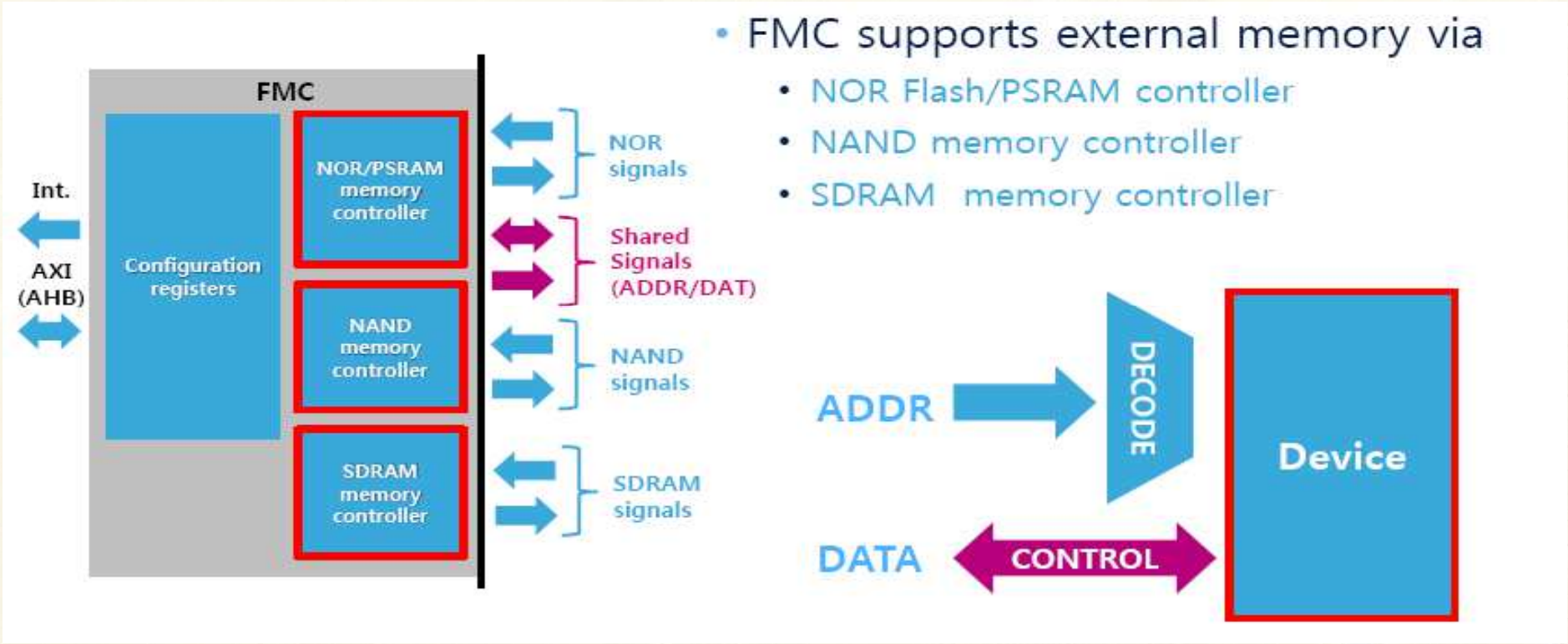
ARM_FSMC – FLEXIBILNÍ KONTROLÉR STATICKÉ PAMĚTI

ARM_FSMC zajišťuje přístup procesoru k externím pamětem. Operace na sběrnici AHB jsou FSMC přizpůsobeny komunikaci s externím zařízením (pamětí). Přenosy mohou být 8,16 nebo 32 bitové. Asynchronní přístup k pamětem po bytech (SRAM, ROM, PSRAM) zajišťuje FSMC pomocí vývodů NBL [1:0]. Čtou se všechny byty a nepoužitelné jsou vyřazeny. Z paměti (NOR) lze číst, ale zapisovat se mohou pouze 16bitová slova.



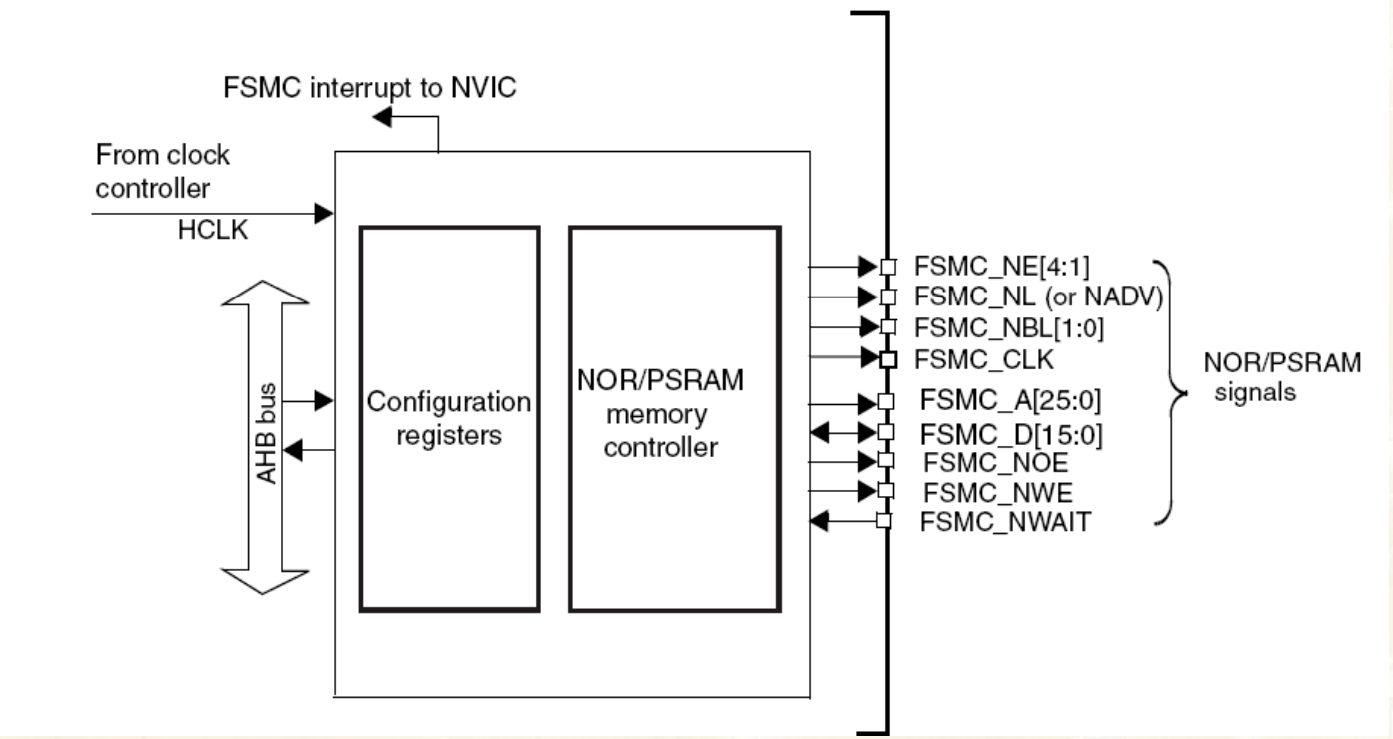
ARM_FSMC – FLEXIBILNÍ KONTROLÉR STATICKÉ PAMĚTI

Externí paměti – prostor 0x60000000÷0x9FFFFFFF

Externí periferie – od adresy 0xA0000000.

Dostupné přes FSMC naprogramovaným protokolem.

Prostor pro paměti rozdělen na 4 banky po 256MB rozlišené bity A27:A26 sběrnice AHB. Banka 1 je dále rozdělena do 4 částí po 64MB aktivovaných CS1 až CS4. Banka je adresována 26 bity A25:A0 (8bitů) nebo A25:A1 (16bitů).



ARM_FSMC – FLEXIBILNÍ KONTROLÉR STATICKÉ PAMĚTI

Kontrolér FSMC generuje následující signály pro obsluhu vnějších pamětí:

❖ CLK	Výstup	Hodiny (pro synchronní přístup)
❖ A[25:0]	Výstup	Adresová sběrnice
❖ D[15:0]	I/O	Obousměrná sběrnice pro data
❖ NE[x]	Výstup	Chip select, x = 1..4
❖ NOE	Výstup	Output enable
❖ NWE	Výstup	Write enable
❖ NL(=NADV)	Výstup	Latch enable (signál označuje platnou adresu NADV, pro některé NOR)
❖ NWAIT	Vstup	NOR Flash – vstup pro zpoždění signálů FSMC

Při multiplexovaném přístupu do pamětí NOR je adresa a data rozdělena:

❖ A[25:16]	Výstup	Adresová sběrnice
❖ AD[15:0]	I/O	16-bitová multiplexovaná obousměrná adresová/datová sběrnice

ARM_FSMC – FLEXIBILNÍ KONTROLÉR STATICKÉ PAMĚTI

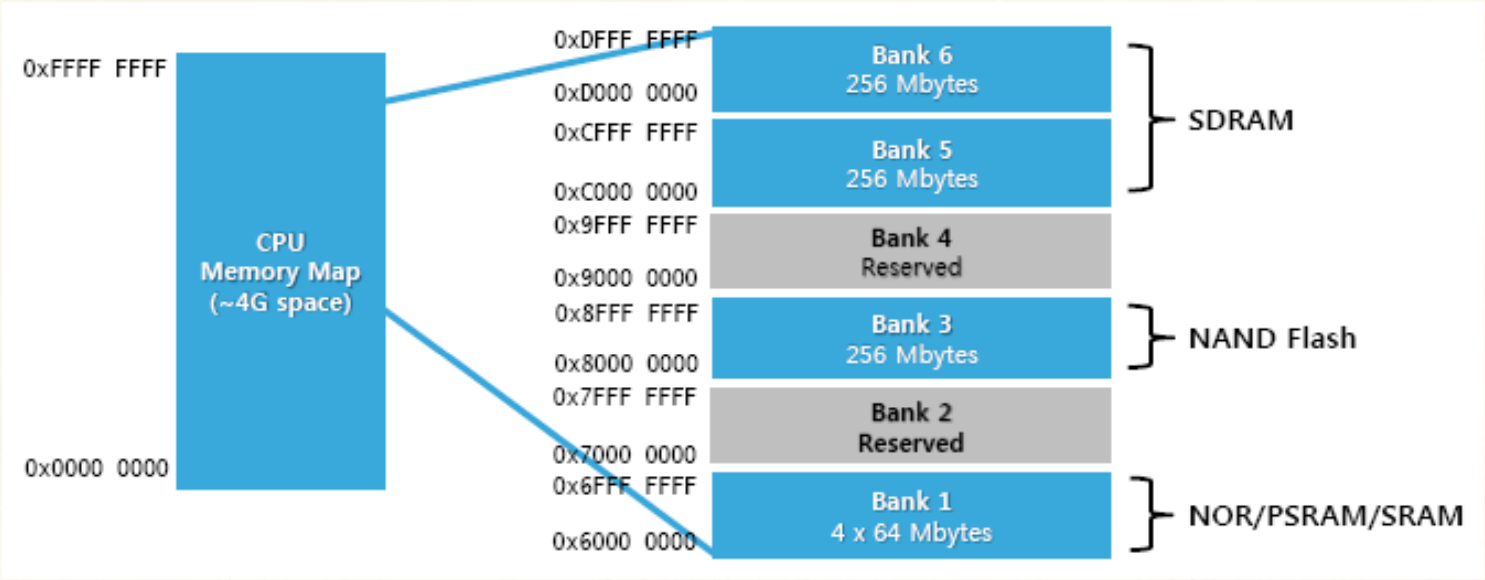
Při nemultiplexovaném přístupu do pamětí PSRAM/SRAM:

- NE[x] Výstup Chip select, x = 1..4 (označený NCE pro PSRAM (dynamické s interním refrešováním)
- NBL[1] Výstup Aktivace horního bytu (na paměti označovaný - NUB)
- NBL[0] Výstup Aktivace dolního bytu (na paměti označovaný - NLB)

Při multiplexovaném přístupu do pamětí PSRAM/SRAM:

- A[25:16] Výstup Adresová sběrnice
- AD[15:0] I/O 16-bitová multiplexovaná obousměrná adresová/ datová sběrnice
- NE[x] Výstup Chip select, x = 1..4 (označený NCE pro PSRAM (dynamické s interním refrešováním)
- NBL[1] Výstup Aktivace horního bytu (na paměti označovaný - NUB)
- NBL[0] Výstup Aktivace dolního bytu (na paměti označovaný - NLB)

ARM_FSMC – FLEXIBILNÍ KONTROLÉR PAMĚTÍ

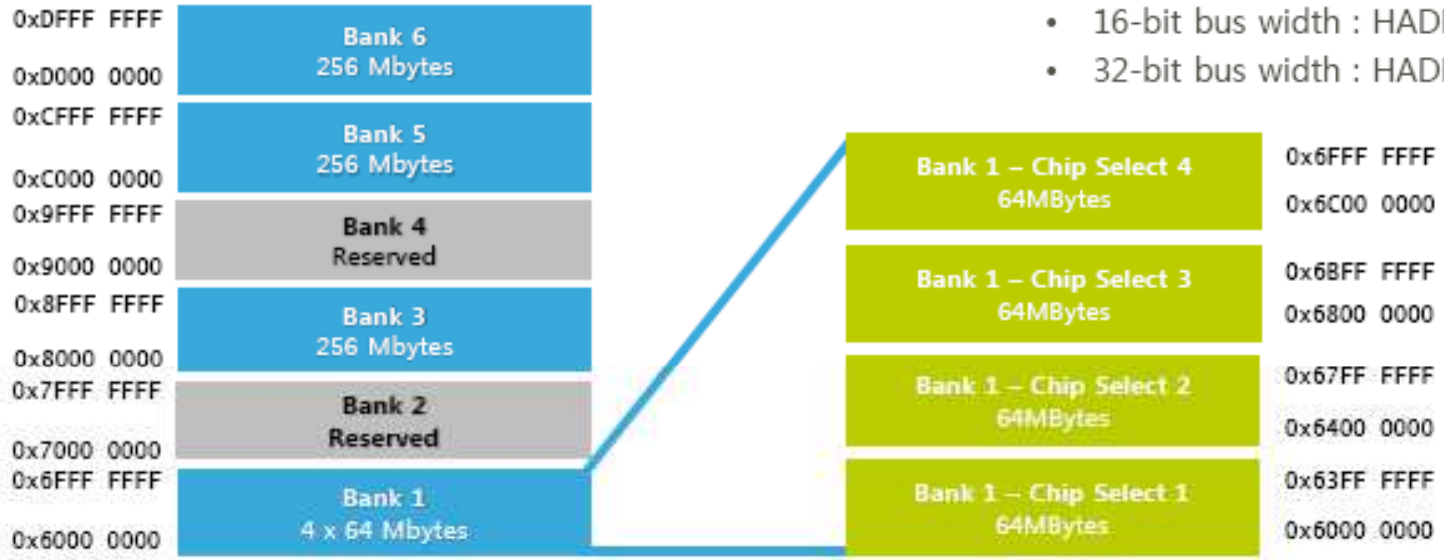


Jednotlivé bloky pamětí lze rekonfigurovat prohozením banky1 s bankou 5 a získat 2 prostory pro připojení SDRAM.

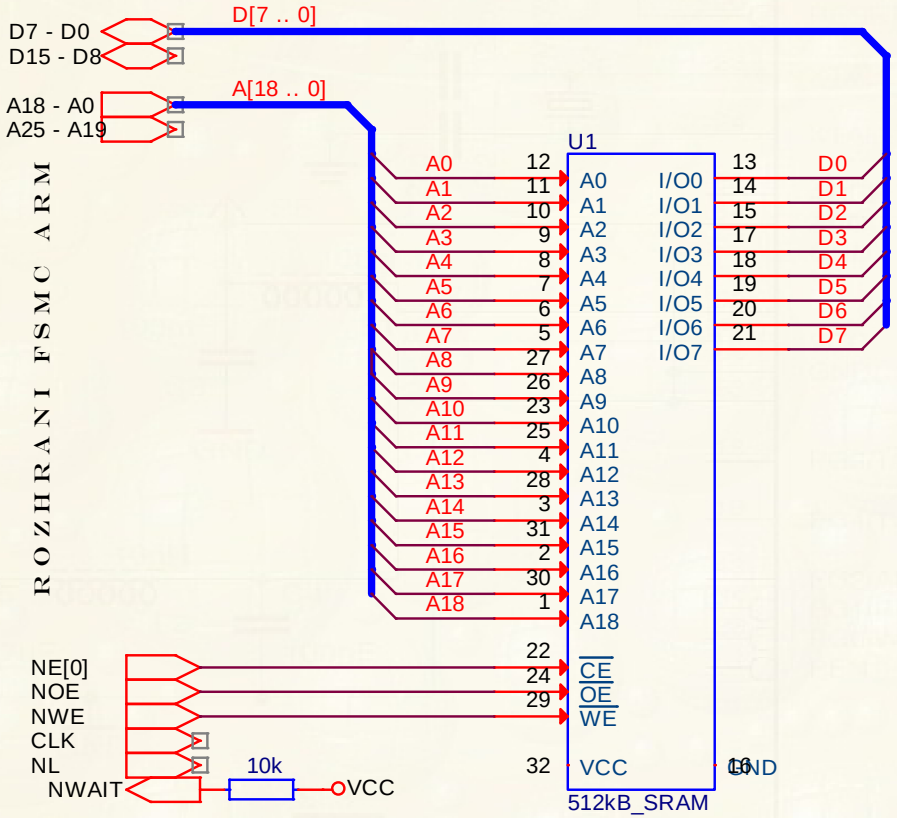
Pro každou banku umožňuje FSMC nezávislou konfiguraci časování připojené paměti s externím řízením čekacích stavů. Zápis může být i do FIFO paměti s rozsahem 16x32bitů.
Statické paměti – SRAM, ROM, NOR, PSRAM
NAND Flash – ECC kontrola až 8kB dat pro čtení i zápis
SDRAM – interface pro synchronní paměti DRAM a SDRAM.

ARM_FSMC – FLEXIBILNÍ KONTROLÉR STATICKÉ PAMĚTI

- Bank 1 is divided into 4 banks of 64 Mbytes each to interface with 4 external NOR / PSRAM memories (4 Chip Selects) which support
  - NOR Flash: 8/16/32-bit synchronous/asynchronous, multiplexed or non-multiplexed
  - SRAM: 8/16/32-bit
  - PSRAM: 8/16/32-bit synchronous/asynchronous
- **HADDR[27:26] :** Bank select
- **HADDR[25:0] :**
  - 8-bit bus width : HADDR[25:0]
  - 16-bit bus width : HADDR[25:1] >> 1
  - 32-bit bus width : HADDR[25:2] >> 2

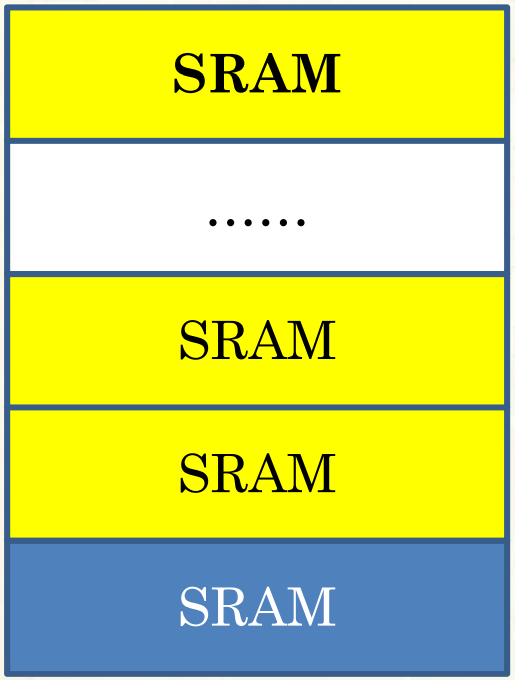


ARM_FSMC – FLEXIBILNÍ KONTROLÉR STATICKÉ PAMĚTI



SRAM o kapacitě 512 kB v prostoru 64MB banky 1 části 0.

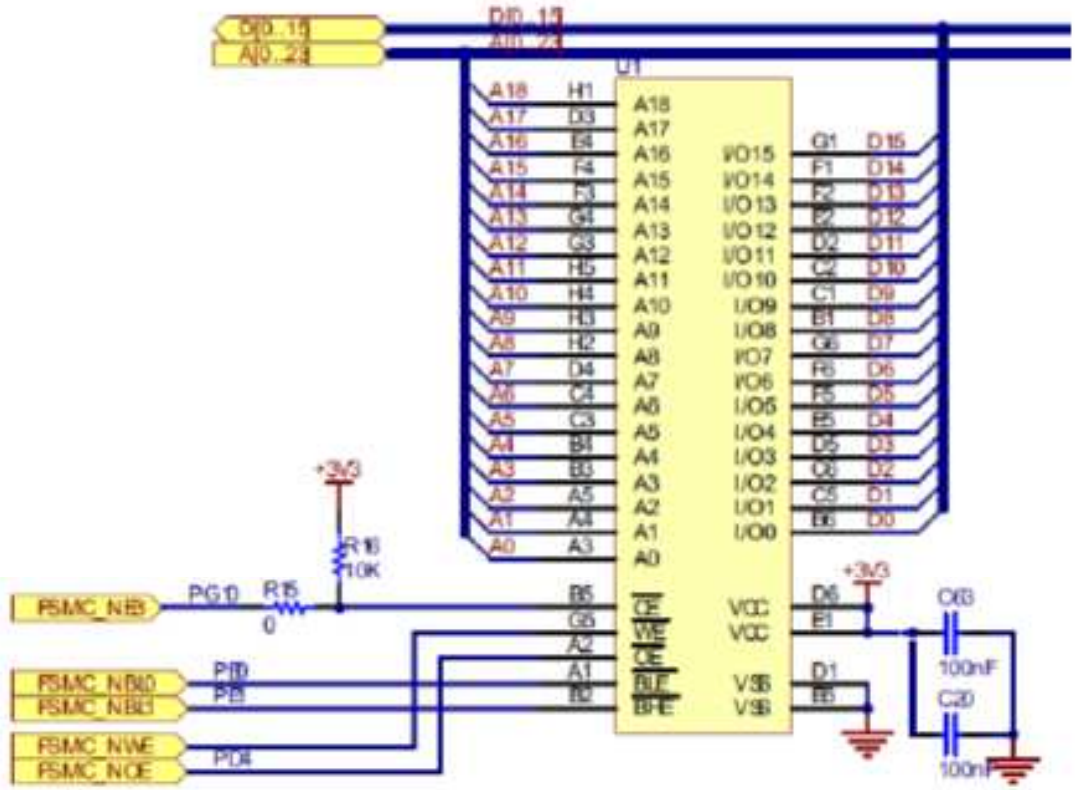
```
0x63FF FFFF

0x63F8 0000

...

0x6017 FFFF

0x6010 0000
0x600F FFFF

0x6008 0000
0x6007 FFFF

0x6000 0000
```

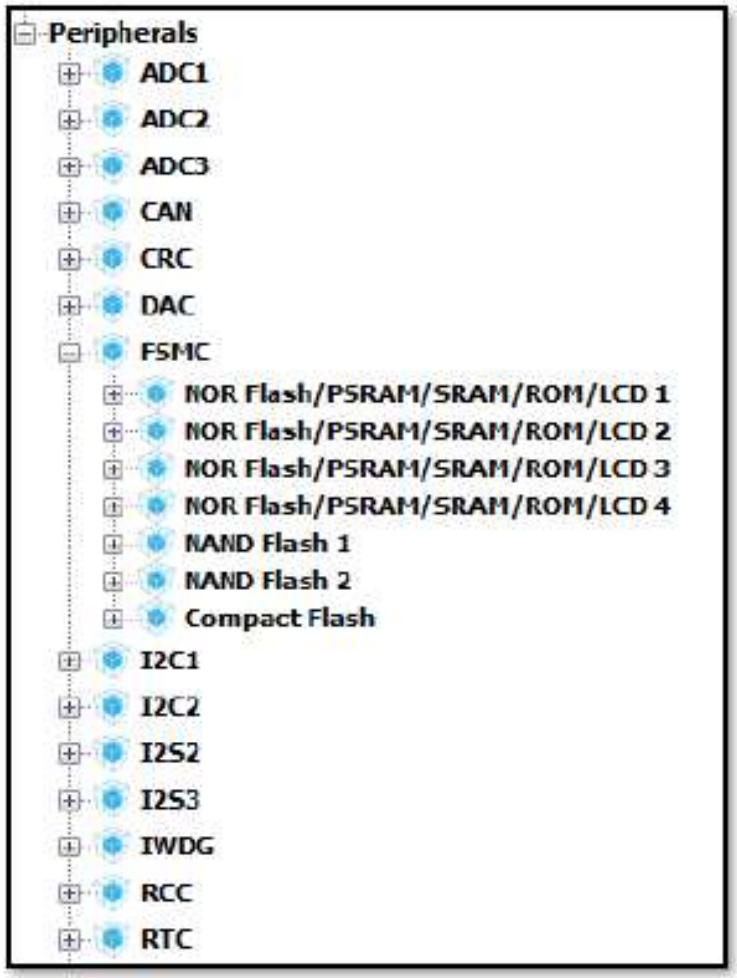


Jak zabránit 128 násobnému opakování SRAM o kapacitě 512 kB v prostoru 64MB?

Za jakých podmínek by bylo možné umístit do prostoru 64MB další paměti nebo periferie?

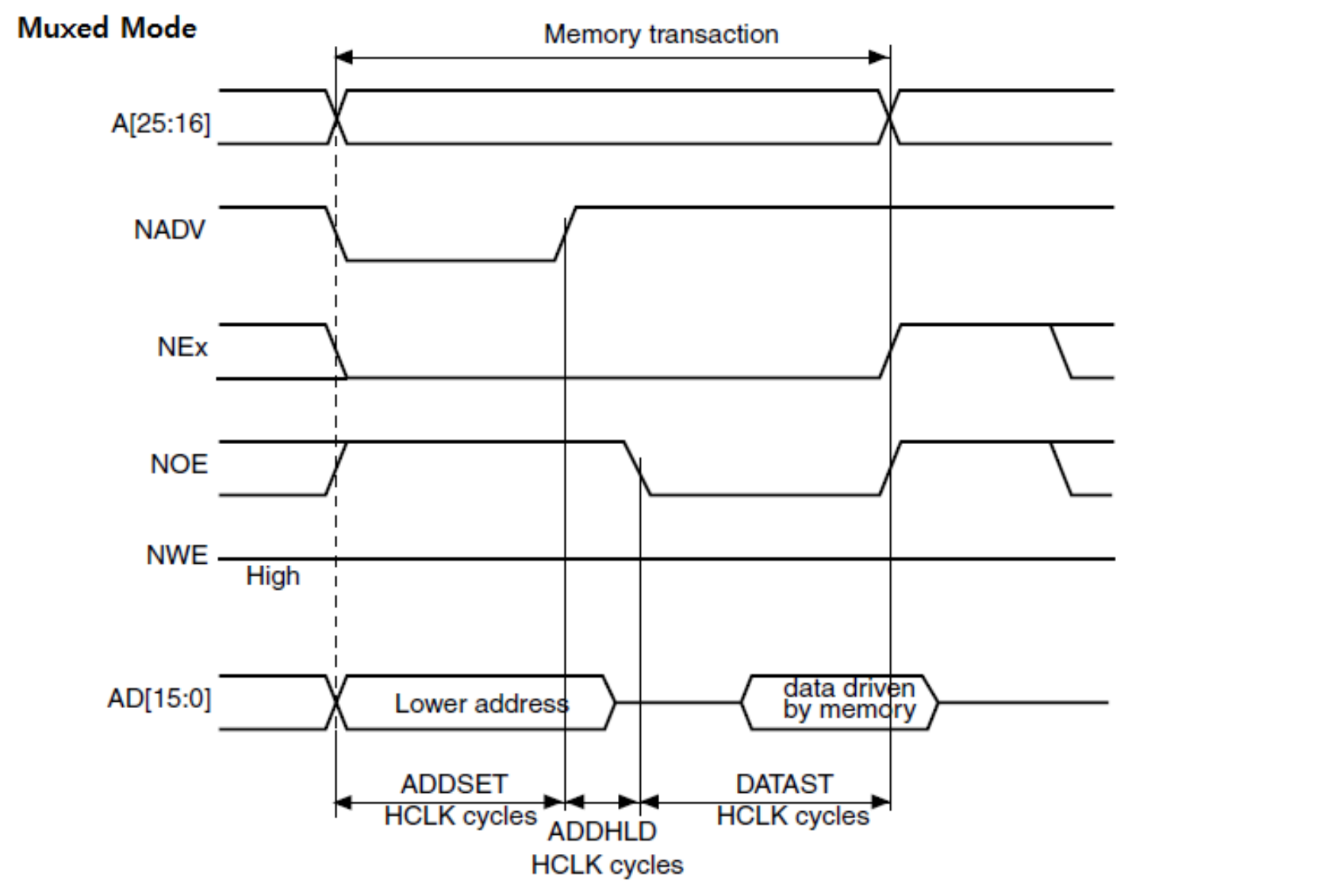
ARM_FSMC – CUBEMX KONFIGURACE PAMĚTI SRAM



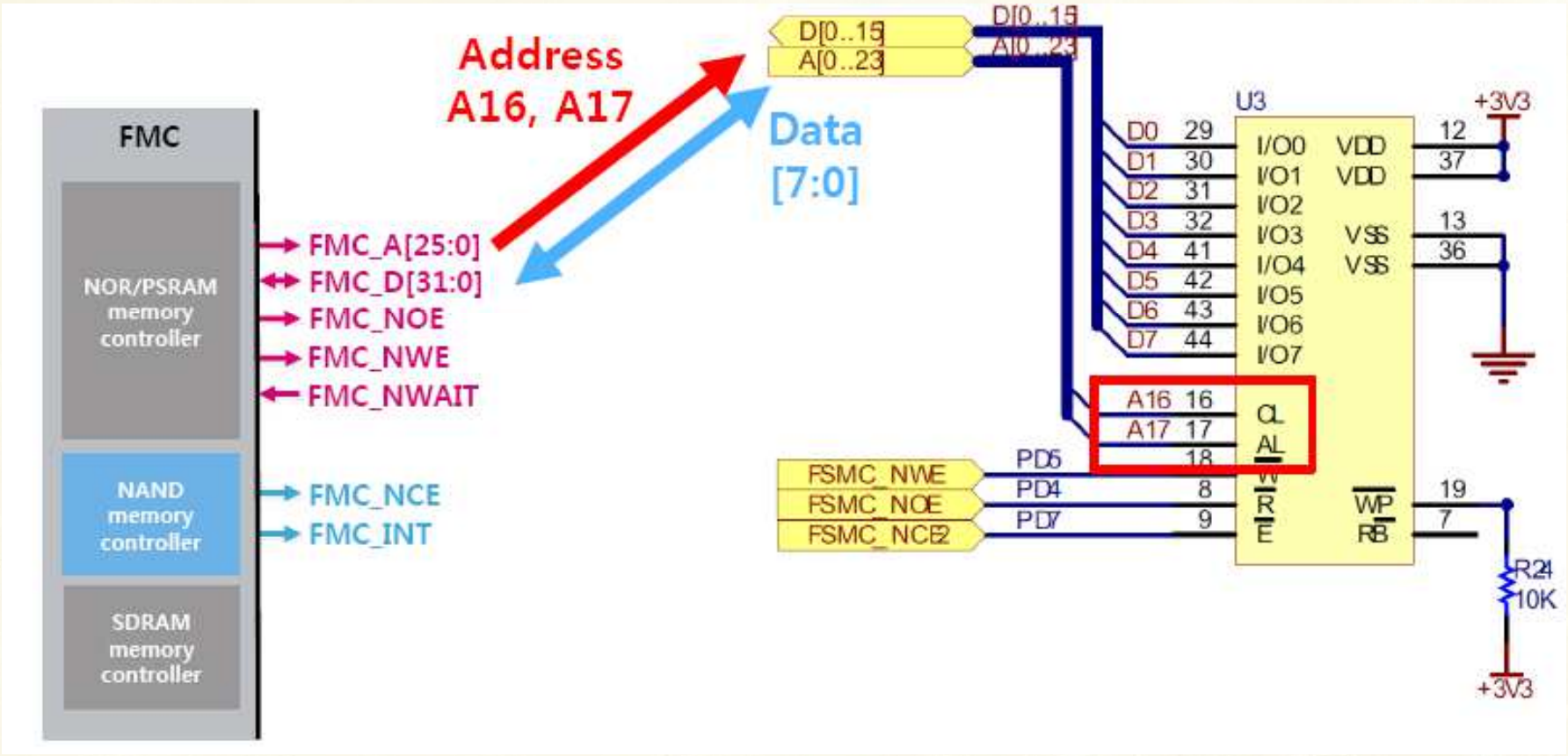


ČASOVÁNÍ PRO ČTENÍ A ZÁPIS

FSMC umožňuje řadu konfiguračních módů (Mód 1, A, 2/B, C, D, MUX) pro přizpůsobení čtení a zápisu do příslušné externí paměti (další v dodatku).



ARM_FSMC – MULTIPLEXOVANÝ PŘÍSTUP DO PAMĚTI NAND



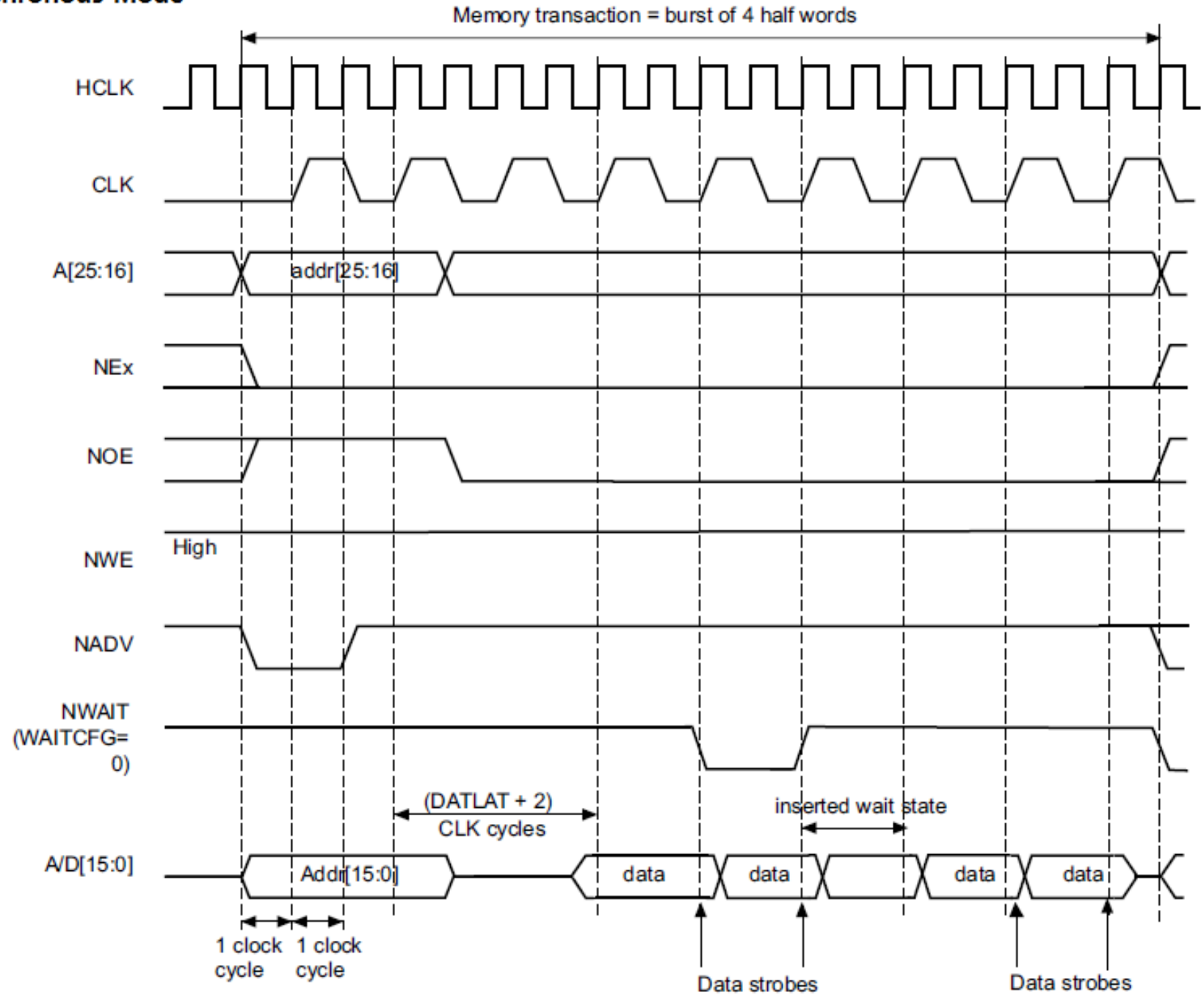
8-bit NAND Flash memory

Table 75. 8-bit NAND Flash

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus

A17:16 [0b1x]	0x3FFFF 0x20000	Address Section
A17:16 [0b01]	0x1FFFF 0x10000	Command Section
A17:16 [0b00]	0x0FFFF 0x00000	Data Section

Synchronous Mode

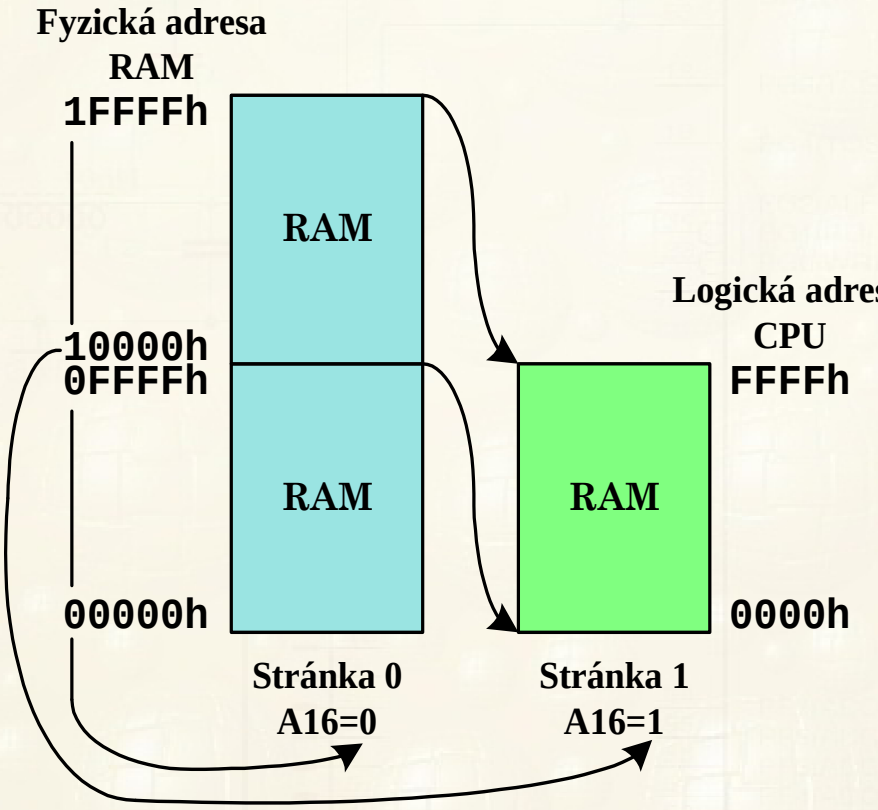


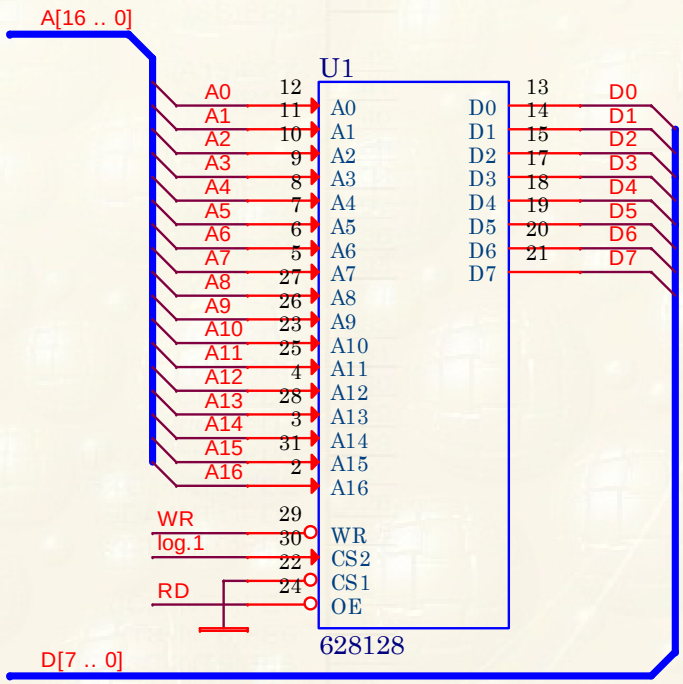
PŘIPOJENÍ PAMĚTI PŘESAHUJÍCÍ PŘÍMO ADRESOVATELNÝ PROSTOR

Jak vytvořit **chybějící** adresové vodiče A16, A17, atd.?

Jak vyřešit adresový dekodér?

Řešení s jednou pamětí 128kB připojené k procesoru s přímým adresovým prostorem 64kB



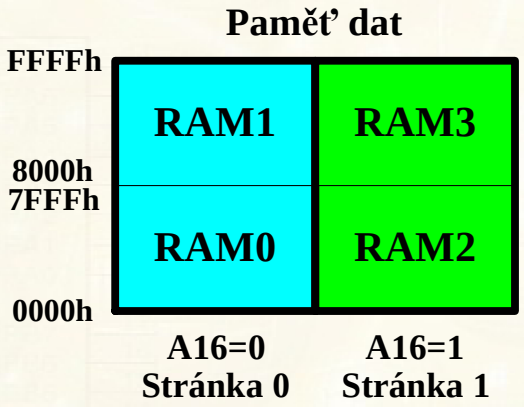


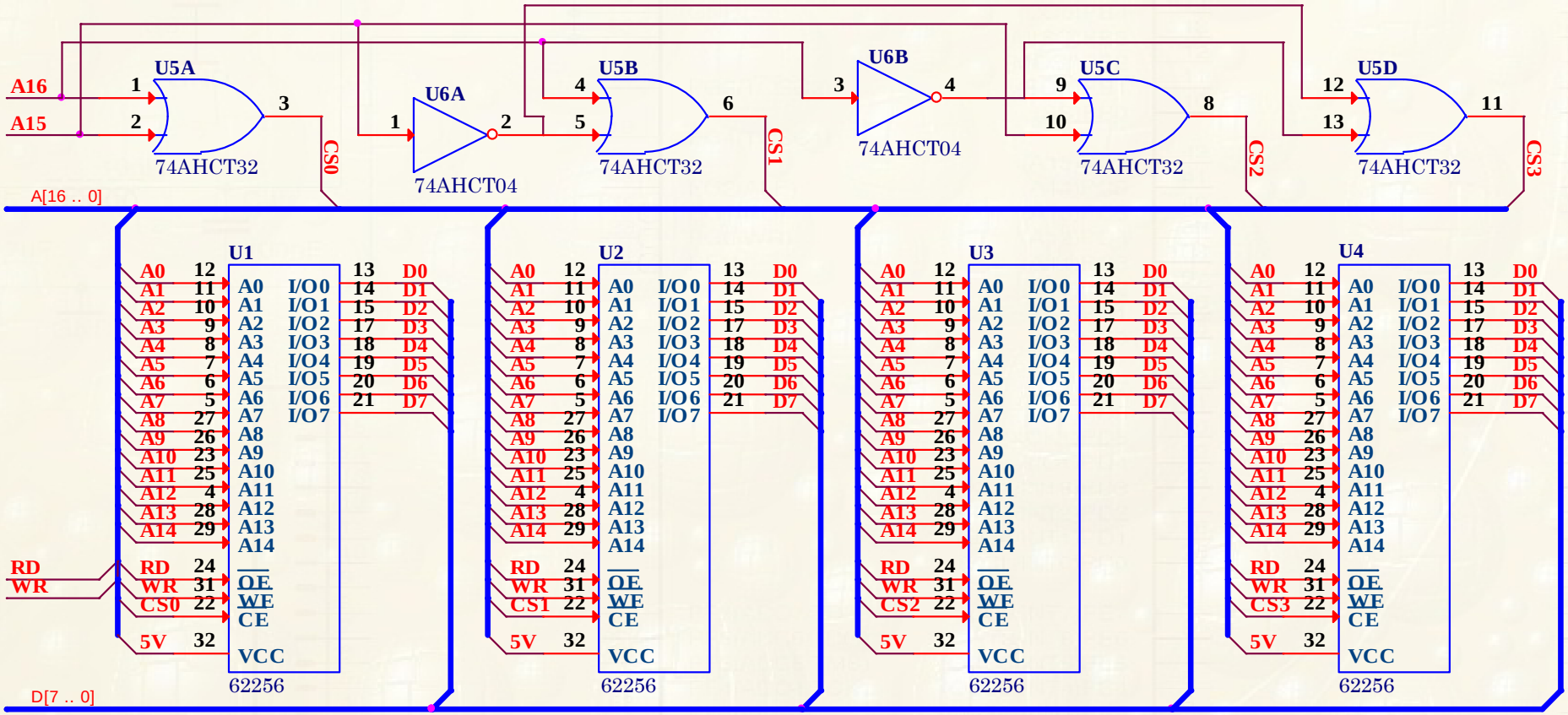
PARALELNÍ PŘIPOJENÍ VELKÉ DATOVÉ PAMĚTI

Pro čtyři 32kB pamětí RAM musíme vytvořit 4 aktivační signály CS0, CS1, CS2 a CS3.

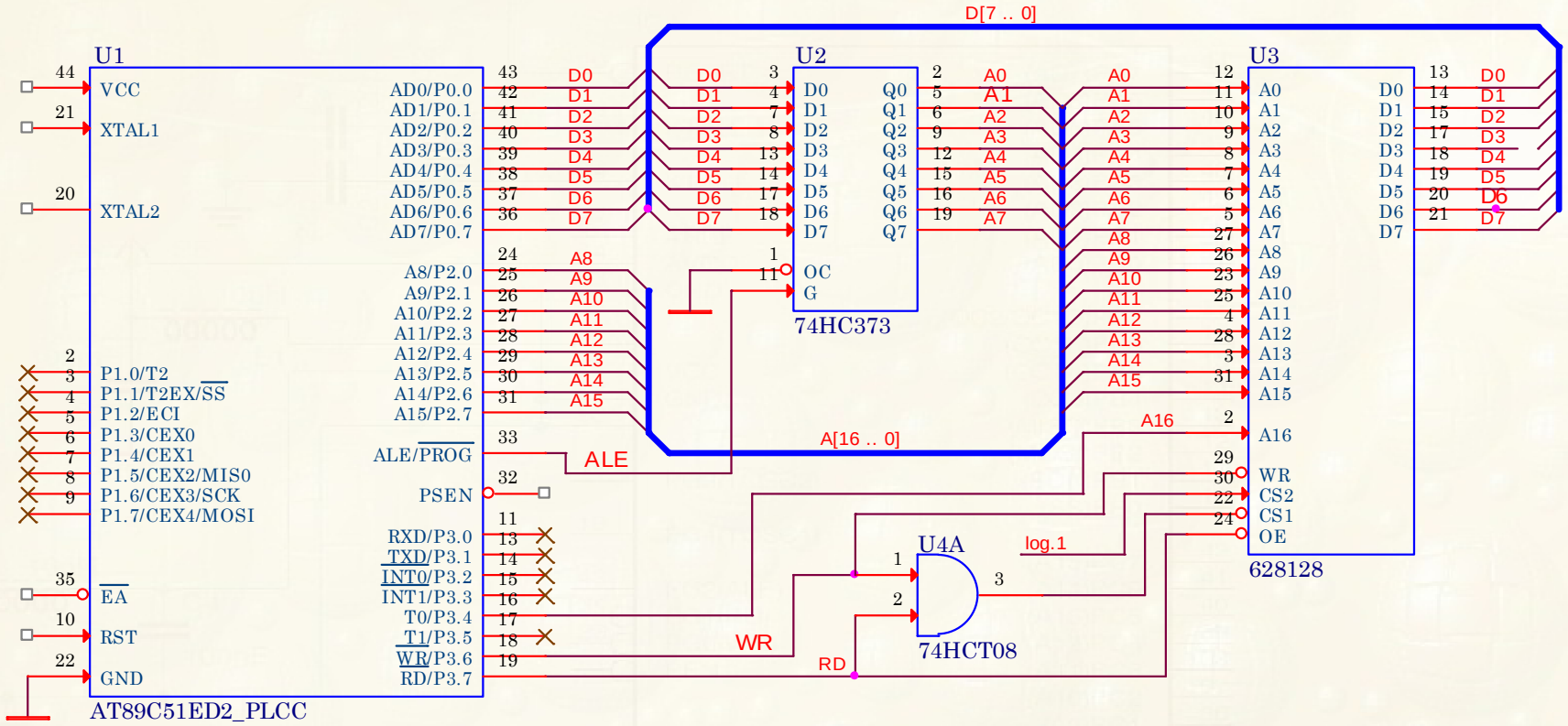
$$CS0 = A15 + A16 \qquad CS1 = \overline{A15} + A16$$

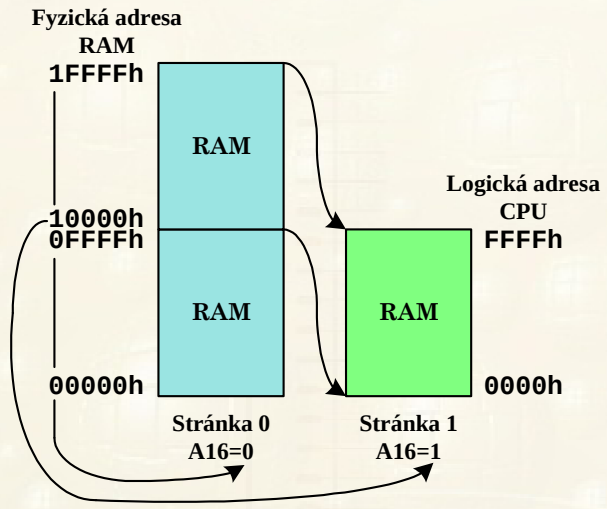
$$CS2 = A15 + \overline{A16} \qquad CS3 = \overline{A15} + \overline{A16}$$





UKÁZKA PROGRAMOVÉ OBSLUHY VELKÉ DATOVÉ PAMĚTI





Vodiče paměti A15÷A0 připojeny na sběrnici, vodič A16 například na vývod P3.4 procesoru. Tím je paměť rozdělena do dvou bloků takto:

A16 P3.4	**Adresa procesoru A15÷A0**	**Adresový prostor paměti**
0	0000h÷FFFFh	00000h÷0FFFFh
1	0000h÷FFFFh	10000h÷1FFFFh

UKÁZKA PROGRAMOVÉ OBSLUHY VELKÉ DATOVÉ PAMĚTI

Podprogram pro zápis registru ***R7*** do paměti na adresu uloženou v paměťových místech ***adr_ram, adr_ram+1 a adr_ram+2***

```
PIS_DO_RAM:
        MOV A,adr_ram       ; Střadač = A23÷A16
        RRC A               ; Příznak přenosu C = A16
        MOV P3.4,C          ; A16 nastaveno
        MOV DPH,adr_ram+1   ; Horní část ukazatele =
                            ; A15÷A8
        MOV DPL,adr_ram+2   ; Spodní část ukazatele =
                            ; A7÷A0
        MOV A,R7            ; Zapisovaná hodnota do
                            ; střadače
        MOVX @DPTR,A        ; Zápis do paměti
        RET                 ; Návrat z podprogramu
```

UKÁZKA PROGRAMOVÉ OBSLUHY VELKÉ DATOVÉ PAMĚTI

Podprogram v jazyce C - zápis proměnné ***hodnota*** na adresu RAM určenou proměnnou ***adr_ram*** (knihovní funkce XBYTE slouží pro přímý přístup do vnější datové paměti).

```
void  pis_do_ram (unsigned long adr_ram,
                  unsigned char hodnota)
{     if ((adr_ram&0x10000)!=0)  A16=1; else A16=0;
      XBYTE[adr_ram&0xFFFF]=hodnota;
      return; }
```

Nebo efektivněji (přístup k bytu adresy závisí na modelu Endian)

```
void  pis_do_ram (unsigned long adr_ram,
                  unsigned char hodnota)
{     if ((*((char*)(&adr_ram)+1)&0x01)!=0)  A16=1;
      else A16=0;
      XBYTE[adr_ram&0xFFFF]=hodnota;
      return; }
```

PŘIPOJENÍ VELKÉ PROGRAMOVÉ PAMĚTI

- Změna vodičů A16, A17 a vyšších jednou nebo více instrukcemi
- **Problém změny programové stránky paměti**
- Návaznost programu při změně stránky v asembleru a jazyce C
- **Obvodové řešení** pomocí adresového dekodéru
- Příklad

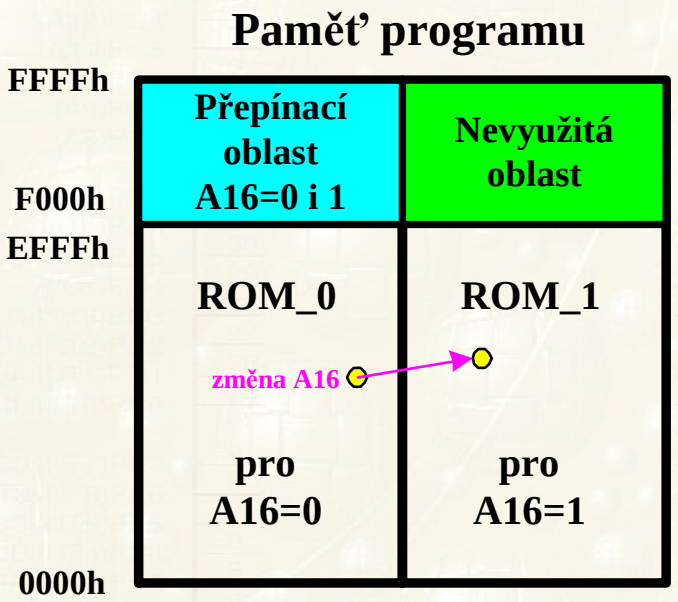
  0000h÷EFFFh paměť ROM0 pro A16=0
  0000h÷EFFFh paměť ROM1 pro A16=1
  F000h÷FFFFh vždy paměť ROM0 nezávisle na A16.

  Pro signály CS0 a CS1 můžeme psát

$$CS0 = A16 * \overline{A15 * A14 * A13 * A12}$$

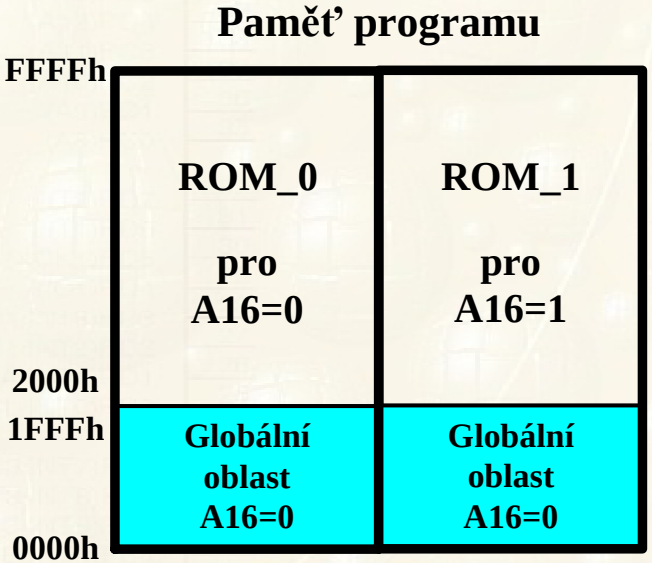
$$CS1 = \overline{A16} + A15 * A14 * A13 * A12$$

V přepínací oblasti musí být umístěny všechny globálně přístupné podprogramy, přerušení, řetězce, atd.



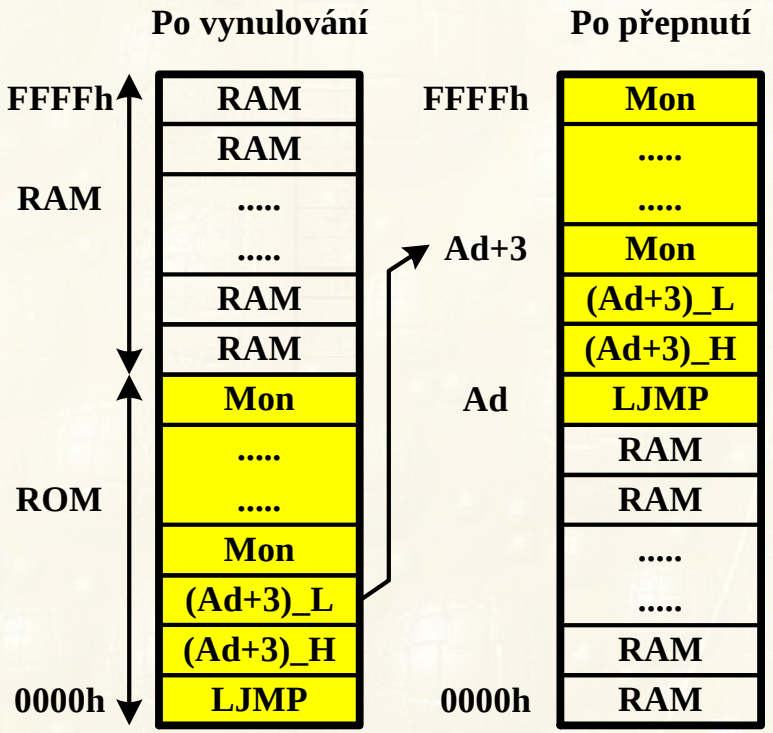
PŘIPOJENÍ VELKÉ PROGRAMOVÉ PAMĚTI U 8051

- **Přepínací oblast** umístěna v dolní části adresového prostoru, na obrázku označená jako **globální oblast**
- **Umístění obslužných programů přerušovacího systému,** na které **je nutné se dostat** z libovolné stránky programové paměti.
- Zjednodušení **adresového dekodéru**
- **Velikost a obsah** globální oblasti – efektivita
- Uvedený model s programovou pamětí přesahující přímo adresovatelný prostor procesoru podporuje C51.exe a „bankovací linker“ BL51.exe
- Procesory (ST electronic, Atmel) mají kapacitu programové paměti > 64kB
- Rozdělení na globální a zbývající oblast je někdy určeno výrobcem.



PŘEADRESOVÁNÍ PROGRAMOVÉ PAMĚTI ZA PROVOZU

- Umístění programové a datové paměti u **vývojového systému. Výhodnost shody s aplikační konfigurací.**
- Pro rychlý vývoj je vhodné využívat RAM jako programovou a současně i datovou paměť.
- Problém umístění programové a datové paměti **RAM** od adresy **0000h**.
- **Po zapnutí musí být na adrese 0000h paměť non-volatile (ROM, EPROM, Flash, FRAM, atd.).**
- Řešení se dvěma pamětmi např. FLASH
- **Řešení s jednou pamětí Flash**
- **Obvodové předadresování**

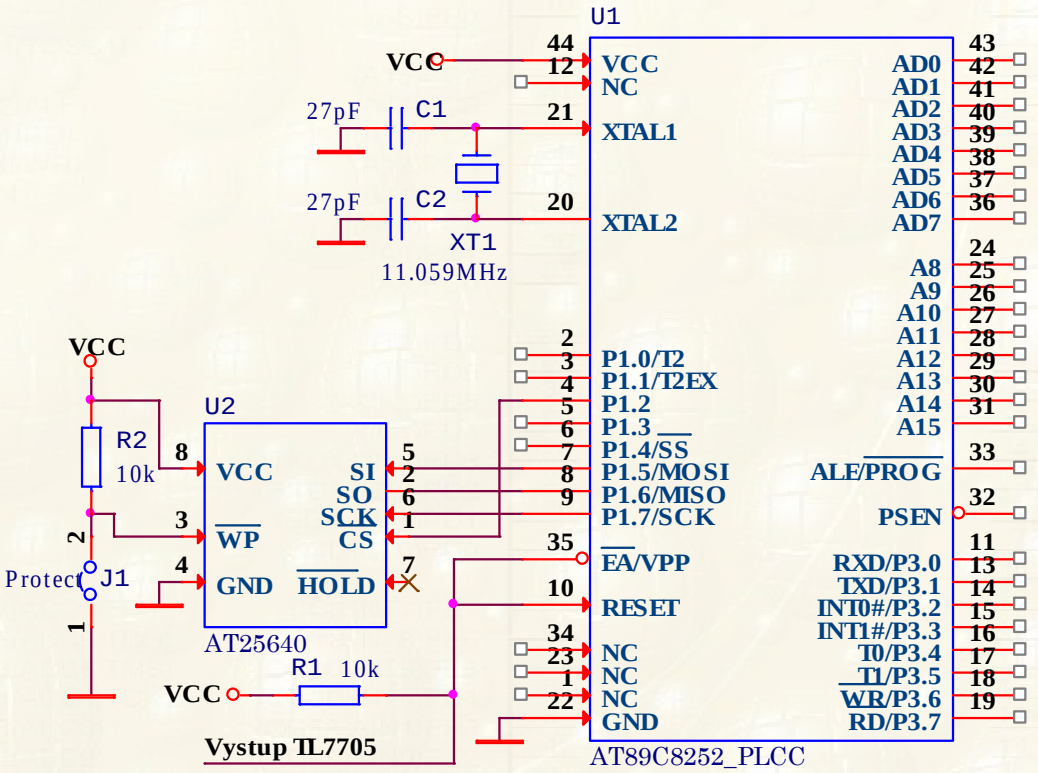


Sériové připojení paměti k mikroprocesoru

SÉRIOVÉ PŘIPOJENÍ PAMĚTI EEPROM

- Sériové připojení pamětí se realizuje přes přístrojová rozhraní SPI, I2C, Microwire, 1-Wire nebo vývody I/O procesoru.
- Přístup do požadované paměti je možné realizovat pomocí
  - **programu** (simulujícího příslušnou sběrnici - pomalejší přístup)
  - **řadiče sběrnice** (integrovaném v procesoru - rychlejší přístup).

- Paměť může obsahovat: **konfigurační konstanty, přístupové kódy, nelineární funkce, uživatelský interpretovaný program.**
- **Obsahem nemůže být program ve strojovém kódu procesoru**



SÉRIOVÉ PŘIPOJENÍ PAMĚTI EEPROM-PROGRAMOVÁ OBSLUHA

Malý počet připojovacích vodičů umožňuje snazší realizaci PCB a lepší odolnost proti EMC (lepší vykrytí zemní plochou).

```
#ifdef SPIEEPROM
#ifndef SPI_KANAL     // PROGRAMOVY PRISTUP DO SPI EEPROM
unsigned char cti_exbyte(unsigned int adresa)
{    unsigned char i,znak;
     CLKEE=0; CSEE=0; znak=0x03;  // Čteni
     for (i=0; i<8; i++)          // Vysílání hodnoty znak
     {    MOSI=0; if ((znak&0x80)!=0) MOSI=1;
          znak+=znak; CLKEE=1; CLKEE=0; }  // Hod.impulz
     for (i=0; i<16; i++)         // Vysílání adresy
     {    MOSI=0; if ((adresa&0x8000)!=0) MOSI=1;
          adresa+=adresa; CLKEE=1; CLKEE=0; } // Hod.impulz
     znak=0;
     for (i=0; i<8; i++)
     {    znak+=znak;             // Posun přijímaných bitů
          if (MISO==1) znak+=1; CLKEE=1; CLKEE=0;}
          CSEE=1;                 // Deaktivace EEPROM
          return(znak);
}
```

SÉRIOVÉ PŘIPOJENÍ PAMĚTI EEPROM-S VYUŽITÍM ŘADIČE SPI

```
//   PRISTUP DO SPI EEPROM POMOCI SPI KANALU pro
//   STM32F042K6 bez přerušení
unsigned char cti_eexbyte(unsigned int adresa)
{    unsigned char i;
     GPIOA->BSRR |= 0x08000000;  // Aktivace EEPROM
     i=vysli_prijmi_byte(0x03);  // Instrukce čtení
     i=vysli_prijmi_byte(adresa>>8); // Vyšší část
     i=vysli_prijmi_byte(adresa&0xFF); // Nižší část
     i=vysli_prijmi_byte(0x00);      // Hod
     GPIOA->BSRR |= 0x0800;      // Daktivace EEPROM
     return(i);
}
unsigned char vysli_prijmi_byte(unsigned char vysli)
{    *((uint8_t *)&(SPI1-> DR))=vysli; // Byte do vys.buff
     while ((SPI1->SR&SPI_SR_TXE)!=SPI_SR_TXE);
                                    // Čekání na TXE=1
     while ((SPI1->SR&SPI_SR_RXNE)!=SPI_SR_RXNE);
                                  // Čekání na naplnění RX
     return(SPI1->DR);
}
```

SÉRIOVÉ PŘIPOJENÍ PAMĚTI EEPROM - DOBA PŘÍSTUPU

Čtení bytu závisí hodinovém kmitočtu procesoru a rychlosti vykonání jednotlivých instrukcí po přeložení do strojového kódu procesoru.

- **Programová obsluha** EEPROM po SPI

  Význam symbolických názvů použitých v programu:

  CLKEE - výstup vytvářející hodinový signál pro paměť SPI
  MOSI - výstup přenášející data do vstupu MOSI paměti SPI
  MISO - vstup pro příjem dat přenášených z paměti SPI

  Řádově to budou stovky µs.

- **Přístup přes řadič SPI**

  Význam použitých názvů je následující:

  TXE –vysílací buffer prázdný,
  RXNE – přijímající registr není prázdný.

  Doba čtení bytu závisí na nastavení kmitočtu SCK generovaného procesorem. Maximální SCK pamětí > SCK procesoru.

  Doba přístupu k bytu bude řádově za $32x(1/f_{CLK})$ (8÷25µs).

Vstupy a výstupy

VSTUPY A VÝSTUPY MIKROPROCESOROVÉHO SYSTÉMU

- U aplikací s univerzálními a signálovými procesory, u rozsáhlých (někdy i malých) aplikací s jednočipovými procesory a procesory realizovanými v PLD je **potřeba připojit vstupy a výstupy ke společné sběrnici.**
- Vstupy/výstupy mohou být realizovány:
  - Z klasických logických obvodů
  - Programovatelných obvodů (CPLD, FPGA)
  - Specializovaných obvodů se sériovým rozhraním (SPI, I^2C, 1-wire, Mikrowire).

V závislosti na připojení můžeme jednotlivé varianty rozdělit:

- **Paralelní V/V brány -** obvody s direktivním řízením (obvody musí svými parametry vyhovět časování µP) nebo obvody s potvrzováním přenosu (handshake)
- **Sériové V/V brány -** obvody se synchronním nebo asynchronním přenosem s bitovou synchronizací. Specializovanými obvody s přístrojovou sběrnicí (I^2C, SPI, Microwire, 1-Wire).

VSTUPY A VÝSTUPY MIKROPROCESOROVÉHO SYSTÉMU

Rozšiřující vstupy a výstupy můžeme připojit

- K obecně použitelným **V/V vývodům procesoru** (program)
- Pomocí **společné sběrnice** (instrukcí)
  - do vnějšího V/V prostoru procesoru (s využitím řídících signálů daného procesoru např. WR, RD, IORQ, (Z80) nebo R/W, IS, (TMS320))
  - do vnějšího datového prostoru procesoru (např. WR, RD (AVR,8051), WR, RD, MERQ (Z80) nebo R/W, DS (TMS320)).
  - Vstupní bránu můžeme začlenit i do programového prostoru.
- Vstupy a výstupy mohou být za pomoci posuvných registrů ovládány pomocí synchronního nebo asynchronního sériového kanálu.
- Vstupem a výstupem může být i FIFO paměť, která se využívá k rychlému přenosu dat zvláště mezi signálovými procesory.

VSTUPY A VÝSTUPY MIKROPROCESOROVÉHO SYSTÉMU

Rozhodnutí o použité variantě rozšíření V/V bran ovlivňuje

- počet dostupných vývodů na uP
- potřebný datový tok
- velikost navrhované desky
- cena materiálu

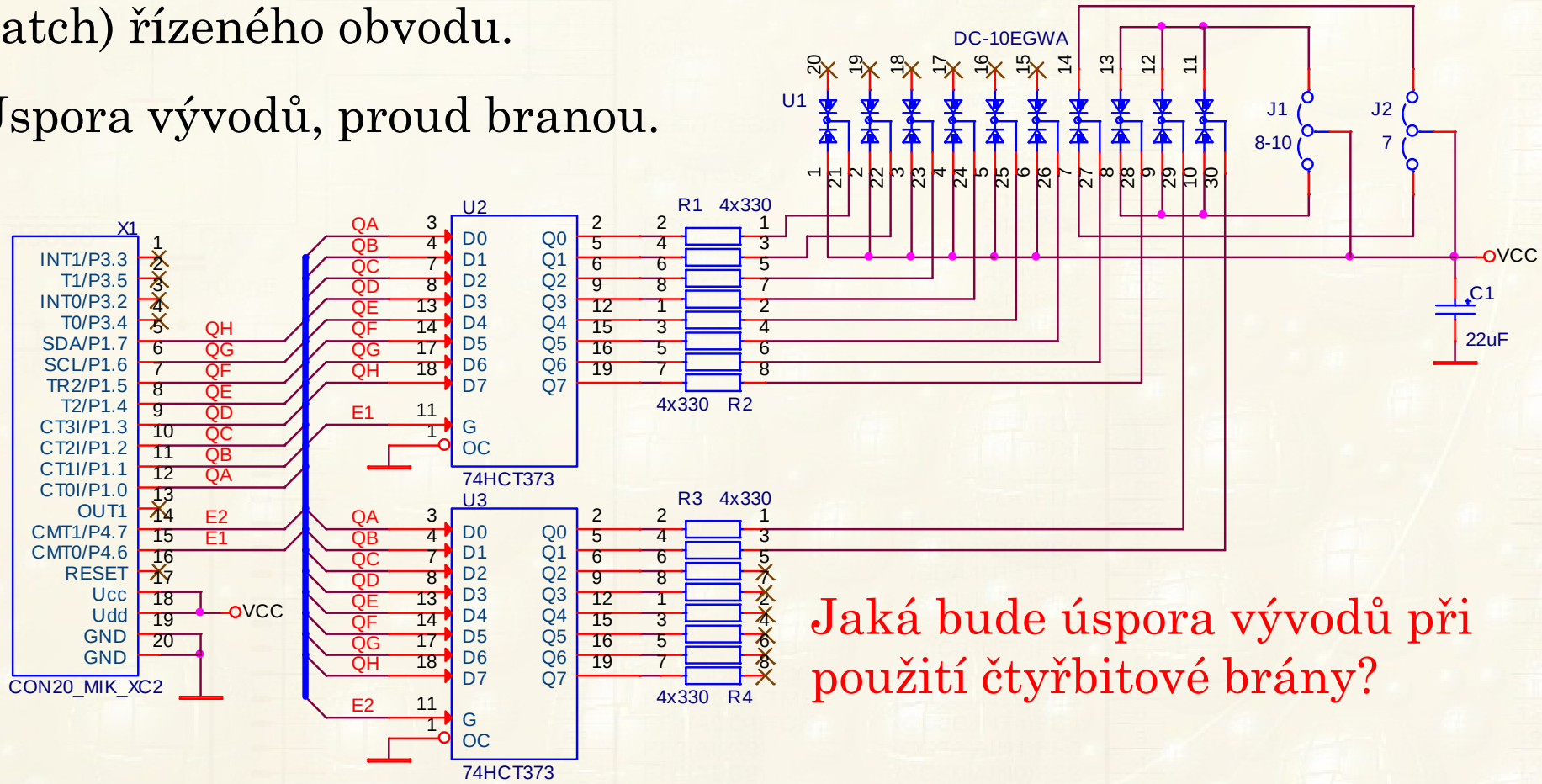
Postupná náhrada klasických obvodů **programovatelnými obvody**, které

- Mají vyšší cenu (u některých typů výrazně klesá pro 3,3V nebo 1,8V).
- Šetří prostor na desce plošných spojů.
- Možné úpravy logických funkcí bez zásahu do vlastního návrhu PCB
- Použití nemusí být výhodné při rozvodu signálů po celé desce.
- Pro malý počet klasických obvodů <6 nemusí být použití PLD ekonomické řešení

PROGRAMOVĚ OVLÁDANÉ ROZŠÍŘENÍ VÝSTUPŮ µP – PARALELNĚ - SÉRIOVÉ

Ukázka dřívější úlohy **světelný had,** tvořené 10 místným zobrazovačem BAR = jedno z možných zvětšení počtu výstupů uP. Realizována 4, 6 nebo 8 bitová sběrnice doplněná zapisovacími (hodinovými) signály E1 a E2. Program – uložení hodnoty na sběrnici – generován zapisovací impulz do hranově (registr) nebo úrovňově (latch) řízeného obvodu.

Úspora vývodů, proud branou.



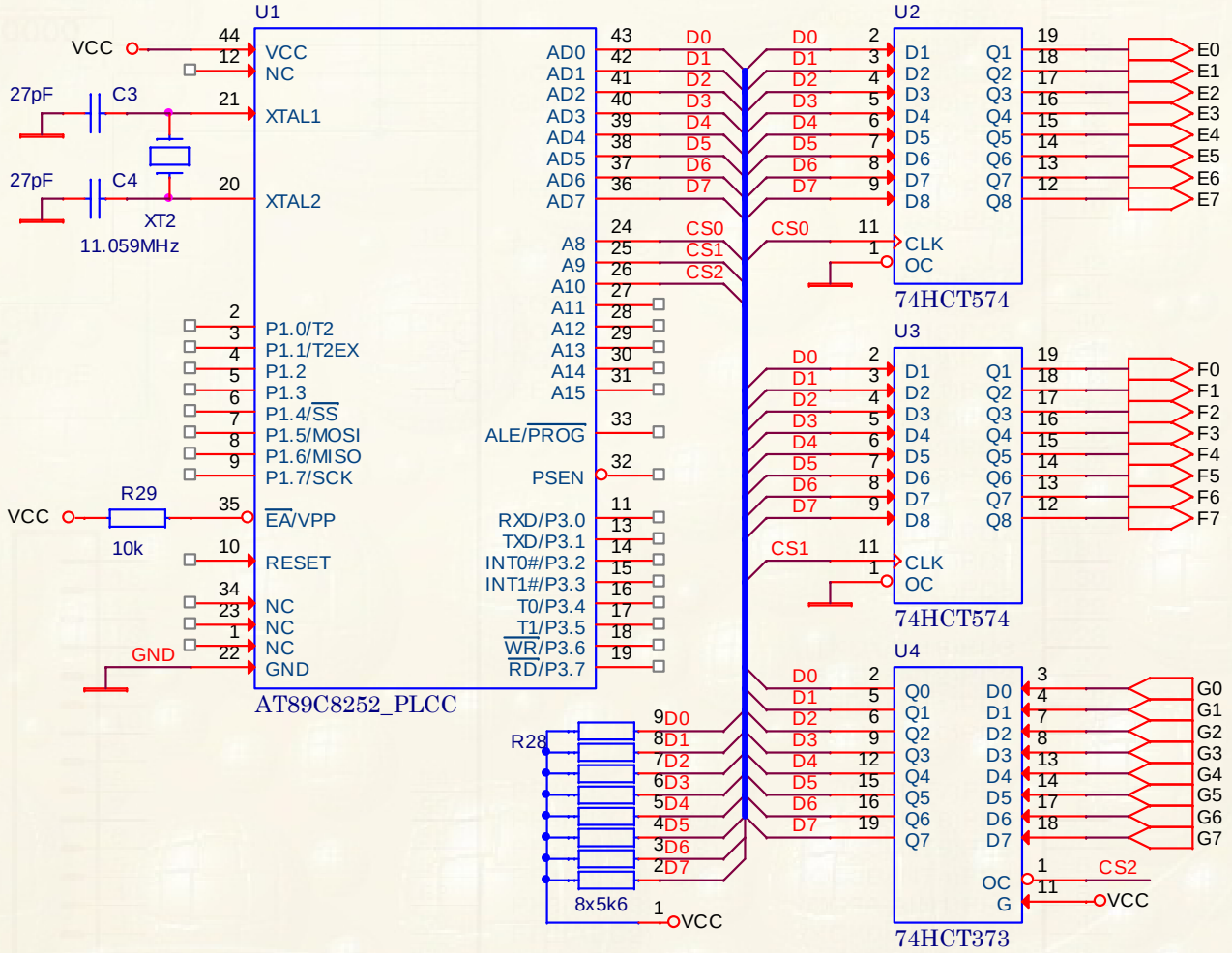
Jaká bude úspora vývodů při použití čtyřbitové brány?

PROGRAMOVĚ OVLÁDANÉ ROZŠÍŘENÍ VÝSTUPŮ µP – PARALELNĚ - SÉRIOVÉ

Rozšíření předcházejícího zapojení o paralelní vstupní bránu.

Změna obslužného programu

Úspora aktivačních signálů (pro >3 obvody) ⇒ dekodérem

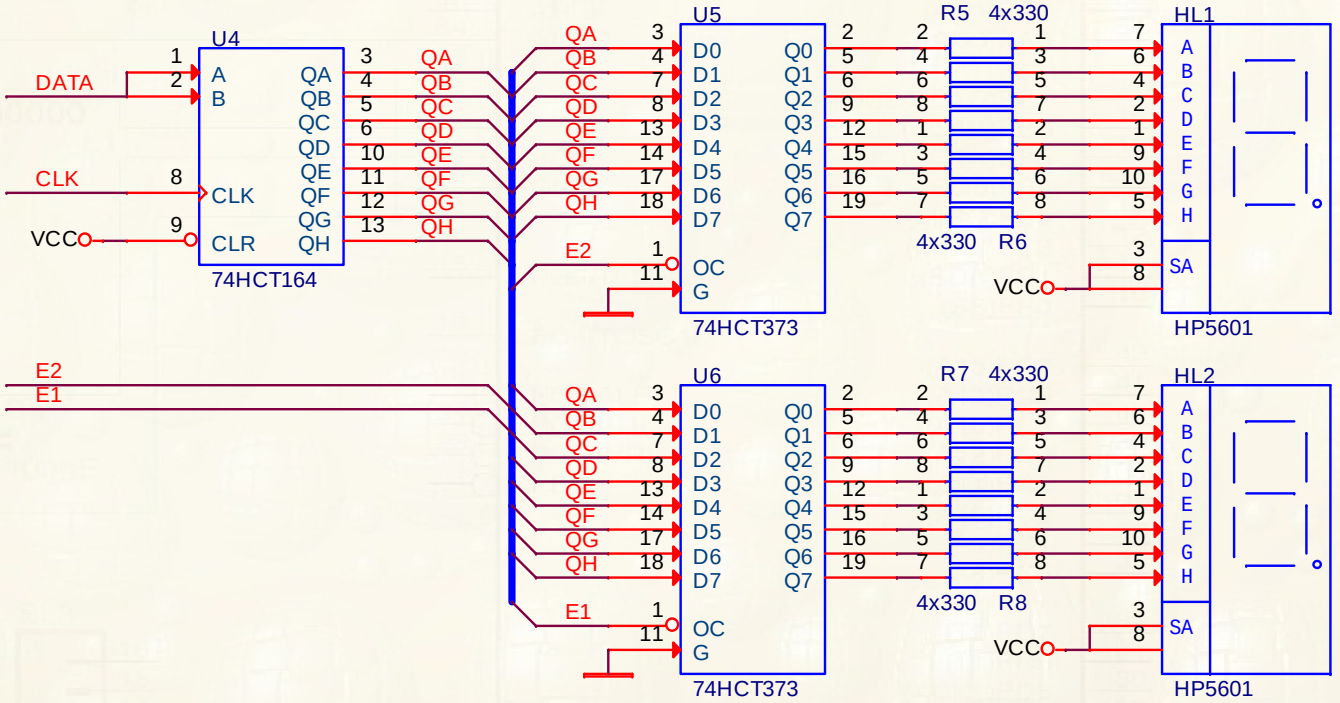


Ukázka obslužného programu

```
void zapis_U2(unsigned char hodnota)
{     P0=hodnota; // Zapisovaná hodnota na sběrnici
      CS0=0;      // Hodiny na log.0
      CS0=1;      // Vlastní zápis do obvodu
      CS0=0;      // U µP se slabou log.1 vhodnější
                  // neponechávat hod.signál v log.1
      P0=0xFF;}   // P0 do vstupního režimu

unsigned char cteni_U4(void)
{     unsigned char hodnota;
      P0=0xFF;    // P0 do vstupního režimu
      CS2=0;      // Aktivace třístavového budiče U4
      hodnota=P0; // Uložení přečtené hodnoty
      CS2=1;      // Deaktivace třístavového budiče
      return(hodnota);
}
```

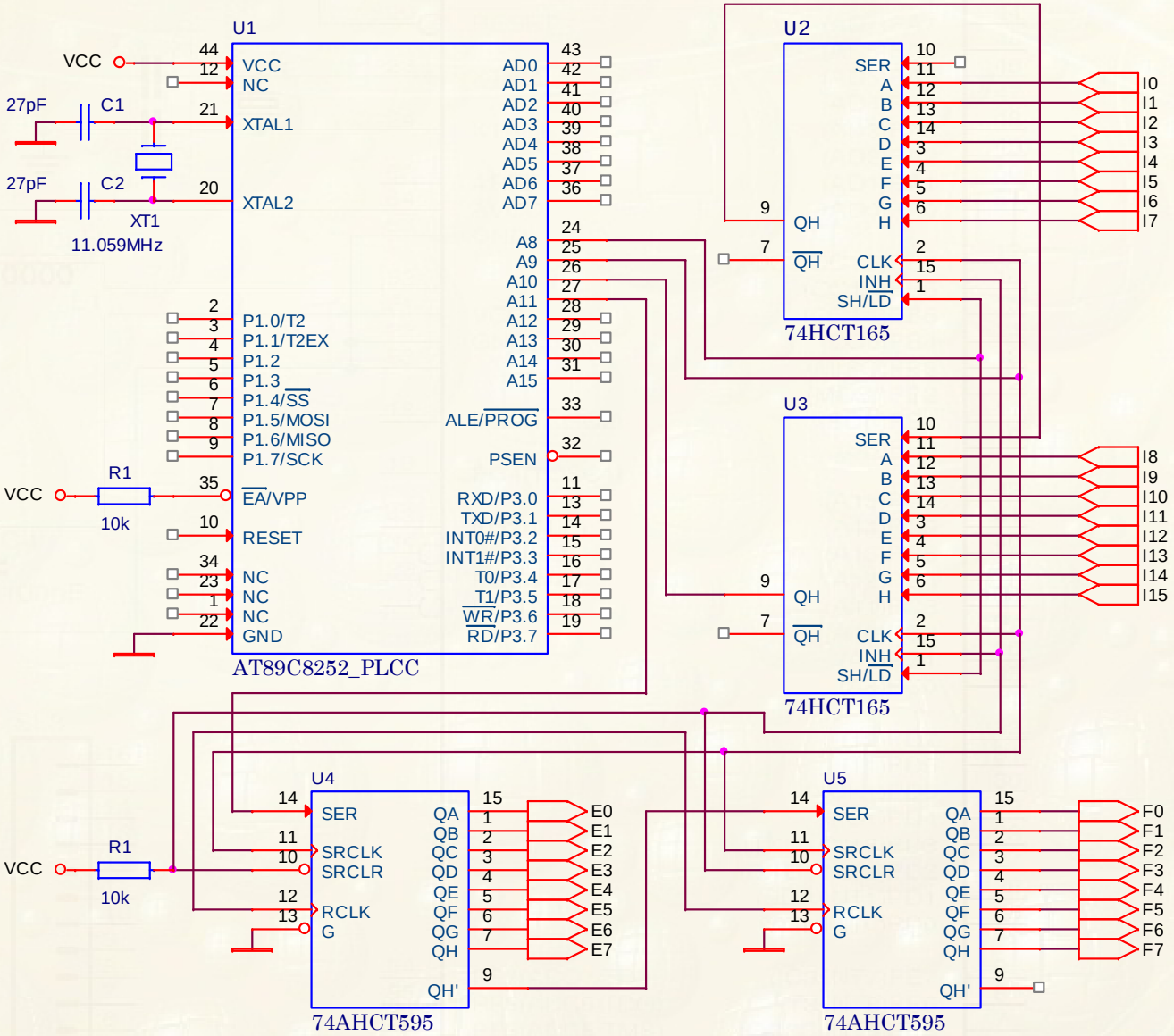
MOŽNÉ ZPŮSOBY ROZŠÍŘENÍ VÝSTUPŮ µP – SÉRIOVĚ-PARALELNÍ

Ukázka dříve realizované úlohy **stopky,** která je dalším možným rozšířením počtu výstupů uP. Sériový přenos na sběrnici, zápis do registru nebo latch.



Spojení posuvného registru s paralelním registrem se vyrábí jako IO (např. 594, 595). Na následující bláně jako rozšíření o 2x8 výstupů a vstupů.

SÉRIOVĚ OVLÁDANÉ VSTUPY A VÝSTUPY PROGRAMEM



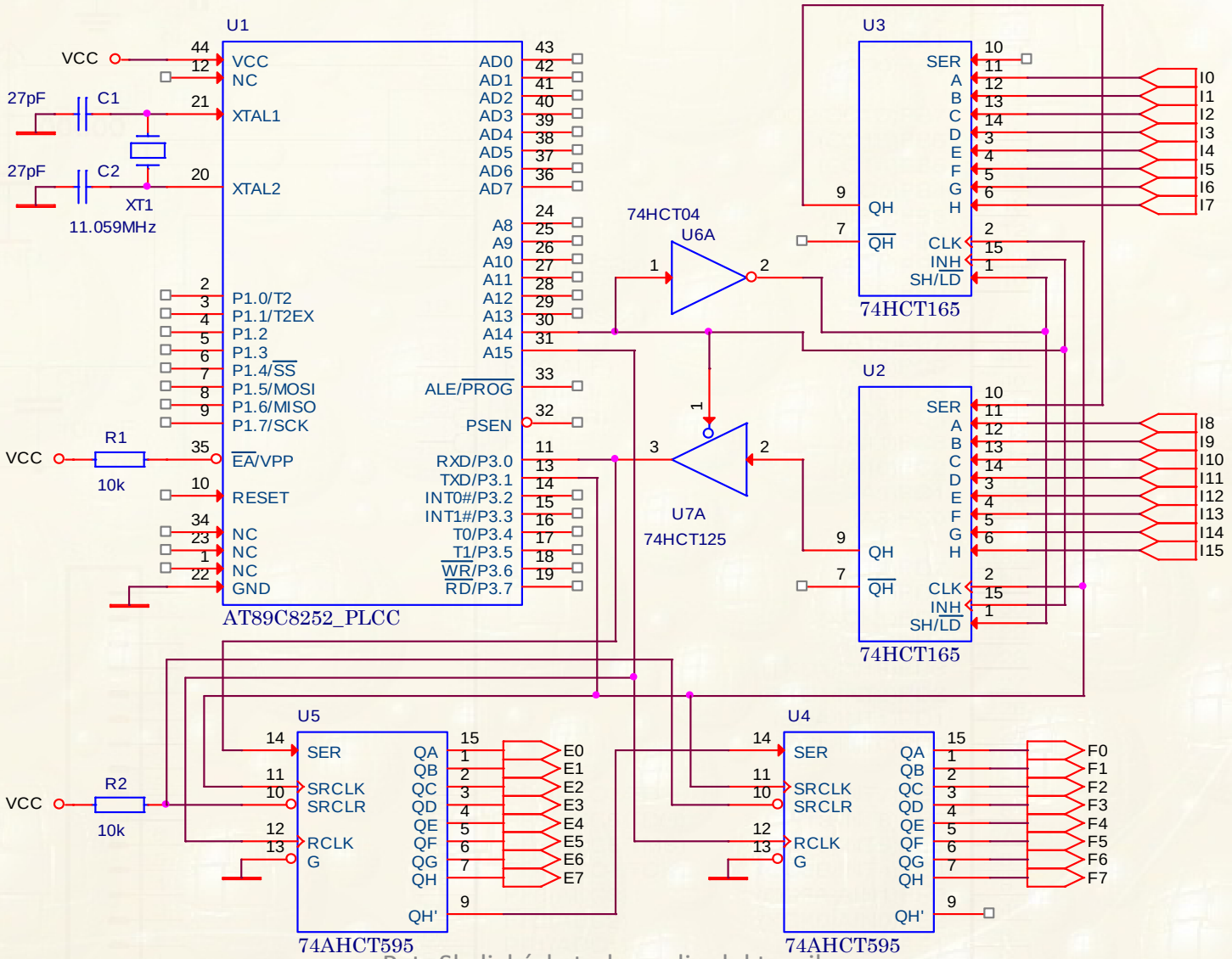
Sériové připojení k µP – pomalejší, než paralelní, ale potřebujeme menší počet V/V vývodů. Vodiče mohou být lépe stíněny zemní vrstvou ⇒ **vyšší odolnost vůči EMC**.

Symbolické názvy - LOAD=P2.0, CLK=P2.1, DOUT=P2.3, WRITE=P2.4

```
unsigned int vstup(void)
{    unsigned char opak; unsigned int hodnota;
     CLK=0;       // Hodinový signál do log.0
     LOAD=0;      // Load celé brány
     LOAD=1;      // Logické vstupy přečteny
     for (opak=0; opak<16; opak++)  // Cyklus
     {   hodnota=+hodnota;  // Rotace hodnoty o 1bit
                            // Výstup pos.registru U3
                 // na nejnižší bit proměnné hodnota
         if (DOUT==1) *((char*)(&hodnota)+1)|=0x01;
         CLK=1; CLK=0;          // Hodinový impulz
     }
     return(hodnota);           // Návrat z podprogramu
}
```

VSTUPY A VÝSTUPY OVLÁDANÉ SYNCHRONNÍM SÉRIOVÝM PŘENOSEM

Rychlejší přenos dosáhneme využitím sériového kanálu se synchronním přenosem, rozhraním SPI nebo I²S viz. modifikované zapojení s využitím sériového kanálu.



VSTUPY A VÝSTUPY V ADRESOVÉM PROSTORU

Vstupní a výstupní brána v prostoru datové paměti 9000h÷97FFh .

Adresa (hex)	Adresové vodiče															
	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0000h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0001h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
÷	0							÷								
8FFFh	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
9000h	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
÷								÷								
97FFh	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
9800h	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
÷								÷								
FFFFh	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Charakterizována stavem A15=1, A14=0, A13=0, A12=1 a A11=0, který se v daném prostoru nemění.

Výstupní brána - **registr (latch)**

Vstupy registru připojeny na datovou sběrnici.

Výstupy registru = výstupní brána.

Hodinový signál = výstup adresového dekodéru spolu s řídícím signálem WR.

Výkonná hrana pro **hranově řízený obvod =** přechod WR 0→1.

Hodinový vstup pro **úrovňově řízený obvod** bude H=1 pro WR=0.

VSTUPY A VÝSTUPY V ADRESOVÉM PROSTORU

Vstupní brána = obvod s **třístavovým výstupem.**

Výstupy obvodu připojeny k datové sběrnici.

Vstupy obvodu = vstupní brána.

Řízení třístavových výstupů OE = adresový dekodér dekódující vodiče A15 až A11 (10010) a řídící signál RD=0. V opačném případě **musí být výstupy ve stavu vysoké impedance.**

Možné problémy:

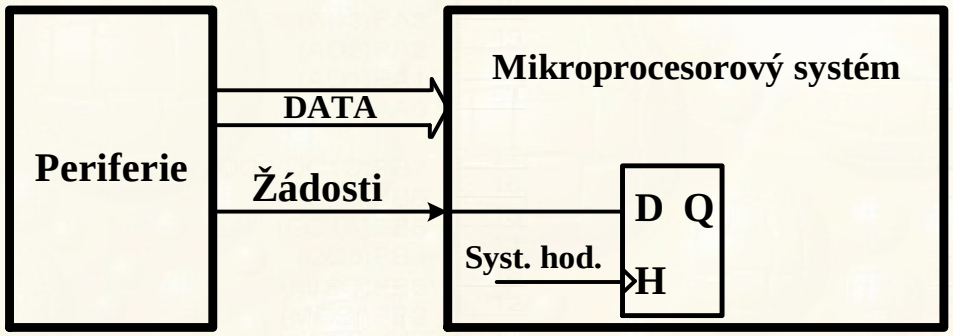
- Čtení hodnoty procesorem v okamžiku změn na vstupu
- **Asynchronnost** změn na vstupní bráně a hodinového kmitočtu procesoru
- Dodržení doby $\mathbf{t_{setup}}$ registru.

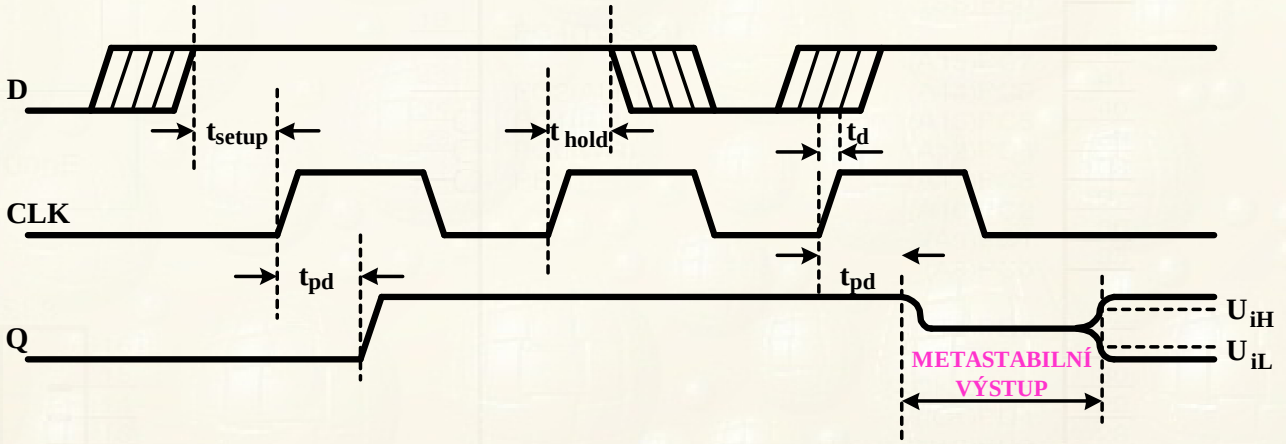
Metastabilita

Problém synchronizace

PROBLÉM SYNCHRONIZACE - METASTABILITA

Změny vstupních hodnot (dat) budou skoro vždy asynchronní k hodinovému signálu procesoru (vzorkujícímu obvodu) ⇒ **nedodržení doby předstihu t_{setup} pro zachycující obvody**
⇒ **problém metastability.**
Kritické časové okénko leží v rozsahu $\mathbf{t_d \in (10 \div 150)[ps]}$.





Střední doba mezi výskyty metastabilního chování závisí na t_d, synchronizačním kmitočtu systému f_{clk} a kmitočtu vstupního signálu nebo jeho změn f_{in}.

$$MFBT = \frac{1}{f_{in} * f_{clk} * t_d}$$

PROBLÉM SYNCHRONIZACE – METASTABILNÍ CHOVÁNÍ

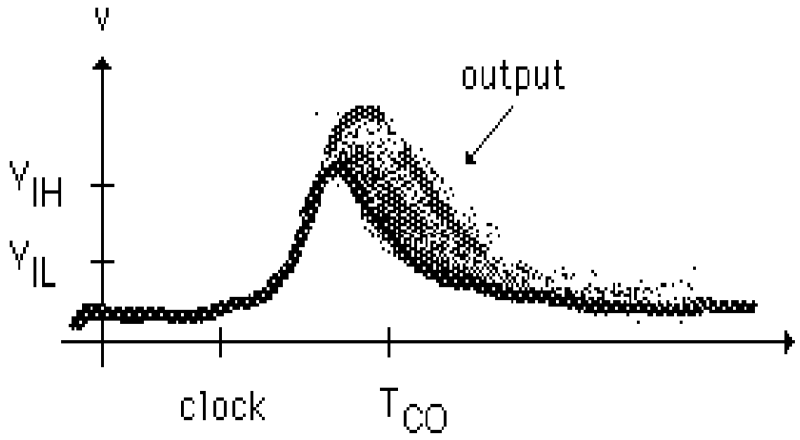


Figure 1a. Runt Pulse

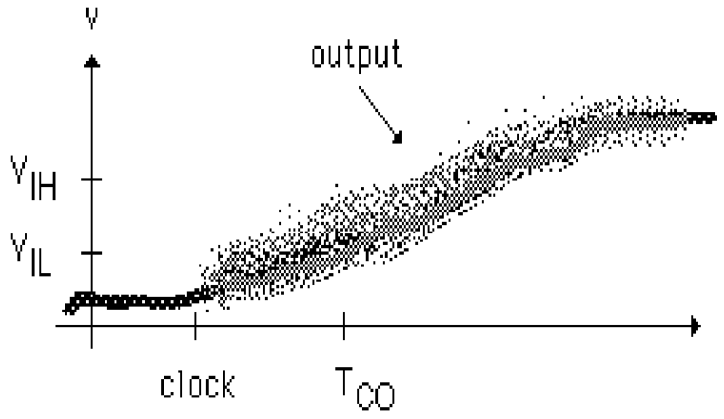


Figure 1b. Decreased Slew Rate

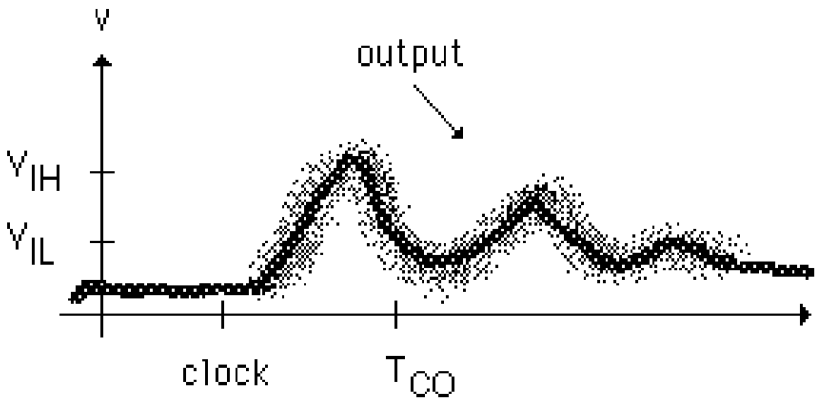


Figure 1c. Output Oscillation

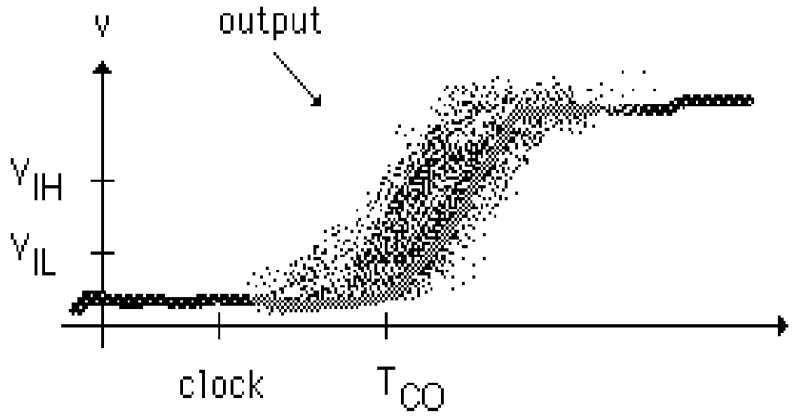
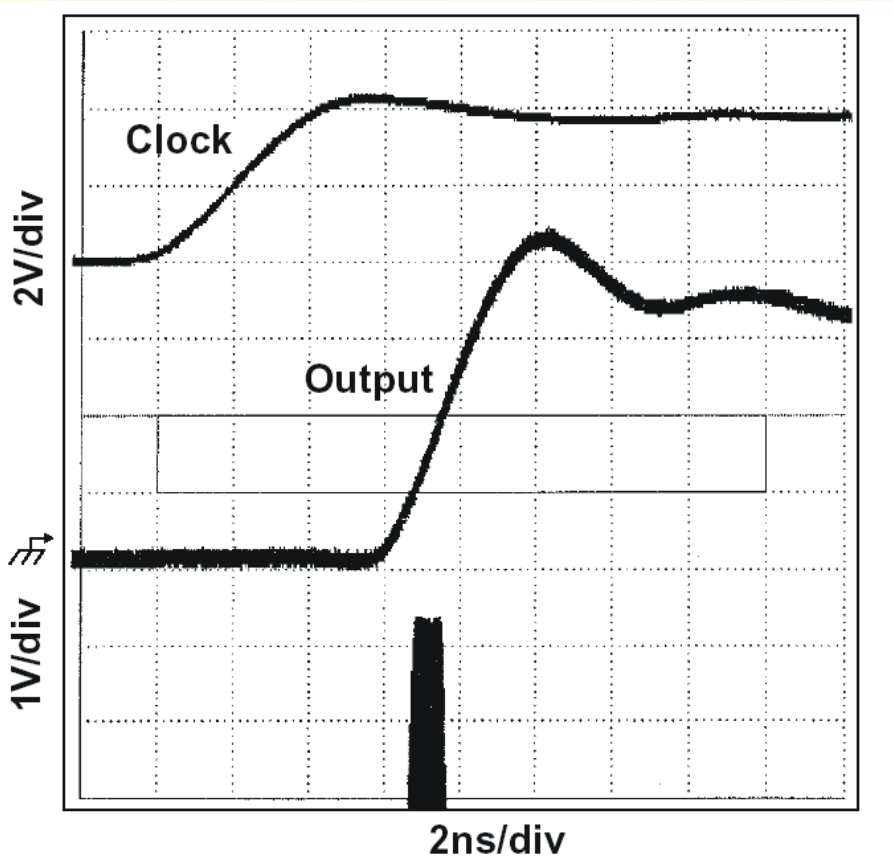
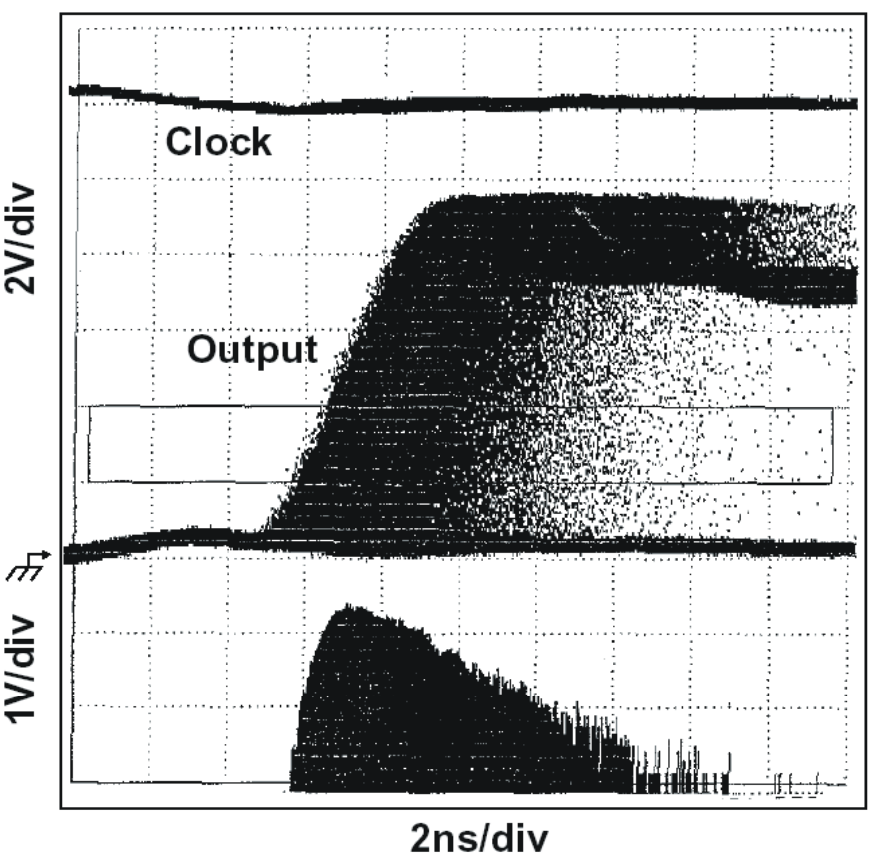


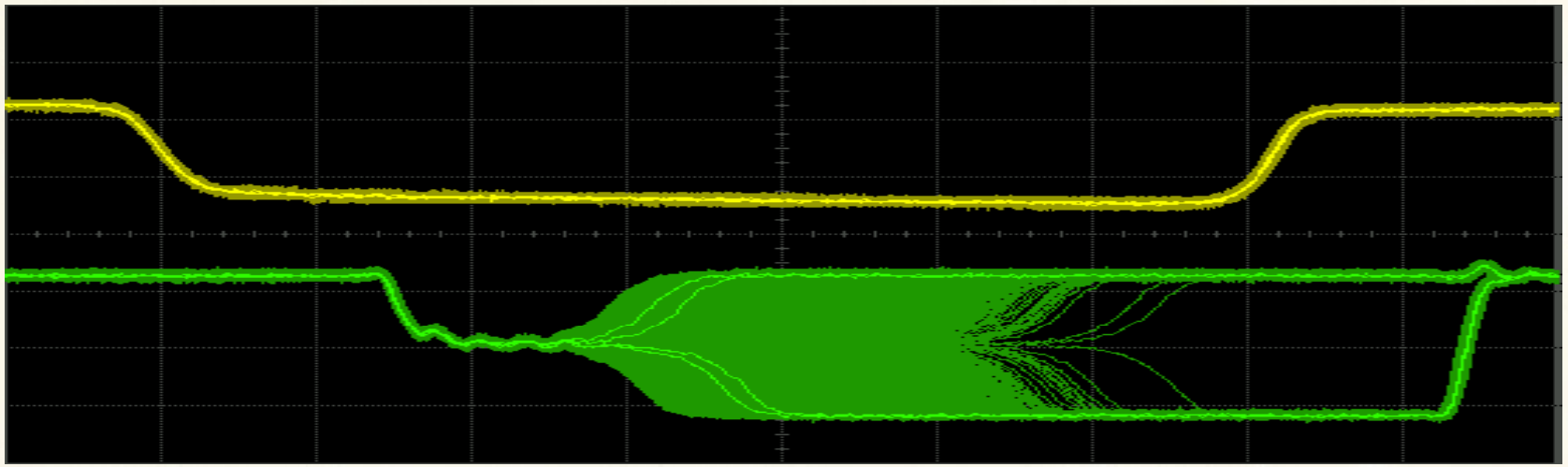
Figure 1d. Increased T_{co}

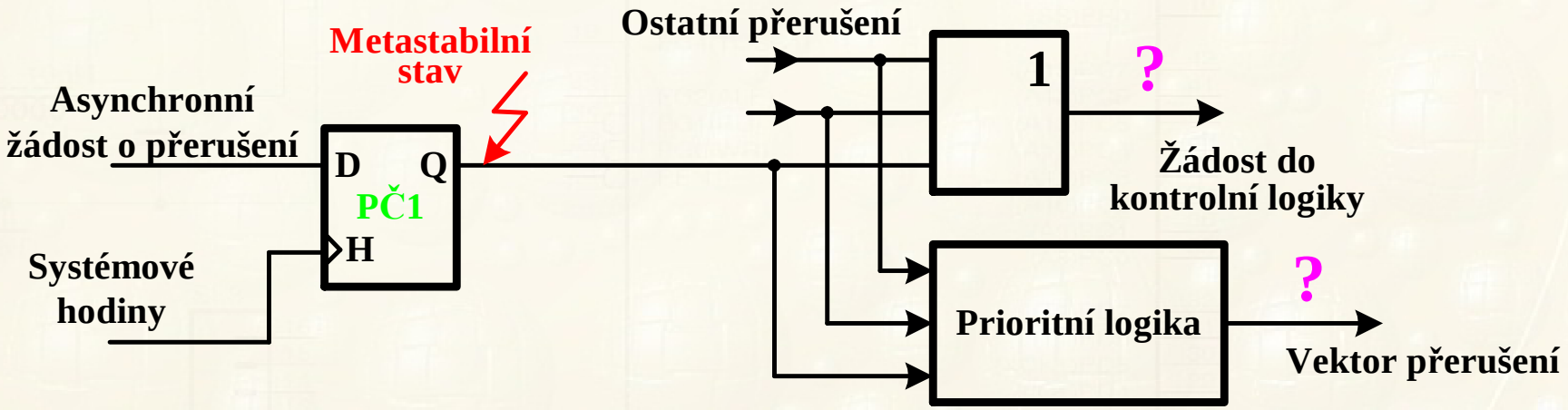
PROBLÉM SYNCHRONIZACE – SN74AS74





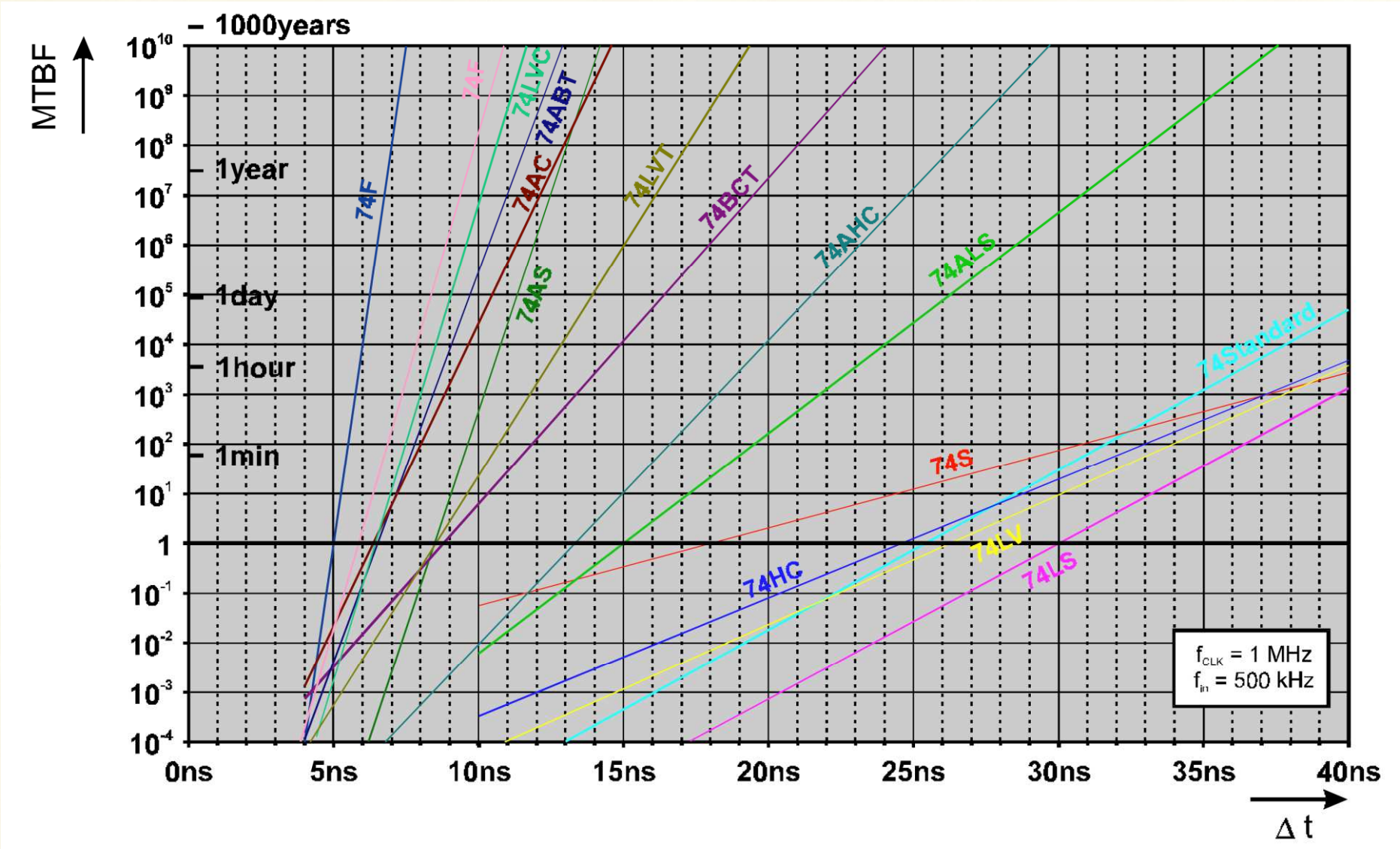
SYNCHRONIZACE – DOPADY METASTABILNÍHO CHOVÁNÍ





Dojde-li u vstupu přerušení μP k metastabilnímu stavu ⇒ může v důsledku nedefinované (nesprávné logické úrovně) dojít k chybnému vyhodnocení přerušení v kontrolní logice stavu přerušení nebo přerušovacího vektoru v logice stanovující adresu přerušení.

PROBLÉM SYNCHRONIZACE – VLASTNOSTI OBVODŮ



Z diagramu zjistíme, že CMOS obvody se rychleji opouští metastabilní stav (zdroj Texas Instruments).

PROBLÉM SYNCHRONIZACE – VÝPOČET SPOLEHLIVOSTI

Střední doba mezi chybami v synchronizaci je dána vztahem

$$MFBT = \frac{e^{(T.\Delta t)}}{f_{in} * f_{clk} * t_d}$$

kde T a t_d jsou parametry technologie LO a Δt je doba zpožděného testování správnosti výstupu. **Jsou-li signály f_{in} a f_{clk} asynchronní**, vzniku **metastabilního stavu** zabránit nemůžeme.

- ✓ Pravděpodobnost chyby se rychle zmenšuje s rostoucí hodnotou $\Delta t \Rightarrow$ možnost zpožděného testování výstupu klopného obvodu. Hodnota T = směrnice v grafu z předcházející blány, hodnota t_d je dána polohou přímky v grafu.

$$T = \frac{\ln MTBF(2) - \ln MTBF(1)}{\Delta t(2) - \Delta t(1)} \qquad t_d = \frac{e^{(T.\Delta t)}}{MTBF * f_{clk} * f_{in}}$$

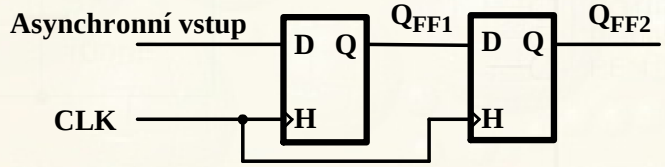
Například pro střední dobu chyby 100let v technologii ALS, f_{clk}=10MHz a f_{in}= 100kHz zjistíme, že musíme testovat výstup synchronizačního obvodu se zpožděním 37[ns].

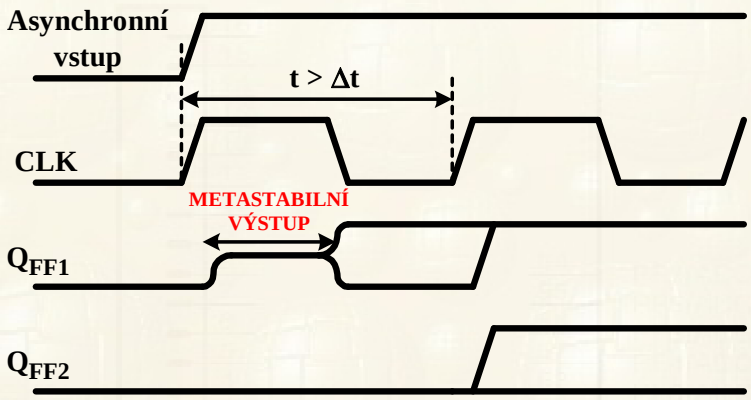
$$100let = 3.2 * 10^9 s = \frac{e^{(1,O2*\Delta t)}}{10^5 * 10^7 * 8.8 * 10^{-6}} \Rightarrow \Delta t = 37\,[ns]$$

DVOJNÁSOBNÝ SYNCHRONIZÁTOR – VÝPOČET CHYBOVOSTI

Testování výstupu synchronizačního paměťového členu zajistíme druhým paměťovým členem = **dvojnásobný synchronizátor**.

- ✓ Chybovost výstupu druhého PČ se odvozuje z frekvence chybných výstupů prvého paměťového členu.
- ✓ Pro nízkou chybovost synchronizace potřebujeme dlouhou dobu k testování ⇒ **omezení hodinového** synchronizačního kmitočtu.
- ✓ ⇒ Vývoj obvodů s velmi krátkou dobou trvání metastabilního stavu.





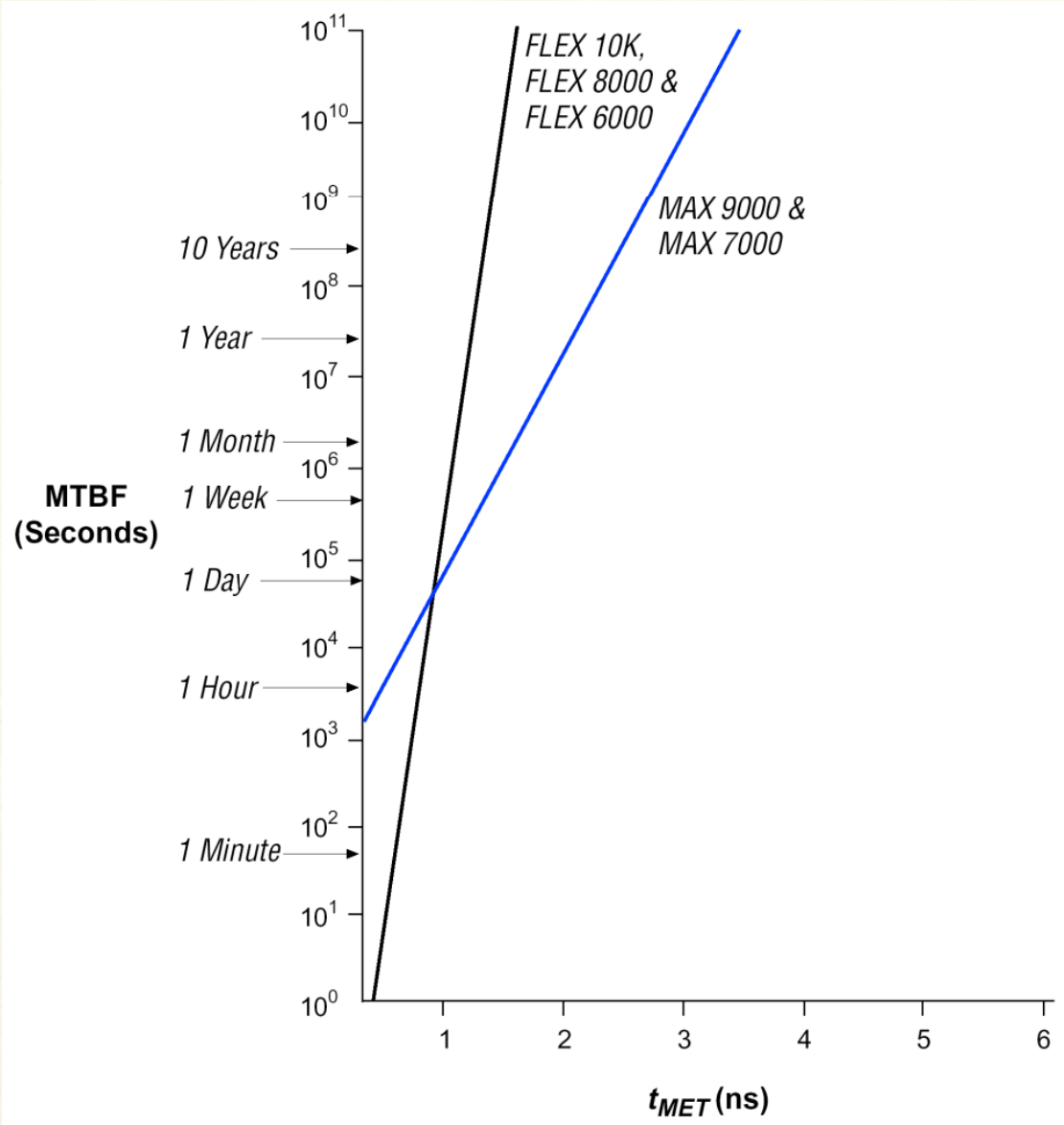
$$f_{in(2)} = \frac{1}{MFBT(1)} = \frac{f_{in(1)} * f_{clk} * t_d}{e^{(T.(1/f_{clk}))}}$$

$$MFBT(2) = \frac{e^{(T.\Delta t)}}{f_{in(2)} * f_{clk} * t_d}$$

$$MFBT(2) = \frac{e^{(T.\Delta t)} * e^{(T.(1/f_{clk}))}}{f_{in(1)} * f_{clk}^2 * t_d^2}$$

PARAMETRY MODERNÍCH OBVODŮ FIRMY ALTERA

Potřeba dosažení vysokého hodinového kmitočtu, donutila výrobce PLD k výrobě obvodů s velmi krátkou dobou trvání metastabilního stavu.

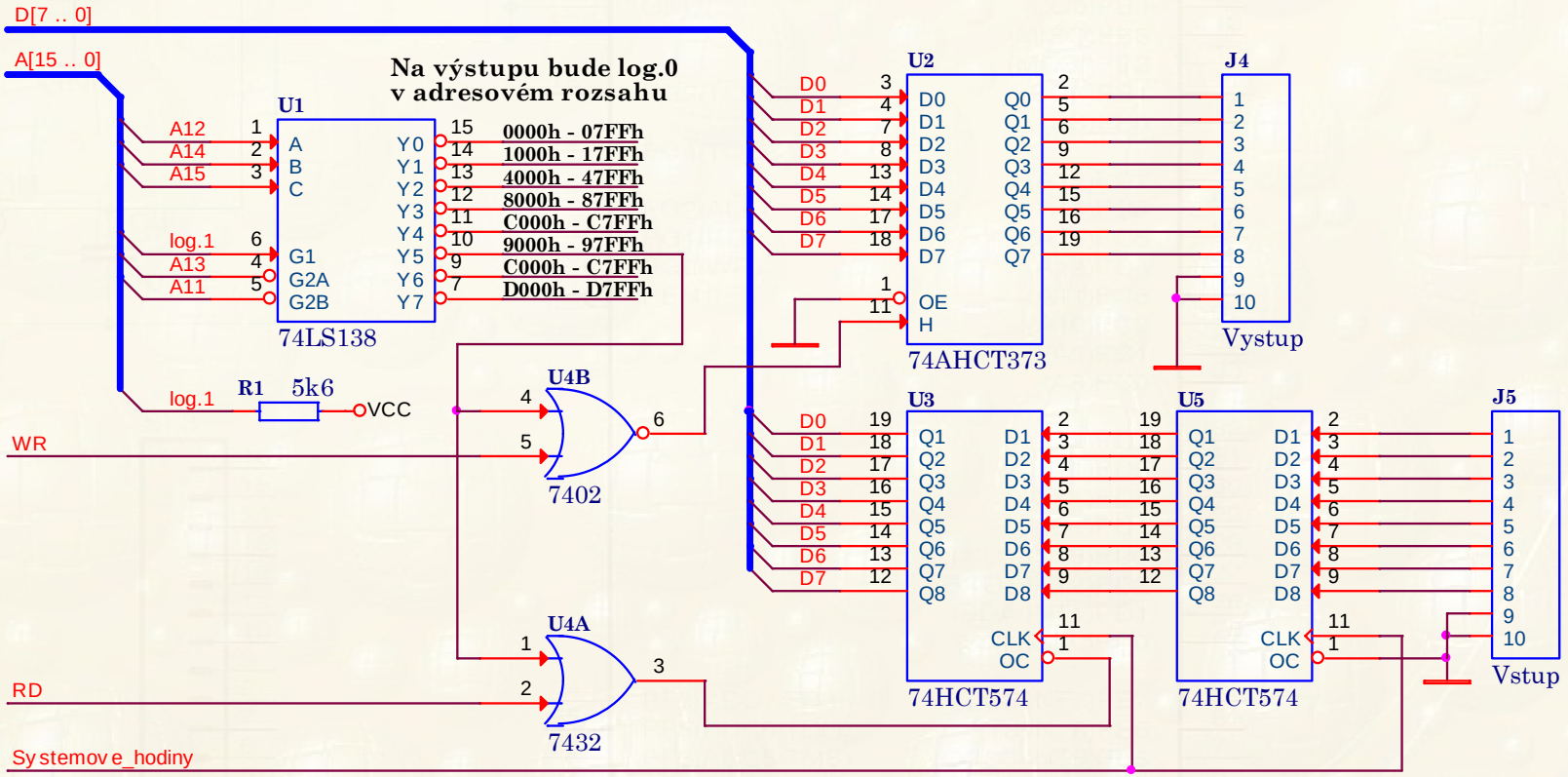


VSTUPY A VÝSTUPY V ADRESOVÉM PROSTORU

Vstupní brána s **dvojnásobným synchronizátorem** (procesor AVR).

Systémové hodiny musí mít takový kmitočet, aby přepis hodnoty z U5 do U3 probíhal v době ustáleného výstupu U5.

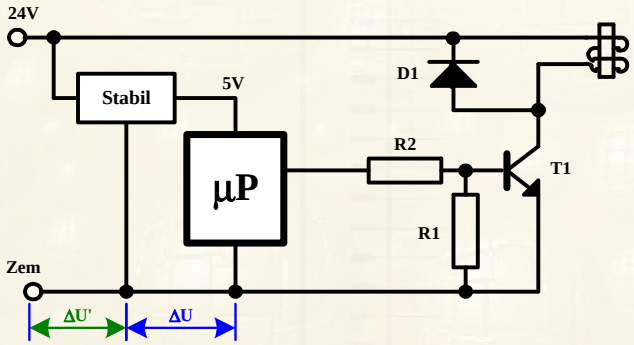
Okamžiku vlastního čtení obsahu U3 procesorem musí předcházet výkonná hrana systémových hodin minimálně o dobu $t_{pH \rightarrow Q}$.

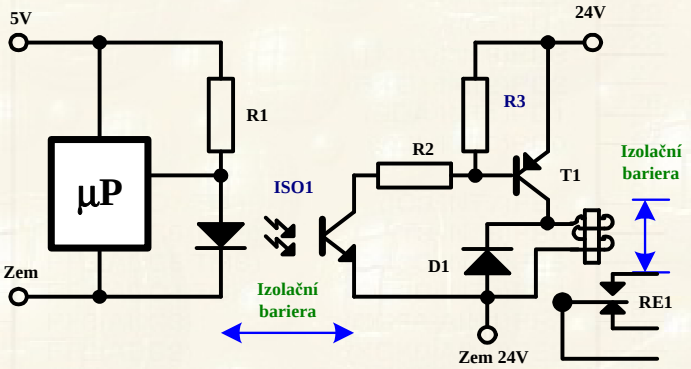


CHOVÁNÍ µP PŘI SPÍNÁNÍ VELKÝCH PROUDŮ

Spínání větších a velkých proudů µP můžeme rozdělit:

- **spínání větších proudů v galvanicky spojeném obvodu**
  - Úbytek napětí na **společném úseku zemního vodiče**
  - Změna napájecího napětí µP o hodnotu ΔU (modře označené)
  - Může způsobit desinterpretaci programu procesoru.
  - Jaký vliv má úbytek napětí ΔU (zeleně označený)
- **spínání velkých proudů v galvanicky odděleném obvodu**
  - Izolační bariéra relé, vliv magnetického pole kontaktů na cívku relé
  - Optické oddělení vývodů procesoru od cívky relé





EKONOMICKÉ ASPEKTY NÁVRHU µP SYSTÉMU

Návrh systému je primárně ovlivněn zadáním:

- ✓ **velikostí plošného spoje**
- ✓ **spotřebou**
- ✓ **možným přeprogramováním obvodové části**
- ✓ **možností dálkového přeprogramování systému**
- ✓ **možností dalšího rozšiřování programového vybavení**
- ✓ **předpokládanou prodejní cenou**

Pokud po zvážení omezujících podmínek zadání zůstává více možných řešení, přichází na řadu **ekonomické aspekty návrhu**.

1. **Systém s malým programem a větším počtem V/V vývodů**
2. **Systém se složitější logikou**
3. **Systém se složitou logikou (akcelerátory atypických operací)**