Kenwood RC-10 / RC-20 communication protocol analysis

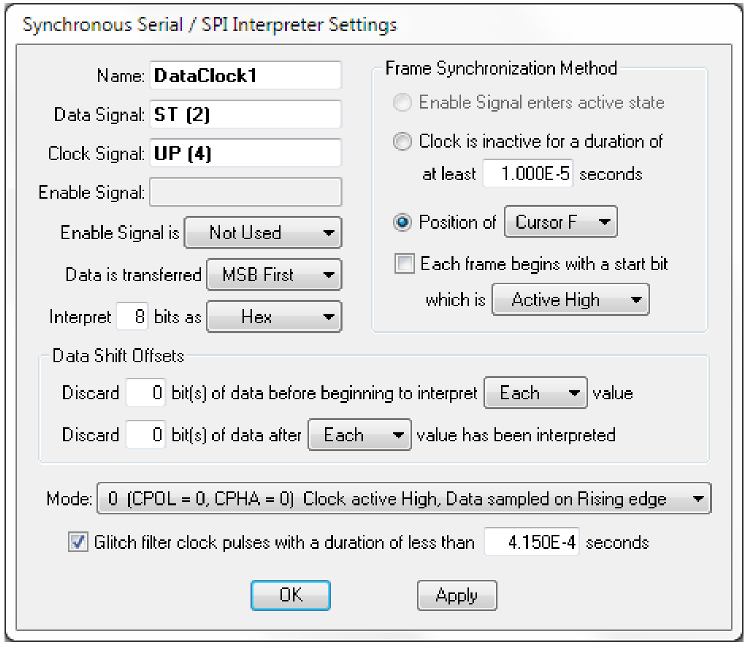
I update this document as often as I find something new, so if you're planning to use this document as reference for your project, you should ***check for updates****.*

These protocols' reverse engineering was performed on my TM-241E and TM-441E.
Additional information regarding other models have been provided by reader's feedbacks.
TM-231 and TM-431 are reported to provide full support of RC-10, and very limited support of RC-20 commands set (like software power on/off), while TM-221 and TM-421 only support RC-10 commands set.

Data frame format

The data is sent from/to the remote controller using a synchronous serial (SPI like without Enable signal) bus at 1200 kbps.
The transceiver's protocol has to be set to send the appropriate data frame since it can be used with the RC-10 which can only display the frequency, and the RC-20 which can also display the transceiver's LCD indicators and has more commands available.
To enable RC-10 mode, press CALL then switch the transceiver on.
To enable RC-20 mode, press MR then switch the transceiver on (all settings & memories will be lost).

Synchronous serial interpreter settings on LogicPort:



Synchronous Serial / SPI Interpreter Settings

Name: DataClock1
Data Signal: ST (2)
Clock Signal: UP (4)
Enable Signal:
Enable Signal is Not Used
Data is transferred MSB First
Interpret 8 bits as Hex

Frame Synchronization Method
○ Enable Signal enters active state
○ Clock is inactive for a duration of at least 1.000E-5 seconds
● Position of Cursor F
□ Each frame begins with a start bit which is Active High

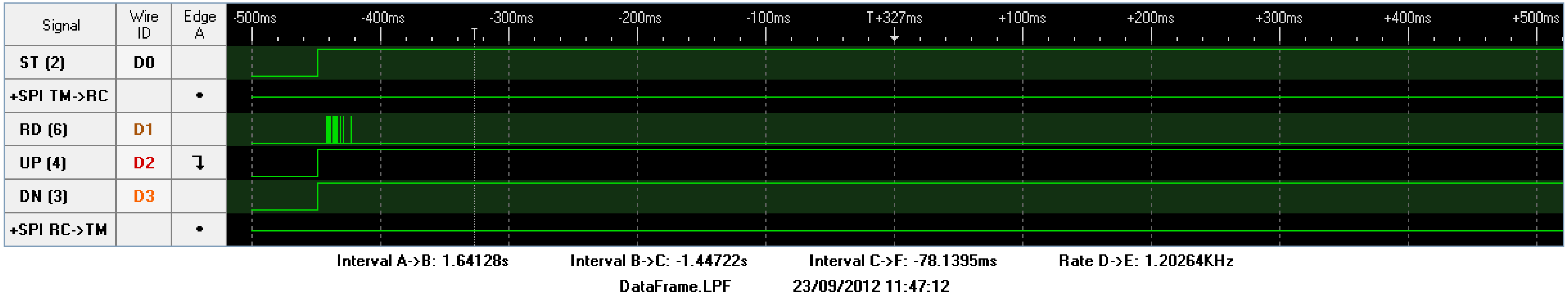
Data Shift Offsets
Discard 0 bit(s) of data before beginning to interpret Each value
Discard 0 bit(s) of data after Each value has been interpreted

Mode: 0 (CPOL = 0, CPHA = 0) Clock active High, Data sampled on Rising edge

☑ Glitch filter clock pulses with a duration of less than 4.150E-4 seconds

Start sequence

This is what happens when the transceiver is powered up with the MC-44:



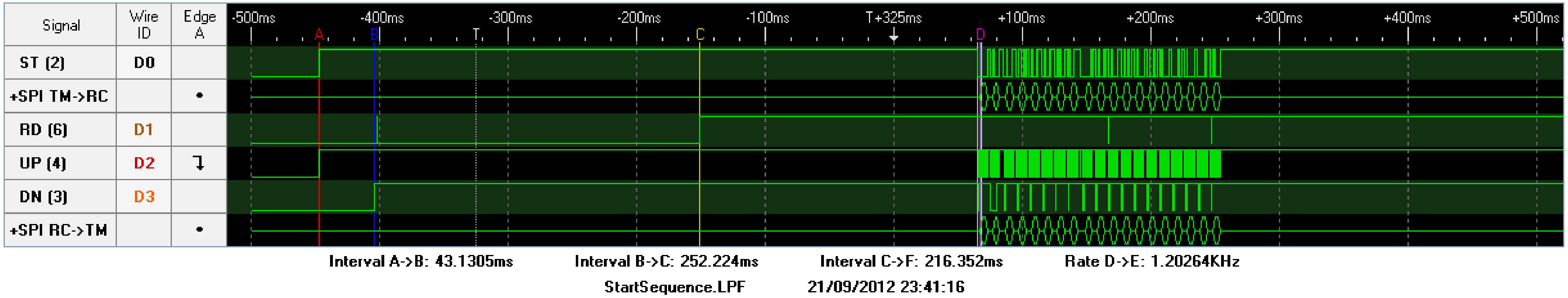
Signal	Wire ID	Edge A
ST (2)	D0	
+SPI TM->RC		•
RD (6)	D1	
UP (4)	D2	↓
DN (3)	D3	
+SPI RC->TM		•

Interval A->B: 1.64128s Interval B->C: -1.44722s Interval C->F: -78.1395ms Rate D->E: 1.20264KHz

DataFrame.LPF 23/09/2012 11:47:12

RC-10 Start sequence + 1st frame

This is what happens when the transceiver is powered up with the RC-10:

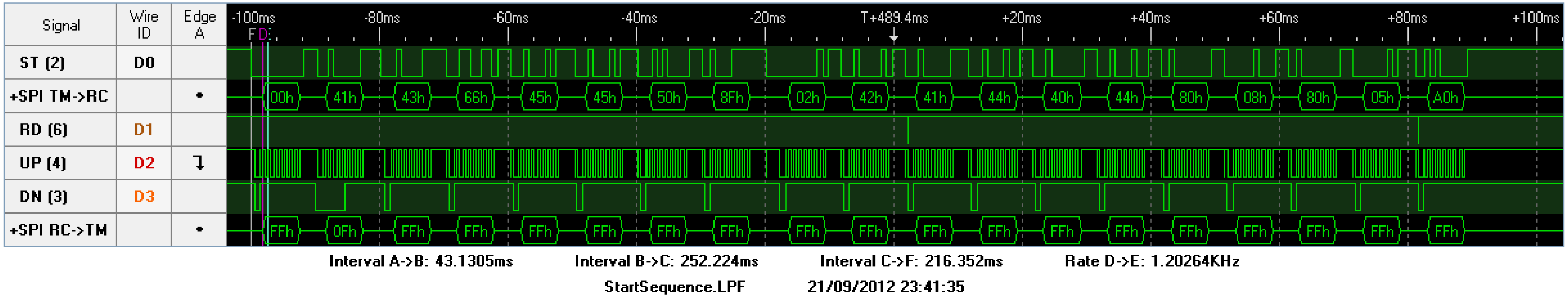


Signal	Wire ID	Edge A
ST (2)	D0	
+SPI TM->RC		•
RD (6)	D1	
UP (4)	D2	↓
DN (3)	D3	
+SPI RC->TM		•

Interval A->B: 43.1305ms Interval B->C: 252.224ms Interval C->F: 216.352ms Rate D->E: 1.20264KHz

StartSequence.LPF 21/09/2012 23:41:16

1st frame details (from the TM-241E):



Signal	Wire ID	Edge A
ST (2)	D0	
+SPI TM->RC		•
RD (6)	D1	
UP (4)	D2	↴
DN (3)	D3	
+SPI RC->TM		•

+SPI TM->RC: 00h 41h 43h 66h 45h 45h 50h 8Fh 02h 42h 41h 44h 40h 44h 80h 08h 80h 05h A0h

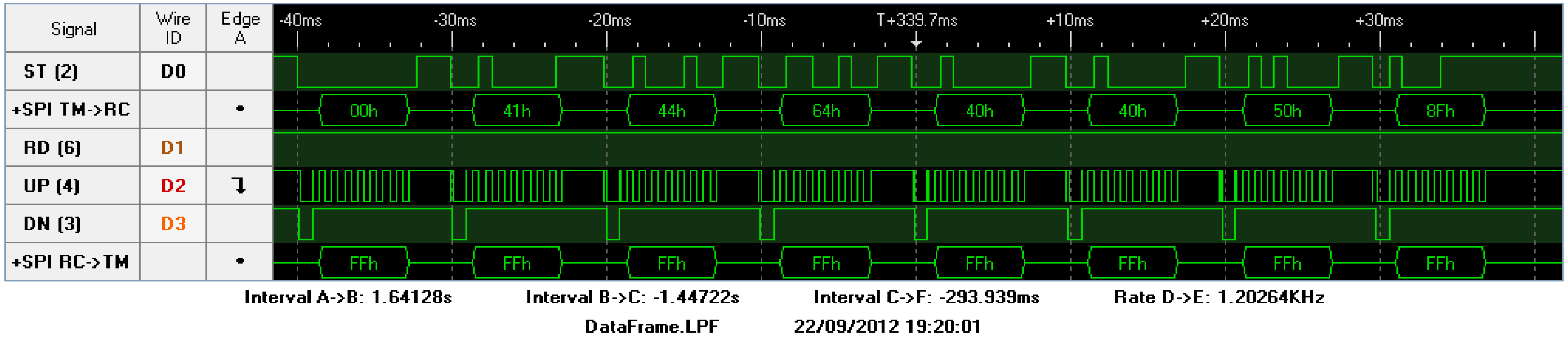
+SPI RC->TM: FFh 0Fh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 43.1305ms Interval B->C: 252.224ms Interval C->F: 216.352ms Rate D->E: 1.20264KHz

StartSequence.LPF 21/09/2012 23:41:35

In this frame, the RC-10 sends a 0F command (VOL MAIN) while the transceiver sends its data.
The frame ends with 05, indicating that the following byte contains the squelch status and signal strength. A = BUSY not displayed, signal strength = S0.

Frequency (RC-10 compliant data frame)



Signal	Wire ID	Edge A
ST (2)	D0	
+SPI TM->RC		•
RD (6)	D1	
UP (4)	D2	↴
DN (3)	D3	
+SPI RC->TM		•

+SPI TM->RC: 00h 41h 44h 64h 40h 40h 50h 8Fh

+SPI RC->TM: FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 1.64128s Interval B->C: -1.44722s Interval C->F: -293.939ms Rate D->E: 1.20264KHz

DataFrame.LPF 22/09/2012 19:20:01

Frequency data frame content on TM-2x1, TM-3x1 and TM-4x1:

y = 1: ON AIR	x = MHz * 100	x = MHz * 10	x = MHz	x = kHz * 100	x = kHz * 10	x = kHz	x = Hz * 100
0000 000y	0100 xxxx	0100 xxxx	0110 xxxx	0100 xxxx	0100 xxxx	0101 xxxx	1000 xxxx

Frequency data frame content on TM-5x1:

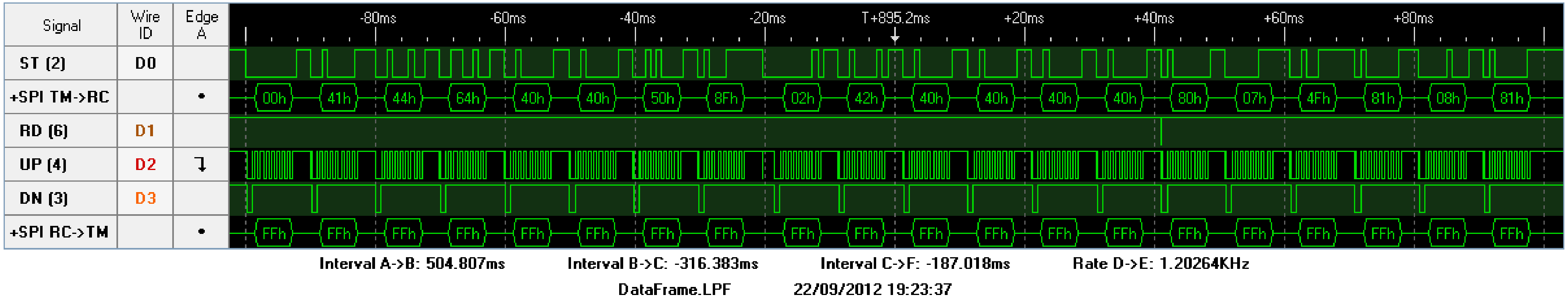
y = 1: ON AIR	x = GHz	x = MHz * 100	x = MHz * 10	x = MHz	x = kHz * 100	x = kHz * 10	x = kHz	x = Hz * 100
0000 000y	0100 xxxx	0100 xxxx	0100 xxxx	0110 xxxx	0100 xxxx	0100 xxxx	0101 xxxx	1000 xxxx

Different values for xxxx:

Display :	0	1	2	3	4	5	6	7	8	9	None	Hyphen
xxxx :	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1111	1010

Other LCD data frame

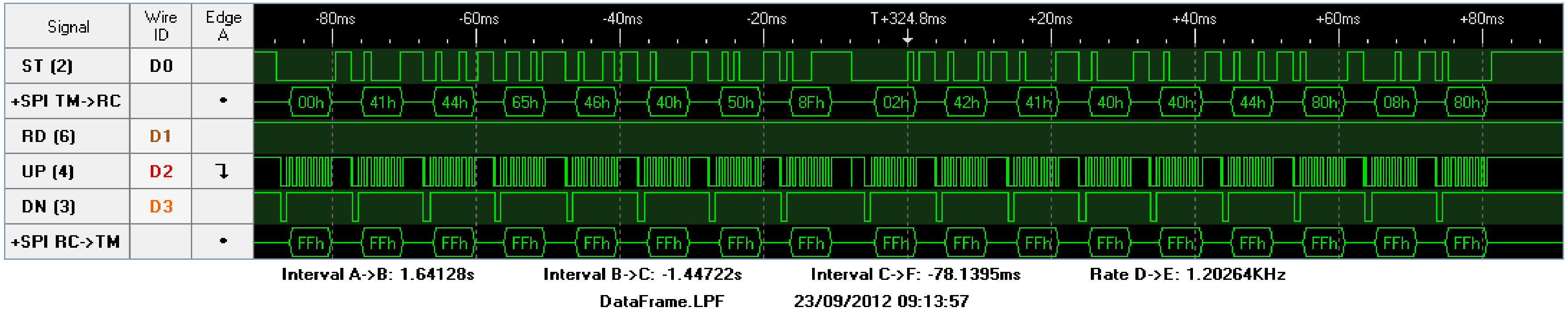
1°) Frequency with LCD Indicators (RC-20 compliant data frame)


Signal
Wire ID
Edge A
-80ms
-60ms
-40ms
-20ms
T+895.2ms
+20ms
+40ms
+60ms
+80ms
ST (2)
D0
+SPI TM->RC
00h 41h 44h 64h 40h 40h 50h 8Fh 02h 42h 40h 40h 40h 40h 80h 07h 4Fh 81h 08h 81h
RD (6)
D1
UP (4)
D2
DN (3)
D3
+SPI RC->TM
FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 504.807ms Interval B->C: -316.383ms Interval C->F: -187.018ms Rate D->E: 1.20264KHz

DataFrame.LPF 22/09/2012 19:23:37

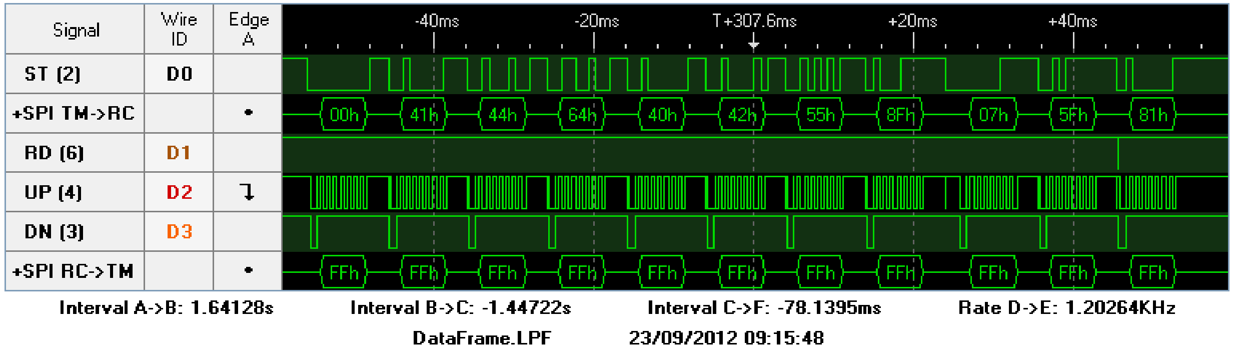
CALL channel:


Signal
Wire ID
Edge A
-80ms
-60ms
-40ms
-20ms
T+324.8ms
+20ms
+40ms
+60ms
+80ms
ST (2)
D0
+SPI TM->RC
00h 41h 44h 65h 46h 40h 50h 8Fh 02h 42h 41h 40h 40h 44h 80h 08h 80h
RD (6)
D1
UP (4)
D2
DN (3)
D3
+SPI RC->TM
FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 1.64128s Interval B->C: -1.44722s Interval C->F: -78.1395ms Rate D->E: 1.20264KHz

DataFrame.LPF 23/09/2012 09:13:57

SCAN in progress:

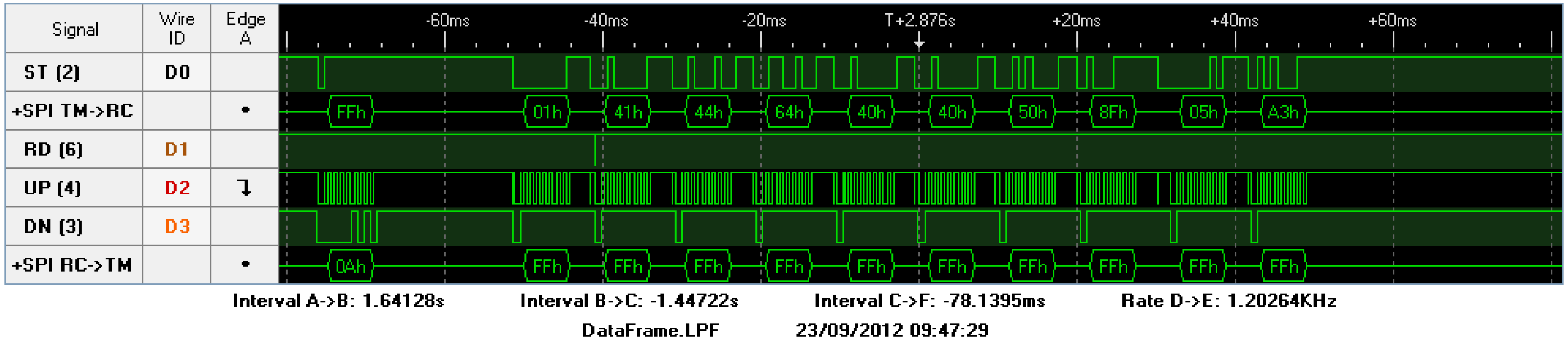


Signal	Wire ID	Edge A
ST (2)	D0	
+SPI TM->RC		•
RD (6)	D1	
UP (4)	D2	↓
DN (3)	D3	
+SPI RC->TM		•

Interval A->B: 1.64128s Interval B->C: -1.44722s Interval C->F: -78.1395ms Rate D->E: 1.20264KHz

DataFrame.LPF 23/09/2012 09:15:48

PTT pressed (TX):

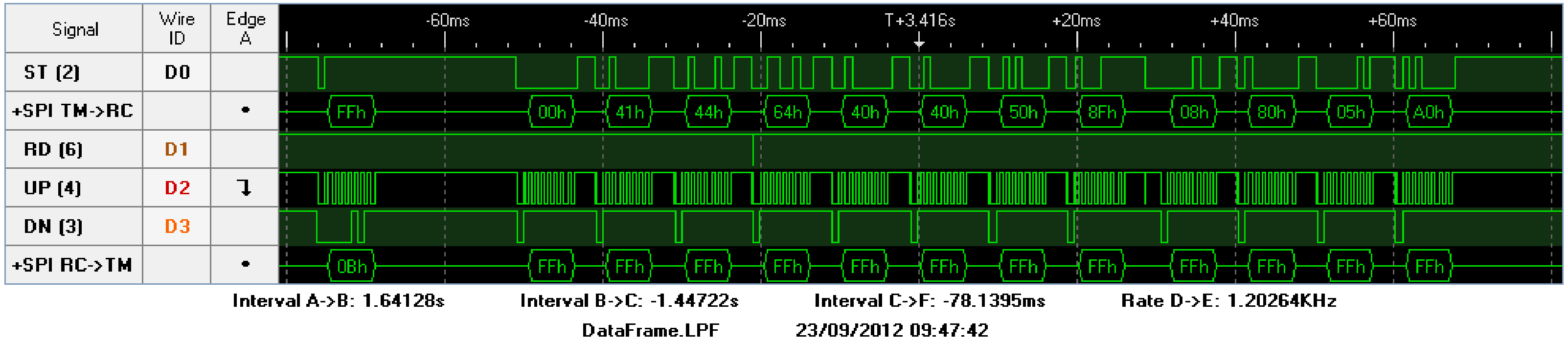


Signal	Wire ID	Edge A
ST (2)	D0	
+SPI TM->RC		•
RD (6)	D1	
UP (4)	D2	↓
DN (3)	D3	
+SPI RC->TM		•

Interval A->B: 1.64128s Interval B->C: -1.44722s Interval C->F: -78.1395ms Rate D->E: 1.20264KHz

DataFrame.LPF 23/09/2012 09:47:29

PTT released (RX):

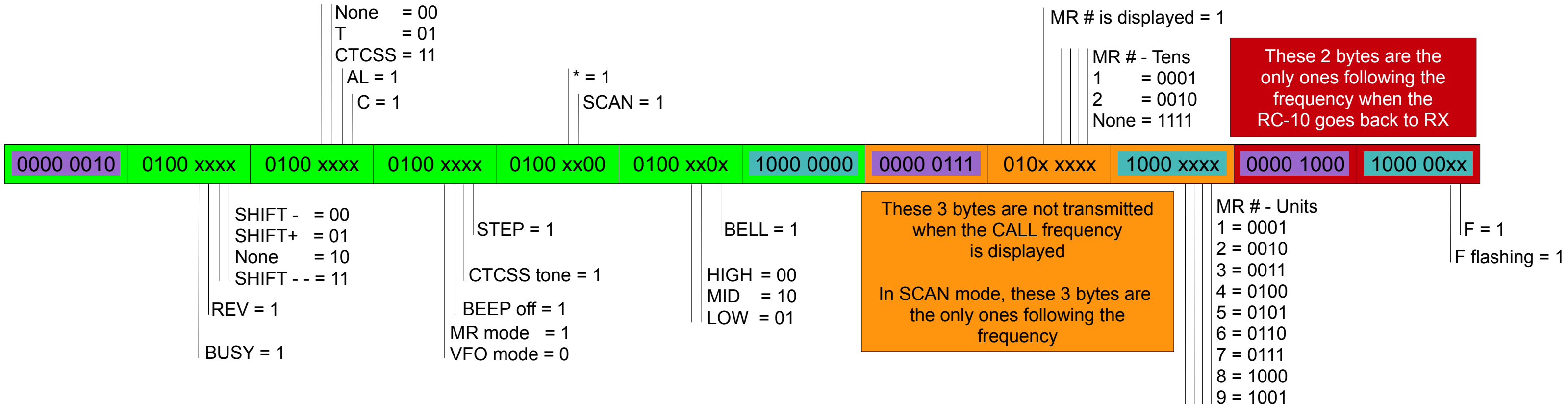


Interval A->B: 1.64128s Interval B->C: -1.44722s Interval C->F: -78.1395ms Rate D->E: 1.20264KHz

DataFrame.LPF 23/09/2012 09:47:42

The last two data frames always end with the S-METER data.

Bit field details:



These 2 bytes are the only ones following the frequency when the RC-10 goes back to RX

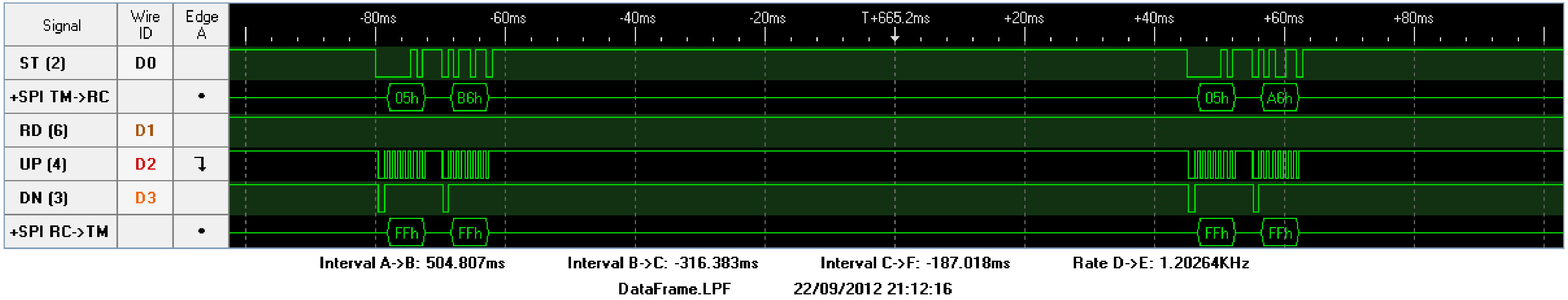
These 3 bytes are not transmitted when the CALL frequency is displayed

In SCAN mode, these 3 bytes are the only ones following the frequency

🟩 LCD indicators frame 🟧 MR frame 🟥 F frame

The LOCK, A.LOCK, TOT and APO indicators are not transmitted, in any data frame.
Automatic Power Off is disabled when using the transceiver with the remote controller.

2°) S-METER



Interval A->B: 504.807ms Interval B->C: -316.383ms Interval C->F: -187.018ms Rate D->E: 1.20264KHz

DataFrame.LPF 22/09/2012 21:12:16

Bit field details:

BUSY OFF = 0
BUSY ON = 1

0000 0101	101x 0xxx

Signal strength:	Output power:
S0 = 000	
S1 = 001	
S3 = 010	LOW = 010
S5 = 011	
S7 = 100	
S9 = 101	MID = 011
S9+20 = 110	
S9+40 = 111	HIGH = 111

This frame is sent by the transceiver every time the S-Meter value changes.

3°) ◄ ALT ► indicators (TM-5x1 only)

0000 0110	1000 0xxx

None = 000
ALT = 100
► = 101
◄ = 110

Each block of data frame can be identified by its “Start Of Frame” marker (SOF): 0000 xxxx.
Each one of these blocks also end with a specific byte that is used as an “End Of Frame” marker (EOF): 10xx xxxx.
These markers help to distinguish data when more than one block is sent at a time.
The RC-10 doesn't follow that protocol when sending commands to the transceiver, but the RC-20 does.

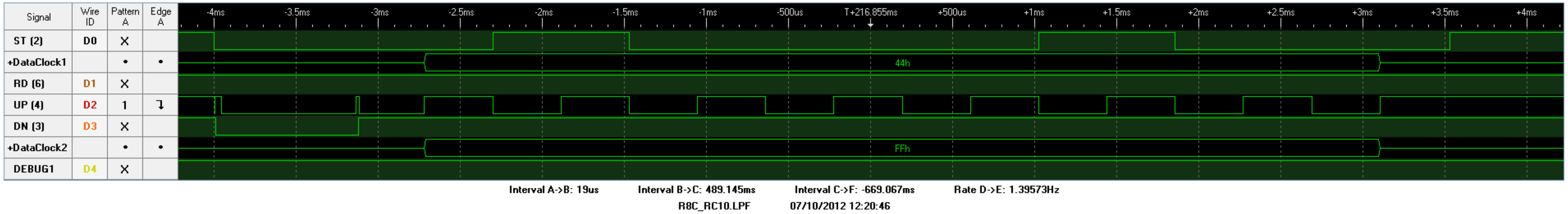
SOF:

0000 0000	Transceiver: Frequency
0000 0010	Transceiver: Most LCD indicators
0000 0101	Transceiver: Squelch and signal strength
0000 0110	Transceiver: ALT indicators
0000 0111	Transceiver: Memory number
0000 1000	Transceiver: F key indicator

Communication protocol

The transceiver acts as the master device on the bus, which means that the clock signal is never generated by the remote controller.

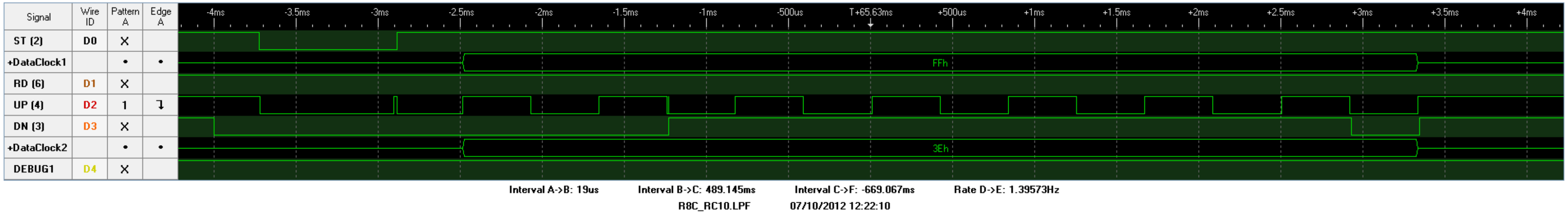
1°) Data transmission from the transceiver to the remote controller:



Interval A->B: 19us Interval B->C: 489.145ms Interval C->F: -669.067ms Rate D->E: 1.39573Hz
R8C_RC10.LPF 07/10/2012 12:20:46

- The transceiver pulls its data line low and waits for the remote controller to acknowledge
- The remote controller acknowledges by pulling its data line low too
- The transceiver then pulls the clock line low, the remote controller's data line remains unchanged during this time
- The transceiver pulls the clock line high for a very short time (~ 19µs) and both devices start sending their first bit
- The first byte is then transmitted simultaneously by both devices.
  - The bits are set on data lines on falling edges of the clock and sampled on rising edges
- Whatever is the last sent bit, each data lines are pulled high at the end of each transmitted byte

2°) Data transmission from the remote controller to the transceiver



Interval A->B: 19us Interval B->C: 489.145ms Interval C->F: -669.067ms Rate D->E: 1.39573Hz
R8C_RC10.LPF 07/10/2012 12:22:10

The sequence is quite identical, excepted that the transmission is started by pulling the remote controller's data line low and acknowledged by the transceiver by pulling its data line low.

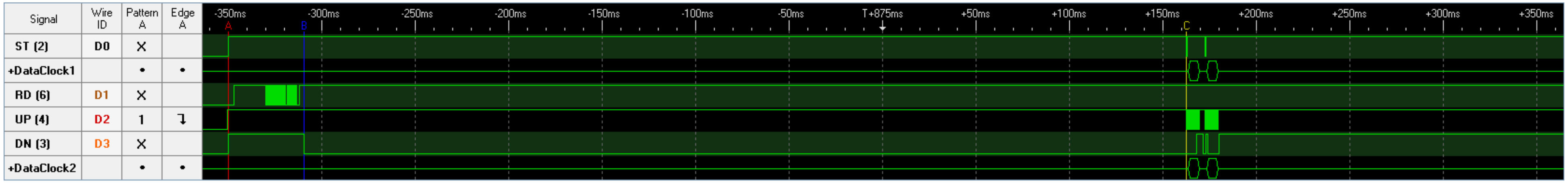
RC-10 commands list

00	Unknown*	10	Save VFO to Memory # 10	20	Unknown* (DUP)	30	VFO: keypad, M: Memory # [3]
01	Unknown*	11	Save VFO to Memory # 1	21	Unknown*	31	VFO: keypad, M: Memory # [3]
02	Unknown* (SEL)	12	Save VFO to Memory # 2	22	SQL: Open / Close	32	VFO: keypad, M: Memory # [3]
03	Unknown	13	Save VFO to Memory # 3	23	SHIFT: + / - / - - / None	33	VFO: keypad, M: Memory # [3]
04	Unknown*	14	Save VFO to Memory # 4	24	REV: ON / OFF	34	VFO: keypad, M: Memory # [3]
05	Unknown	15	Save VFO to Memory # 5	25	T / CTCSS / NONE	35	VFO: keypad, M: Memory # [3]
06	Unknown*	16	Save VFO to Memory # 6	26	Unknown*	36	VFO: keypad, M: Memory # [3]
07	Unknown*	17	Save VFO to Memory # 7	27	Mem channel Lockout On/Off	37	VFO: keypad, M: Memory # [3]
08	Unknown+ (beep on TM-x31)	18	Save VFO to Memory # 8	28	SQL: Close	38	VFO: keypad, M: Memory # [3]
09	Unknown+ (higher pitch beep on TM-x31)	19	Save VFO to Memory # 9	29	SQL: Open	39	VFO: keypad, M: Memory # [3]
0A	Transmit	1A	Switch between VFO/M	2A	Go to / Exit CALL channel	3A	VFO / M
0B	Receive	1B	Unknown*	2B	Unknown* (BAND)	3B	Start / Stop SCAN
0C	Stop remote volume +/- [1]	1C	Unknown*	2C	Go to VFO	3C	Increase remote volume [1]
0D	Stop remote volume +/- [1]	1D	Unknown*	2D	Go to Memory channels	3D	Decrease remote volume [1]
0E	Remote volume control [2]	1E	Unknown*	2E	Down	3E	Down
0F	Transceiver volume control	1F	Unknown*	2F	Up	3F	Up

* This command is recognized by the transceiver (data frames returned) but its purpose is still unknown.
+ This command is recognized by the transceiver (BEEP) but its purpose is still unknown.
[1] The volume level is automatically incremented after sending a 3C command and automatically decremented after sending a 3D command. It must then be stopped using the 0C or 0D command, otherwise the volume won't stop increasing or decreasing until reaching its limits. This is automatically done by the RC-10.
[2] This command must be sent prior to use the 3C and/or 3D command. When going back to “Transceiver volume control”, the “Remote volume control”'s value is lost.
[3] In VFO mode: enter frequency digits, in Memory mode: go to memory number 1 to 10.
RED TEXT indicates a function with the same code on the RC-20.

Any other value seem to be ignored by the transceiver.

RC-20 Start sequence

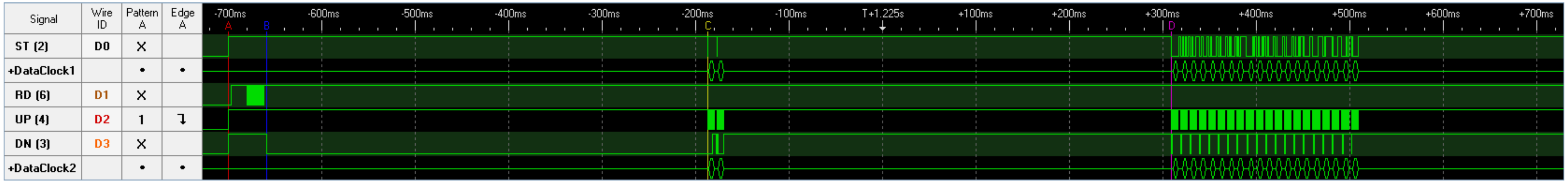


Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	1	↴
DN (3)	D3	X	
+DataClock2		•	•

Interval A->B: 40.4185ms Interval B->C: 471.96ms Interval C->D: 496.712ms Rate D->E: N/A

RC20_StartSequence.LPF 22/10/2012 20:34:22

RC-20 Start sequence + 1st frame

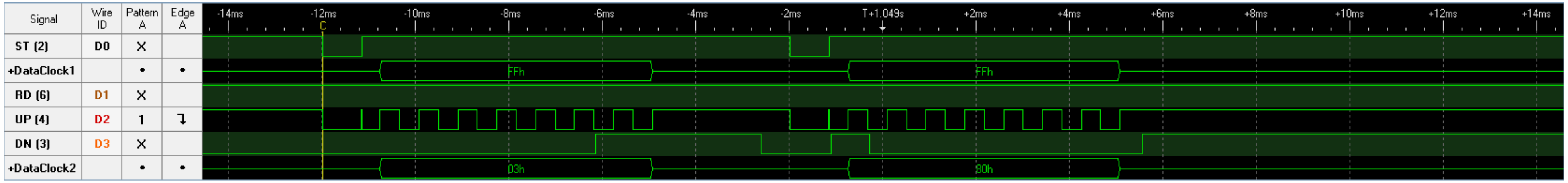


Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	1	↴
DN (3)	D3	X	
+DataClock2		•	•

Interval A->B: 40.4185ms Interval B->C: 471.96ms Interval C->D: 496.712ms Rate D->E: N/A

RC20_StartSequence.LPF 22/10/2012 20:31:22

Power ON command



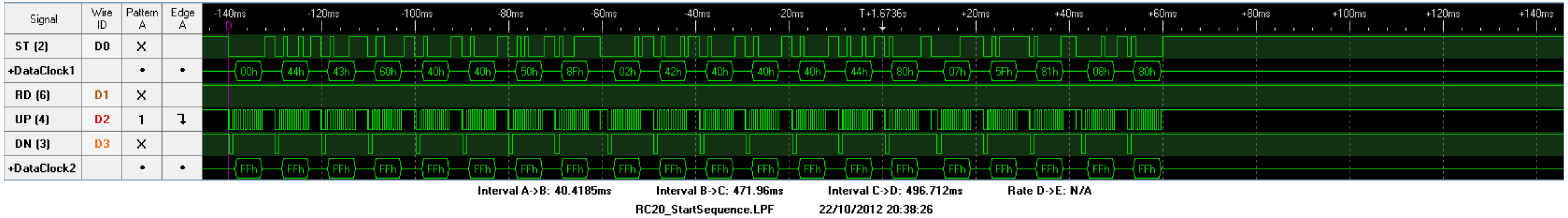
Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	1	↴
DN (3)	D3	X	
+DataClock2		•	•

Interval A->B: 40.4185ms Interval B->C: 471.96ms Interval C->D: 496.712ms Rate D->E: N/A

RC20_StartSequence.LPF 22/10/2012 20:36:06

The Power ON command is always sent by the RC-20 when the transceiver is powered on.

1st frame details (from the TM-441E):



Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	1	↴
DN (3)	D3	X	
+DataClock2		•	•

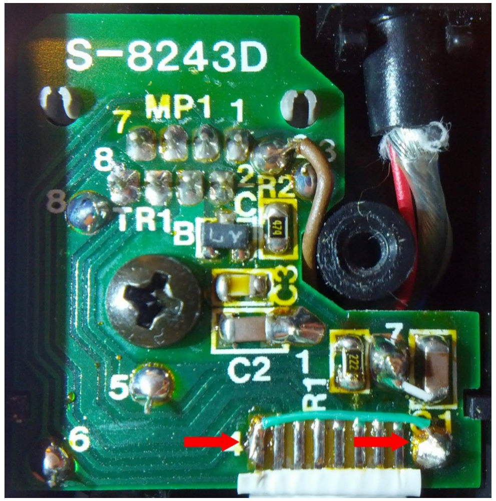
+DataClock1: 00h 44h 43h 60h 40h 40h 50h 8Fh 02h 42h 40h 40h 40h 44h 80h 07h 5Fh 81h 08h 80h

+DataClock2: FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 40.4185ms Interval B->C: 471.96ms Interval C->D: 496.712ms Rate D->E: N/A

RC20_StartSequence.LPF 22/10/2012 20:38:26

The RC-20 doesn't need the clock signal to communicate with the transceiver, but the transceiver sends it anyway.



In order to get the clock signal from the transceiver with the RC-20 connectors adapter attached to it, you must add a wire as shown on the picture.
This will connect pin number 4 to both microphone plugs.
This modification is only helpful when sniffing the data communication between both devices.

RC-20 commands list (link with a transceiver)

02 84 02 92	F + any of the 4 VOL ▲ and ▼buttons*	03 80	**Power** ON	03 97	**ALT**	03 91	**M**
		03 8A	**Power** OFF	02	F_ + DRS	03 A4	F_ + MR
02 86 02 93	F + SQL ▲ or ▼*	03 81	**F**	22	**SQL**	03 89	**MHz**
		03 86	**F_** [2]	03 AB	F_ + MUTE	03 A3	F_ + MHz
04 3D 04 0D	**SUB VOL ▼***	23	**SHIFT**	27	**L.O**	3D	**MAIN VOL ▼**[1]
		03 93	**ACC**	03 AC	F_ + ABC	0F	F_ + MAIN VOL ▼
04 3C 04 0D	**SUB VOL ▲***	03 88	F_ + SHIFT: Set VFO higher limit	20	**DUP**	3C	**MAIN VOL ▲**[1]
		25	**TONE**: T / CTCSS	03 AD	F_ + DUAL	0E	F_ + MAIN VOL ▲
02 89	**MUTE**	03 94	**SEL** (CTCSS freq. select)	2B	**BAND**	03 A8	F_ + SUB VOL ▼
02 8B	**DUAL**	26	F_ + TONE	02	**SEL**	03 A9	F_ + SUB VOL ▲
02 95	Initialize IF-20 (ENT + POWER ON)	24	**REV**	03 AE	F_ + BAND	03 A7	F_ + SQL ▼
02 A8	**ABC**	03 95	**STEP**	03 83	**LOW**	03 A6	F_ + SQL ▲
02 8F	F + SUB	03 9A	F_ + REV (key BEEP ON/OFF)	03 99	F_ + LOW	03 A5	F_ + SUB
		03 84	**T.ALT**	2C	**VFO**	2A	**CALL**
10 to 19	F + ENT + key 0 to 9	03 96	**AL**	03 90	**M ►V**	03 92	F + CALL
03 9B	Initialize VFO (VFO + POWER ON)	03 AA	F_ + T.ALT	03 98	F_ + VFO	03 87	F_ + CALL: Set VFO lower limit
03 9C	Factory default (MR + POWER ON)	03 85	**DRS**	3A	**MR**		

* The RC-20 sends the first 2 bytes which are followed by the frequency & LCD indicators, then the RC-20 sends the other 2 bytes which are also followed by the frequency & LCD indicators.

[1] The volume level is automatically incremented after sending a 3C command and automatically decremented after sending a 3D command. It must then be stopped using the 0C or 0D command, otherwise the volume won't stop increasing or decreasing until reaching its limits. This is automatically done by the RC-20.

[2] **F_** indicates that the **F** key is pressed longer than 1 second.

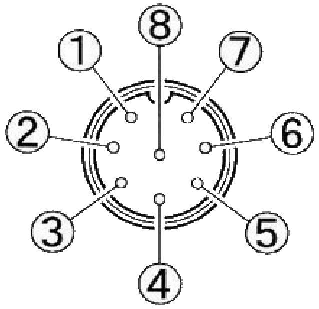
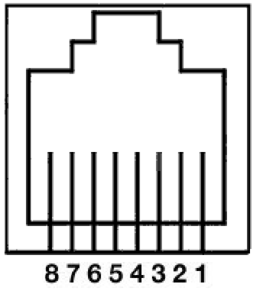
The commands that start with 02 or 04 are used for the communication with the IF-20.

BOLD text indicate a key name, otherwise it's a combination of keys.

Some unlisted keys like **ENT** or **LOCK** just send a 08 byte to the transceiver, which only makes it BEEP.

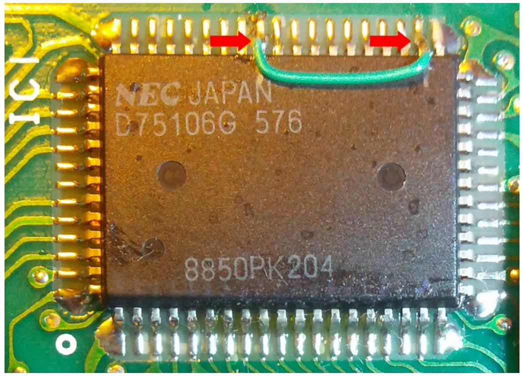
The RC-20 **LOCK** key only locks its own keys.

IF-20 to TM-xx1 connecting cables (PG-4H - Kenwood ref.: E30-2146-05)

View from front of transceiver	Microphone connector pin number	Signal	Color	RJ45 connector pin number *	View from front of interface *
	1	Mic	White	7	
	2	PTT / Data out (from TX)	Blue	5	
	3	Down / Data in (from RC)	Brown	4	
	4	Up / Clock (from TX)	Red	1	
	5	≃ 6VDC	Yellow	3	
	6	AF out (from TX)	Black	6	
	7	Analog ground (Mic & AF)	Shield	8	
	8	Digital ground	Green	2	

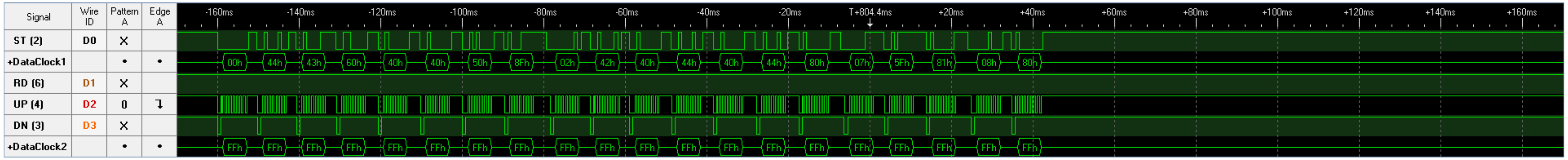
* This is the RJ45 female connector

The IF-20 doesn't generate the clock signal to communicate with the RC-20.



You must add a wire as shown on the picture in order to get the clock signal from the IF-20.
This will connect pin number 34 (“Down” input from the microphone connector) to the $\overline{\text{SCK}}$ signal output from the microcontroller, just like it's done on the transceiver's PCB.
This modification is very useful when connecting the IF-20 to a home made device, as it simplifies the data transmission and reception protocol to be coded on the microcontroller's embedded software.

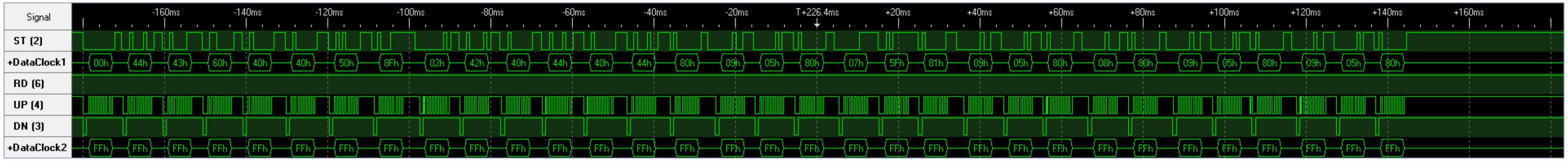
LCD informations from MAIN transceiver, without SUB transceiver connected:


Signal
Wire ID
Pattern A
Edge A
ST (2) D0 X
+DataClock1 • •
RD (6) D1 X
UP (4) D2 0 ↴
DN (3) D3 X
+DataClock2 • •
-160ms -140ms -120ms -100ms -80ms -60ms -40ms -20ms T+804.4ms +20ms +40ms +60ms +80ms +100ms +120ms +140ms +160ms
00h 44h 43h 60h 40h 40h 50h 8Fh 02h 42h 40h 44h 40h 44h 80h 07h 5Fh 81h 08h 80h
FFh

Interval A->B: 273ms Interval B->C: 110ms Interval C->F: -386ms Rate D->E: 5.4945Hz
IF20.LPF 04/11/2012 14:44:22

This data frame is exactly the same as when the RC-20 is directly connected to a transceiver. This is the same for every other data frame.

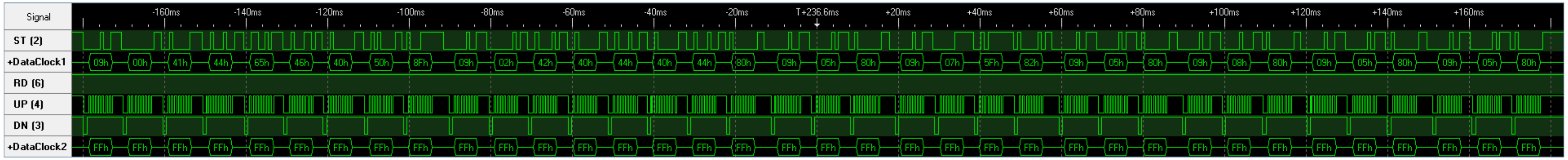
LCD informations from MAIN transceiver, with SUB transceiver connected:


Signal
ST (2)
+DataClock1
RD (6)
UP (4)
DN (3)
+DataClock2
-160ms -140ms -120ms -100ms -80ms -60ms -40ms -20ms T+226.4ms +20ms +40ms +60ms +80ms +100ms +120ms +140ms +160ms
00h 44h 43h 60h 40h 40h 50h 8Fh 02h 42h 40h 44h 40h 44h 80h 09h 05h 80h 07h 5Fh 81h 09h 05h 80h 08h 80h 09h 05h 80h 09h 05h 80h
FFh

Interval A->B: 273ms Interval B->C: 110ms Interval C->F: -386ms Rate D->E: 5.4945Hz
IF20.LPF 04/11/2012 14:49:29

There is a 3 bytes frame – 09 05 80 – at the end of the LCD indicators, MR and F frames, and once again at the end of the whole MAIN data frame.

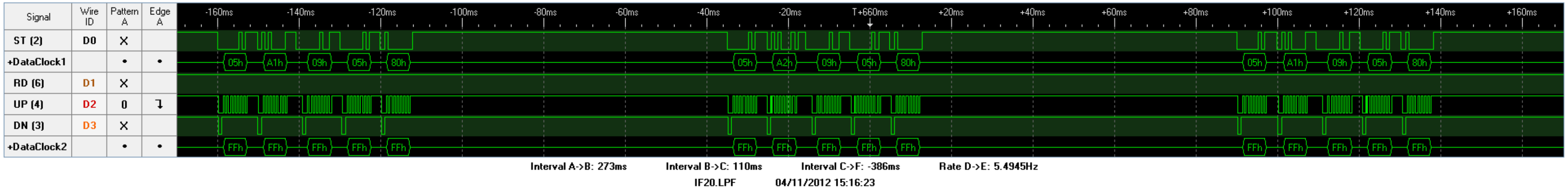
LCD informations from SUB transceiver:


Signal
ST (2)
+DataClock1
RD (6)
UP (4)
DN (3)
+DataClock2
-160ms -140ms -120ms -100ms -80ms -60ms -40ms -20ms T+236.6ms +20ms +40ms +60ms +80ms +100ms +120ms +140ms +160ms
09h 00h 41h 44h 65h 46h 40h 50h 8Fh 09h 02h 42h 40h 44h 40h 44h 80h 09h 05h 80h 09h 07h 5Fh 82h 09h 05h 80h 09h 08h 80h 09h 05h 80h 09h 05h 80h
FFh

Interval A->B: 273ms Interval B->C: 110ms Interval C->F: -386ms Rate D->E: 5.4945Hz
IF20.LPF 04/11/2012 14:50:08

There's a new byte again – 09 – which indicates that each following frame is from the SUB transceiver.

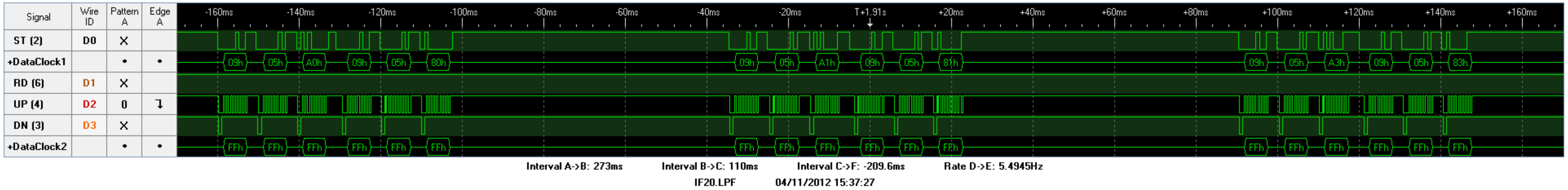
S-METER frame from MAIN transceiver, with SUB transceiver connected:



Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	0	↴
DN (3)	D3	X	
+DataClock2		•	•

Interval A->B: 273ms Interval B->C: 110ms Interval C->F: -386ms Rate D->E: 5.4945Hz
IF20.LPF 04/11/2012 15:16:23

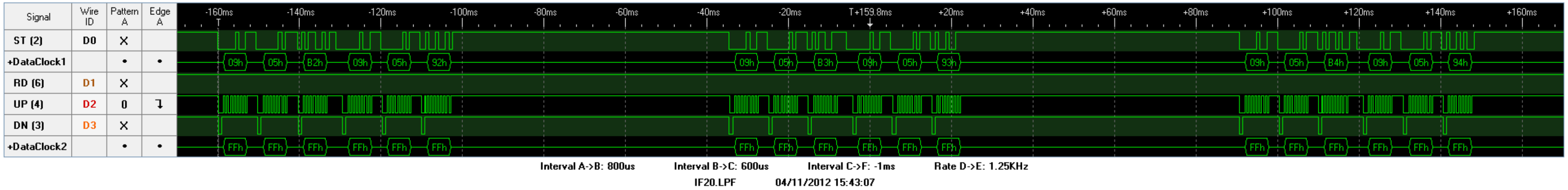
S-METER frame from SUB transceiver:



Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	0	↴
DN (3)	D3	X	
+DataClock2		•	•

Interval A->B: 273ms Interval B->C: 110ms Interval C->F: -209.6ms Rate D->E: 5.4945Hz
IF20.LPF 04/11/2012 15:37:27

The “BUSY” information does not come from the transceiver, but from the IF-20, since it has its own squelch settings. This means that the transceiver may display “BUSY” for a S5 signal but not the RC-20 if the IF-20 has been set to receive S9 signals for instance.

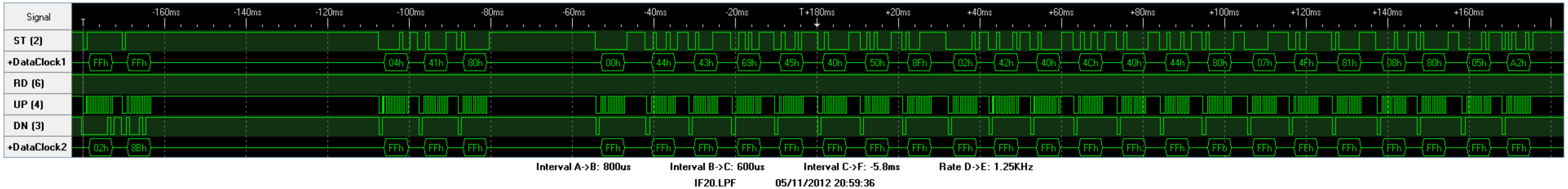


Signal	Wire ID	Pattern A	Edge A
ST (2)	D0	X	
+DataClock1		•	•
RD (6)	D1	X	
UP (4)	D2	0	↴
DN (3)	D3	X	
+DataClock2		•	•

Interval A->B: 800us Interval B->C: 600us Interval C->F: -1ms Rate D->E: 1.25KHz
IF20.LPF 04/11/2012 15:43:07

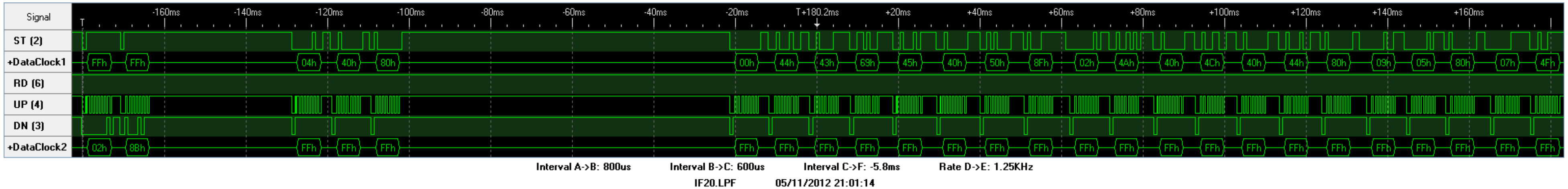
The EOF varies with the signal strength for some reason, which doesn't happen in the MAIN S-METER data frame.

Switching the SUB transceiver OFF by pressing DUAL on the RC-20:


Signal
-160ms -140ms -120ms -100ms -80ms -60ms -40ms -20ms T+180ms +20ms +40ms +60ms +80ms +100ms +120ms +140ms +160ms
ST (2)
+DataClock1
FFh FFh 04h 41h 80h 00h 44h 43h 69h 45h 40h 50h 8Fh 02h 42h 40h 4Ch 40h 44h 80h 07h 4Fh 81h 08h 80h 05h A2h
RD (6)
UP (4)
DN (3)
+DataClock2
02h 8Bh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

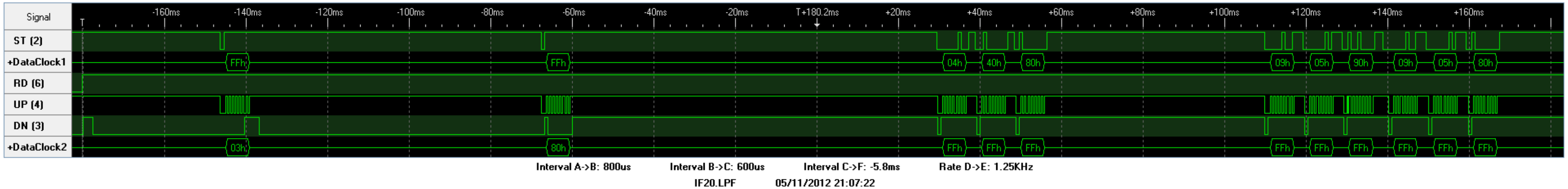
Interval A->B: 800us Interval B->C: 600us Interval C->F: -5.8ms Rate D->E: 1.25KHz
IF20.LPF 05/11/2012 20:59:36

Switching the SUB transceiver ON by pressing DUAL on the RC-20:


Signal
-160ms -140ms -120ms -100ms -80ms -60ms -40ms -20ms T+180.2ms +20ms +40ms +60ms +80ms +100ms +120ms +140ms +160ms
ST (2)
+DataClock1
FFh FFh 04h 40h 80h 00h 44h 43h 69h 45h 40h 50h 8Fh 02h 4Ah 40h 4Ch 40h 44h 80h 09h 05h 80h 07h 4Fh
RD (6)
UP (4)
DN (3)
+DataClock2
02h 8Bh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

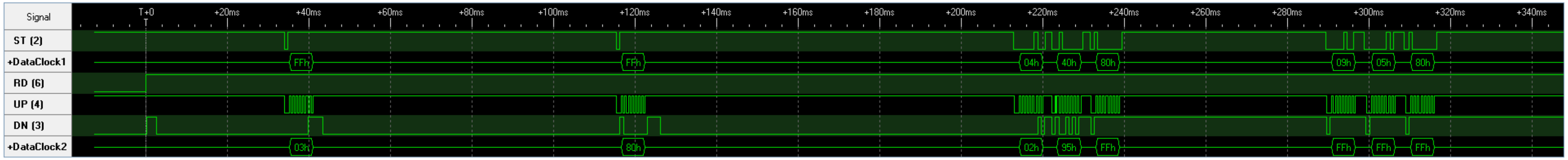
Interval A->B: 800us Interval B->C: 600us Interval C->F: -5.8ms Rate D->E: 1.25KHz
IF20.LPF 05/11/2012 21:01:14

Switching transceivers ON with the RC-20 POWER button (the transceivers data frames that follow the ones below are not shown):

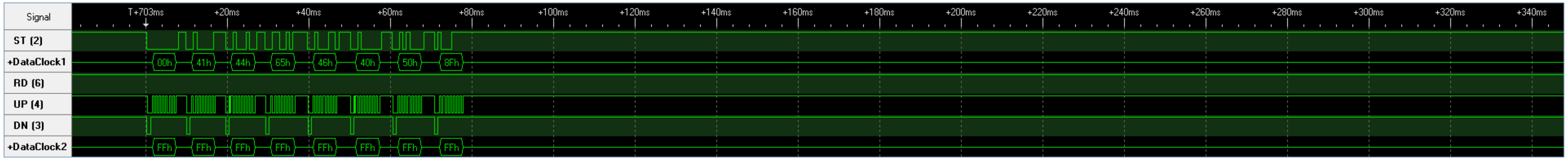

Signal
-160ms -140ms -120ms -100ms -80ms -60ms -40ms -20ms T+180.2ms +20ms +40ms +60ms +80ms +100ms +120ms +140ms +160ms
ST (2)
+DataClock1
FFh FFh 04h 40h 80h 09h 05h 90h 09h 05h 80h
RD (6)
UP (4)
DN (3)
+DataClock2
03h 80h FFh FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 800us Interval B->C: 600us Interval C->F: -5.8ms Rate D->E: 1.25KHz
IF20.LPF 05/11/2012 21:07:22

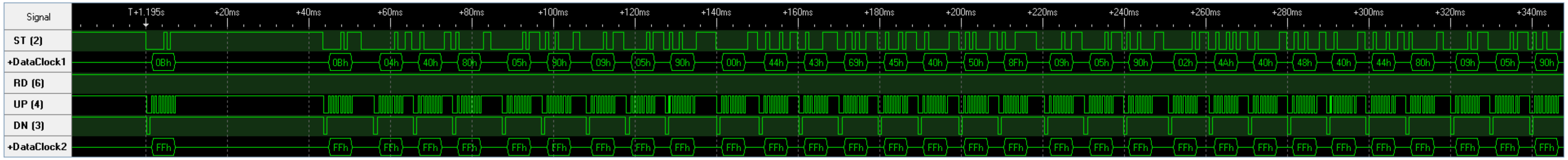
IF-20 initialization (when pressing ENT then the POWER button on the RC-20):



Interval A->B: 800us Interval B->C: 600us Interval C->F: -5.8ms Rate D->E: 1.25KHz
IF20.LPF 05/11/2012 21:14:31



Interval A->B: 800us Interval B->C: 600us Interval C->F: -5.8ms Rate D->E: 1.25KHz
IF20.LPF 05/11/2012 21:16:17


Signal
T+1.195s
+20ms
+40ms
+60ms
+80ms
+100ms
+120ms
+140ms
+160ms
+180ms
+200ms
+220ms
+240ms
+260ms
+280ms
+300ms
+320ms
+340ms
ST (2)
+DataClock1
RD (6)
UP (4)
DN (3)
+DataClock2
0Bh 0Bh 04h 40h 80h 05h 90h 09h 05h 90h 00h 44h 43h 69h 45h 40h 50h 8Fh 09h 05h 90h 02h 4Ah 40h 48h 40h 44h 80h 09h 05h 90h
FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

Interval A->B: 800us Interval B->C: 600us Interval C->F: -5.8ms Rate D->E: 1.25KHz
IF20.LPF 05/11/2012 21:16:34

Note the timing from T+0 (when the protocol analyser was triggered) in the upper left of each part of the capture.

IF-20 Commands list (link with the RC-20)

From IF-20 to RC-20				From RC-20 to IF-20			
05 8x or 05 9x	Bargraph level – When setting the VOL or SQL level x from 0 to 7			04	Byte sent before every command to the SUB transceiver		
09	Byte sent before every frame from the SUB transceiver						
09 05 80 or 09 05 90	Data sent after every frame when a SUB transceiver is connected						
04 40 80	Data sent when the IF-20 is powered up by the RC-20 or reset (when hot plugging or unplugging a transceiver)						
04 41 80	Data sent after pressing **DUAL** on the RC-20, when the SUB transceiver is turned off						
0B	IF-20 Initialized or reset						

The drafting of this part of the document is still in progress, please come back to check for updates in a few weeks (see link on page 1).

Thanks

I had this hack project in mind for years but I didn't go further because I could not get any of these devices used for remote controlling the TM-xx1 transmitters.
I've found N9XLC's blog who was also working on this hack, in an umpteenth attempt to control my own transceivers, during 2012, but despite all the work he already had achieved, I wanted to know the whole commands set.
All this was made possible thanks to Peter who sold me his RC-10, François (ON4NYO) from whom I got an unused RC-20 and JBE who almost gave their new IF-20.
TM-5x1 specific commands have been added to this documents thanks to jun's researches.
TM-221, TM-421, TM-231 and TM-431 tests have been preformed by Jonathan from radiorework.com, using jun's arduino interface.

Projects list

Here the list of projects that have been carried out by using this document :
https://jun930.hatenablog.com/entry/2019/06/16/213556 (use the google translate toolbar on the top of the page)

If you want your project to be listed here, send me an e-mail at: blog@shibby.fr or post a message on my blog.

Notes

Various documents regarding the transceivers and interfaces can be found here.
I am still looking for service manuals for the TM-441, IF-20 and RC-20. You can send me PDF, DjVu or any scan you have at: blog@shibby.fr