# Using the TLK2711-SP With Minimal Protocol

*Wade Vonbergen*

**ABSTRACT**

The TLK2711-SP can be implemented using a minimal protocol with various levels of additional complexity for added features. This can run the range of streaming raw data with occasional comma character for transmission to packetizing data with checksum transmission and link level commands.

**Contents**



**List of Figures**



**List of Tables**

## 1 Introduction

The TLK2711-SP is a 1.6-Gbps to 2.5-Gbps Class V SERDES transceiver capable of approximately 1.28-Gbps to 2-Gbps of data payload. It has a 16-bit parallel interface with built in 8B/10B encoding.

In order to successfully implement a TLK2711 transceiver, one must have a working knowledge of 8B/10B encoding.

## 2 8B/10B Primer:

8B/10B is an encoding scheme that takes an 8-bit word, and converts it to a 10-bit value for transmission. The encoding scheme was developed by IBM, and has several advantages over transmitting raw data without encoding.

1. DC balance. The running disparity will always be bounded between +2 and -2. The running disparity is defined as the difference in the number of 1 s and 0 s transmitted.
2. Maximum run length is bounded. The encoding scheme ensures that no more than 5 consecutive 1 s or 0 s will occur. This insures that a minimum transition density exists for proper clock recovery.
3. Defined control codes, and specifically the "comma" character to align 8-bit boundary at receiver. The comma character is contained in a K28.5, that gets decoded to a (0011111)010 or 1100000101. The () indicates the comma value that the TLK2711 needs for alignment.

8B/10B uses tables to encode the transmitted word, and to decode it at the receiver. It actually uses two sets of codes. 8B/10B is defined by a 5B/6B and a 3B/4B code. The common convention used to describe the 8B/10B values is to utilize upper case for 8B, and lower case for the 10B. For example an 8-bit word is defined as ABCDEFGH with A being the least significant bit. This gets broken down as EDCBA HGF. This is commonly referred to as Dx.y. Where X is 0-31, and Y is 0-7 and represents the 5b and 3b bit values. An 8-bit value of 0xBC breaks down as 101 11100 and is D28.5. Encoding tables for 5B/6B and 3B/4B shown below.

A full table of all 8B/10B values is published within the IEEE 802.3.2000 specification. Within the tables, many of the 5B, and 3B codes have multiple encodings. The appropriate coding is selected based on the current running disparity.

**Table 1. 5B/6B Table Lookup**

| **INPUT** | | **RD = -1** | **RD = +1** | | **INPUT** | | **RD = -1** | **RD = +1** |
|---|---|---|---|---|---|---|---|---|
| | EDCBA | abcdei | | | | EDCBA | abcdei | |
| D.00 | 00000 | 100111 | 011000 | | D.16 | 10000 | 011011 | 100100 |
| D.01 | 00001 | 011101 | 100010 | | D.17 | 10001 | 100011 | |
| D.02 | 00010 | 101101 | 010010 | | D.18 | 10010 | 010011 | |
| D.03 | 00011 | 110001 | | | D.19 | 10011 | 110010 | |
| D.04 | 00100 | 110101 | 001010 | | D.20 | 10100 | 001011 | |
| D.05 | 00101 | 101001 | | | D.21 | 10101 | 101010 | |
| D.06 | 00110 | 011001 | | | D.22 | 10110 | 011010 | |
| D.07 | 00111 | 111000 | 000111 | | D.23 | 10111 | 111010 | 000101 |
| D.08 | 01000 | 111001 | 000110 | | D.24 | 11000 | 110011 | 001100 |
| D.09 | 01001 | 100101 | | | D.25 | 11001 | 100110 | |
| D.10 | 01010 | 010101 | | | D.26 | 11010 | 010110 | |
| D.11 | 01011 | 110100 | | | D.27 | 11011 | 110110 | 001001 |
| D.12 | 01100 | 001101 | | | D.28 | 11100 | 001110 | |
| D.13 | 01101 | 101100 | | | D.29 | 11101 | 101110 | 010001 |
| D.14 | 01110 | 011100 | | | D.30 | 11110 | 011110 | 100001 |
| D.15 | 01111 | 010111 | 101000 | | D.31 | 11111 | 101011 | 010100 |

**Table 2. 3B/4B Table Lookup**

| INPUT | | RD = -1 | RD = +1 |
|---|---|---|---|
| | HGF | fghj | |
| D.x.0 | 000 | 1011 | 0100 |
| D.x.1 | 001 | 1001 | |
| D.x.2 | 010 | 0101 | |
| D.x.3 | 011 | 1100 | 0011 |
| D.x.4 | 100 | 1101 | 0010 |
| D.x.5 | 101 | 1010 | |
| D.x.6 | 110 | 0110 | |
| D.x.P7 | 111 | 1110 | 0001 |
| D.x.A7 | 111 | 0111 | 1000 |

**Table 3. Control Code Lookup**

| INPUT | | RD = -1 | RD = +1 |
|---|---|---|---|
| | HGF EDCBA | abcdei fghj | abcdei fghj |
| K.28.0 | 000 11100 | 001111 0100 | 110000 1011 |
| K.28.1 | 001 11100 | 001111 1001 | 110000 0110 |
| K.28.2 | 010 11100 | 001111 0101 | 110000 1010 |
| K.28.3 | 011 11100 | 001111 0011 | 110000 1100 |
| K.28.4 | 100 11100 | 001111 0010 | 110000 1101 |
| K.28.5 | 101 11100 | 001111 1010 | 110000 0101 |
| K.28.6 | 110 11100 | 001111 0110 | 110000 1001 |
| K.28.7 | 111 11100 | 001111 1000 | 110000 0111 |
| K.23.7 | 111 10111 | 111010 1000 | 000101 0111 |
| K.27.7 | 111 11011 | 110110 1000 | 001001 0111 |
| K.29.7 | 111 11101 | 101110 1000 | 010001 0111 |
| K.30.7 | 111 11110 | 011110 1000 | 100001 0111 |

Within the 8B/10B encoding, there are special symbols defined for control information. These are commonly referred to as K characters. These characters are used for higher level control such as start of frame (SOF), end of frame (EOF), retransmit, errors, etc. The single most important K character is the comma. The TLK2711 uses the comma for byte alignment of the parallel data at the receiver. The comma is contained in multiple K characters, but for practical purposes it is defined as the K28.5. From Table 3, K28.5 is decoded as a 001111 1010 or a 1100000101 depending on current running disparity. Note, the TLK2711 only achieves alignment on the 001111 1010 comma. Specifically it is looking for the 0011111. This has a significant implication. This implies that the user must send the 0011111 comma. However, a single K28.5 may not generate this comma, as the current running disparity is not deterministic. This is solved by sending two inverting (or correcting) idles back to back. An inverting IDLE is two words that taken as a whole will cause the running disparity to flip. So two back-to-back inverting idles will ensure that one of them will be the correct decoding of the K28.5. The inverting idle is defined as a /K28.5/D5.6/. There are other data values that will cause an inversion when paired with the K28.5. However, this symbol pair is the standard.

## 3 Quick Experiment to Obtain Byte-Alignment With TLK2711-SP EVM

This is a trivial example, to help visually explain the minimum requirements to achieve byte-alignment utilizing the TLK2711-SP EVM. With TXCLK being driven at a clock frequency within the operational range of 80 Mhz to 125 Mhz, the comma character can be sent by configuring jumper pins on the TXD interface.

Start with a configuration that is only sending data with internal loopback.

The jumper settings show a D28.5/D5.6 being transmitted on the TX bus. Note, that this does not include the comma, as it is coded as data with TKLSB set to low.

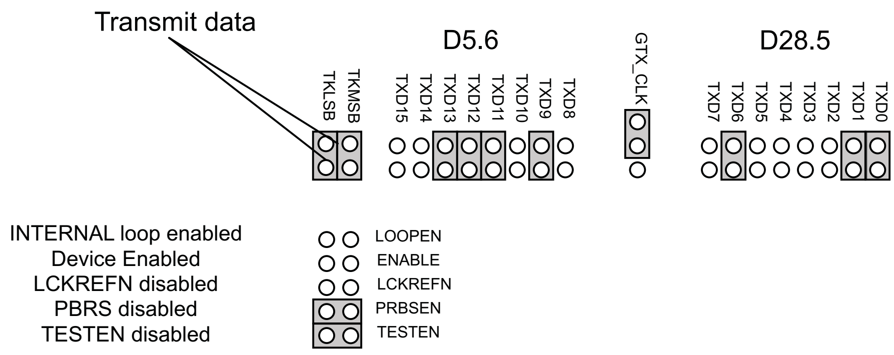


**Figure 1. EVM Jumper Settings**

Internal LOOPEN may be disabled (add jumper), if external loopback is used. Connect TXP to RXP and TXN to RXN via SMA cables. With LOOPEN enabled, the transmitted signal is routed internally to receiver.

Next, supply power and TXCLK to board.

The user can now probe the received signals on the RXD buss. When probing signals, notice that the data does not represent what is on the transmit buss. It is possible, but unlikely that device will happen to align to proper boundary.

Now, remove and replace the TKLSB jumper.

This will cause many inverting idles to be sent on the buss, and will give the TLK2711 the necessary comma character to achieve byte alignment. Now probe the RXD bus. Signals should now match the D28.5/D5.6 mirroring the TXD buss.

## 4 Defining a Simple Protocol

The simplest protocol would be transmitting raw data, with a periodic transmission of 2 inverting idles, then resume transmitting raw data. A step beyond this would be the inclusion of framing or packetizing the data. Sending periodic commas in the stream is necessary, as an error in the bit stream could cause false byte-alignment on receiver.

A simple and effective protocol can be implemented as follows.

- K28.5 D5.6
- K28.5 D5.6
- K28.5 D11.5 (SOF optional and user definable)
- Dxx Dxx (Transmitted packet. Repeat for packet depth.)
- Dxx Dxx (Packet CRC if desired)
- K28.5 D5.6
- K28.5 D5.6
- K28.5 D11.5 (SOF optional)

Repeat with next packet.

Note, that a packet begins with 2 inverting idles. This will achieve byte alignment on the receiver, and provide for re-alignment if a bit error causes a mis-alignment at some point during operation.

The (SOF) sequence is not necessary, but can be used by receiving link layer to manage capturing the data.

The use of a CRC is also optional, but depending on source of the data, this is a convenient means to detect if an error occurred in the data. A more elaborate scheme could include a bidirectional control path. On receipt of a data error, the receiving link could transmit back to request a retransmit of the erroneous packet. This is beyond the scope of this document.

## 5 References:

A.X. Widmer, Peter A. Franaszek (1983). "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code" *IBM Journal of Research and Development* **27**(5): 440.

Texas Instruments TLK2711-SP Data sheet (SGLS307D)

Texas Instruments TLK2711-SP EVM manual (SGLU001)