EVALUATION COPY (EXCERPT)

CONTAINING:
PREFACE
DOCUMENT HISTORY
QUICK START
TABLE OF CONTENTS
INTRODUCTION

Description of the ANSI-C RDS/RBDS Decoder Library

(Library version 4.4)

December 12, 2008

Specification Version 2.30

Confidential and subject to NDA and license agreement.
Controlled Copy Number: 2008XXX/2.
Recipient: XXXXXX

Esslinger Data Engineering ■ Espenstr. 20a ■ D-77656 Offenburg
Phone +49-(0)781-990 79 79 Fax +49-(0)781-990 79 85
info@esslinger.de ■ www.esslinger.de

ESSLINGER data engineering

Important

Legal Notice

The information contained in this publication is subject to change without notice, and replaces and supersedes all information previously supplied. This publication is supplied "as is" without warranty of any kind, either express or implied, including but not limited to the implied warranties or conditions of merchantability or fitness for a particular purpose. In no event shall Esslinger Data Engineering be liable for errors contained herein or for incidental or consequential damages, including lost profits, in connection with the performance or use of the supplied material whether based on warranty, contract, or other legal theory. Esslinger Data Engineering assumes no responsibility for any consequences arising from possible infringement of patents or the rights of third parties arising from the use of any of the supplied material. No license is granted by implication or otherwise under any patent or other rights of Esslinger Data Engineering.

This publication, as well as the associated software, contains proprietary information which is protected by copyright. No part may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording, or otherwise, without the prior written permission of Esslinger Data Engineering.

This document and the information given herein are not to be understood as a replacement for the RDS specification, nor the U.S. RBDS standard, nor any other related specification. Any functionality required for proper operation and compliance is solely defined by the appropriate official specifications. Any recommendations in this document are given for advisory purposes only. No part of this document must be understood to release a design engineer from the care he must take with RDS system design, evaluation and test. The customer is responsible for assuring that proper design and operating safeguards are observed to minimize inherent and procedural hazards. Esslinger Data Engineering assumes no responsibility for applications assistance or customer's product design.

The default values and / or default compiler switch settings used in this RDS decoder library might not be the optimal choice for your specific application. You are advised to consider and try alternative settings in order to find the configuration which offers the best performance for your specific application.

Document History

Version	Changes / Comments	Author	Issued
2.30	• Chapter 3.3: AF reliability / AF usage considerations added • Chapter 4.2: New flags added • New configuration symbols added to chapter 1.6. • New API functions added: - 3.2.5 RDSGSetGDCallback - 3.2.6 RDSGSetPSValidationMethod - 3.3.2 RDSGIsAFDataReliable • Chapter 4.3 added • Annexes A and B added • All API functions: Revised channel selector parameter description. • Minor editorial changes in various chapters. Chapter 1.7 splitted. • Chapter 7: Screen grabs updated	MT	12/2008
2.20	Not publically released	WF	10/2007
2.10a	• Chapter 3.3.2: API function's name corrected in headline.	MT	11/2006
2.10	• Chapter 2.1 splitted in parts a and b, part b added. • Chapters 2.4 and 8 added. • New API functions added: 2.3.24 RDSBGetEBlockCounter, 2.3.25 RDSBResetEBlockCounter, 2.3.26 RDSBGetNumberOfResyncsTriggered, 2.3.27 RDSBGetTotalNumberOfGroupsFround, 3.2.27 RDSGGetRTSegNumber. • API function 2.3.3 RDSBSetRBDSMode described as obsolete. • Introduced the new compiler symbols RDSBBDEC_EN_RBDS_SUPPORT, RDS_DEBUG_TRACE_BBDEC, RDS_DEBUG_TRACE_GRPDEC in chapter 1.6. Updated description of some other symbols in the chapter. • Updated chapter 1.8 • Updated parameter description of API func. 3.2.13 RDSGGetRadioText. • Removed CHG_RT_ABFLAG from basic RDS data change flags, see 4.1. • Minor editorial changes to chapters 1.1, 1.7, 2, 3.3, 3.4,3.5, 3.7, 4.and 7. • Throughout the text: updated spec quotes to IEC 62106.	MT	08/2006
2.02	• Added two new customizing symbols in chapter 1.6: (BASEBANDDEC_NO_TABLES and ENABLE_MJD_CONVERSION) • Revised description of certain compiler symbols in chapter 1.6. • Minor editorial changes to chapter 6.	MT	09/2005
2.01	• Minor editorial changes and clarifications.	WF	08/2005
2.0	• Reflecting the changes in API functions naming due to the new support for the CodeVision compiler. See chapter 1.8 for details. • Editorial changes due to removal of the input buffer from baseband decoder: chapters 1.3, 1.4, 1.5, 2.2, 2.3 Previous chapter 1.3.3 removed (became obsolete) • Reflecting the changes in TMC handling; see chapter 3.6 for details. Replaced TMC_INDICATION flag by GRP8AFOUND flag (see 3.2.12 RDSGGetFlags). Removed the obsolete TMC change flags from the data application change register (see 4.2 RDSGGetDataAppChanges). • Changed switching between RDS and RBDS mode; see description of the RDSBSetRBDSMode / RDSGSetRBDSMode functions • Removed the RDS/RBDS selector parameter from the functions RDSxInitDecoder, RDSxResetDecoder. • Editorial changes in Chapter 1.3.4, "Flowchart for handling traffic announcements" and chapter 1.6 • Renamed chapter 5, includeded further functions to 5. • Inserted chapter 6, renumbering previous chapter 6 into 7.	WF	07/2004

(Earlier change notes were removed from this list. They are available on request.)

A quick start: Topics you should read

Thank you for using our RDS decoder library. We would like to help you with some "do's and don't's" around the library which are good to know when using the library for the first time.

Knowing these basics will prevent your design from showing malfunction or inexplicable effects, and possibly helps you to save a lot of debugging time!

Who are you?

If you are already a **user** of the RDS library and replace an earlier library version with the current one, please review chapter **1.8.1 Migrating from earlier library versions**.

If you are **new to the RDS library**, we recommend that you read at least these chapters prior to designing your system:

- 1.3 General Decoding concept
- 1.4 Bitstream Packing

They will give you a general understanding of how the library needs to interact with the other parts of your application software.

Chapter

- 1.6 Customizing the RDS Library

tells you how to adapt the library to your specific requirements, helping you to save system resources like RAM or CPU cycles which normally are scarce on embedded systems-

When starting application implementation you should familiarize with

- 1.5 Buffer Architecture and Buffer Sizes
- 4. Notification on Data Changes and Events (introduction only)
- 6. Architecturing an RDS decoding project

to learn details of RAM consumption, how RAM availability affects functionality and how a valid project configuration must look like.

Whilst integrating the library to your overall application framework, always remember that chapters 2 and 3 are listing the Application Programming Interface (API) functions **in the sequence you normally are calling them in your application**. Hence, it is good to have a look at every single API function as it is listed one by one and ask yourself if you are calling it at the right time in your application.

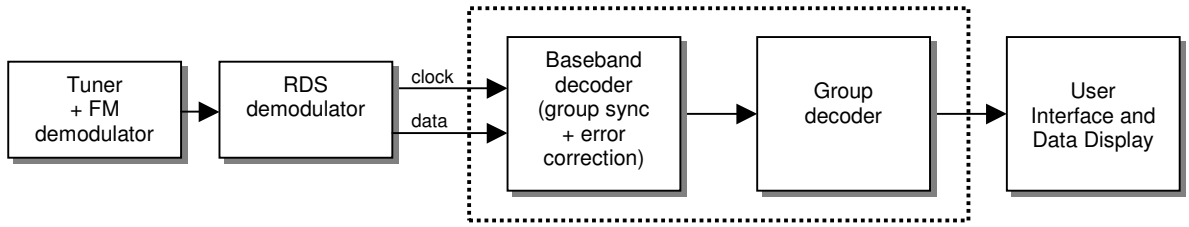
Contents

1. Introduction

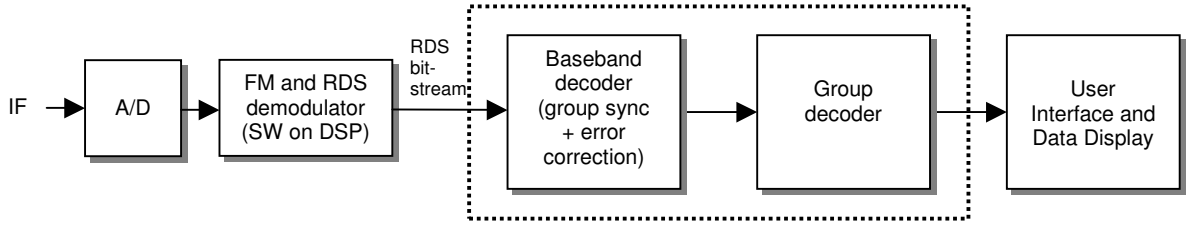
1.1 Overview of RDS / RBDS Decoding

The library described in this document provides functions to decode the Radio Data System (RDS) as defined in IEC 62106:1999. As RDS and the U.S. Radio Broadcast Data System (RBDS) share the same core features, it can egally be used for RBDS. Throughout this document, both the **RDS** and the **RBDS** system are referenced through the common term "**RDS**". Whenever differences between the systems become relevant, a distinct notice is given in the text. Note, however, that this library does not cover any features which are proprietary to RBDS (apart for handling RDS/MMBS multiplex signals on baseband level [see chapter 2.1], and of the different handling of PI code and Programme Type).

A standard RDS-enabled receiver usually comprises the following blocks:



Or, for a software-defined radio:



The RDS demodulator is usually a chip which outputs a serial bitstream together with a clock signal of 1187.5 Hz. Examples for RDS demodulator chips are the SAA6581 (manufacturer nxp), or TDA7478 (manufacturer ST). Alternatively, the demodulator can be implemented in software on a digital radio's DSP.

The data provided by the demodulator must be fed into a data decoder in order to obtain the desired useful data (e.g. Programme Service Name, Alternative Frequencies or Radio Texts). In standalone RDS receivers this decoder is usually implemented as a firmware running on a microcontroller. The first decoder layer performs processing of the baseband signal. Baseband decoding comprises group synchronisation and error correction.

The second decoder layer performs decoding of the RDS groups provided by the baseband decoder. The output of this layer provides the 'useful' RDS data which is presented to the user or used internally for tuning purposes.

In accordance with the abovely given explanations the RDS decoder library is split into **two functional blocks**. The first block is the baseband decoder, performing group synchronisation as well as error checking and correction. The input signal to this block is called the **RDS bitstream** which is output by any kind of RDS demodulator.

At its output the baseband decoder provides the common **RDS group structure** which comprises 4 RDS blocks with each one having 16 bits. In addition to group synchronisation and error checking, the

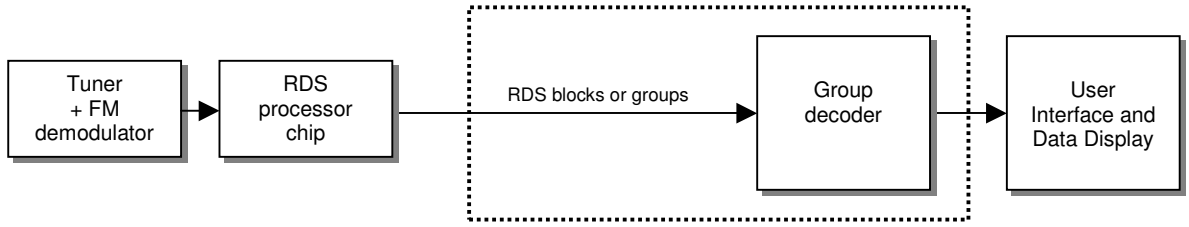
baseband decoder maintains statistics about the RDS signal quality. The baseband decoder is described in chapter 2.

The second functional block is a group decoder which takes RDS groups at its input and provides **decoded RDS data** at its output. The user application interacts with this decoder by calling the appropriate "Application Programming Interface“ (**API**) functions. The group decoder is described in chapter 3.

An alternative RDS receiver design

Baseband decoding hasn’t necessarily to be done in software. There are also chipsets available where the baseband decoder is combined with the RDS demodulator in a single chip. Examples are the SAA6588 (nxp) or TDA7333 (ST). Usually these chips output RDS blocks or entire RDS groups via the I^2C bus.

The benefit of a RDS processor chip is that it helps to save CPU resources on the application-processing microcontroller. And that you don't have to deal with baseband decoding (what, however, you don't have to do anyway when using our RDS library). In practice, receiver designers must find the tradeoff between possibly reduced costs by using a microcontroller with lower performance, and the increased costs of a RDS processor (in comparison to a simple demodulator chip).



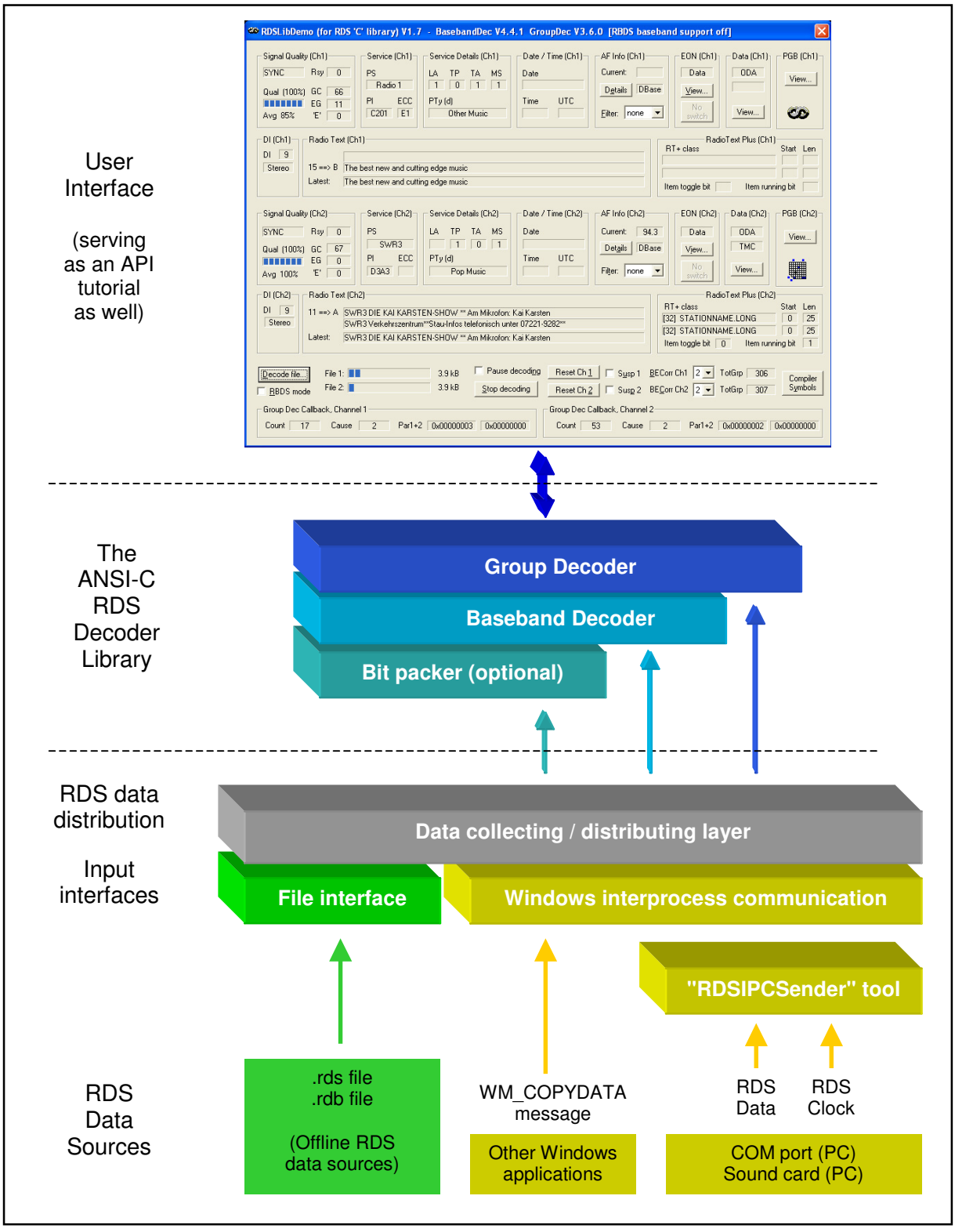
Using the RDS library with RDS processors is egally possible. In this case the data source for the group decoder is directly the RDS processor chip. The baseband decoder software layer can be omitted. The group decoder's API usage, as well as its output data, remain the same, no matter whether a software or a hardware baseband decoder is used. See also chapter 1.3 General Decoding concept

However, please note that a few things are different when using a hardware baseband decoder:

- Some small intermediate software layers may become necessary to transfer the RDS processor's output data into a format that can be handled by the group decoder's input. Such intermediate software layers are not provided with the RDS library code. The Esslinger field application engineers are at your disposition, should you need assistance.
- RDS signal quality information can only be obtained from the RDS processor. So the extent of the quality information depends on the chip. Some RDS processors provide just basic signal quality information.
- On some hardware baseband decoders, the bitstream sync-up speed and the error correction logic was found to work not as efficiently than in the software baseband decoder. The software decoder uses sophisticated correction strategies which also consider a certain "signal history". Hardware decoders usually implement a straightforward error correction approach.

1.2 RDS Tutorial Application Architecture

Together with the RDS decoder library a test / demo / tutorial environment for the PC is provided. It comprises the "RDSLibDemo" application pictured below, helper applications and RDS data samples. The demo application is provided with all source codes as a project for Microsoft® Visual C++® 6 and higher. Detailed information is provided in chapter 7 as well as in the "Readme" file in the CD-ROM's root directory.



1.3 General Decoding concept

RDS decoding is performed by calling the tasks pictured below in an endless loop (the so called **decoding loop**). Usually the RDS functions in the loop are processed every **100 to 500 milliseconds** (the so called **decoding interval**). The decoding interval's value is uncritical and can be selected according to the the systems designer's reqirements. However, it should be considered that the longer the period is, the more RAM must be provided for data buffering in order to avoid data loss. RAM availability may be a problem on certain microcontroller-based systems.