life.augmented

STM32WBA Workshop GPDMA

Agenda

#	Overview	🧠
#	Structure of example	🧠
#	ADC+UART+TIM with LLI controlled GPDMA	0101 1111 1010
#	Conclusion	🧠
#	What's next	0101 1111 1010

🧠	Theory
0101 1111 1010	Hands-on

life.augmented

DMA overview

life.augmented

DMA overview

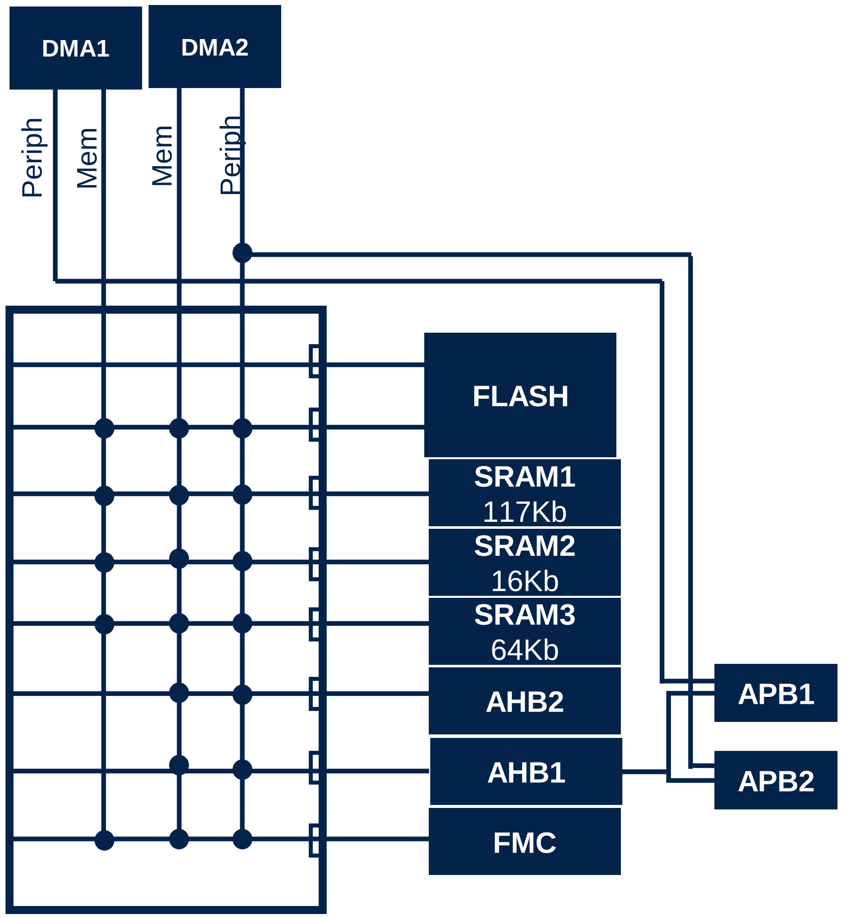
- A new DMA module
  - 2 hardware instances
    - GPDMA with symmetric configuration
  - Dual port DMA with dedicated path to APB
  - Integrated DMAMUX features
  - Linked-list based programming
  - Flexible intra-channel and inter-channel input/output control
  - Run-time isolation features

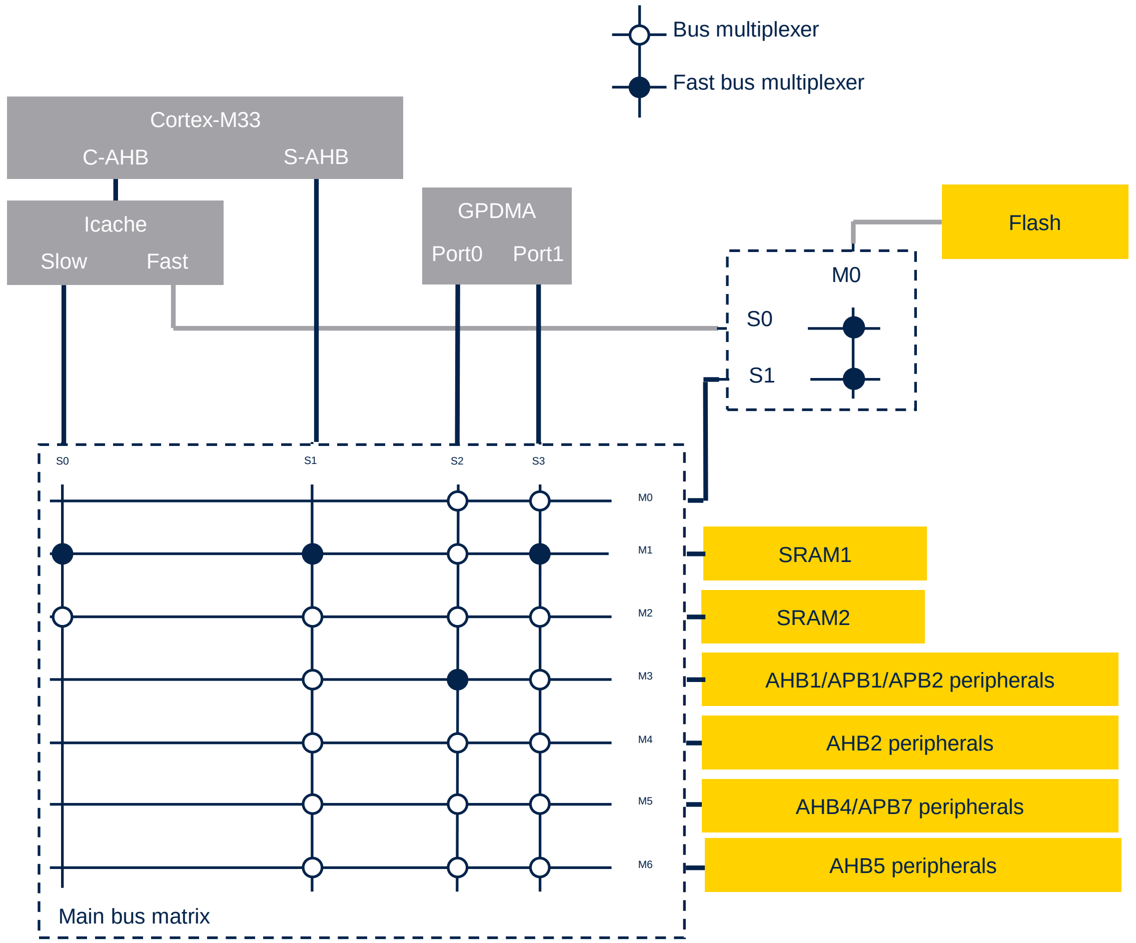
Application benefit

- Off-load CPU for data transfers from a memory-mapped source to a memory-mapped destination

STM32WBA GPDMA integration

STM32F4x9





life.augmented

AVNET SILICA

GPDMA key features 1/2

- Bidirectional AHB master port(s): GPDMA1: 2 ports
- Memory-mapped data transfers from a source to a destination
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during Sleep mode
- 8 Concurrent DMA channels for each controller
- Transfers arbitration is based on a 4-grade priority policy
  - One reserved highest priority queue for time-sensitive traffic
  - Three lower priority queues with weighted round robin allocation

life.augmented

GPDMA key features 2/2

- **GPDMA data handling**
  - Byte-based reordering
  - packing/unpacking
  - padding/truncation
  - sign extension
  - left / right alignment
- **Linear addressing mode**
  - Fixed addressing (typically for peripheral data register)
  - Contiguously-incremented addressing (typically for memory access)
  - Blocks up to 64kB (16-bit BNDT)
- **2D addressing mode (ch6..7), additional**
  - Repeated block mode: programmable repeated block counter (11-bit BRC, up to 2k blocks)
  - Programmable source/destination signed burst address offset (2x 14bit, up to +/-8kB)
    - Non-contiguous incremented/decremented addressing after each burst
  - Programmable source/destination signed block address offset (2x 17bit, up to +/-64kB)
    - Non-contiguous incremented/decremented addressing after each block

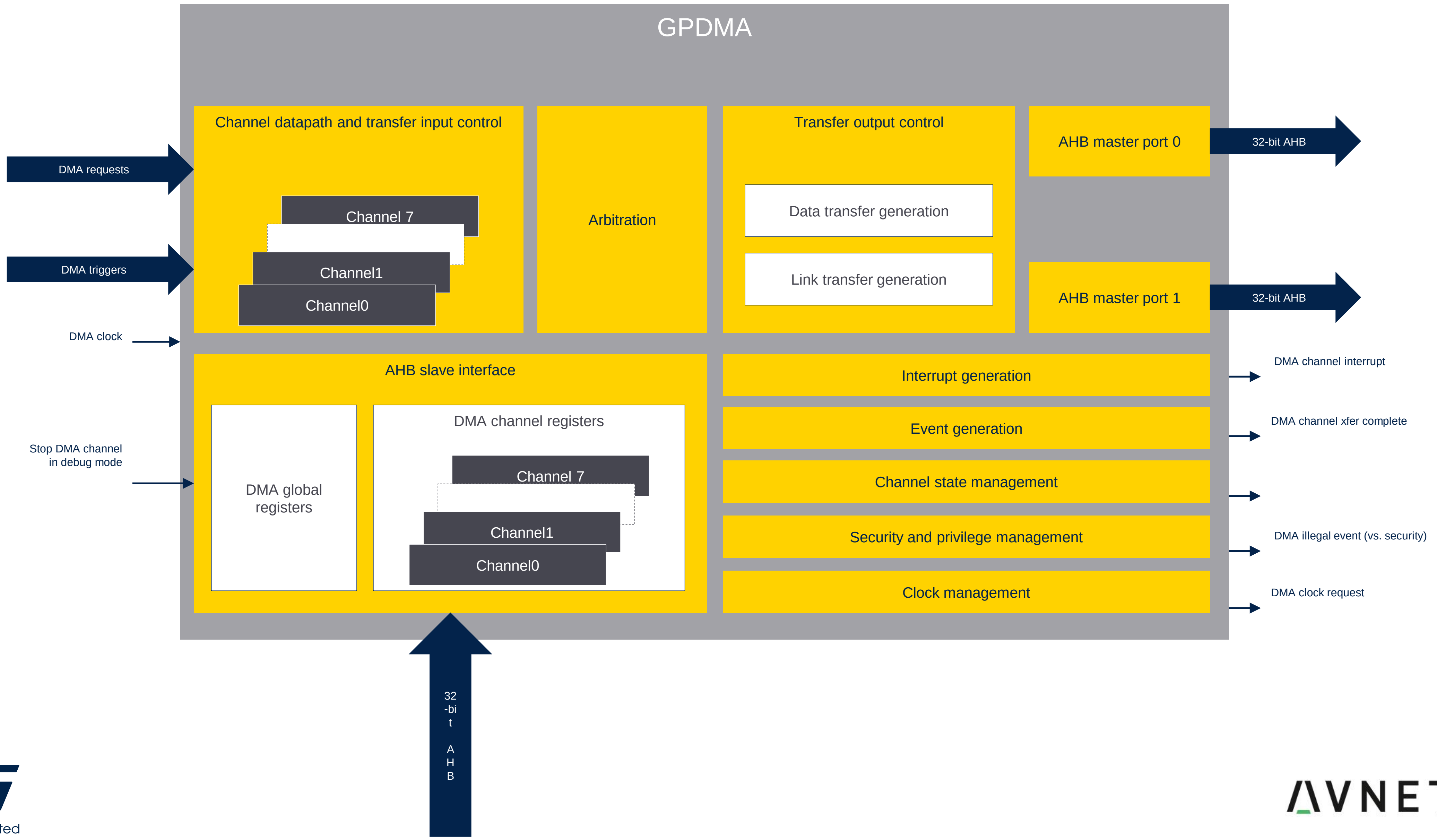
GPDMA linked-list register file

The GPDMA will use nodes for those registers

- TR1 - Transfer register 1
- TR2 - Transfer register 2
- BR1 - Block register 1
- SAR - Source address register
- DAR - Destination address register
- TR3 - Transfer register 3 ... SA & DA offsets incremet for **2D**
- BR2 - Block register 2 ... block repeated SA & DA offsets for **2D**
- LLR - Linker list register ... link to next LL node & update parameters

In our case each linked list node will update this GPDMA registers after previous GPDMA node is finished. This is automatically reconfiguring the GPDMA channel.

DMA block diagram



DMA specific implementation & user guidelines

Feature	GPDMA
Number of channels	8
Master port(s)	2x (32-bit) AHB ❖ Port#0 should be typically allocated for transfers to/from peripherals ➢ There is a direct hardware datapath to APB peripherals, outside the AHB matrix ❖ Port#1 should be typically allocated for transfers to/from memory ❖ In any case, any GPDMA target can be addressed from any port
DMA transfers	Single and bursts
DMA scheduler	FIFO-based bursts (dual issue)
Channel FIFO size	Ch 0-3: 8 bytes (2 words) ❖ These channels should be typically allocated for transfers from/to an APB/AHB peripheral and SRAM Ch 4-7: 32 bytes (8 words) ❖ These channels may be also used for transfers from/to a data-demanding AHB peripheral and SRAM. ❖ 4-word burst should be privileged when applicable for faster performances (faster back-to-back transfers, lower bus utilization)
Channel addressing mode	linear
DMA request from	ADC4, SPI1,3, I2C1,3, USART1,2, LPUART1, TIM1,2,3,16,17, AES, SAES, HASH, LPTIM1,2
Trigger from	EXTI, TAMP, LPTIM1,2, RTC, GPDMA, TIM2, ADC4

Structure of example

life.augmented

ADC and UART with LLI controlled GPDMA transfer triggered by TIMER

- Use **NUCLEO-WBA52 board** and **STM32CubeIDE**
- Setup ADC to
  - convert 4 channels in circle - channels 0, 2, 6, 13
  - generate DMA requests
- Setup USART to
  - transmit obtained data
- Setup TIM to
  - generate a 1 second trigger
- Set GPDMA with linked list
  - to get data from ADC and transfer them to buffer after trigger
  - to get data from buffer and transfer them to UART3

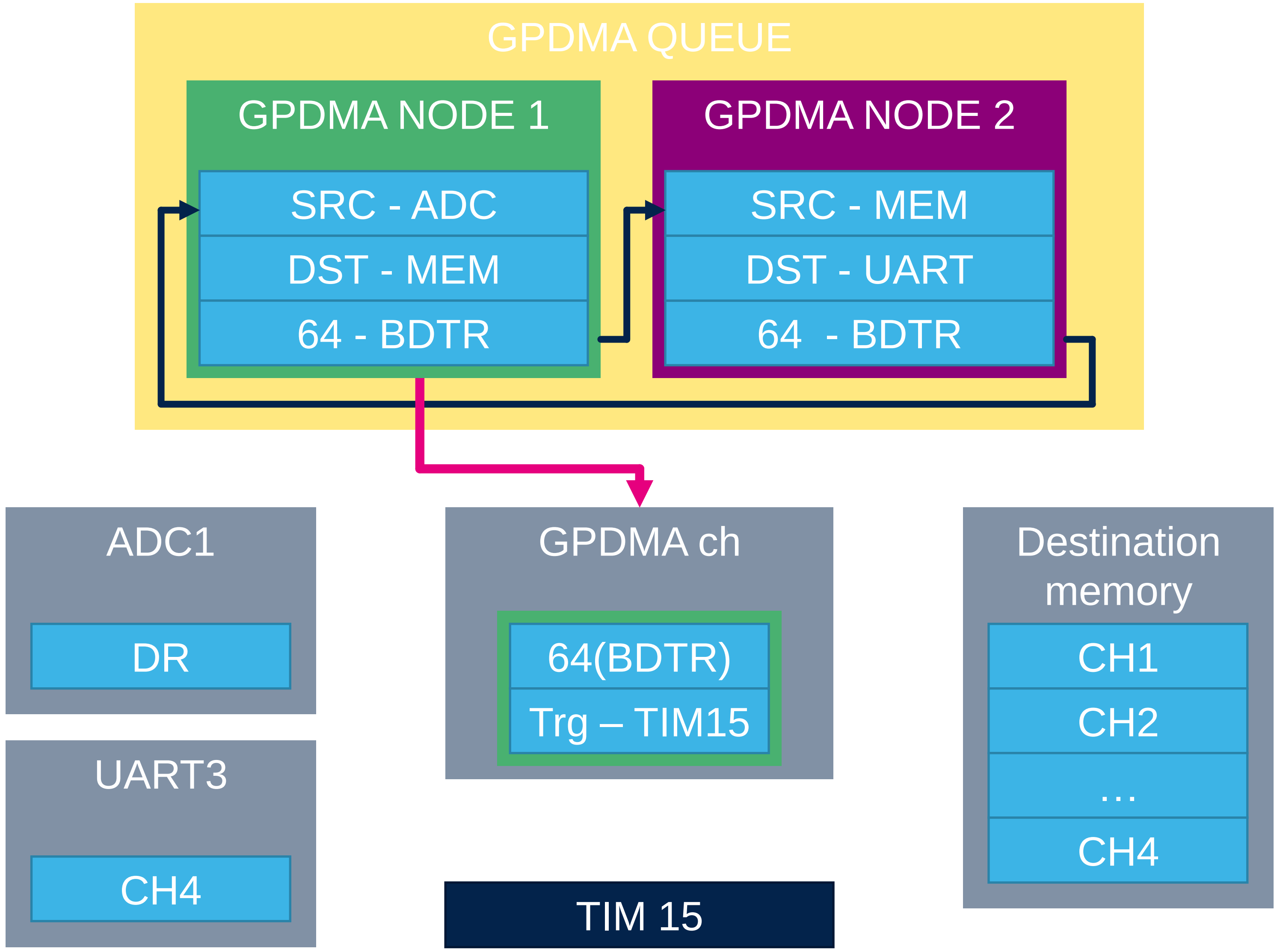
life.augmented

AVNET SILICA

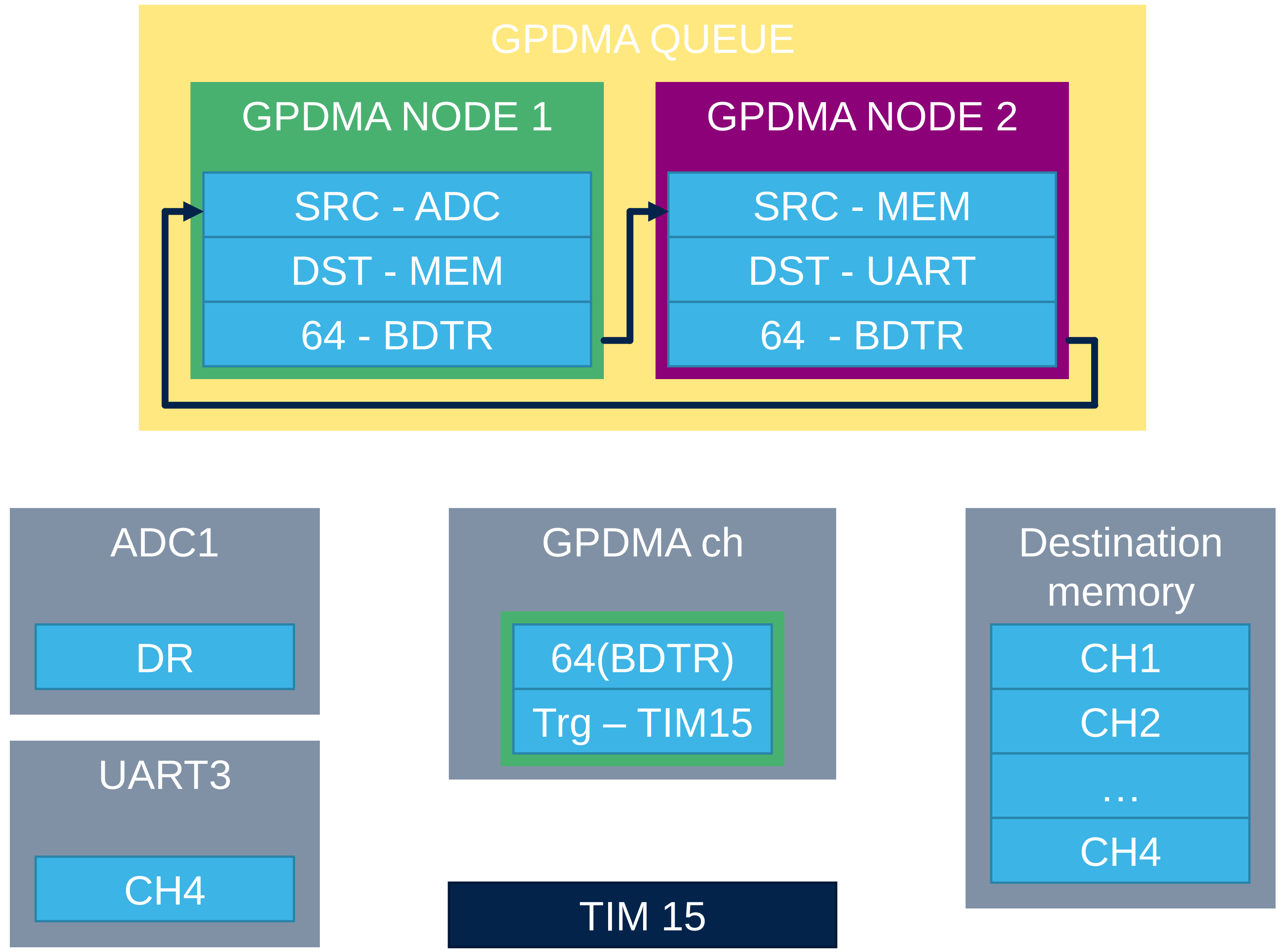
ADC + UART + TIM with LLI controlled GPDMA transfer

- If ADC and UART work in loop without a trigger, it may cause a crash of terminal due to high baud rate.
- We can slow down the GPDMA by adding trigger.
- As trigger source in our case, we will use a timer TIM15 with period of 1s.
- Then the GPDMA transfer will be conditioned by this trigger event.

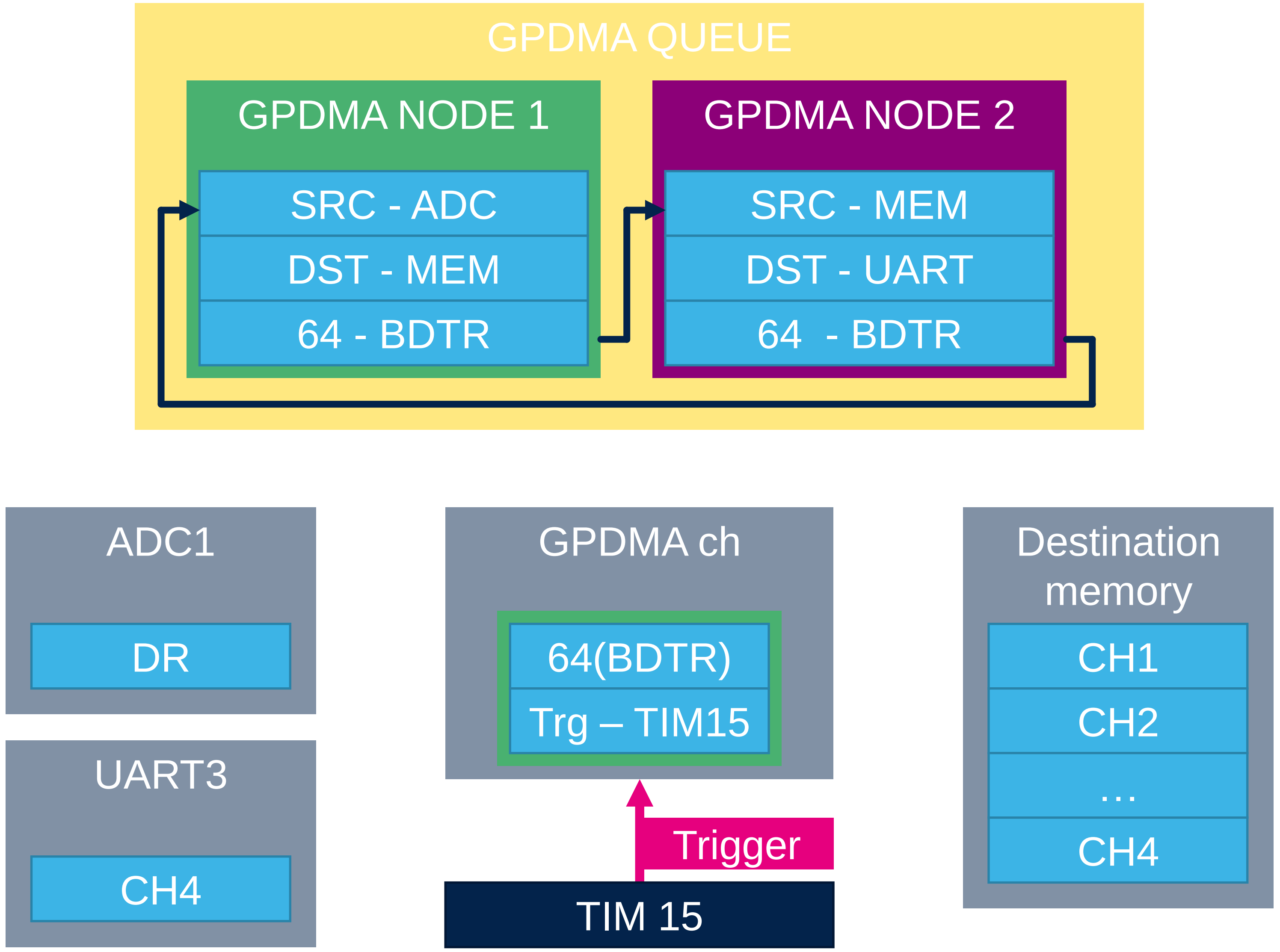
ADC + UART + TIM with LLI controlled GPDMA transfer



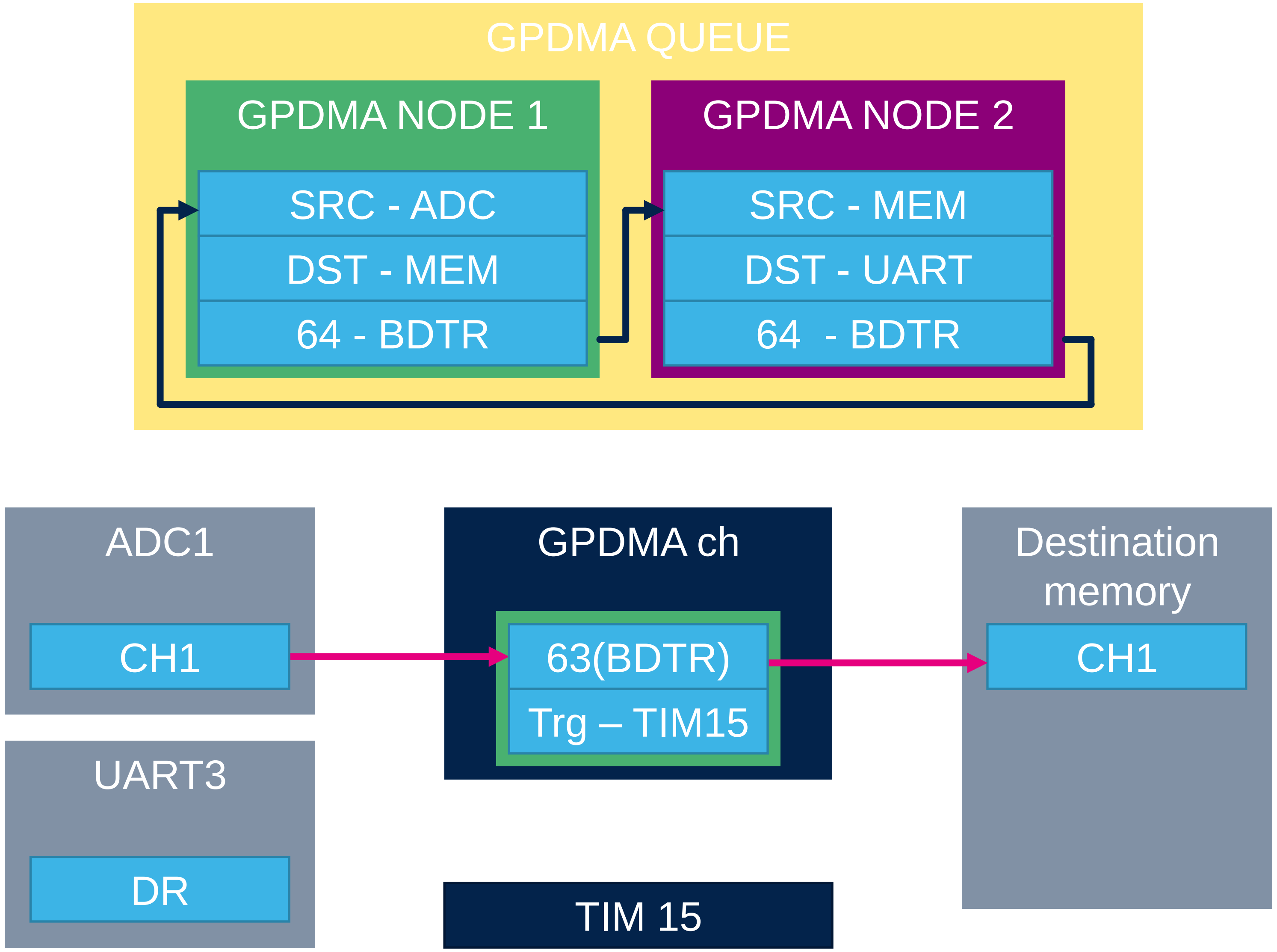
ADC + UART + TIM with LLI controlled GPDMA transfer



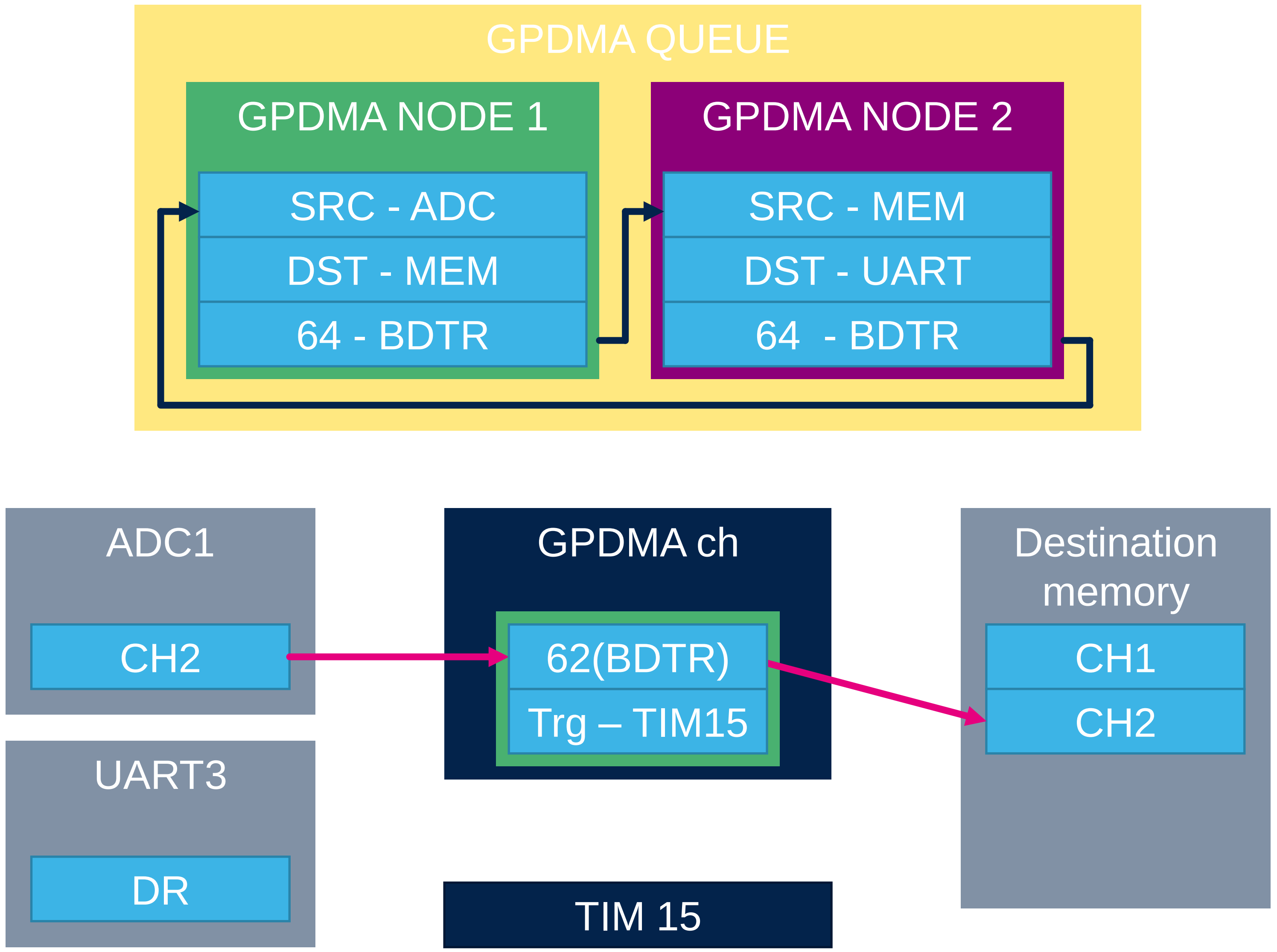
ADC + UART + TIM with LLI controlled GPDMA transfer



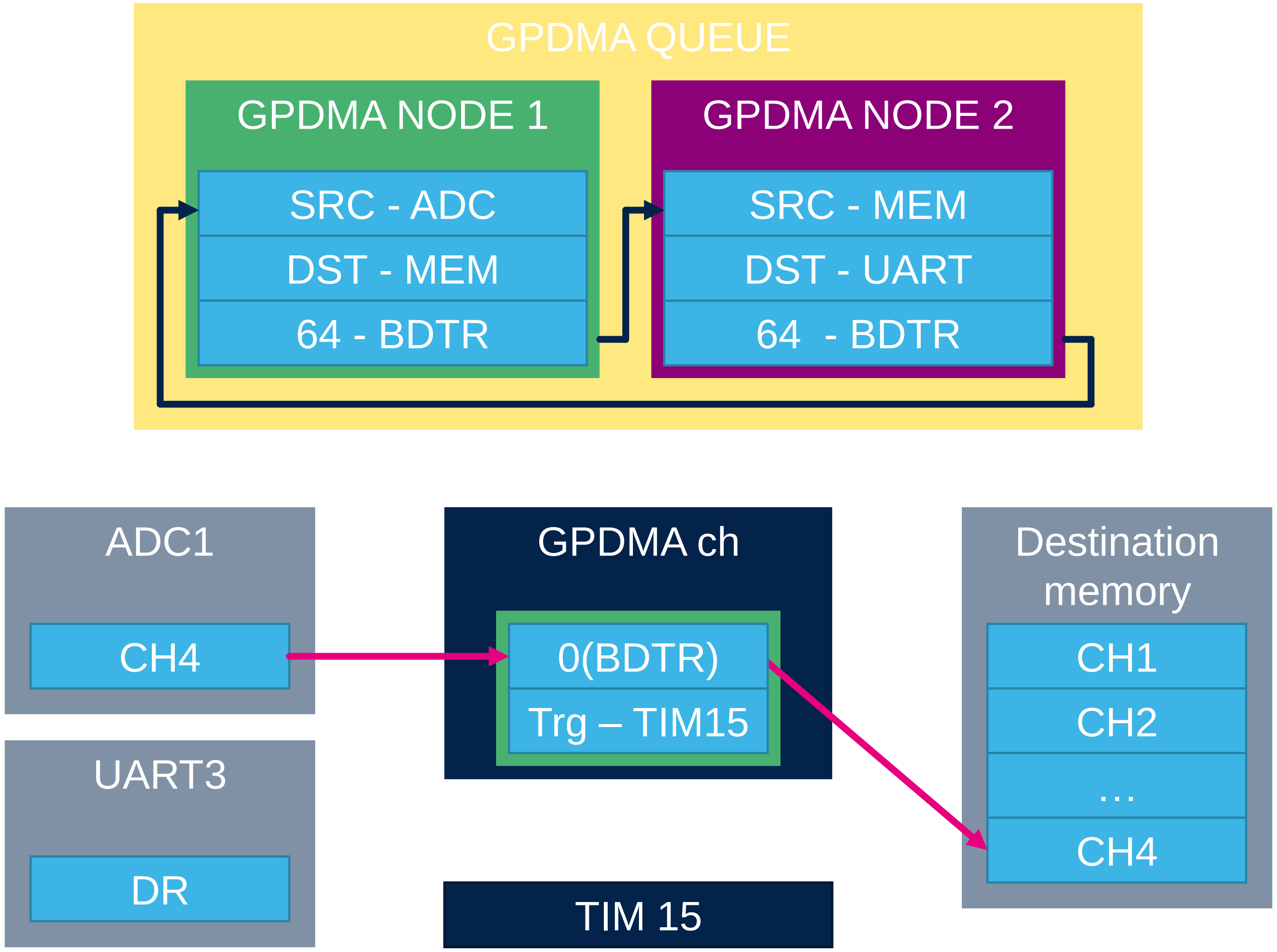
ADC + UART + TIM with LLI controlled GPDMA transfer



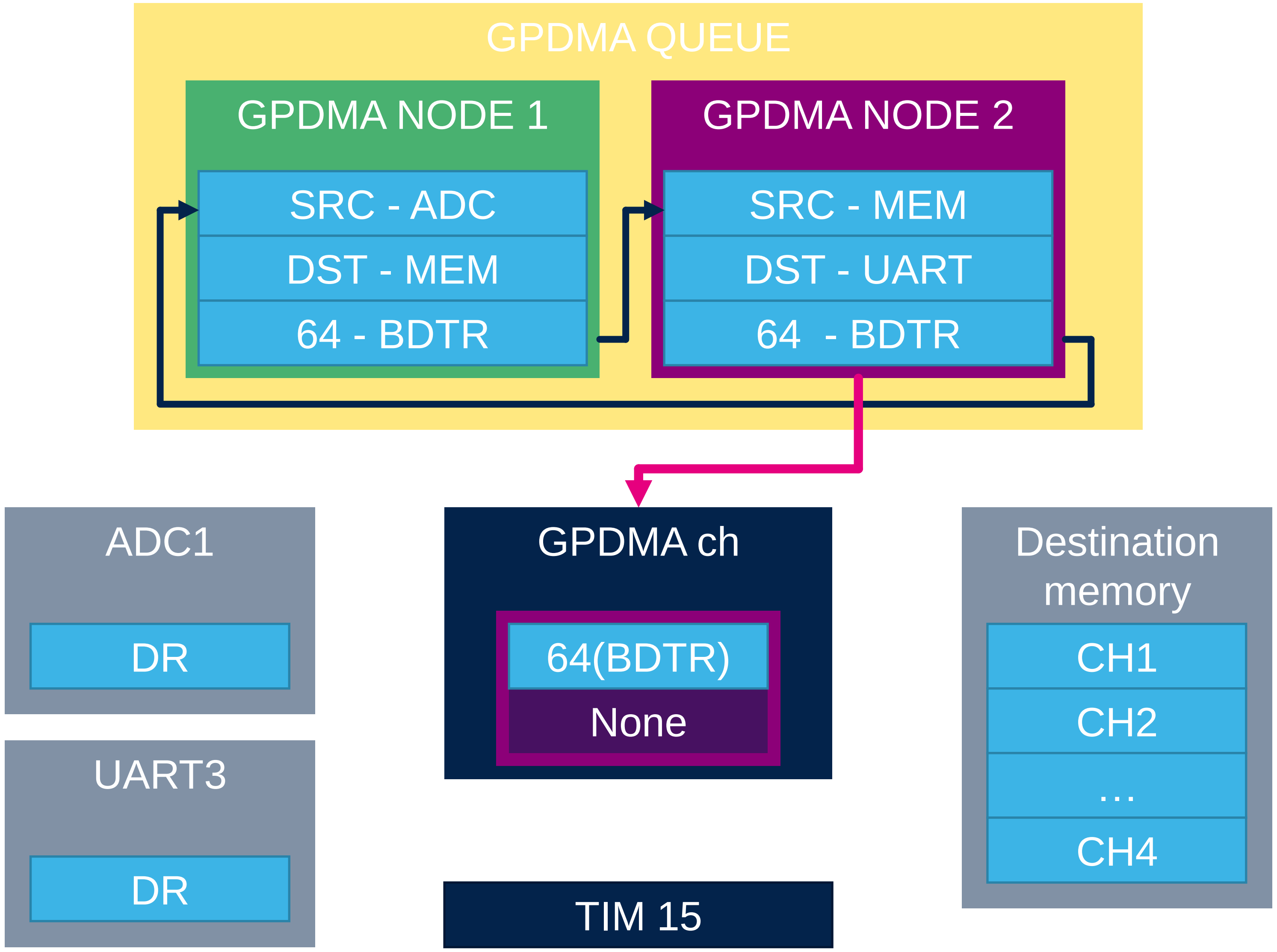
ADC + UART + TIM with LLI controlled GPDMA transfer



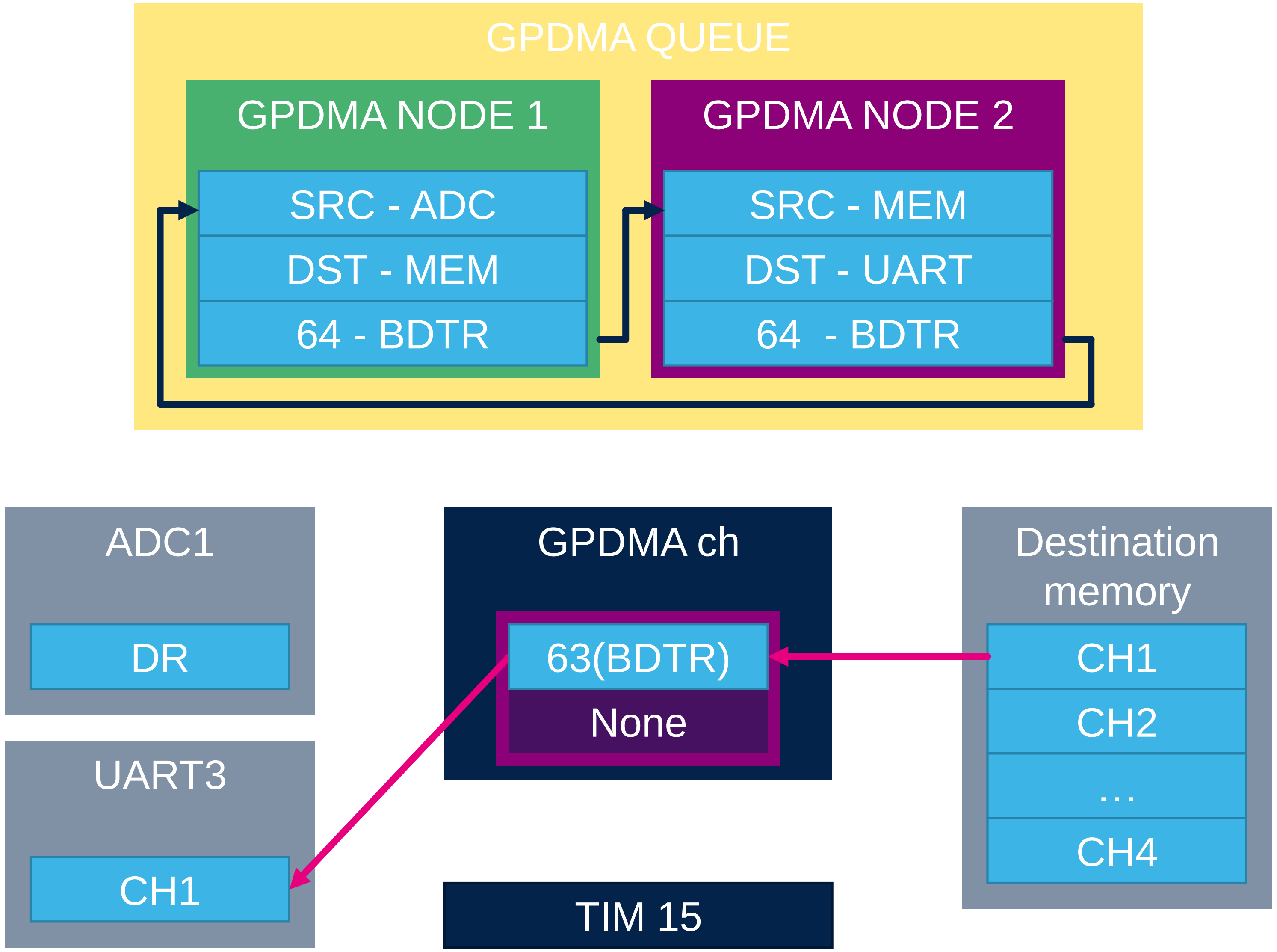
ADC + UART + TIM with LLI controlled GPDMA transfer



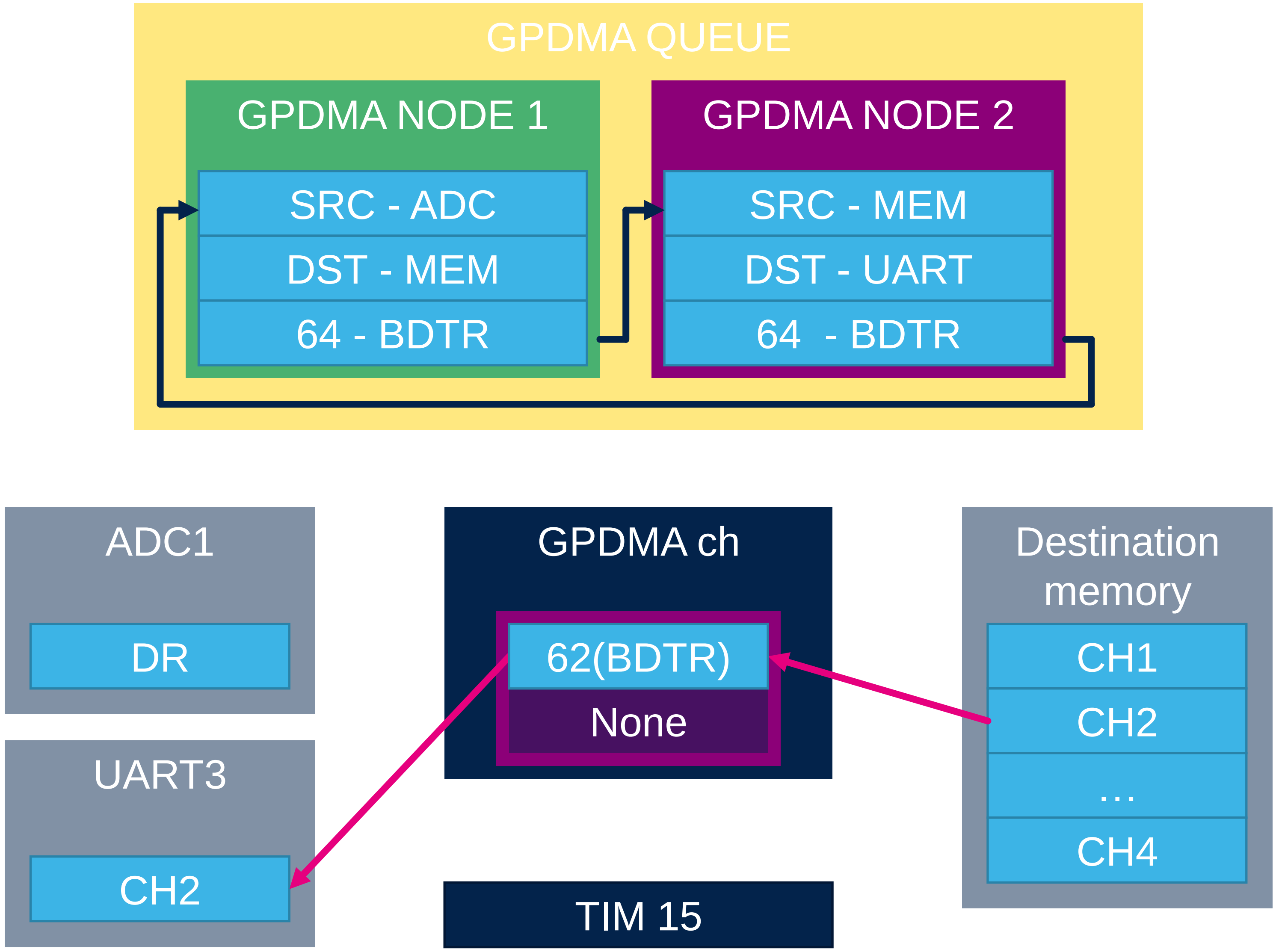
ADC + UART + TIM with LLI controlled GPDMA transfer



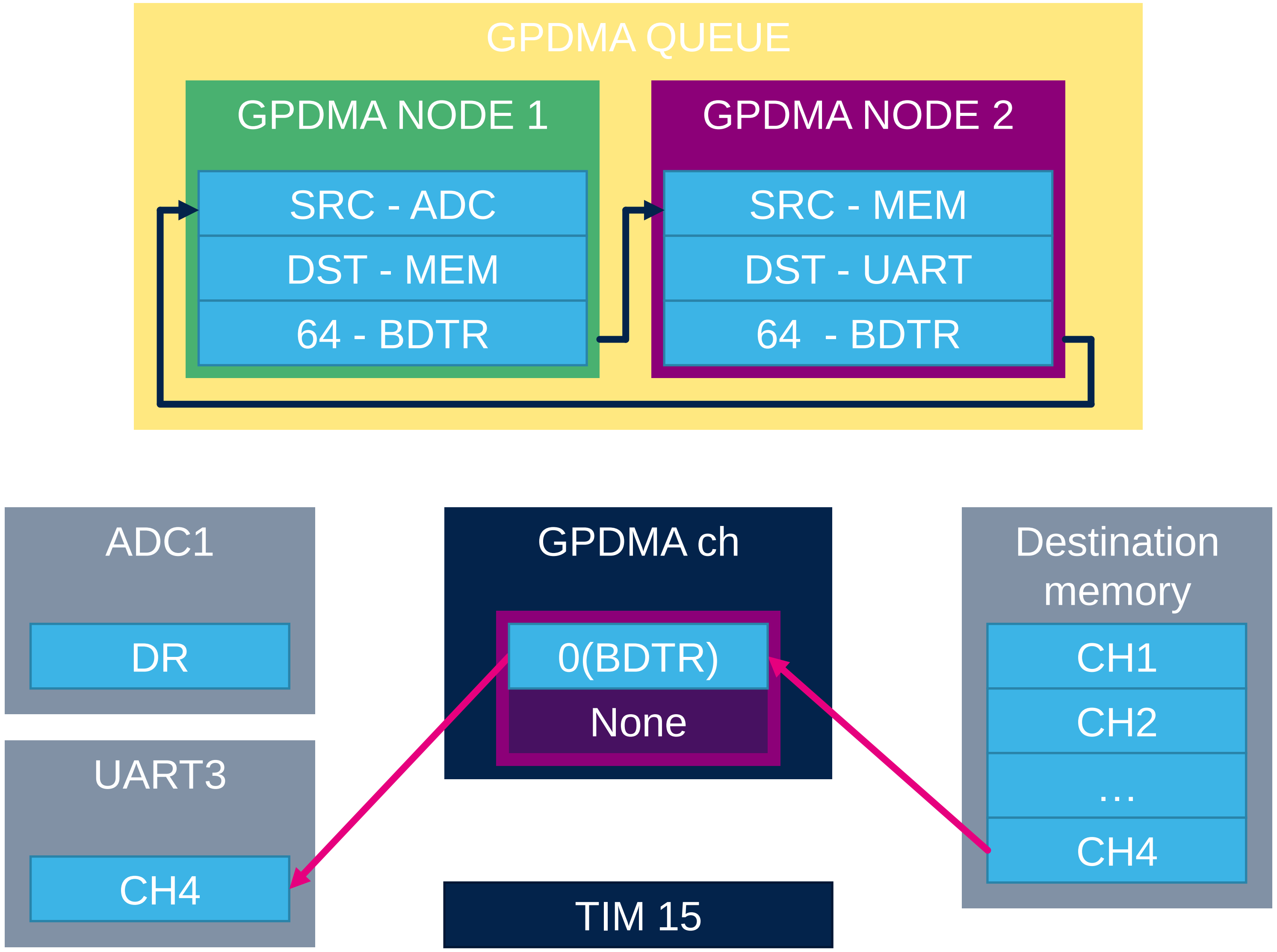
ADC + UART + TIM with LLI controlled GPDMA transfer



ADC + UART + TIM with LLI controlled GPDMA transfer



ADC + UART + TIM with LLI controlled GPDMA transfer



Hands-on

Please open STM32CubeIDE

Conclusion

life.augmented

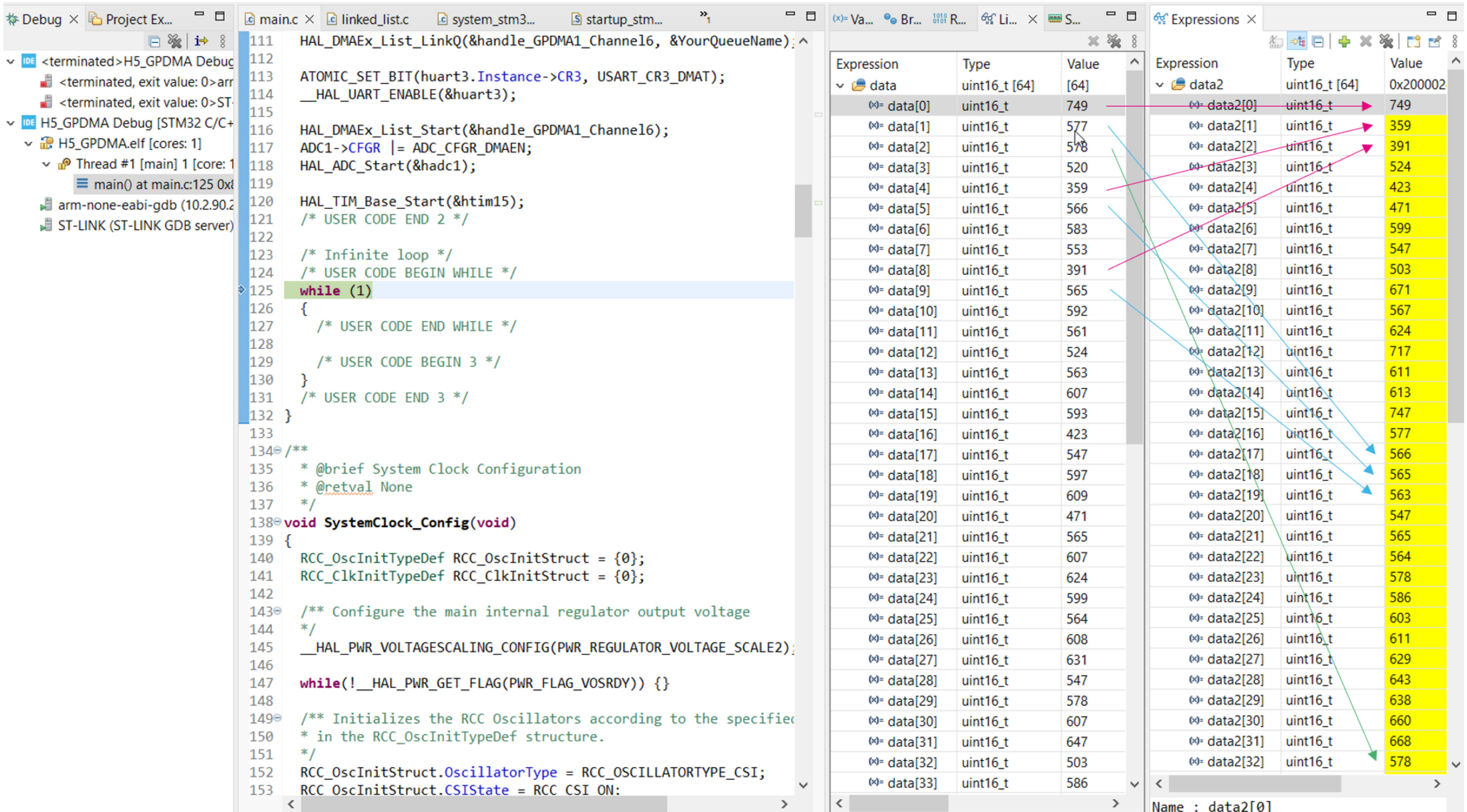
Conclusion

- GPDMA implementation on WBA
- GPDMA key features
- Autonomous data manipulation with no CPU load
- Standard request x Linked list based programing

What’s next

life.augmented

ADC + UART + TIM + 2D – LLI GPDMA



```
<terminated>H5_GPDMA Debug
  <terminated, exit value: 0>arr
  <terminated, exit value: 0>ST-
H5_GPDMA Debug [STM32 C/C+
  H5_GPDMA.elf [cores: 1]
    Thread #1 [main] 1 [core: 1
      main() at main.c:125 0x8
  arm-none-eabi-gdb (10.2.90.2
  ST-LINK (ST-LINK GDB server)
```

```c
  HAL_DMAEx_List_LinkQ(&handle_GPDMA1_Channel6, &YourQueueName);

  ATOMIC_SET_BIT(huart3.Instance->CR3, USART_CR3_DMAT);
  __HAL_UART_ENABLE(&huart3);

  HAL_DMAEx_List_Start(&handle_GPDMA1_Channel6);
  ADC1->CFGR |= ADC_CFGR_DMAEN;
  HAL_ADC_Start(&hadc1);

  HAL_TIM_Base_Start(&htim15);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

  while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}

  /** Initializes the RCC Oscillators according to the specifie
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_CSI;
  RCC_OscInitStruct.CSIState = RCC_CSI_ON;
```

Expression	Type	Value
data	uint16_t [64]	[64]
data[0]	uint16_t	749
data[1]	uint16_t	577
data[2]	uint16_t	578
data[3]	uint16_t	520
data[4]	uint16_t	359
data[5]	uint16_t	566
data[6]	uint16_t	583
data[7]	uint16_t	553
data[8]	uint16_t	391
data[9]	uint16_t	565
data[10]	uint16_t	592
data[11]	uint16_t	561
data[12]	uint16_t	524
data[13]	uint16_t	563
data[14]	uint16_t	607
data[15]	uint16_t	593
data[16]	uint16_t	423
data[17]	uint16_t	547
data[18]	uint16_t	597
data[19]	uint16_t	609
data[20]	uint16_t	471
data[21]	uint16_t	565
data[22]	uint16_t	607
data[23]	uint16_t	624
data[24]	uint16_t	599
data[25]	uint16_t	564
data[26]	uint16_t	608
data[27]	uint16_t	631
data[28]	uint16_t	547
data[29]	uint16_t	578
data[30]	uint16_t	607
data[31]	uint16_t	647
data[32]	uint16_t	503
data[33]	uint16_t	586

Expression	Type	Value
data2	uint16_t [64]	0x200002
data2[0]	uint16_t	749
data2[1]	uint16_t	359
data2[2]	uint16_t	391
data2[3]	uint16_t	524
data2[4]	uint16_t	423
data2[5]	uint16_t	471
data2[6]	uint16_t	599
data2[7]	uint16_t	547
data2[8]	uint16_t	503
data2[9]	uint16_t	671
data2[10]	uint16_t	567
data2[11]	uint16_t	624
data2[12]	uint16_t	717
data2[13]	uint16_t	611
data2[14]	uint16_t	613
data2[15]	uint16_t	747
data2[16]	uint16_t	577
data2[17]	uint16_t	566
data2[18]	uint16_t	565
data2[19]	uint16_t	563
data2[20]	uint16_t	547
data2[21]	uint16_t	565
data2[22]	uint16_t	564
data2[23]	uint16_t	578
data2[24]	uint16_t	586
data2[25]	uint16_t	603
data2[26]	uint16_t	611
data2[27]	uint16_t	629
data2[28]	uint16_t	643
data2[29]	uint16_t	638
data2[30]	uint16_t	660
data2[31]	uint16_t	668
data2[32]	uint16_t	578

Name : data2[0]

life.augmented