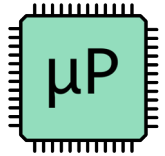
Direct Memory Access (DMA)

Lecture 20

Microprocessor-based Systems (E155)

Prof. Josh Brake

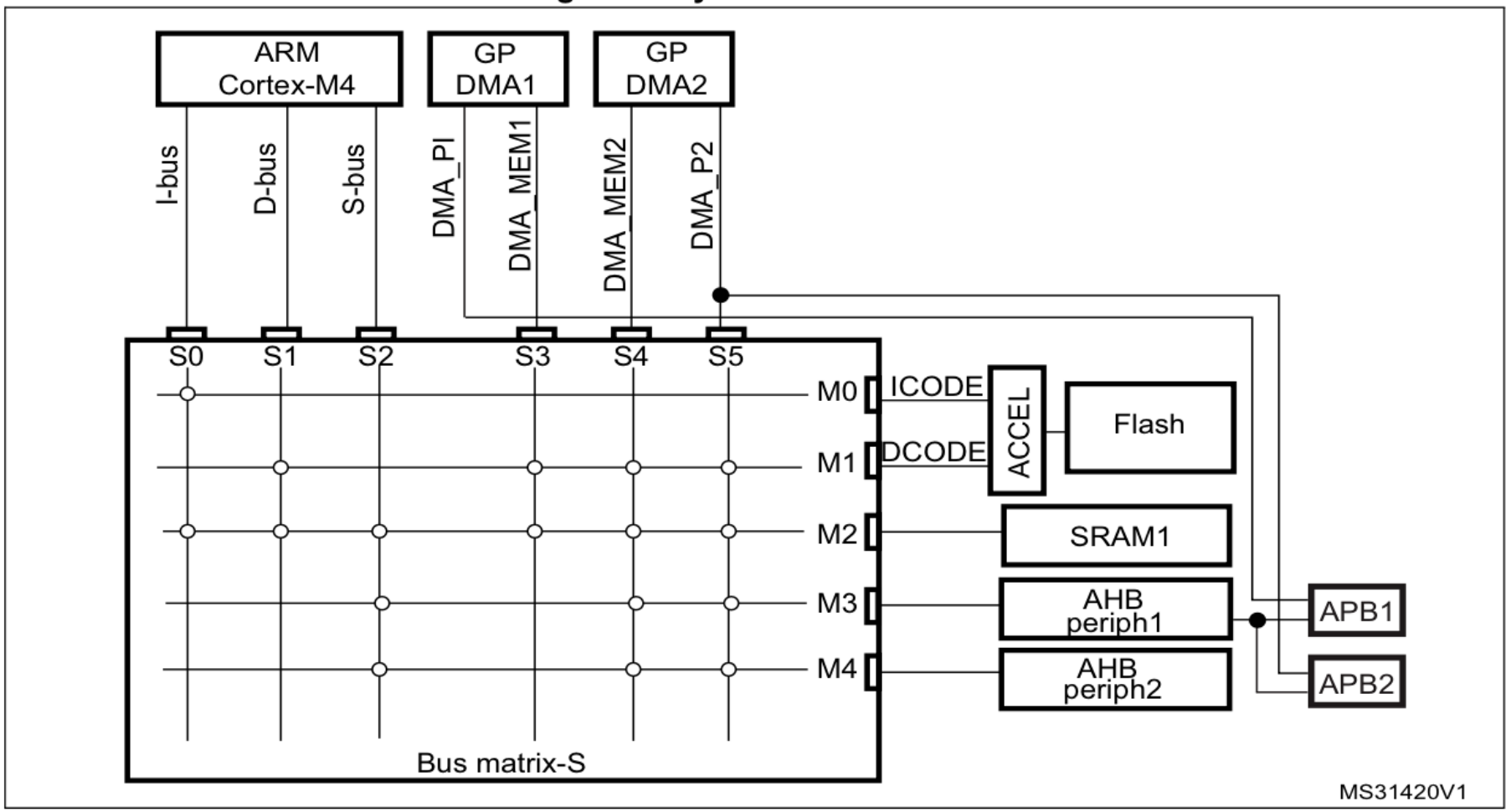


Outline

- What is Direct Memory Access?
- How does DMA work on the STM32F401RE?
- Bitfields with CMSIS
- Activity

Recall STM32F401RE System Architecture

Figure 1. System architecture



1. STM32F401xB/C: 128 KBytes / 256 KBytes Flash with 64 KBytes SRAM.
   STM32F401xD/E: 384 KBytes / 512KBytes Flash with 96 KBytes SRAM.

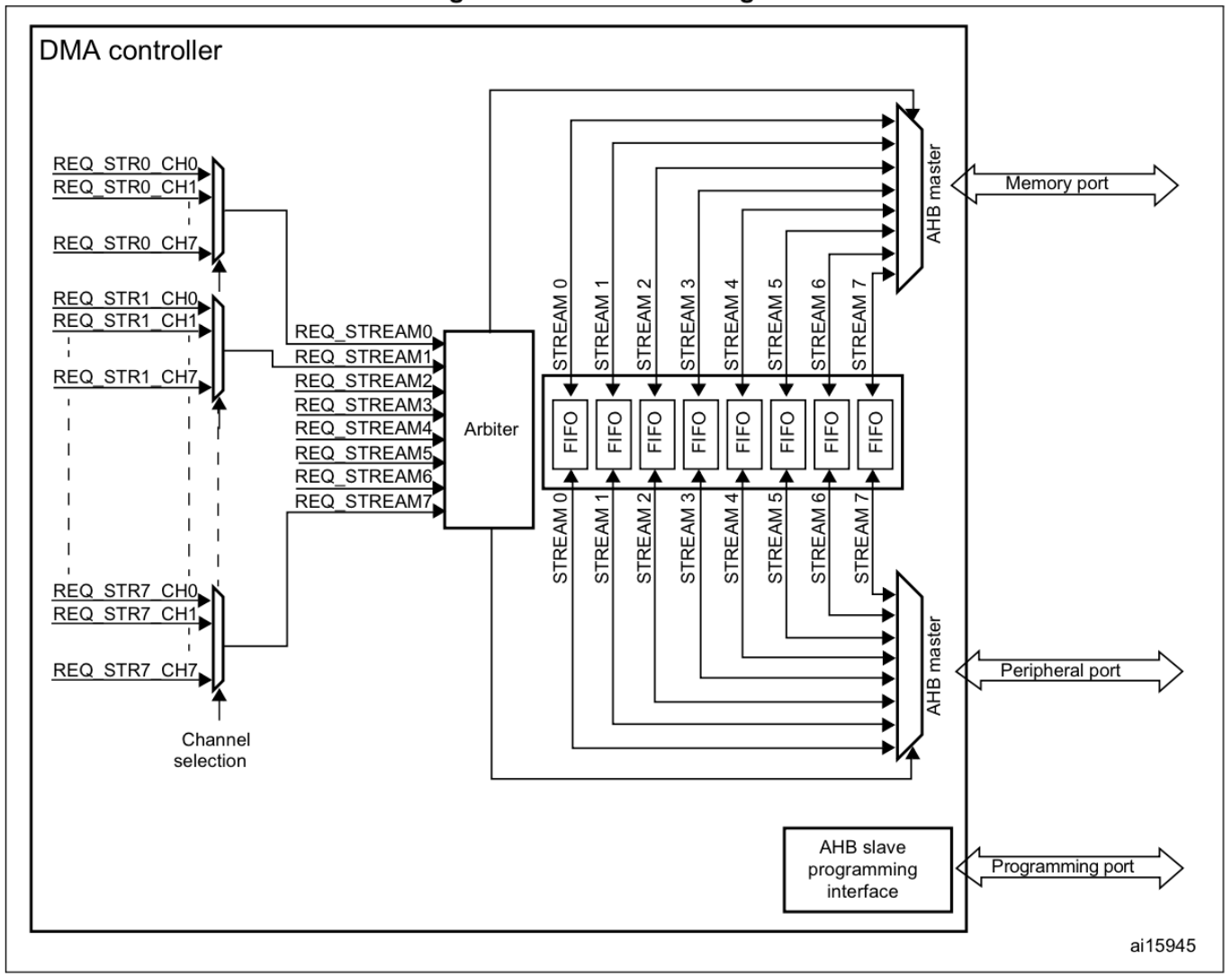
Direct Memory Access

- Used to provide high-speed data transfer between peripherals and memory without CPU action
- DMA controller connects to the AHB bus with an independent FIFO to optimize bandwidth
- STM32F401RE has two DMA controllers with 8 streams each (total of 16 streams)
- Each stream has 8 channels

DMA Block Diagram

- 3 modes
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory

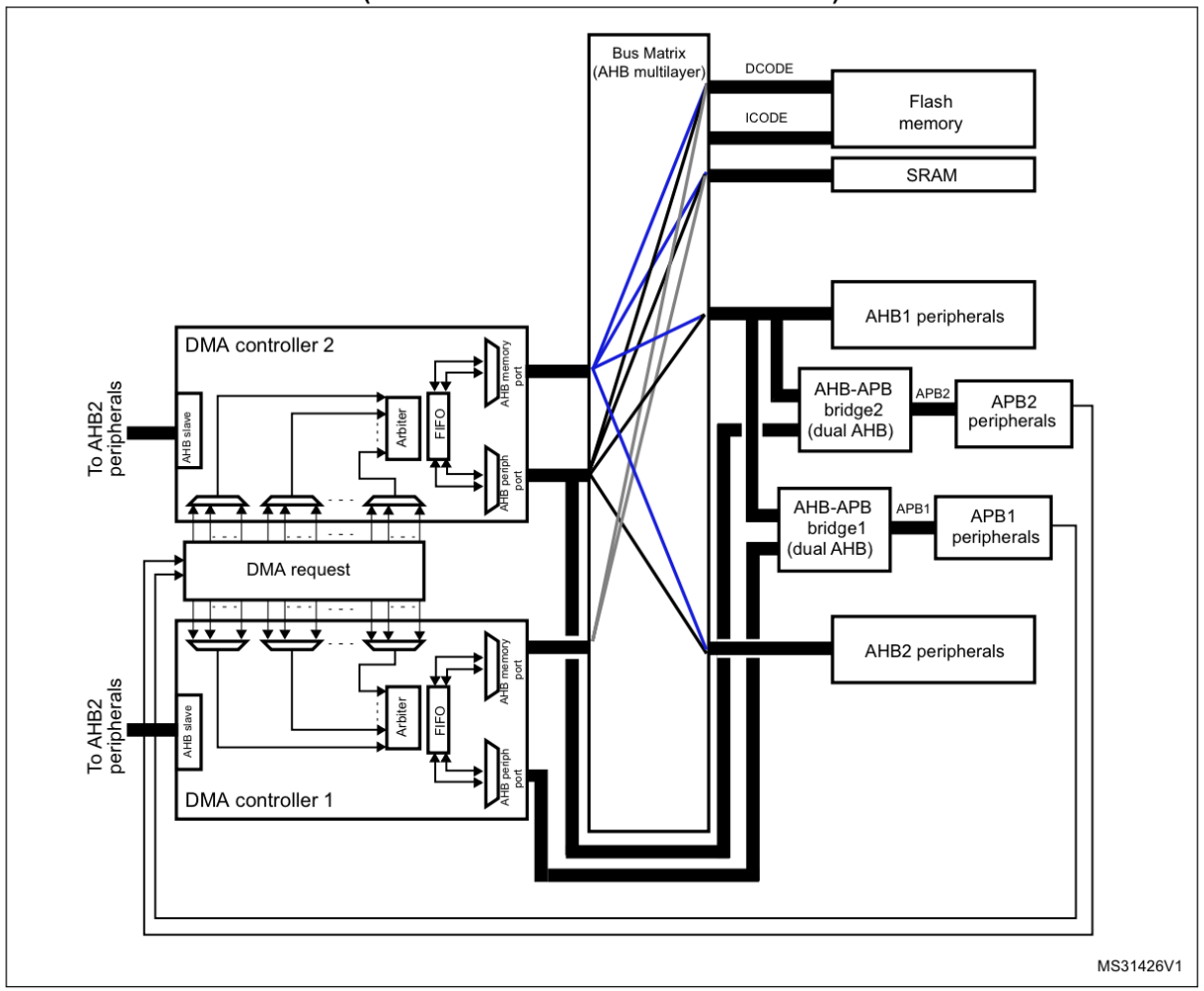
Figure 22. DMA block diagram



DMA block diagram Figure 22 p. 168 from STM32F401RE Reference Manual (PDF)

DMA Controller Implementation

Figure 23. System implementation of the two DMA controllers (STM32F401xB/C and STM32F401xD/E)



DMA block diagram Figure 23 p. 169 from STM32F401RE Reference Manual (PDF)

DMA Channel Selection

Figure 24. Channel selection

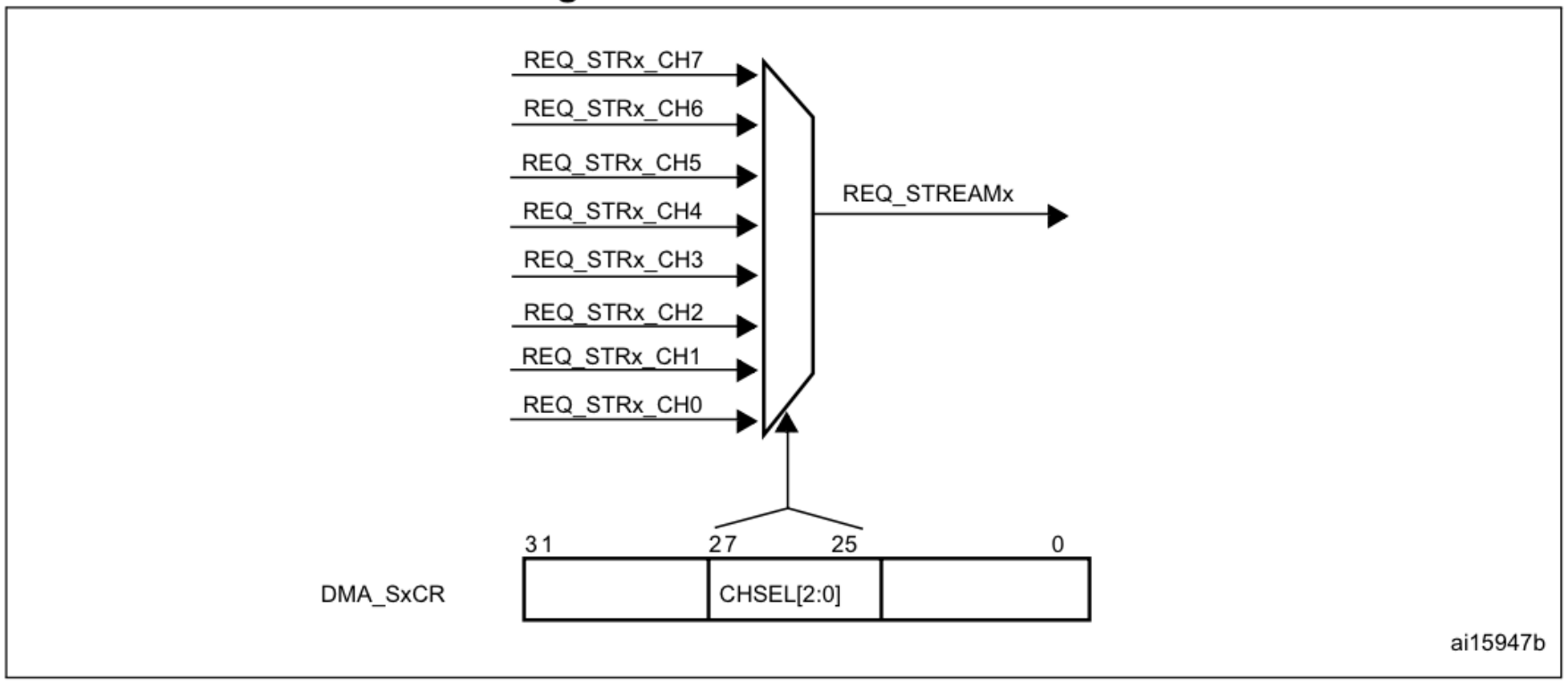


Figure 24 p. 170 from STM32F401RE Reference Manual

DMA1 Request Mapping Table

Table 28. DMA1 request mapping (STM32F401xB/C and STM32F401xD/E)

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	I2C3_RX	-	-	-	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	-	-	-	-	-	USART2_RX	USART2_TX	-
Channel 5	-	-	TIM3_CH4 TIM3_UP	-	TIM3_CH1 TIM3_TRIG	TIM3_CH2	-	TIM3_CH3

STM32F401RE Reference Manual pp. 170-171

DMA Stream Configuration Register

9.5.5 DMA stream x configuration register (DMA_SxCR) (x = 0..7)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[2:0]			MBURST [1:0]		PBURST[1:0]		Reser-ved	CT	DBM	PL[1:0]	
				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
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CMSIS Structures for Bitfields

- <x> - The name of the bitfield syntax is
  `<Peripheral Name>_<Register Name>_<Bitfield Name>`
- `<x>_Pos` - position in register of lsb of the bitfield
- `<x>_Msk` - masked out bits (1's shifted by <x>_Pos)
- `<x>_<N>` - mask of specific bit within bitfield

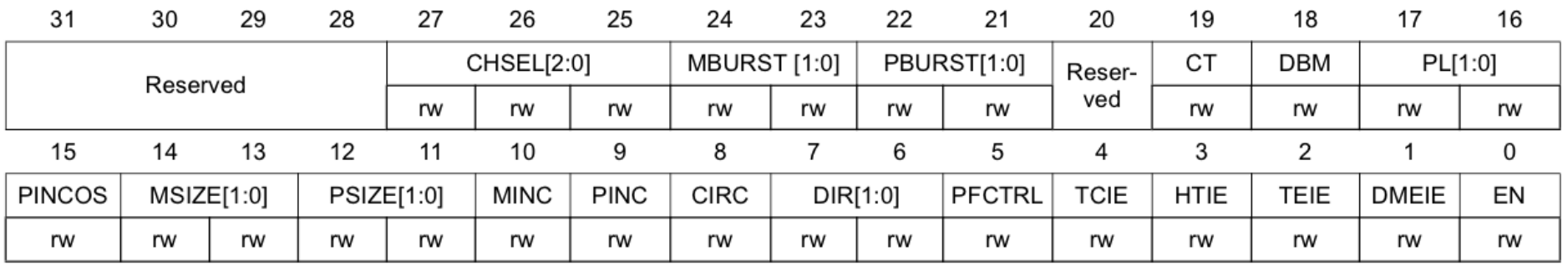
Examples from `stm32f401xe.h`

```
226   /**
227     * @brief DMA Controller
228     */
229
230   typedef struct
231   {
232     __IO uint32_t CR;     /*!< DMA stream x configuration register      */
233     __IO uint32_t NDTR;   /*!< DMA stream x number of data register     */
234     __IO uint32_t PAR;    /*!< DMA stream x peripheral address register */
235     __IO uint32_t M0AR;   /*!< DMA stream x memory 0 address register   */
236     __IO uint32_t M1AR;   /*!< DMA stream x memory 1 address register   */
237     __IO uint32_t FCR;    /*!< DMA stream x FIFO control register       */
238   } DMA_Stream_TypeDef;
```

Examples from `stm32f401xe.h`

```
805   #define DMA1                ((DMA_TypeDef *) DMA1_BASE)
806   #define DMA1_Stream0        ((DMA_Stream_TypeDef *) DMA1_Stream0_BASE)
807   #define DMA1_Stream1        ((DMA_Stream_TypeDef *) DMA1_Stream1_BASE)
808   #define DMA1_Stream2        ((DMA_Stream_TypeDef *) DMA1_Stream2_BASE)
809   #define DMA1_Stream3        ((DMA_Stream_TypeDef *) DMA1_Stream3_BASE)
810   #define DMA1_Stream4        ((DMA_Stream_TypeDef *) DMA1_Stream4_BASE)
811   #define DMA1_Stream5        ((DMA_Stream_TypeDef *) DMA1_Stream5_BASE)
812   #define DMA1_Stream6        ((DMA_Stream_TypeDef *) DMA1_Stream6_BASE)
813   #define DMA1_Stream7        ((DMA_Stream_TypeDef *) DMA1_Stream7_BASE)
814   #define DMA2                ((DMA_TypeDef *) DMA2_BASE)
815   #define DMA2_Stream0        ((DMA_Stream_TypeDef *) DMA2_Stream0_BASE)
816   #define DMA2_Stream1        ((DMA_Stream_TypeDef *) DMA2_Stream1_BASE)
817   #define DMA2_Stream2        ((DMA_Stream_TypeDef *) DMA2_Stream2_BASE)
818   #define DMA2_Stream3        ((DMA_Stream_TypeDef *) DMA2_Stream3_BASE)
819   #define DMA2_Stream4        ((DMA_Stream_TypeDef *) DMA2_Stream4_BASE)
820   #define DMA2_Stream5        ((DMA_Stream_TypeDef *) DMA2_Stream5_BASE)
821   #define DMA2_Stream6        ((DMA_Stream_TypeDef *) DMA2_Stream6_BASE)
822   #define DMA2_Stream7        ((DMA_Stream_TypeDef *) DMA2_Stream7_BASE)
```

Examples from `stm32f401xe.h`: DMA_SxCR



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[2:0]			MBURST [1:0]		PBURST[1:0]		Reser-ved	CT	DBM	PL[1:0]	
				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
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

```
1428    #define DMA_SxCR_CHSEL_Pos       (25U)
1429    #define DMA_SxCR_CHSEL_Msk       (0x7UL << DMA_SxCR_CHSEL_Pos)          /*!< 0x0E000000 */
1430    #define DMA_SxCR_CHSEL           DMA_SxCR_CHSEL_Msk
1431    #define DMA_SxCR_CHSEL_0         0x02000000U
1432    #define DMA_SxCR_CHSEL_1         0x04000000U
1433    #define DMA_SxCR_CHSEL_2         0x08000000U
1434    #define DMA_SxCR_MBURST_Pos      (23U)
1435    #define DMA_SxCR_MBURST_Msk      (0x3UL << DMA_SxCR_MBURST_Pos)         /*!< 0x01800000 */
1436    #define DMA_SxCR_MBURST          DMA_SxCR_MBURST_Msk
1437    #define DMA_SxCR_MBURST_0        (0x1UL << DMA_SxCR_MBURST_Pos)         /*!< 0x00800000 */
1438    #define DMA_SxCR_MBURST_1        (0x2UL << DMA_SxCR_MBURST_Pos)         /*!< 0x01000000 */
```

DMA Stream – Number of Data Register

9.5.6 DMA stream x number of data register (DMA_SxNDTR) (x = 0..7)

Address offset: 0x14 + 0x18 × *stream number*

Reset value: 0x0000 0000

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

DMA Stream Peripheral Address Register

9.5.7 DMA stream x peripheral address register (DMA_SxPAR) (x = 0..7)

Address offset: 0x18 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAR[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
PAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PAR[31:0]**: Peripheral address

Base address of the peripheral data register from/to which the data will be read/written.

These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

DMA Stream - Memory address register

9.5.8 DMA stream x memory 0 address register (DMA_SxM0AR) (x = 0..7)

Address offset: 0x1C + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[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
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M0A[31:0]**: Memory 0 address

Base address of Memory area 0 from/to which the data will be read/written.

These bits are write-protected. They can be written only if:

- the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '1' in the DMA_SxCR register (in Double buffer mode).

Activity

- Configure the USART peripheral to send data using DMA. Send an array of characters every second to the terminal.
- Set up DMA transfer to see mystery ASCII art printed to the terminal
- Create new CMSIS project for STM32F401RE
- Download and run demo code (DMA_USART_CHAR_main.c) from GitHub (put in src)

https://github.com/joshbrake/E155_FA2020/tree/master/L20

Demo Code

```c
// Standard library includes.
#include <stdint.h>
#include <stdlib.h>
// Vendor-provided device header file.
#include "stm32f4xx.h"

#define TIM TIM2
#define USART USART2
// 32-character array
const size_t CHAR_ARRAY_SIZE = 20;
const uint8_t CHAR_ARRAY[20] = "This is a DMA Test!\n";
```

```c
int main(void) {
    // Enable peripherals: GPIOA, DMA, TIM2.
    RCC->AHB1ENR  |= (RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_DMA1EN);
    RCC->APB1ENR  |= (RCC_APB1ENR_TIM2EN);

    // Configure USART2
    RCC->APB1ENR |= (RCC_APB1ENR_USART2EN); // Set USART2EN
    // Set PA2 to ALT function
    GPIOA->MODER &= ~(GPIO_MODER_MODER2);
    GPIOA->MODER |= (0b10 << GPIO_MODER_MODER2_Pos);
    // Configure pin modes as ALT function
    GPIOA->AFR[0] &= ~(GPIO_AFRL_AFSEL2 | GPIO_AFRL_AFSEL3);
    // Configure correct alternate functions (AF07)
    GPIOA->AFR[0] |= (0b0111 << GPIO_AFRL_AFSEL2_Pos | 0b0111 << GPIO_AFRL_AFSEL3_Pos);

    USART->CR1 |= (USART_CR1_UE); // Enable USART
    USART->CR1 &= ~(USART_CR1_M); // M=0 corresponds to 8 data bits
    USART->CR2 &= ~(USART_CR2_STOP); // 0b00 corresponds to 1 stop bit
    USART->CR1 &= ~(USART_CR1_OVER8); // Set to 16 times sampling freq

    USART->BRR |= (8 << USART_BRR_DIV_Mantissa_Pos);
    USART->BRR |= (11 << USART_BRR_DIV_Fraction_Pos); // 11/16

    USART->CR1 |= (USART_CR1_TE); // Enable USART2
```

```
    // DMA1 configuration (channel 3 / stream 1).
    // SxCR register:
    // - Memory-to-peripheral
    // - Circular mode enabled.
    // - Increment memory ptr, don't increment periph ptr.
    // - 8-bit data size for both source and destination.
    // - High priority (2/3).

    // Reset DMA1 Stream 1
    DMA1_Stream1->CR &= ~( DMA_SxCR_CHSEL |
                           DMA_SxCR_PL |
                           DMA_SxCR_MSIZE |
                           DMA_SxCR_PSIZE |
                           DMA_SxCR_PINC |
                           DMA_SxCR_EN );
    // Set up DMA1 Stream 5
    DMA1_Stream1->CR |=  ( ( 0x2 << DMA_SxCR_PL_Pos ) |
                           ( 0x0 << DMA_SxCR_MSIZE_Pos ) |
                           ( 0x0 << DMA_SxCR_PSIZE_Pos ) |
                           ( 0x3 << DMA_SxCR_CHSEL_Pos ) |
                           DMA_SxCR_MINC |
                           DMA_SxCR_CIRC |
                           ( 0x1 << DMA_SxCR_DIR_Pos ) );

    // Set DMA source and destination addresses.
    // Source: Address of the character array buffer in memory.
    DMA1_Stream1->M0AR  = (uint32_t) &CHAR_ARRAY;
    // Dest.: USART data register
    DMA1_Stream1->PAR   = (uint32_t) &(USART->DR);
    // Set DMA data transfer length (# of samples).
    DMA1_Stream1->NDTR  = (uint16_t) CHAR_ARRAY_SIZE;

    // Enable DMA1 Stream 1.
    DMA1_Stream1->CR   |= ( DMA_SxCR_EN_Msk );

    // TIM2 configuration.
    // Set prescaler and autoreload to issue DMA request at 10 Hz
    TIM->PSC  =  0x0000;
    TIM->ARR  =  SystemCoreClock/10;

    // Enable trigger output on timer update events.
    TIM->CR2 &= ~(TIM_CR2_MMS);
    TIM->CR2 |=  ( 0x2 << TIM_CR2_MMS_Pos);
    TIM->CR2 |= (TIM_CR2_CCDS); // Set DMA request when update event occurs

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

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

    while (1) {
    }
}
```

Activity Steps and Hints

- Use DMA_USART_main_demo.c
- Enable USART2 (connected to serial port via ST-LINK)
- Configure DMA
  - Figure out which DMA controller, stream, and channel you need to use.
  - Configure the stream
    - Set control register
    - Set memory source address
    - Set peripheral destination address
    - Set number of data elements to be transferred
  - Enable the DMA stream
- Configure timer to trigger DMA transactions on update event
  - Set up timer interrupt to reset DMA for next transaction

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

Lecture Feedback

- What is the most important thing you learned in class today?
- What point was most unclear from lecture today?

https://forms.gle/Ay6MkpZ6x3xsW2Eb8