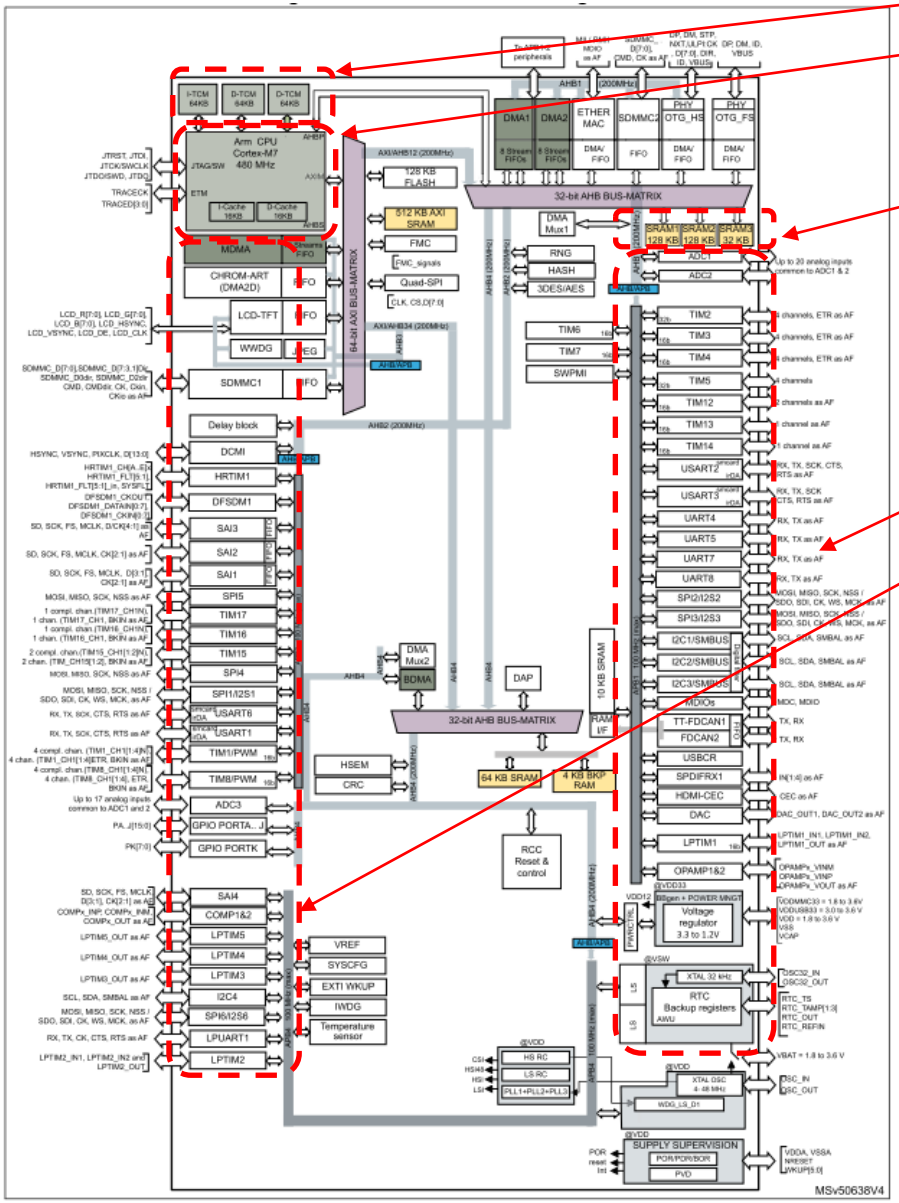
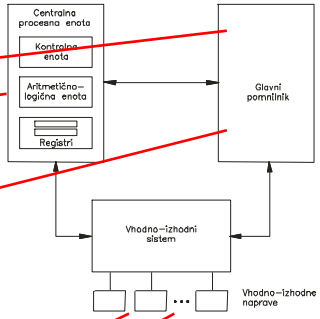
STM32H7

Vhodno / izhodne naprave

USART Serijska komunikacija
z uporabo DMA krmilnika

STM32H750XB





Delo na STM32H7 razvojnem sistemu

Priključitev :

- **Mikro USB** priklop na **daljši stranici (nad LCD, srednji !!!)**

Poseben začetni projekt (github) in info za *STM32H7* (e-učilnica):

- ***dodajanje vsebine (Main.s):***

CubeIDEWorkspace - stm32h7-asm/Core/Src/Main.s - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- CubeIDE_Workspace
  - stm32f4-asm-qemu
- Delo
  - ARM9Template
  - stm32f4-asm (in STM32AsmTemplate)
  - ARM9Template.zip
- Node_V4 (in node_v4)
- Sluzba
  - CAN_IEX_Module
  - ORLab-STM32H7
    - stm32h7-asm
      - Binaries
      - Includes
      - Core
        - Src
          - Main.s
        - Startup
          - startup_stm32h750xbhx.s
      - Debug
      - out
      - makefile
      - README.md
      - STM32H750X.svd
      - STM32H750XBHX_FLASH.ld
      - STM32H750XBHX_RAM.ld
    - README.md
  - RALab-STM32H7
    - stm32h7-asm_RA_LED
    - README.md
  - STM32_USB_Key_AdvDebug
  - STM32_USB_Key_FreeRTOS_AdvDebug
  - STM32CubeIDE_Adv_Debug
  - STM32F4_Discovery_VIN_Projects

Main.s × startup_stm32h750xbhx.s

```
12
13 ////////////////////////////////////////////////////////////////////////////////
14 // Definitions
15 ////////////////////////////////////////////////////////////////////////////////
16 // Definitions section. Define all the registers and
17 // constants here for code readability.
18
19 // Constants
20
21
22 // Start of data section
23         .data
24
25         .align
26
27 STEV1:  .word   0x10    // 32-bitna spr.
28 STEV2:  .word   0x40    // 32-bitna spr.
29 VSOTA:  .word   0       // 32-bitna spr.
30
31
32 // Start of text section
33         .text
34
35         .type  main, %function
36         .global main
37
38         .align
39 main:
40         ldr r0, =STEV1   // Naslov od STEV1 -> r0
41         ldr r1, [r0]     // Vsebina iz naslova v r0 -> r1
42
43         ldr r0, =STEV2   // Naslov od STEV1 -> r0
44         ldr r2, [r0]     // Vsebina iz naslova v r0 -> r2
45
46         add r3,r1,r2     // r1 + r2 -> r3
47
48         ldr r0, =VSOTA   // Naslov od STEV1 -> r0
49         str r3,[r0]      // iz registra r3 -> na naslov v r0
50
51 __end:  b    __end
52
```

--------- Razvojni sistem STM32H750-DK ----------

- STM32H750B-DK Discovery kit with STM32H750XB MCU
- ORLab-STM32H7 - GitHub repozitorij
- User Manual Discovery kit stm32h750xb Uploaded 11/11/22, 10.15
- DataSheet_stm32h750xb Uploaded 11/11/22, 10.16
- Reference Manual rm0433-stm32h750xb Uploaded 11/11/22, 10.17
- Programming_Manual_pm0253-stm32h750xb Uploaded 11/11/22, 10.17
- Errata_es0396-stm32h750xb Uploaded 11/11/22, 10.19

STM32H750B-DK - Schematic

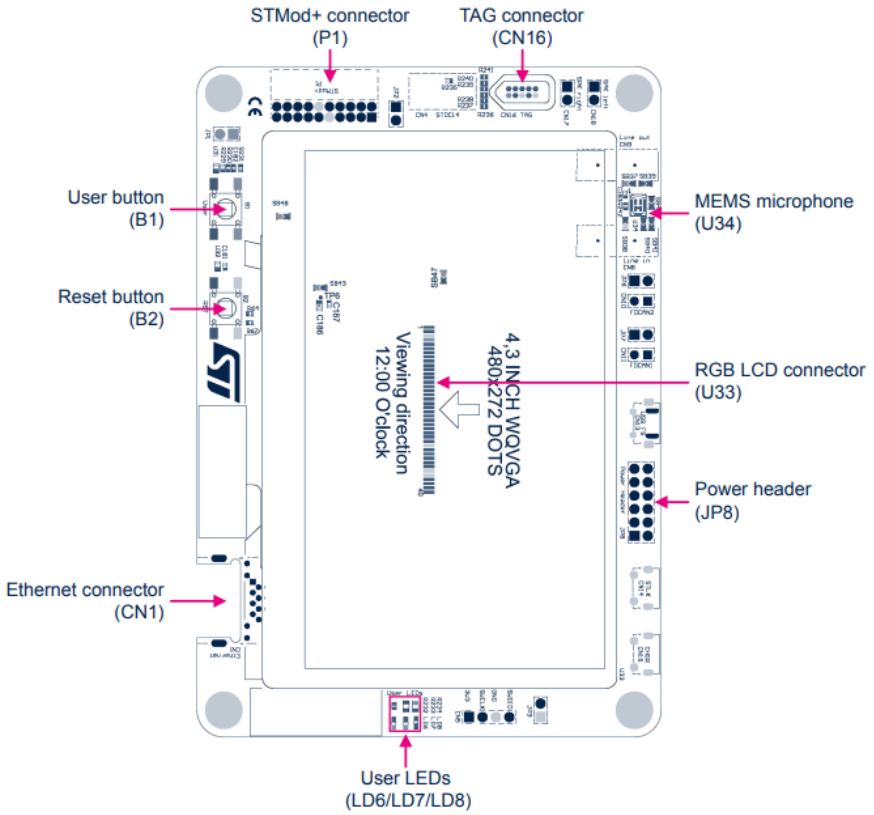
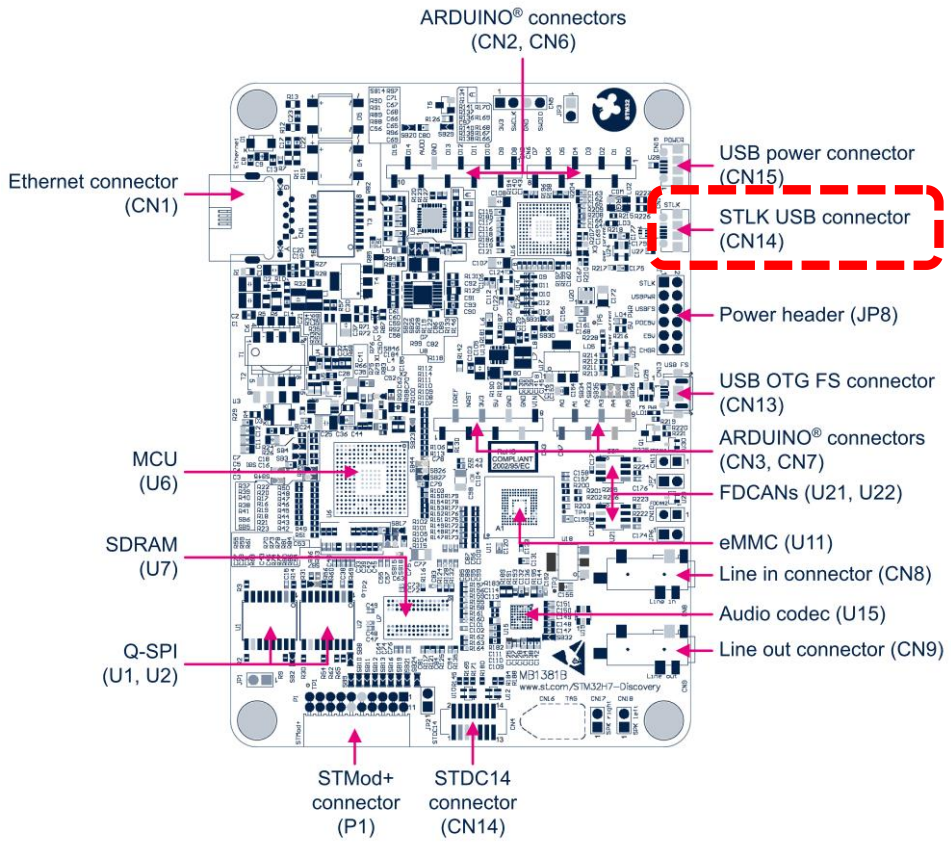
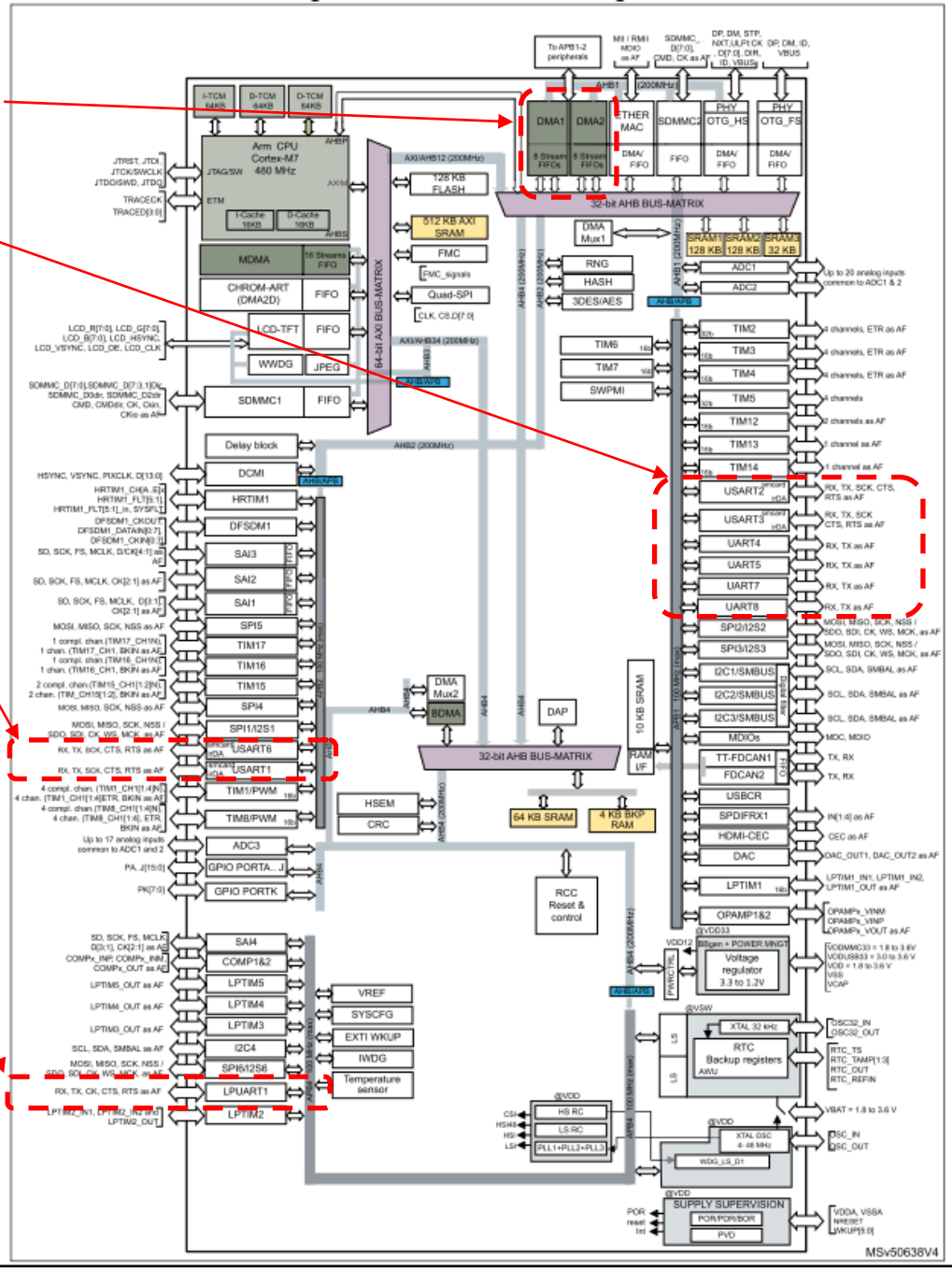


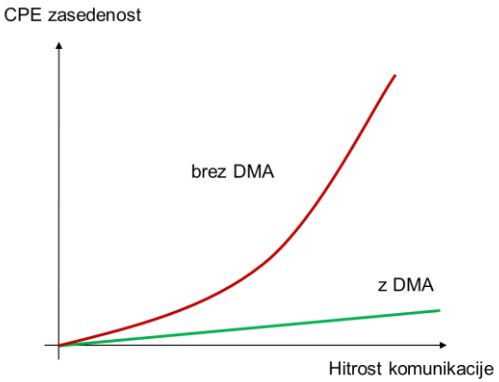
Figure 5. **STM32H745I-DISCO and STM32H750B-DK Discovery board bottom layout**



U(S)ART in DMA naprave

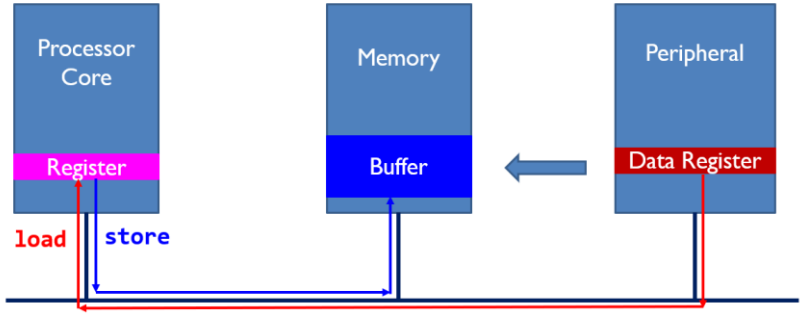


DMA- Splošno



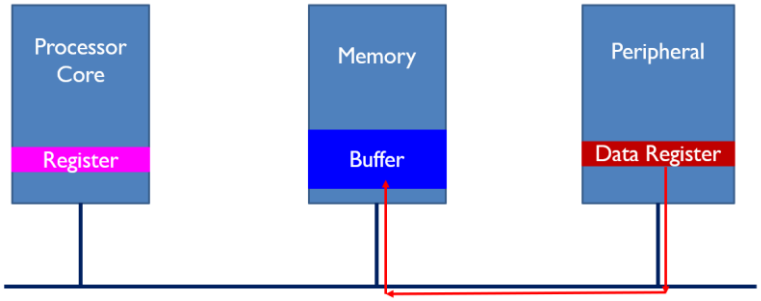
Primer 1: CPE opravlja prenose V/I <-> pomnilnik

- Čaka na zastavico
- ldr Rx, … in str Rx, …



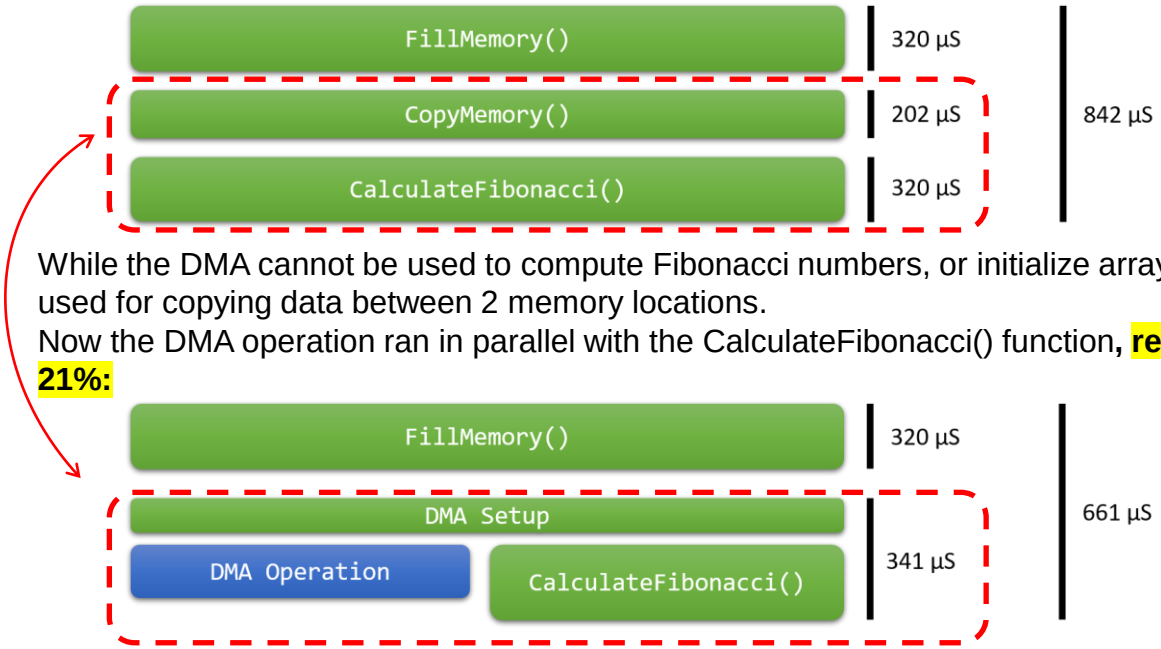
Primer 2: DMA opravlja prenose V/I <-> pomnilnik :

- CPE nastavi DMA krmilnik za prenos :
  - Vrsta prenosa, naprave
  - Naslovi vira in ponora
  - Velikost in število podatkov
  - Sproži prenos
- DMA krmilnik (neodvisno od CPE):
  - Čaka na zastavico
  - Prebere in shrani podatek
  - odšteva preostale podatke in zaključi prenos
- CPE izvaja svoj program

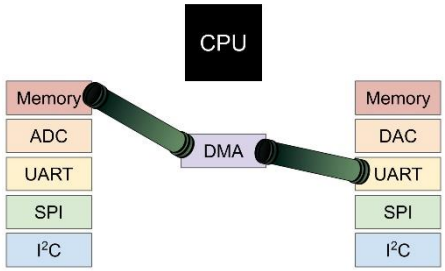


DMA- Splošno

Case1: In this example, filling the first and the third buffer took the exactly the same time, while copying the first buffer to the second one took slightly less time:



While the DMA cannot be used to compute Fibonacci numbers, or initialize arrays with non-constant values, it can be used for copying data between 2 memory locations.
Now the DMA operation ran in parallel with the CalculateFibonacci() function**, reducing the overall program time by 21%:**



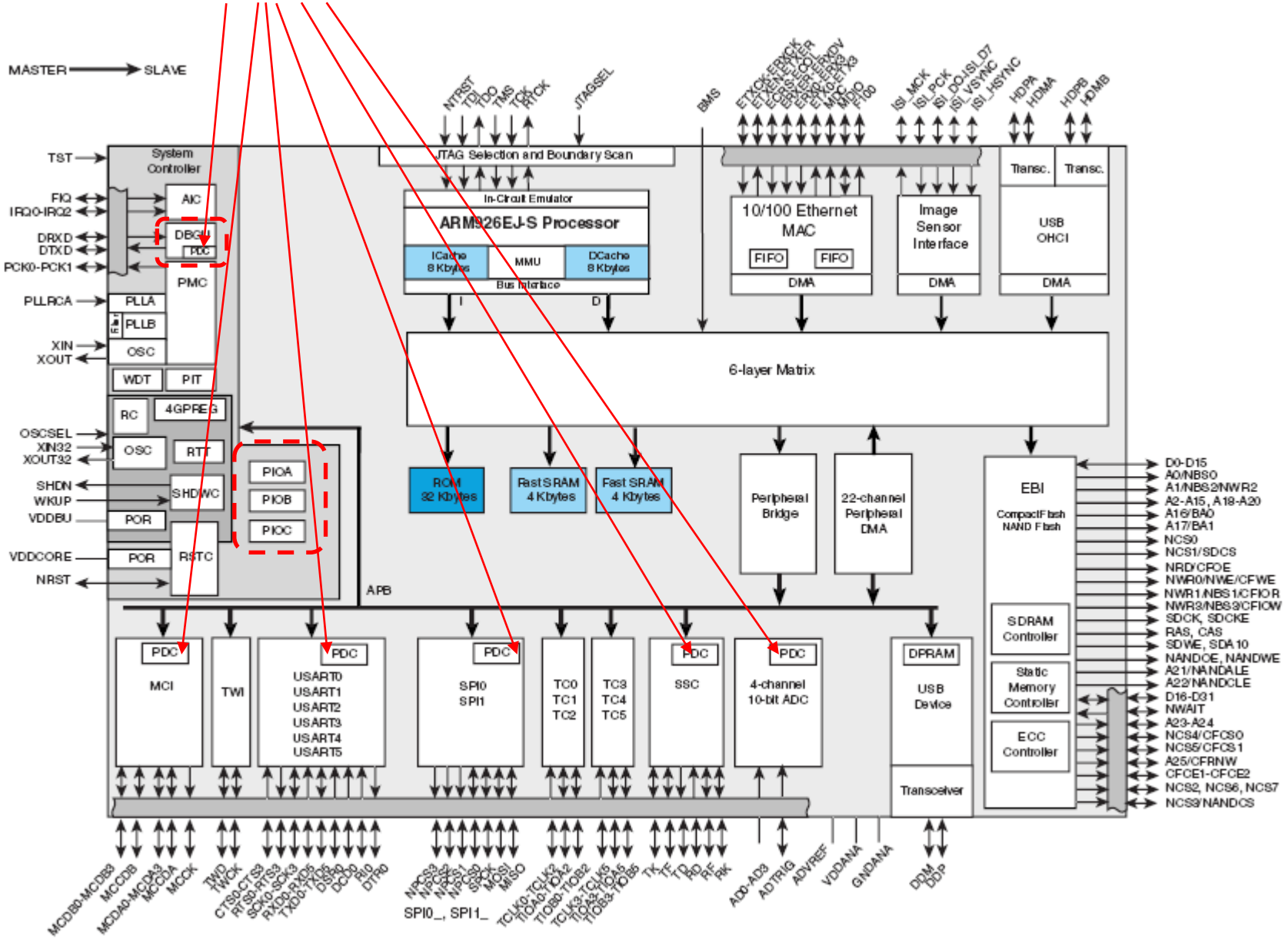
Case2: CRC calculation case:

The results are in:

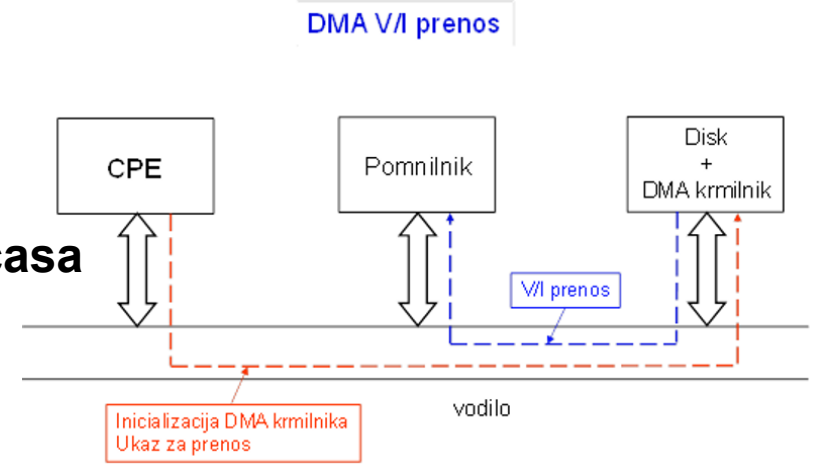
80 DMA CRCs per second.
63 manual CRCs per second

On my processor, **DMA gives a 27% advantage over iterative memory assignment**. I think it is because everything is done with a hardware mover that doesn't have to increment, involve registers, gotos, branch less than, and so on.

FRI-SMS: DMA Krmilnik (PDC – Peripheral DMA Controller)



FRI-SMS: DMA Krmilnik

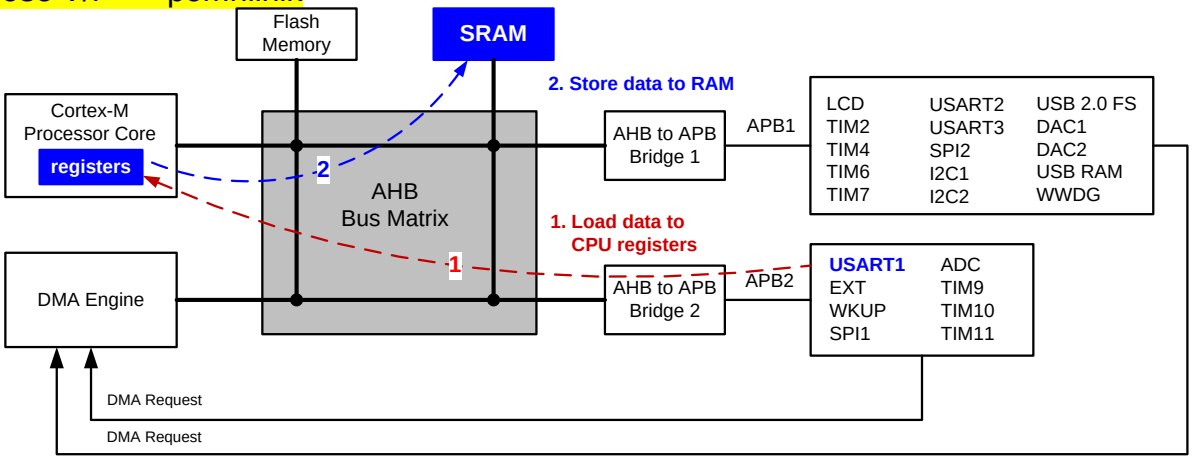


- **Za prenos podatkov, ki ne troši CPE časa**
- **22 DMA kanalov**
- **V/I naprave z DMA kanali**
  - DBGU, SPI, USART, SSC, MCI, EMAC, ISI, ADC
- **Hkratno dvosmerne V/I naprave (full duplex) imajo dva DMA kanala**
- **Enosmerne in izmenično dvosmerne (half duplex) V/I naprave imajo po en DMA kanal**
- **Preprosto programiranje. Potrebno je vpisati le:**
  - začetni naslov in
  - dolžino bloka za prenos
- **DMA krmilnik je dostopen preko naslovnega prostora vsake naprave posebej od odmika 0x100 dalje**

DMA- STM32H7 (stikalna matrika)

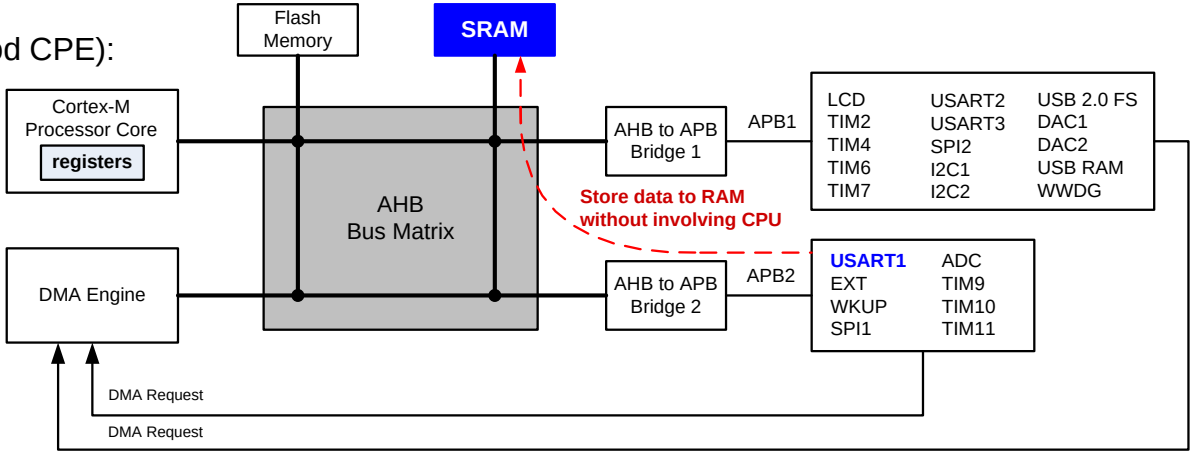
Primer 1 (brez DMA): CPE opravlja prenose V/I <-> pomnilnik

- Čaka na zastavico
- ldr Rx, … in str Rx, …



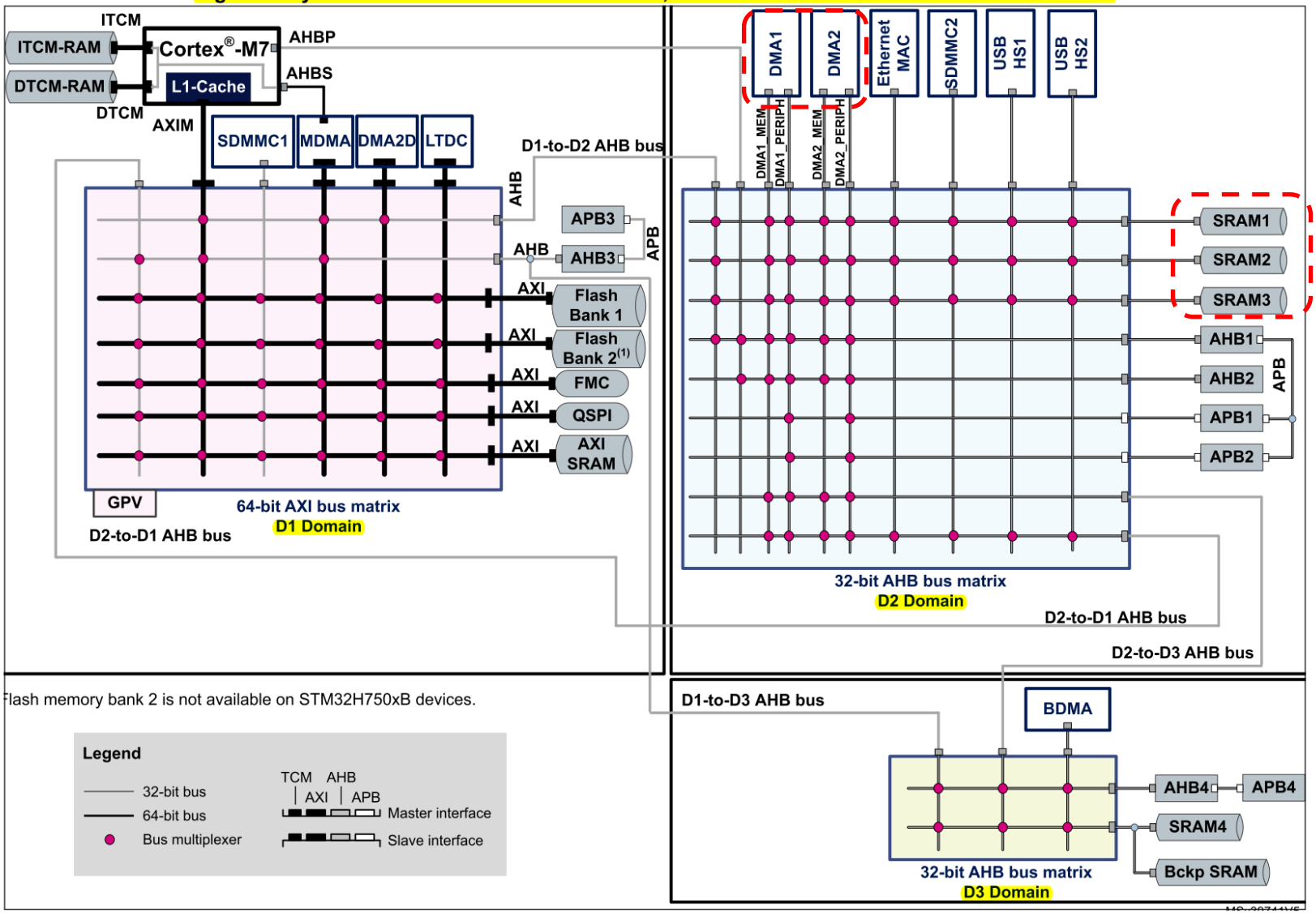
Primer 2 (z DMA): DMA opravlja prenose V/I <-> pomnilnik :

- CPE nastavi DMA krmilnik za prenos :
- DMA krmilnik (neodvisno od CPE):
- CPE izvaja svoj program

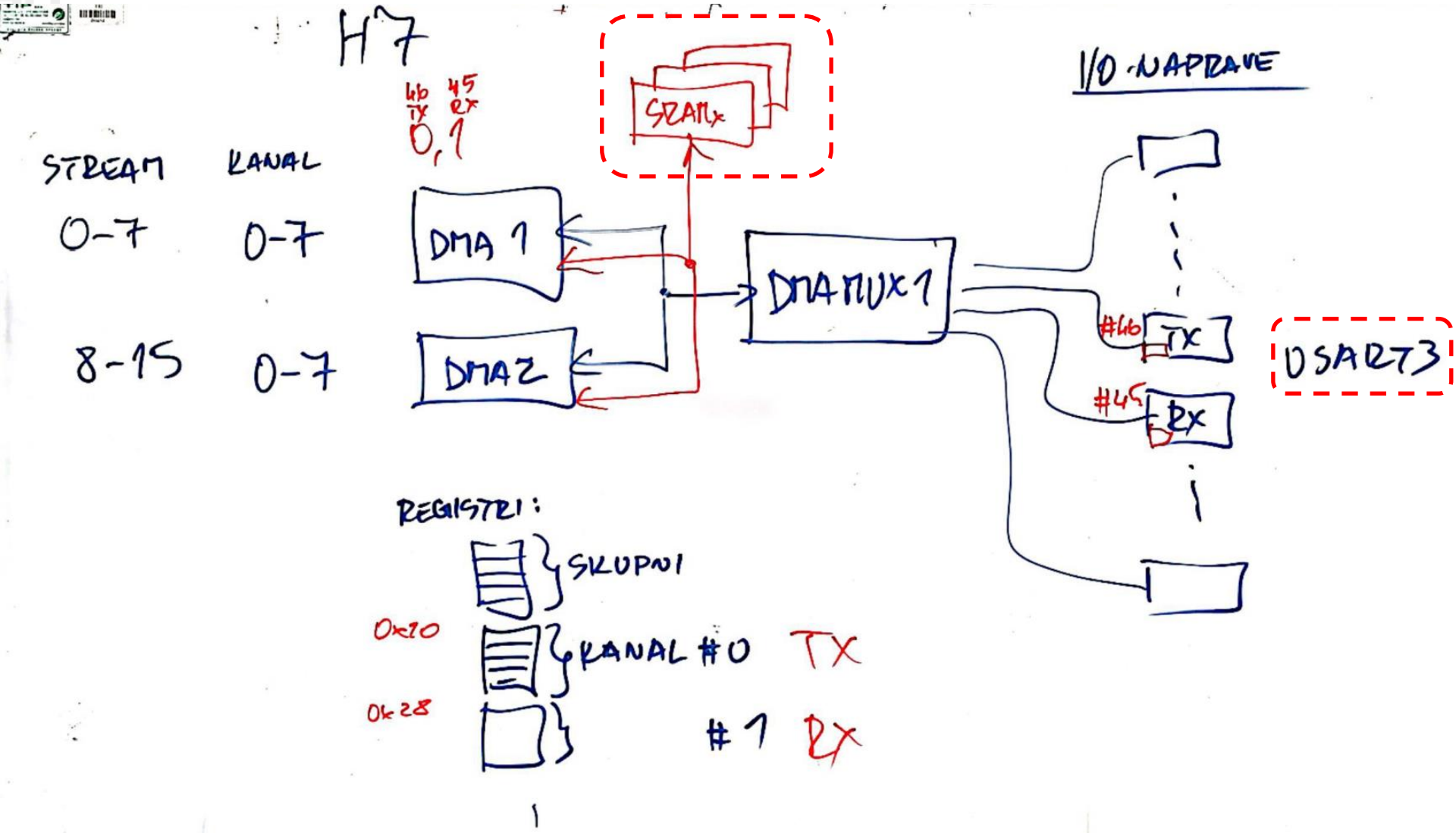


DMA - STM32H7

Figure 1. System architecture for STM32H742xx, STM32H743/53xx and STM32H750xB devices



DMA - STM32H7 + USART3



Viri USART3 : User & Programming manuals, vezalna shema

life.augmented

UM2488

User manual

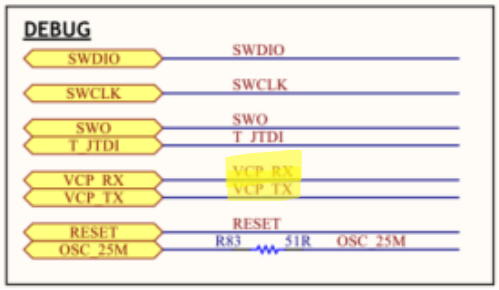
Discovery kits with STM32H745XI and STM32H750XB microcontrollers

6.14 Virtual COM port

The USART3 serial interface is directly available as a virtual COM port of the PC, connected to the STLINK-V3 USB connector (CN14). The virtual COM port settings as the following:

- 115200 baud
- 8-bit data
- No parity
- 1 stop bit
- no flow control

Vezalna shema



life.augmented

RM0433

Reference manual

STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs

Table 8. Register boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Register map
0x40005000 - 0x400053FF	UART5	APB1 (D2)	Section 48.7: USART registers
0x40004C00 - 0x40004FFF	UART4		Section 48.7: USART registers
0x40004800 - 0x40004BFF	USART3		Section 48.7: USART registers
0x40004400 - 0x400047FF	USART2		Section 48.7: USART registers

Table 395. USART/LPUART features

USART modes/features(1)	USART1/2/3/6	UART4/5/7/8	LPUART1
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain and wakeup from low-power mode	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size	16		

1. X = supported.

https://st-onlinetraining.s3.amazonaws.com/STM32H7-Peripheral-USART-interface_%28USART%29/index.html

USART – krmiljenje

1. INIT_USART3 - Inicializacija USART3 naprave (prejšnja LAB vaja)

Potrebni koraki za krmiljenje USART naprave:

1. **Vklop USART3 naprave**
   - **RCC_APB1LENR : b_{18}=1 (USART3 Enable Clock)**
2. **Nastavitev GPIOB priključkov 10,11 na AF7(Alt. Function7)**
   - **RCC_AHB4ENR (Peripheral Clock Register): b_1=1 .. Port B Enable**
   - **GPIOB_MODER (Mode Register): 0b10, AF on pins PB10,PB11**
   - **GPIOB_AFRH (AF Register): 0x07700 AF7 on pins PB10,PB11**
3. **Nastavitev hitrosti delovanja (BaudRate)**
   - **USART3_BRR (BaudRate Register): 64M/115200 ≈ 556**
4. **Sprožitev delovanja**
   - **USART3_CR1 (Control Register 1): `0b1101` TX, RX, USART Enable bits**

~~**5. Delovanje**~~

~~**Oddaja znaka:**~~

DMA omogoča delo z nizi znakov !

- ~~**USART3_ISR: ko TXE=1, vpis znaka v USART3_TDR**~~

~~**Sprejem znaka:**~~

- ~~**USART3_ISR: ko RXNE=1, preberi znak iz USART3_RDR**~~

Vir DMA: Reference manual

Razlike v primerjavi s Cortex M4 (STM32F4):

- **DMAMUX** omogoča poljubne preslikave naprav na 16 DMA kanalov

- DMA napravi (DMA1,DMA2) delujeta samo v SRAM pomnilnik(e)
  - Pozor pri zagonu kode iz RAM-a !!

life.augmented

RM0433
Reference manual

STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs

Table 8. Register boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Register map
0x40024400 - 0x400247FF	Reserved	AHB1 (D2)	Reserved
0x40022000 - 0x400223FF	ADC1 - ADC2		*Section 25.7: ADC common registers*
0x40020800 - 0x40020BFF	DMAMUX1		*Section 17.6: DMAMUX registers*
0x40020400 - 0x400207FF	DMA2		*Section 15.5: DMA registers*
0x40020000 - 0x400203FF	DMA1		*Section 15.5: DMA registers*

USART + DMA – krmiljenje

Potrebni koraki za krmiljenje USART naprave s pomočjo DMA naprave :

1. **INIT_USART3 - Inicializacija USART3 naprave**
   - Enako kot za samostojno delovanje naprave USART3 (OR Vaja 10)
     - za sprejem ali oddajo posameznega znaka

2. **INIT_DMA - Inicializacija DMA in dodatne nastavitve USART3**
   - RCC: vklop DMA1 naprave
   - Izklop DMA kanalov (RX, TX) in njune nastavitve
     - Splošne nastavitve za prenose :
       - smer prenosa (pomn.<-> naprava), povečevanje naslovov v pomnilniku
   - DMAMUX1:
     - Možnost poljubne preslikave med V/I napravami in kanali DMA krmilnika (angl. stream)
     - Nastavitve DMA kanalov 1,0 in povezava z napravo USART3 (št. 45 = RX, št. 46 = TX)
   - Izklop USART3 naprave in vklop DMA krmiljenja ter USART3:
     - Dodatna nastavitev USART3- vklop DMA krmiljenja prenosov na USART3 napravi

3. **SND_DMA, RCV_DMA: Nastavitve DMA za vsak prenos**
   - za sprejem ali oddajo niza znakov
     - Naslov podatkovnega registra V/I naprave, pomnilnika in št. znakov za prenos
     - Brisanje zastavic za prejšnje dogodke
     - Vklop ustreznega kanala DMA (TX ali RX)

2. INIT_DMA - Inicializacija DMA in USART3

RCC: vklop DMA1 naprave

8.7.41 RCC AHB1 Clock Register (RCC_AHB1ENR)

This register can be accessed via two different offset address.

Table 66. RCC_AHB1ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB1ENR	0x0D8	0x0000 0000
RCC_C1_AHB1ENR	0x138	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	USB2OTGHSEN	USB1OTGHSULPIEN	USB1OTGHSEN	Res.	Res.	Res.	Res.	Res.	Res.	USB2OTGHSULPIEN	ETH1RXEN	ETH1TXEN
				rw	rw	rw							rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETH1MACEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12EN	Res.	Res.	Res.	DMA2EN	DMA1EN
rw										rw				rw	rw

Bit 1 **DMA2EN**: DMA2 Clock Enable
Set and reset by software.
0: DMA2 clock disabled (default after reset)
1: DMA2 clock enabled

Bit 0 **DMA1EN**: DMA1 Clock Enable
Set and reset by software.
0: DMA1 clock disabled (default after reset)
1: DMA1 clock enabled

```
// Enable DMA1 Peripheral Clock (bit 0 in AHB1ENR
register)
ldr r6, =RCC_BASE           // Load peripheral clock reg
base address to r6
ldr r5, [r6,#RCC_AHB1ENR]   // Read its content to r5
orr r5, #1                 // Set bit 0 to enable DMA1 clock
str r5, [r6,#RCC_AHB1ENR]   // Store result in peripheral
clock register
```

2. INIT_DMA - Inicializacija DMA in USART3

Izklop DMA kanalov (RX, TX) in njune nastavitve

15.5.5 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 * x, (x = 0 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.	MBURST[1:0]		PBURST[1:0]		TR BUFF	CT	DBM	PL[1:0]	
							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

```
ldr r6, =DMA1_BASE    // Load DMA1 BASE address to r6

// ------- Primer: DMA1 TX Settings
// Disable DMA TX Channel
ldr r5, [r6,#DMA_SxCR_TX] // Read its content to r5
bic r5, #1
str r5, [r6,#DMA_SxCR_TX] // Store result

wt0_EN0:
ldr r5, [r6,#DMA_SxCR_TX] // wait until bit is read as 0
tst r5, #1
bne wt0_EN0
```

Bit 0 **EN**: stream enable / flag stream ready when read low

This bit is set and cleared by software.

0: stream disabled

1: stream enabled

This bit may be cleared by hardware:

- on a DMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB master buses
- when the FIFO threshold on memory AHB port is not compatible with the size of the burst

When this bit is read as 0, the software is allowed to program the configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

Note: Before setting EN bit to 1 to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

```
.equ    DMA_USART3_TX_STREAM, 0    //Channel 0 on DMA1
.equ    DMA_USART3_RX_STREAM, 1    //Channel 1 on DMA1
```

2. INIT_DMA - Inicializacija DMA in USART3

Izklop DMA kanalov (RX, TX) in njune nastavitve

Default nastavitve, ki ustrezajo

15.5.5 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Nastavitve za spremembo

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		TR BUFF	CT	DBM	PL[1:0]	
							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

Bit 10 **MINC**: memory increment mode
This bit is set and cleared by software.
0: memory address pointer is fixed
1: memory address pointer is incremented after each data transfer (increment is done according to MSIZE)
This bit is protected and can be written only if EN = 0.

Bits 7:6 **DIR[1:0]**: data transfer direction
These bits are set and cleared by software.
00: peripheral-to-memory
01: memory-to-peripheral
10: memory-to-memory
11: reserved
These bits are protected and can be written only if EN = 0.

$b_{10}=1$
- povečevanje naslova v pomn.

$b_{7,6}=$
- 01 TX
- 00 RX

```
// ------- Primer: DMA1 TX Settings
// Set MINC (increment memory pointer) and Direction (Memory->Peripheral)
orr r5,r5,#(0b10001 << 6)     // b10=1 and bit7,6 = 01
str r5, [r6,#DMA_SxCR_TX]      // Store result
```

2. INIT_DMA - Inicializacija DMA in USART3

Izklop DMA kanalov (RX, TX) in njune nastavitve

15.5.5 DMA stream x configuration register (DMA_SxCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		TR BUFF	CT	DBM	PL[1:0]	
							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

Default nastavitve, ki ustrezajo (podrobneje)

Bits 24:23 **MBURST[1:0]**: memory burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN = 0.
In direct mode, these bits are forced to 0x0 by hardware as soon as bit EN = 1.

Bits 22:21 **PBURST[1:0]**: peripheral burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN = 0.
In direct mode, these bits are forced to 0x0 by hardware.

Bits 17:16 **PL[1:0]**: priority level
These bits are set and cleared by software.
00: low
01: medium
10: high
11: very high
These bits are protected and can be written only if EN = 0.

Bit 15 **PINCOS**: peripheral increment offset size
This bit is set and cleared by software
0: The offset size for the peripheral address calculation is linked to the PSIZE
1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).
This bit has no meaning if bit PINC = 0.
This bit is protected and can be written only if EN = 0.
This bit is forced low by hardware when the stream is enabled (EN = 1) if the direct mode is selected or if PBURST are different from 00.

Bits 14:13 **MSIZE[1:0]**: memory data size
These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN = 0.
In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as EN = 1.

Bits 12:11 **PSIZE[1:0]**: peripheral data size
These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN = 0.

Bit 9 **PINC**: peripheral increment mode
This bit is set and cleared by software.
0: peripheral address pointer fixed
1: peripheral address pointer incremented after each data transfer (increment done according to PSIZE)
This bit is protected and can be written only if EN = 0.

Bits 7:6 **DIR[1:0]**: data transfer direction
These bits are set and cleared by software.
00: peripheral-to-memory
01: memory-to-peripheral
10: memory-to-memory
11: reserved
These bits are protected and can be written only if EN = 0.

Bit 5 **PFCTRL**: peripheral flow controller
This bit is set and cleared by software.
0: DMA is the flow controller.
1: The peripheral is the flow controller.
This bit is protected and can be written only if EN = 0.
When the memory-to-memory mode is selected (bits DIR[1:0]=10), then this bit is automatically forced to 0 by hardware.

2. INIT_DMA - Inicializacija DMA in USART3

Izklop DMA kanalov (RX, TX) in njune nastavitve

Nastavitev direktnega načina (že nastavljeno na 0)

15.5.10 DMA stream x FIFO control register (DMA_SxFCR)

Address offset: 0x24 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0021

Default nastavitve, ki ustrezajo

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEIE	Res.	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

Bit 2 **DMDIS**: direct mode disable

This bit is set and cleared by software. It can be set by hardware.

0: direct mode enabled

1: direct mode disabled

This bit is protected and can be written only if EN = 0.

This bit is set by hardware if the memory-to-memory mode is selected (DIR bit in DMA_SxCR are 10) and the EN = 1 in DMA_SxCR because the direct mode is not allowed in the memory-to-memory configuration.

```
// ------- Primer: DMA1 TX Settings
// Enable direct mode
ldr r5, [r6,#DMA_SxFCR_TX]   // Read its content to r5
bic r5, #0b100
str r5, [r6,#DMA_SxFCR_TX]     // Store result
```

2. INIT_DMA - Inicializacija DMA in USART3

DMAMUX1:

Nastavitev preslikave DMAMUX1 (naprava <-> stream)

17.6.1 DMAMUX1 request line multiplexer channel x configuration register (DMAMUX1_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

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.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE
					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
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
						rw	rw		rw	rw	rw	rw	rw	rw	rw

RM0433 Block interconnect

Table 101. DMAMUX1, DMA1 and DMA2 connections(1) (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D2	APB1	I2C1	i2c1_rx_dma	dmamux1_req_in33				
			i2c1_tx_dma	dmamux1_req_in34				
D2	APB1	I2C2	i2c2_rx_dma	dmamux1_req_in35				
			i2c2_tx_dma	dmamux1_req_in36				
D2	APB2	SPI1	spi1_rx_dma	dmamux1_req_in37				
			spi1_tx_dma	dmamux1_req_in38				
D2	APB1	SPI2	spi2_rx_dma	dmamux1_req_in39				
			spi2_tx_dma	dmamux1_req_in40				
D2	APB2	USART1	usart1_rx_dma	dmamux1_req_in41				
			usart1_tx_dma	dmamux1_req_in42				
D2	APB1	USART2	usart2_rx_dma	dmamux1_req_in43				
			usart2_tx_dma	dmamux1_req_in44				
D2	APB1	USART3	usart3_rx_dma	dmamux1_req_in45				
			usart3_tx_dma	dmamux1_req_in46				

RM0433 Rev 7 599/3319

Bits 6:0 **DMAREQ_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

RM0433 DMA request multiplexer (DMAMUX)

Table 121. DMAMUX1: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux1_req_gen0	44	usart2_tx_dma	87	sai1a_dma
2	dmamux1_req_gen1	45	usart3_rx_dma	88	sai1b_dma
3	dmamux1_req_gen2	46	usart3_tx_dma	89	sai2a_dma

2. INIT_DMA - Inicializacija DMA in USART3

DMAMUX1:

Nastavitev preslikave DMAMUX1 (naprava <-> stream)

17.6.1 DMAMUX1 request line multiplexer channel x configuration register (DMAMUX1_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

Reset value: 0x0000 0000

Bits 6:0 **DMAREQ_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE
					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
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
						rw	rw		rw	rw	rw	rw	rw	rw	rw

Table 121. DMAMUX1: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux1_req_gen0	44	usart2_tx_dma	87	sai1a_dma
2	dmamux1_req_gen1	45	usart3_rx_dma	88	sai1b_dma
3	dmamux1_req_gen2	46	usart3_tx_dma	89	sai2a_dma


```
// ------------- DMAMUX1 Settings
// Set channels to devices translations (multiplexing)

ldr r6, =DMAMUX1_BASE          // Load reg base address to r6

mov r5,#46  // USART3_TX DMA Device Nr. is 46
str r5, [r6, DMAMUX1_C0CR]// DMAREQ for Channel 0 to USART3_TX

mov r5,#45  // USART3_Rx DMA Device Nr. is 45
str r5, [r6, DMAMUX1_C1CR]// DMAREQ for Channel 1 to USART3_RX
```

2. INIT_DMA - Inicializacija DMA in USART3

Izklop USART3 naprave in vklop DMA krmiljenja ter USART3:

Izklop, vklop delovanja USART 3 (za konfiguracijo CR3)

48.7.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		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
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

2. INIT_DMA - Inicializacija DMA in USART3

Izklop USART3 naprave in vklop DMA krmiljenja ter USART3:

Vklop DMA prenosov na USART3

48.7.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
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
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

```
// ------------- USART3 Settings
ldr r6, =USART3_BASE       // Load USART3 BASE address to r1

 // Disable USART3
ldr r5, [r6,#USART_CR1]   // Read its content to r5
bic r5, #1
str r5, [r6,#USART_CR1]   // Store result

// Enable DMA Transmit and Receive for USART3
ldr r5, [r6, #USART_CR3]
orr r5, #(0b11<<6)          // Set bits 7 and 6 to enable DMAT and DMAR bits
str r5, [r6,#USART_CR3]    // Store result

// Enable USART3
ldr r6, =USART3_BASE       // Load USART3 BASE address to r6
ldr r5, [r6,#USART_CR1]   // Read its content to r5
orr r5, r5, #1
str r5, [r6,#USART_CR1]   // Store result
```

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

3. RCV_DMA ,SND_DMA: Nastavitve DMA

Nastavitev naslova USART3 naprave (Data Register)

15.5.7 DMA stream x peripheral address register (DMA_SxPAR)

Address offset: 0x18 + 0x18 * x, (x = 0 to 7)

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 is read/written.
These bits are write-protected and can be written only when bit EN = 0 in DMA_SxCR.

Naslov USART3 Data registra:
USART3_BASE+USART_RDR

```
ldr r6, =DMA1_BASE          // Load reg base address to r6

WAIT_EN: // Wait EN bit to become zero
ldr r5, [r6, #DMA_SxCR_RX]
tst r5, #1
bne WAIT_EN

//  Receive (RX) DMA Init
ldr r5, =USART3_BASE+USART_RDR// RX peripheral address to r5
str r5, [r6,#DMA_SxPAR_RX]   // Store peripheral DMA pointer
str r0, [r6,#DMA_SxM0AR_RX]   // Store address pointer in r0
str r1, [r6,#DMA_SxSNDTR_RX]   // Store number of units in r1
```

3. RCV_DMA ,SND_DMA: Nastavitve DMA

Nastavitev naslova v pomnilniku (za prenos)

15.5.8 DMA stream x memory 0 address register (DMA_SxM0AR)

Address offset: 0x1C + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Naslov v pomnilniku (začetek niza v r0)

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 is read/written.

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

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

```
ldr r6, =DMA1_BASE          // Load reg base address to r6

//  Receive (RX) DMA Init
ldr r5, =USART3_BASE+USART_RDR// RX peripheral address to r5
str r5, [r6,#DMA_SxPAR_RX]   // Store peripheral DMA pointer
str r0, [r6,#DMA_SxM0AR_RX]   // Store address pointer in r0
str r1, [r6,#DMA_SxSNDTR_RX]   // Store number of units in r1
```

3. RCV_DMA ,SND_DMA: Nastavitve DMA

Nastavitev števila podatkov (za prenos)

15.5.6 DMA stream x number of data register (DMA_SxNDTR)

Address offset: 0x14 + 0x18 * x, (x = 0 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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: number of data items to transfer (0 up to 65535)

This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:

- when the stream is configured in circular mode.
- when the stream is enabled again by setting EN bit to 1.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

Število znakov v r1

```
ldr r6, =DMA1_BASE          // Load reg base address to r6

//  Receive (RX) DMA Init
ldr r5, =USART3_BASE+USART_RDR// RX peripheral address to r5
str r5, [r6,#DMA_SxPAR_RX]   // Store peripheral DMA pointer
str r0, [r6,#DMA_SxM0AR_RX]   // Store address pointer in r0
str r1, [r6,#DMA_SxSNDTR_RX]   // Store number of units in r1
```

3. RCV_DMA ,SND_DMA: Nastavitve DMA

Brisanje prejšnjih DMA dogodkov

15.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2
				w	w	w	w		w	w	w	w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[3:0]**: stream x clear transfer complete interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register.

```
ldr r6, =DMA1_BASE          // Load reg base address to r6


// Clear flags in Status register
mov r5,#(0b111101<<6)           // clear flags for Ch1 in ISR
str r5, [r6,# DMA_LIFCR]       // Store
```

3. RCV_DMA ,SND_DMA: Nastavitve DMA

Vklop delovanja (streama) DMA krmilnika (EN=0)

15.5.5 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 * x, (x = 0 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.	MBURST[1:0]		PBURST[1:0]		TR BUFF	CT	DBM	PL[1:0]	
							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

Bit 0 **EN**: stream enable / flag stream ready when read low

This bit is set and cleared by software.

0: stream disabled

1: stream enabled

This bit may be cleared by hardware:

- on a DMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB master buses
- when the FIFO threshold on memory AHB port is not compatible with the size of the burst

When this bit is read as 0, the software is allowed to program the configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

Note: Before setting EN bit to 1 to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

```
//  Enable DMA Channel
ldr r5, [r6, #DMA_SxCR_RX]
orr r5, r5, #1
str r5, [r6, #DMA_SxCR_RX]// Enable channel
```

3. RCV_DMA ,SND_DMA: Nastavitve DMA

Čakanje na zaključek prenosov

Sprejem (RCV_DMA):

```
// ---------   RCV_DMA
// Wait for the end of reception (TCIF)
      ldr r6, =DMA1_BASE          // Load reg base address to r6

WAIT_RC:
      ldr r5, [r6, #DMA_LISR]
      tst r5, #(1 << 11)        // TCIF1 flag
      beq WAIT_RC
```

Oddaja (SND_DMA):

```
// ---------   SND_DMA
// Wait for the end of transmission
    ldr r6, =USART3_BASE

WAIT_TC:
    ldr r5, [r6, #USART_ISR]
    tst r5, #(1 << 6)              // Test TC bit
    beq WAIT_TC
```

DMA – stanje , nastavitve

Table 114. DMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	DMA_LISR	Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2	Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0004	DMA_HISR	Res.	Res.	Res.	Res.	TCIF7	HTIF7	TEIF7	DMEIF7	Res.	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Res.	FEIF6	Res.	Res.	Res.	Res.	TCIF5	HTIF5	TEIF5	DMEIF5	Res.	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Res.	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0008	DMA_LIFCR	Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	TEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2	Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x000C	DMA_HIFCR	Res.	Res.	Res.	Res.	CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Res.	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Res.	CFEIF6	Res.	Res.	Res.	Res.	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Res.	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Res.	CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0010	DMA_S0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	DMA_S0NDTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	DMA_S0PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA_S0M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	DMA_S0M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	DMA_S0FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEIE	Res.	FS[2:0]			DMDIS	FTH[1:0]		
	Reset value																								0		1	0	0	0	0	1	
0x0028	DMA_S1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Status reg.

Clear flags reg.

DMA

Ctrl reg. #0

Število znakov

Naslov DReg

Naslov Pomn.

Ch0

Ctrl reg. #1

Ch1

...

DMA – projekt

STM32H750XBHX_RAM.ld:

- *Dodamo sekcijo .sram*

```
/* Define output sections */
SECTIONS
{

      …
      /* used by the startup to initialize data */
      _sisram = LOADADDR(.sram); /* this will be start address of init values in RAM_EXEC */

      /* Initialized data sections goes into SRAM, have to copy content manually */
      .sram :
      {
      . = ALIGN(4);
      _ssram = .; /* create a global symbol at sram start */
      KEEP(*(.sram)) /* .data sections */
      *(.sram*) /* .data* sections */


      . = ALIGN(4);
      _esram = .; /* define a global symbol at sram end */

      } >RAM_D2 AT> RAM_EXEC
      …

}
```

DMA – program

Naslovi registrov:

```
//-----------------------------
// DMA related definitions
//-----------------------------

// AHB2ENR register offset is 0xDC (for enabling SRAMs)
.equ RCC_AHB2ENR, 0xDC // RCC AHB2ENR periph. clk reg.

// AHB1ENR register offset is 0xE8
.equ RCC_AHB1ENR, 0xD8 // RCC AHB1ENR periph. clk reg.


// DMAMUX1 base address is 0x40020800
.equ DMAMUX1_BASE, 0x40020800 // DMAMUX1 base address


.equ DMAMUX1_C0CR, 0x00 // CR for Channel 0
.equ DMAMUX1_C1CR, 0x04 // CR for Channel 1


// DMA1 base address is 0x40020000
.equ DMA1_BASE, 0x40020000 // DMA1 base address
// DMA2 base address is 0x40020400
.equ DMA2_BASE, 0x40020400 // DMA2 base address


.equ DMA_USART3_TX_STREAM, 0 //Channel 0 on DMA1
.equ DMA_USART3_RX_STREAM, 1 //Channel 1 on DMA1
```

```
// DMA Registers definitions
.equ DMA_LISR, 0x00
.equ DMA_HISR, 0x04
.equ DMA_LIFCR, 0x08
.equ DMA_HIFCR, 0x0C


.equ DMA_SxCR_TX,0x10 + 0x18 * DMA_USART3_TX_STREAM
.equ DMA_SxFCR_TX,0x24 + 0x18 * DMA_USART3_TX_STREAM
.equ DMA_SxSNDTR_TX,0x14 + 0x18 * DMA_USART3_TX_STREAM
.equ DMA_SxPAR_TX,0x18 + 0x18 * DMA_USART3_TX_STREAM
.equ DMA_SxM0AR_TX,0x1C + 0x18 * DMA_USART3_TX_STREAM


.equ DMA_SxCR_RX,0x10 + 0x18 * DMA_USART3_RX_STREAM
.equ DMA_SxFCR_RX,0x24 + 0x18 * DMA_USART3_RX_STREAM
.equ DMA_SxSNDTR_RX,0x14 + 0x18 * DMA_USART3_RX_STREAM
.equ DMA_SxPAR_RX,0x18 + 0x18 * DMA_USART3_RX_STREAM
.equ DMA_SxM0AR_RX,0x1C + 0x18 * DMA_USART3_RX_STREAM
```

DMA – program (zagon z RAM povez. skripto)

Main.s - spremembe:

- *Dodamo spremenljivke v sekcijo .sram*

```
// Start of sram section
      .section .sram,"a",%progbits
      .align

NIZ1: .space 12
NIZ2: .asciz "Testni niz!" // 12 bytes
       .equ NIZ_LEN, 12
```

- *Za delovanje moramo vklopiti tudi SRAM pomnilnike :*

```
/* Enable SRAMs and copy initial values only with RAM Linker script */
// Enable SRAMs internal memories (for DMA1)
bl SRAM_ENABLE
```

- *Startup koda ne vsebuje obravnave te sekcije, zato moramo za kopiranje začetnih vrednosti poskrbeti sami:*

```
// Copy initial values for .sram section (from .text to SRAM)
ldr r0,=_sisram
ldr r1,=_ssram
ldr r2,=_esram
bl MEM_COPY
```

```
MEM_COPY:
   push {r3, lr}

copy:
   ldr r3,[r0],#4
   str r3,[r1],#4

   cmp r1,r2
   blo copy

   pop {r3, pc}
```

DMA – program (zagon z RAM povez. skripto)

Main.s:

```
// Comment following line when FLASH Linker Script is used
#define RAM_LinkScript
…
#ifdef RAM_LinkScript
// Start of sram section
            .section .sram,"a",%progbits
#endif

            .align
NIZ1: .space 12
            .align
NIZ2: .asciz "Testni niz!" // 12 bytes

.equ NIZ_LEN, 12


main:

#ifdef RAM_LinkScript

// Relocating Vector table to RAM (only for RAM Linker Script)
            bl RELLOC_VECTBL

/* Enable SRAMs and copy initial values only with RAM Linker script */
// Enable SRAMs internal memories (for DMA1)
           bl SRAM_ENABLE

// Copy initial values for .sram section (from .text to SRAM)
           ldr r0,=_sisram
           ldr r1,=_ssram
           ldr r2,=_esram
           bl MEM_COPY

#endif
```

```
RELLOC_VECTBL:

push {r0, r1, lr}

ldr r1, =VTOR // Set Vector table addr. to
0x24000000
ldr r0, =0x24000000
str r0, [r1]


pop {r0, r1, pc}
```

```
SRAM_ENABLE:
push {r5, r6, lr}

// Enable internal SRAMs (bits 31,30,29 in AHBeENR register)
ldr r6, =RCC_BASE // Load periph. clk reg base address to r6
ldr r5, [r6,#RCC_AHB2ENR] // Read its content to r5
orr r5, #(0b111 << 29) // Set bits 31,30,29 to 1 to enable
SRAMs clock
str r5, [r6,#RCC_AHB2ENR] // Store result in periph. clk
register

pop {r5, r6, pc}
```

```
MEM_COPY:
   push {r3, lr}

copy:
   ldr r3,[r0],#4
   str r3,[r1],#4

   cmp r1,r2
   blo copy

   pop {r3, pc}
```

Vklop SRAM pomnilnikov (DMA1,2 delujeta le v tej domeni)

8.7.42 RCC AHB2 Clock Register (RCC_AHB2ENR)

This register can be accessed via two different offset address.

Table 67. RCC_AHB2ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB2ENR	0x0DC	0x0000 0000
RCC_C1_AHB2ENR	0x13C	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM3EN	SRAM2EN	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2EN	Res.	Res.	RNGEN	HASHEN	CRYPTEN	Res.	Res.	Res.	DCMIEN
						rw			rw	rw	rw				rw

Bit 31 **SRAM3EN**: SRAM3 block enable
Set and reset by software.
When set, this bit indicates that the SRAM3 is allocated by the CPU. It causes the D2 domain to take into account also the CPU operation modes, i.e. keeping D2 domain in DRun when the CPU is in CRun.
0: SRAM3 interface clock is disabled. (default after reset)
1: SRAM3 interface clock is enabled.

Bit 30 **SRAM2EN**: SRAM2 block enable
Set and reset by software.
When set, this bit indicates that the SRAM2 is allocated by the CPU. It causes the D2 domain to take into account also the CPU operation modes, i.e. keeping D2 domain in DRun when the CPU is in CRun.
0: SRAM2 interface clock is disabled. (default after reset)
1: SRAM2 interface clock is enabled.

Bit 29 **SRAM1EN**: SRAM1 block enable
Set and reset by software.
When set, this bit indicates that the SRAM1 is allocated by the CPU. It causes the D2 domain to take into account also the CPU operation modes, i.e. keeping D2 domain in DRun when the CPU is in CRun.
0: SRAM1 interface clock is disabled. (default after reset)
1: SRAM1 interface clock is enabled.

```
SRAM_ENABLE:
push {r5, r6, lr}

// Enable internal SRAMs (bits 31,30,29 in AHBeENR register)
ldr r6, =RCC_BASE // Load periph. clk reg base address to r6
ldr r5, [r6,#RCC_AHB2ENR] // Read its content to r5
orr r5, #(0b111 << 29) // Set bits 31,30,29 to 1 to enable SRAMs clock
str r5, [r6,#RCC_AHB2ENR] // Store result in periph. clk register

pop {r5, r6, pc}
```

DMA – program (glavna zanka)

Main.s:

```
bl INIT_DMA // Priprava DMA naprave za prenose preko USART3

// Main loop for USART3+DMA Echo test for blocks of 12 characters with change to upper case
loop:

	ldr r0,=NIZ2
	mov r1,#NIZ_LEN
	bl SND_DMA

	bl LED_ON // Vklop LED diode

	mov r0,#500
	// bl DELAY // Zakasnitev SW Delay: r0 x 1msec
	bl DELAYTC // Zakasnitev SysTick : r0 x 1msec

	ldr r0,=NIZ1
	mov r1,#NIZ_LEN
	bl RCV_DMA

	ldr r0,=NIZ1
	ldr r1,=NIZ2
	ldr r2, =NIZ_LEN
	bl CHANGE

	bl LED_OFF // Izlop LED diode

	mov r0,#500
	// bl DELAY // Zakasnitev SW Delay: r0 x 1msec
	bl DELAYTC // Zakasnitev SysTick : r0 x 1msec


	b loop // skok na vrstico loop:
```

```
CHANGE:
  push {r3-r4,lr}


ch_zanka:
  ldrb r4, [r0], #1
  bic r3, r4, #0b100000 // zbrisi b5


  cmp r3, #'A'
  blo pisi

  cmp r3, #'Z'
  bhi pisi

  eor r4, r4, #0b100000 // spremeni crko

pisi:
  strb r4, [r1], #1 /* shranimo v niz2*/

  subs r2, r2, #1
  bne ch_zanka

  pop {r3-r4,pc}
```

DMA_INIT (1/3 – DMA1, DMA TX):

```
INIT_DMA:

  push {r5, r6, lr}

//-----------------------------------------
// ------------- DMA1 Settings

// Enable DMA1 Peripheral Clock (bit 0 in AHB1ENR register)
  ldr r6, =RCC_BASE // Load peripheral clock reg base address to r6
  ldr r5, [r6,#RCC_AHB1ENR] // Read its content to r5
  orr r5, #1 // Set bit 0 to enable DMA1 clock
  str r5, [r6,#RCC_AHB1ENR] // Store result in peripheral clock register

  ldr r6, =DMA1_BASE // Load DMA1 BASE address to r6

// ------- DMA1 TX Settings
// Disable DMA TX Channel
  ldr r5, [r6,#DMA_SxCR_TX] // Read its content to r5
  bic r5, #1
  str r5, [r6,#DMA_SxCR_TX] // Store result

wt0_EN0:
  ldr r5, [r6,#DMA_SxCR_TX] // wait until bit is read as 0
  tst r5, #1
  bne wt0_EN0

// Set MINC (increment memory pointer) and Direction (Memory->Peripheral)
  orr r5,r5,#(0b10001 << 6) // b10=1 and bit7,6 = 01
  str r5, [r6,#DMA_SxCR_TX] // Store result

// Enable direct mode
  ldr r5, [r6,#DMA_SxFCR_TX] // Read its content to r5
  bic r5, #0b100
  str r5, [r6,#DMA_SxFCR_TX] // Store result
```

DMA_INIT (2/3 – DMA TX, DMAMUX):

```
// ------- DMA1 RX Settings
// Disable DMA RX Channel
  ldr r5, [r6,#DMA_SxCR_RX] // Read its content to r5
  bic r5, #1
  str r5, [r6,#DMA_SxCR_RX] // Store result

wt1_EN0:
  ldr r5, [r6,#DMA_SxCR_RX] // wait until bit is read as 0
  tst r5, #1
  bne wt1_EN0

// Set MINC (increment memory pointer) and Direction (Memory<-Peripheral)
  orr r5,r5,#(0b10000 << 6) // b10=1 and bit7,6 = 00
  str r5, [r6,#DMA_SxCR_RX] // Store result

// Enable direct mode
  ldr r5, [r6,#DMA_SxFCR_RX] // Read its content to r5
  bic r5, #0b100
  str r5, [r6,#DMA_SxFCR_RX] // Store result

//-------------------------------------------
// ------------- DMAMUX1 Settings
// Set channels to devices translations (multiplexing)

  ldr r6, =DMAMUX1_BASE // Load reg base address to r6

  mov r5,#46 // USART3_TX DMA Device Nr. is 46
  str r5, [r6, DMAMUX1_C0CR] // DMAREQ for Channel 0 to USART3_TX
  mov r5,#45 // USART3_Rx DMA Device Nr. is 45
  str r5, [r6, DMAMUX1_C1CR] // DMAREQ for Channel 1 to USART3_RX
```

DMA_INIT (3/3 – USART3+DMA):

```
//-----------------------------------------------
// ------------- USART3 Settings
  ldr r6, =USART3_BASE // Load USART3 BASE address to r1


// Disable USART3
  ldr r5, [r6,#USART_CR1] // Read its content to r5
  bic r5, #1
  str r5, [r6,#USART_CR1] // Store result


// Enable DMA Transmit and Receive for USART3
  ldr r5, [r6, #USART_CR3]
  orr r5, #(0b11<<6) // Set bits 7 and 6 to enable DMAT and DMAR bits
  str r5, [r6,#USART_CR3] // Store result


// Enable USART3
  ldr r6, =USART3_BASE // Load USART3 BASE address to r6
  ldr r5, [r6,#USART_CR1] // Read its content to r5
  orr r5, r5, #1
  str r5, [r6,#USART_CR1] // Store result

  pop {r5, r6, pc}
```

DMA – oddaja

Potrebni koraki za krmiljenje DMA naprave (SND_DMA):

1. **Počakaj EN bit = 0 v DMA_SxCR_TX** registru (zaključek prejšnjega prenosa)
   - **DMA_SxCR_TX** **b_0=0 (Stream disabled)**
2. **Nastavitve naslovov**
   - **DMA_SxPAR_TX** **naslov DR registra (**USART3_BASE+USART_TDR)
   - **DMA_SxM0AR_TX** **naslov v pomnilniku v r0**
   - **DMA_SxSNDTR_TX** **število znakov v r1**
3. **Brisanje zastavic v statusnem registru DMA_LIFCR :**
   - Clear all bits for channel #0
     - `0b111101 -> DMA_LIFCR`
4. **Vklop DMA kanala**
   - **DMA_SxCR_TX** **b_0=1 (Stream enabled)**

5. **Čakanje na konec prenosa :**
   - **USART_ISR** TC = 1 **(DMA označi konec prenosa)**

DMA – oddaja (SND_DMA)

```
SND_DMA:
push {r5, r6, lr}

ldr r6, =DMA1_BASE // Load reg base address to r6

WAIT_EN1: // Wait EN bit to become zero
ldr r5, [r6, #DMA_SxCR_TX]
tst r5, #1
bne WAIT_EN1

ldr r6, =DMA1_BASE // Load reg base address to r6
// Transmit (TX) DMA Init
ldr r5, =USART3_BASE+USART_TDR // RX peripheral address to r5
str r5, [r6,#DMA_SxPAR_TX] // Store result in peripheral DMA
pointerclock register
str r0, [r6,#DMA_SxM0AR_TX] // Store address pointer
str r1, [r6,#DMA_SxSNDTR_TX] // Store result in peripheral DMA
pointerclock register

// Clear flags in Status register
mov r5, #0b111101 // Clear all bits for channel 0
str r5, [r6,#DMA_LIFCR] // Store result in peripheral DMA
pointerclock register

// Enable DMA Channel
ldr r6, =DMA1_BASE // Load reg base address to r6
ldr r5, [r6, #DMA_SxCR_TX]
orr r5, r5, #1
str r5, [r6, #DMA_SxCR_TX] // Enable channel
```

```
// Wait for the end of transmission
ldr r6, =USART3_BASE

WAIT_TC:
ldr r5, [r6, #USART_ISR]
tst r5, #(1 << 6) // Test TC bit
beq WAIT_TC


pop {r5, r6, pc}
```

DMA – sprejem

Potrebni koraki za krmiljenje DMA naprave (RCV_DMA):

1. **Počakaj EN bit = 0 v DMA_SxCR_RX** registru (zaključek prejšnjega prenosa)
   - **DMA_SxCR_RX** **b_0=0 (Stream disabled)**
2. **Nastavitve naslovov**
   - **DMA_SxPAR_RX** **naslov DR registra (**USART3_BASE+USART_RDR)
   - **DMA_SxM0AR_RX** **naslov v pomnilniku v r0**
   - **DMA_SxSNDTR_RX** **število znakov v r1**
3. **Brisanje zastavic v statusnem registru DMA_LIFCR :**
   - Clear all bits for channel #0
     - `(0b111101<<6) -> DMA_LIFCR`
4. **Vklop DMA kanala**
   - **DMA_SxCR_RX** **b_0=1 (Stream enabled)**

5. **Čakanje na konec prenosa :**
   - **DMA_LISR** TCIF1 = 1 **(DMA označi konec prenosa)**

DMA – sprejem (RCV_DMA)

```
RCV_DMA:
push {r5, r6, lr}

ldr r6, =DMA1_BASE // Load reg base address to r6

WAIT_EN: // Wait EN bit to become zero
ldr r5, [r6, #DMA_SxCR_RX]
tst r5, #1
bne WAIT_EN

ldr r6, =DMA1_BASE // Load reg base address to r6

// Receive (RX) DMA Init
ldr r5, =USART3_BASE+USART_RDR // RX peripheral address to r5
str r5, [r6,#DMA_SxPAR_RX] // Store peripheral DMA pointer
str r0, [r6,#DMA_SxM0AR_RX] // Store address pointer
str r1, [r6,#DMA_SxSNDTR_RX] // Store number of units

// Clear flags in Status register
mov r5,#(0b111101<<6) // clear flags for Ch1 in ISR
str r5, [r6,# DMA_LIFCR] // Store

// Enable DMA Channel
ldr r5, [r6, #DMA_SxCR_RX]
orr r5, r5, #1
str r5, [r6, #DMA_SxCR_RX] // Enable channel
```

```
// Wait for the end of reception (TCIF
ldr r6, =DMA1_BASE // Load reg base
address to r6

WAIT_RC:
ldr r5, [r6, #DMA_LISR]
tst r5, #(1 << 11) // TCIF1 flag
beq WAIT_RC

pop {r5, r6, pc}
```

STM32H7

Vhodno / izhodne naprave

USART Serijska komunikacija
z uporabo DMA krmilnika

Podrobnejši izseki iz dokumentacije
(Ref.Man. - rm0433)

DMA – sprejem (RM)

Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

DMA – sprejem (RM)

48.5.19 Continuous communication using USART and DMA

15.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2
				w	w	w	w		w	w	w	w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

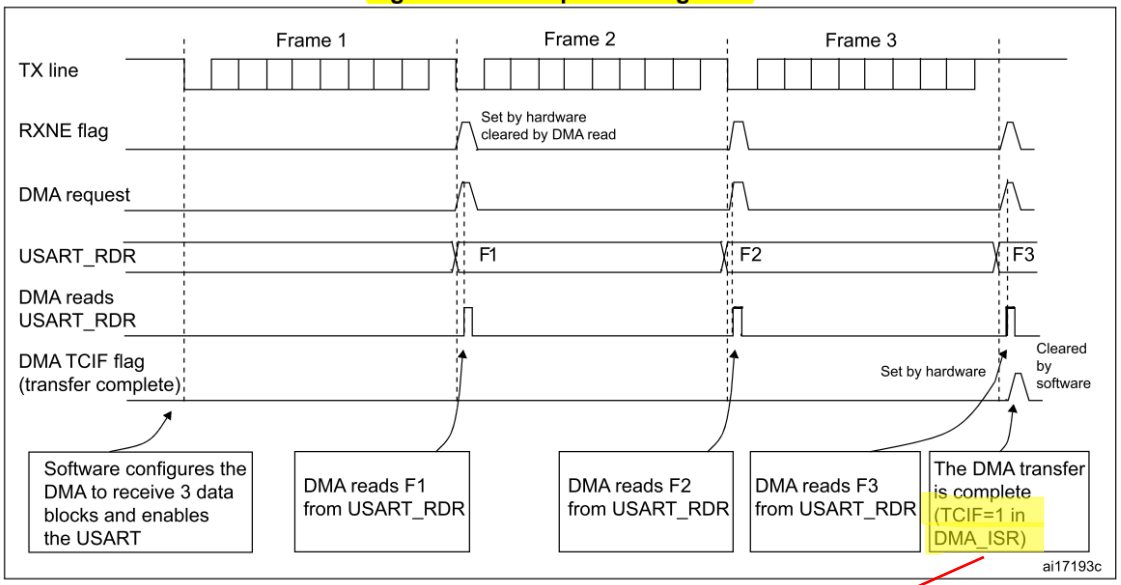
Bits 27, 21, 11, 5 **CTCIF[3:0]**: stream x clear transfer complete interrupt flag (x = 3 to 0)

Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register.

DMA – sprejem (RM)

48.5.19 Continuous communication using USART and DMA

Figure 589. Reception using DMA



15.5.1 DMA low interrupt status register (DMA_LISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2
				r	r	r	r		r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[3:0]**: stream x transfer complete interrupt flag (x = 3 to 0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: no transfer complete event on stream x

1: a transfer complete event occurred on stream x

DMA – oddaja (RM)

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

DMA – oddaja (RM)

48.7.11 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

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.	WUCF	Res.	Res.	CMCF	Res.
											w			w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

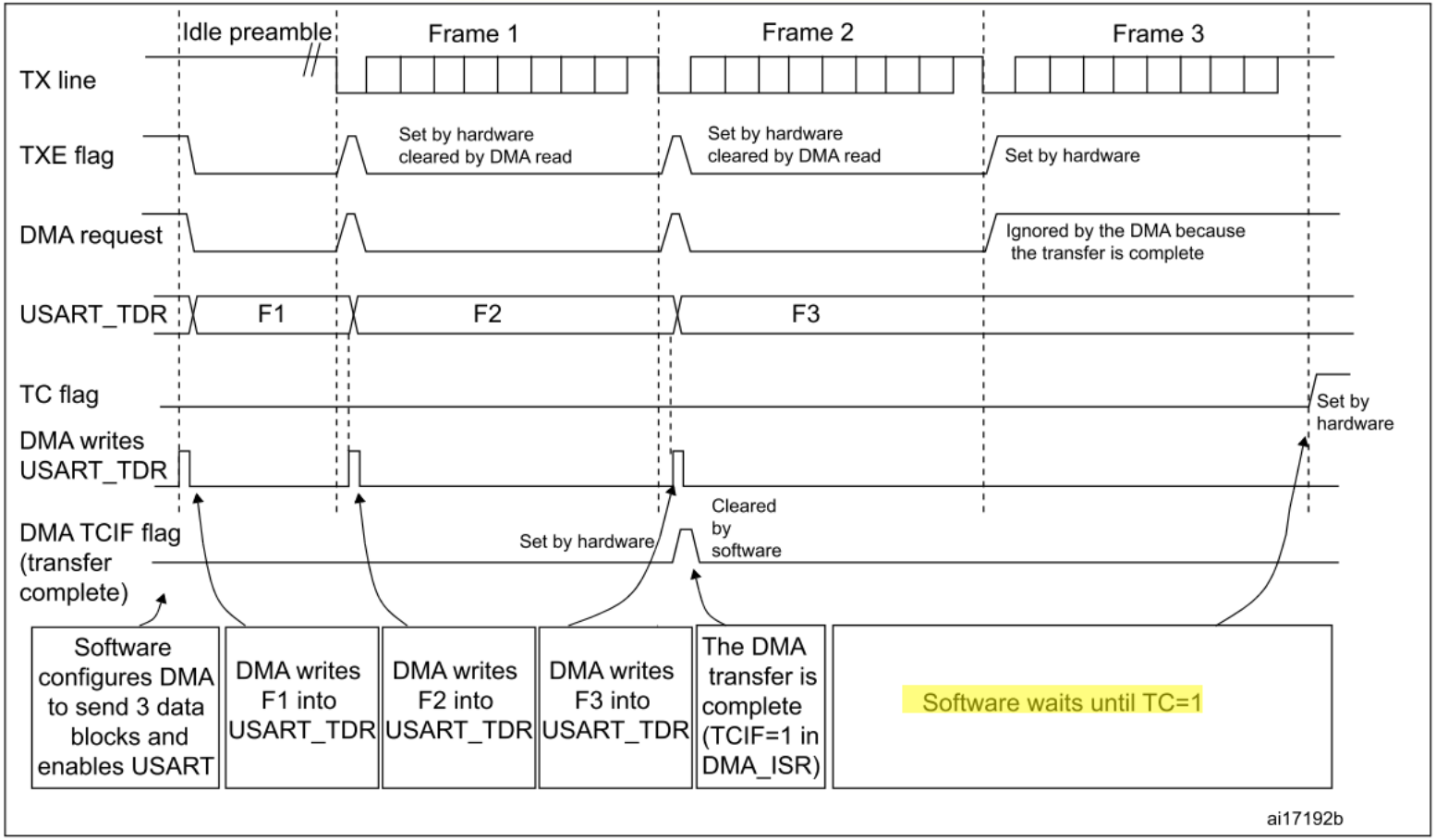
Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USART_ISR register.

DMA – oddaja (RM)

48.5.19 Continuous communication using USART and DMA

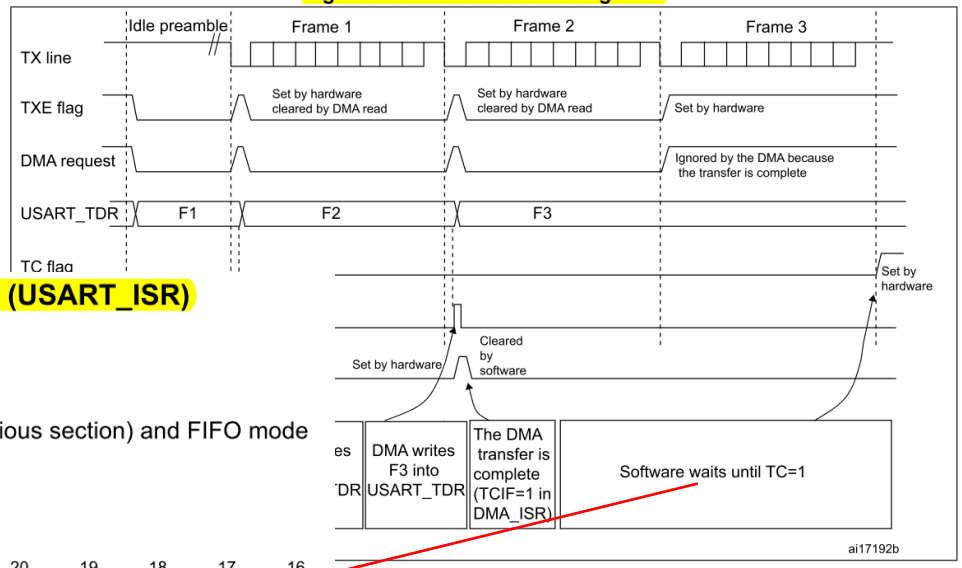
Figure 588. Transmission using DMA



DMA – oddaja (RM)

48.5.19 Continuous communication using USART and DMA

Figure 588. Transmission using DMA



48.7.10 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.