UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

PRODUCT NAME	MANUFACTURER PART NUMBER	SMD #	DEVICE TYPE	INTERNAL PIC NUMBER
Arm Cortex M0+	UT32M0R500	5962-17212	PWM Module	QS30

Table 1: Cross Reference of Applicable Products

1.0 Overview

Pulse Width Modulation (PWM) is a simple digital technique to control the ON state or theswitching period of voltage output. PWM applications range from controlling motors to controlling the brightness of an LED. The latter is used for illustration in this application note.

Figure 1 shows the basic diagram of a PWM channel output.

Figure 1: PWM basic diagram

By keeping the period constant, the average value of the PWM output is linearly proportional to the duty cycle, which is defined as follows:

$$Duty\ Cycle = \frac{Pulse\ ON}{Period}$$

Figure 2 shows a PWM output when the duty cycle equals to ¼.

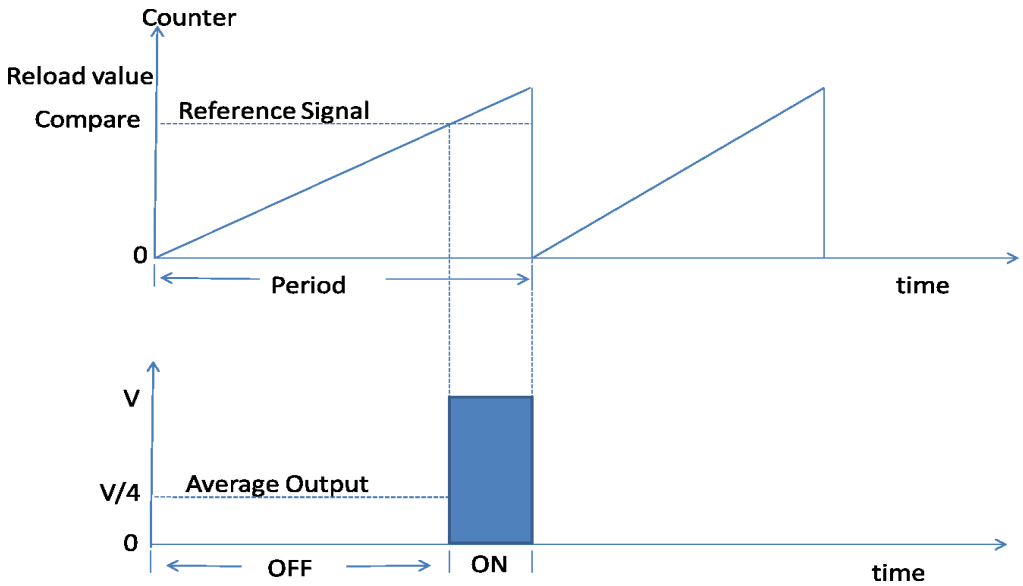

Figure 2: PWM output when Duty Cycle is ¼

CAES PIONEERING ADVANCED ELECTRONICS
4350 Centennial Blvd. | Colorado Springs CO 80907 USA | 1-800-645-8862 | cobhamaes.com

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

2.0 Application Note Layout

This application note (AN) provides a brief description of the PWM unit's memory map, configuration and programming.

3.0 PWM Module Hardware

There are three PWM controllers within the UT32M0R500 PWM module. The PWM module is mapped to the memory region from 0x4000_3000 to 0x4000_3FFF. It has 23 registers. For more information on each register, refer to Chapter 17 of the UT32M0R500 Functional Manual.

3.1 PWM Unit 8-bit Prescaler

The system clock drives each prescaler for the three different PWM channels. When the system clock scaler underflows, the SCALREG reloads the user preset value for each of the different PWM channels.

3.2 PWM Unit Control Register

The Control Register (CTRLREG) selects the number of system clock scalers by setting the appropriate (SCALERSEL), bits [10:8]. The first scaler register is address 0. Core enable (EN), bit [0], enables the PWM unit to generate outputs.

3.3 PWM Unit IRQ Register

The Interrupt Pending Register (IRQREG) in the core sets the corresponding bit and generates an interrupt. Software can read IRQREG to check which PWM channel generated an interrupt. The bit for the particular channel is cleared by writing 1 to it.

3.4 PWM Channel 16-bit Period value

Each of the three PWM channels has its own 16-bit period value in the PWM_REG_<x> register, and unlike the Prescaler referred in Section 3.1, the PWM channel period value is programmable. Hardware automatically reloads the counter value with the 16-bit value from the PWM period register. When the PWM period register is updated by software, the core value is not updated immediately, instead a shadow register is used to hold the new value until a new PWM period starts. Minimal operational value for the PWM period is 2.

3.5 PWM Channel 16-bit Compare value

Each of the three PWM channels has its own 16-bit compare value in the PWM_COMP_<x> register. When the PWM counter reaches the compare value, the PWM output switches to the ON position.

When the PWM compare register is updated by software, the core value is not updated immediately; instead a shadow register is used to hold the new value until a new PWM period starts. Minimal operational value for the PWM compare is 1.

3.6 PWM Channel 8-bit Dead Band Register

Each of the three PWM channels has its own 8-bit dead band register, PWM_DBCOMP_<x>. The dead band time

expires when the dead band counter reaches the compare value. When the PWM dead band register is updated by software, the core value is not updated immediately, instead a shadow register is used to hold the new value until a new PWM period starts.

3.6.1 PWM Channel Control Register

Each of the three PWM channels has its own control register, PWM_CTRL_<x>. The basic setup for each channel consist of selecting the system clock scaler used by the particular PWM channel (SCALERSEL), bit[12:10]; setting the PWM channel to active high or low (POL) bit[1]; and enabling the PWM channel (EN), bit [0]—Enabling this bit takes effect on the next scaled-clock rising edge. For more information on more configurations, refer to Chapter 17 of the UT32M0R500 Functional Manual.

4.0 PWM Unit Initialization

Each PWM channel supports a maximum period of 335ms given by:

$$Period = pclk * PWMCounter * ClockScaler$$
$$Period = 20ns * 65536 * 256$$
$$Period = 335ms$$

For the LED example, the period is:

$$Period = 20ns * 1024 * 256$$
$$Period = 5.2ms$$

And keeping the period fixed, the duty cycle of ¼ is:

$$Duty\ Cycle = 1 - \frac{1024}{768}$$
$$Duty\ Cycle = 0.25$$

Code 1 initializes PWM 0 as the LED output with GPIO44 set as alternate function, and for specifics on the API's, refer to StdPeriphLib at www.cobhamaes.com/hirel.

```
// Init GPIO2
GPIO_StructInit(&GPIO_InitStruct);
```

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

```
GPIO_Init(GPIO2, &GPIO_InitStruct);

//GPIO44, alternate function: PWM 0
GPIO_SetPinDirectionsRaw (GPIO2, GPIO2_PIN_GPIO44_OUTPUT);
GPIO_SetPinFunctionsRaw (GPIO2, GPIO2_PIN_GPIO44_ALT_FUNC);

// Set PWM Init structure defaults;
// ScalerSelect = PWM_SCALER_0; Scaler = 0xFF; NoUpdate = FALSE;
PWM_StructInit (&PWM_InitStruct);
```

Code 1: PWM 0 Initialization

5.0 PWM Unit Programming

Section 3.0 presented some of the basic configurations for the PWM core and each of the PWM channels. The following sections show programming examples by making use of CAES API's for the UT32M0R500.

5.1 PWM 16-bit Period Register

By setting the period in the `PWM_ChanInitStruct` and passing PWM number to the `PWM_SetChanConfig`, the API configures the period for the particular channel, see Code 2.

```
// Period register bits [15..0]
PWM_ChanInitStruct.Period = 1024;
PWM_SetChanConfig (PWM,0, &PWM_ChanInitStruct);
```

Code 2: PWM 0 API for the period value

5.2 PWM 16-bit Compare Register

By setting period in the `PWM_ChanInitStruct` and passing PWM number to the `PWM_SetChanConfig`, the API configures the period for the particular channel, see Code 3.

```
// Compare register bits [15..0]
PWM_ChanInitStruct.SwitchComparator = 768;
PWM_SetChanConfig (PWM, 0, &PWM_ChanInitStruct);
```

Code 3: PWM 0 API for the compare value

5.3 PWM Channel Control Register

By setting the period in the `PWM_ChanInitStruct` and passing the PWM number to the `PWM_SetChanConfig`, the API configures the period for the particular channel.

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

```
PWM_ChanInitStruct.Polarity = 1;
PWM_SetChanConfig (PWM, 0, &PWM_ChanInitStruct);

// Enable CHAN_PWM_0
PWM_ChanCmd (PWM, CHAN_PWM_0, 1);
```

Code 4: PWM 0 API for control Register

5.4 PWM Channel Interrupt

All PWM channels share one interrupt (IRQ), which is mapped to number 3 in the Interrupt Vector Table of the UT32M0R500. The address of interrupt 3 in the Interrupt Vector Table is mapped to the `PMW_IRQHandler` which is the interrupt service routine (ISR) for all three different channels. In the ISR, software must check for which interrupt happened.

```
// Enable interrupt
IntConfig.IRQ_Enable = ENABLE;

// Type of interrupt, irqt bit = 0, generates interrupt on PWM period match
IntConfig.IRQ_Type = PWM_IRQ_PERIOD_MATCH;

// Type of interrupt, irqt bit = 1, generates interrupt on PWM compare match//
IntConfig.IRQ_Type = PWM_IRQ_COMPARE_MATCH; // Uncomment this for compare
// match

IntConfig.IRQ_Scaler =  0x0000;
PWM_IntConfig (PWM, CHAN_PWM_0, &IntConfig);
```

Code 5: PWM 0 API for setting interrupt

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

5.4.1 PWM Channel Interrupt Period Match

Figure 4 shows enabling the interrupt Period match.

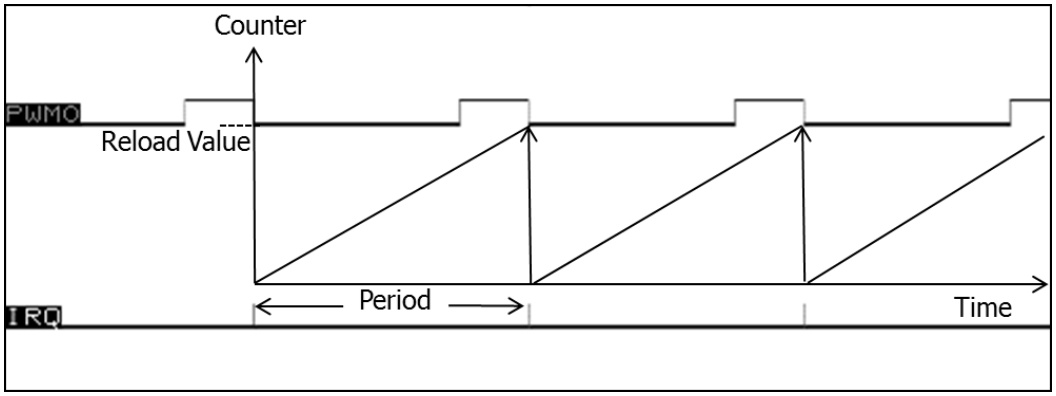

Figure 4: PWM Period Match

5.4.2 PWM Channel Interrupt Compare Match

Figure 4 shows enabling the interrupt Compare match.

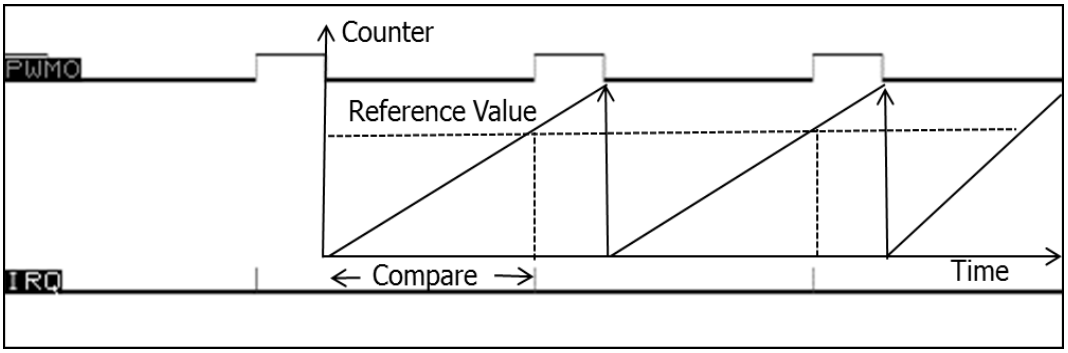

Figure 5: PWM Compare Match

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

Putting it all together, code 6 shows the main subroutine with a placeholder for initialization from the previous sections and an endless loop for doing something useful. The `PWM_IRQHandler` is the interrupt service routine for handling the particular PWM channel output. It uses linked list to check what PWM channel happened. Figure 3 shows the PWM 0 channel output for the LED example with period of 5.2ms and ¼ duty cycle.

```
int main (void){
// initialization and setting from previous sections go here.
       for(;;){
       // do something useful here.
       __ASM_volatile("nop");
       }
}

void PWM_IRQHandler(void){
       NodePtr Ptr;
       Ptr = &PWMTasks[0]; //Points to first task in linked list
       while (Ptr){ // Handles all PWM requests
              If (PWM -> IRQREG & (Ptr->Mask)){
                     (*Ptr->PWMHandler)(); // Execute PWM handler
                     }
              Ptr = Ptr->Next; // Poll next device
       }
}
```

Code 6: Sample program for the LED

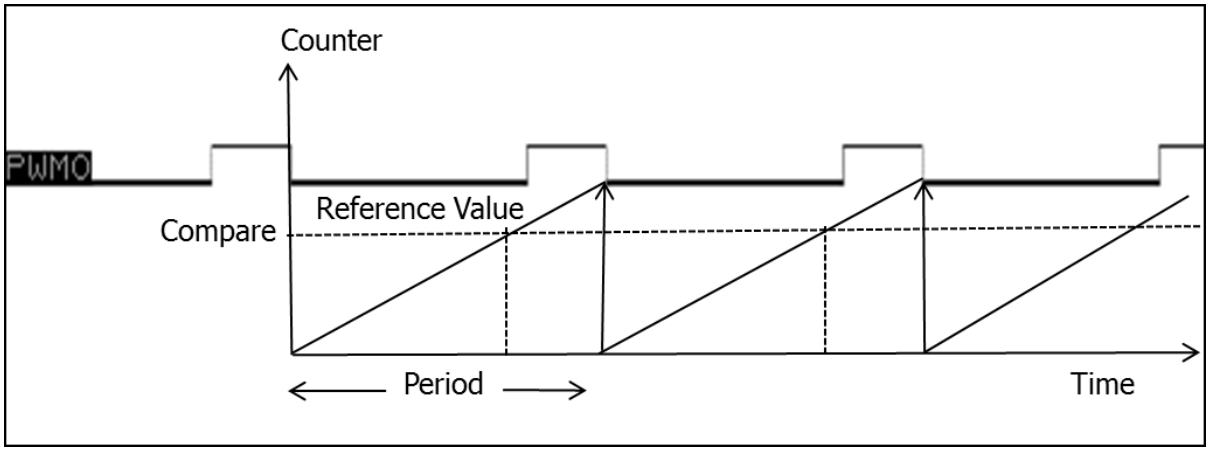

Figure 3: PWM output for LED with period 5.2 when Duty Cycle is 1/4

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

6.0 Summary and Conclusion

While Pulse Width Modulation (PWM) is a simple digital technique to control the ON state or the switching period of voltage output, the steps for initializing and configuring it to turn on an LED can be quite involved as illustrated from Section 5.0.

For more information about our UT32M0R500 microcontroller and other products, please visit our website: www.cobhamaes.com/HiRel.

CAES PIONEERING ADVANCED ELECTRONICS
4350 Centennial Blvd. | Colorado Springs CO 80907 USA | 1-800-645-8862 | cobhamaes.com

UT32M0R500 32-bit Arm™ Cortex® M0+ Microcontroller - Enable the PWM Module

7.0 Revision History

Date	Rev. #	Author	Change Description
Dec 2017	1.0.0	JA	Initial Release

The following United States (U.S.) Department of Commerce statement shall be applicable if these commodities, technology, or software are exported from the U.S.: These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.

CAES PIONEERING ADVANCED ELECTRONICS

4350 Centennial Blvd. | Colorado Springs CO 80907 USA | 1-800-645-8862 | cobhamaes.com