1. Installation:

First, I used another SD memory card and performed a clean install. To avoid problems with my RPi4B environment…
This will get rid of the problem with my RPi4 environment.

(1). Install Pi OS from Pi Imager(v1.6.1) & 'sudo apt-get update' and 'sudo apt-get upgrade', and set RPi4 Configuration according to Radioberry's Wiki

(2). Install the developement release of Radioberry2 (ver.73-0, driver::0.9)

```
$ cd /tmp
$ wget https://raw.githubusercontent.com/pa3gsb/Radioberry-2.x/master/SBC/rpi-4/releases/dev/radioberry_install.sh
$ sudo chmod +x radioberry_install.sh
$ ./radioberry_install.sh  (select CL016)
```

Chek

$ sudo modinfo radioberry

```
filename:       /lib/modules/5.10.17-v7l+/kernel/drivers/sdr/radioberry.ko
version:        0.9
license:        GPL
description:    Radioberry SDR device driver. (rpi-4)
author:         Johan Maas - pa3gsb@gmail.com
srcversion:     5B8DB71716B53CAC5B9D980
alias:          of:N*T*Csdr,radioberryC*
alias:          of:N*T*Csdr,radioberry
depends:
name:           radioberry
vermagic:       5.10.17-v7l+ SMP mod_unload modversions ARMv7 p2v8
```

$ sudo radioberry

```
================================================================
================================================================
Radioberry V2.0


Supports 4 receivers and 1 transmitter.

Build version: 2021.04.25

Have fun Johan PA3GSB

Report requests or bugs to <pa3gsb@gmail.com>.
================================================================
================================================================
Radioberry gateware version 73-0.
No Alex, N2ADR or generic filters interface board connected to radioberry
Radioberry amplifier config failed; only a problem if amplifier is installed.
Radioberry, Starting packet rx part.
Radioberry, Starting packet control part.
Radioberry, Starting packet tx part.
Your Radioberry is registered: http://www.pa3gsb.nl/radioberry/api/read.php
```

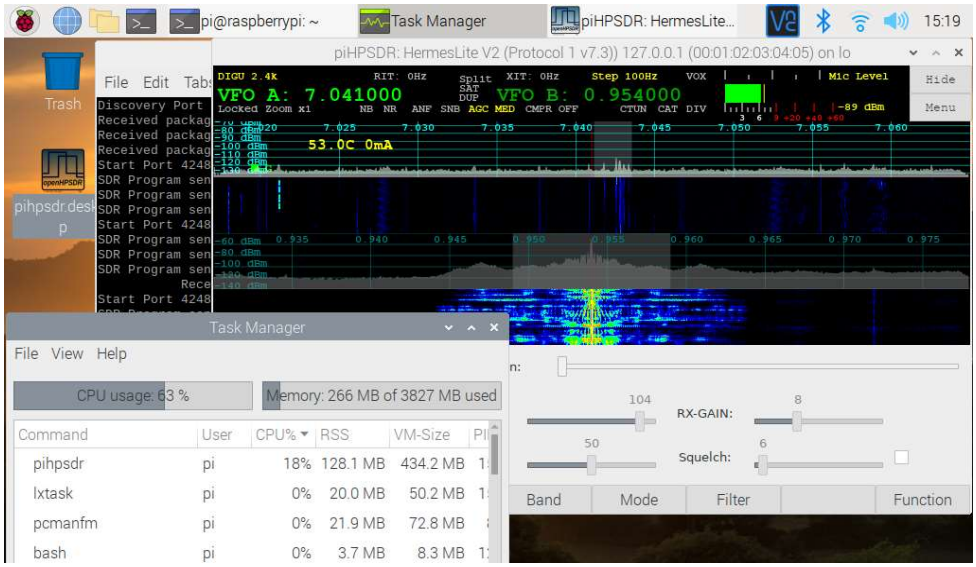
(3). Install the development version of pihpsdr

```
$ cd /tmp
$ wget https://raw.githubusercontent.com/pa3gsb/Radioberry-2.x/master/SBC/rpi-4/releases/dev/pihpsdr_install.sh
$ sudo chmod +x pihpsdr_install.sh  ( select 'wdsp' first, 'pihpsdr' second )
$ ./pihpsdr_install.sh
```

Chek 1

Configure GPIO as https://github.com/pa3gsb/Radioberry-2.x/wiki/piHPSDR-running-CW-mode

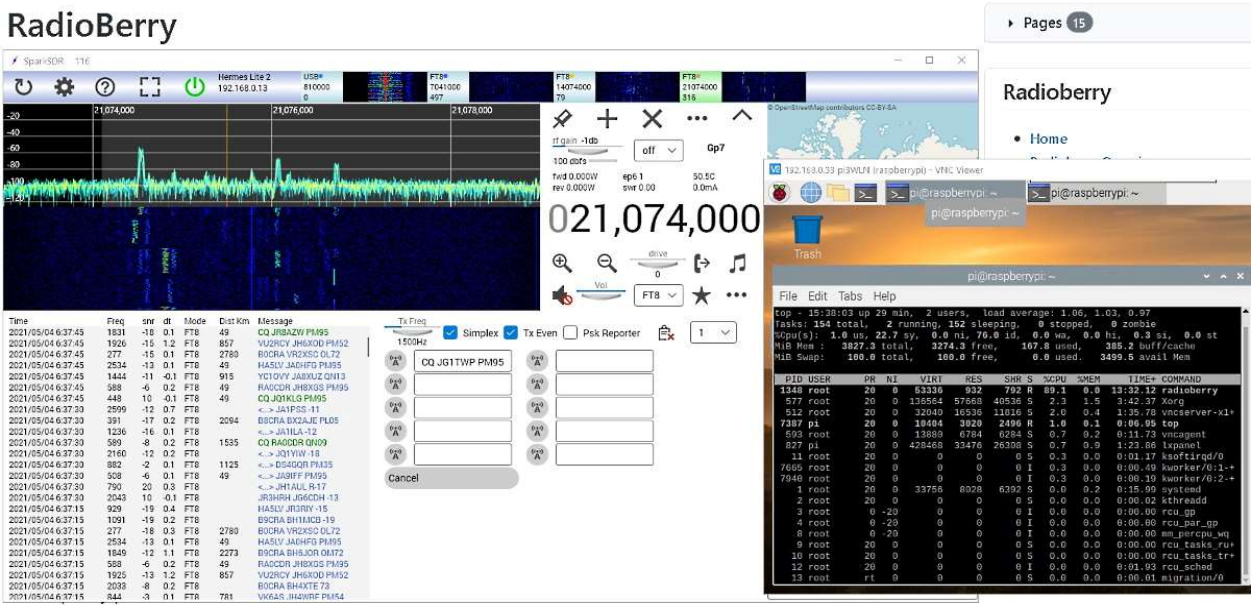
Nice working. ..Receiving 2 bands, Sampling:48k



Chek 2

SparkSDR 2.0.5.7 (Windows10 PC)

Nice working. ..Receiving 4 bands. Sampling:48k



(4). Install the Radioberry FT8 decoder

copy ft8_directory
from https://github.com/pa3gsb/Radioberry-2.x/tree/master/SBC/rpi-4/deoders/ft8/* to ~/decodrs/ft8/.
copy radioberry_ioctl.h
from https://github.com/pa3gsb/Radioberry-2.x/tree/master/SBC/rpi-4/device_driver/driver to ~/decodrs/ft8/.

copy radioberry.rbf for receivers using a sample rate of 4000 Hz (CL016)
from
https://github.com/softerhardware/Hermes-Lite2/blob/master/gateware/variants/radioberry_cl016_4000/build/radioberry.rb
to ~/decodrs/ft8/.

The following must be done to make the decoder work

```
$ sudo systemctl stop radioberry
$ sudo modprobe -r radioberry
$ cd ~/decoders/ft8
$ sudo cp /lib/firmware/radioberry.rbf ./radioberry_730cl016.rbf (backup gw73.0)
$ sudo cp radioberry.rbf /lib/firmware
$ sudo modprobe radioberry
```

2. Debug

Problem: With 'Radioberry FT8 decoder', there were no results only RX1 (Freq Array[0]).

Checking the operation of the Radioberry FT8 Decoder.

```
$ cd ~/decodes/ft8
$ vi recording-ft8.c  ( change NRX , Freq. )
$ make  ( make 'recording-ft8' )
$ sudo ./recording-ft8    ( Receiving about 3 min )
$ sudo cp ./recordings/*.c2  tmp/.  ( for backup recording files )
$ sudo ./decode-ft8.sh
$ sudo cp ./decodes/*.txt  tmp/.  ( for backup decoded results )
```

2.1 Changing details(1)

To see if I could get a decoded result, I changed the NRX of 'recording-ft8.c'.

$ diff recording-ft8.c recording-ft8.c orginal

```
25c25
< #define NRX 8              <-- original
---
> #define NRX 2              <-- change NRX = 8, 7, 6, 5, 4, 3, 2, 1
28c28,30
< u_int32_t freqArray[8] = {1841500, 3574500, 5358500, 7075500, 10137500,  14075500, 18101500,  21075500  <-- original
---
> // u_int32_t freqArray[8] = {1841500, 3574500, 5358500, 7075500, 10137500,  14075500, 18101500,  21075500  };
> u_int32_t freqArray[8] = {14075500, 7042500, 18101500, 21075500, 7075500, 10137500, 24916500, 28075500 <-- change Freq.1 *1
> //u_int32_t freqArray[8] = {7042500, 14075500, 18101500, 21075500, 7075500, 10137500, 24916500, 280755 <-- change Freq.2 *2
```

Notes: *1 listed in order of the number of FT8 on air stations.
*2 To show that there ware FT8 stations in FreqArray[0], Swapped Freq. of FreqArray[0] and [1].
7042500 is FT8 Freq. for domestic in Japan

Decode Result(1)

NRX	FreqArray[0]	FreqArray[1]	FreqArray[2]	FreqArray[3]	FreqArray[4	FreqArray[5]	FreqArray[6]	FreqArray[7]
	20m	40m Japan	17m	15m	40m	30m	12m	10m
8	NG	OK	OK	OK	OK	OK	^	^
7	NG	OK	OK	OK	OK	OK	^	^
6	NG	OK	OK	OK	OK	OK	^	^
5	NG	OK	OK	OK	OK	^	^	^
4	NG	OK	OK	OK	^	^	^	^
3	NG	OK	OK	^	^	^	^	^
2	NG	OK	^	^	^	^	^	^
1	NG	^	^	^	^	^	^	^

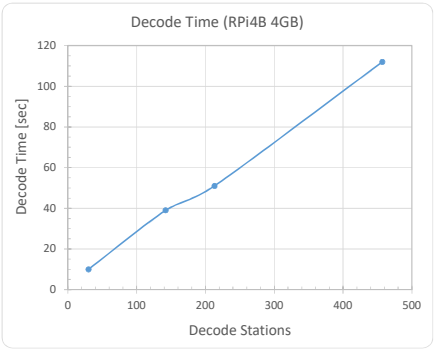
NRX	FreqArray[0]	FreqArray[1]	FreqArray[2]	FreqArray[3]	FreqArray[4	FreqArray[5]	FreqArray[6]	FreqArray[7]
	40m Japan	20m	17m	15m	40m	30m	12m	10m
2	NG	OK	^	^	^	^	^	^
1	NG	^	^	^	^	^	^	^

OK: Decoding results available
NG: No decodeing result.

Decode Time.

Band	Decoded Stations	Decoded time*
4	457	112
3	213	51
2	142	39
1	30	10

* unit: sec



Conclusion(1)

~~Only the Freq. set to RX1'FreqArray[0]) has no decoding results.~~ <- See Conclusion(2)
Up to 6 frequencies in FPGA CL016. (if FreqArray[0] OK)
Decode time: About 250decodes/60sec

consideration(1)

The command to set RX1(FreqArray[0]) is not correctly received by the FPGA.
This is not a problem with the FPGA, as the other frequencies are fine. (*Refer to 3-(3) RTL*)
It is most likely a timing problem or GateWare of CL016 failure.

2.2 Changing Details(2)

The command sending RX1 seems to different timing from the others, so I tried to change the command line. (NRX=2)

(0) recording-ft8.c **Original bellow.** (*Refer to 3-(1),(2) Plotocol*)

```
:
memset(commands,0,256); // initialise the commands.

commands[0x00] = 0x00000000;
commands[0x04] = freqArray[0];        //f1

send_control(0x04);
send_control(0x00);
usleep(100000);

if (NRX > 1) {
      commands[0x00] = 0x00000008;
      commands[0x06] = freqArray[1];      //f2

      send_control(0x06);
}

if (NRX > 2) {
      commands[0x00] = 0x00000010;
      commands[0x08] = freqArray[2];      //f3

      send_control(0x08);
}

if (NRX > 3) {
:
```

Log (NRX=2)

```
$ sudo ./recording-ft8

Radioberry gateware version 72-5.
RB-Command = 01 Command = 04  command_data = 00D6C66C  // RX1(20m) NG
RB-Command = 01 Command = 00  command_data = 00000000  // Set 48kHz, Dup off, Number of RX=1, ...
usleep(100000);
RB-Command = 01 Command = 06  command_data = 006B75C4  // RX2(40mJ) OK
RB-Command = 01 Command = 14  command_data = 0000005F  // Pure off, Set LNA Gain
RB-Command = 01 Command = 00  command_data = 00000008  // Set 48kHz, Dup off, Number of RX=2, ...
usleep(100000);

FT8-recording process...
FT8 recording timer started.
............................................................  <- 59dots, every dot/1sec (60sec/line)
FT8 recording done, now writing to c2 files....                          4k sampling/sec
FT8 recordings written to c2 files. Ready for decoding.
............................................................  <- 59dots, every dot/1sec (60sec/line)
FT8 recording done, now writing to c2 files....                          4k sampling/sec
FT8 recordings written to c2 files. Ready for decoding.
..............
```

(1) recording-ft8.c (try1) **add send_control(0x04): RX1 after send_control(0x00),usleep**

```
memset(commands,0,256); // initialise the commands.

commands[0x00] = 0x00000000;
commands[0x04] = freqArray[0];        //f1 to rx1

send_control(0x04);
send_control(0x00);
usleep(100000);

commands[0x00] = 0x00000000;
commands[0x04] = freqArray[0];        //f1 to rx1
send_control(0x04);

if (NRX > 1) {
      commands[0x00] = 0x00000008;
      commands[0x06] = freqArray[1];      //f2 to rx2

      send_control(0x06);
}

if (NRX > 2) {
      commands[0x00] = 0x00000010;
      commands[0x08] = freqArray[2];      //f3

      send_control(0x08);
}
```

Log (NRX=2) and Result (1)

```
$ sudo ./recording-ft8

Radioberry gateware version 72-5.
RB-Command = 01 Command = 04  command_data = 006B75C4  // RX1(40mJ)
RB-Command = 01 Command = 00  command_data = 00000000
usleep(100000);
RB-Command = 01 Command = 04  command_data = 006B75C4  // RX1(40mJ) NG
RB-Command = 01 Command = 06  command_data = 00D6C66C  // RX2(20m) OK
RB-Command = 01 Command = 14  command_data = 0000005F
RB-Command = 01 Command = 00  command_data = 00000008
usleep(100000);

FT8-recording process...
FT8 recording timer started.
.............................................................
```

(2) recording-ft8.c (try2) **send_control(0x06): RX2 1st, send_control(0x04): RX1 2nd**

```
memset(commands,0,256); // initialise the commands.

commands[0x00] = 0x00000006;
commands[0x06] = freqArray[0];        //f1 to rx2

send_control(0x06);
commands[0x00] = 0x00000000;
send_control(0x00);
usleep(100000);

if (NRX > 1) {
      commands[0x00] = 0x00000008;
      commands[0x04] = freqArray[1];      //f2 to rx1

      send_control(0x04);
 }

if (NRX > 2) {
      commands[0x00] = 0x00000010;
      commands[0x08] = freqArray[2];      //f3

      send_control(0x08);
}
```

Log (NRX=2) and Result (2)

```
$ sudo ./recording-ft8

Radioberry gateware version 72-5.
RB-Command = 01 Command = 06  command_data = 006BF6AC  // RX2(40m) NG
RB-Command = 01 Command = 00  command_data = 00000000
usleep(100000);
RB-Command = 01 Command = 04  command_data = 00D6C66C  // RX1(20m) OK
RB-Command = 01 Command = 14  command_data = 0000005F
RB-Command = 01 Command = 00  command_data = 00000008
usleep(100000);

FT8-recording process...
FT8 recording timer started.
.............................................................
```

(3) recording-ft8.c (try3) **send_control(0x06): RX1 <- f2 1st, send_control(0x04): RX2 <- f1 2nd**

```
memset(commands,0,256); // initialise the commands.

commands[0x00] = 0x00000008;
commands[0x04] = freqArray[1];        //f2  to rx1
send_control(0x04);
commands[0x00] = 0x00000000;
send_control(0x00);
usleep(100000);

if (NRX > 1) {
      commands[0x00] = 0x00000008;
      commands[0x06] = freqArray[0];      //f1 to rx2

      send_control(0x06);
}

if (NRX > 2) {
      commands[0x00] = 0x00000018;
      commands[0x08] = freqArray[2];      //f3

      send_control(0x08);

}
```

Log (NRX=2) and Result (3)

```
$ sudo ./recording-ft8

Radioberry gateware version 72-5.
RB-Command = 01 Command = 04  command_data = 006BF6AC  // RX1(40m) OK
RB-Command = 01 Command = 00  command_data = 00000000
usleep(100000);
RB-Command = 01 Command = 06  command_data = 00D6C66C  // RX2(20m) NG
RB-Command = 01 Command = 14  command_data = 0000005F
RB-Command = 01 Command = 00  command_data = 00000008
usleep(100000);

FT8-recording process...
FT8 recording timer started.
.............................................................
```

(4) recording-ft8.c (try4) **send_control(0x06): RX2 <- f2 1st, send_control(0x08): RX3 <- f1 2nd**

_int32_t freqArray[8] = { 7075500, 14075500, 18101500,

```
memset(commands,0,256); // initialise the commands.

commands[0x00] = 0x00000008;
commands[0x06] = freqArray[1];          //f2 to rx2
send_control(0x06);
commands[0x00] = 0x00000000;
send_control(0x00);
usleep(100000);

if (NRX > 1) {
      commands[0x00] = 0x00000008;
      commands[0x08] = freqArray[0];      //f1 to rx3

      send_control(0x08);
}

if (NRX > 2) {
      commands[0x00] = 0x00000018;
      commands[0x08] = freqArray[2];      //f3

      send_control(0x08);

}
```

Log (NRX=2) and Result (4)

```
$ sudo ./recording-ft8

Radioberry gateware version 72-5.
RB-Command = 01 Command = 06  command_data = 006BF6AC  // RX2 (40m) OK
RB-Command = 01 Command = 00  command_data = 00000000
usleep(100000);
RB-Command = 01 Command = 08  command_data = 00D6C66C  // RX3 (20m) NG
RB-Command = 01 Command = 14  command_data = 0000005F
RB-Command = 01 Command = 00  command_data = 00000008
usleep(100000);

FT8-recording process...
FT8 recording timer started.
.............................................................
```

(5) recording-ft8.c (try5) **send_control(0x06): RX2 <- f1 1st, send_control(0x08): RX3 <- f2 2nd**

```
memset(commands,0,256); // initialise the commands.

commands[0x00] = 0x00000008;
commands[0x06] = freqArray[0];          //f1 to rx2
send_control(0x06);
commands[0x00] = 0x00000000;
send_control(0x00);
usleep(100000);

if (NRX > 1) {
      commands[0x00] = 0x00000008;
      commands[0x08] = freqArray[1];      //f2 to rx3

      send_control(0x08);
}
```

Log (NRX=2) and Result (5)

```
$ sudo ./recording-ft8

Radioberry gateware version 72-5.
RB-Command = 01 Command = 06  command_data = 00D6C66C // RX2 (20m) NG
RB-Command = 01 Command = 00  command_data = 00000000
usleep(100000);
RB-Command = 01 Command = 08  command_data = 006BF6AC // RX3 (40m) OK
RB-Command = 01 Command = 14  command_data = 0000005F
RB-Command = 01 Command = 00  command_data = 00000008
usleep(100000);

FT8-recording process...
FT8 recording timer started.
.............................................................
```

Summary:

(1)

Order	RX1 [f1]	RX2 [f2]
Result	NG	OK

(3)

Order	RX1 [f2]	RX2 [f1]
Result	OK	NG

(5)

Order	RX2 [f1]	RX3 [f2]
Result	NG	OK

(4)

Order	RX2 [f2]	RX3 [f1]
Result	OK	NG

(2)

Order	RX2 [f1]	RX1 [f2]
Result	NG	OK

Conclusion(2)

It's not a problem of the timing of the first RX1 command. From result (1)
It's not only RX1 problem, RX2 or RX3... can be the problem. From result (2-4)
The result varies depending on the order of freqArrey!

Consideration.(2)

The command timing may not be a problem.
This problem seems to be related to the way the RXn command is sent. (protocol?)
* I'm not sure if it is possible to skip RX1 and set it from RX2 or RX3, but it seems to work in my experiments.

// end sheet

3. Reference

(1) Radioberry-2.x/SBC/rpi-4/decoders/ft8/recording-ft8.c

The original excerpts are as follows

```
/*
        FT8 Recorder.


        Johan Maas PA3GSB
*/
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
  :
  :
#include "radioberry_ioctl.h"

// NRX defines the number of channels. CL025 users can select 8 and CL016 can select 4 channels
#define NRX 8

// FT8 Frequencies
u_int32_t freqArray[8] = {1841500, 3574500, 5358500, 7075500, 10137500,  14075500, 18101500,  21075500  };
  :
  :
void send_control(unsigned char command)  {

    u_int32_t command_data = commands[command];

    rb_info.rb_command = 0x01;
    rb_info.command = command;
    rb_info.command_data = command_data;

    fprintf(stderr, "RB-Command = %02X Command = %02X  command_data = %08X¥n", rb_info.rb_command, command, comm

    if (ioctl(fd_rb, RADIOBERRY_IOC_COMMAND, &rb_info) == -1) {
        fprintf(stderr, "Could not sent commando to radioberry device.");
    }
 }

int initDecoder()
{
    if((fd_rb = open("/dev/radioberry", O_RDWR)) < 0){
        perror("Failed to open /dev/radioberry");
        exit(-1);
    }

    rb_info.rb_command = 0x01;

    //required to retrieve gateware information.
    if (ioctl(fd_rb, RADIOBERRY_IOC_COMMAND, &rb_info) == -1) {
        fprintf(stderr, "RADIOBERRY_IOC_COMMAND Error.");
        exit(-1);
    }
    fprintf(stderr, "Radioberry gateware version %d-%d.¥n", rb_info.maior, rb_info.minor);

    memset(commands,0,256); // initialise the commands.

    commands[0x00] = 0x00000000;                        <- [0x00] = 48kHz, Dup off, Set number of RX 1, ···)
    commands[0x04] = freqArray[0];          //f1

    send_control(0x04);                                 <- ADDR(0x02) : Set RX1 NCO = freqArray[0]
    send_control(0x00);                                 <- ADDR(0x00) : 48kHz, Dup off, Set number of RX 1, ···)
    usleep(100000);

    if (NRX > 1) {
        commands[0x00] = 0x00000008;                    <- [0x00] = 48kHz, Dup off, Set number of RX 2, ···)
        commands[0x06] = freqArray[1];          //f2

            send_control(0x06);                         <- ADDR(0x03) : Set RX2 NCO = freqArray[1]
      }

      if (NRX > 2) {
            commands[0x00] = 0x00000010;                <- [0x00] = 48kHz, Dup off, Set number of RX 3, ···)
            commands[0x08] = freqArray[2];          //f3

            send_control(0x08);                         <- ADDR(0x04) : Set RX3 NCO = freqArray[2]
      }

      if (NRX > 3) {
            commands[0x00] = 0x00000018;                <- [0x00] = 48kHz, Dup off, Set number of RX 4, ···)
            commands[0x0A] = freqArray[3];          //f4

            send_control(0x0A);                         <- ADDR(0x05) : Set RX4 NCO = freqArray[3]
      }

      if (NRX > 4) {
            commands[0x00] = 0x00000020;                <- [0x00] = 48kHz, Dup off, Set number of RX 5, ···)
            commands[0x0C] = freqArray[4];          //f5

            send_control(0x0C);                         <- ADDR(0x06) : Set RX5 NCO = freqArray[4]
      }

      if (NRX > 5) {
            commands[0x00] = 0x00000028;                <- [0x00] = 48kHz, Dup off, Set number of RX 6, ···)
            commands[0x0E] = freqArray[5];          //f6

            send_control(0x0E);                         <- ADDR(0x07) : Set RX6 NCO = freqArray[5]
      }

      if (NRX > 6) {
            commands[0x00] = 0x00000030;                <- [0x00] = 48kHz, Dup off, Set number of RX 7, ···)
            commands[0x10] = freqArray[6];          //f7

            send_control(0x10);                         <- ADDR(0x08) : Set RX7 NCO = freqArray[6]
      }
```

```
        if (NRX > 7) {
                commands[0x00] = 0x00000038;                    <- [0x00] = 48kHz, Dup off, Set number of RX 8, ···)
                commands[0x24] = freqArray[7];          //f8

                send control(0x24);                             <- ADDR(0x12) : Set RX8 NCO = freqArray[7]
        }

        rb info.rb command = 0x01;
        commands[0x14] = 0x0000005f;            //att
        send control(0x14);                                     <- ADDR(0x0A) = Pure off, Set LNA Gain

        send control(0x00);                                     <- ADDR(0x00) : 48kHz, Dup off, Set  Number of RX, ···)
        usleep(100000);

        return 0;
}
  :
  :
void ft8 recording(void)
{
  :
  :
        fprintf(stderr, "¥nFT8-recording process... ¥n");

        iq buffer = (complex float*) malloc(sizeof(complex float)*240000 * NRX);        <- samp(60sec)*NRX

        while (!stopRecording) {

          if (sample ==0) {
                  time(&rawtime);
                  info = gmtime(&rawtime );

                  memset(iq buffer, 0.0, 240000 * 8 * NRX);             <- saml(=60sec)*8*NRX
          }
  :
  :


                if (sample !=0 && sample % 4000 == 0) fprintf(stderr, "."); // progress indi <- dot every second

                if (sample == 236000) {                                 <- if samp(=59sec) then output recording file

                        fprintf(stderr, "¥nFT8 recording done. now writing to c2 files.... ¥n");
  :
  :
}
  :
  :
void *decodeTiming(void *arg) {

        fprintf(stderr,"FT8 recording timer started. ¥n");

        while(1) {
                //wait till 59 sec in a minute are passed.
                struct timespec t;
                clock gettime(CLOCK REALTIME, &t);
                t.tv sec = 58 - t.tv sec % 60;
                t.tv nsec = 999999999L - t.tv nsec;
                nanosleep(&t, NULL);

                recording = 1;
        }
}

int main(int argc,char **argv)
{
        if (initDecoder()) exit (-1);
        signal(SIGINT, handle sigint);

        pthread t pid;
        pthread create(&pid, NULL, decodeTiming, NULL);

        ft8 recording();

   return 0;
}

// end sheet
```

Quoted : https://github.com/softerhardware/Hermes-Lite2/wiki/Protocol

(2) Protocol

mi0bot edited this page on 19 Feb · 66 revisions

This protocol is based on the original protocol1 from openHPSDR consisting of USB_protocol_V document and Metis. It is intended to remain compatible with a core subset of the openHPSDR protocol such that the Hermes-Lite2 may operate in basic mode with sta The Hermes-Lite2 will use Board_ID 0x06.

Data from PC to Hermes-Lite2

Discovery, Start, Stop

There is no change to the Metis Discovery packet <0xEFFE><0x02><60 bytes of 0x00>, but the Reply packet is extended. See below. The Metis Start packet is <0xEFFE><0x04>< Command><60 bytes of 0x00> where Command bit[0] starts the radio and bit[1] starts th

Interpretation of Original Protocol Command & Control

Command & Control	Bits	Description
C0	[7]	
	[6:1]	ADDR[5:0]
	[0]	MOX (1 = active, 0 = inactive)
C1	[7:0]	DATA[31:24]
C2	[7:0]	DATA[23:16]
C3	[7:0]	DATA[15:8]
C4	[7:0]	DATA[7:0]

Base Memory Map

This table shows the Hermes-Lite2 64 word memory map. These 64 addresses correspond to the first 64 addresses of the original openHPSDR's 128 address space. Since only 17 addresses are currently in use by the original openHPSDR protocol, no existing funti Please re to the original openHPSDR protocol when adding or repurposing locations. As of version 1.58, openHPSDR defines uses for addresses from 0x00 up to and including 0x11.

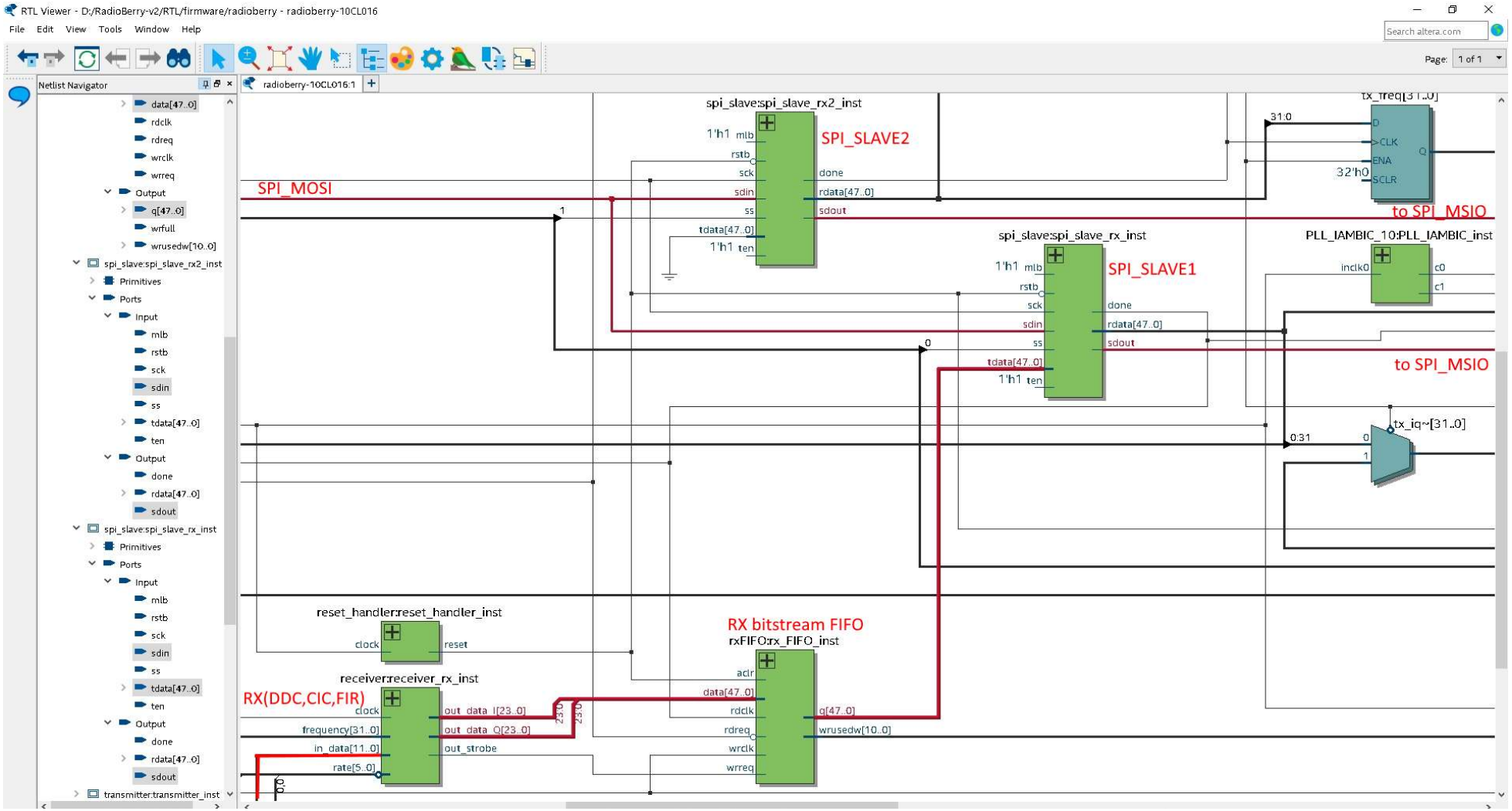
ADDR	DATA	Description
0x00	[25:24]	Speed (00=48kHz, 01=96kHz, 10=192kHz, 11=384kHz)
	[23:17]	openHPSDR Open Collector Outputs; see Filter Selection below
	[13]	openHPSDR Rx Antenna; see Filter Selection below
	[12]	FPGA-generated power supply switching clock (0=on, 1=off)
	[11]	Fan or Band Volts PWM (0=Fan, 1=Band Volts)
	[10]	VNA fixed RX Gain (0=-6dB, 1=+6dB)
	[6:3]	Number of Receivers (0000=1 to max 1011=12)
	[2]	Duplex (0=off, 1=on)
0x01	[31:0]	TX1 NCO Frequency in Hz
0x02 (04h)	[31:0]	RX1 NCO Frequency in Hz
0x03 (06h)	[31:0]	If present, RX2 NCO Frequency in Hz
0x04 (08h)	[31:0]	If present, RX3 NCO Frequency in Hz
0x05 (0Ah)	[31:0]	If present, RX4 NCO Frequency in Hz
0x06 (0Ch)	[31:0]	If present, RX5 NCO Frequency in Hz
0x07 (0Eh)	[31:0]	If present, RX6 NCO Frequency in Hz
0x08 (10h)	[31:0]	If present, RX7 NCO Frequency in Hz
0x09	[31:24]	Hermes TX Drive Level (only [31:28] used)
	[23]	VNA mode (0=off, 1=on)
	[22]	Alex manual mode (see Filter Selection below – Not Yet Implemented)
	[20]	Tune request: Set during TX spot or tune to initiate an ATU tune or bypass request
	[19]	Onboard power amplifier PA (0=off, 1=on)
	[18]	If the PA is Off, disable the T/R relay (1=antenna connector always Rx)
	[17]	For tune request: 1=send the bypass command; 0=send the normal tune request
	[15:8]	Alex Rx filter (see Filter Selection below); or VNA count MSB
	[7:0]	Alex Tx filter (see Filter Selection below); or VNA count LSB

0x0a (14h)	[22]	PureSignal (0=disable, 1=enable)
	[6]	See LNA gain section below
	[5:0]	LNA[5:0] gain
0x0e	[15]	Enable hardware managed LNA gain for TX
0x0e	[14]	See LNA gain section below
0x0e	[13:8]	LNA[5:0] gain during TX if enabled
0x0f	[24]	Enable CWX, I[0] of IQ stream is CWX keydown
0x10	[31:24]	CW Hang Time in ms, bits [9:2]
0x10	[17:16]	CW Hang Time in ms, bits [1:0]
0x12 (24h)	[31:0]	If present, RX8 NCO Frequency in Hz
0x13	[31:0]	If present, RX9 NCO Frequency in Hz
0x14	[31:0]	If present, RX10 NCO Frequency in Hz
0x15	[31:0]	If present, RX11 NCO Frequency in Hz
0x16	[31:0]	If present, RX12 NCO Frequency in Hz
0x17	[12:8]	PTT hang time, default is 12ms
0x17	[6:0]	TX buffer latency in ms, default is 20ms
0x2b	[31:24]	Predistortion subindex
0x2b	[19:16]	Predistortion
0x39	[27:24]	Misc Commands
		0x0 No command
		0x8 Enable watchdog timer
		0x9 Disable watchdog timer
0x39	[23]	Enable update of locked receivers
0x39	[21]	Lock RX12 to RX 11
0x39	[20]	Lock RX10 to RX 9
0x39	[19]	Lock RX8 to RX7
0x39	[18]	Lock RX6 to RX5
0x39	[17]	Lock RX4 to RX3
0x39	[16]	Lock RX2 to RX1
0x39	[11:8]	Master Commands
		0x0 No command
		0x8 Disable Master
		0x9 Enable Master
0x39	[7:4]	Synchronization Commands
		0x0 No command
		0x8 Reset all filter pipelines
		0x9 Reset and align all NCOs
0x39	[3:0]	Clock Generator Commands
		0x0 No command
		0x8 Synchronize clock outputs
		0xA Disable CL2 clock output
		0xB Enable CL2 clock output
		0xC Disable CL1 clock input
		0XD Enable CL1 clock input
0x3a	[0]	Reset HL2 on disconnect (0=no reset, 1=reset)
0x3b	[31:24]	AD9866 SPI cookie, must be 0x06 to write
0x3b	[20:16]	AD9866 SPI address
0x3b	[7:0]	AD9866 SPI data
0x3c	[31:24]	I2C1 cookie, must be 0x06 to write, 0x07 to read
0x3c	[23]	I2C1 stop at end (0=continue, 1=stop)
0x3c	[22:16]	I2C1 target chip address
0x3c	[15:8]	I2C1 control
0x3c	[7:0]	I2C1 data (only for write)
0x3d	[31:24]	I2C2 cookie, must be 0x06 to write, 0x07 to read
0x3d	[23]	I2C2 stop at end (0=continue, 1=stop)
0x3d	[22:16]	I2C2 target chip address
0x3d	[15:8]	I2C2 control
0x3d	[7:0]	I2C2 data (only for write)
0x3f	[31:0]	Error for responses

LNA Gain

When bit 6 at address 0x0a is set, then LNA LNA[5:0] is passed directly to the AD9866 for full −12dB (0) to +48dB (60) gain range. When bit 6 is not set, Hermes backwards compatibility is selected. Only gain levels from −12dB to +20dB are available. The v Additional Co

(3) ./firmware/rtl/radioberry.v



4. Tips

(1). Virtual file system:

Temporary files (FT8 recoding/decoding files) rewrite the SD memory card over and over again.
To avoid SD card wearing these files could be written to a virtual drive in RAM.

See.. https://www.domoticz.com/wiki/Setting_up_a_RAM_drive_on_Raspberry_Pi

```
$ sudo mkdir /var/tmp_recordings     <-- make ft8 recording directory
$ sudo mkdir /var/tmp_decodes        <-- make ft8 decoding results directory
$ sudo vi /etc/fstab
  - add the following two lines -
  tmpfs /var/tmp_recordings tmpfs nodev, nosuid, size=10M 0 0     <--  need abt. 2MB/1band
  tmpfs /var/tmp_decodes tmpfs nodev, nosuid, size=2M 0 0

$ sudo mount -a  (or Reboot)
$ df -h      <-- for check
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  4.9G   23G  18% /
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  8.6M  1.9G   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
tmpfs            10M     0   10M   0% /var/tmp_decodes      <-- ok
tmpfs             2M     0    2M   0% /var/tmp_recordings   <-- ok
/dev/mmcblk0p1  253M   54M  199M  22% /boot
tmpfs           383M  8.0K  383M   1% /run/user/1000
```

Modified files

```
decode-ft8.sh
Line7   <- ft8 recording directory
Line16  <- ft8 recording directory
Line23  <-   ft8 decoding results directory
Line27  <- ft8 recording directory

upload-ft8.sh
Line4   CALL=JA1xxx  <- UR CALL
Line5   GRID=PM95VQ  <-  UR GL
Line9   ANTENNA="DP" <- UR ANT
Line18  <-   ft8 decoding results directory
Line20  <-   ft8 decoding results directory
Line38  <-   ft8 decoding results directory
```

(2). NTP server

When the RPi is connected to the Internet, the time is automatically set by the NTP server.
It is important that the clock is correct for FT8 decoding.
I tried to specify the NTP server connection. Auto Synchronize periodically.

```
$ sudo vi /etc/systemd/timesyncd.conf      <- Modify
  (Add 2 lines below)
  NTP=ntp.nict.jp                          <- Specify NTP      (NICT: National Institute of Information and Communications Technology, Japan)
  FallbackNTP=time.google.com

$ sudo systemctl restart systemd-timesyncd <- Synchronize

$ systemctl status systemd-timesyncd       <- Check
```

```
● systemd-timesyncd.service - Network Time Synchronization
   Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; vendor preset: enabled)
  Drop-In: /usr/lib/systemd/system/systemd-timesyncd.service.d
           └─disable-with-time-daemon.conf
   Active: active (running) since Sat 2021-05-09 06:54:09 JST; 29s ago
     Docs: man:systemd-timesyncd.service(8)
 Main PID: 1773 (systemd-timesyn)
   Status: "Synchronized to time server for the first time [2001:df0:232:eea0::fff3]:123 (ntp.nict.jp)."
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/systemd-timesyncd.service
           └─1773 /lib/systemd/systemd-timesyncd

May 08 06:54:09 raspberrypi systemd[1]: Starting Network Time Synchronization...
May 08 06:54:09 raspberrypi systemd[1]: Started Network Time Synchronization.
May 08 06:54:09 raspberrypi systemd-timesyncd[1773]: Synchronized to time server for the first time [2001:df0:232:eea0::fff
lines 1-15/15 (END)
```

Modify for Synchronize periodically.

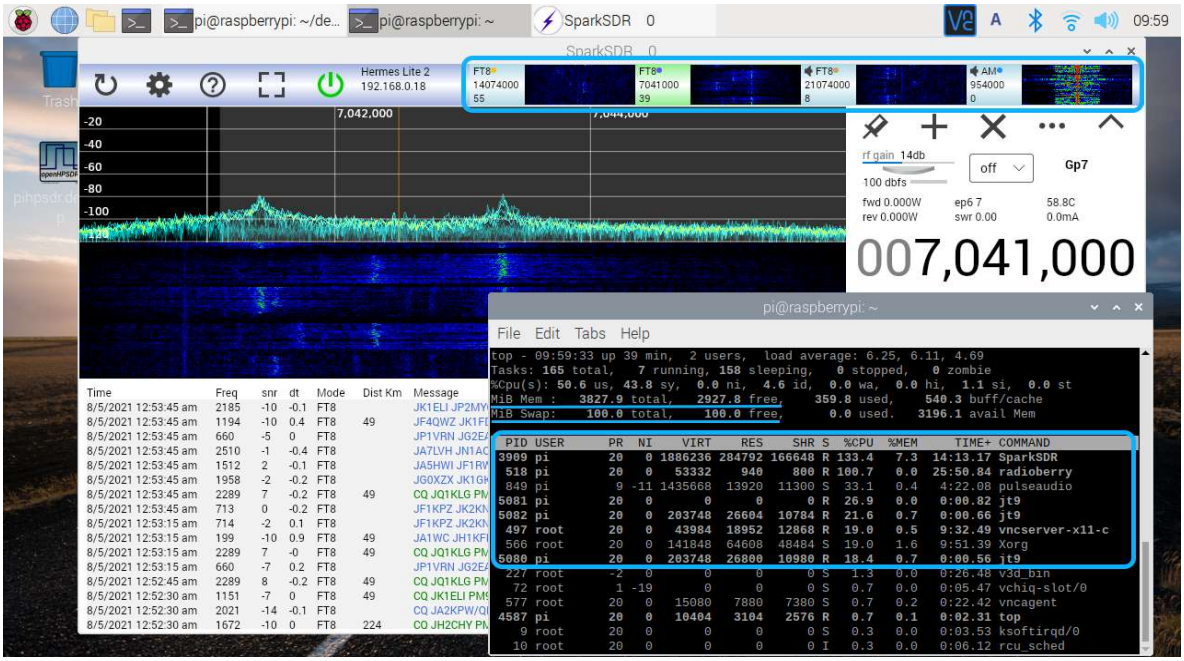
```
$ sudo crontab -e
  (Add a line below)
  45 * * * * systemctl restart systemd-timesyncd  <- Synchronize every ##:45

$ sudo service cron restart
  also useful command
  $ sudo service cron { status | stop | start }
```

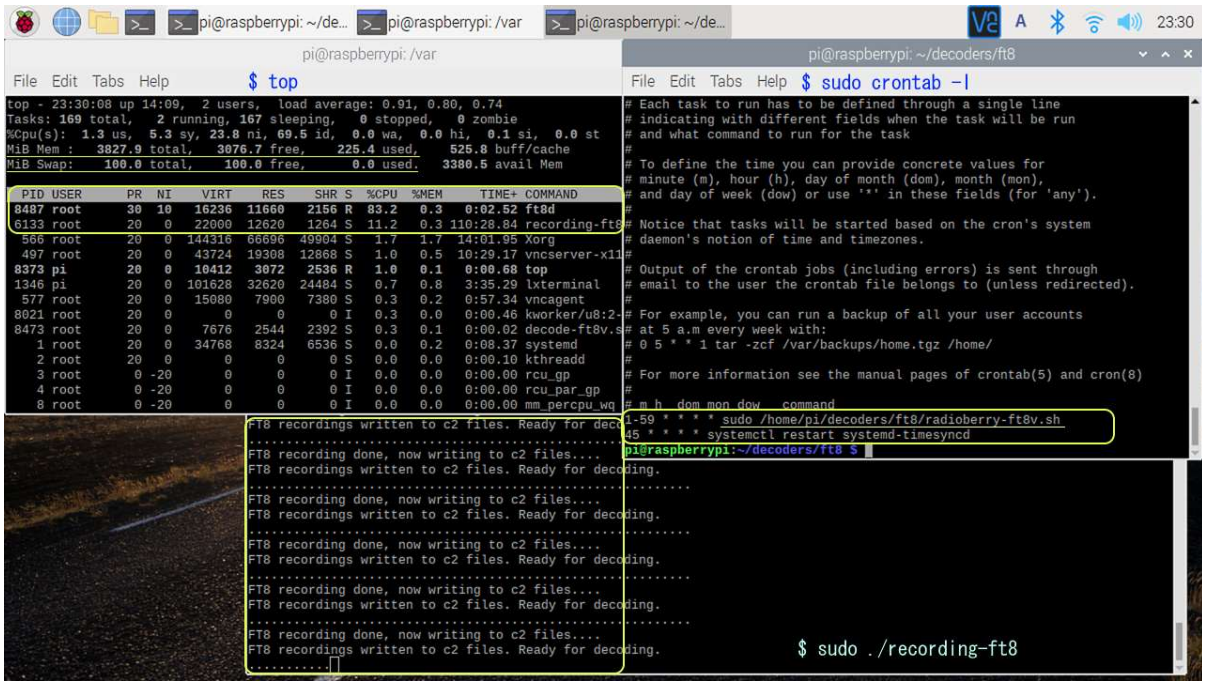
(3). SparkSDR on RPi3,4

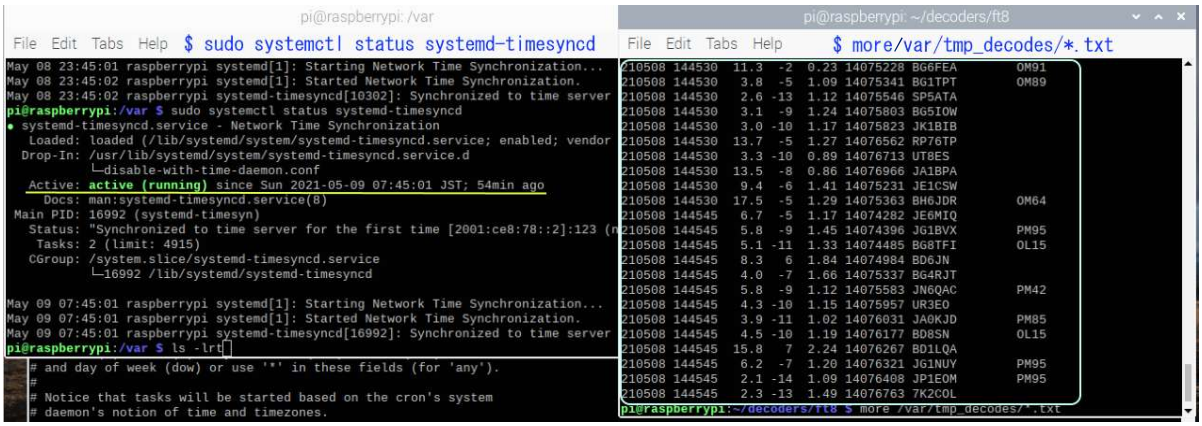
If Locale set to ja(Japanese) or ko(Korean) or zn(Chinese) or .. , SparkSDR crashes on Rpi3,4,
SparkSDR works if I set Locale to en(English) in Rpi4.
I didn't know what the root solution is..

SparkSDR 2.0.5.8 on Radioberry2 (RPi4B 4GB) 4*RX Fs:48kHz (too heavy to operate !)



(4). Snapshot of Radoberry FT8 Decoder running Nice working! (5bands FT8 decoding,)



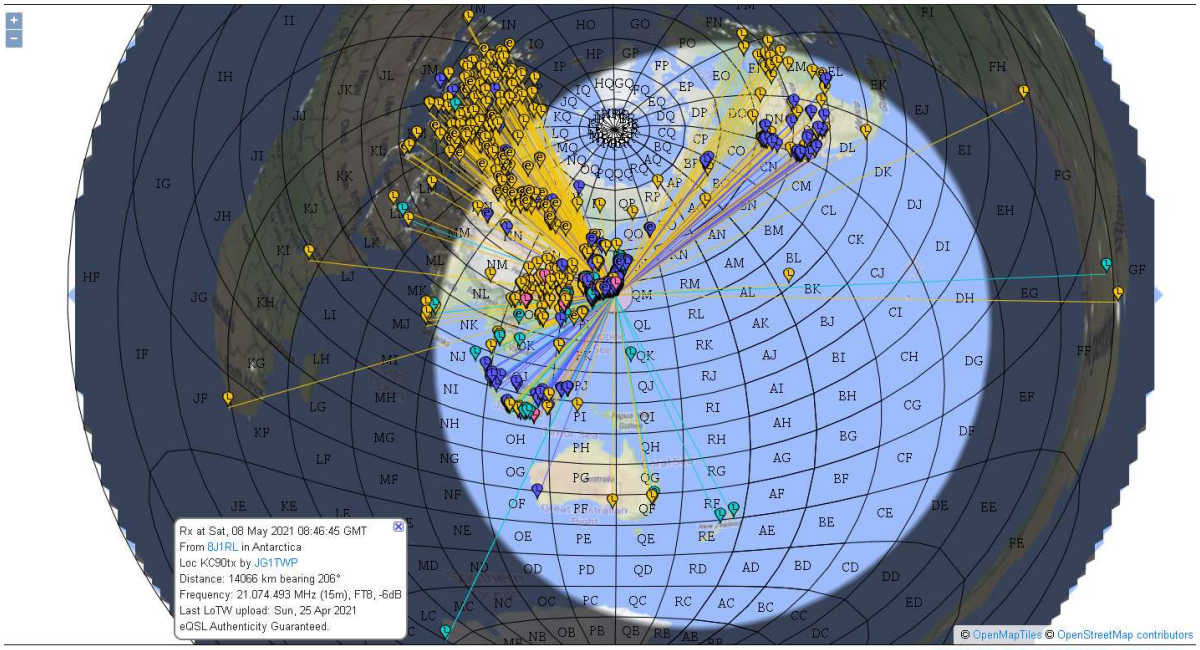


https://pskreporter.info/pskmap.html

On all bands, show signals sent/rcvd by the callsign jg1twp using all modes over the last 24 hours Go! Display options Permalink

Monitoring JG1TWP (last heard 3 days ago). Automatic refresh in 5 minutes. Small markers are the 1342 transmitters (show logbook) heard (distance chart) at JG1TWP (10365 reports, 67 countries last 24 hours; 10594 reports, 67 countries last week).

There are 5843 active monitors: 15 on 70cm, 2 on 23cm, 58 on 2m, 1 on 4m, 1005 on 6m, 293 on 10m, 4 on 11m, 94 on 12m, 317 on 15m, 619 on 17m, 1541 on 20m, 450 on 30m, 1027 on 40m, 77 on 60m, 189 on 80m, 14 on 160m, 4 on 600m, 67 on unknown, 8 on 2200m. Legend



Statistics — Comments to Philip Gladstone — Online discussions — Reception records: 20,011,745,026 (0/sec) — Hosting by Fast Serv Networks, LLC

PSKREPORTER.INFO

09 May 2021 JG1TWF

// end sheet