

Digital Radio Projects

IP400 Radio Node for Phase 1 and 2

Document Number: TBD

Revision: 0.1

Status: Preliminary

Written By: Martin C. Alcock, M. Sc, MIEEE, VE6VH

Table of Contents

| | |
|--|----|
| Revision Status | iv |
| Reference Documents | iv |
| Intellectual Property Notice | v |
| Disclaimer | v |
| Introduction | 1 |
| Roadmap | 2 |
| Phase 1 ST Micro NUCLEO WL33CC2 | 3 |
| Connecting a GPS receiver | 4 |
| Phase 2 Raspberry Pi HAT | 5 |
| Connecting the STLink Debugger or a GPS Unit | 6 |
| Connecting a power amplifier | 6 |
| Migrating from Nucleo to the Pi | 6 |
| Operating the software | 7 |
| Main Menu | 8 |
| Setup Parameters | 9 |
| Mesh Table | 10 |
| Chat Mode | 10 |
| Frame Statistics | 11 |
| Setting the clock | 11 |
| Release Notes | 12 |
| V0.3b | 12 |
| V0.4 | 13 |

List of Tables

| | |
|--|----|
| Table 1 Revision status | iv |
| Table 2 Reference Documents | iv |
| Table 3 Connecting the STLink debugger | 6 |
| Table 4 External PA connection | 6 |
| Table 5 SPI connections from Nucleo to RPi | 6 |
| Table 6 Main Menu Items | 8 |
| Table 7 Setup parameters | 9 |
| Table 8 Mesh Table explanation..... | 10 |
| Table 9 Chat mode control keys | 10 |
| Table 10 Frame statistics | 11 |

List of Figures

| | |
|--|----|
| Figure 1 IP400 Node Development Roadmap..... | 2 |
| Figure 2 Phase 1 Nucleo board | 3 |
| Figure 3 Adafruit GPS setup | 4 |
| Figure 4 Raspberry Pi HAT (Zero form factor) | 5 |
| Figure 5 Main Menu..... | 8 |
| Figure 6 Setup Parameters..... | 9 |
| Figure 7 Mesh Status Table | 10 |

References

- [1] gnu.org, "General Public Licence," [Online]. Available: <https://www.gnu.org/licenses/gpl-3.0.en.html>. [Accessed 25th February 2018].
- [2] ST Microelectronics, "STM32WL33CC: Sub-GHz Wireless Microcontrollers.," ST Microelectronics, [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32wl33cc.html>. [Accessed 28 1 2025].
- [3] ST Microelectronics, "STM32 Nucleo-64 development board with STM32WL33CC MCU," [Online]. Available: <https://www.st.com/en/evaluation-tools/nucleo-wl33cc2.html>. [Accessed 26 01 2025].
- [4] ST Microelectronics, "Integrated Development Environment for STM32," [Online]. Available: <https://www.st.com/en/development-tools/stm32cubeide.html>. [Accessed 26 1 2025].
- [5] AdaFruit Industries, "Adafruit Ultimate GPS Logger Shield," [Online]. Available: <https://www.adafruit.com/product/1272#technical-details>. [Accessed 26 1 2025].

Revision Status

| Revision | Date | Description |
|----------|----------------------------------|------------------------------|
| 0.1 | January 26th ^h , 2025 | Initial draft |
| 0.2 | February 5 th , 2025 | Added release notes for V0.4 |
| | | |
| | | |
| | | |
| | | |

Table 1 Revision status

Reference Documents

| Author | Issue Date | Description |
|-----------|------------|------------------------------|
| M. Alcock | Jan 2025 | IP400 Protocol Specification |
| | | |

Table 2 Reference Documents

Intellectual Property Notice

The hardware components and all intellectual property described herein is the exclusive property of the Alberta Digital Radio Communications Society and others ("the owners"), all rights are reserved.

The owners grant licence to any Amateur for personal or club use, on an as is and where is basis under the condition that its use is for non-commercial activities only, all other usages are strictly prohibited. Terms and conditions are governed by the GNU public licence [1].

No warranty, either express or implied or transfer of rights is granted in this licence and the owner is not liable for any outcome whatsoever arising from such usage.

Copyright © Alberta Digital Radio Communications Society, all rights reserved. Not for publication.

Disclaimer

This document is a preliminary release for a product still in development and may be subject to change in future revisions. The software described herein may be subject to unpredictable behaviour without notice. You are advised to keep a can of RAID™ Ant, Roach and Program Bug killer handy. Spray liberally on the affected area when needed.

If any page in this document is blank, it is completely unintentional.

Introduction

The IP400 project was launched to experiment with digital mesh networking on the 400 MHz band, using commercial devices designed to run in this band.

To get the project rolling, code has been developed for an STM32WL33 microcontroller [2], using an off the shelf evaluation board called a 'nucleo' [3]. This will be superseded with other platforms for the Raspberry Pi and Arduino form factors, and eventually new devices that will include signal processing for higher orders of modulation, and RF components for other bands.

Operation of the node is segmented into a physical layer that runs on the microcontroller, and other layers that run on different host processors. The lower layer code contains the bare minimum to send and receive frames, repeating frames and building a mesh table, and contains a simple application.

The application code includes a simple setup menu, and the ability to change and store station and radio parameters, as well as a simple chat application to demonstrate the capabilities. It periodically sends a 'beacon' frame to build the mesh tables, which contains information about the station, including latitude, longitude and grid square. Provision has also been made to connect to a GPS receiver to update the position information dynamically.

Roadmap

Figure 1 illustrates the roadmap for the IP400 node development.

IP400 Node Development Roadmap

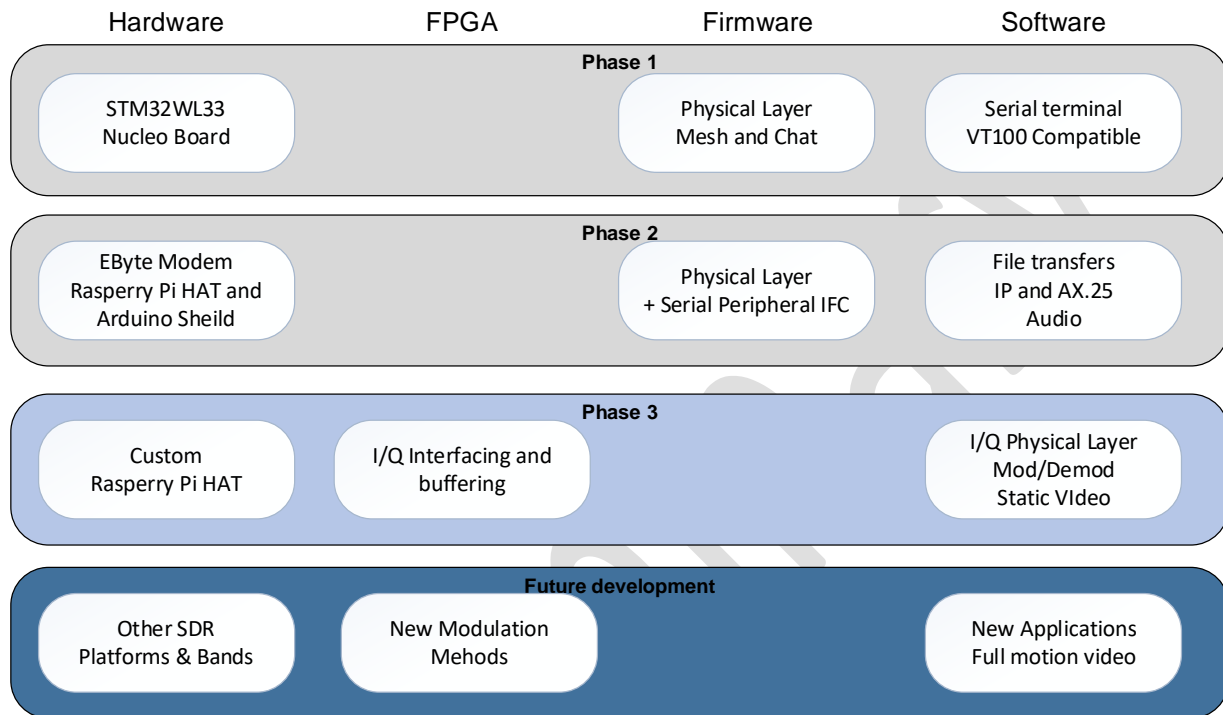


Figure 1 IP400 Node Development Roadmap

There are four phases of development for the IP400 node:

- Phase 1 is the initial evaluation phase using the nucleo board and a simple chat application and is purely a firmware exercise. Using the PuTTY application, a connection can be made to the USART on the board. The software will implement a basic frame transmitter and receiver and be able to build a mesh table and repeat frames. The firmware can only be loaded using the ST integrated development environment (IDE) [4] and an integrated STLINK Debugger.
- Phase 2 utilizes an off the shelf module to support an integrated HAT for the raspberry Pi platform, or an Arduino shield. The firmware will be upgraded to include a high speed SPI, plus a downloader will be developed to alleviate the absolute requirement for the IDE, but support for an external STLINK debugger has been provided, so the IDE can also be utilized as an option. New applications on the Pi can be developed for audio, file transfers, AX.25 and encapsulated IP.
- Phase 3 will introduce new custom hardware and FPGA-based signal processing for higher modulation methods and add new applications, to be determined.

- The final phase will migrate to other bands and platforms.

Phase 1 ST Micro NUCLEO WL33CC2



Figure 2 Phase 1 Nucleo board

The module consists of two PCB's, once contains the STM32WL33 microcontroller, the second an STLink debugger with a type 'C' USB connection. To install the firmware, the STM32CubeIDE must be used. Download the zip file and source code from the Github site, unzip the project, import it into the IDE, and add the source files from the IP400 directory. Compile the code and run it using the IDE downloader.

Start a PuTTY session in serial mode to the first COM port on the debugger, set it to 115200 bits/sec and DEC VT100 emulation mode. This can remain in place, even when the code is restarted or the reset key on the board is hit.

The basic functionality includes a menu selection to set station or radio parameters, a mesh table builder, packet repeater, and a simple chat application.

Connecting a GPS receiver

A GPS receiver may be connected to the Nucleo using the Arduino connectors. The recommended module is the 'Ultimate GPS logging sheild' from AdaFruit industries [5]. Connection is straightforward, but to enable the serial data the switch in the top right must be in the Soft Serial position. See Figure 3.

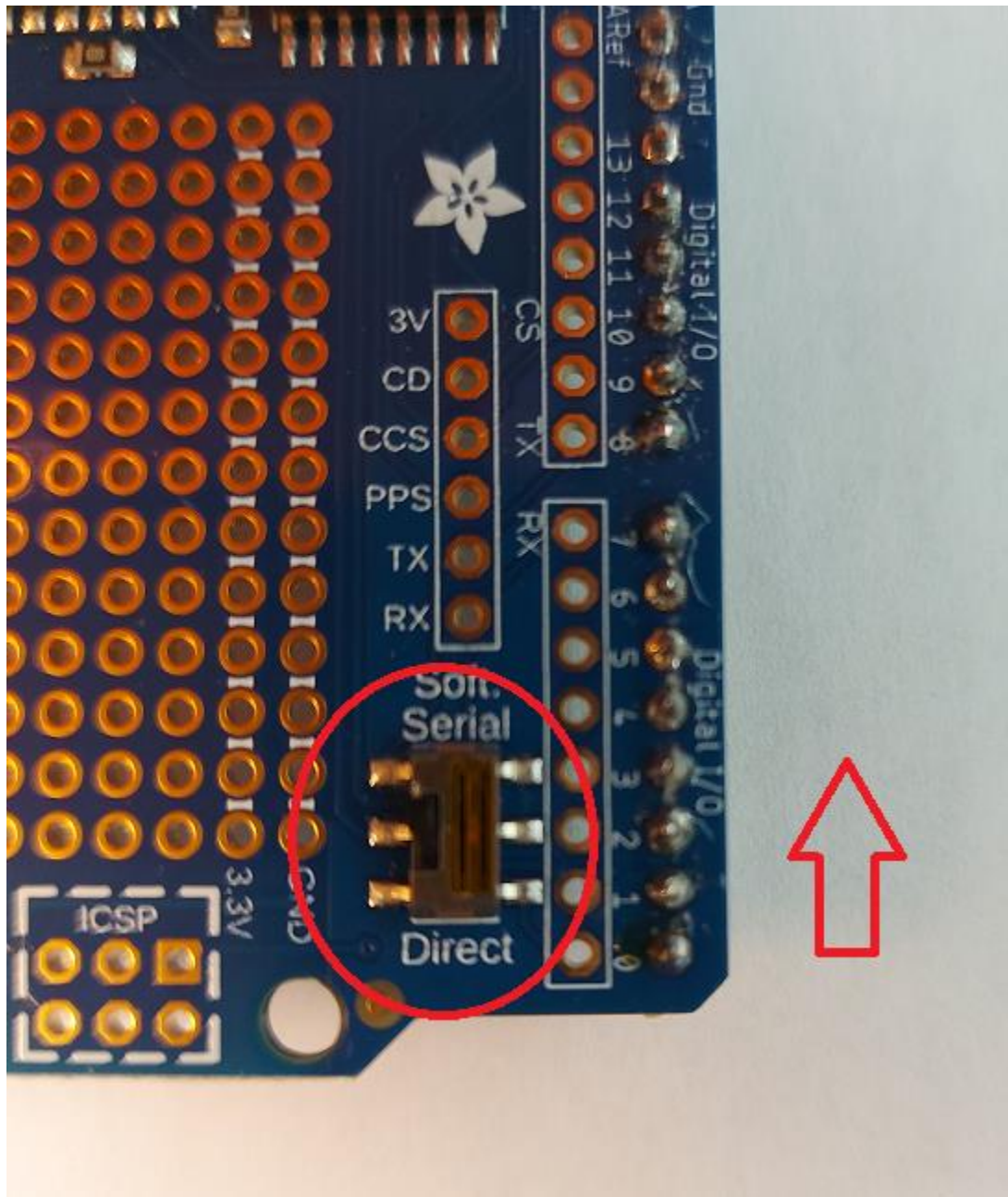


Figure 3 Adafruit GPS setup

Phase 2 Raspberry Pi HAT

Figure 4 illustrates the Pi HAT in the Zero form factor.

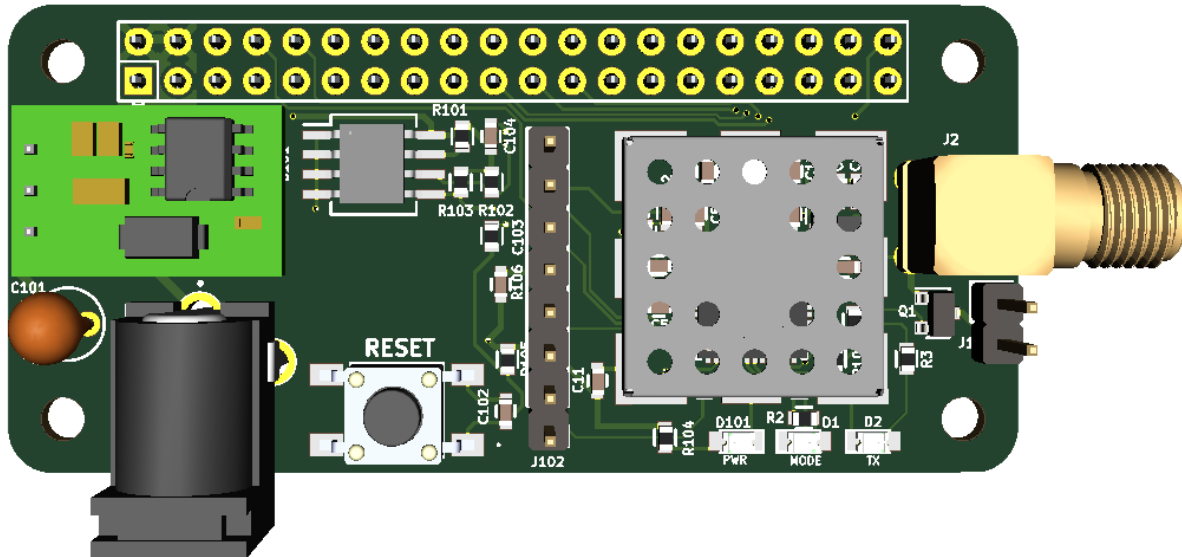


Figure 4 Raspberry Pi HAT (Zero form factor)

The Pi HAT features an EByte module that contains the same processor as the Nucleo board, but in a module form. Its basic feature set includes:

- EEPROM for Pi HAT standard identification
- STMWL33 processor with 400 MHz coupler
- PA control for external amplifier
- Switching power supply from 7-33V to power the Pi
- Reset Switch
- 3 LED's: one power, one bi-colour, and one red.
- STLink connector with VCOM port (LPUART)
- TTYama0 connection to USART1
- SPI connection for high speed data
- Reset/Boot from the Pi.

The application code can be installed in one of two ways: using the IDE can be used with an external STLINK debugger in the same manner or copy the files to the target platform and use the makefile and bootloader to install the code.

Connecting the STLink Debugger or a GPS Unit

Table 3 illustrates the connections for the debugger, or GPS receiver.

| Pin | Function | Debugger | GPS Receiver |
|-----|----------|-------------------------------|-------------------|
| 1 | VCP_TX | Transmit data from the LPUART | GPS receive data |
| 2 | VCP_RX | Receive data to the LPUART | GPS transmit data |
| 3 | VCC | 3.3V supply | 3.3V supply |
| 4 | SWCLK | Debugger serial clock | NC |
| 5 | GND | Ground | Ground |
| 6 | SWDIO | Debugger serial data | NC |
| 7 | NRST | Reset input | NC |
| 8 | NC | No connection | NC |

Table 3 Connecting the STLink debugger

Connecting a power amplifier

The Pi module supports a connection to a power amplifier that is buffered with an FET transistor. The signal can also be found on CN3 on the Nucleo board, but an external buffer is required. The buffered signal is grounded when active, the morpho connector is at 3.3V. Connections are shown in Table 4.

| Signal | WL33 Pin | GPIO | Morpho CN3 | Buffered RPi |
|-----------|----------|------|------------|--------------|
| PA Enable | PB14 | 21 | 11 | J201-1 |
| Ground | | - | | J201-2 |

Table 4 External PA connection

Migrating from Nucleo to the Pi

The same code runs on the module as on the Nucleo board, the only difference is the SPI high speed data connection. However, a Nucleo board can be connected to a RPi to offer the same functionality as the HAT board.

The connection is made using cable jumpers from CN4 on the Nucleo board's morpho connector, to the HAT connector on the raspberry Pi. The table below lists the signal name, it origin on the WL33 chip, the GPIO designation, and the pin where it can be found on the morpho, plus the corresponding pin on the Raspberry Pi HAT connector.

| Signal | WL33 Pin | GPIO | Morpho CN4 | RPi |
|----------|----------|------|------------|-----|
| SPI_MOSI | PB8 | 33 | 11 | 25 |
| SPI_MOSI | PB9 | 34 | 13 | 19 |
| SPI_SS | PB10 | 37 | 17 | 24 |
| SPI_SCK | PB11 | 31 | 15 | 23 |

Table 5 SPI connections from Nucleo to RPi

Accessing the Menu

If you are not using the supplied image, there are three steps that need to be carried out to be able to connect to the application menu:

1. Run the configuration program 'raspi-config', and choose Interface Options, then Serial Port (I6). Turn off the login shell by answering 'no' to the first question. Enable the serial port by answering 'Yes' to the second question. Then re-boot the Pi.
2. Download and install minicom if you do not have it, with 'sudo apt-get install minicom'.
3. Launch minicom with the command 'minicom -b 115200 -D /dev/ttyAM0'.

You may also want to enable SSH access if you have an ethernet connection to the Pi and PuTTY on a remote machine.

Installing the software on the HAT

The module is shipped preconfigured, but if you want to update it, download the IP400 code and flash loader from the Github site as follows:

```
git clone https://github.com/adrcs/ip400/code/ip400.git
```

If you are not using a debugger, you can flash the code using the following method:

Starting in the home directory:

```
cd STM32Flash
make
make install
cd ..

cd IP400
./flash.sh Debug/WL_33_rPi.elf
```

Operating the software

Main Menu

The main menu is enabled after a restart. It is shown Figure 5.

```
A) List setup parameters
B) Mesh Status
C) Chat Mode
D) Dump Frame stats
L) LED test
R) Set Radio Parameters
S) Set Station Parameters
T) Set clock (HH:MM)

W) Write Setup Values
X) Exit

Select an item->
```

Figure 5 Main Menu

The menu items are described in Table 6.

| Menu Item | Function |
|-----------|--|
| A | Lists the current setup parameters, including clock, station and radio |
| B | Displays the mesh table contents |
| C | Enters chat mode |
| D | Dumps the frame statistics |
| L | Runs an LED test |
| R | Sets the radio parameters |
| S | Sets the station parameters |
| T | Sets the time of day clock |
| W | Writes the setup data to the flash |
| X | Exits the menu and puts the node in silent mode |

Table 6 Main Menu Items

Most are self-explanatory, however further explanation is required.

Setup Parameters

Figure 6 illustrates the setup parameters. Menu Items 'S' and 'T' set the station and radio parameters; menu item 'T' sets the time of day clock. Table 7 explains them.

```
System time is 00:02:00
Station Callsign->VE6VH
Latitude->51.08
Longitude->-114.10
Grid Square->DO21vd
Capabilities->FSK
Repeat mode on by default
Beacon Interval->5 mins

RF Frequency->445.750 MHz
Modulation method->4FSK
Data Rate->100.0 Kbps
Peak Deviation->25.0 KHz
Channel Filter Bandwidth->200.0 KHz
Output Power->14 dBm
PA Mode->TX_HP 20dBm Max
Rx Squelch->-60

Hit enter to continue->
```

Figure 6 Setup Parameters

| Item | Purpose | Format |
|--------------------------|---|---------------------------------|
| Time | Time of day clock with 10 second resolution | HH:MM |
| Callsign | Station callsign | Up to 6 characters ¹ |
| Latitude | Latitude, positive for N, negative for S | ±xxx.xxx ² |
| Longitude | Longitude, positive for E, negative for W | ±xxx.xxx |
| Grid Square | Home grid square | XXNNxx |
| Capabilities | Lists station operating modes | FSK or OFDM |
| Repeat Mode | Set the repeat flag in sent frames | On or Off |
| Beacon Interval | Interval between beacons, normally 5 mins | xx minutes |
| RF Frequency | RF operating frequency from 420 to 450 MHz | XXX.XXX or NNNNNNNNNN |
| Modulation | Modulation method, usually 4FSK | 2FSK or 4FSK |
| Data Rate | Data rate 1.2 to 600Kb/s | xx.xx KHz or NNNNNN |
| Peak Deviation | Peak FM deviation | xx.xx KHz or NNNNNN |
| Channel Filter BW | BW of the receiver | xx.xx KHz or NNNNNN |
| Output power | Sets the transmitter power | 0 to +20 dBm |
| Rx Squelch | Sets the receiver threshold | -30 to -130 dBm |

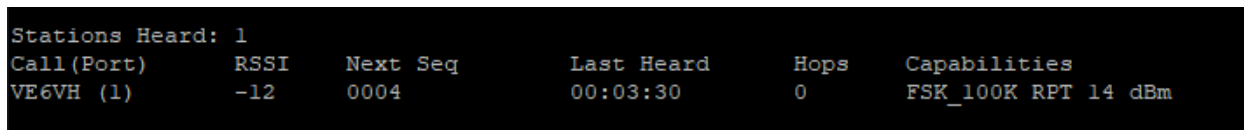
Table 7 Setup parameters

¹ An extended method has been designed but not yet implemented

² Used in beacon frames, unless a GPS receiver is connected

Mesh Table

Figure 7 illustrates the mesh table contents, and Table 8 explains them.



```
Stations Heard: 1
Call (Port)      RSSI      Next Seq      Last Heard      Hops      Capabilities
VE6VH (1)        -12       0004          00:03:30        0          FSK_100K RPT 14 dBm
```

Figure 7 Mesh Status Table

| Item | Explanation |
|--------------|--|
| Call(port) | Callsign of the transmitting station, and port number heard |
| RSSI | Receive signal strength when SYNC received (in dBm) |
| Next Seq | Next anticipated sequence number |
| Last Heard | TOD clock reading when last heard |
| Hops | Number of repeat hops, 0 indicates a direct signal |
| Capabilities | Data rate, repeat capabilities and transmitted signal strength |

Table 8 Mesh Table explanation

Chat Mode

There are several control keys in the chat mode that are interpreted, as shown in Table 9.

| Key | Purpose |
|-----------|---|
| ESC | Enables entering a new destination address, normally broadcast. |
| CTRL+'R' | Toggles repeat mode |
| CTRL+'D' | Toggles dump mode |
| CTRL+'Z' | Exits chat and returns to the main menu |
| ENTER | Sends the current frame |
| BACKSPACE | Deletes the last character entered, or more if repeated |

Table 9 Chat mode control keys

Chat frames are displayed as follows:

Originating callsign(source port)>Destination Callsign(dest port)[number of repeats]:text message...
NOCALL(17)>BROADCAST(1085)[0]:hello to you...

The chat port number is always 17.

Frame Statistics

The statistics for each frame are shown in Table 10.

| Item | Meaning |
|---------------------------|--|
| Transmitted Frames | The number of frames transmitted |
| Good Rx Frames | The number of frames with a good CRC |
| CRC Errors | The number of frames with CRC errors |
| Rx Timouts | The number of times the receiver timed out |

Table 10 Frame statistics

Setting the clock

The time of day is entered as HH: MM. The clock has a granularity of 10 seconds and uses 24 hour format, and knows nothing about time zones, as it considers you are in your home zone.

Release Notes

V0.3b

Minor change made to the radio parameters display and entry. The frequency is now displayed as xxx.yyy and can be entered the same way. The old method of specifying it with all the digits is still supported.

Added the ability to set the clock. Format is HH:MM, 24 hours, time zone agnostic.
Granularity is 10 seconds. Current time added to setup display.

Chat mode was not changed, a reminder on how it works:

In the chat mode, to send a line of text just key it in and hit enter. Backspace removes one key at time. CTRL+R changes the repeat flag. This flag tells a receiving station to repeat the frame. Defaults to setup. CTRL+D: sets dump mode. When enabled, the frame header will be dumped instead of interpreted. ESC key (only at the beginning of a line) enables entry of a destination callsign. Defaults to broadcast. CTRL+Z. Exits chat mode and returns to the main menu.

The received frames are interpreted as:

Originating callsign(source port)>Destination Callsign(dest port)[number of repeats]:text message...
NOCALL(17)>BROADCAST(1085)[0]:hello to you...

The number of repeats is the number of times the frame has been repeated.

Beacon frames can be dumped to the console, by enabling `__DUMP_BEACON` in `frame.c`. If enabled, ensure that the receiving station is in receive mode to avoid running out of heap space.

Limited support has been added for a GPS receiver using the LPUART. It uses the DMA mode to receive a message and parses a GGA message for lat/long and a timestamp. These fields are used in the beacon frame. It is enabled by `__ENABLE_GPS` in `beacon.c`

BUGS FIXED

The beacon mode is now fully implemented. See the protocol spec for a description. Frequency is in the station setup, it can be overwritten by enabling `__SPEED_DAEMON` in `beacon.c`.

The mesh table is built using beacon frames. It lists the callsign, capabilities and last heard time.

The setup parameters can now be changed and stored in the internal flash.

NEW BUGS

The repeat mode has not yet been implemented.

A facility to set the clock needs to be added to the main menu.

The GPS code has yet to be fully tested.

V0.4

The receive frame processing was overhauled to make it consistent with the transmitter. Both now work in units of IP400_FRAMES.

The mesh table is now operational, and now contains an RSSI reading, timestamp, hop count, and capabilities of each station heard.

The repeat mode has been implemented. An inbound frame with the repeat flag set and hop count less than the maximum is repeated, providing it was not sourced by the receiving station.

The frame now contains a sequence number which is stored in the mesh table. If a frame is received with a previously known sequence, it is dropped.

The transmitted power is sent in the beacon frame to enable calculation of the path loss.

An LED manager has been added to control the LED's and provide better error indications. See the description in the documentation.

Menu changes

The current firmware and time of day clock have been moved to the list setup parameters menu item. The mesh status now lists the receive RSSI, next expected sequence number, a timestamp, number of hops, and capabilities.

The frame status now reports how frames were processed, as beacons, repeated, duplicated and dropped. It also shows the current radio errors, if any, and the current radio FSM state.

An LED test has been added, as well as a set time of day clock.

BUGS FIXED

Repeat has been implemented.

NEW BUGS

GPS code is still pending.