Mikropočítačové Systémy

MIPS

Prednáška 9.
TWI - I2C.

Byte-oriented 2-wire serial interface (Phillips I2C compatible)

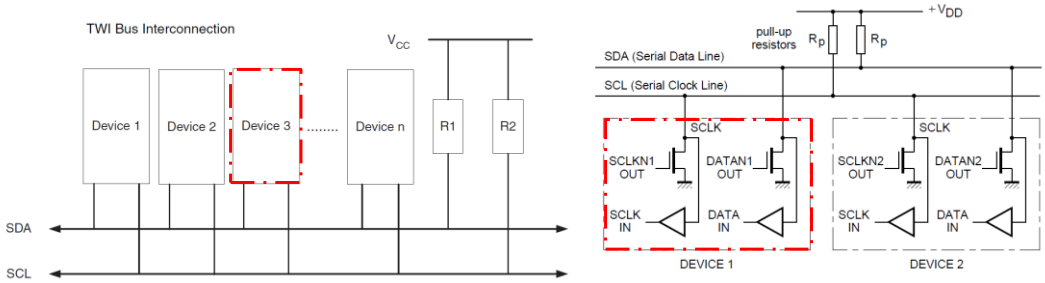
I2C je tak zložitá, že museli použiť stavový automat.
Je tak jednoduchá, že: MacGyver + 2 vodiče + 2 rezistory + baterka (+ LED)
a informácia v I2C zariadení je zmenená , resp. z neho prečítaná.
Len máločo dokáže plynulo meniť rýchlosť od nuly po max.
Otázka: Aké dlhé ...? Odpoveď: 400 pF.

I^2C je skratka pre Inter-IC. I^2C zbernica je obojsmerný dvojvodičový komunikačný kanál používaný na výmenu informácií medzi viacerými IC (integrovanými obvodmi). Zbernica bola vyvíjaná na prácu v rámci jedného zariadenia (pôvodne televízora). Hovorí sa o dvojvodičovom prenose - Serial Data (SDA) a Serial Clock (SCL). Zem býva v rámci zariadenia spoločná a preto ju netreba viesť zvláštnym vodičom. Niekedy sa hovorí o dvojvodičovej zbernici. Vodiče sú ale štyri.

I2C (TWI) *Inter Integrated Circuit*

- **I2C** je sériová synchronná zbernica vyvinutá firmou Philips na komunikáciu medzi IC vo vnútri zariadenia, napr. televízora.
- Na **I2C** zbernicu možno pripojiť EEPROM, ADC, LCD budiče, (>1000 IC's)
- Počet obvodov pripojených na zbernicu je obmedzený počtom adries a celkovou kapacitou zbernice < 400 **pF** („jednotkou dĺžky je **pF**").
- Na prenos informácie sú použité dve nesymetrické vedenia
  - SCL (hodiny)
  - SDA (data/adresa)

Tieto vodiče sú obojsmerné a pomocou PULL UP rezistorov ťahané hore. Všetky zariadenia pripojené na zbernicu musia mať „otvorený kolektor" (alebo …). Budiče zbernice majú implementované „drôtové AND".



2

TWI umožňuje prepojiť až 128 zariadení pomocou dvoch vodičov jeden pre hodiny – SCL a jeden pre dáta - SDA. Oba tieto vodiče musíme pomocou pull-up rezistorov pripojiť na napájanie. Každé pripojené zariadenie má vlastnú adresu.

Budiče zbernice všetkých pripojených zariadení na I2C musia byť typu otvorený kolektor (drôtové AND).

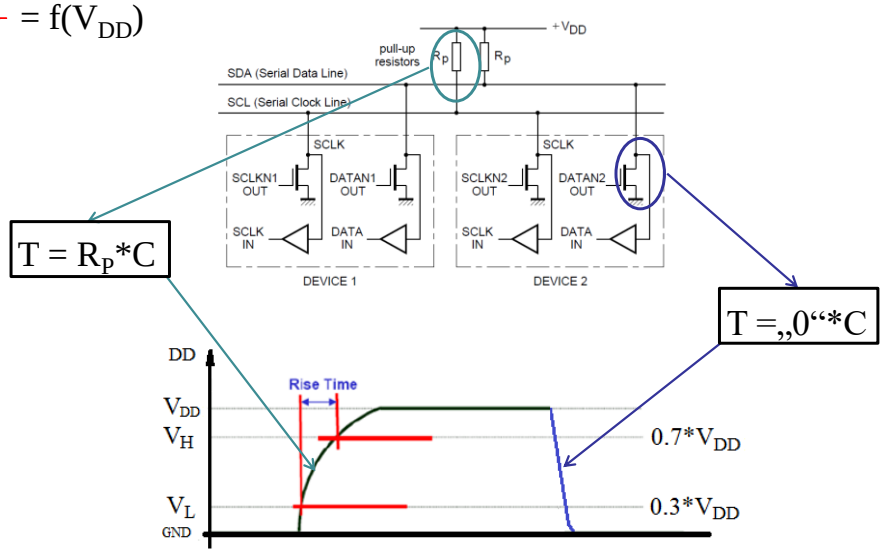
Nízka úroveň na zbernici je generovaná, tak že aspoň jedno výstupné zariadenie je vstave NULA. Vysoká úroveň na zbernici je generovaná tak, že všetky zariadenia pripojené na zbernicu sú v treťom stave. Vtedy je vodič ťahaný do vysokej úrovne pull-up rezistorom.

Počet zariadení pripojených na zbernicu je obmedzený kapacitou zbernice 400pF a 7-bitovou adresou Slave zariadení.

I2C

Zdroj:
Log. 1
= f(V_{DD})
Log. 0

Prijímač filtruje krátke impulzy na zbernici.



I2C

Každé zariadenie pripojené k zbernici je identifikované
7-bitovou, resp. (128 zariadení *mínus* 16 rezervovaných = 112)
10-bitovou adresou (1024 zariadení *mínus* rezervované adresy = 1008
„mostíkom“ medzi “7b –10b” je rezervovaná adresa 1111 0XX xxxx xxxx).

Módy prenosu (prenosová rýchlosť):

- **Standard mode**: <100kb/s
- **Fast mode**: <400kb/s
- **Fast mode plus**: <1000kb/s
- **High-speed mode:** <3400kb/s (10bit adresa, až 1024 zariadení)

Dôležité pojmy:

- **Transmitter** – zariadenie vysielajúce dáta na zbernicu.
- **Receiver** –zariadenie prijímajúce data zo zbernice.
- **Master** – inicializujúce (zahajuje) prenos na zbernici, generuje hodinové signály a ukončuje prenos. MASTER môže byť vo funkcii vysielača aj prijímača.
- **Slave** – zariadenie adresované MASTER-om. SLAVE môže byť aj vysielač aj prijímač.
- **Multi-master** – schopnosť súčasnej koexistencie viacerých MASTER-ov na zbernici bez kolízií a strát dát.
- **Arbitration** – vlastnosť, ktorá zabezpečí, že v danom čase len jeden MASTER riadi zbernicu.
- **Synchronization** – vlastnosť, ktorá zabezpečí, synchronizáciu hodinových signálov, ktoré generujú dve, resp. viaceré zariadenia typu MASTER.

4

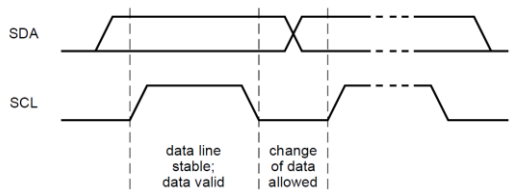
I^2C zbernica je plne statická čo znamená, že frekvencia zbernice môže ísť až do nuly.

Na zbernici sa môžu nachádzať dva druhy zariadení: master (zahajuje a riadi komunikáciu) a slave (oslovené zariadenie). Ak MASTER chce čítať/zapisovať z/do SLAVE zariadenia, musí najskôr poslať adresu SLAVE. Zariadenia pracujú v dvoch režimoch: prijímač a vysielač. Tento režim určuje MASTER. I^2C je zbernica typu multi-master, to znamená, že môže byť riadená viacerými MASTER zariadeniami. Každé zariadenie je určené sedem bitovou jednoznačnou adresou. Je dôležité si uvedomiť, že prenos môže začať iba ak zbernica nie je obsadená. Pokiaľ zbernica neprenáša údaje oba signály musia byť high. Ak by sa pokúšalo naraz pracovať viacero MASTER zariadení, arbitráciou sa rozhodne, kto dostane prioritu.

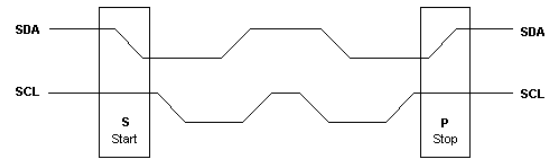
I2C protokol:

- Data sú po SDA prenášané bytovo. Niekoľko bytov ohraničených **S** a **P**.
- Na prenesenie jedného Byte-u MASTER generuje
  - 8-f_{I2C} pre data a
  - 1-f_{I2C} pre ACK. Prijímač potvrdzuje ACK.
    Tento jeden bit „vysiela“ prijímač.
- **Prenos bitu(ov)**
  Prenos bitu je podmienený jedným pulzom na SCL. Signál na SDA vodiči musí byť stabilný ak je CLK signál v úrovni Log. 1.

Jediná výnimka z tohto pravidla je mechanizmus generovania **S**tart a sto**P** podmienky.



Bit transfer on the I²C-bus.



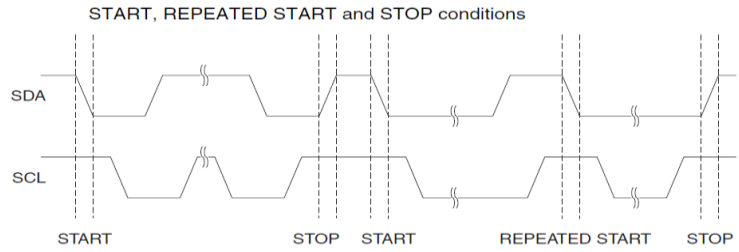
5

Údaje na SDA linky musia byť stabilné počas trvania hornej úrovne hodín. Úroveň na SDA linky sa smie meniť iba keď sú hodinové impulzy na SCL linky na dolnej úrovni (obrázok vľavo).

Riadiace signály môžu byť interpretované ako Štart alebo Stop podmienky podľa zmeny úrovne signálu.
Zmena úrovne SDA z high na low pri SCL HIGH je považovaná za (S)tart podmienku.
Zmena úrovne SDA z low na high pri SCL HIGH je považovaná za stop(P) podmienku.

I2C protokol: Prenos **S**tart a sto**P** podmienky.

- Ak je zbernica voľná, môže sa uskutočniť prenos. T.j. oba vodiče sú v logickej jednotke.
- MASTER inicializuje a ukončuje prenos.
  - Prenos sa začne, ak MASTER odvysiela **S**tart podmienku
  - prenos sa ukončí, ak MASTER odvysiela sto**P** podmienku.
  - Zbernica je medzi **S**tart a sto**P** podmienkou v stave „busy“.
  - Ak sa medzi **S**tart a sto**P** podmienkou objaví opakovaný **S**tart, tento stav sa označuje ako REPEATED **S**tart podmienka (Sr, S).
  - **S**tart a sto**P** podmienka sa realizujú ako zmena na SDA počas vysokej úrovne na SCL vodiči.

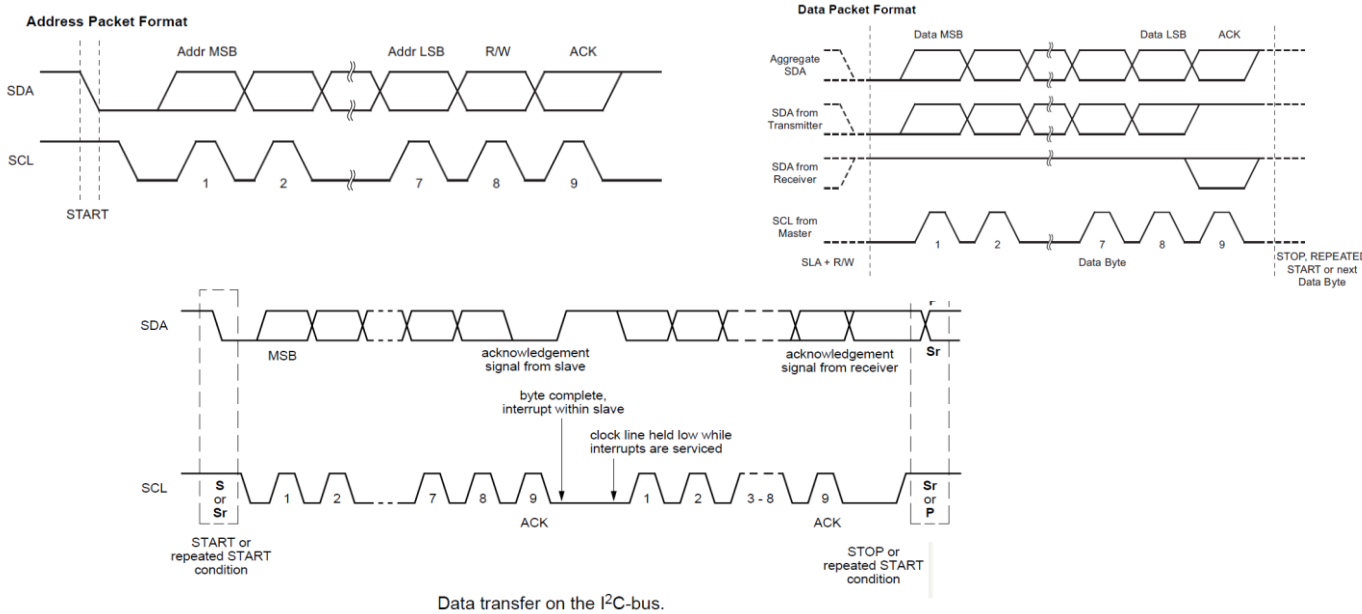


6

Každý bajt ôsmich bitov je nasledovaný acknowledge bitom (potvrdením). Počet bajtov prenesených medzi štart a stop podmienkami nie je obmedzený. Acknowledge bit je generovaný prijímačom, SCL je stále riadené masterom. Pretože je tento bit reprezentovaný low úrovňou, musí vysielač uvoľniť SDA aby mohol prijímač poslať low ak neobjavil žiadnu chybu. Ak slave prijímač nie je schopný poslať acknowledge drží SDA na high aby mohol master generovať stop podmienku a ukončiť prenos. V prípade, že master vysielač drží SDA high cez acknowledge, oznamuje tým koniec prenosu. Slave uvoľní SDA a master môže generovať stop podmienku.

I2C protokol: Prenos dát a formáty prenášaného rámca

- Všetky vysielané data pomocou I2C zbernice sú 9 bitové.



Data transfer on the I²C-bus.

7

Po štart podmienke master posiela adresu slave zariadeniu. Adresa je dlhá sedem bitov, ôsmy určuje smer toku údajov (read/write). Master vždy ukončuje prenos stop podmienkou. Pokiaľ by chcel master ďalej komunikovať po zbernici môže vygenerovať štart podmienku s inou adresou bez generovania stop podmienky.

Ak je nastavený bit čítania / zápisu, vykoná sa operácia čítania, ak nie je, vykoná sa operácia zápis. Keď SLAVE rozpozná, svoju adresu, mal by to potvrdiť počas 9-teho clk tak že nastaví linku SDA na nízku úroveň (ACK). Ak je busy ponechá vodič SDA na vysokej úrovni (NACK). Master potom môže generovať stoP alebo opakovaný Start. Poznáme dva adresné pakety: adresa slave a bit čítanie/ zápis: SLA + R, resp. SLA + W.

Najskôr sa prenáša MSB adresný byte. Slave adresy sú „volné“, okrem adresy 0000 000, ktorá je určená na všeobecné volanie.

Na výzvu „veobecné volanie SLA (0)+W“ by mali všetci SLAVE reagovať stiahnutím SDA na nulu počas cyklu ACK.

Toto volanie sa využíva na to aby MASTER mohol preniesť správu naraz všetkým SLAVE-om. SLAVE zariadenia, ktoré dajú počas ACK SDA do nuly, príjmu nasledujúce dátové pakety.

SLA(0) + R nemá zmysel, lebo by súčasne vysielali na zbernicu viacerý rôzne informácie.

Všetky adresy formátu 1111 xxx sú rezervované pre budúce použitie.

Aj všetky datové pakety su na TWI 9-bitové (Jeden dátový byte a ACK bit). MSB datový byte je prenášaný ako prvý. CLK generuje MASTER. A SLAVE posiela ACK, resp. NACK.

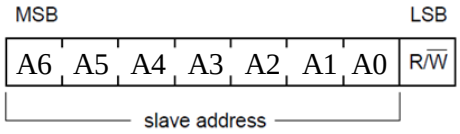
Prenos adresy a dát možno zlúčiť do jedného prenosu.

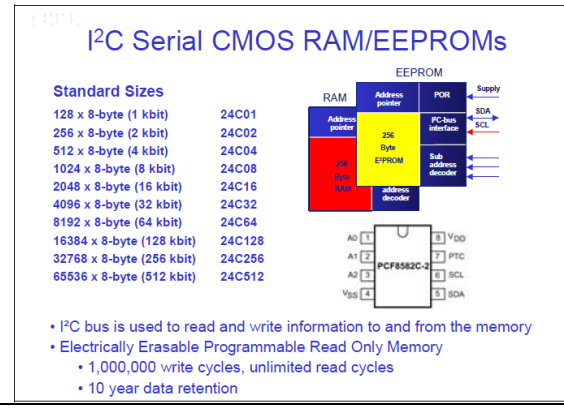
Prenos je obvykle tvorený Start podmienkou, SLA+R/W , jedným alebo níkoľkými dátovymy paketmi a stoP podmienkou

Tzv. drátové AND môžeme použiť pri výmene informácií medzi MASTER a SLAVE (handshaking). Slave môže podržať SCL na nule . To sa dá využiť, vtedy ak Master vysiela príliš rýchle. Samozrejme, že to nemá vplyv na trvanie SCL vysokej úrovne.

I2C protokol:

- Prvý byte (niekedy sa nazýva riadiaci byte) po **S**tart podmienke je rozdelený na:
- 7 bitov - adresa – (Pevná a programovateľná časť)
- 1 bit - Read/Write riadiaci bit. Tento bit nastavuje typ operácie.
- 1 bit – acknowledge bit (ACK). Ak adresované zariadenie rozpozná svoju adresu, potvrdí to tak, že počas 9-teho SCL hodinového signálu stiahne SDA na nízku úroveň. MASTER potom môže odvysielať sto**P** podmienku alebo opakovaný **S**tart, aby mohol inicializovať nový prenos. Adresný packet pozostávajúci z adresy SLAVE zariadenia a READ alebo WRITE podmienky sa označuje: SLA+R resp. SLA+W.



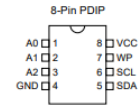


8

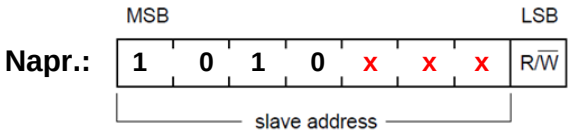
I2C protokol: „riadiaci byte“

Napr.:

AT24C32



- 7 bitov - adresa – (Pevná a programovateľná časť)



- 10bitová adresa

Table 2 Definition of bits in the first byte

SLAVE ADDRESS	R/W BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte(1)
0000 001	X	CBUS address(2)
0000 010	X	Reserved for different bus format(3)
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

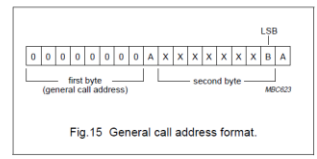
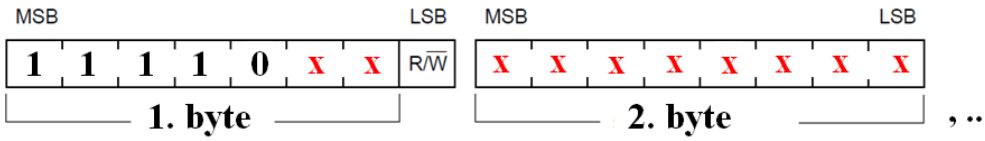


Fig.15 General call address format.

When bit B is a 'zero'; the second byte has the following definition:

- 00000110 (H'06'). Reset and write programmable part of slave address by hardware.



9

Adresný priestor zbernice (sedem bitov + R/W bit) je rozdelený na rôzne oblasti.

0000 000-0 Táto adresa znamená všeobecné volanie. Je to využívané na komunikáciu so *všetkými* zariadeniami, ktoré vedia toto volanie spracovať. Údaje prenášané týmto spôsobom môžu ale nemusia byť využité jednotlivými zariadeniami. Jeho význam je daný ďalším bajtom:

0000-0110 Reset zariadenia. Zariadenie spustí inicializáciu (okrem iného znova načíta svoju nastavenú adresu). Tento príkaz sa môže využívať pre reset celej zbernice.

0000-0100 To isté ale bez resetu. Vhodné pre zariadenia, u ktorých

nastala zmena na adresných pinoch. Takto môže zariadenie zmeniť svoju adresu.

xxxx-xxx1 Ak zariadenie potrebuje pozornosť mastera, bez toho aby vedelo o ktorý master sa jedná využíva toto volanie. Toto volanie je niečo ako „prosím ozvite sa, potrebujem obslúžiť“. (Nepripomína vám to prerušenie?) Všetci masteri toto zachytia a podľa toho, ktorý vie ako dané zariadenie (xxxx-xxx je jeho adresa) obslúžiť, ho skontaktuje.

0000 000-1 Toto je vlastne kópia štart podmienky. Používa sa pri masteroch, ktorý majú implementované I^2C softwarovo. Tieto masteri by museli po celý čas monitorovať zbernicu na štart / stop podmienky. S touto pomocou to robia len občas a pri detekovaní SDA low (až sedem cyklov) zvýšia frekvenciu monitoringu aby mohli zachytiť štart podmienku.

0000 001-x Pre už nepoužívanú C-bus. Táto adresa slúžila na prenos iných ako I^2C dát.

0000 010-x Rezervované pre iný formát. Táto adresa slúži na prenos iných ako I^2C dát.

0000 011-x Tieto adresy sú rezervované pre budúce rozširovanie a nie sú povolené.

0000 1xx-x Rezervované pre budúce použitie.

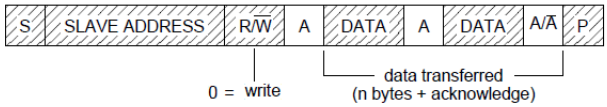
1111 1xx-x Rezervované pre budúce použitie.

1111 0xx-x Využívané pri rozšírení na 10bit adresovanie.

10bit Address: Pretože sa na I^2C zbernicu dá zapojiť maximálne 127 zariadení (aj to z jedného druhu len veľmi malý počet kvôli pevným rozsahom adries) nastalo rozšírenie adresy na 10 bitov, čo umožňuje zapojiť až 1024 zariadení (tým pádom aj viac rovnakých druhov). Je to spätne kompatibilné a transparentné riešenie pre staršie zariadenia. Toto adresovanie sa dá použiť pre všetky rýchlosti zbernice I^2C. Počet zariadení je ešte limitovaný celkovou pripojenou kapacitou na zbernicu - 400pF. Typické zariadenia pripájané na zbernicu mávajú okolo 10pF. 10bit adresovanie funguje vysielaním dvoch adresných bajtov. Prvý bajt je 11110(A_9)(A_8)(R/W). Zariadenia, ktoré toto nepodporujú, zistia inú ako svoju adresu a začnú čakať na stop podmienku. Druhý bajt obsahuje zvyšných osem bitov adresy (A_7)..(A_0). Zariadenie, ktoré detekuje svoju 10bit adresu odpovie úplne štandardne pomocou acknowledge. Komunikácia prebieha ďalej ako v 7bit adresovaní.

I2C protokol:

MASTER (TRANSMITTER) – SLAVE (RECEIVER)

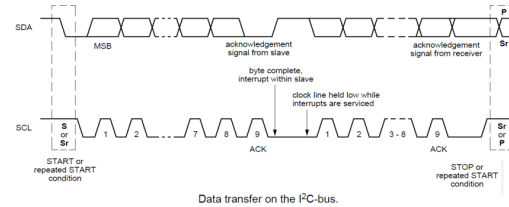


A master-transmitter addressing a slave receiver with a 7-bit address.
The transfer direction is not changed.

from master to slave

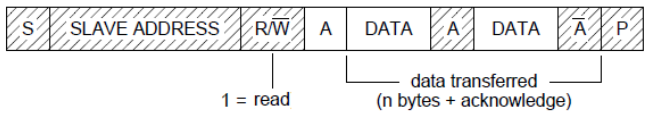
from slave to master

A = acknowledge (SDA LOW)
Ā = not acknowledge (SDA HIGH)
S = START condition
P = STOP condition



Data transfer on the I²C-bus.

MASTER (RECEIVER) – SLAVE (TRANSMITTER)



A master reads a slave immediately after the first byte.

I2C protokol: Riešenie problémov na zbernici

Riešenie konfliktov je založené na počúvaní.

Vysielač „budí zbernicu“ a zároveň kontroluje - číta, či sa na zbernici objaví to, čo vysielač na zbernicu poslal:

- Log.1 – mäkký zdroj signálu
- Log.0 – tvrdý zdroj signálu

Dva dôvody, že sa neobjaví Log. 1

– Log. 1 (*) Log. 0 = Log. 0.
Dôvod tohto stavu. Napr.: Iný Master (Log. 0) pôsobiaci na zbernici

– Mäkký zdroj Log. 1.
Dôvod tohto stavu. Vysielač nedokáže v danom čase „nabiť“ cele vedenie (celú kapacitu)

Synchronný prenos: môžeme spomaliť, pozastaviť

Pomalšie zariadenie (SLAVE) môže taktovanie MASTER:

– **spomaliť**. SLAVE zariadenie môže v každej perióde podržať SCL na log. 0.

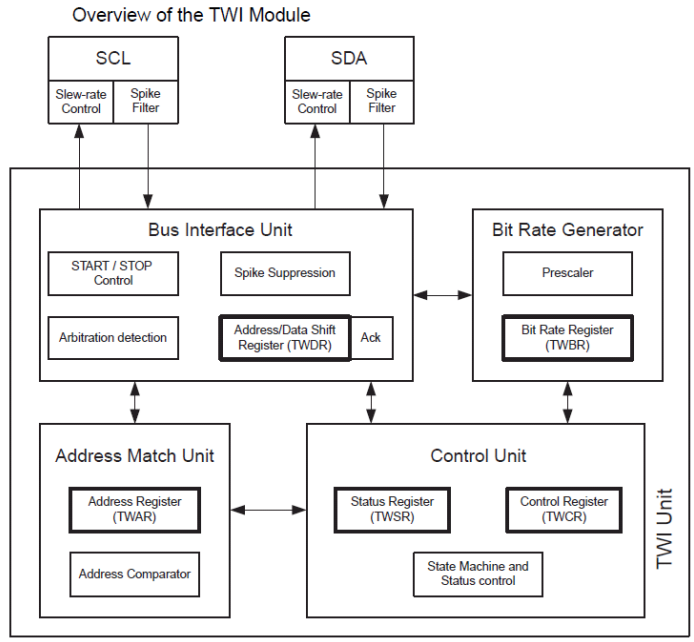
– **pozastaviť**. Ak potrebuje SLAVE zariadenie čas na spracovanie dát podrží SCL na úrovni log. 0 po ACK bite.

MultiMaster režim:

MASTER – inicializujúce (zahajuje) prenos na zbernici, generuje hodinové signály a ukončuje prenos. MASTER môže byť vo funkcii vysielača aj prijímača. Ak je na zbernicu pripojený len jeden MASTER, je komunikácia po zbernici jednoduchá.
Ak je na zbernicu pripojených viacero MASTER-ov, treba určiť, kto v danom stave riadi zbernicu.
Treba riešiť dve úlohy:

– **synchronizáciu hodín** (SCL)**:** Ak čo i len jeden MASTER nastaví SCL na Log. 0, zbernica je na Log. 0. Ak MASTER uvoľní SCL, musí monitorovať (počúvať) a môže taktovať Log. 1 až vtedy, keď všetci uvoľnia linku.

– **arbitráž:** Rieši kolízie na vodiči SDA. MASTER musí „počúvať" či to, čo na linku nastavil, aj prečíta. Ak MASTER prečíta, iný stav ako nastavil, musí uvoľniť linku (prejsť do stavu SLAVE – čo ak je adresovaný ako SLAVE zariadenie?). Je zrejmé, že tento stav nastane, ak jeden MASTER vysiela log. 1 a druhý log. 0. „zvíťazí" ten, ktorý vysiela na SDA log. 0.



Na vstupe TWI je zabudovaný filter, ktorý odstráni impulzy kratšie ako 50ns.
AVR TWI má zabudovaný stavový automat, ktorý generuje požiadavku o prerušenie pri každej zmene stavu na zbernici.

TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – TWINT: TWI Interrupt Flag sa nastaví, keď sa požaduje reakcia softvéru na zmenu stavu zbernice. Nulovanie zápisom Log.1 softvérom, ako posledná operácia s registrami TWAR (adresný reg.), TWSR (status reg.) a TWDR (data reg.).

Bit 6 – TWEA: TWI Enable Acknowledge Bit riadi činnosť počas ACK impulzu. Po zápise Log.1 je generovaný ACK impulz ak: 1. Bola prijatá adresa SLAVE zariadenia. 2. Všeobecné volanie. 3. MASTER alebo SLAVE prijal data.

Bit 5 – TWSTA: TWI Start Condition Bit. Tento bit nastavuje MASTER. Ak sú splnené podmienky odvysiela sa Start podmienka. Tento bit treba vynulovať softverovo, po odvysielaní Start podmienky.

Bit 4 – TWSTO: TWI Stop Condition Bit. Tento bit nastavuje MASTER. Ak chce odvysielať stoP. Nulovaný je automaticky.

Bit 3 – TWWC: TWI Write Collision Flag. Tento bit sa nastavíak nie je dovolené zapísať do TWDR reg.

Bit 2 – TWEN: TWI Enable Bit. Zapnutie TWI. TWI preberá riedenie SCL a SDA. Pinov.

Bit 0 – TWIE: TWI Interrupt Enable. Ak nastavíme aj I-bit v SREG, aktivujeme požiadavku o prerušenie.

Bit 7..3 – TWS: TWI Status reprezentujú stav TWI .

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

Poznánka: Status = TWSR & 0xF8;

TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

V režime vysielania obsahuje TWDR ďalší odosielaný bajt.
V režime príjmu obsahuje TWDR posledný prijatý bajt.

TWI – taktovanie, hodiny

$$f_{CPU} = 16\ MHz$$

$$f_{TWI} = \frac{f_{CPU}}{16 + 2(TWBR)(Prescaler\ Value)}$$

TWBR = Value of the TWI Bit rate register
PrescalerValue = Value of the prescaler

Požadujeme $f_{TWI} \equiv 100kHz \Longrightarrow$
$TWBR = 5 \quad \Longrightarrow$

$$f_{TWI} = \frac{16MHz}{16 + 2 * (16) * (5)} \cong 91kHz$$

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

Preddelič TWI zbernice.

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

Deliaci pomer pre CLK generátor TWI v MASRER móde.

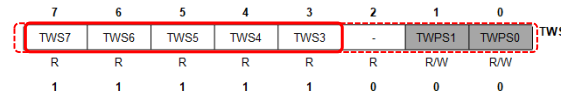
16

Činnosť Slave zariadenia nesúvisí s nastavením taktovania MASTER, ale f_{CPU} SLAVE musí byť aspoň 16 krát vyššia ako je frekvencia CLK.

Módy komunikácie (4):

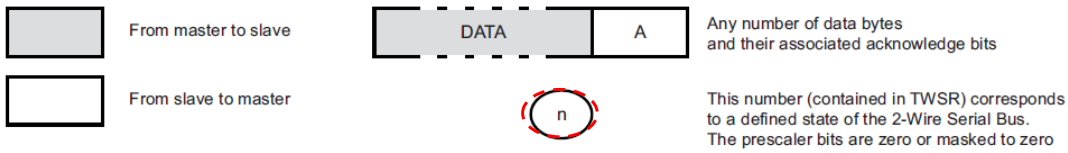
- MASTER vysiela (MT)
- MASTER prijíma (MR)
- SLAVE vysiela (ST)
- SLAVE prijíma (SR).

Tieto módy môžu byť aj kombinované navzájom.



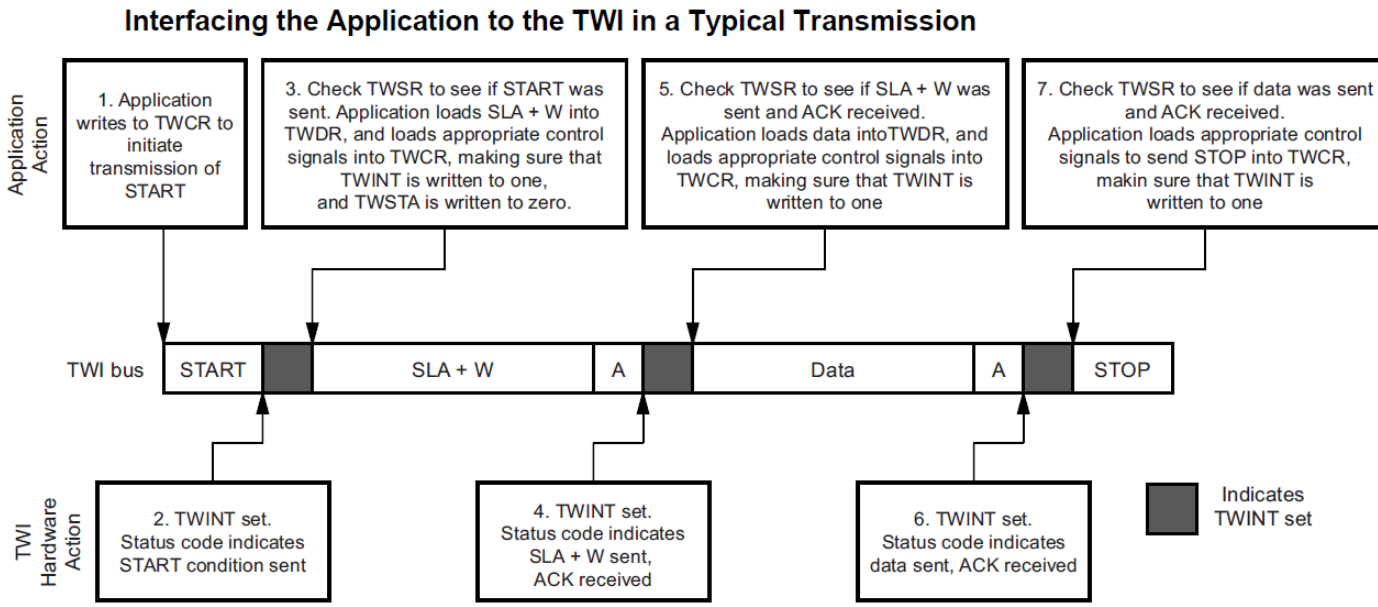
Vysvetlívky:

S:	Start podmienka
Rs:	Repeated Start podmienka
R:	Read bit (high level at SDA)
W:	Write bit (low level at SDA)
A:	Acknowledge bit (low level at SDA)
/A, $\overline{A}$:	Not acknowledge bit (high level at SDA)
Data:	8-bit data byte
P:	stoP podmienka
SLA:	Slave address

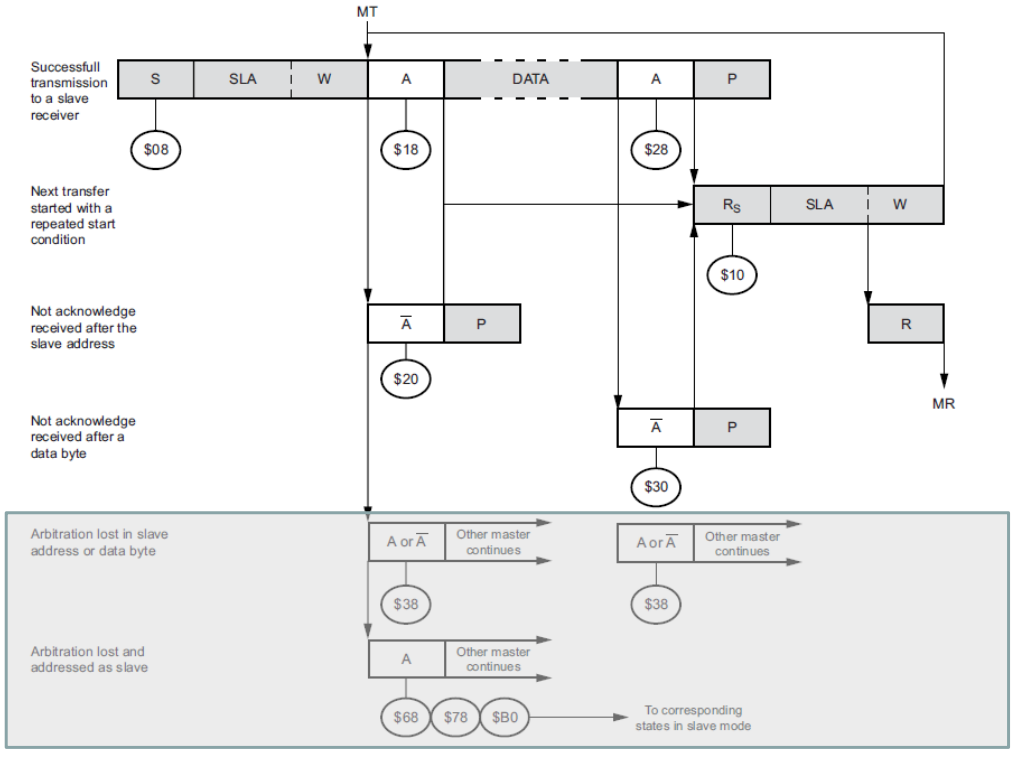


17

Príklad kooperácie softvéru a TWI hardvéru:



Formats and States in the Master Transmitter Mode



19

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

SLA+W ⟼ TWDR

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

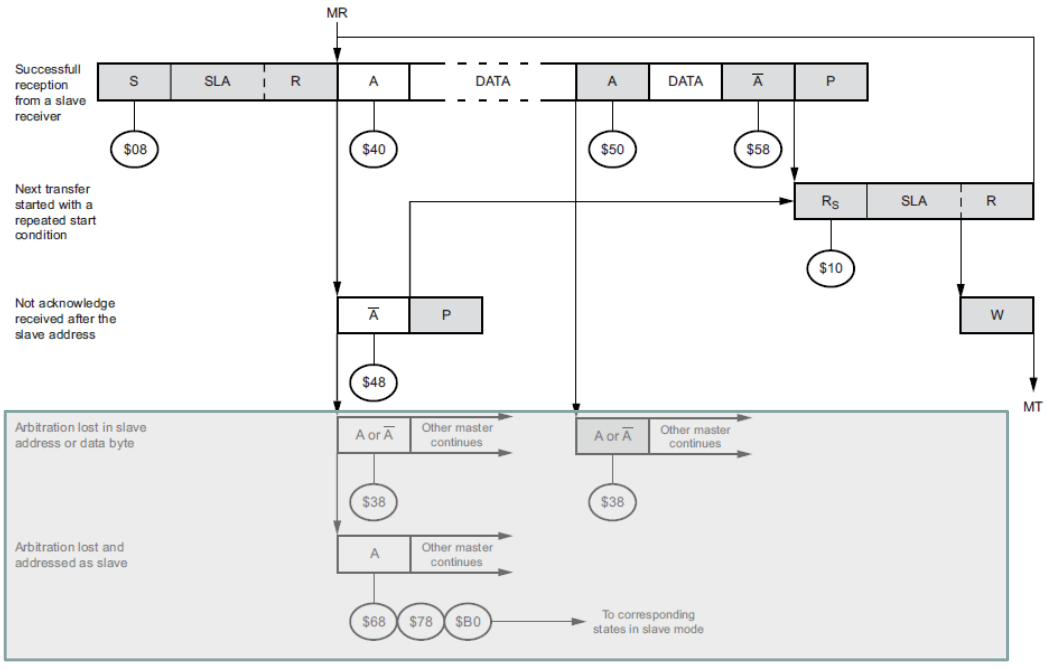
Zápisom 1-ky vynulujeme.

Status Codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to master receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset

20

Formats and States in the Master Receiver Mode

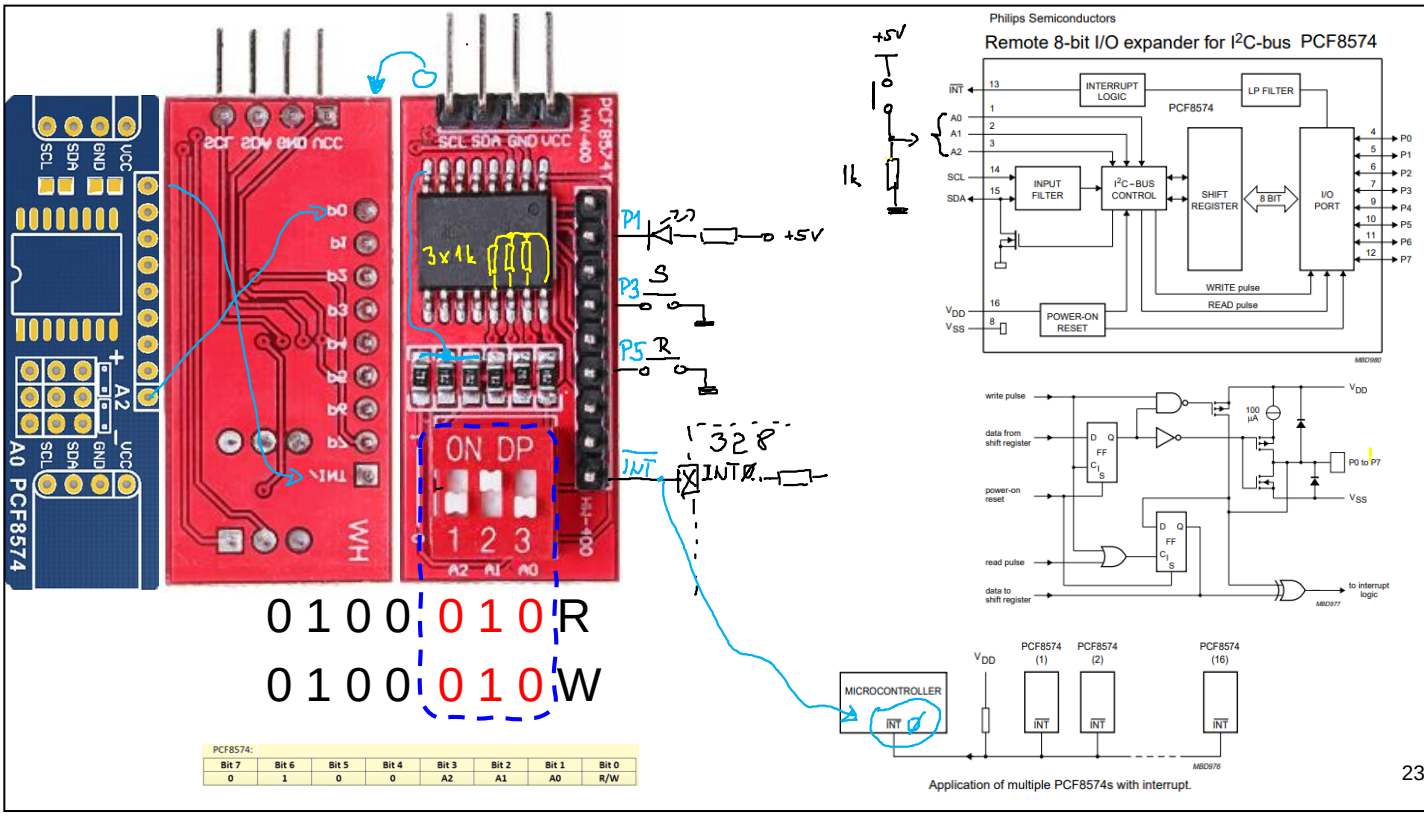


21

Status Codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted logic will switch to master transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	2-wire serial bus will be released and not addressed slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset

22



23

Pozor vyrábajú sa dva typy: PCF8574 (pevná časť adresy 0100) a PCF8574A (pevná časť adresy 0111).

Našou úlohou: Nastaviť na začiatku port ako In. T.j- zapíšeme jednotky. On je aj po pripojení napajania ako Input.

1. Potom P3 (zatlačenie) bude Zapínať led a P5 (zatlačenie) bude Vypínať led.
   Vytvoríme jednoduchý RS klopný obvod.
2. Naprogramujeme TWI.
3. Aby sme nemuseli neustále testovať stav klávesnice, využijeme pin /INT na generovanie prerušenia (privedieme na pin INT0 MMP). Tento pin sa zhodí automaticky po vstupe do prerušenia.

Červený modul:

- má ošetrené A0, A1 a A2 odporom 1k do nuly.
- INT a SCL a SDA má pullup 1k.

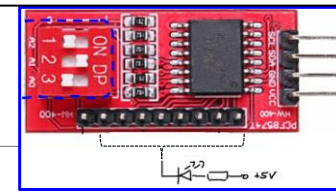
Modrý modul:

- SCL a SDA má pullup 1k.
- INT musí mať externý.
- A0, A1 a A2 má pripojené na GND, resp. +5V cez prepojky.

-

Cv_x_TWI.c

```
#define Adr_exp_8574 0x22 // Nastavenie adresy SLAVE zariadenia
#include "p_f_1.h"

volatile uint8_t i_TWI = 0x00;
volatile uint8_t pocet_TWI_WR,pocet_TWI_RD;
volatile uint8_t TWI_WRB[10];
volatile uint8_t TWI_RDB[10];
volatile uint8_t Stav_exp = 0;
volatile uint8_t flag_exp = 0;

int main(void)
{
  ini_TWI(); // ini  TWI
  sei();
  for(uint8_t i=0;i<10;i++){
        TWI_WRB[i] =0x0;TWI_RDB[i] =0x0;
  }

  while(1) {
    WR_exp(Adr_exp_8574,0xFF);
    _delay_ms(500);
    WR_exp(Adr_exp_8574,0x00);
   _delay_ms(500);
    }
}
```



p_f_1.c

```
/* ********************************************/
/* zapis do  expandera PCF8574                */
/* mozne adresy 0x20,0x21,...0x26,0x27        */
/* zapis do  expandera PCF8574A               */
/* mozne adresy 0x70,0x71,...0x76,0x77        */
/*  Bit0 = 0 (write)                          */
/********************************************/
void WR_exp(char adr_IO,char wr_data) {
 while(check_bit(TWCR,TWIE)); //WDR();
 pocet_TWI_WR = 2; // pocet prenasanych byte-ov
 TWI_WRB[0] = adr_IO<<1;
 TWI_WRB[1] = wr_data;
 i_TWI = 0;
```

TWCR

7	6	5	4	3	2	1	0
TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
R/W	R/W	R/W	R/W	R	R/W	R	R/W
// 1	0	1	0	0	1	0	1

```
 // K_TWI_START= 0xA5
 // TWINT, /TWEA, TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
 TWCR = K_TWI_START;
}
```

24

Úloha č. 1.

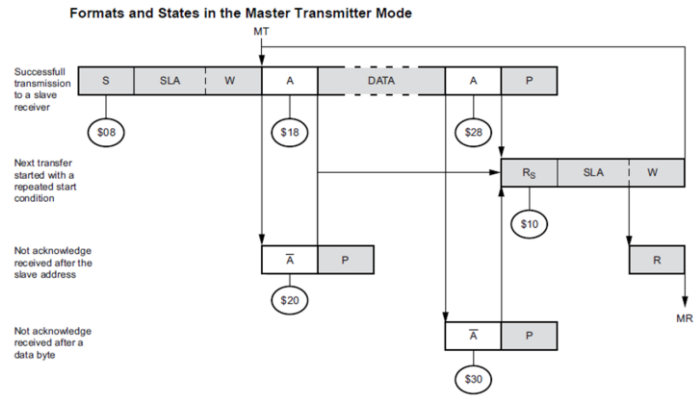
Celý expander nastavíme opakovane do log.1 a potom do log. 0.

p_f_1.c

```
#include "p_f_1.h"

void ini_TWI(void) {// ini  TWI
  TWBR = 0x05;
  TWSR = 0x03;
}

void WR_exp(char adr_IO,char wr_data) {
 // vid. predch. strana
}
```

Nech: $TWBR = 5$ a $TW\_B\_R\_P = 64 \Rightarrow$

$$f_{TWI} = \frac{16MHz}{16 + 2 * (64) * (5)} \cong 24{,}4kHz$$




```
ISR(TWI_vect) {
  switch(TWSR & 0xf8)
  { case StK_TWI_START:          // [0x08]   START has been transmitted
    case StK_TWI_REP_START:      // [0x10]   Repeated START has been transmitted
    case StK_TWI_MTX_ADR_ACK:    // [0x18]   SLA+W has been tramsmitted and ACK received
         TWDR=TWI_WRB[i_TWI++]; // adr_..., data
         TWCR=K_TWI_ADDRESS;    // =0x85  TWINT, /TWEA, /TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
    break;
    case StK_TWI_MTX_ADR_NACK:   // [0x20]   SLA+W has been tramsmitted and NACK received
    case StK_TWI_MTX_DATA_ACK:   // [0x28]   Data byte has been tramsmitted and ACK received
    case StK_TWI_MTX_DATA_NACK:  // [0x30]   Data byte has been tramsmitted and NACK received
    case StK_TWI_BUS_ERROR:      // [0x00]   Bus error due to an illegal START or STOP condition
    default:
    TWCR = K_TWI_STOP;           // [0x94]   TWINT, /TWEA, /TWSTA,  TWSTO, /TWWC, TWEN, 0, /TWIE
    break;
  }
}
```

25

Delička je nastavená na 64.

p_f_1.h

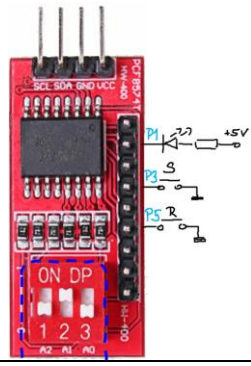
```
#ifndef P_F_1_H_
#define P_F_1_H_
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

extern volatile uint8_t i_TWI;
extern volatile uint8_t pocet_TWI_WR,pocet_TWI_RD;
extern volatile uint8_t TWI_WRB[10];
extern volatile uint8_t TWI_RDB[10];
extern volatile uint8_t Stav_exp;
extern volatile uint8_t flag_exp;

#define set_bit(Adress,Bit)((Adress) |= (1<<Bit))
#define clear_bit(Adress,Bit)((Adress) &= ~(1<<Bit))
#define toggle_bit(Adress,Bit)((Adress) ^= (1<<Bit))
#define check_bit(Adress,Bit)((Adress)&(1<<(Bit)))

extern void WR_exp(char adr_IO,char wr_data);
extern char RD_exp(char adr_IO);
extern void ini_TWI(void);
extern void ini_Int0(void);

#endif /* P_F_1_H_ */
```

```
// TWI State codes
// General TWI Master status codes
#define StK_TWI_START           0x08  // START has been transmitted
#define StK_TWI_REP_START       0x10  // Repeated START has been transmitted
#define StK_TWI_ARB_LOST        0x38  // Arbitration lost
// TWI Master Transmitter status codes
#define StK_TWI_MTX_ADR_ACK     0x18  // SLA+W has been tramsmitted and ACK received
#define StK_TWI_MTX_ADR_NACK    0x20  // SLA+W has been tramsmitted and NACK received
#define StK_TWI_MTX_DATA_ACK    0x28  // Data byte has been tramsmitted and ACK received
#define StK_TWI_MTX_DATA_NACK   0x30  // Data byte has been tramsmitted and NACK received
// TWI Master Receiver staus codes
#define StK_TWI_MRX_ADR_ACK     0x40  // SLA+R has been tramsmitted and ACK received
#define StK_TWI_MRX_ADR_NACK    0x48  // SLA+R has been tramsmitted and NACK received
#define StK_TWI_MRX_DATA_ACK    0x50  // Data byte has been received and ACK tramsmitted
#define StK_TWI_MRX_DATA_NACK   0x58  // Data byte has been received and NACK tramsmitted

// TWI Slave Transmitter staus codes
// ….
// TWI Slave Receiver staus codes
// …
// TWI Miscellaneous status codes
#define StK_TWI_NO_STATE        0xF8  // No relevant state information available; TWINT = “0”
#define StK_TWI_BUS_ERROR       0x00  // Bus error due to an illegal START or STOP condition

// ********************************************************************************
#define K_TWI_START             0xA5  // TWINT, /TWEA,  TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
#define K_TWI_R_START           0xA5  // TWINT, /TWEA,  TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
#define K_TWI_ADDRESS           0x85  // TWINT, /TWEA, /TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
#define K_TWI_READATAA          0xC5  // TWINT,  TWEA, /TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
#define K_TWI_READATNA          0x85  // TWINT, /TWEA, /TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE
#define K_TWI_STOP              0x94  // TWINT, /TWEA, /TWSTA,  TWSTO, /TWWC, TWEN, 0, /TWIE
```


26

Cv_x_TWI.c

```
...
while(1) {
  WR_exp(Adr_exp_8574,0xFF); _delay_ms(500);
  WR_exp(Adr_exp_8574,0x00); _delay_ms(500);

 Stav_exp = RD_exp(Adr_exp_8574);
 // ?? Test: Reset (VYPNI)
 if(n_check_bit(Stav_exp, 5))
   WR_exp(Adr_exp_8574,0b11111111);
 // ?? Test: Set (ZAPNI)
 if(n_check_bit(Stav_exp, 3))
   WR_exp(Adr_exp_8574,0b11111101);
}
...
```



p_f_1.c

```
/* ****************************************/
/* citanie z expandera PCF8574                */
/* mozne adresy 0x20,0x21,...0x26,0x27        */
/* zapis do  expandera PCF8574A               */
/* mozne adresy 0x70,0x71,...0x76,0x77        */
/* Bit0 = 1 (read)                            */
/**********************************************/
char RD_exp(char adr_IO) {
  while(check_bit(TWCR,TWIE)); //WDR();
  TWI_WRB[0] = (adr_IO<<1) | 1;// adr_obv + RD
  i_TWI = 0;
```

TWCR

7	6	5	4	3	2	1	0
TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
R/W	R/W	R/W	R/W	R	R/W	R	R/W

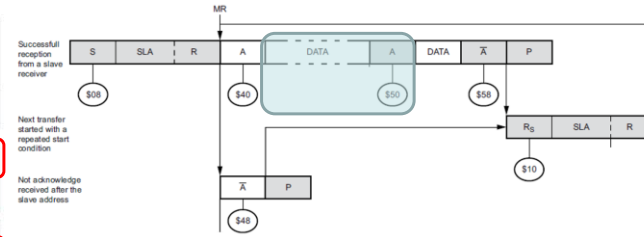
```
 //               1    0    1    0    0    1    0    1
 // K_TWI_START= 0xA5
 // TWINT, /TWEA, TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE

 TWCR = K_TWI_START;
 // ????
 while(check_bit(TWCR,TWIE)); //WDR();
 return(TWI_RDB[0]);
}
```

27

Teraz pridáme aj tlačítka SET(Zapni) a RESET (Vypni) ledku.

Status Codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response: To/from TWDR	To TWCR: STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset



p_f_1.c

```
ISR(TWI_vect) {
 switch(TWSR & 0xf8)
 {  ...
    break;
    case StK_TWI_MRX_DATA_NACK: // [0x58]
    // Data byte has been received and NACK tramsmitted
          TWI_RDB[0] = TWDR; // READ data
    case StK_TWI_MTX_ADR_NACK:
    case StK_TWI_MRX_ADR_NACK:
    case StK_TWI_MTX_DATA_ACK:
    case StK_TWI_MTX_DATA_NACK:
    case StK_TWI_BUS_ERROR:
    default:
      TWCR = K_TWI_STOP;
    break;
 }}
```

p_f_1.c

```
/* citanie z expandera PCF8574                                  */
char RD_exp(char adr_IO) {
  while(check_bit(TWCR,TWIE)); //WDR();
  TWI_WRB[0] = (adr_IO<<1) | 1;// adr_obv + RD
  i_TWI = 0;
  // K_TWI_START= 0xA5
  // TWINT, /TWEA, TWSTA, /TWSTO, /TWWC, TWEN, 0,  TWIE

  TWCR = K_TWI_START;
  // ????
  while(check_bit(TWCR,TWIE)); //WDR();
  return(TWI_RDB[0]);
}
```

28

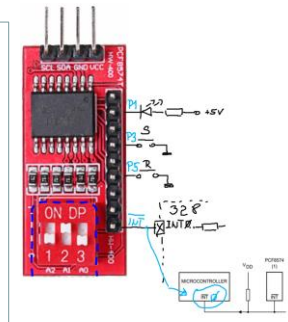
Teraz pridáme aj tlačítka SET(Zapni) a RESET (Vypni) ledku.

Cv_x_TWI.c

```
...
ini_TWI();
ini_Int0();
sei();

while(1) {
 WR_exp(Adr_exp_8574,0xFF);
 _delay_ms(500);
 WR_exp(Adr_exp_8574,0x00);
 _delay_ms(500);

 Stav_exp = RD_exp(Adr_exp_8574);
 // ?? Test: Reset (VYPNI)
 if(n_check_bit(Stav_exp, 5))
   WR_exp(Adr_exp_8574,0b11111111);
 // ?? Test: Set (ZAPNI)
 if(n_check_bit(Stav_exp, 3))
   WR_exp(Adr_exp_8574,0b11111101);

}
...
```

Cv_x_TWI.c

```
ISR(INT0_vect){
 flag_exp = 1;
}

void ini_Int0(void){
 DDRD &= ~(1<<DDB2);  // PD2 PORTD  input
 PORTD |= (1<<DDB2);  // INT0 pullup
 EICRA =0b00000010;   // dobezna hrana
 EIMSK |= (1 << INT0);// Turns on INT0
}
```

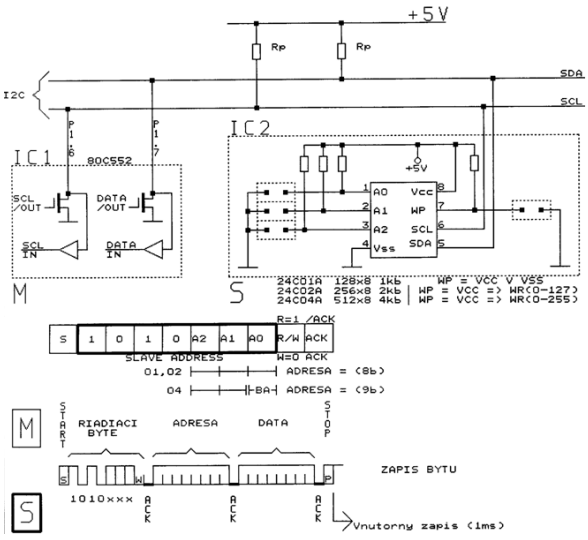


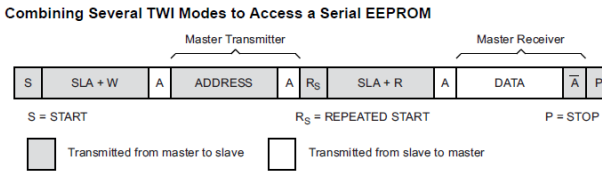

```
if(flag_exp){
  flag_exp = 0;
  Stav_exp = RD_exp(Adr_exp_8574);

  if(n_check_bit(Stav_exp, 5))
   WR_exp(Adr_exp_8574,0b11111111);
  if(n_check_bit(Stav_exp, 3))
   WR_exp(Adr_exp_8574,0b11111101);
 }
```

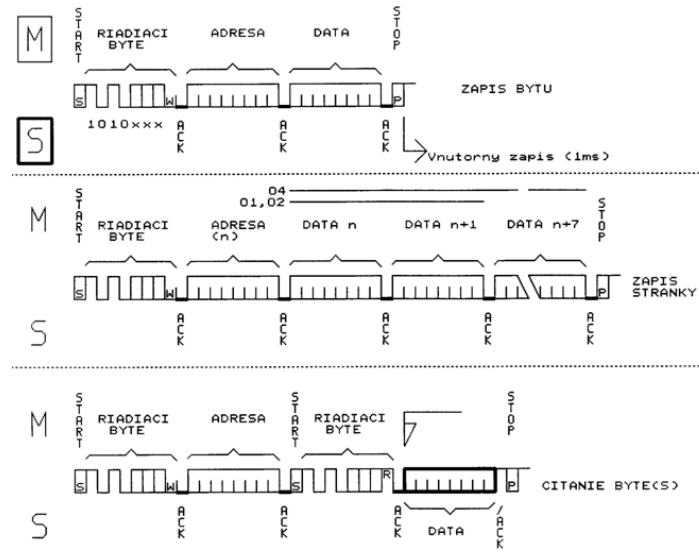
29

Teraz pridáme aj prerušenie od expandera.

I2C – pripojenie EEPROM







30

Piny:

1,2,3:Adresa IC (max 8)

4: V_{SS}

5: SDA

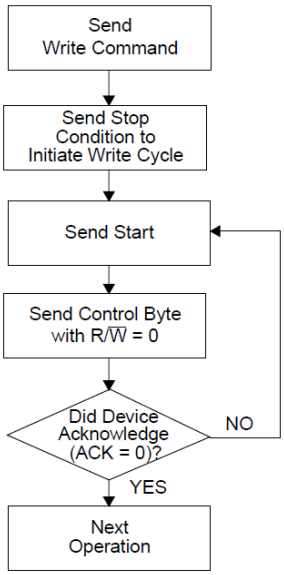
6:SCL

7:WP = log. 1

8: V_{CC}

EEPROM majú vnútorný AC, ktorý sa automaticky inkrementuje

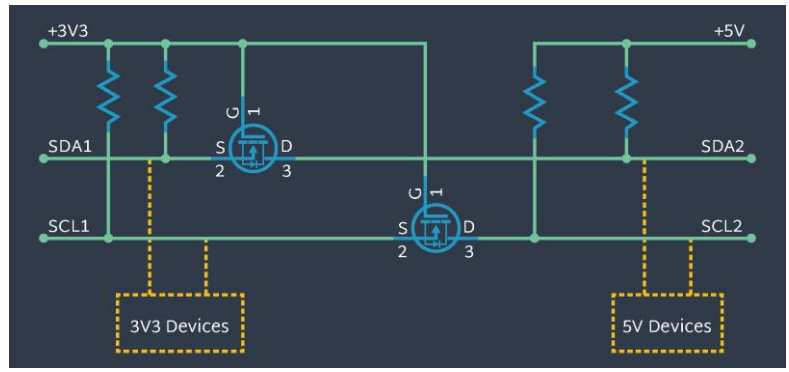
“ACKNOWLEDGE POLLING”

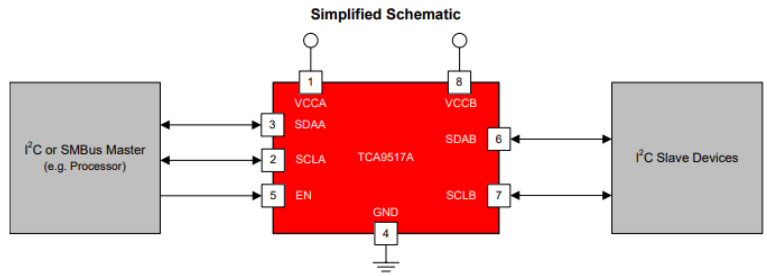


PART NUMBER	FUNCTION	I²C ADDRESS						
		A6	A5	A4	A3	A2	A1	A0
—	General call address	0	0	0	0	0	0	0
—	Reserved addresses	0	0	0	0	X	X	X
PCD3311/12	Tone generator DTMF/modem/musical	0	1	0	0	1	0	A
PCF8200	Voice synthesizer (male or female)	0	0	1	0	0	0	0
PCF8566	96-segment LCD driver 1:1–1:4 Mux	0	1	1	1	1	1	A
PCF8568	LCD row driver for dot matrix displays	0	1	1	1	1	0	A
PCF8569	Column driver for dot matrix displays	0	1	1	1	1	0	A
PCF8570/71	256 × 8, 128 × 8 static RAM	1	0	1	0	A	A	A
PCF8570C	256 × 8 static RAM	1	0	1	1	A	A	A
PCF8573	Clock/calendar	1	1	0	1	0	A	A
PCF8574	I²C bus to 8-bit bus converter	0	1	0	0	A	A	A
PCF8574A	I²C bus to 8-bit bus converter	0	1	1	1	A	A	A
PCF8576	160-segment LCD driver 1:1–1:4 Mux	0	1	1	1	0	0	A
PCF8577	64-segment LCD driver 1:1–1:2 Mux	0	1	1	1	0	1	0
PCF8577A	64-segment LCD driver 1:1–1:2 Mux	0	1	1	1	0	1	1
PCF8578	Row/column LCD dot-matrix driver	0	1	1	1	1	0	A
PCF8579	Row/column LCD dot-matrix driver	0	1	1	1	1	0	A
PCF8581	128-byte EEPROM	1	0	1	0	A	A	A
PCF8582	256 × 8 EEPROM	1	0	1	0	A	A	A
PCF8583	256 × 8 RAM with clock/calendar	1	0	1	0	0	0	A
PCF8591	4-channel, 8-bit A/D plus 8-bit D/A	1	0	0	1	A	A	A
PCF8594	512-byte EEPROM	1	0	1	0	A	A	A
SAA1064	4-digit LED driver	0	1	1	1	0	A	A
SAA1136	PCM-Audio indent-word interface	0	0	1	1	1	1	0
SAA1300	5-bit high current driver	0	1	0	0	0	A	A
SAA5243/45	Enhanced teletext circuit	0	0	1	0	0	0	1
SAA7191	S-VHS digital multistandard decoder “square pixel”	1	0	0	0	1	A	1
SAA7192	Digital color space converter	1	1	1	0	0	0	A
SAA7199	Digital encoder	1	0	1	1	0	0	0
SAA9020	Field memory controller	0	0	1	0	1	A	A
SAA9051	Digital multi-standard TV decoder	1	0	0	0	1	0	1
SAA9068	(PIPCO) Picture-in-picture controller	0	0	1	0	0	1	A
SAB3035/36/37	(CITAC) CPU interface for tuning and control	1	1	0	0	0	A	A
SAF1135	Data line decoder	0	0	1	0	0	A	A
TDA4670	Picture signal improvement circuit	1	0	0	0	1	0	0
TDA4680	Video processor	1	0	0	0	1	0	0
TDA8421	Hi-fi stereo audio processor	1	0	0	0	0	0	A
TDA8425	Audio processor w/loudspeaker channel	1	0	0	0	0	0	1
TDA8440	Switch for CTV receivers	1	0	0	1	A	A	A
TDA8442	Interface for color decoders	1	0	0	0	1	0	0
TDA8443	YUV/RGB interface circuit	1	1	0	1	A	A	A
TDA8444	Octuple 6-bit DAC	0	1	0	0	A	A	A
TDA8461	PAL/NTSC color decoder	1	0	0	0	1	0	A
TEA6100	FM/IF and tuning interface	1	1	0	0	0	0	1
TEA6300/6310T	Sound fader control circuit	1	0	0	0	0	0	0
TSA5511/12/14	PLL frequency synthesizer for TV	1	1	0	0	0	A	A
TSA6057	Radio tuning PLL frequency synthesizer	1	1	0	0	0	1	A
UMF1009	Frequency synthesizer	1	1	0	0	0	A	A
—	Reserved addresses	1	1	1	1	X	X	X

X = Don't care.
A = Can be connected high or low by the user.

31





32

Level-Shift: Existuje niekoľko prevedení. Jednoduchý most (len dva tranzistory) umožňuje komunikáciu medzi zariadeniami pracujúcimi s rôznymi úrovňami napájania. Tento mostík umožňuje obojsmerný prenos.