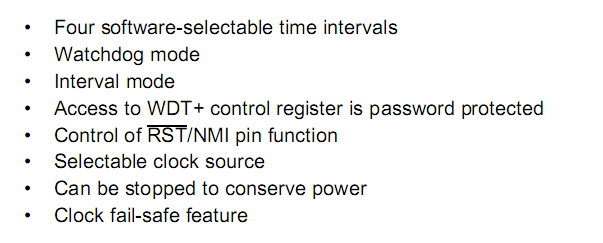
**MSP430 Uyg.19 – WatchDog Timer Uygulaması\_1**

Ferudun GÖKCEGÖZ, 07 Ağustos 2011, Pazar

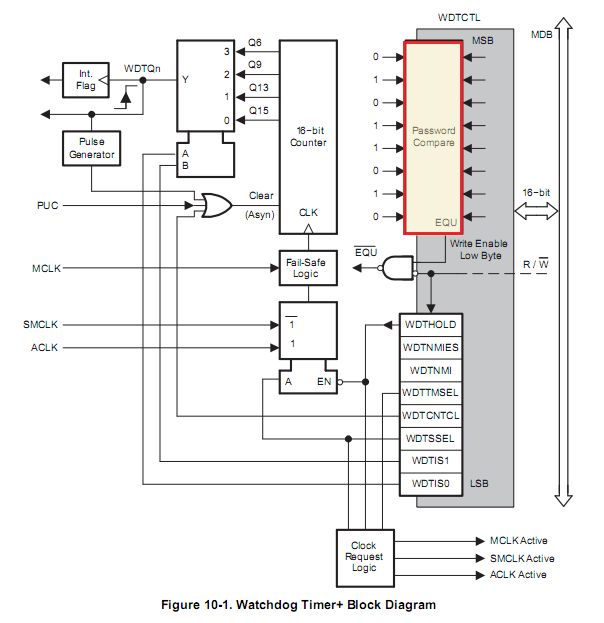
Merhaba arkadaşlar yeni bir dersimizle yine sizlerle birlikteyiz. Bu yazımızda WatchDog Timer ile ilgili bir uygulama yapacağız. WatchDog Timer mikrodenetleyicilerin çoğunda bulunan, mikrodenetleyici yazılımsal veya donanımsal bir problemle karşılaşıp kilitlenme durumu oluştuğunda mikrodenetleyiciye reset attrıran veya ilgili vektör adresine dallandıran bir çevre birimidir. WDT Timer içerisinde bulunan bir sayıcı belirli periyotlarla temizlenip sıfırlanmadığı taktirde, kilitlenme durumu oluştuğu anlaşılır.

Bu dersimizde uygulamamızı proteus üzerinde değil LaunchPad üzerinde yapacağız. İlk uygulamamız WDT periyodik olarak resetlenmediği durumda mikrodenetleyicinin resetlenmesiyle ilgili olacaktır. İkinci WDT uygulamamızda ise WDT ı sıfırladığımızda mikrodenetleyicinin resetlenmediğine ilişkin bir uygulamamız olacaktır. İsterseniz ilk olarak MSP430 lardaki WDT birimini tanıyalım…

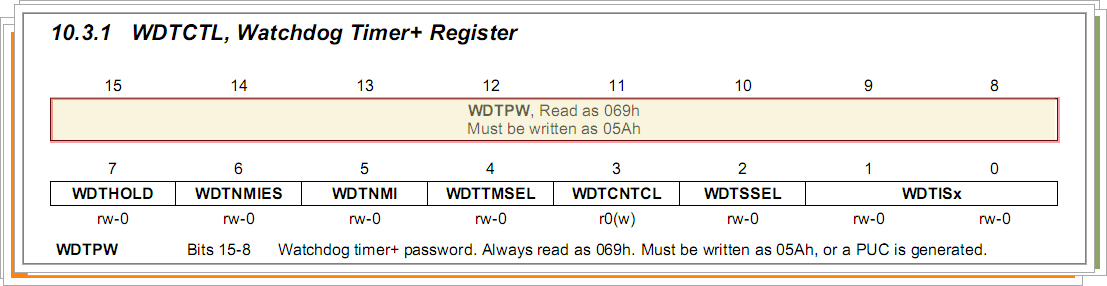
MSP430 lardaki WDT biriminin genel özelliklerini aşağıdaki gibi sıralayabiliriz.



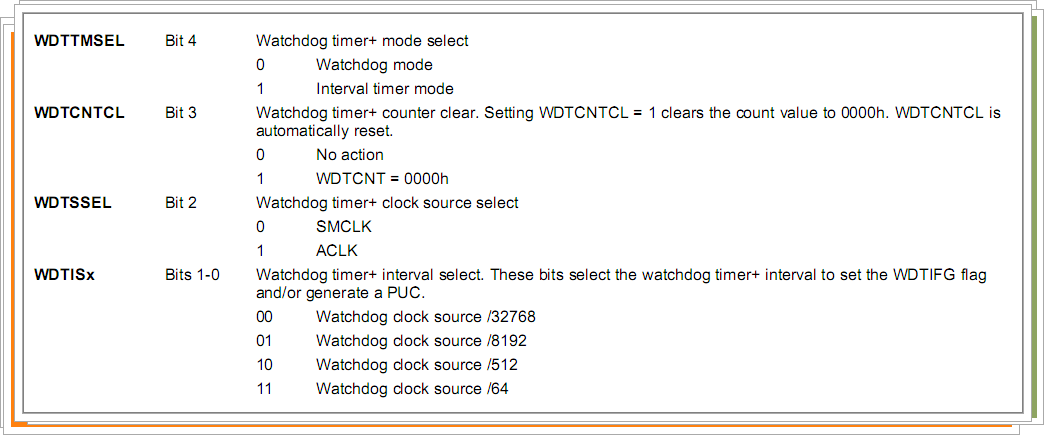
Aşağıda watchdog timer ile ilgili blok diyagramı incelersek, WDT biriminde bir password kaydedicisinin olduğunu görürüz.



Bu password kaydedicisi WDTCTL (WDT Control Register) içerisindeki son 8 biti (HighByte) oluşturmaktadır. WDTCTL kaydedicisine bir değer yüklenirken, bu kaydedicinin değeri ile birlikte yüklenmek zorundadır. Aşağıdan da görüldüğü üzere bu password kaydedicisi okunurken, 0×59 değeri olarak okunur, yazılırken ise 0x5A değeri ile yazılmaldır.



Bu kaydedicideki diğer önemli bitlere göz atarsak..

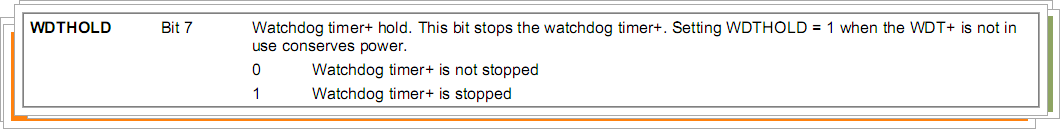


İlk olarak WDTTMSEL bitinden bahsedelim. Bu bit ile WDT birimi için mode seçimi yapılır. Watchdog mode seçilir ise, wdt periyodu dolduğunda eğer hala wdt birimi resetlenmemişse, mikrodenetleyiciye reset attırılır. Bu bit, Lojik 1 yapılıp interval mod seçildiğinde ise, wdt periyodu dolduğunda mikrodenetleyici reset yerine ilgili kesme vektörüne dallandırılır.

Bir sonraki WDTCNTL biti ise, wdt birimini temizlemek için kullanılır. Eğer bu bit Lojik0 ise hiçbir işlem yapılmaz. Lojik1 çekildiği anda (WDT  password ile birlikte) WDT birimi resetlenir.

WDTSSEL biti ise WDT birimi için gereken saat darbelerinin kaynağını seçmek için kullanılır. Bu bit Lojik0 ise clock kaynağı SMCLK, Lojik1 ise ACLK dır.

Son olarak WDTISx bitleri ise WDT birimine gelen clock darbeleri için bölücü seçme bitleridir. Bu bitler 00 ise 32768, 01 ise 8192, 10 ise 512, 11 ise 64 e bölünür.



Şundan da bahsetmeden geçmek istemiyorum. Bundan önceki uygulamalarımızın hepsinde main fonksiyonumuzun başında;

|  |  |
| --- | --- |
| **[Source code](http://www.mcu-turkey.com/msp430-uyg-19/" \l "codesyntax_1" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-19/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-19/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

WDTCTL = WDTPW + WDTHOLD;

satırını kullandık. Yukarıda görüldüğü gibi bu satırdaki kod ile, WDT birimini durdumuş olduk. Yani yazılım içerisinde WDT birimini resetlemeye ihtiyaç duymamak için WDT birimini kapattık. Tabi bunu yaparken, WDTCTL register ına bir değer yüklerken Password byte ı ile birlikte bir değer yükledik.

IAR derleyicisi WDT birimi için gerekli ayarlamaları bizim için seçtiğimiz mikrodenetleyicinin kütüphane dosyasına eklemiş. IAR derleyicisinin sol kısmındaki workspace kısmından, main.c dosya ağacı açılıp kullandığımız mikrodenetleyiciye ait kütüphane dosyasını açarsak oradan WDT ile ilgili tanımlamaları görebiliriz.

|  |  |
| --- | --- |
| **[Source code](http://www.mcu-turkey.com/msp430-uyg-19/" \l "codesyntax_2" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-19/#codesyntax_2)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-19/#codesyntax_2)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

*#define \_\_MSP430\_HAS\_WDT\_\_ /\* Definition to show that Module is available \*/*

*#define WDTPW (0x5A00u)*

*/\* WDT is clocked by fSMCLK (assumed 1MHz) \*/*

*#define WDT\_MDLY\_32 (WDTPW+WDTTMSEL+WDTCNTCL) /\* 32ms interval (default) \*/*

*#define WDT\_MDLY\_8 (WDTPW+WDTTMSEL+WDTCNTCL+WDTIS0) /\* 8ms " \*/*

*#define WDT\_MDLY\_0\_5 (WDTPW+WDTTMSEL+WDTCNTCL+WDTIS1) /\* 0.5ms " \*/*

*#define WDT\_MDLY\_0\_064 (WDTPW+WDTTMSEL+WDTCNTCL+WDTIS1+WDTIS0) /\* 0.064ms " \*/*

*/\* WDT is clocked by fACLK (assumed 32KHz) \*/*

*#define WDT\_ADLY\_1000 (WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL) /\* 1000ms " \*/*

*#define WDT\_ADLY\_250 (WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL+WDTIS0) /\* 250ms " \*/*

*#define WDT\_ADLY\_16 (WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL+WDTIS1) /\* 16ms " \*/*

*#define WDT\_ADLY\_1\_9 (WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL+WDTIS1+WDTIS0) /\* 1.9ms " \*/*

*/\* WDT is clocked by fSMCLK (assumed 1MHz) \*/*

*#define WDT\_MRST\_32 (WDTPW+WDTCNTCL) /\* 32ms interval (default) \*/*

*#define WDT\_MRST\_8 (WDTPW+WDTCNTCL+WDTIS0) /\* 8ms " \*/*

*#define WDT\_MRST\_0\_5 (WDTPW+WDTCNTCL+WDTIS1) /\* 0.5ms " \*/*

*#define WDT\_MRST\_0\_064 (WDTPW+WDTCNTCL+WDTIS1+WDTIS0) /\* 0.064ms " \*/*

*/\* WDT is clocked by fACLK (assumed 32KHz) \*/*

*#define WDT\_ARST\_1000 (WDTPW+WDTCNTCL+WDTSSEL) /\* 1000ms " \*/*

*#define WDT\_ARST\_250 (WDTPW+WDTCNTCL+WDTSSEL+WDTIS0) /\* 250ms " \*/*

*#define WDT\_ARST\_16 (WDTPW+WDTCNTCL+WDTSSEL+WDTIS1) /\* 16ms " \*/*

*#define WDT\_ARST\_1\_9 (WDTPW+WDTCNTCL+WDTSSEL+WDTIS1+WDTIS0) /\* 1.9ms " \*/*

Gelelim bizim uygulama yazılımıza…

|  |  |
| --- | --- |
| **[Source code](http://www.mcu-turkey.com/msp430-uyg-19/" \l "codesyntax_3" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-19/#codesyntax_3)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-19/#codesyntax_3)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

*#include "io430.h"*

*#include "in430.h"*

*#define LED0 P1OUT\_bit.P0*

void delay(void)

{

unsigned short i=20000;

while(i--);

}

void main(void)

{

WDTCTL = WDT\_ARST\_1000;

DCOCTL=CALDCO\_1MHZ;

BCSCTL1=CALBC1\_1MHZ;

P1OUT = 0x00;

P1DIR = 0x01;

LED0 = 0;

delay();

LED0 = 1;

for(;;);

}

Yazılımımız oldukça sade. Kısaca şöyle anlatabiliriz. WDT birimi clock kaynağı ACLK olarak seçilmiş ve WDT periyodu 1 sn olarak ayarlanmış. Bu arada ACLK seçili olduğu, launchpadlerinize kristali lehimlemedi iseniz uygulamayı çalıştıramazsınız. Devam edecek olursak, P1 portunun P1.0 pini çıkış olarak tanımlanmış. İlk olarak Lojik0 a çekilip ardından da bir süre gecikme verilip, Lojik1  e çekilmiş. Biz bu uygulamamızda WDT birimini kullanıma açtık, fakat periyodik olarak resetlemediğimiz için WDT süresi(1sn) dolduğuna mikrodenetleyici resetlenecektir. Bu nedenle yazılım baştan yürütülecektir. Bu resetlenme yüzünden de, sonsuz döngüden önceki LED değişkenini ilk önce Lojik0 a ardından bir süre gecikme verilip Lojik1 çekilme kısmı, her resetlenme gerçekleşecektir. Bizde bunu ledin sönüp tekrar yanması olarak izleyeceğiz

Geldik bir dersimizin daha sonuna. Umarım faydalı olabilmişimdir. Bir sonraki dersimizde de WDT birimi ile uygulama yapıp, daha sonra da PWM uygulamalarına geçeceğiz. Görüşmek üzere. Şimdilik Hoşçakalın…

***Ferudun GÖKCEGÖZ***

***fgokcegoz@yahoo.com***