

La technologie bit-banging (bit-frappé) : [http://en.wikipedia.org/wiki/Bit\\_banging](http://en.wikipedia.org/wiki/Bit_banging)

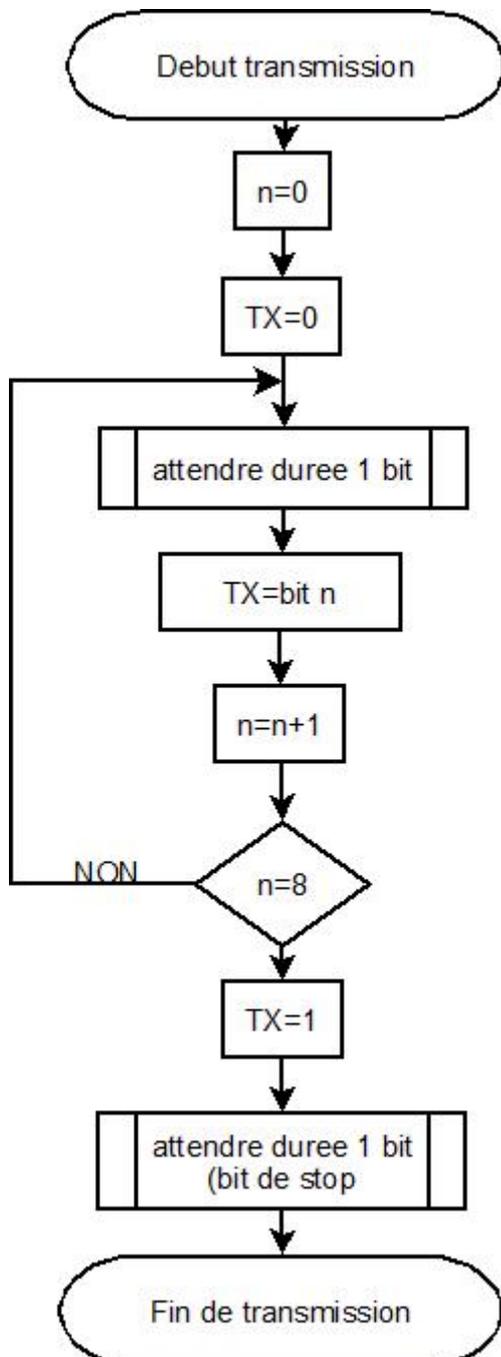
## UART Bit Banged sur PIC18

Les microcontrôleurs PIC18 possèdent un périphérique de communications asynchrones (UART) matériel.

IL est généralement préférable de l'utiliser. Néanmoins il est parfois nécessaire de disposer d'un deuxième UART, certains microcontrôleurs en possèdent deux, mais dans la plupart des cas l'utilisation de la technologie bit-banged sera nécessaire.

Le bit-banging consiste à gérer une interface série (UART, I2C, SPI ...) par logiciel.

Exemple de bit-banged pour une transmission NRZ par UART :



## BBUART

La bibliothèque BBUART remplit cette fonction, deux versions sont proposées :

- BBUART, sans interruption,
- **BBUARTit, avec interruption sur RX (INT0)**  
cette solution est préférable si RX est nécessaire, lors de l'arrivée d'un caractère une interruption est générée (front descendant sur INT0), le caractère est alors récupéré. Le microcontrôleur n'a donc pas à scruter en permanence la ligne RX. *En revanche il est obligatoire de placer RX sur RBO.*

```
void initBBUARTit(void);
```

initialise les ports choisis et active l'interruption INT0 sur RBO

```
unsigned char getBBUART(void);
```

recupère le premier octet disponible dans le tampon de reception

```
char carBBUSARTdispo(void);
```

indique si le tampon de reception est vide (0) ou si au moins un caractère est arrive (1)

```
void ecritBBUART( unsigned char);
```

emet un caractère

```
unsigned char *getstBBUART(unsigned char *s, unsigned char finst);
```

récupère une chaine de caractères se terminant par (finst), il est ainsi possible de remplacer le caractère '0' de fin de chaine ASCII par un autre.

### Exemple :

The screenshot displays the MPLAB IDE environment. The main window shows the source code for `tst_bb_uart_it.c`. The code includes headers for the PIC18F2620, the BBUARTit library, and `stdio.h`. It defines a `chaine[40]` array and a `putch` function that uses `ecritBBUART`. The `main` function configures the oscillator to 32MHz, initializes the BBUARTit library, and enters a loop that checks for received characters using `carBBUSARTdispo` and prints them using `putch`. A build window at the bottom left shows a successful compilation message.

On the right, the Proteus VSM MPLAB Viewer shows a circuit diagram for a PIC18F2620. The PIC is connected to a 10k resistor (R1) and a 100nF capacitor (C3) to VDD and VSS. The PIC pins are labeled with their functions: RA0/AN0/C1IN, RA1/AN1/C2IN, RA2/AN2/C1IN+VREF-/CVREF, RA3/AN3/C1IN+VREF+, RA4/TOCLK/C1OUT, RA5/AN4/S/H/LVDIN/C2OUT, RA6/OSC2/CLKO, RA7/OSC1/CLKI, RB0/AN12/INT0/FLTO, RB1/AN10/INT1, RB2/AN8/INT2, RB3/AN9/C0F2A, RB4/AN11/KBD, RB5/KB1/P0M, RB6/KB2/P0C, RB7/KB3/P0D, and RE3/MCLR/VPP.