ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Systèmes embarqués

Les timers du MSP430

Pierre-Yves Rochat

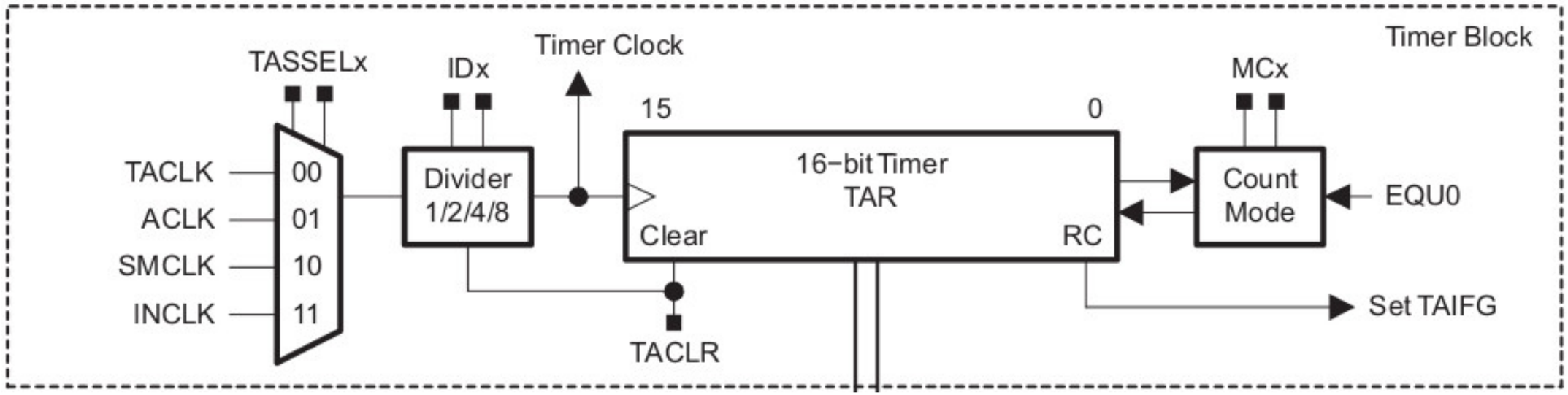
Introduction aux interruptions

Pierre-Yves Rochat

Les timers

- Gestion du temps
- Timers, prédivision, logique de gestion et registres de comparaison
- Mise en œuvre : exemple du MSP430
- Interruptions des timers

Le timer A du MSP430



Le registre de contrôle

12.3.1 TACTL, Timer_A Control Register

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Unused Bits 15-10 Unused

TASSELx Bits 9-8 Timer_A clock source select

- 00 TACLK
- 01 ACLK
- 10 SMCLK
- 11 INCLK (INCLK is device-specific and is often assigned to the inverted TBCLK) (see the device-specific data sheet)

IDx Bits 7-6 Input divider. These bits select the divider for the input clock.

- 00 /1
- 01 /2
- 10 /4
- 11 /8

Le registre de contrôle

12.3.1 TACTL, Timer_A Control Register

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

MCx Bits 5-4 Mode control. Setting MCx = 00h when Timer_A is not in use conserves power.

- 00 Stop mode: the timer is halted.
- 01 Up mode: the timer counts up to TACCR0.
- 10 Continuous mode: the timer counts up to 0FFFFh.
- 11 Up/down mode: the timer counts up to TACCR0 then down to 0000h.

Unused Bit 3 Unused

TACLR Bit 2 Timer_A clear. Setting this bit resets TAR, the clock divider, and the count direction. The TACLR bit is automatically reset and is always read as zero.

TAIE Bit 1 Timer_A interrupt enable. This bit enables the TAIFG interrupt request.

- 0 Interrupt disabled
- 1 Interrupt enabled

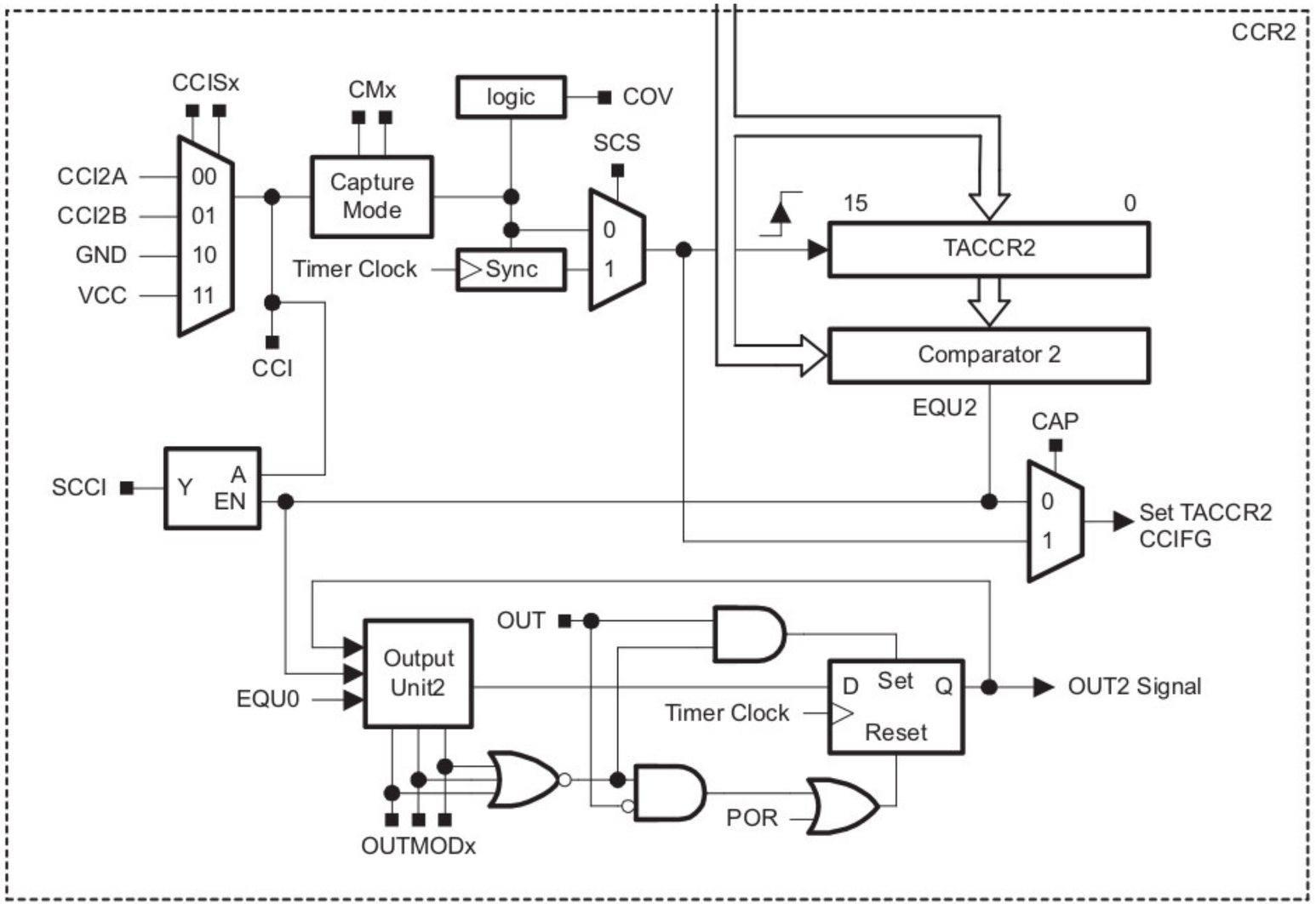
TAIFG Bit 0 Timer_A interrupt flag

- 0 No interrupt pending
- 1 Interrupt pending

Programme de mise en oeuvre

```
1  int main() {
2    WDTCTL = WDTPW + WDTHOLD; // Watchdog hors service
3    BCSCTL1 = CALBC1_1MHZ;
4    DCOCTL = CALDCO_1MHZ;      // Fréquence CPU
5    P1DIR |= (1<<0); // P1.0 en sortie pour la LED
6    TACTL0 = TASSEL_2 + ID_3 + MC_2;
7    while (1) {                // Boucle infinie
8      if (TACTL0 & TAIFG) {
9        TACTL0 &= ~TAIFG;
10       P1OUT ^= (1<<0);       // Inversion LED
11     }
12   }
13 }
```

Les registres de comparaison



Les registres de comparaison

12.3.4 TACCTLx, Capture/Compare Control Register

15	14	13	12	11	10	9	8
CMx		**CCISx**		**SCS**	**SCCI**	**Unused**	**CAP**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	r0	rw-(0)
7	6	5	4	3	2	1	0
OUTMODx			**CCIE**	**CCI**	**OUT**	**COV**	**CCIFG**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

CCIE Bit 4 Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag.
- 0 Interrupt disabled
- 1 Interrupt enabled

CCI Bit 3 Capture/compare input. The selected input signal can be read by this bit.

OUT Bit 2 Output. For output mode 0, this bit directly controls the state of the output.
- 0 Output low
- 1 Output high

COV Bit 1 Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software.
- 0 No capture overflow occurred
- 1 Capture overflow occurred

CCIFG Bit 0 Capture/compare interrupt flag
- 0 No interrupt pending
- 1 Interrupt pending

Les registres de comparaison

```
14 int main() {
15   ...
16   TACCR0 = 62500; // 62500 * 8 us = 500 ms
17   while (1) {     // Boucle infinie
18     if (TACCTL0 & CCIFG) {
19       TACCTL0 &= ~CCIFG;
20       TACCR0 += 62500;
21       P1OUT ^= (1<<0); // Inversion LED
22     }
23   }
24 }
```

L'interruptions de dépassement de capacité

```
14 int main() {
15   ...
16   TACTL |= TAIE; // Interruption de l'overflow
17   _BIS_SR (GIE); // Autorisation générale des interruptions
18   while (1) {    // Boucle infinie vide
19   }
20 }
21 // Timer_A1 Interrupt Vector (TAIV) handler
22 #pragma vector=TIMER0_A1_VECTOR
23 __interrupt void Timer_A1 (void) {
24   switch (TAIV) {    // discrimination des sources d'interruption
25   case  2:           // CCR1 : not used
26     break;
27   case  4:           // CCR2 : not used
28     break;
29   case 10:           // Overflow
30     P1OUT ^= (1<<0); // Inversion LED
31     break;
32   }
33 }
```

L'interruption de comparaison

```
14 int main() {
15   ...
16   TACCTL0 |= CCIE; // Interruption de la comparaison
17   _BIS_SR (GIE);   // Autorisation générale des interruptions
18   while (1) {      // Boucle infinie vide
19   }
20 }
21 #pragma vector=TIMER0_A0_VECTOR
22 __interrupt void Timer_A0 (void) {
23   CCR0 += 62500;
24   P1OUT ^= (1<<0); // Inversion LED
25 }
```

PWM par interruption

```
14 int main() {
15   ...
16   TACTL |= TAIE;    // Interruption de l'overflow
17   TACCTL0 |= CCIE; // Interruption de la comparaison
18   _BIS_SR (GIE);    // Autorisation générale des interruptions
19   while (1) {       // Boucle infinie vide
20   }
21 }
22 #pragma vector=TIMER0_A1_VECTOR
23 __interrupt void Timer_A1 (void) {
24   switch (TAIV) {     // discrimination des sources d'interruption
25   case  2:            // CCR1 : not used
26     break;
27   case  4:            // CCR2 : not used
28     break;
29   case 10:            // Overflow
30     P1OUT |= (1<<0); // Activer le signal au début du cycle
31     break;
32   }
33 }
34 #pragma vector=TIMER0_A0_VECTOR
35 __interrupt void Timer_A0 (void) {
36   P1OUT &=~(1<<0); // Désactiver le signal au moment donné
37 }                   // par le registre de comparaison
```

Les timers

- Gestion du temps
- Timers, prédivision, logique de gestion et registres de comparaison
- Mise en œuvre : exemple du MSP430
- Interruptions des timers