GSM: A PHY Layer Implementation

Karol Wadolowski

***Abstract*—This document serves to explain the physical layer implementation of several parts of the Global System for Mobile (GSM) standard. The implementation was done in MATLAB version 2019b. The implementation consists of two main parts. The first part is the control channel and the second is one of the full rate data channels. The performance of the data channel is also observed.**

***Index Terms*—GSM, TDMA, GMSK**

I. Introduction

This report goes over portions of the GSM standard, one of the leading 2G standards in Europe. This includes data structures and communication channels. The structure of GSM will be covered first, followed by an explanation of the various communication channels. Then two of the communication channels will be explored in more detail followed by an explanation of modulation and what happens at the receiver.

II. Data Structure

GSM consists of a five layered structure. The lowest layer is a single burst. A single burst consists of 156.25 bits and lasts $\frac{120}{208}\ ms \approx 577\ \mu s$. For purposes of MATLAB implementation each frame will consist of 156 bits. There are multiple different types of bursts that can be sent that will be explored in Sec. II-A. The next level is called a TDMA (time division multiple access) frame. A frame consists of 8 bursts and lasts $\frac{120}{26}\ ms$ $\approx 4.615\ ms$. The following layer is called the multiframe. The multiframe is in most cases made up of either 26 or 51 frames. The size of the multiframe depends on the purpose of the channel. The 9.6 $kbit/s$ full rate data traffic channel (TCH/F9.6) uses 26 TDMA frames per multiframe while the control channel (CCH) uses a 51 TDMA multiframe. The 26-frame multiframe lasts 120 ms and the 51-frame multiframe lasts $\frac{3060}{13}\ ms$. The second highest layer is the superframe. The superframe consists of 1,326 TDMA frames which is 51 26-frame multiframes or 26 51-frame multiframes. It lasts about 6.12 s. The final layer is the hyperframe which consists of 2,048 superframes and lasts almost 3.5 hours. The hyperframe period is this long for encryption purposes which is not addressed here. This structure can be seen in Fig. 1.

A. Burst Types

There are several burst types in the GSM standard. Each burst consists of the same number of bits (156.25) but the contents of those bits can be different. Here are several of the burst types [1] that will be important in order to understand the limited PHY layer implementation.

- Broadcast Control Channel (BCCH) - Sends information regarding the network identity, operating characteristics, and a list of available channels.

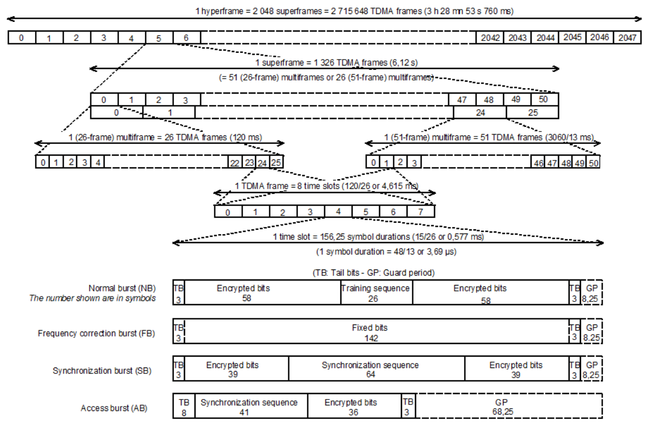

1 hyperframe = 2 048 superframes = 2 715 648 TDMA frames (3 h 28 mn 53 s 760 ms)
1 superframe = 1 326 TDMA frames (6,12 s)
(= 51 (26-frame) multiframes or 26 (51-frame) multiframes)
1 (26-frame) multiframe = 26 TDMA frames (120 ms)
1 (51-frame) multiframe = 51 TDMA frames (3060/13 ms)
1 TDMA frame = 8 time slots (120/26 or 4,615 ms)
1 time slot = 156,25 symbol durations (15/26 or 0,577 ms)
(1 symbol duration = 48/13 or 3,69 µs)
(TB: Tail bits - GP: Guard period)
Normal burst (NB)
The number shown are in symbols
Frequency correction burst (FB)
Synchronization burst (SB)
Access burst (AB)
Encrypted bits
Training sequence
Fixed bits
Synchronization sequence
GP

Figure 1. The GSM data structure and some burst types. Figure taken from 3GPP TS 45.001.

- Frequency Correction Channel (FCCH) - Used by receivers to synchronize clocks with the transmitter.
- Synchronization Channel (SCH) - Sends frame number (FN) and base station identity code (BSIC) information so that a user can locate there location within a hyperframe.
- Common Control Channel (CCCH) - CCCH can be any of the following: paging channel, random access channel, access grant channel. Sends information to the user (depends on specific channel type).
- Normal Burst - The most common burst for data transmission.

As can be seen in Fig. 1 the data in these bursts is organized in different ways. The specifics will be observed in upcoming sections.

III. The Control Multiframe

The purpose of the control multiframe is to help users on the receiving end tune into, synchronize, and get channel information from the transmitter. The control multiframe is made up of 51 frames and follows the pattern shown in Fig. 2.

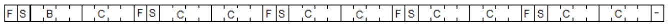


Figure 2. The 51-frame control sequence. F - FCCH, S - SCH, C - CCCH, B - BCCH, - - idle. Figure taken from 3GPP TS 45.001.

A. Frequency Correction Channel

The purpose of the FCCH is for receivers to be able to synchronize their clocks to that of the transmitter. The 156.25 bits of the FCCH are organized as follow: 3 start bits, 142 zero bits, 3 stop bits, and 8.25 guard bits. The start and stop bits are also all zeros. The 8 guard bits are all ones (note again that for MATLAB purposes the 0.25 bits are ignored).

B. Synchronization Channel

The purpose of the synchronization channel is to provide the receiver with the BSIC and the FN. The frame number is important for encryption reasons but serves no real purpose in this implementation. Frame numbers range from 0 to 2715647 uniquely identifying a frame within a hyperframe. The SCH is organized as follow: 3 start bits, 39 encoded bits, 64 training bits, 39 encoded bits, 3 stop bits, and 8.25 guard bits. The start, stop, and guard bits are the same as they where for the FCCH. There are a total of 78 encoded bits that contain the BSIC and FN. The training bits are used for channel estimation and are always the same (these can be found in GSM 05.02).

The 78 encoded bits are derived from 25 information bits. The BSIC takes up 6 bits and the FN the other 19. The FN is represented as follows (taken from GSM 05.02):

- T1 (11 bits) = $\lfloor \frac{FN}{26*51} \rfloor$, Range = 0 to 2047
- T2 (5 bits) = FN mod 26, Range = 0 to 25
- T3' (3 bits) = $\lfloor \frac{T3-1}{10} \rfloor$, Range = 0 to 3
  T3 = FN mod 51, Range = 0 to 50

T1 represents the superframe in which the frame is located. T2 and T3' can then uniquely determine a frame within the superframe (Chinese Remainder Theorem required). So through these 19 bits each frame in the hyperframe has a unique identifier composed of 19 bits.

Let these 25 bits (BSIC and FN) be denoted by d(0), ..., d(24). These 25 bits will be systematically encoded to produce 10 parity bits. To do this the cyclic code

$$g(x) = x^{10} + x^8 + x^6 + x^5 + x^4 + x^2 + 1$$

over GF(2) will be used. First let d(x) be the Galois field polynomial of degree 24 with coefficients that are d(0), ..., d(24). Next multiply d(x) by x^{10}. This will shift the data bits so that they are the coefficients of the degree 10 to 34 terms and the coefficients for the degree 0 to 9 terms will be zero. Continue by dividing $d(x)x^{10}$ by g(x) and keeping the remainder polynomial r(x). r(x) is a polynomial of degree 9 where the degree 9 coefficient is not necessarily 1. Inverting the 10 coefficients of r(x) gives the parity bits (or parity polynomial p(x)). The actual criterion set forward by the GSM standard is that $d(x)x^{10} + p(x)$ must have a remainder polynomial of $\sum_{m=0}^{9} x^m$ when divided by g(x). The construction described in [2] (explained here) satisfies this criterion.

Now there are 35 bits. To these, 4 zero bits are added to bring the total to 39 bits. Lastly a rate 1/2 convolutional code is applied. The convolutional code is defined by Eq.1. This takes our 39 bits and returns 78 encoded bits. The bits are then organized as the burst requires.

$$\begin{aligned} g_0(x) &= 1 + x^3 + x^4 \\ g_1(x) &= 1 + x + x^3 + x^4 \end{aligned} \tag{1}$$

C. BCCH and CCCH

In the implementation performed the BCCH and CCCH channel where both implemented as normal bursts (CCCH can have different bit organization). They both consist of 3 start bits, 57 encoded bits, 1 stealing flag, 26 training bits, 1 stealing flag, 57 encoded bits, 3 stop bits, and 8.25 guard bits. The stealing flags and training bits where all set to 0. The GSM standard does define the training sequence bits but there are 8 possible sequences and it was unclear how they where assigned. Setting these bits to zero had no impact on the rest of the implementation as no equalization techniques were performed at the receiver.

BCCH and CCCH bursts are done 4 at a time (4-frame extent). Throughout these 4 frames 184 bits are encoded into 456 bits and are interleaved. First the 184 bits are systematically encoded to have 40 parity bits. The procedure is the same as for the SCH except this time the generator polynomial is given by

$$g(x) = (x^{23} + 1)(x^{17} + x^3 + 1)$$

d(x) represents the 184 bits. d(x) is multiplied by x^{40} and the remainder of $d(x)x^{40} + p(x)$ should be equal to $\sum_{m=0}^{39} x^m$ when divided by the generator polynomial. After this is done we have 224 bits. To this we append 4 zero bits to reach 248 bits. These bits are also encoded with the convolutional code defined by Eq. 1 to give 456 encoded bits.

After the bits are fully encoded the 456 bits are interleaved among the 4 frames. Each of the 4 frames holds 114 of these encoded bits. The encoded bits are originally ordered e(0), ..., e(455). The mapping for the interleaved bits i(B,j) is given by Eq. 2. B represents 1 of the 4 frames and j indicates which of the 114 positions in the frame the encoded bit is being assigned. This interleaving not only shuffles the bits within a single frame but among frames as well. Once the interleaving is performed the CCCH / BCCH bursts is ready to be sent.

$$\begin{aligned} i(B, j) &= e(k) \\ k &= 0, 1, ..., 255 \\ B &= k \bmod 4 \\ j &= 2((49k) \bmod 4) + \left\lfloor \frac{k \bmod 8}{4} \right\rfloor \end{aligned} \tag{2}$$

This concludes the examination of all the burst types in the control multiframe. This bit organization within a frame for the burst types listed can be seen in Fig. 3.

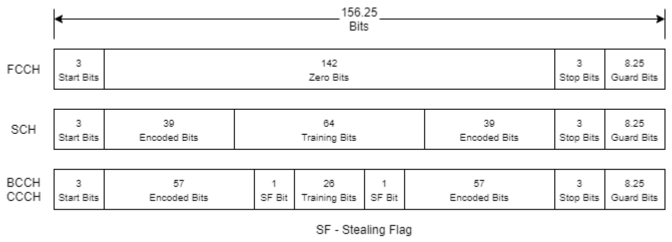


Figure 3. Burst structure for the burst types mentioned.

IV. TCH/F9.6

There are many different traffic channels (TCH) supported in the GSM standard. There are some for full rate and some for half rate data transmission. The traffic channels support both voice and data. Only the 9.6 $kbit/s$ full rate data TCH (TCH/F9.6) was implemented. TCH/F9.6 has the same bit organization as BCCH and CCCH however the bits are encoded and interleaved differently.

TCH/F9.6 takes 240 bits at a time and encodes and interleaves them over 22 consecutive frames. The process starts with appending 4 zero bits onto the 240 bits that are going to be encoded and sent. These four bits are meant to return the convolutional coder back to the zero state. These 244 bits are convolutionally encoded using the same 1/2 rate encoder defined by Eq. 1. This outputs 488 bits denoted by c(0), ..., c(487). Next 32 bits are removed by puncturing. The punctured bits are c(11 + 15j) where j = 0, ..., 31. They are not transmitted. This leaves 456 bits that need to be transmitted. Denote these bits e(0), ..., e(455).

The next step in the process is to interleave the bits. The encoded bits are interleaved so that they span 22 frames. Multiple blocks of 456 are interleaved into the same frames (two blocks of 456 overlap in the frames they use but not fully). The block and index position are given by B and j respectfully in Eq. 3.

$$\begin{gathered} i(B,j) = e(k) \\ k = 0, 1, ..., 455 \\ n = 0, 1, ..., N, N+1, ... \\ B = B_0 + 4n + (k \bmod 19) + \left\lfloor \frac{k}{114} \right\rfloor \\ j = (k \bmod 19) + 19(k \bmod 6) \end{gathered} \tag{3}$$

In the above B_0 is the starting FN, N represents N-240 bit blocks of information, n represents the current block of 240 bits being interleaved, and i represents the interleaved bit in frame B position j. Again, as in the SCH, the interleaving spreads the bits among frames and within frames. This is done to combat burst errors and packet drops.

V. Modulation

After the bits have been properly placed in their frames they are ready to be modulated. The modulation schemes used by GSM are gaussian minimum shift keying (GMSK) and 8 phase shift keying (8PSK). GMSK is the primary modulation used so it was the only one implemented.

GMSK is a variant of minimum shift keying (MSK). The main functioning principle behind MSK is to differentially encode bits and send use that to send symbols. The symbols in adjacent time slots will share either their in-phase (I) or quadrature (Q) component.

The differential encoding for MSK and GMSK is done according to Eq. 4 where d_n is the differentially encoded bit at time n and b_n and b_{n-1} are the bits being encoded (these are just the bits in the frames) and $\oplus$ is the symbol for the XOR operation.

$$d_n = b_n \oplus b_{n-1} \tag{4}$$

The symbol is then chosen based on the current symbol output and the current differential bit. Fig. 4 can be used to visualize this modulation. The blue dots represent the 4 possible symbols and are located at $(\pm\frac{\sqrt{2}}{2}, \pm\frac{\sqrt{2}}{2})$. When the differential bit is a 0 the next symbol is located in the 90° clockwise direction. When the differential bit is a 1 the next symbol is located in the 90° counterclockwise direction. Fig. 5 and 6 show the I and Q outputs from the modulator. Both figures have the same starting point at $(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and the horizontal black dashed lines represent $y = \pm\frac{\sqrt{2}}{2}$. The differential bit d_n is associated with the transition of I and Q from time t_n to t_{n+1}. So at each integer n the correct value from the constellation matches up with the I and Q output of the modulator.

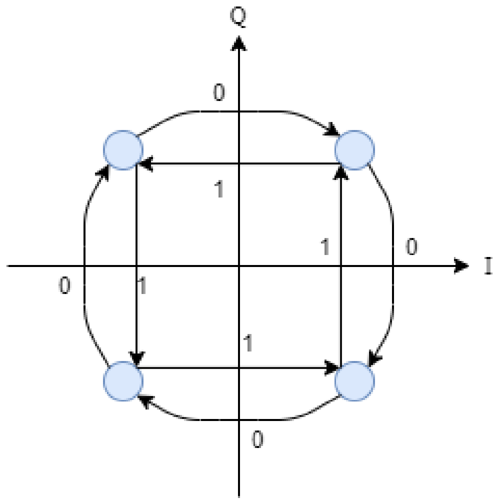


Figure 4. MSK modulation constellation.

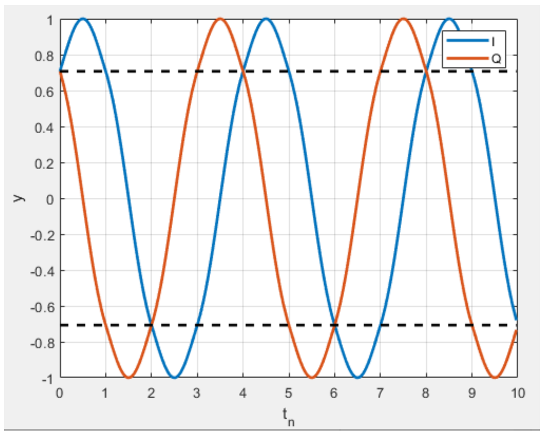


Figure 5. MSK modulated sequence. The differential bits are all 0.

GMSK aims to trade off one problem present in the MSK modulation scheme for another. Looking at Fig. 6 we can see

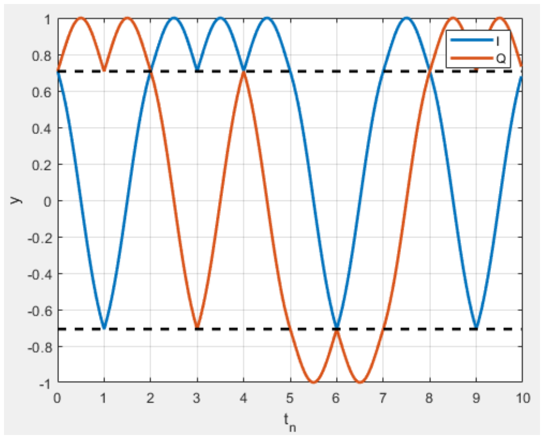


Figure 6. MSK modulated sequence. The differential bit sequence is 100100110.

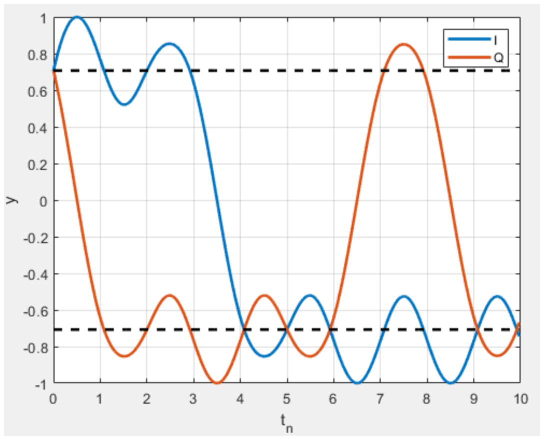


Figure 7. GMSK modulated sequence. The differential bit sequence is 100100110. Gaussian filter of length 4.

cusps in the I and Q waveforms. GMSK employs a gaussian filter to smooth out those cusps which will reduce the power present in the sidebands. The first step is to convert the differential encoded bits d_n to modulated values α_n through Eq. 5.

$$\alpha_n = 1 - 2d_n \tag{5}$$

The modulated values are then filtered with the gaussian filter $g(t)$ defined by Eq. 6. The output for the an input stream of all zeros is still the same as in Fig. 5. However, for the case when the differential bit stream is 100100110 (same as was used for Fig. 6) the output is different using GMSK, as seen in Fig. 7. While the cusps no longer exists, the trade off for this was that the I and Q waveforms don't transition according to the differential bit sequence. This is due to the fact we are convolving the modulated values with the filter. This causes intersymbol interference (that can be undone).

$$\begin{gathered} g(t) = h(t) * rect\Big(\frac{t}{T}\Big) \\ rect\Big(\frac{t}{T}\Big) = \begin{cases} \frac{1}{T} & |t| < \frac{T}{2} \\ 0 & otherwise \end{cases} \\ h(t) = \frac{1}{\sqrt{2\pi}\delta T} exp\Big(\frac{-t^2}{2\delta^2 T^2}\Big) \\ \delta = \frac{\sqrt{ln(2)}}{2\pi BT} \\ BT = 0.3 \end{gathered} \tag{6}$$

These I and Q waveforms (that are at baseband) are then combined with the carrier and then sent into the environment. In the simulation the signal was not placed on a carrier but kept at baseband.

VI. At the Receiver

In the simulations performed the receiver receives the modulated signal that was been corrupted by white noise. In order to recover the originally sent data several steps must be taken.

The first step is to demodulate the GMSK modulated data. This was done by using the built-in MATLAB GMSK demodulator object which employs the Viterbi algorithm to recover the differential bits. The differential bits are also converted back into frame bits (undoing the differential encoding).

Once the frames have been recovered, the appropriate sections of bits can be taken from the frame, since their location in the frame is known. The interleaving is also undone easily by just mapping the bits back in the reverse direction. Again the mapping is known so the bits can be reordered back into their original configuration.

Now the convolutional error correction code is undone by using the Viterbi algorithm once again. Note that for the TCH/F9.6 data the Viterbi algorithm has to account for the 32 punctured bits. Once this is done, the bits for TCH/F9.6 have been recovered. For the BCCH, CCCH, and SCH burst types the secondary codes need to be undone (the ones used to get the 10 or 40 parity bits). This is a simple task as the encoding method was systematic meaning the bits can just be taken straight from their corresponding position in the array of bits.

VII. Results

The implementation was tested in several ways. The first was to obtain a bit error rate (BER) curve of the TCH/F9.6 data channel. The second was to send an actual message through the same channel and see what happens at different SNRs. The last test was on the Control Multiframe. The goal in this last test was to properly encode and decode the control info.

A. *BER for TCH/F9.6*

The BER curve can be seen in Fig. 8. This curve was obtained using four samples per symbol. Using more samples per symbol is advantageous as it provides more information that can be used to boost the signal strength and thus require lower SNR for the same BER. This is up to receiver design. Using more samples per symbol causes the BER curve to move to the left. For the case shown, the GSM standard provides an SNR gain of 10 dB at a BER of 10^{-3}.

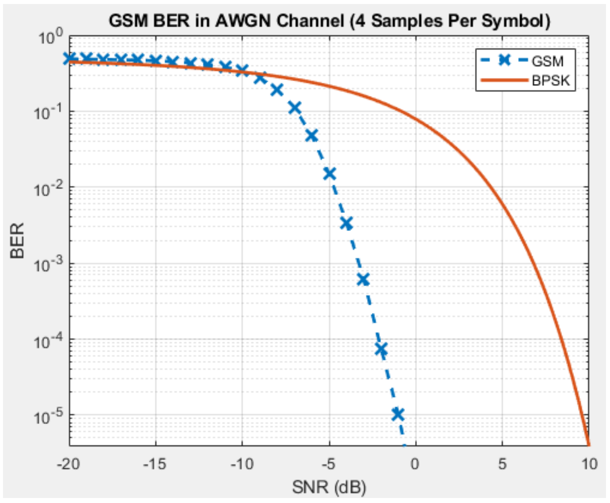
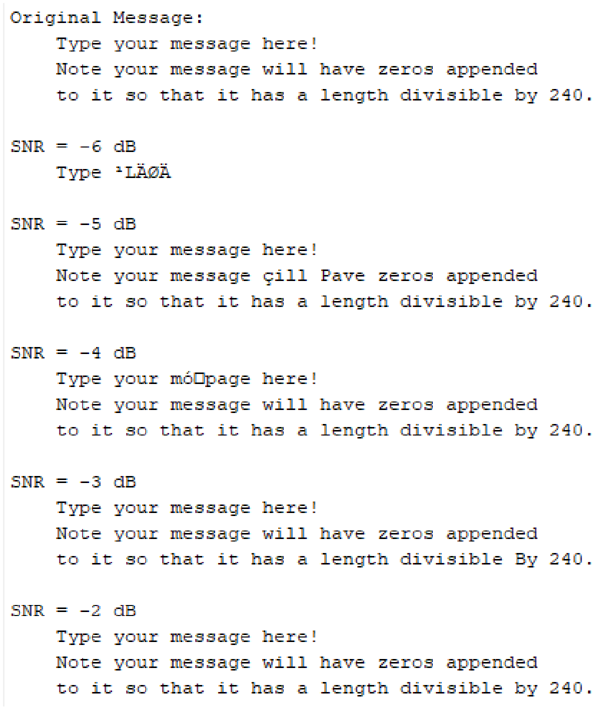


Figure 8. BER curve for TCH/F9.6 compared against uncoded BPSK.

B. *Message over TCH/F9.6*

The next test was to send a message over various SNRs. The results can be seen in Fig. 9. As can be seen the -6 dB case just loses it and MATLAB can't even display the characters. It gets better by the -4 dB case but words are still messed up. At -3 dB the message conveys the same information but the b in by is uppercase instead of lowercase. Finally in the -2 dB case the message comes across unharmed.



```
Original Message:
    Type your message here!
    Note your message will have zeros appended
    to it so that it has a length divisible by 240.

SNR = -6 dB
    Type ¹LÄØÄ

SNR = -5 dB
    Type your message here!
    Note your message çill Pave zeros appended
    to it so that it has a length divisible by 240.

SNR = -4 dB
    Type your mó□page here!
    Note your message will have zeros appended
    to it so that it has a length divisible by 240.

SNR = -3 dB
    Type your message here!
    Note your message will have zeros appended
    to it so that it has a length divisible By 240.

SNR = -2 dB
    Type your message here!
    Note your message will have zeros appended
    to it so that it has a length divisible by 240.
```

Figure 9. Message sent over TCH/F9.6 at various SNRs.

C. *The Control Multiframe*

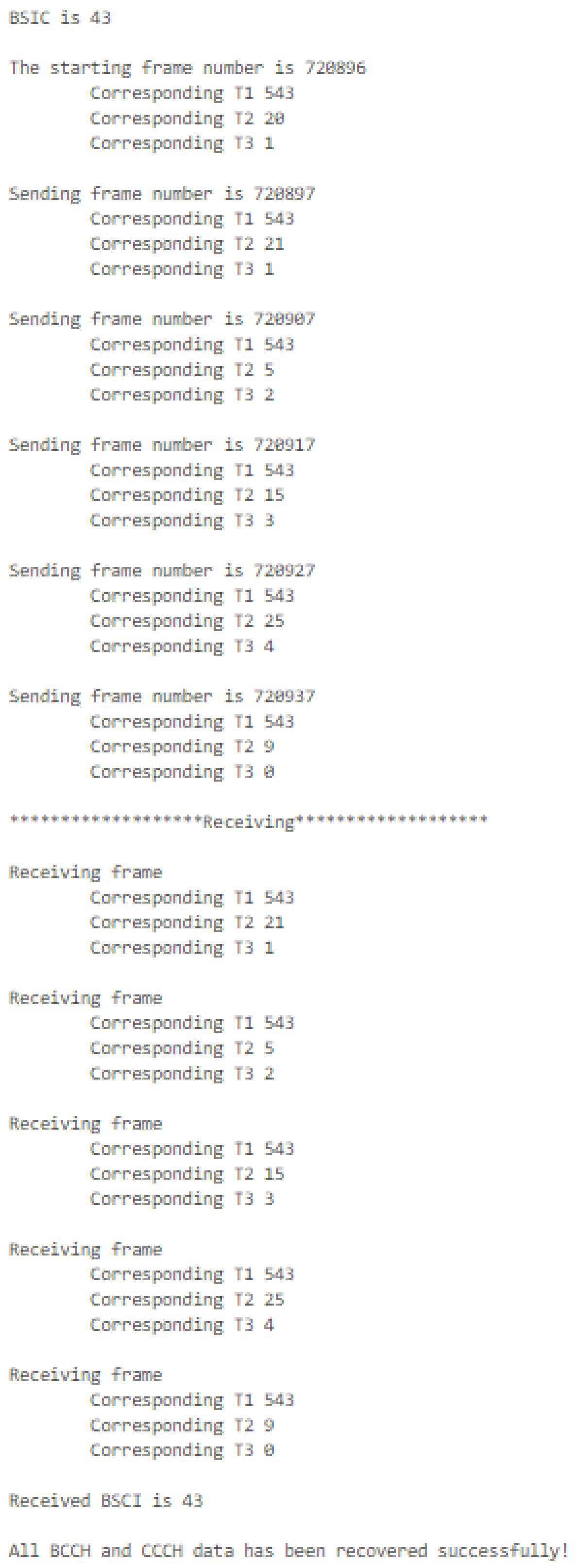
The results of testing the control multiframe can be seen in Fig. 10. This test was done with no noise added since we just need to confirm this was done correctly. Noise can make that annoying to do. As can be seen from the figure the BSIC was properly transmitted as well as the frame information. The data also present in the BCCH and CCCH channels was also correctly received (check the code to verify I'm not cheating). Also the output is clear of parity incorrect errors meaning the systematic codes where applied correctly (again check the code to see the errors where not triggered).

VIII. Conclusion

The GSM wireless standard effectively uses a TDMA messaging scheme to send data reliably to users. This is done using a control channel that helps a user match their clock speeds, synchronize their frames, and receive channel information. The data is also then sent to the users with a low BER of $10{-}3$ at a 10 dB SNR gain compared to BPSK at the same BER. GSM effectively uses error control coding, interleaving, and modulation to help obtain this level of performance.

References

[1] Theodore S. Rappaport. *Wireless Communications Principles and Practice*.
[2] Patric Ostergard. http://www.comlab.hut.fi/studies/3410/slides_08_6_4.pdf.



```
BSIC is 43

The starting frame number is 720896
        Corresponding T1 543
        Corresponding T2 20
        Corresponding T3 1

Sending frame number is 720897
        Corresponding T1 543
        Corresponding T2 21
        Corresponding T3 1

Sending frame number is 720907
        Corresponding T1 543
        Corresponding T2 5
        Corresponding T3 2

Sending frame number is 720917
        Corresponding T1 543
        Corresponding T2 15
        Corresponding T3 3

Sending frame number is 720927
        Corresponding T1 543
        Corresponding T2 25
        Corresponding T3 4

Sending frame number is 720937
        Corresponding T1 543
        Corresponding T2 9
        Corresponding T3 0

*******************Receiving*******************

Receiving frame
        Corresponding T1 543
        Corresponding T2 21
        Corresponding T3 1

Receiving frame
        Corresponding T1 543
        Corresponding T2 5
        Corresponding T3 2

Receiving frame
        Corresponding T1 543
        Corresponding T2 15
        Corresponding T3 3

Receiving frame
        Corresponding T1 543
        Corresponding T2 25
        Corresponding T3 4

Receiving frame
        Corresponding T1 543
        Corresponding T2 9
        Corresponding T3 0

Received BSCI is 43

All BCCH and CCCH data has been recovered successfully!
```

Figure 10. Checking the control multiframe.