Universal platform for digital amateur radio (UP4DAR)

Draft 0.2

20.06.2012

Issued by

Denis Bederov (DL3OCK)

QTH: Berlin

QRL: Salzburg

Contents

1 GENERAL INFORMATION....4

1.1 History....4
1.2 Main Authors....4
1.3 References....5
1.4 Abbreviations and definitions....6

2 PLATFORM OVERVIEW....7

2.1 Hardware architecture....7
2.2 Use cases....8
2.2.1 Single user operation....8
2.2.2 Digipeater operation....8
2.3 Electrical characteristics....9
2.3.1 Connection to a TRX....9

3 SPECIFICATION OF UP4DAR INTERFACE....11

3.1 States of PHY....11
3.1.1 Service mode....12
3.1.2 Single user mode @ DSTAR-PHY....12
3.1.3 Digipeater mode @ DSTAR-PHY....13
3.1.4 DVR-mode @ DSTAR-PHY....14
3.1.5 APRS-PHY....15
3.1.6 PR9K6-PHY....15
3.2 Parameters of PHY....16
3.3 Command overview....17
3.4 Command transmission order....17
3.5 Command specification....17
3.5.1 sysInfo....18
3.5.2 SysParam....18
3.5.3 reqSysParam....19
3.5.4 setSysParam....19
3.5.5 uploadVoiceWarning....20
3.5.6 set_QRG....21
3.5.7 ind_QRG....21
3.5.8 modeInfo....22
3.5.9 set_mode....22
3.5.10 cmd_exe_ind....23
3.5.11 tx_on....24
3.5.12 tx_off....24
3.5.13 send_header....24
3.5.14 send_voice....25
3.5.15 send_data....26
3.5.16 change_duration....26
3.5.17 rcvd_header....27
3.5.18 rcvd_voice....27
3.5.19 rcvd_syncFlag....28
3.5.20 rcvd_data....28
3.5.21 rcvd_termFlag....29
3.5.22 rcvd_FrameSync....29
3.5.23 tx_test....30
3.6 Procedures....31

3.6.1 Power On procedure....31
3.6.2 Transmitting procedure....31
3.6.3 Aborted transmitting procedure....33
3.6.4 Receiving procedure in SUM....35
3.6.5 Receiving procedure in DM....36
3.6.6 Procedure for receiving of unwanted signal in DM....37
3.6.7 Procedure for receiving of corrupted signal in DM....38
3.6.8 MATFST expiry procedure....41
3.6.9 PHY update procedure....43

4 ANNEX....44

4.1 Frame structure of DV packet in D-STAR....44
4.2 Description of D-STAR application layer....46
4.2.1 Mapping of LSDC....46
4.2.2 Empty voice data channel....47
4.3 Proposals for desktop software....51
4.3.1 GUI....51
4.4 Proposal for a low delay inter digipeater transmission....52

1 General information

Currently there is no a lot solutions on market for the new digital voice operation mode in amateur radio. Especially the creating of innovation in such new and interesting area is not really easy by today.

The target for the development of the universal platform for digital amateur radio (UP4DAR) is to create a very flexible platform with an open and good described interface. In my opinion this is a needed pre condition for enabling of a lot software talented YLs and OMs under us to create an always growing open source project, which should be open for all innovations we will have in the next future.

The main goal of this document is to encourage and to enable a wide number of people interesting in digital amateur radio to develop software for the OS (definition follows in the next sections), desktop software for comfortable work (in case a computer is connected), each kind of back bone routing software for world wide interconnections (e.g. www.xreflector.net) and some really innovative things that there is not by today. I am hinting at new “network services” (e.g. digital routeable voice mailboxes, digipeater with special ports not for a voice communication but with special measuring functionality, which should report back to the user, which physical parameters would be more better to set in his hardware configuration menu, etc...).

1.1 History

Version	Date	Chapter	Changes / Reasons
0.1	29.11.09	all	DL3OCK, initial version started on 04.11.2009
0.2	20.06.11	all	DL3OCK

Rule for version number:

- next integer number for major changes
- + 0.1 for minor changes

1.2 Main Authors

Denis Bederov (DL3OCK); Denis.Bederov@gmx.de

1.3 References

[STD4]	http://www.jarl.com/d-star/shiryou/STD4_3C.pdf
[shogen]	http://www.jarl.com/d-star/shogen.pdf
[dvsi]	AMBE-2020TM Vocoder Chip , User's Manual , Version 4.8 September 2007 http://www.dvsinc.com/products/a2020.htm
[APRS]	APRS Protocol Reference – APRS Protocol Version 1.0
[3GPP]	3GPP TS 23.032 V8.0.0 (2008-12) Universal Geographical Area Description (GAD) http://www.3gpp.org/ftp/Specs/html-info/23032.htm

1.4 Abbreviations and definitions

AF	audio frequency
APRS	Automatic Position Reporting System is the trademark of Bob Bruninga, WB4APR
digipeater	digital repeater
D-STAR	Digital Smart Technologies for Amateur Radio
DV	digital voice
FEC	forward error protection
GMSK	Gaussian minimum shift keying
HF	high frequency
LO-QRG	frequency of local oscillator
LSB	least significant byte
LSDC	low speed data channel
NMEA	National Marine Electronics Association
PLL	phased lock loop
PTT	push to talk
RX-QRG	receiving frequency
t.b.d.	to be defined
UP4DAR	Universal platform for digital amateur radio

2 Platform overview

2.1 Hardware architecture

The universal platform for digital amateur radio (UP4DAR) is a very flexible, high integrated platform which allows to host many different digital communication procedures from the amateur radio area. The UP4DAR consists of two important parts:

- Intelligent physical layer functionality (PHY)
- Operating System functionality (OS)

On the PHY the very time critical signal processing of incoming noisy AF signal is implemented. The PHY is connected to the amateur radio equipment by the traditional and well known data Mini-DIN 6 jack, which many transceiver by today have.

The OS has a lot of purposeful components which should allow to implement many comfortable software applications with new innovative ideas.

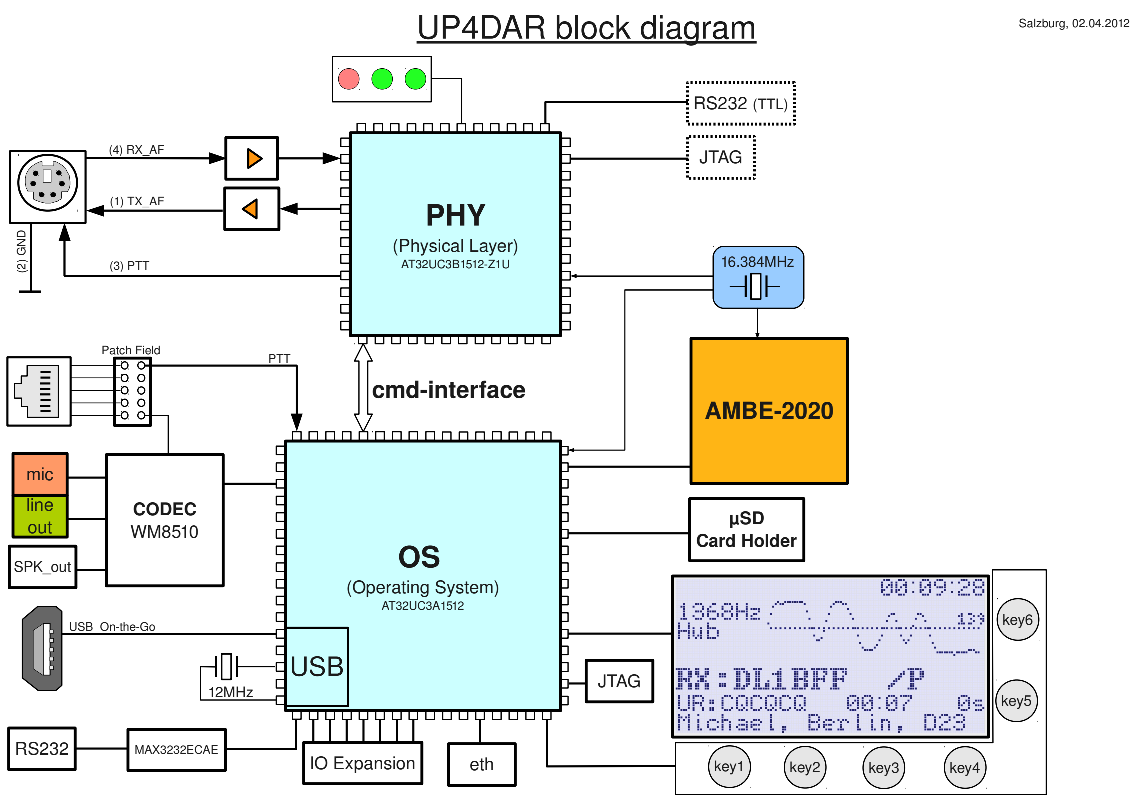

Figure 1: UP4DAR block diagram

2.2 Use cases

2.2.1 Single user operation

The most common use case would be the usage of this universal platform as an end user equipment. At the following figure the corresponding scenario is shown. As already mentioned above the physical layer does not need any external squelch information. The internal state machine observed the incoming noisy AF signal and estimates the fact of any ongoing transmission. So the best sensitivity can be achieved if the available squelch is completely open!

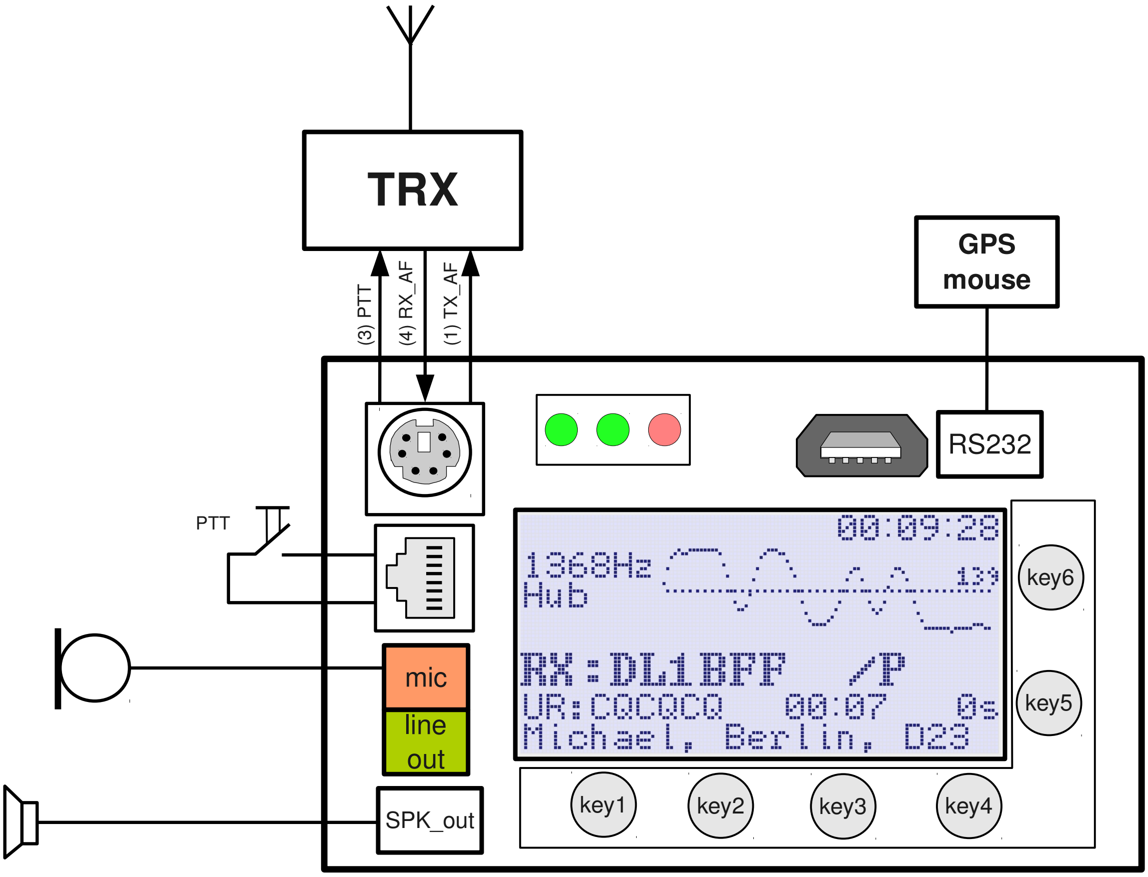

Figure 2: UP4DAR as a single user equipment

2.2.2 Digipeater operation

The UP4DAR can be also used for creating of a digital repeater. The PHY provides for this purpose some helpful feature. This allows to make a repeater on very simple way by using of a minimum of external components. The following figure shows how a digipeater can looks like.

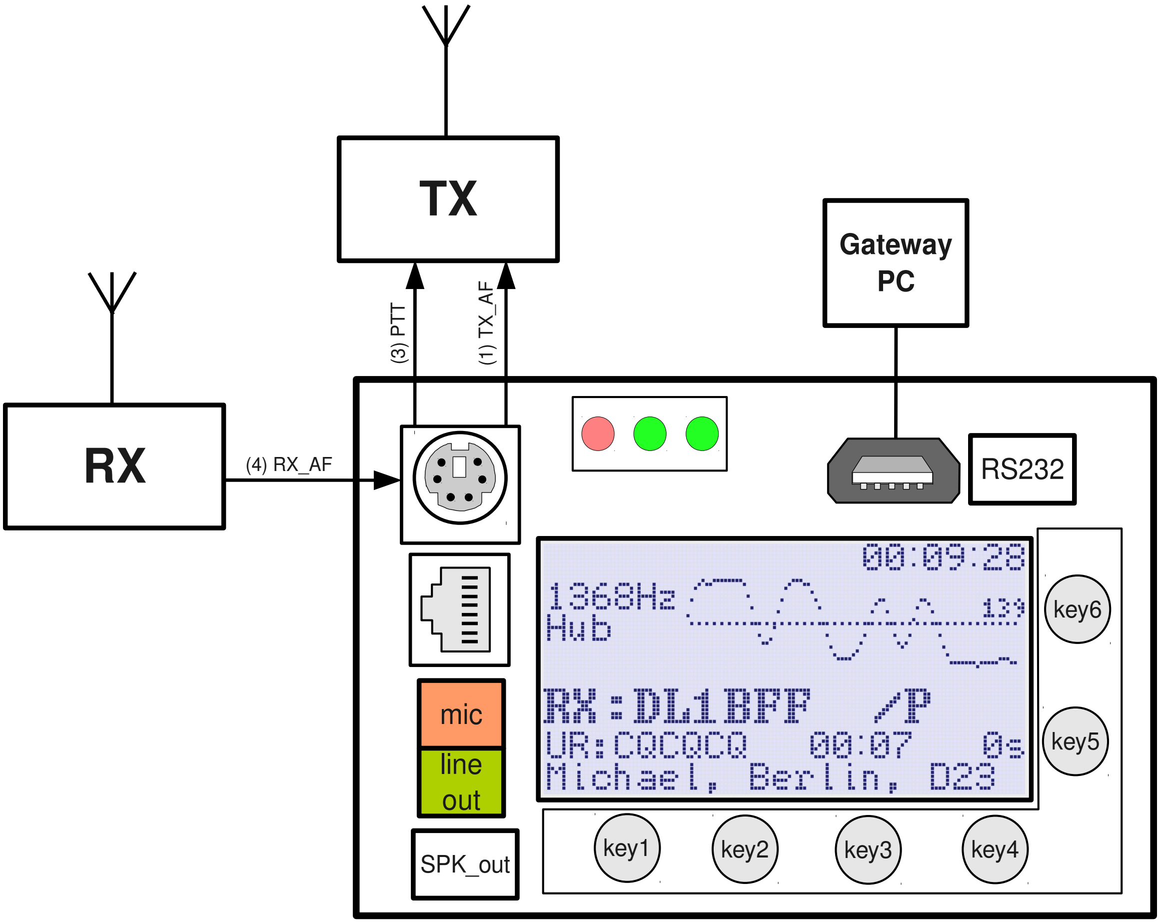

Figure 3: UP4DAR as a digipeater

2.3 Electrical characteristics

2.3.1 Connection to a TRX

The analog connection towards an amateur transceiver is a usual Mini-DIN 6 jack, that a lot of transceivers of today have.

The AF signal from receiver, which is connected to the pin 4, should be not higher than 660mVpp. On the input of the pin 4 inside is a series connection of a 4.7µF capacitance and a 11KOhm resistor. The “cold site” of the capacitance is negative, so the constant component of the input signal must be smaller than 1.3V.

The maximal possible output AF signal at pin 1 has 2400mVpp. This output has a series capacitance of 4.7µF with a negative “cold site”, so the input of the transmitter must be DC free.

Neither input nor output have any trimmer, so the level is controlled by software parameter (see 3.2) only! This should allow in the OS application software to store configuration for each particular amateur radio device. So a particular profile can be loaded ad hoc by connection to different transceiver (e.g. fixed station, transceiver built in the car or a portable transceiver). Such configuration file can be also exchanged with other OMs using the same transceiver.

Thank to a smart switching power supply concept the voltage of the power supply for UP4DAR can vary in a wide range from 4V to 20V.

NOTE! The 5V supply is only working, if input power supply is in the range 6V...20V.

NOTE! The 8V supply is only working, if input power supply is in the range 8.5V...20V.

3 Specification of UP4DAR interface

The “PHY” can be considered as a kind of a well known GMSK chip CMX589 extended with a couple of features. So we can now speak about an intelligent physical layer, which is closed source. In order to be able to develop the software for the operating system (OS) at UP4DAR the physical layer should be considered as a black box with strongly defined behaviour and well known interface. In this chapter the interface will be presented in detail.

3.1 States of PHY

After power supply is switched on the physical layer starts to boot. After this very fast procedure is finished the PHY goes into **`service-mode`**. In the further operation also other modes are possible. Each operational mode of PHY can have different states. In the following figures all states are described in more detail.

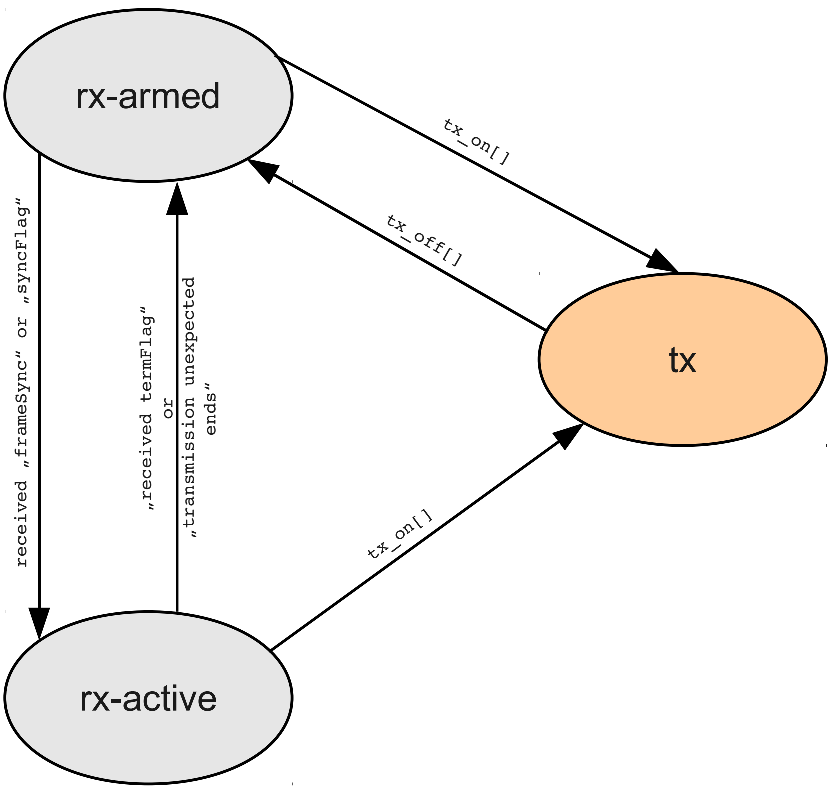

Figure 4: states of UP4DAR in DSTAR-PHY-SUM mode

3.1.1 Service mode

As already mentioned the PHY goes in this state immediately after its has booted after the power was switched on. In this state some parameter of PHY (see 3.2) can be configured from outside via the UP4DAR interface by using of special commands described in the next sections. In order to adjust the level of transmitted AF signal some test sequences can be transmitted. Also the input of the PHY can be adjusted to the polarity of received AF-signal.

3.1.2 Single user mode @ DSTAR-PHY

This mode is basically used in the most cases together with amateur radio transceiver for a direct QSO or operation via a digital repeater (digipeater).

*3.1.2.1 **`rx-armed`** state*

In this state the PHY stays its most time and investigates the incoming noisy signal if there is a valid data transmission. It does mean that the PHY does not need any external squelch circuit. Therefore the best idea it is to open completely the squelch (if available). As soon a valid data transmission is detected the PHY goes to the next state.

*3.1.2.2 **`rx-active`** state*

Being in this state the PHY provides to the OS the received voice data and data of low speed data channel (see 4).

*3.1.2.3 **`tx`** state*

If the user want to make a digital voice transmission by activation of its PTT the OS enforces the PHY to go into the transmitting (**`tx`**) state. Being in this state the PHY get the data to be transmitted and creates a corresponding analogue GMSK AF signal.

*3.1.2.4 **`tx-termFlag`** state*

Should an ongoing transmission be finished the user simply deactivates the PTT. Thereupon the OS enforces the PHY to go into the **`tx-termFlag`** state in which the PHY sends the terminating flag and deactivates the HF transmitter. Shortly after this the PHY changes to the **`rx-armed`** state.

3.1.3 Digipeater mode @ DSTAR-PHY

In this mode the PHY of the UP4DAR has all basic features needed to make a voice digital repeater according to D-STAR (see 4). A potential SysOp needs only to have a traditional analogue repeater equipment, in which the receiver has a data capable AF output and the transceiver a data capable AF input (see 2.2.2).

In this mode the PHY has the same states like in the SUM **`rx-armed`**, **`rx-active`**, **`tx`**, **`tx-termFlag`** with the same meaning as it was presented above. Additionally the digipeater mode (DM) has following states.

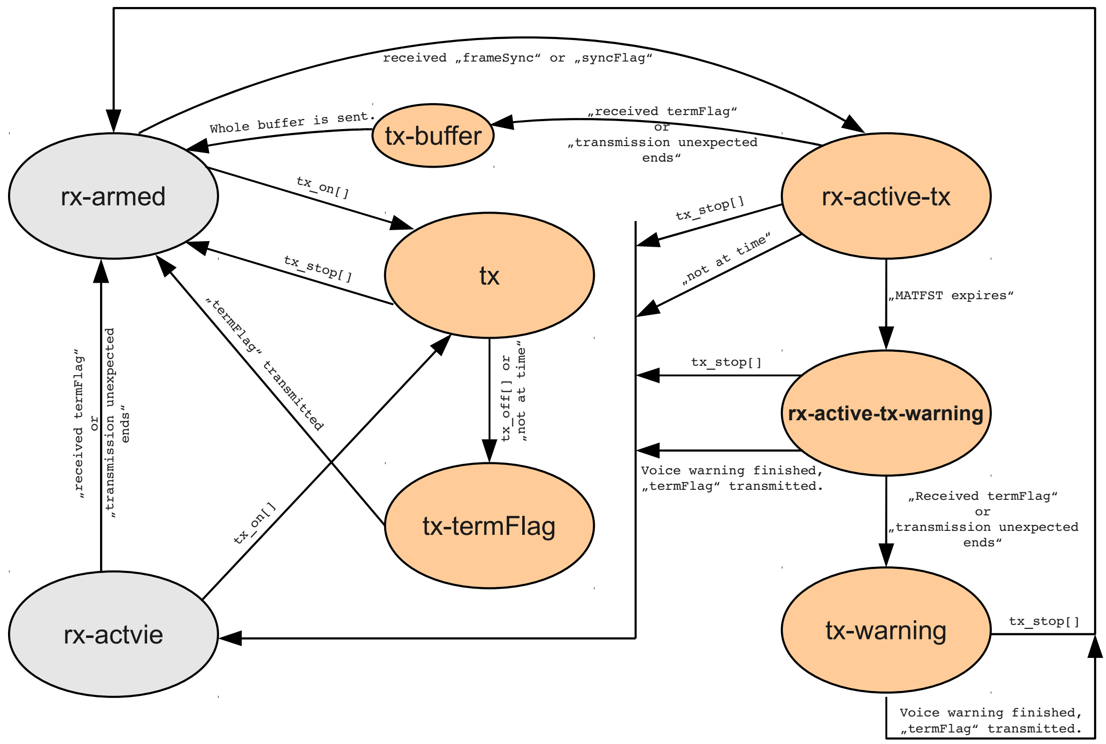

Figure 5: states of UP4DAR in DSTAR-PHY-DM mode

*3.1.3.1 **`rx-active-tx`** state*

This is most native state of a repeater, in which a valid data transmission is (almost) directly retransmitted immediately by PHY to the transmitter and in addition provided to the OS for further processing (e.g. sending of voice data and data of low speed data channel to a gateway function for establishing of an inter repeater operation).

*3.1.3.2 **`tx-buffer`** state*

In this state the PHY goes if a receiving transmission is finished in order to send out data still buffered (due to a time delay between incoming and outgoing digital signal).

3.1.3.3 ***rx-active-tx-warning*** *state*

If the maximal allowed time for a single transmission in repeater mode is configured to be limited and this time was expired during a long transmission the PHY goes in this state. Here the PHY still provides received data to the OS but does not retransmit them to the AF output. Instead of received signal the PHY sends a short voice warning message (e.g. "*Here is DB0DF. The last transmission was too long!*" or simply "*transmission too long!*"). The data of the voice warning message can be uploaded into the PHY during configuration procedure.

3.1.3.4 ***tx-warning*** *state*

In this state the PHY goes if a too long transmission was finished during the voice warning is transmitting in order to finalize the completely voice warning message.

3.1.4 DVR-mode @ DSTAR-PHY

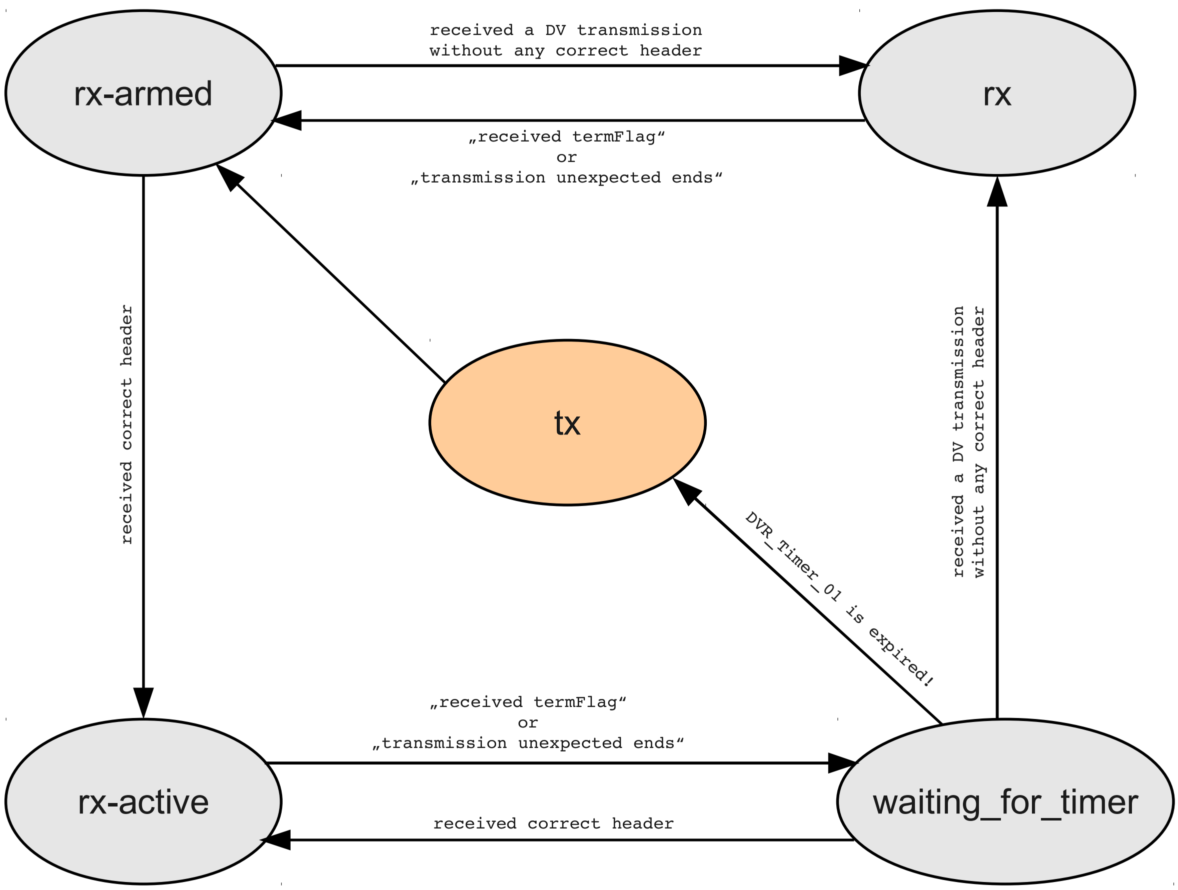

Figure 6: states of UP4DAR in DSTAR-PHY-DVR mode

3.1.4.1 `rx-armed` state

3.1.4.2 `rx-active` state

3.1.4.3 `waiting-for-timer` state

3.1.4.4 `rx` state

3.1.4.5 `tx` state

t.b.d.

3.1.5 APRS-PHY

t.b.d.

3.1.6 PR9K6-PHY

t.b.d

3.2 Parameters of PHY

The physical layer can be optimised to his environment by configuration of following parameters:

Parameter name	Parameter ID	Value(s)	Description
TX_DELAY	0x01	0..255	Transmitter delay time between PTT activation and transmission of information data in 8bit steps (additionally to 64 mandatory bits!). For a D-STAR voice frame this means that after activation of PTT the sequence {1,0,1,0,1,0,...,1,0} of length TX_DELAY*8+64 in front of FrameSync flag is transmitted.
TX_GAIN	0x02	-128..-1, 1,..127	Linear factor for transmitted AF signal. The maximal range of DAC is achieved if TX_GAIN = 127. By using of negative values of TX_GAIN the output AF signal can be inverted, if it is needed.
RX_inv	0x03	0,1	If the incoming AF signal is inverted (due to an inverting amplifier some where in receiver or if in a IF-mixer LO-QRG > RX-QRG for instance) RX_inv should be set to 1.
TX_DC_SHIFT	0x04	-128..127	This parameter describes the linear offset of transmitted signal. It is relevant for DC-coupled transmitter only.
MATFST	0x05	0..255	Maximal allowed time for a single transmission in digipeater mode in 2sec steps. The time is not limited if MATFST=0 .
Length_of_VW	0x06	1..255	Number of 20ms blocks of recorded voice warning message for user information about MATFST expiry.

3.3 Command overview

Command ID	Service Mode		DSTAR-PHY-SUM			DSTAR-PHY-DVR					DSTAR-PHY-DM				
	ready	tx	rx-armed	rx-active	tx	rx-armed	rx-active	rx	Waiting-for-timer	tx	rx-armed	rx-active	rx		tx
0x01	SysInfo[] / req_SysInfo[]														
0x10	tx_test[]		tx_on[]												
0x11		tx_off[]			tx_off[]										
0x20					send_header[]										
0x21					send_voice[]										
0x22					send_data[]										
0x23					change_duration[]										
0x30				rcvd_header[]		rcvd_header[]			rcvd_header[]						
0x31				rcvd_voice[]			rcvd_voice[]								
0x32				rcvd_SyncFlag[]											
0x33				rcvd_data[]			rcvd_data[]								
0x34				rcvd_termFlag[]			rcvd_termFlag[]								
0x35			rcvd_FrameSync[]						rcvd_FrameSync[]						
0x40	SysParam[]														
0x41	req_SysParam[]														
0x42	set_SysParam[]														
0x44	set_QRG[]														
0x45	ind_QRG[]														
0xD1	modeInfo[]		modeInfo[]			modeInfo[]									
0xD3	set_mode[]		set_mode[]			set_mode[]									
0xD4	cmd_exe_ind[]		cmd_exe_ind[]			cmd_exe_ind[]									
0xE1	start_update[]														
0xE2	transfer_update_block[]														
0xE3	update_cmplt[]														
0xE4	rdy_for_update[]														

3.4 Command transmission order

Alle messages on command interface are encapsulated in the structure <DLE><STX> ....<DLE><ETX>, with the usual values as follow:

<DLE> = 0x10

<STX> = 0x02

<ETX> = 0x03

In case that the encapsulated payload has a Byte 0x10 (which is equal to <DLE>), then this byte 0x10 should be doubled.

All commands as well responses are transmitted serial over the UP4DAR command interface byte wise. Each particular byte is transmitted with LSB first.

<u>Example:</u>

`modeInfo[mode=service_mode, status=ready]`

is transmitted as **`0x10,0x02,0xD1,0x01,0x00,0x10,0x03.`**

For the physical transfer one of typical USART ports is used. The configuration of this port is [8N1, LSB first, 115200Baud]

3.5 Command specification

In the following section the syntax of all commands is described in detail.

3.5.1 sysInfo

Syntax: **`sysInfo[version]`**

Length of the command: 70Byte.

Parameter	Value(s)	Description
Command name	**`sysInfo`**	This command informs the OS about the version of currently used PHY software. This commands is only sends immediately after PHY was powered on. This command can also be sent to the PHY as request at any time to get the sysInfo!
Command ID	**`0x01`**	
version	**`byte1,..., byte70`**	Last 15 Bytes are the ATMEL's unique serial number of the UC3B-Chip

3.5.2 SysParam

Syntax: **`SysParam[parameter, value]`**

Length of the command: 3Byte.

Parameter	Value(s)	Description
Command name	**`SysParam`**	This command shows the content of any particular parameter.
Command ID	**`0x40`**	
parameter	**`0x01`** = `TX-DELAY` **`0x02`** = `TX-GAIN` **`0x03`** = `RX-inv` **`0x04`** = `TC-DC-SHIFT` **`0x05`** = `MATFST` **`0x06`** = `Length_of_VW`	
value	`byte1`	

3.5.3 reqSysParam

Syntax: **`reqSysParam[parameter]`**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**`reqSysParam`**	This command requests the information about any particular parameter.
Command ID	**`0x41`**	
parameter	**`0x01`** = `TX-DELAY` **`0x02`** = `TX-GAIN` **`0x03`** = `RX-inv` **`0x04`** = `TX-DC-SHIFT` **`0x05`** = `MATFST` **`0x06`** = `Length_of_VW`	

3.5.4 setSysParam

Syntax: **`setSysParam[parameter, value]`**

Length of the command: 3Byte.

Parameter	Value(s)	Description
Command name	**`setSysParam`**	This command allows to change any particular parameter.
Command ID	**`0x42`**	
parameter	**`0x01`** = `TX-DELAY` **`0x02`** = `TX-GAIN` **`0x03`** = `RX-inv` **`0x04`** = `TX-DC-SHIFT` **`0x05`** = `MATFST` **`0x06`** = `Length_of_VW`	
value	`byte1`	

3.5.5 uploadVoiceWarning

Syntax: **`uploadVoiceWarning[Length_of_VW, voice_message]`**

Length of the command: depending on **`Length_of_VW`**.

Parameter	Value(s)	Description
Command name	**`uploadVoiceWarning`**	This command allows to upload into PHY the voice data for a voice warning about expiry of maximal allowed time for single transmission.
Command ID	**`0x43`**	
Length_of_VW	`1..255`	Number of 20ms blocks of recorded voice warning message for user information about MATFST expiry.
voice_message	`byte1..byteN`	N = Length_of_VW * 9

3.5.6 set_QRG

Syntax: **`set_QRG[rx_QRG, tx_QRG]`**

Length of the command: 9Byte.

Parameter	Value(s)	Description
Command name	**`set_QRG`**	This command allows to set the transmission and receiving frequency of a radio part, which is directly connected to the PHY.
Command ID	**`0x44`**	
rx_QRG	`byte1..byte4`	**`unsigned int rx_QRG;`** byte1 = MSB, ..., byte4=LSB
tx_QRG	`byte1..byte4`	**`unsigned int tx_QRG;`** byte1 = MSB, ..., byte4=LSB

3.5.7 ind_QRG

Syntax: **`ind_QRG[set_QRG_resp]`**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**`ind_QRG`**	This command responses if the seting of new frequences was successful or not.
Command ID	**`0x45`**	
set_QRG_resp	**`0x00`** `= both QRGs set up` **`0x01`** `= tx-QRG is out of allowed range` **`0x02`** `= rx-QRG is out of allowed range` **`0x03`** `= PLL is not able to lock tx-QRG` **`0x04`** `= PLL is not able to lock tx-QRG`	

3.5.8 modeInfo

Syntax: **modeInfo[mode, state]**

Length of the command: 3Byte.

Parameter	Value(s)	Description
Command name	**modeInfo**	This command provides the information about the current operation mode / state.
Command ID	**0xD1**	This id is from the mode independent range [0xD1...0xFF, 0x00]
mode	**0x01** = service-mode **0x02** = DSTAR-PHY-SUM **0x03** = DSTAR-PHY-DM **0x04** = DSTAR-PHY-DVR **0x05** = APRS-PHY **0x06** = PR9K6-PHY **0x07** = F3E-PHY	
state	**0x00** = ready **0x01** = rx-armed **0x02** = rx-active **0x03** = rx **0x04** = tx **0x05** = tx-termFlag **0x06** = rx-active-tx **0x07** = tx-buffer **0x08** = rx-active-tx-warning **0x09** = tx-warning **0x0A** = waiting_for_timer	"ready" is used in "service-mode" only.

3.5.9 set_mode

Syntax: **set_mode[mode]**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**set_mode**	This command enforces the PHY to change in some particular operation mode.

Command ID	**0xD3**	This id is from the mode independent range [0xD1...0xFF, 0x00]
mode	**0x01** = service-mode **0x02** = DSTAR-PHY-SUM **0x03** = DSTAR-PHY-DM **0x04** = DSTAR-PHY-DVR **0x05** = APRS-PHY **0x06** = PR9K6-PHY **0x07** = F3E-PHY	

3.5.10 cmd_exe_ind

Syntax: **`cmd_exe_ind[execution-status]`**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**cmd_exe_ind**	This command provides the feedback about the execution of recently invoked command.
Command ID	**0xD4**	This id is from the mode independent range [0xD1...0xFF, 0x00]
execution_status	**0x01** = successful execution **0x02** = command is not specified or can not be used in the current situation **0x03** = syntax error **0x04** = command queue of PHY was busy	

3.5.11 tx_on

Syntax: **`tx_on[]`**

Length of the command: 1Byte.

Parameter	Value(s)	Description
Command name	**`tx_on`**	This command starts a regular DV transmission.
Command ID	**`0x10`**	

3.5.12 tx_off

Syntax: **`tx_off[]`**

Length of the command: 1Byte.

Parameter	Value(s)	Description
Command name	**`tx_off`**	This command stops an ongoing DV transmission on a regular way by appending of the terminating flags.
Command ID	**`0x11`**	

3.5.13 send_header

Syntax: **`send_header[header_bytes]`**

Length of the command: 40Byte.

Parameter	Value(s)	Description
Command name	**`send_header`**	This command provides to the PHY all needed bytes for a header transmission according to D-STAR protocol specification.
Command ID	**`0x20`**	
header_bytes	**`byte1,..., byte39`**	

3.5.14 send_voice

Syntax: **send_voice[VFI, voice_bytes]**

Length of the command: depending on **VFI**.

Parameter	Value(s)	Description
Command name	**send_voice**	This command provides to the PHY encoded bytes of 20ms speech period.
Command ID	**0x21**	
VFI	**0x01** = 2400/1200 BC FEC **0x02** = 2400	See [dvsi] section 5.2.4 for further information. In current release the voice format indicator (VFI) = (2400/1200 BC FEC) is supported only.
voice_bytes	(VFI = **0x01**): byte1,..., byte9	

3.5.15 send_data

Syntax: **`send_data[data]`**

Length of the command: 4Byte.

Parameter	Value(s)	Description
Command name	**`send_data`**	This command provides to the PHY the data bytes of the low speed data channel for current 20ms.
Command ID	**`0x22`**	
data	**`byte1,..., byte3`**	

3.5.16 change_duration

Syntax: **`change_duration[n]`**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**`change_duration`**	This command allows to overcome the slip problem by faster or slower provision of information to send during the transmission. Usage of this command in front of each SyncFlag can adjust the own transmission up to +-99ppm in comparison to 4800Symbols per second!
Command ID	**`0x23`**	
n	`0x01 = duration 2x(-10%)` `0x02 = duration 2x10%`	This command can be used once in between of two SyncFlag. Two last symbols in front of following SyncFlag will be ether extended or shorted by 10% of its regular duration.

3.5.17 rcvd_header

Syntax: **rcvd_header[crc_result, header_bytes]**

Length of the command: 41Byte.

Parameter	Value(s)	Description
Command name	**rcvd_header**	This command provides the received and decoded header payload.
Command ID	**0x30**	
crc_result	**0x00** = CRC was OK **0x01** = bad CRC	
header_bytes	**byte1,..., byte39**	For the mapping of the header byte see [STD4], [shogen] or 4.1.

3.5.18 rcvd_voice

Syntax: **rcvd_voice[VFI, voice_bytes]**

Length of the command: depending on **VFI**.

Parameter	Value(s)	Description
Command name	**rcvd_voice**	This command provides the received voice data.
Command ID	**0x31**	
VFI	**0x01** = 2400/1200 BC FEC **0x02** = 2400 **0x20** = 2400/1200 BC FEC SD	See [dvsi] section 5.2.4 for further information. In current release the voice format indicator (VDF) = (2400) is not supported.
voice_bytes	(VFI = **0x01**): byte1,..., byte9 (VFI = **0x20**): byte1,..., byte36	

3.5.19 rcvd_syncFlag

Syntax: **`rcvd_syncFlag[time_correction]`**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**`rcvd_syncFlag`**	This command indicates that a sync flag was received and provides the done time correction in respect to the recently received sync flag before.
Command ID	**`0x32`**	
time_correction	**`tc_byte`**	**`0`** <= tc_byte <= **`13`** The time correction value gives a result of the measured time difference between the received Synchronisation Flag and its ideal time point. Based on this measurement it is possible to calculate the relative time/frequency error between the transmitter and the receiver crystal according to the following formula: **error_in_ppm**:= (tc_byte-7)*84

3.5.20 rcvd_data

Syntax: **`rcvd_data[n, data]`**

Length of the command: 5Byte.

Parameter	Value(s)	Description
Command name	**`rcvd_data`**	This command provides the received data of the low speed data channel.
Command ID	**`0x33`**	
n	**`1..20`**	Number of current received block counted since the last sync flag.
data	**`byte1,..., byte3`**	

3.5.21 rcvd_termFlag

Syntax: `rcvd_termFlag[]`

Length of the command: 1Byte.

Parameter	Value(s)	Description
Command name	`rcvd_termFlag`	This command indicates that a termination flag was received.
Command ID	`0x34`	

3.5.22 rcvd_FrameSync

Syntax: `rcvd_FrameSync[mean, deviation_x2, samples]`

Length of the command: 93Byte

Parameter	Value(s)	Description
Command name	`rcvd_FrameSync`	This command provides the analogue samples of received FrameSync-Flag.
Command ID	`0x35`	
mean	`0..0xFF`	Mean value of the received signal
deviation_x2	`0..0xFF`	The doubled peak deviation of the received signal (calculated based on FrameSync).
samples	`byte1,..., byte90`	The frame sync flag consist of 15 symbols which are represented by 6 samples per symbol. `0x00` means ground level and `0xff` means VCC level of RX_AF (pin4).

3.5.23 tx_test

Syntax: **`tx_test[sequence]`**

Length of the command: 2Byte.

Parameter	Value(s)	Description
Command name	**`tx_test`**	This command enforces the PHY to transmit some predefined periodical signals. This is very helpful for adjustment activity by connection of UP4DAR to a new amateur radio equipment.
Command ID	**`0x10`**	
sequence	**`0x01`** = “1111100000...” **`0x02`** = PRN9	This test data sequences are transmitted with 9600Baud!

3.6 Procedures

In this section all important procedures are presented as formal message flows.

3.6.1 Power On procedure

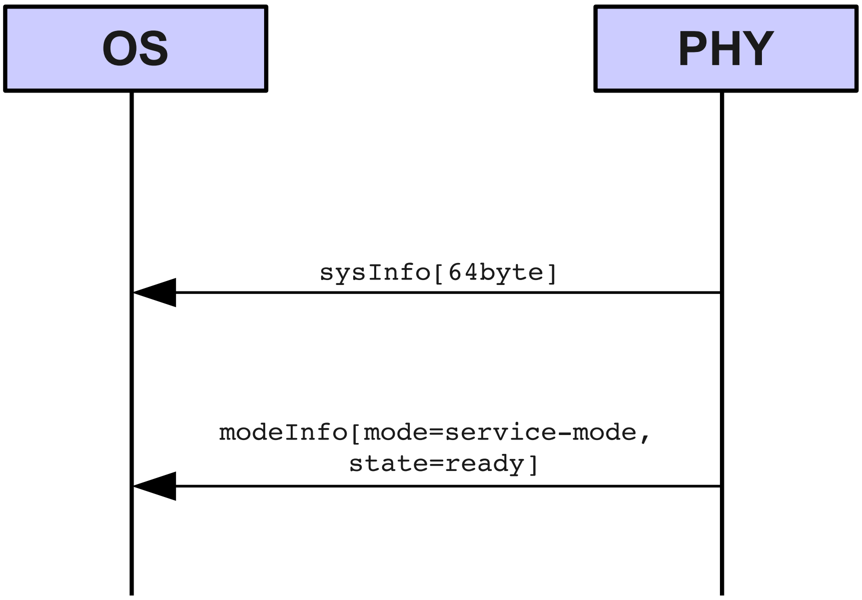

Figure 7: Power On procedure

After the power on was switched on the PHY starts its operation. After the boot process is finished the PHY sends a **`sysInfo`** with a string of 70 bytes and a **`modeInfo`** message as shown in the figure above.

3.6.2 Transmitting procedure

In the DSTAR-PHY-SUM as well in DSTAR-PHY-DM the transmitting procedure is executing on the same way as shown in the figure below. The main work for creating the output AF signal is done by PHY so the program on the OS does not need to take care about. However it is very important to fulfil the timing requirements of a D-STAR DV frame, which is shown with blue arrow on the right site of the PHY instance.

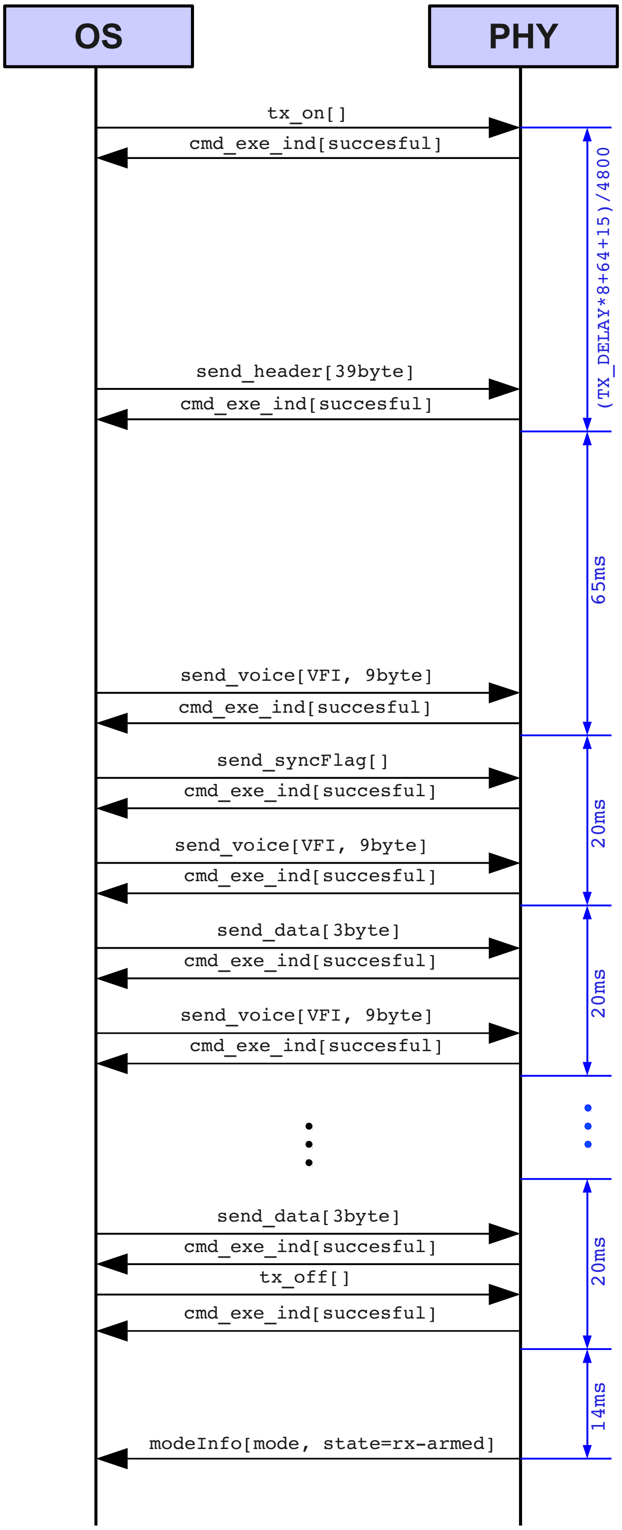

Figure 8: transmitting procedure

3.6.2.1 Abnormal behavior

As soon the OS does not provide the next portion of needed data during the transmitting procedure at time the PHY finishes the transmission on correct way. The OS will be informed by indicating of change into the corresponding state. The PHY will reject all further data coming not at time.

3.6.3 Aborted transmitting procedure

This procedure is executed on very native way and does not really need to be mentioned in an extra section. The only reason for this section is to give an idea how the internally command pipeline of the PHY works.

The assumption here is that it is going about some kind of voice mail application or transmitting of any information, which was already recorded and stored. In such a case the OS can provide to the PHY all the voice (and LSDC) data in very short time interval much shorter than the corresponding “play” time. This data can be easily received and stored in the PHY. However the real time transmitting will of course need the corresponding “play” time anyway.

If in such a scenario the OS decides whatever reasons to stop the transmission it simply sends the `tx_stop[]` command to the PHY. In this case the PHY immediately stops the transmission, deletes the data buffer and goes in the `rx-armed` state. This procedure is shown in the following figure.

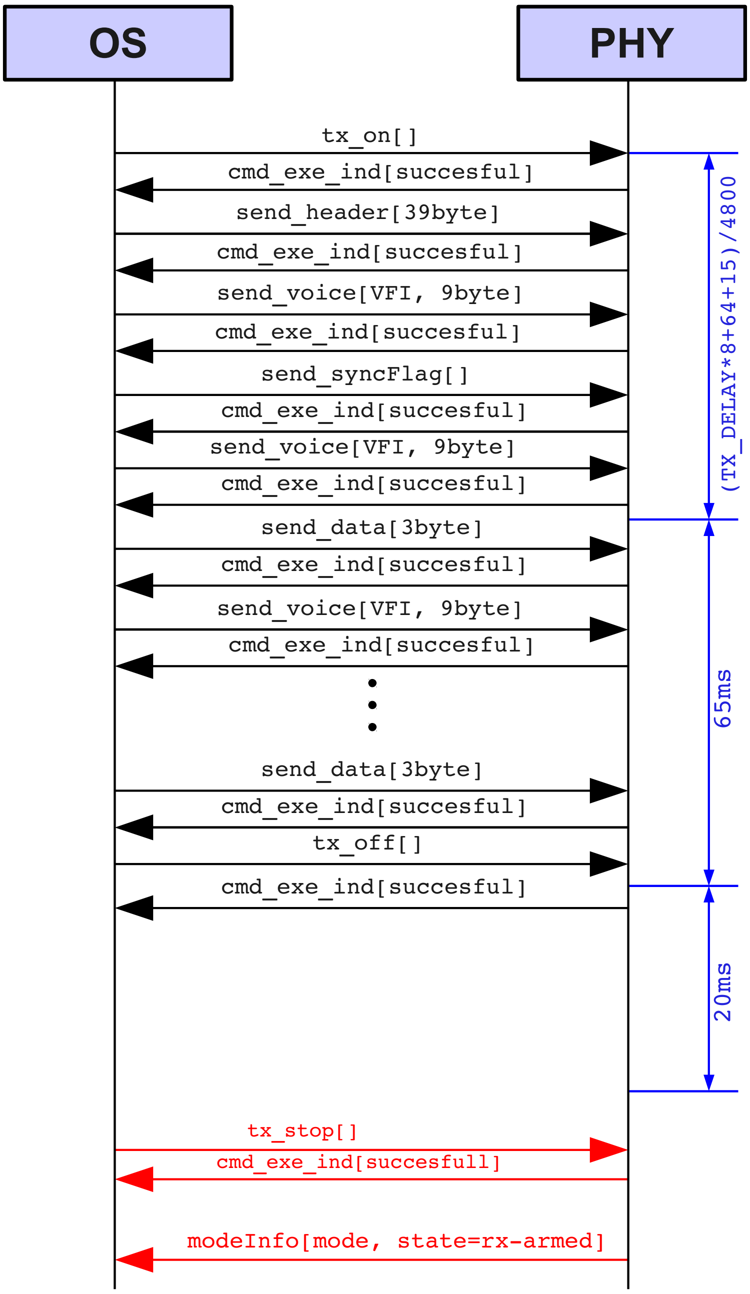

Figure 9: aborted transmitting procedure

3.6.4 Receiving procedure in SUM

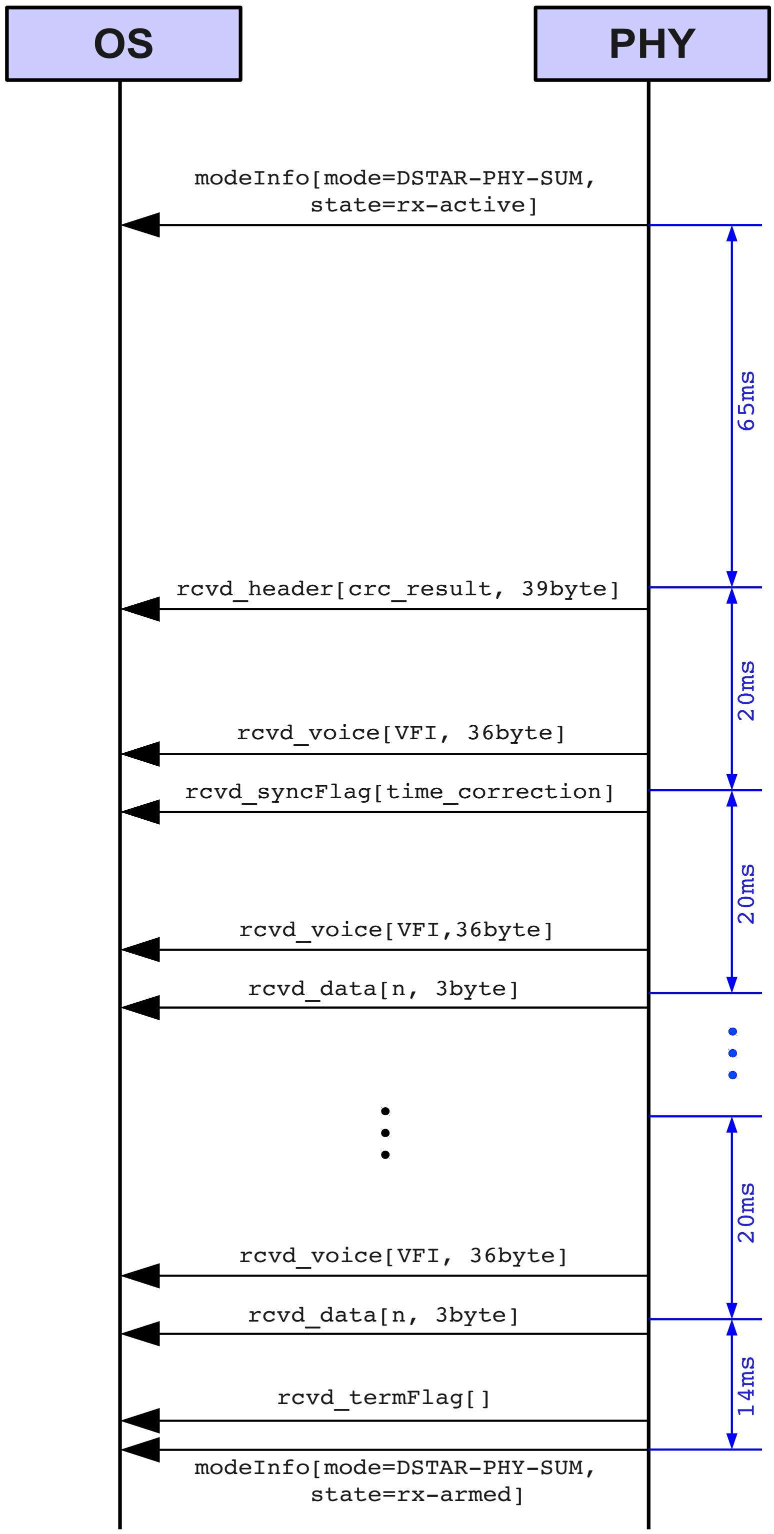

Figure 10:receiving procedure in SUM

3.6.5 Receiving procedure in DM

The PHY-TX instance is only introduced to give an idea how PHY will internally work. This should improve understanding of timings in Digipeater mode.

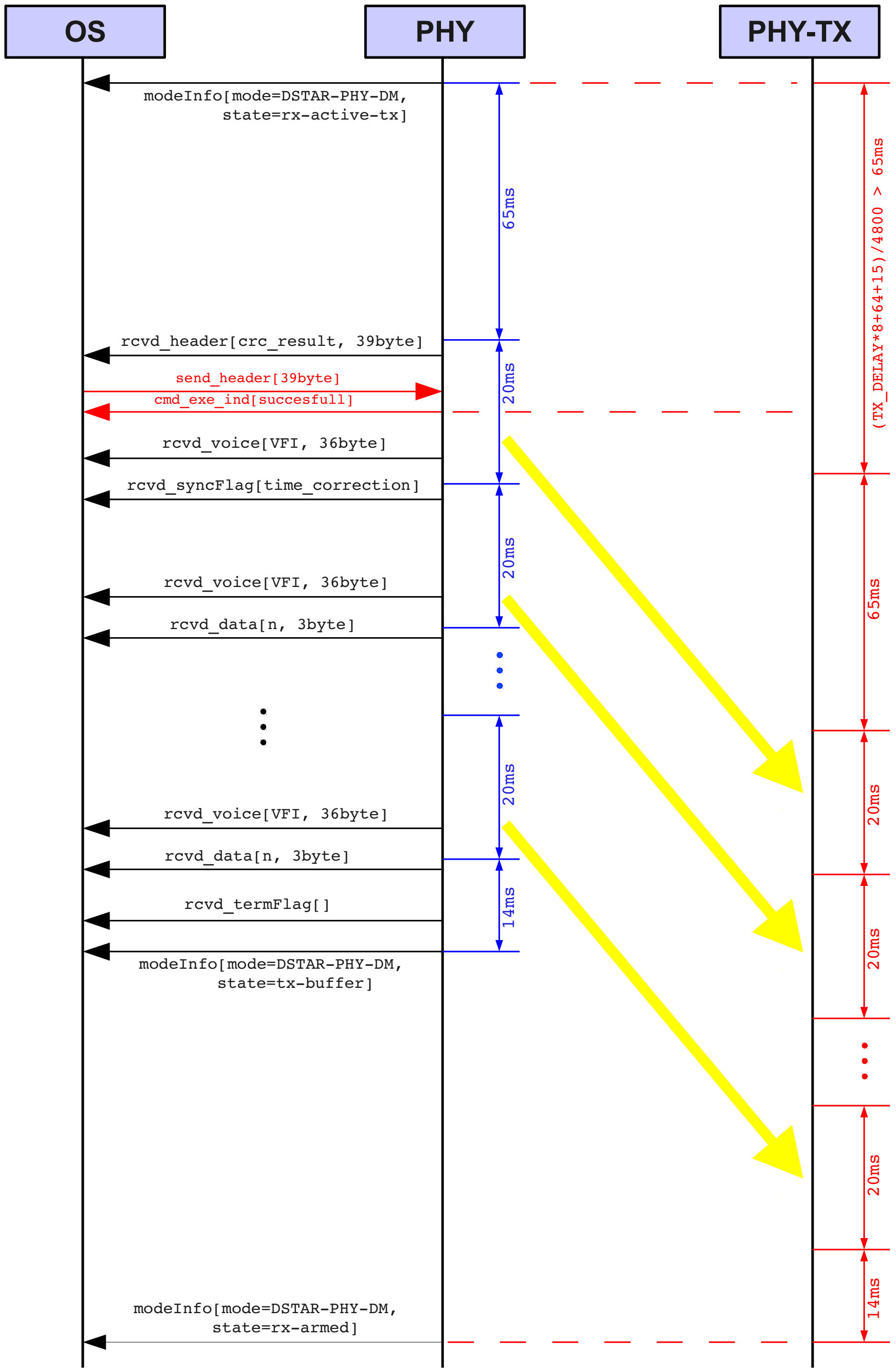

Figure 11: receiving procedure in DM

3.6.5.1 Abnormal behavior

If the OS does not provide the header information for the retransmitted signal at time the PHY finishes the transmission. However the PHY will still provide the received data to the OS while the current received transmission is ongoing.

3.6.6 Procedure for receiving of unwanted signal in DM

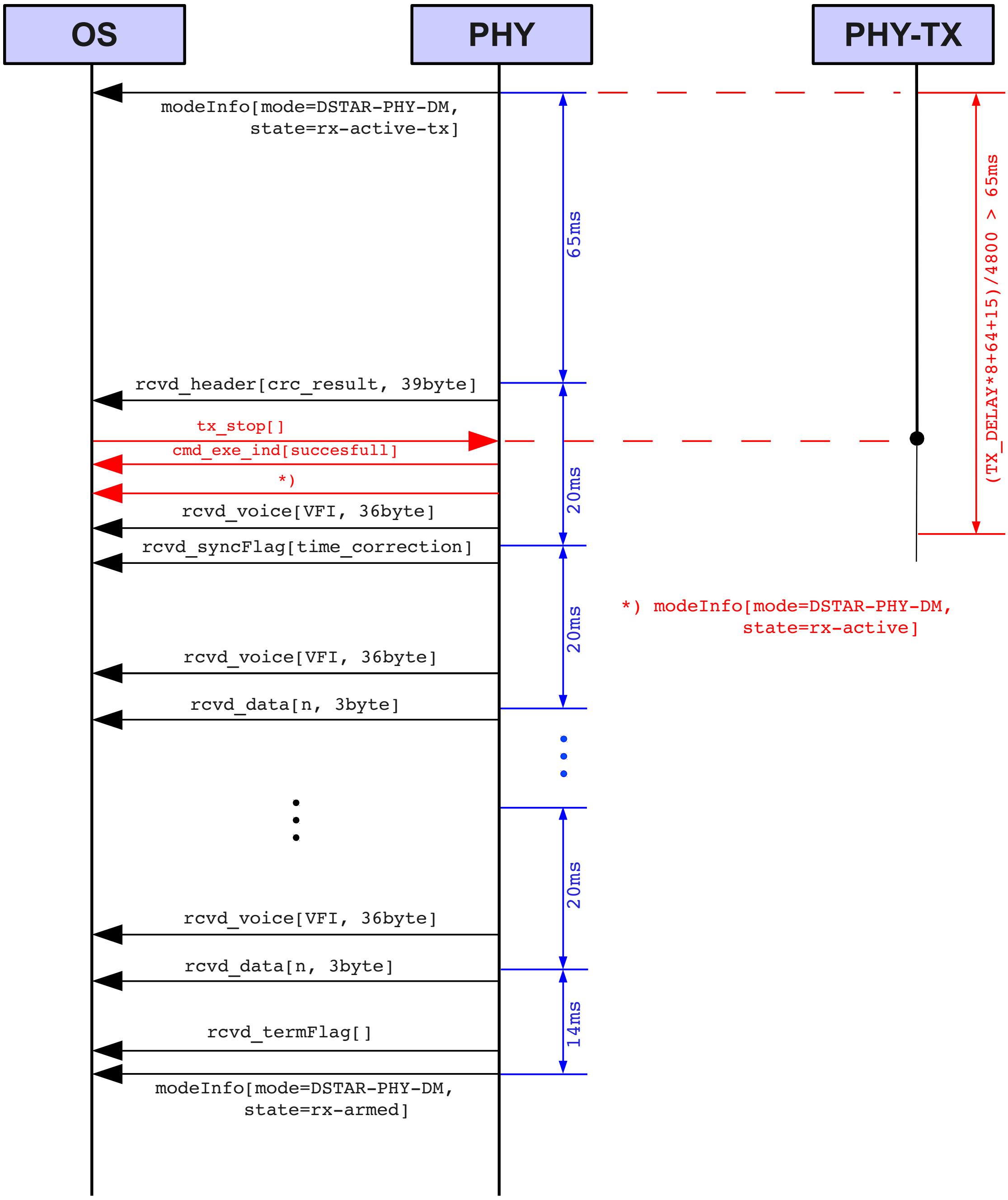

Figure 12: aborting of a unwanted retransmission in DM

With this procedure the OS has a chance to stop the retransmission of unwanted signal. This is the case if the just received header does not fulfil some particular requirements for digipeating.

3.6.7 Procedure for receiving of corrupted signal in DM

If a transmission does not have header information due to some reasons we are speaking about a corrupted transmission. This is the case if the transmitting mobile station was not receivable due to fading or if the receiver of the repeater was just switched on. In such case the PHY goes into **`rx-active-tx`** state by receiving of the next synchronization flag as following figure shows.

In comparison to the receiving procedure in DM (presented in 3.6.5) where the retransmission stops if we do not make any action in time, the received corrupted signal will be retransmitted while it is ongoing.

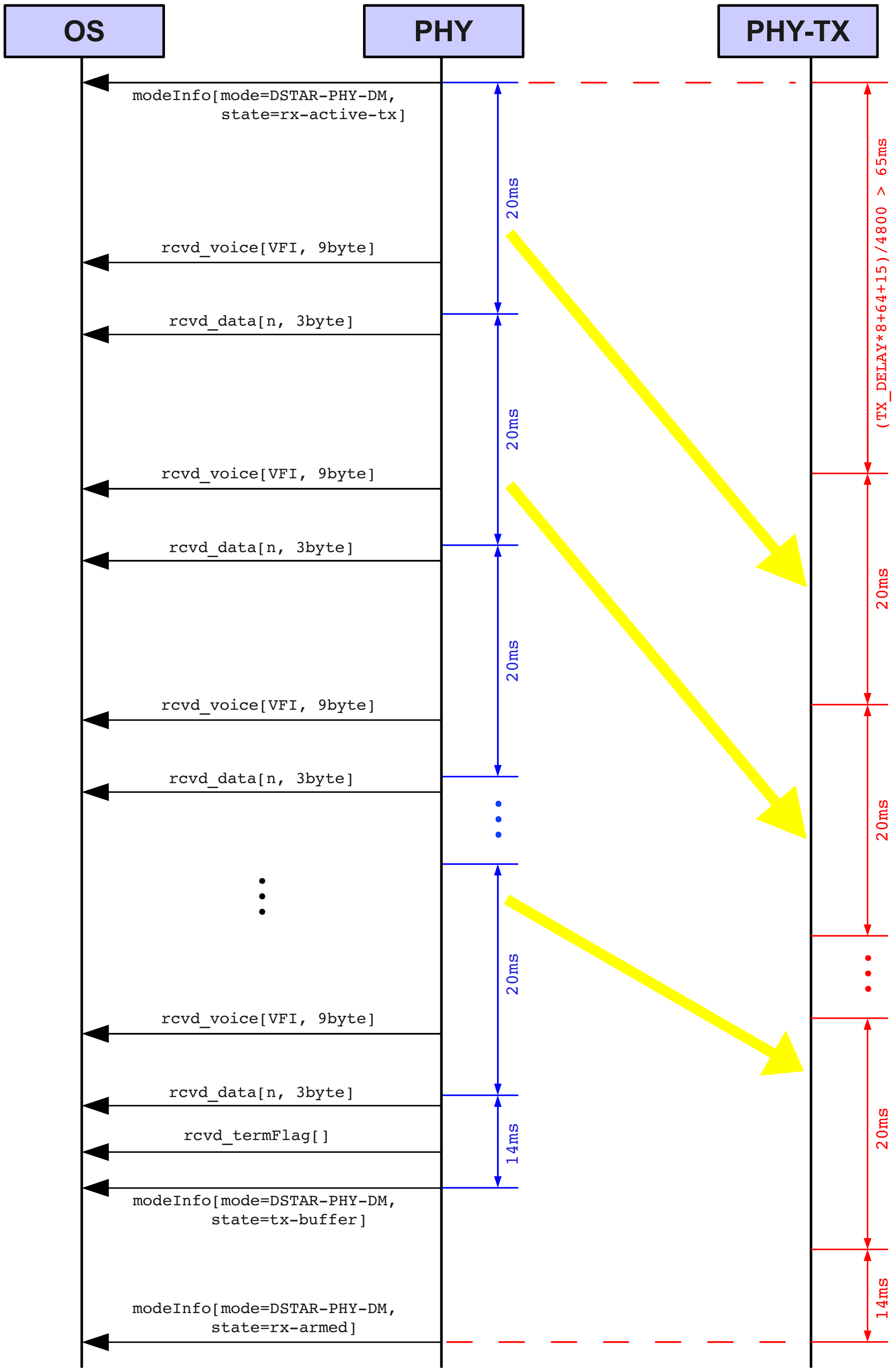

Figure 13: receiving of corrupted signal in DM

3.6.7.1 *Abnormal behavior*

If a retransmission of a corrupted signal is not desired the OS can stop the retransmission during the <u>transmitter power on time</u> (given by **TX-DELAY**). The PHY provides the received data to the OS anyway. The following picture shows this approach.

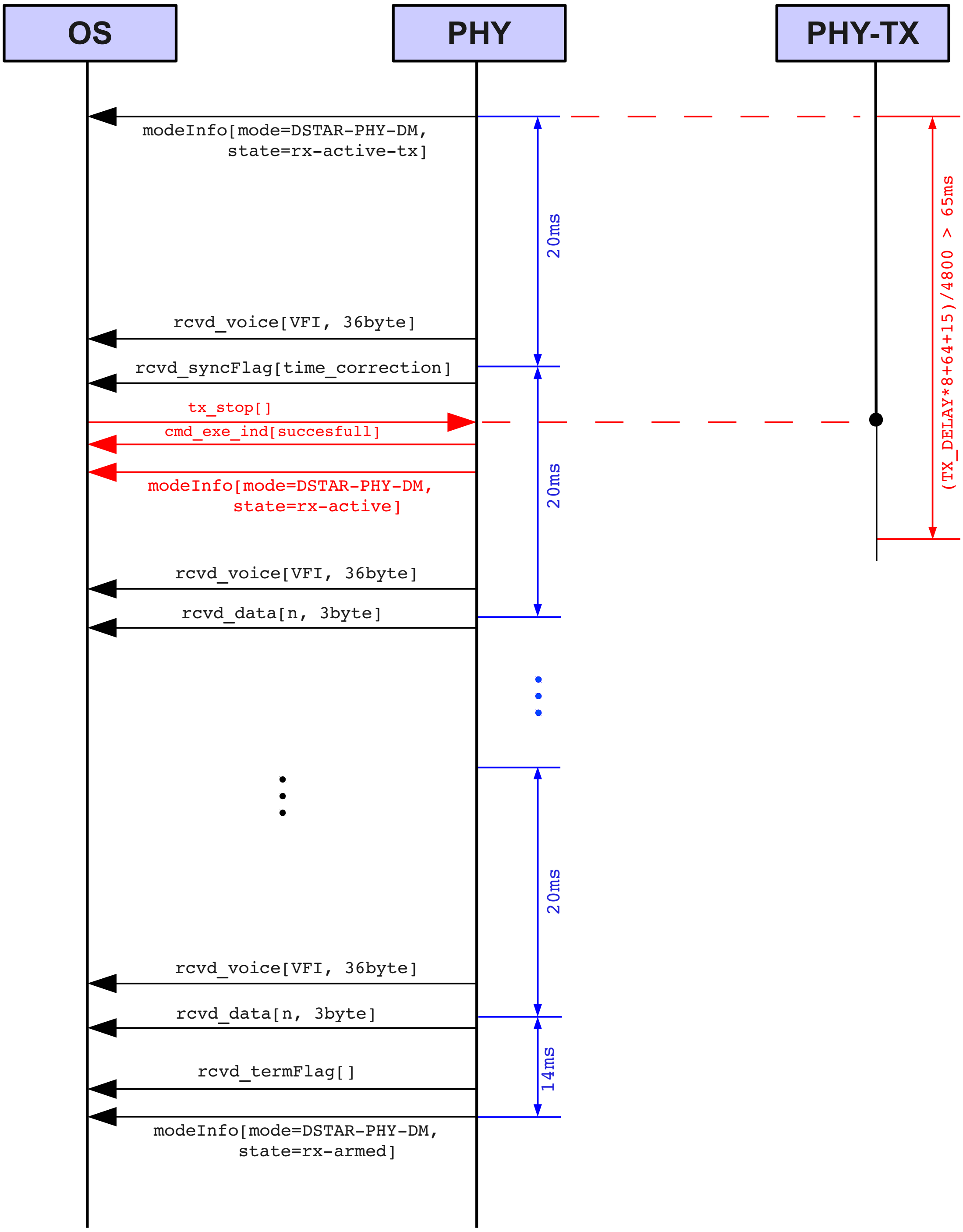

Figure 14: stopping of retransmission of a corrupted signal in DM

3.6.8 MATFST expiry procedure

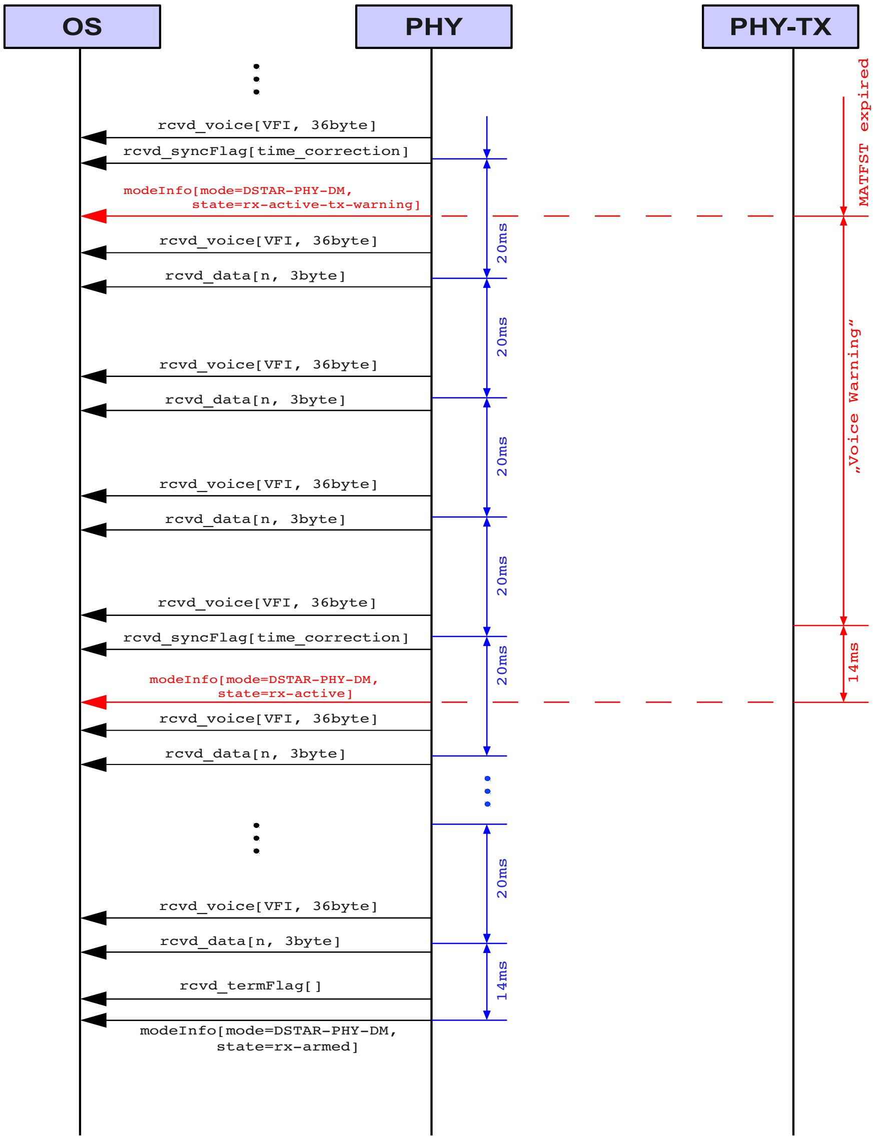

Figure 15: MATFST expiry procedure in DM

If a received transmission is ongoing longer than the MATFST than the PHY stops the retransmission and sends a short voice warning message for indicating of MATFST expiry. The PHY provides the received data to the OS anyway.

3.6.8.1 Abnormal behavior

The following picture shows how the MATFST expiry procedure will be performed if the received transmission stops in the time period of an ongoing voice warning message.

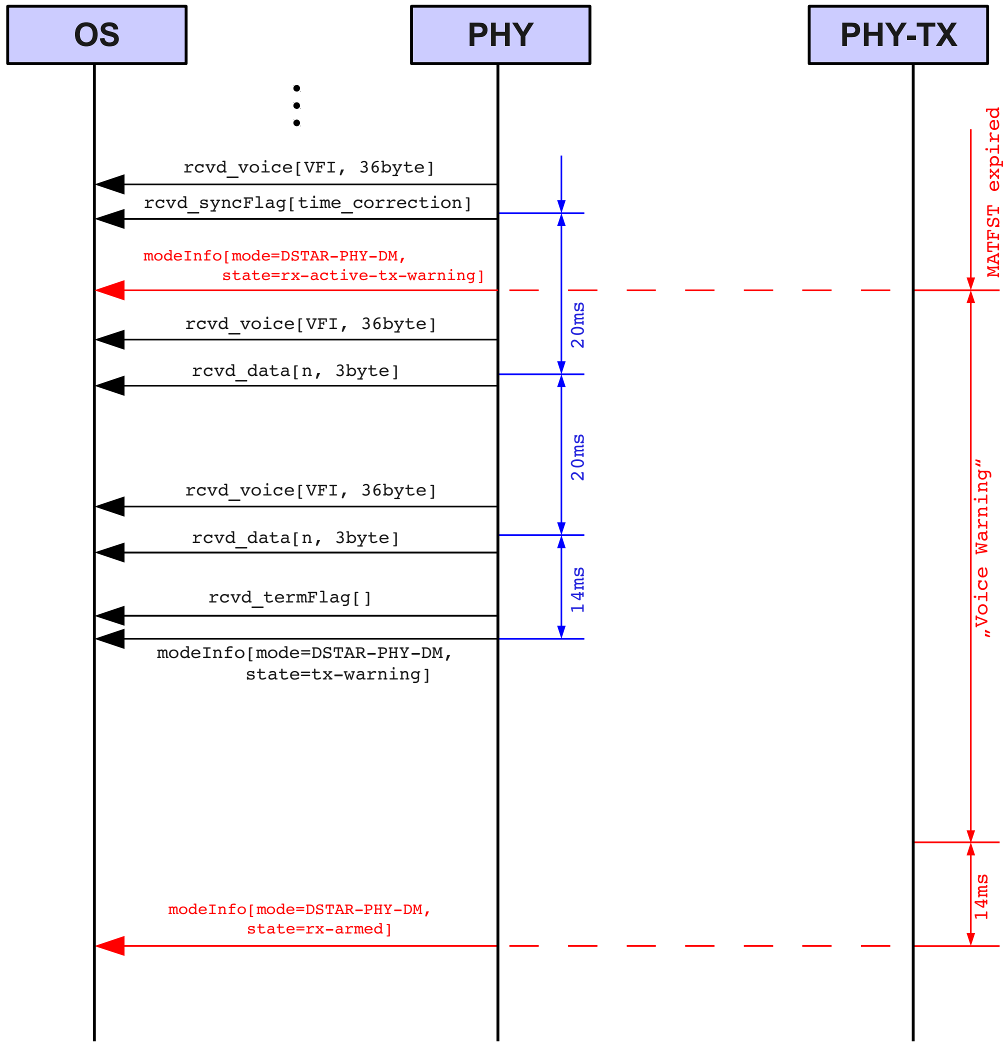

Figure 16: stopped transmission during an ongoing voice warning message

3.6.9 PHY update procedure

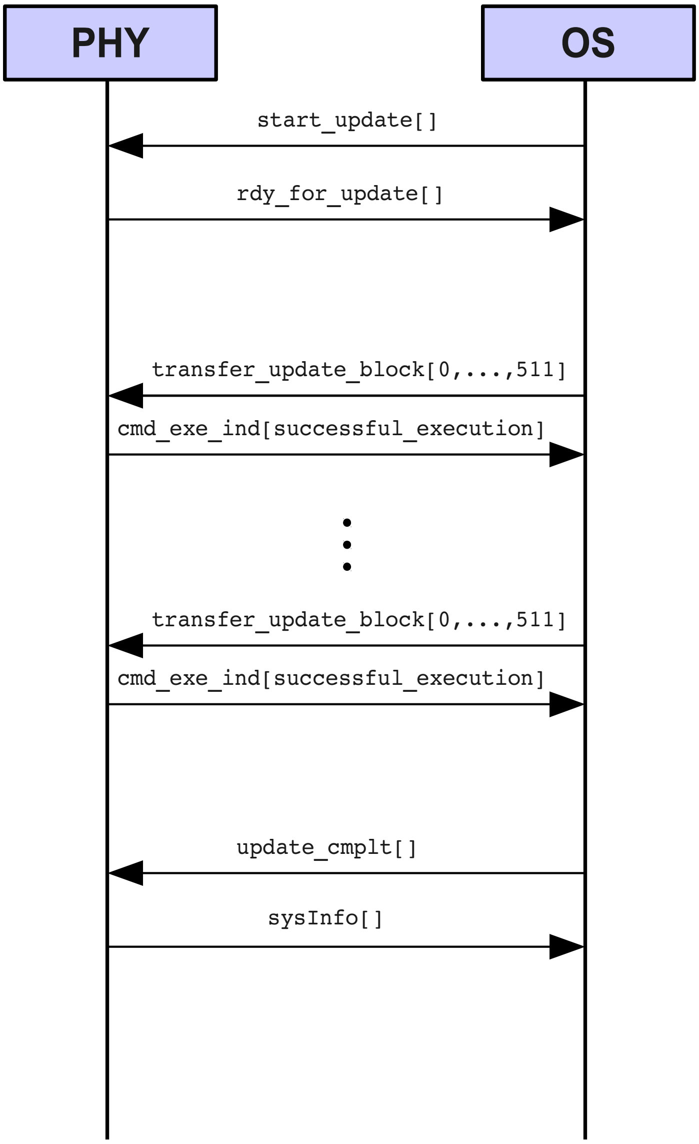

Figure 17: PHY update procedure

4 Annex

4.1 Frame structure of DV packet in D-STAR

A the D-STAR System has two different operation modes:

- digital voice (DV)
- digital data (DD)

In this document the DV mode is considered only. On the following figure the structure of a typical DV radio frame is presented.

1. The preamble of radio frame according [shogen] consists of 64 bits (alternating 1 und 0). Based on the fact, that conventional amateur transmitter needs more time to lock its PLL after the PTT was activated, the preamble is transmitted up to 550bits (and longer), before the “frame sync” will be started.
2. The „frame sync“ is following bit pattern: {1,1,1,0,1,1,0,0,1,0,1,0,0,0,0}.
3. The “sync flag” consists of following bit pattern: {1,0,1,0,1,0,1,0,1,0} + {1,1,0,1,0,0,0} + {1,1,0,1,0,0,0}. The „sync flag“ is transmitted in the 1^{st} and than in each 21^{th} data time slot. The „sync flag“ is used in the receiver for correction of time synchronization to the transmitter as well it is used as preamble in case if the receiver was switched on during an ongoing transmission and missed the regular header.
4. The “terminating flag” is send at the end of the ongoing transmission in order to indicate a regular end of it. The “terminating flag” consists of $\{1,0,1,0,...,1,0,1,0\}_{32\ Bit}$+ $\{0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,0\}_{16Bit}$. After that approximately 20 Zeros or Ones are send. The Transmitter is switched of (PTT deactivation) after 10^{th} such Zero or One.

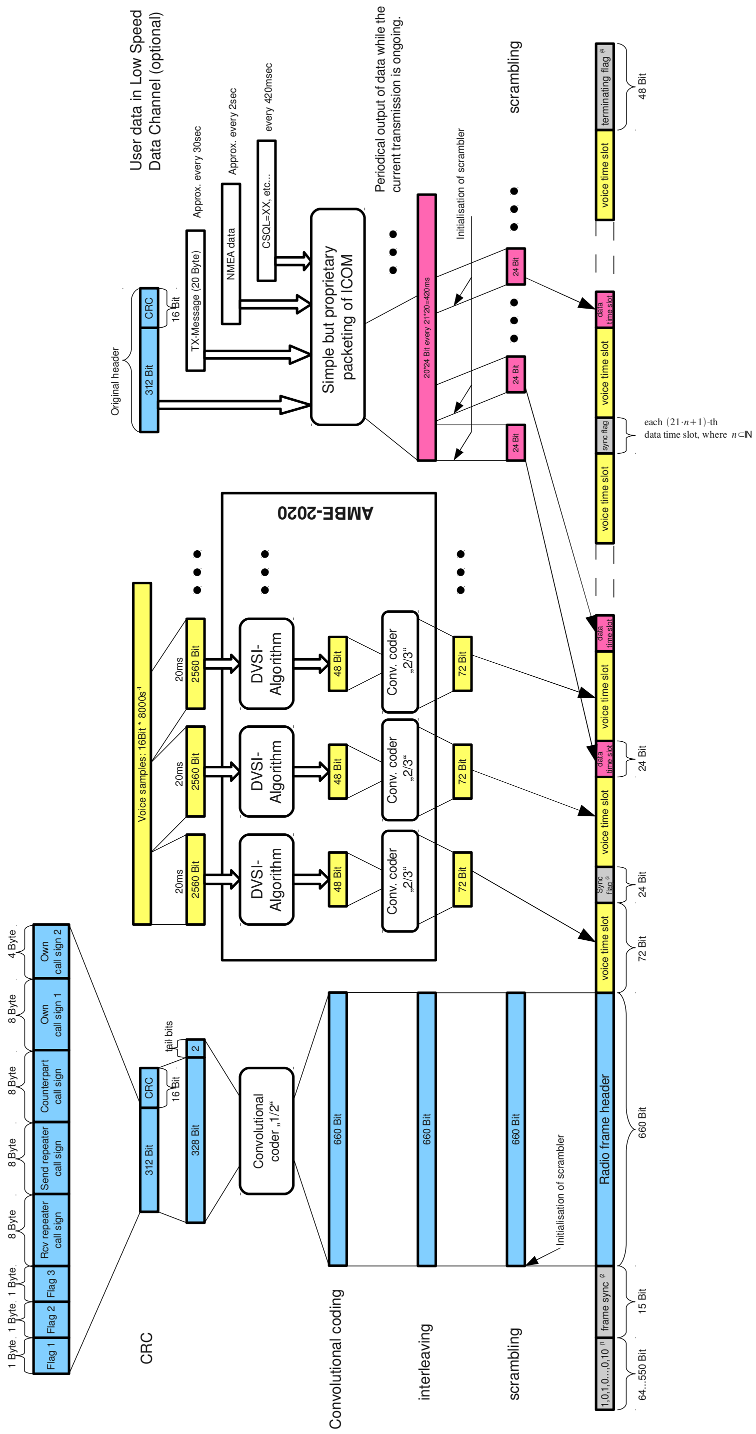

Figure 18: D-STAR radio frame structure in DV mode

4.2 Description of D-STAR application layer

As already presented a DV radio frames transmits together with digital voice data also user data in so called **low speed data channel (LSDC)**. In this section the bit mapping and organization of data in LSDC of current commercial available D-STAR capable amateur radio transceiver should be discussed. Based on this information a outlook for potential future extensions can be given.

4.2.1 Mapping of LSDC

Each 21th data time slot is used for synchronization issues. The remaining subsequent received 20 data slots in between (3 bytes each of them) builds 10 subsequent data containers. Each data container has therefore exactly 6 bytes. The 1st byte indicates how the remaining 5 bytes of this container are used. Thereby the 1st hex digit means the container's type and the 2nd hex digit indicates how many following bytes of remaining 5 are used. The unused bytes are set to `0x66`. The following table shows the known container's type.

Container's type	Number of used bytes	Description
`0x40`		1st 5 bytes of the TX-Message
`0x41`		2nd 5 bytes of the TX-Message
`0x42`		3rd 5 bytes of the TX-Message
`0x43`		5th 5 bytes of the TX-Message
3	N=1,..,5	This container type is used for transmission of GPS data
5	N=1,..,5	This container type is used for transmission of the original header[1]. This container can be transmitted every time if the LSDC does not have any other data to transmit.
6	6	This is a data block “NOP”. Usually all 6 bytes are `0x66` if nothing is available for transmission.
C	2	This container (if used) must be sent always in the 1st position. This means immediately after synchronization flag and use only 2 bytes in which the digital squelch code (CSQL) is repeated twice. The remaining unused 3 bytes contains 0x66. **NOTE**: The valid CSQL range is $(1..99)_{dec}$. If the desired CSQL=19_{dec}, than the both byte have to be $0x19_{hex}$!

[1]Basically the header transmitted in LSDC should be identical with the information in the DV radio frame header. But if the DV radio frame is retransmitted via a digipeater, the digipeater will generally change the header during the retransmission. All other repetitions of header in the LSDC remains of course unchanged. So we have a chance to “see” what the transmitter originally has set in his header.

4.2.2 Empty voice data channel

If the OS does not have any voice data to transmit (e.g. some periodical GPS transmissions), it fills the voice data time slots with

"Voice-NOPs" = "**`9e 8d 32 88 26 1a 3f 61 e8`**".

4.2.2.1 Example

```
SyncFlag :
data     :  010000111001100010011000  : c2 19 19 : ...
data     :  011001100110011001100110  : 66 66 66 : fff
data     :  101011000010010011100010  : 35 24 47 : 5$G
data     :  000010101110001011100010  : 50 47 47 : PGG
data     :  000000100010001000110010  : 40 44 4c : @DL
data     :  110011001111001011000010  : 33 4f 43 : 3OC
data     :  101011001000001000110100  : 35 41 2c : 5A,
data     :  100011001000110010101100  : 31 31 35 : 115
data     :  100000101101001000000100  : 41 4b 20 : AK
data     :  001000101010001001110010  : 44 45 4e : DEN
data     :  101011000000110011001100  : 35 30 33 : 503
data     :  100111000111010000001100  : 39 2e 30 : 9.0
data     :  010000101001001011001010  : 42 49 53 : BIS
data     :  000001000001001010001100  : 20 48 31 :  H1
data     :  101011000100110000110100  : 35 32 2c : 52,
data     :  101011000100110011001100  : 35 32 33 : 523
data     :  110000101100110000000100  : 43 33 20 : C3
data     :  000001000000010000000100  : 20 20 20 :
data     :  101011000000110001110100  : 35 30 2e : 50.
data     :  100011001100110001101100  : 31 33 36 : 136

SyncFlag :
data     :  010000111001100010011000  : c2 19 19 : ...
data     :  011001100110011001100110  : 66 66 66 : fff
data     :  101011001110110000110100  : 35 37 2c : 57,
data     :  011100100011010000001100  : 4e 2c 30 : N,0
data     :  101011001000110011001100  : 35 31 33 : 513
data     :  100011001001110001110100  : 31 39 2e : 19.
data     :  101011001001110000011100  : 35 39 38 : 598
data     :  000111001010110000110100  : 38 35 2c : 85,
data     :  101011001010001000110100  : 35 45 2c : 5E,
data     :  100011000011010000001100  : 31 2c 30 : 1,0
data     :  101011001010110000110100  : 35 35 2c : 55,
data     :  110011000111010000001100  : 33 2e 30 : 3.0
```

```
data     :  101011000011010001101100  : 35 2c 36 : 5,6
data     :  100011000111010011001100  : 31 2e 33 : 1.3
data     :  101011000011010010110010  : 35 2c 4d : 5,M
data     :  001101000010110010001100  : 2c 34 31 : ,41
data     :  101011000111010010001100  : 35 2e 31 : 5.1
data     :  001101001011001000110100  : 2c 4d 2c : ,M,
data     :  101011000011010001010100  : 35 2c 2a : 5,*
data     :  101011000110110010110000  : 35 36 0d : 56.
```

In the presented trace the following information can be identified:

CSQL = **19**

TX-Message = "**DL3OCK DENIS H13** "

$GPGGA,115039.02,5230.1367,N,01319.9885,E,1,05,3.0,61.3,M,41.1,M,,*56<CR>

4.2.2.2 Transmission of GPS data

The current commercial available D-STAR equipment transmits in GPS mode (NOT GPS-A) following data using container type "**3**" (see 4.2.1):

$GPGGA,210530.03,5230.1381,N,01319.9732,E,1,05,4.2,53.7,M,41.1,M,,*56<CR><LF>

$GPRMC,210531.03,A,5230.1381,N,01319.9731,E,0.00,173.7,141108,1.9,E,A*0C<CR><LF>

DL3OCK ,BN DENIS*09 <CR><LF>

$GPGGA,211022.03,5230.1380,N,01319.9710,E,1,05,4.1,52.0,M,41.1,M,,*55<CR><LF>

$GPRMC,211023.03,A,5230.1380,N,01319.9711,E,0.00,27.7,141108,1.9,E,A*38<CR><LF>

DL3OCK ,BN DENIS*09 <CR><LF>

In the GPS mode the user equipment seems to retransmit the original NMEA messages coming directly form a GPS device (in the case above only GGA and RMC NMEA messages were chosen in the GPS pre configuration menu). A standard NMEA GPS device protects it statements by a kind of check sum, which is simply XOR addition of all bytes between "**$**" and "*". In addition to the NMEA messages a current commercial available D-STAR device transmits an additional message, which is needed in order to allows of the software running on the current commercial available digipeater to forward such positioning reports to the world wide amateur radio APRS network.

The main part of this datagram is the so called C1-message. It consists of maximal 13 left aligned characters (ASCII-bytes) plus "*" and 2 bytes of "XOR check sum" of all message bytes left of "*". The following picture give the overview about the structure of the whole information message. If the C1-message plus "*" and 2 bytes of "XOR check sum" is shorter than 16 bytes, than it will be padded right by spaces (see the example above).

call sign	0x2C	symbol	0x20	C1-message + checkSum	0x0D	0x0A
8bytes	,	3bytes	„space“	16bytes	<CR>	<LF>

Figure 19: structure of the information message in GPS mode

The call sign is limited to 8 characters. If it is shorter it will be left aligned and padded right by spaces. The call sign may contain only alphanumerical characters and “**-**”. Here are examples of valid combinations:

“**DL3OCK **“

“**DL3OCK-9**”

“**DL1AA-9 **“

“**DL1A   **“

“**DL1A-AM **“

“**DL1A-99 **“

The current commercial available D-STAR radio equipment is not able to calculate the check sum within GPS information message. No idea why! Therefore users, each time they want to use this GPS mode, are enforced to use a web based tool for this:

http://www.aprs-is.net/DPRSCalc.aspx

This tool computes the check sum and creates only the coloured part of the whole message, which should be put to a GPS configuration menu. Of course an OS software developer for UP4DAR will integrate this easy computation in the program and compute the checksum immediately after the user have changed the C1 message. In my opinion there is nothing against a flexible usage of two different call signs. For example for the header of the whole DV radio frame MY_CALLSIGN = “**DL3OCK M**“ and for the GPS information message inside the same radio frame “**DL3OCK-9**”.

The other available GPS mode is so called GPS-A mode. The advantage of the GPS mode described above in comparison to the GPS-A is the fact that the current commercial available D-STAR equipment is able to visualize the position of the received station on its display. However a GPS-A message (which is APRS very likely) has a more efficient error check method which is a good known CRC and not a simple check sum:

```
$$CRC8082,DL3OCK>API282,DSTAR*:/211248h5230.13N/01319.98E-027/000/Denis zu Hause<CR>
```

4.2.2.3 Hints for an advanced usage of LSDC

Currently used data transmission in LSDC does not have any error protection. In order to protect the transmission the UP4DAR interface (tanks to the number indication of received data block) allows to design some effective data transmission frames for more efficient and robust data transmission in LSDC.

The following figure shows a proposal for a possible advanced usage. In this figure all 10 subsequent containers in between of two sync flag are presented. It is proposed to use first 3 containers on the same way, the currently commercial available equipment do (see 4.2.1) in order to have backward compatibility.

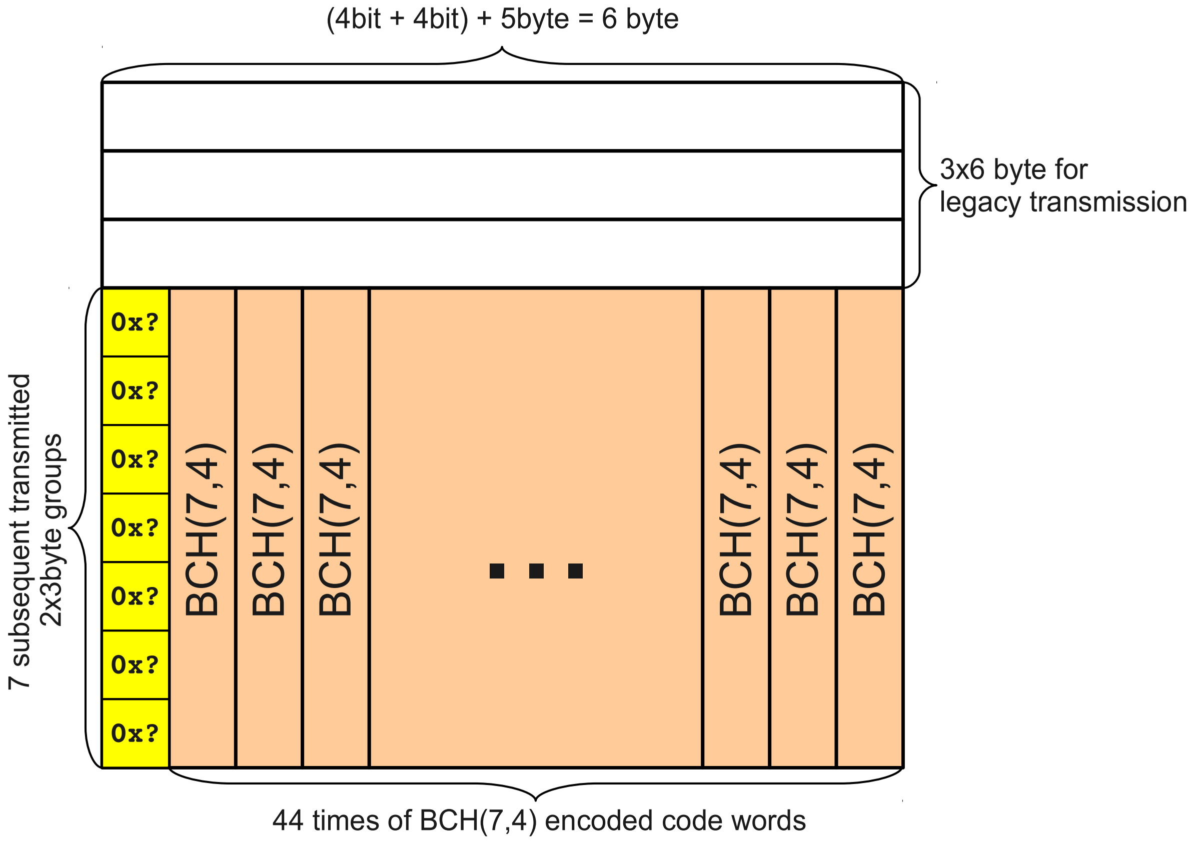

Figure 20: advanced usage of LSDC

The remaining 7 containers should build the new advanced data frame. For this we have to find out in further investigations, which container type is not used in currently commercial available equipment, in order to not alloy theirs normal operation. Potentially it could be each hex number, which is currently missing in 4.2.1. This container type will transmitted in each place of the yellow column. The remaining coloured bits are encoded with a linear cycle block code BCH(7,4). This means that each code word of the length 7 carries 4 information bits. Note, it is important to build the code words on the shown way in the vertical and not in the horizontal direction! Therefore our new advanced data frame has 22 high protected information bytes. These bytes can be used as follow. The 1st byte is CRC of the remaining 21 bytes. The 2nd byte is so called transport format indicator (TFI) and of course 20 information bytes. The advantage of such advanced forward error protection (FEC) is huge. For this special proposal it means, that during the transmission time of **280ms** for an advanced data frame a continuous part of **40ms** allows to be corrupted in order that all information bits can be detected without any error!

The following table give an idea for definition of the TFI:

TFI	Description
`0x01`	20 byte of TX-Message
`0x02`	1st part of original header
`0x03`	2nd part of original header
`0x04`	1st part of position fix (APRS call sign, message and position fix compressed according to [APRS] or [3GPP])
`0x05`	2nd part of position fix (APRS call sign, message and position fix compressed according to [APRS] or [3GPP])
`0x06`	General text message to be displayed on receiver. This is mainly foreseen for transmitting of short summary during the weekly amateur radio broadcasting (Rundspruch).

Figure 21: proposal for transport format indicator (TFI)

4.3 Proposals for desktop software

4.3.1 GUI

t.b.d.

4.4 Proposal for a low delay inter digipeater transmission

It is basically no matter how the connection and its protocols are designed for a inter control signalling between two digipeater (updates of the routing information for instance). But for a critical voice data of an ongoing transmission the link between two digipeaters should be as low delay as possible.

In order to achieve a low delay connection it is proposed to use UDP (or UDP lite) protocol which is primary designed for such issues. For avoiding of requests of missing data or UDP packets coming to late the following usage of the UDP packets is proposed. See the next figure:

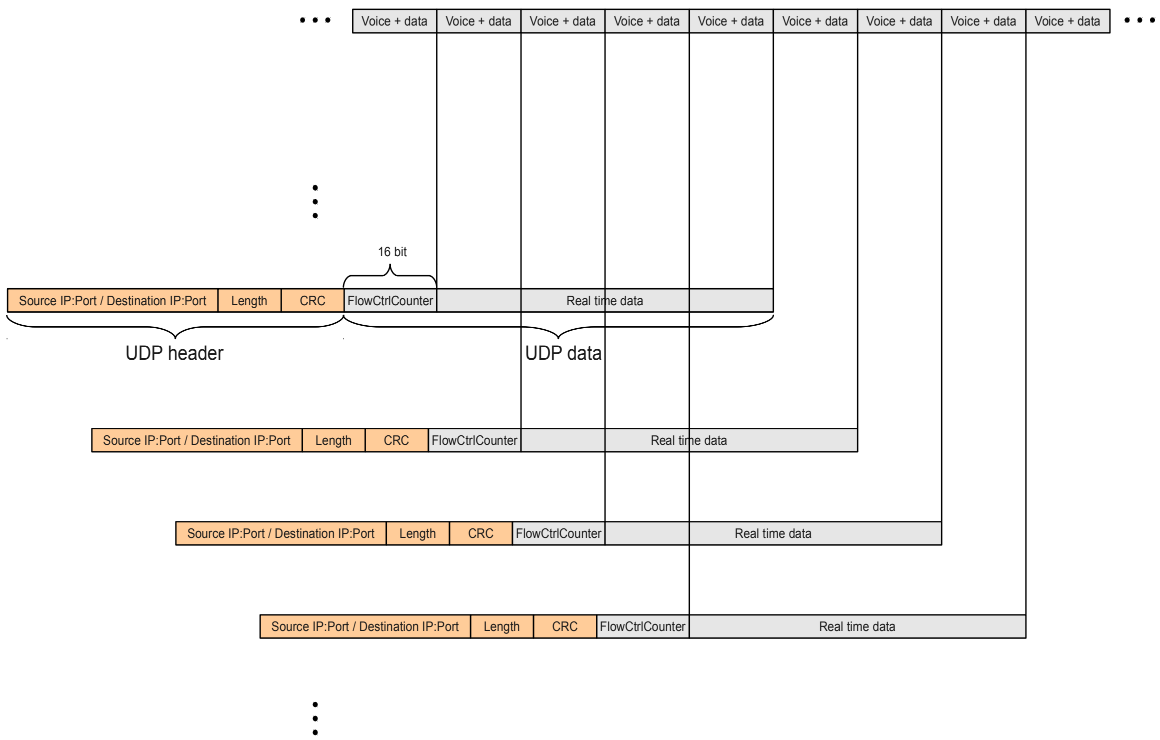

Figure 22: proposal for a low delay inter digipeater transmission