**Watchdog**

datasheet, strana 325, kap. 27.5

U nejdokonalejšího operačního systému na světě se vám jistě mnohokrát stalo, že na obrazovce vyskočilo hlášení „Program neodpovídá“. Něco jste udělali a vyskočila další hláška: „Chvíli počkejte. Možná budete moci pokračovat“. Počkali jste drahnou dobu a zjistili jste, že nebudete moci pokračovat. Stiskneme nějakou klávesu a objeví se hláška: „Došlo k výjimce 0x5687er12sdfd5 na adrese 0xdf147ba4893bbfd45 “. Něco dalšího jste udělali, obrazovka zmodrala a pak už jste nemohli udělat vůbec nic. V takovém případě je nutno stisknout čudlík RESET na počítači.

U dobře napsaného programu samozřejmě takováto situace nenastane. U programů, které píšeme do našeho PICa, by také neměla nastávat. Až ale budete psát „doopravdický“ program, který využívá mnoho desítek přerušení, mnoho cyklů, podmínek atd. atd. atd., takováto situace nastat může. Zejména proto, že není možné odzkoušet všechny kombinace, které kdy přicházejí v úvahu, navíc v případech, že se mohou objevit třeba jednou za měsíc. I v programu napsaném zkušeným programátorem pro PICa je tedy možné, že program „zabloudí“ ( protože programátor někde udělal chybu, na kterou během testování nepřišel ). Je tedy nutno PICovi zmáčknout tlačítko RESET. No jo ! Ale PIC je někde hluboko ve spleti jiných obvodů, a uživatel zařízení pravidelně vůbec neví, že někde nějakého PICa má.

Nu a tehdy přichází na pomoc watchdog.

Jeho role je jednoduchá: čas od času udělá RESET.

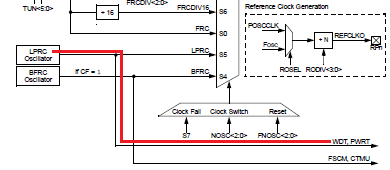
Ovšem my nechceme, aby náš program byl přerušován RESETem !

To uděláme následovně: program je vždy komponován tak, že má úvodní část, kterou udělá jednou, a potom dokolečka cyklí ve věčném cyklu. Z něj se občas odskakuje do přerušení, volají se podprogramy atd. atd., ale program pravidelně prochází hlavním cyklem.

Sem umístíme instrukci **clrwdt**

Ta vynuluje watchdog ( watchdog je čítač ), a tím zabrání RESETu procesoru. Doufáme, že program, pokud zabloudí, tak zabloudí někde mimo hlavní cyklus. Potom samozřejmě po nějaké době watchdog udělá RESET. Pokud program napíšete tak blbě, že zabloudí v cyklu, kde se periodicky opakuje instrukce clrwdt , pak není pomoci.

Jako každý čítač i watchdog potřebuje hodinový signál. Ten se získává z oscilátoru LPRC (kap. 9.0 oscilator, strana 123)



LPRC kmitá na kmitočtu 32kH. To je tedy vstupní kmitočet WATCHDIGu.

Watchdog sestává ze dvou děliček, prescaller a postscaller. ( strana 325, Figure 27-2 ). Dole je ještě "WDT window select" , tím se zatím nezabývejte.

Prescaller dělí buď 32 nebo 128 (EX-OR). Postscaller dělí 1, 2, 4, 8, 16, …… , po mocninách dvou až do 32768 ( postscaller je 15-tibitový čítač )

Nejkratší čas, po kterém nastane reset, je tehdy, když prescaller dělí 32 a postscaller 1

32kHz : 32 : 1 = 1kHz, tedy čas je 1ms

Nejdelší čas, po kterém nastane reset, je tehdy, když prescaller dělí 128 a postscaller 32768

32kHz : 128 : 32768 = 0,00762939453125 Hz, tedy čas je 131 s

Děličky WATCHDOGu nastavujeme pomocí konfiguračních slov (fuses). Ta najdeme v include file v C:\Program Files (x86)\Microchip\xc16\v1.33\support\dsPIC33E\inc zhruba od řádku 5000 – hledejte FWDT (0x57a0)

Watchdog Timer Postscaler Bits:

; WDTPOST\_PS1 1:1

; WDTPOST\_PS16384 1:16,384

; WDTPOST\_PS32768 1:32,768

;

; Watchdog Timer Prescaler Bit:

; WDTPRE\_PR32 1:32

; WDTPRE\_PR128 1:128

;

; Watchdog Timer Enable Bits:

; FWDTEN\_OFF WDT and SWDTEN Disabled

; FWDTEN\_ON WDT Enabled

Úmyslně zde neuvádíme všechno, podívejte se do include file.

Ve file wdt.s máte příklad nastavení WATCHDOGu.

Podíváme se na řádek config \_\_FWDT, za tím jsou hodnoty, na které chceme WATCHDOG nastavit. Identifikátory znamenají čísla, jejich hodnotu najdeme v include file po tím, v sekci začínající .equiv

S čísly se dělá AND, úplně stejně, jak jsme se to učili u logických instrukcí ( operátor & )



Takže do konfiguračního registru se uloží jedno jediné číslo ( račte si ho sami určit ). Taktéž si vypočtěte, po jakém čase watchdog udělá reset.