

PŘÍMÝ PŘÍSTUP DO PAMĚTI DMA - OBECNĚ

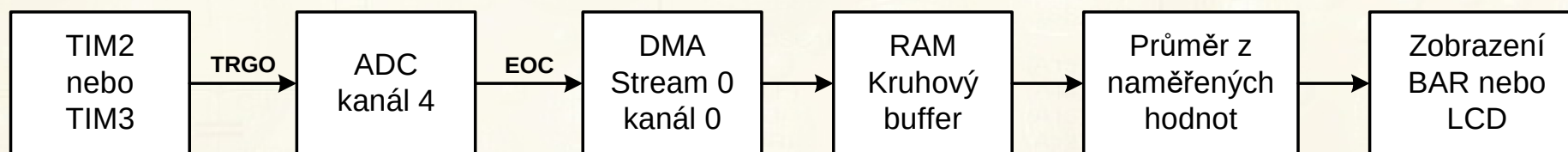
- Přenos dat bez účasti procesoru
- Data se zároveň čtou z V/V zařízení a zapisují do paměti
- Funkce procesoru je nutné nahradit $\Rightarrow$ řadič DMA
- Metody DMA
 - o Zastavením procesoru z důvodu uvolnění sběrnic procesoru nebo vytvoření speciálních sběrnic pro přenosy DMA
 - o Zastavování cyklů procesoru nebo nezávislý přenos DMA
 - o Sdílení paměti - přenos v části cyklu, kdy procesor nekomunikuje po sběrnici

Výhody a nevýhody

- Velká rychlost přenosu hodnot – odstranění neproduktivní činnosti procesoru
- Data nelze nijak upravovat
- Zpomalení nebo zastavení činnosti při některých metodách DMA
- Nutnost dalšího technického vybavení

VYUŽITÍ DMA KANÁLU K UKLÁDÁNÍ A/D HODNOT DO PAMĚTI

Předposlední domácí úloha ARM6 bude pojednávat o spolupráci časovače, generujícího spouštěcí impulzy pro A/D převodník. Naměřené hodnoty převodníku bude přenášet kanál DMA (Direct Memory Access) do paměti RAM. Nastrádané hodnoty budou zprůměrovány a zobrazeny na zobrazovači BAR tvořeného diodami na přípravku MAM_shield. Na obrázku je blokový diagram realizace uvedeného zadání úlohy.



Zpracování naměřených hodnot může být realizováno

- ❖ Po naplnění bufferu - další hodnoty se neukládají, výpočet, zobrazení a opětovné spuštění DMA přenosu.
- ❖ Kontinuální přenos DMA - po naplnění bufferu - stihnout výpočet a zobrazení mezi posledním a novým prvním vzorkem.
- ❖ Kontinuální přenos DMA - po naplnění poloviny bufferu – dílčí výpočet, po naplnění celého bufferu dokončit výpočet a zobrazení.

DMA – umožňuje rychlý přenos hodnot mezi periferií a pamětí nebo pamětí a pamětí bez účasti CPU.

Architektura sběrnice AHB je zdvojená, jedna pro zajištění přístupu do paměti a druhá pro přístup do periferií. V procesoru jsou dva DMA kontroléry, každý disponuje 8 streamy s až 8 požadavky (kanály) přenosu.

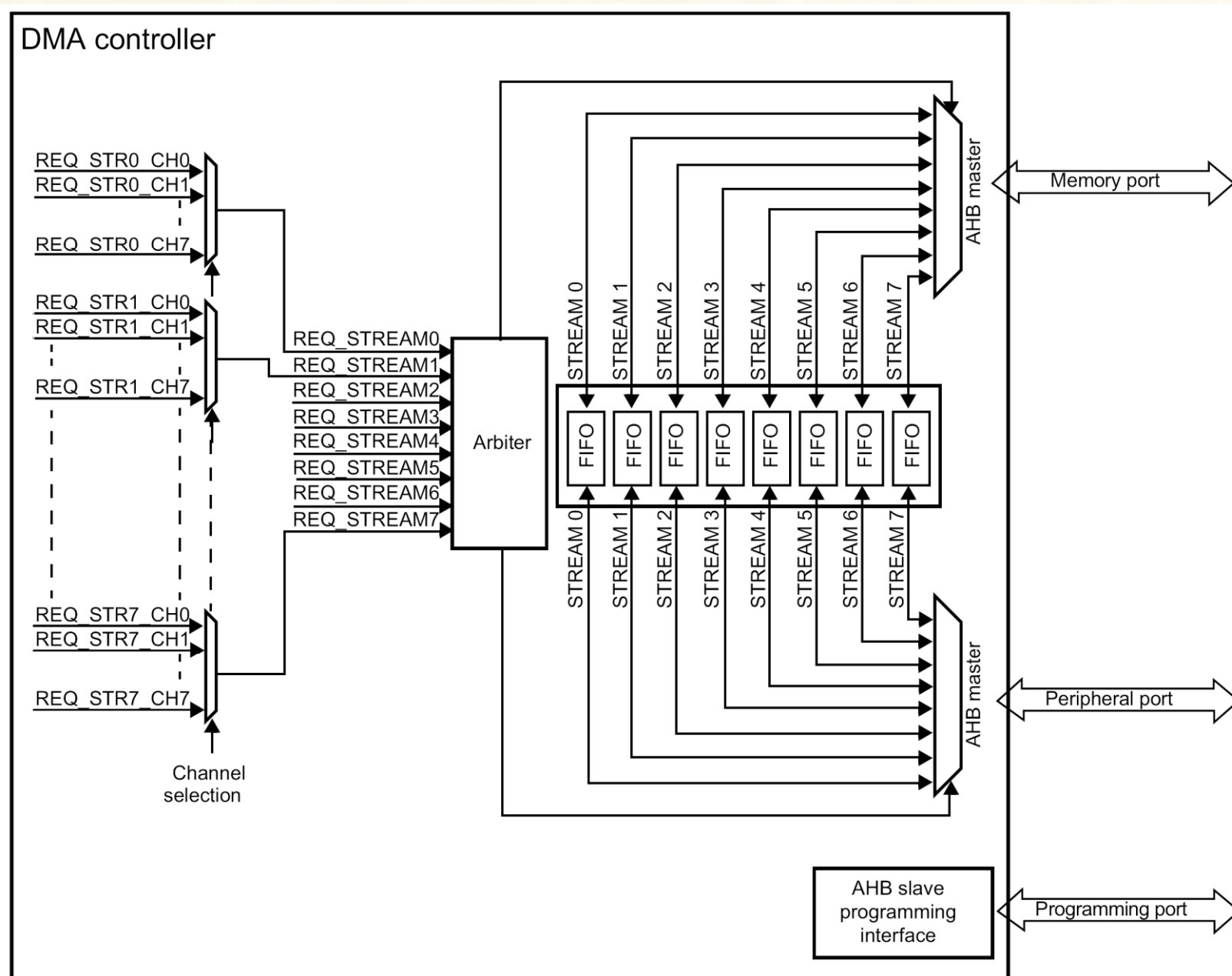
Každý stream disponuje čtyř úrovněnou FIFO pamětí o šířce 32 bitů. Řadič DMA zajišťuje skládáním a rozdělováním slov při nestejně šířce slova zdroje a cíle s programovatelnou prahovou úrovní $\frac{1}{4}$, $\frac{1}{2}$ a $\frac{3}{4}$. Přenos může být realizován přímo nebo přes FIFO.

Priority mezi jednotlivými požadavky (kanály) streamu DMA jsou programovatelné do čtyřech úrovní (velmi vysoké, vysoké, střední, nízké). V případě rovnosti (kanál 0 má přednost před kanálem 1 atd.).

Každý stream podporuje práci s kruhovým buffrem.

DMA generuje 5 příznaků - DMA Half Transfer, DMA Transfer complete, DMA Transfer Error, DMA FIFO Error, Direct Mode Error)

VLASTNOSTI DMA NA PROCESORU M4



INICIALIZACE ČASOVAČE PRO SPOUŠTĚNÍ A/D PŘEVODNÍKU

```
// TIM2 je připojen ke sběrnici APB1, nastavenou na kmitočet 24MHz

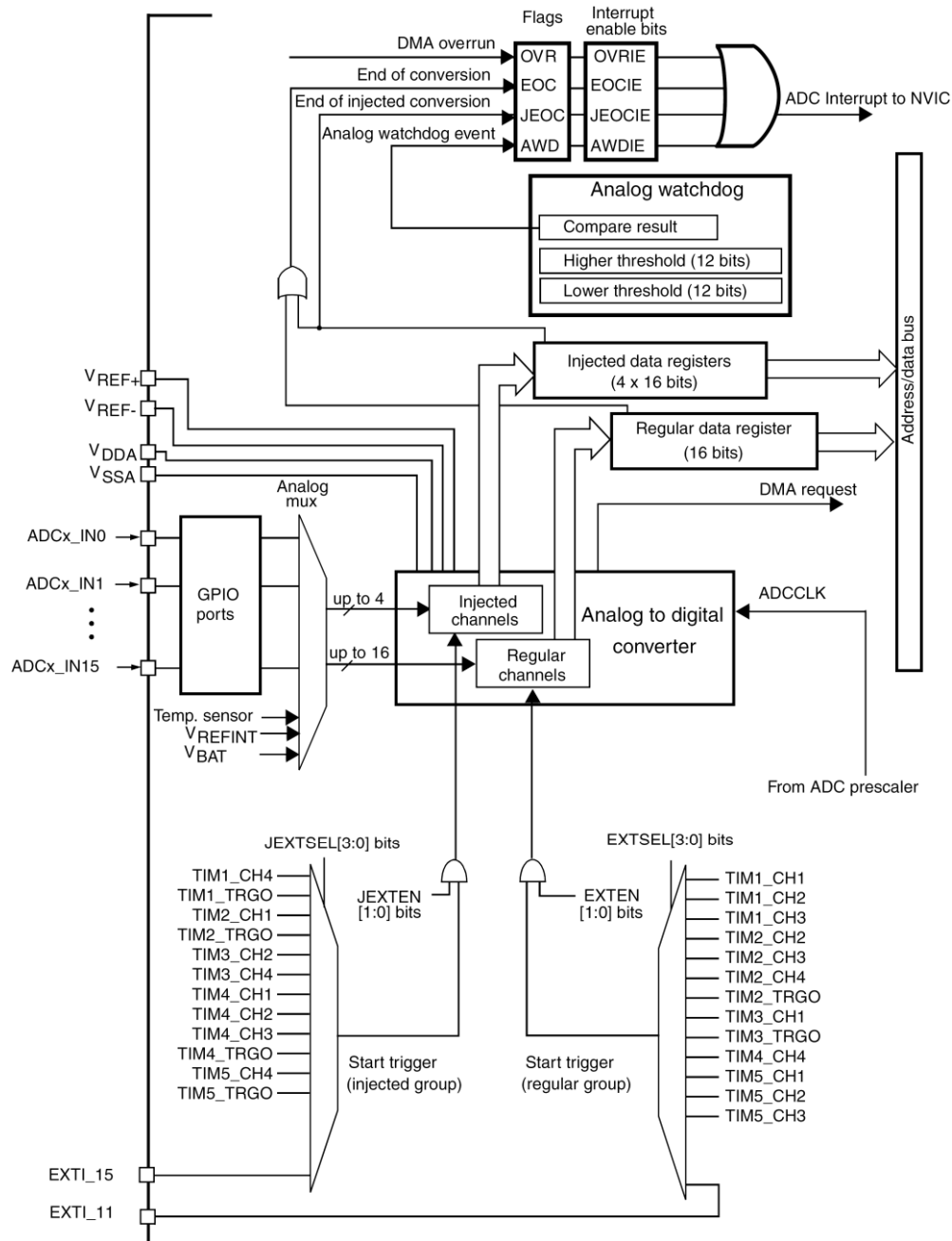
void TIM2_Inicializace(void)
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;    // Povolení hodinového signálu pro TIM2
    TIM2->PSC = 24;                          // Nastavení předděliče čítače
    TIM2->ARR = 1000;                        // Nastavení maximální hodnoty čítače
    TIM2->CR2 = TIM_CR2_MMS_1;              // Bity MMS[2:0]=010, Update generuje signál TRGO
    TIM2->DIER |= TIM_DIER_UIE;             // Povolení přerušení od TIM2
    NVIC_EnableIRQ(TIM2_IRQn);              // Povolení přerušení od TIM2 v kontroléru NVIC
    TIM2->CR1 |= TIM_CR1_CEN;                // Spuštění časovače TIM2
    TIM2->SR |= TIM_SR_UIF;                 // Nulování příznaku události časovače TIM2
}

void TIM2_IRQHandler()
{
    TIM2->SR &= ~TIM_SR_UIF;                // Nulování příznaku události od časovače TIM2

// Možná realizace zobrazení na dynamickém displeji

    togglebit(GPIOC->ODR, 10);              // Identifikace funkce přerušení časovače
                                           // Měření periody generované TIM2
}
```

A/D PŘEVODNÍK NASTAVENÍ



❖ Hodinový signál pro A/D převodník je vyvedený přímo z oscilátoru HSI o kmitočtu 16MHz. Povolený hodinový signál pro převodník, může být vydělen hodnotou 1, 2 nebo 4 v registru **ADC_CCR** bity **ADCPRE**.

❖ **Převodník** může být spuštěn programově nebo obvodově (viz. Obrázek) z vpravo dole uvedených periférií.

INICIALIZACE PŘEVODNÍKU PRO JOYSTICK HORIZONTÁLNÍ SMĚR - PA4

```
// A/D převodník bude spouštěn signálem TRGO

void Inicializace_ADC(void){
    RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;    // Povolení hodin pro ADC1
    // ADC->CCR |= ADC_CCR_TSVREFE;          // Povolení teplotního senzoru a Uref
    ADC->CCR |= ADC_CCR_ADCPRE_1;            // Předdělič ADC = HSI/4
    ADC1->CR2 |= ADC_CR2_EXTEN_0;            // Spouštění převodu na náběžnou hranu
    ADC1->CR2 |= ADC_CR2_EXTSEL_2|ADC_CR2_EXTSEL_1; // ADC spouštěn událostí TIM2 TRGO
    ADC1->SQR3 &= ~ADC_SQR1_L;               // Jeden ADC převod - nuluj L
    // ADC1->SQR1 |= ADC_SQR1_L_0;           // Dva převody
    ADC1->SQR3 &= ~ADC_SQR3_SQ1;             // Smazání všech kanálů pro první převod
    ADC1->SQR3 |= ADC_SQR3_SQ1_2;           // První převod z kanálu 4 (tj. PA_4)
    ADC1->SMPR2 |= ADC_SMPR2_SMP4;          // Vzorkování kanálu 4 - 480 cyklů
    ADC1->CR1 &= ~ADC_CR1_RES;              // 12 bitová konfigurace (Tcon=15 ADCCLK cyklů)
    ADC1->CR2 &= ~ADC_CR2_ALIGN;            // Zarovnání doprava
    // ADC1->CR1 |= ADC_CR1_SCAN;            // Nastavení skenovacího režimu
    ADC1->CR2 |= ADC_CR2_DMA;               // Povolení DMA módu
    // ADC1->CR2 |= ADC_CR2_DDS;             // Přenos DMA po každém ADC převodu
    //                                         // Nastavením můžeme zajistit kontinuální
    //                                         // přenos naměřených hodnot od počátečního
    //                                         // spuštění
    ADC1->CR2 |= ADC_CR2_EOCS;              // Nastavení EOC po každém převodu
    ADC1->CR2 |= ADC_CR2_ADON;              // Spuštění ADC převodníku
}
```

INICIALIZACE PŘEVODNÍKU PRO JOYSTICK HORIZONTÁLNÍ SMĚR - PA4

```
// DMA Stream 0 kanál 0 bude realizovat přenos periferie(A/D)-paměť RAM (ADC_Buffer)

void Inicializace_DMA(void){
    RCC->AHB1ENR |= RCC_AHB1ENR_DMA2EN;           // Povolení hodin pro DMA2
    // ADC1->CR2|= ADC_CR2_DMA;                     // DMA mód povolen v inicializaci ADC1
    DMA2_Stream0->CR &= ~DMA_SxCR_EN;               // Zakázání Streamu
    while((DMA2_Stream0->CR &0x0001)!=0) ;           // Čekej na ukončení DMA2_Stream0

    DMA2_Stream0->PAR = (uint32_t)&ADC1->DR;          // Nastavení adresy DR registru ADC1
    DMA2_Stream0->M0AR = (uint32_t)&ADC_Buffer;       // Nastavení adresy ADC_Buffer

    DMA2_Stream0->CR |= DMA_SxCR_CIRC;               // Povolení kruhového módu (kruhový bufer)
    DMA2_Stream0->CR |= DMA_SxCR_MINC;               // Adresa do buffru je inkrementována
    DMA2_Stream0->CR |= DMA_SxCR_PSIZE_0;            // Z ADR1 se přenáší se slova o 16 bitech
                                                    // (half word)
    DMA2_Stream0->CR |= DMA_SxCR_MSIZE_0;            // Do paměti se přenáší slova o 16 bitech
    DMA2_Stream0->CR &= ~DMA_SxCR_CHSEL;             // Přenos kanálem 0
    DMA2_Stream0->NDTR |= 16;                        // Přenáší se 16 hodnot
    DMA2_Stream0->CR |= DMA_SxCR_EN;                 // Zapnutí DMA2 Stream0 kanál 0
}

void DMA2_Stream0_IRQHandler() {
    if ((DMA2->LISR & DMA_LISR_TCIF0 )!=0)
    {
        DMA2->LIFCR |= DMA_LIFCR_CTCIF0;            // Nuluj návěští naplnění bufferu
        DMA2->LIFCR |= DMA_LIFCR_CHTIF0;            // Nuluj flag naplnění poloviny bufferu

        // Nutno znovu obnovit přenos přes DMA, pokud nebylo použito ADC1->CR2|= ADC_CR2_DDS;
        ADC1->CR2&= ~ADC_CR2_DMA;                   // DMA mód zakázán
        ADC1->CR2|= ADC_CR2_DMA;                     // DMA mód povolen - Nové měření
                                                    // Nemusí být umístěno v přerušení
    }
}
```