Direct Memory Access

Lecture 18

Josh Brake
Harvey Mudd College

Learning Objectives

By the end of this lecture you will be able to:

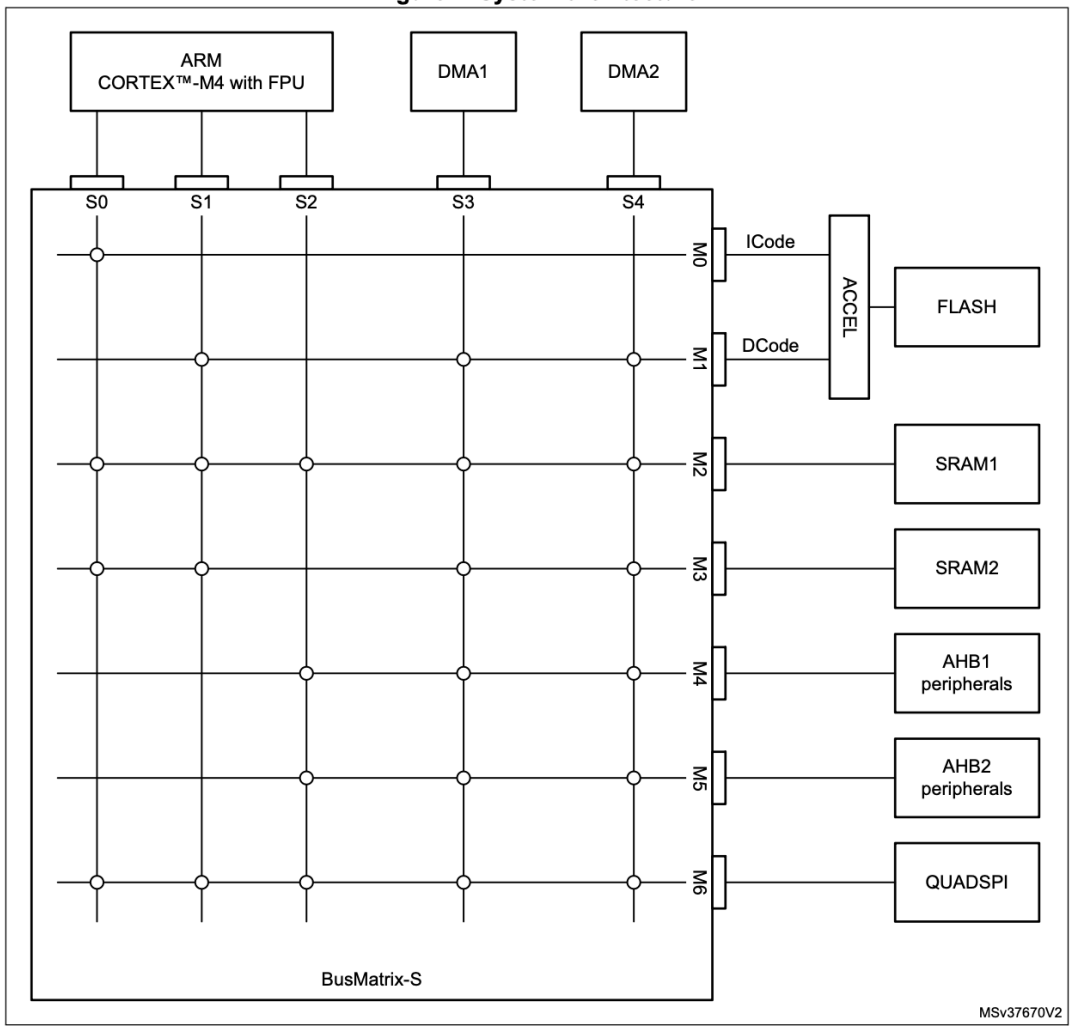
- Explain what direct memory access is used for and how it works
- Configure DMA on our MCU

Outline

- What is Direct Memory Access?
- How does DMA work on the MCU?
- Activity

Recall STM32L432KC System Architecture

Figure 1. System architecture



RM 0394 p. 64

Direct Memory Access

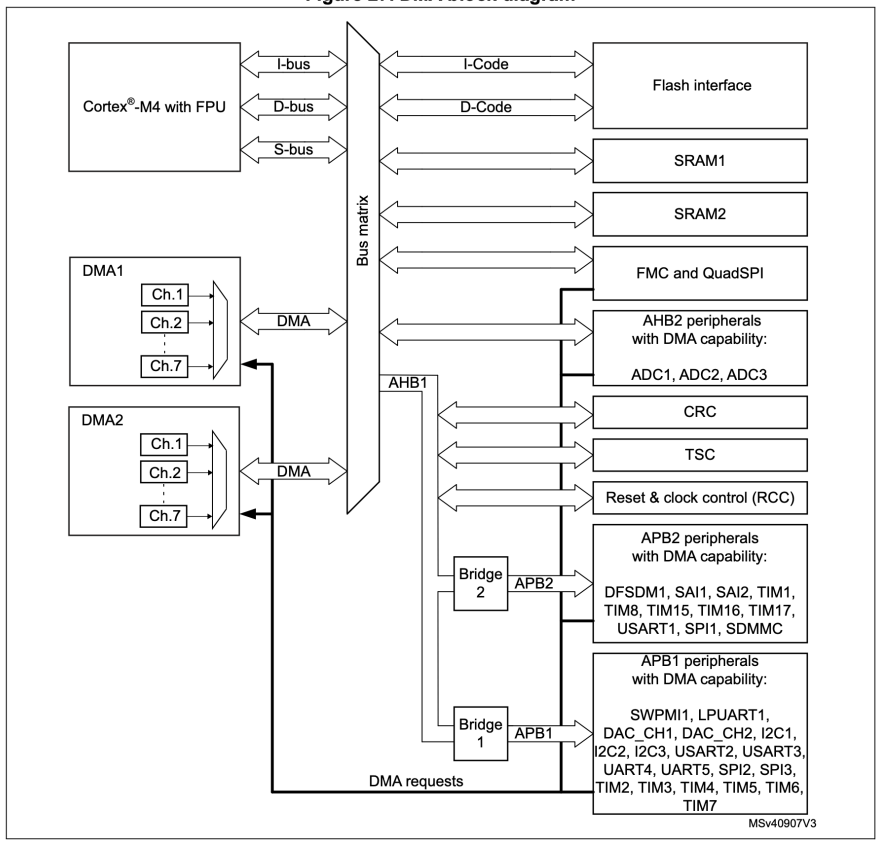
- Used to provide high-speed data transfer between peripherals and memory without CPU action
- DMA controller connects to the AHB bus
- The MCU has two DMA controllers with 7 channels each (total of 14 channels)
- Each channel has a mux which enables various DMA request sources to be selected

DMA Block Diagram

4 modes

1. Peripheral-to-memory
2. Peripheral-to-peripheral
3. Memory-to-peripheral
4. Memory-to-memory

Figure 27. DMA block diagram



RM 0394 p. 300

Activity

The goal of this example project is to use the direct memory access (DMA) controller on the MCU to enable automatic printing of data to the computer terminal via UART without the need for processor intervention. We will be designing a system to meet the following requirements.

Specifications

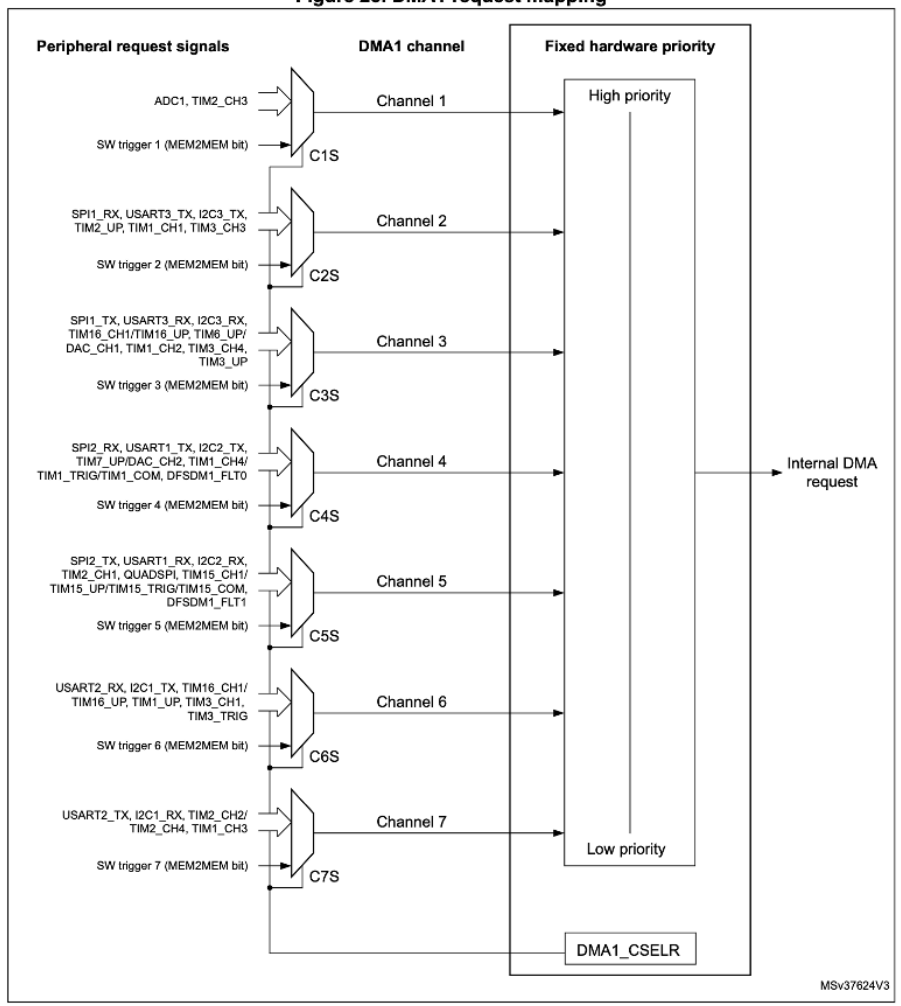
- Print out a single character from a specified character array at a frequency of ~10 Hz (one character every 100 milliseconds).
- Use a UART baud rate of 9600.
- Use update events from Timer 2 (TIM2) to trigger the DMA requests.
- Use DMA Controller 1 (DMA1) to handle the direct memory transfers from the character array to the UART peripheral.

Activity Steps and Hints

- Work through worksheet.
- After completing worksheet, create a new project and import demo source code.

DMA Channel Selection

Figure 25. DMA1 request mapping



RM0394 p. 297

DMA1 Request Mapping Mux Selects

Table 41. DMA1 requests for each channel

CxS[3:0]	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
0000	ADC1	ADC2(1)	-	-	DFSDM1_FLT0(2)	DFSDM1_FLT1(2)	-
0001	-	SPI1_RX	SPI1_TX	SPI2_RX(3)	SPI2_TX(3)	SAI2_A(4)	SAI2_B(4)

Table 41. DMA1 requests for each channel (continued)

CxS[3:0]	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
0010	-	USART3_TX (3)	USART3_RX (3)	USART1_TX	USART1_RX	USART2_RX	USART2_TX
0011	-	I2C3_TX	I2C3_RX	I2C2_TX(3)	I2C2_RX(3)	I2C1_TX	I2C1_RX
0100	TIM2_CH3	TIM2_UP	TIM16_CH1 TIM16_UP	-	TIM2_CH1	TIM16_CH1 TIM16_UP	TIM2_CH2 TIM2_CH4
0101	-	TIM3_CH3(2)	TIM3_CH4(2) TIM3_UP(2)	TIM7_UP. DAC_CH2(5)	QUADSPI	TIM3_CH1(2) TIM3_TRIG (2)	-
0110	-	-	TIM6_UP DAC_CH1	-	-	-	-
0111	-	TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM	TIM1_UP	TIM1_CH3

RM0394 p. 298

DMA Channel Configuration Register

11.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RM0394 p. 312

DMA Channel Selection Register

11.6.7 DMA channel selection register (DMA_CSELR)

Address offset: 0xA8

Reset value: 0x0000 0000

This register is used to manage the mapping of DMA channels as detailed in *Section 11.3.2: DMA request mapping*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	C7S[3:0]				C6S[3:0]				C5S[3:0]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C4S[3:0]				C3S[3:0]				C2S[3:0]				C1S[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RM0394 p. 317

DMA Channel – Number of Data to Transfer Register

11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx)

Address offset: 0x0C + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RM0394 p. 315

DMA Stream Peripheral Address Register

11.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RM0394 p. 315

DMA Channel - Memory Address Register

11.6.6 DMA channel x memory address register (DMA_CMARx)

Address offset: 0x14 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RM0394 p. 316

Summary

- DMA enables efficient and low-latency access between memory and peripherals
- Need to configure DMA controller, then configure DMA requests from the peripheral (timer, USART, SPI, etc.)
- Use interrupts to handle and reset flags as necessary

DMA Worksheet

Learning Goals

The goal of this example project is to use the direct memory access (DMA) controller on the MCU to enable automatic printing of data to the computer terminal via UART without the need for processor intervention. We will be designing a system to meet the following requirements.

Specifications

1. Print out a single character from a specified character array at a frequency of ~10 Hz (one character every 100 milliseconds).
2. Use a UART baud rate of 9600.
3. Use update events from Timer 2 (TIM2) to trigger the DMA requests.
4. Use DMA Controller 1 (DMA1) to handle the direct memory transfers from the character array to the UART peripheral.

Instructions

Answer the questions that follow.

In your answers, write down the field for each and the value it should be set to. Where applicable, use CMSIS notation of `<Peripheral>_<Register>_<Field>` to specify your answer.

For example, to configure PA2 as an output: `GPIO_MODER_MODE2, 0b10`.

Configure the DMA controller

1. Turn on the DMA controller in RCC. (Hint: Look in RCC registers).

```
1  RCC->AHB1ENR  |= (RCC_AHB1ENR_DMA1EN);
```

2. Find the correct DMA channel that is triggered by update events from Timer 2 (TIM2).

DMA Channel 2 (RM 0394 p. 299, Table 41)

3. Configure the DMA channel with the following settings.

Set the priority level to 2

```
1  _VAL2FLD(DMA_CCR_PL,0b10)
```

Turn on memory address incrementing

```
1  _VAL2FLD(DMA_CCR_MINC, 0b1)
```

Turn on circular addressing

```
1  _VAL2FLD(DMA_CCR_CIRC, 0b1)
```

Set the direction to be from memory to peripheral

```
1  _VAL2FLD(DMA_CCR_DIR, 0b1)
```

4. Set the DMA source memory address to be the address of the character array

```
1 // Source: Address of the character array buffer in memory.
2 DMA1_Channel2->CMAR = _VAL2FLD(DMA_CMAR_MA, (uint32_t) &CHAR_ARRAY);
```

5. Set the DMA data transfer length to be the length of the character array (set with `#define` macro to be `CHAR_ARRAY_SIZE`)

```
1 // Set DMA data transfer length (# of samples).
2 DMA1_Channel2->CNDTR  |= _VAL2FLD(DMA_CNDTR_NDT, CHAR_ARRAY_SIZE);
```

6. Set the DMA destination memory address to be the address of the USART transmission data register.

```
1 // Dest.: USART data register
2 DMA1_Channel2->CPAR = _VAL2FLD(DMA_CPAR_PA, (uint32_t) &(USART->TDR));
```

7. Set the channel selection mux to the appropriate setting to select updates from TIM2.

```
// Select 4th option for mux to channel 2
DMA1_CSELR->CSELR |= _VAL2FLD(DMA_CSELR_C2S, 4);
```

8. Enable the DMA channel

```
// Enable DMA1 channel.
DMA1_Channel2->CCR  |= DMA_CCR_EN;
```

Configure Timer

Set up timer to run at 10 Hz

1. Set prescaler register to 0 (`TIM_PSC_PSC`).

```
1  TIM->PSC  =  0x0000;
```

2. Set ARR to SystemCoreClock/10 (`SystemCoreClock` stores the current bus clock frequency in Hz).

```
1  TIM->ARR  =  SystemCoreClock/CHAR_PER_SECOND;
```

Set up the timers to generate DMA requests when an update event is triggered.

1. Configure DMA request to be generated when an update event is triggered instead of when a capture compare event occurs (TIMx_CR2).

```
// Enable trigger output on timer update events.
TIM->CR2 |= (TIM_CR2_CCDS); // Set DMA request when update event occurs
```

2. Enable DMA/Interrupt generation from update events (TIMx_DIER).

```
// Setup DMA request on update event for timer
TIM->DIER |= (TIM_DIER_UDE);
```

3. Enable the counter (TIMx_CR1).

```
// Start the timer.
TIM->CR1 |=  (TIM_CR1_CEN);
```