E'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'M'
            incf  FSR,F
            movlw 'M'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'P'
            incf  FSR,F
```

```
            movlw 'P'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
            incf  stevec,F          ;'?'
            incf  FSR,F
            movlw '?'
            subwf INDF,W
            btfss STATUS,Z
            goto  SMS
;-------------------------------------
odposlji_TEMP?                      ;pošljemo nastavitve termostata ter
                                    ;trenutno temperaturo v SMS sporočilu
            call  cakaj_AT_odziv
            movf  RCREG,W
            movf  RCREG,W
            movf  RCREG,W
            movlw poslji_SMS_3_text ;"AT+CMGS=42", .13
            movwf text_zacetni
            call  sklop_1
lahko_pisem8?
            btfsc PIR1,RCIF         ;čakaj na znak
            goto  sprejemam8
            goto  lahko_pisem8?
sprejemam8
            movf  RCREG,W
            sublw ' '               ;» «?
            btfss STATUS, Z
            goto  lahko_pisem8?
pisanje_v_memory8
            call  RS232_pavza
            movlw PDU_header_text   ;"07918346401400F011000B81"
            movwf text_zacetni
            call  sklop_1
            movlw .12               ;vrinemo tel. št.
            movwf tmp2
            clrf  tmp3
vrini_telefonsko8
            movlw .1
            addwf tmp3,W
            movwf lokacija
            call  BERI_EE
            movf  podatki,W
            movwf TXREG
            call  CAKAJ
            incf  tmp3,F
            decfsz      tmp2,F
            goto  vrini_telefonsko8
            movlw delni_PDU_text    ;"0000FF20"
            movwf text_zacetni
            call  sklop_1
            bsf   text_tabela,0     ;prepis v RAM lokacije (za obdelavo)
            movlw termostat_text    ;"Term:nast. ###,#'C,temp. ###,#'C"
```

```
            movwf text_zacetni
            call  sklop_1
            bcf   text_tabela,0
morebiti_alarm                      ;se medtem pojavi alarm na PIR modulu?
            call  sprejem_serijski
            movlw SMS_sporocilo_alarm_PIR_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  naslednji_alarm2  ;ne, preveri naslednji alarm (poplava)
            movlw SMS_sporocilo_alarm_PIR_TX    ;da, alarm PIR!
            movwf bajt
            call  oddaja_serijski
            call  cakaj_AT_odziv    ;pošlji SMS z vsebino o alarmu (»Alarm
                                    ;PIR!«)
            movlw SMS_1_text        ;"AT+CMSS=1", .13, .10
            movwf text_zacetni
            call  sklop_1
            goto  odposlji_TEMP?
naslednji_alarm2
            movlw SMS_sporocilo_alarm_popl_RX   ;alarm poplava?
            subwf bajt,W
            btfss STATUS,Z
            goto  ni_vmesnih_alarmov      ;ne, ni vmesnih alarmov
            movlw SMS_sporocilo_alarm_popl_RX   ;da, alarm poplava!
            movwf bajt
            call  oddaja_serijski
            call  cakaj_AT_odziv
            movlw SMS_2_text        ;"AT+CMSS=2", .13, .10
            movwf text_zacetni
            call  sklop_1
            goto  odposlji_TEMP?
ni_vmesnih_alarmov
            movlw klici_SMS         ;splošni paket za kumunikacijo?
            subwf bajt,W
            btfss STATUS,Z
            goto  morebiti_alarm
            movlw SMS_trenutna_temp_TX
            movwf bajt
            call  oddaja_serijski
            call  sprejem_serijski
            movlw SMS_trenutna_temp_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  morebiti_alarm
            potrditev_temp?
            call  sprejem_serijski  ;v naslednjih 4 paketih bomo dobili
                                    ;podatke o trenutni temperaturi ("xyz,q"
                                    ;°C)
                                    ;počakamo da MASTER modul pridobi
                                    ;temperaturo od TEMP modula; čakamo na
                                    ;drugo potrditev
            movlw SMS_trenutna_temp_RX    ;drugič
            subwf bajt,W
            btfss STATUS,Z
```

```
            goto  potrditev_temp?
            movlw SMS_trenutna_temp_TX
            movwf bajt
            call  oddaja_serijski
            clrf  tmp1              ;štetje znakov (0, 1 ...)
            movlw .4
            movwf tmp2              ;odštevanje do konca
            movlw .17               ;odmik za EEPROM branje
            movwf tmp3
sprejem_temperature
            call  sprejem_serijski
            clrw
            subwf bajt,W
            btfsc STATUS,Z
            goto  NAPAKA_modula     ;ni sprejema (bajt = 0x00), javi napako
            movlw .11               ;odmik do začetka oznake nastavitve
                                    ;termostata v SMS-ju "...nast.
                                    ;###,#'C,.."
            addlw PDU_sporocilo     ;sporočilo imamo na lokacijah
                                    ;»PDU_sporocilo«[+odmik]
            addwf tmp1,W
            movwf FSR
            movf  tmp1,W
            addwf tmp3,W            ;»termostat« v EEPROM-u
            movwf lokacija
            movlw .1                ;preskok vejice
            subwf tmp2,W
            btfsc STATUS,Z
            incf  FSR,F
            call  BERI_EE           ;branje EEPROM-a
            movf  podatki,W
            movwf INDF
            movlw .14               ;št. znakov med »term:nast.« in »temp«
            addwf FSR,F
            movf  bajt,W
            movwf INDF
            movlw SMS_trenutna_temp_TX
            movwf bajt
            call  oddaja_serijski
            incf  tmp1,F
            decfsz      tmp2,F
            goto  sprejem_temperature
            goto  preurejanje_SMS
;---------------------------------
NAPAKA_modula                       ;napaka pri prenosu
            movlw SMS_napaka_modul_TX
            movwf bajt
            call  oddaja_serijski
            goto  morebiti_alarm
;---------------------------------
preurejanje_SMS                     ;spremenimo v PDU obliko
            bcf         STATUS,RP1
            bcf         STATUS,RP0
            call  sestavi           ;sestavi septete
```

```
            call  HEX_v_ASCII       ;sestavi oktete iz binarnih vrednosti
            clrf  stevec
odposlji_konec_SMSja2
            movlw PDU_sporocilo
            addwf stevec,W
            movwf FSR
            movf  INDF,W
            xorlw 0x00
            btfsc STATUS, Z
            goto  konec_prihodni_SMS_2
            movwf TXREG
            call  CAKAJ
            incf  stevec,F
            goto  odposlji_konec_SMSja2
```

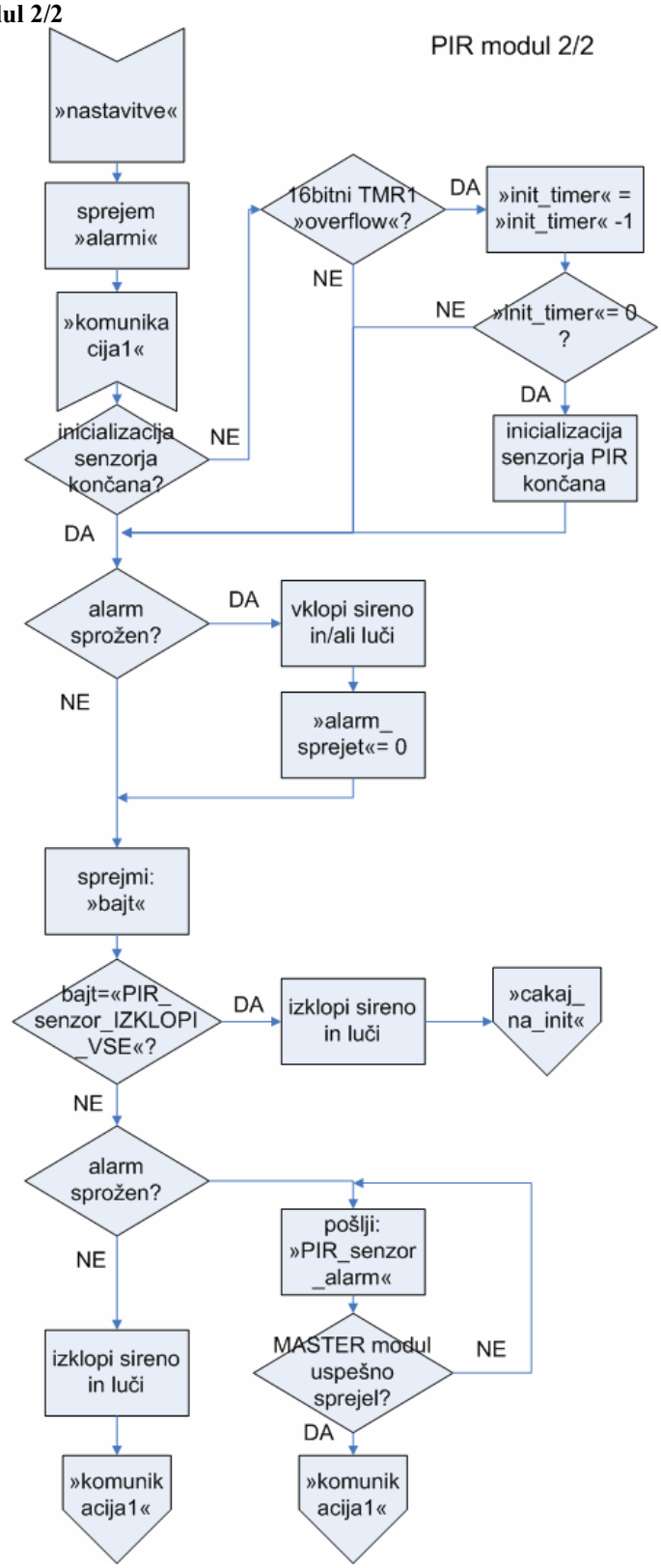
4.2.6 PIR modul

PIR detektorji lahko zaznajo premikanje človeka na opazovanem območju z zelo veliko verjetnostjo detekcije. Že majhna pozitivna ali negativna termalna sprememba glede na ozadje ob pomoči primernih optičnih elementov (leče) sproži vgrajen senzor. Ker imajo dobro razmerje med ceno in učinkovitostjo, se pogosto uporabljajo v sistemih hišnih alarmnih central ter kot ločeni senzorji z releji za vklop in izklop razsvetljave.

Delovanje tega modula kažeta diagrama poteka 4.13 do 4.14.

Diagram 4.13 PIR modul 1/2

Diagram 4.14 PIR modul 2/2



Programska zasnova modula je enostavnejša od do zdaj prikazanih. Naslovnega prostora, ki smo ga potrebovali, je bilo za manj kot 350 pomnilniških lokacij. Med cilji diplomske naloge je tudi predstavitev ekonomsko ugodne rešitve, zato smo uporabili »manjši« mikrokrmilnik (PIC16F628A), ki pa je vseeno pin-kompatibilen s tistim z več spomina (PIC16F648A). Tako je mogoče uporabiti tudi tega. Cenovna razlika znaša približno 12%[12], kar bi pri velikoserijski proizvodnji pomenilo precejšen prihranek.

Primer 4.50 PIR modul: začetek programa

```
      list p=16f628a                ;uporabili smo PIC z manj pomnilniškega
                                    ;prostora, saj ga ne potrebujemo toliko
      #include    "p16f628a.inc"    ;vključitvena datoteka za PIC
      #include    "serijska_komunikacija_macro.inc"
                                    ;serijska komunikacija
      #define     RXtris      TRISB,4     ;
      #define     TXtris      TRISB,5
      #define     RX          PORTB,4
      #define     TX          PORTB,5

      ERRORLEVEL  0,    -302, -205, -207

      cblock 0x20       ;začetek RAM spremenljivk
      bajt                          ;serijska komunikacija
      count                         ;za zanke štetja
      poskus                        ;serijska komunikacija, število poskusov
                                    ;sprejemanja enega bajta
      counta                        ;za časovne zakasnitve
      countb                        ;za časovne zakasnitve
      count1                        ;za časovne zakasnitve
      pomozni                       ;pomožni register
      inicializacija_PIR            ;"0" = inicializacija še poteka,
                                    ;"1" = inicializacija že opravljena
      init_timer                    ;pomožni register (pri inicializaciji
                                    ;PIR senzorja)
      alarm_sprejet                 ;0. bit setiran, če MASTER modul že
                                    ;sprejel obvestilo o alarmu
      alarmi                        ;1. bit = luči, 2. bit = sirena/zvočnik
      frekvenca                     ;frekvenca sirene/zvočnika
      zvocnik3s                     ;"0" = ne zvoni,
                                    ;"1" = zvoni (3 sekunde)
      zvocnik_pomozni               ;pomožni register
      TMR1overflow                  ;prekoračitev TIMER1 časovnika
      temp                          ;začasni register
      endc              ;konec RAM spremenljivk
```

[12] Za julij 2009 je cena posameznega kosa PIC16F628A znašala 1.47 $, PIC16F648A pa 1.67 $. Vir: http://www.microchip.com

Uporabljen DSC PIR senzor že vključuje začetni (angl. »settle down«) postopek. V PIC-u zato stanje vhodnega signala s strani senzorja opazujemo šele po tem dogodku (v primeru, da uporabnik aktivira sistem že prej). Medtem *PIR modul* še vedno spremlja linije serijske komunikacije (primer 4.51). Za določanje časovne zakasnitve smo uporabili zanke, enake tistim v primeru 4.23 za *LCD modul*.

Primer 4.51 PIR modul: priprava senzorja

```
cakaj_na_init
      btfss inicializacija_PIR,0    ;je senzor že prešel v stanje delovanja?
      call  je_inicializiran        ;ne
      btfsc zvocnik3s,0             ;da, vklopim zvočnik za 3 sekunde?
      call  zvocnik                 ;da
      call  sprejem_serijski        ;ne
      movlw PIR_senzor_IZKLOPI_VSE_RX     ;ukaz za izklopljen sistem?
      subwf bajt,W
      btfss STATUS,Z
      goto  PIR_senzor_init         ;preveri še druge ukaze, npr. vklopi PIR
                                    ;modul (sprejem nastavitev itd.)
      movlw PIR_senzor_IZKLOPI_VSE_TX     ;potrdimo, da je PIR že izklopljen
IZKLOP SISTEMA
      movwf bajt
      call  oddaja_serijski
      goto  cakaj_na_init
PIR_senzor_init
      movlw PIR_senzor_VKLOPI_RX    ;vklopimo modul?
      subwf bajt,W
      btfsc STATUS,Z
      goto  nastavitve              ;da
      movlw PIR_sirena_frekvenca_RX ;ne, sprejem frekvence za sireno /
                                    ;zvočnik?
      subwf bajt,W
      btfss STATUS,Z
      goto  cakaj_na_init           ;ne
      movlw PIR_sirena_frekvenca_TX ;da, vrnemo potrdilo za sprejem ukaza
      movwf bajt
      call  oddaja_serijski
      call  sprejem_serijski
      movf  bajt,W                  ;sprejeli smo nastavitve za frekvenco
                                    ;zvočnika (v MASTER modulu register
                                    ;»alarmi«)
      movwf frekvenca               ;shranimo v register »frekvenca«
      call  PIR_frekv               ;funkcija, ki vklopi sireno, in sicer z
                                    ;določeno frekvenco
      goto  cakaj_na_init
;-------------------------------------------------------
;inicializacija PIR senzorja DSC (približno 15 sekund)
;-------------------------------------------------------
je_inicializiran
      btfss PIR1,TMR1IF             ;(16-bitni) časovnik TIMER1 prekoračen?
```

```
        return                          ;ne
        bcf   PIR1,TMR1IF               ;da, pobrišemo zastavico
        incf  TMR1overflow,F            ;povečaj »TMR1overflow« (število
                                        ;prekoračitev časovnika TMR1 za ena)
        decfsz      init_timer,F
        return
        movlw .1                        ;inicializacija končana
        movwf inicializacija_PIR        ;»inicializacija_PIR« = 1
        return
```

Frekvenco zvočnika nastavimo s pulzno širinsko modulacijo ali PWM (angl. »Pulse Width Modulation«). Slika 4.6 nakazuje nastanek PWM signala, ki ga opisujejo naslednje enačbe:

- PWM perioda:

$$\mathrm{PWM}_{\mathrm{PERIODA}} = \left[\mathrm{PR}\,2 + 1\right] * 4 * \mathrm{T}_{\mathrm{OSCILATOR}} * \mathrm{TMR}\,2_{\mathrm{PREDMNOŽILNIK}}$$

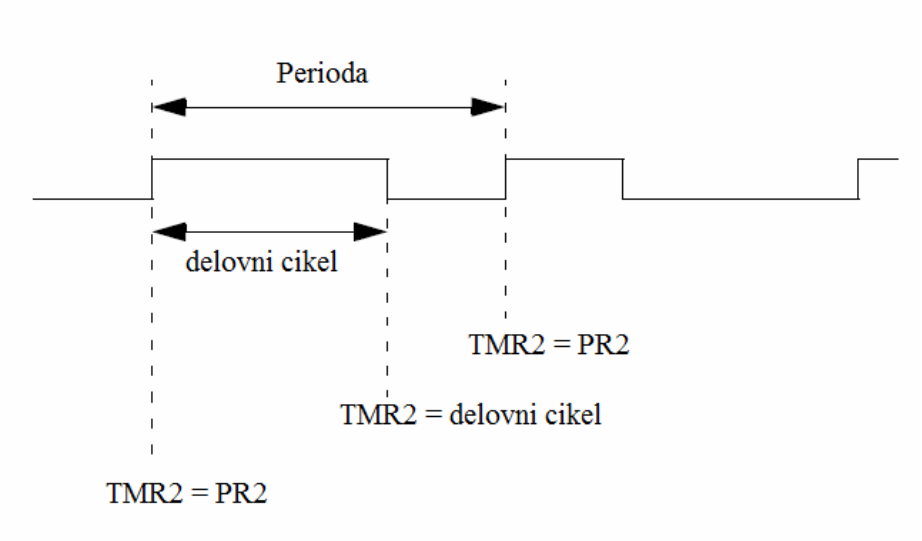
- PWM delovni cikel:

$$\mathrm{PWM}_{\mathrm{DELOVNI\_CIKEL}} = (\mathrm{CCPR}\,1\mathrm{L} : \mathrm{CCP}\,1\mathrm{CON} < 5 : 4 >) * \mathrm{T}_{\mathrm{OSCILATOR}} * \mathrm{TMR}\,2_{\mathrm{PREDMNOŽILNIK}}$$

- PWM resolucija:

$$PWM_{RESOLUCIJA} = \frac{\log\left(\frac{F_{OSCILATOR}}{F_{PWM} * TMR2_{PREDMNOŽILNIK}}\right)}{\log(2)} bitov$$

Pri tem je PR2 – TIMER2 periodni register, $\mathrm{TMR2}_{\mathrm{PREDMNOŽILNIK}}$ – predmnožilna vrednost za časovnik TIMER2, $\mathrm{T}_{\mathrm{OSCILATOR}}$ – časovna baza oscilatorja, $\mathrm{F}_{\mathrm{OSCILATOR}}$ – frekvenčna baza oscilatorja, $\mathrm{F}_{\mathrm{PWM}}$ – frekvenca PWM signala, CCPR1L – nižji bajt PWM registra, CCP1CON – kontrolni register za PWM.



Slika 4.6 PIR modul: PWM modulacija

Frekvence, ki smo jih programsko prednastavili za naš projekt, so prikazane v tabeli 4.7.

Tabela 4.7 PIR modul: Prednastavljene vrednosti frekvence

št. bita v registru »ALARMI«[13]								frekvenca (Hz)
7	6	5	4	3	2	1	0	
0	0	0	x	x	x	x	x	300
0	0	1	x	x	x	x	x	1000
0	1	0	x	x	x	x	x	1700
0	1	1	x	x	x	x	x	2400
1	0	0	x	x	x	x	x	3000

Katero frekvenco bomo nastavili na sireni, nam podajajo najvišji 3 biti v sprejetem bajtu alarmov. Frekvenco nastavimo s funkcijo »PIR_frekv« (primer 4.52).

Primer 4.52 PIR modul: nastavitev frekvence sirene, funkcija »PIR_frekv«

```
PIR_frekv
PIR_frekv300
      clrw
      subwf frekvenca,W             ;register z maskiranimi biti za
                                    ;frekvenco
```

[13] x = karkoli

```
      btfss STATUS,Z
      goto  PIR_frekv1000
      bsf   STATUS,RP0              ;300 Hz
      movlw .207
      movwf PR2                     ;PWM register za periodo
      bcf   STATUS,RP0
      movlw b'10000000'             ;nastavitve za TIMER2 (za PWM)
      movwf CCPR1L                  ;nastavimo duty cycle za PWM (na eno
                                    ;polovico)
      bsf   T2CON,TMR2ON            ;vklop TIMER2 z bitom TMR2ON
      movlw .15
      movwf zvocnik_pomozni
      movlw .1
      movwf zvocnik3s               ;sprožimo sireno (za 3 sekunde) s to
                                    ;frekvenco
      return
PIR_frekv1000
      movlw b'00100000'
      subwf frekvenca,W
      btfss STATUS,Z
      goto  PIR_frekv1700
      bsf   STATUS,RP0              ;1000 Hz
      movlw .62
      movwf PR2                     ;PWM register za periodo
      bcf   STATUS,RP0
      movlw b'00100000'             ;nastavimo duty cycle za PWM (na eno
                                    ;polovico)
      movwf CCPR1L
      bsf   T2CON,TMR2ON            ;vklop TIMER2 z bitom TMR2ON
      movlw .15
      movwf zvocnik_pomozni
      movlw .1
      movwf zvocnik3s               ;sprožimo sireno (za 3 sekunde) s to
                                    ;frekvenco
      return
PIR_frekv1700
      movlw b'01000000'
      subwf frekvenca,W
      btfss STATUS,Z
      goto  PIR_frekv2400
      bsf   STATUS,RP0              ;1700 Hz
      movlw .36
      movwf PR2                     ;PWM register za periodo
      bcf   STATUS,RP0
      movlw b'00010010'
      movwf CCPR1L
      bsf   T2CON,TMR2ON
      movlw .15
      movwf zvocnik_pomozni
      movlw .1
      movwf zvocnik3s
      return
PIR_frekv2400
      movlw b'01100000'
```

```
      subwf frekvenca,W
      btfss STATUS,Z
      goto  PIR_frekv3100
      bsf   STATUS,RP0              ;2400 Hz
      movlw .25
      movwf PR2                     ;PWM register za periodo
      bcf   STATUS,RP0
      movlw b'00001100'
      movwf CCPR1L
      bsf   T2CON,TMR2ON
      movlw .15
      movwf zvocnik_pomozni
      movlw .1
      movwf zvocnik3s
      return
PIR_frekv3100                       ;privzeta vrednost
      bsf   STATUS,RP0
      movlw .19
      movwf PR2                     ;PWM register za periodo
      bcf   STATUS,RP0
      movlw b'00001000'
      movwf CCPR1L
      bsf   T2CON,TMR2ON
      movlw .15
      movwf zvocnik_pomozni
      movlw .1
      movwf zvocnik3s
      return
```

Ko je modul aktiviran, preverjamo odziv senzorja. Če se le-ta sproži (logično nizek nivo), glede na odgovarjajoče nastavitve vklopimo opozorilne znake – luči in/ali sireno. Pošlje se še podatkovni paket do gospodarja (le-ta obvesti uporabnika preko SMS sporočila in prikaza na LCD zaslonu). Spremenljivka »alarm_sprejet« označuje, ali je *MASTER modul* že sprejel trenutni alarm ali ne. Programsko logiko prikazuje primer 4.53.

Primer 4.53 PIR modul: alarmiranje in obveščanje uporabnika

```
komunikacija1
      btfsc inicializacija_PIR,0
      goto  init_konec              ;inicializacija senzorja je potekla,
                                    ;senzor je pripravljen
      call  je_inicializiran        ;inicializacija senzorja še ni potekla
      goto  init_2
;-----------------------------------
init_2
      btfsc zvocnik3s,0
      call  zvocnik
```

```
      call  sprejem_serijski
      movlw klici_PIR               ;splošni paket za komunikacijo?
      subwf bajt,W
      btfss STATUS,Z
      goto  komunikacija2           ;pregladamo, če je uporabnik izklopil
                                    ;sistem (prejetje ukaza
                                    ;»PIR_senzor_IZKLOPI_VSE_RX«)
      goto  je_alarm?
;-----------------------------------
je_alarm?
      btfss zvocnik3s,0             ;smo v stanju alarma?
      goto  ni_alarma               ;ne
      btfss alarm_sprejet,0         ;da, smo obvestili MASTER modul?
      goto  poslji_do_MASTER
      movlw PIR_senzor_OK_TX        ;samo potrdimo signal (ne obveščamo
                                    ;MASTER modula še enkrat o alarmu, ki ga
                                    ;je že sprejel)
      movwf bajt
      call  oddaja_serijski
poslji_do_MASTER
      movlw PIR_senzor_alarm_TX     ;PIR alarm
      movwf bajt
      call  oddaja_serijski
      call  sprejem_serijski
      movlw PIR_senzor_alarm_RX     ;MASTER modul potrdil alarm?
      subwf bajt,W
      btfsc STATUS,Z
      goto  je_alarm?
      bsf   alarm_sprejet,0         ;da, MASTER modul je sprejel
                                    ;»PIR_senzor_alarm«
      goto  komunikacija1
ni_alarma
      bcf   PORTA,3                 ;izklopimo luči (če še niso)
      bcf   alarm_sprejet,0
      movlw PIR_senzor_OK_TX        ;vrnemo odziv
      movwf bajt
      call  oddaja_serijski
      goto  komunikacija1           ;nazaj na začetek
```

4.2.7 TEMP modul

Modul skrbi za ohranjanje primerne temperature v zbiralniku vode (lahko tudi za centralno kurjavo in ogrevanje). Na njega je povezan digitalni termometer DS18B20, ki opravlja konverzijo temperature v digitalno (binarno) obliko. Izmerjene in nastavljene

vrednosti temperature se v/iz PIC-a pošljejo/prejmejo kot ASCII znaki števil. Potek delovanja prikazujeta diagrama **Napaka! Vira sklicevanja ni bilo mogoče najti.** in 4.16.

Diagram 4.15 TEMP modul 1/2

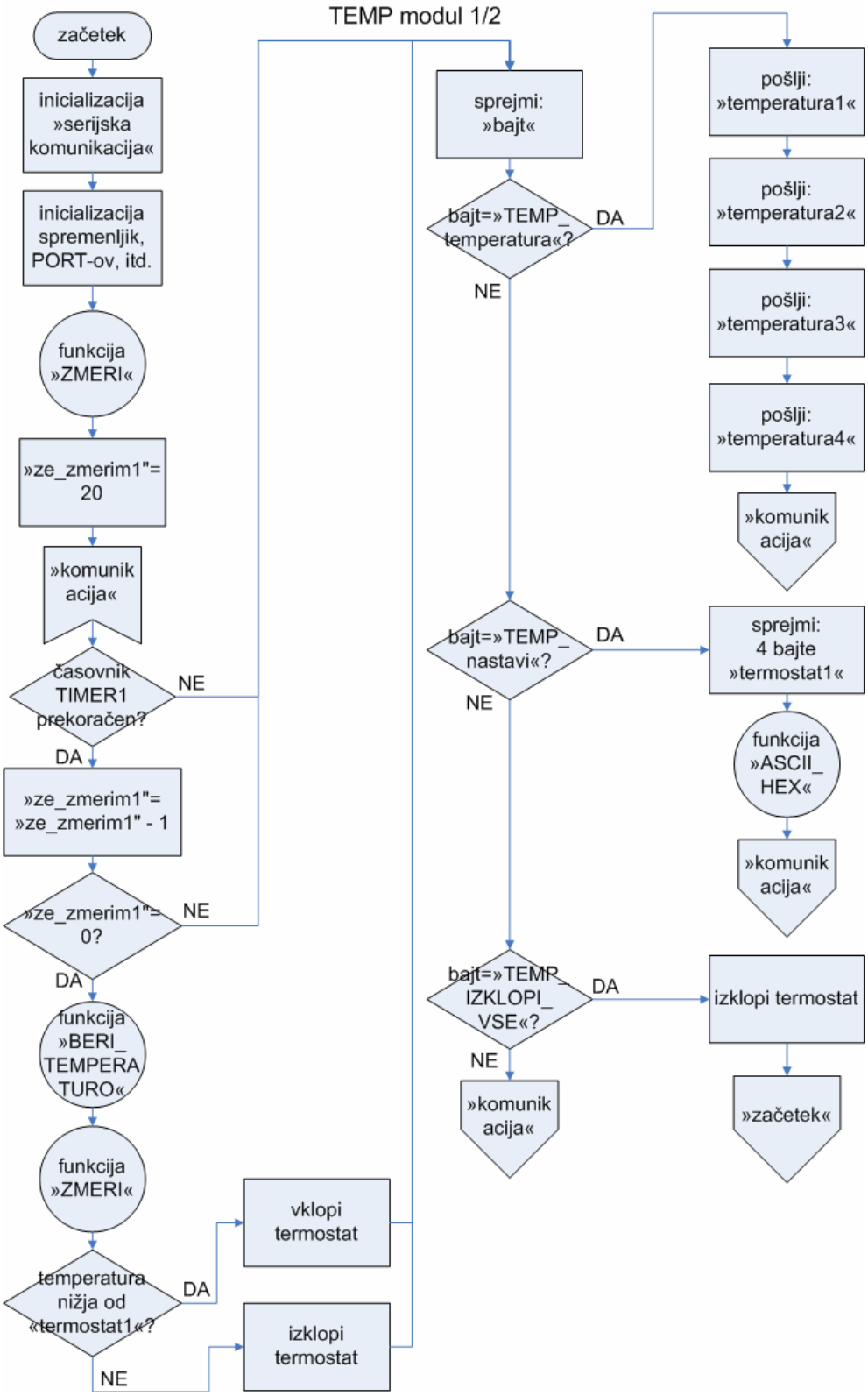
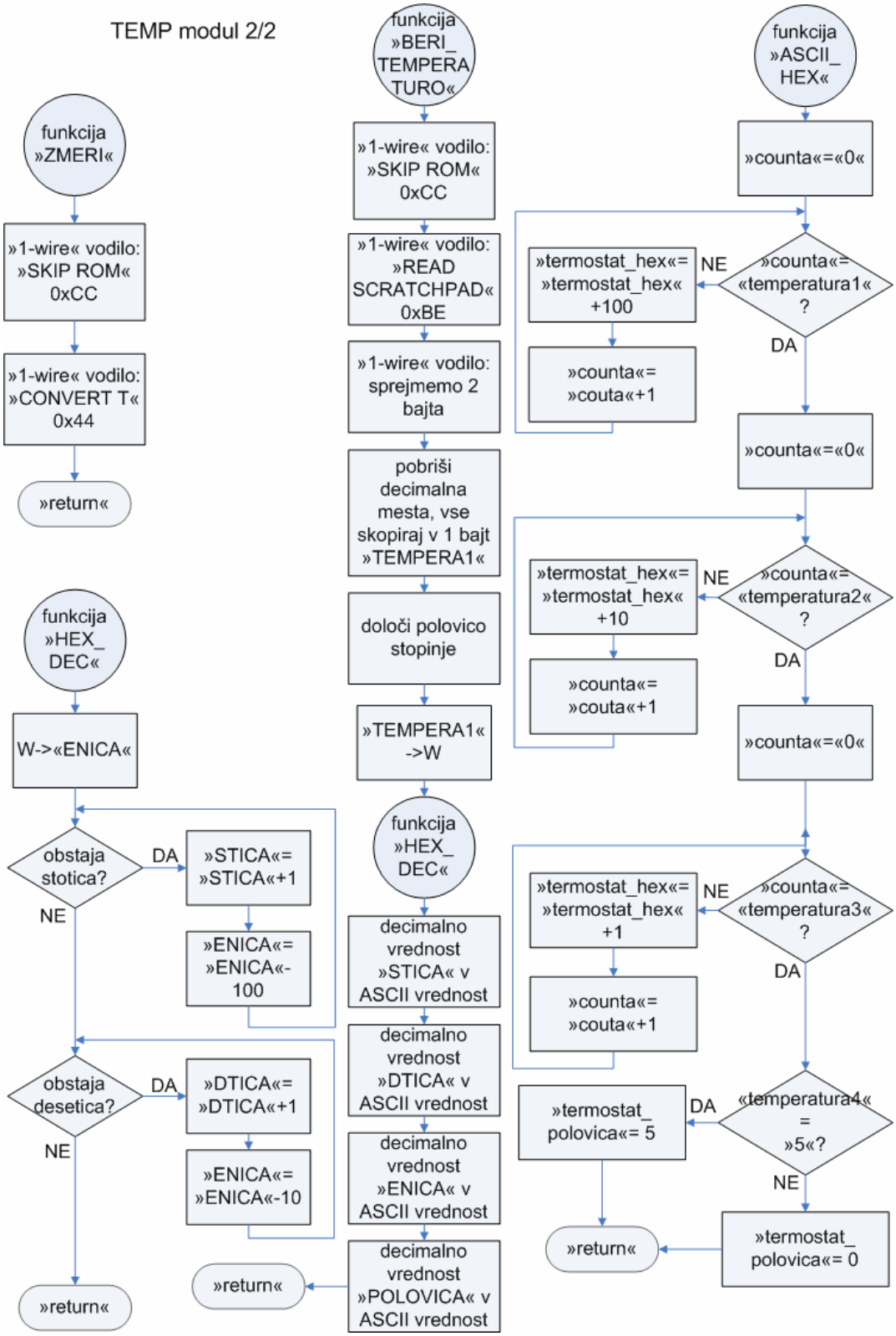


Diagram 4.16 TEMP modul 2/2



Vlogo in opis RAM registrov prikazuje primer 4.54. Uporabili smo PIC16F628A, ki je zadostoval z 2 kB spomina.

Primer 4.54 TEMP modul: RAM registri

```
      list p=16f628a

      #include "p16f628a.inc"       ;standardna knjižnica za PIC16F628A
      #include "serijska_komunikacija_macro.inc"
                                    ;makro za inicializacijo serijske
                                    :komunikacije do gospodarja (MASTER
                                    ;modul)
      #define     PIN_1WIRE PORTA,3 ;pin z digitalnim termometrom
      #define     RXtris      TRISB,4     ;serijska komunikacija
      #define     TXtris      TRISB,5
      #define     RX          PORTB,4
      #define     TX          PORTB,5

      ERRORLEVEL  0,    -302, -205, -207

      cblock 0x20
      ZNAK
      STEVEC
      T1,T2,T3
      TMP_1WIRE
      DATA_1W
      COUNT_1W
      ENICA,DTICA,STICA
      temperatura1                  ;ASCII vrednost trenutno izmerjene
                                    ;temperature 'X'yz,q °C
      temperatura2                  ;ASCII vrednost trenutno izmerjene
                                    ;temperature x'Y'z,q °C
      temperatura3                  ;ASCII vrednost trenutno izmerjene
                                    ;temperature xy'Z',q °C
      temperatura4                  ;ASCII vrednost trenutno izmerjene
                                    ;temperature xyz,'Q' °C
      TEMPERA1                      ;binarna vrednost izmerjene temperature
;TEMPERA1:
;      ---------------------------------------------------------------
;     | 7. bit  | 6. bit| 5. bit| 4. bit| 3. bit| 2. bit| 1. bit| 0. bit |
;     |predznak |  2^6  |  2^5  |  2^4  |  2^3  |  2^2  |  2^1  |  2^0   |
;      ---------------------------------------------------------------
      TEMPERA2                      ;   -//-
      POLOVICA                      ;določa, ali imamo polovico stopinje ali
                                    ;ne (to je naša največja resolucija)
                                    ;»POLOVICA«= 0: .0°C
                                    ;»POLOVICA«= 5: .5°C
      PREDZNAK                      ;ASCII predznak temperature (torej minus
                                    ;ali presledek)
      termostat1                    ;ASCII vrednost temperature termostata
                                    ;'X'yz,q °C
```

```
      termostat2                    ;ASCII vrednost temperature termostata
                                    ;x'Y'z,q °C
      termostat3                    ;ASCII vrednost temperature termostata
                                    ;xy'Z',q °C
      termostat4                    ;ASCII vrednost temperature termostata
                                    ;xyz,'Q' °C
      termostat_hex                 ;binarna vrednost termostata
      termostat_polovica            ;določa, ali imamo polovico stopinje ali
                                    ;ne (binarni zapis)
      bajt,count,poizkus            ;serijska komunikacija
      counta,countb,count1          ;časovne zakasnitve in zanke
      pomozni                       ;pomožni register
      ze_zmerim1                    ;register določa, če je čas za ponovno
                                    ;merjenje temperature (DS18B20 v
                                    ;skrajnem primeru potrebuje 750 ms za
                                    ;pretvorbo)
      pomozni2                      ;pomožni register za zanke
      endc
```

Da dig. termometer zmeri temperaturo, mu je to potrebno ukazati (ukaz »CONVERT T [0x44]«, poglavje »3.8 Digitalni termometer«). Program vedno čaka vsaj maksimalen čas med posameznimi meritvami (> 750 ms). Meritev sproži funkcija »ZMERI«, branje pomnilnika na termometru pa funkcija »BERI_TEMPERATURO« (primer 4.55).

Primer 4.55 TEMP modul: merjenje in branje temperature, funkciji »ZMERI«, »BERI_TEMPERATURO«

```
ZMERI
      call  RESET1W                 ;resetiramo »1-wire« vodilo
      movlw 0xCC                    ;»SKIP ROM« - naslovimo vse naprave na
                                    ;»1-wire« vodilu
      call  VPIS1W                  ;vpis podatkov (iz registra W) na vodilo
      movlw 0x44                    ;»CONVERT T« (vse naprave naenkrat)
      call  VPIS1W                  ;vpis podatkov (iz registra W) na vodilo
      return
;---------------------------------------
BERI_TEMPERATURO
      call  RESET1W                 ;resetiramo »1-wire« vodilo
      movlw 0xCC                    ;SKIP ROM - naslovimo vse naprave na »1-
                                    ;wire« vodilu
      call  VPIS1W
      movlw 0xBE                    ;»READ SCRATCHPAD« - branje pomnilnika
                                    ;na DS18B20
      call  VPIS1W
      call  BRANJE1W
      movwf TEMPERA1                ;preberemo samo prva dva bajta (to je
                                    ;temperatura v »1-wire DS18B20«
                                    ;specifikaciji)
```

```
      call  BRANJE1W                ;TEMPERA1 - LSB, TEMPERA2 - MSB
      movwf TEMPERA2
      rrf   TEMPERA1,F              ;uporabili bomo samo 9-znakovni prikaz
                                    ;temp. (natančnost 0.5 °C) in znak +/-
      rrf   TEMPERA1,F
      rrf   TEMPERA1,F
      rrf   TEMPERA1,F
      btfss STATUS,C
      movlw 0x00
      btfsc STATUS,C
      movlw .5
      movwf POLOVICA                ;določimo polovico stopinje (5 - obstaja
                                    ;polovica, 0 - ne obstaja)
      bcf   TEMPERA1,7
      bcf   TEMPERA1,6
      bcf   TEMPERA1,5
      bcf   TEMPERA1,4              ;pobrišemo 7., 6., 5., 4. bit (dobimo
                                    ;nedecimalno vrednost)
      btfsc TEMPERA2,0              ;vrednost vpišemo v en samcat bajt
                                    (»TEMPERA1«)
      bsf   TEMPERA1,4
      btfsc TEMPERA2,1
      bsf   TEMPERA1,5
      btfsc TEMPERA2,2
      bsf   TEMPERA1,6
      btfsc TEMPERA2,7              ;preverimo predznak temperature ('S' = 0
                                    ;pozitivna temp., 'S'=1 negativna temp.)
      goto  MINUS
;pozitiven predznak
      movlw .10                     ;presledek v »TABELA2«
      movwf PREDZNAK
      bcf   TEMPERA1,7
      goto  BERI_TEMPERATURO_2
;negativen predznak
MINUS
      movlw .11                     ;minus v »TABELA2«
      movwf PREDZNAK
      bsf   TEMPERA1,7              ;če je 7. bit v »TEMPERA1« enak '1',
                                    ;potem je temperatura negativna
BERI_TEMPERATURO_2
      movf  TEMPERA1,W
      call  HEX_DEC                 ;rutina pretvori HEX(BIN) število v DEC
                                    ;v »STICA« dobimo binarno vrednost
                                    ;stotic, v »DTICA« desetic in »ENICA«
                                    ;enic
;--------------------------
;stotice
;--------------------------
      movf  STICA,W
      btfsc STATUS,Z
      movlw .0                      ;"0" v »TABELA2«
      call  TABELA2
      movwf temperatura1
;--------------------------
```

```
;desetice
;-------------------------
      movf  DTICA,W
      call  TABELA2
      movwf bajt
      movwf temperatura2
;-------------------------
;enice
;-------------------------
      movf  ENICA,W
      call  TABELA2
      movwf bajt
      movwf temperatura3
;-------------------------
;decimalka 0.5 °C
;-------------------------
      movf  POLOVICA,W
      call  TABELA2
      movwf bajt
      movwf temperatura4
      return
;-----------------------------------
;»TABELA2«- tabela za klicanje znakov
;-----------------------------------
TABELA2
      addwf PCL,f
      DT "0123456789 -"
```

Prenos bajta po »1-wire« vodilu dosežemo s procedurama »BRANJE1W« in »VPIS1W«. Posamezen bit je poslan znotraj »1-wire« reže (angl. »slot«), in sicer z logično nizkim nivojem na pinu mikrokontrolerja (logična nič) ali stanjem visoke impedance, kjer 4.7 kΩ upor na vodilu potegne (angl. »pull-up«) linijo na stanje V_{PU} (logična enica). Procedure so predstavljene v primeru 4.56 in predstavljajo signale »1-wire« protokola iz slike 3.13.

Primer 4.56 TEMP modul: »1-wire« procedure

```
;-------------------------------------------------
;»RESET1W« - procedura resetira senzor »1-wire«
;-------------------------------------------------
RESET1W
      call  L1WIRE                  ;»1-wire« pin je nizek
      movlw .49
      call  PAVZA1W                 ;približno 60 µS nizek pin
      call  H1WIRE                  ;»1-wire« pin je visok
      movlw .8
      call  PAVZA1W                 ;pavza dolžine približno 420 µs
      movlw .41
```

```
      call  PAVZA1W
      return
;-----------------------------------------------------
;»VPIS1W« - procedura pošlje podatke na »1-wire« vodilo
;-----------------------------------------------------
VPIS1W
      movwf DATA_1W                ;»DATA_1W« paket podatkov, ki jih
                                   ;pošljemo na vodilo
      movlw .8
      movwf COUNT_1W               ;število bitov v paketu
Z_VPIS1W
      call  L1WIRE                 ;nizek nivo
      nop
      nop
      nop
      nop
      nop
      rrf   DATA_1W,f              ;rotiramo paket, da dosežemo vsakega
                                   ;od bitov v paketu
      btfsc STATUS,C
      call  H1WIRE                 ;visok nivo
      movlw .6
      call  PAVZA1W
      call  H1WIRE
      decfsz      COUNT_1W,f
      goto  Z_VPIS1W               ;ponovi
      return
;-----------------------------------------------------------
;»BRANJE1W« - procedura prebere podatke z vodila »1-wire«
;-----------------------------------------------------------
BRANJE1W
      clrf  DATA_1W
      movlw .8
      movwf COUNT_1W
Z_BRANJE_1W
      call  L1WIRE
      nop
      nop
      nop
      nop
      nop
      nop
      call  H1WIRE
      nop
      nop
      nop
      nop
      clrc                         ;pobriši »carry« zastavico v »STATUS«
                                   ;registru
      btfsc PIN_1WIRE
      setc
      rrf   DATA_1W,f
      movlw .6
      call  PAVZA1W
```

```
      decfsz      COUNT_1W,f
      goto  Z_BRANJE_1W
      movf  DATA_1W,W
      return
;-------------------------------------------------------------------
;»L1WIRE« - procedura preklopi »1-wire« pin na izhod z nizkim stanjem
;-------------------------------------------------------------------
L1WIRE
      bcf   PIN_1WIRE
      bsf   STATUS,RP0
      bcf   PIN_1WIRE
      bcf   STATUS,RP0
      return
;-----------------------------------------------------
;»H1WIRE« - procedura preklopi »1-wire« pin na vhod
;-----------------------------------------------------
H1WIRE
      bsf   STATUS,RP0
      bsf   PIN_1WIRE
      bcf   STATUS,RP0
      return
;-----------------------------------------------------------
;»PAVZA1W« - procedura definira pavzo približne dolžine W*10
;-----------------------------------------------------------
PAVZA1W
      movwf TMP_1WIRE
ZP1W                                ;št. inštrukcijskih ciklov:
      nop                           ;     - 7 x »nop«
      nop                           ;     - 1 x »decfsz«
      nop                           ;     - 2 x »goto«
      nop                           ;SKUPAJ: 10 inštrukcijskih ciklov
      nop
      nop
      nop
      decfsz      TMP_1WIRE,f
      goto  ZP1W
      return
```

Po branju binarne vrednosti temperature, le-to pretvorimo v ASCII obliko (4 znaki). Funkcija »HEX_DEC« nam heksadecimalno obliko zapisa razčleni v tri registre glede na desetiško potenco, in sicer:

- stotice (10^2) v register »STICA«,
- desetice (10^1) v register »DTICA« in
- enice (10^0) v register »ENICA«.

Kasneje te vrednosti kličemo iz tabele ASCII znakov števk. Obratni postopek se zahteva po sprejetju »termostat temperature« (funkcija »ASCII_HEX«). Obe pretvorbi (pripadajoči funkciji) kaže primer 4.57.

Primer 4.57 TEMP modul: pretvorba heksadecimalnih števil v ASCII in obratno, funkciji »HEX_DEC«, »ASCII_HEX«

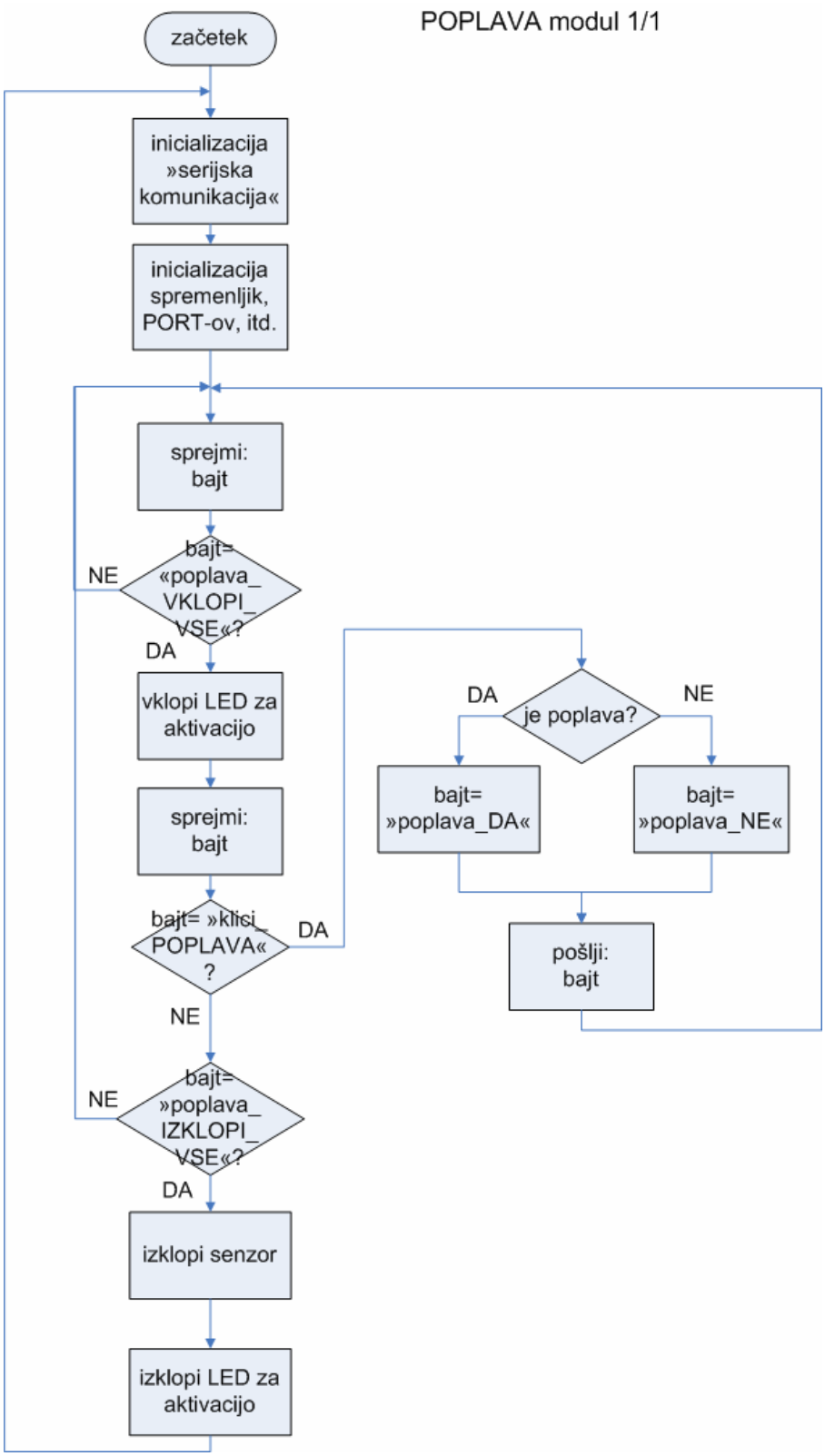
```
;-----------------------------------------------------
;»HX_DEC« - iz binarne vrednosti iz W v decimalne potence
;-----------------------------------------------------
HEX_DEC                         ;kot argument potrebuje binarno vrednost
                                ;v splošnem registru WREG
      movwf ENICA
      clrf  STICA
      clrf  DTICA
NA_STOTICO
      movlw .100
      subwf ENICA,W             ;od registra »ENICA« odštejemo register
                                ;W
      btfss STATUS,C
      goto  NA_DESETICO
      incf  STICA,F             ;stotica = stotica + 1
      movwf ENICA
      goto  NA_STOTICO
NA_DESETICO
      movlw .10
      subwf ENICA,W
      btfss STATUS,C
      return
      incf  DTICA,F             ;desetica = desetica + 1
      movwf ENICA
      goto  NA_DESETICO
;-----------------------------------------------------
;«ASCII_HEX« - iz ASCII znakov v binarno vrednost
;-----------------------------------------------------
ASCII_HEX
      movlw '0'                 ;ASCII = "0" (s tem znakom začnemo)
      movwf counta              ;pomožna spremenljivka
STOTICE
      movf  counta,W
      subwf termostat1,W
      btfsc STATUS,Z            ;stotica?
      goto  NA_DESETICE         ;ne (več)
      movlw .100                ;da
      addwf termostat_hex,F
      incf  counta,F            ;povečamo counta, najprej na "1", nato
                                ;"2" itd.
      goto  STOTICE
NA_DESETICE
      movlw '0'                 ;ASCII = "0"
      movwf counta              ;pomožna spremenljivka
```

```
DESETICE
      movf  counta,W
      subwf termostat2,W
      btfsc STATUS,Z                ;desetica?
      goto  NA_ENICE                ;ne (več)
      movlw .10                     ;da
      addwf termostat_hex,F
      incf  counta,F                ;povečamo counta, najprej na "1", nato
                                    ;"2" itd.
      goto  DESETICE
NA_ENICE
      movlw '0'                     ;ASCII = "0"
      movwf counta                  ;pomožna spremenljivka
ENICE
      movf  counta,W
      subwf termostat3,W
      btfsc STATUS,Z                ;enice?
      goto  NA_POLOVICE             ;ne
      movlw .1                      ;da, povečaj za ena
      addwf termostat_hex,F
      incf  counta,F
      goto  ENICE
NA_POLOVICE
      movlw '5'
      subwf termostat4,W            ;polovica stopinje (.5 °C)?
      btfsc STATUS,Z                ;če karkoli drugega kot "5"(= ASCII),
                                    ;obravnavamo kot "0"
      movlw .5                      ;polovica stopinje!
      btfss STATUS,Z
      movlw .0
      movwf termostat_polovica      ;sedaj smo zaključili, v »termostat_hex«
                                    ;je shranjena binarna vrednost
      return
```

4.2.8 POPLAVA modul

Inteligentna hiša ne zaznava samo tistih alarmnih situacij, povzročenih z zunanjimi dejavniki (npr. vstop vlomilca v objekt), temveč tudi kontrolira notranje stanje hiše – *TEMP modul* vzdržuje temperaturo vode/zraka, *POPLAVA modul* obvesti uporabnika v primeru poplave v hiši (npr. zaradi počene vodovodne cevi). *POPLAVA modul* je sprogramiran kot nadzorovalec v kopalnici, vendar ga je z majhnimi spremembami mogoče preprogramirati v senzor odprtih vrat/oken. Ker je veliko nožic še prostih (12), lahko realiziramo tudi več senzorjev z istim PIC-em. Potek delovanja prikazuje diagram 4.17.

Diagram 4.17 POPLAVA modul 1/1



Nabor možnih ukazov je skop, vendar zadovoljujejo svojemu namenu. Možni so naslednji komunikacijski bajti:

- »poplava_VKLOPI_VSE«: vklopimo modul, ki začne nadzorovati prostor,
- »poplava_NE«: pin senzorja na logično nizkem nivoju (tj. ni alarma),
- »poplava_DA«: pin senzorja na logično visokem nivoju (tj. sprožimo alarm),
- »poplava_IZKLOPI_VSE«: postavimo modul v stanje pripravljenosti (senzorji deaktivirani).

Za serijsko komunikacijo je uporabljena že omenjena rutina, prav tako za časovne zakasnitve (glej nazaj). Program prikazuje primer 4.58.

Primer 4.58 POPLAVA modul: nadzor senzorjev

```
START                               ;začetek programa
            movlw 0x07
            movwf CMCON             ;izklop komparatorjev (niso potrebni)
            bsf   STATUS,RP0
            movlw b'00000000'
            movwf TRISB
            movwf TRISA
            movlw b'00001000'
            movwf OPTION_REG        ;interni izvor ure za časovnik TIMER0,
                                    ;predmnožilnik nastavljen na WDT, »pull-
                                    ;up« upori nastavljeni na PORTB
            movlw b'00100000'
            movwf INTCON
            bcf   STATUS,RP0
            init_serijski     RXtris,TXtris     ; inicializacija »serijska
                                                ;komunikacija«: RX= RB4 TX=
                                                ;RB5
cakaj_na_init                       ;smo v stanju pripravljenosti
            call  sprejem_serijski
            movlw poplava_VKLOPI_VSE_RX
            subwf bajt,W            ;vklopimo modul?
            btfss STATUS,Z
            goto  cakaj_na_init     ;ne
            movlw b'10100111'       ;da, vrnemo odziv
            movwf bajt
```

```
            call  oddaja_serijski
komunikacija
            call  sprejem_serijski
            movlw klici_POPLAVA     ;MASTER modul poziv?
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija2     ;ne
            movlw javi_MASTER       ;da
            movwf bajt
            btfsc ALARM             ;preverimo stanje senzorja poplave
            movlw poplava_DA        ;alarm!
            btfss ALARM
            movlw poplava_NE        ;ni alarma
            iorwf bajt,F
            call  oddaja_serijski
            goto  komunikacija
komunikacija2
            movlw poplava_IZKLOPI_VSE_RX
            subwf bajt,W
            btfss STATUS,Z
            goto  komunikacija      ;ni primernega odziva (ali ga sploh ni)
            movlw poplava_IZKLOPI_VSE_TX  ;izklopimo modul in gremo na
                                          ;začetek
            movwf bajt
            call  oddaja_serijski
            goto  START
```

5 SKLEP

Prvi del diplomske naloge je obsegal teoretični opis pojma inteligentne hiše in njene prednosti za uporabnika – tako z ekonomskega vidika kot tudi z vidika zagotavljanja kakovosti bivanja. Predstavili smo nekatere bolj uveljavljene produkte, ki jih lahko uvrščamo v kategorijo inteligentnih sistemov, ki so trenutno na tržišču. Proizvajalcev takšnih sistemov je na svetu veliko, čeprav pri nas pojem inteligentne hiše še ni dobro razširjen in je slovenskih ponudnikov bore malo (prevladujejo Dometra, KomfortKlik in Electes[14]). Pojavljajo se sicer izdelki, ki kažejo določeno stopnjo inteligence (torej sprejemajo parametre iz okolice ter se odzivajo nanje v skladu z določenimi algoritmi), vendar so le-ti načeloma namenjeni zgolj opravljanju določenih nalog, recimo samo varovanju, samo zbiranju vremenskih podatkov, samo kontroli električne porabe in podobno. Za razliko od tega ponuja pametna hiša medsebojno povezavo naštetih nalog ter dobro izgrajeno komunikacijsko mrežo. Laiku se investicija mogoče zdi neprimerna (cenovni razred od nekaj sto do nekaj tisoč evrov), vendar se po izračunu že samo na primeru varčevanja s toplotno energijo (vklapljanje in izklapljanje ogrevalnega sistema) letni prihranek dvigne za več kot tretjino. Tako dosežemo povrnitev stroškov investicije v manj kot letu in pol[19].

Drugi, večji del diplomskega dela smo posvetili izdelavi dejanskega primerka sistema inteligentne hiše. Uporabili smo mikrokrmilnike, ki so se nam zdeli najprimernejša oblika kontrolnega vezja, saj imajo vgrajeno večino periferije, ki smo jo pri projektu potrebovali. Dodatno smo vključili vse potrebne senzorje ter jim vhode in izhode prilagodili na napetostne nivoje, ki jih razume mikrokrmilnik. Programsko kodo smo vpisovali v zbirniku, ki je pogosta izbira programskega jezika za mikrokrmilnike z majhno količino pomnilniških lokacij. Če bi programe kreirali v ANSI C programskem jeziku, bi potrebovali dva- do trikrat več pomnilnika[20]. S tem smo izboljšali cenovno dostopnost za uporabnike, v primeru da takšen izdelek ponudimo na tržišču. Zbirnik je prišel prav še zato, ker imamo veliko časovno relevantnih procedur (npr. inicializacijo LCD zaslona in digitalnega termometra), ki jih

[14] Podjetje Dometra: http://www.dometra.si,
podjetje KomfortKlik: http://www.comfortclick.com,
podjetje Electes: http://www.electes.com.

enostavno napišemo, saj vemo, da vsak ukaz v zbirniku določa točno določeno število ciklov oscilatorja.

Možne izboljšave izdelanega prototipa, ki so se nam še porodile, so recimo: večje število funkcij in izpopolnjeni algoritmi delovanja ter protokoli komunikacije, vgradnja neprekinjenega napajanja UPS, dodatni moduli in brezžična komunikacijska mreža.

Težave pri programiranju so pogosto nastale zaradi same kompleksnosti algoritmov, saj zbirnikova koda eksponentno postaja nepregledna in težko berljiva. Tako so bili potrebni dolgotrajni in temeljiti pregledi programske kode ter testiranja na testni plošči (angl. »protoboard«). Pomagali smo si še z osciloskopom, testnimi LED-i in izpisom v terminalsko okno na osebnem računalniku (za prenos RS-232 znakov). Od strojne opreme smo največ težav imeli z mobilnim telefonom, ki se je med obdelovanjem sprejetih podatkov odzival časovno neenakomerno, zato smo bili primorani vključevati velike časovne zakasnitve. Po uspešno sprogramiranih programih na vseh modulih smo izdelali še vzorčno tiskano vezje (v prilogi).

6 LITERATURA IN VIRI

[1] Archiform Ltd., »Bridgeway house renderings and illustrations«, http://house-renderings.archiform3d.com/3d-gallery/06-bridgeway/plan-illustrations/03-bridgeway-plan-illustration.jpg, obiskano: julij 2009

[2] Wikipedia, »LonTalk«, http://en.wikipedia.org/wiki/LonTalk, obiskano: avgust 2009

[3] Wikipedia, »KNX (standard)«, http://en.wikipedia.org/wiki/KNX_(standard), obiskano: avgust 2009

[4] Wikipedia, »X10 (industry standard)«, http://en.wikipedia.org/wiki/X10_(industry_standard), obiskano: avgust 2009

[5] Wikipedia, »ZigBee«, http://en.wikipedia.org/wiki/Zigbee, obiskano: avgust 2009

[6] Microchip, »MPLAB ICD2 In-Circuit Debugger User's Guide«, http://ww1.microchip.com/downloads/en/devicedoc/51331B.pdf, obiskano: julij 2008

[7] Ian Harries, »HD44780 Instruction Set«, http://www.doc.ic.ac.uk/~ih/doc/lcd/instruct.html, obiskano: julij 2008

[8] Siemens, »AT Command Set«, http://www.sendsms.cn/download/tc35i_atc_v0103_1073581.pdf, obiskano: julij 2008

[9] Maxim, »+5C-Powered, Multichannel RS-232 Drivers/Receivers«, http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf, obiskano: avgust 2009

[10] Wikipedia, »Passive infrared sensor«,
http://en.wikipedia.org/wiki/Passive_infrared_sensor,
obiskano: julij 2009

[11] AARtech Canada, »LC-104-PIMW-W- DSC PIR and Microwave Motion Detector«,
http://www.aartech.ca/images/cache/193c02be08789163b387b32bf1893d43.jpg,
obiskano: julij 2009

[12] Maxim, »DS18B20 Programmable Resolution 1-Wire Digital Thermometer«,
http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf,
obiskano: april 2009

[13] Microchip, »MPLAB Integrated Development Environment«,
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002,
obiskano: julij 2008

[14] Microchip, »PIC 16F627A/628A/648A Data Sheet«,
http://ww1.microchip.com/downloads/en/DeviceDoc/40044F.pdf,
obiskano: junij 2008

[15] Microchip, »AN556 Implementing a Table Read«,
http://ww1.microchip.com/downloads/en/AppNotes/00556e.pdf,
obiskano: avgust 2008

[16] Ian Harries, »Initialisation of the HD44780«,
http://www.doc.ic.ac.uk/~ih/doc/lcd/initiali.html,
obiskano: julij 2008

[17] Microchip, »PICmicro™ Mid-Range MCU Family Reference Manual, Section 7. Data EEPROM«,
http://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf,
obiskano: november 2008

[18] Lars Pettersson, »SMS messages and the PDU format«,
http://www.dreamfabric.com/sms/,
obiskano: junij 2009

[19] Aljoša Doberšek, »Upravljanje in nadzor inteligentne hiše«,

http://lpa.feri.uni-mb.si/Diplome/AljosaDobersek.pdf,
obiskano: januar 2009

[20] Michael L. Monaghan, »Is the C or Assembly Programming Language Better for Programing PIC Microcontrollers?«,
http://www.usc.edu/CSSF/Current/Projects/J1610.pdf,
obiskano: januar 2009

7 PRILOGE

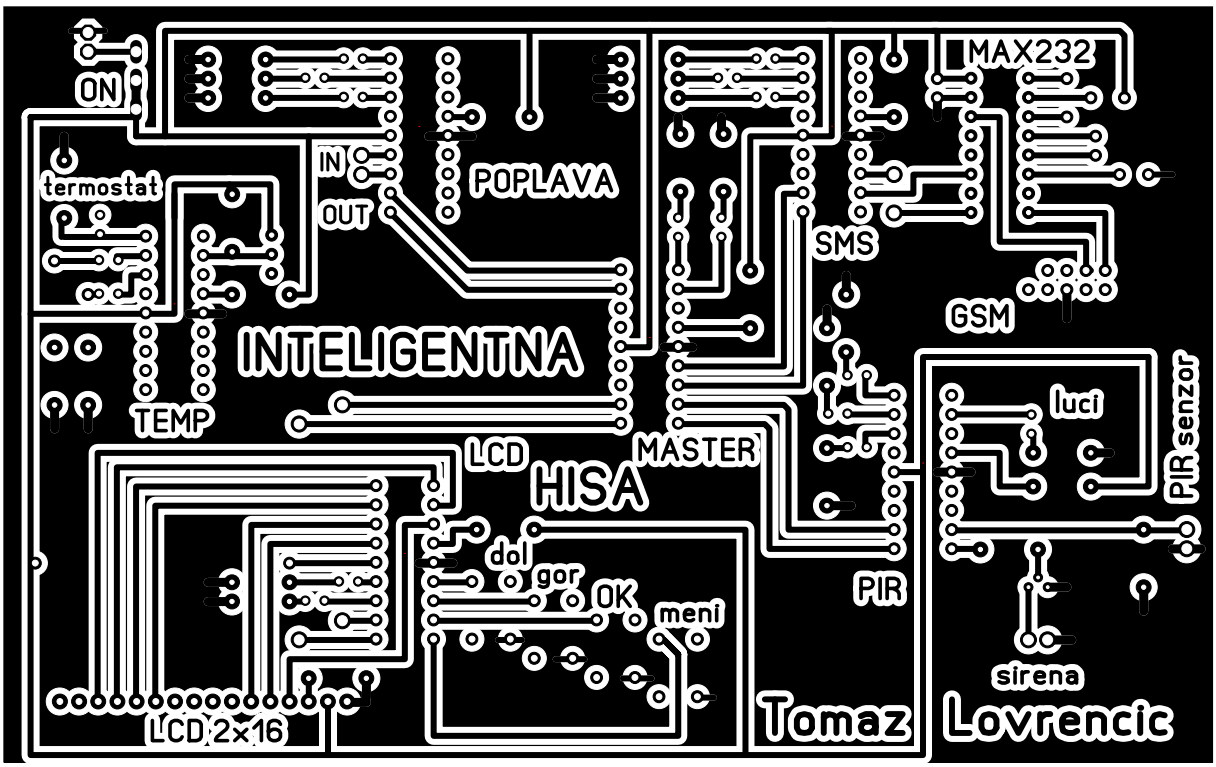
Tabela 7.1 Diagram poteka: legenda

Simbol	Opis
	Proces, korak v programu, dejanje, inicializacijski postopek.
	Odločitveno drevo. Preverjanje vpisanega pogoja.
	Klic funkcije, povezava za izvajanje vmesnega programa.
	Vhodno/izhodna operacija.
	Referenca na drugi list.
	Referenca z drugega lista, nadaljevanje programa s prejšnjega lista.
	Oznaka (angl. »label«).
	Začetek ali konec programa. Konec funkcije: »RETURN«, začetek funkcije: »začetek«.

Tabela 7.2 Strojna oprema: kosovnica

Količina (kosov)	Del	Opombe
3	Microchip PIC 16F648A PDIP	18-pinski mikrokontroler v plastičnem dvovrstičnem ohišju (»PDIP«), *http://ww1.microchip.com/downloads/en/DeviceDoc/40044F.pdf*
3	Microchip PIC 16F628A PDIP	18-pinski mikrokontroler v plastičnem dvovrstičnem ohišju (»PDIP«), *http://ww1.microchip.com/downloads/en/DeviceDoc/40044F.pdf*
1	Dallas DS18B20	digitalni termometer, *http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf*
1	DSC LC-100-PI	komercialni PIR senzor, *http://www.dsc.com/index.php?n=products&o=view&id=93*
1	MAX232	+5 V gonilnik za RS-232 signale, *http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf*
1	Siemens RS232 podatkovni kabel	*http://pinouts.ru/CellularPhones-P-W/siemens_c25_s25_pinout.shtml*
1	Siemens M50	GSM aparat, *http://safemanuals.com/user-guide-instructions-owner-manual/SIEMENS/M50-_E*
1	konektor GSM	
1	LCD zaslon	LCD zaslon 2 × 16 alfanumeričnih znakov *http://www.cct.com.my*
12	LED	svetleče diode za indikacijo stanja modula
12	upor 180 Ω	za LED omejitev napetosti iz +5 V
6	upor 10 kΩ	»RESET« za mikrokontroler PIC
2	upor 4.7 kΩ	»PULL UP« ali »PULL DOWN«
1	senzor poplava	2 nestaknjeni žici
1	LM7805	stabilizator napetosti +5 V, *http://www.datasheetcatalog.org/datasheet/fairchild/LM7805.pdf*
4	mikro tipka	tipka za tiskano vezje
1	potenciometer 4.7 kΩ	linearni potenciometer za nastavitev kontrasta LCD zaslona
4	kondenzator 1 μF	elektrolitski kondenzator

Tiskanina sistema inteligentne hiše (razmerje 1:1)



Slika modela inteligentne hiše