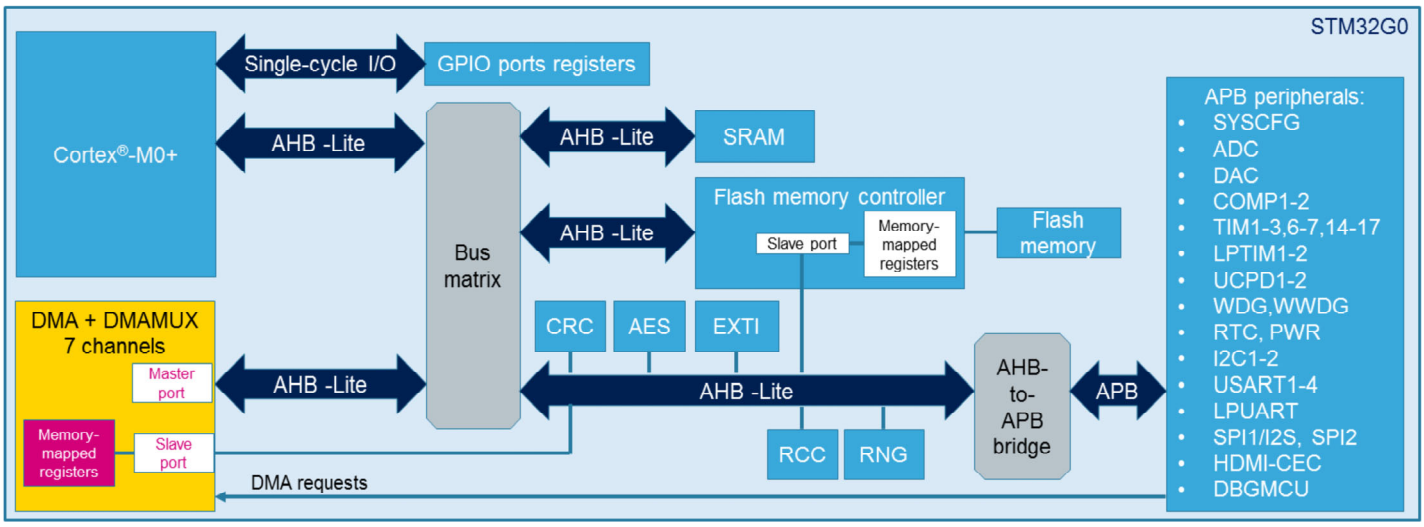
Welcome to this presentation of the STM32 direct memory access controller (DMA). It covers the main features of this module, which is widely used to handle the STM32 peripheral data transfers.

Overview

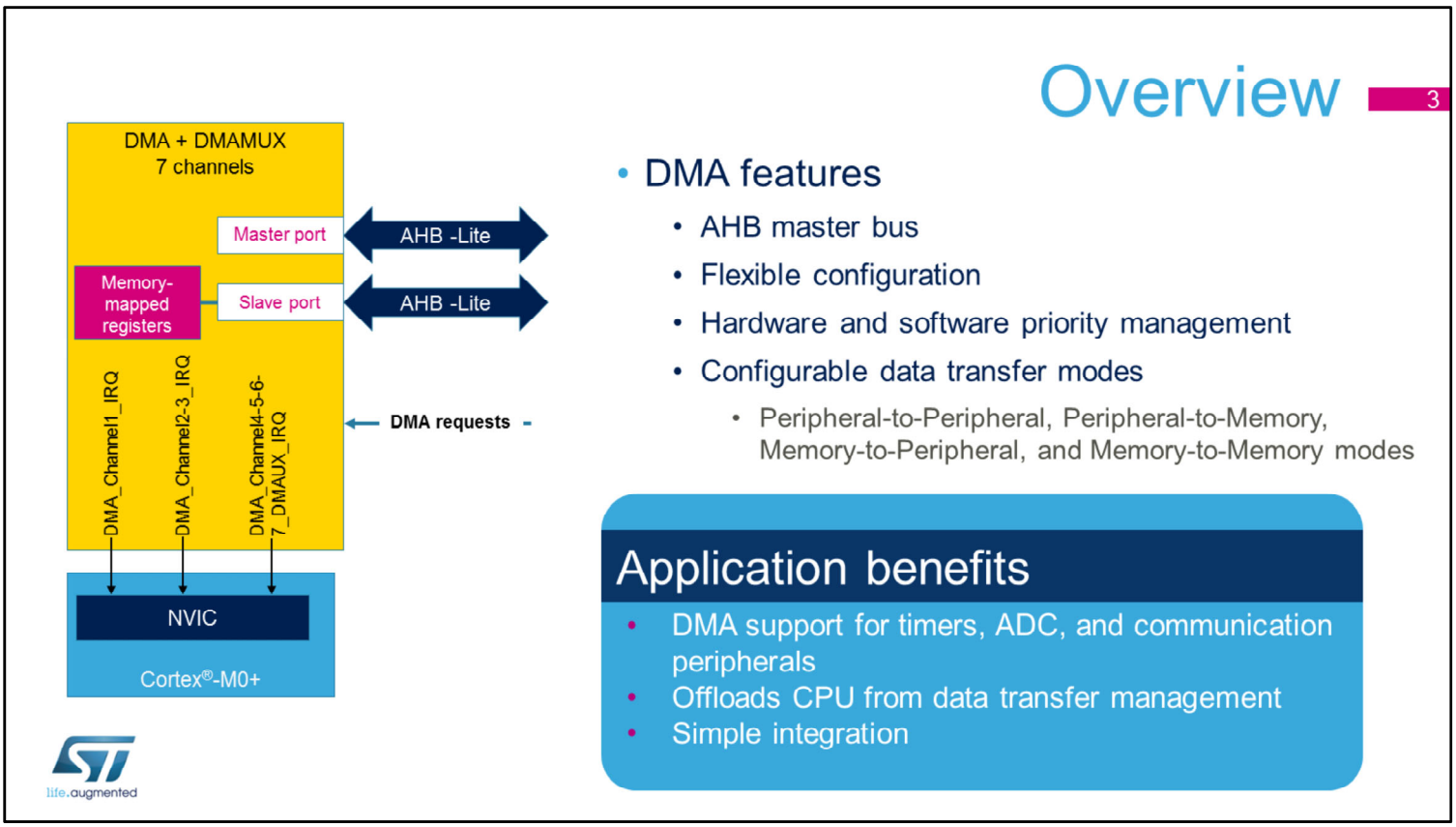


life.augmented

The Direct memory access (DMA) embedded in the STM32G0 microcontrollers is used to provide high-speed data transfers between peripherals and memory and between memory and memory. Data can be quickly moved by the DMA without any CPU action. This keeps CPU resources free for other operations

The DMA channels can access any memory-mapped location, including:

- AHB peripherals, for instance the CRC generator,
- AHB memories, for instance the SRAM,
- APB peripherals, for instance the USART peripheral.

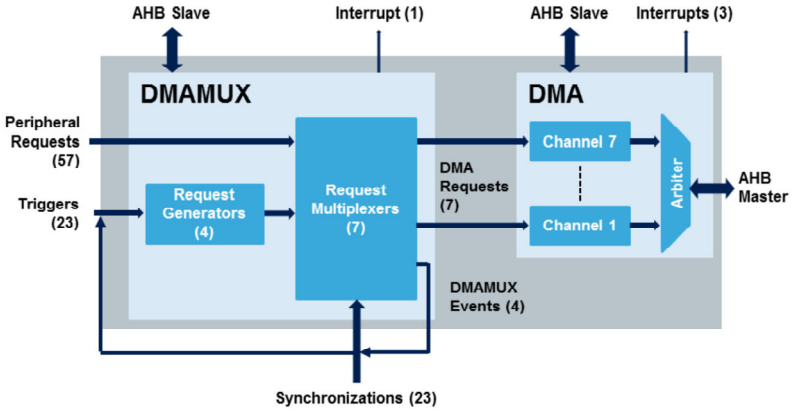


The DMA controller supports two AHB-Lite ports, one is the master port used by the DMA channels to autonomously access memory-mapped locations, memory or peripherals registers, the other is a slave port providing access to the DMA controller control and status registers.
Most APB peripherals can be configured to assert DMA requests. This is particularly useful for communication peripherals and converters (ADC and DAC). For example, let us focus on the ADC controller. It acquires samples that are temporarily stored in an internal FIFO. To transfer these samples to a buffer in SRAM, the STM32G0 offers two possibilities: asserting an interrupt request and transferring samples from FIFO to memory by software or relying on a DMA channel to empty the FIFO and transfer the contents to a buffer in SRAM. The second solution requires much less workload from the

CPU.
The DMA controller has three interrupt outputs connected to the nested vectored interrupt controller (NVIC).
The DMA request bus is a collection of requests issued by APB peripherals. The mapping of these requests to DMA channels is performed by the DMA request multiplexer (DMAMUX) unit.
Note that timer events can be used to periodically trigger DMA transfers.

Features

- STM32G0 DMA features
  - 1x DMA controller
    - Programmable block transfers with 7 concurrent channels, independently configurable
    - Programmable channel-based priority
    - Data transfers via the AHB master port (connected to the bus matrix)
  - 1x new DMA request multiplexer (DMAMUX)
    - Programmable mapping of a DMA request e.g. from any peripheral
    - Event-triggered & synchronized DMA request generation



life.augmented

Two units are in charge of handling DMA transfers: the DMA request multiplexer (DMAMUX) and the DMA controller.
The DMA controller transfers data from a source address to a destination address and manages the priority between the channels.
The DMAMUX enables the user to map requests to channels. It also handles triggers and synchronizations.
The DMAMUX is described in a dedicated presentation.

Key features

5

- 7 independent configurable channels
  - Channels provide a unidirectional transfer link between a source and a destination
  - Hardware request or software trigger on each stream
  - Software-programmable priorities with hardware priority in case of equality
- Independent and flexible stream configuration
  - Fully programmable transfer (data format, increment type, and address)
  - Independent stream interrupt flags (half transfer, transfer complete, and transfer error)
  - Support for circular buffer
- Faulty stream is automatically disabled in case of errors

life.augmented

The DMA controller has 7 channels in total, each dedicated to managing memory access requests from many peripherals. Each channel has flexible hardware requests and support for software triggers. The channel software priority is programmable and a hardware priority is used in case of equality. Channels are independently configurable. Each channel has its own data format, increment type and data address for both source and destination.

Independent channel interrupt flags allow triggering half transfer, transfer complete, and transfer error events. In case of a transfer error, the faulty channel is automatically disabled without any impact on the other active DMA channels.

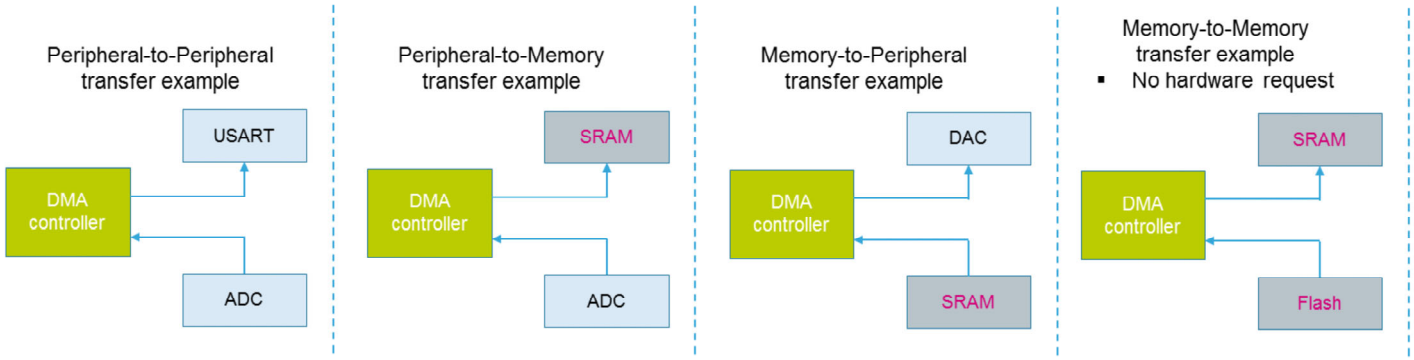
Individual DMA stream flexibility

- Programmable features
  - Independent source and destination data size (8/16/32 bits)
  - Independent source and destination start addresses
  - Independent source and destination pointer address increment (contiguously incremented / fixed address)
  - Programmable number of data to be transferred up to 65,535 requests
- Circular mode
  - Handles circular buffers with continuous data flow
  - Source and Destination addresses are automatically reloaded
  - Data transfer size is automatically reloaded

life.augmented

For each channel, the source and destination data size format is independently configurable for 8-, 16- or 32-bit packets. The source and destination addresses and pointer increment are also independently configurable. The transfer data size can be pre-programmed up to 65535 bytes. Circular buffer mode is available to support a continuous flow of data. The source and the destination addresses and the number of data to be transferred are automatically reloaded after the transfer completes.

Stream transfer management



- The peripheral DMA request signal is routed to the DMA controller stream request line via the DMA request router (DMAMUX)

life.augmented

Memory-to-memory mode allows transfers from one address location to another without a hardware request. Once the stream is configured and enabled, the transfer starts immediately. When data is transferred to or from a peripheral, the hardware request coming from the selected peripheral is used to trigger the data transfer on the DMA Peripheral port. Once the transfer is completed, the request is acknowledged.

Programmable data width, data alignment

- When PSIZE and MSIZE are not equal, the DMA performs data alignments

Source port width = 8-bit
Destination port width = 32-bit
Number of data transfers = 4

Address	Data[7:0]
0xXXXX_XXX0	B0
0xXXXX_XXX1	B1
0xXXXX_XXX2	B2
0xXXXX_XXX3	B3

Address	Data[31:0]
0xXXXX_XXX0	000000B0
0xXXXX_XXX4	000000B1
0xXXXX_XXX8	000000B2
0xXXXX_XXXC	000000B3

Source port width = 32-bit
Destination port width = 16-bit
Number of data transfers = 4

Address	Data[31:0]
0xXXXX_XXX0	B3B2B1B0
0xXXXX_XXX1	B7B6B5B4
0xXXXX_XXX2	BBBAB9B8
0xXXXX_XXX3	BFBEBDBC

Address	Data[15:0]
0xXXXX_XXX0	B1B0
0xXXXX_XXX2	B5B4
0xXXXX_XXX4	B9B8
0xXXXX_XXX6	BDBC

life.augmented

The DMA controller does not pack or unpack data.
When the source port is narrower than the destination port, data are aligned in the destination memory based on the destination port width. In the example on the left, the alignment is 32 bits. So each byte received from the source device is aligned on a 32-bit word address.
When the source port is wider than the destination port, data are truncated to fit the destination port width. In the example on the right, the 32-bit words received from the source device are truncated so that only the 16-bit lower part is written to the destination address.
When pointers are incremented on source and destination, the increment is equal to the port width.

Circular & Double buffer modes

- Circular mode:
  - All FIFO features and DMA events (TC, HT, TE) are available in this mode
  - The number of data items is automatically reloaded and transfer restarted
  - **This mode is NOT available for Memory-to-Memory transfers**

DMA_CCRx = Configuration
DMA_CNDTRx register = Number of data to be transferred
DMA_CPARx = Peripheral address
DMA_CMARx = Memory address

Transfer completion

DMA_CCRx[CIRC]=0
Circular mode disabled

DMA_CCRx = Unchanged
DMA_CNDTRx register = **Cleared**
DMA_CPARx = Unchanged
DMA_CMARx = Unchanged

- In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, **the DMA channel must be disabled**

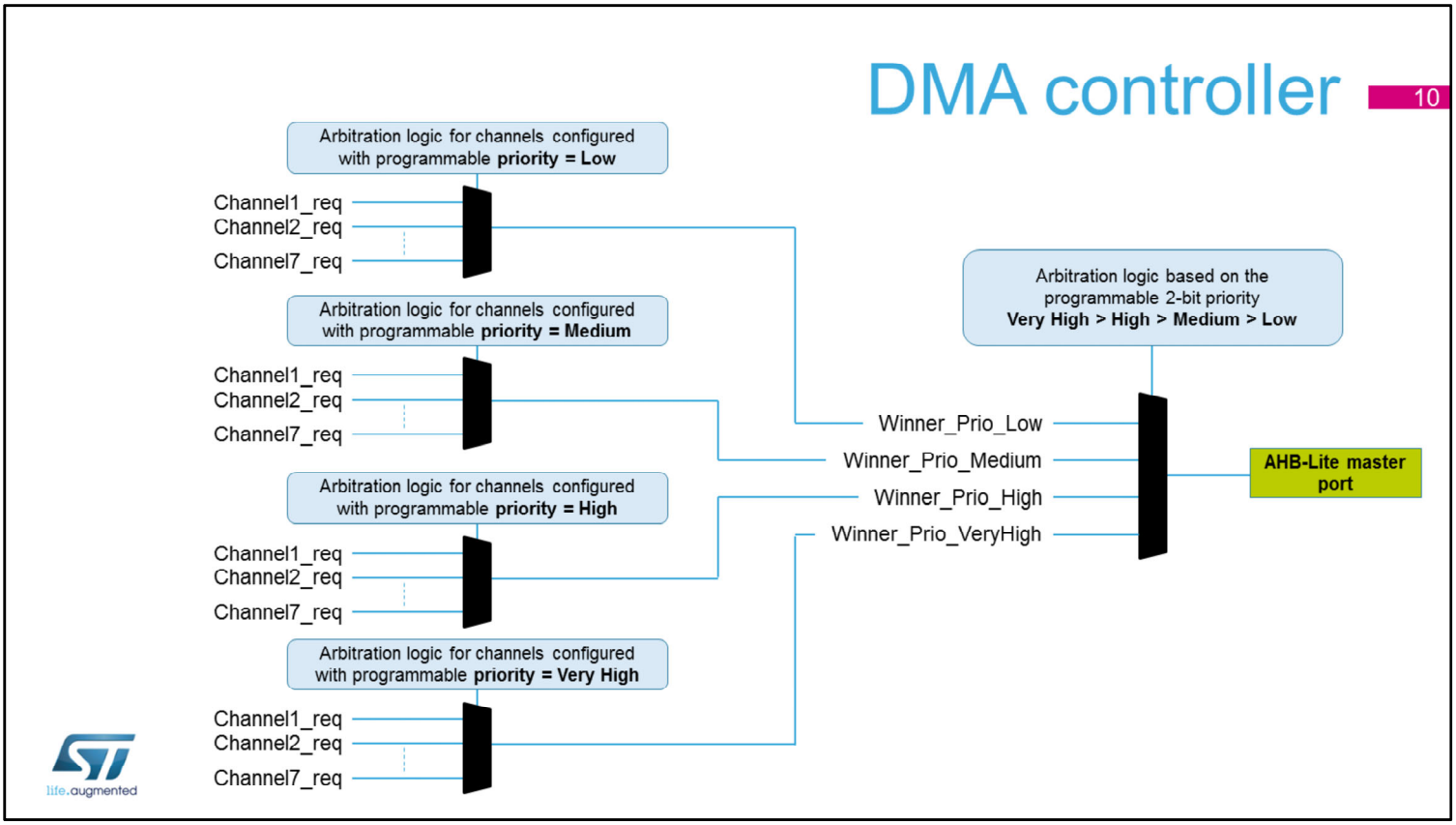
DMA_CCRx[CIRC]=1
Circular mode enabled

DMA_CCRx = Unchanged
DMA_CNDTRx register = **Automatically reloaded**
DMA_CPARx = **Automatically reloaded**
DMA_CMARx = **Automatically reloaded**

- Transfer automatically restarts when the DMA request is asserted

life.augmented

DMA controllers support Circular mode allowing to configure the number of data items to transfer once, and automatically restart the transfer after a Transfer Complete event.
This is convenient to support continuous transfers, such as ADC scan mode.
When Circular mode is active, the internal registers containing current source and destination addresses, which are not visible by the software, are automatically reloaded upon transfer completion from DMA_CPARx and DMA_CMARx registers.



The priorities between the DMA stream requests are software-programmable (4 levels consisting of very high, high, medium, and low) or hardware in case of equality (request 0 has priority over request 1, etc.).
Consequently, the arbitration is performed in two stages.
The first stage selects the winner among all channels programmed with same priority. This arbitration is instantiated four times, one per programmable priority level. The channel with the lowest number will get priority versus the channel with the highest number.
The second stage selects the winner among all channels programmed with different priority.
Channel arbitration is reconsidered between every data transfer.

Interrupts

- Interrupt events for each channel

Interrupt event	Description
Half transfer	Set when half of the data transfer size has completed
Transfer complete	Set when the full data transfer size has completed
Transfer error	Set when a bus error occurs during the data transfer

life.augmented

Each DMA stream is designed with this group of interrupt events. The Half Transfer interrupt flag is set when half the data has been transferred; the Transfer Complete flag is set when the transfer is complete; the Transfer Error flag is set when an error occurs during the data transfer.

DMA in low-power modes

Mode	Description
Run	Active.
Low-power run	Active.
Sleep	Active. DMA interrupts can wake up the CPU.
Low-power sleep	Active. DMA interrupts can wake up the CPU.
Stop 0/Stop 1	Clocked-off & frozen. DMA registers retention.
Standby	Powered-down. DMA must be reinitialized after exiting Standby mode.
Shutdown	Powered-down. DMA must be reinitialized after exiting Shutdown mode.

This table indicates the state of the DMA controller according to the current power mode.
In Sleep and Low-power sleep modes, the DMA controller remains active and can be used to transfer UART or I2C received characters to memory while waking-up the microcontroller.

Main differences with STM32F0

- The DMA Controller is similar to the one implemented in STM32F0 microcontrollers, but with the additional DMA request multiplexer (DMAMUX)

	STM32F0	STM32G0
DMA	2 DMAs	1 DMA
DMA Features	Same	
DMAMUX	No	Yes

life.augmented

The STM32G0 includes a DMA request multiplexer (DMAMUX) that maps transfer requests issued by peripherals to DMA channels.

Related peripherals

- Refer to these trainings linked to this peripheral:
  - DMA request multiplexer (DMAMUX)

life.augmented

You can refer to training slides related to the DMA request multiplexer (DMAMUX) for additional information.