2001

THE APPLICATION OF REAL-TIME SOFTWARE IN THE IMPLEMENTATION OF LOW-COST SATELLITE RETURN LINKS

SLADER, JAMES TOM

http://hdl.handle.net/10026.1/2434

http://dx.doi.org/10.24382/3639
University of Plymouth

THE APPLICATION OF REAL-TIME SOFTWARE IN THE IMPLEMENTATION OF LOW-COST SATELLITE RETURN LINKS

by

JAMES TOM SLADER

A Thesis submitted to the University of Plymouth
in partial fulfilment for the Degree of

DOCTOR OF PHILOSOPHY

Satellite Research Centre
Department of Communication and Electronic Engineering
Faculty of Technology

October 2001

THE APPLICATION OF REAL-TIME SOFTWARE IN THE IMPLEMENTATION OF LOW-COST SATELLITE RETURN LINKS

by James Tom Slader

Abstract

Digital Signal Processors (DSPs) have evolved to a level where it is feasible for digital modems with relatively low data rates to be implemented entirely with software algorithms. With current technology it is still necessary for analogue processing between the RF input and a low frequency IF but, as DSP technology advances, it will become possible to shift the interface between analogue and digital domains ever closer towards the RF input. The software radio concept is a long-term goal which aims to realise software-based digital modems which are completely flexible in terms of operating frequency, bandwidth, modulation format and source coding. The ideal software radio cannot be realised until DSP, Analogue to Digital (A/D) and Digital to Analogue (D/A) technology has advanced sufficiently. Until these advances have been made, it is often necessary to sacrifice optimum performance in order to achieve real-time operation. This Thesis investigates practical real-time algorithms for carrier frequency synchronisation, symbol timing synchronisation, modulation, demodulation and FEC. Included in this work are novel software-based transceivers for continuous-mode transmission, burst-mode transmission, frequency modulation, phase modulation and orthogonal frequency division multiplexing (OFDM).

Ideal applications for this work combine the requirement for flexible baseband signal processing and a relatively low data rate. Suitable applications for this work were identified in low-cost satellite return links, and specifically in asymmetric satellite Internet delivery systems. These systems employ a high-speed (>>2Mbps) DVB channel from service provider to customer and a low-cost, low-speed (32-128 kbps) return channel. This Thesis also discusses asymmetric satellite Internet delivery systems, practical considerations for their implementation and the techniques that are required to map TCP/IP traffic to low-cost satellite return links.

Contents

List of Abbreviations

ADC	Analogue to Digital Converter
ADSL	Asymmetrical Digital Subscriber Line
ARP	Address Resolution Protocol
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPS	Bits per Second
BPSK	Binary Phase Shift Keying
CNR	Carrier To Noise Ratio
DA-TDMA	Demand Assigned TDMA
DAC	Digital to Analogue Converter
DBPSK	Differential Binary Phase Shift Keying
DFT	Discrete Fourier Transform
DIT	Decimation in Time
DNS	Domain Naming System
DQPSK	Differential Quadrature Phase Shift Keying
DSL	Digital Subscriber Line
DSP	Digital Signal Processor/Processing
DVB	Digital Video Broadcasting
EPROM	Erasable Programmable Read Only Memory
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIFO	First In First Out (Memory)
FIR	Finite Impulse Response (Filter)
FM	Frequency Modulation
FPGA	Field-Programmable Gate Array
FSK	Frequency Shift Keying
FTP	File Transfer Protocol
GEO	Geosynchronous Earth Orbit
HTTP	Hyper-Text Transfer Protocol

ICI	Inter-Channel Interference
IF	Intermediate Frequency
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response (Filter)
IP	Internet Protocol
ISI	Inter-Symbol Interference
ISP	Internet Service provider
LAN	Local Area Network
LEO	Low Earth Orbit
LSB	Lest Significant Bit
LUT	Look-Up Table
MAC	Medium Access Controller
MEO	Medium Earth Orbit
MTU	Maximum Transmission Unit
NIC	Network Interface Controller
OFDM	Orthogonal Frequency Division Multiplexing
PC	Personal Computer
PLL	Phase-Locked Loop
PSTN	Public Switched Telephone Network
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
ROM	Read Only Memory
SNR	Signal to Noise Ratio
SPS	Symbols Per Second
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UW	Unique Work
VSAT	Very Small Aperture Terminal

List of Illustrations

List of Tables

Acknowledgements

I'd like to thank my supervisors Prof. Martin Tomlinson and Dr. Graham Wade for their wisdom and guidance.

I would also like to thank my colleagues Paul Smithson, Peter van Eetvelt and Adrian Ambroze for their advice, encouragement and friendship throughout.

This work is dedicated to Jenny, Wendy & John Slader, and could not have been completed without their continuing love and support.

Author's Declaration

At no time during the registration for the Degree of Doctor of Philosophy has the author been registered for any other University award.

This study was financed by a University of Plymouth studentship.

Relevant scientific seminars and conferences were regularly attended and work was presented at major International conferences. Three papers were prepared for publication and published (Slader et al., 1996; Smithson et al., 1996; Slader et al., 1998).

Signed [illegible]

Date 28/10/2001

1. Introduction

This Thesis is concerned with digital software-based modems for low-cost satellite links using both Low Earth Orbiting (LEO) and Geosynchronous Earth Orbiting (GEO) satellites. The work considers carrier frequency synchronisation, filtering, symbol timing synchronisation, burst demodulation and forward error correction. The main aims are to investigate the tradeoffs in performance which are required to realise the signal processing algorithms in real-time. The work investigates low-complexity, software-based signal processing algorithms for satellite modulator/demodulators and their real-time implementation on signal processing hardware based upon the Texas Instruments TMS320C50 fixed-point DSP, see Appendix A. The work divides into three strands and considers the carrier frequency and symbol timing synchronisation requirements for satellite data communication using frequency modulation (FM), differential phase modulation (D-BPSK, D-QPSK) and orthogonal frequency division modulation (OFDM). Much of this work is embodied in a satellite system which is now operational and is discussed later in the Thesis. This work also includes transport protocols for mapping TCP/IP applications to low-cost satellite return links and channel access protocols for fair and efficient utilisation of shared return channels.

1.1 Introduction and Background to Digital Modems & Software Radio

The following is an introductory background to digital modems and software radio. Figure 1-1 shows a block diagram of a typical digital communication system.

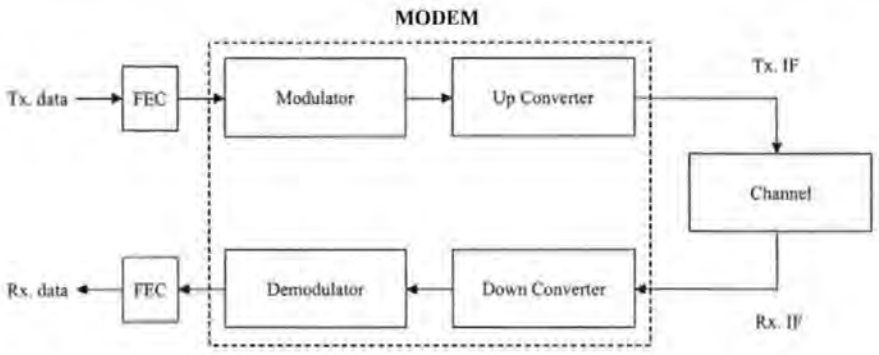
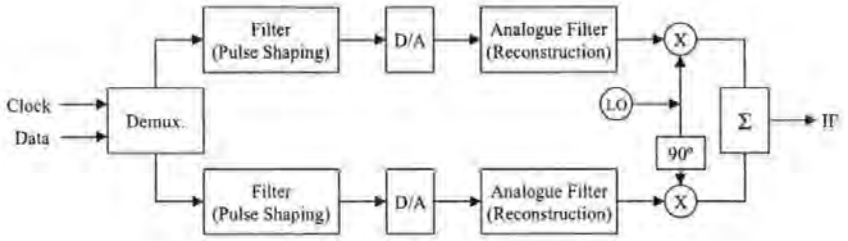


Figure 1-1. Digital communication system overview.

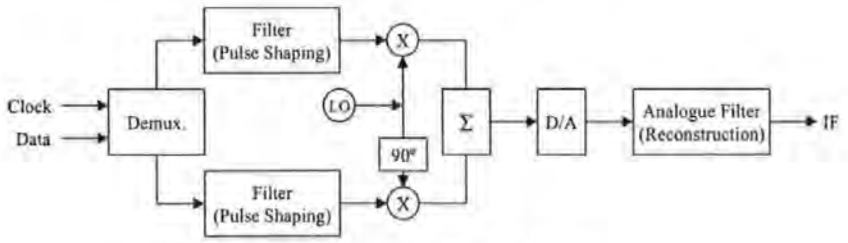
Data for transmission is encoded for forward error correction (FEC), to protect against channel errors, and is applied to the input of the modem where the modulator processes the data depending upon the choice of modulation and coding schemes. The modulation scheme depends upon whether power efficiency or spectral efficiency is of greater importance. Spectral efficiency of 2bits/sec/Hz is achieved practically with high order modulation schemes such as 8-phase shift keying (8-PSK) and 16-Quadrature amplitude modulation (16-QAM). The output from the modulator is a bandpass signal centred on an intermediate frequency (IF), and the upconverter performs translation of the signal to the desired transmit IF. In satellite communication it is common for the upconverter to provide a 70MHz transmit IF interface for satellite Earth Station applications. The channel adds noise to the signal and, depending upon the application, the signal can also be distorted in a number of ways. Satellite links are particularly susceptible to signal attenuation due to climatic effects such as rain. The downconverter operates in reverse to the upconverter, mixing the signal back down from the receive IF to a second (lower frequency) IF for the demodulator. The demodulator is responsible for receive filtering and sampling the wanted signal. Once the signal is sampled it can be processed digitally to compensate for carrier frequency and symbol timing errors allowing the received bits to be extracted. Occasional demodulated bit errors may be tolerated because of the FEC scheme, and the original transmitted data stream is reproduced at the output of the modem.

1.1.1 Outline Baseband I/Q and Digital Low IF Modulators

Figure 1-2a shows a baseband I/Q modulator structure. The demultiplexer generates odd and even bit sequences and each is oversampled and shaped by a digital filter to eliminate intersymbol interference (ISI) between successive symbols. After digital to analogue conversion (D/A) the signals are passed through analogue reconstruction filters and mixed up to an IF by a quadrature analogue modulator. Problems associated with this structure are mainly due to the accuracy of the analogue modulator, where any amplitude and phase imbalance result in a degradation to system performance. The characteristics of the D/A pair are also critical and must be matched in terms of output range and conversion times. These factors become even more significant as the order of the modulation scheme increases.



a. Baseband I/Q analogue modulator.



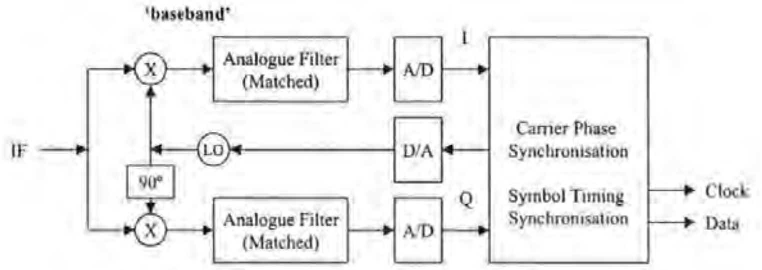
b. Digital low IF modulator.

Figure 1-2. Basic I/Q modulator structures.

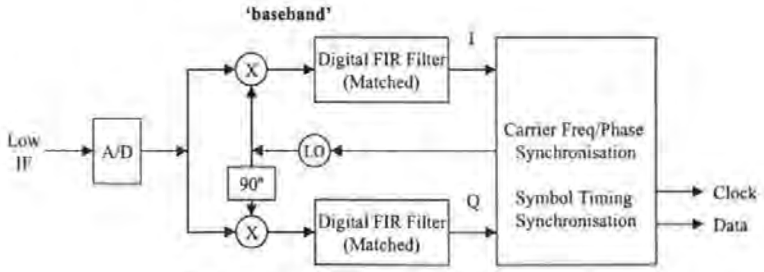
The problems associated with the baseband I/Q analogue modulator may be overcome by shifting the signal from baseband digitally and reconstructing the bandpass signal on a low frequency IF, Figure 1-2b. With upconversion performed in the digital domain the problems associated with the analogue modulator are avoided, only one D/A converter is required and several analogue components are eliminated. Since the low IF modulator structure is digital, functions may be reordered and novel architectures developed; such refinements are presented later in the Thesis.

1.1.2 Outline Baseband I/Q and Low IF Demodulator Structures

Figure 1-3a shows the front end of a baseband I/Q demodulator with analogue quadrature demodulator. The signal is shifted down to 'nominal' baseband by an analogue quadrature down converter and is subject to the same potential for amplitude and phase imbalance as the modulator equivalent. The signal plus noise is filtered using analogue filters which are matched to those pulse shaping filters in the modulator. After filtering, the I and Q components are sampled and their digital outputs processed to remove symbol timing and carrier phase offsets. Feedback from the digital section to the analogue downconverter is subject to additional digital to analogue conversion.


'baseband'
IF
X
LO
90°
Analogue Filter (Matched)
A/D
D/A
I
Q
Carrier Phase Synchronisation
Symbol Timing Synchronisation
Clock
Data

a. Baseband analogue I/Q demodulator.


'baseband'
Low IF
A/D
X
LO
90°
Digital FIR Filter (Matched)
I
Q
Carrier Freq/Phase Synchronisation
Symbol Timing Synchronisation
Clock
Data

b. Digital low IF demodulator.

Figure 1-3. Basic I/Q demodulator structures.

The front end of a digital low IF demodulator is shown in Figure 1-3b. The downconversion process is similar to the baseband I/Q structure, but the difference is in their implementation. Instead of mixing down to baseband in the analogue domain, the signal is mixed down to a low frequency IF and mixed to baseband in the digital domain. The A/D must therefore have sufficient bandwidth and signal to noise ratio to sample the wideband signal so that performance is not degraded over the frequency range of operation. Another difference is that receive filtering is conducted in the digital domain and can therefore be perfectly matched with the transmit filters. In a noiseless environment no ISI will occur between adjacent symbols, and digital filters provide the additional advantage of constant group delay. As with the modulator, the low frequency IF demodulator eliminates many analogue components making performance more predictable and repeatable. In terms of digital components, only one A/D is required but with somewhat greater performance requirements. Digital downconversion may be realised efficiently using look-up tables (LUTs) and the digital filters with finite impulse response (FIR) digital filter structures, carrier frequency and phase synchronisation can be conducted completely in the digital domain. As with the modulator, the low IF demodulator structure is also completely digital giving potential for functions to be reordered and novel architectures to be developed. Novel architectures and refinements to downconversion, filtering, carrier frequency synchronisation, symbol timing synchronisation and many other areas are discussed throughout the Thesis.

1.1.3 Software Radio

The low IF digital modem structures described previously are commonly implemented with discrete digital hardware but recent advances in Digital Signal Processor (DSP) technology are now beginning to offer a software-based alternative. The main advantage of software-based modems over custom hardware and application specific integrated circuits (ASICs) is that of flexibility and upgradeability. A software implementation also allows generic hardware to be developed which can be tailored to specific applications. This software can be replaced at any time when improvements are made or if changes are required, and such updates can even be applied automatically over-the-air.

Building on the above, the software radio concept is summarised in [1] as follows; *"Software radio in an emerging technology to build flexible radio systems which are multi-service, multi-standard, multi-band, reconfigurable and reprogrammable by software"*. Current modems employ an analogue RF front end and digital processing only in the latter stages, the 'ideal' software radio moves the boundary between analogue and digital to the RF input and performs <u>all</u> processing with software in the digital domain, Figure 1-4.

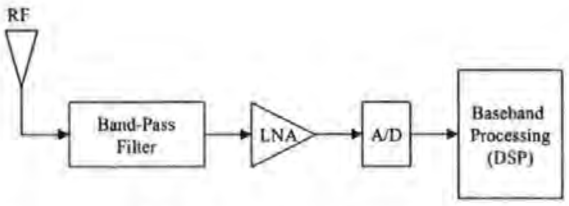


Figure 1-4. Ideal 'Software Radio' receiver.

The 'ideal' software radio transceiver is unrealisable with today's technology [2] and Figure 1-4 is considered as a long term target. The main limitations are associated with A/D and digital signal processor technologies. In the former case jitter performance is currently unacceptable at high sample rates and input resolution must be sacrificed as speed increases. ie. 6-bit to 8-bit resolution at Gsamples/second, 10-bit resolution for Msamples/second and 16-bit resolution for ksamples/second. In the latter case today's signal processors have relatively long instruction cycle times and cannot yet achieve real-time operation at the highest sample rates.

From the literature [1]-[20] it is clear that the 'ideal' software radio is not expected by many researchers in the field for another decade, when it is predicted that technology will have advanced sufficiently. The first step is to employ flexible analogue front-ends where the selection of operating frequency, channel and bandwidth can be made under software control, and with software-based signal processing between baseband frequencies and a low IF. Numerous researchers are working on enabling technologies including D/A, filtering and receiver architectures [13]-[18] so that, as technology improves, the boundary between analogue and digital can gradually shift towards the RF input. Other researchers are striving to define standards for the implementation of software modules so that a consistent and flexible interface is employed throughout [3]. Part of this research includes protocols for software download which will allow over-the-air updates to be conducted either on-demand or automatically as required [5]. Other topics of research include mobility management, security and user applications. The author's main interests are in novel real-time software architectures for combined modulation, demodulation, frequency synchronisation, timing synchronisation and FEC based upon the low IF structures described previously. The work in this Thesis also explores compromises to optimum theoretical performance which are required to achieve real-time operation with today's DSP technology and identifies suitable applications.

1.2 Application of Software-Based Signal Processing in Low-Cost Satellite Return Links

Where high data rates (>>1Mbps) are concerned, ASICs and (or) custom digital hardware is required to achieve real-time execution of complex signal processing algorithms. At lower data rates (<1Mbps) Digital Signal Processors and software-based algorithms can also achieve real-time execution and offer increased flexibility. Applications for software-based modems must therefore combine a requirement for complex signal processing with a relatively low data throughput. The following provides a brief introductory background to one such application in low-cost satellite return links.

1.2.1 Introduction to Low-Cost Satellite Return Links

The use of MPEG-2 compressed digital TV provides increased channel capacity in comparison to conventional analogue satellite broadcasting and allows a greater range of services to be provided. These new services can include video on demand, home shopping, distance learning, delivery of Internet data and a wealth of other interactive multimedia applications. These services are typically asymmetric with large amounts of data sent from the service provider to the user but with little, or nothing, sent from the user in return. To provide a fully interactive service, and to allow true user interaction, a return channel from the user to the service provider is required. A return channel can be provided terrestrially by the public switched telephone network (PSTN), the integrated services digital network (ISDN), cable modem, dedicated terrestrial circuit or by radio link. All terrestrial return channels have the requirement for an existing communications infrastructure, which may not be available due to local investment or geographic location.

Another solution, which has none of the limitations associated with terrestrial return channels, is to provide a satellite data return channel, Figure 1-5. A satellite return channel can be provided anywhere within a satellite's footprint and has no requirement for an existing local communications infrastructure.

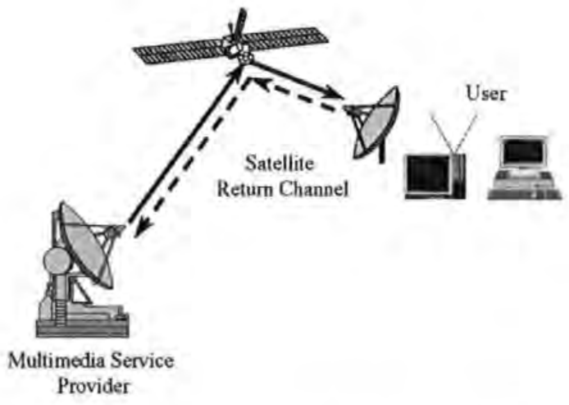
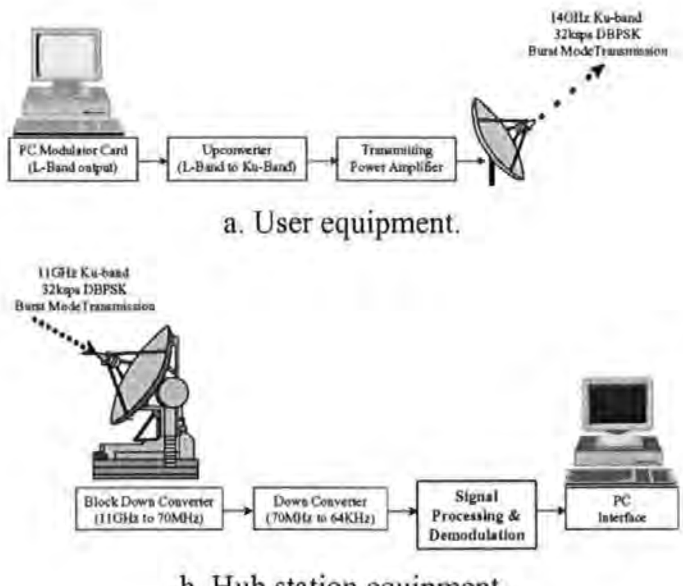


Figure 1-5. Interactive satellite DVB system with satellite return channel.

1.2.2 Signal Processing Requirements in Low-Cost Satellite Return Links

The cost of conventional Very Small Aperture Terminal (VSAT) technology is currently prohibitive and part of the work in this Thesis was driven by the desire to provide low cost satellite data reply links using a conventional Television Receive Only (TVRO) antenna equipped with a small transmitting power amplifier, Figure 1-6a. The frequency stability of this equipment is typically not very good and, as a consequence, signal processing algorithms are required at the Hub Station to compensate, Figure 1-6b. Other low-cost satellite links can be provided using Low or Medium Earth Orbiting (LEO or MEO) satellites where the requirements for signal processing differ.


14GHz Ku-band
32kbps DBPSK
Burst Mode Transmission
PC Modulator Card (L-Band output)
Upconverter (L-Band to Ku-Band)
Transmitting Power Amplifier
11GHz Ku-band
32kbps DBPSK
Burst Mode Transmission
Block Down Converter (11GHz to 70MHz)
Down Converter (70MHz to 64KHz)
Signal Processing & Demodulation
PC Interface

a. User equipment.

b. Hub station equipment.

Figure 1-6. Outline Ku-band satellite data reply link system.

1.2.3 Satellite Internet Delivery Systems

A great deal of interest is currently being shown towards high-speed delivery of Internet data by satellite. Benefits of such systems include higher download speeds, alleviation of congestion on terrestrial network segments and the ability to simultaneously reach large and widely separated user groups. Some are adopting the multicast approach where the most popular Web pages are broadcast to all terminals and stored (cached) locally for faster access. In these systems the user may be required to form a temporary PSTN connection in order to request information that has not been cached. Other systems, including investigations involving the author [53], employ a low-cost satellite return link to provide a 'permanently connected' return channel. In these systems the author's main interests are with channel access protocols, to allow request channels to be shared efficiently by multiple users, and transport protocols for mapping TCP/IP traffic to low-cost satellite links. Novel systems and protocols are discussed in detail later in the Thesis.

1.3 Organisation of Thesis

Chapter 2 is concerned with frequency synchronisation in modems for use with LEO (Low Earth Orbiting) microsatellites. These investigations were funded by I.S.T. (Instituto Superior Technico) of Lisbon, Portugal in relation to the PoSAT-1 microsatellite [33]. The main focus for these investigations was tracking the Doppler frequency error experienced at the demodulator input, providing correction within the demodulator and also providing pre-correction for the uplink at the modulator output. Algorithms are presented in this Chapter which satisfy these aims and are suitable for real-time implementation on TMS320C50 fixed-point DSPs. To allow the synchronisation algorithms to be evaluated both modulator and demodulator algorithms were also investigated. Many optimisations and compromises are required to achieve real-time execution, hence the Chapter is also concerned with efficient algorithms for frequency correction, frequency modulation, frequency discrimination and data shaping filters. The algorithms were verified in back-to-back simulations and the results are discussed at the end of the Chapter. A Doppler tracking modem was implemented using four DSPs and is described in Appendix B.

Chapter 3 is concerned with DSP software-based burst demodulator algorithms, and in particular with rapid carrier frequency acquisition and symbol timing synchronisation. In contrast to Chapter 2 these investigations were conducted with respect to Geostationary satellites and employ differential phase modulation. These investigations were funded in part by BNSC (The British National Space Council) and ESA (The European Space Agency) and satellite trials were conducted with collaboration from Eutelsat. Algorithms for a burst demodulator are discussed in Chapter 3 which satisfy the aims of the investigation and are evaluated in both back-to-back and satellite trials towards the end of the Chapter. Carrier frequency acquisition and symbol timing synchronisation is achieved after 32 symbols have been received with the use of a modified FFT algorithm, known as the Offset-FFT [36], and a 'delay and multiply' symbol clock recovery technique based upon an IIR resonator filter [42]. For operation at low SNR it is necessary to employ FEC over the satellite channel, but it was found that a Viterbi decoder algorithm for a standard constraint length 7 ½-rate convolutional code represented a significant computation overhead. A modification to the standard Viterbi decoder algorithm is also discussed which reduces computation overhead to acceptable levels. Part of the work from this Chapter was published in (Slader et al., 1996), Appendix H, and the complete burst demodulator is described in Appendix D.

Chapter 4 is concerned with OFDM (Orthogonal Frequency Division Multiplexing) as a means to achieve higher data throughput for the equivalent DSP computation capacity. These investigations were funded by EPSRC (Engineering and Physical Sciences Research Council). The aims for these investigations were to identify suitable synchronisation algorithms to form the basis of a coherent software-based OFDM modem for use in satellite applications. The Chapter begins with an overview of OFDM modems, and a mathematical comparison between real-sampling and complex-sampling configurations is made in Appendix E. Carrier frequency and symbol timing synchronisation are critical with OFDM and the effects of frequency error and symbol timing error are explored analytically within the Chapter. The Offset-FFT [36] is identified as a means to achieve simultaneous demodulation and frequency correction which allows synchronisation algorithms to operate on the demodulated data directly. Low complexity synchronisation algorithms are derived which, in conjunction with an optimised preamble sequence, are suitable for DSP software implementation. These algorithms are evaluated by simulation and results are presented at the end of the Chapter.

To confirm the relevance of the work in this Thesis Chapter 5 is concerned with a practical application for DSP software-based modems in asymmetric satellite Internet delivery systems. The author has contributed to investigations into asymmetric satellite Internet delivery systems employing both terrestrial and satellite data reply channels. In the latter case, techniques from Chapter 3 were successfully utilised. The Chapter begins with a generic overview of Internet delivery systems and defines symmetrical and asymmetric system configurations. This introduction is complemented by a review of the Internet Protocol Suite and IP networking concepts in Appendix F. A novel asymmetric satellite Internet delivery system is described along with the software techniques employed to allow IP transmission over a satellite data reply channel. The delays of each satellite link impose a fundamental limitation on the average transfer rate which can be achieved in satellite based TCP/IP networks. These limitations are discussed and modifications to overcome these limitations are reviewed. The most important components of the system are the low-speed satellite return channels which must be shared efficiently by the users without contention (collision). The final section of Chapter 5 identifies characteristics of the return channel transmission and evaluates the effectiveness of frequency-hopping and demand-assigned TDMA channel access mechanisms. Two publications resulting from this work, (Smithson et. al., 1996) and (Slader et. al., 1998), can be found in Appendix I and Appendix J respectively.

In Chapter 6 the work in this Thesis is summarised, conclusions are drawn and topics for further investigations identified.

2. A Doppler Tracking Modem for LEO Microsatellites

The PoSAT-1 satellite [33] is one of a series of Polar orbiting LEO (Low Earth Orbiting) microsatellites manufactured by the University of Surrey. These typically contain a store and forward processing payload which allows messages to be transmitted to the satellite as it passes overhead and downloaded on request by a receiver in another part of the world. The data rate to and from the satellite is 9.6kbps and employs frequency modulation. As a result of the Doppler frequency shift experienced by the receiver, due to the satellite's orbital velocity, the centre frequency of the signal received from the satellite may be shifted by ±10kHz. This is a significant amount relative to the bandwidth of the signal and must be corrected prior to demodulation. Similarly, transmissions to the satellite must be pre-corrected (tuned in the opposite direction) so that the centre frequency is correct when received by the satellite. Both Doppler tracking and modulator/demodulator functions are suitable for implementation with DSP software due to the increased flexibility offered. Funding from I.S.T. (Instituto Superior Technico) Lisbon, Portugal was provided to investigate DSP algorithms for a Doppler-tracking modulator and demodulator for use with PoSAT-1 and similar LEO microsatellites [32]. This research was concerned specifically with frequency synchronisation and Doppler tracking, and a two-stage frequency synchronisation process was devised involving FFT-based algorithms for coarse frequency synchronisation and feedback within the demodulator to achieve fine frequency synchronisation (Doppler tracking); sections 2.1.1 and 2.1.2 respectively.

For evaluation of the frequency synchronisation algorithms it was necessary to implement both 9.6kbps FM modulator and demodulator [34]. The FM modulator was implemented with software on a single DSP while the FM demodulator was implemented with software on dual-DSP hardware; due to the intensive nature of the demodulator algorithms. Descriptions of the modulator and demodulator and signal plots from each stage of the software are presented in Appendix B. In order to achieve real-time execution, processing overheads were minimised with a combination of optimised software routines and by making carefully selected compromises to theoretical performance, the main techniques employed are discussed in section 2.2. Pre-Doppler compensation is introduced within the modulator and Doppler correction is applied to the demodulator using a synthesised local oscillator. In section 2.2.1, the manner in which the local oscillator is implemented is shown as a means to reduce processing overhead. The system employs matched root 40% raised cosine filtering, which would generally require a FIR filter with many (hundreds of) coefficients. To reduce processing overhead this was implemented with a pre-computed LUT in the modulator and with an IIR filter approximation in the demodulator, section 2.2.2. The most intensive portion of the FM demodulator was frequency discrimination, and a phase detector implemented with a LUT is discussed in section 2.2.3. Finally, in section 2.2.4, the use of block sample processing and selective sub-sampling is shown as an effective means of dramatically reducing processing overhead. Results and conclusions are presented in sections 2.3 and 2.4 respectively.

2.1 Frequency Synchronisation and Doppler Tracking

The algorithms investigated employ a dual approach to Doppler frequency correction and tracking. In summary, an FFT algorithm is proposed for coarse approximation of instantaneous Doppler error and also as a means of rapidly acquiring initial frequency lock. This alone is not sufficiently accurate to fully correct Doppler error, so a further fine Doppler tracking process is required using feedback from the FM demodulator output in proportion to the average value of the bipolar data; frequency error manifests as a DC offset. Comparing both coarse and fine tracking signals allows the case of lost frequency lock to be detected and for re-synchronisation to be rapidly conducted. The Doppler error is corrected in the demodulator during frequency correction and pre-Doppler compensation is applied to the modulator during frequency modulation (see section 2.2.1). In both cases, the correction is introduced by a fixed adjustment to the frequency of a synthesised local oscillator. For pre-compensation in the uplink path, the modulator is tuned in the opposite direction to the demodulator.

2.1.1 Coarse Frequency Synchronisation

Coarse frequency synchronisation may be conducted using an FFT. In general, the frequency resolution of this technique is determined by the FFT length N and the sampling frequency f_s. The FFT bin spacing Δf is given by

$$\Delta f = \frac{f_s}{N} Hz \tag{2-1}$$

and the maximum estimation error from the FFT is one half of Δf. For a system with f_s = 307,200 Hz and 256-point FFT, the maximum estimation error is 600Hz. For a transmitted signal with centre frequency f_0 Hz, the nominal centre bin k_0 is given by

$$k_0 = \frac{f_0}{\Delta f} \tag{2-2}$$

For a system with f_0 = 52,000 Hz and Δf = 1,200 Hz, k_0 = 43 (to the nearest integer). If Doppler frequency shift is present the indicated centre frequency bin k_1 is shifted in frequency from its nominal position giving the frequency bin error Δk

$$\Delta k = k_1 - k_0 \tag{2-3}$$

If the signal is received at a higher frequency than expected, Δk is positive and the demodulator must be tuned to a higher frequency. Conversely, Δk is negative when the signal is received at a lower frequency than expected, and the demodulator must be tuned to a lower frequency. A coarse frequency synchronisation algorithm based upon the peak FFT frequency bin is sufficient to determine the frequency of an un-modulated carrier, but not for a carrier which is frequency modulated.

For a digital demodulator it is essential that the sampling frequency f_s Hz is an integer multiple of the transmitted bit rate f_b Hz. Ideally the FFT length will be sufficiently long to span many (>100) bit periods $1/f_b$, and to produce a frequency spectrum which is independent of the transmitted bits (random data assumed). If the FFT length spans a few bit periods only, the resulting spectrum will be influenced heavily by the underlying modulation and large variations between successive measurements can be introduced. For a system with f_b = 9.6 kHz, f_s = 302,700 Hz (f_s = 32×f_b) and FFT length N = 256, the FFT spans just 8 bit periods. For randomised data over 8 bit periods, it is highly likely that the instantaneous distribution of logic '1' and '0' will also cause variations to the indicated centre bin k_l over successive measurements. The effects of the underlying modulation on the coarse frequency estimation algorithm may be reduced if many more bit periods are considered. Improved performance (better averaging) results when the FFT length is increased to span more bit periods. However, finite DSP memory and processing power impose limits to practical improvements obtainable in this manner.

A computation efficient, but memory intensive, solution is proposed to reduce the effects of instantaneous data sequences in the situation described above, Figure 2-1. This solution considers the individual FFT power spectra from several successive measurements to produce a 'running average' spectrum with more accurate indication of the centre frequency bin k_l. The memory requirements of this algorithm are increased by a factor of M, since the last M spectra must be stored in order to efficiently maintain the 'running average' spectrum. For a system employing a 256-point FFT, the power spectrum is fully represented by 256 memory locations. If M = 64, then 16,384 memory locations will be required; this requirement may be halved by considering that the power spectrum is symmetrical for a real input signal. A second characteristic of this algorithm is that the first estimate cannot be produced until such time as M FFTs have been computed; subsequent 'running average' estimates may be produced once each additional FFT has been computed.

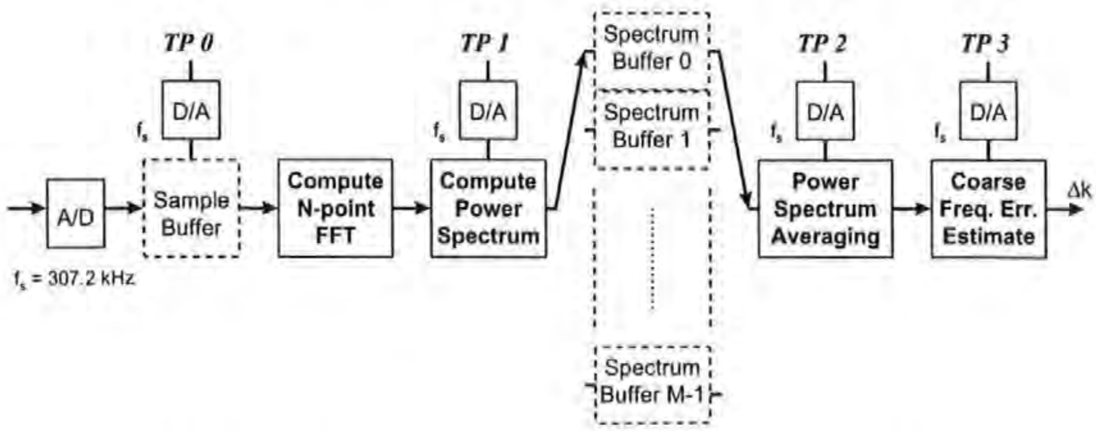
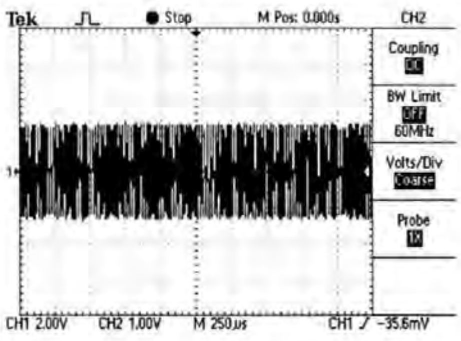
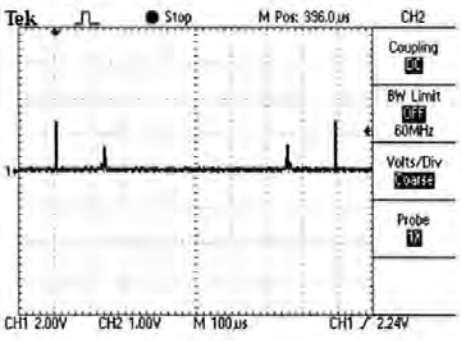


Figure 2-1. Coarse frequency synchronisation sub-system (*N*=256, *M*=64).

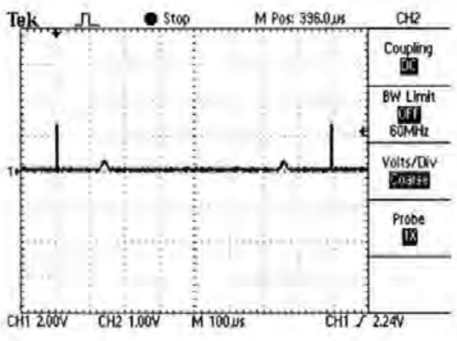
Coarse frequency synchronisation DSP software is depicted in Figure 2-1 as a block diagram most closely representing the author's implementation. The input signal is sampled and the resulting signal samples written to a sample buffer. For each block of N samples, an N-point FFT is performed and the power spectrum computed. The most recent M spectra are stored in separate buffers so that a 'running average' spectrum can be maintained; the sample buffers are addressed in a circular manner. The frequency error estimate Δk is derived from the averaged power spectrum using equation (2-3) , which limits the theoretical frequency resolution to half the frequency bin spacing Δf Hz. The test points TP 0 - TP 3 allow particular samples to be monitored by an oscilloscope, typical signals from these test points are shown in Figure 2-2a to Figure 2-2d and are discussed below.



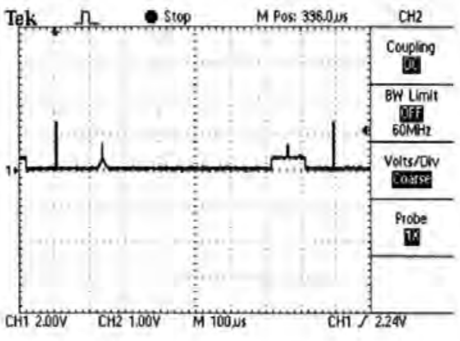
a) TP0 - 52kHz input signal.



b) TP1 - 256-point power spectrum (0 to f_s).



c) TP2 - Averaged spectrum (64 FFTs).



d) TP3 - Location of centre bin k_l.

Figure 2-2. Coarse frequency synchronisation signals (N = 256, M = 64).

Figure 2-2a to Figure 2-2d show typical signals measured at the four test points of the coarse frequency synchronisation sub-system (Figure 2-1). Figure 2-2a shows the input signal at TP 0 which is a 52 kHz carrier frequency modulated by root 40% raised cosine filtered data at 9.6kbps (±5 kHz *deviation*). The remaining signals represent FFT power spectra, and the largest peaks have been inserted as a means to trigger the oscilloscope and to indicate frequency bin 0. Figure 2-2b shows the FFT power spectrum for a typical input signal where the peak frequency bin is influenced heavily by the underlying modulation and the 8 data bits received during the FFT input capture period. In Figure 2-2c 64 such spectra have been averaged, hence smoothing the variance, so that 512 bit periods are considered. This spectrum provides better averaging and provides a more reliable indication of the true centre frequency. Figure 2-2d shows the averaged spectrum after it has been modified by two different centre bin location algorithms. For the first algorithm, over the range 0 to $f_s/2$ Hz, the centre bin is selected as the frequency bin containing maximum power. For the second algorithm, over the range $f_s/2$ Hz to f_s Hz, those bins which exceed a pre-set SNR are highlighted and the centre bin is selected such that it lies at the mid point. In practice it was found that the second method of determining the centre bin produced more stable and consistent results (see Figure 2-3). Improvements to performance result if M is increased, but at the expense of greater memory requirements and slower acquisition times. For fixed-point DSP implementation it is most convenient if $M=2^K$ (K integer) so that scaling can be achieved without software overhead using a 'bit-shift' operation.

The first 4096 outputs from the two centre bin location algorithms are plotted in Figure 2-3 for an ideal noise free input (random data) over an interval of 131,072 bit periods (4096 FFT periods). The test signal was generated by software routines, applied digitally to the DSP hardware, and results were captured by diagnostic software developed by the author for this purpose. In this case, the nominal centre bin k_l = 43, but the effect of the underlying modulation is to introduce variations of ±2 bins to the value of k_l indicated by algorithm 1 and variations of ±1 bins to the value of k_l indicated by algorithm 2. The superior performance of the second algorithm was confirmed by statistical analysis of a population of 16384 samples. The standard deviation for algorithm 1 was found to be 0.683036 while this improved to 0.499843 for algorithm 2; in both cases the mean value indicated was $k_1 \approx 43$. It was found that AWGN added to the input signal produced no significant degradation at CNR above the level at which the demodulator naturally exhibits high bit error rates. Further results are presented at the end of this chapter in section 2.3.

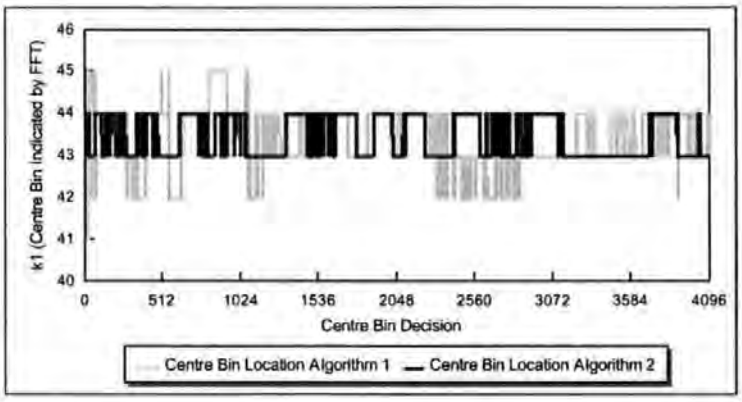


Figure 2-3. Coarse frequency synchronisation performance with ideal noise-free input signal - $k_1 = 43, N = 256$ & $M = 128$.

2.1.2 Fine Doppler Tracking

The FFT signal alone is not accurate or stable enough to provide instantaneous Doppler tracking, so feedback within the demodulator is proposed to finely measure residual Doppler error. Residual Doppler error manifests as a DC offset in the average level of the demodulator output (random data assumed). If the signal is received at a higher frequency than expected, a positive DC offset is introduced. Likewise, a negative DC offset is generated at the output if the signal is received at a lower frequency than expected. For similar reasons to those given in section 2.1.2, the DC offset measured at the demodulator output is influenced heavily by the underlying modulation and data transmitted, and many (>100) bit periods should be considered. If coarse frequency synchronisation is conducted by averaging M FFTs it is acceptable (and desirable) to also measure the DC offset over an equivalent period; i.e. for an N-point FFT and sampling frequency f_s Hz, to measure the demodulator output DC offset over $N \times M / f_s$ seconds. For $f_s = 307{,}200$ Hz, $N = 256$, $M = 64$ and a transmitted data rate of 9.6kbps, this duration is equivalent to 512 bit periods.

The resolution of fine Doppler tracking is determined by the frequency resolution of a synthesised local oscillator within both the modulator and demodulator. If the local oscillator is implemented with a 'phase accumulator' technique, and employs a pre-computed Cosine table of length L, the frequency resolution δf is given by

$$\delta f = \frac{f_s}{L} Hz \tag{2-4}$$

The residual Doppler error resulting from the measured numerical offset n_1 at the demodulator output may be corrected by an increase to the Cosine table index of id_d. If it is known that a numerical offset n_0 results at the demodulator output for a frequency error of 1000 Hz, id_d is given by

$$id_d = \frac{n_1}{n_0} \cdot \frac{1000}{\delta f} \tag{2-5}$$

Doppler tracking is provided by accumulating measurements of the DC offset n_1 at the demodulator output. To prevent oscillation (over correction), n_1 should be scaled by factor r ($0 < r < 1$) so that tracking occurs more slowly. Doppler pre-compensation is achieved with an adjustment to the modulator's Cosine table index of $K.id_d$, where K is negative and accounts for other factors such as the up-link to down-link frequency ratio. A block diagram of a Doppler tracking sub-system, along with inter-connections to the modulator and demodulator, is shown in Figure 2-4. The 'Doppler Tracker' in Figure 2-4 integrates $r.n_1$.

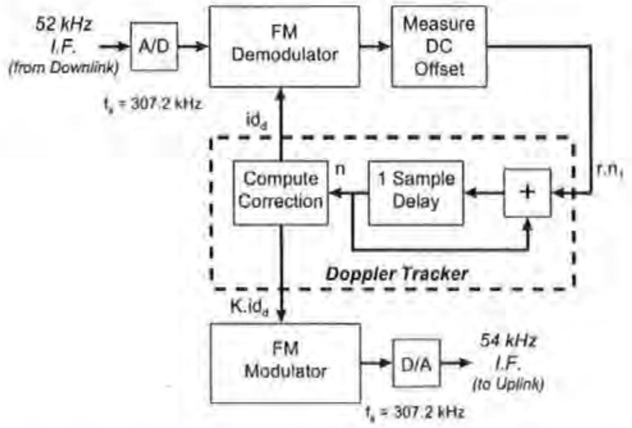


Figure 2-4. Fine Doppler tracking for modulator & demodulator.

Once initialised, the fine Doppler tracking technique proposed above is sufficient to track Doppler shifts over a much larger range than might be experienced in practice; eg. the author's implementation tracked Doppler shifts of ±48 kHz where shifts of only ±10 kHz were anticipated. Assuming that the system is initialised with Doppler correction set for nominal conditions (no Doppler shift), this technique also has a useful acquisition range within which the algorithm can adapt unaided. The acquisition range is limited by the characteristic of the demodulator and the level of un-corrected Doppler shift that can be tolerated before distortion is introduced at the output. This is particularly relevant for fixed-point DSP implementation, where large un-corrected Doppler shift introduces large DC offset at the demodulator output and may result in numerical overflow (severe distortion). By utilising both coarse frequency synchronisation and fine Doppler tracking techniques simultaneously, the benefits of each may be exploited. The solution implemented is shown as a block diagram in Figure 2-5 and is discussed below.

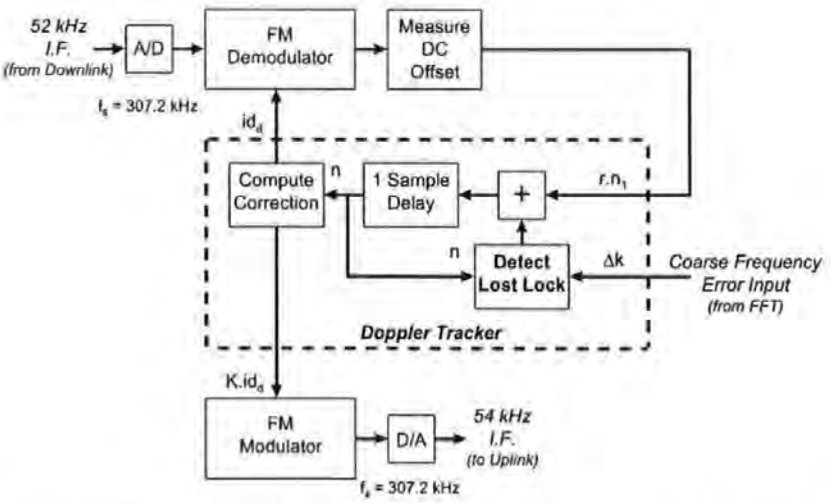


Figure 2-5. Practical fine Doppler tracking for modulator and demodulator with automatic (re)initialisation from coarse frequency synchronisation FFT sub-system.

With respect to Figure 2-5, operation is similar to the explanation already given for Figure 2-4. In this case, an extra function compares the level of Doppler correction applied by the tracker *n* with the coarse frequency error measured from the FFT algorithm *Δk*. If the tracker is locked to the frequency of the signal received from the down-link, both *n* and *Δk* will indicate similar frequency errors and *n* may be fed back to the accumulator. If the tracker is not locked to the frequency of the received signal, the frequency error indicated by n and Δk will be sufficiently different. Lost frequency lock may be declared, for example, when the difference between respective frequency error estimates exceeds the natural (unaided) acquisition range of the tracker. If frequency lock is lost, the tracker may be immediately re-initialised by the coarse frequency error estimate Δk; ie. during power-up initialisation or as a satellite first appears over the horizon.

In Figure 2-6, the Doppler error indicated by the numerical offset n_I at the demodulator output is plotted over a 3 second interval for a noise-free and Doppler-free input signal. The results were first plotted with Doppler tracking inactive, to observe variations due to the underlying modulation, and then with Doppler tracking active, to measure any loss of stability. Statistical analysis of a population of 16384 samples show that the standard deviation was 9.749Hz with Doppler tracking inactive and 9.817Hz with Doppler tracking active; with mean values of 0.779Hz and 2.947Hz respectively. It is clear that the effect of the underlying modulation and instantaneous data sequence is to introduce variations of up to ±50Hz in the Doppler error indicated and that Doppler tracking introduces a marginal performance degradation. This behaviour can be attributed to rounding errors in the fixed-point implementation of the demodulation and tracking algorithms. Further results are presented towards the end of this chapter in section 2.3.

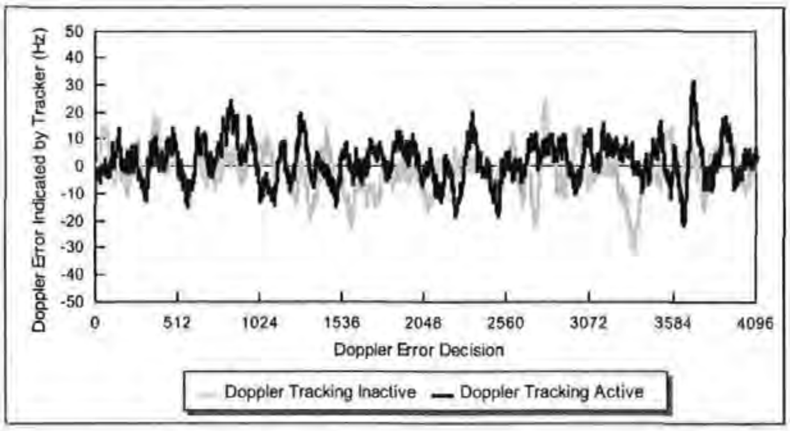


Figure 2-6. Doppler error indicated at tracker input for ideal noise-free & Doppler-free input signal.

2.2 *Methods Employed to Minimise Processing Overheads*

Several optimisations were required to the modulator and demodulator algorithms in order to allow real-time implementation with the Texas Instruments TMS320C50 fixed-point DSP. The most significant optimisations are discussed in this section.

2.2.1 Frequency Modulation and Frequency Correction

A synthesised local oscillator is required in the FM modulator for frequency modulation and in the FM demodulator for frequency correction, and in both cases a 'phase accumulator' technique was employed, Figure 2-7.

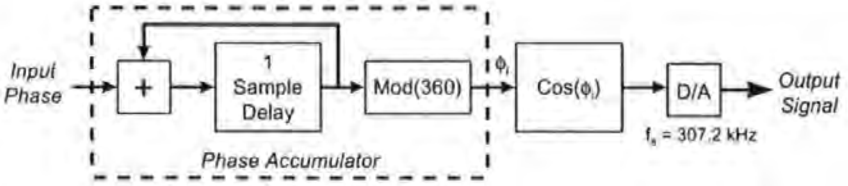


Figure 2-7. General 'Phase Accumulator' for frequency synthesis.

For practical and efficient implementation of a phase accumulator, the Cos function must be replaced by a pre-computed Cosine Table, and frequency synthesis is achieved by addressing the Cosine Table with the appropriate addressing increment or *index*. In this case it is more logical to refer to an 'index accumulator', Figure 2-8.

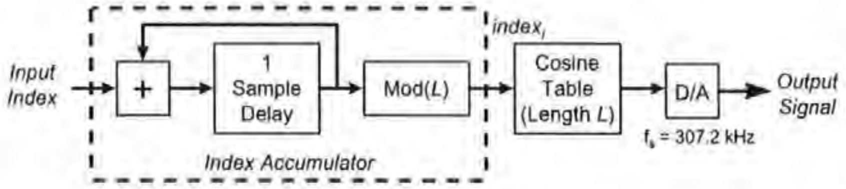


Figure 2-8. Practical 'Index Accumulator' for frequency synthesis.

It was found in practice that three factors must be considered; frequency resolution, memory utilisation and processing overhead. The frequency resolution δf of the 'index accumulator' in Figure 2-8 is determined by the sampling frequency f_s, the Cosine Table length L, and was given by equation (2-4) Memory utilisation L increases in direct proportion with improved (smaller) frequency resolution. With the TMS320C50 DSP instruction set, it was found that the 'index accumulator' can be most efficiently implemented when $L = 2^K$ (K integer) and when the Cosine Table is also located at an absolute memory address which is an integer multiple of 2^{K+1}. However, the frequency resolution desired does not guarantee that an optimum Cosine table length L can be employed and finite memory in the target hardware does not guarantee that the desired frequency resolution can be achieved.

The 'index accumulator' implemented for frequency modulation in the FM modulator has three inputs and a single output, Figure 2-9. The 'Carrier Frequency' input is a constant which sets the nominal (un-modulated) output frequency to the 54kHz required by the uplink. The 'Doppler Compensation' input is driven by Doppler tracking sub-system and provides a means to finely adjust the output frequency in opposition to the Doppler shift experienced by the demodulator. The final input is the modulating signal which is pre-scaled to produce the desired frequency deviation.

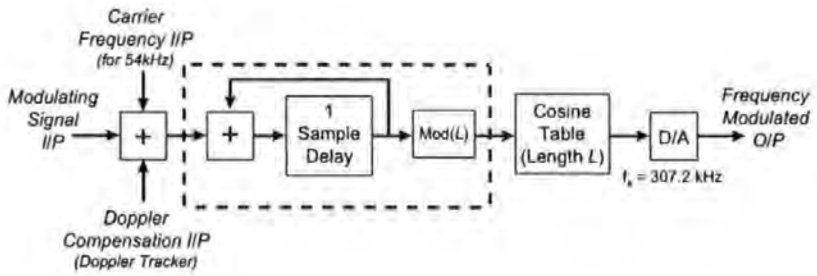


Figure 2-9. 'Index Accumulator' for frequency modulation - FM modulator.

The 'index accumulator' implemented for frequency correction in the FM demodulator has two inputs and two outputs, Figure 2-10. The 'Carrier Frequency' input is a constant which sets the 52kHz output frequency to match the nominal carrier frequency of the signal received from the downlink. The 'Doppler Correction' input is also driven by Doppler tracking sub-system and provides a means to finely adjust the output frequency to match the current level of Doppler shift experienced by the demodulator. Both Cosine and -Sine components are required at the output so that frequency correction may be conducted; this is achieved by addressing the Cosine table a second time but with a fixed $L/4$ (90°) addition to the index.

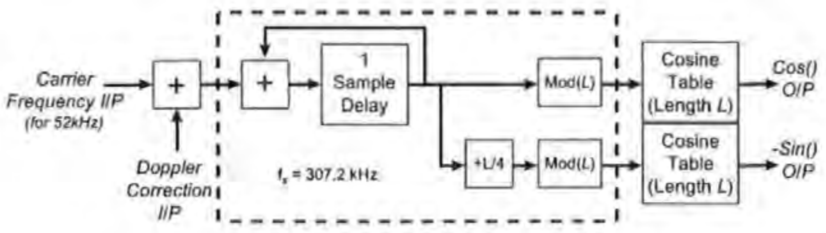


Figure 2-10. 'Index Accumulator' for frequency correction - FM demodulator.

Frequency resolution was considered ahead of processing efficiency in the modulator, and the Cosine Table length was set to achieve maximum frequency resolution within the constraints imposed by the target DSP's memory. It was found that L = 19,200 (δf = 16Hz) best satisfied these requirements and allowed all significant frequencies (54kHz & 52kHz) to be synthesised precisely. It was stated earlier that the 'index accumulator' can be implemented most efficiently when the Cosine Table length $L = 2^K$ (K integer) and when the Cosine Table is located at an absolute address which is an integer multiple of 2^{K+1}. Due to the intensive nature of the demodulator algorithms, Cosine Table length L = 16,384 (δf = 18.75Hz) and start address 32,768 (0x8000) was selected to minimise processing overheads. The significance of the Cosine Table length L can be clearly seen in the assembly code for frequency modulation and frequency correction, Figure 2-11 and Figure 2-12. To allow direct comparisons to be made, those instructions directly related to the 'index accumulator' are highlighted and non-essential housekeeping code has been removed. In the case of the frequency modulator assembly code (Figure 2-11) it can be seen that 10 instructions are required within the loop for the 'index accumulator' while only 4 are required in the frequency correction assembly code (Figure 2-12). In the former case, 6 instructions are required to implement the modulo(L) function while only 2 instructions are required in the latter case to perform the same operation twice. The effect of the 'optimum' length and absolute memory location for the Cosine table is to dramatically simplify implementation of the modulo(L) function. This is achieved because it is no longer necessary to test if a pointer has exceeded the Cosine Table's end address, and to subtract L when it does. Instead, when $L = 2^K$ and the start address is an integer multiple of 2^{K+1}, bit-K of the pointer may be reset after <u>each</u> pass of the loop and the (expensive) comparison avoided. Had the demodulator been implemented with a 'non-optimum' Cosine Table length, 10 extra instructions would be added to the frequency correction program loop. To add perspective, 10 single-cycle instructions represent approximately 10% of the average number of instruction cycles available for processing each input sample with the 80MHz TMS320C50 DSP at this sampling frequency (307.2 kHz).

```
FMMODULATOR .macro
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*!! AR1 - Pointer to Cosine Table
*!! AR2 - Pointer to Output Buffer
*!! L   - Cosine Table Length (19,200)
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    lacc    FMCARRIER       ;FM Carrier input
    add     FMDOPPLER       ;FM Doppler frequency compensation input
    sacl    TEMP            ;TEMP = FM Carrier + FM Doppler (for efficiency)

    splk    #(N-1),brcr     ;Repeat block N times
    rptb    FMLOOP?-1       ;Start of repeat block
     lacc    FMSIGNAL        ;Read FM Modulating signal input
     add     TEMP            ;add FM Carrier & FM Doppler inputs
     sacl    indx            ;Set COS Table Addressing Index
     lacl    *+,ar2          ;Read COS table
     sacl    *+,0,ar1        ;Write to output buffer
*!    Mod(L) addressing of COS table
     mar     *0+             ;Increment COS table pointer
     cmpr    2               ;Test if AR1 has passed end of COS table
     lacl    ar1             ;AR1 to ACCl
     xc      2,tc            ;If AR1 passed end of COS table
      subs    L               ;Subtract COS table length from ACCl
      sacl    ar1             ;Update AR1
FMLOOP?                     ;End of repeated block
  .endm
```

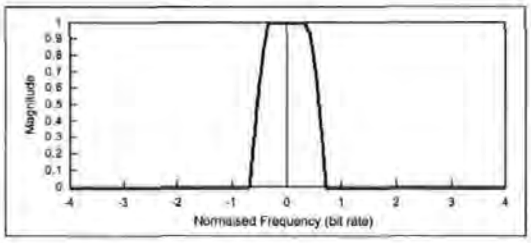
Figure 2-11. FM modulator DSP assembly code for frequency modulation - 'Index Accumulator' instructions highlighted.

```
FREQUENCYCORRECTION .macro
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*!! AR0 - Pointer to input buffer Xi (sampled signal received from downlink)
*!! AR1 - Pointer to Cosine Table    (generates  Cos() component)
*!! AR2 - Pointer to Cosine Table    (generates -Sin() component)
*!! AR3 - Pointer to I-Stream output buffer Ai ( Xi *  Cos() )
*!! AR4 - Pointer to Q-Stream output buffer Bi ( Xi * -Sin() )
*!! Cosine Table Length L = 16384, start address = 32768
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    lacc    FMCARRIER       ;FM Carrier input
    add     FMDOPPLER       ;FM Doppler frequency compensation input
    sacl    indx            ;Set COS Table Addressing Index

    splk   #(N-1),brcr      ;Set to repeat N times
    rptb    LOOP?-1         ;Start of repeat block
     lt      *+,ar1          ;Xi to TREG
     mpy     *0+,ar3         ;PREG := i COS()    - Index added to AR1
     sph     *+,ar2          ;Ai   := Xi COS()
     mpy     *0+,ar4         ;PREG := Xi -SIN()  - Index added to AR2
     sph     *+,ar0          ;Bi   := Xi -SIN()
*!    Mod(L) addressing of COS table
     apl     ar1             ;Mod(16384) for AR1 - 2 PLP Rqrd. (reset bit 14)
     apl     ar2             ;Mod(16384) for AR2 - 2 PLP Rqrd. (reset bit 14)
LOOP?                       ;End of repeat block

  .endm
```

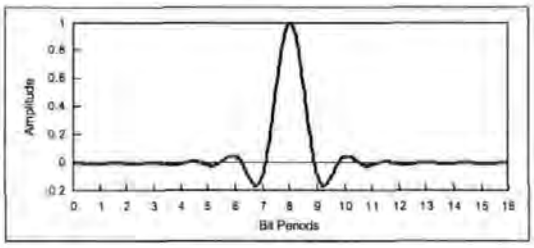
Figure 2-12. FM demodulator DSP assembly code for frequency correction - 'Index Accumulator' instructions highlighted.

2.2.2 Root 40% Raised Cosine Filtering

A root 40% raised Cosine filter was required for both modulator and demodulator, and the classical roll-off spectrum and impulse response for such a filter are shown in Figure 2-13. Typically, a filter of this type is implemented with a finite impulse response (FIR) filter which, due to the number of coefficients required, potentially represents a significant processing overhead.



a) Root 40% raised Cosine roll-off spectrum.



b) Root 40% raised Cosine filter impulse response.

Figure 2-13. Root 40% raised Cosine filter responses.

Using a 4096-point IFFT, up to 4095 filter coefficients may be computed for such a FIR filter. If this filter is to be implemented with a 16-bit fixed-point processor, the coefficients must be converted to Q15 format (integer representation) in order to minimise quantisation errors. For the filtered required, if the coefficients are truncated at the point where they can no longer be represented with Q15 notation, 345 coefficients will still remain; approximately 11 bit periods at 32 samples per symbol. With sampling frequency f_s = 307.2 kHz, there is on average approximately 3.26μs of DSP processing time per sample. The 80MHz TMS320C50 DSP provides 40 MIPS (40 Million Instructions Per Second) so only 130 single-cycle instructions can be performed within 3.26μs. If it is assumed that each MAC instruction (Multiply And Accumulate) can be conducted with a single instruction cycle, 345 instructions are required to implement the FIR filter alone. Clearly, real-time operation would not be possible at the specified sample rate with the 80MHz TMS320C50 DSP.

A root 40% raised cosine pulse shaping filter is required for the FM modulator. Processing overhead may be reduced to acceptable levels if the number of FIR filter coefficients is reduced, but with the penalty of a poorer approximation of the target filter. An alternative approach is to employ a pre-computed LUT which can be used to transform the outgoing data directly to shaped pulses for transmission. For a LUT, the quality of the approximation to the target filter is determined only by the available memory in the target hardware. In general, if the filter coefficients span n bit periods (n odd), the memory required to store such a LUT is given by

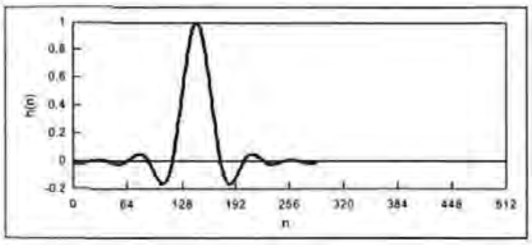
$$LUT\_length = 2^n \cdot \frac{f_s}{f_d} \quad locations \qquad (2\text{-}6)$$

where f_d is the data rate and f_s the sample rate. For the FM modulator implemented, n was constrained to 9 (*LUT_length* = 16,384 locations) by the target hardware.

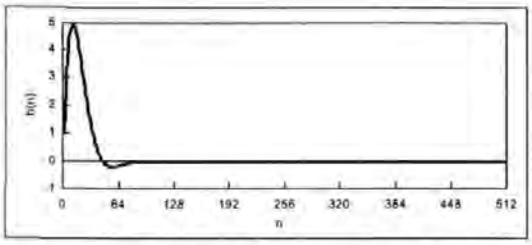
The FM demodulator also requires a root 40% raised Cosine filter which is matched to the modulator's pulse shaping filter. It has been shown that a FIR filter implementation represents significant processing overhead and a practical alternative to a FIR filter was also required. In the FM demodulator the matched filter was implemented with infinite impulse response (IIR) filter having impulse response I(z) and given by the z-transform expression

$$I(z) = \frac{1}{P(z)} \qquad (2\text{-}7)$$

An optimisation process was carried out to find the best IIR polynomial P(z) that matches the impulse response of the root 40% raised Cosine filter. Through experimentation, the result $P(z) = 1 - 1.86z^{-1} + 0.869\ z^{-2}$ was found to be the closest match. Impulse responses for the root 40% raised Cosine FIR filter (implemented with a LUT in the modulator) and the IIR approximation filter are shown in Figure 2-14.



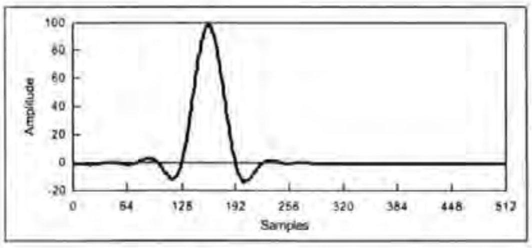
a) FIR filter implemented as look-up table - Modulator.



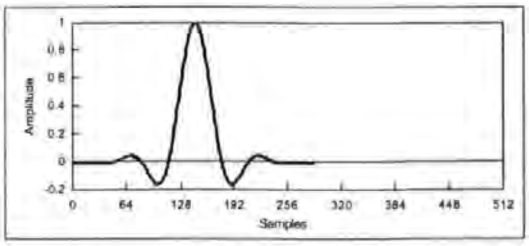
b) IIR filter approximation ($P(z) = 1 - 1.86z^{-1} + 0.869\,z^{-2}$) - Demodulator.

Figure 2-14. Impulse responses for practical root 40% raised Cosine filters.

The convolved response of the IIR filter with the root 40% raised Cosine impulse response is shown in Figure 2-15 in comparison with a 40% raised Cosine response. These two responses should be identical and it can be seen that, apart from a different delay, this is nearly the case.



a) Convolved response of IIR filter with root 40% raised Cosine response.



b) Impulse response of 40% raised Cosine filter.

Figure 2-15. Impulse responses for practical 40% raised Cosine filters.

In terms of software overhead, the difference between a FIR filter, LUT and IIR filter implementations are illustrated by the assembly code shown in Figure 2-16. To produce just one output sample, the FIR filter requires 295 instructions and occupies 288 locations in both program and data memory. To achieve comparable results, the LUT requires just 4 instructions but occupies 16,384 locations in program memory. Finally, the IIR filter requires 9 instructions and occupies less than ten memory locations in total. The LUT implementation is clearly preferable in terms of software overhead providing that sufficient free memory is available. In the demodulator, where the LUT cannot be used because the incoming bit sequence is not known, the IIR filter approximation provides the best compromise with which to achieve real-time performance.

```
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*!! ROOTRAISEDCOSINE_FIR      - Requires 7+288 instructions per output sample
*!!                           - FIR_LENGTH = 9*32 = 288 coefficients
*!!                           -  AR0 - Pointer to Input Buffer
*!!                           -  AR1 - LUT Index (Input data Sequence)
*!!                           -  AR2 - Pointer 1 to FIR Filter Data Memory
*!!                           -  AR3 - Pointer 2 to FIR Filter Data Memory
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     lacl    *+,ar2           ;Read  input (data impulse sample)
     sacl    *,0,ar3          ;Write input to first location of FIR shift register
     zap                      ;Reset Accumulator
*!!  NOTE: The MACD instruction must be repeated 288 times
     rpt     #(FIR_LENGTH-1)  ;Repeat next instruction FIR_LENGTH times
      macd    #FIR,*-          ;Multiply and Accumulate
     apac                     ;Accumulate result of last multiplication
     mar     *,ar1            ;ARP=1
     sach    *+,0,ar0         ;Store result (RC Pulse Sample) in output buffer


*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*!! ROOTRAISEDCOSINE_LUT      - Requires 4 instructions per output sample
*!!                           - LUT_Length = 16,384 locations
*!!                           -  AR0 - Pointer to Output Buffer
*!!                           -  AR1 - LUT Index (Input data Sequence)
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     lacl    ar1              ;Input to ACC (LUT index)
     adds    TEMP0            ;Start address of LUT added to ACC
     tblr    *+,ar1           ;Store result (RC Pulse Sample) in output buffer
     mar     *+,ar0           ;Increment counter


*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*!! ROOTRAISEDCOSINE_IIR      - Requires 9 instructions per output sample
*!!                           -  AR0 - Pointer to Input Buffer
*!!                           -  AR1 - Pointer to IIR Filter memory
*!!                           -  AR2 - Pointer to Output Buffer
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     lacc    *0+,16,ar1       ;ACCh = x(n)
     lt      *+               ;y(n-2) ¯ TREG
     mpy     COEFF1           ;K1*y(n-2)/2 ¯ PREG
     lts     *                ;y(n-1) ¯ TREG, x(n) - K1*y(n-2) ¯ ACCh
     add     *+,16            ;x(n) + y(n-1) - K1*y(n-2) ¯ ACCh
     mpy     COEFF0           ;y(n-1) * K0  ¯ PREG
     apac                     ;[x(n) + (1+K0)y(n-1) - K1y(n-2)] ¯ ACCh
     sach    *+,0,ar2         ;y(n) := [x(n) + Cy(n-1) - K1y(n-2)]
     sach    *+,0,ar0         ;Set output
```

Figure 2-16. DSP assembly code required to produce one output sample for 'FIR Filter', 'Look-up Table' & 'IIR Filter Approximation' implementations of a root 40% raised Cosine filter.

2.2.3 Frequency Discrimination

The FM demodulator receives from the downlink a 52kHz I.F. which is sampled at 307.2 kHz. The input signal samples are frequency corrected and low pass filtered to produce in-phase and quadrature sample streams with 0 Hz (baseband) centre frequency. Since the transmitted signal is frequency modulated, the transmitted data signal must be recovered with a frequency discriminator. The frequency discriminator was implemented as a phase detector followed by a differential phase detector. For the FM demodulator implemented, frequency discrimination represents the greatest software overhead.

When both in-phase and quadrature sample streams (I_i and Q_i) are provided at the input to the phase detector, the instantaneous phase may be found with the extended arc Tangent function (0° to 360°) given by

$$phase_i = \begin{cases} 000°+Tan^{-1}\left(\frac{Q_i}{I_i}\right) & if\ quadrant=0 \\ 180°+Tan^{-1}\left(\frac{Q_i}{I_i}\right) & if\ quadrant=1 \\ 180°+Tan^{-1}\left(\frac{Q_i}{I_i}\right) & if\ quadrant=2 \\ 360°+Tan^{-1}\left(\frac{Q_i}{I_i}\right) & if\ quadrant=3 \end{cases} \tag{2-8}$$

For the TMS320C50 DSP, 16-bit division can only be conducted when both numerator and denominator are positive integers; hence when the result is also positive. It is therefore necessary to take the magnitude of each input and to provide correction based upon the input quadrant. A new extended arc Tangent function (-180° to 180°), more suited to fixed-point DSP implementation, is given by

$$phase_i = \begin{cases} +000°+Tan^{-1}\left(\frac{|Q_i|}{|I_i|}\right) & if\ quadrant=0 \\ +180°-Tan^{-1}\left(\frac{|Q_i|}{|I_i|}\right) & if\ quadrant=1 \\ -180°+Tan^{-1}\left(\frac{|Q_i|}{|I_i|}\right) & if\ quadrant=2 \\ +000°-Tan^{-1}\left(\frac{|Q_i|}{|I_i|}\right) & if\ quadrant=3 \end{cases} \tag{2-9}$$

It is not possible for a DSP's instruction set to support Tan^{-1} directly, and a LUT provides the most efficient solution. Given unlimited memory for the LUT, it would be possible to address the table with a concatenation of I and Q, and to obtain the result directly. With finite memory, the division of $|Q|$ by $|I|$ must be performed first and a (smaller) LUT addressed with the result. Tan^{-1} is not ideally suited to LUT implementation because it is a non-liner function. Potential problems are best demonstrated with a LUT where the input is the division of $|Q|$ by $|I|$ (integer) and the output is its arc Tangent, Table 2-1. Such a LUT must span the output range 0° to 90° and it can be seen that increasing the length above 128 locations provides no significant gain in terms of reaching 90°. Of greater significance is the lack of precision at the beginning of the table where only 6 (out of 2048) entries span the range 0° to 80°; clearly unacceptable.

LUT Input $\left(\frac{\lvert Q\rvert}{\lvert I\rvert}\right)$	**Computation**	**LUT Output** $\tan^{-1}\left(\frac{\lvert Q\rvert}{\lvert I\rvert}\right)$
0	$\mathrm{Tan}^{-1}(0)$	0.0000°
1	$\mathrm{Tan}^{-1}(1)$	45.000°
2	$\mathrm{Tan}^{-1}(2)$	63.435°
3	$\mathrm{Tan}^{-1}(3)$	71.356°
4	$\mathrm{Tan}^{-1}(4)$	75.964°
5	$\mathrm{Tan}^{-1}(5)$	78.690°
6	$\mathrm{Tan}^{-1}(6)$	80.538°
⋮	⋮	⋮
15	$\mathrm{Tan}^{-1}(15)$	86.186°
⋮	⋮	⋮
31	$\mathrm{Tan}^{-1}(31)$	88.152°
⋮	⋮	⋮
63	$\mathrm{Tan}^{-1}(63)$	89.091°
⋮	⋮	⋮
127	$\mathrm{Tan}^{-1}(127)$	89.549°
⋮	⋮	⋮
1023	$\mathrm{Tan}^{-1}(1023)$	89.944°
⋮	⋮	⋮
2047	$\mathrm{Tan}^{-1}(2047)$	89.972°

Table 2-1. A 'Tan^{-1} Look-Up Table' with unacceptable output precision in the range 0° - 80°.

For the same LUT length, the output precision at the beginning of the LUT may be increased without greatly affecting the overall output range. Table 2-2, shows that output precision may be increased to satisfactory levels if the input to the LUT is multiplied by 32, and the pre-computed output calculated after a compensatory division of 32 has been applied. Equation (2-9) can be updated to account for this change and a new expression for the instantaneous phase is given by

$$phase_i = \left\{ \begin{bmatrix} +000° + Tan^{-1}\left(\frac{1}{32}\left(32 \cdot \frac{|Q_i|}{|I_i|}\right)\right) & if\ quadrant=0 \\ +180° - Tan^{-1}\left(\frac{1}{32}\left(32 \cdot \frac{|Q_i|}{|I_i|}\right)\right) & if\ quadrant=1 \\ -180° + Tan^{-1}\left(\frac{1}{32}\left(32 \cdot \frac{|Q_i|}{|I_i|}\right)\right) & if\ quadrant=2 \\ +000° - Tan^{-1}\left(\frac{1}{32}\left(32 \cdot \frac{|Q_i|}{|I_i|}\right)\right) & if\ quadrant=3 \end{bmatrix} \right\} \tag{2-10}$$

where $\left(32 \cdot \frac{|Q_i|}{|I_i|}\right)$ is the integer input x to a LUT that implements a $\mathrm{Tan}^{-1}(x/32)$ operation.

In practice, the output from Table 2-2 must be scaled to best utilise the dynamic range of the DSP. It was found that a fixed scaling factor which transforms 90° to one quarter of the maximum positive integer is most effective since this also reduces potential for numerical overflow in subsequent operations.

LUT Input $\left(32 \cdot \frac{\lvert Q\rvert}{\lvert I\rvert}\right)$	**Computation**	**LUT Output** $\tan^{-1}\left(\frac{1}{32}\left(32 \cdot \frac{\lvert Q\rvert}{\lvert I\rvert}\right)\right)$
0	$\mathrm{Tan}^{-1}(0/32)$	0.0000°
1	$\mathrm{Tan}^{-1}(1/32)$	1.790°
2	$\mathrm{Tan}^{-1}(2/32)$	3.576°
3	$\mathrm{Tan}^{-1}(3/32)$	5.356°
4	$\mathrm{Tan}^{-1}(4/32)$	7.125°
5	$\mathrm{Tan}^{-1}(5/32)$	8.881°
6	$\mathrm{Tan}^{-1}(6/32)$	10.620°
⋮	⋮	⋮
15	$\mathrm{Tan}^{-1}(15/32)$	25.11°
⋮	⋮	⋮
31	$\mathrm{Tan}^{-1}(31/32)$	44.09°
⋮	⋮	⋮
63	$\mathrm{Tan}^{-1}(63/32)$	63.072°
⋮	⋮	⋮
127	$\mathrm{Tan}^{-1}(127/32)$	75.858°
⋮	⋮	⋮
1023	$\mathrm{Tan}^{-1}(1023/32)$	88.208°
⋮	⋮	⋮
2047	$\mathrm{Tan}^{-1}(2047/32)$	89.104°

Table 2-2. A practical 'Tan^{-1} Look-Up Table'.

It was stated earlier that frequency discrimination represents the greatest processing overhead within the FM demodulator. The main reason for this is the 16-bit integer division which is performed prior to addressing the Tan^{-1} LUT; this alone requires 16 single-cycle instructions. Whenever divisions are performed it is necessary to detect and prevent 'divide by zero' errors, in this case it is also necessary to limit the result of the division to the input range of the LUT. Finally, the extended arc Tangent function requires the input quadrant to be detected so that the corresponding phase correction can be applied; equation (2-10) . If the differential phase detector is also considered, a decision must also be made on each output sample to ensure that the smallest phase difference is always selected. In total, approximately 70 instructions are required for each sample to implement frequency discrimination; this equates to more than half of the processing capability with the chosen DSP and sampling frequency.

2.2.4 Block Processing and Sub-Sampling

The FM demodulator was implemented with a series of software algorithms. In general, the DSP instructions that make up each algorithm can be split into two categories; 'housekeeping instructions' and 'processing instructions'. In this case, 'housekeeping instructions' refer to those instructions which restore and save the conditions (context) required for the algorithm to execute correctly; no useful output is generated. 'Processing instructions' refer to all the remaining operations required to read one input sample, perform interim calculations and generate one output sample. When processing is performed on a sample-by-sample basis, the number of instruction cycles can be expressed in terms of 'housekeeping instruction cycles' *H-Cycles* and 'processing instruction cycles' *P-Cycles* as follows

$$Cycles\_Per\_Sample = H\_Cycles + P\_Cycles \qquad (2\text{-}11)$$

Software efficiency is poor when processing is performed on a sample-by-sample basis because the 'housekeeping instructions' must be executed for each input sample. Software efficiency was improved significantly by processing blocks of N samples within each algorithm. If an algorithm reads N input samples and writes N output samples, the average number of instruction cycles per input sample is given by

$$Cycles\_Per\_Sample = \frac{H\_Cycles + (N \cdot P\_Cycles)}{N} \qquad (2\text{-}12)$$

It can be seen from equation (2-12) that the average number of instruction cycles reduces as N increases because the 'housekeeping' overhead is executed less frequently. However, the memory requirement of each algorithm may increase by a factor of N as a result of this improvement, as may the processing delay. In this case it was found that $N = 256$ provided the best compromise between memory occupancy and improvement to software efficiency.

Block sample processing alone was found to be insufficient to allow real-time execution of the FM demodulator software and a second optimisation was also considered. Each individual algorithm was analysed and those which represented the greatest overhead were identified. For these algorithms it was found that sub-sampling offered the only practical solution with which real-time execution could be achieved. For diagnostic purposes it is desirable to maintain the same sample rate throughout the software, and the use of over-sampling (by repetition) at the output of these (sub-sampled) algorithms achieves this goal. If input sub-sampling and output over-sampling by a factor of S is applied, equation (2-12) may be re-expressed

$$Cycles\_Per\_Sample = \frac{H\_Cycles + \left(\frac{N}{S} \cdot (P\_Cycles + S)\right)}{N} \qquad (2\text{-}13)$$

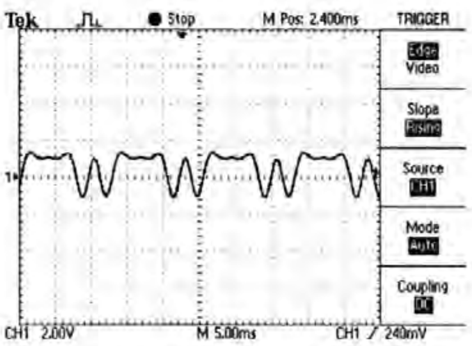
The 80MHz TMS320C50 DSP provides 40MIPS (40 Million Instructions Per Second). With sampling frequency $f_s = 307.2$ kHz, 130 instruction cycles are available on average to process each sample. In Table 2-3, the main algorithms in the demodulator software have been listed and general expressions for the average number of instruction cycles are given for each (assuming optimum conditions). If processing is conducted on a sample-by-sample basis ($N = 1$, $S = 1$) it can be seen that a total of 361 instruction cycles are required for each sample and that real-time execution is clearly not possible. If block processing ($N = 256$, $S = 1$) is used, the average is reduced to 112; this still does not guarantee real-time execution since the processor's instruction pipeline has not been considered. However, it can be seen that the phase detector and differential phase detector represent the greatest overhead. By employing a sub-sampling factor $S = 2$ in these functions, the overall average is reduced to just 67 instruction cycles; sufficient to guarantee real-time execution.

Function	Average No. of Cycles per Sample (General Case)	Average No. of Cycles per Sample (N=1, S=1)	Average No. of Cycles per Sample (N=256, S=1)	Average No. of Cycles per Sample (N=256, S=2)
Get Signal Samples	$\frac{8+[N \cdot 2]}{N}$	10	2.031	2.031
Frequency Correction	$\frac{70+[N \cdot 7]}{N}$	70	7.273	7.273
Low Pass Filter (I-Stream)	$\frac{25+[N \cdot 3]}{N}$	28	3.097	3.097
Low Pass Filter (Q-Stream)	$\frac{25+[N \cdot 3]}{N}$	28	3.097	3.097
Phase Detector	$\frac{37+\left[\frac{N}{SS} \cdot (62+S)\right]}{N}$	99	62.145	32.145
Differential Phase Detector	$\frac{30+\left[\frac{N}{SS} \cdot (20+2 \cdot S)\right]}{N}$	52	22.117	12.117
Root 40% RC Filter	$\frac{28+\left[\frac{N}{SS} \cdot (9+S)\right]}{N}$	37	10.109	3.359
Transfer output to DSP 2	$\frac{26+[N \cdot 2]}{N}$	28	2.102	5.609
Total	$\frac{249+[N \cdot 17]+\left[\frac{N}{S} \cdot (91+(4 \cdot S))\right]}{N}$	**361**	**111.971**	**67.473**

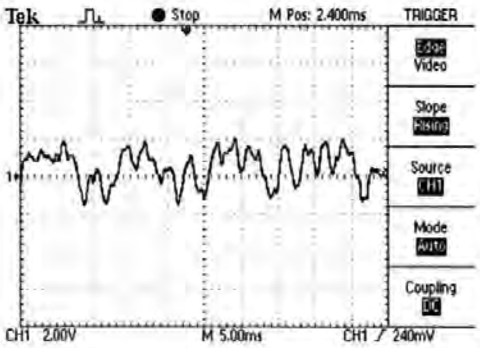
Table 2-3. Illustration of improvements to FM demodulator software efficiency with the use of block processing and sub-sampling.

2.3 *Simulation Performance*

For the frequency synchronisation algorithms to be evaluated in a meaningful way, it was first necessary to determine the limitations of the DSP software-based demodulator implemented by the author with respect to input carrier to noise ratio (CNR) and residual uncorrected Doppler error. Throughout this section, results were obtained using a simulated test signal applied digitally to the DSP demodulation equipment. The test signal was generated by PC software written by the author to allow controlled Doppler error and wide-band AWGN to be introduced. This software also compared the demodulated data to the data transmitted for automated bit error rate (BER) measurements. The transmitted data stream consisted of random data assembled into 512-bit frames and preceded by a short unique word to synchronise BER measurements. The CNR was first reduced from 99dB (effectively noise-free) until demodulated bit errors were too frequent to allow BER to be measured; CNR = 0dB was found to represent the lower operating threshold. Figure 2-17 shows the demodulator's root 40% raised cosine data output for noise-free and CNR = 0dB inputs. Figure 2-17b shows clearly that the data signal is heavily distorted by high-level AWGN and that the distinction between logic '1' and logic '0' is marginal at this level. However, it is clear that the Doppler frequency synchronisation algorithms must still operate reliably at CNR = 0dB.



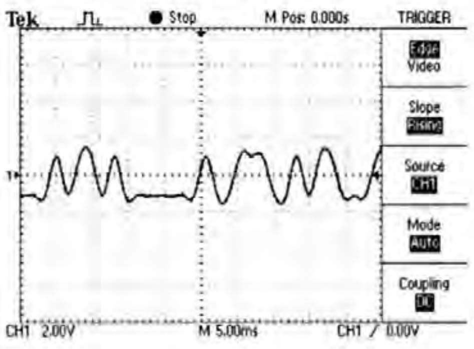
a. Noise-free Input - Error free.



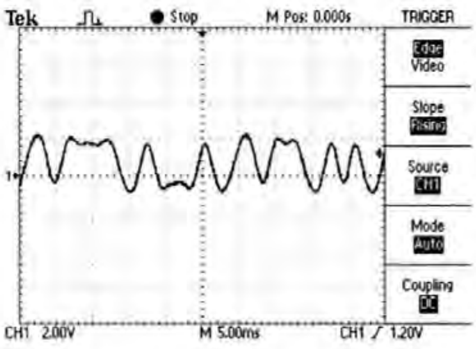
b. Input CNR = 0dB - High error rate.

Figure 2-17. Demodulator root 40% raised Cosine filtered output - data = 0xf5.

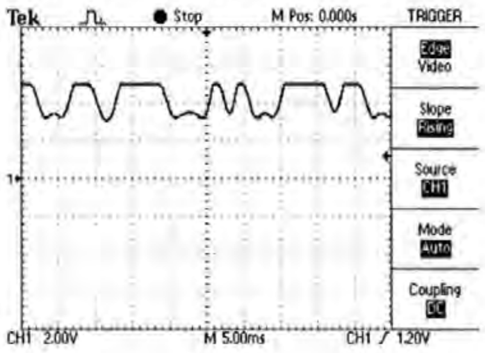
In terms of residual (uncorrected) Doppler error Δf_D, Figure 2-18 shows the demodulator's root 40% raised cosine data output for a noise-free input signal at four significant levels that were identified. With Δf_D = 0Hz (Figure 2-18a), the demodulator's output has zero DC offset and performance is error free. As Δf_D approaches 1.5kHz (Figure 2-18b) the zero crossings are no longer well defined and symbol timing recovery and bit decision algorithms begin to fail. However, since the demodulator output is not distorted, fine Doppler tracking will not be affected. At Δf_D = 10kHz (Figure 2-18c), the demodulator output begins to distort (is clipped) due to the DSP's 16-bit fixed-point numeric range. At this level Doppler tracking becomes less reliable because the DC offset can no longer be reliably computed. At Δf_D = 15kHz and above (Figure 2-18d), severe distortion occurs at the demodulator output due to phase ambiguity within phase detector and differential phase detector (frequency discrimination) algorithms. With this degree of output distortion Doppler tracking is impossible.



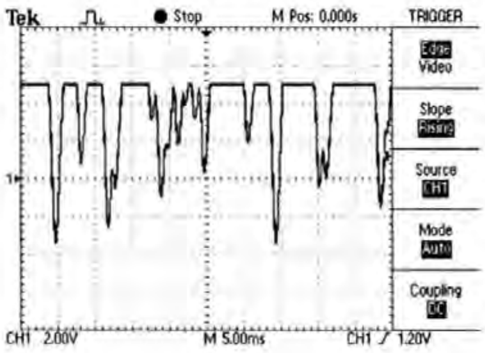
a. Residual Doppler error Δf_D = 0Hz.



b. Residual Doppler Error Δf_D = 1.5kHz.



c. Residual Doppler Error Δf_D = 10kHz.



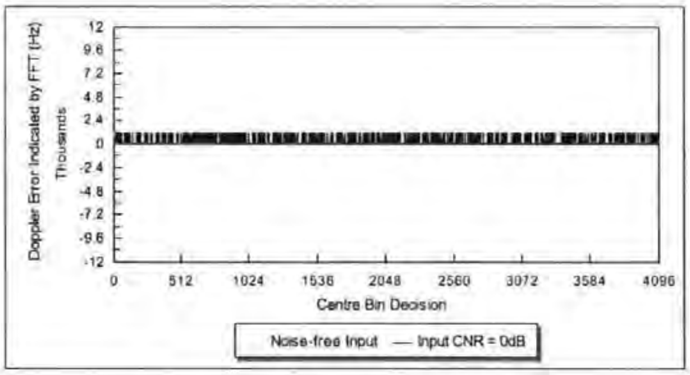
d. Residual Doppler Error Δf_D = 15kHz.

Figure 2-18. Demodulator root 40% raised Cosine filtered output - random data.

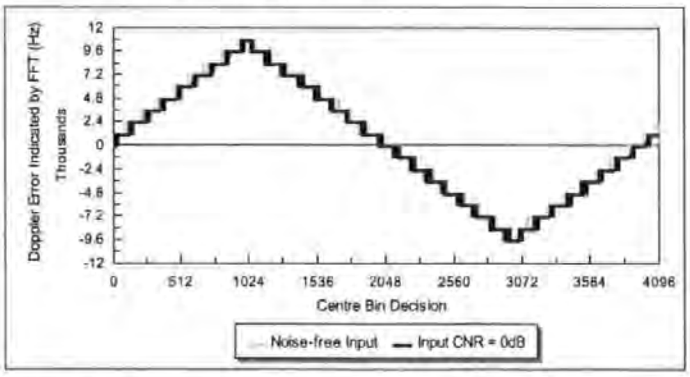
In summary, the demodulated data is error-free providing that residual Doppler error is kept below 1.5kHz. Residual Doppler errors below 10kHz may be corrected by Doppler tracking alone, while residual Doppler errors greater than 10kHz require assistance from the FFT coarse frequency synchronisation sub-system.

2.3.1 Coarse Doppler Tracking Performance

Figure 2-19a shows coarse FFT synchronisation performance with zero Doppler error and Figure 2-19b with a linearly changing Doppler error, and in each case results are plotted for noise-free and CNR = 0dB inputs. Without Doppler error, Figure 2-19a, the FFT indicates variations of 1200Hz (equivalent to 1 frequency bin) which may be attributed to the underlying modulation and to the fact that the nominal carrier frequency does not correspond exactly with a FFT frequency bin; with sampling frequency 307.2kHz, a 52kHz carrier corresponds with bin 43.33 of a 256-point FFT. Statistical evaluation of these results show that the standard deviation in the frequency bin indicated is 0.49992 Hz and 0.374667 Hz respectively for the noise-free and CNR = 0dB cases; these results are explained by effects of rounding errors in the fixed-point implementation of the demodulation and tracking algorithms. With a linearly changing Doppler error of 187.5 Hz per second, Figure 2-19b, the FFT operates successfully over the full 20kHz frequency range anticipated in practice.



a. Zero Doppler error introduced.

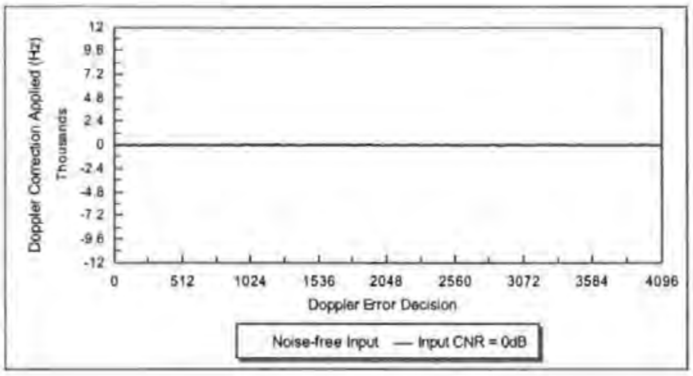


b. Linearly changing Doppler error of 187.5 Hz per second.

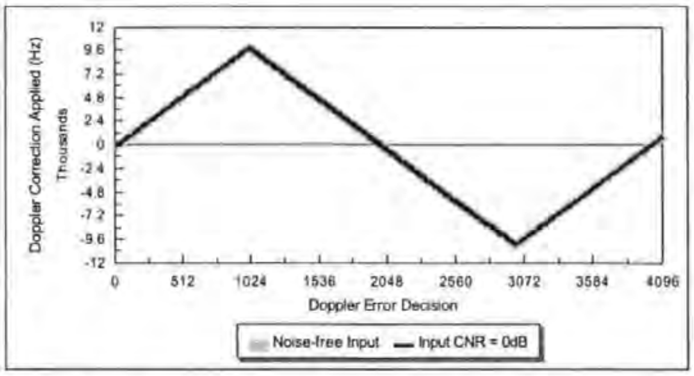
Figure 2-19. Coarse FFT synchronisation over 4096 decision cycles (3.64 minutes at 9,600bps).

2.3.2 Fine Doppler Tracking Performance

Figure 2-20a shows Doppler tracking performance with zero Doppler error and Figure 2-20b with a linearly changing Doppler error, and in each case results are plotted for noise-free and CNR = 0dB inputs. Without Doppler error, Figure 2-20a, Doppler tracking exhibits random variations over the range ±100Hz Hz which may be attributed to the underlying modulation. Statistical evaluation of these results show that the standard deviation for the Doppler correction applied in the noise-free and CNR = 0dB cases is 43.47 Hz and 44.14 Hz respectively. With a linearly changing Doppler error of 187.5 Hz per second, Figure 2-20b, tracking operates successfully over the full 20kHz frequency range anticipated in practice. These results show that tracking performance is unrelated to the input CNR.



a. Zero Doppler error introduced.

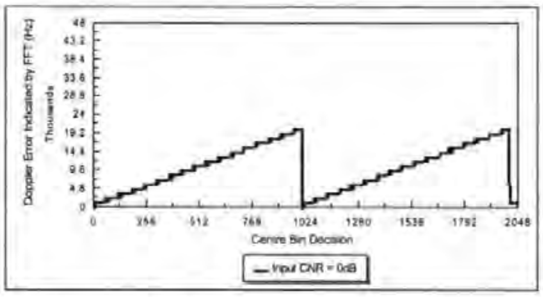


b. Linearly changing Doppler error of 187.5 Hz per second.

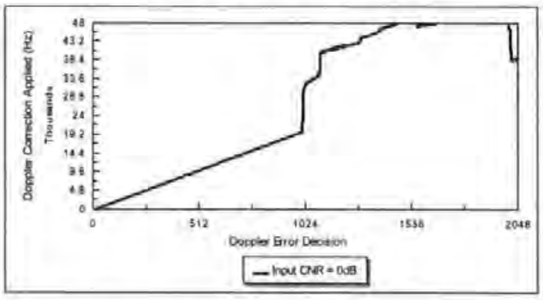
Figure 2-20. Fine tracking over 4096 decision cycles (3.64 minutes at 9,600bps).

2.3.3 Coarse + Fine Doppler Tracking Performance

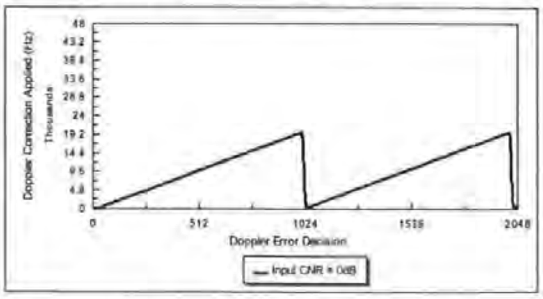
The coarse and fine Doppler tracking algorithms were finally evaluated with a large instantaneous change to the Doppler error; simulating lost frequency lock or initial acquisition. For these tests, a linearly changing Doppler error was introduced followed by a 20kHz step change; these tests were conducted with worst-case input CNR (0dB). The FFT, Figure 2-21a, tracks this error and remains locked throughout. With the FFT inactive, Figure 2-21b, fine Doppler tracking is successful with a linearly increasing Doppler error but fails after the 20kHz step change. With the FFT active, Figure 2-21c, lost lock is detected and fine Doppler tracking is rapidly re-initialised by the FFT.



a. Coarse FFT frequency synchronisation.



b. Fine Doppler tracking with coarse FFT synchronisation inactive.



c. Fine Doppler tracking with coarse FFT synchronisation active.

Figure 2-21. Coarse & fine Doppler tracking over 2048 decision cycles (1.82 minutes at 9,600bps) - linearly changing Doppler error of 375 Hz per second.

2.4 *Summary and Conclusions*

In this chapter the author presented frequency synchronisation algorithms to provide compensation for Doppler shift which results from the orbital velocity of LEO microsatellites, section 2.1. A Doppler tracking modem is ideally suited to DSP implementation, and algorithms suitable for implementation on low-cost, fixed-point DSPs were presented. Coarse synchronisation may be provided by a FTT algorithm with theoretical frequency resolution equivalent to half the frequency bin spacing. However, due to the underlying modulation, this technique generates variations to the carrier frequency indicated and is not sufficiently stable to provide Doppler tracking directly. Doppler error manifests as a DC offset at the FM demodulator output, and fine Doppler tracking may be achieved using feedback from the demodulator output. The Doppler tracking signal may be used to provide Doppler correction in the demodulator and pre-Doppler compensation in the modulator. The coarse frequency synchronisation sub-system is required to aid initial frequency lock and allows Doppler tracking to be re-initialised rapidly in the event of lost frequency lock. In order to evaluate the frequency synchronisation algorithms, it was necessary to fully implement both modulator and demodulator. To achieve real-time execution, several optimisation techniques were described by the author in section 2.2. These techniques included efficient DSP assembly code for frequency correction, frequency modulation and frequency discrimination, alternative implementations of a root 40% raised cosine filter and the use of block processing and sub-sampling. The techniques presented were described with particular reference to the TMS320C50 DSP, but are equally applicable to other fixed-point DSPs.

The algorithms presented in this chapter were evaluated in section 2.3. The results identified four significant ranges for residual (uncorrected) Doppler error for the demodulator implemented. These characteristics arise due to the limited numeric range of the TMS320C50 fixed-point DSP, and the distortion which manifests at the demodulator output when residual Doppler error is too great. In the most severe case, the FFT coarse frequency synchronisation must be relied upon to re-initialise Doppler tracking. These results also showed that the Doppler tracking and frequency synchronisation algorithms implemented are largely independent of the input carrier to noise ratio and perform satisfactorily over the 20kHz range of Doppler error anticipated in practice.

3. A Burst Demodulator for a Satellite Data Return Link

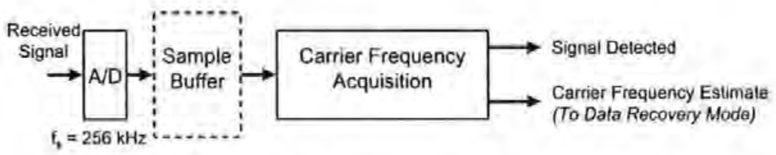
In this chapter the author presents algorithms and techniques identified during investigations into a DSP software-based burst demodulator for use in a satellite data return link. It is well known that a transmitted burst with *minimal* length synchronisation preamble facilitates efficient utilisation of the transmission channel and the main focus for these investigations were algorithms for rapid carrier frequency acquisition and symbol timing synchronisation. Another factor considered was computation efficiency so that low-cost fixed-point DSP hardware could be exploited and further investigations were conducted into modified (optimised) error correction algorithms for real-time DSP implementation. These algorithms were combined in a novel manner to form a burst demodulator with the potential to acquire carrier frequency and symbol timing synchronisation after just 32 symbols have been received.

A general description of a burst demodulator is given in section 3.1 where requirements for rapid carrier frequency acquisition and symbol timing synchronisation are highlighted. Carrier frequency acquisition is a fundamental requirement of a burst demodulator and techniques based upon a modified FFT algorithm (the Offset-FFT) are discussed in section 3.2. Benefits from these algorithms, over the standard FFT, are achieved with minimal additional processing overhead and include improved frequency resolution and signal detection performance at low SNR. Symbol clock recovery is identified as another area where rapid synchronisation is required and section 3.3 discusses the use of a 'delay and multiply' technique based upon an IIR filter resonator which exhibits both rapid acquisition performance and computation efficiency. Forward error correction is required to combat the effects of demodulated bit errors at low SNR and a standard Viterbi Decoder algorithm was found to represent significant processing overhead. In section 3.4, modifications to the standard Viterbi Decoder algorithm are discussed which reduce computation overhead to acceptable levels. A complete DSP software-based BPSK/QPSK burst demodulator was implemented by the author in order to evaluate the aforementioned algorithms. The burst demodulator is described in Appendix D where details of sub-sections not directly relevant to the main themes of this chapter are also given. In addition, Appendix D contains a comprehensive set of signal plots captured from each stage of the author's software. Overall performance of the burst demodulator in simulated and satellite trials is discussed in section 3.5 and conclusions from the chapter are summarised in section 3.6.

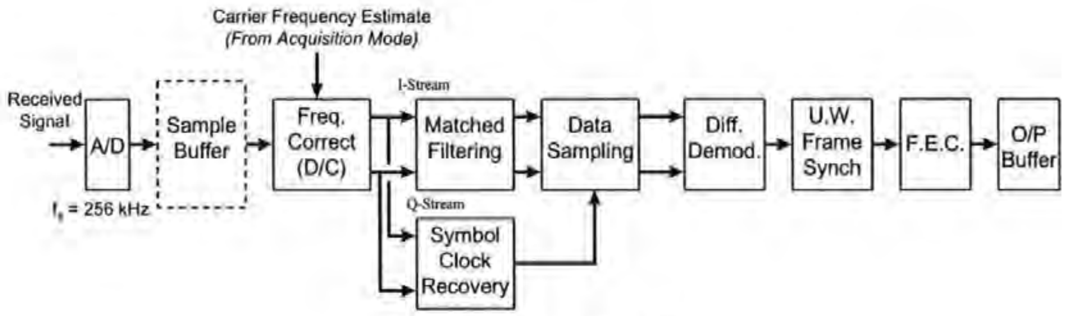
3.1 Burst Demodulator Overview

For any demodulator synchronisation is required in many forms; frequency, phase, symbol timing and message frame synchronisation. For continuous demodulation synchronisation can occur over a relatively long period since, once it has been achieved, variations can be tracked and corrections continually applied (see Chapter 2). In contrast, a burst demodulator must achieve synchronisation quickly because the duration of transmissions are much shorter; potentially shorter than the time a continuous demodulator requires just to synchronise. For this reason, a burst demodulator requires synchronisation algorithms which provide rapid acquisition. For a burst demodulator the term 'acquisition' is particularly relevant because the demodulator is also required to detect the presence of each transmission before synchronising to it. It is common therefore to refer to two modes of operation; 'Acquisition Mode', for detecting transmissions, and 'Data Reception Mode' once acquisition has been achieved. Figure 3-1 shows a block diagram of a general digital burst demodulator with acquisition and data reception modes clearly labelled. The received signal is applied to an analogue to digital converter and the resulting signal samples written to a large memory buffer, the remaining processing is performed by software algorithms which can be likened to analogue equivalents; by inserting a digital to analogue converter (D/A) at strategic points in the software similar analogue signals can be observed. This equivalence is assumed throughout the chapter.

During acquisition mode, Figure 3-1a, the only requirements are to detect the presence of a transmission and to determine the instantaneous carrier frequency; hence the two outputs shown. Data reception mode, Figure 3-1b, is initiated once the transmitted carrier frequency has been acquired. It is important to emphasise that the signal samples which triggered carrier frequency acquisition are maintained in the sample buffer since they may also facilitate symbol timing synchronisation. The carrier frequency estimate from acquisition mode is used to tune a local oscillator which frequency shifts the modulated signal to baseband frequencies (frequency correction). The local oscillator may be derived from the received signal in a coherent manner and the carrier frequency estimate used to aid a phase locked loop circuit. Alternatively, for non-coherent reception and more rapid synchronisation, the carrier frequency estimate may set the local oscillator directly. In this chapter only non-coherent reception is considered and it is assumed that the local oscillator is set directly by the carrier frequency estimate for the duration of the burst. Differences between the actual carrier frequency and the estimated carrier frequency (residual frequency error) manifest as a linearly increasing phase error in the demodulated data. When a relatively low symbol rate is employed (32ksps), residual frequency error becomes significant and must be kept to acceptable levels. Requirements for carrier frequency acquisition are therefore rapid acquisition and minimal residual frequency error.



a. Carrier frequency acquisition mode.



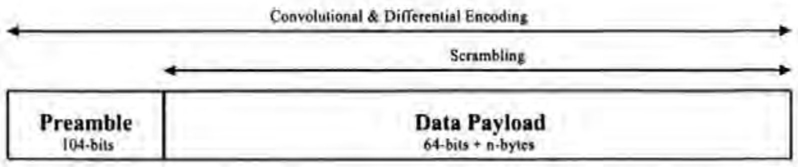
b. Data recovery mode.

Figure 3-1. General digital burst demodulator block diagram.

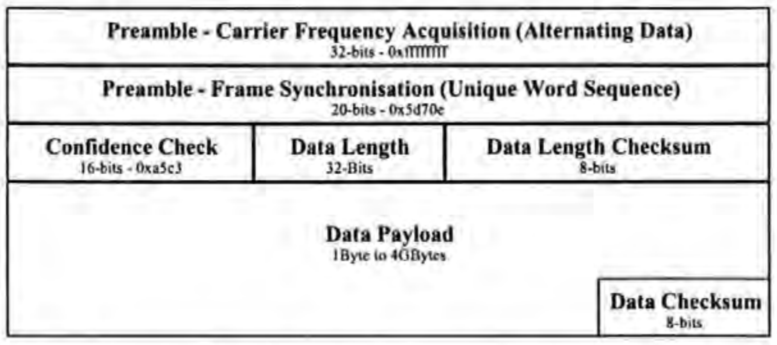
The baseband data signals, after down-conversion, are applied simultaneously to matched filters and a symbol clock recovery circuit; the matched filter produces the optimum signals for sub-sampling and the clock recovery circuit identifies the optimum sampling instants. Once data sampling has been conducted, differential detection is applied to eliminate the phase ambiguity associated with non-coherent demodulation. The symbol clock is derived from the received signal and a rapid timing reference is required for burst demodulation.

3.1.1 Frame Structure for Transmitted Burst

The transmitted data burst begins with a data sequence (preamble) designed to optimise both carrier frequency acquisition and symbol timing acquisition which must be discarded once both have bee achieved. To be clear, the transmitted data is organised into frames which adhere to a defined format, the beginning of each frame represents the preamble sequence while the remainder represents the data payload. If a known position within the frame can be located, the preamble may be discarded and the data payload recovered. It is common to append a unique word sequence to the preamble so that frame synchronisation may be declared after the last bit of the unique word sequence has been detected. Once frame synchronisation has been obtained, bit synchronisation for the forward error correction (FEC) is automatically obtained. As an example, the structure of transmitted bursts defined by the author for these investigations is depicted in Figure 3-2. These frames consist of a fixed-length preamble to aid carrier frequency acquisition, symbol timing synchronisation and unique word (frame) synchronisation in the burst demodulator and a variable-length data payload, Figure 3-2a. The data payload is scrambled to avoid long runs of consecutive bits and guarantee sufficient bit transitions for symbol clock recovery to be reliably conducted. This scrambling also provides a modest level of security because the correct shift register polynomial must be known by the receiver. ½-rate convolutional FEC is applied to the whole burst in order to combat the effects of demodulated bit errors and to simplify generation of the preamble in the transmitter hardware. Differential encoding is applied to the whole bust to combat the phase ambiguity associated with a non-coherent receiver.



a. Transmitted burst with scrambling , differential encoding and FEC.



b. Transmitted frame structure (prior to encoding).

Figure 3-2. Transmitted burst format (frame structure).

The individual fields of the transmitted burst are shown in Figure 3-2b prior to scrambling and encoding. The frame begins with a 32-bit 'Carrier Frequency Acquisition Preamble' field which, when encoded, produces an optimum signal for carrier frequency acquisition and symbol clock recovery. The carrier frequency acquisition algorithm requires a preamble of 32 symbols duration but, since the timing of transmissions is unknown, it is necessary to extend this preamble to 64 symbols (after ½-rate encoding) in order to guarantee that 32 symbols periods are experienced by the demodulator. The 'Frame Synchronisation Preamble' field, when encoded, produces a unique word sequence for frame synchronisation. In the data payload, the 'FEC Confidence Check' field is a 16-bit sequence whose purpose is to confirm that synchronisation, FEC initialisation and de-scrambling have been successfully initialised. Unless the first 16-bits decoded constitute this known sequence, false acquisition is assumed and the demodulator resets. The 'Data Length' field indicates in bytes the length of the data payload and the 'Data Length Checksum' field is an 8-bit XOR of the bytes which make up the 'Data Length' field. The remainder of the frame consists of the 'Data Payload' and a 'Data Checksum' field for detection of errors in data payload. It should be noted that in both cases the checksum and the fields they protect can be corrupted in a manner whereby errors are not detected. In practice, maximum transmitted frame lengths are known in advance and a second test confirms that an upper limit is not exceeded. In the case of corrupted data it is assumed that other more powerful error detection mechanisms are employed by protocols encapsulated within the payload.

3.2 *Carrier Frequency Acquisition*

Carrier frequency acquisition is the process of detecting the presence of a burst transmission and measuring the instantaneous carrier frequency so that frequency correction can be applied. For frequency correction in a coherent system a local carrier is ideally derived from the received signal and carrier frequency acquisition used to aid initialisation of a Phase Lock Loop (PLL) circuit. At low signal to noise ratios, standard techniques such as PLLs and Costas loops do not perform well [42], [43] and acquisition times are often measured in hundreds of symbols; clearly not suitable for rapid acquisition. Another approach is to regenerate a non-coherent local carrier at the demodulator based upon the instantaneous carrier frequency detected at acquisition. In this situation any difference between the estimated and actual carrier frequency is referred to as residual frequency error and manifests as a linearly increasing phase error in the recovered data. At low data rates residual frequency error is particularly significant and must be kept to acceptable levels. For these investigations transmissions are assumed to have nominal duration of 30ms to 1 second and the instantaneous carrier frequency at acquisition is used for the entire burst duration; there is not time for the carrier frequency to drift significantly during the burst. Further transmissions will generate new carrier frequency estimates and frequency drift will be tracked automatically. For a burst demodulator, acquisition time, probability of successful acquisition and probability of false acquisition are fundamental measures of performance. In addition, for non-coherent reception, residual frequency error is equally important. The acquisition algorithm must attempt to optimise these criteria and trade improvements in one at the expense of another where necessary.

3.2.1 FFT Carrier Frequency Acquisition

The frequency domain representation of a signal is commonly used for spectral estimation. By performing the Discrete Fourier Transform (DFT), frequency components of a signal can be found and exploited for carrier frequency acquisition. The Fast Fourier Transform (FFT) algorithm is often used as an efficient means of performing the DFT and Appendix C contains a derivation of the Decimation In Time (DIT) FFT algorithm for N=8.

$$F(k)=\sum_{n=0}^{N-1} x_n W_N^{nk} \qquad k=0,1,..,N-1 \quad (3\text{-}1)$$

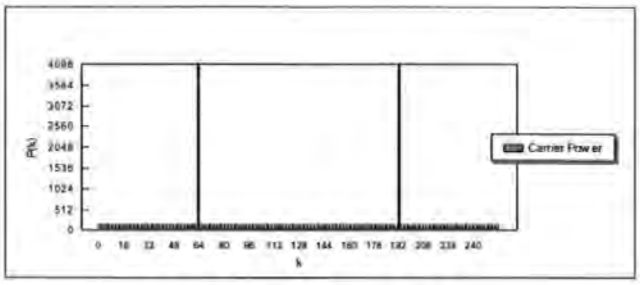
where $W_N = e^{-j2\pi/N}$

Consider a burst transmission scheme where each transmission begins with a short period of unmodulated carrier for frequency synchronisation purposes. A receiver performing a continuous FFT would be able to detect the preamble and produce an estimate of its carrier frequency based upon the frequency bin containing the greatest power. There are two factors which limit performance;

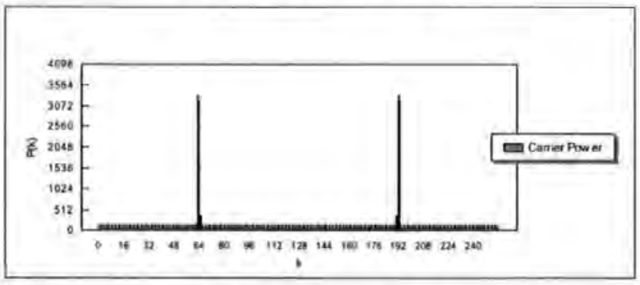
1. The ability to detect a signal above background noise is significant. At low signal to noise ratios (SNRs), a signal to noise power threshold must be set to minimise the probability of false acquisition (acquisition triggered by noise) while simultaneously maximising the probability of successful acquisition. These are conflicting requirements since increasing the threshold improves the former at the expense of the latter, and reducing the threshold has the opposite effect; a balance must be found. In the optimum case the carrier frequency is an integer division of the sampling frequency f_s Hz and hence the carrier power is contained within a single frequency bin at the FFT output. In the worst case the carrier frequency is such that the majority of the carrier power is shared equally between adjacent frequency bins at the FFT output; it is this situation which imposes a limit on the acquisition performance.
2. Carrier frequency estimation based upon the FFT frequency bin containing the greatest power has a resolution limited to half the frequency bin

spacing. With a sample rate f_s Hz, an N-point FFT has a maximum estimation error of $f_s/(2N)$ Hz. Increased frequency resolution is obtained by increasing N, but this is accompanied by an increase in the time taken to produce a result; since more input samples must be considered. Further limitations are imposed because an FFT requires $\log_{(2)}(N).(N/2)$ complex multiplications [44]. When the FFT size is doubled, the number of calculations increases by a factor greater than two and a point soon reached where a processor can no longer perform an FFT in the time available. Frequency resolution, and therefore residual frequency error, is determined by the FFT size N; which is ultimately governed by the processing power available.

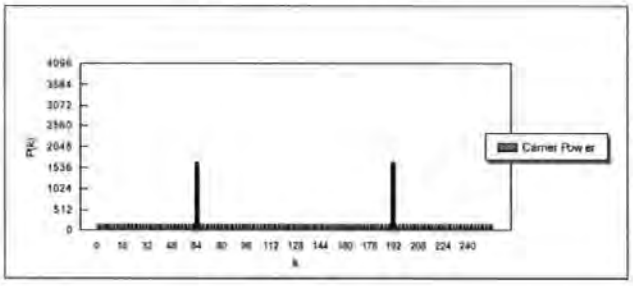
To illustrate these points Figure 3-3 shows 256-point FFT power (magnitude squared) spectra for carrier frequencies $f_c = 64 \times f_s/N$, $64.25 \times f_s/N$ and $64.5 \times f_s/N$ which represent optimum, intermediate and worse case situations.



a. $f_c = 64.00 \times f_s/N$ Hz.



b. $f_c = 64.25 \times f_s/N$ Hz.



c. $f_c = 64.50 \times f_s/N$ Hz.

Figure 3-3. 256-point FFT power spectra.

For a real input, the power spectrum is symmetrical about $f_s/2$, therefore the range k=N/2 to k=N-1 (the repeat spectrum) gives no useful information. Figure 3-3a demonstrates the case where power is contained within a single frequency bin and Figure 3-3c where power is shared between adjacent frequency bins, Figure 3-3b is an intermediate case. Clearly the receivers acquisition performance is governed by its ability to detect a carrier in Figure 3-3c where the peak power is half that of Figure 3-3a. The peak frequency bin k_{max}, carrier frequency estimate f_{c_est}, and the residual frequency error f_{err} resulting from the these examples are summarised in Table 3-1. The residual frequency error in the worst case, Figure 3-3c, has magnitude $0.50 \times f_s/N$ Hz, and the peak frequency bin k_{max} = 64 or k_{max} = 65 will be indicated with 0.5 probability (see Table 3-1).

f_c	k_{max}	f_{c_est}	$\lvert f_{err}\rvert$
$64.00\times f_s/N$ Hz	64	$64.00\times f_s/N$ Hz	$0.00\times f_s/N$ Hz
$64.25\times f_s/N$ Hz	64	$64.00\times f_s/N$ Hz	$0.25\times f_s/N$ Hz
$64.50\times f_s/N$ Hz	64 (0.5 probability)	$64.00\times f_s/N$ Hz	$0.50\times f_s/N$ Hz
$64.50\times f_s/N$ Hz	65 (0.5 probability)	$65.00\times f_s/N$ Hz	$0.50\times f_s/N$ Hz

Table 3-1. FFT carrier frequency acquisition performance.

3.2.2 Offset-FFT Carrier Frequency Acquisition

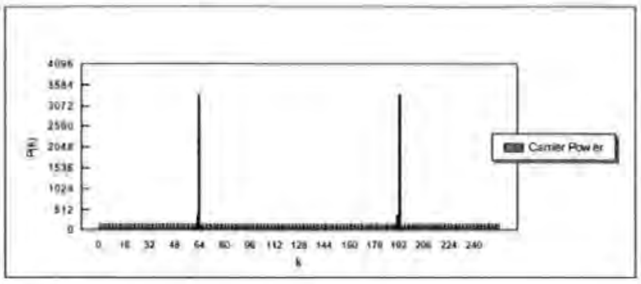
The Offset-FFT (OFFT) was described by Tomlinson in [45] and, when used for carrier frequency acquisition at low SNR, improves upon the two limitations imposed by the FFT for little or no processing overhead. The OFFT is a modification to the DIT FFT algorithm which adds a small frequency offset (c), typically $0.25\times f_s/N$ Hz, to each frequency bin. An expression for an Offset-DFT is given by (3-2) and the derivation for an 8-point OFFT given in Appendix C.

$$F(k+c)=\sum_{n=0}^{N-1} x_n W_N^{n(k+c)} \qquad k=0,1,..,N-1 \quad (3\text{-}2)$$

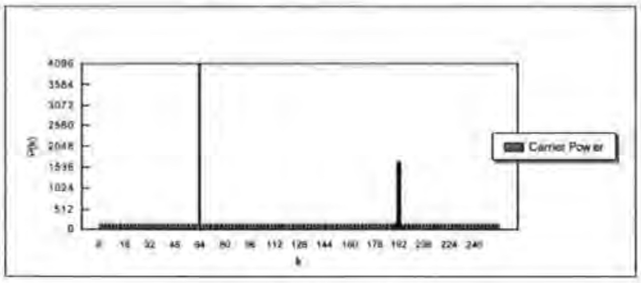
where $$W_N = e^{-j2\pi/N}$$

From the derivation of the FFT and the OFFT it can be seen that the algorithms are identical and only the coefficients $W_N^{n(k+c)}$ change; an FFT is in fact an OFFT with offset c=0. A general DSP algorithm can perform both the FFT and OFFT providing appropriate coefficients are supplied. Many optimisations to the FFT algorithm have been described [46], [47] which exploit trivial coefficients such as 1, -1, j and -j, it should be noted that OFFT coefficients are rarely trivial and that these optimisations cannot be applied.

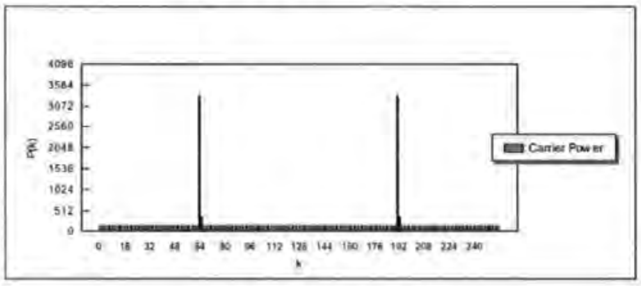
Figure 3-4 shows 256-point OFFT power spectra for carrier frequencies f_c = $64\times f_s/N$, $64.25\times f_s/N$ and $64.5\times f_s/N$ which represent worse case, optimum and worst case situations respectively. In order to fully appreciate the advantage the OFFT offers over the FFT, Figure 3-4 must be compared with Figure 3-3.



a. $f_c = 64.00 \times f_s/N$ Hz.



b. $f_c = 64.25 \times f_s/N$ Hz.



c. $f_c = 64.50 \times f_s/N$ Hz.

Figure 3-4. 256-point Offset-FFT power spectra (offset=0.25).

For a real input the OFFT power spectrum is no longer symmetrical about $f_s/2$, and the range k=N/2 to k=N-1 (the repeat spectrum) gives useful information. When compared to the FFT, for the same input, the OFFT power spectrum is offset by -0.25× f_s/N Hz in the range k = 0 to k = (N/2)-1 and offset by +0.25× f_s/N Hz in the range k = N/2 to k = N-1. Put another way, P(k) gives the power at frequency (k+0.25)×f_s/N Hz in first half of the power spectrum and the power at frequency (N-k-0.25) × f_s/N Hz in the repeat spectrum. The worst case for the OFFT, Figure 3-4a and Figure 3-4c, show that the maximum power is approximately 50% greater than for the FFT in Figure 3-3b. This fact allows the acquisition threshold to be increased, thus reducing the probability of false acquisition, while simultaneously improving the probability of successful acquisition. The peak frequency bin k_{max}, carrier frequency estimate f_{c_est}, and the residual frequency error f_{err} resulting from the OFFT examples are summarised in Table 3-2. The residual frequency error in the worst case, corresponding to Figure 3-4a and Figure 3-4c, has magnitude $0.25\times f_s/N$ Hz; half that of its FFT equivalent. In these cases the peak frequency bin $k_{max} = 64$ or $k_{max} = 192$ and $k_{max} = 64$ or $k_{max} = 191$ respectively will be indicated with 0.5 probability (see Table 3-2).

F_c	k_{max}	f_{c_est}	$\lvert f_{err}\rvert$
$64.00\times f_s/N$ Hz	64 (0.5 probability)	$64.25\times f_s/N$ Hz	$0.25\times f_s/N$ Hz
$64.00\times f_s/N$ Hz	192 (0.5 probability)	$63.75\times f_s/N$ Hz	$0.25\times f_s/N$ Hz
$64.25\times f_s/N$ Hz	64	$64.25\times f_s/N$ Hz	$0.00\times f_s/N$ Hz
$64.50\times f_s/N$ Hz	64 (0.5 probability)	$64.20\times f_s/N$ Hz	$0.25\times f_s/N$ Hz
$64.50\times f_s/N$ Hz	191 (0.5 probability)	$64.75\times f_s/N$ Hz	$0.25\times f_s/N$ Hz

Table 3-2. Offset-FFT carrier frequency acquisition performance.

3.2.3 Carrier Frequency Acquisition with Phase Reversing Preamble

It is common for a burst transmission system to use a synchronising preamble which begins with a period of unmodulated carrier specifically for the purpose of carrier frequency acquisition. Typically this part of the preamble is then followed by a period of carrier modulated by 180° phase reversing (alternating) data as an aid to symbol clock recovery. For these investigations, a phase reversing preamble is utilised for both carrier frequency acquisition and symbol clock recovery so as to reduce the overall preamble length. For carrier frequency acquisition, the received signal is sampled at 256kHz and a continuous 256-point OFFT (offset = 0.25) is performed. The power spectrum which results contains characteristics which also serve to reduce the probability of falsely acquiring on background noise; these characteristic are less likely to occur randomly. The power spectrum, Figure 3-5, shows two spectral components which are separated by 32 frequency bins (32k symbols per second) and centred about frequency bin 64 (64kHz carrier). Upon detection of peak spectral components with the desired frequency separation, a sample of the background noise is taken by averaging the contents of several adjacent frequency bins. Acquisition is declared when both components exceed a pre-determined SNR; 11dB for these investigations. A carrier frequency estimate is produced by finding the centre bin and providing an adjustment of $\pm 0.25\times f_s/N$ Hz depending on whether the peaks are detected in the lower or upper range of the OFFT power spectrum.

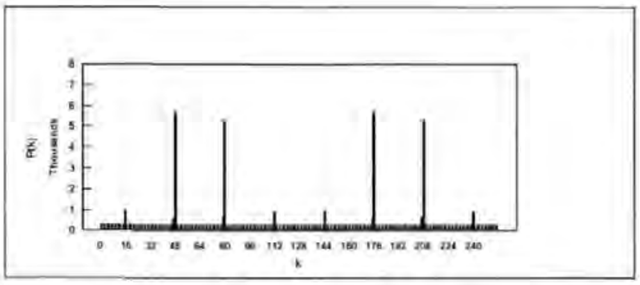


Figure 3-5. Offset-FFT power spectrum for 32ksps phase reversing preamble (f_s=256kHz, f_c=64kHz, N=256, offset=0.25) .

For a 256-point OFFT, with sample rate f_s=256kHz, the preamble should ideally be 1ms in duration. Since the timing of transmissions is assumed to be random it is necessary to extend the preamble to 2ms duration in order to guarantee that the optimum 1ms duration of phase reversing preamble is applied to the OFFT; a 2ms phase reversing preamble is assumed throughout this section.

3.2.4 Carrier Frequency Acquisition DSP Implementation

The carrier frequency acquisition algorithm was implemented in software on the dual-C50 DSP hardware described elsewhere in this Thesis. First a general OFFT/FFT algorithm was produced, the algorithm was then modified for fixed-point TMS320C50 implementation and finally adjusted to operate in real-time on the target hardware platform. These three stages are briefly described in this section.

3.2.4.1 A General Offset-FFT Algorithm

FFT algorithm implementation has been described in many Communications texts [48] [49] and is not described here for brevity. The author instead focuses on one particular area of the FFT algorithm required to achieve an efficient and general implementation; the coefficients or 'Twiddle Factors'. For an FFT these may be derived dynamically or pre-calculated depending upon whether execution time or memory utilisation is of most significance. For these investigations speed of execution was most important and pre-calculated coefficients were selected. Another factor to consider is the manner and order in which the pre-calculated coefficients are stored since this can help to make a general and flexible FFT algorithm.

The standard N-point FFT algorithm requires N/2 unique coefficients, for an 8-point FFT these are W^0, W^1, W^2 and W^3. Since each coefficient has both real and imaginary components N memory locations are required to store the coefficients, Table 3-3. In software the coefficients are addressed in a circular manner so that only W^0 is referenced during the first pass of the FFT, W^0 and $W^{N/4}$ for the second pass and eventually W^0 to $W^{(N/2)-1}$ on the final pass. This manner of addressing the coefficients does not represent significant processing overhead but occasionally a less memory efficient but more flexible convention is adopted; it is this alternate convention which is required for the OFFT.

Memory Locations	Contents
0, 1	W^0
2, 3	W^1
4, 5	W^2
6, 7	W^3

Table 3-3. Optimum FFT coefficient memory utilisation.

The N-Point OFFT requires N-1 unique coefficients, for an 8-point OFFT these are W^{0+4c}, W^{0+2c}, W^{2+2c}, W^{0+c}, W^{1+c}, W^{2+c} and W^{3+c}. In this case 2N-2 memory locations are required, Table 3-4 shows how 8-point OFFT and FFT coefficients can be stored. A general algorithm, for both FFT and OFFT, is guaranteed providing this convention for coefficient storage is adopted. From Table 3-4 it is clear that only the coefficients change and that the FFT is in fact an OFFT with offset c=0.

Memory Locations	Contents (OFFT)	Contents (FFT)
0, 1	W^{0+4c}	W^0
2, 3	W^{0+2c}	W^0
4, 5	W^{2+2c}	W^2
6, 7	W^{0+c}	W^0
8, 9	W^{1+c}	W^1
10, 11	W^{2+c}	W^2
12, 13	W^{3+c}	W^3

Table 3-4. General FFT coefficient memory utilisation.

3.2.4.2 Fixed-Point DSP Considerations

Coefficients for an FFT algorithm have magnitude between 0 and 1. The TMS320C50 DSP is a fixed-point device which deals with 16-bit two's complement integers and cannot manipulate fractional numbers directly; an alternate notation for the coefficients is required. The chosen notation is referred to as 'Q15 Format' and requires each coefficient to be multiplied by 2^{15} in order to maintain sign information and minimise quantisation errors. It is worth noting that the value 1 multiplied by 2^{15} becomes 32768 (8000_{16}), which is the 16-bit two's complement representation for the maximum negative number; not what is desired. Either special measures must be taken to detect this situation and subtract 1 from the result, thus giving the maximum positive integer, alternatively the coefficients should be multiplied by 2^{15}-1 (32767) in preference to 2^{15} (32768).

The TMS320C50 DSP is a 16-bit device with a 32-bit accumulator, that means two 16-bit integers may be multiplied and a 32-bit result generated. This is particularly significant when the coefficients are stored in Q15 format since two 16-bit integers must be multiplied during the 'Butterfly' calculation; a fundamental element of the FFT algorithm. A similar issue is that of growth within the FFT algorithm and the potential for numeric overflow within a fixed-point processor. It is common to assume growth by a factor of two for each butterfly calculation [50], hence each butterfly output is scaled by a factor of 0.5 so that overflow will not occur. Figure 3-6 shows the author's TMS320C50 DSP routine for an FFT butterfly calculation. A detailed explanation of the assembly code in Figure 3-6 is beyond the scope of this text. The reader's attention instead is brought to two highlighted lines which represent good examples of instructions which allow scaling to be applied without additional processing overhead on the TMS320C50 DSP. The first causes the 32-bit accumulator to be loaded with a 16-bit integer which is simultaneously scaled by 2^{14} (a 14-bit shift to the left), the second causes the highest 16-bits of the 32-bit accumulator to be saved after first scaling by 2^1 (a 1-bit left shift). The reader should note that the butterfly calculation is performed 'in-place', the output overwrites the input, and that only two temporary registers are used. The sequence of instructions is therefore determined by the need to overwrite each location only after it is no longer required.

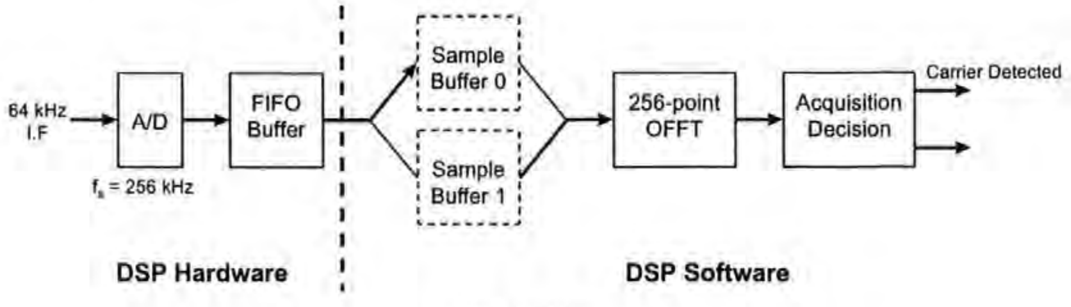
```
BUTTERFLY .macro IP
************************************************************************
* Author:     James T Slader                                           *
* Purpose:    Radix-2 DIT Butterfly Macro for OFFT, scales by 0.5      *
* Modified:   19/01/96 - Optimised                                     *
*                                                                      *
* PM = 00 - No shift  (Product shift Mode), SXM set                    *
* AR2 controls the number of B'flies per group (In BRCR format - count-1)  *
* AR3 points to Twiddles - At exit, pointer incremented by 2  (handed on)  *
* AR5 points to P-Data   - At exit, pointer incremented by 2  (handed on)  *
* AR7 points to Q-Data   - At exit, pointer incremented by 2  (handed on)  *
************************************************************************
      lt      *+,ar3          ;QR ¯ TREG
      mpy     *+,ar7          ;QR*WR/2 ¯ PREGh
      ltp     *-,ar3          ;QI ¯ TREG, QR*WR/2 ¯ ACCh
      mpy     *-              ;QI*WI/2 ¯ PREGh
      mpya    *+,ar7          ;QI*WR/2 ¯ PREGh, [QR*WR + QI*WI]/2 ¯ ACCh
      sach    TEMP1           ;[QR*WR + QI*WI]/2 ¯ TEMP1
      ltp     *,ar3           ;QR ¯ TREG, QI*WR/2 ¯ PREGh
      mpy     *+,ar5          ;QR*WI/2 ¯ PREGh, Hand on Twiddle-Pointer
      spac                    ;[QI*WR - QR*WI]/2 ¯ ACCh
      sach    TEMP2           ;[QI*WR - QR*WI]/2 ¯ TEMP2
      lacc    *,14,ar5        ;PR/4 ¯ ACCh
      add     TEMP1,15        ;[PR + (QR*WR + QI*WI)]/4 ¯ ACCh
      sach    *+,1,ar7        ;[PR + (QR*WR + QI*WI)]/2 ¯ PR location
      sub     TEMP1,16        ;[PR - (QR*WR + QI*WI)]/4 ¯ ACCh
      sach    *+,1,ar5        ;[PR - (QR*WR + QI*WI)]/2 ¯ QR location
      lacc    *,14,ar5        ;PI/4 ¯ ACCh
      add     TEMP2,15        ;[PI + (QI*WR - QR*WI)]/4 ¯ ACCh
      sach    *+,1,ar7        ;[PI + (QI*WR - QR*WI)]/2 ¯ PI location
      sub     TEMP2,16        ;[PI - (QI*WR - QR*WI)]/4 ¯ ACCh
      sach    *+,1,ar7        ;[PI - (QI*WR - QR*WI)]/2 ¯ QI location
          .endm
```

Figure 3-6. In-place FFT 'Butterfly' routine for the TMS320C50 DSP.

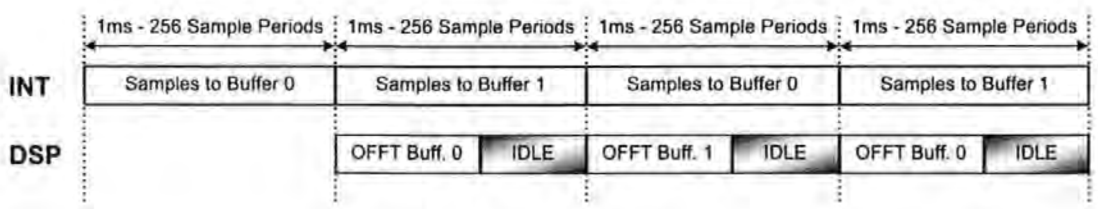
3.2.4.3 Real-time DSP Implementation

For these investigations the ADC sampled the incoming signal at 256kHz and the nominal carrier frequency selected to be one quarter the sampling frequency (64kHz) due to advantages offered by this relationship [51]. The transmitted symbol rate must be an integer division of the sample rate and 32ksps (8 samples per symbol) was selected. With 256kHz sampling it takes exactly 0.9909375ms to collect the 256 samples required at the input of a 256-point OFFT algorithm, leaving 3.90625μs for the OFFT execution. The reader should note that this strategy assumes an extremely fast DSP and that the suggestion is made only to aid further discussions. Considering the FFT butterfly calculation given in Figure 3-6, which consists of 22 DSP instructions, under optimum conditions each of these instructions requires a single processor cycle to execute. For the 40MHz TMS320C50 DSP each processor cycle has 50ns duration and this reduces to 25ns for the 80MHz variant. Given that a 256-point FFT algorithm consists of $\log_{(2)}(256).(256/2)$ butterflies, it is clear that the complete OFFT requires more than 22,520 processor cycles. In addition to the butterfly calculations there is further overhead associated with fetching the input samples from an analogue to digital converter (ADC) and generating the program loops within which the butterfly calculations occur; this overhead is ignored in order to simplify the discussion. The 22,520 processor cycles are equivalent to 1.126ms at 40MHz and 0.563ms at 80MHz; much longer than the 3.90625μs quoted previously. A strategy is therefore required which extends the time the DSP has available to perform the OFFT if real-time execution is to be achieved.

Considering first an 80MHz DSP, it is necessary to extend the time available for OFFT computation from 3.90625μs to more than 0.563ms in order to achieve real-time execution. A block diagram of the DSP hardware and the authors DSP software, configured for real-time carrier frequency acquisition, is shown in Figure 3-7a. The input signal is sampled by the ADC and the resulting samples stored in a 'First In First Out' (FIFO) buffer. Once the FIFO buffer contains 256 samples, a DSP interrupt is asserted which causes the samples to be transferred to a buffer within the DSP memory. The reader should note that the two buffers are toggling and while writing to one buffer the OFFT algorithm reads from the opposite buffer; it is this which extends the time available to perform the OFFT. It should also be noted that the sample buffers are not destroyed by the in-place OFFT and remain available for further use once acquisition is declared. In terms of a timing diagram, Figure 3-7b, samples are written to buffer 0 while the OFFT is simultaneously performed on the contents of buffer 1. Periods where the processor is idle are highlighted, and it is at the start of this interval that carrier frequency acquisition will be declared. The minimum time required for acquisition consists of 1ms to collect the signal samples plus the time taken to perform the OFFT acquisition algorithm (<1ms).



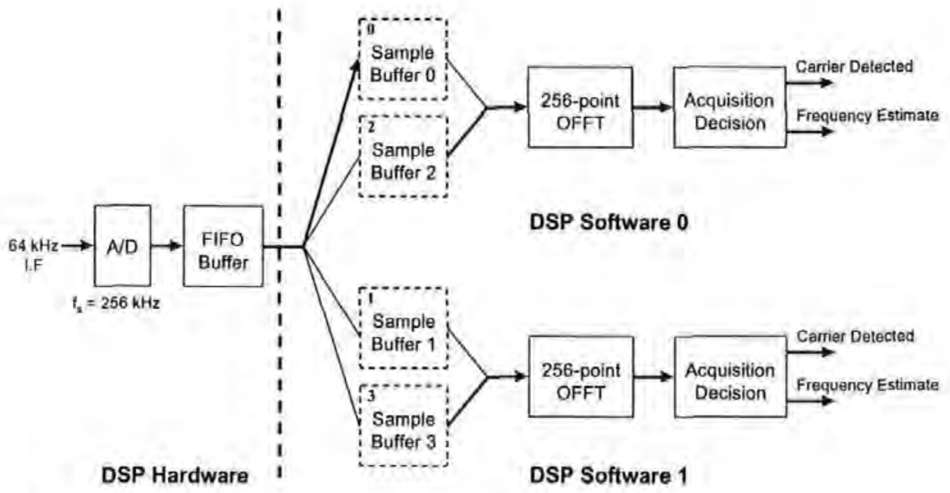
a. Block diagram.



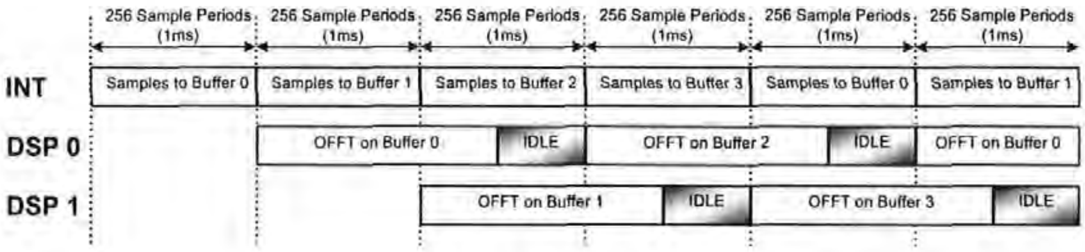
b. Timing diagram.

Figure 3-7. Real-time OFFT carrier frequency acquisition (single 80MHz DSP).

For the 40MHz DSP it is necessary to extend the time available for OFFT computation from 3.90625μs to more than 1.126ms in order to achieve real-time execution. It should be clear that a single 40MHz DSP is incapable of providing the desired performance since it is unable to perform an OFFT within 1ms. However, the target hardware contains two DSPs and collectively these can provide the desired performance. A block diagram of the dual-DSP hardware and the authors DSP software, configured for real-time carrier frequency acquisition, is shown in Figure 3-8a. As before, the input signal is sampled by the ADC and the resulting samples stored in a FIFO buffer. Once the FIFO buffer contains 256 samples an interrupt is asserted which causes the samples to be transferred to buffers within memory which is shared by both DSPs, these buffers are written to in the order indicated by bold type. Each DSP operates as described previously described but, since they are offset in time relative to each other, up to 2ms is now available to perform each OFFT. Operation is described more clearly by the timing diagram in Figure 3-8b, at any time up to two OFFTs can be in progress and that either DSP may declare acquisition. Additional logic (not shown) ensures that both DSPs are synchronised and that a coherent changeover from acquisition to data recovery mode occurs. Since the sample buffers are located in memory shared by the both DSPs, 1024 consecutive samples are available to each DSP for further processing upon acquisition. The minimum time required for acquisition consists of 1ms to collect the signal samples plus the time taken to perform the OFFT acquisition algorithm (>1ms).



a. Block diagram.



b. Timing diagram.

Figure 3-8. Real-time OFFT carrier frequency acquisition (dual 40MHz DSP).

It should be clear that the dual-DSP configuration could also be employed with the 80MHz DSPs with a 0.5ms time offset to further reduce the acquisition time. This allows a shorter preamble and therefore more efficient utilisation of the communication channel.

3.2.5 Carrier Frequency Acquisition Performance

Throughout these investigations the ability to test each algorithm was treated with high priority and the carrier frequency acquisition algorithms implemented in a manner which allowed many intermediate signal samples and results to be monitored.

Figure 3-9 shows the full carrier frequency acquisition algorithm in a form most closely representing the author's DSP software implementation. It is brought to the readers attention that the OFFT, 'power spectrum evaluation', and 'peak power location' algorithms are separated and that each has memory in which to buffer its output. The author's strategy allows the signal samples at the output of any algorithm to be applied to a D/A and viewed in real-time as a continuous signal on an oscilloscope. Two methods of supplying the input to the system were also provided; analogue or digital. Under normal operation an analogue signal is applied via the A/D, but for testing a digital interface was employed so that simulated signal samples could be written directly to the input sample buffers. PC software was developed to generate these simulated signal samples and to provide control over signal power, noise power and the carrier frequency of the test signal. Communication between the PC and DSP is made via an interface card developed (by others) for this purpose.

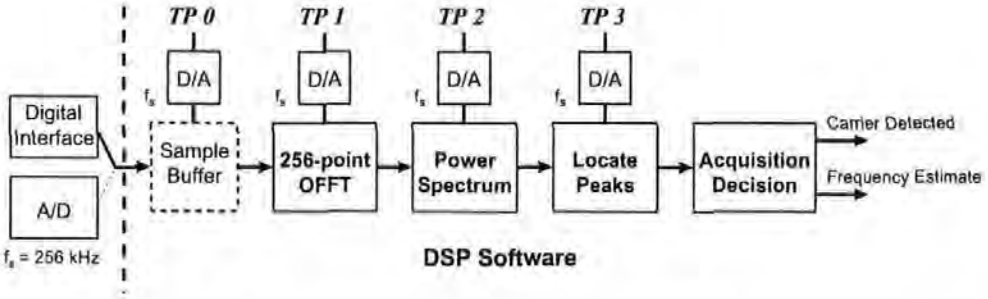
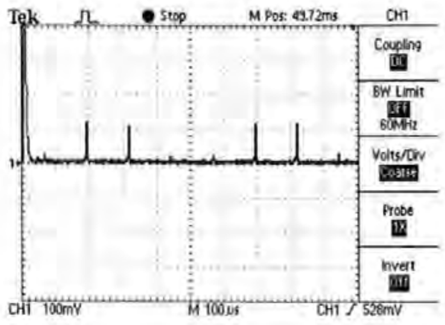
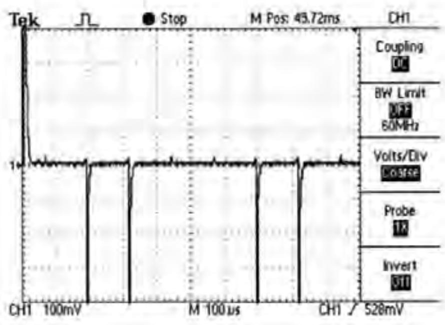


Figure 3-9. Burst demodulator implementation - carrier frequency acquisition.

Figure 3-9 shows four test points (TP 0 to TP 3) where samples can be applied to a D/A and viewed on an oscilloscope; analogous to inserting an oscilloscope probe directly into an analogue circuit. Of particular interest are those signals generated at test points TP 2 and TP 3, the FFT power spectrum and the peak power locations, examples of which are shown in Figure 3-10a and Figure 3-10b respectively. At the left hand side of each trace a high positive spike can be observed, this corresponds to frequency bin zero and is set such that the oscilloscope may be triggered correctly. The first trace represents the OFFT Power spectrum for the phase reversing preamble over the range 0 to f_s Hz (frequency bins 0 to N-1). For efficiency, the second trace is a simple modification to the first and shows the location of spectral peaks by changing the bin contents to a large negative value.



a. TP 2 - OFFT power spectrum.



b. TP 3 - Location of spectral peaks.

Figure 3-10. Carrier frequency acquisition test outputs (optimum conditions).

3.2.5.1 OFFT & FFT Acquisition Performance

Tests were performed which enabled carrier frequency acquisition performance with both the FFT and OFFT algorithms to be compared. The carrier frequency was set to 64.375 kHz in order to lie an equal distance from a 'worst case' frequency for both FFT and OFFT, but also at a frequency where the OFFT should offer improvement over the FFT. Simulated additive white gaussian noise (AWGN) was added to a signal corresponding to the maximum input level. The percentage of successful acquisitions (from 5000 test cycles) was recorded over the CNR range where the algorithms begin to fail. In this particular case, it was confirmed that the OFFT provides a 3dB improvement over the FFT at the failure point for no additional processing overhead, Figure 3-11.

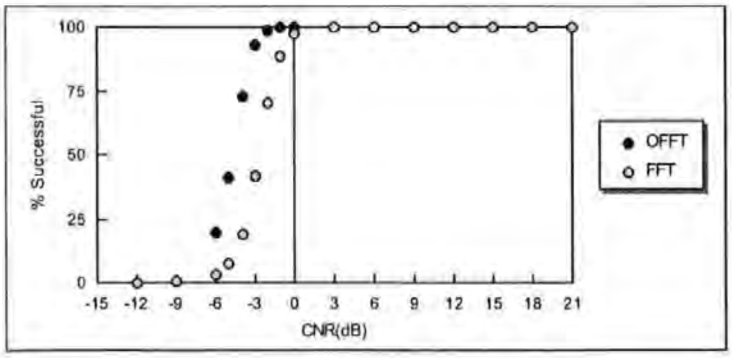


Figure 3-11. FFT verses OFFT acquisition performance (f_c=64.375kHz, maximum signal level).

3.2.5.2 Sensitivity to Carrier Frequency Offset

A second series of tests was conducted to measure the algorithms sensitivity to small carrier frequency offset. As shown in sections 3.2.1 and 3.2.2, there are optimum and 'worst case' frequencies for both FFT and OFFT carrier frequency acquisition. For a fixed CNR of -3dB this was confirmed by sweeping the carrier frequency over a 2kHz range in 0.125kHz steps and recording the results from 5000 test cycles at each frequency, Figure 3-12. For the FFT, optimum frequencies are clearly integer multiples of 1kHz and for the OFFT they are offset by ±0.25kHz. For both FFT and OFFT, the 'worst case' frequencies are located between their respective optimums. In the case of the FFT, extremely poor performance is exhibited at the 'worst case' frequencies. This is because the peak powers are distributed equally between adjacent frequency bins, hence there are four combinations of peaks which can be detected with equal probability; only one of these combinations results in successful acquisition. Figure 3-12 clearly demonstrates that the OFFT offers superior and more consistent performance than the FFT when used in conjunction with the phase reversing carrier preamble.

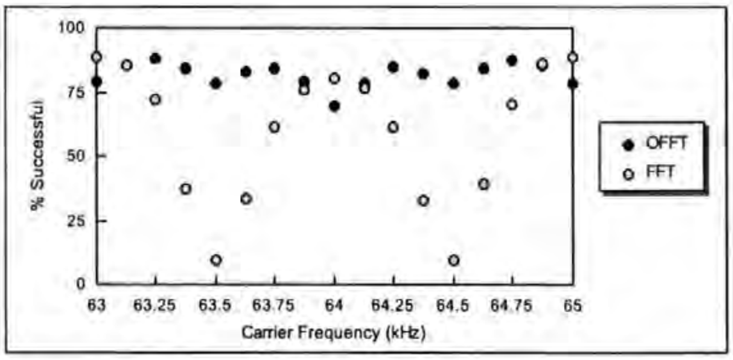


Figure 3-12. FFT verses OFFT carrier frequency acquisition performance (maximum signal level, CNR = -3dB).

3.2.5.3 OFFT Carrier Frequency Acquisition Range

The OFFT based algorithm alone was finally evaluated with 5000 test cycles at each frequency over a 32kHz range in steps of 0.25kHz to determine its useful carrier frequency acquisition range, Figure 3-13. For reasons discussed later, the permissible carrier frequency acquisition range has been artificially limited by the author to lie between 49kHz and 79kHz, and it is this which imposes both upper and lower frequency limits. It can be seen that the algorithm nominally operates over a 30kHz range and that there are two distinct trends which correspond to upper and lower performance characteristics.

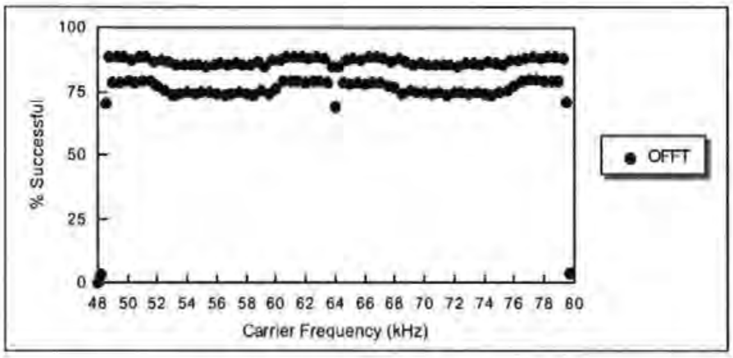


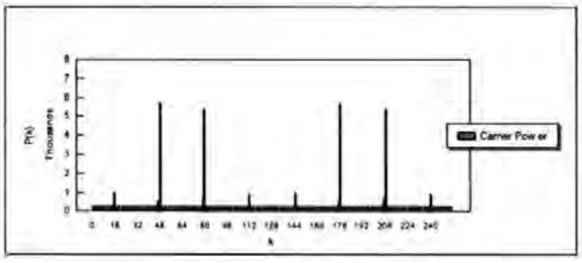
Figure 3-13. OFFT carrier frequency acquisition performance (maximum signal level, CNR = -3dB).

Of more significance is a noticeable reduction in performance as the carrier frequency approaches 64kHz. This is caused by a combination of odd harmonics and aliasing which result when a signal with no transmit filtering is sampled. Table 3-5 shows, for a 64kHz carrier, bin locations of the odd harmonics resulting from each peak in the phase reversing preamble. It can be seen that 7th, 9th and 15th harmonics occur at the same frequency bins as the peak powers (bold type). At other carrier frequencies different combinations of harmonics become dominant and the characteristics of Figure 3-13 could be completely explained. At both f_c=48kHz and f_c=80kHz the situation is extreme, other sections of the burst demodulator could be adversely affected, and it is for this reason that the author chose to artificially limit the carrier frequency acquisition range.

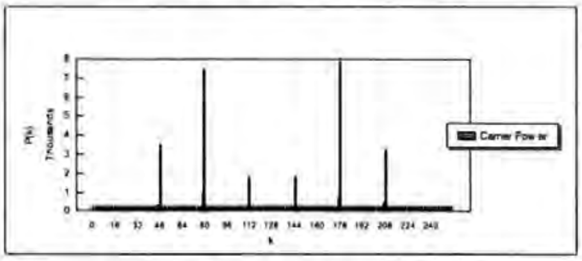
Harmonic	Magnitude	Frequency	Frequency	Frequency	Frequency
		Bin 0	Bin 1	Bin 2	Bin 3
1st	1.000	48	80	176	208
3rd	0.333	16	112	144	240
5th	0.200	240	144	112	16
7th	**0.143**	**208**	**176**	**80**	**48**
9th	**0.111**	**176**	**208**	**48**	**80**
11th	0.091	144	240	16	112
13th	0.077	112	16	240	144
15th	**0.067**	**80**	**48**	**208**	**176**

Table 3-5. Odd harmonic bin locations for phase reversing preamble (f_c=64kHz).

The effect of the harmonics in Table 3-5 is dependent upon the carrier phase. For completeness Figure 3-14 shows simulated OFFT preamble power spectra for initial carrier phases of 0° and 45°, which represent the two performance extremes. Considering only the lower half of the power spectrum, for an initial carrier phase of 0° the spectral peaks have approximately equal magnitude. For an initial carrier phase of 45° the peaks are no longer equal and the lower peak has reduced probability of exceeding the acquisition threshold.



a. Initial carrier phase = 0°.



b. Initial carrier phase = 45°.

Figure 3-14. OFFT power spectrum for 32ksps phase reversing preamble (f_s=256kHz, N=256, c=0.25, f_c=64kHz) .

3.3 Symbol Clock Recovery

For these investigations the transmitted symbol clock (32kHz) is derived in the digital transmitter hardware by integer division of a high frequency (8.064 MHz) crystal oscillator. In a similar manner the DSP burst demodulator derives its sampling frequency clock (256kHz) from a 77.824MHz master crystal; 256kHz was selected because it is an exact multiple of the symbol rate (256kHz = 8×32kHz). Initially it would appear that both transmit and receive hardware are matched in terms of symbol timing, that is symbols transmitted at a rate of 32kHz correspond exactly to 8 samples per symbol at a receiver with sampling frequency f_s=256kHz. When factors such as crystal oscillator tolerances, Doppler error over the satellite channel and operating temperature are considered it is extremely unlikely that the sending and receiving stations will be synchronous. In most cases either the transmitter runs marginally faster than the receiver or the receiver runs marginally faster than the transmitter. The ideal solution to this problem is for the receiver to adjust its local timing so that it becomes synchronised to the transmitter's symbol timing. In order to achieve this the receiver requires a sample clock that can be adjusted with extremely fine resolution. If this is not practical, as is assumed during these investigations, the receiver must continuously adjust its local symbol timing in order to best match that of the transmitter. For example, if the transmitter is faster then, from the receiver's viewpoint, most symbols will have duration of 8 samples but occasionally of 7. Similarly, when the transmitter runs slower than the receiver, most symbols consist of 8 samples but occasionally of 9. The receiver must compensate by occasionally advancing or retarding its symbol timing as required. In this case the recovered symbol clock will have maximum timing error of $\pm t_s/2$ (half the sample period) even under ideal conditions.

Symbol clock recovery algorithms are often based upon a classic 'delay and multiply' technique whereby the data signal is delayed (by half the symbol period) and multiplied with itself in order to produce a signal with a strong frequency component at the symbol rate; a further circuit is then used to derive a stable symbol clock. In [42] [43] a symbol clock recovery algorithm, based upon the 'delay and multiply' technique, is described which provides the performance characteristics required for a burst demodulator; namely rapid acquisition. The algorithm is implemented as a two-stage infinite impulse response IIR filter, Figure 3-15, which translates to an efficient DSP software implementation. Another feature of this algorithm is good flywheel performance in the presence of a signal fade or non-optimal stimulus (unchanging data).

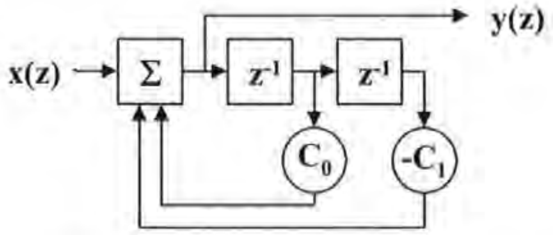


Figure 3-15. Clock recovery IIR filter structure.

In this section the author provides an analysis of the clock recovery algorithm. IIR filters are notoriously unstable, a fact which is exploited by the clock recovery algorithm, and this proves a significant problem for fixed-point DSP implementation due to the potential for numerical overflow. The majority of this section is concerned with the author's DSP implementation of the clock recovery algorithm and its integration into a burst demodulator. The algorithm's jitter performance and tolerance to carrier frequency offset at the input to the burst demodulator were measured. These results are presented and analysed at the section ending.

3.3.1 Symbol Clock Recovery IIR Filter

The clock recovery filter, Figure 3-15, may be implemented for a range of Q-factors by modifying the filter coefficients C_0 and C_1, where coefficient C_1 is critical and must be close to unity. For signed 16-bit fixed-point DSP implementation suitable values are given by

$$C_1 = \frac{2^n - 1}{2^n} \qquad n = 2,3,4,..,14 \qquad (3\text{-}3)$$

Once coefficient C_1 has been selected, according to [42], the second coefficient may be found using

$$C_0 = \sqrt{4 \cdot \frac{C_1}{\tan\left(2 \cdot \pi \cdot \frac{f_o}{f_s}\right)^2 + 1}} \tag{3-4}$$

where f_s is the normalised sampling frequency and f_o the normalised symbol rate. The Q-factor, Q, of the filter is then given by

$$Q = \frac{\pi \cdot f_o}{1 - \sqrt{C_1}} \tag{3-5}$$

For fixed-point implementation of the filter, the fractional components of coefficients C_1 and C_0 are converted to Q15 format as described in section 3.2.4.2. Coefficient C_1 will not be altered but often coefficient C_0 must be truncated or rounded as part of Q15 conversion; the effect of changing C_0 is to modify the centre frequency f_o of the filter. For completeness, the final centre frequency of the filter can be found by re-arranging (3-4) and is given by

$$f_o = \frac{f_s}{2 \cdot \pi} \cdot \tan^{-1}\left(\sqrt{4 \cdot \frac{C_1}{C_Q^{\,2} - 1}}\right) \tag{3-6}$$

where C_Q represents a quantised version of the value obtained for C_0 with (3-4) . Table 3-6 shows the corresponding coefficient C_0 and Q-factor for a range of coefficient C_1 values obtained from (3-3) , the final column of indicates the centre frequency f_o which results when coefficient C_0 is truncated during conversion to Q15 format.

C_1	C_0	Q-factor	f_o (fixed-point)
$\frac{4095}{4096}$	1.414040918	3216.7945157494	0.1250010058
$\frac{2047}{2048}$	1.413868253	1608.2990651228	0.1250021811
$\frac{1023}{1024}$	1.413522860	804.0513218178	0.1250010895
$\frac{511}{512}$	1.412831819	401.9274141511	0.1250023138
$\frac{255}{256}$	1.411448724	200.8653881659	0.1250012105
$\frac{127}{128}$	1.408678459	100.3342303734	0.1250019851
$\frac{63}{64}$	*1.403121520*	*50.0683598749*	0.1250016822

Table 3-6. Clock recovery IIR filter parameters for Q-factors from 50 to 3200 - target f_o= 0.125, f_s=1.

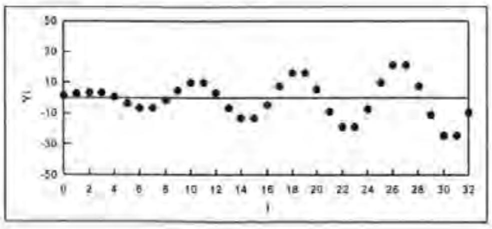
The most significant parameter in Table 3-6 is the Q-factor of the filter since this influences both clock recovery performance and fixed-point DSP implementation. Using an ideal input sample sequence x_i as stimulus, the output sample sequence Y_i was plotted for test filters with Q-factors of 50 and 200 in order to illustrate the effect of increasing the Q-factor. The input sample sequence x_i is given by

$$x_i = \begin{cases} 1 & if\ i \bmod ulo(8) \le 3 \\ -x_{i-4} & otherwise \end{cases} \tag{3-7}$$

and the output sample sequence Y_i by

$$Y_i = x_i + C_0 \cdot Y_{i-1} - C_1 \cdot Y_{i-2} \tag{3-8}$$

Figure 3-16 shows the first 32 samples (4 symbol periods) produced at the filter's output after initial application of the stimulus. For a burst demodulator symbol timing acquisition performance is of great importance, and it can be seen from Figure 3-16 that a stable timing reference is obtained after just 8 samples. The initial output is not influenced heavily by Q-factor since neither Figure 3-16a or Figure 3-16b demonstrate any significant advantage.


a. Q-factor = 50.

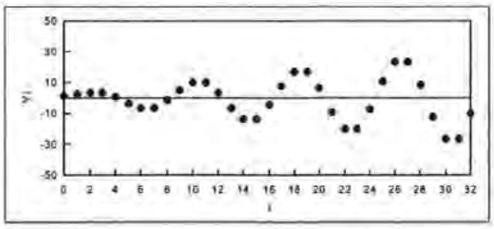
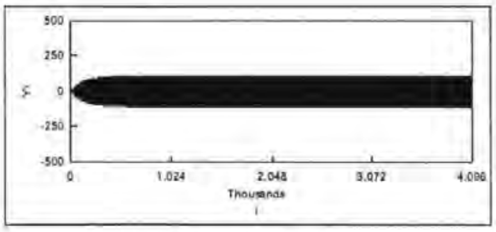

b. Q-factor = 200.

Figure 3-16. Clock recovery filter acquisition performance - optimum stimulus.

By observing the filter output for a much longer period, the effect of increasing the Q-factor is revealed. In Figure 3-17 the filter output is shown over the first 4096 samples (512 symbol periods) for a Q-factor of 50 in Figure 3-17a and for a Q-factor of 200 in Figure 3-17b. In both cases the filter output increases exponentially until a stable output level is reached. For the lower Q-factor, a peak output level of 100 is achieved after approximately 512 samples (16 symbol periods), for the higher Q-factor a peak of 400 is reached after more than 2048 samples (64 symbol periods). For fixed-point implementation it should be noted that the first filter has lower output growth and can therefore tolerate a larger range of input levels, in both cases the filter stimulus should be scaled so as not to exceed the numerical range of the target DSP. These results also suggest that the second filter (higher Q-factor) will exhibit lower clock jitter in the presence of noise since it responds more slowly to changes at it its input. For the same reason, the second filter will also take longer to adapt to the differences between transmitter and receiver symbol timing.


a. Q-factor = 50.

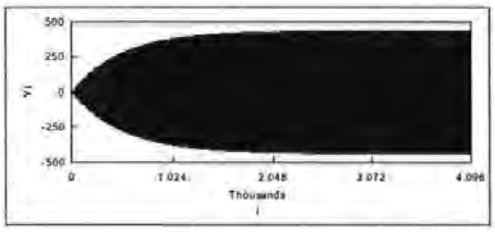
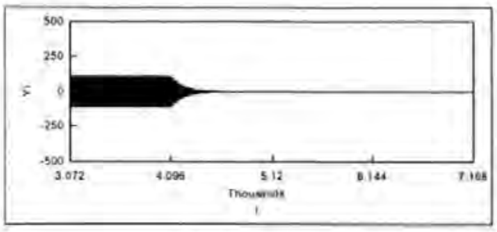
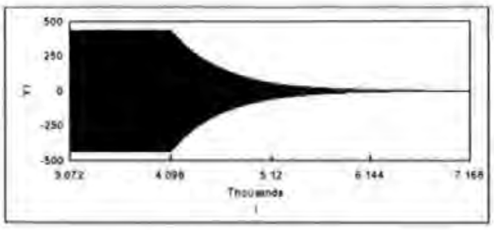

b. Q-factor = 200.

Figure 3-17. Clock recovery filter output growth - optimum stimulus.

In Figure 3-18, the stimulus has been removed at a point where the filter output has stabilised in order to asses the effect of Q-factor on flywheel performance. In both cases the output decays exponentially, over 512 samples (16 symbol periods) for the lower Q-factor and over more than 2048 (64 symbol periods) for the higher Q-factor. This confirms that a higher Q-factor corresponds to a longer flywheel period and offers greater tolerance to input fade or period of non-optimal stimulus.



a. Q-factor = 50.



b. Q-factor = 200.

Figure 3-18. Clock recovery filter flywheel performance - stimulus removed.

3.3.2 Symbol Clock Recovery DSP Implementation

The clock recovery filter must be stimulated by a signal with frequency component at the symbol rate. In the burst demodulator, a suitable signal may be derived from the frequency corrected sample sequence a_i+jb_i (see Appendix D) using a delay and multiply function. The clock recovery filter stimulus clk_ip_i is given by

$$clk\_ip_i = a_i \cdot a_{i-4} + b_i \cdot b_{i-4} \qquad (3\text{-}9)$$

It should be noted that (3-9) also produces an unwanted high frequency component, but that this is outside of the filters narrow pass band and is therefore heavily attenuated. The author's TMS320C50 assembly code implementation of the delay and multiply function is of little interest and is not shown.

Figure 3-19 shows the authors main TMS320C50 assembly code implementation of the clock recovery IIR filter. In summary, the routine executes a loop which requires 256 samples at its input and produces 256 samples at its output. Three memory pointers, AR0, AR1 & AR2, are used to reference the input buffer, filter memory and output buffer respectively. In this routine, nine instructions are required to read a sample from the input buffer and to write the filtered sample to the output buffer; representing a greater overhead than either frequency correction or matched filtering. An expression, matching the software in Figure 3-19, for recovered clock sample sequence clk_op_i is given by

$$clk\_op_i = SF \cdot clk\_ip_i + (1+(C-1)) \cdot clk\_op_{i-1} - K \cdot clk\_op_{i-2} \qquad (3\text{-}10)$$

where SF is a scaling factor for the filter stimulus clk_ip_i. In Figure 3-19, scaling is implemented with zero overhead in the line preceded by an exclamation mark. Since coefficient C has a value greater than one, for fixed-point implementation it cannot be completely represented using Q15 notation. The coefficient is therefore split into integer and fractional components, 1 and (C-1) respectively, which increases (by 1) the main loop's instruction count.

To implement the IIR filter in software, samples clk_ip_{i-1} and clk_ip_{i-2} must be constantly maintained in memory. The authors implementation requires a delay where clk_ip_{i-2} is overwritten by clk_ip_{i-1}, this may be achieved using the 'dmov' (Data MOVe) instruction. Subject to certain conditions, the TMS320C50 circular buffer feature can also be exploited to automatically provide the delay; hence reducing the instruction count by one. A circular buffer is defined by setting the buffer start and end addresses and by nominating a memory pointer; AR1 in this case. Once initialised, the pointer is reset automatically (with zero overhead) to the start address if the circular buffer's end address is exceeded. In the author's software, Figure 3-19, the 'dmov' instruction has been 'commented out' and the circular buffer technique adopted. Upon close inspection the reader will notice that the filter memory pointer (AR1) is incremented three times during the main loop and, since the filter memory buffer contains just two locations, the pointer is effectively toggled by each pass of the loop. At the start of even passes the filter memory contains samples clk_ip_{i-1} and clk_ip_{i-2}, for odd passes the contents are reversed and the filter memory contains samples clk_ip_{i-2} and clk_ip_{i-1}. Since pointer AR1 toggles during each pass, the sample clk_ip_{i-2} will always be referenced first; as is desired.

```
IIR2TAP .macro SHIFT
***************************************************************************
* 2 TAP IIR filter for clock recovery                                     *
*                                                                         *
* Example macro call;                                                     *
*   IIR2TAP SHIFT           ;MACRO - IIR 2-Tap Filter                     *
*                                                                         *
***************************************************************************

    lar     ar0,#IP           ;AR0 points to input buffer
    lar     ar1,#FILTER       ;AR1 points to filter memory
    lar     ar2,#OP           ;AR2 points to output buffer

    splk    #07feoh,COEFFK    ;Set Q15 constant for coefficient K     = 1023/1024
    splk    #034eeh,COEFFC    ;Set Q15 constant for coefficient (C-1) - 0.413522860
    .set    SHIFT 9           ;Set input scaling 1/512 (9-bit shift right)

    spm     1                 ;O/P from PREG left shifted by 1 bit
                              ;(CAN'T OVERFLOW as COEFFs never 8000h)
    splk    #(N-1),brcr       ;Repet block N times
    rptb    IIRLPF?-1         ;Start of repeated block
!    lacc    *0+,(16-SHIFT),ar1;ACCh = x(n)
     lt      *+                ;y(n-2) ¯ TREG
     mpy     COEFFK            ;COEFFK*y(n-2)/2 ¯ PREG
     lts     *                 ;y(n-1) ¯ TREG, x(n) - K*y(n-2) ¯ ACCh
     add     *+,16             ;x(n) + y(n-1) - K*y(n-2) ¯ ACCh
     mpy     COEFFC            ;COEFFC*y(n-1)/2 ¯ PREG
     apac                      ;[x(n) + C*y(n-1) - K*y(n-2)] ¯ ACCh
*    dmov    *                 ;y(n-1) -> y(n-2)
     sach    *+,0,ar2          ;y(n) := [x(n) + C*y(n-1] - K*y(n-2)]
     sach    *+,0,ar0          ;Set output
IIRLPF?                       ;End of repeated block
    spm     0                 ;PLF & Set product shift mode back to 'NO SHIFT'

    .endm
```

Figure 3-19. Clock recovery filter TMS320C50 assembly code.

Once the symbol clock has been produced, the sampling instants for the matched filtered data signals must be identified. Convention is to detect positive going zero crossings of the recovered symbol clock and to sample the data signal at these instants. In this case it was necessary to introduce a fixed delay to the symbol clock in order to correctly align its positive zero crossings with peaks in the matched filtered data signals. To aid the measurement of clock jitter, and to simplify later sections of the burst demodulator, it was decided that a function to identify the zero crossings (sampling instants) should be introduced. This function produces a sample sequence zerox$_i$ which is defined by

$$zerox_i = \begin{Bmatrix} 1 & if\left(clk\_op_{i+delay} \geq 0\right) AND \left(clk\_op_{i+delay-1} < 0\right) \\ 0 & otherwise \end{Bmatrix} \qquad (3\text{-}11)$$

where *delay* is an integer between 0 and 7 used to align zero crossings correctly with the matched filtered data samples.

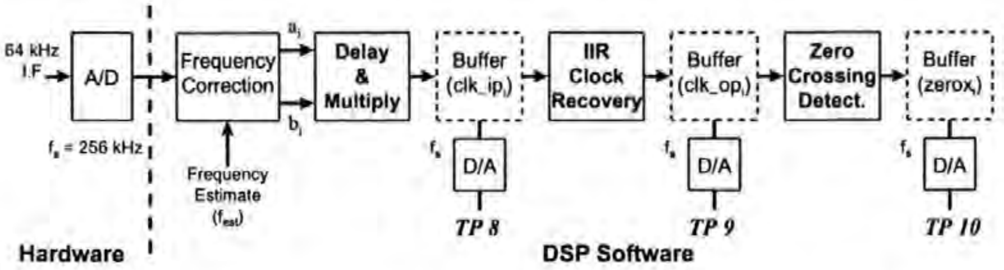
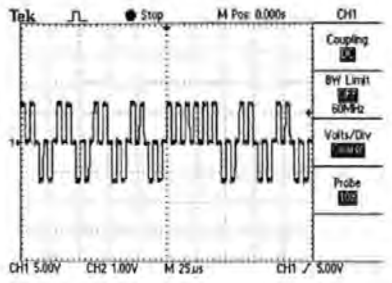
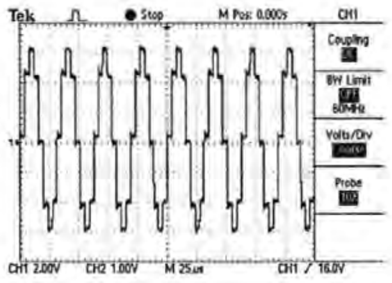


Figure 3-20. DSP burst demodulator implementation - symbol clock recovery.

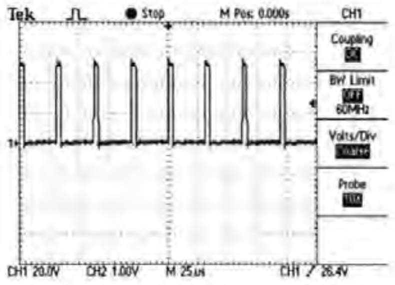
The symbol clock recovery algorithms are depicted in Figure 3-20 as a block diagram which most closely represents the author's DSP software implementation. Three test points are shown, TP8, TP9 & TP10, which allow sample sequences clk_ip_i, clk_op_i and $zerox_i$ to be monitored with an oscilloscope. A further test point TP11 (not shown) allows the correct alignment of the sampling instants to be confirmed by combining the outputs from TP10, the sampling instants $zerox_i$, and the matched filtered data sample sequence A_i (see Figure D-20). Typical signals obtained from test points TP8 - TP11 are shown in Figure 3-21 for 8 symbol periods. The frequency corrected sample sequences a_i and b_i are applied to the 'delay and multiply' function to produce the filter stimulus clk_ip_i, Figure 3-21a. Application of the stimulus to the IIR clock recovery filter produces a sample sequence clk_op_i which represents a sinusoidal symbol clock, Figure 3-21b. The high Q-factor of the filter results in a narrow pass band and hence high frequency components in the filter stimulus are heavily attenuated. The clock samples are applied to the 'zero crossing detect.' function in which positive going zero crossings in the clock signal are located. The output sample sequence $zerox_i$ defaults to zero, but is set to a high value (+ve) when an appropriate input transition is detected, Figure 3-21c. Non-zero values in the $zerox_i$ sample sequence represent the instants at which filtered data signals should be sampled, and a pre-set delay is introduced to ensure that it is correctly aligned with the filtered baseband data signal. Confirmation of the correct delay is given in Figure 3-21d where the sampling instants are combined with the matched filtered data sample sequence A_i (see Figure D-20); the sampling instants align correctly with the signal peaks.

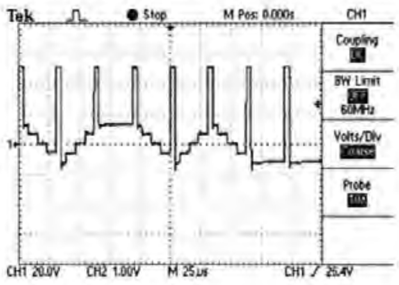


a. TP 8 - Filter stimulus (clk_ip$_i$).



b. TP 9 - Clock filter output (clk_op$_i$).





c. TP 10 - Sampling instants (zerox$_i$). d. TP 11 - Clock alignment (zerox$_i$+A$_i$).

Figure 3-21. Clock recovery signal samples (8 symbol periods shown).

3.3.3 Symbol Clock Recovery Performance

During investigations it was found that the symbol clock recovery algorithm was sensitive to carrier frequency offset at the demodulator input and that attenuation of the recovered clock signal resulted if the carrier frequency was offset from the nominal 64kHz carrier. In Figure 3-22, the clock output level is plotted in dB against carrier frequency across the whole carrier frequency acquisition range. The peak output occurs at 64kHz and attenuation rises toward -50dB as the carrier frequency is either decreased towards 48kHz or increased towards 80kHz. A stable timing reference is still produced across this frequency range for noise free conditions but, with the addition of noise, it is certain that that performance would suffer; this is confirmed by results in section 3.5.5

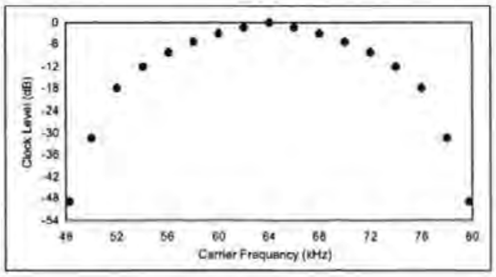


Figure 3-22. Clock recovery filter sensitivity to carrier frequency offset.

3.4 *Forward Error Correction Algorithms*

The burst transmissions in these investigations employ ½-rate convolutional encoding for forward error correction (FEC). The target burst demodulator hardware contains two TMS320C50 DSPs and it was found that a single DSP (40MHz or 80MHz varieties) could reliably execute all acquisition, synchronisation and data recovery algorithms in real-time at a transmitted symbol rate of 32ksps. However, it was concluded that a single DSP could not execute the FEC algorithm in addition to the aforementioned tasks and that this should be assigned to the second DSP. It is also necessary for the burst demodulator to interface with a personal computer, if it is to provide any useful function, and this interface should also be assigned to the second DSP. From Figure 3-23, input to DSP 1 is a sample sequence x_i representing the signal applied to the demodulator, and input to DSP 2 is the parity symbols detected from transmission bursts by DSP 1. DSP 2 applies FEC to the parity symbols and the recovered (transmitted) burst data is applied to a PC for further system processing.

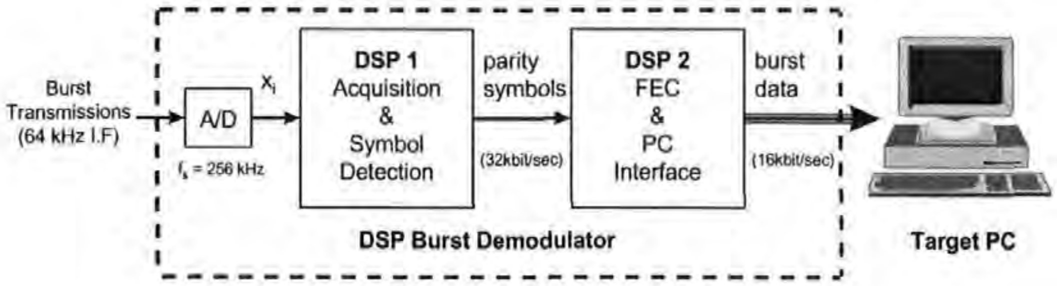


Figure 3-23. DSP burst demodulator external system interfaces.

This section is dedicated to the FEC algorithm and the techniques employed for DSP software implementation. In this section a review of convolutional coding theory is presented to enable the reader to understand the remainder of the text. The Viterbi Decoder algorithm is relatively intensive and it was not possible to achieve real-time operation with its standard software implementation. Implementation of the Viterbi Decoder algorithm is discussed in detail with particular emphasis placed upon optimisations to the algorithm employed during these investigations.

3.4.1 Review of Convolutional Coding Theory

Convolutional codes are the main alternative to block codes for FEC. They are particularly attractive when used with soft decision decoding and probabilistic decoding algorithms, such as the Viterbi Decoding Algorithm. In general, during encoding, the message is split into frames of k_0 symbols and stored in a register of length m_e frames ($m_e k_0$ symbols). Using modudulo-2 addition, a frame of n_0 symbols are generated for each input frame of k_0 symbols, Figure 3-24.

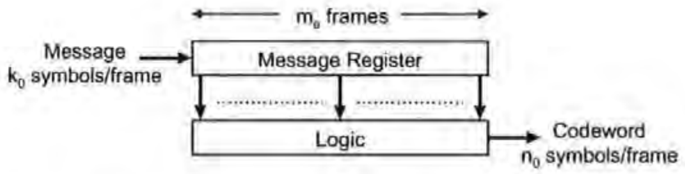


Figure 3-24. Convolutional encoder structure.

The code rate R is given by

$$R = \frac{k_0}{n_0} \tag{3-12}$$

and the total span of symbols which influence the coder output, the constraint length K, by

$$K = m_e k_0 \tag{3-13}$$

where K is the length of shift register that contains the input symbols.

A block code can be defined by a single generator polynomial, g(x), but a convolutional code must be defined by $k_0 n_0$ polynomials. Most often $k_0=1$ and this is assumed from this point onwards. A convolutional code is compactly defined by the generator matrix

$$G(x) = [g_1(x), g_2(x), \ldots\ldots, g_n(x)] \tag{3-14}$$

where

$$g(x) = \left[g_{j0} + g_{j1}x + g_{j2}x^2 + \ldots\ldots + g_{j(K-1)}x^{(K-1)}\right] \qquad (3\text{-}15)$$

Most convolutional codes are found by computer search. For example, the code defined by G=[133,171] is often used in satellite communication [66] and provides the best error correcting performance of all R=1/2 and K=7 codes; this code was selected for the 'satellite return link' investigations. It is convention to use octal notation when expressing the generator polynomials so G[133, 171] is interpreted

$$g_1(x) = 133_8 = 1011011_2 = 1 + x^2 + x^3 + x^5 + x^6$$
$$g_2(x) = 171_8 = 1111001_2 = 1 + x + x^2 + x^3 + x^6$$

The encoder for the G=[133,171] code is shown in Figure 3-25 and an important point is highlighted; The code is non-systematic because the message i(x) is not identifiable in the codeword v(x); neither $g_1(x)i(x)$ or $g_2(x)i(x)$ are identical to i(x). Non-systematic codes are preferred when Viterbi decoding is used since they generally offer the maximum possible free distance for a given rate and constraint length.

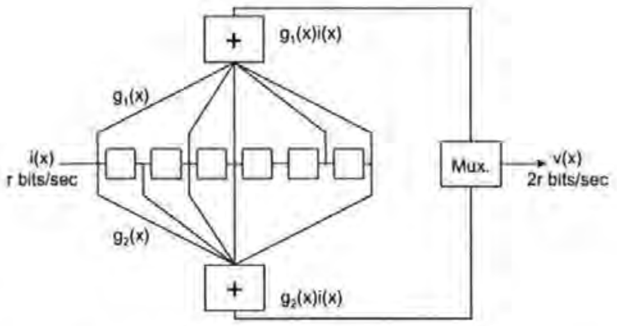


Figure 3-25. G[133,171] convolutional encoder.

Convolutional codes fall under the general heading of tree codes since their coding operation may be visualised by tracing through a coding tree. For codes with small constraint lengths, a more compact representation is the trellis diagram. The coders state is defined by the first K-1 shift register stages, and by the most recent K-1 input symbols, and therefore the coder has 2^{K-1} states with which to define a repetitive trellis structure. The R=1/3, K=3 code defined by G[5,7,7] is used to illustrate trellis representation in the example that follows. Assuming that the shift register is initially cleared (state 00), encoding of the sequence 101100 is illustrated in tabular form, Table 3-7, and then as a trellis diagram, Figure 3-26. The trellis representation forms the basis of the Viterbi Decoding Algorithm and will be referred to frequently in the text that follows.

Frame	Input i(x)	Output v(x)	Shift Register	Encoder State
0	*Reset*		00_2	00_2 (0_{10})
1	1	111	10_2	10_2 (2_{10})
2	0	011	01_2	01_2 (1_{10})
3	1	000	10_2	10_2 (2_{10})
4	1	100	11_2	11_2 (3_{10})
5	0	100	01_2	01_2 (1_{10})
6	0	111	00_2	00_2 (0_{10})

Table 3-7. G[5,7,7] convolutional code encoding example.

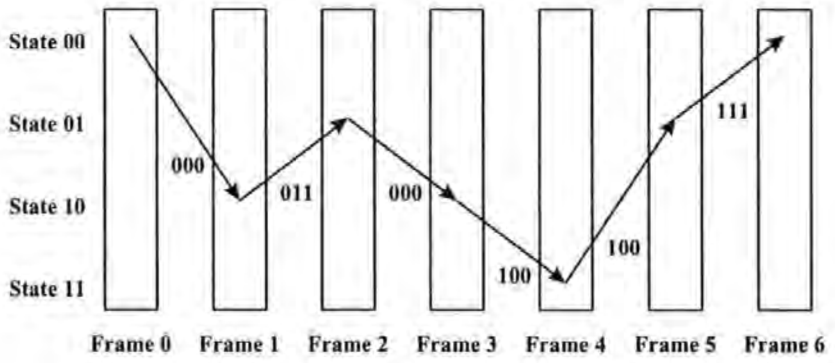


Figure 3-26. G[5,7,7] convolutional code encoding illustration - trellis diagram.

3.4.2 Viterbi Decoder Algorithm Software Implementation

This section provides a detailed description of how the important elements of the Viterbi Decoder Algorithm were implemented by the Author in software for the TMS320C50 DSP. At all times emphasis is placed upon efficient software and minimising execution time. Individual elements of the program are discussed in the order they were developed and the section ends with a description of the final algorithm. Unless otherwise stated, all discussions relate to the G[133,171] convolutional code (see Figure 3-25).

3.4.2.1 Decoder Trellis Representation

The Viterbi Decoder trellis typically consists of 5K or more frames and the author's implementation stores two items of information for each of the decoder states;

1) The running metric total.
2) An indication of the previous state (from the preceding frame).

Each frame therefore occupies 2^K locations in memory and the complete decoder trellis occupies $5\times K\times 2^K$ locations, Figure 3-27. The memory requirements may be halved if the running metrics are stored separately to avoid repetition but, since memory is a plentiful resource on the target DSP hardware, this configuration was adopted to facilitate testing and debugging.

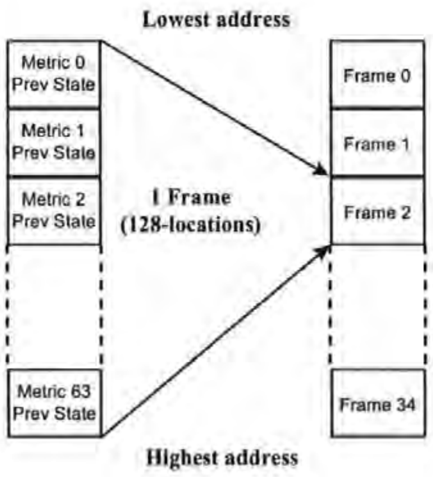


Figure 3-27. Viterbi decoder algorithm trellis - memory organisation.

It is desirable for the 'previous address' to be stored in preference to the 'previous state' so that tracing back through the trellis is made more efficient; implemented with a single TMS320C50 instruction. From Figure 3-27, the previous address *pAddress* is given in general by

$$pAddress = (pState \cdot 2) + (\#pFrame) + 1 \qquad (3\text{-}16)$$

where *#pFrame* is the start address of the previous frame.

By aligning the decoder trellis in memory so that the start address of each frame corresponds with a 128-location memory boundary, a direct relationship between the previous state *pState* and the previous address *pAddress* was established. This is particularly important when a decoded bit decision is made, after tracing back through the decoder trellis, because the start address of the oldest frame is not required. With appropriate alignment of the decoder trellis in memory, the previous state *pState* may be determined from the previous address *pAddress* directly and efficiently using

$$pState = \frac{\operatorname{mod}(pAddress)_{128} - 1}{2} \tag{3-17}$$

During operation, author's implementation of the Viterbi Decoder Algorithm refers to a 'Previous frame', 'Current frame' and 'Next frame' in the decoder trellis, Figure 3-28. The frames are addressed in a circular manner so that, after a decoded bit decision, the oldest frame in memory becomes the 'next frame' and it's contents overwritten. Upon initialisation, the 'current frame' is set with the starting metrics whereby, if the initial state is assumed to be state 0, the running metric total corresponding to state 0 is set to a high value while the remainder are set to a much lower value. This ensures that the trellis settles correctly once K-1 frames have been processed; all 2^{K-1} metrics are influenced by this starting metric once K-1 frames have been processed.

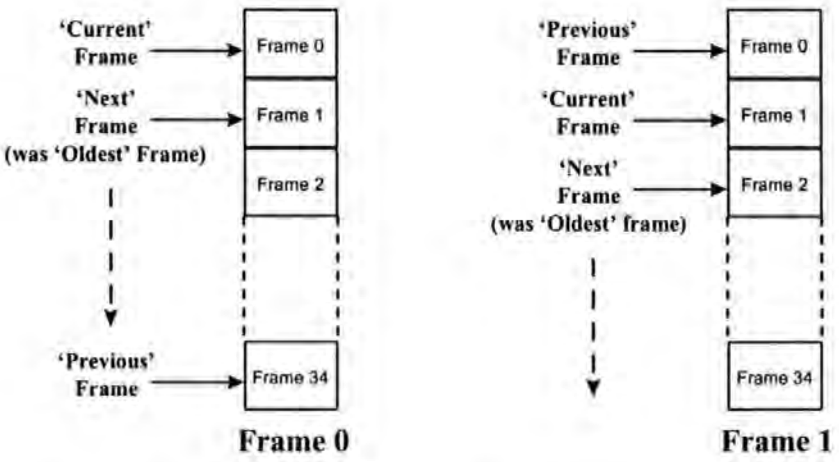


Figure 3-28. Viterbi decoder algorithm trellis - circular memory addressing .

3.4.2.2 Calculating 'Metric Updates'

The 'metric updates' are the values used to update the running metric sum for each decoder state. For the G[133,171] code, the latest parity symbols, p_0 and p_1, are compared to the 2^2 possible encoder outputs and a value proportional to the probability that each was generated by the encoder calculated as shown;

$$\begin{aligned}
P(g_1 = \text{logic '0' } \& \ g_2 = \text{logic '0'}) &\approx -1\times p_0 + -1\times p_1 \\
P(g_1 = \text{logic '1' } \& \ g_2 = \text{logic '0'}) &\approx 1\times p_0 + -1\times p_1 \\
P(g_1 = \text{logic '0' } \& \ g_2 = \text{logic '1'}) &\approx -1\times p_0 + 1\times p_1 \\
P(g_1 = \text{logic '1' } \& \ g_2 = \text{logic '1'}) &\approx 1\times p_0 + 1\times p_1
\end{aligned}$$

For a case where $p_0= \pm 1$ and $p_1= \pm 1$, the values 2, 0, 0,-2 would be computed, corresponding to cases where both symbols are correct, 1 symbol is wrong, 1 symbol is wrong and both symbols are wrong. In the author's software, p_0 and p_1 are signed integers with maximum value ± 32767. When the running metric totals are updated, scaling is used to prevent excessively high growth. Periodic conditioning of the running metric totals is also used to ensure that numerical overflow is avoided.

Starting with the 'current frame', the new input parity symbols are examined and the metrics for the 'next frame' of the decoder trellis are computed. Two states from the 'current frame' can lead to each state in the 'next frame' and the least probable path must be eliminated. For each state in the 'next frame' the new running metric total and a link to the selected state in the 'current frame' are created. For the metrics to be updated, the decoder must compute the two states which lead to each state in the 'next frame' and, from the corresponding encoder outputs, assign a probability to each based upon the parity symbols received. These probabilities are added to the running metric total for the respective paths and the lowest cumulative metric (lowest probability) determines which path is eliminated. In general, significant computation overhead can result in three essential areas;

1) Computing which states in the 'current frame' lead to a particular state in the 'next frame'.

2) Computing the probability that the parity symbols received correspond with the encoder outputs for each state transition ('metric updates').
3) Computing encoder outputs for each state transition from the 'current frame' to the 'next frame'.

Each of these areas was addressed in the author's implementation of the Viterbi Decoder Algorithm so that maximum execution speed was achieved with the TMS230C50 DSP. In each of the above cases, a LUT was created to eliminated on-the-fly or repetitive computations. These LUTs are used concurrently in the author's software but, for clarity, are described individually in the following sections.

3.4.2.2.1 'Previous States' Look-up Table

It is desirable to employ a LUT which gives the states in the 'current frame' which may lead to each state in the 'next frame'. This can significantly reduce execution time because the overhead associated with re-computing these state transitions is removed. In general, expressions for the two states, $cState_0$ and $cState_1$, that lead to the next state *nState* are given by

$$\begin{aligned} cState_0 &= (2 \cdot nState) \\ cState_1 &= (2 \cdot nState) + 1 \end{aligned} \qquad (3\text{-}18)$$

where multiplication is performed with modulo-2^{K-1} arithmetic. The LUT employed by the author, Table 3-8, takes this principle one stage further by storing address offsets (from the start of the current frame) which directly locate the 'previous address' information associated with each state in the current frame. The main benefit of this strategy is realised when tracing back through the decoder trellis.

LUT Address	cState	nState	LUT Contents
0	0	0	1
1	1	0	3
2	2	1	5
3	3	1	7
:	:	:	:
:	:	:	:
62	62	31	125
63	63	31	127
64	0	32	1
65	1	32	3
:	:	:	:
:	:	:	:
124	60	62	121
125	61	62	123
126	62	63	125
127	63	63	127

Table 3-8. 'Previous States' look-up table.

3.4.2.2.2 'Metric Update' Look-up Table

To increase software execution speed it is desirable to use a LUT in which all potential 'metric updates' (probabilities) have been pre-calculated; so as to avoid repetitive calculations. In the author's software, for each pair of parity symbols received p_0 and p_1, 2^2 'metric updates' are computed and stored in a LUT, Table 3-9. For any state transition, the encoder output may be computed and the LUT used to determine the corresponding 'metric update' without recalculating that value.

LUT Address	Encoder Output (g_1, g_2)	LUT Contents
0	00	$-p_0-p_1$
1	01	$-p_0+p_1$
2	10	$+p_0-p_1$
3	11	$+p_0+p_1$

Table 3-9. 'Metric Update' look-up table.

3.4.2.2.3 'Encoder Output' Look-up Table

The use of LUTs can also be used to provide additional software efficiency by pre-computing the encoder output associated with each state transition. In the author's software, for even greater efficiency, an address within the 'metric update' LUT is stored in place of the encoder output to eliminate all on-the-fly computations. When constructing the lookup table, the states in the 'next frame' $nState_0$ and $nState_1$, resulting from the state *cState* in the 'current frame' are given by

$$\begin{aligned} nState_0 &= \frac{cState}{2} \\ nState_1 &= \frac{cState}{2} + 2^{K-2} \end{aligned} \tag{3-19}$$

where divisions are performed using integer arithmetic. The LUT, Table 3-10, is pre-calculated and requires application of the encoding algorithm to each state transition in turn to find the corresponding encoder output, which is then added as an offset to the 'metric update' LUT base address.

LUT Address	cState	nState	LUT Contents
0	0	0	#Metric Update 0
1	0	32	#Metric Update 3
2	1	0	#Metric Update 3
3	1	32	#Metric Update 0
4	2	1	#Metric Update 1
5	2	33	#Metric Update 2
6	3	1	#Metric Update 2
7	3	33	#Metric Update 1
:	:	:	:
64	32	16	#Metric Update 1
65	32	48	#Metric Update 2
66	33	16	#Metric Update 1
67	33	48	#Metric Update 2
68	34	17	#Metric Update 3
69	34	49	#Metric Update 0
70	35	17	#Metric Update 0
71	35	49	#Metric Update 3
:	:	:	:
126	63	31	#Metric Update 0
127	63	63	#Metric Update 3

Table 3-10. 'Encoder Output' look-up table.

Analysis of Table 3-10 confirms the following properties;

1. For the 2 paths leaving any state, the encoder outputs are complementary.
2. For the 2 paths entering each state, the encoder outputs are complementary.

By exploiting these characteristics, 75% of the LUT is redundant and its length can be reduced, Table 3-11. This provides additional efficiency within the decoder software since fewer references to the LUT are required.

LUT Address	cState	nState	LUT Contents
0	0	0	#Metric Update 0
1	2	1	#Metric Update 1
2	4	2	#Metric Update 0
3	6	3	#Metric Update 2
4	8	4	#Metric Update 3
5	10	5	#Metric Update 1
6	12	6	#Metric Update 3
7	14	7	#Metric Update 1
:	:	:	:
:	:	:	:
30	60	30	#Metric Update 0
31	62	31	#Metric Update 3

Table 3-11. Optimised 'Encoder Output' look-up table.

3.4.2.3 Re-tracing Through the Decoder Trellis

Once the first K×5 frames of the decoder trellis have been computed, a decision can be made on the earliest bit. After this point has been reached, one decision is made after each new frame has been computed. To make the decision on the earliest bit, the following steps are performed;

1) Determine which state has the highest metric in the 'current frame'.
2) Using the 'last address' information corresponding to the highest metric, trace back thorough (K×5)-1 frames of the decoder trellis.
3) Using equation (3-17) , convert the final 'last address' to a decoder state.
4) The decoded data is given by the most significant bit of this state.

When the last parity symbols of the message have been processed, there are still (K×5)-1 symbols remaining within the decoder trellis. Decisions on these symbols can be made by tracing back through the decoder in a similar manner or, as is the case for these investigations, additional message bits can be transmitted to 'flush' the decoder through. It both cases, the message length must be communicated to the decoder either by prior arrangement or as part of the decoded message, section 3.1.1.

3.4.3 Modified Viterbi Algorithms

With standard implementation of the Viterbi Decoder Algorithm described in section 3.4.2 it was not possible to achieve real-time execution at the 16kbps target decoded data rate. Modifications to the standard Viterbi Decoder Algorithm were investigated in order to develop more efficient decoder software. The following discussions relate in particular to the TMS320C50 instruction set but similar performance gains can be expected with other processors.

3.4.3.1 'Double Clocked' Viterbi Decoder Algorithm

The basic idea behind the 'double clocked' Viterbi Decoder is to reduce, by half, the memory required to store the decoder trellis. This modification to the standard algorithm is particularly relevant when the memory in which to store the decoder trellis is scarce but processing power is plentiful. The 'double clocked' Viterbi Decoder stores alternate frames of the decoder trellis and, to compensate for the modified trellis, the decoder makes decisions on two decoded bits each time it traces back through the trellis, Figure 3-29.

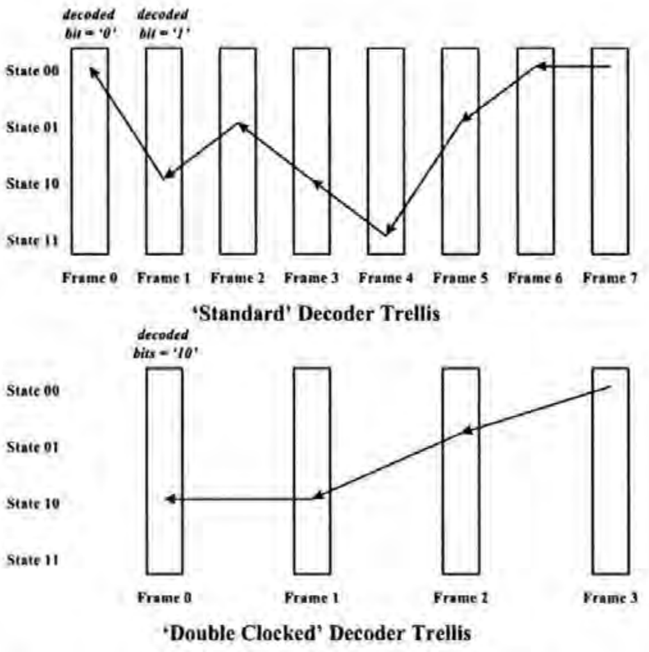


Figure 3-29. Comparison of decoder trellis and decoded bit decisions for 'Standard' and 'Double Clocked' Viterbi decoder algorithms - (K=3 code).

Figure 3-29 shows, for a constraint length K=3 code, a comparison of the decoder trellis and decoded bit decisions for 'standard' and 'double clocked' decoders. Clearly, the 'double clocked' decoder halves the number of frames that must be stored in memory and as a result, halves the memory required to implement the decoder trellis, halves the number of frames that must be retraced and also halves the frequency at which the trellis must be retraced. The most significant points from Figure 3-29 are that the frames labelled 1, 3, 5 and 7 for the standard trellis are identical to the frames labelled 0, 1, 2 and 3 in the 'double clocked' trellis, and that the oldest two decoded bits are given by the most significant bits of the state reached in frames 0 and 1 respectively in the standard trellis but by the two most significant bits of the state reached in frame 0 of the 'double clocked' trellis. It should be noted that the oldest two decoded bits are also given by the state reached in frame 1 of the standard decoder trellis and that this is exploited in the algorithm developed by the author in later discussions.

For the 'double clocked' decoder to offer a performance advantage over the standard decoder it must be implemented so that the average number of instructions required for each decoded bit is lower. The 'double clocked' decoder does offer advantages in terms of the number of states which must be retraced in order to make decoded bit decisions and also in terms of the frequency at which the trellis must be retraced. However, for the 'double clocked' decoder, the process of computing each frame committed to memory is made more intensive and more paths must be eliminated; equating to inefficiency. For the standard decoder there are just two paths leading to each state in the following frame and only one path must be eliminated, for the 'double clocked' decoder there are four potential paths and three must be eliminated, Figure 3-30.

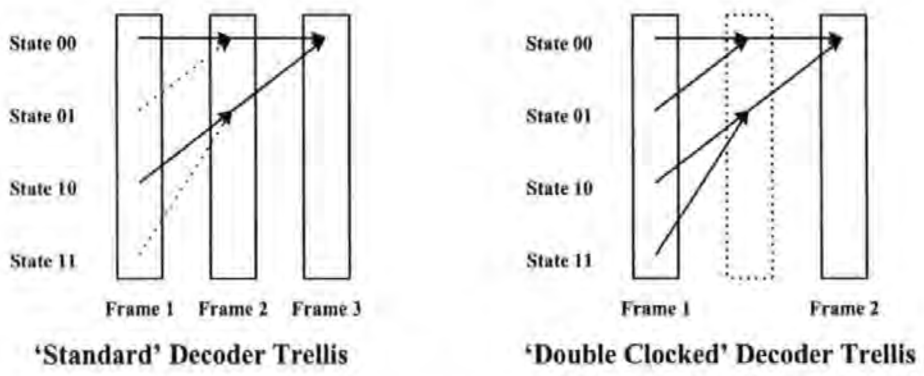


Figure 3-30. Comparison of path elimination for 'Standard' and 'Double Clocked' Viterbi decoder algorithms - (K=3 code).

To find the best metric for state 0 in Figure 3-30, the 'double clocked' decoder must compute four potential metrics and subsequently eliminate three of the corresponding paths; this is repeated for all 2^{K-1} states. For the standard decoder to achieve the same goal, two metrics are produced for each state and only one path is eliminated, however the whole process is repeated for two frames. This comparison is presented more clearly in Table 3-12. In terms of the number of metrics computed, neither algorithm produces significant overhead providing 'metric update' LUTs are utilised, section 3.4.2.2.2. In terms of the number of metrics that must be eliminated, the standard decoder algorithm is clearly preferable.

Task	'Standard' Decoder	'Double Clocked' Decoder
Total Metrics Computed	$2\times(2\times2^{K-1})$ - 2 parity symbols	$1\times(4\times2^{K-1})$ - 4 parity symbols
Total Metrics Eliminated	$2\times(2^{K-1})$	$1\times(3\times2^{K-1})$

Table 3-12. Comparison of metric computation overhead in 'Standard' & 'Double Clocked' Viterbi decoder algorithms.

From these investigations, the most significant benefit identified of the 'double clocked' decoder was that of reduced memory utilisation. Gains in terms of software efficiency were exhibited in the decoded bit decision process because the decoder trellis is retraced less frequently and decisions are made on two bits simultaneously. However, with the TMS320C50 DSP's instruction set and in the time available, the author was unable to implement the double clocked decoder more efficiently than the standard decoder due to increased complexity when eliminating three out of the four potential paths from one frame to each state in the following frame. However, the areas of the algorithm identified as more efficient were exploited by the author to create another variation of the Viterbi Decoder Algorithm which did show improvement in terms of execution speed for the TMS320C50 DSP, section 3.4.4.

3.4.4 Optimised Viterbi Decoder Algorithm for the TMS320C50

After investigating the 'Double-Clocked' Viterbi Decoder Algorithm, the author was able to develop an algorithm which exhibits faster execution speed on the TMS320C50 DSP than a standard implementation of the Viterbi Decoder Algorithm. It was found in practice that the new algorithm increases the maximum output rate for a K=7 decoder from 13 kbit/sec to 19 kbit/sec (40MHz TMS320C50 DSP). The target operating rate was 16 kbit/sec so clearly the new algorithm made the difference between meeting and failing to meet the requirement of working in real-time.

The principle of the 'double-clocked' decoder can be extended to a 'triple-clocked' to further reduce the memory required to store the decoder trellis. In this situation, every third frame of the decoder trellis would be computed directly and decoded bit decisions made at 1/3rd rate on 3 bits simultaneously. As shown in section

3.4.3.1, additional efficiency would result because the trellis must be retraced fewer times but inefficiency also results due to the increased number of paths from one frame to the next, Table 3-13. In fact, the decoder algorithm may be over-clocked at much higher rates and the principle is only limited by the constraint length K of the code; by definition, any state in the decoder trellis indicates the previous K-1 encoded bits. Based upon this limit, the author's modification to the Viterbi Decoder Algorithm for the G[133,171] code retraces the decoder trellis at 1/6th rate and makes decoded bit decisions on 6 bits simultaneously to maximise the associated efficiency gains. The author's modification does not provide a corresponding (1/6th) reduction in memory utilisation since the full decoder trellis is computed in order to eliminate the inefficiencies identified.

Task	**'Standard' Decoder**	**'Triple Clocked' Decoder**
Total Metrics Computed	$3\times(2\times2^{K-1})$ - 2 parity symbols	$1\times(8\times2^{K-1})$ - 8 parity symbols
Total Metrics Eliminated	$3\times(2^{K-1})$	$1\times(7\times2^{K-1})$

Table 3-13. Comparison of metric computation overhead in 'Standard' & 'Triple Clocked' Viterbi decoder algorithms.

In terms of the decoder's path history length it is necessary to increase the decoder trellis from length 5K frames to (5K)+6 to ensure that the latest of the decoded bits have equal probability of being decoded correctly as in the standard Viterbi Decoder Algorithm. The overhead introduced by an extension to the decoder trellis is negligible because retracing through the trellis was implemented by the author with a single TMS320C50 instruction repeated an appropriate number of times; the TMS320C50 DSP is most efficient when it executes a 'single instruction' repeat loop [52].

3.4.5 Performance Analysis

The author's DSP implementation of the modified Viterbi Decoder Algorithm was tested in isolation from the DSP burst demodulator so that its performance could be compared to theoretical BER (bit error rate) curves for the G[133,171] convolutional code. For this test, DSP software was produced by the author to generate the input parity symbols, add noise to the decoder input (from AWGN samples embedded in the program EPROMs) and compare the decoded bits with the source bits. Additional functions allowed the history of state transitions to be compared to the decoder's state transitions in order to trap programming and implementation errors. In this case the operating speed of the decoder was not of concern and the test software is best illustrated as a block diagram, Figure 3-31.

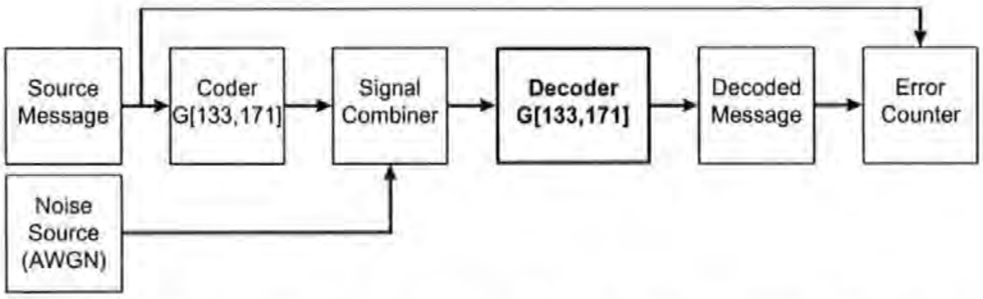


Figure 3-31. Block diagram of DSP Viterbi decoder test software.

For the performance tests, the noise power was kept constant and the signal power varied. Due to limited memory in the target DSP hardware, only BERs down to approximately 10^{-4} could be measured. The purpose of these tests were to confirm that the decoder produced a BER curve similar to that predicted by theory for the G[133,171] convolutional code; more comprehensive tests were conducted later, section 3.5, on the Burst Demodulator as a whole. The following table and graph show results from the BER test and comparison with a theoretical curve. The experimental results confirm that the author's modified Viterbi Decoder algorithm performs as expected.

E_b/N_0	Decoded Errors (out of 5120 bits)	BER
3.02	734	1.43×10^{-1}
3.31	687	1.34×10^{-1}
3.75	413	8.07×10^{-2}
3.76	375	7.32×10^{-2}
3.78	425	8.30×10^{-2}
4.77	81	1.58×10^{-2}
4.82	58	1.13×10^{-2}
5.26	7	1.37×10^{-3}
5.26	14	2.73×10^{-3}
5.71	4	7.81×10^{-4}
5.71	7	1.37×10^{-3}
6.84	0	0
7.85	0	0
11.37	0	0

Table 3-14. Experimental BER test results - G[133,171] code.

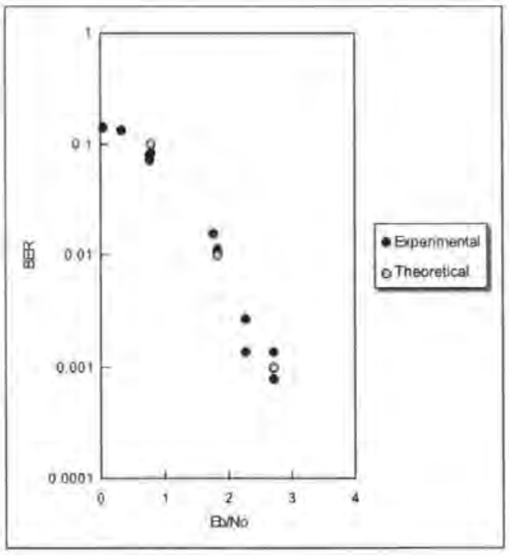


Figure 3-32. Experimental & theoretical BER curves - G[133,171] code.

3.5 Burst Demodulator Performance

The burst demodulator was evaluated extensively using simulated signals generated by a PC so that a range of tests could be conducted. An opportunity was also made available to the author to conduct preliminary satellite trials. In this section, results from the both simulated and practical trials are presented.

3.5.1 Simulated Test Rig

For early trials, software was written by the author to generate simulated test signals on-the-fly with a PC for direct (digital) application to the burst demodulator. This method was selected because it gave complete control over all parameters and in particular, frequency, signal level, noise level and modulation mode. The same software also accepts the burst demodulator data output and produces automated statistics based upon a comparison of the packets transmitted and the data recovered from them, Figure 3-33.

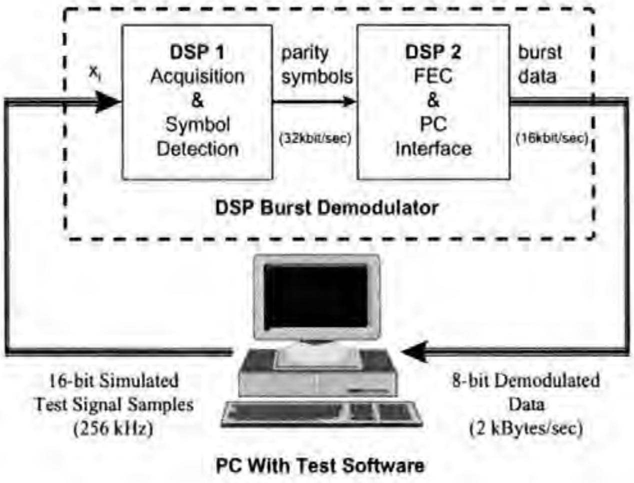


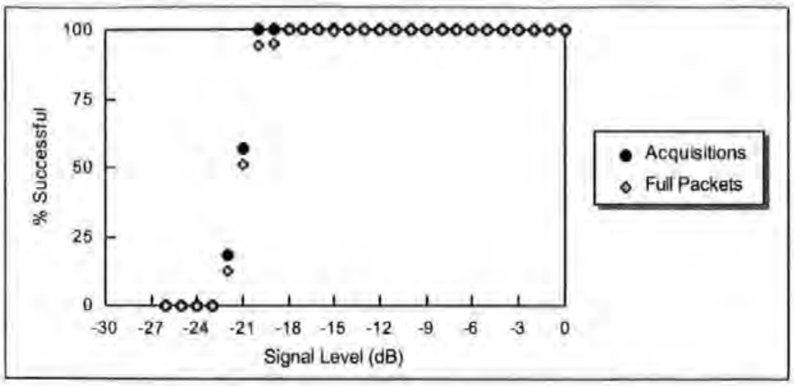
Figure 3-33. Simulated test-rig for DSP burst demodulator.

The links from the PC to the burst demodulator are made using a proprietary 16-bit parallel interface card designed (by others) for this purpose. From PC to DSP, all 16-bits of the interface are used to pass 16-bit signed integer samples across the link at a rate proportional to 256ksps. A low-level communication protocol was implemented from the DSP to PC as an aid to testing and for synchronisation purposes. In this direction, 8-bits act as status flags for the DSP to indicate the beginning of each data burst and the remaining 8-bits bits carry demodulated data (re-assembled into bytes). It was not possible to achieve real-time operation with the simulated test rig because a PC is slow in comparison to the DSP hardware. However, since the test rig forms a closed loop, operating speeds are irrelevant.

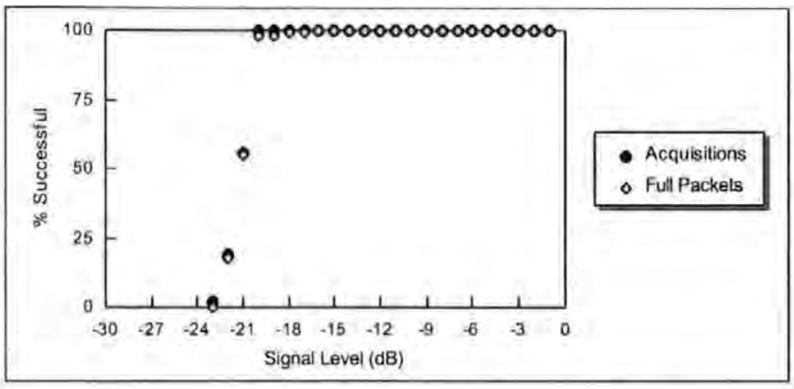
Unless otherwise stated, all tests were conducted by transmitting 2,500 packets and varying either signal level, AWGN noise level or carrier frequency. References to default parameters can be interpreted as a signal level corresponding to half the dynamic range of the ADC and a carrier frequency of 64kHz in the following sections.

3.5.2 Sensitivity to Input Signal Level

Since the burst demodulator software was implemented using a fixed-point DSP, and because scaling is applied within several algorithms to prevent numerical overflow, it was necessary to determine the sensitivity of the demodulator to input signal level at the nominal carrier frequency (64kHz). This was achieved by first sending 2,500 (noise free) test packets at a signal level corresponding to the maximum dynamic range of the ADC so as to generate a 0dB reference level. The test was then repeated at successively reduced signal levels until complete failure was observed. For each test, the number of acquisitions and error free packets was plotted for a BPSK version of the demodulator in Figure 3-34a and for a QPSK version of demodulator in Figure 3-34b. The results indicate an 18dB operating range over which 100% reliability is achieved for packet reception in a noise free environment. This relatively low figure is due to the extensive scaling employed in the acquisition algorithm and the need to avoid numerical overflow; an 18dB operating range however is adequate in a satellite receiving environment. A direct correlation between acquisition and error free packet probabilities suggests that performance is limited in this case by the sensitivity of the acquisition algorithm.



a. BPSK Demodulator - 56-Byte test packets.



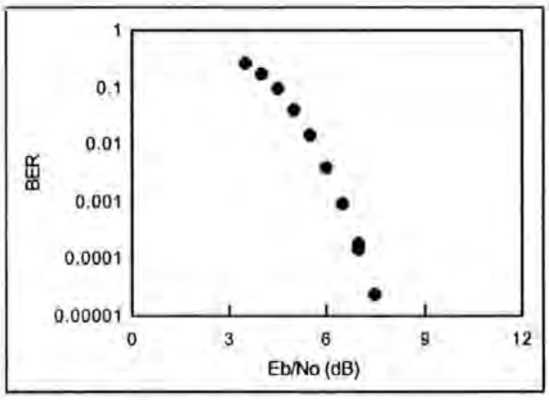
b. QPSK Demodulator - 112-Byte test packets.

Figure 3-34. Burst demodulator sensitivity to input signal level (noise free).

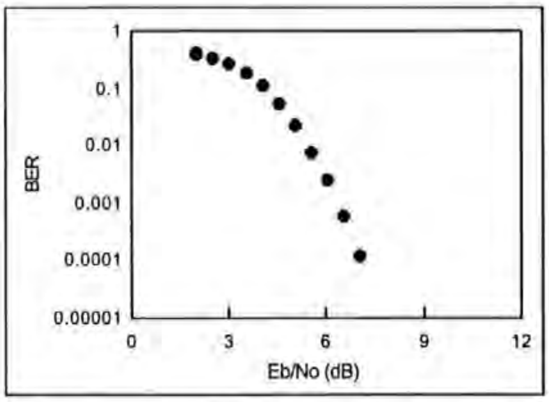
3.5.3 BER Performance

BER performance of both BPSK and QPSK demodulators was measured to allow comparisons to be made. It is important to stress that BER could only be measured if acquisition was conducted successfully and that the results presented do not take account of packets that are 'missed' due to acquisition failures. i.e. in the event of acquisition failure, no received bits or bits in error can be recorded. Acquisition failures are frequent at low E_b/N_0, which extends the time between measurements of decoder errors. BER curves were plotted for both BPSK and QPSK versions of the demodulator by transmitting 2,500 750-byte test packets and comparing the data demodulated to the data transmitted for packets received in full only. In this case the signal level corresponded with half the dynamic range of the ADC and E_b/N_0 was altered by increasing the noise signal level. The BER curves for

BPSK and QPSK versions of the demodulator, Figure 3-35a and Figure 3-35b respectively, are similar and in line with theoretical predictions.



a. BPSK demodulator - 750-byte packets.



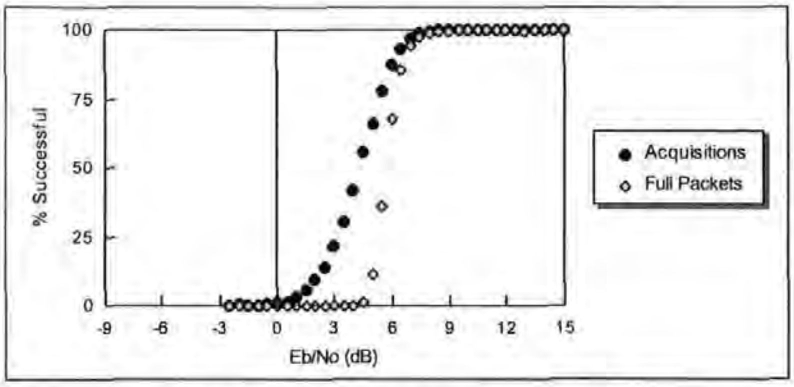
b. QPSK demodulator - 750-byte packets.

Figure 3-35. Burst demodulator BER performance.

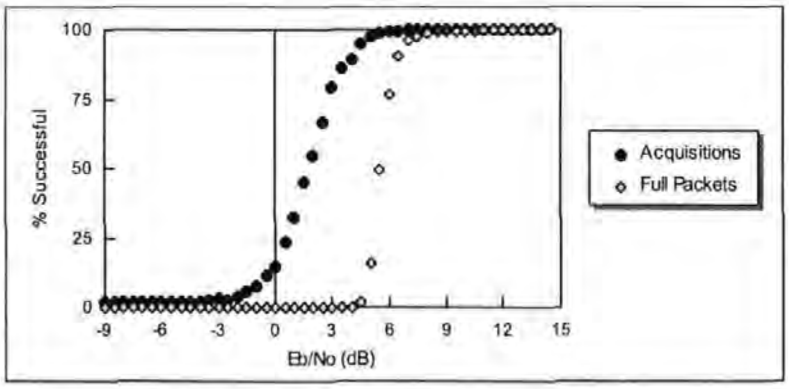
3.5.4 Error-Free Packet Performance

Of greatest significance to these investigations is the demodulator's error-free packet performance; the ability to demodulate error free packets. For these tests, 2,500 short packets were transmitted at the nominal carrier frequency (64kHz) and the acquisition and error-free packet probabilities plotted for both BPSK and QPSK versions of the demodulator, Figure 3-36a and Figure 3-36b respectively. In this case, the signal level corresponded with half the dynamic range of the ADC and E_b/N_0 was altered by varying the noise level. In terms of the probability of error-free packets being demodulated, both BPSK and QPSK demodulators exhibit similar performance.

The acquisition performance for the QPSK demodulator appears to offer a 3dB improvement over the BPSK demodulator, but this is explained by the fact that the preamble is transmitted with twice the power in QPSK mode. To achieve a 99% probability of demodulating error-free packets, it can be seen that an E_b/N_0 of between 7dB and 8dB is required.



a. BPSK demodulator - 56-byte packets.

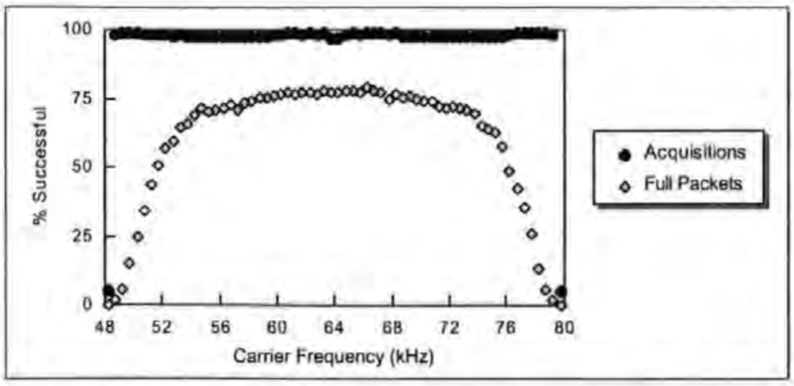


b. QPSK demodulator - 112-byte packets.

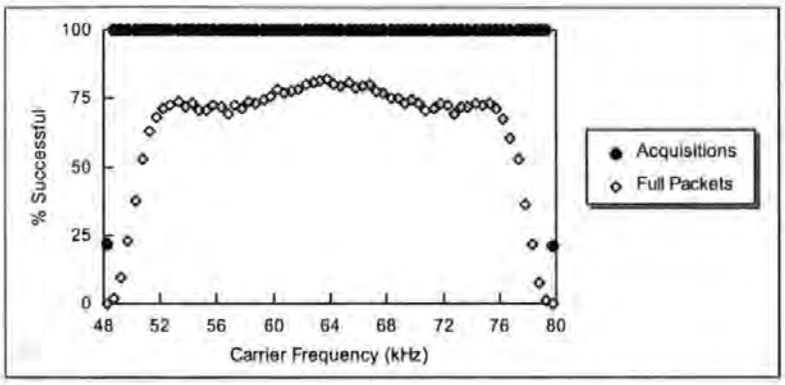
Figure 3-36. Burst demodulator error-free packet performance.

3.5.5 Sensitivity to Carrier Frequency Offset

A final point of interest was the demodulators sensitivity to carrier frequency offset. For these tests, a fixed E_b/N_0 corresponding to 6dB was selected so as to operate within a range at which failures would be measured. The test was repeated at 0.5kHz intervals over the entire range of the carrier frequency acquisition algorithm, 48kHz to 80kHz, for both BPSK and QPSK versions of the demodulator. The acquisition and error-free packet performance is shown in Figure 3-37 and the BER performance in Figure 3-38.



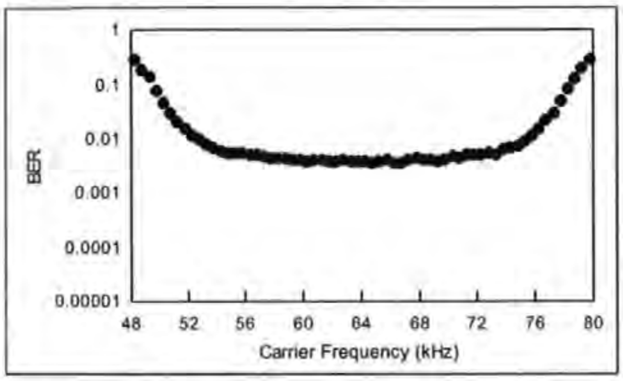
a. BPSK demodulator - 56-byte packets, E_b/N_0=6dB.



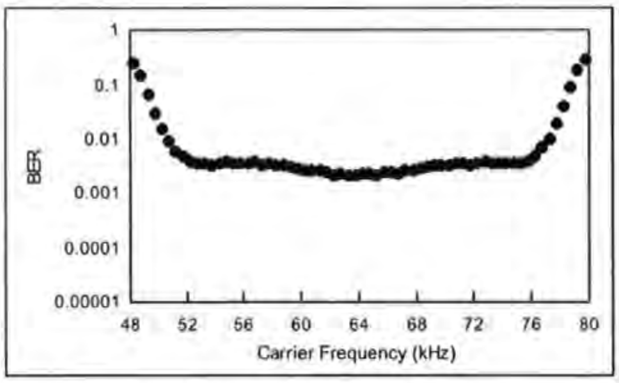
b. QPSK demodulator - 112-byte packets, E_b/N_0=6dB.

Figure 3-37. Burst demodulator sensitivity to carrier frequency offset.

From Figure 3-37 it can be seen that error-free packet performance is reduced as the carrier frequency is offset from 64kHz and that it reduces significantly when the carrier frequency approaches 48kHz and 80kHz. Comparison with Figure 3-38 indicates that there is a direct correlation between the BER performance and that of error free packet reception. These characteristics may be largely attributed to the sensitivity of the symbol timing algorithm to carrier frequency offset, section 3.3 Figure 3-22.



a. BPSK demodulator - 56-byte packets, E_b/N_0=6dB.



b. QPSK demodulator - 112-byte packets, E_b/N_0=6dB.

Figure 3-38. Burst demodulator BER sensitivity to carrier frequency offset.

3.5.6 Satellite Trials

A series of satellite trials were conducted using European Space Agency's Digilease widebeam transponder on the Eutelsat II F3 satellite located at 7° East. The uplink used vertical polarisation and the downlink horizontal. Satellite reception and monitoring was provided by the European Space Agency's TDS4B Satellite Earth Station located at the University of Plymouth. Known data packets were transmitted from a return link terminal and received by TDS4B where the signal was down-converted to a 70MHz I.F. The 70MHz I.F. was further down-converted to the 64kHz for compatibility with the burst demodulator by with hardware developed (by others) for this purpose. A series of back-to-back tests were also conducted at the 70MHz I.F. so that comparisons could be made. The BER performance is plotted in Figure 3-39 and compared to theoretical predictions for differentially detected BPSK with ½-rate soft decision convolutional FEC. It can be seen that there is a 0.7dB degradation from the theoretical for the back-to-back test while satellite tests introduced a further 0.3dB degradation.

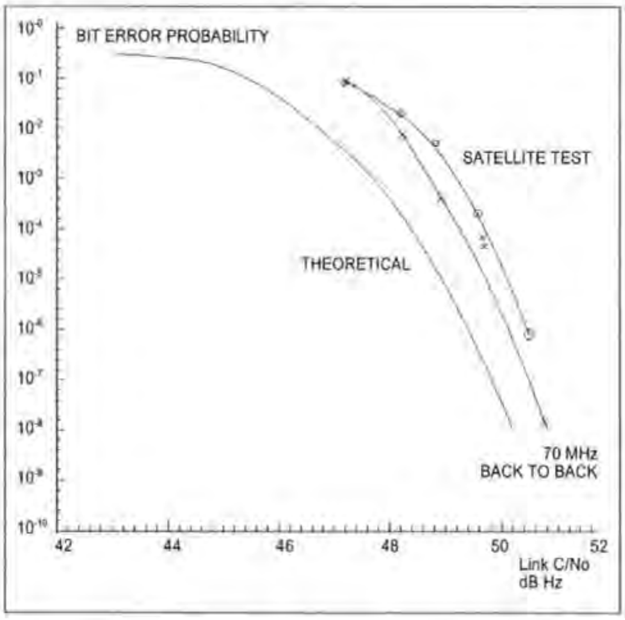


Figure 3-39. Burst demodulator BER performance.

The packet acquisition performance is plotted in Figure 3-40. For an acquisition probability of 99% or better, an E_b/N_0 ratio of 6.8dB is required for back-to-back tests while approximately 8dB is required to guarantee successful acquisition over the satellite.

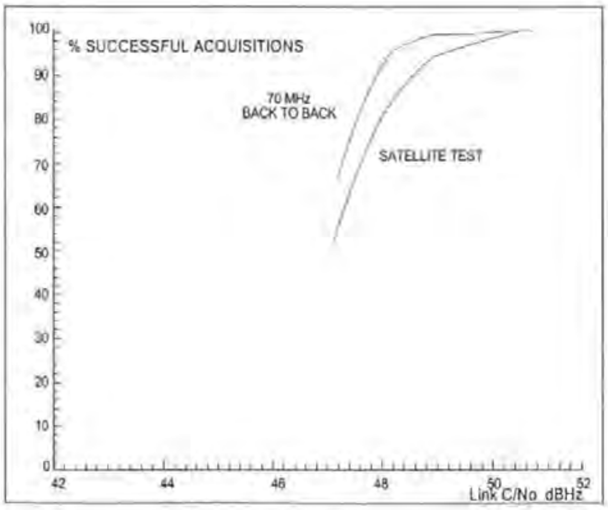


Figure 3-40. Burst demodulator packet acquisition performance.

3.6 Summary and Conclusions

In section 3.2 the author discussed carrier frequency acquisition using FFT based algorithms. The Offset-FFT was introduced and improved performance over the FFT, for little processing overhead, was demonstrated in terms of probability of acquisition and resolution of frequency estimation. The author's real-time DSP implementation of an OFFT carrier frequency acquisition algorithm was discussed and results from performance evaluation presented. Subtle mechanisms were identified, carrier frequency offset and aliasing, and their impact on carrier frequency acquisition were characterised. Combined with a 'delay and multiply'/ IIR resonator filter clock recovery technique, section 3.3, both carrier frequency acquisition and symbol timing synchronisation were achieved with a 32 symbol preamble. The <u>same</u> signal samples from the preamble provide a suitable frequency spectrum for carrier frequency acquisition and also provide the optimum stimulation for the clock recovery circuit. For these investigations the acquisition time (preamble duration) is set by the time taken for the DSP to execute the Offset-FFT algorithm. For the same transmitted symbol rate, an increase of DSP execution speed would allow a decrease to the preamble duration to be made. The implementation of a complete burst demodulator is also discussed in the Chapter and Appendix, and many compromises and optimisations were made for real-time execution to be achieved. In particular a modification to the Viterbi Decoder Algorithm was described in section 3.4 which helps to reduce processing overheads to acceptable levels. Evaluation of the burst demodulator performance in simulated and satellite trials, section 3.5, shows that implementation losses are well within acceptable levels.

4. Synchronisation of OFDM Demodulators for Satellite Applications

4.1 Introduction

Orthogonal Frequency Division Multiplexing (OFDM) is now established as the preferred modulation method for transmitting digital terrestrial radio and television services [35]. Its appeal is largely due to the relatively long symbol period and a 'guard' interval which provides protection against multipath echoes and interference from adjacent transmitters in a single frequency network. Current terrestrial systems also transmit differentially encoded data in order to reduce receiver complexity. In this chapter the author proposes using coherent demodulation for satellite based OFDM systems. For satellite systems the problems associated with multipath propagation and single frequency networks do not exist, and so the guard interval may be omitted in order to increase system efficiency. In addition, it is well known that coherent demodulation provides a 3dB improvement over differential encoding.

	Transmitted OFDM symbol →											
i/p data ↓	s_0	s_1	s_2	....	s_n	s_{n+1}	s_{n+2}	....	s_m	s_{m+1}	s_{m+2}	....
c_0	0	0	0	....	0	0	0	....	-1+j	-1+j	-1+j	....
c_1	0	0	0	....	1+j	-1-j	1+j	....	1+j	1+j	1+j	....
c_2	1+j	1+j	1+j	....	0	0	0	....	-1-j	-1-j	-1-j	....
:	:	:	:	:	:	:	:	:	:	:	:	:
c_{N-1}	0	0	0	....	0	0	0	....	1-j	1-j	1-j	....
	Frequency synchronisation (8 OFDM symbols)				*Coarse timing synchronisation (6 OFDM symbols)*				*Fine timing synchronisation (6 OFDM symbols)*			

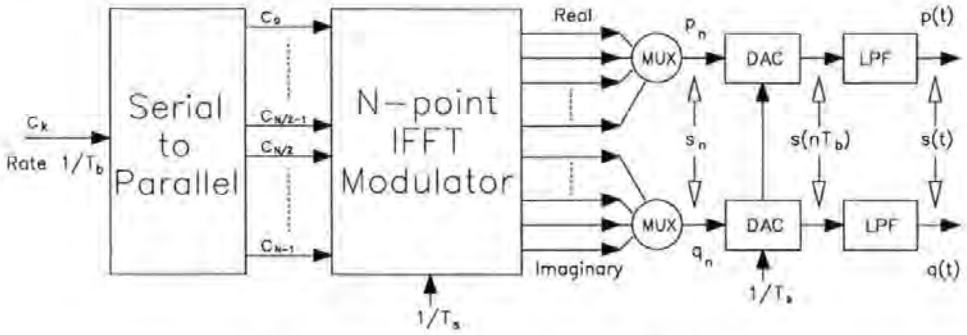
Table 4-1. Typical preamble data structure for OFDM carrier frequency and symbol timing synchronisation.

Frequency and symbol timing synchronisation are critical factors in OFDM receivers, and particularly in the case of coherent demodulation. This chapter describes algorithms for frequency and symbol timing synchronisation based upon the preamble sequence shown in Table 4-1; with the design of such a receiver discussed later. Table 4-1 indicates a suitable symbol sequence $\{s_j\}$ for an N-channel QPSK-OFDM scheme given complex data symbols c_i, $i = 0, 1, \ldots, N-1$. Small differences in the crystal timing references between modulator and demodulator result in demodulator symbol timing drift over time, whilst satellite motion has a similar effect on the carrier frequency. The preamble sequence must therefore be repeated at regular intervals to allow frequency and symbol timing drift to be tracked, and re-synchronisation to be performed. Residual errors in carrier frequency synchronisation manifest as phase errors in the demodulated data which must be corrected using phase tracking techniques. Residual symbol timing error may be continuously monitored and adjustments made during demodulation.

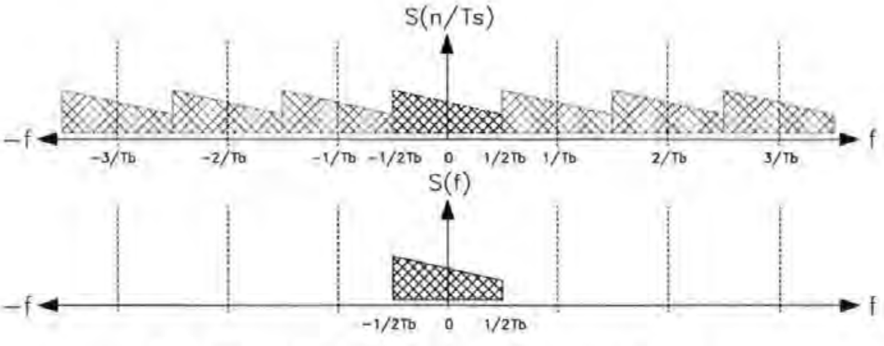
There are two distinct methods for generating the same QPSK-OFDM signal; the 'complex-sampling' and 'real-sampling' modulator models. Similarly, there are two distinct methods for demodulating this QPSK-OFDM signal; the 'complex-sampling' and 'real-sampling' demodulator models. Each model offers practical advantages in terms of either analogue or digital hardware requirements and are described, mathematically analysed and compared in Appendix E. The chapter begins with a brief review of OFDM systems and terminology. The remainder of the text is divided into two sections; frequency synchronisation and symbol timing synchronisation. For each type of error the effect on the demodulated data is discussed and suitable preamble sequences and synchronisation algorithms are presented. The algorithms are verified by simulation.

4.2 OFDM System Models

The ‘complex-sampling’ model was selected for these investigations due to lesser digital hardware requirements; hence lower overheads for DSP software implementation. The models used for theoretical investigations are described briefly below.



a. Complex sampling QPSK-OFDM modulator.



b. QPSK-OFDM modulator baseband spectra.

Figure 4-1. Baseband QPSK-OFDM modulator and spectra.

Efficient generation of a baseband N-channel QPSK-OFDM signal uses an N-point complex input IFFT, Figure 4-1a. The complex input symbols $c_k = a_k+jb_k$ are at rate $1/T_b$ Hz, where $a_k, b_k \in \{-1,+1\}$ for QPSK, and modulation extends the symbol period to $T_s = NT_b$ seconds. The sampled baseband OFDM signal $s(nT_b)$ is therefore

$$s(nT_b) \equiv s_n = \sum_{k=0}^{N-1} c_k \cdot e^{j\frac{2\pi kn}{N}} \qquad n = 0,1,\ldots,N-1 \qquad (4\text{-}1)$$

Since equation (4-1) represents a sampled signal, images appear throughout the frequency spectrum at multiples of the sampling frequency $1/T_b$ Hz. However, the maximum frequency that can be generated at the output of each individual DAC is $1/2T_b$ Hz; half the sampling frequency. After low pass filtering, the continuous time signal s(t) = p(t) + jq(t) is given by

$$s(t) = \sum_{k=0}^{N-1} c_{k+\frac{N}{2}\bmod(N)} \cdot e^{j\frac{2\pi\left(k-\frac{N}{2}\right)t}{T_S}} \qquad 0 \le t < T_s \qquad (4\text{-}2)$$

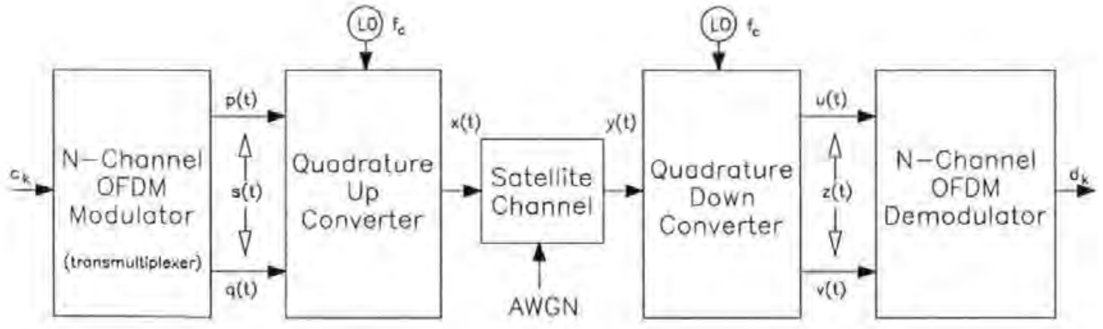
and contains N unique subchannels over the band $-1/2T_b$ to $1/2T_b$ Hz. In (4-2) the lowest subcarrier ($-1/2T_S$ Hz) is modulated by QPSK symbol $c_{N/2}$ and the highest subcarrier is modulated by QPSK symbol $c_{N/2-1}$.

Figure 4-2a shows an OFDM system overview. The complex baseband signal s(t) is applied to a quadrature up converter to give a real signal x(t), which is centred at a carrier frequency f_c Hz and suitable for transmission over the satellite channel. Since this discussion is concerned only with carrier frequency and symbol timing errors in the OFDM system, linear distortion and noise in the satellite channel are neglected. In this case, the downconverter input y(t) is identical to the upconverter output x(t)

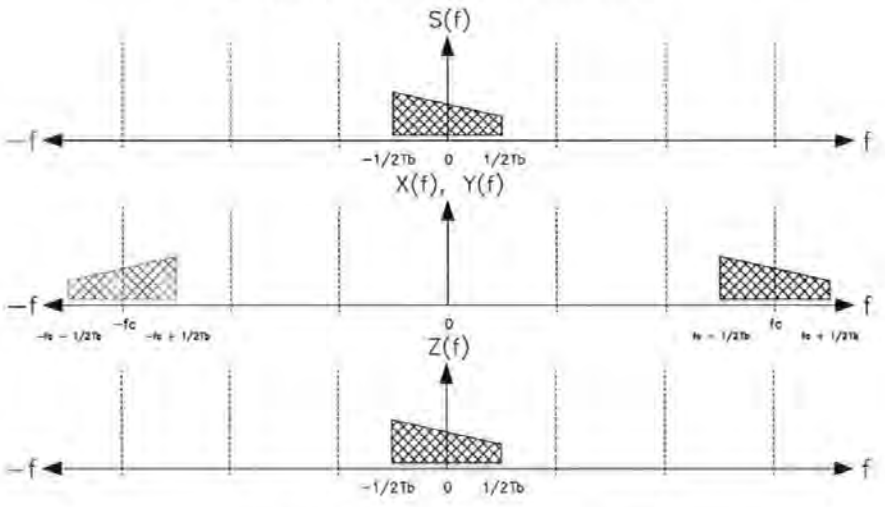
$$y(t) \equiv x(t) = \mathrm{Re}\left[\sum_{k=0}^{N-1} c_{k+\frac{N}{2}\bmod(N)} \cdot e^{j2\pi\left(f_c + \frac{k-\frac{N}{2}}{T_S}\right)\cdot t}\right] \qquad 0 \le t < T_s \qquad (4\text{-}3)$$

and the demodulator input z(t) is identical to the modulator output s(t)

$$z(t) \equiv s(t) = \sum_{k=0}^{N-1} c_{k+\frac{N}{2}\bmod(N)} \cdot e^{j\frac{2\pi\left(k-\frac{N}{2}\right)t}{T_S}} \qquad 0 \le t < T_s \qquad (4\text{-}4)$$



a. QPSK-OFDM system overview.



b. OFDM system spectra

Figure 4-2. QPSK-OFDM system overview and spectra.

A QPSK-OFDM demodulator is shown in Figure 4-3a. The components of the continuous time demodulator input z(t) = u(t) +j v(t) are bandlimited to $1/2T_b$ Hz and can be sampled at rate $1/T_b$ Hz

$$z(nT_b) \equiv z_n = \sum_{k=0}^{N-1} c_{k+\frac{N}{2}\bmod(N)} \cdot e^{j\frac{2\pi\left(k-\frac{N}{2}\right)n}{N}} \qquad n = 0,1,\ldots,N-1 \qquad (4\text{-}5)$$

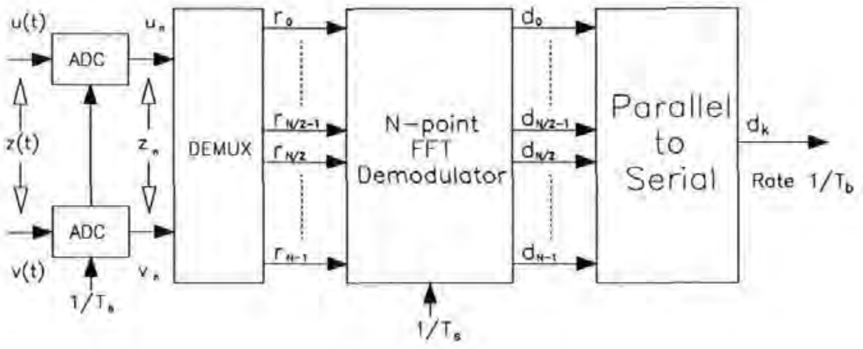
Since $z(nT_b)$ represents a sampled signal, its spectrum repeats at integer multiples of the sampling frequency $1/T_b$ Hz. The demultiplexed sample sequence r_n may therefore be more clearly expressed

$$r_n = z_n = \sum_{k=0}^{N-1} c_k \cdot e^{j\frac{2\pi kn}{N}} \qquad n = 0,1,\ldots,N-1 \qquad (4\text{-}6)$$

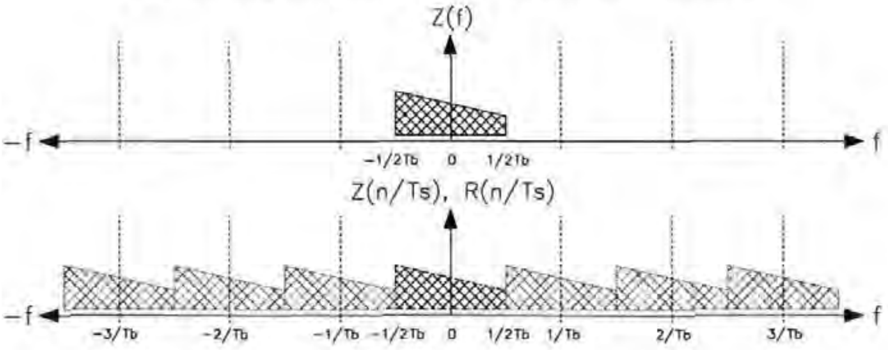
It becomes apparent that an FFT of sequence r_n yields the required coefficients ie. neglecting scaling,

$$d_k = c_k = \sum_{n=0}^{N-1} r_n \cdot e^{-j\frac{2\pi kn}{N}} \qquad k = 0,1,\ldots,N-1 \qquad (4\text{-}7)$$

and the original data sequence ($1/T_b$ Hz) is restored with a parallel to serial converter.



a. Complex sampling QPSK-OFDM demodulator.



b. QPSK-OFDM demodulator baseband spectra.

Figure 4-3. Baseband QPSK-OFDM demodulator and spectra.

For frequency and timing synchronisation investigations, the analogue components of the system were omitted. These investigations were conducted by generating the baseband modulator sample sequence s_n and adding both frequency error and timing error to produce the baseband demodulator input sample sequence r_n directly. Frequency errors were added with a frequency shift operation and timing errors were added using a padded FFT to increase the sample rate.

4.3 Carrier Frequency Synchronisation

In practice, a fixed carrier frequency error will be introduced if the up and down converters are not perfectly matched (see Figure 4-2), and a slowly changing carrier frequency error arises due to satellite motion. For a normalised instantaneous carrier frequency error Δf_c, the demodulator input sample sequence r_n is given by

$$r_n = \sum_{k=0}^{N-1} c_k \cdot e^{j\frac{2\pi(k+\Delta f_c)\cdot n}{N}} \qquad n = 0,1,\ldots,N-1 \qquad (4\text{-}8)$$

If a suitable preamble signal is transmitted, carrier frequency error may be measured directly at the FFT demodulator output, and a compensating frequency shift, f_o, introduced. Figure 4-4 shows simultaneous demodulation and carrier frequency correction using an Offset-FFT (OFFT) [36] demodulator. The OFFT modifies the FFT's twiddle factors to give a normalised frequency offset f_o to its output bins. By performing demodulation and frequency correction simultaneously, the OFFT provides a significant reduction to processing overheads. The demodulated symbols d_k at the OFFT demodulator output are given by

$$d_k = \sum_{n=0}^{N-1} r_n \cdot e^{-j\frac{2\pi(k+f_o)\cdot n}{N}} \qquad k = 0,1,\ldots,N-1 \qquad (4\text{-}9)$$

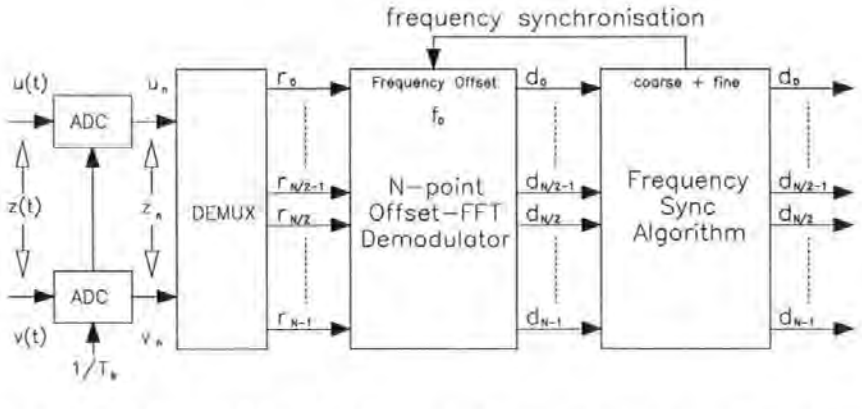


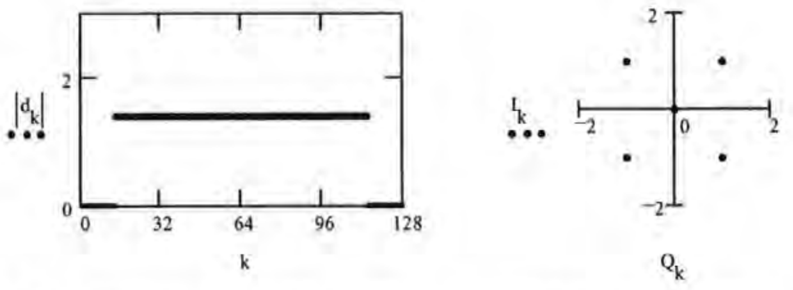
Figure 4-4. QPSK-OFDM demodulator with carrier frequency synchronisation.

It is clear from (4-8) and (4-9) that the frequency error Δf_e is completely eliminated when f_o and Δf_e are equal. However, it should be emphasised that the OFFT will not eliminate the linearly increasing phase error which is associated with the frequency error Δf_e; it is assumed that either differential coding or phase tracking techniques are employed to combat the effects of the linearly increasing phase error. The benefits of this approach are that both demodulation and frequency correction may be conducted simultaneously by the Offset-FFT to minimise software processing overheads and overall receiver complexity.

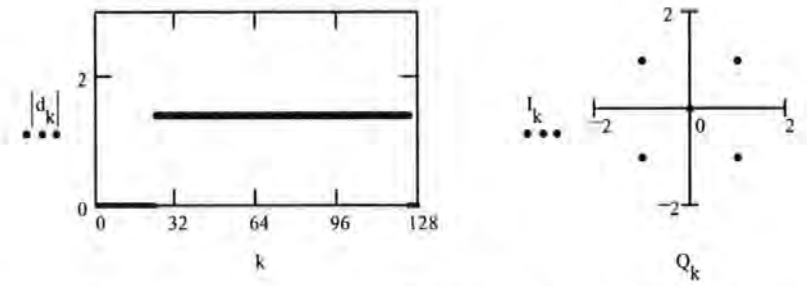
4.3.1 Effects of Frequency Error

A frequency error Δf_e may be expressed $\Delta f_e = \Delta f + \delta f$, where Δf is a coarse frequency error component and δf is a fine frequency error component. The coarse component is equal to an integer multiple of the sub-channel spacing $1/T_s$ Hz, while the fine component is restricted to $0 \leq |\delta f| \leq 0.5/T_s$ Hz. In OFDM systems it is common to introduce guard bands at the upper and lower edges of the transmitted spectrum to allow for the roll-off characteristics of analogue filters. For the following discussions upper and lower guard bands of 13 sub-channels have been used for clarity.

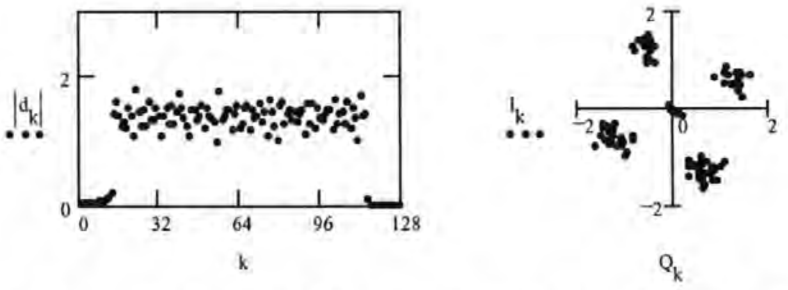
Figure 4-5 demonstrates the effect of coarse and fine frequency error on the demodulator outputs d_k for random data in an QPSK-OFDM system. In the ideal case, Figure 4-5a, the demodulator outputs are orthogonal, have equal magnitude and exhibit four distinct symbol phases. In Figure 4-5b, the effect of a coarse carrier frequency error $\Delta f_e = 10/T_s$ Hz is to shift the demodulator outputs cyclically by 10 frequency bins. A coarse frequency error is clearly not too destructive since orthogonality is maintained and correction is trivial; i.e. the outputs are read with a matching cyclic shift. In Figure 4-5c, a fine frequency error $\Delta f_e = 0.1/T_s$ Hz causes loss of orthogonality and creates inter-channel interference (ICI). As a result, the symbols phases and amplitudes are no longer well defined. As the fine frequency error component increases towards $0.5/T_s$, demodulation rapidly becomes impossible.



a) No frequency error ($\Delta f_e = 0$ Hz) - demodulation successful.



b) Coarse frequency error ($\Delta f_e = 10/T_s$ Hz) - orthogonality maintained.



c) Fine frequency error ($\Delta f_e = 0.1/T_s$ Hz) - orthongality destroyed.

Figure 4-5. The effect of coarse and fine frequency error on demodulator outputs, $d_k = I_k + jQ_k$, in terms of magnitude and phase (N=128).

4.3.2 Combined Coarse and Fine Frequency Synchronisation

From Figure 4-5 it is clear that the effect of fine frequency error δf is more destructive than that of the coarse error Δf. The carrier frequency synchronisation algorithm depicted in Figure 4-6 is therefore designed to first eliminate Δf so that the fine frequency error δf can be accurately measured. To this end, a simple preamble is transmitted by applying the input $c_{N/2} = 1+j$, $c_k = 0$ $k \neq N/2$. This corresponds to a constant signal for several symbol periods at just one of the modulator inputs (see Table 4-1) and results in a single unmodulated subcarrier at a preamble frequency f_p Hz. Since there is no modulation the receiver does not require symbol timing information; symbol timing synchronisation is conducted only after carrier frequency synchronisation has been conducted.

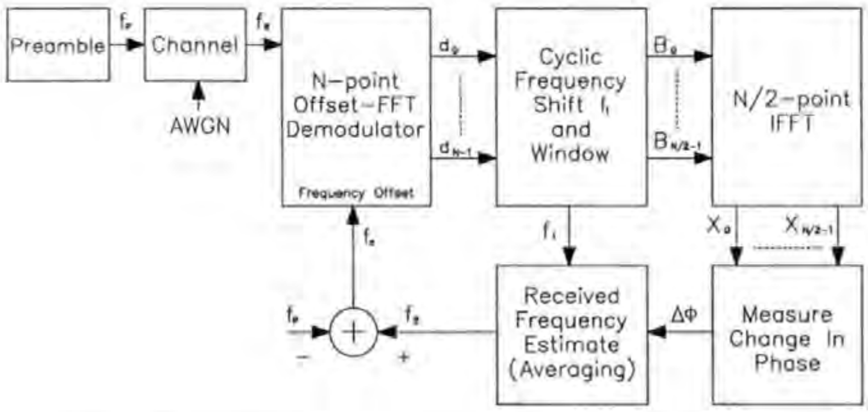


Figure 4-6. Combined carrier frequency error estimation algorithm (residual error f_R-f_2).

Referring to Figure 4-6, the OFFT demodulator's frequency offset f_o is initially zero and by observing the demodulator output d_k with maximum magnitude a coarse carrier frequency estimate f_1 (which corresponds to Δf) is easily obtained; with resolution limited to $\pm 0.50/T_s$ Hz. Figure 4-7 shows the effect of frequency error $\Delta f_e = 10.125/T_s$ Hz on the outputs of a 128-point FFT demodulator with preamble frequency $f_p = 64/T_s$ Hz. The effect of the coarse component $\Delta f = 10/T_s$ Hz is to cyclically shift the peak output from bin d_{64} to bin d_{74} which results in a coarse frequency estimate $f_1 = 74/T_s$ Hz.

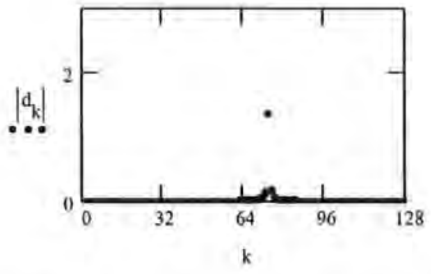


Figure 4-7. Effect of frequency error $\Delta f_e = 10.125/T_s$ Hz on demodulator outputs d_k in terms of magnitude (N=128, $f_P = 64/T_s$ Hz).

The residual frequency error Δf_l is defined as the difference between the received preamble frequency f_R and estimate preamble frequency f_l and is therefore an approximation of the fine frequency error component δf. Once f_l has been measured, a cyclic shift equivalent to -f_l Hz is applied to the demodulator outputs so that only Δf_l remains. Figure 4-8 shows the demodulator outputs Figure 4-7 after a cyclic shift of -f_l (-74/T_s Hz) has been applied. Since the frequency error $\Delta f_e = 10.125/T_s$ Hz and the preamble frequency $f_p = 64/T_s$ Hz, the demodulator outputs after the cyclic shift (B_k) represent a residual frequency error $\Delta f_l = 0.125$ Hz.

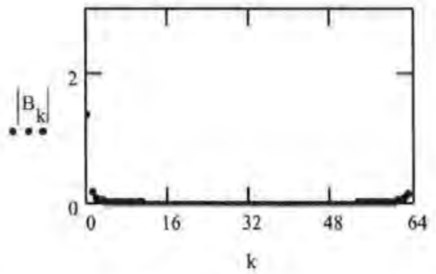
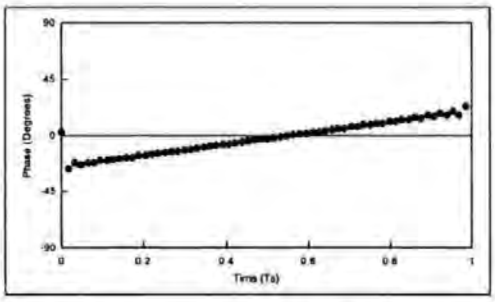
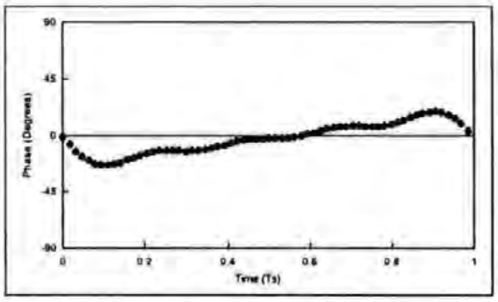


Figure 4-8. Demodulator outputs B_k in terms of magnitude after -f_l Hz cyclic shift to eliminate coarse frequency error component (N = 128, $\Delta f_l = 0.125/T_s$ Hz).

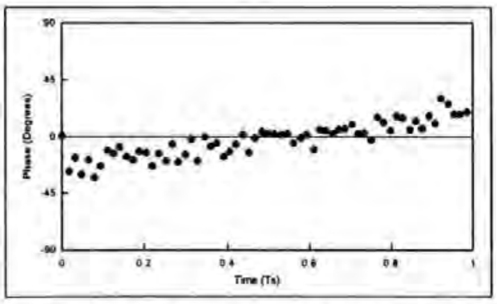
To determine Δf_l with any accuracy, the signal in Figure 4-8 needs to be transformed back to the time domain using a complex input IFFT so that the phase change over the symbol period arising from Δf_l can be accurately measured. The frequency error Δf_l can be estimated directly in the frequency domain from the FFT outputs but the accuracy is not good due to noise. For computation efficiency it is advantageous to use an IFFT of smaller size than that of the demodulator OFFT; a larger IFFT simply provides interpolation in the time domain with no significant benefits. At the IFFT output, complex samples $X_n = a_n + jb_n$ correspond to instantaneous phases ϕ_n, where $\phi_n = \arg(b_n/a_n)$. The samples X_0 to $X_{N/2-1}$ span one symbol period T_s, and so ϕ_0 to $\phi_{N/2-1}$ indicate the phase change $\Delta\phi$ due to the residual frequency error Δf_1. The effects of additive white gaussian noise (AWGN) on the system must also be considered. By applying a rectangular windowing function in the frequency domain, centred at B_0, the carrier to noise ratio (CNR) may be increased, thus reducing the effect of noise on the final carrier frequency estimate f_2. This improvement must be traded against distortion introduced in the time domain due to convolution with the windowing function. Figure 4-9 shows the instantaneous phase over the symbol period for a 64-point IFFT and a selection of rectangular window length and CNR combinations.



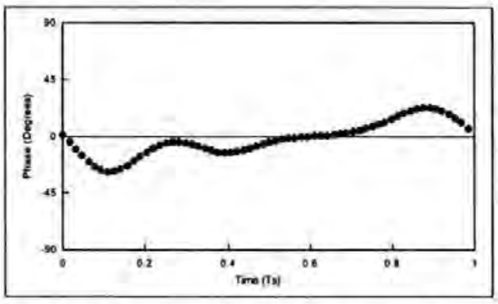
a) Window length = 63, no noise.



b) Window length = 9, no noise.



c) Window length = 63, CNR = 12dB.



d) Window length = 9, CNR = 12dB.

Figure 4-9. Signal phase ϕ_n over OFDM symbol period T_s arising from residual frequency error $\Delta f_1 = -0.125/T_s$ Hz (IFFT length = N/2, CNR measured at demodulator input).

By comparing Figure 4-9a and Figure 4-9b it can be seen that a larger window introduces little distortion while a smaller window introduces noticeable ripple at high CNR. Conversely, by comparing Figure 4-9c and Figure 4-9d, it can be seen that a smaller window reduces noise variance at low CNR. In each case it can be seen that the residual frequency error $\Delta f_1 = 0.125/T_s$ Hz results in a phase change of 45 degrees over the symbol period.

The phase change over the symbol period $\Delta\phi$ is determined using linear regression and several of the transient phase samples at the start and end of the symbol period should be ignored in order to improve performance with smaller window sizes. Once $\Delta\phi$ has been determined, the residual frequency error Δf_1 is computed using

$$\Delta f_1 = \frac{\Delta\phi}{2\pi \cdot T_s} \tag{4-10}$$

and an estimate f_2 of the received preamble frequency is given by

$$f_2 = f_1 + \Delta f_1 \tag{4-11}$$

The frequency offset f_o applied to the OFFT for carrier frequency synchronisation is found by comparing the transmitted preamble frequency f_P with the estimated received preamble frequency f_2

$$f_o = f_2 - f_P \tag{4-12}$$

The frequency offset f_o is therefore positive when the received preamble frequency is too high and negative when the received preamble frequency is too low. The algorithm may be executed once for instantaneous frequency synchronisation or results may be averaged over several iterations for improved performance.

4.3.3 Simulated Carrier Frequency Synchronisation Performance

The carrier frequency synchronisation algorithm in Figure 4-6 was simulated for N = 128 and the results assessed in terms of the rms value of the final residual frequency error $f_R - f_2$. It was first necessary to determine the effect of input window size w, and this is shown in Figure 4-10 for the worst case residual frequency error $\Delta f_I = 0.5/T_s$ Hz. i.e. one half bin error. Here the rms error has been measured after one iteration of the frequency synchronisation algorithm and has been repeated 2000 times for each point plotted. Two interesting observations can be made from Figure 4-10. First, it is apparent that the algorithm fails as the window size is increased beyond a certain threshold, for example, at 3dB the algorithm fails for w>27. Secondly, w = 5 is optimum at low CNR whilst w = 15 is a better choice at high CNR. In summary, a large window introduces less distortion whilst a smaller window reduces the noise variance. This suggests that for optimum performance the algorithm should measure and adapt to the instantaneous CNR.

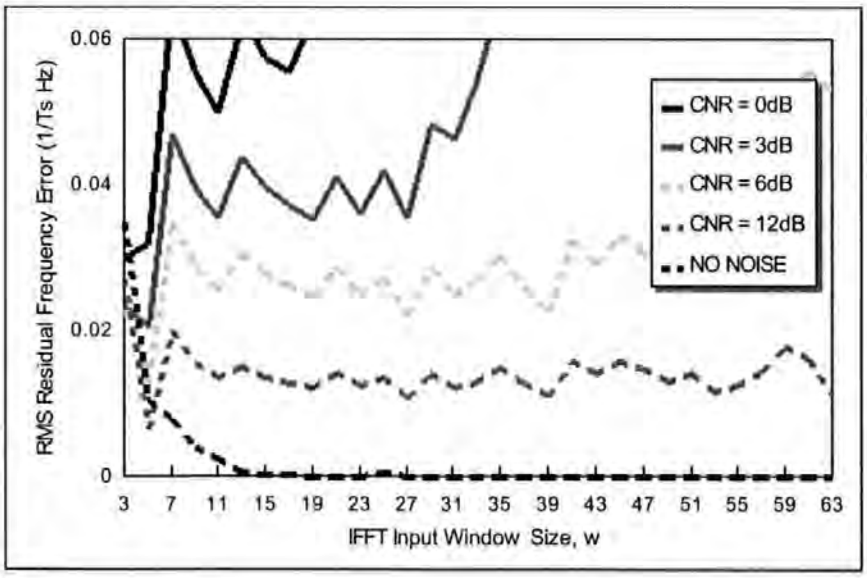


Figure 4-10. Effect of IFFT window size ($\Delta f_I = 0.5/T_s$ Hz).

Figure 4-11 shows simulated performance for a fixed window size w = 5 and with frequency synchronisation conducted over 8 iterations of the algorithm (f_2 averaged over 8 OFDM symbols). The residual error at 0dB is about 1% of the subchannel spacing (and increases to 4% without averaging). Performance could be improved by allowing the preamble to utilise some or all of the power normally allocated to the other (N-1) carriers thus artificially raising the CNR for the duration of the preamble.

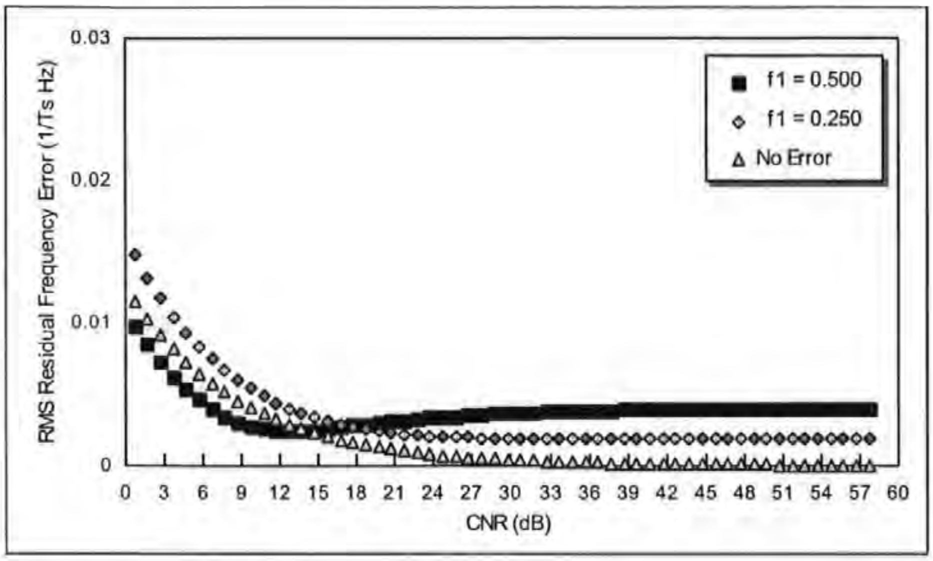


Figure 4-11. Carrier frequency synchronisation error (N=128, w=5, f_2 averaged over 8 iterations).

The carrier synchronisation algorithm can be summarised as follows:

1. Find maximum demodulator output giving coarse frequency estimate f_1.
2. Introduce frequency shift f_1 and window in the frequency domain.
3. Perform a complex input N/2-point IFFT.
4. Determine $\Delta\phi$ by linear regression, and hence Δf_1.
5. Compute $f_2 = f_1 + \Delta f_1$ and hence the offset f_o required for the OFFT.

Steps 3 and 4 respectively are the most computation intensive, and significant savings can be made by using an IFFT of the minimum size required to achieve the desired carrier frequency resolution. However, the carrier frequency synchronisation algorithm will require fewer DSP instructions than the demodulator FFT, providing that the IFFT length is less than that of the demodulator FFT.

4.4 *OFDM Symbol Timing Synchronisation*

The continuous time signal applied to the OFDM demodulator consists of a series of concatenated OFDM symbols of period T_s, and the demodulator samples each symbol N times at rate $1/T_b$ seconds, Figure 4-12. In practice the demodulator input capture period will be incorrectly synchronised with respect to the incoming symbol by ΔT_s seconds, in the worst case ($\Delta T_s = T_s/2$) equal contributions from adjacent symbols are simultaneously applied to the demodulator. An objective in timing synchronisation is therefore to enable the demultiplexer in Figure 4-12 to correctly identify the start of each new OFDM symbol.

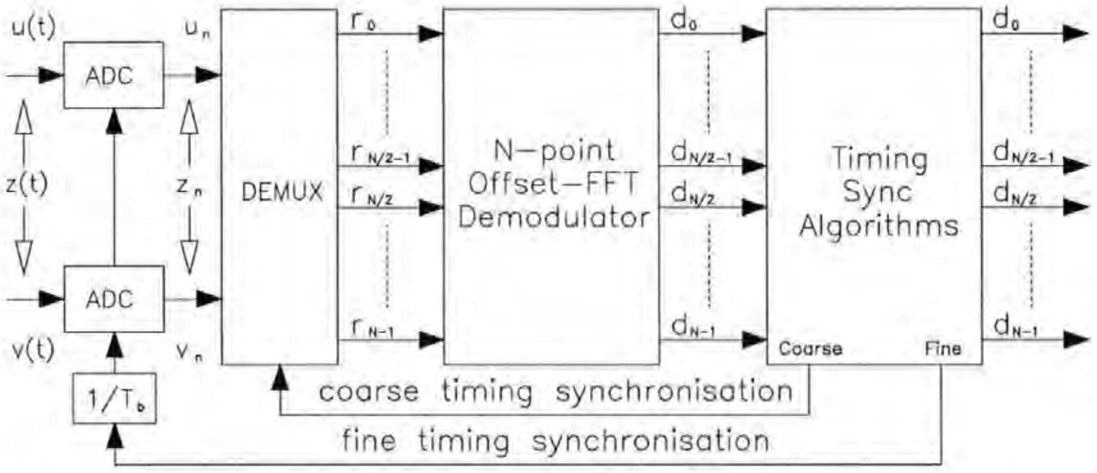


Figure 4-12. OFDM demodulator with symbol timing synchronisation.

The assumption is first made that the sampling clocks in both modulator and demodulator are derived from high precision crystal sources and therefore have the same period T_b seconds. Next it is assumed that the total timing error is the sum of coarse and fine components $\Delta T_s = \Delta T + \delta t$ so that, as for carrier frequency synchronisation, symbol timing synchronisation may be achieved in two stages. Hence $|\Delta T| = kT_b$, k integer and $|\delta t| < T_b/2$, where the latter corresponds to a fixed phase offset in the sampling clock source with respect to the optimum sampling instant. The coarse synchronisation algorithm requires a special preamble sequence (Table 4-1) with a maximum frequency component at half the symbol rate (1+j -1-j 1+j -1-j ...) and this is used to produce an estimate t_c of ΔT. Coarse correction is then made by re-synchronising the demultiplexer, as indicated in Figure 4-12. Coarse timing correction leaves a residual error $\delta_c = \Delta T_s - t_c$, where $|\delta_c| < T_b/2$ under noise-free conditions. The second algorithm produces a fine timing estimate t_f of δ_c to leave an overall residual timing error $\delta_f = \Delta T_s - t_c - t_f$. The fine timing correction is achieved with a small adjustment to the sampling clock phase and, providing δ_f is sufficiently small, demodulation can take place without error.

4.4.1 Analysis of Symbol Timing Error

The effects of OFDM symbol timing error can be explored analytically with a continuous-time model. The model consists of 3 consecutive OFDM symbols where $s_1(t)$ represents the continuous time baseband signal for the OFDM symbol currently being demodulated, $s_0(t)$ for the preceding OFDM symbol and $s_2(t)$ for the following OFDM symbol, Figure 4-13.

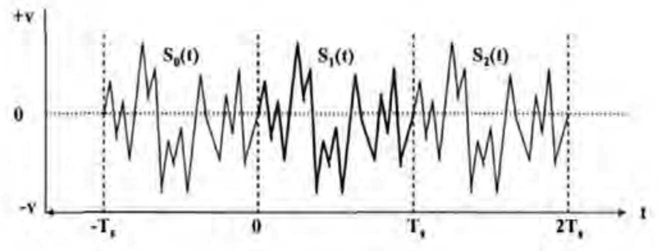


Figure 4-13. Continuous time signal for three consecutive OFDM symbols.

From Figure 4-13, the transmitted baseband signal s(t) is given by

$$s(t) = \begin{cases} s_0(t) & -T_s \le t < 0 \\ s_1(t) & 0 \le t < T_s \\ s_2(t) & T_s \le t < 2T_s \\ 0 & otherwise \end{cases} \tag{4-13}$$

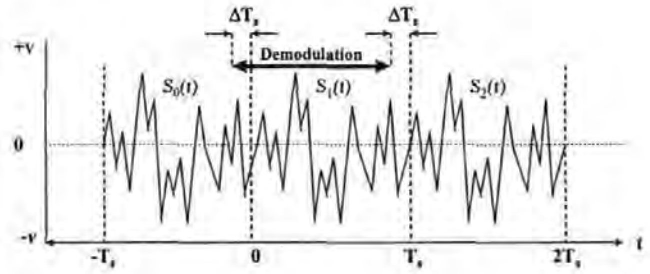
and from (4-2)

$$s_m(t) = \sum_{n=0}^{N-1} c_{n+mN} \cdot e^{j\frac{2\pi nt}{T_s}} \qquad m = 0, 1, 2 \tag{4-14}$$

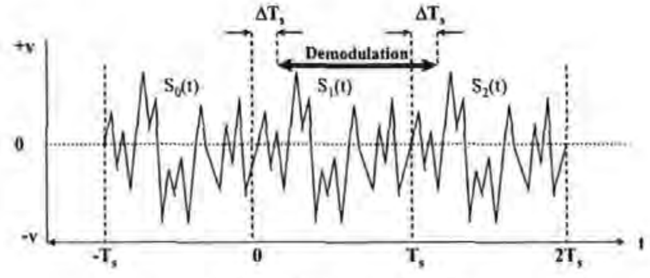
If the baseband signal at the demodulator input is $s(t+\Delta T_s)$, then demodulation over the period $0 \le t < T_s$ gives the output

$$d_k = \frac{1}{T_s} \cdot \int_0^{T_s} s(t+\Delta T_s) \cdot e^{-j\frac{2\pi kt}{T_s}} dt \qquad k = 0, 1, \ldots, N-1 \qquad (4\text{-}15)$$

Figure 4-14 illustrates both negative and positive timing errors at the demodulator with respect to the transmitted baseband signal. When the symbol timing error ΔT_s is negative, Figure 4-14a, demodulation begins early with respect to the transmitted signal and a contribution from the preceding symbol (S_0) is also applied to the demodulator. When the symbol timing error ΔT_s is positive, Figure 4-14b, demodulation begins late with respect to the transmitted signal and a contribution from the following symbol (S_2) is also applied to the demodulator.



a. $\Delta T_s < 0$ - Demodulation early.



b. $\Delta T_s > 0$ - Demodulation late.

Figure 4-14. Demodulator symbol timing error with respect to transmitted OFDM signal.

Using (4-13) and (4-15) it follows that, for the middle symbol s_1, the demodulator outputs d_k are given by

$$d_k = \begin{cases} \dfrac{1}{T_s}\cdot\left[\int_0^{-\Delta T_s} s_0(t+\Delta T_s)\cdot e^{-j\frac{2\pi k t}{T_s}}dt + \int_{-\Delta T_s}^{T_s} s_1(t+\Delta T_s)\cdot e^{-j\frac{2\pi k t}{T_s}}dt\right] & \Delta T_s \le 0 \\ \dfrac{1}{T_s}\cdot\left[\int_0^{T_s-\Delta T_s} s_1(t+\Delta T_s)\cdot e^{-j\frac{2\pi k t}{T_s}}dt + \int_{T_s-\Delta T_s}^{T_s} s_2(t+\Delta T_s)\cdot e^{-j\frac{2\pi k t}{T_s}}dt\right] & \Delta T_s \ge 0 \end{cases} \tag{4-16}$$

and letting u = t + ΔT_s gives

$$d_k = \begin{cases} \dfrac{1}{T_s}\cdot\left[\int_{\Delta T_s}^{0} s_0(u)\cdot e^{-j\frac{2\pi k(u-\Delta T_s)}{T_s}}du + \int_0^{T_s+\Delta T_s} s_1(u)\cdot e^{-j\frac{2\pi k(u-\Delta T_s)}{T_s}}du\right] & \Delta T_s \le 0 \\ \dfrac{1}{T_s}\cdot\left[\int_{\Delta T_s}^{T_s} s_1(u)\cdot e^{-j\frac{2\pi k\cdot(u-\Delta T_s)}{T_s}}du + \int_{T_s}^{T_s+\Delta T_s} s_2(u)\cdot e^{-j\frac{2\pi k\cdot(u-\Delta T_s)}{T_s}}dt\right] & \Delta T_s \ge 0 \end{cases} \tag{4-17}$$

After substituting for $s_0(u)$, $s_1(u)$ and $s_2(u)$, integrating and further simplification the expression becomes

$$d_k = \begin{cases} \dfrac{e^{j\cdot\frac{2\pi(k\cdot\Delta T_s)}{T_s}}}{T_s}\cdot\sum_{n=0}^{N-1}\left[c_n\cdot F(n,k,\Delta T_s)+c_{n+N}\cdot G(n,k,\Delta T_s)\right] & \Delta T_s \le 0 \\ \dfrac{e^{j\cdot\frac{2\pi(k\cdot\Delta T_s)}{T_s}}}{T_s}\cdot\sum_{n=0}^{N-1}\left[c_{n+N}\cdot F(n,k,\Delta T_s)+c_{n+2N}\cdot G(n,k,\Delta T_s)\right] & \Delta T_s \ge 0 \end{cases} \tag{4-18}$$

where

$$F(n,k,\Delta T_s) = \begin{cases} \begin{cases} 1-\Delta T_s & if\, \Delta T_s \le 0 \\ -\Delta T_s & otherwise \end{cases} & if\, n=k \\ \dfrac{1-e^{j2\pi(n-k)\Delta T_s}}{j2\pi(n-k)} & if\, n \ne k \end{cases} \tag{4-19}$$

and

$$G(n,k,\Delta T_s) = \begin{cases} \begin{cases} \Delta T_s & if\, \Delta T_s \le 0 \\ 1+\Delta T_s & otherwise \end{cases} & if\, n = k \\ \\ \dfrac{e^{j2\pi(n-k)\Delta T_s} - 1}{j2\pi(n-k)} & if\, n \ne k \end{cases} \tag{4-20}$$

The expression in (4-18) is the product of two terms; an exponential term and a summation term. The exponential term simply adds a phase shift to the demodulated data which is proportional to the product of the subchannel index k and the symbol timing error ΔT_s. The summation terms generate intercarrier interference (ICI) from within the same symbol and also intersymbol interference (ISI) from the adjacent symbol. For optimum timing synchronisation ($\Delta T_s = 0$) the exponential terms is eliminated and the summation term effectively reduces to $d_k = c_k$. It should be noted that the discrete system will differ in detail with respect to the above continuous time analysis except for the case where consecutive OFDM symbols are identical. The symbol timing synchronisation algorithms discussed later exploit both the ISI and ICI and phase shift components of (4-18) .

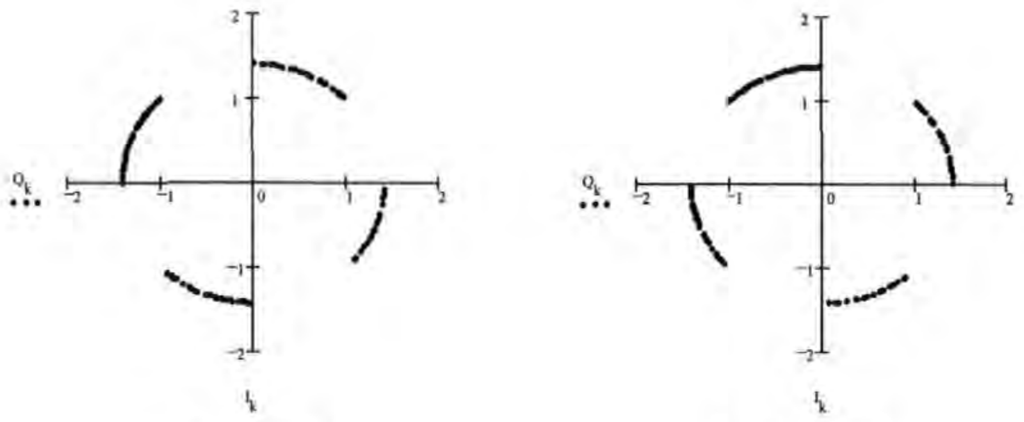
A minimum requirement for symbol timing may be established by examining the exponential term in (4-18) . The phase shift $\Delta\Phi_k$ experienced by demodulator output d_k is given by

$$\Delta\Phi_k = k \cdot \frac{2\pi \cdot \Delta T_s}{NT_b} \qquad k = 0, 1, \ldots, N-1 \tag{4-21}$$

A symbol error is defined as a demodulated QPSK symbol that appears in the wrong quadrant. i.e. a $\pi/4$ phase shift in a noise free environment. From (4-21) it is clear that the greatest phase shift is experienced when k = N-1 and that the following condition must therefore be satisfied

$$\Delta T_s < \frac{N}{(N-1)} \cdot \frac{T_b}{8} \tag{4-22}$$

According to (4-22), symbol timing algorithms must ensure that residual timing error δ_f is below $T_b/8$. The effect of a demodulator timing error $|\Delta T_s| = T_b/8$ is shown in Figure 4-15 for random QPSK symbols. From Figure 4-15 it can be seen that a demodulator timing error $|\Delta T_s| = T_b/8$ represents the threshold at which symbol error begins to occur.



a. $\Delta T_s = +T_b/8$ b. $\Delta T_s = -T_b/8$

Figure 4-15. Phase error in demodulated QPSK symbols $d_k = I_k + jQ_k$ resulting from symbol timing error $|\Delta T_s| = T_b/8$ (N = 128).

4.4.2 Coarse Symbol Timing Synchronisation

In contrast to carrier frequency synchronisation, which employed a single unmodulated carrier, coarse symbol timing modulates a single carrier by a 180° phase reversal (see Table 4-1). For a selected carrier frequency $N/4T_s$ Hz, the m$^{th.}$ symbol in the coarse timing preamble can be written

$$s_m(t) = (-1)^m \cdot (1+j) \cdot e^{j\frac{2\pi N t}{4T_s}} \qquad m = 0,\ 1 \text{ and } 2 \qquad (4\text{-}23)$$

Any timing error will then generate ISI which in turn modulates the amplitude of $d_{N/4}$ at the demodulator output. Here, only $|d_{N/4}|$ is used since the coarse algorithm utilises the ISI component of (4-18) and ignores the phase shift term; ICI is prevented because just one carrier frequency is transmitted. Figure 4-16 shows that ISI from adjacent symbols in the preamble linearly decreases $|d_{N/4}|$ as $|\Delta T_s|$ increases from zero and that complete cancellation occurs for $\Delta T_s = \pm T_s/2$.

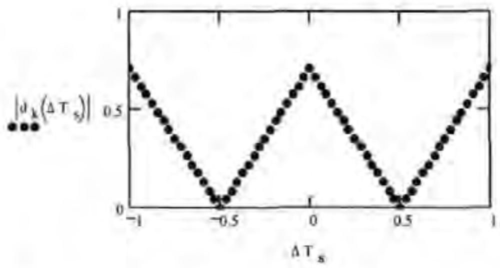


Figure 4-16. $|d_k|$ as a function of ΔT_s during coarse timing synchronisation preamble (T_s=1, k=N/4 and N=32).

The optimum coarse timing correction may be discovered therefore by introducing demodulator timing offsets over the range $t_{offset} = 0, T_b, \ldots\ldots, N\text{-}1 \cdot T_b$ and identifying the offset at which $|d_{N/4}|$ is maximised. Since only a single output bin needs to be computed it is unnecessary, and inefficient, to perform the full FTT. Goetzel's algorithm is an alternative to the FFT algorithm and enables efficient computation of a single FFT output bin [39],[40],[41]. $|d_{N/4}|$ may therefore be found with a modified form of Goetzel's algorithm which includes the frequency offset required for carrier frequency synchronisation. In general for an N-point FFT and normalised frequency offset f_o, the value x of output bin k is given after the N^{th} iteration of the recursive algorithm

$$x_n = C_n + W \cdot x_{n-1} \qquad n = N-1, N-2, \ldots\ldots, 0 \qquad (4\text{-}24)$$

where $W = e^{\frac{-j \cdot 2\pi \cdot (k+f_o)}{N}}$, C_n are the time-domain input samples and x is initially zero.

For improved noise immunity $|d_{N/4}|$ may also be averaged over several symbols before a decision is made. Once the $|d_{N/4}|$ maximum is identified, the optimum coarse timing correction t_c (i.e. the smallest positive or negative adjustment) is given by

$$t_c = \begin{cases} t_{offset} & if\ t_{offset} < \frac{T_s}{2} \\ t_{offset} - T_s & otherwise \end{cases} \qquad (4\text{-}25)$$

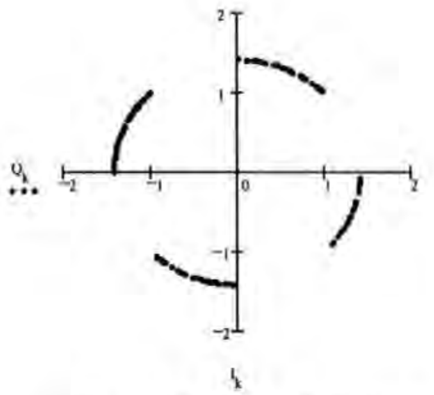
The coarse timing correction is applied to the demultiplexer in Figure 4-12.

4.4.3 Fine Symbol Timing Synchronisation

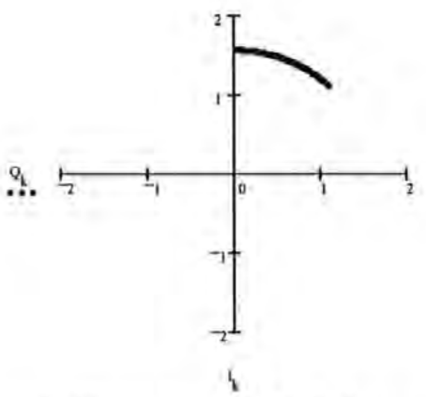
The fine synchronisation preamble exploits the phase shift term in (4-18) while interference terms are suppressed. This is achieved by repeated transmission of the *same* pseudo random OFDM symbol so that the transmitted preamble is continuous across symbol boundaries (see Table 4-1). For this preamble, a detailed examination of (4-18) shows that the effects of the ISI and ICI terms cancel leaving only the phase shift term. Since the N carriers are modulated by random QPSK symbols the first step in the algorithm is to normalise the phase of the demodulated symbols $d_k = I_k + jQ_k$ (Figure 4-17a) according to the following rule;

$$\theta_k = \begin{cases} \tan^{-1}\left(\dfrac{|Q_k|}{|I_k|}\right) & if(I_k \cdot Q_k) \geq 0 \\ \tan^{-1}\left(\dfrac{|I_k|}{|Q_k|}\right) & otherwise \end{cases} \qquad k = 0, 1, \ldots, N-1 \qquad (4\text{-}26)$$

This removes the QPSK modulation and places the phase of all N symbols within the first quadrant, Figure 4-17.



a. Demodulated symbol phases.



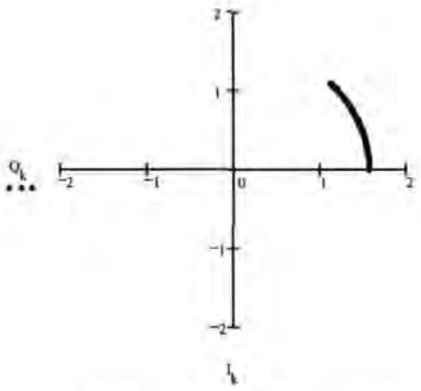
b. Normalised symbol phases.

Figure 4-17. Phase error in demodulated and normalised QPSK symbols $d_k = I_k + jQ_k$ resulting from symbol timing error $\Delta T_s = T_b/8$ (N = 128).

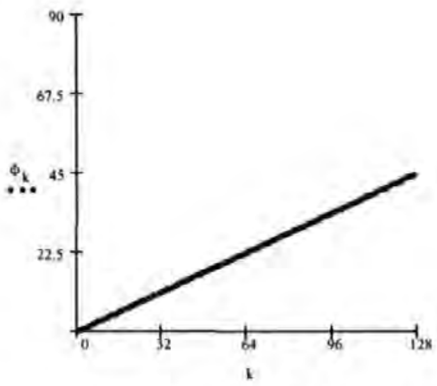
The next step in the algorithm is to eliminate phase ambiguity introduced by the $\tan^{-1}$ operation in (4-26) using

$$\Phi_k = \begin{cases} 0 & \text{if } k = 0 \\ \sum_{m=1}^{k} (\theta_m - \theta_{m-1}) \bmod \left(\frac{\pi}{4}\right) & \text{otherwise} \end{cases} \quad k = 0, 1, \ldots, N-1 \qquad (4\text{-}27)$$

The corrected symbol phases Φ_k are rotated to begin at 0°, Figure 4-18a, and under low noise conditions will be a monotonically increasing function of k, Figure 4-18b.



a. Corrected symbol phases.



b. Symbol phase vs frequency.

Figure 4-18. Phase error in corrected QPSK symbols $d_k = I_k + jQ_k$ resulting from symbol timing error $\Delta T_s = T_b/8$ (N = 128).

In practice, each Φ_k may be averaged over several symbols to minimise the effect of noise. The slope, $\Delta\Phi$, of Φ_k versus k is found using linear regression, and this can be used to accurately measure the residual timing error δ_c after coarse timing correction. Specifically;

$$\delta_c = \frac{\Delta\Phi}{2\pi} \cdot T_s \qquad (4\text{-}28)$$

Note that due to the modulo $\pi/4$ arithmetic in (4-27) , the fine timing synchronisation algorithm described above will fail if $\Delta\Phi > \pi/4$. However, $\Delta\Phi = \pi/4$ corresponds to a relatively large residual timing error of $T_s/8$ after coarse synchronisation and is unlikely to occur in practice.

4.4.4 Simulated Symbol Timing Synchronisation Performance

The coarse and fine symbol timing synchronisation algorithms were simulated for a 128-carrier demodulator in the presence of AWGN, Figure 4-19. Frequency error was not considered in these simulations but initial random timing errors, with resolution $\pm T_b/32$, were introduced by way of a padded IFFT modulator, Figure 4-20. The algorithms were assessed on the residual timing error after synchronisation had been conducted.

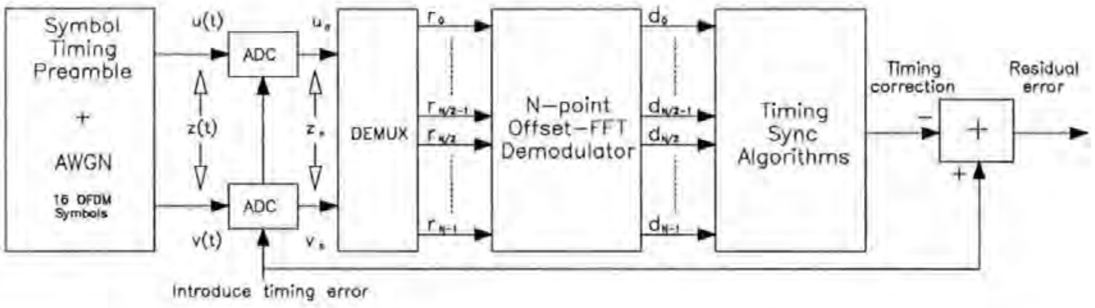


Figure 4-19. Simulated OFDM symbol timing synchronisation system.

The coarse and fine synchronisation fields of the preamble test signal each consisted of 8 OFDM symbols with averaging performed in each algorithm over 6 OFDM symbols for improved noise immunity; 4 additional symbols were added so that random timing errors over the range $-T_s$ to T_s could be easily introduced. A zero padded IFFT was used to simulated band-limited noise in the over-sampled preamble test signal, Figure 4-20.

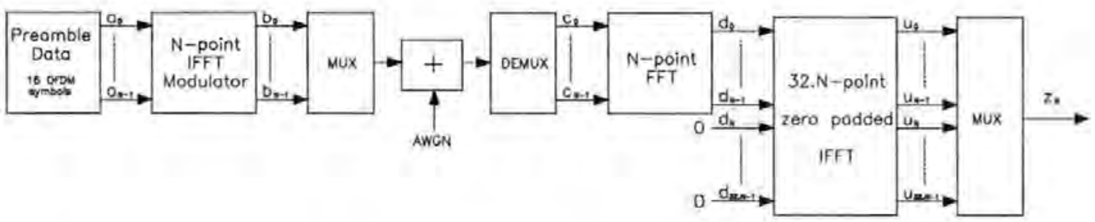
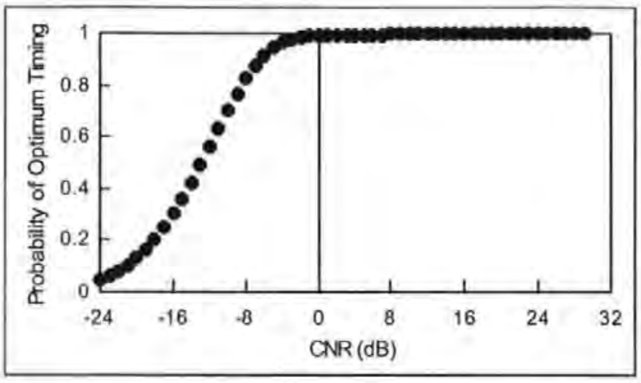
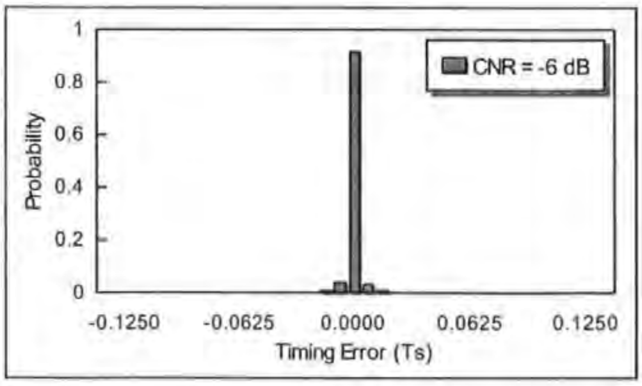


Figure 4-20. Generation of over-sampled OFDM symbol timing test signal.

Figure 4-21 shows performance of the coarse algorithm alone expressed in terms of the probability of optimum symbol timing ($|\delta_c|<T_b/2$) and the residual timing error probability distribution for CNR = -6dB (a point at which the algorithm still performs reasonably well). At CNR $\geq$ 8dB performance is optimum, and below 8dB the algorithm begins to degrade. For CNR = -6dB, the distribution peak indicates a 91% probability of optimum synchronisation, although residual timing errors of up to $\pm 2T_b$ are apparent. However, since the receiver has still to perform fine symbol timing synchronisation, these results are not an indication of overall performance.



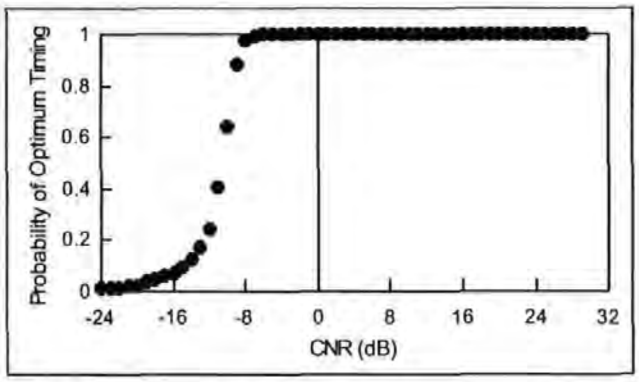
a) CNR verses probability of optimum symbol timing.



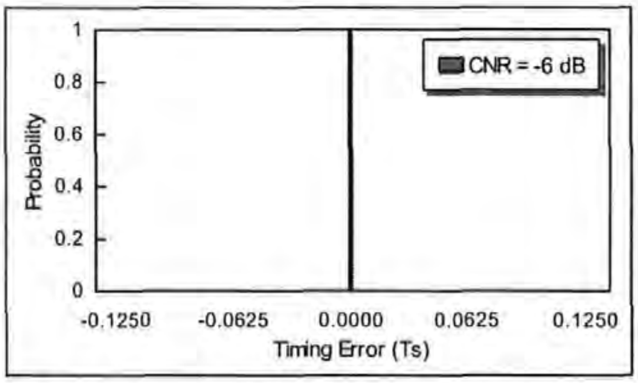
b) Residual timing error distribution at CNR = -6dB, bin size = T_b.

Figure 4-21. Simulated timing synchronisation performance after coarse synchronisation algorithm (N=128).

Figure 4-22 shows overall timing synchronisation performance, after both coarse and fine timing synchronisation, for direct comparison with Figure 4-21. For CNR $\geq$ -6dB, synchronisation has a residual error $|\delta_f| < T_b/32$ which equates to optimum performance. These results indicated that the fine timing synchronisation algorithm can provide optimum performance even when the coarse algorithm performs sub-optimally. The rapid drop in performance below -6dB is the result of high residual timing errors after coarse synchronisation which result in a failure of the modulo $\pi/4$ arithmetic in (4-27) .



a) CNR verses probability of optimum symbol timing.



b) Residual timing error distribution at CNR=-6dB, bin size = $T_b/32$.

Figure 4-22. Simulated timing synchronisation performance after both coarse and fine synchronisation algorithms (N=128).

4.5 Conclusions

In this chapter the author has presented coarse-fine algorithms for carrier frequency and symbol timing synchronisation in coherent QPSK-OFDM. Both carrier frequency and symbol timing errors are measured directly at the demodulator output by transmitting simple preamble sequences. Carrier frequency and coarse symbol timing synchronisation are performed entirely in software while fine timing correction is achieved by adjusting the phase of the demodulator's sampling clock. Simultaneous carrier frequency synchronisation and demodulation can be achieved using the Offset-FFT algorithm with a suitable preamble sequence, and a synchronisation algorithm has been described. For a preamble length of 8 OFDM symbols and CNR≥0dB, the carrier frequency can be synchronised to within 1% of the subchannel spacing and performance could be improved by adapting the algorithm to the target CNR. Symbol timing algorithms have also been demonstrated which reduce residual symbol timing error to insignificant levels at CNR≥-6dB with a preamble length of 12 OFDM symbols. When combined with algorithms for unique word synchronisation, to detect the end of the preamble, and phase tracking to provide phase synchronisation, these algorithms form the basis for a coherent OFDM receiver.

5. Applications in Highly Asymmetric Satellite Internet Delivery Systems

The concept of a data reply link for multimedia satellite applications was introduced in Chapter 1 and frequency and timing synchronisation algorithms for a suitable software-based DSP Burst Demodulator were discussed in Chapter 3. It became apparent during these investigations that the *Satellite Data Reply Link* principle and the DSP Burst Demodulator could be applied to asymmetric satellite Internet delivery systems. Commercial systems, such as Europe On-Line (EOL), exploit a shared high-speed DVB satellite broadcast channel to deliver Internet content at much higher rates than could be offered by PSTN modems, but these systems still use a PSTN modem to provide a request channel. The *Satellite Data Reply Link* hardware provides a satellite-based request channel, and allows high-speed Internet access to be provided anywhere within the footprint of a satellite. Three asymmetric satellite Internet delivery systems are presented in this chapter which represent evolution towards a viable system. At the time of writing, a commercial system based upon these investigations is being operated by Web-Sat Ltd. of Dublin, Ireland.

An overview of the Internet Protocol Suite and some background information relating to Internet addressing and routing is given in Appendix G. Although much information is presented, the author has restricted detail to that which is required to follow the basic Networking discussions in this chapter. Section 5.1 defines the terms 'symmetrical' and 'asymmetric' Internet delivery systems and presents a general model that forms the basis of this chapter. In section 5.2, three asymmetric Internet delivery systems are described. An overview is given for each system, and novel concepts are highlighted; much of this information is contained within the author's publications (Appendix I and Appendix J). For these systems, the author contributed software device drivers to link the satellite modulator and demodulator hardware to the PC Operating System (Internet Protocol Suite) using 'Ethernet emulation'. This software, problems related to uni-directional Ethernet emulation, and other practical considerations are outlined in section 5.3. With respect to IP networking, a satellite link is described as having a high 'delay×bandwidth' product, and this presents a fundamental problem to TCP/IP throughput performance. In section 5.4, the detrimental effects of satellite delays are measured and standard methods proposed by others to overcome these limitations are reviewed with respect to asymmetric satellite Internet delivery systems. In section 5.5, shared request channel performance is measured and two channel access protocols are evaluated, frequency hopping and TDMA (Time Division Multiple Access), with the objective to maximise throughput and minimise collisions. Finally, in section 5.6, conclusions are drawn and topics highlighted for future research.

5.1 Definition of Symmetrical and Asymmetric Internet Delivery Systems

Internet delivery systems may be divided into two main categories; 'symmetrical systems' and 'asymmetrical systems'. For this chapter, the term 'symmetrical system' describes a system where just one network connection provides both the outgoing request channel and incoming response channel, Figure 5-1. The term 'asymmetric system' is used in this chapter to describe a system where separate network connections provide request and response channels, Figure 5-2. It is common for Internet data exchanges to produce more incoming data than outgoing data, and for the data rates of the request and response channel to take advantage of this characteristic; this ratio often exceeds 10:1 for HTTP (Hyper-Text transfer Protocol) and FTP (File Transfer Protocol)[53].

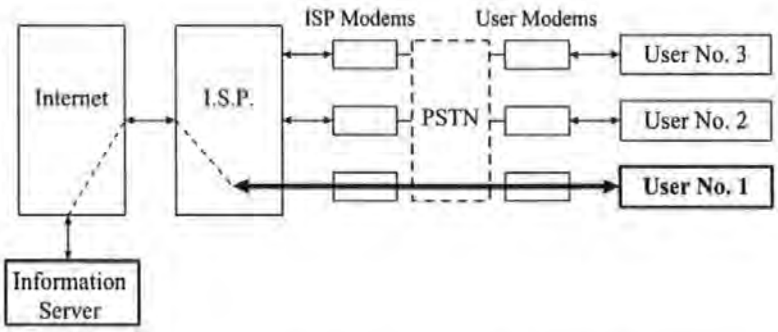


Figure 5-1. Symmetrical Internet delivery system.

An overview of a typical home Internet delivery system is shown in Figure 5-1, this is classified as a 'symmetrical system'. The corresponding 'asymmetric' system is shown in Figure 5-2, where outgoing requests are also sent via the PSTN modem but incoming responses received via a separate (high-speed) response channel.

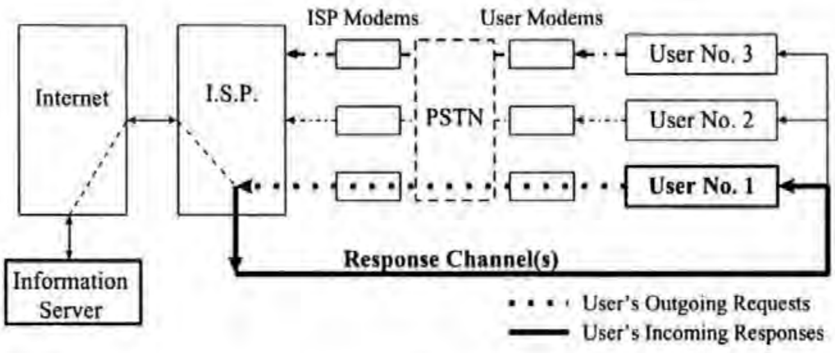


Figure 5-2. Asymmetric Internet delivery system.

For clarity it is assumed that the IP address assigned to each network interface in Figure 5-1 and Figure 5-2 is known in advance, and that direct routing is employed throughout; no Proxy Server or IP address translation is used. For 'User 1' to communicate with the 'Information Server', a request is sent via the user's PSTN modem and received by the ISP. Based upon the destination IP address (that of the information server), the ISP routes this request onto the Internet where it is delivered to the intended recipient. In the reverse direction, the original source IP address (that of the user's modem) becomes the destination IP address, and responses are delivered back via the Internet to the ISP; the registered owner of the addresses. At this point the two systems differ; in Figure 5-1 responses are delivered back to the user via the modem, in Figure 5-2 they are delivered via a dedicated (high-speed) response channel. Operation is determined by IP routing tables at the ISP, which specify an alternative response path to the user in the asymmetric case. The satellite Internet delivery systems discussed in this chapter follow the 'asymmetric system' model of Figure 5-2. For these systems, the response channel is provided by a high-speed satellite data broadcast link. For interaction, request channels are provided by *Satellite Data Reply Links* (see Chapter 1) and employ DSP Burst Demodulators (see Chapter 3).

5.2 *Novel Asymmetric Satellite Internet Delivery Systems*

The author was involved with three pilot system projects with funding received from BNSC (The British National Space Council), ESA (The European Space Agency) and with collaboration from Eutelsat during satellite trials. In these projects the author was responsible for writing the software drivers which link the DSP modulation and demodulation hardware to the standard PC operating system (Internet Protocol Suite). These drivers, as described in section 5.3, had to incorporate features to compensate for system degradation caused by the satellite communication path.

The first system employs a standard dial-up request channel at 9.6 - 28.8kbps and a modified analogue satellite data broadcast system, known as SatLink[54], to provide a response channel at 90kbps. This system is discussed in the author's publication "The Development of an Operational Satellite Internet Service Provision" in Appendix I. A brief overview of this hybrid terrestrial/satellite system and descriptions of the novel satellite hardware/software can be found in Appendix G. The second system exploits the *Satellite Data Reply Link* concept to provide a low-speed (16kbps burst-mode) satellite request channel. This system also employs a proprietary DVB satellite data broadcast system (based upon Satlink[54] hardware) to provide a response channel at 2Mbps. This system is described in the author's publication "A Novel Internet Delivery System Using Asymmetric Satellite Channels" in Appendix J. A brief overview of this asymmetric satellite system and descriptions of novel satellite hardware/software can also be found in Appendix G. A third system evolved from initial investigations, and is described in section 5.2.1. This 'improved system' includes enhancements which are required to serve a greater number of users and to provide a test platform for evaluating shared channel access protocols, network control and user authentication strategies.

5.2.1 An Improved Asymmetric Satellite Internet Delivery System

The improved system, Figure 5-3, uses the asymmetric model defined in section 5.1. The response channel is provided by commercial DVB hardware compliant with ETSI (European Telecommunications Standards Institute) standards for encapsulation of IP Datagrams within a DVB transport stream[56] and warrants no further description. Request channels are provided by the *Satellite Data Reply Links* (see Chapter 1), and each channel is received by a separate Burst Demodulator (see Chapter 3). Network control features strongly in the improved system, and is shown as an input to the ISP. In addition, Internet access is made by Proxy Server to provide the following advantages;

1. A Proxy server allows Private IP addresses to be assigned to the clients, hence the ISP requires few legitimate IP addresses to serve many (thousands of) users.
2. A Proxy server can store (cache) frequently accessed information and satisfy future requests more efficiently from the local cache.
3. A modified Proxy Server may overcome TCP/IP throughput performance issues associated with the satellite delays (see section 5.4.3).

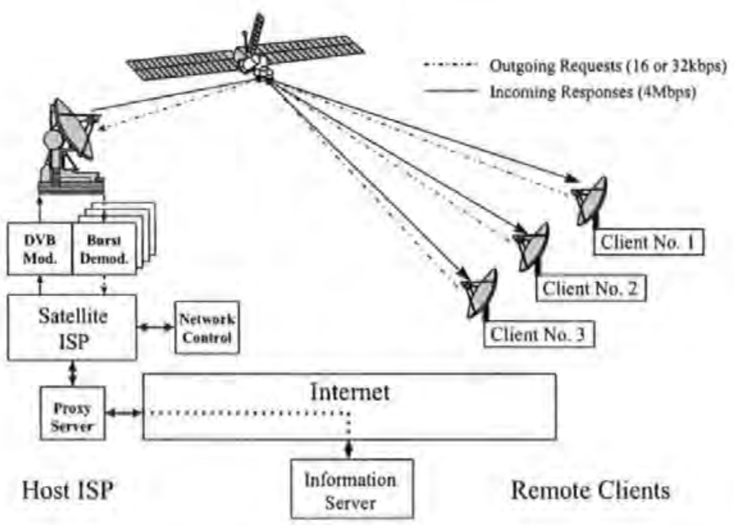


Figure 5-3. Improved asymmetric satellite Internet delivery system - overview.

In terms of a network diagram, Figure 5-4, Client terminals are assigned IP addresses from subnets 172.17.1.0 to 172.17.254.0 of the private class B IP network 172.17.0.0. Requests are received by one of several *RlMux* (Return Link Multiplexer) PCs and are passed to the Proxy Server via a router. The Proxy Server fetches the requested information from the Internet (using legitimate IP addresses) and sends responses back to the client via the router and DVB Gateway. Since all IP Datagrams pass through the router, this provides an opportunity to measure traffic for diagnostic purposes. The DVB Gateway maintains MAC address and IP address associations so that filtering may be conducted (most efficiently) by the DVB receiving hardware; based upon MAC address. Operational system parameters are broadcast directly from the Network Control terminal to Client terminals via the DVB gateway.

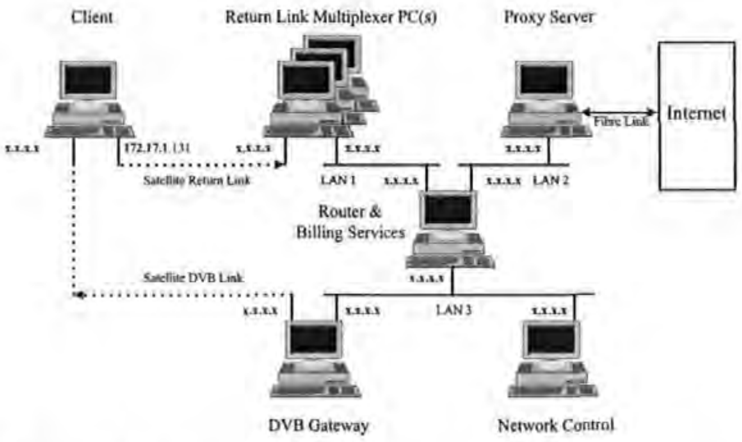


Figure 5-4. Improved asymmetric satellite Internet delivery system - network diagram.

5.3 *Network Device Driver Software for IP Transmission Over a Satellite Data Reply Link*

A network interface controller (NIC), such as an Ethernet network adapter, must be provided with low-level network device driver software. The purpose of a network 'device driver' is to hide proprietary details of the underlying hardware from the operating system (Internet Protocol Suite) by providing a standard software interface. For these systems, the Client PC's NIC is a *Satellite Data Reply Link* transmitter PC interface card and NICs at the ISP are PC cards which interface with the DSP Burst Demodulators. This section outlines the software drivers created by the author to provide Ethernet emulation over the *Satellite Data Reply Link.*

5.3.1 Network Device Driver Overview

Network device driver software for any underlying network technology presents the same software interface to the Internet Protocol Suite; even though adapter hardware can vary considerably. A network device driver provides a software to hardware translation, Figure 5-5. For any operating system, a network device driver is required to provide a number of generic functions in order to 'initialise', 'reset' and 'halt' the adapter hardware. It is also required to provide functions for sending and receiving physical frames which are specific to the underlying network technology. A number of underlying network technologies are supported by most implementations of the Internet Protocol Suite, and these include Ethernet, Token Ring and ATM. When the underlying technology is Ethernet, for example, the Internet Protocol Suite generates Ethernet Frames to send over the physical network connection and expects to receive Ethernet Frames back from the network device driver.

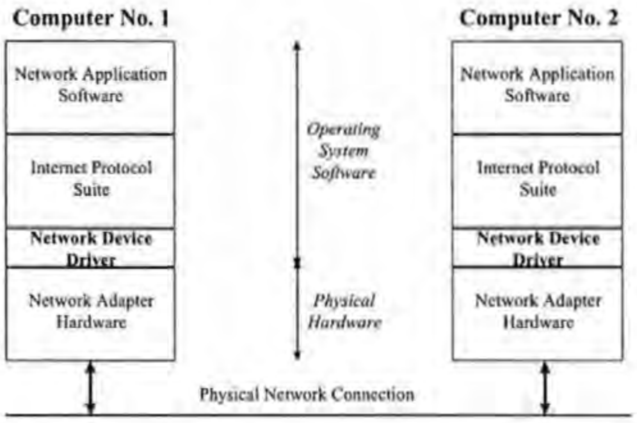


Figure 5-5. Network device driver software.

5.3.2 Ethernet Emulation over the Satellite Data Reply Link

Ethernet emulation was selected because Ethernet is the most popular and widely supported underlying network technology [58]. This emulation strategy allows an experimental network technology such as the *Satellite Data Reply Link* to operate with existing implementations of the Internet Protocol Suite. At its simplest, Ethernet emulation is achieved with device driver software that converts transmitted Ethernet Frames to match the characteristics of the experimental network technology. In this case, the *Satellite Data Reply Link* hardware has no network (MAC) address, while Ethernet uses 6-byte MAC addresses. Conversely, the *Satellite Data Reply Link* packet's 'preamble' and 'checksum' fields duplicate the 'preamble' and 'frame check sequence' fields of the Ethernet frame, Figure 5-6. For Ethernet emulation, the network device driver encapsulates unique fields of the Ethernet Frame within a *Satellite Data Reply Link* packet as the data payload and, upon reception, presents the data payload to the Internet Protocol Suite as though a standard Ethernet frame had been received.

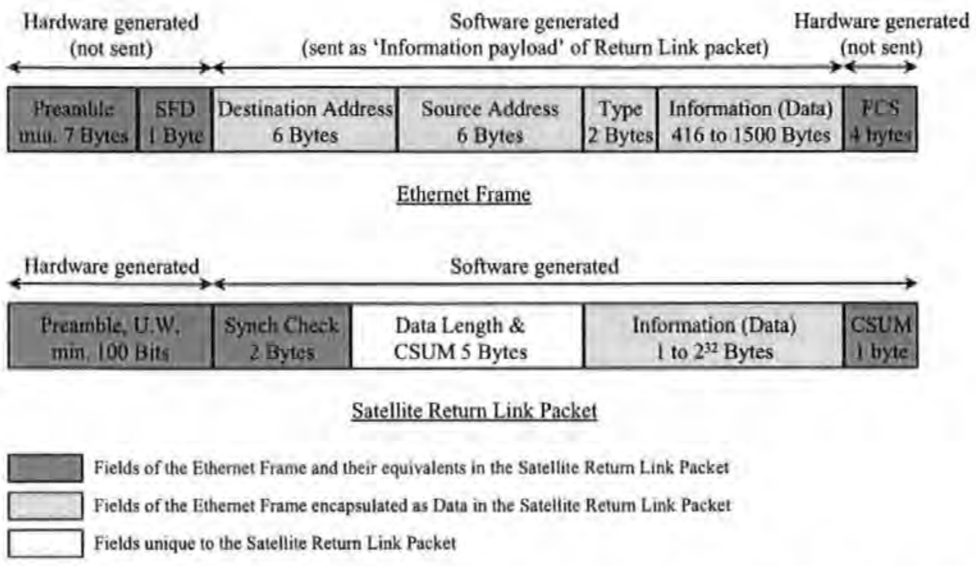

Hardware generated (not sent)
Software generated (sent as 'Information payload' of Return Link packet)
Hardware generated (not sent)
Preamble min. 7 Bytes
SFD 1 Byte
Destination Address 6 Bytes
Source Address 6 Bytes
Type 2 Bytes
Information (Data) 416 to 1500 Bytes
FCS 4 bytes
Ethernet Frame
Hardware generated
Software generated
Preamble, U.W. min. 100 Bits
Synch Check 2 Bytes
Data Length & CSUM 5 Bytes
Information (Data) 1 to 2^32 Bytes
CSUM 1 byte
Satellite Return Link Packet
Fields of the Ethernet Frame and their equivalents in the Satellite Return Link Packet
Fields of the Ethernet Frame encapsulated as Data in the Satellite Return Link Packet
Fields unique to the Satellite Return Link Packet

Figure 5-6. Encapsulation of Ethernet Frames within Satellite Return Link Packets.

5.3.2.1 Ethernet Emulation for Uni-Directional Links

The Ethernet standard provides two-way communication, therefore extra considerations are required to provide Ethernet Emulation over the one-way *Satellite Data Reply Link.* The most fundamental issue arises with a 'transmit only' NIC due to the Address Resolution Protocol (ARP). ARP allows a sending terminal to dynamically discover a MAC address corresponding to the IP address with which communication is desired. With ARP, an *ARP Request* is broadcast over the local network inviting the destination terminal to respond with an *ARP Response.* If no *ARP Response* is received, as is the case with a 'transmit only' NIC, the Internet Protocol Suite will not send any further traffic because there appears to be no destination terminal. ARP spoofing was built into all network device drivers for this series of investigations. ARP spoofing requires the network device driver to test the 'protocol type' field in the header of each outgoing Ethernet Frame and, upon detection of an *ARP Request*, to construct a 'spoofed' *ARP Response.* The 'spoofed' *ARP Response* is passed back to the Internet Protocol Suite, as though it was received from the network, and the original *ARP Request* is discarded. Upon receipt of a 'spoofed' *ARP Response*, the Internet Protocol Suite begins to send Ethernet Frames with a randomly generated 'destination MAC address' field.

In the case of a 'receive only' NIC, the issues are slightly different. Although transmission is not possible, the network device driver must still indicate to the Internet Protocol Suite that each request to send was completed successfully. If this does not occur, it may be falsely concluded that the NIC has failed. In turn, the NIC hardware may be reset in an attempt to clear the fault, and incoming packets will be lost as a result. A second consideration is incoming Ethernet Frames received from a 'transmit only' NIC; where ARP spoofing has been employed. It is likely that the 'destination MAC address' in these Frames was randomly generated and will not match that of the receiving NIC. To prevent the Internet Protocol Suite from rejecting these Frames unnecessarily, the destination MAC address must be modified by the network device driver to match the receiving NIC. A more efficient solution is to remove the destination MAC address field (6 bytes) altogether upon transmission and to re-insert a suitable MAC address upon reception; this strategy was adopted to make more efficient use of the limited *Satellite Data Reply Link* capacity.

5.3.3 Packet Filtering and Prioritising

It has been described how Ethernet Frames are encapsulated within *Satellite Data Reply Link* packets. An Ethernet Frame encapsulates an IP Datagram which in turn encapsulates TCP Segments or UDP (User Datagram Protocol) Datagrams. To identify individual applications, TCP and UDP also use 16-bit 'port numbers' in their respective headers. Details of the Ethernet Frame, IP Datagram, TCP Segment and UDP Datagram structures are given in Appendix G. The packet filtering and prioritising algorithms discussed in later sections were performed at the device driver level by examining the contents of each outgoing or incoming Ethernet Frame. For filtering, rules are set within the device driver to specify which Protocols and Port Numbers should be blocked. For prioritising, rules are set within the device driver to assign a priority to each Protocol or Port Number.

5.3.4 User Authorisation and Distribution of System Parameters

During early investigations, parameters such as request channel transmit frequencies and the response channel receive frequencies were pre-configured for each terminal. In practice, any satellite operator will insist that a user terminal cannot transmit until it has first been authorised to do so. A major reason for this requirement is to prevent interference to adjacent satellites in the event that the transmit antenna becomes mis-aligned. In keeping with this requirement, the systems approached user authorisation with a passive strategy; ie. broadcasts are made from the ISP on a cyclic basis containing details of users who are permitted to use the system. Operational system parameters, such as transmit frequencies and filtering/prioritising algorithm rules, are also broadcast on a cyclic basis so that the system may be dynamically reconfigured from a central location. In this case, it is only necessary to pre-configure the frequency at which the Terminal initially 'listens'.

5.4 Improving Throughput in Satellite Internet Delivery Systems

The Internet 'Request For Comments' (RFC) are maintained by the Internet Engineering Task Force (IETF) [55] and provide an electronic forum for researchers to present new ideas and to exchange experimental results. From the RFCs, Internet standards are produced which influence future implementations of the Internet Protocol Suite. In 1972, J. Postel issued RFC 346 suggesting that research should be conducted into the effect of satellite links on "servers with character-at-a-time remote echo" [59]. This topic has since been revisited by others with reference to TCP/IP over satellite links. A satellite link is generally considered to have a high 'delay×bandwidth' product, which is acknowledged as a fundamental problem with respect to TCP/IP throughput performance [57] [63]. In section 5.4.1, the detrimental effect of satellite delays on TCP/IP performance are measured. Several modifications to TCP have been proposed in order to overcome these limitations. In section 5.4.2, the most significant mechanisms from the RFC literature are summarised with respect to highly asymmetric satellite Internet systems. In section 5.4.3, an alternative strategy is described where UDP is used in place of TCP. It is the author's opinion that the latter strategy is best suited to optimisation of the highly asymmetric satellite Internet delivery systems under discussion.

5.4.1 The Effect of Satellite Delay on TCP/IP Throughput

Practical measurements of TCP/IP throughput were made with the asymmetric satellite Internet delivery system in Figure 5-3. For these tests, the request channel provided 16kbps and the response channel 1Mbps. Average TCP/IP throughput was computed by measuring the time taken to download a 6Mbyte file from a local server with the file transfer protocol (FTP). Each time, the test was conducted first in a back to back configuration (eliminating the satellite links) and then with the satellite links in place. TCP provides flow control using a 'sliding window' principle (see Appendix G), and it is well known that increasing the 'TCP Acknowledgement Window' field in the TCP Segment header can improve TCP throughput for high delay networks. This is possible because most Internet Protocol Suite implementations are configured for acceptable performance on low delay networks (such as Ethernet), whereas two geostationary satellite links constitute a high delay network. TCP/IP throughput in back to back and satellite tests was measured first for a default acknowledgement window size of 8192 bytes and then with a near maximum acknowledgement window of 61320 bytes, Figure 5-7.

For the default acknowledgement window, a TCP/IP throughput of 40 kBytes per second was measured in back to back tests and a throughput of 11.7 kBytes per second in satellite tests. With the extended acknowledgement window, a throughput of 100 kBytes per second was measured in back to back tests and a throughput of 42 kBytes per second for satellite tests. These results confirm that the satellite delays result in a large reduction to TCP/IP throughput performance. These results also confirm that extending the TCP Acknowledgement Window can improve TCP/IP throughput performance by a factor of 4 over the settings for a low delay network. For a high delay network, the round trip time (RTT) is much longer and the sender must wait longer before each acknowledgement is received. Increasing the acknowledgement window allows more data to be injected into the network between receipt of acknowledgements; hence more efficient use of available capacity.

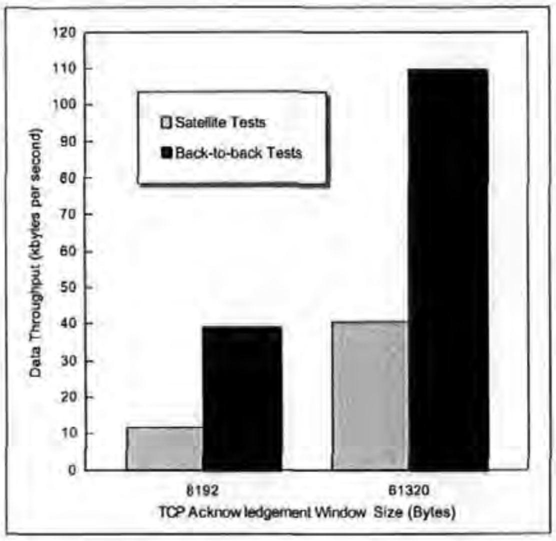


Figure 5-7. Effect of satellite delays (600ms RTT) & TCP acknowledgement window size on TCP/IP throughput performance.

5.4.2 Enhancing Throughput With Standard TCP/IP Mechanisms

In [57], Allman, et. al. summarise the IETF standards and upcoming standards that are considered the Best Current Practice (BCP) for enhancing TCP over satellite channels, In Table 5-1. It can be seen that only the 'Slow Start' and 'Congestion Avoidance' mechanisms are required in current implementations of the Internet Protocol Suite and that the remaining mechanisms are only recommended. It is significant that these proposals mainly optimise TCP throughput for a single connection, and offer most benefit to large transfers. A typical Web Page is made up of many small icons, images and regions of text. If a separate TCP connection is required for each item, and each item represents a small amount of data, little or no improvement may be obtained. The relevance of each mechanism is summarised below with respect to highly asymmetric satellite Internet delivery systems;

Area	Mechanism	Use	Implementation
General	Path-MTU Discovery	Recommended	Sender
General	Link FEC	Recommended	Over Satellite Link
Congestion Control	Slow Start	Required	Sender
Congestion Control	Congestion Avoidance	Required	Sender
Congestion Control	Fast Retransmit	Recommended	Sender
Congestion Control	Fast Recovery	Recommended	Sender
TCP Large Windows	Window Scaling	Recommended	Sender, Receiver
TCP Large Windows	PAWS	Recommended	Sender, Receiver
TCP Large Windows	RTTM	Recommended	Sender, Receiver
Acknowledgements	TCP SACKs	Recommended	Sender, Receiver

Table 5-1. Modifications for enhancing TCP over satellite channels [57].

Path MTU (Maximum Transmission Unit) Discovery

Path MTU discovery requires the data sender to discover the Maximum Transmission Unit of the current connection so that TCP segments will not need to be fragmented in transit. This has little impact on the systems under investigation.

Link FEC (Forward Error Correction)

Forward error correction is recommended for each satellite link to correct transmission errors. The DVB response channel employs FEC and can be considered 'virtually error-free'. The *Satellite Data Reply Link* also employs FEC and also has low error rate.

Slow Start, Congestion Avoidance, Fast Retransmit & Fast Recovery [60] [61] [62]

The *slow start* and *congestion avoidance* algorithms must be used by the data sender to control the amount of data injected into the network [60]. When a connection is established, TCP slowly probes the network to determine the available capacity using the *slow start* algorithm. Once a balance has been achieved, the *congestion avoidance* algorithm maintains this level of throughput while probing less aggressively for further capacity. These mechanisms are mostly responsible for poor TCP throughput performance and are heavily constrained by the 600ms round trip time (RTT) of the satellite system. The *slow start* algorithm is also used when the TCP connection recovers from packet loss. The *fast retransmit* and *fast recovery* algorithms attempt to improve recovery time so that the sender need not revert to the *slow start* algorithm. These algorithms are implemented by the sender, but rely upon a particular acknowledgement policy by the receiver. The *fast retransmit* and *fast recovery* algorithms offer advantages to an asymmetric satellite Internet delivery system if supported by both sender and receiver.

Windows Scaling, PAWS & RTTM [63]

For *window scaling*, an extra parameter is communicated which increases the normal TCP acknowledgement window (with a bit shift). Since more unacknowledged data can be present on the network, *protection against wrap-around sequence* (PAWS) and *round trip time measurement* (RTTM) algorithms help to resolve ambiguity in the event that the TCP 16-bit sequence counter cycles between receipt of acknowledgements. These mechanisms, in particular *window scaling*, offer advantages to an asymmetric satellite Internet delivery system if implemented by both sender and receiver.

TCP SACKs (Selective ACKnowledgements) [63] [64] [65]

TCP may experience poor performance when multiple packets are lost from one window of data. As standard, TCP acknowledgements are cumulative and a TCP sender can only learn about a single lost packet per round trip time. An aggressive sender can retransmit early, but such retransmitted segments may have already been received. *Selective acknowledgement* (SACK) allows the data receiver to inform the sender about all segments that have arrived successfully; so that the sender retransmits only those which have been lost. This mechanism also offers advantages in an asymmetric satellite Internet delivery system if implemented by both sender and receiver.

Delayed Acknowledgements (DACKs) [63]

Since TCP acknowledgements are cumulative, many TCP implementations acknowledge only every Kth segment to reduce traffic on the request channel; this policy is known as *delayed acknowledgement* (DACK). Since the request channel in the highly asymmetric satellite Internet delivery system has much lower data rate than the response channel, implementation of this policy will reduce request channel traffic, and can improve throughput on the response channel as a result. Providing that the acknowledgement policy for other mechanisms is not violated, redundant ACKs could also be discarded by an intelligent network device driver to further reduce request channel traffic.

5.4.3 Enhancing Throughput with UDP/IP

UDP (the User Datagram Protocol) has been suggested as an alternative to TCP in order to improve throughput over high delay satellite links. TCP is usually responsible for providing reliability and flow control, but its 'slow start' and 'congestion avoidance' algorithms have identified as significant factors in reducing throughput performance over satellite links. UDP provides no reliability or flow control and passes this responsibility to the application-layer software; where optimisations can be made to match system characteristics. In the asymmetric satellite Internet delivery system, UDP/IP may be used to optimise transfers between ISP and User, but cannot be used throughout the Internet. For this reason, specialist 'Proxy Server' software is required at the ISP to fetch the requested information from the Internet using TCP/IP over low delay terrestrial links, Figure 5-8.

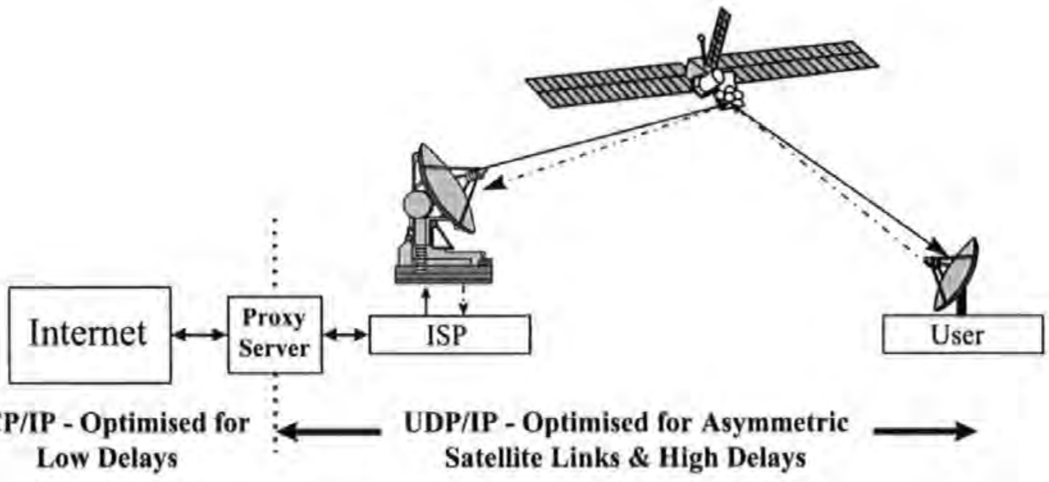
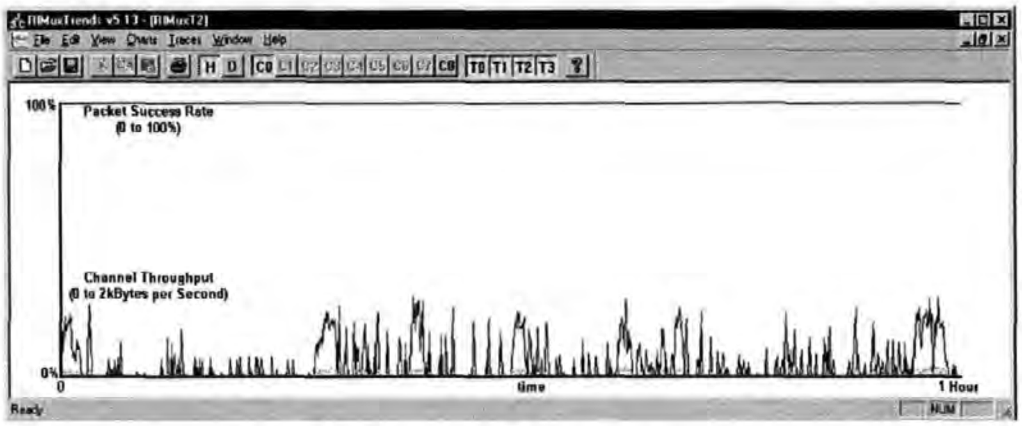

Internet
Proxy Server
ISP
User
TCP/IP - Optimised for Low Delays
UDP/IP - Optimised for Asymmetric Satellite Links & High Delays

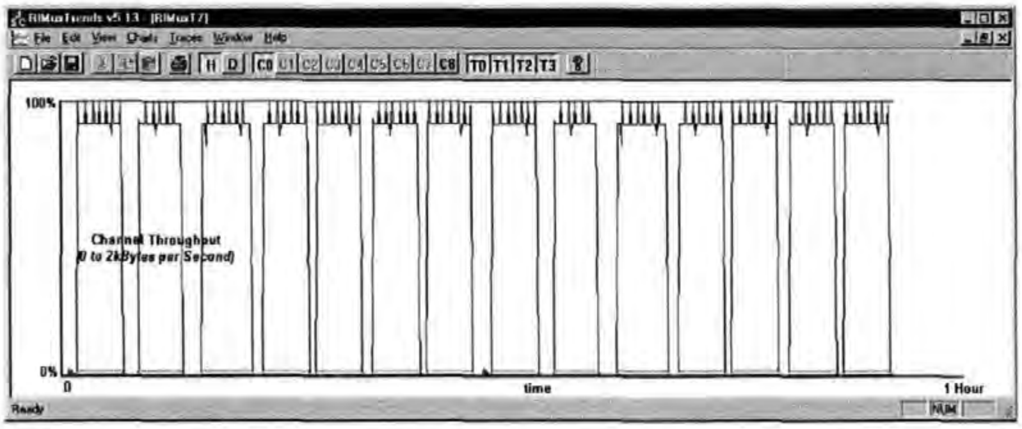
Figure 5-8. An optimised asymmetric satellite Internet delivery system.

5.5 System Performance

Figure 5-3 may be classified as a "highly asymmetric system" because the response channel provides 4Mbps and the low-speed request channel provides just 16kbps; a 262:1 ratio. However, this level of asymmetry is still sufficient to allow good Web Browsing and file upload performance. Figure 5-9a shows typical request channel activity for a single user over a 1 hour interval during aggressive HTTP activity (Web Browsing) and Figure 5-9b for repeated FTP (File Transfer Protocol) uploads of a 310 kByte test file. In both cases, transmissions were made on an ad-hoc basis over a request channel allocated exclusively to the user. For HTTP, throughput peaks at approximately one third of its potential but only a small fraction of the overall capacity is used. In contrast, during FTP uploads, maximum throughput is generated and the full request channel capacity is utilised; maximum theoretical throughput (2kBytes per second) cannot be achieved due to synchronisation overheads. These results indicate that each 16kbps request channel provides the potential capacity to support many simultaneous HTTP users or a single FTP upload.



a. Typical HTTP activity - Web Browsing.



b. Repeated 310kByte upload - FTP or E-mail upload.

Figure 5-9. Request channel performance for a single user - 1 hour interval.

5.5.1 Collision Reduction with Frequency Hopping Algorithms

For HTTP, a large percentage of request channel transmissions consist of TCP acknowledgements (ACKs). TCP acknowledgements are cumulative, and occasional packet loss may be tolerated without any perceivable effect; providing that another ACK arrives successfully soon after. Another common request channel transmission occurs when TCP wishes to establish a new connection; hereafter referred to as TCP opens (OPENs). OPENs are generated when new information is requested, and their loss results in retransmission after a short delay. Successive losses increase (double) the retransmission delay until the process is eventually aborted; an error is reported by the Web Browser. For FTP uploads, transmissions consist of maximum length (1514 byte) packets sent in quick succession; hereafter referred to as data packets (DATAs). As with OPENs, DATAs are retransmitted in the event of packet loss and termination of the connection occurs after repeated loss. Each of the aforementioned traffic types have different characteristics with respect to packet length, transmission frequency and throughput; this information is summarised in Table 5-2.

Packet Type	Typical Length	Transmission Frequency	Average Throughput	Probability of collision with ACKs	Probability of collision with OPENs	Probability of collision with DATAs
ACKs	56-Bytes	Often, bursty	Low	Low	Low	High
OPENs	< 200 Bytes	Often, bursty	Low	Low	Low	High
DATAs	1500 Bytes	Quick succession	High	High	High	High

Table 5-2. General characteristics of ACK, OPEN and DATA traffic.

Considering multiple simultaneous users, ACKs and OPENs have lowest probability of collision with other ACKs and OPENs due to their shorter length. DATAs have much longer packet length, and therefore generate a significantly higher probability of collision with all traffic types. In principle, for HTTP, the probability of collision between simultaneous users will reduce if the aforementioned traffic types are transmitted on separate request channels. Performance may be further improved if multiple request channels are provided for each traffic type. An intelligent 'frequency hopping' algorithm was implemented by the author within network device driver software to measure the effectiveness of this strategy. Figure 5-10 shows typical characteristics for request channels assigned to DATA traffic, ACK traffic and OPEN traffic over a 24 hour period (12:00 to 12:00). In this case, there were approximately 200 users, of which up to 20% are assumed to be active at any instant. For these measurements, 8 16kbps request channels were assigned to DATA traffic, 4 to ACK traffic and 4 to OPEN traffic. Results are discussed below;

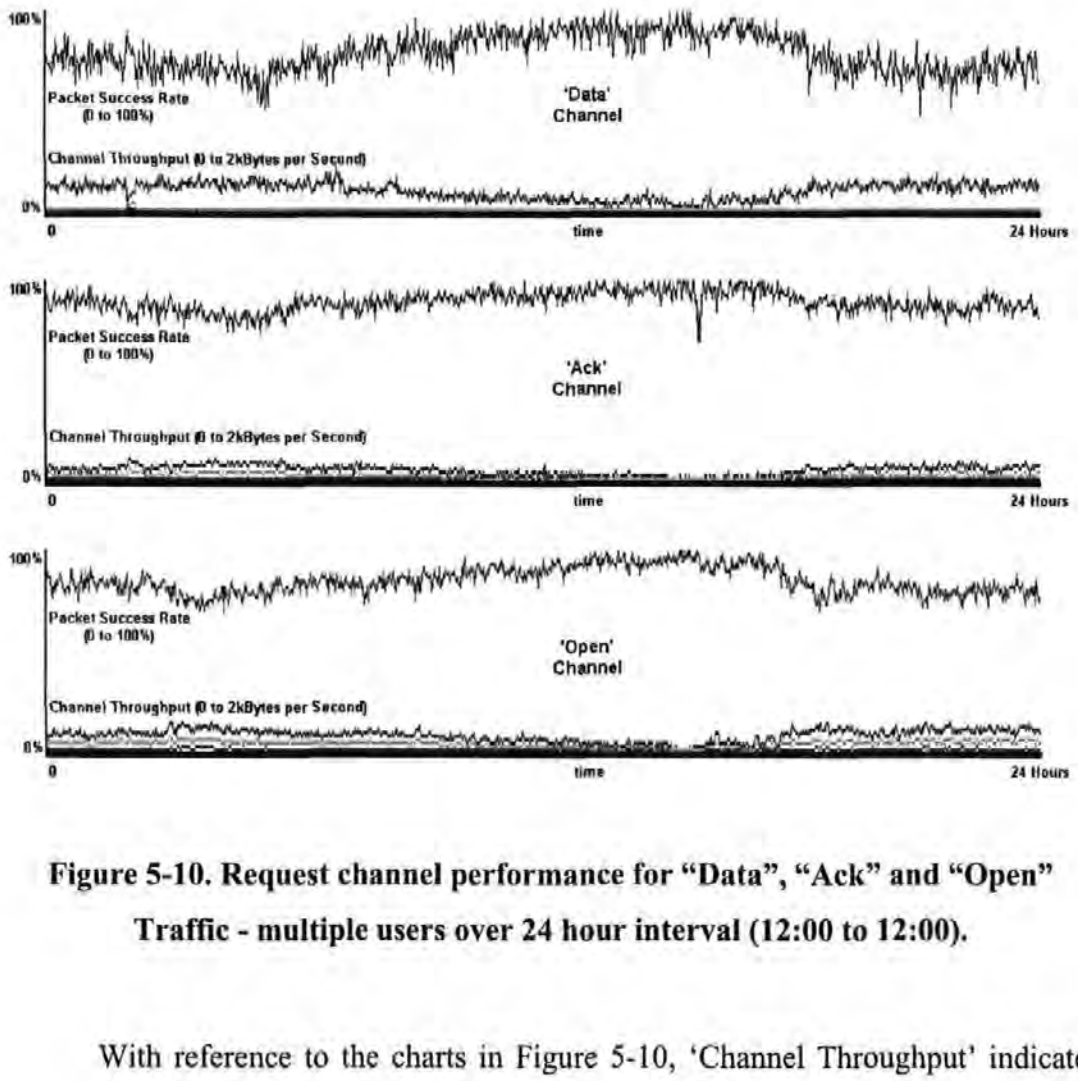


Figure 5-10. Request channel performance for "Data", "Ack" and "Open" Traffic - multiple users over 24 hour interval (12:00 to 12:00).

With reference to the charts in Figure 5-10, 'Channel Throughput' indicates the data throughput after demodulator synchronising overheads have been removed and 'Packet Success Rate' indicates (as a percentage) the number of Burst Demodulator acquisitions which resulted in reception of an error-free packet. The packet success rate can reduce from 100% due to transmission error (FEC failure), burst demodulator false acquisition or collision on the request channel but, since the packet success rate reduces as channel throughput increases, collision is clearly the dominant source of error in this case. The results in Figure 5-10 serve to support earlier analysis of the three traffic types; ie. DATA traffic has highest throughput and highest probability of collision, ACK and OPEN traffic produce lower throughput and lowest probability of collision. The results in Figure 5-10 also show that each channel is not being used efficiently, and that such 'frequency hopping' strategies are ineffective as a means to reduce collision probability.

5.5.2 Time Division Multiple Access (TDMA)

It was shown in section 5.5.1 that collision is a significant problem on shared request channels and that intelligent frequency hopping algorithms are ineffective as a means to reduce collision to acceptable levels. For the 'highly asymmetric' satellite Internet delivery system (Figure 5-3), the results in Figure 5-9a demonstrate that approximately one fifth of the 16kbps request channel capacity is utilised during HTTP activity (Web Browsing). In contrast, the results in Figure 5-9b demonstrate that the full request channel capacity can be utilised during FTP or SNMP uploads. In the former case, a TDMA (Time Division Multiple Access) channel sharing protocol is ideally suited as a means to eliminate collision, while providing negligible impact on overall performance. In the latter case, any reduction to the channel capacity offered will increase upload times accordingly. In general, TDMA transmissions are most accurately initiated under hardware control in order to minimise the guard interval between TDMA time-slots; make most efficient use of the channel. However, due to the practicalities of redesigning the *Satellite Data Reply Link* hardware, a TDMA protocol was implemented by the author within device driver software. Penalties paid for software implementation include a coarse timing resolution (10ms) and increased timing skew. Advantages of software implementation include increased flexibility and potential to dynamically vary the TDMA frame structure. The nominal TDMA frame structure employed for this investigation was determined by results in Figure 5-9a and consists of 5 time-slots of 150ms duration separated by 50ms guard intervals. This frame structure is repeated several times before a synchronising timing reference must be received from the ISP. Additional software was developed by others to deliver a timing reference from the ISP to the device driver and to manage assignment of TDMA time-slots.

The TDMA protocol implemented by the author operates as follows; Details about a group of request channels, on which proprietary TDMA control traffic may be transmitted, are sent from the ISP along with other system parameters. When frames are detected in the transmission queue, the device driver sends a 'Time-slot Request' message to the ISP. Depending upon current system loading, anything from 1 to the maximum number of time-slots may be allocated in response. If two or more adjacent slots are allocated to the same terminal, greater throughput is achieved by transmitting during the guard interval. The device driver continues to transmit until either the time-slot(s) are reclaimed by the ISP or the transmission queue has been emptied. Since there are often pauses between transmissions, when the queue is momentarily emptied, the device driver waits for a pre-set delay before releasing the time-slots; with a 'Time-slot Release' message. The TDMA protocol software contains a further level of complexity in that it continuously monitors utilisation of the allocated time-slots, and makes periodic requests for higher or lower transmit capacity. In practice it was found that a single time-slot is requested for moderate Web Browsing while the maximum number of slots are requested during e-mail and FTP uploads. In the author's opinion, this general strategy provides the most fair and efficient use of the request channel capacity for a highly asymmetric satellite Internet delivery system.

Figure 5-11 shows typical TDMA request channel utilisation during a large FTP upload. In this case, the request channel was reserved specifically for the test and the dynamic bandwidth algorithm artificially slowed to provide clearer results. At the beginning of the chart, only one time-slot is allocated to the user. Additional slots are requested at 12 minute intervals, and the next adjacent slot is allocated each time by the ISP. This process is repeated until all 5 time-slots have been allocated and maximum throughput is achieved. It should be noted that throughput is accurately controlled, and that variations indicated in Figure 5-11 are a function of the sampling interval in the author's charting software. The benefit of transmitting in the guard band between adjacent slots can be appreciated by comparing throughput for a single slot with that of two adjacent slots; throughput increases by a factor greater than two. For Web-Browsing, it was observed that better results are achieved when two equally spaced time-slots are allocated. This is explained by the corresponding reduction to the maximum round trip time.

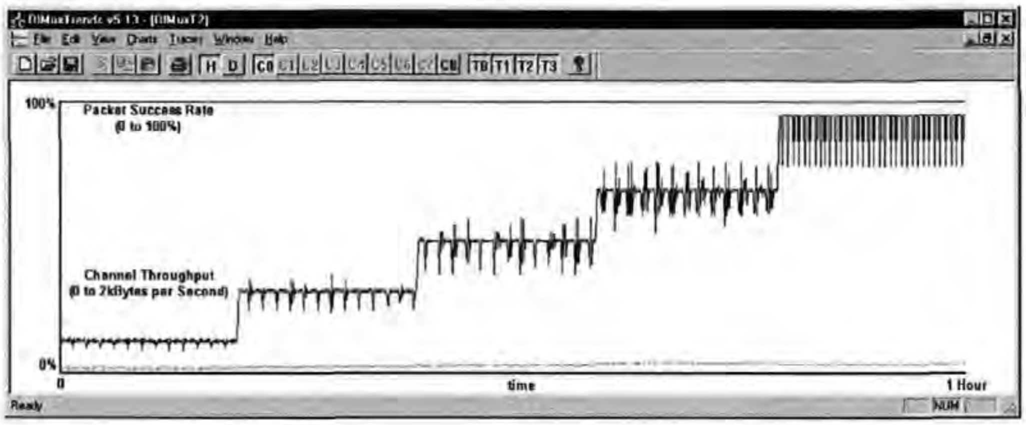


Figure 5-11. TDMA request channel performance - single user over 1 hour interval, FTP upload with dynamic bandwidth allocation.

5.6 Conclusion

In this chapter the author demonstrated a practical application for the *Satellite Data Reply Link* principle (Chapter 1) and the DSP Burst Demodulator algorithms (Chapter 3) in highly asymmetric satellite Internet delivery systems. Three systems were presented in which the author contributed software drivers to link the DSP modulator and demodulator hardware to the Internet Protocol Suite of a standard PC's operating system. This software provided emulation of a common network technology and additional functionality to overcome limitations of the system. It was shown by experiment that satellite delays result in a reduction to TCP/IP throughput performance and many enhancements to TCP have been proposed. In reviewing these proposals it was concluded that significant improvements would only result for large transfers; such as FTP downloads. Another solution was discussed where TCP/IP is replace by UDP/IP to optimise transfers over the satellite segments of the system. This solution requires the use of specialised 'Proxy Server' techniques but provides, in the author's opinion, the most effective mechanism with which to optimise throughput performance.

In section 5.5, it was suggested the there are two main applications for these systems; Web-Browsing (receiving information) and FTP/SNMP uploads (sending information). By experimental result it was shown that approximately one fifth of a 16kbps request channel capacity is required in the former case, while the full capacity may be utilised in the latter case. These results indicate that each request channel may be shared by several users for Web Browsing, but that the potential for collision (packet loss) must be considered. TCP/IP is a robust protocol, and occasional packet loss is tolerated with the use of acknowledgements and re-transmissions. Three main traffic types were identified and a prioritised 'frequency hopping' channel access protocol was investigated as a means to reduce packet loss. Experimental results showed that this protocol was ineffective, and that collisions were a fundamental system limitation. A dynamic TDMA channel access protocol was also investigated which eliminates the potential for collision and assigns request channel capacity 'on-demand'. Experimental results showed that this solution provided the most flexible and efficient use of the *Satellite Data Reply Link*.

A fundamental limitation of the systems presented in this chapter results from the low request channel data rate (16kbps). When this is combined with a TDMA channel access protocol, the request channel may be reduced to just 2kbps. This imposes a severe limitation on data upload times and precludes applications such as video conferencing. To support future Internet applications, it is clear that the trend must be towards more symmetrical systems. The request channel data rate may be increased with the use of higher level modulation schemes such as QPSK (Quadrature Phase Shift Keying) and QAM (Quadratue Amplitude Modulation), and other improvements might be obtained with more powerful FEC schemes (Turbo Codes) and faster DSP hardware. However, a balance must always be maintained between data rate, cost of the user terminal and bandwidth requirements for such a system to remain viable.

6. Conclusions

6.1 Contributions to Knowledge

The author's contributions to knowledge are summarised below.

- ***Chapter 2:***
- Description of a real-time DSP software-based implementation of a Doppler Tracking modem for use with PoSat-1 and other LEO microsatellites.
- Description of the effects of Doppler frequency error in FM-based systems.
- Evaluation of novel FFT-based algorithms for coarse/fine Doppler tracking and frequency synchronisation.
- Description and evaluation of compromises and optimisations to DSP software-based algorithms for frequency modulation, frequency correction, root raised cosine filtering and frequency discrimination in terms of processing overheads and memory utilisation.
- Evaluation of block processing and sub-sampling strategies to further reduce processing overheads in DSP software implementation.
- Discussion of limitations associated with fixed-point software implementation.

- ***Chapter 3:***
- Description of a real-time DSP software-based implementation of a Burst Demodulator for use in a Satellite Data reply Link.
- Description of an efficient and general frame structure for transmitted bursts.
- Comparison of FFT and Offset-FFT algorithms for carrier frequency acquisition and a description of the improvements offered by the Offset-FFT in terms of estimation error and acquisition performance at low SNR.

- Implementation details of a general Offset-FFT/FFT algorithm and strategies for organisation of coefficients in memory.
- Description of single and dual-DSP implementations of the Offset-FFT algorithm for real-time execution.
- Description and evaluation of a delay and multiply symbol clock recovery algorithm utilising a 2nd order IIR resonator filter, and real-time DSP software implementation.
- Modifications to standard Viterbi decoder algorithm to aid real-time DSP implementation.
- Results from back-to-back and satellite trials of a DSP software-based burst demodulator for D-BPSK and D-QPSK modulation.

- ***Chapter 4:***
- Description and mathematical comparison of real-sampling and complex-sampling OFDM modulator and OFDM demodulator models.
- Analysis of the effects of coarse and fine carrier frequency error on demodulated data in OFDM systems.
- Use of the Offset-FFT to provide simultaneous frequency correction and OFDM demodulation with minimal additional overhead.
- Description and evaluation of low complexity DSP software-based algorithms for simultaneous coarse and fine frequency synchronisation with a suitable preamble data sequence.
- Mathematical analysis of the effects of coarse and fine OFDM symbol timing error on demodulated data.

- Description of a low complexity DSP software-based algorithm for coarse OFDM symbol timing synchronisation with a suitable preamble sequence.
- Description of a low complexity DSP software-based algorithm for fine OFDM symbol timing synchronisation with a suitable preamble sequence.
- Results from simulated tests of coarse and fine OFDM symbol timing synchronisation algorithms.

- ***Chapter 5:***
- Definition of symmetric and asymmetric Internet delivery systems and overview of a practical asymmetric satellite Internet delivery system.
- Description of standard and novel software and techniques and their implementation to provide IP transmission over a satellite data reply link.
- Discussion of strategies for packet prioritising/filtering, user authentication and distribution of system parameters.
- Discussion of the effects of satellite delay on TCP/IP throughput and review of standard modifications to TCP with respect to the systems proposed.
- Description of a suitable demand-assigned TDMA protocol for optimising Web-Browsing and File transfer performance and efficient channel sharing in asymmetric Internet delivery systems.
- Comparison of frequency-hopping protocols and demand-assigned TDMA protocols for minimising collisions on shared return channels.

6.2 Conclusions and Future Work

The aims for the work in this Thesis were to investigate practical algorithms for real-time implementation of software-based digital modems for use in low-cost satellite links. Suitable algorithms and modem structures have been described which meet these aims and provide a reasonable compromise between speed and performance. Throughout this work the execution speed of the digital signal processor (DSP) has determined the maximum symbol rate that could be achieved but, as technology improves, this work can be scaled to operate at higher sample rates, to achieve higher symbol rates and to provide greater timing and frequency resolution. Three modulation formats were investigated in this work; frequency modulation, phase modulation and orthogonal frequency division modulation (OFDM). The digital modems discussed in this Thesis cannot be directly compared although it is interesting to note that a low-IF transceiver with frequency and timing synchronisation would require approximately 4 DSPs at 9600bps for frequency modulation, 3 DSPs at 96kbps for QPSK phase modulation and 4 DSPs at 256kbps for OFDM. It is clear from this that frequency modulation is the most intensive in terms of demodulation algorithm computations and that phase modulation and OFDM can help to achieve much higher channel data rates with the same signal processing hardware. In terms of frequency and symbol timing synchronisation requirements OFDM is most sensitive to frequency and symbol timing errors while phase and frequency modulation are more tolerant. For software-based digital modems phase modulation therefore provides an acceptable compromise between sensitivity to synchronisation errors and data throughput.

Practical application of this work in asymmetric satellite Internet delivery systems has also been discussed along with the novel techniques required to map TCP/IP traffic to low-cost satellite links. A dynamic demand-assigned time division multiple access (DA-TDMA) protocol was described which allows each user terminal to determine its return channel bandwidth requirements. This helps to reduce processing overheads at the hub station and allows many terminals to share return channels in a fair and efficient manner.

The algorithms in this work were implemented directly using DSP assembly code because, at the time, it allowed the most efficient software to be produced. The DSPs used in this work have a relatively complex architecture which was considered when producing optimised real-time software. The most recent DSPs employ multiple parallel processing units and much deeper instruction pipelines, so their operation can be significantly harder to predict. This means that writing directly in assembly code for the latest DSPs, although still possible, is no longer such a manageable task. As with the processors, DSP software development tools have also improved significantly over the last few years. With the latest tools it is now possible to implement all DSP algorithms in a higher-level language, such as C, and for a compiler to produce the final optimised real-time software. Development tools can now automatically identify bottlenecks within intensive algorithms and can even rearrange algorithms to best match the target architecture and eliminate bottlenecks altogether. Since optimising DSP assembly code requires significant manual effort, these new development tools allow the researcher to dedicate more time to investigating novel algorithms and architectures.

Since this work commenced there have been significant improvements to Digital Signal Processor (DSP) technology. When this work commenced the performance of the fastest DSP was quoted in tens or hundreds of MIPS (Millions of Instructions Per Second), at its conclusion performance is quoted in thousands of MIPS and higher. A DSP performance increase of this order of magnitude is exciting and can be applied to the work in this Thesis in three main ways; The first is to implement the <u>same</u> digital modems and exploit the additional processing potential to reduce (or eliminate) compromises to theoretical performance which were originally required in order to achieve real-time operation. The second is to implement the <u>same</u> digital modems and exploit the additional processing potential to include superior algorithms, such as Turbo FEC algorithms, which were previously too intensive to be implemented in real-time. The third, following the software radio concept, is to shift the boundary between the analogue and digital domains a step further towards the R.F. input.

A practical application from this work for increased processing potential would be to simultaneously implement multiple modems within a single processor. Specifically, the burst demodulator in Chapter 3 required approximately 80 MIPS and so a more recent 1600 MIPS processor could simultaneously implement up to 20 of these burst demodulators. In terms of satellite Internet delivery systems, Chapter 5, this alone could dramatically reduce the overall cost and size of the Hub station signal processing equipment. Multiple modems could also be arranged within a single processor with specific timing and frequency offsets to reduce the synchronising preamble required at the start of each transmitted burst. Alternatively, a more powerful processor might implement a more intensive frequency synchronisation algorithm, covering a much wider bandwidth, and sequence the operation of multiple (less complex) modems implemented within the same processor. Such a strategy could be applied to on-board satellite processing where signals can be demodulated and regenerated by the satellite to improve link margins.

The software radio concept is a long term goal and aims to build flexible radio systems which are multi-service, multi-standard, multi-band, reconfigurable and reprogrammable by software. Technology does not yet exist to allow the input RF signal to be sampled directly and for all signal processing to be performed in the digital domain. The work in this Thesis shows how the same signal processing hardware can be reconfigured by software to operate with various modulation formats and to provide baseband signal processing from a low IF onwards. For high sample rates (>>1Msps) even the latest signal processors cannot provide the necessary execution speed to implement all sections of a digital modem. Field programmable gate arrays (FPGAs) can perform certain tasks, such as FIR filtering, with much greater speed and efficiency than is possible with software algorithms. However, FPGAs cannot match the flexibility offered by software in terms of the ability to branch to different locations within a programme. Software radio is the long-term future, but will have to rely upon FPGA and analogue technologies for many years until signal processors have advanced sufficiently.

A. DSP Hardware

The algorithms and techniques presented in this Thesis have been either implemented on or designed for digital signal processing (DSP) hardware based around the Texas Instrument TMS320C50 (C50) fixed-point digital signal processor. This chapter begins with an overview of the C50 processor and the features it provides, and a generic C50 DSP system is described. Industrial funding was provided to develop a general purpose dual-C50 DSP system board for use in these investigations. The dual-C50 system was designed by others but is discussed in detail as its characteristics have influenced the work presented in this Thesis. The chapter ends with a discussion of the methods and strategies used in assembly code software.

A.1 TMS320C50 Fixed-Point Digital Signal Processor

In 1982, Texas Instruments introduced the first fixed-point digital signal processor in the TMS320 series, the TMS320C10. When research for this Thesis commenced, the TMS320 family consists of five generations; 'C1x', 'C2x', 'C3x', 'C4x' and 'C5x'. The 'C1x', 'C2x' and 'C5x' generations are fixed-point processors while the 'C3x' and 'C4x' are floating-point processors. Source code is upwards compatible from one fixed-point generation to the next and likewise from one floating-point generation to the next. Due to its low cost, and potential performance gains using fixed-point arithmetic, the TMS320C50 was chosen over its the TMS320C40 for this and other research. Table A-1 lists applications for the C50.

Telecommunications 1	**Telecommunications 2**	**Voice/Speech**
1200 - 19200 bps Modems	DTMF Encoding/Decoding	Speech Enhancement
Adaptive Equaliser	Echo Cancellation	Speech Recognition
ADPCM Transcoder	FAX	Speech Synthesis
Cellular Telephone	Line Repeater	Speaker verification
Channel Multiplexing	Speaker Phone	Speech Vocoding
Data Encryption	Spread Spectrum communications	Voice Mail
Digital PBXs	Video Conferencing	Text-to-Speech
Digital Speech Interpolation	X.25 Packet Switching	
Instrumentation	**Medical**	**Military**
Digital Filtering	Diagnostic Equipment	Image processing
Function Generation	Fetal Monitoring	Missile Guidance
Pattern matching	Hearing Aids	Navigation
Phase-Locked Loops	Patient Monitoring	Radar Processing
Seismic Processing	Prosthetics	Radio Frequency Modems
Spectrum Analysis	Ultrasound Equipment	Secure Communications
Transient Analysis		Sonar processing

Table A-1 Applications for the TMS320C5x DSPs (Source - Texas Instruments).

The C50, like all DSPs, is a microprocessor with an instruction set designed to optimise digital signal processing algorithms. The C50 has an advanced Harvard Architecture which provides separate buses for accessing program memory, data memory and input/output (I/O) space as shown in Figure A-1. Using this architecture, future instructions can be fetched and decoded while other instructions are still executing. The 'instruction pipeline' that is formed allows up to 4 instructions to be executing at any time.

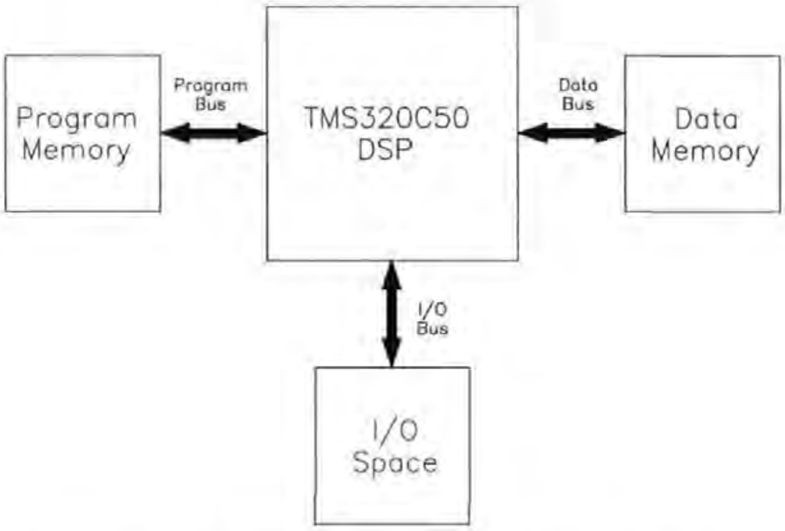


Figure A-1. TMS320C50 'Harvard' architecture.

When this research commenced, the C5x processors were available with instruction cycle times of 25ns, 35ns and 50ns corresponding to 80MHz, 57MHz and 40MHz variants. As a 16-bit processor, the C50 can address 64k (2^{16}) locations on each of the program, data and I/O busses and has provision to address a further 32k locations of Global RAM (accessible by multiple devices simultaneously). The processor has on-chip memory, 9.5K of which may be dynamically mapped to program and data address space. Figure A-2 shows memory maps for program and data address space and the possible configurations for on-chip single access RAM (SARAM) and on-chip dual access RAM (DARAM) using the OVLY, CNF and RAM control bits in the status register ST1. For optimum performance a program executes from on-chip program memory and likewise interim results are read from and written to on-chip data memory. The on-chip SARAM requires one full machine cycle to perform a read or write while the DARAM may be read from and written to in the same cycle.

Slower external memory is accessed by inserting between 1 and 7 wait states using on-chip programmable wait state generators.

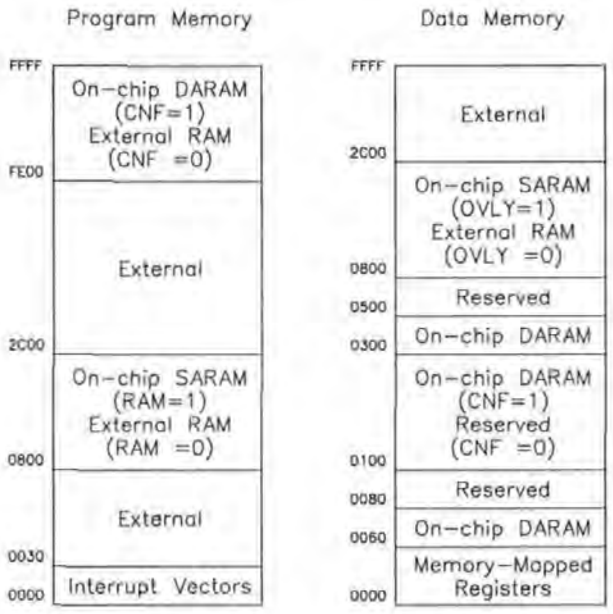


Figure A-2. C50 memory maps.

The C5x processors have a 32-bit arithmetic logic unit (ALU), 32-bit accumulator, 32-bit product register and a 16-bit parallel logic unit (PLU). For indirect addressing there are 8 auxiliary registers and 11 shadow registers allow the main register's context to be saved and restored during interrupt service routines (ISRs). For signal processing applications, the major features are a single-cycle multiply and accumulate instruction, two indirectly addressable circular buffers, bit reversed addressing (for radix-2 fast fourier transform (FFT) algorithms) and a 16-bit barrel shifter. The C50 DSP has several on-chip peripherals, of particular interest are the two high speed serial interfaces which operate at one fourth of the processors cycle time. The first is a synchronous, full-duplex port capable of transmitting data framed as either bytes or words. The second is a full-duplex port which can be configured to operate in either synchronous or time division multiple access (TDM) modes. The former can be used to provide high speed communication between two devices only while the latter allows communication between up to eight devices. The high speed serial ports form the basis of the multi-processor DSP system discussed elsewhere in this Thesis.

A.1.1 A Generic TMS320C50 DSP System

To interface with analogue signals and perform a signal processing function, the C50 must be provided with I/O peripherals and a program. The program is generally stored in EPROM mapped to the start of the program address space; it can be later copied to faster memory for execution. Depending upon the size of the program and the amount of interim data storage required, external RAM may be provided to supplement on-chip memory. Optionally, if there is the need for two devices to simultaneously access the same data, Global RAM is mapped to the data address space. Finally, to interface with analogue signals, a digital to analogue converter (DAC) and analogue to digital converter (ADC) must be mapped to the I/O address space. The generic system described above is shown in Figure A-3.

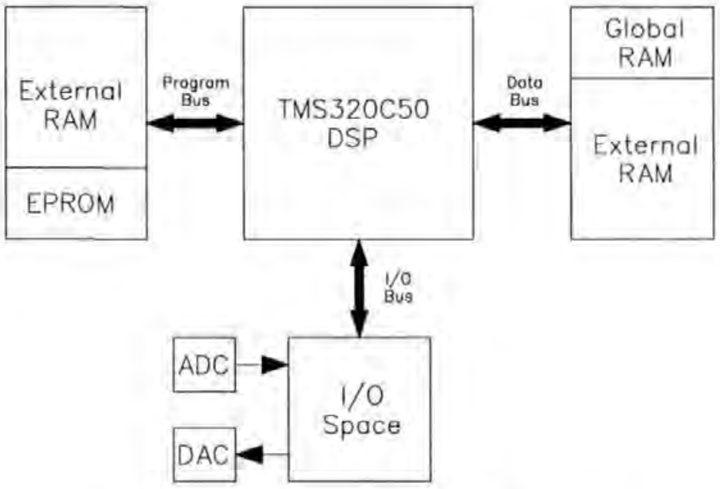


Figure A-3. Generic C50 DSP system.

A.2 Dual-C50 DSP System

A dual-C50 DSP system was developed by the Satellite Communications Research Centre at the University of Plymouth to provide a flexible platform for implementing DSP systems and algorithms. The dual-C50 system board contains two generic C50 systems, Figure A-4, and each processor is provided with memory and peripherals which allow it to operate independently of the other processor. Additional devices and connections are provided on the board to allow inter-processor communication (between processors on the same system board) and inter-board communication (between processors on different system boards). Processing power can therefore be effectively doubled by using both processors on the system board and, where the application permits, further increased by using multiple system boards. This flexible design allows up to 4 system boards (8 processors) to be used together. It should be noted that inter-processor communication incurs additional overhead therefore, to take full advantage of multiple processors, algorithms must be split most efficiently.

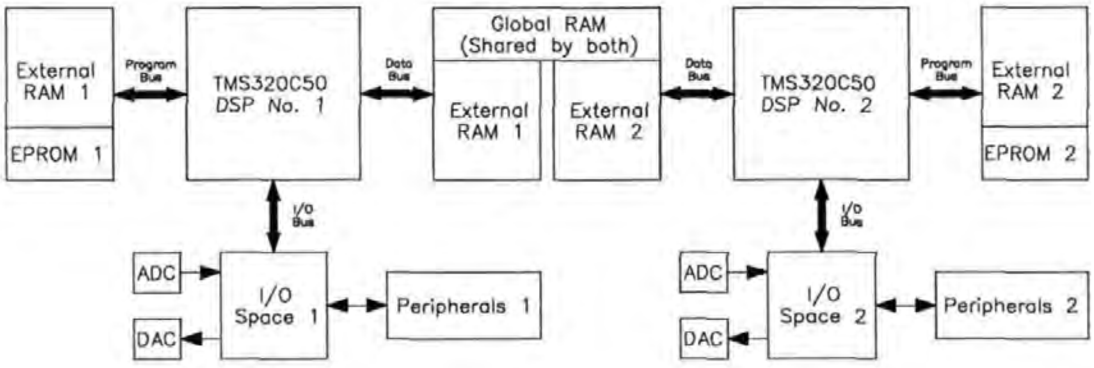


Figure A-4. Dual-C50 DSP system overview.

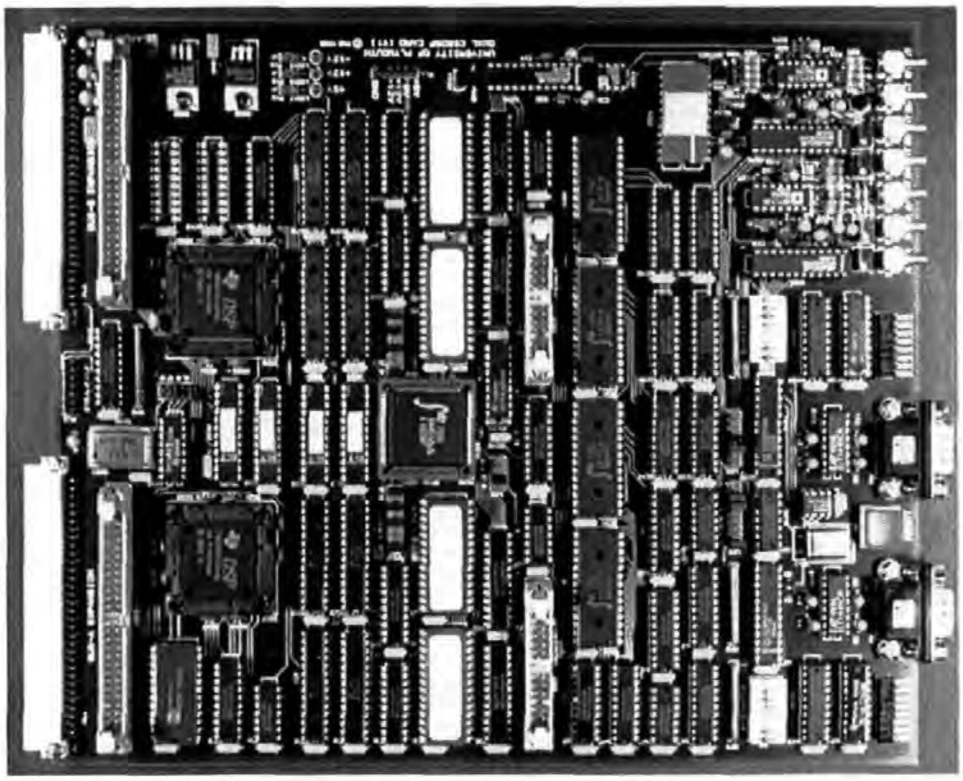


Figure A-5. Dual-C50 DSP system board.

A.2.1 Dual-C50 System Board

A photograph of the dual-C50 system board is shown in Figure A-5 and a block diagram in Figure A-6. Referring to the photograph, the hardware is nominally arranged so that the first DSP system occupies the lower half of the board and the second system occupies the upper half. The PCB has eight layers and all devices are socketed to allow upgrades and modifications to be carried out. To the right of the photograph (at the front of the board) the analogue input/output connectors appear at the top of the picture and serial interfaces towards the bottom. Parallel interfaces appear towards the centre of the picture and expansion sockets at the left hand side. The peripherals of particular interest and typical uses for them are summarised below. Unless otherwise stated, each is duplicated on the board.

12-bit ADC

The main ADC is buffered by a 1K 'first in first out' (FIFO) memory and can be read by either processor, a second ADC is provided for processor 2. Sample clocks are derived from the processors on-chip timer or from the master crystal oscillator.

12-bit DAC

Processor 1 has a high speed single channel DAC which is FIFO buffered, processor 2 has a 4 channel DAC. Sample clocks are derived from the processors on-chip timer or from the master crystal oscillator.

16-bit Digital Interface

Provides a parallel interface to a personal computer and may be configured as 16 outputs, 16 inputs or 8 inputs/8 outputs.

UART (Universal Asynchronous Receiver Transmitter)

Provides a serial interface with a personal computer. Also allows a dumb terminal to provide input to the system.

LED (Light Emitting Diode) Bank

8 LEDs which may be used for diagnostic or status displays.

DIL (Dual In Line) Switches

8 DIL switches which may be used to provide basic input to the system.

Programmable Counters

Used to generate sample clocks and can only be programmed by processor

Global RAM

Accessible by both processors and allows high speed transfer of information.

Watchdog Timer

A reset is applied to the board one second after processor activity ceases.

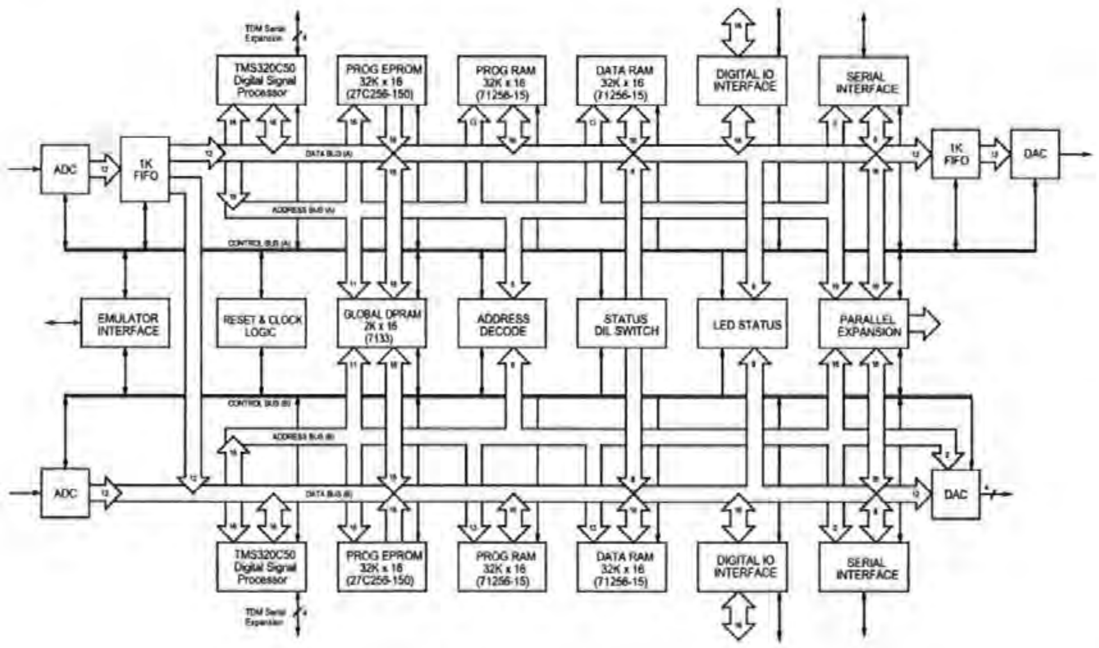


Figure A-6. Dual-C50 DSP system block diagram.

Referring to Figure A-6, the upper DSP system is designed for optimum real-time performance as both the ADC and DAC have associated with them a 1k FIFO buffer. This is significant as processing of a sampled signal does not have to be performed on a sample by sample basis. The FIFO buffer allows up to 1024 samples to be taken with subsequent processing performed on the block of samples. This improves efficiency and serves to eliminate lost samples due to the processor not being able to service the ADC or DAC interrupt reliably in real-time. In contrast, the lower DSP system does not have FIFO buffers associated with it's ADC and DAC. In all but the most trivial case, an algorithm requiring analogue to digital conversion and then digital to analogue conversion could be implemented far more efficiently, and with less chance of lost samples, on the upper DSP system. The major advantage of the lower DSP system, and the strategy behind its design, is that it provides a DAC with 4 output channels that can generate 4 phase locked signals for debug purposes. In all other respects, the two DSP systems are identical and when the ADC and DAC are not required, either system may be selected.

Two mechanisms provide inter-processor communication on the system board. The first is the on-chip serial ports and the second is the shared memory. Information can be passed between processors considerably faster if it is written to shared memory by the sending processor and read back by the receiving processor than via the serial ports. Optimum performance is achieved by using the serial port as a signalling channel to inform the other processor that data is waiting in shared memory to be read.

A.2.2 Dual-C50 DSP System Board Memory Maps

Three memory maps for the dual-C50 system will be described. The first, Figure A-7, is the default memory map (corresponding to when power is first applied to the system) for program and data space and gives a compact description of the external memory provided in the system. The second, Figure A-8, shows an optimised configuration which was used extensively for executing real-time DSP software. Both memory configurations are valid for either processor on the system board. The third memory map, Figure A-9, shows the organisation of I/O space.

By default (after a hardware reset) the program counter is reset, and the C50 begins execution at location zero in program memory. In this default state the processor is configured for maximum wait states so that processor initialisation code may execute from slow memory. Program memory in this mode consists of a 32k EPROM and 32k RAM, and data memory consists of a 32k RAM. The first 48 locations of program space are reserved for interrupt vectors and many locations at the start of data space are reserved for on-chip memory mapped registers.

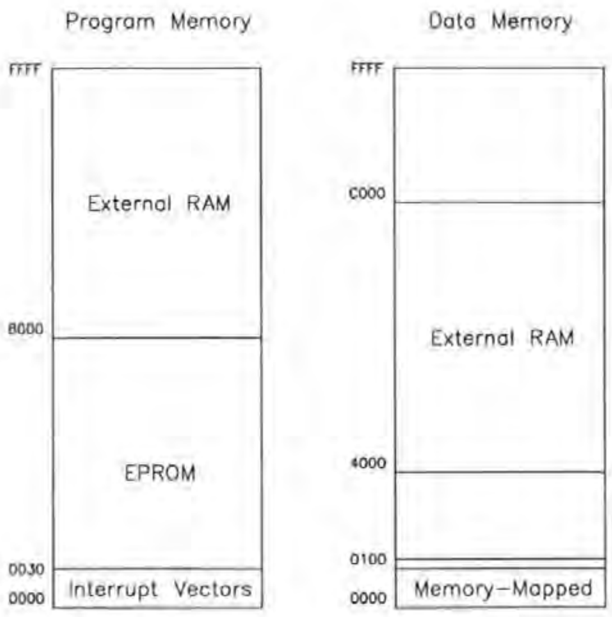


Figure A-7. Dual-C50 DSP system default memory maps.

The EPROMs used in the dual-C50 system requires 7 wait states for each read or write, corresponding to the slowest memory the C50 may access, so significant performance gains are achieved by copying the main program to on-chip program RAM where it can execute without wait states. The external RAM used in the dual-C50 system requires 2 wait states, hence significant gains can also be achieved by enabling and using on-chip data memory where possible. Figure A-8 shows the memory maps for an optimised configuration which has been used throughout development of software for the dual-C50 system. By comparing Figure A-8 with Figure A-7 the reader will notice that on-chip SARAM has replaced a range of locations between 0x0800 and 0x2C00 in both program and data space. These 9k locations are unique as they simultaneously appear in both program and data space and may be accessed via both busses. A decision was made to reserve the first 2k locations for use in program space and the remaining 7k locations for data space, and all software produced adheres to this rule. This does however leave a 9k hole in program and data memory where the external memory cannot be used. This is particularly significant in program space as the EPROM can no longer be read at these locations. A similar situation also exists in data space where ranges of the external RAM are replaced by on-chip memory. A final point of interest is the addition of 2k Global RAM locations at the top of data space. This RAM is common to both processors on the system board and on-chip logic prevents contention if both processors simultaneously attempt to access the same location.

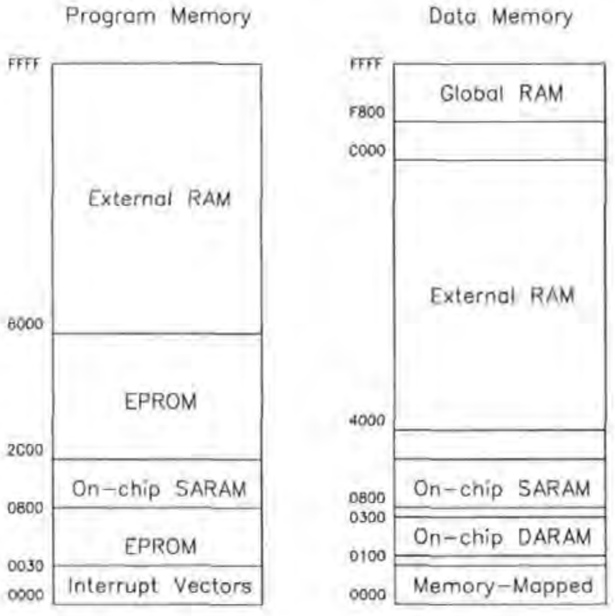
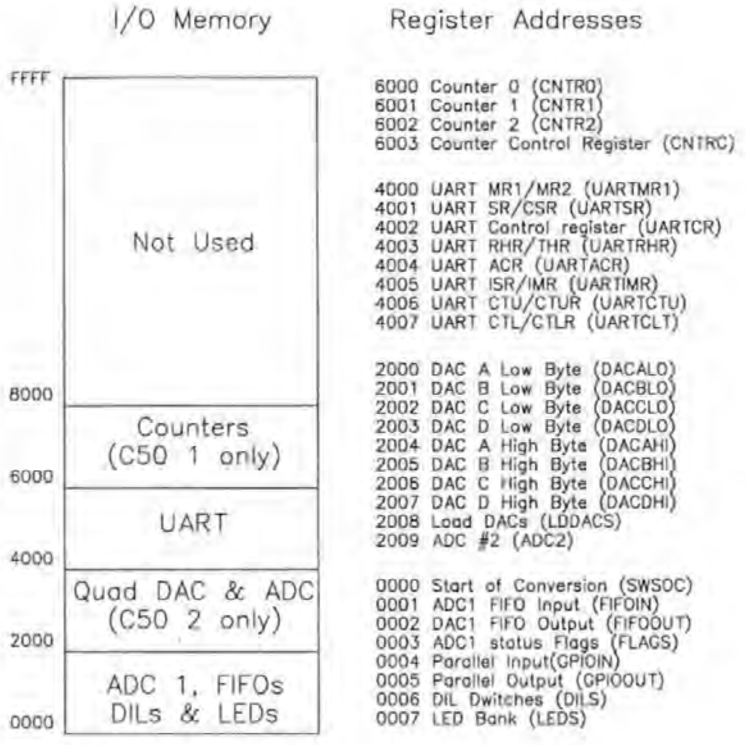


Figure A-8. Dual-C50 DSP system optimised memory maps.

Figure A-9 shows the memory map for I/O space and the addresses assigned to the registers within each peripheral. I/O space may be divided into 8k regions so that each may be assigned a different number of wait states for accessing the peripherals mapped within the address range. The first 16 locations of I/O space are particularly important as they also appear as memory mapped registers at the beginning of data space. Although this may seem confusing, locations 0x0000 to 0x000f in I/O space can be accessed by a greater range of instructions than the remainder of I/O space; which may only be accessed using the IN and OUT instructions. By placing the most commonly accessed peripherals in this range, significant performance gains may be achieved.



6000	Counter 0 (CNTR0)
6001	Counter 1 (CNTR1)
6002	Counter 2 (CNTR2)
6003	Counter Control Register (CNTRC)
4000	UART MR1/MR2 (UARTMR1)
4001	UART SR/CSR (UARTSR)
4002	UART Control register (UARTCR)
4003	UART RHR/THR (UARTRHR)
4004	UART ACR (UARTACR)
4005	UART ISR/IMR (UARTIMR)
4006	UART CTU/CTUR (UARTCTU)
4007	UART CTL/CTLR (UARTCLT)
2000	DAC A Low Byte (DACALO)
2001	DAC B Low Byte (DACBLO)
2002	DAC C Low Byte (DACCLO)
2003	DAC D Low Byte (DACDLO)
2004	DAC A High Byte (DACAHI)
2005	DAC B High Byte (DACBHI)
2006	DAC C High Byte (DACCHI)
2007	DAC D High Byte (DACDHI)
2008	Load DACs (LDDACS)
2009	ADC #2 (ADC2)
0000	Start of Conversion (SWSOC)
0001	ADC1 FIFO Input (FIFOIN)
0002	DAC1 FIFO Output (FIFOOUT)
0003	ADC1 status Flags (FLAGS)
0004	Parallel Input(GPIOIN)
0005	Parallel Output (GPIOOUT)
0006	DIL Dwitches (DILS)
0007	LED Bank (LEDS)

Figure A-9. Dual-C50 DSP system I/O memory map.

A.3 C50 DSP Software

DSP software can be produced directly in assembly code or using a higher level language compiler. The software tools supplied by Texas Instruments include a standard C compiler but, the code generated is not always as efficient as well written assembly code. Programming in assembly code however requires an intermediate understanding of the hardware in order to achieve optimum results. In this section, the author briefly describes a generic strategy for developing assembly code for the C50 and the steps that must be followed. Where possible, examples relating to the dual-C50 system are given.

A.3.1 C50 Assembly Code Program Structure

A C50 assembly program is made up from text files which define 'initialised' and 'uninitialised' sections. 'Initialised Sections' refer to executable code and tables of data which must reside in program memory, usually EPROM, when power is first applied to the system. Executable code may be transferred to faster program memory during the initialisation phase and data tables are subsequently transferred to data memory. 'Uninitialised' sections refer to areas of data memory which are reserved for storing interim results or data tables during execution of the main program. When developing software, the writer uses relocatable code and avoids absolute addressing where possible. A program must be accompanied by a 'Link Command File' which tells the linker where each section of code is to appear in the memory map. This method means that only the link command file needs to be altered to reflect a change in the target system. As an example, if system A has in data memory a 32k RAM based at address 0x4000 and system B a 32k RAM based at location 0x8000, only the link command file needs to be changed to transfer the program from system A to system B. If absolute addressing was used this would require all references to addresses in data RAM to be changed.

For a simple C50 assembly program there are generally four main 'initialised sections' required; interrupt vectors, initialisation routine, interrupt service routines and the main program. Each is described briefly below;

Interrupt Vectors

When the processor is reset it begins execution at the first location in program memory, where the interrupt vectors must be initially stored. The first is the reset vector which is an instruction that causes the processor branch to the start of the initialisation routine. Other interrupt vectors must be defined so that the processor branches to the correct interrupt service routine when its corresponding interrupt is asserted. For optimum performance, and minimum interrupt latency, the interrupt vectors can be transferred to faster memory during the initialisation routine.

Initialisation Routine

The initialisation routine refers to the 'housekeeping' code that must be executed in order to set the processor to the desired operating modes in preparation for the main program. For the C50 this includes setting the wait state generators for the memory attached to the system, enabling on-chip memory, initialising the peripheral devices and transferring 'initialised sections' to their run-time memory locations. This code is usually executed just after the reset vector is taken and the final instruction often causes the processor to branch to the start of the main program.

Main Program

The main program defines the algorithm or algorithms that the processor must implement. The main program will often be further divided into several sections, with most critical code assigned to the fastest memory.

Interrupt Service Routines

Interrupt service routines are small programs which are used to improve performance when accessing external peripherals. Rather than regularly polling each peripheral to test if the processors attention is required, which can be extremely inefficient, each peripheral provides a hardware signal to the processor called an interrupt. Each interrupt has a corresponding interrupt vector which provides a simple branch instruction to the interrupt service routine that deals with that device. For example, if an ADC samples an analogue signal at regular intervals it would generate an interrupt at the end of conversion (EOC) to indicate that it has new data for collection.

A.3.2 C50 Assembly Code Program Example

A simple example C50 assembly code program, which contains the four 'initialised sections' discussed in the previous section, is given below. Sufficient information is provided in the program's comments to allow the reader to follow the code hence, only points of particular interested are discussed in the main text. Each section of code begins with the assembler command '.sect' which assigns the symbolic name used in the link command file to specify where the code will be located in memory.

Interrupt Vectors

Interrupt vectors are a series of branch instructions to the code that must be executed when the corresponding interrupt is asserted. In the following code, 'b SETUPC50' is the first instruction executed and tells the processor to branch to symbolic address 'SETUPC50' if the reset interrupt is asserted. Likewise, 'b INT21ISR' tells the processor to branch to symbolic address 'INT21ISR' if interrupt 21 is asserted.

```
***************************************************************************
* TESTLEDS.ASM V1.0    JAMES T SLADER 5/03/1996                           *
*                                                                         *
* PURPOSE;                                                                *
* Simple program designed to test a new C50 board. When running the program *
* produces a 'Knight Rider' display on the LEDs                           *
*                                                                         *
* SIGNIFICANT MODIFICATIONS;                                              *
* 1. Changed to run from a 7ws EPROM                                      *
* 2. Code modified for inclusion in PhD. Thesis                           *
*                                                                         *
***************************************************************************

    .sect "VECTORS"
***************************************************************************
*   INTERRUPT VECTORS                                                     *
***************************************************************************
    b       SETUPC50        ;RS   - External Reset              P1 (Highest)
    rete                    ;INT1 - IP FIFO full/hfull/empty    P3
    rete                    ;
    rete                    ;INT2 - ADC2 EOC                    P4
    rete                    ;
    rete                    ;INT3 - Digital Input Received      P5
    rete                    ;
    rete                    ;TINT - Internal Timer              P6
    rete                    ;
    rete                    ;RINT - Serial Port Receive         P7
    rete                    ;
    rete                    ;XINT - Serial Port Transmit        P8
    rete                    ;
    rete                    ;TRNT - TDM Port Receive            P9
    rete                    ;
    rete                    ;TXNT - TDM Port Transmit           P10
    rete                    ;
    rete                    ;INT4 - User Interrupt 4  (UART)    P11
    rete                    ;
    .space 14*16            ;RESERVED
    rete                    ;TRAP - Software Trap               N/A
    rete                    ;
    rete                    ;NMI  - Nonmaskable Interrupt       P2
    rete                    ;
    .space 4*16             ;RESERVED FOR EMULATION & TEST
    b       INTR21ISR       ;SW1  - Software Interrupt 1        N/A
    rete                    ;SW2  - Software Interrupt 2        N/A
    rete                    ;
    rete                    ;SW3  - Software Interrupt 3        N/A
    rete                    ;
```

Initialisation Routine

After the interrupt vector is taken, the initialisation routine is executed. In the code that follows, the on-chip wait state generators are configured to match the memory provided in the dual-C50 system. The final instruction causes a branch to the start of the main program.

```
    .sect "C50CFG"
***************************************************************************
*   SETUP C50 & I/O DEVICE MODES etc...                                   *
***************************************************************************
SETUPC50                    ;Start of execution after reset interrupt taken

*   !! Program, Data & IO space wait state generator setup
    ldp     #0h             ;Point to data page 0

    splk    #015h,cwsr      ;C50 Control Wait State Register - 10101
                            ;BIG = 1  - IO configured as 8k blocks
                            ;IO hi = X, IOlo = 1
                            ;Data Space = 0, Program Space = 1 (LSB)
    splk    #0955fh,pdwsr   ;C50 P/D Wait State Register - 1001 0101 0101 1111
                            ;Data memory       c000 - ffff = 2ws    (MSB)
                            ;Data memory       0000 - bfff = 1ws
                            ;Program memory    8000 - ffff = 1ws
                            ;Program memory    0000 - 7fff = 7ws    (LSB)
    splk    #0fdh,iowsr     ;IO Wait State Register - 0000 0000 1111 1101
                            ;IO Hi  8000 - ffff = Xws NOT USED      (MSB)
                            ;IO Lo  6000 - 7fff = 7ws COUNTERS
                            ;IO Lo  4000 - 5fff = 7ws UART
```

```
                                ;IO Lo  2000 - 3fff = 7ws QUAD DACs & ADC2
                                ;IO Lo  0000 - 1fff = 1ws FIFOs, LEDs, DILs etc..
    splk    #0f8h,greg          ;Define 0000 - f7ff as data memory
                                ; and   f800 - ffff as GLOBAL memory

*   !! C50 Status and Control Register Initialization
    spm     0                   ;Product Register Shift mode 0 (no shift of o/p)
    setc    sxm                 ;Set sign extension mode (2's Compliment)
    setc    ovm                 ;Set overflow mode (Overflow to maximum value)
    splk    #02eh,pmst          ;Choose C50 enhanced modes
                                ;SARAM 9k block mapped to Data space only

*   !! DEFINE 1/O LABLES
LEDS     .set     0007h         ;LEDs

*   !! Configure IO Peripherals
    splk    #00h,PA7            ;MM write to LEDs - Switch OFF all LEDs
    b       PROGINIT            ;Branch to Program initialisation code
```

Main Program

The main program in this example causes a 'walking one' pattern to be displayed on the LED bank. The reader's attention is drawn to the instruction 'intr 21' which causes interrupt 21 to be asserted. This forces the processor to execute the corresponding interrupt service routine which, in this case, simply generates a short delay.

```
    .sect "PROG"
***************************************************************************
* MAIN PROGRAM CODE                                                       *
* ???? cycles  ????? WORDS                                                *
* 50 ns = 0.?????? ms  - 40MHz    C50 Processor                           *
* 35 ns = 0.?????? ms  - 57.14MHz C50 Processor                           *
***************************************************************************
PROGINIT
    larp    ar0                 ;Set ARP
    lacl    #01h                ;Initialise display pattern
    sacl    DISPLAY             ;Set DISPLAY

START
    lar     ar0,#06h            ;Set repeat 7 times banz loop counter

LOOP1
    out     DISPLAY,LEDS        ;Set LEDs with latest pattern
    intr    21                  ;Call ISR to generate delay
    sfl                         ;Shift display pattern left by 1 bit
    sacl    DISPLAY             ;Set Display to new test pattern
    banz    LOOP1,*-,ar0        ;Repeat until AR0=0
    lar     ar0,#06h            ;Set repeat 7 times banz loop counter

LOOP2
    out     DISPLAY,LEDS        ;Set LEDs with latest pattern
    intr    21                  ;Call ISR to generate delay
    sfr                         ;Shift display pattern right by 1 bit
    sacl    DISPLAY             ;Set Display to new test pattern
    banz    LOOP2,*-,ar0        ;Repeat until AR0=0
    b       START
```

Interrupt Service Routine

The only interrupt service routine in this program generates a short delay using nested repeat loops. The reader should note that the delay produced depends upon the speed of the memory that this code executes from.

```
    .sect "ISRCODE"
****************************************************************************
*  INTERRUPT SERVICE ROUTINES                                              *
*                                                                          *
* INTR 21 - APPROX 1s delay (50ns C50 with program running in a 7ws EPROM) *
****************************************************************************

INTR21ISR
    mar     *,ar1            ;ARP=1
    lar     ar2,#04h         ;Outer loop counter (Repeat 5 times)
R1? lar     ar1,#0ffffh      ;Inner loop counter (Repeat 65536 times)
R2? banz    R2?,*-,ar1       ;Repeat until AR1=0
    larp    ar2              ;Set ARP
    banz    R1?,*-,ar1       ;Repeat until AR2=0
    larp    ar0              ;Set ARP for main program
    ret                      ;Return from subroutine

    .end
```

Link Command File

A link command file for the program follows;

```
/**************************************************************************/
/*  Default link command file for UOP C50 systems (No Debug Monitor)      */
/*  James T Slader, University of Plymouth 05/03/96                       */
/*                                                                        */
/*  Program space;  32k EPROM (3ws) & 32k external RAM (0ws)              */
/*                                                                        */
/*  Data space;     32k external RAM (0ws)                                */
/*                                                                        */
/**************************************************************************/
-o TESTLEDS.OUT                                /* Output filename       */
-m TESTLEDS.MAP                                /* Map filename          */
-e SETUPC50                                    /* Entry point           */
TESTLEDS.OBJ                                   /* Linker input filenames  */

MEMORY
{
/* PROGRAM MEMORY - Without SARAM re-mapped */
 PAGE 0:
 VECS:   origin = 0000h,  length = 030h   /* EPROM - Interrupts/Reserved   */
 EROM0:  origin = 0200h,  length = 7e00h  /* EPROM - 31.95k                */
 RAM0:   origin = 08000h, length = 8000h  /* R0 32k external RAM           */

/* DATA MEMORY    - Without SARAM re-mapped */
 PAGE 2:
 IOREG:  origin = 0050h,  length = 0010h  /*                               */
 DARAM2: origin = 0060h,  length = 0020h  /* B2 32w  on-chip DARAM         */
 RAM1:   origin = 04000h, length = 8000h  /* R1 32.0k external RAM         */
 DPRAM:  origin = 0f800h, length = 0800h  /* DP 2.0k external DP-RAM       */
}

SECTIONS
{
        VECTORS {} >VECS     PAGE 0  /* Interrupt vectors to start of EPORM */
        C50CFG  {} >EROM0    PAGE 0  /* Configuration code in EPROM         */
        PROG    {} >EROM0    PAGE 0  /* Program code to EPROM               */
        ISRCODE {} >EROM0    PAGE 0  /* ISR code to EPROM                   */
}
```

The first part of the file simply defines the memory provided in program and data space. Of most interest is the text following the 'SECTIONS' label. This specifies where the initialised sections 'VECTORS', 'C50CFG', 'PROG' and 'ISRCODE' will be placed in the memory map. Careful examination if the text reveals that all code will be placed in the EPROM. To ensure that the interrupt vectors are placed at the start of the EPROM, the author has chosen to define the first 48 locations separately.

A.3.3 Strategy for Developing Real-Time DSP Code

In this section the author describes a generic strategy used to develop reliable real-time DSP code for the dual-C50 system board. A worked example is beyond the scope of this text so only an overview is given.

The first stage, and the most important, is to define the algorithm or system that has to be implemented. This includes producing a mathematical model so that the problem is understood and can be split into manageable sections. Once the mathematics are understood, a block diagram should be constructed to show individual algorithms and their relationships with others. After several iterations, the final block diagram should represent the system with all its inputs and outputs clearly defined. At this stage it should be possible to identify those blocks which represent greater processing overhead.

The second stage is to compare the mathematical block diagram to that of the target system, in this case the dual-C50 system board. The memory requirements for each algorithm should be assessed and compared to the memory available on the target system. At this stage it should be possible to determine if the target system is capable of implementing the system. For the purpose of this discussion the reader assumes that the system requires an analogue input and an analogue or digital output to some external device. These characteristics should determine whether the system can be implemented on a single DSP system, whether it requires more than one DSP system and which is most appropriate for receiving the input and generating the output. The original block diagram should then be re-drawn so that it relates more closely to the target hardware.

The third stage is to produce a development plan, which starts with a minimal working system from the outset, and define tests that can be applied at each stage of development. For a system that takes a sampled analogue input and produces a sampled analogue output, a sensible starting point is to simply pass the input directly to the output. This type of system will be controlled by the sample rate used with the ADC and DAC, which in turn imposes the limits for real-time operation. Early development can be conducted at a lower sample rate, if necessary, before the code has been optimised. In the early stages it can often be more productive to test the code without the added complication of ensuring real-time operation.

The fourth stage is to produce working code. For this stage it is important to test each module thoroughly with all possible input conditions. Following the development plan prepared earlier, each module developed should always be added to a working system so that errors are flagged early. At the end of the fourth stage, the system should be fully implemented but not necessarily operating at the final sample rate. The final stage is to optimise the code to ensure that the system operates at the desired sample rate. This optimisation phase may include the following;

1. Optimise modules starting with the most intensive first
2. Ensure that frequently accessed code executes from fastest memory
3. Ensure that frequently accesses data is stored in fastest memory
4. Identify nested loops and minimise the instruction count for the inner loop
5. Re-order instructions to make use of the C50's Harvard architecture
6. Where possible, use loops to process data in blocks

Finally, it is important to recognise that conditional branches can cause unexpected real-time problems, especially if they are within loops. As an example, consider a module which contains a conditional branch where the two possible outcomes require 10 and 100 processor cycles respectively. If this module is repeated within a loop 1000 times, a significant range of execution times could result (10,000 processor cycles to 100,000 processor cycles). Under normal operating conditions the code may execute quickly but occasionally the worst case may be encountered causing samples to be dropped or data to be lost, all testing should consider the worst case.

B. Description of Doppler Tracking Modem and Oscilloscope Signal Plots.

B.1 Description of 9.6kbps FM Modulator

A 9.6kbps FM modulator was implemented by the author using a single TMS320C50 fixed-point DSP, a block diagram of DSP hardware and software is shown in Figure B-1. Data is applied to the modulator using an 8-bit parallel interface and its output is a frequency modulated 54 kHz I.F. For transmission to take place, additional filtering, fixed frequency up-conversion and power amplification must be applied after the modulator in the uplink path. The only other external input is the 'Doppler compensation component' which is received from a Doppler tracker (see Chapter 2).

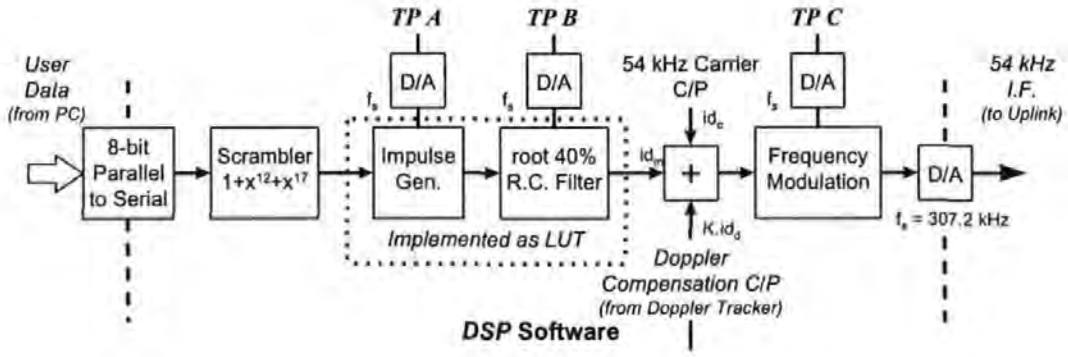


Figure B-1. 9.6kbps FM modulator (single DSP).

With reference to Figure B-1, input data is serialised and scrambled ($1+x^{12}+x^{17}$) to aid symbol timing recovery at the receiver. The sampling frequency at the output (DAC) is 307.2 kHz, so the scrambler output is converted to impulses at intervals of 32 samples to generate the 9.6kbps data rate. Frequency modulation is performed using a 'phase accumulator' technique and employs a pre-computed Cosine table for frequency synthesis. The index to the Cosine table is derived from the sum of three components; the modulating signal, a carrier frequency component and a Doppler pre-compensation component. The modulating signal, id_m, is generated by applying the data impulses to a root 40% raised cosine filter which is scaled to produce the desired frequency deviation. The carrier component, id_c, is a constant input which sets the nominal 54kHz carrier frequency. The Doppler pre-compensation component, $K.id_d$, introduces pre-compensation into the uplink path which is proportional to, but in opposition to, the Doppler shift detected in the downlink path. The frequency modulated sample stream is applied to a digital to analogue converter to generate an analogue I.F signal for transmission. Two sections of the modulator, root 40% RC filtering and frequency modulation, are described with greater detail in the main text due to the techniques employed to achieve real-time operation. Oscilloscope signal plots from test points TP A to TP C are presented in section B.1.1.

B.1.1 9.6kbps FM Modulator Signal Plots

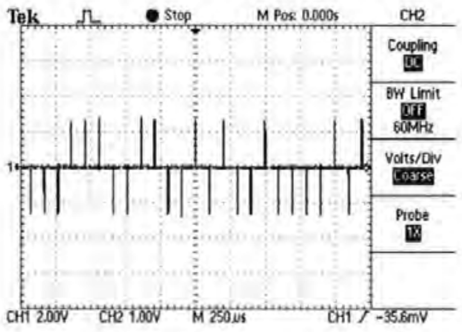


Figure B-2. FM modulator TP A - Notional mapped ±1 data at 9.6kbps input to root 40% raised Cosine filter (25 symbol periods).

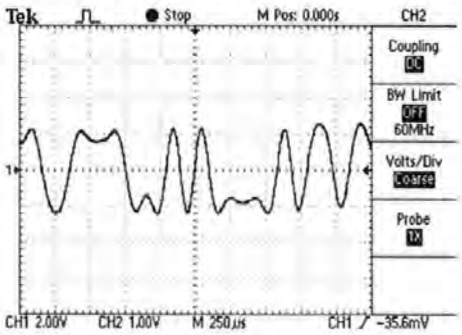


Figure B-3. FM modulator TP B - Root 40% raised cosine filtered data obtained from a look-up table (25 symbol periods).

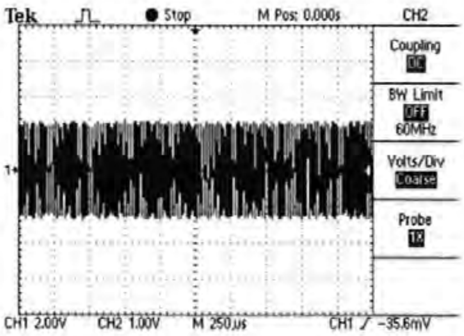


Figure B-4. FM modulator TP C - FM output at nominal 54kHz centre frequency (25 symbol periods).

B.2 Description of 9.6kbps FM Demodulator

A 9.6kbps FM Demodulator was implemented by the author using two TMS320C50 fixed-point DSPs due to the intensive nature of the associated algorithms. Block diagrams of DSP hardware and software for DSP 1 and DSP 2 are shown in Figure B-5 and Figure B-6 respectively. The demodulator algorithms are assigned so that DSP 1 performs FM demodulation and Doppler tracking while DSP 2 performs symbol timing recovery, descrambling and external interfacing.

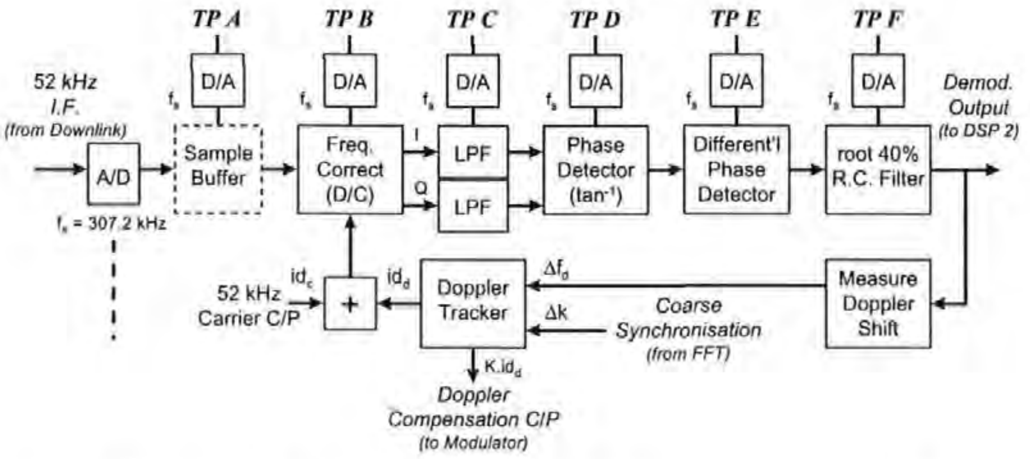


Figure B-5. 9.6kbps FM demodulator DSP 1 - FM demodulation & Doppler tracking.

With reference to Figure B-5, a 52 kHz I.F. signal is received from the downlink and applied to the demodulator hardware (DSP 1) where it is sampled at 307.2 kHz. The resulting signal samples are written to a memory buffer and the remaining processing is performed by software algorithms. Frequency correction (to baseband) is performed by mixing a synthesised local oscillator with the received signal to produce both in-phase and quadrature sample streams. The local oscillator employs a 'phase accumulator' technique and a pre-computed Cosine table. The Cosine table index is derived from the sum of two components; a carrier frequency component and a Doppler correction component. The carrier frequency component, id_c, is a constant which produces the nominal 52 kHz I.F. received from the downlink hardware, the Doppler correction component, id_d, provides fine Doppler correction and is controlled by the Doppler tracker (see Chapter 2). The frequency corrected sample sequences are low pass filtered to remove unwanted products of the mixing process and applied to a phase detector. The phase detector implements an extended (-180° to +180°) $\tan^{-1}$ function and outputs the instantaneous phase of the signal. Frequency discrimination is performed with a differential phase detector which outputs a signal proportional to the instantaneous frequency. Finally, matched root 40% raised cosine filtering is applied to maximise the data eye at the demodulator output. Un-corrected Doppler error manifests as a DC offset at the demodulator output and this offset is measured so that correction can be applied via the Doppler tracker. By comparison with a coarse Doppler estimate derived from a FFT algorithm (see Chapter 2) the Doppler tracker is immediately reset upon detection of lost frequency lock. The demodulator output is applied to DSP 2 for the remaining processing to be conducted, Figure B-6. Oscilloscope signal plots from test points TP A to TP F are presented in section B.2.1.

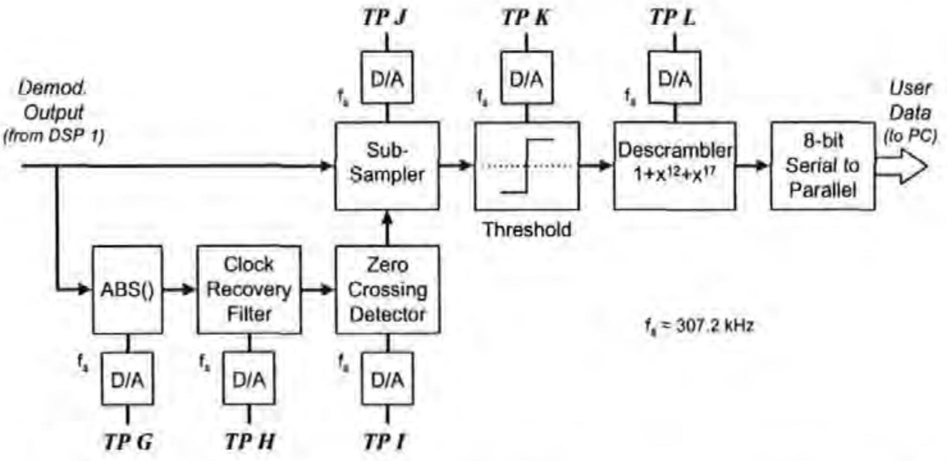


Figure B-6. 9.6kbps FM demodulator DSP 2 - Symbol timing synchronisation.

With reference to Figure B-6, the demodulator output is transferred from DSP 1 to DSP 2 and simultaneously applied to a clock recovery circuit and sub-sampler. Clock recovery is performed by a 'delay and multiply' technique which employs a narrow IIR filter and is discussed at length elsewhere in this Thesis (Burst Demodulator Chapter). The input to the clock recovery filter is derived from the demodulator output by first taking the magnitude and then introducing a fixed DC shift to generate a stimulus with strong frequency component at twice the symbol rate. The clock recovery filter 'rings' at the symbol rate and positive going zero crossings at its output indicate the optimum sub-sampling instants. After the demodulator output has been sub-sampled, the sample rate is reduced from 32 samples per symbol (at the demodulator input) to 1 sample per symbol. The original user data is restored by applying a decision threshold and with the descramber $(1+x^{12}+x^{17})$. The user data is re-assembled into bytes and transferred to the receiving PC using a parallel interface. No provision is made for byte synchronisation within the demodulator because synchronisation is conducted at a higher level by protocol software on the transmitting and receiving PCs; eg. the KISS (Keep It Simple Stupid) Protocol. Oscilloscope signal plots from test points TP G to TP L are presented in section B.2.1.

B.2.1 9.6kbps FM Demodulator Signal Plots

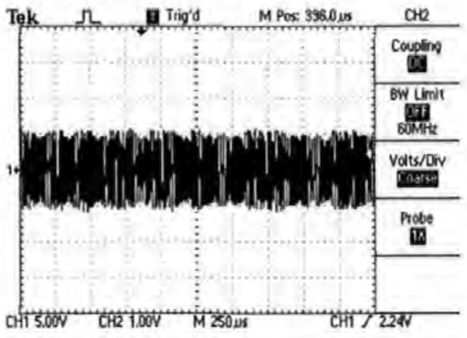


Figure B-7. FM demodulator TP A - 9.6kbps FM signal received from ADC (25 symbol periods).

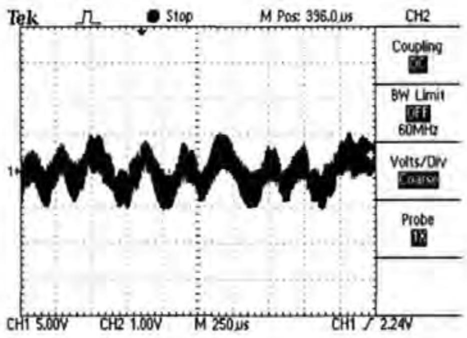


Figure B-8. FM demodulator TP B - Down converted in-phase stream prior to filtering (25 symbol periods).

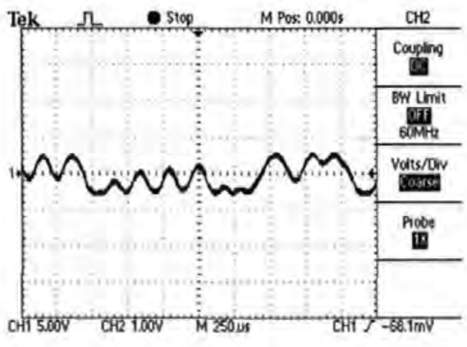


Figure B-9. FM demodulator TP C - Down converted in-phase stream after low pass filtering (25 symbol periods).

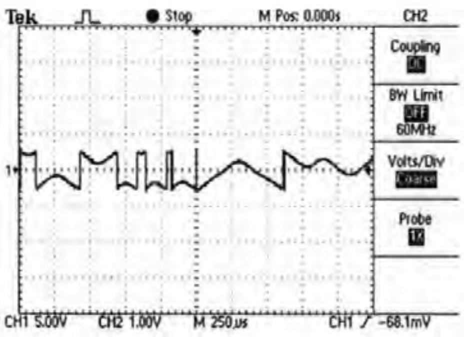


Figure B-10. FM demodulator TP D - Output from extended Tan^{-1} phase detector (25 symbol periods).

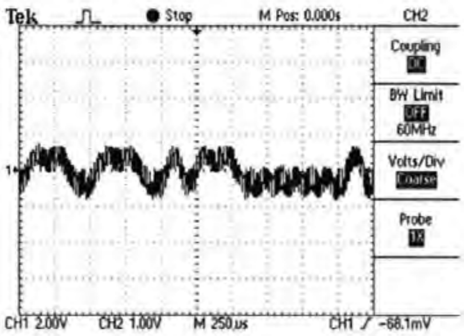


Figure B-11. FM demodulator TP E - Differential phase detector output prior to matched filtering (25 symbol periods).

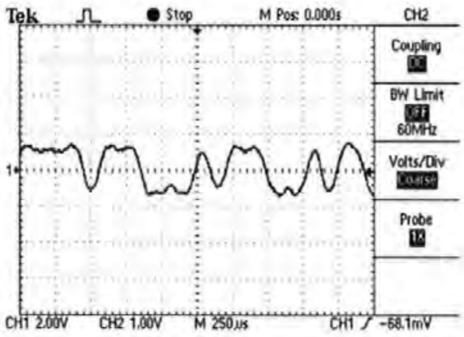


Figure B-12. FM demodulator TP F - Differential phase detector output after root 40% raised Cosine matched filter (25 symbol periods).

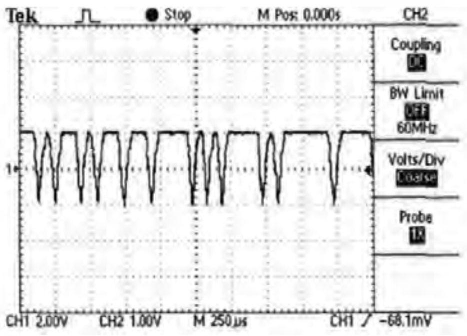


Figure B-13. FM demodulator TP G - Full wave rectified data signal stimulus for IIR clock recovery filter (25 symbol periods).

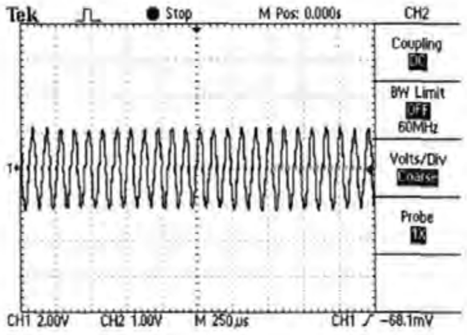


Figure B-14. FM demodulator TP H - Output from IIR clock recovery filter (25 symbol periods).

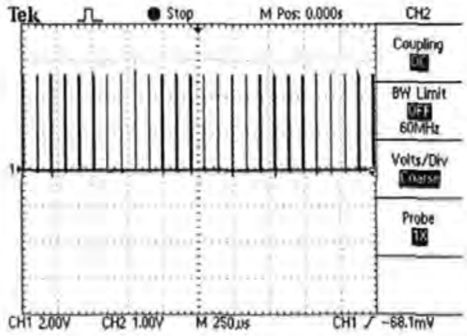


Figure B-15. FM demodulator TP I - Symbol clock positive going zero crossing detector output (25 symbol periods).

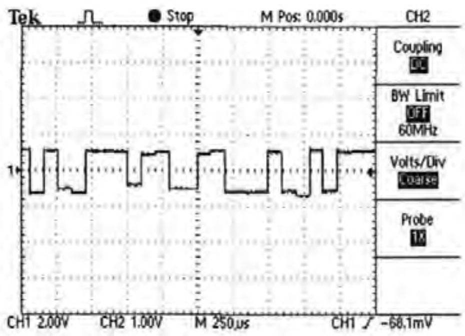


Figure B-16. FM demodulator TP J - Sampled data output prior to data decision threshold (25 symbol periods).

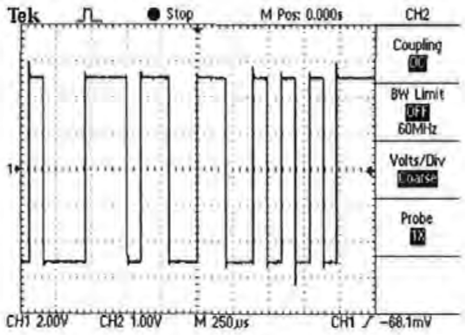


Figure B-17. FM demodulator TP K - Detected synchronous (clocked) data after decision threshold (25 symbol periods).

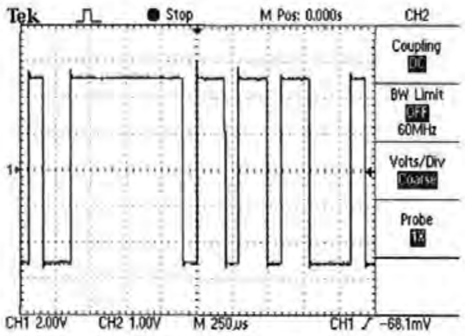


Figure B-18. FM demodulator TP L - De-scrambled data applied to output buffer (25 symbol periods).

C. Offset-FFT Derivation

The Decimation In Time (DIT) Fast Fourier Transform (FFT) algorithm is the simplest and best known. The derivation is shown below for a modified FFT algorithm known as the Offset-FFT (OFFT). Analysis of the previous discussions reveals that the 8-point Offset-FFT algorithm requires 7 (N-1 in general) unique coefficients or *twiddle factors*; W_N^{0+4c}, W_N^{0+2c}, W_N^{2+2c}, W_N^{0+c}, W_N^{1+c}, W_N^{2+c} and W_N^{3+c}. In contrast a standard FFT algorithm requires only 4 (N/2 in general) unique coefficients; W_N^0, W_N^1, W_N^2 and W_N^3. It can be easily shown that the FFT is in fact an Offset-FFT with offset c set to zero. Reviewing equations (C-23) , (C-27) , (C-31) and (C-35) , for the first *Pass* of the Offset-FFT, demonstrates the need to re-order or *'shuffle'* the input samples x_i so that the final results F(k+c), k=0,1,....,N-1 occur in their natural order.

C.1 8-point OFFT Pass 2, Group 0 (Final Pass)

An expression for the Discrete Fourier Transform (DFT) is given by

$$F(k)=\sum_{n=0}^{N-1} x_n W_N^{nk} \qquad \text{where } W_N = e^{-j2\pi/N} \quad \text{k=0,1,..,N-1} \quad \text{(C-1)}$$

and the corresponding expression for the Offset-Discrete Fourier Transform by

$$F(k+c)=\sum_{n=0}^{N-1} x_n W_N^{n(k+c)} \qquad \text{where } W_N = e^{-j2\pi/N} \quad \text{k=0,1,..,N-1} \quad \text{(C-2)}$$

Considering an 8-point OFFT (N=8) for simplicity, we may decimate x_n into odd and even sequences to give two N/2-point (4-point) DFTs. ie.

$$F(k+c)=\left[\sum_{n=0}^{\frac{N}{2}-1} x_{2n}\cdot W_N^{(2\cdot n)(k+c)}\right]+\left[\sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\cdot W_N^{(2\cdot n+1)(k+c)}\right] \text{ k=0,1,..,N-1(C-3)}$$

(n is replaced with 2n for the even term and n is replaced with 2n+1 for the odd term)

Equation (C-3) may be rewritten

$$\begin{aligned} F(k+c) &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cdot W_N^{2\cdot n(k+c)} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cdot W_N^{2\cdot n(k+c)} \cdot W_N^{k+c} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cdot W_{\frac{N}{2}}^{n(k+c)} + W_N^{(k+c)} \cdot \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cdot W_{\frac{N}{2}}^{n(k+c)} \end{aligned} \tag{C-4}$$

and expressed for convenience as

$$F(k+c) = F_e(k+c) + W_N^{(k+c)} \cdot F_o(k+c) \tag{C-5}$$

The important step in deriving an FFT algorithm is to exploit the periodicity in W_N, in particular that

$$W_{\frac{N}{2}}^{n\cdot k} = W_{\frac{N}{2}}^{n\left(k+\frac{N}{2}\right)} \tag{C-6}$$

(C-4) may therefore be rewritten

$$F(k+c+\tfrac{N}{2}) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cdot W_{\frac{N}{2}}^{n(k+c)} + W_N^{\left(k+c+\frac{N}{2}\right)} \cdot \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cdot W_{\frac{N}{2}}^{n(k+c)} \tag{C-7}$$

and expressed for convenience as

$$F(k+c+\tfrac{N}{2}) = F_e(k+c) + W_N^{\left(k+c+\frac{N}{2}\right)} \cdot F_o(k+c) \tag{C-8}$$

Also, since

$$\begin{aligned} W_N^{\left(k+c+\frac{N}{2}\right)} &= +W_N^{(k+c)} \cdot e^{-j\cdot\pi} \\ &= -W_N^{(k+c)} \end{aligned} \tag{C-9}$$

(C-8) reduces to

$$F(k+c+\tfrac{N}{2}) = F_e(k+c) - W_N^{(k+c)} \cdot F_o(k+c) \tag{C-10}$$

Combining (C-5) and (C-10) gives a pair of equations covering all k values, i.e.

$$\left.\begin{aligned} F(k+c) &= F_e(k+c) + W_N^{(k+c)} \cdot F_o(k+c) \\ F(k+c+\tfrac{N}{2}) &= F_e(k+c) - W_N^{(k+c)} \cdot F_o(k+c) \end{aligned}\right\} \text{k=0,1,..,N/2-1} \tag{C-11}$$

The advantage is that (C-11) need only be evaluated over the range 0 to N/2-1, whereas (C-8) must be evaluated over the range 0 to N-1, and this approximately halves the number of complex multiplications required. Equation (C-11) represents the *butterfly* computation used in the last pass of the OFFT, and the $W_N^{(k+c)}$ represents the coefficient or '*twiddle factor*'. Equation (C-11) is used in Table C-1 to show the butterfly computations for the last pass of an 8-point OFFT.

k	Butterfly Computation
0	$\left.\begin{aligned} F(0+c) &= F_e(0+c) + W_N^{0+c} \cdot F_o(0+c) \\ F(4+c) &= F_e(0+c) - W_N^{0+c} \cdot F_o(0+c) \end{aligned}\right\}$
1	$\left.\begin{aligned} F(1+c) &= F_e(1+c) + W_N^{1+c} \cdot F_o(1+c) \\ F(5+c) &= F_e(1+c) - W_N^{1+c} \cdot F_o(1+c) \end{aligned}\right\}$
2	$\left.\begin{aligned} F(2+c) &= F_e(2+c) + W_N^{2+c} \cdot F_o(2+c) \\ F(6+c) &= F_e(2+c) - W_N^{2+c} \cdot F_o(2+c) \end{aligned}\right\}$
3	$\left.\begin{aligned} F(3+c) &= F_e(3+c) + W_N^{3+c} \cdot F_o(3+c) \\ F(7+c) &= F_e(3+c) - W_N^{3+c} \cdot F_o(3+c) \end{aligned}\right\}$

Table C-1. Butterfly computations for Pass 2 of an 8-point Offset-FFT.

C.2 8-point OFFT Pass 1, Group 0

From (C-7) and (C-8)

$$F_e(k+c)=\sum_{n=0}^{\frac{N}{2}-1} x_{2n}\cdot W_{\frac{N}{2}}^{n(k+c)} \tag{C-12}$$

We may also decimate x_{2n} into odd and even sequences to give two N/4-point (2-point) Offset-DFTs. i.e.

$$F_e(k+c)=\left[\sum_{n=0}^{\frac{N}{4}-1} x_{4n}\cdot W_{\frac{N}{2}}^{(2\cdot n)(k+c)}\right]+\left[\sum_{n=0}^{\frac{N}{4}-1} x_{2\cdot(2n+1)}\cdot W_{\frac{N}{2}}^{(2\cdot n+1)(k+c)}\right] \tag{C-13}$$

and equation (C-13) may be rewritten

$$\begin{aligned}F_e(k+c)&=\sum_{n=0}^{\frac{N}{4}-1} x_{4n}\cdot W_{\frac{N}{2}}^{2\cdot n\cdot(k+c)}+\sum_{n=0}^{\frac{N}{4}-1} x_{4n+2}\cdot W_{\frac{N}{2}}^{2\cdot n\cdot(k+c)}\cdot W_{\frac{N}{2}}^{(k+c)}\\&=\sum_{n=0}^{\frac{N}{4}-1} x_{4n}\cdot W_{\frac{N}{4}}^{n\cdot(k+c)}+W_N^{2\cdot(k+c)}\cdot\sum_{n=0}^{\frac{N}{4}-1} x_{4n+2}\cdot W_{\frac{N}{4}}^{n\cdot(k+c)}\\&=F_e{}'(k+c)+W_N^{2\cdot(k+c)}\cdot F_o{}'(k+c)\end{aligned} \tag{C-14}$$

Using (C-6) , (C-9) and (C-14) gives another pair of equations which only need to be evaluated over the range k = 0 to N/4-1; instead of over the range k = 0 to N/2-1.

$$\left.\begin{aligned}F_e(k+c)&=F_e{}'(k+c)+W_N^{2(k+c)}\cdot F_o{}'(k+c)\\F_e(k+c+\tfrac{N}{4})&=F_e{}'(k+c)-W_N^{2(k+c)}\cdot F_o{}'(k+c)\end{aligned}\right\}\quad \text{k=0,1,..,N/4-1} \tag{C-15}$$

These equations represent the butterfly computations in Pass 1, Group 0 of an 8-point OFFT, Table C-2.

k	Butterfly Computation
0	$\left.\begin{aligned}F_e(0+c)&=F_e{}'(0+c)+W_N^{0+2\cdot c}\cdot F_o{}'(0+c)\\F_e(2+c)&=F_e{}'(0+c)-W_N^{0+2\cdot c}\cdot F_o{}'(0+c)\end{aligned}\right\}$
1	$\left.\begin{aligned}F_e(1+c)&=F_e{}'(1+c)+W_N^{2+2\cdot c}\cdot F_o{}'(1+c)\\F_e(3+c)&=F_e{}'(1+c)-W_N^{2+2\cdot c}\cdot F_o{}'(1+c)\end{aligned}\right\}$

Table C-2. Butterfly computations for Pass 1, Group 0 of an 8-point Offset-FFT.

C.3 8-point OFFT Pass 1, Group 1

From (C-7) and (C-8)

$$F_o(k+c) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cdot W_{\frac{N}{2}}^{n(k+c)} \qquad \text{(C-16)}$$

We may also decimate x_{2n} into odd and even sequences to give two N/4-point (2-point) Offset-DFTs. i.e.

$$F_o(k+c) = \left[\sum_{n=0}^{\frac{N}{4}-1} x_{4n+1} \cdot W_{\frac{N}{2}}^{(2\cdot n)(k+c)}\right] + \left[\sum_{n=0}^{\frac{N}{4}-1} x_{2\cdot(2n+1)+1} \cdot W_{\frac{N}{2}}^{(2\cdot n+1)(k+c)}\right] \qquad \text{(C-17)}$$

and equation (C-17) may be rewritten

$$\begin{aligned} F_o(k+c) &= \sum_{n=0}^{\frac{N}{4}-1} x_{4n+1} \cdot W_{\frac{N}{2}}^{2\cdot n(k+c)} + \sum_{n=0}^{\frac{N}{4}-1} x_{4n+3} \cdot W_{\frac{N}{2}}^{2\cdot n(k+c)} \cdot W_{\frac{N}{2}}^{(k+c)} \\ &= \sum_{n=0}^{\frac{N}{4}-1} x_{4n+1} \cdot W_{\frac{N}{4}}^{n(k+c)} + W_N^{2(k+c)} \cdot \sum_{n=0}^{\frac{N}{4}-1} x_{4n+3} \cdot W_{\frac{N}{4}}^{n(k+c)} \qquad \text{(C-18)} \\ &= F_e''(k+c) + W_N^{2(k+c)} \cdot F_o''(k+c) \end{aligned}$$

Using (C-6) , (C-9) and (C-18) gives another pair of equations which only need to be evaluated over the range k = 0 to N/4-1; instead of over the range k = 0 to N/2-1.

$$\left.\begin{aligned} F_o(k+c) &= F_e''(k+c) + W_N^{2(k+c)} \cdot F_o''(k+c) \\ F_o(k+c+\tfrac{N}{4}) &= F_e''(k+c) - W_N^{2(k+c)} \cdot F_o''(k+c) \end{aligned}\right\} \quad \text{k=0,1,..,N/4-1} \quad \text{(C-19)}$$

These equations represent the butterfly computations for Pass 1, Group 1 of an 8-point OFFT, Table C-3.

k	Butterfly Computation
0	$\left.\begin{aligned} F_o(0+c) &= F_e''(0+c) + W_N^{0+2c} \cdot F_o''(0+c) \\ F_o(2+c) &= F_e''(0+c) - W_N^{0+2c} \cdot F_o''(0+c) \end{aligned}\right\}$
1	$\left.\begin{aligned} F_o(1+c) &= F_e''(1+c) + W_N^{2+2c} \cdot F_o''(1+c) \\ F_o(3+c) &= F_e''(1+c) - W_N^{2+2c} \cdot F_o''(1+c) \end{aligned}\right\}$

Table C-3. Butterfly computations for Pass 1, Group 1 of an 8-point Offset-FFT.

C.4 8-point OFFT - Pass 0, Group 0

From (C-14)

$$F_e'(k+c) = \sum_{n=0}^{\frac{N}{4}-1} x_{4n} \cdot W_{\frac{N}{4}}^{n(k+c)} \tag{C-20}$$

We may decimate x_{4n} into odd and even sequences to give two N/8-point (2-point) Offset-DFTs. i.e.

$$F_e'(k+c) = \left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n)} \cdot W_{\frac{N}{4}}^{(2n)(k+c)}\right] + \left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n+1)} \cdot W_{\frac{N}{4}}^{(2n+1)(k+c)}\right] \tag{C-21}$$

and equation (C-21) may be rewritten

$$\begin{aligned} F_e'(k+c) &= \sum_{n=0}^{\frac{N}{8}-1} x_{8n} \cdot W_{\frac{N}{4}}^{2n(k+c)} + \sum_{n=0}^{\frac{N}{8}-1} x_{8n+4} \cdot W_{\frac{N}{4}}^{2n(k+c)} \cdot W_{\frac{N}{4}}^{(k+c)} \\ &= \sum_{n=0}^{\frac{N}{8}-1} x_{8n} \cdot W_{\frac{N}{8}}^{n(k+c)} + W_N^{4(k+c)} \cdot \sum_{n=0}^{\frac{N}{8}-1} x_{8n+4} \cdot W_{\frac{N}{8}}^{n(k+c)} \\ &= x_0 + W_N^{4(k+c)} \cdot x_4 \end{aligned} \tag{C-22}$$

Using (C-6) , (C-9) and (C-22) gives another pair of equations which only need to be evaluated over the range k = 0 to N/1-1; evaluated once.

$$\left.\begin{aligned} F_e'(k+c) &= x_0 + W_N^{4(k+c)} \cdot x_4 \\ F_e'(k+c+\tfrac{N}{8}) &= x_0 - W_N^{4(k+c)} \cdot x_4 \end{aligned}\right\} \tag{C-23}$$

These equations operate directly on the input samples (x_n) and represent the butterfly computations for Pass 0, Group 0 of an 8-point OFFT, Table C-4.

k	Butterfly Computation
0	$\left.\begin{aligned} F_e'(0+c) &= x_0 + W_N^{0+4c} \cdot x_4 \\ F_e'(1+c) &= x_0 - W_N^{0+4c} \cdot x_4 \end{aligned}\right\}$

Table C-4. Butterfly computation for Pass 0, Group 0 of an 8-point Offset-FFT.

C.5 8-point OFFT - Pass 0, Group 1

From (C-14)

$$F_o'(k+c)=\sum_{n=0}^{\frac{N}{4}-1} x_{4n+2}\cdot W_{\frac{N}{4}}^{n(k+c)} \tag{C-24}$$

We may decimate x_{4n} into odd and even sequences to give two N/8-point (2-point) Offset-DFTs. i.e.

$$F_o'(k+c)=\left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n)+2}\cdot W_{\frac{N}{4}}^{(2\cdot n)(k+c)}\right]+\left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n+1)+2}\cdot W_{\frac{N}{2}}^{(2\cdot n+1)(k+c)}\right] \tag{C-25}$$

and equation (C-25) may be rewritten

$$\begin{aligned}F_o'(k+c)&=\sum_{n=0}^{\frac{N}{8}-1} x_{8n+2}\cdot W_{\frac{N}{4}}^{2\cdot n(k+c)}+\sum_{n=0}^{\frac{N}{8}-1} x_{8n+6}\cdot W_{\frac{N}{4}}^{2\cdot n(k+c)}\cdot W_{\frac{N}{4}}^{(k+c)}\\ &=\sum_{n=0}^{\frac{N}{8}-1} x_{8n+2}\cdot W_{\frac{N}{8}}^{n\cdot(k+c)}+W_N^{4(k+c)}\cdot\sum_{n=0}^{\frac{N}{8}-1} x_{8n+6}\cdot W_{\frac{N}{8}}^{n(k+c)}\\ &=x_2+W_N^{4(k+c)}\cdot x_6\end{aligned} \tag{C-26}$$

Using (C-6) , (C-9) and (C-26) gives another pair of equations which only need to be evaluated over the range k = 0 to N/1-1; evaluated once.

$$\left.\begin{aligned}F_o'(k+c)&=x_2+W_N^{4(k+c)}\cdot x_6\\ F_o'(k+c+\tfrac{N}{8})&=x_2-W_N^{4(k+c)}\cdot x_6\end{aligned}\right\} \tag{C-27}$$

These equations operate directly on the input samples (x_n) and represent the butterfly computations for Pass 0, Group 1 of an 8-point OFFT, Table C-5.

k	Butterfly Computation
0	$\left.\begin{aligned}F_e'(0+c)&=x_2+W_N^{0+4\cdot c}\cdot x_6\\ F_e'(1+c)&=x_2-W_N^{0+4\cdot c}\cdot x_6\end{aligned}\right\}$

Table C-5. Butterfly computations for Pass 0, Group 1 of an 8-point Offset-FFT.

C.6 8-point OFFT - Pass 0, Group 2

From (C-18)

$$F_e''(k+c)=\sum_{n=0}^{\frac{N}{4}-1} x_{4n+1}\cdot W_{\frac{N}{4}}^{n\cdot(k+c)} \tag{C-28}$$

We may decimate x_{4n} into odd and even sequences to give two N/8-point (2-point) Offset-DFTs. i.e.

$$F_e''(k+c)=\left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n)+1}\cdot W_{\frac{N}{4}}^{(2\cdot n)(k+c)}\right]+\left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n+1)+1}\cdot W_{\frac{N}{2}}^{(2\cdot n+1)(k+c)}\right] \tag{C-29}$$

and equation (C-28) may be rewritten

$$\begin{aligned}F_e''(k+c)&=\sum_{n=0}^{\frac{N}{8}-1} x_{8n+1}\cdot W_{\frac{N}{4}}^{2\cdot n\cdot(k+c)}+\sum_{n=0}^{\frac{N}{8}-1} x_{8n+5}\cdot W_{\frac{N}{4}}^{2\cdot n\cdot(k+c)}\cdot W_{\frac{N}{4}}^{(k+c)}\\&=\sum_{n=0}^{\frac{N}{8}-1} x_{8n+1}\cdot W_{\frac{N}{8}}^{n\cdot(k+c)}+W_N^{4\cdot(k+c)}\cdot\sum_{n=0}^{\frac{N}{8}-1} x_{8n+5}\cdot W_{\frac{N}{8}}^{n\cdot(k+c)}\\&=x_1+W_N^{4\cdot(k+c)}\cdot x_5\end{aligned} \tag{C-30}$$

Using (C-6) , (C-9) and (C-30) gives another pair of equations which only need to be evaluated over the range k = 0 to N/1-1; evaluated once.

$$\left.\begin{aligned}F_e''(k+c)&=x_1+W_N^{4\cdot(k+c)}\cdot x_5\\F_e''(k+c+\tfrac{N}{8})&=x_1-W_N^{4\cdot(k+c)}\cdot x_5\end{aligned}\right\} \tag{C-31}$$

These equations operate directly on the input samples (x_n) and represent the butterfly computations for Pass 0, Group 2 of an 8-point OFFT, Table C-6.

k	Butterfly Computation
0	$\left.\begin{aligned}F_e''(0+c)&=x_1+W_N^{0+4c}\cdot x_5\\F_e''(1+c)&=x_1-W_N^{0+4c}\cdot x_5\end{aligned}\right\}$

Table C-6. Butterfly computations for Pass 0, Group 2 of an 8-point Offset-FFT.

C.7 8-point OFFT - Pass 0, Group 3

From (C-18)

$$F_o''(k+c)=\sum_{n=0}^{\frac{N}{4}-1} x_{4n+3}\cdot W_{\frac{N}{4}}^{n\cdot(k+c)} \qquad \text{(C-32)}$$

We may decimate x_{4n} into odd and even sequences to give two N/8-point (2-point) Offset-DFTs. i.e.

$$F_o''(k+c)=\left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n)+3}\cdot W_{\frac{N}{4}}^{(2\cdot n)(k+c)}\right]+\left[\sum_{n=0}^{\frac{N}{8}-1} x_{4(2n+1)+3}\cdot W_{\frac{N}{2}}^{(2\cdot n+1)(k+c)}\right] \qquad \text{(C-33)}$$

and equation (C-33) may be rewritten

$$\begin{aligned} F_o''(k+c) &= \sum_{n=0}^{\frac{N}{8}-1} x_{8n+3}\cdot W_{\frac{N}{4}}^{2\cdot n\cdot(k+c)}+\sum_{n=0}^{\frac{N}{8}-1} x_{8n+7}\cdot W_{\frac{N}{4}}^{2\cdot n\cdot(k+c)}\cdot W_{\frac{N}{4}}^{(k+c)} \\ &= \sum_{n=0}^{\frac{N}{8}-1} x_{8n+3}\cdot W_{\frac{N}{8}}^{n\cdot(k+c)}+W_N^{4(k+c)}\cdot\sum_{n=0}^{\frac{N}{8}-1} x_{8n+7}\cdot W_{\frac{N}{8}}^{n\cdot(k+c)} \\ &= x_3+W_N^{4(k+c)}\cdot x_7 \end{aligned} \qquad \text{(C-34)}$$

Using (C-6) , (C-9) and (C-34) gives another pair of equations which only need to be evaluated over the range k = 0 to N/1-1; evaluated once.

$$\left.\begin{aligned} F_o''(k+c) &= x_3+W_N^{4(k+c)}\cdot x_7 \\ F_o''(k+c+\tfrac{N}{8}) &= x_3-W_N^{4(k+c)}\cdot x_7 \end{aligned}\right\} \qquad \text{(C-35)}$$

These equations operate directly on the input samples (x_n) and represent the butterfly computations for Pass 0, Group 3 of an 8-point OFFT, Table C-7.

k	Butterfly Computation
0	$\left.\begin{aligned} F_o''(0+c) &= x_3+W_N^{0+4c}\cdot x_7 \\ F_o''(1+c) &= x_3-W_N^{0+4c}\cdot x_7 \end{aligned}\right\}$

Table C-7. Butterfly computations for Pass 0, Group 3 of an 8-point Offset-FFT.

D. Supplemental Burst Demodulator Information

An simplified overview of the burst demodulator implemented by the author is depicted in Figure D-1. It is well known that replacing BPSK (Binary Phase Shift Keying) modulation with QPSK (Quadrature Phase Shift Keying) modulation allows data throughput to be increased by a factor of two with the penalty of increased transmit power (+3dB). In principal it is possible for a burst demodulator to dynamically detect modulation format during acquisition and to avoid the need for separate demodulators. Applications anticipated for this configuration include situations where some users need to trade lower data throughput for increased reliability. The author's solution to this requirement was to simultaneously conduct demodulation and unique word frame synchronisation for both BPSK and QPSK whereupon a decision is made automatically by the unique word correlator that first indicates synchronisation. Once synchronisation has been achieved, the selected parity symbols are applied to the FEC algorithm and the alternate demodulation path is ignored. Carrier frequency acquisition, symbol detection and unique word frames synchronisation are conducted by a single TMS320C50 fixed-point DSP and ½-rate FEC is conducted by a second TMS30C50 DSP due to the intensity of the decoder algorithm. Carrier frequency acquisition, symbol timing synchronisation and a modified (optimised) Viterbi decoder algorithm are discussed at length in Chapter 3. Frequency correction (section D.2), matched filtering (section D.3), differential demodulation (section D.4), Unique Word frame synchronisation (section D.5) and software FIFO buffers (section D.6) are described below.

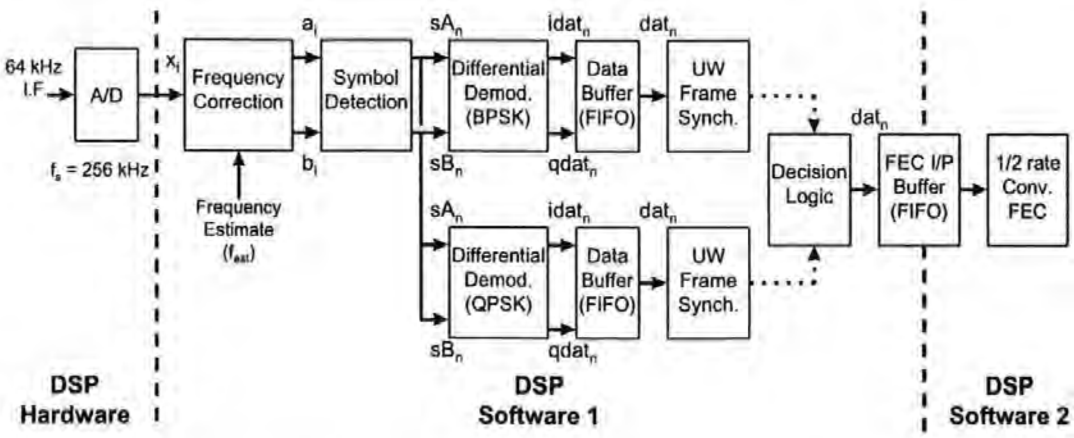
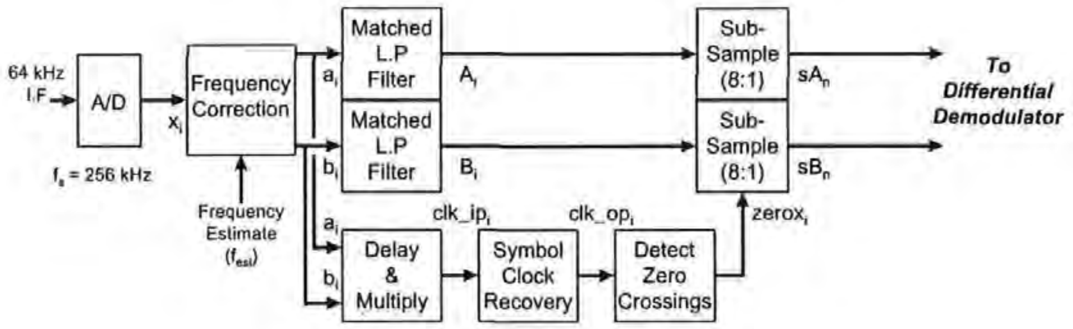


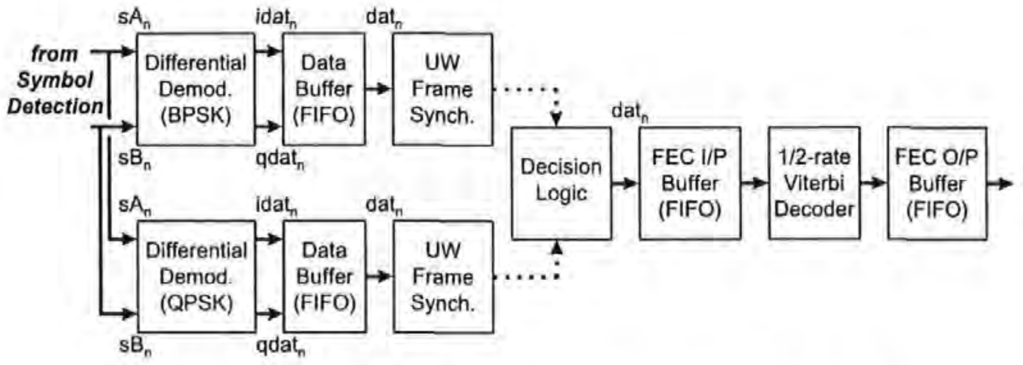
Figure D-1. BPSK/QPSK software-based DSP burst demodulator - overview.

D.1 Burst Demodulator Summary

A detailed overview of the burst demodulator is depicted in Figure D-2 and signal plots captured from each stage of the software follow in section D.1.1 for demodulation of BPSK transmissions and in section D.1.2 for demodulation of QPSK transmissions.



a. Symbol detection.



b. Differential demodulation and unique word synchronisation.

Figure D-2. BPSK/QPSK software-based DSP burst demodulator - detailed overview.

D.1.1 Signal Plots for BPSK Demodulation

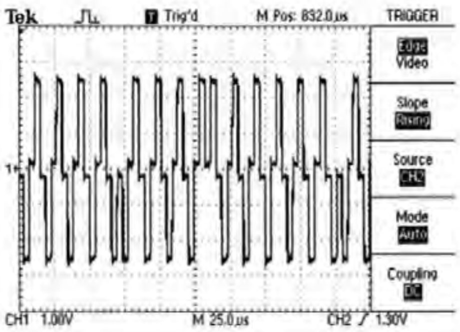
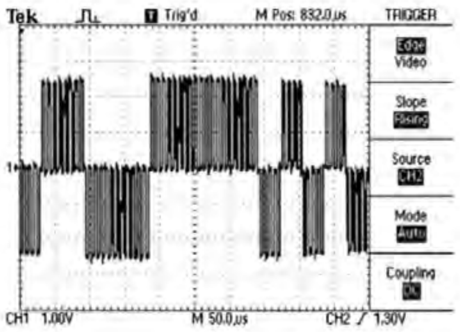
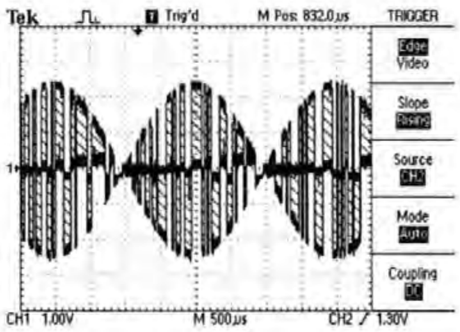


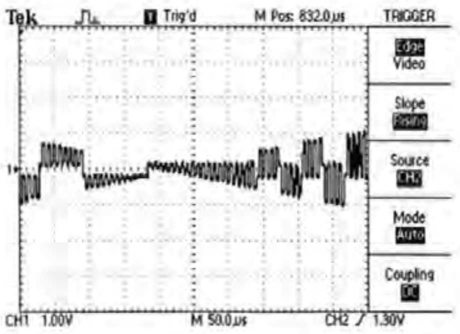
Figure D-3. Demodulator input sample stream x_i (32ksps BPSK signal, f_c=64kHz, 16 symbol periods shown).



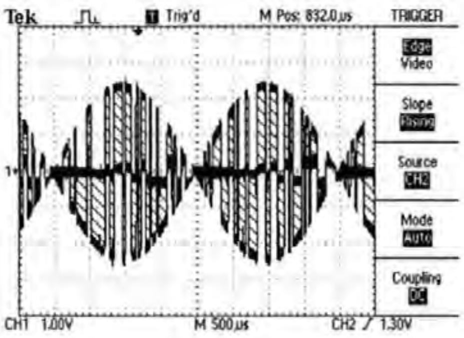
a. a_i - 16 symbol periods.



b. a_i - 160 symbol periods.

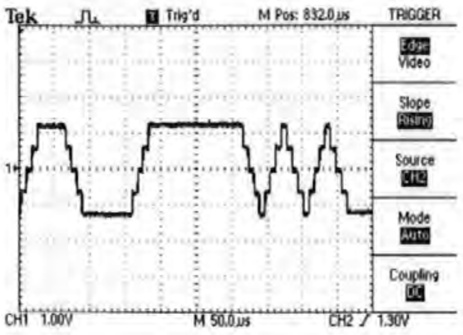


c. b_i - 16 symbol periods.

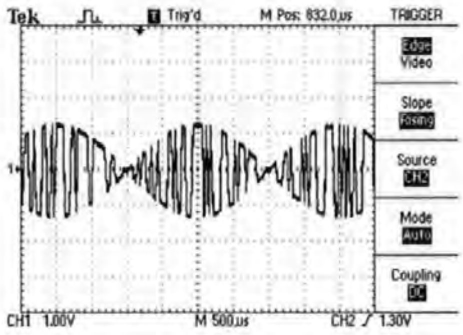


c. b_i - 160 symbol periods.

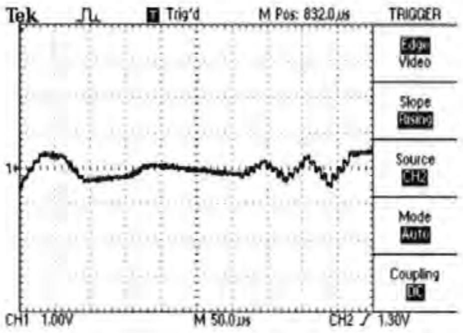
Figure D-4. Frequency corrected sample streams a_i+jb_i (250Hz residual frequency error, 16 & 160 symbol periods shown).



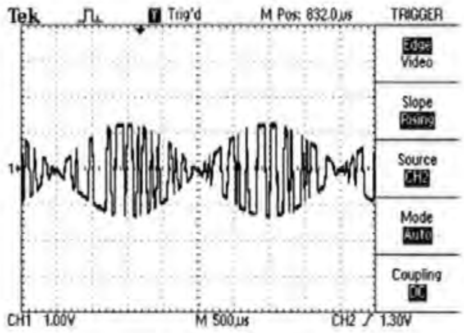
a. A_i - 16 symbol periods.



b. A_i - 160 symbol periods.

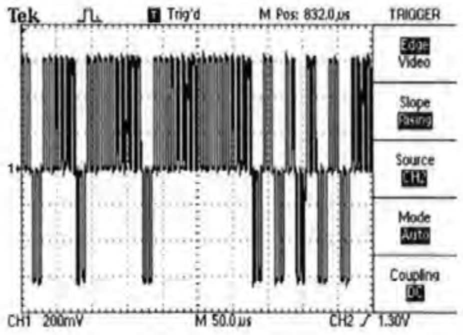


c. B_i - 16 symbol periods.

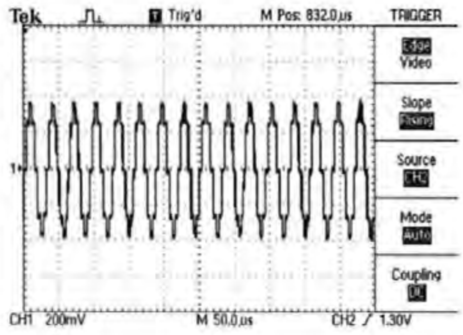


d. B_i - 160 symbol periods.

Figure D-5. Matched filtered frequency corrected sample streams A_i+jB_i (250Hz residual frequency error, 16 & 160 symbol periods shown).

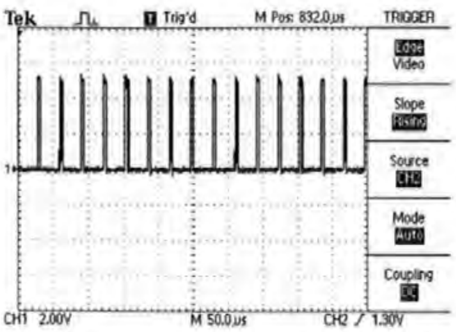


a. 'Delay & multiply' stimulus - clk_ip$_i$

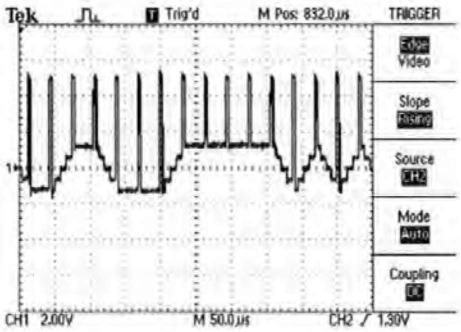


b. Recovered symbol clock - clk_op$_i$.

Figure D-6. Symbol clock recovery filter signal samples clk_ip$_i$ and clk_op$_i$ (250Hz residual frequency error, 16 symbol periods shown).

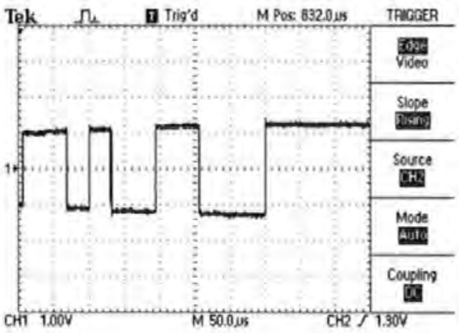


a. +ve zero crossings.

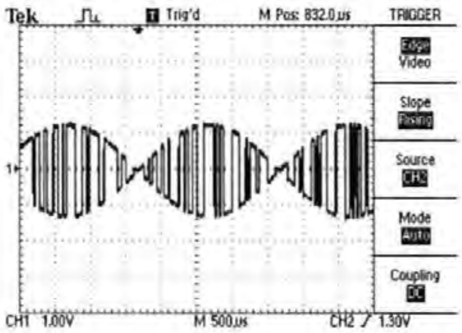


b. in relation to matched filtered data

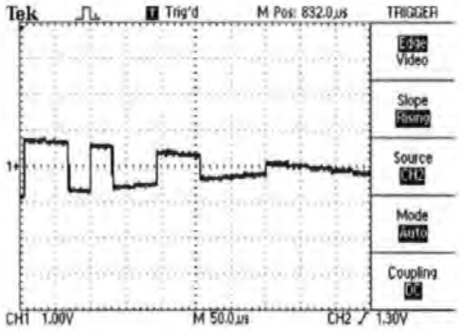
Figure D-7. Location of +ve zero crossings in recovered symbol clock $zerox_i$ (16 symbol periods shown).



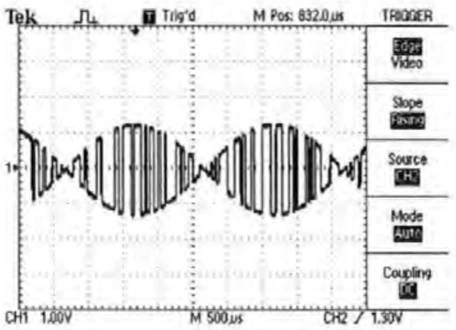
a. sA_n - 16 symbol periods.



b. sA_n - 160 symbol periods.

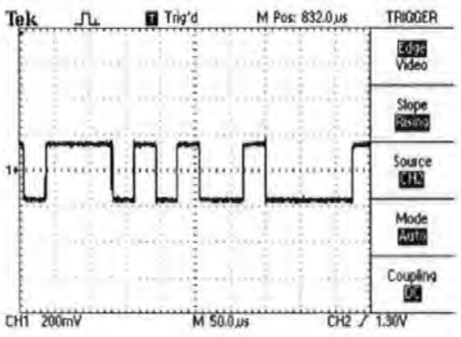


c. sB_n - 16 symbol periods.

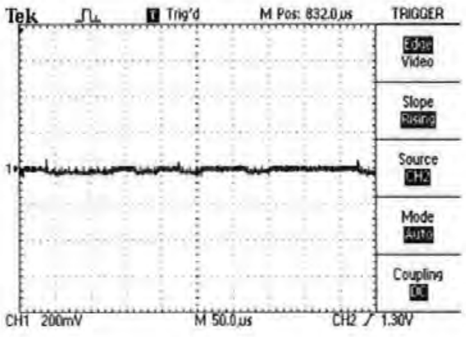


d. sB_n - 160 symbol periods.

Figure D-8. Detected symbols sA_n+j sB_n (250Hz residual frequency error, 16 & 160 symbol periods shown).

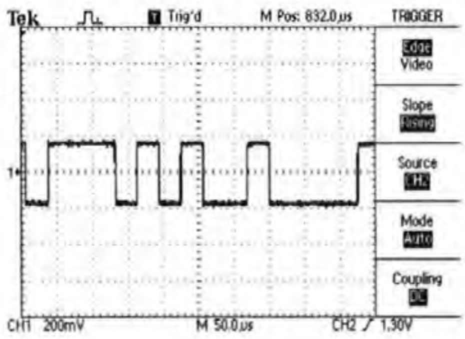


a. $idat_n$ - wanted data.

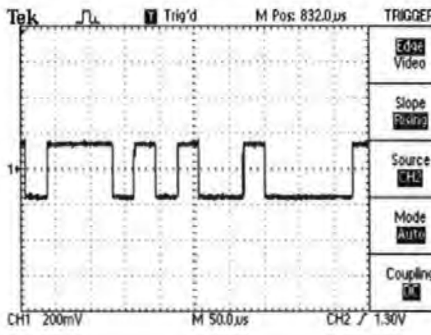


b. $qdat_n$- degradation due to f_{error}.

Figure D-9. BPSK differential demodulator output $idat_n$+j$qdat_n$ (BPSK signal transmitted, 250Hz residual frequency error, 16 symbol periods shown).



a. $idat_n = qdat_i$ - suggests BPSK i/p.



b. $qdat_n = idat_i$ - suggests BPSK i/p.

Figure D-10. QPSK differential demodulator output $idat_n$+j$qdat_n$ (BPSK signal transmitted, 250Hz residual frequency error, 16 symbol periods shown).

D.1.2 Signal Plots for QPSK Demodulation

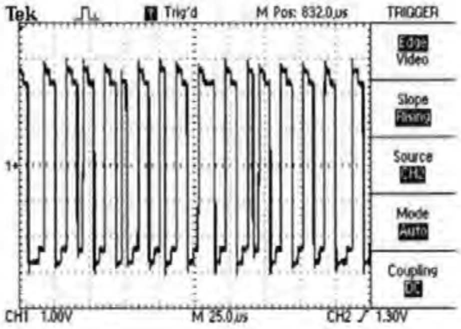
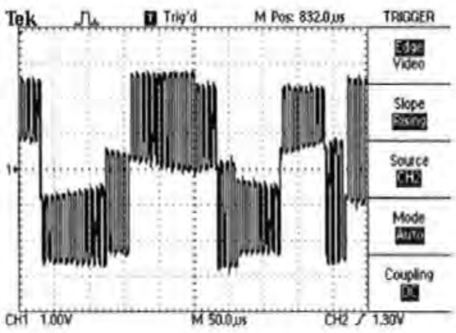
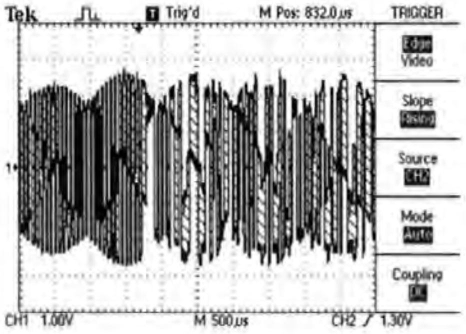


Figure D-11. Demodulator input sample stream x_i

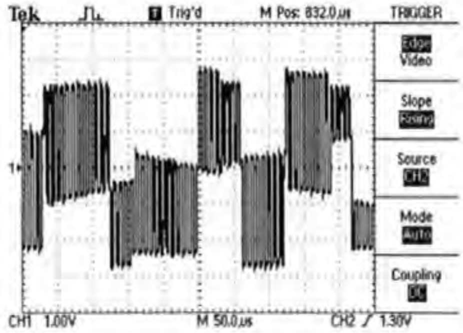
(32ksps QPSK signal, f_c=64kHz, 16 symbol periods shown).



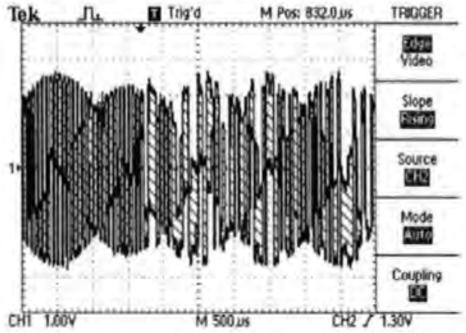
a. a_i - 16 symbol periods.



b. a_i - 160 symbol periods.

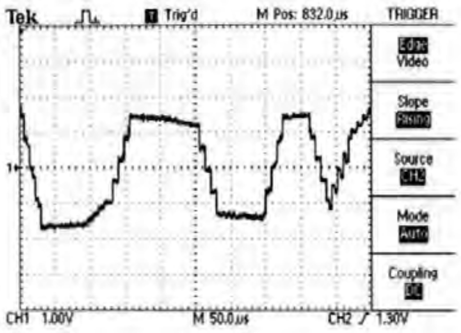


c. b_i - 16 symbol periods.

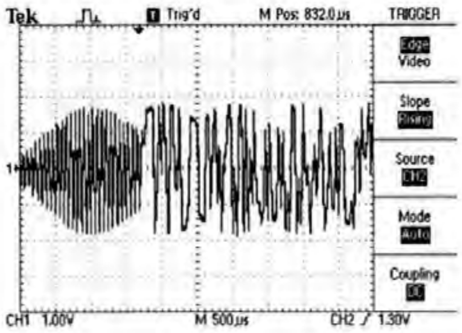


d. b_i - 160 symbol periods.

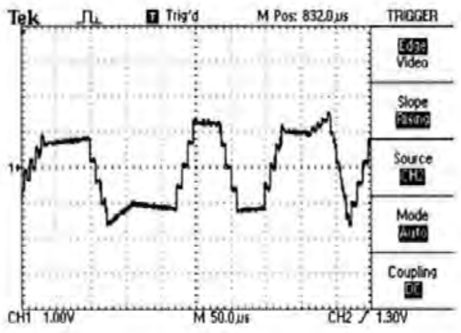
Figure D-12. Frequency corrected sample streams a_i+jb_i (250Hz residual frequency error, 16 & 160 symbol periods shown).



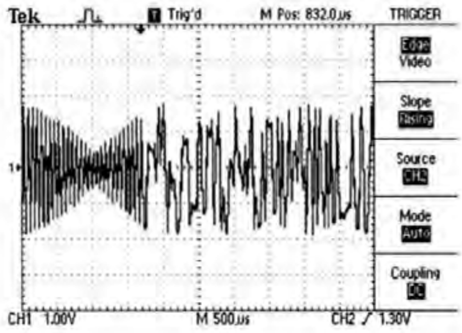
a. A_i - 16 symbol periods.



b. A_i - 160 symbol periods.

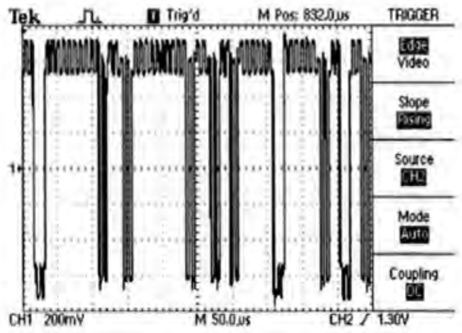


c. B_i - 16 symbol periods.

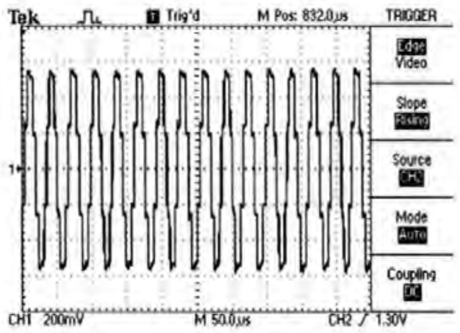


d. B_i - 160 symbol periods.

Figure D-13. Matched filtered frequency corrected sample streams A_i+jB_i (250Hz residual frequency error, 16 & 160 symbol periods shown).

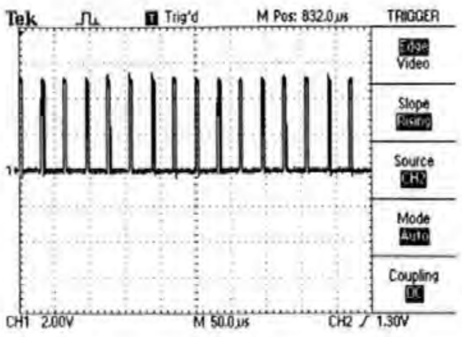


a. 'Delay & multiply' stimulus - clk_ip$_i$

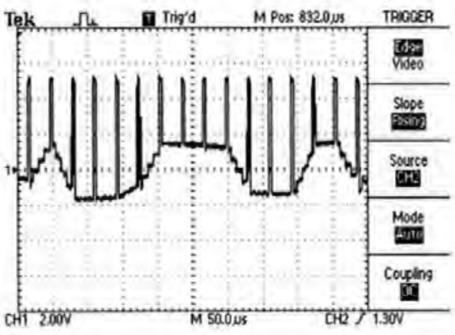


b. Recovered symbol clock - clk_op$_i$.

Figure D-14. Symbol clock recovery filter signal samples clk_ip$_i$ and clk_op$_i$ (250Hz residual frequency error, 16 symbol periods shown).

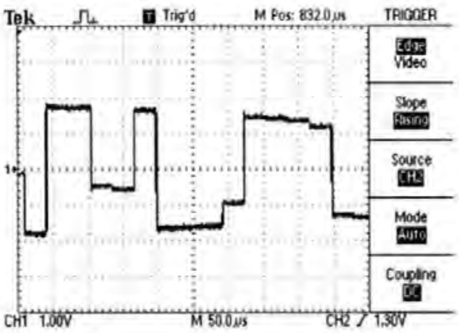


a. +ve zero crossings.

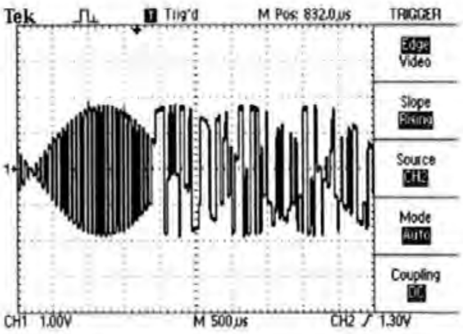


b. in relation to matched filtered data

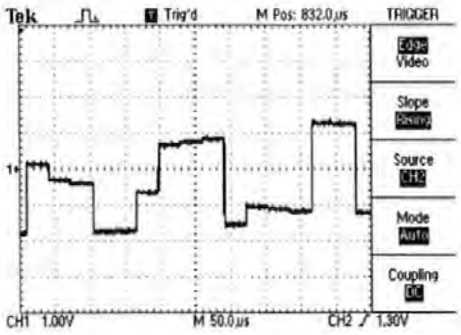
Figure D-15. Location of +ve zero crossings in recovered symbol clock zerox$_i$ (250Hz residual frequency error, 16 symbol periods shown).



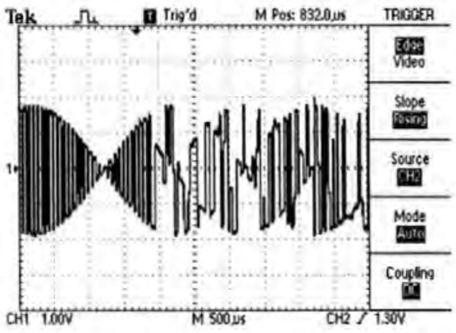
a. sA_n - 16 symbol periods.



b. sA_n - 160 symbol periods.

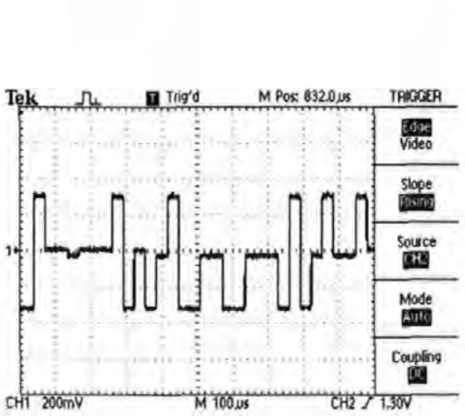


c. sB_n - 16 symbol periods.

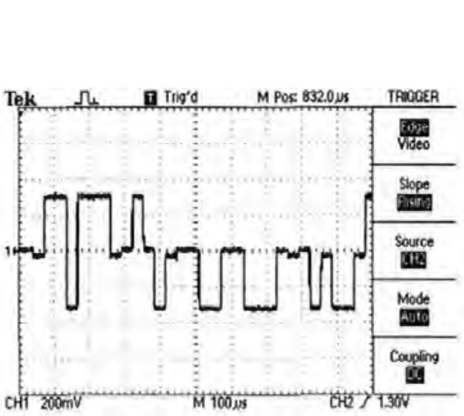


d. sB_n - 160 symbol periods.

Figure D-16. Detected symbols sA_n+j sB_n (250Hz residual frequency error, 16 & 160 symbol periods shown).

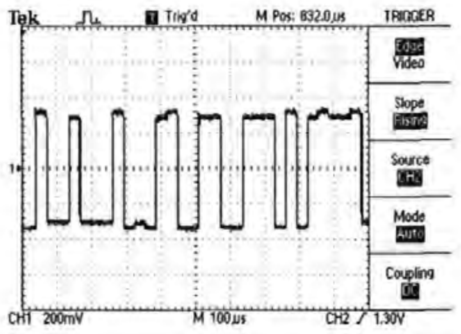


a. $idat_n$ - null bits indicate QPSK i/p.

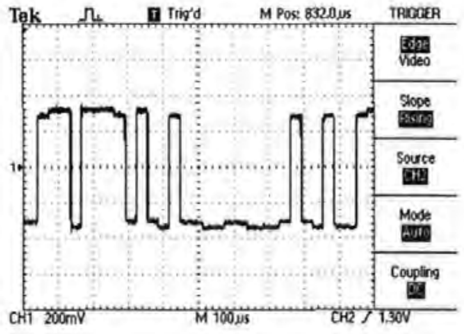


b. $qdat_n$ - null bits indicate QPSK i/p.

Figure D-17. BPSK differential demodulator output $idat_n$+j$qdat_n$ (QPSK signal transmitted, 250Hz residual frequency error, 32 symbol periods shown).



a. $idat_n$ - degradation due to f_{error}.



b. $qdat_n$ - degradation due to f_{error}.

Figure D-18. QPSK differential demodulator output $idat_n$+j$qdat_n$ (QPSK signal transmitted, 250Hz residual frequency error, 32 symbol periods shown).

D.2 Frequency Correction

Frequency correction is performed once carrier frequency acquisition has been declared. The term 'frequency correction' is used in place of the term 'demodulation' because the local oscillator is automatically tuned to the incoming signal. For these investigations frequency correction was required to match carrier frequency acquisition in terms of frequency range and resolution. Frequency correction is first described mathematically and the author's DSP implementation is described with respect to software optimisations.

D.2.1 Mathematical Analysis

The input sample sequence x_i is frequency corrected to form the complex sample sequence $a_i + jb_i$ by the carrier frequency estimate f_{est} as shown;

$$a_i + jb_i = x_i \cdot e^{-j\cdot\frac{2\cdot\pi\cdot f_{est}\cdot i}{f_s}} \qquad \text{(D-1)}$$

$$a_i + jb_i = \left[x_i \cdot \cos\left(\frac{2\cdot\pi\cdot f_{est}\cdot i}{f_s}\right)\right] - j\cdot\left[x_i \cdot \sin\left(\frac{2\cdot\pi\cdot f_{est}\cdot i}{f_s}\right)\right] \qquad \text{(D-2)}$$

The input sample sequence x_i, for the phase reversing preamble with carrier frequency f_c and BPSK modulation may be expressed

$$x_i = s_i \cdot \cos\left(\frac{2\cdot\pi\cdot f_c\cdot i}{f_s} + \phi\right) \qquad \text{(D-3)}$$

where $s_i = 1$ for the range $i = 0,1,\ldots,7$ and $-s_{i-8}$ elsewhere, the phase term ϕ indicates that the carrier has unknown phase. From equations (D-2) and (D-3) expressions for the frequency corrected sample sequences a_i and b_i may be derived

$$a_i = +\frac{s_i}{2}\cdot\left[\cos\left(\frac{2\cdot\pi\cdot(f_c - f_{est})\cdot i}{f_s} + \phi\right) + \cos\left(\frac{2\cdot\pi\cdot(f_c + f_{est})\cdot i}{f_s} + \phi\right)\right] \qquad \text{(D-4)}$$

$$b_i = -\frac{s_i}{2}\cdot\left[\sin\left(\frac{2\cdot\pi\cdot(f_c - f_{est})\cdot i}{f_s}+\phi\right)+\sin\left(\frac{2\cdot\pi\cdot(f_c + f_{est})\cdot i}{f_s}+\phi\right)\right] \qquad \text{(D-5)}$$

Equations (D-4) and (D-5) show that the frequency corrected sample sequences a_i and b_i contain an unwanted high frequency component (f_c + f_{est}) which is removed using a low pass filter. In the presence of residual frequency error, when f_{est} and f_c are not equal, the wanted data signal modulates a low frequency carrier (f_c- f_{est}) Hz with unknown phase offset ϕ. If residual frequency error is eliminated, f_{est} and f_c equal, only the wanted data signal remains. It is shown in Chapter 3 for a 256-point Offset-FFT and offset = 0.25 that the highest residual frequency error is one quarter the FFT frequency bin spacing f_s/N Hz.

D.2.2 DSP Implementation

A fundamental element of frequency correction is a variable frequency local oscillator with the desired frequency resolution. For digital signal processing this is often achieved using a 'phase accumulator' concept from which sine and cosine signals can be easily derived. This concept is extended for optimum performance, for minimum processing overhead, with a pre-computed data table consisting of a sampled cycle of a sinusoidal signal. For these investigations a frequency resolution of 250Hz was required and, with a sample rate f_s=256kHz, the cosine table therefore requires 1024 (256,000/250) entries. The addressing increment (index) required to synthesise carrier frequency f_{est} is given by

$$index = 1024\cdot\frac{f_{est}}{f_s} \qquad \text{(D-6)}$$

Both cosine and sine components are required for frequency correction and this is achieved with two pointers which address the cosine table with 90° (256-location) offsets. Each pointer must address the cosine table in a modulo 1024 manner, and by locating the data table at specific memory addresses processing overhead can be significantly reduced. The optimisation requires the table to begin at an address which is a multiple of twice the table length, 2048 in this case, and assumes that the table will not be addressed with an index larger than its length. The optimisation exploits the principal that resetting bit-11 after each increment of the pointer performs the desired modulo 1024 function. With more general 're-locatable code', the equivalent requires a decision to determine if the table's end address has been exceeded by the pointer, and the subtraction of a constant if it has. With the TMS320C50 DSP's instruction set, the optimised case requires a single instruction to implement a modulo 1024 operation while the general case requires several instructions.

Figure D-19 shows the authors main TMS320C50 assembly code for implementing frequency correction and Figure D-20 shows a block diagram most closely representing the software. Five pointers are used, AR0 to AR4, which address the x_i, a_i and b_i sample buffers and the cosine and sine table respectively. The code consists of a loop which is repeated 256 times in order to minimise the overhead associated with saving and restoring the context for each pointer. To be perfectly clear, 256 'x_i' samples are applied to the input of this algorithm and 256 samples of both 'a_i' and 'b_i' produced at its output. The two lines highlighted in Figure D-19 implement modulo 1024 addressing based upon the optimisation discussed above.

```
FREQCORRECTION .macro
**************************************************************************
* Purpose; Mix real signal with cos and -sin to generate complex OP       *
*                                                                         *
* Example macro call;                                                     *
*   FREQCORRECTION          ;MACRO - Quadrature Downconverter             *
*                                                                         *
* Approx number of cycles (code in SARAM, data in SARAM);                 *
*   12 + (N * 7)                                                          *
**************************************************************************
    ldp     #0              ;Must be in data page 0
    mar     *,ar0           ;ARP=0

*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*!! AR0, AR1, AR2, AR3, AR4, INDX & DBMR should already be initialised
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    spm     1               ;Product shift mode = 1 (multiply PREG o/p by 2)
                            ;* CAN'T OVERFLOW SINCE COS table NEVER reaches 8000h

    splk   #(N-1),brcr      ;Set to repeat 256 times
    rptb    QUADDC?-1       ;Start of repeat block
     lt      *+,ar1          ;Xi ¯ TREG
     mpy     *0+,ar3         ;Xi COS()/2 ¯ PREGh
     sph     *+,ar2          ;ai := Xi COS()  - (due to PSM)
     mpy     *0+,ar4         ;Xi -SIN()/2 ¯ PREGh
     sph     *+,ar0          ;bi := Xi -SIN() - (due to PSM)
     apl     ar1             ;Mod(DBMR) AR1 - 2 PLP Rqrd. (reset bit)
     apl     ar2             ;Mod(DBMR) AR2 - 2 PLP Rqrd. (reset bit)
QUADDC?                     ;End of repeat block

    spm     0               ;PLP & Set product shift mode back to 'NO SHIFT'
    nop                     ;PLP for apl - Ensure AR2 update happens

    .endm
```

Figure D-19. Frequency correction TMS320C50 assembly code.

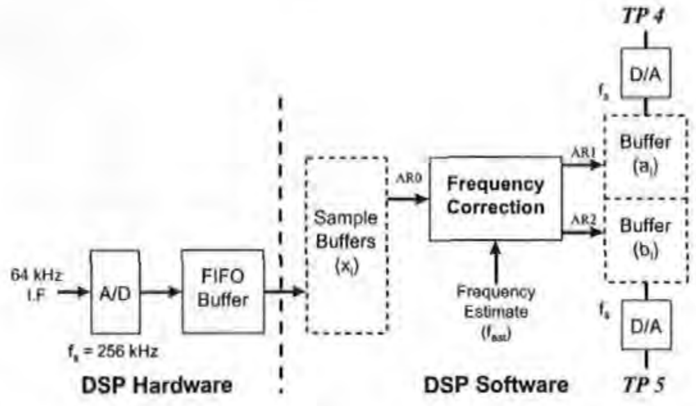


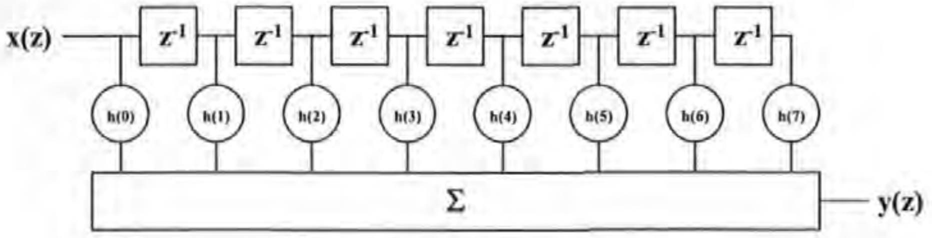
Figure D-20. DSP burst demodulator implementation - frequency correction.

D.3 Matched Filtering

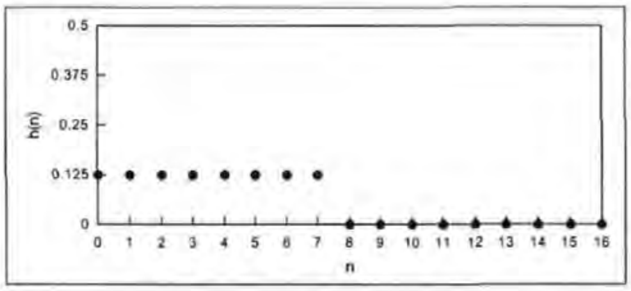
After frequency correction the resulting sample streams a_i and b_i are given by equations (D-4) to (D-5) . Two mechanisms are needed if the optimum data signal is to be recovered; low pass filtering and matched data filtering. Low pass filtering is required to remove unwanted high frequency components and a filter matched to the transmitted data is required to maximise the data eye at the sampling instants. In terms of DSP software implementation filtering can be computation intensive and only computation efficient solutions were considered. A means of combining the low pass filter and matched data filter is described mathematically and in terms of DSP implementation below.

D.3.1 Mathematical Analysis

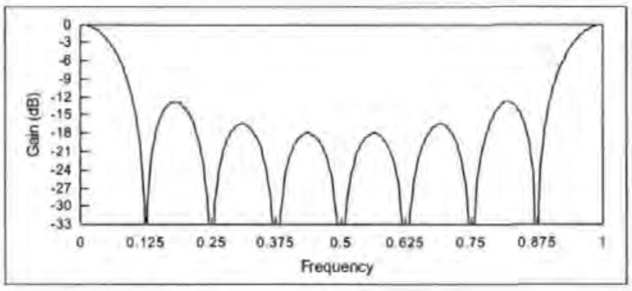
For these investigations the transmitted data is unshaped (square pulses) and the optimum receive filter is therefore an integrator. The matched filter for square data pulses at normalised symbol rate 0.125 (8 samples per symbol) is shown in Figure D-21a as a finite impulse response (FIR) structure. For fixed-point DSP implementation it is necessary to scale the filter coefficients to prevent numerical overflow, Figure D-21b.



a. FIR filter structure.



b. Impulse response.



c. Frequency response.

Figure D-21. Matched data filter (integrator).

As would be expected, this filter has a low-pass characteristic and gives 3dB attenuation at approximately $0.125 \times f_s/2$, Figure D-21c. Referring to equations (D-4) and (D-5) , the unwanted high frequency components appear at frequency (f_c+f_{est}). For the nominal carrier frequency f_c = 64kHz and a carrier frequency estimates of either f_{est} = 64.25kHz or f_{est} = 63.75kHz it can be seen that 12dB or greater attenuation will be provided. From these results it can be seen that the filter in Figure D-21a can be used to provide a computation efficient, but non-optimal, solution to both matched data and low-pass filtering.

D.3.2 DSP Implementation

Figure D-22 shows the Author's main TMS320C50 assembly code implementation of the matched filter. The code is not a standard FIR filter 'multiply and accumulate' implementation and exploits the fact that filter coefficients h(0) to h(7) are identical to reduce the instruction count; significant because the routine must be executed twice. Two memory pointers, AR0 and AR1, are used to reference the input samples and pointer AR2 is used to write the filter output samples to a new memory buffer. The FIR filter is implemented with a 'running average' technique whereby seven input samples are initially placed in the accumulator and pointers AR0 and AR1 offset by seven locations. In the main loop (highlighted instructions) the 8th sample is added to the accumulated total (AR0) and the result saved in the output buffer (AR2). The oldest sample is then subtracted (AR1) and the loop repeated. Scaling to prevent numerical overflow and pointer increments do not generate additional. overhead. It should be noted that 256 filtered samples are written to the output buffer and that 256+7 samples are required at the input, additional code (not shown) maintains the extra samples at the input buffer.

```
MATCHFLTR .macro IP,OP,N0,SCALE
**************************************************************************
* Purpose; Lowpass Integrate & Dump Matched Filter                        *
*                                                                         *
* Example macro call;                                                     *
*   MATCHFLTR IP,OP,N0,SCALE;MACRO - Integrate & dump Matched Filter      *
**************************************************************************
    ldp     #0                 ;Must be in data page 0
    mar     *,ar0              ;ARP=0

*   Set pointers
    lar     ar0,#IP            ;AR0 points to input
    lar     ar1,#IP            ;AR1 points to input (trailing pointer)
    lar     ar2,#OP            ;AR2 points to output

*   Start a pipeline with N0-1 values in ACC - AR0
    zap                        ;ACC=0
    rpt     #(N0-2)            ;Repeat next instruction N0-1 times
     add     *+,SCALE          ;Add to ACCh (scale by N0)

*   Continue the pipeline until N outputs have been generated
    splk    #(N-1),brcr        ;Repeat block N times
    rptb    MF?-1              ;Start of repeated block
     add     *+,SCALE,ar2       ;Add newest value (scaled)
     sach    *+,0,ar1           ;Write next output
     sub     *+,SCALE,ar0       ;Sub oldest value (scaled)
MF?                            ;End of repeated block
        .endm
```

Figure D-22. Matched filter TMS320C50 assembly code.

Figure D-23 shows a block diagram of the burst demodulator, which most closely represents the author's DSP implementation. For clarity, detail from earlier sections of the demodulator have been omitted. The frequency correction algorithm writes its outputs to separate staging buffers which, in turn, are applied to the matched FIR filters. Each filter output is written to a further staging buffer which provides an opportunity to monitor the filtered signal samples at the test points TP6 and TP7.

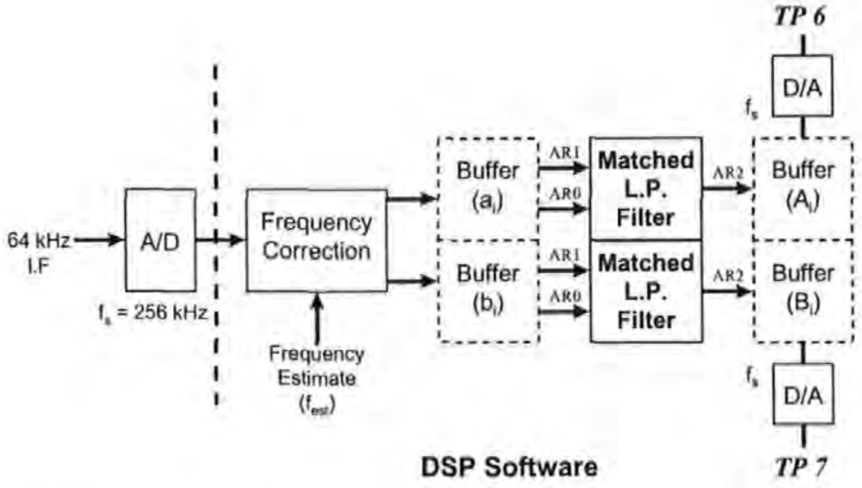


Figure D-23. DSP burst demodulator implementation - matched filtering.

D.4 Differential Demodulation

A characteristic of the burst demodulator produced during these investigations is a relatively high residual frequency error after frequency correction in comparison with the transmitted symbol rate; 250Hz and 32ksps respectively. Differential encoding is applied at the transmitter to compensate for phase ambiguity associated with non-coherent reception; information is conveyed in the phase change between transmitted symbols. Residual frequency error manifests as a linearly increasing phase error in the received data, with a relatively low phase error differential demodulation will still be possible. For the transmitted symbols to be recovered, the sample rate must be reduced from 8 samples per symbol to 1 sample per symbol. The sub-sampling process is controlled by a local symbol clock (see Chapter 3) which accurately identifies the instants at which sub-sampling should occur. Differential demodulation (data recovery) occurs once sub-sampling has been applied. In this section the author presents a mathematical overview of differential decoding and details of its implementation in the DSP burst demodulator software.

D.4.1 Mathematical Analysis

The maximum residual frequency error f_{err_max} for carrier frequency acquisition with an N-point Offset-FFT, frequency offset *off* and sample rate f_s is given by

$$f_{err\_max} = off \cdot \frac{f_s}{N} Hz \tag{D-7}$$

and with N = 256, f_s = 256 kHz and off = 0.25, f_{err_max} = 0.25 kHz. The residual frequency error leads to a linear phase variation over the period of each symbol causing a degradation in the differential phase detector. The maximum phase change from one symbol to the next ϕ_{err_max}, caused by the maximum residual frequency error f_{err_max}, is given in degrees by

$$\phi_{err\_max} = \frac{360° \cdot f_{err\_max}}{f_{symbol}} \tag{D-8}$$

where f_{symbol} is the transmitted symbol rate. With f_{err_max} = 0.25 kHz and f_{symbol} = 32,000 Hz, ϕ_{err_max} = 2.8125°. The matched filtered sample sequences A_i and B_i are sub-sampled at instants determined by the recovered symbol clock to detect the incoming symbols. Differential demodulation is achieved with a complex multiplication of samples representing the current symbol and complex conjugate of the previous symbol. For BPSK, the demodulated data is given by

$$data = \mathrm{Re}\left[-\left(A_i + jB_i\right) \cdot \left(A_{i-8} - jB_{i-8}\right)\right] \tag{D-9}$$

The effect of the maximum phase error on differentially demodulated data is illustrated in Table D-1 where the first column represents raw data prior to differential encoding and the second shows the differentially encoded symbols which will be transmitted. For BPSK modulation, both the raw data and the transmitted symbol have only in-phase components. The third column shows a linear phase error of 2.8125° per symbol which is effectively applied to transmitted symbols in the event of maximum residual frequency error. The final columns show, as a result of the phase error, the symbol which is detected after sub-sampling and differential detection. A comparison of the first and last columns indicates that differential decoding has been degraded by the phase error, and further investigation reveals that each detected symbol has been subject to a 2.8125° phase shift. Since the maximum phase error is relatively small, and the corresponding performance degradation is minimal, further processing overhead to correct this phase shift is not justified.

Raw Data	Transmitted Symbol	Phase Error	Detected Symbol	Differentially Demodulated Data
	1+j0	*0°*	*1.0000 + j0.0000*	
1+j0	-1+j0	2.8125°	-0.9988 - j0.0491	0.9988 + j0.0491
1+j0	1+j0	5.6250°	0.9952 + j0.0980	0.9988 + j0.0491
1+j0	-1+j0	8.4375°	-0.9892 - j0.1467	0.9988 + j0.0491
1+j0	1+j0	11.2500°	0.9808 + j0.1951	0.9988 + j0.0491
1+j0	-1+j0	14.0625°	-0.9700 - j0.2430	0.9988 + j0.0491
1+j0	1+j0	16.8750°	0.9569 + j0.2904	0.9988 + j0.0491

Table D-1. Effect of linear phase error on differentially detected BPSK symbols.

D.4.2 DSP Implementation

Figure D-24 shows a block diagram of the burst demodulator most closely representing the author's DSP implementation, detail from early stages of the demodulator have been omitted for clarity and to emphasise the stages most closely linked associated with differential detection. In Figure D-24, the input sample sequence x_i is frequency corrected to form the frequency corrected sample sequences a_i and b_i. The frequency corrected sample sequences are applied simultaneously to low pass matched data filters and to the symbol clock recovery sub-system. The clock recovery sub-system identifies the optimum instants at which to sub-sample the filtered data signals A_i and B_i with the sample sequence $zerox_i$. This has the effect of reducing the sample rate from 8 samples per symbol to 1 sample per symbol to form the sample sequences sA_i and sB_i in preparation for differential demodulation. Differential demodulation produces two outputs, $idat_i$ and $qdat_i$, representing in-phase and quadrature components respectively. For BPSK, the wanted demodulated data is represented by $idat_i$, and is written to a software FIFO buffer ready for further processing. Under ideal circumstances $qdat_i$ is zero and to generate it represents an unnecessary processing overhead. As already shown, this is not the case when phase error is present in the demodulator and $qdat_i$ provides a primitive indication of the degradation due to phase error or noise. At test points TP12 to TP15 a higher sample rate is maintained through over-sampling to be compatible with the master sampling frequency f_s Hz. The availability of the timing reference $zerox_i$ at each stage ensures the test signals remain synchronous to the received signal.

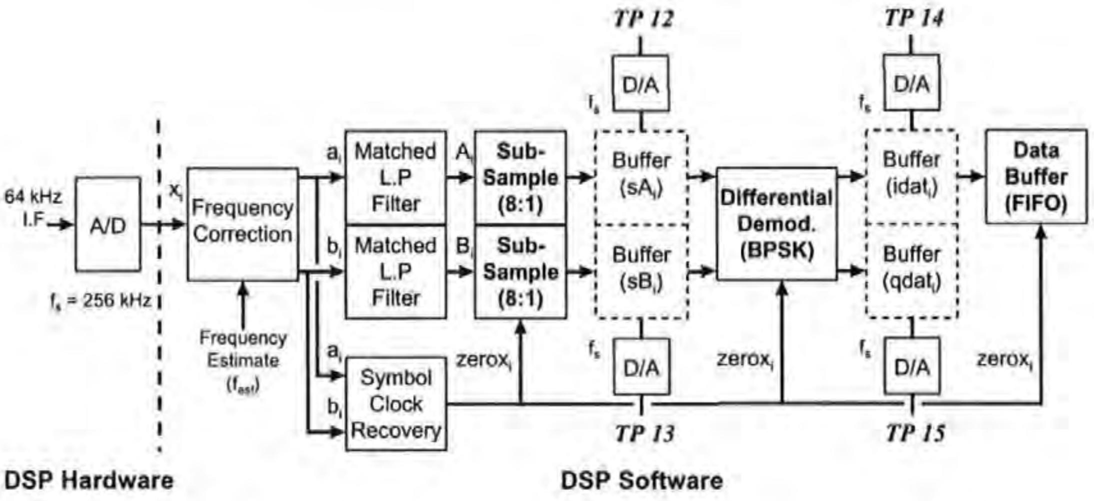


Figure D-24. DSP burst demodulator implementation - differential detection.

D.5 Unique Word Frame Synchronisation

Frame synchronisation is the process of detecting a strategic part of the transmitted data burst whereupon the redundant synchronisation preamble can be discarded and the information payload retained. In general a synchronisation preamble sequence is followed by a unique word sequence so that the receiver can identify the end of the preamble (the last bit of the unique word sequence) and hence a known position within the transmitted frame. For a burst demodulator the unique word sequence should be minimal length to reduce the preamble length but sufficiently long to allow a high probability of detection. The unique word sequence selected for these investigations was determined by the transmitter hardware configuration and in particular by the channel coding and error correction schemes employed. For this reason the section begins with an overview of the transmitter digital hardware. Once the hardware constraints have been defined, the author describes the process through which a suitable Unique Word sequence was selected. Finally DSP implementation of unique word frame synchronisation is discussed.

D.5.1 Digital Transmitter Hardware

Digital transmitter hardware was developed by others for these investigations. The digital hardware is contained on a PC interface card and contains numerous encoders, scramblers and control logic implemented within a field programmable gate array (FPGA). Figure D-25 shows a block diagram of the transmitter digital hardware for the main configuration adopted during these investigations; BPSK modulation, 16kbps scrambled user data with ½-rate FEC and differential encoding. Data for transmission is written to the FIFO buffer where it remains until burst transmission is initiated. Input to the ½-rate convolutional encoder is initially derived from the preamble ROM and then from the data path. Input to the convolutional encoder is at 16kbps and the two outputs are mutliplexed to produce the transmitted data rate (32kbps). Differential encoding is also applied prior to transmission to allow non-coherent reception and two outputs from the digital hardware provide synchronous clock and data signals for application to an external modulator. The transmitted burst ends once the FIFO buffer has emptied.

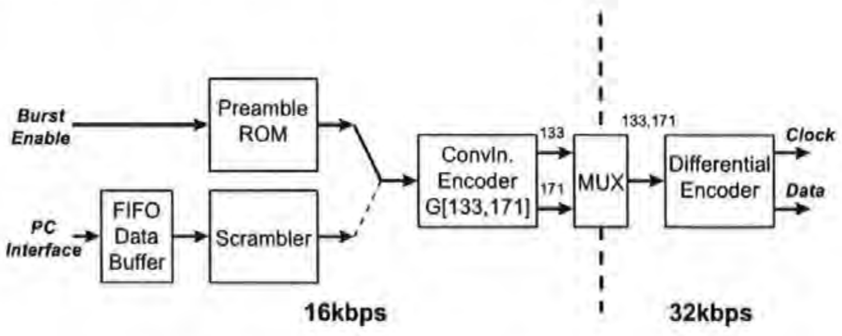


Figure D-25. Burst transmitter digital hardware.

It is important to emphasise that the preamble ROM, Figure D-25, contains raw data which produces the preamble sequence only after convolutional and differential encoding have been applied. It has already been described (in Chapter 3) that the transmitted preamble begins with alternating data so that carrier frequency acquisition may be conducted; in this case the corresponding preamble ROM contents can be easily deduced; Considering the differential encoder, it is easily shown that an alternating output occurs when logic '1' is continually applied to the input and, for this to happen, both convolutional encoder outputs must be logic '1'. The convolutional encoder consists of a seven-stage shift register and each output is derived from an odd number of shift register taps; logic '1' is produced at both outputs when the shift register contains only logic '1'. Alternating data for transmission is therefore guaranteed once the convolutional encoder has been primed seven times with logic '1' from the preamble ROM.

D.5.2 Unique Word Selection

The hardware shown in Figure D-25 is simple and further complexity is undesirable. In this form, the hardware imposes constraints which the author was required to address when selecting a suitable unique word. Specifically, that the use of a convolutional encoder limits the bit sequences which can be generated and eliminates the possibility of adopting sequences, such as Barker Sequences, which are commonly used for this purpose. The hardware imposes a further constraint in that the synchronising preamble consists of a period of alternating data prior to the unique word. To be clear, generation of a unique word sequence is also constrained by the preceding bits of the preamble. In principal, it was decided that frame synchronisation would employ a 40-bit unique word sequence with matching unique word correlator in the demodulator. The demodulator produces 16-bit two's compliment soft samples dat_i which represent the demodulated data; logic '1' maps to a positive value and logic '0' to a negative value. If the chosen 40-bit unique word is given by the values $uw_n = \pm 1$, over the range $n = 0$ to 39, the unique word correlator output uwc_i may be expressed

$$uwc_i = \sum_{n=0}^{39} uw_n \cdot dat_{(i-39)+n} \tag{D-10}$$

Ideally the correlator output is zero at all times except upon detecting the unique word, whereupon a known output occurs. However, since transmissions can originate from many stations with varying signal levels, a fixed detection threshold is not appropriate. A secondary process was selected to accommodate the variation to signal anticipated in practice. The unique word detection algorithm employed produces a result, $uwsynch_i$, which is defined by

$$uwsynch_i = uwc_i - \sum_{n=1}^{6} \left| uwc_{i-n} \right| \qquad \text{(D-11)}$$

and unique word detection is declared when uwsynch$_i$ $\geq$ 0. In this way, frame synchronisation is not influenced by signal level, and is determined by a relative increase in the correlator output only. The optimum 40-bit unique word was found by an exhaustive search of possible sequences that could be generated by the hardware and equations (D-10) to (D-11) provided rules with which many could be quickly eliminated. Since ½-rate FEC is employed, the search space is reduced from a potential 2^{40} (1.0995×10^{12}) 40-bit sequences to just the 2^{20} (1,048,576) that can be generated with the transmitter hardware. As a result, and assuming 1,000,000 tests per hour, the search time is reduced from hundreds of years to several hours on a typical personal computer. The 40-bit unique word selected is represented prior to encoding by the 20-bit hexadecimal value 0x5d70e (LSB encoded first), and after encoding by the hexadecimal value 0x0d23a9ac04 (LSB transmitted first). Results, Figure D-26, indicate that uwsynch$_i$ peaks at -26 prior to detection and reaches 32 at detection. It should be noted that other sequences gave equal performance to the parameters quoted.

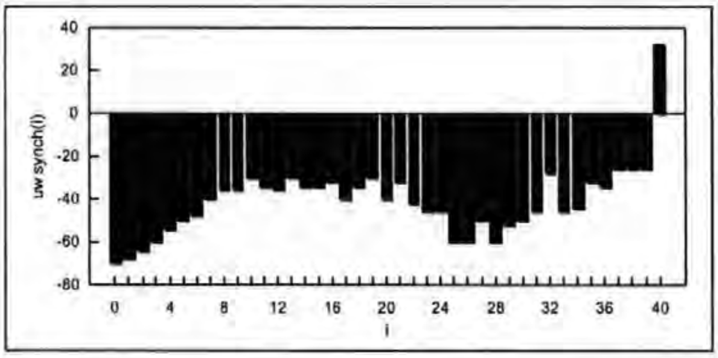


Figure D-26. Optimum 40-bit Unique Word search result - sequence 0x5d70e.

D.5.3 DSP Implementation

Figure D-27 shows a block diagram of the burst demodulator most closely representing the author's DSP implementation. Detail from early stages of the demodulator have been omitted for the sake of clarity and to emphasise the stages most closely linked associated with unique word frames synchronisation. In Figure D-

27, the input sample sequence x_i is frequency corrected to form the frequency corrected sample sequences a_i and b_i. After clock recovery, the filtered data signals A_i and B_i are sub-sampled, using the symbol timing reference $zerox_i$, to give the samples sequences sA_n and sB_n which represent the detected symbols. Differential demodulation is applied and the demodulated data represented by $idat_i$ is written to a software data FIFO buffer ready for further processing. The data FIFO buffer contents are applied to the unique word frame synchronisation algorithm which detects a unique word bit sequence transmitted as part of the synchronisation preamble. No input is applied to the error correction algorithm's input buffer until such time as the unique word has been detected. Upon detection of the unique word sequence, a known position in the transmitted frame has been reached and the frame synchronisation algorithm is immediately disabled so that the remaining data can be written directly to the error correction algorithm's input buffer. It is possible, at low SNR, for the burst demodulator to falsely acquire on background noise, and for the demodulator to be effectively placed out of action. Since the preamble length is fixed, unique word synchronisation must occur within a fixed time after carrier frequency acquisition. The frame synchronisation algorithm automatically indicates a 'false acquisition' if the unique word is not detected within this period, and the demodulator is immediately reset to acquisition mode if this happens. False acquisition is also declared if unique word frame synchronisation fails for any other reason.

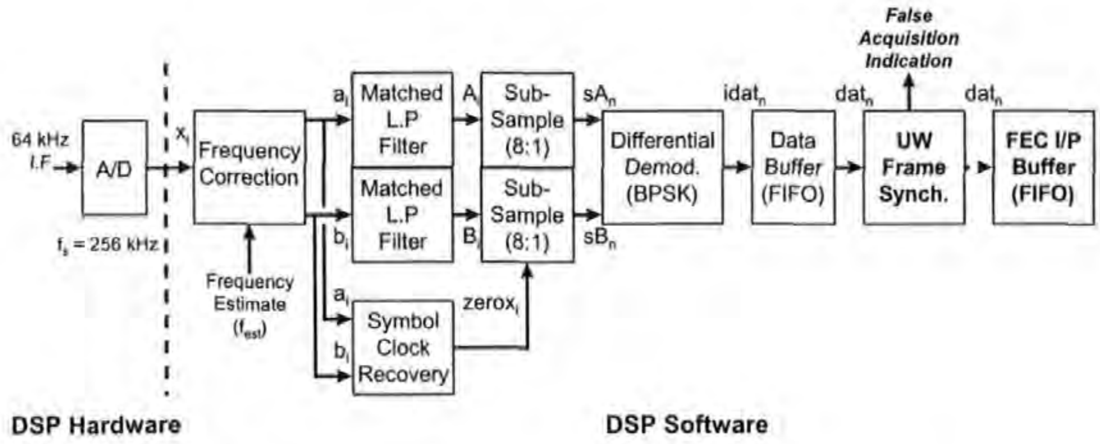


Figure D-27. DSP burst demodulator implementation - UW frame synch.

D.6 Software FIFO Buffers & Inter-Processor Communication

It should be emphasised that the Burst Demodulator algorithms depicted in Figure D-27 were implemented entirely as software on a single TMS320C50 DSP and that two DSPs are provided on the target hardware. It became clear that the processing requirements of existing algorithms prevented further major tasks from being assigned to the first DSP, and that the second should be employed. To complete the demodulator, and for it to perform a useful function, forward error correction algorithms and external PC interface buffering and control algorithms would be required. The 'FEC I/P Buffer (FIFO)', Figure D-27, provides an convenient and logical point at which to transfer responsibility for the remaining processing to the second DSP. In particular, the sample rate is reduced from a nominal 8 samples per symbol to 1 sample per symbol in the demodulator section and demodulated data is written to the FEC input buffer at a rate of 32ksps; since the sample rate is lower, the overheads associated with inter-processor communication are also reduced.

Inter-processor communication options provided by the TMS320C50 processor and the designer of the dual-DSP target hardware include high-speed (7Mbps) on-chip serial ports, high-speed (7Mbps) on-chip TDMA serial ports, parallel interfaces and global (shared) RAM. Global RAM was selected by the author as the preferred solution to inter-processor communication for the reasons given below. It should first be clarified that 'Global RAM' refers to 2k locations of 16-bit memory, provided by the designer of the target hardware, which appear simultaneously in the data memory space of both processors. This type of memory is supported directly by the TMS320C50 DSP which contains the additional logic to prevent contention; when two processors simultaneously access the same memory location. If this situation occurs, one processor is given access to the memory and the other is stalled until the risk of contention has passed, hence the integrity of each access to global memory is assured. The validity of DSP algorithms which utilise shared memory cannot be guaranteed and rely upon the use of semaphores or similar principal. The TMS320C50 on-chip high-speed serial port provides 7Mbit per second transfer rates between processors but when the use of interrupt service routines or polling loops is considered, the associated overheads are prohibitive. Similar arguments can be made for inter-processor communication using parallel interfaces and it is clear that Global RAM can provide higher throughput. Figure D-24 and Figure D-27 depict sections of the author's DSP burst demodulator software as block diagrams and in both cases functions referred to as 'Buffer (FIFO)' can be seen. A FIFO buffer, implemented as a software algorithm, is used many times within the burst demodulator and also provides the mechanism through which inter-processor communication was achieved. The software FIFO buffer is detailed in this section, with particular emphasis placed on inter-processor communication.

D.6.1 Burst Demodulator Applications for Software FIFO Buffer

In hardware terms, a FIFO (First In First Out) memory buffer is used to provide flexibility when devices with different characteristics must be interfaced. A typical example would be a device with a high speed burst output which must be interfaced with a device that operates at a slower speed. Providing the maximum burst length can be accommodated, the FIFO buffer allows these two incompatible devices to function together. In general, a FIFO buffer allows input to occur with a different characteristic to that of its output. A FIFO buffer can also be used in software where a function that writes to the buffer has a different characteristic to a function that reads from the buffer.

Throughout the early stages of the author's 'Burst Demodulator' software, samples are processed in fixed blocks of 256-samples. In these cases, the output from one function is directly compatible with the input of the next, since the block length is already known. At a certain point in the 'Burst Demodulator' software it becomes necessary to apply sub-sampling that is not always an integer division of the original sample rate, at this stage the block length becomes variable and it is necessary for the first function to communicate the new block length to functions that follow. A more general solution to this situation is to employ a FIFO buffer into which a variable number of samples are written. A function which reads from the FIFO buffer need not be aware of the block length and simply reads from the buffer until it becomes empty. This solution is particularly suited to situations where the buffer input is written in blocks of samples but the output processed on a sample by sample basis. Another application of a FIFO buffer in the burst demodulator software is encountered when the burst demodulator hardware/firmware is interfaced to a PC. After error correction and de-scrambling, the recovered data bytes are written to a temporary buffer for subsequent transfer to the PC. Using a parallel interface, provided by the designer of the dual-DSP target hardware, an interrupt is asserted by the PC to read directly from the temporary buffer. In this situation it is the main program that writes to the temporary buffer and an interrupt service routine (ISR) which reads from it. Since an interrupt can occur at any time, the main program and ISR must be considered as executing concurrently so as to avoid buffer errors. To be clear, the main program must never perform any operation relating to the buffer that would cause an error to occur if an interrupt was asserted at that instant.

The final application of a FIFO buffer in the burst demodulator software is associated with inter-processor communication. Unlike the aforementioned situation, inter-processor communication involves programs which are truly executing concurrently. It is essential that both programs access the buffer in a manner which respects that the other program is directly affected by its actions. In this situation the first processors writes a variable number of samples to the FIFO buffer in shared memory and the second processor reads from it. This maximises efficiency for the first processor because it continues executing the main program independently, the second processor is given maximum timing flexibility by the FIFO buffer and may execute its program asynchronously with respect to the first. The only requirements dictated by this arrangement is that the buffer must be sufficiently large to accommodate the maximum input block length and the second processor should read from the FIFO buffer faster (on average) than it is written to.

D.6.2 Software FIFO Buffer DSP Implementation

A software FIFO buffer implementation is described below specifically for the inter-processor communication application; the same basic principal was adopted for other applications. It is first necessary to allocate a block of global RAM in which to implement the FIFO buffer, and this must be repeated for both DSPs so that inter-processor communication can be achieved. Two locations at the start of this memory block are used to store a 'write pointer' (for input) and a 'read pointer' (for output), and the remaining (consecutive) locations form the buffer, Table D-2. For error free operation, the 'write pointer' is modified by DSP 1 when writing to the buffer but tested by DSP 2 when reading from the buffer. For the similar reasons the 'read pointer' is only modified by DSP 2 when reading from the buffer and tested DSP 1 when writing to the buffer. The remaining locations can only be written to by DSP 1 and read from by DSP 2.

Memory Address	**Contents of Location**	**DSP 1 (Input) Access Rights**	**DSP 2 (Output) Access Rights**
0	Write Pointer	Modify	Test
1	Read Pointer	Test	Modify
2	Buffer Data	Write	Read
3	Buffer Data	Write	Read
4	Buffer Data	Write	Read
5	Buffer Data	Write	Read
6	Buffer Data	Write	Read

Table D-2. Memory organisation for FIFO buffer (depth = 5 locations).

Using Table D-2 as an example, the buffer is initialised (and emptied) by setting both read and write pointers to memory address 2. In this general state, when both pointers are equal, the buffer is empty and DSP 2 cannot read from it. When writing to the buffer, DSP 1 increments the 'write pointer' only when data written to the current location is valid. Similarly, when reading from the buffer, DSP 2 increments the 'read pointer' only after data at the current location is no longer required. The buffer memory is addressed in a circular manner by both DSP 1 & 2 and, providing the buffer never fills, reliable operation is guaranteed. If the buffer fills, operation becomes erratic, hence it is essential that the buffer provides the required depth.

Figure D-28 shows a TMS30C50 assembly code macro, from the author's DSP burst demodulator software, that writes to the decoder input FIFO buffer located in shared Global memory; those instructions directly related with writing to the decoder input FIFO buffer are highlighted. It can be seen that the input buffer write and read pointers (#DBUFFW & #DBUFFR) are first restored and that a TMS320C50 circular buffer is defined so that the decoder input buffer is addressed in a circular manner. The input to the buffer occurs within a loop and italicised text shows an additional test to detect if the decoder input FIFO fills. The write pointer (#DBUFFW) is updated in global memory at the end of the macro, and only when this has occurred can the DSP 2 detect the new input.

```
BPSKTODECODERIP .macro
*****************************************************************************
* Purpose; If BPSK UW was detected, BPSK demodulator output sent to decoder *
*                                                                           *
* Example macro call;                                                       *
*    BPSKTODECODERIP            ;MACRO - Send BPSK stream to decoder input  *
*****************************************************************************
     ldp     #0                 ;Must be in data page 0
     mar     *,ar0              ;ARP=0

*    !! Skip if the BPSK UW has NOT been detected
     bit     PROGFLAGS,(15-2);BPSK UW detected flag
     bcnd    SKIP_BPSKOP?,ntc;Ignore BPSK output if NOT set

*    !! Set pointers
     lmmr    arcr,#BBUFFW       ;ARCR set with write pointer  ( BPSK output )
     lmmr    ar0,#BBUFFR        ;AR0 set with  read  pointer  ( BPSK output )
     lmmr    ar1,#DBUFFW        ;AR1 set with  write pointer  (decoder input)
     lmmr    ar2,#DBUFFR        ;AR2 set with  read  pointer  (decoder input)

*    !! Initialise circular buffers
     apl     #0h,cbcr           ;Disable CB1 and CB2
     splk    #BBUFFS,cbsr1      ;Set CBSR1 to start of buffer
     splk    #BBUFFE,cber1      ;Set CBER1 to end of buffer
     splk    #DBUFFS,cbsr2      ;Set CBSR2 to start of buffer
     splk    #DBUFFE,cber2      ;Set CBER2 to end of buffer
     opl     #098h,cbcr         ;Enable CB1 with AR0 and CB2 with AR1

*    !! Copy from BPSK output buffer to decoder input buffer
BPSKOP_LOOP?
      cmpr    0                 ;Are BPSK buffer read and write pointers equal?
      bcnd    BPSKOP_END?,tc    ;End if so

      lacl    *+,ar1            ;Read from output buffer
      sacl    *+,0,ar0          ;Write to input buffer

      lacl    ar1               ;Compare input buffer write pointer
      xor     ar2               ; and input buffer read pointer
      nop                       ;PLP for xc
      xc      2,eq              ;Next instruction if pointers equal
       opl     #020h,DISPLAY     ;Switch on LED 5 to indicate o/p buffer overrun

      b       BPSKOP_LOOP?      ;Repeat loop
BPSKOP_END?

     smmr    ar1,#DBUFFW        ;Update write pointer  (decoder input)
     smmr    ar0,#BBUFFR        ;Update read  pointer  ( BPSK output )
     apl     #0h,cbcr           ;Disable CB1 and CB2
SKIP_BPSKOP?
     .endm
```

Figure D-28. TMS320C50 assembly code - write to I/P FIFO buffer (DSP 1).

For completeness, an extract from the decoder TMS320C50 assembly code (DSP 2) is shown in Figure D-29. In this code the read pointer (#DBUFFR) is compared with the write pointer (#DBUFFW) to test if the decoder input FIFO buffer is empty. When the buffer is empty, when the pointers are equal, the test is repeated. Since the write pointer is modified by DSP 1, Figure D-28, it is subject to change at any time and must read again from global memory by DSP 2 prior to each test.

```
*!! Test for a 171 symbol in the decoder input buffer
    lmmr    ar6,#DBUFFR       ;AR6 becomes read pointer (CB2)
L1?
    lmmr    arcr,#DBUFFW      ;ARCR refreshed with write pointer (2 PLP)
    cmpr    0                 ;Does read pointer == write pointer?
    bcnd    L1?,tc            ;Loop if equal
```

Figure D-29. TMS320C50 assembly code - read from I/P FIFO buffer (DSP 2).

E. OFDM Modulator and Demodulator Models

In order to aid further discussion of OFDM synchronisation techniques, two system models are described; the 'Complex Sampling Model' and the 'Real Sampling Model'. These provide two distinctive methods of generating the same transmitted OFDM signal and two methods of demodulating the same received OFDM signal; each offers advantages in terms of either digital or analogue hardware requirements. Section E.1 contains a mathematical analysis of a modulator and demodulator for each model. In section E.2, practical considerations with respect to analogue hardware and digital hardware requirements are summarised.

E.1 OFDM System Models

The complex sampling and real sampling OFDM modulator models analysed mathematically in sections E.1.1 and E.1.2 respectively provide two distinctive methods of generating the same band of N unique carriers. Similarly, the complex sampling and real sampling OFDM demodulator models analysed mathematically in sections E.1.3 and E.1.4 provide two distinctive methods of demodulating this band of N unique carriers.

E.1.1 The Complex Sampling OFDM Modulator Model

The complex sampling model is most commonly used for theoretical analysis of ODFM transmissions, and a classic analogue structure will be described. An N-point IFFT (Inverse Fast Fourier Transform) may be used to generate N unique carriers if the principals of complex sampling and signalling are employed. A complex sampling OFDM modulator is shown in Figure E-1.

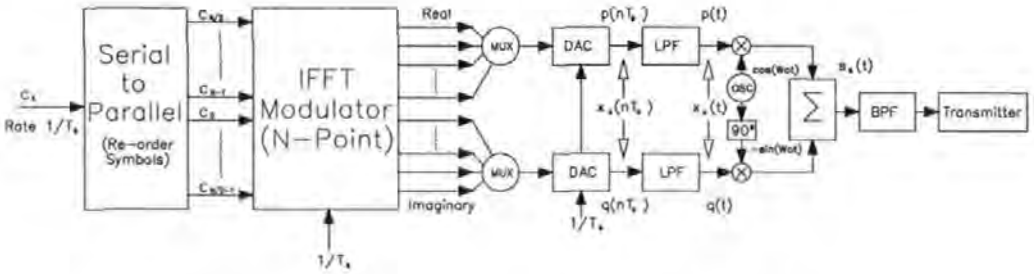


Figure E-1 Complex sampling OFDM modulator.

In Figure E-1, QPSK symbols $c_k = a_k+jb_k$, where a_k and b_k are restricted to ± 1, are applied at rate $1/T_b$ Hz to a serial to parallel converter. During each OFDM symbol period T_s, where $T_s = NT_b$, N QPSK symbols are applied to the IFFT modulator. To allow a direct comparison to be made later with the real modulator model, the symbols undergo a cyclic shift of $N/2$ so that the first symbol c_0 is applied to input $N/2$ of the IFFT modulator. The input time sequence is effectively modulated onto subcarriers, corresponding to a TDM (Time Division Multiplexing) to FDM (Frequency Division Multiplexing) conversion. The real and imaginary outputs from the modulator are separately multiplexed and applied to DACs (Digital to Analogue Converters) clocked at a rate of $1/T_b$ Hz. This gives the sampled complex baseband OFDM signal $x_c(nT_b) = p(nT_b) + jq(nT_b)$ where

$$x_c(nT_b) = \frac{1}{N}\sum_{k=0}^{N-1} c_{\left(k+\frac{N}{2}\right)\bmod N}\, e^{j\frac{2\pi kn}{N}} \qquad n = 0....N-1 \qquad \text{(E-1)}$$

Due to the sample rate, the maximum frequency that can be represented at the DAC outputs is $1/2T_b$ Hz, $x_c(nT_b)$ can therefore be expressed

$$x_c(nT_b) = \frac{1}{N}\sum_{k=0}^{N-1} c_k e^{j\frac{2\pi\left(k-\frac{N}{2}\right)n}{N}} \qquad n = 0....N-1 \qquad \text{(E-2)}$$

The frequency term $(k-N/2)$ in equation (E-2) shows that the OFDM signal is centred about zero frequency; the first symbol c_0 (k = 0) modulates a carrier with negative frequency while the last symbol c_{N-1} (k = N-1) modulates a carrier with positive frequency. As $x_c(nT_b)$ is a sampled signal, images appear throughout the spectrum at multiples of the sample rate $1/T_b$ Hz. Low pass reconstruction filters remove spectral image components and a continuous baseband OFDM signal $x_c(t) = p(t) + jq(t)$ is obtained where

$$x_c(t) = k_c\sum_{k=0}^{N-1} c_k e^{j\frac{2\pi\left(k-\frac{N}{2}\right)t}{T_S}} \qquad \text{(E-3)}$$

and k_c is a scaling factor. In order to retain the information necessary to transmit *N* unique channels, a complex signal must be transmitted. This is achieved with a quadrature up-conversion to a centre frequency of f_0 Hz, where only the real component needs to be transmitted. Equation (E-4) shows the frequency up conversion process in a form corresponding to the modulator structure of Figure E-1.

$$s_c(t) = [p(t)\cdot\cos(2\pi f_0 t)] + [q(t)\cdot -\sin(2\pi f_0 t)] \qquad \text{(E-4)}$$

Further manipulation of equation (E-4) , expanding *p(t)* and *q(t)* and rearranging in terms of a_k and b_k, allows the transmitted OFDM signal $s_c(t)$ to be expressed in a form which aids future comparison of the real and complex sampling models;

For convenience, let $A = 2\pi\left(\frac{2k-N}{2T_s}\right)\cdot t$, $B = 2\pi f_0 t$ and

$$s_c(t) = k_c \sum_{k=0}^{N-1} [a_k \cdot [\cos(A)\cos(B) - \sin(A)\sin(B)] - b_k \cdot [\sin(A)\cos(B) + \cos(A)\sin(B)]]$$

After further reduction,

$$s_c(t) = k_c \sum_{k=0}^{N-1} [a_k \cos(A+B) - b_k \sin(A+B)] \qquad \text{(E-5)}$$

and substituting back for A and B gives

$$s_c(t) = k_c \sum_{k=0}^{N-1} \left[a_k \cdot \cos\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right)\cdot t\right) - b_k \cdot \sin\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right)\cdot t\right)\right] \qquad \text{(E-6)}$$

Figure E-2 shows the spectrum $X_c(n/T_s)$ of the sampled baseband OFDM signal $x_c(nT_b)$, the spectrum $X_c(f)$ of the continuous baseband OFDM signal $x_c(t)$ and the spectrum $S_c(f)$ of the transmitted signal $s_c(t)$. The signal $x_c(nT_b)$ has both positive and negative frequency components and, because it is a complex signal, there is no symmetry about zero frequency. It has a maximum frequency $1/2T_b$ Hz and spectral images occur at multiples of the sample rate $1/T_b$ Hz. From a theoretical viewpoint perfect low pass reconstruction filters, with a cut off frequency $1/2T_b$ Hz, are used to remove spectral images. The resultant continuous signal $x_c(t)$ is still complex and remains centred about zero frequency. The final transmitted signal $s_c(t)$ is real and centred at frequency f_0 Hz.

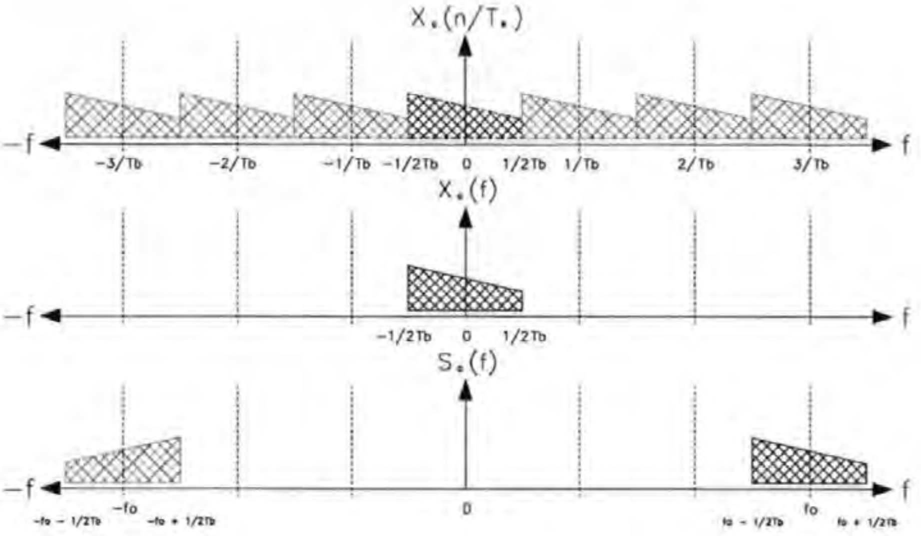


Figure E-2. OFDM spectra - a. sampled baseband OFDM signal $x_c(nT_b)$, b. baseband OFDM signal $x_c(t)$, c. transmitted OFDM signal $s_c(t)$.

E.1.2 The Real Sampling OFDM Modulator Model

The real sampling model is better suited to practical implementation as it avoids the complexity of complex sampling and quadrature up-conversion. The penalty paid for these advantages is higher digital hardware requirements. To generate N unique carriers, with the same bandwidth as the complex model, the IFFT modulator must have a length of twice that of the complex model; ie. *2N*. In order to maintain the same OFDM symbol period T_s, the DAC must be clocked at twice the sample rate. A real sampling modulator is shown in Figure E-3 and parameters have been labelled to allow a direct comparison to be made with the complex modulator in Figure E-1.

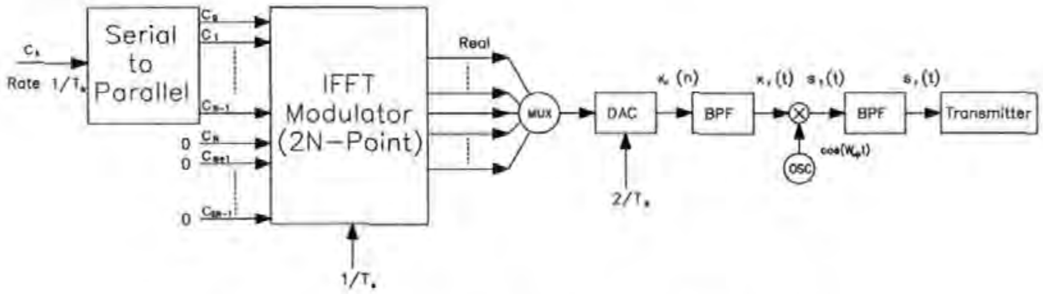


Figure E-3 Real sampling OFDM modulator.

Referring to Figure E-3, QPSK symbols $c_k = a_k + jb_k$, where a_k and b_k are restricted to ± 1, are applied at rate $1/T_b$ Hz to a serial to parallel converter. During each OFDM symbol period T_s, where $T_s = NT_b$, N complex samples are applied to the first N inputs of a *2N*-point IFFT modulator. The remaining N inputs are set to zero. With this model, only the real outputs from the IFFT are retained. It should be noted that the *2N* IFFT outputs are clocked out at rate $2/T_b$ Hz. The following equations define the sampled baseband OFDM signal $x_r(nT_b/2)$ generated at the output of the DAC. The notation Re[Z] is used to indicate the real component of Z.

$$x_r(n\tfrac{T_b}{2}) = \frac{1}{2N}\mathrm{Re}\left[\sum_{k=0}^{2N-1} c_k e^{j\frac{2\pi nk}{2N}}\right] \qquad n = 0....2N\text{-}1 \quad \text{(E-7)}$$

Equation (E-7) is the real component of the *2N*-point IDFT of sequence c_k. This may be expanded to take advantage of the fact that only the first N inputs are used.

$$x_r(n\tfrac{T_b}{2}) = \frac{1}{2N}\sum_{k=0}^{N-1}\left[a_k\cos\left(\tfrac{2\pi nk}{2N}\right) - b_k\sin\left(\tfrac{2\pi nk}{2N}\right)\right] \qquad n = 0....2N-1 \quad \text{(E-8)}$$

Signal $x_r(nT_b/2)$ has conjugate symmetry about zero frequency, unlike the complex model, and contains a band of carriers centred about frequency $N/(2T_b)$ Hz. Due to the sampling rate, images appear throughout the spectrum at multiples of $2/T_b$ Hz. A band pass filter retains selected image components so that the continuous baseband OFDM signal $x_r(t)$ applied to the mixer is given by

$$x_r(t) = k_r\sum_{k=0}^{N-1}\left[a_k\cos\left(\tfrac{2\pi(k+2N)\cdot t}{T_s}\right) - b_k\sin\left(\tfrac{2\pi(k+2N)\cdot t}{T_s}\right)\right] \qquad \text{(E-9)}$$

where k_r is a scaling factor. $x_r(t)$ must undergo a frequency up conversion such that the spectrum is also centred about f_0. The local oscillator in the mixer is therefore set to frequency f_{up}, where $f_{up} = f_0 - 2.5N/T_s$ Hz. At the mixer output signal, $s_1(t)$ is produced where

$$s_1(t) = x_r(t)\cdot\cos(2\pi f_{up}t) \qquad \text{(E-10)}$$

Further manipulation of equation (E-10) , expanding $x_r(t)$, allows the transmitted OFDM signal $s_1(t)$ to be expressed in a form that will aid future comparison of real and complex sampling modulator models.

For convenience, let $A = 2\pi\left(\frac{k+2N}{T_s}\right)\cdot t$, $B = 2\pi\left(f_0 - \frac{2.5N}{T_s}\right)\cdot t = 2\pi f_{up}\cdot t$ and

$$s_1(t) = \frac{k_r}{2}\cdot\sum_{k=0}^{N-1}\left[a_k(\cos(B+A)+\cos(B-A)) - b_k(\sin(B+A)-\sin(B-A))\right]$$

A band pass filter removes the unwanted sideband, the (B-A) terms, to give the transmitted signal $s_r(t)$. Substituting back for A and B gives;

$$s_r(t) = \frac{k_r}{2}\sum_{k=0}^{N-1}\left[a_k \cos\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right)\cdot t\right) - b_k \sin\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right)\cdot t\right)\right] \qquad \text{(E-11)}$$

Equation (E-11) is expressed in a form which allows comparison with the output from the complex modulator model in equation (E-6) . Clearly, neglecting the scaling factors, they are the same signal. Figure E-4 shows the spectrum $X_r(n/T_s)$ of the sampled baseband OFDM signal $x_r(nT_b)$, the spectrum $X_r(f)$ of the continuous baseband OFDM signal $x_r(t)$, the spectrum $S_I(f)$ of the mixer output $s_I(t)$ and the spectrum $S_r(f)$ of the transmitted signal $s_r(t)$. The signal $x_r(nT_b/2)$ is real and therefore has conjugate symmetry about zero frequency. It has maximum frequency N/T_b Hz with images throughout the spectrum at multiples of the sample rate $2/T_b$ Hz. For clarity a perfect band pass filter, extending from $2/T_b$ Hz to $3/T_b$ Hz, is used to retain selected components and produce the continuous signal $x_r(t)$. After up conversion, $s_I(t)$ has a band of carriers at the desired frequency f_0 but additional unwanted sideband resulting from the mixing process. Further bandpass filtering removes the unwanted sideband to leave the transmitted signal $s_r(t)$.

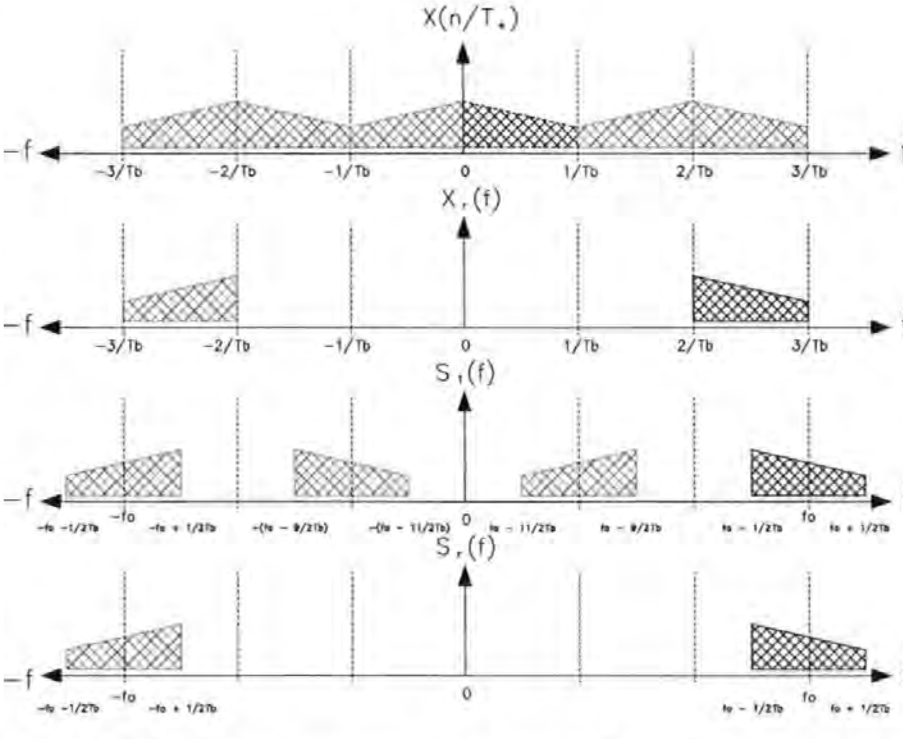


Figure E-4. OFDM spectra - a. sampled baseband OFDM signal $x_r(nT_b/2)$, b. baseband OFDM signal $x_r(t)$, c. mixer output $s_I(t)$, d. transmitted signal $s_r(t)$.

E.1.3 The Complex Sampling Demodulator Model

The complex sampling demodulator model has a similar structure to the complex sampling modulator model. It also employs the principals of complex sampling to provide the same advantages over the real model; half the FFT size and half the sampling frequency. A diagram of a complex sampling OFDM demodulator is shown in Figure E-5.

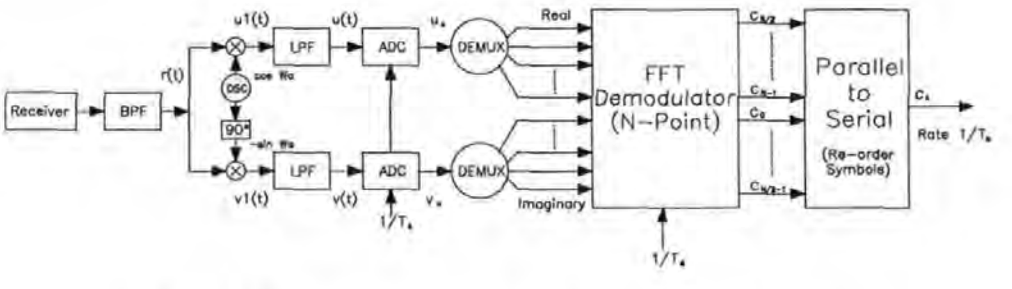


Figure E-5 Complex sampling OFDM demodulator.

Neglecting scaling factors, the received signal *r(t)* is a band of *N* modulated carriers centred at frequency f_0 of the form shown in equation (E-12) .

$$r(t) = \sum_{k=0}^{N-1}\left[a_k \cdot \cos\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right)t\right) - b_k \cdot \sin\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right)t\right)\right] \tag{E-12}$$

In order to recover a baseband OFDM signal, *r(t)* is applied to a quadrature down converter whose local oscillator is set to frequency f_0 Hz. In the down converter *r(t)* is mixed with a cosine carrier to produce signal $u_1(t)$ and with a -sine carrier to generate signal $v_1(t)$. These may be combined to form the complex signal $y_1(t)$, where $y_1(t) = u_1(t) + j\,v_1(t)$.

$$y_1(t) = r(t) \cdot e^{-j2\pi f_0 t}$$

$$y_1(t) = r(t) \cdot \cos(2\pi f_0 t) - jr(t)\sin(2\pi f_0 t)$$

For convenience, let $A = 2\pi\left(f_0 + \frac{2k-N}{2T_s}\right) \cdot t$ and $B = 2\pi f_0 t$.

$$y_1(t)=\sum_{k=0}^{N-1} a_k\cos(A)\cos(B)-b_k\sin(A)\cos(B)+j\sum_{k=0}^{N-1}-a_k\cos(A)\sin(B)+b_k\sin(A)\sin(B)$$

$$y_1(t)=\sum_{k=0}^{N-1} a_k\left(\frac{\cos(A+B)+\cos(A-B)}{2}\right)-b_k\left(\frac{\sin(A+B)+\sin(A-B)}{2}\right),\ j\sum_{k=0}^{N-1} a_k\left(\frac{\sin(A-B)-\sin(A+B)}{2}\right)+b_k\left(\frac{\cos(A-B)-\cos(A+B)}{2}\right)$$

Low pass filters remove the high frequency *(A+B)* terms leaving the complex baseband OFDM signal *y(t)*, where *y(t) = u(t) + j v(t)*, which is centred about zero frequency. Scaling factors are combined within the constant *K*.

$$y(t)=K\cdot\sum_{k=0}^{N-1} a_k\cos(A-B)-b_k\sin(A-B)+K\cdot\sum_{k=0}^{N-1} a_k\sin(A-B)+b_k\cos(A-B)$$

$$y(t)=K\cdot\sum_{k=0}^{N-1}(a_k+jb_k)\cdot e^{j(A-B)}$$

$$y(t)=K\cdot\sum_{k=0}^{N-1} c_k\cdot e^{j2\pi\left(\frac{2k-N}{2T_s}\right)t} \qquad \text{(E-13)}$$

u(t) and *v(t)* have maximum frequency $1/2T_b$ Hz and are sampled by ADCs at rate $1/T_b$ Hz to form the complex baseband OFDM sample sequence y_n, where $y_n = u_n + j\ v_n$.

$$y_n=K\cdot\sum_{k=0}^{N-1} c_k\cdot e^{\frac{j2\pi\left(k-\frac{N}{2}\right)n}{N}}$$

$$y_n=K\cdot\sum_{k=0}^{N-1} c_{\left(k+\frac{N}{2}\right)\bmod N}\cdot e^{\frac{j2\pi kn}{N}} \qquad \text{where } n=0....N-1 \qquad \text{(E-14)}$$

Equation (E-14) may be recognised as the *N*-point IDFT of sequence $c_{k+N/2}$. Every symbol period T_s, where $T_s = NT_b$, *N* complex samples y_n are applied to the input of an *N*-point FFT. At its output, the transmitted symbols c_k are subject to a cyclic shift of *N/2* bins such that the first symbol c_0 appears in bin *N/2*.

$$c_{\left(k+\frac{N}{2}\right)\mathrm{mod}\,N} = K \cdot \sum_{n=0}^{N-1} y_n \cdot e^{\frac{-j2\pi kn}{N}} \qquad \text{where } k = 0.....N\text{-}1 \qquad \text{(E-15)}$$

The FFT output is applied to a multiplexer, which also restores the natural symbol order, to produce the original symbol sequence c_k at rate $1/T_b$ Hz. Figure E-6 shows the spectra of the received OFDM signal *R(f)*, the continuous baseband OFDM signal *Y(f)* and the sampled baseband OFDM signal *Y(n/T_b)*. *r(t)* has a band of *N* carriers centred about frequency f_0 Hz and, as it is real, has conjugate symmetry about zero frequency. *y(t)* is the result of quadrature down conversion and is centred about zero frequency; there is no symmetry. *y(n)* is a sampled version of *y(t)* and therefore has images throughout the spectrum at multiples of the sample rate $1/T_b$ Hz.

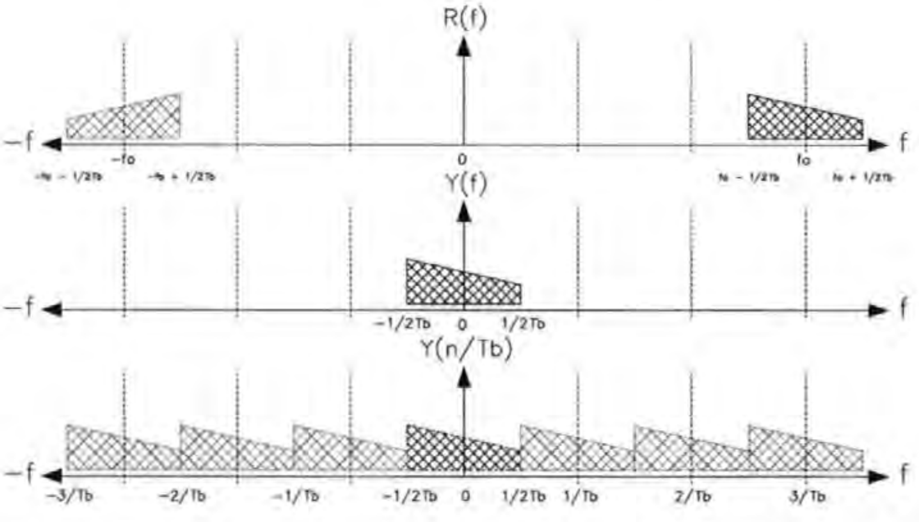


Figure E-6. OFDM spectra - a. received OFDM signal *r(t)*, b. baseband OFDM signal *y(t)*, c. sampled baseband OFDM signal *y(nT_b)*.

E.1.4 The Real Sampling Demodulator Model

The real sampling demodulator model has a similar structure to the real sampling modulator model. It also utilises a 2N-point FFT, and a single ADC clocked at twice the rate of the complex model. A diagram of a real sampling OFDM demodulator is shown in Figure E-7.

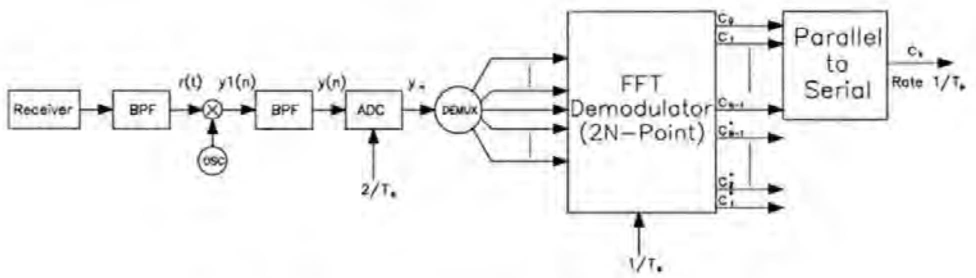


Figure E-7 Real sampling OFDM demodulator.

Neglecting scaling factors, the received signal *r(t)* is a band of *N* modulated carriers centred at frequency f_0 of the form shown in equation (E-16) .

$$r(t)=\sum_{k=0}^{N-1}\left[a_k\cdot\cos\left(2\pi\left(f_0+\tfrac{2k-N}{2T_s}\right)t\right)-b_k\cdot\sin\left(2\pi\left(f_0+\tfrac{2k-N}{2T_s}\right)t\right)\right] \tag{E-16}$$

To recover a baseband OFDM signal, *r(t)* is applied to a mixer whose local oscillator is set to frequency f_0 - $N/2T_s$ Hz. Signal $y_1(t)$ is produced by mixing *r(t)* with a cosine carrier.

$$y_1(t)=r(t)\cdot\cos\left(2\pi\left(f_0-\tfrac{N}{2T_s}\right)\cdot t\right) \tag{E-17}$$

For convenience, let $A=2\pi\left(f_0+\frac{2k-N}{2T_s}\right)\cdot t$, $B=2\pi\left(f_0-\frac{N}{2T_s}\right)\cdot t$ and

$$y_1(t)=\sum_{k=0}^{N-1}a_k\cdot\cos(A)\cos(B)-b_k\cdot\sin(A)\cos(B)$$

$$y_1(t)=\sum_{k=0}^{N-1} a_k\cdot\frac{\cos(A+B)+\cos(A-B)}{2}-b_k\cdot\frac{\sin(A-B)-\cos(A+B)}{2} \qquad \text{(E-18)}$$

A band pass filter removes the high frequency *(A+B)* terms to leave the baseband OFDM signal *y(t)*. *y(t)* has a band of *N* carriers centred about frequency *1/2*T_b Hz.

$$y(t)=K\cdot\sum_{k=0}^{N-1} a_k\cdot\cos(A-B)-b_k\cdot\sin(A-B)$$

$$y(t)=K\cdot\mathrm{Re}\left[\sum_{k=0}^{N-1}(a_k+jb_k)\cdot e^{j(A-B)}\right]$$

$$y(t)=K\cdot\mathrm{Re}\left[\sum_{k=0}^{N-1}c_k\cdot e^{j\frac{2\pi kt}{T_s}}\right] \qquad \text{(E-19)}$$

y(t) contains frequencies from *0* Hz up to *N-1/N*T_b Hz and must therefore be sampled at rate *2/*T_b Hz to form the sampled sequence y_n.

$$y_n=K\cdot\mathrm{Re}\left[\sum_{k=0}^{N-1}c_k\cdot e^{j\frac{2\pi kn}{2N}}\right] \qquad \text{where n = 0....2N-1} \quad \text{(E-20)}$$

Equation (E-20) may be recognised as the real component of the *2N*-point IDFT of sequence c_k. Every symbol period T_s, where $T_s = NT_b$, *2N* real samples are applied to the input of a *2N*-point FFT. At its output, the transmitted symbols c_k appear in bins *0* to *N-1* while the conjugate of the transmitted symbols appear in bins *N* to *2N-1*.

$$c_k=K\cdot\sum_{n=0}^{2N-1}y_n\cdot e^{-j\frac{2\pi kn}{2N}} \qquad \text{where k = 0....N-1} \quad \text{(E-21)}$$

$$c^*_{(2N-k)}=K\cdot\sum_{n=0}^{2N-1}y_n\cdot e^{-j\frac{2\pi kn}{2N}} \qquad \text{where k = N....2N-1} \quad \text{(E-22)}$$

As shown in equation (E-21) , only the first N outputs from the FFT are applied to a multiplexer to reproduce the original symbol sequence c_k at rate $1/T_b$ Hz. The conjugate symbols, from equation (E-22) are of little consequence.

Figure E-8 shows spectra of the received OFDM signal *R(f)*, the continuous baseband OFDM signal *Y(f)* and the sampled baseband OFDM signal $Y(n/T_b)$. *r(t)* has a band of *N* carriers centred about frequency f_0 Hz and, as it is real, has conjugate symmetry about zero frequency. After down conversion *y(t)* is centred about frequency $1/2T_b$ Hz and is also real. *y(n)* is a sampled version of *y(t)* and therefore has images throughout the spectrum at multiples of the sample rate $2/T_b$ Hz.

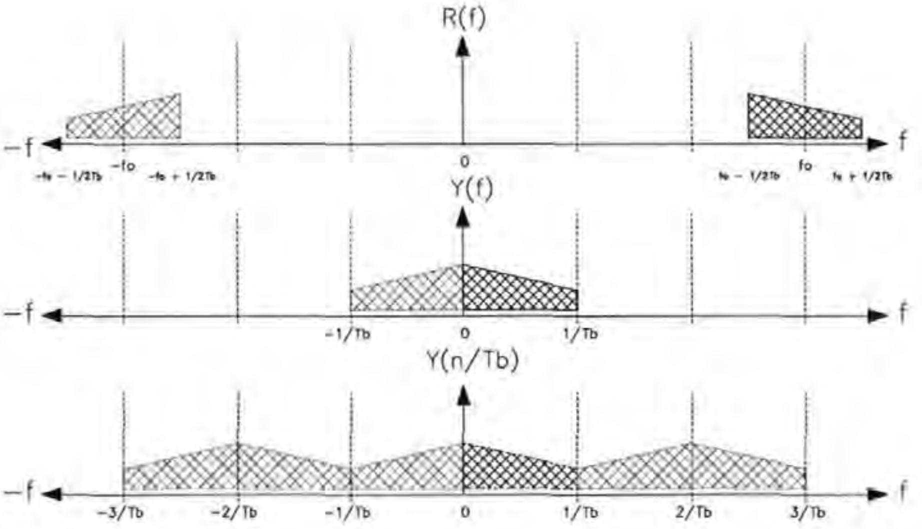


Figure E-8. OFDM spectra - a. received OFDM signal *r(t)*, b. baseband OFDM signal *y(t)*, c. sampled baseband OFDM signal $y(nT_b)$.

E.2 Comparison of Theoretical OFDM Models

Two theoretical methods of generating a bandpass OFDM signal have been described. Referring to equations (E-6) and (E-11) , expressions for the transmitted signal generated by the complex sampling model and real sampling model, $s_c(t)$ and $s_r(t)$ respectively, were derived.

$$s_c(t) \equiv 2 \cdot s_r(t) = k_0 \sum_{k=0}^{N-1} \left[a_k \cdot \cos\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right) \cdot t\right) - b_k \cdot \sin\left(2\pi\left(f_0 + \tfrac{2k-N}{2T_s}\right) \cdot t\right) \right] \quad \text{(E-23)}$$

From these discussions it is clear that these models are completely interchangeable. Similarly, two theoretical methods of demodulating a signal of the form shown in equation (E-23) were demonstrated. These demodulator models are also completely interchangeable. The hardware requirements of the complex sampling and real sampling models are summarised in the following tables;

Parameter	Complex Sampling Model	Real Sampling Model
Modulator IFFT Size	N-point	2N-point
D/A Requirement	Two DACs	One DAC
Sample Rate	$1/T_b$ Hz	$2/T_b$ Hz
Reconstruction filtering	Two Low Pass Filters	One Bandpass Filter
Up Conversion	Quadrature Up Converter	Mixer
Sideband Suppression	N/A	One Bandpass filter

Table E-1 Summary of OFDM modulator model requirements.

Parameter	Complex Sampling Model	Real Sampling Model
Down Conversion	Quadrature Down Converter	Mixer
Anti-aliasing Filtering	Two Low Pass Filters	One Bandpass Filter
Sample Rate	$1/T_b$ Hz	$2/T_b$ Hz
A/D Requirement	Two ADCs	One ADC
Demodulator FFT Size	N-Point	2N-Point

Table E-2 Summary of OFDM demodulator model requirements.

Mathematically the models produce identical results, if scaling factors are neglected, so a choice must be made based upon practical considerations. The choice of which model to use for the modulator and demodulator will be dictated by factors such as digital and analogue hardware requirements, analogue filter characteristics, hardware stability and manufacturing considerations like size and cost.

E.2.1 Digital Hardware Requirements

The digital hardware is most likely to impose the limits on the number of carriers transmitted and hence the overall data rate of the system. The digital hardware of the complex model is more desirable as it is clocked at half the rate of the real model. In addition, the complex model utilises N-point FFTs instead of 2N-point FFTs. An obvious disadvantage with the complex model is that two devices are required when converting from the digital to analogue domain in the modulator and from the analogue to digital domain in the demodulator. If the modulator and demodulator are to be implemented as digital signal processing (DSP) software algorithms, the complex model is the optimum choice since it offers lower computation overhead.

E.2.2 Analogue Hardware Requirements

Neglecting filtering, there is one major difference between the complex and real models; The complex model requires quadrature up and down conversion while the real model utilises a simple mixing process. Any mismatches in the quadrature up converter at the modulator will cause orthogonality in the transmitted OFDM signal to be lost. Similarly, mismatches in the quadrate down converter at the demodulator will also have the same effect. The DACs in the modulator and the ADCs in the demodulator must be well matched in terms of gain and conversion time, if orthogonality is to be maintained throughout the system. Loss of orthogonality results in inter carrier interference at the demodulator output and may result in corrupt data.

E.2.3 Analogue Filter Requirements

The real sampling model requires two bandpass filters while the complex model uses two low pass filters. In terms of complexity, low pass filters are more desirable. For theoretical discussions perfect filters were assumed but, in practice, filters with 'brick wall' type characteristics are not realisable. To allow analogue filters to roll off, a guard band must be provided. This means that several carriers at the upper and lower frequencies of the band must be removed; not transmitted. The result of providing guard bands is that the overall throughput of the system is reduced from the theoretical maximum of N symbol per second.

F. The Internet Protocol Suite

The Internet Protocol Suite, Figure F-1, is a suite of protocols aimed at providing meaningful computer communication. The Transmission Control Protocol (TCP) and the Internet Protocol (IP) are the two major protocols within the suite; hence the Internet Protocol Suite is commonly referred to as just TCP/IP. The Internet Protocol Suite was originally developed for the United States department of Defence Advanced Research Projects Agency (DARPA) network and research began during the 1970's; by 1983 the conversion to the new TCP/IP protocols was complete. Figure F-1 shows the major protocols of the Internet Protocol Suite and a brief description of each protocol group follows.

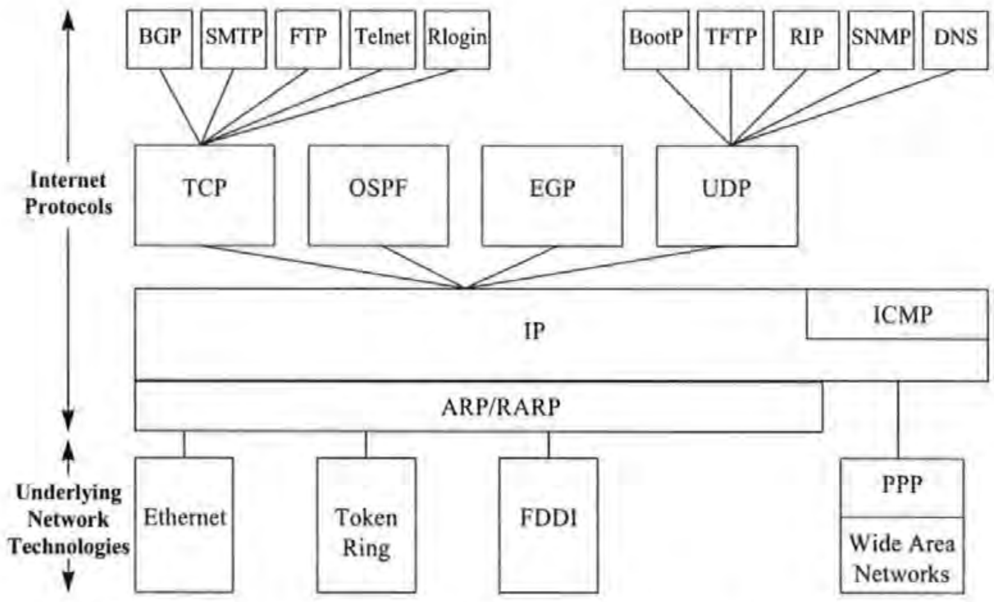


Figure F-1. The Internet Protocol Suite (Major Protocols).

Underlying Network Technologies

The Internet Protocol Suite does not define underlying network technologies and relies upon those already defined by bodies such as the IEEE. Two protocols, the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP), are included to map the addressing scheme used by IP to that used by the underlying network. The suite also includes the Point-to-Point Protocol (PPP) in order to control wide area links.

The Internet Protocol (IP)

IP is the most important protocol of the suite and provides the mechanism that all other protocols use. It is a connectionless protocol and adds minimal overhead in terms of control information. IP has no error reporting facilities and relies upon the upper layer protocols to provide reliability. Some lower-level error reporting is required and the Internet Control Message Protocol (ICMP) provides this.

Routing Protocols

For networks to be adaptive, it is important to use protocols that can detect changes in the network and react accordingly. Interior Gateway Protocols (IGPs) define the policies in use by a local network and include the Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF) protocol. Exterior Gateway Protocols (EGPs) define the policies that join these networks together and includes the Border Gateway Protocol (BGP) and the Exterior Gateway Protocol (EGP).

End User Applications

The Internet Protocol Suite provides a number of user applications, the best known are the File Transfer Protocol (FTP) and Telnet. Another important protocol is the Simple Mail Transfer Protocol (SMTP) through which electronic mail (e-mail) services are provided. Other common applications, which are not part of the suite, are Web Browsers such as those provided by Netscape and Microsoft.

Supporting Services

Many of the user applications rely on a standard naming convention called the Domain Naming System (DNS). As ARP allows the mapping of IP addresses to those of the underlying network, the DNS protocol allows mapping of names to IP addresses to make things easier for human operators. As an example, 'www.satnet.plymouth.ac.uk' is easier to remember than 141.163.53.50. Other examples are the Bootstrap Protocol (BootP) and the Trivial File Transfer Protocol (TFTP).

Management

The Simple Network management Protocol (SNMP) is widely used.

F.1 Underlying Network Technology

The underlying network technology used in these investigations is that of the satellite data return link. In its general form, the satellite data return link does not require or have the need for hardware addresses. In a TCP/IP networking environment it is convenient to use hardware addresses as they can aid filtering and traffic management. It is common for experimental or new technology to emulate an existing, and well supported, technology. The most common Local Area Network (LAN) medium is collectively referred to as Ethernet although two similar, but different, standards exist; Ethernet and IEEE 802.3. Due to its support in major operating systems, and as the technology employs a hardware addressing mechanism, Ethernet emulation was employed for investigations into Satellite Internet Delivery Systems.

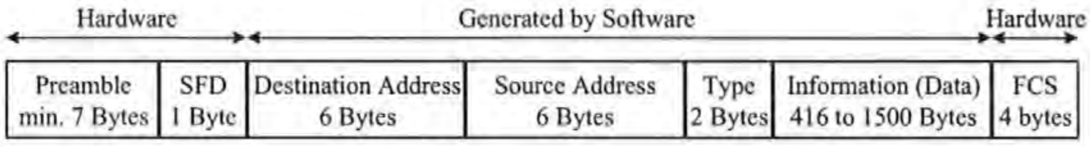


Figure F-2. Ethernet Frame structure

Of particular significance to this investigation is the frame structure used by Ethernet, Figure F-2. An Ethernet frame begins with a preamble of alternating 1's and 0's followed by the start frame deliminator (which has the binary pattern 10101011). This sequence is automatically generated by the transmitting Ethernet network interface connection (NIC) and provides a means for other NICs to synchronise to the transmission. The remaining fields of the Ethernet frame are generated by software on the sending terminal except for the Frame Check Sequence, a cyclic redundancy check (CRC), which is also hardware generated and allows transmission errors to be detected.

The destination address field is a unique 6-byte media access controller (MAC) address which is unique to the NIC of the intended recipient of the frame. All NICs on the same physical network receive each frame but subsequently discard it if a comparison of its own MAC address with that of the destination address field of the frame does not match. The source address field gives the 6-byte MAC address of the station that sent the frame and may be used in order to reply to the correct sender. For Ethernet, the following field is a 2-byte protocol type code which identifies which protocol is being carried in the information field of the frame. Expressed in hexadecimal format, the lowest type code defined in the Ethernet standard is 0x0600 and corresponds to the Xerox XNS protocol; other common type codes are shown in Table F-1.

Type Code	Protocol
0x0800	Internet Protocol (IP)
0x0806	Address resolution Protocol (ARP)
0x8035	Reverse Address resolution Protocol (RARP)
0x8137	Novell IPX
0x8138	Novell IPX

Table F-1. Common Ethernet 'type' codes.

For the IEEE 802.3 standard the type field is replaced by a *length field* which specifies the length of the information field in the frame. Its maximum value is 1500 bytes (0x05dc in hexadecimal) and so all Ethernet frames will be discarded since they appear to be too long. Conversely, to an Ethernet station, 802.3 frames have an invalid type and are also discarded. This small difference ensures that Ethernet and IEEE 802.3 networks may coexist on the same physical network without misinterpretation; but also prevents them from communicating. For the remainder of this text only to the Ethernet standard is considered.

F.2 Internet Addressing Scheme

It is a requirement that each station on a network has a unique address. Each station already has a hardware or media access controller (MAC) address, 6-bytes for Ethernet, but this is not sufficient to allow communication with other networks which may use 2-byte, 8-byte or some other length for their MAC addresses. The Internet Protocol Suite therefore uses its own addressing scheme to allow communication to take place between stations using any underlying network technology and to accommodate future expansion. The first aim was satisfied but the unexpected in the number of stations connected to the Internet has meant the address base is not large enough. The designers of the Internet used 32-bit (4-byte) addresses thus giving 2^{32}

(4294967295) possible stations. This required a central authority to oversee the allocation of addresses to each station. To make the address space as flexible as possible it was decided that the 32-bits should be divided into a universally administered Network ID (Network Address) and a locally administered Host ID (Host Address) to reduce administrative overhead. By carefully encoding the address bits, the following network types were defined;

A small number of networks with a large number of hosts	- Class A
A moderate number of networks with a moderate number of hosts	- Class B
A large number of networks with a small number of hosts	- Class C

Class D and class E addresses are also defined. The former is used for Multicasting, using a single address to transmit to a group of stations, and the latter is reserved for experimental use. These address classes can be distinguished by examining the first bits of an address, Figure F-3.

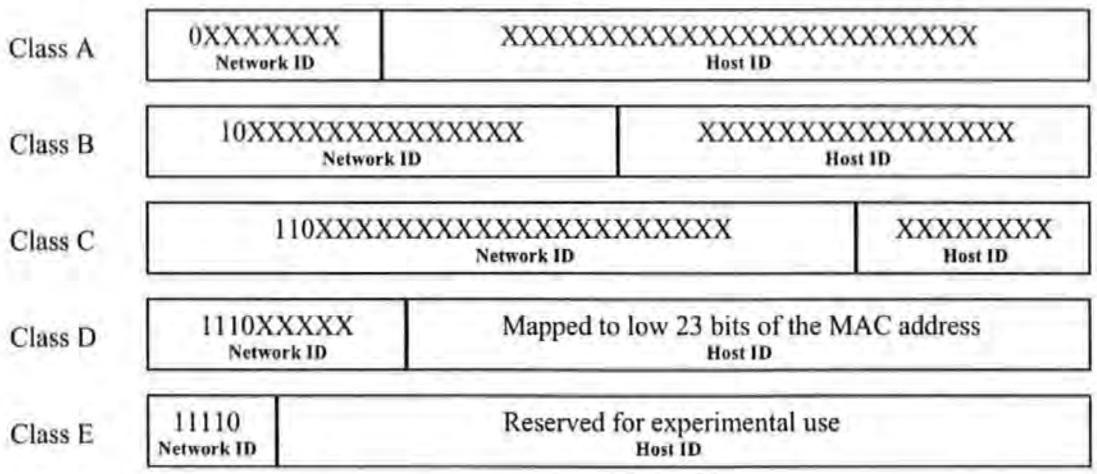


Figure F-3 Internet address classes.

IP addresses are most commonly represented using decimal notation by dividing the address into 4 octets and using the numbers 0 (for all 0's) to 255 (for all 1's). To further aid readability the octets are usually separated by decimal points (dotted decimal notation). With Class A addresses, the first octet takes the value 0 to 127. A value of 0 has the meaning 'this Class A network' and a value 127 is reserved for 'loop-back testing'. Class A addresses therefore take the range 1.0.0.1 to 126.255.255.254. With Class B addresses, the first octet will always be in the range

128 to 191 and the last two octets form the Host ID. Class B addresses are therefore in the range 128.0.0.1 to 191.255.255.254. With Class C stations, the first octet is in the range 192 to 223 and the last octet is reserved as the Host ID. Class C stations have addresses in the range 192.0.0.1 to 223.255.255.254. Class D addresses are special addresses and have a first octet set between 224 and 239. With Multicast the lower 24-bits do not specify an individual Host, instead they identify a group of Hosts which respond to their own address and that of the multicast group to which they belong. Class E addresses have their first byte set to between 240 and 255 but are only used by users who have registered them for experimental purposes. For completeness it should also be brought to the reader's attention that two Host IDs also have special meaning. A station may not be assigned a Host ID of all 0's since this is used to refer to a network or to a broadcast to all Hosts on that network (10.0.0.0 refers to Class A network 10). Similarly a Host ID of all 1's is the most common way to represent a broadcast to all Hosts on a network (10.255.255.255 refers to a broadcast to all Hosts on Class A network 10).

F.2.1 Free Addresses

Many organisations are not connected to, or have no need to connect to, the Internet but still wish to use the TCP/IP Protocol Suite for network communication. The are 3 common groups of addresses which are reserved for private networks which all Internet Service providers (ISPs) are obliged to block traffic to and from. Organisations are free to use these addresses within their private networks without risk of conflicting with 'legal' users. The reserved addresses were utilised during investigations into a Satellite Internet Delivery system and are as follows;

A single Class A network	10.0.0.0
A single Class B network	172.16.0.0
254 individual Class C networks	192.168.0.0

F.2.2 Routing

The aim of any addressing scheme is to provide successful communication between any co-operating stations. Hosts may communicate directly if they both exist on the same network (both physically and with the same Network ID portion of their IP address); this is referred to as direct routing, Figure F-4.

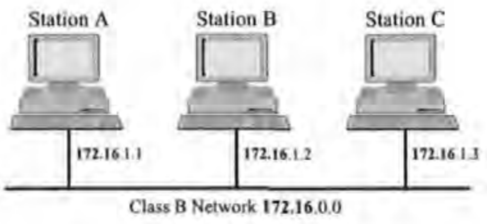


Figure F-4 Network on which direct routing may be used.

When hosts reside on different networks, Figure F-5, a router must be employed. A router may be likened to a postal service as it facilitates communication between remote locations. Before one host can communicate with another it must first know its own IP address and that of the destination host. Once equipped with this information the sending host can examine the Network ID portion of both addresses to determine if Direct Routing can be employed. Routers, as in Figure F-5, have connections to two or more networks and appear as hosts on all the networks to which they connect. When a station wishes to communicate with a remote host it must also be equipped with a third piece of information; the IP address of a router which can take responsibility for delivering the message. This information is generally referred to as the 'Default Gateway', as is often the router itself. For example, in Figure F-5, if station A wishes to communicate with station D it will determine that direct routing is not possible as each station resides on a different network; Class B network 172.16.0.0 and Class A network 10.0.0.0 respectively. If station A is equipped with the IP address of the router it will trust that the router can send the message on towards station D on its behalf. In practice the Internet is much more complex, to communicate between a host at the University of Plymouth in the UK and a host at the University of South Australia IP datagrams may pass through more than 20 routers. The principle however is very simple; each host and router has the address of a 'default gateway' it should use if it cannot deliver a datagram using direct routing or has not been equipped with specific routing information relating to that destination. Using this principle, the message eventually gets delivered to its destination even though the sending host has no knowledge of the route it takes.

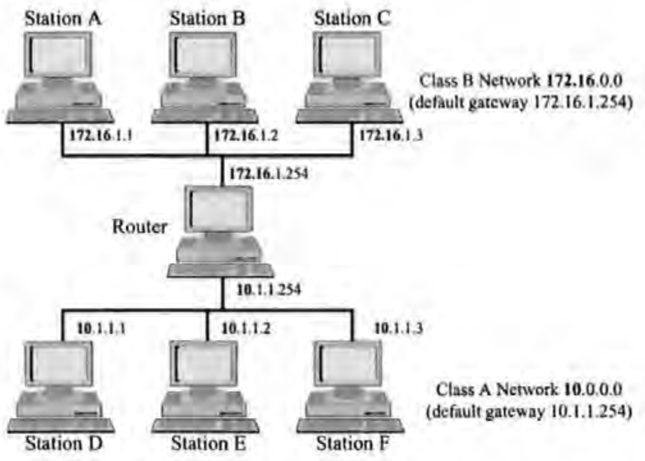


Figure F-5 Two Networks linked by a Router.

F.2.3 Subnetting

A large organisation with a Class A or Class B address will have a very complex network which may consist of many local area networks (LANs) and wide area networks (WANs). These organisations may wish to assign logical groupings to their hosts to simplify management. There are a number of reasons why grouping is desirable;

- Incompatible LAN technologies may be employed ie. Ethernet, Token Ring, Fibre…
- LAN technologies impose limits as to the number of hosts they support due to parameters such as cable length; many hosts can cause these limits to be exceeded.
- Some groups of hosts may generate significantly more traffic than others; it is often convenient to concentrate these hosts on a separate LAN.

Ideally each logical grouping would be assigned a different Network ID, but this makes inefficient use of the available IP addresses. So far in these discussions addresses have two levels of hierarchy; a Network ID and a Host ID. Subnets provide a further level of hierarchy by allowing some of the Host ID bits to be assigned to the Network ID. This allows a Network to be further divided into subnets corresponding to the logical groupings required. Each host on the subnet will consider this as an independent IP network.

From earlier discussions the Network ID, and thus the Class of network, is determined by the leading bits of the IP address. The logical groups created are seen as subnets of the Network while the hosts within these groups see them as individual networks. For example, IP address 192.168.1.2 could be considered as Host 1.2 on network 192.168.0.0 or as Host 2 on subnet 1 of network 192.168.0.0. The subnet mask, or netmask, provides a programmable mechanism to specify which bits the host should consider as being the Network ID and which should be considered as the Host ID. A subnet mask is usually given in dotted decimal form and uses the decimal numbers which place 1's in the positions of the IP address that the host should treat as the Network ID. With the previous example, IP address 192.168.1.2, netmask 255.255.0.0 indicates that a host will consider itself as host 1.2 on network 192.168.0.0. IP address 192.168.1.2, netmask 255.255.255.0 indicates that a host will consider itself as host 2 on subnet 1 of network 192.168.0.0, Figure F-6.

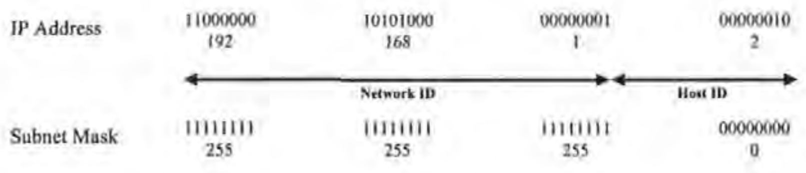


Figure F-6 Using a Subnet Mask to define Network and Host ID.

For the three major Classes of networks, the subnet masks are as follows;

Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

If no subnet mask is specified, these 'Natural subnet masks' are employed. Subnet masks are locally administered and there is no restriction on their form providing they respect the Network ID that was assigned to the organisation; subnet masks tend to have consecutive leading 1's but this is not a requirement. As already discussed, Host

IDs of all 1's and all 0's are not allowed and this is also true after subnetting has been applied.

Up to this point an IP address has specified a particular host on a network but many hosts have multiple network interfaces, connecting them simultaneously to multiple networks, as is the case for a router. More strictly an IP address must be considered as identifying a physical connection and not the host. Furthermore, it is sometime convenient to assign multiple IP addresses to the same physical connection to link logical networks that reside on the same physical network, Figure F-7, this is called multi-homing.

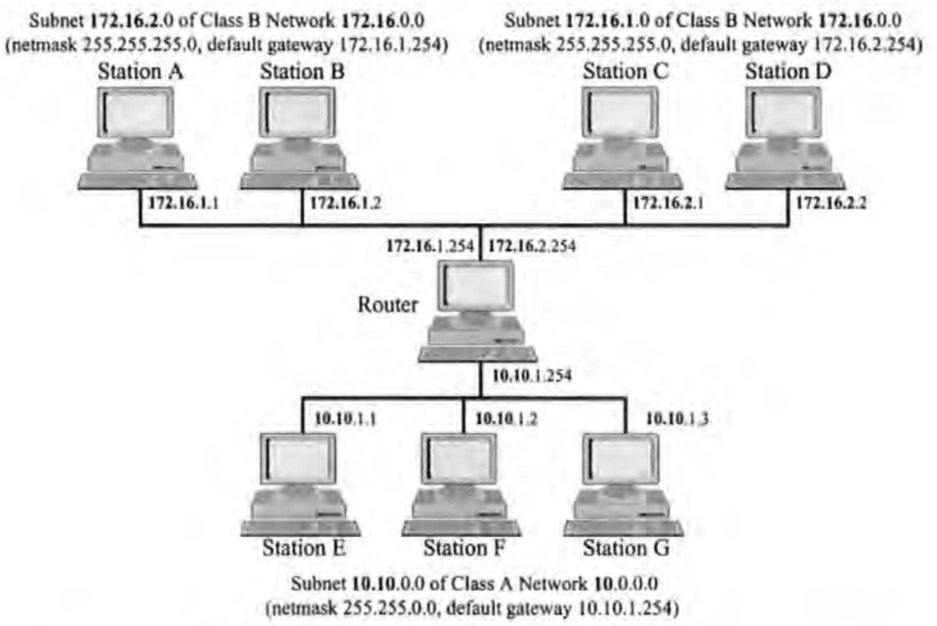

Subnet 172.16.2.0 of Class B Network 172.16.0.0
(netmask 255.255.255.0, default gateway 172.16.1.254)
Subnet 172.16.1.0 of Class B Network 172.16.0.0
(netmask 255.255.255.0, default gateway 172.16.2.254)
Station A
Station B
Station C
Station D
172.16.1.1
172.16.1.2
172.16.2.1
172.16.2.2
172.16.1.254
172.16.2.254
Router
10.10.1.254
10.10.1.1
10.10.1.2
10.10.1.3
Station E
Station F
Station G
Subnet 10.10.0.0 of Class A Network 10.0.0.0
(netmask 255.255.0.0, default gateway 10.10.1.254)

Figure F-7 A Multi-homed Router linking three logical Networks.

Figure F-7 shows two physical networks which have been divided into three logical networks, 172.16.1.0, 172.16.2.0 and 10.10.0.0 respectively. Network 172.16.1.0 sees a router with IP address 172.16.1.254 and network 172.16.2.0 sees a router with IP address 172.16.2.254, but they are in fact both the same interface on the same router. Network 10.10.0.0 sees the router at IP address 10.10.1.254, which is corresponds to its second interface. In addition to linking the three logical networks, the router provides a means of filtering traffic between stations A and B and stations C and D even though they are on the same physical network.

F.2.4 Internet Addressing Summary

The addressing scheme used by IP is flexible and allows networks and hosts to be uniquely identified. A two level hierarchy is sufficient for routing but doesn't make efficient use of the address space. Subnetting provides a third level of hierarchy so that the address space can be further divided into logical groups which more closely represent physical networks that these hosts reside on. The 32-bit IP address space is rapidly becoming too limited and plans are in motion to implement a new version of IP with 128-bit addresses.

F.3 The Internet Protocol

The Internet Protocol (IP) is described as a 'connectionless datagram delivery system' and it is said to be unreliable since no guarantees are made that datagrams are delivered to their destination. It is connectionless since the datagrams are delivered in isolation; with IP there are no connections or logical circuits. Datagrams may be lost, duplicated or arrive out of sequence at the destination. IP is also described as a 'best efforts' delivery system because datagrams may be legitimately discarded, due to insufficient resources, without informing the source host. Even so, IP is regarded as a robust and versatile protocol. An exhaustive description of IP and its features is beyond the scope of this text, instead only those details pertinent to the satellite Internet delivery systems are presented.

IP datagrams travel encapsulated within physical or MAC frames such as Ethernet Frames or Satellite Return Link packets, Figure F-8. Due to this encapsulation, a limit is imposed on the length of IP datagrams by the Maximum Transmission Unit (MTU) of the physical frames. LANs are commonly based upon the Ethernet standard which has an MTU of 1514 octets or bytes. The IP datagram plus the Ethernet overhead (header and deliminator) must therefore not exceed 1514 octets in an Ethernet-type network. IP may legitimately reduce the datagram length if the destination network has a smaller MTU. Since this investigation aimed to emulate the Ethernet standard, the Satellite Return Link Packets are also subject to an MTU of 1514 bytes. Figure F-8 shows the IP datagram format and how it is encapsulated within a MAC frame.

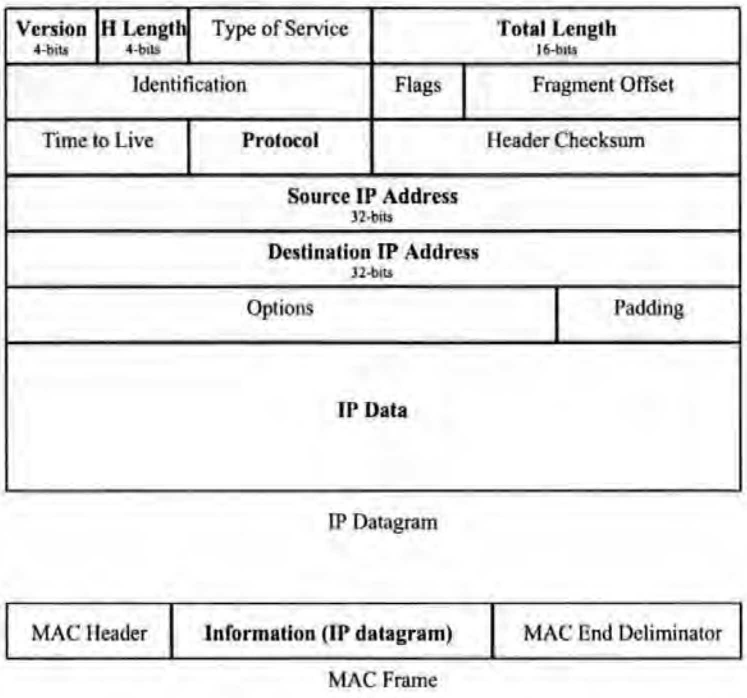


Figure F-8. IP Datagram format and encapsulation within a MAC Frame.

The IP datagram, Figure F-8, consists of numerous fields but only those referred to later in the text will be highlighted. The 4-bit Version field identifies the version of the protocol and therefore the format of the header; only version 4 is currently supported widely on the Internet. The 4-bit Header Length field specifies the number of 32-bit words that make up the IP datagram header; this information is essential if one wishes to locate, examine or modify the IP Data as was done during these investigations. Similarly, the Total Length field specifies the total length of the datagram in octets and, as it is 16-bits long, gives a theoretical maximum of 2^{16} octets or 64k Bytes. As already explained, the MTU of the underlying network generally limits datagrams to much less than this length. The 8-bit Protocol field indicates which protocol is being carried within the datagram and indicates the format and contents of the IP Data field. Table F-2 shows the main Protocol Codes of significance to this investigation. Finally, the 32-bit source and destination fields specify the IP address of the transmitting and destination host respectively. Other fields are equally important, but need not be considered or understood.

Code	Protocol
0x04	IP in IP Encapsulation
0x06	The Transmission Control Protocol (TCP)
0x11	The User Datagram Protocol (UDP)

Table F-2. Common IP protocol codes.

F.4 The Transmission Control Protocol

The Transmission Control Protocol (TCP) is a 'connection oriented', 'end to end reliable' protocol. It is not designed to interface with the underlying network technology since it does not provide any means to address remote Hosts. Similarly it provides no methods for fragmentation or reassembly, nor for the transport through intermediate routers. The Internet Protocol performs these tasks on behalf of upper layer protocols, such as TCP. TCP makes few assumptions as to the reliability of the underlying protocol and upper layers, such as applications, do not need to consider reliability since TCP dictates that all data sent must be acknowledged within specified timeout periods. When these acknowledgements fail to arrive, TCP re-sends the data and hence overcomes the unreliable nature of IP. TCP is a reliable process to process service and is used as an interface between applications and IP. An application passes data to TCP for transmission on the network, and delivery to a destination host. TCP calls upon IP, the underlying protocol, to package the TCP data into datagrams. Finally IP passes the datagram to the underlying network access protocol (Ethernet) for encapsulation of the datagram in a physical frame on the network, Figure F-9.

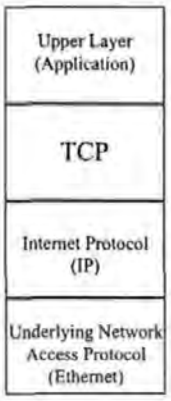

Upper Layer (Application)
TCP
Internet Protocol (IP)
Underlying Network Access Protocol (Ethernet)

Figure F-9. TCP and the four-layer model.

F.4.1 TCP Reliability and Flow Control

TCP provides reliable communication through a 'positive acknowledgement' system. This requires that each transmitted segment be acknowledged; a timer is started when the segment leaves the transmitting host and an acknowledgement must be received before the timer expires. TCP must be able to recover from segments that are lost, duplicated or arrive out of sequence. TCP therefore assigns a sequence number to each octet transmitted and requires the receiving host to acknowledge each octet received. When acknowledgements are not received, or the acknowledgement itself is lost, retransmission takes place from the sequence number of the oldest unacknowledged octet. Segments arriving out of sequence are catered for by examining the sequence number upon reception. It would be extremely inefficient to send an acknowledgement packet for each octet received since many octets are contained in each segment. To increase efficiency, TCP increments the sequence number by the number of octets contained in the transmitted segment and the acknowledgement returned is incremented by the number of octets contained in the received segment. In this way a whole segment, multiple octets, are acknowledged with a single transmission.

So far a means of providing reliability has been described but, if the sending host has to wait for each segment to be acknowledged before sending the next, it doesn't make efficient use of the available bandwidth. TCP also employs a system known as 'Sliding Windows' which allows multiple segments to be unacknowledged at any time in order to make better use of the available transmission bandwidth. Conceptually, a window is placed over the data so that any data to the left of the window has been transmitted and acknowledged; data within the window has been transmitted but not acknowledged and data to the right of the window has not yet been transmitted, Figure F-10.

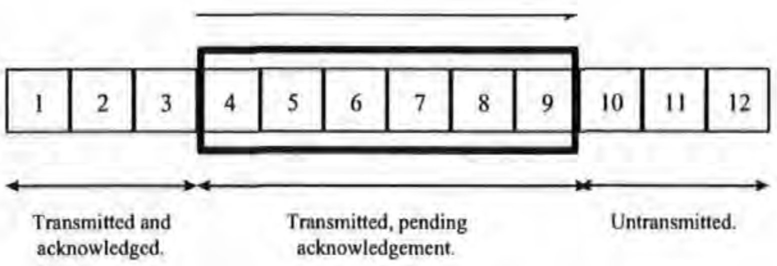

1
2
3
4
5
6
7
8
9
10
11
12
Transmitted and acknowledged.
Transmitted, pending acknowledgement.
Untransmitted.

Figure F-10. Sliding Window principle.

As multiple segments may be sent before an acknowledgement is received, multiple segments may be acknowledged at the same time. The example in Figure F-11 first shows an initial state where the first 6 segments have been transmitted but not yet acknowledged. Some time later, but before segment 1 was presumed to have been lost, an acknowledgement for segment 1 arrives. The window is incremented and the next segment is transmitted. Some time later an acknowledgement for segment 5 is received which also acknowledges the previous segments. The acknowledgements for segments 2 to 4 might have been lost or simply not sent but, in either case, the acknowledgement for segment 5 is sufficient to increment the window by 4 locations.

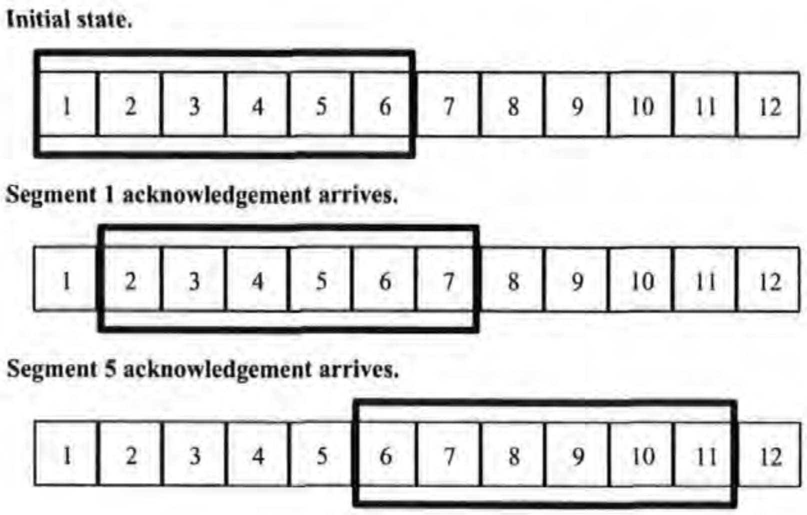

Initial state.
1
2
3
4
5
6
7
8
9
10
11
12
Segment 1 acknowledgement arrives.
Segment 5 acknowledgement arrives.

Figure F-11. Sliding Window example.

In order to provide flow control TCP provides a means to govern the amount of data the transmitting host may send. To achieve this, each acknowledgement sent by the receiving host contains a 'window' that indicates the number of octets that the receiver is prepared to accept. In this way, a transmitting host will know how much data to send and the receiving host may communicate the state of its buffers. The acknowledgement window, as it is sometimes called, is particularly significant when using TCP over networks with high delay and bandwidth. Satellite links in particular have relatively long delays and can result in the sending host constantly pausing while it waits for an acknowledgement to come back. Satellite links also tend to have larger bandwidth and this effect can result in extremely poor efficiency. For many operating systems a terrestrial network is assumed and the acknowledgement window is set for optimum performance with low delays. When a satellite link or other high delay/bandwidth network is employed, significant performance gains are achieved by modifying the algorithm for window assignment. This is discussed in greater detail later in this chapter.

F.4.2 TCP Segment Format

The TCP Segment format is shown in Figure F-12 and, as for IP, an exhaustive description of each field is beyond the scope of this text. Instead, emphasis is placed upon those fields which were used for packet filtering, spoofing and prioritising during investigations of satellite Internet delivery systems using the satellite data reply link. The 16-bit Source Port and Destination Port fields identify an application on the source and destination host respectively; these are particularly relevant when it is necessary to trap packets sent by particular applications. The 32-bit Sequence Number specifies the sequence number of the first octet of data carried in the segment, except when the Syn. Flag is set. The 32-bit Acknowledgement Number field is valid only when the Ack. Flag is set and will contain the sequence number of the first octet in the next segment the receiver expects to receive and has the effect of acknowledging all previous octets. The 4-bit Data Offset filed indicates the length of the TCP segment in 32-bit words (4 octets). Of the 6 flag bits (Urg, Ack, Psh, Rst, Syn and Fin), Ack and Syn are particularly interesting. The Ack Flag indicates that a segment header contains a valid Acknowledgement Number while the Syn Flag is set only when a connection is being established. The loss of a segment containing a Syn Flag is relatively destructive, resulting in a long wait until the segment is retransmitted. Conversely, a segment containing only an acknowledgement (as many do) may be lost without detrimental affect if a later acknowledgement is successfully received. Since the acknowledgements are cumulative, a degree of loss may be tolerated. The ability to recognise the two aforementioned segment types is fundamental to Packet Steering algorithms discussed later in this chapter. Finally, the Window field forms part of the flow control mechanism and states the number of octets the receiving station is prepared to accept before sending an acknowledgement.

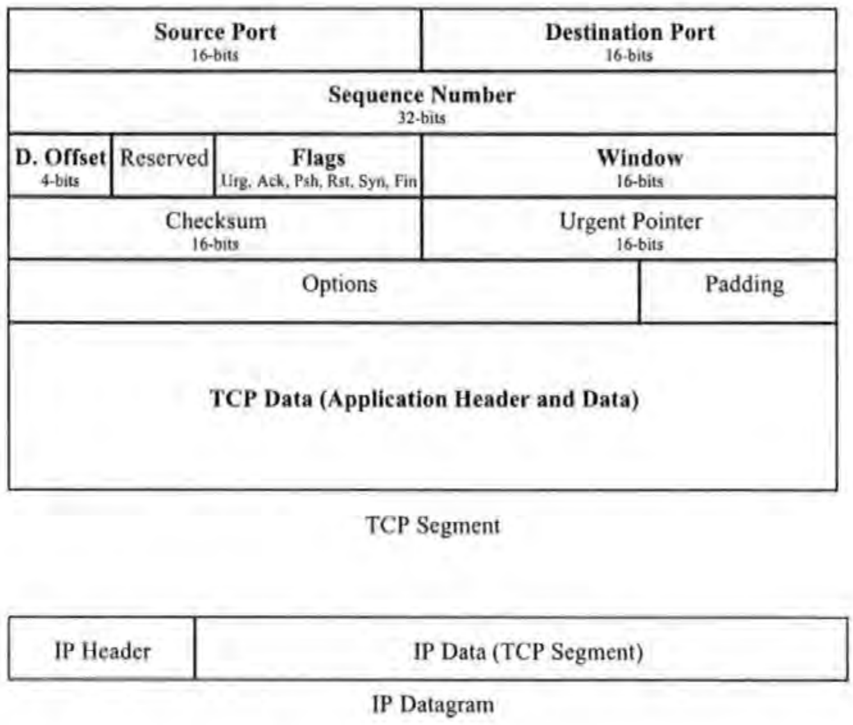


Figure F-12. TCP Segment format and encapsulation within an IP Datagram.

F.5 The User Datagram Protocol

The User Datagram Protocol (UDP) is designed to provide applications with the ability to transfer data to other applications on remote machines with minimal overhead. UDP assumes that IP is in use as the underlying protocol, Figure F-13, and as with TCP, UDP is unable to interface directly with the access protocol of the underlying network technology. Unlike TCP, UDP provides no added reliability and does not send acknowledgements; UDP also has no flow control mechanism. Instead UDP relies upon the application to provide reliability over that supplied by IP. UDP is a fundamental protocol upon which many functions of the Internet are built.

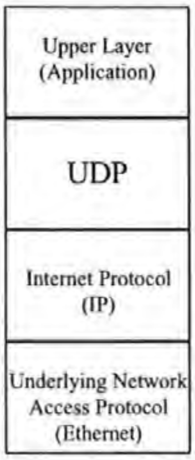

Upper Layer (Application)
UDP
Internet Protocol (IP)
Underlying Network Access Protocol (Ethernet)

Figure F-13. UDP and the four-layer model.

UDP utilises a fixed length header and variable length data area and travels encapsulated within an IP Datagram, Figure F-14. The two 16-bit Source Port and Destination Port fields identify applications on the source and destination hosts as with TCP while the Length field identifies the length of the whole UDP datagram in octets.

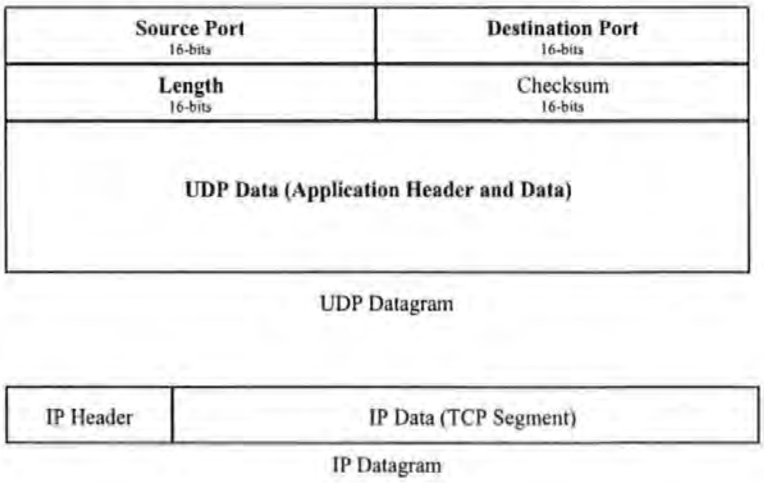


Figure F-14. UDP Datagram format and encapsulation within an IP Datagram.

G. Satellite Internet Delivery Systems - Supplemental Information

G.1 A Hybrid Terrestrial/Satellite Internet Delivery System

PSTN modems using the v90 standard provide a downstream data rate (from ISP to the customer) of up to 56kbps over a high quality telephone line from a digital exchange. These modems are backwards compatible with the older v. standards and can also operate at 33.6kbps, 28.8kbps, 14.4kbps, 9600bps and 4800bps; upon connection the modems negotiate the highest connection rate that can be mutually supported. In 1997, a pilot investigation was conducted in Amman, Jordan, where reliable modem connections above 9600bps were not frequently achieved; not sufficient for browsing complex pages on the World Wide Web or for large file transfers. At that time, the Jordanian ISP's main connection to the Internet backbone was also of a low quality and could not support any significant increase in throughput. Objectives for investigations were twofold;

1) Demonstrate a faster Internet connection to the Home user.
2) Minimise/remove additional load on the local ISP's network infrastructure.

A novel solution was devised (by others) which utilised both an existing Jordanian ISP and a second ISP located in the UK, Figure G-1. The author's most significant contribution to these investigations was novel network device driver software to provide IP communication over the satellite broadcast channel. Additional contributions were made with regard to IP networking and performance analysis in Amman during pilot trials. Further information relating to novel satellite software and hardware may be found in section G.1.1 and additional information is contained within the author's publication "The Development of an Operational Satellite Internet Service Provision" in Appendix I. A brief system overview is given below.

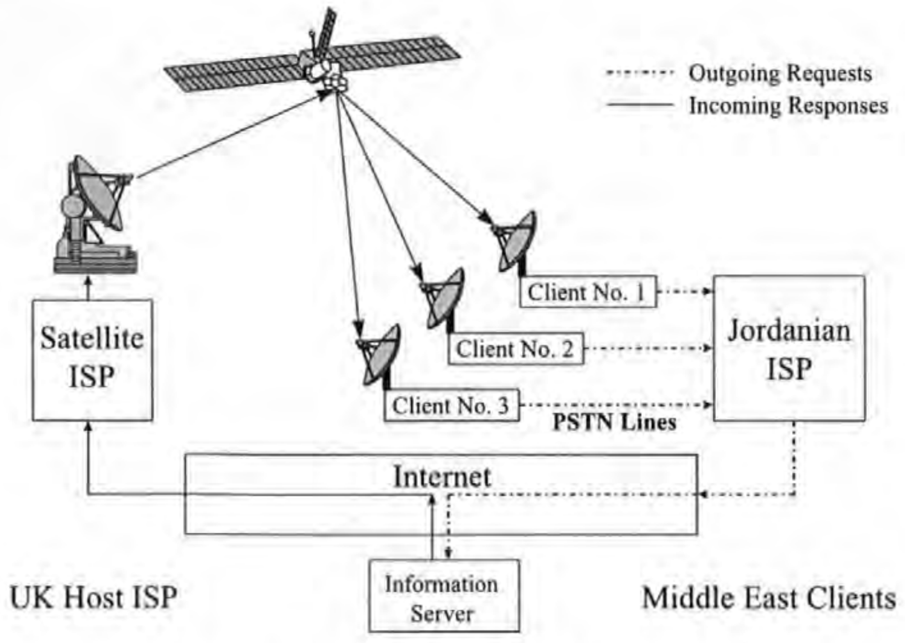


Figure G-1. Hybrid satellite Internet delivery system - system overview.

With reference to Figure G-1, each client terminal (located in the Middle East) is equipped with a standard dial-up connection to a local Internet Service Provider for the request channel and a satellite data modem for the response channel. Following the asymmetric principle introduced in Chapter 5, requests for information were sent via terrestrial modem at 9.6kbps and responses received back at much higher rates over the satellite broadcast data channel (90kbps). In terms of an IP network diagram, Figure G-2, clients are assigned IP addresses from subnet 162.16.2.0 of IP network 162.16.0.0. Requests are routed onto the Internet Backbone in the Middle East but responses are delivered to the UK Host ISP; since it is the registered owner of IP network 162.16.0.0. Destination addresses from subnet 162.16.2.0 at the UK Host ISP are recognised as the Middle East clients and routing tables ensure that responses are sent back via the Host PC using the satellite broadcast response channel. It is significant that each client terminal has two network interfaces and is therefore associated with two different IP addresses (multi-homed). However, since IP datagram filtering is provided by the Internet Protocol Suite of the client terminal, based upon the IP address assigned to the modem, the IP address assigned to the data broadcast interface is of no consequence.

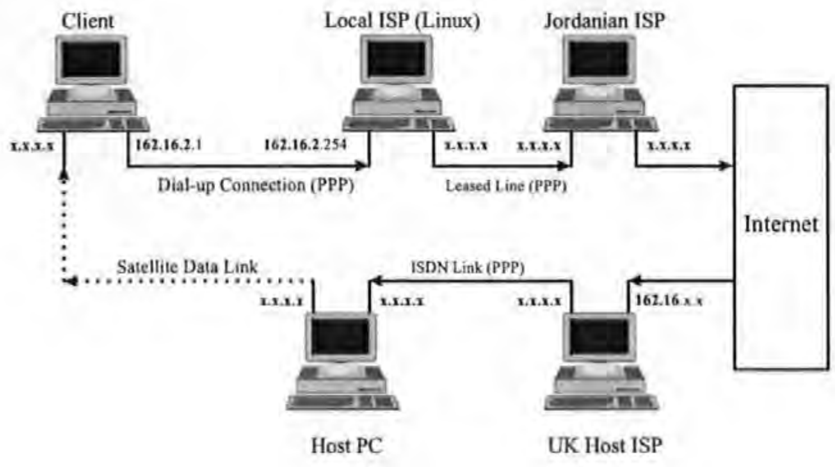


Figure G-2. Hybrid satellite Internet delivery system - network diagram.

The Hybrid Terrestrial/Satellite Internet Delivery System described provided a useful investigation into the feasibility of Satellite Internet Delivery Systems. The principle was novel at the time these investigations were conducted and many commercial systems have since been launched based upon this system model; although most are unconnected to these investigations. Results were positive and the speed of delivery was perceived by typical users to be approximately ten times faster when compared with a standard Jordanian dial-up Internet service. However, since the 90kbps response channel was shared among all users, the system rapidly became congested with several active users. The pilot system did show that satellite Internet delivery was feasible and that users could greatly benefit from a system of this type. In conclusion, this first investigation showed that the satellite broadcast channel should be much faster than 90kbps in order to serve many simultaneous users. The investigation also showed that a single ISP, ideally with the satellite uplink on-site, would make such a system more practical.

G.1.1 Novel Network Software and Hardware

For clarity, it is necessary to present information relating to both satellite hardware and network software for the Host and Client PC since the two elements are inter-dependant. It is stated once again that the author's contribution was to the network device driver software and the transmitted frame structure; existing hardware was utilised.

The Host PC transmits TCP/IP datagrams encapsulated within Ethernet frames over the satellite broadcast channel using a modified SatLink data broadcast PC interface card, Figure G-3. Frames for transmission, 'SatLink frames', provide fields for synchronisation purposes and carry the Ethernet frame generated by the Internet Protocol Suite as a data payload. The Satlink hardware provides continuous transmission and generates an idle sequence of (scrambled) zeros between transmissions. For bit synchronisation of the Client hardware and software, a unique 32-bit sequence is transmitted twice during the first field of the Satlink frame. Following, this a 'length field' is transmitted which communicates to the Client PC's driver software the number of bytes in the frame that follow in the data payload. To avoid false synchronisation, the device driver must employ byte stuffing to prevent the 32-bit sequence from occurring again until the next SatLink frame is transmitted. The SatLink frames are written to a parallel buffer in the transmit card and transformed into a serial bitstream so that encryption can be applied. The encrypted bitstream is applied to a differential phase shift keying (DPSK) modulator which produces a modulated data sub-carrier at 7.74MHz. A 7.74MHz sub-carrier is frequently used to transmit additional audio channels alongside a standard FM TV signal; in this case it is a data sub-carrier which is transmitted.

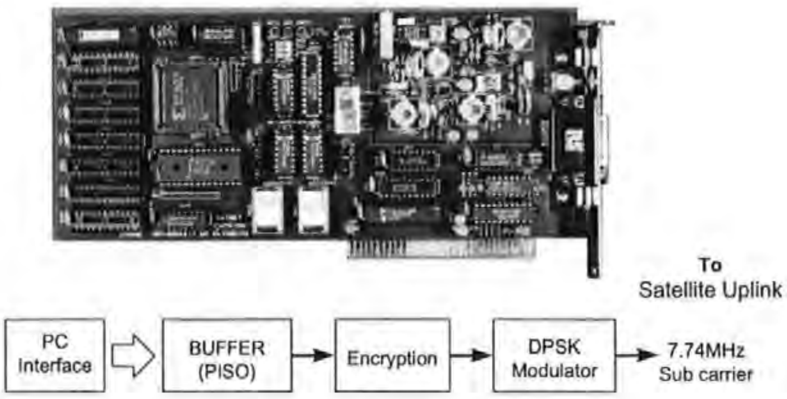


Figure G-3. Data broadcast PC interface.

The Client's data broadcast receive interface card, Figure G-4, accepts the decoder video (baseband) signal from a domestic TVRO (TeleVision Receive Only). The receive card contains a digital demodulator to extract the transmitted data stream from the 7.74MHz data sub-carrier. After decryption, the receive hardware is synchronised by the first 32-bit sequence (added by the transmit software) so that data written to the parallel FIFO buffer has correct byte alignment. For greatest efficiency the recovered data is written to PC's memory using direct memory access (DMA). The device driver reads the incoming data stream and uses the 32-bit sequences to identify the start of each SatLink frame. After reversing any bit stuffing that was performed, the device driver extracts Ethernet Frames and passes them to the Internet Protocol Suite where unwanted IP datagrams are discarded; based upon destination IP address.

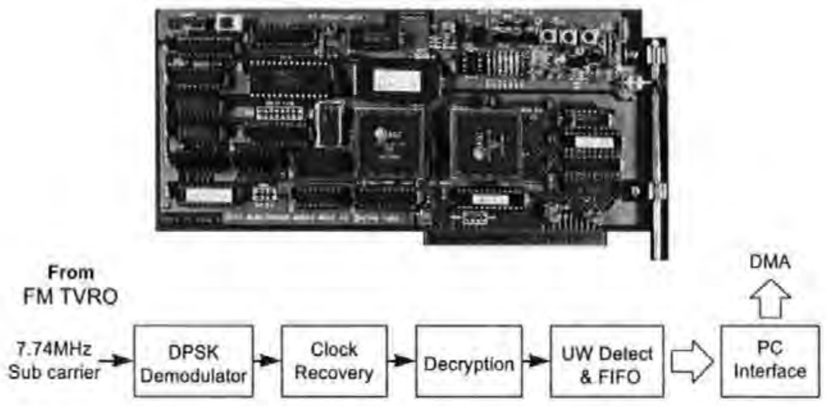


Figure G-4. Data broadcast receive interface card.

Both Client and Host network device driver software provided Ethernet emulation but with respective modifications for a 'receive only' and 'transmit only' links. For the Client device driver, since the data broadcast receive card provides no hardware filtering of incoming packets, all datagrams are received. Given that the transmitted data rate is relatively low (90kbps), it was decided that IP could be used to discard unwanted datagrams with negligible processing overhead. For the Host transmit driver, the main task was spoofing responses to ARP requests, generated by the Internet Protocol Suite, so that transmissions would take place.

G.2 An Asymmetric Satellite Internet Delivery System

A natural progression from the system described in section F was to investigate the feasibility of an Internet Delivery System provided completely by satellite. Perceived benefits to the Internet Service Provider were mainly that the location of potential users is only limited by the satellite's footprint or coverage area. To the user, the main benefit is an effective permanent connection to the Internet since there is no logging on or logging off procedure. A second system investigation was conducted in association with BNSC (The British National Space Council) with the following objectives;

1) Investigate the feasibility of an asymmetric Satellite Internet Delivery System.
2) Demonstrate prototype hardware and software.
3) Identify areas for further investigation.

The second investigation was conducted using the University of Plymouth as the Host Internet Service Provider and Satellite Earth Station, Figure G-5. Initially Client terminals were located at the University of Plymouth and finally at a number of adult learning centres around the South West of England for evaluation. During these investigations, the author's significant contributions included novel network device driver software, IP network design, system configuration and system evaluation. Further information relating to novel satellite software and hardware may be found in the following sections and within the author's publication "A Novel Internet Delivery System Using Asymmetric Satellite Channels" (Appendix J). A brief system overview is given below.

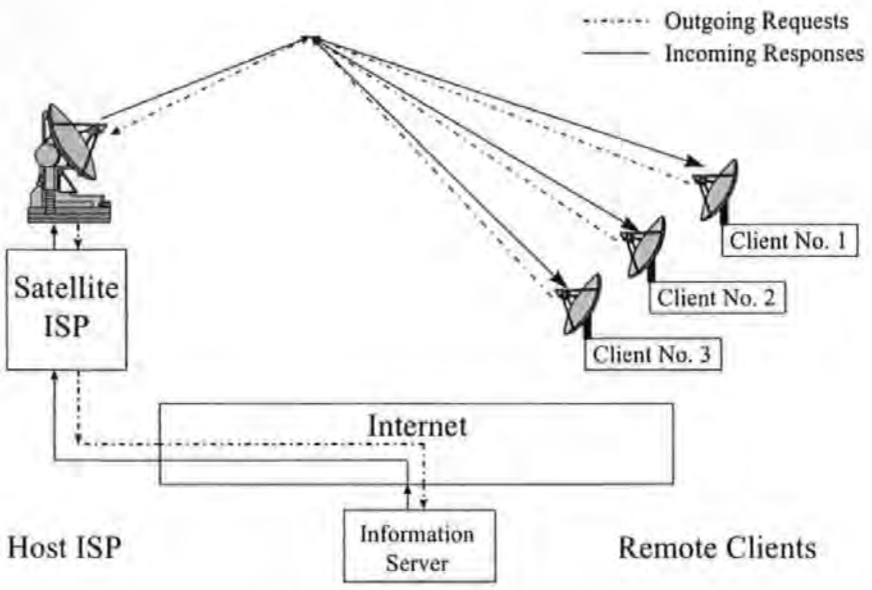


Figure G-5. Asymmetric satellite Internet delivery system - system overview.

With reference to Figure G-5, each client terminal is provided with satellite hardware for transmission on a shared request channel (16kbps burst-mode) and reception of the data broadcast channel (2048kbps). The request channel was provided by the satellite return link system described in earlier chapters, and with terminals accessing the channel on an ad-hoc basis; no channel sharing protocol was employed. The satellite data broadcast channel utilised a standard DVB transport stream with proprietary hardware to change the response channel data rate progressively from 128kbps up to 2048kbps. In terms of a network diagram, Figure G-6, each client is pre-assigned an IP address from subnet 141.163.54.0 of IP network 141.163.0.0. Communication in both directions is conducted via the 'Host PC', which provides a gateway between satellite clients and the 'Host ISP'. Routing tables at the 'Host PC' and 'Host ISP' ensure that responses from the Internet are delivered back to the Clients where IP datagram filtering is performed by low-level software based upon MAC addresses; MAC address and IP address associations are maintained by the 'Host PC'. It is of particular significance that both Client and Host PC terminals contained hardware for two fundamentally different satellite channels, yet are assigned just one IP address. To achieve this, novel Network device driver software was implemented so that Ethernet emulation was achieved over the satellite channels; ie. two uni-directional satellite channels are made to appear as a standard bi-directional Ethernet connection.

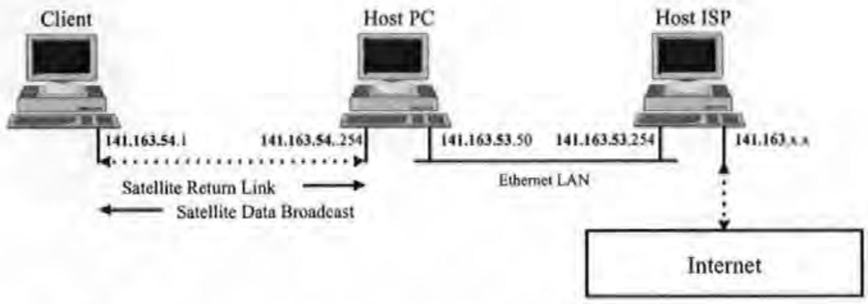


Figure G-6. Asymmetric Internet delivery system - network diagram.

An investigation into the feasibility of a low-cost satellite Internet delivery system provided a valuable insight into what further study would be required. A fundamental problem was experienced due to TCP/IP's intolerance of the large delays provided by two satellite links. This particular topic has recently received a great deal of attention in the research world and many novel solutions based upon TCP/IP spoofing and intelligent proxy servers had been proposed. The most significant problem identified was collisions on the request channel which resulted in a range of symptoms from total failure to a slight reduction in throughput. A dedicated request channel for each user is not efficient or viable, but to serve multiple active users it was clear that many return channels were required.

G.2.1 Data Broadcast Hardware

For transmission of TCP/IP datagrams over the DVB satellite broadcast channel the Host PC contains a modified SatLink (reference to paper) data broadcast PC interface card, Figure G-7; a similar transmit card has already been described in section G.1.1. In this case the encrypted bitstream is made available in an RS-422 'clock and data' format and interfaces with a DVB MPEG-2 encoder/multiplexer.

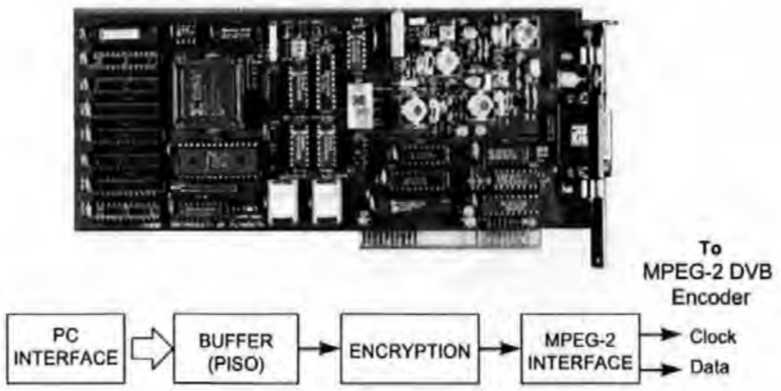


Figure G-7. Host PC data broadcast PC interface card.

The transmitted signal is received by a MPEG-2 DVB decoder/de-multiplexer, and the encrypted bitstream applied to a modified Satlink receive PC interface card, Figure G-8; a similar transmit card has already been described in section G.1.1. In this case the device driver filters incoming frames, thus discarding unwanted datagrams more efficiently; filtering is based upon the destination MAC address of the Ethernet frame encapsulated within each SatLink frame.

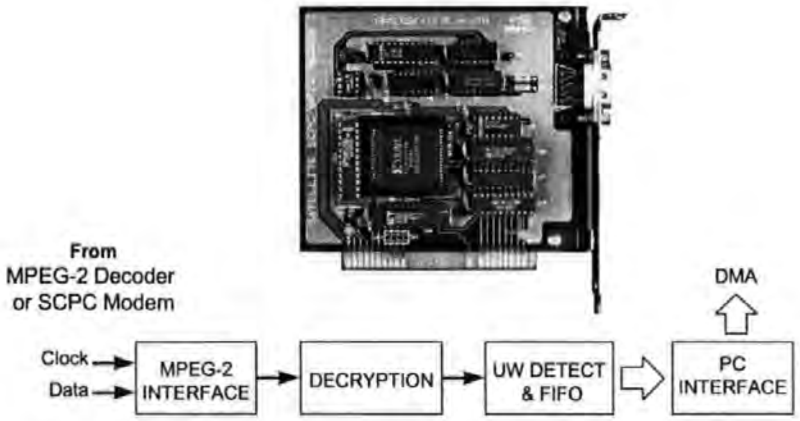


Figure G-8. User MPEG-2 receive PC interface card.

G.2.2 User Return Link Hardware

The digital interface of the prototype user return link transmit card was used in conjunction with an external Ku-Band BPSK modulator and power amplifier for the request channel, Figure G-9. A number of interlock signals (not shown) switch the external 14GHz outdoor equipment on for the duration of the transmission burst.

Ethernet frames for transmission were encapsulated within Return Link packets by the device driver and written to the card for burst transmission. A generic Return Link packet format was utilised but with a minor modification to distinguish network traffic from proprietary control traffic.

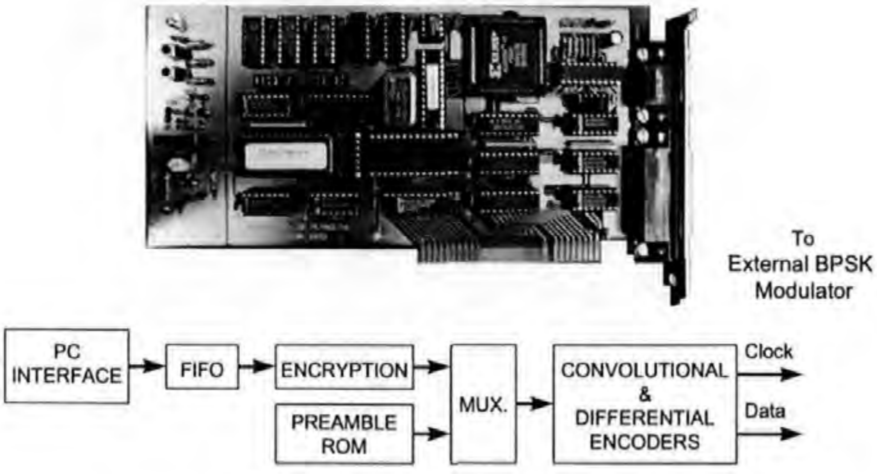

To External BPSK Modulator
PC INTERFACE
FIFO
ENCRYPTION
PREAMBLE ROM
MUX.
CONVOLUTIONAL & DIFFERENTIAL ENCODERS
Clock
Data

Figure G-9. User Return Link PC interface card.

The 14GHz outdoor equipment consists of a 90cm TVRO antenna equipped with a direct 14GHz BPSK modulator and 200 mW transmitting power amplifier produced (by others) for this investigation, Figure G-10.

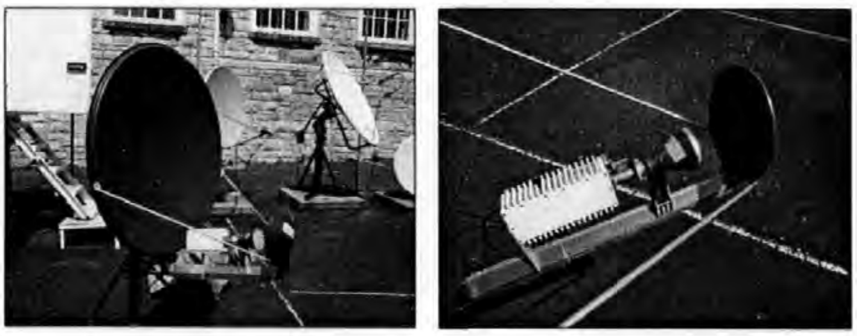


Figure G-10. User 14GHz outdoor equipment.

H. Paper 1 - "A Data Reply Link System for Satellite TV Applications".

A Data Reply Link System for Satellite TV Applications

J.T. Slader, P.M. Smithson, and M. Tomlinson

The Satellite Communications Research Centre, SECEE, University of Plymouth, England, U.K.

ABSTRACT - A low cost satellite data reply link is presented which may be used to interactively demand the broadcast of data or video "Direct to Home". Due to the burst nature of the reply packets, rapid acquisition to the signal is necessary. Frequency detection, demodulation and clock recovery algorithms are presented which have been successfully implemented in real time using Digital Signal Processing (DSP). Signal acquisition, demodulation and clock recovery synchronisation are achieved within just 1ms (32 symbols), allowing message data to be successfully decoded for forward error correction and off-line processing.

I. INTRODUCTION

The use of MPEG-2 compressed digital TV will allow increased TV channel capacity offering many more TV services than that used by conventional analogue broadcasting. These new broadcast services "Direct to Home" can include video on demand, armchair shopping facilities and many other interactive services. For user interaction, clearly a return link to the service provider is required. Terrestrial telephone networks offer inferior return link quality, while the cost of conventional VSAT technology is prohibitive. This project is concerned with the provision of a low cost data reply channel using a conventional TVRO antenna but with a modified feed horn, equipped with a small transmitting power amplifier. The data return signal is placed adjacent to the TV broadcast carrier and is capable of sustaining a data rate of 16Kbps, which is suitable for computer based data replies or voice communications. The forward data channel is provided by means of an operational data broadcasting system that has been developed by the University of Plymouth and features a SatLink PC based receive card [1].

A system overview is shown in Fig.1. Voice, fax or binary data files are output from the users set top unit, using serial binary data, to the outdoor unit (synthesiser, BPSK modulator and 200mW power transmitter) which is located in the antenna feed. The serial binary data stream includes configuration data for programming a Direct Digital Synthesiser (DDS) to set the return link transmission frequency within the Ku band. To assist rapid acquisition, the transmitted message is preceded by a short preamble of 100 symbols, comprising of phase reversals and a Unique Word (UW). The phase reversals are used by the hub station receiver to detect the presence of a signal, provide frequency correction (demodulation) and symbol timing recovery. The UW is used to initialise (ramp-up) a 1/2 rate convolutional decoder whilst also providing decoder synchronisation.

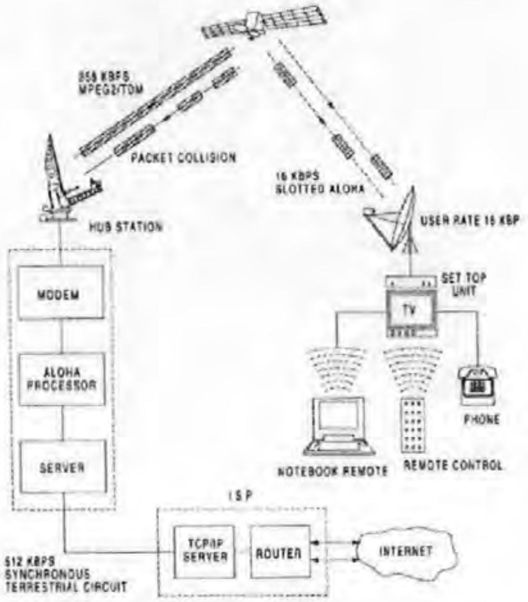


Figure 1 Return Link Slotted ALOHA System Overview

II. THE USERS SET TOP UNIT

The prototype set top unit consists of a PC (personal computer) with a remote control keyboard and mouse. Located within the PC are two ISA (Industrial Standard Architecture) compatible satellite modem cards which operate in a multitasking environment under Microsoft Windows 95. The inbound data channel uses a SatLink receive card [1] while the return channel is provided by a Data Reply Link transmit card. A block diagram of the transmit card is shown in Fig.2.

The transmit card provides two outputs, a 70MHz I.F. for connection to a conventional VSAT and a V11 interface for connection to the Return Link outdoor unit. The transmit card has a flexible architecture based around an FPGA (Field Programmable Gate Array) allowing various signal processing options to be enabled under software control. The user's message is encrypted, where the encryption algorithm is secured within a PLD (Programmable Logic Device), providing a first level of data security.

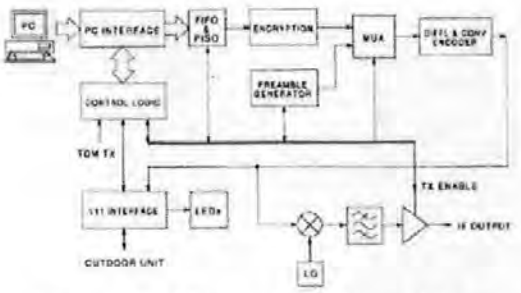


Figure 2 Block Diagram of User Transmit Card

A transmit sequence is initiated under software control or alternatively in hardware with a TDMA (Time Division Multiple Access) pulse. The preamble comprising of phase reversals and a unique word, are clocked serially out of a ROM (Read Only Memory) for direct transmission. The software application writes the message data to a 4KByte FIFO (First In First Out memory) which also provides parallel to serial conversion. When transmission of the preamble is almost complete, data bits are clocked out of the FIFO into the FPGA which encodes symbols concatenate to the preamble transmission. The software continues to top-up the FIFO for messages larger than 4KByte until the entire message has been transmitted. When the message has propagated through the encoders and transmission is complete, the transmitter is switched off and the software application prepares for the next message to be transmitted.

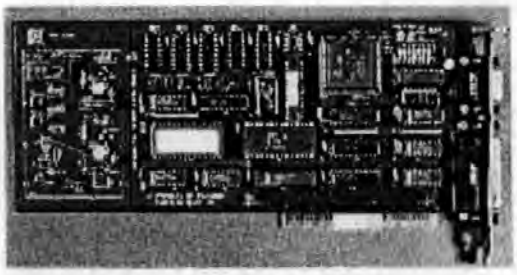


Figure 3 User Transmit Card

III. THE HUB STATION RECEIVER

The received signal is down-converted to a 64kHz I.F. where it is applied to a DSP board and sampled at 256kHz by a 12 bit ADC (Analogue to Digital Converter). The DSP hardware consists of a single extended 6U multi-layer printed circuit board, containing two 80MHz TMS320C50 fixed point digital signal processors, jointly providing 80 MIPS (Million Instructions Per Second). The frequency acquisition algorithm is computationally intensive and utilises both processors. Once frequency acquisition has been successfully achieved the processors are released, to perform modem and decoding algorithms. A single processor runs algorithms for frequency correction (demodulation), symbol timing recovery and unique word synchronisation, while the second processor provides phase tracking, forward error correction and a buffered interface to the hub processor for off-line processing.

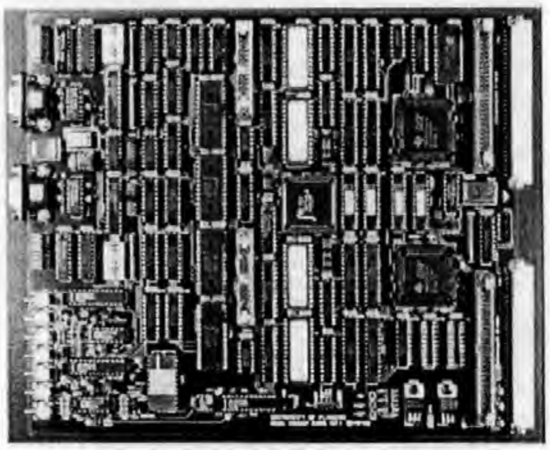


Figure 4 Hub Station Receiver DSP Hardware

IV. FREQUENCY ACQUISITION

Due to oscillator tolerances and propagation over the satellite link, the received signal can deviate from the nominal 64kHz I.F. by as much as 10kHz. Frequency Acquisition is the process of detecting and estimating a carrier frequency, whereupon frequency correction (demodulation) can be applied. The term 'Frequency Correction' is used because a variable frequency local oscillator is used for demodulation.

The Fast Fourier Transform (FFT) algorithm is often used as an efficient method of evaluating the Discrete Fourier Transform (DFT). The frequency domain representation provides a convenient means of detecting and estimating a carrier frequency. At low Signal to Noise Ratios (SNRs) there are two factors that limit the performance of the FFT.

1. When carrier power is shared equally between two frequency bins the signal can be hard to detect above background noise and imposes a signal detection threshold.
2. The resolution of carrier frequency estimates are limited to half the frequency bin spacing.

The Offset Fast Fourier Transform (OFFT) offers superior signal detection and frequency acquisition performance compared to the standard FFT at low SNR [4]. Equations for the DFT and the modified 'Offset DFT' are given below;

DFT:

$$F(k)=\sum_{n=0}^{N-1} x_n W_N^{nk} \qquad k=0,1,..,N-1 \qquad ..... (1)$$

Offset DFT:

$$F(k+c)=\sum_{n=0}^{N-1} x_n W_N^{n(k+c)} \quad k=0,1,..,N\text{-}1 \quad c=0.25 \quad .....(2)$$

In practice, a standard algorithm can be used for both the FFT and OFFT since only the coefficients change.

Comparing the power spectra of the FFT and OFFT demonstrates how the OFFT can give superior signal detection performance. Fig. 5a and Fig. 5b show power spectra of a carrier applied to a 64-point FFT and 64-point OFFT (c=0.25) respectively.

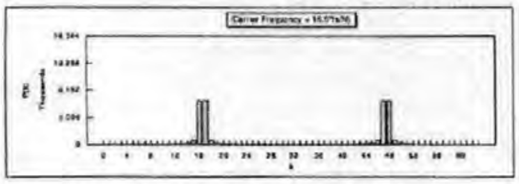


Figure 5a 64-Point FFT Power Spectrum

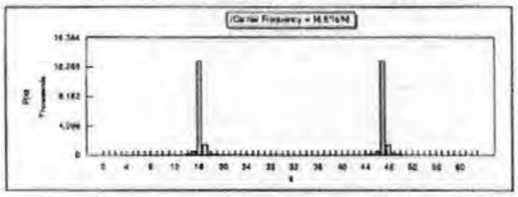


Fig. 5b 64-Point OFFT Power Spectrum (offset c = 0.25)

The FFT power spectrum is symmetrical and shows the carrier power shared equally between two frequency bins. The OFFT power spectrum, for the same input, is not symmetrical since a frequency offset has been introduced. The majority of the power now appears in a single frequency bin. The 'worst case' peak power, as shown in Fig. 5a and Fig. 5b, is increased by a factor of 50% thus significantly improving the signal detection threshold.

The OFFT can also be used to derive a more accurate carrier frequency estimate than the FFT, within $0.25f_s/N$ and $0.5f_s/N$ respectively, at low SNR. Closer inspection of Fig. 5b reveals that the Power Spectrum is offset by $-0.25f_s/N$ in the range k=0 to 31 and offset by $0.25f_s/N$ in the range k=32 to 63.

Using the following algorithm the carrier frequency can be found:

Find the frequency bin k_{max} containing greatest power .. (3a)

If, $k_{max} < N/2$ $f_{carrier} = (k_{max} + 0.25)*f_s/N$ Hz ...... (3b)

If, $k_{max} => N/2$ $f_{carrier} = (N - k_{max} - 0.25)*f_s/N$ Hz .... (3c)

In this case $f_{carrier} = 16.25f_s/N$ Hz or $f_{carrier} = 16.75f_s/N$ Hz, both having a residual frequency error equal to a quarter of the frequency bin spacing ($f_s/4N$ Hz). A similar estimate derived from the FFT power spectrum in Fig. 5a would yield $f_{carrier} = 16f_s/N$ Hz or $f_{carrier} = 17f_s/N$ Hz, both having a residual frequency error equal to half the frequency bin spacing ($f_s/2N$ Hz). Residual frequency errors manifest as phase errors in the recovered data and are removed using phase tracking techniques or differential demodulation. It is desirable to minimise the phase error by generating the most accurate frequency estimate.

V. HUB STATION RECEIVER OPERATION

Transmitted messages are preceded with a 32kHz phase reversing preamble to aid frequency acquisition and symbol timing recovery. After performing a 256-point OFFT on the incoming samples the receiver examines the power spectrum. The preamble appears as two peaks separated by 32kHz and centred around the carrier frequency as shown in Fig. 6. If a suitable SNR (Signal to Noise Ratio) is exceeded, frequency acquisition is declared and the local oscillator is set using the carrier frequency estimate.

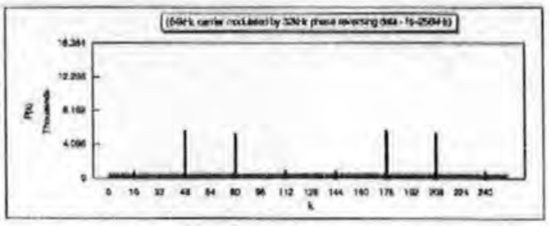


Figure 6 Power Spectrum of the Reply Link Preample

The input sequence x_i is frequency corrected to form the frequency corrected sample sequence a_i+jb_i as shown by equations (4) & (5);

$$a_i + jb_i = x_i \times e^{-j\frac{2\pi i f_{correction}}{N}} \quad ..... (4)$$

$$a_i + jb_i = x_i \times \cos(\frac{2\pi i f_{correction}}{N}) - jx_i \times \sin(\frac{2\pi i f_{correction}}{N}) .... (5)$$

The maximum residual frequency error after frequency correction is equal to $f_s/4N$ Hz. Since f_s = 256kHz and N = 256, the residual frequency error is 250Hz. At a symbol rate of 32,000 symbols per second the maximum phase error, due to the residual frequency error, is given by equation (6);

$$phase\_error = \frac{360 \times 250}{32000} = 2.8125^{\circ} \quad ...... (6)$$

In keeping with the rapid acquisition and synchronisation targets, the phase reversal symbols that caused the receiver to 'acquire' are also used to begin symbol clock recovery. After frequency correction the signal is applied to the clock recovery algorithm and a stable symbol clock is produced within the first 32 symbols. Prior to Forward Error Correction (FEC) the decoder is initialised and synchronised by detecting the Unique Word at the end of the preamble. Within 1ms (32 symbols) of receiving the start of the preamble, frequency acquisition, frequency correction and symbol lock recovery are achieved. Once the end of the preamble (100 symbols) has been received forward error correction (FEC) begins.

VI. SYMBOL TIMING RECOVERY

A novel DSP based clock recovery scheme demonstrates superior performance compared to that of a PLL; particularly in terms of clock acquisition [2,3]. The scheme exploits the feature of a long impulse response in an IIR (Infinite Impulse Response) filter, when the poles are close to the unit circle. In the context of clock recovery this feature is desirable since it can be exploited to provide a good flywheel effect over periods of lost input; i.e. a fade. The clock recovery system and associated waveforms are shown in Fig. 7

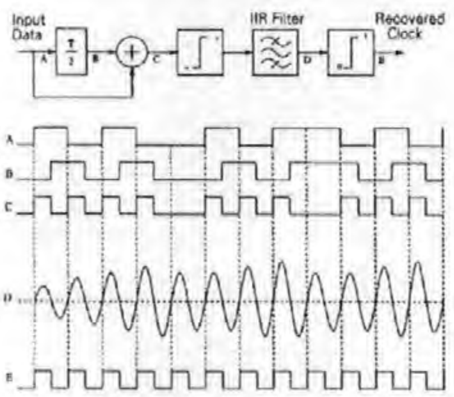


Figure 7 Clock Recovery Scheme

Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible stimulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at it's tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E; the recovered clock output.

An IIR filter is normally formed as a second order section containing both poles and zeros, this provides a centre frequency and stop-band nulls. In this application, however, the stop-band null is of no consequence and the design can be simplified to have feedback paths only.

The DSP-based clock recovery system has been simulated on Alta SPW (Signal Processing Worksystem) operating on CAD workstations. Examples of simulation results are shown in Figs. 8 & 9. A T/2 pre-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing. After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronisation is achieved immediately as the output is in phase with the applied data.

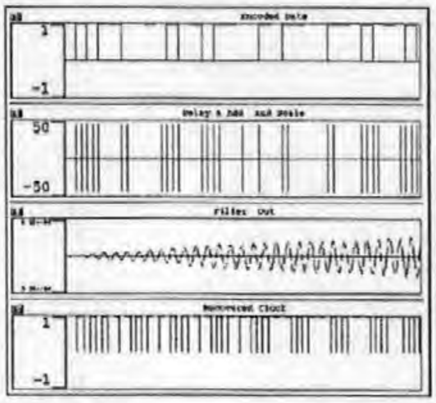


Figure 8 Clock Recovery Acquisition Performance

Fig. 9 shows the flywheel performance over a 40-bit data fade. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock.

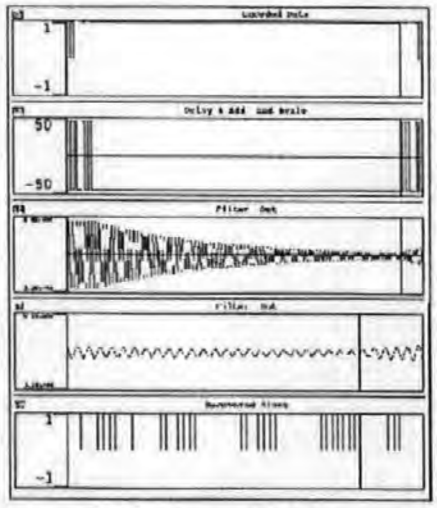


Figure 9 Clock Recovery Fade Performance

Zooming in to the end of the fade shows the recovered clock remaining in phase when data are reapplied; this represents a flywheel performance of 80 clock periods. This flywheel performance can be further improved by cascading IIR filter sections where each section approximately doubles the flywheel performance, i.e. the output of first filter section decays to zero before the stimulus to the second filter is affected.

VII. SATELLITE TRIALS AND TEST RESULTS

A series of satellite trials have been successfully completed using EUTELSAT II F3 widebeam at 7° East (ESA's Digilease transponder). The uplink used vertical polarisation and the downlink horizontal. Satellite reception and monitoring was provided by ESA's TDS4B Satellite Earth Station situated at the University of Plymouth. Known data packets were transmitted by the return link terminal and received on TDS4B where the signal was down-converted to 70MHz. A series of back-to-back tests were also carried out at 70MHz I.F. The 70MHz I.F. was applied to the return link electronics where the signal was demodulated, decoded and output to a PC where the BER (Bit Error Rate) performance was measured. The BER performance is plotted in Fig. 10 and compared to theoretical results. It can be seen that for the back-to-back test there is approximately 0.7dB degradation from the theoretical, while the satellite test introduced a further 0.3dB degradation.

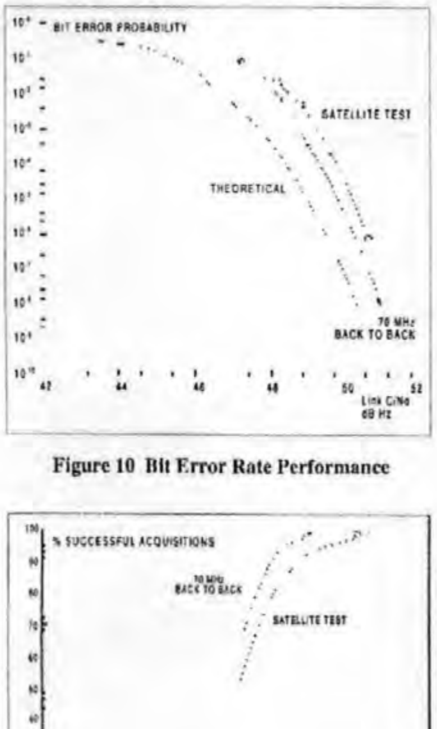


Figure 10 Bit Error Rate Performance

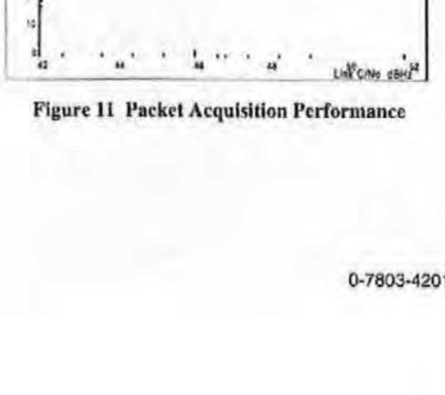


Figure 11 Packet Acquisition Performance

The packet acquisition performance is plotted in Fig. 11. For an acquisition probability of 99% or better, an Eb/No ratio of 6.8dB is required for back-to-back test while approximately 8dB is required to guarantee successful acquisition over the satellite.

VIII. CONCLUSIONS

The satellite data reply link described, operates at low power using the same transponder as the broadcasting service, occupying just a small part of the unused bandwidth. The system operates at low signal to noise ratios at a data rate of 16Kbps employing an efficient 1/2 rate FEC algorithm. A short synchronisation preamble of just 100 symbols is applied enabling rapid signal acquisition, symbol timing and decoder synchronisation. The system is tolerant of frequency deviation of up to 10kHz and employs differential demodulation or phase tracking to remove residual phase errors. The rapid acquisition techniques described increase the probability of successful message decoding and lend themselves to a polling or TDM type access. With an increase in TV channel capacity many more broadcast channels will become available and these will include armchair pay-on-demand "Direct to Home" services. A low cost satellite data reply link will satisfy these interactive requirements. Applications include video on demand, home shopping, interactive data as well as other tele-educational markets.

A series of user trials operating application software is planned for late 1997.

IX. REFERENCES

[1] Smithson PM, Tomlinson M, "The Development of an Operational Computer Based Satellite Data Broadcasting System", *IEE International Conference on Digital Satellite Communications ICDSC-10 Proceedings ISBN 0-85296-6393, V2, Page 405. 1995*

[2] Smithson PM, Tomlinson M and Donnelly T, "DSP-Based Clock Recovery for a Digital Magnetic Recording Channel", *IEEE Globecom'94 Proceedings ISBN 0-7803-1820-X, V3, Page 1467. 1994.*

[3] Smithson PM, Tomlinson M and Donnelly T, "DSP-Based Clock Recovery Implemented in a Field Programmable Gate Array", *IEE Colloquium Proceedings New Synchronisation Techniques for Radio Systems ISSN 0963-3308, Ref No: 1995/220. 1995.*

[4] Tomlinson M, "The Offset Fast Fourier Transform," *MOD SRDE Report No: 76025. 1979.*

X. ACKNOWLEDGEMENTS

The authors would like to thank the European Space Agency and Armstrong Electronics for their support during the development of this project. Thanks are also extended to the University of Plymouth satellite uplink staff for their co-operation during satellite trials.

I. Paper 2 - "Development of an Operational Satellite Internet Service".

The Development of an Operational Satellite Internet Service Provision

P.M. Smithson, J.T. Slader, D.F. Smith and M. Tomlinson

The Satellite Communications Research Centre, SECEE, University of Plymouth, England, U.K.

ABSTRACT - The development of a low cost operational Satellite Internet Service Provision (SISP) is presented, which delivers the full functionality of the internet to clients, who due to their geographical location may only have access to a slow public telephone network. Return packets are broadcast on a satellite TV channel, at data rates up to 2Mbps. The clients receive hardware consists of a standard satellite TV receiver (analogue FM or digital MPEG-2) and a special SatLink card which plugs directly into the clients personal computer. Satellite trials have been successfully completed and the system is expected to operate commercially with effect from mid 1997.

I. INTRODUCTION

It is well known that modern domestic internet traffic is asymmetric in nature, there are many more packets coming into client machines than going out. The ratio can be greater than 10:1 for clients accessing the World Wide Web (WWW). The Internet Service Provider (ISP) will often find that this imbalance leads to inefficient use of connection bandwidth.

The Satellite Internet Service Provision (SISP), allows a clients outgoing packets, which are relatively few in number, to be routed via the slow telephone network to a satellite uplink which has access to the internet backbone. The return packets are then delivered at high data rates using a satellite TV broadcast channel, resulting in much faster reception of complex WWW pages and other internet services. The users receive hardware consist of a standard satellite television receiver and a special SatLink demodulator card [1] which plugs directly into the clients personal computer. SatLink is an established operational satellite data broadcasting system, developed by the University of Plymouth.

The broadcast channel can use analogue television (FM TV), where data are transmitted at 90Kbps utilising a 7.74MHz audio sub carrier bandwidth slot. Alternatively, a digital MPEG-2 broadcast channel can be used where the data rate is between 128K and 2048Kbps.

The satellite internet data bandwidth is shared between all clients, but the cost is significantly lower than buying further bandwidth over a leased line. SISP was originally designed as a solution to provide an ISP in a remote country that has a low-quality Public Switched Telephone Network (PSTN), where local analogue exchanges may only offer 1200 to 9600bps to standard telephone modems.

II. SYSTEM OVERVIEW

A block diagram of the SISP system is shown in Figure 1, with an internet connection in the UK and the ISP clients located in the Middle East.

A client in the Middle East dials into the Local ISP to establish a Point to Point Protocol (PPP) connection. All traffic to and from the Local ISP is via PSTN modems. If the clients web browser requests a connection to an external machine, the IP packets are routed, via a 64Kbps leased line, to a commercial Host ISP located in the UK. Return packets are modulated and transmitted via the SISP satellite data broadcast system and received on the clients satellite TV receiver. If FM TV is used as the broadcast medium, then the standard baseband output from a domestic satellite receiver is connected to a SatLink demodulator card located within the clients PC. Alternatively, in the case of an MPEG-2 digital TV transmission, the internet data are demultiplexed from the MPEG-2 transport stream within the receiver/decoder and synchronous clock and data are connected to an MPEG version of the SatLink card, again located within the clients PC.

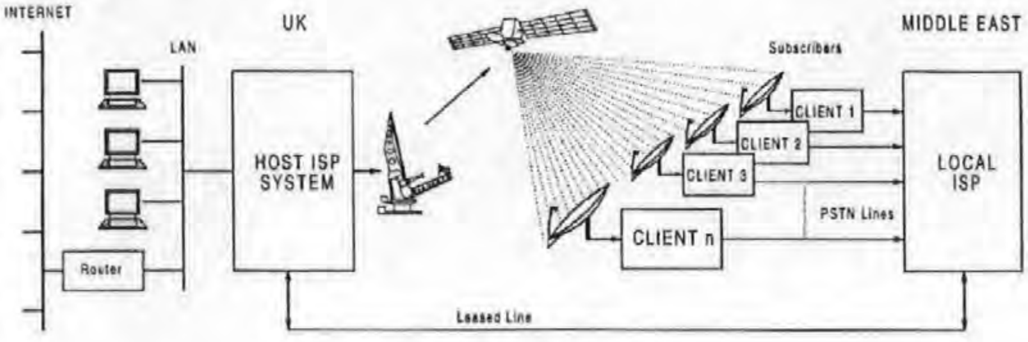


Figure 1 SISP System

The SatLink receive cards pass the data stream to a special Windows device driver which regenerates the Internet Protocol (IP) packets, completing the cycle.
The system is controlled at the Local ISP, which provides authentication and accounting information in addition to operating modem banks for up to 100 clients (additional banks can be added by networking additional PC's).

III. THE HOST ISP

An overview of the hardware and software for the Host ISP is shown in Figure 2.

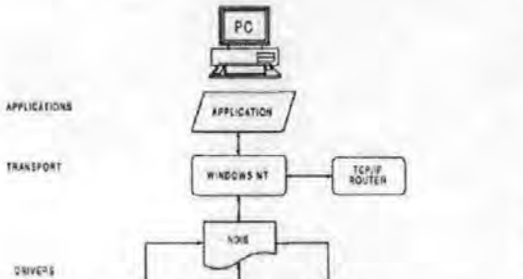


Figure 2 Host ISP System

The transmit software drivers provide an interface between the TCP/IP protocol and the SISP transmit hardware. Due to the broadcast nature of the transmission, the transmit driver contains some features that are not usual to other network packet drivers. In order to maintain maximum compatibility, the driver has been written to appear to the Windows-NT/95 operating system, as a receive-only or transmit-only Ethernet card. Although the drivers themselves appear to the Network Device Interface Specification (NDIS) layer as Ethernet cards, much of the unnecessary header information has been stripped from the transmitted data. The transmit driver operates a proxy-Address Resolution Protocol (ARP) policy for remote clients to remove more overhead. In addition, the 12-byte Ethernet addresses have been stripped from the header, and all packets are delivered on the basis of IP address alone. However, the 4-byte error detection block is maintained in a modified form, to preserve good data integrity. There are also extensions build into the satellite protocol to allow for packet compression, forward error correction and statistical information packets, should the need arise.
The clients SatLink receive hardware is required to reassemble the received serial bit stream back into bytes, also clients must be allowed to start receiving at any time during a broadcast. To achieve this a 32-bit Unique Word (UW) is transmitted periodically to reset the synchronisation counters within the SatLink receive card. Clearly the UW must not occur in the message sequence to avoid false synchronisation, hence any unique words that appear in the data stream are remapped by the transmit driver by means of "bit stuffing". The clients receive driver recognise when a sequence has been remapped and restores the original data.
Figure 3 provides a block diagram of the SISP transmit card.

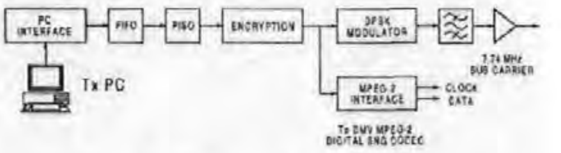


Figure 3 SISP Transmit Card Block Diagram

The transmit driver writes bytes for transmission to a 4Kbyte First In First Out (FIFO) memory which also provides Parallel In Serial Out (PISO) conversion. The serial data stream is scrambled with an encryption algorithm secured within a Programmable Logic Device (PLD), the SatLink receive cards located within the Client PC's must have an identical PLD in order to successfully decode the data. The encryption code is run length limited to break up long sequences of binary ones or zeros in the data stream, to assist clock recovery in the receiver. The data rate can be adjusted by means of onboard links to select 90Kbps (for sub carrier applications) or 128K to 2048Kbps (for MPEG-2 applications).
The transmitted data stream from the card is modulated using Differential Phase Shift Keying (DPSK) onto a 7.74MHz sub carrier and transmitted along with a standard satellite TV signal. The uplink signal is therefore the same as for normal FM TV signals except for the addition of this extra data sub carrier in a spare part of the channel spectrum. Alternatively, synchronous clock and data are available (RS422 and RS232 interfaces) for connection to an MPEG-2 Digital SNG Codec for multiplexing into an MPEG-2 transport stream.

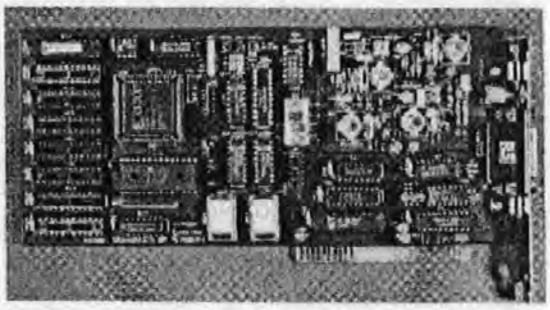


Figure 4 SISP Transmit Card

IV. THE CLIENT SYSTEM

An overview of the hardware and software for the Client System is shown in Figure 5.

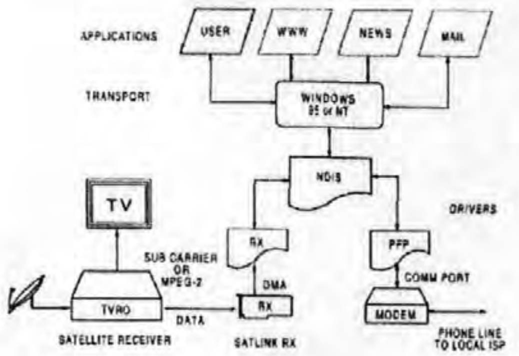


Figure 5 The Client System

The Clients receive (demodulator) card accepts the standard decoder video (baseband) signal from a domestic TVRO (TeleVision Receive Only) receiver.
The video and audio outputs from the TVRO may also be connected to a television and simultaneous TV programmes can be viewed without interference from the data modem.

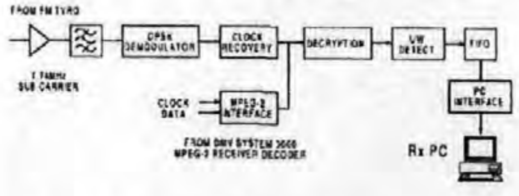


Figure 6 The SatLink Receive Card Block Diagram

The SatLink sub carrier receive card contains a digital demodulator to extract the data sub carrier and recover the clock and data. A novel clock recovery scheme has been implemented digitally as a multiplier-less recursive filter utilising a field programmable gate array [2,3].
The hardware reassembles the data stream into valid bytes which are stored into the PC's memory at high speed using Direct Memory Access (DMA), under control of the receive driver.
The SatLink card has been designed using a digital receiver and demodulator thus eliminating alignment and drift problems normally associated with analogue electronics. Programmable logic has been extensively utilised, enabling hardware design flexibility at reduced board size and cost.

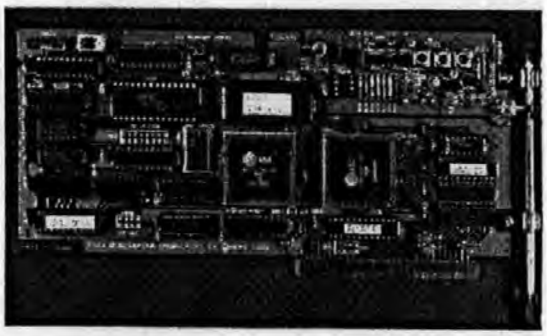


Figure 7 Sub Carrier SatLink Receive Card

An alternative card has also been developed for MPEG-2 applications, where synchronous clock and data (RS422 interface) are received from an MPEG-2 Receiver/Decoder which demultiplexes SatLink data from the MPEG-2 transport stream. This interface is also suitable for connection to a conventional Single Channel Per Carrier (SCPC) satellite modem.

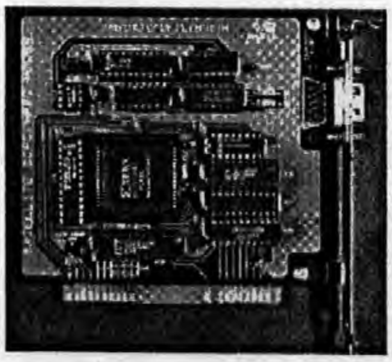


Figure 8 MPEG-2 SatLink Receive Card

The Receive Driver provides the interface between the SatLink receive hardware and the TCP/IP protocol driver. The clients receive driver recognises when a UW sequence has been remapped and removes the bit stuffing to restore Ethernet frames, passing them up to the TCP/IP protocol driver. Ethernet hardware addresses are not transmitted over the satellite link to improve efficiency, so dummy hardware addresses are generated by the receive driver. Since the transmitted data rate over the sub carrier system is relatively low (90Kbps), IP packet filtering is provided by the TCP/IP protocol. At higher data rates the SatLink card must filter IP packets to prevent the clients PC from being swamped by packets destined for other users, although this has not yet been implemented.

V. THE LOCAL ISP

A block diagram of the Local ISP system is shown in Fig-9.

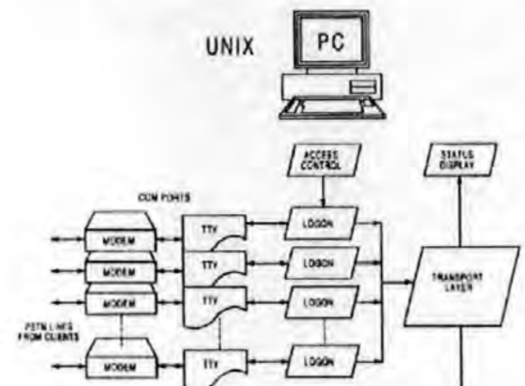


Figure 9 The Local ISP

The Local ISP system, operating under the UNIX operating system, is completely oblivious to the SatLink hardware. It is a standard PPP-based ISP system, where the packets from up to 100 modems are multiplexed down a leased line for connection to the host system located in the UK. Statistical information is displayed and stored providing a record of clients access times and packet usage, which may be used for accounting purposes. Other extensions have been implemented using a graphical X-Windows user interface, to allow client accounts to be added or removed by the local provider. The local ISP can also provide a useful service giving the status of the connections and other maintenance feedback, that may not come via the satellite link.

VI. SATELLITE TRIALS

A series of satellite trials have been successfully completed using EUTELSAT II F3 16^{o}E, widebeam transponder-20, receiving on a 1 metre dish at 11.575160GHz with vertical polarisation. The uplink facility was provided by ESA's TDS4B satellite earth station sited at Plymouth. The TV standard used was PAL and the power of the unmodulated video signal into the uplink modulator was set to 72dBW. A threshold extension TVRO receiver was used and errors were recorded for the period of the satellite trials. The signal level of the 7.74MHz data sub carrier at the TVRO decoder output was -30dBm and the signal to noise ratio was reduced to 22dB. With adjacent audio sub carriers placed at 7.56 and 7.92MHz, an average error rate of 10^{-7} was measured. Internet Explorer and Netscape were used to surf the internet and large files (8Mbyte) were successfully transferred to the Client PC's. Network performance and packet delay (500 to 800ms for ping test) were measured and found to operate as expected over a satellite link.

A series of similar satellite trails were also successfully performed using MPEG-2 DMV Digital SNG Codec and DMV System 3000 Receiver Decoder. The uplink was again provided by TDS4-B, transmitting at 62dBW on the same satellite but receiving with horizontal polarisation.

VII. COMMERCIAL IMPLEMENTATION

At the time of print the system was undergoing commercial installation and had been successfully installed and tested over a commercial satellite link.
The uplink is provided by BT at Madley which is located at Herefordshire in the UK. The SISP data is transmitted on a 7.74MHz sub carrier on the EBN (European Business News) TV channel. The TV channel carries six secondary sub carriers, two of which are data.
EBN is received on EUTELSAT's HotBird 1 at 13^{o}E at 11.265GHz with horizontal polarisation, and shares transponder 3 with a digital TV channel. The signal was received in the UK with 1.2 metre dishes and domestic satellite TV receivers. The signal level of the 7.74MHz data sub carrier at the TVRO decoder output was measured as -30dBm with a signal to noise ratio of 20dB. The adjacent audio sub carrier at 7.56MHz was measured as +3.8dB above the data sub carriers at 7.74 and 7.92MHz.
Error rate performance were monitored over a period of 3 weeks over diverse weather conditions and similar error performance was observed as for the trials.
The internet service providers located in Jordan are within the 40 to 45dB contour of Hotbird's Super-Widebeam footprint, and have successfully received EBN and internet data from the data sub carrier using a 1.8 metre dish. The system is expected to operate commercially with effect from August 1997.

VIII. CONCLUSIONS

The Satellite Internet Service Provision (SISP) allows the low cost delivery of an internet service by satellite, which removes the background load from the usual wired connections, freeing them up for page requests and outgoing mail messages. SISP can be used to provide a high speed connection to locations where only slow local analogue exchanges exist and may also provide a lower cost solution to leasing dedicated high speed leased lines.
The system is not limited to operating while clients are connected, since multicast IP packets are received whether the PSTN modems are operating or not. Although this has not yet been implemented, it allows for e-mail signalling and off-line news spooling. It also has some interesting possibilities for advertising in screen-saver programs.
When operating at 2Mbps, the clients terrestrial modems sending acknowledgment packets at 9.6Kbps, were found to limit the speed of reception to 160Kbps (20Kbytes/s). However the broadcast data

rate is shared between all clients, therefore at least 13 clients should be simultaneously on line and actively transfering data, in order to justify the extra bandwidth.

IP addresses are allocated to active modems, not to the receive cards, so because of the layering, the receive driver does not know its IP address. Hence the SatLink receive card has no choice but to pass every packet received to the operating system. This is not a problem at 90Kbps, but is a significant overhead for a 2048Kbps data stream. If the system proves to be popular, then extending the unique word to 48 bits or more, and assigning each card its own number, will allow hardware packet filtering.

IX. REFERENCES

[1] Smithson PM, Tomlinson M, "The Development of an Operational Computer Based Satellite Data Broadcasting System", *IEE International Conference on Digital Satellite Communications.ICDSC-10 Proceedings ISBN 0-85296-6393, V2, Page 405. 1995*

[2] Smithson PM, Tomlinson M and Donnelly T, "DSP-Based Clock Recovery for a Digital Magnetic Recording Channel", *IEEE Globecom'94 Proceedings ISBN 0-7803-1820-X, V3, Page 1467. 1994*

[3] Smithson PM, Tomlinson M and Donnelly T"DSP-Based Clock Recovery Implemented in a Field Programmable Gate Array", *IEE Colloquium Proceedings New Synchronisation Techniques for Radio Systems ISSN 0963-3308, Ref No: 1995/220. 1995*

X. ACKNOWLEDGEMENTS

The authors would like to thank Communicado Data Ltd. for commissioning this commercial project. Particular thanks to Mike Hendry and Phil Sabin for the initial concept and their contribution to the success of the project. Thanks are also extended to the University of Plymouth satellite uplink staff and ESA for their co-operation during satellite trials.

The Satellite Communication Research Centre is part of the School of Electronic, Communication and Electrical Engineering, University of Plymouth, Drake Circus, Plymouth, United Kingdom, PL4 8AA.

J. Paper 3 - "A Novel Internet Delivery System Using Asymmetric Satellite Channels".

IAF-98-M.5.06

A Novel Internet Delivery System Using Asymmetric Satellite Channels

J. Slader, P. Smithson, M. Tomlinson, A. Ambroze and J. Wade

The Satellite Communications Research Centre,
SECEE, University of Plymouth, Plymouth, Devon PL4 8AA, England.

49th International Astronautical Congress
Sept 28-Oct 2, 1998/Melbourne, Australia

Appendix J: Paper 3 - “A Novel Internet Delivery System Using Asymmetric Satellite Channels”.

A NOVEL INTERNET DELIVERY SYSTEM USING ASYMMETRIC SATELLITE CHANNELS

J.T. Slader, P.M. Smithson, M. Tomlinson, A.M. Ambroze and J.G. Wade

The Satellite Communications Research Centre,
SECEE, University of Plymouth, Plymouth, Devon PL4 8AA, England.

ABSTRACT - A novel two-way satellite Internet delivery system is presented. The system employs an MPEG-2 data broadcast channel to deliver data to the user and a domestic 90cm satellite antenna, equipped with a 200mW transmitting power amplifier, to provide the reply channel. The reply carriers occupy spare capacity of the satellite transponder at no extra cost to the service provider. A burst demodulator, for reception of the reply channels, has been implemented using digital signal processors. The operation and algorithms, including rapid carrier frequency acquisition and synchronisation, are discussed. Results and observations from successful satellite trials, associated with the launch of a commercial pilot-scheme within the UK, are presented.

I. INTRODUCTION

Domestic Internet traffic is asymmetric in nature. Complex Web pages generate heavy incoming data while only short request and acknowledgement packets are transmitted. This ratio often exceeds 10:1. A Satellite Internet Service Provision (SISP) system previously developed by the University of Plymouth[2] has outgoing packets sent by a terrestrial link while incoming packets are received at higher data rates using a satellite TV broadcast channel. The result is significantly faster downloads than with a standard Public Switched Telephone Network (PSTN) connection to an Internet Service Provider (ISP). The developments outlined in this paper replace the terrestrial return link with a satellite based data reply link[1].

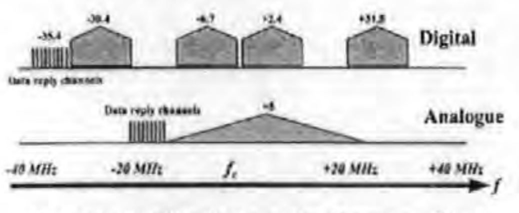


Figure 1 - Satellite Transponder Frequency Plans

The reply channels have both low power and bandwidth and occupy spare capacity of the satellite transponder; which has already been allocated to and paid for by the service provider. Figure 1 shows typical transponder frequency plans used during satellite field trials. The first is for MPEG-2 digital television and the second for analogue FM television. In both cases the reply channel carriers are placed close to the broadcast carrier without interference. Additional reply channels are added when user numbers grow and to allow different services to coexist on the same transponder. Reply channels may be prioritised so that users who elect to pay a premium will receive a faster service.

II. SYSTEM OVERVIEW

A system overview is shown in Figure 2. The Host ISP System is located at the satellite earth station. Client terminals may be placed anywhere within the footprint of the satellite. Clients send using 32ksps burst transmissions on the data reply channels. These transmissions are received by burst demodulators at the satellite earth station. The transmitted packets are reassembled and transferred to the Host ISP system where they are routed onto the Internet. In the reverse direction, traffic destined for Clients is multiplexed into an MPEG-2 transport stream as user data and broadcast back over the satellite.

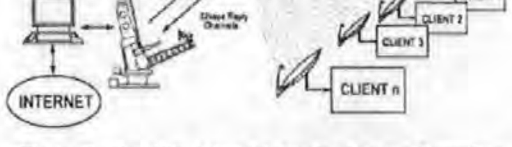


Figure 2 - Satellite Internet Service System Overview

A custom network device driver, on both Client and Host terminals, provides the interface between TCP/IP protocol software and the satellite hardware. The device driver also provides Ethernet emulation such that, to the operating system, the satellite hardware appears to be a standard Ethernet Network Interface Controller (NIC). This ensures compatibility with existing network applications. Each set of Client hardware contains a unique ID number so that users may be individually addressed. System control commands are sent transparently over the satellite channels under the control of a management application. A user database and individual billing logs are maintained at the Host ISP System.

III. CLIENT RECEIVE EQUIPMENT

Broadcast data is demultiplexed from the MPEG-2 transport stream by a domestic satellite TV receiver and applied to a modified SatLink[2,3] receive card, which is shown in Figure 3. The broadcast data channel is continuous and extra frame synchronising information is added to allow the receive cards to obtain synchronisation.

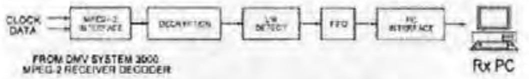
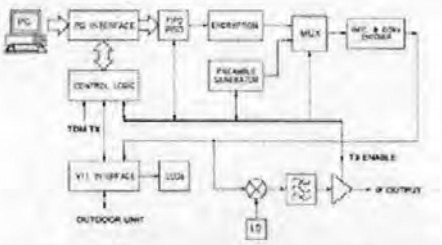


Figure 3 - Data Broadcast Receive Card

After detecting a synchronising sequence, incoming data is assembled into bytes and stored in the PC's memory using Direct Memory Access (DMA). The device driver reassembles incoming Ethernet frames and examines the destination address field. Packets with the correct destination address are passed on to the TCP/IP protocol driver while others are rejected.

IV. CLIENT TRANSMIT EQUIPMENT

Client transmissions on the data reply channels are achieved using the data reply link transmit card and outdoor unit[1] shown in Figure 4 and Figure 5 respectively. The transmit card has a flexible architecture based around a Field Programmable Gate Array (FPGA) which allows various signal processing options to be enabled under software control. An Ethernet frame for transmission is written to a First In First Out (FIFO) buffer where it is stored until burst transmission is initiated. Once triggered the preamble, comprising of phase reversals and a unique word, is clocked serially from Read Only Memory (ROM) for direct transmission. When transmission of the preamble is almost complete, data is clocked out of the FIFO, FEC encoded and concatenated with the preamble. Once transmission is complete, at the end of the burst, the transmitter is switched off and prepared for the next burst transmission.



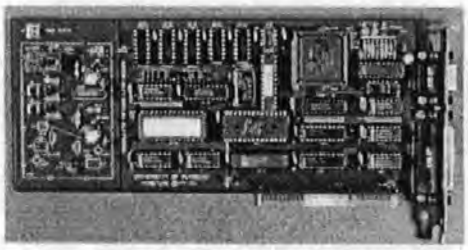


Figure 4 - Data Reply Link Transmit Card

The transmit card provides two outputs, a 70MHz I.F. for connection to a conventional VSAT and a digital interface for connection to the rest of the system.

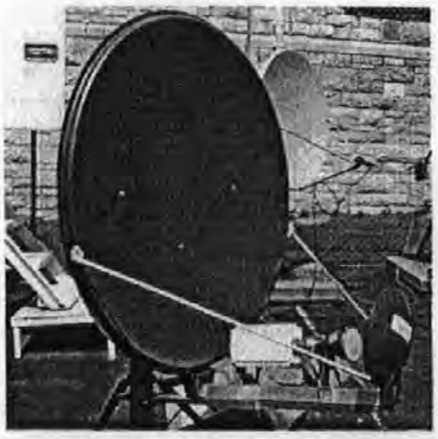
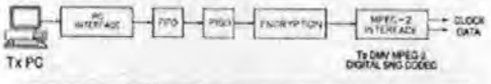


Figure 5 - 14GHz Outdoor Unit

The outdoor unit is mounted on a domestic 90cm satellite antenna and contains an L-Band to 14 GHz up converter and a 200mw transmitting power amplifier. The transmit frequency is set under software control and may be changed prior to each transmission within the 14 GHz to 14.5 GHz band.

V. HOST SYSTEM TRANSMIT EQUIPMENT

For transmission over the data broadcast channel the Host PC contains the SatLink[2,3] data broadcast transmit card shown in Figure 6. Transmitted frames are written to a FIFO buffer along with synchronising information. They are then encrypted and made available as synchronous clock and data for multiplexing into the MPEG-2 transport stream.



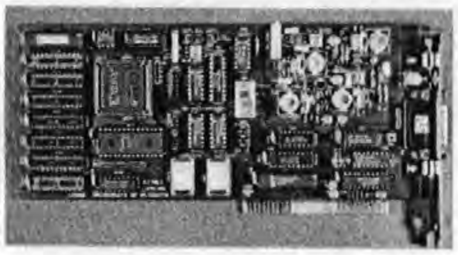


Figure 6 - Data Broadcast Transmit Card

VI. HOST SYSTEM RECEIVE EQUIPMENT

Incoming burst transmissions, at 11GHz, are received on a 3.5m Earth station antenna. After down-conversion, a 64kHz I.F. is applied to the Burst Demodulator Digital Signal Processing (DSP) board where it is sampled at 256kHz by an Analogue to Digital Converter (ADC). The DSP hardware consists of a single extended 6U multi-layer printed circuit board containing two 80MHz TMS320C50 fixed point digital signal processors and is shown in Figure 7.

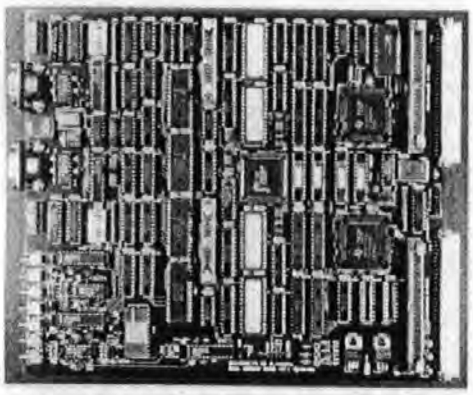


Figure 7 - DSP Burst Demodulator Hardware

The remaining processing is performed by DSP software algorithms. Recovered Ethernet frames are transferred to the Host PC where the outputs from several burst demodulators are multiplexed by the device driver. The frequency acquisition algorithm is computationally intensive and utilises both processors. Once frequency acquisition has been successfully achieved the processors are released, to perform modem and decoding algorithms. A single processor runs algorithms for frequency correction (demodulation), symbol timing recovery and unique word synchronisation, while the second processor provides phase tracking, forward error correction and a buffered interface to the Host PC.

VII. CARRIER FREQUENCY ACQUISITION

Due to oscillator tolerances and propagation over the satellite link, the received signal may deviate from the nominal 64kHz I.F. by several kHz. Successive received bursts arriving from different users may differ substantially in carrier centre frequency. Frequency Acquisition is the process of detecting and estimating each carrier frequency, whereupon frequency correction (demodulation) can be applied. The term 'Frequency Correction' is used because the local oscillator is tuned to the incoming signal. The Fast Fourier Transform (FFT) algorithm is often used for carrier frequency acquisition. At low Signal to Noise Ratios (SNRs) there are two factors that limit the FFT's performance;

1. When carrier power is shared equally between two frequency bins, the signal can be hard to detect above background noise.
2. The resolution of coarse carrier frequency estimates are limited to half the frequency bin spacing.

At low SNR, the Offset Fast Fourier Transform (OFFT) offers superior signal detection and frequency acquisition performance compared to that of the FFT[6]. Equations for the DFT and the modified 'Offset DFT' are given below;

DFT:

$$F(k)=\sum_{n=0}^{N-1} x_n W_N^{nk} \qquad k=0,1,..,N-1 \qquad (1)$$

$$W_N = e^{-j2\pi/N}$$

Offset DFT:

$$F(k+c)=\sum_{n=0}^{N-1} x_n W_N^{n(k+c)} \qquad k=0,1,..,N-1 \qquad (2)$$

In practice, a standard algorithm can be used for both the FFT and OFFT as only the coefficients change.

Comparing power spectra from the FFT and OFFT demonstrates how signal detection performance is improved. The FFT spectrum of Figure 8 shows the case where carrier power is shared equally between adjacent frequency bins.

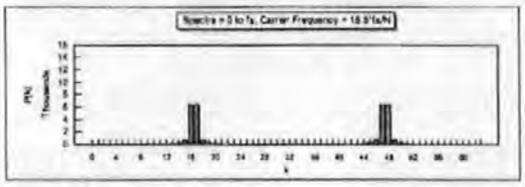


Figure 8 - 64-Point FFT Power Spectrum

With the OFFT (c=0.25) a frequency offset is introduced and this situation can not occur. When carrier power is shared equally between adjacent bins in one half of the output it appears in a single bin in the other as in Figure 9.

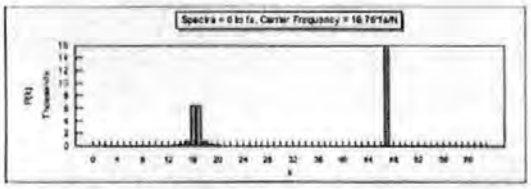


Figure 9 - 64-Point OFFT Power Spectrum

For the FFT power spectrum, a coarse carrier frequency estimate with a resolution of $0.5f_s/N$ can be obtained using the following algorithm;

Find the frequency bin (k_{max}) with maximum power in the range k=0 to k=(N/2)-1.

If $k_{max} < N/2$, $f_{carrier} = k_{max} \times f_s/N$ Hz

Otherwise, $f_{carrier} = (N-k_{max}) \times f_s/N$ Hz (3)

For little overhead, a resolution of $0.25f_s/N$ can be achieved using the OFFT and the following algorithm;

Find the frequency bin (k_{max}) with maximum power in the range k=0 to k=N-1.

If $k_{max} < N/2$, $f_{carrier} = (k_{max}+0.25) \times f_s/N$ Hz

Otherwise, $f_{carrier} = (N-k_{max}-0.25) \times f_s/N$ Hz (4)

VIII. SYMBOL TIMING RECOVERY

A novel DSP based clock recovery scheme demonstrates superior performance compared to that of a PLL; particularly in terms of clock acquisition[4,5]. The scheme exploits the feature of a long impulse response in an Infinite Impulse Response (IIR) filter, when the poles are close to the unit circle. In the context of clock recovery this feature is desirable since it can be exploited to provide low levels of clock jitter. The clock recovery system and associated waveforms are shown in Figure 10. Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible stimulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at its tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E; the recovered clock output.

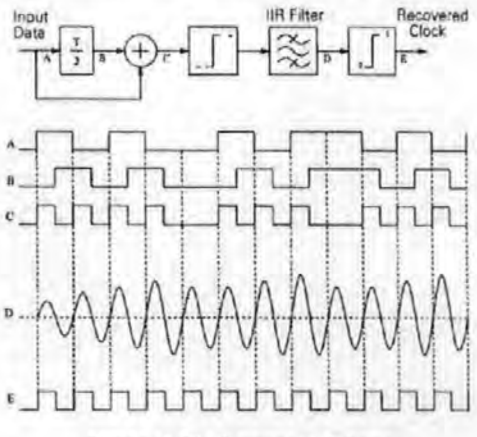


Figure 10 Clock recovery Scheme

An IIR filter is normally formed as a second order section containing both poles and zeros; this provides a centre frequency and stop-band nulls. In this application, however, the stop-band null is of no consequence and the design can be simplified to have feedback paths only. The DSP-based clock recovery system has been simulated on Alta SPW (Signal Processing Worksystem) operating on CAD workstations. Examples of simulation results are shown in Figure 11 and Figure 12.

4

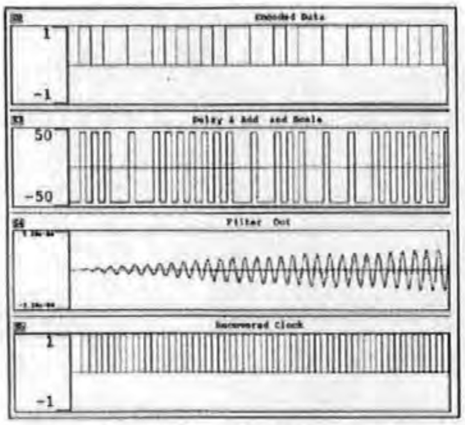


Figure 11 Clock Recovery Acquisition

In Figure 11, a T/2 pre-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing. After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronisation is achieved immediately as the output is in phase with the applied data.

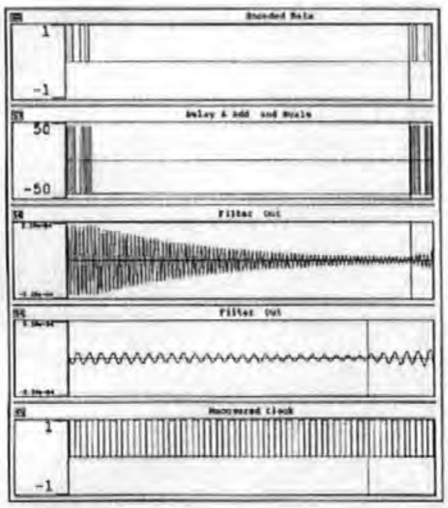


Figure 12 - Clock Recovery Impulse Response

Figure 12 shows the flywheel performance over a 40-bit data fade. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock. Zooming in to the end of the fade shows the recovered clock remaining in phase when data is reapplied; this represents a flywheel performance of 80 clock periods and serves to reduce clock jitter.

IX. BURST DEMODULATOR OPERATION

A diagram of the Burst Demodulator software is shown in Figure 13.

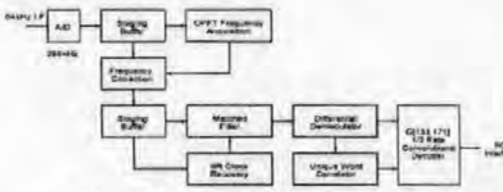


Figure 13 - Burst Demodulator with FEC

Transmitted messages are preceded with a 32kHz phase reversing preamble to aid frequency acquisition and symbol timing recovery. After performing a 256-point OFFT on the incoming samples the receiver examines the power spectrum. The preamble appears as two spectral peaks separated by 32kHz and centred around the carrier frequency as shown in Figure 14.

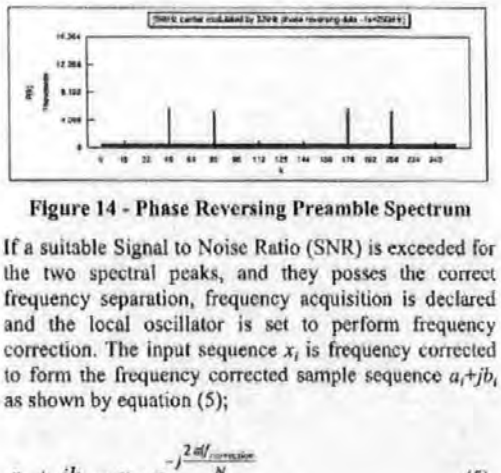


Figure 14 - Phase Reversing Preamble Spectrum

If a suitable Signal to Noise Ratio (SNR) is exceeded for the two spectral peaks, and they posses the correct frequency separation, frequency acquisition is declared and the local oscillator is set to perform frequency correction. The input sequence x_i is frequency corrected to form the frequency corrected sample sequence a_i+jb_i as shown by equation (5);

$$a_i + jb_i = x_i \cdot e^{-j\frac{2\pi f_{correction}}{N}} \quad (5)$$

The maximum residual frequency error after frequency correction is equal to $f_s/4N$ Hz. Since f_s = 256 kHz and N = 256, the worst case residual frequency error is 250 Hz. At a symbol rate of 32,000 symbols per second the maximum phase error between symbols is given by equation (6);

$$\text{phase error}_{max} = \tfrac{250Hz}{32000Hz} \cdot 360° = 2.8125° \quad (6)$$

While a phase tracker can cope with this phase change from symbol to symbol, a differential demodulator is used in order to accommodate the relatively large levels of phase noise associated with the low cost frequency sources in the user equipment.

In keeping with rapid acquisition and synchronisation targets, the phase reversal symbols that caused the receiver to 'acquire' are also used to begin symbol clock recovery. After frequency correction the signal is applied to the clock recovery algorithm and a stable symbol clock is produced within the first 32 symbols. Prior to Forward Error Correction the decoder is initialised and synchronised by detecting the Unique Word at the end of the preamble. Within 32 symbols of receiving the start of the preamble, frequency acquisition, frequency correction and symbol clock recovery are achieved. Once the end of the preamble has been received forward error correction begins. Recovered frames are transferred to the Host PC for processing TCP packets.

X. DATA REPLY LINK SATELLITE TRIALS

A series of satellite trials have been successfully completed using EUTELSAT II F4 widebeam at 7° East and EUTELSAT II F3 at 16° East. The uplink used vertical polarisation and the downlink horizontal polarisation. Satellite reception and monitoring was provided by the European Space Agency's (ESA's) TDS4B Satellite Earth Station situated at the University of Plymouth. Known data packets were transmitted by the return link terminal and received on TDS4B where the signal was down-converted to 70MHz. A series of back-to-back tests were also carried out at 70MHz I.F. The 70MHz I.F. was applied to the return link receiver where the signal was demodulated, decoded and output to a PC where the Bit Error Rate (BER) performance was measured. The BER performance is plotted in Figure 15 and compared to theoretical results. It can be seen that for the back-to-back tests there is approximately 0.7dB degradation from the theoretical, while the satellite tests introduced a further 0.3dB degradation, due mostly to phase noise. The packet acquisition performance is plotted in Figure 16. For an acquisition probability of 99% or better, an E_b/N_o ratio of 6.8dB is required for back-to-back tests while approximately 8dB is required to guarantee successful acquisition performance over the satellite.

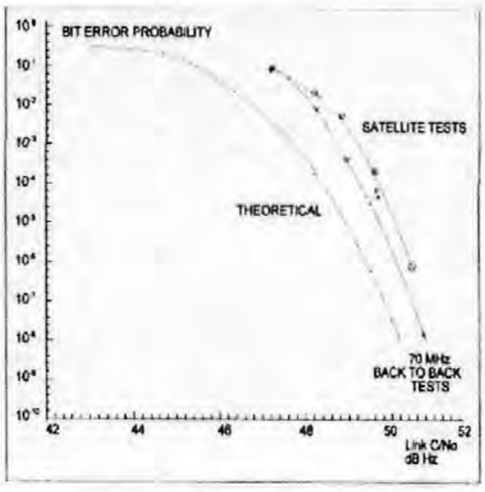


Figure 15 - Bit Error Rate Performance

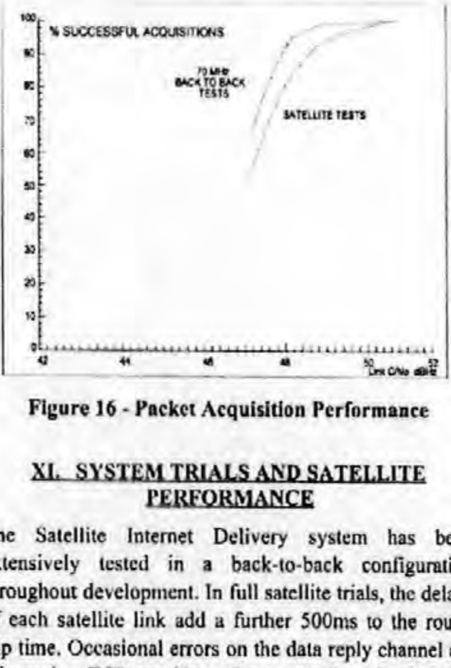


Figure 16 - Packet Acquisition Performance

XI. SYSTEM TRIALS AND SATELLITE PERFORMANCE

The Satellite Internet Delivery system has been extensively tested in a back-to-back configuration throughout development. In full satellite trials, the delays of each satellite link add a further 500ms to the round trip time. Occasional errors on the data reply channel are tolerated as TCP provides a flow control mechanism that automatically recovers from transmission errors. When performance in a back-to-back configuration was compared to the satellite performance, two main effects were observed;

1. When a Web Page is requested, there is an additional one to two second delay before data is received. The TCP connection establishment phase involves a three way handshake between the TCP modules of the client and server. Due to the increased round trip delay, the time taken to establish a connection also increases.
2. In the back-to-back mode, a near optimum transfer rate is consistently achieved. Over the satellite, the data transfer rate is sub-optimum due to the increased round trip time. Performance may be dramatically improved by increasing the TCP acknowledgement window size indicated by the client TCP module. The effect of this is to allow the server to make better use of available bandwidth by minimising idle time waiting for acknowledgements to be received.

Figure 17 shows the maximum throughput per user as a function of the TCP acknowledgement window size. The round trip time for the satellite tests in this particular case was 610ms.

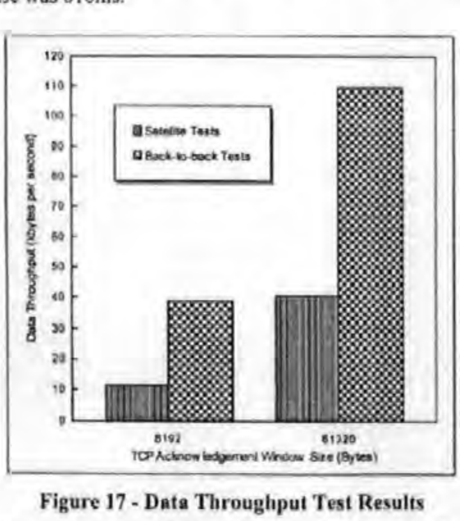


Figure 17 - Data Throughput Test Results

These results were obtained by downloading a 6Mbyte test file over the satellite using the File Transfer Protocol (FTP), first with the default acknowledgement window size of 8192 bytes and then with a near maximum acknowledgement window of 61320 bytes. Using an incoming data rate of 1Mbps and an outbound rate of 16kbps, the effect of increasing the TCP acknowledgement window size can be clearly seen. The measured transfer rate increases from 11.7kbytes per second to over 40kbytes per second. It is interesting to note that with the satellite delay a single user is not able to monopolise the total available bandwidth and is limited to approximately 300kbps. In back-to-back tests, excluding the satellite, a single user can achieve a transfer rate in excess of 100kbytes per second, thus utilising the whole of the available bandwidth. This confirms that the system delays serve to reduce overall throughput for a single user.

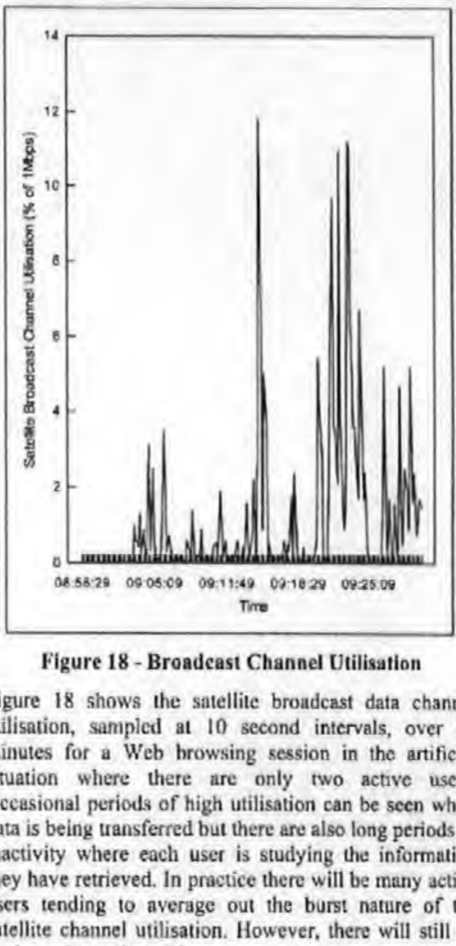


Figure 18 - Broadcast Channel Utilisation

Figure 18 shows the satellite broadcast data channel utilisation, sampled at 10 second intervals, over 30 minutes for a Web browsing session in the artificial situation where there are only two active users. Occasional periods of high utilisation can be seen while data is being transferred but there are also long periods of inactivity where each user is studying the information they have retrieved. In practice there will be many active users tending to average out the burst nature of the satellite channel utilisation. However, there will still be peaks and troughs evident.

At the time of writing, user equipment has been installed at a number of sites within the South West of England in order to obtain user feedback prior to the launch of a commercial system. A series of commercial satellite trials are planned shortly so that typical users can evaluate the system and provide additional feedback which will aid future development.

XII. CONCLUSIONS

The satellite data reply link described operates at low power and uses the same transponder as the broadcasting service; the data reply carriers occupy a small part of the satellite bandwidth assigned to the broadcast service which is typically unused bandwidth. The system operates at low signal to noise ratios with a user return data rate of 16kbps employing an efficient 1/2 rate FEC algorithm. A short synchronisation preamble is applied enabling rapid signal acquisition, symbol timing and decoder synchronisation. The system is tolerant of frequency errors and employs either differential demodulation or phase tracking to remove residual phase errors depending upon the level of phase noise experienced. The rapid acquisition techniques described exhibit high probabilities of successful message decoding.

The data reply link has been combined with an MPEG-2 data broadcast channel to provide a novel satellite Internet delivery system. The system exploits the fact that domestic Internet traffic is asymmetric in nature. The system has particular appeal to regions without good access to the PSTN, and is useful for portable applications, as a high speed Internet link may be established anywhere within the satellite's footprint. Successful technical and user satellite trials have been conducted and commercial trials are planned for the near future.

XIII. REFERENCES

[1] Slader JT, Smithson PM, Tomlinson M, "A Data Reply Link System for Satellite TV Applications", *IEEE Globecom'97 Proceedings ISBN 0-7803-4198-8, V2, Page 1142. 1997.*

[2] Smithson PM, Slader JT, Smith DF, Tomlinson M, "The Development of an Operational Satellite Internet Service Provision", *IEEE Globecom'97 Proceedings ISBN 0-7803-4198-8, V2, Page 1147. 1997.*

[3] Smithson PM, Tomlinson M, "The Development of an Operational Computer Based Satellite Data Broadcasting System", *IEE International Conference on Digital Satellite Communications ICDSC-10 Proceedings ISBN 0-85296-6393, V2, Page 405. 1995*

[4] Smithson PM, Tomlinson M and Donnelly T, "DSP-Based Clock Recovery for a Digital Magnetic Recording Channel", *IEEE Globecom'94 Proceedings ISBN 0-7803-1820-X, V3, Page 1467. 1994.*

[5] Smithson PM, Tomlinson M and Donnelly T, "DSP-Based Clock Recovery Implemented in a Field Programmable Gate Array", *IEE Colloquium Proceedings New Synchronisation Techniques for Radio Systems ISSN 0963-3308, Ref No: 1995/220. 1995.*

[6] Tomlinson M, "The Offset Fast Fourier Transform," *MOD SRDE Report No: 76025. 1979.*

XIV. ACKNOWLEDGEMENTS

The authors would like to thank BNSC (The British National Space Council), Communicado Data Ltd., ESA (The European Space Agency) and Armstrong Electronics Ltd. for their support. Thanks are also extended to the University of Plymouth satellite uplink staff and the RATIO project staff for their assistance during satellite tests and user trials

K. References

[1] Burranchini E., 'The software radio concept'.
IEEE Communications Journal, vol. 38, no. 9, page 138-143, September 2000.

[2] Srikanteswara S., Reed J.H., Athanas P., Boyle R., 'A soft radio architecture for reconfigurable platforms'.
IEEE Communications Journal, vol. 38, no. 2, page 140-147, February 2000.

[3] Shepherd R., 'Engineering the embedded software radio'.
IEEE Communications Journal, vol. 37, no. 11, page 70-74, November 1999.

[4] Efsatgiou D., Fridman J., Zvonar Z., 'Recent developments in enabling technologies for software defined radio'.
IEEE Communications Journal, vol. 37, no. 8, page 104-106, August 1999.

[5] Cummings M., Heath S., 'Mode switching and software download for software defined radio: The SDR forum approach'.
IEEE Communications Journal, vol. 37, no. 8, page 104-106, August 1999.

[6] Tuttlebee W.H.W., 'Software radio technology: a European perspective'.
IEEE Communications Journal, vol. 37, no. 2, page 118-123, February 1999.

[7] Turletti T., Tennenhouse D., 'Complexity of a software GSM base station'.
IEEE Communications Journal, vol. 37, no. 2, page 113-117, February 1999.

[8] Cummings M., Haruyama S., 'FPGA in software radio'.
IEEE Communications Journal, vol. 37, no. 2, page 108-112, February 1999.

[9] Mitola J., 'Technical challenges in the globalisation of software radio'.
IEEE Communications Journal, vol. 37, no. 2, page 84-89, February 1999.

[10] Gatherer A., Stetzler T., McMahan M., Auslander E., 'DSP-Based architectures for mobile communications: Past, present and future'.
IEEE Communications Journal, vol. 38, no. 1, page 84-90, January 2000.

[11] Mirabbasi S., Martin K., 'Classical and modern receiver architectures'.
IEEE Communications Journal, vol. 38, no. 11, page 132-139, November 2000.

[12] Khoury J., Tao H., 'Data converters for communication systems'.
IEEE Communications Journal, vol. 36, no. 10, page 113-117, October 1998.

[13] Walden R.H., 'Performance trends for analog-todigital converters'.
IEEE Communications Journal, vol. 37, no. 2, page 96-101, February 1999.

[14] Tsurumi H., Suzuki Y., 'Broadband RF stage architecture for software-defined radio in handheld terminal applications'.
IEEE Communications Journal, vol. 37, no. 2, page 90-95, February 1999.

[15] Chester D.B., 'Digital IF filter technology for 3G systems: an introduction'.
IEEE Communications Journal, vol. 37, no. 2, page 102-107, February 1999.

[16] Svvevenhaus J., Verstraeten B., Taraborrelli S., 'Trends in silicon radio large scale integration: Zero IF receiver! Zero I & Q transmitter! Zero discrete passives!'.
IEEE Communications Journal, vol. 38, no. 1, page 142-147, January 2000.

[17] Heutmaker M.S., Le D.K., 'An architecture for self-test of a wireless communication system using sampled IQ modulation and boundary scan'.
IEEE Communications Journal, vol. 37, no. 6, page 98-102, June 1999.

[18] Dick C., Harris F.J., 'Configurable logic for digital communications: Some signal processing perspectives'.
IEEE Communications Journal, vol. 37, no. 8, page 107-111, August 1999.

[19] Ramsdale P.A., 'The development of personal communication'.
IEE Electronics and Communication Engineering Journal, vol. 8, no. 3, page 143-151, June 1996.

[20] Kenington P.B., 'Emerging technologies for software radio'.
IEE Electronics and Communication Engineering Journal, vol. 11, no. 2, page 69-83, April 1999.

[21] Forrest J.R., 'Telemedia: A survival guide to the fifth dimension'.
IEE Electronics and Communication Engineering Journal, vol. 8, no. 1, page 13-23, February 1996.

[22] Woolfe C.D., 'Yacht video system for the Whitbread Round the World race'.
IEE Electronics and Communication Engineering Journal, vol. 8, no. 6, page 281-288, December 1996.

[23] Dixit S., 'Data rides high on high-speed remote access'.
IEEE Communications Journal, vol. 37, no. 1, page 130-141, January 1999.

[24] Kwok T.C., 'Residential broadband architecture over ADSL and G.Lite (G.992.2): PPP over ATM'.
IEEE Communications Journal, vol. 37, no. 5, page 84-89, May 1999.

[25] Cioffi J.M., Oksman V., Werner J., Pollet T., Spruyt P.M.P., Chow J., Jacobson K.S., 'Very-high-speed digital subscriber lines'.
IEEE Communications Journal, vol. 37, no. 4, page 72-79, April 1999.

[26] Cook J.W., Kirkby R.H., Booth M.G., Foster K.T., Clarke D.E.A., Young G., 'The noise and crosstalk environment for ADSL and VDSL systems'.
IEEE Communications Journal, vol. 37, no. 5, page 73-78, May 1999.

[27] Narumiya K., 'A consideration of ADSL service under NTT's network'.
IEEE Communications Journal, vol. 37, no. 5, page 98-101, May 1999.

[28] Oksman V., Werner J., 'Single-carrier modulation technology for very high-speed digital subscriber line'.
IEEE Communications Journal, vol. 38, no. 5, page 82-89, May 2000.

[29] Azcorra A., Larrabeiti D., Hernandez-Valencia E.J., Berrocal J., 'IP/ATM integrated services over broadband access copper technologies'.
IEEE Communications Journal, vol. 37, no. 5, page 90-97, May 1999.

[30] Saltzburg B.R., 'Comparison of single-carrier and multitone digital modulation for ADSL applications'.
IEEE Communications Journal, vol. 36, no. 11, page 114-121, November 1998.

[31] Oksman V., Werner J., 'Single-carrier modulation technology for very high-speed digital subscriber line'.
IEEE Communications Journal, vol. 38, no. 5, page 82-89, May 2000.

[32] Magliacane J.A., 1992, "Spotlight on: UoSAT-OSCAR-22"
AMSAT Journal, Volume 15 No. 3, May/June 1992.
http://www.amsat.org/amsat/sats/n7hpr/uo22_kd2.html

[33] "PoSat-1 Technical Features"
http://www.laer.ineti.pt/posat/techfeat/techf0.html

[34] Miller J., 1995 "The shape of bits to come".
ftp.amsat.org/amsat/articles/g3ruh/a108.zip

[35] "Digital video broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television (DVB-T)".
European Telecommunication Standards Institute, ETS 300744, 1997

[36] Tomlinson M. 1979, "The Offset Fast Fourier Transform".
MOD SRDE Report No: 76025. 1979.

[37] Stott M.A., 1995, "The effect of frequency errors in OFDM".
BBC Research Department Report No RD 1995/15

[38] Wood D., 'The DVB project: Philosophy and core system'.
IEE Electronics and Communication Engineering Journal, vol. 9, no. 1, page 5-10, February 1997

[39] Oppenheim A. V., Schafer R.W., 'Digital Signal Processing'.
Prentice Hall International Editions 1975, ISBN 0-13-214107-8, chapter 6, section 6.1, page 287-289.

[40] Burrus C.S., Parks T.W., 'DFT/FFT and Convolution Algorithms - Theory and implementation'.
John Wiley and Sons 1985, ISBN 0-471-81932-8, chapter 2, section 2.2.3, page 32-36.

[41] Mitra S. K., 'Digital Signal Processing - A computer-based approach'.
McGraw Hill 1998, ISBN 0-07-0429531-7, chapter 8, section 8.3.1, page 520-523.

[42] Smithson P. M., Tomlinson M., Donnelly T., 1994, "DSP-Based Clock Recovery for a Digital Magnetic Recording Channel",
IEE Globecom '94 Proceedings ISBN 0-7803-1820-X, V3, Page 1467

[43] Smithson P. M., Tomlinson M., 1996, "The Development of an Operational Computer Based Satellite data Broadcasting system"
IEE Conference on Digital Satellite Communications ICDSC-10 Proceedings ISBN 0-85296-6393, V2, Page 405.

[44] Wade G., "Signal Coding and Processing - Second Edition",
Cambridge University Press, ISBN 0-521-42336-8, page 346.

[45] Tomlinson M., 1979, "The Offset Fast Fourier Transform",
MOD SRDE Report No. 76025.

[46] Blair G. M., 1995, "A Review of the discrete Fourier Transform Part 1: Manipulating the Powers of two",
Electronics & Communication Journal, page 169-177, August 1995.

[47] Blair G. M., 1995, "A Review of the discrete Fourier Transform Part 2: Non-Radix Algorithms, Real Transforms and Noise",
Electronics & Communication Journal, page 187-194, October 1995

[48] Haykin S., "Communication Systems",
John Wiley & Sons, ISBN 0-471-30584-7, page 102 - 109.

[49] Wade G., "Signal Coding and Processing - Second Edition",
Cambridge University Press, ISBN 0-521-42336-8, page 341 – 382.

[50] Wade G., "Signal Coding and Processing - Second Edition",
Cambridge University Press, ISBN 0-521-42336-8, page 350.

[51] Guidi A., McIllree P.., 1995, "Development of a spectrally efficient, high speed modem for microwave terrestrial and satellite communications",
IEEE Communication Journal, ISBN 0-8186-7085-1/95, page 211-216, 1995.

[52] "TMS320C5x – User's Guide",
Texas Instruments, 1992, ISBN 254701-9761 revision D, *page 3-42.*

[53] Slader JT, Smithson PM, Tomlinson M, Ambroze A, Wade JG "A Novel Internet Delivery System Using Asymmetric Satellite Channels",
IAF-98-M.5.06. Presented at the 49th International Astonautical Congress, 1988.

[54] Smithson PM, Tomlinson M, "The Development of an Operational Based Satellite Data Broadcasting System",
IEE International Conference on Digital Satellite Communications. ICDSC-10 Proceedings ISBN 0-85296-6393, V2, Page 405. 1995

[55] Ohata M., 'IETF and Internet standards'.
IEEE Communications Journal, vol. 36, no. 9, page 126-129, September 1998.

[56] European telecommunications Standards Agency "Digital Video Broadcasting (DVB); DVB Specification for Data Broadcasting"
ETSI Standard Document Reference EN 301 192 v1.2.1 (1999-06)

[57] Allman M, Glover D, Sanchez L, "Enhancing TCP Over Satellite Channels using Standard Mechanisms"
Internet Request for Comment 2488, Best Current Practice 28, January 1999

[58] Miller P "TCP/IP Explained"
Chapter 3, Section 3.1, page 27.
Digital Press ISBN 1-55558-166-8

[59] Postel J "Satellite Considerations"
Internet Request for Comment 346, ftp://ftp.isi.edu/in-notes/rfc346.txt

[60] Allman M, Paxson V, Stevens, W "TCP Congestion Control"
Internet Request for Comment 2581, ftp://ftp.isi.edu/in-notes/rfc2581.txt

[61] Floyd S, Henderson T, "The NewReno Modification to TCP's Fast Recovery Algorithm"
Internet Request for Comment 2582, ftp://ftp.isi.edu/in-notes/rfc2582.txt

[62] Stevens W, "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast recovery Algorithms"
Internet Request for Comment 2001, ftp://ftp.isi.edu/in-notes/rfc2001.txt

[63] Jacobson V, Branden R, "TCP Extensions for Long-Delay Paths"
Internet Request for Comment 1072, ftp://ftp.isi.edu/in-notes/rfc1072.txt

[64] Floyd S, Mahdavi J, Mathis M, Podolsky M "An Extension to the Selective Acknowledgement (SACK) Option for TCP"
Internet Request for Comment 2883, ftp://ftp.isi.edu/in-notes/rfc2883.txt

[65] Mathis M, Mahdavi J, Floyd S, Romanow A, "TCP Selective Acknowledgement Options"
Internet Request for Comment 2018, ftp://ftp.isi.edu/in-notes/rfc2018.txt

[66] "Recommendation for space data system standards – telemetry channel coding",
Consultative Committee for Space Data Systems, CCDSD 101.0.B-3 Blue Book, May 1992, Section 2, page 2-1.
http://ftp.ccsds.org/documents/pdf/CCSDS-101.0-B-3.pdf

L. Bibliography

OFDM

1 Cioffi J.M., 1991, "A multicarrier primer".
ANSI T1E1.4/91-157, Ft. Lauderdale November 1991

2 Allard M., Lassalle R. 1987, "Principals of modulation and channel coding for digital broadcasting for mobile receivers".
EBU Review Technical, No. 224, August 1987, page 168-190

3 Le Floch B., Allard M., Lassalle R., Castelain D. 1989, "Digital sound broadcasting to mobile receivers".
IEE Trans. Consumer Electronics, vol. 35, no. 3, August 1989.

4 Bingham J.A.C., 1990 "Multicarrier modulation for data transmission: An idea whose time has come".
IEE Communications Journal, vol. 28, no. 5, page 5-14, May 1990.

5 Maddocks M.C.D., 1993, "An introduction to digital modulation and OFDM techniques".
BBC Research Department Report No RD 1993/10

6 Lee M.B.R., 1993, "Predicted coverage of a single frequency network for UHF digital terrestrial TV Broadcasting".
BBC Research Department Report No RD 1993/12

7 Maddocks M.C.D., Pullen I.R. 1993, "Digital Audio Broadcasting: Comparison of coverage at different frequencies and with different bandwidths".
BBC Research Department Report No RD 1993/11

8 van de Beek J.J., Sandel M., Isaksson M., Borjesson P.O., 1995, "Low-complex frame synchronisation in OFDM systems".
Proceedings of International Conference on Universal Personal Communication. ICUPC '95, November 1995.

9 Sandel M., van de Beek J.J., Borjesson P.O., 1995, "Timing and frequency synchronisation in OFDM systems using the cyclic prefix".
Proceedings of the 1995 IEEE International Symposium on Synchronisation, page 16-19, December 1995.

10 Pollet T., Moeneclaey M., 1996, "The effect of carrier frequency offset on the performance of band limited single carrier and OFDM signals".
Proceedings of the 1996 IEEE Global Telecommunications Conference, conference record 1 of 3, page 719-723, November 1996.

11 Malmgren G., 1996, "Impact of carrier frequency offset, Doppler spread and time synchronisation errors in OFDM based single frequency networks".
Proceedings of the 1996 IEEE Global Telecommunications Conference, conference record 1 of 3, page 729-733, November 1996.

12 Cimini L.J., Daneshrad B., Sollenberger N.R., 1996, "Clustered OFDM with transmitter diversity and coding". *Proceedings of the 1996 IEEE Global Telecommunications Conference, conference record 1 of 3, page 703-707, November 1996.*

13 Czylwik A., 1996, "Adaptive OFDM for wideband radio channels". *Proceedings of the 1996 IEEE Global Telecommunications Conference, conference record 1 of 3, page 713-718, November 1996.*

14 Fischer R.F.H., Huber J.B., 1996, "A new loading algorithm for discrete multitone transmission". *Proceedings of the 1996 IEEE Global Telecommunications Conference, conference record 1 of 3, page 724-728, November 1996.*

15 van Nee R.D.J., 1996, "OFDM codes for peak-to-average power reduction". *Proceedings of the 1996 IEEE Global Telecommunications Conference, conference record 1 of 3, page 740-744, November 1996.*

16 Sari H., Karem G., Jeanclaude I., 1995, "Transmission techniques for digital terrestrial TV broadcasting". *IEEE Communications magazine, vol. 33, no. 2, page 100-109, February 1995.*

17 Reimers U., 'DVB-T: the COFDM-based system for terrestrial television'. *IEE Electronics and Communication Engineering Journal, vol. 9, no. 1, page 28-32, February 1997.*

18 Drury G.M., 'DVB channel coding standards for broadcasting compressed video services'. *IEE Electronics and Communication Engineering Journal, vol. 9, no. 1, page 11-20, February 1997.*

19 Robertson P., 1997, "Close-to-optimal one-shot frequency synchronisation for OFDM using pilot carriers". *Proceedings of the 1997 IEEE Global Telecommunications Conference, vol. 4, page 97-102, November 1997.*

20 Young G., Foster K.T., Cook J.W., "Broadband multimedia delivery over copper". *IEE Electronics and Communication Engineering Journal, February 1996, page 25-36.*

21 Kyees P.J., McConnel R.C., Sistanizadeh K., 1995, "ADSL: A new twisted-pair access to the information highway". *IEEE Communications Magazine, April 1995, page 52-59.*

22 Young G., 1994, "Asymmetric digital subscriber line (ADSL) technology: Introduction and overview". *Electronics and Communication Engineering Journal, February 1996, page 25-36.*

Burst Demodulator Applications

23 Miller P "TCP/IP Explained" *Digital Press ISBN 1-55558-166-8*

24 Socolofsky T, Kale C, "A TCP/IP Tutorial" *Internet Request for Comment 1180*

25 Allman M, Dawkins S, Glover D, Griner J, Tran D, Henderson T, Heidemann Touch J, "Ongoing TCP Research Related to Satellites"

Internet Request for Comment 2760

26 Poduri K, Nichols K, "Simulation Studies of Increased Initial TCP Window Size"
Internet Request for Comment 2415

27 Allman M, Floyd S, Partridge C, "Increasing TCP's Initial Window"
Internet Request for Comment 2414

28 Jocobson V, Braden R, Borman D, "TCP Extensions for High Performance"
Internet Request for Comment 1323

29 Jacobson V, Braden R, Zhang L, "TCP Extensions for High-Speed Paths"
Internet Request for Comment 1185

30 McKenzie A, "A Problem with the TCP Big Window Option"
Internet Request for Comment 1110

31 Fox R, "TCP Big Window and Nak Options"
Internet Request for Comment 1106