Chapter 11

Direct Memory Access (DMA)

Some devices in the MSP430 family support a multi-channel Direct Memory Address (DMA) controller that can move data from one location to another, without CPU intervention. This increases the throughput of peripheral modules and also allows the CPU to remain in a low-power mode, without needing to wake up to perform the data transfer. This gives the benefit of reduced power consumption. Data transfers to/from peripherals can be initiated by external and internal events, using triggers.

This chapter covers DMA operation, supported addressing and transfer modes, trigger selection, channel priorities and DMA controller interrupts.

Topic **Page**

11.1 Direct Memory Access (DMA) capability

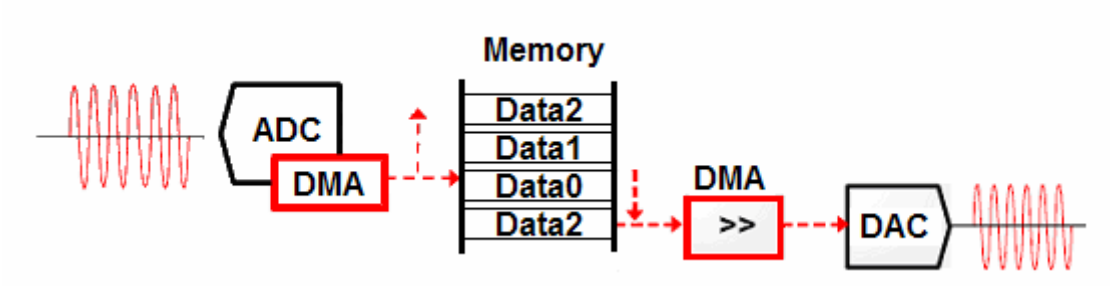
The MSP430 is well suited to low-power applications, and DMA is a very useful facility to have in order to achieve this. The following devices in the MSP430 family support DMA: 5xxx, FG4xx(x), F261x, F16x(x) and F15x. The Experimenter's Board uses the MSP430FG4618.

When a low power application requires data handling, the direct memory access (DMA) capability automatically handles data without CPU intervention, lowering the power consumption because the CPU remains sleeping.

The objective of DMA is to move functionality from the CPU to peripherals (see *Figure 11-1*) because:

- ❑ Peripherals use less current than the CPU;
- ❑ Performing operations directly between peripherals allows the CPU to shut down, saving system power;
- ❑ "Intelligent" peripherals are the most capable, providing more opportunity for CPU shutoff;
- ❑ DMA can be enabled for repetitive data handling, increasing the throughput of peripheral modules;
- ❑ Minimal software requirements and CPU cycles.

Figure 11-1. DMA data handling example.



The TI webpage gives some application notes, which explain the use of the DMA controller for different applications, with the objective of reducing power consumption:

- ❑ Streamlining the mixed-signal path with the signal-chain-on-chip MSP430F169 <slyt078.pdf>
  - ■ An integrated signal chain contains a variable resistance that generates a voltage level sampled by the ADC. The conversion result is processed and used to determine the update rate of the DAC and consequently, the analogue output signal frequency. The DAC output frequency adjustment is made by interrupting the DMA instead of the CPU, freeing up CPU resources for other tasks.

- Interfacing the MSP430 with MMC/SD Flash Memory Cards <slaa281b.pdf>
  - The MSP430F161x microcontroller is used to communicate with an MMC or SD flash memory card via a serial peripheral interface (SPI). The DMA module is used for data transmission between the MSP430 and the MMC card, resulting in higher communication speed and less CPU load.

- Digital FIR Filter Design Using the MSP430F16x <slaa228.pdf>
  - A FIR filter is implemented using the MSP430F16x family of devices. The complete filter algorithm is executed by the 3-channel DMA peripheral and the hardware multiplier peripheral. The 3-channel DMA peripheral is used to handle the required data, coefficients and movement of results between the memory and the multiply-and-accumulate (MAC). This dramatically improves the efficiency of the computation of the real-time FIR filter algorithm running on-chip, without the intervention of the CPU.

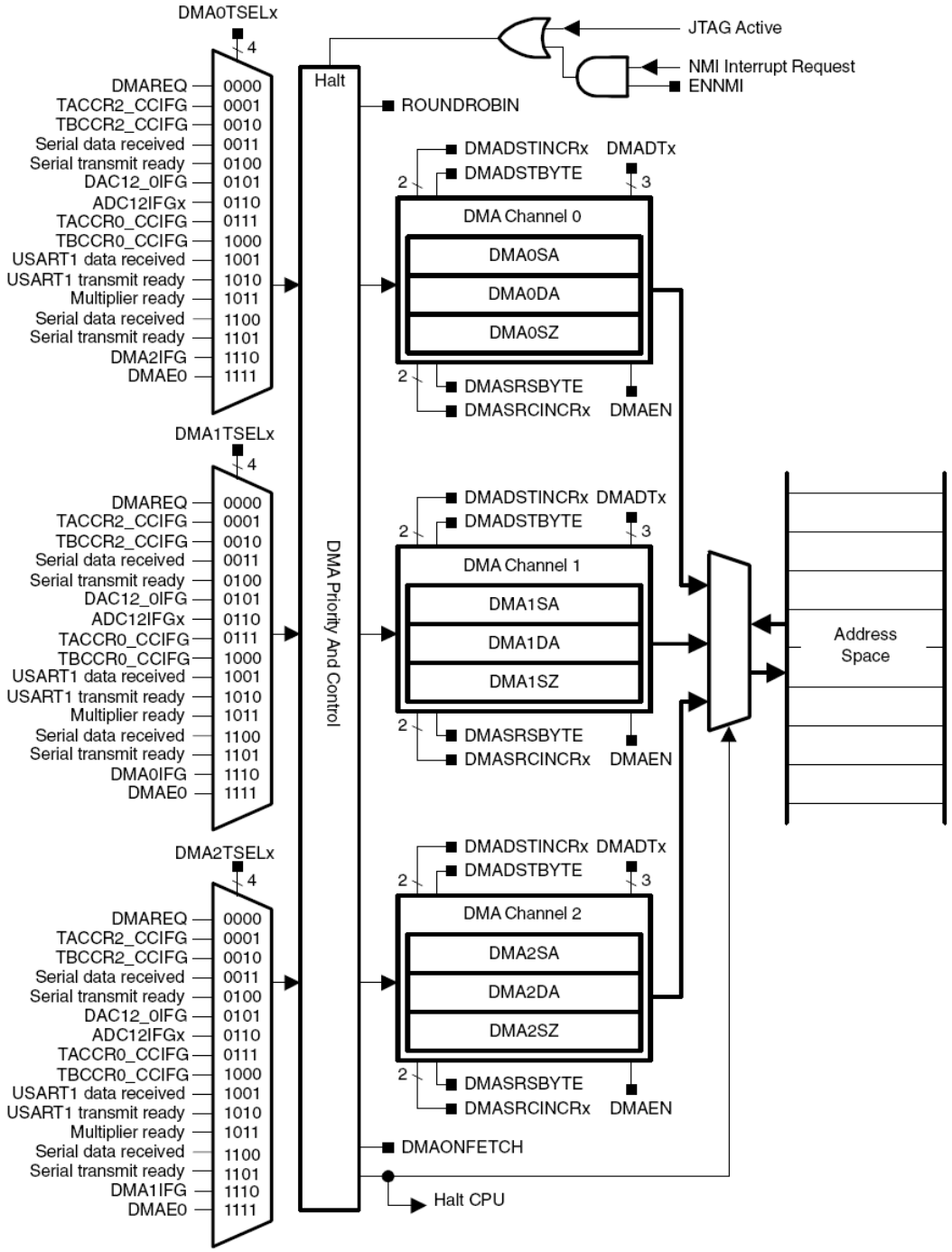
- Using the USCI I^2C Master <slaa382.pdf>
  - Use of the I^2C master function, for MSP430 devices with the USCI module. These functions can be used by the MSP430 master device to ensure proper initialization of the USCI module and provide I^2C transmit and receive functionality. The DMA module manages the loading of seven data bytes that need to be sent, because during the transmission, the CPU is in Low Power Mode 0.

11.2 DMA configuration and operation

The direct memory access (DMA) controller (see block diagram in *Figure 11-2*) allows movement of data from one memory address to another, across the entire address range, without CPU intervention.

Three DMA channels are implemented on the MSP430FG4618 device on the Experimenter's board.

Figure 11-2. DMA block diagram.



DMA controller features:

- Three independent transfer channels;
- Configurable (with the ROUNDROBIN bit) DMA channel priorities (default: DMA0–DMA1–DMA2);
- DMA Transfer cycle time:
  - Requires only two MCLK clock cycles per transfer;
  - Each byte/word transfer requires two MCLK cycles after synchronization, and one cycle of wait time after the transfer.
- Byte or word and mixed byte/word transfer capability:
  - Byte-to-byte;
  - Word-to-word;
  - Byte-to-word (upper byte of the destination word is cleared when the transfer occurs);
  - Word-to-byte (lower byte of the source word transfers).
- Block sizes up to 65535 bytes or words;
- Configurable selection of transfer trigger (see Table 11-1);

Table 11-1. DMA trigger modes.

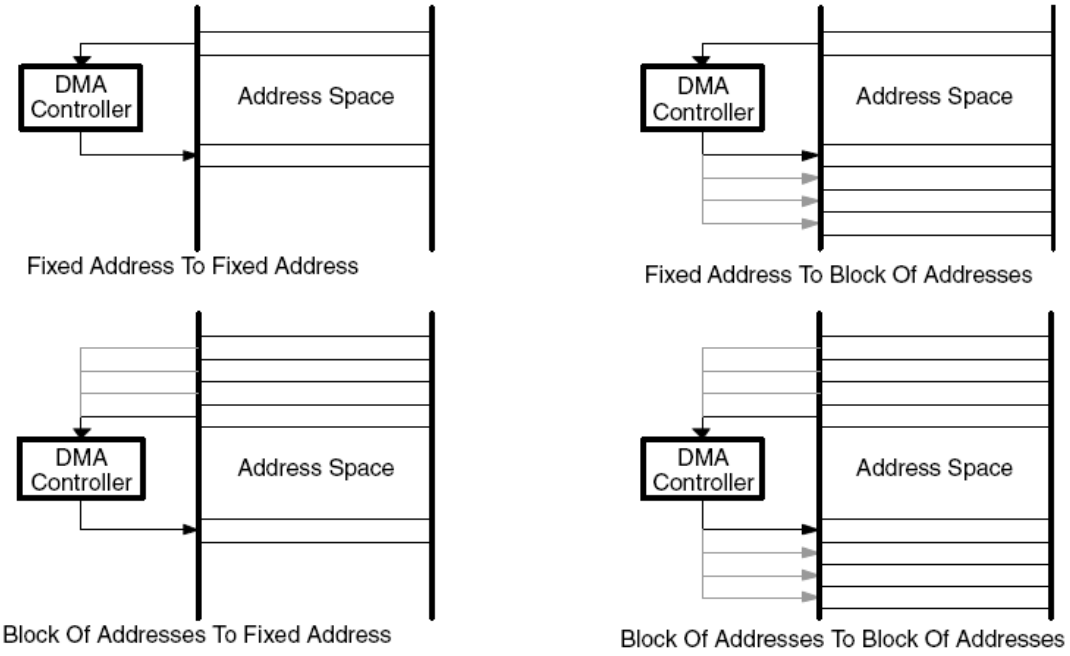
DMAxTSELx	Transfer triggered
0000	when DMAREQ = 1 (DMAREQ = 0 automatically when the transfer starts)
0001	<Timer_A> when TACCR2 CCIFG = 1 (CCIFG = 0 automatically when the transfer starts) If CCIE = 1, CCIFG does not trigger a transfer
0010	<Timer_B> when TBCCR2 CCIFG = 1 (CCIFG = 0 automatically when the transfer starts) If CCIE = 1, CCIFG does not trigger a transfer
0011	<USART0>: when URXIFG0 = 1 (URXIFG0 = 0 automatically when the transfer starts) If URXIE0 = 1, URXIFG0 flag does not trigger a transfer <USCI_A0>: when UCA0RXIFG = 1 (UCA0RXIFG = 0 automatically when the transfer starts) If UCA0RXIE = 1, UCA0RXIFG flag does not trigger a transfer
0100	<USART0>: when UTXIFG0 =1 (UTXIFG0 = 0 automatically when the transfer starts) If UTXIE0 = 1, UTXIFG0 flag does not trigger a transfer <USCI_A0>: when UCA0TXIFG = 1 (UCA0TXIFG = 0 automatically when the transfer starts) UCA0TXIE = 1, UCA0TXIFG flag does not trigger a transfer
0101	<DAC12> when DAC12_0CTL DAC12IFG = 1 (DAC12IFG = 0 automatically when the transfer starts) If DAC12IE = 1, DAC12IFG does not trigger a transfer

Table 11-1. DMA trigger modes (continued).

DMAxTSELx	Transfer triggered
0110	<ADC12> when ADC12IFGx = 1 (corresponding ADC12IFGx flag for single-channel conversions, and the ADC12IFGx for the last conversion for sequence conversions) (All ADC12IFGx = 0 automatically when the associated ADC12MEMx register is accessed by the DMA controller)
0111	<Timer_A> when TACCR0 CCIFG = 1: CCIFG = 0 automatically when the transfer starts If CCIE = 1, CCIFG flag does not trigger a transfer
1000	<Timer_B> when TBCCR0 CCIFG = 1 (CCIFG = 0 automatically when the transfer starts) If CCIE = 1, CCIFG does not trigger a transfer
1001	<USART1>: when URXIFG1 = 1 (URXIFG1 = 0 automatically when the transfer starts) If URXIE1 = 1, URXIFG0 flag does not trigger a transfer
1010	<USART1>: when UTXIFG1 =1 (UTXIFG1 = 0 automatically when the transfer starts) If UTXIE1 = 1, UTXIFG0 flag does not trigger a transfer
1011	<Hardware Multiplier> when the hardware multiplier is ready for a new operand
1100	<USCI_B0>: when UCB0RXIFG = 1 (UCB0RXIFG = 0 automatically when the transfer starts) If UCB0RXIE = 1, UCB0RXIFG flag does not trigger a transfer
1101	<USCI_B0>: when UCB0TXIFG = 1 (UCB0TXIFG = 0 automatically when the transfer starts) UCB0TXIE = 1, UCB0TXIFG flag does not trigger a transfer
1110	when the DMAxIFG = 1: DMA0IFG triggers channel 1 DMA1IFG triggers channel 2 DMA2IFG triggers channel 0 (None of the DMAxIFG = 0 automatically when the transfer starts)
1111	When an external trigger DMAE0 = 1

- Selectable edge or level-triggered transfer (DMALEVEL bit);
- Four addressing modes (see *Figure 11-3*) for each DMA channel independently configurable (DMASRCINCRx and DMADSTINCRx control bits):
  - Fixed address to fixed address;
  - Fixed address to block of addresses;
  - Block of addresses to fixed address;
  - Block of addresses to block of addresses.

Figure 11-3. DMA addressing modes.



- Six transfer modes. Each channel is individually configurable by the DMADTx bits (see *Table 11-2*).

Table 11-2. DMA transfer modes.

DMADTx	Transfer mode	Description	DMAEN after transfer
000	Single transfer	Each transfer requires a trigger	0
001	Block transfer	A complete block is transferred with one trigger	0
010, 011	Burst-block transfer	CPU activity is interleaved with a block transfer	0
100	Repeated single transfer	Each transfer requires a trigger	1
101	Repeated block transfer	A complete block is transferred with one trigger	1
110, 111	Repeated burst-block transfer	CPU activity is interleaved with a block transfer	1

System interrupts

DMA transfers are not interruptible by system interrupts, but system interrupt service routines (ISRs) may be interrupted by DMA transfers.

Only non-maskable interrupts (NMIs) can be configured to interrupt the DMA controller, if the ENNMI bit is set. If it is not set, system interrupts remain pending until the completion of the transfer.

DMA controller interrupts

Each DMA channel has its own DMAIFG flag, which is set when the corresponding DMAxSZ register counts to zero (all modes). If the corresponding DMAIE and GIE bits are set, an interrupt request is generated.

The MSP430FG4618 device implements the interrupt vector register DMAIV. In this case, all DMAIFG flags are prioritized and combined to source a single interrupt vector. The interrupt vector register DMAIV is used to determine which flag requested an interrupt.

- USCI_B I^2C module with DMA
  - Two trigger sources for the DMA controller;
  - Triggers a transfer when new I^2C data is received and when data is needed for transmit.

- ADC12 with DMA
  - Automatically moves data from any ADC12MEMx register to another location;

- DAC12 with DMA
  - Automatically moves data to the DAC12_xDAT register;

- Flash memory with the DMA
  - Automatically moves data to the flash memory;
  - Supports word/byte data transfers to the flash memory;
  - The write timing control is performed by the flash controller;
  - Write transfers to the flash memory succeed if the flash controller set-up is prior to the DMA transfer and if the flash is not busy.

All these DMA transfers occur without CPU intervention and independently of any low-power modes. This increases throughput of the modules and enhances low-power applications by allowing the CPU to remain off while data transfers occur.

11.3 DMA registers

The DMA controller registers are shown for the MSP430FG4618:

DMACTL0, DMA Control Register 0

15	14	13	12	11	10	9	8
Reserved				DMA2TSELx			

7	6	5	4	3	2	1	0
DMA1TSELx				DMA0TSELx			

See *Table 11-1*. All DMAxTSELx registers are the same.

DMACTL1, DMA Control Register 1

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
0	0	0	0	0	DMAONFETCH	ROUNDROBIN	ENNMI

Bit		Description
2	DMAONFETCH	DMA on fetch: DMAONFETCH = 0 ⇒ DMA transfer occurs immediately DMAONFETCH = 1 ⇒ DMA transfer occurs on next instruction fetch after the trigger
1	ROUNDROBIN	Round robin: ROUNDROBIN = 0 ⇒ DMA channel priority is DMA0 − DMA1 − DMA2 ROUNDROBIN = 1 ⇒ DMA channel priority changes with each transfer
0	ENNMI	Enable NMI when ENNMI = 1, allowing a NMI interrupt to interrupt a DMA transfer

DMAxCTL, DMA Channel x Control Register

15	14	13	12	11	10	9	8
Reserved	DMADTx			DMADSTINCRx		DMASRCINCRx	

7	6	5	4	3	2	1	0
DMADSTBYTE	DMASRCBYTE	DMALEVEL	DMAEN	DMAIFG	DMAIE	DMAABORT	DMAREQ

Bit		Description
14-12	DMADTx	DMA transfer mode: DMADT2 DMADT1 DMADT0 = 000 ⇒ Single transfer DMADT2 DMADT1 DMADT0 = 001 ⇒ Block transfer DMADT2 DMADT1 DMADT0 = 010 ⇒ Burst-block transfer DMADT2 DMADT1 DMADT0 = 011 ⇒ Burst-block transfer DMADT2 DMADT1 DMADT0 = 100 ⇒ Repeated single transfer DMADT2 DMADT1 DMADT0 = 101 ⇒ Repeated block transfer DMADT2 DMADT1 DMADT0 = 110 ⇒ Repeated burst-block transfer DMADT2 DMADT1 DMADT0 = 111 ⇒ Repeated burst-block transfer
11-10	DMADSTINCRx	DMA destination address increment/decrement after each byte or word transfer: When DMADSTBYTE = 1, the destination address increments / decrements by one When DMADSTBYTE = 0, the destination address increments / decrements by two. DMADSTINCR1 DMADSTINCR0 = 00 ⇒ Address unchanged DMADSTINCR1 DMADSTINCR0 = 01 ⇒ Address unchanged DMADSTINCR1 DMADSTINCR0 = 10 ⇒ Address decremented DMADSTINCR1 DMADSTINCR0 = 11 ⇒ Address increment
9-8	DMASRCINCRx	DMA source address increment/decrement after each byte or word transfer: When DMASRCBYTE = 1, the source address increments/decrements by one When DMASRCBYTE = 0, the source address increments/decrements by two. DMASRCINCR1 DMASRCINCR0 = 00 ⇒ Address unchanged DMASRCINCR1 DMASRCINCR0 = 01 ⇒ Address unchanged DMASRCINCR1 DMASRCINCR0 = 10 ⇒ Address decremented DMASRCINCR1 DMASRCINCR0 = 11 ⇒ Address increment
7	DMADSTBYTE	DMA destination length (byte or word): DMADSTBYTE = 0 ⇒ Word DMADSTBYTE = 1 ⇒ Byte
6	DMASRCBYTE	DMA source length (byte or word): DMASRCBYTE = 0 ⇒ Word DMASRCBYTE = 1 ⇒ Byte
5	DMALEVEL	DMA level: DMALEVEL = 0 ⇒ Edge sensitive trigger (rising edge) DMALEVEL = 1 ⇒ Level sensitive trigger (high level)
4	DMAEN	DMA enable when DMAEN = 1
3	DMAIFG	DMA interrupt flag DMAIFG = 1 when interrupt pending
2	DMAIE	DMA interrupt enable when DMAIE = 1
1	DMAABORT	DMA Abort DMAABORT = 1 when a DMA transfer is interrupted by NMI
0	DMAREQ	DMA request DMAREQ = 1 starts DMA

DMAxSA, DMA Source Address Register

The 32-bit DMAxSA register points to the DMA source address for single transfers or to the first source address for block transfers.

- Bits 31−20 are reserved and always read as zeros;
- Reading or writing bits 19-16 requires the use of extended instructions;
- When writing to DMAxSA with word instructions, bits 19-16 are cleared.

DMAxDA, DMA Destination Address Register

The 32-bit DMAxDA register points to the DMA destination address for single transfers or to the first source address for block transfers.

- Bits 31−20 are reserved and always read as zeros;
- Reading or writing bits 19-16 requires the use of extended instructions;
- When writing to DMAxDA with word instructions, bits 19-16 are cleared.

DMAxSZ, DMA Size Address Register

The 16-bit DMA size address register defines the number of byte/word data values per block transfer.

- DMAxSZ register decrements with each word or byte transfer;
- When DMAxSZ = 0, it is immediately and automatically reloaded with its previously initialized value.

DMAIV, DMA Interrupt Vector Register

The 16-bit DMA Interrupt Vector value only uses bits 3 to 1. The remaining bits are always read as zero.

The content of the DMAIV provides the priority of the interrupt source:

- DMAIV = 02h: DMA channel 0 (highest priority);
- DMAIV = 04h: DMA channel 1;
- DMAIV = 06h: DMA channel 2;

…

- DMAIV = 0Eh: Reserved (lowest priority);

11.4 Laboratory 7: Direct Memory Access

11.4.1 Lab7A: Data Memory transfer triggered by software

Project files

- C source files: **Chapter 11 > Lab7 > Lab7A_student.c**
- Solution files: **Chapter 11 > Lab7 > Lab7A_solution.c**

Overview

During this laboratory, the data transfer between two regions of memory is analyzed. The order of transfer is controlled by software.

A. Resources

The following resource is used in this laboratory:

- DMA controller.

B. Organization of the software application

The software begins by disabling the watchdog timer. Port P2.1 is set as an output with a logic low level.

The memory addresses of the data vectors are passed to the source data address DMA0SA and destination address DMA0DA registers.

The number of words to be transferred is loaded in the DMA0SZ (size) register.

The DMA channel 0 is configured so that the data transfer trigger is controlled by software, in order that after each transfer, the source and destination addresses are correctly incremented.

The application enters an infinite loop, where port P2.1 state is switched just before initiating the data transfer.

C. System configuration

- **DMA channel configuration:**

The source address and destination address of the data must be loaded into their respective registers:

```
DMA0SA = ____________________;
DMA0DA = ____________________;
```


To move a total of 32 words, what is the value to write to the data size register?

```
DMA0SZ = ____________________;
```

The DMA channel must be configured to transfer the word under software control. The source and destination addresses should be incremented immediately after each of the transfers.

```
DMA0CTL = ____________________;
```

D. Analysis of operation

In the **Memory** window, the addresses of data vector ***Tab_1*** and ***Tab_2*** addresses are displayed. The contents of these blocks must be identified in memory.

Add a breakpoint at line 60 of code, corresponding to the line of code that performs the switching of port P2.1 state.

Execute the application, and whenever the breakpoint is reached, the execution of the application will be suspended. Observe the data being gradually transferred from source to destination.

The data transfer is suspended once the 32 elements of the source data vector have been transferred.

MSP-EXP430FG4618 **SOLUTION**

Using the DMA controller included in the MSP-EXP430FG4618 Development Tool, analyse the data transfer between two regions of memory.

- DMA channel configuration:

```
// Start block address:
DMA0SA = (void (*)( )) &tab_1;
// Destination block address:
DMA0DA = (void (*)( )) &tab_2;

DMA0SZ = 0x0020;                                    // Block size

DMA0CTL=DMADT_0+DMASRCINCR_3+DMADSTINCR_3+DMAEN;
// Single transfer,
// DMA source and destination addresses increment,
// Enable DMA0
```


11.4.2 Lab7B: Sinusoidal waveform generator

Project files

- C source files: **Chapter 11 > Lab7 > Lab7B_student.c**
- Solution files: **Chapter 11 > Lab7 > Lab7B_solution.c**

Overview

This laboratory uses the DMA controller to automatically transfer data between data memory and the DAC12 data register. A sinusoidal waveform is produced at the output of the DAC, without CPU intervention.

A. Resources

This laboratory uses the following peripherals:

- DMA controller;
- DAC;
- ADC (reference generator: Vref+);
- Timer_A;
- Low power mode.

B. Organization of the software application

The successive samples needed to produce the sinusoidal waveform using the DAC are stored in the data vector ***Sin_tab***, which contains 32 points.

The software begins by disabling the watchdog timer, followed by activating the internal reference voltage Vref+. The source and destination registers of the data vector to be transferred by the DMA channel are loaded into the data vector ***Sin_tab*** (source) address and with the DAC12 data register (destination) address. There are 32 data values to be transferred.

The data transfer is initiated whenever the DAC12IFG flag is enabled. In this application, the DAC interrupt should be disabled.

The DMA controller is configured to operate in repeat mode, to transfer a word whenever the previous event occurs. The data source address is set to increment after each transfer, while the destination address must remain constant.

The timer is set to generate the PWM signal through the capture/compare unit TACCR1. SMCLK is the clock signal that counts up to the value in the TACCR0 register.

Finally, the settings and interrupts are enabled and the device enters into low power mode LPM0.

C. System configuration

- **DAC12 reference voltage activation:**

The DAC12 requires a reference voltage. One of the options is to use the internal voltage Vref+. Set the ADC12CTLO register to activate this voltage:

```
ADC12CTL0 =_____________________;
```

- **DMA Controller configuration:**

Configure the registers DMA0SA (source), DMA0DA (destination) and DMA0SZ (size) to transfer 32 words between the source vector ***Sin_tab*** and the DAC12_0DAT data destination register:

```
DMA0SA = _____________________;
DMA0DA = _____________________;
DMA0SZ = _____________________;
```

Configure the register DMACTL0 to provide a data transfer whenever the DAC12IFG flag is set:

```
DMACTL0 = _____________________;
```

Configure the register DMA0CTL to carry out a repeated simple data transfer, increasing the data source address:

```
DMA0CTL = _____________________;
```

- **Setup DAC12:**

The DAC12 will update its output whenever there is the activation of the signal TA1. The DAC full-scale should be 1x reference voltage. Choose a medium relationship between the DAC's current and average conversion speed:

```
DAC12_0CTL = _____________________;
```

- **Timer_A configuration:**

Timer_A is responsible for synchronizing data transfers between memory and the DAC12. The Timer_A input receives as the SMCLK signal (1.048576 MHz) and must have a 30 msec counting period.

What value needs to be written to TACCR0, in order to achieve this counting period:

```
TACCR0 = _____________________________;
TACTL =_____________________________;
```

The capture/compare unit TACCR1 should generate a PWM signal in set/reset mode. Configure the unit appropriately:

```
TACCTL1 = _____________________________;
TACCR1 = _____________________________;
```

D. Analysis of operation

The verification of this laboratory is achieved by using an oscilloscope probe to monitor the output of the DAC12 Channel 0, available on header 8 pin 6.

MSP-EXP430FG4618	**SOLUTION**
Using the DMA controller, which is included in the MSP-EXP430FG4618 Development Tool, transfer a sinusoidal waveform to the DAC.	

- DMA Controller configuration:

```
// Source block address:
DMA0SA = (void (*)( ))&Sin_tab;
// Destination single address:
DMA0DA = (void (*)( ))&DAC12_0DAT;
// Block size:
DMA0SZ = 0x20;

// DMA control register 0:
DMACTL0 = DMA0TSEL_5;                 // DAC12IFG trigger

DMA0CTL = DMADT_4 + DMASRCINCR_3 + DMAEN;
// Repeated single transfer,
// DMA source address increment,
// since DMASRCBYTE = 0, the source address increments by
// two (word-word)
```


- Setup DAC12:

```
DAC12_0CTL = DAC12LSEL_2 + DAC12IR + DAC12AMP_5 + DAC12IFG + DAC12ENC;
// Rising edge of Timer_A.OUT1 (TA1),
// DAC12 full-scale output: 1x reference voltage,
// Input and output buffers: Medium freq./current,
// Enable DAC12
```

- Timer_A configuration:

```
TACCR0 = 32-1;                  // Clock period of TACCR0
TACTL = TASSEL_2 + MC_1;        // SMCLK, contmode

TACCTL1 = OUTMOD_3;            // TACCR1 set/reset
TACCR1 = 20;                   // TACCR1 PWM Duty Cycle
```

11.5 Quiz

1. The DMA controller allows:

(a) Movement of data from one location to another without CPU intervention;

(b) An increase in throughput of peripheral modules;

(c) A reduction of system power consumption (CPU in low power mode);

(d) All of above.

2. The upper byte of a byte-to-word transfer:

(a) Retains the previous value;

(b) Is cleared;

(c) Is loaded with the same value as the lower byte;

(d) None of above.

3. When the DMA transfer mode DMAxTSELx = 1 is selected:

(a) Each transfer requires a trigger;

(b) A complete block is transferred with one trigger;

(c) CPU activity is interleaved with a block transfer;

(d) None of above.

4. NMI interrupts can interrupt the DMA controller when:

(a) GIE bit is set;

(b) DMAIE is set;

(c) ENNMI bit is set;

(d) DMAEN is set.

5. The DMA destination address when the DMADSTBYTE = 1 and DMADSTINCRx = 3:

(a) Remains unchanged;

(b) Increments by two;

(c) Decrements by two;

(d) Increments by one.

6. Starting with the default DMA channel priority (DMA0 – DMA1 – DMA2), if ROUNDROBIN is cleared after transferring DMA0, the next priority is:

(a) DMA2 – DMA0 – DMA1;

(b) DMA1 – DMA2 – DMA0;

(c) DMA0 – DMA1 – DMA2;

(d) DMA1 – DMA0 – DMA2.

Solution: 1. (d); 2. (b); 3. (b); 4. (c); 5. (d); 6. (c)

11.6 FAQs

1. Why is the program flow not correct when the DMA is used to write to flash memory?

A probable reason could be that the CPU is halted immediately while executing instructions and the transfer begins when a trigger is received. This condition occurs because DMAONFETCH = 0. Setting DMAONFETCH ensures that the CPU finishes the currently executing instruction, before being halted by the DMA controller.

