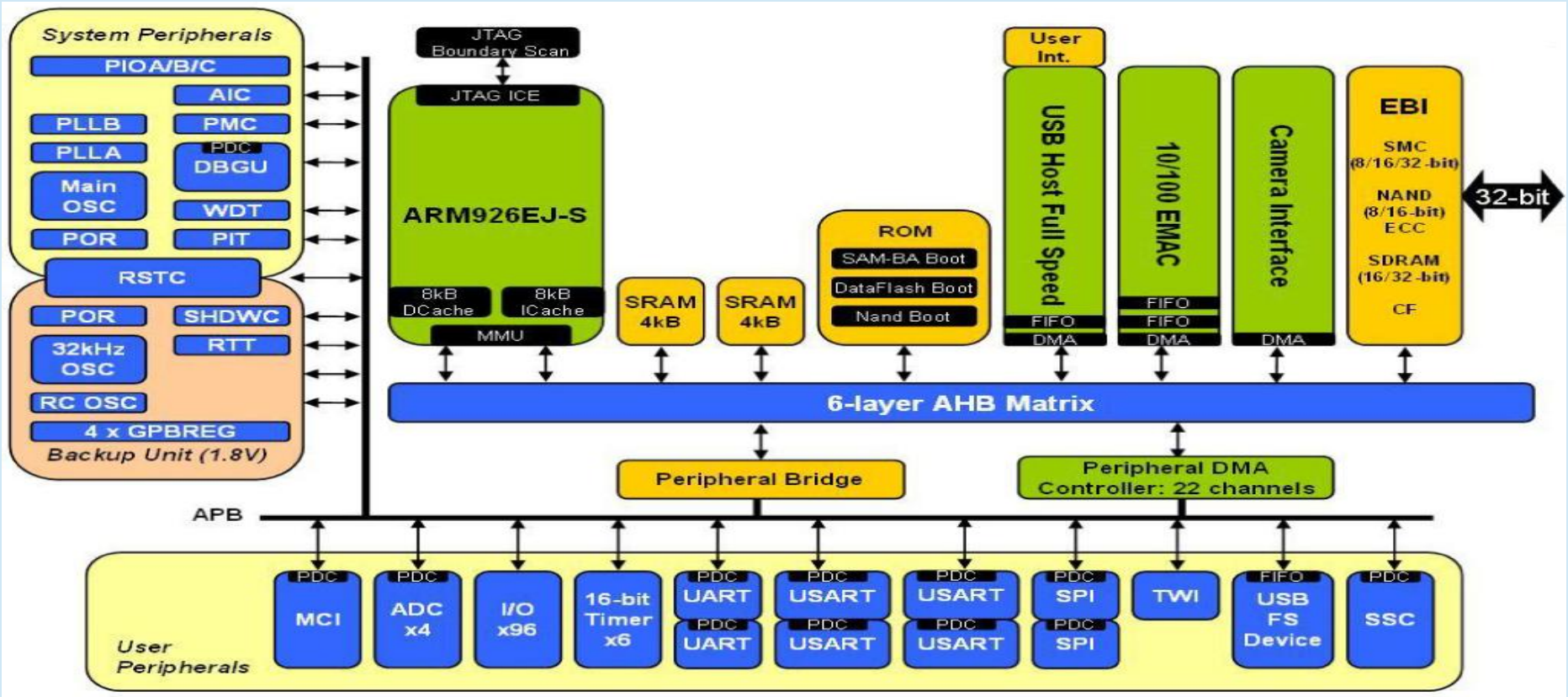
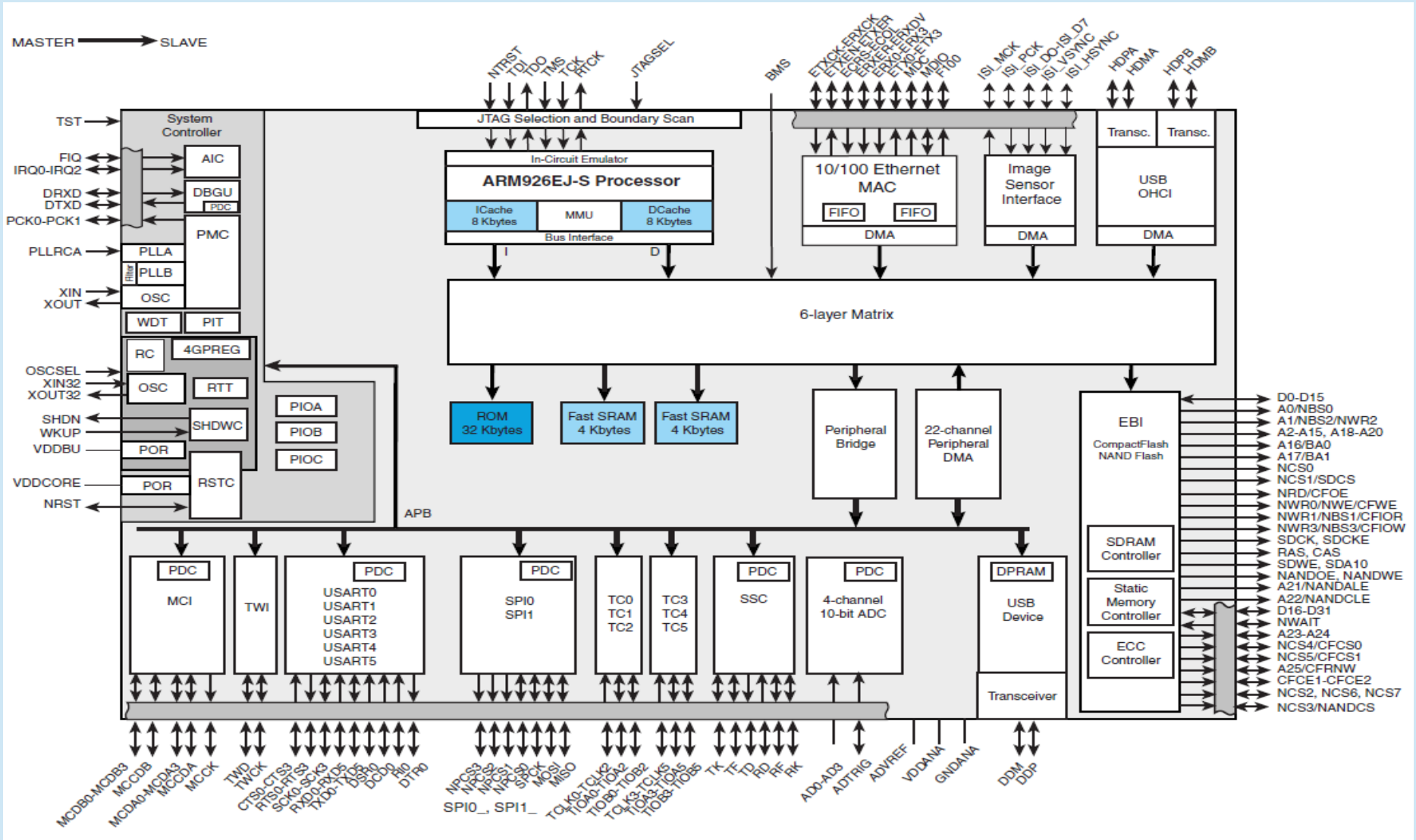
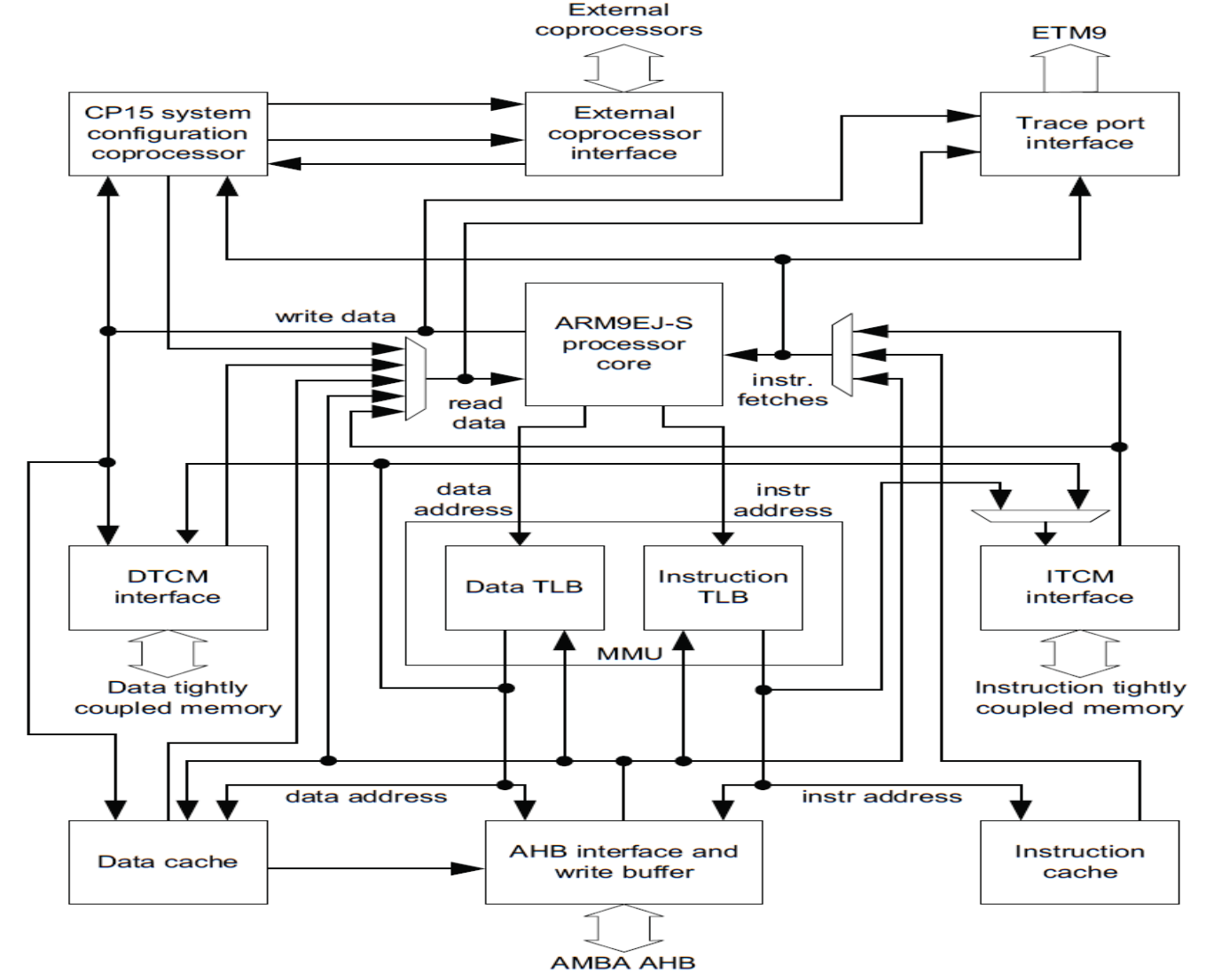
# AT91SAM9260



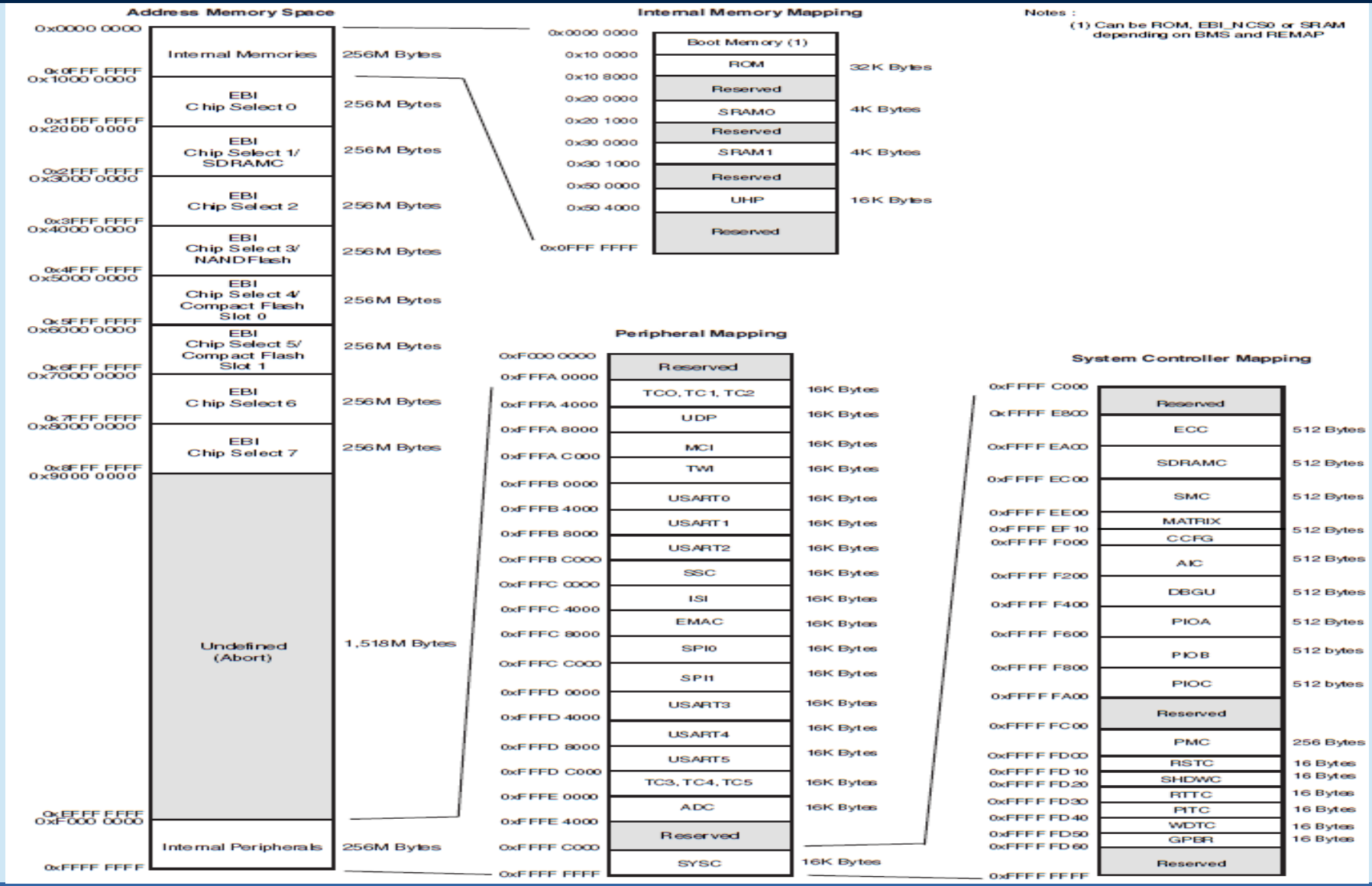
# Blokové schéma AT91SAM9260



# Blokové schéma jádra ARM926



# Mapa fyzického paměťového prostoru

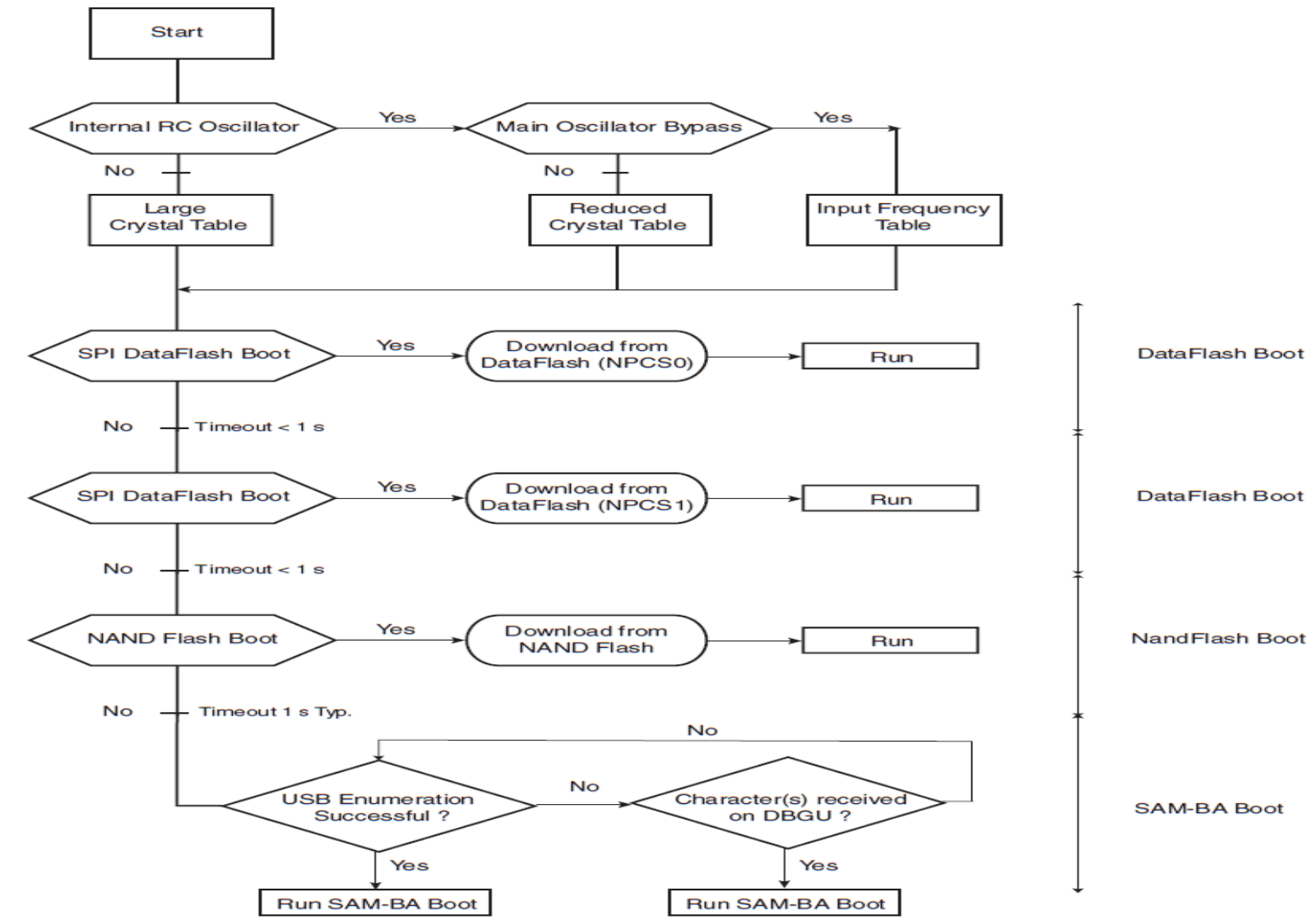


# Zavedení programu - BOOT

- **Vnitřní paměti procesoru**
  - ROM 32kB – pevný firmware zavaděče programu z externích pamětí
  - RAM 2x4kB – pro zavedení a spuštění zavaděče
- **Volba spuštění programu při startu (RESET)**
  - Signálem na pinu BMS
    - ROM při BMS = 1, umožňuje zavést a spustit program z těchto zdrojů
      - SPI DataFlash® na rozhraní SPI 0
      - 8/16 bitová NAND FLASH
      - Sériové rozhraní DBGU (nástroj SAMBA)
      - Port USB Device (nástroj SAMBA)
    - EBI_NCS0 při BMS = 0 (spuštění programu přímo v externí paměti)
- **Možnost přemapování interní RAM od adresy 0 (vektory přerušení)**

| Address | REMAP = 0 | | REMAP = 1 |
|---|---|---|---|
| | BMS = 1 | BMS = 0 | |
| 0x0000 0000 | ROM | EBI_NCS0 | SRAM0 4K |

# Algoritmus zavedení programu v int. ROM



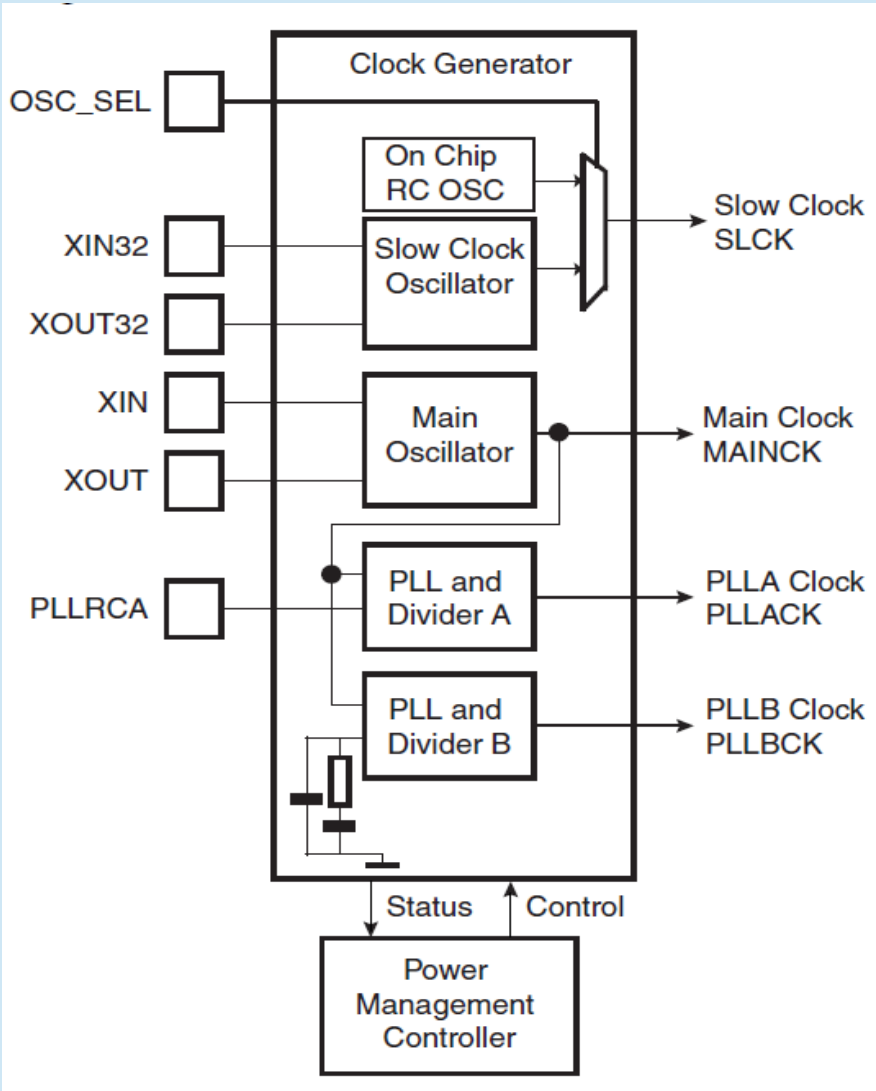
# Externí paměti - možnosti

- **Statické paměti (SRAM, ROM, FLASH …)**
- **SDRAM dynamické paměti**
- **ECC řadič (NAND FLASH)**
- **Datová sběrnice 32 bitů, adresová sběrnice 26 bitů (max. 64 MB / banka)**
- **8 signálů pro výběr banky**
  - Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2
  - Static Memory Controller on NCS3, Optional NAND Flash support
  - Static Memory Controller on NCS4 - NCS5, Optional CompactFlash support
  - Static Memory Controller on NCS6-NCS7

# Systémový řadič

- **Reset management**
- **Shutdown, power save management**
- **Generátor hodinových signálů**
  - Pomalý oscilátor 32.768 kHz (RTC, úsporný režim...)
  - Nízkopříkonový RC oscilátor
  - Hlavní krystalový oscilátor 3-20 MHz
  - 2 PLL moduly
  - PLLA <80..240> MHz
  - PLLB <70..130> MHz
- **Periodic Interval Timer**
- **Watchdog Timer**
- **Časovač – reálný čas**
- **Zálohované registry (4x32b)**
- **Advanced Interrupt Controller (AIC)**
- **Podpora ladění programu - Debug Unit**

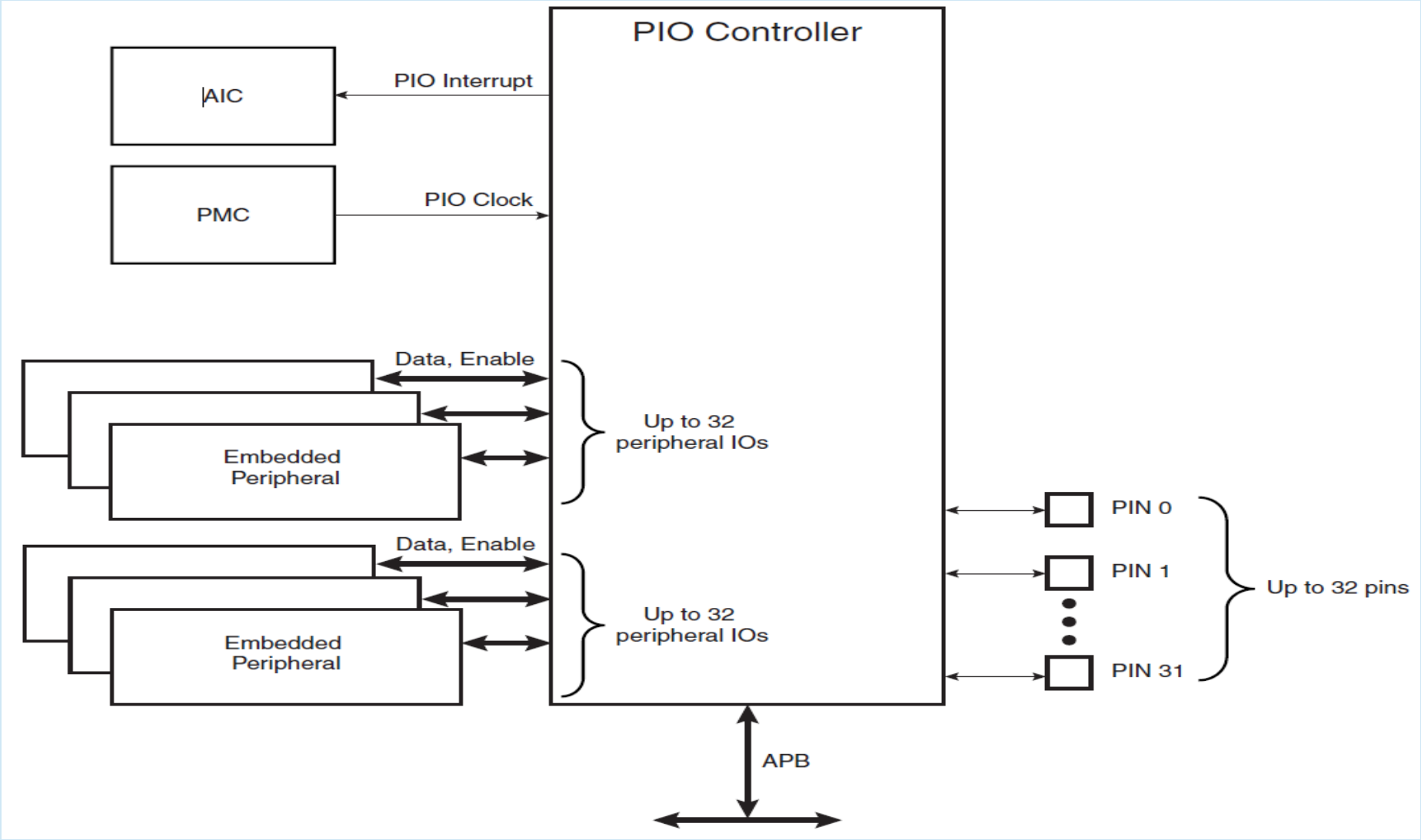
**A**



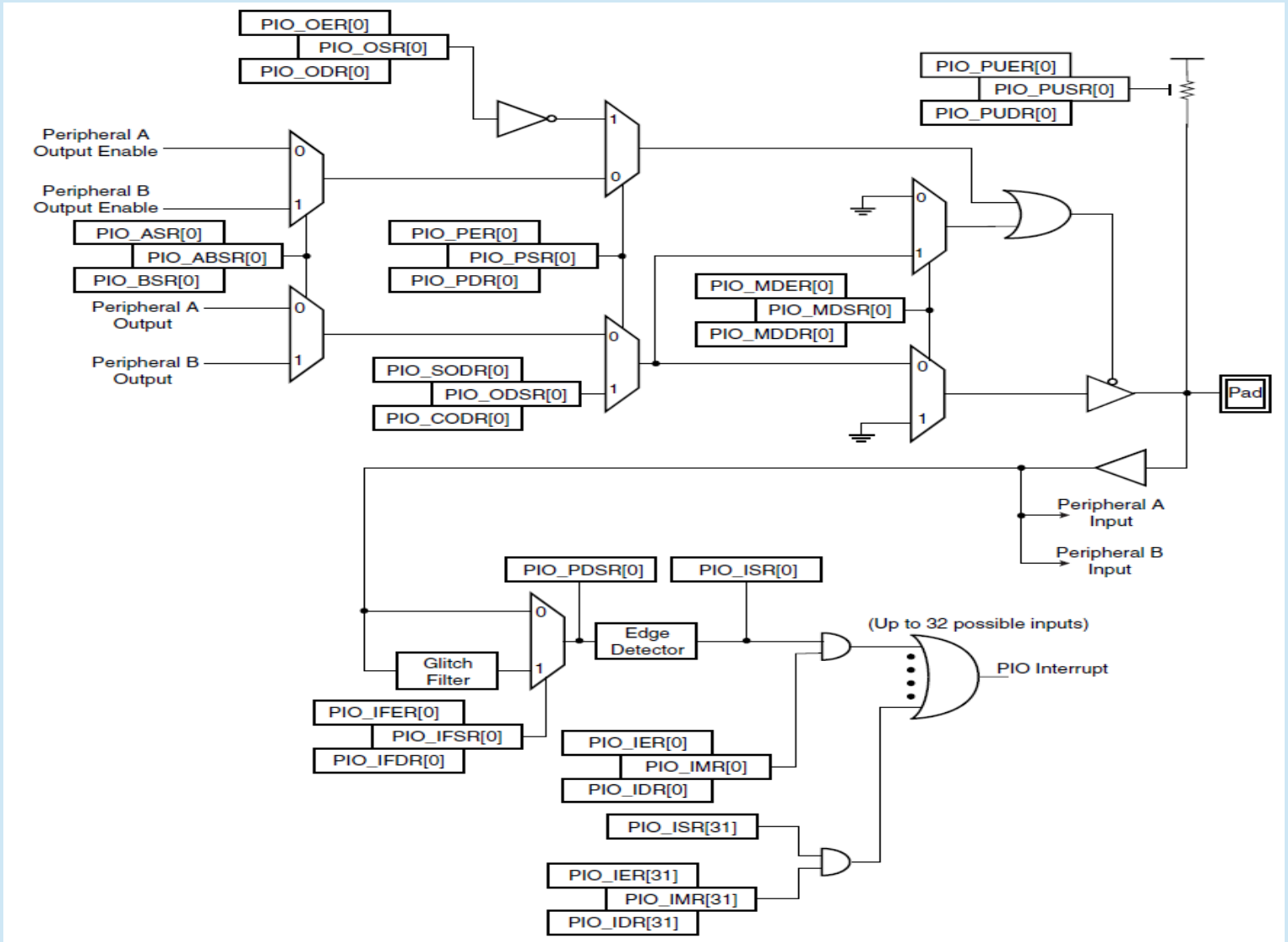
# PIO – univerzální vstupy/výstupy

- Organizovány po 32 bitech do portů A, B, C
- Nastavitelné individuálně jako vstupy, vstupy s PU rezistorem, výstupy PP a výstupy OC
- Ovládání jednotné stejnou sadou registrů pro každý port, různá bázová adresa
- Možnost dvou alternativních funkcí pro periferní zařízení
- Lze generovat přerušení při změně stavu libovolného pinu

# Struktura portu



# Blokové schéma portu (jeden signál)



# Registry pro ovládání portů

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | PIO Enable Register | PIO_PER | Write-only | – |
| 0x0004 | PIO Disable Register | PIO_PDR | Write-only | – |
| 0x0008 | PIO Status Register | PIO_PSR | Read-only | (1) |
| 0x000C | Reserved | | | |
| 0x0010 | Output Enable Register | PIO_OER | Write-only | – |
| 0x0014 | Output Disable Register | PIO_ODR | Write-only | – |
| 0x0018 | Output Status Register | PIO_OSR | Read-only | 0x0000 0000 |
| 0x001C | Reserved | | | |
| 0x0020 | Glitch Input Filter Enable Register | PIO_IFER | Write-only | – |
| 0x0024 | Glitch Input Filter Disable Register | PIO_IFDR | Write-only | – |
| 0x0028 | Glitch Input Filter Status Register | PIO_IFSR | Read-only | 0x0000 0000 |
| 0x002C | Reserved | | | |
| 0x0030 | Set Output Data Register | PIO_SODR | Write-only | – |
| 0x0034 | Clear Output Data Register | PIO_CODR | Write-only | |
| 0x0038 | Output Data Status Register | PIO_ODSR | Read-only or(2) Read-write | – |
| 0x003C | Pin Data Status Register | PIO_PDSR | Read-only | (3) |
| 0x0040 | Interrupt Enable Register | PIO_IER | Write-only | – |
| 0x0044 | Interrupt Disable Register | PIO_IDR | Write-only | – |
| 0x0048 | Interrupt Mask Register | PIO_IMR | Read-only | 0x00000000 |
| 0x004C | Interrupt Status Register(4) | PIO_ISR | Read-only | 0x00000000 |
| 0x0050 | Multi-driver Enable Register | PIO_MDER | Write-only | – |
| 0x0054 | Multi-driver Disable Register | PIO_MDDR | Write-only | – |
| 0x0058 | Multi-driver Status Register | PIO_MDSR | Read-only | 0x00000000 |
| 0x005C | Reserved | | | |
| 0x0060 | Pull-up Disable Register | PIO_PUDR | Write-only | – |
| 0x0064 | Pull-up Enable Register | PIO_PUER | Write-only | – |
| 0x0068 | Pad Pull-up Status Register | PIO_PUSR | Read-only | 0x00000000 |
| 0x006C | Reserved | | | |

# Registry pro ovládání portů

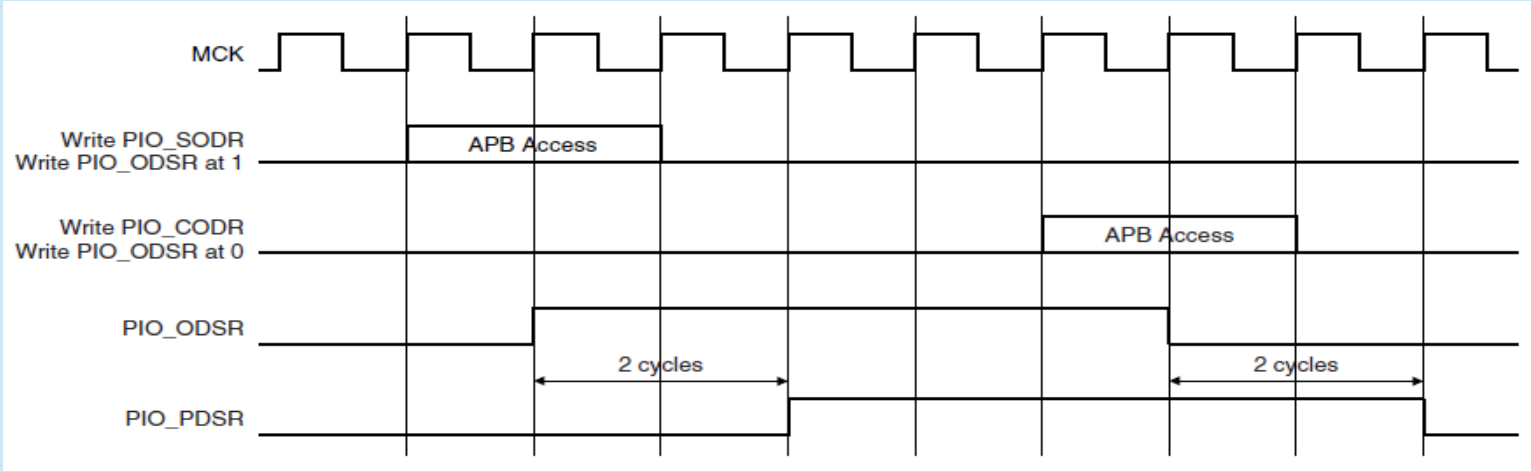
| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0070 | Peripheral A Select Register[5] | PIO_ASR | Write-only | – |
| 0x0074 | Peripheral B Select Register[5] | PIO_BSR | Write-only | – |
| 0x0078 | AB Status Register[5] | PIO_ABSR | Read-only | 0x00000000 |
| 0x007C to 0x009C | Reserved | | | |
| 0x00A0 | Output Write Enable | PIO_OWER | Write-only | – |
| 0x00A4 | Output Write Disable | PIO_OWDR | Write-only | – |
| 0x00A8 | Output Write Status Register | PIO_OWSR | Read-only | 0x00000000 |
| 0x00AC | Reserved | | | |

Notes:
1. Reset value of PIO_PSR depends on the product implementation.
2. PIO_ODSR is Read-only or Read-write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.

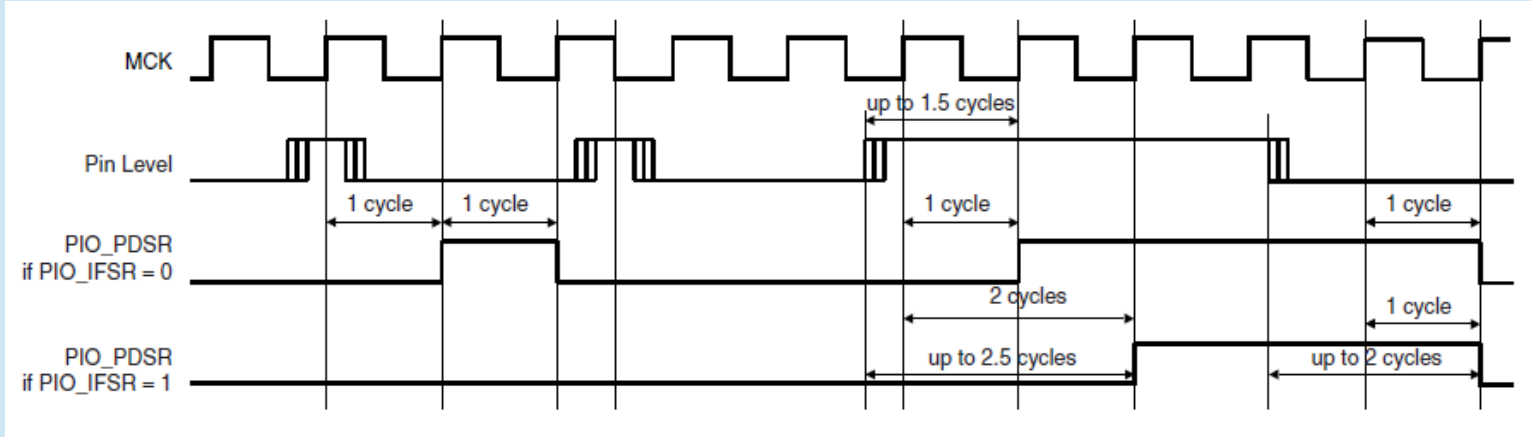
- Podrobný popis registrů v datasheetu AT91SAM9260

# Zpoždění signálů

Výstupy:




Vstupy:



# Podpůrné funkce (knihovny)

- **Výrobce CPU dává k dispozici zdrojové texty (knihovny nebo příklady)**
- **Lze použít přímo funkce pro inicializaci a ovládání periferií, např.:**
  - unsigned char PIO_Configure(const Pin *list, unsigned int size)
  - void PIO_Set(const Pin *pin)
  - Void PIO_Clear(const Pin *pin)
  - char PIO_Get(const Pin *pin)
  - ...
- **Pin je struktura popisující nastavení jednoho nebo více pinů portu**
- **Díky koncepci Set/Reset u většiny řídících registrů je možno nastavení a operace provádět po skupinách pinů jednoho portu**

# Sériová komunikační rozhraní

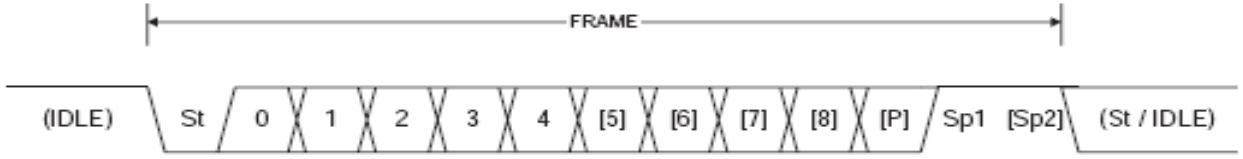
- **Interní**
  - I2C (I2S)
  - SPI
- **Externí**
  - UART (USART)
    - Fyzické rozhraní RS232, RS485, RS422
  - USB
  - CAN BUS
  - ETHERNET

# Princip asynchronního rozhraní U(S)ART

- USART = Universal Synchronous/Asynchronous Receiver Transmitter
- UART – Asynchronní, nepřenáší hodinový signál
- Synchronní varianta musí mít společný synchronní hodinový signál
- Umožňuje plně duplexní spojení
- Přenosová jednotka = 1 Byte
- Umožňuje zabezpečení paritou
- Je možno přenášet navíc 1 bit informace (adresa/data...)

# UART

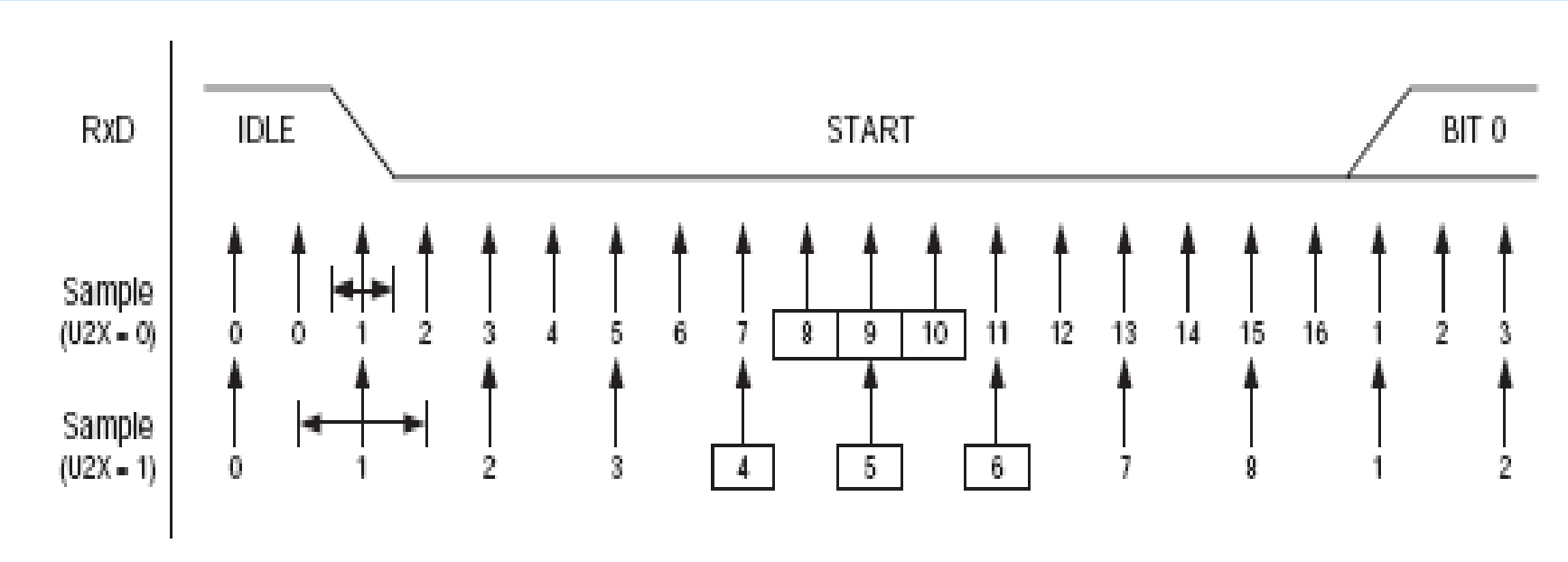
- Volitelný přenos 5-8 bitů
- S paritou sudou, lichou nebo bez parity
- 1-2 STOP bity
- Rychlosti standardní (9600, 19200, 57600, 115200 Bd) nebo libovolné pro pokud není vyžadována kompatibilita
- Vzorkování při příjmu 8x nebo 16x během jednoho bitu
- Přípustná odchylka bitové frekvence vysílače a přijímače cca 2% (resynchronizace po každé přenesené jednotce)




| | |
|---|---|
| St | Start bit, always low. |
| (n) | Data bits (0 to 8). |
| P | Parity bit. Can be odd or even. |
| Sp | Stop bit, always high. |
| IDLE | No transfers on the communication line (RxD or TxD). An IDLE line must be high. |

# UART Synchronizace

- Běžně se používá 16x vyšší interní frekvence než je komunikační rychlost (minimálně 8x)
- Příchozí data jsou průběžně vzorkována a po příchodu začátku START bitu je další kontrola po 6 interních cyklech ve 3 vzorcích
- Pokud není signál stabilní → signalizace chyby

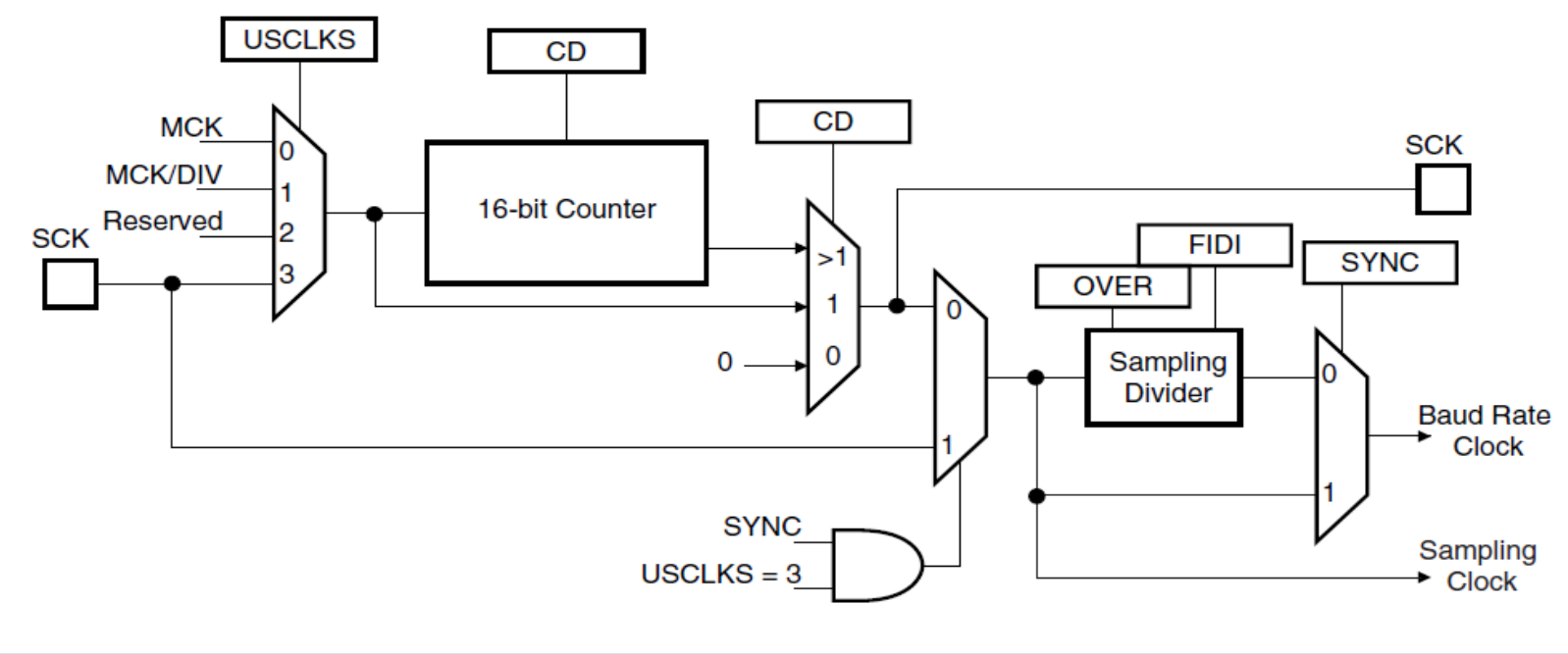


# Fyzické rozhraní UART

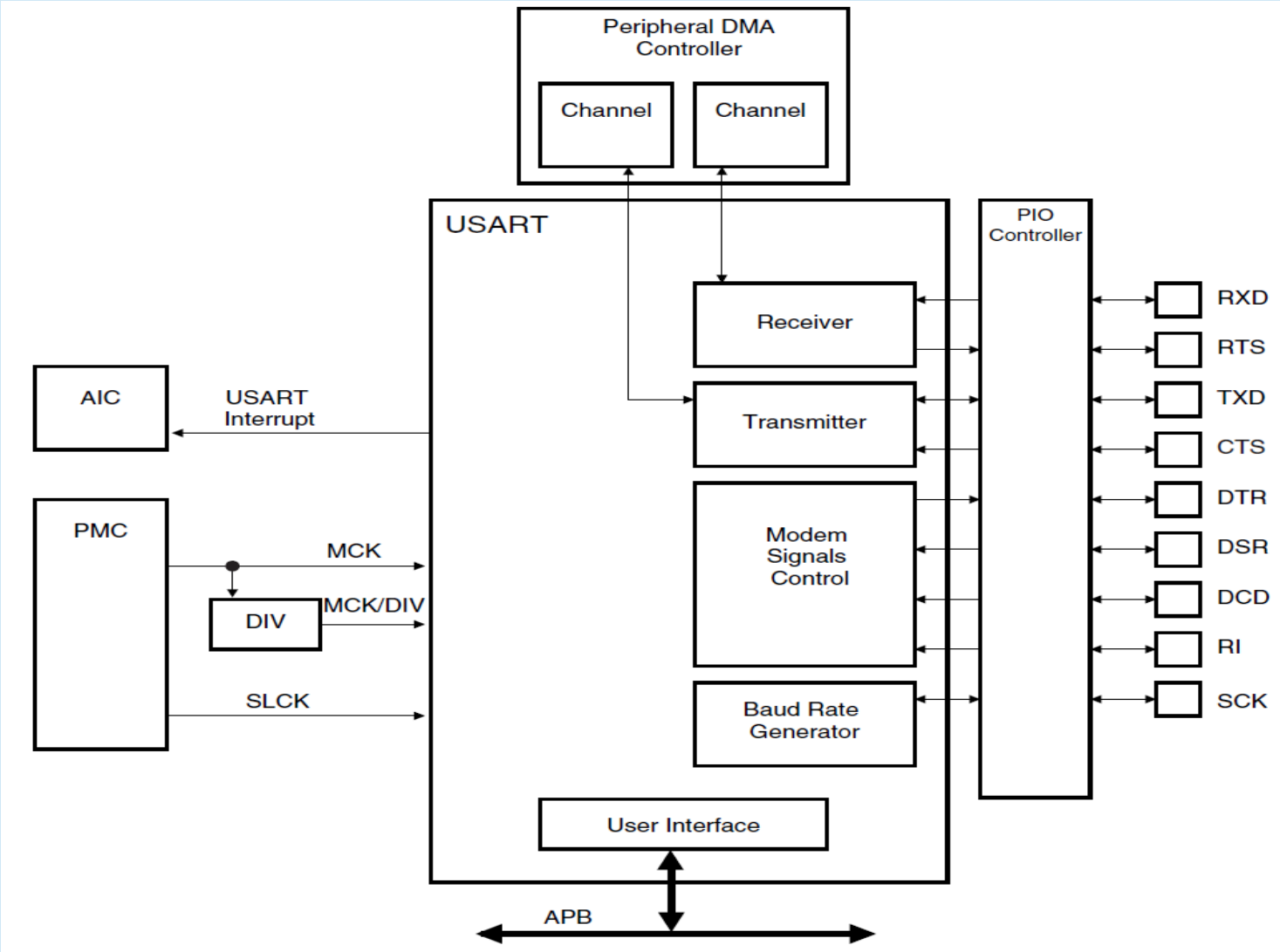
- **TTL – přímo úrovně 5 nebo 3,3V, pro interní komunikaci a přímo připojené moduly, smart karty... Plně duplexní nebo poloduplexní (OC výstup), klidová úroveň H**
- **RS-232 – napěťové úrovně H=<-3..-12V>, L=<+3..+12V>, plně duplexní rozhraní, speciální IO obsahující nábojové pumpy z 5V (3,3V)**
- **RS422 – symetrické vedení (kroucený pár impedančně přizpůsobený), diferenciální signál <150mV .. 5V>, plně duplexní (čtyřvodičové propojení)**
- **RS485 - symetrické vedení (kroucený pár impedančně přizpůsobený), diferenciální signál <150mV .. 5V>, polo-duplexní (dvouvodičové propojení), umožňuje připojení více zařízení na sběrnici, multimaster architektura je problematická z důvodu nemožnosti detekce kolize**
- **IRDA – optický přenos volným prostorem na krátké vzdálenosti v infračerveném spektru**

# UART v AT91SAM9260

- 4xUSART + 2xUART (jen RxD,TxD) + DBG UART (jen RxD,TxD)
- Možnost přenosu dat prostřednictvím DMA
- Vlastní generátor hodinového signálu pro každý kanál
  Baud rate = MCK / (8 * (2 - OVER) * CD)
  CD... dělící poměr generátoru

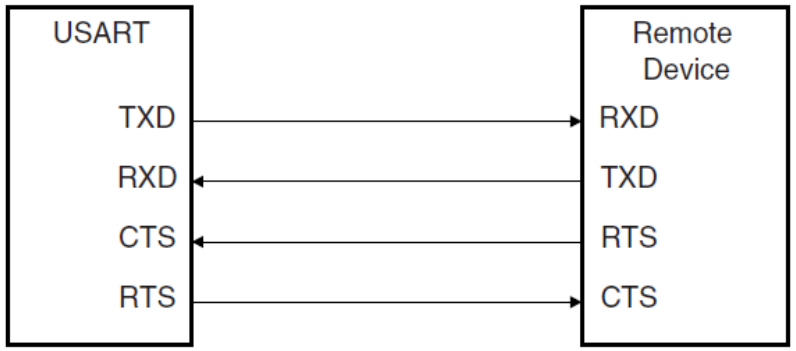


# Blokové schéma USART

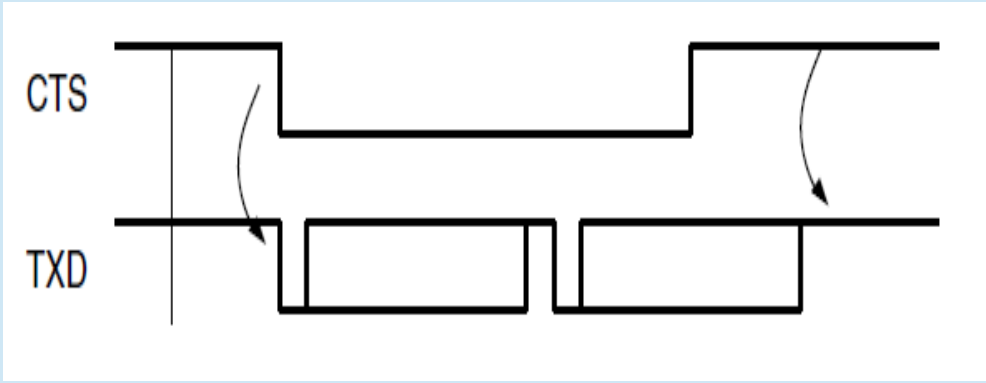


# Možnost hardwarového řízení přenosu

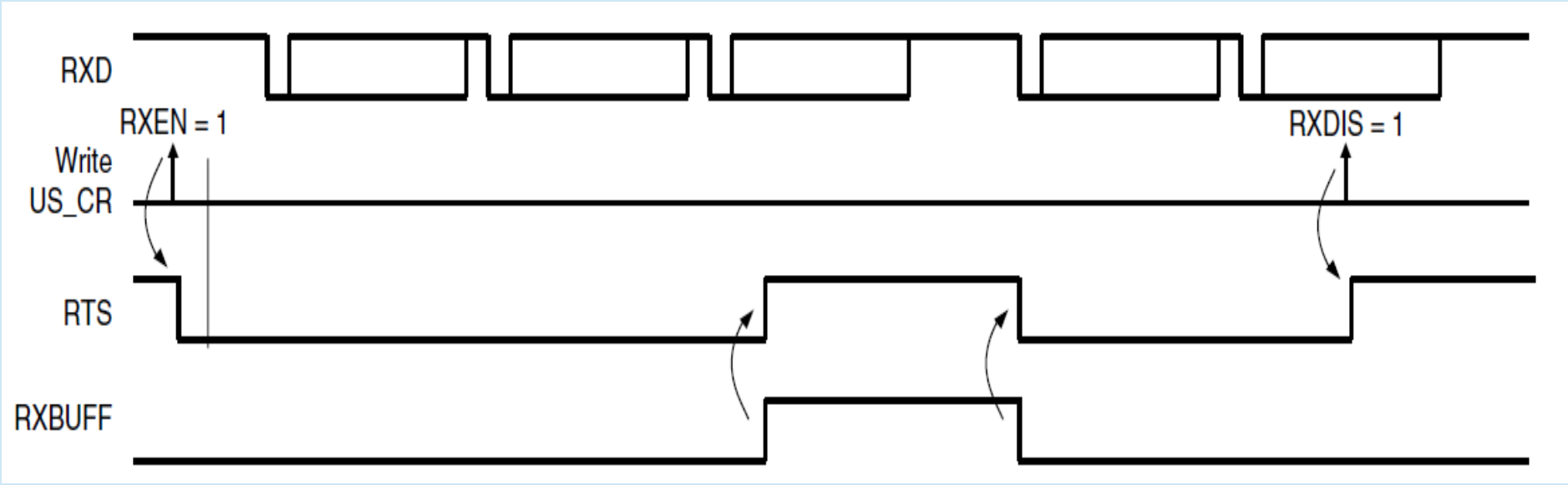
Propojení signálů modemu



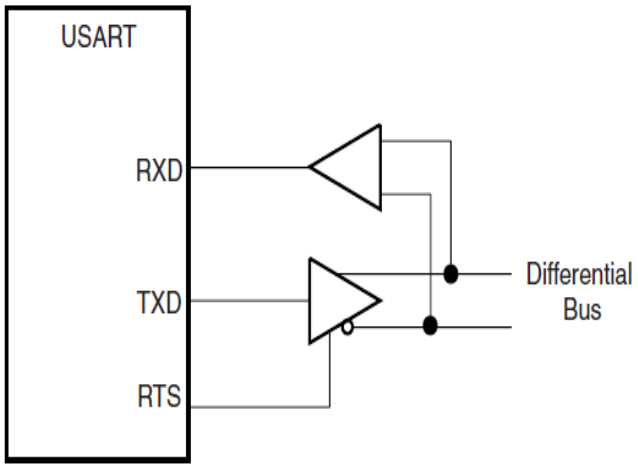

Vysílání

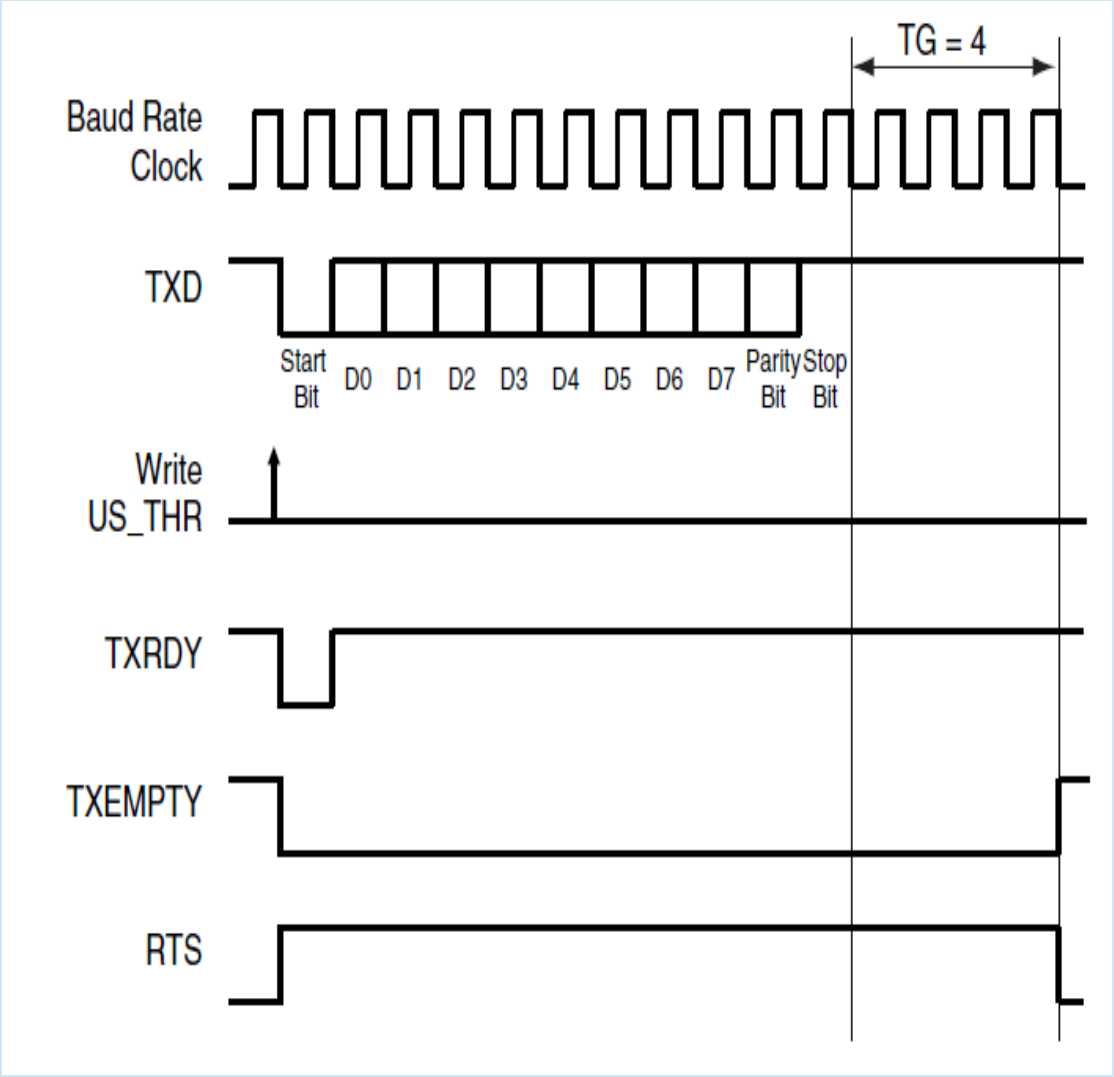



Příjem



# RS-485 řízení budiče



# Mapa registrů pro každý USART

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | Control Register | US_CR | Write-only | – |
| 0x0004 | Mode Register | US_MR | Read-write | – |
| 0x0008 | Interrupt Enable Register | US_IER | Write-only | – |
| 0x000C | Interrupt Disable Register | US_IDR | Write-only | – |
| 0x0010 | Interrupt Mask Register | US_IMR | Read-only | 0x0 |
| 0x0014 | Channel Status Register | US_CSR | Read-only | – |
| 0x0018 | Receiver Holding Register | US_RHR | Read-only | 0x0 |
| 0x001C | Transmitter Holding Register | US_THR | Write-only | – |
| 0x0020 | Baud Rate Generator Register | US_BRGR | Read-write | 0x0 |
| 0x0024 | Receiver Time-out Register | US_RTOR | Read-write | 0x0 |
| 0x0028 | Transmitter Timeguard Register | US_TTGR | Read-write | 0x0 |
| 0x2C - 0x3C | Reserved | – | – | – |
| 0x0040 | FI DI Ratio Register | US_FIDI | Read-write | 0x174 |
| 0x0044 | Number of Errors Register | US_NER | Read-only | – |
| 0x0048 | Reserved | – | – | – |
| 0x004C | IrDA Filter Register | US_IF | Read-write | 0x0 |
| 0x5C - 0xFC | Reserved | – | – | – |
| 0x100 - 0x128 | Reserved for PDC Registers | – | – | – |

# Príklad detailného popisu registru

**32.7.1 USART Control Register**

**Name:** US_CR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RTSDIS | RTSEN | DTRDIS | DTREN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RETTO | RSTNACK | RSTIT | SENDA | STTTO | STPBRK | STTBRK | RSTSTA |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDIS | TXEN | RXDIS | RXEN | RSTTX | RSTRX | – | – |

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

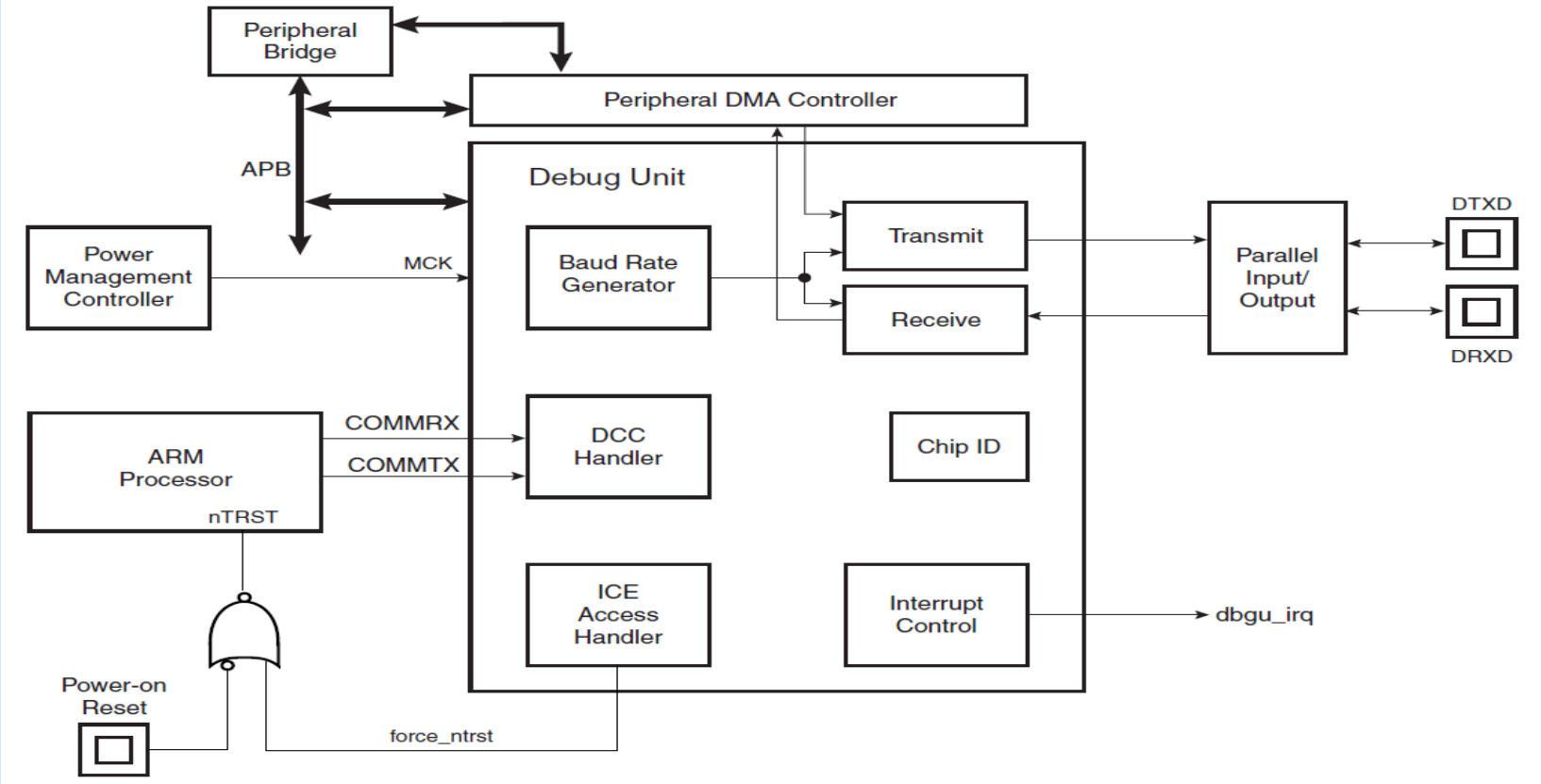
- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

# Rozhraní DBGU

- Zjednodušené rozhraní UART, pouze RxD, TxD
- Používáno BOOT loaderem, možno použít v programu
- V přípravku připojeno na kanál B rozhraní FTDI



# Registry DBGU (DS:28.5)

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | Control Register | DBGU_CR | Write-only | – |
| 0x0004 | Mode Register | DBGU_MR | Read-write | 0x0 |
| 0x0008 | Interrupt Enable Register | DBGU_IER | Write-only | – |
| 0x000C | Interrupt Disable Register | DBGU_IDR | Write-only | – |
| 0x0010 | Interrupt Mask Register | DBGU_IMR | Read-only | 0x0 |
| 0x0014 | Status Register | DBGU_SR | Read-only | – |
| 0x0018 | Receive Holding Register | DBGU_RHR | Read-only | 0x0 |
| 0x001C | Transmit Holding Register | DBGU_THR | Write-only | – |
| 0x0020 | Baud Rate Generator Register | DBGU_BRGR | Read-write | 0x0 |
| 0x0024 - 0x003C | Reserved | – | – | – |
| 0x0040 | Chip ID Register | DBGU_CIDR | Read-only | – |
| 0x0044 | Chip ID Extension Register | DBGU_EXID | Read-only | – |
| 0x0048 | Force NTRST Register | DBGU_FNR | Read-write | 0x0 |
| 0x004C - 0x00FC | Reserved | – | – | – |
| 0x0100 - 0x0124 | PDC Area | – | – | – |

# Programová podpora UART (knihovna)

- Ke stažení soubor knihoven, příkladů a dokumentace
  http://www.atmel.com/tools/SAM9260-EK.aspx

Část souboru uart.h:

```
//------------------------------------------------------------------------------
//         Exported functions
//------------------------------------------------------------------------------

extern void USART_Configure(
    AT91S_USART *usart,
    unsigned int mode,
    unsigned int baudrate,
    unsigned int masterClock);

extern void USART_SetTransmitterEnabled(AT91S_USART *usart,
                        unsigned char enabled);

extern void USART_SetReceiverEnabled(AT91S_USART *usart,
                        unsigned char enabled);

extern void USART_Write(
    AT91S_USART *usart,
    unsigned short data,
    volatile unsigned int timeOut);

extern unsigned char USART_WriteBuffer(
    AT91S_USART *usart,
    void *buffer,
    unsigned int size);

extern unsigned short USART_Read(
    AT91S_USART *usart,
    volatile unsigned int timeOut);

extern unsigned char USART_ReadBuffer(
    AT91S_USART *usart,
    void *buffer,
    unsigned int size);

extern unsigned char USART_IsDataAvailable(AT91S_USART *usart);

extern void USART_SetIrdaFilter(AT91S_USART *pUsart, unsigned char filter);
```

# Programová podpora DBGU

Soubor dbgu.h

```
//------------------------------------------------------------------------------
//         Global functions
//------------------------------------------------------------------------------

extern void DBGU_Configure(
    unsigned int mode,
    unsigned int baudrate,
    unsigned int mck);

extern unsigned char DBGU_GetChar(void);

extern void DBGU_PutChar(unsigned char c);

extern unsigned int DBGU_IsRxReady(void);
```

Inicializace a použití

```
#define TRACE_CONFIGURE(mode, baudrate, mck) { \
    const Pin pinsDbgu[] = {PINS_DBGU}; \
    PIO_Configure(pinsDbgu, PIO_LISTSIZE(pinsDbgu)); \
    DBGU_Configure(mode, baudrate, mck); \
    }
……
```

```
TRACE_CONFIGURE(DBGU_STANDARD, 115200, BOARD_MCK);        //inicializace v main.c
int _read(int file, char *ptr, int len){                    //čtení znaků v syslib.c
int characters=0;
while (!DBGU_IsRxReady());
while (DBGU_IsRxReady() && len != 0)
    {
        *ptr = (char)DBGU_GetChar();
        len --;
        characters++;
ptr++;
    }
```