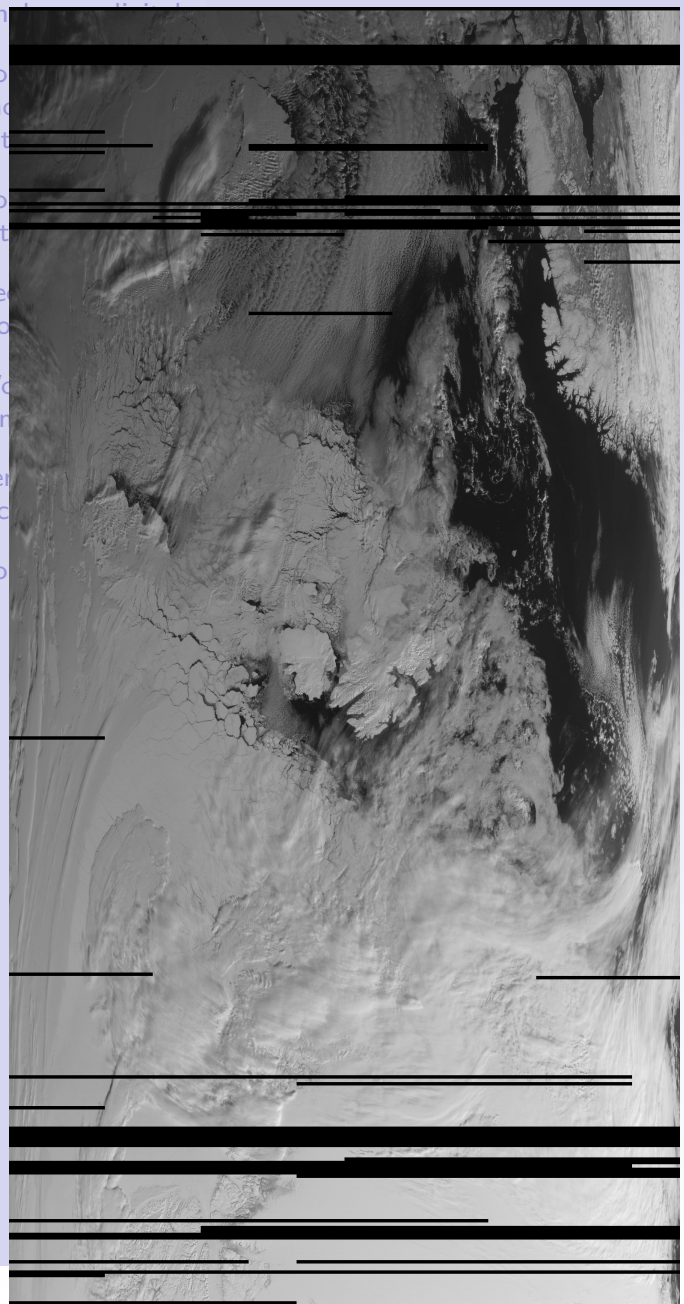
Decoding MeteorM2: QPSK, Viterbi, Reed Solomon and JPEG



from IQ coefficients to images, analysis of digital weather satellite transmissions

J.-M Friedt

FEMTO-ST Time & Frequency,
Besançon, France

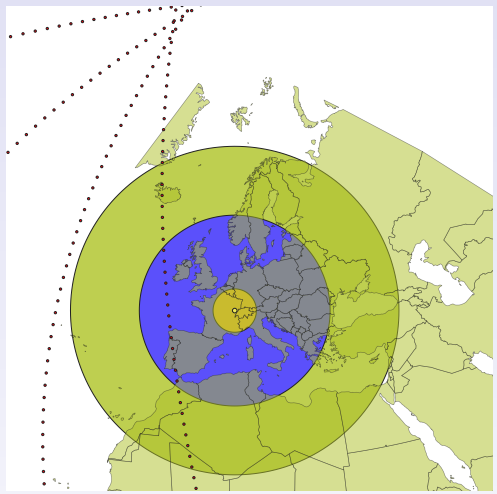
slides at
`jmfriedt.free.fr/fosdem2019_meteor.pdf`

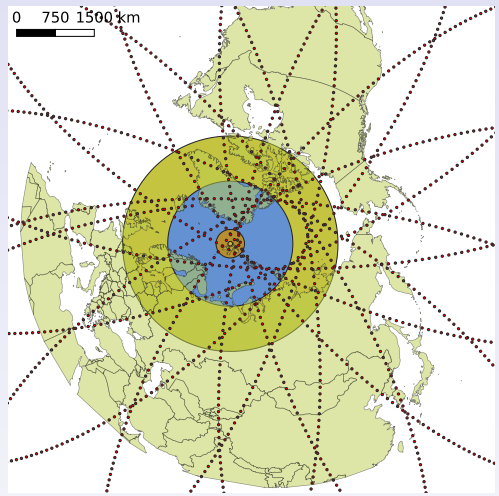
⟵ Image decoded using `medet` at
`github.com/artlav/meteor_decoder`

February 3, 2019

From NOAA to METEOR-M2N

- Low Earth Orbiting (LEO) satellite – polar sun-synchronous orbit (830 km, 98.6°, 101 min. period)
- Emission frequency: 137.9 MHz
- NOOA: noise and poorer resolution for a given spectral occupation



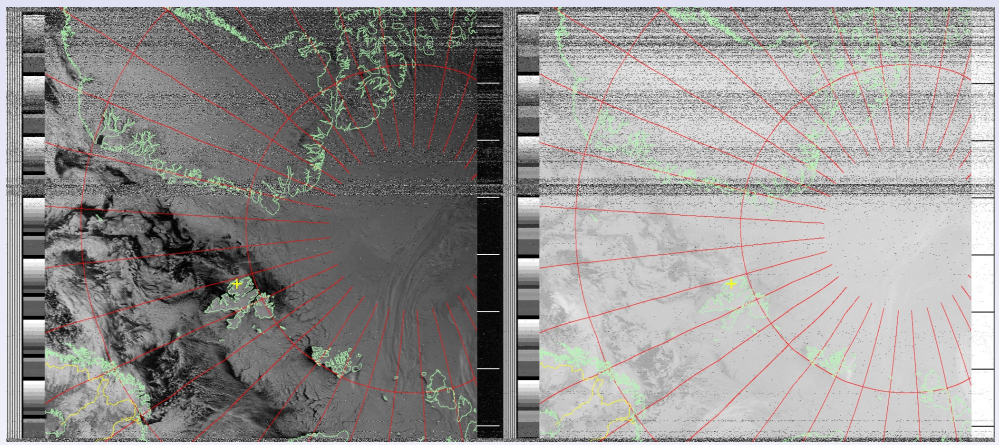


Celestrak TLE (`https://www.celestrak.com/NORAD/elements/weather.txt`):

```
1 40069U 14037A   18357.94082717 -.00000025  00000-0  80534-5 0  9999
2 40069  98.5783  43.7489 0006659  57.2213 302.9601 14.20657031231323
```

From NOAA to METEOR-M2N

- Low Earth Orbiting (LEO) satellite – polar sun-synchronous orbit (830 km, 98.6°, 101 min. period)
- Emission frequency: 137.9 MHz
- NOOA: noise and poorer resolution [1] for a given spectral occupation

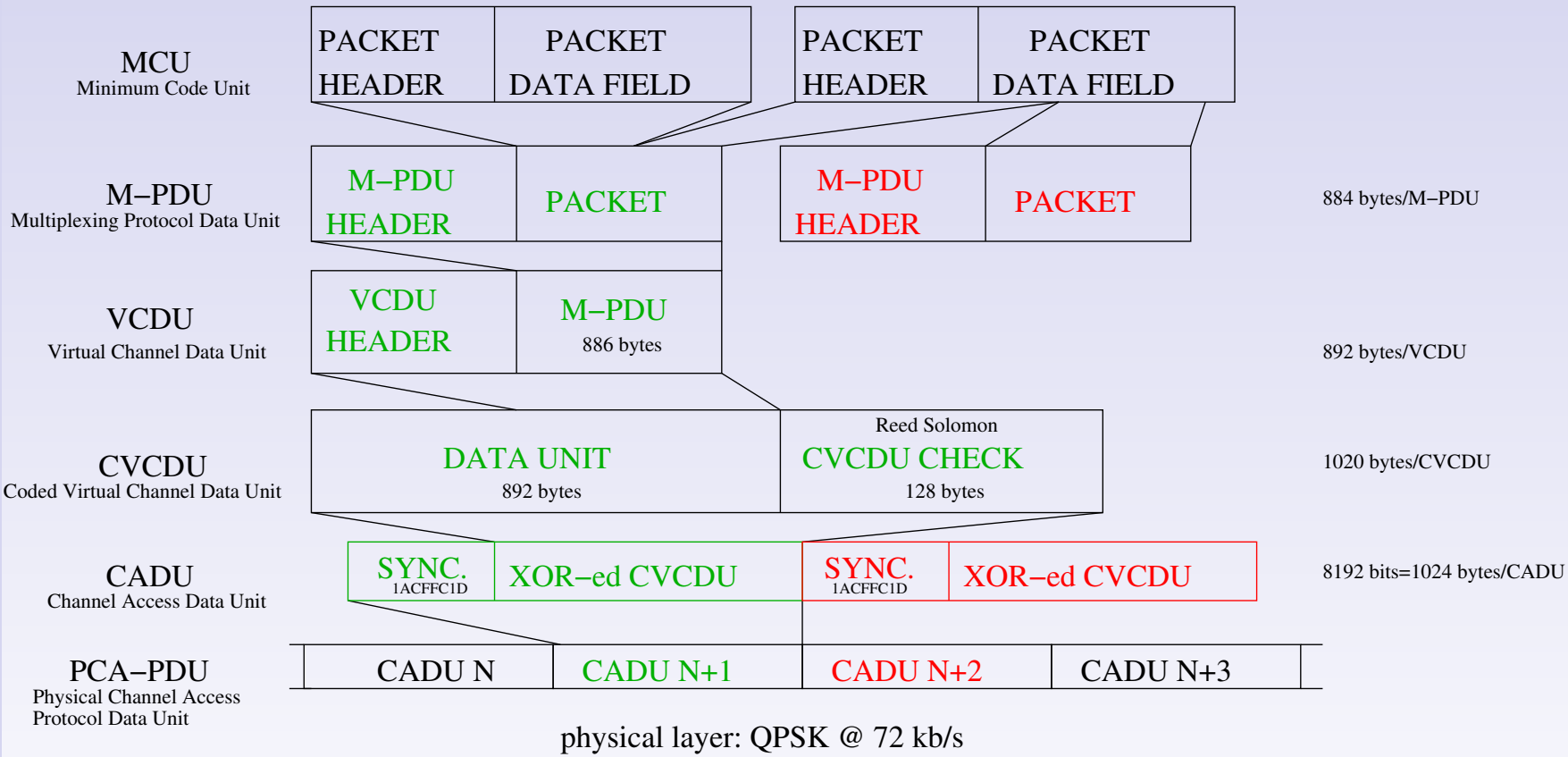


Celestrak TLE (`https://www.celestrak.com/NORAD/elements/weather.txt`):

```
1 40069U 14037A   18357.94082717 -.00000025  00000-0  80534-5 0  9999
2 40069  98.5783  43.7489 0006659  57.2213 302.9601 14.20657031231323
```

[1] J.-M Friedt, *Satellite image eavesdropping: a multidisciplinary science education project European Journal of Physics*, **26** 969–984 (2005)

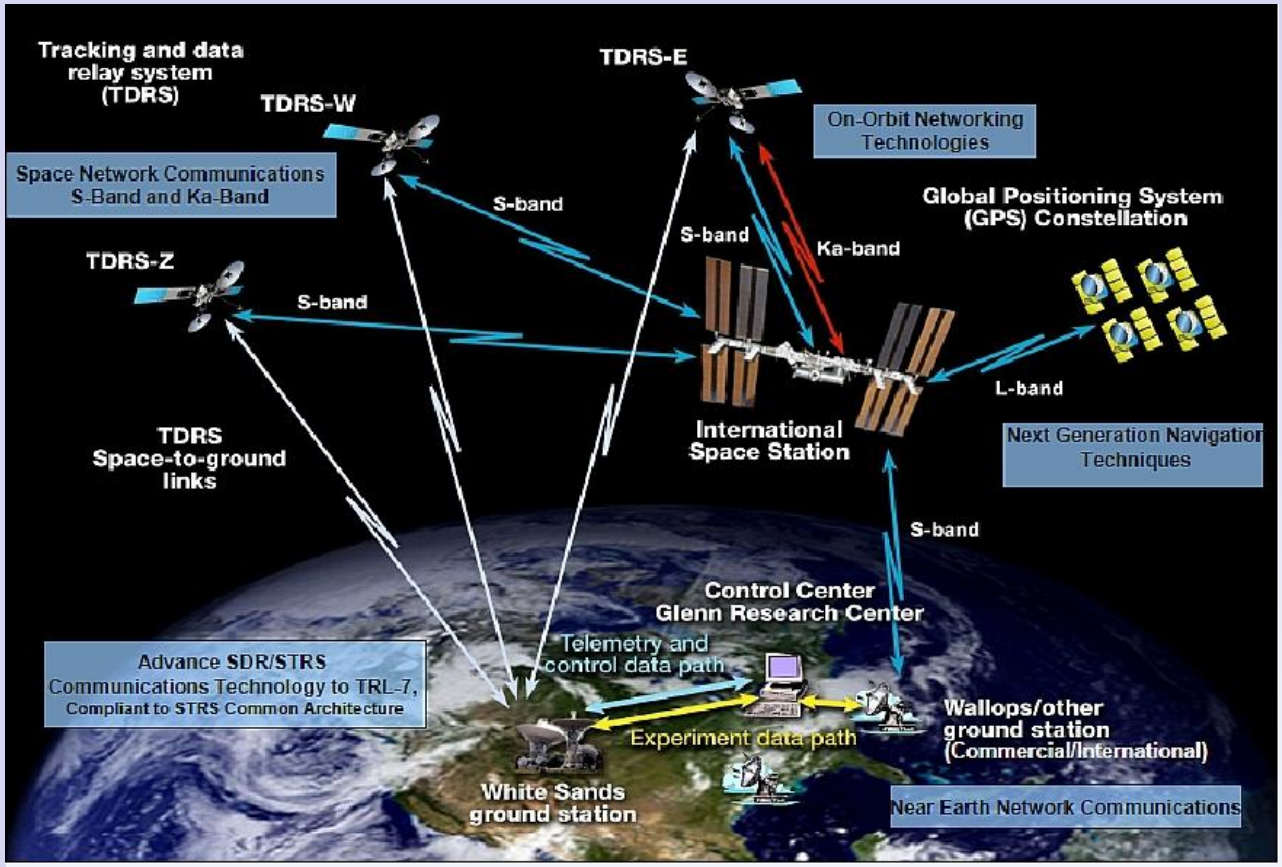
The big picture: data routing in space communication [2]

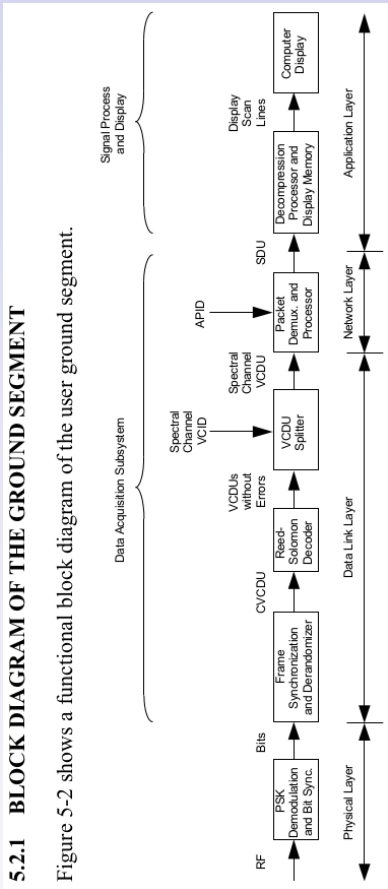


physical layer: QPSK @ 72 kb/s

[2] *Lossless Data Compression* CCSDS 120.0-G-2 (2006) at `public.ccsds.org/Pubs/120x0g2s.pdf` (p.5-2)

The big picture: data routing in space communication [2]





earth.esa.int/web/eoportal/satellite-missions/i/iss-scan

ISS @ 400 km is visible for 7 minutes at most from any ground-based receiver ⇒ continuous control and monitoring using TDRS

[2] *Lossless Data Compression* CCSDS 120.0-G-2 (2006) at public.ccsds.org/Pubs/120x0g2s.pdf (p.5-2)

Predicting M2N passes

SatTrack (patched with Y2K bug correction), WXtoImg (cheat by replacing NOAA TLE with Meteor M2), web-based Heavens Above

METEOR M2 - All Passes

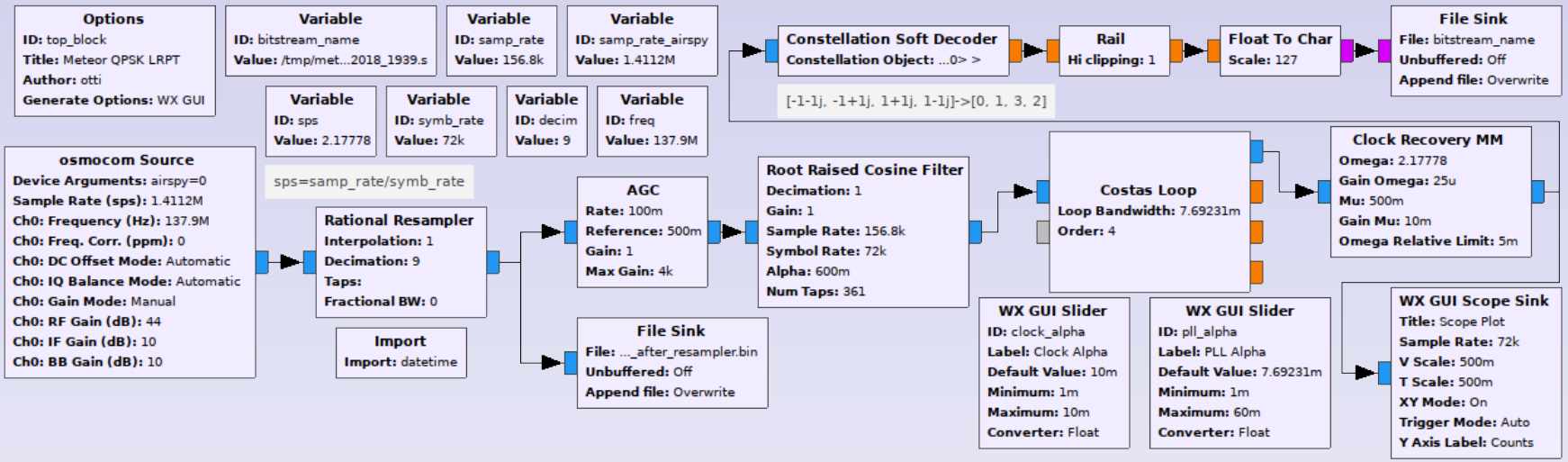
Search period start: 01 October 2018 00:00 < >
Search period end: 11 October 2018 00:00
Orbit: 820 x 826 km, 98.6° (Epoch: 30 September)

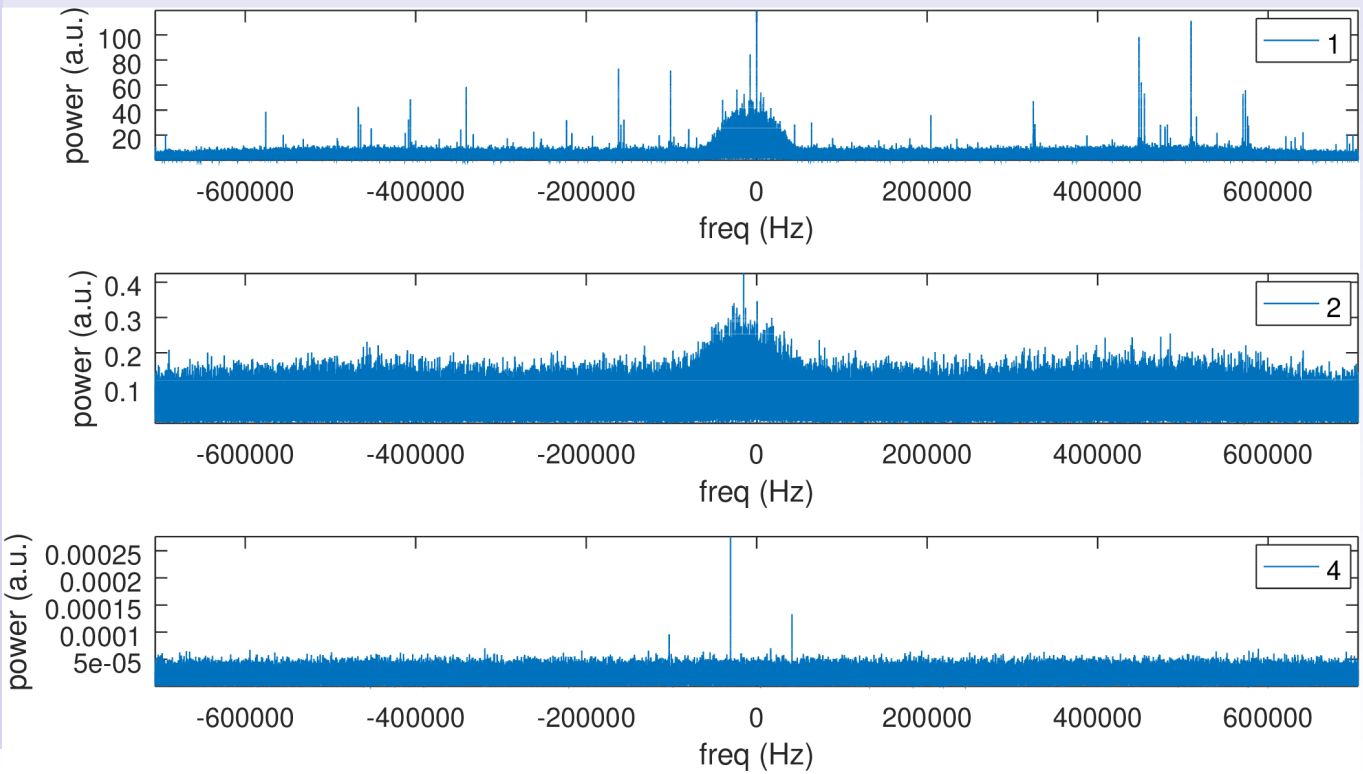
Passes to include: ○ visible only ◉ all

Click on the date to see the ground track during the pass.

Date	Brightness	Start			Highest point			End			Pass type
	(mag)	Time	Alt.	Az.	Time	Alt.	Az.	Time	Alt.	Az.	
01 Oct	-	00:20:53	10°	W	00:24:30	18°	WNW	00:28:08	10°	NNW	visible
01 Oct	-	02:04:58	10°	NW	02:06:59	12°	NNW	02:09:00	10°	N	visible
01 Oct	-	05:31:12	10°	N	05:32:44	11°	NNE	05:34:16	10°	NE	visible
01 Oct	-	07:12:02	10°	NNE	07:15:20	16°	NE	07:18:36	10°	E	daylight
01 Oct	-	08:52:54	10°	NNE	08:57:23	26°	ENE	09:01:53	10°	SE	daylight
01 Oct	-	10:33:40	10°	NNE	10:38:50	43°	E	10:44:00	10°	S	daylight
01 Oct	-	12:14:17	10°	NE	12:19:43	71°	SE	12:25:08	10°	SSW	daylight
01 Oct	-	13:54:41	10°	NE	14:00:07	81°	NNW	14:05:33	10°	WSW	daylight
01 Oct	-	15:34:52	10°	ENE	15:40:16	69°	N	15:45:39	10°	W	daylight
01 Oct	-	17:14:56	10°	ESE	17:20:20	74°	NNE	17:25:45	10°	WNW	daylight
01 Oct	-	18:55:10	10°	SE	19:00:36	85°	SW	19:06:03	10°	NW	daylight
01 Oct	-	20:35:56	10°	S	20:41:14	55°	WSW	20:46:35	10°	NW	visible

Data collection





$$I + jQ = e^{j(\Delta\omega t + \varphi)}$$

with

$$\varphi = \frac{n\pi}{2},\ n \in \mathbb{N}$$

$$\Rightarrow (I + jQ)^4 = e^{j(4\Delta\omega t + 4\varphi)}$$

$$e^{j(4\Delta\omega t + 0[2\pi])}$$

$$e^{j(4\Delta\omega t)}$$

Data collection: QPSK modulation

Did we collect useful information ?

- Constant block size ⇒ repetition of the synchronization header `1ACFFC1D` [3] [4]
- autocorrelation peaks every 16384 samples

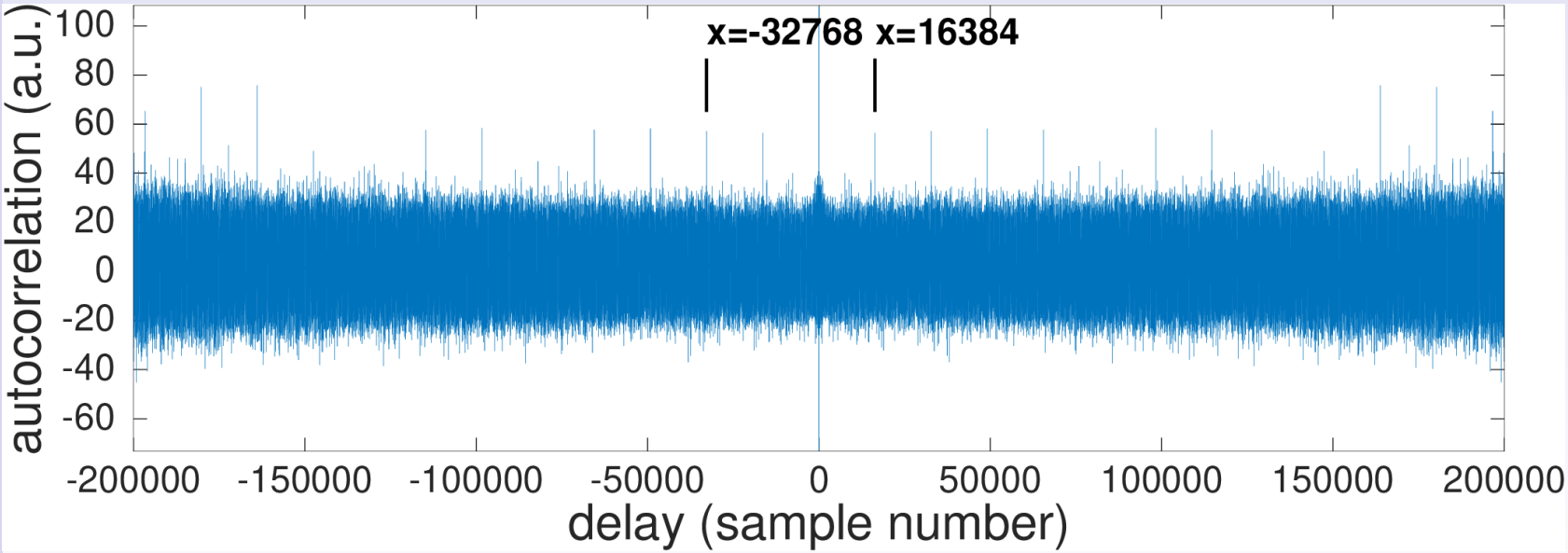
2.5.2.1 ***Bit Domain Frame Synchronization***

The Consultative Committee for Space Data Systems has adopted the 32-bit ASM shown in Figure 4 for synchronization in the bit domain. The pattern is represented in hexadecimal as 1ACFFC1D but any pattern having a length of 8 to 64 bits such as the Inter-range Instrumentation Group (IRIG) patterns can be accommodated.

1	A	C	F	F	C	1	D
0001	1010	1100	1111	1111	1100	0001	1101

FIRST TRANSMITTED BIT (BIT 0) — LAST TRANSMITTED BIT (BIT 31)

Figure 4. CCSDS Recommended 32-bit Attached Synchronization Marker

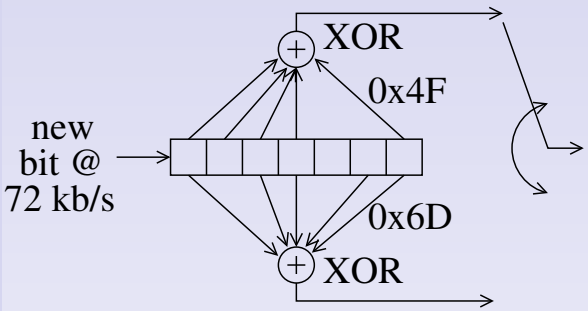


[3] `https://deepspace.jpl.nasa.gov/dsndocs/810-005/208/208B.pdf`

[4] `http://www.teske.net.br/lucas/satcom-projects/satellite-projects`

Viterbi convolution: encoding

Convolutional encoding to correct for flipped bits: various approaches to the same problem of introducing long term redundancy to correct for randomly distributed noise



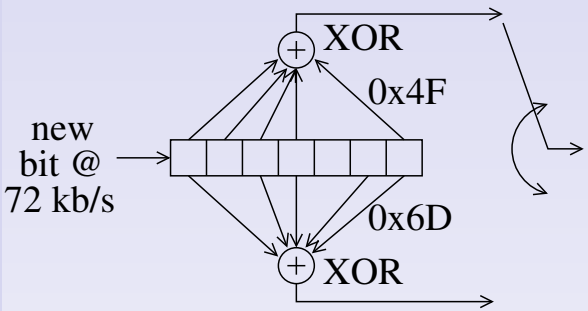

```
d=[0 1 0 1 1 1]
G1=[1 1 1]  % 0x7  r=1/2, K=3 (3-bit long shift register)
G2=[1 0 1]  % 0x5
G=[G1(1) G2(1) G1(2) G2(2) G1(3) G2(3)   0     0     0     0     0    0;
     0    0    G1(1) G2(1) G1(2) G2(2) G1(3) G2(3)   0     0     0    0;
     0    0       0    0   G1(1) G2(1) G1(2) G2(2) G1(3) G2(3)   0    0;
     0    0       0    0     0     0   G1(1) G2(1) G1(2) G2(2) G1(3) G2(3);
     0    0       0    0     0     0     0     0   G1(1) G2(1) G1(2) G2(2);
     0    0       0    0     0     0     0     0     0     0   G1(1) G2(1);
]
mod(d*G,2)  % matrix product modulo 2 = XOR
```

Input bit	Input state	Name	Output bits	Output state	Transition
0	000000	a	00	000000	a→a
1	000000	a	11	100000	a→b
0	100000	b	10	010000	b→c
1	100000	b	01	110000	b→d
0	010000	c	11	001000	c→...
1	010000	c	00	101000	c→...
0	110000	d	01	011000	d→e
1	110000	d	10	111000	d→...
0	011000	e	00	001100	e→...
1	011000	e	11	101100	e→f
0	101100	f	01	010110	f→...
1	101100	f	10	110110	f→...
...	...	...	...	...	...
0	100001	u	01	010000	u→c
1	100001	u	10	110000	u→d
...	...	...	...	...	...
0	000001	z	11	000000	z→a
1	000001	z	00	100000	z→b

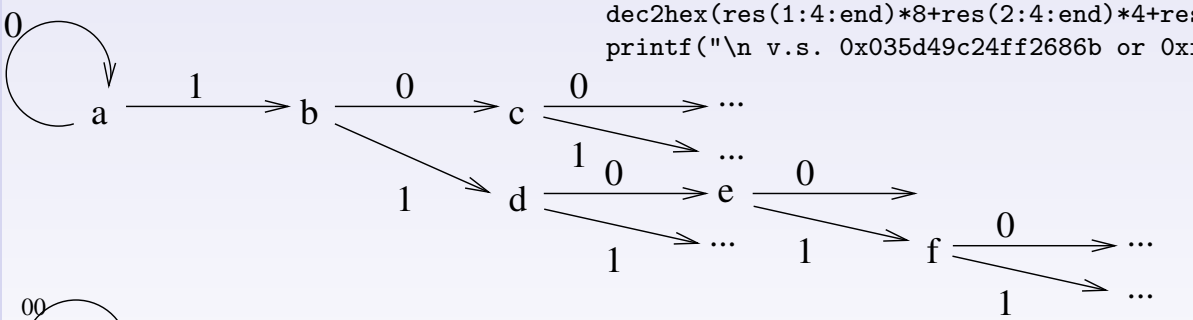
Two output bits for a single input bit during **encoding** with long term correlation.

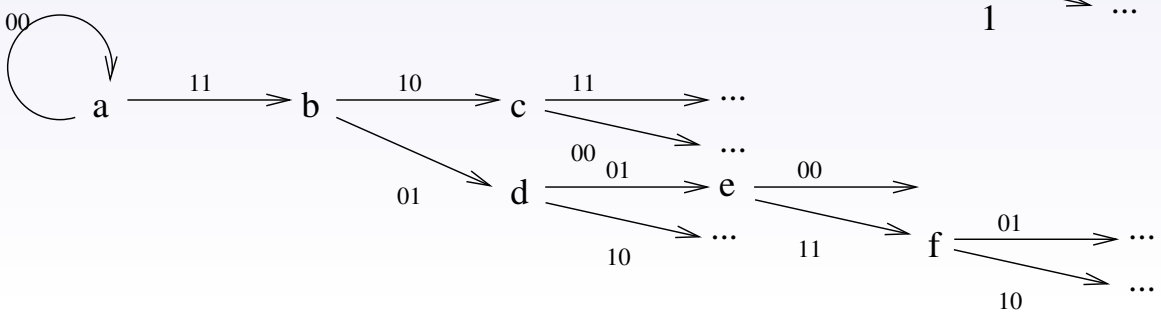
Viterbi convolution: encoding

Convolutional encoding to correct for flipped bits: various approaches to the same problem of introducing long term redundancy to correct for randomly distributed noise



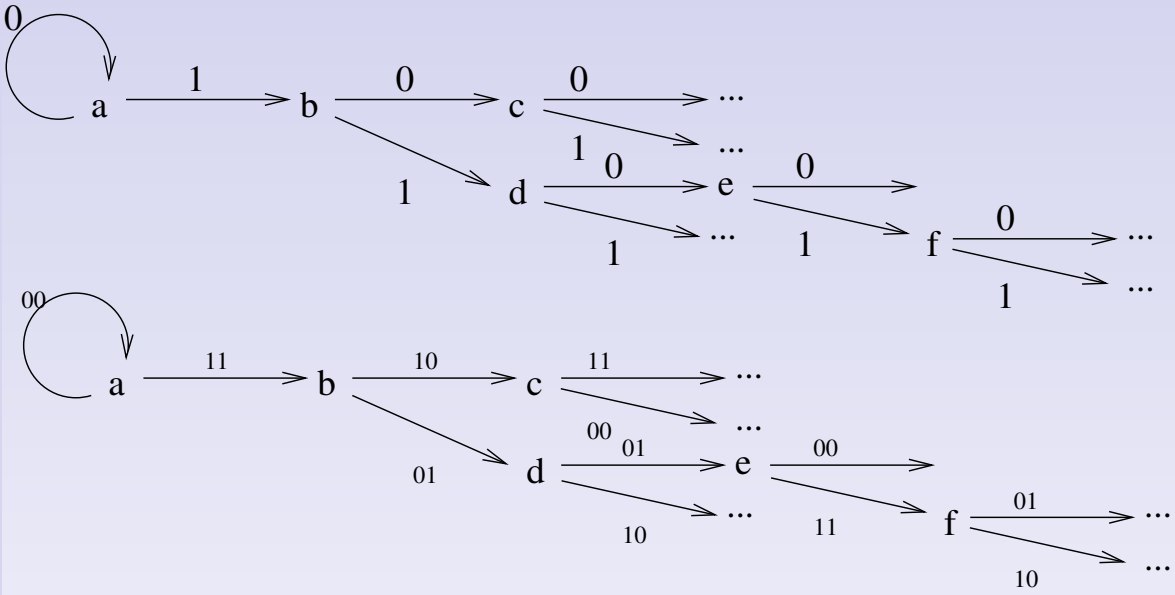

```
   % synchronization bytes to be encoded: 1A CF FC 1D
d=[0 0 0 1 1 0 1 0  1 1 0 0 1 1 1 1  1 1 1 1 1 1 0 0  0 0 0 1 1 1 0 1 ];
G1=[1 1 1 1 0 0 1 ] % 4F polynomial 1
G2=[1 0 1 1 0 1 1 ] % 6D polynomial 2
Gg=[];
for k=1:length(G1)
  Gg=[Gg G1(k) G2(k)]; % creates the interleaved version of the two
end                                              % generating polynomials
G=[Gg zeros(1,2*length(d)-length(Gg))] % 1t line of the convolution matrix
for k=1:length(d)-length(G1)
  G=[G ; zeros(1,2*k) Gg zeros(1,2*length(d)-length(Gg)-2*k)] ;
end
for k=length(Gg)-2:-2:2
  G=[G ; zeros(1,2*length(d)-(length(Gg(1:k)))) Gg(1:k)];
end     i               % last lines of the convolution matrix
res=1-mod(d*G,2);       % mod(d*G,2)
dec2hex(res(1:4:end)*8+res(2:4:end)*4+res(3:4:end)*2+res(4:4:end))'
printf("\n v.s. 0x035d49c24ff2686b or 0xfca2b63db00d9794\n")
```



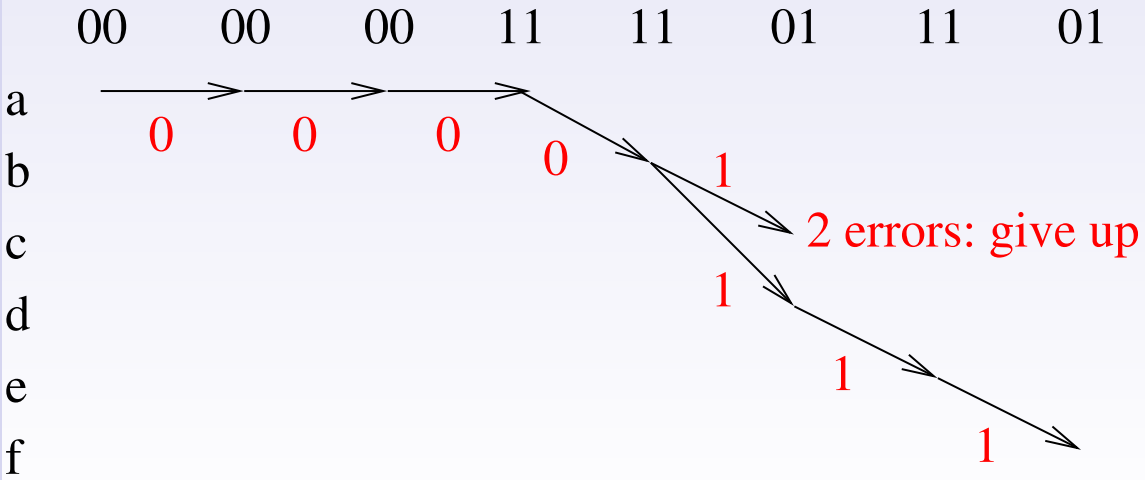


Viterbi convolution: decoding

From encoding state machine



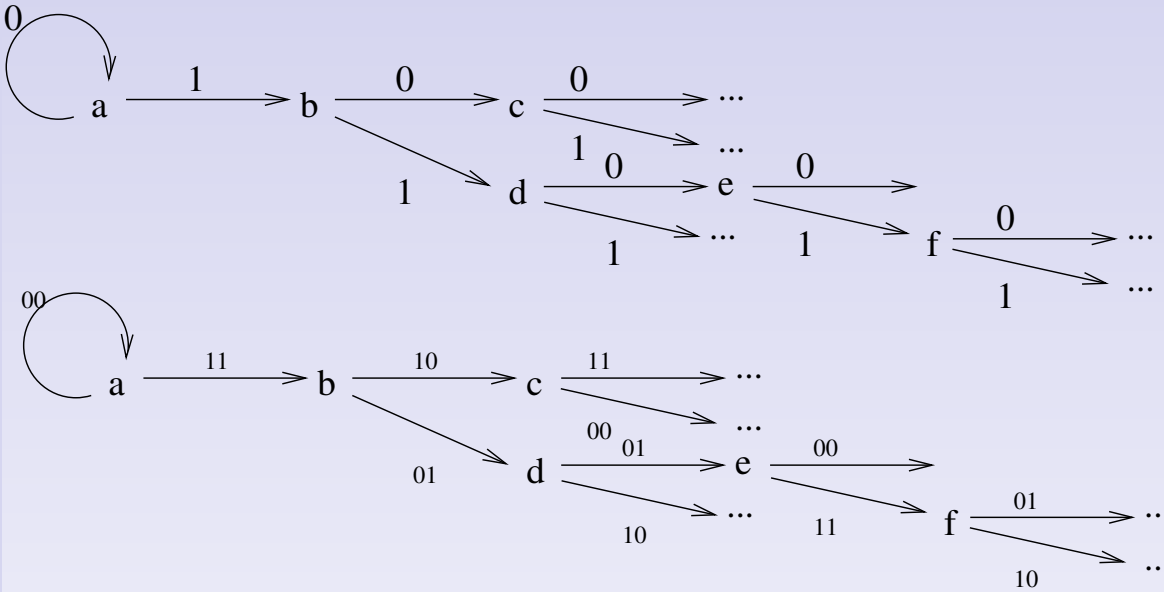
to decoding the (corrupted) received bit sequence 0b0000001111011101



(red: number of accumulated errors)

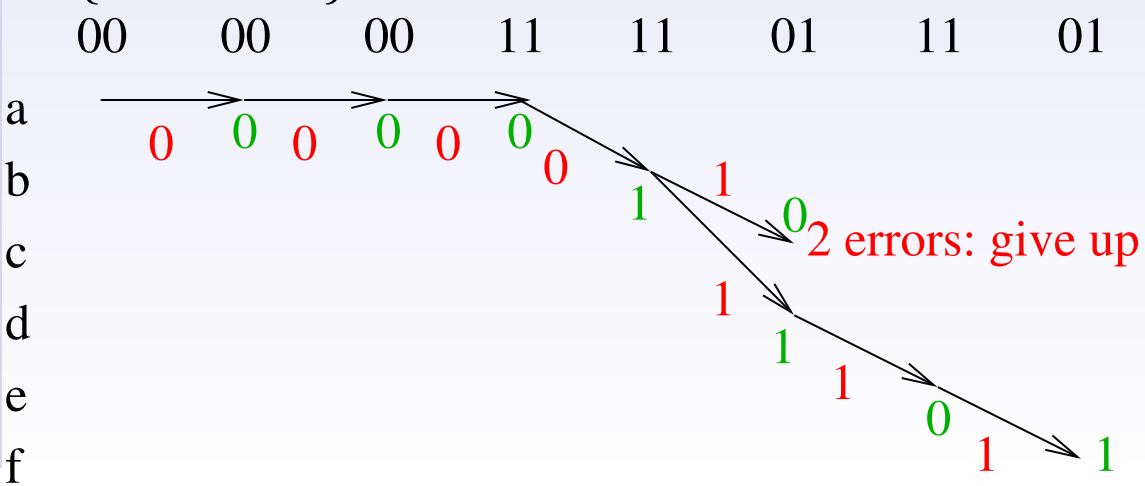
Viterbi convolution: decoding

From encoding state machine



to decoding the (corrected) received bit sequence 0b0000001101011101

→ {00011010}=0x1A



Viterbi convolution: `libfec`

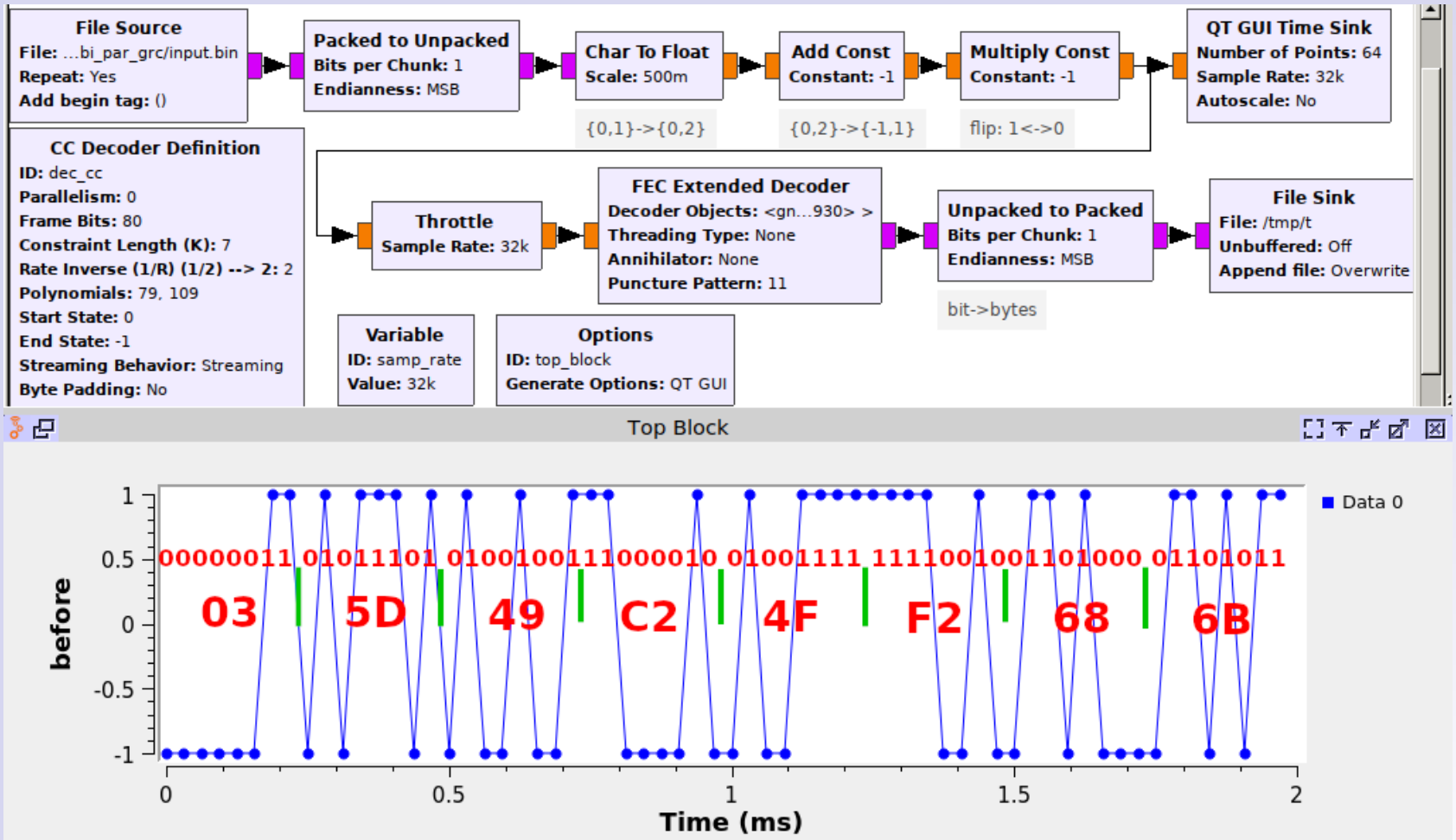
Decoding using libfec (`github.com/quiet/libfec`) the sentence encoded as `FC A2 B6 3D B0 0D 97 94` (resulting from the convolutional code encoding of the synchronization word)

```
#include <stdio.h>  // gcc -Wall -o t t.c -I./libfec ./libfec/libfec.a
#include <fec.h>
#define MAXBYTES (4)  // final message is 4-byte long

#define VITPOLYA 0x4F // 0d79  : polynomial 1 taps
#define VITPOLYB 0x6D // 0d109 : polynomial 2 taps

int viterbiPolynomial[2] = {VITPOLYA, VITPOLYB};
unsigned char symbols[MAXBYTES*8*2]=  // *8 for byte->bit, and *2 Viterbi
      {1,1,1,1,1,1,0,0, // fc
       1,0,1,0,0,0,1,0, // a2
       1,0,1,1,0,1,1,0, // b6
       0,0,1,1,1,1,0,1, // 3d
       1,0,1,1,0,0,0,0, // b0
       0,0,0,0,1,1,0,1, // 0d
       1,0,0,1,0,1,1,1, // 97
       1,0,0,1,0,1,0,0};// 94

int main(int argc,char *argv[]){
  int i,framebits;
  unsigned char data[MAXBYTES]; // *8 for bytes->bits & *2 Viterbi
  void *vp;
  framebits = MAXBYTES*8;
  for (i=0;i<framebits*2;i++) symbols[i]=1-symbols[i];    // flip bits
  for (i=0;i<framebits*2;i++) symbols[i]=symbols[i]*255; // bit -> byte /!\
  set_viterbi27_polynomial(viterbiPolynomial);    // definition of taps
  vp=create_viterbi27(framebits);
  init_viterbi27(vp,0);
  update_viterbi27_blk(vp,symbols,framebits+6);
  chainback_viterbi27(vp,data,framebits,0);
  for (i=0;i<MAXBYTES;i++) printf("%02hhX",data[i]);  // 1ACFFC1D is the right answer
  printf("\n");
}
```

Viterbi convolution: GNU Radio

Input must be $\in [-1 : 1]$

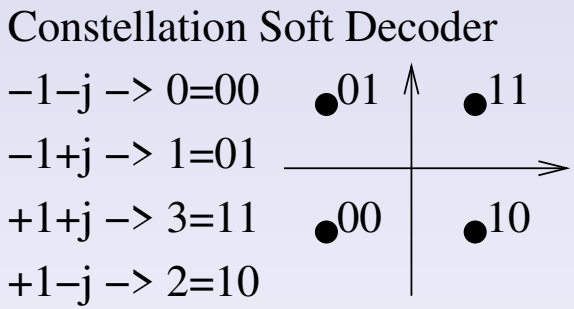
```
echo -e -n "\xfc\xa2\xb6\x3d\xb0\x0d\x97\x94" > input.bin
```



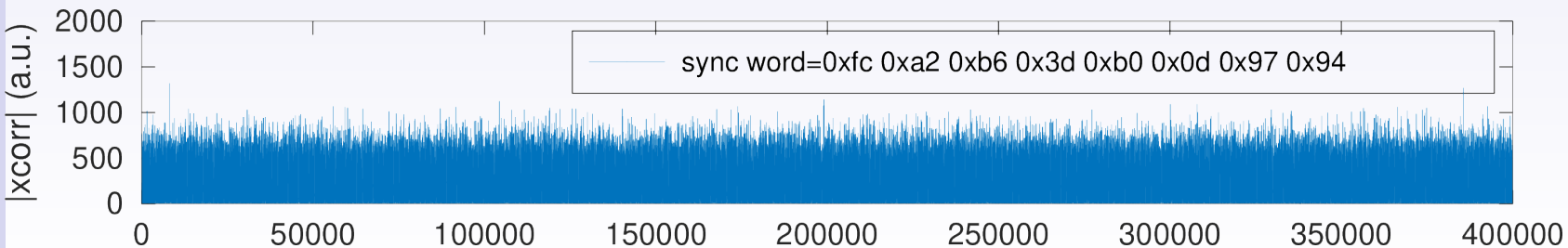

```
$xxd result.bin | cut -d: -f2
 1acf fc1d 1acf fc1d 0334 53c0 1acf fc1d  .........4S.....
```

Searching for the synchronization word

BPSK only has two options ($[0, \pi] \rightarrow [-1, 1]$ or $[1, -1]$) so that cross-correlation yields $+1$ or -1 when synchronization word is met
Encode `0x1acffc1d` using Viterbi: `0xfca2b63db00d9794` or `0x035d49c24ff2686b`

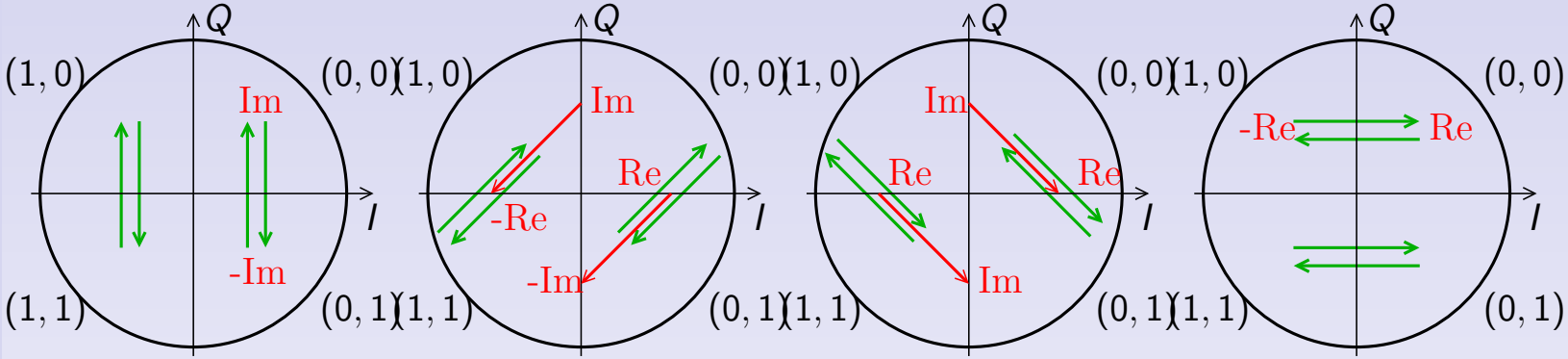


Assign each symbol to bit pairs and cross-correlate the encoded word to find frame synchronization marks



Searching for the sync. word

QPSK has four options with zero-correlation in case the symbols are not assigned properly → constellation rotation



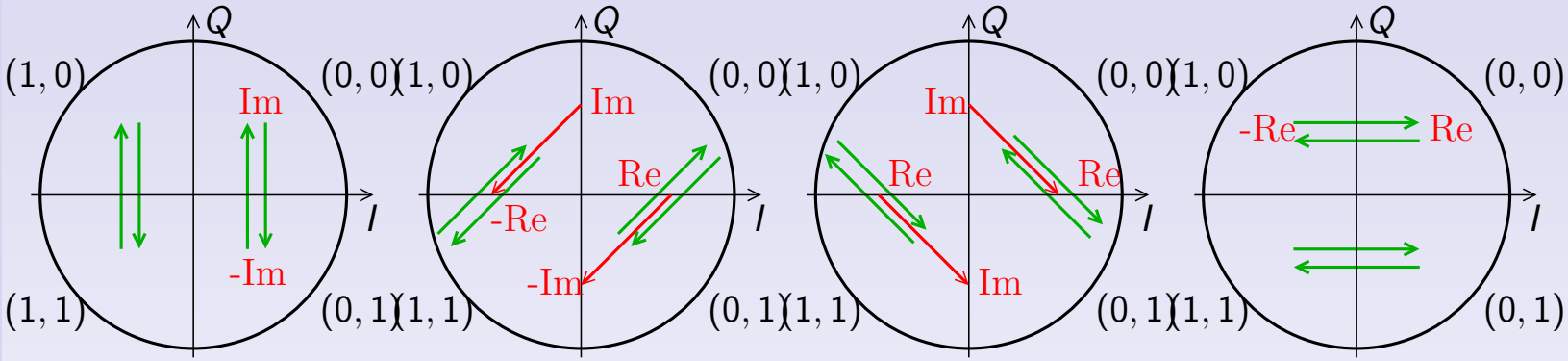

```
procedure process_file(fn,out_name:string;inp_mode,conv_mode:integer);
[...]
begin
 mj_init;
 mtd_init(m);
 for l:=0 to pattern_cnt-1 do begin          // debug: display
   for k:=0 to pattern_size-1 do begin       // correlation
       write(round(m.c.patts[k][l]/255));    // words
   end;writeln();                            //
 end;
```

yields [5]

[5] https://github.com/artlav/meteor_decoder

Searching for the synchronization word

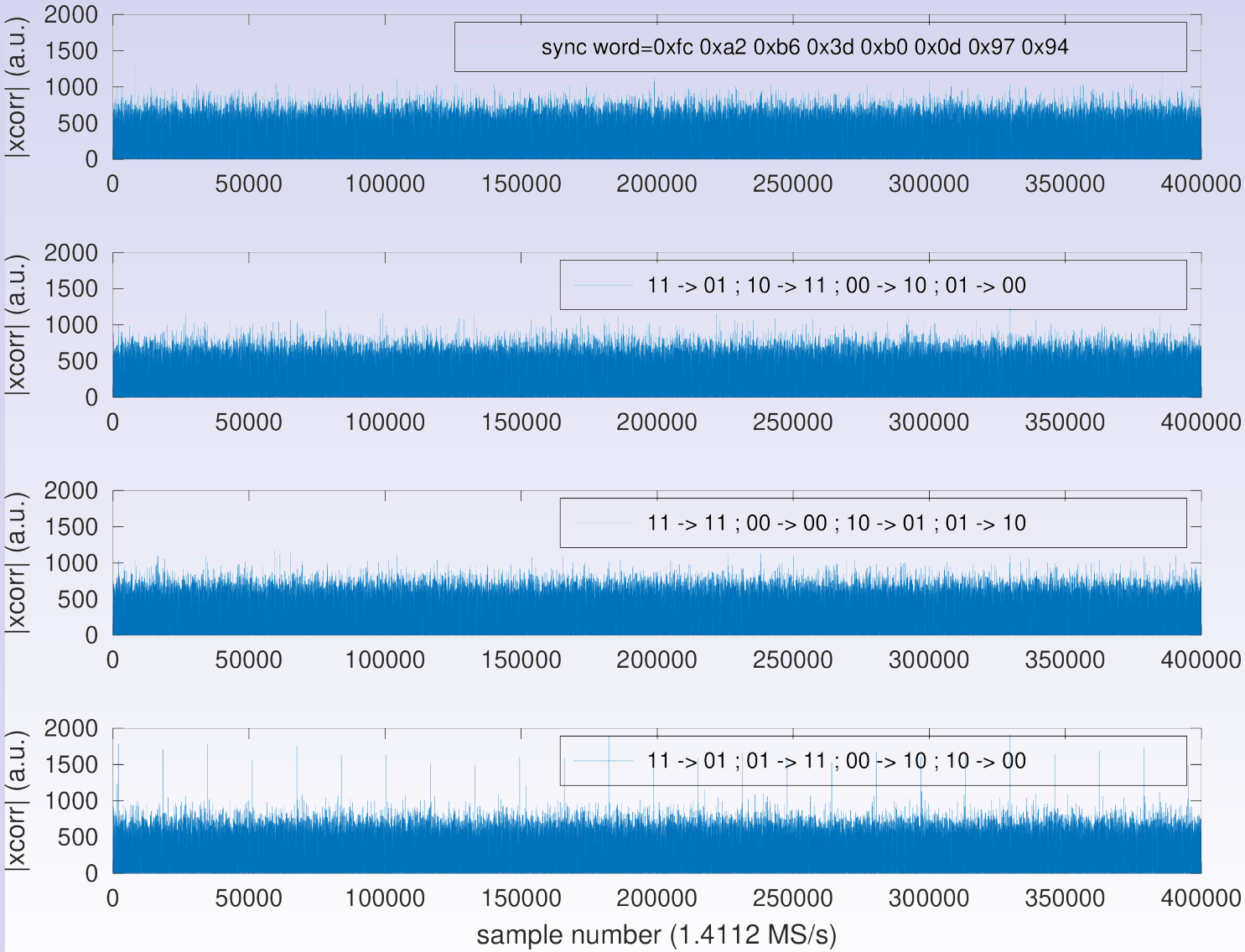
QPSK has four options with zero-correlation in case the symbols are not assigned properly → constellation rotation




```
11111100101000101011011000111101101100000000110110010111100101000
01010110111110111101001110010100110110101010010011000001110000100
00000011010111010100100111000010010011111111001001101000011010110
10101001000001000010110001101011001001010101101100111110001111010
11111100010100010111100100111110011100000000111001101011011010000
01010110000010000001110010010111000110101010011100111101001111100
00000011101011101000011011000001100011111111000110010100100101110
10101001111101111110001101101000111001010101100011000010110000010
```

8 possible options (divided by 2 since $|-1| = 1$)

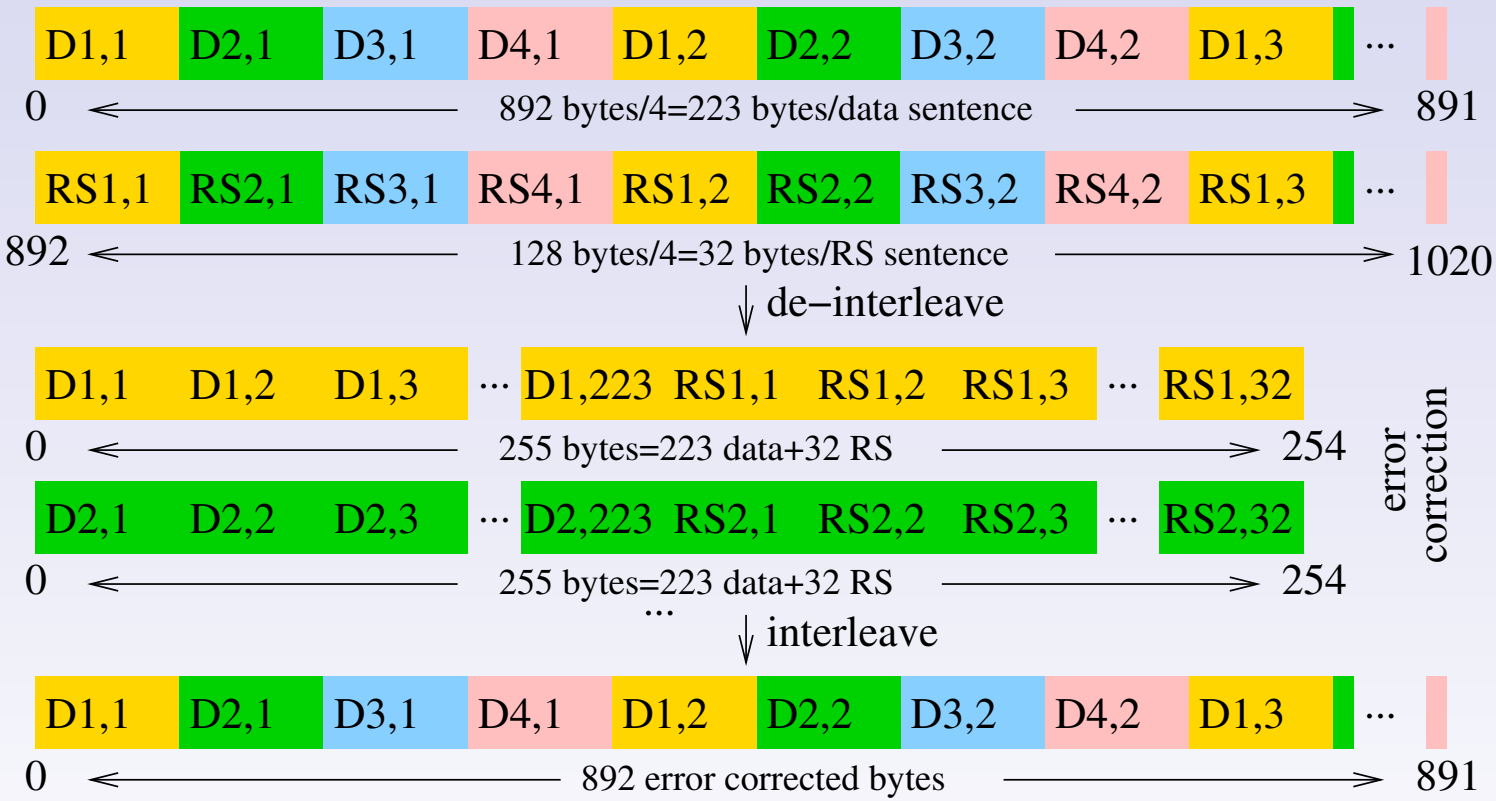
Searching for the synchronization word



Conclusion: the last bitswap provides the "right" solution

Reed Solomon (can be skipped)

Block encoding [6] to correct for noise bursts: CVCDU organization as 892 byte payload + 128 byte block correction



Interleaving to avoid repetition and maximize mixing

[6] RDS decoding: `http://jmfriedt.free.fr/lm_rds_eng.pdf` (BCH)

Reed Solomon (can be skipped)

Block encoding to correct for noise bursts

```c
#include <fec.h>    // gcc -o jmf_rs jmf_rs.c -I./libfec ./libfec/libfec.a

int main()
{int j;
 uint8_t rsBuffer[255];

 uint8_t tmppar[32];
 uint8_t tmpdat[223];

 for (j=0;j<255; j++) rsBuffer[j]=(rand()&0xff);  // received data
 for (j=0;j<223;j++)  tmpdat[j]=rsBuffer[j];      // backup data
 encode_rs_ccsds(tmpdat,tmppar,0);                // create RS code
 for (j=223;j<255;j++) rsBuffer[j]=tmppar[j-223]; // append RS after data
 rsBuffer[42]=42; tmpdat[42]=42;                  // introduce errors
 rsBuffer[43]=42; tmpdat[43]=42;                  // ... on purpose
 rsBuffer[44]=42; tmpdat[44]=42;    // ... to check correction capability
 rsBuffer[240]=42;tmppar[240-223]=42;
 printf("RS:%d\n",decode_rs_ccsds(rsBuffer, NULL, 0, 0)); // check that RS can correct
 for (j=0;j<223;j++)
  if (rsBuffer[j]!=tmpdat[j]) {printf("%d: %hhd->%hhd ; ",j,tmpdat[j],rsBuffer[j]);}
 for (j=223;j<255;j++)
  if (rsBuffer[j]!=tmppar[j-223]) {printf("%d: %hhd->%hhd ; ",j,tmppar[j-223],rsBuffer[j]);}
}
```

Encoding & decoding again implemented using `libfec`

```
RS:4
42: 42->5 ; 43: 42->23 ; 44: 42->88 ; 240: 42->95
```

Bits to frames: identifying the date/telemetry frame

Check that Viterbi decoding went well by searching for a known synchronization word

The contents of the user data field in a partial package with MSU-MR housekeeping data

A.1 The Structure of the user data is presented in table A1. All of the data are transmitting by the most significant bits forward.

Table A1

ByteNo	Stage of register								Note
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1...8	PRS-64 (in decimal mode: 2, 24, 167, 163, 146, 221, 154,191)								8 bytes
9	0	0	0	17d	16d	15d	14d	13d	Hours
10	0	0	12d	11d	10d	9d	8d	7d	Minutes
11	0	0	6d	5d	4d	3d	2d	1d	Seconds
12	The delay began of scanning line in relation to a second tag (weight of the least significant discharge is 4 ms)								

Searching for the synchronization words [7] in the decoded frames and displaying the time information yields **11 48 33** consistent with `medet` information **Onboard time: 11:48:33.788**

[7] `planet.iitp.ru/english/spacecraft/meteor_m_n2_structure_2_eng.htm`

Frames to sentences

```
octave:28> final(1:16,:)                                                                    20020081350.pdf NASA p.9 du PDF
    64    64    64    64    64    64    64    64    64    64    64    64    64    64  <- 01 00000000 VCDU version & Id
     5     5     5     5     5     5     5     5     5     5     5     5     5     5  <- 000101 VCDU Type
   140   140   140   140   140   140   140   140   140   140   140   140   140   140 \   pdf_ten_eps-metop-sp2gr.pdf p.14
   163   163   163   163   163   163   163   163   163   163   163   163   163   163 - VCDU counter (3bytes)
    43    44    45    46    47    48    49    50    51    52    53    54    55    56 /
     0     0     0     0     0     0     0     0     0     0     0     0     0     0 signaling field=0 for real time data
     0     0     0     0     0     0     0     0     0     0     0     0     0     0 VCDU insert ...  => encryption flag
     0     0     0     0     0     0     0     0     0     0     0     0     0     0 ... zone         => key number
     0     0     2     1     0     0     0     0     0     0     0     0     2     0 M_PDU header spare should be 5 bits
    18   142    28    54    18   130    78   226     0    20    70    28    82    32 M_PDU 1st header pointer 11 bits pdf
end of M-PDU header : next 882 bytes are M-PDU (Virtual channel field)                      |
    77   166   239   222    73    82    83   199     8    28   232   247   165   183    x=final(9,:)*256+final(10,:)+12 ;
   133   188   229    42    24    23   220    94    68   117    92   151    87   203    for k=1:length(x);final(x(k),k),(
    75   177   221   215     0    48    49   128    13   166     8   218   126   212       64 64 64 64 65 65 65 65 68 64
    42   138   254    87    12    32   249    87    34   172   247   107     9   142
   146   238   236    80   215    96   143   121     0   124    12    89    86   191
   179   227    64   144    89    59   240   105   105   251    46    43     0   199
                                                     ^
                                                     \ 8 = 0000 1000 = version ID (000)/Type Indicator (0)/Secondary Header
                                                     \68 = APID   cf http://planet.iitp.ru/english/spacecraft/meteor_m_n2
                                                     \0 105=packet length
```

VCDU Type [8] : 0d5 (the scanner information is present), 0d63 (the scanner information is missing)

[8] planet.iitp.ru/english/spacecraft/meteor_m_n2_structure_2_eng.htm

Sentences to pictures

Assembly of JPEG pictures for create one large image

- JPEG images with varying size depending on the encoded features
- one JPEG image can span multiple frames
- Huffman encoding + Discrete Cosine Transform + Quality quantization

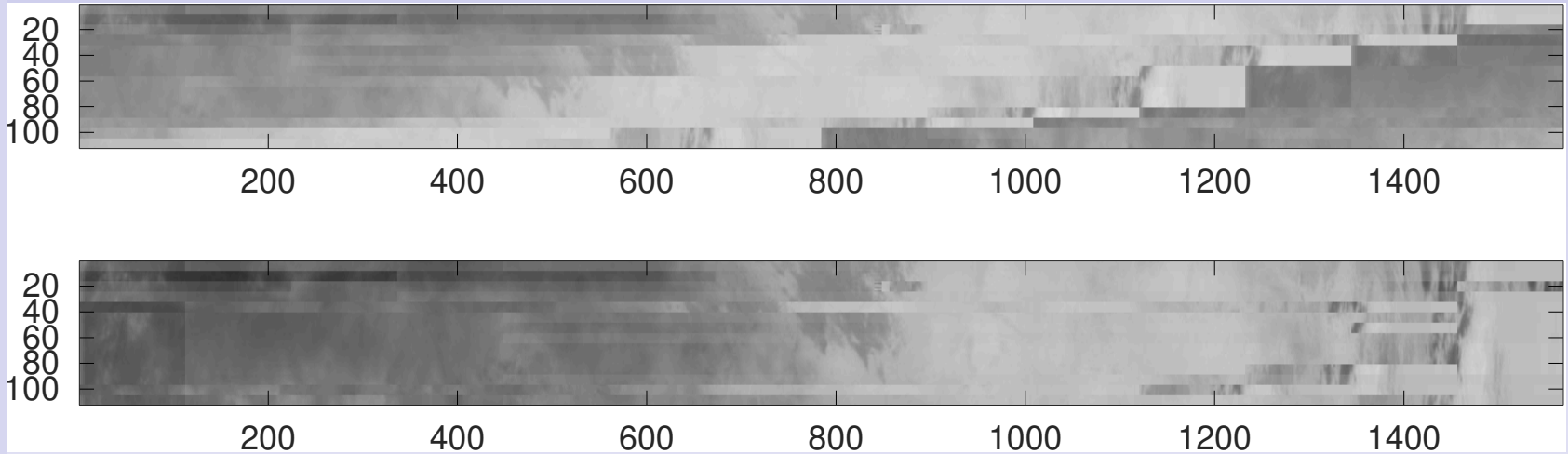
Use existing library [9] [10] to decode JPEG

```
#include "meteor_image.h"

using namespace gr::starcoder::meteor;

int main(int argc,char **argv)
{int fd,len,k,quality=77;
 unsigned char packet[1100];
 imager img=imager();

 if (argc>1) quality=atoi(argv[1]);
 fd=open("jpeg.bin",O_RDONLY);
 len=read(fd, packet, 1100);
 close(fd);
 img.dec_mcus(packet, len, 65,0,0,quality);
}
```

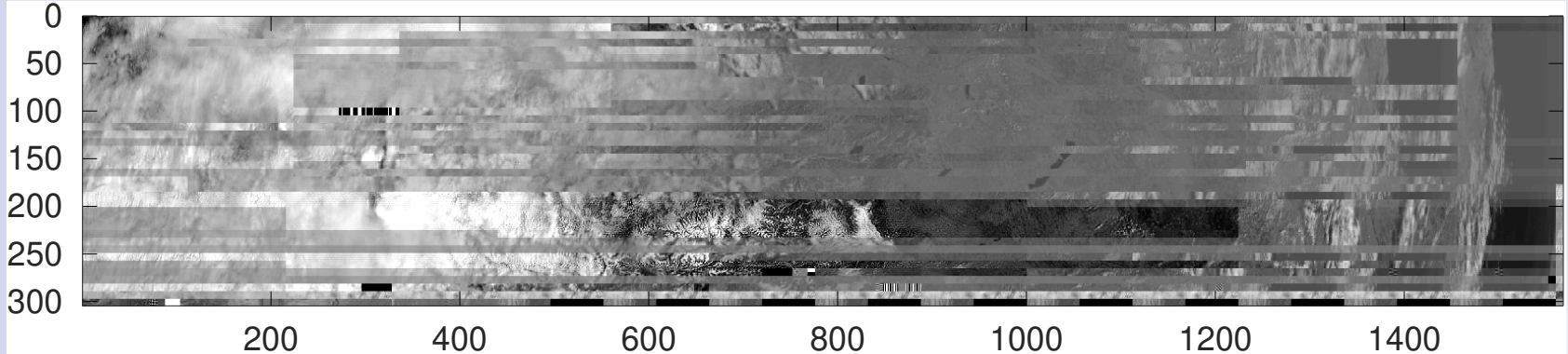
[9] www.rtl-sdr.com/new-gnu-radio-block-for-decoding-meteor-m2-images/

[10] github.com/infostellarinc/starcoder/tree/master/gr-starcoder/lib/meteor

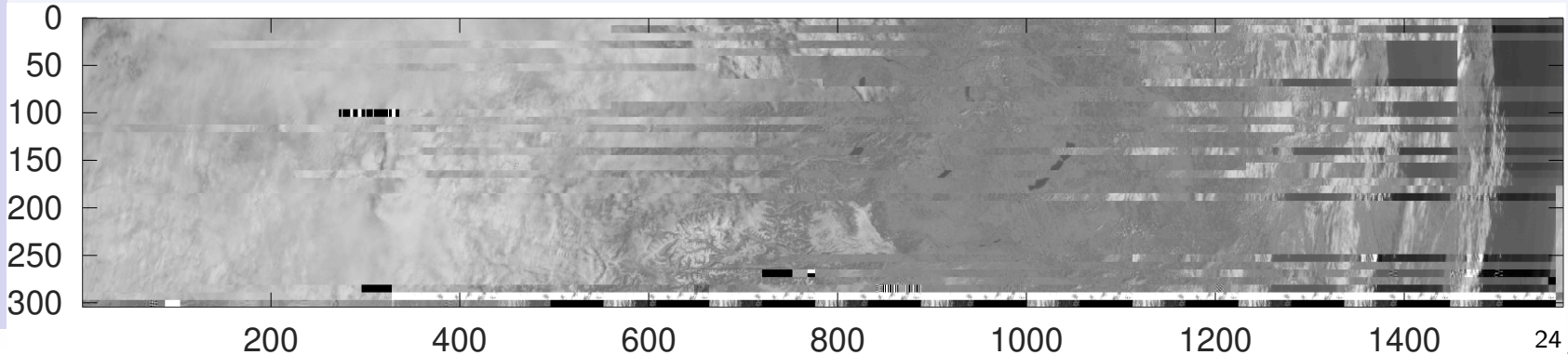
Adding missing frames



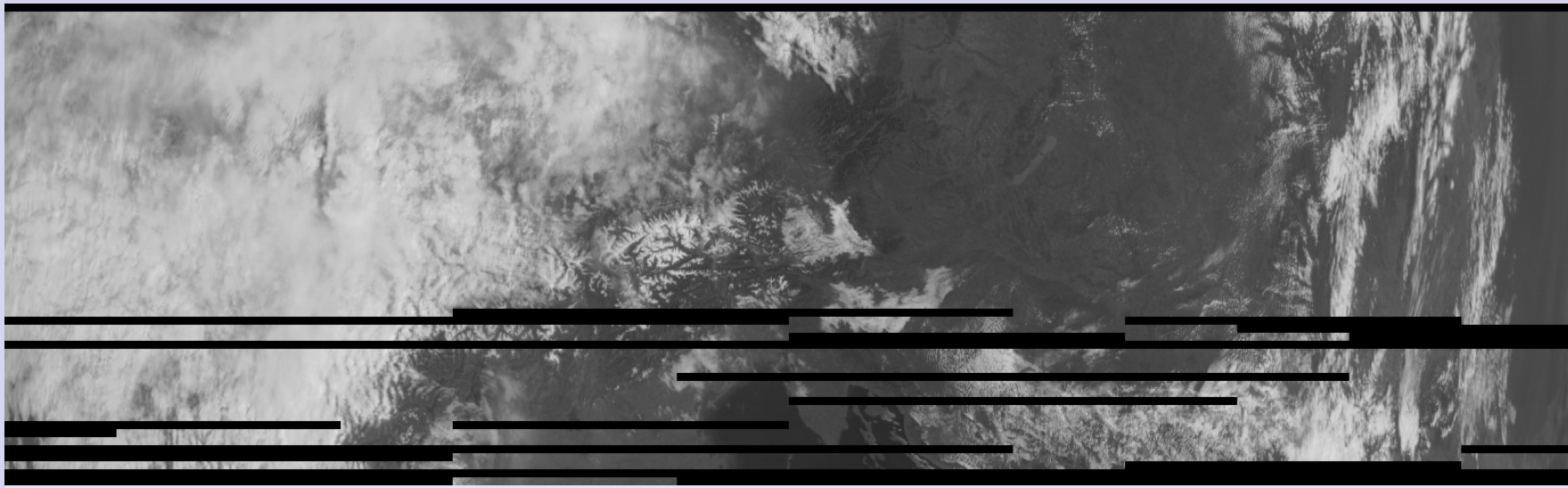
No JPEG quality (Q coefficient)



With JPEG quality (Q coefficient)



Reference picture decoded by `medet` at `https://github.com/artlav/meteor_decoder`



Conclusion

- Understanding all decoding steps, from I/Q coefficient to JPEG image assembly
- Processing method valid for all space communication compliant with CCSDS
- SDR for practical applications of basic theory (QPSK, FEC, lossy compression)

References:

1. P. Vladut *& al.*, *LRPT weather satellite image acquisition using a SDR-based reception system*, Acta Technica Napocensis **56** (2), 20–25 (2015)
2. `http://web.archive.org/web/20160616220044/http://www.meteor.robonuka.ru/`
3. L. Teske, *GOES Satellite Hunt* at `www.teske.net.br/lucas/satcom-projects/satellite-projects/`
4. D. Estévez blog at `destevez.net/2017/01/coding-for-hit-satellites-and-other-ccsds-satellites`

European GNU Radio Conference

gnuradio-fr-19.sciencesconf.org/

- **What:** 1-day oral presentations & demos, 1-day tutorials (Analog Devices PlutoSDR)
- **When:** 17-18 June 2019
- **Where:** Besançon, France (2.5 hour from Paris by high-speed train)
- Call for contrib.: March 21
- Registration (free): May 1