<http://www.picaxeforum.co.uk/showthread.php?16773-nrf24L01-and-Picaxe-20x2-how-to-read-Registers>

# Thread: [nrf24L01+ and Picaxe 20x2 - how to read Registers](http://www.picaxeforum.co.uk/showthread.php?16773-nrf24L01-and-Picaxe-20x2-how-to-read-Registers&s=4aa447738a8a44ce08a6daef843b5331)

Hi,

I want to interface the nrf24L01+ ([http://www.nordicsemi.com/index.cfm?...display&pro=94](http://www.nordicsemi.com/index.cfm?obj=product&act=display&pro=94)) with a Picaxe 20x2 and I’ve started reading this tutorial made by Brennen [http://www.diyembedded.com/tutorials...tutorial\_0.pdf](http://www.diyembedded.com/tutorials/nrf24l01_0/nrf24l01_tutorial_0.pdf)

This tutorial is C language oriented and I want to build my own functions (in programming Editor) for Picaxe 20x2

At the end of page 6 of this Tutorial Brennen wrote:

“First, you would bring CSN low and then send the command byte ‘00010000’ to the 24L01. This instructs the 24L01 that you want to read register 0x10, which is the TX\_ADDR register. Then you would send five dummy data bytes (it makes absolutely no difference what the data bytes contain), and the 24L01 will send back to you the contents of the TX\_ADDR register” Finally, you would bring the CSN pin back high. All totaled, you will receive six bytes. When you send any command byte, the 24L01 always returns to you the STATUS register.

My problem is to understand when do I need to use “hspiin” and “hspiout” to obtain the response to a “instruction”...

e.g - To read “TX\_ADDR”... is this statement correct? am I using hspiout and hspiin correctly? 

Je veux interfacer le nRF24L01 + (http://www.nordicsemi.com/index.cfm?...display&pro=94) avec un 20x2 Picaxe et j'ai commencé à lire ce tutoriel réalisé par Brennen http: //www.diyembedded .com / tutoriels ... tutorial\_0.pdf

Ce tutoriel est le langage C orienté et je veux construire mes propres fonctions (dans l'Éditeur de programmation) pour Picaxe 20x2

A la fin de **la page 6 de ce Brennen Tutorial** écrit:

"Tout d'abord, vous apporteriez CSN bas et **ensuite envoyer l'octet de commande '00010000'** du 24L01. Cela indique au 24L01 que vous souhaitez **lire le registre 0x10 (0x10 =d.16 = b.00010000)**, qui est le registre TX\_ADDR. Ensuite, **vous envoyer cinq octets de données fictives** (il fait absolument aucune différence ce que les octets de données contiennent), et le 24L01 renverrez pour vous le contenu de la TX\_ADDR inscrivez «Enfin, vous apporteriez la broche CSN dossier haut. Tous ont totalisé, vous recevrez **six octets**. Lorsque vous envoyez tout octet de commande, le 24L01 revient toujours à vous le registre STATUS.

Mon problème est de comprendre quand dois-je utiliser "hspiin" et "hspiout" pour obtenir la réponse à une "instruction" ...

e.g - Pour lire "TX\_ADDR" ... cette affirmation est correcte? j'utilise hspiout et hspiin correctement?

Code:

hspisetup spimode11e, spimedium

READTXADDR: {

low cs ; enable chip select

hspiout (%00010000) ; read register 0x10 = TX\_ADDR

hspiout (1,2,3,4,5) ; send five dummy data

hspiin(b1,b2,b3,b4,b5,b6) ; obtain the 5 bytes address

high cs ; disable chip select

}

If it's not asking to much... what will be the code to read the CONFIG register to "b0" variable?

(I really need to understand how to run these two commands to be able to forward next step...)

Si elle ne demande pas grand-chose ... quel sera le code pour lire le CONFIG registre à la variable "b0"?

(Je vraiment besoin de comprendre comment exécuter ces deux commandes pour être en mesure de transmettre l'étape suivante ...)

The code looks right in that it seems to match what is described; what happens when you try it, does it work as expected ?

Do you have access to a scope or a logic analyser ? If so that makes it much easier to see what's happening on the bus lines, otherwise it's a case of analysing or guessing and trying what needs to change.

I'm not sure how you mean by "what will be the code to read the CONFIG register to "b0" variable?"; this seems to read the CONFIG register, which returns six bytes.

Le code semble bon en ce qu'il semble correspondre à ce qui est décrit; ce qui se passe lorsque vous essayez, ça marche comme prévu?

Avez-vous accès à une portée ou un analyseur logique? Si oui qui le rend beaucoup plus facile de voir ce qui se passe sur les lignes de bus, sinon il est un cas d'analyse ou de deviner et d'essayer ce qui doit changer.

Je ne sais pas comment vous dire par «ce qui sera le code pour lire le CONFIG enregistrer pour" b0 "variable?"; cela semble lire le registre CONFIG, qui retourne six octets.

I've edit the post... the adress 0x10 is the TX\_ADDR.

For CONFIG the adress is 0x00 (as you can see in the page 9).

I think I will have to write something like:

J'ai éditer le post ... l'adresse 0x10 est le TX\_ADDR.

Pour CONFIG l'adresse est 0x00 (comme vous pouvez le voir dans la page 9).

Je pense que je vais devoir écrire quelque chose comme:

READ\_CONFIG:{

 low cs ; enable chip select

 hspiout (%00000000) ; send instruction register 0x00 = CONFIG

 hspiout (1) ; send dummy byte

 hspiin(b0) ; obtain the value of the CONFIG register

 high cs

}

I've always to use 2 hspiout commands and 1 hspiin right?

note: I have bought 2 nrf24L01+ but i'm still waiting for the arrive and make all this experiments.

Je me suis toujours d'utiliser 2 hspiout commandes et 1 hspiin OK?

remarque: je l'ai acheté 2 nRF24L01 + mais je suis toujours en attente pour l'arrivée et de faire toutes ces expériences.

Well, we're both trying to come to grips with the nRF24L01... 

I haven't yet tried to control it from a Picaxe so can't help you much with it. However, and from the code I've been writing for an AVR, I doubt it will work that way. SPI communcation is truly bidirectional. You're sending data at the same time it is responding.

I don't think you'll have much luck reading the status byte that is sent back while you're pushing out a command, unless you go bit banging.

As for reading a register, the nRF will answer back while you're sending the dummy bytes. You'd better be in hspiin mode by then. Try instead:

hspiout (%00010000) ; read register 0x10 = TX\_ADDR
hspiin(b2,b3,b4,b5,b6) ; obtain the 5 bytes address

Eh bien, nous sommes tous les deux essayer de venir à bout de l'nRF24L01 ...

Je ne l'ai pas encore essayé de le contrôler à partir d'un Picaxe ne peut donc pas vous aider beaucoup avec elle. Cependant, et à partir du code que j'ai écrit pour un AVR, je doute qu'il va travailler de cette façon. SPI communcation est vraiment bidirectionnelle. Vous envoyez des données en même temps, il répond.

Je ne pense pas que vous aurez beaucoup de chance de lire l'octet d'état qui est renvoyé pendant que vous poussez une commande, sauf si vous allez peu cogner.

Quant à la lecture d'un registre, le nRFO répondra dos pendant que vous envoyez les octets factices. Vous feriez mieux d'être en mode hspiin alors. Essayez plutôt:

hspiout (% 00010000); lire le registre 0x10 = TX\_ADDR

hspiin (b2, b3, b4, b5, b6); obtenir l'adresse 5 octets

A further glance through the tutorial and the use of these dummy bytes make more sense.

What the tutorial is saying is to read the five ADDR bytes, you need to clock those bytes in, and to achieve that one can send five dummy bytes. This is because the device's SPI protocol sends bytes as it is receiving bytes.

Un autre coup d'oeil à travers le tutoriel et l'utilisation de ces octets factices plus de sens.

Qu'est-ce que le tutoriel dit est de lire les cinq octets ADDR, vous devez horloge ces octets, et de réaliser que l'on peut envoyer cinq octets factices. En effet, le protocole SPI de l'appareil envoie des octets qu'il reçoit octets.

To read ADDR it's probably ...

low cs
hspiout (%00010000)
hspiin(b1,b2,b3,b4,b5,b6)
high cs

To read CONFIG it's probably ...

low cs
hspiout (%00000000)
hspiin(b0)
high cs

I'm not sure you have the right SPI Mode; it may be SPIMODE10 or SPIMODE10e.

Je ne suis pas sûr que vous avez le mode SPI correct; il peut être SPIMODE10 ou SPIMODE10e.

 Originally Posted by **Chavaquiah** 

*I don't think you'll have much luck reading the status byte that is sent back while you're pushing out a command, unless you go bit banging.*

*Je ne pense pas que vous aurez beaucoup de chance de lire l'octet d'état qui est renvoyé pendant que vous poussez une commande, sauf si vous allez peu cogner.*

It may be possible to issue the HSPIOUT to send a byte command and then PEEKSFR SSPBUF to obtain what was passed back at the same time, which will be the status bits. You'd have to test that.

Il peut être possible d'émettre HSPIOUT pour envoyer une commande d'octet puis PEEKSFR SSPBUF pour obtenir ce qui a été renvoyé en même temps, qui seront les bits d'état. Il faudrait tester cela.

 Originally Posted by **hippy** 

*It may be possible to issue the HSPIOUT to send a byte command and then PEEKSFR SSPBUF to obtain what was passed back at the same time, which will be the status bits. You'd have to test that.*

Hippy, can you explain me how to obtain this value: "SSPBUF"? I've tried to search all over and can't figure out how to do it...
Hippie, pouvez-vous expliquer moi comment obtenir cette valeur: "SSPBUF"? J'ai essayé de chercher partout et ne peux pas comprendre comment le faire ...

You need to look in the appropriate datasheet for the underlying PICmicro, then under Memory Organisation find the address of SSPBUF. For the X2 PICAXE's this is $FC9, or $C9 considering only the 8-lsb's of the address.

Then this should work ...

Vous devez regarder dans la fiche technique appropriée pour le PICmicro sous-jacente, puis sous Organisation Mémoire de trouver l'adresse de SSPBUF. Pour cela est $ FC9 du X2 PICAXE ou C9 $ ne considérant que les 8-LSB de l'adresse.

Ensuite, cela devrait fonctionner ...

Symbol SSPBUF = $C9

HSpiOut ( $xx )
PeekSfr SSPBUF, b0

Here's a short demo program for the 20X2. Connect SPI SDO to SPI SDI on the X2, ignore SPI CLK. This sends a byte, reads the value in SSPBUF ( it's what was simultaneously sent and recirculated back via the SDO to SDI wire ), displays, it chooses next number and so on ...

Voici un programme de démonstration court pour le 20X2. Connecter SPI SDO à SPI SDI sur le X2, ignorer SPI CLK. Cela envoie un octet, lit la valeur dans SSPBUF (il est ce qui a été envoyé simultanément et recirculé via le SDO à fil SDI), affiche, il choisit le numéro suivant et ainsi de suite ...

Code:

#Picaxe 20X2

#No\_Table

#No\_data

#Terminal 9600

HSpiSetup SPIMODE10e, SPISLOW

Symbol SSPBUF = $C9

Do

 HSpiOut (b0)

 PeekSfr SSPBUF, b0

 SerTxd( #b0, " " )

 b0 = b0 + 1

 Pause 1000

Loop

# <http://www.picaxeforum.co.uk/showthread.php?19036-Using-INHAOS-RF2400-2-4-gHz-RF-Modules&highlight=NRF24L01>

# Discussion: [Using INHAOS RF2400 2.4 gHz RF Modules](http://www.picaxeforum.co.uk/showthread.php?19036-Using-INHAOS-RF2400-2-4-gHz-RF-Modules)

The RF2400x is a 2.4Ghz transceiver module from INHAOS ( China). These modules are meant **to be compatible with the Nordic NRF24L01+ modules**. While these modules can communicate with the Nordic modules, the programing is slightly different.

The INHAOS modules cost $2.99 each with free shipping from China, compared to > $19.00 each plus shipping for the Nordic modules from Spark Fun. Functionally they are basically the same, however the Nordic Module has an on board 3.6V regulator and can take 5v Vdd. The INHAOS module has no regulator and can only take Vdd up to 3.6 volts. The Nordic Module module has an Antenna Factor chip antenna while the INHAOS has a PCB trace antenna. This gives Nordic module an edge on range. However the INHAOS has a max output of 5 dBm compared to the Nordic 0dBm. It would be easy enough to peel off the trace antenna on the INHAOS and solder on a chip antenna . This would give the INHAOS a big range advantage over the NORDIC with 5dBm vs 0dBm.

1. The INHAOS has no 250Khz AIR Data Rate option.
2. To write to registers 28 & 29 you must first write the "Activate" command byte
then write the register and then do a hard reset.
3. A hard reset may be required after changing the default receive addresses for the Data Pipes
Suggest keeping the default addresses unless multiple star networks are set up in
the same location.

These programs are for basic testing or the INHAOS RF2400 Transceivers. One program sets the Module
as a dedicated receiver and the other as a dedicated transmitter. Automatic packet handling is not enabled for the purpose of this demo. These are intended to show how to program the modules and to use the HSPI interface and to do range testing.

I will post code for automatic packet handling soon.

There are several subroutines ( functions) that are not used. These functions have been tested to
work and can be used in a modified program.

The code is written for a 20x2. You will need to modify the pins / variables to suit your Picaxe

Load I\_2400\_TX into one Picaxe/Module combo
Load I\_2400\_RX into another Picaxe/Module combo
Keep Download connector connected to receive module.
Hard reset both boards.

NOTE: The receiver was detuned to allow close testing between modules. For maximum range, you will need to change the line for register 6 in config\_registers to:

Le RF2400x est un module émetteur-récepteur 2.4Ghz de INHAOS (Chine). **Ces modules sont destinés à être compatibles avec les modules Nordic nRF24L01 +**. Bien que ces modules peuvent communiquer avec les modules nordiques, la programmation est légèrement différente.

Les modules de INHAOS coûtent 2,99 $ chacun avec la livraison gratuite à partir de la Chine, par rapport à 19,00> $ chacun plus frais pour les modules nordiques de Spark Fun. Fonctionnellement, ils sont fondamentalement les mêmes, mais le module Nordic dispose d'un régulateur de 3.6V à bord et peuvent prendre 5v Vdd. Le module INHAOS n'a pas de régulateur et ne peut prendre Vdd jusqu'à 3,6 volts. Le module Module nordique a une antenne à puce Antenna Factor tandis que le INHAOS a une antenne de trace de PCB. Cela donne le module nordique un bord sur la gamme. Cependant, le INHAOS a une sortie maximum de 5 dBm par rapport à la 0dBm nordique. Il serait assez facile de décoller l'antenne de trace sur le INHAOS et la soudure sur une antenne à puce. Cela donnerait à la INHAOS un gros avantage de gamme sur NORDIC avec 5dBm vs 0dBm.

1. Le INHAOS n'a pas d'option 250kHz Data Rate AIR.

2. Pour écrire aux registres 28 et 29 vous devez d'abord écrire l'octet "Activer" de commande

puis écrire le registre puis effectuez une réinitialisation matérielle.

3. Une réinitialisation matérielle peut être nécessaire après la modification de la valeur par défaut de recevoir des adresses pour les tuyaux de données

Proposer en gardant les adresses par défaut, sauf si les réseaux d'étoiles multiples sont mis en place dans

au même endroit.

Ces programmes sont destinés aux tests de base ou l'INHAOS RF2400 transceivers. Un programme définit le module

un récepteur dédié et l'autre comme un émetteur dédié. le traitement automatique de paquets n'a pas été activé dans le but de cette démonstration. Elles sont destinées à montrer comment programmer les modules et utiliser l'interface HSPI et de faire tester la portée.

Je vais poster le code pour la gestion automatique des paquets bientôt.

Il existe plusieurs sous-programmes (fonctions) qui ne sont pas utilisés. Ces fonctions ont été testées

travailler et peut être utilisé dans un programme modifié.

Le code est écrit pour un 20x2. Vous aurez besoin de modifier les broches / variables en fonction de votre Picaxe

Charge I\_2400\_TX dans un Picaxe / combo Module

Charge I\_2400\_RX dans un autre Picaxe / combo Module

Gardez le connecteur de téléchargement connecté pour recevoir le module.

Hard Reset les deux cartes.

NOTE: Le récepteur a été désaccordé pour permettre de tester à proximité entre les modules. Pour une portée maximale, vous devrez changer la ligne pour le registre 6 en config\_registers à:

Code:

'write Register 6

low csn

**hspiout ($26,%00000111)** 'REG 6 Data Rate 1 mbps LNA Normal Output 5 dbm

high csn

Have fun !

20 x 2 Connections
Physical Pins ....................... RF2400 Pins
--------------------------------------------------
Picaxe .................................. RF2400
---------------------------------------------------
Pin 1 Vdd ( 3.6v)
Pin 2 SerIn
Pin 3 Serout
Pin 4 NC
Pin 5 C.5 ................. > INHAOS IRQ Pin 8 / **10K Pullup**
Pin 6 NC
pin 7 NC
Pin 8 NC
Pin 9 HSPI SDO .............. > INHAOS MOSI Pin 6
Pin 10 NC
Pin 11 SPI Clock ........... > INHAOS SCK Pin 5
Pin 12 NC
Pin 13 HSPI SDI ............ > INHAOS MISO Pin 7
Pin 14 B.4 ................. > INHAOS CSN Pin 4
Pin 15 NC
Pin 16 NC
Pin 17 B.1 ................. > INHAOS CE Pin 3
Pin 18 NC
Pin 19 NC
Pin 20 Ov

I now have these modules communicating with automatic packet handling and auto ACK. Almost identical code to the Nordic nRF24L01+ with a few minor but important differences.

If enough are interested, I will write a detailed tutorial on how to use this transceiver with a Picaxe. The more that reply the more motivated i will be to write it up.

J'ai maintenant ces modules communiquant avec la manipulation de paquets ACK automatique et automatique. Code Presque identique à la Nordic nRF24L01 + avec quelques différences mineures mais importantes.

Si un nombre suffisant sont intéressés, je vais écrire un tutoriel détaillé sur la façon d'utiliser cet émetteur-récepteur avec un Picaxe. Plus cette réponse, les plus motivés, je serai à écrire vers le haut.

I am certainly interested in these modules and would love it if you get sufficient interest to warrant effort of writing a tutorial.

The devices seem just right for use in a beehive instrumentation and logging application, where a significant number of such devices could eventually be used.

Je suis certainement intéressé par ces modules et aimerais si vous obtenez un intérêt suffisant pour justifier l'effort d'écrire un tutoriel.

Les dispositifs semblent juste pour une utilisation dans une instrumentation et applications de logging, où un nombre important de ces dispositifs pourrait éventuellement être utilisé.

 Envoyé par **Goeytex** 

*I now have these modules communicating with automatic packet handling and auto ACK. Almost identical code to the Nordic nRF24L01+ with a few minor but important differences.

If enough are interested, I will write a detailed tutorial on how to use this transceiver with a Picaxe. The more that reply
the more motivated i will be to write it up.*

Count me among those interested. I have a pair in transit from China - should be here in a couple of weeks.
Comptez-moi parmi ceux qui sont intéressés. J'ai une paire en transit en provenance de Chine - devrait être ici dans quelques semaines.
John

## Par défaut

 Envoyé par **papaof2** 

*Count me among those interested. I have a pair in transit from China - should be here in a couple of weeks.

John*

Hey John,

I got mine in about 7 days. Shipped to Texas

 Envoyé par **Paix** 

*The devices seem just right for use in a beehive instrumentation and logging application, where a significant number of such devices could eventually be used.*

Yes.

They work extremely well where long range is not needed. With over 100 channels and an almost infinite number of addresses a lot of these could be set up in close proximity without conflict.

Ils travaillent très bien où longue portée est pas nécessaire. Avec plus de 100 canaux et un nombre presque infini d'adresses beaucoup d'entre eux pourraient être mis en place à proximité sans conflit.

I have decided not to do a formal tutorial at this time. But the good news is ....

I have written the attached code that is highly commented and includes a whole bunch of function routines that will drastically simplify programming and shorten your learning curve. Each function is commented .

All you have to do is change/write code in the program loop while calling the functions as needed. There are no "gotos, only gosubs (calls). So be sure keep track and not get nested too deep.

This code should act as a tutorial as you read & study it and experiment with these very nice little modules. If you find a bug in any of the functions, please let me know.

Note that I have changed the CSN pin to b.4 to allow use of hserin on b.6 if so desired.

I did my homework now you get the rewards !

Note:
Picaxe interpreter overhead will prevent really hi speed data transfer in certain modes,
However sending a 32 byte packet with ACK every few ms should be very doable when
running an X2 Picase at 64mhz.

For full speed data streaming use Shockburst mode + DYNPD & no ACK which basically takes the Picaxe out of the picture except for sending the data via hspi to the RF2400. Just keep pumping data to the TX FIF0S as fast as you can < 2mbps) . and it auto-magically appears on the receiving module. You could stream audio or even some video with this module with the air data rate set to 2 mbps and HSPI at ful speed.
Refer to the BK2401 datasheet or the Nordic NRF24L01 + Datasheet. Just need need to digitize the audio or video data first. In this scenario the Picaxe is simply used to initially configure the registers and act as "glue logic."

J'ai décidé de ne pas faire un tutoriel formel à ce moment. Mais les bonnes nouvelles sont ....

J'ai écrit le code ci-joint qui est fortement commenté et comprend toute une série de routines de fonction qui va considérablement simplifier la programmation et de raccourcir votre courbe d'apprentissage. Chaque fonction est commentée.

Tout ce que vous avez à faire est de changer le code / écriture dans la boucle de programme tout en appelant les fonctions selon les besoins. Il n'y a pas "gotos, seulement gosubs (appels). Alors, soyez sûr garder la piste et ne pas s'imbriquée trop profondément.

Ce code devrait agir comme un tutoriel que vous lisez et étudiez et expérimenter avec ces très bons petits modules. Si vous trouvez un bogue dans l'une des fonctions, s'il vous plaît laissez-moi savoir.

Notez que j'ai changé la broche CSN à B.4 pour permettre l'utilisation de hserin sur b.6 si désiré.

Je faisais mes devoirs maintenant vous obtenir les récompenses!

Remarque:

Picaxe interprète les frais généraux permettra d'éviter le transfert de données de vitesse vraiment salut dans certains modes,

Cependant l'envoi d'un paquet de 32 octets avec ACK toutes les quelques ms devraient être très faisable lorsque

l'exécution d'une Picase X2 à 64MHz.

Pour les données streaming de vitesse utiliser le mode ShockBurst + DYNPD & no ACK qui prend essentiellement la Picaxe hors de l'image, sauf pour l'envoi des données via HSPI le RF2400. Il suffit de continuer à pomper des données aux FIF0S TX aussi vite que possible <2mbps). et il apparaît comme par magie sur le module de réception. Vous pouvez diffuser de l'audio ou même une vidéo avec ce module avec le débit de données d'air réglé à 2 mbps et HSPI à vitesse ful.

Reportez-vous à la fiche BK2401 **ou Nordic nRF24L01 +** Fiche technique. Juste besoin besoin de numériser les données audio ou vidéo en premier. Dans ce scénario, le Picaxe est simplement utilisé pour configurer initialement les registres et agissent comme «logique de colle».

 Fichiers attachés

*  [RF\_2400\_FUNCT.bas](http://www.picaxeforum.co.uk/attachment.php?attachmentid=8836&d=1313316606) (11,6 Ko, 225 affichages)

Many thanks Goeytex, I'm sure that what you have provided should more than suffice and is gratefully received.

I'm on the verge of being away for a few weeks, so can't take the chance of an order for RF modules getting sent back to China by the Post Office because there is no one to sign for them, so will pick up three to play with when I get back.

Un grand merci Goeytex, je suis sûr que ce que vous avez fourni devrait plus que suffisant et est reçu avec gratitude.

Je suis sur le point d'être absent pendant quelques semaines, ne peut donc pas prendre le risque d'une commande de modules RF s'envoyé en Chine par le bureau de poste, car il n'y a personne à signer pour eux, donc vais ramasser trois à jouer avec quand je reviens.

Thank you! A pair of modules should be here Tuesday, so I have time to read the code before I start to play ;-)

Je vous remercie! Une paire de modules devrait être ici mardi, je dois donc le temps de lire le code avant que je commence à jouer;-)

John

Very interesting, and thanks very much for the well-commented code.

I found the data sheet for the module (under "Download", here: <http://www.inhaos.com/product_info.php?products_id=35>), but not for the chip. Can you point to where you got the information about controlling the chip?

 Envoyé par **lbenson** 

*Very interesting, and thanks very much for the well-commented code.

I found the data sheet for the module (under "Download", here:*[*http://www.inhaos.com/product\_info.php?products\_id=35*](http://www.inhaos.com/product_info.php?products_id=35)*), but not for the chip. Can you point to where you got the information about controlling the chip?*

<http://www.inhaos.com/downcount.php?download_id=41>

The file to download is RF2410U DocumentPack.rar. The BK2421 Datasheet is in a sub folder.

I also relied upon my experience with the Nordic nRF24L01+ as most of programming commands are the same. [http://www.nordicsemi.com/kor/Produc...z-RF/nRF24L01P](http://www.nordicsemi.com/kor/Products/2.4GHz-RF/nRF24L01P)

Thanks for the links, and thanks again for all the work in making this picaxeable.

Hi
I am running the I\_2400\_RX.bas in the simulator trying to learn how it is working but it gives me a ''stack error - more nested gosubs than the stack allows'' after the program loops 8 times.
also it seems to miss one digit to the instruction on line74 **hspiout (%00100011,%0000001)**
I think it should be hspiout (%00100011,%00000011)
Je cours les I\_2400\_RX.bas dans le simulateur en essayant d'apprendre comment il fonctionne, mais il me donne une '' erreur de pile - gosubs plus imbriqués que la pile permet '' après le programme boucle 8 fois.

il semble aussi manquer un chiffre à l'instruction sur line74 hspiout (% 00100011,% 0000001)

Je pense qu'il devrait être hspiout (% 00100011, 00000011%)
Cacho

It appears to be instantly re-interrupting before the return command.

Il semble être instantanément ré-interrompre avant que la commande de retour.

I also notice that the I\_2400\_TX.bas is doing the same thing, could it be a bug in the simulator program? I am using the 5,4,0 version

Like to see what goeytex thinks about it.

Je remarque aussi que les I\_2400\_TX.bas est en train de faire la même chose, pourrait-il être un bogue dans le programme de simulateur? J'utilise la version 5.4.0

Comme pour voir ce que goytex pense.

For clarity the the extra "0" should be included but hspi will still send out a Hex byte "1" or a Decimal byte "1" regardless. In other words a binary 000001 is the same as a binary 00001 or a 0001. The code operation should not be affected. That was simply a typo, that does not affect anything. Certainly change it in your code though.

As far as the stack error goes, my guess is that the code does not necessarily lend itself to simulation because the interrupts are not being generated in the simulator properly. The simulator is not aware of the pullup resistor on c.5 which  is used to detect the interrupt generated by the RF module on data receive, therefore going directly to the interrupt routine without actually receiving data. Set the interrupt pin (c.5) high in the simulator before running the code and then step through it.

To simulate a data packet received in the receiver code momentarily take pin C.5 low.

Yes that was it. It simulates ok if you put c.5 high before stepping through the code.

Pour plus de clarté l'extra "0" devrait être inclus, mais HSPI va quand même envoyer un octet Hex "1" ou un octet Décimal "1" indépendamment. Autrement dit binaire 000001 est identique à un binaire 00001 ou 0001. L'opération de code ne doit pas être affectée. Ce fut tout simplement une faute de frappe, cela ne porte pas atteinte à quoi que ce soit. Certainement changer dans votre code cependant.

En ce qui concerne l'erreur de la pile va, je suppose que le code ne se prête pas nécessairement à la simulation parce que les interruptions ne sont pas générées dans le simulateur correctement. Le simulateur est pas au courant de la résistance de pullup sur c.5 qui est utilisé pour détecter l'interruption générée par le module RF sur les données reçoivent donc aller directement à la routine d'interruption sans données réellement réception. Réglez la broche d'interruption (c.5) élevée dans le simulateur avant d'exécuter le code, puis l'étape à travers elle.

Pour simuler un paquet de données reçu dans le code du récepteur prendre momentanément broche C.5 faible.

Oui, ce fut elle. **Il simule ok si vous mettez c.5 haute** avant de monter dans le code.

Goeytex, I'm playing with a couple of the NRF24L01+ and am trying to get them going with a 20x2 without any joy.

Do you have any simpler code snippets that can steer me in the right direction as I'm feeing particularly dumb not being able to get these talking !

Goeytex, je joue avec un couple de l'nRF24L01 + et essaie de les faire aller avec un 20x2 sans joie.

Avez-vous des extraits de code plus simples qui peuvent me orienter dans la bonne direction que je suis particulièrement stupide véoyant ne pas être en mesure d'obtenir ces parler!

@FlyBoy

Attached is 20x2 Code for the NRF2401+ It should get you going.

Pay attention to the notes in the code as to where to include pull up and pull down resistors.

Don't let the size of the code daunt you. While not exactly "simple", it is really quite "straight forward" and is organized into sub routines to make testing & experimenting easier.

I am working on a 20 dBm version of this module/chip that will have some better range.

Attaché est Code 20x2 pour le nRF2401 + Il devrait vous permettre de continuer.

Faites attention aux notes dans le code à l'endroit où inclure tirer vers le haut et tirer vers le bas des résistances.

Ne laissez pas la taille du code Daunt. Bien que pas exactement "simple", il est vraiment tout à fait «tout droit» et est organisé en sous-programmes pour faire des essais et des expériences plus facile.

Je travaille sur une version 20 dBm de ce module / puce qui aura une certaine meilleure gamme.

 Fichiers attachés

*  [24NLF01-4-PTX .bas](http://www.picaxeforum.co.uk/attachment.php?attachmentid=11164&d=1336122062) (5,3 Ko, 110 affichages)
*  [24NLF01-4-PRX .bas](http://www.picaxeforum.co.uk/attachment.php?attachmentid=11165&d=1336122063) (4,7 Ko, 75 affichages)

Thanks Goeytex,

an evening with your code, the manual, and the digital 'scope narrowed down the culprit - the breadboard !

Scotch for you whenever you're in this part of the world 

These aren't that bad once you get the hang of them

une soirée avec votre code, le manuel et un scope numérique réduit le coupable - la platine d’essai!

Scotch pour vous quand vous êtes dans cette partie du monde

Ce ne sont pas si mal une fois que vous obtenez le coup d'entre eux

I have completely re-written the code for these INHAOS RF modules. It is much cleaner now and also supports the new 20dBM RF2400PA module. Look soon for an even more powerful 2400 MHz Transceiver from INHAOS with range > 1Km .
Je suis complètement ré-écrit le code pour ces modules INHAOS RF. Il est beaucoup plus propre maintenant et prend également en charge le nouveau module 20dBm RF2400PA. Regardez bientôt pour un encore plus puissant 2400 MHz Transceiver de INHAOS avec la gamme> 1 km.
Attached is the new code and a schematic.

 Fichiers attachés

*  [RF2400\_PICAXE \_DEMO.bas](http://www.picaxeforum.co.uk/attachment.php?attachmentid=11955&d=1344610039) (8,4 Ko, 101 affichages)

Hi Goeytex
I have notice that you have added a 10K pullup resistor r1 in your new schematic. Why since c.5 pin is not open collector?
Also you failled to put a resistor on both LED's connected on b.1 b.2

Thanks for the new code and schematic
salut Goeytex

J'ai remarqué que vous avez ajouté une résistance r1 10K pullup dans votre nouveau schéma. Pourquoi depuis broche c.5 est pas collecteur ouvert?

Aussi, vous failled de mettre une résistance sur les deux LED connecté sur b.1 b.2

Merci pour le nouveau code et schématique
Cacho

Good Catch on the LED resistors. I will ad them on the schematic.

However I do not see how "open collector" has any relevance to the pullup resistor on Pin C.5. Pin C.5 is an input used to detect the low interrupt signal supplied by the RF2400 IRQ output. At power up C.5 defaults to an input and will be floating until the RF2400 supplies either a high or a low signal. Putting a pullup on c.5 insures that the pin will not float and cause a false interrupt.

Good Catch sur les résistances de LED. Je vais ad eux sur le schéma.

Cependant, je ne vois pas comment "collecteur ouvert" a un rapport quelconque avec la résistance de pullup sur Pin C.5. C.5 broche est une entrée utilisée pour détecter le signal basse d'interruption fournie par la sortie RF2400 IRQ. Au démarrage C.5 par défaut à une entrée et sera flottante jusqu'à ce que les fournitures de RF2400 soit un haut ou un signal faible. Mettre un pullup sur c.5 assure que la broche ne flotte pas et provoquer une fausse alarme.

Hello Goyetex, nice work out there!
I'm also trying to run Inhaos RF2400P but with my AVR project. I've got similiar library but for nRF24L01+. You mentioned in first page that there are "minor but important differences" between those two modules. Could You put some light on those differences, please?
I cannot connect between two modules, I don't know what I'm doing wrong...

Bonjour Goyetex, beau travail là-bas!

Je suis aussi essayer de courir Inhaos RF2400P mais avec mon projet AVR. J'ai bibliothèque semblable mais pour nRF24L01 +. Vous avez mentionné dans la première page qu'il ya des «différences mineures mais importantes» entre ces deux modules. **Pourriez-vous mettre un peu de lumière sur ces différences, s'il vous plaît?**

Je ne peux pas se connecter entre les deux modules, je ne sais pas ce que je fais mal ...

The differences are that the INHAOS RF module does not support the 250Khz data rate and that it has a second register bank (Bank1) that can only be accessed by first using the the ACTIVATE command byte followed by 0x53.

Bank0 must be active in order for the RF2400 module to work. ACTIVATE followed by 0x53 toggles the active bank. You can check which bank is active by reading Status register, bit 7. The best way to to this is to send 0xFF (NOP) to the module and then read in the status register bits. If bit 7 = 0 then Bank0 is active. If bit 7 = 1 then Bank1 is active. Most of the the bank 1 registers are undocumented and write only.

You can find AVR reference code [HERE](http://www.inhaos.com/downcount.php?download_id=33)

**Les différences sont** que le module RF INHAOS ne supporte pas le débit de données de 250kHz et qu'il a une seconde banque de registres (Bank1) qui ne peut être accessible en utilisant d'abord l'octet de commande ACTIVATE suivi par 0x53.

BANK0 doit être actif pour que le module RF2400 fonctionne. ACTIVATE suivi par 0x53 basculer la banque active. Vous pouvez vérifier quelle banque est active en lisant le registre d'état, bit 7. La meilleure façon de cette est d'envoyer 0xFF (NOP) au module, puis lu dans les bits de registre d'état. Si le bit 7 = 0 alors BANK0 est active. Si le bit 7 = 1 alors Bank1 est active. La plupart des la banque 1 entrées sont sans papiers et écrire seulement.

Vous pouvez trouver le code de référence AVR ICI

glad i found this, i have a couple of RF 2400 nano modules coming in a few days hopefully.
i have not worked with rf with the picaxe yet or any sort of spi so im a bit uneasy at the moment but have read the datasheet for the chip and have read through your provided code goeytex which will save alot of first time headaches,thanks.
hopefully it wont be too daunting of a task to finally get my head around and get working successfully.

content d'avoir trouvé ce, j'ai un couple de RF 2400 modules de nano venir dans quelques jours nous l'espérons.

je n'ai pas travaillé avec rf avec le picaxe encore ou toute sorte de spi ainsi im un peu mal à l'aise pour le moment mais ai lu la fiche technique de la puce et avoir lu à travers votre code fourni goeytex qui permettra d'économiser beaucoup de premiers maux de tête de temps, merci.

nous l'espérons il ne sera trop intimidante d'une tâche pour finalement obtenir ma tête autour et faire fonctionner avec succès.

after going through the datasheet again and referring to the test code you have provided i am starting to get a better understanding of the basics of starting to use these modules,thankfully it does not seem so daunting now and im actually quite eager to recieve these now and have a little play around, they were sent out on the 5th and the tracking shows its being classed as an expresspost package so hopefully not too long before i have them, i bought them because they were offered at a damn good price and so small (12mmx12mm) that they can be squeezed into the tiniest of spaces.
so until they arrive i shall keep re-reading both the datasheet and your highly commented and informative code provision and start to work out a basic interpretation of what i can use as a starting ground to move on from.

thanks for the writeup bill it most definately has helped in understanding these modules and no doubt saved alot of time with trying to understand a new area of which i have yet avoided

après avoir traversé la fiche à nouveau et en se référant au code de test que vous avez fourni je commence à avoir une meilleure compréhension des bases de commencer à utiliser ces modules, heureusement, il ne semble pas si intimidante maintenant et im en fait assez désireux de recevoir ces maintenant et ont un peu de jeu autour, ils ont été envoyés sur le 5ème et le suivi montre son être classé comme un paquet Expresspost donc heureusement pas trop longtemps avant que je les ai, je l'ai acheté parce qu'ils ont été offerts à un putain de bon prix et si petit (12mmx12mm) qu'ils peuvent être entassés dans le plus petit des espaces.

jusqu'à ce qu'ils arrivent i conserverai re-lecture à la fois la fiche et votre disposition de code très commenté et informatif et commencer à travailler sur une interprétation de base de ce que je peux utiliser comme motif de départ pour se déplacer à partir.

merci pour le projet de loi writeup le plus certainement a contribué à la compréhension de ces modules et sans aucun doute sauvé beaucoup de temps à essayer de comprendre un nouveau domaine dont j'ai encore évité

INHAOS has provided me with several new high power 2.4GHz transceiver modules
for testing. These have a PA with TX power > 20dBm and should give an LOS
range of > 1000 meters. I will be writing evaluation / demo code for these over
the holidays and will post in a new thread when done.

May even try doing some frequency hopping with these ....

INHAOS m'a fourni plusieurs nouveaux modules émetteurs-récepteurs haute puissance 2.4GHz

pour tester. Ceux-ci ont un PA avec la puissance TX> 20dBm et devraient donner une LOS

gamme de> 1000 mètres. Je vais écrire le code d'évaluation / de démonstration pour ces cours

les vacances et affichera dans un nouveau fil lorsque vous avez terminé.

Peut même essayer de faire quelques sauts de fréquence avec ces ....

i look forward to reading that, i still shamefully have yet to play with my modules due to being occupied with other projects for my brother but have ordered another 2 rf nano modules to add to the others, just have to invest in a few 20x2 chips as i dont feel like even attempting the m2 bit bang method lol

Je suis impatient de lire cela, je encore ai honteusement encore jouer avec mes modules en raison d'être occupé avec d'autres projets pour mon frère, mais ai commandé 2 autres modules rf nano pour ajouter aux autres, viens d'investir dans quelques copeaux de 20x2 que je ne me sens pas comme même tenter la méthode de bang bit m2 lol

Goeytex,

How is the code for the new modules coming along? And can you point us toward the description on their website?
Comment le code pour les nouveaux modules est long à venir? Et pouvez-vous nous pointer vers la description sur leur site web?
Jeff

Hi Jeff,

INHAOS has provided me with 2 versions of a high power 2.4 GHz module for testing. One with a PCB antenna and a pre-release version with SMA connector. The SMA version is said to have an LOS range > 800 meters. I also have an RS232 low power version that needs to be tested. The code for the high power modules will be nearly the same as the code posted in this thread the only difference being that there is a PA on board and a PA switch that needs to be controlled when switching from RX to TX mode.

I am currently finishing up code on a Picaxe controlled RSSI meter. After that I will test and post test data on these INHAOS modules.

Look back here in about a week or so or send me a PM.

INHAOS m'a fourni avec 2 versions d'un module de haute puissance GHz 2.4 pour les tests. Une avec une antenne de PCB et d'une pré-version avec connecteur SMA. La version SMA est dit d'avoir une gamme LOS> 800 mètres. Je également une version de faible puissance RS232 qui doit être testé. Le code pour les modules de grande puissance sera à peu près le même que le code affiché dans ce fil à la seule différence qu'il y a un PA à bord et un commutateur PA qui doit être contrôlé lors du passage du RX au mode TX.

Je termine actuellement en place un code sur un compteur RSSI Picaxe contrôlé. Après cela, je vais tester et publier des données de test sur ces modules INHAOS.

Regardez en arrière ici dans environ une semaine ou envoyez-moi un PM.

[2400PA Module](http://www.inhaos.com/product_info.php?products_id=54)

[LC1000 Module](http://www.inhaos.com/product_info.php?products_id=60)

[RF2400PA Datasheet](http://www.inhaos.com/downcount.php?download_id=64)

It's been almost a month...how's the code coming for these?

Il a été presque un mois ... comment est le code à venir pour ceux-ci?

16-08-2013, 00:01

studying routines: 24NL01-4-PTX and PRX 24NL01-4-thread mentioned in # 18, there are lines of code that confused me, the mention:

étudier routines: 24NL01-4-PTX et PRX 24NL01-4-fil mentionnés dans # 18, il y a des lignes de code qui me confondre, la mention:

Code:

symbol IRQ = pinb.0 'hardware IRQ1

symbol CE = B.1 'Chip Enable

symbol CSN = B.6 'Chip Select Active low)...

but later in the routine that responds to the interrupt appears that:

mais plus tard dans la routine qui répond à l'interruption apparaît que:

Code:

setint %00000000,%00100000,c

I think there is a confusion of using the IRQ pin that reads, I think mixed ideas interrupt handling in programs for nrf24LF01 and inhaos rf2400 ... or am I mistaken?

Je pense qu'il ya une confusion d'utiliser la broche IRQ qui lit, je pense que les idées mixtes gestion des interruptions dans les programmes pour nrf24LF01 et inhaos rf 2400 ... ou je me trompe?

.



; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*\*\*\*\* **INHAO 2400** **Transmit Test** \*\*\*\*\*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Filename: **I\_2400\_TX.bas**

; Date: 08-02-2011

; File Version: Beta Test

; Written by: Bill Roth / Goeytex

; Target PICAXE: 20X2

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' \* Open Program Editor

' \* Options > Editor > Use Collapsing Blocks \*

' \* Options > Editor . Serial Termina . Open Ater Download

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#picaxe 20x2

#no\_data

#no\_table

**symbols:**

{

symbol config = 0 'Address of Config Register

symbol status = 7 'Address of Status Register

symbol write\_tx\_fifo = 160 'Command Byte (%10100000) See Datasheet

symbol write\_tx\_addr = 48 'Address 16 DEC + 32 DEC = "Write to Address"



symbol pipe\_0 = 231 'Address

symbol pipe\_1 = 194 'Address

symbol pipe\_2 = 195 'Address

symbol pipe\_3 = 196 'Address

symbol pipe\_4 = 197 'Address

symbol pipe\_5 = 198 'Address

symbol CE = B.1 'Chip Enable

symbol CSN = B.6 'Chip Select/Active low)

symbol data1 = b0 'Dynamic Variables

symbol addr = b1

}

**init:**

{

input b.0

output b.1

output b.6

setfreq m16

hspisetup spimode00,spifast

high CSN 'active low

low CE

**gosub clear\_interrupts**

setint %00000000,%00100000,c 'Picaxe 20x2 Interrupt Pin5 Low

}

**Config\_Registers:**

{

'write register 0

'enable PtX, power off, en\_crc/2bytes, interrupts unmasked

low csn

hspiout ($20,%00001100)

high csn

pause 1

;write register 1

low csn

hspiout ($21,%00000000) ;Shockburst OFF ALL Pipes (no ACK)

high csn

;write register2

low csn

hspiout ($22,%00111111) ' Enable all Pipes

high csn ' All Pipes can receive Data

pause 1

;write register 3

low csn

hspiout ($23,%00000001) ' RX Address width = 5 bytes wide

high csn

;write register 4

low csn

hspiout ($24,%00010000) '//ARC Disbled/ ARDelay = 500u

high csn ' See Datasheet

'write register 5

low csn

hspiout ($25,%00000010) '// RF channel = 2

high csn ' Receiver must match

'write Register 6

low csn

hspiout ($26,%00000110) 'REG 6 Data Rate 1 mbps

high csn

'write Regiters 17 - 22 'Payload Width

for b0 = 49 to 54 'Reg Address + 32 decimal to Write

low csn 'Could also do this in hex

hspiout (b0,5) 'Pipe\_0 - Pipe\_5 Payload Width = 5

high csn

next b0

'write ACTIVATE (see RF2400 Datasheet)

low csn

hspiout (%01010000,$73) '//Enable Writing to feature Register

high csn 'Not Necessery with Nordic NRF2401+

'Reg 29 "Feature Register"

low csn

hspiout (%00111101,%00000001) 'Enable No\_Ack (Not used with this Demo)

high csn 'Hard Reset of RF2400 Needed

} 'Configure regiaters & set TX mode

**MAIN:**

{

'========================================================

'This routine powers up the RF2400 and sends 5 bytes of

'data to each of the Data Pipes starting with Pipe\_0.

'The Pipe Addresses are the default values.

'A Hard Reset of the RF2400 is necessary after changing

'RX Pipe Addresses on Receiver.

'The receiving RF2400 will receive the data on the proper

'data Pipe, then dipslay the Pipe if was received on and

'the values of each of the 5 data bytes.

'The receiving RF2400 Must be running I\_2400\_RX.bas

'=========================================================

**gosub power\_on** 'turn RF2400 Power On

pause 2

**do**

pause 5000 'these 5000 pauses are just to make things easier

 'to see in this test/demo and have nothing

 'to do with timing

'Set tx address of the Data Pipe to send data to:

Low csn

hspiout (write\_tx\_addr,pipe\_0,pipe\_0,pipe\_0,pipe\_0,pipe\_0)

high csn 'see "Symbols" for Address Values"

'Write Data to TX FIFO

low csn

hspiout (write\_tx\_fifo,55,241,77,83,177) 'write 5 bytes to tx fifo

high csn

pulsout CE,100 'TRANSMIT !

pause 5000

;set tx address to Pipe\_1

low csn

hspiout (write\_tx\_addr,pipe\_1,pipe\_1,pipe\_1,pipe\_1,pipe\_1)

high csn

low csn

hspiout (write\_tx\_fifo,63,198,23,87,143) 'write 5 byts to tx fifo

high csn 'writing TX address

pulsout CE,100 'TRANSMIT !

pause 5000

;set tx address to Pipe\_2

low csn

hspiout (write\_tx\_addr,pipe\_2,pipe\_1,pipe\_1,pipe\_1,pipe\_1)

high csn

low csn

hspiout (write\_tx\_fifo,67,174,29,88,146) 'write 5 byts to tx fifo

high csn 'writing TX address

pulsout CE,100 'TRANSMIT !

pause 5000

;set tx address to Pipe\_3

low csn

hspiout (write\_tx\_addr,pipe\_3,pipe\_1,pipe\_1,pipe\_1,pipe\_1)

high csn

low csn

hspiout (write\_tx\_fifo,99,182,143,187,113) 'write 5 byts to tx fifo

high csn 'writing TX address

pulsout CE,100 'TRANSMIT !

pause 5000

;set tx address to Pipe\_4

low csn

hspiout (write\_tx\_addr,pipe\_4,pipe\_1,pipe\_1,pipe\_1,pipe\_1)

high csn

low csn

hspiout (write\_tx\_fifo,39,101,213,116,111) 'write 5 byts to tx fifo

high csn 'writing TX address

pulsout CE,100 'TRANSMIT !

pause 5000

;set tx address to Pipe\_5

low csn

hspiout (write\_tx\_addr,pipe\_5,pipe\_1,pipe\_1,pipe\_1,pipe\_1)

high csn

low csn

hspiout (write\_tx\_fifo,153,17,114,119,13) 'write 5 byts to tx fifo

high csn 'writing TX address

pulsout CE,100 'TRANSMIT !

**loop**

}

**interrupt:**

{

low csn

**gosub clear\_interrupts**

high csn

setint %00000000,%00100000,c

return

}

**Clear\_interrupts:**

{

'==========================================

'This function clears all RF2400 interrupts

'by clearing bits 4, 5, & 6 of register 7.

' while leaving the other bits unchanged &

'intact

'==========================================

addr = status

low csn

hspiout (addr,1,1)

hspiin (data1) 'Read Status Register into B0 (data\_1)

high csn

let bit4 = 1 'Modify B0

let bit5 = 1

let bit6 = 1

pause 200 'leave LED on a whiie

addr = status + 32 'write

low csn

hspiout (addr,data1) 'Write New Value of B0 to status register)

high csn

return

} 'Clear RF 2400 Interrupts

**get\_status:**

{

addr = status

low csn

hspiout (addr,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Status = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

} 'Read RF2400 status register and sertxd out

**get\_config:**

{

addr = Config

low csn

hspiout (addr,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Config = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

} 'Read RF2400 config register and sertxd out

**power\_on:**

{

'==============================

addr = config

low csn

hspiout (addr,1,1)

hspiin (data1)

high csn

let bit1 = 1

addr = config + 32 'write

low csn

hspiout (addr,data1)

high csn

return

'=============================

} 'Turn RF2400 Power On

**power\_down:**

{

addr = config

low csn

hspiout (addr,1,1)

hspiin (data1) '(B0)

high csn

let bit1 = 0

addr = config + 32

low csn

hspiout (addr,data1)

high csn

return

} 'Turn RF2400 Power Off

**prx\_on:**

{

addr = config

low csn

hspiout (addr,1)

hspiin (data1)

high csn

let bit0 = 1

addr = config + 32 'write

low csn

hspiout (addr,data1)

high csn

return

} 'Enable RF2400 Receive mode

**ptx\_on:**

{

addr = config

low csn

hspiout (addr,1)

hspiin (data1)

high csn

let bit0 = 0

addr = config + 32 'write

low csn

hspiout (addr,data1)

high csn

return

} 'Enable RF2400 Transmit mode

**read\_all:**

{

'=========================

sertxd (CR,LF,"Register values are in decimal format",CR,LF)

for b1 = 0 to 09 'registers 0 - 9 (decimal)

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

next b1

for b1 = 10 to 16 'Addresses with 5 byte registers

low csn

hspiout (b1,1,2,3,4,5)

hspiin (b4,b5,b6,b7,b8)

high csn

sertxd ("Register ",#b1," = ", #b4,#b5,#b6,#b7,#b8,CR,LF)

next b1

for b1 = 17 to 22

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

sertxd ("Register ",#b1," = ", #b0,cr,lf)

next b1

low csn

b1 = 23

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

low CSN

b1 = 28

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

inc b1

low CSN

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

return

} 'Read all RF2400 all registers and sertxd out

**serial\_out1:**

{

'======================serial\_out1:

sertxd ("Register ",#b1," = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

Return

} 'Used with Read\_all



; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*\*\*\*\* INHAO 2400 **Receiver Test** \*\*\*\*\*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Filename: **I\_2400\_RX.bas**

; Date: 08-02-2011

; File Version: Beta Test

; Written by: Bill Roth / Goeytex

; Target PICAXE: 20X2

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' \* Open Program Editor

' \* Options > Editor > Use Collapsing Blocks \*

' \* Options > Editor . Serial Termina . Open Ater Download

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#picaxe 20x2

#no\_table

#no\_data

**Symbols:**

{

symbol config = 0 'Address of Config Register

symbol status = 7 'Address of Status Register

symbol data1 = b0

symbol addr = b1

symbol CE = B.1 'Chip Enable

symbol CSN = B.6 'Chip Select Active low)

symbol read\_rx\_payload = %01100001 ' Command Byte - see datasheet

}

**init:**

{

input b.0

output b.1

output b.6

setfreq m16

hspisetup spimode00,spifast

high CSN 'active low

low CE

**gosub clear\_interrupts**

setint %00000000,%00100000,c 'Picaxe 20x2 Interrupt Pin5 Low

}

**Config\_registers:**

{

'write register 0

'enable PtX, power off, en\_crc/2bytes, interrupts unmasked

low csn

hspiout ($20,%00001101)

high csn

pause 1

;write register 1

low csn

hspiout (%00100001,%00000000) ;shockburst OFF (No ACK)

high csn ;MUST be set same in PTX

;write register 2 enable Pipes 0 - 5 (all)

low csn

hspiout (%00100010,%00111111)

high csn

;write register3

low csn

hspiout (%00100011,%0000001) ' Pipe ID Address = 3 bytes wide

high csn ' Default is 5 bytes wide

pause 1

;write register 4

low csn

hspiout ($24,%00010000) '//ARC Disbled/ ARDelay = 500u

high csn ' See Datasheet

'write register 5

low csn

hspiout ($25,%00000010) '// RF channel = 2

high csn ' Receiver must match

;write register 6

low csn

hspiout (%00100110,%00000110) ' REG 6 (Data Rate = 1 mbps)

high csn

'write Regiters 17 - 22 Payload Width

for b0 = 49 to 54 'Reg Address + 32 decimal to Write

low csn 'Could also do this in hex

hspiout (b0,5) 'Pipe\_0 - Pipe\_5 Payload Width = 5

high csn

next b0

'write ACTIVATE (see RF2400 Datasheet)

low csn

hspiout (%01010000,$73) '//Enable Writing to feature Register

high csn 'Not Necessery with Nordic NRF2401+

'Reg 29 "Feature Register"

low csn

hspiout (%00111101,%00000001) 'Enable No\_Ack (Not used with this Demo)

high csn 'Hard Reset of RF2400 Needed

} 'Configure registers & set RX mode

**Gosub read\_all**

pause 1000

**MAIN:**

low csn

hspiout (%00100000,%00001111) 'Write to config register' EN PRX + Power ON

high csn

setint %00000000,%00100000,c '("20x2 Pin5 LOW)

**do**

 high CE ' Receiver enabled

 ' wait for interrupt

**loop**

**interrupt:**

{

pause 200

low csn

data1 = 0

addr = status

low csn

hspiout (addr,1,1) 'b1

hspiin (data1) 'b0

high csn

b1 = 0

let bit8 = bit1

let bit9 = bit2

let bit10 = bit3

sertxd ("Data in Pipe\_", #b1,cr,lf)

low ce 'Receiver in standby

low csn

hspiout (read\_rx\_payload) ' Read Data in RX FIFO

hspiin (b10,b11,b12,b13,b14,b15)

high csn

SERTXD ("Data Received " ,#b11," ",#b12," ",#b13," ",#b14," ",#B15, CR,LF,cr,lf)

**gosub clear\_interrupts**

setint %00000000,%00100000,c 'reset Picaxe interrupt

return

}

**Clear\_interrupts:**

{

'==========================================

'This function clears all RF2400 interrupts

'by clearing bits 4, 5, & 6 of register 7.

' while leaving the other bits unchanged &

'intact

'==========================================

addr = status

low csn

hspiout (addr,1,1)

hspiin (data1) 'Read Status Register into B0 (data\_1)

high csn

let bit4 = 1 'Modify B0

let bit5 = 1

let bit6 = 1

pause 200 'leave LED on a while

addr = status + 32 'write

low csn

hspiout (addr,data1) 'Write New Value of B0 to status register)

high csn

return

} 'Clear RF 2400 Interrupts

**get\_status:**

{

data1 = 0

addr = status

low csn

hspiout (addr,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Status = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

'==========================

;if data1 = 64 then let pipe = 0 endif

;if data1 = 66 then let pipe = 1 endif

;if data1 = 68 then let pipe = 2 endif

;if data1 = 70 then let pipe = 3 endif

;if data1 = 72 then let pipe = 4 endif

;if data1 = 74 then let pipe = 5 endif

return

} 'Read RF2400 status register and sertxd out

**get\_Config:**

{

addr = Config

low csn

hspiout (addr,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Config = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

} 'Read RF2400 config register and sertxd out

**power\_on:**

{

'==============================

addr = config

low csn

hspiout (addr,1,1)

hspiin (data1)

high csn

let bit1 = 1

addr = config + 32 'write

low csn

hspiout (addr,data1)

high csn

return

'=============================

} 'Turn RF2400 Power On

**power\_off:**

{

addr = config

low csn

hspiout (addr,1,1)

hspiin (data1) '(B0)

high csn

let bit1 = 0

addr = config + 32

low csn

hspiout (addr,data1)

high csn

return

} 'Turn RF2400 Power Off

**PRX\_ON:**

{

addr = config

low csn

hspiout (addr,1)

hspiin (data1)

high csn

let bit0 = 1

addr = config + 32 'write

low csn

hspiout (addr,data1)

high csn

return

} 'Enable RF2400 Receive mode

**PTX\_ON:**

{

addr = config

low csn

hspiout (addr,1)

hspiin (data1)

high csn

let bit0 = 0

addr = config + 32 'write

low csn

hspiout (addr,data1)

high csn

return

} 'Enable RF2400 Tramsmit TX mode

**read\_all:**

{

'=========================

sertxd (CR,LF,"Register values are in decimal format",CR,LF)

for b1 = 0 to 09 'registers 0 - 9 (decimal)

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

next b1

for b1 = 10 to 16 'Addresses with 5 byte registers

low csn

hspiout (b1,1,2,3,4,5)

hspiin (b4,b5,b6,b7,b8)

high csn

sertxd ("Register ",#b1," = ", #b4,#b5,#b6,#b7,#b8,CR,LF)

next b1

for b1 = 17 to 22

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

sertxd ("Register ",#b1," = ", #b0,cr,lf)

next b1

low csn

b1 = 23

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

low CSN

b1 = 28

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

inc b1

low CSN

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

return

} 'Read all RF2400 all registers and sertxd out

**serial\_out1:**

{

'======================serial\_out1:

sertxd ("Register ",#b1," = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

Return

} 'Used with read\_all



; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*\*\*\*\* **INHAOS 2400 FUNCTIONS** Test \*\*\*\*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Filename: **RF\_2400\_FUNCT.bas**

; Date: 08-02-2011

; File Version: Beta Test

; Written by: Bill Roth / Goeytex

; Target PICAXE: 20X2

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' \* Open Program Editor

' \* Options > Editor > Use Collapsing Blocks \*

' \* Options > Editor . Serial Terminal . Open Ater Download

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'Directives

#picaxe 20x2

#no\_data

#no\_table

#Com 1

**symbols:**

{

symbol write\_tx = %10100000 'write payload to buffer

symbol write\_tx\_no\_ack = %10110000 'write payload to buffer no ACK

 'must enable with Activate

symbol write\_tx\_addr = 48 'Register 16(DEC) + 32

symbol read\_rx\_payload = %01100001 ' Command Byte - see datasheet

symbol data1 = b0 ' bit0 - bit7

symbol data2 = b1 ' bit8 - bit15

symbol data3 = b2 ' future use bit16 - bit23

symbol data4 = b3 ' future use bit24 - bit31

symbol reg\_addr = b4 ' register address

symbol PRX = 1

symbol PTX = 0

symbol mode = b5 'PRX = 0 PTX = 1

symbol pld\_width = b6 'Received payload width in # of bytes

 'Used when Dynamic Payload is enabled

symbol rf\_int = b7

symbol ce = b.1 'Chip Enable

symbol csn = b.4 'Chip Select Active low)

}

**init:**

{

setfreq m32

hspisetup spimode00,spifast

high CSN 'active low

input b.0

output b.1

output b.4

low CE

**gosub clear\_interrupts**

setint %00000000,%00100000,c 'Picaxe 20x2 Interrupt Pin5 Low

} 'Configure Picaxe

**config\_registers:**

{

#rem

To write to a 2400 register the syntax is:

 (byteA, byteB)

Where byteA is the command byte and byteB contains the data bits to be written.

ByteA must have bit 5 set(1) to enable writing. Bits 0-4 are the

5 bit register address to write to..

 001 00000

 \_\_\_ \_\_\_\_\_ = write to address 0

 WRT ADDR

#endrem

**gosub power\_off**

pause 2

'write register 0

'enable PtX, power off, en\_crc/2bytes, interrupts unmasked

low csn

hspiout (%00100000,%00001100) 'EN PTX

high csn

;write register 1

low csn

hspiout (%00100001,%00000000) ;Shockburst OFF ALL Pipes (no ACK)

high csn

;write register2

low csn

hspiout (%00100010,%00000011) ' Enable pipes 1 & 2

high csn

pause 1

;write register 3

low csn

hspiout (%00100011,%00000001) ' RX Address width = 3 bytes wide

high csn

;write register 4

low csn

hspiout (%00100100,%00010000) ' ARC Disbled/ ARDelay = 500u

high csn ' See Datasheet

'write register 5

low csn

hspiout (%00100101,%00000010) ' RF channel = 2

high csn ' Receiver must match

'write Register 6

low csn

hspiout (%00100110,%00110101) 'REG 6 Data Rate 1 mbps

high csn

'write Regiters 17 - 22 'Payload Width

for b0 = 49 to 54 'Reg Address + 32 decimal to Write

low csn 'Could also do this in hex

hspiout (b0,1) 'Pipe\_0 - Pipe\_5 Payload Width = 5

high csn

next b0

} 'Configure RF2400 registers

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PROGRAM LOOP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**main:**

{

'==========================

**Gosub power\_on**

**call show\_all**

**call get\_int**

'===========================

**do**

 pause 5000

 low csn

 hspiout (write\_tx,55) 'write 1 byte to tx fifo

 high csn

 pulsout ce,100 ; RF TRANSMIT

 'now it automatically goes to interrupt (TX DS)

**loop**

pause 2000

**call prx\_on**

**do**

high CE 'receiver on

**loop**

**call ptx\_on**

}

**interrupt:**

{

**call get\_int**

#rem

'Note: This is where the program always goes when

 1. A data packet is successfully transmittted

 2. A data packet is received

 3. Error on Max Retries (Auto ACK enabled)

 Here is where decision made as to what to do based upon

 interrupt condition.

 Interrupt condition can be determined by reading interrupt bits

 status register. See function "get\_int" for example code

 I have instead passed a mode variable set in either PTX\_ON or PTX\_ON

 functions telling if the module was in TX mode or RX mode.

 #endrem

if mode = PRX then

 sertxd ("PRX DR interrupt",cr,lf) 'for debugging

 low ce ' standby

 low csn

 hspiout (read\_rx\_payload) ' GET PAYLOAD FROM RX FIFO

 hspiin (data2) 'Put it into data2

 high csn

 sertxd (#data2,cr,lf) ' send it terminal

endif

low csn

**gosub clear\_interrupts**

high csn

setint %00000000,%00100000,c

return

}

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PROGRAM LOOP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'\*\*\*\*\*\*\*\*\*\*\*\*"FUNCTIONS" START HERE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**get\_int:**

{

**gosub read\_status** 'into b0 > bits 0 - 7

if bit4 = 1 then

rf\_int = 1

serrxd ("Interrupt on MAX\_RT",cr,lf)

return endif

if bit5 = 1 then

rf\_int = 2

sertxd("Interrupt on TX\_DS",cr,lf)

return endif

if bit6 = 1 then

rf\_int = 3

sertxd ("Interrupt on RX\_DR",cr,lf)

return endif

rf\_int = 0

sertxd ("There Is No RF2400 Interrupt Condition",cr,lf)

return

}

**read\_rx\_data:**

{

low csn

hspiout (read\_rx\_payload) ' Read Data in RX FIFO

hspiin (b16)

high csn

return

} 'Into var b16

**prx\_on:**

{

low csn

hspiout (0,1)

hspiin (data1)

high csn

let bit0 = 1

low csn

hspiout (32,data1)

high csn

mode = PRX ' Passed back to calling routine / function

return

} 'Enable RF2400 Receive (PRX) mode

**ptx\_on:**

{

low csn

hspiout (0,1)

hspiin (data1)

high csn

let bit0 = 0

low csn

hspiout (32,data1)

high csn

mode = PTX 'passed back to calling routine / function

return

} 'Enable RF2400 Tramsmit (PTX) mode

**power\_on:**

{

low csn

hspiout (0,1,1)

hspiin (data1)

high csn

let bit1 = 1

low csn

hspiout (32,data1)

high csn

return

'=============================

} 'Turn RF2400 Power On

**power\_off:**

{

low csn

hspiout (0,1,1)

hspiin (data1) '(B0)

high csn

let bit1 = 0

low csn

hspiout (32,data1)

high csn

return

} 'Turn RF2400 Power Off

**clear\_interrupts:**

{

'==========================================

'This function clears all RF2400 interrupts

'by clearing bits 4, 5, & 6 of register 7.

' while leaving the other bits unchanged

'==========================================

low csn

hspiout (7,1,1)

hspiin (data1) 'Read Status Register into B0 (data\_1)

high csn

let bit4 = 1 'Modify B0

let bit5 = 1

let bit6 = 1

'pause 10 'leave LED on a whiie

low csn

hspiout (39,data1) 'Write New Value of B0 to status register)

high csn

return

} 'Clear All RF 2400 Interrupts

**activate:**

{

'0nly used when tx\_noack command and. or Dynpd is enabled

'This must be run first to give R/W access to feature

'& DYNPD registers. R\_RX\_PL\_WID, W\_ACK\_PAYLOAD, and W\_TX\_PAYLOAD\_NOACK

low csn

hspiout ($1D,1,1)

hspiin (b0)

high CSN

if bit0 = 0 then ;this prevents activate from toggling on reset

 low csn

 hspiout (%01010000,$73)

 high csn

 'Reg 29 "Feature Register"

 low csn

 hspiout (%00111101,%00000101) ;enable no\_ack & DynPL

 high csn

 low csn

 hspiout (%00111100,%00000001) ;enable no\_ack command

 high csn

 endif

return

} 'Open this block to see description

**read\_status:**

{

low csn

hspiout (7,1,1)

hspiin (data1)

high csn

return

} 'Read contents of status register into B0

**read\_config:**

{

low csn

hspiout (0,1,1) 'reg\_addr

hspiin (data1) 'b0

high csn

return

} 'Read cintents of config register into B0

**show\_status:**

{

low csn

hspiout (7,1,1)

hspiin (data1)

high csn

sertxd ("Status = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

} 'Read contents of status register and send out via sertxd

**show\_config:**

{

low csn

hspiout (0,1,1)

hspiin (data1)

high csn

sertxd ("Config = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

} 'Read contents of config register send out via sertxd

**show\_all:**

{

sertxd (CR,LF,"Register Values Are In Decimal Format",CR,LF)

for reg\_addr = 0 to 09 'registers 0 - 9 (decimal)

low CSN

hspiout (reg\_addr,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

next reg\_addr

for reg\_addr = 10 to 16 'Addresses with 5 byte registers

low csn

hspiout (reg\_addr,1,2,3,4,5)

hspiin (b7,b8,b9,b10,b11)

high csn

sertxd ("Register ",#reg\_addr," = ", #b7,#b8,#b9,#b10,#b11,CR,LF)

next reg\_addr

for reg\_addr = 17 to 22

low CSN

hspiout (reg\_addr,1,1)

hspiin (b0)

high CSN

sertxd ("Register ",#reg\_addr," = ", #b0,cr,lf)

next reg\_addr

low csn

reg\_addr = 23

low CSN

hspiout (reg\_addr,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

low CSN

reg\_addr = 28

hspiout (reg\_addr,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

reg\_addr = 29

low CSN

hspiout (reg\_addr,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

return

} 'Read all RF2400 all registers and send out

**serial\_out1:**

{

'======================serial\_out1:

sertxd ("Register ",#reg\_addr," = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

Return

} 'Used with Show All

**HELP:**

{

#rem

1.To read a 2400 register

 a. Take CSN low

 b. hspout ( register address,byte,byte,byte)

 where "byte" is a only placholder. The number

 placeholders determines the number of bytes out

 c. hspiin (var, var, var) where the number of variables

 equals the number of bytes expected out

 d. Take CSN High

 Example to read the status register

 low csn

 hspiout (7,1) ' read register 7 - 1 byte out

 hspiin (b0) ' put contents of regieter into b0

 high csn

 if register 7 was a 5 byte register (it isnt) it would be:

 low csn

 hspiout (7,1,1,1,1,1) or even hspiout (7,0,0,0,0,0)

 hspiin (b0,reg\_addr,b2,b3,b4,) ' put contents of register into b0 - b4

 high csn

 2. To write to a register;

 \*\*\* A write is performed by adding 32 to the register address \*\*\*

 a. Take CSN low

 b. hspiout ( 32 + register address ,data)

 Where data is the bit values to write.

 This is better done using binary format for readability)

 c, take csn high

 Example to wrtie to the config register (Register 0)

 low csn

 hspiout (32,%01111101) 'WRITE to Address 0 (32 + 0)

 high csn 'set bit 0 to 1

 'set bit 1 to 0

 'set bit 2 to 1

 'etc

 This would:

 bit 0 = 1 . Enable Primary PRX mode

 bit 1 = 0 Power off

 bit 2 = 1 Set 2 byte CRC encoding

 bit 3 = 1 Enable CRC

 bit 4 = 1 enable max\_rt interrupt (Similar to "setint" on a Picaxe)

 bit 5 = 1 enable TX\_DataSent interrupt

 bit 6 = 1 enable RX\_DataReceived interrupt

 bit 7 = 0 only 0 allowed in bit 7

3. Whenever a read or write is done the 2400 ALWAYS sends out the

 contents of the status register as the last byte in a read or

 in parallel with a write.

 Example, we want to write to register 1

 low csn

 hspiout (33,%11010101)

 hspiin (b0)

 high csn

 B0 will now contain the content of Status register

 This allows the testing of a value if needed, for example;

 If bit 0 = 1 then goto wherever

 #endrem

 } 'Read this

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' **nRF24L01+ Test Code TRANSMIT**

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'GO TO "OPTIONS" > "EDITOR" AND check SERIAL TERMINAL

' TO "OPEN AFTER DOWNLOAD" and check "Use Colapsing Blocks"

'===================================================

 'THIS IS CODE FOR THE TRANSMITTER

#picaxe 20x2

#no\_data

#no\_table

**Symbols:**

{

symbol config = 0 symbol en\_aa = 1

symbol en\_rxaddr = 2 symbol setup\_aw = 3

symbol setup\_retr = 4 symbol rf\_ch = 5

symbol rf\_setup = 6 symbol status = 7

symbol observe\_tx = 8 symbol rpd = 9

symbol rx\_addr\_p0 = 10 symbol rx\_addr\_p1 = 11

symbol rx\_addr\_p2 = 12 symbol rx\_addr\_p3 = 13

symbol rx\_addr\_p4 = 14 symbol rx\_addr\_p5 = 15

symbol tx\_adddr = 16 symbol rx\_pw\_p0 = 17

symbol rx\_pw\_p1 = 18 symbol rx\_pw\_p2 = 19

symbol rx\_pw\_p3 = 20 symbol rx\_pw\_p4 = 21

symbol rx\_pw\_p5 = 22 symbol fifo\_status = 23

symbol dynpd = 28 symbol feature = 29

symbol data1 = b0

symbol nRF\_Address = b1

symbol IRQ = pinb.0 'hardware IRQ1

symbol CE = B.1 'Chip Enable

symbol CSN = B.6 'Chip Select Active low)

}

**init:**

{

input b.0

output b.1

output b.6

setfreq m16

hspisetup spimode00,spislow

high CSN 'active low

low CE

**gosub clear\_interrupts**

}

'NOTE: Shockburst is not used. NO ACK

'The PTX reads the ADC and then sends out

'that data byte every few seconds

'Adddress 00

'enable PtX, power on, en\_crc/2bytes, all interrputs on

low csn

hspiout (%00100000,%00001110)

high csn

low csn

hspiout (%00100110,%00000110) 'REG 6 Data Rate 1 mbps

high csn

**MAIN:**

'========================================================

setint %00000000,%00100000,c '("20x2 Pin5 LOW)

**do**

PAUSE 20000

Readadc 3,B10

sertxd (#B10,CR,LF)

low csn 'FLUSH TX FIFO redundant

hspiout (%11100001)

high csn

low CSN

hspiout (%10100000,B10) 'write 1 byte to tx fifo

high csn

low csn

hspiout (%00100000,%00001110) 'POWER UP

high csn

PAUSE 5

pulsout CE,220 'TRANSMIT DATA

pause 5

low csn 'FLUSH TX FIFO Redundant

hspiout (%11100001)

high csn

**loop**

'=============== END MAIN =================================

**interrupt:**

{

'the nRF has generated an interupt

;pause 500

;LOW CSN

;hspiin (b0)

;HSPIOUT (%11111111)

;HIGH CSN

'status data is now in b0 (data1) bits 4,5, & 6

 'Bit 4 is Max\_RT

 'Bit 5 is TX\_DS (Data Sent

 ;bit 6 is RX\_DR (data received)

;if bit4 = 1 then

;sertxd ("Interrupt on MAX\_RT",cr,lf)

;else

;if bit5 = 1 then

;sertxd ("Interrupt on TX\_DS",cr,lf)

;else

;if bit6 = 1 then

;sertxd ("Interrupt on RX\_DR",cr,lf)

;endif endif endif

**gosub clear\_interrupts**

setint %00000000,%00100000,c

return

}

**Clear\_interrupts:**

{

nRF\_address = status + 32

low csn

hspiout (nRF\_address,%01110000)

high csn

return

}

**get\_status:**

{

nRF\_Address = status

low csn

hspiout (nRF\_Address,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Status = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

}

**get\_Config:**

{

nRF\_Address = Config

low csn

hspiout (nRF\_Address,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Config = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

}

**power\_up:**

{

'==============================

nRF\_Address = config

low csn

hspiout (nRF\_Address,1,1)

hspiin (data1)

high csn

let bit1 = 1

nRF\_Address = config + 32 'write

low csn

hspiout (nRF\_Address,data1)

high csn

return

'=============================

}

power\_down:

{

nRF\_Address = config

low csn

hspiout (nRF\_Address,1,1)

hspiin (data1) '(B0)

high csn

let bit1 = 0

nRF\_Address = config + 32

low csn

hspiout (nRF\_Address,data1)

high csn

return

}

**PRX\_ON:**

{

nRF\_Address = config

low csn

hspiout (nRF\_Address,1)

hspiin (data1)

high csn

let bit0 = 1

nRF\_Address = config + 32 'write

low csn

hspiout (nRF\_Address,data1)

high csn

return

}

**PTX\_ON:**

{

nRF\_Address = config

low csn

hspiout (nRF\_Address,1)

hspiin (data1)

high csn

let bit0 = 0

nRF\_Address = config + 32 'write

low csn

hspiout (nRF\_Address,data1)

high csn

return

}

**read\_all:**

{

'=========================

sertxd (CR,LF,"Register values are in decimal format",CR,LF)

for b1 = 0 to 09 'registers 0 - 9 (decimal)

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

gosub serial\_out1

next b1

for b1 = 10 to 16 'Addresses with 5 byte registers

low csn

hspiout (b1,1,2,3,4,5)

hspiin (b4,b5,b6,b7,b8)

high csn

sertxd ("Register ",#b1," = ", #b4,#b5,#b6,#b7,#b8,CR,LF)

next b1

for b1 = 17 to 22

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

sertxd ("Register ",#b1," = ", #b0,cr,lf)

next b1

low csn

b1 = 23

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

low CSN

b1 = 28

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

inc b1

low CSN

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

return

} 'READ ALL REGISTERS AND SEND THEM OUT ON SERRXD

**serial\_out1:**

{

'======================serial\_out1:

sertxd ("Register ",#b1," = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

Return

}

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' **nRF24L01+ Test Code Receiver**

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'GO TO "OPTIONS" > "EDITOR" AND check SERIAL TERMINAL

' TO "OPEN AFTER DOWNLOAD" and check "use colapsing blocks"

'===================================================

 'ENHANCED SHOCBURST NOT USED

 'THIS IS CODE FOR THE RECEVEIVER

#picaxe 20x2

#no\_table

#no\_data

Symbols:

{

symbol CE = C.2 'Chip Enable

symbol CSN = C.0 'Chip Select Active low)

symbol config = 0 symbol en\_aa = 1

symbol en\_rxaddr = 2 symbol setup\_aw = 3

symbol setup\_retr = 4 symbol rf\_ch = 5

symbol rf\_setup = 6 symbol status = 7

symbol observe\_tx = 8 symbol rpd = 9

symbol rx\_addr\_p0 = 10 symbol rx\_addr\_p1 = 11

symbol rx\_addr\_p2 = 12 symbol rx\_addr\_p3 = 13

symbol rx\_addr\_p4 = 14 symbol rx\_addr\_p5 = 15

symbol tx\_adddr = 16 symbol rx\_pw\_p0 = 17

symbol rx\_pw\_p1 = 18 symbol rx\_pw\_p2 = 19

symbol rx\_pw\_p3 = 20 symbol rx\_pw\_p4 = 21

symbol rx\_pw\_p5 = 22 symbol fifo\_status = 23

symbol dynpd = 28 symbol feature = 29

symbol data1 = b0

symbol nRF\_Address = b1

symbol IRQ = pinb.0 'hardware IRQ1

}

**init:**

{

input b.0

output b.1

output b.6

setfreq m16

hspisetup spimode00,spislow

high CSN 'active low

low CE

**gosub clear\_interrupts**

}

**CONFIG\_RX\_MODE:**

{

low csn

hspiout (%00100000,%00001111) 'Write to config register' EN PRX + Power ON

high csn

low csn

hspiout (%00100110,%00000110) ' REG 6 (Data Rate = 1 mbps)

high csn

low csn

hspiout (%00110001,1) ' reg 17 (hex 11) payload = 1 bytes for data pipe 0

high csn

}

**MAIN:**

setint %00000000,%00100000,c '("20x2 Pin5 LOW)

**do**

 high CE

 pause 10 'waiting for interrupt on data received

 high CE

 pause 10

**loop**

**interrupt:**

'The nRF has received data and has generated an interupt on RX\_DR

pause 3

low ce 'Receiver in standby

low csn

hspiout (%01100001) ' Read Data in RX FIFO

hspiin (b10)

high csn

SERTXD (#B10,CR,LF)

low csn

hspiout (%11100001) 'FLUSH RX FIFO

high csn

gosub clear\_interrupts

setint %00000000,%00100000,c 'reset Picaxe interrupt

return

**Clear\_interrupts:**

{

nRF\_address = status + 32

low csn

hspiout (%00100111,%01110000)

high csn

return

}

**get\_status:**

{

nRF\_Address = status

low csn

hspiout (nRF\_Address,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Status = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

}

**get\_Config:**

{

nRF\_Address = Config

low csn

hspiout (nRF\_Address,1,1) 'b1

hspiin (data1) 'b0

high csn

sertxd ("Config = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

return

}

**power\_up:**

{

'==============================

nRF\_Address = config

low csn

hspiout (nRF\_Address,1,1)

hspiin (data1)

high csn

let bit1 = 1

nRF\_Address = config + 32 'write

low csn

hspiout (nRF\_Address,data1)

high csn

return

'=============================

}

**power\_down:**

{

nRF\_Address = config

low csn

hspiout (nRF\_Address,1,1)

hspiin (data1) '(B0)

high csn

let bit1 = 0

nRF\_Address = config + 32

low csn

hspiout (nRF\_Address,data1)

high csn

return

}

**PRX\_ON:**

{

nRF\_Address = config

low csn

hspiout (nRF\_Address,1)

hspiin (data1)

high csn

let bit0 = 1

nRF\_Address = config + 32 'write

low csn

hspiout (nRF\_Address,data1)

high csn

return

}

**PTX\_ON:**

{

nRF\_Address = config

low csn

hspiout (nRF\_Address,1)

hspiin (data1)

high csn

let bit0 = 0

nRF\_Address = config + 32 'write

low csn

hspiout (nRF\_Address,data1)

high csn

return

}

**read\_all:**

{

'=========================

sertxd (CR,LF,"Register values are in decimal format",CR,LF)

for b1 = 0 to 09 'registers 0 - 9 (decimal)

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

next b1

for b1 = 10 to 16 'Addresses with 5 byte registers

low csn

hspiout (b1,1,2,3,4,5)

hspiin (b4,b5,b6,b7,b8)

high csn

sertxd ("Register ",#b1," = ", #b4,#b5,#b6,#b7,#b8,CR,LF)

next b1

for b1 = 17 to 22

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

sertxd ("Register ",#b1," = ", #b0,cr,lf)

next b1

low csn

b1 = 23

low CSN

hspiout (b1,1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

low CSN

b1 = 28

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

inc b1

low CSN

hspiout (b1,1)

hspiin (b0)

high CSN

**gosub serial\_out1**

return

} 'READ ALL REGISTERS AND SEND THEM OUT ON SERRXD

**serial\_out1:**

{

'======================serial\_out1:

sertxd ("Register ",#b1," = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

Return

}



; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*\*\*\*\* **INHAOS RF2400 Transmit / Receive** \*\*\*\*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Filename: **RF2400\_PICAXE\_DEMO.bas**

; Written by: William Roth

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DIRECTIVES:**

 #picaxe 20x2

 #no\_data

 #no\_table

 #com 4 '// Picaxe on Com4 (AXE027) Change this line as needed

**SYMBOLS:**

 symbol Config = $00 'Reg\_address of Config Register

 symbol Status = $07 'Reg\_address of Status Register

 symbol Write\_Tx\_Fifo = $A0 'Command Byte (%10100000) See Datasheet

 symbol Read\_Rx\_Payload = $61 'Commmnd Byte (%01100001) See datasheet

 symbol ACTIVATE = $50

 symbol NOP = 255

 symbol CE = C.4 '// Chip Enable

 symbol CSN = C.3 '// Chip Select/Active low)

 symbol RX\_LED = B.1

 symbol TX\_LED = B.2

 symbol data1 = b0 '// Dynamic Variables

 symbol reg\_addr = b1

 symbol data2 = B55

 symbol PKT\_ID = B14

**INIT\_PICAXE:**

 setfreq m16 '// Freq = 16 Mhz

 hspisetup spimode00,spifast

 high CSN

 low CE

 **gosub clear\_interrupts**

 setint %00000000,%00100000,c '// Picaxe 20x2 Interrupt Pin5 Low

**INIT\_RF2400:**

 **gosub config\_bank0**

 **gosub config\_bank1** '//Unremark for RF2400PA

**MAIN:**

'//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'// \*

'//goto Receiver '// SELECT TX OR RX \*

**goto Transmitter** '// \*

'// \*

'//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**RECEIVER:**

'//=====================================

 **gosub prx\_on** '// Set RX ON

 **gosub power\_on** '// Turn RF2400 Power On

 **do**

 pause 100

 high CE '// Receiver enabled

 pause 100 '// Wait for interrupt

 **loop**

'//============== END RECEIVER =============

**TRANSMITTER:**

'//=========================================

 **gosub ptx\_on** '// Set TX ON

 **gosub power\_on** '// Turn RF2400 Power On

 PKT\_ID = 0

 **do**

 pause 5000

 low csn

 hspiout (write\_tx\_fifo,55,241,77,83,PKT\_ID) '// Write 5 bytes to tx fifo

 high csn

 pulsout CE,10 '// TRANSMIT ! (~30us pulse)

 pause 500 '// Will interrupt during this pause

 inc PKT\_ID

**loop**

'//======================= END TRANSMITTER =================================

'//\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sub Routines Begin Here \*\*\*\*\*\*\*\*\*\*\*\*\*

**Config\_Bank1:**

'================================================

'// ONLY USED WITH RF2400PA 20BM Module

'// There is no need to write or modify any Bank1 Registers with RF2400P or 2400W

'// NOTE:

 '// See RF2400PA Datasheet for these settings.

 '// Datasheet offers no explanation to what these

 '// settings actually do.

 '// These registers are write only. Therefore there

 '// is no way to tell if write was successful!

 '// BEKIN WEBSITE offers no support on Bank 1

 '// Registers

 **gosub read\_status**

 if bit7 = 0 then '// Toggle Active Bank if needed

 low csn

 hspiout (ACTIVATE,$53)

 high csn

 endif

reg\_addr = $03

setbit reg\_addr,5 '//Set write bit

low csn

hspiout (reg\_addr,$F9,$00,$39,$41) '// See RF2400PA DATASHEET

high csn

reg\_addr = $04

setbit reg\_addr,5 '//Set write bit)

low csn

hspiout (reg\_addr,$C1,$9E,$9A,$0B) '//See RF2400PA DATASHEET

high csn

**gosub read\_status**

 if bit7 = 1 then '// Toggle Bank0 to Active

 low csn

 hspiout (ACTIVATE,$53)

 high csn

 endif

reg\_addr = $06 '// Set RF Power to -5dBm - RF2400PA Only

data1 = %00000111

**gosub write\_register**

RETURN

'=============== END SUB =======================

**Config\_Bank0:**

'//=========================================

 **gosub read\_status**

 if bit7 = 1 then '// Toggle Active Bank if needed

 low csn

 hspiout (ACTIVATE,$53)

 high csn

 endif

 reg\_addr = $00 : data1 = %00001100 : **gosub Write\_register** '// CONFIG

 reg\_addr = $01 : data1 = %00000000 : **gosub Write\_register** '// EN\_AA (OFF)

 reg\_addr = $02 : data1 = %00000011 : **gosub Write\_register** '// EN\_RXReg\_addr

 reg\_addr = $03 : data1 = %00000001 : **gosub Write\_register** '// SETUP\_AW

 reg\_addr = $04 : data1 = %00010000 : **gosub Write\_register** '// SETUP\_RETR

 reg\_addr = $05 : data1 = %00000010 : **gosub Write\_register** '// RF Channel 2

 reg\_addr = $06 : data1 = %00000111 : **gosub Write\_register** '// RF POWER 5dBm

 for reg\_addr = $11 to $16 '// Payload width = 5 bytes all registers

 data1 = 5

 **gosub write\_register**

 next

 '// No DYNPD

 '// No Feature

 '// No AUTO ACK

 return

'//======================= END SUB ================================

**Interrupt:**

'//=============================================================

'//Program jumps here when Picaxe Detects interrupt on Pinc C.5

 pause 100

 **gosub read\_status** '// Determine source of interrupt

 if bit6 = 1 then '// RX\_DR ...Valid Packet Received

 pulsout RX\_LED, 3000 '// Blink LED on Pin B.1

 sertxd ("Data Received ")

 low ce '// Receiver in standby

 data2 = data1 & %00001111

 data2 = data2 >> 1

 sertxd ("in Pipe\_", #Data2,cr,lf)

 low csn

 hspiout (read\_rx\_payload) '// Read Data in RX FIFO

 hspiin (b10,b11,b12,b13,PKT\_ID)

 high csn

 SERTXD ("Data Received ",#b10," ",#b11, \_

 " ",#b12," ",#b13," ",#PKT\_ID,cr,lf)

 SERTXD ("Packet ID = ",#PKT\_ID,cr,lf,cr,lf)

 endif

 if bit5 = 1 then '// TX\_DS ...VALID PACKET SENT

 pulsout TX\_LED, 3000 '// Blink LED on Pin B.2

 sertxd ("Data Packet Sent",cr,lf)

 sertxd ("Packet ID = ",#PKT\_ID,cr,lf,cr,lf)

 endif

 **gosub clear\_interrupts** '// Clear BK2421 Interrupts

 setint %00000000,%00100000,c '// Reset Picaxe Interrupt

 return '// Returns to either Receiver or Transmiter

'//=========================== END SUB ===================================

**Read\_Register:**

'//=================================

 low csn

 hspiout (Reg\_addr)

 hspiin (data1)

 high csn

 return

'//============== END SUB ==========

**Write\_Register:**

'//==================================

 setbit Reg\_Addr,5 '// set write bit

 low csn

 hspiout (Reg\_Addr,data1)

 high csn

 clearbit Reg\_Addr,5

 return

'//========= END SUB ===============

**Clear\_Interrupts:**

'==========================================

'This function clears all BK2421 interrupts

'by clearing bits 4, 5, & 6 of register 7.

 **gosub read\_status**

 data1 = data1 OR %01110000

 Reg\_addr = status

 **gosub write\_register**

 return

'//========== END SUB =======================

**Read\_Status:**

'//======================

 low csn

 hspiout (NOP)

 hspiin (data1)

 high csn

 return

'//======= END SUB =====

**Disp\_Status:**

'//===============================================

 **gosub read\_status**

 sertxd ("Status = ", #bit7,#bit6,#bit5,#bit4,#bit3, \_

 #bit2,#bit1,#bit0,CR,LF)

 return

'//================= END SUB =====================

**Power\_On:**

'//==========================

 reg\_addr = config

 **gosub read\_register**

 setbit data1,1

 **gosub write\_register**

 return

'//======== END SUB ========

**Power\_Off:**

'//==========================

 reg\_addr = config

 **gosub read\_register**

 clearbit data1,1

 **gosub write\_register**

 return

'//======== END SUB =========

**Prx\_On:**

'//============================

 reg\_addr = config

 **gosub read\_register**

 setbit data1,0

 **gosub write\_register**

 return

'//======== END SUB ===========

**Ptx\_On:**

'//============================

 reg\_addr = config

 **gosub read\_register**

 clearbit data1,0

 **gosub write\_register**

 return

'//======= END SUB ===========

 **Disp\_register:**

 '//==============================

 low csn

 hspiout (reg\_addr)

 hspiin (data1)

 high csn

 sertxd ("Register ",#Reg\_Addr," = ", #bit7,#bit6,#bit5,#bit4,#bit3,#bit2,#bit1,#bit0,CR,LF)

 return

'//=========== END SUB ============

'//END OF PROGRAM

**Polices à chasse fixe :**

Courrier New

Lucida Console

<http://fr.fontriver.com/basic/fixed_width/>