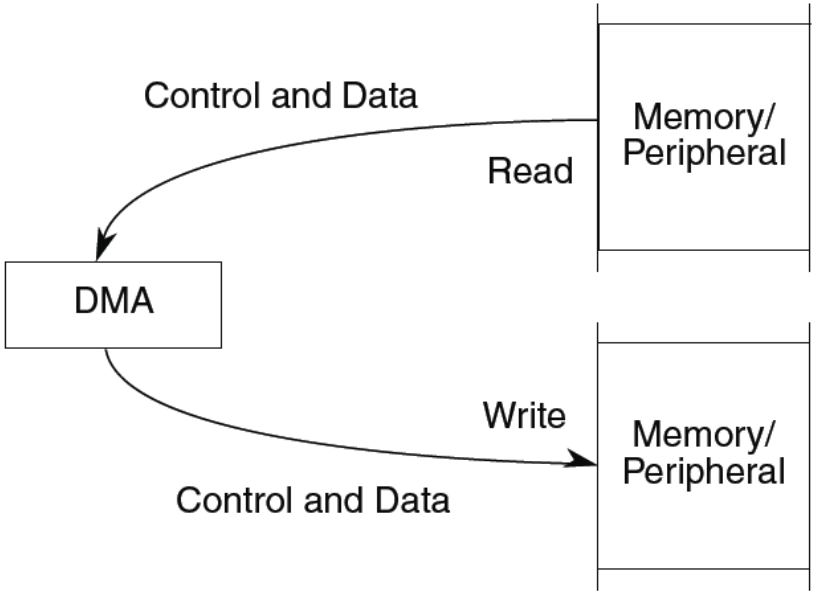
Using Direct Memory Access to Improve Performance

Overview

- **Basic Concepts**
- **DMA Peripherals in STM32F4**
- **DMA Applications**
  - Data Transfer
  - Replacing ISRs

Basic Concepts

- **Special hardware to read data from a source and write it to a destination**
- **Various configurable options**
  - Number of data items to copy
  - Source and destination addresses can be fixed or change (e.g. increment, decrement)
  - Size of data item
  - When transfer starts
- **Operation**
  - Initialization: Configure controller
  - Transfer: Data is copied
  - Termination: Channel indicates transfer has completed
  - Post-transfer operation (assert an interrupt)



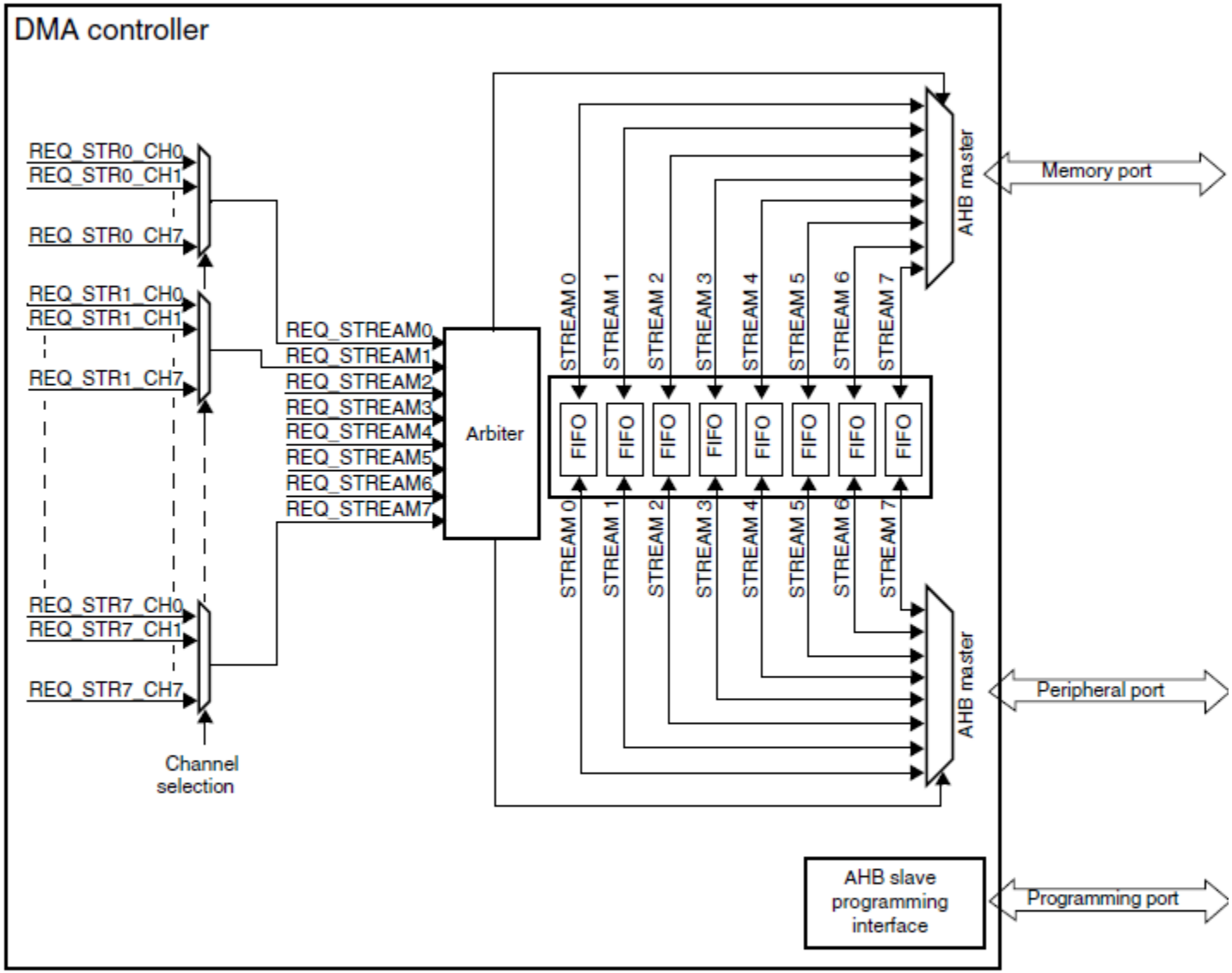
DMA Controller Features

- Two DMA controllers
- 16 streams in total (8 for each controller)
- Up to 8 channels (request) per stream
- Each channel is responsible for specific peripheral or memory requests
- Arbiter handle the priority between DMA requests
- 4 levels of software programmable priority
- 4 separate 32 first-in, first-out memory buffers(FIFOs) per stream

Channels and Streams

- Streams are concurrent
- Each stream can be only triggered by one of 8 channels at a time

DMA Controller



Registers

- **DMA_SxCR**
  - Stream x configuration register
- **DMA_SxPAR**
  - 32-bit Stream x Peripheral address register
- **DMA_SxM0AR**
  - 32-bit Stream x Memory address register
- **Both source and destination transfers can address the entire 4 GB area, at address comprised between 0x0000 0000 and 0xFFFF FFFF**

DMA Stream x Configuration Register DMA_SxCR(x=0..7)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[3:0]			MBURST [1:0]		PBURST[1:0]		Reserved	CT	DBM or reserved	PL[1:0]	
				rw	rw	rw	rw	rw	rw	rw		rw	rw or r	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

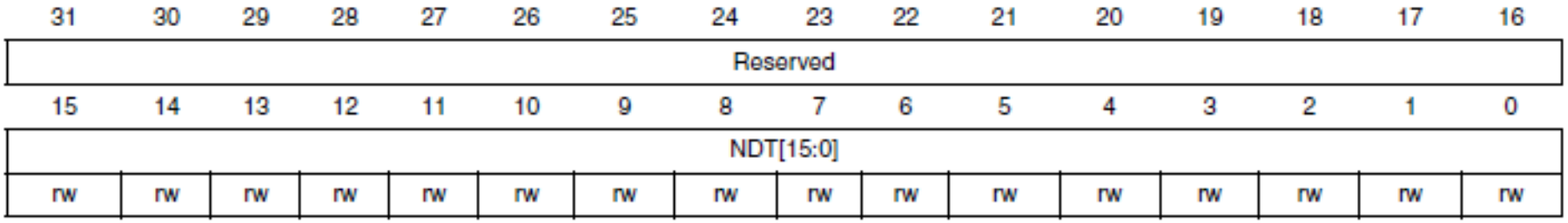
- **Configure the concerned stream**
  - CHESEL[2:0]: channel selection, can be written if EN is cleared.
  - PL[1:0]: Priority level, with 11 being very high and 00 being low.
  - MSIZE[1:0]: Memory data size
  - PSIZE[1:0]: Peripheral data size
  - DIR[1:0]: Data transfer direction
    - 00: P to M; 01: M to P; 10: M to M; 11: reserved
  - EN: Stream enable/ flag stream ready when read low

DMA Stream x Configuration Register DMA_SxCR(x=0..7)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[3:0]			MBURST [1:0]		PBURST[1:0]		Reserved	CT	DBM or reserved	PL[1:0]	
				rw	rw	rw	rw	rw	rw	rw		rw	rw or r	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

- **Configure the concerned stream**
  - MSIZE[1:0]; PSIZE[1:0] data size configuration
    - 00: byte; 01:half-word; 10:word; 11:reserved
  - TCIE: transfer complete interrupt enable
  - Other configuration includes:
    - Direct mode
    - Error interrupt
    - Circular mode
    - Burst transfer
    - Increment

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



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

- **NDT[15:0] Number of data items to transfer**
  - 0 up to 65535
  - Writable only when the stream is disabled
  - When the stream is enabled, it is read-only, indicating the remaining data items to be transmitted
  - When the value is zero, no transaction will be served

Basic Use of DMA

- Wait for the stream operation to be finished
- Clear DMA_LISR and DMA_HISR
- Set the source address
- Set the destination address
- Specify numbers of datum and width of each datum
- Configure the channel and stream
- Configure the priority
- Configure the FIFO usage
- Configure the data transfer direction
- Enable the DMA channel
- Start to transfer
- Wait for end of transfer

End of transfer

- **It is possible to interrupt in case of the following events:**

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

- **If the interrupt is disabled, there are still two ways to check if the transfer has ended (in normal mode):**
  - The EN bit of CR will be hardware cleared at the end of the transfer
  - The value of DMA_SxNDTR register will be 0, indicating no items to be transmitted

Demonstration: Flash to SRAM transfer

- Software-triggered by enabling the Channel
- Transfer word data buffer from Flash memory to embedded SRAM memory
- DMA2 Steam 0 Channel 0 is configured to transfer the 32-word data
- Only DMA2 Streams are able to perform memory to memory transfers
- Could be used as a fast version of memcpy function, but performed by DMA instead of CPU

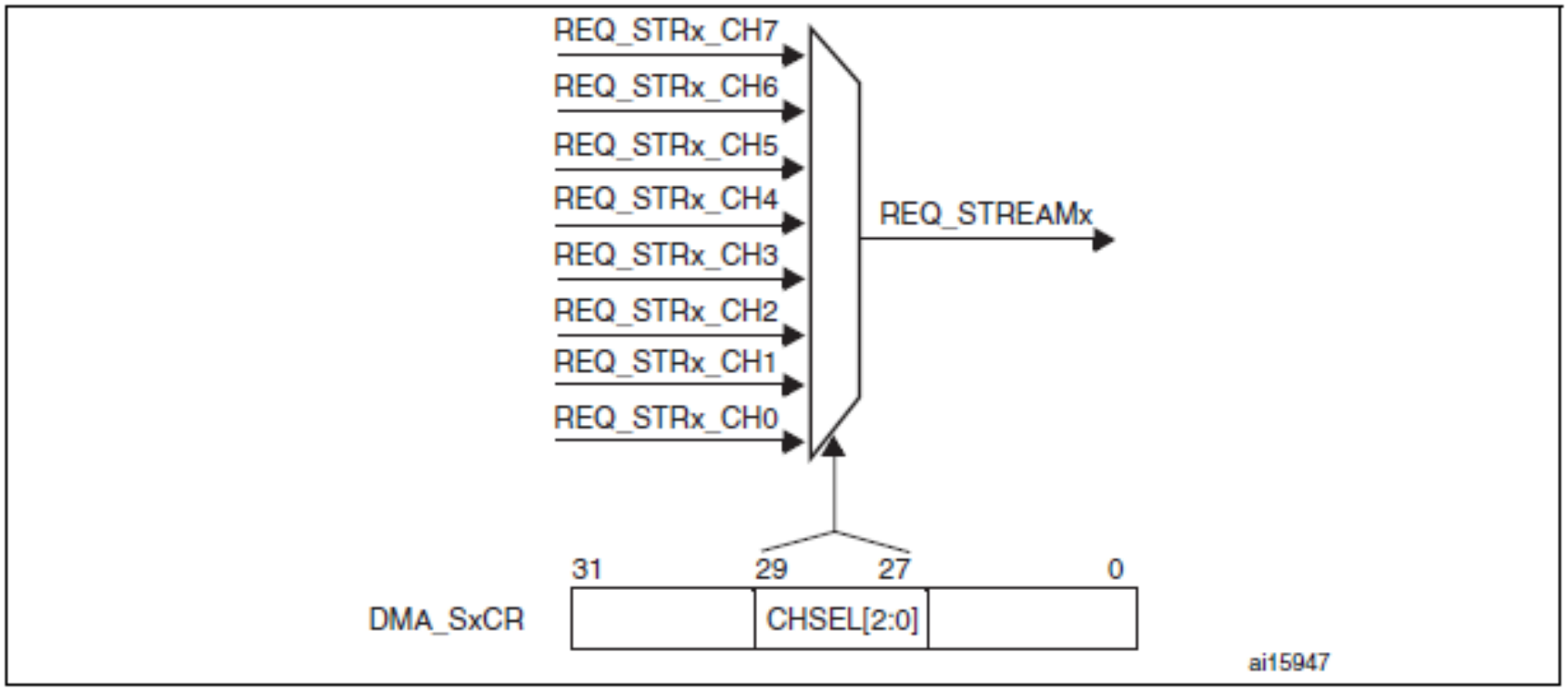
DMA vs. ISR

- **To communicate between CPU and peripherals, there are basically four approaches:**
  - Polling
  - Interrupt
  - DMA
  - Channel I/O

- **Interrupt improves the performance of the CPU in many ways compared to polling. However, the overhead of interrupt may scale up as the number of peripherals increases.**

- **DMA, on the other hand, take care of the peripheral once the configuration is made by the CPU. Exempt the CPU from being interrupt too frequently.**

Triggering DMA Activity Using Peripherals



- In general cases, have to configure DMA and peripherals at the same time so that DMA can be triggered by a specific peripherals.
- Many peripherals have DMA supports (e.g. ADC), but it have to be at least enabled!
- Memory to peripheral or peripheral to memory mode.

DMA1 Requests

Peripheral Requests	S0	S1	S2	S3	S4	S5	S6	S7
C0	SPI3_RX		SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX		SPI3_TX
C1	I2C1_RX		TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
C2	TIM4_CH1		I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
C3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
C4	UART5_RX	USART3_RX	UART4_RX	UART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
C5			TIM3_CH4 TIM3_UP		TIM3_CH1 TIM3_TRIG	TIM3_CH2		TIM3_CH3
C6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2		TIM5_UP	
C7		TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2S2_TX

DMA2 Requests

Peripheral Requests	S0	S1	S2	S3	S4	S5	S6	S7
C0	ADC1		TIM8_CH1 TIM8_CH2 TIM8_CH3		ADC1		TIM8_CH1 TIM8_CH2 TIM8_CH3	
C1		DCMI	ADC2	ADC2				DCMI
C2	ADC3	ADC3				CRYP_OUT	CRYP_IN	HASH_IN
C3	SPI1_RX		SPI1_RX	SPI1_TX		SPI1_TX		
C4			USART1_RX	SDIO		USART1_RX	SDIO	USART1_TX
C5		USART6_RX	USART6_RX				USART6_TX	USART6_TX
C6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
C7		TIM8_IP	TIM8_CH1	TIM8_CH2	TIM8_CH3			TIM8_CH4 TIM8_TRIG TIM8_COM

Demonstration of ISR Replacement: ADC to DMA

- Recall that ADC can be read either by polling or can be interrupt-driven (EOC: end of conversion or EOCIE end of conversion interrupt enable)
- ADC converted result will be automatically read by DMA thus giving the CPU freedom to perform other tasks.
- DMA can interrupt the CPU whenever it is necessary, or a periodic interrupt can perform a regular check of the data collected.
- The most optimized way to make full use of both CPU and DMA is quite application-specific. Needs to be decided accordingly.

Performance Comparison(Test result from MKL25Z4)

- **Traces**
  - Yellow: ISR is executing when trace is low
  - Blue: DAC output
- **Without DMA: Interrupt per sample**
  - 4.7 microseconds per 620 microseconds
  - 0.758% of processor's time
- **With DMA: Interrupt per cycle**
  - 5.0 microseconds per 20 milliseconds
  - 0.025% of processor's time
- **How is this useful?**
  - Saves CPU time
  - Reduces timing vulnerability to interrupts being disabled
  - Enables CPU to sleep longer, wake up less often (20 milliseconds vs. 620 microseconds)