PWM SIGNAL GENERATION USING UT3

Product Family: **EM BLE System-On-Chip solution**

Part Number: EM9305
Keywords: Universal Timer, PWM, UT3, square wave, duty cycle

PURPOSE

The PWM driver being no available yet the purpose of this application note is to provide code samples and guidelines to configure hardware universal timer to generate PWM signal on GPIO pins of the EM9305 device.

SCOPE

Applicable to EM9305, the pulse width modulated signal generation using hardware timers requires the CPU system to remain in active state.

SOFTWARE PRE-REQUISITE

Synopsys Metaware IDE toolchain has been installed (Full or Lite version) in Windows environment.
Version 2021.03 or more recent version can be used. EM9305 SDK EMBLUE v0.3 or more recent is required.

PRINCIPLE

In addition to sleep timer and protocol timer reserved for EM system and BLE activities, EM9305 integrates two universal timers (identified by UT2 and UT3) which can be used by the application. They have the following features:

- 32-bit up counter, selectable auto-reload
- clock source: system clock, GPIO
- 7-bit pre-scaler
- SW start/stop
- HW start/stop
- input capture on HW events (GPIO)
- input capture on SW event
- limit value
- 4 channels output compare value (PWM)
- output to GPIO
- interrupt on limit value, compare and input capture

The universal timer configuration may be changed only when the universal timer is disabled. The universal timer can run when system is in not in sleep mode. The PWM signal generation is using limit value to set frequency and compare value to set the duty cycle. Timer has up to four comparison values allowing to generate allowing to generate up to four square wave signals with same frequency and different duty cycles.

CODE SAMPLES

Timer output compare values needs to be associated with GPIO pads, EM9305 IO mapping provides full flexibility. Code sample taking care of GPIO configuration should done in NVM_ConfigModules function when device is booting from a reset state:

```
        // Update the GPIO default configuration.
        // No need to modify the GPIO config after a wake-up from sleep because
        // the configuration is kept in persistent memory.
        // Disable all inputs.
        gGPIO_Config.hardwareState.RegGPIOInputEn.r32 = 0x000;
        // All other output pins low.
        gGPIO_Config.hardwareState.RegGPIODataOut.r32 =  0x000;
        // Enable outputs test pins + debug (10u)
        gGPIO_Config.hardwareState.RegGPIOOutputEn.r32 = (uint32_t)(GPIO_MASK(6u)|GPIO_MASK(7u)|GPIO_MASK(8u)|GPIO_MASK(9u)|GPIO_MASK(10u));
        // Disable pull-downs on all pins.
        gGPIO_Config.hardwareState.RegGPIOPdEn.r32 = (uint32_t)0x000;
        // Disable pull-up on all pins.
        gGPIO_Config.hardwareState.RegGPIOPuEn.r32 = (uint32_t)0x000;
        // Set output function on GPIO6.
        gGPIO_Config.hardwareState.RegGPIOOutSel1.regs.GPIOOutSel6 = (uint8_t)GPIO_PIN_FUNC_OUT_UNITIMER3_OUT0;
        // Set output function on GPIO7.
        gGPIO_Config.hardwareState.RegGPIOOutSel1.regs.GPIOOutSel7 = (uint8_t)GPIO_PIN_FUNC_OUT_UNITIMER3_OUT1;
        // Set output function on GPIO8.
        gGPIO_Config.hardwareState.RegGPIOOutSel2.regs.GPIOOutSel8= (uint8_t)GPIO_PIN_FUNC_OUT_UNITIMER3_OUT2;
        // Set output function on GPIO9.
        gGPIO_Config.hardwareState.RegGPIOOutSel2.regs.GPIOOutSel9 = (uint8_t)GPIO_PIN_FUNC_OUT_UNITIMER3_OUT3;
```

PML API from <pml.h> call to keep the device in active mode by disabling automatic sleep mode:

```
(void) PML_AutomaticSleepModeDisable( PML_AUTO_SLEEP_DISABLE_ALL, true);
```

To resume sleep mode capability set boolean argument to false:

```
(void) PML_AutomaticSleepModeDisable( PML_AUTO_SLEEP_DISABLE_ALL, false);
```

T9305_periph.h header file providing access to t9305_uni_tim.h

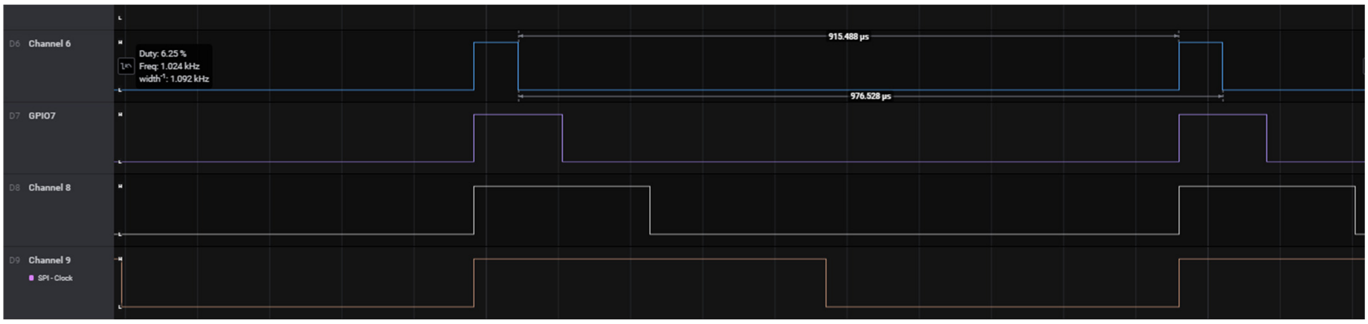
```
#include <t9305_periph.h>
```

UT3 code sample to generate 4 signals of 1024KHz with 50%, 25%, 12,5% and 6,25% duty cycles.

```
void Melody_SetBuzzer( uint16_t frequency )
{
    // Disable the timer prior to setting a new value.
    UTIM->RegUniTimer3Ctrl.r32 = 0;
    if ( 0 < frequency)
    {
        // no prescaler ((24 Mhz) / frequency - 1 )
        uint32_t Limit=  ( ( 24000000u ) / (( uint32_t ) frequency ) ) - 1u;
        UTIM->RegUniTimer3Cfg.r32 = UT3_AUTO_RESTART_CFG(1)|UT3_ACT_LIMIT(3)| UT3_ACT_COMPARE(3);
        UTIM->RegUniTimer3Limit.r32 = Limit;
        UTIM->RegUniTimer3Ctrl.r32 = UT3_ENABLE(1) | UT3_START_SW(1) | UT3_OUT_EN0(1) | UT3_OUT_EN1(1) | UT3_OUT_EN2(1)| UT3_OUT_EN3(1);
        UTIM->RegUniTimer3Compare0.r32 = Limit>>4;  // 6.25%
        UTIM->RegUniTimer3Compare1.r32 = Limit>>3;  // 12.5%
        UTIM->RegUniTimer3Compare2.r32 = Limit>>2;  // 25%
        UTIM->RegUniTimer3Compare3.r32 = Limit>>1;  // 50%
    }
}
```

Timer Auto reload is achieved with `UT3_AUTO_RESTART_CFG(1)` configuration when reaching the limit value.
Output is toggled when reaching the limit value using `UT3_ACT_LIMIT(3)` configuration.
Output is toggled when reaching the limit value using `UT3_ACT_COMPARE(3)` configuration.
UT3 is enable using `UT3_ENABLE(1)` control.
UT3 compare output are enabled using `UT3_OUT_EN0(1) | UT3_OUT_EN1(1) | UT3_OUT_EN2(1)| UT3_OUT_EN3(1);` controls.
UT3 clock source is using system clock of 24MHz by default for accurate timing generation the XTAL HF clock source should be preferably activated. A timer clock pre-scaler configuration allows to divide selected clock source by factor from 1 to 128 using `UT3_PRESCALER_SEL(x)` with x in range 0 (no division) to 7 (division by 128)
Start of timer under software control is performed using `UT3_START_SW(1)` control.

Example of signal waveform generated with frequency set to 1024Hz:



CONCLUSION

Please contact your EM support channel in case you need more information this application note still in preliminary stage.

EM Microelectronic-Marin SA ("EM") makes no warranties for the use of EM products, other than those expressly contained in EM's applicable General Terms of Sale, located at http://www.emmicroelectronic.com. EM assumes no responsibility for any errors which may have crept into this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein.

No licenses to patents or other intellectual property rights of EM are granted in connection with the sale of EM products, neither expressly nor implicitly.

In respect of the intended use of EM products by customer, customer is solely responsible for observing existing patents and other intellectual property rights of third parties and for obtaining, as the case may be, the necessary licenses.

Important note: The use of EM products as components in medical devices and/or medical applications, including but not limited to, safety and life supporting systems, where malfunction of such EM products might result in damage to and/or injury or death of persons is expressly prohibited, as EM products are neither destined nor qualified for use as components in such medical devices and/or medical applications. The prohibited use of EM products in such medical devices and/or medical applications is exclusively at the risk of the customer