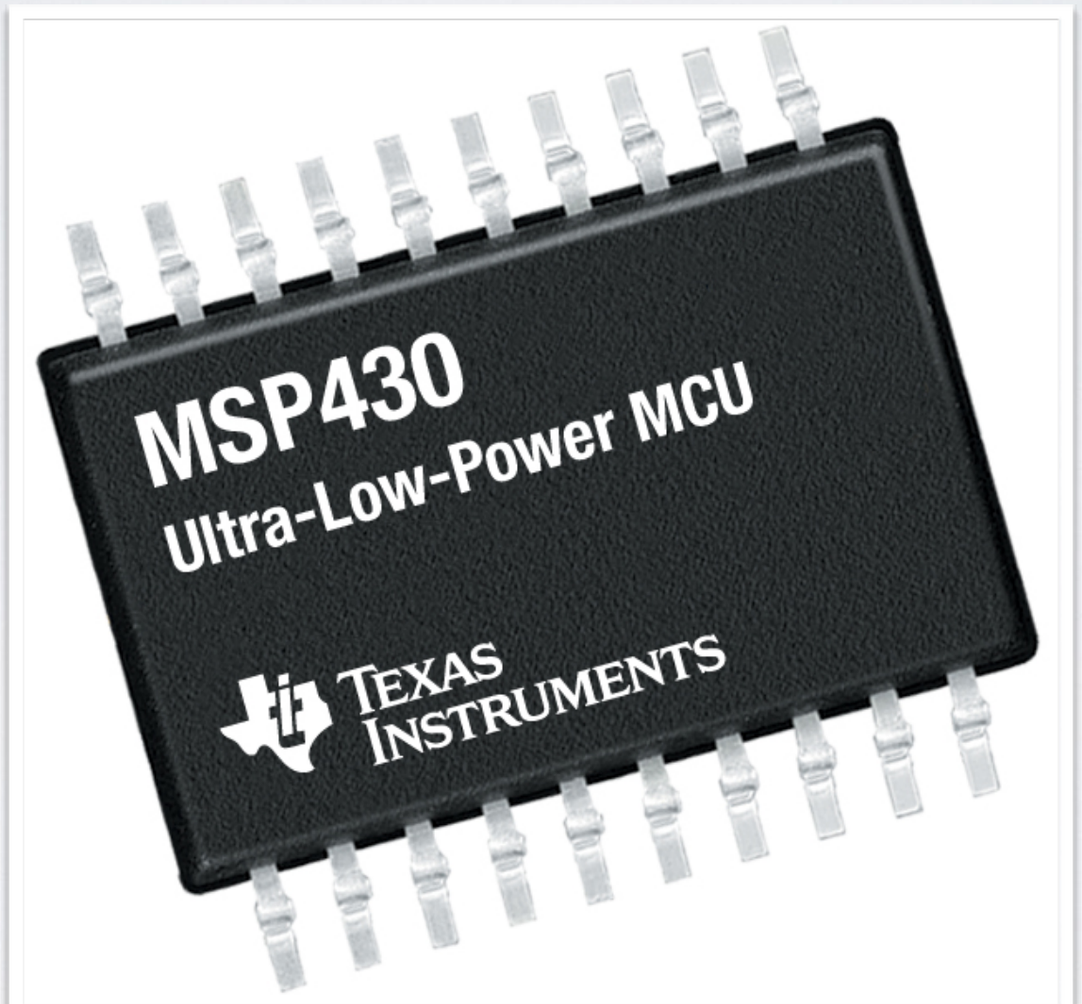
MICROPROCESSADORES E MICROCONTROLADORES



WATCHDOG TIMER

Contador que reinicia o sistema (reset) quando termina sua contagem. O software deve zera-lo periodicamente para evitar o reset.

Protege o sistema contra falhas de software (loops infinitos indesejados etc.)

WATCHDOG TIMER

O Watchdog Timer é controlado pelo registrador WDTCTL, que só pode ser mudado quando se escreve a senha WDTPW=0x5A no seu byte superior. Caso contrário, ocorre o reset, o que protege contra alterações acidentais do WDTCTL.

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r0(w)	rw–(0)	rw–(0)	rw–(0)

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0(w)	rw-(0)	rw-(0)	rw-(0)

A maioria desses bits começa zerado, mas eles não são afetados por um reset pelo Watchdog Timer. Se WDTSSEL= 1, por exemplo, quer dizer que houve um reset pelo Watchdog Timer.

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r0(w)	rw–(0)	rw–(0)	rw–(0)

A exceção é WDTCNTCL. Ler este bit sempre retorna o valor 0, mas o valor 1 pode ser escrito.

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r0(w)	rw–(0)	rw–(0)	rw–(0)

WDTHOLD=1 desliga o watchdog timer.

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNM	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0(w)	rw-(0)	rw-(0)	rw-(0)

Escolhe o modo de funcionamento: watchdog timer (0) ou contador normal, sem causar reset (1).

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0(w)	rw-(0)	rw-(0)	rw-(0)

WDTCNTCL=1 zera o contador do watchdog timer (WDTCNT), que não é visível ao usuário.

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r0(w)	rw–(0)	rw–(0)	rw–(0)

Escolhe o sinal de clock para o watchdog timer: SMCLK (0) ou ACLK (1).

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSE	WDTISx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0(w)	rw-(0)	rw-(0)	rw-(0)

Escolhe o limite de contagens para o watchdog timer: 32768 (00), 8192 (01), 512 (10) ou 64 (11).

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r0(w)	rw–(0)	rw–(0)	rw–(0)

Controla a função do pino RST/NMI: reset (0) ou interrupção não-mascarável (1).

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0(w)	rw-(0)	rw-(0)	rw-(0)

Escolhe a borda para a interrupção não-mascarável: subida (0) ou descida (1).

WATCHDOG TIMER

Byte inferior de WDTCTL:

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	r0(w)	rw–(0)	rw–(0)	rw–(0)

Ao completar a contagem, o watchdog timer seta a flag WDTIFG no registrador IFG1. Ela não é zerada após o reset.

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6

#define BTN   BIT3

void main(void){

   WDTCTL = WDTPW | WDTCNTCL;

   P1DIR = LED1+LED2;   P1OUT = P1REN = BTN;

if(IFG1 & WDTIFG) P1OUT |= LED2;

for(;;){

   if(P1IN & BTN) {

      P1OUT &= ~LED1;

      WDTCTL = WDTPW | WDTCNTCL;

   } else P1OUT |= LED1;

}}
```

LEDs e botão do MSP430 Launchpad.

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6

#define BTN   BIT3

void main(void){

  WDTCTL = WDTPW | WDTCNTCL;

  P1DIR = LED1+LED2;   P1OUT = P1REN = BTN;

  if(IFG1 & WDTIFG) P1OUT |= LED2;

  for(;;){

    if(P1IN & BTN) {

      P1OUT &= ~LED1;

      WDTCTL = WDTPW | WDTCNTCL;

    } else P1OUT |= LED1;

}}
```

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6

#define BTN   BIT3

void main(void){
  WDTCTL = WDTPW | WDTCNTCL;
   P1DIR = LED1+LED2;   P1OUT = P1REN
   if(IFG1 & WDTIFG) P1OUT |= LED2;
for(;;){
   if(P1IN & BTN) {
      P1OUT &= ~LED1;
      WDTCTL = WDTPW | WDTCNTCL;
   } else P1OUT |= LED1;
}}
```

Watchdog timer zerado, obtido do SMCLK, e causando reset após 32768 contagens.

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6
```

Configuração dos LEDs (apagados inicialmente) e do botão (com resistor de pull-up).

```
P1DIR = LED1+LED2;   P1OUT = P1REN = BTN;

if(IFG1 & WDTIFG) P1OUT |= LED2;

for(;;){

   if(P1IN & BTN) {

      P1OUT &= ~LED1;

      WDTCTL = WDTPW | WDTCNTCL;

   } else P1OUT |= LED1;

}}
```

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6

#define BTN   BIT3

void main(void){

   WDTCTL = WDTPW | WDTCNTCL;

   P1DIR = LED1+LED2;   P1OUT = P1R

   if(IFG1 & WDTIFG) P1OUT |= LED2;

   for(;;){

      if(P1IN & BTN) {

         P1OUT &= ~LED1;

         WDTCTL = WDTPW | WDTCNTCL;

      } else P1OUT |= LED1;

   }}
```

LED2 apresenta o estado da flag de interrupção do watchdog timer.

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6

#define BTN  BIT3

void main(void){
```

Enquanto não se pressiona o botão, o LED1 fica apagado, e o contador do watchdog timer é zerado.

```
  if(P1IN & BTN) {
     P1OUT &= ~LED1;
     WDTCTL = WDTPW | WDTCNTCL;
   } else P1OUT |= LED1;
}}
```

```
#include <msp430g2553.h>

#define LED1  BIT0

#define LED2 BIT6

#define BTN   BIT3

void main(void){

   WDTCTL = WDTPW | WDTCNTCL;

   P1DIR = LED1+LED2;   P1OUT = P1REN = BTN;

   } else P1OUT |= LED1;

}}
```

Se o botão for pressionado, o LED1 é aceso, e se continuar pressionado até contador do watchdog timer atingir o limite, o sistema reinicia, e WDTIFG=1.