AN12863

How to Generate 50% Duty of PWM on LPC802

Rev. 0 — May 2020

Application Note

Contents

1 Introductions

1.1 LPC802 overview

The LPC802 is an entry level of part which are an Arm® Cortex® -M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 15 MHz. It supports 16 KB of flash memory and 2 KB of SRAM.

The peripheral complement of the LPC802 includes:

- One I2C-bus interface
- Up to two USARTs
- One SPI interface
- One multi-rate timer
- Self-wake-up timer
- One general purpose 32-bit counter/timer
- One 12-bit ADC
- One analog comparator
- Function-configurable I/O ports through a switch matrix
- Up to 17 general-purpose I/O pins

Figure 1 shows the LPC802 block diagram.

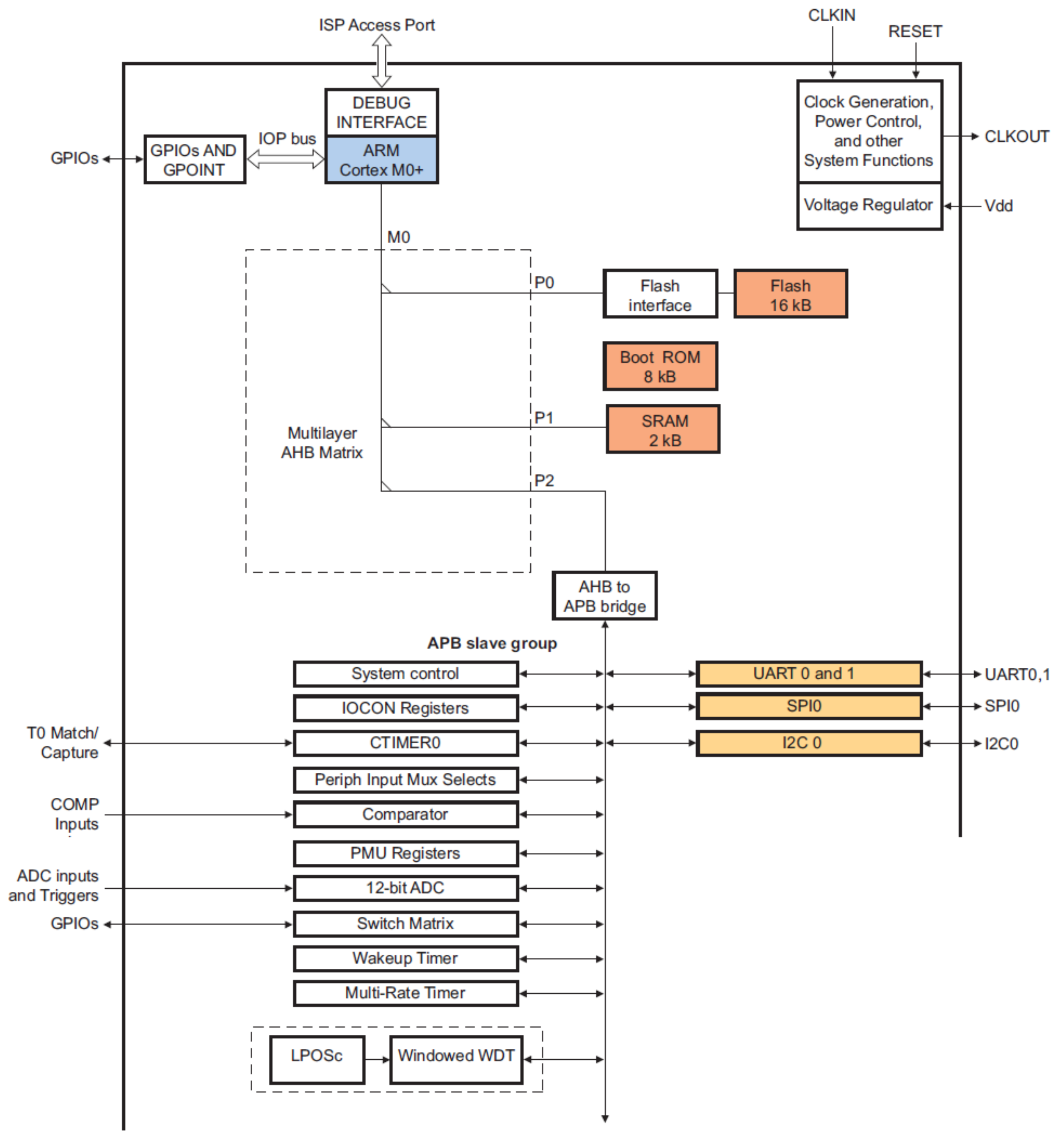


Figure 1. LPC802 block diagram

1.2 Why this application note?

The LPC802 is designed to be easily used and applied to sensor gateway, simple motor control, gaming controllers, 8/16-bit application, and so on.

In some applications on LPC802, a PWM signal is required to be generated to keep the signal output without any software interfering. This application is easy to be implemented. The general counter/timer (CTIMER) on LPC802 can support it. However, there is a restriction on the CTIMER that it cannot generate a 50% duty of PWM when the frequency of PWM is got from a clock source divided by an odd number. For example, the 1.8 Mhz of PWM frequency is got from 9 Mhz of the main clock divided by 5 (odd number). At this point, the CTIMER cannot meet the requirements, while with no problem if the divisor is an even number.

There is another simple module – CLKOUT for making a clock source signal output to a pin on LPC802. It is also a good choice for generating a PWM signal with it. Unfortunately, the restriction is also applied to it.

There appears no way for this requirement. What should be done?

This application note introduces an ingenious and simple solution to generate a 50% duty of PWM at the frequency got from an odd number of divider.

2 Solution

In summary, the requirement is generating a 50% duty of PWM at the frequency got from an odd number of divider and the PWM signal output is continuous without software interfering once it generates. The solution is to use USART clock output in synchronous master mode.

To understand this solution better, it is assumed that the frequency of PWM output is 1.8 Mhz from 9 Mhz of main clock frequency divided by 5 and the main clock is generated from 9 Mhz of internal FRO. And USART0 is used. This means the USART0 clock frequency in synchronous master mode will be 1.8 Mhz since it is used as a PWM signal.

The solution code is based on LPC802 SDK and contains the below basic points:

- Get the desired frequency of main clock from FRO.
- Assign USART0 clock output to a pin.
- Set USART0 clock output frequency.
- Configure USART0 to synchronous master mode.
- Configure the USART0 clock output to be continuous.
- Start USART0 clock output.

2.1 Getting the desired frequency of main clock from FRO

The internal Free Running Oscillator (FRO) provides a selectable 9 MHz, 12 MHz and 15 MHz outputs that can be used as a system clock. FRO output frequency can be configured through a simple call to the ROM. It sets up the `fro_oscout` (30 MHz/24 MHz/18 MHz) to provide frequencies of 15 MHz, 12 MHz, and 9 MHz.

It means that calling the ROM API with 18 Mhz of parameter can get 9 Mhz of FRO frequency output and the FRO is selected as the clock source of main clock. Figure 2 shows the main clock generation from FRO.

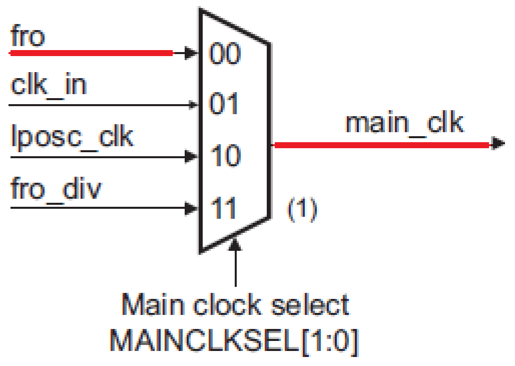
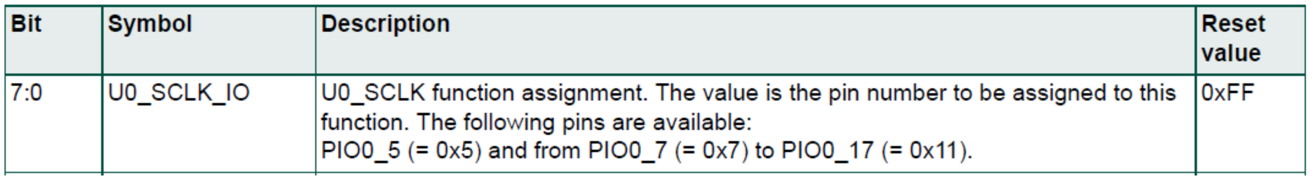


Figure 2. Main clock generation

In LPC802 SDK, 9 Mhz of main clock can be got directly from FRO just calling the function `BOARD_BootClockFRO18M()` where the ROM API is called.

2.2 Assigning USART0 clock output to a pin

After getting the main clock for system, a pin is required to be assigned for PWM signal output – USART0 clock output. The digital peripheral function signal can be assigned flexibly to pins by SWM module on LPC802. Assignment of USART0 clock signal (named `U0_SCLK`) to pins is configured in the related register, as shown in Figure 3.



Bit	Symbol	Description	Reset value
7:0	U0_SCLK_IO	U0_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_17 (= 0x11).	0xFF

Figure 3. USART clock generation

According to the description of the register, `PIO0_5` and from `PIO0_7` to `PIO0_17` can be assigned as the pin for USART0 clock signal. In this application note, `PIO0_8(=0x8)` is assigned. In LPC802 SDK, there is a function `SWM_SetMovablePinSelect()` to set it.

2.3 Setting USART0 clock output frequency

Figure 4 shows the block diagram of USARTn (n=0,1) clock generation.

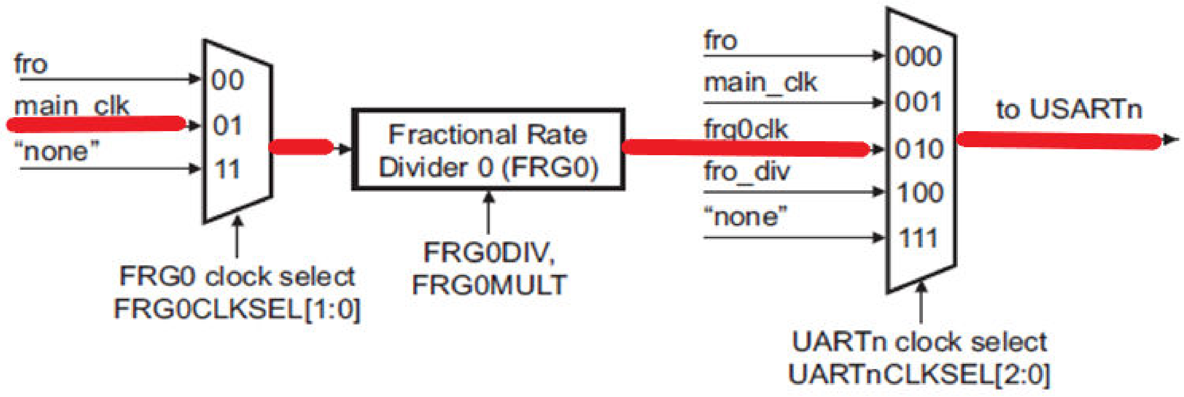


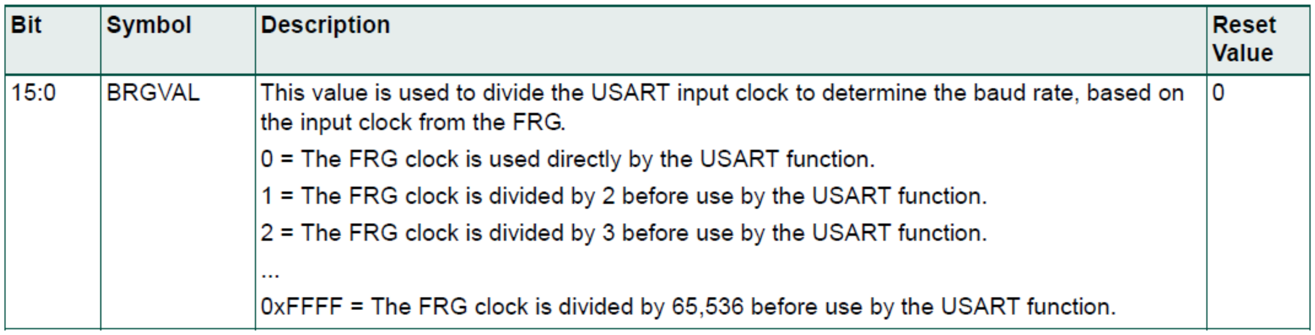
Figure 4. USART clock generation

In this application note, USART clock generation route is as marked in red.

NOTE

The generated USART clock (see **to USARTn**) is the base clock used for USART internal operation instead of the USART clock output as PWM signal.

In synchronous mode, the USART clock output as PWM signal is Un_SCLK = FCLK/(BRGVAL+1), where the FCLK is right the USART clock in the block diagram and BRGVAL is configured in the USART Baud Rate Generator register, as shown in Figure 5.



Bit	Symbol	Description	Reset Value
15:0	BRGVAL	This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0 = The FRG clock is used directly by the USART function. 1 = The FRG clock is divided by 2 before use by the USART function. 2 = The FRG clock is divided by 3 before use by the USART function. ... 0xFFFF = The FRG clock is divided by 65,536 before use by the USART function.	0

Figure 5. USART Baud rate generator register

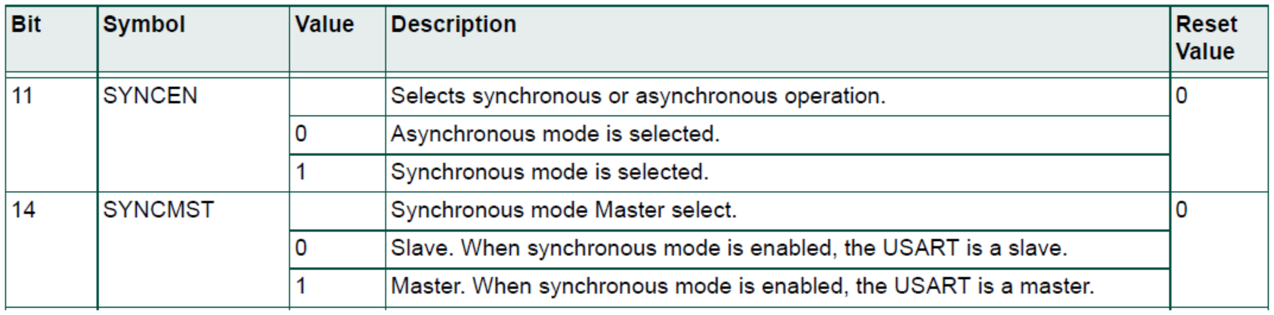
So to get 1.8 Mhz of desired frequency of output signal, the BRGVAL should be 4 and FCLK should be 9 Mhz per the above formula. It is, Un_SCLK = 9Mhz/(4+1)=1.8 Mhz. In the LPC802 SDK, the function of `USART_SetBRGValue()` can be called to complete it.

NOTE

When using the reset default values in Fractional Rate Divider registers, the FCLK clock will be 9 Mhz equal to the main clock. So there is no need to do anything for the registers.

2.4 Configuring USART0 to synchronous master mode

USART can provide clock output on a pin as PWM signal output only when it is in synchronous master mode. This is why USART must be configured as synchronous master mode. See Bit 11 and 14 in USART configuration register for the detailed settings of the modes in Figure 6.



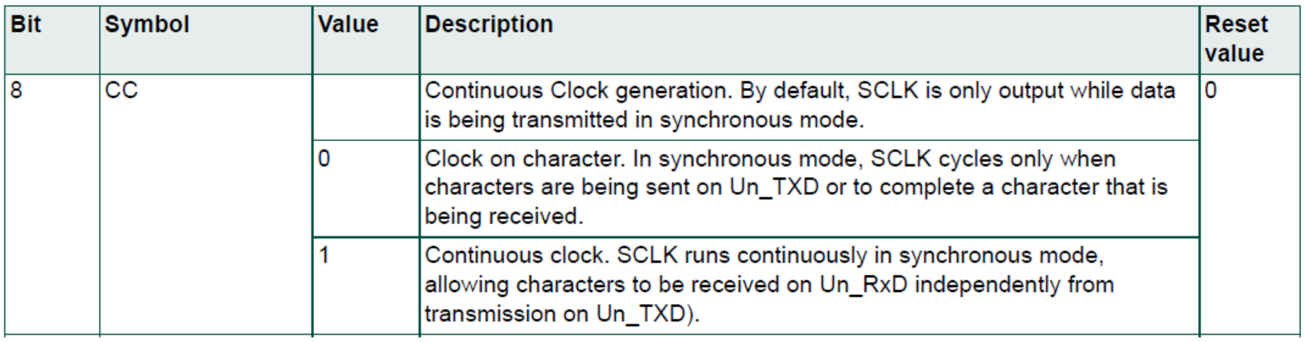
Bit	Symbol	Value	Description	Reset Value
11	SYNCEN		Selects synchronous or asynchronous operation.	0
		0	Asynchronous mode is selected.	
		1	Synchronous mode is selected.	
14	SYNCMST		Synchronous mode Master select.	0
		0	Slave. When synchronous mode is enabled, the USART is a slave.	
		1	Master. When synchronous mode is enabled, the USART is a master.	

Figure 6. USART configuration register for synchronous master mode settings

When set **1** to bit 11 and 14, the synchronous and master modes are configured.

2.5 Configuring USART0 clock output to be continuous

With the above operations, the USART0 clock can be output as a 50% duty of PWM at 1.8 Mhz of frequency as long as it is started. But there is one more requirement that the output must be continuous without software interfering. In the synchronous mode, the USART supports to generate continuous clock by setting a bit in USART control register. See the descriptions for the details in Figure 7.

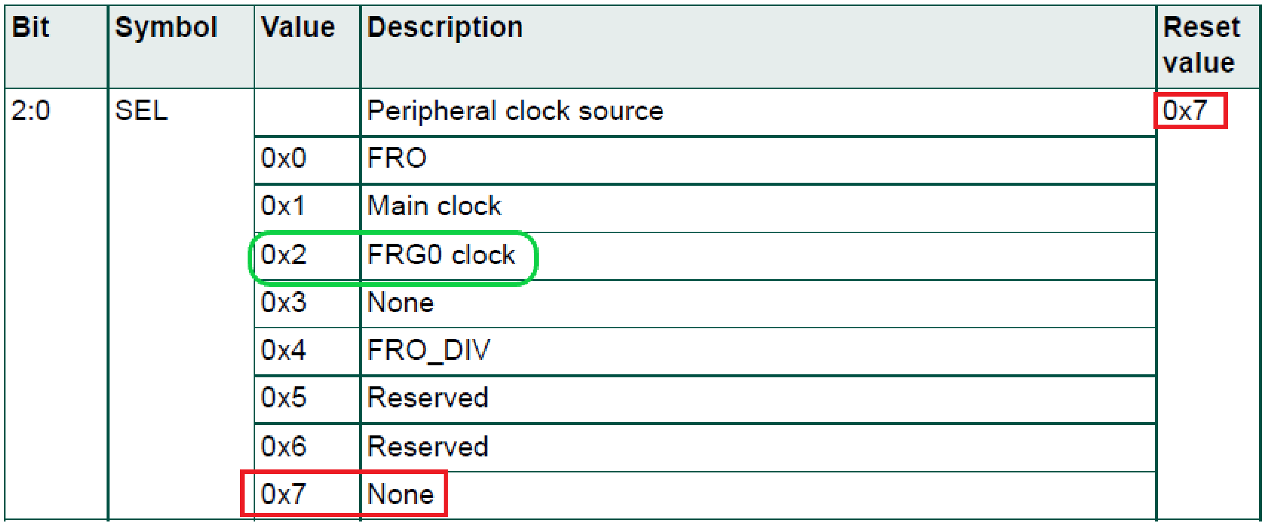


Bit	Symbol	Value	Description	Reset value
8	CC		Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode.	0
		0	Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received.	
		1	Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD).	

Figure 7. Configuration of continuous clock generation in USART control register

2.6 Starting USART0 clock output

It is to start USART0 clock output finally when all the settings are completed. In this application note, it is designed to start the clock output by selecting **frg0clk** as USART0 clock source (see Figure 3). It means the `frg0clk` is not selected to activate USART0 clock output until all the settings are completed. It is selected by setting `0x2` in UART0CLKSEL register where the reset default value is 0x7 that means no clock source selected. This also shows the value `0x7` can be used to stop the USART0 clock output. See the register in Figure 8 for the details.

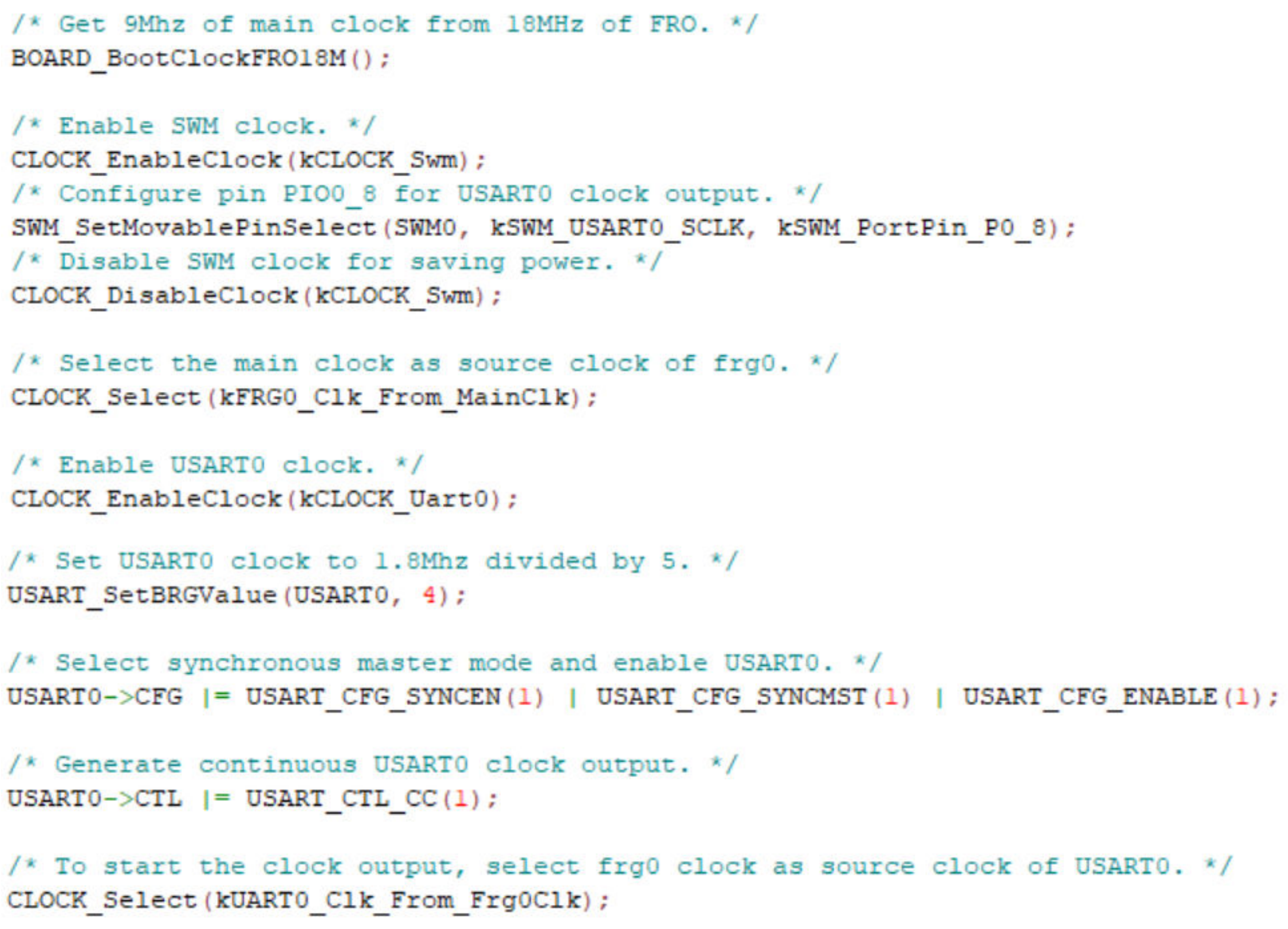


Bit	Symbol	Value	Description	Reset value
2:0	SEL		Peripheral clock source	0x7
		0x0	FRO	
		0x1	Main clock	
		0x2	FRG0 clock	
		0x3	None	
		0x4	FRO_DIV	
		0x5	Reserved	
		0x6	Reserved	
		0x7	None	

Figure 8. UART0 clock source select register (UART0CLKSEL)

2.7 Reference codes for the PWM output

Base on the above basic points for the solution, the entire reference codes for the solution are integrated as shown in Figure 9. With the codes merged into LPC802 SDK, the required PWM output can be generated.



```
/* Get 9Mhz of main clock from 18MHz of FRO. */
BOARD_BootClockFRO18M();

/* Enable SWM clock. */
CLOCK_EnableClock(kCLOCK_Swm);
/* Configure pin PIO0_8 for USART0 clock output. */
SWM_SetMovablePinSelect(SWM0, kSWM_USART0_SCLK, kSWM_PortPin_P0_8);
/* Disable SWM clock for saving power. */
CLOCK_DisableClock(kCLOCK_Swm);

/* Select the main clock as source clock of frg0. */
CLOCK_Select(kFRG0_Clk_From_MainClk);

/* Enable USART0 clock. */
CLOCK_EnableClock(kCLOCK_Uart0);

/* Set USART0 clock to 1.8Mhz divided by 5. */
USART_SetBRGValue(USART0, 4);

/* Select synchronous master mode and enable USART0. */
USART0->CFG |= USART_CFG_SYNCEN(1) | USART_CFG_SYNCMST(1) | USART_CFG_ENABLE(1);

/* Generate continuous USART0 clock output. */
USART0->CTL |= USART_CTL_CC(1);

/* To start the clock output, select frg0 clock as source clock of USART0. */
CLOCK_Select(kUART0_Clk_From_Frg0Clk);
```

Figure 9. Solution reference code for the required PWM generation

3 Test environment and result

To verify if the required PWM can be generated, the software and hardware environments are set up as below.

3.1 Hardware environment and setup

- LPCXpresso802 OM40000 Rev A board
- Personal computer
- Micro USB cable and some wires
- An oscillograph

As introduced, pin `PIO0_8` is designed as USART0 clock output for PWM generation. The pin is reserved on Arduino connector (named **CN3**) on LPCXpresso802 OM40000 Rev A board. The `PIO0_8` and GND pins are highlighted in the screenshot of the schematic shown as Figure 10. With a oscillograph connecting to `PIO0_8` pin and one GND pin via wires, the PWM signal can be captured and observed. And connect the USB port (CN1) on the board to the PC with micro USB cable.

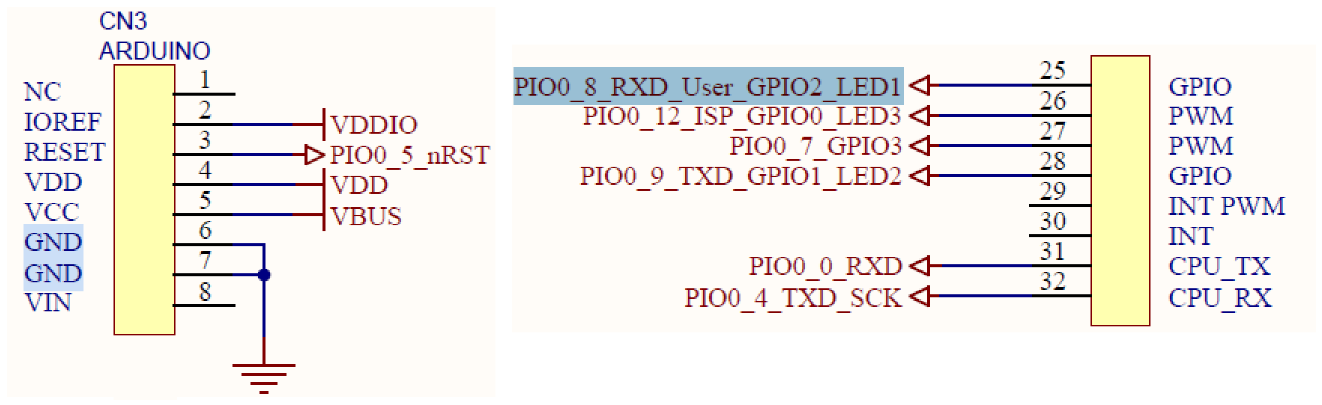


Figure 10. PIO0_8 pin on LPCXpresso802 board for measuring

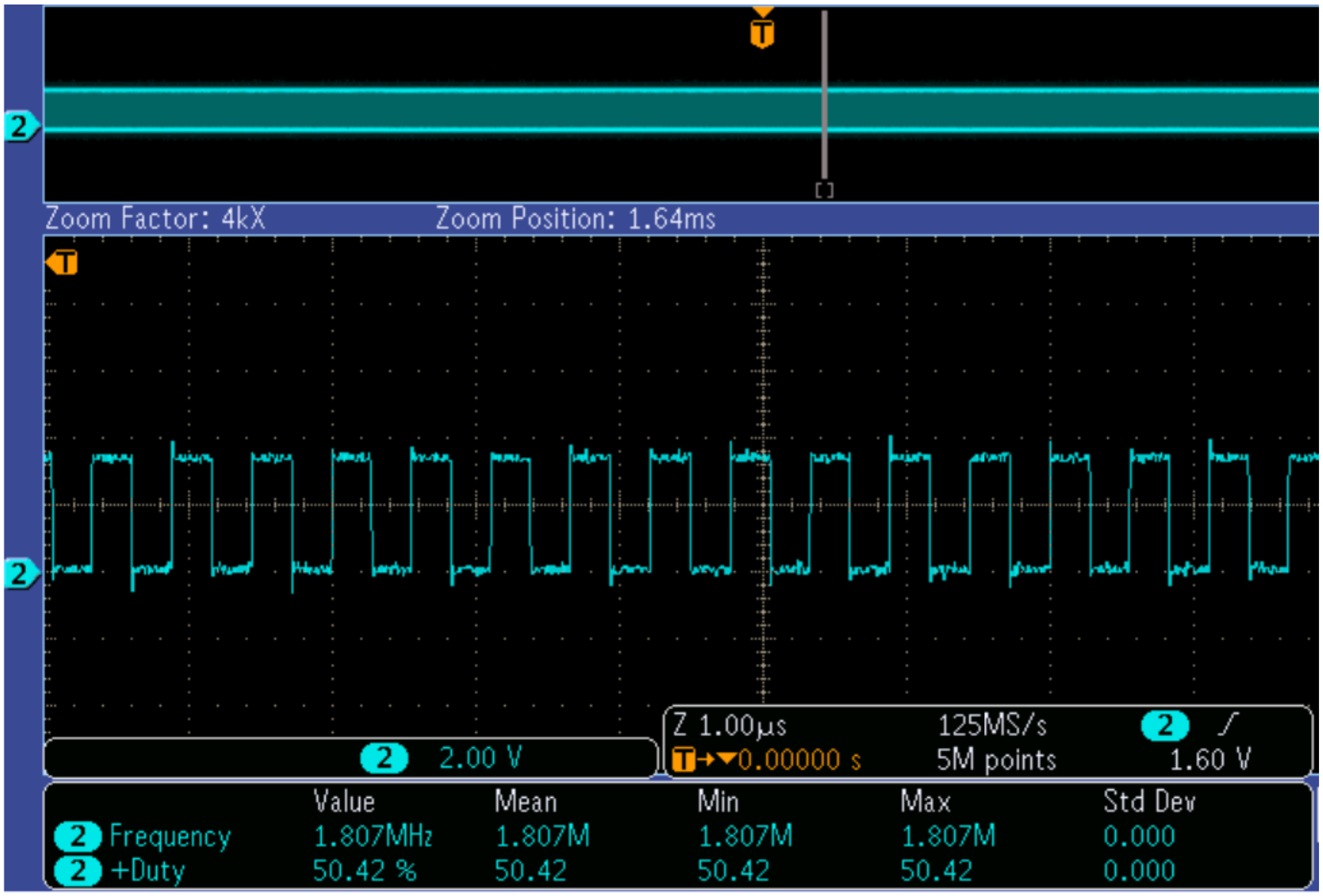
3.2 Software environment and setup

- LPCXpresso802 SDK (v2.4.0)
- KEIL MDK v5.27

A software package is attached with this application note. After abstracting it to the folder of *usart_pwm*, copy the folder under the path *[LPCXpresso802 SDK]\boards\lpcxpresso802\demo_apps\*.

3.3 Test result

After the hardware and software environment set up, open the project file *usart_pwm.uvprojx* with KEIL MDK, build, and program to the target board. With the system running, the PWM waveform can be observed with the oscillograph (see Figure 11). And it shows the measured frequency of the waveform is 1.8 Mhz and duty is 50% on the screen of oscillograph. Also the waveform is continuous. So it verifies that the required PWM is generated with the solution.



	Value	Mean	Min	Max	Std Dev
2 Frequency	1.807MHz	1.807M	1.807M	1.807M	0.000
2 +Duty	50.42 %	50.42	50.42	50.42	0.000

Figure 11. Required PWM output waveform

4 Summary

Usually, the standard counter/timer peripheral can be used for PWM generation. However, there is a restriction on LPC802 that the counter/timer cannot generate a 50% duty of PWM at the frequency got from an odd number of divider. The same restriction also applies to the CLKOUT module though it can generate a PWM as well.

This application note introduces an ingenious and simple solution to fix it, and with the solution, the PWM output can be continuous without software interfering once it generates. The solution is just to use USART0 clock output in synchronous master mode.

The solution code is based on LPC802 SDK and contains the below basic points:

- Get the desired frequency of main clock from FRO.
- Select a pin as the USART0 clock output.
- Set USART0 clock output frequency.
- Configure USART0 to synchronous master mode.
- Configure the USART0 clock output to be continuous.
- Start USART0 clock output.

The solution can be verified on the LPCXpresso802 OM40000 Rev A board.

Date of release: May 2020

Document identifier: AN12863