Freescale Semiconductor
Application Note

Document Number: AN4419
Rev. 0, 01/2012

Using DMA and GPIO to emulate timer functionality on Kinetis Family devices

by: Donnie Garcia, Hiroki Maruoka
Microcontroller Solutions Group

1 Introduction

Today's embedded applications require increasing levels of functionality. One key function that an MCU performs is the timer functionality. Freescale Kinetis devices already contain numerous timer peripherals. For example the MK device contains 3 FTM modules, 8ch, 2ch, 2ch, an RTC, PIT channels CMT and LPTMR. Even with all these timer resources, today's applications may require additional timer functionality. This application note explains "how to generate a PWM or clock by using the PIT, DMA and GPIO on Kinetis". Generally, these functions are implemented by a timer peripheral. Kinetis has the FlexTimer (FTM) as a timer module. However, the user will be able to control more timers using these methods given in the application note.

1.1 Features

- Clock
  - Clock can be generated by 1ch PIT, 1ch DMA and GPIO pin
  - Clock duty is 50%.
  - Clock maximum frequency is (PIT counter frequency)/2. The maximum PIT frequency is Bus clock.
- PWM

Contents

- PWM can be generated by 1ch PIT, 2ch DMA and GPIO pin
- PWM duty can be changed from 0% to 100%. When duty is 0%, GPIO output is L and when duty is 100%, GPIO output is H

2 Implementation details

This chapter explains the implementation of each function.

2.1 Clock

2.1.1 Initialization

The user needs to initialize each module as the following:

1. GPIO
   - Turn on the PORTx CLOCK and Interrupt Configuration of the Pin Control Register, PCR. The pin mux is set to enable the GPIO. The interrupt configuration is set to trigger the DMA Request on rising or falling edge.
   - Set the port data direction, PDDR for output.
2. PIT
   - Turn on the PIT clock. Set the PIT Module Control Register MDIS bit. MDIS is the module disable. This is used to disable the module clock. This bit must be enabled before any other setup is done.
   - Set the initial LDVAL register value. The PIT counter value is loaded from this register.
3. DMA
   - Turn on the DMA Mux clock.
   - Set the DMAMUX channel configuration register. The SOURCE field is set to the number of the accessing GPIO port.
   - Set the transfer control descriptor as the following:
   - saddr: the address of the output GPIO port bit information.
   - daddr: the address of the output GPIO port toggle register.
   - nbyte: 4
   - tcdAtter: the both of the ssize and dsize are 2
   - channelno: 0
   - Set the DMA Enable Request register

2.1.2 Procedure

1. Start the PIT timer.
2. The PIT triggers the DMA if PIT counter becomes 0. PIT counter loads the LDVALn register.
3. DMA toggles the GPIO output signal. GPIO output signal can be toggled by accessing the PTOR register.
4. The steps 2 and 3 are repeated automatically.

Figure below provides the procedure overview. PIT is used channel 0 (ch0), DMA is used ch0. PIT chn will trigger the DMA chn. PIT triggers the DMA and loads the counter value from LDVAL if the PIT counter value becomes 0. DMA toggles the GPIO output signal.

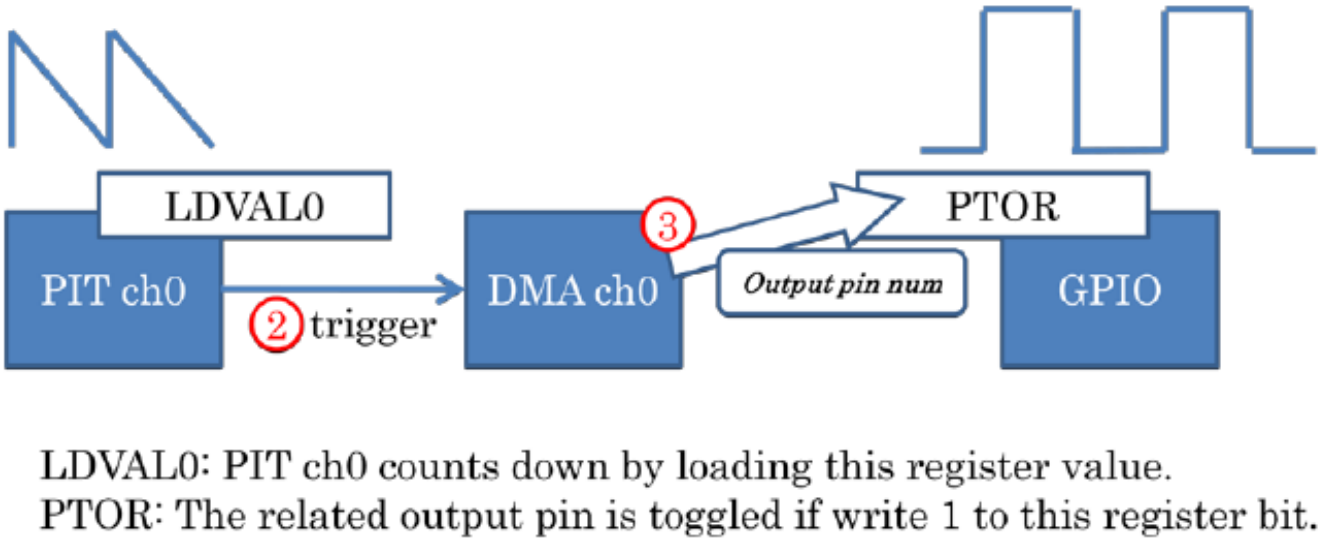

Figure 1. Clock procedure overview

2.2 PWM

2.2.1 Initialization

The user needs to initialize each module as given below:

1. GPIO
   - Set the Pin Mux Control and Interrupt Configuration of the Pin Control Register, PCR. The pin mux is set to the GPIO. The interrupt configuration is set to the DMA Request on either edge.
   - Set the port data direction, PDDR.
2. PIT
   - Turn on the PIT clock
   - Set the PIT Module Control Register MDIS bit. MDIS is the module disable. This is used to disable the module clock. This bit must be enabled before any other setup is done.
   - Set the initial LDVAL register value. The PIT counter value is loaded from this register.
3. DMA
   - Turn on the DMA Mux clock.
   - Set the DMAMUX channel configuration register. The SOURCE field is set to the number of the accessing GPIO port.
   - Set the transfer control descriptor as the followings:
     - tcd0
       - saddr: the address of the output GPIO port bit information.
       - daddr: the address of the output GPIO port toggle register.
       - nbyte: 4
       - tcdAtter: the both the ssize and dsize are 2
       - csr: set the link channel to tcd1
       - channelno: 0~4
     - tcd1
       - saddr: next PIT counter value address.
       - daddr: the address of the PIT LDVAL address.
       - nbyte: 4

- tcdAtter: the both the ssize and dsize are 2
- soff: 4
- slast: -8
- loopcount: 2
- channelno: 5~

- Set the DMA Enable Request register

2.2.2 Procedure

1. Start the PIT timer.
2. The PIT triggers the DMA if PIT counter becomes 0. PIT counter is loaded the LDVALn register.
3. DMA toggles the GPIO output signal. GPIO output signal can be toggled by accessing the PTOR register. This DMA is linked to the other DMA so this DMA will trigger the next DMA for loading new LDVALn register value.
4. The next DMA stores new value to LDVALn register. Then, the user makes the 2 words array for 2 PIT countdown values. In the odd number cycle, the first word is stored the LDVAL register, in the other number cycle, the second word is stored the LDVAL register.
5. Repeat from 2 to 4.

NOTE

Duty and frequency are determined by the 2 word array value stored in LDVAL register.

Figure below is the PWM procedure overview. PIT is used channel 0 (ch0), DMA is used ch0 and ch5. PIT chn will trigger the DMA chn. Kinetis has 4 PIT channels. So in this example, DMA ch5 is used for storing new PIT counter value.

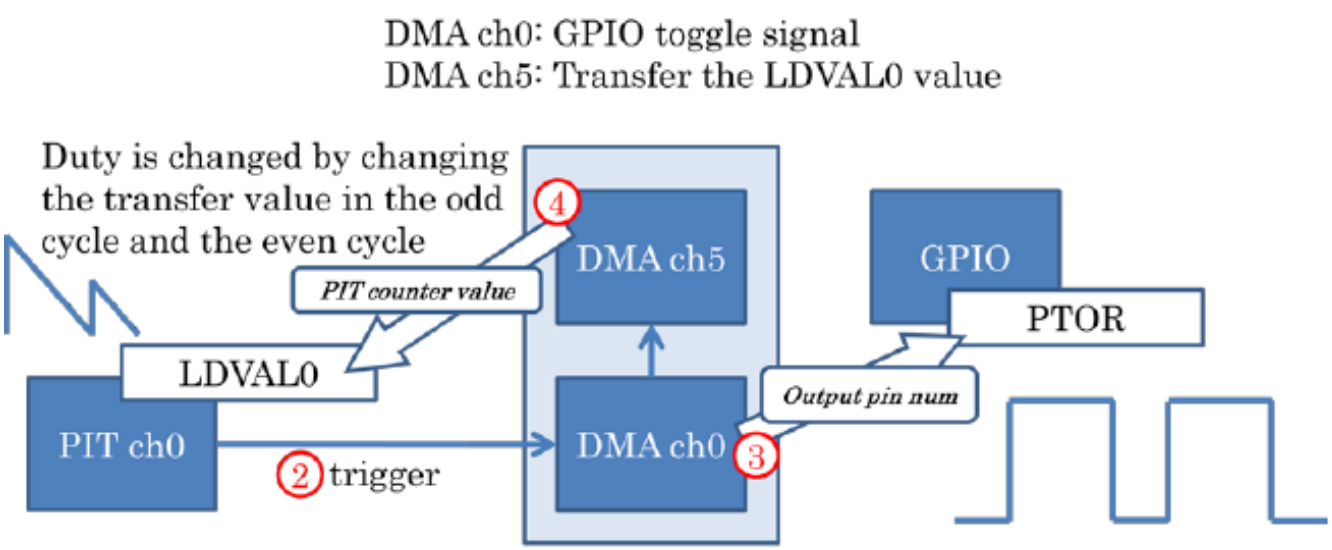

Figure 2. PWM procedure overview

Figure below is the relations of the counter value and the GPIO output signal. The upper wave means PIT counter value and the lower wave is the GPIO output. When the PIT counter value becomes 0, GPIO signal is toggled. Otherwise, PIT loads new counter value from the LDVAL and starts the countdown. DMA triggers the next DMA ch5, and DMA ch5 will set the new counter value to the LDVAL.

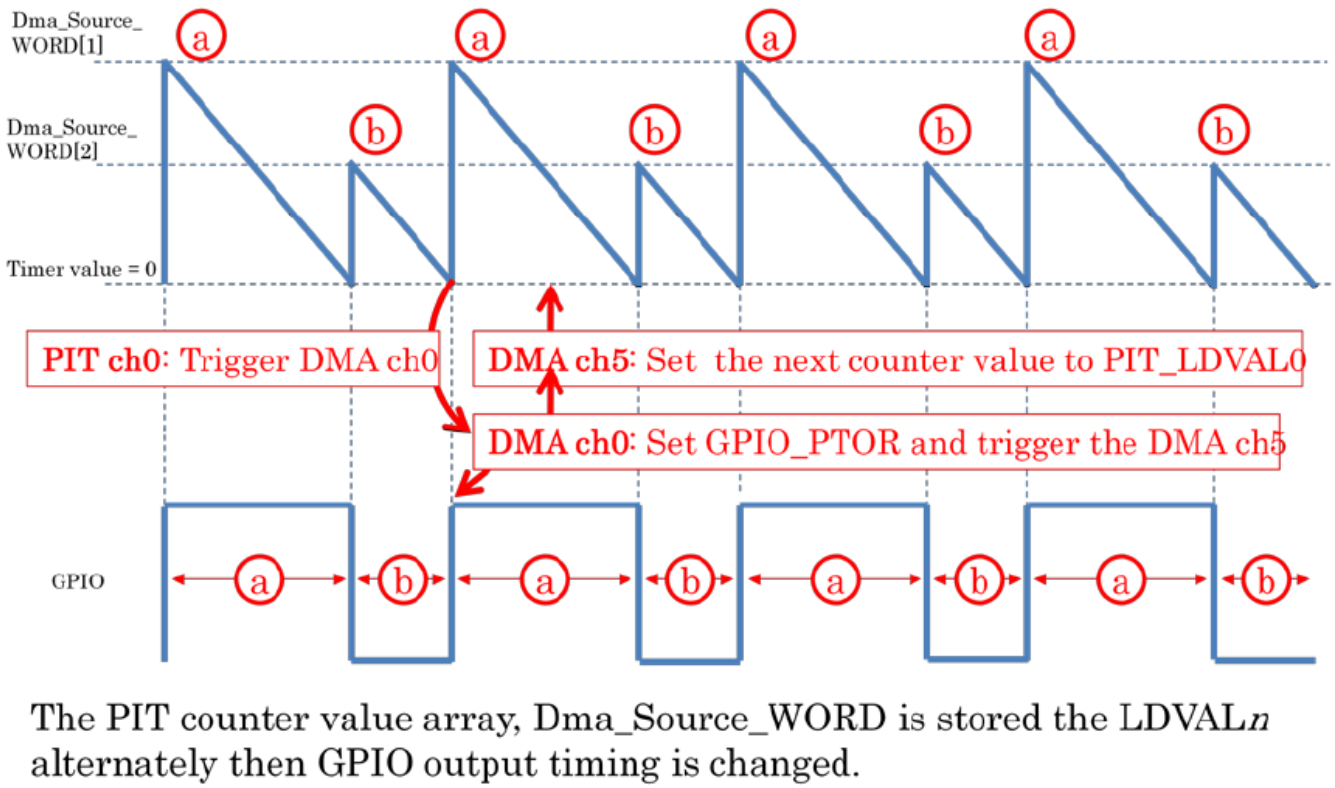

Figure 3. The relations of the counter value and the GPIO output signal

3 Limitation

These implementations have some limitations.

1. The PIT generates periodic trigger events to the DMA Mux as shown in the table below.

Table 1. PIT channel assignments for periodic DMA triggering

DMA Channel Number	PIT channel
DMA channel 0	PIT channel 0
DMA channel 1	PIT channel 1
DMA channel 2	PIT channel 2
DMA channel 3	PIT channel 3

2. There are some spike signals if PWM duty is set 0 or 100%.
3. The output GPIO pin number is 1 per each GPIO port. 2 output pins from 1 port is not accepted. Because GPIO can accept only 1 DMA trigger.
4. The maximum frequency is a half of the PIT frequency. However, up to 1.5MHz when CPU 100MHz, Bus 50MHz, PIT 50MHz.
5. The jitter: about 300nsec if you set the LDVAL less than 30.

Following are the recommendations when you use these functions.

1. For best performance, no other DMA transfers or writes to the GPIO/Peripheral Bridge 1 should be done. Also if the DMA is transferring from SRAMU, any CPU or other master RAM accesses should be limited to SRAML and vice versa.
2. If the above cannot be met, then before accesses like the above, software should check the PIT counter register to ensure that the pending DMA transfer is not scheduled during the desired access.

4 Examples

4.1 Environment

This application note uses the following device as the test environment.

- TWR-K40X256
- Core/System frequency: 100MHz
- Bus frequency: 50MHz
- PIT frequency: 50MHz
- IAR Embedded Workbench v6.10.1

  In the example code, GPIOD 7pin is used as the output pin.

4.2 Sample code

4.2.1 Clock

- void main()

  Set the frequency configuration and call some functions to initialize the GPIO, PIT and DMA. Finally start the PIT.

- init_gpio()

  Initialize the GPIO and set some configurations.

- void Set_Pit0(void)

  Initialize the PIT0.

- void dma_32bit()

  Initialize the DMA and set the configurations to toggle the GPIO output and update the LDVAL0.

Global array:

- Dma_Source_WORD[2]

  Element 0 takes the GPIO pin number value.

  Element 1 takes the GPIO pin number value.

```
void main (void)
{
volatile uint32 *tempptr;
```

```
printf("TWR-K40X256 GPIO Example!¥n");
/* Turn on all port clocks */
SIM_SCGC5 = SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTC_MASK |
SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;
/* Initialize GPIO on TWR-K40X256 */
init_gpio();
//PTD
Dma_Source_WORD= 0x00000080; //7pin toggle
// 1kHz PIT_LDVAL0
Dma_Source_WORD[1]= 0x00000957;
GPIOD_PTOR = 0x00000080; //PTD7 toggle
dma_32bit(); // DMA setting
Set_Pit0(); // PIT module enbale
PIT_LDVAL0 = 0x00000957;
PIT_TFLG0 = PIT_TFLG_TIF_MASK;
GPIOD_PSOR = 0x00000080; // PTD7 output
PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK;
while(1){
run_cmd();
}//While(1)
}//Main
/*
* Initialize GPIO
*/
void init_gpio()
{
//DMA outputs
PORTD_PCR7|=(0|PORT_PCR_MUX(1)|PORT_PCR_IRQC(0x1));
//Change PTD7 to outputs
GPIOD_PDDR|=GPIO_PDDR_PDD(GPIO_PIN(7));
}
void Set_Pit0(void)
{
SIM_SCGC6 |= SIM_SCGC6_PIT_MASK; // turn on PIT cloccks
PIT_MCR = 1; // reset MDIS -> enable the module
}
void dma_32bit (void)
{
volatile uint32 *tempptr;
SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;
//////////////////////////////////////////
//Set up DMA from PIT0
//////////////////////////////////////////
DMAMUX_CHCFG0|=DMAMUX_CHCFG_SOURCE(52); //PORTD
DMAMUX_CHCFG0 |= DMAMUX_CHCFG_ENBL_MASK|DMAMUX_CHCFG_TRIG_MASK;
tempptr = &Dma_Source_WORD[0];
tcd.saddr = (uint32_t)tempptr;
tcd.daddr = 0x400Ff0CC; //ADDRESS of PTD_PTOR
tcd.nbytes = 4;
tcd.tcdAttr = DMA_ATTR_SSIZE(2) | DMA_ATTR_DSIZE(2); //EDMA_TCD_ATTR_SSIZE_32BIT|
EDMA_TCD_ATTR_DSIZE_32BIT ;
tcd.soff = 0x0;
tcd.doff = 0x0;
tcd.slast = 0x0;
tcd.loopcount = 0x1;
tcd.dlast_sga = 0x0;
tcd.csr = 0x0;
tcd.channelno = 0;
dma_config(CONFIG_BASIC_XFR, &tcd);
DMA_ERQ|= 1;
}
```

4.2.2 PWM

Functions:

Examples

- void main()

  Set the frequency configuration and call some functions to initialize the GPIO, PIT and DMA. Finally start the PIT.

- init_gpio()

  Initialize the GPIO and set some configurations.

- void Set_Pit0(void)

  Initialize the PIT0.

- void dma_32bit()

  Initialize the DMA and set the configurations to toggle the GPIO output and update the LDVAL0.

Global array:

- Dma_Source_WORD[3]

Element 0 provides the value of the GPIO pin number. Element 1 and 2 provides the value of updating the LDVAL0. The value of the element 1 is the H level time and the value of the element 2 is the L level time.

```
void main (void)
{
volatile uint32 *tempptr;
printf("TWR-K40X256 GPIO Example!¥n");
/* Turn on all port clocks */
SIM_SCGC5 = SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTC_MASK |
SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;
/* Initialize GPIO on TWR-K40X256 */
init_gpio();
//PTD
Dma_Source_WORD[0]= 0x00000080; //7pin toggle
// 1kHz duty 50% PIT_LDVAL0
Dma_Source_WORD[1]= 0x00000957;
Dma_Source_WORD[2]= 0x00000957;
GPIOD_PTOR = 0x00000080; //PTD7 toggle
dma_32bit(); // DMA setting
Set_Pit0(); // PIT module enbale
PIT_LDVAL0 = 0x00000957;
PIT_TFLG0 = PIT_TFLG_TIF_MASK;
GPIOD_PSOR = 0x00000080; // PTD7 output
PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK;
while(1){
run_cmd();
}//While(1)
}//Main
/*
* Initialize GPIO
*/
void init_gpio()
{
//DMA outputs
PORTD_PCR7|=(0|PORT_PCR_MUX(1)|PORT_PCR_IRQC(0x1));
//Change PTD7 to outputs
GPIOD_PDDR|=GPIO_PDDR_PDD(GPIO_PIN(7));
}
void Set_Pit0(void)
{
SIM_SCGC6 |= SIM_SCGC6_PIT_MASK; // turn on PIT cloccks
PIT_MCR = 1; // reset MDIS -> enable the module
}
void dma_32bit (void)
{
volatile uint32 *tempptr;
SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;
////////////////////////////////////////
```

```
//Set up DMA from PIT0
////////////////////////////////////////
DMAMUX_CHCFG0|=DMAMUX_CHCFG_SOURCE(52); //PORTD
DMAMUX_CHCFG0 |= DMAMUX_CHCFG_ENBL_MASK|DMAMUX_CHCFG_TRIG_MASK;
tempptr = &Dma_Source_WORD[0];
tcd.saddr = (uint32_t)tempptr;
tcd.daddr = 0x400Ff0CC; //ADDRESS of PTD_PTOR
tcd.nbytes = 4;
tcd.tcdAttr = DMA_ATTR_SSIZE(2) | DMA_ATTR_DSIZE(2); //EDMA_TCD_ATTR_SSIZE_32BIT|
EDMA_TCD_ATTR_DSIZE_32BIT ;
tcd.soff = 0x0;
tcd.doff = 0x0;
tcd.slast = 0x0;
tcd.loopcount = 0x1;
tcd.dlast_sga = 0x0;
tcd.csr = 0x0520;
tcd.channelno = 0;
dma_config(CONFIG_BASIC_XFR, &tcd);
DMA_ERQ|= 1;
tempptr = &Dma_Source_WORD[1];
tcd.saddr = (uint32_t)tempptr;
tcd.daddr = 0x40037100; //ADDRESS of PIT_LDVAL0
tcd.nbytes = 4;
tcd.tcdAttr = DMA_ATTR_SSIZE(2) | DMA_ATTR_DSIZE(2);
//EDMA_TCD_ATTR_SSIZE_32BIT|EDMA_TCD_ATTR_DSIZE_32BIT ;
tcd.soff = 0x04;
tcd.doff = 0x0;
tcd.slast = -8;
tcd.loopcount = 0x2;
tcd.dlast_sga = 0x0;
tcd.csr = 0x0;
tcd.channelno = 5;
dma_config(CONFIG_BASIC_XFR, &tcd);
DMA_ERQ|= 0x1 << 1;
}
```

4.3 Result

We observed the waves of PTD7 and PTA16.

In the Figure below, each pin output the 1kHz clock.

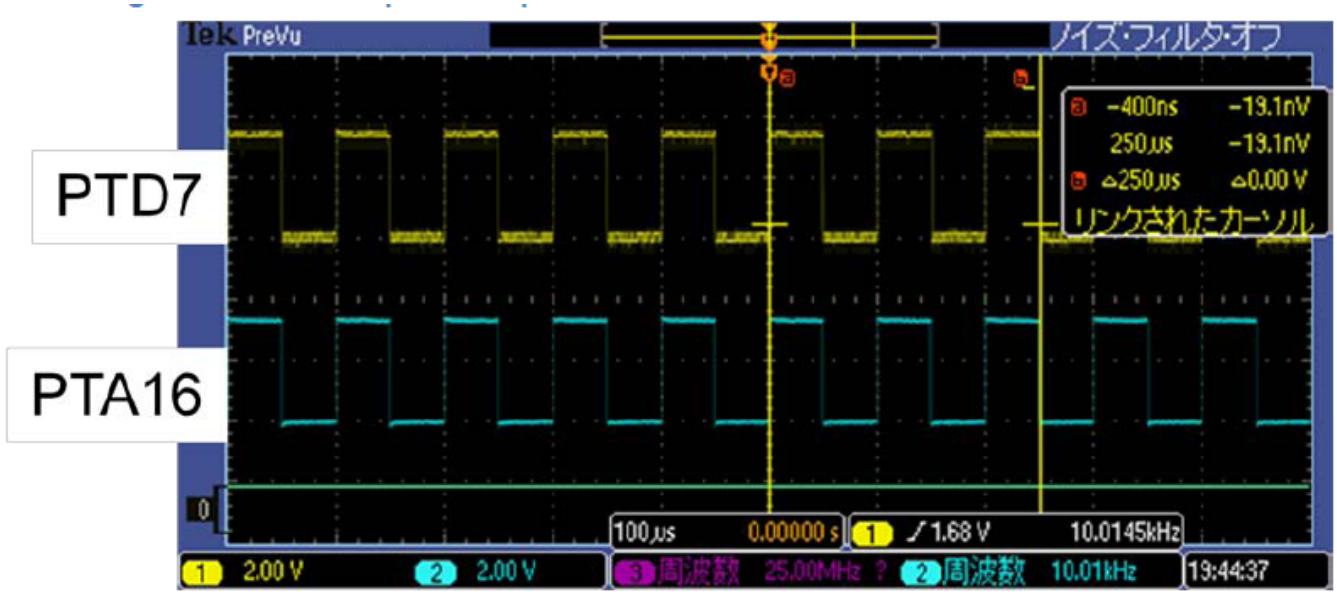

Figure 4. Output 1kHz clock

In the figure given below, PTD7 outputs 500Hz frequency and 75% duty. And PTA16 outputs 2kHz frequency and 50% duty.

Figure 5. PWM output

5 Conclusion

This application note provides information on how to use DMA and GPIO to emulate timer functionality on Kinetis Family devices. If you need more timer function, please refer to this application note. The source code provided along with this application note (AN4419SW) can be used as a basis for your configurations. For more information on the Freescale Kinetis family, please see

http://www.freescale.com/webapp/sps/site/homepage.jsp?code=KINETIS&tid=VANKINETIS

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4419
Rev. 0, 01/2012