



# TP : INFORMATIQUE INDUSTRIELLE N°8

# Réception d'information sur module LORA

# Réception d'information sur module LORA

## 1) Présentation

La technologie LoRa signifie Long Range ou « longue portée » en français. Il s'agit d'un procédé de modulation qui permet aux objets connectés d'échanger des données de faible taille en bas débit. **LoRaWAN** est un protocole réseau fondé sur la technologie LoRa. Dans le cadre de l'Internet des objets, il permet de structurer un réseau étendu à basse consommation (LPWAN), intégrant des équipements terminaux à faible consommation électrique par l'intermédiaire de passerelles.

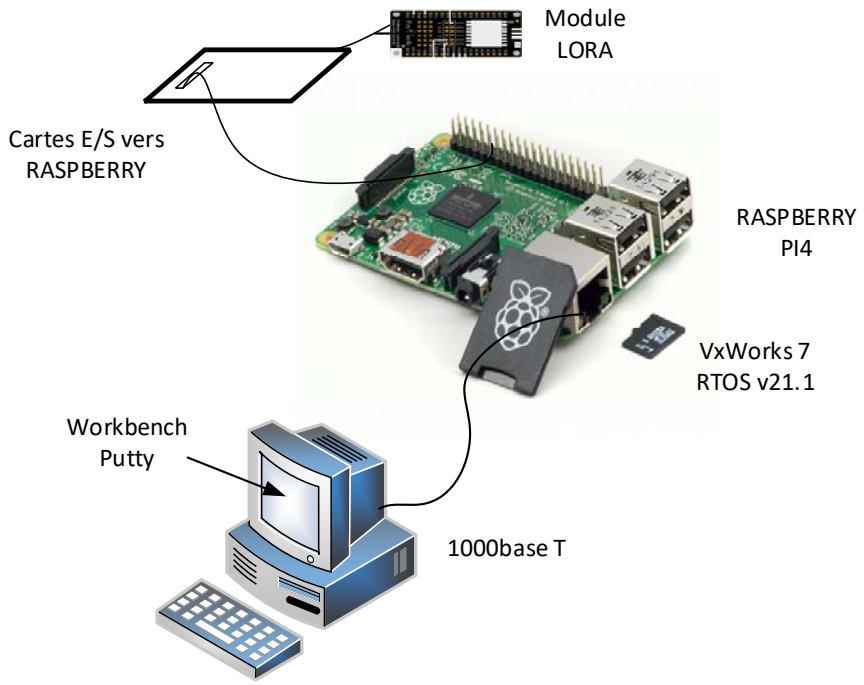
## 2) Objectifs

- Découvrir le procédé de modulation LORA
- Fournir les éléments logiciels afin d'émettre ou de recevoir avec un module LORA
- S'initier aux protocoles de communication SPI

## 3) Pré-requis

- Cours sur les systèmes temps réel et les événements
- Connaissance des outils de développement.
- Dossier ressource sur VxWorks 7 et le bus SPI

## 4) Schéma de câblage




## 5) Le procédé de modulation LORA

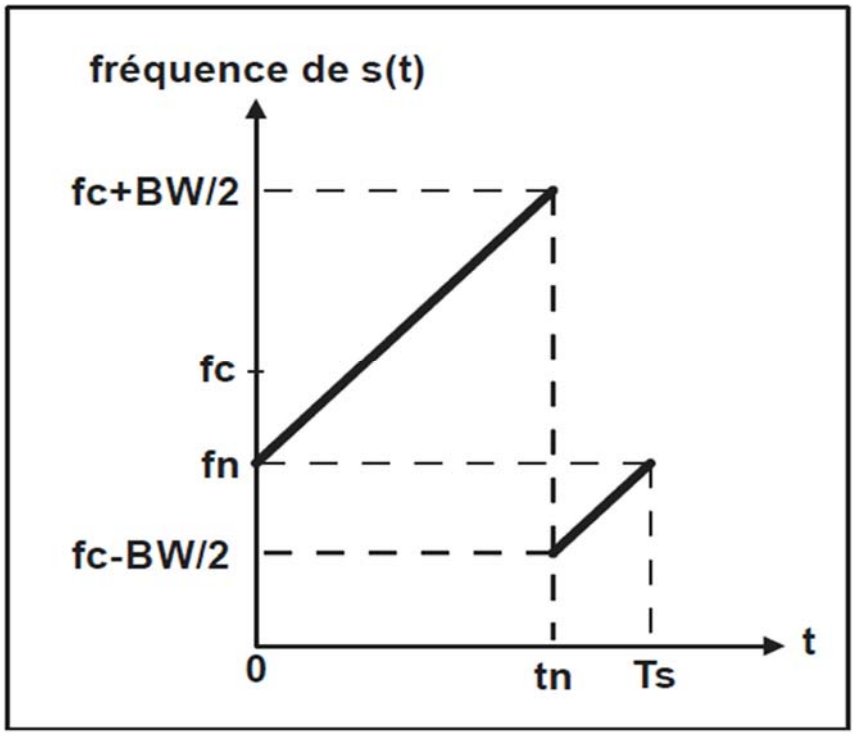
**Les caractéristiques du signal modulé s(t)**

Le modulateur LoRa permet, à partir du message numérique m(t) à transmettre (signal modulant), d'élaborer le signal radio modulé s(t) de caractéristiques conformes au **standard radio LoRa.**




Le signal s(t) est un signal sinusoïdal modulé en fréquence dont la fréquence évolue

linéairement au cours du temps (signal que l'on désigne par le terme *chirp*).
Le signal modulant m(t) est numérique et constitué d'une succession de symboles n exprimés en décimal et transmis les uns à la suite des autres.
La figure ci-dessous représente l'évolution de la fréquence du signal modulé s(t) au cours de la transmission d'un symbole n constitutif du message m(t) :




la fréquence f du signal modulé s(t) évolue de part et d'autre d'une fréquence porteuse fc = 1/Tc, entre fmin = fc - BW/2 et fmax = fc + BW/2, où BW (BandWidth) correspond à la variation de fréquence du signal modulé ;
- fn correspond à la valeur de la fréquence de s(t) à l'instant t = 0 de début de transmission du symbole n ;
- tn correspond à l'instant où le signal modulé s(t) présente une discontinuité de fréquence ;
- TS (période symbole) correspond à la durée de transmission d'un symbole n.

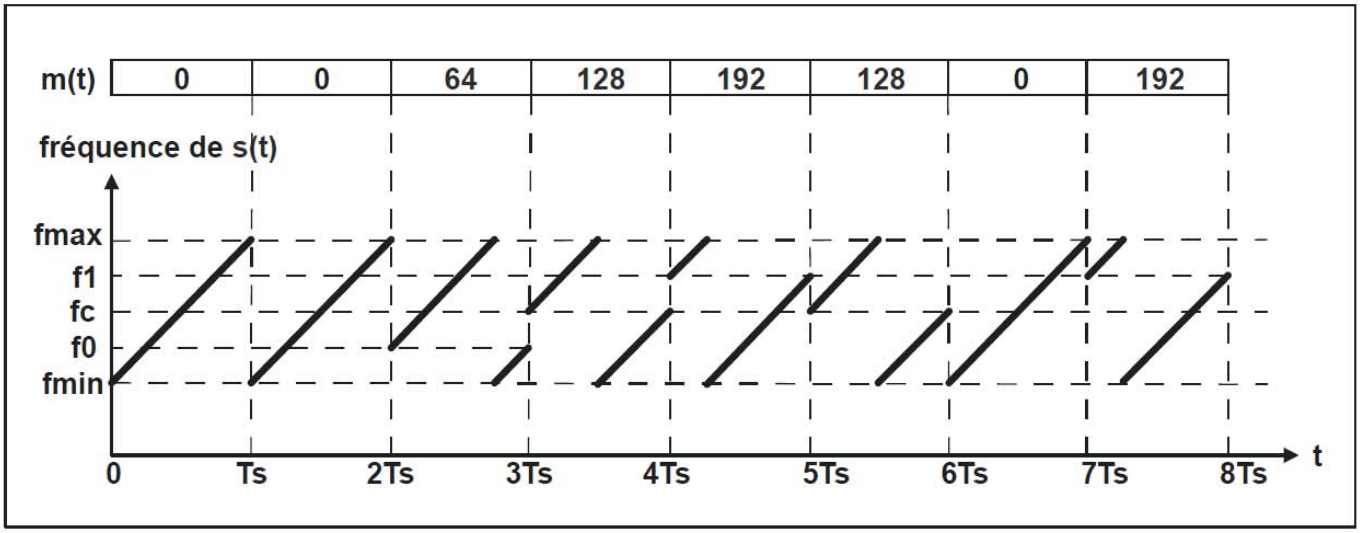
On précise les caractéristiques du signal module s(t) telles que définies par l'alliance LoRa :
- les symboles n sont codés en binaire sur un nombre de bits noté SF (Spreading Factor) pouvant prendre les 6 valeurs distinctes 7, 8, 9, 10, 11 et 12 ;
- les valeurs de BW normalisées sont 125 kHz et 250 kHz ;
- les 7 combinaisons possibles de valeurs de SF et de BW, sont les suivantes :

| SF | BW (kHz) |
|---|---|
| 7 | 250 |
| 7 | 125 |
| 8 | 125 |
| 9 | 125 |
| 10 | 125 |
| 11 | 125 |
| 12 | 125 |

les valeurs de SF et de TS sont liées par la relation

$$TS = \frac{2^{SF}}{BW}$$

En Europe, le standard LoRa, impose 3 canaux distincts, caractérisés par les 3 valeurs de fréquence porteuse fc égales à 868,1 MHz, 868,3 MHz et 868,5 MHz,
mais chaque opérateur peut ajouter à ces 3 canaux, 5 canaux supplémentaires de fréquences porteuses fc égales à 867,1 MHz, 867,3 MHz, 867,5 MHz, 867,7 MHz et 867,9 MHz.
On donne ci-dessous l'illustration de l'évolution temporelle de la fréquence du signal modulé s(t) correspondant à la transmission par voie radio des 8 symboles successifs 0, 0, 64, 128, 192, 128, 0 et 192, ces derniers correspondant à des nombres non signés codés sur 8 bits (SF = 8).



Le projet Workbench LORAR_DKM est architecturé autour de deux fichiers sources :

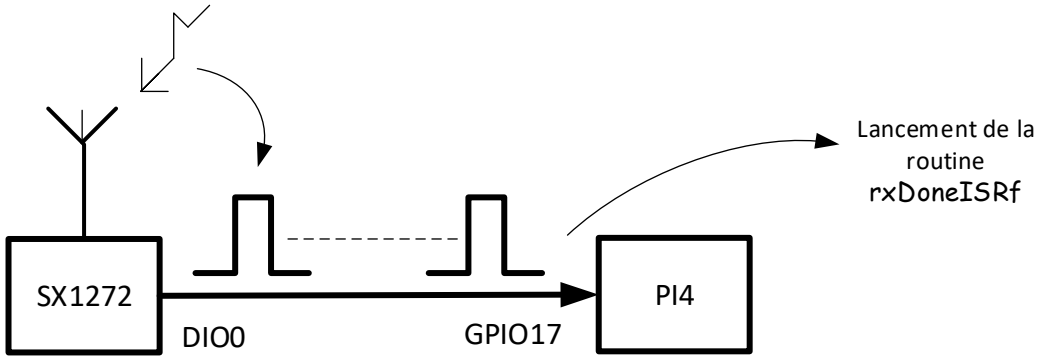
- ✓ Lora.c ou on retrouve les fonctions de bas niveau permettant de configurer et programmer le contrôleur LORA SX1272 grâce au bus SPI
- ✓ Lorareception.c qui va lancer deux taches noyaux temps réel de même priorité :

  - Rspi qui permet d'initialiser les différents registres du contrôleur SX1278 de lancer la réception des informations sur la liaison radio et d'envoyer un événement à la tache handleEvents pour sortir du programme
  - handleEvents cette tache va permettre de gérer la réception des événements et notamment l'événement reçu lorsque le controleur SX1278 a réceptionné des informations sur sa liaison radio

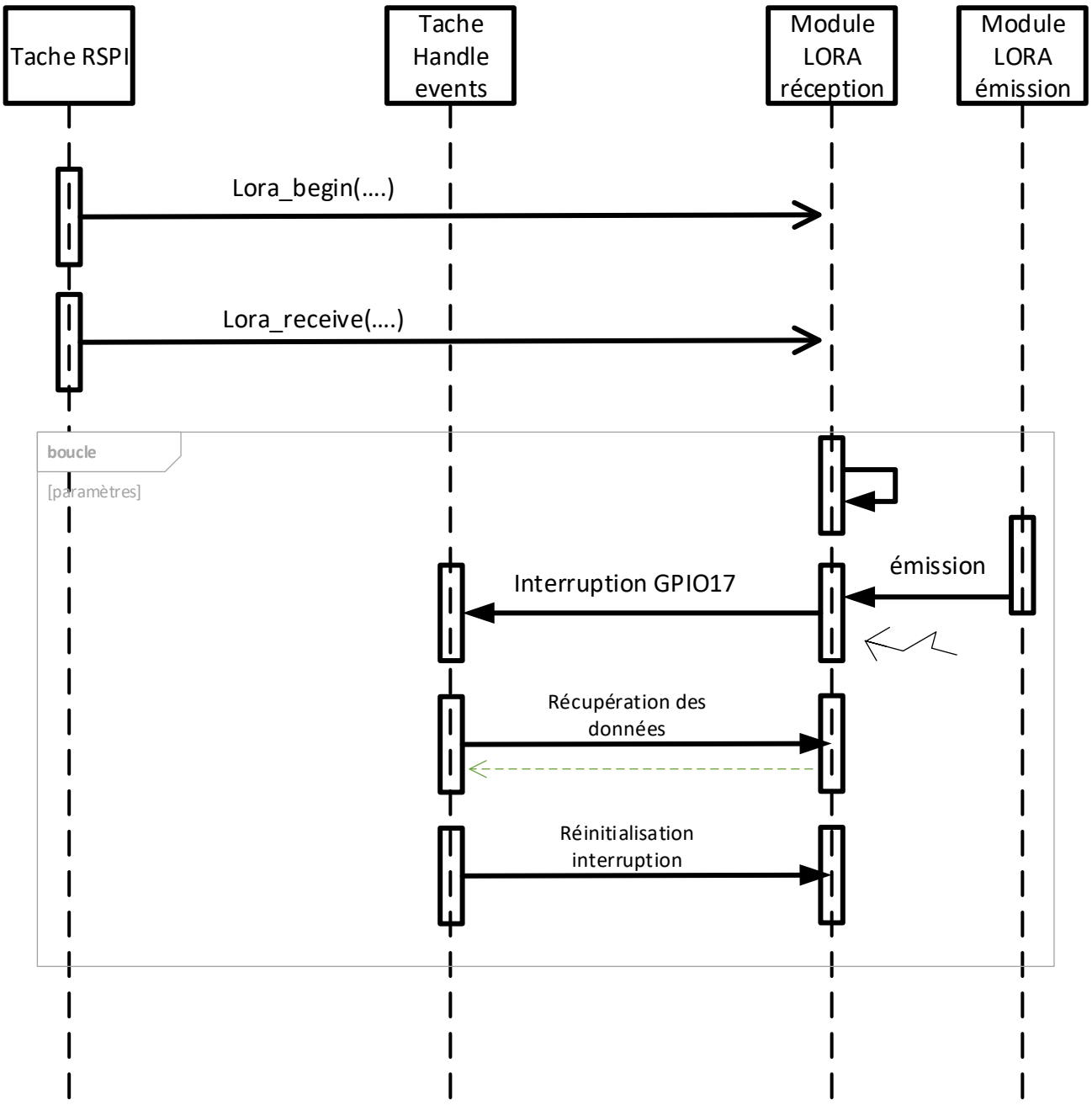
Le contrôleur SX1278 est programmé de façon à générer sur sa broche DIO0 une impulsion à chaque réception d'information sur la liaison radio.
Dans la fonction Lora_receive , l'appel de la fonction

lora_set_rxdone_dioISR(modem->eth.dio0GpioN, rxDoneISRf, modem)

permet de programmer une interruption sur la broche GPIO17 de la PI4 et d'associer l'ISR rxDoneISRf à cette interruption.




La séquence est la suivante :




## 6) Questions préliminaires

6.1) Vérifiez la câblage (voir dossier annexe 1)

6.2) En vous aidant de la fonction Rspi en déduire les paramétres de configuration suivants :

6.3) En déduire le débit symbole DS ,le débit binaire Db et le debit utile Du

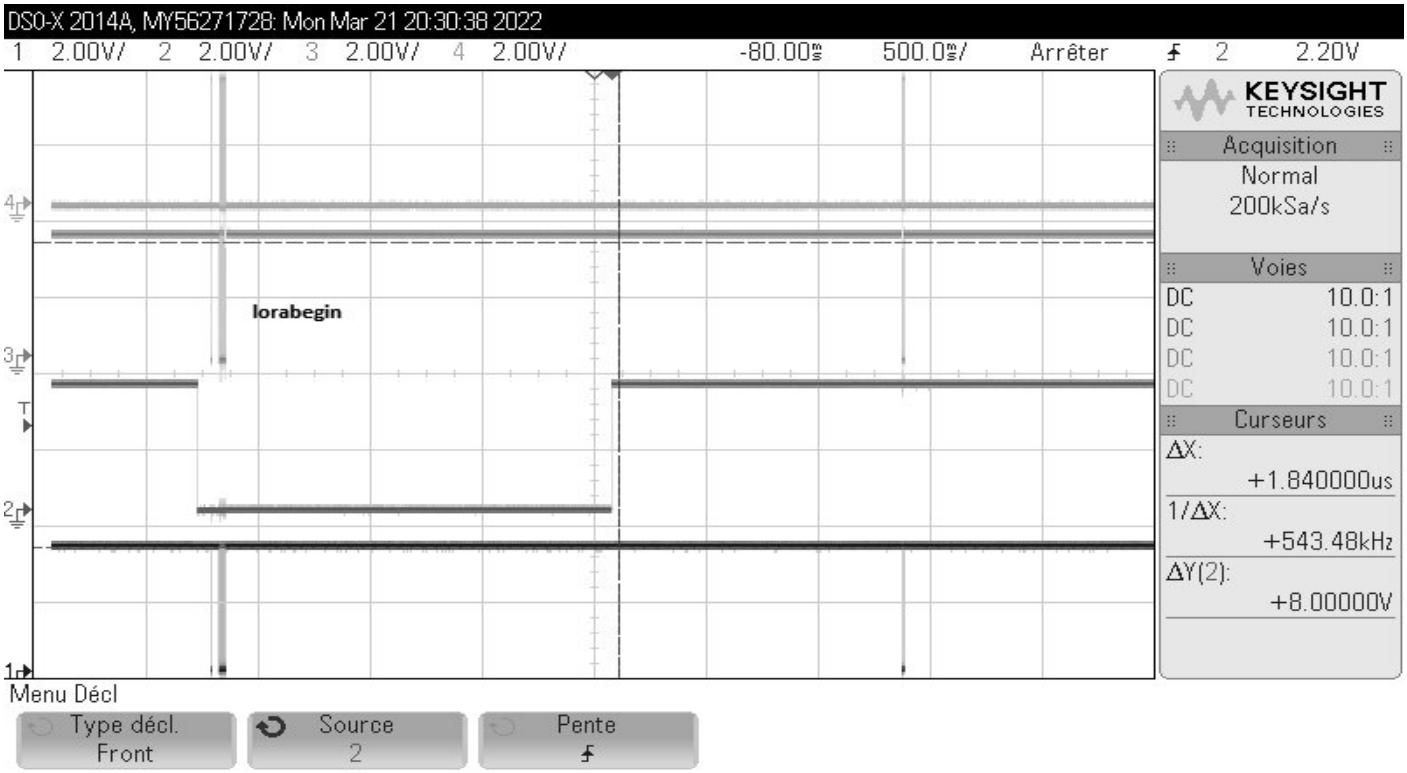
La fonction ISR rxDoneISRf est appelée à chaque interruption pour chaque réception de données . Une fonction d'interruption sous VxWorks ne peut faire d'appel système pour récupérer les données (trop gourmand en temps et peu déterministe pour un RTOS). Un moyen est de dédier cette récupération de données à une tache noyau . Cette tache noyau sera prévenue par l'ISR grâce au mécanisme des événements. On se propose dans cette partie de mettre en place ce mécanisme d'événement.

## 7) Gestion des événements

7.1) Complétez la fonction rxDoneISRf afin d'envoyer un événement du type VXEV01 à la tache associée à la fonction **handleEvents**

7.2) Modifiez la foncion handleEvents afin de recevoir et d'être sensible aux événements du type `VXEV01,VXEV02,VXEV03` . Décomnentez le bloc de commentaire dans cette fonction

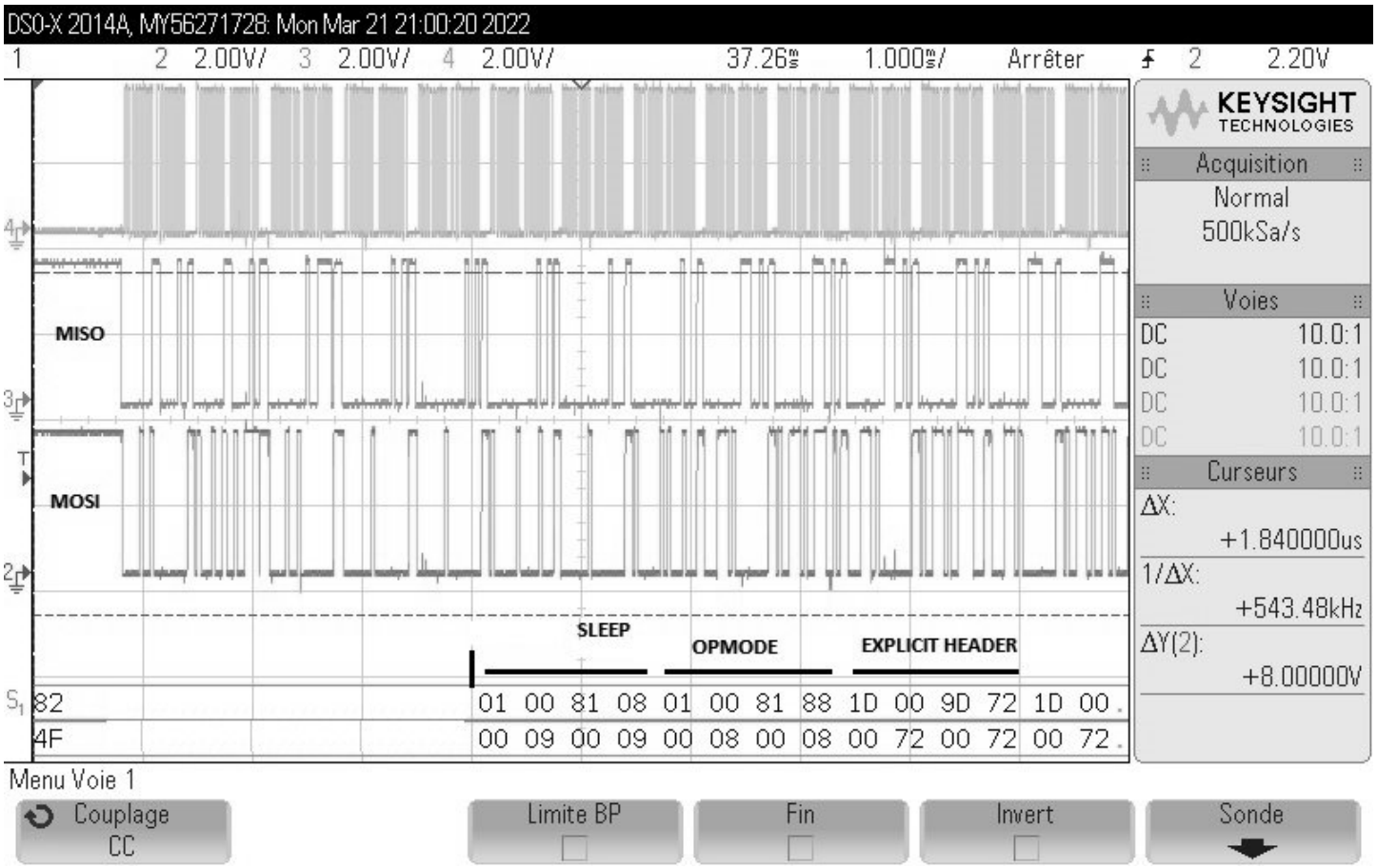
7.3) Testez votre programme (l'émetteur doit être actif) et relevez à l'oscilloscope au démarrage les signaux sur le bus SPI



La fonction Lora_begin démarre par la fonction lora_set_sleep_mode. Cette fonction agit sur le bus SPI par l'intermédiaire de deux autres fonctions en surgras

**lora_reg_write_byte**(spid, REG_OP_MODE, (**lora_reg_read_byte**(spid, REG_OP_MODE) & 0xf8) | SLEEP_MODE );

Le début de la trame est présenté ci-dessous :




7.4) Identifiez sur la trame les deux octets générés sur MOSI par la fonction **lora_reg_read_byte**

7.5) Donnez la valeur numérique retournée par la fonction **lora_reg_read_byte**

7.4) Identifiez sur la trame les deux octets générés sur MOSI par la fonction **lora_reg_write_byte**

7.5) En analysant le registre RegOpmode dossier annexe 2 , quel impact l'execution de la fonction lora_set_sleep_mode a eu sur le circuit SX1278

On souhaite obtenir du contrôleur SX1278 la valeur du rapport signal sur bruit en db et du RSSI à l'instant t.

**RSSI (Received Signal Strength Indication )** est une mesure du niveau de puissance en réception d'un signal issu d'une antenne (un signal radiofréquence). Il est exprimée en dbm (une valeur de 0 dBm correspond à une puissance reçue de 1 mW)

$$P_{dbm} = 10 \times log_{10} \frac{Precue(w)}{1mw}$$

Une première fonction nous permettra d'obtenir la valeur du RSSI courant

**void** `lora_get_rssi_pkt` (LoRa_ctl *modem);

Elle mettra à jour la variable `RSSI de la structure du type rxData`

Une deuxième fonction nous permettra d'obtenir la valeur du rapport signal sur bruit

**void lora_get_snr**(LoRa_ctl *modem);

Elle mettra à jour la variable `SNR de la structure du type rxData`

## 8) Contrôle de la qualité de la réception

8.1) On donne le code de la fonction **lora_get_rssi_pkt**, expliquez en quelques lignes ce que réalise cette ligne de code (on pourra s'appuyer sur le dossier annexe 2)

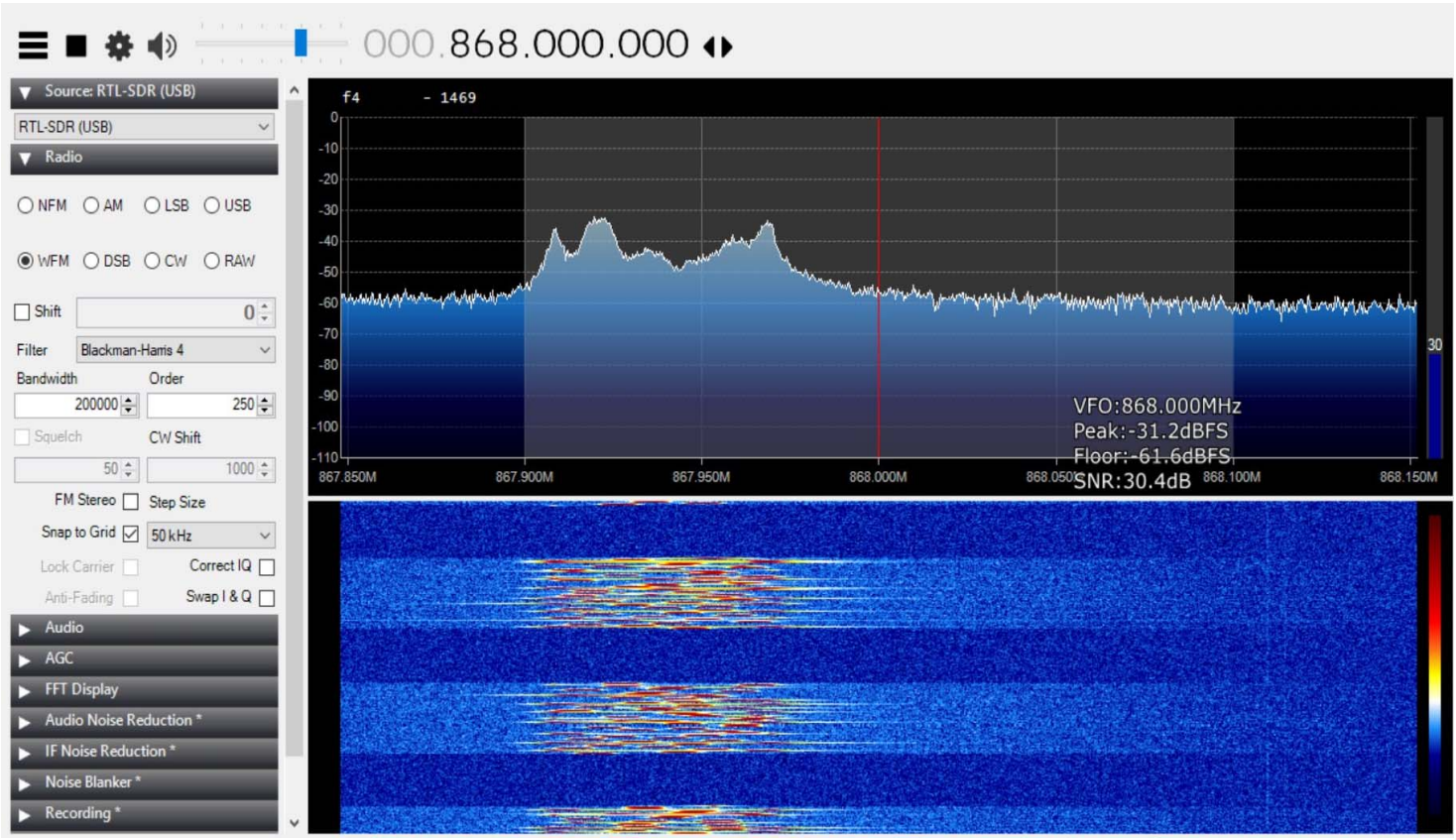
```
void lora_get_rssi_pkt(LoRa_ctl *modem){
    modem->rx.data.RSSI = lora_reg_read_byte(modem->spid, REG_PKT_RSSI_VALUE) - (modem->eth.freq
< 779E6 ? 164 : 157);
}
```

8.2) Fournir les éléments de code pour la fonction **lora_get_snr** dans le fichier Lora.c et déclarer la fonction dans le fichier lora.h

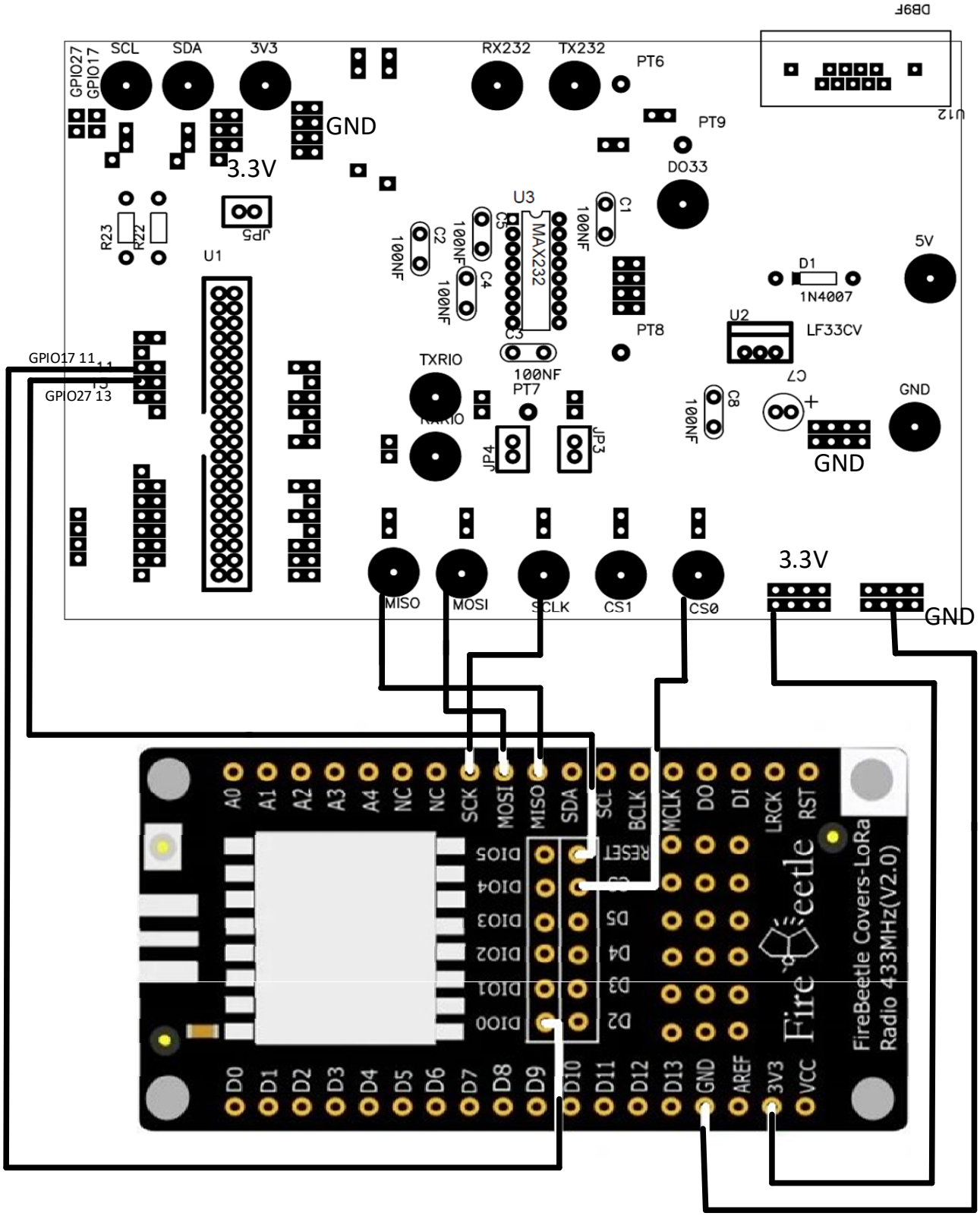
8.3) Afficher par un printf dans la fonction rx_f les valeurs du RSSI et du SNR

8.4) Testez votre programme

8.5) On obtient sur SDRSHARP l'onde de forme du signal ainsi que son spectre ci-dessous. On émet très peu de caractère. Pourquoi obtient on plus de saut de fréquence que prévu ?



# Dossier annexe N°1



# Dossier annexe N°2

## Architecture interne du SX1278

### 1.1. Simplified Block Diagram

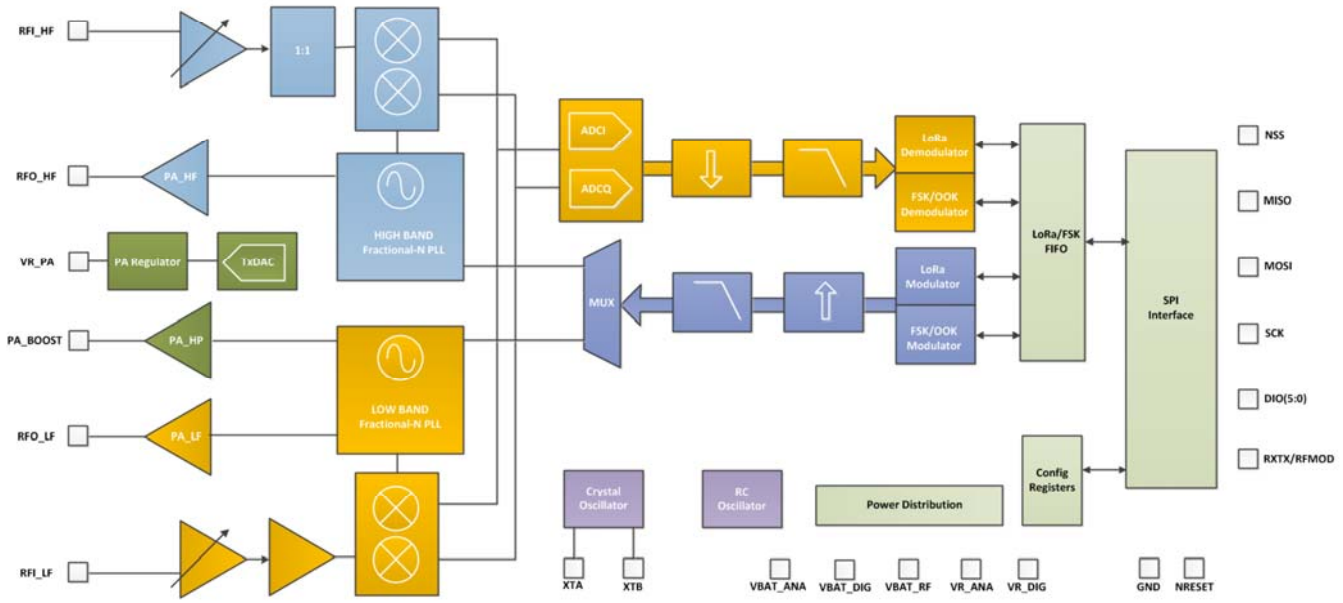
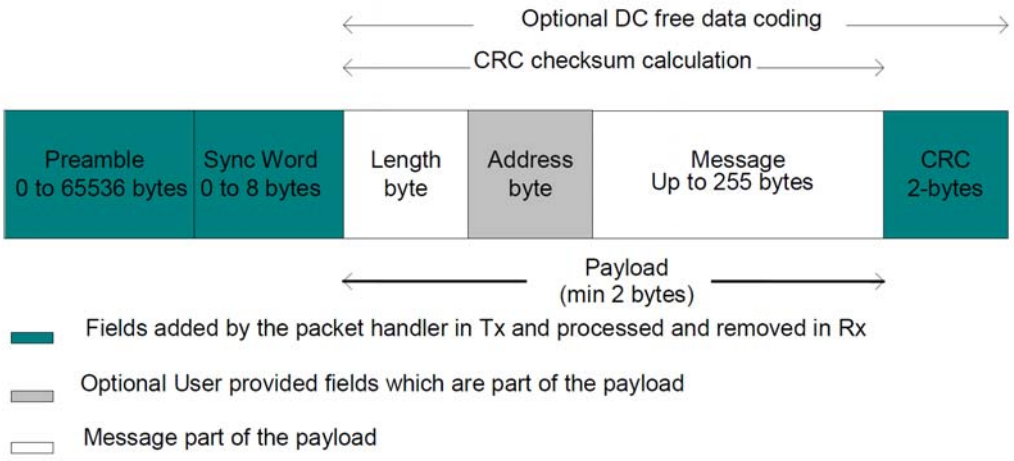



*Figure 1. Block Diagram*

## Constitution d'une trame SX1278



## Les 30 premiers registres SX1278

| Address | Register Name | | Reset (POR) | Default (FSK) | Description | |
|---|---|---|---|---|---|---|
| | FSK/OOK Mode | LoRa™ Mode | | | FSK Mode | LoRa™ Mode |
| 0x00 | RegFifo | | 0x00 | | FIFO read/write access | |
| 0x01 | RegOpMode | | 0x01 | | Operating mode & LoRa™ / FSK selection | |
| 0x02 | RegBitrateMsb | Unused | 0x1A | | Bit Rate setting, Most Significant Bits | |
| 0x03 | RegBitrateLsb | | 0x0B | | Bit Rate setting, Least Significant Bits | |
| 0x04 | RegFdevMsb | | 0x00 | | Frequency Deviation setting, Most Significant Bits | |
| 0x05 | RegFdevLsb | | 0x52 | | Frequency Deviation setting, Least Significant Bits | |
| 0x06 | RegFrfMsb | | 0x6C | | RF Carrier Frequency, Most Significant Bits | |
| 0x07 | RegFrfMid | | 0x80 | | RF Carrier Frequency, Intermediate Bits | |
| 0x08 | RegFrfLsb | | 0x00 | | RF Carrier Frequency, Least Significant Bits | |
| 0x09 | RegPaConfig | | 0x4F | | PA selection and Output Power control | |
| 0x0A | RegPaRamp | | 0x09 | | Control of PA ramp time, low phase noise PLL | |
| 0x0B | RegOcp | | 0x2B | | Over Current Protection control | |
| 0x0C | RegLna | | 0x20 | | LNA settings | |
| 0x0D | RegRxConfig | RegFifoAddrPtr | 0x08 | 0x0E | AFC, AGC, ctrl | FIFO SPI pointer |
| 0x0E | RegRssiConfig | RegFifoTxBaseAddr | 0x02 | | RSSI | Start Tx data |
| 0x0F | RegRssiCollision | RegFifoRxBaseAddr | 0x0A | | RSSI Collision detector | Start Rx data |
| 0x10 | RegRssiThresh | FifoRxCurrentAddr | 0xFF | | RSSI Threshold control | Start address of last packet received |
| 0x11 | RegRssiValue | RegIrqFlagsMask | n/a | n/a | RSSI value in dBm | Optional IRQ flag mask |
| 0x12 | RegRxBw | RegIrqFlags | 0x15 | | Channel Filter BW Control | IRQ flags |
| 0x13 | RegAfcBw | RegRxNbBytes | 0x0B | | AFC Channel Filter BW | Number of received bytes |
| 0x14 | RegOokPeak | RegRxHeaderCntValueMsb | 0x28 | | OOK demodulator | Number of valid headers received |
| 0x15 | RegOokFix | RegRxHeaderCntValueLsb | 0x0C | | Threshold of the OOK demod | |
| 0x16 | RegOokAvg | RegRxPacketCntValueMsb | 0x12 | | Average of the OOK demod | Number of valid packets received |
| 0x17 | Reserved17 | RegRxPacketCntValueLsb | 0x47 | | - | |
| 0x18 | Reserved18 | RegModemStat | 0x32 | | - | Live LoRa™ modem status |
| 0x19 | Reserved19 | RegPktSnrValue | 0x3E | | - | Espimation of last packet SNR |
| 0x1A | RegAfcFei | RegPktRssiValue | 0x00 | | AFC and FEI control | RSSI of last packet |

| Address | Register Name | | Reset (POR) | Default (FSK) | Description | |
|---|---|---|---|---|---|---|
| | FSK/OOK Mode | LoRa™ Mode | | | FSK Mode | LoRa™ Mode |
| 0x1B | RegAfcMsb | RegRssiValue | 0x00 | n/a | Frequency correction value of the AFC | Current RSSI |
| 0x1C | RegAfcLsb | RegHopChannel | 0x00 | n/a | | FHSS start channel |
| 0x1D | RegFeiMsb | RegModemConfig 1 | 0x00 | n/a | Value of the calculated frequency error | Modem PHY config 1 |
| 0x1E | RegFeiLsb | RegModemConfig 2 | 0x00 | n/a | | Modem PHY config 2 |

| | | | | | |
|---|---|---|---|---|---|
| RegOpMode (0x01) | 7 | LongRangeMode | rw | 0x0 | 0 → FSK/OOK Mode<br>1 → LoRa™ Mode<br>This bit can be modified only in Sleep mode. A write operation on other device modes is ignored. |
| | 6 | AccessSharedReg | rw | 0x0 | This bit operates when device is in Lora mode; if set it allows access to FSK registers page located in address space (0x0D:0x3F) while in LoRa mode<br>0 → Access LoRa registers page 0x0D: 0x3F<br>1 → Access FSK registers page (in mode LoRa) 0x0D: 0x3F |
| | 5-4 | reserved | r | 0x00 | reserved |
| | 3 | LowFrequencyModeOn | rw | 0x01 | Access Low Frequency Mode registers<br>0 → High Frequency Mode (access to HF test registers)<br>1 → Low Frequency Mode (access to LF test registers) |
| | 2-0 | Mode | rwt | 0x01 | Device modes<br>000 → SLEEP<br>001 → STDBY<br>010 → Frequency synthesis TX (FSTX)<br>011 → Transmit (TX)<br>100 → Frequency synthesis RX (FSRX)<br>101 → Receive continuous (RXCONTINUOUS)<br>110 → receive single (RXSINGLE)<br>111 → Channel activity detection (CAD) |

| | | | | | |
|---|---|---|---|---|---|
| RegRssiValue (0x1B) | 7-0 | Rssi | r | n/a | Current RSSI value (dBm)<br>RSSI[dBm] = -157 + Rssi (using HF output port)<br>or<br>RSSI[dBm] = -164 + Rssi (using LF output port)<br>(see section 5.5.5 for details) |

| | | | | | |
|---|---|---|---|---|---|
| RegPktSnrValue (0x19) | 7-0 | PacketSnr | r | n/a | Estimation of SNR on last packet received.In two's compliment format mutiplied by 4.<br>$SNR[dB] = \frac{PacketSnr[twos\ complement]}{4}$ |

# Les commandes Vxworks

## Les commandes usuelles

| | |
|---|---|
| C | pour revenir à l'interpréteur de code de l'interpréteur de commande |
| cmd | pour revenir à l'interpréteur de commande de l'interpréteur de code |
| ifconfig -a | Info sur la config réseau |
| ls | liste les noms des fichiers dans le répertoire |
| ls -l | |
| i | Info sur les taches exécutées |
| pwd | Info sur notre répertoire et l'arborescence |
| td <nomdelatache> | tuer une tache |
| ti <nomdelatache> | info sur une tache |
| h | affiche les dernières commandes tapées |
| printLogo | affiche le logo Vxworks |
| ld <nomdumodule> | charger un module |
| unld <nomdumodule> | pour décharger un module |
| cd / | revenir à la racine |
| devs | liste les périphériques gérés |
| cp | copiez un fichier d'un répertoire à un autre (idem Linux) |
| mkdir <nomrépertoire> | créez un répertoire |
| netstat -a | Information sur les ports ouverts |
| ping <adresse IP> | ping une adresse IP |
| version | pour connaitre la version de Vxworks |
| ipd list | liste des serveurs daemon |
| ipd kill | |
| ipd start | |
| sdShow | liste des SHared data |

esc K - go back in history
esc J - go forward in history

## Les commandes non usuelles

| | |
|---|---|
| moduleShow | voir la liste des modules chargées |
| lkup <name> | Donne la table des symboles (fonctions) et variables chargés par le noyau |
| semShow | Donne la liste des sémaphores |
| iosDrvShow | Donne la liste des drivers de périphériques chargés et leur numéro |
| vxbDrvShow | Donne la liste des drivers de périphériques chargés et leurs classes |
| vxbDevShow | Donne la liste des drivers de périphériques chargés et leurs classes |
| vxbDevShowPaths | Donne la liste des drivers de périphériques et leurs chemins dans le device tree |
| iosFdShow | Donne la liste des descripteurs de périphériques |
| vxFdtDtbDump | Donne la liste du device tree DTB |
| vxbIntShow | Donne la liste des interruptions gérées le numéro de vecteur |
| isrShow | Donne la liste des ISR et des numéros de vecteurs associés |
| vxbPinMuxShow | liste les contrôleurs de pinmux |
| vxbGpioShow | liste les broches du GPIO monopolisées |
| iosDevShow | |
| rtpShow | liste les processus real time processus RTP chargés |
| adrSpaceShow | info sur la mémoire |
| sp | Spawn a task |

```
rtpSp  *.vxe
checkStack()
```

sdShow