Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN2373

Ranking Original Pseudo-Noise Codes In A DSSS System

Author: Jeffrey McLarty

Associated Project: Yes

Associated Part Family: WirelessUSB

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN2373.

If a WirelessUSB™ or any direct-sequence spread spectrum (DSSS) system is to be used with unique pseudo-noise (PN) codes, an assessment should be done of the code's auto-correlation properties. This Application Note explains about assessing the quality of a given PN code relative to another, because it is important in choosing a code appropriate for an application.

1 Introduction

An algorithm was developed to rank PN codes by assigning a score based on the length of the groups of overlapping sections of the code. The algorithm is too inefficient to use to find the optimal PN codes. However, optimal PN codes are not necessary, but good PN codes are. Range testing results prove the ranked PN codes out performed each other by the average number of retries needed for successful communication. Therefore, the ranking of the codes should suffice in order to determine that any PN codes chosen are acceptable for use. The codes provided by Cypress were determined not to be the optimal found by the algorithm. This is because the criteria that the algorithm assesses do not include cross correlation properties. This rationale postulates that the list provided by Cypress is a good starting point to generate original PN codes, if needed.

Some PN codes are better than others for certain applications, although in general, any code used must have low numbers of lengthy repeated patterns when compared to itself shifted left or right by any amount. This shifting property is known as the auto-correlation factor. Any new codes generated for use should be assessed for this factor. To make this assessment, a utility has been provided that outputs a score for a list of PN codes.

To understand the reasoning behind the algorithm pseudo code, the reader should take the opportunity to read the associated application notes prior to proceeding.

2 Pseudo-Noise Codes

2.1 The Algorithm

Any WirelessUSB™ application can use the gold codes provided by Cypress, modify the codes provided with this note, or completely generate new codes. In any case, ranking the created or modified PN codes for the auto-correlation factor is the first step. An algorithm was written to do this. It is downloadable with this Application Note. Basically, the input is a list of PN codes, and the algorithm assigns a score for the auto-correlation factor.

2.2 Pseudo Code Ranking Algorithm for accessing Pseudo Codes

To avoid confusion, it should be mentioned that Code 1 is pseudo code of an algorithm to assess pseudo codes. That is, two entirely different definitions of "pseudo code."

An engineer needs a mathematical way to analyze the strength of a given code with respect to this issue. Code 1 shows the pseudo code for the algorithm provided by the utility.

Code 1. Algorithm Pseudo Code

```
NumberOfOnes, Score = 0;
PNCodeResult = 0;
PNCode = Generate or Input a PN Code;
PNCodeCopy = PNCode;
For (every bit in PN Code)
```

```
// 64 for an 8 byte PN Code
{
// Roll the copy left by one chip
PNCodeCopy = PNCodeCopy x 2;

// Check for overlap
PNCodeResult =
~(PNCodeCopy XOR PNCode);
For (Every group of "One"s)
{
// Large groups of ones are worse
// than inevitable small groups
// of ones so count the length of
// the "ones" grouping.
NumberOfOnes =
"One"s grouping length;
// then add it to the cumulative
// score by ranking it harder if
// it is larger
Score = Score + (NumberOfOnes * NumberOfOnes)/2;
}
}
return Score;
```

Each algorithm iteration rolls a PN code left by one bit, and then complements the result of the exclusive OR of the rolled and the original. An example of a 20-chip PN code iteration is shown in Figure 1. Groups of ones are counted. The number of groups and their length are measured. Larger groupings of ones add more to a cumulative score.

Figure 1 shows the end of one iteration and the start of another, up to the point where the PN code copy is rolled again. Analyzing a 20-chip PN code, it would take 20 such iterations. The algorithm runs once per PN code. However, for every bit in the length of the PN code, the algorithm loops once to roll, XOR and complement a result. Inside that loop another loop runs for the length of the PN code in order to count the number of groups of ones and their length. That means if 'n' represents the length of the PN code in chips, one iteration of the algorithm approaches $O(n^2)$.

Figure 1. One Iteration of a 20-Chip PN Code

1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	1	0	1	1	Original
1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	Roll Left Once
0	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	0	0	XOR
1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1	Complement
1					2								3					2		Length of Groups of Ones
0.5				+	2						+		4.5				+	2		Squared and Half
=	9																			Add to Cumulative Score
0	1	0	1	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	1	Repeat by Rolling Original Left Twice

2.3 Timing of the Algorithm

Due to the complement effect of transmitting a true or false bit, the unique scores are essentially halved. Therefore, of the 2^{64} possible PN codes, only half are going to have unique scores. Therefore, the complements do not have to be scanned, leaving only 2^{63} codes to rank. However, time is a limiting problem. The algorithm compiled in Visual C++ running on a 2.4 GHz Intel Pentium 4 running Windows 2000 with 512 MB of RAM took 12 hours to scan approximately 2^{31} PN codes that were each 8 bytes long. To scan 2^{63} numbers at this rate, it would take over 5.8 million years. Furthermore, the results would need to be sorted for use. So, the algorithm was used along with human intervention to find and test only needed and interesting PN codes.

2.4 Algorithm Output

The algorithm generates scores for every PN code. Table 1 shows a list of interesting and expected results. These results underscore the issue of unnecessary analysis of complement PN codes. These are scores for the auto-correlation factor, so higher scores are worse.

Table 1. Complement PN Code Score Comparison

PN Code	Score
FF FF FF FF FF FF FF FC	79580
FF FF FF FF FF FF FF FD	83423
FF FF FF FF FF FF FF FE	83423
FF FF FF FF FF FF FF FF	131072
00 00 00 00 00 00 00 00	131072
00 00 00 00 00 00 00 01	83423
00 00 00 00 00 00 00 02	83423
00 00 00 00 00 00 00 03	79580

By inspection, 0x00 00 00 00 00 00 00 00 is the worse case, and it should therefore be the highest scoring PN code, as Table 1 clearly shows. Table 2 shows a PN code, identical to a similar code except shifted left by one chip. As expected, they have the same correlation properties, thus, the scores are the same.

Table 2. Complement PN Code Score Comparison

PN Code	Score
AA AA AA AA AA AA AA AA	65536
55 55 55 55 55 55 55 55	65536

WirelessUSB is distributed by Cypress and comes with 57 gold PN codes plus one pre-defined code in the transceiver, for a total of 58. Table 3 starts with a single PN code taken from the list provided by Cypress Semiconductor. The least significant byte was incremented with every entry in the table and a new matching score was found. The bolded entries are scores that were found to be lower than the one provided by Cypress.

Table 3. Scores Found by Starting with Cypress PN Code

PN Code	Score
D7 A1 54 B1 5E 89 AE 86	5154
D7 A1 54 B1 5E 89 AE 87	5117
D7 A1 54 B1 5E 89 AE 88	5205
D7 A1 54 B1 5E 89 AE 89	5214
D7 A1 54 B1 5E 89 AE 8A	5358
D7 A1 54 B1 5E 89 AE 8B	5281
D7 A1 54 B1 5E 89 AE 8C	5144
D7 A1 54 B1 5E 89 AE 8D	5167
D7 A1 54 B1 5E 89 AE 8E	5153
D7 A1 54 B1 5E 89 AE 8F	5130
D7 A1 54 B1 5E 89 AE 90	5147
D7 A1 54 B1 5E 89 AE 91	5142

Before the availability of the ranking algorithm, a subset of the WirelessUSB gold codes was chosen at random to be used for a confidential application. This subset was altered minutely by switching the most significant nibble and the least significant nibble in a particular byte. Table 4 shows the comparison of the five original and five altered PN codes. As can be quickly observed, randomly changing one byte produced randomly higher and lower ranking PN codes.

Table 4. Scores of Original and Minimally Modified PN Codes

Gold Code Scores	Modified Confidential PN Code Scores
4916	4954
4806	4940
5678	5560
5250	5348
5418	5330

3 Range Testing

When the algorithm assigns a score to a given PN code, it can be assessed by inspection as to why certain codes are scoring higher than others. However, it is necessary to prove that the rankings actually do correlate to better performance. To do so, the codes listed in Table 5 were installed in the WirelessUSB Development Kit with the debugging features enabled.

Table 5. PN Codes Selected for Range Testing

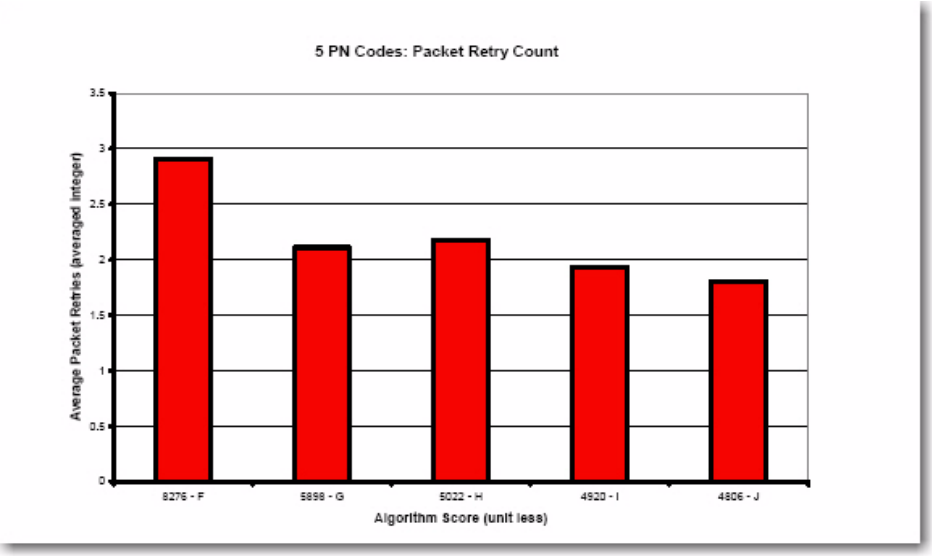
	PN Code	Score	Notes
A	00 00 00 00 00 00 00 00	131072	Worst Possible Case
B	00 00 00 00 00 00 00 07	75863	
C	55 55 55 55 55 55 55 55	65536	
D	01 01 01 01 01 01 01 01	22208	
E	01 01 01 01 80 80 80 80	13502	
F	E1 D6 3F FF FF BD 40 93	8276	
G	80 69 26 80 08 F8 49 E7	5898	Higher WirelessUSB
H	82 C7 90 36 21 9E FF 17	5022	Random WirelessUSB
I	## ## ## ## ## ## ## ##	4920	Confidential PN Code
J	93 40 BD 5F 26 31 D6 E1	4806	Lower WirelessUSB

With serial output logged on both a transmitter and receiver, ten packets were attempted to be sent at various distances. In Table 5, PN codes A through D did not even bind; as a result the test for them could not be completed. PN Code E did bind however; it needed 18 attempts to do so. This was expected and proves that a PN code scoring above 10 000 should not be used.

For the remaining codes from Table 5, F through J, the number of retries needed for ten packets to be transmitted correctly were counted at equal distances then averaged and summarized in Figure 2.

The number of retries seen in Figure 2 demonstrates that there is a correlation between the algorithm's score of a PN code and the quality of the code in the application.

Figure 2. Algorithm Scores Compared to Performance



3.1 Sources of Uncertainty and Error

The results of the range test were needed to assess the validity of the algorithm's scores at a qualitative level. A quantitative relationship between score and reliability is not required. Because of this, during range testing, certain issues were not controlled, and should be noted. These issues include:

1. Environmental interference existed as a nearby 802.11b wireless network operated near the same frequency range. However, the channel-scanning algorithm implemented with WirelessUSB largely compensated for this possible interference source.
2. Channel selection was not constant. The protocol continuously searched for the quietest channel in which to operate. Therefore, every PN code was tested on the quietest channel available at the time. The channel selection algorithm in use was the one distributed with the WirelessUSB Development Kit.
3. Device and protocol latency can affect the retry count. Latency affects all PN codes, almost randomly; as a result, it was left uncontrolled and unmonitored.

4 Summary

The results of the range testing suggest that the algorithm was successful in choosing PN codes that are preferred over others for use in a DSSS system.

Since the PN codes chosen by Cypress ranked with good scores, they were not absolutely the best found. This leads to hypothetical conclusions regarding the source of the 58 gold codes. The gold codes are not the absolute highest ranking with the auto-correlation factor so sacrificing auto-correlation factor quality must have happened in order to gain on other factors such as the cross correlation. This means Cypress may have sacrificed the best codes in order to obtain extremely unique codes. Whichever way they were generated, the codes sold by Cypress do provide an excellent foundation for a DSSS system and should be used if resources and time are limited.

If fewer PN codes are needed in an application, the algorithm affirmatively demonstrates that better codes, in terms of auto correlation, than provided by Cypress can be found if the algorithm is given enough time or improved.

5 Recommendations

Based on the level of knowledge needed to understand the application of PN codes, it is assumed the reader is comfortable enough to write a simple program to generate new codes. A few simple loops can easily generate a new list to scan. It is advised to generate new PN codes using software (rather than building test hardware).

PN codes included with the development kit make a good starting point to generate more codes if needed. Many methods of mutation are easily possible. For instance, swapping the most significant nibble with the least significant nibble in two random bytes between 1 and 8, could work to generate more codes to start from.

Modifying the gold codes provided based on the auto-correlation assessing algorithm score in excess could sacrifice quality in terms of other factors not assessed by the algorithm. Caution should be used, if doing so.

Since the algorithm does not test every deciding factor of a PN code, the ranking is only an assessment of the phase overlap factor it tests. The results of this algorithm should be used along with other considerations and potentially other algorithms, which are application dependent.

The ranking algorithm score can manage to rank PN codes relative to each other. Some similar method should be used if any spread spectrum technology user is choosing brand new PN codes. To use the algorithm to find the optimal PN codes is interesting, however, unnecessary to run completely since a difference in a score of less than 250 is barely noticeable in the resulting system. Therefore, any user randomly choosing 1-byte values along with common sense of not picking obvious repeating patterns is most likely a good seed for an adequate code with a modest score.

Please see the attached utility, *readme.txt*, and *PNCodes.txt*.

About the Author

Name: Jeffrey McLarty.

Document History

Document Title: AN2373 - Ranking Original Pseudo-Noise Codes in a DSSS System

Document Number: 001-21520

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1412604	ZHC	08/27/2007	New Application Note.
*A	3355460	ZHC	08/26/2011	Updated Abstract. Added Document History.
*B	4504361	LIP	09/16/2014	Updated to new template. Completing Sunset Review.
*C	5879575	AESATMP8	09/20/2017	Updated logo and Copyright.
*D	5900083	CHYY	09/28/2017	No technical updates. Completing Sunset Review.
*E	5997849	SEG	12/18/2017	Updated author information Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.