Řízení LCD modulu KTM-S1201 -

µPD7225

Napsal/a: Žirafka

Datum zveřejnění: : 18. 12. 2015 v 17:30

Před časem jsem někde získala levný LCD modul KTM-S1201 ve kterém je řadič NEC µPD7225. Dlouho ležel ve škatulce, ale teďka mne napadlo, že bych si mohla pohrát a tak jsem si pohrála.

Řízení displeje není tak triviální, jak to na první pohled vypadalo. Ačkoli vlastní komunikace se mi podařila velmi rychle, tak některé věci mi dali docela dost zabrat a nejvíce šedých vlasů mám z pochopení toho, jak funguje režim „blikání“ a jak jej pořádně zprovoznit. Nicméně se nakonec všechno podařilo a výsledky si tu můžete prohlédnout.

Obr. 1 – LCD modul KTM-S1201

Displej je připojený pomocí těchto signálů:

Sck – hodinový signál pro synchronizaci přenosů dat a příkazů mezi řídícím procesorem a vlastním řadičem
Si – sériová data
Cd – přepínání přenosu dat nebo příkazů (CD=0 pro data, CD=1 pro příkazy)
Cs – uvolnění přenosu (CS=0 přenos povolen, CS=1 přenos blokován)
Res – reset řadiče displeje (Res=0 reset displeje, Res=1 normální provoz)

Po resetu řadiče je potřeba do modulu poslat inicializační sekvenci. Tu mám přímo z dokumentace modulu a vlastně není potřeba ji nějak měnit.

&H40 - vychází ze zapojení modulu (viz níže)
&H30 – asynchronní režim
&H18 – vypnuté blikání
&H11 – zapnutí displeje
&H15 – zapnutí sedmi segmentového dekodéru
&H20 – vymazání obrazové paměti

První příkaz nastaví režim řadiče takto:

Divide-by-4 time division drive
1 / 3 Bias Metod
Frequency division ratio $1/2^7$

V některých případech je to potřeba vědět, hlavně pokud chcete přímo zapisovat do obrazové paměti a nebo používat režim blikání.

Zbytek parametrů je nejlépe si prohlédnout přímo v programu a nebo dokumentaci od výrobce řadiče. Nemá cenu zde program znovu rozepisovat. Nicméně je potřeba se podívat na režim blikání, protože jsem přes intenzivní Googlení na webu nenašla příklad, který by správně fungoval. Dost často se v ukázkových, i knihovních, příkladech píše, že některé segmenty blikají a jiné ne a nikdo neví proč. No a já to vím

Blikání jednoho znaku

Pokud se potřeba zapnout blikání jednoho znaku, je potřeba trocha programování. Nejprve se musí smazat „blinking memory“ kterážto určuje, které segmenty displeje budou blikat a které ne. Adresování této paměti je stejné jako adresování zobrazovací paměti a tam, kde je nula, tak segment nebliká a kde je jednička, tam bliká.

&H00 smazání celé blikací paměti.
&HE0 + n nastavení adresy "n"
&HCF zápis dat do paměti, zapisuje se první část adresy
&HE + n+1 nastaveni adresy "n+1"
&HCF zápis dat do paměti, zapisuje se druhá část adresy
&H1A povolení blikání

Celá sekvence se zapíše jako příkaz čili CD=1. Poslední hodnota &H1A zapne pomalé blikání, hodnota &H1B blikání rychlé. Rychlosti blikání jsou závislé na frekvenci oscilátoru řadiče a mohou se lišit. Pomalé blikání je přibližně jednou za sekundu, rychlé asi třikrát za sekundu.

Blikání celého displeje

Pokud chci rozblikat celý displej, musím nastavit celou „blinking memory“ na hodnotu logické jedničky. Postup je stejný jako u blikání jednoho znaku, jen se musí zopakovat pro každou pozici.

V čem tedy všichni autoři, které jsem našla, dělali chybu? Pouze zapnuli blikání ale nechali v „blinking memory“ hodnoty, které se tam objeví po startu. Ty jsou víceméně náhodné a tak docházejí k divným výsledkům. Já jsem k nim docházela také, než jsem na to přišla. Paměť je potřeba vymazat a zapsat do ní správná data.

Adresování displeje

Jsou dvě možnosti, jak se dá komunikovat se zobrazovací pamětí. Jednak se dá adresovat každý segment jednotlivě, pak je postup obdobný, jako u blinking memory, nebo se pomocí patřičného příkazu vypne dekodér znaků a pošlou se přímo zobrazovaná data. Drobná komplikace, no někdy pořádná, je to, že nultá pozice je úplně vpravo a hodnoty stoupají směrem doleva.

Zobrazování dat bez pomocí dekodéru

Dekodér vypne příkaz &H14 přičemž je vhodné poslat ještě příkazy &H18 a &H20, které vypnou blikání a smažou paměti. Data pro displej se pak pošlou v pořadí „degf.cba“ přičemž logická jednička znamená svítící segment a logická nula segment zhasnutý.

Zobrazení desetinné tečky

Desetinou tečku je možné zobrazit buď pomocí Zobrazení bez dekodéru a nebo touto sekvencí příkazů:

&H14 vypnutí dekodéru
&HE0 + X nastavení pozice tečky, nula je úplně vpravo, další jsou 2 4 6 8 ...
&HB8 hodnota zapínající desetinnou tečku
&H15 zapnutí dekodéru

Ve skutečnosti se provede to, že první příkaz vypne dekodér, druhý příkaz nastaví ukazatel paměti na pozici a třetí příkaz provede OR mezi hodnotou v paměti a konstantou „08“ čímž se rozsvítí tečka. Poslední příkaz opět zapne dekodér.

Obr. 2 – nastavení programátoru a hlavně pojistek.

Zkušební program jsem napsala ve svém oblíbeném Bascomu a můžete si jej stáhnout a prostudovat. Program je docela dlouhý a nevleze se do procesoru ATtiny2313 je potřeba použít ATtint4313 nebo nějaký ještě větší. Je úmyslně psaný velmi nehospodárně a tak, aby bylo všechno jasně vidět. Také uložení znaků do RAM není moc dobré, ale je to jednoduché a pro ukázku velmi názorné. V programu pro jiný účel bych data uložila jinam. Třeba do EEPROM nebo FLASH.

Obr. 3 – Běžící demonstrační program (a chaoz u počítače )

Řadič toho umí ještě více, umí dělat různé operace s pamětí a podobně, ale to už je celkem jasné z dokumentace. Katalogový list si můžete stáhnout přímo z Žirafovin.

Pokud tato část zápisníku někomu pomůže, budu ráda. A pokud ne, tak nevadí, já jsem se toho při psaní programu naučila dost. A potvrdila se mi přitom i jedna stará moudrost, že když se člověk moc snaží na něco přijít, tak se to často nedaří. Pak stačí naťuknutí od jiného člověka a rázem se dostaví nápad a je to tu. Toto naťuknutí přišlo od lidí z http://www.mikrozone.sk a to naťuknutí bylo vlastně úplně špatně, ale mne to prostě pak došlo a výsledek je fungující program.