DESIGN AND DEVELOPMENT OF A FREQUENCY HOPPER BASED ON THE DECT SYSTEM FOR THE 902-928 MHz ISM BAND

by

Francis Dominique

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

[illegible]

Dr. Jeffrey H. Reed
(Chairman)

[illegible]

Dr. Theodore S. Rappaport

[illegible]

Dr. Brian D. Woerner

December 1995

Blacksburg, Virginia

c.2

LD
5655
V855
1995
D665
c.2

DESIGN AND DEVELOPMENT OF A FREQUENCY HOPPER BASED ON THE DECT SYSTEM FOR THE 902-928 MHz ISM BAND

by

Francis Dominique

Committee Chairman: **Dr. Jeffrey Hugh Reed**

Bradley Department of Electrical Engineering

ABSTRACT

The Digital European Cordless Telecommunications (DECT) standard defines a high performance digital communications system for two-way transfer of speech and data information. This thesis is concerned with the development of a frequency hopping spread spectrum transceiver based on the DECT radio frequency (RF) front end, and fully compliant with FCC Part 15 rules for transmissions in the 902-928 MHz ISM band. A theoretical design of a slow frequency hopper (SFH) based on an existing DECT RF front end and a custom designed system controller was developed. An innovative FH synchronization technique that eliminates the need for a separate FH tracking loop and can be implemented completely at baseband was designed. The practicality of the design was verified through extensive simulations. The SFH design was then implemented in prototype hardware using a digital signal processor acting as the system controller and using the DECT RF front end. Results show that the existing DECT physical layer can be successfully modified for spread spectrum transmissions in the ISM band fully compliant with FCC regulations.

Acknowledgements

I would like to convey my heart-felt thanks to Dr. Reed for provoding me with the technical support to complete this thesis. He has always been a source of inspiration to me. I look forward to working with him in the years to come.

I am indebted to Dr. Rappaport and Dr. Woerner for their valuable advice. I appreciate their help in carefully reviewing this thesis report.

I thank Michael Schwartz, Dan Fague and Michael O' Hearn at the Wireless Communications Group at National Semiconductor Corporation for their support of the DECT systems used for the project.

I thank Dan Bailey for his help in implementing the frequency hopper.

I wish to thank my parents for their support and encouragement.

I would like to thank Prab Koushik for his help and support.

I would like to thank Rias Muhamed, Paul Petrus and all my friends and colleagues at MPRG for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Portable (cordless) radio telephone communication systems are used extensively today. Cordless telecommunications have been recognized to complement mobile radio telephony in many areas of applications. The first generation cordless telephone systems were analog Frequency Modulation (FM) systems with limited range and no multiple access capability. Today's cordless systems are fully digital and provide more and better services than conventional analog cordless systems.

The Digital European Cordless Telecommunications (DECT) standard [1] defines a high performance wireless communications system suitable for the two-way transfer of voice, data and video information. DECT is a Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), Time Division Duplexed (TDD) multi-purpose telecommunication system based on low power, portable telephones (or data terminals) that access a fixed micro-cellular infrastructure. An effective range of up to 100 meters is possible indoors, 300 meters outdoors. DECT is designed to provide full, on site, mobile telecommunications for both voice and data services.

The objective of this thesis is to study the DECT system, model one of its implementations, and finally devise a transceiver design for portable communications that would enable the system to operate in the 902-928 MHz ISM band. The 902 - 928 MHz band has been assigned, on a non-protected basis, to unlicensed spread spectrum devices subject to constraints outlined in the FCC Part 15.247 specifications [2]. The rules allow for the transmissions of Direct Sequence, Frequency Hopping, and hybrid combinations of both.

Since transmissions in the ISM band have to be spread spectrum for unlicensed operation, this calls for the design of a spread spectrum transceiver compliant with FCC Part 15 rules, and which makes use of the existing DECT radio frequency front end. The DECT implementation used for this thesis is based on the implementation by National Semiconductor Corporation (NSC). The entire DECT physical layer has been extensively modeled and simulated using the Signal Processing WorkSystem (SPW), which is a popular simulation environment. A study of the different spread spectrum techniques and their suitability for implementation using the DECT RF front end was then carried out. The spread spectrum technique chosen is packet-based slow frequency hopping (SFH) with each data frame transmitted on a different carrier frequency.

Slow frequency hopped portable systems have a number of advantages over non-hopped systems:

1. Since the system is designed to be a portable cordless transceiver, the channel can be frequency selective and fairly static, and in this situation SFH provides a form of frequency diversity through exploitation of frequency selectivity over the given bandwidth (of 26 MHz). The fading process is decorrelated from hop to hop over the duration of long fades [3]. This highlights the advantage of the SFH system, which is able to sample a number of channels, over a non-hopped system, which if assigned a poor channel, would experience a prolonged period of unsatisfactory performance.

2. In a multi-user environment, the intercell co-channel interference is sampled rather than summed. Hence there is no danger of a single rogue user disrupting an entire cell. This interference reduction effect has been termed interferer diversity. Interference diversity causes the interference experienced by any user to be reduced since the interference experienced by the user comes not from a single dominant interferer, but rather from the aggregate of all users, each sampled one at a time.

3. The hardware architecture of the DECT system makes it more amenable to implement a slow frequency hopper.

A theoretical design of the SFH system based on the DECT RF front end was developed. The SPW models developed for the DECT system were then suitably modified to model and simulate the SFH system. Once the simulations verified the practicality of the design, the SFH transceiver was then implemented in prototype form. One of the main challenges in the design of the SFH system was the development of a simple, rapid, and robust frequency hop synchronization technique. An innovative SFH synchronization technique has been developed that permits its implementation entirely at baseband and in software. The flexibility of this (software) approach is that the (synchronization) algorithm can be modified or even changed without any (major) hardware modifications. In addition, this synchronization technique eliminates a separate tracking loop for FH synchronization. Instead, the existing DECT bit timing recovery loop is also used as the FH tracking loop, and this results in a significant saving in terms of the implementation complexity.

1.2 Format of the Thesis

The thesis is organized as follows:

Chapter 2 describes the technical details of the physical layer of the DECT system and National Semiconductor's implementation of a DECT transceiver.

Chapter 3 deals with the issues associated with the design of the frequency hop spread spectrum transceiver in detail.

Chapter 4 details the Signal Processing Workstation (SPW) modeling of the frequency hopping radio based upon the DECT RF front end.

Chapter 5 details the hardware implementation of the frequency hopping radio. The hardware interface between the DECT RF front end and the Analog Devices AD-2111/ AD-21020 DSP development systems that serve as the burst mode controllers is discussed in detail. The software that controls the scheduling of the various housekeeping jobs and the frequency management algorithms is detailed.

Chapter 6 details the simulation results obtained from SPW.

Chapter 7 summarizes the results of this work and also details how the transceiver can be further improved and expanded to include more modes of operation.

Chapter 2

The Digital European Cordless Telecommunications System

2.1 Introduction to the DECT System

The Digital European Cordless Telecommunications (DECT) standard was defined by the European Telecommunications Standards Institute (ETSI). The DECT standard defines a high performance wireless communications system suitable for the two-way transfer of voice, data and video information. DECT is a multi-purpose telecommunication system based on low power, portable telephones (or data terminals) that access a fixed micro-cellular infrastructure. The DECT system is a microcellular system with cell sizes ranging from 50 to 300 meters in diameter. The DECT standard is extremely robust, and when configured in a multi-cell configuration, the handsets are able to roam throughout an area and maintain continuous connectivity during the call. This is possible due to the mobile directed cell to cell handoff capability built into the DECT standard. The Generic Access Protocol (GAP) specification allows interoperability between handsets and base stations manufactured by different companies. The DECT system was originally designed for the cordless telephone segment, and in this regard was designed for the transmission of Adaptive Differential Pulse Code Modulation digitized speech data at 32 kbps. Since then, applications for DECT have evolved and some of the more recent ones include wireless PBX for business, wireless local area networks, and wireless local loops.

2.1.1 Applications

A wide number of applications for the DECT system have developed over the years. Some of the more important ones [4] are discussed below :

2.1.1.1 Wireless Private Branch Exchange (WPBX)

WPBXs will significantly enhance business communications. One of the most valuable features of a DECT WPBX is a roaming capability and seamless hand-over between cells. This means that a user may move freely throughout a building or a campus while maintaining uninterrupted phone connections. Dynamic channel allocations greatly simplify the installation of base stations throughout the site since the need for complex frequency planning is eliminated.

2.1.1.2 Residential DECT Cordless Telephones

The current generation of cordless telephones suffers from poor voice quality, lack of privacy, interference and limited range. The DECT standard provides a much higher level of quality through the use of digitized voice. Privacy is guaranteed by the use of sophisticated encryption techniques. In addition, the DECT access scheme allows a single base station to serve up to 12 cordless extensions throughout the home simultaneously. They are also expected to be less expensive than current systems.

2.1.1.3 Wireless DECT Local Area Networks (WLANs)

WLANs will remove the limitations of current LANs. WLANs will allow multiple laptop or palmtop computers to coordinate and collaborate on common projects independent of their physical location. DECT's high throughput and built-in data and networking capability, coupled with seamless hand-over, makes it ideal for WLANs and for applications such as batch file transfers, real time file access, and remote terminal service.

2.1.1.4 Wireless Local Loop

In places where the telecommunications infrastructure is relatively undeveloped, the use of radio links to replace the copper links is an attractive alternative for providing subscribers fast and low cost links.

2.2 Details of the Physical Layer of the DECT System

The DECT system is based on a microcellular system architecture, with cell sizes ranging from 50 to 300 meters in diameter. Ten radio carrier frequencies in the 1.88-1.90 GHz band, each 1.728 MHz wide, carry 12 full duplex channels organized in a Time Division Multiplex, Time Division Duplex (TDM/TDD) manner [5][6]. Figure 2.1 shows the structure of the DECT frame. The entire frame is 10 ms in duration, with a 5 ms transmit and a 5 ms receive sub-frame. Each sub-frame is divided into 12 time slots, each containing 480 bits for a total duration of 415.67 microseconds. The DECT specifications are detailed in Table 2.1.

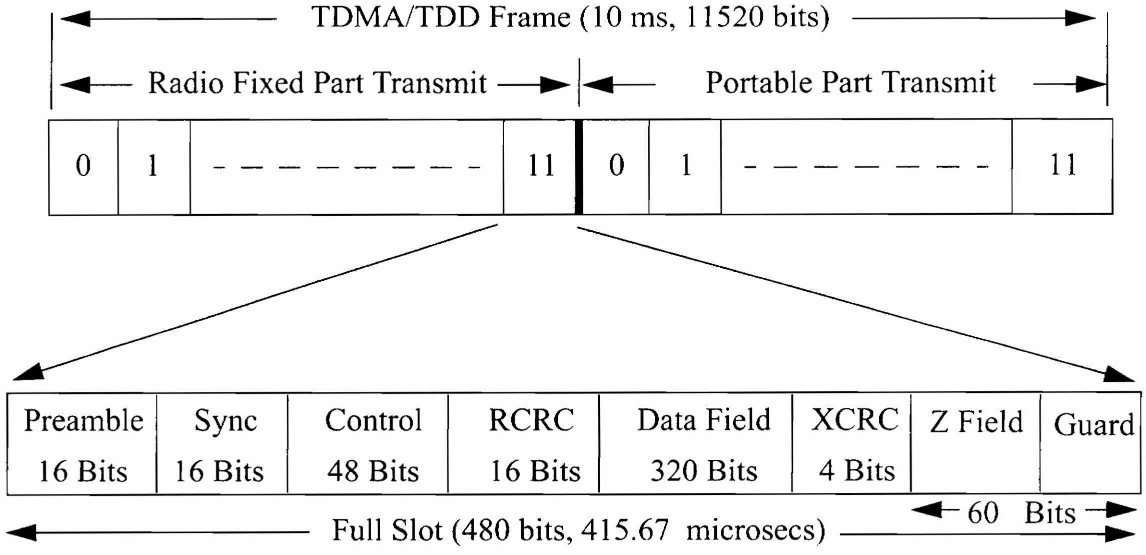


Figure 2.1 Slot and Frame Structure of the DECT System

Table 2.1 Technical Specifications of the DECT System

Parameters	**Specification**
Access	TDMA/TDD & FDMA
RF Channels	0 : 1.897344 GHz 9 : 1.881792 GHz
Number of RF Channels	10
Channel Spacing	1.728 MHz
Peak Transmitted Power	250 milliwatts
Frequency Accuracy	50 KHz

Parameters	**Specification**
Synthesizer Switching Speed	Base Station : 30 μsecs Handset : 450 μsecs
Frame Length	10 milliseconds
Modulation	GMSK
Data Rate	1.152 Mbps
FM Deviation	288 KHz
Baseband Filter	Gaussian BT = 0.5
Speech Codec	ADPCM @ 32 Kbps

2.3 General Implementation of the DECT System

Figure 2.2 shows the general block diagram of the DECT transmitter. Figure 2.3 shows the block diagram of the DECT receiver [7]. Speech is sampled at 8 KHz, and digitized using an ADPCM codec at 32 kbps. A transmit frame is constructed by the burst mode controller and the resulting frame is then sent to the gaussian filter for pulse shaping in the proper time slot. The burst mode controller simultaneously programs the frequency sythesizer for the desired RF channel. Gaussian Minimum Shift Keying (GMSK) modulation is then carried out. The RF signal is then sent to the power amplifier, and finally through the final bandpass filter to attenuate out of band spectral energy before being fed to the antenna. The receiver at the proper time (after timing synchronization has been achieved) switches the antenna output to the Low Noise Amplifier (LNA). After the received signal voltage has been amplified by the LNA, the RF signal is downconverted to an Intermediate Frequency (IF), which for DECT has been specified to be 110.592 MHz.

The output of the mixer is then filtered by the IF bandpass filter to remove the image frequencies before being demodulated at the IF by a Limiter-Discriminator. The output of the demodulator is then lowpass filtered to recover the baseband data signal. The analog baseband signal is then shaped into a Non-Return to Zero (NRZ) digital signal before being sent to the burst mode controller where the header and the control information are stripped from the received signal. The digitized speech data is then buffered and fed to the ADPCM codec for reconstruction of the speech signal.

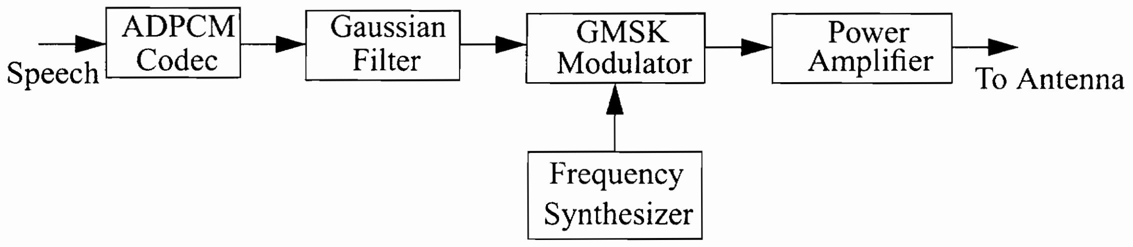


Figure 2.2 Block Diagram of the DECT Transmitter

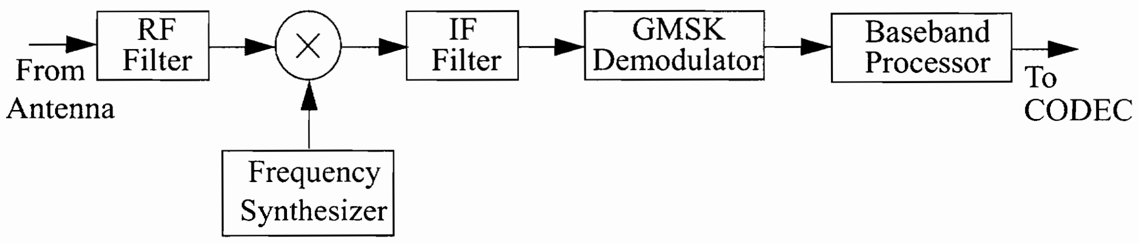


Figure 2.3 Block Diagram of the DECT Receiver

2.4 National Semiconductor Implementation of the DECT System

National Semiconductor's solution for the DECT system makes use of a direct conversion transmitter and a single conversion limiter discriminator receiver [8][9]. The entire transceiver can be implemented using four National chips along with a few other basic components. The four National Semiconductor integrated circuits which form the basis of the DECT transceiver are

1. LMX2411 Baseband Processor.
2. LMX2315 Phase Locked Loop.
3. LMX2240 IF Limiter Discriminator.
4. LMX2216 Low Noise Amplifier.

The block diagram of the NSC implementation is shown in Figure 2.4.

2.4.1 Transmitter

The transmitter essentially consists of the speech CODEC, gaussian filter, the FM modulator, and the RF power amplifier.

2.4.1.1 Gaussian Filter

The transmit frame is constructed by the burst controller using the digitized speech samples from an external ADPCM speech CODEC. The data bits are then fed to the LMX2411 baseband processor [10][11] through the TX_Data input pin. The transmit portion of the LMX2411 serves as a pulse shaper for the input serial data, delivering a gaussian filtered data stream capable of modulating a VCO directly to create the GMSK signal. The LMX2411 uses a mask-programmable Read-Only Memory (ROM) look-up table to

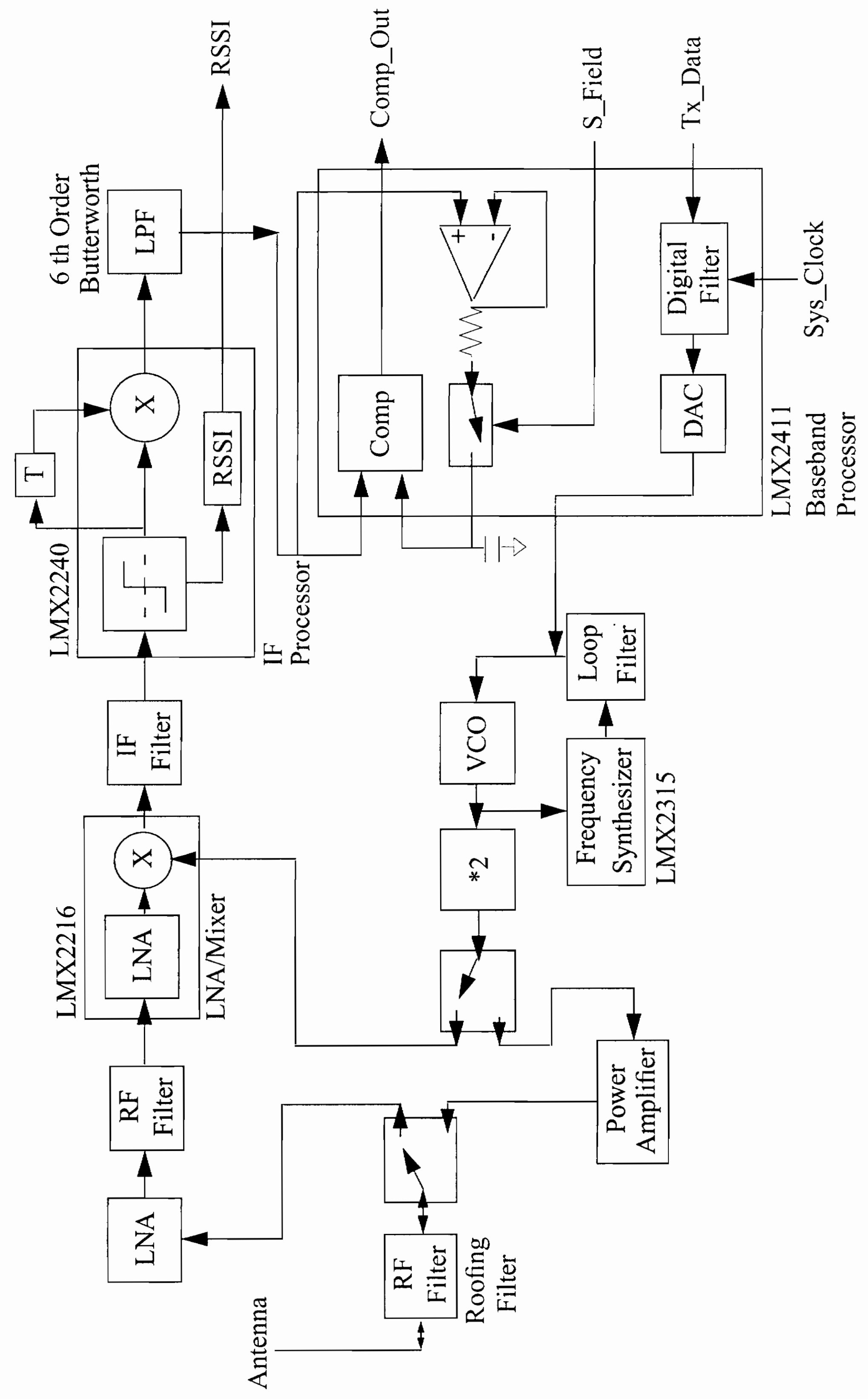


Figure 2.4 Implementation of the DECT System

construct the data pulse response of the gaussian filter. The gaussian filter ROM DAC uses a three bit memory to represent the filter's pulse response. The result is an effective 3 bit time delay from input of the first bit to the time when that bit is actually output from the filter. When using the LMX2411 transmit section, the bits must be sent two bit times before they are seen at the antenna to account for this small delay in the ROM DAC. There is also a half bit delay to allow the LMX2411 to sample the data near the center of the data bit. Also, the end of the information data stream must be padded by three data bits to push the last data bit through the filter. Finally, it should be noted that after the TX_PD pin goes LOW, the ROM filter output will be at the mid-band voltage until the first edge of TX_Data, which is used for synchronizing the internal clock with the transmitted clock. For DECT, this filter has a bandwidth of half the input bit rate (BT=0.5). Three different system clocks selected by two external pins can be used with the ROM based filter. These pins (ROM Sel1 & 2) choose the desired oversampling clock. The oversampling clocks used are 6X, 8X and 9X. The output of the ROM addresses an 8 bit voltage mode digital to analog converter (DAC).

2.4.1.2 FM Modulator

The output of the DAC of the LMX2411 Baseband Processor is used to modulate directly the VCO of the LMX2315 PLL [12]. There are several techniques that can be used to modulate a VCO directly. The most common are modulation "in the loop" and modulation "over the loop" [13]. Modulation in the loop can be used when the output signal is narrowband with respect to the loop filter, and modulation over the loop can be used when the time required to switch frequencies is relatively long. Neither of these conditions are valid for the DECT system. The lock times of the phase locked loop must be very fast (about 30 microseconds) and the output signal is much wider than the loop filter bandwidth. National's solution is to use a technique called "open loop modulation." In open loop modulation, the PLL loop is actually opened for a brief period to allow the modulation to occur. When transmission is required, the loop is closed to lock the VCO to the desired carrier frequency. The modulating signal is then turned on, and the loop remains closed to re-lock to the center frequency. The modulating voltage is then added to the loop filter voltage (the voltage required to maintain the VCO at the desired center frequency) at either a modulation port or a tuning port. The loop is then opened and the modulation occurs. Once the modulation is finished, the loop is closed again, and the PLL can be tuned to the receive frequency. This technique can be used since the DECT transmissions are bursty in nature, thus allowing PLL loop to be opened and closed as required. There are some elements of open loop modulation that can limit performance. Some of the main ones include frequency pushing, load pulling, and frequency drift. Frequency pushing is normally described as a change in the VCO output frequency caused by a change in the supply voltage. Load pulling is a change in the VCO output frequency caused by a change in the load that the VCO output buffer sees. Frequency drift can be caused by RF coupling or by droop in the VCO tuning voltage. Droop in the VCO tuning voltage can be caused by leakage from the PLL charge pump, the loop filter components, or the VCO's tuning varactor. The PLL is rated to 2.0 GHz operation and the VCO used is about 130 MHz wide. The PLL is optimized for fast lock times.

2.4.2 Receiver

The receiver essentially consists of a front end RF filter, a low noise amplifier/mixer, an IF limiter discriminator, a baseband lowpass filter, and a baseband processor to reshape the pulses, provide DC compensation, and recover timing information.

2.4.2.1 Front End RF Filter

The front end RF filter is a ceramic bandpass filter with a passband from 1.88-1.90 GHz. Any commercially available RF filter meeting the bandpass specifications can be used.

2.4.2.2 LNA/Mixer

The LNA/Mixer is implemented by the LMX2216 [14]. The LMX2216 is a low voltage and bias current device. The mixer used is based on the Gilbert cell structure, which has highly desirable characteristics in terms of the isolation, and the harmonic suppression due to its balanced structure.

2.4.2.3 IF Filter

The output of the mixer is then filtered by the IF bandpass filter to eliminate the double frequency terms and higher harmonics generated by the mixing operation. The IF filter used is a SAW filter with a center pass frequency of 110.592 MHz.

2.4.2.4 IF Limiter-Discriminator

NSC's DECT implementation uses an IF based limiter discriminator to demodulate the GMSK signal [15][16]. There are two types of discriminators that can be used to demodulate FM signals. The first is a delay line discriminator, which uses a delay in one path of the received signal to introduce a phase difference between it and the received signal. The operation of the delay line discriminator is derived as follows [17]:

Assume that the FM modulated signal can be expressed as

$$S(t) = \cos(\omega_c t + m(t)) \tag{2.1}$$

where $m(t) = m \int_{-\infty}^{t} b(t)dt$ and *b(t)* is the modulating baseband signal. The constant m is defined as $m = 2\Delta f T_b$, where Δf is the frequency deviation constant and T_b is the bit period. The signal *S(t)* must be delayed by some τ so that

$$I(t) = S(t+\tau) = \cos(\omega_c(t+\tau) + m(t+\tau)) \quad (2.2)$$

If the delay τ is such that

$$\omega_c\tau = 2n\pi + \frac{\pi}{2}, n = 0, 1, 2, \ldots \quad (2.3)$$

then

$$S(t+\tau) = \sin(\omega_c t + m(t+\tau)) \quad (2.4)$$

Multiplying (2.1) and (2.2), we get

$$S(t)I(t) = \cos(\omega_c(t) + m(t))\sin(\omega_c(t) + m(t+\tau)) \quad (2.5)$$

$$= \frac{1}{2}\sin(2\omega_c t + m(t) + m(t+\tau)) + \frac{1}{2}\sin(m(t+\tau) - m(t)) \quad (2.7)$$

The double frequency component can be filtered off with a lowpass filter. If τ is kept small, then

$$\frac{1}{2}\sin(m(t+\tau) - m(t)) \approx \frac{1}{2}[m(t+\tau) - m(t)] \quad (2.8)$$

$$= \frac{m}{2}\int_{-\infty}^{(t+\tau)} b(t)d(t) - \frac{m}{2}\int_{-\infty}^{t} b(t)d(t) \quad (2.9)$$

$$= \frac{m}{2}\int_{t}^{(t+\tau)} b(t)dt \quad (2.10)$$

$$\approx \frac{m}{2}\tau b(t) \quad (2.11)$$

Therefore the object of a delay line is to maximize the delay while retaining the approximations necessary to satisfy (2.3), $\tau < 0.1T_b$.

The second type of discriminator relies on a quadrature tank to introduce directly a phase shift in the received signal. A quadrature is produced by passing the FM signal through a capacitor connected in series with a parallel resonant circuit that is tuned to the carrier frequency. The series capacitance produces a 90 degree phase shift, and the resonant circuit produces an additional phase shift proportional to the instantaneous frequency deviation from the carrier frequency. The operation of the quadrature discriminator is derived as follows :

The FM signal can be expressed as

$$S(t) = \cos(\omega_c t + m(t)) \tag{2.12}$$

and the quadrature signal as

$$I(t) = K_1 \sin\left(\omega_c t + m(t) + K_2 \frac{dm(t)}{dt}\right) \tag{2.13}$$

where K_1 and K_2 are constants that depend upon the component values used in the series capacitor and the parallel resonant circuit. These two signals are multiplied together to produce the output signal, which after lowpass filtering can be expressed as

$$V_{out}(t) = \frac{1}{2}K_1 \sin\left(K_2 \frac{dm(t)}{dt}\right) \tag{2.14}$$

For K_2 and $\frac{dm(t)}{dt}$ sufficiently small, $\sin(x) \approx x$, and the output can be written [18] as

$$V_{out}(t) = \frac{1}{2}K_1 K_2 mb(t) \tag{2.15}$$

The discriminator operates best when the inputs to it are hard limited. If the input signal is small enough such that the IF amplifier cannot limit it, the limiter output voltage swing will suffer. The two inputs to the discriminator can have different peak to peak voltage swings as long as both are over the minimum limit for the discriminator. This allows some insertion loss for the quadrature tank circuit. The quadrature circuit can also affect the discriminator output voltage swing. The discriminator output voltage swing specified assumes perfect quadrature at the frequency of interest. Practically, this is not possible. This is in part due to the high frequency of the IF and the proportionally very narrow bandwidth of the desired signal, thus making the quadrature circuit difficult to construct. With moderately high quality factor components, however, a reasonable phase shift can be achieved with a single pole tank.

The IF limiter discriminator used is the LMX2240 [15]. The LMX2240 is a low power IF processor that includes a frequency discriminator, an IF hard limiting amplifier and a received signal strength indicator (RSSI). The LMX2240 is capable of differentially demodulating a FM or AM signal with as high an IF as 150 MHz, thus avoiding a costly second down conversion. The limiting amplifier has a typical gain of 70 dB and a sensitivity of -70 dBm. The limiter is implemented as a five stage amplifier with internal compensation at each stage to further enhance stability. The input to the limiter is matched to an impedance of 150 ohms at a frequency of 110 MHz. The output impedance is about 100 ohms. The received signal strength indicator has a range of 70 dB. Its output voltage is proportional to the logarithm of the input signal level. The RSSI has a sensitivity of -82 dBm. The frequency discriminator is a Gilbert cell mixer that requires an external tank circuit. When using a quadrature tank, one's goal is to create a nominal 90 degree phase shift between the inputs to the mixer, and then introduce instantaneous phase shifts based on the instantaneous frequency of the input. A dual tuned tank circuit is used to perform the instantaneous phase shifts, with a series inductor between them to achieve the nominal $90°$ phase shift. The dual tuned quadrature tank for high frequency discriminators places the two poles of the bandpass filter at the edges of the signal bandwidth. A quad tank with as high a quality factor (Q) as possible is desired as finite Q's will reduce the amount of phase shift achievable.

2.4.2.5 Lowpass Filter

The output of the limiter discriminator is lowpass filtered using a sixth order butterworth lowpass filter to remove the double the frequency terms and harmonics. A 3 dB bandwidth of about 1 MHz is used.

2.4.2.6 Baseband Processor

The baseband processor consists of two sections: transmit and receive. The transmit section basically acts as a pulse shaper and is described in detail in section (2.4.1.1). The receive section of the baseband processor consists of a high speed comparator and a DC compensation circuit [10].

The high speed comparator is used to shape the lowpass filtered data stream into a rectangular waveform suitable for further digital processing. The comparator's threshold can be set by using either an external voltage or the internal DC compensation circuit. The internal analog DC compensation circuit is designed to provide a simple yet accurate way to track and correct the effects of DC drift due to center carrier drift. This loop provides an accurate representation of the center voltage of the received signal. Here, the received demodulated signal is input to a sample and hold (S & H) buffer amplifier. The S & H circuit allows a single RC filter to average the DC value of the received signal without distorting it. The acquisition time of the DC compensation circuit from full discharge of the hold capacitor to a 35% duty cycle is about 37 milliseconds. This DC value is connected to the "-" input of the comparator.

DC compensation works as follows: When the DECT preamble starts, the CMOS switch of the sample and hold buffer is closed, allowing the hold capacitor to charge up to the mean DC value of the demodulated signal. When the preamble is over, the switch is opened and the threshold voltage is held until the next burst. This solution avoids the problem of long strings of 1's and 0's that conventional averaging circuits have, while still reacting quickly to acquire the proper DC average at the beginning of a burst.

2.4.2.7 Symbol Timing Recovery

National Semiconductor does not have a custom solution, and the symbol timing recovery block is usually implemented using the squaring loop [19]. The function of the symbol timing recovery circuit is to extract the frequency and phase of the transmitters' data clock. The timing recovery scheme must also account for delays encountered over the channel and within the transmitter and receiver hardware. The recovered clock is never ideal, and the error it causes manifests itself in the form of jitter. Timing jitter is usually dominated by the following [20]:

1. Pattern noise which is data dependent and caused by the intersymbol interference present within the GMSK signal. Pattern noise is dependent on the BT product.
2. Zero crossing jitter, which is mainly caused by the receiver IF filter. Zero crossing jitter increases as the receiver IF bandwidth decreases. The jitter also increases rapidly as the IF filter group delay deviates from a constant.
3. Additive noise, which is due to the receiver thermal noise. Its impact is dependent on the received signal level.

Pattern noise and zero crossing jitter set a lower bound on the recovered clock jitter, regardless of the signal strength. The signal at the output of the discriminator has a continuous spectrum and hence cannot be directly used for timing recovery. If the discriminator output is applied to an even order nonlinearity, the output of the nonlinearity contains a discrete component at the transmitter clock frequency. The power spectral density (PSD) at the output of the nonlinearity contains three components :

1. A DC component that accounts for almost half the total power.
2. A discrete component at the data clock frequency.
3. A continuous component that is directly related to the received data pattern.

The discrete component at the data clock frequency can then be filtered out, to be used as the recovered clock.

One of the common implementations is the squaring loop technique. In this technique, the input signal is delayed and then multiplied by the signal in the undelayed branch. The output of the multiplier can be represented in an exponential Fourier series with coefficients [21]:

$$C_n = (-1)\left(n\frac{|\tau|}{2T_S}sinc\left(n\frac{\tau}{T_B}\right)\exp\left(\frac{-j2\pi n|\tau|}{T_B}\right)\right), 0 \le \tau \le T_B \qquad (2.16)$$

where τ is the delay and T_S is the symbol period. The power in the spectral component at the bit clock is $2|C_1|^2$, and it can be shown that it is a maximum for a delay of $\tau = \frac{T_B}{2}$. The output of the multiplier is then applied to a bandpass filter to eliminate the DC component and lightly filter the continuous component. The discrete frequency at the bit clock frequency is not affected by the filtering, and is then tracked by a narrowband PLL. In steady state, the VCO of the PLL will follow the phase of the discrete component, with a slight jitter due to the pattern noise, zero crossing jitter and the additive noise.

Chapter 3

Design of the Frequency Hopper

3.1 Introduction

The objective of this chapter is to outline the issues and the design of the frequency hopping transceiver for operation in the 902-928 MHz ISM band. The chapter begins with a review of the FCC Part 15 requirements for the 902-928 MHz band. A design is then detailed and the various issues and trade-offs are discussed.

3.2 FCC Part 15 Specifications for Frequency Hop Spread Spectrum Transmissions in the 902-928 MHz ISM Band

Minimum number of hopping frequencies	= 50.
Maximum allowed 20 dB bandwidth	= 500 kHz.
Average occupancy time	= 0.4 secs / 20 secs.
Maximum peak output power	= 1 watt.
Maximum antenna gain	= 6 dBi.

3.3 Design Issues

3.3.1 Choice of the Spread Spectrum Modulation

The spread spectrum technique chosen for use is frequency hopping. The reasons why frequency hopping is chosen are as follows :

1. In order to keep the complexity, and hence the cost of the transceiver down, hybrid combinations of Direct Sequence (DS) and Frequency Hopping (FH) were not considered.

2. Multipath is a major impairment. DS is able to reject multipath by a factor proportional to the processing gain, but only if the delays are of the order of a chip period or more. The DS system if used with DECT, can have a maximum chipping rate of 1.152 Mchips per second since the existing DECT RF front end can accomodate a maximum data rate of 1.152 Mchips per second without any modifications to the existing hardware. It is possible to go to a higher chipping rate but this would require the bandwidths of the filters used in both the transmitter and the receiver to be widened. The chip period is thus 868 nanoseconds; therefore, the minimum delay spread that the DS system will be able to resolve is 868 nanoseconds. Typical indoor delay spreads are usually less than 300 nanoseconds. In this case, the DS signal will fade just as a narrowband signal will. The effect on a frequency hopper is different. The frequency selective fading caused by multipath will produce a series of notches across the band. An echo of about 200 nanoseconds would produce about four 1 MHz notches across the 26 MHz band [2]. Since the maximum allowed FH bandwidth is 500 KHz, 8 out of the 50 frequencies would be unusuable, a reduction in the throughout by about 15 %. Thus in this situation, the DS signal experiences severe fading whereas a FH system suffers from a slight throughput loss.

3. Frequency hopping shows more tolerance for high power narrowband interferers than direct sequence [2]. The FH system is able to avoid narrowband interferers, at least part of the time, as it hops across the band of interest. A DS system on the other hand will be sub-

ject to the interferer all of the time if it exists in the same band of interest. If the processing gain of the DS system is not sufficient to overcome the interference power, the BER performance of the DS system will be degraded severely. Since a proceesing gain of only about 10 to 15 is possible for this application, a high power interferer could easily disrupt operation.

4. FH systems do not suffer from the near-far problem that plague conventional DS receivers [22].

3.3.2 Frequency Hopping System Parameters

Frame duration	= 4.608 milliseconds.
Digitized speech data rate	= 52.083 kb/s.
Number of CVSDM data bits in each frame	= 480 bits/frame.
Overhead added for the purposes of synchronization and timing recovery	= 128 bits/frame.
Overhead for setting up and maintaining the physical connection	= 112 bits/frame.
Total number of bits in a frame	= 720 bits.
Overall bit rate for each channel	= 156.25 kb/s.
20 dB double sided bandwidth for GMSK	= 1.5*Bit Rate.
	= 234.375 KHz.
Number of frequency hopping channels available	= 26 MHz/234.3 KHz.
	= 111 channels.
Bit period	= 6.4 microseconds.
Synthesizer switching speed	= 100 microseconds.
Transmitted power	= 200 milliwatts.
Antenna	= $\lambda/4$ monopole.

3.3.3 Frequency Hop Architecture

The system being proposed is a Time Division Duplex / Frequency Hopping system (TDD/FH). Each frequency hopping channel will accommodate a 32 kb/s full duplex voice/data communication link, with the forward and reverse voice/data link being organized in a TDD fashion. The use of a TDD architecture simplifies the hardware considerably. By removing the need to transmit and receive simultaneously, the diplexer/bandpass filters required to separate the transmit and receive signals can be removed. In addition, a number of the RF hardware sub-systems such as the roofing RF filter and the frequency synthesizer can now be shared between the transmit and receive sections of the transceiver. Finally, the TDD structure permits a simpler frequency synthesizer to be used since the guard interval between the transmit and receive slots can be used to switch the synthesizer.

3.3.4 Hop Rate

The proposed system has a hop rate of 109 hops per second which corresponds to the frame rate, i.e. each frame is transmitted on a different carrier frequency. The FH system can thus be described more accurately as a frequency agile packet radio system. Although a higher hop rate would have resulted in better performance especially in the presence of multipath, this hop rate was chosen for three reasons :

1. A higher hop rate would have required the use of a more complex frequency synthesizer, such as a dual PLL frequency synthesizer, since the current frequency synthesizer has a switching speed of about 100 microseconds, and thus cannot switch frequencies within the current bit period of 6.4 microseconds.

2. The use of multiple hops within a frame would have significantly complicated the synchronization scheme.

3. The use of multiple hops within a frame would have reduced the data throughput because of the extra overhead required for synchronization.

3.3.5 Frequency Hop Synchronization

Synchronization is the most critical element and drives the cost and complexity of the proposed system. Time synchronization of the frequency hop receiver to the transmitted hop sequence can be separated into two phases. There is an initial acquisition phase and a tracking phase after the signal timing has been acquired.

3.3.5.1 Acquisition

Frequency hop synchronization generally involves a two dimensional search in time and frequency. The problem of initial acquisition can be viewed as one in which we attempt to synchronize in time the receiver clock to the transmitter clock. In addition, there is another timing uncertainty due to range uncertainty between the transmitter and the receiver. The classic acquisition FH techniques are the serial search technique [23] and the matched filter technique [24]. The serial search technique requires extensive use of RF circuitry such as multipliers, filters and voltage controlled oscillators. In addition, the serial search technique is not well suited for a slow FH scheme because of the time required to achieve synchronization. Although the matched filter technique is quite fast (in achieving synchronization), implementation is quite costly. Neither of these techniques are very flexible if future changes need to be made. The technique proposed for use here is known as the transmitted reference technique, and is described in detail below.

3.3.5.1.1 The Transmitted Reference Technique

In this technique, all transmissions start at one pre-assigned frequency channel designated the acquisition channel. Since all transmissions start at one pre-assigned frequency, the acquisition process reduces to a single dimensional search in time to establish timing synchronization. In every frame, the transmitter transmits a 31 bit pseudo-noise sequence that is used to establish timing synchronization at the receiver. There are two techniques to detect this synchronization sequence: matched filtering and correlation. The process of matched filtering provides a completely asynchronous method of detecting the synchronization sequence. The process of correlation against an actively generated reference code provides a synchronous method of detection. While matched filtering creates an output that is continuous in time and represents all the time relationships of the reference and the signal, correlation creates only one output sample, representing one value of the time relationship between the reference and the signal [25]. This means that the correlator would take longer to detect the synchronization sequence than the matched filter. The matched filter was chosen as the detector of choice in this design since we need to detect the synchronization sequence immediately. A software digital matched filter is used to search for this synchronization sequence, and once the sequence is detected, the receiver initializes its own hop sequence generator. Knowing the number of bits that are being transmitted and when the (incoming) transmission ends, the receiver would know when to change its frequencies, either to receive an incoming transmission or to transmit its own frame. The main feature of this technique is that synchronization is immediate and is achieved in the first frame itself, provided the synchronization sequence is detected. This technique pushes all the processing necessary (for the synchronization) to baseband, and can thus be readily implemented by the DSP processor with attendant advantages in flexibility. The digital matched filter is implemented as a 31 stage tapped delay line FIR filter, with the filter coefficients being the mirror image of the 31 bit PN sequence used for the synchronization. Figure 3.1 illustrates the process of acquisition. Two different receiver algorithms have been implemented for the transmitted reference technique, and are dicussed in detail below.

3.3.5.1.1.1 Algorithm 1

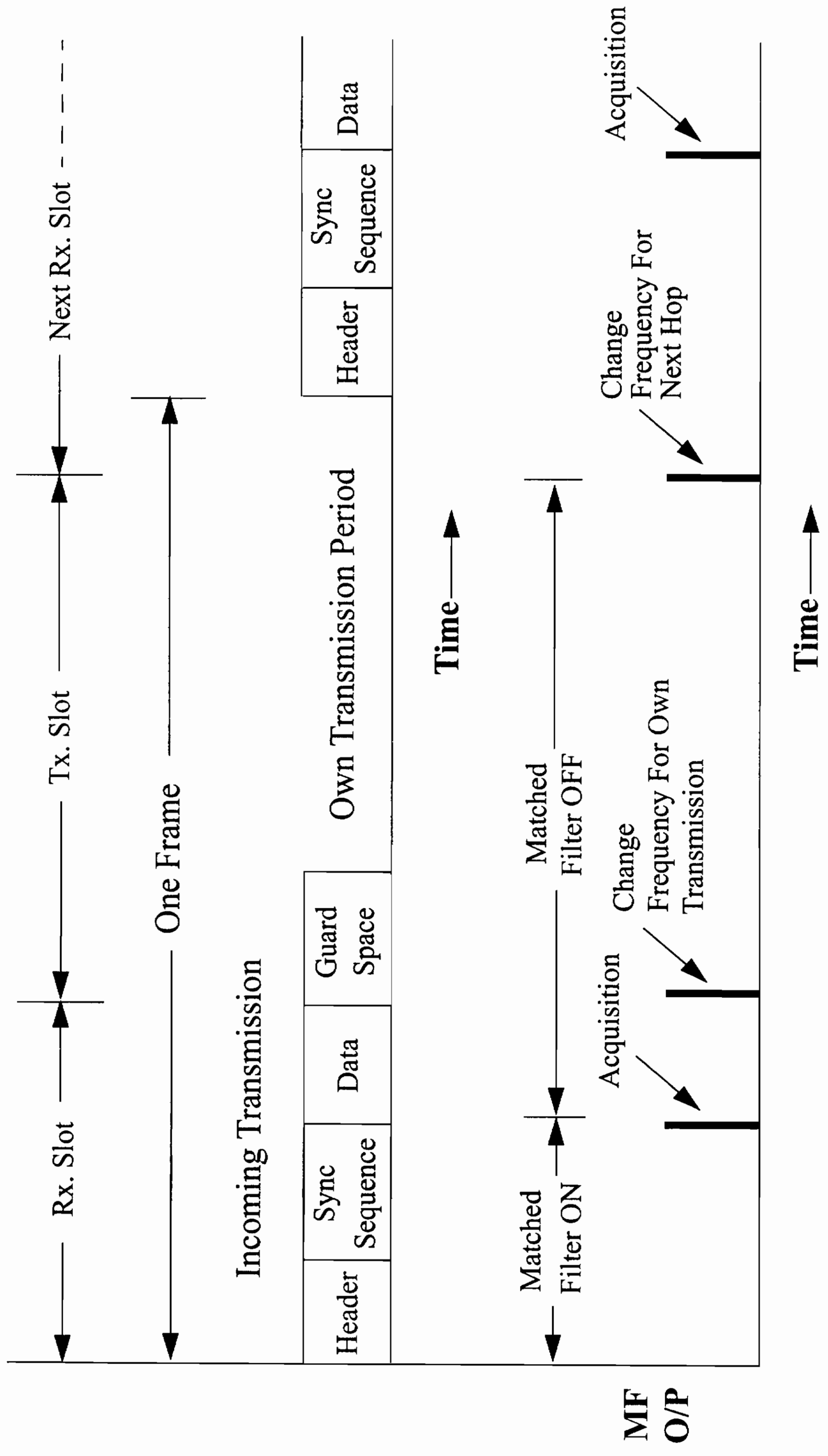


Figure 3.1 The Acquisition Process for the FH Transceiver

Algorithm 1 is the basic receiver algorithm for the transmitted reference technique. The flow chart for the algorithm is shown in Figure 3.2, and the signal flow graph is shown in Figure 3.3. Since all transmissions start at one particular frequency channel, the receiver remains camped at this particular frequency channel until it detects the 31 bit PN frame synchronization sequence. Since the hopping sequence is known apriori at the receiver, once a timing reference has been established at the receiver, the receiver maintains synchronization by tracking the number of received bits, and switching the receiver local oscillator frequency to the next receive channel at the end of the transmit frame. This algorithm requires that the synchronization sequence be detected in each frame for the algorithm to maintain synchronization. The system loses synchronization lock if the synchronization sequence is not detected.

A modified version of the above algorithm uses multiple frame synchronization sequences instead of only one as in the original algorithm. Here, the same 31 bit synchronization sequence is repeated four times in each transmit frame. The rationale behind this scheme is that atleast one of the four synchronization sequences will be detected. The signal flow graph for this technique is the same as that of the original algorithm and is shown in Figure 3.3. There are two disadvantages which make this scheme impractical. The first is that data throughput is reduced because of the increased overhead. The second reason is that there is a timing uncertainty since the receiver does not know which of the four repeated synchronization sequences has been detected.

3.3.5.1.1.2 Algorithm 2

In algorithm 1, if the synchronization sequence is not detected due to the presence of burst errors, the system will lose lock and the receiver will remain camped at that frequency channel. Synchronization can be reacquired only when the transmitted frequency channel matches the channel at which synchronization was lost. For a *N* hop pattern, this means that synchronization can be reacquired only after another *N-1* hops after the time

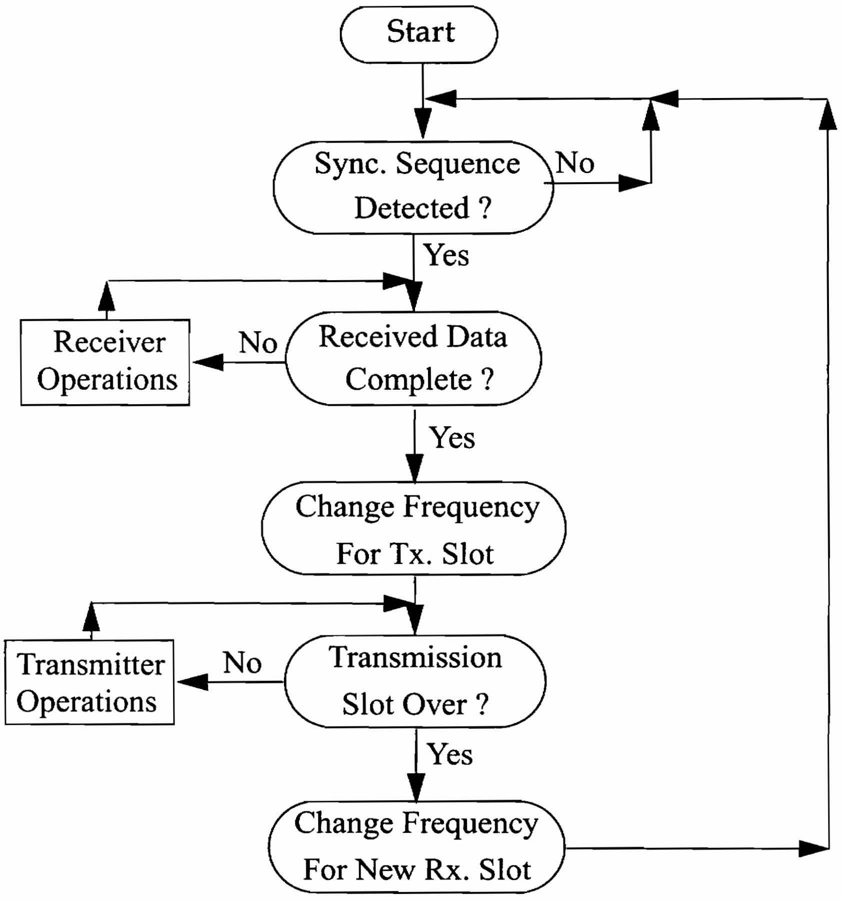


Figure 3.2 Flowchart for Acquisition Algorithm 1

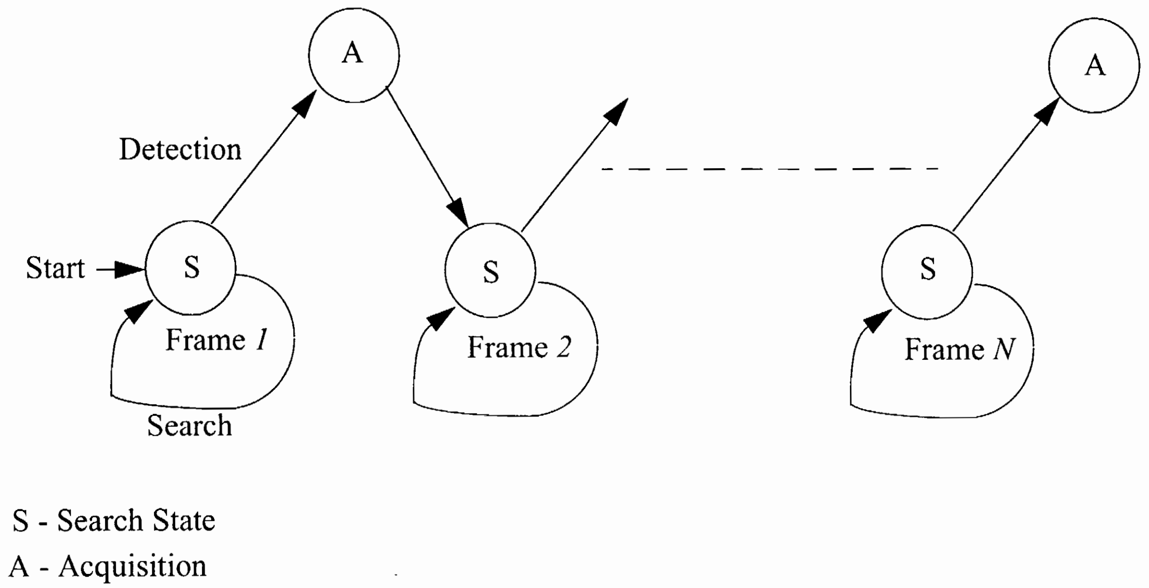


Figure 3.3 Signal Flow Graph for Acquisition Algorithm 1

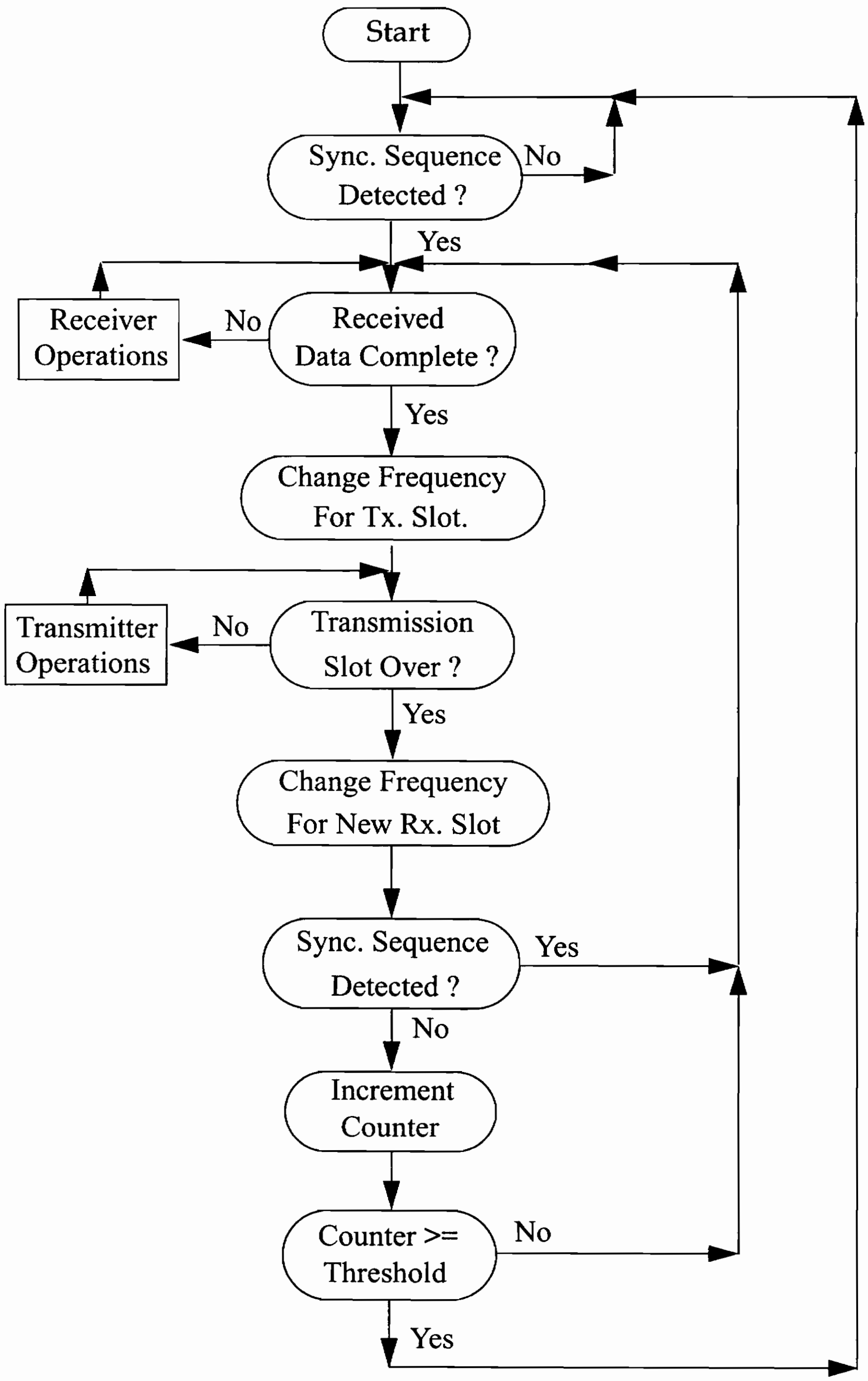
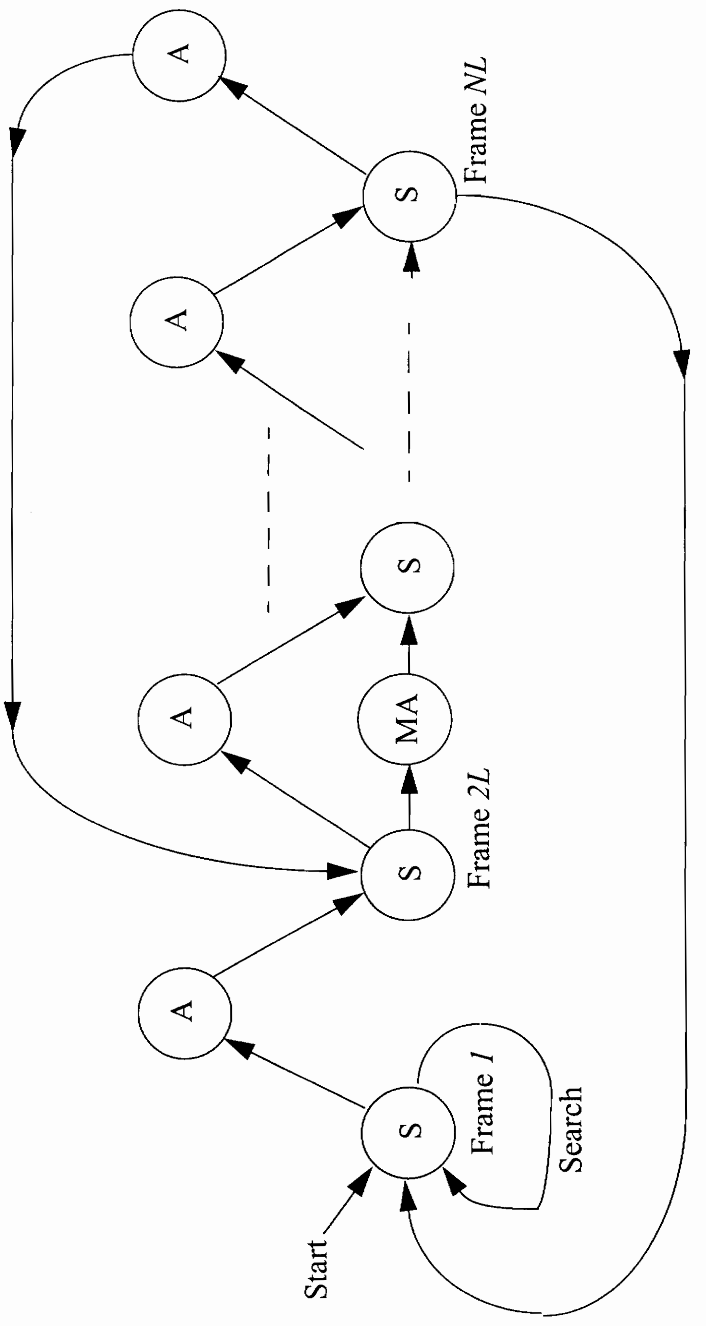


Figure 3.4 Flowchart for Acquisition Algorithm 2



S - Search State
A - Acquisition
MA - Maintain Acquisition

Figure 3.5 Signal Flow Graph for Acquisition Algorithm 2

slot during which synchronization was lost. Algorithm 2 is designed to prevent the system from losing lock even if it does not detect the synchronization sequence. This is possible since the processing is digital, and the system's past states are stored in memory. The algorithm tries to maintain synchronization using this past state information. The algorithm is able to maintain synchronization since the number of bits in the transmitted frame, the order of the hopping, and the hopping frequencies are known to the receiver. All that the algorithm requires to maintain synchronization lock is knowledge of the end of the time slots in order to switch the frequency to the next hop channel. Since the algorithm keeps track of the number of data bits received, knowledge of the end of the time slots is available to the algorithm. The algorithm thus maintains synchronization by switching frequencies according to the hopping pattern at the end of each time slot. If, after a certain number of hops, the receiver has still not detected the synchronization sequence, the algorithm decides that synchronization has been lost completely and restarts the synchronization procedure. The flowchart for this technique is shown in Figure 3.4, and the signal flow graph is shown in Figure 3.5.

3.3.5.2 Tracking

Since the number of bits in the transmitted frame, the order of the hopping, and the hopping frequencies are known apriori, and since time synchronization (acquisition) has already been established, FH tracking reduces to the task of tracking the frequency variations in the data clock. Since this task is already carried out by the DECT timing recovery subsystem, a separate FH tracking loop is not required. The timing recovery technique used is based on the 'squaring loop'.

3.3.6 Slot and Frame Format

The slot and frame format is shown in Figure 3.6. Each time slot is 9.216 milliseconds long and consists of two frames. As seen from the figure, each frame consists of

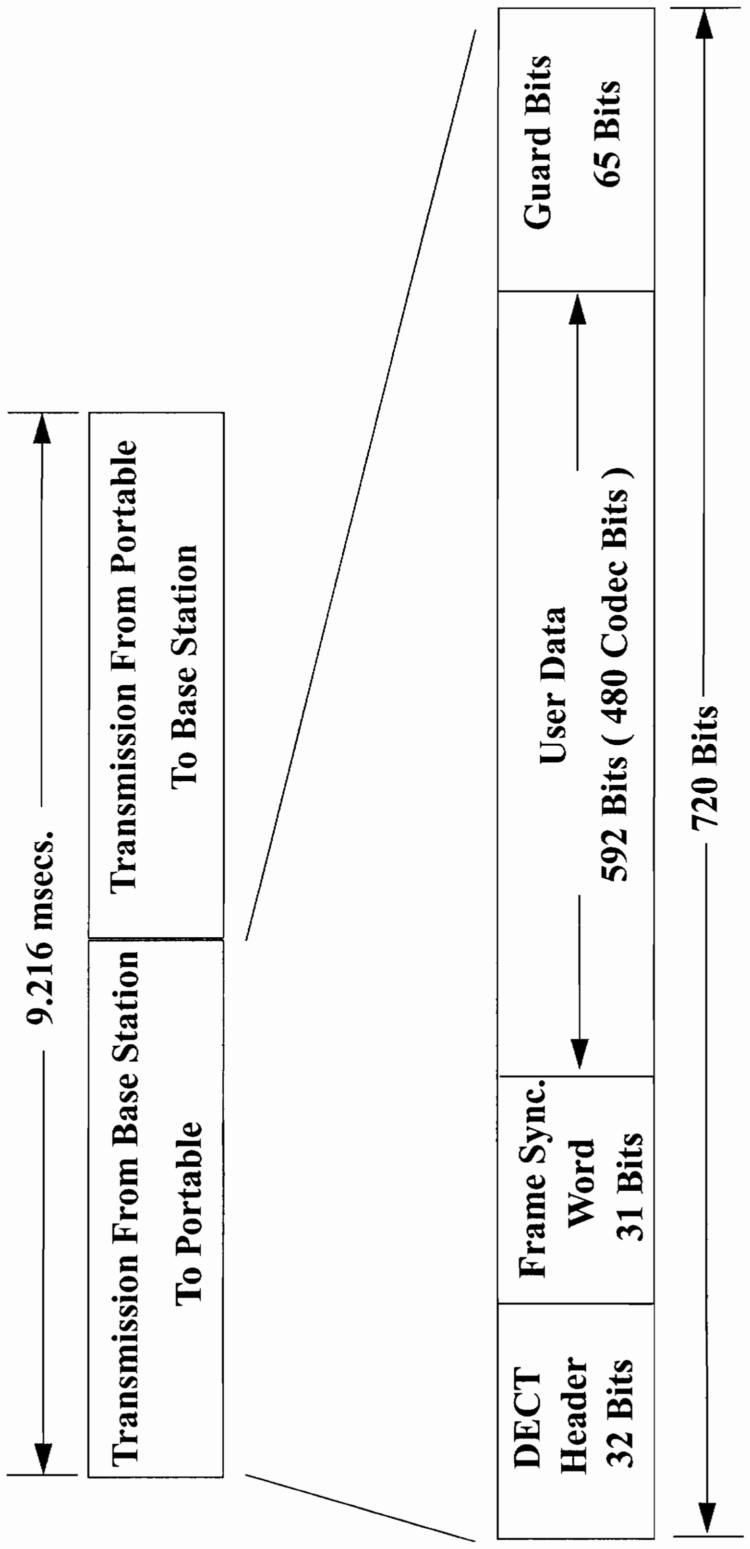

9.216 msecs.
Transmission From Base Station To Portable
Transmission From Portable To Base Station
DECT Header 32 Bits
Frame Sync. Word 31 Bits
User Data
592 Bits (480 Codec Bits)
Guard Bits
65 Bits
720 Bits

Figure 3.6 Slot and Frame Structure of the FH Transceiver

four fields:

1. Header field.
2. Synchronization field.
3. Data field.
4. Guard field.

The header field is 32 bits long and consists of the 32 bit DECT header, and is used for recovering the average DC value of the demodulated signal at the receiver. The 32 bit pattern is also used for symbol clock synchronization. The DECT header pattern is : 10101010101010101110100110001010

The synchronization field is 31 bits long and contains a 31 bit PN sequence, which is used to establish frequency hop timing synchronization at the receiver. The PN sequence used is a maximal length PN sequence, and the generator polynomial of this sixth order generator is: $1 + X + X^6$ [26]. The 31 bit PN sequence used is: 0110111010100001001011001111100.

The data field is 592 bits wide and contains the 480 CVSDM data bits. The guard field is 65 bits long and is useful in preventing ISI. It is also used to program the frequency synthesizer for the next transmit/receive frequency.

3.3.7 The Executive

The executive refers to the command and control sub-system that is responsible for scheduling and implementing the various tasks required of the FH transceiver. Since the FH architecture is time division duplexed, the command and control executive can be distinctly separated into two executive suites: transmit and receive. The system executives for both the transmit and receive segments are implemented by DSP processors. Both hardware and software interfaces are required to interface the DSP systems with the RF front end. There are several advantages to using a DSP processor to act as the system executive rather than hard wiring the executive. The main advantage is the flexibility provided to the designer and the user. Most of the sub-systems are based on signal processing algorithms that are best implemented by DSP processors which are optimized for such tasks. Finally, additional sub-systems can be added without any major impact to other sub-systems. The executive block for the transmitter is implemented by an Analog Devices AD-2111 DSP microcontroller, and that for the receiver by an Analog Devices AD-21020 microprocessor. The AD-2111 is a 16 bit fixed point processor with a computational rate of 15 million instructions per second [27]. The AD-21020 is a 32 bit floating point processor with a computational rate of 25 million instructions per second [28]. Analog Devices DSP processors were chosen because of the simplicity of the assembly language of their DSP processors. This is important in this application since the computational complexity required is quite high and handcoded assembly optimizes the computational power available to the designer. It has been widely reported in literature [29][30] that the use of a cross-compiler to generate DSP assembly from a higher level language such as '*C*' decreases the computational power available to the designer by as much as an order of magnitude. A general block diagram of the hardware interface architecture is shown in Figure 3.7.

3.3.7.1 The Transmit Executive

The flowchart for the transmit sequence of operations is shown in Figure 3.8. As soon as the calling subscriber initiates the call by pressing the 'Talk' key on the handset, an interrupt is generated to the DSP micro-controller (the executive). On reception of the interrupt, the transmit section of the DECT RF front end which is initially in the power down mode to save battery power, is 'woken up' by the DSP for transmission. Serial data bits from the Continuously Variable Slope Delta Modulator (CVSDM) codec are accumulated in a data buffer until there are enough bits to transmit in a frame. A transmit frame is

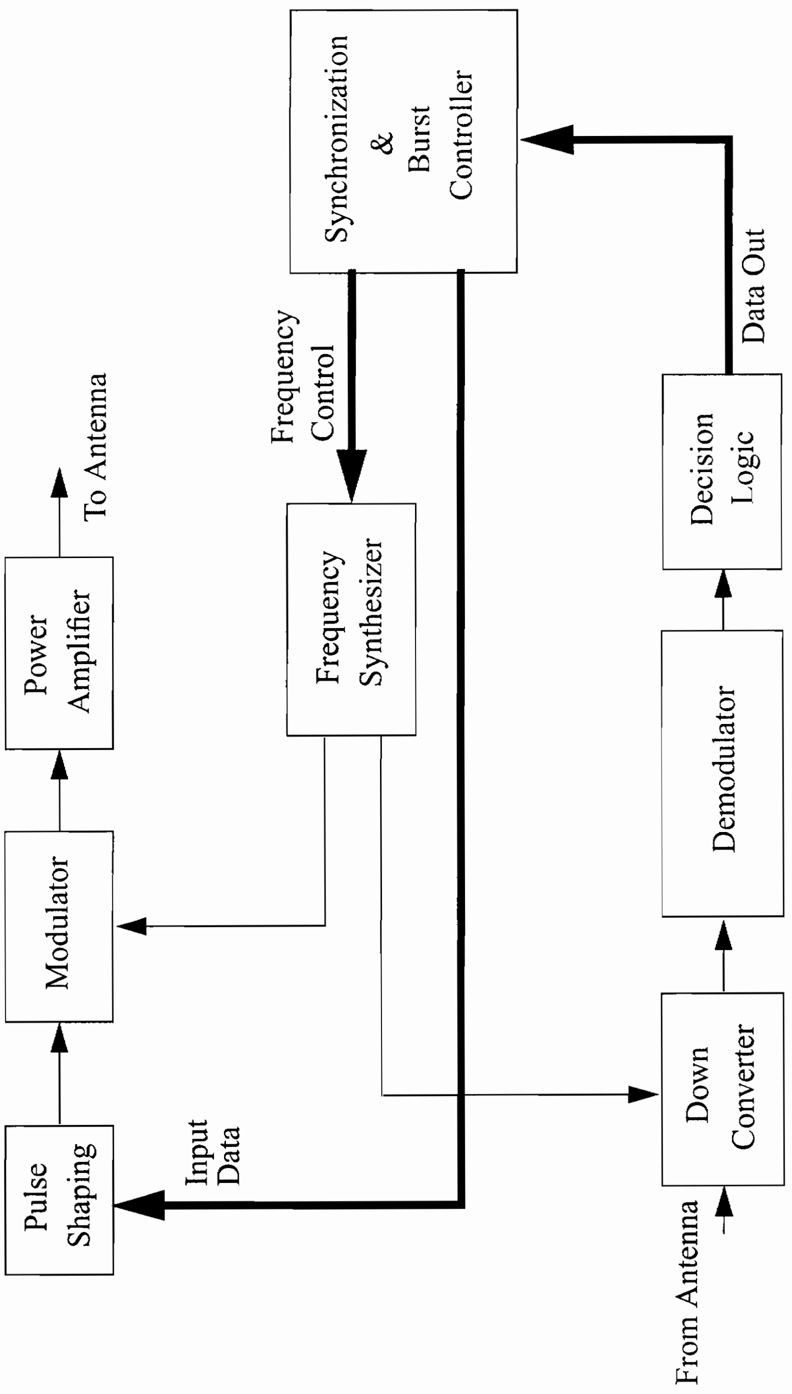


Figure 3.7 Conceptual Block Diagram of the Transceiver Hardware Architecture

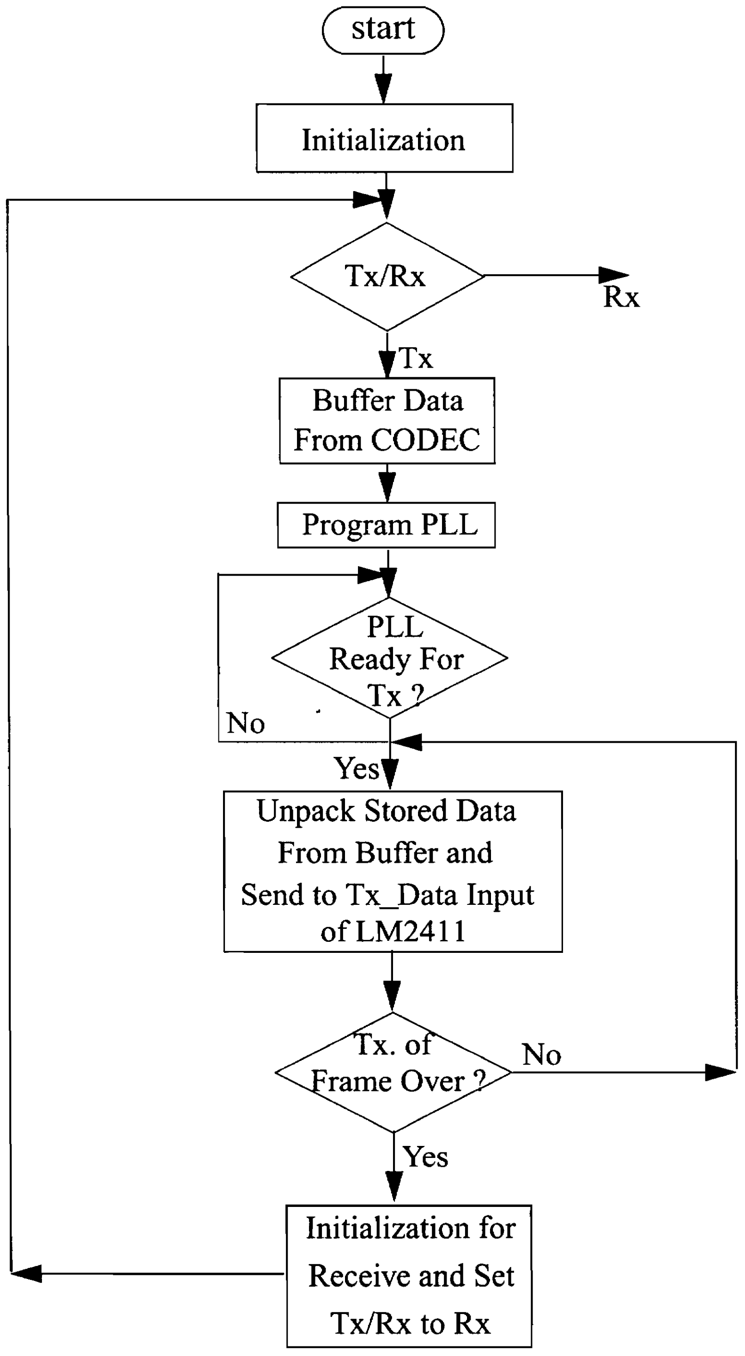

start
Initialization
Tx/Rx
Rx
Tx
Buffer Data From CODEC
Program PLL
PLL Ready For Tx ?
No
Yes
Unpack Stored Data From Buffer and Send to Tx_Data Input of LM2411
Tx. of Frame Over ?
No
Yes
Initialization for Receive and Set Tx/Rx to Rx

Figure 3.8 Flowchart for the Transmit Executive

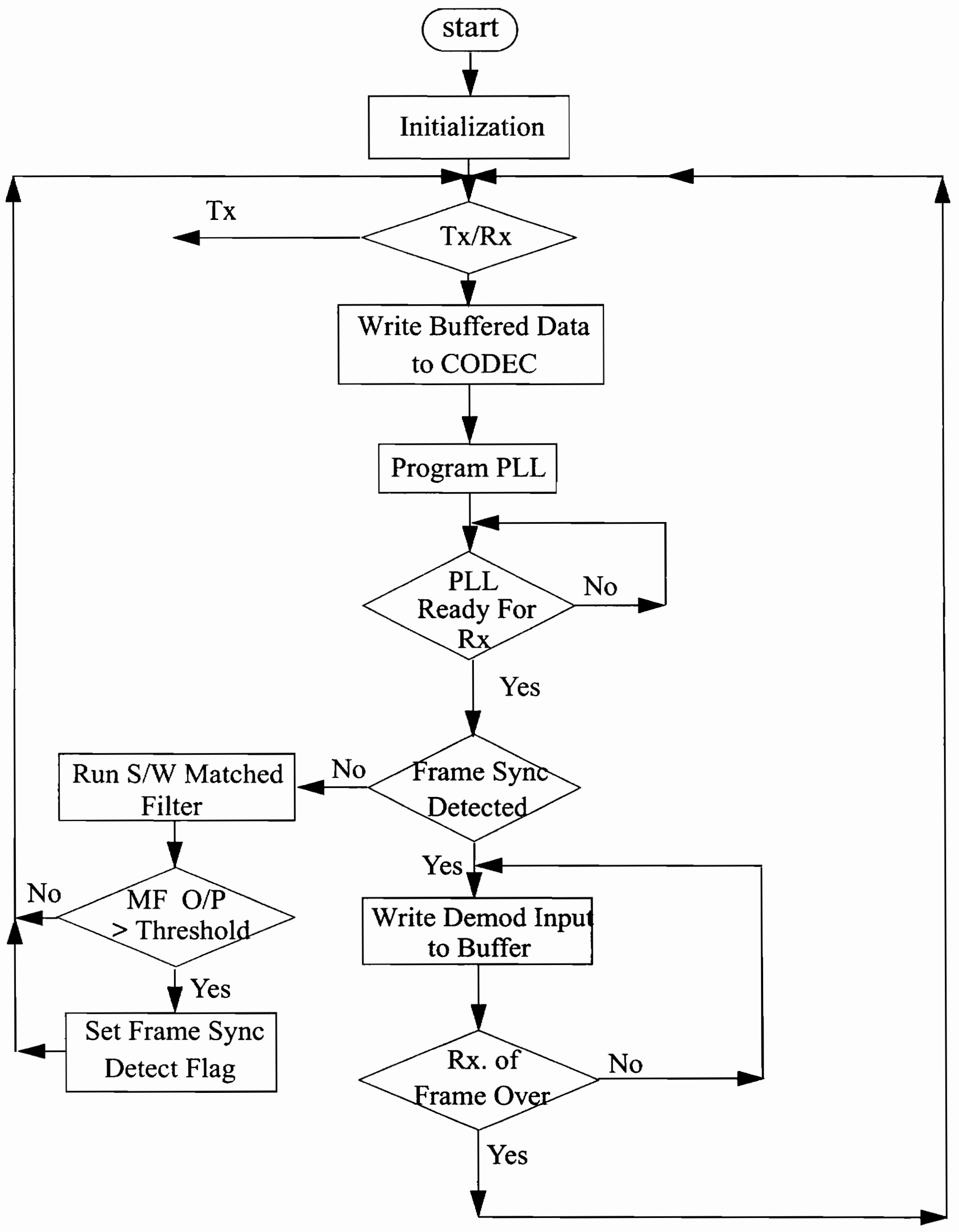


Figure 3.9 Flowchart for the Receive Executive

then constructed. The DSP then programs the PLL in the frequency synthesizer to the desired RF channel. Once the PLL has 'settled down' to its designated carrier frequency, the PLL feedback loop is opened and the data bits of the transmit frame are allowed to frequency modulate the VCO directly. Once all the bits have been transmitted, the PLL loop is closed and the DSP readies the transceiver for reception. Data bits from the CVSDM codec continue to accumulate for transmission in the next transmit frame.

3.3.7.2 The Receive Executive

The flowchart for the receive sequence of operations is shown in Figure 3.9. In the receive mode, the DSP switches the receive portion of the RF front end to active mode from the power down mode. The PLL is then programmed to the appropriate receive frequency. Once the PLL has been programmed, the antenna output is switched to the low noise amplifier/mixer circuit. The acquisition sub-block implemented by the software matched filter is also activated at the same time. The matched filter is used to search for the 31 bit synchronization sequence in the transmit frame and establish timing synchronization between the transmitter and the receiver. Once the PN sequence has been detected, the matched filter sub-block is deactivated for the rest of the receive frame, and the rest of the receiver baseband functions are activated. The recovered average value of the demodulated signal is held for the remainder of the receive frame by disconnecting the input to the hold capacitor in the LMX2240 baseband processor. The data bits from the received frame are then buffered and the receive frame deformatted. The transmitted CVSDM codec data bits are buffered and sent to the CVSDM codec for speech reconstruction. At the end of the receive frame, the frequency synthesizer is programmed to the next transmit frequency. The antenna is connected to the power amplifier output, and the input to the hold capacitor is restored. This process is repeated until the link is broken by one or both of the parties involved.

Chapter 4

SPW Models of the Frequency Hopping Transceiver

4.1 Introduction

The entire FH transceiver has been modeled and simulated in a software simulation environment called the Signal Processing WorkSystem (SPW) [31]. SPW is a powerful software package for developing, simulating, debugging, and evaluating DSP and communications systems algorithms. The SPW modeling of the FH transceiver has been split into three main segments: the transmitter, the receiver, and the system executive (controller). The details of the various blocks are discussed in detail in the following sections.

4.2 The Transmitter

4.2.1 The Data Generation Block

The data generation block is a sub-system of the controller block (discussed in detail later) and is responsible for the generation of a logical binary data bit stream. The block diagram of the data generation block is shown in Figure 4.1.

The data generator is a custom coded block, code in ‘C’. This block reads an ASCII file where the baseband data bits pertaining to a typical DECT transmission are stored as binary digits (0s and 1s). Any data file can be specified. The parameters of the block are given below:

Default Parameters:

1. File name	: ‘/home/u6/csi/caedata/dom/data.dat’.
2. Data rate	: 100 kb/s.
3. Sampling frequency	: 2.1 MHz.

Editable Parameters:

1. Data rate.
2. Sampling frequency.
3. Data file name.

4.2.2 The Level Shifter

The level shifter is a hierarchical block that has been constructed using existing SPW blocks. Two types of level shifters are used. The first type designated ‘level shifter #1’is designed to convert the binary data stream represented by 0’s and 1’s into a bi-level non-return to zero representation. Any desired signal level can be generated. The SPW model of this level shifter is shown in Figure 4.1. The second type, which is designated ‘level shifter #2’, is designed to convert a bi-level NRZ data stream into a logical 1’s and 0’s bit stream. The SPW model for this level shifter is shown in Figure 4.13. The parameters of these blocks are given below:

Default Parameters:

Level Shifter #1

1. Output amplitude	: +1 to -1 volt.

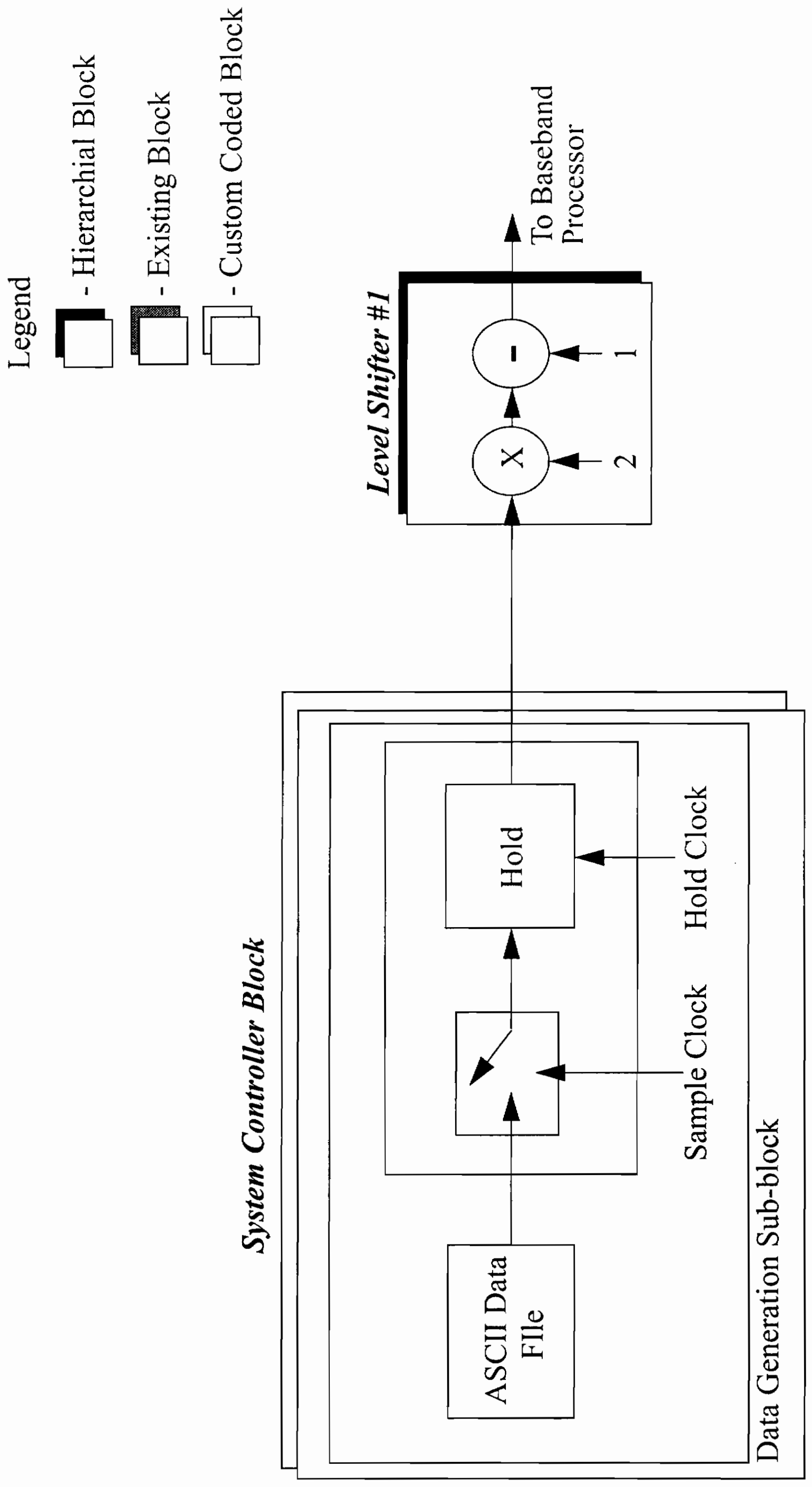


Figure 4.1 SPW Model of the Data Generation Sub-block of the System Controller

Level Shifter #2

1.Output amplitude	: +1 to 0 volt.

Editable Parameters:

1. Output amplitude.

4.2.3 The LMX 2411 Baseband Processor (Transmit Section)

The transmit section of the baseband processor implements the pulse shaping for the non-return to zero baseband data stream. This is an hierarchical block. The Gaussian filter is implemented by the complex Gaussian filter block available in the existing SPW library. The filter modeled has a 3 dB bandwidth of 50 kHz, and the tap length used is 32 (the tap length of the complex Gaussian filter available in SPW must be a power of 2; 32 is chosen as it is the closest to 21, which is the number of samples per data bit). The output of this filter is complex and only the real part is used. The Gaussian pulse shaping has been implemented as a filtering operation rather than as a ROM based readout (as actually implemented in practice), since it enables the bandwidth of the pulse shaping filter to be varied easily. The Gaussian filter output is then scaled by a suitable factor and then passed through the quantizer block to simulate the effects of finite precision representation of the Gaussian pulse samples. The quantizer output is then passed through a first order IIR butterworth filter to smooth the quantized values. The block diagram of the transmit section of the baseband processor is shown in Figure 4.2. The parameters of the block are given below:

Default Parameters:

1. 3 dB frequency	: 50 kHz.
2. Filter order	: 32.
3. Scale factor	: 0.25

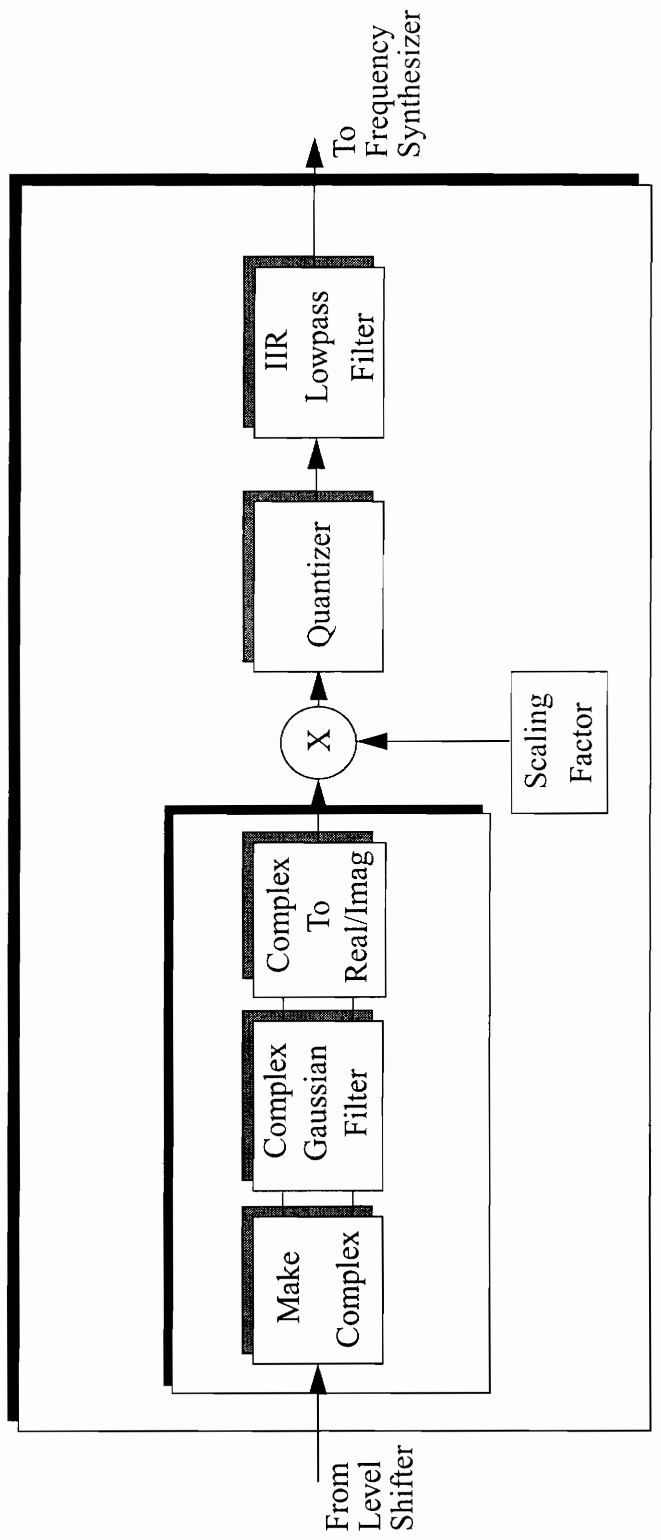


Figure 4.2 SPW Model of the Transmit Section of the LMX 2411 Baseband Processor

4. Number of bits used in the quantization	: 8.
5. Order of the smoothing IIR filter	: 1.
6. Bandwidth of the IIR filter	: 200 kHz.

Editable Parameters:

1. 3 dB frequency.
2. Order of the complex gaussian filter.
3. Scale factor.
4. Number of bits used for the quantization.
5. Order of the smoothing IIR filter.
6. Bandwidth of the IIR filter.

4.2.4 The LMX 2315 Frequency Synthesizer

Two versions of the LM2315 frequency synthesizer have been modeled. The first version models a simplified version of the frequency synthesizer. As seen in Figure 4.3, since the synthesizer is modeled as a VCO, this model of the synthesizer does not require any settling time. Frequencies can be changed instantaneously. The advantage of this block is that during bit error simulations, it is possible to reduce simulation time since there would be no time wasted in waiting for the PLL to settle down. The second version of the synthesizer models the PLL exactly as implemented. The first version of the synthesizer is used for the model of the system simulated at the intermediate frequency.

4.2.4.1 Version 1

This version of the frequency synthesizer is based on the fact that the VCO as modeled in SPW would not experience any frequency drift (unless specifically induced to do so) and thus there is no need to use a phase locked loop (locked to a crystal reference oscillator) to stabilize the center frequency of the frequency synthesizer. As seen in Figure

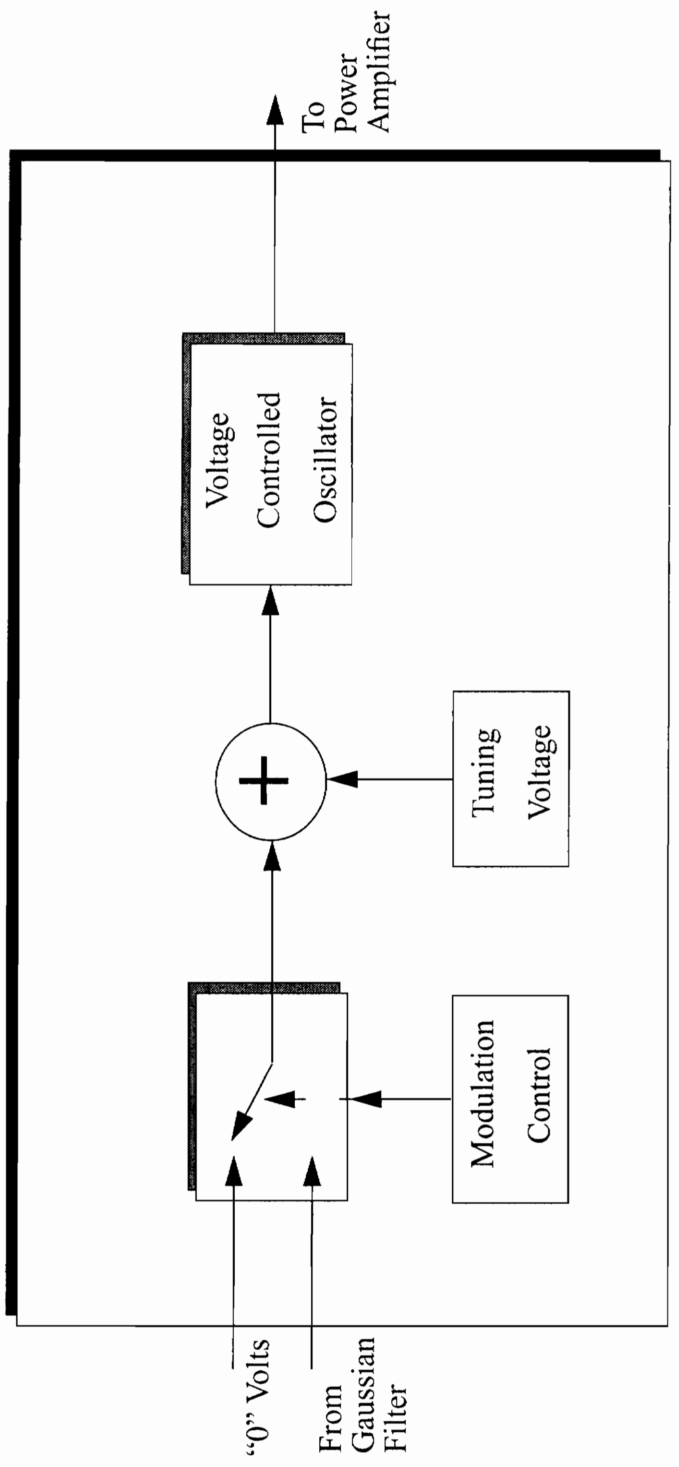


Figure 4.3 SPW Model of Version 1 of the Frequency Synthesizer

4.3, the carrier (center) frequency of the VCO is determined by the control voltage applied at the tuning voltage port of the adder. The modulating voltage is applied to the second input of the adder. The two to one multiplexer is used either to switch in the modulating voltage to allow GMSK modulation to occur or to use the frequency synthesizer as a local oscillator.

<u>Default Parameters:</u>

VCO Block:

1. Sampling frequency	: 2.1 MHz.
2. Quiescent frequency	: 0 Hz.
3. VCO gain	: 100 kHz/Volt.
4. Value of tuning voltage	: Tables 4.1 and 4.2.

Table 4.1 Tuning Voltages for the Transmitter

Frequency	Value of Tuning Voltage
350 kHz	3.5
500 kHz	5
650 kHz	6.5
800 kHz	8
950 kHz	9.5

Table 4.2 Tuning Voltage for the Receiver

Frequency	Value of Tuning Voltage
150 kHz	1.5
300 kHz	3
450 kHz	4.5
600 kHz	6
750 kHz	7.5

Editable Parameters:

VCO Block:

1. Sampling frequency.
2. Quiescent frequency.
3. VCO gain.

4.2.4.2 Version 2

This version of the frequency synthesizer models the actual synthesizer exactly. The block diagram of this version is shown in Figure 4.4. The reference oscillator used is modeled here by another VCO with the appropriate voltage applied at its control input. A VCO has been used to model the reference oscillator since it would then be possible to simulate (reference) frequency drift by varying the control voltage applied to the VCO. The use of the conventional frequency generator block would not allow the reference frequency to be varied as the simulation is run. The reference frequency used is 100 kHz. The loop filter used is modeled by a third order IIR filter with a bandwidth of 55 kHz. A one sample delay is required by SPW for feedback loops. The VCO used in the frequency synthesizer has a fixed linear transfer function of 100 kHz/Volt. The gain of the VCO transfer function can be varied by varying the parameter called Loop Gain. By varying this parameter, the settling time of the loop can be varied. The 'Hold' block in the loop is used so that the loop can be broken and the error voltage which is feedback to the VCO control input can be frozen at any time instant. The tuning voltage used to pretune the VCO, and the modulating voltage are added to the error voltage before being applied to the control input of the VCO. The frequency divider used is digital, and is modeled as such. The zero crossing detector is used to convert the sinusoidal signal produced by the VCO into a digital signal so that it can be applied to the (digital) frequency divider. The frequency divider is modeled by a bank of counters that count up to the appropriate divide ratio. The block diagram of the frequency divider is shown in Figure 4.5. The divide

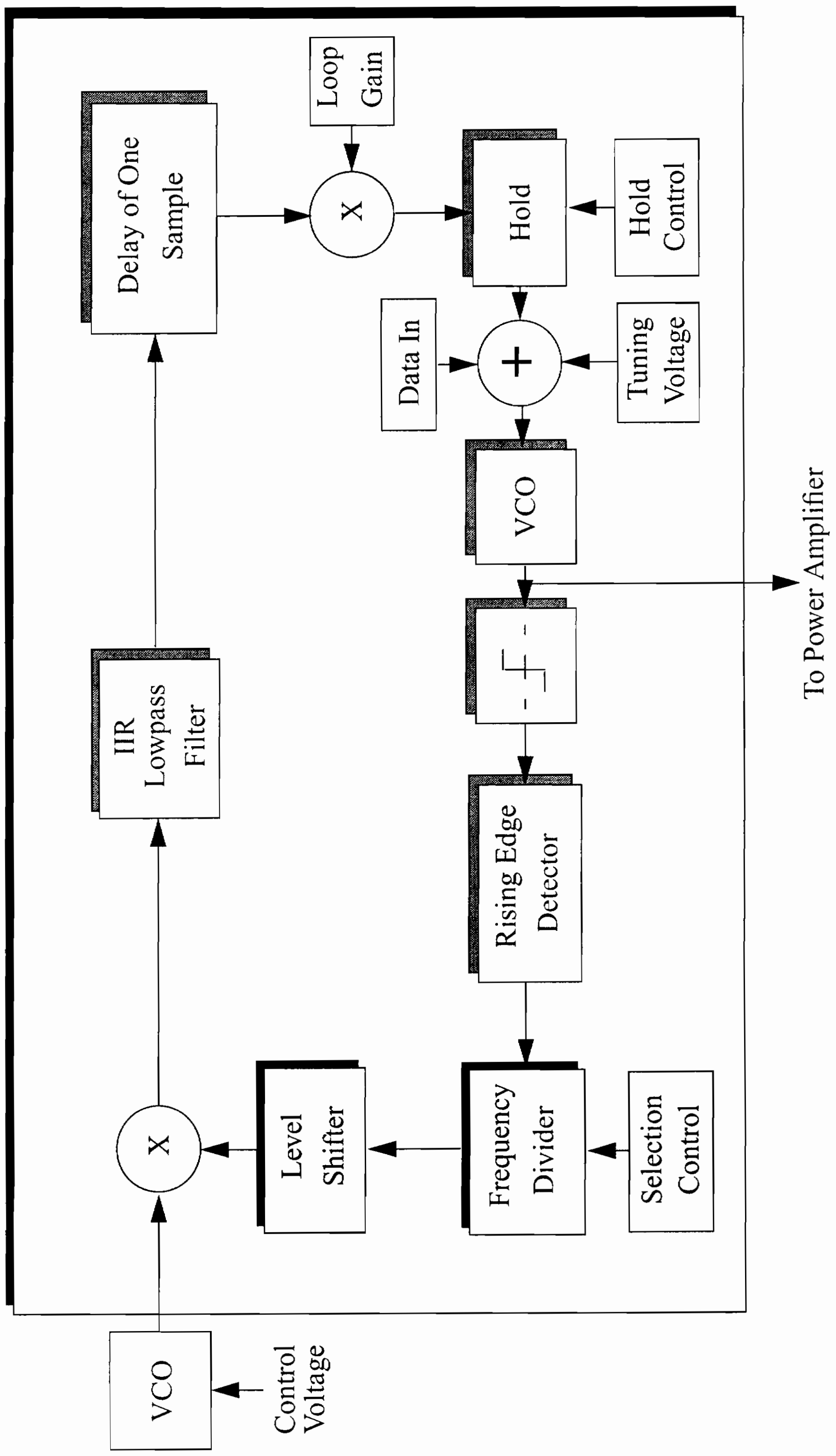


Figure 4.4 SPW Model of Version 2 of the Frequency Synthesizer

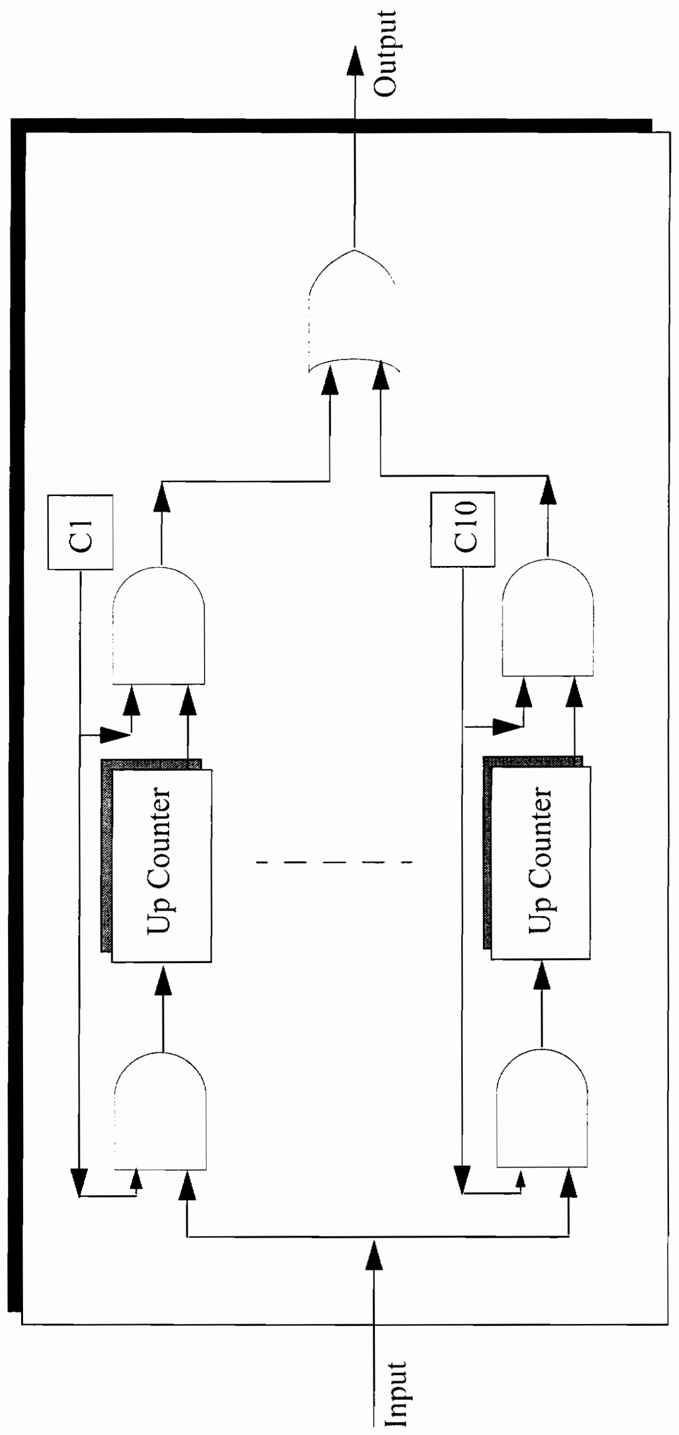


Figure 4.5 SPW Model of the Frequency Divider Used in Version 2 of the Frequency Synthesizer

ratios implemented are from 13 to 22. The selection control is used to select the appropriate divide ratio.

Default Parameters:

1. Reference frequency	: 100 kHz.
2. Order of the IIR loop filter	: 3.
3. IIR filter type	: Butterworth.
4. Bandwidth of the IIR filter	: 55 kHz.
5. Loop gain multiplier	: 0.0318287.
6. Transfer function of the VCO	: 100 kHz/Volt.
7. Quiescent frequency of the VCO	: 0 Hz.
8. Value of tuning voltage	: as in version 1.
9. Selection control	: Table 4.3.

Table 4.3 Selection Control

Division Ratio	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
13	1	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0
16	0	0	0	1	0	0	0	0	0	0
17	0	0	0	0	1	0	0	0	0	0
18	0	0	0	0	0	1	0	0	0	0
19	0	0	0	0	0	0	1	0	0	0
20	0	0	0	0	0	0	0	1	0	0
21	0	0	0	0	0	0	0	0	1	0
22	0	0	0	0	0	0	0	0	0	1

Editable Parameters:

1. Reference frequency.
2. Order of the IIR loop filter.
3. IIR filter type.
4. Bandwidth of the IIR filter.
5. Loop Gain.
6. Transfer function of the VCO.
7. Quiescent frequency of the VCO.

4.3 The Receiver

The receiver chain for this version of the system consists of the LNA/Mixer, IF filter, IF limiter discriminator, lowpass baseband filter, receive section of the baseband processor and the symbol timing recovery block.

4.3.1 The LNA/Mixer

The LNA/Mixer is modeled by a simple multiplier. Figure 4.6 shows the SPW model for this block.

4.3.2 The IF Filter

The IF filter needs to be a linear phase filter to avoid signal distortion . As such, the IF filter is modeled as a Finite Impulse Response (FIR) filter. The FIR filter is a linear phase filter, with a constant group delay across the band of interest. The magnitude response of the FIR filter is almost exactly equal to that of the actual implemented filter. This block has been implemented as a general purpose filter block in SPW, and it requires the filter coefficients to be input in a separate data file. The filter coefficients have been generated using a custom MATLAB program. This MATLAB program can be used to

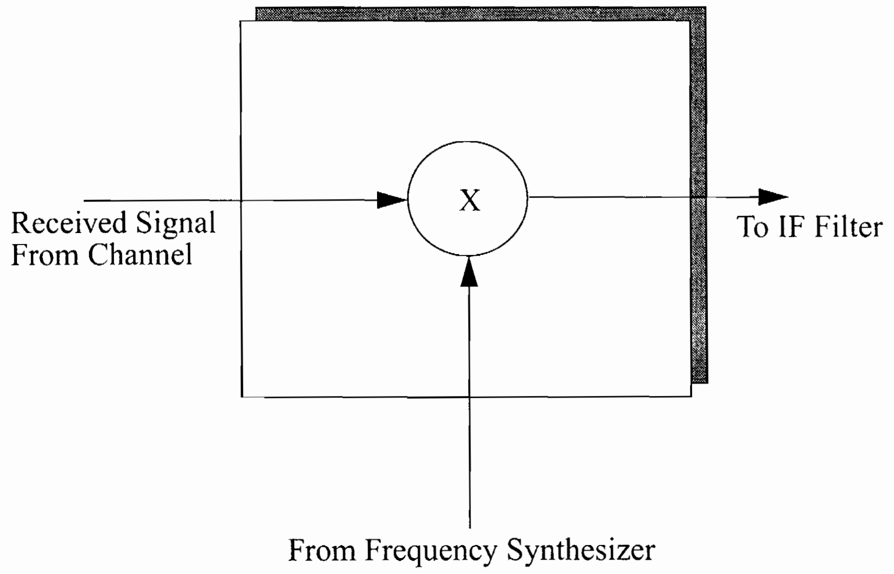


Figure 4.6 SPW Model of theLMX 2216 LNA/Mixer

design any bandpass FIR filter. SPW's filter design system (FDS) can also be used to design the filter. The magnitude and phase response of the IF FIR filter are shown in Figures 4.7 and 4.8.

Default Parameters:

1. Sampling frequency	: 2.1 MHz.
2. Center frequency	: 200 kHz.
3. Bandwidth	: 150 kHz.
4. Order of filter	: 20 (21 coefficients).
5. Data file name	: '/home/u6/csi/caedata/dom/if_filter_ber.asc-filt'.

Editable Parameters:

1. Data file name.
2. Sampling frequency.
3. Center frequency.
4. Bandwidth.
5. Filter order.

4.3.3 The LMX 2240 IF Limiter Discriminator

This is a hierarchical block. The block diagram of the LM2240 is shown in Figure 4.9. The limiter as modeled saturates at +1 and -1 volts. The frequency discriminator used is a quadrature detector. The frequency discriminator effectively uses an analog mixer as a phase detector and translates instantaneous frequencies (rates of phase changes) to different voltage levels. To achieve optimum performance from the discriminator, a $180°$ phase shift should be introduced across the band of interest. Ideally, the phase shift achieved should be $90° \pm 90°$. The quality factor of the tank used to implement the frequency dependent phase shift should be as high as possible to keep the phase linear across

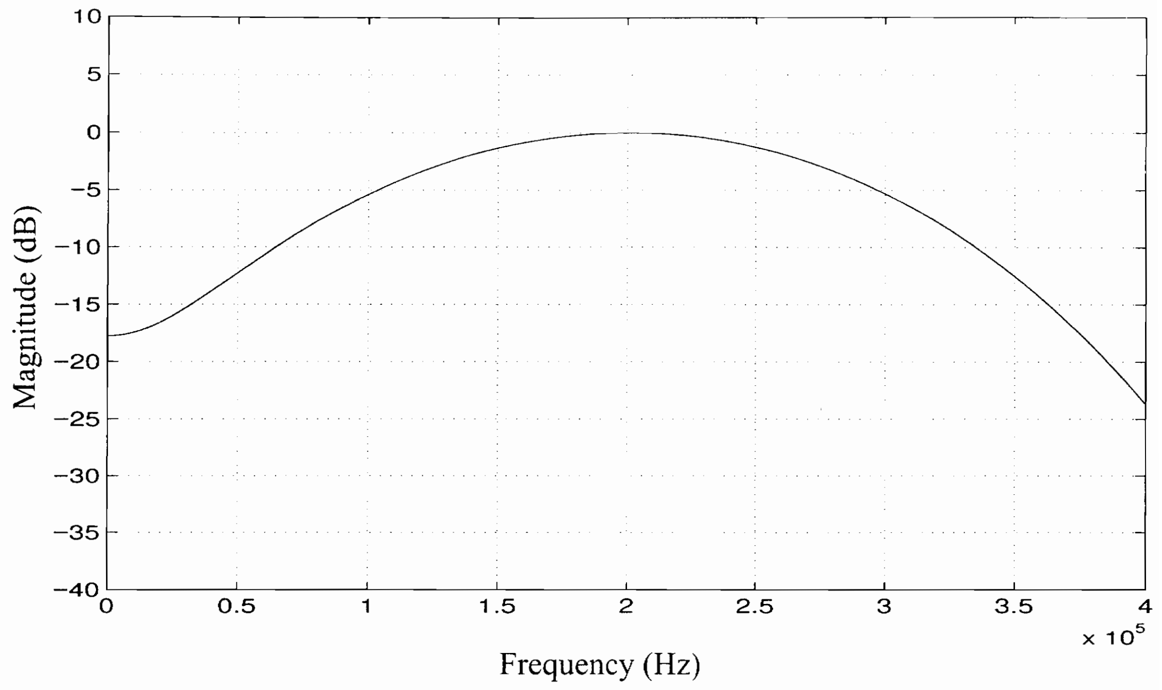


Figure 4.7 Magnitude Response of the IF FIR Band Pass Filter

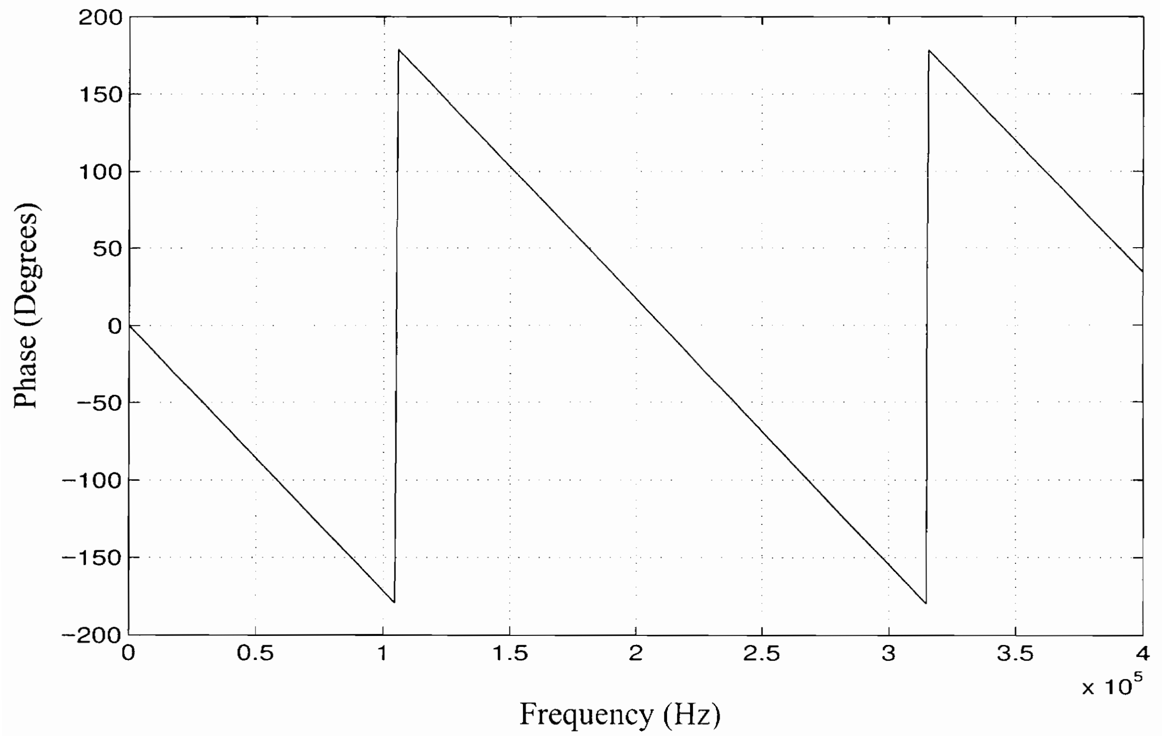


Figure 4.8 Phase Response of the IF FIR Band Pass Filter

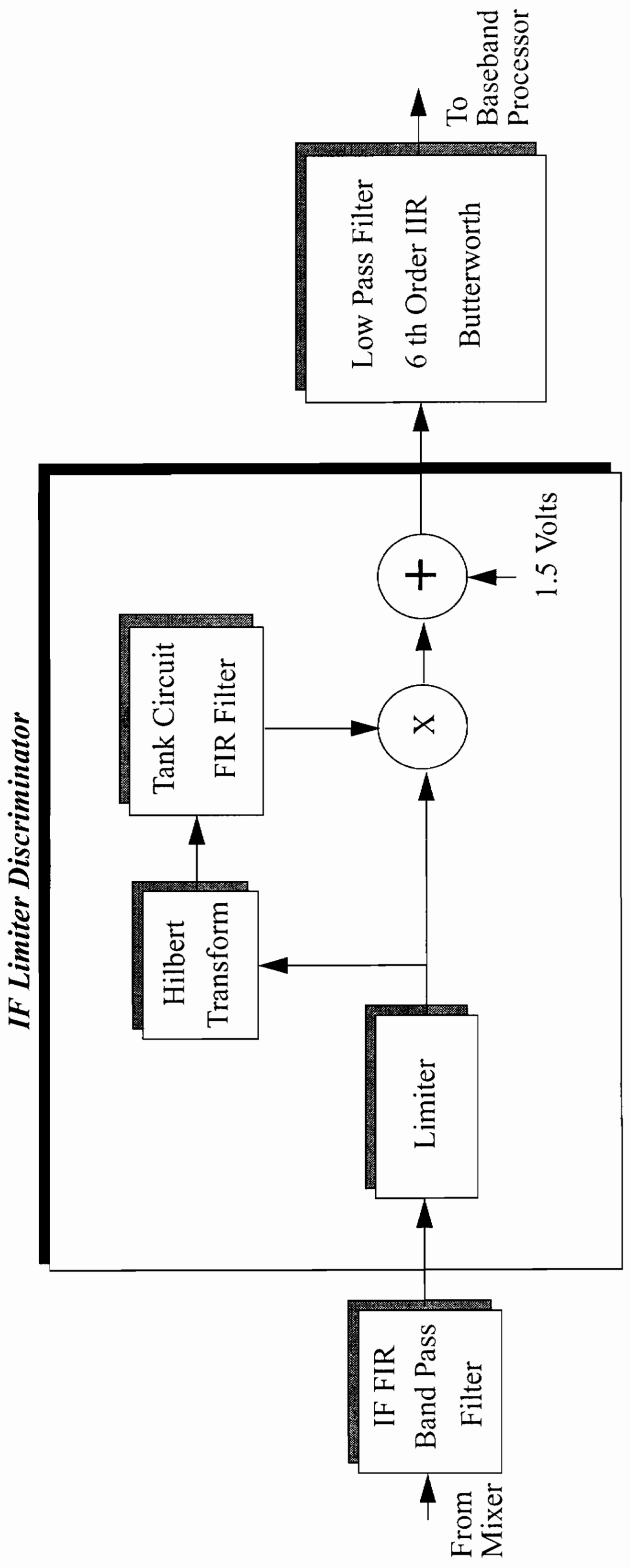


Figure 4.9 SPW Model of theLMX 2240 IF Limiter Discriminator

the band of interest. The tank circuit used has a Q of 70. The tuned circuit is modeled in SPW as a Hilbert Transformer in conjunction with a FIR filter. The Hilbert Transformer is used to introduce the nominal 90° phase shift as required, and it exists as a SPW block. The bandpass filter used to achieve the frequency dependent phase shift for quadrature detection is modeled by a FIR filter of suitable order designed to give a phase shift of about 50° (which is realizable with practical circuits) across a frequency span of 150 KHz. A FIR filter has been used since we require that the filter be linear phase. The FIR filter is implemented by the general filter block in SPW, which can model both FIR and IIR filters. This block requires that the filter coefficients be given in a separate data file. The filter is designed in MATLAB using the window technique, implemented by the MATLAB function FIR1 [32]. The magnitude and phase responses of the tank circuit are shown in Figures 4.10 and 4.11. The output of the discriminator is centered at 1.5 volts.

Default Parameters:

1. Order of the Hilbert Transformer	: 32.
2. Order of the bandpass FIR filter	: 10 (11 coefficients).
3. Center frequency of the FIR filter	: 200 kHz.
4. Bandwidth of the FIR filter	: 150 kHz.
5. File name of the coefficient file	: '/home/u6/csi/caedata/dom/tank_ber.ascfilt'.

Editable Parameters:

1. Order of the Hilbert Transformer.
2. Order of the FIR filter.
3. Center frequency of the FIR filter.
4. Bandwidth of the FIR filter.

4.3.4 The Lowpass Baseband Filter

The lowpass filter used in the actual implementation is a sixth order butterworth fil-

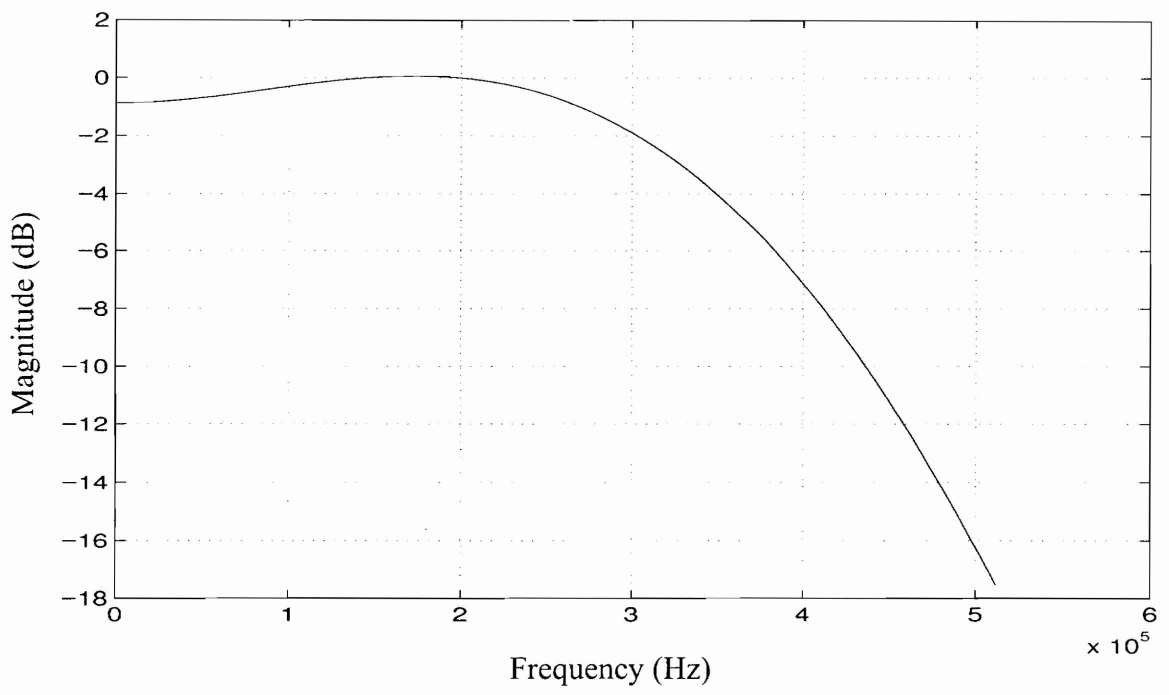


Figure 4.10 Magnitude Response of the Tank Circuit

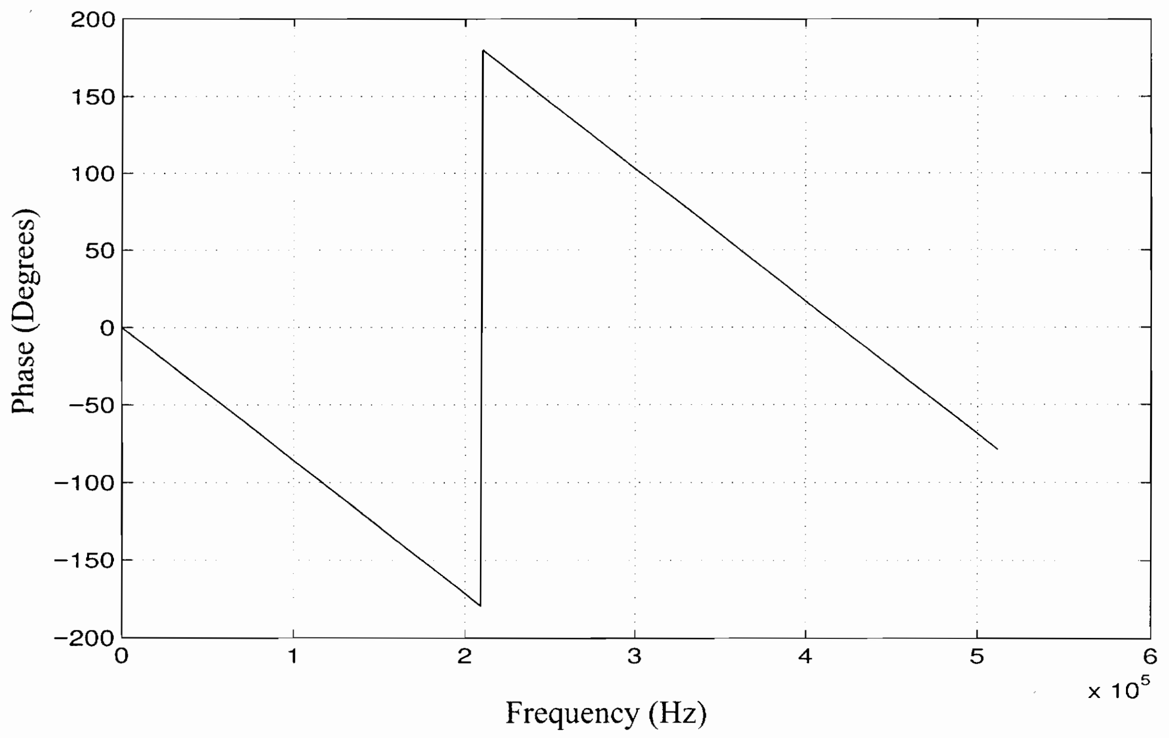


Figure 4.11 Phase Response of the Tank Circuit

ter and is modeled in SPW as such by a sixth order butterworth Infinite Impulse Response (IIR) filter.

Default Parameters:

1. Sampling frequency	: 2.1 MHz.
2. 3 dB frequency	: 100 kHz.
3. Filter order	: 4.

Editable Parameters:

1. Sampling frequency.
2. 3 dB frequency.
3. Filter order.
4. Filter type.

4.3.5 The LMX 2411 Baseband Processor (Receive Section)

The objective of the baseband processor is to recover the DC value from the lowpass filtered demodulated signal, and to use this value to shape the analog lowpass signal into a non-return to zero digital signal. The DC value is obtained by averaging over the first 16 bits of the header sequence. The first order integrator used to average the DC value is modeled by a first order butterworth IIR filter. The block diagram of the baseband processor is shown in Figure 4.12. The control signals required for the various operations are summarized in Table 4.4.

Default Parameters:

1. Sampling frequency	: 2.1 MHz.
2. Filter order	: 1.
3. 3 dB frequency	: 4 kHz.

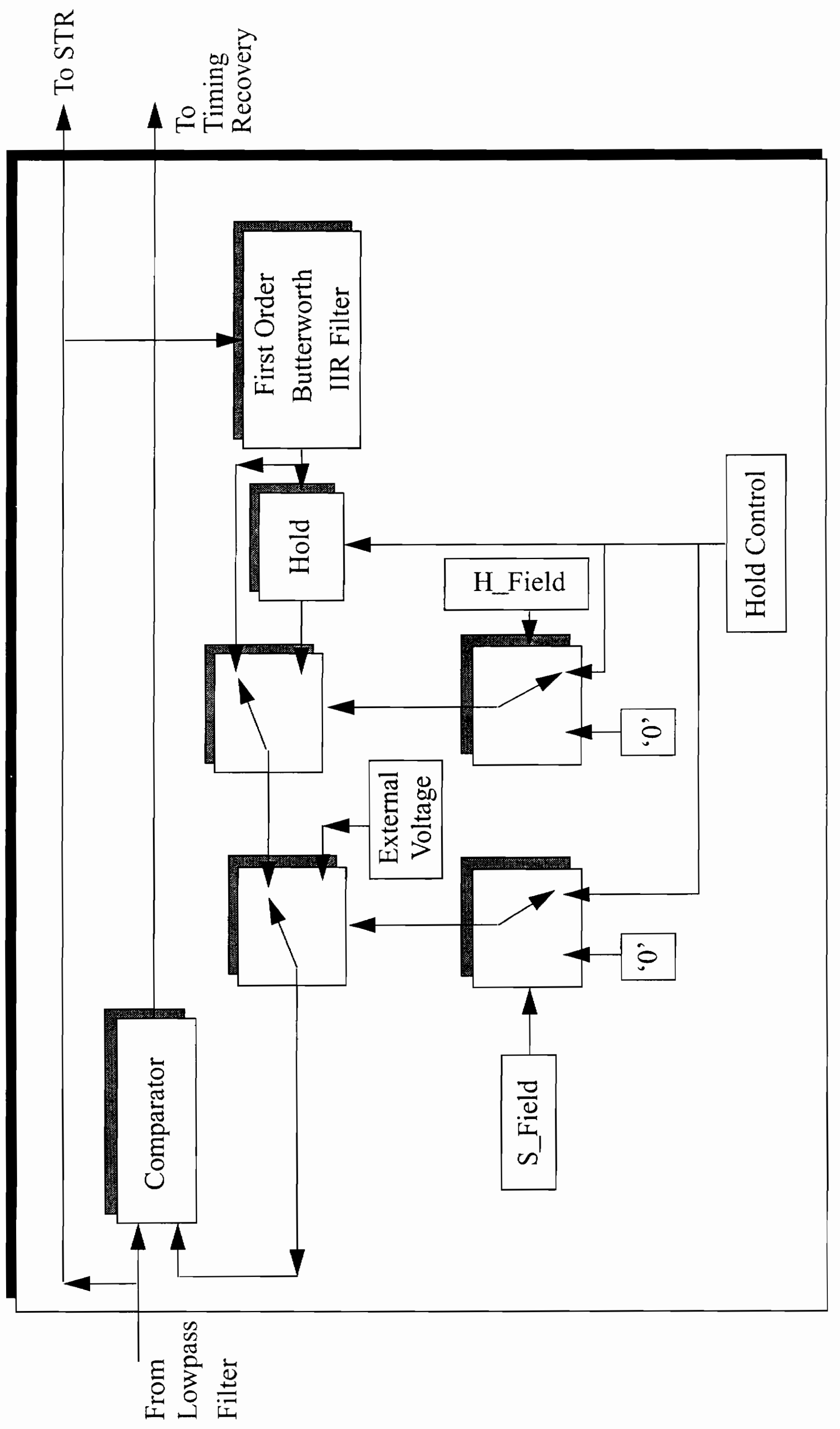


Figure 4.12 SPW Model of the Receive Section of the LMX 2411 Baseband Processor

Table 4.4 Control Signals Required for the Receive Section of theLMX 2411 Baseband Processor

FUNCTION	H_Field	S_Field	Control	Ext_Voltage
External threshold is used	X	0	X	V_{EXT}
Internal compensation with hold after 'x' bits of preamble	1	1	Connect to output of Timing block	X
Internal compensation with continuous update	0	1	1	X

<u>Editable Parameters:</u>

1. Sampling frequency.
2. Filter order.
3. 3 dB frequency.

4.3.6 The Symbol Timing Recovery Block

The symbol timing recovery technique modeled is the squaring loop technique. Figure 4.13 shows the block diagram of this block. The input signal is delayed by half the data bit period, using the bulk delay block in SPW. The delayed signal and the original (undelayed) signal are then multiplied together using an ex-or gate to generate a discrete spectral line at the bit rate. The output of the ex-or gate is then appropriately level shifted to transition from the logic levels required for the ex-or operation to a polar NRZ signal. The spectrum at the output of the multiplier consists of a discrete spectral line at multiples of the bit rate, a continuous component, and a DC component. The level shifted signal is then coarsely filtered using a fourth order butterworth IIR digital bandpass filter centered at 100 kHz to eliminate the DC component and lightly filter the continuous component. The discrete spectral component at the data rate passes unchanged through the bandpass filter and is then tracked by a second order PLL. The lock range of the PLL is 10 kHz. In steady state, the PLL will track the phase of the discrete spectral component at the bit rate. The output of the PLL, which is sinusoidal, is then sent to a zero crossing detector to shape the recovered clock into a digital signal. This is a hierarchical block.

We can also make use of the analog version of the squaring loop technique, where instead of using the delay and ex-or method, we would make use of an absolute value circuit to generate the spectral component at the bit rate. In this case, the symbol timing recovery block input would be taken from the port labeled 'To STR', and not from the output of the comparator in the baseband processor as in the digital implementation.

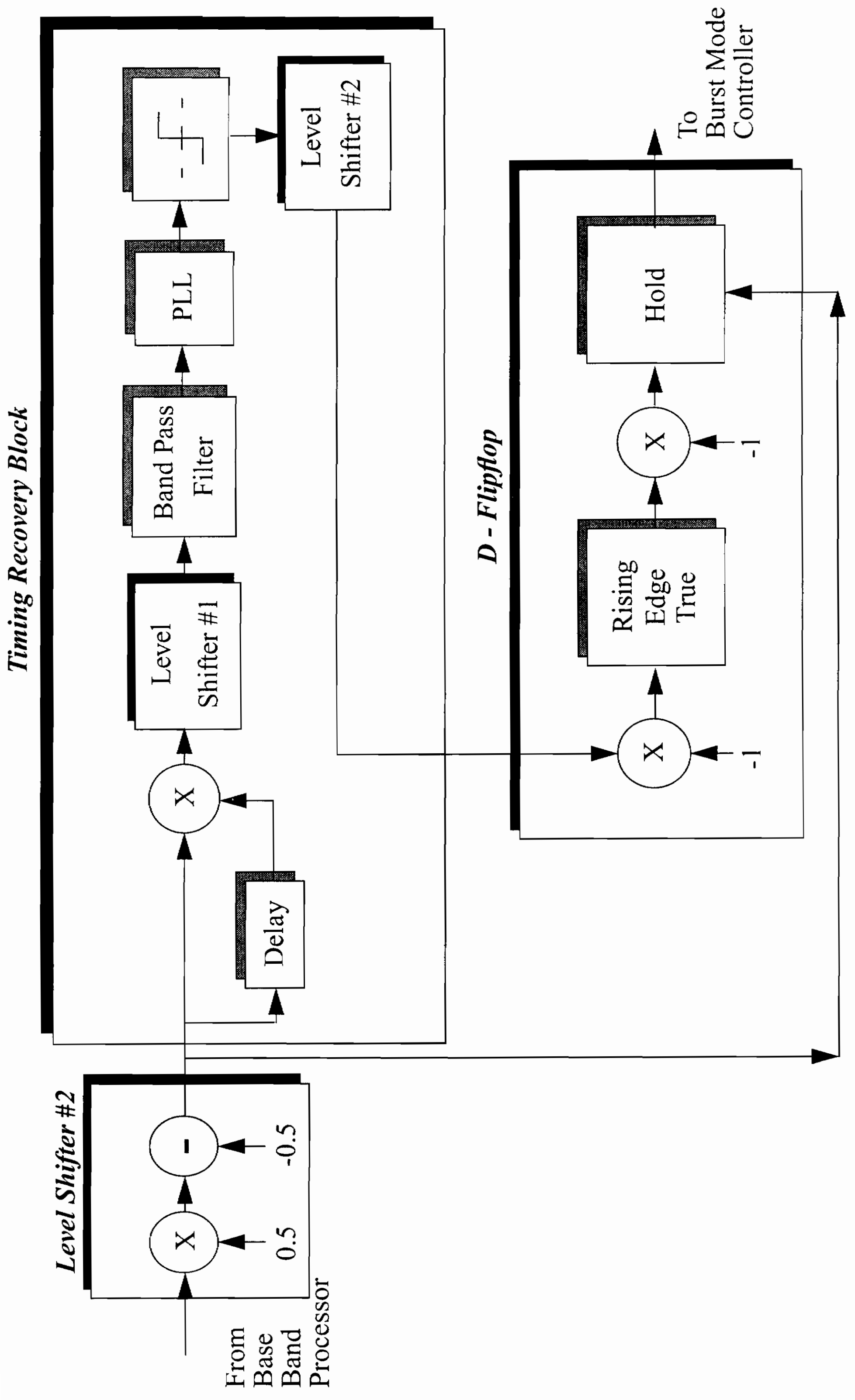


Figure 4.13 SPW Model of the Timing Recovery Block

Default Parameters:

1. Sampling frequency	: 2.1 MHz.
2. Amount of delay	: 10 samples.
3. Filter order of BPF	: 4.
4. Center frequency of BPF	: 100 kHz.
5. 3 dB bandwidth	: 10.0 kHz.
6. PLL center frequency	: 100 kHz.
7. Lock range of PLL	: 10 kHz.

Editable Parameters:

1. Sampling frequency.
2. Amount of delay.
3. Filter order of BPF.
4. Center frequency of BPF.
5. 3 dB bandwidth.
6. PLL center frequency.
7. Lock range of PLL.
8. level shifts.

4.3.7 The D Flip-flop

The *D* flip-flop is used to resample the incoming recovered data stream from the baseband processor using the data clock recovered by the symbol timing recovery block. The *D* flip-flop samples the input data and it is rising edge active. This is a hierarchical block.

Default Parameters:

None.

Editable Parameters:

None.

4.3.8 The Bit Error Rate Block

The bit error rate (BER) block is a sub-system of the controller block and is used to determine the data bit error rate for a simulation. This block has been custom coded in 'C'. This block reads the original (transmitted) data directly from the input data file and compares it with the demodulated data to determine the error rate. The bit error rate is then written to an output file. The functional block diagram of this block is shown in Figure 4.14.

Default Parameters:

1. Number of data bits for the simulation	:10000.
2. Input file name	: '/home/u6/csi/caedata/dom/data.dat'.
3. Output file name	: '/home/u6/csi/caedata/dom/BER.dat'.

Editable Parameters:

1. Number of data bits for the simulation.
2. Input file name.
3. Output file name.

4.4 The Controller

The controller block is the system executive, and is responsible for scheduling, synchronizing and carrying out the various housekeeping tasks required to operate the transceiver. The controller performs all the functions as outlined in section 3.3.7 in chapter 3. The controller block can be divided into two main blocks: transmit and receive. This is a custom coded block, coded in 'c'. A general block diagram of the SPW model of this

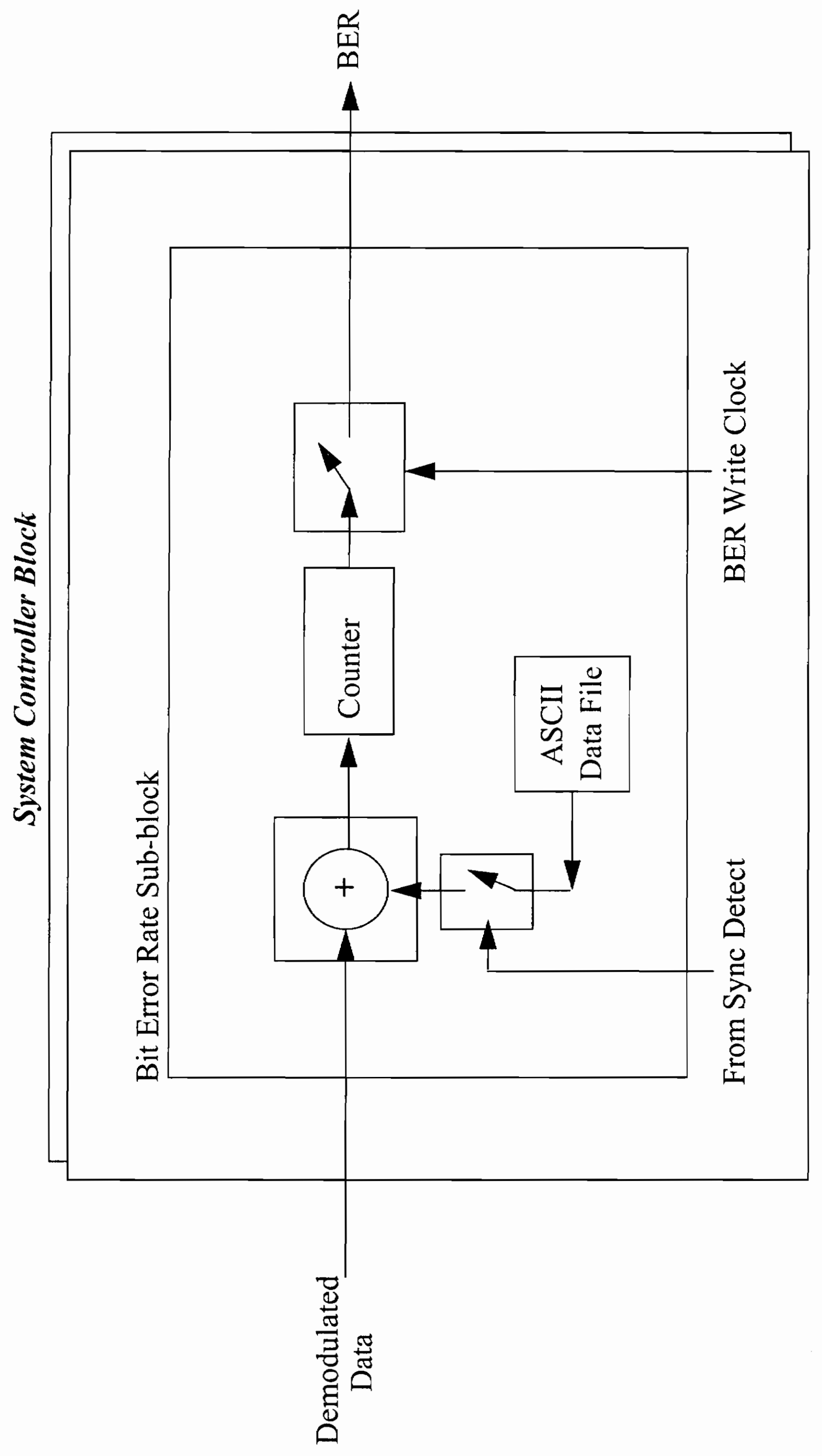


Figure 4.14 SPW Model of the BER Sub-block of the System Controller

block is shown in Figure 4.15.

4.4.1 The Transmit Block

The operation of this block is in accordance with the flowchart shown in Figure 3.8. The transmit block can be further divided into three main sub-blocks: the data generation sub-block, the transmit frequency control sub-block and the antenna switch control sub-block.

The data generation sub-block is responsible for the generation of the transmit frame. There is a difference in the way the data generation is simulated. To reduce the simulation time, data is assumed to be available in an ascii file, and the data generation sub-block thus simply reads out the binary bits stored in the data files. An ASCII text file is created for the data file and binary data in the form of 1s and 0s is entered without any carriage returns. Generation of the samples is in accordance with the desired sampling rate. The transmit frequency control sub-block is responsible for generating the proper control signals required to program the frequency synthesizer for the different transmit carrier frequencies. At the beginning of each frame, the frequency control sub-block outputs a control voltage to the tuning voltage port of the frequency synthesizer. The control voltages required depends upon the quiescent frequency of the frequency synthesizer VCO, and the transfer gain of the VCO. The control voltage values used in the model are given in Table 4.1. If version 1 of the frequency synthesizer is used, then only the control voltage inputs are required for operation. If version 2 is used, two extra control signals are required. The first is the 'hold' signal, which is used to open the feedback loop of the PLL in the frequency synthesizer to allow FM modulation to occur. The second is the 'selection control' signal required to select the proper frequency division ratio in the feedback loop of the PLL frequency synthesizer. The required selection control values are as shown in Table 4.2. Both the signals are generated by the transmit frequency control sub-block. The antenna switch control sub-block is responsible for the proper routing of the transmit

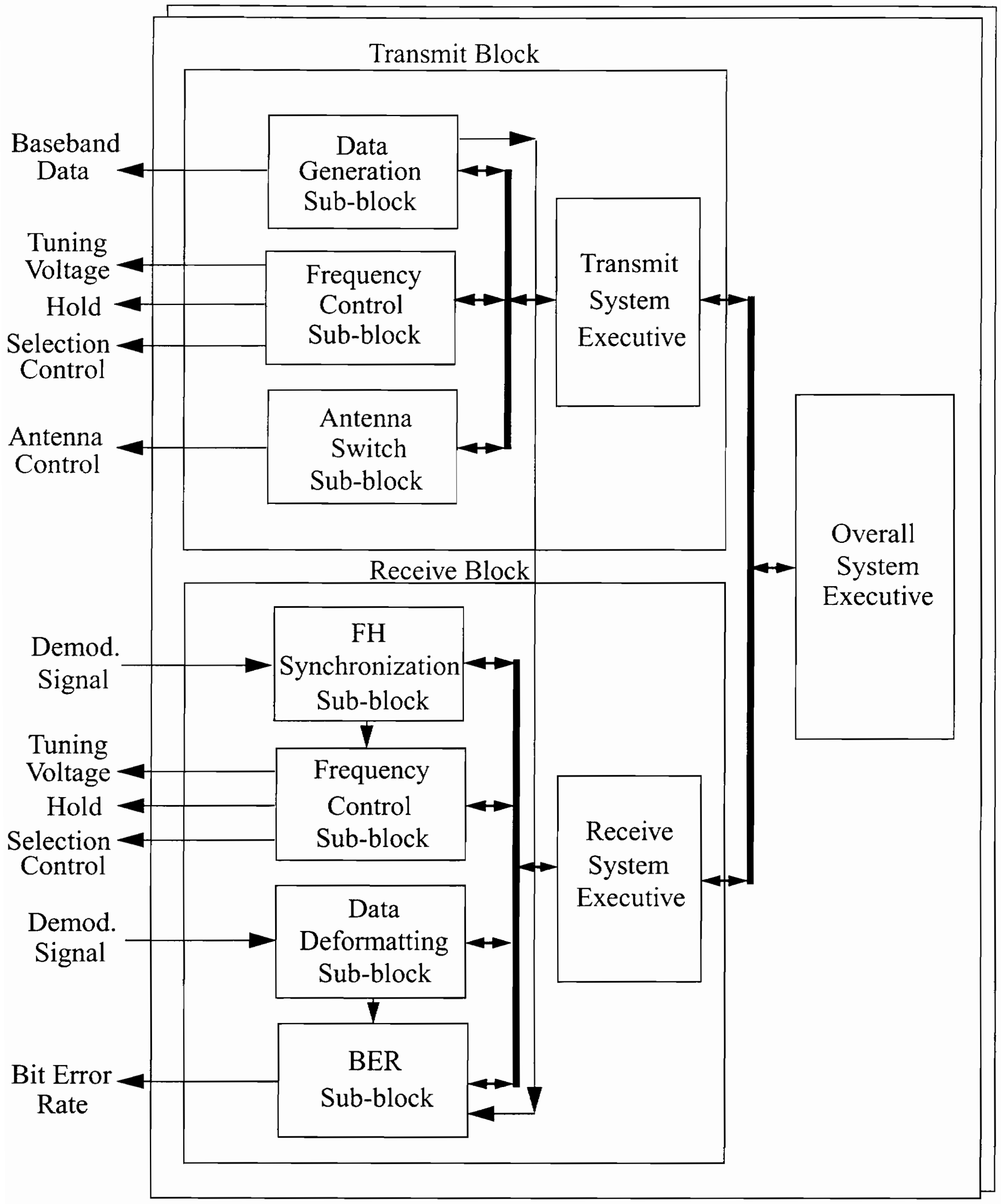


Figure 4.15 SPW Model of the System Controller Block

and receive signals to and from the antenna.

Default Parameters:

1. Sampling frequency	: 2.1 MHz.
2. Bit rate	: 100 kbps.
3. Frame count	: 100.
4. Frame counter	: 1.
5. Number of hop frequencies	: 5.
6. Total number of bits in one frame	: 100.
7. Number of data bits in one frame	: 100.
8. Length of the synchronization sequence	: 31.
9. Threshold for synchronization detection	: 26.
10. Transmit data file name	: '/home/u6/csi/caedata/dom/data.dat'.
11. Transmit frequency file name	: '/home/u6/csi/caedata/dom/freqber.dat'.
12. Receive frequency file name	: '/home/u6/csi/caedata/dom/freqber1.dat'.
13. Sync code file name	: '/home/u6/csi/caedata/dom/ref1.dat'.
14. Initial1	: 1.
15. Flag for transmit or receive	: 1.
16. Flag for full or half duplex operation	: 0.
17. Counter1	: 1.
18. Counter2	: 1.
19. Counter3	: 1.

Editable Parameters:

1. Sampling frequency.
2. Bit rate.
3. Frame count.
4. Frame counter.
5. Number of hop frequencies.

6. Total number of bits in one frame.
7. Number of data bits in one frame.
8. Length of the synchronization sequence.
9. Threshold for synchronization detection.
10. Transmit data file name.
11. Transmit frequency file name.
12. Receive frequency file name.
13. Sync code file name.

4.4.2 The Receive Block

The operation of the receive block is shown in the flowchart in Figure 3.9. The receive block can be subdivided into four main sub-blocks: the receive frequency control sub-block, the FH synchronization sub-block, the data deformatter sub-block and the bit error rate sub-block.

The operation of the receive frequency control block is the same as that of the transmit frequency control sub-block. The same set of control signals are required for receive frequency control as those required for transmit frequency control. The FH synchronization sub-block is responsible for acquiring and tracking the hopping pattern at the receiver. This block consists of a 31 bit digital matched filter that is used for the initial acquisition. Tracking is performed by tracking the bit clock, and it is carried out by the timing recovery circuit. Once synchronization has been achieved, the data deformatter sub-block deformats the received frame; i.e. it strips the desired data bits off the received frame. This is accomplished by buffering the data arriving after the sync sequence has been detected. A counter is used to track the number of data bits buffered from the time the sync sequence is detected. Once the counter reaches the desired value of 592 bits, the data buffering is stopped. The bit error rate sub-block is used to determine the BER of the system. The demodulated and deformatted data bits are compared with the original transmitted data bits available in the ASCII data file, and the BER is then calculated.

Default Parameters:

1. Sampling frequency	: 2.1 MHz.
2. Bit rate	: 100 kbps.
3. Frame count	: 100.
4. Frame counter	: 1.
5. Number of hop frequencies	: 5.
6. Total number of bits in one frame	: 100.
7. Number of data bits in one frame	: 37.
8. Length of the synchronization sequence	: 31.
9. Threshold for synchronization detection	: 26.
10. Transmit data file name	: '/home/u6/csi/caedata/dom/data.dat'.
11. Transmit frequency file name	: '/home/u6/csi/caedata/dom/freqber.dat'.
12. Receive frequency file name	: '/home/u6/csi/caedata/dom/freqber1.dat'.
13. Sync code file name	: '/home/u6/csi/caedata/dom/ref1.dat'.
14. Initial1	: 0.
15. Flag for transmit or receive	: 0.
16. Flag for full or half duplex operation	: 1.
17. Counter1	: 1.
18. Counter2	: 1.
19. Counter3	: 1.

Editable Parameters:

As in the transmit block.

Chapter 5

Implementation of the Frequency Hopper

5.1 Introduction

The transceiver is split into two distinct segments for implementation: the transmitter and the receiver. Each of these segments is further sub-divided into two other segments: the radio frequency front end and the baseband processing.

Figure 5.1 shows the various functional blocks that make up the transceiver. The design is aimed at achieving maximum flexibility by providing the required processing and control functions in a set of programmable software modules that can be easily changed or replaced without changing the hardware architecture of the radio. Handcoded assembly language modules were used due to timing constraints. Figure 5.2 shows a detailed block diagram of the system architecture and the various interfaces between the baseband processing segment and the radio frequency front end.

5.2 The Transmitter

The transmitter consists of an RF front end and a baseband controller. A photograph

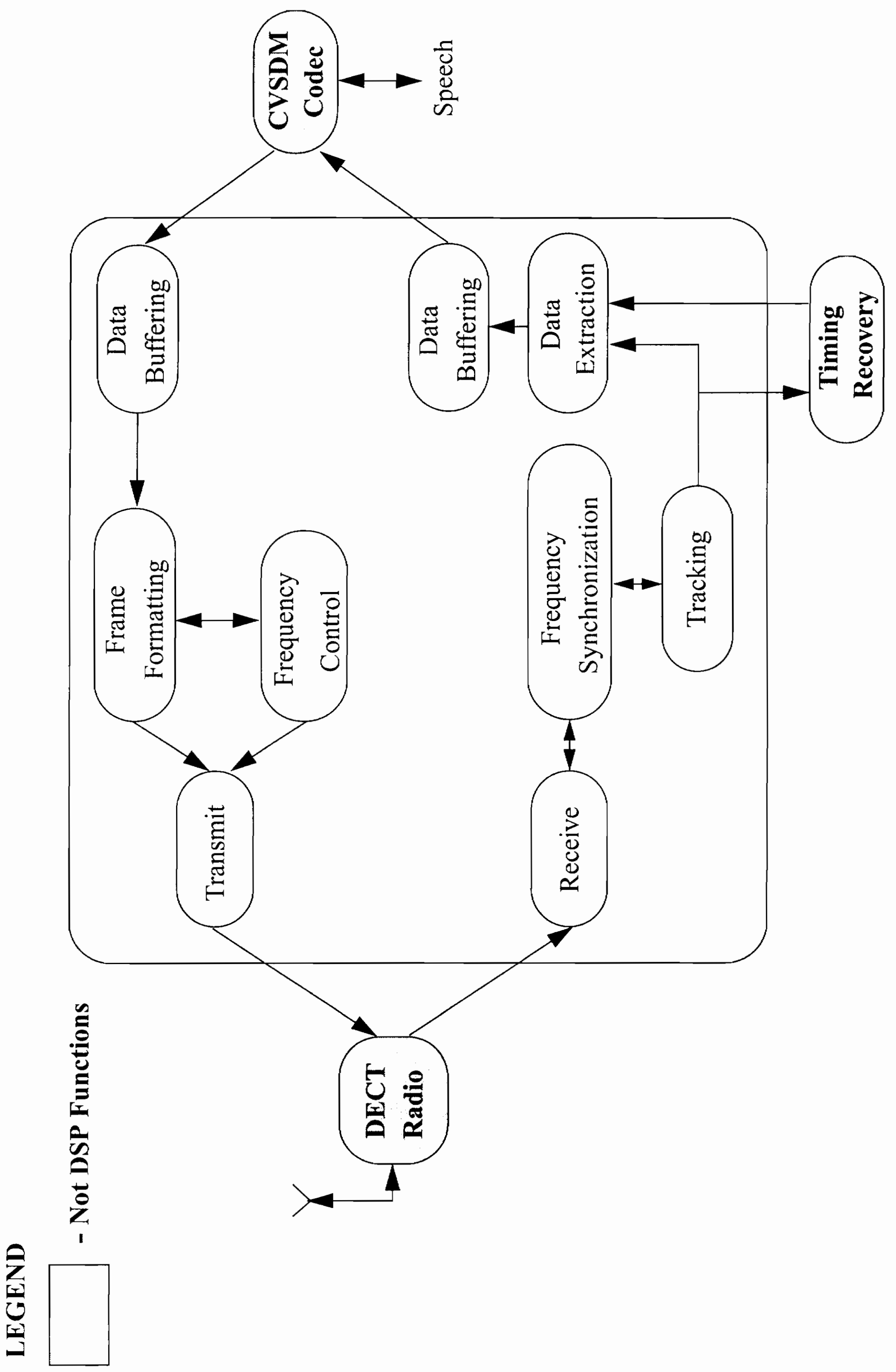


Figure 5.1 Functional Block Diagram of the FH Transceiver

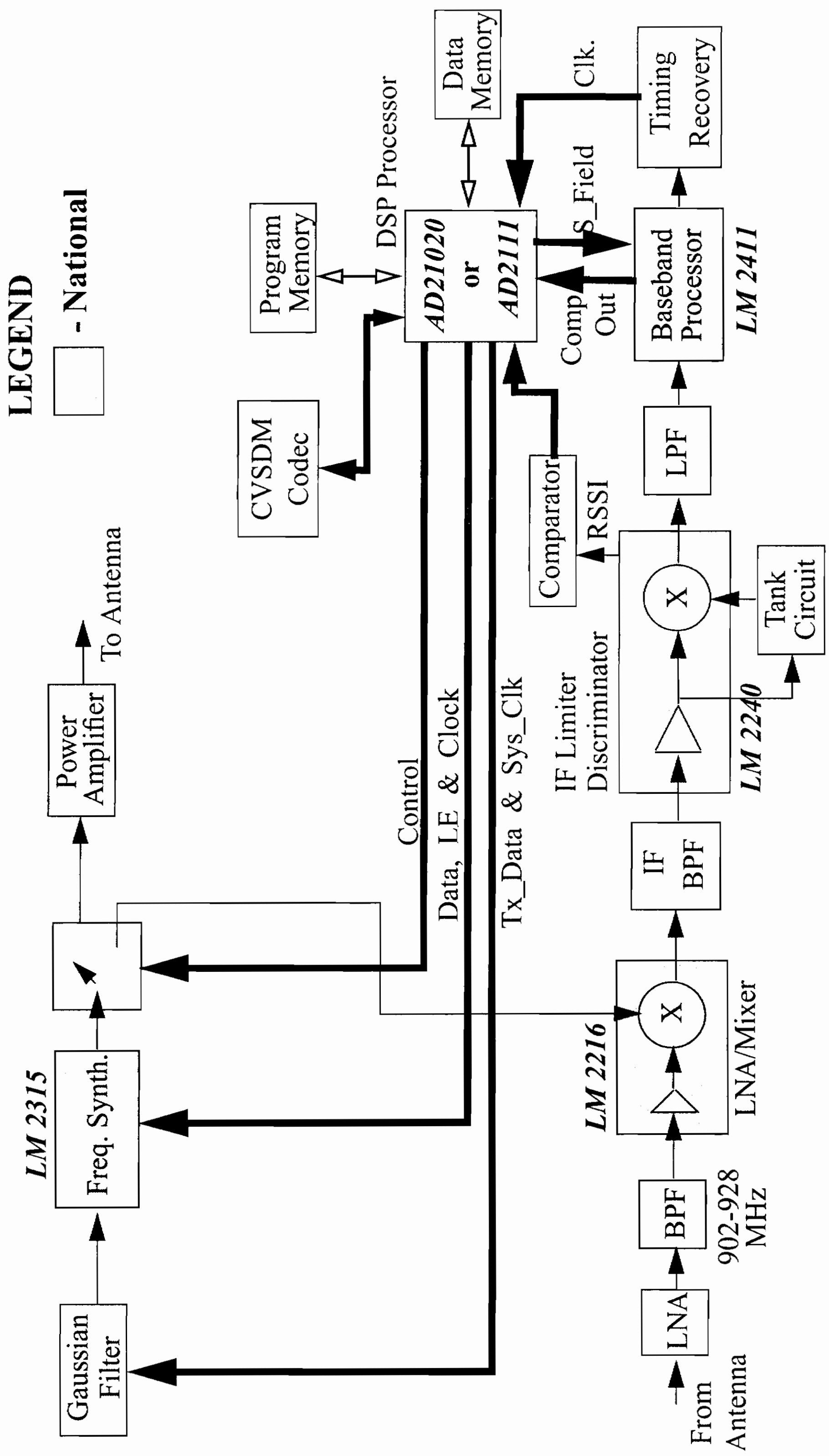


Figure 5.2 Block Diagram of the FH System Architecture and the Various Interfaces

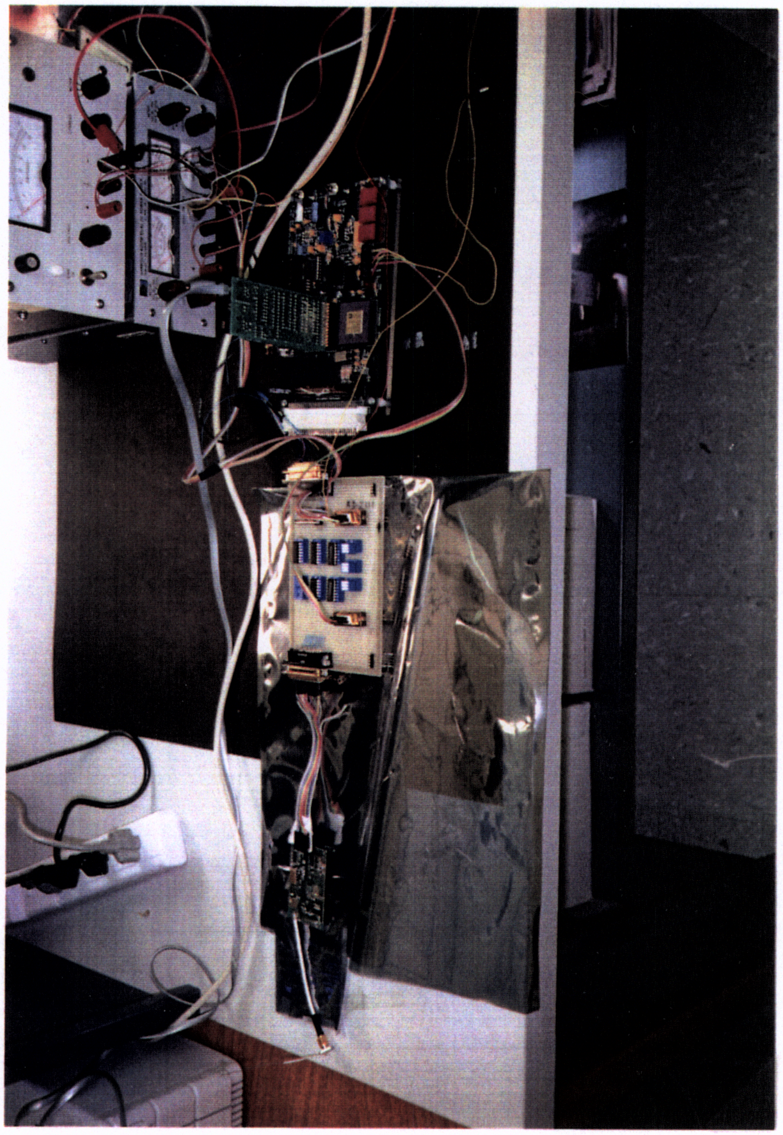


Figure 5.3 Photograph of the FH Transmitter

of the FH transmitter is shown in Figure 5.3.

5.2.1 The Radio Frequency (RF) Front End

The RF front end used is a regular DECT RF front end provided by National Semiconductor. The specifications of this front end are given in Table 2.1. It should be noted that the RF specifications for the DECT system are different from those of the transceiver in terms of the data signal bandwidth. Thus, it is important to study the effects of the differences in order to optimize system performance. The different data rates affect two parameters in the transmitter:

1. The clock used for the Gaussian pulse ROM DAC has to be 1.40625 MHz instead of the 10.368 MHz used for DECT.

2. The bandwidth of the lowpass smoothing filter used after the ROMDAC should be 156 KHz instead of 1 MHz.

An external frequency synthesizer operating at 9.842 MHz is used as the frequency reference for the PLL in the frequency synthesizer.

5.2.2 The Baseband Processing

The baseband processing for the transmitter is carried out by an Analog Devices AD-2111 DSP microcontroller. This is a 16 bit fixed point processor with 512 bytes of on chip data memory and 1 kilobyte of program memory. The AD-2111 has a computational rate of 15 million instructions per second [27]. The baseband processing stage is composed of a hardware interface,and software required for the operation of the transmit segment of the FH transceiver.

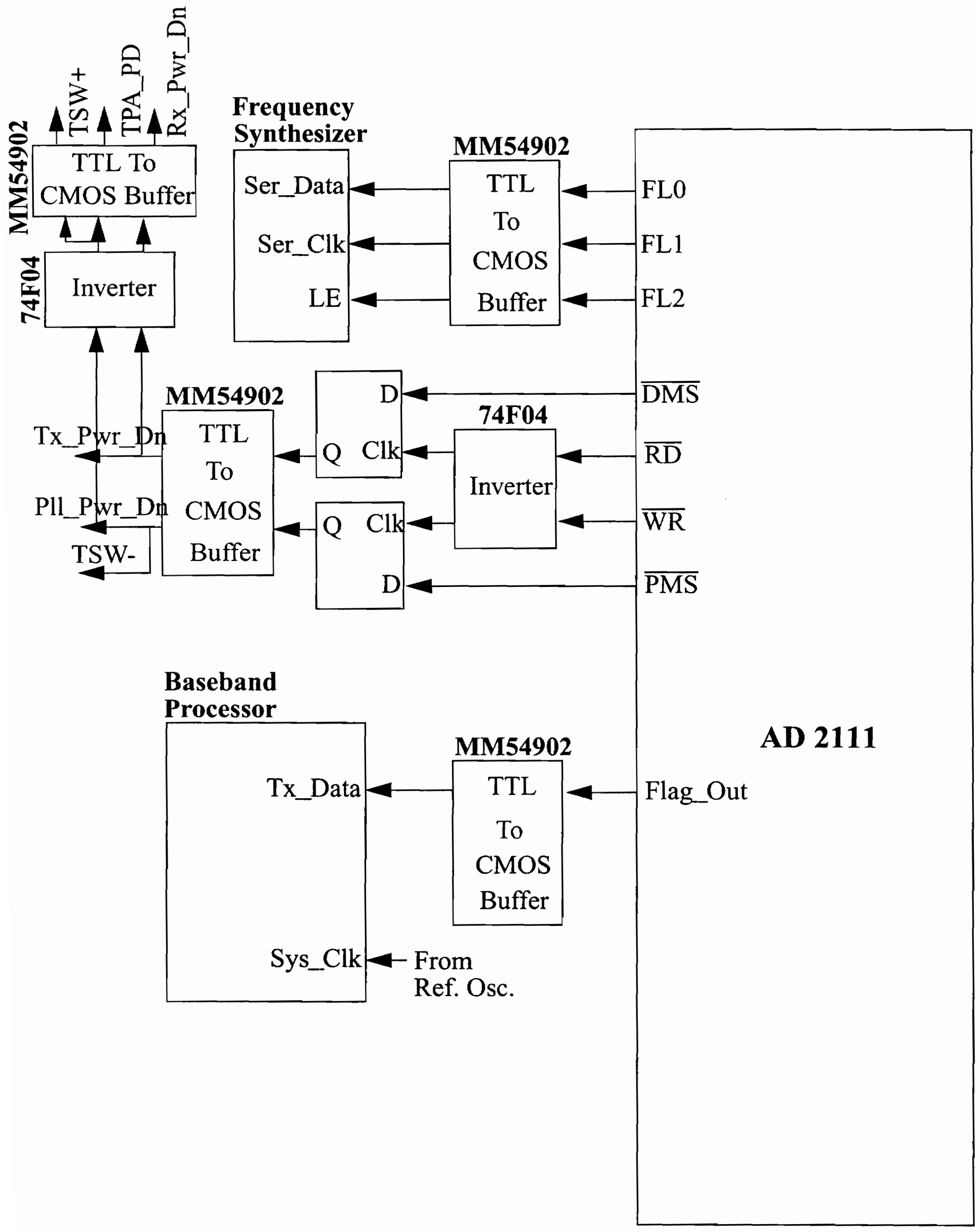


Figure 5.4 Block Diagram of the Logical Hardware Interface for the Transmitter

5.2.2.1 The Hardware Interface

The block diagram of the logical hardware interface for the transmitter is shown in Figure 5.3. The following signals are required by the RF front end for proper transmit operation:

1. Transmitter Power Down (Tx_Pwr_Dn).
2. Receiver Power Down (Rx_Pwr_Dn).
3. PLL Power Down (Pll_Pwr_Dn).
4. Transmitter Switch Positive (TSW+).
5. Transmitter Switch Negative (TSW-).
6. Transmit Power Amplifier Down (TPA_PD).
7. VCO Power Down (VCO_PD).
8. Serial Data (Ser_Data).
9. Serial Clock (Ser_Clk).
10. Load Enable (LE).
11. Transmit Data (Tx_Data).
12. System_Clk (Sys_Clk).

The Tx_Pwr_Dn signal is required to turn on and off the transmit section of the LMX2411 baseband processor and to conserve battery power. The transmit section of the baseband processor consists of the pulse shaping Gaussian filter. The logic levels required for proper operation are shown in Table 5.1.

Table 5.1 Tx_Pwr_Dn Operation

Function	**Tx_Pwr_Dn Logic State**
Tx Section of LMX2411 switched on	0
Tx Section of LMX2411 powered down	1

Tx_Pwr_Dn is generated by using the AD-2111 tp perform a dummy data memory write to some unused external memory location. This dummy write in executed in assembly software. By doing so, the Data Memory Select (DMS) line becomes active. This is used in conjunction with the Read (RD) signal generated at the same instant to write to the D-flipflop which serves as a latch here, to hold the value written until it is written into again. The output of the D-flipflop is sent to a TTL-CMOS buffer (MM54902), which is required since the D-flipflop is TTL logic and all the interfaces to the RF front end need to be CMOS. The RD signal is passed through an inverter (SN74F04) to delay RD so that the voltage level on the DMS line will have stabilized when it is read into the latch. The dummy read should be to a memory address that is mapped in the external memory space of the AD 2111 since only then would the DMS and RD lines become active. The logic levels required at for proper operation DMS and RD are shown in Table 5.2. DMS and RD are active LOW.

Table 5.2 Generation of the Tx_Pwr_Dn Signal

Value Written Into Latch	$\overline{\textbf{DMS}}$	$\overline{\textbf{RD}}$
1	1	0
0	0	0

Since the FH radio architecture used is TDD, the receive section of the LMX2411 baseband processor will not be active when the transmit section is active and vice versa. This means that Tx_Pwr_Dn can be inverted and used as the Rx_Pwr_Dn signal. The inverter used is the SN74F04.

Pll_Pwr_Dn is used to open the feedback loop of the PLL in the frequency synthesizer to allow open loop modulation of the VCO to occur during the transmit frame. Pll_Pwr_Dn is generated in a manner similar to the generation of Tx_Pwr_Dn. In this case, the signals used to write into the D-flipflop are Program Memory Select (PMS) and Write (WR). The logic levels required at PMS and WR for proper operation are shown in

Table 5.3.

TSW+ and TSW- are used for the Tx/Rx switch at the antenna and the VCO output. These signals are the inverse of each other, and both are used in the implementation. The logic levels required for operation are shown in Table 5.4. Since TSW+ and TSW- are required to be active at the beginning of the data transmission or the reception of the signal, TSW+ and TSW- are derived from Pll_Pwr_Dn. The assignments are given in Table 5.5.

Table 5.3 Generation of the Pll_Pwr_Dn Signal

Value Written Into Latch	$\overline{\text{PMS}}$	$\overline{\text{WR}}$
1	1	0
0	0	0

Table 5.4 TSW+ Operation

Function Carried Out	TSW+
Output of VCO Goes to Power Amplifier	1
Output of VCO Goes to Rx Mixer	0

Table 5.5 Assignments for TSW+ and TSW-

TSW+	- Pll_Pwr_Dn
TSW-	Pll_Pwr_Dn

TPA_PD is used for turning the power amplifier on and off. This signal should enable the power amplifier at least 27 microseconds prior to start of transmission. TPA_PD is derived from Pll_Pwr_Dn and is the logical complement of Pll_Pwr_Dn.

VCO_PD is used to power down the VCO of the PLL in the frequency synthesizer. The VCO in the design is always active during a link and is never powered down.

VCO_PD which is active LOW is thus permanently tied LOW.

Ser_Data, Ser_Clock, and LE are the three signals through which the PLL of the frequency synthesizer is programmed. This interface has write only capability. The PLL has two counters, the 14 bit programmable reference divider (*R*) counter and the 18 bit programmable divider (*N*) counter, and a single bit prescaler *S* which needs to be programmed to the desired center frequency. The PLL also has a 19 bit data register in which the data written into the counter is stored before it is transferred to the *R* or the *N* counter. The configurations of the counters are shown in Figure 5.4. If the Control Bit, which is the last bit shifted into the data register is HIGH, data is transferred from the 19 bit shift register into a 14 bit latch which sets the 14 bit *R* counter and the one bit *S* latch which sets the prescaler. The data format for the *R* counter is shown in Table 5.6.

Table 5.6 Data Format of the R Counter

Divide Ratio R	**S 14**	**S 13**	**S 12**	**S 11**	**S 10**	**S 9**	**S 8**	**S 7**	**S 6**	**S 5**	**S 4**	**S 3**	**S 2**	**S 1**
3	0	0	0	0	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
16383	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The data format for the prescaler is shown in Table 5.7. If the Control Bit is LOW,

Table 5.7 Prescaler Format

Prescaler Select P	**S 15**
128/129	0
64/65	1

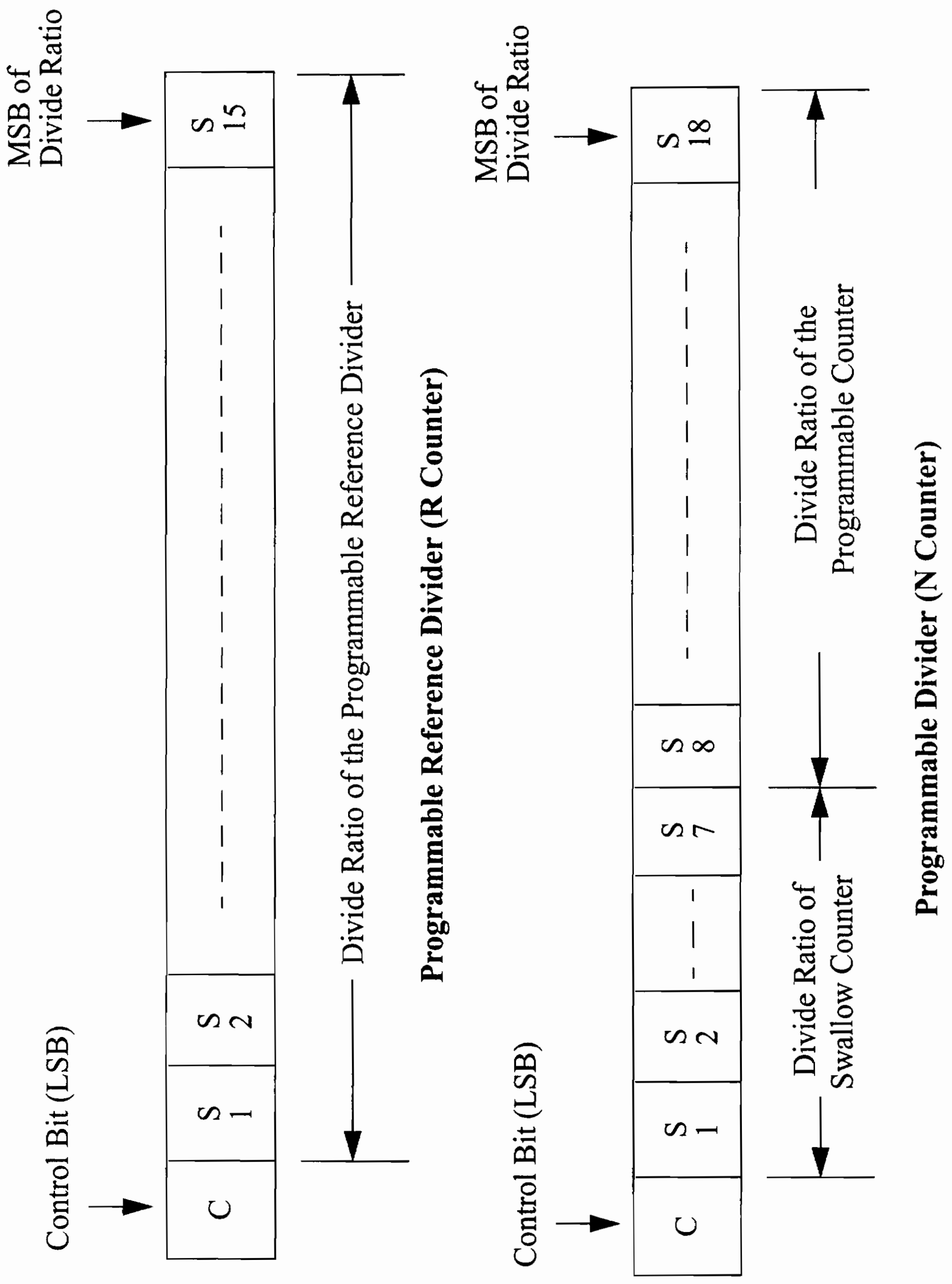

Control Bit (LSB)
MSB of Divide Ratio
C
S 1
S 2
S 15
Divide Ratio of the Programmable Reference Divider
Programmable Reference Divider (R Counter)
Control Bit (LSB)
MSB of Divide Ratio
C
S 1
S 2
S 7
S 8
S 18
Divide Ratio of Swallow Counter
Divide Ratio of the Programmable Counter
Programmable Divider (N Counter)

Figure 5.5 Configurations of the Counters Used by the PLL

the data is transferred from the data register to the *N* counter. The *N* counter consists of a 7 bit swallow counter (*A* counter) and a 11 bit programmable counter (*B* counter). The serial data format of the *N* counter is given in Tables 5.8 and 5.9.

Table 5.8 Data Format of the A Field of the N Counter

Divide Ratio A	**S 7**	**S 6**	**S 5**	**S 4**	**S 3**	**S 2**	**S 1**
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.
127	1	1	1	1	1	1	1

Table 5.9 Data Format of the B Field of the N Counter

Divide Ratio B	**S 18**	**S 17**	**S 16**	**S 15**	**S 14**	**S 13**	**S 12**	**S 11**	**S 10**	**S 9**	**S 8**
3	0	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.	.	.
2047	1	1	1	1	1	1	1	1	1	1	1

The AD-2111 has three asynchronous input/output (I/O) pins called the FLAG pins that can be configured as either inputs or outputs. These three pins are configured during the initialization to be output pins and are used as the interface for the Ser_Data, Ser_Clk, and the LE signals required for programming the PLL. The assignment of the pins is shown in Table 5.10.

The *R* counter is programmed first by the AD-2111 then by the *N* counter. For details of the timing for the various signals, refer to [33].

Table 5.10 Pin Assignment for the PLL Interface

AD 2111 Pin	PLL Interface
FLAG0	Ser_Data
FLAG1	Ser_Clk
FLAG2	LE

The data to be transmitted is sent to the Tx_Data pin of the LMX2411 baseband processor. The FLAG_OUT asynchronous output of the AD-2111 is used to interface with the Tx_Data input of the LMX2411. The Sys_Clk is the oversampling clock required by the gaussian filter. The Sys_Clk used here is 9.842 MHz and is fed from an external frequency source.

The circuit diagram for the transmitter hardware interface between the DECT RF front end and the AD-2111 is shown in Figure 5.5.

5.2.2.2 The Software Executive

The software executive for the transmitter consists of two distinct blocks: the baseband data block and the RF interface block.

The baseband data block software executive can be further divided into two sub-blocks: the software data buffering for the CVSDM data and the formatting block. When a voice communications link has been established, data from the CVSDM codec is available to the DSP processor for transmission. The CVSDM data cannot be fed directly to the GMSK modulator for two reasons:

1. The CVSDM data rate is 52.083 kbps, while the transmission rate is 156.25 kbps.

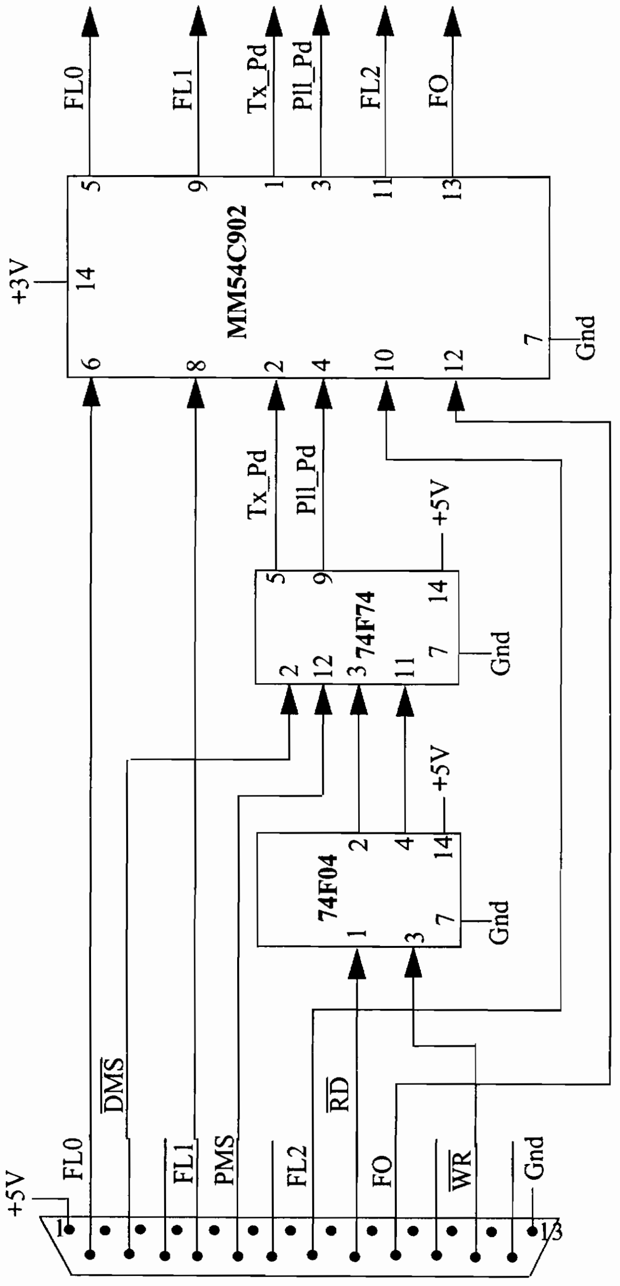


Figure 5.6 Circuit Diagram of the Transmitter Hardware Interface

2. Packet transmission is used, and hence a frame has to be built.

A double buffered scheme is used to buffer data from the codec for transmission. The principle of operation of the double buffer is shown in Figure 5.6. As seen from the figure, the buffer consists of two separate memory blocks each of which can be addressed independently of the other. The operation of this scheme is as follows. First, both memory blocks are initialized to zero by the DSP whenever a link is established. Data from the codec at 52.083 kbps is written into buffer #2. Data for transmission is read from memory buffer #1 at the same time for transmission at a rate of 156.25 kbps. Once 480 bits have been read into buffer #2 by the codec (480 bits is the number of CVSDM bits required for 9.216 milliseconds of speech at a codec rate of 52.083 kbps), the sequence of operations is switched between the two buffers. Data from the codec is now written into buffer #1, and data for transmission is read from buffer #2 by the DSP processor. It should be noted here that the DSP reads data from the buffer at a much faster rate than data is written into the buffer. This means that the operations on the buffer from which data is read from for transmission will end sooner than the other buffer. This sequence of operations is repeated until the link is deactivated.

The formatting block is used to build the transmit frame. The 32 bit header and the 31 bit PN synchronization sequence are stored in memory. Once a frame is ready for transmission, the DSP retrieves the header and the synchronization sequence from memory and sends them to the Tx_Data input of the LMX2411 baseband processor. Data from the appropriate memory buffer is then transmitted at the channel rate of 156.25 kbps.

The RF interface block is responsible for programming the PLL, activating the transmit power amplifier and switching the power amplifier output to the antenna. The programming of the PLL is performed in two stages. In the first stage, the R and N counters of the PLL are programmed with the appropriate words. In the second stage, the PLL is allowed sufficient time to settle down, before further operations are carried out. The con-

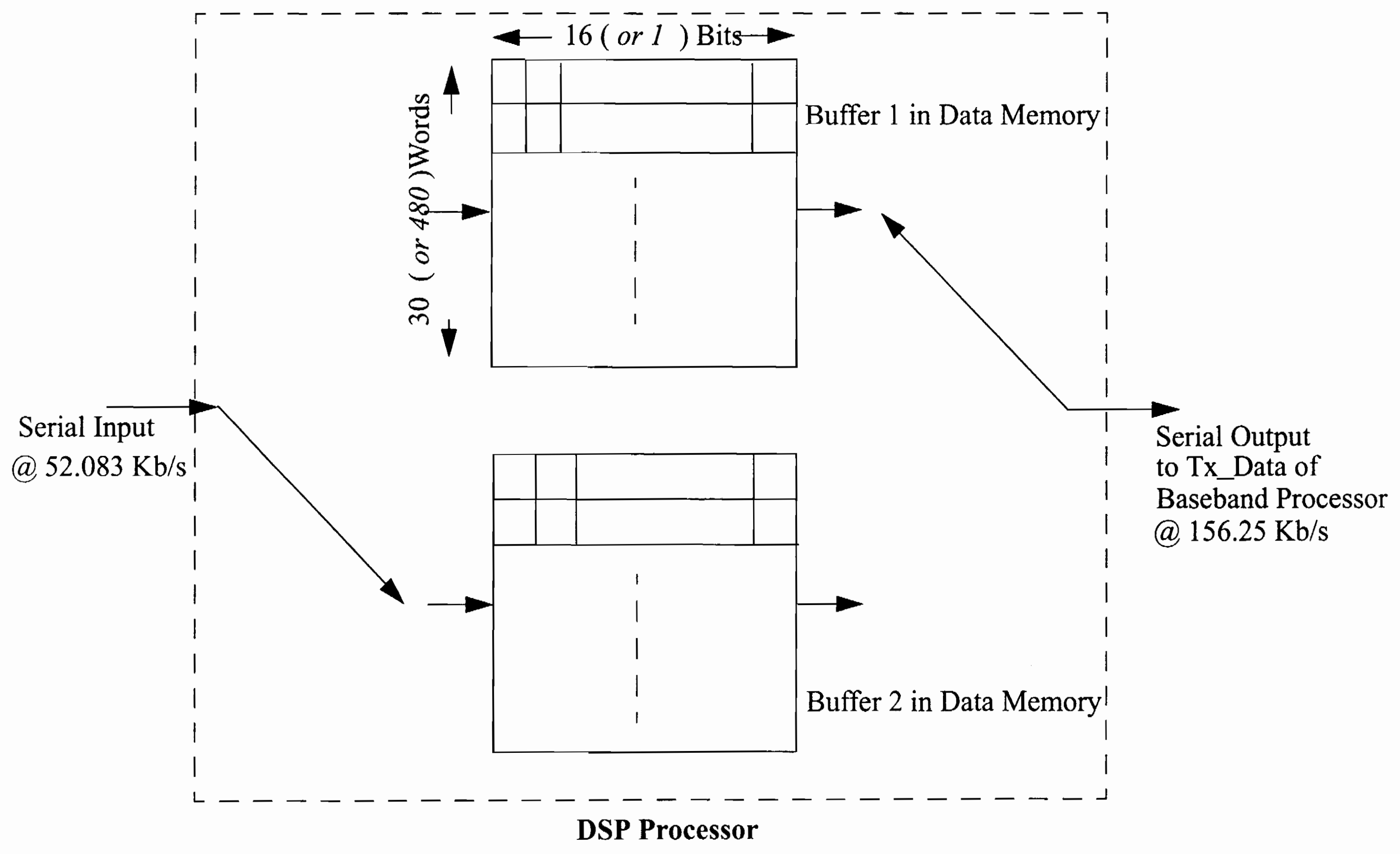


Figure 5.7 Implementation of the Double Buffer by the DSP Processor

trol signals for the power amplifier and the antenna are generated through dummy external memory reads and writes as per the format described in section 5.2.2.1.

The flowchart for the transmitter executive as implemented on the DSP is shown in Figures 5.7, 5.8, 5.9, and 5.16. The labels used are the same as those used in the assembly code. The two memory buffers for the double buffer are implemented as two separate blocks of internal data memory on the AD-2111 processor. The header and the synchronization sequence are also stored in separate internal data memory locations, which can be accessed by the DSP in one clock cycle. The AD-2111 is interrupted by the on chip timer interrupt every 6.4 microseconds. The timer interrupt service routine (ISR) is then activated. The timer ISR consists of a transmit ISR and a receive ISR block, only one of which is active at any instant of time. The transmit timer ISR carries out all the tasks required for the transmit executive. Every three timer interrupts, data from the codec is written into internal data memory. Before the beginning of each transmit frame, the PLL is programmed to the appropriate carrier frequency, the power amplifier is powered up and the antenna is connected to the output of the power amplifier. The appropriate words for the R and N counters for the various hop frequencies are stored in internal data memory. The PLL is programmed during the guard interval of 65 bit periods. It takes 16 bit periods to program the R counter and 19 bits to program the N counter. The remaining 30 bit periods of the guard interval are used to allow the PLL to settle down. Once these operations are complete, a transmit frame is built and data is then transmitted. The timer ISR keeps track of the number of bits transmitted, and at the end of the transmit frame, sets a flag that hands over control to the receive timer ISR.

5.3 The Receiver

The receiver consists of an RF front end and the same baseband controller. A photograph of the FH receiver is shown in Figure 5.10.

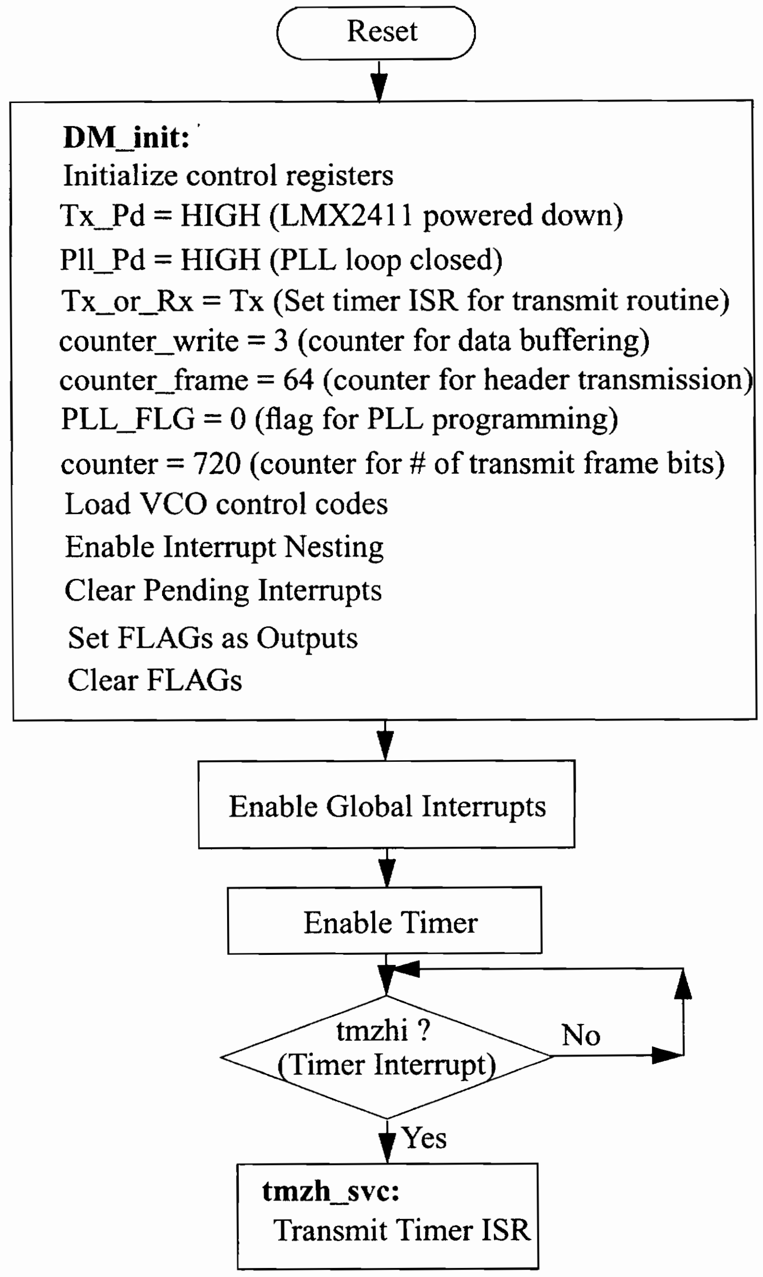

Reset
DM_init:
Initialize control registers
Tx_Pd = HIGH (LMX2411 powered down)
Pll_Pd = HIGH (PLL loop closed)
Tx_or_Rx = Tx (Set timer ISR for transmit routine)
counter_write = 3 (counter for data buffering)
counter_frame = 64 (counter for header transmission)
PLL_FLG = 0 (flag for PLL programming)
counter = 720 (counter for # of transmit frame bits)
Load VCO control codes
Enable Interrupt Nesting
Clear Pending Interrupts
Set FLAGs as Outputs
Clear FLAGs
Enable Global Interrupts
Enable Timer
tmzhi ? (Timer Interrupt)
No
Yes
tmzh_svc:
Transmit Timer ISR

Figure 5.8 Flowchart for the Transmit Executive

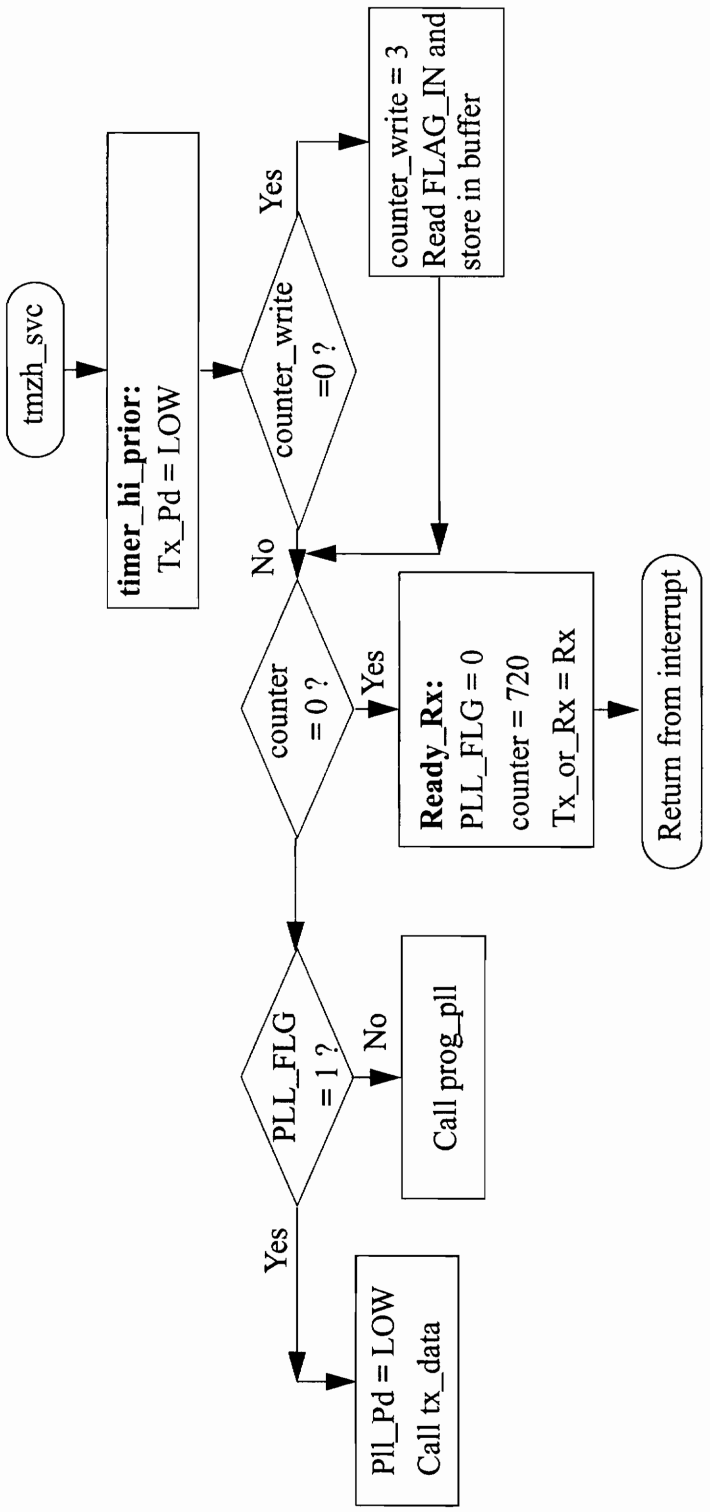


Figure 5.9 Flowchart for the Timer ISR of the Transmit Executive

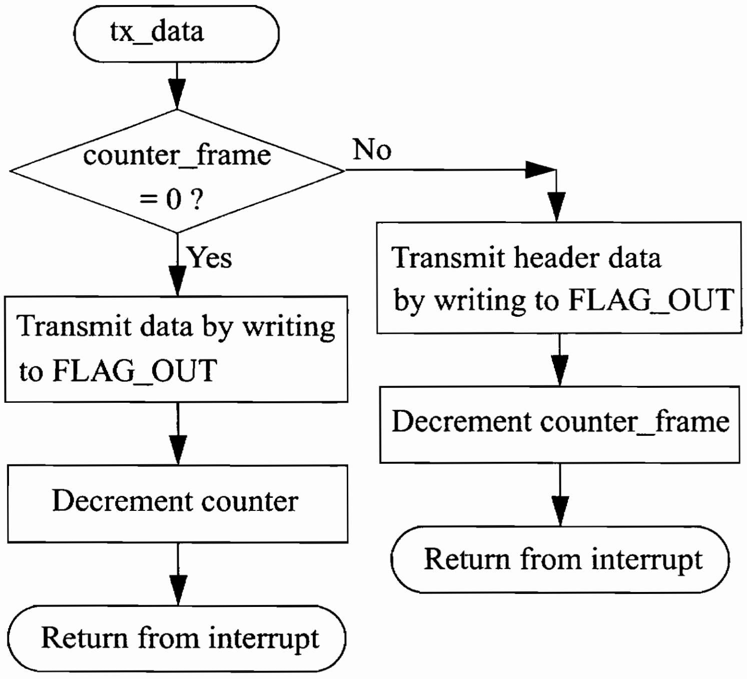


Figure 5.9 Flowchart for the Transmit Data Section of the Transmit Timer ISR

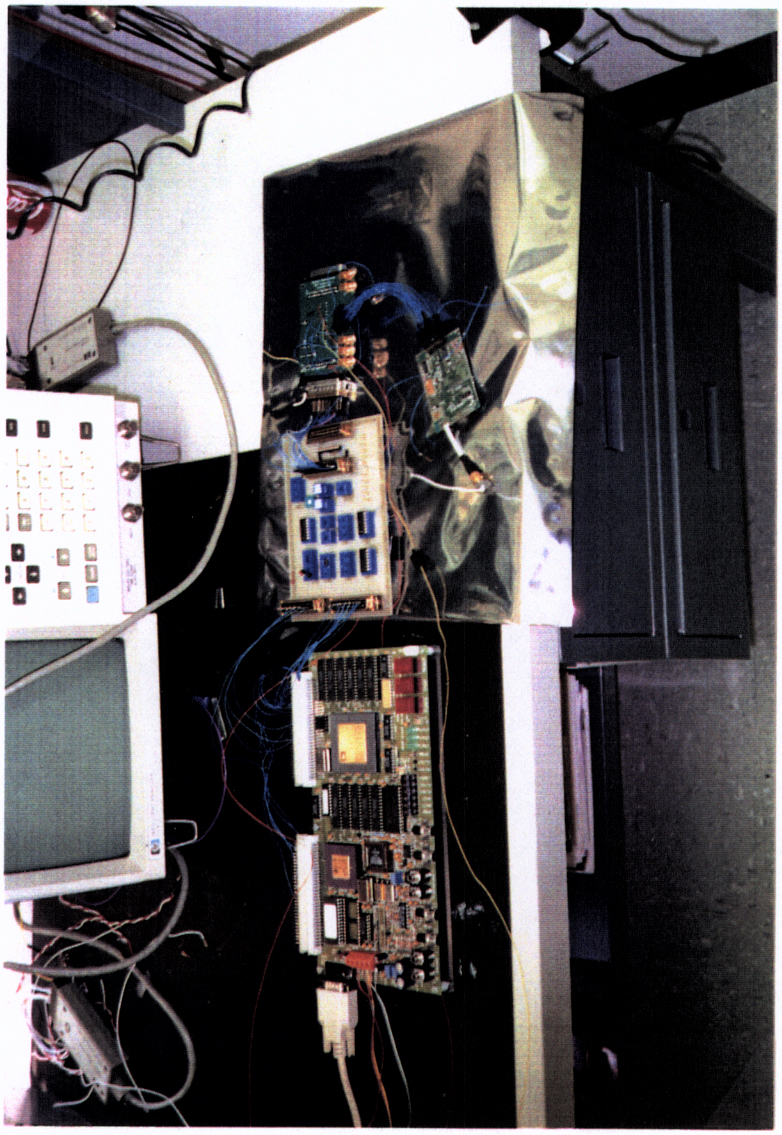


Figure 5.10 Photograph of the FH Receiver

5.3.1 The Radio Frequency (RF) Front End

The difference in the data rates between the two systems impacts the receiver much more severely than it does the transmitter, since the original data bandwidth determines the bandwidth of the various bandpass filters, the GMSK demodulator and the lowpass filter that filters the demodulated signal to extract the baseband data sequence. The effect on the various blocks is discussed in more detail below:

1. The required bandwidth of the IF filter is 234.375 KHz whereas the existing IF bandwidth is about 1.4 MHz. This larger bandwidth results in noise outside the 234.375 KHz bandwidth of the transceiver, being present at the input to the demodulator.

2. The bandwidth of the LC-tank circuit that is used in the IF Limiter-Discriminator GMSK demodulator is about 1.5 MHz, whereas the required bandwidth is only 234.375 KHz. This larger bandwidth results in noise present outside the desired signal band of interest, aliasing into the signal band of interest due to the non-linear operation of the demodulator. In addition, the amplitude of the eye produced at the demodulator output is reduced. This reduction occurs because the maximum phase shift that the desired signal experiences as it passes through the tank is much smaller, since the tank is designed to handle a data signal bandwidth of 1.728 MHz. Both of these effects decrease the signal to noise ratio from what is possible with this architecture.

3. The bandwidth of the lowpass filter used to filter the demodulated signal is 1 MHz, while the bandwidth required is 156.25 KHz. In the implementation, the existing lowpass filter is replaced by a four pole active butterworth filter with a bandwidth of 156 KHz.

4. The time constant of the RC integrator used to recover the average value of the demodulated signal should be 102.4 microseconds instead of the 18 microseconds that is generally used in DECT.

5.3.2 The Baseband Processing

The baseband processing for the receiver is carried out by an Analog Devices AD-21020 DSP microprocessor. This is a 32 bit floating point processor with a computational rate of 25 million instructions per second. The baseband processing stage is composed of a hardware interface and software required for the operation of the transmit segment of the FH transceiver.

5.3.2.1 The Hardware Interface

The block diagram of the logical hardware interface for the receiver is shown in Figure 5.10. The following signals are required by the RF front end for proper receive operation.

1. Transmitter Power Down (Tx_Pwr_Dn).
2. Receiver Power Down (Rx_Pwr_Dn).
3. PLL Power Down (Pll_Pwr_Dn).
4. Transmitter Switch Positive (TSW+).
5. Transmitter Switch Negative (TSW-).
6. Transmit Power Amplifier Down (TPA_PD).
7. VCO Power Down (VCO_PD).
8. Serial Data (Ser_Data).
9. Serial Clock (Ser_Clk).
10. Load Enable (LE).
11. Comparator Output (Comp_Out).
12. S_Field.

Signals 1 to 9 are the same as those for the transmit section and hence only the physical interface will be discussed here. Tx_Pwr_Dn is generated by writing the appropriate

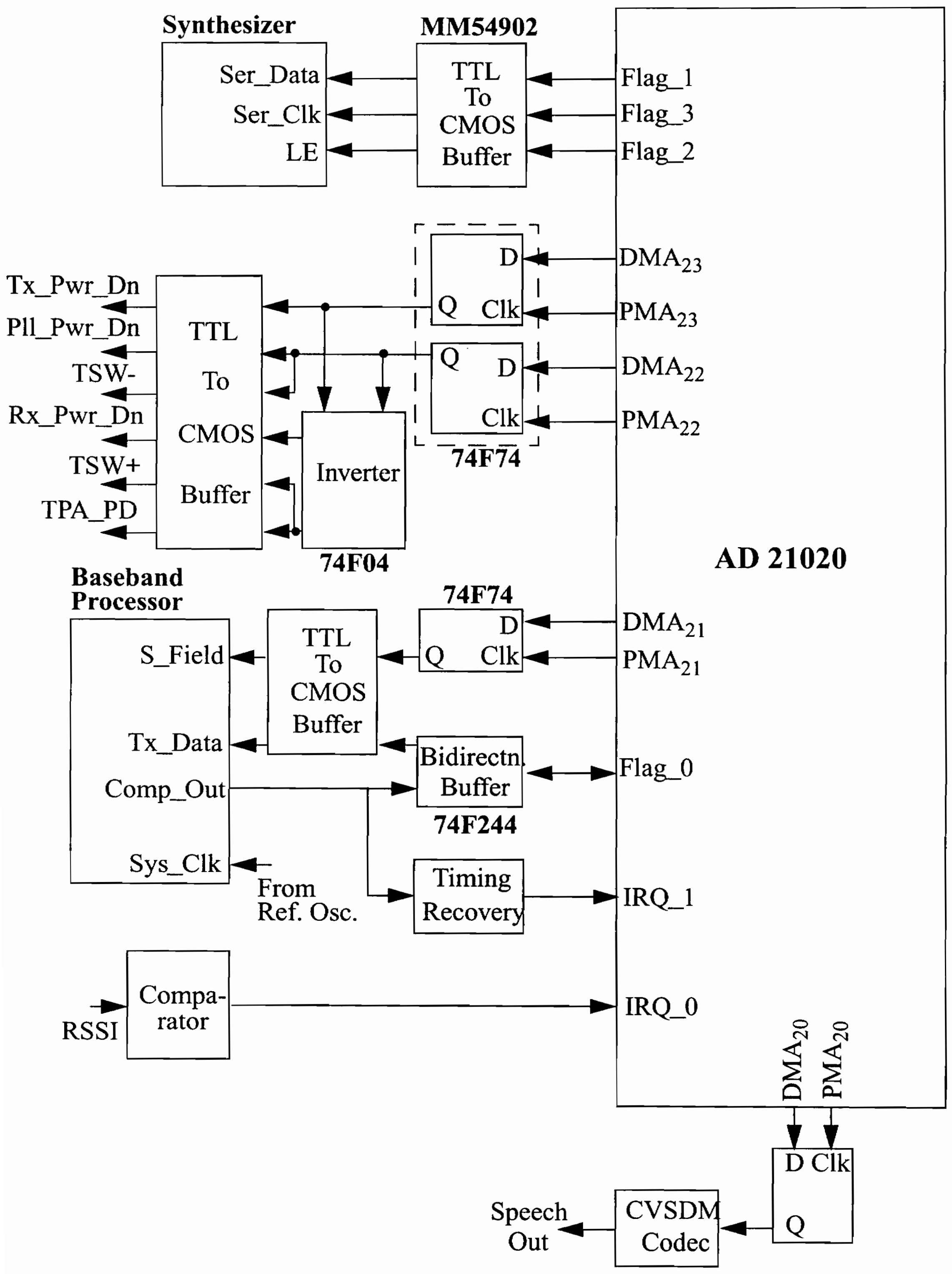


Figure 5.11 Block Diagram of the Logical Hardware Interface for the Receiver

value into a D-flipflop (SN74F74). The required value is written into the D-flipflop by having bit 23 of the Program Memory Address bus (PMA_{23}) clock in the value at bit 23 of the Data Memory Address bus (DMA_{23}). Since the radio architecture is TDD, Tx_Pwr_Dn is inverted and used as Rx_Pwr_Dn.

Pll_Pwr_Dn is generated by writing the appropriate value into a D-flipflop (SN74F74). The required value is written into the D-flipflop by having bit 22 of the Program Memory Address bus (PMA_{22}) clock in the value at bit 22 of the Data Memory Address bus (DMA_{22}).

TSW+ and TSW- are both derived from Pll_Pwr_Dn in a manner equivalent to that of the transmit section. TPA_PD is also derived from Pll_Pwr_Dn. VCO_PD is permanently tied LOW. Ser_Data, Ser_Clk and LE are connected to the asynchronous FLSG pins FLAG1, FLAG2, and FLAG3 respectively. The FLAG pins are all configured to be outputs.

Comp_Out which is the reshaped demodulated data output of the LMX2411 baseband processor, is connected to FLAG0. S_Field is used to hold the average value of the demodulated data stream and is generated by writing the appropriate value into a D-flipflop (SN74F74). The required value is written into the D-flipflop by having bit 21 of the Program Memory Address bus (PMA_{21}) clock in the value at bit 21 of the Data Memory Address bus (DMA_{21}).

Symbol timing recovery is performed external to the DSP processor. The technique implemented is the squaring loop. The block diagram of the timing recovery loop implemented is shown in Figure 5.11. A single pole RC filter is used as the delay line. The RC filter is used to introduce a delay of 3.2 microseconds which corresponds to half the data bit period. A comparator (LM 411) is used at the output of the RC filter to shape the ana-

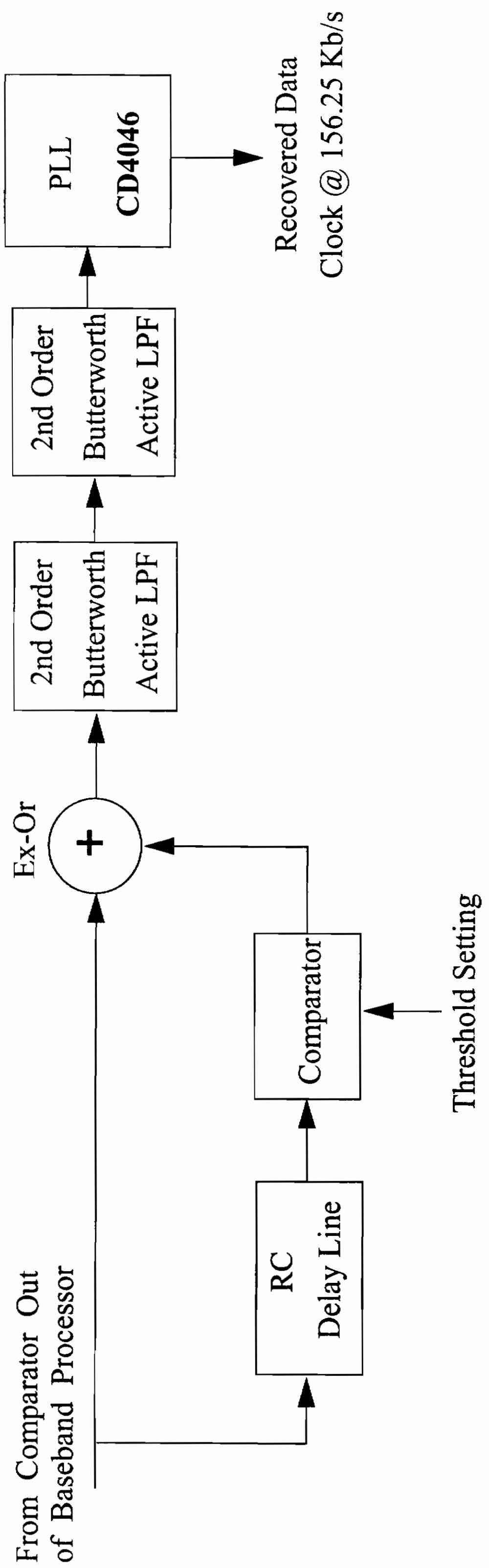


Figure 5.12 Block Diagram of the Timing Recovery Implementation

log signal into a NRZ pulse shape. The delayed and the original signal are then multiplied together by the ex-or gate (SN74F86). The output of the ex-or gate is then filtered by a fourth order butterworth active filter implemented by two cascaded second order sections. A PLL (CD4046) is then used to track the discrete clock component at the output of the filter. The PLL tracking bandwidth is 10 KHz.

The circuit diagram for the receiver hardware interface between the DECT RF front end and the AD-21020 is shown in Figure 5.12.

5.3.2.2 The Software Executive

The software executive for the transmitter consists of two distinct blocks: the base-band data block and the RF interface block.

The RF interface block is responsible for the programming of the PLL, switching the output of the frequency synthesizer to the mixer and routing the signal from the antenna to the LNA. The generation of these signals is carried out in a fashion similar to that discussed in section 5.2.2.2.

The baseband software executive can be divided into 3 sub-blocks: the synchronization block, deformatting block and the data buffering block. The synchronization executive is responsible for the receiver FH generator in step with that of the transmitter. The synchronization executive must first acquire the FH signal and then track it. The acquisition executive is basically concerned with the detection of the synchronization sequence transmitted in each frame. This is done by passing the demodulated signal through a matched filter implemented in software. The equivalent architecture of the software matched filter is shown in Figure 5.13. The tapped delay line is represented by a memory array in the internal data memory of the AD-21020. The filter coefficients, which are a mirror image of the transmitted synchronization sequence, are also stored in internal data

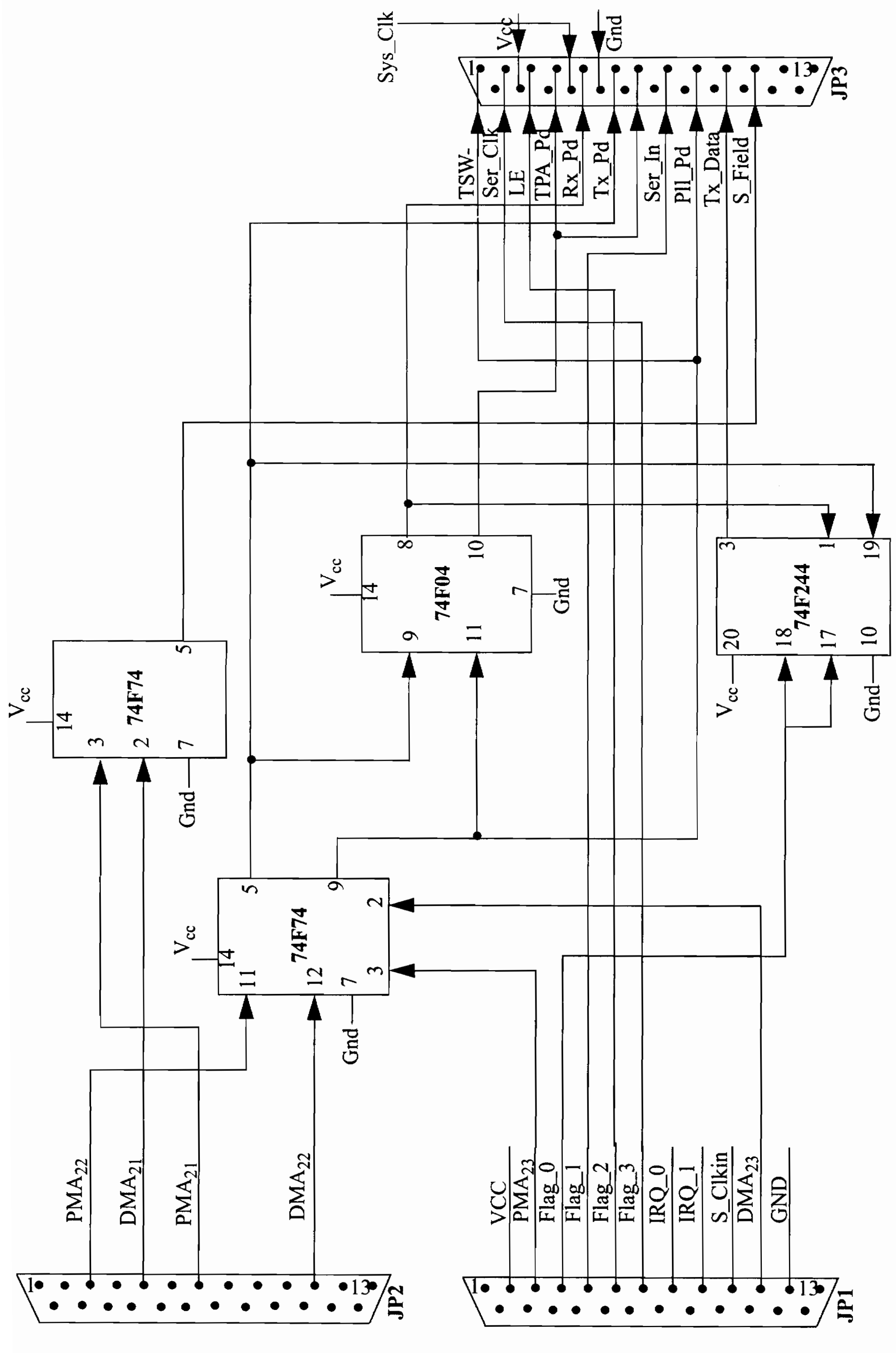


Figure 5.13 Circuit Diagram of the Receiver Hardware Interface

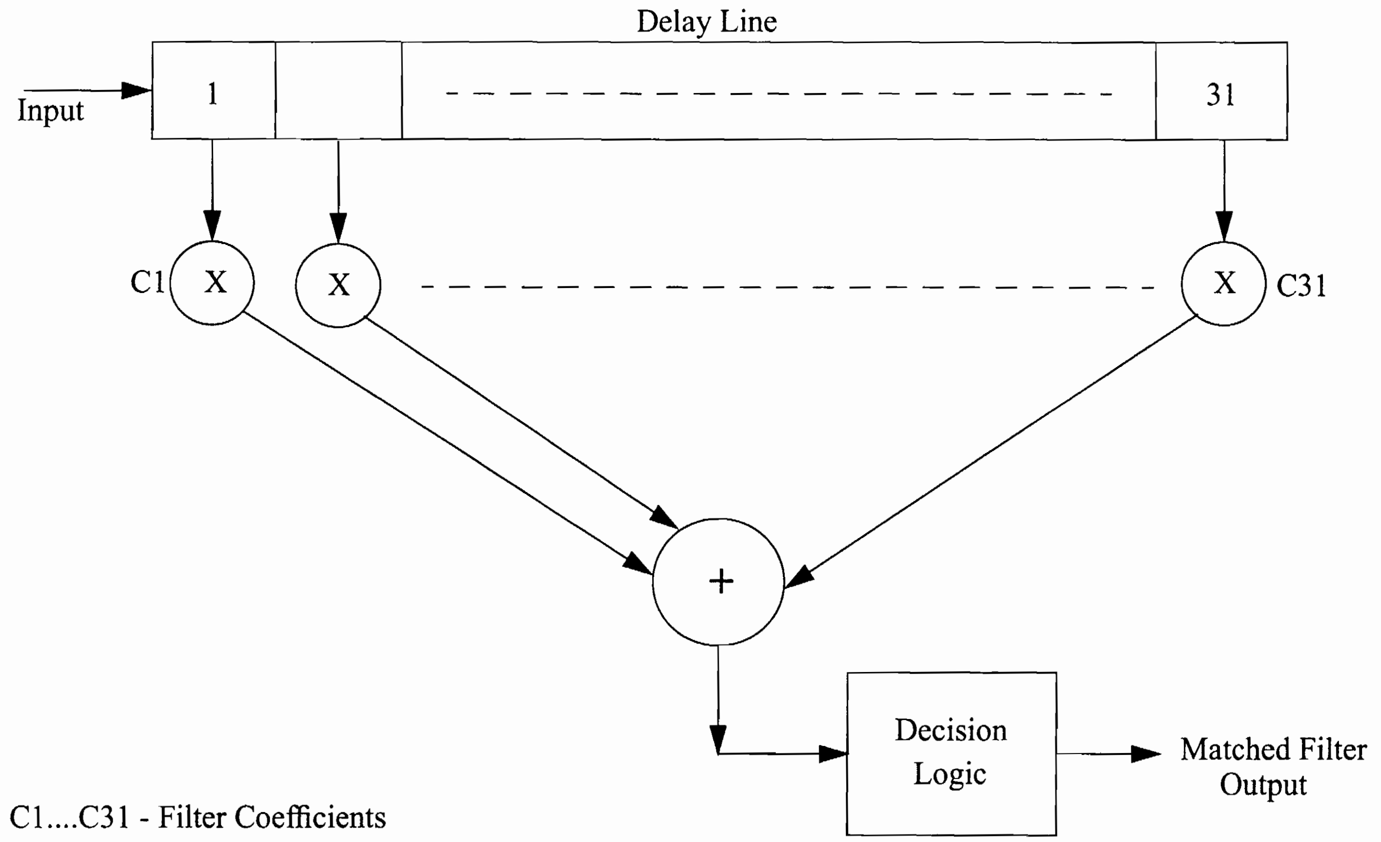


Figure 5.14 Matched Filter Configuration Used for the Detection of the Synchronization Sequence

memory. Once acquisition has been achieved, the tracking software suite keeps track of the number of bits received (with the received data bits being read into the DSP by the recovered clock), thereby keeping in step with the transmitter. During tracking, the acquisition executive is deactivated.

The deformatting block is used to strip the CVSDM data bits from the received frame, and it is activated only after the synchronization block has detected the presence of the synchronization sequence.

The CVSDM data from the received frame is first buffered through a double buffer similar to the one implemented in the transmitter. The double buffering is done for two reasons:

1. Received data is available at 156.25 kbps whereas the data to be fed to the codec for reconstruction has a data rate of 52.083 kbps.
2. Data is received in packets, whereas the bits have to be fed in a continuous stream to the receive codec.

The double buffered scheme used is the same as that used for the transmit section except that data is written to the buffer at 156.25 kbps and read from the buffer at 52.083 kbps.

The flowchart for the receiver executive as implemented on the AD-21020 is shown in Figures 5.14, 5.15, 5.16, 5.17 and 5.18. The two memory buffers for the double buffer are implemented as two separate blocks of data memory on the AD-21020 processor. The AD-21020 is interrupted by the on chip timer interrupt every 6.4 microseconds. The timer interrupt service routine (ISR) is then activated. The timer ISR consists of a transmit ISR and a receive ISR block, only one of which is active at any instant of time. The receive timer ISR carries out all the tasks required for the receive executive. Before the beginning

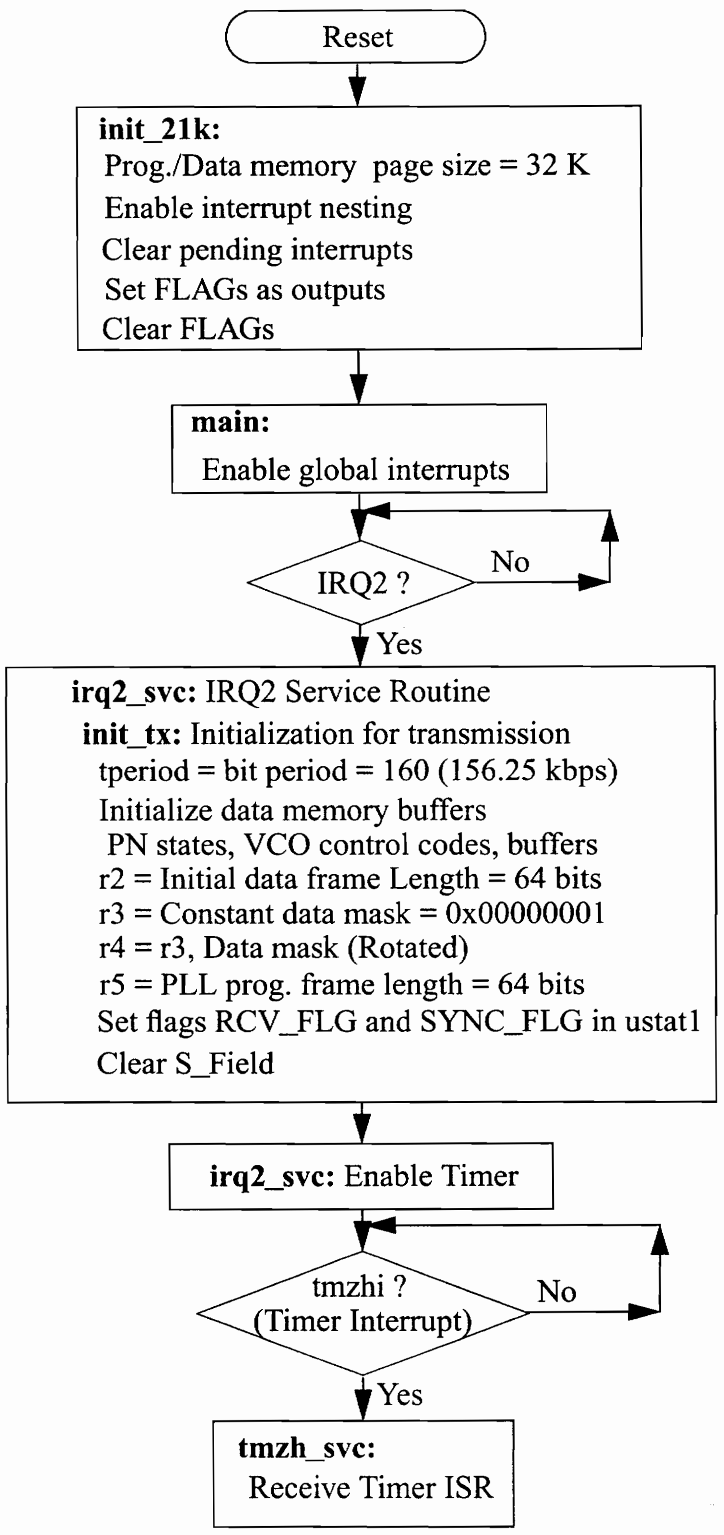

Reset
init_21k:
Prog./Data memory page size = 32 K
Enable interrupt nesting
Clear pending interrupts
Set FLAGs as outputs
Clear FLAGs
main:
Enable global interrupts
IRQ2 ?
No
Yes
irq2_svc: IRQ2 Service Routine
init_tx: Initialization for transmission
tperiod = bit period = 160 (156.25 kbps)
Initialize data memory buffers
PN states, VCO control codes, buffers
r2 = Initial data frame Length = 64 bits
r3 = Constant data mask = 0x00000001
r4 = r3, Data mask (Rotated)
r5 = PLL prog. frame length = 64 bits
Set flags RCV_FLG and SYNC_FLG in ustat1
Clear S_Field
irq2_svc: Enable Timer
tmzhi ?
(Timer Interrupt)
No
Yes
tmzh_svc:
Receive Timer ISR

Figure 5.15 Flowchart for the Receive Executive

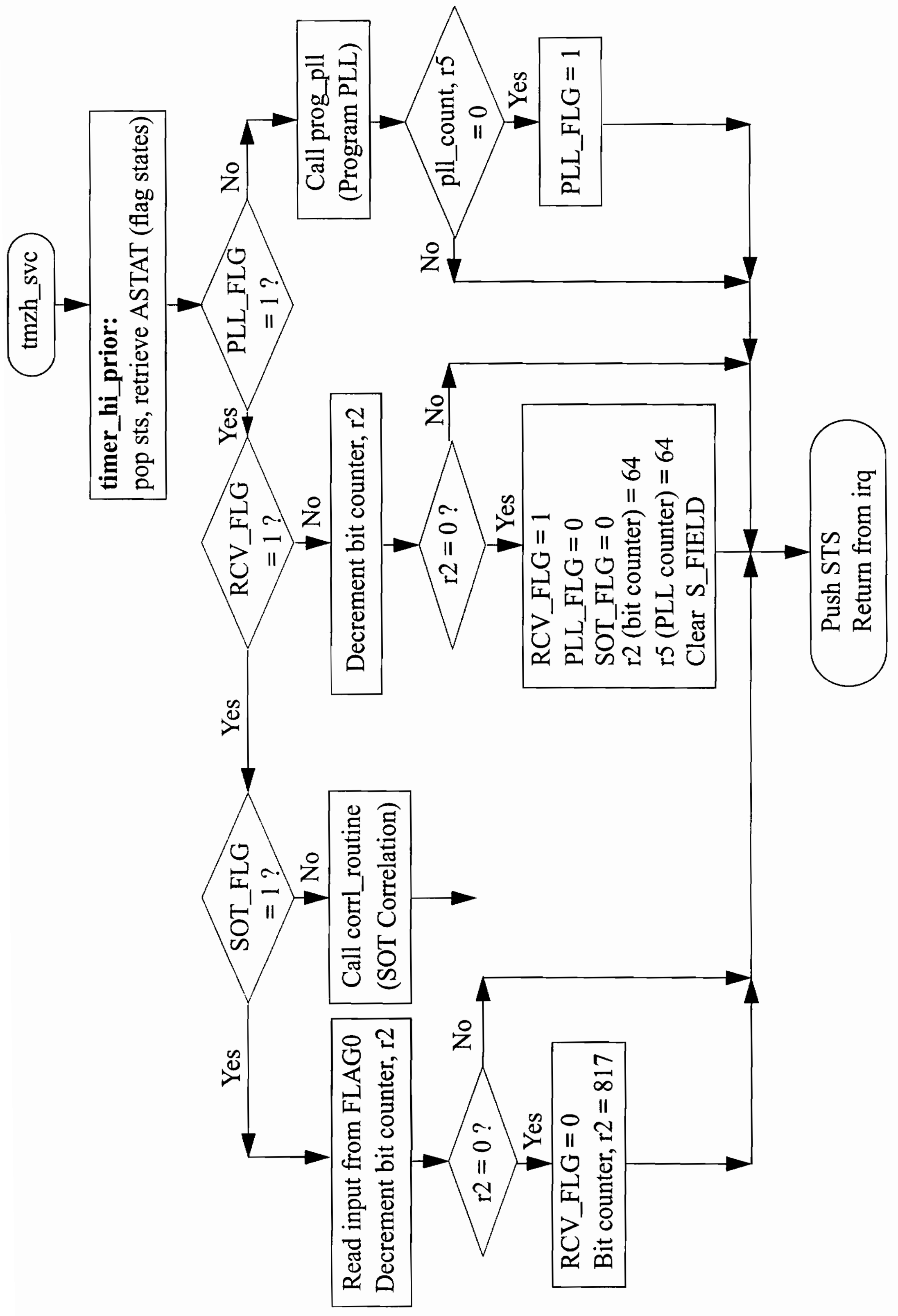
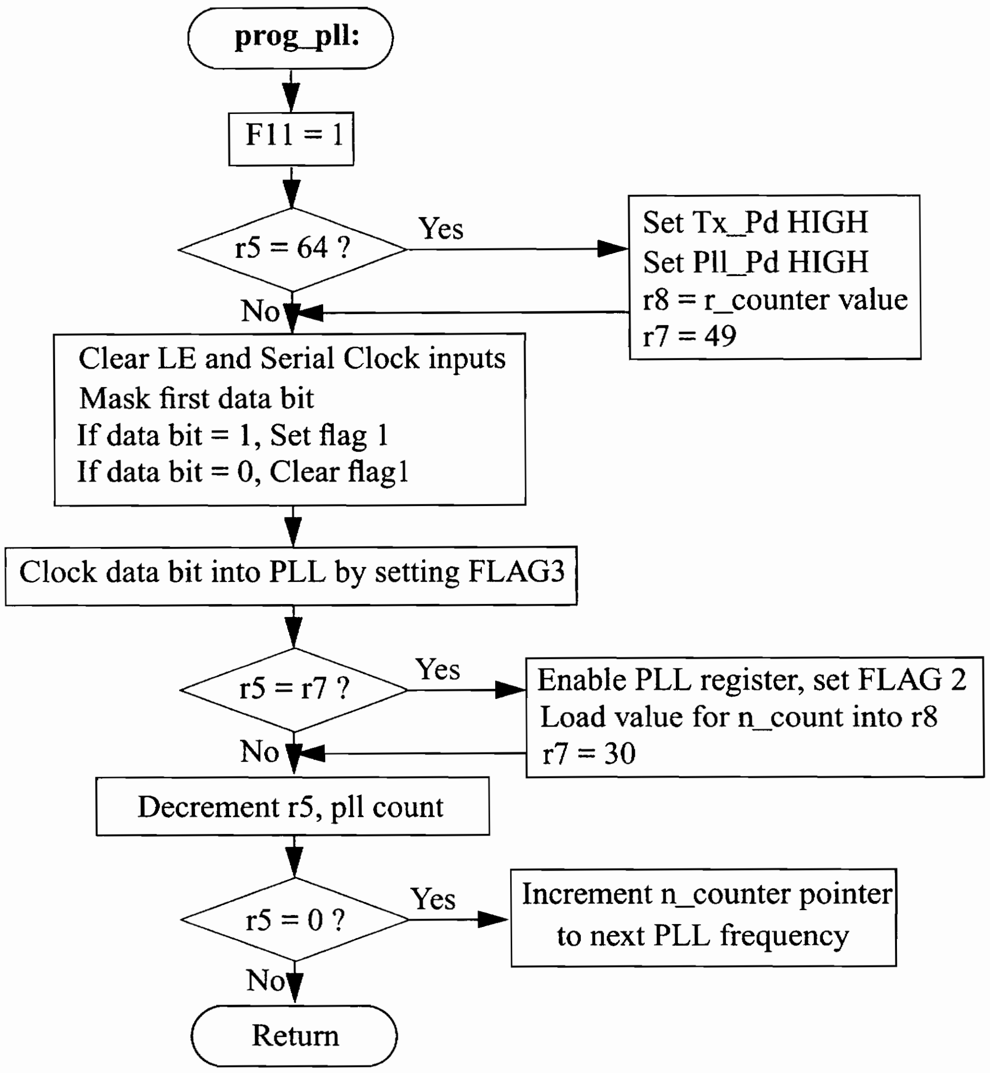


Figure 5.16 Flowchart for the Timer ISR of the Receive Executive


prog_pll:
F11 = 1
r5 = 64 ?
Yes
No
Set Tx_Pd HIGH
Set Pll_Pd HIGH
r8 = r_counter value
r7 = 49
Clear LE and Serial Clock inputs
Mask first data bit
If data bit = 1, Set flag 1
If data bit = 0, Clear flag1
Clock data bit into PLL by setting FLAG3
r5 = r7 ?
Yes
No
Enable PLL register, set FLAG 2
Load value for n_count into r8
r7 = 30
Decrement r5, pll count
r5 = 0 ?
Yes
No
Increment n_counter pointer to next PLL frequency
Return

r5 = PLL programming bit counter

r7 = Holds pll bit count for end of register data

r8 = Value to be programmed into PLL

Figure 5.17 Flowchart for the PLL Programming Block

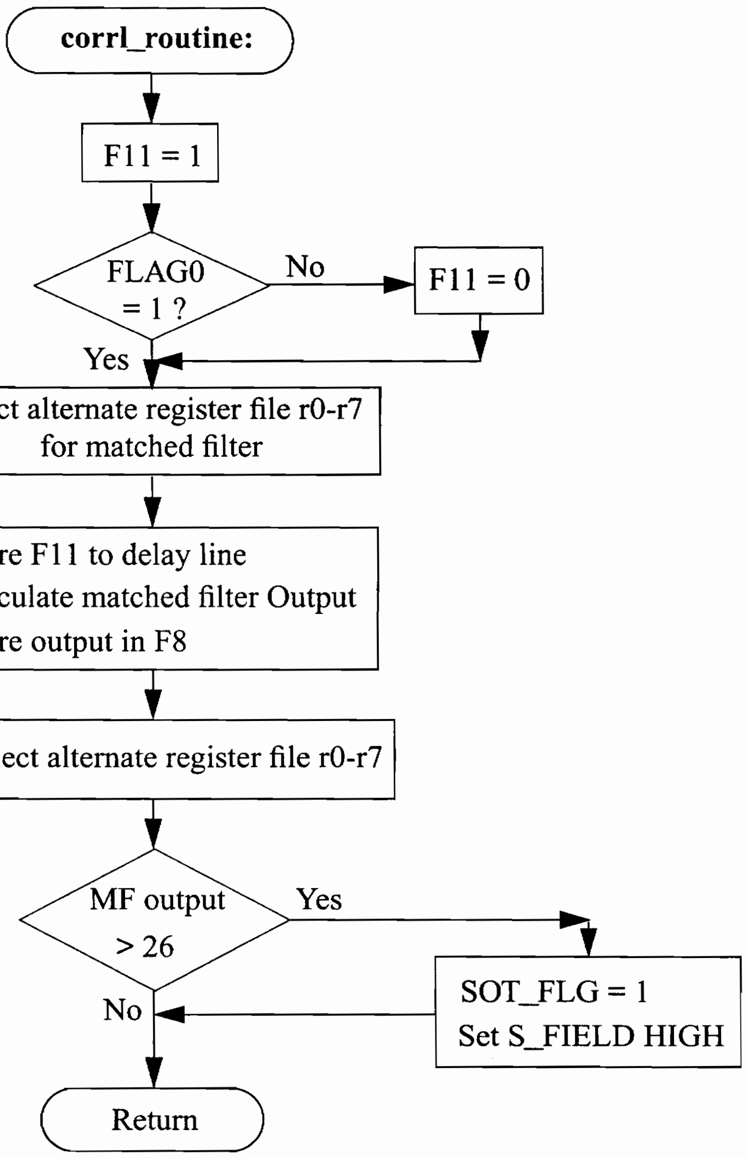


Figure 5.18 Flowchart for the Synchronization Block - Algorithm 1

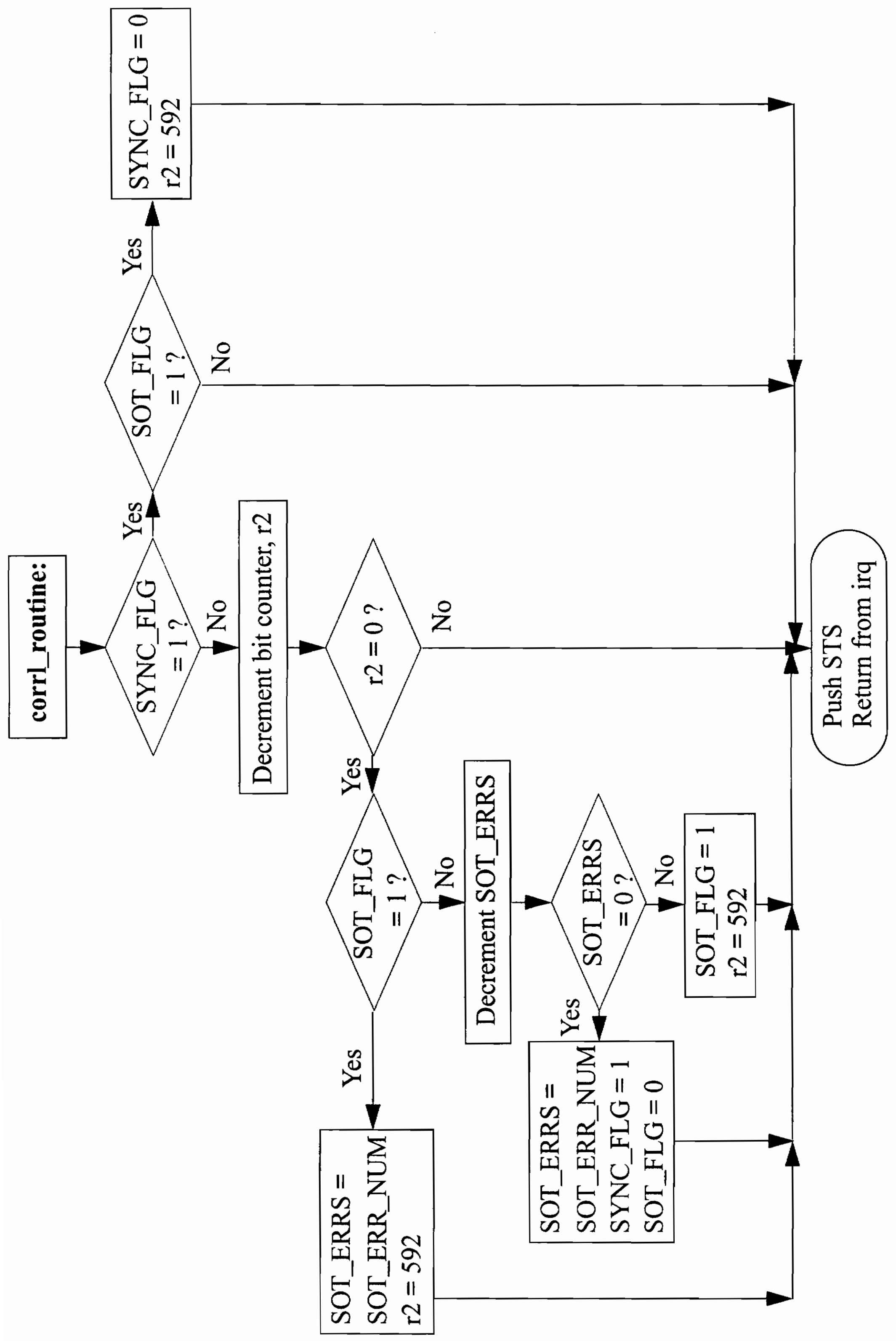


Figure 5.19 Flowchart for the Synchronization Block - Algorithm 2

of each receive frame, the PLL is programmed to the appropriate carrier frequency, the antenna output connected to the LNA/Mixer, and the output of the frequency synthesizer is connected to the mixer. The appropriate words for the *R* and *N* counters for the various hop frequencies are stored in internal data memory. The PLL is programmed during the guard interval of 65 bit periods. It takes 16 bit periods to program the *R* counter and 19 bits to program the *N* counter. The remaining 30 bit periods of the guard interval are used to allow the PLL to settle down. Once these operations are complete, the synchronization software suite is activated. When acquisition has been achieved, the acquisition sub-block of the synchronization suite is deactivated, and the deformatter and the data buffer software suites are activated and remain in operation untill the end of the receive frame. At the end of the receive frame, the flag which is used to decide which timer ISR (transmit or receive) the DSP processor is going to execute is toggled to activate the transit timer ISR.

Chapter 6

Simulation Results

6.1 Introduction

This chapter describes the simulation procedure to test the FH system and the results obtained. The performance of the FH transceiver as implemented with the two different synchronization algorithms is compared to the performance of a single slot DECT trans ceiver. This chapter discusses the performance criteria and the channels used. All the simulations were done using the SPW models that were developed, and were performed on SUN Microsystems SPARC 10/20.

6.2 FH Transceiver Simulation

The FH transceiver have been extensively simulated in SPW and their operation verified. This section details the results of the simulations of the operation of the transceiver. Figure 6.1a shows the baseband GMSK signal at the transmitter. The demodulated GMSK signal is shown in Figure 6.1b. The signal to noise ratio (SNR) is 17 dB for these plots. The spectrum of the transmitted baseband GMSK signal is shown in Figure 6.2a. The x-axis for this plot is normalized with respect to the sampling frequency used in the

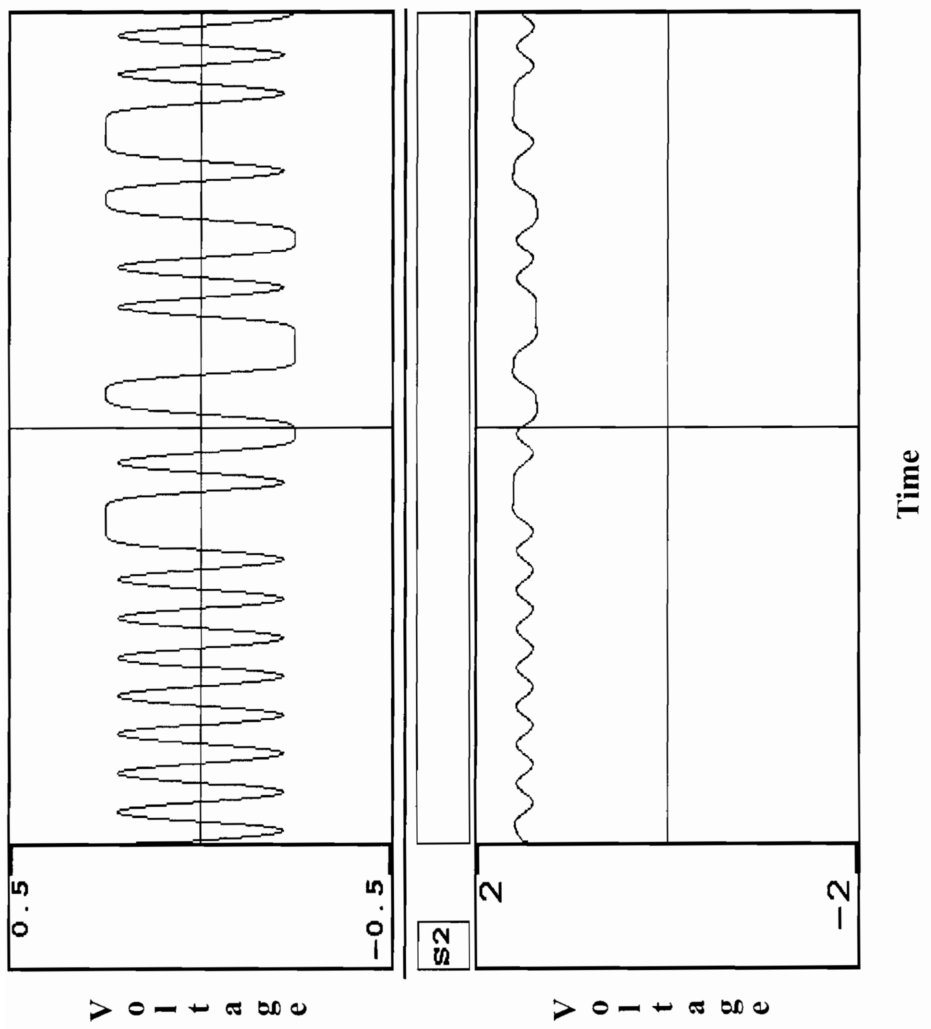


Figure 6.1 (a) Transmitted GMSK signal
(b) Demodulated Signal

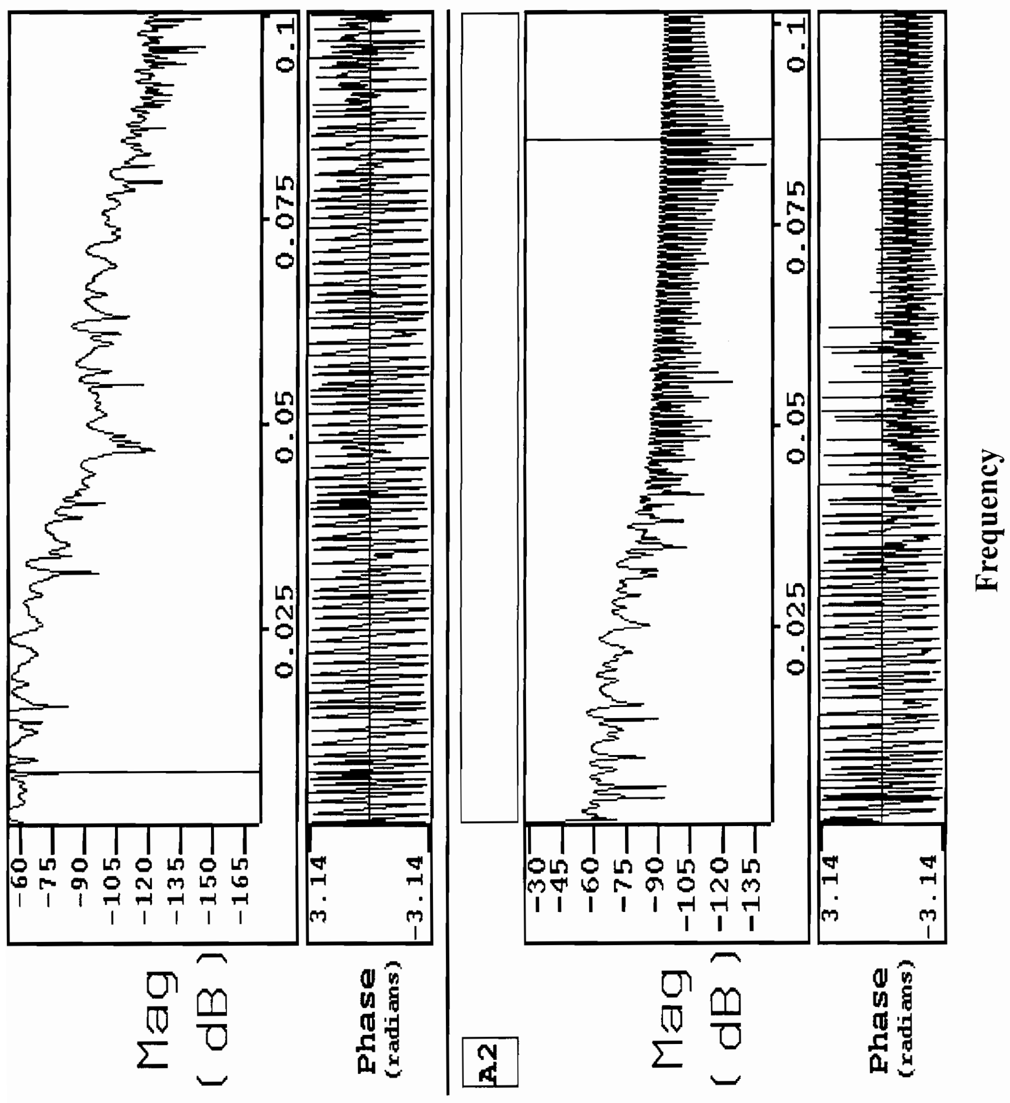


Figure 6.2 (a) Spectrum of the Transmitter GMSK Signal
(b) Spectrum of the Demodulated Signal

simulation which is 2.1 MHz. Note that the -20 dB point occurs at about 75 kHz. Figure 6.2b shows the spectrum of the demodulated GMSK signal. The eye diagram of the gaussian filtered pulse is shown in Figure 6.3a. It can be seen from the figure that there is a very slight closing of the eye. The eye diagram of the demodulated GMSK signal is shown in Figure 6.3b. It can be seen that the eye has closed a bit more due to the fact that the bandwidths and the characteristics of the filters used in the receiver are not optimal. Note the jitter introduced. Figure 6.4b shows the eye diagram of the demodulated signal in the frequency selective channel characterized by equation (6.4). It can be seen that the eye has closed even more than for the AWGN channel and is asymmetric. Timing jitter has also increased.

Figure 6.5a shows the frequency synthesizer control signal at the FH transmitter. Each voltage value represents a particular carrier frequency. It can be seen that a linear hopping pattern with five different hopping channels is used for this particular simulation. The operation of version 1 of the FH synchronization algorithm can be seen in Figure 6.5b. This example is shown for the frequency selective channel characterized by equation (6.4). It can be seen from the figure that synchronization lock is lost at the sixth hop and is not reacquired until the sixteenth hop. The FH receiver remains camped at the frequency where it lost lock until it reacquires synchronization lock as can be see in the figure. Figure 6.6a shows the output of the FH acquisition matched filter for the same frequency selective channel. The matched filter output does not cross the detection threshold from the sixth hop till the fifteenth hop. The effect of the frequency selective channel on the different hop frequencies can be seen in Figure 6.6b which shows the demodulated signal for different hop frequencies. It can be seen that the demodulated signal envelope is different for different hop frequencies.

Figures 6.7, 6.8 and 6.9 show the operation of the timing recovery system. It can be seen from Figure 6.7a which shows the spectrum of the reshaped demodulated signal that there is no strong discrete spectral line at the data rate of 100 kbps, which corresponds to

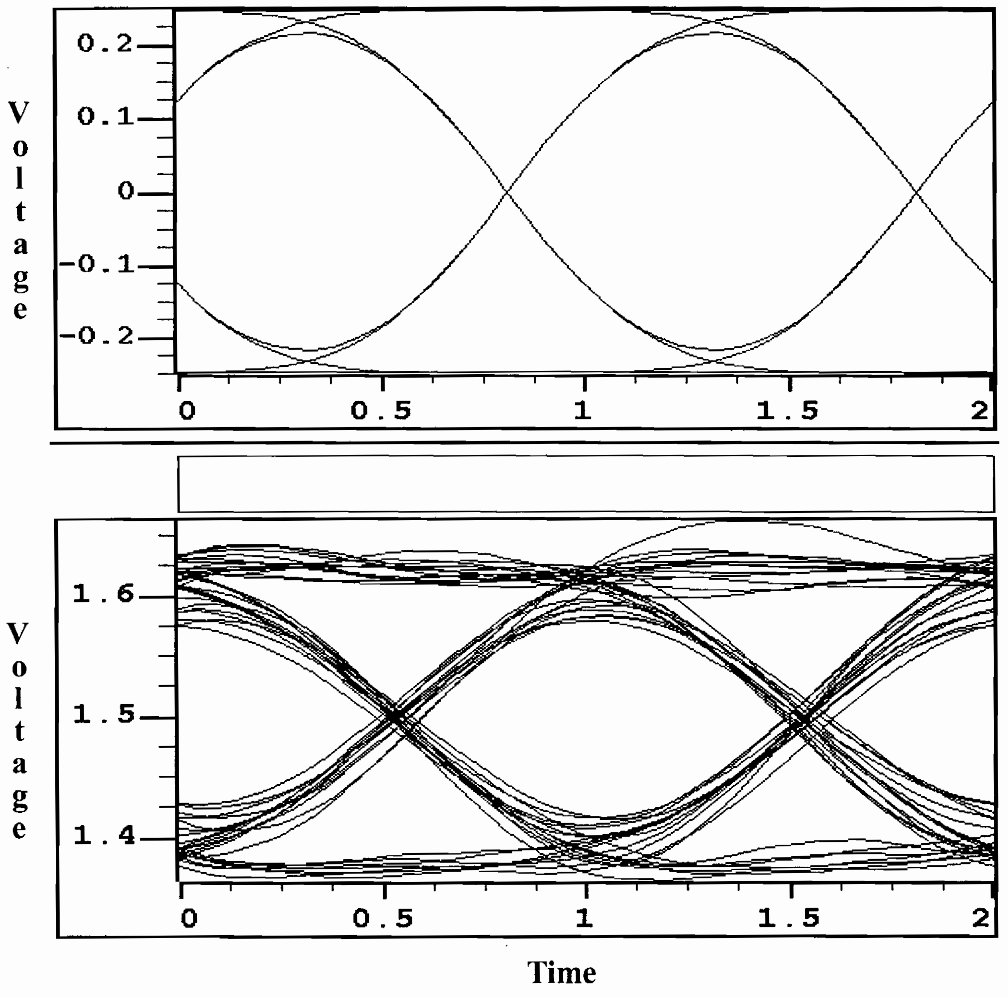


Figure 6.3 (a) Eye Diagram at Transmitter
(b) Eye Diagram of the Demodulated Signal

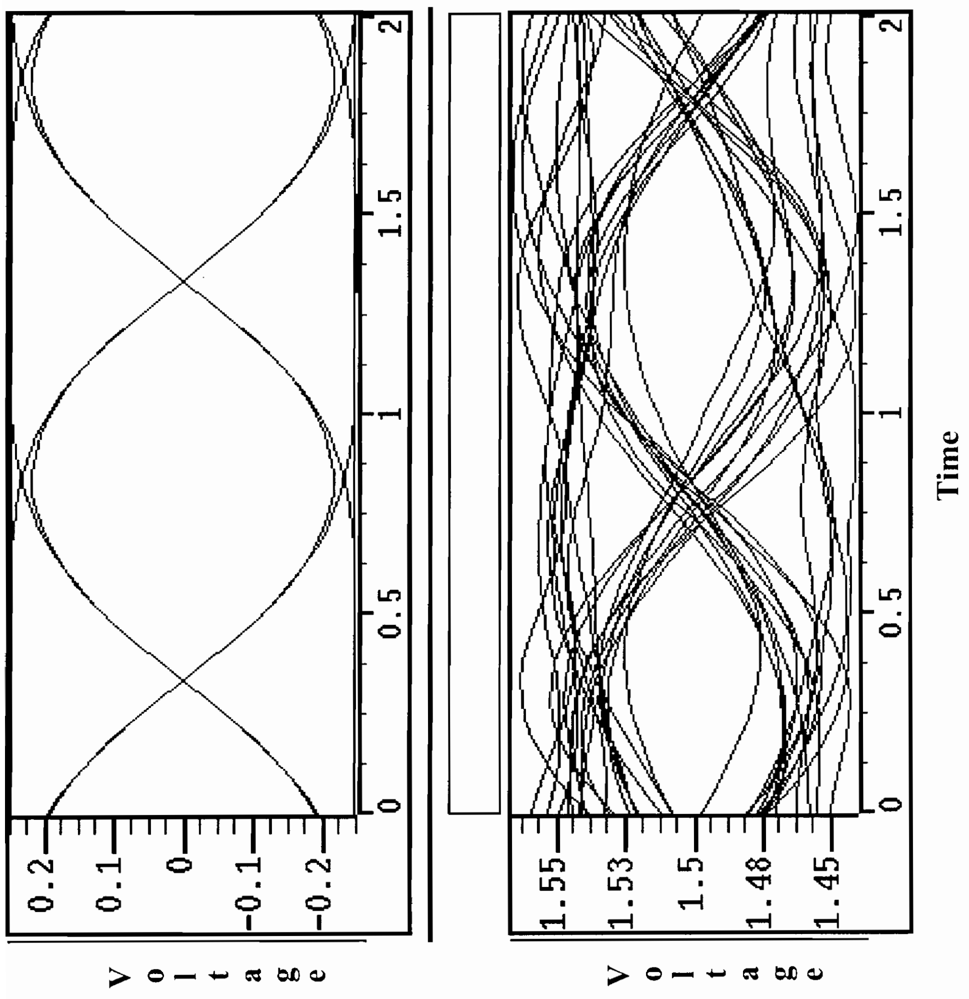


Figure 6.4 (a) Eye Diagram at Transmitter
(b) Eye diagram of the Demodulated Signal in the Frequency Selective Channel

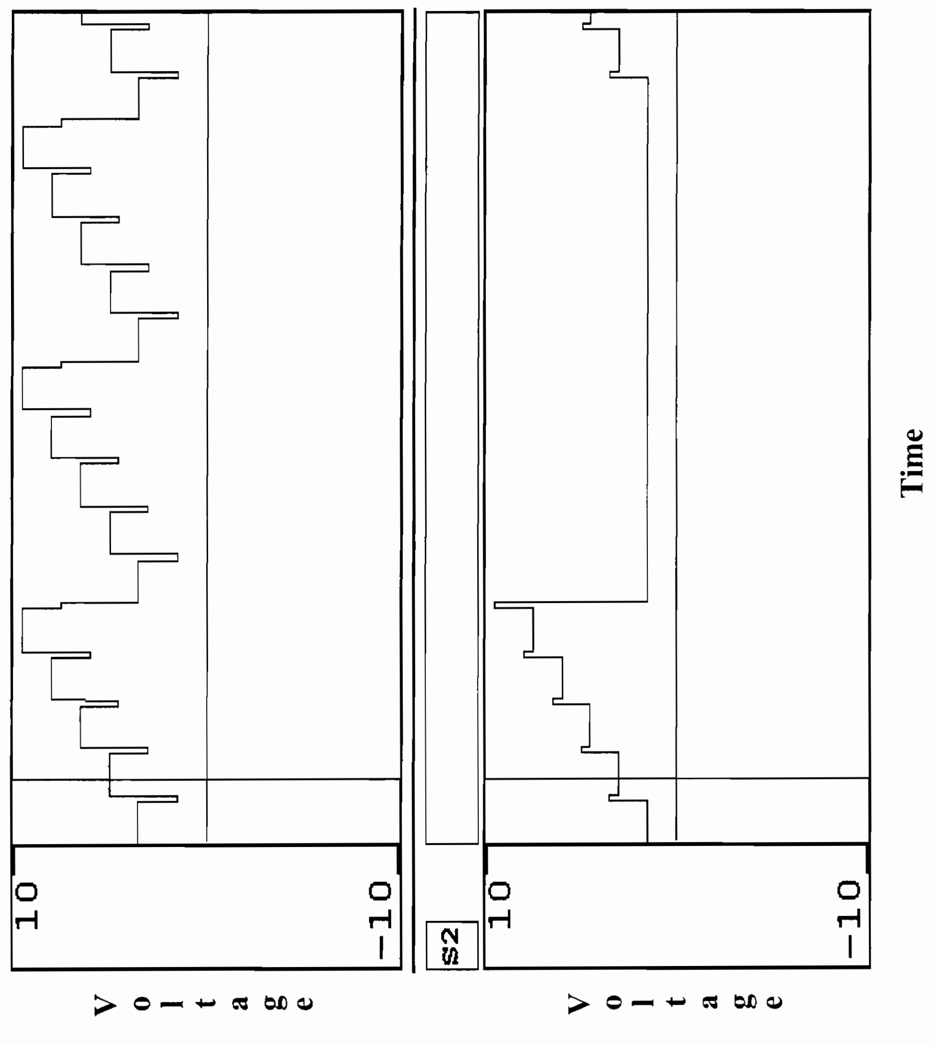


Figure 6.5 (a) Frequency Synthesizer Control Signal for the FH Transmitter
(b) Frequency Synthesizer Control Signal for the FH Receiver

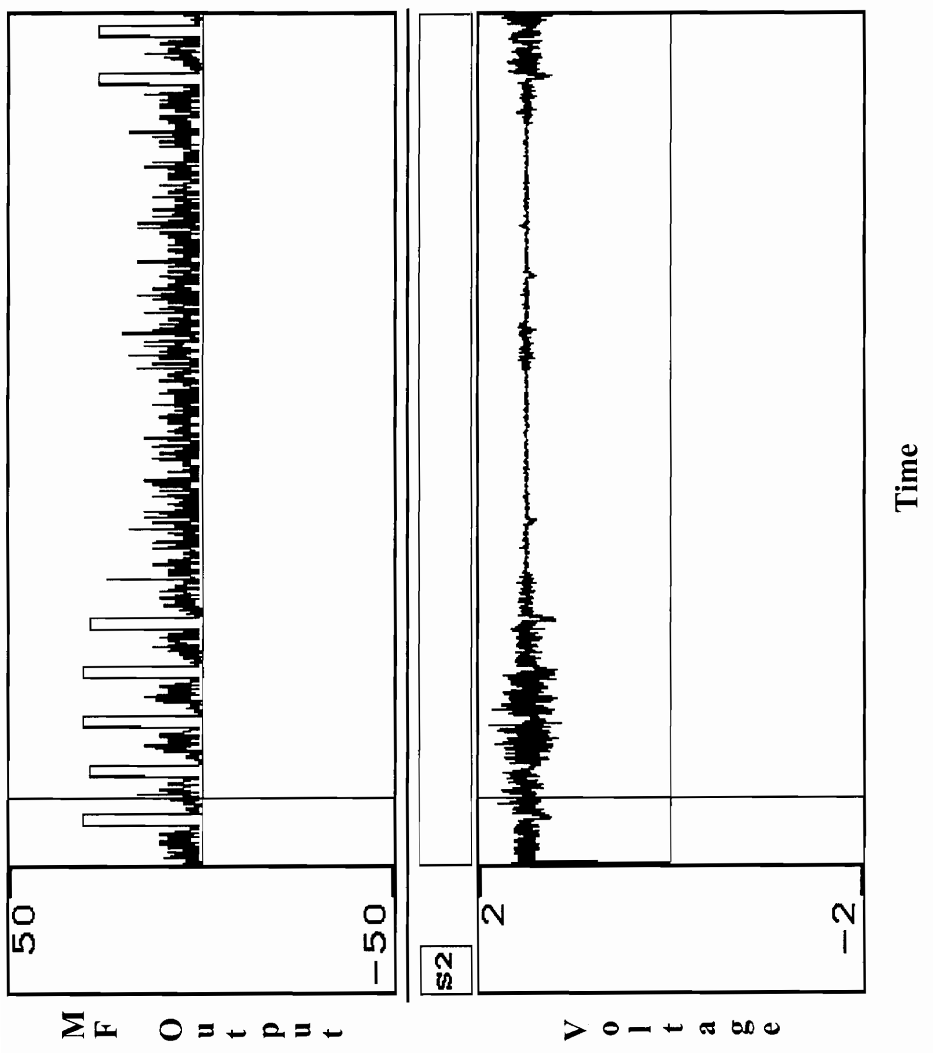


Figure 6.6 (a) Matched Filter Output at the FH Receiver
(b) Demodulated Signal in the Frequency Selective Channel

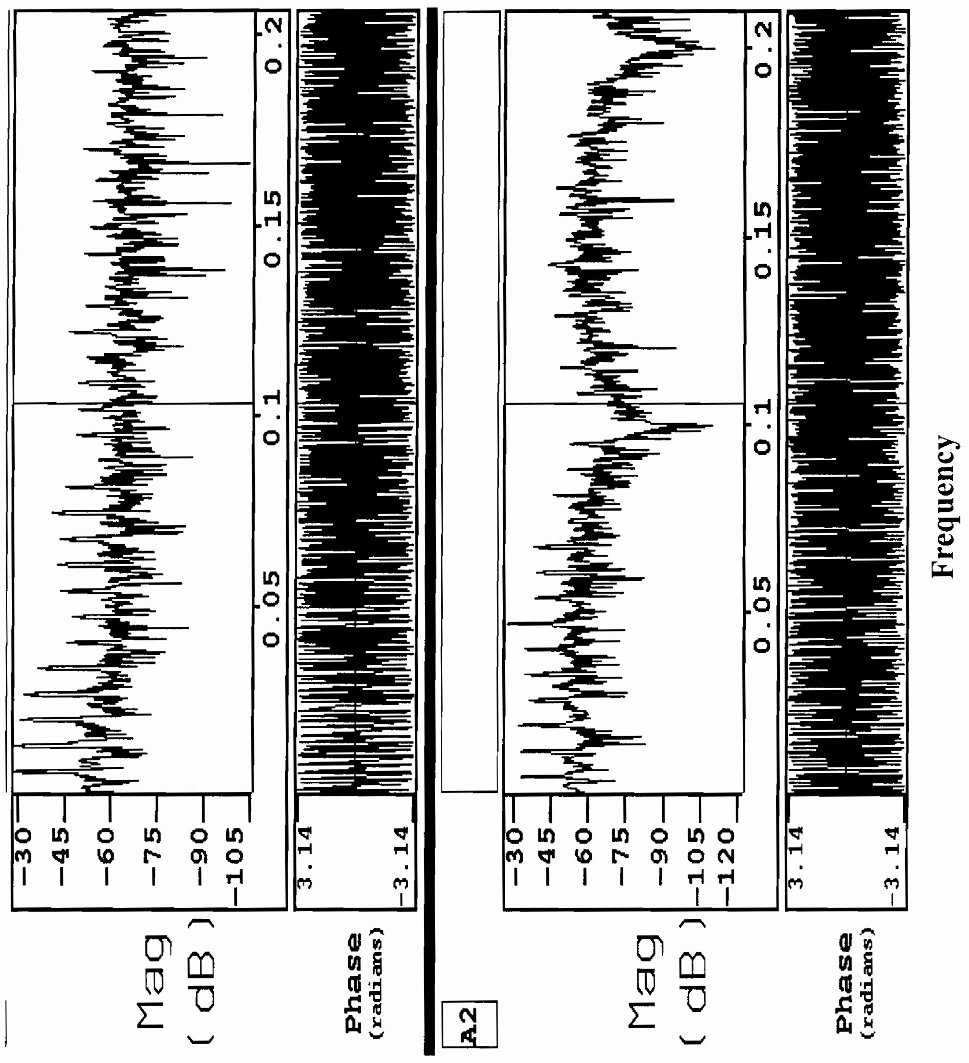


Figure 6.7 (a) Spectrum of the Reshaped Demodulated Signal
(b) Spectrum of the Signal After the Delay and Multiply

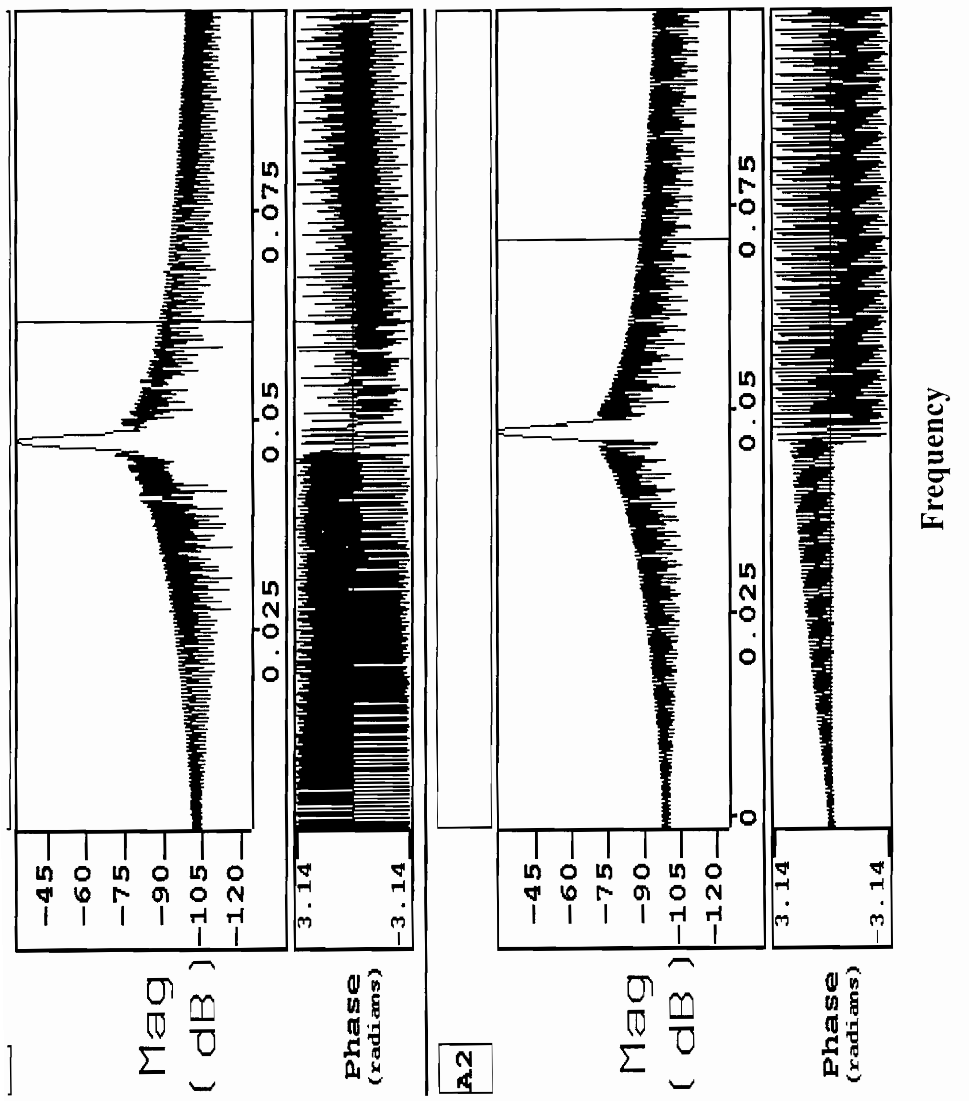


Figure 6.8 (a) Spectrum of the Signal After the Bandpass Filter
(b) Spectrum at the Output of the Timing Recovery PLL

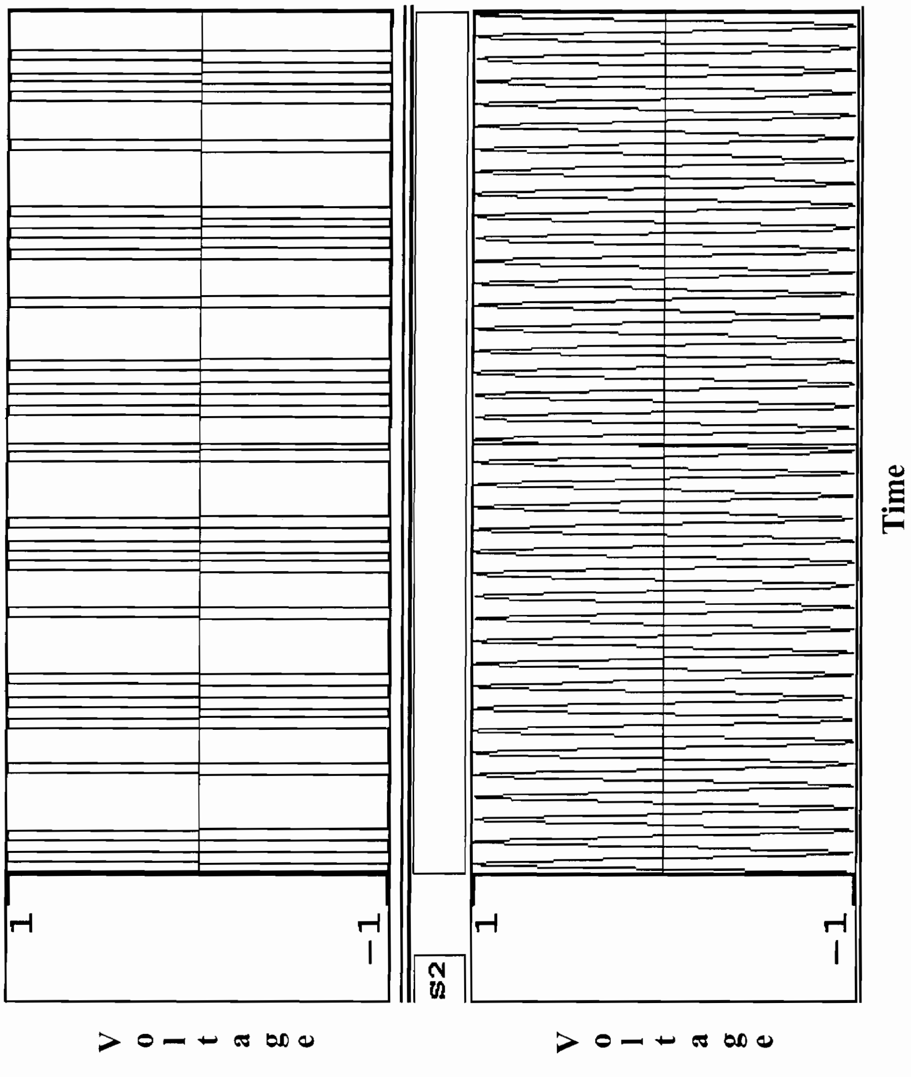


Figure 6.9 (a) Signal at the Output of the Delay and Multiply
(b) Output of the Timing Recovery PLL

the 0.047 point on the x-axis of the plot. The x-axis of the plot is normalized with respect to the sampling frequency. The output of the delay and square block has a strong discrete spectral line at the data rate as can be from Figure 6.7b. The outputs of the bandpass filter and the PLL which are used to extract and track the recovered clock are shown in Figures 6.8a and 6.8b. Figure 6.9a shows the time domain plot of the signal at the delay and multiply block output. Figure 6.9b shows the output of the PLL used in the timing recovery system.

6.3 Performance Measures

The main performance measures used in the performance evaluation of the FH transceiver are the frame error rate (FER), the bit error rate in detected slots (BERDS) and the bit error rate in all slots (BERAS), and are discussed in detail below:

6.3.1 Frame Error Rate

The FER is defined as

$$\text{FER} = \frac{\text{Number of frames detected}}{\text{Total number of frames transmitted}} \tag{6.1}$$

The frame error rate is an important performance because in existing DECT systems, a frame is discarded if an error is detected in the CRC checksum.

6.3.2 Bit Error Rate in Detected Slots (BERDS)

The BERDS [34] is defined as

$$\text{BERDS} = \frac{\text{Number of errors in detected slots}}{\text{Total number of bits in the detected slots}} \tag{6.2}$$

The BERDS is useful as it is a direct measure of the quality of the link since slots which contain CRC errors are discarded and these frames are muted.

6.3.3 Bit Error Rate in All Slots (BERAS)

The BERAS [34] is defined as

$$\text{BERAS} = \text{BERDS} * (1\text{-FER}) + 0.5 * \text{FER} \qquad (6.3)$$

Although the FER and the BERDS are good performance measures, there is a need to establish a link between the FER and the overall BER. This is done with the BERAS. The BERAS should be more directly related to the overall link quality.

6.4 Channels Used

The performance of the DECT and the FH system have been simulated for two channels. The channels used in the simulation are the additive white gaussian noise (AWGN) channel and a static frequency selective channel. The static frequency selective channel used is:

$$H(z) = 1 + 0.4z^{-3} \qquad (6.4)$$

where z^{-3} represents a *3* sample delay (the number of samples per data symbol is 21). A similar two path profile is used in [35]. A delay of *3* samples represents a delay of 124 nanoseconds for DECT. It has been reported in literature [36] that the performance of DECT will start to be affected by dispersion when the delay is about 50 nanoseconds and will be significantly affected for delays greater than about 100 nanoseconds. The maximum delays encountered in a typical indoor environment for DECT is around 200 nanoseconds [37]. The delay of 124 nanoseconds is thus sufficient to characterize the dispersion that would degrade DECT system performance. The magnitude response of this channel is shown in Figure 6.10. As seen from Figure 6.1, this channel has a single fade centered at 350 kHz. 350 kHz is the center frequency for one of the FH channels. Therefore only one of the five FH channels would be in a fade. Since the simulations are carried out with the system bandwidth used in the simulation being scaled down from 26 MHz to 1.05 MHz, a single frequency notch in a bandwidth of 1.05 MHz is representative of the effect of a typical frequency selective channel in a 26 MHz bandwidth. A static channel is used. It enables us to determine the maximum performance gain possible with frequency hopping, and is also typical of the situation for which the system is designed for. It should be noted that the static frequency selective channel is used here to highlight the performance gains possible with frequency hopping and is not representative of the performance of the FH system in all the types of channels where the system might be used. More detailed simulations can be carried out making use of channels such as those generated by the Simulation of Indoor Radio Channel Impulse response Model (SIRCIM) software package [38].

6.5 Simulation Results

6.5.1 AWGN Channel

The parameters used for the simulation are the same as those given for the FH system in Chapter 4. White Gaussian noise with the desired variance is generated by the SPW white gaussian noise generator block. The same detection threshold of 26 out of 31 bits in the synchronization sequence is used for all the three systems. Figures 6.11, 6.12 and 6.13 show the BERDS, BERAS and FER performance of the DECT system and the FH system implemented with the two different synchronization algorithms. Although the BERDS performance of the three systems are similar, their FER performance is quite different. It is seen from Figure 6.13 that the FH system implemented with version 1 of the synchronization algorithm has the poorest performance of the three. This is due to the fact the system would lose synchronization lock if it does not detect the synchronization sequence in each burst. The system would remain out of lock for another 4 (*N-1*) hops. This increases the FER at low SNR where more errors occur. It is also seen that there is a crossover point after which the FH system has better performance. This result highlights the importance of a robust FH synchronization scheme. This disparity in the FER performance is reflected in the BERAS performance of the three systems, where it is seen from Figure 6.12 that the FH system implemented with version 1 of the synchronization algorithm has the poorest performance. The DECT transceiver has very similar performance to the FH transceiver implemented with version 2 of the synchronization algorithm. It should be noted that the FER performance of the systems depend upon the threshold used for detecting the synchronization sequence and in the case of the FH transceiver implemented with version 2 of the synchronization algorithm, also depends on the number of consecutive undetected frames over which the algorithm can still maintain synchronization lock.

6.5.2 Static Frequency Selective Channel

The parameters used for the simulation are the same as those used for the simulations in the AWGN channel. The channel is modeled based on equation (6.4). Figures 6.14, 6.15 and 6.16 show the BERDS, BERAS and FER performance of the DECT system and the FH system implemented with the two different synchronization algorithms. The BERDS performance of both the FH systems are better than the DECT system since the FH systems use channels that are not affected. The difference in the FER performance between the two FH systems is again apparent in Figure 6.16. It is worth noting that the FER performance of the DECT system and version 1 of the FH system are almost identical whereas for the AWGN channel there is a crossover point below which the DECT system performs better than version 1 of the FH system. This again highlights the need for a robust FH synchronization algorithm, without which the potential performance gains possible would not be achieved. It can also be seen that the SNRs required to achieve FER performance similar to that in the AWGN channel are now higher. This is mainly due to the closing of the received eye caused by the intersymbol interference (ISI) introduced by the multipath channel. From Figure 6.15, the advantages of the FH system in a frequency selective channel are obvious. A gain of about 4 dB in the BERDS performance at low SNRs is obtained with frequency hopping for this particular frequency selective channel. This gain decreases to about 1.5 dB at medium SNRs. The gains are even more impressive with the BERAS figures: from about 4.5 dB at low SNRs to about 2.2 dB at medium SNRs. The FH transceiver performs better because FH provides a form of frequency diversity through exploitation of frequency selectivity over the given bandwidth. The fading process is usually decorrelated from hop to hop over the duration of long fades and the FH transceiver by sampling a number of channels achieves better performance as only a small fraction of these channels is affected by the multipath.

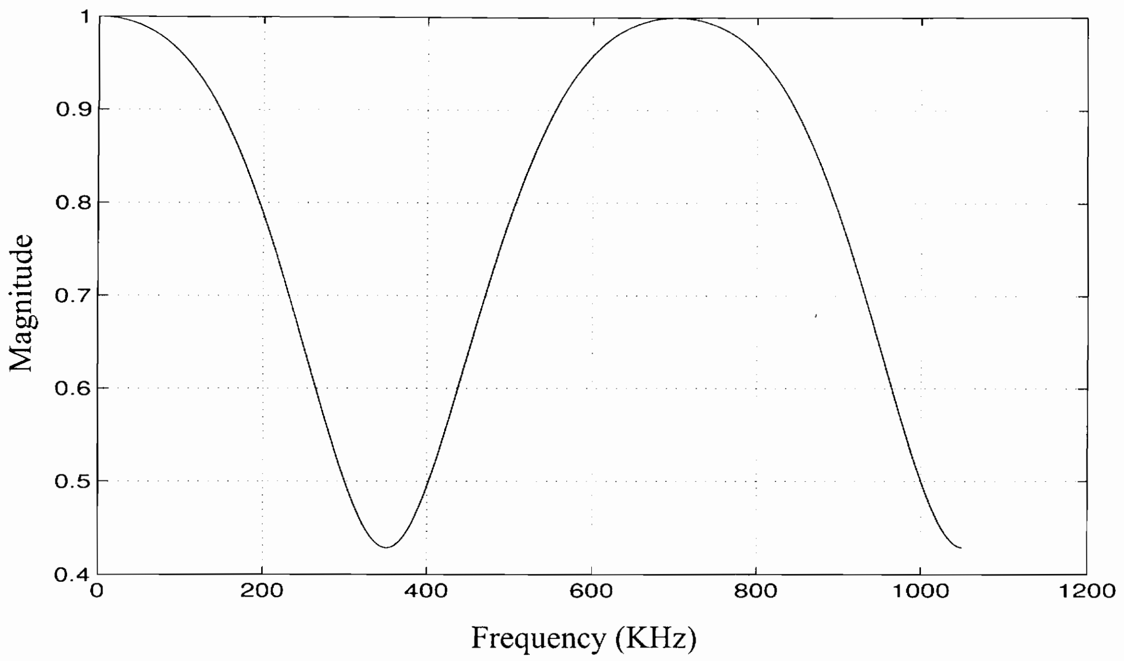


Figure 6.10 Magnitude Response of the Frequency Selective Channel

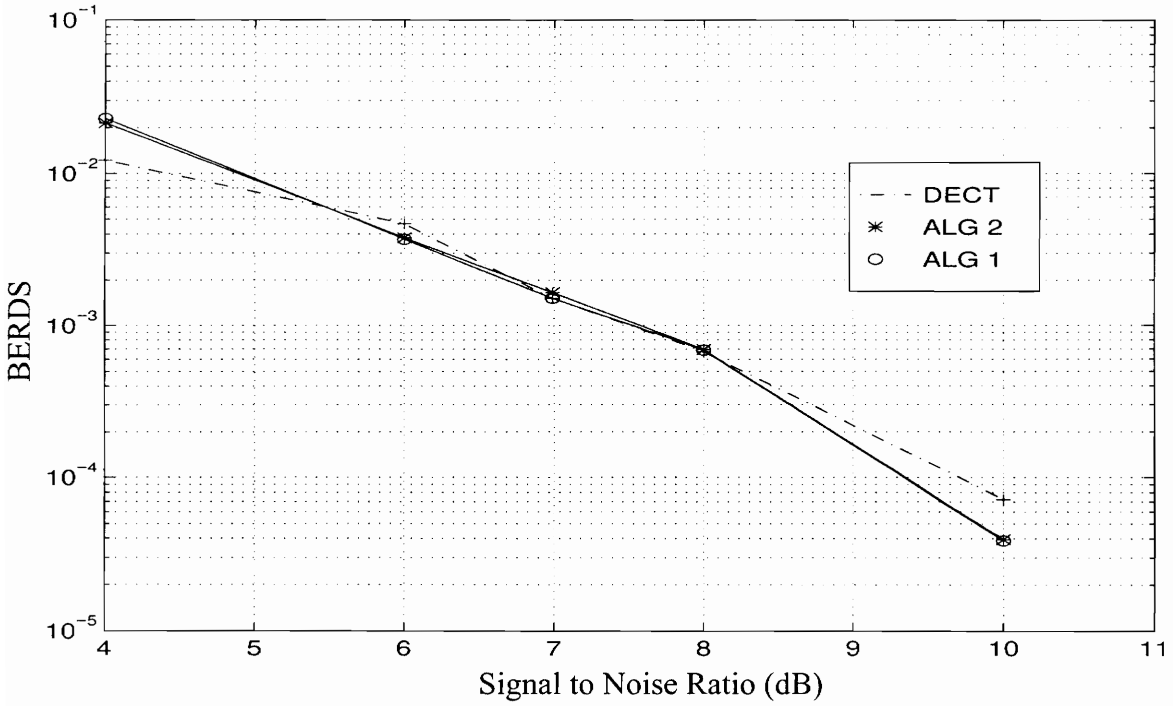


Figure 6.11 BERDS for the AWGN Channel

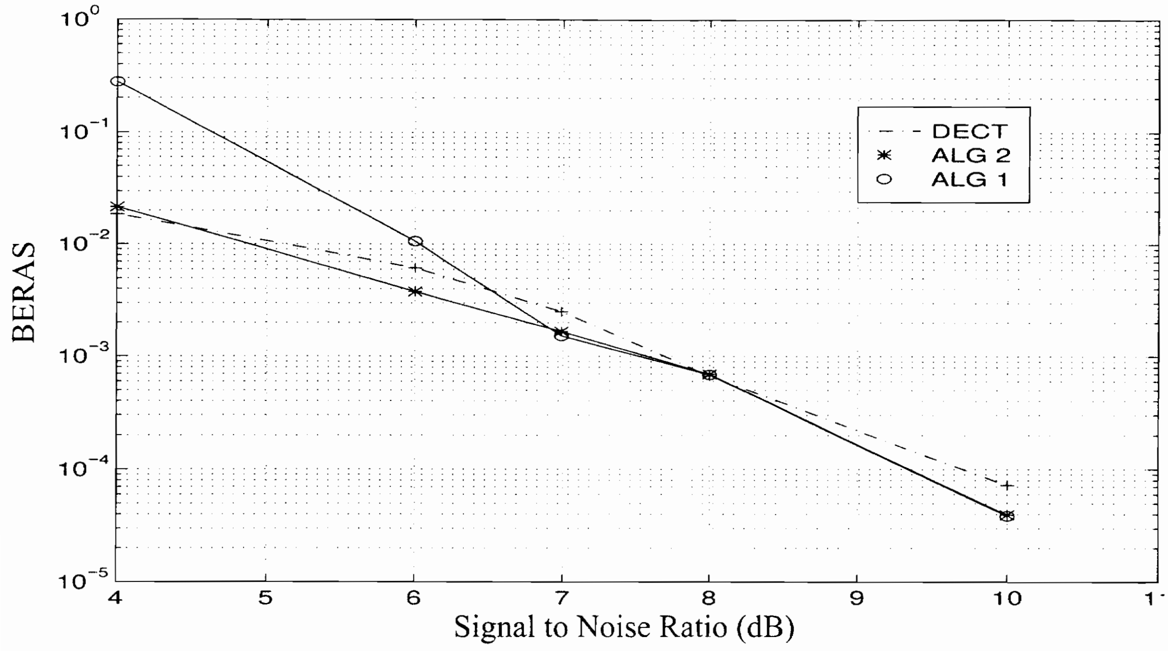


FIgure 6.12 BERAS for the AWGN Channel

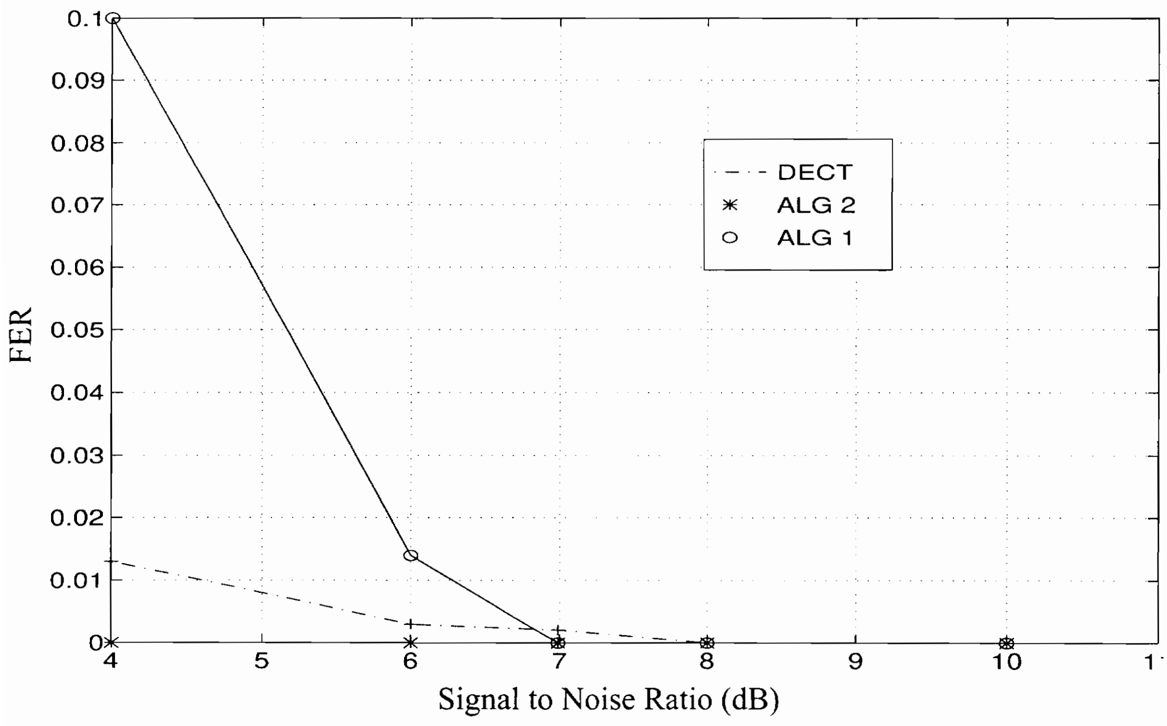


Figure 6.13 FER for the AWGN Channel

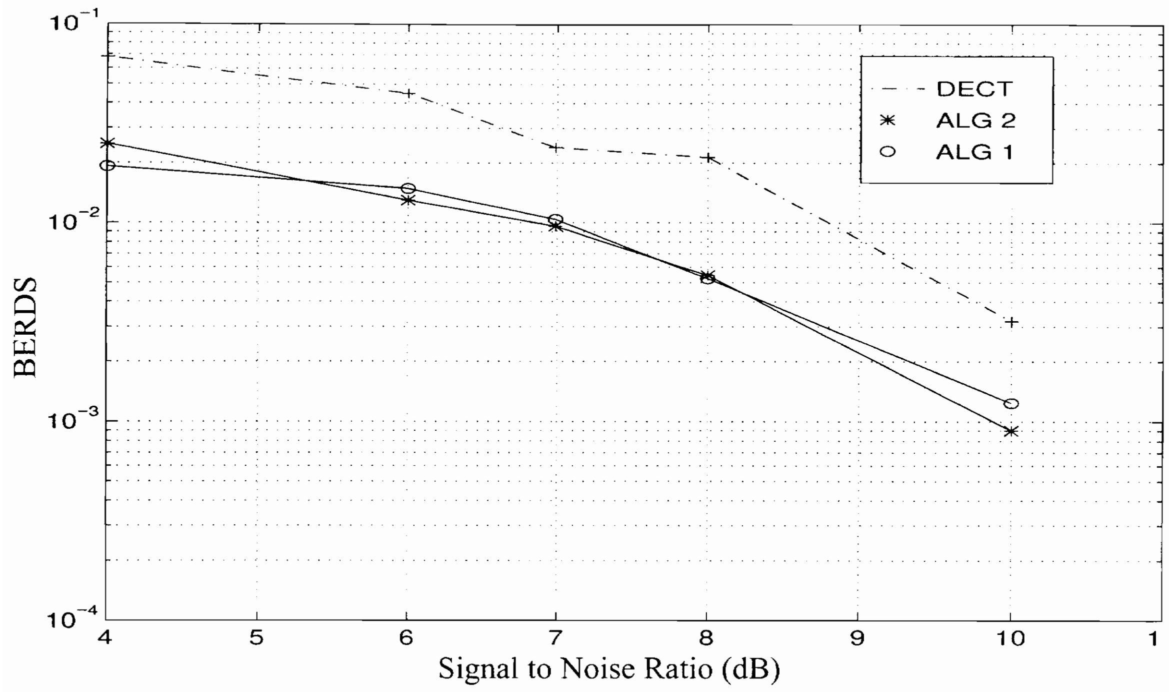


Figure 6.14 BERDS for the Frequency Selective Channel

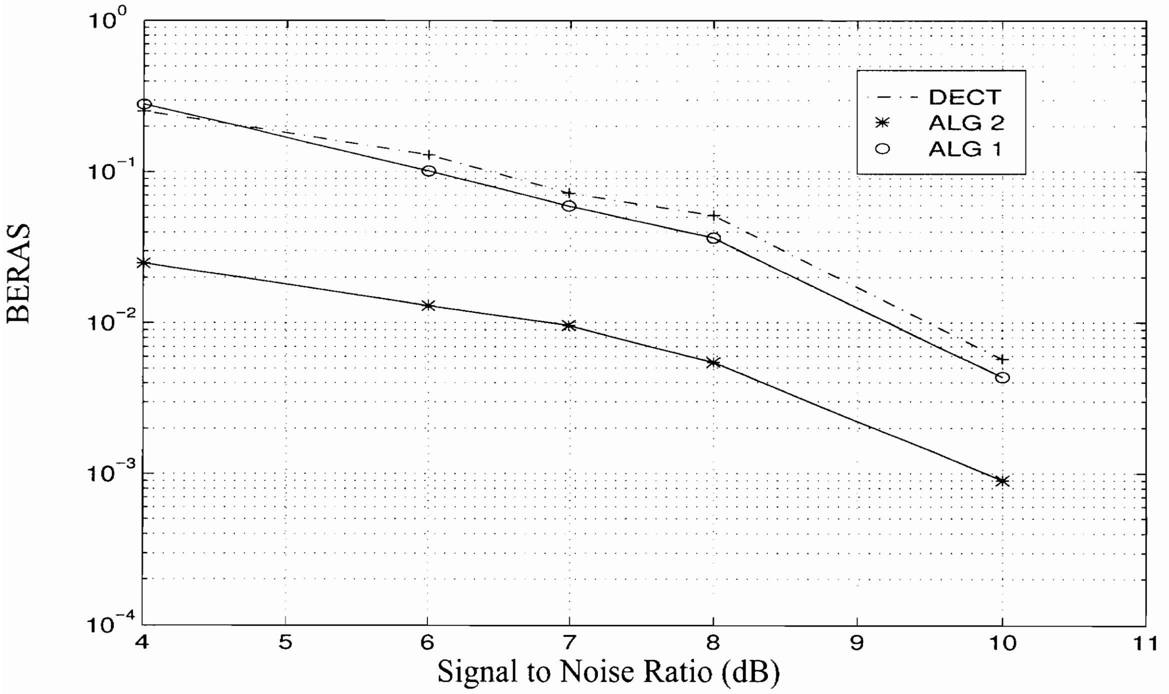


Figure 6.15 BERAS for the Frequency Selective Channel

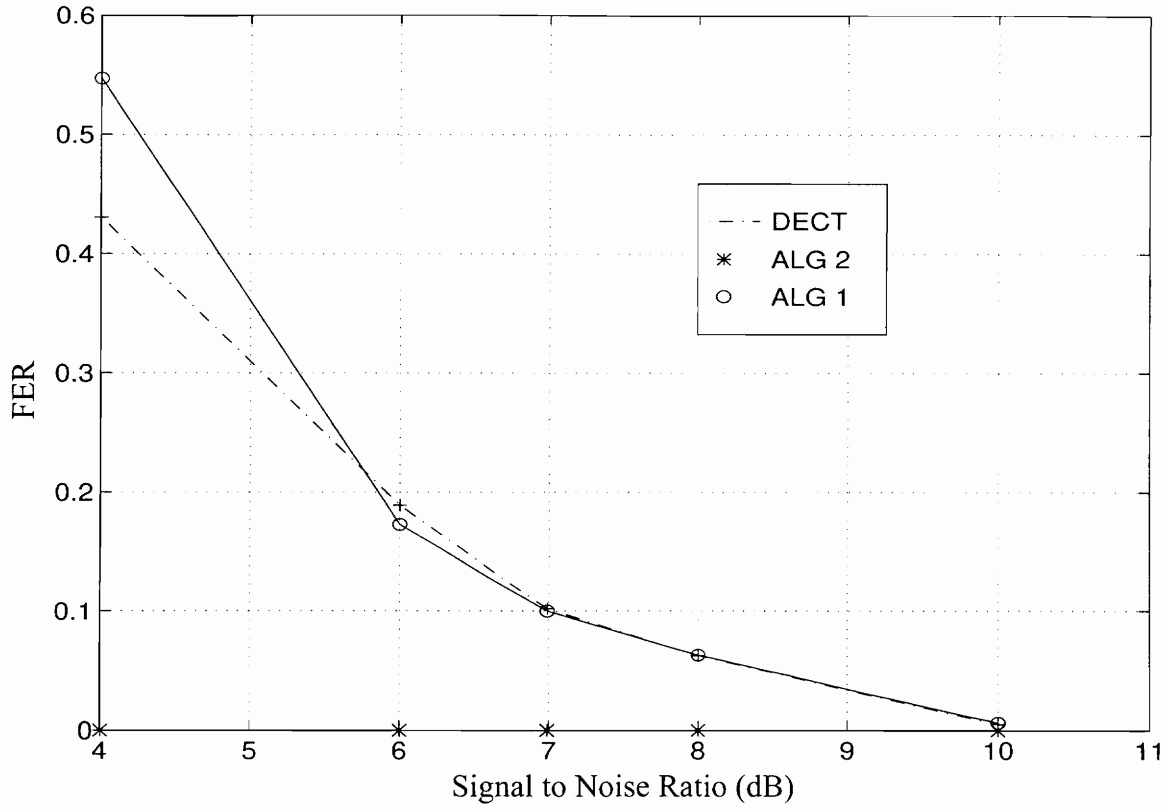


Figure 6.16 FER for the Frequency Selective Channel

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has proven that an existing DECT system can be easily modified for frequency hopping spread spectrum transmissions, fully compliant with FCC part 15.427 specifications for the 902-928 MHz ISM band. This work has shown that no architecture modifications are required for the four existing National Semiconductor integrated circuits that form the basis of the DECT system. Significant objectives such as the development of cheap and robust frequency hop synchronization algorithms and the design of a burst mode controller have been achieved. The aim of achieving maximum flexibility based on the software radio design has also been attained. All the operational functions are controlled through software modules that can be easily and rapidly changed. Finally a hardware prototype of the FH transceiver has been successfully implemented and tested.

System performance can be further improved as follows:

1. The use of an RF front end with the various RF bandpass filter bandwidths matched to the FH transmission rather than for DECT.

2. The use of a better timing recovery system than the squaring loop which has difficulties in maintaining bit timing synchronization in burst transmissions since the PLL used is not designed for burst operation.

3. The use of even more robust frequency hop synchronization algorithms.

In conclusion, an affordable and functional FH transceiver architecture has been designed, modeled, extensively simulated and implemented in prototype hardware. Two novel FH synchronization algorithms implemented completely at baseband have been developed.

7.2 Future Work

Possible future work should concentrate on the development of Frequency Hop Multiple Access (FHMA) or a hybrid combination of TDMA and FHMA. Work can be done on the development of an all digital receiver architecture that can take advantage of the flexibility of the digital processing. This would involve digitizing the signal at IF using bandpass sampling, and implementing inphase and quadrature demodulation. System performance can be further improved by using forward error correction with suitable bit interleaving to randomize burst errors caused by the channel. The existing transceiver can also be easily modified for use as a direct sequence radio as well as for hybrid combinations of frequency hopping and direct sequence. This future expansion would not involve any change in the hardware, but would only require different software modules. Ten separate direct sequence channels with a spread double sided bandwidth of 1.728 MHz can be accommodated with the existing DECT RF front end. The system can be designed such that it is fully compatible with the existing DECT physical layer and would not require modifications such as those required for the frequency hopper. Finally, the performance of the FH transceiver needs to be characterized for more realistic indoor and outdoor channels.

Appendix

AD-2111 Assembly Code for the FH Transmitter

```
.MODULE/ram/abs=0/boot=0  fh_routine;

{ Variable definitions }
.CONST taps                     = 4;
.CONST taps1                    = 1;
.CONST taps2                    = 2;
.CONST tapsframe                = 4;
.CONST tapsbin                  = 30;
.CONST no_frame                 = 64;
.CONST no_data                  = 480;
.CONST no_total                 = 720;
.CONST bits_delay               = 00;

.VAR/PM/RAM/CIRC Rdata[taps1];
.VAR/PM/RAM/CIRC Ndata[taps2];
.VAR/DM/RAM/CIRC datatest[tapsbin];
.VAR/PM/RAM databin1[tapsbin];
.VAR/PM/RAM databin2[tapsbin];
.VAR/PM/RAM dataframe[tapsframe];
.VAR/DM/RAM data_frame[tapsframe];
.VAR/DM/RAM data_bin1[tapsbin];
.VAR/DM/RAM data_bin2[tapsbin];
.VAR/DM/RAM data_bin11[tapsbin];
.VAR/DM/RAM data_bin22[tapsbin];
.VAR/DM/RAM/CIRC R_data[taps1];
.VAR/DM/RAM/CIRC N_data[taps2];
.VAR/DM/RAM counter;
.VAR/DM/RAM counter_pll;
.VAR/DM/RAM r_counter;
.VAR/DM/RAM counter_frame;
.VAR/DM/RAM counter_data;
.VAR/DM/RAM counter_write;
.VAR/DM/RAM counterdata;
.VAR/DM/RAM val;
```

```
.VAR/DM/RAM storage;
.VAR/DM/RAM n_counter;
.VAR/DM/RAM flag_le;
.VAR/DM/RAM flag_reg1;
.VAR/DM/RAM bin_no;
.VAR/DM/RAM Tx_or_Rx;
.VAR/DM/RAM N1_or_N2;
.VAR/DM/RAM shift_value1;
.VAR/DM/RAM shift_value2;
.VAR/DM/RAM shift_value3;

.init Rdata: <rdata.dat>;              { Data for programming the PLL's R counter }
.init Ndata: <ndata.dat>;              { Data for programming the PLL's N counter }
.init databin1: <bin1.dat>;
.init databin2: <bin2.dat>;
.init dataframe: <frame.dat>;          { Frame to be Transmitted }

call DM_init;jump start;nop;nop;       { Reset Interrupt vector }
rti;nop;nop;nop;                       { All other interrupts are inactive }
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
jump timer_hi_prior;nop;nop;nop;       { Timer interrupt }

{ DSP system register initialization }
DM_init: l0          = 0;
         m0          = 1;
         i0          = h#3ffb;
         dm(i0,m0) = 0x0000;
         dm(i0,m0) = 0x0060;           { 4b for 200 kbps }
         dm(i0,m0) = 0x0060;
         dm(i0,m0) = 0x0000;
         dm(i0,m0) = 0x0800;
         rts;

{ Load data memory from program memory }
start:  i1 = ^R_data; m1 = 1;
        i4 = ^Rdata; m4 = 1;
        l4 = 1; l1 = 1;
        cntr = 1;
        do load1 until ce;
          ar = pm(i4,m4);
load1: dm(i1,m1) = ar;

      i4 = ^Ndata; i1 = ^N_data;
      l4 = taps2; l1 = taps2;
      cntr = taps2;
      do load2 until ce;
```

```
            ar = pm(i4,m4);
load2:  dm(i1,m1) = ar;

             i4 = ^databin1; i1 = ^data_bin1; l4 = 0; l1 = 0;
             cntr = tapsbin;
             do loadbin1 until ce;
               ar = pm(i4,m4);
loadbin1: dm(i1,m1) = ar;

             i4 = ^databin2; i1 = ^data_bin2; l4 = 0; l1  = 0;
             cntr = tapsbin;
             do loadbin2 until ce;
               ar = pm(i4,m4);
loadbin2: dm(i1,m1) = ar;

            i4 = ^dataframe; i1 = ^data_frame; l4 = 0; l1 = 0;
            cntr = tapsframe;
            do loadframe until ce;
             ar = pm(i4,m4);
loadframe: dm(i1,m1) = ar;

         i1 = ^data_bin11; l1 = 0; m0 = 1;
         ar = 0;
         cntr = tapsbin;
         do clrbin1 until ce;
clrbin1: dm(i1,m0) = ar;

        i1 = ^data_bin22; l1 = 0; m0 = 1;
        ar = 0;
        cntr = tapsbin;
        do clrbin until ce;
clrbin:  dm(i1,m0) = ar;

      i1 = ^data_bin1; m1 = 1; l1 =0;
      i6 = ^datatest; m6 = 1; l6 = 30;
      cntr = tapsbin;
      do ddd until ce;
        ar = dm(i1,m1);
ddd: dm(i6,m6) = ar;

{ Initialize memory pointers and modifiers }
     m0 = 1;   l0 = 0;
     m1 = 0;   l1 = 1;
     m2 = 0;   l2 = 0;
     m3 = 0;   l3 = taps2;
     m6 = 1;   l6 = 30;

     i0 = ^data_bin11;
     i1 = ^R_data;
     i3 = ^N_data;
     i6 = ^datatest;
```

```
      reset FL0;
      reset FLAG_OUT;
      ax1                     = 0;
      ay1                     = 1;
      sr0                     = ay1;
      ar                      = no_total;
      dm(counter)             = ar;      { Total number of bits in the TX or RX frame }
      ar                      = 64;
      dm(counter_pll)         = ar;      { No of clock cycles used for programming the PLL }
                                         { the PLL to stabilize}
      ar                      = 16;
      dm(r_counter)           = ar;      { Bit tracking counter used for R-counter programming }
      ar                      = 19;
      dm(n_counter)           = ar;      { Bit tracking counter used for N-counter programming }
      ar                      = no_frame;
      dm(counter_frame)       = ar;
      ar                      = no_data;
      dm(counter_data)        = ar;
      dm(counterdata)         = ar;
      ar                      = 3;
      dm(counter_write)       = ar;      { Counter to indicate when CVSDM data should be accessed }
      dm(storage)             = ax1;     { Flag to indicate which storage buffer is being written into }
      dm(bin_no)              = ax1;
      dm(flag_reg1)           = ax1;     { Flag to indicate whether the R or N counter }
                                         { of the PLL is to be programmed }
      dm(N1_or_N2)            = ax1;     { Flag to indicate which segment of the 19 bit  }
                                         { word for the N counter is to be programmed }
                                         { Since the N counter word is 19 bits long and }
                                         { since the AD2111 reisters are 16 bits long }
                                         { The 19 bit word is stored as two 16 bit words }

      dm(shift_value1)        = ay1;     { rotating mask value used in bit packing }
      dm(shift_value2)        = ay1;
      dm(shift_value3)        = ay1;
      dm(val)                 = ay1;
      dm(flag_le)             = ay1;     { Flag to indicate if PLL programming is to }
                                         { be continued or wait for PLL to stabilize }
      dm(Tx_or_Rx)            = ay1;     { Flag to indicate Transmit or Receive function }

      i5                      = h#2000;
      m5                      = 0;
      l5                      = 0;
      ar                      = pm(i5,m5); { pull Tx_Pd  HIGH }
      dm(i5,m5)               = ar;      { pull Pll_Pd HIGH }

      ICNTL                   = b#00000;
      MSTAT                   = b#0111100;
      IMASK                   = b#00000001;

wait: idle;
       jump wait;
```

```
timer_hi_prior:

{ This section reads in a bit from the Codec and stores it }
            ay0                     = dm(counter_write);
            ar                      = ay0-1;
            dm(counter_write)       = ar;
            ar                      = pass ar;      { Check if codec is to be sampled }
            if NE jump beg_timer;                   { Codec is not to be sampled }
            ar                      = 3;            { Codec is to be sampled }
            dm(counter_write)       = ar;           { Reload count for codec sampling time }

{ This section packs the input bits into 16 bit words for storage }
            ax0                     = 0;
            sr0                     = dm(val);      { Reload rotating value of shift register for packing }
            ar                      = dm(i6,m6);
            ar                      = pass ar;
            if EQ jump zz2;
            ax0                     = sr0;
zz2:          ay0                   = dm(i0,m1);    {Get 16 bit word to be packed }
            ar                      = ax0 or ay0;   {OR existing contents with new data bit }
            dm(i0,m1)               = ar;           { Store new word }
            sr                      = lshift sr0 by 1 (LO); { Shift rotating pattern }
            dm(val)                 = sr0;          { Store rotating pattern }
            ar                      = pass sr0;     { Check if 16 bits have been packed }
            if NE jump ahead;                       { No, continue packing }
            ay0                     = dm(i0,m0);    { Get new 16 bit word to be packed }
            dm(val)                 = ay1;          { Reinitialize rotating pattern to 1 }
ahead:        ay0                   = dm(counterdata);
            ar                      = ay0-1;
            dm(counterdata)         = ar;
            ar                      = pass ar;      { Check if 480 bits have been packed }
            if NE jump beg_timer;                   { No, continue packing with existing buffer }
            ar                      = 480;
            dm(counterdata)         = ar;
            ar                      = dm(storage);
            ar                      = NOT ar;       { Toggle storage bin flag to change buffers }
            dm(storage)             = ar;
            i0                      = ^data_bin11;
            ar                      = pass ar;      { Check which buffer is to be packed }
            if EQ jump beg_timer;
            i0                      = ^data_bin22;  { Repoint pointer to point to new buffer for packing }

beg_timer: reset FL1;                               { clear SER_CLK }
            ar                      = dm(Tx_or_Rx);{ Check if Transmit or Receive }
            ar                      = pass ar;
            if EQ jump Receive;                     { Jump to Receive section }

            { Transmit section }
            ar                      = dm(i5,m5);    { pull Tx_Pd LOW }

            ay0                     = dm(counter);  { Check if all Transmit bits have been transmitted }
            ar                    = pass ay0;
            if EQ jump Ready_Rx;             { Transmission is over, jump to section }
                                             { to initialize parameters for Receive  }

            ar                    = ay0-1;
            dm(counter)           = ar;
            ay0                   = dm(counter_pll); { Check if PLL programming }
                                                        { is over }
            ar                    = pass ay0;
            if EQ jump cont;                 { Pll programming is over }
            ar                    = 3;
            dm(n_counter)         = ar;
            dm(shift_value1)      = ay1;
            dm(N1_or_N2)          = ay1;
            m3                    = 1;
            ay0                   = dm(i3,m3);   { Dummy write to increment pointer to next word }
            m3                    = 0;
            reset FLAG_OUT;
            rti;

set_N1:   set FL1;
            ar                    = 16;
            dm(r_counter)         = ar;
            dm(shift_value1)      = ay1;
            dm(N1_or_N2)          = ax1;
            dm(flag_le)           = ax1;
            m3                    = 1;
            ay0                   = dm(i3,m3);
            m3                    = 0;
            reset FLAG_OUT;
            set FL2;
            rti;

prog_rcounter:  ay0               = dm(i1,m1);
            m1                    = 0;
            sr0                   = dm(shift_value2);
            ar                    = sr0 and ay0;
            sr                    = lshift sr0 by 1 (LO);
            dm(shift_value2)      = sr0;
            af                    = pass sr0;
            if NE jump stepover1;
            sr0                   = ay1;
            m1                    = 1;
            ay0                   = dm(i1,m1);
            m1                    = 0;
stepover1: ar                     = pass ar;
            if EQ jump btzero1;
            set FL0;
            jump continuee;
btzero1:    reset FL0;
continuee:  ay0                   = dm(r_counter);
```

```
            ar                          = ay0-1;
            dm(r_counter)               = ar;
            ar                          = pass ar;
            if NE jump contin;
            set FL1;
            dm(flag_reg1)               = ay1;
            dm(shift_value2)            = ay1;
            ar                          = 16;
            dm(n_counter)               = ar;          { Reload tracking counter for N-counter }
            dm(N1_or_N2)                = ax1;
            reset FLAG_OUT;
            set FL2;
            rti;

{ Operations during the RECEIVE mode }
Receive:    ar                          = pm(i5,m5);   { Set Tx_Pd  & Pll_Pd HIGH }
            dm(i5,m5)                   = ar;
            reset FLAG_OUT;
            ay0                         = dm(counter);
            ar                          = ay0-1;
            dm(counter)                 = ar;
            ar                          = pass ar;
            if EQ jump ready_tx;
            rti;

{ Initialize various parameters for the TRANSMIT mode of operation }

ready_tx:   ar                          = no_total;
            dm(counter)                 = ar;
            ar                          = 64;
            dm(counter_pll)             = ar;
            ar                          = no_data;
            dm(counter_data)            = ar;
            ar                          = no_frame;
            dm(counter_frame)           = ar;
            ar                          = dm(bin_no);
            ar                          = NOT ar;
            dm(bin_no)                  = ar;

            dm(Tx_or_Rx)                = ay1;
            dm(flag_le)                 = ay1;
            dm(flag_reg1)               = ax1;
            dm(shift_value1)            = ay1;
            dm(shift_value2)            = ay1;
            rti;

{ Initialize various parameters for the RECEIVE mode }
Ready_Rx:   ar                          = no_total;
            dm(counter)                 = ar;
            ar                          = 64;
            dm(counter_pll)             = ar;
            dm(Tx_or_Rx)            = ax1;          { was ax1 }
            dm(shift_value1)        = ay1;
            dm(shift_value2)        = ay1;
            rti;

{ Pause for PLL to stabilize. No operations carried out during this period }
dont_do:    reset FL1;
            set FL0;
            reset FL2;
            reset FLAG_OUT;
            sr0                     = ay1;
            rti;

cont:        pm(i5,m5)              = ar;           { Pull Pll_Pd LOW }
contt:       ay0                    = dm(counter_write);
            ar                      = pass ay0;
            if EQ jump dont_delay;
            CNTR                    = 30;
            do delay1 until CE;
delay1:      ar                     = 0;

dont_delay:ay0                      = dm(counter_frame); { Check if header and frame  }
                                                     { sync have been transmitted }

            ar                      = pass ay0;
            if EQ jump ss1;
            ax0                     = no_frame;
            ar                      = ax0-ay0;
            ar                      = pass ar;
            if NE jump xx1;
            i2                      = ^data_frame; { Point pointer to array containing the }
                                                   { Header and the Frame Sync word}
xx1:        ar                      = ay0-1;
            dm(counter_frame)       = ar;
            jump contt1;

ss1:        ay0                     = dm(counter_data);
            ar                      = pass ay0;
            if EQ jump rs;
            ax0                     = no_data;
            ar                      = ax0-ay0;
            ar                      = pass ar;
            if NE jump xx2;
            i2                      = ^data_bin11;
            ar                      = dm(bin_no);
            ar                      = pass ar;
            if NE jump xx2;
            i2                      = ^data_bin22;
xx2:        ar                      = ay0-1;
            dm(counter_data)        = ar;

contt1:      ay0                    = dm(i2,m2);
```

```
            sr0                 = dm(shift_value3);
            ar                  = sr0 and ay0;
            sr                  = lshift sr0 by 1 (LO);
            dm(shift_value3)    = sr0;
            af                  = pass sr0;
            if NE jump stepp;
            dm(shift_value3)    = ay1;
            ay0                 = dm(i2,m0);   { Dummy write to increment pointer to next word }
stepp:       ar                 = pass ar;
            if EQ jump btzero;                 { Check if data bit to be txted is a 1 or 0 }
            set FLAG_OUT;                      { TX_DATA = 1 }
            rti;
btzero:      reset FLAG_OUT;                   { TX_DATA = 0 }
            rti;

contin:      set FL1;
rs:         reset FLAG_OUT;                    { TX_DATA = 0 }
            rti;


.ENDMOD;
```

AD-21020 Assembly Code for the FH Receiver

```
#define TIMER_PERIOD        159             { 1/Tb =  bps }
#define TIMER_COUNT         TIMER_PERIOD
#define TAPS               0x1f             { Matched Filter Length = 31 }
#define RCOUNTER            0x8601          { Initial PLL r_counter value }
#define NUMFREQ            10               { Number of Frequency for Hopping }
#define SOT_ERR_NUM         10              { Number of errored frames tolerated }

#include "def_fh.h"                         { ADSP-21020 System Register bit definitions }

.SEGMENT/DM dm_sram;                        { selected by DMS0~ }
 .var dline[TAPS];
.var                                                                    n_counter[NUM-
FREQ]=0x2e5c0,0x165c0,0x3a5c0,0x0a5c0,0x225c0,0x1c5c0,0x345c0,0x045c0,0x285c0,0x105c0;
 .var r_counter=RCOUNTER;
 .var sot_errs;
.ENDSEG;

.SEGMENT/DM dac_a;
 .PORT dac_a;
.ENDSEG;

.SEGMENT/DM dac_b;
 .PORT dac_b;
.ENDSEG;

.SEGMENT/PM   rst_svc;
 jump main;
.ENDSEG;

.SEGMENT/PM   irq2_svc;
 bit set mode2 TIMEN;                       { Enable Timer }
 nop;
wait:  jump twait;                          { Wait for timer Interrupt }
 nop;
ENDSEG;


.SEGMENT/PM   tmzh_svc;
 jump timer_hi_prior;
 nop;
.ENDSEG;

.SEGMENT/PM   pm_sram;
 .VAR sot_coef[TAPS]    = "sot_coef.dat";   { Data file containing the synchronization sequence }

main:
call init_rcv;
```

```
bit set mode1 IRPTEN;                          { enable global int.}
nop;
mwait:  jump mwait;                            { Wait for timer (irq2), request transmit }

{ Receiver initialization }
init_rcv:
        pmwait          = 0x1c21;              { pgsz=32K,pmwtstates=0,sw.wtstates only}
        dmwait          = 0x709421;            { pgsz=32K,bank2_dmwtstates=1,sw.wtstates only}
        bit set mode1 NESTM;                   {Interrupt Nesting Enabled }
        irptl           = 0;                   { clear any pending interrupts}
        bit set mode2 0x10; nop; read cache 0; bit clr mode2 0x10; {clr cache}
        { irq2 enable, timer enable high priority}
        bit set imask TMZHI|IRQ2I;
        bit clr mode2 FLG0O;                   { irq2,3 edge sens, flag0 = input, flag 1,2,3 = outputs}
        bit set mode2 IRQ2E|FLG1O|FLG2O|FLG3O;
        bit clr astat FLG1|FLG2|FLG3;          { turn flag LEDs off}


        tperiod         = TIMER_PERIOD;        { Set Timer for Bit Rate }
        tcount          = TIMER_COUNT;
        b5              = n_counter;
        l5              = NUMFREQ;
        m5              = 1;                   { Buffer pll n_counter }
        b8              = sot_coef;
        l8              = TAPS;
        m8              = 1;                   { Buffer for SOT correlator coefficients }
        b6              = dline;
        l6              = TAPS;
        m6              = 1;                   { Buffer for correlator delay line  }
        dm(0x100000) = r1;                     { Set sfield Low }
        nop;
        r1              = pm(0x200000);
        lcntr           = TAPS, do clr_dline until lce;  { Clear Delay Line Storage Buffer }
clr_dline:  dm(i6,m6) =0x0;
        r1              = SOT_ERR_NUM;
        dm(sot_errs)    = r1;
        r2              = 64;                  { Initial bit_count (592) }
        r5              = 64;                  { Initial pll_count }
        r3              = 0x00000001;          { r3 is initial data bit mask (constant) }
        r4              = r3;                  { r4 is rotated mask }
        ustat1          = 0x0;                 { Clear rcv status register }
        bit set ustat1 RCV_FLG;                { Set Rcv Flag in User Status Register }
        bit set ustat1 SYNC_FLG;               { Set Sync SOT Flag in User Status Register }
        rts;
        nop;


{____________________________Timer Interrupt____________________________________}
timer_hi_prior:
        pop sts;                               { get ASTAT from stack, pushed there by IRQ }
        bit tst ustat1 PLL_FLG;                { Program PLL during Guard space. }
```

```
            if NOT TF jump prog_pll;
            bit tst ustat1 RCV_FLG;                 { test for rcv mode }
            if TF jump test_sot;
            r2              = r2-1;
            if ne jump end_rcv;
            bit set ustat1 RCV_FLG;
            bit clr ustat1 PLL_FLG;
            bit clr ustat1 SOT_FLG;
            {bit set astat FLG2;}
            r2              = 64;
            r5              = 64;
            dm(0x100000) = r1;
            nop;
            r1              = pm(0x200000);
            jump end_rcv;

test_sot:

          bit tst ustat1 SOT_FLG;
           if TF jump start_rcv;                    { SOT detected, start deformatting received data }
            call corrl_routine;                     { SOT not detected, continue matched filtering }
            bit tst ustat1 SYNC_FLG;
            if TF jump sot_sync;
            r2              = r2-1;
            if ne jump end_rcv;
            bit tst ustat1 SOT_FLG;
            if NOT TF jump no_sot;
            r1              = SOT_ERR_NUM;
            dm(sot_errs)    = r1;
            r2              = 592;
            jump end_rcv;
no_sot:
            r1              = dm(sot_errs);         { Counter tracking the number of missed frames }
            r1              = r1-1;
            if eq jump sync_err;                    { # of missed frames = threshold, re-synchronize }
            dm(sot_errs)    = r1;
            bit set ustat1 SOT_FLG;
            r2              = 592;
            dm(sot_errs)    = r1;
            jump end_rcv;
sync_err:
            bit set ustat1 SYNC_FLG;
            r1=SOT_ERR_NUM;
            dm(sot_errs)    = r1;                   { Reload counter with threshold for missed frames }
            bit clr ustat1 SOT_FLG;                 { Clear SOT detect flag }
            jump end_rcv;                           { Jump to end of routine }

sot_sync:
            bit tst ustat1 SOT_FLG;
            if NOT TF jump end_rcv;
            bit clr ustat1 SYNC_FLG;
            r2              = 592;
            jump end_rcv;

start_rcv:
            if NOT FLAG0_IN jump flag1_clr;     { Read in received data bit }
            bit set astat FLG1;                 { If bit = 1, set FLAG 1}
            jump flag1_set;
flag1_clr:
            bit clr astat FLG1;                 { If bit = 0, clear FLAG 1}
flag1_set:
            r2              = r2-1;             { Decrement counter keeping track of # of bits rxed }
            if ne jump end_rcv;
            {bit clr astat FLG2;}               { Frame over, reinitialize }
            dm(0x200000) = r1;                  { Set S_FIELD High }
            nop;
            r1              = pm(0x200000);     { Dummy write to generate Clock for S_FIELD latch }
            bit clr ustat1 RCV_FLG;
            r2              = 720;              { Reload counter with total number of bits in the frame }
end_rcv:
            push sts;                           { Push back popped stack }
            rti;                                { Return from interrupt }
            nop;

{ Correlation routine for sync dtection }
corrl_routine:
            { Read flag0 Input }
            f11             = 1.0;              { read flag0 input status }
            if NOT FLAG0_IN f11= -f11;
            bit set mode1 SRRFL;                { Select alternate register3 r0-r7 (f0-f7) }
            nop;
            f0              = f11;              { Input data passed to f0.}
            r12             = r12 xor r12, dm(i6,m6) = f0;         { clear r12 & data move}
            r8              = r8 xor r8, f0 = dm(i6,m6), f4 = pm(i8,m8); { clear r8 & data move}
            lcntr           = TAPS-1, do macs until lce;
macs:       f12 = f0*f4, f8 = f8+f12, f0 = dm(i6,m6), f4 = pm(i8,m8);
            f12             = f0*f4, f8 = f8+f12;
            f8              = f8+f12;
            f12             = 3.0;              { normalize correlation }
            f11             = f8*f12;
            f12             = 128.0;
            f11             = f11+f12;
            r11             = fix f11;          { convert to FIXED left justified }
            r11             = lshift r11 by 24;
            dm(dac_a)       = r11;              { output DAC value }
            bit clr mode1 SRRFL;                { Deselect alternate registers }
            f11             = 26.0;
            f8              = abs f8;
            comp(f8,f11);                       { Correlator_Out > Threshold ?}
            if lt jump end_corrl;
            bit set ustat1 SOT_FLG;
end_corrl:
```

```
            rts;

{ Program the PLL }
prog_pll:

{ program r_counter first }

            r1              = 64;                   { if pll_count=120, init. programming }
            comp(r5,r1);
            if ne jump pll_start;
            dm(0x800000) = r1;                      { Set TX_PD High }
            nop;
            r1              = pm(0x800000);
            nop;
            dm(0x400000) = r1;                      {Set PLL_PD High }
            nop;
            r1              = pm(0x400000);
            r8              = dm(r_counter);        { Initial PLL r_Counter value }
            r7              = 49;                   { bit_count for r_counter enable }
            r4              = r3;                   { reset data mask }

pll_start:
            bit clr astat FLG2|FLG3;                { clear LE and Pll Ser_Clock Inputs }
            r1              = r8 and r4;            { parse bit from data word }
            if ne jump bit_hi;
            bit clr astat FLG1;
            jump bit_lo;
bit_hi:
            bit set astat FLG1;
bit_lo:
            r1              = rot r4 by 1;          { rotate data mask bit }
            r4              = r1;
            comp(r5,r7);
            if eq jump clk_enbl;
            lcntr = 30, do delay1 until lce;
delay1:   nop;
            bit set astat FLG3;
            jump end_pll;

{ Last n_counter bit to be programmed: Clock data in, wait, then assert LE }
clk_enbl:
            lcntr = 20, do delay2 until lce;
delay2:    nop;
            bit set astat FLG3;                     { Clock data }
            lcntr = 15, do delay3 until lce;
delay3:    nop;
            bit set astat FLG2;                     { assert Load Enable }
            r4              = r3;
            m5              = 0;
            r8              = dm(i5,m5);            { load value for n_counter }
            r7              = 30;                   { bit_count for n_counter enable }
```

```
end_pll:
            r5              = r5-1;             { Decrement pll_counter }
            r1              = 29;
            comp(r5,r1);
            if ne jump end_pll2;
            bit set ustat1 PLL_FLG;
            m5              = 1;                { Get next pll n_counter value }
            r8              = dm(i5,m5);
            bit clr astat FLG2|FLG3;            { clear LE and Pll Ser_Clock Inputs }

end_pll2:
            push sts;
            rti;


.ENDSEG;
```

Bibliography

[1] ETSI, "European Telecommunications Standard," *Digital European Cordless Telecommunications Common Interface*, ETS 300 175-1 to 300 175-8, Sophia Antipolis, France, October 1992.

[2] J. G. Acres and A. N. Severinson, "Spread Spectrum Transmissions in the 902-928 MHz Band," *Proc. ICWC*, pp. 441-444, 1992.

[3] P. D. Rasky et al., "Slow Frequency-Hop TDMA/CDMA for Macrocellular Personal Communications," *IEEE Personal Communications Magazine*, pp. 26-35, Second Quarter 1994.

[4] National Semiconductor, "First Integrated Chipset for DECT," *Waveguide*, pp. 1-3, Spring 1993.

[5] ETSI, "Radio Equipment and Systems," *Digital European Cordless Telecommunications Common Interface*, ETS 300 175-1 to 9, 1992.

[6] Theodore S. Rappaport, *Wireless Communications, Principles and Practice*, Prentice Hall, pp. 535-540, 1996.

[7] G. Schultes et al., "Physical and Medium Access Layer DECT-Testbed," *COST 231*, TD(92) 028, Vienna, January 1992.

[8] National Semiconductor, *DECT Radio Demo Manual*, June 1994.

[9] National Semiconductor, "Performance Evaluation of a Low Cost, Solid State Radio Front End for DECT," *DECT Radio Demo Manua*l, June 1994.

[10] National Semiconductor, *LMX2411 Baseband Processor for Radio Communica-*

tions Manual, 1993.

[11] D. E. Fague, “An Integrated Baseband Processor for Universal Personal Communications Applications,” *Proc. PIMRC*, pp. 32-35, October 1993.

[12] National Semiconductor, *LMX2315 Frequency Synthesizer for RF personal Communications Manual*, 1993.

[13] D. E. Fague, A. Dao and C. Karmel, “Techniques for Open Loop Modulation of a Wideband VCO for DECT,” *Proc. of the RF Expo*, March 1994.

[14] National Semiconductor, *LMX2216 LNA/Mixer for RF personal Communications Manual*, 1993.

[15] National Semiconductor, *LMX2240 Intermediate Frequency Receiver Manual*, 1993.

[16] D. E. Fague, “A Fully Integrated Modulator/Demodulator for DECT,” *Proc. of the RF Expo West*, pp. 228-231, March 1993.

[17] National Semiconductor, *Applications Information-LMX2240 Intermediate Frequency Receiver Manual*, pp. 10, 1993.

[18] L. W. Couch II, *Digital And Analog Communication Systems*, Macmillan, 1993.

[19] M. K. Simon and C. C. Wang, “Bit Synchronization of Differentially Detected MSK and GMSK,” *Proc. ICC*, pp. 583-590, June 1985.

[20] M. S. El-Tanany et al., “Data Detection and Timing Recovery for a Noncoherent Discriminator Based GMSK Receiver,” *Proc. ICC*, pp. 243-248, 1989.

[21] R. E. Ziemer and R. L. Petersen, *Introduction to Digital Communication*, Macmillan, 1992.

[22] J. G. Proakis, *Digital Communications*, McGraw-Hill, 1989.

[23] R. C. Dixon, *Spread Spectrum Systems*, John Wiley & Sons, Inc., 1984.

[24] C. A. Putman et al., “A Comparison of Schemes for Coarse Acquisition of Frequency Hopped Spread Spectrum Signals,” *IEEE Transactions on Communications*, pp. 183-188, Feb. 1983.

[25] J. H. Fischer et al., “Wide-Band Packet Radio fpr Multipath Environments,” *IEEE Transactions on Communications*, pp. 564-575, May 1988.

[26] M. C. Jeruchim et al., *Simulation of Communication Systems*, Plenum Press, 1992.

[27] Analog Devices, *ADSP-2100 Family User's Manual*, Prentice Hall, 1993.

[28] Analog Devices, *ADSP-21020 Family User's Manual*, Analog Devicesl, 1994.

[29] J. Mitola, "The Software radio Architecture," *IEEE Communications Magazine*, pp. 26-38, May 1995.

[30] R. J. Lackey et al., "Speakeasy: The Military Software Radio," *IEEE Communications Magazine*, pp. 56-61, May 1995.

[31] Comdisco Systems, *SPW-The DSP Framework User's Guide and Tutorial*, Comdisco Systems, March 1994.

[32] T. P. Krauss et al., *Signal Processing Toolbox*, The Math Works, June 1994.

[33] National Semiconductor, *Specification for the DECT ARi Interface to the Radio Frequency Front End*, 1993.

[34] G. Schultes et al., "Performance of the Siemens DECT Prototype Gigaset 95x in a Dispersive Indoor Environment," *Proc. VTC*, pp. 1079-1083, 1994.

[35] I. Cronin et al., "Irreducible Error Performance of a Digital Portable Communication System in a Controlled Time-Dispersion Indoor Channel," *IEEE Journal on Sel. Areas In Communications*, pp. 1024-1033, Sept. 1993.

[36] L. B. Lopes, "On the Radio Link Performance of the Digital European Cordless Telecommunications (DECT) System," *Proc. Globecom*, pp. 1013-1017, 1990.

[37] G. Schultes and I. Cronin, "Measured Performance of DECT Transmission in Low Dispersive Indoor Radio Channel," *IEE Electronics Letters*, pp. 1625-1627, August 1992.

[38] T. S. Rappaport et al., " Statistical Channel Impulse Response Models for factory and Open Plan Building Radio Communication System Design," *IEEE Transactions on Communications*, pp. 794-806, May 1991.

Vita

Francis Dominique was born on February 23, 1969, in Pondicherry, India. He received the B. Tech degree in Electrical Engineering with distinction in May 1990 from Pondicherry University and M.Tech degree in Electrical Engineering with distinction in February 1992 from Pondicherry University. He joined Virginia Tech in Fall 1993 and the MPRG in Summer 1994 where he currently works for Dr. Jeffrey H. Reed. He is the author of four conference papers and one journal paper. His areas of interest are digital communication architectures, adaptive signal processing for interference suppression, neural networks, adaptive antenna arrays and data compression.

Francis Dominique