MRF24J40MA Lib descriptions

Module constants:

All constants defined up to now is in the module MRF24J40_Const.mbas.

Global module var's:

headerlenght	*as byte external* *the length of packet header sent.*
DataLenght	*as byte external* *This var is used to control the length of TXBUFFER*
Src_seq_Number	*as byte external* *This var starts seq_number for packets.*
P2PSTATUS	*as word external* *the P2PSTATUS is used by program for flag some states of module and stack* *the P2PSTATUS REG not ended up to now.* *Only bit's 0-3 of P2PSTATUS already defined.*
TX_BUFFER	*as byte[5..127] external* *data of TXn Buffer.* *The minimum recommended to this buffer is 5 bytes.* *But in this lib compilation the minium is 15.*
RX_BUFFER	*as byte[5..127] external* *data received in RXBUFFER.* *The minimum recommended to this buffer is 5 bytes* *But in this lib compilation the minium is 15.*
this_MAC	*as Byte[8] external* *the long MAC of module*
this_PANID	*as byte[2] external* *the PANID of module*

Global module bit's:

_RXHWIE	*as sbit sfr external* *Flag to enable/disable external interrupt of MCU*
_RXHWIF	*as sbit sfr external* *Flag that indicate one event in the external interrupt of MCU. When a packet is received the MRF put this pin down.*
WAKE	*as sbit sfr external* *Used by MCU to restore the MRF from sleeping.*
RST	*as sbit sfr external* *Used for resets the MRF module(hardware reset)*

How we can declare PORT TRIS in the declaration section.

```
TRIS direction
      dim MRF_RX_RF_INT_DIRECTION as sbit at TRIS.bit
      dim MRF_CSDIRECTION         as sbit at TRIS.bit
      dim MRF_RSTDIRECTION        as sbit at TRIS.bit
      dim MRF_WAKEDIRECTION       as sbit at TRIS.bit

PORT settings
      dim MRF_RX_RF_INT           as sbit at PORT.bit
      dim MRF_CS                  as sbit at LAT.bit
      dim MRF_WAKE                as sbit at LAT.bit
      dim MRF_RST                 as sbit at LAT.bit
```

Procedures:

sub procedure MRF_Ena_RX_Packet()
Description:
Enable module to receive packet

sub procedure MRF_Disa_RX_Packet()
Description:
Disable module to receive packet

sub procedure MRF_DiscardPacket()
Description:
Clear RX FIFO Buffer.

sub procedure MRF_SetTurboMode()
Description:
Set's the module to work at high speed
625kbs if more band is required.

sub procedure MRF_ClearINT()
Description:
clear module interrupt register

sub procedure MRF_SetIndirectMessage()
Description:
Set's the module to send a indirect message

sub procedure MRF_FillData(dim txbufdata as byte)
Description:
Fill the array TX_BUFFER byte by byte and set correct
value to the "DataLength" var.

Param txbufdata: byte
Var, const or literal that represents the value to be
placed in the TX_BUFFER.

sub procedure MRF_init(dim byref PAN as byte[2])
Description:
Initializes module with PANID

Pre condition:
Module need to be configured with MRF_HW_Config() before issue this
call.

sub procedure MRF_HW_Config()
Description:
Configures the MCU PORT,PIN's to be used with module.
This is the first call for initializes the module.

sub procedure MRF_RequestConnection(dim cmd, channel, connection_size as byte)

Description:
Prepares the packet to send a requisition of connection to the other module

Param cmd: byte
var, const or literal that describes the request connection in P2P mode.

Param channel: byte
var, const or literal that inform who channel we want start One Connection.

Param connection_size: byte
var, const or literal.

sub procedure MRF_WrtLngAddress(dim addr as word, dim ddata as byte)

Description:
Procedure to write any value to any valid long RAM address.

Param addr: word
var, const or literal valid to that RAM position.

Param ddata: byte
var, const or literal to be write at this RAM position.

sub procedure MRF_WrtShortAddress(dim addr, ddata as byte)

Description:
Procedure to write any value to any valid short RAM address.

Param addr: byte
var, const or literal valid to that RAM position.

Param ddata: byte
var, const or literal to be write at this RAM position.

sub procedure MRF_CreateNewConnection_WithPaylod(dim PayLoadData as ^byte,
dim PayLoadLen,
p2p_cmd_connectRequest,
currChannel,
p2pCapacityInfo as byte)

Description:
Creates new connection with more data in this connection.

Param PayLoadData: byte array
an array previously filled with these additional data and need to be passed as pointer.

Param PayLoadLen: byte
var, const or literal with value of amount of data that match with array PayLoadData.

Param p2p_cmd_connectRequest: byte
var, const or literal with the value of Connection request.

Param CurrChannel: byte
var, const or literal with current channel set.

Param CapacityInfo: byte
var, const or literal with the value of it value.

sub procedure MRF_CreateNewConnection(dim p2p_cmd_connectionRequest,
currChannel,
p2pcapacityInfo as byte)

Description:
Creates new connection to the other device.

Param p2p_cmd_connectionRequest: byte
var, const or literal that represents this command.

Param currChannel: byte
var, const or literal that represent the current channel.

Param p2pcapacityinfo: var, const or literal that represents its value.

Functions:

sub function MRF_RdLngAddress(dim addr as word) as byte

Description:
Read a long RAM address.

Return : byte
the byte value read of this address.

Param addr: word
var, const or literal that represents its address.

sub function MRF_RdShortAddress(dim addr as byte) as byte

Description:
Read a short RAM address.

Return: byte
the byte value of its address.

Param addr: byte
Var, const or literal that represents its address.

sub function MRF_GetPacket() as byte

Description:
Reads the RXn buffer and place all values in byte array RX_BUFFER.

Return: bate
byte with value of packetlen.

sub function MRF_GetPkt_Any(dim BaseAddr as word) as byte

Description:
Reads any RX buffer based in the parameter BaseAddr and place it in the byte array RX_BUFFER.

Return: byte
a byte with value of packetlen.

Param BaseAddr: word
Var, const or literal that represents one of four base RX buffer address.

Note: *this function reads all RX buffer more 2 FCS's but The packet length don't have the FCS's added.*

sub function MRF_GetPacketLen(dim addr as word) as byte

Description:
Read the value of any RX buffer without read packet.

Return: byte
a byte with a packet length of one these 4 RX buffer.

Param addr: word
Var, const or literal that represents its buffer address.

sub function MRF_ReadInterrupt() as ***Byte***

Description:
Read and clear the interrupt register.

Return: byte
a byte with status of register interrupts.

Note*: this function overwrite the function MRF_PacketReceived*

sub function MRF_PacketReceived() as boolean

Description:
Return if one packet was received or not.

Return: Boolean
False if no packet received.
True if a packet was received.

Note*: this function overwrite the function MRF_ReadInterrupt.*

sub function MRF_SetPower(Dim Pwr as Byte) as byte

Description:
Sets the Tx power level of module.

Return: byte
a byte with the current power set.

Param Pwr: byte
Var, const or literal from 0..31 the zero represents the 0.0dBm
1= -0.5dBm and so on.

Note*: these values are described in the datasheet of module.*

sub function MRF_GetPower() as byte

Description:
Read the Power level of module.

Return: byte
a byte with the currently power set 0..31 the zero represents
0.0dBm, 1= -0.5dBm and so on.

Note*: these values are described in the datasheet of module.*

sub function MRF_ReadSTATUS() as byte

Description:
Read the STATUS register.

Return: byte
a byte with value of STATUS.

sub function MRF_GetChannel() as Byte

Description:

Read the current channel set in the module.

Return: byte

a byte with the channel read from 11-26.

sub function MRF_SetChannel(dim nCh as Byte) as byte

Description:

Set new channel to the module.

Return: byte

a byte with currently channel set.

Param nCh: byte

Var, const or literal with value from 11-26.

sub function MRF_SendDataPkt(dim dstPAN as ^byte, dim dstADDRESS as ^byte,
dim Broadcast as Boolean,
dim SEND_AS as byte,
dim Sec_Enabled as boolean) as Boolean

Description:

Send a data packet to the other device.

Return: Boolean

false if a packet was not sent or true if it was sent ok.

***Note**: in this compiled version this function will return true*
Every time.

Param dstPAN: destintion PANID, byte array[2]

need to be passed as pointer.

Param dstADDRESS: long destination ADDRESS, byte array[8]

Need to be passed as pointer.

Param Broadcast: Boolean

is a Boolean that changes the mode how us sent the packet. If the packet will be sent as BROADCASTING mode its value is true. If the packet will be sent as unicasting mode its value is false. Like this the function will set de correctly value for headerLength of packet, and its are 15 for broadcast and 21 for unicast.

Param SEND_AS: byte

Var, const or literal that defines the mode how the packet will be sent. If a packet is command or Data.

Param Sec_Enabled: Boolean

If a packet is secured its value is true
If a packet is not secured its value is false

***Note**: for Sec_Enabled, maintain as false every time. It is not supported in this compiled version.*

sub function MRF_get_pktDC() as byte

Description: get data context.

function to retrieve only data/command of packet.

return:

the length of byte in the buffer from 0..127
the 7th bit inform if it's a command or
data received.
0=command, 1=data.

Note: *it's don't clear the rx buffer*

Improvements:

In this new release of MRF24J40MA lib I've added one more module
With some delays for when we change the MCU clock, to it don't return
Wrong values delay inner these delays.
This module need to be in the same folder that MRF24J40 lib.
It's called mrf_delays.mbas and is called by module lib.

By Marcio Nassorri
Any suggestion and or bug write to: nassorri@ttelecom.com.br
Enjoy.