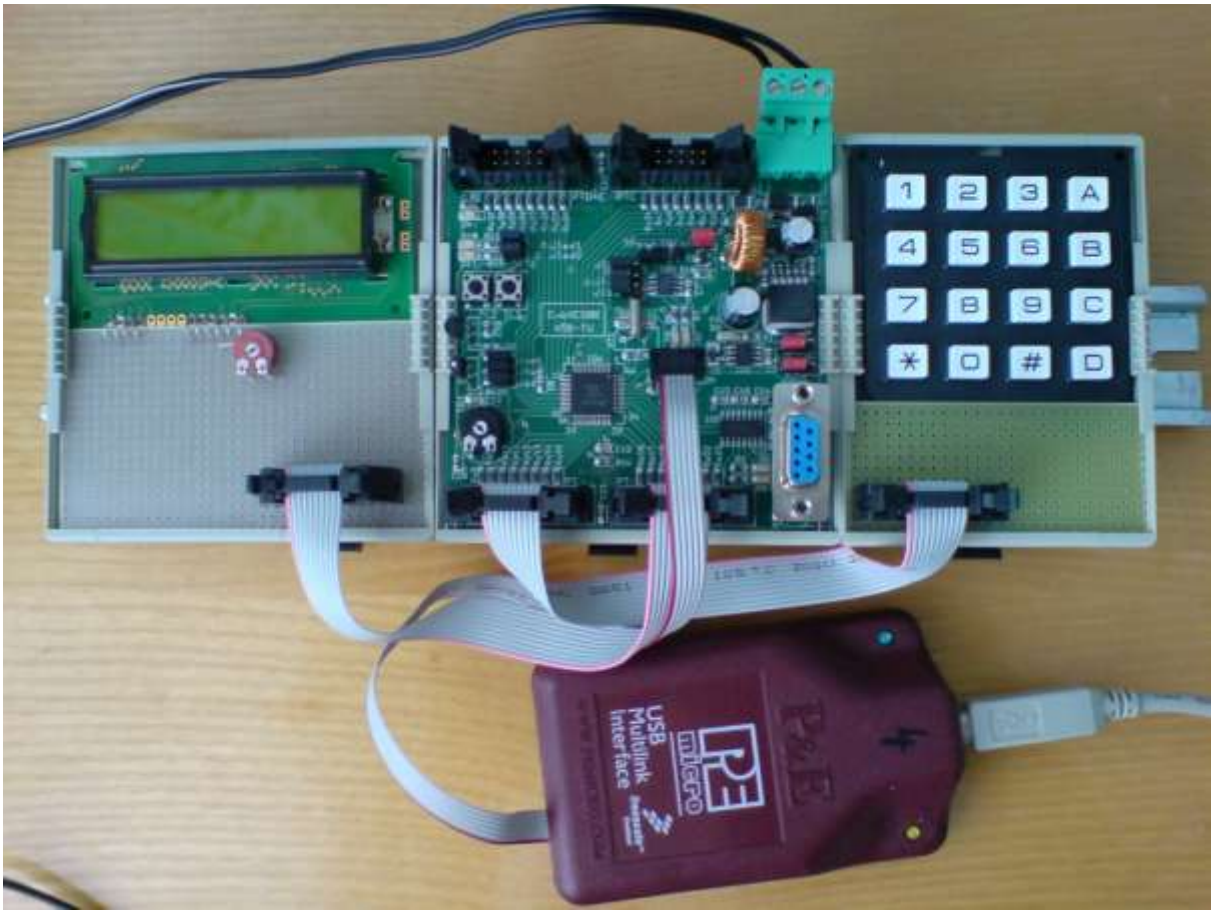
Ústav počítačové a řídicí techniky VŠCHT Praha

MIKROPROCESORY



Obr. 1 Celkový přehled

OBSAH

1 ÚVOD

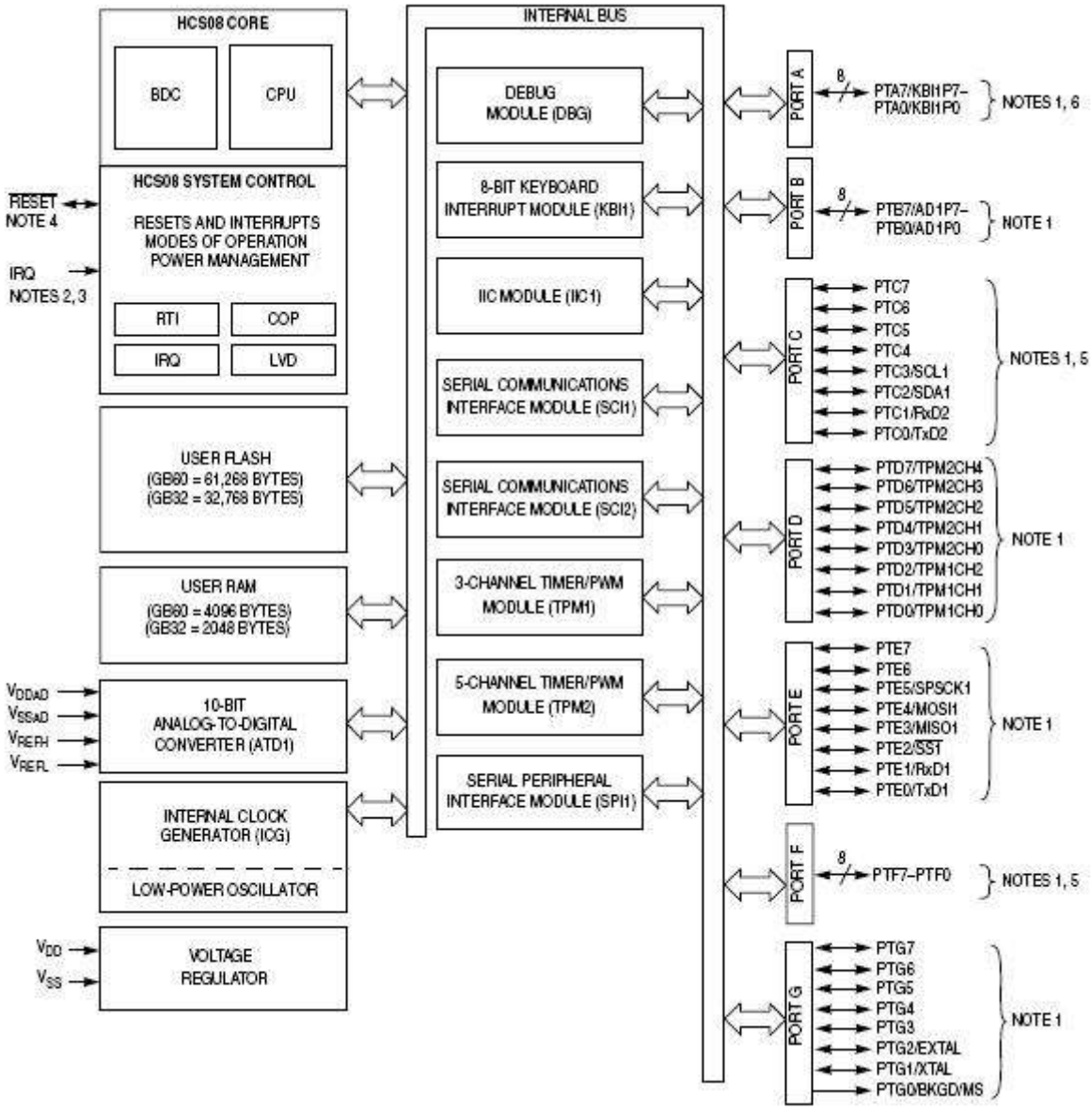
Hlavním cílem vaší práce bude využití mikroprocesoru pro odečítání analogové hodnoty, teploty nebo vlhkosti a její zobrazování na displej nebo odesílání do počítače přes sériovou linku. Každý student nejprve vytvoří prázdný program, který postupně rozšíří o vysílání dat po sériové lince nebo jejich zobrazování na displej. Nakonec bude program rozšířen o odečítání analogové hodnoty, vlhkosti nebo teploty. Přiřazení konkrétních úkolů bude provedeno v předstihu. Zadání volitelných parametrů proběhne až v laboratoři. Vypracování předpokládá základní znalosti práce s mikroprocesory a základy jazyka C. Především se však očekává aktivní práce s teoretickou částí.

2 TEORETICKÁ ČÁST

2.1 MIKROPROCESOR MC9S08GT60

2.1.1 Obecný popis

Mikroprocesor MC9S08GT60 je založen na architektuře typu von Neumann. Základem je jádro HCS08. Paměť se dělí na RAM a FLASH. Obsahuje interní zdroj hodinového signálu, analogově-digitální převodník, časovač, moduly pro sériovou komunikaci a další prvky.

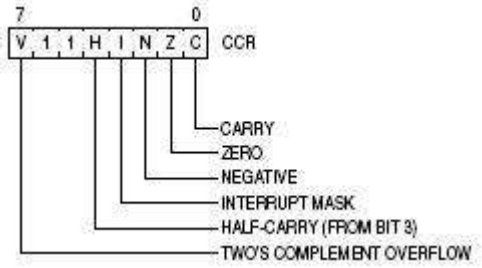


Obr. 2 Schéma MC9S08GT60

2.1.2 Jádro HCS08

Jádro se skládá s centrální procesorové jednotky CPU a kontroléru pro ladění programu BDC. Procesor obsahuje osmibitový akumulátor A, registr příznaků CCR (Obr. 3), šestnáctibitoví

ukazatel zásobníku SP, šestnáctibitový programový čítač PC obsahující adresu následující instrukce a šestnáctibitový indexoví registr HX adresující 64 KB paměti.



Obr. 3 registr příznaků CCR

Obsahuje také soubor adresovacích módů (Tab. I).

Adresování	**obsah**
Inherentní	pouze čtení konstanty bez adresování
Bezprostřední	cílová adresa (8 bitů)
Bezprostřední	cílová adresa (16 bitů)
Přímé	adresa (8 bitů) buňky s cílovou adresou
Rozšířené	adresa (16 bitů) buňky s cílovou adresou
Indexové	cílová adresa posunuta od aktuální o indexový registr (16 bitů)
Indexové s postinkrementem	cílová adresa posunuta od aktuální o indexový registr (16 bitů) +1
Indexové s offsetem	cílová adresa posunuta od aktuální o indexový registr (16 bitů) a offset (8 bitů)
Indexové s offsetem a postinkremenem	cílová adresa posunuta od aktuální o indexový registr (16 bitů), offset (8 bitů) a +1
Indexové s offsetem	cílová adresa posunuta od aktuální o indexový registr (16 bitů) a offset (16 bitů)
Zásobníkové s offsetem	cílová adresa posunuta od aktuální o hodnotu ze zásobníku (16 bitů) a offset (8 bitů)
Zásobníkové s offsetem	cílová adresa posunuta od aktuální o hodnotu ze zásobníku (16 bitů) a offset (16 bitů)
Relativní adresování	určuje se počáteční i cílová adresa

Tab. I Adresovací metody

Umožňuje přesun dat bez použití akumulátoru, zrychlené násobení a dělení, dva režimy stop a jeden wait. Probuzení je pak možné libovolným přerušením.

2.1.3 Resety a interrupty

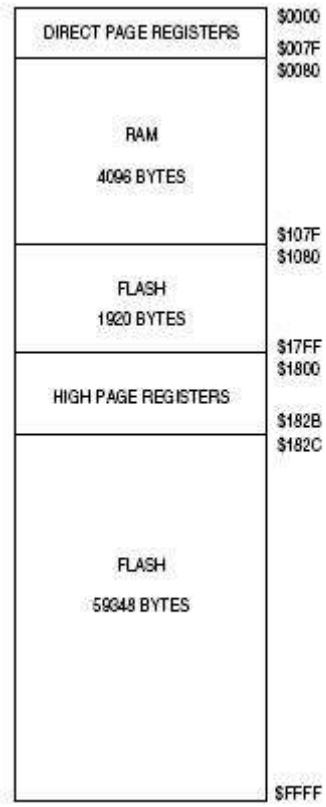
Resety a interrupty způsobí přerušení běhu programu. Při resetu se program ihned zastaví a vrátí do startovních podmínek. Zdroje resetů mohou být externí a interní. Mezi externí patří vypnutí napájení a resetovací pin. Interní reset vyvolá watchdog COP, nízké napětí LVI a chybný kód nebo instrukce. Interrupt program dočasně pozastaví a po odbavení interruptu pokračuje od místa přerušení. Zdroje resetů mohou být hardwarové nebo softwarové. Mezi hardwarové patří pin IRQ, přerušení od I/O portů, A/D převodníku, časovače a sériové komunikace. Softwaroví intrrupt je vykonán jako část instrukčního toku a nemůže být na rozdíl od hardwarového maskován.

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lower	26 through 31	$FFC0/FFC1 through $FFCA/FFCB	Unused Vector Space (available for user program)				
	25	$FFCC/FFCD	Vrti	System control	RTIF	RTIE	Real-time interrupt
	24	$FFCE/FFCF	Viic1	IIC	IICIS	IICIE	IIC control
	23	$FFD0/FFD1	Vatd1	ATD	COCO	AIEN	AD conversion complete
	22	$FFD2/FFD3	Vkeyboard1	KBI	KBF	KBIE	Keyboard pins
	21	$FFD4/FFD5	Vsci2tx	SCI2	TDRE TC	TIE TCIE	SCI2 transmit
	20	$FFD6/FFD7	Vsci2rx	SCI2	IDLE RDRF	ILIE RIE	SCI2 receive
	19	$FFD8/FFD9	Vsci2err	SCI2	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI2 error
	18	$FFDA/FFDB	Vsci1tx	SCI1	TDRE TC	TIE TCIE	SCI1 transmit
	17	$FFDC/FFDD	Vsci1rx	SCI1	IDLE RDRF	ILIE RIE	SCI1 receive
	16	$FFDE/FFDF	Vsci1err	SCI1	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI1 error
	15	$FFE0/FFE1	Vspi1	SPI	SPIF MODF SPTEF	SPIE SPIE SPTIE	SPI
	14	$FFE2/FFE3	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
	13	$FFE4/FFE5	Vtpm2ch4	TPM2	CH4F	CH4IE	TPM2 channel 4
	12	$FFE6/FFE7	Vtpm2ch3	TPM2	CH3F	CH3IE	TPM2 channel 3
	11	$FFE8/FFE9	Vtpm2ch2	TPM2	CH2F	CH2IE	TPM2 channel 2
	10	$FFEA/FFEB	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
	9	$FFEC/FFED	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
	8	$FFEE/FFEF	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
	7	$FFF0/FFF1	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2
	6	$FFF2/FFF3	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
	5	$FFF4/FFF5	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
	4	$FFF6/FFF7	Vicg	ICG	ICGIF (LOLS/LOCS)	LOLRE/LOCRE	ICG
	3	$FFF8/FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect
	2	$FFFA/FFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	$FFFC/FFFD	Vswi	Core	SWI Instruction	—	Software interrupt
Higher	0	$FFFE/FFFF	Vreset	System control	COP LVD RESET pin Illegal opcode	COPE LVDRE — —	Watchdog timer Low-voltage detect External pin Illegal opcode

Tab. II Vektory přerušení

2.1.4 Uživatelská paměť

Paměť se dělí na několik částí (Obr. 4)



Obr. 4 Paměť MC9S08GT60

Část Dirrect Page Registers slouží pro uložení registrů nejčastěji používaných částí procesoru. Registry ostatních částí se ukládají do High Page Registers. Pro nejčastěji používaná data se užívá paměť RAM. Do paměti FLASH se ukládá především kód programu. V paměti se nachází zásobník typu LIFO. Princip spočívá ve čtení informací v pořadí opačném k zápisu. Využívá se při přerušení pro obsluhu registrů CPU, ukládání programového čítače při volání podprogramů a umožňuje také ukládání libovolných dat.

2.1.5 Zdroje hodinového cyklu

Modul pro generování hodinového kmitočtu ICG nastavuje frekvenci sběrnice Bus Clock od 32 kHz do 20 MHz. Lze použít tři zdroje hodinového signálu. Vnitřní zdroj, krystal nebo rezonátor.

2.1.6 Paralelní porty

Mikroprocesor obsahuje paralelní porty A, B, C, D, E, F a G po osmi pinech. Využít však lze pouze některé porty a piny. Jednotlivé piny jsou navíc sdíleny s dalšími částmi procesoru. Každému portu přísluší čtyři registry. Datový registr (Obr. 5) obsahuje vlastní zpracovávaná data. Směrový registr (Obr. 6) nastaví příslušné piny portu jako výstupní při logické jedničce nebo vstupní při logické nule. Pullup enable registr (Obr. 7) povoluje při nastavení pinů specifické interní pollup zařízení. Slew rate registr (Obr. 8) umožňuje nastavením pinů aktivovat sledování rychlosti.Všechny registry ve svém označení zahrnují i typ portu.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
Reset:	0	0	0	0	0	0	0	0

Obr. 5 Registr PTAD

Read: Write:	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
Reset:	0	0	0	0	0	0	0	0

Obr. 6 Registr PTADD

Read: Write:	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
Reset:	0	0	0	0	0	0	0	0

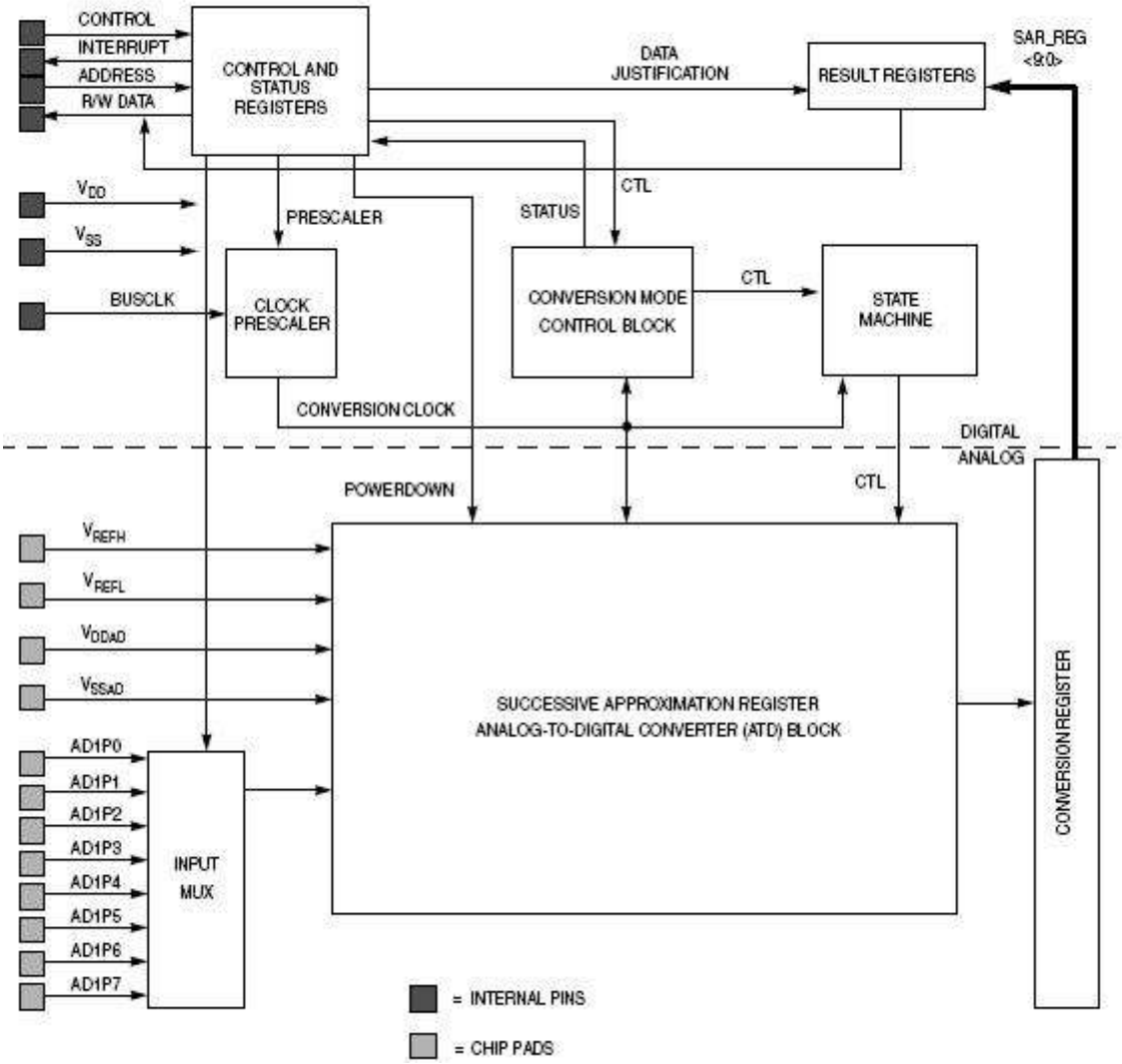
Obr. 7 Registr PTAPE

Read: Write:	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
Reset:	0	0	0	0	0	0	0	0

Obr. 8 Registr PTASE

2.1.7 Analogově-digitální převodník

Přítomen je desetibitový analogový aproximační převodník s osmibitovým multiplexem. Může pracovat v režimu jednorázového nebo kontinuálního měření. Po převodu lze generovat přerušení.



Obr. 9 Analogově-digitální převodník

Před převodem se zvolí vstupní kanál multiplexu a následná konverze je provedena aproximačním bázovým čítačem. Po převodu následuje uložení hodnoty do datového registru. Kontrolní registr ATD1C (Obr. 10) nastavuje parametry převodu.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ATDPU	DJM	RES8	SGN	PRS			
Reset:	0	0	0	0	0	0	0	0

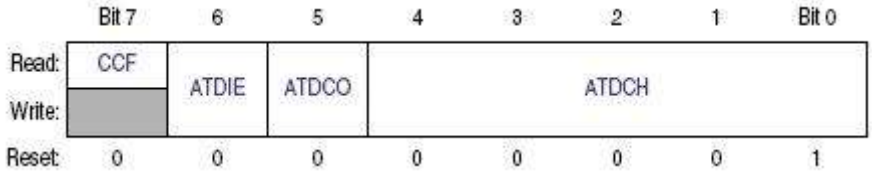
Obr. 10 Registr ATD1C

Význam jednotlivých bitů je následující

bit	význam
ATDPU	povolení A/D převodníku
DJM	zarovnání bitů v datovém registru doprava (jinak vlevo)
RES8	nastavení formátu výsledku převodu na 8 bitů (jinak 10 bitů)
SGN	převod na znaménkový typ
PRS PRS PRS PRS	prescaler (nastavení konverzních hodin v převodníku)

Tab. III Bity registru ATD1C

Způsob převodu lze nastavit v ATD1SC registru



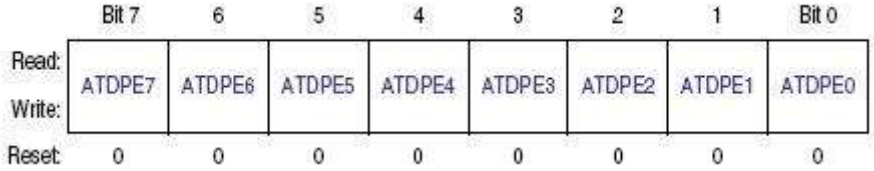
Obr. 11 Registr ATD1SC

Význam jednotlivých bitů je následující

bit	význam
CCF	příznakový bit nastaven při dokončení konverze
ATDIE	povolení interruptu
ATDCO	povolení kontinuální konverze
ATDCH 4 ATDCH 3 ATDCH 2 ATDCH 1 ATDCH 0	výběr vstupního kanálu pro analogový vstup do převodníku (pro nultý kanál 0b00000, pro první kanál 0b00001)

Tab. IV Bity registru ATD1SC

Povolení pinů pro práci s A/D převodníkem lze nastavit v ATD1PE registru

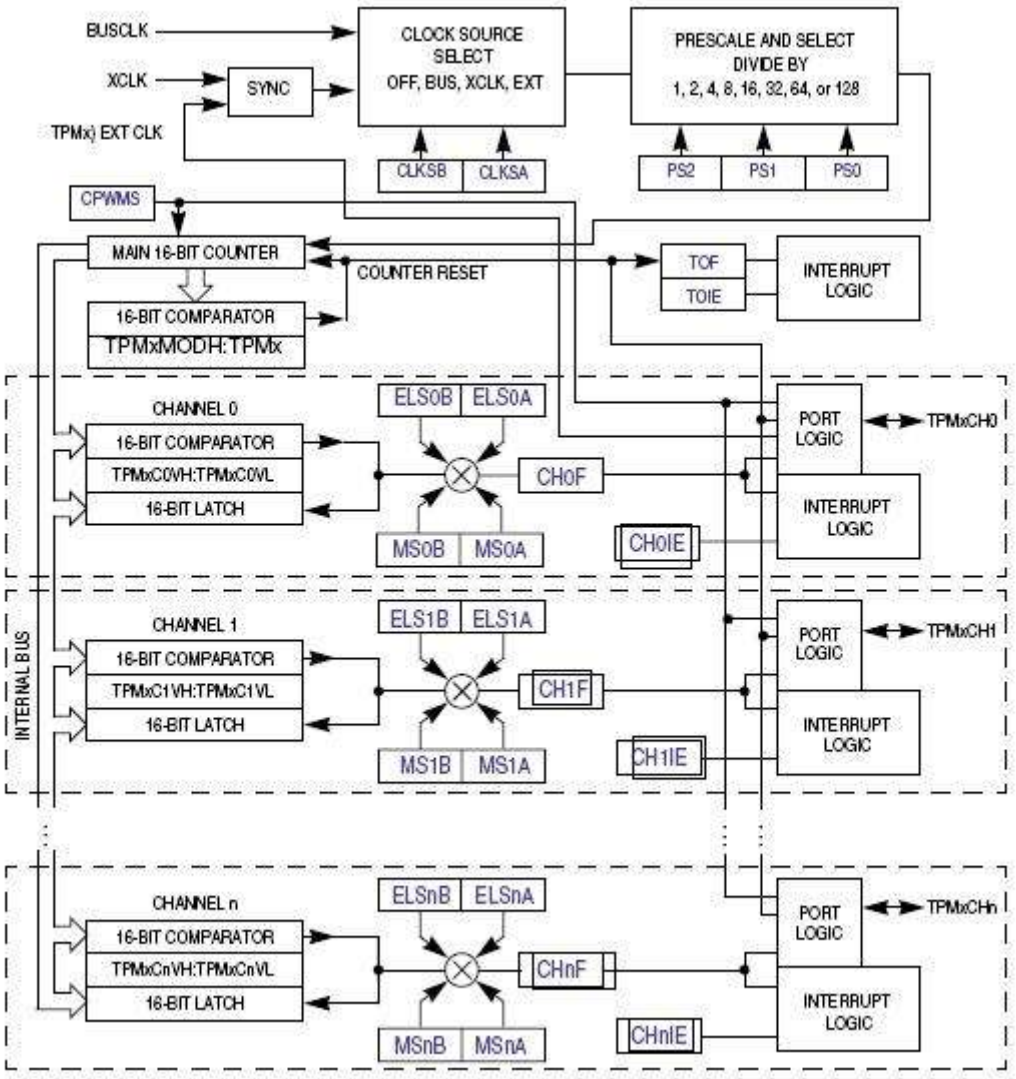


Obr. 12 Registr ATD1PE

Výsledek převodu se ukládá do ATD1R registru. Tento registr je šestnáctibitový, ale využívá maximálně 10 bitů dle registru ADT1C.

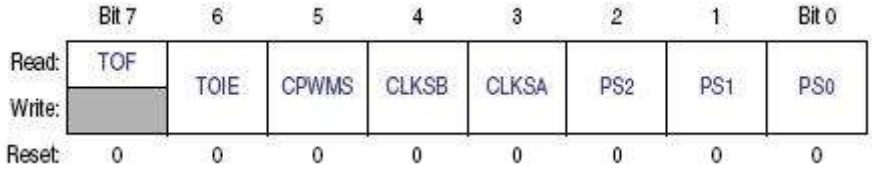
2.1.8 Časovač

Mikroprocesor obsahuje dva časovače. Veškerý popis bude ukázán na prvním časovači TPM1. Nastavení pro druhý časovač TPM2 lze získat substitucí číslice.



Obr. 13 Schéma časovače

Časovač TPM1 má dva kanály a umožňuje tak obsluhovat při jednom nastavení současně dva procesy. Časovač může pracovat ve třech módech. Mód input capture s detekcí sestupné, nástupné nebo jiné hrany vstupního signálu slouží pro zpracování vstupního signálu. Mód output capture umožňuje po uplynutí zadaného časového úseku generovat logickou jedničku nebo nulu nebo invertovat výstupní signál. Poslední mód PWM využívá pulsně šířkovou modulaci signálu. Princip spočívá v generování signálu s konstantní periodou, ale proměnnou délkou pulzu. Časování spočívá v nastavení cílové hodnoty do TPM1MOD a porovnáváním komparátorem s čítačem. Časovač umožňuje generovat přerušení. Pro úpravu hodinové signálu slouží prescaler. K nastavení časovače slouží registr TPM1SC.



Obr. 14 Registr TPM1SC

Význam jednotlivých bitů je následující

bit	**význam**
TOF	příznakový bit nastaven je-li dosažena zadaná hodnota časového registru
TOIE	povolení interruptu při TOF
CPWMS	viz Tab. VII
CLKSB	výběr zdroje hodin (pro Bus Clock 0)
CLKSA	výběr zdroje hodin (pro Bus Clock 1)
PRS PRS PRS	prescaler

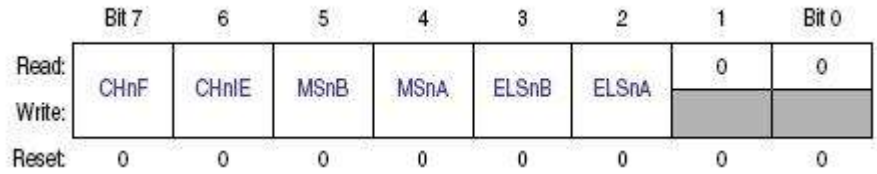
Tab. V Bity pro registr TPM1SC

Závislost délky cyklu na nastavení prescaleru při dané BUSCLK v MHz

P02	PS1	PS0	dékla cyklu (ns)
0	0	0	1000/BUSCLK
0	0	1	2000/BUSCLK
0	1	0	4000/BUSCLK
0	1	1	8000/BUSCLK
1	0	0	16000/BUSCLK
1	0	1	32000/BUSCLK
1	1	0	64000/BUSCLK
1	1	1	128000/BUSCLK

Tab. VI Nastavení prescaleru časovače

Cílová hodnota je ukládána do šestnáctibitový časového registru TPM1MOD. Hodnota vzestupného čítače pak do šestnáctibitového registru TPM1CNT. V Případě shody obou hodnot je nastaven příznakový bit TOF. Nastavení pro konkrétní kanál časovače provádí registr TPM1CnSC, kde n odpovídá danému kanálu



Obr. 15 Regist TPM1CnSC

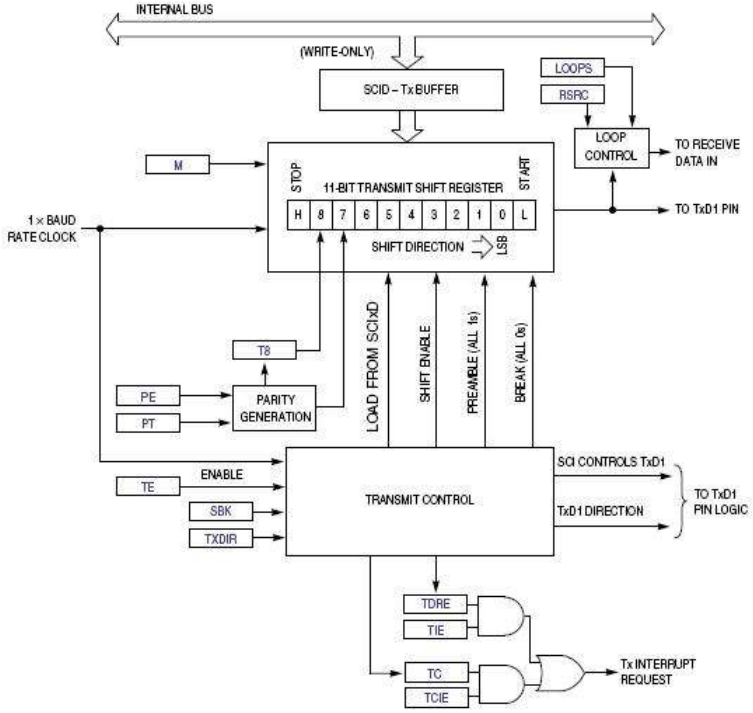
Bit CHnF se nastaví při interruptu a CHnIE povoluje interrupt pro daný kanál význam nastavení ostatních bitů plyne z Tab. VII

CPWMS	MSnB	MSnA	ELSnB	ELSnA	Mód	funkce
x	x	x	0	0		nemá pro časování význam
0	0	0	0	1	INPUT CAPTURE	Zachycení NÁSTUPNÉ hrany
	0	0	1	0		Zachycení SESTUPNÉ hrany
	0	0	1	1		Zachycení JAKÉKOLIV hrany
	0	1	0	0	OUTPUT COMPARE	pouze softwarové porovnání
	0	1	0	1		INVERZE výstupu při porovnání
	0	1	1	0		NULOVÁNÍ výstupu při porovnání
	0	1	1	1		NASTAVENÍ výstupu při porovnání
	1	x	1	0	Edge-aglined PWM	Nulování výstupu při porovnání
	1	x	x	1		Nastavení výstupu při porovnání
1	x	x	1	0	Center-Aglined PWM	Nulování výstupu při horním porovnání
	x	x	x	1		Nastavení výstupu při horním porovnání

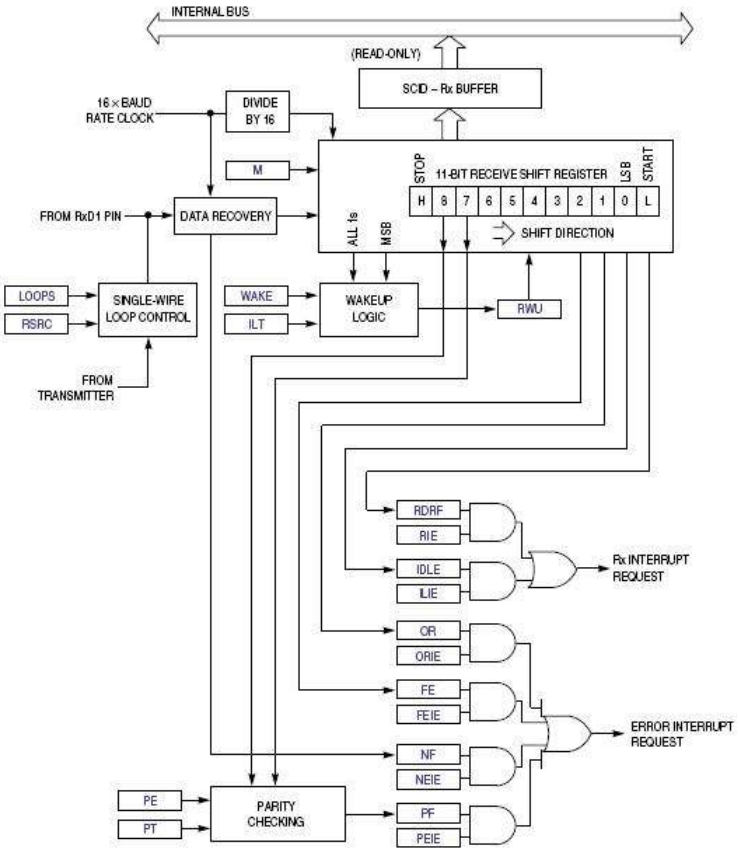
Tab. VII nastaveni režimu časovače

2.1.9 Sériová komunikace

Mikroprocesor obsahuje dva moduly pro sériovou komunikaci SCI1 a SCI2. Veškerá nastavení budou dále uvedena pro první modul. Každý modul umožňuje příjem (Obr. 17) i vysílání (Obr. 16) po sériové lince. Komunikační slovo popisuje Tab. VIII



Obr. 16 Schéma vysílání po sériové lince



Obr. 17 Schéma příjmu po sériové lince

bit	význam
1.	start bit
2.	datový bit 0
3.	datový bit 1
4.	datový bit 2
5.	datový bit 3
6.	datový bit 4
7.	datový bit 5
8.	datový bit 6
9.	datový bit 7
10.	datový bit 8 (volitelně)
11.	paritní bit
12.	stop bit

Tab. VIII struktura přenášeného slova

Přenosová rychlost se vypočte z (1), kde Baud Rate je daná přenosová rychlost v baudech, BUSCLK je hodinová frekvence a BD odpovídá hodnotě registru rychlosti SCI1BD, který je šestnáctibitový a lze v něm nastavit prvních deset bitů při zarovnání zleva.

$$Baud\ Rate = \frac{BUSCLK}{16 \cdot BD} \tag{1}$$

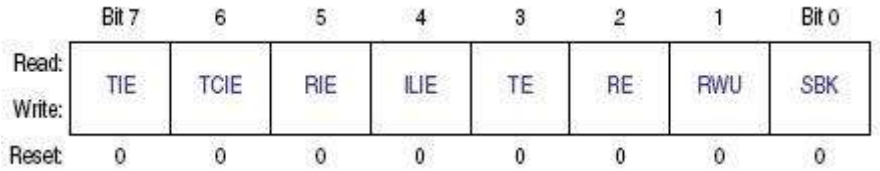
Při vysílání jsou data přenášena z datového registru do jedenáctibitového vysílacího registru (Obr. 16). Při příjmu naopak z přijímacího registru do registru datového (Obr. 17). Pro nastavení parametrů komunikace slouží následující tři registry

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset:	0	0	0	0	0	0	0	0

Obr. 18 Registr SCI1C1

bit	význam
LOOPS	zapíná testovací mód (příjem právě vyslaných dat)
SCISWAI	zmrazení SCI při WAIT módu procesoru
RSRC	Při nastavení jsou v testovacím módu užívány piny RxD1 a TxD1 (jinak řešeno interně)
M	Zapíná délku rámce 9 bitů (jinak 8 bitů)
WAKE	nastavuje wake-up podmínku
ILT	počítá jedničkové bity od start bitu (jinak od stop bitu)
PE	povolení parity
PT	nastavuje sudou paritu (jinak lichá)

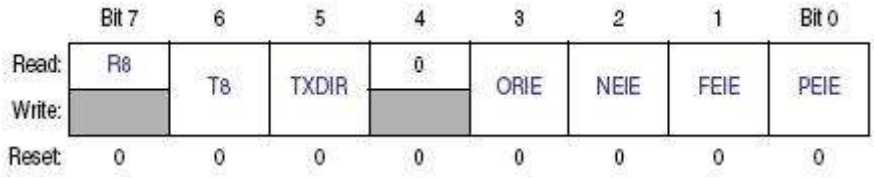
Tab. IX Bity registru SCI1C1



Obr. 19 Registr SCI1C2

bit	význam
TIE	povolení interruptu vysílání
TCIE	povolení interruptu dokončení vysílání
RIE	povolení interruptu příjmu
ILIE	povolení interruptu pro idle
TE	povolení vysílání
RE	povolení příjmu
RWU	při nastavení je interrupt příjdu podtlačent do detekce wake-up podmínky
SBK	nastavuje odesláníl logické jedničky po break znaku

Tab. X Bity registru SCI1C2

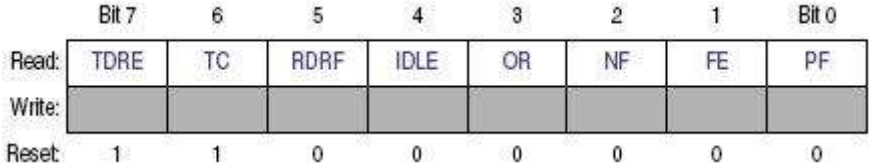


Obr. 20 Registr SCI1C3

bit	význam
R8	devátý bit příjmu
T8	devátý bit vysílání
TXDIR	Nastvení pinu TxD1 jako výstupu při Single-Wire módu (jinak jako vstup)
ORIE	Příznak overrun generuje interupt
NEIE	Příznak noise generuje interupt
FEIE	Příznak farming error generuje interupt
PEIE	Příznak chyba parity generuje interrupt

Tab. XI Bity registru SCI1C3

Pro zjištění informací o přenosu slouží dva stavové registry



Obr. 21 Registr SCI1S1

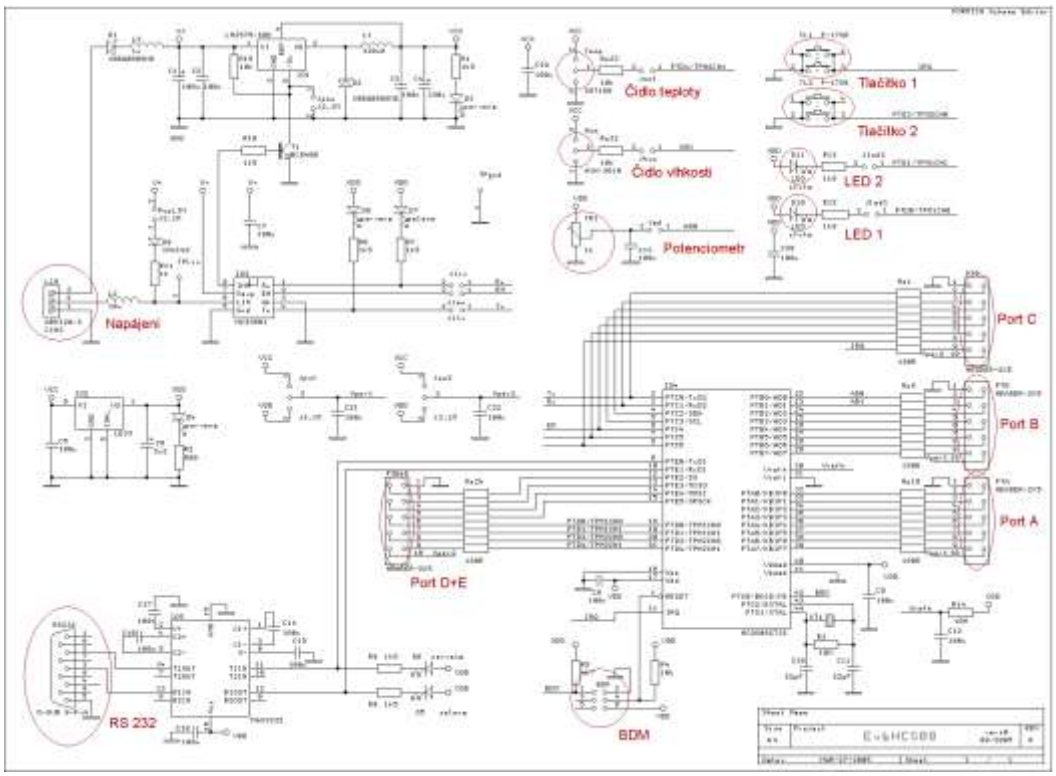
bit	význam
TDRE	nastaven při vyprázdnění datového registru (bufferu)
TC	nastaven při ukončení přenosu
RDRF	nastaven při plném příjimacím registru
IDLE	příznak detekce idle
OR	příznak při detekci overrun (přetečení)
NF	příznak detekování noise
FE	příznak framing error (detekována nula na místě stop bitu)
PF	příznak chyby parity

Tab. XII Bity registru SCI1S1

Z registru SCI1S2 lze získat pouze bit RAF na nulté pozici, který nastavuje bit RxD1 pro příjem. Jinak je bit RxD1 nastaven pro idle.

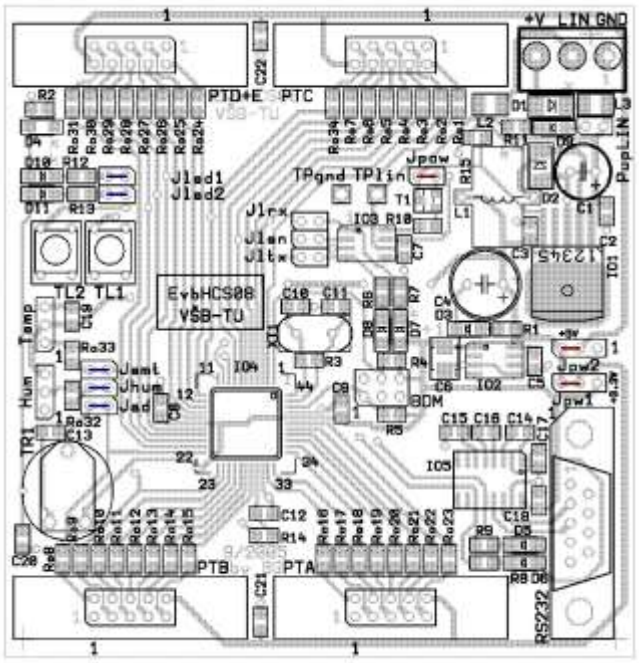
2.2 DESKA EVBHCS08

2.2.1 Přehledové schéma



Obr. 22 Schéma desky EvbHCS08

2.2.2 Zapojení jumperů



Obr. 23 Zapojení jumperů

2.2.3 Komunikace s PC – BDM rozhraní

Pro nahrávání a práci s programem slouží konektor, který se ve schématech označuje BDM (Background Debug Mode). Tímto lze přistupovat k programu, provádět úpravy a nahrávat nový program. Připojení k tomuto konektoru vyžaduje speciální převodník. Použít lze například zařízení USB BDM MULTILINK. Na Obr. 1 vidíme reálný pohled na toto zařízení po připojení k základní desce. Při zapojování desky nejprve propojíme převodník s deskou a poté zapojíme tento převodník do počítače. Nakonec přivedeme na desku napájení. Odpojování se provádí opačným postupem.

2.2.4 Napájení

K napájení slouží adaptér se specifickým zakončením pro připojení k desce. Na tomto adaptéru je třeba nastavit požadovanou hodnotu napětí a to 12V. Před připojením desky do elektrické sítě musí být všechny jumpery správně zapojeny (Obr. 23).

2.2.5 Tlačítka

Pro přímé diskrétní zadání slouží na desce dvě tlačítka. První tlačítko plní funkci externího interruptu a označuje se ve schématech TR1. Druhé se ve schématech označuje TR2 a je vyvedeno na nultý kanál druhého časovače (Obr. 22).

2.2.6 Signalizační diody

Pro přímou indikaci lze využít dvě diody. První z nich se ve schématech se označuje D10. Její použití vyžaduje zapojení jumperu Jled1 (Obr. 23). Signál je vyveden jako nultý kanál prvního časovače (Obr. 22). Druhá se ve schématech se označuje D11. Její použití vyžaduje

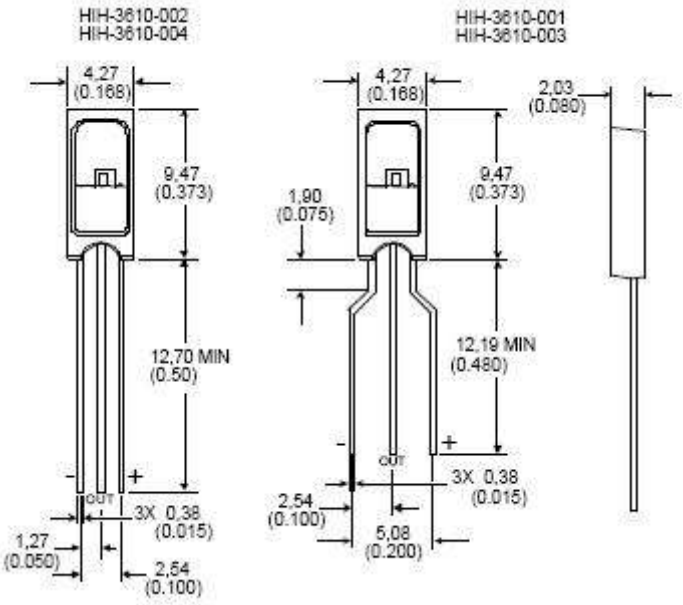
zapojení jumperu Jled2 (Obr. 23). Signál je vyveden jako první kanál prvního časovače (Obr. 22).

2.2.7 A/D převodník - Potenciometr

Za pomoci potenciometru lze přímo vkládat analogovou hodnotu. Potenciometr je ve schématech značen jako TR1. Jeho výstup je vyveden do nultého A/D převodníku (Obr. 22). Rozsah potenciometru se převede na diskrétní rozsah 0 až 255.

2.2.8 Čidlo vlhkosti

Pro měření vlhkosti je použito čidlo HIH 3610, které může být realizováno v několika provedeních.



Obr. 24 provedení čidla vlhkosti

Ve schématech se označuje Hum. Jeho použití vyžaduje zapojení jumperu Jhum (Obr. 23). Signál z čidla je vyveden do prvního A/D převodníku (Obr. 22). Podrobné technické parametry lze nalézt v dokumentaci [3]. Pro získání hodnoty relativní vlhkosti *RH* se nejprve vypočte vlhkost vztažená na 25°C RH_{25}.

$$RH_{25} = \frac{\frac{U}{U_{vst}} - 0{,}16}{0{,}0062} \tag{2}$$

Kde U_{vst} je vstupní napětí 5V a U napětí vstupní dle

$$U = \frac{3{,}9 - 0{,}8}{256} \cdot AD + 0{,}8 \tag{3}$$

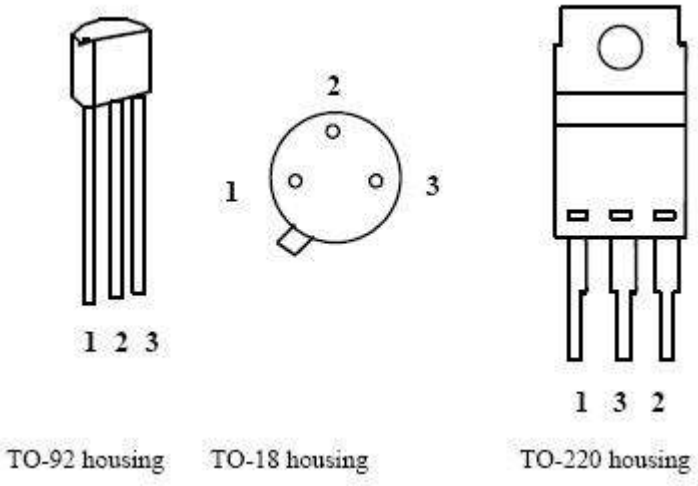
Kde AD je hodnota z A/D převodníku. Pro přepočet na skutečnou teplotu platní

$$RH = \frac{RH_{25}}{1{,}0546 - 0{,}00216 \cdot T} \tag{4}$$

Kde *RH* je relativní vlhkosti při teplotě *T*. Teplota se zadává ve stupních celsia.

2.2.9 Čidlo teploty

Jako teplotní čidlo je použito SMT 160, které může být realizováno v několika provedeních.



Obr. 25 provedení čidla teploty

Ve schématech se označuje Temp. Jeho použití vyžaduje zapojení jumperu Jsmt (Obr. 23). Signál z čidla je vyveden jako první kanál druhého časovače (Obr. 22). Teplotu lze snímat v rozsahu -45°C až 130°C s přesností 0,7°C. Podrobné technické parametry lze nalézt v dokumentaci [2]. Výpočet teploty *T* ve stupních celsia se provede na základě relativní velikosti pulzu *DC*.

$$T = \frac{DC - 0{,}320}{0{,}00470} \tag{5}$$

2.2.10 Sériová linka

Deska umožňuje komunikaci po sériové lince RS232. Konektor je ve schématech značen RS232. Tímto způsobem lze vyčítat data z mikroprocesoru do počítače nebo pracovní stanice. Z počítače lze také do mikroprocesoru zasílat instrukce a tím ovládat probíhající program. Pro funkční komunikaci je třeba na mikroprocesoru i počítači nastavit shodnou rychlost přenosu, délku dat, stop bit, paritu a handshaking.

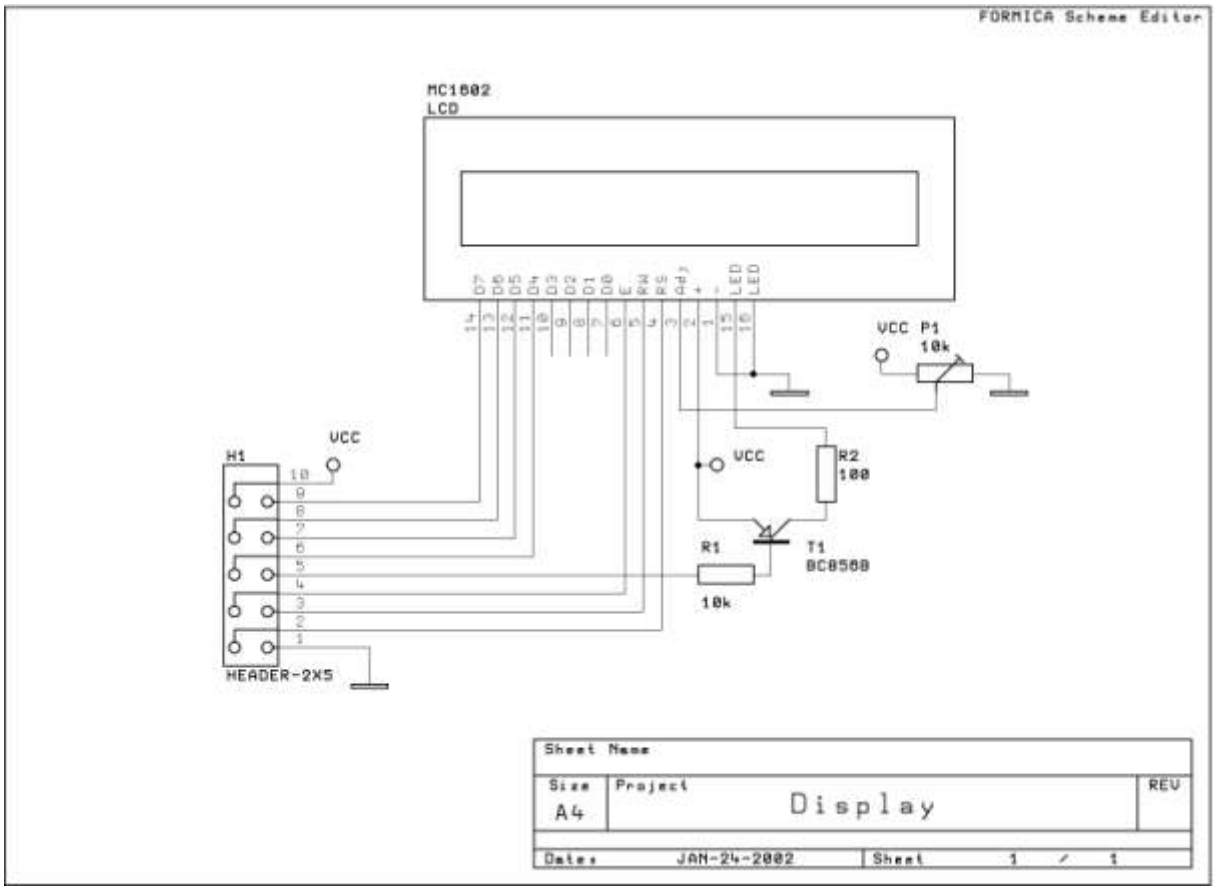
2.2.11 Rozšiřující porty

Deska nabízí celkem čtyři rozšiřující porty. Jsou to porty A, B, C a D+E. Přes tyto porty lze k základní desce připojit desky rozšiřující. Většina pinů na portech je sdílená.

2.3 ROZŠIŘUJÍCÍ DESKY

2.3.1 Displej

Rozšiřující deska s displejem obsahuje dvouřádkový displej o šestnácti znacích na řádek.



Obr. 26 Schéma displeje

Na Obr. 26 vidíme schéma a na Obr. 1 pak reálný pohled na displej po připojení k základní desce. Význam jednotlivých bitů je následující

bit	význam
RS	indikace zadávání znaku (jinak zadávaní příkazu)
RW	povolení čtení
E	povolení přístupu k bitům D4 až D7
D4	příjem nultého a čtvrtého bitu znaku
D5	příjem prvního a pátého bitu znaku
D6	příjem druhého a šestého bitu znaku
D7	příjem třetího a sedmého bitu znaku

Tab. XIII Bity displeje

Znaky lze odesílat ve formátu unsigned char. Zobrazí se pak znak, jehož ascii kód odpovídá přijatému. Bezproblémově jsou zobrazovány znaky od kódu 32 do kódu 127. Kromě znaků lze odesílat na displej i řídící příkazy (viz Tab. XIV)

příkaz	kód
zapnutí displje	0x01
vypnutí displeje	0x08
vymazání displeje	0x01
kurzor na začátek prvního řádku	0x80
kurzor na začátek druhého řádku	0xC0
kurzor neviditelný	0x0C
kurzor podtržítko bliká	0x0D
kurzor plný	0x0E
kurzor plný bliká	0x0F
Inicializace displeje	0x03 0x03 0x03 0x02 0x28 0x08 0x01 0x06 0x28 0x0C 0x06 0x0C

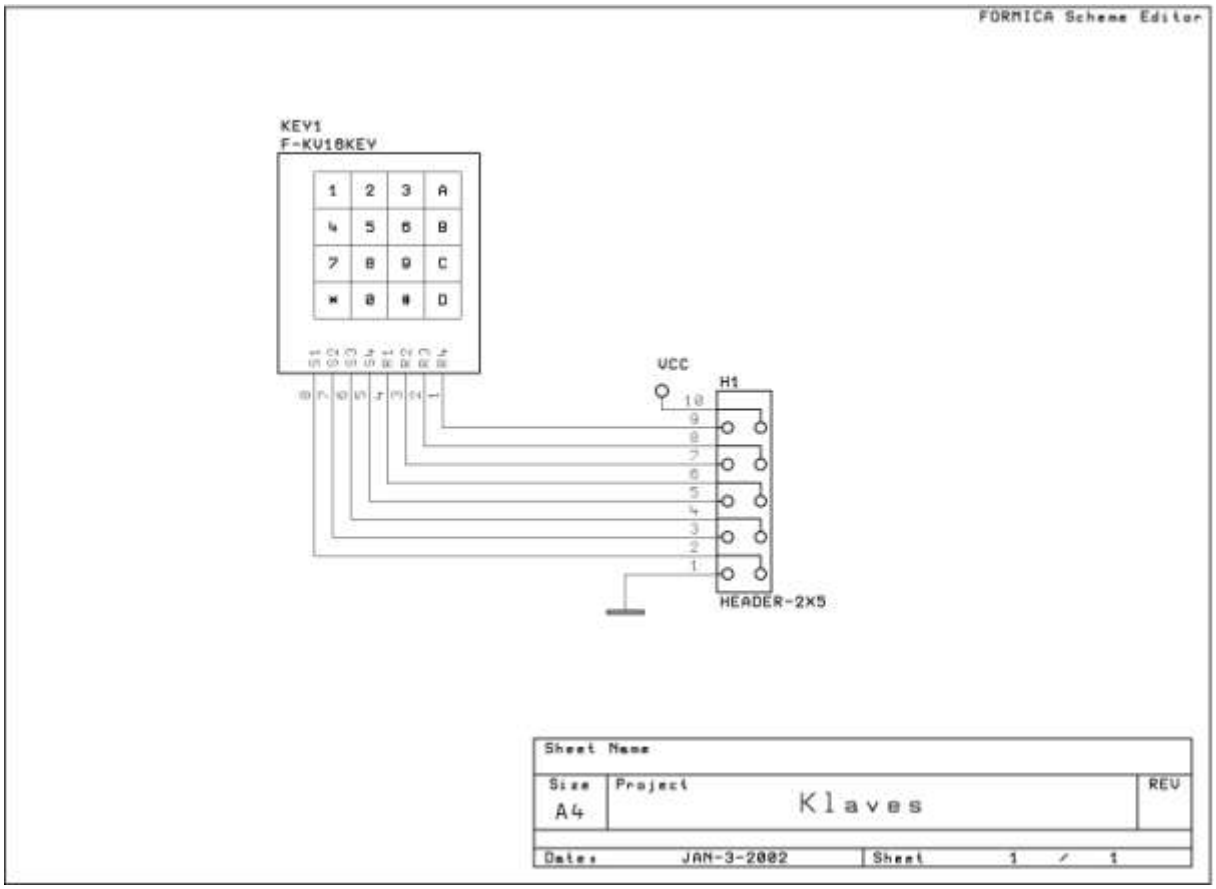
Tab. XIV Příkazy pro displej

Prvnímu použití displeje musí předcházet inicializace. Ještě před odesláním série inicializačních příkazů je potřeba nastavit všechny bity mimo RS na jedna a následně naopak všechny mimo RS vynulovat. Bit RS má tedy při těchto zadáních opačnou hodnotu oproti ostatním bitům. Před každým zápisem nebo čtením se pak nejprve nastaví bity definující druh

přístupu. Tedy RS pro určení jedná-li se o znak nebo příkaz a RW pro určení jedná-li se o čtení nebo zápis. Následně se příslušný znak nebo příkaz zapíše ve dvou krocích. Nejprve jsou zapsány horní čtyři bity do bitů D7 až D4 a následně dolní čtyři byty tamtéž. Před každým zápisem do skupiny bitů D7 až D4 je navíc potřeba povolit přístup bitem E a po zapsání přístup opět zakázat.

2.3.2 Maticová klávesnice

Rozšiřující deska s maticovou klávesnicí obsahuje 16 kláves v matici o čtyřech řádkách a čtyřech sloupích.



Obr. 27 Schéma maticové klávesnice

Na Obr. 27 vidíme schéma a na Obr. 1 pak reálný pohled na maticovou klávesnici po připojení k základní desce. Význam jednotlivých bitů je následující

bit	význam
S1	stisknuto tlačítko v prvním sloupci
S2	stisknuto tlačítko ve druhém sloupci
S3	stisknuto tlačítko ve třetím sloupci
S4	stisknuto tlačítko ve čtvrtém sloupci
R1	stisknuto tlačítko v prvním řádku
R2	stisknuto tlačítko ve druhém řádku
R3	stisknuto tlačítko ve třetím řádku
R4	stisknuto tlačítko ve čtvrtém řádku

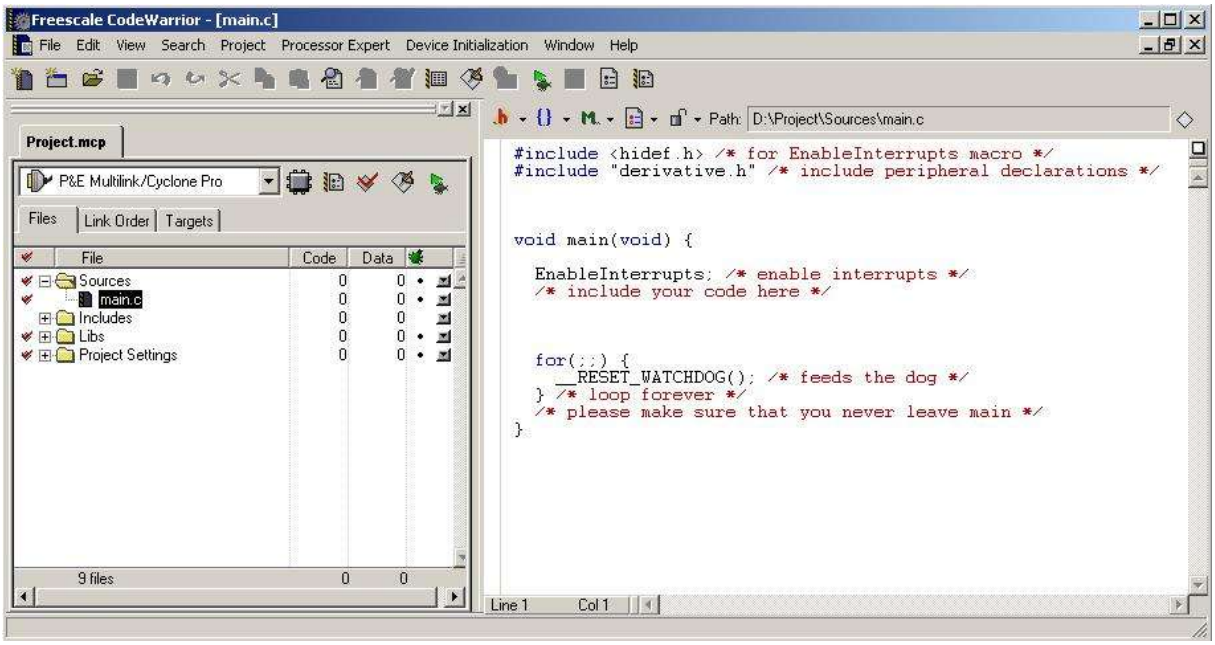
Tab. XV Bity maticové klávesnice

Každému tlačítku nebo skupině zároveň stisknutých tlačítek odpovídá určitá kombinace bitů dle Tab. XV a lze tak přesně určit stisknutá tlačítka. Velkou výhodou této maticové klávesnice je právě možnost odečtení stisku i více jak jednoho tlačítka současně.

2.4 PROGRAMOVÉ VYBAVENÍ

2.4.1 CodeWarrior

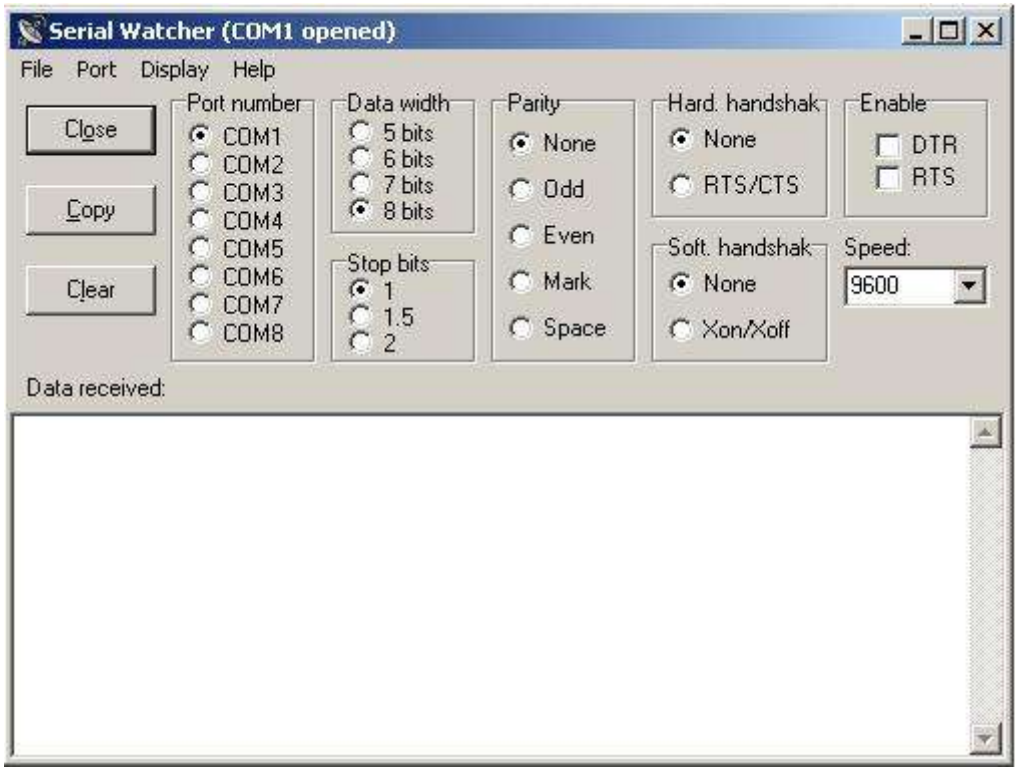
Program CodeWarrior je vývojové prostředí pro aplikace v programovacím jazyce C, C++ nebo Assembleru. Umožňuje vytváření a testování vytvořených programů. Umožňuje také nahrávání a správu aplikací přes v mikroprocesoru přes komunikační převodník. V nastavení programu lze přepínat mezi simulací a reálným napojením k mikroprocesoru.



Obr. 28 Pracovní prostředí programu CodeWarrior

2.4.2 SerialWatcher

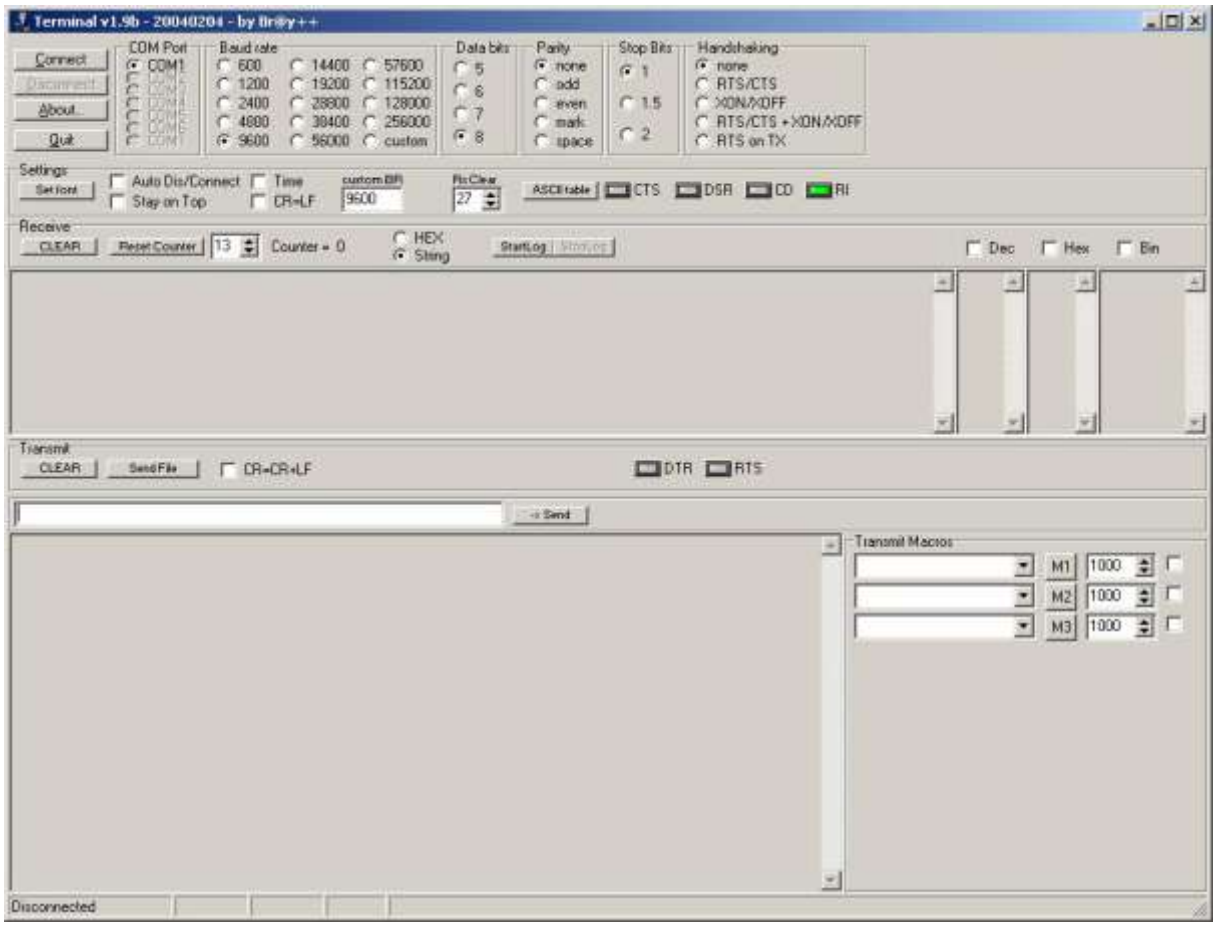
Tento program slouží pro zobrazení dat, která byla do počítače zaslána po sériové lince. Lze v něm nastavit parametry komunikace. Konkrétně použitý port, datovou šířku, stop bit, paritu, handshaking, DTR, RTS a rychlost přenosu. V menu Port lze uzavřít (Close) nebo otevřít (Open) zvolený port. Program umožňuje zobrazení přijatých dat v decimálním, hexadecimálním a ASCII tvaru. Typ zobrazení lze nastavit v menu Display.



Obr. 29 Program SerialWatcher

2.4.3 Terminal

Jedná se o pokročilejší program pro sériovou komunikaci. Lze v něm nastavit parametry komunikace. Konkrétně použitý port, datovou šířku, stop bit, paritu, handshaking, a rychlost přenosu a některé další. Dále lze nastavovat formát zobrazovaných dat. Program umožňuje také odesílání dat z počítače.



Obr. 30 Program Terminal

3 PRACOVNÍ ČÁST

3.1 OBECNÉ POKYNY K VYPRACOVÁNÍ

Vypracovávejte zadané části návodu ve zde uvedeném pořadí. Nejsou-li nastavení pro některé registry nebo piny uvedeny, předpokládá se výchozí nastavení. Definujete-li funkci, nezapomeňte ji v kódu také deklarovat nejpozději před prvním použitím. Po dokončení všech zadaných částí demonstrujte funkčnost vašeho programu vyučujícímu a vypracujte protokol. Protokol bude obsahovat zadání vámi vypracovaných částí, nastavení použitých registrů, použité výpočty, výpis kódu programu a závěrečné zhodnocení.

3.2 VYTVOŘENÍ PRÁZDNÉHO PROJEKTU

3.2.1 Zadání

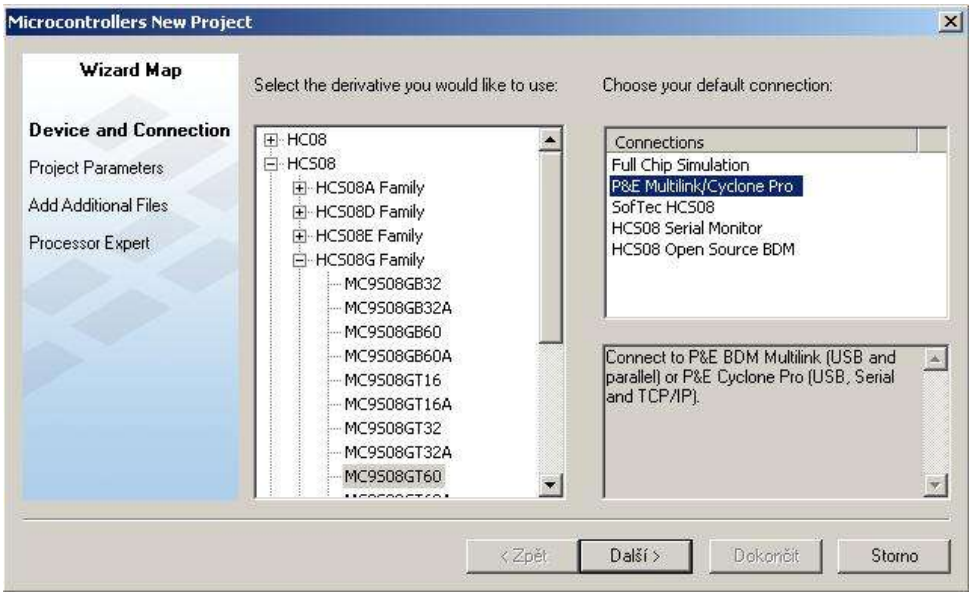
Vytvořte, nahrajte do mikroprocesoru a spusťte program. Program nastavte tak, aby bylo možno pracovat s desetinnými čísli.

3.2.2 Pracovní postup

- Spusťte program CodeWarrior IDE
- Klikněte na tlačítko Create New Project

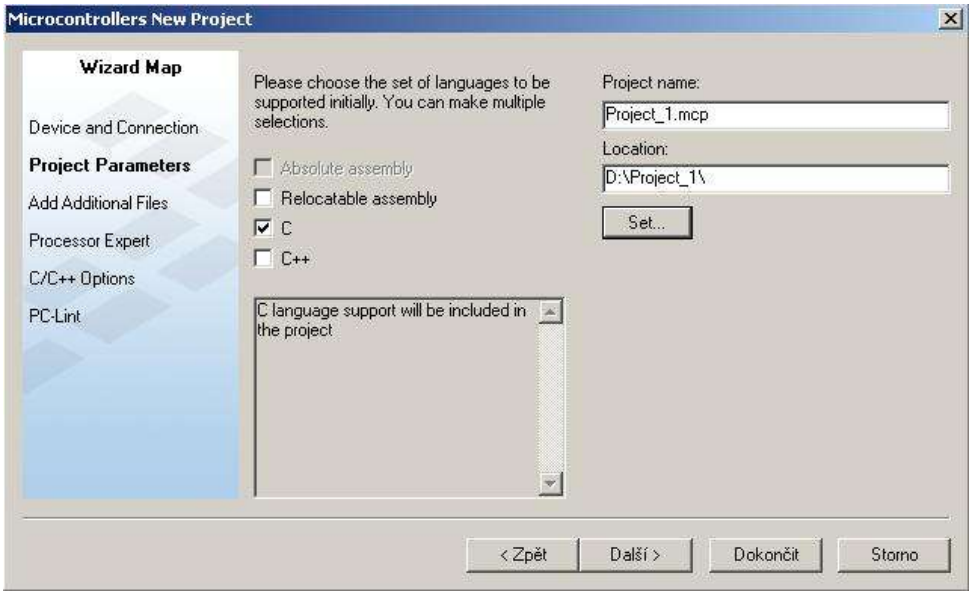
Obr. 31 Vytvoření nového projektu

- Vyberte typ mikroprocesoru MCS08GT60 a typ připojení P&E Multilink/Cyclone Pro a klikněte na tlačítko Další



Obr. 32 Volba typu mikroprocesoru a připojení

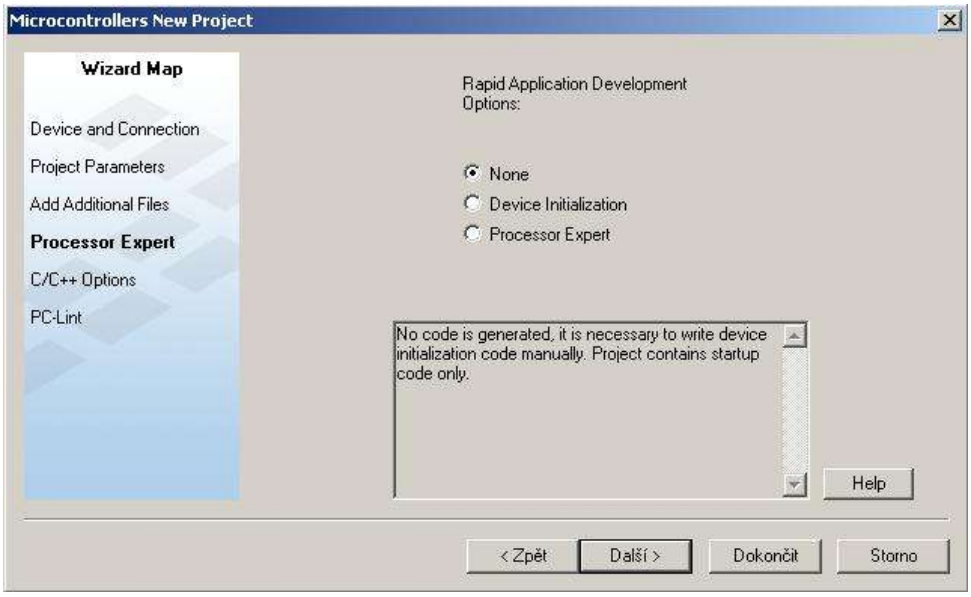
- Zaškrtněte typ jazyka C, zadejte jméno projektu (Project name) a kliknutím na Set… vyberte adresář pro uložení projektu a klikněte na tlačítko Další



Obr. 33 Volba adresáře projektu

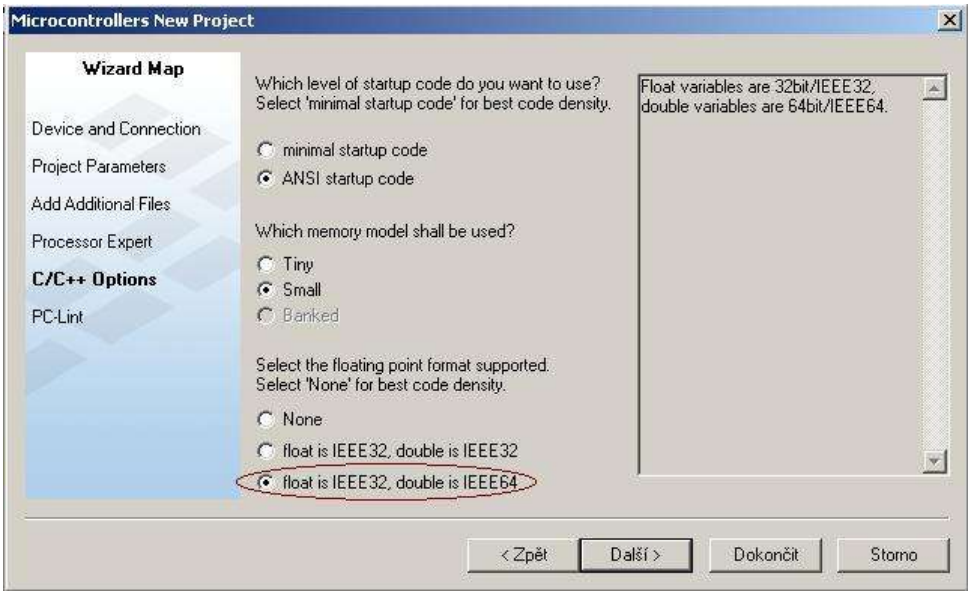
- V následujících dvou oknech neprovádějte žádné úpravy a vždy pouze klikněte na tlačítko Další

Obr. 34 Přidání existujících souborů do projektu



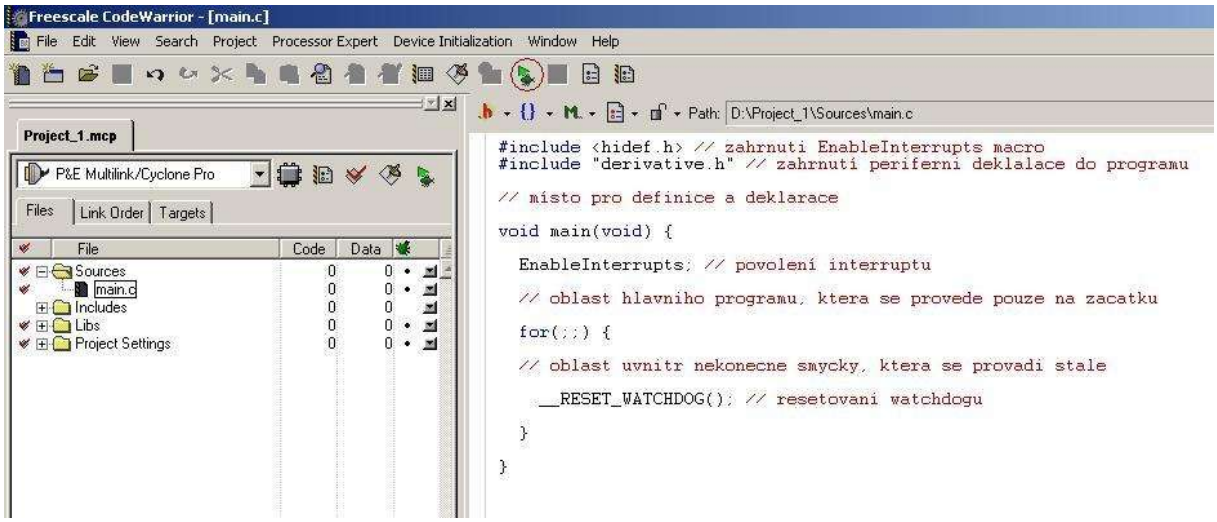
Obr. 35 Expertní nastavení

- Nakonec vyberte volbu pro práci s desetinnými čísly (v obrázku zdůrazněno) a klikněte na Dokončit (nikoliv na Další)



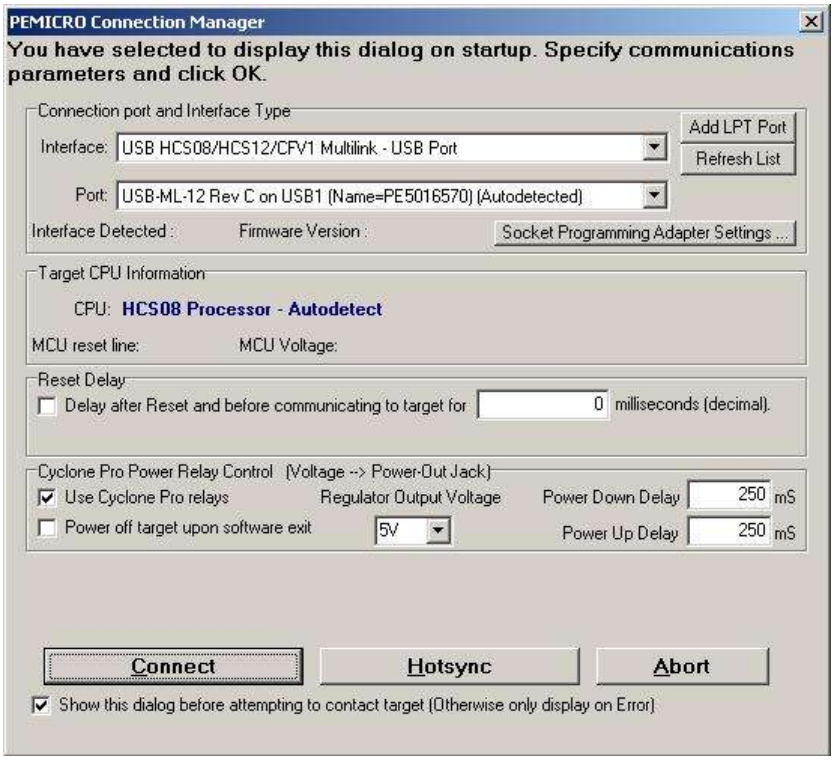
Obr. 36 Nastavení pro C/C++

- Zobrazí se okno pro psaní programu, které je zde uvedeno s komentáři popisující základní kód.



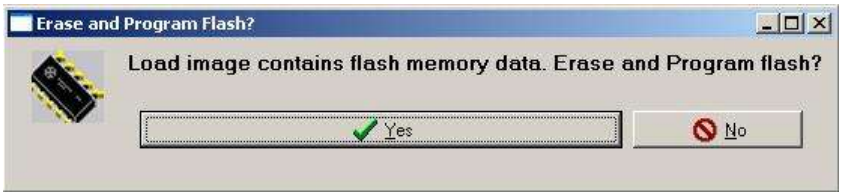
Obr. 37 Základní okno pro psaní programu

- Nyní lze program libovolně upravovat
- Pro zkompilování a nahrání programu do paměti mikroprocesoru klikněte na zelenou šipku s broukem (F5)
- Následuje dotaz na typ připojení. Stiskněte Connect



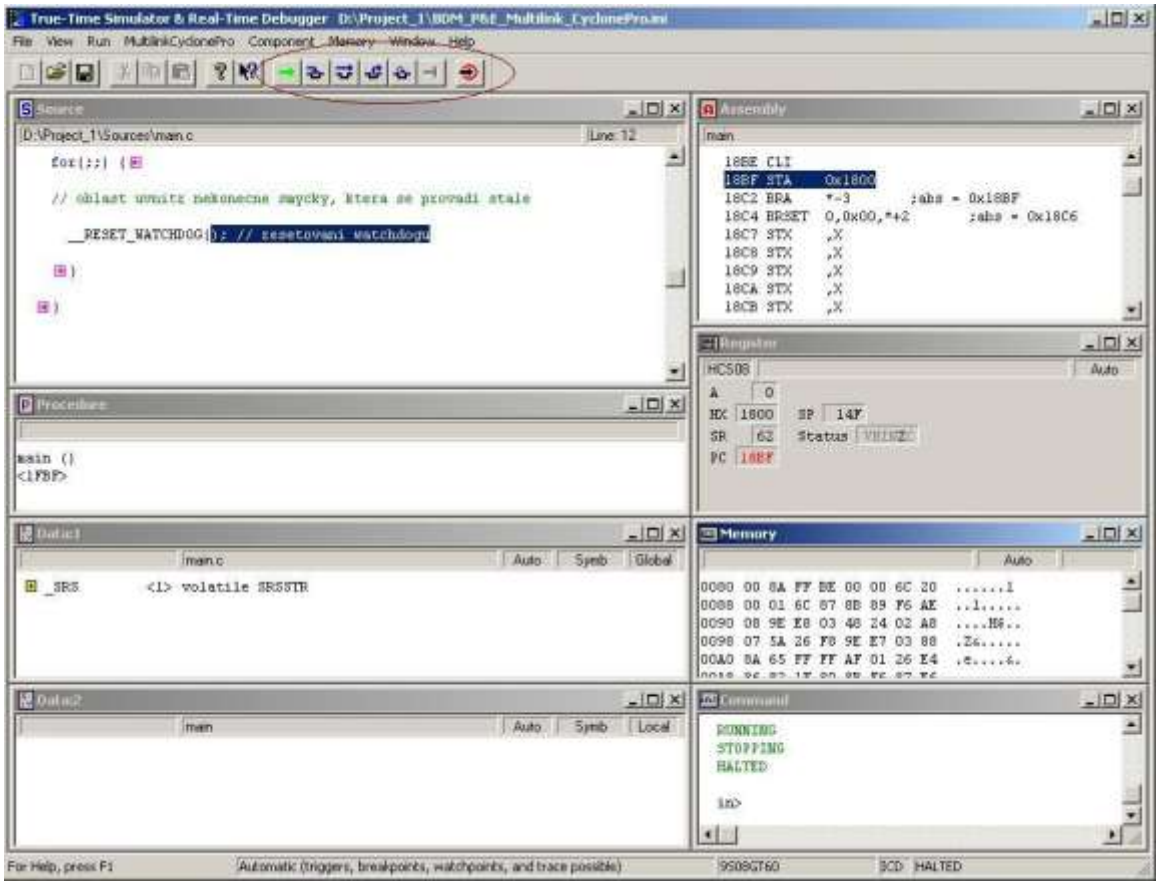
Obr. 38 Typ připojení

- Potvrďte vymazání paměti



Obr. 39 Vymazaní paměti

- Program spustíte kliknutím na zelenou šipku (F5)
- Pro zastavení programu stiskněte tlačítko s červeným ležatým T (F6)
- Pro reset procesoru stiskněte černou šipku v červeném kruhu (CTRL+R)
- Program lze krokovat pomocí tlačítek s modrými šipkami
- Na Obr. 40 vidíme také řadu oken. Kromě okna s výpisem kódu programu také výpis v Assembleru, vnitřní registry procesoru, paměť, globální a lokální proměnné, okno příkazů a okno procedur



Obr. 40 Sledování průběhu programu

3.3 SÉRIOVÁ KOMUNIKACE RS232

3.3.1 Zadání

- Zašlete libovolné číslo z mikroprocesoru do počítače
- Zasílejte libovolné číslo z mikroprocesoru do počítače v cyklu
- Zasílejte libovolné číslo z mikroprocesoru do počítače v cyklu s časováním

3.3.2 Postup práce

- V hlavním programu proveďte nastavení sériové komunikace bez interruptů
  - Registr **SCI1C2** nastavte pro vysílání
    ```
    SCI1C2=0b00001000;
    ```
  - Při zadané přenosové rychlosti (*např. 9600 baudů*) a dané BUSCLK (*např.4 MHz*) nastavte registr **SCIBD** dle (1)
    ```
    SCI1BD=26;
    ```
- Nastavte odeslání zadaného čísla ve formátu unsigned char jeho zápisem do registru **SCI1D** (*např. 123)*
  ```
  SCI1D=123;
  ```
- Testujte dokončení přenosu prázdnou `while` smyčkou s podmínkou nenastavení příznaku vyprázdnění bufferu registru **SCI1S1**
  ```
  while(SCI1S1_TDRE==0){
  }
  ```
- Na počítači spusťte program SerialWatcher a nastavte dle Obr. 29
- Upravte rychlost přenosu na vámi použitou hodnotu
- Ověřte funkčnost vašeho programu po nahrání do mikroprocesoru
- Upravte program tak, aby odesílal data v nekonečné smyčce
- Znovu ověřte funkčnost vašeho programu po nahrání do mikroprocesoru
- V hlavním programu proveďte nastavení prvního časovače
  - Registr **TPM1SC** a časovací registr **TPM1MOD** nastavte pro zadanou délku cyklu (*např. 1000 ms*) při dané BUSCLK (*např.4 MHz*). Nastavte také BUSCLK jako zdroj hodinového signálu
    ```
    TPM1SC = 0b00001111;
    TPM1MOD = 31250;
    ```
  - Pro nultý kanál nastavte mód OUTPUT COMPARE s funkcí pouze softwarového porovnání v registru **TPM1C0SC** a povolte interrupt
    ```
    TPM1C0SC = 0b01010000;
    ```
- Vytvořte `interrupt 5 preruseni_casovac1` pro zpracování přerušení od nultého kanálu a vynulujte v něm příznak tohoto přerušení **TPM1C0SC_CH0F**
  ```
  void interrupt 5 preruseni_casovac1(void){
  TPM1C0SC_CH0F=0;
  }
  ```

- Do `interrupt 5` přesuňte obsah nekonečné smyčky mimo `__RESET_WATCHDOG();`
- Opět ověřte funkčnost vašeho programu po nahrání do mikroprocesoru

3.4 ZOBRAZENÍ NA DISPLEJ

3.4.1 Zadání

- Zobrazte na displej libovolný znak
- Zobrazte na displej libovolný znak v cyklu
- Zobrazte na displej libovolný znak v cyklu s časováním
- Zobrazte na displej libovolné číslo v cyklu s časováním

3.4.2 Postup práce

- Proveďte následující definice

```
#define DISPLEJ              PTAD
#define DISPLEJ_SMER         PTADD
#define DISPLEJ_RS           PTAD_PTAD0
#define DISPLEJ_RnW          PTAD_PTAD1
#define DISPLEJ_E            PTAD_PTAD2
#define DISPLEJ_BIT4         PTAD_PTAD4
#define DISPLEJ_BIT5         PTAD_PTAD5
#define DISPLEJ_BIT6         PTAD_PTAD6
#define DISPLEJ_BIT7         PTAD_PTAD7
```

- Vytvořte funkci `cekani`, která bude obsahovat dvěstěkrát se opakující `for` cyklus obsahující pouze reset watchdogu

```
void cekani (void) {
unsigned int k;
for (k=1;k<200;k++){
__RESET_WATCHDOG();
}
}
```

- Vytvořte funkci `zapis_znak`, která bude odesílat ascii kód znaku na displej
  - Vytvořte prázdnou funkci `posli_znak` se vstupní proměnou znak formátu unsigned char
    ```
    void posli_znak(unsigned char znak) {
    }
    ```
  - Ve funkci zakažte čtení nulováním bitu **DISPLEJ_RnW**
    ```
    DISPLEJ_RnW=0;
    ```
  - Nastavte mód odesílání znaku nulováním bitu **DISPLEJ_RS**
    ```
    DISPLEJ_RS=1;
    ```
  - Povolte přístup k bitům 7 až 4 nastavením bitu **DISPLEJ_E**
    ```
    DISPLEJ_E=1;
    ```

- Odešlete horní čtyři bity na **DISPLEJ_BIT7** až **DISPLEJ_BIT4**
```
DISPLEJ_BIT7=(znak&0x80)>>7;
DISPLEJ_BIT6=(znak&0x40)>>6;
DISPLEJ_BIT5=(znak&0x20)>>5;
DISPLEJ_BIT4=(znak&0x10)>>4;
```
- Zakažte přístup k bitům 7 až 4 nulováním bitu **DISPLEJ_E**
```
DISPLEJ_E=0;
```
- Povolte přístup k bitům 7 až 4 nastavením bitu **DISPLEJ_E**
```
DISPLEJ_E=1;
```
- Odešlete dolní čtyři bity na **DISPLEJ_BIT7** až **DISPLEJ_BIT4**
```
DISPLEJ_BIT7=(znak&0x08)>>3;
DISPLEJ_BIT6=(znak&0x04)>>2;
DISPLEJ_BIT5=(znak&0x02)>>1;
DISPLEJ_BIT4=(znak&0x01);
```
- Zakažte přístup k bitům 7 až 4 nulováním bitu **DISPLEJ_E**
```
DISPLEJ_E=0;
```
- Na konec funkce umístěte volání funkce `cekani`

➢ Vytvořte funkci `posli_znak`, která bude odesílat příkazy na displej

- Tuto funkci vytvořte jako funkci `zapis_znak`. Pouze nastavte mód zadávaní příkazu nulováním bitu **DISPLEJ_RS**
```
DISPLEJ_RS=0;
```

➢ V hlavním programu proveďte inicializaci displeje

- V registru **DISPLEJ_SMER** nastavte všechny bity na jedna
```
DISPLEJ_SMER=0xFF;
```
- Do registru zapište **DISPLEJ** ~1 a následně ~0
```
DISPLEJ=~1;
DISPLEJ=~0;
```
- Zašlete sérii inicializační příkazů dle Tab. XIII
```
posli_znak(0x03);
posli_znak(0x03);
posli_znak(0x03);
posli_znak(0x02);
posli_znak(0x28);
posli_znak(0x08);
posli_znak(0x01);
posli_znak(0x06);
posli_znak(0x28);
posli_znak(0x0C);
posli_znak(0x06);
posli_znak(0x0C);
```

➢ Zapište znak na displej

- V hlavním programu volejte funkci `posli_znak` s příkazem vymazání displeje
  ```
  posli_znak(0x01);
  ```
- Následně volejte funkci `posli_znak` s příkazem pro nastavení kurzoru na první řádek
  ```
  posli_znak(0x80);
  ```
- Nakonec volejte funkci `zapis_znak` s ascii kódem libovolného znaku v rozsahu od 32 do 127. Lze zadat také přímo znak (např. ‘A’) v daném rozsahu ascii tabulky.
  ```
  zapis_znak('A');
  ```
- Ověřte funkčnost vašeho programu po nahrání do mikroprocesoru
- Postupným voláním funkce `zapis_znak` zapište zadané slovo o maximálně patnácti písmenech (*např. Ahoj*)
  ```
  zapis_znak('A');
  zapis_znak('h');
  zapis_znak('o');
  zapis_znak('j');
  ```
- Ověřte znovu funkčnost vašeho programu po nahrání do mikroprocesoru

- Upravte program tak, aby zapisování znaků na displej probíhalo v nekonečné smyčce
- Opět ověřte funkčnost vašeho programu po nahrání do mikroprocesoru
- Upravte program tak, aby zapisování znaků na displej probíhalo při zpracování interruptu časovače (viz Postup práce pro sériovou komunikaci RS232)
- Znovu ověřte funkčnost vašeho programu po nahrání do mikroprocesoru
- Vytvořte funkci `zobraz_char`, která převede zadané číslo formátu unsigned char na tři znaky reprezentující číslice tohoto čísla a ty pak odešle na displej
  ```
  void zobraz_char(unsigned char cislo){
  unsigned char stovky,desitky,jednotky;
  stovky=cislo/100;
  desitky=(cislo-stovky*100)/10;
  jednotky=cislo-stovky*100-desitky*10;
  zapis_znak(stovky+48);
  zapis_znak(desitky+48);
  zapis_znak(jednotky+48);
  }
  ```
- Ve `interrupt 5` nahraďte volání funkce `zapis_znak` za volání funkce `zobraz_char` se zadaným číslem (*např. 76*)
  ```
  zobraz_char(76);
  ```
- Znovu ověřte funkčnost vašeho programu po nahrání do mikroprocesoru

3.5 ZPRACOVÁNÍ ANALOGOVÉ HODNOTY

3.5.1 Zadání

- Odešlete analogovou hodnotu po sériové lince nebo zobrazte na displej v cyklu s časováním (předpokládá se předešlé vypracování alespoň jedné z uvedených úloh)

3.5.2 Postup práce

- V hlavním programu nastavte parametry A/D převodníku bez interruptů
  - V registru **ATD1C** nastavte povolení převodníku, zarovnání vlevo, výsledek na 8 bitů, bezznaménkový typ a prescaler dle zadání (*např.0b0010*)
    ```
    ATD1C=0b10100010;
    ```
  - Povolte v registru **ATDPE** pro A/D převodník nultý pin
    ```
    ATD1PE=0b00000001;
    ```
- Hodnotu registru **ATD1RH** odešlete po sériové lince nebo zobrazte na displej tak, jak jste zpracovávali zadané číslo v příslušné části
- V `interrupt 5` před zpracováním registru **ATD1RH** nastavte v registru **ATD1SC** vstupní kanál nula a testujte dokončení přenosu prázdnou `while` smyčkou s podmínkou nenastavení příznaku dokončení konverze registru **ATD1SC**
  ```
  ATD1SC=0b00000000;
  while(ATD1SC_CCF==0){
  }
  ```
- Ověřte funkčnost vašeho programu po nahrání do mikroprocesoru

3.6 MĚŘENÍ VLHKOSTI

3.6.1 Zadání

- Odešlete hodnotu vlhkosti po sériové lince nebo zobrazte na displej v cyklu s časováním (předpokládá se předešlé vypracování alespoň jedné z uvedených úloh)

3.6.2 Postup práce

- Zaveďte globální proměnné napeti a vlhkost typu float
  ```
  float napeti,vlhkost;
  ```
- Zaveďte globální proměnnou adhodnota typu unsigned char
  ```
  unsigned char adhodnota;
  ```
- V hlavním programu nastavte parametry A/D převodníku bez interruptů
  - V registru **ATD1C** nastavte povolení převodníku, zarovnání vlevo, výsledek na 8 bitů, bezznaménkový typ a prescaler dle zadání (*např.0b0010*)
    ```
    ATD1C=0b10100010;
    ```
  - Povolte v registru **ATDPE** pro A/D převodník první pin
    ```
    ATD1PE=0b00000010;
    ```
- V `interrupt 5` uložte hodnotu registru **ATD1RH** do proměnné adhodnota
  ```
  adhodnota=ATD1RH;
  ```
- Do proměnné napeti vypočtěte napětí dle (3).
  ```
  napeti=0.0121*adhodnota+0.8;
  ```
- Do proměnné vlhkost vypočtěte vlhkost dle (4).
  ```
  vlhkost=(napeti/0.031)-25.8065;
  ```

- Přepočet vlhkosti na skutečnou provádět nemusíte
- Hodnotu proměnné vlhkost odešlete po sériové lince nebo zobrazte na displej tak, jak jste zpracovávali zadané číslo v příslušné části. Proměnnou je třeba používat s přetypováním na formát unsigned char
  `(unsigned char)vlhkost`
- V `interrupt 5` před zpracováním registru **ATD1RH** nastavte v registru **ATD1SC** vstupní kanál jedna a testujte dokončení přenosu prázdnou `while` smyčkou s podmínkou nenastavení příznaku dokončení konverze registru **ATD1SC**
```
ATD1SC=0b00000001;
while(ATD1SC_CCF==0){
}
```
- Ověřte funkčnost vašeho programu po nahrání do mikroprocesoru

3.7 MĚŘENÍ TEPLOTY

3.7.1 Zadání

- Odešlete hodnotu vlhkosti po sériové lince nebo zobrazte na displej v cyklu s časováním (předpokládá se předešlé vypracování alespoň jedné z uvedených úloh)

3.7.2 Postup práce

- Zaveďte globální proměnné pom, perioda a pulz formátu unsigned int
```
unsigned int pom,perioda,pulz;
```
- Zaveďte globální proměnnou teplota typu float
```
float teplota;
```
- V hlavním programu proveďte nastavení druhého časovače
  - V registr **TPM2SC** nastavte zdroj hodinového signálu BUSCLK a prescaler zadejte dle zadání (*např. 0b000*)
    ```
    TPM2SC=0b00001000;
    ```
  - Pro první kanál nastavte mód INPUT CAPTURE s funkcí zachycení jakékoliv hrany v registru **TPM2C1SC** a povolte interrupt
    ```
    TPM2C1SC=0b01001100;
    ```
- Vytvořte `interrupt 10 preruseni_casovac2` pro zpracování přerušení od prvního kanálu a vynulujte v něm příznak tohoto přerušení **TPM2C1SC_CH1F**
```
void interrupt 10 preruseni_casovac2(void){
TPM2C1SC_CH1F=0;
}
```
- Do `interrupt 10` vložte kód pro zjištění periody při vzestupné hraně čtené z pinu **PTDD_PTDD4**
```
if(PTDD_PTDD4==1)
{
if (TPM2C1V>a) perioda=TPM2C1V-a;
a=TPM2C1V;
}
```

- Do interrupt 10 vložte kód pro zjištění pulzu při vzestupné hraně čtené z pinu **PTDD_PTDD4**

```
if(PTDD_PTDD4==1)
{
if (TPM2C1V>a) pulz=TPM2C1V-a;
}
```

- V `interrupt 5` spočtěte teplotu do proměnné teplota dle (5), přičemž za proměnnou DC dosaďte podíl pulzu a periody. Pulz zadejte z přetypováním na formát float.

```
teplota=(((float)pulz/perioda)-0.32)/0.0047;
```

- Hodnotu proměnné teplota odešlete po sériové lince nebo zobrazte na displej tak, jak jste zpracovávali zadané číslo v příslušné části. Proměnnou je třeba používat s přetypováním na formát unsigned char (zanedbává se zde tedy možnost záporné teploty)

```
(unsigned char)teplota
```

- Ověřte funkčnost vašeho programu po nahrání do mikroprocesoru

4 LITERATURA

[1] *MC9S08GB/GT* [online]. Freescale semiconductors,12/2004
Available from www: < http://freescale.com>

[2] *SMT 160-30*. SMARTEC B.V.

[3] *HIH-3610 Series* [online]. Honeywell
Available from www: < www.honeywell.com/sensing>