TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

Module 13

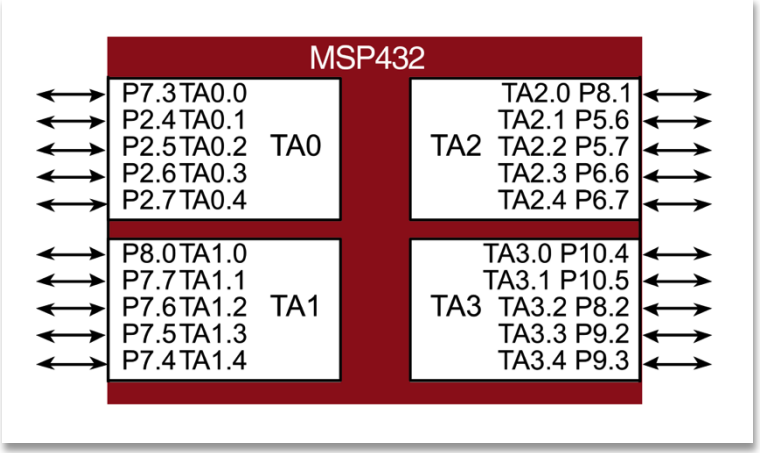
Lecture: Timers – Periodic Interrupts

Periodic interrupts using Timers

You will learn in this module

- Timer A on the TI MSP432
  - Clock input, prescale
  - Counter

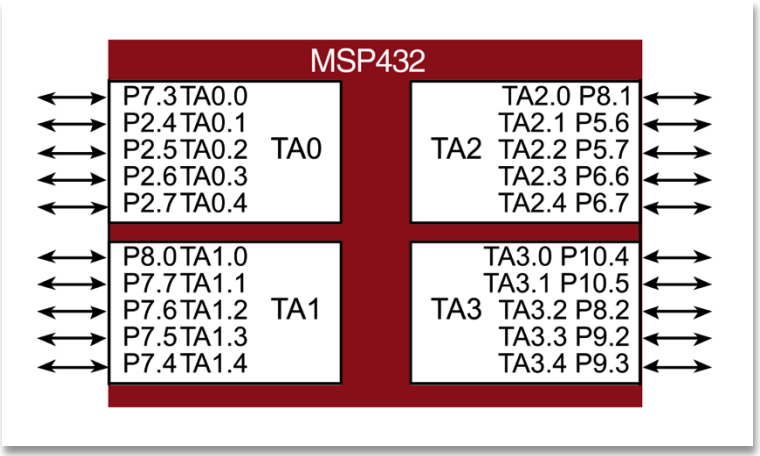
- Periodic interrupts
  - Additional threads
  - Priority



Timer - A for input and output

Timer A Features

- Pins
  - Input capture
  - Output compare
- Precision
  - 16-bits
- Resolution
  - Clock period
  - Prescale



SMCLK = 48MHz/4 = 12 MHz, 83.33ns

Timer - A registers

	15-10	9-8	7-6	5-4	3	2	1	0	Name
0.0000		TASSEL	ID	MC		TACLR	TAIE	TAIFG	TA0CTL

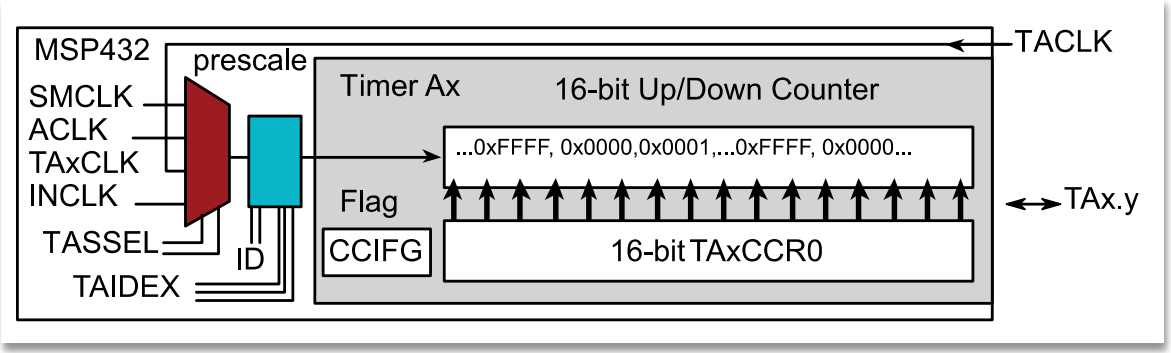
	15-14	13-12	11	10	9	8	7-5	4	3	2	1	0	
0.0002	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL0
0.0004	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL1
0.0006	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL2
0.0008	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL3
0.000A	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL4
0.000C	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL5
0.000E	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL6

	15–0	
0.0010	16-bit counter	TA0R
0.0012	16-bit Capture/Compare 0 Register	TA0CCR0
0.0014	16-bit Capture/Compare 1 Register	TA0CCR1
0.0016	16-bit Capture/Compare 2 Register	TA0CCR2
0.0018	16-bit Capture/Compare 3 Register	TA0CCR3
0.001A	16-bit Capture/Compare 4 Register	TA0CCR4
0.001C	16-bit Capture/Compare 5 Register	TA0CCR5
0.001E	16-bit Capture/Compare 6 Register	TA0CCR6

	15-3	2-0	
0.0020		TAIDEX	TA0EX0

	15-0	
0.002E	TAIV	TA0IV

Clock and prescale



TASSEL	Selected Clock
00	TAxCLK
01	ACLK
10	SMCLK
11	INCLK

ID	Prescale
00	/1
01	/2
10	/4
11	/8

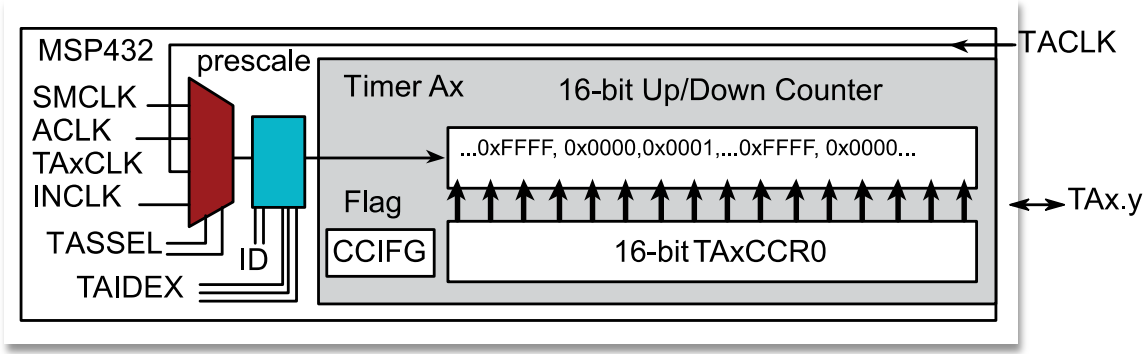
Resolution = $T * 2^{ID} * (\mathbf{TAIDEX}+1)$
Range = Precision * Resolution

Timer - A Modes

MC	Mode control
00	Stop
01	Up mode: Timer counts up to TAxCCR0
10	Continuous mode: Timer counts up to 0xFFFF
11	Up/down mode: Timer counts up to TAxCCR0 then down to 0x0000

OUTMOD	On match to TAxCCRy	On match to TAxCCR0
000	OUT bit value	
001	Set	
010	Toggle	Reset
011	Set	Reset
100	Toggle	
101	Reset	
110	Toggle	Set
111	Reset	Set

Clock and prescale



0) Halt the timer (MC=00),
1) Set the timer clock and prescale,
2) Set submodule 0 to compare, arm interrupt
3) Set the TAxCCR0 to the interrupt period minus 1
4) Set the priority in the correct NVIC Priority register
5) Enable the interrupt in the NVIC Interrupt Enable register
6) Reset the timer and start it in up mode
7) Enable interrupts (in the main program after all devices initialized)

Interrupt Vectors, numbers, names, and priority

Vector	Number	IRQ	ISR name	NVIC priority	Priority
0x0000002C	11	-5	SVC_Handler	SCB_SHPR2	31 – 29
0x00000038	14	-2	PendSV_Handler	SCB_SHPR3	23 – 21
0x0000003C	15	-1	SysTick_Handler	SCB_SHPR3	31 – 29
0x00000060	24	8	TA0_0_IRQHandler	NVIC_IPR2	7 – 5
0x00000064	25	9	TA0_N_IRQHandler	NVIC_IPR2	15 – 13
0x00000068	26	10	TA1_0_IRQHandler	NVIC_IPR2	23 – 21
0x0000006C	27	11	TA1_N_IRQHandler	NVIC_IPR2	31 – 29
0x00000070	28	12	TA2_0_IRQHandler	NVIC_IPR3	7 – 5
0x00000074	29	13	TA2_N_IRQHandler	NVIC_IPR3	15 – 13
0x00000078	30	14	TA3_0_IRQHandler	NVIC_IPR3	[illegible]
0x0000007C	31	15	TA3_N_IRQHandler	NVIC_IPR3	[illegible]
0x00000080	32	16	EUSCIA0_IRQHandler	NVIC_IPR4	[illegible]
0x00000084	33	17	EUSCIA1_IRQHandler	NVIC_IPR4	[illegible]
0x00000088	34	18	EUSCIA2_IRQHandler	NVIC_IPR4	[illegible]
0x0000008C	35	19	EUSCIA3_IRQHandler	NVIC_IPR4	[illegible]
0x00000090	36	20	EUSCIB0_IRQHandler	NVIC_IPR5	[illegible]
0x00000094	37	21	EUSCIB1_IRQHandler	NVIC_IPR5	15 – 13
0x00000098	38	22	EUSCIB2_IRQHandler	NVIC_IPR5	23 – 21
0x0000009C	39	23	EUSCIB3_IRQHandler	NVIC_IPR5	31 – 29
0x000000CC	51	35	PORT1_IRQHandler	NVIC_IPR8	31 – 29
0x000000D0	52	36	PORT2_IRQHandler	NVIC_IPR9	7 – 5
0x000000D4	53	37	PORT3_IRQHandler	NVIC_IPR9	15 – 13
0x000000D8	54	38	PORT4_IRQHandler	NVIC_IPR9	23 – 21
0x000000DC	55	39	PORT5_IRQHandler	NVIC_IPR9	31 – 29
0x000000E0	56	40	PORT6_IRQHandler	NVIC_IPR10	7 – 5

```
void TA2_0_IRQHandler(void){
  TIMER_A2->CCTL[0] &= ~0x0001; // ack
// body
}
```

Look for **interruptVectors[]**
in the file startup_msp432p401r_ccs.c

Periodic Interrupt

```
void (*TimerA2Task)(void);   // user function
void TimerA2_Init(void(*task)(void), uint16_t period){
  TimerA2Task = task;             // user function
  // bits9-8=10,       clock source to SMCLK
  // bits7-6=10,       input clock divider /4
  // bits5-4=00,       stop mode
  // bit2=0,           set this bit to clear
  // bit1=0,           no interrupt on timer
  TIMER_A2->CTL = 0x0280;
  // bits15-14=00,     no capture mode
  // bit8=0,           compare mode
  // bit4=1,           enable capture/compare interrupt on CCIFG
  // bit2=0,           output this value in output mode 0
  // bit0=0,           clear capture/compare interrupt pending
  TIMER_A2->CCTL[0] = 0x0010;
  TIMER_A2->CCR[0] = (period - 1);    // compare match value
  TIMER_A2->EX0 = 0x0005;     // configure for input clock divider /6
  NVIC->IP[3] = (NVIC->IP[3]&0xFFFFFF00)|0x00000040; // priority 2
  NVIC->ISER[0] = 0x00001000;   // enable interrupt 12 in NVIC
  TIMER_A2->CTL |= 0x0014;      // reset and start Timer A in up mode
}
```

Interrupts enabled in the main program after all devices initialized

$T * 2^{ID} * (\mathbf{TAIDEX}+1)(\mathbf{TAxCCR0}+1)$ = 2µs*period

```
void TA2_0_IRQHandler(void){
  TIMER_A2->CCTL[0] &= ~0x0001;
  *TimerA2Task();
}
```

NVIC->IP[2]= (NVIC->IP[2]&0xFF00FFFF)|0x00400000;
NVIC->ISER[0] = 0x00000400; // enable interrupt 10 in NVIC

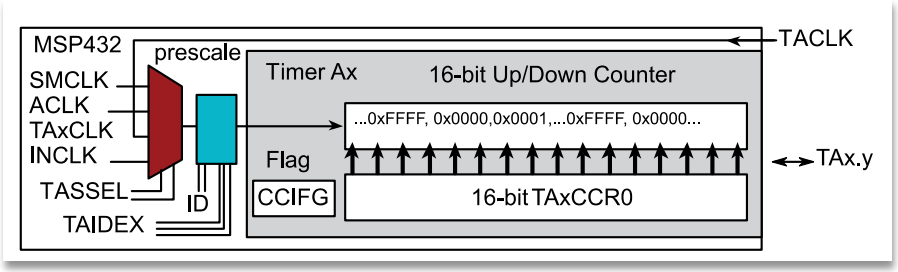
Summary

Timer overview

- Clock input
- Prescale
- Counter

Periodic interrupts

- Additional threads
- Priority




```
void TA0_0_IRQHandler(void){
  TIMER_A0->CCTL[0] &= ~0x0001;
// ack
// body
}
void TA1_0_IRQHandler(void){
  TIMER_A1->CCTL[0] &= ~0x0001;
// ack
// body
}
void TA2_0_IRQHandler(void){
  TIMER_A2->CCTL[0] &= ~0x0001;
// ack
// body
}
```

TimerA0.c

TimerA1.c

TimerA2.c

Module 13

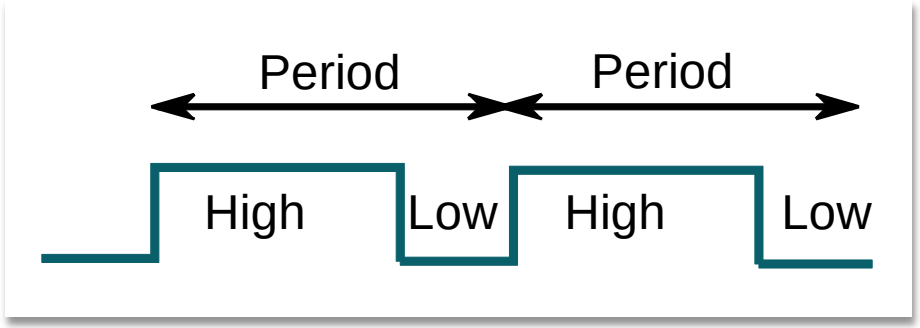
Lecture: Timers – Pulse Width Modulation

PWM using Timers

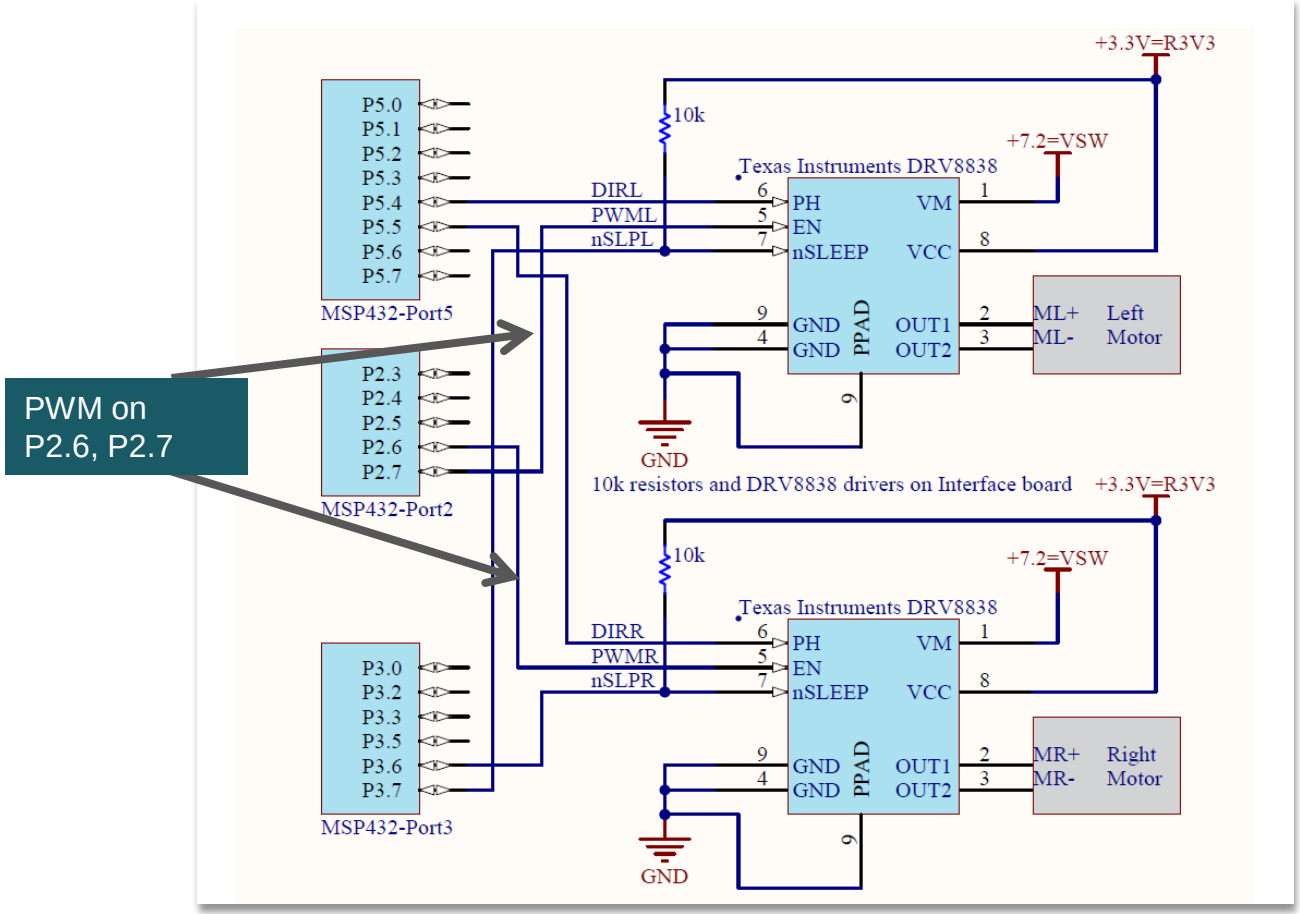
You will learn in this module

- Timer A on the TI MSP432
  - Clock input, prescale
  - Counter

- PWM output
  - Adjust power to motors
  - Two independent outputs



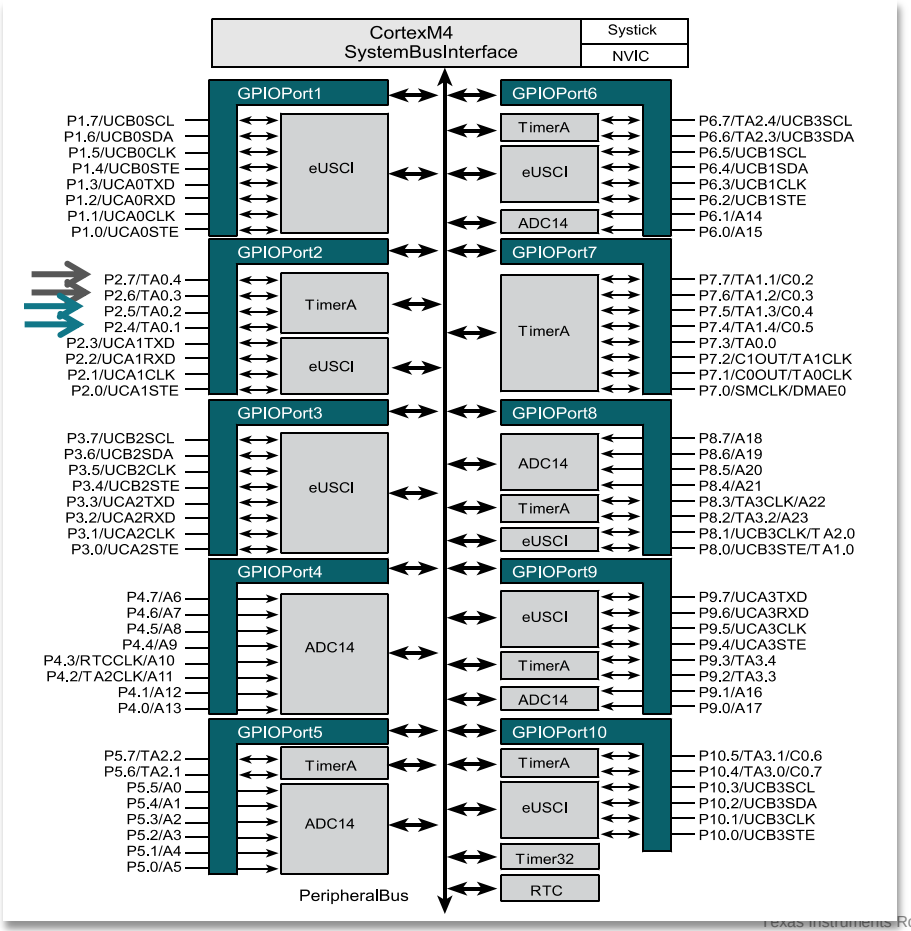
Interface circuit using TI-RSLK chassis board



MSP432 Input/Output

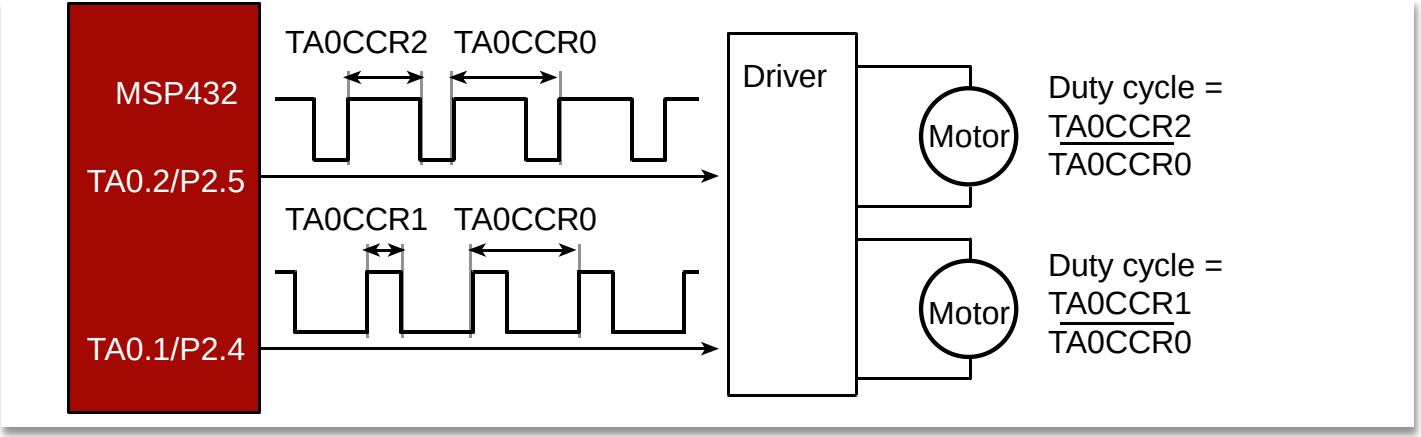
Lab code

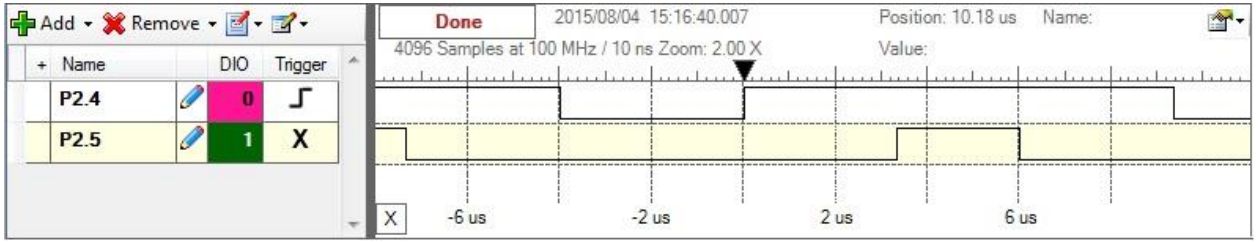
Starter code



Using Timer - A for Two PWM Outputs

You will convert it to
TA0.4/P2.7
TA0.3/P2.6



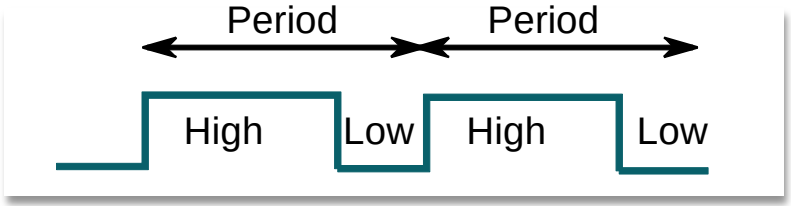


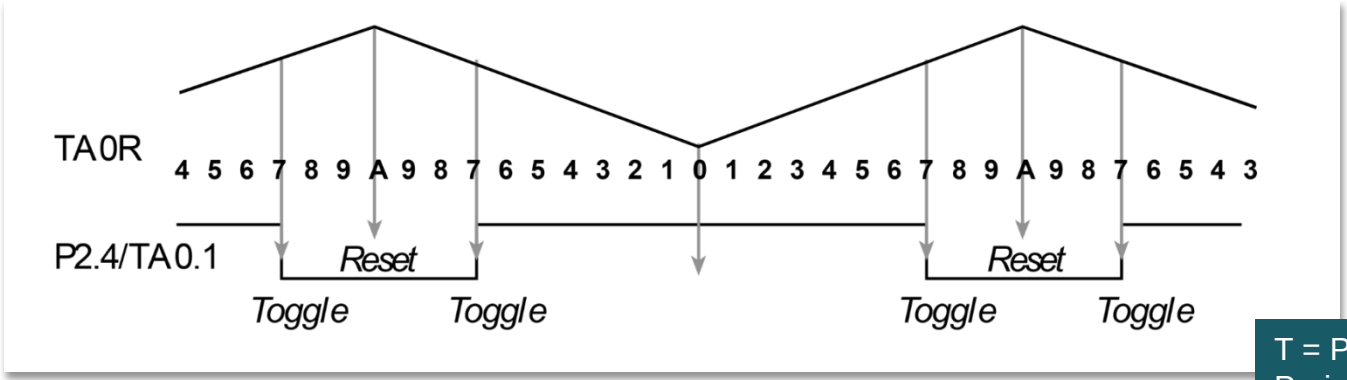
Using Timer - A for Two PWM Outputs

$$\text{Dutycycle} = \frac{\texttt{High}}{\texttt{High} + \texttt{Low}} = \frac{\texttt{High}}{\texttt{Period}}$$

PWM Mode

- Up/down count
  - Up to CCR0 (10)
  - Down to 0
- P2.4=1 when timer equals TA0CCR1 (7) on way down
- P2.4=0 when timer equals TA0CCR1 (7) on way up





T = Prescale/12MHz
Period = 2*T*CCR0
DutyCycle = CCR1/CCR0

PWM Outputs on P2.4 and P2.5

```
// SMCLK = 48MHz/4 = 12 MHz, 83.33ns
// Counter counts up to TA0CCR0 and back down
// Let Timerclock period T = 8/12MHz = 666.7ns
// Period of P2.4 is period*1.333us, duty cycle is duty1/period
// Period of P2.5 is period*1.333us, duty cycle is duty2/period
void PWM_Init12(uint16_t period, uint16_t duty1, uint16_t duty2){
  P2->DIR |= 0x30;          // P2.4, P2.5 output
  P2->SEL0 |= 0x30;        // P2.4, P2.5 Timer0A functions
  P2->SEL1 &= ~0x30;    // P2.4, P2.5 Timer0A functions
  TIMER_A0->CCTL[0] = 0x0080;    // CCI0 toggle
  TIMER_A0->CCR[0] = period;       // Period is 2*period*8*83.33ns is 1.333*period
  TIMER_A0->EX0 = 0x0000;          //    divide by 1
  TIMER_A0->CCTL[1] = 0x0040;    // CCR1 toggle/reset
  TIMER_A0->CCR[1] = duty1;        // CCR1 duty cycle is duty1/period
  TIMER_A0->CCTL[2] = 0x0040;   // CCR2 toggle/reset
  TIMER_A0->CCR[2] = duty2;       // CCR2 duty cycle is duty2/period
  TIMER_A0->CTL = 0x02F0;         // SMCLK=12MHz, divide by 8, up-down mode
// bit           mode
// 9-8   10    TASSEL, SMCLK=12MHz
// 7-6   11    ID, divide by 8
// 5-4   11    MC, up-down mode
// 2     0      TACLR, no clear
// 1     0      TAIE, no interrupt
// 0            TAIFG
}
```

You will convert it to
TA0.4/P2.7
TA0.3/P2.6

T = 8/12MHz
Period = 2*T*15000 = 10ms
DutyCycle = CCR1/15000

PWM Outputs on P2.4 and P2.5

You will convert it to
TA0.4/P2.7
TA0.3/P2.6

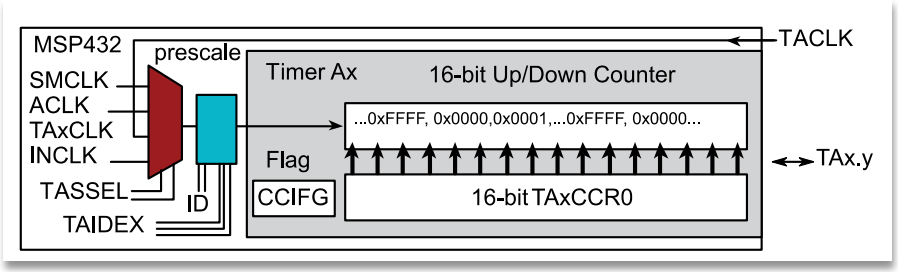
```
//*************************PWM_Duty1*******************************
// change duty cycle of PWM output on P2.4
// Inputs:  duty1
// Outputs: none
// period of P2.4 is 2*period*666.7ns, duty cycle is duty1/period
void PWM_Duty1(uint16_t duty1){
  TIMER_A0->CCR[1] = duty1;        // CCR1 duty cycle is duty1/period
}

//*************************PWM_Duty2*******************************
// change duty cycle of PWM output on P2.5
// Inputs:  duty2
// Outputs: none// period of P2.5 is 2*period*666.7ns, duty cycle is duty2/period
void PWM_Duty2(uint16_t duty2){
  TIMER_A0->CCR[2] = duty2;        // CCR2 duty cycle is duty2/period
}
```

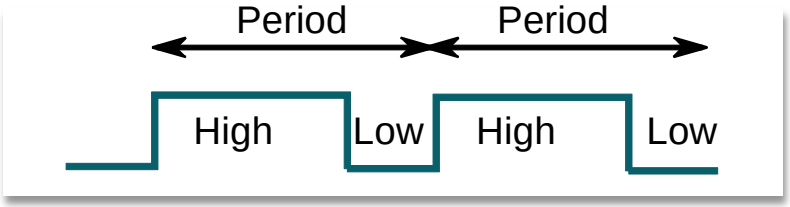
DutyCycle
- Precision 14999 alternatives (0 to 14998)
- Range 0 to 99.99%
- Resolution 0.0067%

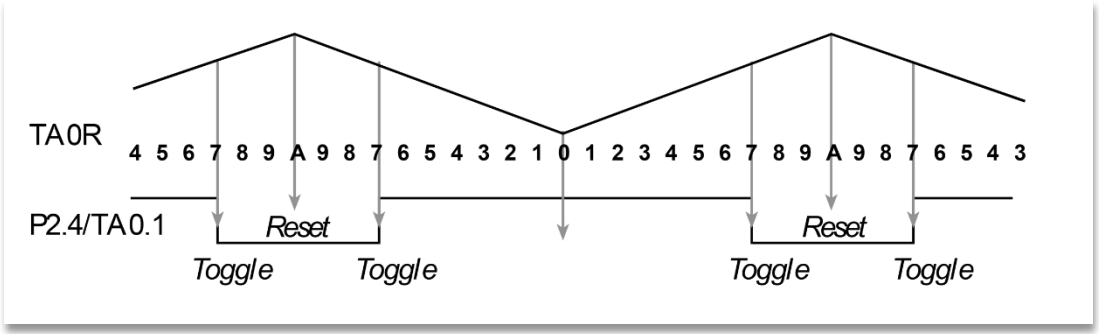
Summary

PWM output

- Adjust power to motors
- Two independent outputs







ti.com/rslk

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265