Principy sériové komunikace, sériová komunikační rozhraní

Mikroprocesorové a vestavěné systémy (IMP)

Přednáší: Michal Bidlo

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Obsah přednášky

- Principy sériové komunikace
  - Synchronní přenos
  - Asynchronní přenos
- Asynchronní komunikační rozhraní UART
- Synchronní komunikační rozhraní
  - SPI
  - I2C
- Vzorové aplikace

Základní princip sériové komunikace

Při sériovém přenosu dat jsou jednotlivé bity přenášeny po jediném vodiči postupně (v čase) jeden za druhým,

zatímco u paralelního přenosu je v jediném časovém okamžiku k dispozici více bitů (pochopitelně každý bit na „svém“ vodiči).

Jestliže bity informace se objevují na jediném datovém vodiči postupně, je třeba jednoznačně určit, v kterém okamžiku je na datovém vodiči hodnota kterého bitu.

Možnosti realizace sériové komunikace

Existují dva základní přístupy:

Synchronní sériový přenos,

kdy spolu s daty je přenášen i hodinový signál. Jeho hrany určují buď kdy se objevuje další bit na datovém vodiči či kdy je možné bezpečně sejmout hodnotu bitu na datovém vodiči.

Asynchronní sériový přenos,

kdy hodinový signál není přenášen, ale přijímač si jej generuje sám.

Zde je třeba zajistit dostatečnou přesnost generátoru hodinového signálu v přijímači a prostředky, kterými je možné generátor hodinového signálu přijímače synchronizovat s generátorem vysílače.

Synchronizace přenosu dat

Synchronní přenos:

V celé soustavě přijímač-vysílač je zajištěn jednotný zdroj hodinového signálu,

za cenu jednoduché synchronizace se však ztrácí elegance přenosu „po jednom vodiči“ (potřebujeme kromě datového ještě hodinový vodič).

Hodinový signál nemusí být nutně periodický.

Synchronizace přenosu dat

Asynchronní přenos:

Periodicita hodinového signálu, kterým vysílač určuje, kdy na datový vodič nastaví hodnotu dalšího bitu, je velmi žádoucí. Přijímač totiž musí svůj generátor hodinových pulsů nastavit tak, aby běžel co nejvíce ve shodě s generátorem vysílače, který určuje časové intervaly pro jednotlivé bity.

Je tedy třeba zajistit, aby oba generátory běžely na stejné frekvenci. Pak už zbývá jen, aby měly oba stejnou fázi.

Synchronizace fáze obou generátorů se provádí zpravidla nějakou předem dohodnutou změnou úrovně na datovém vodiči (dohodnutou, tedy nenesoucí žádnou informaci)

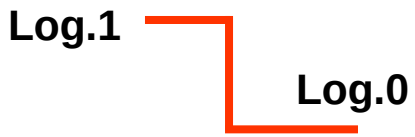
Pokud se synchronizace provádí „vhodně“ často, nemusí být nároky na stálost a přesnost obou generátorů příliš vysoké. Důležité je, aby se za dobu mezi dvěma synchronizacemi oba generátory vzájemně nerozešly o více než půlku doby jednoho bitového intervalu.

Asynchronní rozhraní UART

Asynchronní komunikace je nejčastěji realizována jako zařízení (HW) nazývané UART (Universal Asynchronous Receiver-Transmitter). Různé standardy navíc specifikují podobu konektorů, napěťové úrovně, způsob kódování dat atd., např. RS-232 (COM port).

Problém synchronizace hodin obou komunikujících stran je řešen přechodem z klidové úrovně do opačné, což je zároveň signálem od vysílače zahajující komunikaci. Tato událost musí nastat vždy bez ohledu na hodnotu prvního přenášeného bitu a nazývá se start bit.

Uvažujeme-li, že klidový stav na datovém vodiči je log. „1“, pak komunikace začíná start bitem po sestupné hraně a přijímač „vidí“ tento režijní interval jako log. „0“. (**Nejedná se však o data!**)

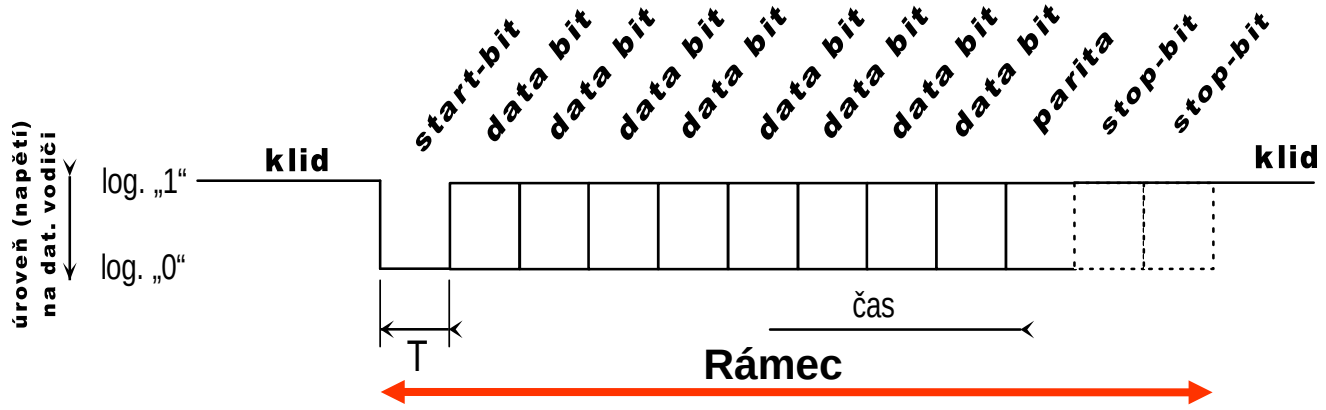


Datový rámec (frame) UART

První informační (datový) bit se na datovém vodiči objevuje až po nějaké době od počáteční hrany.

Tato doba je pochopitelně přesně dána, zpravidla to bývá jeden bitový interval – jeden takt generátoru hodin vysílače (a jeden takt generátoru hodin přijímače, který je nyní s generátorem hodin vysílače zcela čerstvě zasynchronizován).

Tento bitový interval, který je vyplněn hodnotou opačnou, nežli je klidová hodnota a má význam hlavně kvůli hraně, kterou na svém začátku způsobí, se nazývá **start bit**.



Stop bity

Za posledním vyslaným datovým bitem a případným paritním bitem se vždy musí vyslat alespoň jeden tzv. **stop bit**, který má vždy hodnotu klidového stavu (zde log.1). Stop bit slouží k vzájemnému oddělení přenášených slov. Lze zpravidla nastavit počet stop bitů v rozsahu jeden, či dva stop bity.

Stop bity, kromě toho, že zajistí na datovém vodiči klidový stav, z něhož lze hranou přejít do start bitu následujícího slova, slouží také k tomu, aby měl přijímač čas odeslat právě přijaté slovo ze vstupního posuvného registru dále ke zpracování.

*Klidová hodnota linky se někdy literatuře označuje jako „**space**“ (mezera), opačná hodnota, aktivní, se označuje jako „**mark**“ (značka).*

Trvání přenosu rámce

Nechť ***n*** je celkový počet bitů rámce a ***T*** je perioda synchronizačního signálu. Pak doba, za kterou je rámec přenesen, je ***tr*** = ***n*** * ***T***.

n je vyjádřeno ve tvaru:

n = 1 + ***D*** + ***P*** + ***S***,

kde „1“ na začátku vyjadřuje jediný start bit,
D je počet datových bitů,
P je počet paritních bitů (obvykle 0 nebo 1),
S je počet stop bitů.

Přenosová rychlost

Přenosová rychlost se nejčastěji udává v baudech (Bd) což v případě binárního kódování vyjadřuje „bitů za sekundu“.

Protože u popsaného přenosu je pouze dvoustavová modulace, v každé periodě ***T*** lze přenést pouze informaci o velikosti 1 bit – buďto stav log. 0 nebo log. 1. Stav datového vodiče se uprostřed tohoto intervalu nesmí změnit, změny probíhají jen na hranicích. Z toho vychází, že 1 Bd = 1 bit/sec a přenosová rychlost tak odpovídá frekvenci hodinového signálu.

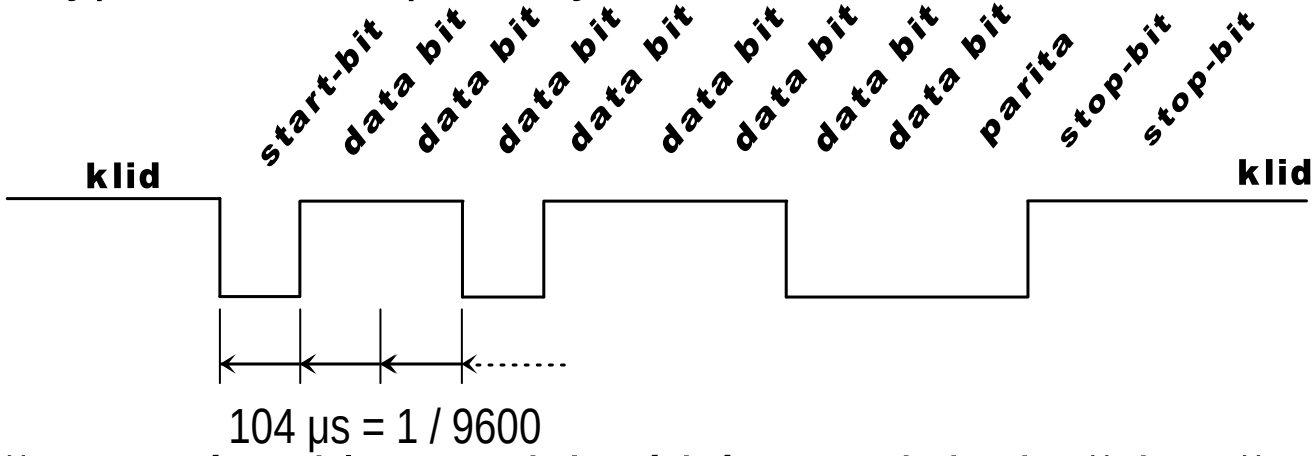
Při určité přenosové rychlosti bude však skutečný počet přenášených datových bitů (tzv. efektivní přenosová rychlost) nižší, což je to dáno tím, že rámec obsahuje několik režijních bitů (start bit, případný paritní bit a stop bity), které nenesou žádná data.

Je-li n = 1+8+parita+stop_bity je počet bitů rámce a N počet bytů přenášených dat, pak doba přenosu přenosovou rychlostí BAUD je

TP = N * n / BAUD [s]

Příklad asynchronního přenosu dat

Nechť přenášený bajt má hodnotu 0x3B, tedy binárně 00111011. Přenos bude doplněn lichou paritou a dvěma stop bity. Jak bude vypadat rámec pro 1 byte?



Přenosová rychlost nechť má být 9600 bd a je třeba přenést 1024 bytů dat. Jak dlouho přenos potrvá?

TP = 1024 * 12 / 9600 = 1,28 s

Jaká bude efektivní přenosová rychlost?

UART na MCU KL05Z

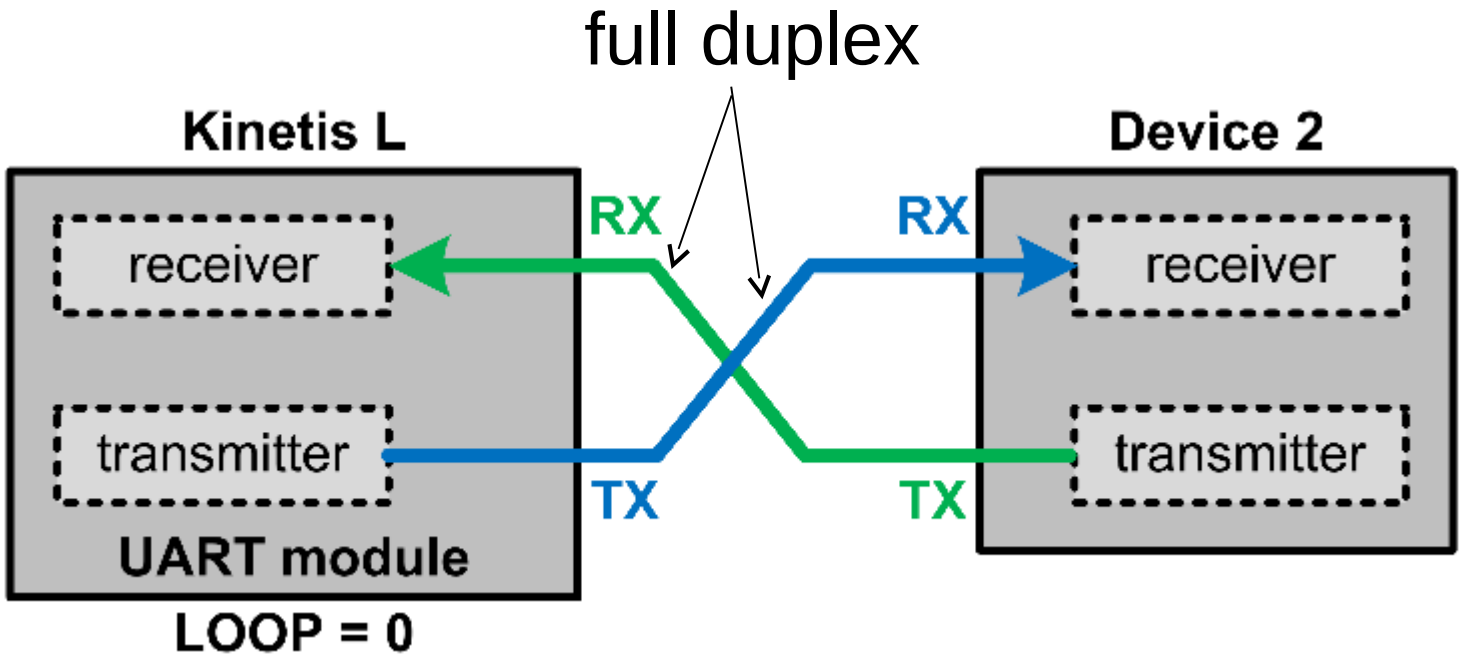
V této části se budeme zabývat popisem a programováním modulu UART v našem MCU MKL05Z32VFM4.

Základní vlastnosti:

- full duplex, samostatný receiver + transmitter,
- programovatelná rychlost přenosu dat,
- volitelný režim poling, interrupt či DMA,
- možnost zabezpečení paritou,
- programovatelná šířka slova (8, 9 nebo 10 bitů),
- nízkopříkonový režim, aktivace idle-line, address-mark nebo address-match.

Blokové schéma modulu UART

…na MCU MKL05Z32VFM4 (FREEDOM BOARD, laboratorní kit)



Registry modulu UART

Registry představují základní programátorské rozhraní k obsluze periferií MCU.

V případě modulu UART jsou to:

- Datový registr UART0_D
- Baud rate registry UART0_BDH, UART0_BDL
- Řídicí registry UART0_C1 (2, 3,…)
- Stavové registry UART0_S1 (2,…)
- Match address reg. UART0_MA1

Podrobný popis registrů a funkcionality UART viz Referenční manuál MCU KL05Z dostupný v modulu e-learning v sekci Lab. cvičení od str. 635

Schéma obsluhy rozhraní UART

Před zahájením přenosu je zpravidla nutno provést konfiguraci modulu (nastavení přenosové rychlosti a řídicích registrů a jeho aktivaci).

UART Ctrl. Reg. (UART0_C2)

Bit	7	6	5	4	3	2	1	0
Read / Write	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Obvyklý postup:

- deaktivace UART (UART0SRC=00 v reg. SIM_SOPT2, TE=0, RE=0 v UART0_C2 – bity Transmitter / Receiver Enable)
- konfigurace přenosové rychlosti (zdroj hodin, dělič taktu,…)
- nastavení řídicích registrů (přerušení, šířka slova, parita,…)
- aktivace UART (TE=1 / RE=1)
- realizace přenosu (test stavu UART, zápis/čtení dat v režimu polling či v obsluze přerušení, ošetření výjimek apod.)
- ukončení, deaktivace UART

Specifikace přenosové rychlosti (baud rate)

Volba zdroje hodin pro UART: System Options Register 2 (SIM_SOPT2) [UART0SRC – bity 27-26];
více k systému časování MCU viz reference manual od str. 105.

00	Clock disabled
01	MCGFLLCLK clock
10	OSCERCLK clock
11	MCGIRCLK clock

Jednotlivé zdroje hodinového signálu se liší zejména způsoben generování (např. obyčejný krystal či pokročilý MCG) a frekvencí.

Příklad nastavení hodnoty děliče pro UART: Mějme zdroj MCGFLLCLK na 48 MHz a požadovanou přenosovou rychlost baud_rate=115200bd. Jaká bude hodnota děliče v reg. UART0_BD, pokud oversampling=4?

Řešení:

$$\mathrm{SBR} = \frac{\text{UART0 source clock frequency}}{\text{baud rate}*(\text{UART0\_}C4[\text{OSR}]+1)} = \frac{48000000}{115200*(3+1)} = 104 \quad = 0x68$$

Specifikace přenosové rychlosti (baud rate)

V registrech UART0_BD pak nastavení SBR vypadá následovně:

UART0_BDH

7	6	5	4	3	2	1	0
LBKDIE	RXEDGIE	SBNS	0	0	SBR	0	0

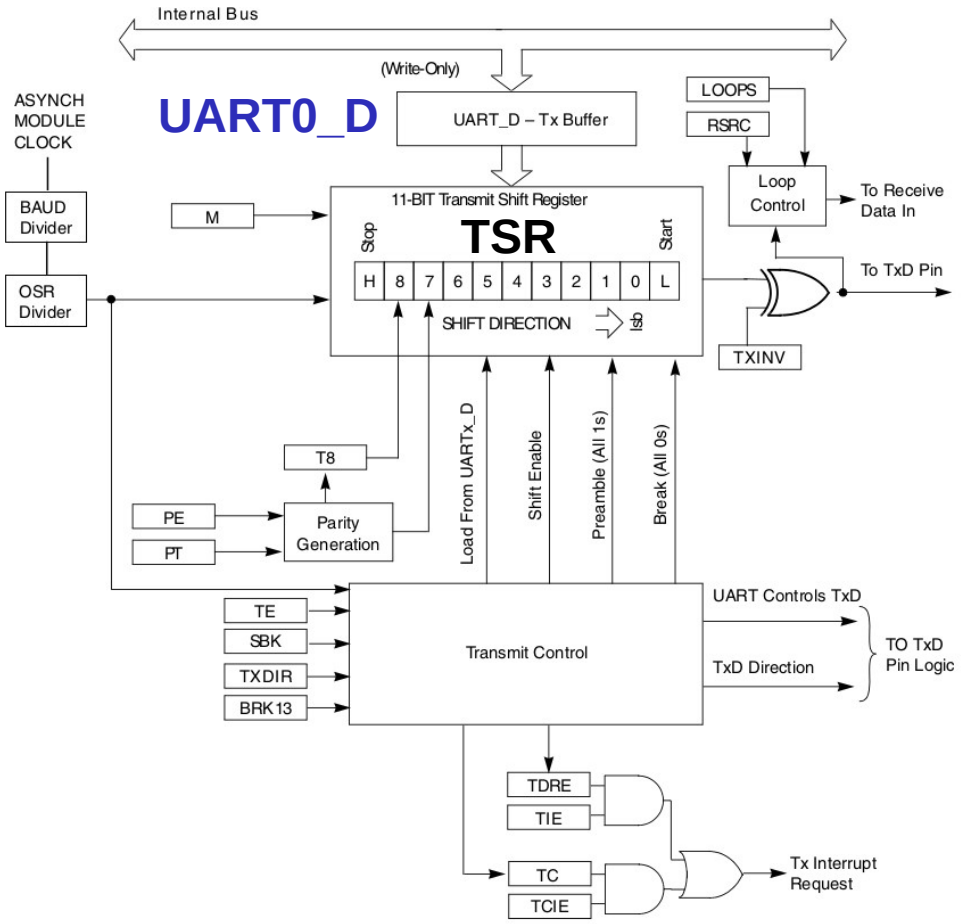
UART0_BDL

7	6	5	4	3	2	1	0
0	1	1	0	SBR 1	0	0	0

UART transmitter (vysílač)

Vysílač i přijímač UART jsou realizovány jako nezávislé moduly, které jsou obsluhovány pomocí dříve zmíněné soustavy registrů.

Hlavními prvky zde jsou časovací obvody, posuvné registry pro serializaci/deserializaci dat a řídicí logika.



Odeslání znaku

Pokud je modul UART správně nastaven, data se vyšlou jednoduše tak, že se programově zapíší do datového registru (**UART0_D**). Vše ostatní zařídí už vysílač sám – zkompletuje rámec a postupně jej odvysílá na výstup TxD prostřednictvím **posuvného registru** se specifikovaným taktem generátoru hodinového signálu.

Před zápisem dat je nutné otestovat, zda je datový registr UART0_D prázdný. Pokud bychom provedli zápis do UART0_D dříve, než byl do posuvného registru přenesen předchozí znak, došlo by k jeho přepsání a tím ke ztrátě dat.

Stav UART0_D indikuje bit **TDRE** – Transmitter Data Register Empty ve stavovém registru **UART0_S1**, kdy **TDRE==1** indikuje, že do datového registru **lze zapsat**.

Bit	7	6	5	4	3	2	1	0
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write				w1c	w1c	w1c	w1c	w1c
Reset	1	1	0	0	0	0	0	0

UART Status Register 1 (UART0_S1)

Odeslání znaku – poznámka k realizaci v MCU

Po zapsání znaku do registru UART0_D je znak přenesen do TSR (Transmit Shift Register), ze kterého probíhá vysílání, a bezprostředně po naplnění TSR je znovu nastaven bit TDRE, což umožňuje zápis dalších dat do UART0_D.

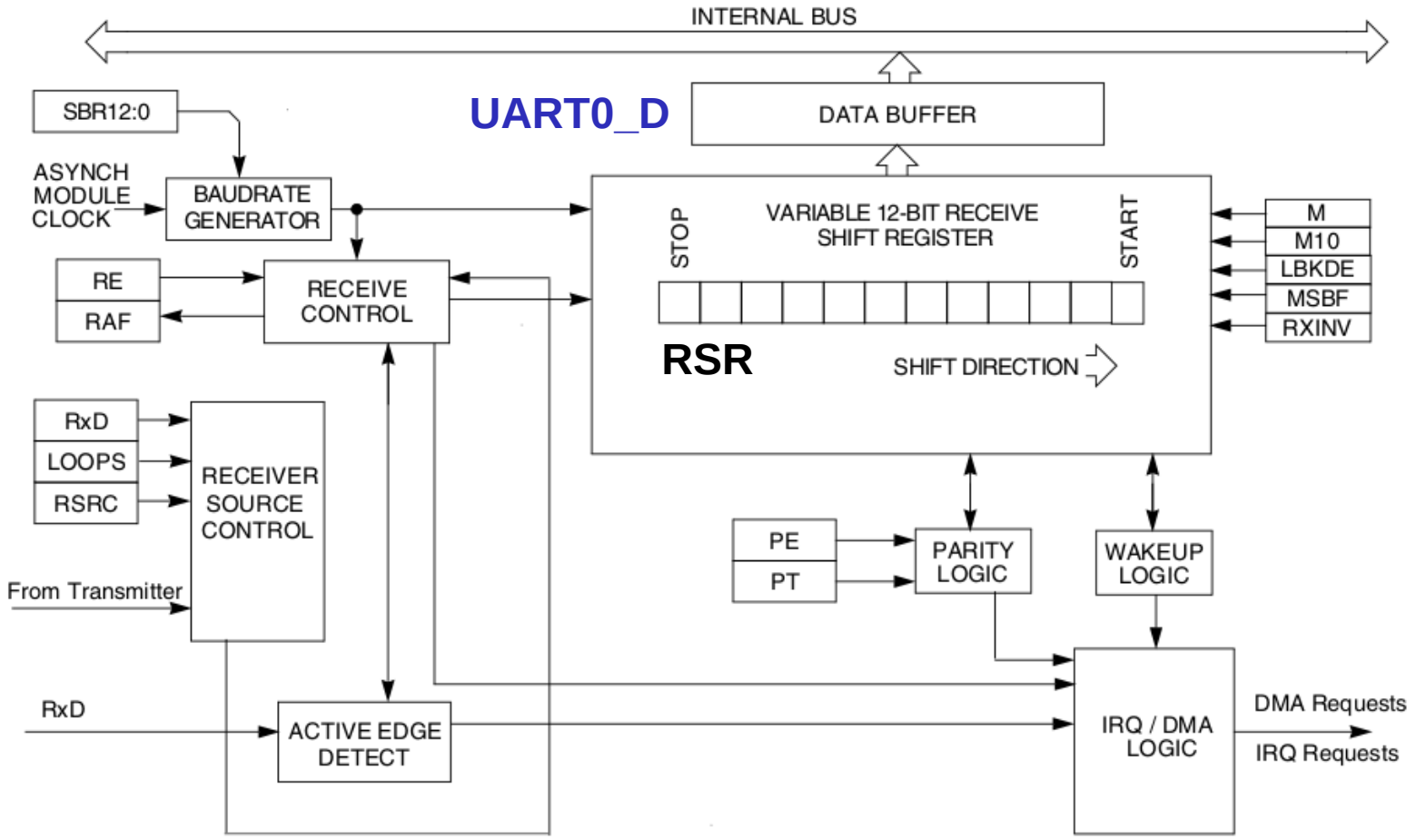
Dokončení vysílání je možné sledovat stavem bitu **TC** (Transmission Complete) v registru UART0_S1, který hodnotou log. 1 indikuje, že data byla odeslána. V tomto okamžiku může proběhnout přesun dalších dat z UART0_D do TSR k odeslání (to se děje zcela automaticky, bez zásahu programátora, jsou-li v UART0_D nějaká data připravena).

Bity TDRE a TC jsou si významem podobné, rozdíl spočívá v tom, že bit TDRE signalizuje prázdný datový registr, ale neříká nic o tom, zda už celý znak, který byl z datového registru přesunut do výstupního posuvného registru, byl odvysílán. TC navíc signalizuje, že data už zcela opustila i posuvný registr vysílače.

Bit	7	6	5	4	3	2	1	0
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write				w1c	w1c	w1c	w1c	w1c
Reset	1	1	0	0	0	0	0	0

UART Status Register 1 (UART0_S1)

UART receiver (přijímač)



Příjem znaku

Příjem znaku automaticky zajišťují obvody přijímače, které řídí proces příjmu, vyhodnocují jeho průběh a přijatý znak připraví do datového registru přijímače.

Okamžik dokončení příjmu dat je indikováno bitem **RDRF** – Receiver Data Register Full) stavového registru UART0_S1. Před čtením z UART0_D přijímače je tedy nutné otestovat, zda je bit RDRF ve stavu log. 1 (data přijata). Teprve poté má smysl přijatá data číst.

Bit	7	6	5	4	3	2	1	0
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write				w1c	w1c	w1c	w1c	w1c
Reset	1	1	0	0	0	0	0	0

UART Status Register 1 (UART0_S1)

Příjem znaku – poznámka k realizaci v MCU

V průběhu příjmu dat jsou jednotlivé bity datového rámce ukládány do posuvného registru RSR (Receive Shift Register) a teprve po dokončení příjmu všech bitů (včetně režijních) jsou data vystavena do registru UART0_D a je nastaven příznak RDRF.

Přijímaná data jsou vzorkována s vyšší frekvencí, než je frekvence generovaná podle nastavení baud rate registru (tzv. **oversampling**). To umožňuje detekci šumu během přenosu, jenž je signalizován, kdykoliv se log. úrovně některých dvou vzorků přijímaného bitu liší (příznak **NF** – Noise Flag v registru UART0_S1).

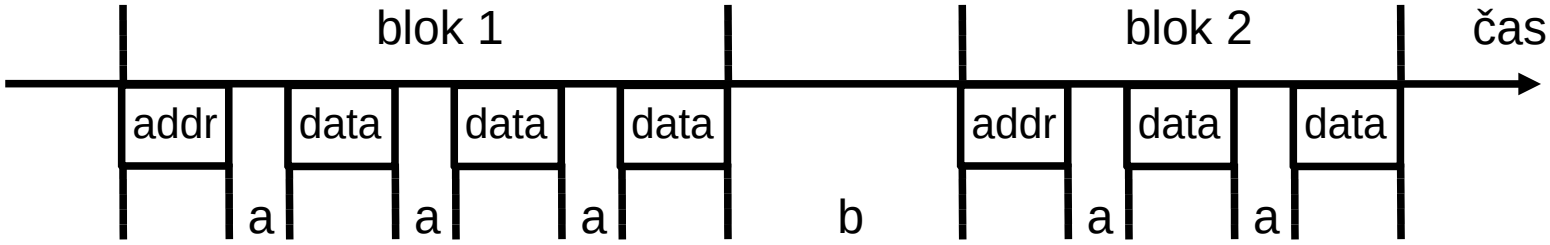
Sběrnice UART, optimalizace spotřeby energie

Efektivní způsob snižování spotřeby energie – komponenty systému, které aktuálně „nemají co dělat“, mohou být v *low-power* módu. Modul UART v KL05 podporuje automatický *wake-up*, je-li MCU adresován pro komunikaci. Možnosti:

- Aktivace na základě shody adresy (Address Match Wake-up): modul UART poskytuje adresové registry (UART0_MAx – Match Address), jejichž obsah je automaticky porovnáván s přijatou hodnotou a v případě shody je zahájena komunikace.

Sběrnice UART, optimalizace spotřeby energie

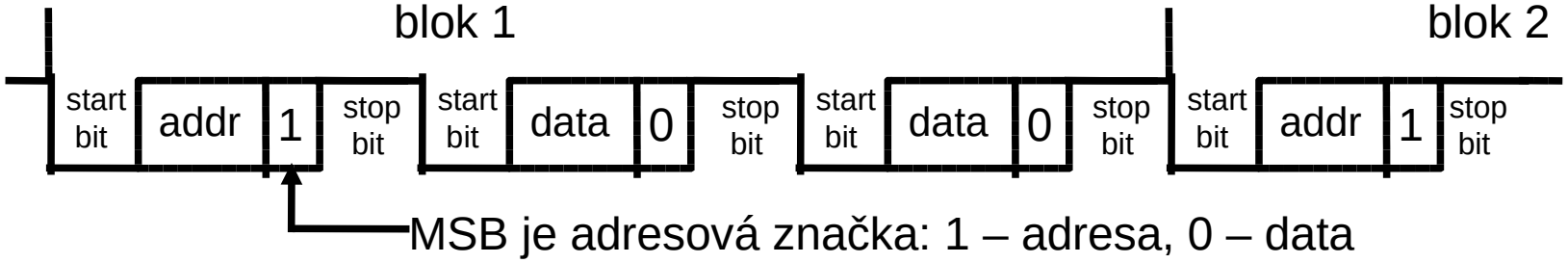
- Aktivace na základě prodlevy (Idle-Line Wake-up)



Přijímač je uspán a čeká na událost *idle.* Ta jej aktivuje a následně přijatý znak je interpretován jako adresa, kterou je třeba porovnat s adresou přijímače. Pokud se shoduje, jsou další znaky (data) adresována tomuto přijímači. V opačném případě je přijímač opět uspán a čeká na další *idle*.

Sběrnice UART, optimalizace spotřeby energie

- Aktivace na základě adresové značky (Address-Mark Wake-up)



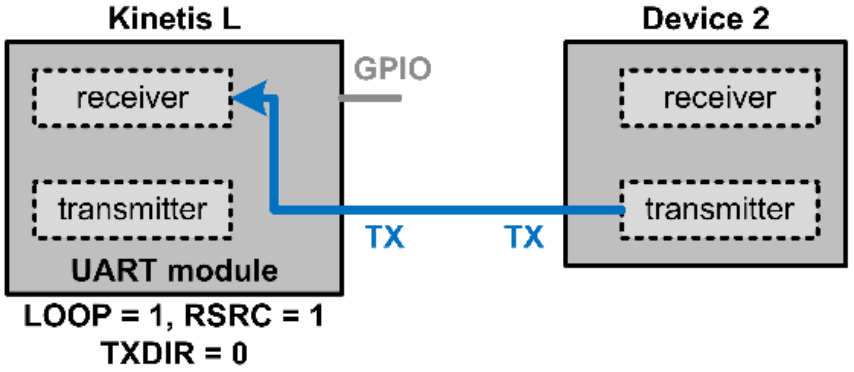
Přijímač je uspán a čeká na příjem znaku, jehož MSB je v log. 1 (adresa). Tento znak jej aktivuje, je nutné porovnat právě přijatou adresu s adresou přijímače a pokud se shodují, další data jsou určena tomuto přijímači. V opačném případě je přijímač opět uspán a čeká na další znak s příznakem adresy.

Jednovodičový (Single Wire) UART

Nejjednodušší asynchronní rozhraní využívající pro komunikaci pouze jediný vodič – half-duplex UART.

Vhodné v případech, kdy má smysl přenášet data pouze jedním směrem (např. čtení teploty ze senzoru), případně potřebujeme-li ušetřit jeden vodič (musí však stačit half-duplex!)

Schéma pro Kinetis receiver:



Konfigurace:

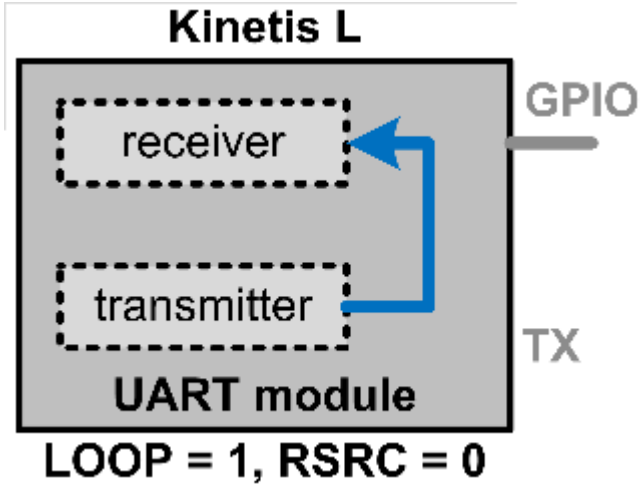
UART0_C1 [RSRC=1, LOOPS=1],

pak UART0_C3 [bit TXDIR] umožňuje volit směr toku dat (0 – příjem, 1 – vysílání).

UART v režimu LOOP

Pro potřeby testování a ladění dovoluje Kinetis vnitřně realizovat spojení vysílač → příjímač, takže funkčnost komunikace lze ověřit bez nutnosti připojení dalšího zařízení.

Konfigurace LOOP režimu: UART0_C1 [RSRC=0, LOOPS=1].



Podrobnosti viz vzorová app.

Ukázka aplikace UART (LOOP mode)

Hlavní části: inicializace, aktivace

```
/* Inicializace UART - nastaveni prenosove rychlosti 115200Bd, 8 bitu, bez parity */
void UART0Init(void) {
     // stop transmitter and receiver before setting up
    UART0->C2 &= ~(UART0_C2_TE_MASK | UART0_C2_RE_MASK);

    UART0->BDH = 0x00;
    UART0->BDL = 0x1A;         // Baud rate 115 200 Bd, 1 stop bit
    UART0->C4 = 0x0F;         // Oversampling ratio 16, match address mode disabled

    UART0->C1 = 0x80;         // 8 data bitu, bez parity, LOOP mode enabled
    UART0->C2 |= ( UART0_C2_TE_MASK | UART0_C2_RE_MASK );    // Zapnout vysilac i prijimac
}
```

vysílání

```
void SendCh(char ch)  {
    while(!(UART0->S1 & UART0_S1_TDRE_MASK) &&
          !(UART0->S1 & UART0_S1_TC_MASK) );
    UART0->D = ch;
}
```

příjem

```
char ReceiveCh(void) {

}
```

Bude na cvičení☺

Rozhraní SPI

Většina mikrokontrolérů obsahuje kromě modulu UART také modul pro synchronní sériovou komunikaci, typicky SPI – Serial Peripheral Interface.

Jedná se o rozhraní určené původně k připojování a komunikaci s periferními zařízeními, využívá se však hojně jako levná varianta pro sériový přenos obecně.

Podobně jako UART umožňuje též SPI kromě spojení *point-to-point*, tedy spojení dvou zařízení, také vytvoření SPI sběrnice, řízení komunikace je však mírně odlišné.

Rozhraní SPI je **plně duplexní**, v každém okamžiku vždy probíhá přenos oběma směry, tj. formou výměny dat.

Jedná se o rozhraní typu **master-slave**.

Master-Slave režim SPI

Zařízení připojené na rozhraní SPI může být v jednom ze dvou režimů: master nebo slave.

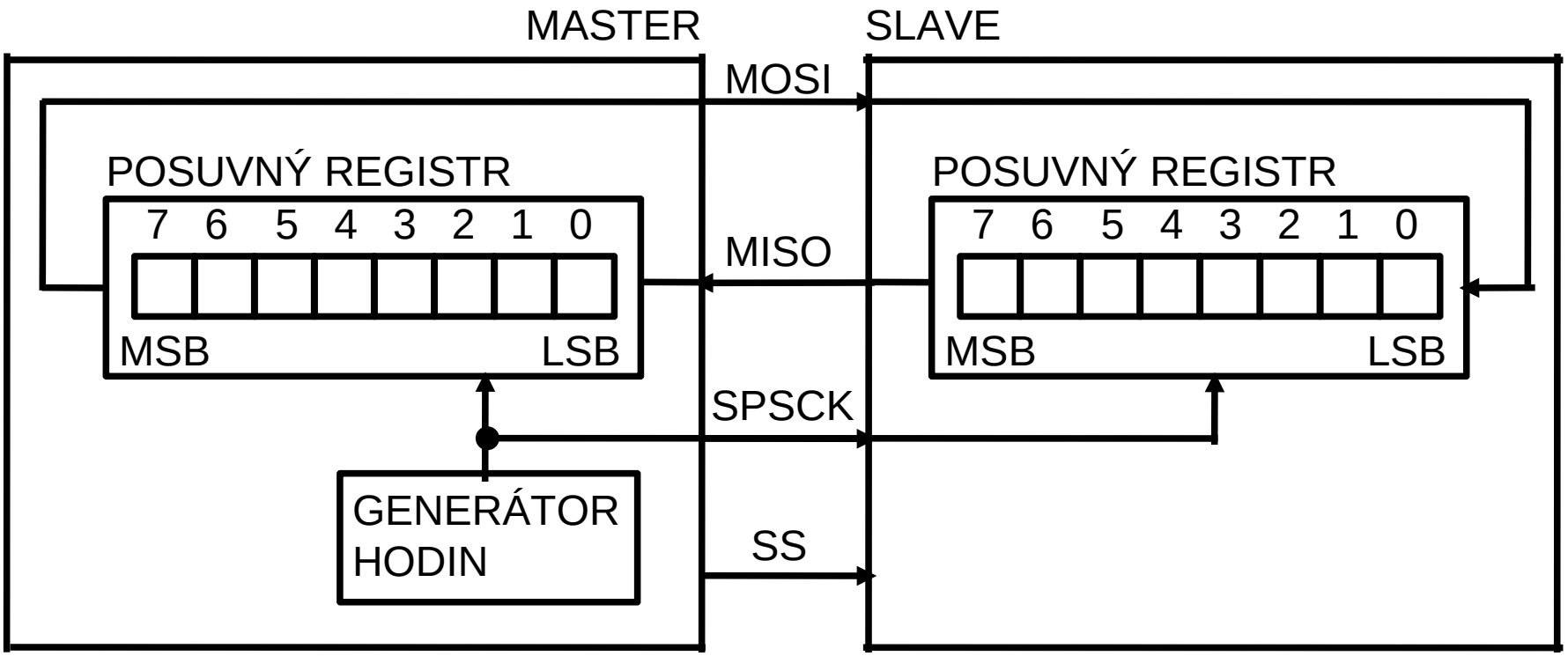
Master je zařízení, které generuje synchronizační hodinový signál, iniciuje a řídí komunikaci přes SPI.
Na sběrnici SPI smí být pouze jediné zařízení v režimu master.

Zařízení typu slave jsou obvykle periferie, se kterými master komunikuje. Právě zařízení master určuje, se kterým zařízením slave bude v daném čase komunikovat.

Master zpravidla bývá mikrokontrolér, periferie pak vystupuje v roli zařízení typu slave.

Základní schéma rozhraní SPI

Komunikace probíhá způsobem vzájemné výměny obsahů posuvných registrů mezi zařízeními master a slave.



Vodiče rozhraní SPI

Datové vodiče se na mikrokontroléru označují jako MISO a MOSI. Je tím vyjádřen směr toku dat podle režimu, v němž se dané zařízení připojené na SPI rozhraní nachází.

Vodič MISO je vstupem (Master In) dat zařízení master a výstupem (Slave Out) zařízení slave. U vodiče MISO (Master Out, Slave In) je tomu přesně naopak. Pokud může zařízení pracovat pouze v režimu slave, což bývá časté u periferií s rozhraním SPI, jsou datové vodiče označeny jednoduše jako SO (výstup) a SI (vstup).

Hodinový signál je generován na vodiči SPSCK (SPI Serial Clock), na zařízení master je to vždy výstup a na zařízení slave vstup. Je tomu tak proto, že hodinový signál vysílá právě zařízení master.

Vodiče rozhraní SPI

Výběrový signál SS (Slave Select), který je aktivní v log. 0, slouží k výběru a aktivaci zařízení, se kterým má probíhat komunikace.

Na zařízení master je signál SS výstupem připojeným k modulu slave. Ten je možné prostřednictvím mastera signálem SS aktivovat a zahájit tak datový přenos. Master musí zajistit, že SS bude po dobu přenosu v log. 0 (aktivní). V daném čase smí probíhat komunikace pouze s jediným modulem slave, ostatní zařízení, jejichž SS vstup je v log. 1 (neaktivní) do komunikace nezasahují. Signál SS má v režimu master ještě další využití (viz dále).

Na zařízení slave je signál SS vstupem, který v aktivní úrovni způsobí synchronní odesílání obsahu posuvného registru na stranu mastera.

Komunikace Master-Slave přes rozhraní SPI

Oproti rozhraní UART, kde vysílač a přijímač pracují zcela samostatně a tok dat oběma směry může probíhat na sobě nezávisle, tok dat přes SPI probíhá odlišně.

Master generuje synchronizační signál SPSCK a tím, že aktivuje výběr modulu slave (obvykle SS do log. „0“), zahájí komunikaci. V taktu SPSCK vysouvá na vývod MOSI postupně datové bity ze svého posuvného registru, zároveň však z registru druhé strany přijímá přes signál MISO data z posuvného registru modulu slave. Jedná se tedy o komunikaci způsobem vzájemné výměny dat, která je pro rozhraní SPI typická.

Komunikace Master-Slave přes rozhraní SPI

Po osmi taktech na SPSCK tak je původních 8 bitů z posuvného registru Master odesláno, současně je v posuvném registru 8 bitů přijatých od modulu slave.

Je to dáno tím, že stejným hodinovým signálem SPSCK byl taktován posuvný registr modulu v režimu slave, který stejným způsobem vysílal/přijímal.

Po těchto osmi taktech mohou master i slave přečíst nově přijatá data ze svého přijímacího registru a zapsat nová data určená k vysílání.

Jednostranný přenos přes SPI je spojen s odesláním nebo příjmem bezvýznamného znaku druhé strany.

Podrobný popis registrů a funkcionality UART viz Referenční manuál MCU KL05Z dostupný v modulu e-learning v sekci Lab. cvičení od str. 571

Hlavní rysy programování komunikace SPI: např. výměna dat ‘M’ ←→ ‘S’

vysílač (master)

```
void SPI_WriteReadData(void)
{
        while(!(SPI0_S & SPI_S_SPTEF_MASK ) );
        SPI0_S |= SPI_S_SPTEF_MASK;
        SPI0_D = transmit;
        while(!(SPI0_S & SPI_S_SPRF_MASK ) );
        received = SPI0_D;
}

int main(void)
{

    MCUInit();
    PinInit();
    SPI_InitMaster();

    SPI_WriteReadData();
    // chvili pockej, az prenos probehne
    delay(10000);

    // prijali jsme ocekavany znak od Slave ('S')?
    if (received == 'S')
        // pokud ano, rozblikej LEDku....forewer
        RGB_blink();
    // the end
}
```

přijímač (slave)

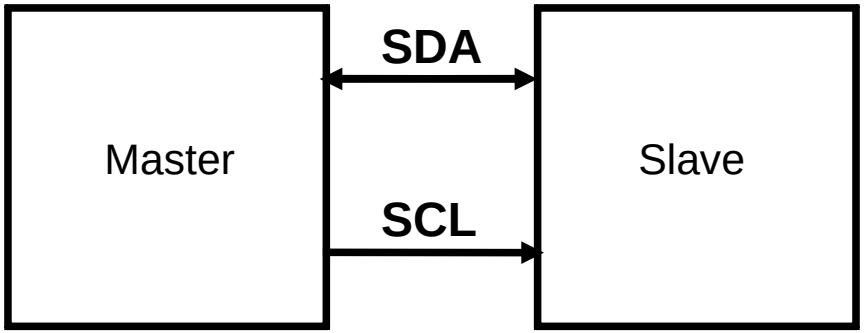
```
void SPI_Irq( void )
{
    if( SPI0_S & SPI_S_SPRF_MASK )
        received = SPI0_D;

    if( SPI0_S & SPI_S_SPTEF_MASK )
        SPI0_D = transmit;
}

int main(void)
{

    MCUInit();
    PinInit();
    SPI_InitSlave();

    while (1);
}
```

Rozhraní IIC (I²C)

I²C (Inter-Integrated Communication) je synchronní sériové rozhraní, jehož hlavním cílem je jednoduchost a levná implementace.

I²C specifikuje fyzickou podobu rozhraní, jeho elektrické charakteristiky a komunikační protokol.

Jedná se o protokol typu master-slave komunikující způsobem half-duplex.



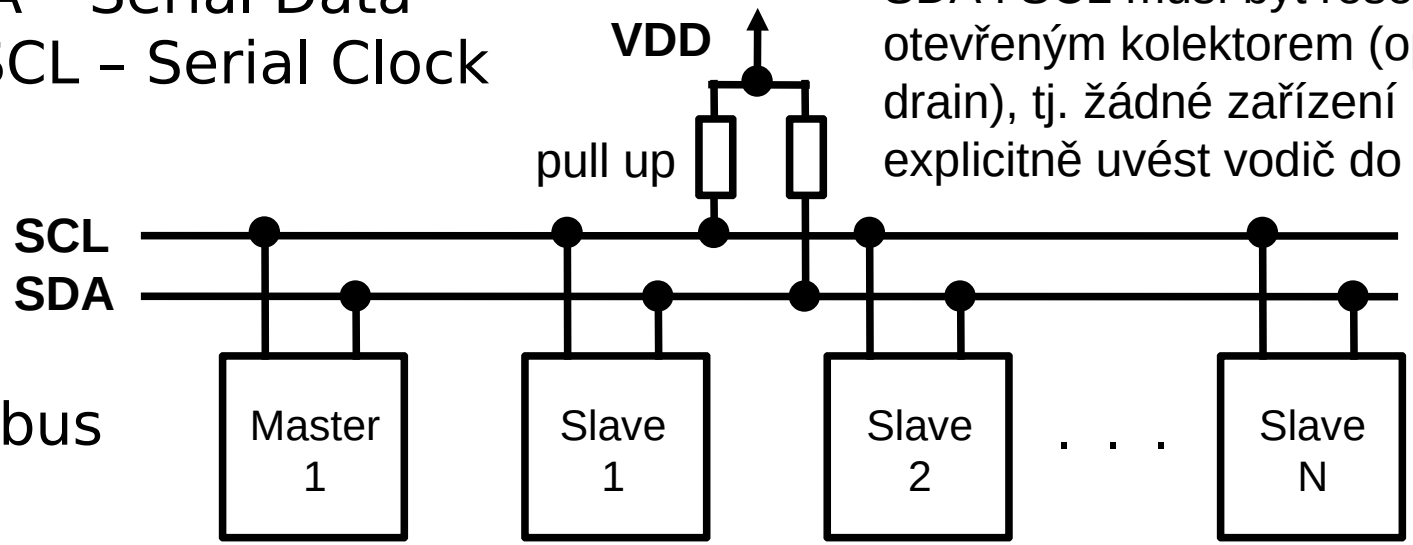
Původní návrh a specifikace (Philips Semiconductor, 1982), později bylo I2C několikrát aktualizováno a představena různá kompatibilní řešení i jinými výrobci.

Základní vlastnosti rozhraní I²C

Rozhraní I²C sestává z jednoho datového vodiče a jednoho vodiče přenášejícího synchronizační signál.

Ke *sběrnici* I²C lze připojit až 127 zařízení slave (dáno formátem adresy – viz dále).

SDA – Serial Data
SCL – Serial Clock



SDA i SCL musí být řešeny otevřeným kolektorem (open-drain), tj. žádné zařízení nemůže explicitně uvést vodič do log. 1.

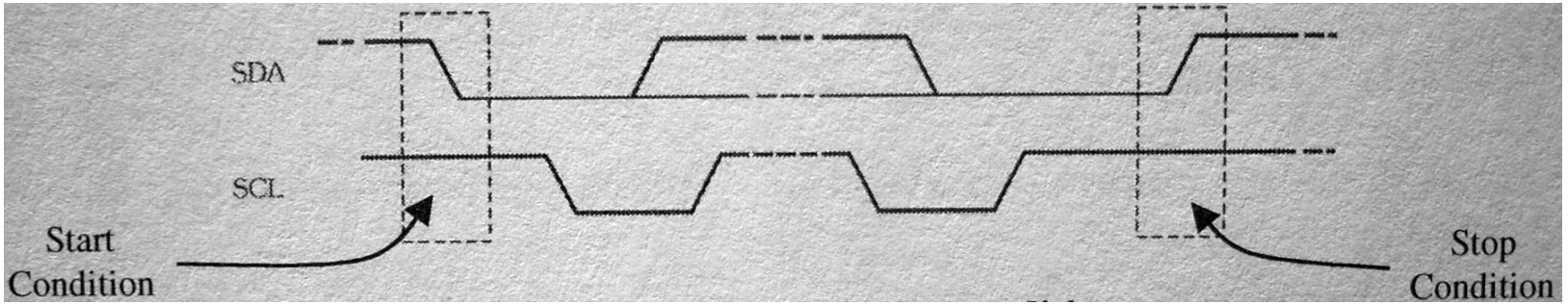
Pravidla komunikace přes I²C

Neprobíhá-li komunikace, master ani slave nijak neovlivňuje vodiče a pull up zařízení zajistí na SDA i SCL stav log. 1.

Komunikaci vždy zahajuje master hranou **1→0** (**start condition**) na SDA při SCL v log. 1.

Data jsou vzorkována výhradně při náběžné hraně SCL, změna hodnoty na SDA je prováděna při SCL v log. 0.

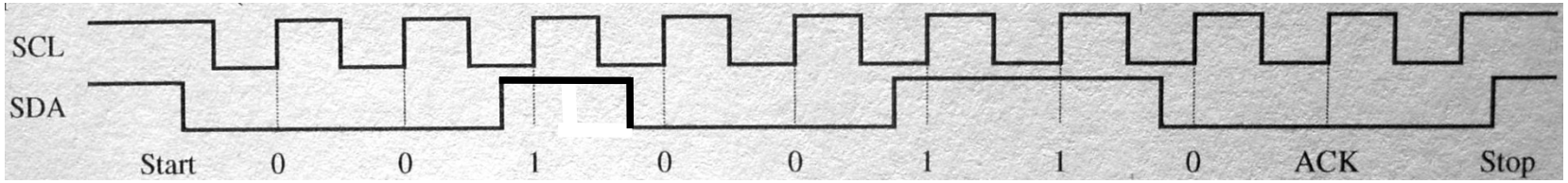
Pro ukončení přenosu generuje master hranu **0→1** (**stop condition**) při SCL v log. 1.



Datový rámec rozhraní I²C

Data jsou přenášená sériově po jednotlivých bitech, standardně 1 byte v datovém rámci.

Datový rámec se skládá ze startovací hrany (Start), datové bloky (po 8 bitech) s potvrzením ze strany modulu slave o rozpoznání znaku (handshake, ACK) a ukončující hrany (Stop).

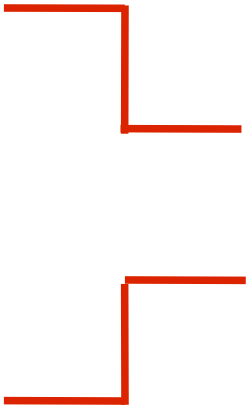


Vodiče rozhraní I²C

Vodiče rozhraní I²C mají několik různých využití pro řízení komunikace.

Serial Data (SDA)

- start condition SCL=1, SDA: 1→0
- přenos adresy/dat
- potvrzení příjmu (handshake)
- stop condition SCL=1, SDA: 0→1



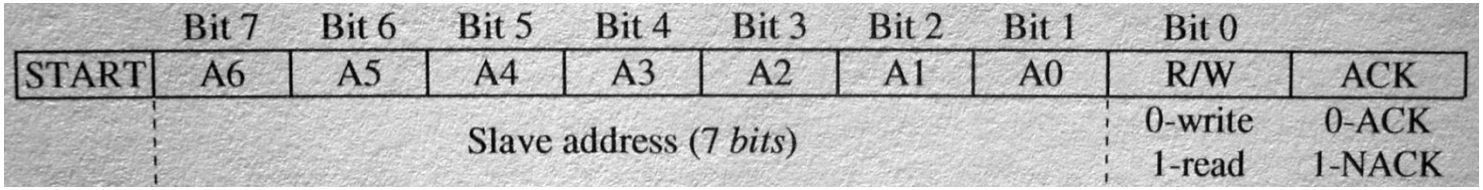
Serial Clock (SCL)

- synchronizační signál
- pozdržení přenosu ze strany modulů slave (clock stretching) podržením SCL v log. 0

Adresování na sběrnici I²C

Abychom mohli jednoznačně identifikovat zařízení pro komunikaci, je třeba zavést unikátní adresy modulů na I²C sběrnici. Zahájení komunikace (výběr modulu slave) pak spočívá v odeslání 8 bitů v prvním (adresovém) rámci, které jsou interpretovány jako adresa.

Adresa v rozhraní I2C má obvykle následující formát:



- 7 bitů adresa (bity A6:A0)
- 1 R/W bit: 1 – čtení (přenos dat slave→master), 0 – zápis (přenos dat master→slave)

Průběh komunikace na sběrnici I^2C (hlavní kroky)

Zařízení master generuje startovací hranu na SDA.

Master odešle adresový rámec.

Podle hodnoty bitu R/W v adrese master buď zasílá data modulu slave (R/W=0) nebo data čte (R/W=1).

Po přenesení všech dat generuje ukončující hranu.

Poznámka: protokol I2C umožňuje tzv. opakovaný start generováním nové startovací hrany a adresy bez nutnosti nejprve generovat ukončující hranu.

Detaily k rozhraní I2C na MCU MKL05Z viz Referenční manuál MCU KL05Z dostupný v modulu e-learning v sekci Lab. cvičení od str. 603.

Příklad programování I2C vizte např. následující „Application Note“: https://www.nxp.com/docs/en/application-note/AN4481.pdf

Konec přednášky

Děkuji za pozornost