SILICON LABS

AN592

Data Whitening Demo Code

1. Introduction

Applications using radio communication typically transmit data, which can, from a radio IC point of view, be considered as random data flow with logic 1s and 0s alternating in a random manner. Such data ensures smooth power distribution over the occupied bandwidth and, additionally, may improve the quality of the RF link due to the nature of the receiver's clock recovery circuitry.

In order to get the transmitted data, the receiver has to re-create the clock of the transmitter. This is primarily done during the beginning of each transmission when preamble bytes (continuous 1s and 0s) are sent. Once they are recognized by the receiver, it is ready to process the incoming data. Ideally, the clocks on both sides are stable and do not change during the rest of the communication; even long strings of 1s and 0s can be sent.

Sometimes, however, the clock frequency on the transmitter side may change slightly during transmission (e.g., when no external crystal oscillator is used to create the modulated data in order to keep the cost low). Also, a frequency drift may exist between the external crystals. To have the same clock frequency on both sides, the receiver continuously fine-tunes its re-created data clock. The more often logic ones and zeroes alternate, the better the fine-tuning. However, if the data to be transmitted contains long strings of logic 1s and 0s, the fine tuning may not be sufficient, which may cause bit errors in the received data. To avoid this, a data whitening technique can be used. This algorithm "whitens" the data in such a way that the 1s and 0s will be better balanced.

Although Silicon Labs' EZRadioPRO family has a built-in data whitening feature implemented by hardware, users may wish to implement a similar technique via software to make the application compatible with other manufacturers' products. A common data whitening method is described in the following sections, and a demo code for the EZRadioPRO platform is provided.

2. Data Whitening

The key of the implemented data whitening method is to XOR (exclusive-or) the data and a continuously-changing pseudo-random number. The “whitened” data can then be sent over the air; running the same whitening procedure again, the original data can be retrieved, i.e. “de-whitened”. For each data byte, a new whitening key is generated. The algorithm is described by the polynomial $x^9+x^5+x^0$.

Figure 1 shows the hardware representation of the method. The whitening key is a 9-bit pseudo-random number between 1 and 511. An initial value must be entered, and then a new whitening key is generated by shifting and XOR-ing. Logically, the whitening key can be divided into two parts: a carry bit (MSB) and the actual key (eight LSBs).

Whitening the data, i.e. XOR-ing data and the key, is done byte-by-byte; a new whitening key is generated for each data byte.

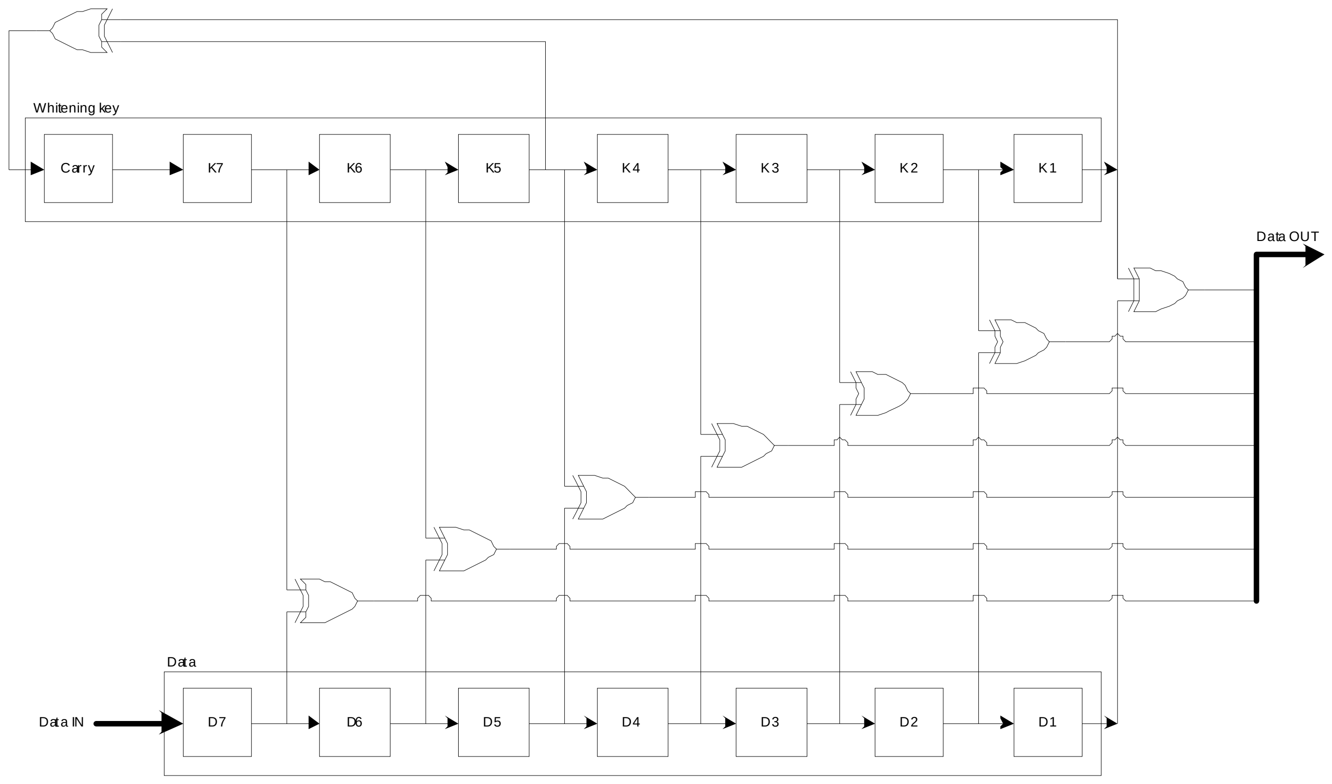

Figure 1. Whitening Method Hardware Representation

3. Example

Taking an example, the initial value of the whitening key is set to all ones (1 1111 1111), the data bytes to be sent are the following: 0x0F, 0x00, 0x01, 0x02,…, 0x0C, 0x0D. The first data byte is XOR-ed with the actual key (eight LSBs of the 9-bit whitening key) resulting in the first “whitened” data byte, which is 0xF0 as shown by Table 1. Then, a new whitening key is generated (1 1110 0001), which is to be used for generating the second “whitened” data byte, and so on.

Table 1. Data after Whitening*

Whitening key		Data	Whitened data	De-whitened data
1	1111 1111	0x0F	0xF0	0x0F
1	1110 0001	0x00	0xE1	0x00
0	0001 1101	0x01	0x1C	0x01
1	1001 1010	0x02	0x98	0x02
	…	…	…	…

***Note:** Running the whitening algorithm on the whitened data results in de-whitening the data.

Once all the bytes of the data base are whitened with a key, logic 1s and 0s are better balanced and are ready to be transmitted via the RF link. When it is available on the receiver side, the same whitening method is to be applied to get the original data. Note that on both the receiver and the transmitter sides, the same initial whitening key must be used.

4. Firmware Implementation

The firmware is a modification of the standard Si443x demo code and can be run on Silicon Labs Software Development Board and EZRadioPRO test card. When pushing PB1 on the transmitter, it transmits the whitened data. When received, it is indicated by an LED blinking once the data is de-whitened and checked.

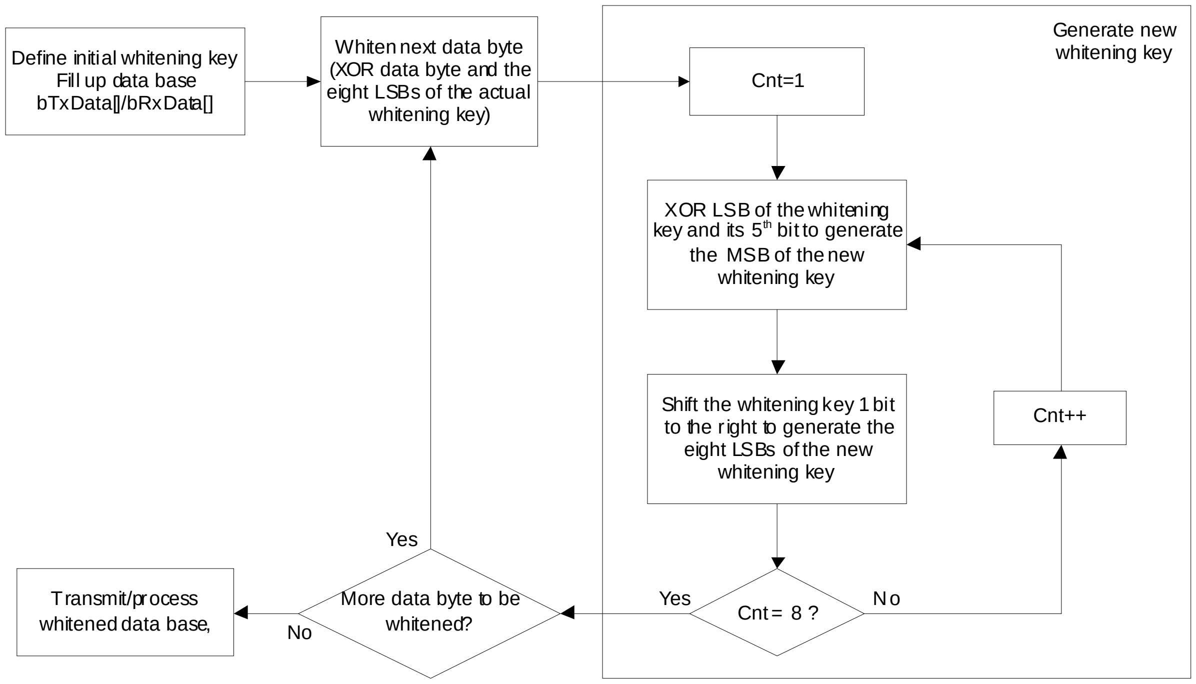
Define initial whitening key
Fill up data base
bTxData[]/bRxData[]
Whiten next data byte (XOR data byte and the eight LSBs of the actual whitening key)
Generate new whitening key
Cnt=1
XOR LSB of the whitening key and its 5th bit to generate the MSB of the new whitening key
Shift the whitening key 1 bit to the right to generate the eight LSBs of the new whitening key
Cnt++
Yes
Transmit/process whitened data base,
More data byte to be whitened?
No
Yes
Cnt= 8 ?
No

Figure 2. Software Implementation of the Data Whitening

Figure 2 shows the software implementation of the whitening method. After defining the initial value of the whitening key and filling the `bTxData[]`, or `bRxData[]` array with data, the whitening is done byte-by-byte; for each data byte, a new whitening key is generated. Note that for both the transmitter and the receiver, the same initial whitening key must be used.

In the code, the length of the packet (`TX_DATA_LENGTH/ RX_DATA_LENGTH`) and the initial value of the whitening key (`DATA_WHITENING_KEY`) can be set by a `#define`, at the beginning of the code. Then, either `bTxData[]` array is filled up with data bytes in function `vTxDataFillUp()` or on the receiver side, `bRxData[]` is filled up by reading the FIFO register of the radio. This is followed by a call to `vWht_DataWhitening()`; it whitens the corresponding data base by overwriting it. For each data byte to be whitened, a new key is generated by the function, `vWht_GetNewKey()`.

NOTES:

Simplicity Studio

One-click access to MCU tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

www.silabs.com/simplicity

MCU Portfolio
www.silabs.com/mcu

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com