INFO2 : INFORMATIQUE EMBARQUEE
TD7 : Mise en œuvre de l'UART du PIC18F4520

Remarque : la plupart des figures et tableaux proviennent de **`39631E.pdf`** (*datasheet* du PIC18F4520), chapitre 18. Tout ce qui est dit ici est aussi valable pour l'UART1 d'un PIC18F87J50.

1 Présentation et câblage

Le PIC18F4520 possède deux liaisons série : une synchrone et une asynchrone. Ces deux liaisons série sont permises par le module périphérique appelé EUSART (*Enhanced Universal Synchronous / Asynchronous Receiver Transmitter*, c'est à dire « Émetteur Récepteur Universel Synchrone Asynchrone Amélioré[1] »).

Nous ne nous intéressons ici qu'à la **liaison série asynchrone**, c'est-à-dire à la partie UART de l'EUSART.

L'UART repose sur des registres à décalages qui permettront d'envoyer et de recevoir les données. Ils sont au nombre de deux (un pour l'émission, un pour la réception), ce qui autorise théoriquement le PIC à envoyer et recevoir des données simultanément (fonctionnement *full duplex*).

Remarque : Le PIC18F4520 est également capable de mettre en œuvre deux **bus série synchrones** : SPI et I2C. Ces bus série sont gérées par le périphérique MSSP et ne seront pas étudiés ici.

1.1 Programmation

- Comme nous l'avons vu en cours, pour utiliser la liaison série asynchrone il est nécessaire de configurer la vitesse de transmission, le nombre de bits de données, la parité et le nombre de bits de *stop*. Il faudra donc mettre la bonne valeur dans les **registres de configuration** de l'EUSART.
- Des **bits d'état** nous permettront de savoir, entre autres choses, si la transmission d'une donnée est en cours ou bien si une donnée a été réceptionnée.
- Enfin, nous aurons **un registre pour écrire les données à envoyer et un autre pour lire les données reçues**.

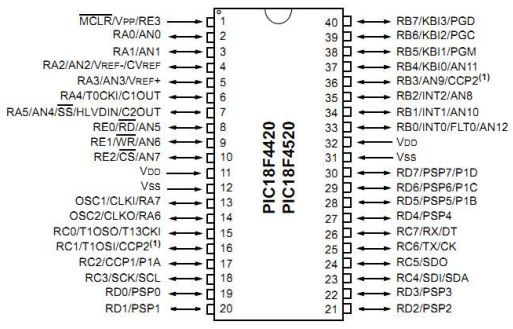
Remarque : en général les données échangées sont au format 8 bits. Associées au standard ASCII, elles correspondent donc à des caractères.

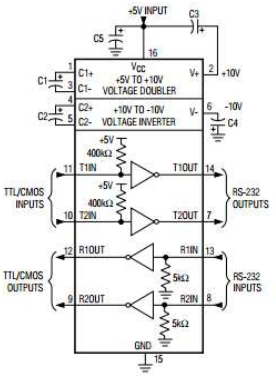
Remarque : Nous nous passerons de contrôles de flux matériel ou logiciel, techniques à implémenter soi-même sur un microcontrôleur, ce qui complique un peu la tâche.

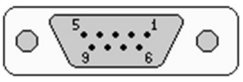
1.2 Aspects matériels

Au niveau des broches, le PIC18F4520 dispose de RC6/TX pour l'envoi et de RC7/RX pour la réception.

1. Proposez un câblage permettant de faire communiquer par liaison série asynchrone un PIC18F4520 avec un PC équipé d'un port COM sous la forme d'un connecteur DB9. Le circuit du milieu est un MAX232.







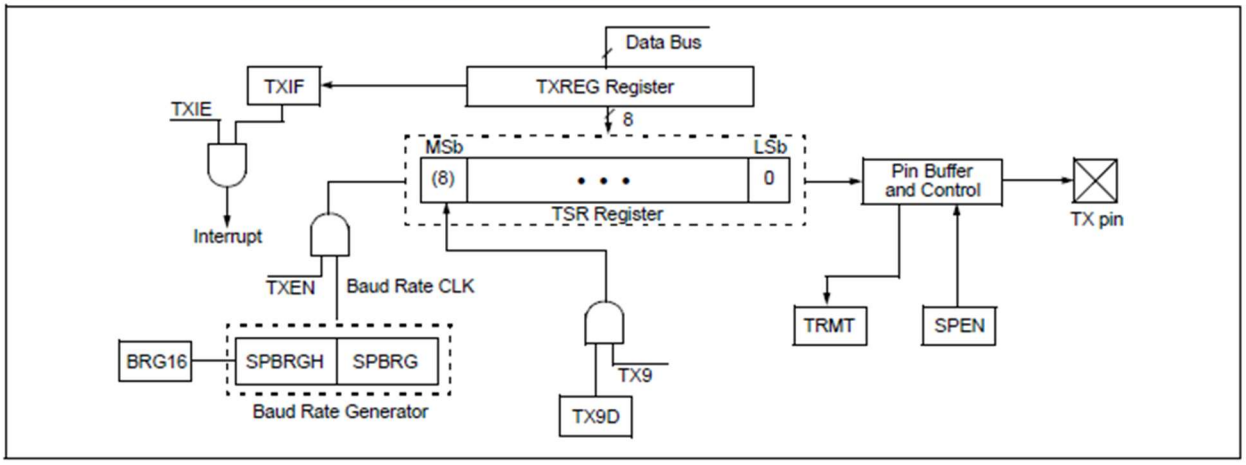
Broche n°	Nom
1	DCD
2	Rx
3	Tx
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

[1] « Amélioré » par rapport au même module périphérique de la famille PIC16F.

2 Fonctionnement de la liaison série en transmission

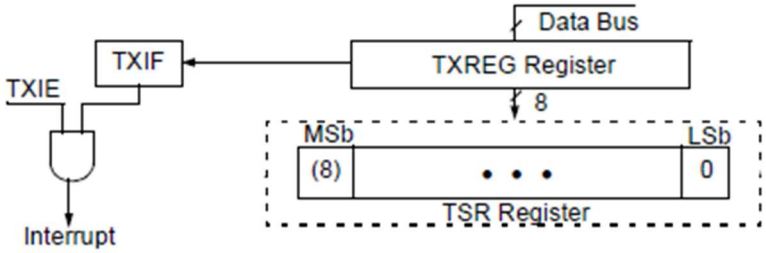
La FIGURE 18-3 nous indique le fonctionnement de l'EUSART en transmission.

FIGURE 18-3: EUSART TRANSMIT BLOCK DIAGRAM



Le module de transmission est basé sur un registre à décalage (TSR ou *Transmit Shift Register*).

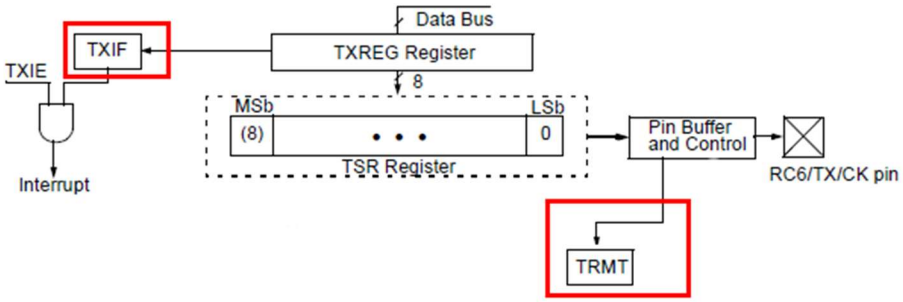
On charge la donnée à transmettre sur 8 bits à l'aide du registre TXREG :



L'écriture par le PIC de TXREG dans TSR met à '1' le drapeau TXIF. Celui-ci est capable de générer une interruption si le bit TXIE est à '1'. Le registre à décalage ajoute tout seul le **bit de start** et le **bit de stop**.

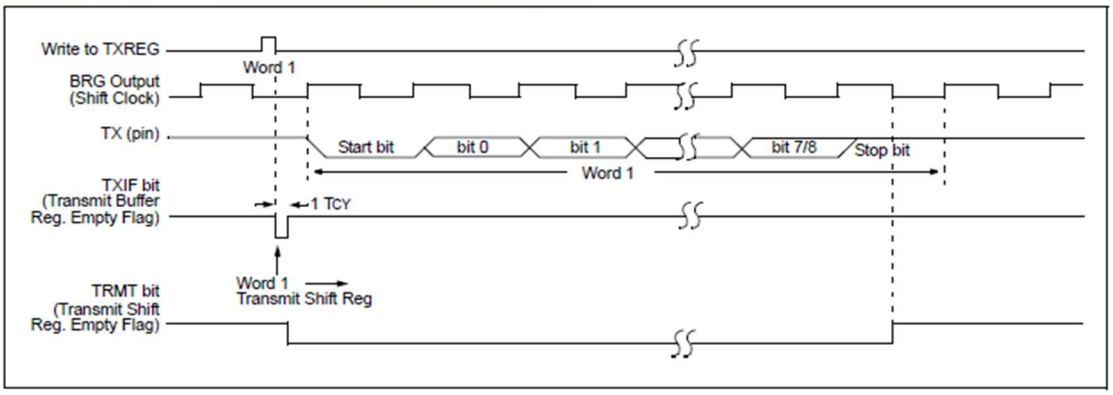
Remarque : Le fait que le drapeau TXIF passe à '1' n'indique pas que la donnée a été transmise, mais uniquement le fait que celle-ci a été déposée dans le registre TXREG.

Le drapeau « TRMT » indique quand le registre à décalage est vide. C'est à dire quand la donnée est complètement envoyée. Il est parfois plus judicieux de tester le bit TRMT que le bit TXIF.

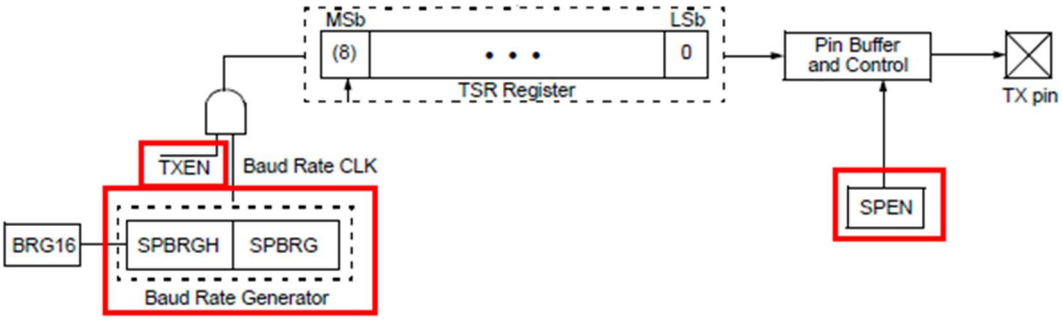


La FIGURE 18-4, nous donne les chronogrammes des bits TRMT et TXIF :

FIGURE 18-4: ASYNCHRONOUS TRANSMISSION



Pour envoyer les données, il faut configurer l'horloge de transmission. Ceci se fait à l'aide du registre SPBRG (composé de 2 registres de 8 bits SPBRGH:SPBRG) qui divise la fréquence de l'horloge interne pour obtenir la bonne vitesse de transmission. La validation de cette horloge se fait par TXEN. L'horloge est la même pour les modules de transmission et de réception.



Les bits sérialisés sont envoyés vers la broche de sortie RC6/TX. Cette patte peut servir de broche d'entrée/sortie TOR standard ou servir comme broche de transmission de la liaison série. La configuration entre ces deux modes de fonctionnement se fait grâce au bit SPEN (respectivement 1 ou 0).

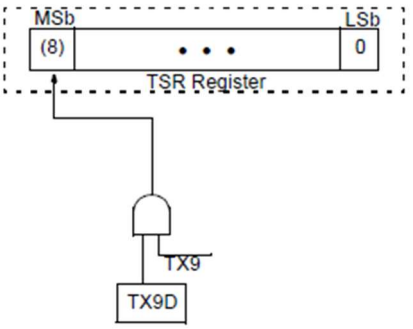
Remarque : Lorsque le bit SPEN est égal à '1', les pattes RC6/TX et RC7/RX sont configurés automatiquement, respectivement en sortie et en entrée :

The pins of the Enhanced USART are multiplexed with PORTC. In order to configure RC6/TX/CK and RC7/RX/DT as an EUSART:

- bit SPEN (RCSTA<7>) must be set (= 1)
- bit TRISC<7> must be set (= 1)
- bit TRISC<6> must be set (= 1)

Note:	The EUSART control will automatically reconfigure the pin from input to output as needed.

Il est possible d'envoyer un neuvième bit avant le bit de *stop* (pour une parité, pour un deuxième bit de *stop* ou un neuvième bit de donnée). Il faut alors valider TX9 et inscrire la valeur du neuvième bit dans TX9D avant de procéder à une émission.



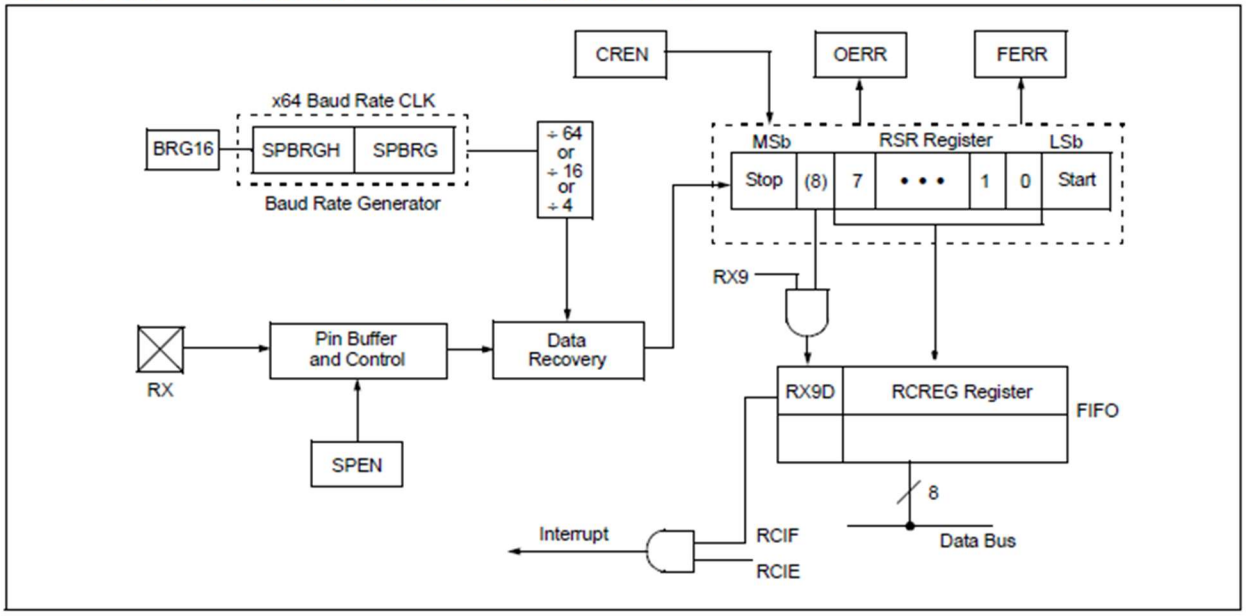
3 Fonctionnement de la liaison série en réception

La FIGURE 18-6 nous indique le fonctionnement de l'EUSART en réception.

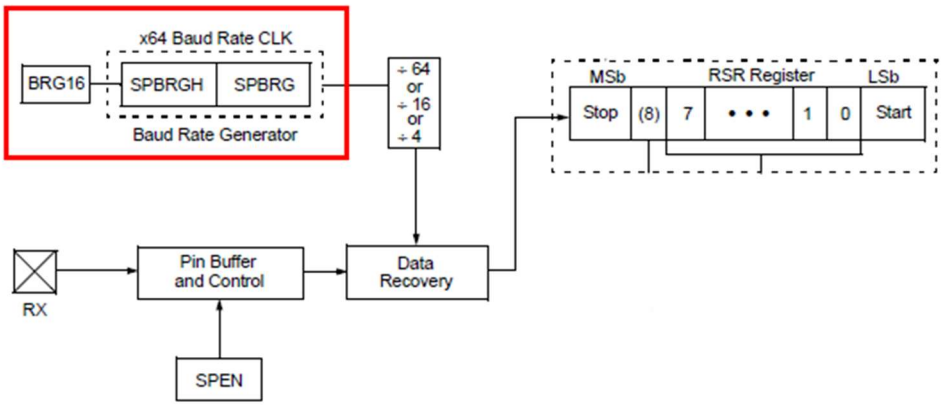
Le module de réception est relativement similaire au module de transmission.

Il est basé sur un registre à décalage (RSR ou *Reception Shift Register*) de 11 bits (un bit de *start*, 8 de données, un éventuel neuvième bit et un bit de *stop*).

FIGURE 18-6: EUSART RECEIVE BLOCK DIAGRAM

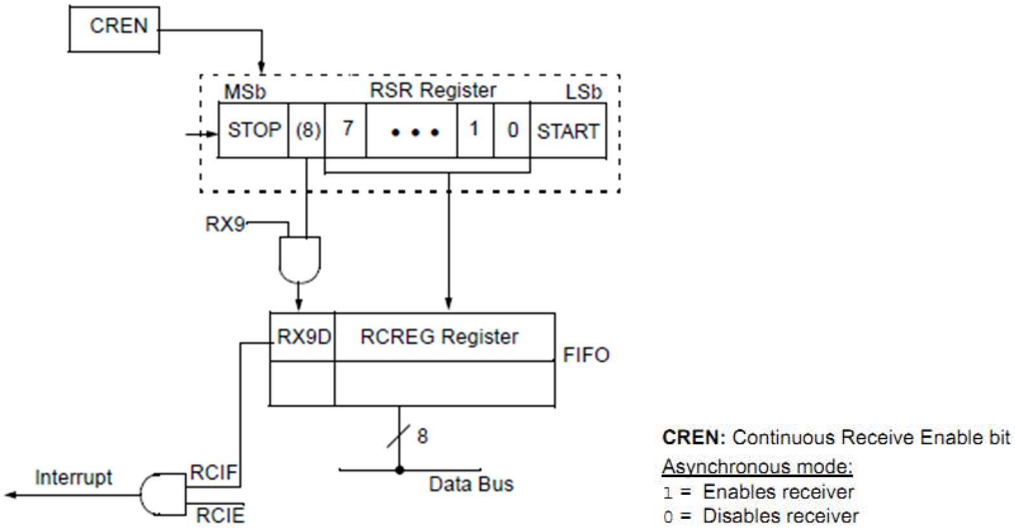


Tout comme le module d'émission, il dispose de l'horloge (SPBRGH:SPBRG).



L'EUSART scrute en permanence le signal qui arrive sur la patte RC7/RX. Il faut donc là aussi indiquer que cette patte sert à la liaison série grâce à SPEN.

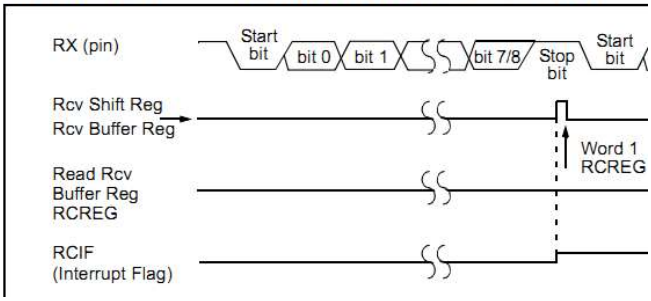
Remarque : Le module de réception ne sera validé que lorsque le bit CREN sera correctement configuré :



Une fois la donnée reçue correctement (le bit de *start* et le bit de *stop* ont été retirés), le PIC stocke les 8 bits de données reçus dans le registre RCREG. Si on lui a validé un neuvième bit en réception à l'aide de RX9, il stockera sa valeur dans RX9D.

En outre, la réception d'une donnée met à 1 le bit RCIF. On peut alors générer une interruption si le bit RCIE est à '1'. Cf. FIGURE 18-7 ci-dessous :

FIGURE 18-7: ASYNCHRONOUS RECEPTION



Remarque : Quand RCREG est lu, alors RCIF retombe à 0 (non représenté sur la FIGURE 18-7).

Remarque : Si on prévoit d'exploiter plusieurs le caractère reçu, il faut sauver RCREG dans une variable et exploiter la variable car celui-ci peut changer au cours des exploitations successives.

Le module de réception peut aussi signaler des erreurs si l'échange s'est mal passé. Il ajuste notamment les bits OERR (*Overrun ERRor*) et FERR (*Framing ERRor*). Nous n'exploiterons pas ces bits dans ce TD.

4 Les registres en détail

4.1 Présentation

Microchip nous propose une liste de registres pour configurer l'EUSART en transmission et une liste pour configurer l'EUSART en réception (seules les cases non grisées sont à considérer).

En regardant bien, les mêmes registres sont utilisés en transmission ou en réception. La seule différence est bien-sûr l'utilisation du registre TXREG ou du registre RCREG.

TABLE 18-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
TXREG	EUSART Transmit Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

Legend: — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.
Note 1: Reserved in 28-pin devices; always maintain these bits clear.

TABLE 18-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
RCREG	EUSART Receive Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

Legend: — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
Note 1: Reserved in 28-pin devices; always maintain these bits clear.

4.2 Registres TXREG et RCREG

La donnée 8 bits à transmettre est à déposer dans le registre TXREG.

Lorsqu'une donnée 8 bits est reçue, elle est disponible dans le registre RCREG.

4.3 Registres INTCON et PIE1

Ces registres sont utilisés pour gérer l'UART en interruption :

- en transmission, on utilisera les bits GIE/GIEH, PEIE/GIEL et TXIE ;
- en réception, on utilisera les bits GIE/GIEH, PEIE/GIEL et RCIE.

4.4 Registre PIR1

Ce registre contient les deux *flags* (drapeaux) permettant de :

- détecter qu'une donnée est reçue : bit RCIF ;
- détecter qu'une donnée a été placée dans le registre de transmission : bit TXIF.

Rappel : Il existe un *flag* pour indiquer que la transmission est achevée : bit TRMT du registre TXSTA.

4.5 Registre IPR1

Ce registre contient les deux bits permettant de définir la priorité des interruptions liées à la transmission (TXIP) et à la réception (RCIP) d'une donnée.

Remarque : ces deux bits n'ont d'effet que si les 2 niveaux de priorité d'interruption sont validés (IPEN = 1).

4.6 Registres SPBRGH:SPBRG, bit BRGH (du registre TXSTA) et bit BRG16 (du registre BAUDCON)

Les registres SPBRGH et SPBRG permettent de configurer, avec le bit BRGH et le bit BRG16, la vitesse de transmission (en bit/s ou bauds[2]) de la liaison série. Les formules sont données dans TABLE 18-1. Dans ce tableau, *n* correspond à la valeur à mettre dans les registres SPBRGH:SPBRG.

TABLE 18-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-Bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-Bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-Bit/Asynchronous	
0	1	1	16-Bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-Bit/Synchronous	
1	1	x	16-Bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

Le bit SYNC, quand il est mis à 0, permet de définir une liaison série asynchrone.

Le bit BRG16, quand il est mis à 1, permet de travailler avec le registre SPBRG sur 16 bits (SPBRGH:SPBRG). Si BRG16 est mis à 0, seul le registre SPBRG sur 8 bits est utilisé.

Par exemple, pour une transmission à 9600 bauds, avec un microcontrôleur cadencé à F_{OSC} = 4 MHz, nous obtenons (SYNC doit forcément être égal à 0) :

- Avec les bits BRG16 = 0 et BRGH = 0, on trouve $n = F_{OSC}/(Baud\ Rate * 64) - 1$.

A.N. : $n = 4000000 / (9600 * 64) - 1 = 5.51$

n ne peut prendre que des valeurs entières. Nous prendrons donc pour *n* la valeur 6.

L'imprécision sur *n* nous amène une erreur sur la vitesse de transmission.

Baud Rate réel = $F_{OSC} / (64 (n + 1))$ soit 8928.59.

L'erreur est donc de -6.99 % (calcul : (8929 - 9600) / 9600)).

- Avec les bits BRG16 = 0 et BRGH = 1, on trouve $n = F_{OSC}/(Baud\ Rate * 16) - 1$.

A.N. : $n = 4000000 / (9600 * 16) - 1 = 25.041$

n ne peut prendre que des valeurs entières. Nous prendrons donc pour *n* la valeur 25.

L'imprécision sur *n* nous amène une erreur sur la vitesse de transmission.

Baud Rate réel = $F_{OSC} / (16 (n + 1))$ soit 9615.38.

L'erreur est donc de 0.16 %.

On retiendra la combinaison minimisant l'erreur commise sur la vitesse de transmission, c'est-à-dire ici la deuxième combinaison.

Ces calculs sont fastidieux. Heureusement, Microchip donne des tableaux (pages 207-208 de la doc. du microcontrôleur) permettant de trouver rapidement les valeurs à mettre dans SPBRGH:SPBRG, BRG16 et BRGH en fonction :

- de la vitesse de transmission (0.3 Kbauds, 1.2 Kbauds, …, 115.2 Kbauds) ;
- de la fréquence du quartz du PIC.

Remarque : la plupart du temps, il sera suffisant de travailler avec un registre SPBRG sur 8 bits (BRG16 = 0 et utilisation de SPBRG uniquement). En effet, la nécessité de travailler sur 16 bits s'impose uniquement pour obtenir de faibles vitesses de transmission avec un quartz de fréquence élevée. Le critère de choix 8 ou 16 bits reste cela dit l'erreur minimale sur la vitesse.

[2] Comme l'unité d'échange est ici le bit et que celui-ci est bivalent, les vitesses en bit/s ou en baud sont les mêmes.

BRG16 = 0 (SPBRG s'exprime sur 8 bits)

TABLE 18-3: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	—	—	—	—	—	—	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BRG16 = 1 (SPBRG s'exprime sur 2 registres 8 bits : SPBRGH-SPBRG)

TABLE 18-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	0.300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1.201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2.403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	0.300	0.04	832	0.300	-0.16	415	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 ~~or SYNC = 1, BRG16 = 1~~											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	0.300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1.200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2.400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9.615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19.230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57.142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117.647	-2.12	16

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 ~~or SYNC = 1, BRG16 = 1~~								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	0.300	0.01	3332	0.300	-0.04	1665	0.300	-0.04	832
1.2	1.200	0.04	832	1.201	-0.16	415	1.201	-0.16	207
2.4	2.404	0.16	415	2.403	-0.16	207	2.403	-0.16	103
9.6	9.615	0.16	103	9.615	-0.16	51	9.615	-0.16	25
19.2	19.231	0.16	51	19.230	-0.16	25	19.230	-0.16	12
57.6	58.824	2.12	16	55.555	3.55	8	—	—	—
115.2	111.111	-3.55	8	—	—	—	—	—	—

4.7 Registres de contrôle et d'état

4.7.1 Bit BRG16 du registre BAUDCON

2. Dans le registre BAUDCON, le seul bit à configurer est BRG16. Par quelle instruction le mettez-vous à 0 par exemple ?

4.7.2 Registres TXSTA et RCSTA

Remarque : Le nom de ces registres n'est pas forcément bien choisi. En effet, dans TXSTA on règle certes des aspects de la transmission mais aussi des aspects communs avec la réception : bits SYNC et BRGH. De même, dans RCSTA, on règle certes des aspects de la réception mais aussi un aspect commun avec la transmission : bit SPEN.

REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **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)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardv
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

Bit 3 pas utilisé : mettre à 0

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

Remarque : Le bit SENDB n'a pas été expliqué. Il faut le mettre à 0.

REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **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.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-Bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the
0 = Disables address detection, all bytes are received and ninth
Asynchronous mode 9-Bit (RX9 = 0):
Don't care.

Bit 3 à bit 1 pas utilisés : mettre à 0

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be cleared by reading RCREG register
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit, CREN)
0 = No overrun error

bit 0 **RX9D:** 9th Bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

5 Programmation en C de l'UART du PIC (sans interruption)

5.1 Initialisation de l'UART

Les caractéristiques (vitesse, etc.) de la liaison à réaliser étant connues, pour initialiser l'UART, il y a donc lieu de :

- initialiser le registre SPBRG (vous avez trouvé une solution page 8) OU SPBRGH et SPBRG (vous avez trouvé une solution page 9) ;
- initialiser le bit BRG16 du registre BAUDCON : à 0 (vous avez trouvé une solution page 8) OU à 1 (vous avez trouvé une solution page 9) ;
- initialiser le registre TXSTA ;
- initialiser le registre RCSTA.

Remarque : Lorsque des bits sont en lecture seule (TRMT, FERR, OERR, RX9D), peu importe la valeur avec laquelle on les initialise, cela n'a pas d'effet.

5.2 Transmission d'une donnée

Pour transmettre une donnée 8 bits une fois l'UART initialisé, il faut :

- déposer dans TXREG la donnée ;
- attendre tant que celle-ci n'a pas été complètement émise en consultant le bit TRMT.

5.3 Réception d'une donnée

Pour recevoir une donnée 8 bits une fois l'UART initialisé, il faut :

- attendre tant qu'il n'y a pas de donnée reçue en consultant le bit RCIF ;
- exploiter la donnée reçue : celle-ci est dans le registre RCREG.

6 Exercice

3. Initialiser la liaison série avec les caractéristiques suivantes : 115200, 8, N, 1, pas de contrôle de flux. L'oscillateur est à 40 MHz.
4. Ensuite :
    - envoyer le caractère 'A' ;
    - attendre la réception d'un caractère. Une fois arrivé, son code ASCII sera déposé sur les leds du port D.