

AN EFFICIENT SCALABLE SYSTEM FOR REMOTE CONTROL OF DISTRIBUTED DEVICES

S. M. Aziz, Hoang Nguyen, Jarred Biddell and Andrew Brine

School of Electrical & Information Engineering, University of South Australia
Mawson Lakes Campus, SA 5095, Australia
E-mail: mahfuz.aziz@unisa.edu.au

ABSTRACT

This paper presents a flexible and cost effective solution to the problem of controlling a large number of remotely distributed devices from a central location. The key components of the system are the use of an efficient low bandwidth scalable communication protocol, the use of existing infrastructure for the communication of control commands, and the development of a scalable power electronic control circuit for the remote sites. A laboratory prototype of the system has been built and successfully tested. This multidisciplinary project involved close collaboration among electronics, communication and computer engineering teams.

1. INTRODUCTION

There is growing demand for systems capable of controlling large number of distributed devices from a central location. The desirable characteristics of such a system include scalability that is, the number of devices that can be controlled, a mechanism to group them into various categories, and the efficiency of communication. The Radio Data System (RDS) [1] offers some unique advantages for such applications. The Open Data Application (ODA) feature [1-3] of RDS makes it suitable for customising the protocol for application specific functions. This paper presents the design of a system using RDS for controlling a large number of distributed devices. The major focus of this work is to design a system that is cost effective, efficient and scalable. The considerations on scalability apply to both the communication as well as the power electronic control aspects. A laboratory prototype of the system has been built in order to test the concept. For simplicity the generation of RDS messages has been emulated using a microcontroller. The decoding of RDS messages (command interpretation) and power electronic control have been implemented

using a second microcontroller. Section 2 presents an overview of the structure and operation of the system. Section 3 briefly presents an RDS ODA based communication protocol model developed for our application. Section 4 presents the I²C decoding scheme for the RDS messages received from the central command station. Section 5 presents the power electronic control circuit.

2. SYSTEM STRUCTURE

Fig. 1 shows an overall structure of a RDS based control system. RDS technology utilises an existing commercial FM transmission facility to embed digital signals over FM channels. A RDS encoder is used at the central station to prepare the RDS messages for the remote devices. Each remote location has a demodulator and a RDS decoder in order to retrieve the original RDS signal and to decode it to fetch the actual message (information) respectively. The command decoder at the remote location interprets the message and performs the necessary function.

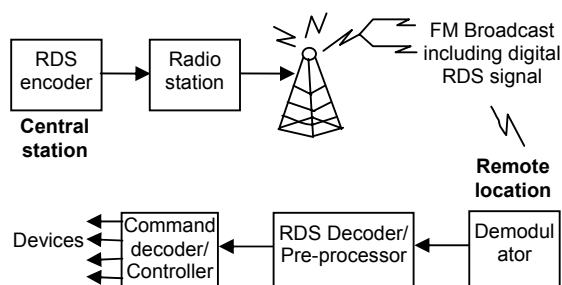


Fig. 1 Overall structure of the control system

3. PROTOCOL DESIGN

First we briefly present the main features of the RDS standard. Then we present the protocol model we have developed based on RDS.

3.1 RDS Standard

The RDS digital signal is multiplexed on a 57 KHz sub-carrier of an FM baseband signal [3-4]. RDS signals are transmitted in groups of 104 bits. Each group is divided into 4 blocks of 26 bits. Each block carries 16 bits of information and 10 bits of overhead for error correction. This effectively leaves only 64 bits per group for information [1]. Some of these 64 bits must always be used to carry certain information according to the standard as shown in the Fig. 2 [1], [3]. For example, all the bits of block 1 are occupied by the Program Identification (PI) code. The 11 most significant bits of block 2 are occupied by the Group Type (GT) code, Traffic Program (TP) flag and the Program Type (PTY) code. This reduces the number of bits carrying actual information to 37 per group.

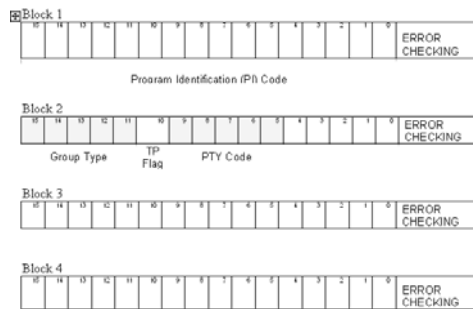


Fig. 2 Four blocks in an RDS group

It is clear that the RDS system offers limited bandwidth for transmitting information. In order to develop a service that is not defined in the standard and to maximise the utilisation of the available bandwidth the ODA features of RDS need to be used [1], [3]. Twelve out of the 32 RDS group types allow ODA features; only 3 of these groups are of type A, which allows all the 37 available bits to be used for transmitting actual information [1]. These are group numbers 11A, 12A and 13A [1], [3]. However, when using a RDS ODA, a message of group type 3A must be transmitted first followed by the ODA group being used. This group type (3A) uses 21 of the 37 bits stated earlier, for application identification and for specifying the ODA group type being used, leaving only 16 bits for carrying actual information. Therefore, in an RDS ODA, a total of 53 bits (16+37) are available for carrying information with the transmission of a type 3A message followed by an ODA group message [1].

3.2 Protocol Features

For a protocol having limited bandwidth, all required functions cannot be specified with the transmission of one message [5]. For example, a certain message

may be required to carry information about time whereas another message may carry text information. To set aside bandwidth for certain specific information when it may not be used in some circumstances is a waste and inefficient use of bandwidth. Hence messages need to be structured in a way that efficiently accommodates the array of functions a system needs while maximising the use of the available bandwidth [6]. This will require a high level of *software decoding*. This means that a device such as a microcontroller has to be programmed to receive and interpret these messages and perform the specified functions.

3.3 Addressing Scheme

An important feature of any protocol architecture is to be able to differentiate between potential recipients of a given transmission. Hence an appropriate addressing scheme needs to be devised. Most protocol architectures have the ability to direct messages to one recipient (unicast), a set of recipients (multicast) or to all recipients (broadcast) [7]. The type of addressing used can be represented using a 2-bit code. The *depth of addressing* would depend on the number of recipient devices and the nature of the system. As stated in Section 3.1 a total of 53 bits are available in an RDS ODA for carrying actual information. Therefore, with 51 bits remaining the addressing can be very deep. Obviously not all 51 bits would be used for addressing, as some bits need to be allocated to specify functionality.

3.4 The Protocol Model

A system based on the proposed protocol model could be used for a number of applications ranging from remotely controlling devices to simply displaying text messages. To describe how this protocol would work for a system the following set of generic specifications are used [8]:

- Each receiving unit has a unique address. A *24-bit code* is used to identify each unit allowing up to 16,777,216 unique addresses.
- Each unit performs 8 different functions and therefore has to interpret 8 different messages. A *3-bit code* in the message identifies the function.
- Each unit is a member of a group of units. A *16-bit group code* is used; this gives 65,536 group addresses.
- A unit can change its group membership if instructed. This means that *unit address* and *group address* must be held independently.

The addressing scheme will affect the number of bits available for specifying functionality. Table 1 illustrates this. It is clear that the number of

commands available for a given application decreases with an increased level of addressing due to the diminishing bandwidth. How much bandwidth is required will depend on the requirements of the system. For example, if the requirement was to control 8 digital switches then 24 bits would be plenty. If it was an application that required higher levels of message encryption it may not be enough.

Table 1: Addressing mode vs. bits available

Addressing Mode	Bits Required	Bits available
Unicast	24 + 2 + 3	24
Multicast	16 + 2 + 3	32
Broadcast	0 + 2 + 3	48

4. I²C DECODING SCHEME

Every remote station has a receiver unit consisting of a demodulator, a RDS decoder and a command decoder as described in Section 2. The FM signal received is first demodulated and then the RDS decoder retrieves the RDS message. These messages are logical RDS data blocks suited to serial communication. Commercial off the Shelf (COTS) RDS decoder chips [9] present the decoded RDS messages onto one of its output pins as a serial bit stream as per I²C specification [10]. I²C works in a master/slave configuration. The RDS messages are sent to a master microcontroller for processing (command interpretation and operation). For our laboratory prototype we used a second microcontroller in place of the RDS decoder in order to emulate the I²C communication interface. Fig. 3 illustrates the set up for our laboratory prototype. The bi-directional I²C bus consists of a serial data line (SDA) and a serial clock line (SCL). Data is sent serially via the SDA line. Control of this line belongs to the device sending the data. The SCL is always controlled by the master. The master manages synchronisation of the data transmission by generating a clock pulse on the SCL. A signal called data available (DAVL) is used to let the master know that the slave has data to send.

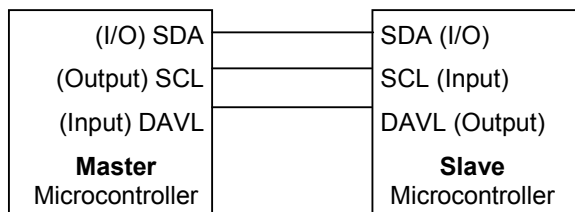


Fig. 3 I²C serial communication setup

The master microcontroller we used is a MC68HC908JL3 [11] which does not have a built in I²C communication interface. This microcontroller was chosen due to its availability, low cost, large number of available I/O ports and familiarity of the designers with the 6808 family of microcontrollers. This meant that the I²C communication control had to be implemented in software. In addition, due to using a second microcontroller (slave) to generate the RDS messages (bit streams) this controller had to be programmed as well for generation of such messages. If a receiver unit is designed with a RDS decoder chip [9] and a microcontroller with built in I²C interface then the need for complex software I²C decoding would be eliminated. Such a scheme would also render better performance. However, the configuration we used was good enough for our goal of testing the overall concept including the RDS based communication protocol algorithms and the electronic control circuits we developed.

The arrival of a RDS message was simulated on the slave microcontroller by the press of a push button. When pressed, the slave asserts DAVL (active low) prompting the master to initiate the data transfer process. The master relinquishes control of the SDA line to the slave for the duration of the data transfer. The master interprets the incoming message and performs the necessary function.

5. CONTROL CIRCUIT

The electronic control circuit we designed is shown in Fig. 4. We chose to test the operation of the remote control system by controlling some electrical load (lamps in our laboratory prototype). Obviously the configuration of the control circuit would be different for different applications. The control circuit receives input from the master microcontroller at point B. Depending on the command received by the master it outputs either a logic LOW or a logic HIGH at point B. For a logic LOW at point B, the load remains connected to the power supply. However, for a logic HIGH at point B the load is disconnected.

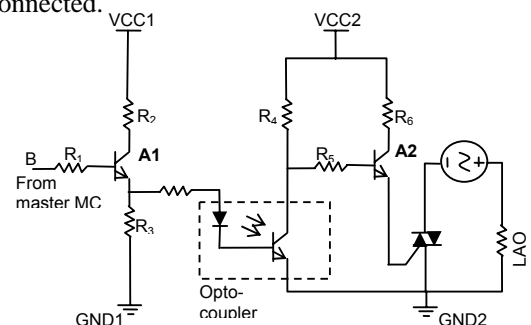


Fig. 4 Electronic control circuit

The current driving capability of the microcontroller output port (B) is increased by amplifier A1. The second amplifier (A2) amplifies the current from the opto-coupler to supply current to the gate of the triac. The opto-coupler physically separates the high power circuit from the low power microcontroller circuit for protection of the later from possible damages. Scalability in the design is achieved by ensuring that the triac can be replaced by one with a different rating and controlling the triac's gate current accordingly by simply adjusting R_5 . This would not require any other change in the circuit for a reasonably wide range of electrical loads.

6. TESTING

Test strategies were devised to test the algorithms developed to implement the remote communication protocol based on RDS. The slave microcontroller was programmed to generate and send various messages to the master for it to perform the following operations:

- Control switches based on group ID.
- Control switches based on unit ID.
- Reassign group code of an individual unit.
- Test all functions of a unit based on group/individual ID
- Set an idle timer based on group/individual ID.
- Set an ignore timer based on group/individual ID.
- Reset a unit based on group/individual ID.

The master microcontroller was programmed to interpret each message and perform the necessary operation. Apart from the above, software programs were developed to ensure full compliance with the I²C interface for serial communication of the messages from the slave to the master. The sequence and timing of various signals for the I²C interface were carefully coded into the software. Tests were carried out by sending messages to perform each of the above operations. In addition to addressing the units individually (unicast) tests were also carried out using multicast and broadcast messages. The test results indicated successful operation of the system in all cases and validated the algorithms developed for both remote communication and I²C interface. Test results also validated the command structure and the various specific commands (messages) designed. With the simple command structure proposed it is possible to control millions of distributed devices.

7. CONCLUSIONS

An RDS based remote control system has been presented in this paper. The system is capable of

addressing more than 16 million remote units with unique address. It has the flexibility of grouping units into more than 65000 groups. It is possible for units to change membership from one group to another. It is a highly scalable system in terms of size and group membership. Therefore it can be used from small to very large distributed systems. Protocol algorithms and software have been developed for both remote communication and the I²C interface used by RDS decoders. A scalable electronic control circuit has also been designed. Successful testing of the laboratory prototype validated the algorithms and messages developed. This work has laid the foundation for the realisation of a practical system involving RDS encoders, decoders and FM communication channels. The use of existing FM communication infrastructure would make the use of this system very attractive for a variety of remote control and information dissemination applications.

REFERENCES

- [1] D. Kopitz and B. Marks, RDS: The Radio Data System, Boston: Artech House Publishers, 1999.
- [2] J. P. Linnartz, "Spectrum efficiency of the Radio Data System," IEEE Transactions on Broadcasting, Vol. 39, No.3, pp. 331-334, September 1993.
- [3] CENELEC, Specification of the Radio Data System, Standard No. EN50067, European Committee for Electrotechnical Standardization, April 1998.
- [4] M. de Ridder - de Groote and R. Prasad, "Analysis of new methods for broadcasting digital data to mobile terminals over an FM channel," IEEE Trans. on Broadcasting, Vol. 40, No.1, pp. 29-37, Mar. 1994.
- [5] D. Kopitz, "The Development of the RDS system from a European point of view and the possibilities for distributing traffic message with RDS," in IEE Colloquium on RDS System, pp. 1 – 10, 2 Dec. 1988.
- [6] A. Chandrakasan, "A Low Power, Low Bandwidth Protocol for Remote Wireless Terminals," in Proc. Global Telecom Conf., Vol. 1, pp. 22-28, 18-22 Nov. 1996.
- [7] W. Stallings, Data and Computer Communications, New York: Prentice Hall, 2003.
- [8] H. Nguyen, S. M. Aziz and Trent Ryan "A Low Bandwidth Communication Protocol for Remote Control Applications," to appear in IEEE Int. Conf. TENCON 2004, Thailand, 21-24 Nov. 2004.
- [9] Phillips Semiconductors "SAA6588 RDS/RBDS pre-processor," Product specification, Sept. 1997.
- [10] Phillips Semiconductors "The I²C-bus specification," Version 2.1, January 2000.
- [11] Motorola, "Technical Data- MC68HC908JL3", Doc. No. MC68HC908JL3/H, Rev. 1.0, Jan. 1999, http://e-www.motorola.com/webapp/sps/library/prod_lib.jsp