Packaging and transmitting RDS messages using the QN8066

Ricardo Lima Caratti

## Introduction

The growing popularity of Arduino and the need for a more accessible development platform for the QN8066 microcontroller motivated the development of this library (PU2CLR QN8066 Arduino Library). By providing an intuitive and familiar interface to Arduino users, this library aims to democratize access to the QN8066 and streamline the development of projects involving FM radio transmissions with additional information via RDS.

The Radio Data System (RDS) allows digital information to be transmitted alongside traditional FM radio broadcasts. To effectively send RDS data, such as station names, it's essential to understand the specific structure and bit sequences involved. This paper presents a study on the implementation of the Radio Data System (RDS) using the QN8066 microcontroller. The study addresses the methodology for **packaging and transmitting** additional digital information, such as the station name, along with the FM radio signal. To this end, the structure of RDS data and the configuration of the QN8066 to generate and transmit these packets efficiently are detailed.

## RDS Message Format

### **1. Program Identification (PI) Block**

**The first block in every group always contains a Program Identification (PI) code.**

The Program Identification (PI) code is a critical component of the RDS (Radio Data System) protocol. It is used to uniquely identify the radio station transmitting the RDS data. The PI code is a 16-bit value (4 hexadecimal digits) assigned to each station.

**PI Code Structure**

The PI code is a 16-bit binary value (4 hexadecimal digits), and its structure is as follows:

1. **Country Identifier (First 4 Bits)** - The first 4 bits of the PI code represent the country identifier. This helps to differentiate stations in different countries. For example, the code 0x1 might represent one country, while 0x2 represents another.
2. **Program Type (Next 4 Bits)** - The next 4 bits indicate the program type. This categorizes the type of content being broadcast, such as music, news, sports, etc.
3. **Program Reference Number (Last 8 Bits)** - The last 8 bits provide a unique reference number for the specific station or program.

**Example**

**Let's consider an example PI code: 0x1234.**

0x1: Country identifier

0x2: Program type

0x34: Program reference number

### **Visualizing the PI Code**

To visualize how the Program Reference Number fits into the PI code, consider the following:

| **Bits** | **15-12** | **11-8** | **7-0** |
| --- | --- | --- | --- |
| PI Code | Country ID | Program ID | Reference # |

The Program Reference Number is a crucial part of the PI code, providing a unique identifier for each station or program within a country and content type category. By using this structure, the RDS system ensures that each station can be uniquely identified and tracked, facilitating features like seamless frequency switching and accurate station identification.

### **2. Group Application Code:**

The first four bits of the second block of every group are allocated to a four-bit code which specifies the application of the group. Groups are referred to as types 0 to 15, according to the binary weighting of A3, A2, A1, and A0. For each type (0 to 15), two "versions" can be defined:

a) Version A (B0 = 0) - The PI code is inserted in block 1 only.

b) Version B (B0 = 1) - The PI code is inserted in blocks 1 and 3 of all group types.

### **3. Program Type Code (PTY) and Traffic Program Identification (TP)**

**The PTY code and TP identification occupy fixed locations in block 2 of every group.** This code is an identification number transmitted with each program item, intended to specify the current program type (e.g., news, sports, etc.).

| **PTY Code** | **Program Type** |
| --- | --- |
| 0 | No PTY (undefined) |
| 1 | News |
| 2 | Current Affairs |
| 3 | Information |
| 4 | Sport |
| 5 | Education |
| 6 | Drama |
| 7 | Culture |
| 8 | Science |
| 9 | Varied |
| 10 | Pop Music |
| 11 | Rock Music |
| 12 | Easy Listening |
| 13 | Light Classical |
| 14 | Serious Classical |
| 15 | Other Music |
| 16 | Weather |
| 17 | Finance |
| 18 | Children's Programs |
| 19 | Social Affairs |
| 20 | Religion |
| 21 | Phone-In |
| 22 | Travel |
| 23 | Leisure |
| 24 | Jazz Music |
| 25 | Country Music |
| 26 | National Music |
| 27 | Oldies Music |
| 28 | Folk Music |
| 29 | Documentary |
| 30 | Alarm Test |
| 31 | Alarm |

### **Bit Structure of Block 2**

Block 2 in an RDS group contains several fields, each with a specific function. Here's the breakdown of the bit structure:

| **Bits** | **Field** | **Description** |
| --- | --- | --- |
| 0-3 | Group Type Code | Specifies the type of RDS group (e.g., 0A, 0B, etc.) |
| 4 | Version Code | Specifies the version of the RDS group (0 for version A, 1 for version B) |
| 5-9 | Program Type (PTY) | Specifies the type of program content (e.g., News, Sports) |
| 10 | Traffic Program (TP) | A flag indicating if the station broadcasts traffic announcements (0 = no, 1 = yes) |
| 11-15 | Traffic Announcement (TA) | A flag indicating if a traffic announcement is currently being broadcast (0 = no, 1 = yes) |
| 16-31 | Other Data | Data specific to the type of RDS group being transmitted |

### Detailed Bit Layout

Here's a more detailed layout of the bit structure of Block 2:

* **Bits 0-3: Group Type Code**
  + Identifies the type of RDS group (e.g., 0 for basic tuning and switching information, 1 for program item number and slow labeling codes).
* **Bit 4: Version Code**
  + Indicates the version of the RDS group (0 = version A, 1 = version B).
* **Bits 5-9: Program Type (PTY)**
  + Specifies the type of content being broadcast. This is a 5-bit field allowing for 32 different types (0-31). The PTY codes are defined in the PTY table provided earlier.
* **Bit 10: Traffic Program (TP)**
  + A single bit flag indicating whether the station regularly broadcasts traffic information. If set to 1, the station provides traffic information.
* **Bits 11-15: Traffic Announcement (TA)**
  + A flag indicating whether a traffic announcement is currently being broadcast. If set to 1, a traffic announcement is currently on air.
* **Bits 16-31: Other Data**
  + This section contains additional data that varies depending on the type of RDS group. For example, it may include radio text, alternative frequencies, or program service names.

**Example**

For a hypothetical RDS group where the Program Type is set to "News" (PTY code 1), the Traffic Program flag is set (TP = 1), and a traffic announcement is currently being broadcast (TA = 1), the bit structure of Block 2 might look like this:

|  |  |  |
| --- | --- | --- |
| **Bits 0-3** | **0000** | **Group Type 0A** |
| **Bit 4** | **0** | **Version A** |
| **Bits 5-9** | **00001** | **PTY code for News** |
| **Bit 10** | **1** | **TP flag set** |
| **Bits 11-15** | **11111** | **TA flag set** |
| **Bits 16-31** | **XXXX** | **Other data specific to group type 0A** |

**In binary, Block 2 would be: 0000 0 00001 1 11111 (followed by other data)**

### **4. Station Name Transmission (PS)**

The main objective is to transmit the station name. Groups 0A and 0B are the most basic groups for this purpose. For transmitting the station name, group 0B is used, as block 3 of group 0B simply repeats the PI code with a different offset word, C'.

**Group Type 0A and 0B**

For transmitting the Program Service name, the group type 0A or 0B is used. Here, we focus on Group Type 0B as it is often used for transmitting the station name.

**Structure of Group Type 0B**

Each group type 0B consists of four blocks, with each block containing 16 bits. To transmit the entire Program Service name (8 characters), multiple groups are needed.

**Group 0B Structure**

|  |  |
| --- | --- |
| **Block 1** | **Program Identification (PI) Code** |
| **Block 2** | **Contains Group Type Code, Version Code, PTY, TP, and TA** |
| **Block 3** | **Contains the Program Identification (PI) Code (repeated)** |
| **Block 4** | **Contains the characters of the Program Service name** |

Each group type 0B can transmit two characters of the PS name in Block 4. Hence, to transmit an 8-character PS name, four consecutive group type 0B messages are required.

**Detailed Bit Structure**

|  |  |  |
| --- | --- | --- |
| **Block 1** |  | **PI Code** |
|  | **Bits 0-15** | **PI Code (Program Identification)** |
|  |  |  |
| **Block 2** |  | **Group Type, PTY, TP, TA** |
|  | **Bits 0-3** | **Group Type Code (0000 for Group Type 0)** |
|  | **Bit 4** | **Version Code (1 for version B)** |
|  | **Bits 5-9** | **Program Type (PTY)** |
|  | **Bit 10** | **Traffic Program (TP) flag** |
|  | **Bits 11-15** | **Traffic Announcement (TA) flag and additional information** |
|  |  |  |
| **Block 3** |  | **Repeated PI Code** |
|  | **Bits 0-15** | **PI Code (repeated)** |
|  |  |  |
| **Block 4** |  | **Program Service Name Characters** |
|  | **Bits 0-7** | **First character of the PS name** |
|  | **Bits 8-15** | **Second character of the PS name** |

**Example Transmission**

Suppose the station name is "RADIO123". Each character is represented in ASCII.

**ASCII Representation**

**'R' = 82**

**'A' = 65**

**'D' = 68**

**'I' = 73**

**'O' = 79**

**'1' = 49**

**'2' = 50**

**'3' = 51**

**Group Type 0B Messages**

**First Group 0B**

|  |  |
| --- | --- |
| **Block 1** | **PI Code** |
| **Block 2** | **Group Type Code (0000), Version Code (1), PTY, TP, TA** |
| **Block 3** | **PI Code (repeated)** |
| **Block 4** | **'R' (82), 'A' (65)** |

**Second Group 0B**

|  |  |
| --- | --- |
| **Block 1** | **PI Code** |
| **Block 2** | **Group Type Code (0000), Version Code (1), PTY, TP, TA** |
| **Block 3** | **PI Code (repeated)** |
| **Block 4** | **'D' (68), 'I' (73)** |

**Third Group 0B**

|  |  |
| --- | --- |
| **Block 1** | **PI Code** |
| **Block 2** | **Group Type Code (0000), Version Code (1), PTY, TP, TA** |
| **Block 3** | **PI Code (repeated)** |
| **Block 4** | **'O' (79), '1' (49)** |

**Fourth Group 0B**

|  |  |
| --- | --- |
| **Block 1** | **PI Code** |
| **Block 2** | **Group Type Code (0000), Version Code (1), PTY, TP, TA** |
| **Block 3** | **PI Code (repeated)** |
| **Block 4** | **'2' (50), '3' (51)** |

**Bit Stream Example**

**For the first group:**

**Block 1: PI Code**

**Block 2: 0000 1 00001 1 00001 (Group Type 0B, Version B, PTY, TP, TA)**

**Block 3: PI Code (repeated)**

**Block 4: 01010010 01000001 ('R' in bits 01010010, 'A' in bits 01000001)**

**Each subsequent group would follow the same structure, with Block 4 containing the next pair of characters in the station name.**

This structure ensures that the Program Service name is correctly transmitted and displayed on RDS-compatible receivers, providing the listener with the station's name.

### **5. Transmission of the Program Service Name (PS)**

A total of four type 0B groups are required to transmit the entire Program Service (PS) name, which comprises eight characters. Therefore, four type 0B groups are needed per second. **The PS name is used only to identify the station and should not be scrolled or altered in a distracting manner (not more frequently than once per minute).**

## Summarizing and Example

Each RDS group consists of four blocks, each with 26 bits. These blocks contain the following fields:

**1. Block 1: Contains the PI code.**

**2. Block 2: Contains the group application bits, PTY code, and TP bit.**

**3. Block 3: Depends on the group type. In group 0B, it contains the repeated PI code.**

**4. Block 4: Contains the service data (such as the station name PS).**

**Packaging Example**

To transmit the station name "RADIO123" using group 0B, you need to split the name into parts and package it in 0B groups. Assuming the PI code is 0x1234, the PTY code is 0x01 (e.g., news), and TP = 0.

**uint16\_t PI = 0x1234;**

**uint8\_t PTY = 0x01;**

**uint8\_t TP = 0;**

**char PS[] = "RADIO123";**

**// Function to calculate the checksum for each block**

**uint16\_t calcChecksum(uint16\_t block) {**

**// Checksum calculation implementation**

**}**

**// Function to package and send the 0B group**

**void sendRDSGroup0B(char \*psName) {**

**for (int i = 0; i < 4; i++) {**

**uint16\_t block1 = PI;**

**uint16\_t block2 = (PTY << 5) | (TP << 4) | (0 << 3) | (0 << 2) | (0 << 1) | 1; // Version B**

**uint16\_t block3 = PI; // PI code repetition**

**uint16\_t block4 = (psName[i \* 2] << 8) | psName[i \* 2 + 1];**

**block1 = (block1 << 10) | calcChecksum(block1);**

**block2 = (block2 << 10) | calcChecksum(block2);**

**block3 = (block3 << 10) | calcChecksum(block3);**

**block4 = (block4 << 10) | calcChecksum(block4);**

**// Send data blocks via FM transmitter**

**sendBlock(block1);**

**sendBlock(block2);**

**sendBlock(block3);**

**sendBlock(block4);**

**}**

**}**

**void sendBlock(uint16\_t block) {**

**// Function implementation to send a data block**

**}**

## Final Considerations

1. Initial Configuration: Set the constant values of the group according to the RDS standard.

2. PS Name Input: The user only needs to enter the PS name, and the system will handle the packaging and sending of the data according to the RDS standard.

3. Checksum Calculation: Implement the `calcChecksum` function to ensure data integrity during transmission.

This approach should enable the correct transmission of the station name using the RDS protocol.