Exploiting The Dynamic Flexibility Of Software Radio In FM Broadcast Receivers

Declan Flood[ϕ], Linda Doyle, Philip Mackenzie, Keith Nolan and Donal O'Mahony

Networks and Telecommunications Research Group (NTRG)
Trinity College, Dublin 2
IRELAND
[ϕ]flooddk@tcd.ie http://ntrg.cs.tcd.ie/swradio.php

Abstract – **This paper demonstrates how the flexibility of software radio may be exploited to optimise radio communications systems. It describes our implementation of a software radio RDS (Radio Data System) receiver. Most FM stations transmit a RDS signal in addition to their audio output. RDS provides information about the current FM broadcast such as the program name etc. RDS is very similar to the North American RBDS (Radio Broadcast Data System). Our implementation of the receiver uses high level RDS information to adapt itself. The objective is to improve the receiver's functionality for the user and to reduce the computational load.**

I INTRODUCTION

In software radio transceivers the lower level functions such as channel filtering, modulation etc are implemented in software. In software radio receivers an A/D converter is used to digitise the received signal as close to the antenna as possible. Typically, there is only one stage, an IF downconverter, between the antenna and processing unit as shown in figure 1.

In general, software radios may be divided into two classes based on what type of processing unit is used. Reconfigurable hardware such as an FPGA or DSP may be used to process the output of the A/D converter. A second class of software radio uses GPPs (General Purpose Processor) such as an Intel Pentium [1].

Software radios implemented using GPPs have a number of advantages over reconfigurable hardware approaches. Software for GPPs may be written in easy to use high-level languages such as C++. These languages allow a faster development time than low-level DSP assembler languages or hardware description languages such as Verilog or VHDL. Software written in high-level languages is portable; it does not need to be rewritten when a faster family of processors becomes available. Also, a GPP based software radio maximizes flexibility. This flexibility is the focus of this paper. As research into GPPs progresses their significant weaknesses such as high power consumption will improve.

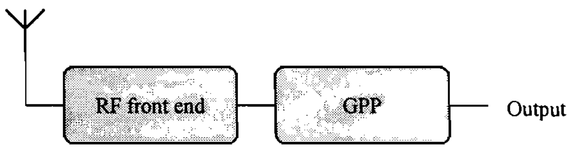


Figure 1: A software radio receiver

By flexibility we mean both the parameters of the stages within the system may be changed and the architecture of the system may be changed. These changes may occur at start-up or during operation.

The flexibility of the system may be exploited to tailor its functionality to user demands or to minimise the use of resources (computational load or power consumption). The system may also have to respond to changes in the channel (multipath effects and signal-to-noise ratios) or government regulations (bandwidth and RF power constraints).

This paper describes an example of exploiting the flexibility of software radios. In this case a software radio FM broadcast receiver responds to information provided by RDS[1](Radio Data Service). RDS is a data signal transmitted by most FM broadcast stations in addition to their audio output. It sends information such as the current time and the name of the radio station etc [2]. Our implementation of the receiver responds to this high-level information to reduce the computational load on the GPP and to improve the listening experience for the user.

Section II details the aims and specifications of RDS. Section III describes our software radio implementation of a RDS receiver. Section IV provides some examples of how we have used high level RDS data to optimise lower level functions. Section V concludes with reference to other work in this area.

II RDS

RDS is a data signal transmitted by most FM broadcast stations in addition to their audio output. The objective of RDS is to improve the functionality of FM receivers by providing additional information to users. This information allows a range of applications such as displaying the programme service name or correcting the receiver's clock. It also facilitates more advanced features such as paging of users or auto-tuning for automotive applications. Auto-tuning information is used when the received signal strength becomes low because the receiver has moved out of the transmitter's range. RDS transmits a list of alternate frequencies where the same radio program is being transmitted. The receiver checks if the signal is stronger on one of the alternate frequencies. This allows users to travel without having to retune their receiver.

The RDS signal is frequency multiplexed with the audio signals. The modulation bandwidth of FM transmitters is 90kHz but only 53kHz of bandwidth is required for stereo signals. Space above 53kHz is used to transmit the RDS signal. The RDS signal is transmitted on a 57kHz sub-carrier. The modulation system used is D-BPSK (Differential Binary Phase Shift Keying). Figure 2 shows the output of the FM discriminator in a receiver. The diagram shows the 'left plus right speaker' signal, as well as the 19kHz pilot tone and the 'left minus right speaker' signal used for stereo reception. The RDS signal has a null at 57 kHz and a width of 4.8kHz.

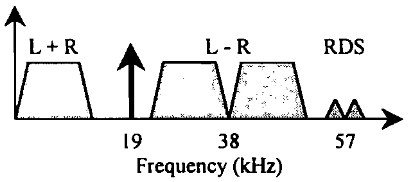


Figure 2: FM discriminator output.

[1] RDS is used in Europe and is defined by the EBU (European Broadcasting Union) [3]. RBDS (Radio Broadcast Data System) is the North American equivalent and is defined by the National Radio Systems Committee [4]. The two standards are very similar [5].

RDS has a data rate of 1187.5 bits per second. The bit stream is divided into groups of 104 bits. Each group is an entity in itself and has a particular purpose such as transmitting a chunk of the program service name, clock-time information, differential GPS information etc.

In traditional radios the output of the FM discriminator is passed to stereo decoders and to a RDS demodulator IC. The stereo decoders generate the audio output. The RDS demodulator IC provides a bit stream to a microprocessor. The microprocessor performs group synchronization and error correction.

III RDS IN SOFTWARE RADIO

We have developed a software radio test bed. The test bed consists of both the hardware to collect data plus a software development environment that allows assembly of software signal processing components. The hardware consists of an antenna, RF front end and an A/D converter. The RF front end amplifies the RF signal and downconverts it to an IF of 10.7Mhz. The A/D converter samples the IF signal, using bandpass sampling.

IRIS (Implementing Radio In Software) is a software development environment allowing assembly of software components to create software radios. It consists of a repository of signal processing components plus a software radio engine. XML documents describe a software radio in terms the components required and their layout. The software radio engine takes this description and assembles the software radio using components from the repository. It is important to note that the software radio engine also allows the architecture of the system or the parameters of the components to be altered during operation. This ability to adapt is exploited in this paper to optimise the radio receiver [6].

The software components required to implement an FM broadcast receiver are shown in figure 3. The data acquisition component controls the flow of data from the A/D converter to the PC. The channel extractor (mixer, low pass filter and decimator) isolates the signal of interest. An FM discriminator performs demodulation. The spectrum of the FM discriminator output is shown in figure 2. This signal is passed to two chains of signal processing components. The first chain generates the audio output using a low pass filter followed by deemphasis and equalizer components. The second chain generates groups of RDS data using a band pass filter, costas loop and RDS decoder component.

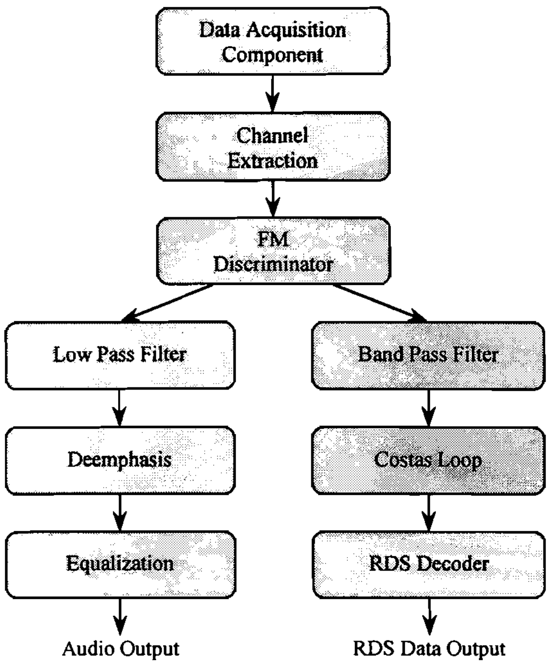


Figure 3: IRIS components in a mono FM receiver.

A block diagram of the RDS decoder component is shown in figure 4. An early-late gate synchronizer performs symbol timing. A bit stream is generated using a symbol decoder. An XOR function performs differential decoding. Finally, a group synchronization algorithm followed by an error correction algorithm (Meggitt decoder) creates groups of RDS data.

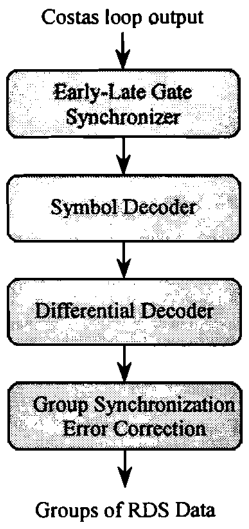
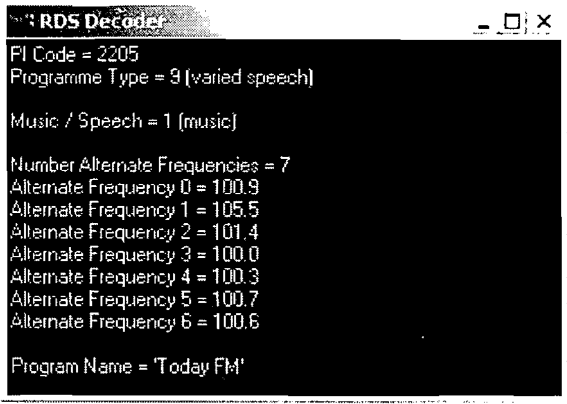


Figure 4: Block diagram of the stages inside the RDS decoder component.

The output of the RDS component is a stream of error corrected groups of RDS data. Each group is an entity in itself and the type of payload carried is indicated by a 4-bit group type code. Figure 5 shows some data collected from a radio station. Our implementation runs in real-time on a 2GHz Pentium IV running Windows 2000.



```
PI Code = 2205
Programme Type = 9 (varied speech)

Music / Speech = 1 (music)

Number Alternate Frequencies = 7
Alternate Frequency 0 = 100.9
Alternate Frequency 1 = 105.5
Alternate Frequency 2 = 101.4
Alternate Frequency 3 = 100.0
Alternate Frequency 4 = 100.3
Alternate Frequency 5 = 100.7
Alternate Frequency 6 = 100.6

Program Name = 'Today FM'
```

Figure 5: RDS data output.

IV EXAMPLES OF USING FLEXIBILITY

RDS was originally designed to improve the functionality of receivers. However the combination of software radio and RDS enables additional functionality. Here we describe three novel ideas we used which improve the functionality of the radio or reduce the computational load on the GPP.

a) Adapting equalizer based on program type

The final stage in generating the audio output is an equalizer as shown in figure 3. The equalizer allows users to change the bass and treble settings to suit their own taste. Users will have preferred settings for each type of music they listen to. For example for jazz they may want to increase the treble and reduce the bass.

RDS transmits a PTY (Program Type Code), this is a 5-bit code indicating the type of program the user is listening to e.g. jazz, rock, country etc. In our implementation, the equalizer settings respond to changes in the program type code.

b) Relaxing filter roll-off requirement

FM transmissions may be in mono or in stereo. The spectrum of a stereo transmission is shown in figure 2. A mono transmission is similar, but it does not have a pilot tone at 19kHz or a 'left minus right speaker' signal centred on 38kHz.

In this paper we consider a mono receiver only. In mono receivers the audio output is generated using a low pass filter. If the transmission is in stereo then a there will be a pilot tone at 19kHz. In this case the roll off of the low pass filter must be steep enough to ensure the pilot tone is strongly attenuated, as shown in figure 6(a).

However if the transmission is in mono then the steepness of the roll off may be relaxed as shown in figure 6(b). This leads to a reduction of the number of taps in this low pass filter and hence fewer operations are required to implement the filter. Information from the RDS decoder is used to determine whether the transmission is in stereo or mono.

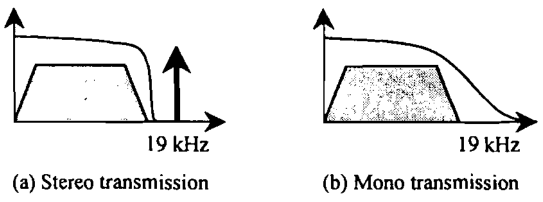


Figure 6: Amplitude response of audio low pass filter

This reduction in CPU load may be used to throttle the clock rate, reducing power consumption.

c) Removing deemphasis component for speech

The noise rejection of FM systems is more favourable at low frequencies than at high frequencies. To counter this effect FM transmitters increase the amplitude of the high frequency components of the audio signal prior to transmission. This technique is called preemphasis. Receivers have a complementary deemphasis component to produce a flat overall frequency response.

While a deemphasis component is always of benefit to audio quality, it is more important for music signals than speech signals. This is because speech signals do not have as many high frequency components. Also, a reduction in audio quality is more acceptable for speech programs than for music programs.

RDS includes a 1-bit flag indicating whether the current transmission is music or speech. Our implementation of the receiver responds to changes in this flag. For speech transmissions the IRIS system will remove the deemphasis component. When a music transmission restarts the deemphasis component will be reinserted in the receiver chain.

V CONCLUSIONS

We have demonstrated how the flexibility of software radio receivers may be exploited in FM broadcast receivers.

When RDS was developed in the late 1970's it was not anticipated that receivers could incorporate such high levels of flexibility. Therefore the three examples given provide only modest improvements in functionality and computational load. This is because we are working within the constraints of a pre-existing system.

Bose [7] describes a novel software radio data communication system. Parameters such as the modulation scheme, bandwidth etc may be changed on a per packet basis in response to user constraints such as data rate etc. This leads to a significant improvement in bandwidth used and power consumption.

While it is possible to implement flexible systems using traditional design techniques, a software radio implementation becomes preferable in maximally flexible systems. For example, consider a system in which the modulation scheme may be switched between BPSK, DQPSK, 16-QAM and GMSK. A hardware implementation will require four different demodulation circuits. Another disadvantage of hardware implementations is that all flexibility must be defined at system specification. Upgrading deployed units requires expensive firmware upgrades. Software radio units that have already been deployed may be easily upgraded by downloading new components or XML documents.

REFERENCES

[1] W. Tuttlebee, "Software-defined radio: facets of a developing technology", *IEEE Personal Communications*, April 1999.

[2] P. Mothersole, "Broadcast data systems: teletext and RDS", 1992.

[3] CENELEC/ECU, "RDS Standard EN500067: 1998", 1998.

[4] National Radio Systems Committee, "United States RBDS Standard", April 1998.

[5] National Radio Systems Committee, "RBDS versus RDS – What are the differences and how can receivers cope with both systems", January 1998.

[6] L. Doyle, P. Mackenzie. "A general purpose processor component based software radio engine". Second European Colloquium on Reconfigurable Radio, Athens Greece, 2002.

[7] V. Bose, R. Hu and R. Morris, "Dynamic physical layers for wireless networks using software radio". *International Conference on Acoustics, Speech, and Signal Processing 2001.*