DESIGN, DEVELOPMENT AND IMPLEMENTATION

OF

MIL-STD-188-110B DATA WAVEFORMS

GROUP MEMBERS

RABEEL HASSAN

ANAM TARIQ KIDWAI

HARRIS HAFEEZ

SAULAT ABBAS

"Attention, the Universe' By kingdoms, right wheel! .

These profound words, sent by Samuel F.B. Morse in 1838, form the first telegraph message on record. With the transmission of this message, was born the era of electrical communication. Since then, the field of telecommunications has developed continuously with the passage of time. From the audible world of radio, to the visual pleasures of television, this Field of information transfer has become an integral part of modem human life. The very essence of information transfer lies in the fact that every single communication device built has to perform a single basic function, that of reliable transfer of data with maximum efficiency and with minimum loss of data. Where the requirement of reliability or secrecy of transmitted data becomes all the more important in military scenario , Our project, titled *'Transmission of Encrypted Data through HF Channel'* is an attempt to setup a communication system that performs the very fundamental task of data communication reliably using military standards.

The physical layer represents the lowest level of the OSI model. It is this layer that conveys the binary “bits” over a HF channel. Standards for this layer define the way individual bits are represented in the waveforms being sent over HF channels.
There are four sets of standards currently available for the physical layer. These are MIL-STD-188-110B [1] and MIL-STD-188-141B [2] from the US Military, STANAGs 4285 [5] and 4539 [6] from NATO, and FED-STD-1052 [9] from the US Federal Government.

2.3 HF Signal Standards

To help improve the reliability and efficiency of HF transmissions, several standards were developed specifically for HF transmissions. The father of these standards was the Military Standard (MIL-STD) 188-110A developed by the U.S. Department of Defense [43]. From this standard North Atlantic Treaty Organization (NATO) released several Standardization Agreements (STANAG), [25, 26] to name a few, providing HF communications standards.

This standard supports 6 different bit rates, 75, 150, 300, 600, 1200, 2400, and 4800

bps. At the bit level the data to be transmitted undergoes forward error correction. This is implemented by a 7*th* order convolutional code and an interleaver. Two di®erent interleaver choices exist, a short interleave and a long interleave. The forward error corrected bits are grouped by prede¯ned amounts, scrambled by a pseudo-random sequence, and mapped to the 8-PSK constellation. The probes are a pre defined sequence that are scrambled by the same pseudo-random sequence as the unknown data and mapped to an 8-PSK constellation. The length of the the unknown data frame and the probe are determined by the data through put rate.

The physical layer represents the lowest level of the OSI model. It is this layer that conveys the binary "bits" over a HF channel. Standards for this layer define the way individual bits are represented in the waveforms being sent over HF channels. There are four sets of standards currently available for the physical layer. These are:-

From the US Military

1. MIL-STD-188-110B

2. MIL-STD-188-141B

From NATO

3 STANAG 4285 [5]

4. STANAG 4539 [6]

5 FED-STD-1052 [9], from the US Federal government.

The standards fully define the modulation schemes that may be used, and the channel coding methods that can be employed. They can also define a method known as “auto-baud” that lets a HF modem automatically determine the bit rate and interleaver settings of the incoming waveform.

3.1.1 MIL-STD-188-110

This standard defines 7 classes of modems in its main body: 75, 150, 300, 600, 1200, 2400 and 4800bps data rates. The latest revision, 110B, contains Appendix C which details HF data modem waveforms for data rates above 2400bps. It defines a 3200, 4800, 6400, 8000 and 9600bps coded waveform and an uncoded waveform of 12800bps.

4.1 Waveforms

There are many HF modems on the market currently with dozens of different waveforms being implemented. Most HF modems targeted at the military support three standard waveforms, the MIL-STD-188-110A series, the MIL-STD-188-110B for faster speeds and the STANAG 4285 waveforms. The remaining waveforms are predominantly designed for non-military customers. The various waveforms will now be discussed, with common performance criteria being used for later comparison.

4.1.1 MIL-STD-110A

This family of waveforms dates from 1991 and is essentially the baseline for all other

military standards. The main waveforms used from this standard are a set of serial tone, 8-ary Phase Shift Keying modulated waveforms designed for data rates up to 4800bps. Each data frame contains a “known” portion (a constant pattern defined in the modem standard) that is used to adapt the modem’s parameters to the channel, and an “unknown” portion containing the data payload. The data payload is Forward Error Corrected (FEC) to produce error protection on the data stream. Lower data rates generally have higher levels of FEC. The details of each waveform rate can be seen in Table 2.

Table 2 - MIL-STD-110A waveform characteristics, (from table XIX in [1])

User Data Rate	Coding Rate (FEC)	Channel Rate	No. of unknown symbols	No. of known symbols
4800	(none)	4800	32	16
2400	½	4800	32	16
1200	½	2400	20	20
600	½	1200	20	20
300	¼	1200	20	20
150	⅛	1200	20	20
75	½	150	ALL	0

The 110A standard also has the concept of “auto-baud” for its waveforms. This feature embeds information in the waveform about its data rate and interleaver depth. Modems can then automatically detect the characteristics of the incoming waveform rather than having to be preset with the waveform type being used. This makes it possible to be agile with the waveform in use, making it easier to respond to changing environmental and operational conditions. The Signal to Noise Ratio (SNR) requirements of these waveforms can be seen in the graph at the end of this section. 110A also defines two parallel tone modems (a 16 tone version in Appendix A and a 39

tone version in appendix B) that offer data rates up to 2400bps. The parallel tone modems operate within the same 3kHz bandwidth of serial tone modems but they can achieve high data rates without having to perform equalization on the channel, which results in a significant drop in processing power needed for the modem. Current high speed/low power DSP technology essentially removes this advantage however. The Advanced Narrowband Digital Voice Terminal (ANDVT) specification uses a 39-tone modem for its voice data transfer at 2400bps.

4.1.2 MIL-STD-110B

MIL-STD-110B contains the 110A standard and adds higher speed data waveforms in Appendix C. These waveforms use various modulation methods to achieve their speeds. Table 3 details the exact waveform for each speed.

Table 3- Modulation types used to obtain data rates in MIL-STD-110B Appendix C (from [1])

Data Rate (bps)	Modulation	Coding Rate (FEC)
12800	64QAM	(none)
9600	64QAM	¾
8000	32QAM	¾
6400	16QAM	¾
4800	8PSK	¾
3200	QPSK	¾

The standard uses FEC and interleavers to achieve its error performance. The data is conveyed in a frame structure that contains a similar known/unknown data sequences as seen in MIL-STD-188-110A.

Figure 2 - Frame structure for all MIL-STD-188-110B Appendix C waveforms (from [1])

The data block in the structure depicted in Figure 2 contains the interleaved FEC data.

The repeated known blocks allow the modem to characterize the current channel

Conditions and the repeated preamble allows the modem to acquire the data signal

after the initial connection stage.

A.6.1 Error Detection and Correction

Data transmission devices depend on the ability to detect errors so that they can provide reliable delivery of data. This is usually accomplished by using a hashing algorithm to produce a fingerprint of the data in the packet. If the data is corrupted in transmission then hopefully the corrupted data will no longer hash to the value stored in the hash field. This hashing relies on probabilities not certainties to detect errors.

The number of errors detectable by a hashing code (or any code for that matter) is

dependent on its length. For every bit in the error detection field a 1-bit error in the

data packet can be detected. So high error channels require lots of error detection bits

to detect errors.

Detection only handles half the problem however. Once the error is detected the correct

data must then be obtained however. A simple method (used on the Internet) is to

simply re-send that corrupt packet (ie ARQ). This has two problems for the HF

environment however. The low data rate means re-sending data wastes a valuable

resource that is already in demand. It is also resending that data across the same high

error channel so there is a high likelihood of an error occurring in the resent packet,

thereby causing the same problem yet again.

A process known as Forward Error Correction (FEC) solves this problem. FEC corrects

errors in packets using extra redundant (FEC) information. The number of errors that

can be corrected by FEC is equal to about half the number of FEC bits sent. For

example, sending 2 bits of FEC per byte means that 1 error in that byte can be

corrected. The 2400bps 110A waveform for example uses ½ rate FEC. For every 1 data

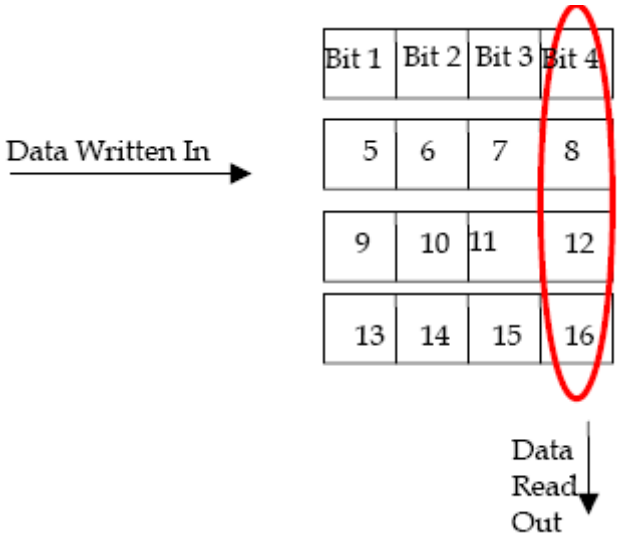
bit a FEC bit is also added, meaning that ½ of the packet's data contents can be possibly corrected. The amount of FEC and the timeouts of the ARQ protocol can be balanced off against each other to maximize the amount of useful throughput of the channel.

A.6.2 Interleaving

FEC coding and error detection in general are based around correcting random errors in data, however HF channels often produce burst errors. Burst errors are runs of errors in the data rather than random events affecting a single symbol. Events such as lightning strikes cause burst errors as they totally swamp the signal for a period of time.

To overcome the effect of burst errors a concept called "interleaving" is used. Interleaving reorders the data in packets so that burst errors are a series of isolated single bit errors in the packet thereby allowing FEC to work properly. An example of interleaving can be seen in the diagram below.

Data is written into the system from the right with each bit filling one cell in the buffer, from left to right, top to bottom. This data is then read out of the buffer in a top to bottom, right to left order. The receiver then reverses the processes to deinterleave the data.



However, interleaving comes at the cost of adding extra latency to the data. It takes time for the whole interleaver structure to be filled, and it must be complete before the first column can be sent. The same effect happens at the receiver. For time critical applications such as secure voice this interleaving latency can have a big effect on the perceived performance of the system.

we introduce generalized tail biting encoding as a means to ameliorate the rate deficiency caused by zero-tail convolutional encoding. This technique provides an important link between quasi-cyclic block and convolutional codes

GLOSSARY

ALE	**Automatic link establishment**
ARQ	**Automatic repeat request**
Bd	**Baud**
BER	**Bit error ratio**
bps	**Bits per second**
BW	**Bandwidth**
dB	**Decibel(s)**
EOM	**End of message**
FEC	**Forward Error correction**
HF	**High Frequency**
Hz	**Hertz**
kHz	**kilohertz (1,000 hertz)**
LSB	**Least significant bit**
LMS	**Least mean square.**
MSB	**Most Significant Bit**
NATO	**North Atlantic Treaty Organization**
PSK	**Phase-shift keying**

QAM	**Quadrature amplitude modulation**
rms	**root-mean-square**
SNR	**Signal-to-noise ratio**
s	**second(s)**
TX	**Transmit**
VHF	**Very high frequency**
VLF	**Very low frequency**

DEDICATION

DEDICATED TO OUR BELOVED PARENTS

WHO HAVE BEEN A CONSTANT SOURCE

OF ENCOURAGEMENT FOR US.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award of qualification either at this institution or at elsewhere.

CONTENTS

SECTION TWO

CHAPTER FOUR: TRANSMITTER

CHAPTER FIVE: RECIEVER

SECTION ONE

OVERVIEW

SECTION TWO

DESIGN

SECTION THREE

IMPLEMENTATION

CHAPTER 1

HF COMMUNICATION OVERVIEW

1 – INTRODUCTION

Radio frequency transmission between 3 and 30 MHz is called high frequency (HF) or "shortwave" radio communications. HF is the widely used communication band for long distances since invention of radio by Markoni and Popov. HF communications are growing at the moment despite appearing and all-around implementation of cellular and satellite communication systems. It's the only way of achieving global communication coverage without using expensive terrestrial and satellite infrastructure. HF communication still remains a fail-safe means for communicating in high latitudes, including communication with mobile objects, e.g. for tracking airplanes on their routes.

2 – HF PROPAGATION SPECIFICS

HF signal propagation is determined by the fact that it's repeatedly reflected from the ionosphere and the earth surface. That's why they often call this communication method the "skywave communication". This phenomenon allows amateur radio operators to communicate with their colleagues on the other side of the globe with the radiated power no more than several Watts. Still, analog communication using SSB without coding doesn't allow providing a reliable and uninterrupted channel. It is determined by significant changes of propagation conditions depending on the geographical location, frequency, time of day, season, solar activity, and other circumstances. Typically, the high-end frequencies are best during the day while the low-end frequencies are best during the day.

However, besides long-term instability of the propagation conditions it turns out that the signal passed via the HF channel undergoes significant distortions and the dynamic of these distortions can have sufficiently little time constants. It is determined by the multipath nature of the signal propagation and distributed reflection from the ionosphere. As a result of it there appears the effect called 'fading'. The multipath is characterized by the value of delay between replicas of the signal. Typical values significantly depend on the length of the signal path and hesitate from 1 to 10 milliseconds. Fadings are characterized

by the so-called 'bandwidth' which in middle latitudes is within the limits of 2 Hz, but in equatorial and high latitudes it can have the value up to 20 Hz. The results of measurements within the DAMSON program in high latitudes showed that quite often the fading bandwidth achieves the value up to 73 Hz.

Frequency shifts and frequency-spread distortions imposed on the transmitted signal by ionosphere reflection are determined by temporal changeability of ionic concentration. Despite the fact that in middle latitudes the frequency shift is measured in fractions of Hertz this phenomenon can cause prolong periods of weakening the signal, cased by coherent subtraction of signals coming with different delays.

3 – HF COMMUNICATION REQUIREMETNS

The information on HF propagation has only statistic nature and forms the background for designing as terminal equipment (modems and others), as communication systems as a whole.

Changeability of propagation conditions makes specific requirements of applying adaptive mechanisms as on the modem level, as on the controlling protocols level, which decreases the demands to the qualification of the operating personnel.

Modems should be able to combat with distortions brought in by the HF channel (multipath, fading). It's very desirable to have the ability of automatic detection of the transmitted signal parameters (autobaud feature). Since military customers represent a significant part of HF communication equipment users, it makes additional requirements for reliability, suppressing interference and jams.

Just 10-15 years ago it was widely believed that 75 bps is the only relevantly reliable rate, while the availability for the rates up to 2400 bps remained very low. Rapid development of digital signal processing during the last ten years made possible to substantially improve features of HF communication systems. Modern standards for HF

communications take into account all the above-mentioned circumstances and set a high level of BER performance for modems. As a result the transmission rates up to 9600 bps in a 3 kHz channel has been achieved. Data rates on benign skywave channels and on ground-wave paths can now reach 64 kbps, though sometimes using wider channel bandwidths, and there is an increasing desire to reliably achieve 16 kbps. From the other side, channel availability for the rates lower than 2400 bps is significantly increased. High-level protocols, such as STANAG 4538 and STANAG 5066 possess mechanisms of adaptive controlling the transmission rate and increasing data transmission reliability under disturbed conditions.

Improvements are achieved by introducing new modern reception techniques such as turbo equalization, MAP decoding, etc. The cost of improved performance is increased hardware requirements to HF modems. However, novel DSPs, such as C55x or C67x from Texas Instruments fit such hardware requirements with low number of external components.

4 – STANDARDIZED WAVEFORMS

There are several types of waveforms, which are using for HF transmission: single-tone, OFDM and orthogonal signal modulation. Each type of waveform has own advantages and modern modems should support several waveforms to fit system needs.

Now, the best characteristics for HF channels at rates below 9600 bps are provided by single-tone modems. They provides robust operation under interference and capable to withstand strong and fast varying ISI. So, let's overview the most widespread single-tone waveform formats.

Channel allocations for military HF communications have a bandwidth of 3 kHz. To fit into this bandwidth, a 1800 Hz subcarrier is I/Q-modulated at a signaling rate of 2400 baud. Sequences of known training symbols are regularly inserted into the stream of transmitted symbols in order to aid the receiver in tracking the time-varying channel impulse response. Typically, waveforms described

here use the ECC with a constraint length 7 convolutional code with the generator polynomials (133,171) of rate 1/2. Lower code rates are achieved by repeating the code bits, higher code rates are achieved by puncturing. To fight deep and long fadings they use interleaving of transmitted symbols. Making the interleaving longer usually results in improving BER. But on the other side it leads to increasing the delay. Particularly it is not acceptable for voice transmission; besides it increases overhead for ARQ protocols. That's why, for example, the STANAG4538 BW3 waveform designed especially for packet data transmission is not supplied with the interleaver. Table below summarizes waveform characteristics for most widespread modems.

Waveform	auto-baud	Modulation	Rates, bps	Interleavers, sec	Coding
110A single tone (150..2400 bps)	Yes	single-tone BPSK, QPSK, 8PSK	150, 300, 600, 1200, 2400, 4800 (uncoded)	block-based 0, 0.6, 4.8	convolutional (133,171), 1/8 ... 1/2
110A (App B) 39-tone	No	OFDM, QDPSK	75, 150, 300, 600, 1200, 2400	depends on rate, from 0 to 6.48	Reed-Solomon, 10/14, 3/7
110B App.C / 4539 HDR	Yes	single-tone QPSK, 8PSK, 16, 32 and 64QAM	3200, 4800, 6400, 8000, 9600, 12800 (uncoded)	block-based 0.12, 0.36, 1.08, 2.16, 4.32, 8.64	convolutional (133,171), 3/4 with tail-biting

STANAG 4415	**Yes**	**orthogonal signals, 4-ary Walsh functions, BPSK**	**75**	**block-based 0, 0.6, 4.8**	**convolutional (133,171), 1/32**
STANAG 4285	**No**	**single-tone, BPSK, QPSK, 8PSK**	**75, 150, 300, 600, 1200, 2400**	**convolutional, 0.853, 10.24**	**convolutional (133,171), 1/16, 1/8, 1/4,1/2, 2/3**
STANAG 4529	**No**	**single-tone, BPSK, QPSK, 8PSK**	**75, 150, 300, 600, 1200**	**convolutional, 1.706, 20.48**	**convolutional (133,171), 1/8, 1/4,1/2, 2/3**
STANAG 4538 BW2	**Yes**	**single-tone, 8PSK**	**up to ~4400**	**no interleaver**	**convolutional (235, 275, 313, 357), 1/1 to 1/4**
STANAG 4538 BW3	**Yes**	**orthogonal signals, 16-ary Walsh functions, BPSK**	**up to ~573**	**block-based, up to 7.2**	**convolutional (133,171) 1/12 to 1/24**

Table 1 – waveform characteristics for widespread modems

5 – CONCLUSION

Development of the HF modems requires deep knowledge of HF propagation specifics, comprehensive testing under simulated HF conditions and strong mathematical skill for algorithm deployment. It is impossible without using of channel simulator software and simulation impairments introduced by HF media as well as by associated communication equipment. Also, porting such computationally intensive tasks onto the target DSP platform (including fixed point processors) is not a trivial thing.

CHAPTER 2

MIL-STD-188-110B

2.1 – INTRODUCTION

MIL-STD-188-110 is a military standard documented by US Government which integrates every aspect of communication system and thus ensures a reliable and efficient High Frequency communication.

Two Basic Versions

- **MIL-STD-188-110 A, a second generation HF modem.**
- **MIL-STD-188-110 B, "A unified third generation HF messaging protocol", implemented on physical layer and operates at lower data rates (75 to 2400 bps).**
- **Appendix C of above mentioned document caters for higher data rates.**

2.2 – MIL-STD-188-110B (APPENDIX C)

This appendix presents specifications for coding and modem waveforms for data rates of 3200, 4800, 6400, 8000, 9600 and 12800 bps. The waveforms specified in this appendix are single-tone and use complex modulation techniques and large data blocks, in order to achieve the requirements necessary to obtain the stated data rates. A block interleaver is used to obtain 6 interleaving lengths ranging from 0.12 s to 8.64 s. Forward Error Correction - a method of error control that enables the system to deliver higher gain figures and degrade gracefully as more errors flood the channel - using a 1/2 rate convolutional code, with a constraint length of 7, which is punctured to rate 3/4, is used for all data rates. To produce block codes from this convolutional code that are the same length as the

interleaver, full-tail-biting approach is used. Since the minimum interleaver length is equal to a single data frame, there is no option of zero interleaving, as there would be no reduction in time delays.

Making use of Viterbi decoding algorithm integrated with an innovative approach that serves best for punctured data and gives an improved bit error rate and lesser processing gain than conventional Viterbi decoding algorithms. Gray coding is used to improve the bit error rate of the system, followed by scrambling, improving the bit error rate by breaking the continuous series of ones and zeros and also providing means of security by jumbling up the data.

The feature of "Auto-baud", which lets the HF modem automatically determine the bit rate and the interleaver settings of the incoming waveform. Data rate and interleaver settings are explicitly transmitted as a part of the waveform, both as part of the initial synchronization preamble and then periodically in a reinserted preamble and in the periodic known symbol blocks. This "autobaud" feature is critical in developing an efficient (ARQ) protocol for high frequency (HF) channels. The receiving modem is required to be able to deduce the data rate and interleaver setting both from the preamble and from the subsequent known data portion of the waveform.

2.1.1 – MODULATION

The symbol rate for all symbols shall be 2400 symbols-per-second. Phase-shift keying (PSK) and quadrature amplitude modulation (QAM) modulation techniques shall be used. The sub-carrier (or pair of quadrature sub-carriers in the case of QAM) shall be centered at 1800 Hz.

2.1.2 – FRAME STRUCTURE

The frame structure consists of an initial preamble of 287 symbols, followed by 72 frames of alternating data and known symbols. Each data frame shall consist of a data block consisting of 256 data symbols, followed by a mini-probe consisting of 31 symbols of known data.

2.1.3 – INTERLEAVING AND CODING

The interleaver used shall be a block interleaver. Each block of input data shall also be encoded using a block encoding technique with a code block size equal to the size of the block interleaver. Thus, the input data bits will be sent as successive blocks of bits that span the duration of the interleaver length selected.

2.1.4 – BER PERFORMANCE

The measured performance of the high data rate mode, using fixed-frequency operation and employing the maximum interleaving period (the 72-frame .Very Long. interleaver), shall achieve coded BER of no more than 1.0E-5.

CHAPTER 3

APPLICATIONS

3.1 – AN EFFECTIVE MEAN OF HIGH FREQUENCY COMMUNICATION

HF communications are growing at the moment despite appearing and all-around implementation of cellular and satellite communication systems. It's the only way of achieving global communication coverage without using expensive terrestrial and satellite infrastructure.

HF Communication provides wide utility in Military arena where adhoc communication is required with minimum assets and planning. VHF and Satellite communications are other means but extensive planning and heavy equipment is required.

3.2 – ROBUST WAVEFORMS

MIL-STD-188-110 B offers an exclusive modem that provides waveforms as a robust mean of communication because it operates at very high data rates (3200 bps to 12800 bps) by using many different modulation schemes and used to send text, digital image and voice on HF channels.

3.3 – PACKET BASED NETWORKS|

MIL-STD-188-110B offers a packet based Network. It uses a method of data transmission in which small blocks of data are transmitted rapidly over a channel dedicated to the connection only for the duration of the packet's transmission.

Packet based networks purpose following advantages

- **Provides a low probability of detection and interception because when you are transmitting in frames, you stay over channel for a short duration of time.**
- **The capacity of packet networks can be stretched further to handle higher traffic volumes.**
- **Sender and receiver can transmit at different rates.**
- **Increase reliability of communication**

3.4 – AUTOMATIC DETECTION

Modems is able to combat with distortions brought in by the HF channel (multipath, fading). It's very desirable to have the ability of automatic detection of the transmitted signal parameters (autobaud feature) which is provided in this modem, which increases efficiency of HF communication.

3.5 – SUSTAINABLITY AGAINST JAMMING

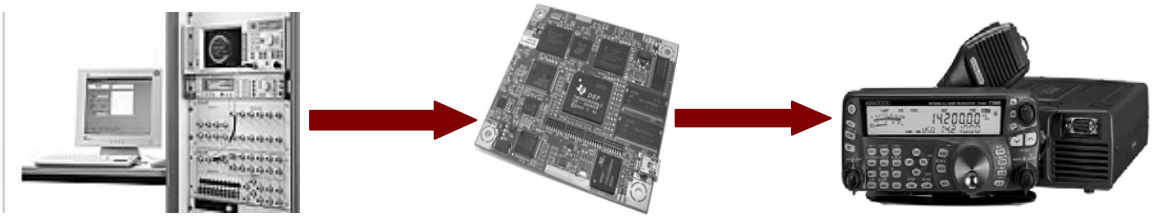
When ever voice is modulated it gets 60 to 65 % modulated and when data is modulated it gets 80- 85 % modulated. Higher the percentage of signal, modulated greater power is required to jam that signal. Since MIL-STD-188-110 B is a data modem hence it is very less susceptible to jamming.

3.6 – CONVERSION OF ANALOG RADIOS TO DIGITAL RADIOS

A digital radio is the one in which transmitter processes sounds into patterns of numbers, or "digits"– hence the term "digital radio." In contrast, traditional analog radios process sounds into patterns of electrical signals that resemble sound waves.

Digital radio reception is more resistant to interference, and eliminates many imperfections of analog radio transmission and reception. There may be some interference to digital radio signals in areas that are distant from a station's transmitter.

Using these waveforms along with Analog Radio sets used in Military field units like R1000, R2000 and R7000, can make them to give performance equivalent to Digital Radio Sets, provided by HARRIS and thus providing a low cost solution and improving efficiency of existing equipment.



USER INTERFACE | *MIL-STD-188-110 B DATA WAVEFORMS* | *ANALOG RADIO SET*

Figure 3-1 A model for conversion of Analog Radio set to Digital

3.7 – A SECONDARY MEANS OF COMMUNICATION

This modem can be used with radio sets to provide a very efficient secondary mean of communication if primary media like, Very High Frequency Network or Satellite Network goes off because a lot of planning and heavy equipment is require to resume transmission for that case.

3.8 – COMPATIBILITY WITH DIGITAL RADIO SETS

Integrating this modem with Analog Radio Sets, they can be made compatible with Digital Radio sets like the one provided by *HARRIS*. Now Analog Radio sets can capture Digital wave forms and decode them.

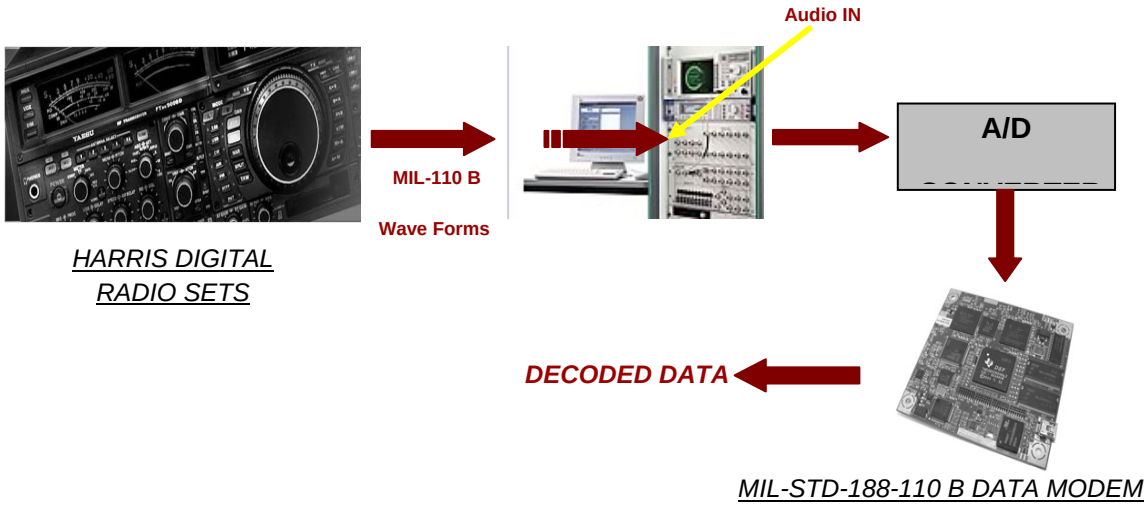


Figure 3-2 A model showing compatibility of Analog Radio set with Digital Radio set

3.9 – ENHANCING SECURITY

The Digital radios provided by HARRIS, which are used by Pakistan Army incorporate this modem, whose architecture was unknown. Now by implementing this modem, the changes in the architecture can be made as per requirements of Pakistan Army, in spite of using only provided system and thus can enhance their security.

3.10 – A FIRST STEP TOWARDS SOFTWARE DEFINED RADIOS

SDR is the radio in which different wave forms can be generated by making a little change in Software only without making any change in Hardware. An SDR performs significant amounts of signal processing in a general purpose computer, or a reconfigurable piece of digital electronics. The goal of this design is to produce a radio that can receive and transmit a new form of radio protocol just by running new software. Software radios have significant utility for the military and cell phone services, both of which must serve a wide variety of changing radio protocols in real time.

Implementation of this Modem is the first step towards SDR

- **Because SDR can handle digitized voice and data using this modem.**
- **Six different data rates offered by this modem allow the use of different waveforms by SDR.**
- **Modem allows a great deal to flexibility in its architecture making its application more diverse.**

CHAPTER 4

TRANSMITTER

4.1 – DATA RATES AND MODULATION SCHEMES

MIL-STD –188-110B is capable of operating on a wide range of data rates starting from 75bps up to 9600bps (coded) and 12800bps (uncoded). The implementation of modules for low data rates (75bps to 4800bps) is a bit different from data rates of 3200bps to 9600bps.Higher data rates of 3200 to 9600bps are covered in Appendix C of MIL-STD-110B.

We have implemented five different modulation schemes for corresponding five data rates giving modem the feature of auto baud.

DATA RATES	MODULATION SCHEMES
3200	QPSK
4800	8-PSK
6400	16 QAM
8000	32 QAM
9600	64 QAM

TABLE 4-1 Data rates and Modulation Schemes

4.2 – MODE OF OPERATION

Our transceiver is a single tone modem capable of working in full duplex mode.

4.3 – TRANSMITTER

A general structure of transmitter as implemented in our project is shown below:-

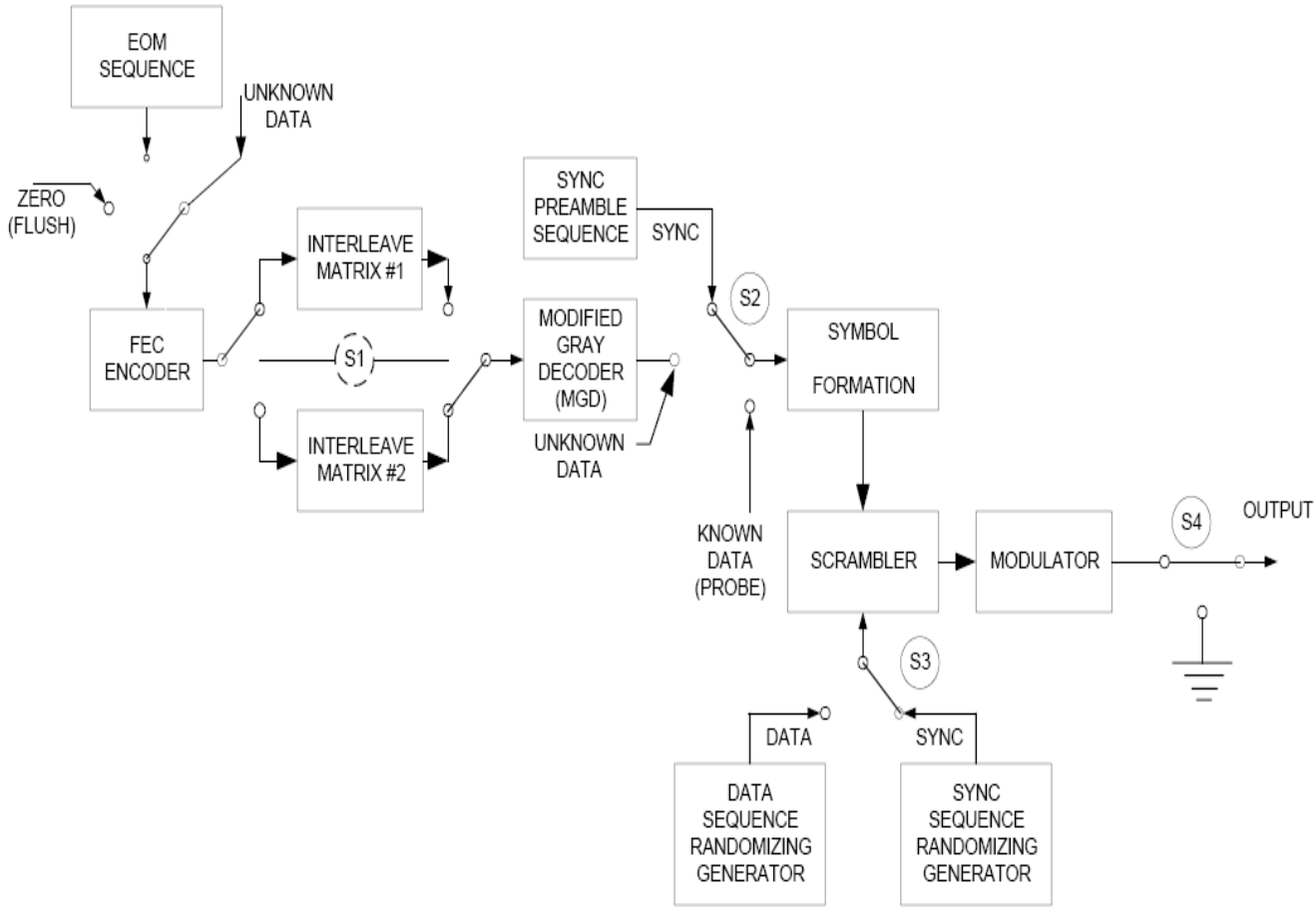


FIGURE 4-1 Transmitter Block Diagram

4.4 – UNKNOWN DATA

Unknown data is the data entered by the user to be transmitted. Our modem is capable of processing following types of input data.

- **Simple Plaintext**
- **Text file**
- **Image**
- **Voice**

Whatever the input data type is, it is first converted into respective ASCII values consisting of binary 1's and 0's.

We have design two options, either the user specify the data rate and number of frames to be transmitted OR only gives the data block to be transmitted as input, the modem will decide the optimum data rate and number of frames for that data block depending upon its length (where it selects the highest optimum data rate, with minimum no of frames to be transmitted, in order to reduce over head).

4.1.1 – SIMPLE PLAIN TEXT

User can enter any text to be transmitted

4.1.2 – TEXT FILE

User can also send text files with extension ".txt".

4.1.3 – IMAGES

The images we have included are given below. These options can however be increased upto any number:

1) **2)** **3)** **4)**

4.1.4 – VOICE CLIPS

Voice clips we have integrated in our project are:-

- **Bean walk**
- **Blip**
- **Bionic ear**
- **Click**

The above options can also be increased.

4.2 – ZERO FLUSH

It is a fixed sequence of 144 zeros. It is placed between the data bits and EOM sequence, its placement at decoder helps to identify and separate the EOM sequence and data.

4.3 – EOM SEQUENCE

The eight-digit hexadecimal number; 4B65A5B2 or equivalently 32 bits represents the EOM sequence, identifying the end of message. EOM is a fixed sequence. The bits are transmitted with the most significant digit first.

The EOM message sequence is given as: [01001011011001011010010110110010]

4.4 – FILLER BITS

The data sequence + FLUSH bits +EOM should be equal to any of the following specified lengths, in case when the length is lesser, more zeros are appended at the end of EOM sequence to achieve any of the fixed interleaver length.

Data Rate (bps)	Interleaver Length in Frames					
	1	3	9	18	36	72
	Number of Input Data Bits per Block					
3200	384	1152	3456	6912	13824	27648
4800	576	1728	5184	10368	20736	41472
6400	768	2304	6912	13824	27648	55296
8000	960	2880	8640	17280	34560	69120
9600	1152	3456	10368	20736	41472	82944

TABLE 4-2 Input data block size in bits as a function of data rate and interleaver length

4.5 – ERROR CORRECTION CODING

Error correction coding or channel coding refers to the class of signal transformation designed to improve the communication performance by enabling the transmitted signals to better withstand the effects of various channel impairments, such as noise, interference and fading. The encoding process provides the coded signal with better distance properties than those of uncoded signals.

Channel coding can be categorized into two study areas:

- **Waveform or signal design coding**
- **Structured sequence coding**

Waveform coding deals with the transformation of waveforms into a better waveform format to make the detection process less subject to errors.

Structured sequences deals with transforming data sequences by introducing redundant bits, making them have structured redundancy. Redundant bits are used to detect and correct errors.

4.5.1 – TYPES OF ERROR CONTROL

There are two types of error control techniques:

4.5.1.1 – ARQ (AUTOMATIC REPEAT REQUEST)

It includes error detection and retransmission, utilizing parity bits. (Redundant bits added to the data). The receiving terminal does not correct the errors in this case; instead it simply requests the transmitter to retransmit the data. Thus for ARQ we need a two way link

4.5.1.2 – FEC (FORWARD ERROR CORRECTION)

It is a one-way link in which the receiver detects an error and corrects it using parity or redundant bits, designed for the purpose of both detection and correction.

FEC encoding can be implemented using three types of coding

- **Block codes**
- **Convolution codes**
- **Serial concatenation of block coding and convolution coding**
- **Turbo codes (most latest having qualities of both block and convolution codes)**

Every type of coding has its merits and demerits.

4.5.1.3 – TYPE OF ERROR CONTROL IMPLEMENTED IN OUR PROJECT

We have implemented forward error correction coding algorithm. The modem specifications also provide features which are critical in developing an efficient ARQ protocol, implemented on higher layers

4.5.2 – CONVOLUTION CODES

Elias first introduced convolutional codes in 1955, which offer an alternative to block codes for transmission over a noisy channel. Convolution coding can be applied to a continuous input stream (which cannot be done with block codes), as well as blocks of data. In fact, a convolution encoder can be viewed as a finite state machine. It generates a coded output data stream from an input data stream.

It is usually composed of one or more shift registers and a network of XOR (Exclusive-OR) gates. The stream of information bits flows in to the shift register from one end and is shifted out at the other end. XOR gates are connected to some stages of the shift registers as well as to the current input to generate the output.

4.5.2.1 – CONVOLUTIONAL ENCODER IN OUR PROJECT

We have implemented full tail biting ½ rate convolution encoding with 3/2 rate puncturing to get ¾ rate punctured data. We call it block coding as it processes blocks of data.

4.5.2.2 – FUNCTIONAL DETAILS

To begin encoding each block of input data, following steps are executed,

- **The encoder is preloaded by shifting in the first six input data bits without taking any output bits.**
- **These six input bits are temporarily saved in a buffer so that they can be used to flush the encoder later.**
- **The first two coded output bits are taken after the seventh bit has been shifted in, and are defined to be the first two bits of the resulting block code.**
- **After the last input data bit has been encoded, the first six saved data bits are encoded separately.**
- **Similarly, the flush bits and EOM sequence are encoded separately.**
- **The six saved data bits are encoded by shifting them into the encoder one at a time, beginning with the earliest of the six.**
- **The encoding thus continues by taking the two resulting coded output bits as each of the saved six bits is shifted in. These encoded bits shall be the final bits of the resulting (unpunctured) block code. Prior to puncturing, the resulting block code has exactly twice as many bits as the input information bits.**

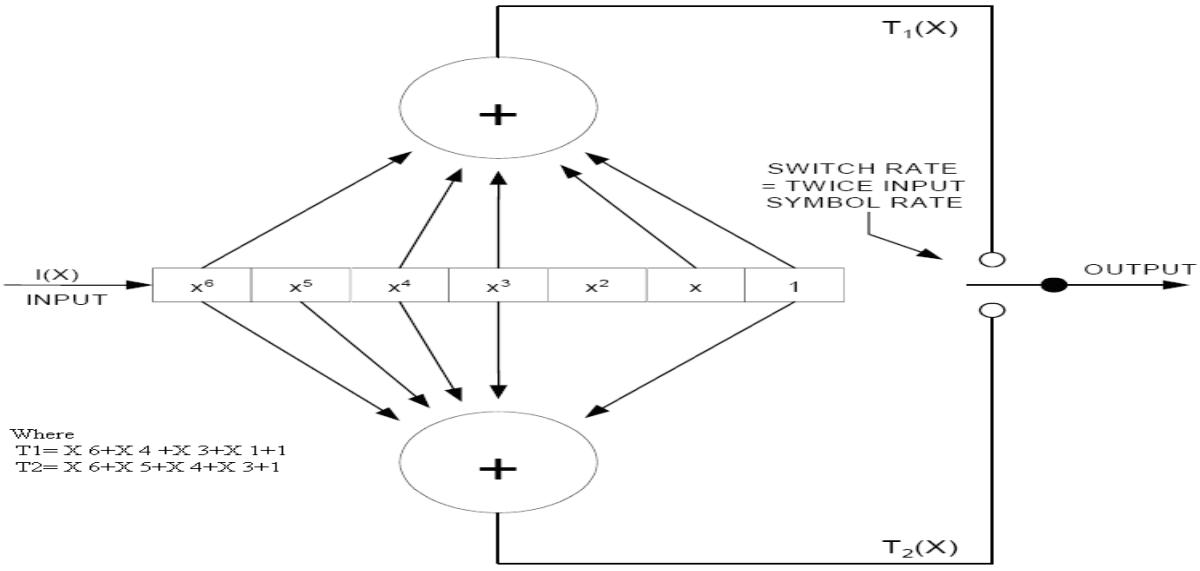


FIGURE 4-2 Encoder Diagram

definition	Symbol	value
input number	K	1
output number	N	2
encoder rate	K/N	1/2
constrain length	L	7
generator	G	[1 1 1 1 0 0 1;
sequence		1 0 1 1 0 1 1];

TABLE 4-3 Encoder Parameter Specifications

4.5.3 – PUNCTURING

In coding theory puncturing is defined the process of removing some of the parity bits after encoding with a code. The benefit of puncturing is that we can send lesser number of bits without losing information, for the same data. This helps us in achieving higher data rates.

We define encoder rates 1/2, 1/3, 1/n obtained as a process of convolutional or block encoding as mother codes. The feature of mother codes is that no matter how many bits are lost, as long as 1 per n coded bit is detected, the bit is saved. This does not affect the decoding process in the receiver side; dummy bit will be added to let decoder work normally. Combining different mother codes, we can produce different rates of puncturing codes, the advantage being to use some simple coding structure repetitively (for example, use only one 1/2 coding hardware),thus the puncturing considerably increases the flexibility of the system without significantly increasing its complexity. The BER for each input is different for puncturing codes.

A predefined pattern of puncturing has been defined, the bits after being encoded are punctured The puncturing mask is 111001, out of six bits two bits at the position of '4' and '5' will not be sent, which literally means that we are getting 3/2 rate code i.e. with every two input bits we are getting three output bits. In effect the coding process and puncturing will result into ½* 3/2 = ¾ code. Punctured data is then send to the interleaver. ***We***

introduce generalized tail-biting encoding as a means to improve the rate deficiency caused by zero-tail convolutional encoding.

4.6 – INTERLEAVER

The block interleaver used is designed to separate neighboring bits in the punctured block code as far as possible over the span of the interleaver with the largest separations resulting for the bits that were originally closest to each other. Because of the 30 different combinations of data rates and interleaver lengths, there is a more flexible interleaver structure.

4.6.1 – INTERLEAVER SIZE IN BITS

The interleaver shall consist of a single dimension array, numbered from 0 to its size in bits. The array size shall depend on both the data rate and interleaver length selected as shown in table 4-4

Data Rate (bps)	Interleaver Length in Frames					
	1	3	9	18	36	72
	Interleaver Size in Bits					
3200	512	1536	4608	9216	18,432	36,864
4800	768	2304	6912	13,824	27,648	55,296
6400	1024	3072	9216	18,432	36,864	73,728
8000	1280	3840	11,520	23,040	46,080	92,160
9600	1536	4608	13,824	27,648	55,296	110,592

TABLE 4-4 Interleaver size in bits as a function of data rate and interleaver length.

4.6.2 – INTERLEAVER LOAD

The punctured block code bits shall be loaded into the interleaver array beginning with location 0. The location for loading each successive bit shall be obtained from the previous location by incrementing by the Interleaver Increment Value - specified in table 4-5 - modulo the Interleaver Size

in Bits. Defining the first punctured block code bit to be B(0), then the load location for B(n) is given by:

Load Location = (n * Interleaver Increment Value)Modulo(Interleaver Size in Bits)

Thus for 3200 bps, with a one frame interleaver (512 bit size with an increment of 97), the first 8 interleaver load locations are: 0, 97, 194, 291, 388, 485, 70, and 167.

Data Rate (bps)	Interleaver Length in Frames					
	1	3	9	18	36	72
	Interleaver Increment Value					
3200	97	229	805	1393	3281	6,985
4800	145	361	1045	2089	5137	10,273
6400	189	481	1393	3281	6985	11,141
8000	201	601	1741	3481	8561	14,441
9600	229	805	2089	5137	10,273	17,329

TABLE 4-5 Interleaver increment value as a function of data rate and interleaver length.

These increment values have been chosen to ensure that the combined cycles of puncturing and assignment of bit positions in each symbol for the specific constellation being used, is the same as if there had been no interleaving. This is important, because each symbol of a constellation contains strong and weak bit positions, except for the lowest data rate. Bit position refers to the location of the bit, ranging from MSB to LSB, in the symbol mapping. A strong bit position is one that has a large average distance between all the constellation points where the bit is a 0 and the closest point where it is a 1. Typically, the MSB is a strong bit and the LSB a weak bit. An interleaving strategy that did not evenly distribute these bits in the way they occur without interleaving could degrade performance.

4.7 – MODIFIED GRAY DECODER (MGD)

MGD is used for data rates 3200 bps and 4800 bps, when PSK modulation is used for higher data rates it is by passed. Following tables are used to carry out the function.

Input bits		
First bit	Last bit	Modified-Gray decoded value
0	0	00
0	1	01
1	0	11
1	1	10

TABLE 4-6 Modified Gray Decoding for 3200 bps

Input bits			
First bit	Middle bit	Last bit	Modified Gray decoded value
0	0	0	000
0	0	1	001
0	1	0	011
0	1	1	010
1	0	0	111
1	0	1	110
1	1	0	100
1	1	1	101

TABLE 4-7 Modified Gray Decoding for 4800 bps

4.8 – SYMBOL FORMATION

4.8.1 – DATA SYMBOLS

For data symbols, the modulation used shall depend upon the data rate. Table 4-8 specifies the modulation that shall be used with each data rate.

Data Rate (bps)	Modulation
3200	QPSK
4800	8PSK
6400	16QAM
8000	32QAM
9600	64QAM
12800	64QAM

TABLE 4-8 Modulation used to obtain each data rate

4.8.1.1 – PSK DATA SYMBOLS

For the PSK constellations, a distinction is made between the data bits and the symbol number for the purposes of scrambling the QPSK modulation to appear as 8PSK, on-air. Scrambling is applied as a modulo 8 addition of a scrambling sequence to the 8PSK symbol number. Transcoding is an operation which links a symbol to be transmitted to a group of data bits.

4.8.1.1.1 – QPSK SYMBOL MAPPING

For the 3200 bps user data rate, transcoding shall be achieved by linking one of the symbols specified in table 4-9 to a set of two consecutive data bits (dibit).

Dibit	Symbol
00	0
01	2
11	4
10	6

TABLE 4-9 Transcoding for 3200 bps

In this table, the leftmost bit of the dibit shall be the older bit; i.e., fetched from the interleaver before the rightmost bit.

4.8.1.1.2 – 8PSK SYMBOL MAPPING

For the 4800 bps user data rate, transcoding shall be achieved by linking one symbol to a set of three consecutive data bits (tribit) as shown in table 4-10. In this table, the leftmost bit of the tribit shall be the oldest bit; i.e., fetched from the interleaver before the other two, and the rightmost bit is the most recent bit.

Tribit	Symbol
000	1
001	0
010	2
011	3
100	6
101	7
110	5
111	4

TABLE 4-10 Transcoding for 4800 bps

4.8.1.2 – QAM DATA SYMBOLS

For the QAM constellations, no distinction is made between the number formed directly from the data bits and the symbol number. Each set of 4 bits (16QAM), 5 bits (32QAM) or 6 bits (64QAM) is mapped directly to a QAM symbol. For example, the four bit grouping 0111 would map to symbol 7 in the 16QAM constellation while the 6 bits 100011 would map to symbol 35 in the 64QAM constellation. Again, in each case the leftmost bit shall be the oldest bit, i.e. fetched from the interleaver before the other bits, and the rightmost bit is the most recent bit. The mapping of bits to

symbols for the QAM constellations has been selected to minimize the number of bit errors incurred when errors involve adjacent signaling points in the constellation.

4.8.2 – KNOWN DATA AND SYNCHRONIZATION PREAMBLE SYMBOLS

Known data and synchronization preamble are always 8-PSK modulated thus table 4-10, 8-PSK transcoding table is used for bits to symbol conversion.

4.9 – KNOWN DATA

Known data is classified as data which is known both at transmitter and receiver. Such known sequences are very important in communication systems as they serve following purposes:-

- **Is used for synchronization (timing and carrier recovery)**
- **Doppler offset[1] removal**
- **Can be used for Automatic gain control[2]**
- **For informed channel[3] equalization**
- **Give information about data rate and interleaver settings which are critical for signal processing in transceiver like descrambling, symbol mapping, decoding etc such synchronization sequences are known as self-identifying preambles that is preambles having embedded information of interleaver settings and data rate.**
- **The above mentioned property can also be stated in terms of "auto baud" feature, a unique property of our modem. It lets the modem choose the modulation scheme**

[1] Phase offset cause during transmission
[2] The AGC is a circuit that keeps the receiver in its linear operating range by measuring the overall strength of the signal and automatically adjusting the gain of the receiver to maintain a constant level of output. The strength of the signal received varies widely, depending on the range to the transmitter, signal path attenuation, and so on. When the signal is strong, the gain is reduced by AGC, and when weak, the gain is increased, or allowed to reach its normal maximum.
[3] Type of equalization in which a priori known sequence, both at transmitter and receiver, is used for equalization.

and data rate, depending on channel conditions. If the channel conditions are bad transmission will be done at lower data rates whereas if channel conditions are good higher modulation schemes can be used.

MIL-STD-110B has predefined known sequence consisting of:-

- **Initial synchronization preamble**
- **Mini probes**
- **Re-inserted synchronization preamble**

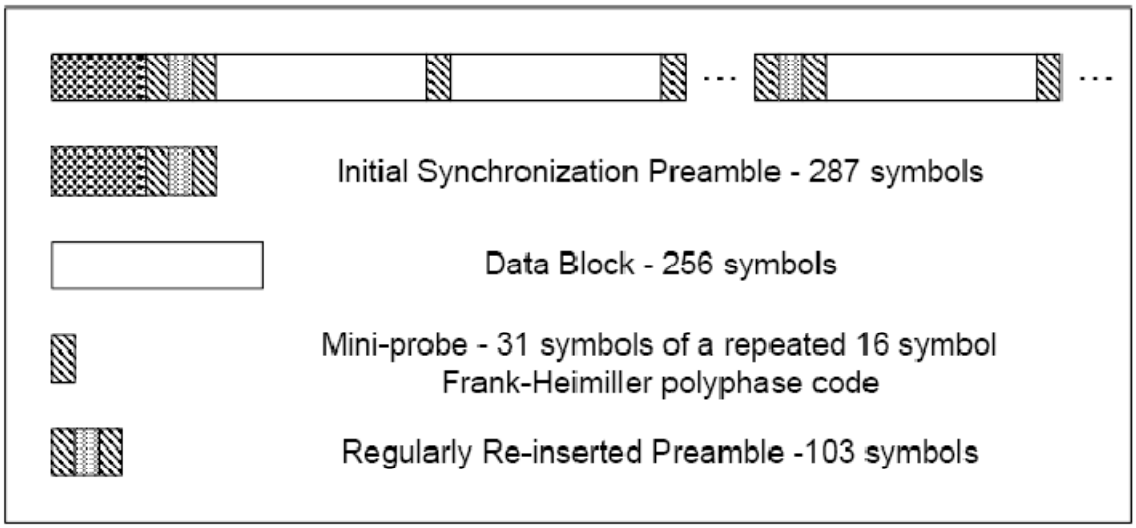


Figure 4-3 Frame structure for all waveforms.

4.9.1 – SYNCHRONIZATION PREAMBLE

The synchronization preamble is used for rapid initial synchronization. The transmitting modem shall send the required synchronization preamble sequence to achieve time and frequency synchronization with the receiving modem. In our modem, the duration of the synchronization preamble phase shall correspond to the exact time required to load the selected interleaver matrix with one block of data. During this phase, switch S1 shall be in the UNKNOWN DATA position, encode and load interleave functions shall be active as the modem begins accepting data from the data terminal equipment (DTE). Whereas switches S2 and S3 shall be in the "sync" position, first generating and then scrambling synchronization preamble, before sending.

The synchronization preamble consists of two parts. The first part consists of at least N blocks of 184 8-PSK symbols to be used exclusively for radio and modem AGC (automatic gain control). The value of N is configurable to range from values of 0 to 7 (for N=0 this first section is not sent at all, larger values of N can be selected when transmission conditions are bad) .These 184 symbols are formed by taking the complex conjugate of the first 184 symbols of the sequence specified for the second section.

The second section consists of 287 symbols. The first 184 symbols are intended exclusively for synchronization and Doppler offset removal purposes while the final 103 symbols, which are common with the reinserted preamble, also carry information regarding the data rate and interleaver settings.

The synchronization preamble shall be as shown below

1, 5, 1, 3, 6, 1, 3, 1, 1, 6, 3, 7, 7, 3, 5, 4, 3, 6, 6, 4, 5, 4, 0,
2, 2, 2, 6, 0, 7, 5, 7, 4, 0, 7, 5, 7, 1, 6, 1, 0, 5, 2, 2, 6, 2, 3,
6, 0, 0, 5, 1, 4, 2, 2, 2, 3, 4, 0, 6, 2, 7, 4, 3, 3, 7, 2, 0, 2, 6,
4, 4, 1, 7, 6, 2, 0, 6, 2, 3, 6, 7, 4, 3, 6, 1, 3, 7, 4, 6, 5, 7, 2,
0, 1, 1, 1, 4, 4, 0, 0, 5, 7, 7, 4, 7, 3, 5, 4, 1, 6, 5, 6, 6, 4, 6,
3, 4, 3, 0, 7, 1, 3, 4, 7, 0, 1, 4, 3, 3, 3, 5, 1, 1, 1, 4, 6, 1, 0,
6, 0, 1, 3, 1, 4, 1, 7, 7, 6, 3, 0, 0, 7, 2, 7, 2, 0, 2, 6, 1, 1, 1,
2, 7, 7, 5, 3, 3, 6, 0, 5, 3, 3, 1, 0, 7, 1, 1, 0, 3, 0, 4, 0, 7, 3,

0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4, 2, 0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4,
2,

(D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0 + 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0) Modulo 8
(D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1 + 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0) Modulo 8
(D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2 + 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0) Modulo 8

6,
4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0, 6, 4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0

TABLE 4-11 Synchronization Preamble

where the data symbols D0, D1, and D2 take one of 30 sets of values chosen from table given below to indicate the data rate and interleaver settings.

	Interleaver Length in Frames (256 Symbol Data Blocks)					
Data Rate (bps)	1	3	9	18	36	72
3200	0,0,4	0,2,6	0,2,4	2,0,6	2,0,4	2,2,6
4800	0,6,2	0,4,0	0,4,2	2,6,0	2,6,2	2,4,0
6400	0,6,4	0,4,6	0,4,4	2,6,6	2,6,4	2,4,6
8000	6,0,2	6,2,0	6,2,2	4,0,0	4,0,2	4,2,0
9600	6,0,4	6,2,6	6,2,4	4,0,6	4,0,4	4,2,6
12800	6,6,2*	reserved	reserved	reserved	reserved	reserved

* For 12800 bps 1 frame interleaver shall be interpreted as no interleaving

TABLE 4-12 D0, D1, D2 8 PSK symbol values as a function of data rate and interleaver length.

The mapping chosen to create table 4-12 uses 3 bits each to specify the data rate and interleaver length. The 3 data rate bits are the 3 most significant bits (MSB) of 3 dibit symbols and the interleaver length bits are the least significant bits (LSB). The phase of the Barker code[4] is determined from the 3 resulting dibit words using the dibit transcoding table.

The 3 bit data rate and interleaver length mappings are shown in table 4-13. Note that the transcoding has the effect of placing the 3 interleaver length bits in quadrature with the 3 data rate bits.

[4] A Barker code is a string of digits a_i of length l>=2 such that

$$\left| \sum_{i=1}^{l-k} a_i \, a_{i+k} \right| \leq 1$$

for all $1 \leq k < l$. Barker codes are used for pulse compression of signals. There are Barker codes of lengths upto 2, 3, 4, 5, 7, 11, and 13 exists.

Data Rate	3 Bit Mapping
reserved	000
3200	001
4800	010
6400	011
8000	100
9600	101
12800	110
reserved	111

Interleaver Length	3 Bit Mapping	Name
illegal: see C.5.2.1.2	000	
1 Frame	001	Ultra Short (US)
3 Frames	010	Very Short (VS)
9 Frames	011	Short (S)
18 Frames	100	Medium (M)
36 Frames	101	Long (L)
72 Frames	110	Very Long (VL)
illegal: see C.5.2.1.2	111	

TABLE 4-13 Bit patterns for specifying data rate and interleaver length.

The combinations are made by alternatively taking one bit from each set like if we want to specify 3200 data rate with interleaver length of 1 frame, the bit pattern will be 000011.

4.9.2 – REINSERTED PREAMBLE

The reinserted preamble is used to facilitate acquisition of an ongoing transmission (acquisition on data).It is identical to the final 72 symbols of the synchronization preamble. In fact, the final 103 symbols are common between the synchronization preamble and the contiguous block consisting of the reinserted preamble and the mini-probe which immediately precedes it. The 103 symbols of known data (including the 31 mini-probe symbols of the preceding data frame) are shown below,

0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4, 2, 0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4,
2,

($D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0, D_0$ + 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0) Modulo 8
($D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1, D_1$ + 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0) Modulo 8
($D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2, D_2$ + 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0) Modulo 8

6,
4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0, 6, 4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0;

TABLE 4-14 Reinserted Preamble

where the data symbols D0, D1, and D2 again take one of 30 sets of values chosen from table 2 to indicate the data rate and interleaver settings as described in the synchronization preamble section above.

3.9.3 – MINI PROBE

Channel probes (KNOWN DATA), are the training bits reserved for channel equalization by the distant receiver modem. These occur during the data phase of the modem.

Mini-probes, 31 symbols in length are inserted following every 256 symbol data block and at the end of each preamble (where they are considered to be part of the preamble). Using the 8PSK symbol mapping, each mini-probe is based on the repeated Frank-Heimiller sequence. The sequence that is used, specified in terms of the 8PSK symbol numbers, is given as:

4.9.3.1 – POSITIVE MINIPROBE

[0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4, 2, 0, 0, 0, 0, 0, 2, 4, 6, 0, 4, 0, 4, 0, 6, 4.] .This mini-probe is designated as (+).

4.9.3.2 – NEGATIVE MINIPROBE

The phase inverted version of this is:

[4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0, 6, 4, 4, 4, 4, 4, 6, 0, 2, 4, 0, 4, 0, 4, 2, 0] and mini-probes using this sequence are designated (-), as the phase of each symbol has been rotated 180 degrees from the (+). There are a total of 73 mini-probes for each set of 72 data blocks. For convenience, each mini-probe are sequentially numbered, with mini-probe 0 being defined as the last 31 symbols of the preceding (reinserted) preamble, mini-probe 1 following the first data block after a (reinserted) preamble. Mini-probe 72 follows the 72nd data block, and is also the first 31 symbols of the next 103 symbol

reinserted preamble. Mini-probes 0 and 72 have been defined as part of the reinsertion preamble to have the signs (-) and (+) respectively. The data rate and interleaver length information encoded into the synchronization and reinserted preambles shall also be encoded into mini-probes 1 through 72. These 72 mini-probes are grouped into four sets of 18 consecutive mini-probes (1 to 18, 19 to 36, 37 to 54, and 55 to 72). The 256 symbol data block that immediately follows the 18th mini-probe, in each of the first three sets, is also the 1st data block of an interleaver block with frame lengths of 1, 3, 9, and 18. The length 36 interleaver block begins after the second set, and a reinserted preamble begins after the fourth set. This structure permits data to begin to be demodulated as soon as the interleaver boundary becomes known.

Each 18 mini-probe sequence shall consist of seven. signs, a + sign, followed by six sign values that are dependent on the data rate and interleaver length, three sign values that specify which of the four sets of 18 mini-probes it is, and then finally a + sign. For the fourth set, this final + sign (mini-probe 72) is also the initial mini-probe of the next reinserted preamble (which uses the + phase).

Pictorially, this length 18 sequence is: - - - - - - - +**S0 S1 S2 S3 S4 S5 S6 S7 S8 +, where the first six sign values are defined in table 4-15. Note that these 6 bit patterns (+ is a 0) correspond to the concatenation of the 3 bit mappings from table 4-13 for the data rate (S0 S1 S2) and the interleaver length (S3 S4 S5). The final three sign values which specify the mini-probe set (count) are defined in table 4-16.**

Data Rate (bps)	Interleaver Length in Frames (256 Symbol Data Blocks)					
	1	3	9	18	36	72
3200	++-++-	++-+-+	++-+--	++--++	++--+-	++---+
4800	+-+++-	+-++-+	+-++--	+-+-++	+-+-+-	+-+--+
6400	+--++-	+--+-+	+--+--	+---++	+---+-	+----+
8000	-++++-	-+++-+	-+++--	-++-++	-++-+-	-++--+
9600	-+-++-	-+-+-+	-+-+--	-+--++	-+--+-	-+---+
12800	--+++-	N/A	N/A	N/A	N/A	N/A

TABLE 4-15 S0, S1, S2, S3, S4, S5 (sign) values as a function of data rate and interleaver setting.

Mini-probe set			
1 to 18	19 to 36	37 to 54	55 to 72
++-	+-+	+--	-++

TABLE 4-16: S6, S7, S8 (sign) values as a function of mini-probe set.

The 1st eight mini-probes in each set (- - - - - - - +) uniquely locate the starting point for the following nine S_i values. This is possible since the S_i sequences used contain at most, runs of four + or - phases. This makes it impossible for a sequence of 7 mini-probes with the same phase followed by one with a phase reversal to occur anywhere else except at the beginning of one of the 18 mini-probe sequences. Once this fixed 8 mini-probe pattern is located, the 0 or 180 degree phase ambiguity is also resolved so that the following 9 mini-probes can be properly matched to the data rate, interleaver length, and mini-probe set count. The entire mini probe sequence shall therefore be as follows:

[rp] - - - - - - - + S0 S1 S2 S3 S4 S5 S6 S7 S8 + - - - - - - - + S0 S1 S2 S3 S4 S5 S6 S7 S8 + - - - - - - - + S0 S1 S2 S3 S4 S5 S6 S7 S8 + - - - - - - - + S0 S1 S2 S3 S4 S5 S6 S7 S8 [rp]

Where the [rp] represents the 103 reinserted preamble symbols (includes mini-probes 72 and 0).

4.10 – SCRAMBLER

Scramblers are simple shift registers with XORing components. Scramblers are used for following two purposes

- **For security reasons so that the data cannot be deciphered easily as the sequence and structure of scrambler that is generating that sequence is known only at transmitter and receiver.**
- **Secondly to counteract the effects of channel noise and other impairments to signal by increasing the hamming distance between two adjacent data symbols.**

4.10.1 – SCRAMBLER AS IMPLEMENTED IN OUR PROJECT

In our project we have used a scrambler with shift register length of 9, initialized with bit sequence of 10000000. Scrambler, depending upon the data rate and type of modulation, generates the scrambling sequence, which is then treated with data symbols in some specific manner producing scrambled sequence. Synchronization preamble is always scrambled using PSK technique; in figure the generator producing scrambling sequence for synchronization preamble is shown as "Synchronization sequence randomizing generator", the other module labeled as "Data sequence randomizing generator" generates randomizing sequence for data.

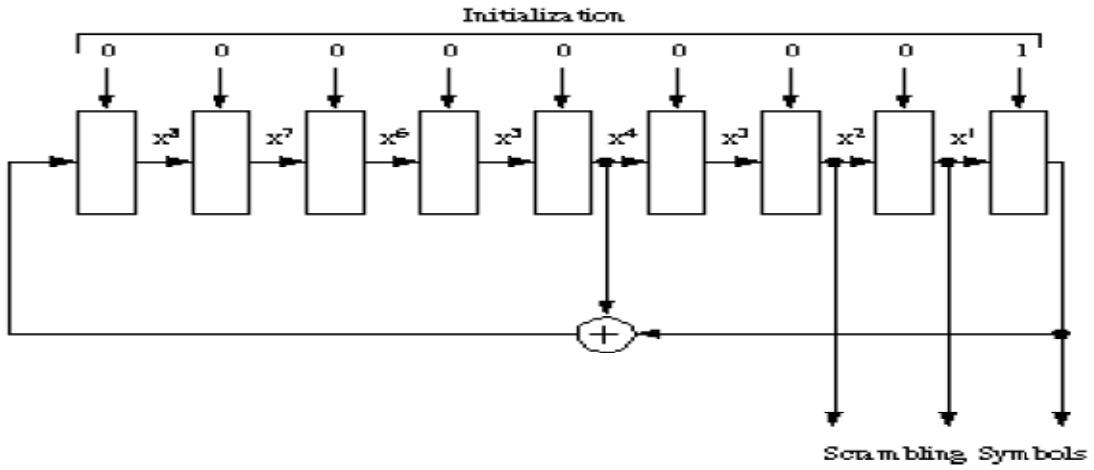


FIGURE 4-4 Scrambling sequence generator illustrating scrambling generator for 8PSK symbols

For 8PSK symbols (3200 bps and 4800 bps), the scrambling shall be carried out taking the modulo 8 sum of the numerical value of the binary triplet consisting of the last (rightmost) three bits in the shift register, and the symbol number (transcoded value). For example, if the last three bits in the scrambling sequence shift register were 010 which has a numerical value equal 2, and the symbol number before scrambling was 6, symbol 0 would be transmitted since: (6+2) Modulo 8 = 0.

For 16QAM symbols, scrambling is done by XORing the 4-bit number consisting of the last (rightmost) four bits in the shift register with the symbol number. For example, if the last 4 bits in the scrambling sequence shift register were 0101 and the 16QAM symbol number before scrambling was 3 (i.e. 0011), symbol 6 (0110) would be transmitted.

For 32QAM symbols, scrambling is carried out by XORing the 5-bit number formed by the last (rightmost) five bits in the shift register with the symbol number.

For 64QAM symbols, scrambling is carried out by XORing the 6-bit number formed by the last (rightmost) six bits in the shift register with the symbol number.

After each data symbol is scrambled, the generator is iterated (shifted) the required number of times to produce all new bits (bit sequence) for use in scrambling the next symbol, the number of iterations for different data rates/modulation types are as follows:

- **3 iterations for 8PSK**
- **4 iterations for 16QAM**
- **5 iterations for 32QAM**
- **6 iterations for 64QAM.**

Since the generator is iterated after the bits are use, the appropriate number of bits from the initialization value of 100000000 shall therefore, scramble the first data symbol of every data frame.

4.11 – CONSTELLATIONS

Constellations are diagrammatic way of observing system performance and analyzing its robustness against errors and are used before the modulation stage of system.

4.11.1 – PSK SYMBOLS

The 3200 bps quadrature phase-shift keying (QPSK) constellation is scrambled to appear, on-air, as an 8PSK constellation.

For the PSK constellations, a distinction is made between the data bits and the symbol number for the purposes of scrambling the QPSK modulation to appear as 8PSK, on-air.

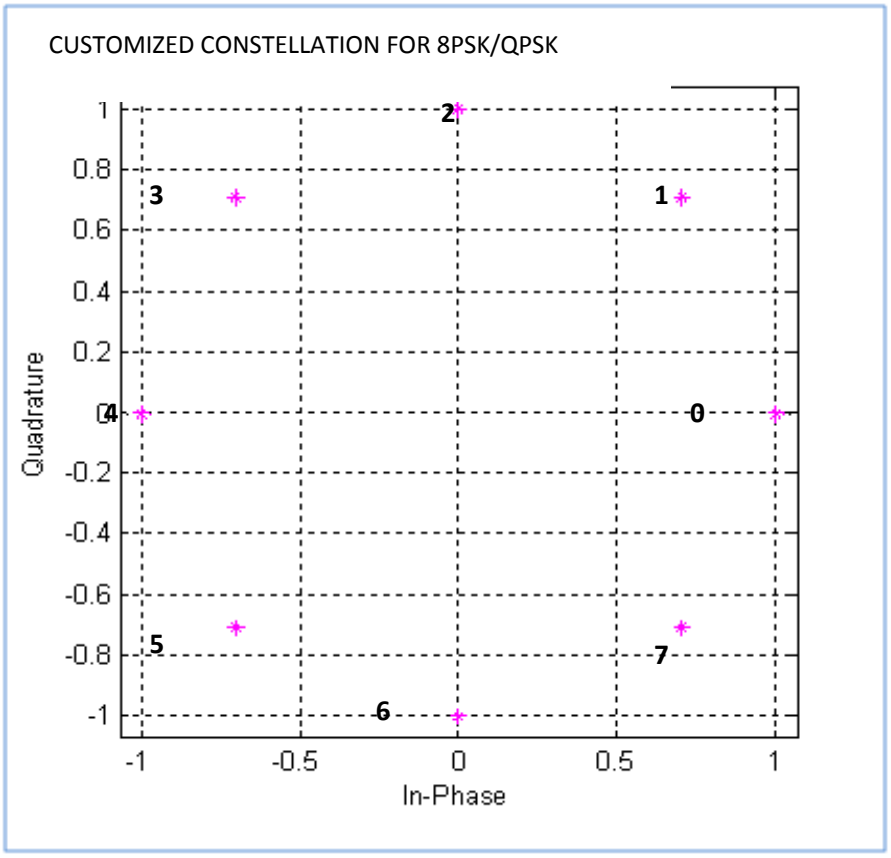


FIGURE 4-5 Constellation for 8PSK/QPSK

SYMBOL	INPHASE VALUE	QUADRATURE PHASE VALUE	PHASE
0	1	0	0
1	0.707	0.707	pi/4
2	0	1	pi/2
3	-0.707	0.707	3pi/4
4	-1	0	pi

5	**-0.707**	**-0.707**	**5pi/4**
6	**0**	**-1**	**3pi/2**
7	**0.707**	**-0.707**	**7pi/4**

TABLE 4-17 Phases of Symbols for 8PSK

4.11.2 – QAM DATA SYMBOLS

For the QAM constellations, no distinction is made between the number formed directly from the data bits and the symbol number. Each set, of 4 bits (16QAM), 5 bits (32QAM) or 6 bits (64QAM), is mapped directly to a QAM symbol. For example, the four bit grouping 0111 would map to symbol 7 in the 16QAM constellation while the 6 bits 100011 would map to symbol 35 in the 64QAM constellation. Again, in each case the leftmost bit shall be the oldest bit, i.e. fetched from the interleaver before the other bits, and the rightmost bit is the most recent bit. The mapping of bits to symbols for the QAM constellations has been selected to minimize the number of bit errors incurred when errors involve adjacent signaling points in the constellation.

4.11.2.1 – 16QAM CONSTELLATION

The constellation diagram and real and imaginary point mapping of symbols are shown below. This constellation contains an outer ring of 16 symbols and an inner square of 4 symbols.

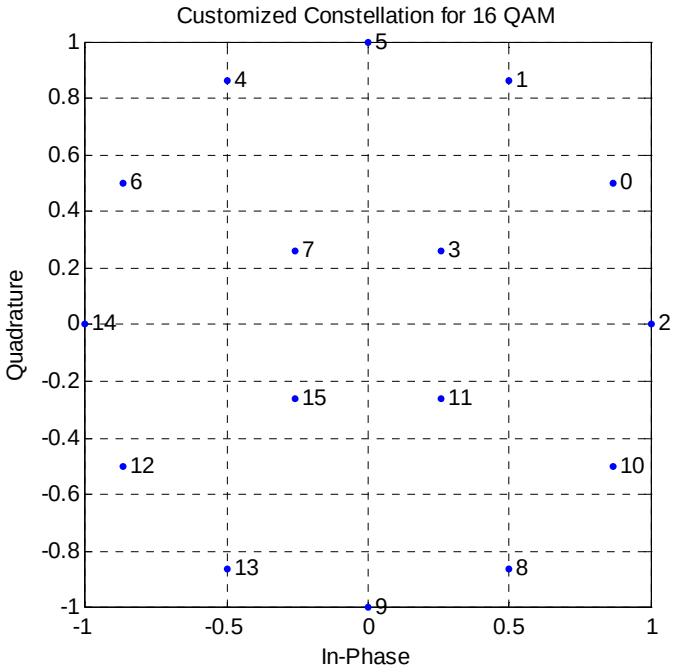


FIGURE 4-6 Constellation for 16QAM

4.11.2.2 – 32QAM CONSTELLATION

The constellation diagram and real and imaginary point mapping of symbols are shown below. This constellation contains an outer ring of 16 symbols and an inner square of 16 symbols.

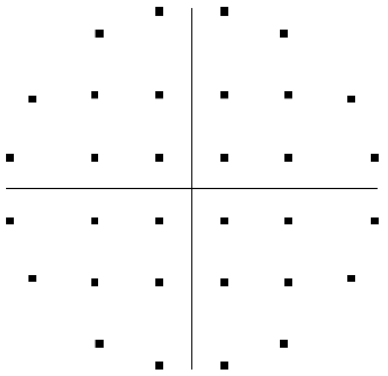


FIGURE 4-7 32QAM signaling constellation.

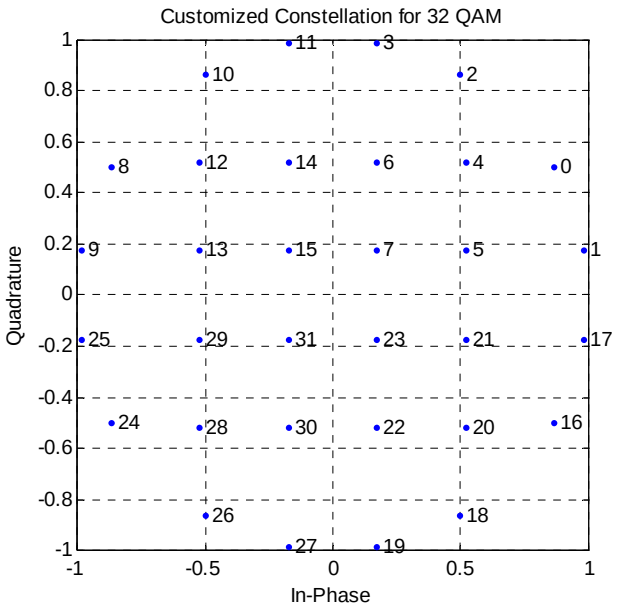


FIGURE 4-8 Customized 32QAM signaling constellation.

Symbol Number	In-Phase	Quadrature	Symbol Number	In-Phase	Quadrature
0	0.866380	0.499386	16	0.866380	-0.499386
1	0.984849	0.173415	17	0.984849	-0.173415
2	0.499386	0.866380	18	0.499386	-0.866380
3	0.173415	0.984849	19	0.173415	-0.984849
4	0.520246	0.520246	20	0.520246	-0.520246
5	0.520246	0.173415	21	0.520246	-0.173415
6	0.173415	0.520246	22	0.173415	-0.520246
7	0.173415	0.173415	23	0.173415	-0.173415
8	-0.866380	0.499386	24	-0.866380	-0.499386
9	-0.984849	0.173415	25	-0.984849	-0.173415
10	-0.499386	0.866380	26	-0.499386	-0.866380
11	-0.173415	0.984849	27	-0.173415	-0.984849
12	-0.520246	0.520246	28	-0.520246	-0.520246
13	-0.520246	0.173415	29	-0.520246	-0.173415
14	-0.173415	0.520246	30	-0.173415	-0.520246
15	-0.173415	0.173415	31	-0.173415	-0.173415

TABLE 4-18 In-phase and Quadrature components of each 32QAM symbol.

4.11.2.3 – THE 64QAM CONSTELLATION

The constellation points which shall be used for the 64QAM modulation are shown in figure below and specified in terms of their In-phase and quadrature components in table 4-19. This constellation is a variation on the standard 8 x 8 square constellation, which achieves better peak to-average without sacrificing the very good pseudo-Gray code properties of the square constellation.

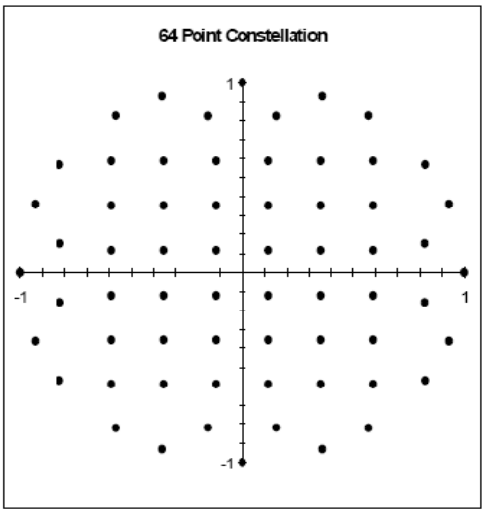


FIGURE 4-9 64QAM signaling constellation.

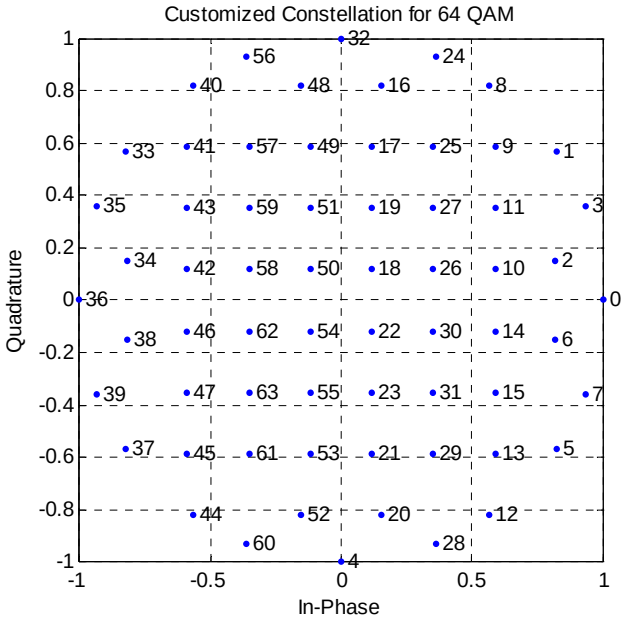


FIGURE 4-10 Customized 64QAM Constellation

Symbol Number	In-Phase	Quadrature	Symbol Number	In-Phase	Quadrature
0	1.000000	0.000000	32	0.000000	1.000000
1	0.822878	0.568218	33	-0.822878	0.568218
2	0.821137	0.152996	34	-0.821137	0.152996
3	0.932897	0.360142	35	-0.932897	0.360142
4	0.000000	-1.000000	36	-1.000000	0.000000
5	0.822878	-0.568218	37	-0.822878	-0.568218
6	0.821137	-0.152996	38	-0.821137	-0.152996
7	0.932897	-0.360142	39	-0.932897	-0.360142
8	0.568218	0.822878	40	-0.568218	0.822878
9	0.588429	0.588429	41	-0.588429	0.588429
10	0.588429	0.117686	42	-0.588429	0.117686
11	0.588429	0.353057	43	-0.588429	0.353057
12	0.568218	-0.822878	44	-0.568218	-0.822878
13	0.588429	-0.588429	45	-0.588429	-0.588429
14	0.588429	-0.117686	46	-0.588429	-0.117686
15	0.588429	-0.353057	47	-0.588429	-0.353057
16	0.152996	0.821137	48	-0.152996	0.821137
17	0.117686	0.588429	49	-0.117686	0.588429
18	0.117686	0.117686	50	-0.117686	0.117686
19	0.117686	0.353057	51	-0.117686	0.353057
20	0.152996	-0.821137	52	-0.152996	-0.821137
21	0.117686	-0.588429	53	-0.117686	-0.588429
22	0.117686	-0.117686	54	-0.117686	-0.117686
23	0.117686	-0.353057	55	-0.117686	-0.353057
24	0.360142	0.932897	56	-0.360142	0.932897
25	0.353057	0.588429	57	-0.353057	0.588429
26	0.353057	0.117686	58	-0.353057	0.117686
27	0.353057	0.353057	59	-0.353057	0.353057
28	0.360142	-0.932897	60	-0.360142	-0.932897
29	0.353057	-0.588429	61	-0.353057	-0.588429
30	0.353057	-0.117686	62	-0.353057	-0.117686
31	0.353057	-0.353057	63	-0.353057	-0.353057

TABLE 4-19 In-phase and Quadrature components of each 64QAM symbol.

4.12 - MODULATION

Modulation of a signal refers to a series of digital signal processing techniques applied on a signal including:

- **up sampling**
- **Pulse shaping**
- **Mixing**

In our project, the baud rate, for all symbols, is 2400 symbols-per-second. Phase-shift keying (PSK) and quadrature amplitude modulation (QAM) modulation techniques are used to modulate the baseband signal.

4.12.1 - UPSAMPLING

When we say we have up sampled the signal we mean that we have increased the number of samples per symbol or in case of untreated binary data number of samples per bit duration.

Up sampling is done on a signal for various reasons stated below:

- **To convert pulses of digital data into impulses.**
- **To make the number of samples of data equal to number of samples of carrier.**

As per the requirement we have up sampled our data which is in the form of symbols by the rate of four, meaning we have four samples per symbol. It is performed by inserting (N-1) zeros between the original data symbols ,where N is the desired rate of up sampling.

4.12.2 – PULSE SHAPING

Pulse shaping is done by filtering the signal through Nyquist filters. A Nyquist filter is one whose frequency transfer function can be represented by a rectangular function convolved with any real even symmetric frequency function and nyquist pulse is one whose shape can be represented by a sinc (t/T) function multiplied by another time function. Amongst the class of Nyquist filters, the most popular ones are raised cosine filters and root raise cosine filters. We have used root raise cosine

filters. For pulse shaping by *root raise cosine filtering* the load of pulse shaping is shared between transmitter and receiver

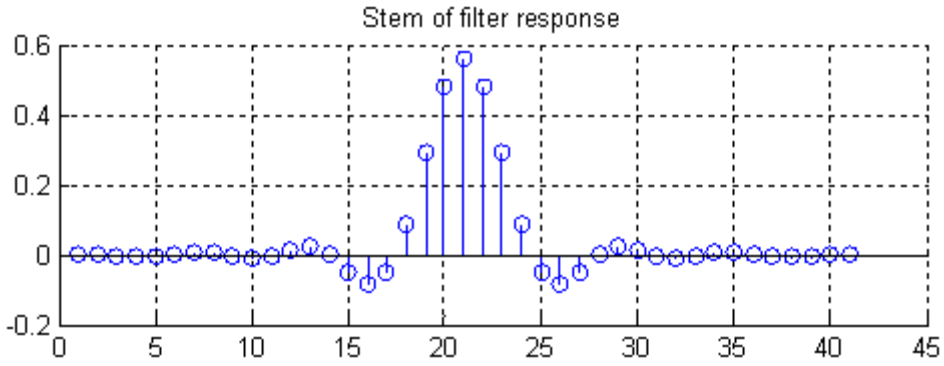


FIGURE 4-11 Response of Root Raise Cosine Filter

4.12.2.1 - PURPOSE

- **Pulse shaping is employed to reduce the bandwidth of the data impulses to some reasonably smaller bandwidth remaining within Nyquist limits.**
- **Pulse shaping is used to circumvent the effect of ISI due to channel induced distortions and filtering.**
- **The design priority of these filters is to maximize the SNR of known signal, in the presence of AWGN, by gathering the signal energy and giving peak at the sampling instants.**

4.12.3 – MIXING STAGE

Mixing also known as heterodyning is the frequency translation of low frequency signal to higher frequency, which is achieved by mixing high frequency carrier with low frequency baseband signal.

The sub carrier (Or pair of quadrature sub-carriers in the case of QAM) are centered at 1800 Hz. The phase of the quadrature sub-carrier relative to the In-phase carrier is 90 degrees. That is the in-phase sub-carrier is Cos (1800 Hz) and the quadrature sub-carrier is -Sin (1800 Hz).

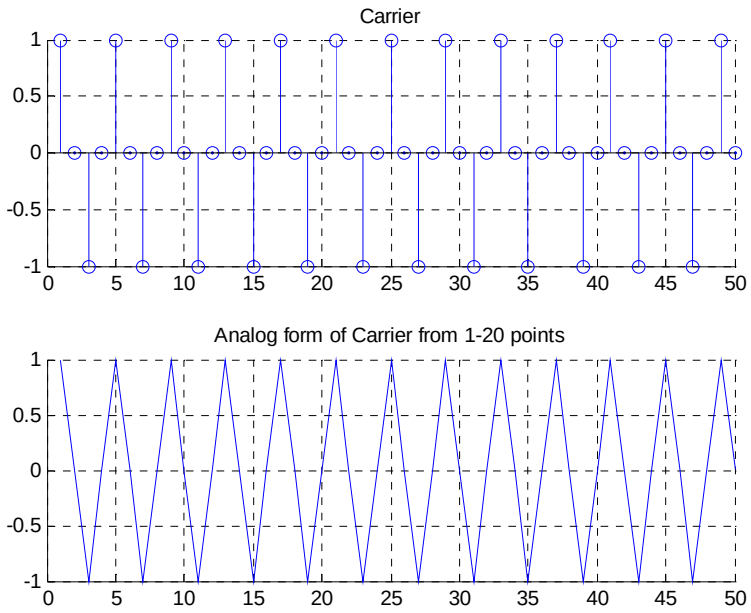


FIGURE 4-12 Carrier Generated

The baseband signal is mixed with the carrier to give translated signal.

Modulated signal=$e^{j*\text{theta}} * e^{j*\text{carrier theta}}$

where,

theta=baseband signal phase

carrier theta=phase of the carrier

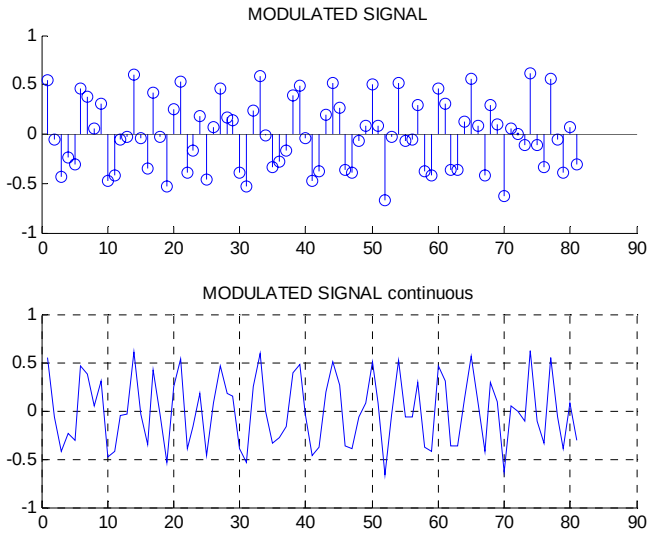


FIGURE 4-13 Modulated Signal to be Transmitted

4.13 – CHANNEL

- **On channel real part of signal is transmitted.**
- **Channel is modeled as a delay line filter having four coefficients**
- **Signal is filtered through this channel**
- **Measured AWGN (Added white Gaussian noise) is added to this signal.**
- **The SNR for the channel has been taken as 13db**

CHAPTER 5

RECIEVER

5.1 – RECEIVED SIGNAL

The received signal will be affected by the non-ideal + AWGN channel and will contain noise

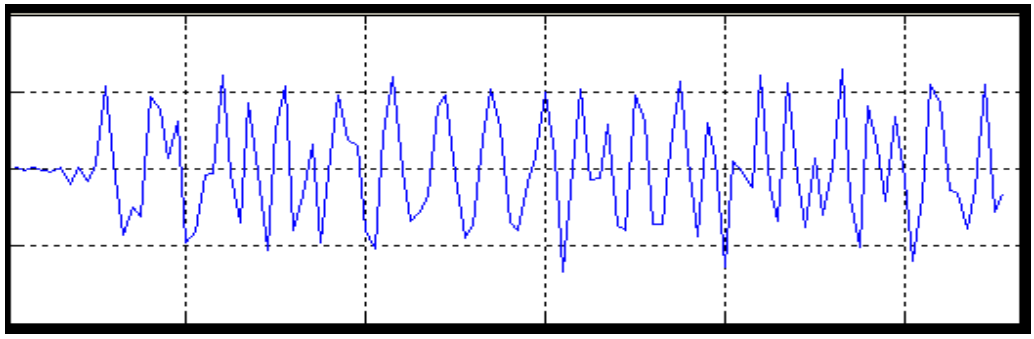


FIGURE 5-1 Received Signal with added Noise

5.2 – DEMODULATION

It consist of following four stages

- **Hilbert Transform**
- **Mixing with conjugate of carrier**
- **Pulse shaping (Matched filtering)**
- **Down sampling**

5.2.1 – HILBERT TRANSFORM

- **Hilbert transform is a digital processing technique by which we can reconstruct the imaginary part of the signal given that we have the real part.**
- **At receiving end the imaginary part of signal is reconstructed using Hilbert transform of the received real signal.**

5.2.2 – MIXING WITH CARRIER

The received signal is mixed with the conjugate of carrier in this step to recover the base band signal.

Signal = $e^{j*\text{theta}} * e^{j*\text{conj carrier theta}}$

theta = received signal phase

conj carrier theta = phase of the conjugate of carrier

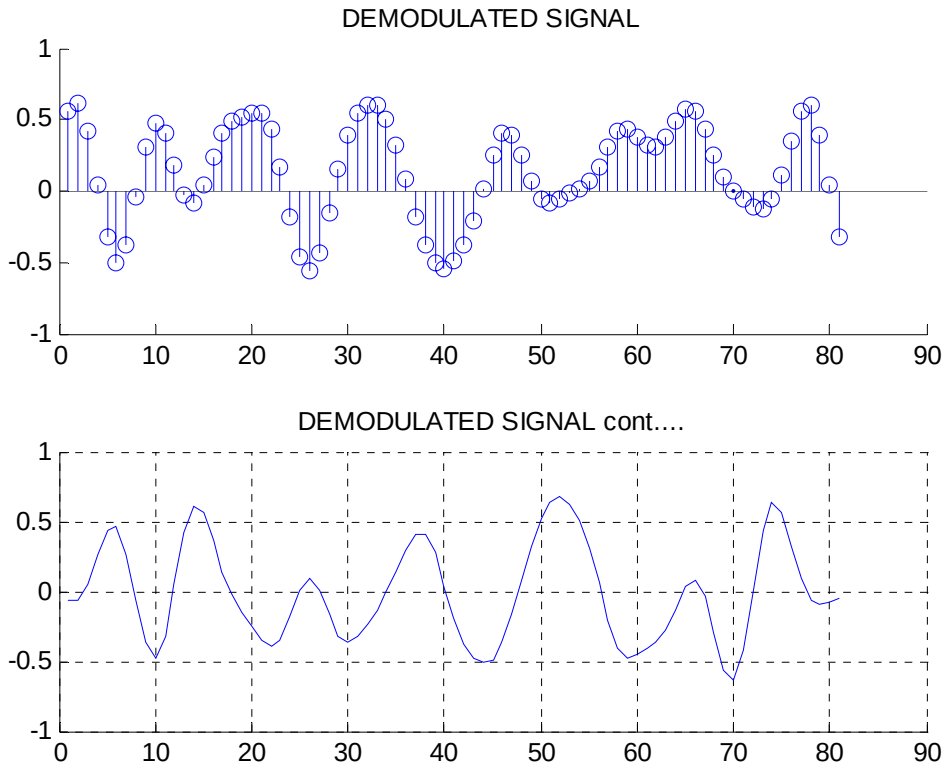


FIGURE 5-2 Demodulated Signal

5.2.3 – PULSE SHAPING (MATCHED FILTERING)

Base band signal is passed through a filter matched to the transmitting filter i.e. having the same impulse response as at transmitter. The transfer function of receiver filter is also root raise cosine, the

product of the transmitter and receiver functions yields a composite raised cosine system transfer function as shown below having zero crossing at the sampling instants of other symbols.

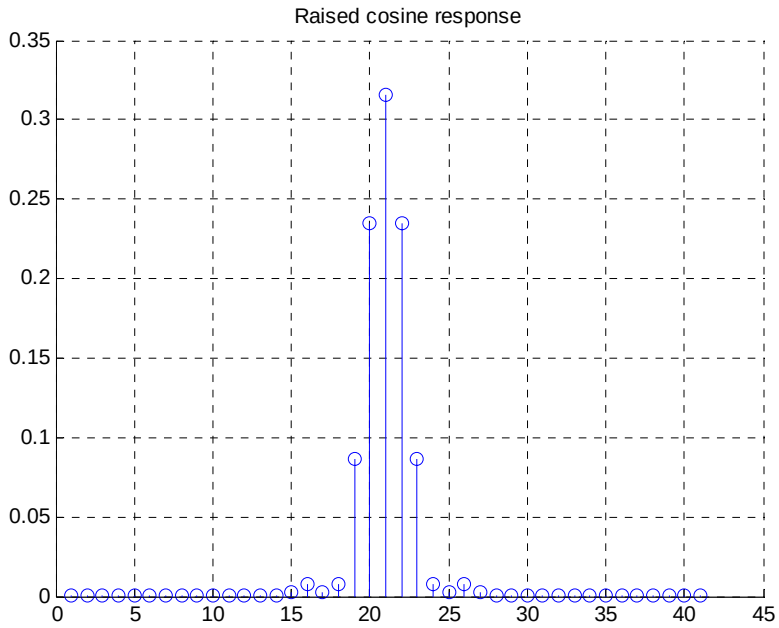


FIGURE 5-3 Response of Raised Cosine Filter

5.2.3.1 - Eye Diagram:

Here we introduced another parameter which shows system response to baseband signal, an eye digram.It is called an eye diagram because it resembles a human eye. The eye pattern is created as a result of superimposing the waveform in each signaling interval onto a single interval [0, T[5]]. The eye diagram qualitatively assesses the extent of ISI.As the eye closes, it means the ISI is increasing and as the eye opens, ISI is decreasing.

How many eye openings are going to be there is decided by the amplitude levels of a particular constellation. If there are M amplitude levels then there will be M-1 eye openings. When plotting eye diagrams of QAM constellation eye openings were very close because there is very little amplitude difference between two consecutive amplitude levels in QAM constellations.

[5] T= symbol duration

INSERT EYE DIAGRAM FROM SIR NOTES

FIGURE 5-4 Eye Diagram

- **The width of the opening indicates the time over which sampling for detection can be performed, with optimum sampling time corresponding to maximum eye opening.**
- **Range of time differences of zero crossing is a measure of timing jitter**
- **Range of amplitude differences is a measure of distortion caused by ISI.**
- **The length of opening depicts noise margin.**
- **Distance from the peak of eye opening to one end is a measure of sensitivity to timing error.**

5.2.3.1.1 – EYE DIAGRAM PLOTS

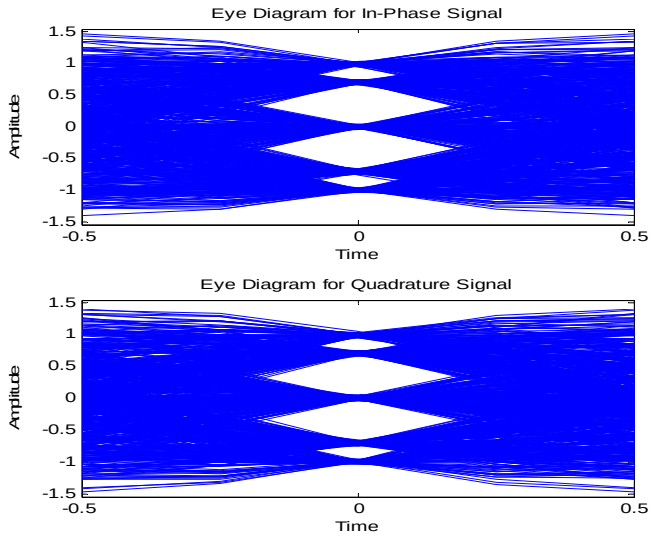


FIGURE 5-5 Eye Diagram for 8PSK

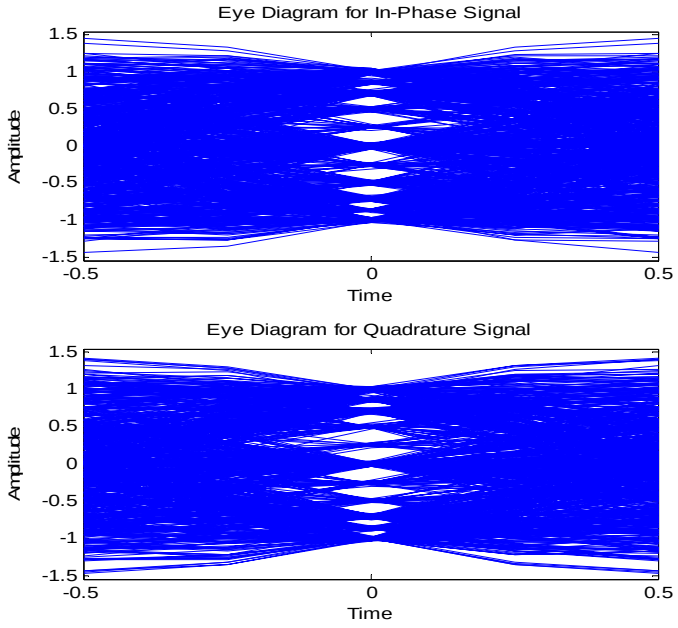


FIGURE 5-6 EYE DIAGRAM OF 16 QAM

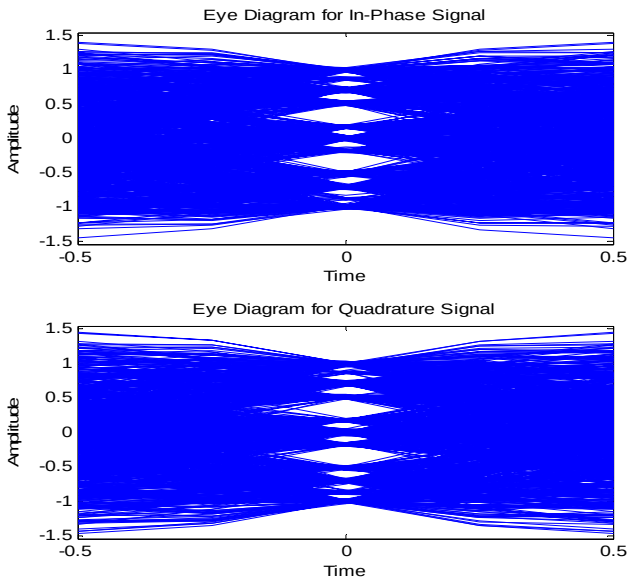


FIGURE 5-7 Eye Diagram for 32QAM

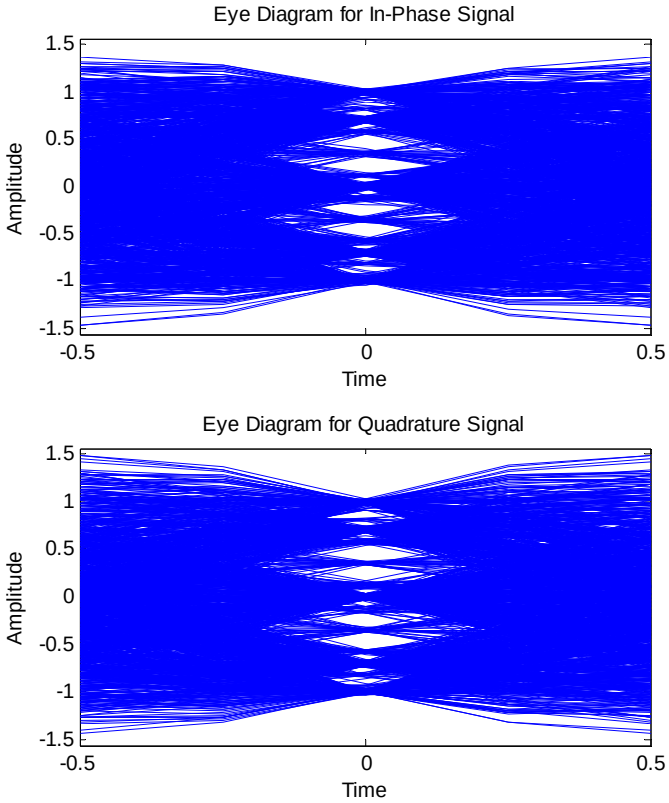
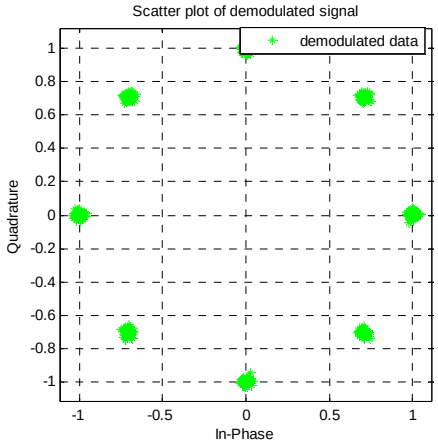


FIGURE 5-8 Eye Diagram for 64QAM

5.2.4. DOWN SAMPLING:

Filtered signal is down sampled go get one sample per symbol.



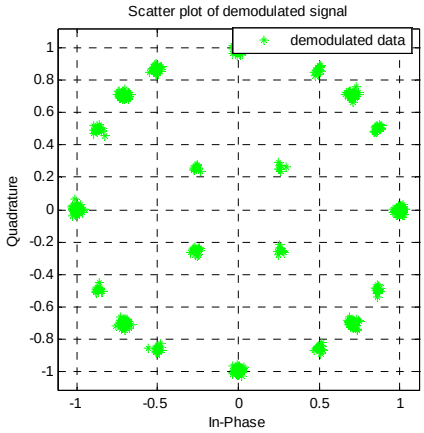
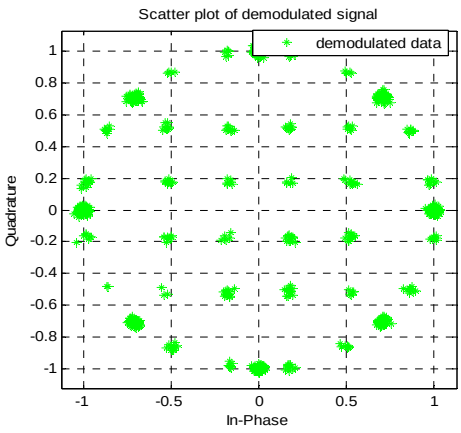


FIGURE 5-9 8PSK Constellation

FIGURE 5-10 16QAM Constellation



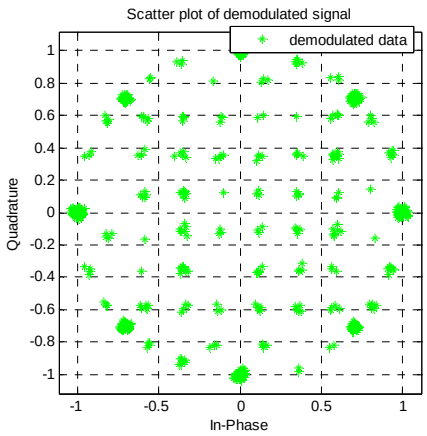


FIGURE 5-11 32QAM Constellation

FIGURE 5-12 64QAM Constellation

5.3 – DETECTION AND ESTIMATION OF SIGNALS

In a digital communication system, at the receiving end, following two main issues are of great importance.

- **Estimation**
- **Detection**

5.3.1 - ESTIMATION

5.3.1.2 - EQUALIZER

5.3.1.2.1 - BACKGROUND

DSP based equalizer systems have become ubiquitous in many diverse applications including voice, data and video communications via various transmission media. The effect of an equalization system is to compensate for the transmission-channel impairments such as frequency phase and amplitude distortions.

5.3.1.2.2 – INVERSE SYSTEM IDENTIFICATION

By placing the unknown system in series with your adaptive filter, your filter becomes the inverse of the unknown system when *e(k)* gets very small. As shown in the figure 5-13, the process requires a delay inserted in the desired signal *d(k)* path to keep the data at the summation synchronized. Adding the delay keeps the system causal.

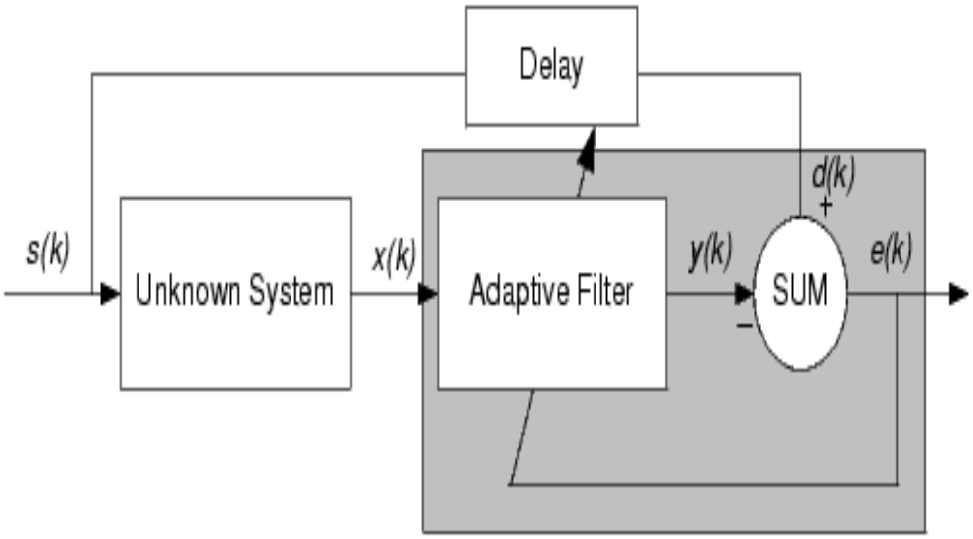


FIGURE 5-13 Block diagram of Inverse System Identi

Without the delay element, the adaptive filter algorithm tries to match the output from the adaptive filter *y(n)* to input data *x(n)* that has not yet reached the adaptive elements because it is passing through the unknown system. In essence, the filter ends up trying to look ahead in time.

As hard as it tries, the filter can never adapt: *e(n)* never reaches a very small value and your adaptive filter never compensates for the unknown system response. And it never provides a true inverse response to the unknown system. Including a delay equal to the delay caused by the unknown system prevents this condition.

In our project we used trained adaptive filters to serve as equalizer. In case of trained adaptive filters a known sequence is transmitted and a synchronized version of this signal is generated in the receiver where it is applied to the adaptive equalizer as the desired response dn. The tap weights of the equalizer are thereby adjusted in accordance with the applied algorithm. We made use of Least Mean Square (LMS) Algorithm.

5.3.1.2.3 – LEAST MEAN SQUARE (LMS) ALGORITHM

The Least Mean Squares (LMS) algorithm is the most widely used and the cheapest of the adaptive filtering algorithms. As any adaptive algorithm, it is based on optimizing a certain value. In the case of LMS algorithm, that value is the Mean Square Error (i.e. the mean square value of the error signal ($E[e_k^2]$). This is achieved by adjusting the values of the weights of FIR filter.

The Least Mean Square (LMS) algorithm tries to minimize the mean square between the input signal and the filtered output signal. The adaptive filter coefficient change according to the following equation,

$$W_n = w_{n-1} + \mu * x_n * e[n]$$

where μ is the step size, x[n] is the measured input signal and e[n] is the error between the desired signal and the filtered signal. The step size μ determines the convergence speed of the algorithm and the optimized value is related to the Eigen value of the input signal correlation matrix.

5.3.1.2.3.1 – BLOCK DIAGRAM

The block diagram of LMS algorithm is shown in figure 5-14

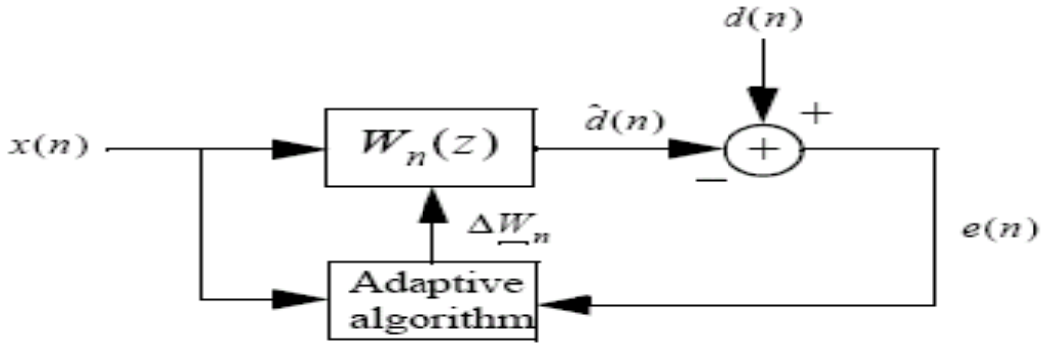


FIGURE 5-14 Block diagram of LMS algorithm

5.3.1.2.3.2 – STABILITY OF LMS ALGORITHM

It can be shown that starting with an arbitrary initial weight vector, the LMS algorithm will converge in the mean and will remain stable as long as the step size (μ) is in the range

$$0 < \mu < 1$$

which is an easy boundary to calculate, Within this margin, the larger the value of μ, the faster the convergence but the less the stability around the minimum value. On the other hand, the smaller the value is, the slower the convergence but will be more stable around the optimum value.

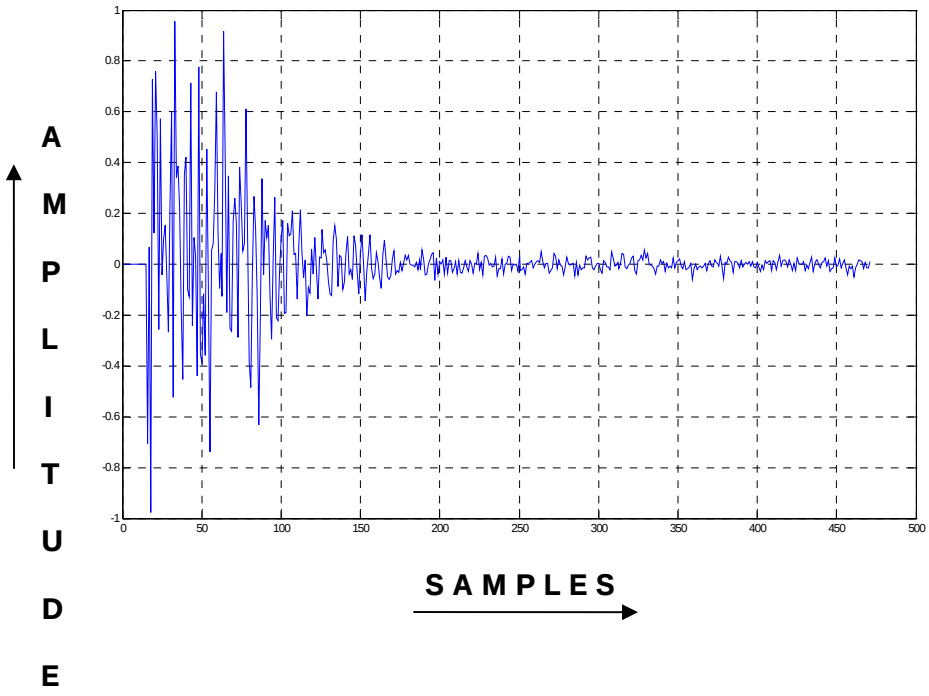


FIGURE 5-14 Convergence Plot of LMS algorithm

The characteristic equations for LMS algorithm are summarized in the table 5-1

Initial Condition:	$0 < \eta < 2$ $w(0) = 0$
For each instant of time, n = 1, 2, ... , compute	
Filter output:	$y(n) = w^H(n)x(n)$
Estimation Error:	$e(n) = d(n)\ \ y(n)$
Tap-Weight Adaptation:	$w(n+1) = w(n) + \eta x(n)e^*(n)$

TABLE 5-1 Characteristic equations for LMS algorithm

5.3.1.2.3.3 – FUNCTIONAL DETAILS

Data is received by the receiver in the form of frames, preceding the synchronization preamble. A part of Synchronization preamble is known to the receiver and is used to update the tap weights of equalizer. Once the tap weights are adjusted they remain same for rest of the received data.

There are total of 39 symbols carrying this information, divide them in 3 groups with 13 symbols each and perform modulo 8 addition with 13 symbol barker code sequence 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 0, 0. In this way values of D0, D1 and D2 are recovered and table 5-1 can be used to determine interleaver length and data rate using these values. And this information is provided to all upcoming blocks, which require this.

5.3.2 - DETECTION

Demodulated and equalized signal points are only an estimation of what signal is most likely to be, the actual signal constellation points are to be detected from the estimated signal to state for sure what the signal process of detection is performed on the signal.

Our detection technique is based on the Maximum Likelihood (ML) detection. The strategy of minimum error criterion is employed where optimum detection threshold regions have been defined for minimizing the probability of making an incorrect decision.

Detection process consist of two steps

STEP 1: Received signal waveform is transformed into a point in the decision space which is referred to as pre-detection point and is variable is formed at the output of demodulator.

Step 2: A symbol decision is made on the basis of comparing the pre-detection point to a threshold.

5.3.2.1 – DETECTION OF SYMBOLS AND MSE

We evaluate the distance between the received signal point and predefined constellation points; the point which is at a minimum distance from received signal point is selected to be the point which was transmitted.

For example if symbol "0" was transmitted that is complex number 1+0j and we received a noisy signal 0.98+0.1j, when compared to 8PSK constellation points it will have minimum distance with symbol zero and thus the received constellation point will be sliced to 1+0j.

5.3.2.2 – DECISION RULE FOR DETECTOR

The MSE described above is often stated in terms of decision rule, where mean of amplitude levels of two consecutive symbols is taken to be defining the threshold boundary between them. For

example if a1 is the amplitude level of symbol 0 and a2 is the amplitude level of symbol 2 then the decision boundary between them will be defined as (a1+a2)/2.

Below are shown 8PSK decision regions. Similar logic and theory is applied to QAM constellations as well.

Following are the shown 8PSK decision regions. Similar logic and theory is applied to QAM constellations as well.

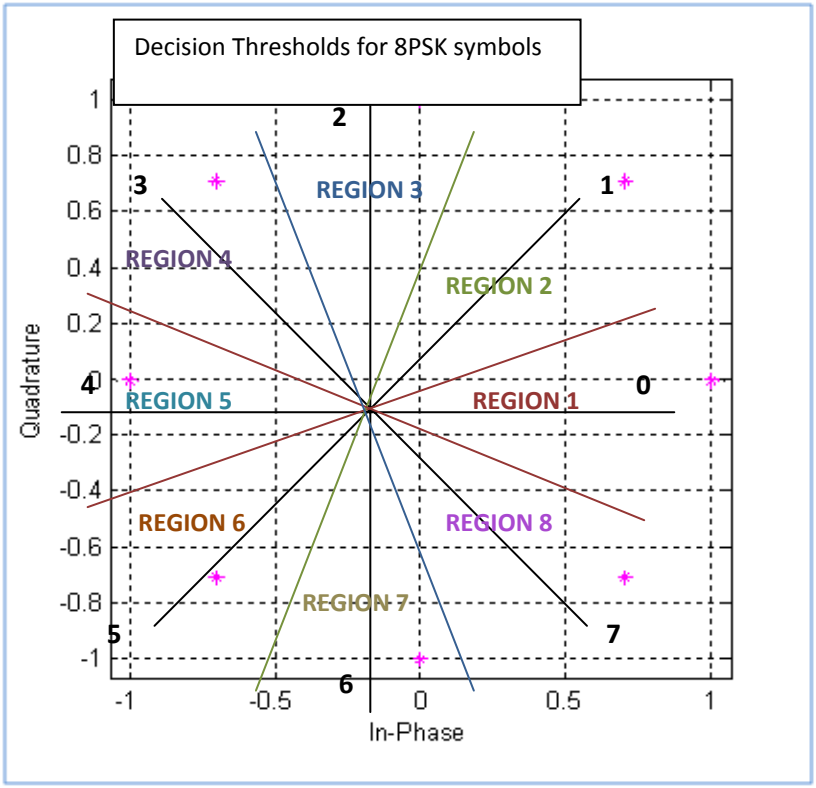


FIGURE 5-15 Decision Threshold for 8PSK Symbols

REGIONS	DECISION
REGION 1	Symbol 0
REGION 2	Symbol 1
REGION 3	Symbol 2
REGION 4	Symbol 3
REGION 5	Symbol 4
REGION 6	Symbol 5
REGION 7	Symbol 6

REGION 8	Symbol 7

TABLE 5-2 Symbols allotted to Regions

5.3.2.3 – SLICER RESULTS

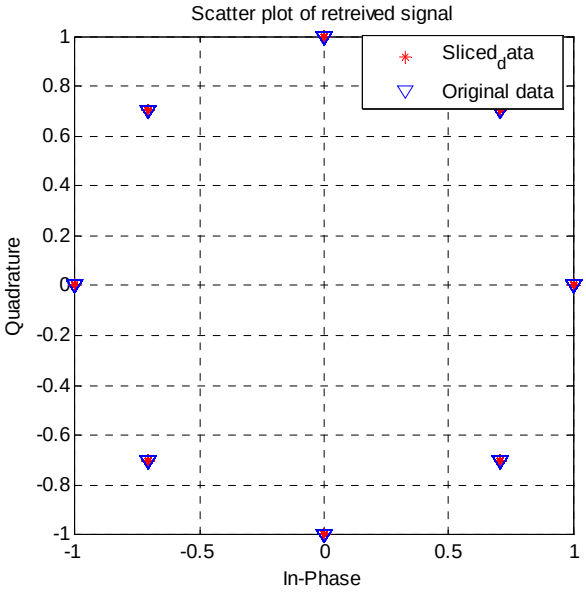


FIGURE 5-16 Slicer 8PSK:

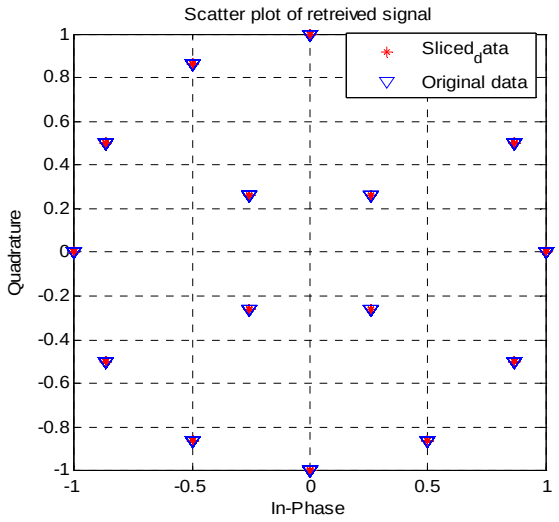


FIGURE 5-17 Slicer 16QAM

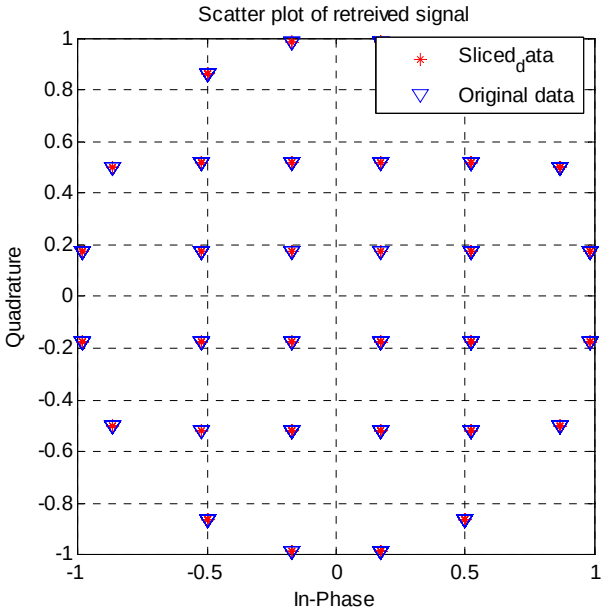


FIGURE 5-18 Slicer 32QAM

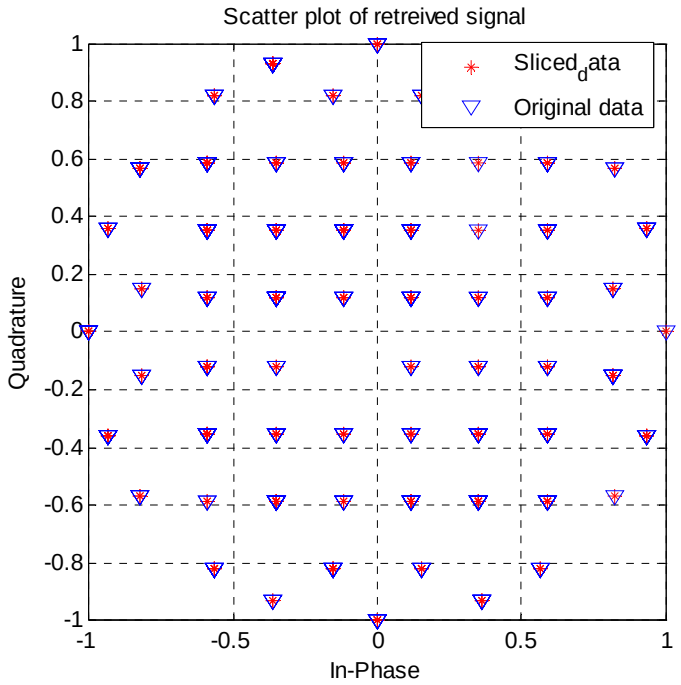


FIGURE 5-19 Slicer 64QAM

5.4 - DESCRAMBLER

Descrambling is the reverse of scrambling. It is implemented by complementing the process done at scrambling end. Scrambling sequence is generated using the same shift register as described in scrambling and whose diagram for 8PSK is given here for reference:

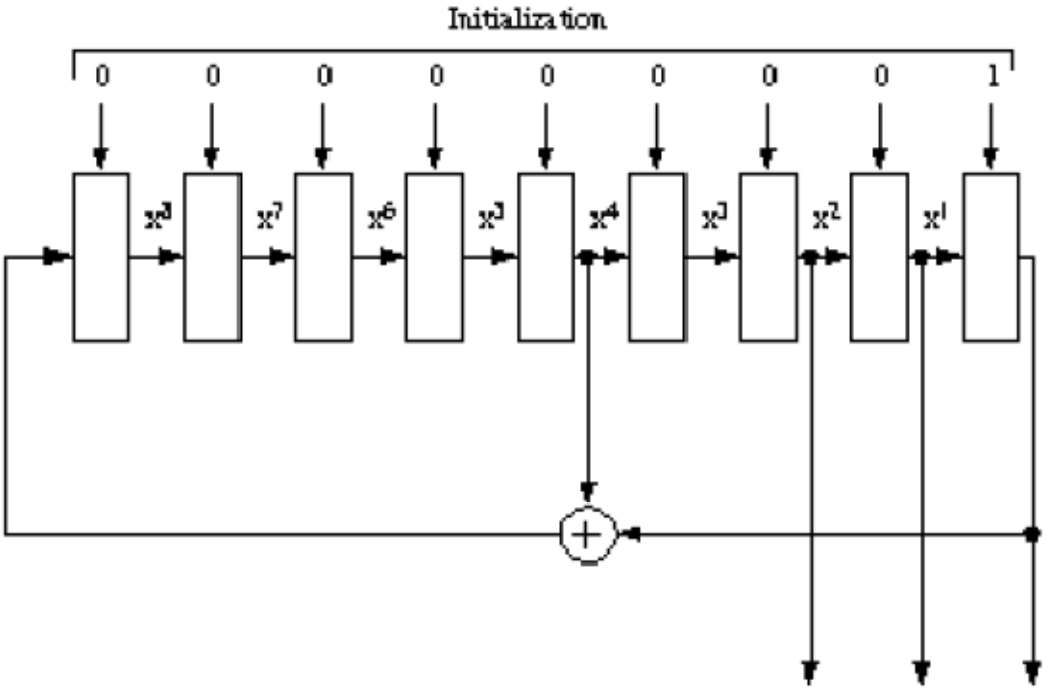


FIGURE 5-20 DESCRAMBLING SEQUENCE

For 8PSK symbols (3200 bps and 4800 bps), the descrambling is carried out by taking the subtraction of numerical value of the binary triplet consisting of the last (rightmost) three bits in the shift register, and the symbol value as received by the receiver . For example, if the last three bits in the scrambling sequence shift register were 010 which has a numerical value equal 2, and the symbol number that is received is 4, symbol would be descrambled will be 2 since: (4-2) Modulo 8 =2 .

For 16QAM symbols, descrambling is done by XORing the 4-bit number consisting of the last (rightmost) four bits in the shift register with the symbol number. For example, if the last 4 bits in the scrambling sequence shift register were 0101 and the 16QAM symbol number received was 6 (i.e. 0110), symbol 3 (0011) would be the descrambled value.

For 32QAM symbols; descrambling is carried out by XORing the 5-bit number formed by the last (rightmost) five bits in the shift register with the symbol number received.

For 64QAM symbols; descrambling is carried out by XORing the 6-bit number formed by the last (rightmost) six bits in the shift register with the symbol number received.

After each data symbol is descrambled, the generator is iterated (shifted) the required number of times to produce all new bits (bit sequence) for use in descrambling the next symbol, as was done in scrambling thus making sure that scrambling and descrambling is carried out with the same bit sequence. The numbers of iterations for different data rates/modulation types are as follows:

- **3 iterations for 8PSK**
- **4 iterations for 16QAM**
- **5 iterations for 32QAM**
- **6 iterations for 64QAM.**

This data rate information is extracted from initial synchronization preamble after *equalization* and given as input to scrambler, to carry out required processing.

5.5 – MODIFIED GRAY ENCODER

After converting data from symbols to bits, and data rate is 3200 bps or 4800 bps it is first passed to Modified Gray encoder and then deinterleaved. If data rate is higher this module is by passed. Table 4-6 and Table4-7 are used to carry out the function.

5.6 - DEINTERLEAVER

The Modified Gray encoded data is deinterleaved. The received data is loaded in another array, using the table 4-4 and table4-5 and following formula

n = (location in interleaved array * Interleaver Increment Value) Modulo (Interleaver Size in Bits)

where n is the actual location of bit before interleavig.

5.7 – SYMBOLS TO BIT CONVERSION

The received symbols, after descrambling are passed to Look up tables where symbols are converted to bits. Depending upon the data rate this process is carried out.

5.7.1 – DATARATE – 3200 bps

If Data Rate is 3200 bps then table 4-9 i-e the transcoding table for 3200 bps is used for de-transcoding. And each symbol is translated in to its dibit equivalent.

5.7.2 – DATARATE – 4800 bps

Table 4-10 i-e the transcoding table for 4800 bps is used for de-transcoding. And each symbol is translated in to its tribit equivalent.

5.7.3 – DATARATE 6000 TO 9600 bps

For these higher data rate the received symbol is converted to its binary equivalent, where no of bits per symbol depends upon data rate selected i-e 4 bits (16QAM), 5 bits (32QAM) or 6 bits (64QAM).

5.8 - DEPUNCTURING

Before decoding data is depunctured. For depuncturing zeros are inserted at punctured bit positions, i-e after every three bits zeros are inserted at 4^{th} and 5^{th} bit positions.

5.9 – EOM DETECTOR

Receiver knows the EOM sequence and thus its encoded form as well. So before carrying out the Decoding algorithm EOM sequence is searched. This allows to detach original data from Flush and Filler bits. Because of this Bit Error Rate performance of Decoder increases significantly and processing gain is also reduced.

5.10 – VITERBI DECODING ALGORITHM

The Viterbi decoding algorithm performs maximum likelihood decoding, however it reduces the computational load by taking advantage of special structure of trellis diagram.

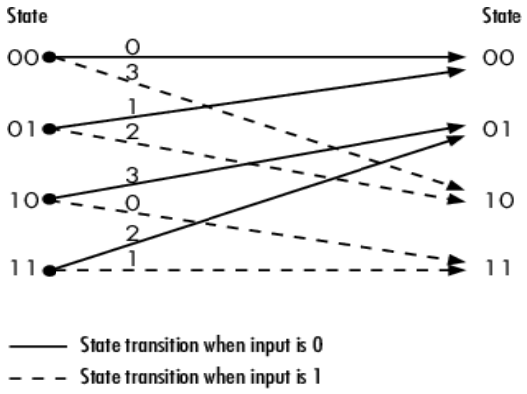


FIGURE 5-21 Trellis diagram for convolutional encoder with constraint length=3 and code rate=1/2

It involves calculating a measure of similarity or distance between the received signal and all the trellis paths entering each state at the same instant of time. The algorithm removes from consideration, all those paths that could not possibly be candidate for the maximum likelihood choice.

Viterbi decoder can receive either soft or hard symbols from the receiver.

5.10.1 – HARD DECODING VS SOFT DECODING

By hard decision we mean a firm irrecoverable decision. In case of hard decision demodulation, data is demodulated in to either 1s or 0s, or quantized in to two levels only. The process described above makes a hard binary decision about each incoming bit and then uses Hamming distances.

By soft decision means the demodulated signal is quantized in to different number of levels. Thus a soft symbol is multileveled, to represent the confidence in the bit being positive or negative. Now the signal value not only indicates whether it's a 0 or 1, but also the probability of their own correctness.

These probability factors can be used by Viterbi algorithm in its computation of the metrics of the sequence using Euclidean distance. As a result the path errors are compared more faithfully, resulting in more informed decisions.

Thus soft decision method achieves better BER performance over hard decision method by about 2 db, and performance improves by using greater number of quantization levels. But by doing this hardware complexity increases exponentially and additional delay in the process of decoding is introduced.

In case of quantization level vs. BER performance, by using quantization, there is a significant processing gain, typically 2db for 3 bit quantized data and about 2.2 db for 4 bit quantized data, over hard decisions.

5.10.2 – A NEW APPROACH

We are often faced with the situations in which hardware complexity and processing gain can not be afforded. And if we have a better channel then why to burden ourselves with increased processing.

But as we can see, through puncturing we lost some data on the transmitter end, and then added 0s to compensate that loss. Thus we ourselves introducing a probable error in the received data, which affects the hard decoder out put, significantly.

Hence we established a better decoder which make use of hard decisions, and thus providing the ease of using less complex hardware and less processing gain, and then assign the probability of errors itself to punctured and non-punctured bit positions, in order to achieve a better performance.

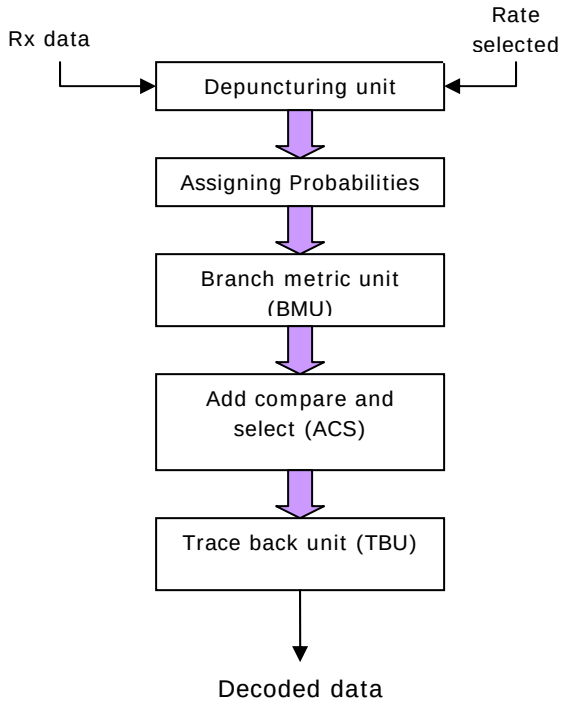


Figure 5-22 Architecture of Viterbi Decoder, following a new approach

5.10.3 – ASSIGNING PROBABILITY OF ERRORS

Since in our new approach we took hard decisions so well defined 0s or 1s are there at unpunctured positions.

Now using the following table the unpunctured bit positions are assigned 0 if they are 0 and 7 if they are 1. i-e we assume them to be hundred percent correct and thus consider them strongest 0s or 1s.

Value	Meaning
0	**strongest 0**
1	**Relatively strong 0**
2	**Relatively weak 0**
3	**weakest 0**
4	**weakest 1**
5	**Relatively weak 1**
6	**Relatively strong 1**
7	**strongest 1**

TABLE 5-3 Interpretations of different values

While for punctured positions we randomly assign 2 or 5 and thus introducing ourselves, a probability of error for punctured bit positions. i-e we define decoder that at this position there might be a 0 or might be a 1.

5.10.4 – BRANCH METRIC UNIT (BMU)

A branch metric unit (BMU) is implemented as a look up table and Euclidean distances are used for comparing the received symbols obtained from probability assigning table. The two symbols of the received code word can be projected on the coordinate system. The Euclidean distance between ideal code words and an arbitrary point in this space (the received code word) gives the branch metrics.

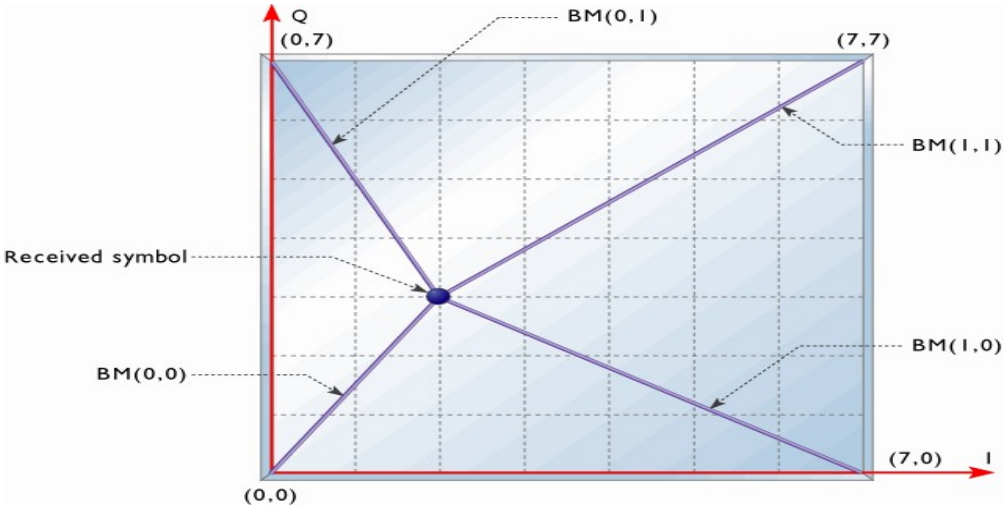


FIGURE 5-23 *Branch matrix generator for a Viterbi decoder.*

5.10.5 – THE ACS UNIT

The ACS unit is the heart of the Viterbi decoder. Each node in the trellis diagram corresponds to an ACS processor in the corresponding Viterbi decoder. The constraint length of the encoder (K) gives the number of ACS processors for a parallel architecture

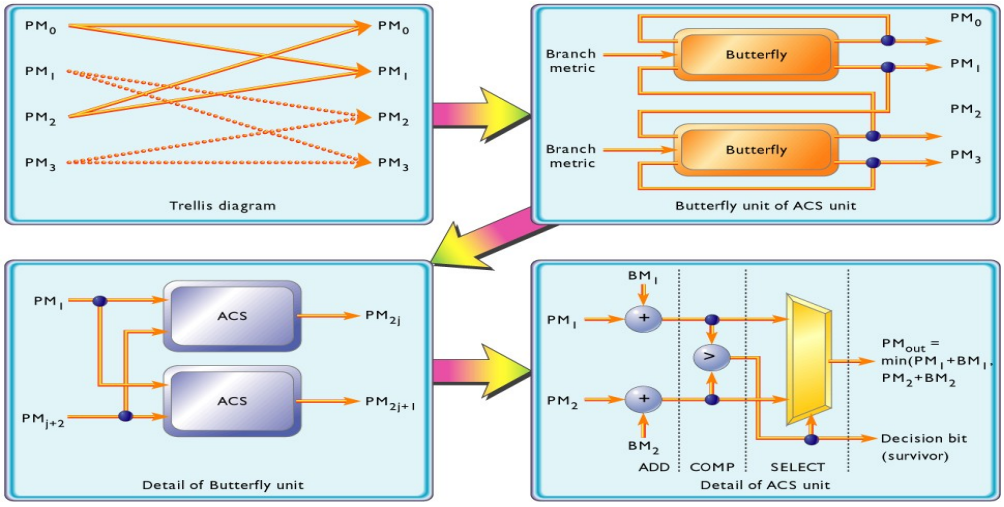


FIGURE 5-24 Diagram of a typical ACS unit.

5.10.6 – TRACE BACK UNIT

In this unit out put of the decoder is formed by tracing the selected paths, stored in Path Metric, to the oldest bit on the path.

This tracing back is carried out after certain time instants, and number of these instants is called trace back depth, and larger this value better will be the performance of decoder but this increases the hardware complexity as because of increase in memory requirement.

In tracing back, we start moving back along the selected paths stored in Path Metric, and it is seen that this path results with a 0 or 1 input to the encoder, when that is determined, we select that 0 or 1 as decoded bit.

CHAPTER 6

INTRODUCTION TO TMS320C6713 DSP PROCESSOR

6.1 – INTRODUCTION

The C6713™ DSK builds on TI's industry-leading line of low cost, easy-to-use DSP Starter Kit (DSK) development boards. The high-performance board features the TMS320C6713 floating-point DSP. Capable of performing 900 million floating-point operations per second (MFLOPS), the C6713 DSP makes the C6713 DSK the most powerful DSK development board on the market.

The DSK is a parallel port interfaced platform that allows TI, its customers, and third parties, to efficiently develop and test applications for the C6713. The DSK consists of a C6713-based printed circuit board that will serve as a hardware reference design for TI's customers' products. With extensive host PC and target DSP software support, including bundled TI tools, the DSK provides ease-of-use and capabilities that are attractive to DSP engineers.

6.2 – DESCRIPTION

The TMS320C62xE DSPs (including the TMS320C6211 device) are the fixed-point DSP family in the TMS320C6000E DSP platform. The TMS320C67xE DSPs (including the TMS320C6713 device) are the floating-point DSP family in the TMS320C6000E DSP platform. The TMS320C6211 (C6211) and TMS320C6713 (C6711) devices are based on the high-performance, advanced Velocity very-long-instruction-word (VLIW) architecture developed by Texas Instruments (TI), making these DSPs an excellent choice for multi channel and multifunction applications.

With performance of up to 1333 million instructions per second (MIPS) at a clock rate of 167 MHz, the C6211 device offers cost-effective solutions to high-performance DSP programming challenges. The C6211 DSP possesses the operational flexibility of high-speed controllers and the numerical capability of array processors.

This processor has 32 general-purpose registers of 32-bit word length and eight highly independent functional units. The eight functional units provide six arithmetic logic units (ALUs) for a high degree of parallelism and two 16-bit multipliers for a 32-bit result. The C6211 can produce two multiply-accumulates (MACs) per cycle for 333 million MACs per second (MMACS).

With performance of up to 900 million floating-point operations per second (MFLOPS) at a clock rate of 150 MHz, the C6711 device also offers cost-effective solutions to high-performance DSP programming challenges. The C6713 DSP possesses the operational flexibility of high-speed controllers and the numerical capability of array processors. This processor has 32 general-purpose registers of 32-bit word length and eight highly independent functional units. The eight functional units provide four floating-/fixed-point ALUs, two fixed-point ALUs, and two floating-/fixed-point multipliers. The C6711 can produce two MACs per cycle for 300 MMACS.

- **The C6713 DSP has application-specific hardware logic, on-chip memory, and additional on-chip peripherals.**
- **The C6713 uses a two-level cache-based architecture and has a powerful and diverse set of peripherals.**

The Level 1 program cache (L1P) is a 32-Kbit direct mapped cache and the Level 1 data cache (L1D) is a 32-Kbit 2-way set-associative cache. The Level 2 memory/cache (L2) consists of a 512-Kbit memory space that is shared between program and data space. L2 memory can be configured as mapped memory, cache, or combinations of the two. The peripheral set includes two multichannel-buffered serial ports (McBSPs), two general-purpose timers, a host-port interface (HPI), and a glue less external memory interface (EMIF) capable of interfacing to SDRAM, SBSRAM and asynchronous peripherals.

The C6211/C6713 has a complete set of development tools, which includes a new C compiler, an assembly optimizer to simplify programming and scheduling, and a Windows E debugger interface for visibility into source code execution.

6.3 – C6713 DSK Board description

The C6713 DSK board is displayed in figure 6-1. Its various components are as follows:

- **A parallel peripheral interface**
- **SDRAM and ROM**

- **A 16-bit analog interface circuit (AIC)**
- **An I/O port**
- **Embedded JTAG emulation support**

Connectors on the C6713 DSK provide DSP external memory interface (EMIF) and peripheral signals that enable its functionality to be expanded with custom or third party daughter boards. In addition to above some features of DSK board are as follows:

- **150-MHz C6713DSP capable of executing 900 million floating-point operations per second (MFLOPS)**
- **Dual clock support; CPU at 150MHz and external memory interface (EMIF) at 100MHz**
- **Parallel port controller (PPC) interface to standard parallel port on a host PC (EEP or bi-directional SPP support)**
- **16M Bytes of 100 MHz synchronous dynamic random access memory (SDRAM)**
- **128 K Bytes of flash programmable and erasable read only memory (ROM).**
- **8-bit memory-mapped I/O port**
- **Embedded JTAG emulation via the parallel port and external XDS510 support**
- **Host port interface (HPI) access to all DSP memory via the parallel port**
- **16-bit audio codec**
- **Onboard switching voltage regulators for 1.8 volts direct current (VDC) and 3.3 VDC**
- **Six light emitting diode (LED) indicators (one power-on indicator, one TBC-in-use indicator, one reset-active indicator, and three user-defined indicators)**
- **External desktop operation utilizing an external power supply and IEEE1284 parallel cable or an XDS510 emulator**
- **Power supply and IEEE1284 parallel cable are provided with the DSK..**
- **Expansion memory and peripheral connectors for daughterboard support.**
- **The DSK provides a C6713 hardware reference design that can assist you in the development of your own C6713-based products. In addition to providing a reference for interfacing the DSP to various types of memories and peripherals, the design also addresses power, clock, JTAG, and parallel peripheral interfaces.**

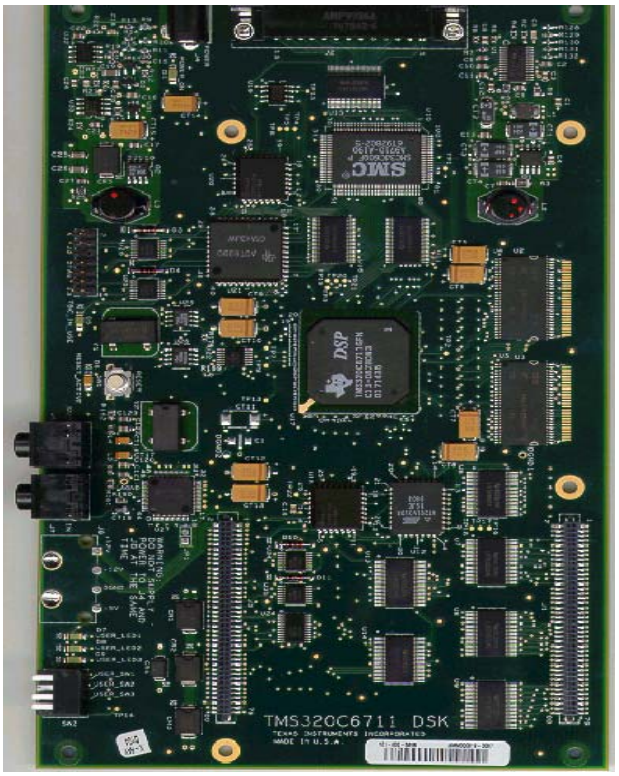


FIGURE 6-1 The 6713 DSK board

6.3.1 – SOME KEY FEATURES OF THE CPU

- **Eight highly independent functional units (including six ALUs and two multipliers)**
- **32-bit general-purpose registers**
- **900 million floating-point operations per second (MIPS)**
- **6.7-ns cycle time.**
- **Up to eight 32-bit instructions per cycle**
- **Byte-addressable (8-, 16-, 32-bit data**
- **32-bit address range**
- **8-bit overflow protection**
- **Little- and big-endian support**
- **Saturation**
- **Normalization**

- **Bit-field instructions (extract, set, clear)**
- **Bit-counting**

6.3.2 – CPU (DSP CORE) DESCRIPTION

The CPU fetches Velocity advanced very-long instruction words (VLIW) (256 bits wide) to supply up to eight 32-bit instructions to the eight functional units during every clock cycle. The Velocity VLIW architecture features controls by which all eight units do not have to be supplied with instructions if they are not ready to execute. The first bit of every 32-bit instruction determines if the next instruction belongs to the same execute packet as the previous instruction, or whether it should be executed in the following clock as a part of the next execute packet. Fetch packets are always 256 bits wide; however, the execute packets can vary in size. The variable-length execute packets are a key memory-saving feature, distinguishing the C62x and C67x CPUs from other VLIW architectures.

The CPU features two sets of functional units. Each set contains four units and a register file. One set contains functional units .L1, .S1, .M1, and .D1; the other set contains units .D2, .M2, .S2, and .L2. The two register files each contain 16 32-bit registers for a total of 32 general-purpose registers. The two sets of functional units, along with two register files, compose sides A and B of the CPU. The four functional units on each side of the CPU can freely share the 16 registers belonging to that side. Additionally, each side features a single data bus connected to all the registers on the other side, by which the two sets of functional units can access data from the register files on the opposite side. While register access by functional units on the same side of the CPU as the register file can service all the units in a single clock cycle, register access using the register file across the CPU supports one read and one write per cycle.

The C67x CPU executes all C62x instructions. In addition to C62x fixed-point instructions, the six out of eight functional units (.L1, .M1, .D1, .D2, .M2, and .L2) also execute floating-point instructions. The remaining two functional units (.S1 and .S2) also execute the new LDDW instruction, which loads 64 bits per CPU side for a total of 128 bits per cycle.

Another key feature of the C67x CPU is the load/store architecture, where all instructions operate on registers (as opposed to data in memory). Two sets of data-addressing units (.D1 and .D2) are responsible for all data transfers between the register files and the memory. The data address driven by the .D units allows

data addresses generated from one register file to be used to load or store data to or from the other register file.

The C62x/C67x CPU supports a variety of indirect addressing modes using either linear- or circular-addressing modes with 5- or 15-bit offsets. All instructions are conditional, and most can access any one of the 32 registers. Some registers, however, are singled out to support specific addressing or to hold the condition for conditional instructions (if the condition is not automatically "true"). The two .M functional units are dedicated for multiplies. The two-.S and .L functional units perform a general set of arithmetic, logical, and branch functions with results available every clock cycle.

The processing flow begins when a 256-bit-wide instruction fetch packet is fetched from a program memory. The 32-bit instructions destined for the individual functional units are "linked" together by "1" bits in the least significant bit (LSB) position of the instructions. The instructions that are "chained" together for simultaneous execution (up to eight in total) compose an execute packet. A "0" in the LSB of an instruction breaks the chain, effectively placing the instructions that follow it in the next execute packet. If an execute packet crosses the fetch-packet boundary (256 bits wide), the assembler places it in the next fetch packet, while the remainder of the current fetch packet is padded with NOP instructions.

The number of execute packets within a fetch packet can vary from one to eight. Execute packets are dispatched to their respective functional units at the rate of one per clock cycle and the next 256-bit fetch packet is not fetched until all the execute packets from the current fetch packet have been dispatched. After decoding, the instructions simultaneously drive all active functional units for a maximum execution rate of eight instructions every clock cycle. While most results are stored in 32-bit registers, they can be subsequently moved to memory as bytes or half-words as well. All load and store instructions are byte-, half-word, or word-addressable.

6.4 – TMS320C6713 CORE, PERIPHERALS, AND EXTERNAL INTERFACES

Because the C6713 DSK is hardware, references design, many of the 'C6713's peripherals have been utilized to demonstrate one, if not more, applications for each peripheral. The JTAG test/emulation interface is used to support both embedded and external emulation for source code debugging. The control interface is used to reset the DSP, provide external interrupts, and choose data endian mode.

The external memory interface (EMIF) supports synchronous SDRAM memories and asynchronous accesses to Flash ROM, the I/O port, and expansion memory. The EMIF is also brought out to an expansion memory interface connector for daughterboard use. The host port interface (HPI) is used for bi-directional data transfers between the PC and the DSP. The timer interfaces are provided on the expansion peripheral interface connector for daughterboard use. The multi-channel buffered serial ports provide interfaces to an onboard analog interface circuit and a daughterboard connected to the expansion peripheral interface. The C6713 DSK external interface diagram is shown in figure 6-2.

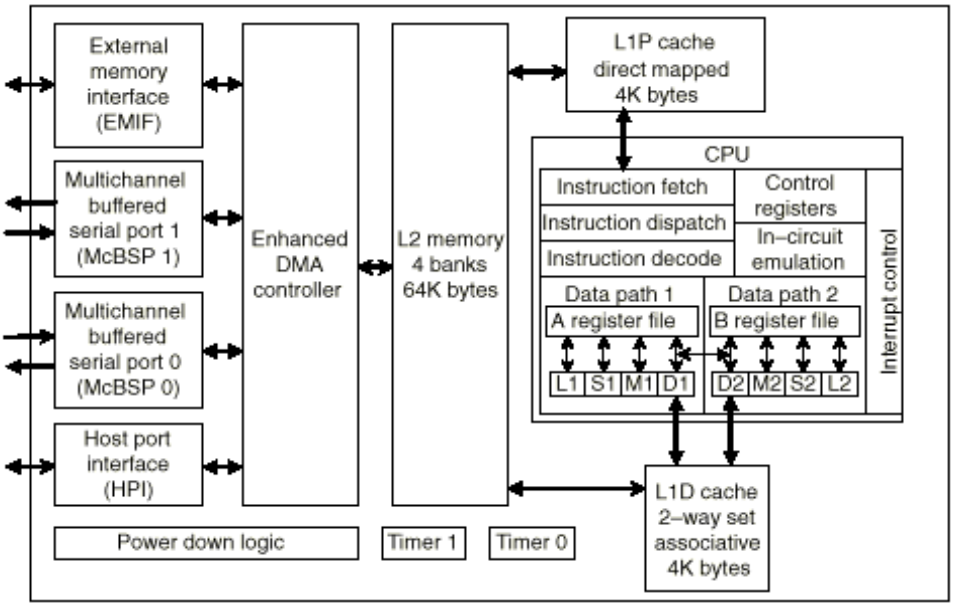


FIGURE 6-2 TMS320C6713 Core, Peripherals, and External Interfaces

6.5 – SUMMARY

The C6713™ DSK builds on TI's industry-leading line of low cost, easy-to-use DSP Starter Kit (DSK) development boards. The C6713 DSP makes the C6713 DSK the most powerful DSK development board on the market.

The various components of the board are as follows: A parallel peripheral interface, SDRAM and ROM, A 16-bit analog interface circuit (AIC), An I/O port, embedded JTAG emulation support.

Some key features of the CPU: Eight highly independent functional units (including six ALUs and two multipliers), 32-bit general-purpose registers. 900 million floating-point operations per second (MIPS), 6.7-ns cycle time, up to eight 32-bit instructions per cycle, byte-addressable (8-, 16-, 32-bit data, 32-bit address

range, 8-bit overflow protection, little-and big-endian support, saturation, normalization, bit-field instructions (extract, set, clear), bit counting.

Another key feature of the C67x CPU is the load/store architecture, where all instructions operate on registers (as opposed to data in memory).

CHAPTER 7

MODELING IN SIMULINK

1 – ABOUT SIMULINK

Simulink® is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates.

Simulink encourages you to try things out. You can easily build models from scratch, or take an existing model and add to it. Simulations are interactive, so you can change parameters on the fly and immediately see what happens. You have instant access to all the analysis tools in MATLAB®, so you can take the results and analyze and visualize them. A goal of Simulink is to give you a sense of the fun of modeling and simulation, through an environment that encourages you to pose a question, model it, and see what happens.

With Simulink, you can move beyond idealized linear models to explore more realistic nonlinear models, factoring in friction, air resistance, gear slippage, hard stops, and the other things that describe real-world phenomena. Simulink turns your computer into a lab for modeling and analyzing systems that simply wouldn't be possible or practical otherwise, whether the behavior of an automotive clutch system, the flutter of an airplane wing, the dynamics of a predator-prey model, or the effect of the monetary supply on the economy.

Simulink is also practical. With thousands of engineers around the world using it to model and solve real problems, knowledge of this tool will serve you well throughout your professional career.

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this interface, you can draw the models just as you would with pencil and paper (or as most textbooks depict them). This is a far cry from previous simulation packages that require you to formulate differential equations and difference equations in a language or program. Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors. You can also customize and create your own blocks.

Models are hierarchical, so you can build models using both top-down and bottom-up approaches. You can view the system at a high level, then double-click blocks to go down through the levels to see increasing levels of model detail. This approach provides insight into how a model is organized and how its parts interact.

After you define a model, you can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in the MATLAB Command Window. The menus are particularly convenient for interactive work, while the command-line approach is very useful for running a batch of simulations (for example, if you are doing Monte Carlo simulations or want to sweep a parameter across a range of values). Using scopes and other display blocks, you can see the simulation results while the simulation is running. In addition, you can change parameters and immediately see what happens, for "what if" exploration. The simulation results can be put in the MATLAB workspace for postprocessing and visualization.

Model analysis tools include linearization and trimming tools, which can be accessed from the MATLAB command line, plus the many tools in MATLAB and its application toolboxes. And because MATLAB and Simulink are integrated, you can simulate, analyze, and revise your models in either environment at any point.

2 – HOW SIMULINK WORKS

Simulink is a software package that enables you to model, simulate, and analyze systems whose outputs change over time. Such systems are often referred to as dynamic systems. Simulink can be used to explore the behavior of a wide range of real-world dynamic systems, including electrical circuits, shock absorbers, braking systems, and many other electrical, mechanical, and thermodynamic systems. This section explains how Simulink works.

Simulating a dynamic system is a two-step process with Simulink. First, a user creates a block diagram, using the Simulink model editor, that graphically depicts time-dependent mathematical relationships among the system's inputs, states, and outputs. The user then

commands Simulink to simulate the system represented by the model from a specified start time to a specified stop time.

3 – SIMULINK BLOCKS

Block are categorized in Simulink in the following ways:

- **Commonly Used**
- **Continuous**
- **Discontinuities**
- **Discrete**
- **Logic and Bit Operations**
- **Lookup Tables**
- **Math Operations**
- **Model Verification**
- **Model-Wide Utilities**
- **Ports & Subsystems**
- **Signal Attributes**
- **Signal Routing**
- **Sinks**
- **Sources**
- **User-Defined Functions**
- **Additional Math & Discrete**

For our purposes, we have used the User-Defined Functions.

4 – USER-DEFINED FUNCTIONS

There are different types of User-Defined Functions in Simulink.

- **Embedded MATLAB Function**
- **Fcn**
- **MATLAB Fcn**
- **M-File S-Function**
- **S-Function**
- **S-Function Builder**

We have made our blocks using the Embedded MATLAB Function.

5 – EMBEDDED MATLAB FUNCTION

An Embedded MATLAB Function block lets you compose a MATLAB function in Simulink. The MATLAB function you create executes for simulation and generates code for a Real-Time Workshop target. You specify input and output data to the Embedded MATLAB Function block in the function header as arguments and return values.

To generate embeddable code, the Embedded MATLAB Function block relies on an analysis that determines the size and class of each variable. This analysis imposes the following additional restrictions on the way in which the above features may be used. The first definition of a variable must define both its class and size. The class and size of a variable cannot be changed once it has been set. Whether data is complex or real is determined by the first definition. Subsequent definitions may assign real numbers into complex storage but may not assign complex numbers into real storage.

6 – MODEL OF MIL-STD-188-110B

The following figure shows the model we have designed,

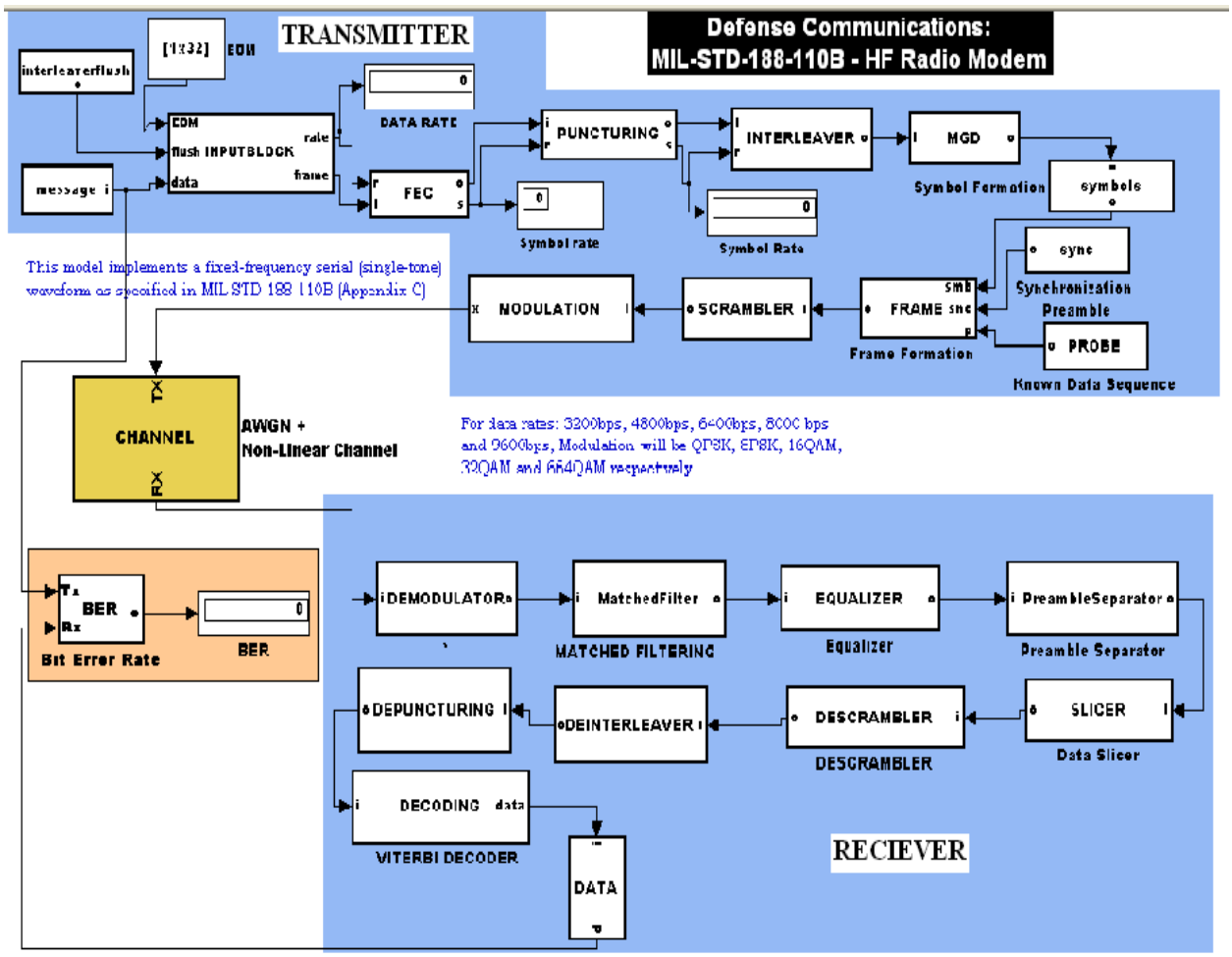


FIGURE 7-1 Simulink Model of MIL-STD-188-110B

In this model, one can clearly make out all the processes going on in the transmitter and the reciever. The inputblock is formed by combining the message, EOM and the interleaver flush. Datarate is calculated and the frame is sent to the FEC for encoding. The data is then punctured, interleaved and passed through the MGD after which the symbols are formed. The final frame is than scrambled and modulated to be sent over the channel.

At the reciever end, the signal is demodulated, equalized and then sliced to retrieve the symbols. Data is then descrambled, deinterleaved, depunctured and finally sent to the viterbi decoder from where the final message is retrieved. Also the bit error rate is calculated at the end.