

3IRC – Mise en œuvre d'un système à microprocesseur

Etude des périphériques dans un microcontrôleur
8051F020 – 3

UART -- Version 2022



Les Périphériques UART

Universal Asynchronous Receiver
Transmitter

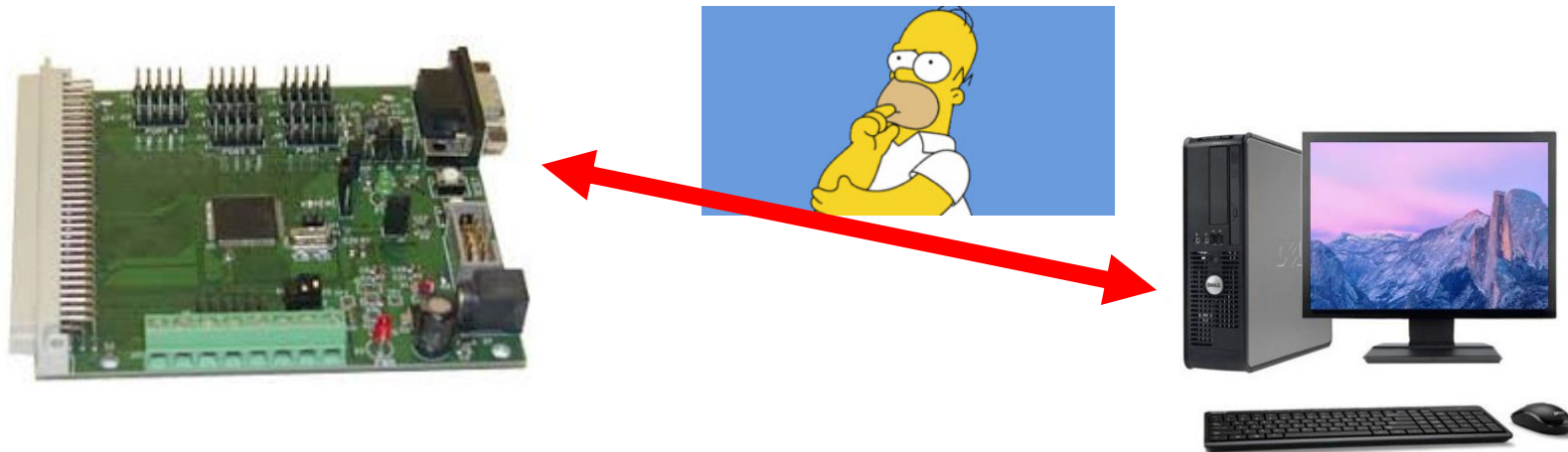
UART – USART Quel usage?

Problématique: Comment établir une communication **simple** entre 2 systèmes à processeurs?

Simple veut dire:

1. Un nombre de signaux à échanger le plus faible possible
2. Un protocole d'échange d'information réduit à sa plus simple expression
3. Utilisation d'interfaces matérielles déjà disponibles sur le micro-ordinateur
4. Coté micro ordinateur, possibilité d'utiliser des applications élémentaires de type terminal de commande pour communiquer

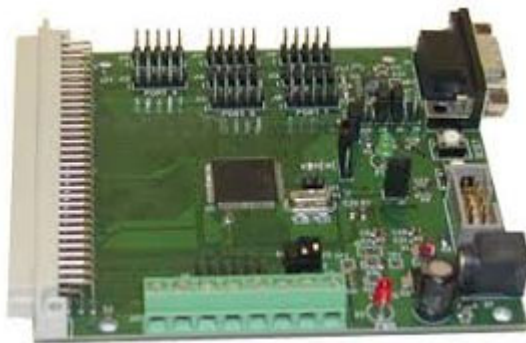
Pourquoi une liaison simple: pour quelle soit gérable pour un micro contrôleur de faible puissance. Ce type de liaison est beaucoup utilisée dans des phases de débogage et de configuration des systèmes.



UART – USART Communication UC <-> PC

Exemple de communication simple au niveau du PC: **la liaison série RS232 asynchrone**

- Condition 1: respectée – dans sa version simplifiée la communication s'établit sur 2 fils, un signal de transmission, et un signal de réception, (sans oublier la liaison équipotentielle 0V sur une « troisième » fil)
- Condition 2: respectée – aucun protocole, tout octet échangé est un octet de données
- Condition 3: respectée (avec une réserve). Les liaisons RS232 équipaient tous les PC, il y a encore quelques années. Aujourd'hui, des adaptateurs USB existent pour émuler côté PC cette liaison série
- Condition 4: respectée - Côté micro ordinateur, la communication peut se faire via un logiciel terminal de commande



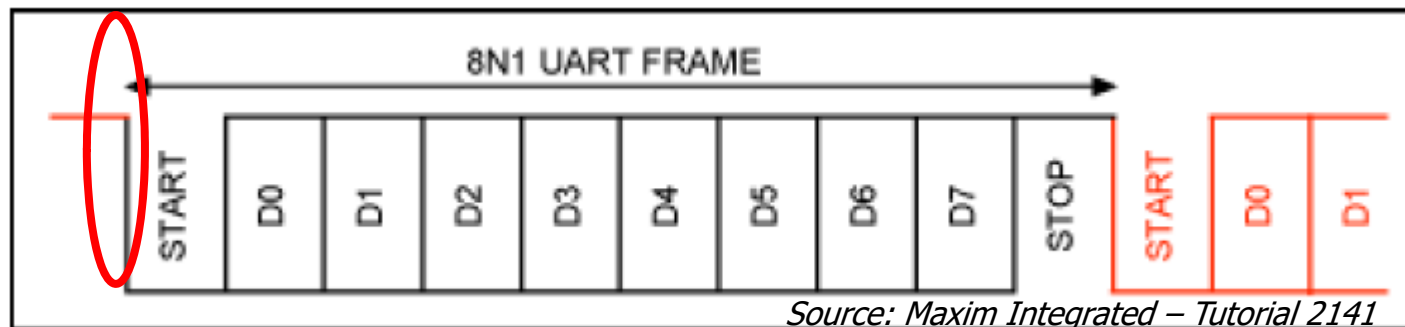
```
COM1 - PuTTY
//-----//
//----- MENU PRINCIPAL -----//
//-----//
*** Selection du test: ***
'P' -- Lancement test des ports
'C' -- Lancement test connectique sur la carte
'D' -- Lancement test des DAC
'A' -- Lancement test des ADC
'Q' -- Lancement test acquisition
//-----//
```



Transmission Asynchrone

Dans ce mode, l'horloge (signal utilisé pour la synchronisation des échanges) n'est pas transmise entre émetteur et récepteur, le récepteur doit donc connaître **précisément** la vitesse de transmission.

Transmission asynchrone: Premier Front descendant (bit start): le récepteur se « cale » dessus pour décoder -



Exemple de transmission typique d'un octet

Côté PC: le Terminal de commande – Exemple: Putty

Une application Terminal de commande sur PC, telle que **Putty** peut assurer la communication sur un port série RS232.

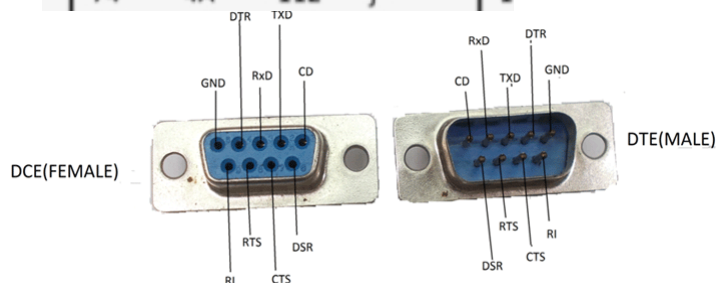
Lorsqu'elle est configurée en mode « Serial », alors l'application fonctionne de la manière suivante:

- A chaque appui sur une touche du clavier, le code ASCII (1 octet) de la touche appuyée est envoyé sur la liaison série. Ex: lors d'un appui sur la touche 'A' (majuscules activées), l'octet de valeur 0x41 est transmis sur la liaison série.
- Inversement, à chaque fois qu'un octet est reçu via la liaison série, on affiche sur l'écran le caractère équivalent au code ASCII. Ex: si on reçoit un octet de valeur 0x48, alors sur l'écran s'affiche le caractère 'H'

	Dec	Hex	Oct	Char	D
	64	40	100	@	9
	65	41	101	A	9
	66	42	102	B	9
	67	43	103	C	9
	68	44	104	D	9
	69	45	105	E	9
	70	46	106	F	9
	71	47	107	G	1
	72	48	110	H	1
	73	49	111	I	1
	74	4A	112	J	1

Extra la ta

Extrait de
la table ASCII



UART – USART - Définition

Côté micro contrôleur: c'est le périphérique UART/USART qui sera chargé de gérer ces communications en mode série.

UART = Universal Asynchronous Receiver-Transmitter

USART = Universal Synchronous Asynchronous Receiver-Transmitter

Rôle de ce périphérique: Recevoir et transmettre des informations sous forme série.

Intérêt: minimiser le nombre de fils (1 fil au lieu de 8)

Inconvénient: débit limité (par rapport à une transmission en parallèle)

Usage: échange point à point entre 2 « systèmes » de signaux de commande et/ou d'informations

Universal: divers modes possibles – nombre de bits, parité, vitesse....

Synchronous: l'horloge est transmise, soit au travers d'un signal supplémentaire, soit elle est associée à l'information transmise.

Asynchronous: transmission asynchrone, il n'y a pas de transmission de l'horloge aussi bien à l'aide d'un signal spécifique, ou associée à l'information transmise

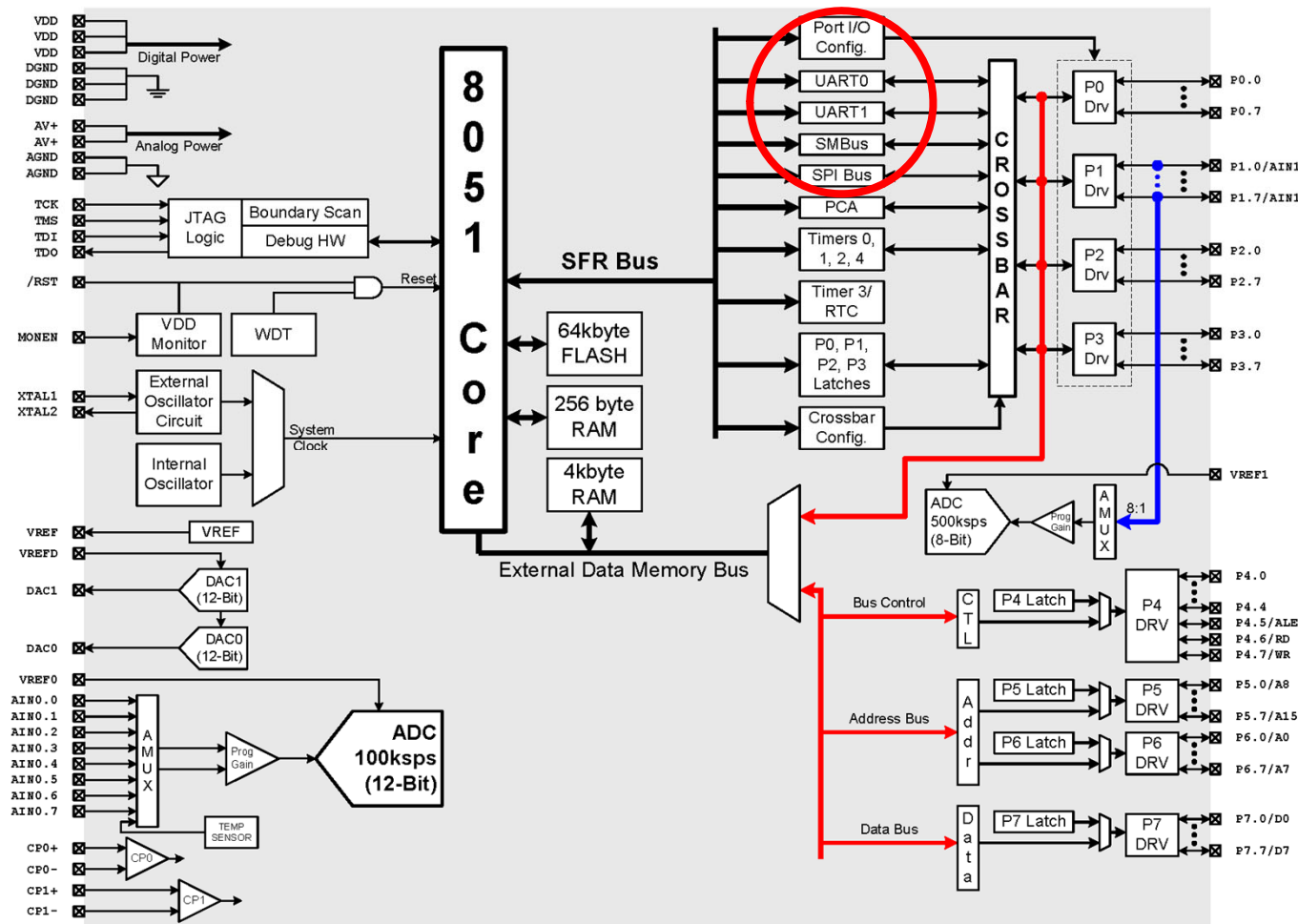
Receiver: peut assurer la réception d'informations

Transmitter: peut assurer la transmission d'informations



Les UART dans le 8051F020

Universal Asynchronous Receiver Transmitter dans le 8051F020



**2 UARTS
Disponibles
UART0 et UART1**
Fonctionnement en
Synchrone
unidirectionnel
ou Asynchrone bi-
directionnel



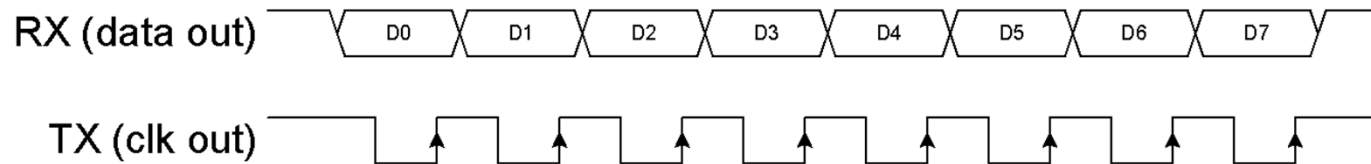
LIVE AND
DISCOVER

Fonctionnement en synchrone

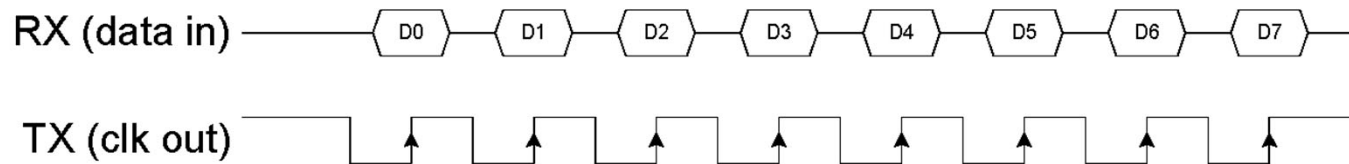
Mode Synchrone **Unidirectionnel**

1 signal d'horloge + 1 signal de donnée TX ou RX

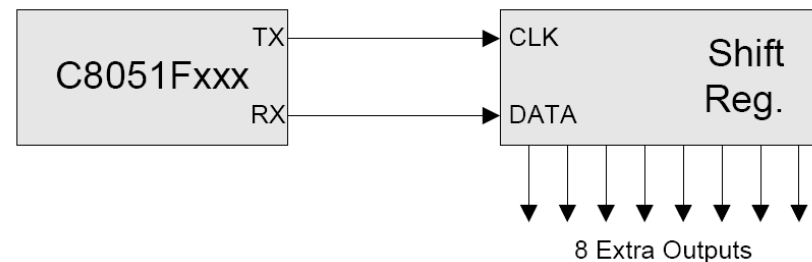
MODE 0 TRANSMIT



MODE 0 RECEIVE



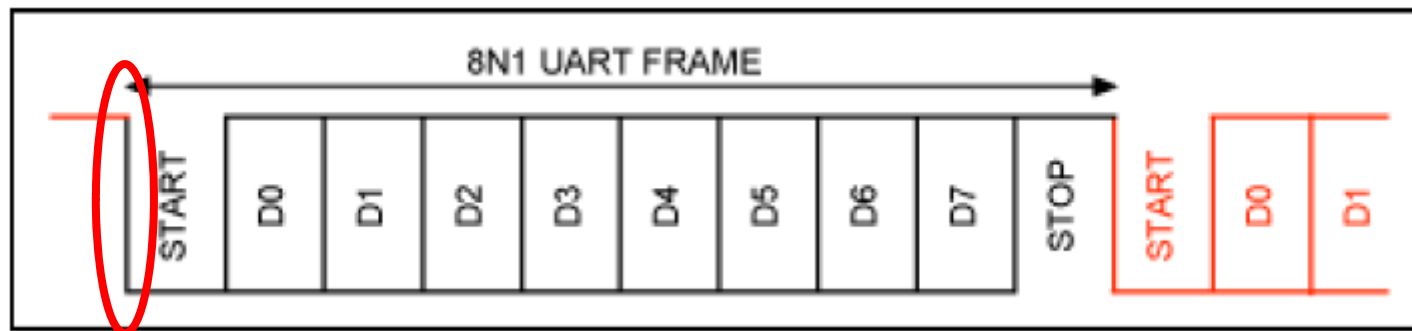
constituer une alternative intéressante pour piloter des entrées sorties additionnelles (Registre à décalage)



Fonctionnement en Asynchrone

Dans ce mode, l'horloge n'est pas transmise entre émetteur et récepteur, le récepteur doit donc connaître **précisément** la vitesse de transmission.

Transmission asynchrone: Premier Front descendant (bit start): le récepteur se « cale » dessus pour décoder -



Source: Maxim Integrated – Tutorial 2141

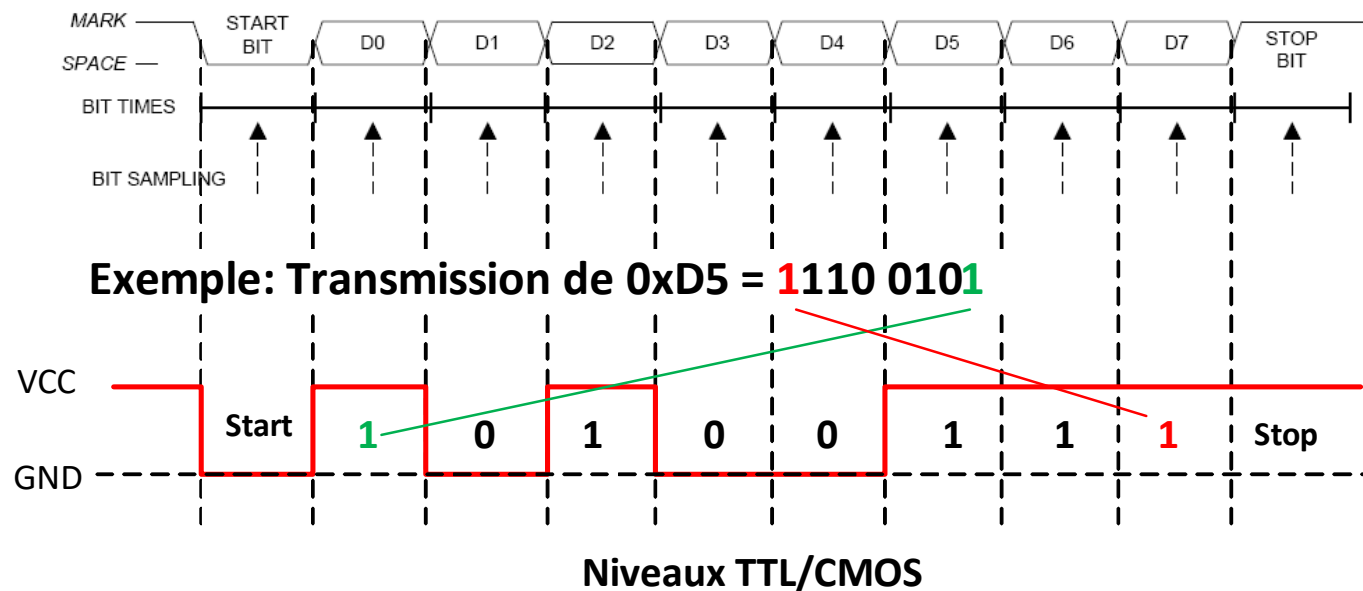
Exemple de transmission typique d'un octet

! IMPORTANT

Fonctionnement des UART en asynchrone

- Mode Asynchrone 8 ou 9 bits
- 1 signal TX (transmission) et un signal RX (réception)
 - **8 bits: communication type RS232**
 - 9 bits : communication multi-processeurs (obsolète)

Chronogramme Transmission de données par UART



Niveaux de tension RS232 - Normalisation

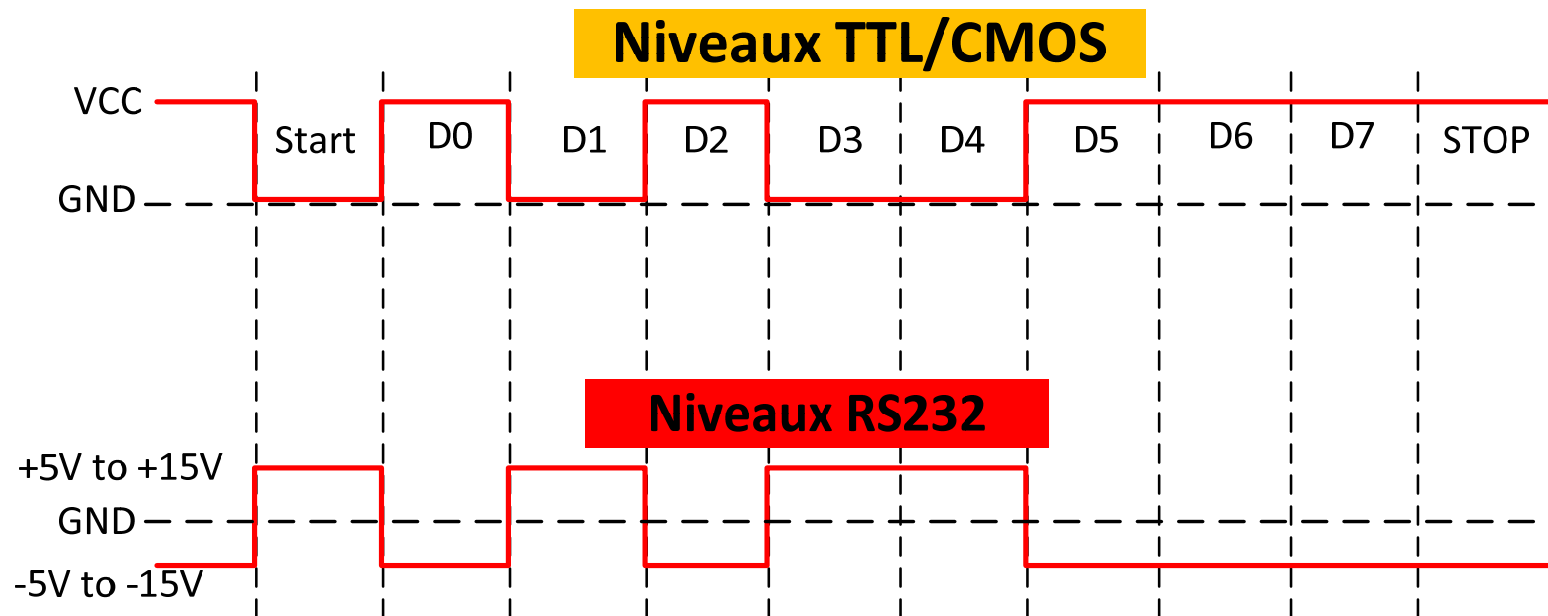
RS-232 Specifications

- **TRANSMITTED SIGNAL VOLTAGE LEVELS:**
 - **Binary 0:** +5 to +15 Vdc (called a “space” or “on”)
 - **Binary 1:** -5 to -15 Vdc (called a “mark” or “off”)
- **RECEIVED SIGNAL VOLTAGE LEVELS:**
 - Binary 0: +3 to +13 Vdc
 - Binary 1: -3 to -13 Vdc
- **DATA FORMAT:**
 - Start bit: Binary 0
 - Data: 5, 6, 7 or 8 bits
 - Parity: Odd, even, mark or space (not used with 8-bit data)
 - Stop bit: Binary 1, one or two bits

Chronogrammes RS232

Niveaux **CMOS/TTL** sur les broches du 8051F020
ou **RS232** sur le connecteur DB9

Exemple: Transmission de 0xE5 = 1110 0101

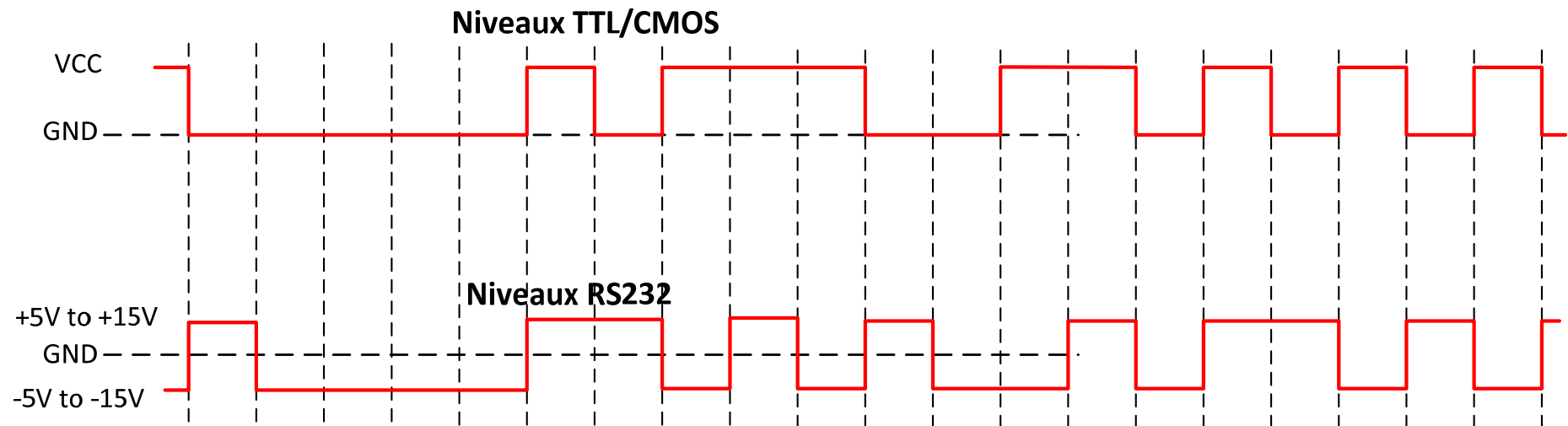


! IMPORTANT

Pause Chronogramme RS232

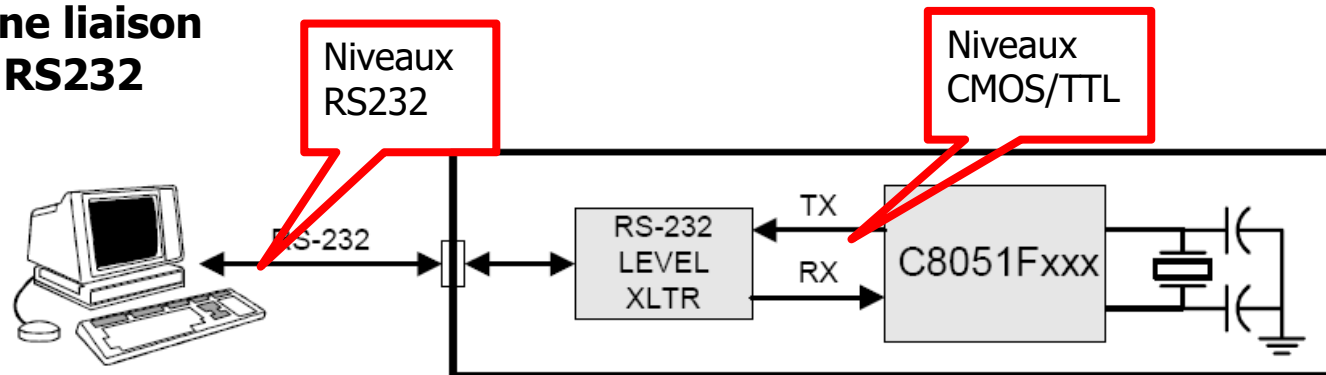
Niveaux CMOS/TTL sur les broches du 8051F020
ou RS232 sur le connecteur DB9

Décoder les signaux Série TTL/CMOS et série RS232

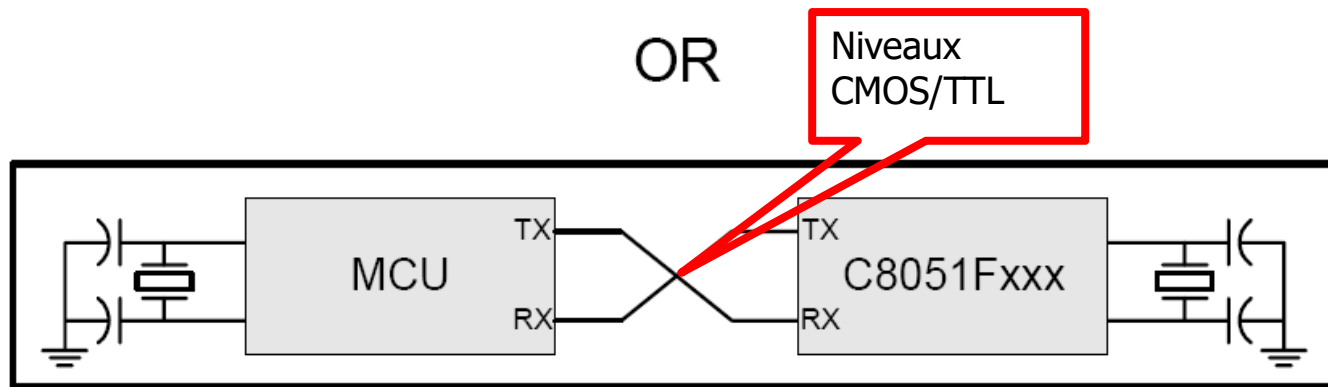


Applications UARTs en asynchrone

Communications Externes au travers d'une liaison normalisée RS232



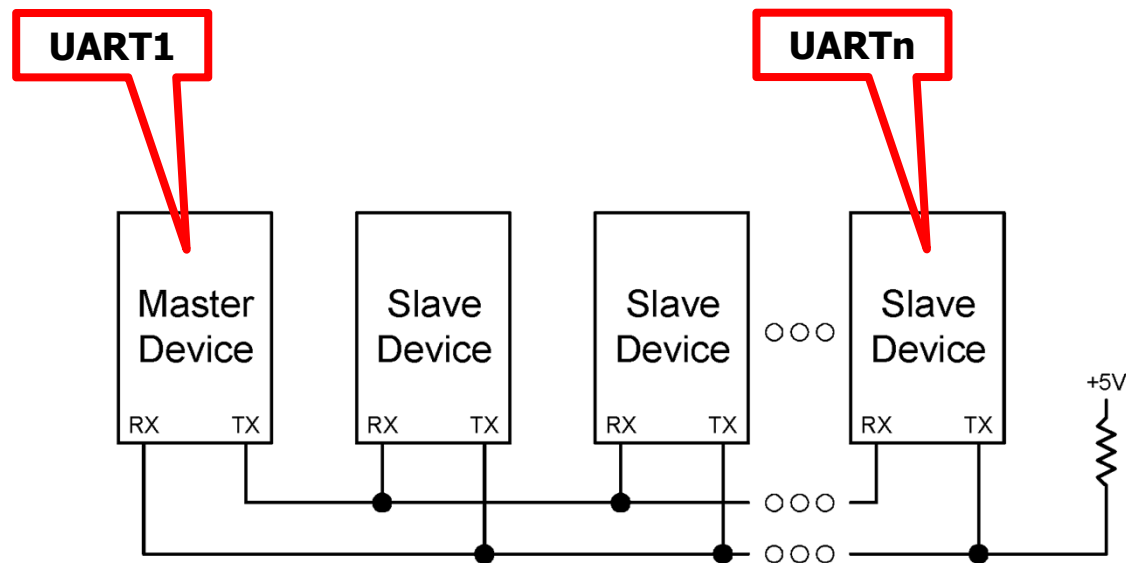
OR



Communications Internes (sur une même carte électronique) entre 2 processeurs au travers d'une liaison non normalisée RS232

Pour info: UART en mode Multiprocesseur

- Communication Multiprocesseurs Maître/Esclaves entre plusieurs UARTs
- Utilisation d'un 9^{ième} bit de donnée pour spécifier l'envoi d'une adresse



Modes de Fonctionnement des UART (UART0 et UART1)

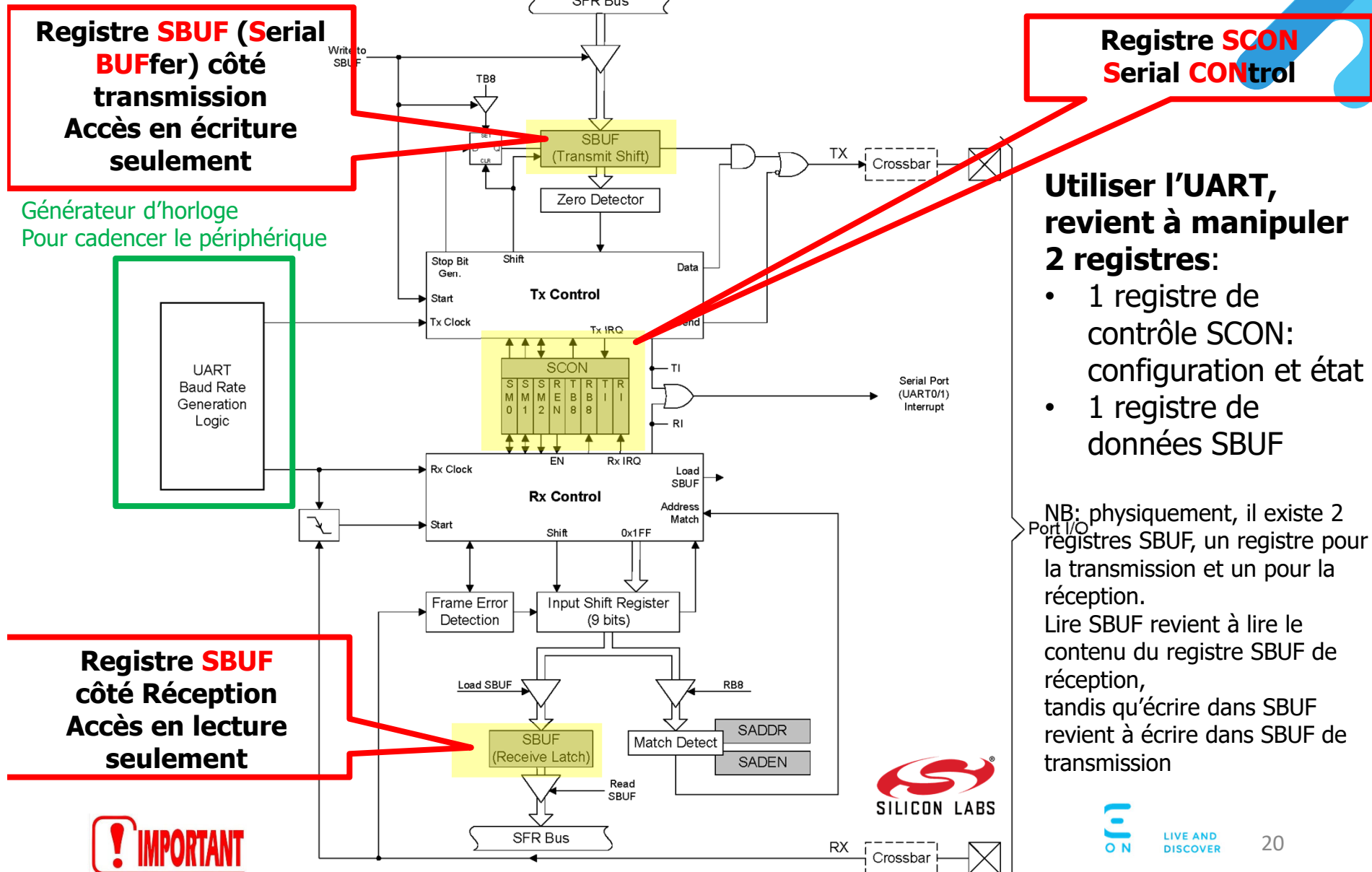
Table 20.1. UART0 Modes

Mode	Synchronization	Baud Clock	Data Bits	Start/Stop Bits
0	Synchronous	SYSCLK / 12	8	None
1	Asynchronous	Timer 1 or 2 Overflow	8	1 Start, 1 Stop
2	Asynchronous	SYSCLK / 32 or SYSCLK / 64	9	1 Start, 1 Stop
3	Asynchronous	Timer 1 or 2 Overflow	9	1 Start, 1 Stop

Mode Multi-processeurs

**LE mode le plus utilisé
Notamment en
« Debug »**

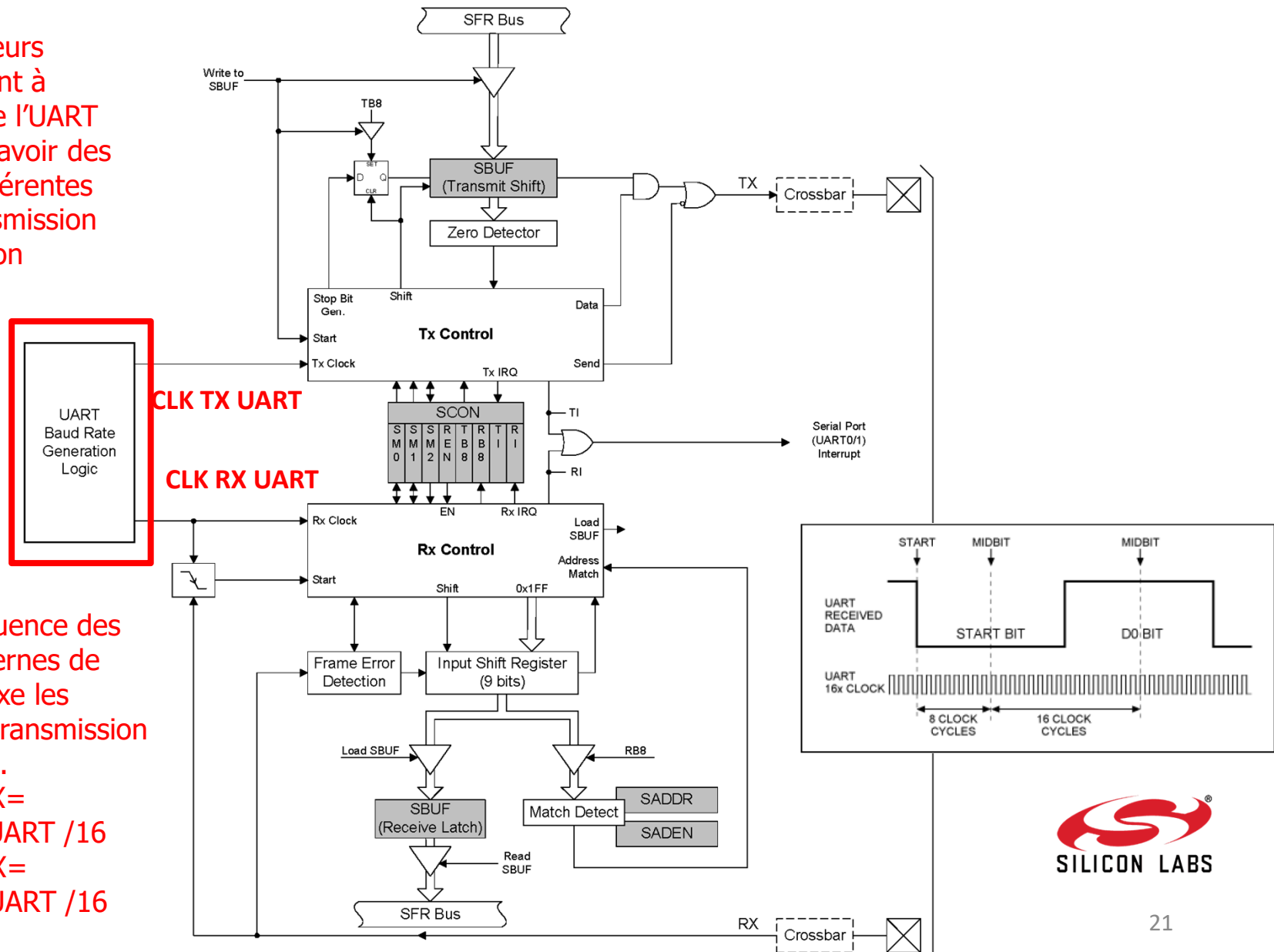
Block Diagram UART0



L'UART0 et ses horloges

Les générateurs d'horloge sont à l'extérieur de l'UART
Possibilité d'avoir des horloges différentes pour la transmission et la réception

C'est la fréquence des horloges internes de l'UART qui fixe les vitesses de transmission sur la liaison.
BaudRate TX=
 $F_CLK_TX_UART / 16$
BaudRate RX=
 $F_CLK_RX_UART / 16$



Génération d'horloge pour les UART – Exemple UART0

Figure 22.13. T2 Mode 2 Block Diagram

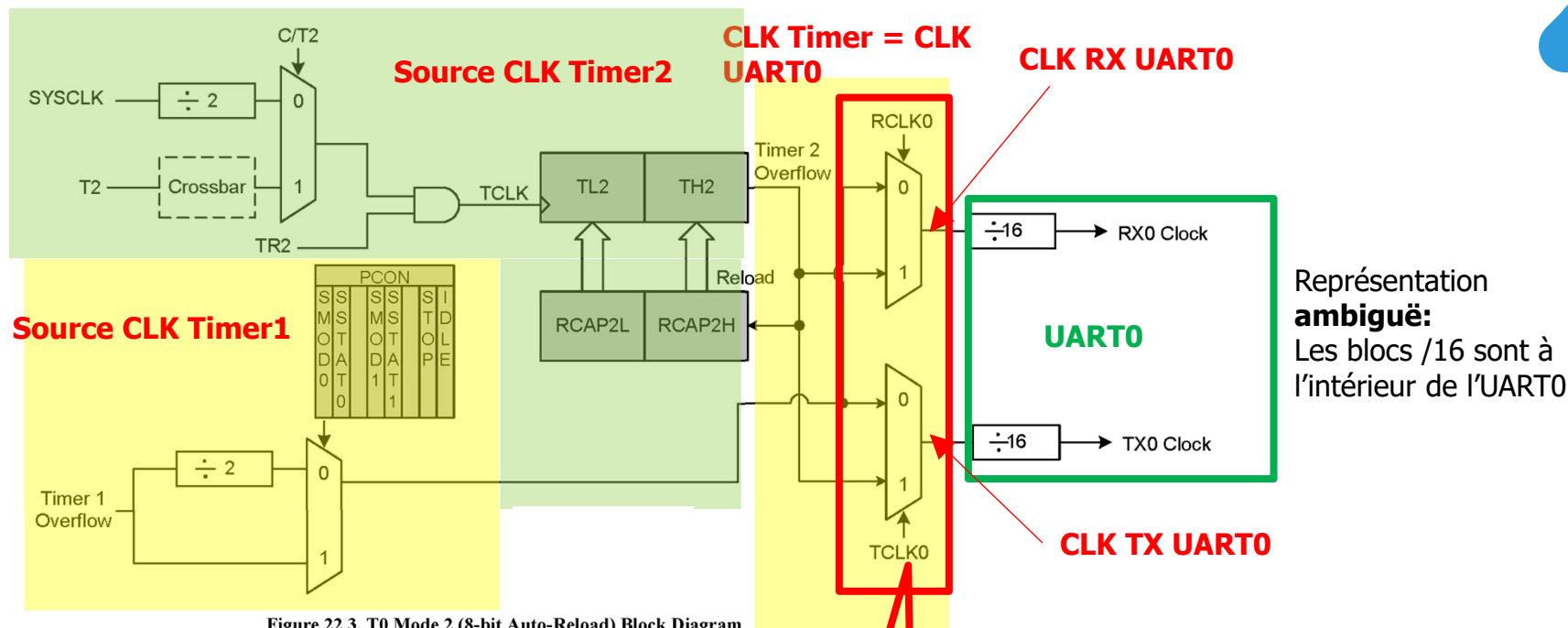
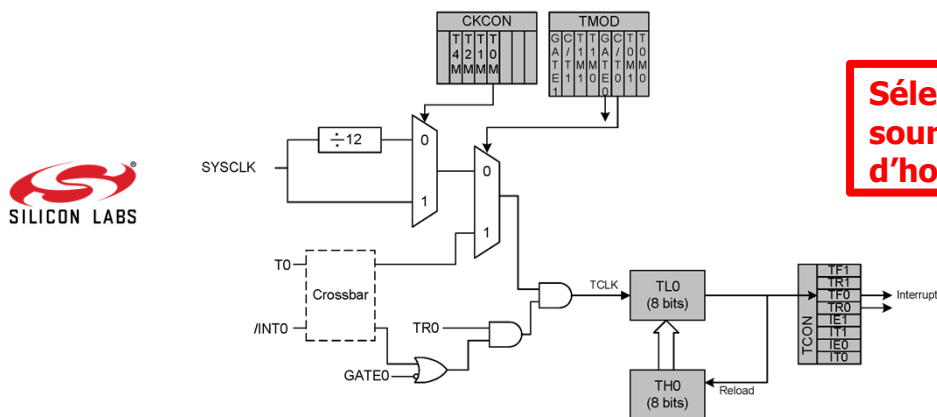


Figure 22.3. T0 Mode 2 (8-bit Auto-Reload) Block Diagram



CLK TX UART0 = 16 X TX BaudRate
CLK RX UART0 = 16 X RX BaudRate

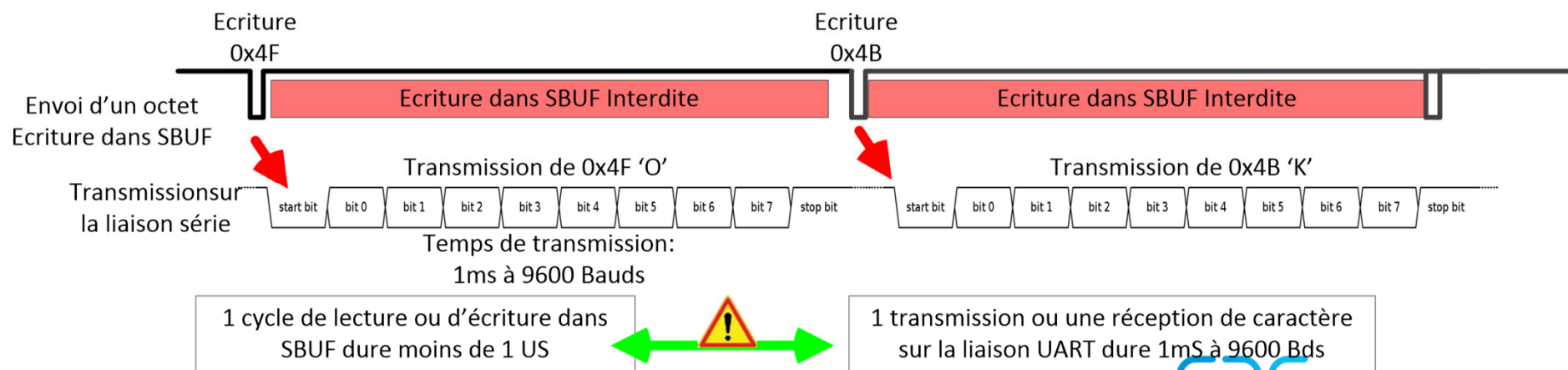
- Pour l'UART0, 2 sources possibles d'horloge: Timer2 ou Timer 1
- Pour l'UART1, 2 sources possibles d'horloge: Timer4 ou Timer 1

Problématique: Transmission de caractères sur la liaison série

Pour envoyer un caractère sur la liaison série, il « suffit » d'écrire dans SBUF.

Exemple: on souhaite produire l'affichage du message « OK » sur un terminal de commande tel que Putty.

- Cela revient à envoyer consécutivement sur la liaison série les octets de valeur 0x4F ('O') et 0x4B ('K')
- Dans un premier temps on écrit la première valeur à transmettre (0x4F) dans SBUF
- Immédiatement la transmission débute, et le contenu de SBUF subit des décalages bit à bit pour transférer les informations sur la broche TX.
- Cette opération de décalages dure $(1 + 8 + 1) / \text{BaudRate}$ soit environ 1ms pour un BaudRate de 9600
- Durant toute cette durée, **SBUF n'est pas disponible pour recevoir un autre octet.**
- L'écriture dans second octet (0x4B) doit donc être différée d'une durée égale ou supérieure à la durée de transmission d'un octet.
- Si cette condition n'est pas respectée, le résultat est imprévisible



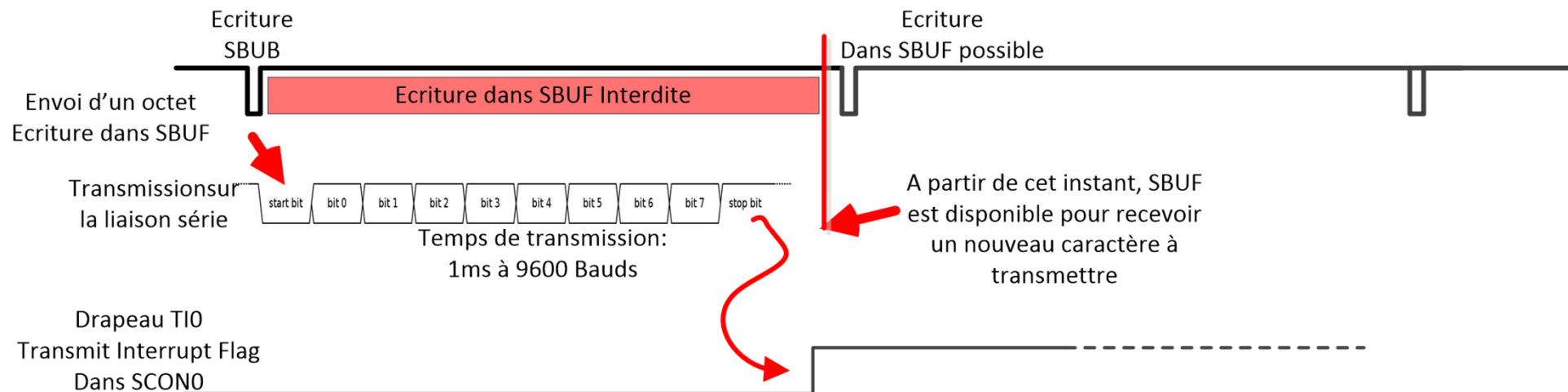
Solution: Transmission de caractères sur la liaison série



Attention avant d'écrire dans SBUF il faut s'assurer que SBUF (côté transmission) soit « libre », c'est-à-dire s'assurer que le dernier octet écrit a été complètement transmis sur la liaison série.

Les méthodes:

- La « rustique » : attente par temporisation (si elle est recevable pour tester une liaison série, elle n'est pas admissible dans une application « professionnelle »).
- La scrutation : on va surveiller un drapeau qui est chargé de signaler que la transmission du caractère en cours est terminée et que SBUF est donc disponible. Ce drapeau (TIO) se trouve dans le registre de contrôle de l'UART. Ne pas oublier de le remettre à zéro....
- L'interruption : il est possible de provoquer le déclenchement d'une interruption pour signaler que la transmission en cours est terminée et que SBUF est donc disponible.

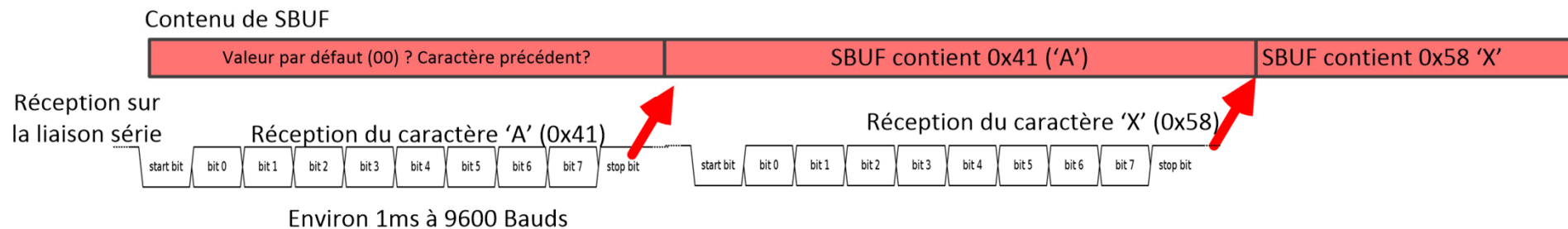


Problématique: Réception de caractères sur la liaison série

Pour récupérer un hypothétique caractère reçu, il « suffit » de lire SBUF

Fonctionnement de la réception dans l'UART:

Lorsque qu'un caractère arrive dans l'UART (une suite de 10 bits), il est reconstitué en un octet dans un registre à décalage, puis lorsque l'octet est entièrement reconstitué, il est transféré dans le registre SBUF de réception. Cette valeur se maintiendra jusqu'à la réception d'un prochain caractère.

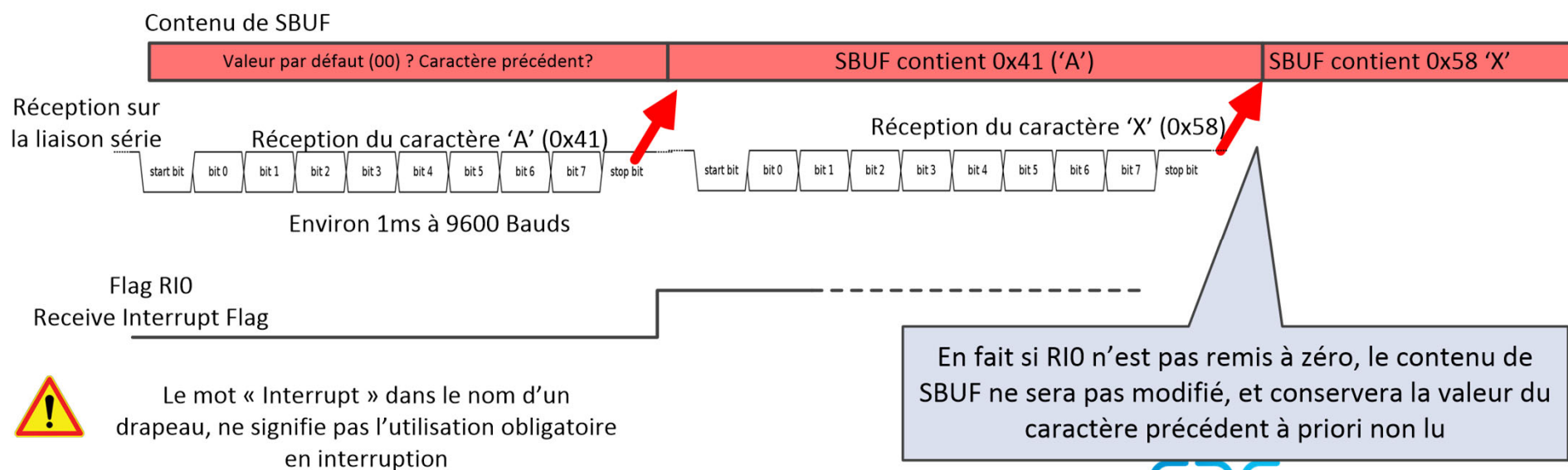


Solution: Réception de caractères sur la liaison série

Attention avant de lire SBUF il faut s'assurer que SBUF contienne bien un caractère nouvellement reçu, si SBUF est « vide » (pas de caractère reçu) ne pas lire et pareillement ne pas lire plusieurs fois le même caractère reçu

Les méthodes:

- La scrutation : on va surveiller un drapeau qui est chargé de signaler qu'un caractère a été reçu et qu'il est donc disponible dans SBUF. Ce drapeau (RIO) se trouve dans le registre de contrôle de l'UART.
- L'interruption : il est possible de provoquer le déclenchement d'une interruption pour signaler qu'un nouveau caractère a été reçu et qu'il est disponible dans SBUF.



Registre SCON0 Bit 7 6 5

Figure 20.8. SCON0: UART0 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SM00/FE0	SM10/RXOV0	SM20/TXCOL0	REN0	TB80	RB80	TI0	RI0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x98

Bits7-6: The function of these bits is determined by the SSTAT0 bit in register PCON. If SSTAT0 is logic 1, these bits are UART0 status indicators as described in [Section 20.3](#). If SSTAT0 is logic 0, these bits select the Serial Port Operation Mode as shown below.

SM00-SM10: Serial Port Operation Mode:

SM00	SM10	Mode
0	0	Mode 0: Synchronous Mode
0	1	Mode 1: 8-Bit UART, Variable Baud Rate
1	0	Mode 2: 9-Bit UART, Fixed Baud Rate
1	1	Mode 3: 9-Bit UART, Variable Baud Rate

Bit5: SM20: Multiprocessor Communication Enable.

If SSTAT0 is logic 1, this bit is a UART0 status indicator as described in [Section 20.3](#).

If SSTAT0 is logic 0, the function of this bit is dependent on the Serial Port Operation Mode.

Mode 0: No effect.

Mode 1: Checks for valid stop bit.

- 0: Logic level of stop bit is ignored.
- 1: RI0 will only be activated if stop bit is logic level 1.

Modes 2 and 3: Multiprocessor Communications Enable.

- 0: Logic level of ninth bit is ignored.
- 1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1 and the received address matches the UART0 address or the broadcast address.

**Mode
Configuration
SSTAT0 = 0
dans PCON**

Configuration

Registre SCON0 Bit 4 3 2 1 0



Configuration

Bit4: REN0: Receive Enable.
This bit enables/disables the UART0 receiver.
0: UART0 reception disabled.
1: UART0 reception enabled.

Bit3: TB80: Ninth Transmission Bit.
The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.

Bit2: RB80: Ninth Receive Bit.
The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM20 is logic 0, RB80 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.

Bit1: TI0: Transmit Interrupt Flag.
Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.

Etat

Bit0: RI0: Receive Interrupt Flag.
Set by hardware when a byte of data has been received by UART0 (as selected by the SM20 bit). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.



Exemple de mise en œuvre de l'UART0 dans le 8051F020

Mise en œuvre de l'UART0

Configuration basique par scrutation



Objectif: Initialiser l'UART0 pour permettre des réceptions et des transmissions asynchrones à 9600 bauds (bits/seconde) avec 8 bits de données, sans parité.

- **Fonctions de configuration (appelées une seule fois)**

- `void CFG_clock_UART(void) ;`

- `Void CFG_uart0_model(void) ;`

- **Fonctions d'utilisation**

- `char getkey(void) ;` réception de caractère par l'UART

- `void putchar(char c) ;` émission de caractère par l'UART

Fichiers Configurables: Gestion des Entrées/Sorties



Basic I/O (Doc Keil)

The following files contain the source code for the low-level stream I/O routines. When you use µVision IDE, you can simply add the modified versions to the project.

C Source File Description



PUTCHAR.C (function `putchar`) Used by all stream routines that output characters. You may adapt this routine to your individual hardware (for example, LCD or LED displays). The default version outputs characters via the serial interface. An **XON/XOFF** protocol is used for flow control. Line feed characters (`'\n'`) are converted into carriage return/line feed sequences (`'\r\n'`).

GETKEY.C (function `_getkey`) Used by all stream routines that input characters. You may adapt this routine to your individual hardware (for example, matrix keyboards). The default version reads a character via the serial interface. No data conversions are performed..

Dans la librairie stdio, `putchar` et `_getkey` sont les fonctions de base pour toutes les fonctions

De gestion d'entrées-sorties telles que `printf` et `scanf`

Par défaut, `putchar` et `_getkey` utilisent l'UART0. `_getkey` est bloquant

Dépendances avec les configurations globales

Configurations globales du microcontrôleur:

- ✗ Les sources de « Reset »
- ✗ Le chien de garde
- ✓ L' (ou les) horloge(s) - on prendra **impérativement** SYSCLK = 22,1184 MHz
- ✗ Les mémoires
- ✗ La gestion de la puissance
- ✓ L'affectation et la configuration des broches d'entrées-sorties (GPIOs).

C'est le quartz externe à 22,1184 MHz, qui peut garantir une précision de l'horloge meilleure que 2% (0,5% environ)

Liaison avec d'autres périphériques

- Timer 1
- Timer 2

Table 14.1. Internal Oscillator Electrical Characteristics

VDD = 2.7V to 3.6V; T_a = -40°C to +85°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Internal Oscillator Frequency	OSCICN.[1:0] = 00	1.5	2	2.4	MHz
	OSCICN.[1:0] = 01	3.1	4	4.8	
	OSCICN.[1:0] = 10	6.2	8	9.6	
	OSCICN.[1:0] = 11	12.3	16	19.2	
Internal Oscillator Current Consumption (from VDD)	OSCICN.2 = 1		200		µA

Oscillateur interne: précision +/- 20% à pleine échelle en température

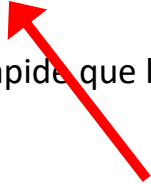
Configurer l'UART0

1. Générer une horloge de pilotage de l'UART0.

- Choix possible Timer 1 ou Timer 2 -> on choisit le Timer1
- Fréquence à générer: $9600 \times 16 = 153600\text{Hz}$

Car l'horloge qui pilote le fonctionnement interne de l'UART doit être 16 fois plus rapide que le débit sur la liaison série.

2. Configurer l'UART proprement dite

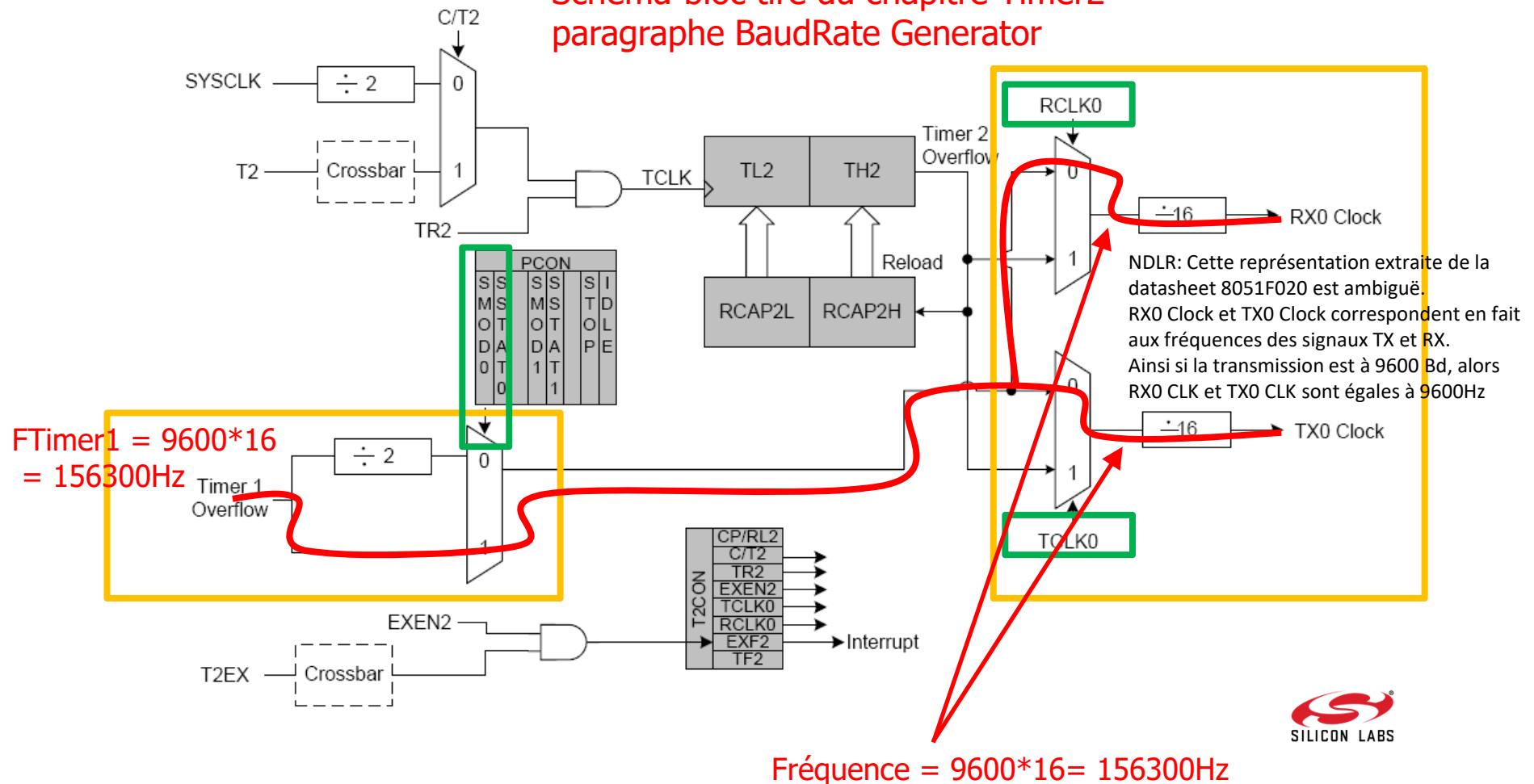


Pourquoi ce choix?
Si un Timer qui fonctionne en auto-rechargement sur 8 bits peut suffire, autant l'utiliser et laisser les Timers 16 bits à d'autres tâches...

Config Horloge UART0

Comme source d'horloge, l'UART0 peut fonctionner avec le Timer 2 ou le Timer 1, le choix a été fait de fonctionner avec le Timer 1

Schéma-bloc tiré du chapitre Timer2 –
paragraphe BaudRate Generator



Config Timer 1 pour UART0

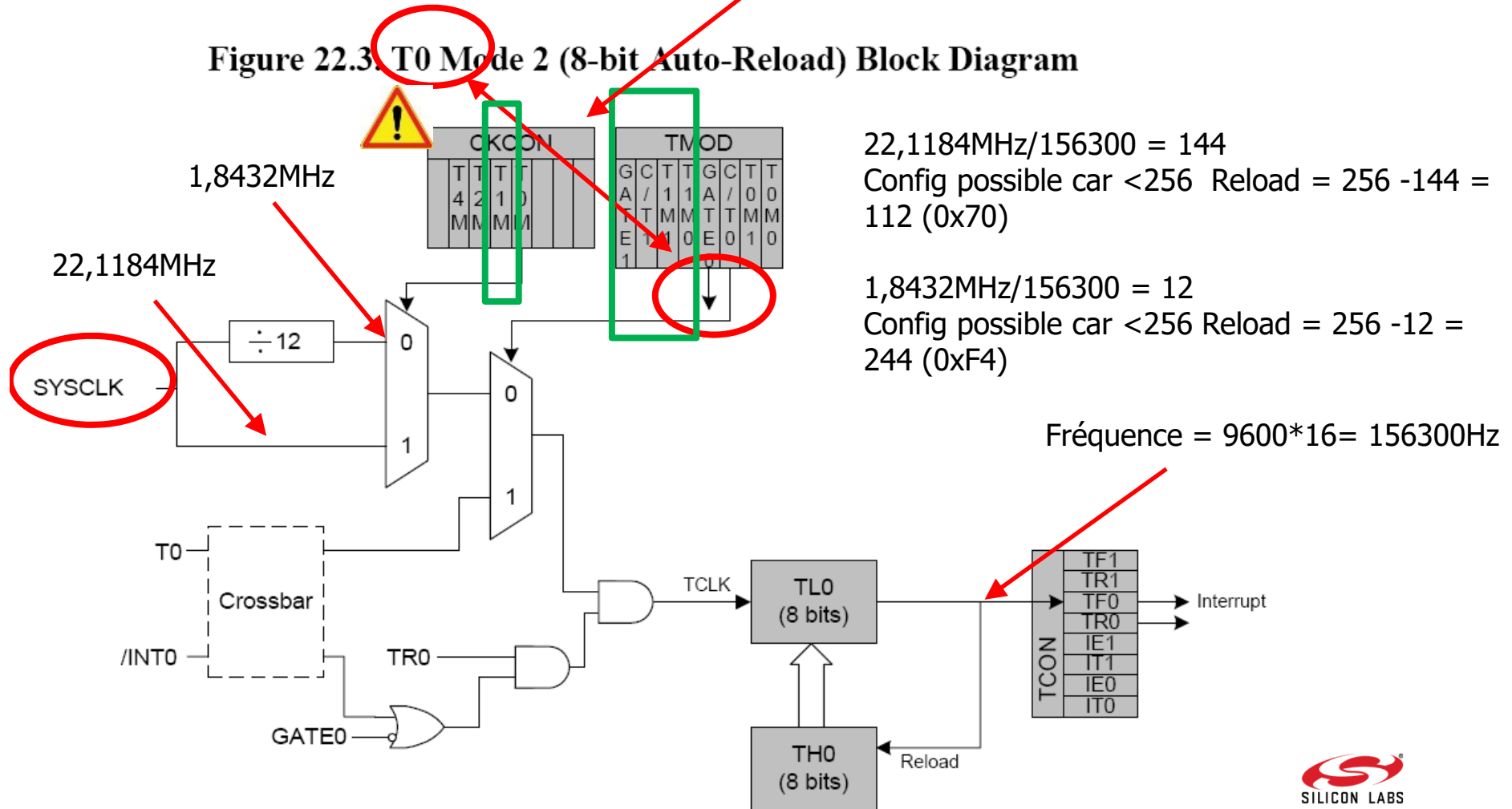
Attention, la figure représente les configurations pour le Timer0.

Or on travaille sur Timer1

Dans CKCON configurer T1M (au lieu de T0M)

Dans TMOD configurer Gate1 et C/T1 au lieu de Gate0 et C/T0

Figure 22.3. T0 Mode 2 (8-bit Auto-Reload) Block Diagram



Registre CKCON

Figure 22.1. CKCON: Clock Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	T4M	T2M	T1M	T0M	Reserved	Reserved	Reserved	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8E

Choix Timer1 CLK = SYSCLOCK

Registre TMOD

Figure 22.6. TMOD: Timer Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x89

Bit7: GATE1: Timer 1 Gate Control.
0 0: Timer 1 enabled when TR1 = 1 irrespective of /INT1 logic level.
 1: Timer 1 enabled only when TR1 = 1 AND /INT1 = logic 1.

Bit6: C/T1: Counter/Timer 1 Select.
0 0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.4).
 1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).

Bits5-4: T1M1-T1M0: Timer 1 Mode Select.
 These bits select the Timer 1 operation mode.

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

Pas accessible bit à bit!!

On doit modifier 4 bits du registre relatifs au Timer1, on ne connaît pas par ailleurs l'état des 4 autres bits du registre dédiés au timer0, donc: **Masquage obligatoire**

Registre TCON

Figure 22.5. TCON: Timer Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0x88

Bit7: TF1: Timer 1 Overflow Flag.
Set by hardware when Timer 1 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine.
0 0: No Timer 1 overflow detected.
1: Timer 1 has overflowed.

Bit6: TR1: Timer 1 Run Control.
1 0: Timer 1 disabled.
1: Timer 1 enabled.

Bit5: TF0: Timer 0 Overflow Flag.
X Set by hardware when Timer 0 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine.
0: No Timer 0 overflow detected.

Code Config CLK UART0

Fonction CFG_clock_UART - Utilisation du Timer 1

Timer1 Stoppé (précaution) - Action sur TR1 de TCON

Flag Timer1 effacé - - Action sur TF1 de TCON

Config CKCON - | T1M: Timer1 ClockSelect - CLK Timer = Sysclk

Config TMOD - Timer1 configuré en timer 8 bit avec auto-reload

Programmation du registre de rechargement TH1

Initialisation du registre Timer TL1 - Nécessaire?

Dévalidation de l'interruption Timer1 - Action sur registre IE

Timer1 démarré - - Action sur TR1 de TCON

Pause... Config Clock UART

- Remplacer le Timer 1 par le Timer 2 comme source d'horloge UART }

Configurer l'UART0



Générer une horloge de pilotage de l'UART0.

- Choix possible Timer 1
- Fréquence à générer: $9600 \times 16 = 153600\text{Hz}$

2. Configurer l'UART – Agir sur SCON0

Config SCON0 Bits 7....5

Figure 20.8. SCON0: UART0 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SM00/FE0	SM10/RXOV0	SM20/TXCOL0	REN0	TB80	RB80	TI0	RI0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x98

Bits7-6: The function of these bits is determined by the SSTAT0 bit in register PCON. If SSTAT0 is logic 1, these bits are UART0 status indicators as described in **Section 20.3**. If SSTAT0 is logic 0, these bits select the Serial Port Operation Mode as shown below.

SM00-SM10: Serial Port Operation Mode:

SM00	SM10	Mode
0	0	Mode 0: Synchronous Mode
0	1	Mode 1: 8-Bit UART, Variable Baud Rate
1	0	Mode 2: 9-Bit UART, Fixed Baud Rate
1	1	Mode 3: 9-Bit UART, Variable Baud Rate

Bit5: SM20: Multiprocessor Communication Enable. If SSTAT0 is logic 1, this bit is a UART0 status indicator as described in **Section 20.3**. If SSTAT0 is logic 0, the function of this bit is dependent on the Serial Port Operation Mode.

Mode 0: No effect.

Mode 1: Checks for valid stop bit.

0: Logic level of stop bit is ignored.

1: RI0 will only be activated if stop bit is logic level 1.

Modes 2 and 3: Multiprocessor Communications Enable.

0: Logic level of ninth bit is ignored.

1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1 and the received address matches the UART0 address or the broadcast address.

**Mode
Configuration
SSTAT0 = 0
dans PCON**

Config SCON0 Bits 4....0

Bit4:	REN0: Receive Enable. This bit enables/disables the UART0 receiver. 1 0: UART0 reception disabled. 1: UART0 reception enabled.
Bit3:	TB80: Ninth Transmission Bit. Non utilisés The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.
Bit2:	RB80: Ninth Receive Bit. Non utilisés The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM20 is logic 0, RB80 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.
Bit1:	TI0: Transmit Interrupt Flag. 0 Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.
Bit0:	RI0: Receive Interrupt Flag. 0 Set by hardware when a byte of data has been received by UART0 (as selected by the SM20 bit). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.

Configuration Registre T2CON_ Bits 7-6-5

On configure ici la source d'horloge pour l'UART: Timer2 ou Timer 1

Figure 22.14. T2CON: Timer 2 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF2	EXF2	RCLK0	TCLK0	EXEN2	TR2	C/T2	CP/RL2	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
						(bit addressable)		0xC8
Bit7:	TF2: Timer 2 Overflow Flag. Set by hardware when Timer 2 overflows. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. TF2 will not be set when RCLK0 and/or TCLK0 are logic 1.							
X								
Bit6:	EXF2: Timer 2 External Flag. Set by hardware when either a capture or reload is caused by a high-to-low transition on the T2EX input pin and EXEN2 is logic 1. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 Interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.							
X								
Bit5:	RCLK0: Receive Clock Flag for UART0. Selects which timer is used for the UART0 receive clock in modes 1 or 3.							
0								
	0: Timer 1 overflows used for receive clock. 1: Timer 2 overflows used for receive clock.							
Bit4:	TCLK0: Transmit Clock Flag for UART0. Selects which timer is used for the UART0 transmit clock in modes 1 or 3.							
0								
	0: Timer 1 overflows used for transmit clock. 1: Timer 2 overflows used for transmit clock.							

Registre PCON

Figure 12.15. PCON: Power Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SMOD0	SSTAT0	Reserved	SMOD1	SSTAT1	Reserved	STOP	IDLE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x87

Bit7: SMOD0: UART0 Baud Rate Doubler Enable.

1

This bit enables/disables the divide-by-two function of the UART0 baud rate logic for configurations described in the UART0 section.

0: UART0 baud rate divide-by-two enabled.

1: UART0 baud rate divide-by-two disabled.

Bit6: SSTAT0: UART0 Enhanced Status Mode Select.

0

This bit controls the access mode of the SM20-SM00 bits in register SCON0.

0: Reads/writes of SM20-SM00 access the SM20-SM00 UART0 mode setting.

1: Reads/writes of SM20-SM00 access the Framing Error (FE0), RX Overrun (RXOV0), and TX Collision (TXCOL0) status bits.

Bit5: Reserved. Read is undefined. Must write 0.

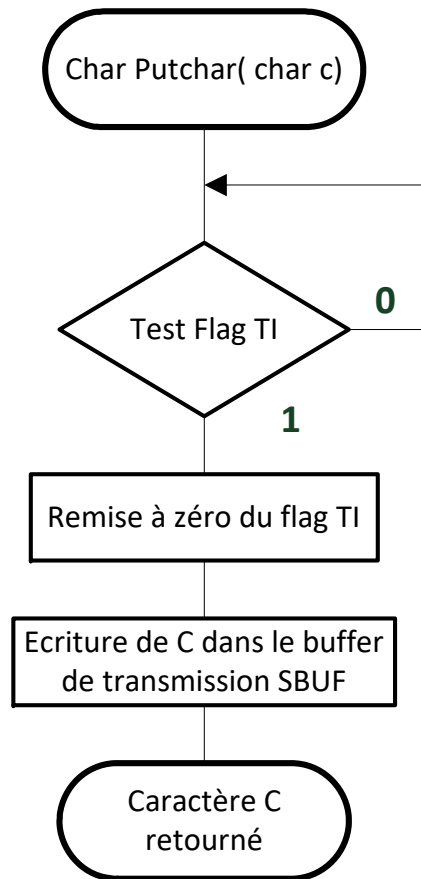
Code Config UART0

Fonction CFG_uart0_model

L'ordre de ces opérations est importante (surtout sur PCON/SCON0)

- Sélection de la source d'horloge (Timer1) pour l'UART – Action sur T2CON
- Configuration de PCON – UART0 Baud Rate Doubler Disabled et accès à SM20-SM00 de SCON0
- Configuration de SCON0 – Mode 1 – Check Stop bit – Réception validée

Code de char putchar(char)



Durée max de la scrutation: le temps de transmission d'un caractère si la configuration est correcte; infinie en cas de mauvaise configuration!!

Config SCON0 Bits 4....0

Si utilisation du Puchar
précédemment codé
« Amorçage » de la transmission

Bit4:	REN0: Receive Enable. This bit enables/disables the UART0 receiver. 0: UART0 reception disabled. 1: UART0 reception enabled.
1	
Bit3:	TB80: Ninth Transmission Bit. The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.
X	
Bit2:	RB80: Ninth Receive Bit. The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM20 is logic 0, RB80 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.
X	
Bit1:	TI0: Transmit Interrupt Flag. Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.
1!!!	
Bit0:	RI0: Receive Interrupt Flag. Set by hardware when a byte of data has been received by UART0 (as selected by the SM20 bit). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.
0	

UART0 avancée

Frame Error: Erreur
de Trame: Stop bit
non valide

RX Overrun: le
caractère reçu n'a pas
été lu à temps

TX Collision: écriture dans
SBUF avant la fin de la
transmission en cours

Des drapeaux supplémentaires dans SCON0 permettent la détection d'erreurs....

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
SM00/FE0	SM10/RXOV0	SM20/TXCOL0	REN0	TB80	RB80	TI0	RI0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

20.3. Frame and Transmission Error Detection

Frame error detection is available in the following modes when the SSTAT0 bit in register PCON is set to logic 1.
Note: The SSTAT0 bit must be logic 1 to access any of the status bits (FE0, RXOVR0, and TXCOL0). To access the UART0 Mode Select bits (SM00, SM10, and SM20), the SSTAT0 bit must be logic 0.

All Modes:

The Transmit Collision bit (TXCOL0 bit in register SCON0) reads '1' if user software writes data to the SBUF0 register while a transmit is in progress. Note that the TXCOL0 bit also functions as the SM20 bit when the SSTAT0 bit in register PCON is logic 0.

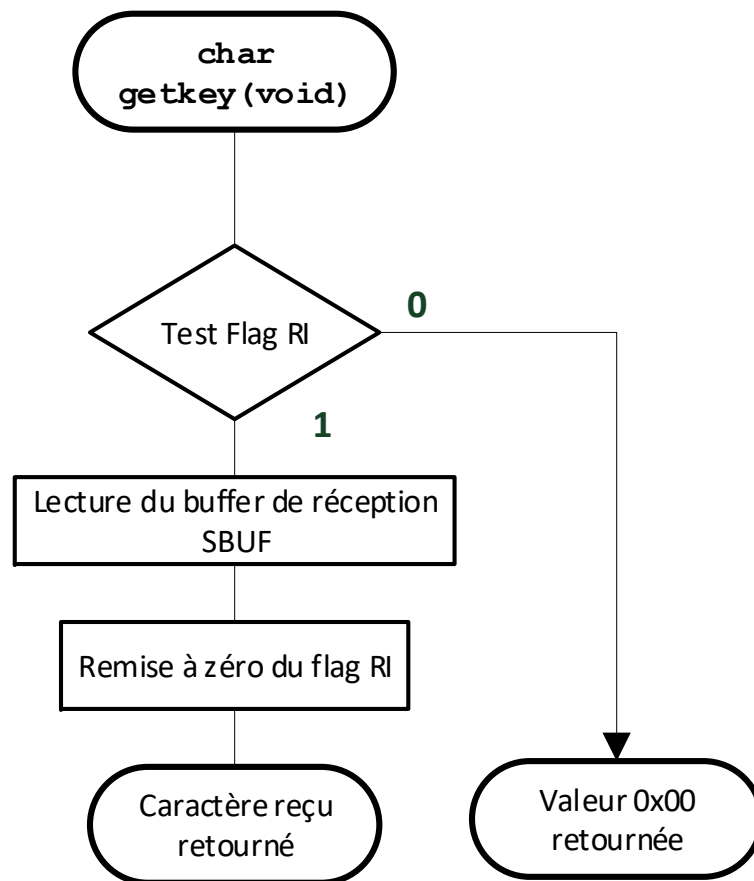
Modes 1, 2, and 3:

The Receive Overrun bit (RXOVR0 in register SCON0) reads '1' if a new data byte is latched into the receive buffer before software has read the previous byte. Note that the RXOVR0 bit also functions as the SM10 bit when the SSTAT0 bit in register PCON is logic 0.

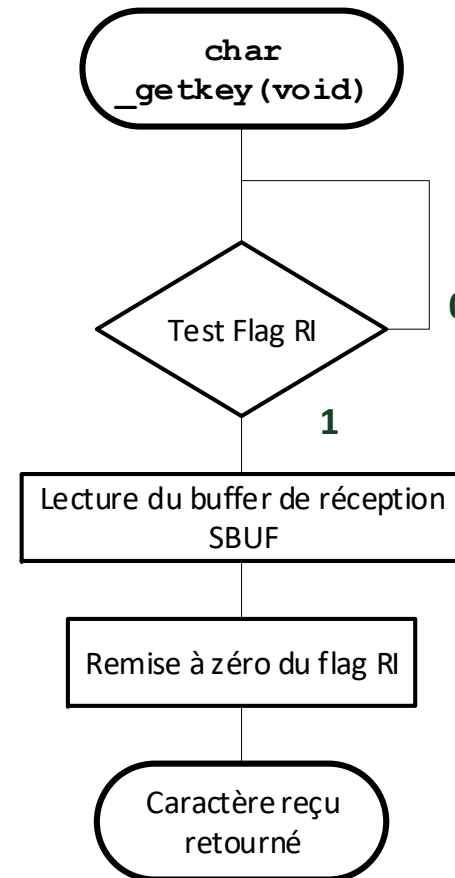
The Frame Error bit (FE0 in register SCON0) reads '1' if an invalid (low) STOP bit is detected. Note that the FE0 bit also functions as the SM00 bit when the SSTAT0 bit in register PCON is logic 0.

Code de `char _Getkey(void)` ou `Getkey(void)`

Version non bloquante



Version bloquante Utilisée par les fonctions telles que `getchar`, `gets`, `scanf`





LIVE AND
DISCOVER

Contact

- François JOLY
- CPE Lyon
- Domaine Scientifique de la Doua
- 43, bd du 11 novembre 1918 – Bâtiment Hubert Curien
- B.P. 2077 – 69616 Villeurbanne cedex – France

Tél. : 04 72 43 13 36
Francois.joly@cpe.fr