Phần Lý thuyết

1. Giới thiệu chung - Mô hình hệ VXL - Nguyên tắc hoạt động
2. Cấu trúc và hoạt động của vi xử lý 8085
3. Quá trình thực hiện 1 lệnh trong VXL 8085
4. Giới thiệu về vi điều khiển PIC
5. Bộ công cụ nạp chương trình, công cụ mô phỏng vi điều khiển
6. Bộ định thời Timer
7. Ghép nối với bộ hiển thị
8. ADC
9. Giao tiếp truyền dữ liệu
10. Ngắt
11. PWM

Nội dung buổi học

1. Master synchronous serial port module (MSSP)
   - Serial Peripheral Interface - SPI
   - Inter-Intergrated Circuit - I2C (Tương tự SPI nên SV tự tìm hiểu)
     - ✓Master mode
     - ✓Multi-master mode
     - ✓Slave mode
2. Universal synchronous receiver transmitter (USART)
   - Không đồng bộ - full duplex
   - Đồng bộ - master - half-duplex
   - Đồng bộ - slave - half-duplex

MSSP trong chế độ SPI

Sử dụng 3 chân tín hiệu

RC5/SDO - serial data out

RC4/SDI/SDA - serial data in

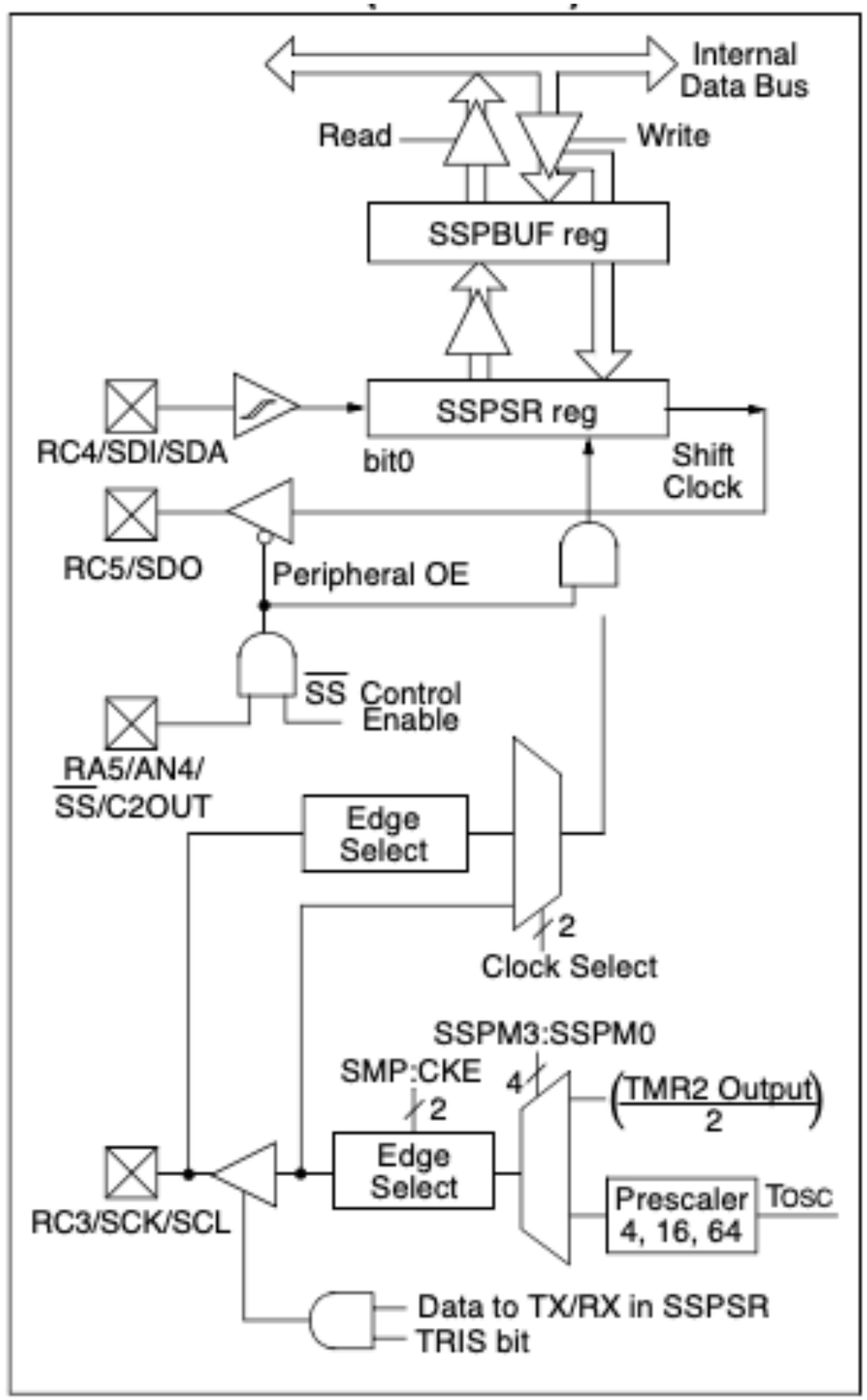
RC3/SCK/SCL - serial clock

Thêm 1 chân trong chế độ slave

RA5/AN4/$\overline{SS}$/C2OUT - slave select

Thanh ghi đệm SSPBUF

Thanh ghi đệm dịch SSPSR



Thanh ghi trong chế độ SPI

Thanh ghi trạng thái SSPTAT

SSPSTAT: MSSP STATUS REGISTER (SPI MODE) (ADDRESS 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

Bit 6 - CKE: SPI Clock Chọn bit

Khi CKP = 0

0 = Dữ liệu được truyền trên cạnh rơi của SCK

1 = Dữ liệu được truyền trên cạnh tăng của SCK

Khi CKP = 1

0 = Dữ liệu được truyền trên cạnh tăng của SCK

1 = Dữ liệu được truyền trên cạnh rơi của SCK

Bit 7 - SMP: bit mẫu

Chế độ SPI chính:

1 = Dữ liệu đâu vào được lấy mẫu ở cuối thời gian đâu ra dữ liệu

0 = Dữ liệu đâu vào được lấy mẫu ở giữa thời gian đâu ra dữ liệu

Chế độ SPI Slave:

SMP phải được xóa khi SPI được sử dụng trong chế độ Slave.

Bit 0 - BF: Bit trạng thái đệm đầy đủ

1 = Nhận hoàn tất, SSPBUF đã đầy.

0 = Nhận không hoàn thành, SSPBUF trống.

Các bit khác dùng trong chế độ I2C

Thanh ghi trong chế độ SPI

Thanh ghi điều khiển SSPCON1

SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE) (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

Bit 3: 0 - SSPM3: SSPM0: Chế độ cổng nối tiếp đồng bộ chính Chọn bit

Các bit này được sử dụng để chọn chế độ Master hoặc Slave và cũng có thể chọn đồng hồ.

SSPM3: SSPM0	Chế độ	Trạng thái pin đồng hồ / SS
0000	Chế độ SPI chính	Fosc / 4
0001	Chế độ SPI chính	Fosc / 16
0010	Chế độ SPI chính	Fosc / 64
0011	Chế độ SPI chính	Đầu ra TMR2 / 2
0100	Chế độ Slave SPI	Đã bật SS
0101	Chế độ Slave SPI	SS bị tắt, được sử dụng làm pin I / O

Thanh ghi trong chế độ SPI

Thanh ghi điều khiển SSPCON1

SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE) (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

Bit 7 - WCOL: Ghi va chạm Phát hiện bit (Chỉ chế độ phát)

1 = Thanh ghi SSPBUF được viết trong khi nó vẫn đang truyền từ trước

(phải được xóa thông qua phần mềm)

0 = Không có va chạm

Bit 4 - CKP: Đồng hồ phân cực Chọn bit

1 = Trạng thái nhàn rỗi cho đồng hồ là mức cao

0 = Trạng thái nhàn rỗi cho đồng hồ là mức thấp

Bit 5 - SSPEN: Cổng nối tiếp đồng bộ chính cho phép bit

1 = Bật cổng nối tiếp và cấu hình SCK, SDO, SDI và SS làm chân cổng nối tiếp.

0 = Tắt cổng nối tiếp và cấu hình các chân này làm chân cổng I / O.

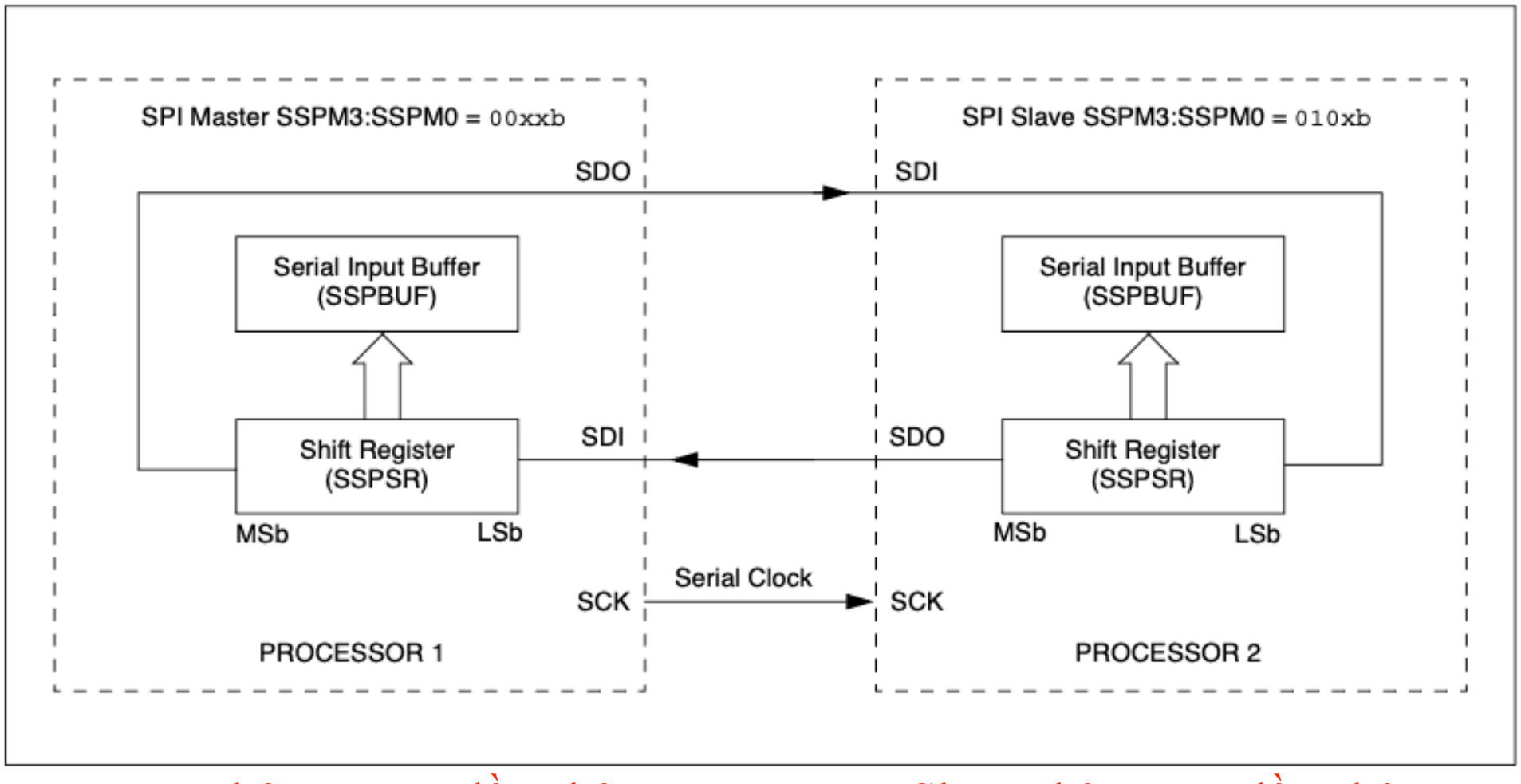
Bit 6 - SSPOV: Nhận bit chỉ báo tràn

Chế độ SPI Slave:

1 = Một byte mới được nhận trong khi thanh ghi SSPBUF vẫn giữ dữ liệu trước đó. Trong trường hợp tràn, dữ liệu trong SSPSR bị mất.

0 = Không tràn

Kết nối trong chế độ SPI



Master phát ra xung đồng bộ Slave nhận xung đồng bộ

SSPSR là thanh ghi thay đổi được sử dụng để chuyển dữ liệu vào/ra. Thanh ghi này không cho phép người sử dụng truy cập

SSPBUF là thanh ghi đệm mà các byte dữ liệu được ghi vào hoặc đọc từ đó

Hoạt động

Nhận dữ liệu

Dữ liệu sẽ được dịch vào/ra qua thanh ghi **SSPSR** Bit MSB sẽ được dịch trước

Khi dữ liệu trong SSPBUF đã đầy, bit BF (thanh ghi SSPSTAT) và SSPIF (thanh ghi PIR1) được set

BF sẽ được tự động reset về 0 khi dữ liệu trong thanh ghi SSPBUF được đọc vào

Nên đọc dữ liệu từ thanh ghi SSPBUF trước khi nhận byte dữ liệu tiếp theo

Truyền dữ liệu

Dữ liệu cần truyền được đưa vào **SSPBUF** đồng thời đưa vào **SSPSR**

Cờ hiệu BF được set

Cho phép tín hiệu SS

Ngắt sẽ xảy ra khi quá trình dịch dữ liệu hoàn tất

Cấu hình cho VĐK Master

1. Định chế độ master cho VĐK, set tốc độ xung nhịp - SSPM3:SSPM0 trong SSPCON1
2. Cho phép truyền nối tiếp bằng cách set bit SSPEN trong SSPCON1
3. Nạp CKP và CKE tương ứng để cấu hình chốt dữ liệu - tra bit 6 của SSPSTAT
4. Cấu hình SMP (thường là xoá SMP)
5. Cấu hình các chân SDO là output, SDI là input, SCK là input, SS là input
6. Cho phép ngắt SPI bằng cách set các bit SSPIE, PEIE và GIE

Code cho master

```
void SPI_Master_Init()
{
SSPM0 = 0;
SSPM1 = 0;
SSPM2 = 0;
SSPM3 = 0;
SSPEN = 1;
CKP = 0;
CKE = 0;
SMP = 0;
TRISC5 = 0; // SDO -> Output
TRISC4 = 1; // SDI -> Input
TRISC3 = 0; // SCK -> Output
// If Interrupts Are Needed, Un-comment
// SSPIE = 1; PEIE = 1; GIE = 1;
}
void SPI_Write(uint8_t Data)
{
SSPBUF = Data; // Transfer The Data To The
// while(!BF); // Un-comment it if you're
// The Above While Loop Protects Against
// The Previous Transmission Ends
}
```

1 2 3 4 5 6

Cấu hình cho VĐK Slave

1 Định chế độ slave cho VĐK + cho phép SS - SSPM3:SSPM0 trong SSPCON1

2 Cho phép truyền nối tiếp bằng cách set bit SSPEN trong SSPCON1

3 Nạp CKP và CKE tương ứng để cấu hình chốt dữ liệu - tra bit 6 của SSPSTAT (giống master)

4 Cấu hình SMP

5 Cấu hình các chân SDO là output, SDI là input, SCK là output

6 Cho phép ngắt SPI bằng cách set các bit SSPIE, PEIE và GIE

Cấu hình cho VĐK Slave

```
void SPI_Slave_Init()
{
SSPM0 = 0;
SSPM1 = 0;
SSPM2 = 1;
SSPM3 = 0;
SSPEN = 1;
CKP = 0;
CKE = 0;
SMP = 0;
TRISC5 = 0; // SDO -> Output
TRISC4 = 1; // SDI -> Input
TRISC3 = 1; // SCK -> Intput
PCFG3 = 0; // Set SS To Be Digital IO
PCFG2 = 1;
PCFG1 = 0;
PCFG0 = 0;
TRISA5 = 1; // SS -> Input
SSPIE = 1;
PEIE = 1;
GIE = 1;}
```

1 2 3 4 5 6

```
uint8_t SPI_Read(){
uint8_t Data;
if(BF) // Kiểm tra xem có DL mới
{
Data = SSPBUF; // Đọc Buffer
BF = 0;
SSPIF = 0; // Xoá cờ ngắt
SSPOV = 0; // Xoá cờ tràn
return Data;
} }
void __interrupt ISR(void)
{
if(SSPIF)
{
Data = SSPBUF; // Đọc Buffer
SSPIF = 0; // Xoá cờ ngắt
} }
```

Sử dụng SPI trong CCD

setup_spi(mode) Thiết lập giao tiếp SPI

Thiết lập giao tiếp	SPI_MASTER	SPI_SLAVE	SPI_SS_DISABLE
Thiết lập kích hoạt	SPI_L_TO_H	SPI_H_TO_L	
Xác định tần số xung	SPI_CLK_DIV_4	SPI_CLK_DIV_16	SPI_CLK_DIV_64
	SPI_CLK_T2		

spi_read(data) Hàm trả về giá trị 8 bit: value = spi_read()

Data có thể có hoặc không, nếu có thì là số 8 bit

Hàm chỉ dùng cho SPI cứng

spi_write(value) Hàm gửi một giá trị value tới SPI, value là giá trị 8 bit

Hàm không trả về giá trị, value là giá trị 8 bit

spi_data_is_in() Hàm kiểm tra xem giá trị nhận về SPI đã đủ 1 byte hay chưa

Hàm trả về 1 nếu nhận đủ 1 byte, 0 nếu chưa nhận đủ

Giao tiếp USART

Các chế độ trong giao tiếp USART

- Đồng bộ - full duplex
- Đồng bộ - Master - half duplex
- Đồng bộ. - Slave - half duplex

Các chân sử dụng

- RC6/TX/CK
- RC7/RX/DT

Giao tiếp USART

Thanh ghi trạng thái và điều khiển truyền dữ liệu

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

CSRC: Clock Source Select bit

Asynchronous mode:
Don't care.

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

TX9: 9-bit Transmit Enable bit

1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

TXEN: Transmit Enable bit

1 = Transmit enabled
0 = Transmit disabled

SYNC: USART Mode Select bit

1 = Synchronous mode
0 = Asynchronous mode

Unimplemented: Read as '0'

BRGH: High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode.

TRMT: Transmit Shift Register Status bit

1 = TSR empty
0 = TSR full

TX9D: 9th bit of Transmit Data, can be Parity bit

Giao tiếp USART

Thanh ghi trạng thái và điều khiển nhận dữ liệu

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

SPEN: Serial Port Enable bit

1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)

0 = Serial port disabled

RX9: 9-bit Receive Enable bit

1 = Selects 9-bit reception

0 = Selects 8-bit reception

SREN: Single Receive Enable bit

Asynchronous mode:

Don't care.

Synchronous mode – Master:

1 = Enables single receive

0 = Disables single receive

This bit is cleared after reception is complete.

Synchronous mode – Slave:

Don't care.

CREN: Continuous Receive Enable bit

Asynchronous mode:

1 = Enables continuous receive

0 = Disables continuous receive

Synchronous mode:

1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)

0 = Disables continuous receive

ADDEN: Address Detect Enable bit

Asynchronous mode 9-bit (RX9 = 1):

1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set

0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit

FERR: Framing Error bit

1 = Framing error (can be updated by reading RCREG register and receive next valid byte)

0 = No framing error

OERR: Overrun Error bit

1 = Overrun error (can be cleared by clearing bit CREN)

0 = No overrun error

Xác lập tốc độ truyền

PIC 16F877A tích hợp sẵn bộ tạo tốc độ baud Baud Rate Generator BRG

Chế độ không đồng bộ → BRG được điều khiển bằng BRGH

BRGH: High Baud Rate Select bit

Asynchronous mode:

1 = High speed

0 = Low speed

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64 (X + 1))	Baud Rate = FOSC/(16 (X + 1))
1	(Synchronous) Baud Rate = FOSC/(4 (X + 1))	N/A

Legend: X = value in SPBRG (0 to 255)

↓

Đây là thanh ghi chứa giá trị để tính tốc độ truyền

Chế độ đồng bộ → BRG không được sử dụng

Xác lập tốc độ truyền (BRGH=0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	-	255	0.225	-	255
LOW	62.500	-	0	57.6	-	0

Xác lập tốc độ truyền (BRGH=1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

Giao tiếp không đồng bộ

Truyền dữ liệu

1. Tạo xung truyền baud bằng cách đưa giá trị cần thiết vào RSBRG và bit điều khiển mức tốc độ baud BRGH
2. Cho phép cổng giao diện nối tiếp không đồng bộ - clear SYNC và set PSEN
3. Set bit TXIE nêu cần sử dụng ngắt truyên
4. Set bit TX9 nếu định dạng cần truyền 9 bit
5. Set bit TXEN để cho phép truyền dữ liệu (lúc này TXIF cũng set)
6. Nếu định dạng 9 bit dữ liệu, đưa bit dữ liệu thứ 9 vào TXD9
7. Đưa 8 bit dữ liệu cần truyền vào thanh ghi TXREG
8. Nếu sử dụng ngắt truyền, kiểm tra GIE và PEIE

Giao tiếp không đồng bộ

Nhận dữ liệu

1. Thiết lập tốc độ baud
2. Cho phép cổng giao diện nối tiếp không đồng bộ - clear SYNC và set PSEN
3. Set bit RCIE nếu cần sử dụng ngắt truyền
4. Set bit RX9 nêu định dạng dữ liệu nhận là 9 bit
5. Cho phép nhận dữ liệu bằng cách set bit CREN
6. Sau khi nhận dữ liệu, bit RCIF được set và ngắt được kích hoạt (nếu RCIE được set)
7. Đọc giá trị RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra quá trình nhận có lỗi không
8. Đọc 8 bit dữ liệu từ thanh ghi RCREG
9. Nếu có lỗi, xoá lỗi bằng cách xoá CREN
10. Nếu sử dụng ngắt nhận, set bit GIE và PEIE

Giao tiếp đồng bộ - Master

Truyền dữ liệu

1. Tạo xung truyền baud bằng cách đưa giá trị cần thiết vào RSBRG và bit điều khiển mức tốc độ baud BRGH
2. Cho phép cổng giao diện nối tiếp không đồng bộ - set SYNC, PSEN và CSRC
3. Set bit TXIE nêu cần sử dụng ngắt truyên
4. Set bit TX9 nếu định dạng cần truyền 9 bit
5. Set bit TXEN để cho phép truyền dữ liệu
6. Nếu định dạng 9 bit dữ liệu, đưa bit dữ liệu thứ 9 vào TXD9
7. Đưa 8 bit dữ liệu cần truyền vào thanh ghi TXREG
8. Nếu sử dụng ngắt truyền, kiểm tra GIE và PEIE

Giao tiếp đồng bộ - Master

Nhận dữ liệu

1.Thiết lập tốc độ baud, đưa giá trị vào SPBRG và BRGH
2.Cho phép cổng giao diện nối tiếp không đồng bộ (SYNC, PSEN và CSRC)
3.Clear bit CREN và SREN
4.Set bit RCIE nếu cần sử dụng ngắt truyền
5.Set bit RX9 nếu định dạng dữ liệu nhận là 9 bit
6.Nếu chỉ nhận 1 word dữ liệu, set SREN, nếu nhận 1 chuỗi dữ liệu, set CREN
7.Sau khi nhận dữ liệu, bit RCIF được set và ngắt được kích hoạt (nếu RCIE được set)
8.Đọc giá trị RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra quá trình nhận có lỗi không
9.Đọc 8 bit dữ liệu từ thanh ghi RCREG
10.Nếu có lỗi, xoá lỗi bằng cách xoá CREN
11.Nếu sử dụng ngắt nhận, set bit GIE và PEIE

Giao tiếp đồng bộ - Slave

Truyền dữ liệu

1.Set bit SYNC, PSEN, và clear CSRC
2.Clear bit CREN và SREN
3.Set bit TXIE nếu cần sử dụng ngắt truyền
4.Set bit TX9 nếu định dạng cần truyền 9 bit
5.Set bit TXEN để cho phép truyền dữ liệu
6.Nếu định dạng 9 bit dữ liệu, đưa bit dữ liệu thứ 9 vào TXD9
7.Đưa 8 bit dữ liệu cần truyền vào thanh ghi TXREG
8.Nếu sử dụng ngắt truyền, kiểm tra GIE và PEIE

Giao tiếp đồng bộ - Slave

Nhận dữ liệu

1. Cho phép cổng giao diện nối tiếp không đồng bộ (SYNC, PSEN và CSRC)
2. Set bit RCIE nếu cần sử dụng ngắt truyền
3. Set bit RX9 nếu định dạng dữ liệu nhận là 9 bit
4. Set bit CREN để quá trình nhận dữ liệu bắt đầu
5. Sau khi nhận dữ liệu, bit RCIF được set và ngắt được kích hoạt (nếu RCIE được set)
6. Đọc giá trị RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra quá trình nhận có lỗi không
7. Đọc 8 bit dữ liệu từ thanh ghi RCREG
8. Nếu có lỗi, xoá lỗi bằng cách xoá CREN
9. Nếu sử dụng ngắt nhận, set bit GIE và PEIE

Cài đặt chương trình bên truyền

```
void UART_TX_Init(void) // Khởi tạo UART
{
BRGH = 1; // Set tốc độ truyền là High speed
SPBRG = 25; // Tốc độ Baud Rate là 9600 bps
//Khởi tạo port truyền đồng bộ
SYNC = 0;
SPEN = 1;
//Set các chân RX-TX vào chế độ UART
TRISC6 = 1;  // Trong chế độ UART thì 2 chân PortC 6 và 7
TRISC7 = 1;  // được set lên 1
TXEN = 1; // Cho phép truyền UART
}
void UART_Write(uint8_t data)
{
while(!TRMT); // TRMT = 1 là truyền xong
TXREG = data;
}
uint8_t UART_TX_Empty()
{
// Kiểm tra bộ đệm truyền có rỗng không
return TRMT;
}
```

Cài đặt chương trình bên nhận

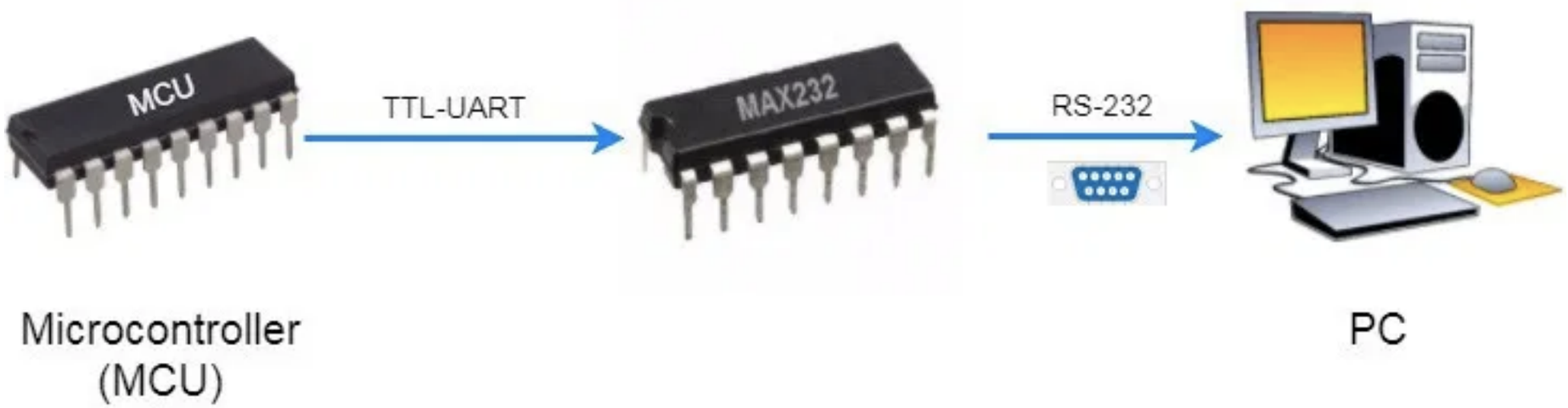
```
void UART_RX_Init()
{
BRGH = 1;    // High-Speed Baud Rate
SPBRG = 25;  // Tốc độ Baud Rate là 9600 bps
// Cho phép chế độ truyền đồng bộ
SYNC = 0;
SPEN = 1;
//Set các chân RX-TX vào chế độ UART
TRISC6 = 1;  // Trong chế độ UART thì 2 chân PortC 6 và 7
TRISC7 = 1;  // được set lên 1
// Cho phép ngắt để nhận dữ liệu
RCIE = 1;    // Bit cho phép nhận ngắt UART
PEIE = 1;    // Bit cho phép ngắt ngoại vi
GIE = 1;     // Bit cho phép ngắt toàn cục
CREN = 1;    // Bit cho phép nhận liên tục
}
```

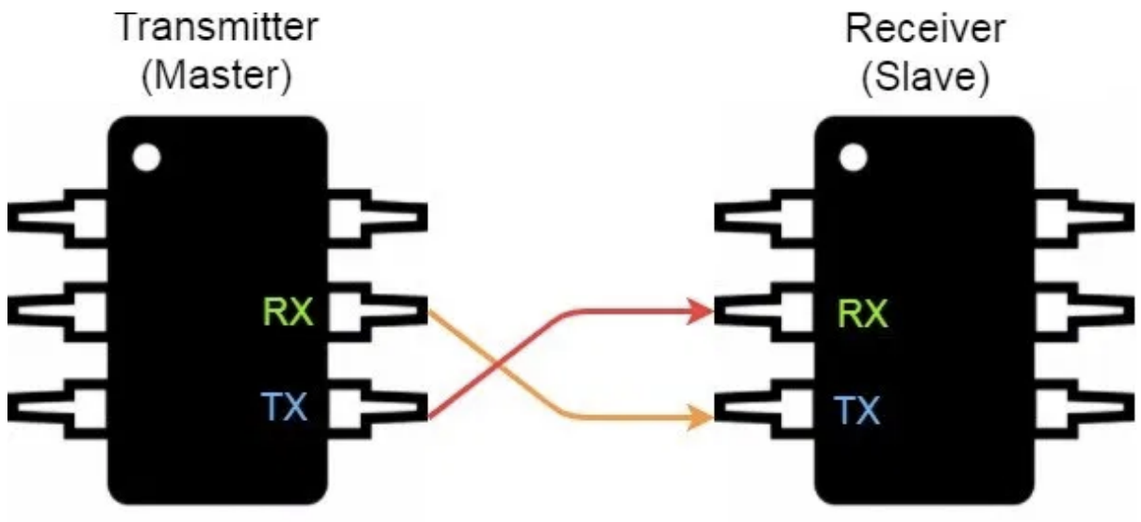
Cài đặt chương trình bên nhận

```
uint8_t UART_Read() // Đọc dữ liệu truyền
{
while(!RCIF);// Kiểm tra RCIF, nếu là 0 thì vẫn đang truyền-nhận
return RCREG;// Truyền xong thì dữ liệu nhận được trong RCREG
}
```

```
// Có thể sử dụng ngắt để nhận dữ liệu bằng chương trình ngắt
void interrupt ISR (void)
{
if (RCIF == 1)
{
Destination = RCREG;  // Đọc dữ liệu nhận được từ thanh ghi
RCIF = 0;             // Xoá cờ
}
}
```

Lưu ý khi truyền USART





USART sử dụng CCS

#use rs232(option) thiết lập đường truyền

baud=x	Tốc độ truyền x baud	sync_slave	truyền đồng bộ slave
xmit=pin	Set chân truyền	sync_master	truyền đồng bộ master
rcv	Set chân nhận	sync_master_cont	truyền đồng bộ master liên tục
uart1	Set các chân xmit, rcv của bộ uart thứ nhất	uart2	Set các chân xmit, rcv của bộ uart thứ hai
uart3	Set các chân xmit, rcv của bộ uart thứ ba	uart4	Set các chân xmit, rcv của bộ uart thứ tư

setup_uart(baud) thiết lập đường truyền với tốc độ là baud

getchar() đọc 1 kí tự từ đường truyền

putchar() đưa 1 kí tự lên đường truyền

gets() đọc 1 chuỗi kí tự từ đường truyền

puts() đưa 1 chuỗi kí tự lên đường truyền

Các hàm khác của CCS có thể tham khảo trong tài liệu gửi kèm