Traduction de :
« Tutorial 0: Everything You Need to Know about the nRF24L01 and MiRF-v2 »
Auteur : Brennen Ball

Introduction

Si vous pensez que la connexion sans fil entre deux ou plusieurs microcontrôleurs est difficile, je vous répondrai que cela dépend largement du matériel utilisé.

Je vais vous montrer qu'il n'est ni difficile, ni coûteux de mettre en place une liaison sans fil de qualité. Vous finirez par vous demander pourquoi vous avez utilisé des câbles!

Entrez dans le monde du circuit nRF24L01, transmetteur 2G4 de nos amis de Nordic Semiconductor. Ce circuit, mis sur le marché il y a quelques années, est le successeur du célèbre nRF2401. Le 24L01 à repris ce qui avait de mieux dans le 2401, en y ajoutant une interface SPI matérielle, ainsi que de multiples buffers de communication et plus encore. De plus, ce circuit est moins cher que la plupart des microcontrôleurs, et il ne nécessite pas trop de composants externes. Quelle est ma motivation pour écrire ce tutoriel ?

Je ne travaille ni pour Nordic, ni pour Sparkfun ou pour aucune société mentionnée dans ce document.En fait, j'adore les liaisons radio, et je souhaiterais que le plus grand nombre puisse les utiliser. J'espère que ce tutoriel vous aidera dans votre mise en œuvre d'une liaison sans fil.

Informations complémentaires

Ce document contient essentiellement de la description de base du circuit lui-même. J'ai essayé d'être le plus exhaustif possible concernant la partie hardware du circuit, et je fournis quelques suggestions concernant les réglages et l'utilisation du circuit.

Si ce qui vous intéresse est l'utilisation proprement dite du circuit, je vous conseille de passer directement aux tutoriels suivants.

Par contre, si vous êtes débitants dans le domaine, ce tutoriel devrait pouvoir vous aider, et vous permettre de décortiquer les différentes fonction du circuit. Dans tout les cas, ce tutoriel pourra vous servir de référence lors de l'utilisation de ce circuit.

Lecture obligatoire

Oui, je sais que ça ressemble à un devoir, mais vous devez le faire.

Si vous voulez réussir alors de la mise en œuvre de votre projet RF, vous devez absolument connaître le microcontrôleur utilisé, ainsi que le langage avec lequel vous allez le programmer. Et l'idéal est d' avoir les manuels de référence à portée de main

Une description matérielle de votre projet, (dessin du PCB et références des I/O) est indispensable. Enfin, le datasheet du 24L01, ainsi que la documentation du breakoutboard que vous utilisez, est aussi indispensable.

Interface matérielle du nRF24L01

Vous utiliserez plus que probablement une carte du commerce pour votre 24L01.

Dans ce cas, sur cette carte, vous retrouverez huit connexions qui sont Vcc, GND, IRQ, CE, et les quatre broches SPI (CSN, SCK, MISO et MOSI).

Si par contre, vous dessinez vous même votre circuit, je vous renvoie à la figure 13 du datasheet, où vous retrouverez un exemple de circuit. De toutes façon, dans ce tutoriel, je me limiterai à la description des huit pins qui viennent d'être énumérées.

Heureusement pour nous, Nordic à conçu les I/O du circuit pour qu'elles soient 5V tolérantes. Ceci vous permettra de connecter les I/O directement à votre microcontrôleur alimenté en 5V sans griller votre circuit.

Par contre, n'alimentez surtout pas votre 20L01 en 5V, car cela aura sans doute pour conséquence de faire sortir la fumée qu'il contient. Le datasheet nous donne une tension d'alimentation comprise entre 1,9V et 3,6V pour VCC. La plupart du temps, la tension qui sera utilisée sera de 3,3V.

L'interface SPI utilise quatre broches, CSN, SCK, MISO, MOSI et pour la transmission de données et la réception.
La broche CSN, (Chip Select) est active bas (d'où le N). De manière générale, elle est maintenue à l'état haut. Lorsqu'elle passe à l'état bas, le 24L01 commence le traitement des données sur le port SPI.
Les trois broches restantes du port SPI doivent être raccordées à l'interface SPI de votre matériel de la manière suivante : SCK à SCK, MISO à MISO et MOSI à MOSI.

Enfin, les deux broches restantes sont CE et IRQ.
CE est utilisée pour contrôler le mode de fonctionnement en RX ou en TX du 24L01.
IRQ est la broche d'interruption, et est active bas. Il existe trois interruptions internes qui agissent sur cette broche. Ces interruptions peuvent-être masquées, de telle sorte que interruption correspondante n'agisse pas sur la pin IRQ.

Description de la carte MiRF-V2.

Dans ce tutoriel, la carte breakoutboard utilisée est la carte MiRF-V2 de chez Sparkfun Electronics (http://www.sparkfun.com).

Ces cartes contiennent le régulateur de tension et tout le matériel nécessaire au raccordement du 24L01 au microcontrôleur ainsi qu'à l'antenne (lorsqu'elle n'est pas intégrée).

Personnellement, j'ai acheté la carte MiRF-V2, avant que la version RP-SMA ne sorte. Elle est donc équipée d'une antenne intégrée sur le circuit imprimé, ce qui fait que cette carte n'a qu'une portée limitée. Le choix de la carte se fera donc en fonction du budget et de la portée nécessaire. N'oubliez pas d'acheter une antenne si vous prenez la version RP-SMA.

(Note du traducteur : Actuellement, il existe beaucoup de sites chinois, où ces cartes sont disponibles pour quelques euros.)

Figure 1. De gauche à droite: MiRF-v2 haut, MiRF-v2 RP-SMA supérieure et inférieure (photos de www.sparkfun.com)

Commençons par regarder les broches disponibles sur la carte MiRF-v2. Si vous utilisez une autre carte, les broches que je décris porterons le même nom, puisqu'en fait, il s'agit des noms de broches du circuit 24L01 lui-même.

En partant du haut, vous avez Vcc. Cette broche est reliée à l'entrée d'un régulateur de tension, et peut traiter de 3,3 à 7 Vdc par le site Web de Sparkfun. Personnellement, je le connecte à 3.3V parce que ce rail d'alimentation principale que j'utilise avec mon microcontrôleur.Remarque: si vous n'utilisez pas le MiRF-v2, vous devrez sans doute raccorder votre circuit en 3,3V, ou utiliser un convertisseur de tension.

La deuxième broche est nommée CE. Cette broche est toujours une entrée par rapport au 24L01. Suivant l'état dans lequel se trouve le 24L01, cette broche aura une fonction différente.Tout d'abord, nous supposons que l'appareil est mis sous tension en interne (en réglant le bit de PWR_UP dans le registre CONFIG ... nous en parlerons plus tard). A ce moment,si le circuit est en mode réception et si la pin CE est à l'état haut, le circuit scrute la fréquence, et éventuellement traite les paquets reçus. Par contre, CE à l'état bas met le circuit en mode veille. Il est alors déconnecté du réseau radio. En mode émetteur, la pin CE doit-être maintenue à l'état bas. Pour utiliser l'émetteur, il faut commencer par charger le buffer TX_FIFO avec les données à envoyer, puis, la pin CE est mise à 1 pendant une durée d'au moins 10µs, puis il faut la remettre à zéro.

La troisième pin est CSN, qui est le « chip select ». Il s'agit de la broche de validation pour le bus SPI, et elle est active bas (d'où le «non» dans le nom). Cette broche est maintenue à l'état haut en permanence. C'est au moment de communiquer avec le circuit que votre microcontrôleur fera passer cette pin à l'état bas afin d'activer le bus SPI.

La quatrième broche est SCK, qui est l'horloge série du bus SPI. Normalement, cette broche est maintenue à l'état bas (seul les fronts montants sont actifs), l'échantillonnage des données se faisant au milieu d'une impulsion d'horloge. Avec le matériel que j'utilise, cette configuration est représentée par les valeurs des bits CPOL et ACSP dans la configuration SPI, les deux bits étant mis à zéro.

La cinquième broche est MOSI. MOSI signifie «Master Out Slave In», le microcontrôleur étant le maître, et le 24L01 étant l'esclave. Les données partent donc du microcontrôleur et arrivent au 24L01. Cette pin est en fait contrôlée par votre microcontrôleur. Physiquement, elle est reliée à la pin MOSI du bus SPI de votre microcontrôleur.

La sixième broche est la broche MISO. Comme vous pouvez l'imaginer, MISO signifie « Master In Slave Out », et c'est donc l'inverse de la pin MOSI. Cette pin sert au 20L01 à envoyer des données au microcontrôleur. Vous pouvez donc constater que le bus SPI est un bus « full duplex » qui permet donc d'envoyer et de recevoir des données simultanément.

Les broches du SPI étant toutes décrites, nous pouvons passer à la septième broche, la broche IRQ. C'est cette pin qui sera utilisée pour que le 24L01 signale à votre microcontrôleur que quelque chose d'important vient d'arriver. Vous pouvez définir des interruptions pour toute combinaison des événements suivants: données reçues, données transmises, et le nombre maximal de tentatives d'émission atteint (celui-ci est notamment lié à la configuration matérielle du 24L01 que nous verrons plus tard). Si vous avez décider de travailler en mode « polling » plutôt qu'en mode « intéruptions », l'utilisation de cette pin ne sera pas nécessaire, mais vous serez obligés de scruter le registre STATUS du 24L01 en permanence. La pin IRQ est toujours à l'état haut, et c'est lors d'une interruption qu'elle sera mise à l'état bas.

La dernière pin est GND. Elle doit-être raccordée à la masse de votre microcontrôleur.

Interfaçage du RF24L01 via le bus SPI.

Maintenant que la description matérielle des broches du bus SPI est terminée, nous pouvons étudier l'utilisation du bus SPI sur le 24L01.

L'interface SPI vous permet de lire ou écrire les registres, de transmettre des données, de recevoir des données, et de procéder à la configuration interne du 24L01.

Le prédécesseur du 24L01 le 2401, avait une interface qui était assez lourde. Heureusement pour nous, Nordic à amélioré l'interface en implémentant un bus SPI hardware qui permet de communiquer simplement avec à peu près n'importe quel microcontrôleur.

Pour ce qui est du taux de transfert, le datasheet est un peu contradictoire. La page 1 indique un taux de transfert maximum de 8Mbps, alors qu'à la page 19, le chiffre de 10Mbps est indiqué. Je ne sais pas vous confirmer ces chiffres, n'ayant jamais travaillé qu'à des taux ne dépassant pas 2Mbps, n'ayant jamais eu besoin de vitesses supérieures.

De manière générale, la seule raison qui pourrait vous pousser à augmenter la vitesse, serait l'utilisation d 'énorme quantité de données lors de la transmission.

Pour ce qu j'en sais, la plupart des microcontrôleurs, ont des registres de configuration très similaires à celui que j'utilise. A savoir les registres CPOL et CPHA qui sont mis à zéro dans le cadre de l'utilisation du 24L01 avec mon microcontrôleur.

La broche CSN du 24L01 doit-être raccordée à n'importe quelle broche GPIO du micro-contrôleur, qui sera configuré en maître du bus SPI avec un transfert de données sur 8 bits.

Cn – SPI Instruction Bit
Sn – Status Register Bit
Dn – Data Bit (note: LSByte to MSByte, MSBit in each byte first)

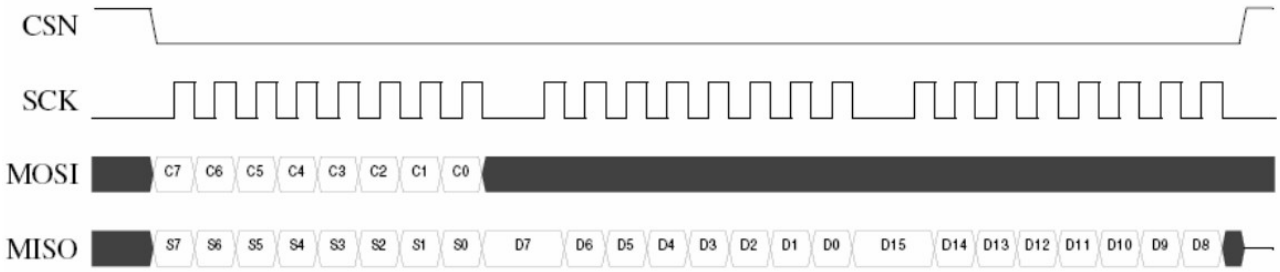

Figure 8 SPI read operation.

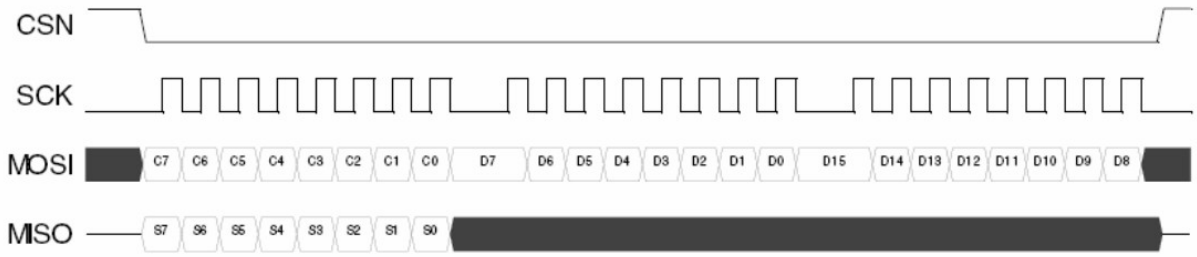

Figure 9 SPI write operation.

Figure 2. Timings SPI de lecture (en haut) et d'écriture (en bas) des opérations SPI (voir figures 8 et 9 du datasheet).

Ces exigences temporelles ne s'appliqueront pour ainsi dire jamais, sauf si vous avez la chance de posséder un microcontrôleur de la mort-qui-tue. De manière générale, vous serez toujours en dessous de ces exigences. En effet, les 50ns minimales sont en général inférieure au temps d'exécution d'une instruction de la plupart des microcontrôleurs.

Si vous n'êtes pas dans les conditions matérielles décrites dans les tutoriels suivants (en gros, si vous utilisez un autre microcontrôleur que le LPC-SP11), vous devrez écrire vous-même les routines nécessaires à l'utilisation du SPI avec le 24L01.

Avec la plupart des microcontrôleurs, il est possible de connecter MISO avec MOSI pour effectuer un re-bouclage et vérifier ainsi le fonctionnement du SPI. Ceci vous permet de vérifier l'efficacité de vos routines. Veillez toutefois, à ce que rien d'autre ne soit connecté sur votre bus SPI. Si vous êtes dans ces conditions, la mise au point des routines nécessaires à l'utilisation du 24L01 en sera grandement facilitée.

Jeu d'instruction résumé du SPI

Une fois la partie matérielle du SPI raccordée, la communication avec le 24L01 est très simple. Je vais vous montrer comment le faire avec le jeu d'instructions SPI.

Ces informations peuvent être trouvées dans le tableau 8, p. 19 du datasheet.

Instruction Name	Instruction Format [binary]	# Data Bytes	Operation
R_REGISTER	000A AAAA	1 to 5 LSByte first	Read registers. AAAAA = 5 bit Memory Map Address
W_REGISTER	001A AAAA	1 to 5 LSByte first	Write registers. AAAAA = 5 bit Memory Map Address *Executable in power down or standby modes only.*
R_RX_PAYLOAD	0110 0001	1 to 32 LSByte first	Read RX-payload: 1 – 32 bytes. A read operation will always start at byte 0. Payload will be deleted from FIFO after it is read. Used in RX mode.
W_TX_PAYLOAD	1010 0000	1 to 32 LSByte first	Used in TX mode. Write TX-payload: 1 – 32 bytes. A write operation will always start at byte 0.
FLUSH_TX	1110 0001	0	Flush TX FIFO, used in TX mode
FLUSH_RX	1110 0010	0	Flush RX FIFO, used in RX mode Should not be executed during transmission of acknowledge, i.e. acknowledge package will not be completed.
REUSE_TX_PL	1110 0011	0	Used for a PTX device Reuse last sent payload. Packets will be repeatedly resent as long as CE is high. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX payload reuse must not be activated or deactivated during package transmission
NOP	1111 1111	0	No Operation. Might be used to read the STATUS register

Figure 3. Jeu d'instructions SPI. (Tableau 8 du datasheet du nRF24L01).

Pour pouvoir transmettre ou recevoir des données avec le 24L01 au travers du bus SPI, il y a plusieurs choses à faire.

Tout d'abord, la pin CSN doit se trouver à l'état haut. Une mise à zéro de la pin CSN indiquera au 24L01 qu'une communication sera établie sur le bus SPI. (Notez que cette pin devra rester à l'état bas pendant toute la durée de la transaction).

Ensuite, il faut envoyer l'octet de commande qui définira la commande à exécuter.
Si vous devez recevoir des octets du 24L01, il vous faudra envoyer un octet pour chacun des octets que vous désirez recevoir du 24L01.

Par contre, si vous désirer envoyer des données au 24L01, il vous suffit de les envoyer par le bus SPI, et en général, vous ne devrez pas tenir compte des octets que le 24L01 pourrait vous envoyer. Lors de la réception de données, les octets suivant la commande que vous avez envoyée, ne sont pas pris en compte par le 24L01. En fait, le 24L01 ne tient plus compte des octets qui suivent le ou les octets de commande.

Une fois votre transaction terminée, la pin CSN doit retourner à l'état haut, et à partir de ce moment, il vous sera possible de traiter les données dans votre microcontrôleur.

A titre d'exemple, disons que vous allez exécuter l'instruction R_REGISTER sur le registre TX_ADDR, qui va lire le contenu du registre d'adresse TX du 24L01.
Le registre TX_ADDR a une taille de 5 octets, et nous supposons que vous utilisez des adresses de 5 octets.
La première chose à faire est de porter la pin CSN à l'état bas et ensuite d'envoyer l'octet de commande '00010000' au 24L01.

Cela indique au 24L01 que vous souhaitez lire le registre 0x10, qui est le registre TX_ADDR.
Ensuite, vous envoyez cinq octets de données fictives, (le contenu de ces octets n'ayant aucune importance), et le 24L01 renverra le contenu du registre TX_ADDR sur cinq octets.
Enfin, il vous faut remettre la pin CSN à l'état haut.
Donc, au total, vous aurez reçu six octets.
A chaque fois que vous envoyez une commande, le 24L01 vous répond en vous envoyant le registre STATUS. Ensuite, vous avez reçu les cinq octets qui sont contenus dans le registre TX_ADDR.
Note: Je vais décrire les registres dans la section suivante, alors ne vous tracassez pas si vous ne connaissez pas encore leur fonctionnalité.

Maintenant que vous connaissez le format d'envoi des instructions au 24L01, je vais décrire le jeu d'instructions.

Premièrement, nous allons décrire l'instruction R_REGISTER.

Cette instruction vous permet de lire l'un des registres dans le 24L01.
Il a le format binaire de '000AAAAA', où 'AAAAA' est l'adresse du registre que vous voulez lire.
Cette adresse doit être comprise entre 0x00 et 0x17, ou «00000» à «10111» en binaire (voir le tableau 11 de la fiche technique). Pour vous faciliter la tâche, ma bibliothèque reprend aussi bien les valeur hexa que binaire pour la définition des registres et du jeu d'instructions. Comme çà, vous ne devez pas vous tracasser pour les valeurs hexa. Donc, pour l'instruction R_REGISTER , vous allez envoyer de un à cinq octets de données, en fonction du registre que vous voulez lire. (Les registres d'adresses peuvent comporter jusqu'à cinq octets, mais tous les autres registres ont une taille de un octet).

Vient ensuite l'instruction W_REGISTER, qui vous permet d'écrire la plupart des registres du 24L01. Il est très similaire au format de l'instruction R_REGISTER, avec un format binaire de '001AAAAA', où 'AAAAA' est l'adresse du registre que vous voulez écrire.
Une fois que vous envoyez l'octet d'instruction, vous allez de nouveau envoyer de un à cinq octets de données, en fonction du registre à écrire, avec les mêmes règles pour les registres que dans l'instruction R_REGISTER.

Maintenant, nous voici à la première des instructions les plus importante : R_RX_PAYLOAD.

Cette opération vous permet de lire le contenu du buffer RX FIFO après la réception d'un paquet (lorsque vous êtes en mode RX). En général, la réception d'un paquet est signalée par l'interruption RX_DR. L'utilisation de cette opération est un peu plus compliqué que les autres, car il vous oblige à utiliser aussi la pin CE.
Pendant la réception d'un paquet, la pin CE doit-être à l'état haut. Une fois le paquet reçu, vous devez commencer par désactiver le récepteur en faisant passer la pin CE à l'état bas. Ceci fait, vous pouvez alors exécuter l'opération R_RX_PAYLOAD. Ensuite, vous envoyez l'octet de commande (0x61), puis un nombre d'octets factices qui doit être égal au nombre d'octets que vous désirez recevoir. Attention que chaque octet sortit du buffer FIFO est effacé dans celui-ci. S'il y a encore des données dans le FIFO de réception (le dispositif peut contenir 3 « charges utiles » dans le RX FIFO - voir la section "FIFO Info" pour plus de détails), vous pouvez alors lire le buffer jusqu'à ce que toutes les données soient lues. Ceci étant fait, vous pouvez effacer l'interruption RX_DR et ramener la pin CE à l'état haut pour réactiver le récepteur. Il ne vous reste plus qu'à traiter les données reçues.

L'autre instruction importante est W_TX_PAYLOAD.

C'est l'instruction que vous utilisez lorsque le 24L01 est en mode émission (TX), et que vous désirez envoyer un paquet.
En mode normal d'émission, la broche CE est maintenue à l'état bas.
Vous envoyez d'abord l'octet de commande (0xA0), puis le paquet à envoyer (« la charge utile »).
Le nombre d'octet que vous allez envoyer doit correspondre au nombre d'octet que le récepteur s'attend à recevoir. Une fois le paquet chargé dans le 24L01, La broche CE doit-être mise à l'état haut pendant au moins 10µs. Une fois le paquet envoyé, l'interruption TX_DS sera générée. Cette interruption vous indique que le paquet à été envoyé avec succès . Si vous avez activé la fonction « auto-ack » interruption TX_DS ne sera générée que si le paquet à été effectivement envoyé. Si le maximum d'essais d'émission est atteint, interruption MAX_RT sera activée. Il vous faudra alors effacer les interruptions avant de pouvoir continuer en tenant compte des erreurs signalées. Rappelez-vous aussi que, comme le FIFO RX, le FIFO TX possède trois niveaux de profondeur.
Cela signifie que vous pouvez charger jusqu'à trois paquets dans le FIFO TX du 24L01 avant de commuter la pin CE pour enclencher l'émission.

Les commandes FLUSH_TX et FLUSH_RX (opcodes 0xE1 et 0xE2) sont utilisés pour effacer toutes les données dans les FIFOs TX et RX, respectivement.
Cela peut être utile pour effacer un paquet que vous savez être faux. Ces opérations n'ont pas d'octets de données.

Une opération pour laquelle je n'ai jamais trouvé un usage réel est REUSE_TX_PL.
Cela vous permet d'envoyer un paquet aussi longtemps que la pin CE est maintenue à l'état haut sur un dispositif TX (le statut de cette opération se retrouve dans le bit TX_REUSE du registre FIFO_STATUS). L' opcode de cette instruction est 0xE3 et il n'a pas d'octets de données.
Si vous décidez d'exécuter cette fonction, vous pouvez l'arrêter temporairement en portant la pin CE à l'état bas. Pour l'arrêter définitivement, vous pouvez envoyer la commande W_TX_PAYLOAD ou la commande FLUSH_TX. Vous devez être sûr de ne pas exécuter cette instruction pendant que le 24L01 envoie un paquet, parce que cela pourrait provoquer des choses étranges...

La dernière instruction est NOP.
Son opcode est 0xFF et n'a aucun octets de données.
La seule utilisation réelle de cette opération est de lire le registre d'état rapidement, mais cela n'est pas très utile, surtout si vous n'utilisez pas la broche IRQ.

Un aperçu rapide des registres internes du nRF24L01

Ci-dessous vous trouverez une description de chaque registre, ainsi que les informations de la fiche technique pour le registre en dessous de la description.

Je vais souligner quelques points importants à chaque pièges potentiels que j'ai détecté.
Oh, et rappelez-vous aussi que les adresses des registres sont en hexa, pas en décimal.

Le premier registre (enfin, techniquement c'est le registre 0 mais bon) est le registre CONFIG.

Il est le registre le plus important dans la mesure où c'est par ce registre que toutes les fonctions du 24L01 seront commandées.
Ce registre contient le bit de PWR_UP, qui doit être activé pour que le 24L01 puisse effectuer n'importe quelle opération. Par conséquent, si vous voulez communiquer avec votre24L01, vous devez écrire dans ce registre au moins une fois.
Un autre bit important est PRIM_RX, qui, lorsqu'il est mis à 1, définit le circuit en récepteur et lorsqu'il est à zéro en émetteur. C'est aussi dans ce registre que vous configurerez vos interruptions, ainsi que la configuration de votre CRC. Personnellement, je recommande un CRC à 2 octets, mais cela dépend beaucoup du flux de données dont vous avez besoin.
La description de ce registre peuvent être consultés sur la figure 4.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
00	CONFIG				Configuration Register
	Reserved	7	0	R/W	Only '0' allowed
	MASK_RX_DR	6	0	R/W	Mask interrupt caused by RX_DR 1: Interrupt not reflected on the IRQ pin 0: Reflect RX_DR as active low interrupt on the IRQ pin
	MASK_TX_DS	5	0	R/W	Mask interrupt caused by TX_DS 1: Interrupt not reflected on the IRQ pin 0: Reflect TX_DS as active low interrupt on the IRQ pin
	MASK_MAX_RT	4	0	R/W	Mask interrupt caused by MAX_RT 1: Interrupt not reflected on the IRQ pin 0: Reflect MAX_RT as active low interrupt on the IRQ pin
	EN_CRC	3	1	R/W	Enable CRC. Forced high if one of the bits in the EN_AA is high
	CRCO	2	0	R/W	CRC encoding scheme '0' - 1 byte '1' – 2 bytes
	PWR_UP	1	0	R/W	1: POWER UP, 0:POWER DOWN
	PRIM_RX	0	0	R/W	1: PRX, 0: PTX

Figure 4. CONFIG

Le registre suivant est EN_AA, ou le registre d'activation de l' »auto-ack ».
Ce registre permet d'activer ou de désactiver l'auto-accusé de réception sur une base « per-pipe ».
Si vous êtes familier avec TCP, vous savez ce que sont les accusés de réception.
Essentiellement, la transmission 24L01 va envoyer un paquet, puis passer momentanément en réception.
Si le 24L01 récepteur reçoit le paquet, il va renvoyer un accusé de réception.
Si le 24L01 émetteur reçoit l'accusé de réception, alors il est content.

Si le 24L01 émetteur ne reçoit pas l'accusé de réception dans le timing spécifié, il envoie le paquet à nouveau et attend un ack.
Il fera cela un certain nombre de fois. Ce nombre est définit dans le registre de SETUP_RETR pour le buffer utilisé, et si aucun accusé de réception est reçu, il activera l'interruption de MAX_RT.
Ceci est une caractéristique extrêmement utile et je recommande de l'utiliser à chaque fois que c'est possible.
Une mise en garde toutefois. L'adresse de réception du tuyau 0 doit-être la même que l'adresse d'émission lorsque l'auto-ack est activé sur le tuyau que vous utilisez (voir p.14 du datasheet).
La description du registre peuvent être consultés à la figure 5.

01	EN_AA Enhanced ShockBurst™				Enable 'Auto Acknowledgment' Function Disable this functionality to be compatible with nRF2401, see page 26
	Reserved	7:6	00	R/W	Only '00' allowed
	ENAA_P5	5	1	R/W	Enable auto ack. data pipe 5
	ENAA_P4	4	1	R/W	Enable auto ack. data pipe 4
	ENAA_P3	3	1	R/W	Enable auto ack. data pipe 3
	ENAA_P2	2	1	R/W	Enable auto ack. data pipe 2
	ENAA_P1	1	1	R/W	Enable auto ack. data pipe 1
	ENAA_P0	0	1	R/W	Enable auto ack. data pipe 0

Figure 5. EN_AA

02	EN_RXADDR				Enabled RX Addresses
	Reserved	7:6	00	R/W	Only '00' allowed
	ERX _P5	5	0	R/W	Enable data pipe 5.
	ERX _P4	4	0	R/W	Enable data pipe 4.
	ERX _P3	3	0	R/W	Enable data pipe 3.
	ERX _P2	2	0	R/W	Enable data pipe 2.
	ERX _P1	1	1	R/W	Enable data pipe 1.
	ERX _P0	0	1	R/W	Enable data pipe 0.

Figure 6. EN_RXADDR

Le registre suivant SETUP_AW. C'est ici que vous définissez la largeur de votre adresse (il doit correspondre à la fois l'émetteur et le récepteur).
Je recommande personnellement 5 octets d'adresse, sauf si vous avez absolument besoin de la bande passante, car plus il y a d'octets d'adresse (et d'octets CRC), moins vous avez de bande passante pour les autres octets. Cela est particulièrement intéressant si votre 24L01 se trouve à proximité d'autres appareils qui pourraient fonctionner dans la gamme de 2,4 GHz (les routeurs sans fil, etc.).

03	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' – 5 bytes LSByte will be used if address width below 5 bytes

Figure 7. SETUP_AW

Le registre SETUP_RETR est le registre suivant.

Ce registre permet de définir le nombre de tentatives que vous allez utiliser (commun à tous les tuyaux) si vous avez le bit EN_AA activé sur un tuyau.

Ce registre est composé de deux parties. La première « ARC », bits 0 à 3, permet de définir le nombre de tentatives en émission (sa valeur par défaut est 3). A vous de le redéfinir en fonction de votre application. La deuxième partie de ce registre « ARD », bits 4 à 7, vous permet de définir le délais entre chaque retransmission. Personnellement, je laisse la valeur par défaut, mais vous pouvez expérimenter d'autres valeurs.

04	SETUP_RETR				Setup of Automatic Retransmission
	ARD	7:4	0000	R/W	Auto Re-transmit Delay '0000' – Wait 250+86uS '0001' – Wait 500+86uS '0010' – Wait 750+86uS …….. '1111' – Wait 4000+86uS (Delay defined from end of transmission to start of next transmission)[14]
	ARC	3:0	0011	R/W	Auto Retransmit Count '0000' –Re-Transmit disabled '0001' – Up to 1 Re-Transmit on fail of AA …… '1111' – Up to 15 Re-Transmit on fail of AA

Figure 8. SETUP_RETR

Le registre 0x05 est le registre de RF_CH. Il vous permet de définir le canal RF (je suis sûr que vous l'aviez compris sans que je vous le dise). Ce registre vous permet de définir le canal entre 0 et 127, et chaque canal est séparé de 1Mhz du suivant. Attention toutefois à respecter les règlements en cours dans votre pays en ce qui concerne la bande ISM des 2G4. La FCC aux États-Unis ne vous permet d'utiliser les fréquences de 2,4000 à 2,4835 GHz dans la bande ISM, qui vous donne 835 Mhz, soit les 84 premiers canaux à utiliser (de 0 à 83). Tout canal supérieur à 83 est hors des limites légales. Afin d'éviter les problèmes potentiels avec l'Oncle Sam et la FCC, le mieux est de rester loin de cette plage de fréquences.

(Note du traducteur : En Belgique, la bande autorisée est identique).

05	RF_CH				RF Channel
	Reserved	7	0	R/W	Only '0' allowed
	RF_CH	6:0	0000010	R/W	Sets the frequency channel nRF24L01 operates on

Figure 9. RF_CH

Le registre suivant est le registre de RF_SETUP.

Il contient tous les paramètres nécessaires (en plus du numéro de canal) pour mettre en place la section RF du 24L01. Les bits PLL_LOCK et LNA_HCURR doivent rester à leurs valeurs par défaut, sinon des choses bizarres pourraient arriver. Le champ RF_DR vous permet de changer votre débit de données, et je recommande la valeur de 2 Mbps tant que la qualité de la transmission est bonne. Le choix de 1 Mbps vous permettra de fiabiliser la transmission avec une portée plus longue tout en gardant un taux d 'erreurs acceptable. Le prix à payer est alors la division de la bande passante par deux.Vous pouvez également modifier le niveau de puissance RF en utilisant le champ RF_PWR.

Je recommande le laisser à sa valeur la plus élevée « 11 » en binaire, qui est également la

valeur par défaut (pour rappel (0dBm=1,25mW, -6dBm=0,25mW, -12dBm=0,06mW et -18dBm=0,02mW). Vous pouvez réduire la puissance si vous avez besoin d'économiser l'énergie et/ou si vous avez des modules proches les un des autres.

06	RF_SETUP				RF Setup Register
	Reserved	7:5	000	R/W	Only '000' allowed
	PLL_LOCK	4	0	R/W	Force PLL lock signal. Only used in test
	RF_DR	3	1	R/W	Data Rate '0' – 1 Mbps '1' – 2 Mbps
	RF_PWR	2:1	11	R/W	Set RF output power in TX mode '00' – -18 dBm '01' – -12 dBm '10' – -6 dBm '11' – 0 dBm
	LNA_HCURR	0	1	R/W	Setup LNA gain

Figure10. RF_SETUP

Et voici votre nouvel ami. J'ai nommé le registre STATUS.

Avec les FIFOs TX/RX, ce sera l'élément que vous utiliserez le plus. (Nous parlerons de ces FIFOS plus tard). Lorsque la broche IRQ est activée (passe au niveau bas), c'est le registre à vérifier pour voir ce qui est arrivé. Il vous indique les interruptions qui sont actuellement activée, pour un récepteur : si le buffer d'entrée contient des données (champ RX_P_NO), et pour un émetteur : si le buffer de sortie est rempli (champ TX_FULL). Même lorsque vous n'utilisez pas les interruptions , c'est quand-même le registre à consulter pour se faire une idée sur ce qui se passe dans le circuit. Lorsqu'une interruption est déclenchée, il faut écrire un 1 dans le bit correspondant pour l'effacer.

07	STATUS				Status Register (In parallel to the SPI instruction word applied on the MOSI pin, the STATUS register is shifted serially out on the MISO pin)
	Reserved	7	0	R/W	Only '0' allowed
	RX_DR	6	0	R/W	Data Ready RX FIFO interrupt. Set high when new data arrives RX FIFO[15]. Write 1 to clear bit.
	TX_DS	5	0	R/W	Data Sent TX FIFO interrupt. Set high when packet sent on TX. If AUTO_ACK is activated, this bit will be set high only when ACK is received. Write 1 to clear bit.
	MAX_RT	4	0	R/W	Maximum number of TX retries interrupt Write 1 to clear bit. If MAX_RT is set it must be cleared to enable further communication.
	RX_P_NO	3:1	111	R	Data pipe number for the payload available for reading from RX_FIFO 000-101: Data Pipe Number 110: Not Used 111: RX FIFO Empty
	TX_FULL	0	0	R	TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.

Figure 11. STATUS

Le registre de OBSERVE_TX est un peu compliqué à décrire.

Le champ PLOS_CNT décrit combien de paquets ont été perdus depuis la dernière réinitialisation. Cela signifie que pour tous les canaux qui ont, ou ont eu, l'auto-ack activé, si un paquet a été retransmis le nombre maximal de fois sans succès, ce champ est incrémenté. Notez toutefois que, faisant 4 bits de long, ce nombre est limité à 15, et reste à 15 si ce maximum est atteint . Pour effacer ce compteur, il faut réaliser une écriture dans RF_CH. L'idée étant que si la liaison est de mauvaise qualité, il est souhaitable de changer de canal.

L'autre partie de ce registre est ARC_CNT, ce qui compte le nombre de fois qu'un packet à été ré-envoyé. Ce compteur se réinitialise lorsqu'un nouveau paquet est transmis.

08	OBSERVE_TX				Transmit observe register
	PLOS_CNT	7:4	0	R	Count lost packets. The counter is overflow protected to 15, and discontinue at max until reset. The counter is reset by writing to RF_CH. See page 14 and 17.
	ARC_CNT	3:0	0	R	Count resent packets. The counter is reset when transmission of a new packet starts. See page 14.

Figure 12. OBSERVE_TX.

Le registre 0x09 est le registre CD.

Ce registre surveille essentiellement la fréquence que vous avez sélectionné pour voir s'il y a du trafic sur ce canal.

Sa valeur est 0 si rien est détecté et 1 s'il y a quelque chose. Je dois vous avouer que je ne l'ai jamais utilisé, mais il serait sans doute utile pour les situations où vous avez plusieurs 24L01s qui tentent de transmettre en même temps (maillage de réseaux qui ne sont pas structurés maître/esclave par exemple). Une chose à noter avec cette fonction, et qui est d'ailleurs indiqué dans la fiche technique, c'est que le 24L01 doit voir un signal sur cette fréquence pendant au moins 128 µS avant que le registre CD ne soit activé.

09	CD				
	Reserved	7:1	000000	R	
	CD	0	0	R	Carrier Detect. See page 17.

Figure 13. CD

Les registres 0x0A à 0x0F sont les adresses de chacun des buffers de réception.

Il n'y a pas vraiment grand chose à en dire de plus, la doc étant suffisamment explicite.

0A	RX_ADDR_P0	39:0	0xE7E7E7E7E7	R/W	Receive address data pipe 0. 5 Bytes maximum length. (LSByte is written first. Write the number of bytes defined by SETUP_AW)
0B	RX_ADDR_P1	39:0	0xC2C2C2C2C2	R/W	Receive address data pipe 1. 5 Bytes maximum length. (LSByte is written first. Write the number of bytes defined by SETUP_AW)
0C	RX_ADDR_P2	7:0	0xC3	R/W	Receive address data pipe 2. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
0D	RX_ADDR_P3	7:0	0xC4	R/W	Receive address data pipe 3. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
0E	RX_ADDR_P4	7:0	0xC5	R/W	Receive address data pipe 4. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
0F	RX_ADDR_P5	7:0	0xC6	R/W	Receive address data pipe 5. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]

Figure 14. RX_ADDR

Le registre suivant est TX_ADDR. Comme les registres de RX_ADDR, il appelle peu de commentaires.

10	TX_ADDR	39:0	0xE7E7E7E7E7	R/W	Transmit address. Used for a PTX device only. (LSByte is written first) Set RX_ADDR_P0 equal to this address to handle automatic acknowledge if this is a PTX device with Enhanced ShockBurst™ enabled. See page 14.

Figure 15. TX_ADDR

Les registres 0x11 à 0x16 donnent la taille de chaque buffer de réception.

N'oubliez pas que pour pouvoir utiliser un buffer, il faut que sa taille soit déclarée non nulle. Comme vous pouvez le voir dans la fiche technique, toutes les valeurs par défaut sont à 0. Par conséquent, la liaison radio ne peut fonctionner que si le registre correspondant au buffer utilisé à été chargé avec une valeur non nulle,(en supposant également que le buffer à été activé dans le registre de EN_RXADDR). La description du registre pour le buffer 0 se trouve en fig. 16 (les autres buffers étant identiques).

11	RX_PW_P0				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P0	5:0	0	R/W	Number of bytes in RX payload in data pipe 0 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes

Figure registre 16. RX_PW_P0

Le dernier registre est FIFO_STATUS (tous les bits non réservés sont en lecture seule).

Ce registre sert à indiquer si les FIFOs TX et RX sont pleins ou vides. Ce qui peut-être très utile lorsque vous envoyez des données à des taux élevés et que vous voulez maximiser l'utilisation des FIFOs. Le bit TX_REUSE, comme mentionné ci-dessus, indique que l'instruction REUSE_TX_PL a été exécutée.

17	FIFO_STATUS				FIFO Status Register
	Reserved	7	0	R/W	Only '0' allowed
	TX_REUSE	6	0	R	Reuse last sent data packet if set high. The packet will be repeatedly resent as long as CE is high. TX_REUSE is set by the SPI instruction REUSE_TX_PL, and is reset by the SPI instructions W_TX_PAYLOAD or FLUSH TX
	TX_FULL	5	0	R	TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.
	TX_EMPTY	4	1	R	TX FIFO empty flag. 1: TX FIFO empty. 0: Data in TX FIFO.
	Reserved	3:2	00	R/W	Only '00' allowed
	RX_FULL	1	0	R	RX FIFO full flag. 1: RX FIFO full. 0: Available locations in RX FIFO.
	RX_EMPTY	0	1	R	RX FIFO empty flag. 1: RX FIFO empty. 0: Data in RX FIFO.

Figure 17. FIFO_STATUS registre

Infos sur les FIFOs

Il y a des FIFOs pour les deux modes TX et RX (comme mentionné dans la section «Jeu d'instruction résumé du SPI"). Si vous ne savez pas ce qu'est un FIFO, cela signifie FirstInFirstOut «premier entré, premier sorti", un peu comme la file chez l'épicier du coin. Les deux FIFOs ont trois niveaux de profondeur, ce qui signifie qu'ils sont capables de contenir trois paquets. Dans le principe, cela veut dire que si vous recevez trois paquets en réception, et que vous ne les lisez pas, le paquet suivant qui sera reçu, « poussera » le paquet le plus ancien, et celui-ci sera perdu. C'est d'ailleurs le même principe pour l'émission. Si vous chargez vos buffers avec trois paquets, et que vous ne les envoyez pas (en activant la pin CE), le paquet suivant que vous chargerez dans le buffer, « poussera » le premier paquet que vous y aviez mis.

Format de données des paquets.

Vous vous demandez peut-être pourquoi se soucier de ce qui se passe sur la fréquence entre les 24L01s. Pour ceux qui ont des faibles débits de données, cela n'est pas très important. Par contre, si vous essayez d'obtenir un certain débit de données, il est très important de le savoir parce que la quantité d'octets généraux qui sont dans le paquet va grandement affecter votre débit de données éffectif.
Pour voir une description graphique du format des paquets pour les modes « ShockBurst » et « Enhanced ShockBurst », voir la page 27 du datasheet, ou les figures 18 et 19 suivantes.
Dans les deux modes, le préambule est transmis en premier lieu, et il est d'un octet de longueur.
Il est composé d'une alternance de zéro et de un, et est utilisé pour permettre au récepteur de sortir un signal du bruit, et de se synchroniser.
Ensuite, dans les deux modes, les octets suivants sont les octets d'adresse. Cette adresse est définie

par l'utilisateur, et sa taille est comprise entre trois et cinq octets.

Figure 18. Paquet de donnée en mode « Shockburst »

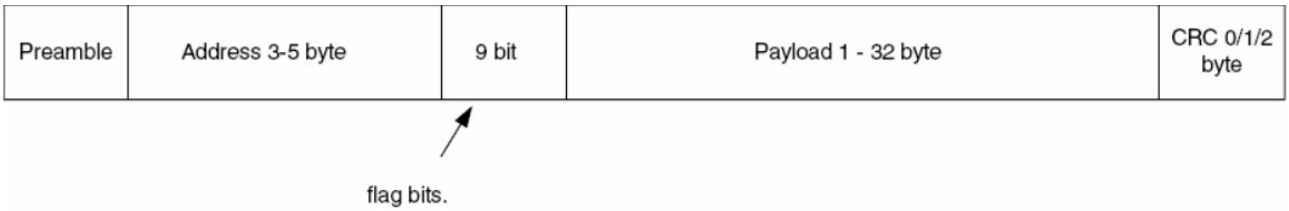

Figure 19. Paquet de donnée en mode « Enhanced Shockburst »

Les éléments suivants qui seront envoyés dépendent du mode utilisé. En mode « Shockburst », il s'agira des octets de données (maximum 32), alors qu'en mode « Enhanced Shockburst », avant d'envoyer les octets de données, un ensemble de 9 bits est envoyé. Cet ensemble est en fait un message de statut concernant les retransmissions. Actuellement, seuls les deux premiers bits sont utilisés (comptage des paquets ré-envoyés), les 7 bits restant étant réservés à une utilisation future. Dans les deux modes, l'envoi du paquet se termine par les CRCs. L'utilisateur peut définir le nombre de CRCs entre 0 et 2.

Fiabilité de la transmission et calcul de la bande passante.

Maintenant que vous connaissez le format du paquet de données, nous pouvons faire quelques calculs sur la bande passante réellement utilisée sur votre liaison HF. Certaines personnes vont vouloir augmenter le plus possible la bande passante malgré l'augmentation du nombre d'erreurs durant la transmission (un peu comme les radios internet). Par contre, d'autres utilisateurs préférerons fiabiliser au maximum la transmission, et cela se fera au détriment de la bande passante. Ce choix se fera évidement en fonction de votre cahier des charges.
Tenant compte des ces remarques, voici un exemple de calcul de bande passante, qui montre le débit maximum qu'il est possible d'obtenir avec un 24L01.
Supposons que vous pouvez constamment envoyer des paquets (pas d'attente entre les paquets), 100% des paquets atteignent le récepteur, le débit de données est de 2 Mbps, et le mode ShockBurst est utilisé. Avec cette configuration, il est possible d'avoir entre quatre et huit octets en plus des données (un octet de préambule, trois à cinq octets d'adresse, et zéro à deux octets de CRC). En obtimisant au maximum, vous pouvez envoyer 32 octets de données avec 4 octets de supplément. Cela donne 88,9% d'utilisation, ou 1,78 Mbps de débit de données (32 octets de données pour un paquet de 36 octets au total). Vous pouvez utiliser ce type de calcul pour toute hypothèse quand au différents formats utilisés, mais ayez toujours en tête qu'il s'agit d'un débit maximum théorique. Rappelez-vous aussi que le calcul précédent est très idéal et que dans le monde réel, vous ne recevrez pas 100% des paquets et qu'il y aura un certain retard entre les paquets.