RD2 Kit – 40 řešených příkladů v C

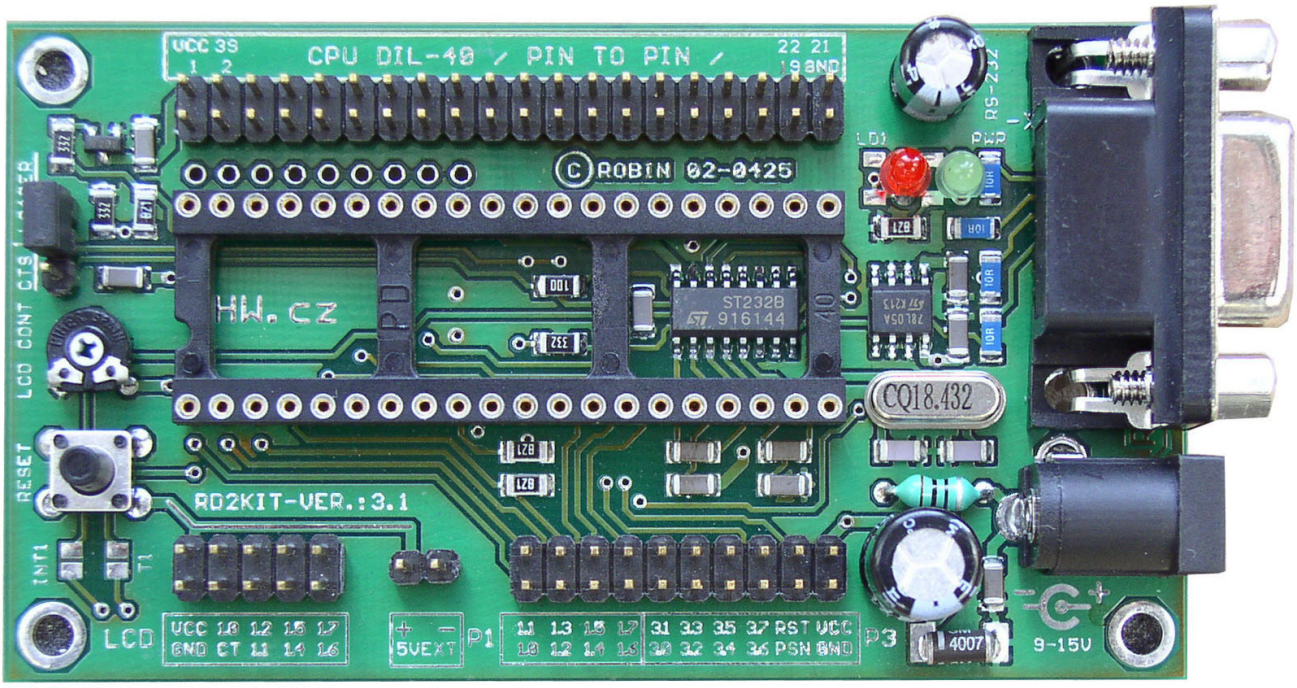
Umíte ASM – začněte programovat i v C

Pro vývoj aplikací v C lze použít řadu platforem, vývojových kitů atd.. Vzhledem ke specifické situaci v ČR, kde je stále ještě většina běžných vývojářů používá ASM a cca 50% lidí programuje x51 kompatibilní procesory jsme připravili dále popsaný vývojový kit, který je cílen především na usnadnění migrace od assembleru k jazyku C pro x51 procesory.

Čím se liší projekt RD2 Kit :

- Jednoduchý HW, který je snadno a levně dostupný.
- Použitý CPU RD2 zajišťuje funkční jednočipové řešení C programu.
- Cca 40 příkladů, které demonstrují základní funkce a rychle Vás vtáhnou do problematiky.
- Všechny příklady jsou k dispozici pro Keil C i SDCC které je k dispozici zdarma.
- Aplikace nejsou omezeny pouze na jedno C konkrétního výrobce.
- Součástí dokumentace je i popis rozdílů jednotlivých kompilerů C.

Pokud vás tedy dále popsaný RD2 Kit zaujal, můžete si jej vyrobit sami, podle popsaných schémat a návodů, nebo si jej můžete objednat z HW serveru.



40 řešených příkladů v C pro Keil i SDCC

Pro studijní účely a pochopení základních principů návrhu jednočipových aplikací s procesory řady 8051 je RD2-kit vybaven sadou příkladů v jazyce C. Ukázkové příklady jsou určeny pro vývojové nástroje Keil C51 evaluation version a SDCC. Porovnáním těchto dvou nástrojů máme možnost si vytvořit konkrétní náhled na problematiku vývoje jednočipových aplikací ve vyšším programovacím jazyce. Naprogramování vytvořené aplikace do JM z PC se provádí metodou ISP přes sériové rozhraní. Tato metoda nevyžaduje žádný další podpůrný hardware a zcela eliminuje použití klasického programátoru.

Kromě zdrojových kódů najdete na CD také .HEX soubory, které lze pomocí Flasheru nebo programu FLIP rovnou nahrát do aplikace a spustit.

Jednotlivé příklady jsou členěny do několika tématických skupin

Úvodní program v C na T89c51RD2

- popis překladu a sestavení jednoduchého programu v jazyce C, naprogramování JM metodou ISP.

Sériový kanál

- nastavení 8-bitové sériové komunikace, vlastní funkce pro čtení a zápis dat přes sériový kanál,
- použití standardních funkcí jazyka C pro formátovaný vstup/výstup dat přes sériový kanál,
- celočíselný kalkulátor.

LCD displej 2 x 16 znaků v 4-bitovém režimu komunikace

- nastavení 4-bitového režimu komunikace, funkce pro čtení a zápis dat na LCD displej,
- uživatelská znaková sada, posuvy textu a animace,
- zasílání příkazů LCD displeji přes sériový kanál,
- přesměrování standardního výstupu funkce printf na LCD a sériový kanál.

Maticová klávesnice 4 x 3

- funkce pro čtení klávesnice, diagnostika klávesnice s výstupem na sériový kanál,
- sdílení datové sběrnice LCD displeje s klávesnicí, výstup dat na LCD.

Systém přerušení

- časovač T2 s obvodovým 16-bitovým přednastavením, obsluha led diody v rutině přerušení,
- hodiny reálného času s časovačem T2,
- obsluha vnějšího přerušení INT0.

Časovač watchdog

- ovládání časovače watchdog a jeho použití při kontrole běhu aplikace,
- řízené nulování JM časovačem watchdog.

Čítačem podporované programovatelné pole PCA, časovač T2

- 8-bitová pulsně šířkové modulace,
- měření délky pulsu,
- programovatelný generátor pulsů.

Paměť programu FLASH

- ověření integrity kódu programu na základě výpočtu kontrolního součtu souvislého bloku paměti FLASH,
- volání API funkcí uživatelského zavaděče pro práci s pamětí FLASH,
- přístup do paměti XAF (eXtra Array Flash).

Paměť EEPROM

- obsluha paměti EEPROM, čtení, zápis a výpis obsahu paměti přes sériový kanál.

Vnější paměť dat XRAM

- nastavení dostupné velikosti vnější paměti dat XRAM,
- test paměťových buněk a výpis obsahu paměti přes sériový kanál.

Speciální funkce procesoru

- vypnutí generování signálu ALE,
- režimy se sníženou spotřebou (Idle Mode, Power-Down Mode),
- programové přepínání násobičky hodin (X2 - Mode).

Diagnostika JM

- ověření funkčnosti jednotlivých bloků JM a připojených periferií.

Pokročilé použití vývojových nástrojů

- vkládání instrukcí jazyka symbolických adres do zdrojového kódu jazyka C,
- sestavení výsledného programu na uživatelem definovaných adresách.

Sériový kanál

Příklad 1.1

Navrhněte program pro komunikaci po sériovém kanálu s přenosovými parametry 19200, N, 8, 1. Napište vlastní funkce pro operace čtení a zápisu dat na sériovém kanálu. Funkčnost demonstrujte na příkladě vysílání textového řetězce zakončeného znakem 00h a jednoduchém emulátoru terminálu (vrací zpět stištěné klávesy).

Sériový kanál přepneme do módu 1 – 8bitový UART. Přenosová rychlost sériového kanálu je volitelná a je dána periodou přetečení časovače T1 a hodnotou bitu SMOD v registru PCON. Časovač T1 pracuje v módu 2 s obvodovým přednastavením a pro přenosovou rychlost platí následující vztah

$$Přenosová\ \ rychlost = \frac{2^{SMOD}}{32} \cdot \frac{F_{OSC}}{12 \cdot [256 - TH1]} \qquad (1)$$

Výpočet hodnoty TH1 časovače T1 pro různé přenosové rychlosti zautomatizujeme pomocí makra bez parametrů (příkaz define) Timer1Mode1ReloadValue. Ve všech následujících příkladech pracujících se sériovým kanálem budeme implicitně používat přenosovou rychlost 19200 baudů.

Obsluha sériového kanálu je řešena bez využití přerušovacího systému. Vysílání a příjem dat provádíme programově pomocí patřičných podprogramů pro vysílání (ser_putch(), ser_puts()) a příjem (ser_getch()).

Sestavení programu a naprogramování JM je shodné s vzorovým popisem uvedeným u programu blikající led diody. K zobrazení dat posílaných po sériovém kanálu použijeme libovolný sériový terminál. Nastavíme přenosové parametry 19200, N, 8, 1 a vypneme řízení toku dat. V případě, že využíváme programové řízení vývodu PSEN při ISP programování je nutné zkontrolovat, aby sériový terminál nenastavoval signál RTS. Nastavením signálu RTS by došlo k uzemnění vývodu PSEN JM během nulování JM a spuštění zavaděče na adrese FC00h místo uživatelské aplikace.

Příklad 1.2

Nahraďte vlastní funkce z příkladu 1.1 za standardní funkce jazyka C pro formátovaný vstup/výstup dat na sériovém kanálu. Použijte funkci printf() a zobrazte numerickou hodnotu čísel datového typu unsigned char, long a int co nejvíce možnými způsoby. Využijte různé typy řídícího řetězce formátu funkce printf().

Pro vstupy a výstupy používáme funkce, které při vstupu čísla načtou celé číslo jako řetězec a převedou ho automaticky do číselné podoby (scanf()) nebo naopak, při výstupu samy konvertují hodnotu číselné proměnné na odpovídající posloupnost číslic (printf()).

Použití standardních funkcí vstupu/výstupu je stejné jak známe z programování na PC. Implicitně se u těchto funkcí předpokládá práce se sériovým kanálem. Drobné odchylky jsou u řídících řetězců formátu, které jsou částečně optimalizované pro použití na platformě x51.

Pokud používáme v SDCC funkci printf() je třeba napsat vlastní funkci putchar() viz příklad 2.4.

Příklad 1.3

Navrhněte program jednoduchého kalkulátoru schopného provádět operace sčítání, odčítání, dělení a násobení s celými čísly. Způsob realizace vstupně/výstupních funkcí volte podle vlastního uvážení.

V případě absence některých standardních funkcí jazyka C implementovaných na platformě x51 musíme využít alternativních řešení. Program kalkulátoru načítá dvě celá čísla a symbol matematické operace. Načtení čísel může obstarat funkce scanf(), která lze však v tomto případě nahradit např. funkcemi sscanf() nebo atoi(). Zobrazení výsledku na sériovém kanálu využívá funkcí popsaných v předchozích příkladech.

LCD displej 2 x 16 znaků v 4 - bitovém režimu komunikace

Příklad 2.1

Navrhněte program pro obsluhu LCD displeje 2 x 16 znaků v 4 - bitovém režimu komunikace. Zaměřte se zejména na správnou inicializaci displeje a programovou kontrolu příznaku BUSY. Napište funkce pro zasílání příkazů a znaků řadiči displeje. Funkčnost demonstrujte na příkladě vysílání textového řetězce zakončeného znakem 00h.

Obsluha displeje v 4 - bitovém režimu komunikace přináší úsporu čtyř datových vodičů D0 až D3. Na druhou stranu využívá složitější komunikační protokol. Displej je připojen na bránu P1 a využívá všech vývodů mimo P1.3. Programově je ošetřeno, že vývod P1.3 není ovlivňován komunikací s displejem a lze ho libovolně použít pro v/v operace.

Použitý LCD displej je osazen řadičem kompatibilním s HD44780. Pro řízení displeje jsou důležité tři základní funkce na kterých založíme naše LCD API rozhraní. Jedná se o inicializaci displeje v 4 - bitovém režimu komunikace, zasílání dat řadiči (příkazy, znaky) a kontrola příznaku BUSY. Detailní popis jednotlivých funkcí je nad rámec tohoto textu. Potřebné informace a průběhy řídících signálů jsou uvedeny v katalogovém listu řadiče HD44780.

V některých případech po ne zcela korektním zapnutí napájecího napětí docházelo k problémům s inicializací displeje a celého JM. Problém byl odstraněn použitím funkce PowerOnReset() detailně popsané v příkladu 5.2.

Příklad 2.2

Příklad 2.1 doplňte o možnost dohrání uživatelské znakové sady. Navrhněte funkce pro plynulý posuv textového řetězce na displeji. Uživatelskou znakovou sadu využijte k jednoduché animaci.

Prvních osm znaků ve znakové sadě lze nastavit, což umožňuje na displeji tvořit různé efekty, grafické symboly, češtinu nebo animace. Použitá zobrazovací matice displeje a nastavený režim určují, zda k zobrazení uživatelsky definovaných znaků bude použit rastr 5 x 7 nebo 5 x 10 bodů.

Definice jednotlivých znaků je uložena v paměti kódu programu. Jedná se celkem o 64 bajtů a znak definuje 8 bajtů, které v paměti řadiče displeje CGRAM na sebe postupně navazují.

Vytvořená animace zobrazuje postupně řetězec, který je posouván na displeji zprava do leva. Vhodnou volbou zobrazované posloupnosti vznikne jednoduchá grafická animace.

Příklad 2.3

Po sériovém kanálu zobrazte tabulku základních příkazů pro ovládání displeje. Uživatel má možnost výběru jednotlivých příkazů, kterými ovládá chování displeje.

Program slouží pro demonstraci vytvořeného API rozhraní displeje. Uživatel vidí jednotlivé popisy API funkcí na sériovém terminálu. Zmáčknutím klávesy přidělené dané funkci vyvolá patřičnou odezvu na displeji.

Příklad 2.4

Využijte možnosti přesměrování standardního výstupu funkce printf() do více zařízení. Funkce printf() pro výpis dat na nejnižší úrovni využívá volání funkce putchar(). Napište vlastní funkci putchar() s možností zobrazení dat na LCD a sériovém kanálu. Funkčnost demonstrujte na příkladu volání funkce printf().

Funkce printf() na základě řídícího řetězce formátu vytváří textový řetězec, který je následně zobrazen pomocí funkce putchar(). Implicitně funkce printf() předpokládá výstup na sériový kanál. V případě, že nahradíme funkci putchar() svou vlastní funkcí naskýtá se nám možnost přesměrovat výstup na další zařízení. Je však třeba mít na paměti následující věc. Při definici vlastní funkce putchar() nesmí dojít k rozporu s deklarací funkce putchar() uvedenou v hlavičkovém souboru stdio.h použitého překladače. Globální proměnnou poté volíme aktuální výstupní zařízení použité funkcí putchar() pro výstup dat.

Maticová klávesnice 4 x 3 tlačítka

Příklad 3.1

Navrhněte program pro čtení maticové klávesnice 4 x 3 tlačítka. Vyzvěte uživatele k zmáčknutí jednotlivých tlačítek a detekované tlačítko zobrazte po sériovém kanálu.

Maticová klávesnice je připojena pomocí sedmi vodičů. K výběru sloupce klávesnice slouží vývody P3.5 až P3.7. Aktuální stav jednotlivých řádků klávesnice čteme na vstupech P1.4 až P1.7. Postupně jednotlivé sloupce přepínáme do stavu log.0 a čtením řádků zjišťujeme zmáčknuté klávesy. Navržené řešení zjišťuje pouze jednu zmáčknutou klávesu. Ověření funkčnosti je demonstrováno na jednoduchém programu, který zmáčknuté klávesy zobrazuje po sériovém kanálu.

Příklad 3.2

Program obsluhy klávesnice upravte tak, aby pro čtení řádků klávesnice využil datové sběrnice použité u LCD displeje v 4 - bitovém režimu komunikace. Funkčnost demonstrujte na příkladu přístupového systému s kódovým zámkem (zadání nového hesla a ověření správnosti hesla) a výstupem na LCD.

U aplikací kde jsme limitováni počtem volných vývodů JM se může uplatnit následující řešení. Data na sběrnici LCD displeje jsou platná pokud se nachází signál ENABLE v log. 1. V opačném případě je sběrnice volně k dispozici ostatním periferiím. Stávající funkce pro obsluhu klávesnice doplníme o instrukci, která přepne signál ENABLE do log. 0 vždy, když provádíme čtení klávesnice. Nechtěnému ovlivňováni komunikace LCD s JM ze strany klávesnice zamezíme přepnutím sloupců klávesnice do stavu log. 1.

Systém přerušení

Příklad 4.1

Navrhněte program obsluhující rutinu přerušení od časovače T2 s obvodovým 16 - bitovým přednastavením. Nastavte periodu blikání led diody 2000 ms. Využijte vstupu P2.0 k prodloužení periody blikání (log. 0 … zvětšení periody).

Přerušení od časovače T2 se vyvolá nastavením příznaku TF2. Výhoda použití časovače T2 v režimu s 16 - bitovým přednastavením je, že současně s přetečením časovače dojde k automatickému přednastavení. Funkce časovače je tím pádem nezávislá na hlavním programu. Přednastavenou hodnotu časovače T2 vypočteme podle následujícího vztahu

$$[TH2, TL2] = 10000h - \frac{F_{OSC} \cdot t}{12} \qquad (2)$$

kde t je čas v sekundách za jak dlouho má dojít k přerušení. F_{OSC} kmitočet použitého oscilátoru v Hz a konstanta 12 je odvozena od počtu period oscilátoru na jeden strojový cyklus.

Definice funkce obsluhy přerušení v jazyce C se vyznačuje použitím atributu interrupt s číslem přerušení. Klíčové slovo using umožňuje volbu použité banky registrů. Pro časovač T2 bude definice funkce obsluhy přerušení vypadat následovně.

```
static void Reload_Timer2_ISR(void) interrupt 5 using 3
{ LD1 = ~LD1;                      // toggle LED every 1000 ms
  TF2 = 0;                         // clear timer 2 overflow flag
}
```

Příklad 4.2

Navrhněte program hodin reálného času úpravou příkladu 4.1. Výstup hodin ve formátu HH:MM:SS zobrazte na LCD nebo sériovém kanálu.

Přesnost hodin je dána dobou mezi dvěmi přetečeními časovače T2. Obsluha přerušení by měla obecně vždy obsahovat jen ty nejnutnější instrukce, např. pro výpočet času postačují proměnné typu unsigned char. Zobrazení času na výstupním zařízení provádíme v hlavní smyčce programu.

Příklad 4.3

Na vstup externího přerušení INT0 připojte tlačítko. Provádějte detekci vstupního signálu externího přerušení INT0 na sestupnou hranu. Vykonání rutiny přerušení INT0 indikujte změnou stavu led diody.

Vnější přerušení /INT0 a /INT1 mohou být vyvolána buď logickou úrovní nula nebo sestupnou hranou na patřičném vstupu JM. Programová obsluha rutiny přerušení je obdobná jako v předchozích dvou příkladech. Při práci s vnějšími přerušeními je důležité jak vypadá průběh vstupního signálu. Konkrétně v našem případě zákmity generované stiskem tlačítka mohou vyvolat nechtěnou žádost o přerušení. Pokud používáme detekci nulové úrovně je třeba, aby tato trvala minimální dobu uvedenou v katalogovém listu výrobce.

Časovač watchdog

Příklad 5.1

Navrhněte program využívající funkce časovače watchdog. Vstupy reálné aplikace simulujte stisknutím klávesy v sériovém terminálu. Včasný stisk klávesy nuluje čítač časovače watchdog v opačném případě se hodnota čítače inkrementuje a může dojít k nulování celého JM.

Časovač watchdog slouží ke kontrole běhu aplikace a nulování JM v případě, že došlo k selhaní programu. Funkci časovače watchdog aktivujeme postupným zápisem hodnot 1Eh a E1h do registru WDTRST. Hodnota čítače časovače watchdog se postupně inkrementuje. Dojde-li k přetečení, JM se automaticky vynuluje. Přetečení můžeme zamezit periodickým zápisem hodnot 1Eh a E1h do registru WDTRST. Periodu časovače watchdog lze prodloužit nastavením dalšího čítače 2^7 v registru WDTPRG.

Příklad 5.2

Navrhněte program využívající informace uvedené v registru PCON příznak Power-Off. Rozlište stav kdy došlo k zapnutí napájecího napětí u JM nebo nulování obvodu nulovacím tlačítkem. Časovač watchdog použijte k programem řízenému nulování JM.
Hodnota příznaku Power-Off se nastaví po zapnutí napájecího napětí do log. 1 a není ovlivněna nulováním JM nulovacím tlačítkem. Této vlastnosti využijeme k detekci stavu po zapnutí napájecího napětí. Spustíme časovač watchdog a po jeho přetečení se automaticky provede cílené nulování JM.

Čítačem podporované programovatelné pole PCA, časovač T2

Příklad 6.1

Navrhněte program využívající funkce čítačem podporovaného programovatelného pole k vytváření 8 - bitové pulsně šířkové modulace. Výstup PWM signálu přiveďte na led diodu P1.3. Měňte hodnotu střídy a na led diodě pozorujte postupnou změnu svítivosti.
PCA se skládá z 16 - bitového čítače/časovače a pěti 16 - bitových komparačních nebo záchytných modulů. Čítač/časovač je společný pro všechny moduly. Jednotlivé moduly lze provozovat v komparačním režimu(16 - bitový časovač, rychlý výstupní mód, pulsně šířkový modulátor) nebo záchytném režimu (měření délky pulsů).
Samotný proces generování PWM signálu spočívá v nastavení patřičného modulu do komparačního režimu. Pulsně šířkový generátor pracuje jako 8 - bitový čítač s obvodovým přednastavením. Po přetečení je registr CCAPLn obnoven z registru CCAPHn. Změnou hodnoty CCAPHn lze plynule nastavovat střídu generovaného signálu.

Příklad 6.2

Navrhněte program využívající funkce čítačem podporovaného programovatelného pole k měření délky pulsů. Zdrojem pulsů je teplotní čidlo SMT160 - 30 převodník teplota/střída. Změřené údaje numericky zpracujte a výslednou teplotu zobrazte na displeji.

V záchytném režimu lze nastavit požadavek o přerušení od náběžné, sestupné nebo obou hran signálu připojeného na vstup PCA modulu. Společně s požadavkem o přerušení dojde k zápisu obsahu čítače CH, CL do registrů CCAPnH a CCAPnL.

Teplotní čidlo SMT160 - 30 připojíme na vstup PCA modulu 0. V rutině přerušení ukládáme hodnotu registrů CCAP0H a CCAP0L do připravené struktury popisující obdélníkový puls. Na základě těchto údajů vypočteme podle následujícího vztahu aktuální teplotu t [° C] a zobrazíme ji na displeji.

$$t = \frac{1446 \cdot T_1 - 681 \cdot T_0}{T_1 + T_0} \qquad (3)$$

T_1 je stav log. 1 a T_0 stav log. 0

Příklad 6.3

Navrhněte program využívající funkce časovače T2 ke generování pulsů na výstupu P1.0. Zadaný kmitočet signálu čtěte ze sériového kanálu. Zobrazte hodnotu požadovaného a reálně generovaného kmitočtu na výstupu P1.0. Vzniklou odchylku zdůvodněte.

Časovač T2 přepneme do režimu generování pulsů (16 - bitový čítač s obvodovým přednastavením). Kmitočet generovaný na výstupu P1.0 je dán vztahem 4

$$F_{OUT} = \frac{F_{OSC}}{4 \cdot [65536 - [RCAP2H, RCAP2L]]} \qquad (4)$$

Zadanou hodnotu kmitočtu dosadíme do vztahu 4 a vypočteme hodnotu registrů RCAP2H, RCAP2L. U vyšších kmitočtů dochází k numerické chybě při zaokrouhlování na celé 16 - bitové číslo. To má za následek vznik odchylky mezi zadaným a generovaným kmitočtem.

Příklad 6.4

Časovač/čítač T0, T2 - měření kmitočtu
Navrhněte program využívající funkce čítače T0 pro čítání externích pulsů na vstupu P3.4 JM. Zdroj pulsů realizujte časovačem T2 ve funkci generátoru pulsů na výstupu P1.0, který připojte na vstup P3.4. Vypočtěte hodnotu kmitočtu a periodu změřeného signálu a zobrazte na sériovém kanálu. Maximální kmitočet, který lze měřit (přivést) na vstupu P3.4 je roven hodnotě 1/24 použitého oscilátoru u JM. Měření provádějte pro kmitočtový rozsah 100 Hz – 750 kHz.

Příklad 6.5

Čítačem podporované programovatelné pole PCA, časovač T2 - měření periody

Navrhněte program využívající funkce čítačem podporovaného programovatelného pole k měření délky pulsů. Zdroj pulsů realizujte časovačem T2 ve funkci generátoru pulsů na výstupu P1.0, který připojte na vstup PCA P1.3. Vypočtěte hodnotu kmitočtu a periodu signálu změřeného signálu a zobrazte na sériovém kanálu. Měření provádějte pro kmitočtový rozsah 100 Hz – 10 kHz.

Příklad 6.6

Systém přerušení - Alarm
Navrhněte program spínající čtyři výstupy JM po uplynutí přednastavené doby u jednotlivých výstupů. Využijte hodin reálného času z příkladu 4.2. pro časovou základnu celého zařízení. Režim nastavení hodin aktivujte zmáčknutím klávesy „*“ na maticové klávesnici. Nastavení jednotlivých alarmů proveďte klávesou „#“ a patřičným číslem alarmu (0 … 3). Aktuální stav jednotlivých alarmů a čas zobrazte na LCD.

Příklad 6.7

Časovač/čítač T0, T2 - měření kmitočtu
Navrhněte program využívající funkce čítače T0 pro čítání externích pulsů na vstupu P3.4 JM. Zdroj pulsů realizujte časovačem T2 ve funkci generátoru pulsů na výstupu P1.0, který připojte na vstup P3.4. Vypočtěte hodnotu kmitočtu a periodu změřeného signálu a zobrazte na sériovém kanálu. Maximální kmitočet, který lze měřit (přivést) na vstupu P3.4 je roven hodnotě 1/24 použitého oscilátoru u JM. Měření provádějte pro kmitočtový rozsah 100 Hz – 750 kHz.

Příklad 6.8

Čítačem podporované programovatelné pole PCA, časovač T2 - měření periody
Navrhněte program využívající funkce čítačem podporovaného programovatelného pole k měření délky pulsů. Zdroj pulsů realizujte časovačem T2 ve funkci generátoru pulsů na výstupu P1.0, který připojte na vstup PCA P1.3. Vypočtěte hodnotu kmitočtu a periodu signálu změřeného signálu a zobrazte na sériovém kanálu. Měření provádějte pro kmitočtový rozsah 100 Hz – 10 kHz.

Paměť programu FLASH

Příklad 7.1

Navrhněte program ověření integrity kódu programu na základě výpočtu kontrolního součtu souvislého bloku paměti FLASH. K tomuto účelu využijte funkci programu RD2 - Flasheru s parametrem –s. Algoritmus výpočtu kontrolního součtu je shodný s algoritmem použitým u souborů typu Intel - HEX. Čtení paměti FLASH provádějte pomocí ukazatele do paměti kódu programu.

Program RD2 - Flasher umožňuje výpočet kontrolního součtu souvislého bloku paměti FLASH. Vypočtená hodnota je uložena na adrese následující za zabezpečeným blokem paměti. Obsah paměti je v pořádku, pokud suma všech bajtů v kontrolovaném bloku paměti FLASH a kontrolního součtu je rovna nule.

Rozšíření jazyka C na platformě 8051 umožňují mimo jiné optimalizovat práci s ukazateli. Programátor může určit paměť, kde je ukazatel uložen a paměť do které odkazuje. Ukazatel uložený v interní paměti JM ukazující do paměti programu definujeme následovně.

```
Keil C51    unsigned char code *data ptr;
SDCC   code unsigned char *data ptr;
```

Příklad 7.2

Navrhněte program komunikující pomocí API rozhraní s uživatelským zavaděčem pro práci s pamětí FLASH. Zobrazte všechny dostupné informace Manufacturer ID, ID1, ID2, … na sériový kanál.

API rozhraní uživatelského zavaděče je detailně popsáno v katalogovém listě JM T89c51RD2 na straně 61 až 63. Princip používání jednotlivých API funkcí spočívá v naplnění patřičných registrů (R1, DPH, DPL, ACC) a volání funkce na adrese FFF0h. Legitimní řešení předávání parametrů v registrech JM si vynutilo napsání podprogramu v jazyce symbolických adres. Tento podprogram připojíme k hlavnímu programu v jazyce C odkud ho následně voláme.

Konvence volání a předávání parametrů se v jednotlivých vývojových nástrojích liší. Zde se osvědčil následující trik. V jazyce C napíšeme funkci, která má vstupní a výstupní parametry shodné s plánovanou funkcí v jazyce symbolických adres. Překladem této funkce získáme funkční kód v jazyce symbolických adres, který doplníme o námi požadované operace a jsme hotovi.

Příklad 7.3

Navrhněte program přistupující přímo do paměti XAF (eXtra Array Flash). Zobrazte obdobně jako v příkladě 7.2 dostupné informace na sériový kanál.

Paměť XAF má velikost 128 B a je zcela oddělena od paměti programu. XAF obsahuje informace používané během ISP režimu programování. Tyto informace lze číst buď voláním API rozhraní zavaděče (viz příklad 7.2) nebo přímým přístupem do paměti FLASH. Čtení a zápis paměti FLASH je řízeno registrem FCON. Adresa jednotlivých údajů v paměti XAF je uvedena katalogovém listu JM T89c51RD2 na straně 68 a 69.

Paměť EEPROM

Příklad 8.1

Navrhněte program pracující s integrovanou pamětí EEPROM na čipu JM. Napište funkce pro čtení a zápis dat do paměti EEPROM. Obsah paměti zobrazte pomocí jednoduchého prohlížeče paměti s výstupem na sériový kanál.
Integrovaná paměť EEPROM je přístupná na adresách 0000h až 07FFh a překrývá se s vnější datovou pamětí XRAM. Výběr paměti EEPROM provedeme nastavením bitu EEE v registru EECON. Následně lze libovolně přistupovat do paměti EEPROM instrukcí MOVX. Registr EETIM nastavuje časování vnitřních operací paměti EEPROM v závislosti na připojeném krystalu. Zápis do paměti lze provádět po jednotlivých bajtech v rámci jedné stránky nebo po stránkách (64 B). Operaci zápisu potvrdíme sekvenčním zápisem hodnot 50h a A0h do registru EECON. Ukončení zápisu je ze strany hardwaru indikováno vynulováním příznaku EEBUSY.

Poznámka: *Seznam chyb JM T89c51RD2 uveřejněný firmou Atmel uvádí, že je třeba provést zápis dat třikrát za sebou, aby došlo ke korektnímu naprogramování paměti EEPROM.*

Vnější paměť dat XRAM

Příklad 9.1

Navrhněte program pracující s integrovanou pamětí XRAM na čipu JM. Nastavte skutečnou velikost dostupné paměti XRAM. Otestujte jednotlivé paměťové buňky a adresové vodiče dekodéru paměti. Obsah paměti zobrazte pomocí jednoduchého prohlížeče paměti s výstupem na sériový kanál.

Velikost integrované paměti XRAM na čipu je 1024 B. Po zapnutí/nulování JM je však k dispozici pouze 768 B (kompatibilní režim s T87c51RD2). V registru AUXR lze programově nastavit velikost dostupné paměti. Jednotlivé paměťové buňky ověříme postupným zápisem a následným čtením konstant AAh a 55h. Přístup do paměti je řešen pomocí makra XBYTE, které slouží k zápisu/čtení jednotlivých položek paměti XRAM. Obdobně lze použít další makra (CBYTE, DBYTE, PBYTE, XWORD, …) pro přístup do ostatních paměťových prostorů JM.

Speciální funkce procesoru

Příklad 10.1

Navrhněte program schopný vypnout/zapnout generování signálu ALE na příslušném výstupu JM.

Pro řízení vnější paměti je JM vybaven řídícími signály ALE (pro zápis spodní poloviny platné adresy A0 až A7), /PSEN (pro čtení z vnější paměti programu), /RD a /WR (pro čtení nebo zápis z vnější datové paměti). Signál ALE je periodicky generován (F_{OSC}/6) nezávisle na tom jestli je připojena externí datová paměť programu nebo dat. Nastavením bitu AO v registru AUXR lze vypnout automatické generování signálu ALE a tím snížit generované rušení. Signál ALE se poté generuje pouze během provádění instrukcí MOVX a MOVC.

Příklad 10.2

Navrhněte program využívající režimů se sníženou spotřebou (Idle Mode, Power - Down Mode). Na led diodě P1.3 zobrazte aktuální stav JM (svítící dioda indikuje probíhající program a naopak). Použijte vstup P2.0 v log. 0 k aktivaci režimu snížené spotřeby. Tlačítkem připojeným na vstup externího přerušení /INT0 jej ukončete.

U aplikací nevyžadujících soustavnou činnost a mají požadavek na nízkou spotřebovanou energie, je možné využít režimů se sníženou spotřebou. JM T89c51RD2 je vybaven dvěmi režimy se sníženou spotřebou (Idle Mode, Power - Down Mode), do kterých lze JM uvést nastavením příslušných bitů v registru PCON. Instrukce nastavující patřiční režim je poslední instrukcí provedenou před přechodem do režimu se sníženou spotřebou. Tento režim v našem případě opustíme po výskytu požadavku o vnější přerušení na vstupu /INT0.

Příklad 10.3

Navrhněte program obsluhující rutinu přerušení od časovače T2 s obvodovým 16 - bitovým přednastavením. Nastavte periodu blikání led diody 2000 ms. Využijte vstupu P2.0 k programovému přepínání násobičky hodin (X2 - Mode … log. 0 … zvětšení periody).

Postup nastavení časovače T2 s obvodovým 16-bitovým přednastavením je popsáno v příkladu 4.1. Programové přepínání hodin se provádí bitem X2 v registru CKCON. U jednotlivých periferií JM lze selektivně nastavit počet period (6 nebo 12) oscilátoru na jeden strojový cyklus.

Diagnostika RD2 Kitu

Příklad 11.1

Navrhněte program ověřující funkčnost jednotlivých bloků JM a připojených periferií.

Zaměřte se na následující testy :

- sériový kanál čtení a zápis dat,
- led dioda na výstupu P1.3,
- integrita kódu programu chráněná kontrolním součtem,
- externí vstup přerušení INT0 s tlačítkem,
- LCD displej 2 x 16 znaků v 4 - bitovém režimu komunikace,
- maticová klávesnice 4 x 3 tlačítka.

Jednotlivé body zadání jsou detailně probrány v rámci dosavadních příkladů zabývajících danou problematikou. Záměrem tohoto příkladu je zopakovat a shrnout znalosti získané v předešlých příkladech. Zároveň vytvoříme testovací nástroj, který ověřuje funkčnost základních bloků JM a připojených periferií. Aplikace je vhodná pro situace, kdy máme podezření na nefunkčnost některé periferie.

Pokročilé použití vývojových nástrojů

Příklad 12.1

Navrhněte program určující paritu jednobajtového čísla zadaného na sériovém kanále. K zjištění parity využijte příznaku parity uloženého ve stavovém slově. Je-li v akumulátoru lichý počet jedniček, potom příznak parity je nastaven na P = 1. Práci s akumulátorem a určení parity proveďte v jazyku symbolických instrukcí, který vložte do zdrojového kódu programu jazyka C.

Keil C51

V okně projektu vybereme zdrojový kód programu *main.c*, zmáčkneme pravé tlačítko a zvolíme položku *Options for file main.c*. Zobrazené okno obsahuje nezaškrtnuté volby *Generate Assembler SRC file* a *Assemble SRC file.* Tyto volby je nutné zaškrtnout, aby došlo k překladu programu obsahující následující pragma pro vkládaný kód jazyka symbolických adres.

```
#pragma asm

zdrojový kód programu v jazyku symbolických adres

#pragma endasm
```

Sestavení programu vyžaduje navíc manuální doplnění všech použitých knihoven. Toto provedeme v okně projektu, kde postupně vložíme použité knihovny z adresáře */C51/LIB*. V našem případě se jedná o knihovnu *C51S.LIB*.

SDCC

Použití vkládaného kódu jazyka symbolických adres je v překladači SDCC mnohem jednodušší. Ve zdrojovém kódu jazyka C stačí pouze uvést klíčová slova _asm a _endasm;, která ohraničují kód v jazyce symbolických adres. Překlad a sestavení programu není třeba nikterak modifikovat.

```
_asm

zdrojový kód programu v jazyku symbolických adres

_endasm;     // je třeba zakončit středníkem
```

Při použití vkládaných instrukcí jazyka symbolických adres je třeba dbát na to, že se jedná o zásah do struktury programu napsaného v jazyce C. Neopatrné použití může vést k fatálním následkům.

Příklad 12.2

Navrhněte program vypisující textový řetěz na sériový kanál. Zaměřte se na fázi sestavení programu linkerem kdy dochází k přidělení relativního kódu absolutním adresám. V závislosti na možnostech použitých vývojových nástrojů nastavte počáteční adresu programu, uložení textových řetězců na stanovenou adresu v paměti programu, atd.

Keil C51

Linker BL51 umožňuje cílené řízení sestavení výsledného programu. Lze specifikovat adresy kde mají být uloženy jednotlivé funkce, moduly, … V menu *Project* → *Options for Target …*→ *BL51 Locate* pole *CODE* uvedeme názvy jednotlivých funkcí(modulů) s patřičnou počáteční adresou. Při zadávání musíme použít názvy uvedené v souboru .M51 (tabulka symbolů). Např. funkci ser_init() umístíme na adresu 900h následujícím zápisem ?PR?SER_INIT?MAIN(0900h).

SDCC

Možnosti SDCC při sestavování programu jsou omezeny pouze na určení počáteční adresy odkud bude uložen výsledný kód programu. Adresa je předána linkeru parametrem *--code-loc* s udanou počáteční adresou.

Závěr

Zdrojové kódy popsaných aplikací, najdete na CD k RD2 Kitu, které bude možné koupit také samostatně v internetovém obchodu HW serveru, nebo v distribuci BEN.

Součástí CD je i sériový kód, který vám zpřístupní aktualizovanou databázi zdrojových kódů a dalších příkladů pro RD2 Kit.

Další informace o této problematice najdete na těchto internetových adresách :

- **www.HW.cz** - HW server : Výrobce RD2 Kitu, stránky celého projektu..
- **www.keil.com** - Výrobce Keil µVision2 v jehož demoverzi jsou popsané příklady kompilovatelné..
- **sdcc.sourceforge.net** – Domovská stránka projektu SDCC

Kolektiv autorů z HW serveru