GPS & Ad-Hoc Network

Jake Maschoff
Blaze Kotsenburg
Raj Patel

Our group worked on the Ad-Hoc network with the LoRa adafruit chips and the GPS module breakout board from adafruit. We were unable to send successful data between the transmitter and the receiver, but we were able to make a connection between the 2 boards. The connection is made through the use of a preamble to synchronize the clocks on the transmitter and the receiver. We had it so when we transmitted "Hello", the receiver would receive "HHHHHHHlllleeeeeeeoooooo" so we believe we are not reading from the FIFO buffer correctly nor clearing it correctly. Throughout the project we made lots of notes on the data sheet of the LoRa chip that was provided. I'll provide the notes below. Documentation wise we were able to produce a readFromReg method and writeToReg method which manipulates and grabs registers from the LoRa board, and transmitLoRa and readLoRa methods. These are abstractions for sending data into the chips (writing to correct buffers). We were also able to grab GPS data every second and parsed it so we only grabbed GGA data. This data was then displayed to the console via USART. The I2C data was also grabbed from the gyroscope and the LEDs turned on depending on the data received.

RFM9x LoRa Radio

GREAT LINK FOR INFO:

https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/using-the-rfm69-radio

- Vin - power in. This is regulated down to 3.3V so you can use 3.3-6VDC in. Make sure it can supply 150mA since the peak radio currents can be kinda high
- GND - ground for logic and power
- EN - connected to the enable pin of the regulator. Pulled high to Vin by default, pull low to completely cut power to the radio.

All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe.

Use whatever logic level is on Vin!

- SCK - This is the SPI Clock pin, its an input to the chip

- MISO - this is the Master In Slave Out pin, for data sent from the radio to your processor, 3.3V logic level

- MOSI - this is the Master Out Slave In pin, for data sent from your processor to the radio

- CS - this is the Chip Select pin, drop it low to start an SPI transaction. Its an input to the chip

- RST - this is the Reset pin for the radio. It's pulled high by default. Pull down to ground to put it into reset

- G0 - the radio's "GPIO 0" pin, also known as the IRQ pin, used for interrupt request notification from the radio to the microcontroller, 3.3V logic level. However, ***it must connect a hardware Interrupt pin***. Not all pins can do this! Check the board documentation for which pins are hardware interrupts, you'll also need the hardware interrupt number.

G1-G5

The radio's have another 5 GPIO pins that can be used for various notifications or radio functions. These aren't used for the majority of uses but are available in case you want them! All are 3.3V logic with no level shifting

LoRa Modes

Spreading Factor 6

SF = 6 Is a special use case for the highest data rate transmission possible with the LoRa modem. To this end several settings must be activated in the SX1276/77/78/79 registers when it is in use. These settings are only valid for SF6 and should be set back to their default values for other spreading factors:

1. Set SpreadingFactor = 6 in RegModemConfig2

2. The header must be set to Implicit mode.

3. Set the bit field DetectionOptimize of register RegLoRaDetectOptimize to value "0b101". Write 0x0C in the register RegDetectionThreshold.

ERROR CORRECTION :

On the receiver side, the bit RxPayloadCrcOn in the register RegModemConfig1 is not used and once the payload has been received, the user should check the bit CrcOnPayload in the register RegHopChannel. If the bit CrcOnPayload is at ‘1’, the user should then check the Irq Flag PayloadCrcError to make sure the CRC is valid.

FREQUENCY HOPPING: (think we might need to disable this)

To ease the implementation of FHSS systems the frequency hopping mode of the LoRaTM modem can be enabled by setting FreqHoppingPeriod to a non-zero value in register RegHopPeriod.

RECEIVER OPERATION:

Status registers provide status information during receiver operation.

It is important to notice that all the received data will be written to the FIFO data buffer even if the CRC is invalid, permitting user defined post processing of corrupted data. It is also important to note that when receiving, if the packet size exceeds the buffer memory allocated for the Rx, it will overwrite the transmit portion of the data buffer.

The RX modem address pointer is never reset as long as this mode is enabled. It is therefore necessary for the companion microcontroller to handle the address pointer to make sure the FIFO data buffer is never full.

In Rx single mode, low-power is achieved by turning off PLL and RF blocks as soon as a packet has been received. The flow is as follows:

1 Set FifoAddrPtr to FifoRxBaseAddr.

2 Static configuration register device can be written in either Sleep mode, Standby mode or FSRX mode.
3 A single packet receive operation is initiated by selecting the operating mode RXSINGLE.
4 The receiver will then await the reception of a valid preamble. Once received, the gain of the receive chain is set.
Following the ensuing reception of a valid header, indicated by the ValidHeader interrupt in explicit mode. The packet reception process commences. Once the reception process is complete the RxDone interrupt is set. The radio then returns automatically to Standby mode to reduce power consumption.
5 The receiver status register PayloadCrcError should be checked for packet payload integrity.
6 If a valid packet payload has been received then the FIFO should be read (See Payload Data Extraction below). Should a subsequent single packet reception need to be triggered, then the RXSINGLE operating mode must be re-selected to launch the receive process again - taking care to reset the SPI pointer (FifoAddrPtr) to the base location in memory (FifoRxBaseAddr).

In continuous mode the received packet processing sequence is given below.
1 Whilst in Sleep or Standby mode select RXCONT mode.
2 Upon reception of a valid header CRC the RxDone interrupt is set. The radio remains in RXCONT mode waiting for the next RX LoRaTM packet.
3 The PayloadCrcError flag should be checked for packet integrity.
4 If packet has been correctly received the FIFO data buffer can be read (see below).
5 The reception process (steps 2 - 4) can be repeated or receiver operating mode exited as desired.

•

In continuous mode status information are available only for the last packet received, i.e. the corresponding registers should be read before the next RxDone arrives.

Payload Data Extraction from FIFO

In order to retrieve received data from FIFO the user must ensure that ValidHeader, PayloadCrcError, RxDone and RxTimeout interrupts in the status register RegIrqFlags are not asserted to ensure that packet reception has terminated successfully (i.e. no flags should be set).

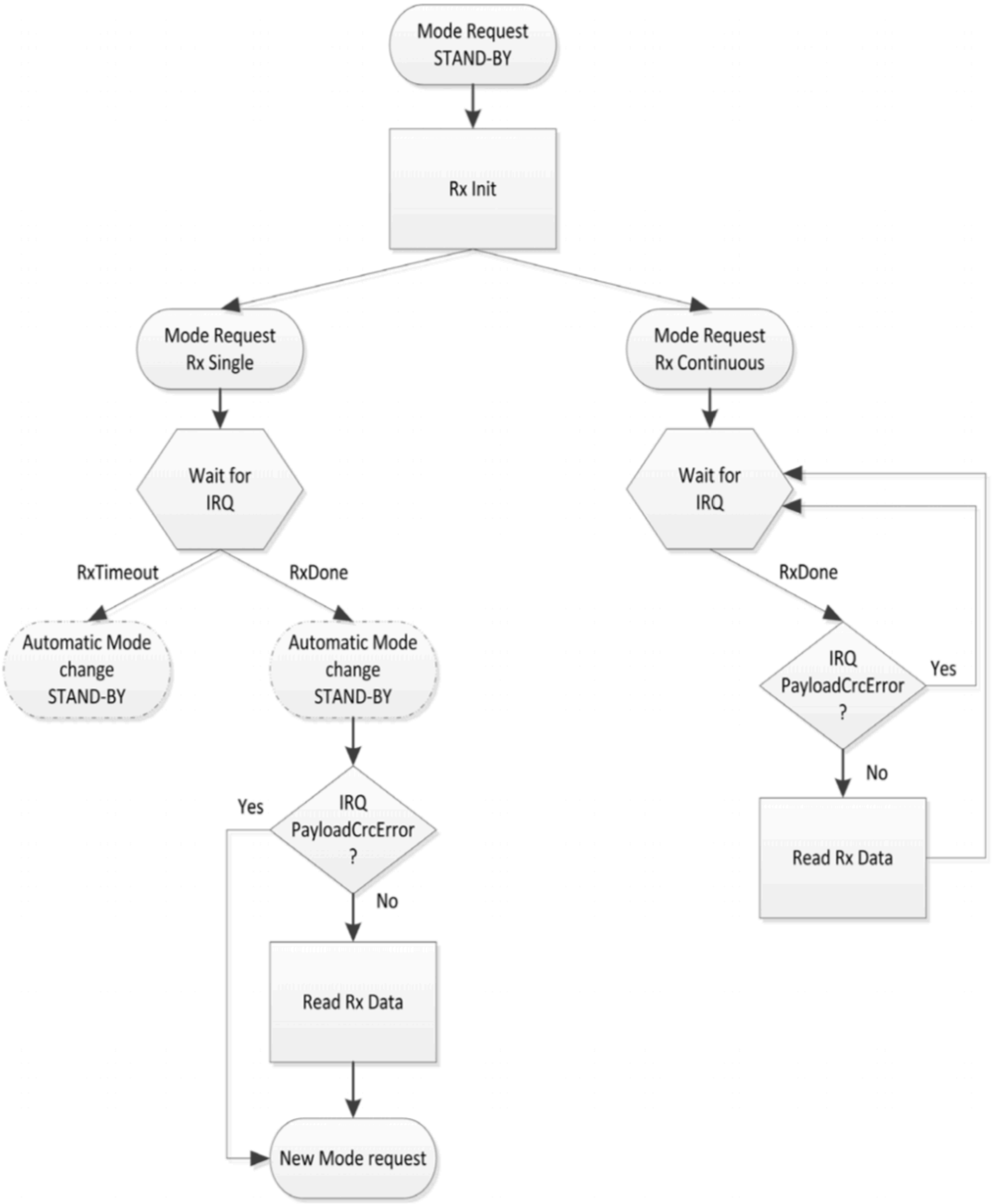
In case of errors the steps below should be skipped and the packet discarded. In order to retrieve valid received data from the FIFO the user must:

1. RegRxNbBytes Indicates the number of bytes that have been received thus far.

2. RegFifoAddrPtr is a dynamic pointer that indicates precisely where the Lora modem received data has been written up to.

3. Set RegFifoAddrPtr to RegFifoRxCurrentAddr. This sets the FIFO pointer to the location of the last packet received in the FIFO. The payload can then be extracted by reading the register RegFifo, RegRxNbBytes times.

4. Alternatively, it is possible to manually point to the location of the last packet received, from the start of the current packet, by setting RegFifoAddrPtr to RegFifoRxByteAddr minus RegRxNbBytes. The payload bytes can then be read from the FIFO by reading the RegFifo address RegRxNbBytes times.



TRANSMISSION OPERATION:

Static configuration registers can only be accessed in Sleep mode, Standby mode or FSTX mode.

The LoRaTM FIFO can only be filled in Standby mode.

Data transmission is initiated by sending TX mode request.

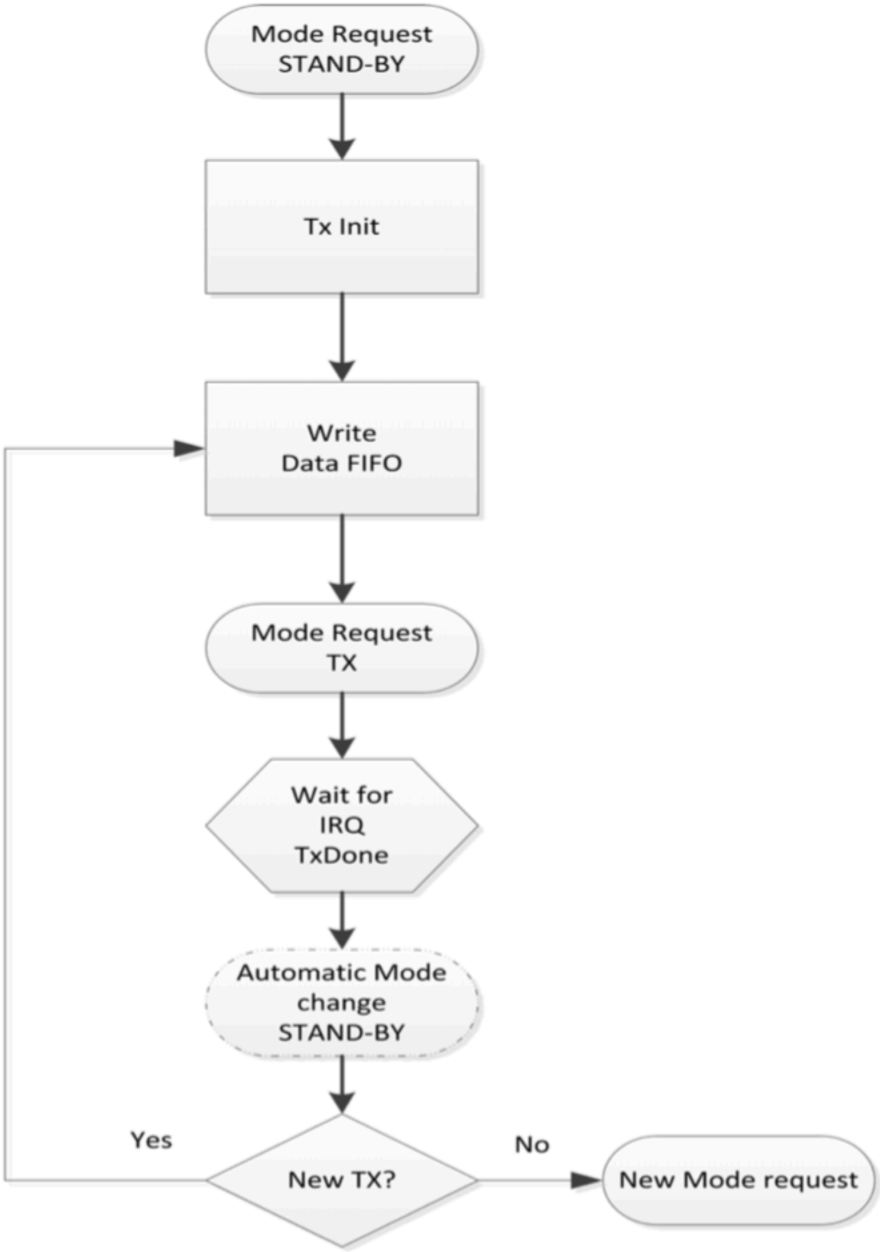
Upon completion the TxDone interrupt is issued and the radio returns to Standby mode. Following transmission the radio can be manually placed in Sleep mode or the FIFO refilled for a subsequent Tx operation.

In order to write packet data into FIFO user should:

1 Set FifoPtrAddr to FifoTxPtrBase.

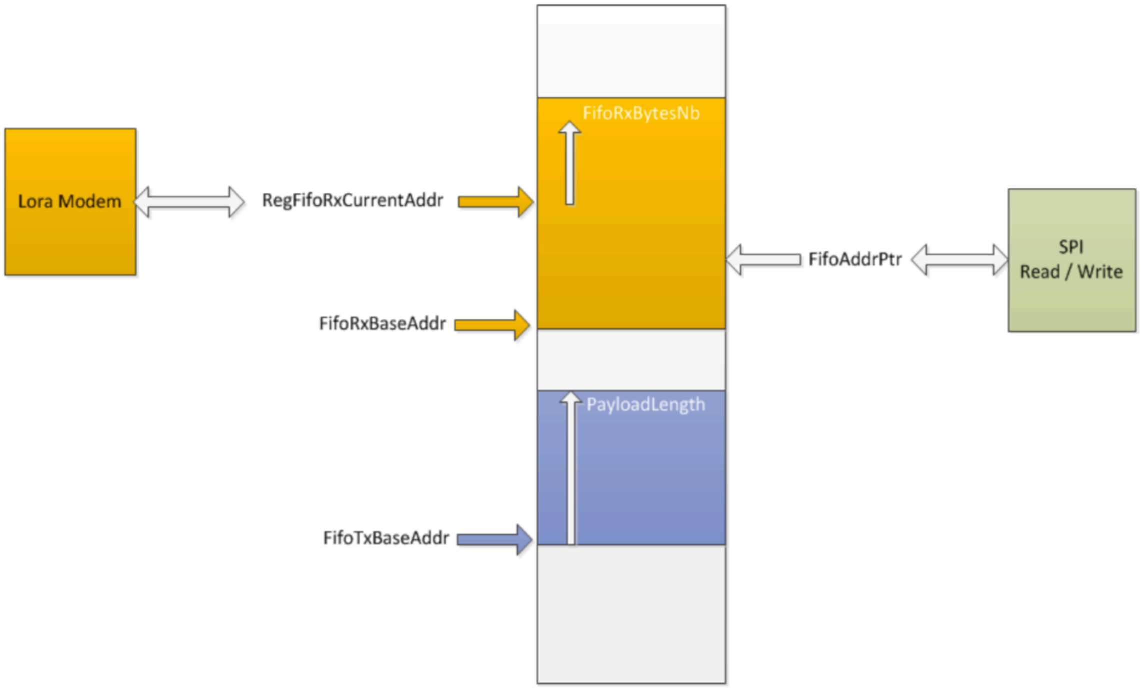
2 Write PayloadLength bytes to the FIFO (RegFifo)

-



DATA BUFFER:

The SX1276/77/78/79 is equipped with a 256 byte RAM data buffer which is uniquely accessible in LoRa mode. This RAM area, herein referred to as the FIFO Data buffer, is fully customizable by the user and allows access to the received, or to be transmitted, data. All access to the LoRaTM FIFO data buffer is done via the SPI interface. A diagram of the user defined memory mapping of the FIFO data buffer is shown below. These FIFO data buffer can be read in all operating modes except sleep and store data related to the last receive operation performed. It is automatically cleared of old content upon each new transition to receive mode.


Lora Modem
RegFifoRxCurrentAddr
FifoRxBytesNb
FifoRxBaseAddr
FifoAddrPtr
SPI
Read / Write
PayloadLength
FifoTxBaseAddr