ADDIS ABABA UNIVERSITY
Faculty of Technology

Electrical Engineering Department
ኤሌክቲል ምህንድሲ ት/ክፍል

P.O. Box 385
Tel. 12 25 30

Date: 11 January 1998
Ref. No EE/ /91

To: Graduate School,
Addis Ababa University

From: Electrical Engineering Department
Faculty of Technology

Subject: **MSc Thesis Defense Results**

The following students, in the Electrical Engineering Department's graduate program, have successfully defended their MSc Thesis in December 1998, as certified by their respective Examining Boards. As a result, they have been recommended for graduation by the DGC of the Department and the FGC of the Faculty of Technology.

1. Ato Laine Berhane
2. W/t Seble Mengesha
3. Ato Yohannes Kassahun

Ato Getachew Kebede who defended his thesis last June has also completed the Program by taking the one remaining course this semester. He is, therefore, recommended for graduation.

I have included all the necessary documents of these students for your office's consideration and recommendation. The total number of graduates with the MSc degree from the Department during the first semester of the current academic year is four.

Thank you.

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

Design and Realization of an Audio Frequency Filter Unit for Shortwave Communication System Using TMS320C50 DSP Kit

By

Seble Mengesha

December 1998

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY
ELECTRICAL ENGINEERING DEPARTMENT

Design and Realization of an Audio Frequency Filter Unit for Shortwave Communication System Using TMS320C50 DSP Kit

A Thesis Submitted to School of Graduate Studies of Addis Ababa University in Partial Fulfilment for the Degree of Masters of Science in Electrical Engineering

By

Seble Mengesha

Advisor

Dr.-Ing habil P. Haferkorn

Addis Ababa University

December 1998

Addis Ababa University
School of Graduate Studies
Faculty of Technology
Department of Electrical Engineering

Design and Realization of an Audio Frequency Filter Unit for Shortwave Communication System Using TMS320C50 DSP Kit

By
Seble Mengesha

Approved by Board of Examiners:

Dr. Eneyew Adugna, _____
Chairman, Department Graduate Committee (DGC)

Dr.-Ing habil P. Haferkorn _____
Advisor

Dr.-Ing Mohammed Abdo _____
Examiner

Dr. Rakesh Ranjan _____
External Examiner

To My Family . . .

Table of Content

List of Figures and Tables

List of Abbreviations

A/D	Analog to Digital
AFSK	Audio Frequency shift Keying
AIC	Analog Interface Circuit
AM	Amplitude Modulation
AMTOR	Amateur Teleprinting over Radio
ARQ	Automatic Repeat Request
ASCII	American Standard Code for Information Exchange
ASK	amplitude Shift Keying
ATV	Amateur Televison
COM port	Communication port
CW	Continuous Wave
D/A	Digital to Analog
dB	Decibel
DSP	Digital Signal Processing
FEC	Forward Error Correction
FIR	Finite Impulse Response
FM	Frequency Modulation
FSK	Frequency Shift Keying
FSTV	Fast Scan Television
G-TOR	Goley Teleprinting over Radio
HF	High Frequency

IRS	Information Receiving Stations
ISS	Information Sending Station
kHz	Kilo Hertz
MHz	Mega Hertz
MIPS	Million Instructions per Second
MS	Master Station
NAKs	Negative Acknowledge Signals
OOK	On-Off Keying
PACTOR	Packet Teleprinting over Radio
PC	Personal Computer
PTT	Push to Talk
RCVR	Receiver
RF	Radio Frequency
RTTY	Radioteletype
S/N	Signal to Noise Ratio
SITOR	Simplex Telex over Radio
SS	Slave Station
SSB	Single Side Band
SSTV	Slow Scan Television
TTYs	Teletype Signals
TU	Terminal Unit
TV	Television
VHF	Very High Frequency
WPM	Word per Minute
XMTR	Transmitter

Preface

Despite the large and varied analog and digital type of equipment in the Electrical Engineering Laboratory of Addis Ababa University, digital signal processor (DSP) kit remains one of the few inadequately available experimental material. Although the theory of digital filter design on DSP kit have been made by many researchers, as we all learn too, our knowledge of the implementation of DSP kit in the laboratory is still far from adequate. This is because the material were not available in our laboratory to prove/disprove the advantage of DSP kit over their analog counterpart upon the development of filter designs.

The TMS320C50 DSP is used to implement the audio-filter unit designed and is tested on the communication modes exist in Amateur Radio Data Communication System. The system gives a radio communication service for the purpose of self training, intercommunication and technical investigation carried by amateur, that is duly authorized person interested in radio technique solely with a personal aim without pecuniary interest.

Part of the objective of this thesis work, therefore, is to fill the gap between the theoretical and practical aspect of a system design, thereby contributing to the improvement of shortwave radio data communication system performance.

The thesis work is divided into six major chapters other than the implementation of the thesis work. The introductory part (chapter one) takes heed of the conventional kind of beginning for a thesis submitted to the School of Graduate Studies paper in Addis Ababa University. All technical questions related to the need for a filter unit implemented on a digital signal processor are discussed in this part.

The analog and digital modes of radio data communications are presented in chapter two. Besides describing the different modes of data communication, the CW (continuous wave), RTTY (radioteletype) and AMTOR (amateur teleprinting over radio) type modes are highlighted here. Although it may sound less relevant to the title of the thesis work, this chapter serves two important purposes. First, it gives (to those who know little or nothing about modes of data communication) a general view of what really meant by modes of radio data communication. Secondly, it serves as a basis for a clear understanding to answer for questions related to CW, RTTY and AMTOR type modes since they are used as a testing signals for this thesis work.

The third chapter describes the communication interface of the amateur radio data communication system. It also discusses the decoding schemes used for the incoming signals. The fourth chapter end by giving a general description and architecture of the TMS320C5X DSP kit.

Attempts are made to digital filters, in chapter five, on their advantages and design specifications. The FIR digital filter design techniques are also discussed in the sixth chapter, meticulously on Kaiser window.

And the last section (which is the implementation part) - chapter seven, comprises the design and realization procedures followed in the implementation of the filter unit. It also compares the actual result with that of the ideal and gives the relevant explanations.

Appendix has three parts. The first part gives the internal hardware and instruction set summary for the TMS320C5X DSP kit. The second part explains about fixed-point representation of binary numbers. The last part is the listing of the assembly source code.

Even if there is no only one scientific presentation of selected bibliography in any research work, numbering will be assumed more convenient to note for the rest of this paper.

Addis Ababa University

Seble Mengesha

17th December 1998

Acknowledgment

I am grateful to my advisor, Dr.-Ing habil P. Haferkorn, for his unreserved cooperation from moral and material support up to his constructive comments that greatly enriches this thesis.

I wish to thank Technology Faculty of Addis Ababa University in facilitating me the Post Graduate Program which is sponsored by DAAD, German Academic Exchange Service. I am very much indebted to DAAD, which covered the cost of this study and other expenses.

My special thanks and appreciation is due to Dr. Eneyew Adugna who shared his materials without reservation. His critical question and constructive suggestions in the time of experimental work were also of great help.

Thanks also to all my teachers, colleagues and my students for their friendly encouragement. Without their help and reassurance this thesis could not have taken shape.

Finally, my heartfelt appreciation and indebtedness goes to my family for their love, support, patience and encouragement throughout my study.

Abstract

Causes that may affect the quality of shortwave data communication over radio are presented. Designing an audio-frequency filter unit, other than the filters exist in the transceiver, is considered as one solution to improve the quality of this shortwave data communication. Among the different data communication modes exist in the communication system, the CW (continuous wave), RTTY (Radio teletype) and AMTOR (amateur teleprinting over radio) types are selected to test the performance of the designed filter unit. The filter unit is of FIR digital filter type which the impulse response coefficients are obtained using the Kaiser Window design technique. The DSK (digital signal processor starter kit) assembler and debugger are used to develop the assembly code so as to realize and implement the filter on TMS320C50 digital signal processor. The magnitude response of the resulting filter unit is estimated by a spectrum analyzer. Finally, the designed filter unit is tested in an Amateur Radio Communication System.

Abstract

Causes that may affect the quality of shortwave data communication over radio are presented. Designing an audio-frequency filter unit, other than the filters exist in the transceiver, is considered as one solution to improve the quality of this shortwave data communication. Among the different data communication modes exist in the communication system, the CW (continuous wave), RTTY (Radio teletype) and AMTOR (amateur teleprinting over radio) types are selected to test the performance of the designed filter unit. The filter unit is of FIR digital filter type which the impulse response coefficients are obtained using the Kaiser Window design technique. The DSK (digital signal processor starter kit) assembler and debugger are used to develop the assembly code so as to realize and implement the filter on TMS320C50 digital signal processor. The magnitude response of the resulting filter unit is estimated by a spectrum analyzer. Finally, the designed filter unit is tested in an Amateur Radio Communication System.

Chapter 1

Introduction

Background of the Problem

Messages transmission by means of radio waves depends on the physical phenomenon that electrical oscillations can be radiated into space and transported over great distances in the form of electromagnetic waves. The massages may be radiated within a band ranging from about 10 kHz to frequencies far beyond 3000 MHz. Within this overall range each subrange is characterized by its peculiar propagation conditions which must be taken into account in designing of communication systems.

The wave band which is of interest in this thesis work is the *high frequency band*, also called shortwave range, covering from 1.5 to 30 MHz of the frequency spectrum. This range is attractive for message transmission due to the fact that great distances can be spanned at moderate equipment cost and with comparatively low transmitting power.

The electromagnetic waves radiated by a radio transmitter and propagated close to the surface will be attenuated under the influence of the earth's surface. In shortwave range, these *ground-waves* are subjects to a comparatively heavy attenuation so that the transmission range is very small. Thus, for spanning great distances in continental and intercontinental radio service only *sky-waves* can be considered as useful means of propagation. If radiated under a favorable angle, these waves are reflected in the ionosphere back to the surface of the earth.

But the field strength produced at the receiving station by the sky-wave is subject to perturbations during the propagation of electromagnetic waves. The continual fluctuation of the ionosphere as regards structure, altitude and ionic density, lead to the corresponding change in reflection conditions which in turn are likely to affect shortwave radio communication. This radio transmission/reception may also be impaired by *fading*, *atmospherics*, *interference caused by electrical apparatus, interference from radio transmitters,* and *white noise*.

If these interferences come within the frequency range of the useful signal, an appreciable reduction in the quality of transmission will be resulted. The influence of interference increases as the difference in the amplitude between the noise signal and the useful signal decreases. The measure of this is the *signal-to-noise ratio* which is expressed is dB. The interference may be classified as [3]

- Noises energy, which is present all the time and is almost uniformly, distributed over a wide frequency band and over the time (white noise).
- Crackling, in most cases only of short duration, but affecting broad frequency band and
- Energy from other radio transmitters and discrete interference which hits a frequency band selectively (intentionally or unintentionally) and which may present continuously.

A knowledge of the disturbance affecting communication over a short wave link is important for the choice of suitable countermeasures intended to permit message transmission with a minimum trouble.

One of the countermeasures is a careful design of the radio equipment giving due consideration to the gain of the transmitter and receiver antenna (directivity), sensitivity of the receiver and the transmission conditions and the noise level at the receiving station. Apart from design of the radio equipment and the associated antennae, the type of modulation

technique employed must also be noted for digital message transmission. The following block diagram shows a general radio data communication equipment. [2]

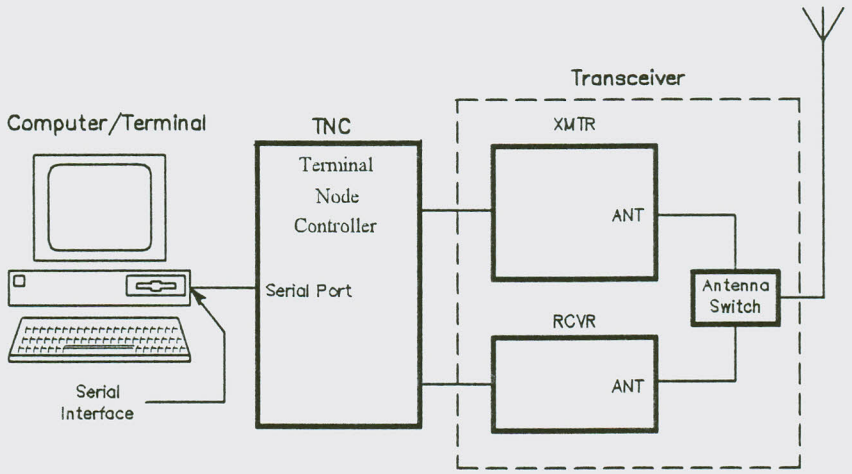


Fig. 1-1 Radio Data Communication Equipment

Despite all these design consideration, the received noise level may sometimes prevail the useful signal and significantly affects the reliability with which the message is correctly received. Thus, in addition to the already existing bandpass and lowpass filters in the radio transmitter and receiver, which are, of course, matched to the transmission speed and having minimum possible attenuation and delay distortion, another filtering unit is a must for successful end-to-end data communication. This additional filtering unit will modify or reshape the audio frequency spectrum of the signal according to a desired specification. The block diagram in Fig. 1-2 depicts the radio communication system with the filter unit included.

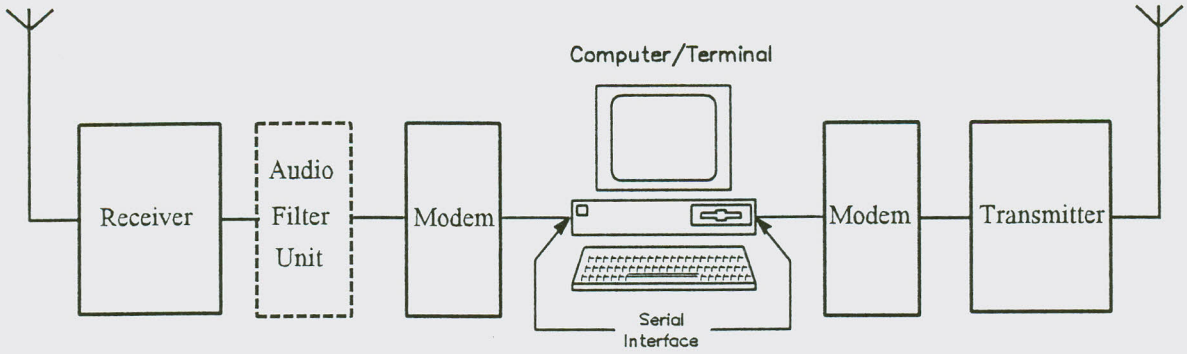


Fig. 1-2 Data communication system over radio with an audio-filter unit

In this thesis work, *designing an audio-filter unit* is considered to be the other countermeasure for the improvement of this shortwave data communication. But the design is restricted to shortwave communication modes such as CW, RTTY and AMTOR signals due to limited access to a wide range shortwave data communication system.

The filter unit may be designed using analog passive or active components. In general, more stringent filter requirements like steeper roll-off in frequency response or minimum distortion in the phase response, call for a complex filter. As a filter gets more complex, it will become necessary to add more inductors and capacitors, there by increasing the sensitivity of the filter's response to small errors in the component values. Thus, precise values of resistance, inductance and capacitance must be maintained if the filter is to operate as designed. Establishing those precise component values is difficult, and so do maintaining them during temperature variation and aging of components.

The above problem can be resolved by realizing the filtering task with *digital filters*, which can be implemented either by software (computer programs) or by means of dedicated hardware.

Implementation of digital filter algorithms by a computer program may still have a problem when dynamic signals are acquired, i.e, the processor will no longer be able to keep up with the flow of data. If the processor cannot keep up with the input data stream, then, invariably data will be lost since in real time systems the backlog of data will continue to build up and no amount of storage will be sufficient. Thus, a specially designed processor is needed to handle the problem efficiently.

The design of the audio-filter unit is this thesis work is, therefore, implemented using a dedicated digital signal processor called TMS32050 DSP kit. This processor has a unique versatility and real time performance that offer a better, more adaptable approach to traditional signal processing problems.

The design of the filter is in such a way that it should have minimum pulse delay distortion. Also, for the received signal to be nearly identical to the transmitted one, the filter's amplitude response should be constant and its phase response should be linear.

The finite impulse response digital filters satisfy such requirements and hence FIR filter types are designed and realized on the DSP kit mentioned above.

Chapter 2

Data Communication

Basically there are two categories of radio communications: the *Analog* and *Digital* type communications. The type of signal transmission determines the choice of these categories. *Radiotelephony* (*voice*) is the most common mode of analog communication to transmit human voice over radio. Under digital radio communications, the following are the major modes of data communication. [2]

- Radiotelegraph (Morse Code)
- Radioteletype (RTTY, AMTOR and SITOR)
- Packet, PACTOR, CLOVER, G-TOR and
- Image (SSTV, FSTV and fax).

Most of the operating modes are incompatible with each other, and each has advantages and disadvantages that make it worthwhile to select the best mode for intended communication. In addition to considering the noise and propagation conditions on the various bands, the user should select the mode of transmission most likely to succeed based on factors like the type of mode the transmitter produce, the type of mode the intended recipient using and the type of mode that will provide the receiving station with the best S/N .

2.1 Radiotelephony (Voice) [2]

Radiotelephone is the transmission of human voice over radio. *Amplitude modulation* was the earliest technique used to transmit voice. It was the dominant phone mode until the late 1940's and 1950's, when single sideband came along. AM uses a full carrier with two modulated sidebands, and takes up a fair amount of bandwidth. In practice, two SSB signals or eight CW signals could be transmitted within the bandwidth used by one AM signal. Because it is less spectrum efficient, it has been reduced to a curiosity on the air.

FM is by far the most popular mode of voice communication over radio. It is a quiet mode, because the technique of angle modulation greatly minimizes the effects of static and noise. The trade-off is that a rather strong signal is needed at the receiver to produce the "quieting" effect that distinguishes FM communications. Amateurs use a form of FM called narrowband FM, which is about 3-20 kHz wide. This is just enough to afford decent voice communication. Commercial FM broadcast stations use wideband (200 kHz) FM, which permits transmitted audio frequencies in the range of 50-12,000 kHz.

Suppressed-carrier single sideband, or simply "single sideband", is another method of transmitting voice over radio. It is an AM signal from which the carrier and one sideband have been removed. The receiving station picks up the SSB signal, adds a carrier, and converts the RF signal back into voice. SSB is useful because it is much easier to copy a weak sideband signal than a weak FM transmission. The signal strength needed to quiet an FM receiver is relatively high. With low signal-to-noise ratios, it's nearly impossible to make out the other operator's voice. In conditions where weak signals are common, SSB can provide intelligible audio through a background of considerable noise.

2.2 Morse Code (CW) [2, 14]

Morse Code telegraphy by *on-off keying* (OOK, *Amplitude-Shift Keying* ASK) of a carrier is the most basic and oldest form of digital radio communication. It is also known as CW[1] (for continuous wave). It involves nothing more than three elements: a short tone, a long tone and a space. In digital perspective, these could be represented by numerals 0,1 and 2. In practice, there would actually be two definite expressions, the 0 and 1, or short tone and long tone. A space by itself would be the same as no signal. Mixing and matching the tones and spaces created a system to denote letters of alphabet, the digits 0-9 and a few punctuation marks. Thus, Morse Code or CW is a series of these tones which can either be transmitted by hand using a morse key or by computer using interface and software.

Language expressed in nothing but long and short tones (dahs and dits) has been in use since before Samuel F.B. Morse devised the American Morse system of telegraphy. In prehistoric times, it was discovered that a prearranged method of sending signals could use a small set of such symbols, yet be potentially powerful means of conveying complex information. The arrangement of the signals (long/short, loud/soft, bright/dim, single/double, on/off) could be used to signify much more than just two alphabetic or numeric symbols. Morse and his colleagues developed a standard code that took advantage of the frequency with which certain English letters are used in language, and made efficient use of the way our ears and mind works.

Perhaps, the one advantage of Morse over Packet and the others, is that a human operator can decode the Morse Code transmission by ear.

---

[1] CW is used by amateurs and other communications to mean OOK telegraphy by Morse Code while parts of the electronic industry use CW to signify an unmodulated carrier.

The speed of Morse telegraphy is usually expressed in *WPM*, rather than baud, which is the common measure in other digital modes. The following formulas relate *WPM* to bauds:

$$WPM = 2.4 \times dot/s \quad (2.1)$$

$$WPM = 1.2 \times B \quad (2.2)$$

where

WPM = telegraph speed in words per minute

2.4 = a constant calculated by comparing dots per second with plain language Morse code sending the word "PARIS"

1.2 = a constant calculated by comparing the signaling rate in bauds with plain language Morse code sending the word "PARIS"

B = telegraph speed in bauds.

For instance, a keying speed of 25 dot/s or 50 bauds is equal to 60 WPM.

Keying a carrier on and off produces double (upper and lower) sidebands corresponding to the period of the keying pulse. A string of dits at 50 baud will have sidebands at multiples of 50 Hz above and below the carrier. The rise time of the pulses affects the distribution of power among the sidebands. As rise time increases or pulse rate decreases, the bandwidth of the signal decreases. In addition, the rise time affects how our ears hear the signal. Publications have long promoted 5-ms rise and fall times for CW keying envelopes as shown in Fig. 2-1. This shape is based on an assumed necessary bandwidth of 150 Hz for a 60 WPM (50-baud) pulse train and an expression relating necessary CW bandwidth to keying speed.

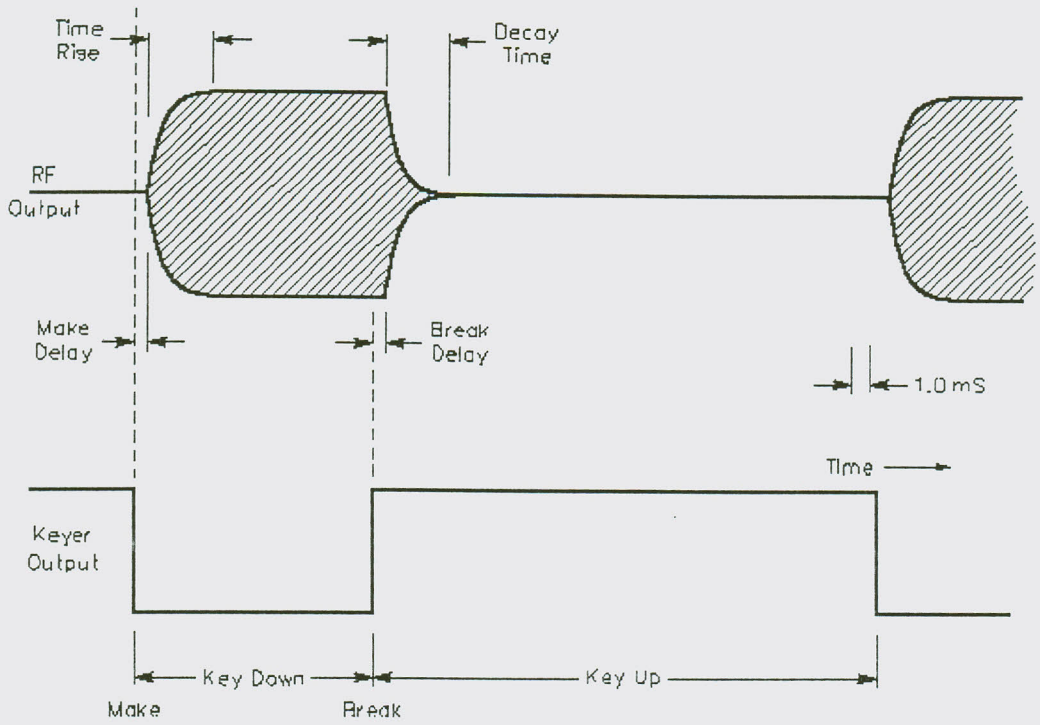


Fig. 2-1 Optimum CW keying waveforms.

For aural reception, a Morse-code OOK RF signal is not completely demodulated to its original dc pulse, because only thumping would be heard. Instead, the signal is moved (by mixing) down to AF, usually near 700 Hz. Proper reception of a Morse-code transmission requires that the receiver bandwidth be at least that of the necessary bandwidth plus any frequency error.

2.3 RTTY [2, 3, 14]

RTTY or Radioteletype is a direct machine to machine communication mode using the Baudot[2] (or Murray[3]) code. It is one of the first data communications codes to receive five bits (traditionally called "levels") to represent the alphabet, numerals, symbols and machine functions.

There are many variations in five-bit coded character sets, principally to accommodate foreign-language alphabets. But, five-bit codes can directly encode only $2^5 = 32$ different symbols. This is insufficient to encode 26 letters, 10 numerals and punctuation. This problem can be solved by using one or more of the codes to select from multiple code-translation tables.

The baudot transmissions must be sent using start-stop pulses as shown in Fig. 2-2. The bits in the figure are arranged as they would appear on an oscilloscope. Each character includes start and stop bit to indicate the beginning and end of that character. This is typical of asynchronous communication.

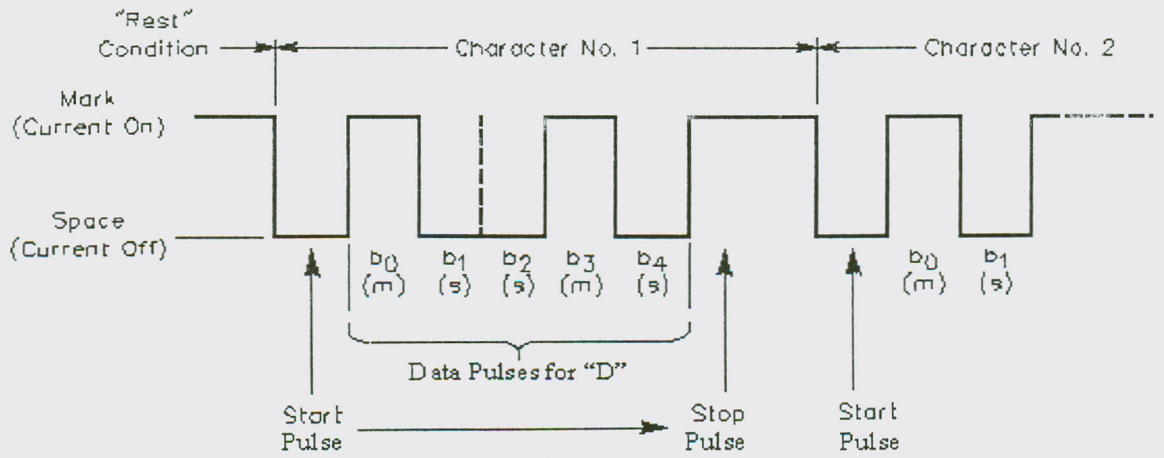


Fig. 2-2 Baudot timing sequence for the letter "D."

---

2 Named after French engineer Emile Baudot and represents the US International Telegraph Alphabet No. 2 (ITA2).

3 Almost identical code used in Great Britain.

The signaling speeds for all forms of RTTY are those used by the old TTYs: 60, 67, 75 or 100 WPM. Table 2-1 relates speeds, signaling rates and pulse times. In practice, the real speeds do not exactly match their names. The names have been rounded through years of common usage.

Table 2-1 Baudot Signaling Rates and Speeds

Signaling Rate (Bauds)	Data Pulse (*ms*)	Stop Pulse (*ms*)	Speed (*WPM*)	Common Name
45.45	22.0	22.0	65.00	Western Union
	22.0	31.0	61.33	"60 Speed"
	22.0	33.0	60.61	45 Bauds
50.00	20	30.0	66.67	European; 50 Bauds
56.92	17.57	25.00	76.68	"75 Speed"
	17.57	26.36	75.89	50 Bauds
74.20	13.47	19.18	100.00	"100 Speed"
	13.47	20.21	98.98	74 Bauds
100.0	10.00	15.00	133.33	100 bauds

There is a problem specifying signaling speed of RTTY because the length of the start and stop pulses vary from that of the data bits. The solution is to base the signaling speed on the shortest pulses used. Thus, *baud* is used as a unit of signaling speed equal to one pulse (event) per second. The signaling rate, in bauds, is the reciprocal of the shortest pulse length.

When TTYs and terminal units roamed the airwaves, frequency-shift keying was the order of the day. DC signals from the terminal units controlled some form of reactance (usually a capacitor or varactor) in a transmitter oscillator stage to shift the transmitter frequency. With FSK, the transmitter is shifted up in frequency every time a *mark* is to be sent, reverting to the lower frequency for a *space*. The amount of the shift is usually 170 Hz for Amateur Radio use although many commercial Teletype signals use other shifts, notably 425Hz and 850 Hz.

Multimode communications processors, however, generally connect to the radio AF input and output, often through the speaker and microphone connectors, sometimes through auxiliary connectors. They simply feed AF tones to the microphone input of an SSB transmitter or transceiver. This is called AFSK for "*Audio Frequency-Shift Keying.*" When it is properly designed and adjusted, this method of modulation cannot be distinguished from FSK on the air.

When using AFSK, make certain that audio distortion, carrier and unwanted sidebands do not cause interference. Particularly when using the low tones the harmonic distortion of the tones should be kept to a few percent. Most modern AFSK generators are of the coherent-phase (CPFSK) type. Older types of noncoherent-FSK (NCFSK) generators had no provisions for phase continuity and produced sharp switching transients. The noise from phase discontinuity caused interference several kilohertz around the RTTY signal.

An AFSK demodulator takes the shifting tones from the audio output of a receiver and produces TTY keying pulses. FM is a common AFSK demodulation method. The signal is first bandpass filtered to remove out-of-band interference and noise. It is then limited to remove amplitude variations. The signal is demodulated in a discriminator. The detector output lowpass filtered to remove noise at frequencies above the keying rate. The result is fed to a circuit that determines whether it is a mark or a space.

AM (limiterless) detectors, when properly designed, permit continuous copy even when the mark or space frequency fades out completely. At 170-Hz shift, however, the mark and space frequencies tend to fade at the same time. For this reason, FM and AM demodulators are comparable at 170-Hz shift. At wider shifts (say 425 Hz and above), the independently fading mark and space can be used to achieve an in-band frequency-diversity effect if the demodulator is capable of processing it. To conserve spectrum, it is generally desirable to stay with 170-Hz shift for 45-baud Baudot and forego the possible in-band frequency-diversity gain. It is good to keep the in-band frequency-diversity gain in mind, however, for higher signaling rates that would justify greater shift.

2.4 AMTOR [2, 14]

AMTOR is a specialized form of RTTY. The term is an acronym for Amateur Teleprinting Over Radio and is derived from the commercial SITOR system developed primarily for Maritime use. AMTOR uses the same mark and space shift frequencies as RTTY, the most commonly used one is 170 Hz.

Unlike the RTTY transmission where one station transmits while the other receives, both stations maintain a link by exchanging transmissions. Hence, AMTOR improves on RTTY by incorporating a simple Error Detection technique. It remains relatively uncomplicated but performs well even in poor HF conditions. It is a partial solution to overcome the incorrect reception of signals when the S/N is inadequate.

The system converts the 5-bit code to a 7-bit code for transmission such that there are 4 mark and 3 space bits in every character. The constant mark/space ratio limits the number of usable combinations to 35. The Baudot takes up 32 of the combinations; the 3 remaining are service information signals.

There are two modes used in AMTOR: ARQ (Automatic Repeat Request) and FEC (Forward Error Correction).

2.4.1 ARQ Mode

This mode, sometimes called Mode A, is a synchronous type protocol, which means that both stations are synchronized to each other's signal. The system transmits a block of three characters from the information sending station to the information receiving station. After each block, the IRS either acknowledges correct receipt (based on the 4/3 mark/space ratio), or requests a repeat. This cycle repeats as shown in Fig. 2-3.

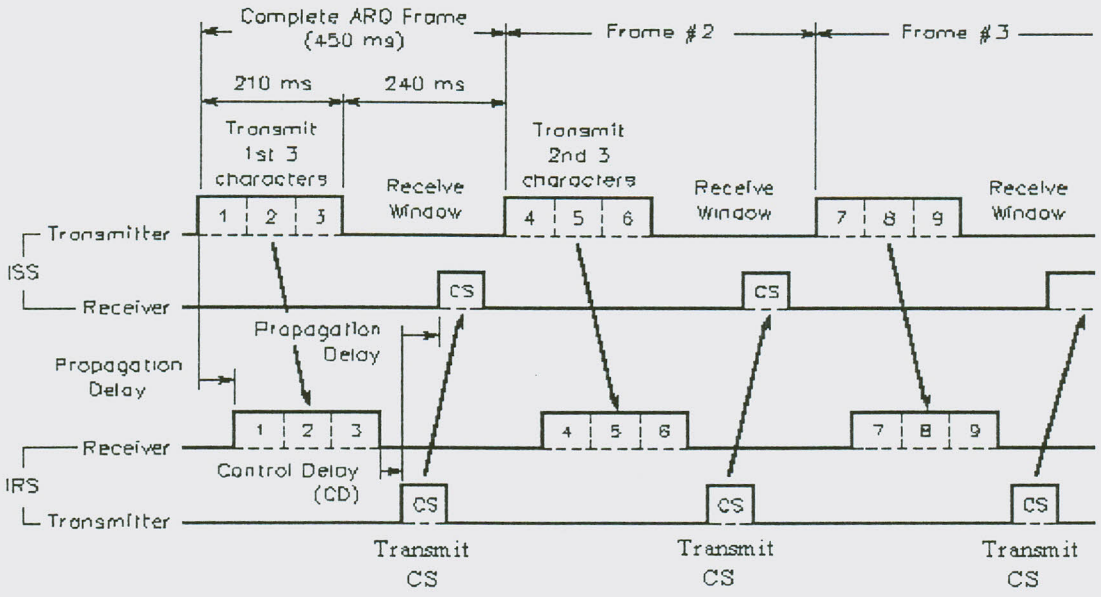


Fig. 2-3 Typical AMTOR timing. Dark arrows indicate the signal path from the ISS to the IRS and vice versa.

The station that initiates the ARQ protocol is known as the Master Station (MS). The MS first sends the selective call of the called station in blocks of three characters, listening between blocks. Four-letter AMTOR calls are normally derived from the first character and the last three letters of the station call sign. The Slave Station recognizes its selective call and answers that it is ready. The MS now becomes the ISS and will send traffic as soon as the IRS says it is ready. When an ISS is done sending, it can enable the other station to become the ISS by sending the three-character sequence. A station ends the contact by sending an "end of communication signal".

On the air, AMTOR *Mode A* signals have a characteristic "chirp-chirp" sound. Even when there is no data actually being transmitted, the transmitting station continues to send idle "chirps" to maintain the link.

2.4.2 FEC Mode

In FEC Mode, sometimes called Mode B, the sending station sends each character twice, so this mode provides a means of transmitting to several station at once. Burst errors are virtually eliminated by delaying the repetition for a period thought to exceed the duration of most noise bursts. In AMTOR, groups of five characters are sent and then repeated. At 70 ms per character, there is 280 ms between the first and second transmissions of a character. The receiving station tests for the constant 4/3 mark/space ratio and prints only unmutilated sent or received characters. If both are mutilated, an error symbol or space prints.

This Mode is still better than an ordinary RTTY but its error detection is not reliable as that in the ARQ Mode.

AMTOR is well suited to traffic handling and passing messages when accuracy is worth a slight trade-off in speed. A disadvantage of AMTOR is that both stations must "hear" each other well, or the communication is reduced to a lengthy exchange of poorly received character groups and NAKs.

2.5 Packet Radio [2, 14]

Packet radio is an error-free mode that uses the complete ASCII character set and supports the transfer of binary data. It is called "packet" because data is not sent as a single, continuous string of characters, but rather transmitted in small bursts, or *packets*. Each packet contains not only the data to be transferred, but also *overhead* information used to route the packets and reassemble them into their original continuous whole. The overhead includes data that identifies the sending and receiving stations, enabling the packet transmission to reach the proper destination. It also provides the required station identification of the sender.

When two packet stations are exchanging data, they are said to be *connected*. One station sends a packet and then waits a specified amount of time for a reply. The receiving station checks the packet for errors and sends an acknowledge signal if the packet is error-free. If not, the receiving station does nothing. When the waiting time expires, the originating station retransmits the packet on the assumption that it did not arrive error-free. Packet stations communicate directly in many cases. If the path is too long to support a direct connection, however, packet relays known as *nodes* or *digipeaters* (packet radio station capable of recognizing and selectively repeating a packet frames) are used.

Packet radio stations are able to communicate with each other because they conform to a standard format, or protocol, described as AX.25[4]. This is a specification derived from a similar protocol, X.25, used by commercial packet networks.

Packet data rates are limited to a maximum of 300 bit/s within the Amateur Radio bands below 28 MHz. Even at this relatively slow rate, noise and interference makes efficient packet communication difficult on the HF bands. Unless conditions are good at both ends of the path, packets must be repeated many times before they arrive error-free. Above 28 MHz, data rates are not so limited. Most VHF packet users operate their systems at 1200 bit/s using 2-m FM. However, networks have been established that use much higher data rates, 9600 bit/s and beyond, to link nodes and packet bulletin board systems over a wide area.

[4] AX.25 is Amateur packet radio link-layer protocol while X.25 is a computer communications protocol over telephone lines.

2.6 PACTOR [2,14]

PACTOR is an HF radio transmission system which combines the best of AMTOR and packet to make a system that is superior to both. It is much faster than AMTOR, yet improves on AMTOR's error-correction scheme. It performs well under both weak-signal and high-noise conditions. PACTOR carries binary data, so it can transfer binary files, ASCII and other symbol sets.

Advantages of PACTOR include:

- An error-correction algorithm called *Memory ARQ*, a method for reconstructing an original block of data by adding together the broken pieces of that block as it is repeated until the block is complete.
- Data-compression techniques (Huffman coding) that can increase the data transfer rate by up to 400% over uncompressed data.
- Compatibility with ASCII and binary data transfers.
- Automatic adjustment of its data rate to compensate for changes in radio conditions.
- Mark or space polarity is inconsequential because it is frequency-shift independent.
- Tolerates interference well, while maintaining the communication link.
- Fast, reliable changes of transmission direction and end of transmission confirmation at both ends of a connection.

Like PACKET and AMTOR, PACTOR is a two-way affair: A transmitting station sends data and a receiving station sends back electronic acknowledgment of each burst of characters. Unlike packet or AMTOR, however, PACTOR dynamically adapts to conditions. Rather than relying on each transmission to provide a solid block of clear characters, PACTOR can accept a series of imperfect or incomplete data segments and intelligently attempt to reassemble them into a solid group. In this way, the number of transmissions is reduced because the receiving station may be able to make out enough detail from two or three successive bursts to provide an errorless segment of data.

2.7 CLOVER [2]

The desire to send data via HF radio at high data rates and the problems that could come across while using AX.25 packet radio on HF radio led to the development of a unique modulation waveform and data transfer protocol called CLOVER.

The CLOVER system has an "AUTO-ARQ" mode that provides a three-pronged attack against the problems caused by HF data signal distortion. This forward-error-correction will correct as many as 31 bytes of erroneous data for every 188 bytes of transmitted data, without requiring repeat transmissions (AMTOR, PACTOR and packet correct errors only by retransmitting data). When erroneous data exceeds 31 of 188 transmitted bytes, only the damaged data blocks are repeated (AMTOR and PACTOR repeat all the data of a transmitted pulse, even for one character error).

There are many advantages and unique features in a CLOVER system. For example, if conditions and propagation permit a high-quality link between two CLOVER stations, they can automatically signal each other to lower power and increase the data-transmission speed. Under poor conditions, a station can request the other to increase power and lower transmission speed to try to reduce errors. But it requires a special plug-in card that goes into a standard expansion slot in an IBM-compatible personal computer.

2.8 G-TOR [2,14]

G-TOR, a short for Golay-TOR, is a new HF digital communication mode for Amateur service. The purpose of the G-TOR protocol is to provide an improved digital radio communication capability for the HF bands. The key features of G-TOR are:

- Standard FSK tone pairs (mark and space)
- Link-quality-based signaling rate: 300, 200 or 100 baud

- 2.4-s transmission cycle
- Low overhead within data frames
- Huffman data compression—two types, on demand
- Embedded run-length data compression
- Golay forward-error-correction coding
- Full-frame data interleaving
- CRC error detection with hybrid ARQ
- Error-tolerant "Fuzzy" acknowledgments

The primary benefit of these innovations is increased throughput, that is, more bits communicated in less time. This is achieved because the advanced processing features of G-TOR provide increased resistance to interference and noise and greatly reduce multipath-induced data errors.

2.9 **Image - FSTV, SSTV and Fax** [2]

Amateur television comes in two main categories: Fast-scan TV and slow-scan TV. ATV allows to send television pictures over the airwaves. SSTV originated earlier; it is a technique used to send color or black-and-white still pictures over HF, using a bandwidth comparable to that used by SSB voice. Because of the large amount of information required to transmit a video image, the bandwidth restrictions for HF operation limit the mode to still pictures. On the other hand, FSTV, is used on the UHF bands, which where much broader bandwidth are possible to use. A UHF FSTV picture can, therefore, be a real-time moving picture.

Fax is an additional form of image communications. It is an abbreviation for *facsimile*, a form of transmission that sends visual "photocopy" of a two-dimensional image, such as a piece of paper, a photograph or diagram. The first fax machines were mechanical devices with paper wrapped around a rotating drum, or scanned by a moving light source. The light and dark spots on the paper (that is blank space and typewritten characters) were converted to electrical impulses. These impulses were sent along wires to a receiving device that converted the impulses back into an image on paper.

Modern amateur faxes are sent using PCs and special software. The images may almost be any computer file format. Images are mostly created by using a device called a scanner to convert a printed image into digital format. This special software converts the image into audio tones that are sent via the transmitter for decoding at the receiving end. The receiving station can display the faxed image on a computer monitor screen or printer. Fax images can also be "captured" as they are received and saved to a disk file for later access (for example viewing, printing, resending, modification and etc.). When a fax transmission begins, the receiving station must be tuned in and listening so that it can receive and decode the synchronization data sent at the beginning of the fax. Without the synchronization data, the receiving station cannot properly interpret the image. The reason is nothing but faxes will take a long time for transmission. So as to enlighten this, operators must know the size of the file that can be faxed in the span of 10 minutes before the transmission requested station identification.

Chapter 3

Communication Interface

This chapter describes the interface between the transceiver and the computer that are used for radio communication. It includes both the software for transmission and reception of the radio signals CW, RTTY, and AMTOR and the interface circuits that connects the transceiver with the computer.

3.1 Interface Circuit [11]

A conventional converter or modem chip is not required to interface the computer with the transceiver. Audio frequency generation and decoding, serial/parallel conversion and all other signal processing is done by a special communication software called *HamComm*.

The audio output of the receiver is connected to the serial port of a computer through a very simple low cost circuit. Only an op-amp, a few diodes, capacitors and resistors are needed to build this circuit. A separate power supply is not also required, the supply current is drawn from the serial port. For transmission, the audio signal from the serial port is connected to the microphone input of the transmitter through a passive R/C filter.

This simple communication interface circuit comprises no filter unit for further reducing the noise level of the incoming signal from the transceiver. The signal directly passes to the *HamComm* for signal decoding. The decoded signal is sometimes meaningless when the noise level prevails the signal and this makes it difficult to sustain communication. Thus, there must be a solution to guarantee a continuous communication and it is *designing an additional filter unit* between this interface circuit and the transceiver.

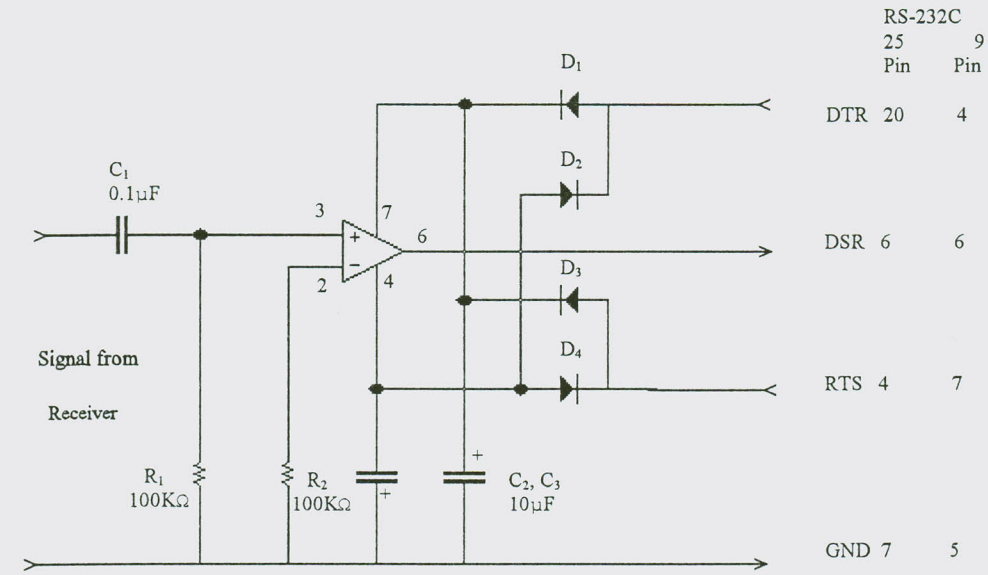


Fig. 3-1 Receive Circuit

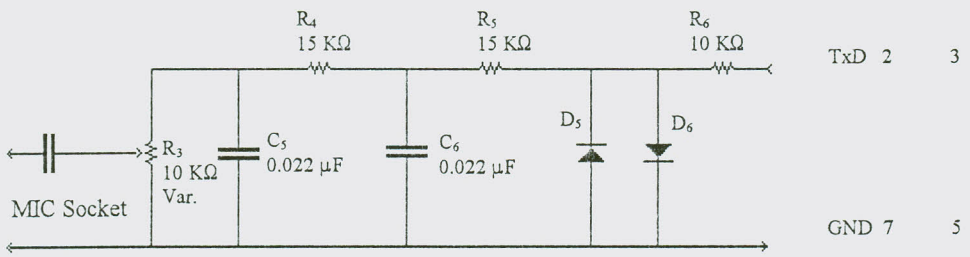


Fig. 3-2 Transmit Audio Circuit

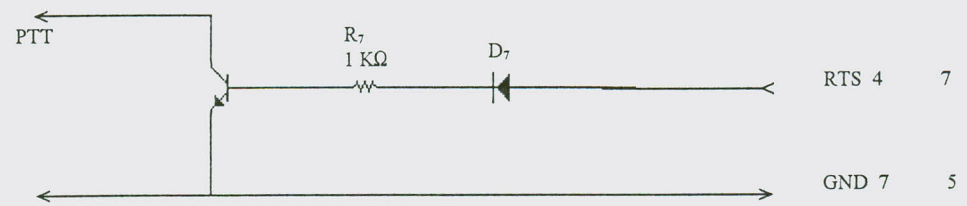


Fig. 3-3 PTT Circuit

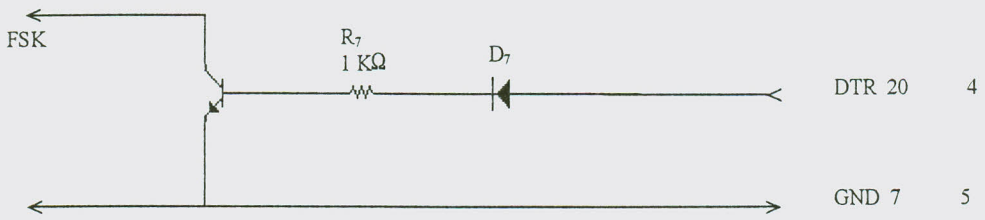


Fig. 3-4 FSK Circuit

Note:

RS-232 Serial Connector

25 Pin	9 Pin	Description
2	3	TxD (Transmit Data)
4	7	RTS (Request to Send)
6	6	DSR (Data Set Ready)
7	5	GND (Signal Ground)
20	4	DTR (Data Terminal Ready)

As shown in the above figures (Fig. 3-1 to 4), the interface circuit has four sections: the *Receive* and *Transmit* circuits, the *PTT* and the *External FSK*. Let's see the function of each circuit briefly.

3.1.1 Receive Circuit

The operational amplifier is used to bring the audio signal from the receiver up to RS232 level. The supply current is drawn from the DTR and RTS pins of the serial port. The four diodes form a standard bridge rectifier. The capacitors are used for buffering. The input signal amplitude should be at least $100mV_{pp}$. The 10nF capacitor removes any DC bias. Since the op-amp runs with maximum gain, there will be more or less a rectangular waveform at its output. It should have an amplitude of at least +/- 5V to reliably drive the RS232 input.

3.1.2 Transmit Circuit

For transmission, an AFSK tone signals are available at the serial port. These signals are then fed to the R/C filter to smooth them out. The microphone input is very sensitive so a variable resistor is used for attenuation and capacitor to remove any DC.

3.1.3 PTT Circuit

The RTS output of the COM port is not only used to provide the supply current for op-amp, but also to key the transmitter. A diode is used to protect the base of the PTT transistor against the negative voltage of RTS output in receive mode. The resistor is used to limit the base current. In transmit mode the RTS and STR pins change polarity. RTS becomes positive and the transistor pulls the PTT line to ground. Caution should be taken since the required current to key the transmitter may be too high.

3.1.4 External AFSK

AFSK tones for transmission are normally available at the TxD pin of the COM port. But it is also possible to use the FSK signal at the DTR pin of the COM port. For RTTY transmission DTR is negative for 'mark' state and positive for 'space' state. In CW mode DTR is negative for 'no-tone' and positive for 'tone', during reception it is always positive. The RTS pin, on the other hand, is negative for receiving mode and always positive during transmission. Thus, it can be used to gate the DTR signal for keying a transmitter in CW mode.

3.2 The HamComm [11]

HamComm is a special program for ham (amateur) radio communication. It supports reception and transmission of radioteletype and Morse code signals. It runs under MS-DOS 3.x of higher on any PC-compatible computer with at least 400KB of free memory. The program will probably not run under any kind of multitasking environment like Desqview, Windows 3.x/95/NT or OS/2 since it requires direct control of the interrupt controller, timer chip and serial I/O hardware.

The HamComm has *scop function*, *spectrum function* and *bitlength function* that can display the various aspects of the incoming signal. It is capable of displaying the graph of the input signal frequency versus time using its *scope function*. The data for the display is obtained from a routine called *tone decoder* which calculates the current signal frequency in use by other parts of the program. When the *spectrum function* is used, the video card and monitor are activated and displays the audio spectrum of the signal. The *bitlength function* is a graphic display of the Mark and Space pulse length. Its main purpose is to give information about the speed of the signal.

3.3 Signal Decoding [11]

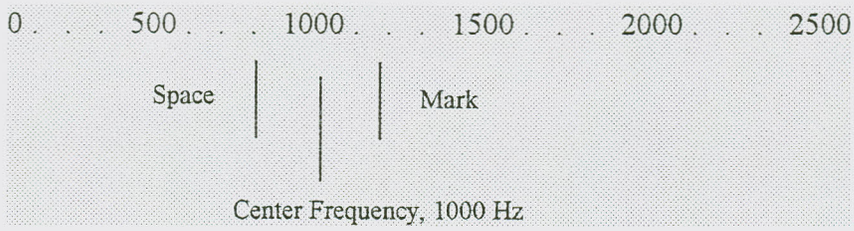
The amplified audio signal is connected to the DSR modem status input of the serial port where every zero crossing generates an interrupt. *HamComm* determines the time between successive interrupts using the PCs timer chip and calculates the corresponding tone frequency. Due to the timer resolution of about one microsecond the result is quite accurate and constitutes the base for all further signal processing.

For RTTY decoding, the tone is compared to the currently selected center frequency to decide whether it is a 'mark' or 'space' signal. The mark/space signal is sampled at proper time according to the current Baud collecting all the bits of a character.

For CW decoding, the program has to differentiate the tone from noise. To be regarded as a valid signal the tone has to be between the currently selected mark/space frequencies for a certain amount of time. The program maintains a floating average of length of the dots and dashes to adjust varying speeds. When the character is completed, it is converted to ASCII code and displayed in the receive window.

3.3.1 RTTY and AMTOR Decoding

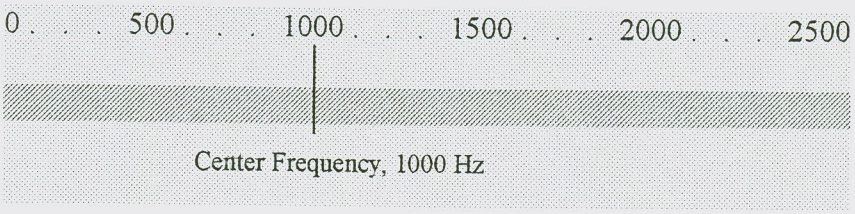
To decode an RTTY and AMTOR signal the center frequency has to be in the middle between the mark and space tones. Also, the peaks of these signals should match the marker lines for the mark and space tones. The center frequency can be varied according to the radio signal reception.



Since RTTY is an asynchronous mode, the sender and receiver do not keep their internal time reference in synchronism. The receiver simply waits for the leading edge of the start bit, collects the data bits according to the signal speed and then waits for the next character. On the other hand, accurate timing is essential to keep the AMTOR link going since it is working in synchronous mode. The FEC and ARQ error detecting modes are also included while decoding the AMTOR signal.

3.3.2 Morse Code (CW) Decoding

To decode a CW transmission the tone frequency of the signal should closely match the currently selected center frequency. The center frequency can be varied according to the signal.



There are two problems encountered with CW decoding in HamComm: *Tone detection* and *Character Decoding*.

a. Tone Detection

Any data transmission requires at least two states for a bit at a time. CW has only a tone and the HamComm requires a certain amount of *noise* to detect that the tone has ended. For this reason CW decoding with HamComm does not work well with narrow filters. There is a tone detector in the HamComm whose output is fed to the character decoder. For the signal to be considered as a valid tone, it has to stay in the currently selected mark/space tones for certain amount of time. Thus, the behavior of the detector depends on the currently selected shift.

b. Character Decoding

Theoretically, a dash is three times as long as a dot, the gaps within a character are dot-sized and the gaps between characters are dash-sized. In reality, there are short and long tones and gaps of variable size because CW is usually 'hand-made'. The speed and length ratio also changes during transmission if the operator gets tired or bored.

HamComm keeps a floating average of the tone duration to adjust to speed changes. If the signal is too noisy, the dots and dashes will get broken into many short ones. The character decoder then gets bursts of very short tones which look like a high-speed CW signal and tries to adjust. Thus, the overall decoding quality mainly depends on the tone detector.

Chapter 4

The TMS320C5X Digital Signal Processor Kit

The TMS320 family consists of 16-bit fixed-point and 32-bit floating-point single-chip digital signal processing device. These processors possess the operational flexibility of high-speed controllers and numerical capability of array processors. Combining these two qualities, the TMS320 processors are inexpensive alternatives to custom-fabricated VLSI and multichip bit-slice processors. The following qualities make this family the ideal choice for wide range applications: [8]

- Very flexible instruction set
- Inherent operational flexibility
- High speed performance
- Innovative, parallel architectural design
- Cost effectiveness

With its unique versatility and real-time performance it offers better, more adaptable approaches to traditional signal-processing problems such as filtering an vocoding. Furthermore, it supports complex applications that often require several operations to be performed simultaneously

In this thesis work, TMS320C50 DSP kit is used for design and implementation of the filter unit to improve the shortwave communication mentioned.

4.1 General Description [8, 9]

The TMS320C5X generation consists of the 'C50, the 'C51, and the 'C53 devices. These digital signal processors (DSPs) are fabricated in accordance with static CMOS, integrated-circuit technology. The combination of advanced Harvard architecture (separate buses for program memory and data memory), additional on-chip peripherals, more on-chip memory, and a highly specialized instruction set is the basis of the operational flexibility and speed of these DSP devices. The 'C5X devices are designed to execute more than 28 MIPS.

The key features of 'C50 are listed below.

- 35-/50-ns single cycle fixed point instruction execution time (28.6/20 MIPS)
- Upward source-code compatible with all 'C1X and C2X devices
- RAM-based memory operation
- 9K x 16 bit single cycle on-chip program/data RAM
- 2K x 16 bit single cycle on-chip boot ROM
- 1056 x 16 bit dual-access on-chip data RAM
- 224K x 16 bit maximum addressable external memory space (64K program, 64K data, 64K I/O and 32K global)
- 32 bit arithmetic logic unit (ALU), 32 bit accumulator (ACC), and 32 bit accumulator buffer (ACCB)
- 16 bit parallel logic unit (PLU)
- 16 x 16 bit parallel multiplier with a 32 bit product capability
- Single cycle multiply/accumulate instructions
- Eight auxiliary registers with dedicated arithmetic unit for indirect addressing
- Eleven context-switch registers (shadow registers) for storing strategic CPU-controlled registers during an interrupt service routine
- Eight level hardware stack
- 0 to 16 bit left and right data barrel-shifters and a 64-bit incremental data shifter
- Two indirectly addressed circular buffers for circular addressing
- Single instruction repeat and block repeat operations for program code

- Block memory move instructions for direct communication between the 'C5X and another serial device
- Time-division multiple-access (TDM) serial port
- Interval timer with period, control and counter registers fir software stop, start, and reset
- 64K parallel I/O ports, 16 of which are memory mapped
- Two high speed serial I/O ports
- Sixteen software-programmable wait state generators for program, data, and I/O memory spaces
- Index addressing mode
- Bit reversal index addressing mode for radix 2-FFTs
- On-chip clock generation

Fig. 4-1 depicts the basic block diagram of the 'C50. It shows the interconnection, which include the host interface, analog interface, and emulation interface. PC communications are via the RS-232 port on the DSK (digital signal processor starter kit) board.

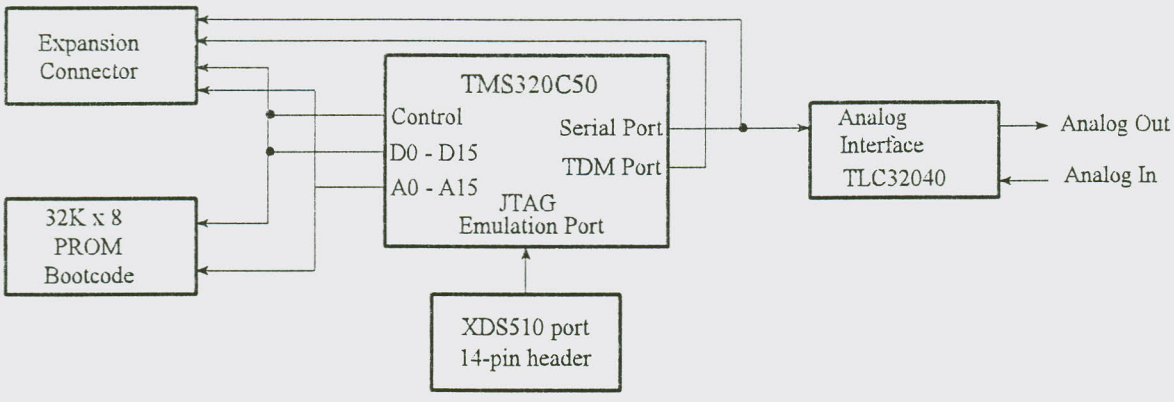


Fig. 4-1 TMS320C50 DSK Diagram

The 32K bytes of PROM contains the kernel program for boot loading. All pins of the 'C50 are connected to the external I/O interfaces. The external I/O interface includes four 24-pin header, a 4-pin header, and a 14-pin XDS510 header. The TLC32040 AIC interfaces to the 'C50 serial port.

4.2 Architecture [8, 9]

The architectural structure a TMS320C5X DSP consists of three basic segments: *Central Processing Unit* (CPU), *Memory* and *Peripheral Interfacing Circuits*.

4.2.1 The Central Processing Unit

The TMS320C5X high performance digital signal processor is designed with an advanced Harvard type architecture that maximizes the processing power by maintaining to separate memory bus structures, program and data, for full speed execution. Instructions support data transfer between the two spaces.

It performs 2's complement arithmetic using the 32 bit *arithmetic logic unit* (ALU) and accumulator. The ALU is a general purpose arithmetic unit that uses 16 bit words taken from data memory or derived from immediate instructions, or the 32 bit result from the multiplier. In addition to arithmetic operations, the ALU can perform Boolean operations. In addition to the main ALU, there is a *parallel logic unit* (PLU) that executes logic operations on data without affecting the contents of the accumulator. The PLU provides the bit manipulation ability required of a high speed controller and simplifies the bit setting, clearing and testing required with control and status registers operations.

The multiplier performs 16 x 16 bit 2's complement multiplication with a 32 bit result in a single instruction cycle. The multiplier consists of three elements: multiplier array, PREG (product register) and TREG0 (temporary register). The 16 bit TREG temporarily stores the multiplicand and the PREG stores the 32 bit product. The multiplier's values come from data memory, comes from program memory when the MAC/MACD/MADS/MADD[5] instructions are used, or are derived immediately from the multiply immediate instructions (MPY #). The fast on-chip multiplier allows the device to efficiently perform fundamental DSP operations such as convolution, correlation and filtering.

---

[5] See Appendix A

The *scaling shifter* has a 16-bit input connected to the data bus and a 32 bit output connected to the ALU. The scaling shifter produces a left shift of 0 to 16 bits on the input data, as programmed in the instruction or defined in the shift count register (TREG1). Additional shift capabilities enable the processor to perform numerical scaling, bit extraction, extended arithmetic and overflow prevention operation.

Eight levels of *hardware stack* save the contents of the program counter during interrupts and subroutine calls. On interrupts, the strategic registers (ACC, ACCB, ARCR, INDEX, PMST, PREG, ST0, ST1, TREGs) are pushed onto a one-deep stack and popped upon interrupt return, thus providing a zero-over-head interrupt context switch.

The functional block diagram shown in Fig. 4-2 outlines the principal blocks and data paths within the 'C5X processor.

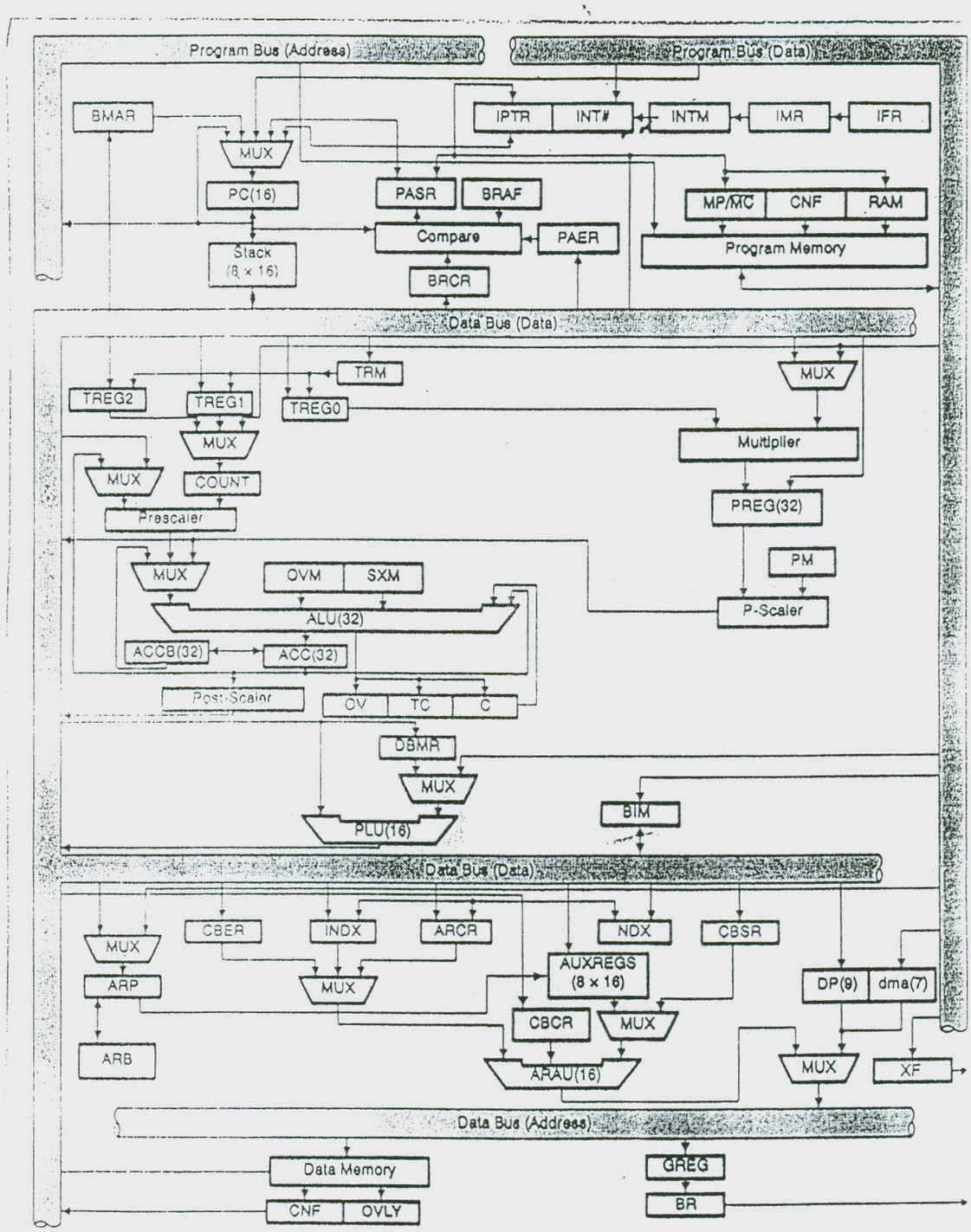


Fig. 4-2 Block diagram of TMS320C50 internal hardware

4.2.2 Memory

The TMS320C5X enhanced Harvard architecture design has multiple memory spaces that can be accessed on three parallel buses. This makes it possible to access both program and data simultaneously. The three parallel buses are *the program read/write bus* (PAB), *data read bus* (DAB1), and *data write bus* (DAB2). Each bus accesses different memory spaces for different aspects of the device operation. The 'C5X memory is organized into four individually selectable spaces: *program*, *local data*, *global data* and *input/output ports* (I/O). These spaces compose an address range of 224K words. Within any of these spaces RAM, ROM, EPROM, EEPROM or memory-mapped peripherals can reside either on- or off-chip.

The program space contains the instructions to be executed as well as tables used in execution. The *local data space* stores data used by the instructions. The *global data space* can share data with other processors within the system or can serve as additional data space. The *I/O space* interfaces to external memory-mapped peripheral can also serve as extra data storage space. Within a given memory machine cycle, the CALU (central arithmetic logic unit) can execute as many as three concurrent memory operations.

The diagram in Fig 4-3 is the memory map of 'C50. Note that when the DSP is in microcomputer mode the internal program ROM is mapped into program memory space whereas in microprocessor mode all the program memory is mapped externally.

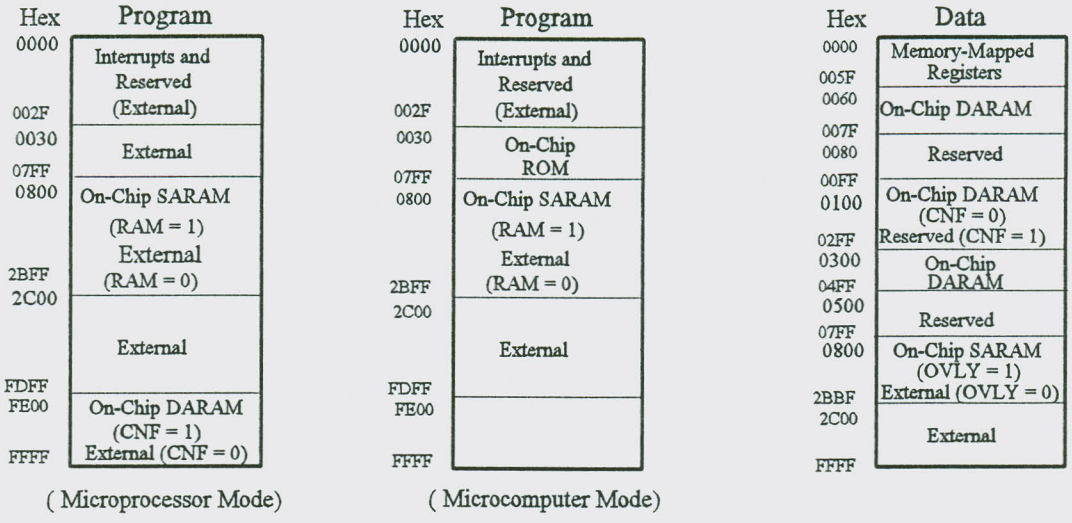


Fig 4-3 TMS320C50 Memory Map

Note that:

SARAM - Single access RAM
DARAM - Dual access RAM
CNF - On-chip RAM configuration bit
OVLY - RAM overlay bit

Memory Addressing Modes

Memory can be addressed in eight different ways in this DSP.

i. Direct Addressing Mode

In this mode the 9 bit DP (data pointer) points to one of 512 pages (1 page = 128 words). The data memory address (dma), specified by the seven LSBs (least significant bits) of the instructions, points to the desired word within the page. the address on the DRB (direct data memory address bus) is formed by concatenating the 9-bit DP with the 7 bit dma.

ii. Memory-Mapped Addressing Mode

This mode operates much like the mode in *(I)* except that the most significant 9 bits of the address are forced to zero instead of being loaded with the contents of the DP. This allows the user to directly address the memory-mapped registers for data page zero without the overhead of changing the DP or auxiliary registers.

iii. Indirect Addressing Mode

In the indirect addressing mode, the currently selected 16-bit auxiliary register AR(ARP) addresses the data memory through the auxiliary register file bus (AFB). While the selected auxiliary register provides the data memory address and the data is manipulated by CALU, the content of the auxiliary register may be manipulated through the ARAU (auxiliary register arithmetic unit).

iv. Short Immediate Mode

The operand may reside as part of the instruction machine code. In the case of the short immediate operand, the operand is contained in the single word instruction.

v. Long Immediate Mode

In this case, the operand immediately follows the opcode in the program sequence. The long immediate operand is 26 bit long.

vi. Register Access Mode

The operand may come from CPU register. This type of operand is used in special cases.

vii. Long Immediate Addressing Mode

In the long immediate addressing mode, an operand is addressed by the second word of two-word instruction. In this case, the program address/data bus (PAB) is used for operand fetch.

viii. Registered Block Memory Addressing Mode

Registered block memory addressing mode operates like the long immediate addressing mode with the exception that the address comes from BMAR (block move address register).

4.2.3 Peripheral Interface Circuits

The peripheral interface circuits connected to the TMS320C5X core CPU are the serial port, TDM serial port, timer, software-programmable wait state generator, I/O ports, divide-by-one clock and XF(external flag) and $\overline{BIO}$ (branch control input). These peripherals are controlled through registers that reside in memory map. The operation of serial ports and timer is synchronized to the core CPU via interrupts or through interrupt polling. Setting and clearing bit can enable, disable, initialize and dynamically reconfigure the peripherals. Data is transferred to and from the peripherals through memory mapped data registers. When a peripheral is not in use, the internal clocks are shut off from that peripherals, allowing for lower power consumption when the device is in normal run mode or idle mode.

The full duplex (bidirectional) on-chip serial port provides direct communication with serial devices such as codecs, serial A/D converters and other serial systems. The serial port may also be used for intercommunication between processors in multiprocessing applications (the TDM port is further optimized for such an application).

Chapter 5

Digital Filters

5.1 Introduction

Almost every field of science and engineering, such as acoustics, physics, telecommunications, control systems and radar, deal with signals. These signals, in general, are classified into *continuous-time* and *discrete-time* signals.

A *Continuous-time signal* is one that is defined at each and every instant of time. A *discrete-time signal*, the other hand, is one that is defined at a discrete instant of time. Both of these signals can be represented by a unique function of frequency called *frequency spectrum* of the signal.

Filtering is a process by which the frequency spectrum of a signal can be modified, reshaped, or, manipulated according to some desired specification. It may entail amplifying or attenuating a range of frequency components, rejecting or isolating one specific frequency component, etc.

Any system or network that exhibits such frequency selective characteristic is called *a filter*. Several types of filters can be identified: *lowpass filters* (LPF) that passes only "low" frequencies, *bandpass* (BPF) that passes a band of frequencies, *high pass filters* (HPF) that passes "high" frequencies and *band-reject filters* that rejects certain frequencies.

Filters are used in variety of applications, such as removing noise from a signal, removing signal distortion due to the transmission channel, separating two or more distinct signals that were mixed in order to maximize communication channel utilization, demodulating signals and converting discrete-time signals into continues-time signals.

A digital filter is a digital system that can be used to filter discrete-time signals. It can be implemented by means of software or by means of dedicated hardware. In either cases, it can be used to filter real-time signals or non real-time (recorded) signals.

5.2 Advantages of Digital Filters

The term "digital filtering" refers to the computational process or algorithm by which a digital signal or sequence of numbers (acting as input) is transformed into a second sequence of numbers termed as the output digital signal. Digital filters involve signals in digital domain (discrete-time signal), whereas analog filters relate signals in the analog domain (continuous-time signal).

Digital filters are used extensively in application, such as digital image processing, pattern recognition, and spectrum analysis. A band-limited continuous-time signal can be converted to a discrete-time signal by means of sampling. After processing, the discrete-time signal can be converted back to continuous-time signal. Some of the advantages of using digital filters over their analog counterparts are:

- High reliability
- High accuracy
- No effect of component drift and spurious environmental signals on the system performance
- Component tolerance not critical
- Small physical size

Another important advantage of digital filters, when implemented with a programmable processor such as the TMS320 family, is the ease of changing filter parameters to modify the filter characteristics. This feature allows the design engineer to effectively and easily upgrade or update the characteristics of the designed filter due to changes in the application environment.

5.3 Digital Filter Specifications

The design of a digital filter comprises four general steps:

1. Approximation
2. Realization
3. Study of arithmetic errors
4. Implementation

The approximation step is the process of generating a transfer function satisfying a set of desired specification. The realization step is the process of converting the transfer function into a filter network. Approximation and realization assume an infinite precision device for implementation. However, implementation is concerned with the actual hardware circuit or software coding of filter using a programmable processor. Since practical devices are of finite precision, it is necessary to study the effects of arithmetic errors on the filter response.

The requirements of a digital filter are normally specified in the frequency domain in terms of the desired magnitude response and/or the desired phase (delay) response. In lowpass filter case, the desired magnitude response is usually given by

$$D(f) = \begin{cases} 1 & for \ \ f \in [0, f_p] \\ 0 & for \ \ f \in [f_s, \pi] \end{cases} \qquad (5.1)$$

and the specification are given for realizable magnitude response |H(f)| as shown in Fig. 5-1.

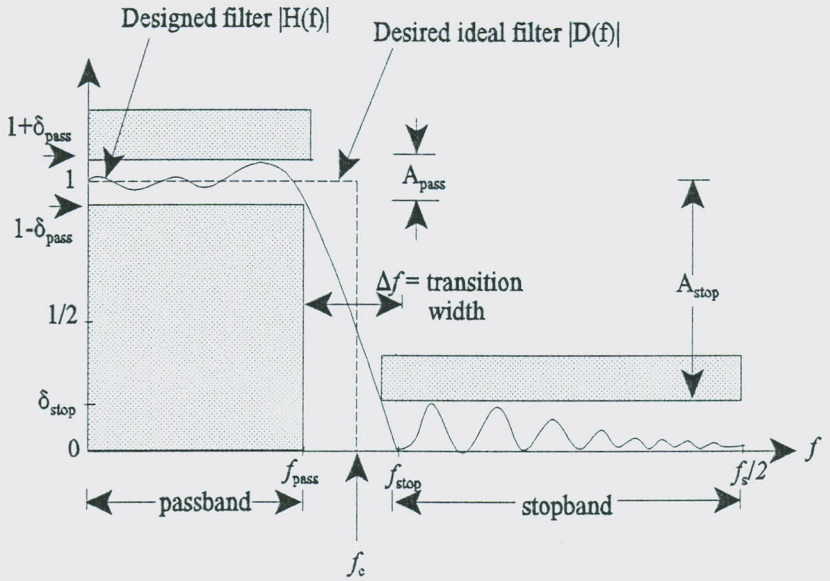


Fig. 5-1 Magnitude response specification for a lowpass filter

It is desired to preserve signal components in the region $[0, f_{pass}]$, called the *passband* of the filter, and to reject signal components in the region $[f_{stop}, \pi]$, called the *stopband* of the filter. f_{pass} and f_{stop} are called the passband edge and stopband edge frequencies, respectively. The permissible errors in the passband and in the stopband are δ_{pass} and δ_{stop}, respectively. To make it possible to approximate the desired function as close as possible with the ideal one, the specification includes a *transition band* of nonzero width, Δf, in which the filter response changes from unity in the passband to zero in the stopband.

Usually, the amplitudes of the allowable ripples for the magnitude response are given logarithmically (in decibels) in terms of the maximum passband variation and the minimum stopband attenuation, which are given by

$$A_{pass} = 20 \log_{10}\left(\frac{1 + \delta_{pass}}{1 - \delta_{pass}}\right) \quad [dB] \tag{5.2a}$$

and

$$A_{stop} = -20 \log_{10}(\delta_{stop}) \quad [dB] \tag{5.2b}$$

Both the A_{pass} and A_{stop} quantities shown above are positive. Another commonly used passband specification is the peak deviation from unity expressed logarithmically as

$$A_{pass} = 20 \log_{10}(\delta_{pass}) \quad [\text{dB}] \tag{5.3a}$$

$$A_{stop} = 20 \log_{10}(\delta_{stop}) \quad [\text{dB}] \tag{5.3b}$$

both of which are negative quantities.

In some applications, it is necessary to preserve the shape of the input signal. This is achieve if the the phase response of the filter is approximately linear in the passband region [0, ω_{pass}]; i.e., arg $H(e^{j\omega})$ approximates on [0, ω_{pass}] of the linear curve

$$\phi(\omega) = -\tau_o \omega + \tau_1 \tag{5.4}$$

where ω is the angular frequency which is normalized by the sampling frequency, f_s, of the filter and τ_0 and τ_1 are constants that can freely be chosen.

Instead of the phase response, the criteria for phase are usually given in terms of the *group delay* response

$$\tau_g(\omega) = -\frac{d}{d\omega}\left(arg\, H(e^{j\omega})\right) \tag{5.5}$$

or the phase delay response

$$\tau_p(\omega) = -\frac{arg H(e^{j\omega})}{\omega} \tag{5.6}$$

These responses have simpler representation forms than the phase response and are often easier to interpret. For a constant phase delay, τ_1 is forced to be zero in Eq. 3.4, whereas in the case of a constant group delay, τ_1 can take any value.

5.4 Types of Digital Filters

Digital filers are classified into FIR (finite impulse response) filters and IIR (infinite impulse response) filters.

An FIR filter has impulse response $h(n)$ that extends only over a finite time interval, say $0 \leq n \leq M$, and is identical to zero beyond that:

$$\{h_0, h_1, h_2, \ldots, h_M, 0, 0, 0, \ldots\} \tag{5.7}$$

M is referred to as the *filter order*. The impulse response coefficients $\{h_0, h_1, h_2, \ldots, h_M\}$ are also called *filter coefficients, filter weights* or *filter taps*. The input/output equation is obtained as a weighted sum of the present input sample $x(n)$ and the past M samples $x(n - 1)$, $x(n - 2)$, $\ldots, x(n - M)$. i.e.,

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) + \ldots + h_M x(n-M) \tag{5.8a}$$

$$y(n) = \sum_{m=0}^{M} h(m)\, x(n-m) \tag{5.8b}$$

An IIR filter, on the other hand, has an impulse response $h(n)$ of infinite duration, defined over the interval $0 \leq n < \infty$. The input/output equation is given by

$$y(n) = \sum_{m=0}^{\infty} h(m)\, x(n-m) \tag{5.9}$$

This I/O equation is not computationally feasible because it is not possible to deal with an infinite number of terms. Thus, the filter coefficients are coupled to each other through *constant-coefficient linear difference equation* allowing an efficient *recursive* computation of the output. And hence, IIR filters are also called *recursive filters.*

More generally, the IIR filters which have impulse response $h(n)$ that satisfy constant-coefficient linear difference equation of the general type

$$h(n) = \sum_{i=1}^{M} a_i\, h(n-i) + \sum_{i=0}^{L} b_i\, \delta(n-i) \tag{5.10}$$

can be reassembled into the same difference equation for $y(n)$ in terms of $x(n)$ as

$$y(n) = \sum_{i=1}^{M} a_i y(n-i) + \sum_{i=0}^{L} b_i x(n-i) \qquad (5.11)$$

where M is the filter order and L is the length of the impulse response vector.

In many digital signal processing applications, FIR filters are preferred over their IIR counterpart. The main advantages of FIR filter designs over their IIR equivalents are the following:

1. FIR filters with exactly linear phase characteristic can easily be designed.
2. There exist computationally efficient realizations for implementing FIR filters. These includes both recursive and nonrecursive realizations.
3. FIR filters realized nonrecursively are inherently stable and free of limit cycle oscillations when implemented on a finite-wordlength digital system.
4. Excellent design methods are available for various kinds of FIR filters with *arbitrary* specifications. The design of arbitrary magnitude IIR filters is usually time-consuming and the convergence to the optimum solution is not always guaranteed.
5. The output noise due to multiplication roundoff errors in an FIR filter is usually very low and the sensitivity to variations in the filter coefficients is also low.

The main disadvantage of conventional FIR filter designs is that they require, especially in applications demanding narrow transition bands, considerably more arithmetic operations and hardware components, such as multipliers, adders and delay elements than do comparable IIR filters. As the transition width of an FIR filter is made narrow, the filter order, and correspondingly the arithmetic complexity, increases inversely proportional to the width. This makes implementation of narrow transition band FIR filters very costly. The cost of implementation of an FIR filter can, however, be reduced by using multiplier-efficient realizations, fast convolution algorithms and multirate filtering.

Coming to this thesis work problem, for the received signal to be nearly identical to the transmitted one, the filter's amplitude response should be *constant* and its phase response should be *linear* with *minimum delay distortion*. This is true because a signal is made up of different frequency components. An ideal transmitting system should delay each frequency component equally, and if the frequency components are delayed by different amounts, the reconstruction of the output signal from its Fourier component would produce a signal of different shape as the input signal. Thus, delay distortion is an essential design consideration.

Such considerations are satisfied with the FIR type filters. Computational difficulty with high order FIR filters is not a problem in this thesis work since it can efficiently be handled by the TMS320C50 digital signal processor. Thus, the next chapter concentrates only on FIR type digital filter design techniques.

5.5 Notch Filters [7]

By determining the pole/zero placement a special application filters such as notch filters, smoothers and resonators can be designed. A notch filter can be constructed with notches at an arbitrary (finite) set of frequencies according to the a polynomial $N(z)$. The notch polynomial $N(z)$ is defined as a polynomial whose zeros are on the unit circle at the desired notch locations.

$$N(z) = \prod_{i=1}^{M} (1 - e^{i\omega_i} z^{-1}) \tag{5.11}$$

where M is the number of notches at the desired notch frequencies ω_i, $i = 1, 2, ..., M$.

Notch filters are used to cancel periodic interferences such as the power frequency pick up and its harmonics. They are also used to enhance signals in noise. In this thesis work, the notch filters are used in enhancing the RTTY and AMTOR signals in noise.

Chapter 6

FIR Filter Design

A filter design problem is the problem of constructing the transfer function of a filter that meets prescribed frequency response specifications. The input to any filter design method is the set of desired specifications and the output is the finite impulse response coefficient vectors $\mathbf{h} = [h_0, h_1, \ldots, h_{N-1}]$ in the case of FIR filters, or the numerator and denominator coefficient vectors $\mathbf{b} = [b_0, b_1, \ldots, b_M]$, $\mathbf{a} = [1, a_1, \ldots, a_M]$ in the case of IIR filters. This chapter is devoted to the *windowing* methods of FIR filter design.

6.1 FIR Filter Design by Windowing [6, 7]

The straightforward approach to designing FIR filters is to determine the infinite-duration impulse response by expanding the frequency response of an ideal filter in a Fourier series then to truncate and smooth this response using a window function.

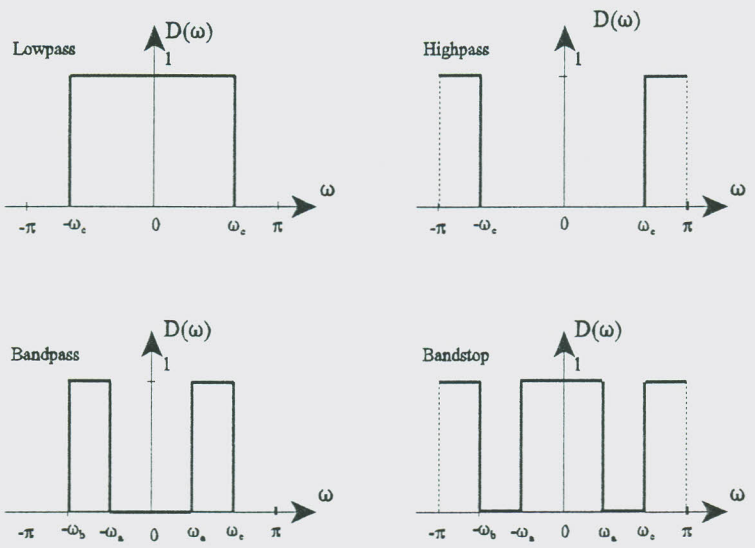


Fig. 6-1 Ideal lowpass, highpass, bandpass and bandstop filters.

A desired frequency response, say $D(\omega)$, being periodic in ω with period 2π, need only be specified over one complete Nyquist interval $-\pi \le \omega \le \pi$. The corresponding impulse response, say $d(k)$, is related to $D(\omega)$ by the DTFT (discrete time Fourier transform) and inverse DTFT relationships:

$$D(\omega) = \sum_{k=-\infty}^{\infty} d(k) e^{-j\omega k} \quad \Leftrightarrow \quad d(k) = \int_{-\pi}^{\pi} D(\omega) e^{j\omega k} \frac{d\omega}{2\pi} \tag{6.1}$$

For lowpass filter shown in Fig.6-1, for example, the quantity $D(\omega)$ is defined over the Nyquist interval by

$$D(\omega) = \begin{cases} 1, & if \quad -\omega_c \le \omega \le \omega_c \\ 0, & if \quad -\pi \le \omega < -\omega_c, \quad or \quad \omega_c < \omega \le \pi \end{cases} \tag{6.2}$$

Thus $d(k)$ is obtained as

$$d(k) = \frac{\sin(\omega_c k)}{\pi k}, \qquad -\infty < k < \infty \tag{6.3}$$

Similarly, an expression for $d(k)$ can be obtained for the highpass, bandpass and bandstop filters of Fig. 6-1 defined over $-\infty < k < \infty$

(highpass filter) $$d(k) = \delta(k) - \frac{\sin(\omega_c k)}{\pi k} \tag{6.4}$$

(bandpass filter) $$d(k) = \frac{\sin(\omega_b k) - \sin(\omega_a k)}{\pi k} \tag{6.5}$$

(bandstop filter) $$d(k) = \delta(k) - \frac{\sin(\omega_b k) - \sin(\omega_a k)}{\pi k} \tag{6.6}$$

In frequency domain, the symmetric types are characterized by a frequency response $D(\omega)$ which is *real* and *even* in ω; the antisymmetric ones have $D(\omega)$ which is *imaginary* and *odd* in ω. One of the main consequences of these frequency properties is the *linear phase* property of the window design.

The window method of filter design consists of truncating and smoothing the ideal filter frequency response obtained. The most frequently used window functions are: *Rectangular*, *Hann*, *Hamming*, *Blackman* and *Kaiser*. In this chapter only the rectangular, Hamming and Kaiser window functions are discussed and compared.

6.1.1 Rectangular Window

Consider the double side $d(k)$ given in Eq.(6.1) for $-M \leq k \leq M$. The total number of coefficients is, thus, $N = 2M+1$. The resulting dimensional coefficient vector is the FIR *impulse response* approximating the ideal response:

$$\mathbf{d} = [d_{-M}, \ldots, d_{-2}, d_{-1}, d_0, d_1, d_2, \ldots, d_M] \tag{6.7}$$

The time origin $k = 0$ is at the middle d_0 of this vector. To make the filter causal, the time origin may be shifted to the left of the vector and re-index the entries accordingly:

$$\mathbf{h} = [h_0, \ldots, h_{M-2}, h_{M-1}, h_M, h_{M+1}, h_{M+2}, \ldots, h_{2M}] \tag{6.8}$$

where $h_0 = d_{-M}$, $h_1 = d_{-M+1}$, . . ., $h_M = d_0$, . . ., $h_{2M} = d_M$. Thus, the vector **d** and **h** are the same, with the understanding that **d**'s origin is in its middle and **h**'s at its left. In other terms,

$$h(n) = d(n - M), \qquad n = 0, 1, \ldots, N - 1 \tag{6.9}$$

Once the impulse response coefficients are calculated, the filter may be implemented by its FIR filtering equation given by Eq. (5.7b). The length-N impulse response $h(n)$ is rectangularly windowed double-sided sequence defined by

$$h(n) = w(n)\, d(n - M), \qquad -\infty \le n \le \infty \tag{6.10}$$

where $w(n)$ is the length-N rectangular window. In frequency domain, the FIR approximation to $D(\omega)$ is equivalent to truncating the DTFT Fourier series expansion to the finite sum:

$$\hat{D}(\omega) = \sum_{k=-M}^{M} d(k) e^{-j\omega k} \tag{6.11}$$

and frequency response:

$$H(\omega) = e^{-j\omega M} \hat{D}(\omega) = e^{-j\omega M} \sum_{k=-M}^{M} d(k) e^{-j\omega k} \tag{6.12}$$

As N increases $\hat{D}(\omega) \rightarrow D(\omega)$. This is true for any ω which is a point of continuity of $D(\omega)$, but it fails at points of discontinuity, such as at the transition edges from passband to stopband. Around this edge the *Gibbs phenomenon* of Fourier series is encountered, which causes the approximation to be bad regardless of how large N is. In other words, the largest ripples tend

to cluster near the passband-to-stopband discontinuity and do not get smaller with N. Instead, their size remains approximately constant, about 8.9%, independent of N.

6.1.2 Hamming Window

To eliminate the 8.9% passband and stopband ripples, the rectangular window $w(n)$ may be replaced by a non-rectangular one, which tapers off gradually at its endpoints, thus reducing the ripple effect. This non-rectangular window is defined as

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, \ldots, N-1 \tag{6.13}$$

and is called *Hamming* window. The Hamming windowed impulse response for a length-N lowpass filter will be

$$h(n) = w(n)d(n - M) = \left[0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)\right] \cdot \frac{\sin(\omega_c(n - M))}{\pi(n - M)} \tag{6.14}$$

The passband/stopband ripples of the rectangular window design are virtually eliminated from the Hamming window design. Actually, there are small ripples with maximum overshoot of about 0.2%. The price of eliminating the ripples is loss of resolution, which is reflected into a wider transition width.

6.1.3 Kaiser Window

The rectangular and Hamming window designs are simple but do not provide a good control over the filter design specifications. With these windows, the amount of overshoot is always fixed to 8.9% and 0.2% and cannot be changed to a smaller value if so desired.

A flexible set of specifications is shown in Fig. 5-1 in which it is possible to arbitrarily specify the amount of passband and stopband overshoot δ_{pass}, δ_{stop} as well as the transition width Δf.

The passband/stopband frequencies $\{f_{pass}, f_{stop}\}$ are related to the ideal cutoff frequency f_c and transition bandwidth Δf by

$$f_c = \frac{1}{2}(f_{pass} + f_{stop}) \quad , \qquad \Delta f = f_{stop} - f_{pass} \tag{6.15}$$

Thus, f_c is chosen to lie exactly in the middle between f_{pass}, and f_{stop}. Eq. (6.15) can be inverted to give

$$f_{pass} = f_c - \frac{1}{2}\Delta f \quad , \qquad f_{stop} = f_c + \frac{1}{2}\Delta f \tag{6.16}$$

The normalized version of the frequencies are digital frequencies:

$$\omega_{pass} = \frac{2\pi f_{pass}}{f_s} \quad , \quad \omega_{stop} = \frac{2\pi f_{stop}}{f_s} \quad , \quad \omega_c = \frac{2\pi f_c}{s} \quad , \quad \Delta\omega = \frac{2\pi\Delta f}{f_s} \tag{6.17}$$

In practice, the passband and stopband overshoots are usually expressed in dB as in Eq. (5.2a) and Eq.(5.2b). From these equations, an expression for the overshoots δ_{pass} and δ_{stop} can be obtained as

$$\delta_{pass} = \frac{10^{\frac{A_{pass}}{20}} - 1}{10^{\frac{A_{pass}}{20}} + 1} \quad , \qquad \delta_{stop} = 10^{-\frac{A_{stop}}{20}} \tag{6.18}$$

Thus one can back and forth between the specification sets:

$$\{f_{pass}, f_{stop}, A_{pass}, A_{stop}\} \quad \Leftrightarrow \quad \{f_c, \Delta f, \delta_{pass}, \delta_{stop}\} \tag{6.19}$$

Although δ_{pass} and δ_{stop} can be specified independently of each other, it is a property of all window designs that the final designed filter will have equal passband and stopband ripples.

Therefore, the ripple is chosen on the basis of the two ripples, i.e.,

$$\delta = \min(\delta_{stop}, \delta_{pass}) \tag{6.20}$$

The value of δ can also be expressed in dB as

$$A = -20\log_{10}{}^{\delta}, \qquad \delta = 10^{-A/20} \tag{6.21}$$

The main limitation of windows is that they have a fixed value of δ, which depends on the particular window shape. Such windows limit the achievable passband and stopband attenuations $\{A_{pass}, A_{stop}\}$ to only certain values. The only windows that do not suffer from the above limitation are the Kaiser window, the Dolph-Chebyshev window and the Saramäki windows. These windows have adjustable shape parameter that allows the window to achieve any desired value of ripple δ or attenuation A.

The Kaiser window is unique in that it has near-optimum performance (in the sense of minimizing the sidelobe energy of the window), as well has having simpler implementation. It depends on two parameters: the length N and the shape parameter α. Assuming odd length $N = 2M + 1$, the window is defined, for $n = 0, 1, \ldots, N - 1$, as follows:

$$(Kaiser\ window) \qquad w(n) = \frac{I_o\left(\alpha\sqrt{1 - (n - M)^2/M^2}\right)}{I_o(\alpha)} \tag{6.22}$$

where $I_o(x)$ is the *modified Bessel function of the first kind and* 0^{th} *order*, given by

$$I_o(x) = \sum_{k=0}^{\infty} \left[\frac{(x/2)^k}{k!} \right]^2 , \qquad for \quad 0 \le x \le \alpha \tag{6.23}$$

Like all window functions, the Kaiser window is symmetric about its middle, $n = M$, and has the value $w(M)$ there. At the endpoints, $n = 0$ and $n = N - 1$, it has the value $1/I_o(\alpha)$ because $I_o(0) = 1$. Fig. 6-2 shows a Kaiser window of length $N = 51$ and shape parameter $\alpha = 7$. The figure also shows a Hamming window of the same length $N = 51$.

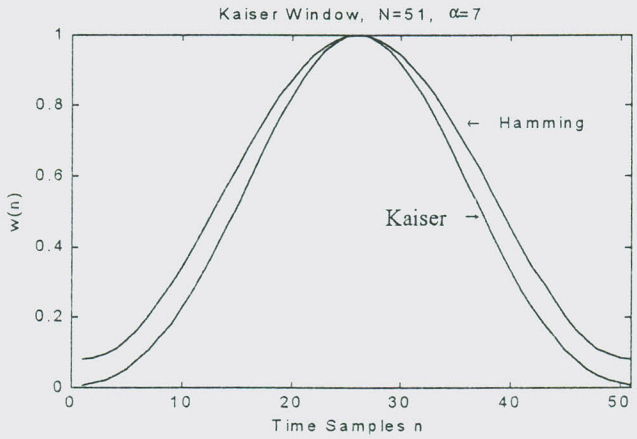


Fig. 6-2 Kaiser and Hamming Windows for $N = 51$ and $\alpha = 7$

The window parameters $\{N, \alpha\}$ are computable in terms of the filter specifications, ripple δ and transition width Δf. The shape parameter α is calculated from:

$$\alpha = \begin{cases} 0.1102(A - 8.7) , & if \quad A \ge 50 \\ 0.5842(A-21)^{0}.4 + 0.07886(A - 21) , & if \quad 21 < A < 50 \\ 0 , & if \quad A \le 21 \end{cases} \tag{6.24}$$

where A is the ripple in dB, given by Eq.(5.2a) and (5.2b). The filter length N is inversely related to the transition bandwidth:

$$\Delta f = \frac{Df_s}{N - 1} \tag{6.25}$$

where the factor D is computed in terms of A by

$$D = \begin{cases} \dfrac{A - 7.95}{14.36}, & if \ A > 21 \\ 0.922, & if \ A \leq 21 \end{cases} \tag{6.26}$$

Thus, for a lowpass filter, the windowed impulse response will be

$$h(n) = w(n)d(n - M) = w(n).\ \frac{\sin(\omega_c(n - M))}{\pi(n - M)} \tag{6.27}$$

for $n = 0, 1, \ldots, N - 1$. The design can be modified to design highpass and bandpass filters.

Fig. 6-3 shows the corresponding Kaiser, Hamming and Rectangular window lowpass filter design for with the same order $N = 103$ and a specification of $f_s = 20$ kHz, $f_{pass} = 4$ kHz, $f_{stop} = 5$ kHz, $A_{pass} = 0.1$ dB and $A_{stop} = 80$ dB.

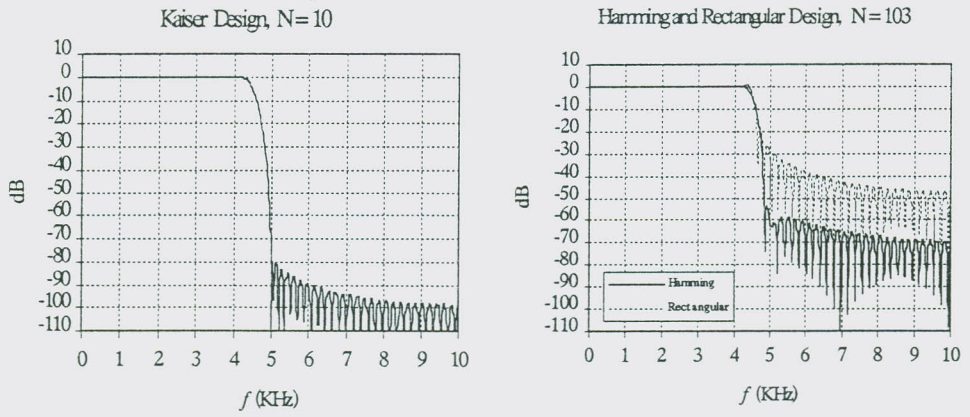


Fig. 6-3 Kaiser, Hamming and rectangular window designs

As can be seen the Kaiser window gives the best design result for the given specifications. Therefore, the *Kaiser window design method* is chosen in this thesis work, for the design and realization of the filter unit.

Chapter 7

Implementation and Result

This chapter describes the design and implementation procedures followed for the audio-frequency filter unit. It also compares the expected output with that of the actual and gives the relevant explanations.

7.1 Determination of Filter Coefficients [5, 6, 7]

As tried to note in the introductory chapter, the design of the filter unit is confined to radio signals CW, RTTY, and AMTOR due to limited access of radio data communication system. These signals are decoded by the HamComm in a manner that the tone frequency should closely match the currently selected center frequency in the case of CW signals and between the marker and space tone frequencies in the case of RTTY and AMTOR signals. Thus, the designed filter got the name "audio-frequency filter" unit due to this center frequency selection in the audio range between 300 Hz and 3000 Hz.

Once the center frequency is selected between the specified audio range, all frequency components other than the tone frequency of CW signal or the marker and space tone frequencies of RTTY and AMTOR, which are 170 Hz apart, should be filtered out. Otherwise, the HamComm will consider the other frequencies as part of the useful signal and begin to decode by counting the zero crossings. The end result of this decoding is a data full of characters with no meaning. Therefore, the audio-frequency filter unit should have a sharp transition frequency and narrow bandwidth characteristics with noise suppression level of greater than 60 dB in the stopband region to reject the unwanted frequency components. An

even better result can be obtained for the RTTY and AMTOR signals if the unwanted frequencies between the mark and space are rejected by designing a notch filter.

The Kaiser window design technique is chosen, as indicated in the previous chapter, to generate the filter coefficients for the required filter specification. In general, the following design procedures can be followed to design a bandpass filter with $\{f_c, A_{pass}, A_{stop}, \Delta f\}$ specified.

1. Calculate f_{pass}, f_{stop} and ω_c using Eqns. (6.16) and (6.17).
2. Calculate δ_{pass} and δ_{stop} from Eqn.(6.18).
3. Calculate $\delta = \min(\delta_{pass}, \delta_{stop})$ and $A = -20 \log \delta$ in dB.
4. Calculate the shape factor α and D using Eqns. (6.24) and (6.26).
5. Determine the order of the filter N from Eqn. (6.25) and round it up to the next odd integer, $N = 2M + 1$ and set $M = (N - 1)/2$.
6. Calculate the window function $w(n)$, for $n = 0, 1, \ldots, N - 1$ from Eqns. (6.22) and (6.23).
7. Finally determine the windowed impulse response of a bandpass filter for $n = 0, 1, \ldots, N\text{-}1$ as:

$$h(n) = w(n).d(n - M) = w(n).\frac{\sin(w_b(n - M)) - \sin(w_a(n - M))}{\pi(n - M)} \qquad (7.1a)$$

and

$$h(M) = \frac{\omega_b - \omega_a}{\pi} \qquad (7.1b)$$

since $w(M) = 1$.

For a notch filter:

$$h(n) = w(n).d(n-m) \tag{7.2a}$$

Where

$$d(n-M) = \frac{\sin(\omega_2(n-M)) - \sin(\omega_2(n-M))}{\pi(n-M)} +$$

$$\frac{\sin(\omega_2(n-M)) - \sin(\omega_2(n-M))}{\pi(n-M)} \tag{7.2b}$$

and

$$h(M) = w(n).\frac{(\omega_2 - \omega_1) + (\omega_4 - \omega_3)}{\pi} = \frac{(\omega_2 - \omega_1) + (\omega_4 - \omega_3)}{\pi} \tag{7.3}$$

Bandpass and notch filters of different order are designed and tested to evaluate the performance of the shortwave radio data communication. This section illustrates filter coefficient determination of bandpass and notch filters by taking a set of specifications.

a. Bandpass Filter type

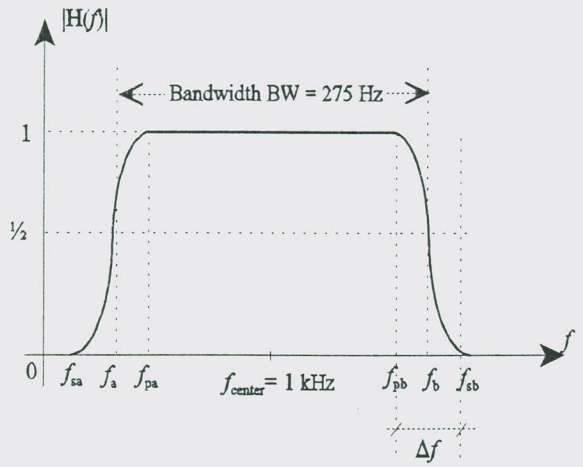


Fig. 7-1 Designed filter specification

1 kHz is chosen as the center frequency of the filter with a sampling frequency of 10 kHz and a bandwidth (BW) of 275 Hz. The transition width Δf is set to 50 Hz and cutoff frequencies are at

$$f_a = f_{center} - \frac{BW}{2} = 862.5Hz$$
$$f_b = f_{center} + \frac{BW}{2} = 1137.5Hz \quad (7.4)$$

From Eqn.(6.16), the passband frequencies f_{pa} and f_{pb} are found to be 887.5 Hz and 1112.5 Hz, respectively. Similarly, the stopband frequencies f_{sa} and f_{sb} are 837.5 Hz and 1162.5 Hz, respectively.

Assuming an attenuation level of 60dB in the stopband and 0.1dB in the passband regions, the corresponding ripple factors δ_{pass} and δ_{stop} will have a value of 0.00576 and 0.001, respectively, using Eqn.(6.18). According to Eqn.(6.21), the ripple factor δ attains the minimum of δ_{pass} and δ_{stop}.

The Kaiser window parameters $\{\alpha, N\}$ are computable in terms of the ripple δ and transition width Δf from Eqn.(6.24) and (6.25), and are found to be 5.65326 and 727. It is expected that for such roll-off (small transition width), the FIR filter have high filter order than the corresponding IIR filter.

The filter coefficients are then determined from Eqn.(7.1) by first generating the Kaiser window function from Eqn.(6.22).

b. Notch Filter

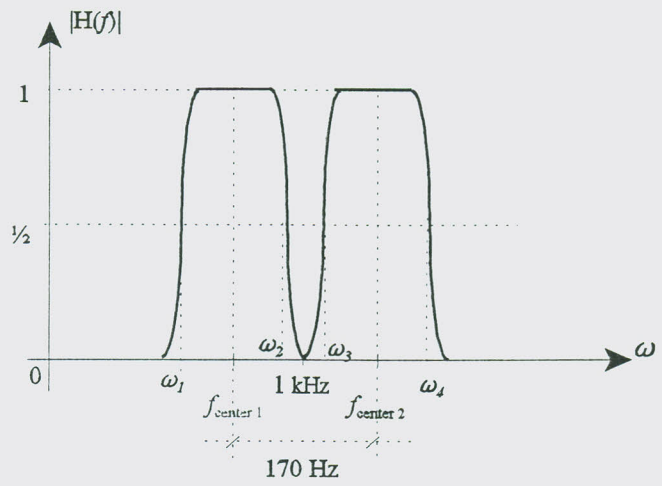


Fig. 7-3 Notch filter magnitude response

This notch filter will have identical order and the Kaiser window function $w(n)$ for the same set of specification as in (**a**) above. The center frequencies f_{center1} and f_{center2} of the notch (the mark and space frequencies) are 170Hz apart, as in the RTTY and AMTOR signals. The four cutoff frequencies, ω_1, ω_2, ω_3 and ω_4 are obtained from Eqn.(6.16). Eventually, Eqn.(7.2) is used to generate the filter coefficients.

7.2 Filter Realization

It is briefly mentioned in the 4th section of chapter 5 that the input/output equation of an FIR filter is given by

$$y(n) = h_0x(n) + h_1x(n - 1) + \ldots + h_nx(n - (N-1)) = \sum_{k=0}^{N-1} h(k)x(n - k) \quad (7.5)$$

with $\mathbf{h} = [h_0, h_1, \ldots, h_M]$ being the impulse responses of the filter. In Z transform the above expression can be written as

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{N-1} h(k)z^{-k} \quad (7.6)$$

The two Eqns.(7.5) and (7.6) may also be represented by a network structure shown in Fig. 7-3.

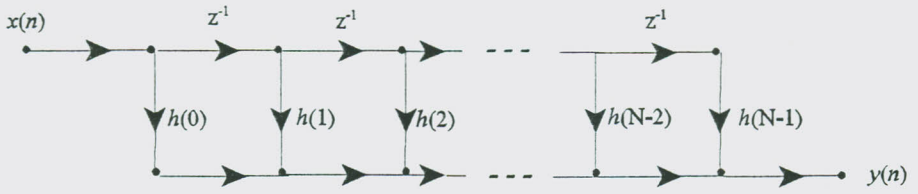


Fig. 7-3 Direct form FIR filter realization

The representation of a desired transfer function into filter network structure is called *realization*. The network shown in Fig. 7-3 is called *direct form realization* since the filter coefficients can be identified directly from the difference equation shown in Eqn.(7.5). The branches labeled z^{-1} corresponds to the delays in Eqn. (7.5) or the multiplier z^{-1} in Eqn.(7.6). This form of realization can be implemented in an efficient manner on the TMS320C50 digital signal processor.

7.3 Implementation on TMS320C50 DSP [4, 8, 9]]

The DSP starter kit (DSK) helps to experiment and use the DSP for real time signal processing. It has an assembler and debugger to develop, test and refine DSK assembly language programs.

The DSK assembler is software interface incorporating the most significant features of an assembler. It differs from other assemblers in that it does not go through a linker phase to create an output file. Instead, the DSK uses special directives to assemble a code at an absolute address during the assembly phase. The debugger is a window-oriented interface capable of loading and executing a code with single-step, breakpoint and run-time halt capabilities.

The realization shown in the preceding section is implemented on the TMS320C50 using this assembly language program. The source program consists of source statements that can contain assembler directives, assembly language instructions and comments. The following diagram shows the DSK software development flow.

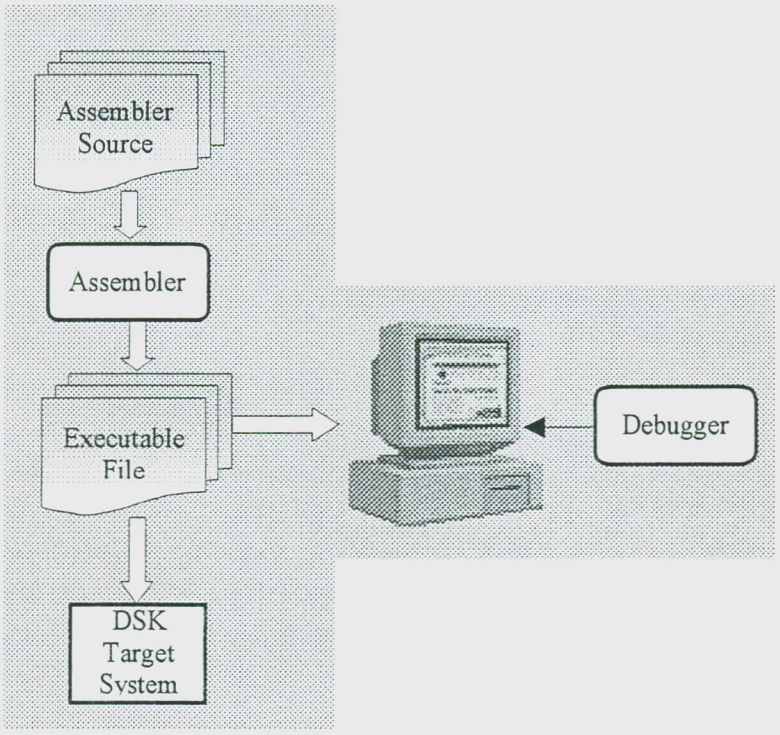


Fig. 7-4 DSK software development flow

In many DSP applications, real time processing of signals is very critical. Important choice must be made in selecting the DSP device capable of real time filtering within the maximum allowable time interval for real time operation. In order to perform the required signal processing task in that interval, it is essential to reduce execution time. This can be accomplished by a single cycle multiply/accumulate instructions. The TMS320C50 is a processor with such capability. Its single-cycle multiply/accumulate with data move instruction and larger on-chip RAM make it possible to implement each filter tap in approximately 50*ns*.

The TMS320C50 provides a total of 1056 word 16-bit word of on-chip dual access RAM (DARAM) configured in three blocks: *block* 0 (B0) is 512 words at address 0100h - 02FFh in local data memory, or 0FE00h - 0FFFFh in program space; *block* 1 (B1) is 512 words at address 0300h - 04FFh in local data memory; and *block* 2 (B2) is 32 words at address 060h in local data memory, as shown in Fig. 3-3. B0 can be programmed as either data or program memory while B1 and B2 are always data memory. The instruction MACD (*multiply and accumulate with data move*) works only on-chip RAM with B0 configured as program memory. This makes MACD to have high performance since no wait state is required for slower external memories. This in turn helps to speed up the filter execution time and thus the instruction is useful for applications such as convolution.

The input/output expression given in Eqn.(7.5) is implemented on the processor as

```
RPT     Nminus1
MACD    (pma), (dma)
```

where *pma* is program memory address and *dma* is data memory address. The RPT Nminus1 instruction loads an immediate 8-bit value N - 1 into the repeat counter. This causes the next instruction to be executed N times (N is the order of the filter). The instruction MACD (*pma*), (*dma*) performs the following functions:

1. Loads the program counter with *pma*,
2. Multiplies the value in data memory location *dma* (on-chip, B1) by the value in the program memory location *pma* (on-chip, B0).
3. Adds the previous product to the accumulator.
4. Copies the data memory value (B1) to the next higher on-chip RAM location. The data move is the mechanism by which the z^{-1} delay can be implemented, and
5. Increments the program counter with each multiply/accumulate to point to the next sample of the unit-sample response.

The diagram in Fig. 7-5 shows a data storage scheme that provides the correct sequence of inputs for the next pass through the filter.

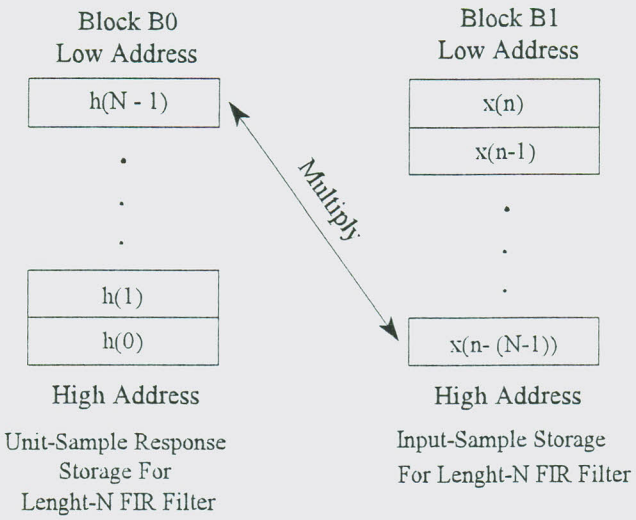


Fig. 7-5 Data Storage Scheme

The assembly source program shown in the *Appendix* C part uses this basic storage scheme for convolving the unit sample response of the length-N FIR filter with the input sample.

The first part of the assembly source program declares the memory-mapped registers and data block addresses as

```
.MMREGS
.ds             0F00h
:
```

This section is also used to declare constants used in the program. The main part of the program comes after initializing of the TMS320C50 port and TLC320C40 (analog interface circuit that performs serial A/D and D/A conversions).

The DSK debugger is used to load this assembly source program on the DSP and execute the code. A spectrum analyzer is used to estimate the frequency response of the filter while running the program code.

7.4 Result

Different types of audio frequency filters are designed and tested by varying the sampling frequency f_s, transition width Δf, and attenuation level while keeping the center frequency at 1 kHz and a bandwidth of 275 Hz. The first part of this section shows the theoretically expected frequency responses of the designed filters. *Matlab* functions are used to plot these expected outputs. The actual frequency responses of the filters are then estimated from the spectrum analyzer by driving the filter with a sinusoidal sweep signal. The figures shown in part II are photographs of these estimates taken from the screen of the spectrum analyzer. The last part of this section depicts the performance of the *amateur radio data communication system* with and without the audio-filter unit for CW, RTTY and AMTOR signals.

Part I Expected Frequency Response

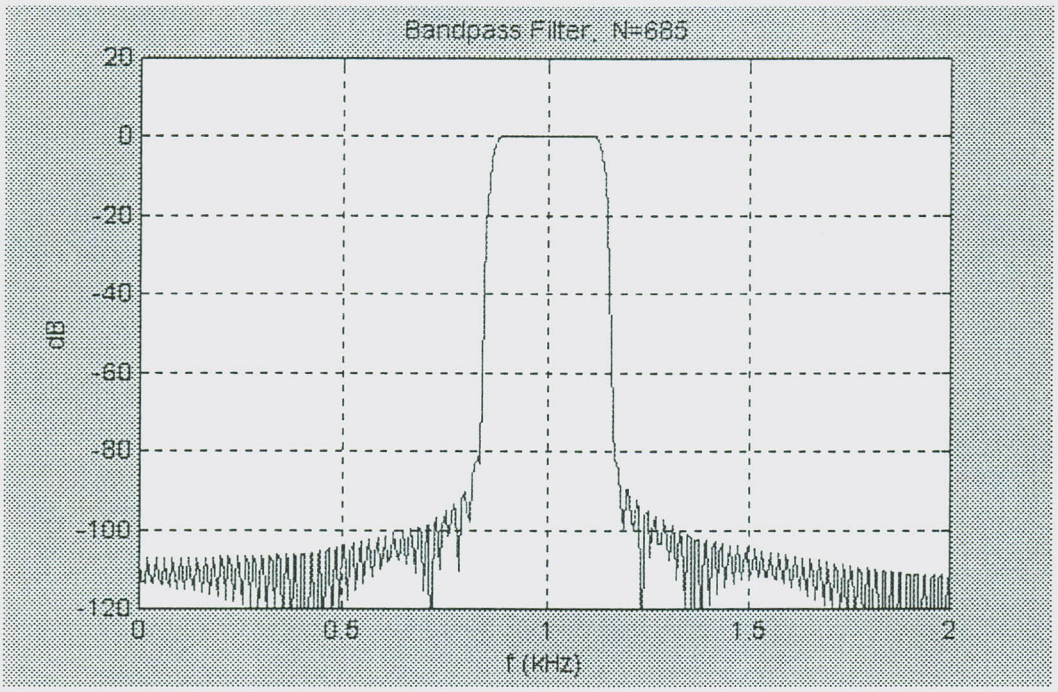


Fig. 7-6 Bandpass filter of order N = 685 with 80dB attenuation level, sampling frequency of 6.8 kHz, and Δf of 50 Hz.

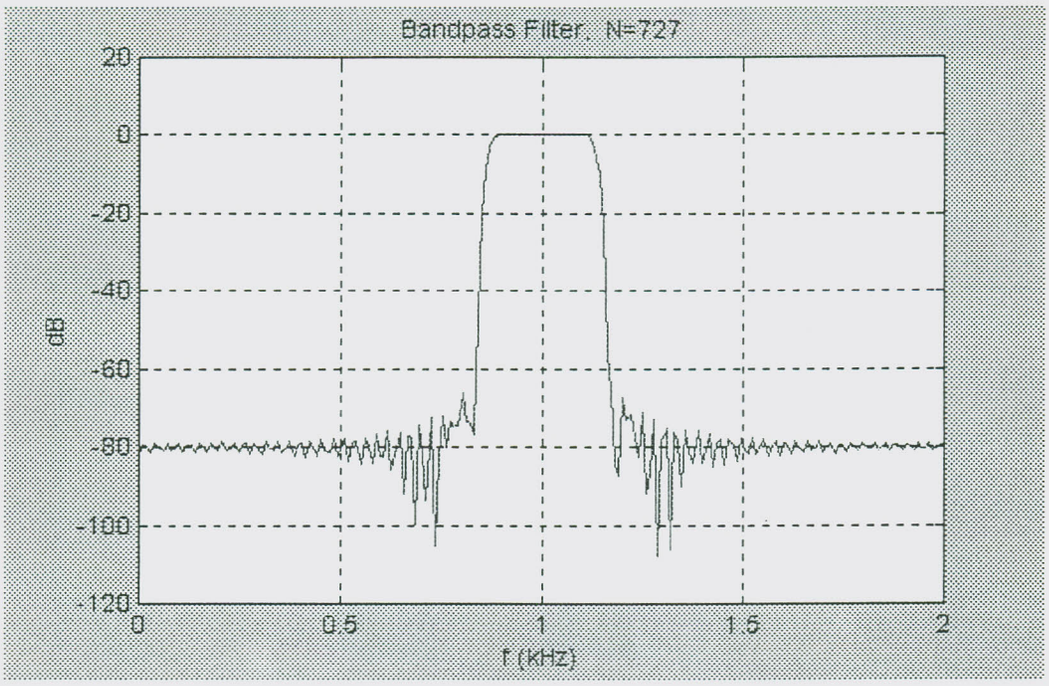


Fig. 7-7 Bandpass filter of order N = 727 with 60dB attenuation level, sampling frequency of 10 kHz, and Δf of 50 Hz.

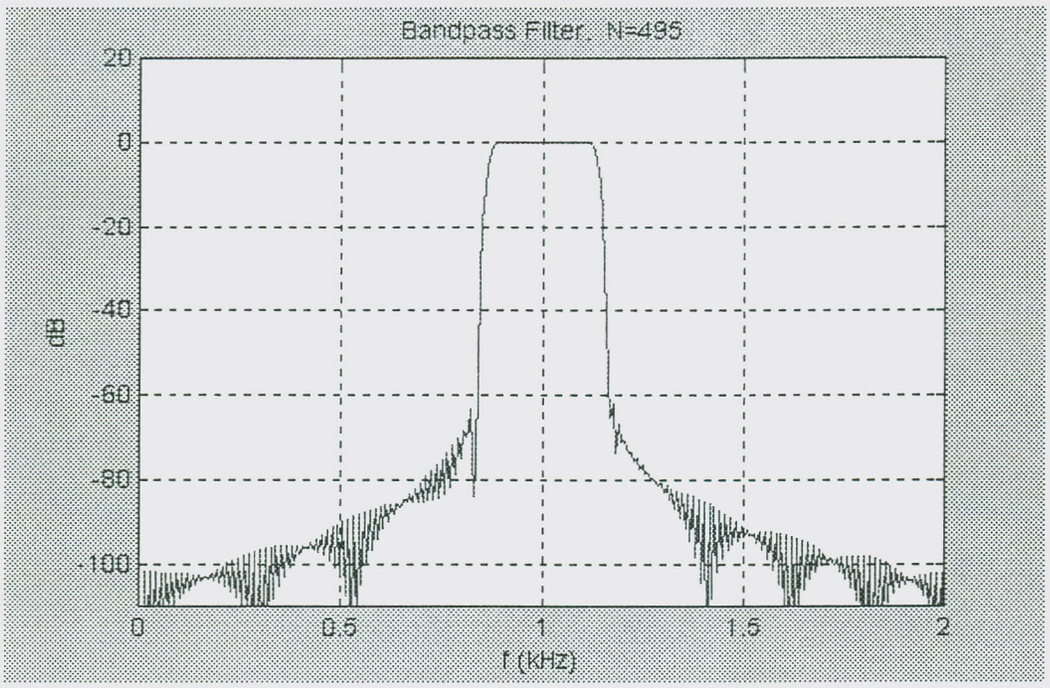


Fig. 7-8 Bandpass filter of order N = 495 with 60dB attenuation level, sampling frequency of 6.8 kHz, and Δf of 50 Hz.

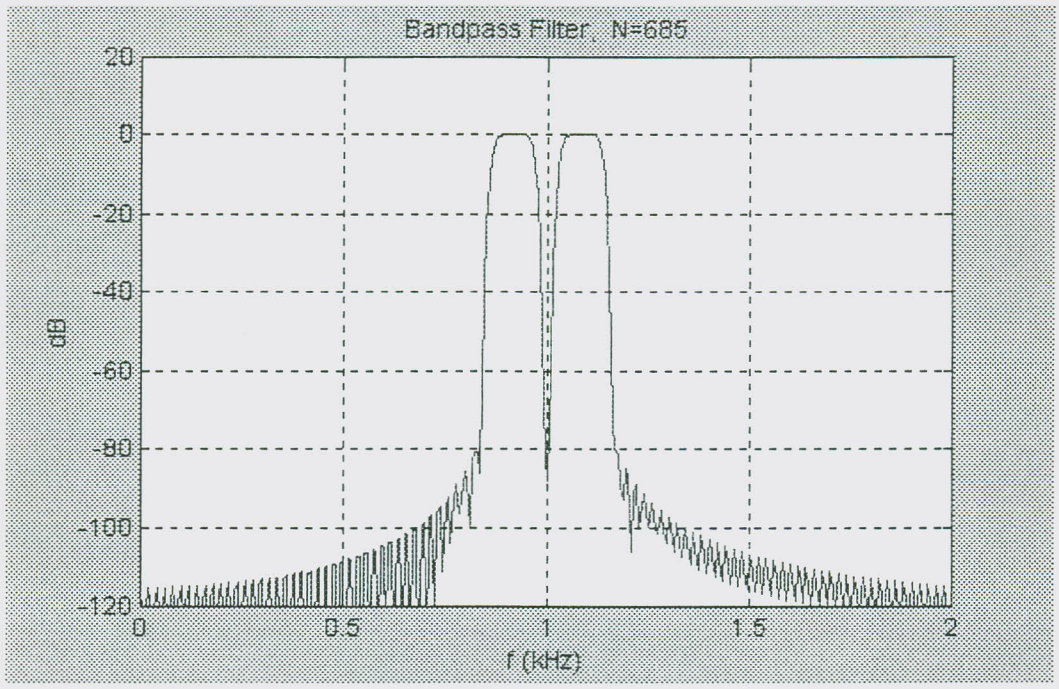


Fig. 7-9 Notch filter of order N = 685 with 80dB attenuation level, sampling frequency of 6.8kHz, and Δf of 50 Hz.

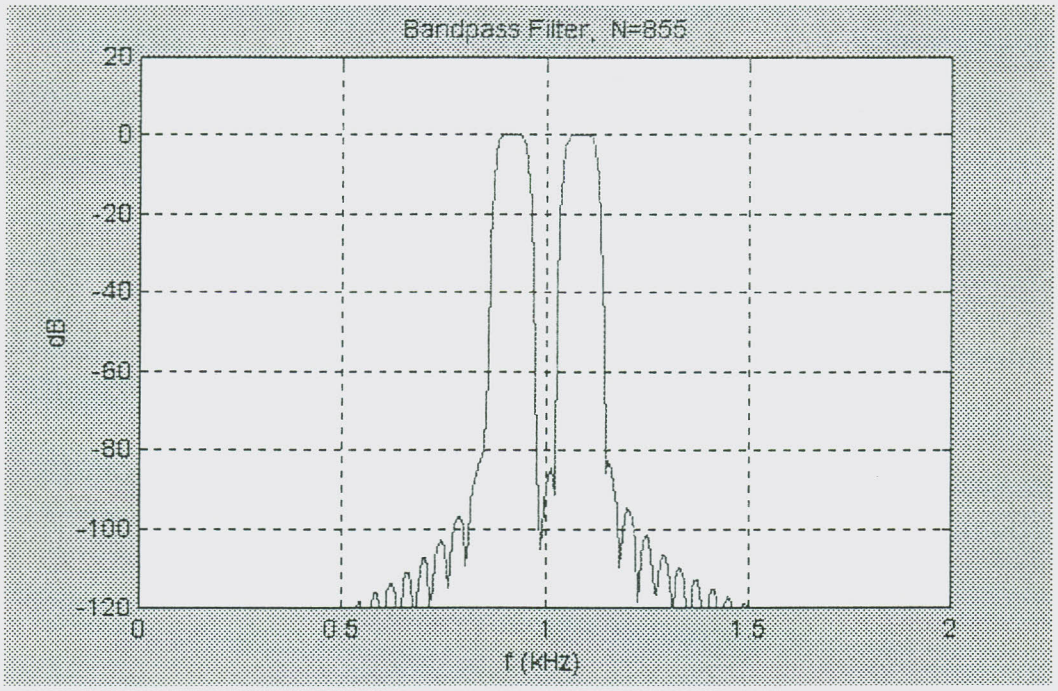


Fig. 7-10 Notch filter of order N = 855 with 80dB attenuation level, sampling frequency of 6.8kHz, bandwidth of 250 Hz and Δf of 40 Hz.

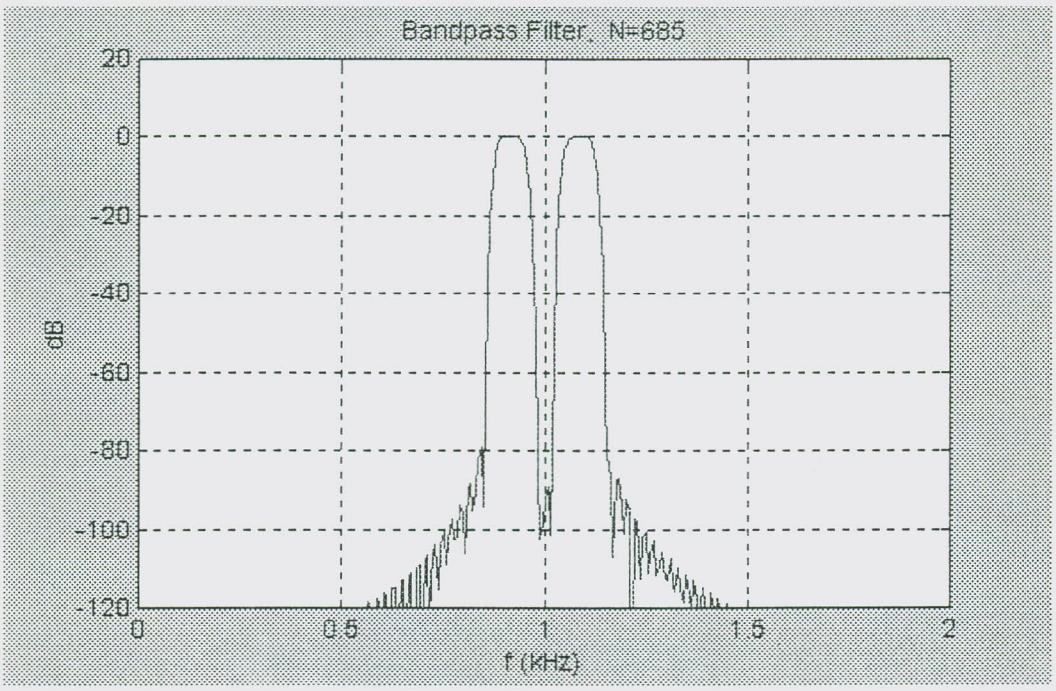


Fig. 7-11 Notch filter of order N = 685 with 80dB attenuation level, sampling frequency of 6.8kHz, Δf of 50 Hz and bandwidth of 250 Hz.

Part II Estimated Frequency Response

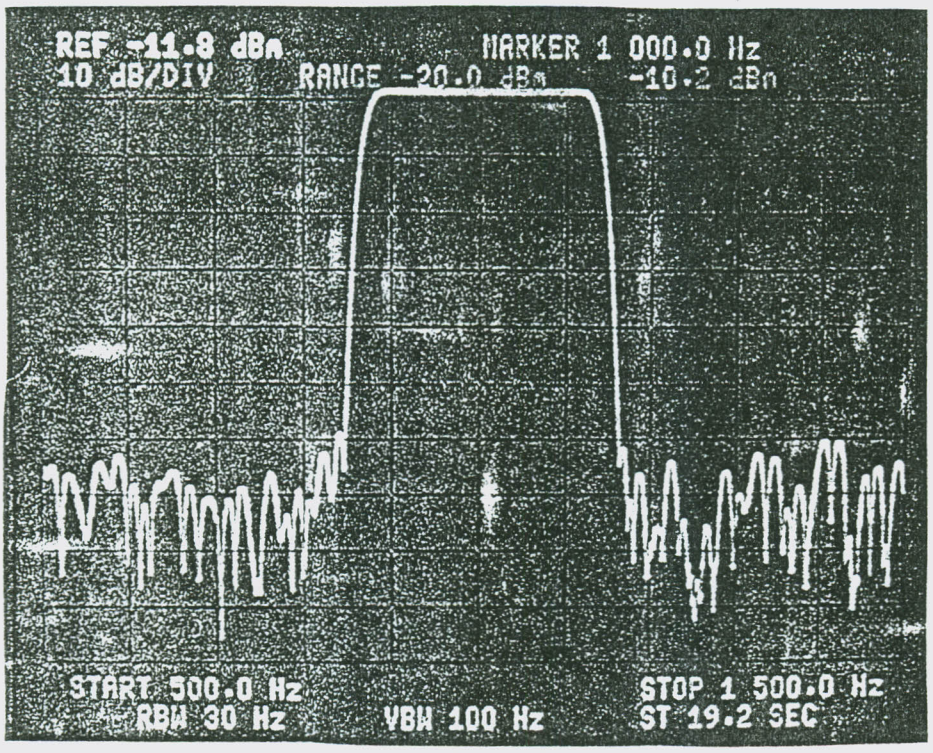


Fig. 7-12 Bandpass filter of order N = 685 with -80dB attenuation level, sampling frequency of 6.8 kHz, and Δf of 50 Hz.

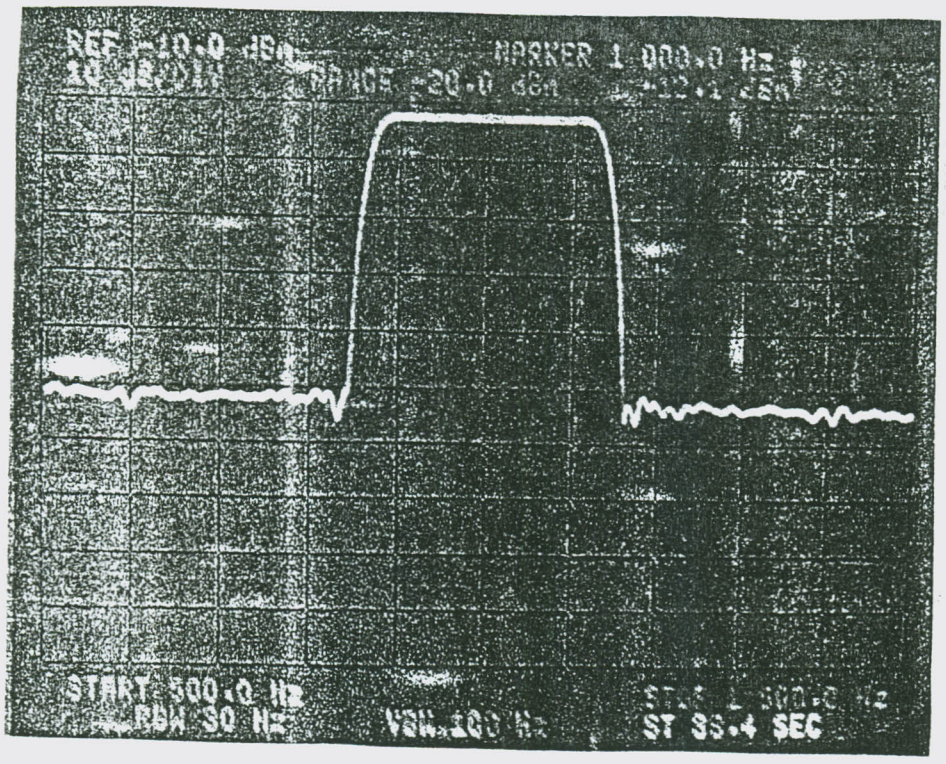


Fig. 7-13 Bandpass filter of order N = 495 with -60dB attenuation level, sampling frequency of 6.8 kHz, and Δf of 50 Hz.

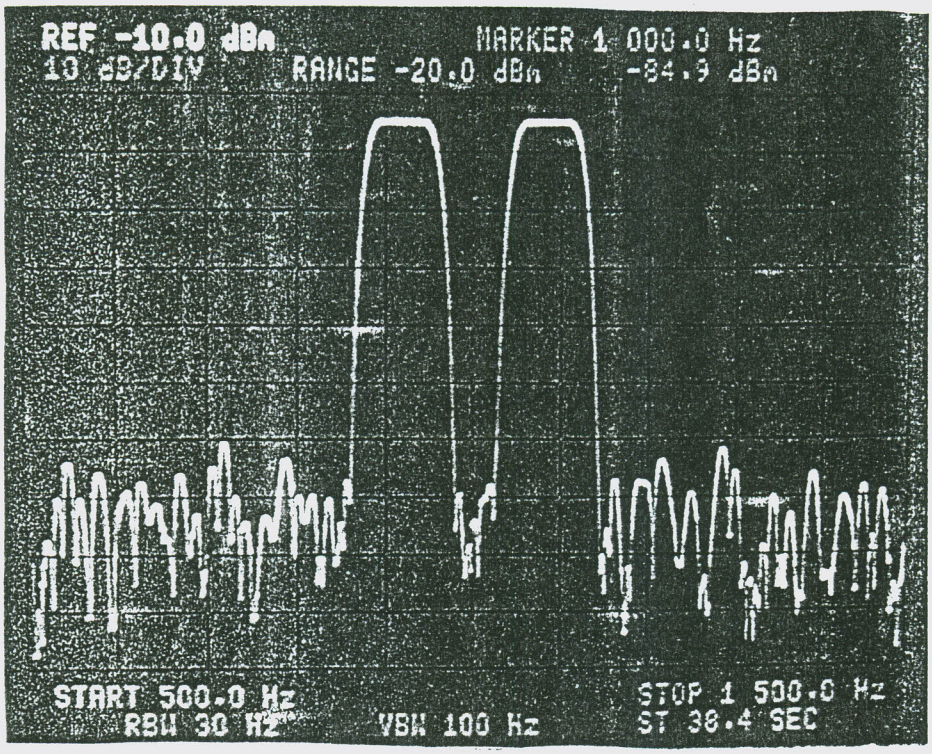


Fig. 7-14 Notch filter of order N = 685 with 80dB attenuation level, sampling frequency of 6.8kHz, and Δf of 50 Hz.

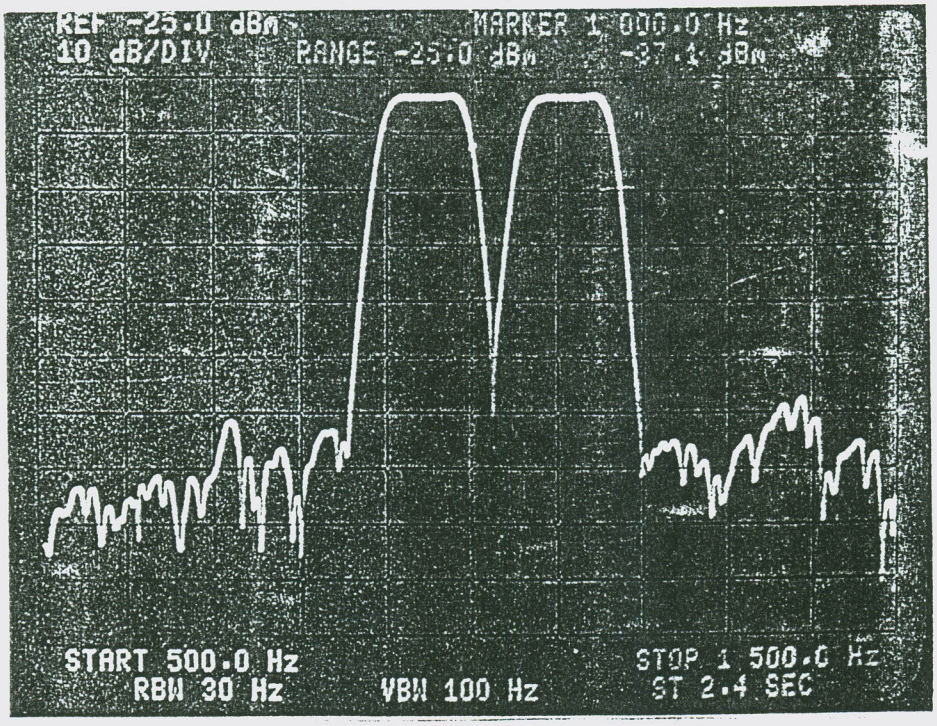


Fig. 7-15 Notch filter of order N = 685 with 80dB attenuation level, sampling frequency of 6.8kHz, and Δf of 50 Hz.

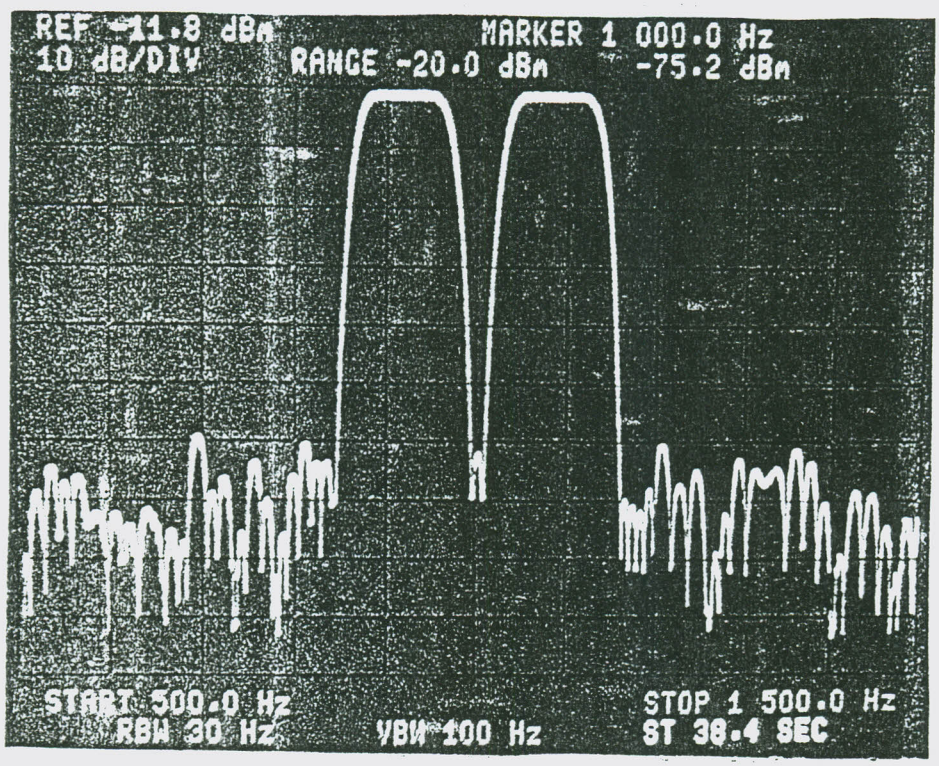


Fig. 7-16 Notch filter of order N = 685 with 80dB attenuation level, sampling frequency of 6.8kHz, Δf of 50 Hz and bandwidth of 250 Hz.

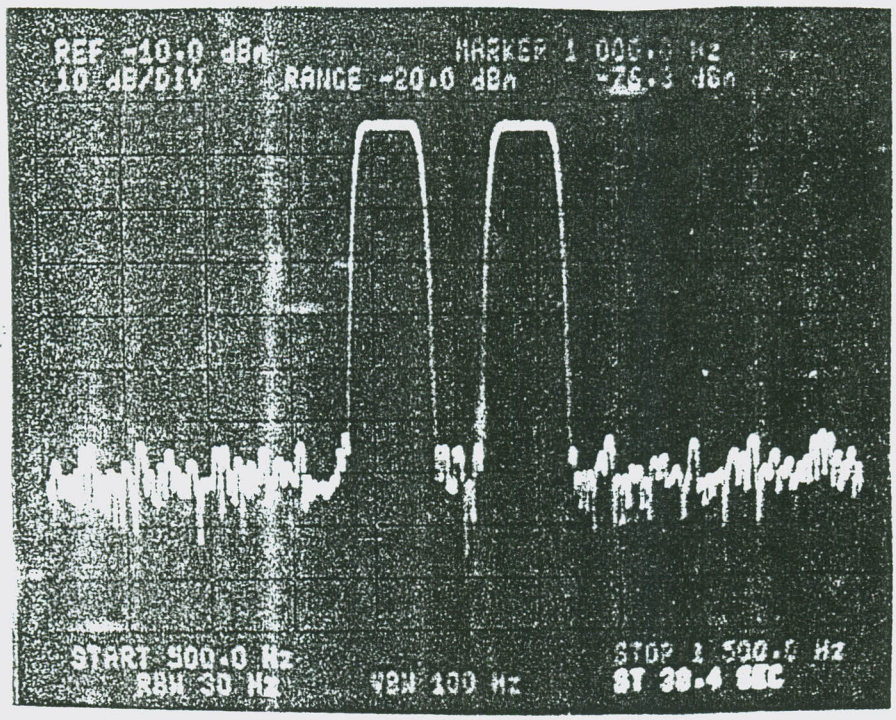


Fig. 7-17 Notch filter of order N = 855 with 80dB attenuation level, sampling frequency of 6.8kHz, bandwidth of 250 Hz and Δf of 40 Hz.

Part III CW, RTTY and AMTOR Signals with and without Filter

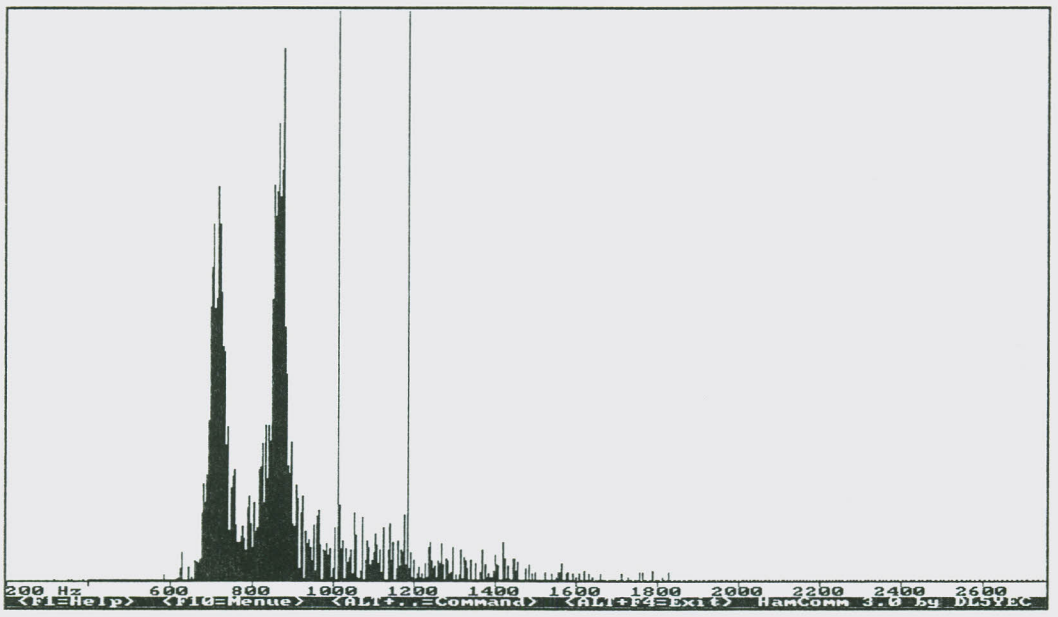


Fig. 7-18 RTTY signal with the mark and space frequencies shifted from the selected center frequency.

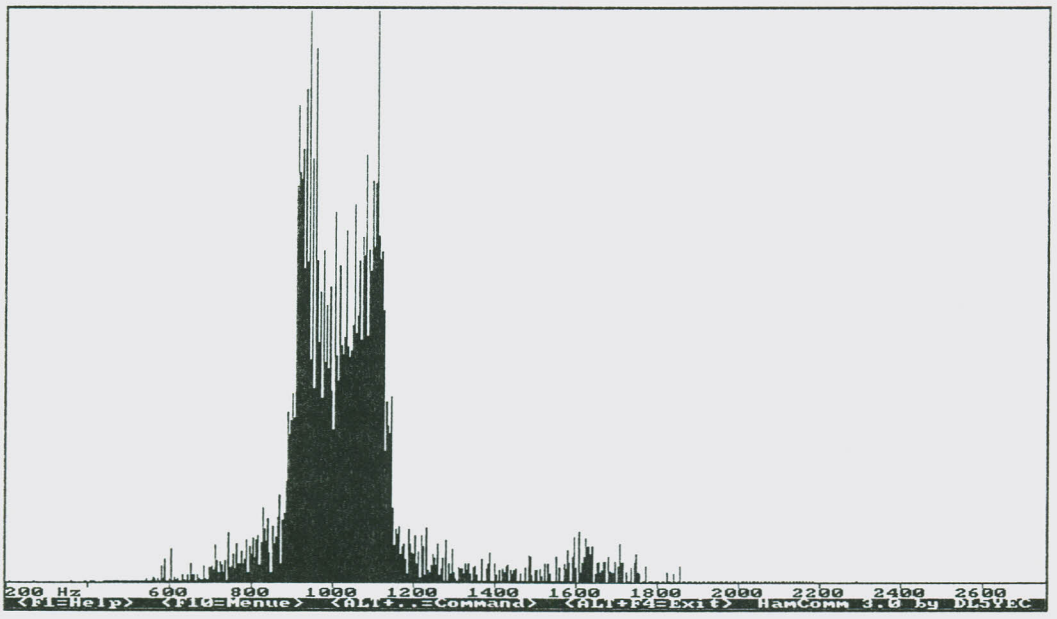


Fig. 7-19 AMTOR signal corrupted with noise

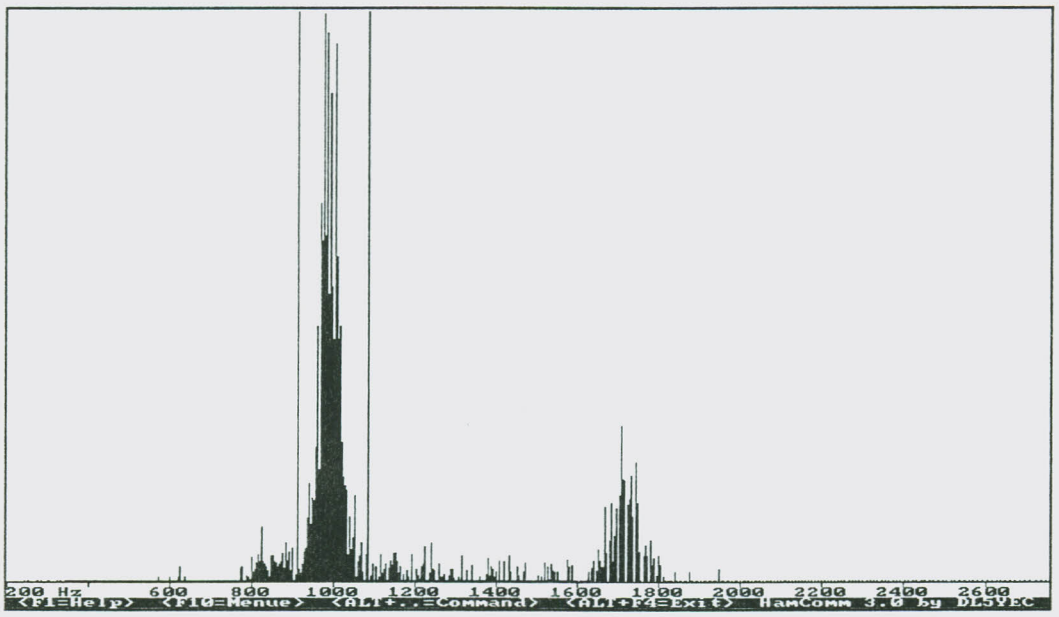


Fig. 7-20 CW signal corrupted with noise

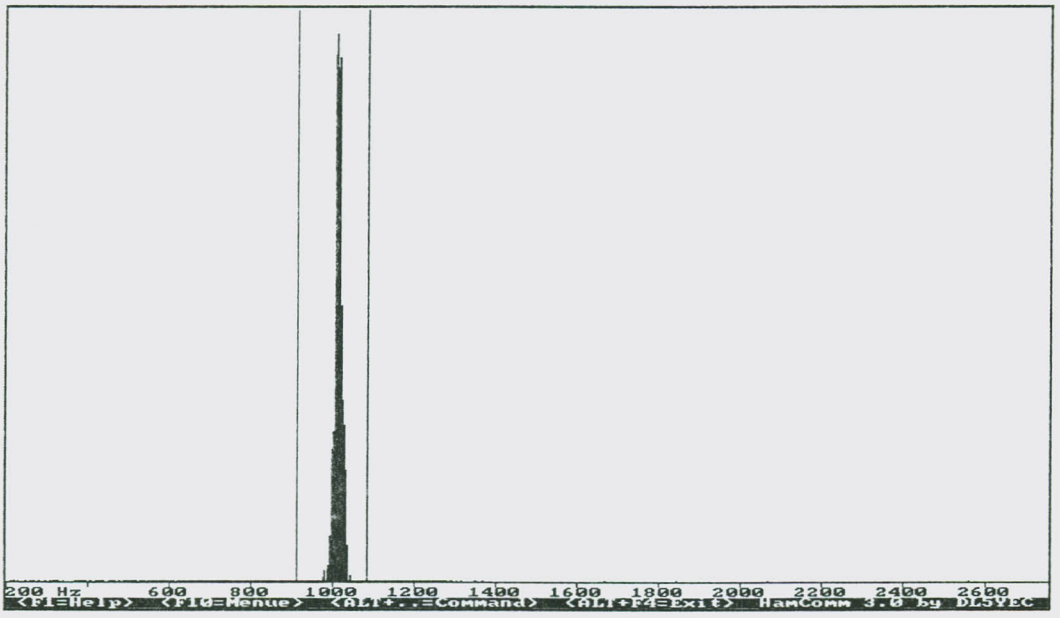


Fig. 7-21 CW signal after the bandpass filter designed

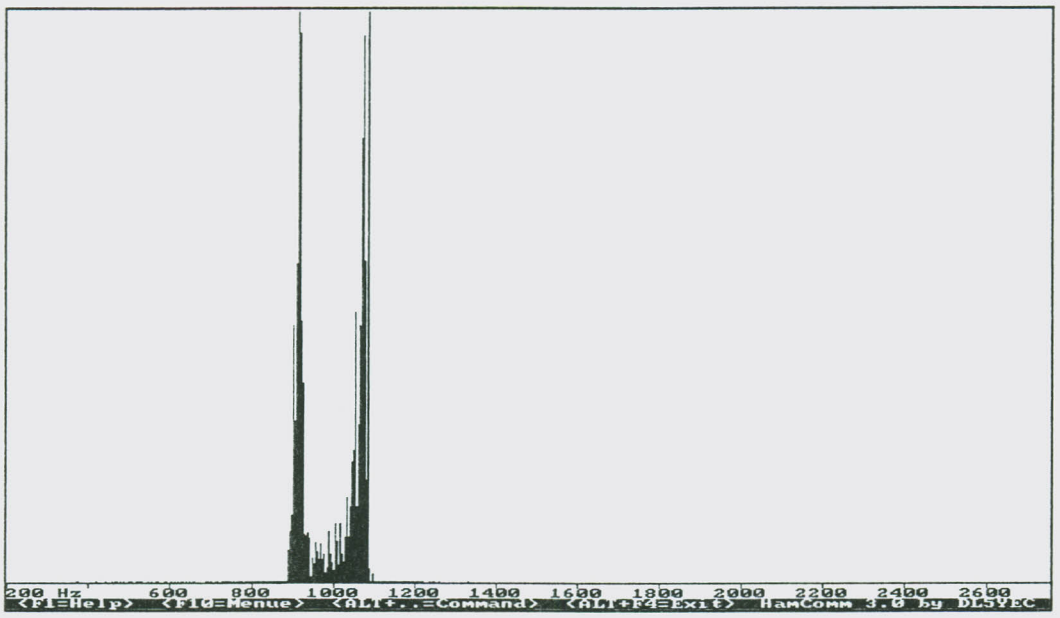


Fig. 7-22 RTTY/AMTOR signal after the bandpass filter designed

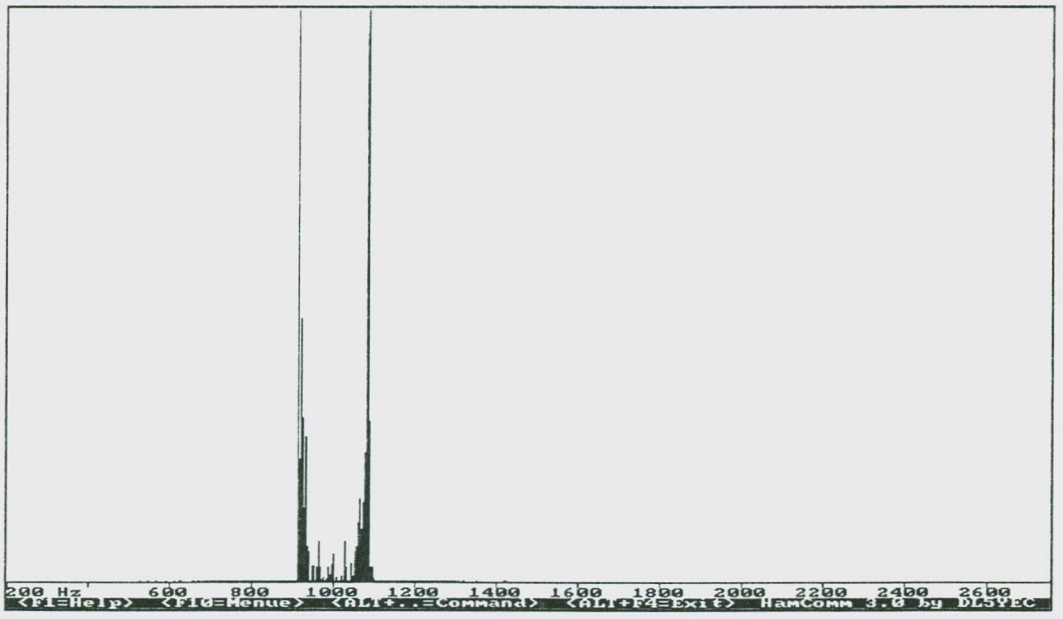


Fig. 7-23 RTTY/AMTOR signal after the bandpass filter designed

Fig. 7-24 RTTY/AMTOR signal after the notch filter designed

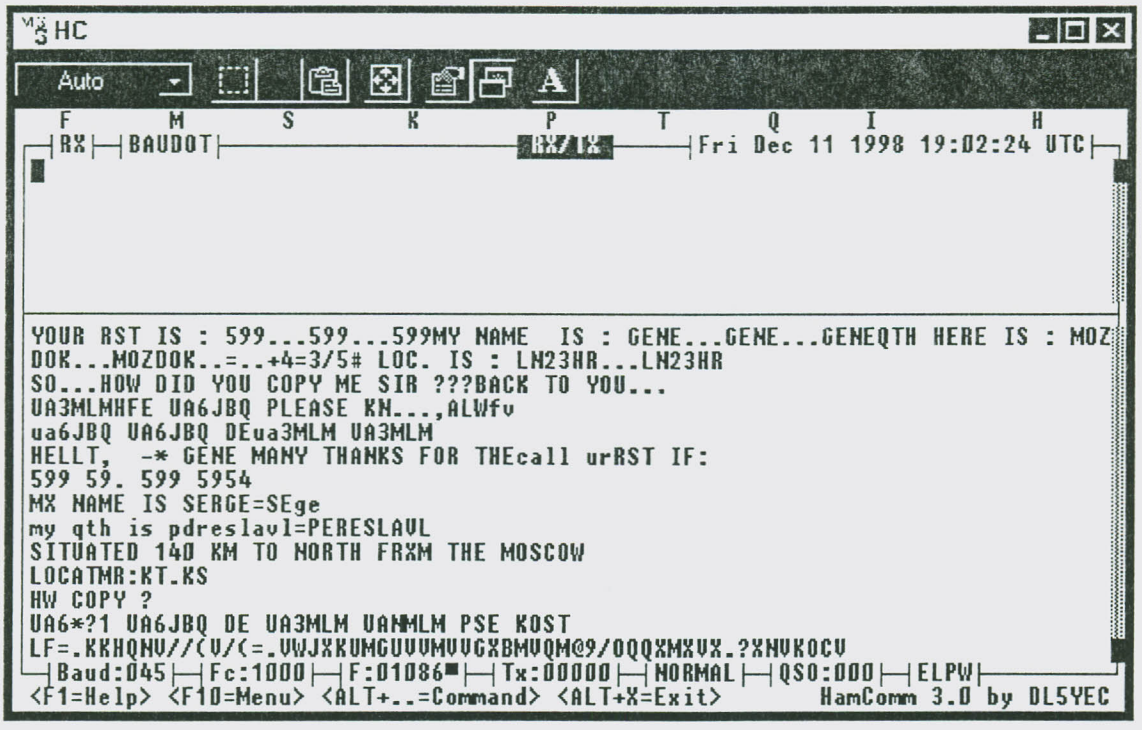


Fig. 7-25 Received text

7.5 Performance Considerations [4, 6]

An important factor of performance consideration of a system is its *stability*. Since the unit-sample response of the designed filter unit is of finite length, it is inherently stable, i.e., bounded input always produces bounded output. This is generally true for all FIR filters while the stability of IIR filters is dependent on the lactation of the poles of the filter.

The other performance consideration is on the precision of the digital signal processor. The filter unit is implemented on the assumption that the processor is of infinite precision. However, there is no processor with such precision and hence the filter coefficients are subjected to an approximation. This approximation introduces coefficient quantization error. The net result due to such imprecise coefficient representation is a deviation of the resulting frequency response from the designed one.

In the implementation of this audio-frequency filter unit on the TMS320C50, there is also quantization error due to finite wordlength effect in the hardware. The input signals are subjected to A/D quantization noise while the output signals are subjected to D/A quantization noise. This noise will introduce an error in representing the instantaneous values of the incoming signal.

In fixed-point arithmetic, multiplication of data samples with filter coefficients produce results that are longer than the available accuracy permits. For example, the product of two 16-bit numbers is 32-bit. If no rounding or truncation of intermediate results is performed, the required accuracy increases with time. Thus, it is necessary to quantize the results of multiplications to the available accuracy, causing errors that can be seen at the outputs of the filter as small fluctuation of the signal around the ideal response, which would otherwise be obtained with unrestricted arithmetic accuracy.

For the above mentioned reasons, the actual filter response slightly deviates from the ideal one, as can be seen from part I and II of section 7-4. The following table summarizes this discrepancy.

Table 7-1 Summary of Filter Response

Filter Type	Specification	Obtained Value	Theoretical Value
Bandpass Filter 1	Attenuation Level	72dB	80dB
	Bandwidth	267	275
	Passband Frequency	890, 1127	887.5, 1112.5
	Stopband Frequency	843, 1170	837.5, 1162.5
Bandpass Filter 2	Attenuation Level	54dB	60dB
	Bandwidth	265	275
	Passband Frequency (Hz)	887, 1115	887.5, 1112.5
	Stopband Frequency (Hz)	838, 1166	837.5, 1162.5
Notch Filter 1	Attenuation Level	72dB	80dB
	Bandwidth	266, 96 each	275, 105 each
	Passband Frequency (Hz)	875, 957, 1050, 1120	887.5, 992.5, 1057.5, 1112.5
	Stopband Frequency (Hz)	835, 1002, 1114, 1169	837.5, 942.5, 1007.5, 1162.5
Notch Filter 2	Attenuation Level	74.5dB	80dB
	Bandwidth	242, 72 each	250, 80 each
	Passband Frequency (Hz)	890, 940, 1062, 1114	895, 935, 1065, 1105
	Stopband Frequency (Hz)	847, 980, 1024, 1150	855, 975, 1025, 1145

7-6 Conclusion/Recommendation

As shown in part three of the result section, CW, RTTY and AMTOR signals contain unwanted frequency components other than the two tone frequencies. These frequency components appear at the decoder due to inadequate bandpass filtering in the transceiver and/or there is no other unit that can reject them after the receiver. The HamComm considers each frequency component as part of the useful signal and begins counting the zero crossings to generate interrupts for calculating the corresponding tone frequency. As a result, the received signal is interpreted as characters which do not convey any message. An example of such decoding is shown in Fig. 7-25.

A remarkable difference is observed when the audio-filter unit is inserted in the communication system. The audio-filter totally rejects all frequency components outside the mark and space. The performance of the filter can be evaluated in two cases.

Case 1 CW Signals

In CW transmission, a single tone frequency is needed anywhere between the mark and space frequencies to correctly receive the transmitted data. Thus, the bandwidth of bandpass filter should slightly be larger than 170 Hz, distance between mark and space. In this bandwidth, it is possible that the useful frequency be mixed with other components, as shown in Fig.7-21. Although the filter totally rejects all frequencies outside the mark and space, it is not narrow enough to pick that single tone frequency. Therefore, the filter cannot be a reliable solution to obtain a message which is nearly identical to the transmitted data.

Case II RTTY/AMTOR Signals

For RTTY and AMTOR signals, the information is carried in the two tone frequencies, mark and space. The need for the filter is not only to reject the noise component outside the mark and space but also to pick the two tone frequencies. When the bandpass filter is inserted in the system, all frequency components outside the mark and space are removed and remain only with the tone frequencies and some other frequency components between them. The result of decoding is, of course, improved but still have some problem because of the noise components exit between the mark and space. There comes the need for the notch filter which can selectively pass the two frequencies. As shown in Fig 7-24, this filter effectively takes out the useful component. Indeed, it shows a great improvement in the system performance.

Since the CW and RTTY modes are not accompanied by any error correction mechanism, having a filter unit may not guarantee receiving an exact replica of the transmitted information. Even in an AMTOR mode, which has FEC and ARQ type error correction mechanism, a 100% system performance should not be expected. The filter will only remove the unwanted

frequency component. There must be a reliable error detection and correction mechanism for best system performance.

During communication, the center frequency of the transmitted signal can only be guessed. It is possible to receive a signal with shifted center frequency as shown in Fig.7-18. Passing such signals through the audio-filter unit, which already has fixed center frequency, will end up in receiving nothing. If the filter has a variable center frequency, such problems may be reduced, if not totally removed.

While testing the filter unit in the communication system, only one type of filter is used at a time. It was not possible to switch between the different filter types since the processor needs to be reloaded every time the code is changed. Thus, the unit should be designed in such a way that it is possible to switch between different types of filters while changing the mode of communication. Also, the filter should be able to run whenever the switch is turned on by loading the source code on an EPROM.

The performance of the communication system can also be evaluated in terms of speed. The speed of receiving a CW signal is 45 WPM and it is not more than 100 bauds for RTTY and AMTOR. In today's technology, where the speed of communication is in terms of Gbps, using CW, RTTY or AMTOR type communication mode with such speed will remain as "curiosity" on the air unless something is done to improve the speed .

Appendix A

A.1 The TMS320C5X Internal hardware Summary

Unit	Symbol	Function
Accumulator	ACC(32) ACCH(16) ACCL(16)	A 32-bit accumulator accessible in two halves: ACCH (accumulator high) and ACCL (accumulator low). Used to store the output of the ALU. See subsection 3.5.2 for more information.
Accumulator Buffer	ACCB (32)	A register used to temporarily store the 32-bit contents of the accumulator. This register has a direct path back to the ALU and therefore can be arithmetically or logically acted upon with the ACC. See subsection 3.5.2 for more information.
Arithmetic logic Unit	ALU	A32- bit 2s-complement arithmetic logic unit having two 32-bit input ports and one 32-bit output port feeding the accumulator. See subsection 3.5.2 for more information.
Auxiliary Register Arithmetic unit	ARAU	An unsigned 16-bit arithmetic unit used to calculate indirect addresses using the auxiliary, index, and compare registers as inputs. See subsection 3.4.3 for more information.
Auxiliary Register Compare	ARCR(16)	A register used as a limit to compare indirect address against. See subsection 3.4.3 for more information.
Auxiliary Register File	AUXREGS	A register file containing eight 16-bit auxiliary registers (AR0-AR7) used for indirect data address pointers, temporary storage, or integer arithmetic processing through the ARAU. See subsection 3.4.3 for more information.
Auxiliary Register Pointer	ARB(3)	A3-bit register used as a pointer to the currently selected auxiliary register. These bits are stored in ST0. See subsection 3.4.3 for more information.
Block move address register	BMAR(16)	A16-bit register that holds an address value for use with block moves or multiply/accumulates. See subsection 3.4.2 for more information.
Block repeat Active Flag	BRAF (1)	A 1-bit flag indicating that a block repeat in currently active. This bit is normally set when the RPTB instruction is executed and cleared when the BRCT register decrements below zero. This bit resides in the PMST register. See subsection 3.6.5 for more details.
Block Repeat Counter Register	BRCR(16)	A16-bit memory-mapped counter register used to limit the number of times the block is to be repeated, see subsection 3.6.5 for more details.
Carry	C	This signal indicates that a data access is mapped to global memory space as defined by the GREG register. See section 6.4 for more details.
Configure RAM	CNF	This bit indicates whether on-chip dual-access RAM blocks are mapped to program or data space. The CNF bit resides in ST1. See subsection 3.6.3 for more information.
Data Bus	DATA	A16-bit bus used to route data.
Data Memory	DATA MEMORY	This block refers to data memory used with the core and defined in specific device descriptions. It refers to both on-and-off-chip memory blocks in data memory space.
Data Memory Address Bus	DATA ADDRESS	A16-bit bus that carries the address for data memory access.
Data Memory Address Immediate Register	Dam(7)	A7-bit register containing the immediate relative address with in a 128-word data page. See subsection 3.4.2 for more information.
Data Memory Page Pointer	DP(9)	A9-bit register containing the address of the current page. Data pages are 128 words each, resulting in 512 pages of addressable data memory space (some locations are reserved). See subsection 3.4.2 for more information.
Dynamic Bit Pointer	TREG2(4)	A4-bit register that hold a dynamic pointer for the BITT instruction. See section 4.3 for more information.
Dynamic Shift Count	TREG1(5)	A5-bit register that holds a dynamic prescaling shift count for data inputs to the ALU. See section 4.3 for more information.
External Flag	XF(1)	This bit drives the level of the external flag pin and resides in ST1. See subsection 3.6.3 for more information.
Global Memory Allocation Register	GREG(8)	An 8-bit memory-mapped register for specifying the size of the global memory space. See section 6.4 for more details.
Interrupt Flag Register	IFR(16)	A 16-bit flag register used to latch the active-low interrupts. The IFR is a memory -mapped register. See section 3.8 for more information.

Interrupt Pointer	IPTR(5)	Five bits pointing to the 2k page where the interrupt vectors currently reside in the system. These bits reside in the PMST register. See section 3.8 for more information.
Interrupt Mask Register	IMR(16)	A16-bit memory-mapped register used to mask interrupt. See section 3.8 for more information.
Microprocessor/ Microcomputer Mode	MP/MC	This bit resides in the PMST register and indicates whether the on-chip ROM is mapped into program address space. See subsection 3.6.3 for more information.
Over flow Flag	OV(1)	This bit resides in SSST0 and indicates an overflow in an arithmetic operation in the ALU. See subsection 3.6.3 for more information.
Parallel Logic Unit	PLU	A16-bit logic unit that executes logic operations from either long immediate operands or the contents of the DBMR directly upon data locations with out interfering with the contents of the CALU registers. See section 3.7 for more information.
Prefetch Counter	PFC (15-0)	A16-bit counter used to prefetch program instruction. The PFC contains the address of the instruction currently being prefeched. It is updated when a new prefetch is initiated. The PFC can also address program memory when the blocks move (BLPD). Multiply-accumulate (MAG/MACD),and table read/writ (TALR/TBLW) instructions are used and can address data memory when the block move (BLDD) instruction is used.
Prescaler Count register	COUNT(4)	a four-bit register that contains the value for the prescaling operation. when the register contents are used as prescaling data, this register is loaded from the dynamic shift count or from the instruction. in conjunction with the BIT and BITT instructions, this register is loaded from the dynamic bit pointer or the instruction word.
Product Register	PREG(32)	A 32-bit product register used to hold the multiplier's product. The high and low words of the PREG can be accessed individually. See subsection 3.5.3 for more information.
Program Bus	PROG DATA	A 16-bit bus sued to route instructions (and data for the MAC and MACO in striations).
Program Memory	PROGRAM MEMORY	This block refers to program memory used with the core and defined in specific device descriptions. It refers to both on-and off-chip memory blocks accessed in program memory space.
Program Memory address Bus	PROG ADDRESS	A 16 -bit bus that carries the program memory address.
Prescaling shifter	PRESCALER	A - to 16- bit left barrel shifter used to prescale data coming into the ALU. Also used to align data for multiprecision operations. This shifter is also used as a-0-to 16- bit right barrel shigter of the ACC. See subsection 3.5.2 for more information.
Postscaling Shifter	OPOSTSCALER	A 0-to 7-bit left berrel shifter used to postscale data coming out of the CALU. See a 0-to 16-bit right barrel shifter of the ACC. See subsection 3.5.2 for more information.
Product Shifter	P-SCALER	A0-, 1-, or 4- bit left shifter that can remove extra sign bits (gained in the multiply operation) when fixed-pint arithmetic is used; or a 6-bit right shifter that ca scale the products down to avoid overflow in the accumulation process. See subsection 3.5.3 for more information.
Product Shifter Mode	PM(2)	These two bits define the product shifter mode; They reside in ST1. See subsection 3.6.3 for more information.
Repeat Counter	RPTC 16)	A16-bit counter used to controls the repeated execution of a single instruction. See subsection 3.6.4 for more information.
Sign Extension Mode	SXM(1)	This bit resides in ST1 and controls whether the arithmetic operation will be sign-extended or not. See subsection 3.6.3 for more information.
Stack	STACK	An 8x16-bit hardware stack used to store the PC during interrupts and calls. The ACCL and data memory values may also be pushed onto and popped from he stack. See section 3.8 for more information.
Status Registers	ST0, ST1, PMST	Three 16-bit status registers that contain status and control bits. See subsection 3.6.3 for more information.
Temporary Multiplicand	TREG0(16)	A16-bit register that temporarily holds an operand for the multiplier. See subsection 3.5.3 for more information.
Temporary Registers Enable	TRM(1)	This bit defines whether an LT(A,D,P,S)instruction loads all three of the TREGs(0,1,2) to maintain compatibility with the 'C25 or loads just TREG0. This bit resides in the PMST register. See subsection 3.6.3 for more information.
Test/Control Flag	TC(1)	This bit resides in ST1 and stores the results of ALU or PLU test bit operations. See subsection 3.6.3 for more information.

A.2 Instruction Set Summary

Accumulator Memory Reference Instructions		
Mnemonic	**Description**	**Words**
ADCB	Add ACCB to ACC wit carry	1
ADD	Add to ACC	½
AND	AND with ACC	½
ANDB	AND ACCB with ACC	1
LACB	Load ACC with ACC	1
LACC	Load ACC With ACCB	½
LAMM	Load ACC with contents of memory-mapped register	1
NGE	Negate accumulator	1
NORM	Normalize contents of ACC	1
OR	OR with accumulator	½
SACB	Store ACC in ACCB	1
SACH	Store high ACC with shift	1
SACL	Store low ACC with shift	1
SAMM	Store ACC to memory-mapped register	1
SFL	Shift ACC LEFT	1
SFL	Shift ACC right	1
XOR	Exclusive - OR with ACC	½
ZAP	Zero ACC and PREG	1
Auxiliary Registers ad Data Page Pointer Instructions		
Mnemonic	**Description**	**Words**
CMPR	Compare ARn with ARCR	1
LAAR	Load ARn	½
LDP	Load data page pointer	1
LDP	Load data page pointer	1
MAR	Modify ARn	1
SAR	Store AR n	1
APL	AND DBMR or constant with data memory value	½
CPL	Compare DBMR or content with data memory value	½
OPL	OR DBMR or constant with data memory value	½
SPLK	Store long immediate to data memory location	2
T Register, P Register, and Multiply Instructions		
Mnemonic	**Description**	**Word**
APAC	Add PREG to ACC	1
LTD	Load TREG0, accumulate previous product, and move data	1
MAC	Multiply and accumulate	2
MACD	Multiply and accumulate with data move	2
MADD	Multiply and accumulate with source pointed at by BMAR	1
MPY	Multiply	½
MADS	Multiply and accumulate bot with source pointed at by BMAR and with data move	1
Branch Instruction		
BAND[D]	Branch conditionally	2
CALD[D]	Call subroutine indirect	1
CALL[D]	Call subroutine	2
CC[D]	Call conditionally	2
INTR	Soft interrupt	1
RETC[D]	Return conditionally	1
RETE	Return with context switch & global interrupt	1
TRAP	Software interrupt	1
XC	Execute next instruction(s) conditionally	1
I/O and Data Memory Operations		
BLDD	Block move from data memory to data memory	½
BLDP	Block move from data memory to program memory	1
BLPD	Block move from program memory to data	½
DMOV	Data move in data memory	1

IN	Input data from port	2
LMMR	Load memory-mapped register	2
OUT	Output data to port	2
SMMR	Store memory-mapped register	2
TBLR	Table read	1
TBLW	Table write	1
Control instructions		
Mnemonic	**Description**	**Words**
BIT	Test bit	1
BITT	Test Bit specified by TREG2	1
CLRC	Clear control bit	1
ILRC	Idle until interrupt	1
IDLE2	Idle until interrupt- low power mode	1
LST	Load status register	1
NOP	No operation	1
POP	Pop top of stack t data memory	1
POPD	Pop top of stack to data memory	1
PSHD	Push low ACC onto stack	1
PUSH	Push low ACC onto stack `	1
RPT	Repeat next instruction	½
RPTB	Repeat block	2
SETC	Repeat next instruction and clear ACC and PREG	2
SETC	Set control bit	1
SST	Store status register	1

Appendix B

B.1 Fixed-Point Representation

The operations in digital signal processors basically operate on numbers stored in binary form. Binary representation of numbers consists of binary digits (bits), which are associated with weights of appropriate powers of 2.

In a b-bit fixed-point representation of a positive number, a prescribed number of bits, say, b_1 is associated with the integer part, the remaining bits represent the fractional part. For example, the representation $1001_{\Delta}01$ is equal to 9.25 in the decimal number system. The right side of the binary point $_{\Delta}$ constitutes the fractional part while the left part constitutes integer part.

A fixed-point number can also have a negative sign. There are three standard conventions for representing such signed numbers. These are the *sign-magnitude convention, one's complement convention* and the *two's complement convention.*

A b-bit fixed-point fraction is shown in Fig. A-1. Such a number has a sign bit denoted by s, followed by the b binary bits $\alpha_1, \alpha_{2,\ldots,} \alpha_b$. Regardless of the convention adopted, $s = 0$ for a positive number and for a negative number $s = 1$. For a positive number the magnitude is always represented by $\sum_{k=1}^{b} \alpha_k 2^{-k}$. The three conventions differ only in the way in which the bits α_k represents the magnitude of a negative number.

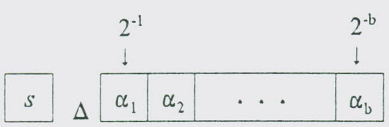


Fig. B-1 A b-bit fixed point fraction with a sign bit.

The TMS320C50 is a 16-bit fixed-point processor which uses two's complement convention for its representation of signed bits. In two's complement convention, the value of the number is always given by $-s + \sum_{k=1}^{b} \alpha_k 2^{-k}$. For such a representation, the magnitude of a negative number is obtained by complementing each bit (including s) and adding 2^{-b}. For example, let $b = 4$ and let $1_{\Delta}1001$ be the 2's complement representation of $-x$. Then its magnitude is represented by $0_{\Delta}0111$, whose decimal equivalent is 7/16. This type of representation is also called Q15 format, in the context of TMS320C50.

Appendix C

C.1 Source Code for Bandpass filter Implementation on TMS20C50 DSP

```
;     This program implements bandpass filter of order N = 685,
      attenuation level of 80 dB, transition frequency Δf of 50Hz,
      sampling frequency of 6.8 kHz, center frequency of 1 kHz and
      bandwidth of 275 Hz. The filter coefficients are obtained by
      Kaiser window design method. The program shown below is not
      complete and is only for one type of filter design, the complete
      program is in the companion diskette.
;


00001 ---- ----  ;
00002 ---- ----  ;
00003 ---- ----    .MMREGS
00004 ---- 0f00    .ds      0f00h
00005 0f00 0015  TA       .word    21          ;
00006 0f01 0015  RA       .word   21             ; These AIC registers
                                                 ; values give
00007 ---- ----                                  ; a sampling freq of
                                                 ; 6.8 kHz
00008 ---- ----                      ;
00009 0f02 0023  TB       .word    35          ;
00010 0f03 0023  RB       .word    35          ;
00011 0f04 0028  AIC_CTR  .word    28h
00012 0f05 0000  OUTPUT   .word    0
00016 ---- ----  ;        .space   03000h
00017 ---- ----  ;        .data    03000h
00018 ---- ----    .include "K_DATA.asm"        ;A file containing
                                                 ;filter coeffiecients
*****************************************
*     OPENING INCLUDE FILE K_DATA.asm
*****************************************
00001 0f09 0000  kwd:    .q15     7.68914E-07
00002 0f0a 0000   .q15       -1.88002E-06
00003 0f0b 0000   .q15       -4.19416E-06
00004 0f0c 0000   .q15       -3.15741E-06
00005 0f0d 0000   .q15       1.43676E-06
00006 0f0e 0000   .q15       6.36258E-06
00007 0f0f 0000   .q15       6.78113E-06
00008 0f10 0000   .q15       1.08845E-06
00009 0f11 0000   .q15       -6.68103E-06
00010 0f12 0000   .q15       -9.97226E-06
00011 0f13 0000   .q15       -4.98286E-06
00012 0f14 0000   .q15       4.50465E-06
00013 0f15 0000   .q15       1.07138E-05
00014 0f16 0000   .q15       8.24848E-06
00015 0f17 0000   .q15       -6.14626E-07
00016 0f18 0000   .q15       -8.06042E-06
00017 0f19 0000   .q15       -8.16192E-06
00018 0f1a 0000   .q15       -2.22757E-06
00019 0f1b 0000   .q15       3.28686E-06
00020 0f1c 0000   .q15       3.77495E-06
```

```
00021 0f1d 0000   .q15       9.61211E-07
00022 0f1e 0000   .q15       2.43886E-07
00023 0f1f 0000   .q15       3.4926E-06
00024 0f20 0000   .q15       5.8846E-06
00025 0f21 0000   .q15       1.24603E-06
00026 0f22 0000   .q15       -9.51651E-06
00027 0f23 0000   .q15       -1.64876E-05
00028 0f24 0000   .q15       -9.52081E-06
00029 0f25 0000   .q15       9.75059E-06
00030 0f26 0000   .q15       2.58982E-05
00031 0f27 0000   .q15       2.24815E-05
00032 0f28 0000   .q15       -1.96347E-06
00033 0f29 0000   .q15       -2.86951E-05
00034 0f2a ffff   .q15       -3.42064E-05
00035 0f2b 0000   .q15       -1.14502E-05
00036 0f2c 0000   .q15       2.2119E-05
00037 0f2d 0001   .q15       3.83641E-05
00038 0f2e 0000   .q15       2.35149E-05
00039 0f2f 0000   .q15       -8.77482E-06
00040 0f30 ffff   .q15       -3.11126E-05
00041 0f31 0000   .q15       -2.64586E-05
00042 0f32 0000   .q15       -3.26134E-06
00043 0f33 0000   .q15       1.55524E-05
00044 0f34 0000   .q15       1.58252E-05
00045 0f35 0000   .q15       4.55998E-06
00046 0f36 0000   .q15       -1.06144E-06
00047 0f37 0000   .q15       5.05383E-06
00048 0f38 0000   .q15       1.01654E-05
00049 0f39 0000   .q15       -1.37411E-06
00050 0f3a 0000   .q15       -2.57045E-05
00051 0f3b ffff   .q15       -3.75347E-05
00052 0f3c 0000   .q15       -1.46736E-05
00053 0f3d 0001   .q15       3.31248E-05
00054 0f3e 0002   .q15       6.55539E-05
00055 0f3f 0001   .q15       4.57519E-05
00056 0f40 0000   .q15       -1.96421E-05
00057 0f41 fffe   .q15       -7.91497E-05
00058 0f42 fffe   .q15       -7.83407E-05
00059 0f43 0000   .q15       -1.13806E-05
00060 0f44 0002   .q15       6.89503E-05
00061 0f45 0003   .q15       9.50229E-05
00062 0f46 0001   .q15       4.47421E-05
00063 0f47 ffff   .q15       -3.85124E-05
00064 0f48 fffe   .q15       -8.47041E-05
00065 0f49 ffff   .q15       -6.03984E-05
00066 0f4a 0000   .q15       4.54702E-06
00067 0f4b 0001   .q15       5.02646E-05
00068 0f4c 0001   .q15       4.48943E-05
00069 0f4d 0000   .q15       1.0074E-05
00070 0f4e 0000   .q15       -1.04242E-05
00071 0f4f 0000   .q15       -1.039E-06
00072 0f50 0000   .q15       1.05825E-05
00073 0f51 0000   .q15       -8.84927E-06
00074 0f52 ffff   .q15       -5.08341E-05
00075 0f53 fffe   .q15       -6.45585E-05
00076 0f54 0000   .q15       -1.12239E-05
00077 0f55 0002   .q15       8.14154E-05
00078 0f56 0004   .q15       0.000129492
00079 0f57 0002   .q15       6.94838E-05
00080 0f58 fffe   .q15       -6.87838E-05
```

```
00081 0f59 fffb   .q15       -0.00017231
00082 0f5a fffc   .q15       -0.000141612
00083 0f5b 0000   .q15       1.28792E-05
00084 0f5c 0005   .q15       0.000167851
00085 0f5d 0006   .q15       0.000190797
00086 0f5e 0001   .q15       6.01021E-05
00087 0f5f fffd   .q15       -0.000114216
00088 0f60 fffa   .q15       -0.000187429
00089 0f61 fffd   .q15       -0.000109037
00090 0f62 0001   .q15       3.95391E-05
00091 0f63 0004   .q15       0.000128179
00092 0f64 0003   .q15       9.97002E-05
00093 0f65 0000   .q15       1.0432E-05
00094 0f66 ffff   .q15       -4.26538E-05
00095 0f67 0000   .q15       -2.64249E-05
00096 0f68 0000   .q15       3.60863E-06
00097 0f69 0000   .q15       -1.85681E-05
00098 0f6a fffe   .q15       -7.94124E-05
00099 0f6b fffe   .q15       -8.99215E-05
00100 0f6c 0000   .q15       1.0601E-05
00101 0f6d 0005   .q15       0.000162125
00102 0f6e 0007   .q15       0.000215663
00103 0f6f 0002   .q15       7.79506E-05
00104 0f70 fffb   .q15       -0.000170815
00105 0f71 fff6   .q15       -0.000319792
00106 0f72 fffa   .q15       -0.000213119
00107 0f73 0002   .q15       9.03373E-05
00108 0f74 000b   .q15       0.000344718
00109 0f75 000a   .q15       0.00032796
00110 0f76 0001   .q15       4.40901E-05
00111 0f77 fff8   .q15       -0.000270919
00112 0f78 fff5   .q15       -0.000357564
00113 0f79 fffb   .q15       -0.000159846
00114 0f7a 0004   .q15       0.000134364
00115 0f7b 0009   .q15       0.00027733
00116 0f7c 0006   .q15       0.000183987
00117 0f7d 0000   .q15       -1.37666E-05
00118 0f7e fffc   .q15       -0.000123609
00119 0f7f fffe   .q15       -8.61115E-05
00120 0f80 0000   .q15       -9.30969E-06
00121 0f81 0000   .q15       -1.86045E-05
00122 0f82 fffd   .q15       -9.75249E-05
00123 0f83 fffd   .q15       -0.0001021
00124 0f84 0001   .q15       5.84612E-05
00125 0f85 0009   .q15       0.000275609
00126 0f86 000a   .q15       0.000312276
00127 0f87 0001   .q15       4.8297E-05
00128 0f88 fff5   .q15       -0.000347383
00129 0f89 ffef   .q15       -0.00052341
00130 0f8a fff8   .q15       -0.000267033
00131 0f8b 0008   .q15       0.000260109
00132 0f8c 0014   .q15       0.000623778
00133 0f8d 0010   .q15       0.000493141
00134 0f8e ffff   .q15       -4.70627E-05
00135 0f8f ffee   .q15       -0.000552105
00136 0f90 ffed   .q15       -0.000604199
00137 0f91 fffb   .q15       -0.000180921
00138 0f92 000b   .q15       0.000338385
00139 0f93 0011   .q15       0.000527589
00140 0f94 0009   .q15       0.000289008
```

```
00141 0f95 fffd   .q15        -0.000100379
00142 0f96 fff7   .q15        -0.00029075
00143 0f97 fffa   .q15        -0.00019339
00144 0f98 0000   .q15        -1.43146E-05
00145 0f99 0000   .q15        1.73783E-05
00146 0f9a fffe   .q15        -8.29404E-05
00147 0f9b fffe   .q15        -9.04373E-05
00148 0f9c 0004   .q15        0.000131481
00149 0f9d 000d   .q15        0.00041019
00150 0f9e 000d   .q15        0.000396948
00151 0f9f ffff   .q15        -4.60399E-05
00152 0fa0 ffec   .q15        -0.000613161
00153 0fa1 ffe7   .q15        -0.000769654
00154 0fa2 fff8   .q15        -0.000262388
00155 0fa3 0012   .q15        0.00056694
00156 0fa4 0021   .q15        0.001018472
00157 0fa5 0015   .q15        0.000652649
00158 0fa6 fff7   .q15        -0.000275639
00159 0fa7 ffe0   .q15        -0.001002568
00160 0fa8 ffe2   .q15        -0.000919649
00161 0fa9 fffd   .q15        -0.000114789
00162 0faa 0017   .q15        0.000716609
00163 0fab 001d   .q15        0.000904246
00164 0fac 000c   .q15        0.000383091
00165 0fad fff6   .q15        -0.000311074
00166 0fae ffed   .q15        -0.000590978

                  .

                  .

                  .

00664 11a0 0000   .q15        2.43886E-07
00665 11a1 0000   .q15        9.61211E-07
00666 11a2 0000   .q15        3.77495E-06
00667 11a3 0000   .q15        3.28686E-06
00668 11a4 0000   .q15        -2.22757E-06
00669 11a5 0000   .q15        -8.16192E-06
00670 11a6 0000   .q15        -8.06042E-06
00671 11a7 0000   .q15        -6.14626E-07
00672 11a8 0000   .q15        8.24848E-06
00673 11a9 0000   .q15        1.07138E-05
00674 11aa 0000   .q15        4.50465E-06
00675 11ab 0000   .q15        -4.98286E-06
00676 11ac 0000   .q15        -9.97226E-06
00677 11ad 0000   .q15        -6.68103E-06
00678 11ae 0000   .q15        1.08845E-06
00679 11af 0000   .q15        6.78113E-06
00680 11b0 0000   .q15        6.36258E-06
00681 11b1 0000   .q15        1.43676E-06
00682 11b2 0000   .q15        -3.15741E-06
00683 11b3 0000   .q15        -4.19416E-06
00684 11b4 0000   .q15        -1.88002E-06
00685 11b5 0000   .q15        7.68914E-07
>>>>> FINISHED READING ALL FILES
```

```
*****************************************
*     CLOSING FILE K_DATA.asm
*****************************************
00019 ---- ----                                ; This part reserves data
                                               ; location for an input
                                               ; (XN) delayed by 685 (z-1 )
00020 ---- ----
00021 ---- ----
00022 ---- ----
00023 11b6 0000  XN  .word
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      11b7 0000
      11b8 0000
      11b9 0000
      11ba 0000
      11bb 0000
      11bc 0000
      11bd 0000
      11be 0000
      11bf 0000
      11c0 0000
      11c1 0000
      11c2 0000
      11c3 0000
      11c4 0000
      11c5 0000
      11c6 0000
      11c7 0000
      11c8 0000
      11c9 0000
      11ca 0000
      11cb 0000
00024 11cc 0000  XN1     .word
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      11cd 0000
      11ce 0000
      11cf 0000
      11d0 0000
      11d1 0000
      11d2 0000
      11d3 0000
      11d4 0000
      11d5 0000
      11d6 0000
      11d7 0000
      11d8 0000
      11d9 0000
      11da 0000
      11db 0000
      11dc 0000
      11dd 0000
      11de 0000
      11df 0000
      11e0 0000
      11e1 0000
00025 11e2 0000  XN2     .word
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      11e3 0000
      11e4 0000
      11e5 0000
```

```
      11e6 0000
      11e7 0000
      11e8 0000
      11e9 0000
      11ea 0000
      11eb 0000
      11ec 0000
      11ed 0000

                  .

                  .

                  .
      1459 0000
      145a 0000
      145b 0000
      145c 0000
      145d 0000
      145e 0000
      145f 0000
00054 1460 0000  XN31    .word   0,0
      1461 0000
00055 1462 0000  XNLAST  .word   0
00056 ---- ----
00057 ---- 080a   .ps      0080ah
00058 080a 7980  rint:    B       RECEIVE
      080b 0000
00059 080c 7980  xint:    B       TRANSMIT
      080d 0000
00060 ---- ----
00061 ---- 0a00   .ps       0a00h
00062 ---- ----   .entry                  ;
>>>>> ENTRY POINT SET TO 0a00
00063 ---- ----
;---------------------------------------------------------------
00064 ---- ----
00065 0a00 be41   SETC    INTM
00066 0a01 bc00   LDP     #0
00067 0a02 5d07   OPL     #0834h,PMST
      0a03 0834
00068 0a04 bf80   LACC    #0
      0a05 0000
00069 0a06 882a   SAMM    CWSR
00070 0a07 8828   SAMM    PDWSR
00071 0a08 be47   SETC    SXM             ;  SXM MUST BE SET
00072 0a09 ae04   SPLK    #022h,IMR       ; This turns on receive
                                          ; interrupt only
      0a0a 0022
00073 ---- ----
00074 0a0b 7a80   CALL    AICINIT
      0a0c 0000
00075 0a0d ae04   SPLK    #12h,IMR
      0a0e 0012
00076 0a0f be42   CLRC    OVM
00077 0a10 bf00   SPM     0
```

```
00078 0a11 be40   CLRC    INTM
00079 ---- ----
00080 ---- ----  WAIT:                    ; a main program will run
                                           ; at this point
00081 0a12 7980   B       WAIT
      0a13 0a12
00082 ---- ----
00083 ---- ----  RECEIVE:
00084 0a14 bc23   LDP     #XN
00085 ---- ----
00086 0a15 be40   CLRC    INTM            ; enable interrupts for
                                           ; debugging, for DSK
                                           ; purposes
00087 ---- ----
00088 ---- ----
00089 ---- ----
00090 0a16 0820   LAMM    DRR             ; load accumulator with
                                           ; word received from AIC
00091 0a17 9036   SACL    XN              ; store the value of
                                           ; received word to a
                                           ; variable
00092 ---- ----
00093 0a18 bf08   LAR     AR0,#XNLAST     ; load AR0 with address of
                                           ; last delay element
      0a19 1462
00094 0a1a be59   ZAP                     ; ZERO ACC and product
                                           ; registers
00095 0a1b 8b88   MAR     *,AR0           ; AR0 is the current AR
                                           ; register.
00096 ---- ----
00097 0a1c bec4   RPT     #684            ; Repeat the next
                                           ; instruction 684 times
                                           ; through the complete
                                           ; filter coefficient data
      0a1d 02ac
00098 0a1e a390   MACD    #kwd,*-     ;
      0a1f 0f09
00099 0a20 be04   APAC                    ; Accumulate last product
00100 ---- ----
00101 0a21 9905   SACH    OUTPUT,1        ; the ACC data is
                                           ; transferred to
                                           ; OUTPUT and gets rid of
                                           ; extra sign bit.
00102 0a22 1105   LACC    OUTPUT,1
00103 0a23 be09   SFL
00104 0a24 bfb0   AND     #0fffch         ; Two LSB's must be zero
                                           ; for AIC
      0a25 fffc
00105 0a26 8821   SAMM    DXR             ; write output word to
                                           ; transmit register
00106 0a27 be3a   RETE                  ;
00107 ---- ----
00108 ---- ----
00109 ---- ----  TRANSMIT:
00110 0a28 be3a   RETE
00111 ---- ----
00112 ---- ----
00113 ---- ----
```

```
*****************************************************************
00114 ---- ----  ; This part initializes the TMS320C50 serial port
00115 ---- ----  ; and the TLC320C40's (AIC) TA,RA,TB,RB and
00116 ---- ----  ; control registers
00117 ---- ----
;*****************************************************************
00118 ---- ----  ;
00119 0a29 ae26  AICINIT: SPLK    #20h,TCR       ; To generate 10 MHz
                                                ; from Tout for AIC
                                                ; master clock
      0a2a 0020
00120 0a2b ae25    SPLK    #01h,PRD
      0a2c 0001
00121 0a2d 8b88    MAR     *,AR0
00122 0a2e bf80    LACC    #0008h              ; Non continuous
                                               ; mode
      0a2f 0008
00123 0a30 9022    SACL    SPC                 ; FSX as input
00124 0a31 bf80    LACC    #00c8h              ; 16 bit words
      0a32 00c8
00125 0a33 9022    SACL    SPC
00126 0a34 bf80    LACC    #080h               ; Pulse AIC reset by
                                               ; setting it low
      0a35 0080
00127 0a36 9821    SACH    DXR
00128 0a37 9005    SACL    GREG
00129 0a38 bf08    LAR     AR0,#0FFFFh
      0a39 ffff
00130 0a3a bec4    RPT     #10000              ; and taking it high
                                               ; after 10000 cycles
      0a3b 2710
00131 0a3c 1088    LACC    *,0,AR0             ; (.5ms at 50ns)
00132 0a3d 9805    SACH    GREG
00133 ---- ----
00134 0a3e bc1e    LDP     #TA
00135 0a3f be47    SETC    SXM
00136 0a40 1900    LACC    TA,9                ; Initialize TA and
                                               ; RA register
00137 0a41 2201    ADD     RA,2
00138 0a42 7a80    CALL    AIC_2ND
      0a43 0000
00139 ---- ----
00140 0a44 bc1e    LDP     #TB
00141 0a45 1902    LACC    TB,9                ; Initialize TB and
                                               ; RB register
00142 0a46 2203    ADD     RB,2            ;
00143 0a47 b802    ADD     #02h            ;
00144 0a48 7a80    CALL    AIC_2ND         ;
      0a49 0000
00145 ---- ----
00146 0a4a bc1e    LDP     #AIC_CTR
00147 0a4b 1204    LACC    AIC_CTR,2           ; Initialize control
                                               ; register
00148 0a4c b803    ADD     #03h            ;
00149 0a4d 7a80    CALL    AIC_2ND         ;
      0a4e 0000
00150 0a4f ef00    RET                     ;
00151 ---- ----
00152 ---- ----  AIC_2ND:
00153 0a50 bc00    LDP     #0              ; Data page point is 0
```

```
                                                    ; (MM regs)
00154 0a51 9821    SACH    DXR                      ; send ACChi 00
00155 0a52 be40    CLRC    INTM                     ; enable interrupts
00156 0a53 be22    IDLE                             ; wait for interrupt
00157 0a54 bf9f    ADD     #6h,15                   ; 0000 0000 0000 0011
XXXX XXXX XXXX XXXX b
      0a55 0006
00158 0a56 9821    SACH    DXR                      ; send ACChi to initiate
secondary protocol
00159 0a57 be22    IDLE                             ; wait for interrupt
00160 0a58 9021    SACL    DXR                      ; send the T register
data
00161 0a59 be22    IDLE                             ; wait for interrupt
00162 0a5a b900    LACL    #0                       ; clear ACClo
00163 0a5b 9021    SACL    DXR                      ; send another to make
sure 1st word got sent
00164 0a5c be22    IDLE                             ; wait for interrupt
00165 0a5d be41    SETC    INTM
00166 0a5e ef00    RET                              ;
00167 ---- ----    .END
>>>>> LINE:167  .END ENCOUNTERED
>>>>> FINISHED READING ALL FILES
>>>>> ASSEMBLY COMPLETE: ERRORS:0    WARNINGS:0
```

C.2 Source Code for Notch Filter Implementation on TMS20C50 DSP

```
      This program implements notch filter of order N = 685,
      attenuation level of 80 dB, transition frequency Δf of 50Hz,
      sampling frequency of 6.8 kHz, center frequency of 1 kHz and
      bandwidth of 275 Hz. The filter coefficients are obtained by
      Kaiser window design method. The program shown below is not
      complete and is only for one type of filter design, the
      complete program is in the companion diskette.
;


00001 ---- ----  ;
00002 ---- ----  ;
00003 ---- ----    .MMREGS
00004 ---- 0f00    .ds      0f00h
00005 0f00 0015  TA       .word    21          ;
00006 0f01 0015  RA       .word   21         ; These AIC
                                              ; registers values give
00007 ---- ----                               ; a sampling freq of ;
                                             ; 6.8 kHz
00008 ---- ----                    ;
00009 0f02 0023  TB       .word    35          ;
00010 0f03 0023  RB       .word    35          ;
00011 0f04 0028  AIC_CTR  .word    28h
00012 0f05 0000  OUTPUT   .word    0
00016 ---- ----  ;        .space   03000h
00017 ---- ----  ;        .data    03000h
00018 ---- ----    .include "Notch.txt"    ;A file containing
                                            ;filter coefficients
*****************************************
*     OPENING INCLUDE FILE Notch.txt
*****************************************
00001 ---- ----  nfd:
00002 0f09 0000   .q15          -3.8E-07
00003 0f0a 0000   .q15          3.65E-07
00004 0f0b 0000   .q15          6.29E-20
00005 0f0c 0000   .q15          -4.3E-07
00006 0f0d 0000   .q15          3.37E-07
00007 0f0e 0000   .q15          1.96E-06
00008 0f0f 0000   .q15          2.47E-06
00009 0f10 0000   .q15          4.43E-07
00010 0f11 0000   .q15          -2.9E-06
00011 0f12 0000   .q15          -4.6E-06
00012 0f13 0000   .q15          -2.4E-06
00013 0f14 0000   .q15          2.18E-06
00014 0f15 0000   .q15          5.19E-06
00015 0f16 0000   .q15          3.93E-06
00016 0f17 0000   .q15          -2.8E-07
00017 0f18 0000   .q15          -3.4E-06
00018 0f19 0000   .q15          -3E-06
00019 0f1a 0000   .q15          -6.2E-07
00020 0f1b 0000   .q15          3.49E-07
00021 0f1c 0000   .q15          -1.1E-06
00022 0f1d 0000   .q15          -1.9E-06
00023 0f1e 0000   .q15          1.27E-06
00024 0f1f 0000   .q15          6.99E-06
00025 0f20 0000   .q15          8.75E-06
```

```
00026 0f21 0000   .q15          1.59E-06
00027 0f22 0000   .q15          -1.1E-05
00028 0f23 0000   .q15          -1.8E-05
00029 0f24 0000   .q15          -9.9E-06
00030 0f25 0000   .q15          9.7E-06
00031 0f26 0000   .q15          2.49E-05
00032 0f27 0000   .q15          2.09E-05
00033 0f28 0000   .q15          -1.8E-06
00034 0f29 0000   .q15          -2.5E-05
00035 0f2a 0000   .q15          -2.9E-05
00036 0f2b 0000   .q15          -9.3E-06
00037 0f2c 0000   .q15          1.73E-05
00038 0f2d 0000   .q15          2.84E-05
00039 0f2e 0000   .q15          1.62E-05
00040 0f2f 0000   .q15          -5.5E-06
00041 0f30 0000   .q15          -1.7E-05
00042 0f31 0000   .q15          -1.1E-05
00043 0f32 0000   .q15          -8.7E-07
00044 0f33 0000   .q15          1.16E-18
00045 0f34 0000   .q15          -7.9E-06
00046 0f35 0000   .q15          -8E-06
00047 0f36 0000   .q15          1.11E-05
00048 0f37 0001   .q15          3.62E-05
00049 0f38 0001   .q15          3.65E-05
00050 0f39 0000   .q15          -3.7E-06
00051 0f3a ffff   .q15          -6E-05
00052 0f3b fffe   .q15          -7.9E-05
00053 0f3c 0000   .q15          -2.9E-05
00054 0f3d 0002   .q15          6.24E-05
00055 0f3e 0003   .q15          0.00012
00056 0f3f 0002   .q15          8.2E-05
00057 0f40 ffff   .q15          -3.5E-05
00058 0f41 fffc   .q15          -0.00014
00059 0f42 fffc   .q15          -0.00014
00060 0f43 0000   .q15          -2E-05
00061 0f44 0004   .q15          0.000125
00062 0f45 0005   .q15          0.000177
00063 0f46 0002   .q15          8.58E-05
00064 0f47 fffe   .q15          -7.7E-05
00065 0f48 fffb   .q15          -0.00018
00066 0f49 fffc   .q15          -0.00014
00067 0f4a 0000   .q15          1.14E-05
00068 0f4b 0004   .q15          0.000143
00069 0f4c 0005   .q15          0.000154
00070 0f4d 0001   .q15          4.64E-05
00071 0f4e fffe   .q15          -8.4E-05
00072 0f4f fffc   .q15          -0.00013
00073 0f50 fffe   .q15          -7.5E-05
00074 0f51 0000   .q15          2.63E-05
00075 0f52 0002   .q15          8.52E-05
00076 0f53 0002   .q15          6.76E-05
00077 0f54 0000   .q15          7.65E-06
00078 0f55 ffff   .q15          -3.6E-05
00079 0f56 ffff   .q15          -3.7E-05
00080 0f57 0000   .q15          -1.2E-05
00081 0f58 0000   .q15          7E-06
00082 0f59 0000   .q15          8.48E-06
00083 0f5a 0000   .q15          2.3E-06
00084 0f5b 0000   .q15          1.11E-17
00085 0f5c 0000   .q15          1.29E-07
```

```
00086 0f5d 0000   .q15          -2.7E-06
00087 0f5e 0000   .q15          -2.7E-06
00088 0f5f 0000   .q15          1.1E-05
                  .
                  .
                  .
00671 11a6 0000   .q15          -3.4E-06
00672 11a7 0000   .q15          -2.8E-07
00673 11a8 0000   .q15          3.93E-06
00674 11a9 0000   .q15          5.19E-06
00675 11aa 0000   .q15          2.18E-06
00676 11ab 0000   .q15          -2.4E-06
00677 11ac 0000   .q15          -4.6E-06
00678 11ad 0000   .q15          -2.9E-06
00679 11ae 0000   .q15          4.43E-07
00680 11af 0000   .q15          2.47E-06
00681 11b0 0000   .q15          1.96E-06
00682 11b1 0000   .q15          3.37E-07
00683 11b2 0000   .q15          -4.3E-07
00684 11b3 0000   .q15          6.29E-20
00685 11b4 0000   .q15          3.65E-07
00686 11b5 0000   .q15          -3.8E-07
>>>>> FINISHED READING ALL FILES
****************************************
*     CLOSING FILE Notch.txt
****************************************
00019 ---- ----
00020 ---- ----   ;This part reserves a a data location for an input
                  ;(XN) delayed by 685 (z^-1 )
00021 ---- ----  ;
00022 ---- ----
00023 11b6 0000  XN       .word
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      11b7 0000
      11b8 0000
      11b9 0000
      11ba 0000
      11bb 0000
      11bc 0000
      11bd 0000
      11be 0000
      11bf 0000
      11c0 0000
      11c1 0000
      11c2 0000
      11c3 0000
      11c4 0000
      11c5 0000
      11c6 0000
      11c7 0000
      11c8 0000
      11c9 0000
      11ca 0000
      11cb 0000
00024 11cc 0000  XN1      .word
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      11cd 0000
      11ce 0000
      11cf 0000
      11d0 0000
      11d1 0000
      11d2 0000
      11d3 0000
      11d4 0000
      11d5 0000
      11d6 0000
      11d7 0000

                .


                .


                .

      1453 0000
      1454 0000
      1455 0000
      1456 0000
      1457 0000
      1458 0000
      1459 0000
      145a 0000
      145b 0000
      145c 0000
      145d 0000
      145e 0000
      145f 0000
00054 1460 0000  XN31    .word   0,0
      1461 0000
00055 1462 0000  XNLAST  .word   0
00056 ---- ----
00057 ---- 080a   .ps      0080ah
00058 080a 7980  rint:    B       RECEIVE
      080b 0000
00059 080c 7980  xint:    B       TRANSMIT
      080d 0000
00060 ---- ----
00061 ---- 0a00   .ps       0a00h
00062 ---- ----   .entry                  ;
>>>>> ENTRY POINT SET TO 0a00
00063 ---- ----
;---------------------------------------------------------------
00064 ---- ----
00065 0a00 be41   SETC    INTM
00066 0a01 bc00   LDP     #0
00067 0a02 5d07   OPL     #0834h,PMST
      0a03 0834
00068 0a04 bf80   LACC    #0
      0a05 0000
00069 0a06 882a   SAMM    CWSR
00070 0a07 8828   SAMM    PDWSR
00071 0a08 be47   SETC    SXM             ;  SXM MUST BE SET
00072 0a09 ae04   SPLK    #022h,IMR       ; This turns on receive
```

```
                                                    ;interrupt only
      0a0a 0022
00073 ---- ----
00074 0a0b 7a80   CALL    AICINIT
      0a0c 0000
00075 0a0d ae04   SPLK    #12h,IMR
      0a0e 0012
00076 0a0f be42   CLRC    OVM
00077 0a10 bf00   SPM     0
00078 0a11 be40   CLRC    INTM
00079 ---- ----
00080 ---- ----  WAIT:                     ; a main program will run
                                          ; at this point
00081 0a12 7980   B       WAIT
      0a13 0a12
00082 ---- ----
00083 ---- ----  RECEIVE:
00084 0a14 bc23   LDP     #XN
00085 ---- ----
00086 0a15 be40   CLRC    INTM            ; enable interrupts for
                                          ; debugging, for DSK
                                          ; purposes
00087 ---- ----
00088 ---- ----
00089 ---- ----
00090 0a16 0820   LAMM    DRR             ; load accumulator with
                                          ; word received from AIC
00091 0a17 9036   SACL    XN              ; store the value of
                                          ; received word to a
                                          ; variable
00092 ---- ----
00093 0a18 bf08   LAR     AR0,#XNLAST     ; load AR0 with address of
                                          ; last delay element
      0a19 1462
00094 0a1a be59   ZAP                     ; ZERO ACC and product
                                          ; registers
00095 0a1b 8b88   MAR     *,AR0           ; AR0 is the current AR
                                          ; register.
00096 ---- ----
00097 0a1c bec4   RPT     #684            ; Repeat the next
                                          ; instruction 684 times
                                          ; through the complete
                                          ; filter coefficient data
      0a1d 02ac
00098 0a1e a390   MACD    #nfd,*-    ;
      0a1f 0f09
00099 0a20 be04   APAC                    ; Accumulate last product
00100 ---- ----
00101 0a21 9905   SACH    OUTPUT,1        ; the ACC data is
                                          ; transferred to
                                          ; OUTPUT and gets rid of
                                          ; extra sign bit.
00102 0a22 1105   LACC    OUTPUT,1
00103 0a23 be09   SFL
00104 0a24 bfb0   AND     #0fffch         ; Two LSB's must be zero
                                          ; for AIC
      0a25 fffc
00105 0a26 8821   SAMM    DXR             ; write output word to
                                          ; transmit register
00106 0a27 be3a   RETE                ;
```

```
register
00148 0a4c b803    ADD     #03h              ;
00149 0a4d 7a80    CALL    AIC_2ND           ;
      0a4e 0000
00150 0a4f ef00    RET                       ;
00151 ---- ----
00152 ---- ----  AIC_2ND:
00153 0a50 bc00    LDP     #0                ; Data page point is 0
(MM regs)
00154 0a51 9821    SACH    DXR               ; send ACChi 00
00155 0a52 be40    CLRC    INTM              ; enable interrupts
00156 0a53 be22    IDLE                      ; wait for interrupt
00157 0a54 bf9f    ADD     #6h,15            ; 0000 0000 0000 0011
XXXX XXXX XXXX XXXX b
      0a55 0006
00158 0a56 9821    SACH    DXR               ; send ACChi to initiate
secondary protocol
00159 0a57 be22    IDLE                      ; wait for interrupt
00160 0a58 9021    SACL    DXR               ; send the T register
data
00161 0a59 be22    IDLE                      ; wait for interrupt
00162 0a5a b900    LACL    #0                ; clear ACClo
00163 0a5b 9021    SACL    DXR               ; send another to make
sure 1st word got sent
00164 0a5c be22    IDLE                      ; wait for interrupt
00165 0a5d be41    SETC    INTM
00166 0a5e ef00    RET                       ;
00167 ---- ----    .END
>>>>> LINE:167  .END ENCOUNTERED
>>>>> FINISHED READING ALL FILES
>>>>> ASSEMBLY COMPLETE: ERRORS:0   WARNINGS:0
```

Bibliography

1. Andreas Antoniou, 1990, *Digital Filters Analysis and Design*, McGraw-Hill, Inc., New York.
2. Edward B. Kalin, 1997, *The American Radio Relay League Hand Book CD* , The ARRL, Inc., Newington.
3. Lothar Wiesner, 1979, *Telegraph and Data Transmission over Shortwave Radio Link*, Heyden & Son Ltd., London.
4. Kun-Shan Lin, 1987, *Digital Signal Processing Applications, with the TMS320 Family*, Volume I, Prentice-Hall, Inc., New Jersey.
5. S. M Bozic, 1994, *Digital and Kalman Filtering*, Edward Arnold, PLC., London.
6. Sanjit K. Mitra and James F. Kaiser, 1993, *Hand Book for Digital Signal Processing*, John Wiley & Sons, Inc., New York.
7. Sophocles J. Orpahnidis, 1996, *Introduction to Signal Processing*, Prentice-Hall, Inc., New Jersey.
8. Texas Instruments Incorporated, 1996, *TMS320C5X User's Guide*, Customs Printing Company, Missouri.
9. Texas Instruments Incorporated, 1996, *TMS320C5X DSP Starter Kit User's Guide*, Customs Printing Company, Missouri.
10. Trevor Housley, 1979, *Data communication and Teleprocessing Systems*, Prentice-Hall, Inc., New Jersey.
11. W. I Schroeder, 1996, *HamComm Manual*, Version 3.1.
12. http://accessone.com
13. http:// kantronics.com/hfafskor.htm
14. http:// siskin.com/smc-comms.com

Declaration

I, the undersigned, hereby declare that this thesis is my original work carried out under the supervision of Dr.-Ing Habil P. Haferkorn, has not been presented as a thesis for a degree program in any other university and that all sources used for the thesis are duly acknowledged.

Seble Mengesha