RC Electronics
communication protocol

Revision 1.0

January 2022

Contents

Revision history

Version	Date	Change log
1.0	07.01.2022	- initial release

Abstract

In this document we are describing a set of commands needed to communicate and transfer data between external device and RC Electronics device.

Communication is using NMEA0813 serial protocol. Speed of communication (baud rate) depends on setting in RC Electronics device but is normally set to BR38400 8N1.

Communication can be established over Data 1 port on device or over integrate / external BT 2.0 module.

Document status

This document is public and can be used by anyone to use, distribute and implement.

Data sent out from the unit:

Data being sent out, depends on unit setting (if enabled or not). By default, each unit will send out:

GPS data: GPGGA, GPRMC, GPRMB

Flarm data (if Flarm is connected to the device): PFLAU, PFLAA

LXWPx data: LXWP0, LXWP1, LXWP2, LXWP3

GPS data

Specification of GPGGA, GPRMC and GPRMB are well defined on web.

Flarm data

Flarm data received by Flarm is just forwarded out, so please check specification of it at www.flarm.com web page.

LXWPx

LXWP0

Contains flight parameters.

$LXWP0,*logger_status,tas,altitude,vario1,vario2,vario3,vario4,vario5,vario6,heading,wind_direction,wind_speed*CS

PARAMETER	TYPE	DESCRIPTION
logger_status	char	'Y' logger is recording, 'N' logger is not recording
tas	float	True airspeed in km/h
altitude	float	Altitude in meters
vario1,2,3,4,5,6	float	Vario measurements in last second (6 measurements)
heading	int16_t	True heading. -1 if compass not connected
wind_direction	string	Wind direction in degrees, blank if wind is 0
wind_speed	string	Wind speed in km/h, blank if wind is 0

LXWP1

Contains basic device information.

$LXWP1,*device_type,serial_nr,fw_version,hw_version*CS

PARAMETER	TYPE	DESCRIPTION
device_type	string	Device type (example: Fenix)
serial_nr	uint32_t	Serial number
fw_version	float	Firmware version
hw_version	float	Hardware version

LXWP2

Contains polar data for calculation of speed command and volume of vario

$LXWP2,*mc,load,bugs,polarA,polarB,polarC,volume*CS

PARAMETER	TYPE	DESCRIPTION
mc	float	MacCready setting
load	float	Load coefficient (total flight mass divided by polar reference mass)
bugs	uint16_t	Bugs factor in %
polarA,B,C	float	Polar coefficients Av2+Bv+C
volume	uint8_t	Volume of vario tone

LXWP3

Contains settings parameters

$LXWP3,*alt_offset,sc_mode,filter,reserved,te_level,int_time,range,sc_silence,sc_switch_mode,sc_speed,polar_name*CS

PARAMETER	TYPE	DESCRIPTION
alt_offset	int16_t	Daily offset of altitude due to daily QNH for flying QFE
sc_mode	uint8_t	0 = manual, 1 = circling, 2 = speed
filter	float	Vario filter in seconds
reserved		
te_level	uint16_t	TE level in %
int_time	uint16_t	Vario integration time in seconds
range	float	Vario scale range in m/s
sc_silence	float	SC silence band in m/s
sc_switch_mode	uint8_t	0 = off, 1 = on, 2 = toggle
sc_speed	uint16_t	SC speed. Works when sc mode is set to 2 (speed)
polar_name	string	Name of polar

Data parsed by unit:

PFLXx

PFLX0

Unit will parse out the requested refresh rate for LXWP0, LXWP1, LXWP2 and LXWP3 sentences

$PFLX0,*data_type_1,interval_data_type_1, data_type_2,interval_data_type_2,…*CS

PARAMETER	TYPE	DESCRIPTION
data_type_x	type	LXWPx
interval_data_type_x	int8_t	Interval at which data type is sent out. If set to 0 then data type output is disabled. If set to -1 then data type is requested only once.

PFLX2

Parsing MC, ballast, bugs and volume control

$PFLX2,*mc,load,bugs,polarA,polarB,polarC,volume*CS

PARAMETER	TYPE	DESCRIPTION
mc	float	MacCready setting
load	float	Load coefficient (total flight mass divided by reference mass)
bugs	uint16_t	Bugs factor in %
polarA,B,C	float	Polar coefficients Av2+Bv+C **- ignored!**
volume	uint8_t	Volume of vario tone in %

RCDT - data / declaration transfer

A set of data transfers can be made between remote device and RC Electronics device using RCDT type of sentence.

Sentence supports SET and GET action and will reply with:

- **ANS,TYPE,parameters… to a GET action**
- **ANS,OK to a SET action**

$RCDT,*action,type,parameter_1,parameter_2,...parameter_nCS**

PARAMETER	TYPE	DESCRIPTION
action	string	GET – unit will send out requested type of parameters SET – unit will set new parameters ANS – units reply with it to a SET/GET action
type	string	INFO – Get unit info TP – SET/GET task turn-point ZONE – SET/GET task turn-point observer zone GLIDER – SET/GET glider data PILOT – SET/GET pilot data TSK_PAR – SET/GET task parameters MC_BAL – SET/GET MacCready, ballast, bugs parameters SC_VAR- SET/GET SC/vario mode of operation NAVIGATE – SET/GET point of navigation SENS – GET sensor values
parameter_x		Check sentence type specification bellow

INFO

GET request: $RCDT,GET,INFO*CS

Response:
$RCDT,ANS,INFO*,device_type,serial_nr,fw_version,hw_vresion,reserved,reserved,reserved, reservedCS**

PARAMETER	TYPE	DESCRIPTION
device_type	string	Device type (example: Fenix)
serial_nr	uint32_t	Serial number
fw_version	float	Firmware version
hw_version	float	Hardware version
reserved		Reserved for future

TP

GET request: $RCDT,GET,TP,*tp_idCS**

PARAMETER	TYPE	DESCRIPTION
tp_id	uint8_t	Task turn-point id. 0 = take-off point

Response: $RCDT,ANS,TP,*tp_id,type,lat,lon,nameCS**

SET request: $RCDT,SET,TP,*tp_id,type,lat,lon,nameCS**

PARAMETER	TYPE	DESCRIPTION
tp_id	uint8_t	Task turn-point id
type	uint8_t	Turn-point type. 1 = point, 2 = landing, 3 = take-off
lat	int32_t	Latitude in thousands of minutes (60000 = 1° 0.0')
lon	int32_t	Longitude in thousands of minutes (60000 = 1° 0.0')
name	string	Name of turn-point

ZONE

GET request: $RCDT,GET,ZONE,*zone_idCS**

PARAMETER	TYPE	DESCRIPTION
zone_id	uint8_t	Task turn-point zone id. 0 = start point

Response: $RCDT,ANS,ZONE,*zone_id,direction,auto_next,line,a1,a2,a21,r1,r2,elevationCS**

SET request: $RCDT,SET,ZONE,*zone_id,direction,auto_next,line,a1,a2,a21,r1,r2,elevationCS**

PARAMETER	TYPE	DESCRIPTION
zone_id	uint8_t	Task turn-point zone id

direction	uint8_t	Zone direction
		`//! Zone direction type`
		`enum Direction {`
		`  dSymmetric = 0,  //!< Sym relative to connecting lines.`
		`  dFixed,  //!< Defined by extra (A21) parameter.`
		`  dNext,  //!< Defined by next point.`
		`  dPrevious,  //!< Defined by previous point.`
		`  dStart  //!< Defined by the start point.`
		`};`

auto_next	boolean	True (1) if autonext is enabled
line	boolean	True (1) if sector is line (start / finish)
a1	uint16_t	Angle A1 in degrees
a2	uint16_t	Angle A2 in degrees
a21	uint16_t	Angle A21 in degrees
r1	uint16_t	Radius R1 in meters
r2	uint16_t	Radius R2 in meters
elevation	uint16_t	Zone elevation in meters. Needed for finish FG

GLIDER

GET request: $RCDT,GET,GLIDER*CS

Response: $RCDT,ANS,GLIDER,*name,reg_nr,comp_id,classCS**

SET request: $RCDT,SET,GLIDER,*name,reg_nr,comp_id,classCS**

PARAMETER	TYPE	DESCRIPTION
name	string	Glider name
reg_nr	string	Registration number
comp_id	string	Competition ID
class	string	Competition class

PILOT

GET request: $RCDT,GET,PILOT*CS

Response: $RCDT,ANS,PILOT,*name,surnameCS**

SET request: $RCDT,SET,PILOT,*name,surname* *CS

PARAMETER	TYPE	DESCRIPTION
name	string	Pilot name
surname	string	Pilot surname

TASK_PAR

GET request: $RCDT,GET,TASK_PAR*CS

Response: $RCDT,ANS,TASK_PAR,*finish_1000,finish_alt_offset,aat_timeCS**

SET request: $RCDT,SET,TASK_PAR,*finish_1000,finish_alt_offset,aat_timeCS**

PARAMETER	TYPE	DESCRIPTION
finish_1000	boolean	True (1) if finish 100m bellow starting point is enabled
finish_alt_offset	uint16_t	Altitude offset in meters between turn-point elevation and finish altitude.
aat_time	string	HH:MM time format for task AAT time

MC_BAL

If any of parameter values are empty (,,) then this will be ignored by the device.

GET request: $RCDT,GET,MC_BAL*CS

Response: $RCDT,ANS,MC_BAL,*mc,ballast,bugs,brightness,vario_vol,sc_vol,qnh*CS**

SET request: $RCDT,SET,MC_BAL,*mc,ballast,bugs,brightness,vario_vol,sc_vol,qnh*CS**

PARAMETER	TYPE	DESCRIPTION
mc	float	MacCready setting
ballast	uint16_t	Ballast in kg
bugs	uint8_t	Bugs factor in %
brightness	uint8_t	Brightness in %
vario_vol	uint8_t	Volume of vario tone in %
sc_vol	uint8_t	Volume of SC tone in %
qnh	uint16_t	QNH in mBar (hPa)

SC_VAR

Remote device can toggle SC mode of RC Electronics device. In order to have this working, user must set in RC Electronics device following setting for SC mode = manual and SC switch = toggle

GET request: $RCDT,GET,SC_VAR*CS

Response: $RCDT,ANS,SC_VAR,*state*CS**

SET request: $RCDT,SET,SC_VAR,*state*CS**

IMPORTANT: If RC Electronics device is already in requested state it responds with $RCDT,ANS,OK*CS, if a change of state occurs it responds with new state, for example: $RCDT,ANS,SC_VAR,1*CS

PARAMETER	TYPE	DESCRIPTION
state	uint8_t	0 = vario, 1 = SC, 2 = toggle current state

NAVIGATE

GET request: $RCDT,GET,NAVIGATE,*typeCS**

PARAMETER	TYPE	DESCRIPTION
type	uint8_t	0 = TP, 1=APT, 2=TSK, 3=NRST

Response:
$RCDT,ANS,NAVIGATE,*type,name,lat,lon,elevation,dist,brg,lendable,freq,rwy_dirCS**

SET request:
$RCDT,SET,NAVIGATE,*type,name,lat,lon,elevation,dist,brg,lendable,freq,rwy_dirCS**

PARAMETER	TYPE	DESCRIPTION
type	uint8_t	0 = TP, 1=APT, 2=TSK, 3=NRST
name	string	Name of point
lat	int32_t	Latitude in thousands of minutes (60000 = 1° 0.0')
lon	int32_t	Longitude in thousands of minutes (60000 = 1° 0.0')
elevation	int16_t	Elevation of point in meters
dist	uint32_t	Distance to point in meters
brg	uint16_t	Bearing to point in degrees
lendable	boolean	Radius True (1) if point is lendable
freq	float	Frequency of airport. If point is not lendable this field is empty
rwy_dir	uint8_t	Runway direction in tens of degree (0-36). If point is not lendable this field is empty

SENS

GET request: $RCDT,GET,SENS*CS

Response:
$RCDT,ANS,SENS,*oat,main_volt,backup_volt,reserved,reserved,reserved,reserved,sc_modeCS**

PARAMETER	TYPE	DESCRIPTION
oat	float	Outside temperature in °C. Empty field if OAT is not valid
main_volt	float	Main power voltage
backup_volt	float	Backup battery voltage
reserved		Reserved for future
sc_mode	boolean	SC/vario mode. 0 = vario, 1 = SC