1. Introduction

The RDS encoder implements a subset of the RDS/RBDS functions and is an add-on module to allow SRK FM transmitters to broadcast an RDS sub-carrier.

Control is implemented by a set of serial commands specifically for the RDS encoder. When installed within an SRK transmitter, the RDS encoder handles all serial communication. Serial commands that are not recognised by the encoder are passed to the transmitter proper for processing in the normal way.

The output of the encoder is a nominal 2Vp-p 57KHz RDS subcarrier. This is fed directly in to the PLL circuit of the transmitter.

All control values are stored in non-volatile memory for standalone operation. Both RDS and RBDS PTY codes are supported as well as appropriate AF for all ITU regions.

The encoder is able to be retrofitted to all SRK FM transmitters.

SRK also supplies a Windows application which allows users to control the RDS encoder with a GUI. The application allows encoder configurations to be saved or read as files. It also allows in-system firmware upgrades.

2. Function summary

The RDS encoder supports the following RDS functions:

- PI
- TA
- TP
- Static PS
- Scrolling PS
- RT
- PTY
- MS
- DI
- AF

The PI (Program Identifier) can be set to any hexadecimal value in the range 0000 to FFFF.

The TA (Traffic Announcement) can be set to 0 or 1.

The TP (Traffic Programme) can be set to 0 or 1.

The PS (Programme Service name) can be programmed as an 8 character static value, or up to 32 characters of scrolling text.

The RT (Radio Text) can be programmed with up to 64 characters of text. Up to 4 different text strings can be stored and displayed for preset time periods.

The PTY (Programme Type) can be set to any of the 32 RDS or RBDS programme types.

The MS (Music/Speech) can be set to either Music or Speech.

The DI (Decoder Identification) bits can be set or reset individually.

Up to 10 AFs (Alternative Frequency) may be set.

3. Detailed functions

PI (Program Identifier)
The PI is a 16 bit hexadecimal number. Care should be taken in setting the PI to ensure that no other services in the same area has the same PI as receivers may switch between stations in an unexpected manner.

TA (Traffic Announcement) and TP (Traffic Program)
TA and TP are single bit flags that can be set to 0 or 1. If not used these should be kept at 0.

PS (Program Service name)
This is an 8 character string used to show the station name. This can either be static or scrolling (so allowing more than 8 characters to be displayed). Note that in many countries scrolling PS is not allowed. The RDS encoder allows up to 32 characters of scrolling PI. The time between steps can be set from 1 to 8 seconds. The number of characters scrolled each step can be set from 1 to 8 characters. Scrolling PI should be used with caution as many receivers will not display fast scrolling PS. One useful and fairly common way to use scrolling PS is set the step size to 8 characters and have a PS string that is a series or 8 character strings concatenated together. The strings will appear in sequence at the rate set by the step time.

RT (Radio Text)
Up to 64 characters of radio text can be displayed by the receiver. This is typically used for “now playing” information. The RDS encoder can store and display up to 4 RT strings. These can be displayed sequentially for multiples of 30 seconds, up to a maximum of 180 seconds per message. Any given message can be disabled by setting its display time to “not used”. If only one message is enabled then that message is displayed indefinitely regardless of its display time setting. If all 4 RT messages are disabled then no RT is displayed.

PTY (Program TYpe)
The PTY is a 5 bit value which describes the type of material being broadcast. Refer to the RBDS or RDS standard for a full list of all codes. The RDS and RBDS standards use different codes for different program types. The RDS encoder accepts both RDS and RBDS program types.

MS (Music/Speech)
This is a single bit that can be set to indicate whether music or speech is currently being broadcast. This should be set to “music” if not used.

DI (Decoder Identification)
The DI consists of 4 bits, Compressed, Stereo, Artificial Head, Static PTY. Each of these bits can be set or cleared independently. In practice, the only bit used nowadays is the stereo flag.

AF (Alternative Frequency)
The AF function allows the RDS encoder to specify up to 10 other frequencies that the receiver can switch to in the event that reception is no longer possible on the currently tuned frequency. This is

particularly useful for networks where a number if transmitters have overlapping coverage with the same programming on different frequencies. Normally these frequencies will be other FM channels (87.6 to 107.9MHz). However, both the RDS and RBDS standards allow LF and MF frequencies to also be specified. The frequency range and channel spacing available depends on the ITU region. Therefore, the RDS encoder can be set to ITU region 1, 2 or 3. Although LF and MF AFs are fully specified in the RDS and RBDS standards, many receivers do not implement this functionality. Therefore use of LF and MF AFs should be done with discretion.

Changing the ITU region can have unpredictable results on frequencies already set. Therefore the ITU region should be set before any AFs are specified.

Not all 10 AFs need be used. Any unused AFs can be set to “not used”.

4. Serial Commands

Below are the commands that the RDS encoder recognises. In the following sections <CR> indicates a carriage return (ASCII 0x0D), <LF> indicates a line feed (ASCII 0x0A) and "_" indicates a space (ASCII 0x20).

All commands will generate a response. The RDS encoder does not generate unsolicited responses.

The serial port uses the following settings: 9600 baud, 1 stop bit, no parity, no handshaking.

After each response the RDS encoder sends a <LF>, <CR> and ">". This is not shown in the following sections.

For automated communication the correct procedure is for the host system to send a command and then wait for the response. No new command must be sent until <CR>, <LF>, ">" has been received by the host system. The host system can send each byte of the serial command with no delay between bytes, except for waiting for the response as described above.

All commands are case sensitive, but the characters to the right of any "=" are not.

XAFx? Read alternative frequency x

Returns alternative frequency x (x=1 to 10) as an FM frequency in MHz, LF/MF frequency in KHz or "not used" if AFx is not used.

Syntax:	XAFx?<CR>	
	Where x is 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10	
Example:	XAF1?<CR>	
Response:	97.6MHz	or;
	1026KHz	or;
	Not_used	

XAFx= Set alternative frequency x

Sets alternative frequency x(x=1 to 10). The frequency can be set to an FM (VHF), MF or LF channel. The range of acceptable MF/LF values will depend on which ITU region has been selected (using the XITU=x command).

For ITU region 1 the range of acceptable frequencies are 87.6MHz to 107.9MHz, 153KHz to 279KHz, 531KHz to 1602KHz. FM channel spacing is 0.1MHz. LF/MF channel spacing is 9KHz.

For ITU region 2 the range of acceptable frequencies are 87.6MHz to 107.9MHz, 540KHz to 1700KHz. FM channel spacing is 0.1MHz. MF channel spacing is 10KHz.

For ITU region 3 the range of acceptable frequencies are 87.6MHz to 107.9MHz, 531KHz to 1701KHz. FM channel spacing is 0.1MHz. MF channel spacing is 9KHz.

Syntax: XAFx=nnnKHz<CR> or:
XAFx=mmmMHz<CR> or:
XAFx=not_used<CR>
Where nnn is an MF/LF frequency in KHz, mmm is an FM frequency in MHz with one decimal place.

Example: XAF1=1080KHz<CR>
XAF3=99.7MHz<CR>
XAF9=not_used<CR>

Response: Ok

XAH? Read artificial head bit of DI
Returns the status of the artificial head bit of the DI

Syntax:	XAH?<CR>	
Example:	XAH?<CR>	
Response:	Not_artificial_head	or:
	Artificial_head	

XAH= Program artificial head bit of DI

Allows the artificial head bit of the DI to be programmed

Syntax: XAH=nnn<CR>

Where nnn is “Artificial_head” or “Not_artificial_head”

Example: XAH=Artificial_head<CR> or:

XAH=Not_artificial_head<CR>

Response: Ok

XCOMP? Read compressed bit of DI
Returns the status of the compressed bit of the DI

Syntax:	XCOMP?<CR>	
Example:	XCOMP?<CR>	
Response:	Non-compressed	or:
	Compressed	

XCOMP= Program compressed bit of DI
Allows the compressed bit of the DI to be programmed

Syntax: XCOMP=nnn<CR>
Where nnn is “Compressed” or “Not_compressed”
Example: XCOMP=Compressed<CR> or:
XCOMP=Non-compressed<CR>
Response: Ok

XDPTY? Read dynamic PTY bit of DI
Returns the status of the dynamic PTY bit of the DI

Syntax:	XDPTY?<CR>	
Example:	XDPTY?<CR>	
Response:	Dynamic_PTY	or:
	Static_PTY	

XDPTY= Program the dynamic PTY bit of DI

Allows the dynamic PTY bit of the DI to be programmed

Syntax: XDPTY=nnn<CR>
Where nnn is “Static_PTY” or “Dynamic_PTY”

Example: XDPTY=Static_PTY<CR> or:
XDPTY=Dynamic_PTY<CR>

Response: Ok

XFW? Read firmware version
Returns the version of firmware.

Syntax:	XFW?<CR>
Exampe:	XFW?<CR>
Response:	1.01

XITU? Read ITU region
Returns the ITU region (1, 2 or 3)

Syntax:	XITU?<CR>	
Example:	XITU?<CR>	
Response:	1	or:
	2	or:
	3	

XITU= Program ITU region

Allows the ITU region to be programmed. The ITU region affects the range of MF and LF frequencies that can be set as an AF.

Syntax:	XITU=n<CR>
	Where n is 1, 2 or 3.
Example:	XITU=2<CR>
Response:	Ok

XMS? Read Music/speech bit
Returns the status of the music/speech bit.

Syntax:	XMS?<CR>	
Example:	XMS?<CR>	
Response:	Music	or:
	Speech	

XMS= Program the music/speech bit.
Allows the music/speech bit to be programmed

Syntax:	XMS=nnn<CR>	
	Where nnn is “Music” or “Speech”	
Example:	XMS=Music<CR>	or:
	XMS=Speech<CR>	
Response:	Ok	

XPI? Read the program identifier code
Returns the PI code in the form of a 4 digit hexadecimal number.

Syntax: XPI?<CR>
Example: XPI?<CR>
Response: 1D5F

XPI= Set program identifier code
Allows the PI code to be programmed.

Syntax:	XPI=nnnn<CR>
	Where nnnn is a hexadecimal number in the range 0000 to FFFF.
Example:	XPI=1D5F<CR>
Response:	Ok

XPSSCROLLING? Read scrolling PS string
Returns the scrolling PS string.

Syntax:	XPSSCROLLING?<CR>
Example:	XPSSCROLLING?<CR>
Response:	Your_station_name

XPSSCROLLING= Set scrolling PS string

Allows the scrolling PS string to be programmed.

Syntax:	XPSSCROLLING=nnn<CR>
	Where nnn is a string of up to 32 characters length. This string is case sensitive.
Example:	XPSSCROLLING=Your_station _name<CR>
Response:	Ok

XPSSTATIC? Read static PS string
Returns the static PS string.

Syntax: XPSSTATIC?<CR>
Example: XPSSTATIC?<CR>
Response: Station_

XPSSTATIC= Set static PS string

Allows the static PS string to be programmed.

Syntax:	XPSSTATIC=nnn<CR>
	Where nnn is a string of up to 8 characters length. This string is case sensitive.
Example:	XPSSTATIC=Station_<CR>
Response:	Ok

XPSSTEP? Read scrolling PS step size.
Returns the number of characters that the scrolling PS string moves.

Syntax: XPSSTEP?<CR>
Example: XPSSTEP?<CR>
Response: 3

XPSSTEP= Set scrolling PS step size.
Allows the number of characters that the scrolling PS string moves to be set.

Syntax:	XPSSTEP+n<CR>
	Where n is a value between 1 to 8 characters.
Example:	XPSSTEP=3<CR>
Response:	Ok

XPSTIME? Read scrolling PS step time.
Returns the time in seconds between each movement of the scrolling PS.

Syntax: XPSTIME?<CR>
Example: XPSTIME?<CR>
Response: 4

XPSTIME= Set scrolling PS step time.
Allows the time in seconds between each movement of the scrolling PS to be set.

Syntax:	XPSTIME+n<CR>
	Where n is a value between 1 to 8 seconds.
Example:	XPSTIME=4<CR>
Response:	Ok

XPSTYPE? Read PS type

Returns the type of PS currently selected, static or scrolling.

Syntax:	XPSTYPE?<CR>	
Example:	XPSTYPE?<CR>	
Response:	Static	or:
	Scrolling	

XPSTYPE= Set PS type

Allows the PS type (scrolling or static) to be set.

Syntax:	XPSTYPE=nnn<CR>
	Where nnn can be “Static” or “Scrolling”<CR>
Example:	XPSTYPE=Static
Response:	Ok

XPTY? Read program type
Returns the program type.

Syntax:	XPTY?<CR>
Example:	XPTY?<CR>
Response:	RDS, news

XPTY= set program type
Allows the program type to be programmed. Possible program types are as follows:

RDS,_undefined
RDS,_news
RDS,_current affairs
RDS,_information
RDS,_sport
RDS,_education
RDS,_drama
RDS,_culture
RDS,_science
RDS,_varied
RDS,_pop_music
RDS,_rock_music
RDS,_easy_listening
RDS,_light_classical
RDS,_serious_classical
RDS,_other_music
RDS,_weather
RDS,_finance
RDS,_children's_programmes
RDS,_social_affairs
RDS,_religion
RDS,_phone-in
RDS,_travel
RDS,_leisure
RDS,_jazz_music
RDS,_country_music
RDS,_national_music
RDS,_oldies_music
RDS,_folk_music
RDS,_documentary
RDS,_alarm_test
RDS,_alarm
RBDS,_undefined
RBDS,_news
RBDS,_information
RBDS,_sports
RBDS,_talk
RBDS,_rock
RBDS,_classic_rock
RBDS,_adult_hits
RBDS,_soft_rock
RBDS,_top_40

RBDS,_country
RBDS,_oldies
RBDS,_soft_music
RBDS,_nostalga
RBDS,_jazz
RBDS,_classical
RBDS,_rhythm_and_blues
RBDS,_soft_rhythm_and_blues
RBDS,_language
RBDS,_religious_music
RBDS,_religious_talk
RBDS,_personality
RBDS,_public
RBDS,_college
RBDS,_spanish_talk
RBDS,_spanish_music
RBDS,_hip_hop
RBDS,_unassigned (27)
RBDS,_unassigned (28)
RBDS,_weather
RBDS,_emergency_test
RBDS,_emergency

Syntax:	XPTY=nnn<CR>
	Where nnn is one of the types listed above.
Example:	XPTY=RDS,_easy_listening<CR>
Response:	Ok

XRTx? Read RT string

Returns one of the four radio text strings, where x can be 1, 2, 3 or 4.

Syntax:	XRTx?<CR>
	Where x is 1, 2, 3 or 4.
Example:	XRT2?<CR>
Response:	RT_test_message

XRTx= Set RT string

Allows one of the four radio text strings to be set, where x can be 1, 2, 3 or 4.

Syntax:	XRTx=nnn<CR>
	Where x is 1, 2, 3 or 4 and nnn is a string of up to 64 characters.
Example:	XRT2=RT_test_message<CR>
Response:	Ok

XRTxTIME? Read RT display time

Returns the display time of the radio text string specified by x.

Syntax:	XRTxTIME?<CR>
	Where x is 1, 2, 3 or 4.
Example:	XRT1TIME?
Response:	120_seconds

XRTxTIME= Set RT display time

Allows the display time to be set for the RT string specified by x. Display time can be set to 30, 60, 90, 120, 150 or 180 seconds. Setting the display time to “not used” disables that RT string.

Syntax:	XRTxTIME=nnn<CR>
	Where x is 1, 2, 3 or 4, nnn can be 30_seconds, 60_seconds, 90_seconds, 120_seconds, 150_seconds, 180_seconds, not_used.
Example:	XRT4TIME=120_seconds<CR>
Response:	Ok

XSTEREO? Read stereo bit of DI
Returns the status of the stereo bit of the DI

Syntax:	XSTEREO?<CR>
Example:	XSTEREO?<CR>
Response:	Stereo or:
	Mono

XDPTY= Program the stereo bit of DI
Allows the stereo bit of the DI to be programmed

Syntax: XSTEREO=nnn<CR>
Where nnn is “Stereo” or “Mono”
Example: XSTEREO=Stereo<CR> or:
XSTEREO=Mono<CR>
Response: Ok

XTA? Read TA bit

Returns the status of the TA bit.

Syntax:	XTA?<CR>	
Example:	XTA?<CR>	
Response:	1	or:
	0	

XTA= Set TA bit

Allows the TA bit to be programmed

Syntax:	XTA=n<CR>	
	Where n is 0 or 1	
Example:	XTA=1<CR>	or:
	XTA=0<CR>	
Response:	Ok	

XTP? Read TP bit

Returns the status of the TP bit.

Syntax:	XTP?<CR>	
Example:	XTP?<CR>	
Response:	1	or:
	0	

XTP= Set TP bit

Allows the TP bit to be programmed

Syntax:	XTP=n<CR>	
	Where n is 0 or 1	
Example:	XTP=1<CR>	or:
	XTP=0<CR>	
Response:	Ok	