INVESTIGATION INTO SYNCHRONIZATION FOR PARTIAL RESPONSE SIGNALS AND THE DEVELOPMENT OF A CLOCK RECOVERY SCHEME FOR 49QPRS SIGNALS

G. D. JORDAAN

INVESTIGATION INTO SYNCHRONIZATION FOR PARTIAL RESPONSE SIGNALS AND THE DEVELOPMENT OF A CLOCK RECOVERY SCHEME FOR 49QPRS SIGNALS

by

Gert Daniel Jordaan

Thesis submitted in fulfilment of the requirements for the

Doctor Technologiae: Engineering: Electrical

in the

Faculty of Engineering

at the

Technikon Free State

October 1997

Promoter: Dr R. M. Braun Ph.D. (Eng)

Co-promoter: Prof. G. Prinsloo Ph.D. (Eng)

DECLARATION

I, GERT DANIEL JORDAAN, hereby declare that the research project which has been submitted to Technikon Free State by me for the obtaining of the degree DOCTOR TECHNOLOGIAE: ENGINEERING: ELECTRICAL, is my independent work and has not been submitted by me or any other person previously in view of attaining any qualification.

G D JORDAAN
Student

1997/10/10

Date

ACKNOWLEDGEMENTS

There are a number of people who gave me invaluable assistance throughout the duration of this research project. In this regard I would like to thank:

Dr Robin Braun for his supervision and guidance since the start of the project.

Prof. Gerhard Prinsloo for excellent advice and assistance.

Ernst du Plooy for his commitment and help during the experimental phase of the project.

For all my colleagues in the communications research group for their enthusiasm and motivation when we were battling with difficult concepts.

Technikon Free State and the FRD for direct and indirect financial and other support which enabled me to attempt the study.

Mr Frank Coetzer and Pat Jonker for philological assistance in preparing the text.

Lastly, in particular I would like to thank my wife, Christa, and children, Johan, Lourens, Madelie and Tania, whose main contribution was to have to forego much of my attention and time for such a long period.

SUMMARY

Data communication is used increasingly in modern society. It is against this background that research is conducted worldwide toward the improvement of existing, as well as the development of new, improved communication techniques.

Correlative encoding of data before transmission is a very frequency-effective communication technique. The extent to which any communication technique is used, however, is dependent on a wide variety of factors. This study regarding the synchronisation of 49QPRS signals was undertaken with this in mind.

Since digital signal processing (DSP) is used increasingly in modern communication systems, both a data transmitter and receiver were implemented by making use of this technique. Not only would this result in a system with all the desirable characteristics inherent to DSP, but, by making limited changes to the supporting software, the evaluation of a wide variety of alternatives became feasible.

During the study a system making use of a pilot tone at one third the frequency of the carrier frequency was developed. The receiver recovers this signal by means of DSP techniques and its frequency is tripled. The phase of this recovered signal is cross-correlated every 650 μs in time with a locally generated signal of the correct frequency - and the phase of the locally generated signal is adjusted accordingly. It was found that the accuracy and stability of the locally generated signal were such that sufficient synchronisation was obtained in this manner. The quality of synchronisation is a function of the level of the pilot tone and if this tone should decrease to below a certain value, unacceptably large phase adjustments have to be made. This results in a serious degradation of the spectral purity of the recovered signal. However, the system as described exhibits extremely good noise immunity.

During the development of the clock frequency recovery system, a baseband filter with a unique frequency response was defined. Making use of this, in conjunction with a limited amount of pre-processing, and an absolute value rectifier, recovery of the clock frequency becomes possible. In order to limit the amount of processing by the receiver, the baseband filter was implemented in its entirety in the transmitter. The recovered signal showed a moderate amount of amplitude variation, but an extremely stable synchronising signal could be derived from this.

During the study both levels of synchronisation required by a hypothetical 49QPRS data communication system were therefore investigated fully and solutions found.

UITTREKSEL

Datakommunikasie vind steeds toenemende gebruik in die moderne samelewing. Dit is teen hierdie agtergrond dat wêreldwyd navorsing in die verbetering van bestaande, en die ontwikkeling van nuwe kommunikasietegnieke gedoen word.

Korrelatiewe enkodering van data voor transmissie is 'n besonders frekwensie-effektiewe kommunikasietegniek. Die mate waartoe enige kommunikasietegniek aangewend word is egter afhanklik van 'n groot verskeidenheid faktore. In die lig hiervan is hierdie studie in die sinkronisering van 49QPRS korrelatief ge-enkodeerde seine onderneem.

Aangesien digitale seinprosessering (DSP) toenemend in moderne komunikasiestelsels aangewend word, is beide 'n datasender en -ontvanger m.b.v. hierdie tegniek geïmplementeer. Nie net sou dit 'n stelsel met al die goeie eienskappe inherent aan DSP verseker nie, maar deur slegs beperkte wysigings in die ondersteunende sagteware te maak was dit moontlik om 'n wye verskeidenheid alternatiewe te evalueer.

Gedurende die studie is 'n stelsel ontwikkel wat voorsiening maak vir die herwinning van die draagfrekwensie met behulp van transmissie van 'n loodstoon teen een derde van die draagfrekwensie. Deur gebruikmaking van DSP tegnieke onttrek die ontvanger hierdie sein en die frekwensie daarvan word verdriedubbel. Die fase van die herwinde sein word elke 650 µs gekruiskorreleer met 'n plaaslik opgewekte sein van die regte frekwensie - en die fase van laasgenoemde word ooreenkomstig aangepas. Daar is bevind dat die akkuraatheid en stabiliteit van die plaaslik opgewekte sein sodanig is dat voldoende sinkronisasie op hierdie wyse verkry kan word. Die kwaliteit van sinkrinisasie wat verkry word is 'n funksie van die vlak van die loodstoon en indien hierdie vlak tot benede 'n vaste, minimum waarde daal, word buitensporige fase-aanpassings gemaak - en word die spektrale suiwerheid van die herwinde sein nadelig beïnvloed. Die tegniek soos ontwikkel toon baie goeie ruisimmuniteit.

Tydens die ontwikkeling van 'n klokpulsregenerator is 'n basisbandfilter met 'n unieke oordragkarakteristiek gedefinieer. Gebruik hiervan, saam met 'n beperkte mate van voorprosessering, en 'n absolute-waarde gelykrigter, het die herwinning van die klokfrekwensie moontlik gemaak. Ten einde prosesseringstyd van die ontvanger te beperk is die basisbandfilter ten volle in die sender geïmplementeer. Die geregenereerde sein het wel matige variasie in amplitude getoon, maar 'n besonders stabiele sinkroniseersein is hiervan afgelei.

Gedurende die studie is beide vlakke van sinkronisering soos benodig vir 'n 49QPRS datakommunikasiestelsel dus volledig ondersoek en oplossings daarvoor gevind.

CONTENTS

LIST OF FIGURES

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

LIST OF TABLES

LIST OF ACRONYMS

3PRS	Ternary partial response signalling
7PRS	7-Level partial response signalling
9QPRS	9-Quadrature partial response signalling
49QPRS	49-Quadrature partial response signalling
ADC	Analogue-to-digital converter
AIC	Analogue interface circuit
ASK	Amplitude shift keying
AVR	Absolute value rectifier
AVR-LP	Absolute value rectifier with limited pre-processing
BER	Bit error rate
BPF	Bandpass filter
CCITT	The international telegraph and telephone consultative committee
dB	Decibel
dc	Direct current
DFE	Decision feedback equalisers
DSBSC	Double sideband, suppressed carrier
DSP	Digital signal processing
FFT	Fast Fourier transform
FIR	Finite impulse response
FLR	Fourth-law rectifier
FSK	Frequency shift keying
IF	Intermediate frequency
ISI	Intersymbol interference
kHz	Kilohertz
LP	Limited pre-processing
LPF	Lowpass filter

MIPS	Million instructions per second
MMSE	Minimum mean-square error
OMR	Operating mode register
PAM	Pulse amplitude modulation
PC	IBM compatible personal computer
PLL	Phase-locked loop
PRS	Partial response signalling
PS	Pulse shaping
PSF	Pulse shaping filter
PSK	Phase shift keying
QAM	Quadrature amplitude modulation
QPRS	Quadrature partial response signalling
RC	Raised cosine
RCF	Raised cosine filter
Rx	Receiver
SCI	Serial communication interface
SIG-56	DSP development board developed by Peralex Electronic Development CC
SLR	Square-law rectifier
SNR	Signal-to-noise ratio
SSI	Synchronous serial interface
Tx	Transmitter
VCO	Voltage-controlled oscillator

CHAPTER 1

INTRODUCTION

The increasing need of modern-day society for the electronic storage and transfer of information necessitates continuing research in, and development of more sophisticated techniques for handling the data. This entails the development of new methods, but often also re-examining - and if possible improving the performance of - well-known techniques. This phenomenon is directly influenced by the development of new electronic devices.

Partial response signalling (PRS) is such a well-known data transmission technique, with both positive and negative attributes. However, although it provides for efficient utilisation of the available bandwidth, it is seldom used in practice. This technique introduces correlation between successive data symbols to achieve a convenient spectral shaping.

1.1 OBJECTIVES OF THESIS AND EXECUTION OF PROJECT

The project entailed the investigation of the synchronisation characteristics of Class 1 PRS. During execution of the project, a 49QPRS data transmission system, transferring 9 600 bits/second in a low-frequency application, was developed by making use of DSP techniques. The transmitter and receiver each utilise a DSP56001 processor to perform the required signal processing functions. This includes PRS encoding, filtering and modulation

at the transmitter, and sampling, carrier recovery, clock synchronisation and demodulation at the receiver.

The main objective of the project was to investigate the possible development of an improved synchronisation scheme for such a system. The availability of an improved synchronisation scheme between the data transmitter and the receiver might eventually lead to a re-evaluation of the suitability of PRS as a major data communication technique.

Synchronisation between the transmitter and the receiver should be maintained on two levels;

- The receiver should generate a signal, identical in frequency and phase to the carrier signal.
- The receiver should generate a signal in synchronism with the symbol rate recovered during demodulation of the transmitted signal.

A carrier recovery scheme utilising transmission of a pilot tone at one third of the frequency of the carrier was implemented. This signal is recovered at the receiver, and used to regenerate both an in-phase and quadrature carrier signal. Very promising results were found. The signal tracked the carrier of the transmitter closely. The amount of phase correction required by the receiver was found to be dependent on the amplitude of the transmitted pilot tone. The acquisition time was found to be extremely short. Extensive

testing was also done at different signal-to-noise ratios, and the carrier recovery system proved to have a very good noise immunity.

The presence of intersymbol interference in a data communication system is avoided by limiting the bandwidth of the transmitted and received signal. The transmitter and the receiver filters normally have a combined raised cosine response. A filter with an alternative characteristic was developed for the purpose of the project. This filter requires a wider bandwidth, but contributes substantially to the amplitude of the recovered clock signal. For practical reasons, this characteristic was not split between the transmitter and the receiver. Only one filter was used to realise the response as described.

Clock synchronisation systems often employ some type of non-linearity. During the course of the project, use was made of an absolute-value rectifier with a limited amount of pre-processing. This scheme was found to provide a substantial spectral line at the symbol frequency, which could be used to maintain synchronisation between the transmitter and the receiver. Extensive evaluation proved that the timing system, as proposed, ensures proper synchronisation.

The research procedure of the project is outlined by the flowchart in Figure 1.1.

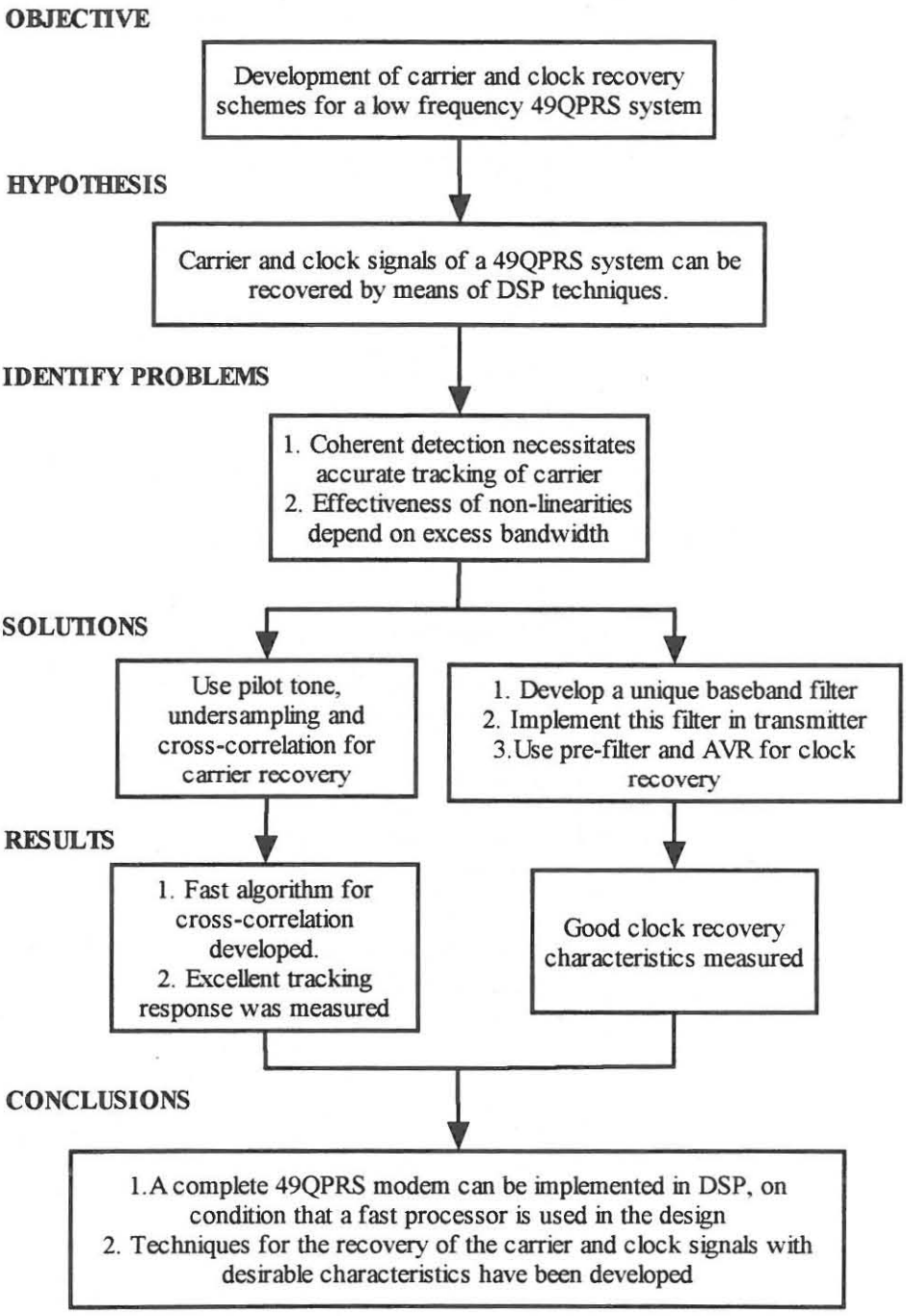


Figure 1.1: Research phases during execution of project.

1.2 ORGANISATION OF THESIS

The flowchart in Figure 1.2 shows the basic layout of the thesis.

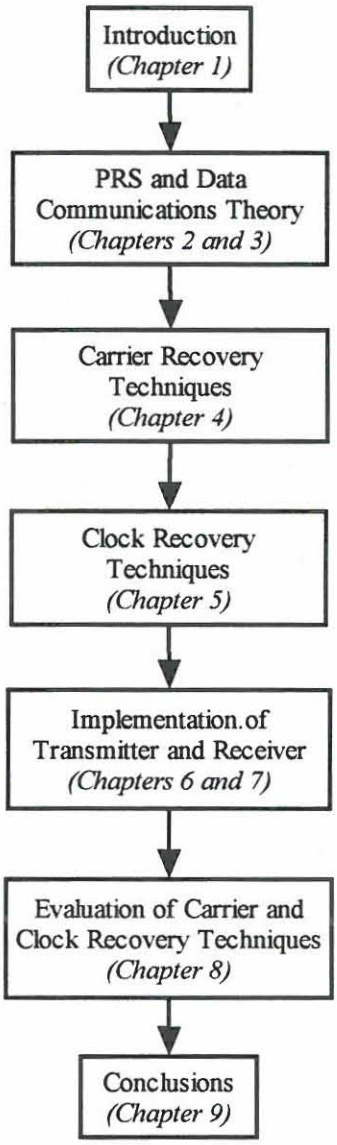


Figure 1.2: Layout of thesis.

Data communication theory is discussed with reference to principles, pulse shaping and partial response signalling. The principles and requirements of carrier synchronisation and clock recovery, as well as typical techniques employed for this purpose, are also considered.

Following this, the design, characteristics and software comprising the transmission system is described. Since the aim of the project was not to investigate DSP, but its implementation in a communication system, little theory in this regard was covered. Rather, only those aspects which influenced the programming decisions are identified.

The operation and characteristics of the data transmitter were measured and are described extensively.

Data was transmitted with the pilot tone at a wide range of different levels, and the acquisition and tracking characteristics of the carrier recovery system were evaluated. This process is described in full. The same range of measurements was also conducted with the pilot tone at a fixed level, but at different SNRs. The results obtained were, however, less conclusive. The clock synchronising characteristics of the receiver were also evaluated and the results are presented in the thesis.

1.3 GENERAL

The novel content of the thesis can be summarised as follows:

1. A special baseband filter was derived from the standard raised cosine response.
2. This response was not divided between the transmitter and the receiver. Rather, the complete response was implemented at the transmitter. This saved valuable processing time at the receiver, without any detrimental effects to the operational characteristics of the system.
3. A system of cross-correlation was used to maintain synchronisation of the local carrier frequency oscillator.
4. A limited amount of pre-processing was applied to the received 7PRS baseband signal as part of the clock recovery scheme. The pre-processing contributed towards the recovery of a substantial spectral component at the clock frequency. This characteristic was evaluated with the transmission of a number of unscrambled pseudo random data sequences. Satisfactory results were found.

CHAPTER 2

SOME PRINCIPLES OF DATA COMMUNICATIONS

Several basic characteristics of baseband data communication systems are discussed in this chapter. In particular, special attention is given to both the time and frequency responses of a number of filters used to limit the bandwidth requirements of the transmission channel used to transfer baseband data signals. During the course of the project a baseband filter with unique characteristics was developed. This response was found to ensure an exceptionally strong spectral line during clock recovery. The use of eye diagrams to evaluate the quality of both binary and multi-level baseband data signals is also considered.

2.1 INTRODUCTION

The following properties are regarded by Haykin [21, p. 200] as desirable for a digital waveform in a communication system:

- Timing content: The transmitted digital waveform should have adequate timing content to permit the extraction of clock information required for the purpose of synchronizing the receiver to the transmitter.
- Ruggedness: The waveform should possess immunity to channel noise.
- Error detection capability: The waveform should enable the detection of errors that may occur in the course of transmission due to the presence of channel noise.
- Matched power spectrum: The power spectral density of the transmitted digital waveform should match the frequency response of the channel as closely as possible in order to minimize signal distortion.
- Transparency: The correct transmission of digital data over a channel should be transparent to the pattern of 0's and 1's contained in the data.

It is hardly possible to meet all of these requirements, and developers of data communication systems normally attempt to optimize their systems with as many of

these elements as possible. Different coding and modulation schemes are therefore continuously investigated and compared in an attempt to improve the performance of communication systems to meet the increasing needs of modern society.

Line coding and modulation of digital data before transmission over a channel are used to achieve the same aim, i.e. the adaptation of a signal to the transmission channel over which it is to be transmitted. Crouch and Jedwab describe the main purposes of line coding as follows [10, p. 27]:

- It can assure that the transmitted signals are dc balanced, that is, that there are equal numbers of 0's and 1's over an extended period of time.
- Coding can aid clock synchronization by minimizing the length of strings of consecutive 0's or 1's. This ensures that there is a high density of signal transitions.
- Together with cyclic redundancy checks it can enable the detection of data errors.

Assume that the message signal is a baseband electric signal. In most cases, the significant spectral components are located close to the zero frequency. A typical transmission channel, however, has a bandpass frequency characteristic. Modulation of the baseband signal transposes the signal about the carrier frequency, i.e. to within the spectrum of the bandpass channel.

According to Lafrance [34, pp. 292-293] there are two fundamentally different techniques of transmitting information with signals:

- One alternative is to use signals which have a few degrees of freedom and distinguishing between different signals on the basis of their distinct amplitudes or phases. This approach leads to pulse amplitude modulation (PAM), phase shift keying (PSK) and other similar waveforms. The motivation for this approach is that the transmission channel may be limited in the number of available signal degrees of freedom. In the case of a severely bandwidth-limited channel, capacity can now be obtained by sufficient signal-to-noise ratio (SNR).

- The other extreme is a channel that is SNR limited. This is equivalent to a limitation on the energy available to transmit a symbol of information. For such applications, energy-efficient signalling is called for and signals with many degrees of freedom (large bandwidth) are used to provide coding gain.

No waveform is entirely time-bandwidth efficient or entirely energy efficient and a compromise is always required.

2.2 BINARY SIGNALLING

If data bits are transferred singly, that is with the symbol rate in the channel equal to the input bit rate, the signal can be written as [4, p. 57]:

$$d(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT) \qquad (2.1)$$

where $a_k = \pm 1$

$T =$ bit duration

$$g(t) = \begin{cases} 1 & for \quad 0 < t < T \\ 0 & otherwise \end{cases}$$

The rectangular pulses, g(t-kT), typically occupy a bandwidth that is much greater than the bandwidth of the channel through which they must be transmitted.

Ideally, each received voltage level should be dependent on only one transmitted bit - without any interference from other bits. However, this is normally not the case, since a pulse of finite duration theoretically requires a channel of infinite bandwidth. Bingham [4, p. 57] summarizes the basic problem for a modem as:

Finite duration $\Leftrightarrow$ Infinite bandwidth

Finite bandwidth $\Leftrightarrow$ Infinite duration

Therefore, if the transmitted/received signal is to be band limited, it cannot be constrained to one bit period. Filtering the frequency spectrum of a baseband digital signal will increase the time extension of the bit waveform so that it will overlap neighboring bit intervals causing Intersymbol interference [62, p. 306].

Expression 2.1 describes the input signal applied to a data transmission system; where $g(t)$ denotes the shaping pulse that is normalized. The amplitude a_k depends on the k-th input bit. The signal is filtered by filter(s) (see paragraph 2.3) with a transfer function $H_f(f)$. This filter output is sampled synchronously with the transmitter, and the sampling instants are determined by timing recovery circuitry. The sequence of samples thus obtained is used to reconstruct the transmitted data by means of a decision device which compares the amplitude of each sample with a threshold. Ignoring the transmission delay through the system, the received signal may be written as:

$$y(t) = \sum_{k=-\infty}^{\infty} A_k p(t - kT) + n(t) \qquad (2.2)$$

where A_k is the amplitude of the received signal. The pulse $A_k p(t)$ is the response of the transmitted pulse, $a_k g(t)$, and of the cascade connection of the above mentioned filter(s) and the channel over which the pulse is transmitted. The term $n(t)$ is the noise produced at the receiver due to the additive noise at the receiver input. The transmitted signal, $g(t)$, and the received signal, $p(t)$, are related in the frequency domain as:

$$A_k P(f) = a_k G(f) H_f(f) H_c(f) \qquad (2.3)$$

where $G(f)$, $P(f)$ and $H_c(f)$ are the Fourier transforms of g(t), p(t) and of the channel characteristics respectively.

The receiver samples $y(t)$ at time $t = iT$ (with i taking on integer values), yielding [19, p. 468]:

$$y(t_i) = \sum_{k=-\infty}^{\infty} A_k p[(i-k)T] + n(t_i)$$

$$= A_i + \sum_{\substack{k=-\infty \\ k \neq i}}^{\infty} A_k p[(i-k)T] + n(t_i) \qquad (2.4)$$

A_i, in expression 2.4, represents the i-th transmitted data bit. The second term represents the residual effect of all other transmitted bits on the decoding circuitry with respect to the i-th bit. This effect is due to intersymbol interference (ISI). The last term, $n(t_i)$, represents the noise sample at time t_i. ISI is due to both precursors and postcursors and it is possible to distinguish between the effect of these two elements by rewriting expression 2.4 as follows [67, p. 36]:

$$y(t_i) = A_i + \sum_{k=1}^{\infty} A_{-k} p_k + \sum_{k=1}^{\infty} A_k p_{-k} + n(t_i) \qquad (2.5)$$

The second term defines the effect due to postcursors, whilst the effect of the precursors is defined by the third term.

From the above it is clear that, in the absence of ISI and noise, signal A_i can be recovered. It is, however, also clear that ISI cannot be avoided, therefore a primary objective in the design of the transmission filter(s) is to minimize the effect of ISI and noise in general.

2.3 DISTORTIONLESS BASEBAND TRANSMISSION

The transfer function of the channel (and the transmitted pulse shape) is specified. Timing inaccuracies in the transmitter cause ISI if the rate of transmission does not conform to the ringing frequency designed into the channel. Timing inaccuracies of this type are insignificant unless extremely sharp filter cut-offs are used while signalling at the Nyquist rate [2, p. 168].

The problem is to determine the transfer functions of the receiver and transmitter filters to optimally reconstruct the transmitted data at the receiver. At this stage, the required combined response of the filtering will be considered.

The ideal solution is to use a filter with no output frequency components exceeding half the bit rate. One signal waveform that produces zero ISI is defined by the sinc function:

$$p(t) = \frac{\sin(2\pi f_1 t)}{2\pi f_1 t} \tag{2.6}$$

$$= \sin c(2 f_1 t) \tag{2.7}$$

where $f_1 = \frac{1}{2T}$.

Figure 2.1 shows plots of the required frequency response and basic pulse shape of an ideal (sinc) system [20, p. 248].

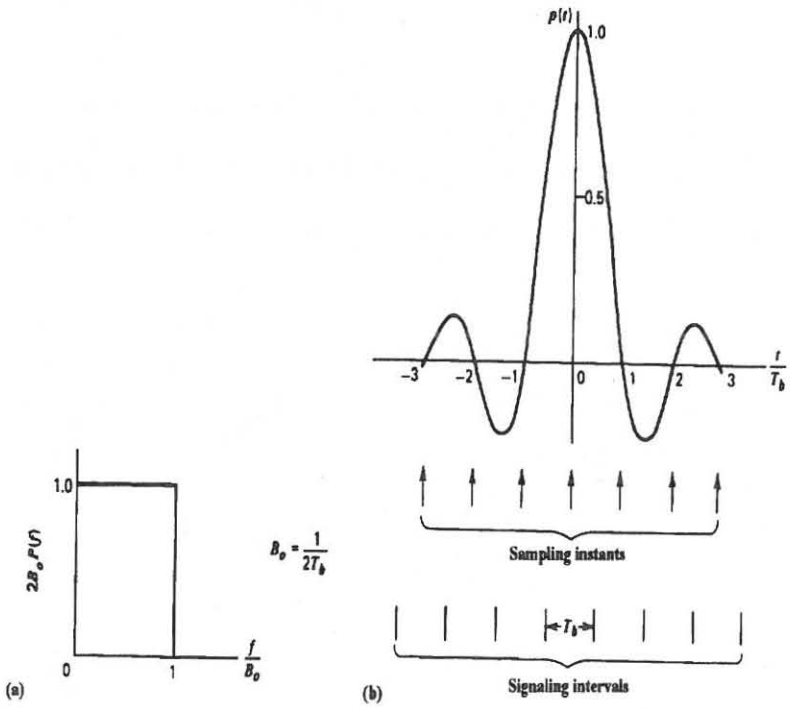


Figure 2.1: (a) Frequency and (b) time response of an ideal lowpass filter [20, p. 248].

Although this choice of pulse shape solves the problem of ISI with a minimum of bandwidth, there are two practical difficulties that make its implementation undesirable [62, p. 305]:

- It requires that the frequency characteristic of the filter ($H_f(f)$ in expression 2.3) be flat from $-f_1$ to f_1, and zero elsewhere - a physically unrealizable condition.
- The function has a very slow rate of decay - as is obvious from Figure 2.1(b). Accordingly, there is practically no margin for error in sampling times in the receiver.

These practical difficulties can be overcome by extending the bandwidth to an adjustable value of between f_1 and twice this value. Several band-limited functions satisfy this condition.

2.3.1 RAISED COSINE RESPONSE

A particular form of response that is suitable for this application is the raised cosine spectrum. It consists of a flat portion at low frequencies and a roll-off portion with a sinusoidal shape. This characteristic can be expressed as [57, p. 387]:

$$X(\omega) = \begin{cases} T & 0 \le |\omega| \le (1-\alpha)W \\ \frac{T}{2}\left\{1 - \sin\left[\frac{\pi}{2\alpha W}(|\omega| - W)\right]\right\} & (1-\alpha)W \le |\omega| \le (1+\alpha)W \\ 0 & |\omega| > (1+\alpha)W \end{cases} \quad (2.8)$$

where $W = \frac{\pi}{T}$

α - roll-off factor

The roll-off factor of the filter, α, is an indication of the excess bandwidth passed by such a filter. Tröndle and Söder [67, p. 190] define this value as follows:

> Frequency f_I is defined as the cut-off frequency of the raised cosine filter, whilst f_2 is the frequency outside of which the frequency response is zero (see Figure 2.2).

From this f_I can be written as:

$$f_I = \frac{f_1 + f_2}{2} \tag{2.9}$$

where $f_1 < f_I < f_2$

Then $$\alpha = \frac{f_2 - f_1}{f_2 + f_1} \tag{2.10}$$

These values are shown in Figure 2.2 [67, p. 191].

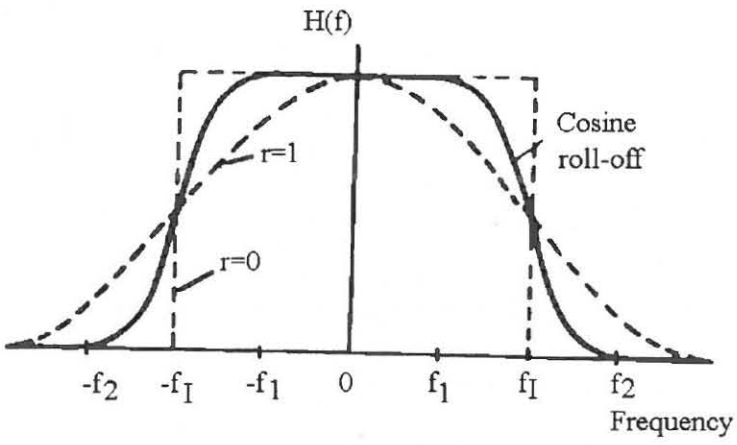


Figure 2.2: Raised cosine frequency response (r = excess bandwidth) [67, p. 191].

Tröndle and Söder [67, p. 191 and p. 192] specify $f_I \approx W$ and $\alpha \approx 0.2$ as optimum values for a non-redundant binary system. Varying the pulse shaper cut-off frequency,

f_1, at the optimum roll-off factor produces a rapid loss in the signal-to-noise ratio relative to the optimum cut-off frequency.

The impulse response of the raised cosine response is [57, p. 388]:

$$x(t) = \left(\frac{\sin(Wt)}{Wt}\right)\left(\frac{\cos(\alpha Wt)}{1-(2\alpha Wt/\pi)^2}\right) \qquad (2.11)$$

The first term represents a $\frac{\sin x}{x}$ waveform. This retains the original zero crossings of the waveform. The second term is a result of the more gradual roll-off, α. The resulting bandwidth exceeds the ideal value by an amount equal to αf_1 [21, p. 236]. Figure 2.3 shows the time response of a raised cosine filter.

To summarize, the bandwidth occupied for a raised cosine-type transmission characteristic varies from a minimum of 1/2T to a maximum of 1/T. The larger values of α lead to faster decaying pulses so that receiver synchronization will be less critical and modest timing errors will not cause large amounts of ISI [57, p. 388].

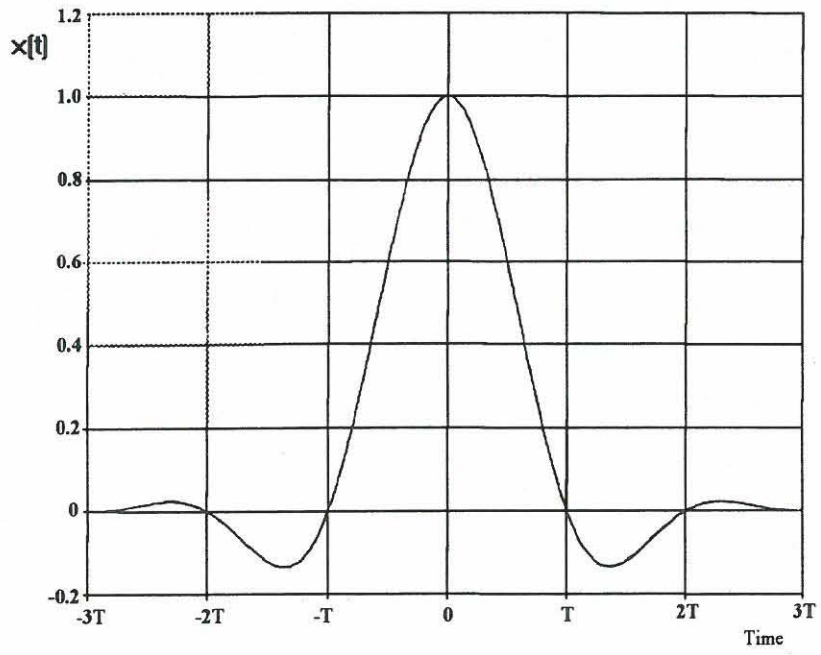


Figure 2.3: Raised cosine time response [15, p. 406]

It is significant that with $\alpha = 1$ the response goes to zero, not only at the zeros of $\frac{\sin(Wt)}{Wt}$, but also at points midway between these sample values. It is therefore possible to sample the input at the same rate as for the ideal channel, with no resulting ISI [49, p. 290].

2.3.2 RESPONSE OF PULSE SHAPING FILTER

Due to the problems associated with the synchronization of PRS signals - the main field of study of the project - a special pulse shaping filter (PSF) has been developed. The impulse response of this filter is:

$$x(t) = \left(\frac{\sin(2Wt)}{2Wt}\right)\left(\frac{\cos(2\alpha Wt)}{1-(4\alpha Wt/\pi)^2}\right) + \left(\frac{\sin(2W(t-T))}{2W(t-T)}\right)\left(\frac{\cos(2\alpha W(t-T))}{1-(4\alpha W(t-T)/\pi)^2}\right) \quad (2.12)$$

This response requires a substantially wider bandwidth than the normal raised cosine filter for the same values of α, but less than a raised cosine response using $2W$ as cut-off frequency. The frequency response of the pulse shaping filter, a raised cosine filter designed for $2W$ (referred to in this document as a $2W$ raised cosine filter) and that of a standard raised cosine filter are shown in Figure 2.4.

The characteristics of these filters have been studied extensively during the course of the project and either the pulse shaping, or the $2W$ raised cosine filter, could be used as a receiver/transmitter filter. However, the narrower bandwidth of the pulse shaping filter makes this filter the preferable type. Figure 2.5 shows the pulse shape of the filters under discussion. In all three cases, a value of 0.2 was used for α.

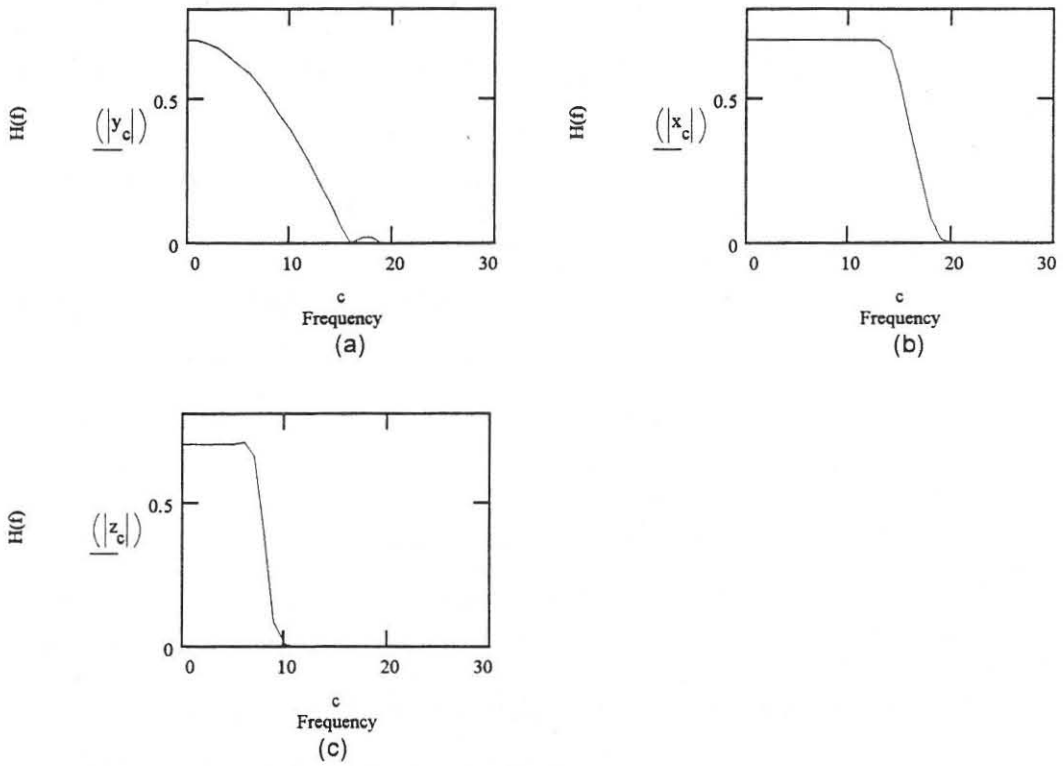


Figure 2.4: Comparative frequency responses of (a) pulse shaping, (b) $2W$ raised cosine and (c) standard raised cosine filters.

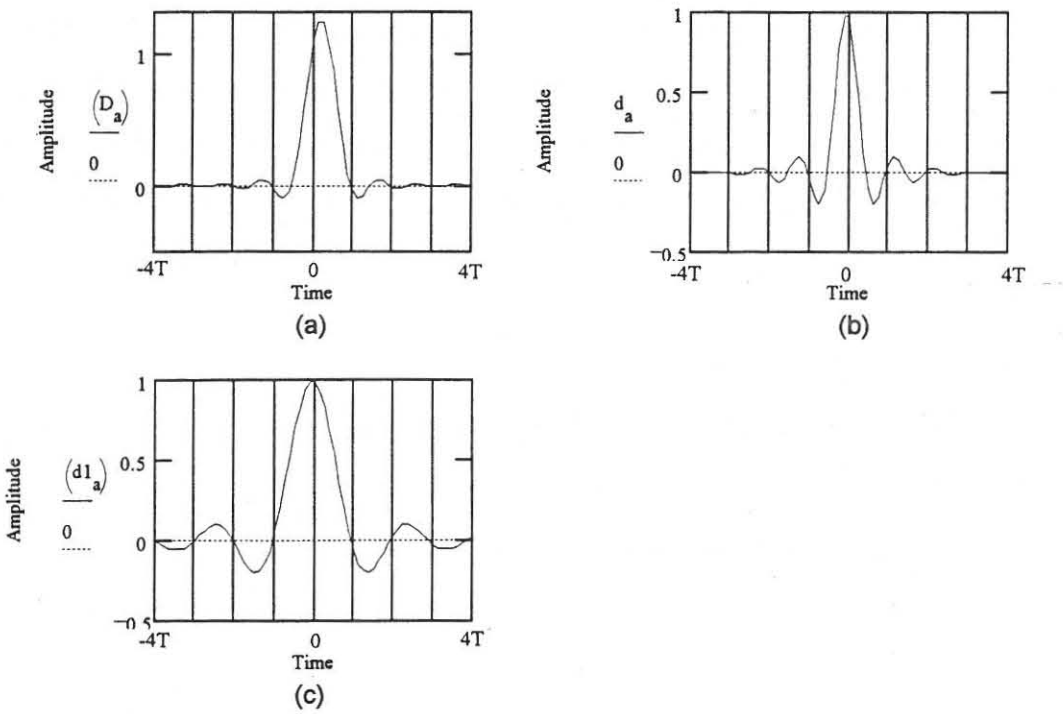


Figure 2.5: Pulse shapes of (a) pulse shaping, (b) $2W$ raised cosine and (c) standard raised cosine filters.

Inspection of Figure 2.5 reveals the following important characteristics:

- The duration of curve (c) is substantially longer than that of the other two curves.
- In all three cases, the amplitude is zero at multiples of the symbol rate (identified by the vertical lines in the figures). Therefore, neither of the responses indicate ISI if sampling is confined to the correct instants.
- In the case of curves (a) and (b), there is also zero amplitude at time $(nT/2)$ where n is an integer. This is not the case with the standard raised cosine filter and should result, in the case of curves (a) and (b), in less noise on the channel between sampling instants.
- Curve (a) does not reach a peak amplitude at time t (the instant when a sample is taken). This will cause a deterioration in the noise level and gives rise to more stringent synchronization requirements.
- Curve (a) decreases much more rapidly than the other two curves and should, therefore, produce less noise on the channel.

2.4 POSITION OF FILTER

An important consideration for the designer of a data communications system is not only the characteristics of the filter to be used, but also its position in the system. However, this can only be addressed once it has been determined whether one combined filter for the transmitter and the receiver can be used, or whether two separate filters, with a combined response as described in paragraph 2.3, are required.

In this regard, [26, pp. 345-346] and [15, p. 407] state that in most applications the filtering operation is split between two filters, one at the transmitter, with a transfer function denoted $H_T(f)$, and one at the receiver, with transfer function denoted by $H_R(f)$. With an impulse at the input to $H_T(f)$, and an ideal channel, the overall response should still be (see paragraph 2.2) :

$$H_f(f) = H_T(f)H_R(f) \qquad (2.13)$$

In the case where an additive noise source is assumed at the receiver input, it can be shown that the optimum partition, in the sense of optimizing the SNR at the receiver output, is to split $H_f(f)$ equally between the transmitter and the receiver [17, p. 236]. Therefore:

$$H_T(f) = H_R(f) = \left[H_f(f)\right]^{0.5} \tag{2.14}$$

It is, therefore, common practice not to use a single raised cosine filter, but rather two filters, each with a root raised cosine response. Bingham also supports this approach [4, p. p64]. He writes:

> If the output power of the transmitter is fixed, and noise is added somewhere in the channel, then it is reasonable - and has been proved - that the filtering should be split equally between the transmitter and the receiver.

Killen [30, p. 201] also supports the utilization of two filters - designed with a composite characteristic so that no ISI occurs at the instant of sampling. He maintains that the transmitter filter can be specified to limit the transmitted signal to the available transmission channel. The receiver filter can then be specified to limit adjacent-channel interference.

Finally, attention can be drawn to the fact that with two filters, the dual criteria of distortionless transmission and minimum noise power (or maximum SNR) can be met [26, p. 348].

The CCITT specifications for V.27 *bis* data communications standard specifies a raised cosine energy spectrum shaping, equally divided between the transmitter and the receiver [7, p. 187]. However, with respect to the V.37 standard, specifying a class IV partial response signalling standard, the following statement is made [7, p. 276]:

> Baseband signal shaping is performed at the transmitter.

Considering the last statement, use of a single filter at the transmitter is not preposterous.

2.5 EYE DIAGRAMS

Eye diagrams are convenient for determining the quality of a pulse train. These diagrams can be used to assess the effect of random noise, jitter and ISI of an incoming signal [54, p. 387]. Since the project is mainly concerned with the synchronization of a PRS signal using a PSF response - which tends to generate substantial ISI - special attention was given to eye diagrams as a means of assessing the characteristics of the system. The quality of an eye opening is affected by two factors - the pulse shaping and the driving symbol sequence [31, p. 1235].

If an eye tends to close from top to bottom, a small amount of noise will cause an error when sampling the received signal. If the eye is not very wide, small variations in the location of the sampling time could result in sampling at instants when the noise margin is small - and errors will be more likely. Figure 2.6 [3, p. 343] shows the effect of noise and timing errors on the eye diagram of a two-level system.

The ratio of the actual opening to the maximum possible opening represents the degradation caused by ISI in the absence of noise. For a two-level system the peak ISI is about 30%. The equivalent degradation in SNR is $20\log(1-0.3) = -3.1$ dB [54, p. 388]. When there is ISI, the distance from the horizontal axis to the inside edge of the eye at the sampling time becomes smaller [17, p. 289].

Specific eye openings in a 7PRS system are numbered as shown in Figure 2.7.

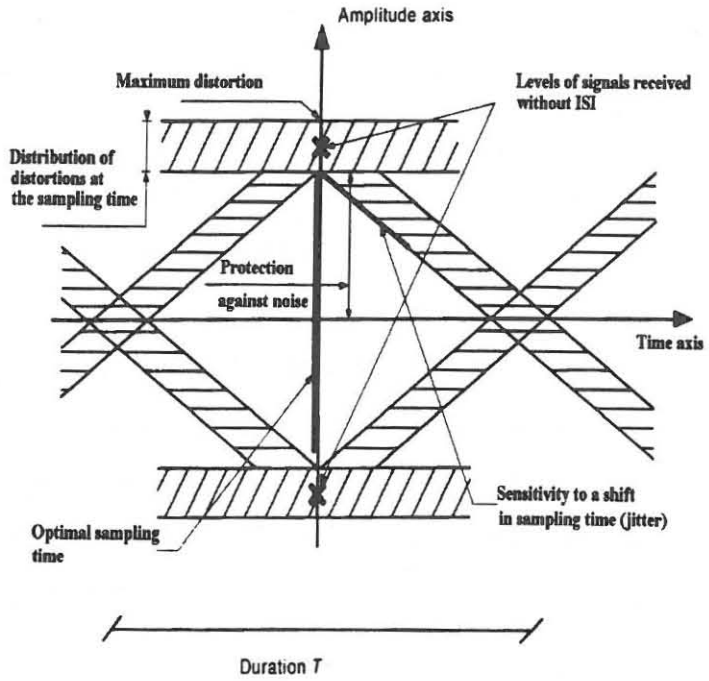


Figure 2.6: Interpretation of the eye diagram of a two-level signal [3, p. 343].

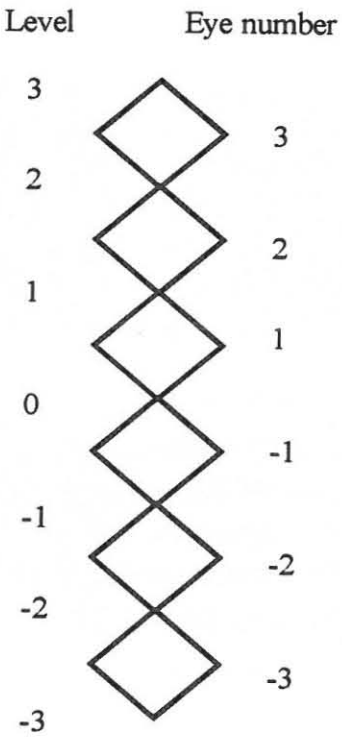


Figure 2.7: Numbering of eye openings in a 7PRS system.

The actual width of a specific eye is a function of its number, with lower bound eye widths of a 7PRS signal (filtered with a raised cosine filter) as follows [31, p. 1238]:

Eye number 1: 0,182T

Eye number 3: 0,093T

where T = symbol period

Obviously, the width of an eye is not only a function of its number, but also a function of the number of levels in a signal. The value for eye number 1 in a 3PRS signal is, for instance, equal to 0,667T. This implies the need for much more stringent stability requirements for the synchronization circuitry of a higher level system than for a lower level system.

Measured eye diagrams for two 7PRS signals are shown in Figure 2.8. The first signal is filtered by an ordinary raised cosine filter, whilst a pulse shaping filter is used for the second. From these diagrams the following comments can be made about the pulse shaping filter response:

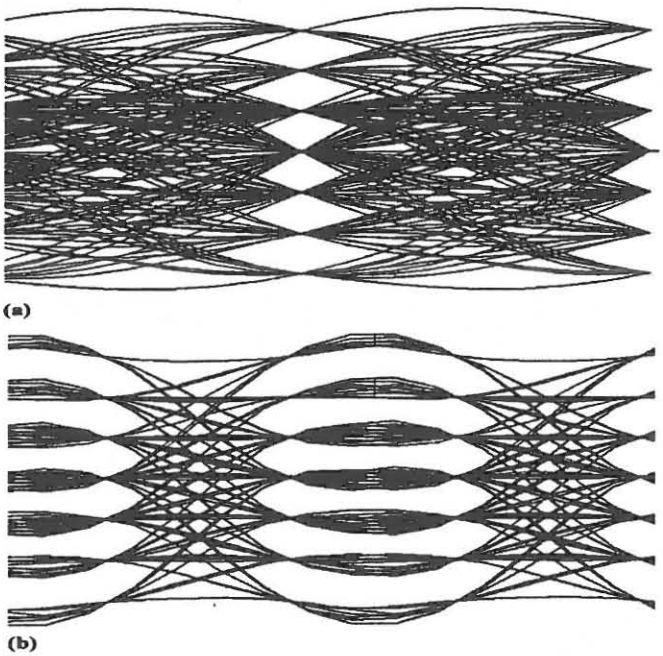


Figure 2.8: 7PRS signals filtered with (a) raised cosine and (b) pulse shaping filters.

- The maximum eye opening is the same as for a raised cosine response. However, Figure 2.8(b) shows that use of the pulse shaping filter results in two such instants during each symbol period.
- Due to the high cut-off frequency of the filter the signal rise and fall times are shorter than for the raised cosine response. This results in a much wider eye opening.
- Substantial ISI occurs between the two ideal sampling instants.

From the above it is clear that, although the PSF causes substantial ISI, it is possible to detect the level of an incoming signal, filtered with such a filter.

2.6 SUMMARY

Due to the limited bandwidth of practical transmission channels some form of pulse shaping of the transmitted signal is required. The raised cosine filter response is suitable for this purpose. Due to problems associated with the synchronization of PRS signals, a special pulse shaping filter response has been defined. This response produces substantial ISI but - considering its eye diagram - can be used as a pulse shaper for partial response signals.

CHAPTER 3

PARTIAL RESPONSE SIGNALLING

The characteristics and coding of a number of different correlative coding techniques are described in this chapter. Special attention is given to Class I 49QPRS. A number of advantages, as well as disadvantages, of using partial response signalling is also discussed.

3.1 INTRODUCTION

Partial response signalling (PRS) or correlative coding, as it is sometimes called, was invented by Lender and generalised by Kretzmer [43, p. 179]. It is a data coding technique that uses controlled amounts of ISI to achieve convenient spectral properties, which (inter alia) can make the system less sensitive to timing errors [29, p. 921]. The advantage of this approach is that, for a given bandwidth and power input to the channel, correlative techniques permit more bits/second/hertz than conventional Nyquist zero-memory systems for a given error probability [63, p. 191], thus it may be regarded as a practical means of achieving the theoretical maximum signalling rate [21, p. 237].

Kim and Kim [32, p. 152], in defining the conditions for minimum-bandwidth codes, state that a condition for non-zero eyewidth in the corresponding line signal is obtained by ideally bandlimiting the code sequence to Nyquist's minimum. In [31, p. 1235] the minimum-bandwidth partial response system is referred to as a linear subclass of minimum-bandwidth system.

It has been proven that the theoretical maximum symbol rate in a bandwidth of f_s hertz is $2f_s$ symbols/second, or equivalently, that the maximum possible bandwidth efficiency is 2 symbols/sec/Hz [9, p. 20]. If 4 bits/Hz are desired, four input levels must be used and the output sequence consists of 7-level symbols. This is referred to as a

7PRS signal. Table 3.1 [9, p. 31] shows the practical bandwidth efficiency of PRS versus memoriless systems.

Table 3.1: Comparative number of levels in PRS and M-ary signalling [9, p. 31].

Bandwidth efficiency (Bits/sec/Hz)	Number of levels in memoriless system with α =1	Number of levels in correlative systems
1	2	
2	4	3
3	8	
4	16	7
5	32	
6	64	15
7	128	
8	256	31

However, PRS also has some negative characteristics, e.g. systems that use symbol by symbol detection possess reduced noise margins due to the fact that the superposition of signal waveforms cause the number of output levels to be larger than the number of input levels [29, p. 921].

3.2 CLASS I PARTIAL RESPONSE SIGNALLING

Although many different types (or classes) of correlative coding have been developed, the discussion will be limited to the so-called Class I coding scheme for the moment. This is the coding technique which has been used during execution of the experimental work of the project.

3.2.1 CODING OF TERNARY PARTIAL RESPONSE SIGNALS (3PRS)

This coding procedure generates the sum of a binary input data stream and a one bit-period delayed version of the same data. Thus, the duobinary signal can be represented by:

$$b_k = a_k + a_{k-1} \tag{3.1}$$

where b_k is the k-th coded value

a_k is the k-th binary input bit

This is implemented by a transversal delay filter - as shown in Figure 3.1 - with the following impulse response [9, p. 21]:

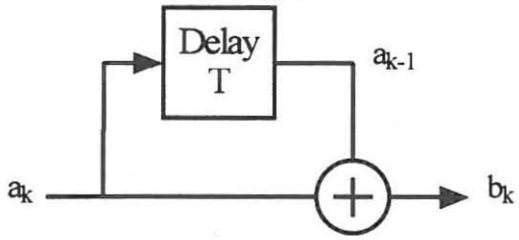


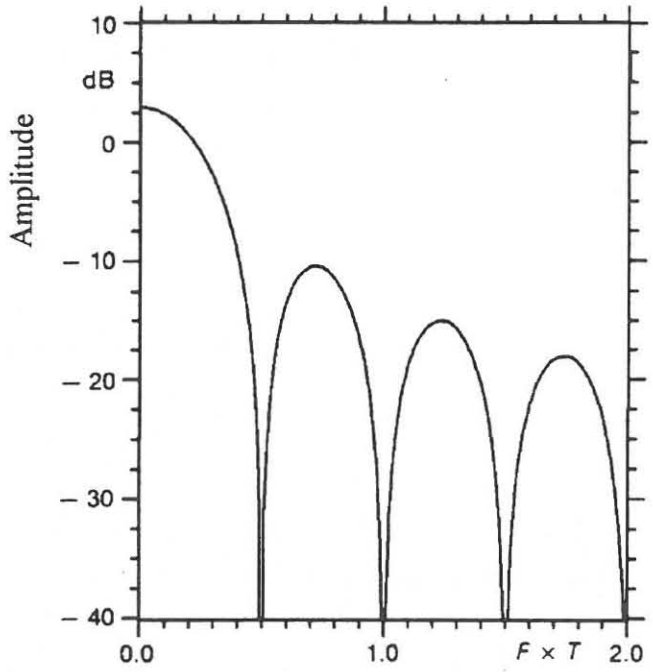
Figure 3.1: Duobinary coder.

$$f'(t) = \delta(t) + \delta(t - T_b) \tag{3.2}$$

where T_b is the symbol spacing. The output of this circuit is a three level signal, such as that for a binary (0 or 1) input, the output is one of three values (0, 1 or 2). This is due to the controlled ISI that is caused by use of the delay filter. This corresponds to a transfer function [19, p. 478]:

$$F'(W) = (1 + e^{-jwt}) \tag{3.3}$$

with the first null occurring at $\frac{1}{2T_b}$. Figure 3.2 shows the power spectral density of the output signal [3, p. 33].



where F = Fundamental frequency of 7PRS signal

T = Period of fundamental frequency of 7PRS signal

Figure 3.2: Power spectral density of Class I duobinary code [3, p. 33].

The output can be lowpass filtered at the first null - which corresponds to the Nyquist frequency. Correlative coding therefore results in ISI in such a way that it is caused only by the preceding symbol - in a controlled manner. The resulting baseband signal has regularly spaced sampling instants when the signal consists of contributions of only two adjacent bits.

The receiver should sample the signal at the optimum instants and decode the output according to the rule:

$$d_k = b_k - d_{k-1} \tag{3.4}$$

where d_k is the k-th decoded output value

b_k is the k-th coded input value

From expression 3.4 it is obvious that, in order to decode any bit correctly, it is required that the correct value of the preceding bit is known. However, if an error occurs and a bit is decoded incorrectly, this error will be propagated. A practical means of avoiding error propagation in detecting a duobinary signal is to use precoding [21, p. 241].

3.2.2 PRECODING

In a duobinary system the precoded signal is formed by the following processing:

$$b_k = (a_k - b_{k-1}) \qquad \text{mod-2} \qquad (3.5)$$

Figure 3.3 shows this configuration. The coder output can now be described by the following expression:

$$c_k = b_k + b_{k-1} \qquad \text{where } 0 \le c_k \le 2 \qquad (3.6)$$

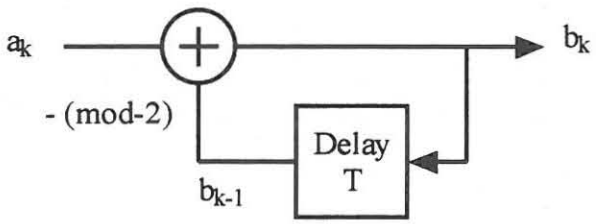


Figure 3.3: Duobinary precoder.

Output c_k can now, like b_k in expression 3.1, assume any of three values. The operation of the encoder is shown in Table 3.2 for a typical sequence of data bits. Figure 3.4 shows a block diagram of a duobinary encoder. The term encoder is used throughout this document to refer to a combination of a precoder and a coder circuit.

Table 3.2: 3PRS coding and decoding of a sample data stream.

Time (k)	-1	0	1	2	3	4	5	6	7	8	9	10
a_k		1	1	0	1	0	0	1	0	1	0	1
b_k	0	1	0	0	1	1	1	0	0	1	1	0
c_k		1	1	0	1	2	2	1	0	1	2	1
d_k		1	1	0	1	0	0	1	0	1	0	1

Table 3.3 (adapted from [62, p. 298]) shows the possible input conditions, as well as the corresponding outputs, of the 3PRS encoder. It should be noted that the encoder output can take on one of three values - but with different probabilities of occurrence. The probability of the output being a one is 0,5, twice that of both zero and two. This is an important characteristic and will be discussed in detail in paragraph 3.5. It is also of significance that the output never changes directly from zero to two, or vice versa. Indeed, a one output indicates a change in b_k. If there is no change in b_k, the output is either zero or two.

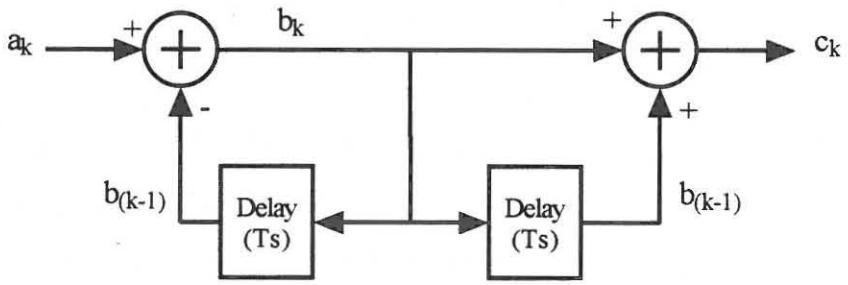


Figure 3.4: Duobinary precoder and coder.

Table 3.3: Encoder input and output conditions of 3PRS signal.

Precoder			Coder	Output	Probability
a_k	b_{k-1}	b_k	$b_{k-1} \rightarrow b_k$	c_k	P
0	0	0	$0 \rightarrow 0$	0	0,25
0	1	1	$0 \rightarrow 1$	1	0,25
1	0	1	$1 \rightarrow 0$	1	0,25
1	1	0	$1 \rightarrow 1$	2	0,25

Decoding of this signal is done on a bit by bit basis and is described by the following rule:

$$d_k = c_k \qquad \text{mod-2} \tag{3.7}$$

If the decoder input is seen as a two digit binary number, the most significant bit is simply discarded and the output, d_k, is equal to the least significant bit (see Figure 3.4). As is to be expected, the output, d_k, equals the input data.

Although precoding was used in the project, according to [29, p. 934], results show that for a reasonable error rate (10^{-5} or below), precoding does not decrease the expected BER significantly - and in some instances, actually increases it.

3.3 OTHER CORRELATIVE CODING TECHNIQUES

3.3.1 7PRS

The principles as described above for 3PRS are also valid for other partial response signalling techniques. The coding of 7PRS signals can also be described with respect to the operation of the precoder and coder. Assume a serial data stream has to be coded into the 7PRS format. The incoming bits are grouped into pairs. The stream of paired bits (symbols) will be referred to as input symbol a_k, each symbol having an integer value between 0 and 3 (both included). The operation of the precoder is as shown in Figure 3.5 and the output is:

$$b_k = (a_k - b_{k-1}) \qquad \text{mod-4} \qquad \text{where } 0 \le b_k \le 3 \tag{3.8}$$

The precoded sequence is now passed through a coder - as described for a 3PRS signal and shown in Figure 3.1. The output of the coder is a 7-level signal.

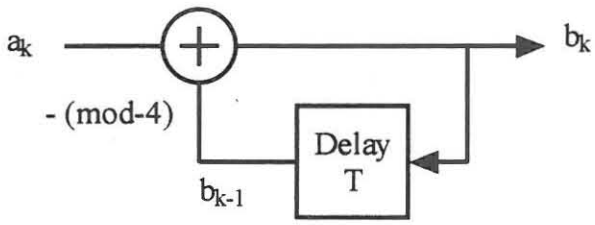


Figure 3.5: 7PRS precoder.

$$c_k = b_k + b_{k-1} \qquad \text{where } 0 \le c_k \le 6 \qquad (3.9)$$

The complete encoding process of a 7PRS system is shown in Figure 3.6, where a_k and b_k are dibits and c_k is the 7PRS output.

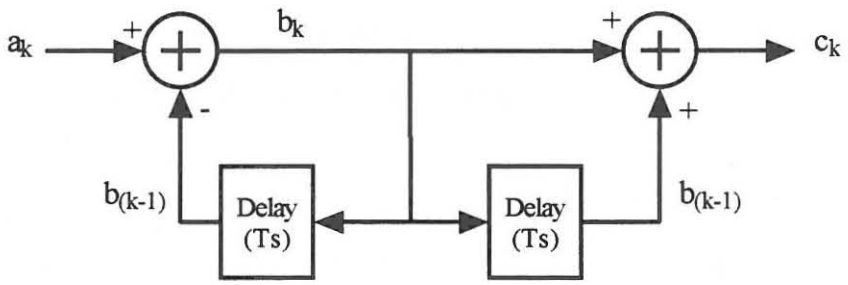


Figure 3.6: Complete 7PRS encoder.

During decoding of the 7PRS signal, a modulo-4 operation is executed on the transmitted signal, converting this into an integer value between zero and three. Thus:

$$d_k = c_k \qquad \text{mod-4} \qquad (3.10)$$

where d_k is the decoded output. The encoding and decoding process is demonstrated in Table 3.4 for a random data sequence.

Table 3.4: 7PRS encoding and decoding of a sample data stream.

Time (k)	-1	0	1	2	3	4	5	6	7	8	9	10
a_k		1	3	0	2	3	3	1	0	1	2	1
b_k	0	1	2	2	0	3	0	1	3	2	0	1
c_k		1	3	4	2	3	3	1	4	5	2	1
d_k		1	3	0	2	3	3	1	0	1	2	1

Incidentally only five of the seven possible levels appear in Table 3.4. With another data stream it might well be that all seven values appear in such a table.

3.3.2 MODIFIED DUOBINARY

If only adjacent bits or symbols are used during encoding, the technique is called duobinary. If bits or symbols separated by one keying interval are combined, the technique is called modified duobinary [30, p. 68].

As shown in Figure 3.1, duobinary coding, as described in paragraphs 3.2 and 3.3.1, causes a direct current component in the spectral response. In many applications this is not acceptable. Modified duobinary (PRS Class IV) does not cause this in the coded signal [21, p. 243]. Figure 3.7 shows the block diagram of such an encoder.

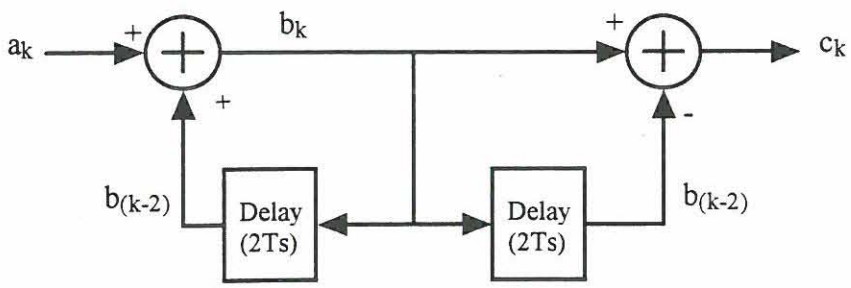


Figure 3.7: Modified duobinary encoder.

The output of the precoder and coder respectively can be expressed as follows:

$$b_k = a_k \oplus b_{k-2} \tag{3.11}$$

$$c_k = b_k - b_{k-2} \tag{3.12}$$

Because the input and precoder output can both assume one of two values - zero or one - the final output, c_k, can be 0, 1 or 2. If $c_k \neq 1$, the decoded output is 0, otherwise it is 1. Thus, knowing c_k, the output can be determined uniquely without depending on the previous digit, thereby eliminating error propagation [35, p. 192].

PRS codes used today are those which introduce zeroes into the spectrum of the transmitted signal with the aim of minimising the effect of certain very localised faults, such as, those created by a transmission channel which will not pass direct current [3, p. 139].

The principle of introducing extra zeroes into the spectrum can be applied to modified duobinary. Such a system, referred to as a polybipolar system, might require the following precoding [4, p. 71]:

$$b_k = a_k - b_{k-1} + b_{k-2} + b_{k-3} \quad \text{mod-}L \qquad \text{where } L > 2 \tag{3.13}$$

L in expression 3.13 identifies the number of levels in the resulting waveform. Longer correlation spans result in more energy being concentrated in the baseband, thus conserving spectrum. However, this also requires larger memories and greater transmitter power [30, p. 68].

Table 3.5 identifies different classes of PRS, as well as the corresponding generating equations [57, p. 576].

Table 3.5: Classes of PRS [57, p. 576].

Class	Generating equation	Magnitude of freq. transfer $\lvert H(\omega)\rvert$
1	$c_k = a_k + a_{k-1}$	$2\cos\left(\frac{\omega T}{2}\right)$
2	$c_k = a_k + 2a_{k-1} + a_{k-2}$	$4\cos^2\left(\frac{\omega T}{2}\right)$
3	$c_k = 2a_k + a_{k-1} - a_{k-2}$	$\left\{\left[2+\cos(\omega T)-\cos(2\omega T)\right]^2 + \left[\sin(\omega T)-\sin(2\omega T)\right]^2\right\}^{0,5}$
4	$c_k = a_k - a_{k-2}$	$2\sin(\omega T)$
5	$c_k = a_k - 2a_{k-2} + a_{k-4}$	$4\sin^2(\omega T)$

From Table 3.5, it is obvious that different spectral responses can be achieved to suit special needs with respect to spectral properties.

3.3.3 PREFERRED CLASSES OF PRS

The two systems which stand out on the basis of performance are duobinary and modified duobinary. These are also the ones that, in the past, have been put into practice both because of their simplicity and their useful spectral shapes.

3.4 POSITION OF CODER

A complete PRS transmitter/receiver system is shown in Figure 3.8(a). However, the coding network contributes to noise-power limitation if the frequency spectrum modifying circuitry is transferred from the transmitter to the receiver [67, p. 185]. This results in an arrangement, with the precoder at the transmitter and the coder at the receiver - as shown in Figure 3.8(b). Application of duobinary codes at the receiver results in a reduction of the noise power by up to 8 dB.

This principle is also addressed in [3, p. 147]. In this case, the possible splitting of the coding process between the transmitter and the receiver is discussed.

3.5 DECISION LEVELS

Figure 2.8(a) and (b) show the eye diagrams of two 7PRS signals, produced by filtering, using different filters. Detection at the receiver is performed by comparing the received continuously varying signal, to a set of decision thresholds that are halfway between adjacent levels. However, since the vertical eye openings of all levels are not necessarily the same, these decision levels may not always be separated by the same amount.

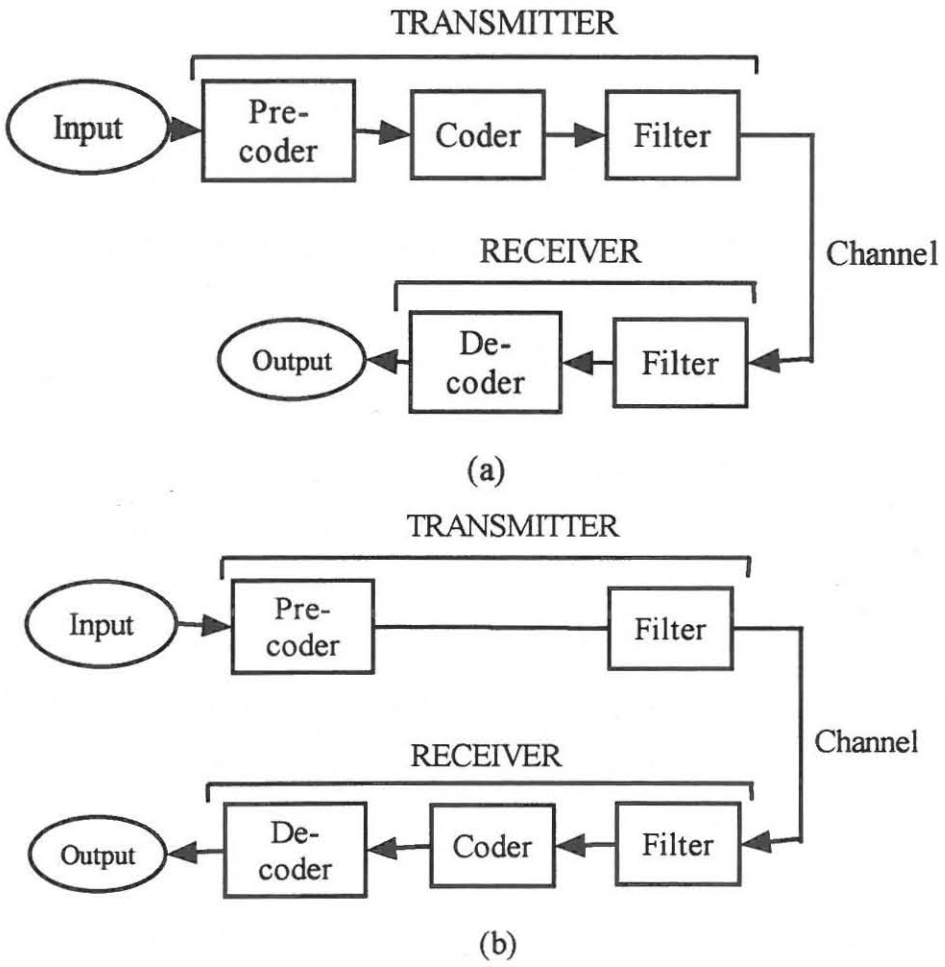


Figure 3.8: Duobinary systems with the coder at (a) the transmitter, (b) the receiver.

Consider a 3PRS signal. As explained in paragraph 3.2.2, different levels of the PRS signal have different probabilities of occurrence; 0,25 for levels 0 and 2, and 0,5 for level 1. Since 1's tend to occur more often than 0's and 2's [P_1>(P_0 and P_2)], the decision level between 0 and 1 should be shifted negatively, and that between 1 and 2 should be shifted positively. This increases the probability of an error in level 1, but improves the overall reliability of the system. This characteristic is also true for 7PRS signals, as shown in Figure 3.9, which indicates that the different levels of a 7PRS signal also have different probabilities of occurrence.

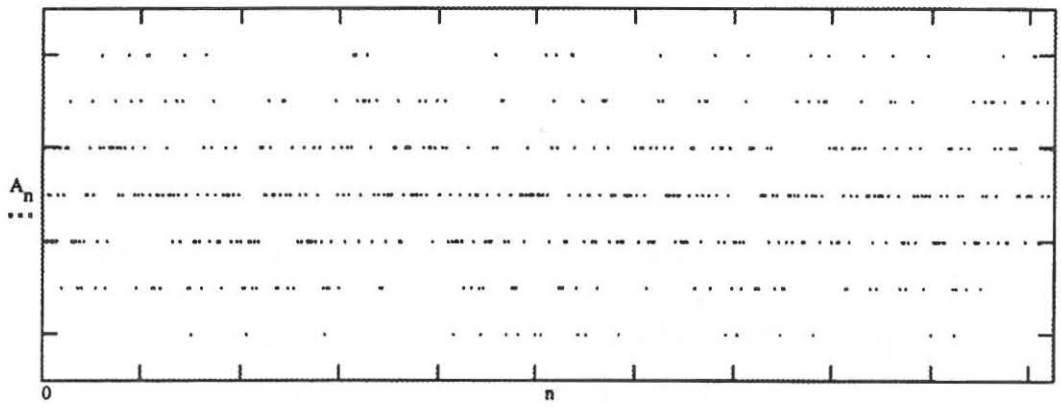


Figure 3.9: Density of different levels of a pseudo random 7PRS sequence.

The optimum decision levels depend on the form of the density functions [51, p. 431]. If the different levels are equally likely to occur, then, in the presence of additive noise, the overall error probability is minimised by placing the threshold halfway between the different expected voltages [5, p. 85].

With the addition of noise, the most likely error is that a signal will cross just one (the closest) threshold [4, p. 65]. It is therefore common practice to use Gray coding in PRS since the incorrect decoding of a symbol - as being one level too high or too low - will result in only one bit being decoded at the wrong logical condition.

3.6 MODULATION OF PRS SIGNALS

Partial response signals are normally modulated before transmission. Phase shift keying (PSK), frequency shift keying (FSK) or amplitude shift keying (ASK) are commonly

used for this purpose. Haykin [21, p. 591] makes the following statement regarding the use of the first two modulation schemes mentioned above:

- In the case of M-ary PSK, as the number of phase levels, M, is increased, the bandwidth efficiency is improved but at the expense of an increase in the required signal energy per bit.
- In the case of M-ary FSK, as the number of frequency levels, M, is increased, the required signal energy per bit is decreased, but at the expense of reduced bandwidth efficiency.

During this project, only amplitude modulation has been investigated.

The bandwidth of a baseband 7PRS signal is $\frac{f_b}{2}$. The total bandwidth of the amplitude modulated signal, which is twice the bandwidth of the baseband signal, is therefore equal to f_b. By splitting the data stream into two parallel series and simultaneously modulating both as 7PRS signals in quadrature (one with a sine wave and the other with cosine), it is possible to maintain the data rate with a 50% reduction in bandwidth (down to $\frac{f_b}{2}$).

If QPSK rather than QPRS were used, the bandwidth required would be f_b, twice that required by ASK [62, p. 311]. Class I 7PRS has a maximum bandwidth efficiency of 4 bits/sec/Hz. Modulating this signal onto a carrier doubles the bandwidth and therefore halves the bandwidth efficiency to 2 bits/sec/Hz. Adding two of these signals in quadrature restores the efficiency to 4 bits/sec/Hz [9, p. 27]. This is referred to as a 49QPRS signal.

In PSK and amplitude modulation it is customary to display the eye diagram as a two-dimensional scatter diagram, illustrating the sampled values that represent the decision variables at the sampling instants [46, p. 530]. Figure 3.10 shows the "constellation" of a 49QPRS signal.

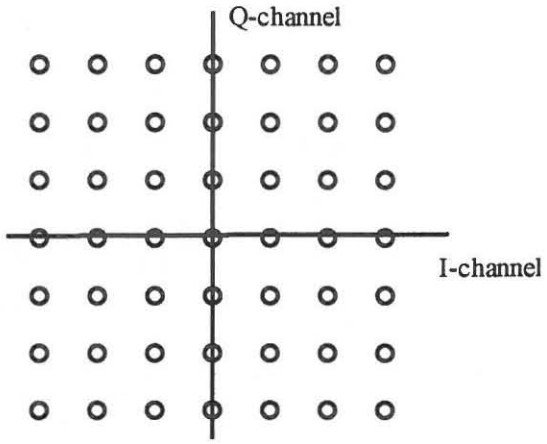


Figure 3.10: 49QPRS signal constellation.

3.7 PERFORMANCE OF PARTIAL RESPONSE SIGNALLING

The probability of error versus SNR of a system is used to express how well a system performs in the presence of noise. This is a standard method and is particularly suitable for comparing the performance of different modulation schemes with one another. Figure 3.11 [3, p. 148] shows the required SNR necessary to obtain a specified bit-error rate on a limited bandwidth transmission channel for an uncoded binary signal and a duobinary signal.

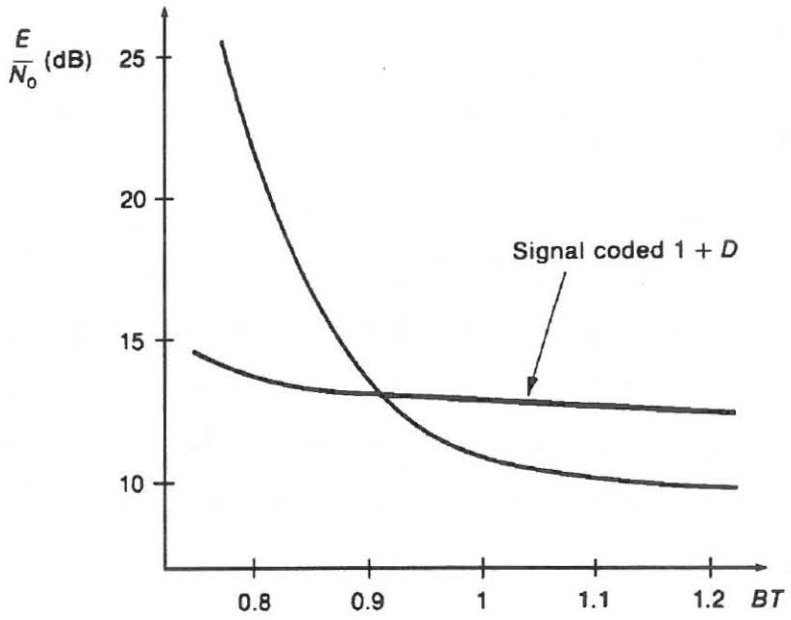


Figure 3.11: SNR required to obtain a bit-error rate of 10^{-4} in a binary and a duobinary coded signal [3, p. 148].

As the bandwidth of the channel diminishes, there is a substantial degradation in transmission quality. However, if the signal is encoded, the degradation is very small.

According to Schwartz [51, p. 223], a roll-off ratio[1] of 1,0 is required for a frequency efficiency of 1 bit/sec/Hz for 4PAM, whereas baseband 7PRS is capable of 4 bits/sec/Hz. Therefore, even if an increase in transmitted power of a 7PRS signal is required, as stated in [9, p. 30], it has the advantage of enabling the use of the more frequency efficient coding scheme. Inspection of Table 3.1 reveals that, for the same bandwidth efficiency, a smaller number of levels are also required for the correlative coding scheme than for memoriless systems.

The decoding of partial response modulation schemes usually requires the use of decision feedback equalisers (DFEs) to remove ISI introduced by the partial response modulation. DFEs could suffer from a loss of performance during certain fading situations. Therefore, partial response schemes have not been very popular [63, p.191].

It is clear that the use of partial response signalling has a number of substantial advantages, but unfortunately also some disadvantages.

3.8 SUMMARY

Partial response signalling is a frequency efficient data communication scheme. Different frequency spectrums can be realised - in order to satisfy specific needs - by using different coding schemes, a number of which are discussed in this chapter. However, these schemes all have nulls at submultiples of the data frequency. This tends to cause problems in clock synchronisation and techniques commonly used for this purpose - such as raising to a power - might not be successful. A price has also to be paid in SNR for realising the desirable spectral-shaping characteristics of PRS

[1] See paragraph 2.3.1

signalling, since the energy required to achieve the same probability of error as binary antipodal signalling is 2,1 dB greater [17, p. 273].

Error propagation in partial response signalling is prevented by precoding. However, the positive effect of this, is questioned by some experts.

The PRS signal is received as a continuously varying voltage and a decision has to be made regarding its level during each symbol interval. Certain definite decision levels are defined for this purpose. These levels are not necessarily separated by an equal amount, but may differ from one level to the next, depending on the probability of occurrence of the specific level.

PRS signals are normally modulated onto a carrier before transmission. 49QPRS, that use two seven-level amplitude modulated PRS signals in quadrature, has a frequency efficiency of 4 bits/sec/Hz.

Even though the use of PRS has not been particularly widespread, it has a number of very desirable qualities and is still regarded as a practical way of realising the maximum possible theoretical frequency in a data communications system. Because of these qualities, researchers are still investigating new ways of utilising this coding technique.

CHAPTER 4

CARRIER SYNCHRONISATION

Since a carrier frequency recovery technique for a hypothetical 49QPRS data communication system was developed during execution of the project, the specifications of such a system are defined before a number of typical recovery systems are described. Both closed- and open-loop techniques receive attention. However, special attention is given to a process of carrier frequency recovery by means of the combined recovery of a pilot tone and repeated calculation of the correlation between the recovered signal and a locally generated signal.

4.1 INTRODUCTION

When data is transmitted from one location to another, the receiver must be capable of reconstructing the time base of the transmitter in order to be able to convert the received signal into a sequence of data symbols [39, p. xiii]. Two levels of synchronisation are investigated for the purpose of the project:

- **Carrier synchronisation:** The phase and frequency of the receiver carrier oscillator should be aligned with that of the transmitter if the bit stream is modulated onto a carrier. This process is discussed in this chapter.

- **Clock synchronisation:** Since the clock of the receiver controls the sampling of the input signal, it should be aligned with that of the transmitter - running at the symbol rate. This process is discussed in Chapter 5.

Lee and Messerschmitt [36, p. 524] emphasise a major difference in the operational characteristics of a clock and carrier recovery circuit. In timing recovery, the frequency of the recovered signal should be equal to the average symbol rate of the input, but transient variations in symbol rate should be ignored - as should noise or any other interference. Any fluctuations in the symbol timing rate can be assumed to be a consequence of interference such as noise. In carrier recovery, by contrast, the phase of the carrier on the input signal should be tracked as closely as possible. Unlike timing phase, several causes can introduce fluctuations in the carrier phase and frequency. To demodulate the signal successfully, these fluctuations should be replicated on the locally generated carrier. From this it is obvious that the ideal operation of clock and carrier recovery systems differ considerably.

Carrier and clock synchronisation techniques will be considered in turn. Some important aspects of the acquisition of synchronisation are discussed in paragraph 4.3.5.2. Since the development and assessment of synchronisation techniques for a hypothetical 49QPRS signal is the main aim of the project, the specifications of the system under consideration are defined.

4.2 SPECIFICATIONS OF 49QPRS SYSTEM UNDER CONSIDERATION

The following are the main characteristics of a 49QPRS system defined in order to study possible synchronisation techniques for partial response systems:

Coding: 7PRS using coders and precoders.

Pulse-shaping filter: In order to limit the amount of processing required at the receiver, this function was not split between the transmitter and the receiver, but was done by only one filter - at the transmitter.[1] The impulse characteristic of the filter is shown in Expression 2.12.

Carrier frequency: 3 600 Hz.

Carrier recovery technique: Pilot tone at 1 200 Hz with correlation loop.

Data rate: 9 600 Bits per second.

Symbol period: 416.67 microsecond.

Clock frequency recovery technique: Absolute value rectifier with limited pre-processing.

Typical carrier recovery techniques are considered, followed by a description of the actual technique used in the project.

[1] See Paragraph 2.6 for the optimum characteristics and position of the pulse-shaping filter.

4.3 CARRIER SYNCHRONISATION

Since the modulation scheme under consideration produces a double-sideband, suppressed-carrier (DSBSC) signal, the regeneration of a local carrier signal by the receiver is required. The recovery of the modulating signal from the modulated signal is called synchronous detection, or coherent detection. For coherent detection, there is no difference in performance between the baseband system and the AM system. Incoherent forms of demodulation lead to poorer performance than that of coherent detectors [49, p. 339].

Coherent detection has the unique feature that the noise component of the output message appears additively with the message, irrespective of the input signal-to-noise ratio [21, p. 497]. This distinguishes coherent detection from all other demodulation techniques.

In order to recover the modulated signal, there should be some knowledge of the relative phase of carrier used for modulation. In fact, both the correct phase and frequency must be known to recover the modulating signal successfully. Stremler [57, p. 223] describes this phenomenon as follows:

Assume that there is a small frequency error, $\Delta\omega$, and a phase error, θ_0, in the locally generated carrier signal at the receiver. During demodulation, the receiver then forms the product,

$$A(t)\cos[(\omega_c + \Delta\omega)t + \theta_o] = h(t)\cos\omega_c t\cos[(\omega_c + \Delta\omega)t + \theta_o]$$

$$= \tfrac{1}{2}h(t)\cos[(\Delta\omega)t + \theta_o] + \tfrac{1}{2}h(t)\cos[(2\omega_c + \Delta\omega)t + \theta_o] \qquad (4.1)$$

where ω_c = carrier frequency

$\Delta\omega$ = frequency error

θ_o = phase error

$h(t)$ = modulating signal

The second term on the right hand side of Expression 4.1 is centred at twice the carrier frequency and can be filtered out by a lowpass filter. The output of this filter is then:

$$e_o = \tfrac{1}{2}h(t)\cos[(\Delta\omega)t + \theta_o] \qquad (4.2)$$

The output is therefore equal to $\tfrac{1}{2}h(t)$ - the modulating signal - only if both $\Delta\omega$ and θ_o are zero. Now consider the case where there is no frequency error, so that there is only phase error. The filter output now reduces to:

$$e_o(t) = \tfrac{1}{2}h(t)\cos\theta_o \qquad (4.3)$$

The phase error in the local oscillator causes a variable gain factor in the output signal that is proportional to the cosine of the phase error. For small, fixed phase errors, this is tolerable, but for errors approaching ±90° the received signal disappears. A randomly, varying phase error results in unacceptable performance.

In closed loop carrier recovery systems, the signal due to the phase error is used to adjust the phase of the locally generated signal and zero phase error indicates generation of a signal of the correct frequency and phase. However, the error signal vanishes not only for zero error, but also for $\theta_o = \pm\pi$. This points to a problem inherent in most tracking systems [39, p. 12].

If there is only frequency error, Expression 4.2 simplifies to:

$$e_o(t) = \tfrac{1}{2}h(t)\cos(\Delta\omega)t \tag{4.4}$$

Instead of recovering the original signal, the output consists of the product of the modulation signal multiplied by a low frequency sinusoid. This results in undesirable distortion.

From the above it is obvious that ideally both the frequency and phase of the locally generated carrier signal should be identical to that of the transmitted signal. The function of the carrier recovery circuit can therefore be described as follows [39, p. 7]:

- Generating a local frequency of ω_c.
- Estimating the phase difference between the outputs of the transmitter carrier frequency generator and the local carrier frequency generator.
- Adjusting the phase of the receiver oscillator to correspond with that of the transmitter.

CCITT recommendations traditionally call for an ability to deal with a user-to-user frequency shift of 7 hertz, but such shifts can only be generated by very old analogue carrier systems. A 1982/1983 survey of the USA telephone network revealed a shift of two hertz on only 0.2% of all connections. All carrier recovery loops are well able to follow this [4, p. 31]. Luise and Reggiannini [37, p. 1169] refer to a preferable oscillator stability better than 10^{-7}.

A number of fast-converging methods for the solution of the carrier frequency recovery problem have been presented in the literature. Among these techniques, the use of a periodic sequence of training bits to achieve an accurate estimate of the frequency at which the input spectrum peaks, is mentioned [37, p. 1169]. Recovery of both carrier and bit timing in a PSK system is often aided by a preamble to the data signal that contains an alternating bit pattern of known phase [56, p. 507]. However, alternative schemes do not require the presence of a training sequence.

The following techniques are typically used for carrier recovery:

- Phase-locked loops
- Costas loops (a type of phase-locked loop)
- Squaring loops and fourth-power devices
- Remodulation
- Transmission of a reinserted carrier or pilot tone

Since a variation of the last type referred to above was implemented in this project, only brief descriptions of the operation of the other types are given.

4.3.1 PHASE-LOCKED LOOPS

A phase-locked loop (PLL) basically consists of three elements, viz. a phase comparator, a lowpass filter and a voltage-controlled oscillator. These devices are connected as shown in Fig. 4.1.

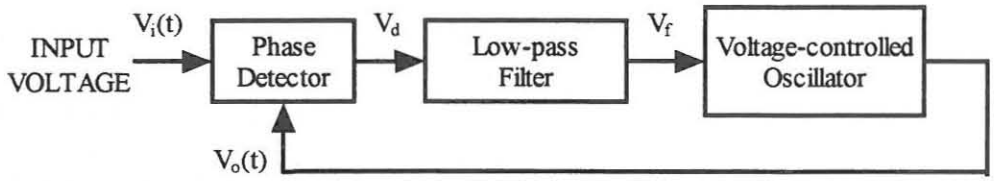


Figure 4.1: Block diagram of basic phase-locked loop.

Killen [30, p. 50] describes the operation of a PLL as follows:

Assume a sinusoidal input voltage described by:

$$V_i(t) = A_1 \sin(\omega_i t + \theta_i) \tag{4.5}$$

where ω_i = frequency of input voltage

θ_i = relative phase of input voltage

The output of the voltage-controlled oscillator (VCO) is:

$$V_o(t) = A_2 \cos(\omega_o t + \theta_o) \tag{4.6}$$

where ω_o = frequency of VCO output

θ_o = relative phase of VCO output

The phase comparator may be modelled as a four-quadrant multiplier. Thus the output of the multiplier is:

$$V_d = A_1 A_2 K_m \sin(\omega_i t + \theta_i)\cos(\omega_o t + \theta_o) \tag{4.7}$$

where K_m is the multiplier gain.

If Expression 4.7 is expanded, it results in an expression that contains sums and differences of the input frequency and the local oscillator frequency. The sum-frequency term is filtered by the filter and its output can be written as:

$$V_f = \frac{A_1 A_2}{4} K_m \sin(\omega_i t - \omega_o t + \theta_i - \theta_o) \tag{4.8}$$

If the PLL is locked to the input signal, $\omega_i = \omega_o$. Expession 4.8 reduces to:

$$V_f = \frac{A_1 A_2}{4} K_m \sin(\theta_i - \theta_o) \tag{4.9}$$

where $\theta_i - \theta_o$ is the phase error.

Expression 4.9 describes the error voltage that is applied to the VCO to maintain lock. This is shown in Figure 4.2 [30, p. 175]. With no phase error present, the error voltage decreases to zero.

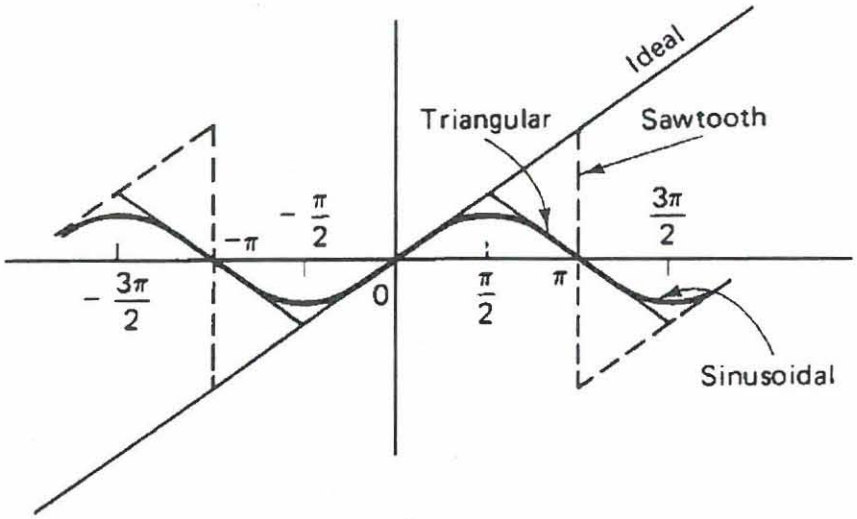


Figure 4.2: Typical phase detector characteristics [30, p. 175].

If, due to a disturbance in the input signal, the error exceeds $\frac{\pi}{2}$, the PLL will skip one or more cycles.

4.3.2 COSTAS LOOPS

Synchronous receiving systems often use costas loops for demodulation of DSBSC signals. Figure 4.3 shows the block diagram of a costas loop.

A costas loop consists of two coherent detectors supplied with the same input signal. Roden [49, pp. 337-338] describes the operation of the costas loop as follows:

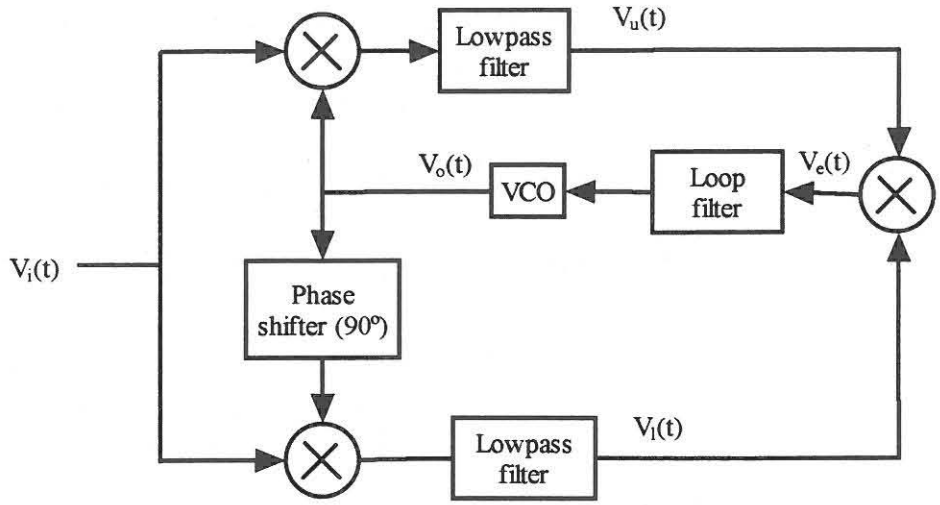


Figure 4.3: Block diagram of costas loop.

Assume an input of:

$$V_{i(t)} = A(t)\cos(2\pi f_c t + \theta_i) \tag{4.10}$$

where θ_i = phase shift of the transmitter oscillator relative to the phase of a hypothetical observer.

With a VCO output of:

$$V_o(t) = \sin(2\pi f_c t + \theta_o) \tag{4.11}$$

where θ_o = relative phase of the local oscillator

the output of the upper lowpass filter is equal to:

$$V_u(t) = \tfrac{1}{2} A(t)\sin(\theta_i - \theta_o) \tag{4.12}$$

This output is therefore proportional to the sine of the phase difference. This is only true if the frequency of the local oscillator is equal to that of the transmitter. If the two frequencies are not equal, the phase difference includes a linearly increasing term. In the same manner, the output of the lower lowpass filter can be given by:

$$V_I = \tfrac{1}{2}\cos(\theta_i - \theta_o) \tag{4.13}$$

The output is therefore proportional to the cosine of the phase difference. These two signals are multiplied together and the result is the error term:

$$\begin{aligned} V_e(t) &= \tfrac{1}{4}A^2(t)\sin(\theta_i - \theta_o)\cos(\theta_i - \theta_o) \\ &= \tfrac{1}{4}A^2(t)\sin[2(\theta_i - \theta_o)] \end{aligned} \tag{4.14}$$

The loop filter extracts the average value of $A^2(t)$. The error term is therefore proportional to the sine of twice the phase difference, and the loop drives this term toward zero. However, it should be noted that the loop drives not only to the correct phase, but also to the correct frequency.

In the case of a quadrature system, if the carrier has the wrong phase, interference between the channels can be expected in addition to loss of desired signal power [56, p. 257]. Channel I would give:

$$V_I(t) = V_I(t)\cos(\theta_i - \theta_o) - V_Q(t)\sin(\theta_i - \theta_o) \tag{4.15}$$

where $\cos(\theta_i - \theta_o)$ indicates loss of power and the second term indicates interference by the Q channel.

A similar situation occurs for Channel Q and the output is:

$$V_Q(t) = V_I(t)\sin(\theta_i - \theta_o) + V_Q(t)\cos(\theta_i - \theta_o) \qquad (4.16)$$

The performance of costas loops can be optimised by using integrate-and-dump filters. The disadvantage of this is the need for symbol timing of the integrators [43, p. 261].

4.3.3 SQUARING LOOPS

Squaring loops are typically used in quadrature amplitude modulation (QAM) applications where the acquisition time of PLLs are too long and if more than 73% of transmitted power is due to one of the quadrature components [43, p. 261]. The decision as to whether to implement a costas loop or the classical squaring loop design amounts to a design decision between the difficulty of implementing the squaring device and the difficulty of keeping the costas loop arm filters nearly matched [55, p. 447]. According to Tomasi [65, p. 44], conventional squaring loops are less able than costas loops to accurately recover a carrier from a poor-quality received signal. Figure 4.4 [19, p. 134] shows the block diagram of a squaring circuit.

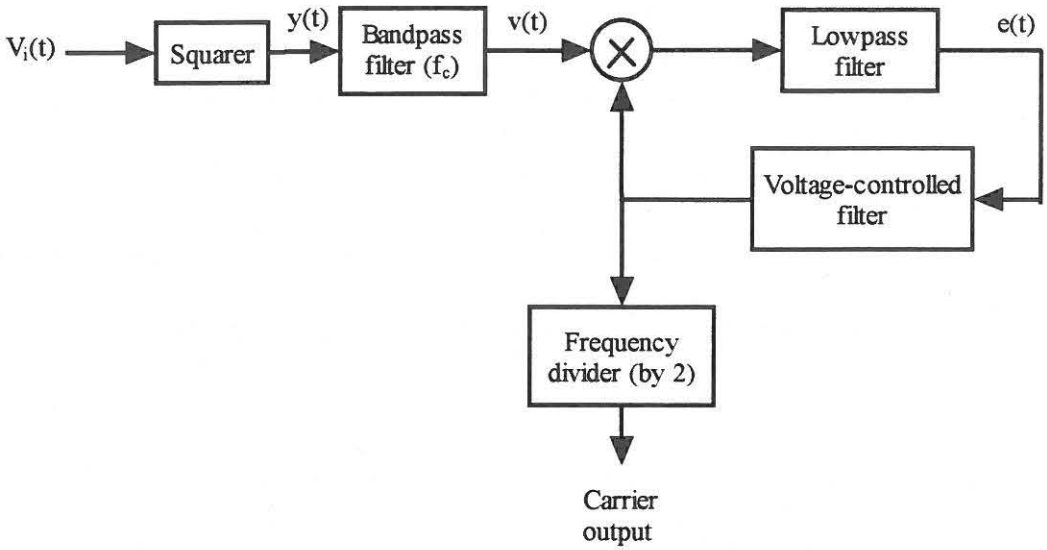


Figure 4.4: Block diagram of a squaring loop used for carrier recovery.

The operation of this circuit can be described as follows:

The DSBSC input signal is:

$$V_i(t) = A(t)\cos(2\pi f_c t) \tag{4.17}$$

This signal is squared, giving:

$$y(t) = A^2(t)\cos^2(2\pi f_c t)$$

$$= \frac{A^2(t)}{2}[1+\cos(4\pi f_c t)] \tag{4.18}$$

This signal is passed through a narrow-band filter centred around $2f_c$. This signal is used to drive a PLL at twice the required output frequency. The output frequency of the PLL is divided by two, to give the required output.

It is also possible to use the output of the frequency divider directly without using a PLL type circuit to track the required frequency. However, since the squaring circuit tends to increase the noise near the desired double-frequency carrier component [57, p. 623], a very narrow bandpass filter is required if a loop is not used.

As a result of the frequency division, there is a phase ambiguity of π radians. If a signal of the incorrect phase is used, the polarity of the demodulated signal will be inverted. In the case of data communications, this can have serious consequences and provision should be made to prevent this from happening.

When less than 73% of the total power is in the largest power quadratic component in QAM, modified versions of the raising-to-a-power technique are required. Modifications involve replacing squarers by fourth-power devices, divide and multiply by two networks by divide and multiply by four devices, and centring filters at $4f_c$ [43, p. 261].

Alternatively, a technique of remodulation can be utilised.

4.3.4 REMODULATION

A remodulator can also be used for carrier recovery [43, p. 261]. The received signal is first demodulated using quadrature versions of the local carrier. The two message outputs are each lowpass filtered, symmetrically limited, and used to multiply

(remodulate) quadrature versions of the input signal, after being delayed slightly to compensate for network delays. A summation of the product outputs (with appropriate sign) is bandpass filtered and limited to remove amplitude fluctuations. The final signal is the local carrier that was used at the beginning of the process, as described. The remodulation technique does not contain a phase ambiguity as found in many other carrier recovery techniques.

4.3.5 PILOT CARRIER SYNCHRONISATION

A residual carrier signal is available for synchronisation purposes in a double-sideband amplitude-modulated system. Isolation of the unmodulated carrier from the composite frequency spectrum implies the use of, inter alia, narrow-band filtering. However, with DSBSC signals, a technique commonly used to enable carrier synchronisation, the addition of a pilot carrier signal is used. If the receiver oscillator is phase-locked to the incoming pilot tone, and the pilot is lost for some reason, the system should continue to operate satisfactorily for a limited period of time [54, p. 100]. When there is no separate pilot, the signal is said to be self-synchronising. Self-synchronisation is not practical as a rule, unless the symbol SNR exceeds about 6 dB [56, p. 504].

Due to the following reasons it was decided to implement carrier synchronisation using a type of pilot tone technique:

- Ease of implementation in a discrete-time system.

- A desire to evaluate the execution time and general performance of a correlation calculating technique of maintaining synchronisation and obtaining an undistorted carrier wave.
- The suitability of the frequency spectrum of the 49QPRS signal under consideration. Figure 4.6 shows spectral nulls at, *inter alia*, 2400 Hz and 1200 Hz. This corresponds to the expected response as derived from Figure 3.2. For this reason a pilot tone was transmitted at 1200 Hz.

4.3.5.1 PRINCIPLES OF OPERATION

The pilot tone is generally sent outside the passband of the modulated signal so that it will not interfere with the modulated signal [57, p. 233]. Early modems that employed vestigial-sideband modulation transmitted a carrier tone in phase quadrature [16, p. 221]. Two pilot tones, at the edges of the data spectrum, could also be transmitted. The received signal can then be represented as [16, p. 221]:

$$\begin{aligned} s(t) &= m(t)\cos[\omega_c t + \Delta\omega t + \theta(t) + \phi] + \hat{m}(t)\sin[\omega_c t + \Delta\omega t + \theta(t) + \phi] \\ &+\alpha\cos[\omega_L t + \Delta\omega t + \theta(t)] + \alpha\cos[\omega_H t + \Delta\omega t + \theta(t)] \end{aligned} \quad (4.19)$$

where $\Delta\omega$ = frequency distortion

$\theta(t)$ = phase jitter

ω_H = upper pilot tone

ω_L = lower pilot tone

The pilot tone components are multiplied together and lowpass filtered to yield a cosine term of frequency $\omega_H - \omega_L$. This term is used to drive a phase-locked loop.

In the system under consideration, a number of frequencies contain virtually no signal power, and the pilot tone was selected so that it fitted into one of these spectral gaps. Figure 4.5 shows a block diagram of the DSP carrier recovery technique developed in the project.

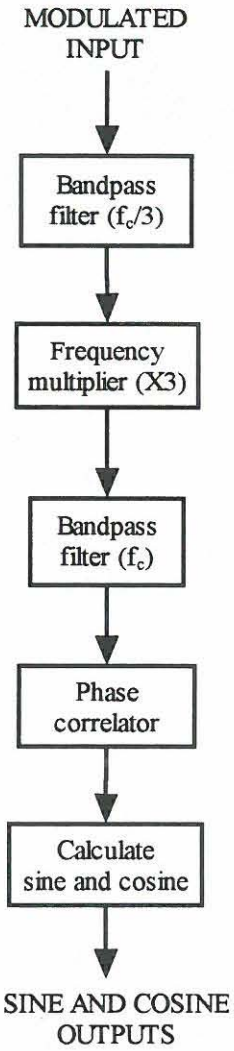


FIGURE 4.5: Flowchart of a DSP pilot tone carrier recovery scheme.

At the transmitter two 7PRS signals are modulated in quadrature - to give a 49QPRS constellation - at a carrier frequency of 3 600 Hz. As shown in Figure 4.6, a spectral null occurs at 1 200 Hz. This frequency is obviously suitable as a pilot frequency.

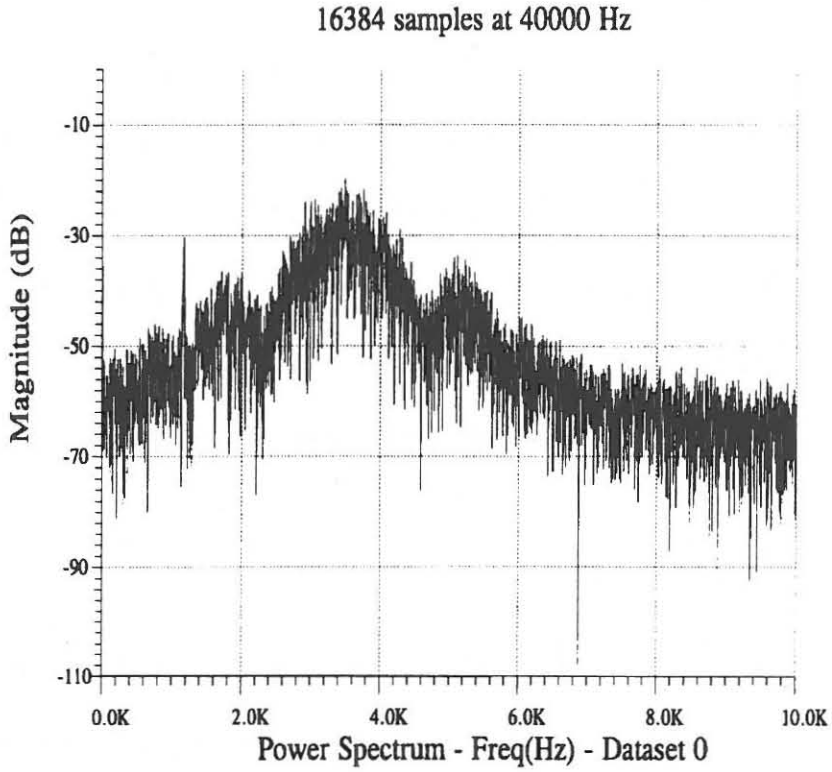


Figure 4.6: Frequency spectrum of 49QPRS system as developed.

The operation of the circuit can be described as follows:

The pilot tone is extracted from the modulated signal by bandpass filtering at 1200 Hz. Using downsampling [13, p. 13], this frequency is multiplied by three and again bandpass filtered - at 3600 Hz. The output is at the correct frequency, but since the pilot tone might have been degraded by noise on the transmission channel, it may also contain a substantial amount of noise. In order to obtain a noise-free signal, as well as

the cosine of the sine wave recovered thus far, this signal is phase correlated to a built-in sine table in the DSP processor.

This correlation process takes place every eleventh sample. At a sampling rate of 19 200 Hz, this corresponds to a correlation/phase adjustment once every 2.0625 cycles of the 3 600 Hz carrier wave. The correlation value is calculated over a period of the previous twelve values of the regenerated carrier wave. The position on the sinetable, corresponding to the maximum correlation value, is determined and used as reference for the duration of the next eleven samples. Sine and cosine values are read from the built-in sinetable for demodulation of the received 49QPRS signal during this time.

Since the sinetable has 256 values, the phase adjustment is done with a resolution of 1.40625 degrees. It is appreciated that a correlation length of slightly more than two cycles is substantially less than the generally accepted minimum value of five cycles [24, pp. 195-196]. A compromise had to be arrived at regarding the correlation length between the better accuracy of a longer length, and the additional processing time required by longer sequences. However, considering that the aim of the correlation procedure is not to determine the relative wave shapes of two signals, but only to determine the relative phase of the recovered signal, the correlation length as specified, was decided on. This decision proved to be correct during the eventual evaluation of the process. The process of cross-correlation between the recovered signal, $g_r(t)$, and the signal, $g_l(t)$, read from the sinetable, can be described as follows [19, p. 60]:

$$R_{12}(\tau) = \frac{1}{T_c} \int_{-T_c/2}^{T_c/2} g_i(t) g_l^*(t-\tau) dt \qquad (4.20)$$

where T_c = the period of the recovered frequency.

The resulting cross-correlation value is also periodic, with a period equal to T_c. This is shown in Figure 4.7. Here the cross-correlation between two periodic signals with the same frequency is plotted at different phase angles between the signals.

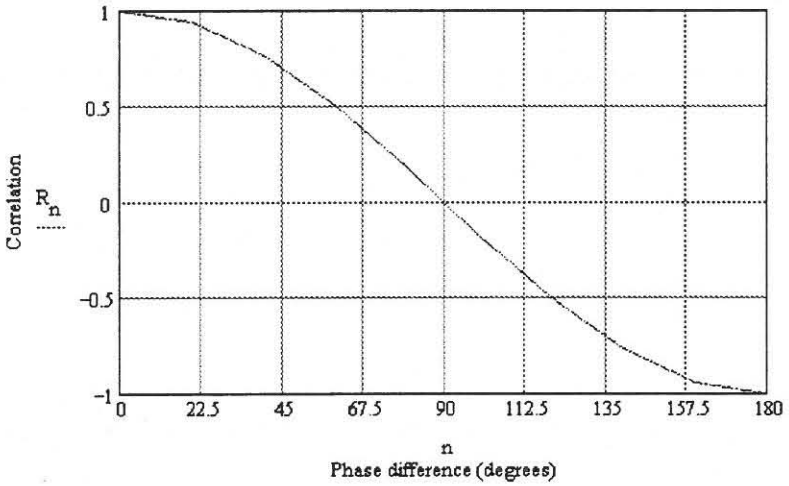


Figure 4.7: Cross-correlation between two periodic signals at different relative phase angles.

Due to the accuracy and stability of the crystal-controlled oscillators on both the transmitter and receiver DSP boards, the process as described, proved to be sufficient and very small phase adjustments were required to keep the two oscillators in phase[2].

[2] See Paragraph 8.4.1 for typical values.

4.3.5.2 ACQUISITION CHARACTERISTICS

Since loops using self-acquisition can be guaranteed to acquire synchronisation in a reasonable time only under very benign circumstances [55, p. 450], acquisition is often carried out with the aid of special predetermined transmitted sequences [17, p. 405]. The time required for loop acquisition will vary widely as a function of the initial phase error [55, p. 450].

A basic characteristic of the system under discussion is that for a maximum period of

$$12 \times \frac{1}{f_s} = 625\mu s$$

(or 2.0625 cycles of the carrier wave) after the beginning of reception, the transmitting and receiving oscillators run unsynchronised. At that stage, the correlation between the oscillators is calculated for the first time. This typically results in a large initial phase adjustment, followed by small (or no) adjustments at subsequent points of calculation. This is totally acceptable, considering that a synchronisation acquisition time of

$$N = 0.7\frac{f_c}{B} \text{ cycles} \qquad (4.20)$$

where B is the -3 dB bandwidth of the carrier bandpass filter

or alternatively, for the system under consideration

$$t = N\frac{1}{f_c}$$

$$= \frac{0.7 \times 3600}{200} \times 3600$$

$$= 3.5ms$$

is considered as typical [30, p. 155]. Figure 4.8 shows the acquisition and tracking response of the carrier recovery technique under discussion. Note that the phase is adjusted every eleventh sample. Also note the big initial adjustment. Although the y-axis of the figure is limited to show a maximum phase adjustment of 45 degrees, in actual fact the initial adjustment in the case shown was 168 degrees. Similar values for the initial adjustment are shown in Chapter 8.

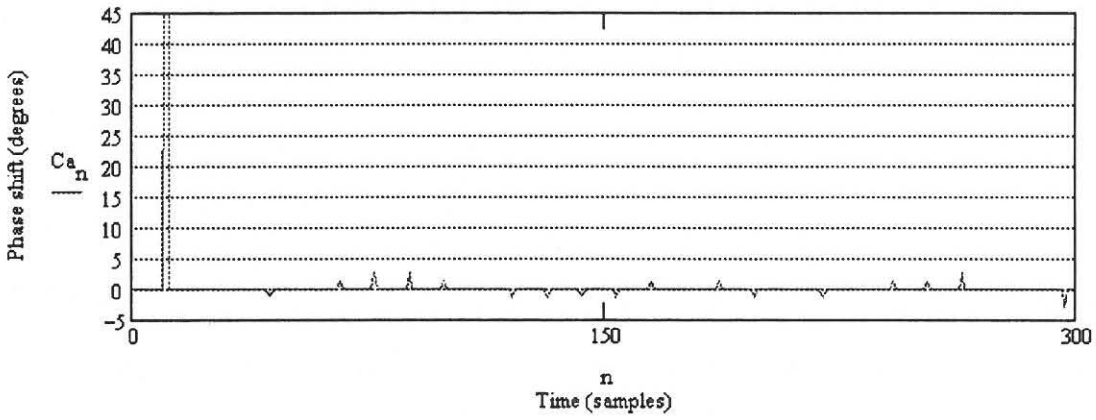


Figure 4.8: Typical acquisition and tracking response of carrier recovery technique.

From the response shown it is also clear that, once the local oscillator has been locked, relatively small phase adjustments are required to maintain synchronisation.

4.3.5.3 PROBLEMS ASSOCIATED WITH PILOT CARRIER SYSTEMS

Kobayashi [33, p. 268] identifies the following problems associated with a pilot carrier system:

- Phase jitter of pilot carrier, caused by data components near the carrier frequency contributes to errors in the phase estimate and results in distortion of the demodulated signal. (Since the pilot is positioned in a spectral null of the modulated 49QPRS signal, this effect is limited.)
- The phase characteristic of the transmission media tends to be highly non-linear near the cut-off frequency region at which the pilot carrier is typically located. (Since the pilot is located at 1 200 Hz, while the signal spectrum extends beyond this frequency in both directions, this problem is limited.)

Sklar [55, p. 451] states that if conditions are such that the phase variance is large, increasing the data signal-to-Gausian-noise ratio may not be effective in reducing the detected error probability. It should be noted here that the presence of an irreducible error in these situations is a characteristic of residual carrier designs with constant loop SNR. Suppressed carrier tracking loops do not tend to have irreducible errors, because an increase in the data SNR will increase the SNR of the suppressed carrier tracking loop, reducing tracking error.

A potentially serious problem associated mainly with suppressed carrier loops - therefore less common with systems using pilot tones - is that of false lock. This can be a problem, especially during acquisition of carrier phase. The interaction of the data

stream with the loop non-linearities and loop filters will produce sidebands in the spectrum that is input to the phase detector. These sidebands can contain stable frequency components. Care must be taken that these stable components are not allowed to capture the tracking loop. If the loop is captured, it will appear to be operating correctly; the VCO control signal will be small but the VCO output will be offset in frequency from the correct carrier component. This is false lock. False lock is a hardware implementation problem that typically sets an effective lower limit on the bandwidth of the loop filters. Residual carrier loops, having fewer non-linear elements are not usually bothered by false locking [55, p. 446].

4.4 SUMMARY

A number of different open- and closed-loop carrier recovery techniques were discussed. Hagiwara and Nakagawa [18, p. 1882] identify the following characteristics as highly desirable for any carrier tracking system:

1. Fast acquisition.
2. Wide pull-in range.
3. Good noise suppression.

Modern DSP processors are particularly suited for calculation of the cross-correlation of data sequences and are frequently used for this purpose in signal processing environments - as described by Sheers [53, p. 31] and Scott and Olasz [52, p. 2263]. A process of correlation to phase synchronise the local oscillator of a receiver with that of the transmitter, on condition that the free-running frequency of the two oscillators is

similar, was therefore developed to phase synchronise the local oscillator of a data receiver with that of a transmitter. This technique appears to fulfil the first and third requirement as specified above. It was implemented in DSP and evaluated. Its performance is described in Chapter 8.

CHAPTER 5

CLOCK RECOVERY

A clock recovery technique for the hypothetical data transmission system, as specified in Chapter 4, was developed during execution of the project. Therefore, a number of typical timing recovery techniques are discussed in this chapter. Special attention is given to spectral line methods, using non-linearities. Use of a special pre-filter, required to maximise the amplitude of the recovered clock signal, is also described. Finally, the possible use of an absolute value rectifier, used in conjunction with the pre-filter as mentioned above, in a clock recovery system is discussed. The combination of a special bandpass filter[1], the pre-filter[2] and an AVR proved to be a very successful, novel technique of recovering the clock signal.

5.1 INTRODUCTION

Synchronisation requirements imply that a certain minimum density of information signal transitions is required to provide a continuous indication of symbol boundaries. Certain provisions might be required to insert artificial transitions into the transmitted waveforms. The following are short descriptions of four techniques used for ensuring the existence of signal transitions for timing recovery [2, p. 171]:

[1] As described in Paragraph 2.3.2.
[2] As described in Paragraph 5.3.4.1.

1. The data sequence is maintained so that long transition-free sequences do not occur.
2. The transmission system can insert transition bearing bits at fixed positions into the data stream. This would avoid extended transition-free periods during which synchronisation can be lost.
3. As in (2), but only when required to terminate a long string of 0's or 1's.
4. Data are scrambled to prevent repetitive data patterns and to generate pseudo-random data sequences for any given input sequence. This enhances the clock recovery capability of data receivers [16, p. 218]. However, some simple scramblers show a vulnerability to a dropped or added bit. In this event, disastrous error propagation ensues [17, p. 453].

Lathi [35, p. 200] identifies three general methods of synchronisation:

1. Derivation from a primary or a secondary standard. In this case, both the transmitter and the receiver are slaved to a master timing source. This method is suitable for large volumes of data and high-speed communication systems. Because of the high cost, it serves no purpose in the system under consideration.
2. Transmitting a separate synchronising signal (pilot clock).
3. Self-synchronisation, where the timing information is extracted from the received signal itself.

During development of the clock recovery scheme under consideration, the study was limited to self-synchronisation only making use of one demodulated sequence, even though there may be advantages in taking both data sequences into account [3, p. 259]. Therefore, only this type of technique is considered. Lee and Messerschmitt [36, p. 561] distinguish between two types of self-synchronisation timing recovery techniques:

1. Deductive timing recovery: This technique extracts a timing signal directly from the incoming signal. As shown in Figures 5.1(a) and (b), a PLL may or may not form part of such a circuit.
2. Inductive timing recovery: It uses a feedback loop to derive a timing signal, and not merely to reduce jitter of the signal already recovered. This technique is shown in Figure 5.1(c).

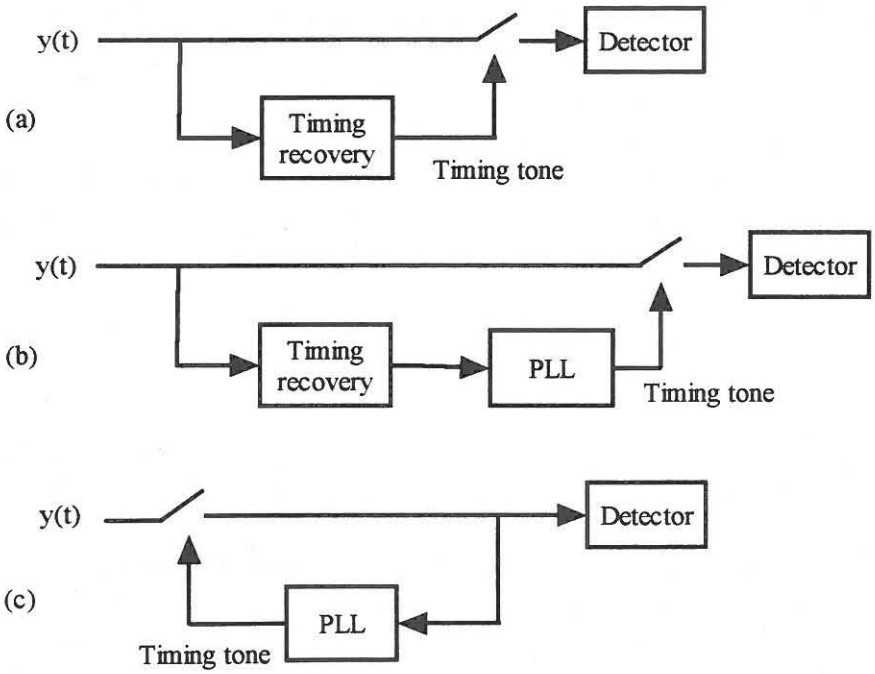


Figure 5.1: Block diagrams of (a) and (b) deductive, and (c) inductive timing recovery schemes.

The following is an adapted version of the description in [39, pp. 3-5] on the operation of clock recovery systems in general:

The transmitted signal can be written as:

$$d(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT - \varepsilon_o T) \tag{5.1}$$

where a_k = the k-th data bit transmitted

T = bit duration

$g(t)$ = filtered signal

ε_o = a constant, unknown time shift of the transmitter time axis relative to the time reference of a hypothetical observer.

In the simplest case, this signal arrives D seconds later - essentially unchanged - at the receiver. D is written as the sum of multiples of T plus a fractional time delay $\varepsilon_c T$. Thus:

$$D = MT + \varepsilon_c T \tag{5.2}$$

Since only the sequence of symbols is of importance, the delay MT is of no concern and the received signal is written as:

$$y(t) = d(t - \varepsilon_c T) + n(t)$$

$$= \sum_k a_k g(t - kT - \varepsilon_T T) + n(t) \qquad (5.3)$$

where $\varepsilon_T = \varepsilon_o + \varepsilon_c$ and $n(t)$ is noise.

The receiver must somehow extract the timing information from the received signal itself.

From Expressions 5.1 and 5.3 it may be concluded that the receiver clock may be used as the time reference if it is assumed that the receiver has a perfect oscillator running at 1/T with a phase ε_R. Only the difference $(\varepsilon_T - \varepsilon_R)$ is of concern. The task of synchronisation can now be defined as:

- Ensuring the correct frequency of the receiver clock by making use of a device such as a non-linearity.
- Estimating the misalignment $(\varepsilon_T - \varepsilon_R)$ between the received signal and the receiver clock.
- Using this estimate to generate a reference time axis in alignment with the received signal.

Meyr and Ascheid [39, pp. 11-12] divide synchronisers into three categories:

1. Error tracking, e.g. PLLs.

2. Maximum seeking and post filtering. The transmitter initially transmits a training sequence of known symbols to the receiver. Assume that the pulse shape, d(t), is known to the receiver. The useful part of the signal,

$$y(t) = s(t, \varepsilon_T) + n(t) \tag{5.4}$$

where $s(t, \varepsilon_T) = \sum a_k d(t - kT - \varepsilon_T T)$

is now known to the receiver, with the exception of the time shift ε_T, which must be estimated during the training period.

3. Non-linearity and post filtering. This alternative was studied in particular during execution of this project.

The following is a description of a number of typical timing recovery techniques.

5.2 APPROXIMATE MAXIMUM-LIKELIHOOD METHODS.

Many timing recovery schemes, approximating *maximum likelihood* timing recovery techniques, have appeared over time. One such method, called the sample-derivative method, is shown in Figure 5.2 [36, p. 576].

This technique will attempt to move the sampling phase until the derivative is zero, which occurs at the peak of the signal.

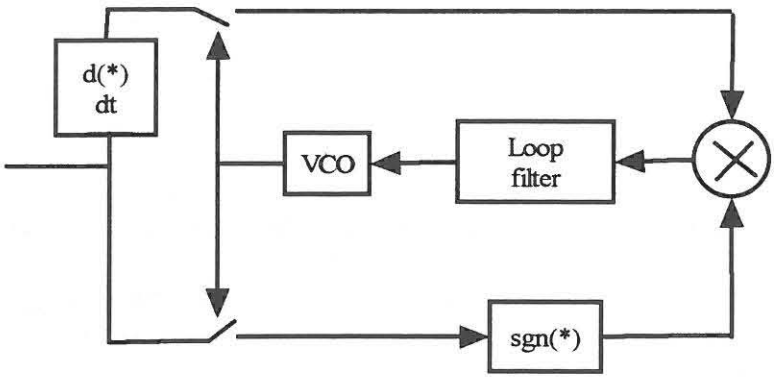


Figure 5.2: Block diagram of sample-derivative timing recovery circuit.

Another, related, closed-loop technique is the early-late gate method. In this type of synchroniser, the input signal is sampled two extra times, once prior to the optimum sampling instant and once after the optimum instant by the same amount. The first of these will tend to include a little of the previous, and the second a little of the next symbol. The sampling instants are adjusted until the two extra samples are equal.

5.3 SPECTRAL LINE METHOD OF TIMING RECOVERY

The spectral line method is often (but not always) suitable for timing recovery. Lee and Messerschmitt [36, pp. 564-565] explain the operating principle of a spectral line method of timing recovery - the most popular method of timing recovery - as follows:

A baseband PAM signal carrying discrete-time digital information,

$$y(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT) \tag{5.5}$$

is cyclostationary, meaning that its moments vary in time and are periodic with a period T, the symbol interval. Consider the new random process

$$Z(t) = f(y(t)) \tag{5.6}$$

where $f(\cdot)$ is a memoriless non-linearity.

The mean-value of $Z(t)$, $E[Z(t)]$ is non-zero and periodic with period T. This mean-value can be regarded as a deterministic component of the random process $Z(t)$, consisting of a fundamental at the baud rate and harmonics. The characteristic can be exploited for timing recovery by forming $Z(t)$ and passing it through a bandpass filter centred at the baud rate. Assuming a random data sequence, the existence of discrete spectral components requires both that the data sequence has a non-zero mean value and that the Fourier transform of the data pulse does not vanish at some multiple of the pulse repetition frequency [14, p. 913].

This is referred to as a linear spectral-line method. In the case of a 7PRS signal, the mean-value is zero and a non-linear spectral-line method is required. In any case, a requirement that the data symbols have a non-zero mean, and that the excess bandwidth be at least 100%, usually rules out the linear spectral line method.

5.3.1 NON-LINEAR SPECTRAL LINE METHOD

A popular method of symbol timing recovery consists of passing the incoming signal, either at an intermediate frequency (IF) or at baseband [41, p. 29], through absolute value rectifiers (AVR), square-law rectifiers (SLR) or fourth-law rectifiers (FLR) in conjunction with bandpass filters to extract the symbol frequency.

It is not necessary to demodulate before timing recovery. By deriving timing directly from a passband signal, timing recovery is completely decoupled from other functions in the receiver. A passband signal can be passed directly through a non-linearity and a tone that can be bandpass filtered will result. This technique is called envelope derived timing [36, p. 570]. However, baseband rectification provides an advantage over intermediate frequency rectification, at least for medium and high signal-to-noise ratios [12, p. 1141].

5.3.2 SQUARE-LAW NON-LINEARITY

Since clock circuits with non-linearities other than square-law devices are hardly tractable mathematically, and their performance is mainly known from simulations [12, p. 1139], the following analysis is based on a hypothetical circuit by making use of a square-law device. Takasaki [61, p. 16] describes this process, shown blockdiagrammatically in Figure 5.3, as follows:

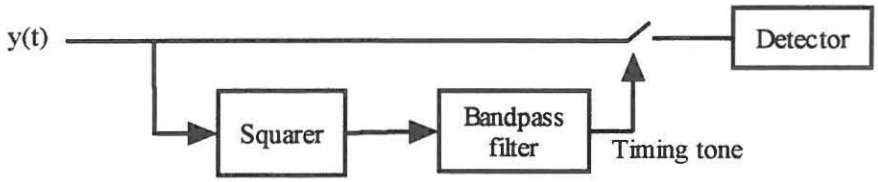


Figure 5.3: Block diagram of a basic non-linear spectral line timing recovery system.

The magnitude (amplitude and phase) of the clock component $X^2(f_o)$, extracted through square-law non-linearity can be expressed by an inner product of two functions.

$$X^2(f_o) = 2\int_0^{f_o/2} \left(|W(f)|\right)^2 \hat{H}(f)df \tag{5.7}$$

where $\left(|W(f)|\right)^2$ = pattern function

$\hat{H}(f)$ = waveform function

The pattern function depends only on the pattern of a pulse sequence, whilst the waveform function depends only on the waveform of a single reshaped pulse on the input of the square-law device.

Assuming a baseband signal as in Expression 5.5, with both a_k and $g(t)$ complex-valued, Lee and Messerschmitt [36, p. 566] describe the operation of the square-law circuit as follows:

The magnitude squared of the process as in Expression 5.5 depends on the correlation function of the data symbols. Assuming they are white - a reasonable assumption for random data -

$$E[a_m a_n *] = \sigma_A^2 \delta_{m-n} \tag{5.8}$$

where a_m and a_n are signal a at times m and n respectively

* indicates correlation

σ_A^2 = variance of A with $\sigma_A^2 = E[|X|^2] - |E[X]|^2$ [36, p. 36]

From this it follows that:

$$E[|x(t)|^2] = \sigma_A^2 \sum_{m=-\infty}^{\infty} |p(t - mT)|^2 \tag{5.9}$$

The expected value is periodic with a period equal to the symbol rate. The fundamental of this signal can be extracted by a bandpass filter. Some of the random component in $|y(t)|^2$ will pass through the filter and result in timing jitter. It is generally desirable to have a larger timing tone. Then random components and noise will contribute relatively less to the jitter.

The behaviour of a clock recovery system using a square-law rectifier (SLR) followed by a resonant circuit depends on the excess bandwidth of the driving pulses in such a way that

performance is satisfactory for medium and large values of α, but it becomes poor as α decreases. In the extreme case of minimum bandwidth Nyquist pulses (with $\alpha = 0$), this method of clock recovery fails [12, p. 1139]. In [14, p. 917] it is stated that bandwidths greater than 50% in excess of the Nyquist bandwidth are unnecessary, but that rms jitter is approximately inversely proportional to the excess bandwidth parameter, α [14, p. 919].

Sklar [55, p. 454] suggests a modified type of square-law non-linearity - as shown in Figure 5.4.

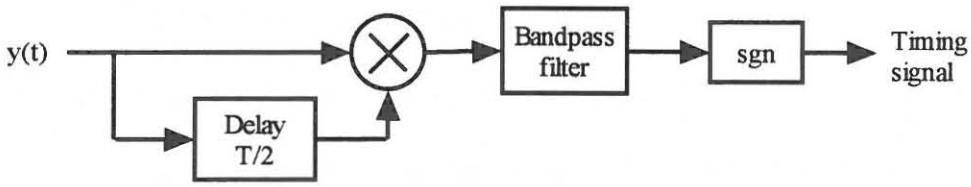


Figure 5.4: Open-loop synchroniser with delay and multiply.

Instead of squaring the incoming baseband signal, the signal is delayed by a half symbol period before multiplying. This delay is the best possible value since it provides the strongest Fourier component [55, p. 455].

5.3.3 ALTERNATIVE TYPES OF NON-LINEARITIES

Even though square-law rectification may be regarded as the "standard" non-linearity to be used in clock recovery systems, it is not the only type of non-linearity used for this

purpose. In fact, in many instances absolute value rectifiers (AVR) or alternatively fourth-law non-linearities (FLR) perform better than SLR.

Widely differing opinions regarding the relative performance of different non-linearities appear in the literature. The following is a short summary of a number of such opinions:

SLR fail with strictly bandlimited pulses. Baseband AVR has the best performance, whereas IF-AVR and FLR are roughly equivalent [12, p. 1141]. Use of an FLR has been suggested in place of an SLR for applications with rolloff as low as 0,12. Under these conditions SLR does not perform well [12, p. 1139]. With duobinary coding, a baseband AVR outperforms a FLR by almost one order of magnitude at large signal-to-noise ratios [12, p. 1142]. For high values of signal-to-noise ratio, AVR gives better performance levels than SLR [3, p. 277]. According to Lee and Messerschmitt [36, p. 568], in discrete-time, a SLR usually has a considerable advantage over both AVR and FLR systems.

In the case of full wave rectification on a binodal signal, the clock component can be enhanced by clipping the upper half of the pulse. It has been shown that a clipping ratio in the neighbourhood of 0,3 maximises the extracted component [61, p. 88]. Ridout describes a timing signal recovery scheme for 3PRS signals, which uses the circuit as shown in Figure 5.5 [48, p. 354].

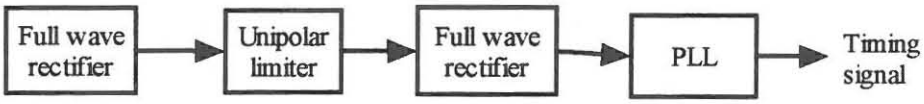


FIGURE 5.5: Block diagram of full wave rectifier clock recovery scheme for 3PRS signals.

The output of this circuit, for a 5 volt peak-to-peak 3PRS input signal, was measured to be approximately 0.06 V r.m.s. [48, p. 355]. Takasaki [60, p. 883] states that full-wave rectification, followed by clipping, approximates square-law rectification for a ternary pulse train.

5.3.4 ABSOLUTE VALUE RECTIFIER WITH LIMITED PRE-PROCESSING

The system investigated for the purpose of this study is a modified version of the normal absolute value rectifier circuit. For ease of reference this circuit will be referred to as a Absolute Value Rectifier with Limited Pre-processing (AVR-LP) system. Figure 5.6 shows a block diagram of the system under discussion.

The basic operation of the circuit is described in the following paragraphs:

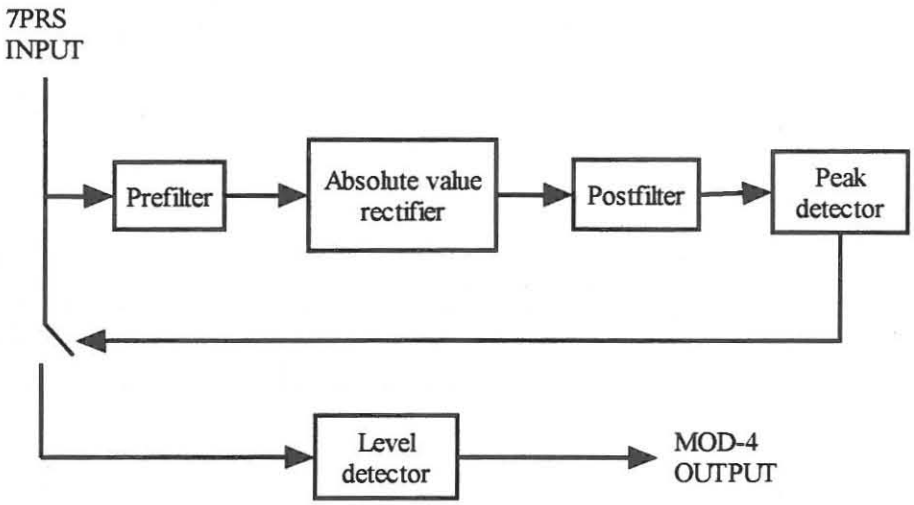


Figure 5.6: Block diagram of AVR-LP clock recovery scheme.

5.3.4.1 PRE-FILTER

The purpose of pre-filtering is to reduce the amount of data dependent jitter of the recovered clock pulse [11, p. 1297 and 52, p. 2263] and to shape the received baseband signal to emphasise frequency components above $\frac{1}{2T}$ and attenuate components below $\frac{1}{2T}$. It is therefore normally a high pass filter with cut-off frequency near $f = \frac{1}{2T}$ [66, p. 18]. Pre-filtering has the added benefit of rejecting the low frequency components of the additive receiver noise that enters the timing path [14, p. 917]. The recovered clock signal normally tends not to have a constant voltage, but rather to vary in voltage. This variation often causes jitter and should preferably be limited as far as possible. According to Takasaki pre-filtering also reduces this variation in amplitude of the recovered clock signal. This effect is shown in Table 5.1 [61, p. 114]. The variation ratio, as referred to in

Table 5.1, is a ratio of the maximum to the minimum peak-to-peak voltage of the recovered signal.

Table 5.1: Reduction in variation ratio through pre-filtering [61, p. 114].

Type of filter	Variation ratio
None	16
Single tap transversal[3]	7.6
Optimum	2.3

In this case, a simple single tap transversal filter was used. Figure 5.7 shows a block diagram of this filter.

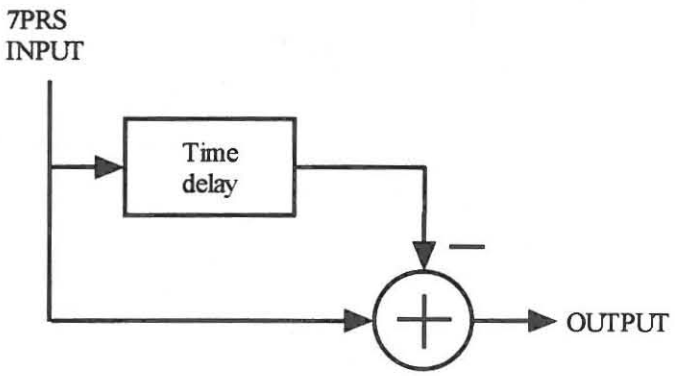


Figure 5.7: Single tap transversal pre-filter.

It was found that the duration of the time delay shown in Figure 5.7 is of critical importance in maximising the eventual amplitude of the recovered clock frequency signal.

[3] Pre-filtering by this type of filter may introduce jitter through other non-linearities.

For this reason, the complete operation of the clock recovery scheme was simulated in MathCAD[4] for different values of time delay.

5.3.4.2 SIMULATION OF AVR-LP TECHNIQUE

Only some results of the MathCAD simulations are shown.[5] From these simulations it became obvious that the duration of the time delay as shown in Figure 5.7 is of importance in realising the maximum possible magnitude of the required frequency component. Figure 5.8 shows the output frequency spectrum with a delay equal to $\frac{T}{2}$. In considering this spectrum, it is essential not to pay too much attention to the actual value of the required spectral line, but rather to compare this line with the immediately surrounding spectral lines. This characteristic determines whether it is possible to recover this signal without substantial noise due to adjacent frequencies in the passband filter.

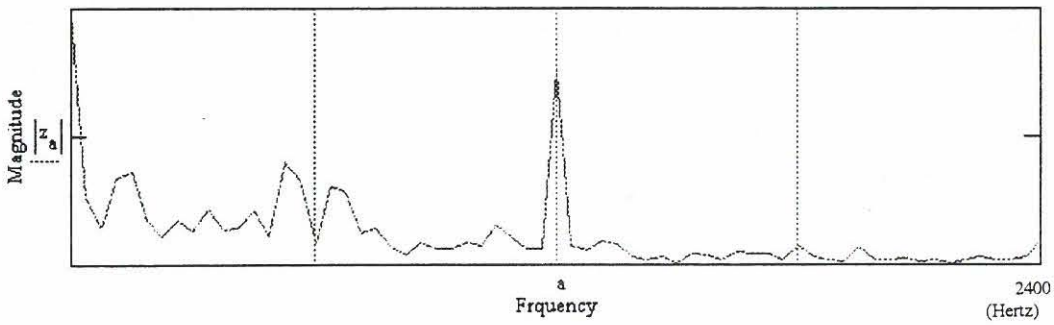


Figure 5.8: Output frequency spectrum with a delay of $\frac{T}{2}$.

[4] MathCAD is owned by MathSoft, Inc. and the programme is copyrighted, with all rights reserved by MathSoft.
[5] Only some results of the simulation are shown in the text. The simulation is shown in more detail in Appendix B.

Figure 5.8 shows the frequency spectrum between direct current and 2 400 Hz, while the communication system as specified in paragraph 4.2 has a symbol frequency of 1 200 Hz - thus equal to the strong component directly above the a in Figure 5.8. The relatively strong spectral line at the required frequency with a delay of $\frac{T}{2}$ is to be expected if Figure 5.9(a) to (d) is considered. Here, a short sequence of the 7PRS signal is plotted in such a way that the signal at time n is plotted against the same signal at time (n-x), for x equal to $\frac{T}{8}$, $\frac{T}{4}$, $\frac{3T}{8}$ and $\frac{T}{2}$ in figures (a) to (d). It is obvious that with x equal to half the symbol period, the transfer of the output from one level to another is accentuated. From these plots the following conclusions can be drawn:

- The output of the pre-filter with a delay time (x) equal to half the symbol period, is maximised with respect to the clock frequency.
- This time delay is not critical and any delay of approximately this duration should give a satisfactory output.

5.3.4.3 ABSOLUTE VALUE RECTIFIER OUTPUT AND POST-FILTERING

The output of the AVR is shown in Figure 5.10. The delay period used in this case was equal to $\frac{T}{2}$. Figure 5.8 shows the frequency content of the signal in Figure 5.10. The output of the AVR is filtered with a bandpass filter centred at the symbol rate, with the output as shown in Figure 5.11.

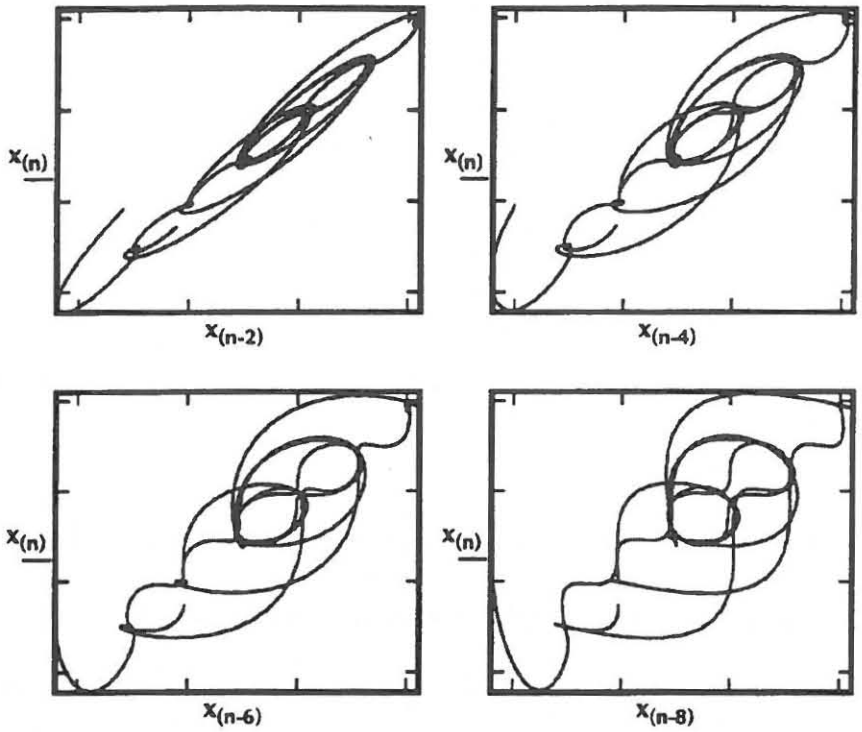


Figure 5.9: Plots of x(n) against (a) x(n-$\frac{T}{8}$), (b) x(n-$\frac{T}{4}$), (c) x(n-$\frac{3T}{8}$), (d) x(n-$\frac{T}{2}$).

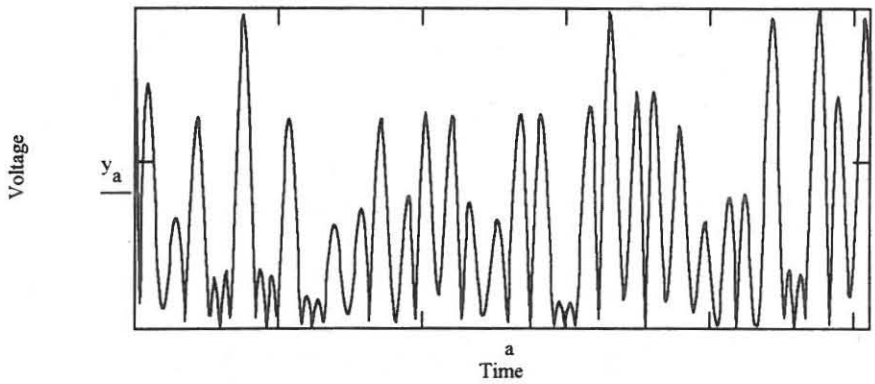


Figure 5.10: Absolute value rectifier output.

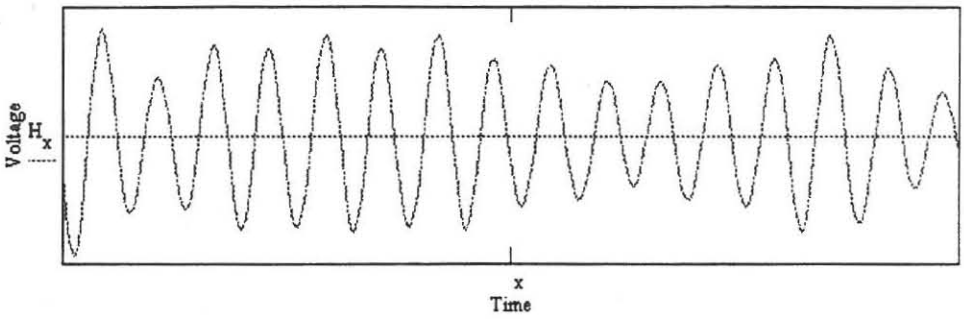


Figure 5.11: Bandpass filtered output of AVR.

Unfortunately, the filtered clock frequency output varies substantially in amplitude, which might have caused an unacceptable amount of jitter.[6] However, since use of the pulse shaping filter as described in Paragraph 2.3.2 results in a fairly wide eye opening, this is not of too much concern.

The circuit as described, was implemented in DSP and evaluated. The results of the evaluation are described in Chapter 8.

5.4 JITTER

The output from a slightly mistuned clock recovery circuit will be a sinusoid at the tuned circuit natural frequency, with a phase dominated by the impulse response from the latest timing pulse. This will result in an output at the required clock frequency. During long gaps in the timing pulses, the oscillations of the tuned circuit will continue at the damped

[6] See paragraph 5.4.

natural frequency and a phase error will build up on the recovered clock [5, p. 176]. This is a major cause of jitter in clock recovery circuits.

It appears that in the case of AVR, there is a gradual increase of jitter as the roll-off decreases [12, p. 1141].

The noise component on the incoming signal causes a shift of the zero crossings of the recovered signal [67, p. 29]. Severe cases of phase jitter may result in pulses moving into time slots allocated for neighbouring data pulses and should therefore be regarded as an undesired form of angle modulation [54, p. 368] which should be avoided as far as possible. Contributing factors to it should be investigated in the study of clock recovery techniques. Takasaki [61, p. 21] identifies two types of jitter, viz.:

- **Type A jitter:** This is caused by mistuning and a finite Q-value of the clock frequency filter. Since a high Q-factor will lead to a clock recovery circuit with less low frequency jitter than one with a lower Q-factor, it is desirable to try and reach the highest possible Q-factor in the clock recovery circuit [5, p. 177].
- **Type B jitter:** This is the predominant type of jitter and is due to asymmetrical waveforms, amplitude-to-phase conversions and pulse overlaps [61, p.21 and 60, p. 877].

It has been shown that the amount of clock jitter can, in general, be kept to a minimum by using a clock recovery circuit with a PLL [67, p. 31].

Conventional full wave rectification, followed by top-half clipping, approximates square-law rectification for ternary pulse trains under some conditions. This non-linearity, however, introduces a considerable amount of jitter in the case of a pulse train with more than three amplitude levels [60, p. 883]. Ridout measured a peak-to-peak jitter of 2% to 4% for a pseudo random data sequence, coded as a 3PRS signal, with the circuit shown in Figure 5.5 [48, p. 355]. As shown in Figure 5.12, rectification of multilevel signals tends to cause jitter [61, p. 92], depending on the relative levels before and after the instant when the signal passes through zero. If the levels before and after rectification are different, the lower voltage level signal would appear to become narrower, whilst the higher voltage level would become wider. Therefore, the instant when the signal passes through zero shifts slightly, depending on whether the higher level signal precedes the lower level signal, or vice versa. Since the timing of the clock signal depends on the moment when the signal passes through zero, this might result in jitter.

The jitter performance of AVR and FLR timing recovery circuits differs considerably. Figure 5.13, as in [12, p. 1142], shows the relative performance of two such systems with baseband 9QPRS input signals.

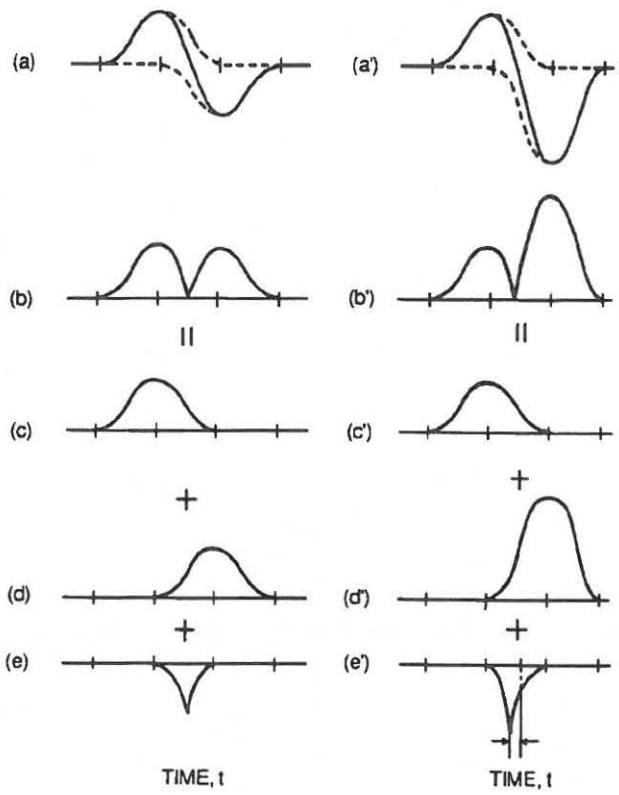


Figure 5.12: Jitter generation due to full wave rectification [61, p. 92].

Amplitude variations of the timing wave are converted into phase variations through imperfections in a timing extractor. It is sometimes very difficult to eliminate these imperfections. So, the amplitude variation must be reduced [60, p. 879].

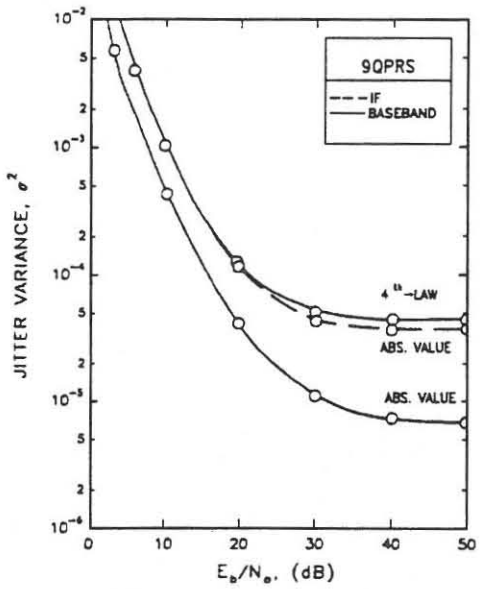


FIGURE 5.13: Comparison of performance of AVR and FLR with 9QPRS inputs [12, p. 1142].

It has been recognised that the amount of timing jitter depends jointly on the shape of the data pulse and the particular sequence of pulse amplitudes (data patterns) [14, p. 914]. Jitter, due to random noise, is not serious [60, p. 877]. According to Killen [30, p. 170] timing jitter is inversely proportional to the signal-to-noise ratio and it can be reduced by reducing the post-filter bandwidth. It has been shown that jitter variance depends on the bandwidth, B, of the clock post-filter in such a way that it is proportional to B^2 at high signal-to-noise ratios and to B at low signal-to-noise ratios [12, pp. 1141-1142].

If timing recovery is done in discrete-time, aliasing must be considered. Any non-linearity will increase the bandwidth. In continuous-time the bandpass filter will reject the higher

frequency components. In the presence of sampling, the high frequency components due to the non-linearity can alias back into the bandwidth of the bandpass filter, resulting in jitter [36, p. 568].

5.5 CONCLUSION

A number of different clock recovery techniques are described in this chapter. The use of non-linearities as a means of regenerating a local clock frequency is discussed in detail. The use of an absolute value rectifier, used in conjunction with a limited amount of pre-filtering, was simulated in MathCAD 6.0 and satisfactory results were obtained. Since the presence of a substantial amount of jitter in the recovered signal can be very detrimental to the performance of such a system, and since use of any type of rectification of a multi-level signal tends to aggravate the occurrence of jitter, the importance of post-filtering is stressed. Use of a PLL in the timing recovery system is also identified as a means of limiting the extent of this problem.

The AVR scheme as described in this chapter was implemented using digital signal processing techniques. The evaluation thereof is described in Chapter 8.

CHAPTER 6

MODEM STRUCTURE AND SIMULATION

Digital signal processing techniques were used in the implementation of the carrier and clock recovery techniques as described in Chapters 4 and 5. Standard SIG-56 DSP development boards, as manufactured by Peralex Electronic Development CC, were used for both the transmitter and the receiver. No hardware was developed during execution of the project. For this reason a minimum general DSP theory is considered in the following chapters. Only the relevant theory is provided to clarify the researcher's programming decisions. The majority of the characteristics of both the data transmitter and the receiver were simulated by making use of MathCAD 6.0. The results of a number of these simulations are shown in this chapter.

6.1 MOTIVATION FOR THE USE OF DSP TECHNIQUES

What made the use of DSP techniques so attractive in this investigation - as in similar modern communication systems - is due to the following key advantages [24, p. 2]:

1. *Guaranteed accuracy.* The accuracy is limited only by the number of bits used.
2. *Perfect reproducibility.* Although the development was not aimed at producing a commercially viable solution to the problems investigated, it is a fact that identical performance from one unit to another is assured in DSP.

3. *There is no drift in performance* as far as temperature or age is concerned.
4. *Flexibility.* DSP systems can be programmed and reprogrammed to perform a variety of functions, without modifying the hardware. Especially in optimising a new technique, as was done during the course of the project, this characteristic of DSP is of the utmost importance.
5. *Superior performance.* DSP can be used to perform functions which are not possible with analogue signal processing.

6.2 DESCRIPTION OF A SIG-56 DSP BOARD

Since both the data transmitter and the receiver consist of a SIG-56 DSP board, a brief description of the main features of these boards is in order.

6.2.1 PROCESSOR

The SIG-56 Board uses a DSP56001 processor. The high through-put of this processor makes it well-suited for communication applications. The main features facilitating this through-put are as follows [40, pp. 1-8 - 1-9]:

- **Speed:** At 10 million instructions per second (MIPS), the DSP56001 can execute a 1024-point complex FFT in 3,23 ms.
- **Precision:** The data paths are 24 bits wide, providing 144 dB of dynamic range. Intermediate results can be held in 56-bit accumulators.
- **Parallelism:** Each on-chip execution unit (such as the programme controller and data arithmetic and logic unit), memory and peripheral operates independently and

in parallel with the other units through a sophisticated bus system. Like most DSP devices [1, p. 8], the DSP56001 uses Harvard architecture, separating programme from data memory.

- **Invisible Pipeline:** The three-stage instruction pipeline is essentially invisible to the programmer, allowing straightforward programme development.
- **Instruction Set:** The 62 instruction mnemonics are mirocontroller-like, making the transition from programming microcontrollers to the DSP56001 easy. The hardware DO loop instruction and the repeat (REP) instruction make writing straight-line code obsolete.

The following is a summary of the main features of the external buses of the DSP56001 processor [9, pp. 39-40]:

- **DATA BUSES:** The four internal data buses are multiplexed to a single external 24-bit data bus.
- **Address Buses:** The three internal address buses are multiplexed to a single external 16-bit address bus.
- **Host Interface:** This consists of an 8-bit parallel port and a group of host control signals. It is used to interface the DSP56001 microprocessor to a host computer (as shown block-diagrammatically in Figure 6.3).
- **Port C Serial Interface:** This consists of a serial communications interface (SCI) and a synchronous serial interface (SSI). These contain all the signals necessary for serial communication and serial data transfer. It can also be used as general purpose input/output pins.

6.2.2 MEMORY MAP OF SIG-56 BOARD

The DSP56001 chip is provided with a substantial amount of internal memory, which, in the case of the SIG-56 boards, is supplemented by external memory chips. The actual configuration of the memory is determined by software control. Depending on the condition in which the control bits of the Operating Mode Register (OMR) are set, the basic memory map can be changed. Figure 6.1 shows a memory map of the SIG-56 board in Mode-0, the mode in which the processor was used during development of the relevant software.

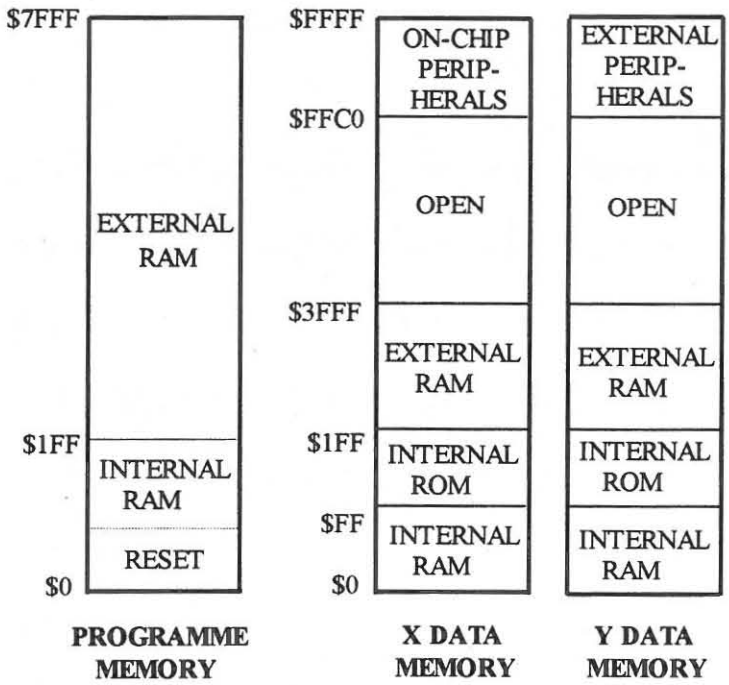


Figure 6.1: SIG-56 memory map.

Y data memory locations $100 - $1FF are factory programmed with a full four-quadrant sinetable [40, p. 3-8]. This table is of special importance with respect to this project since it is used extensively during synchronisation of the carrier frequency.[1]

[1] See Paragraph 4.3.5.1.

6.2.3 ANALOGUE INTERFACE CIRCUIT AND LOWPASS FILTERS

A TLC32044I analogue interface circuit (AIC) provides a complete 14-bit analogue-to-digital and digital-to-analogue converter [64, p. 1]. Use of this chip enabled the realisation of applications requiring sampling rates of up to 19,2 kHz [44, p. 11]. The TLC32044I also includes a programmable switched capacitor lowpass filter. The frequency characteristics of this filter can be adjusted by controlling the values to which two internal registers are set. Since these registers had to be set, the internal timing configuration of the TLC32044I is shown in Figure 6.2 [64, p. 9].

Both the output filter of the data transmitter and the anti-aliasing filter of the receiver make use of the built-in filter of the TLC32044I. In specifying the ideal anti-aliasing filter, it is useful to take the ADC resolution into account. It is preferable that the filter should attenuate the frequencies above the Nyquist frequency to a level not detectable by the ADC, for example to a level less than the quantization noise level [24, p. 19]. Thus, for a system using a 14-bit ADC - such as the TLC32044I - the preferred stopband attenuation of the filter is:

$$A = 20\log(\sqrt{1.5} \times 2^{B+1}) \qquad (6.1)$$

$$= 20\log(\sqrt{1.5} \times 2^{15})$$

$$= 92.1 \text{ dB}$$

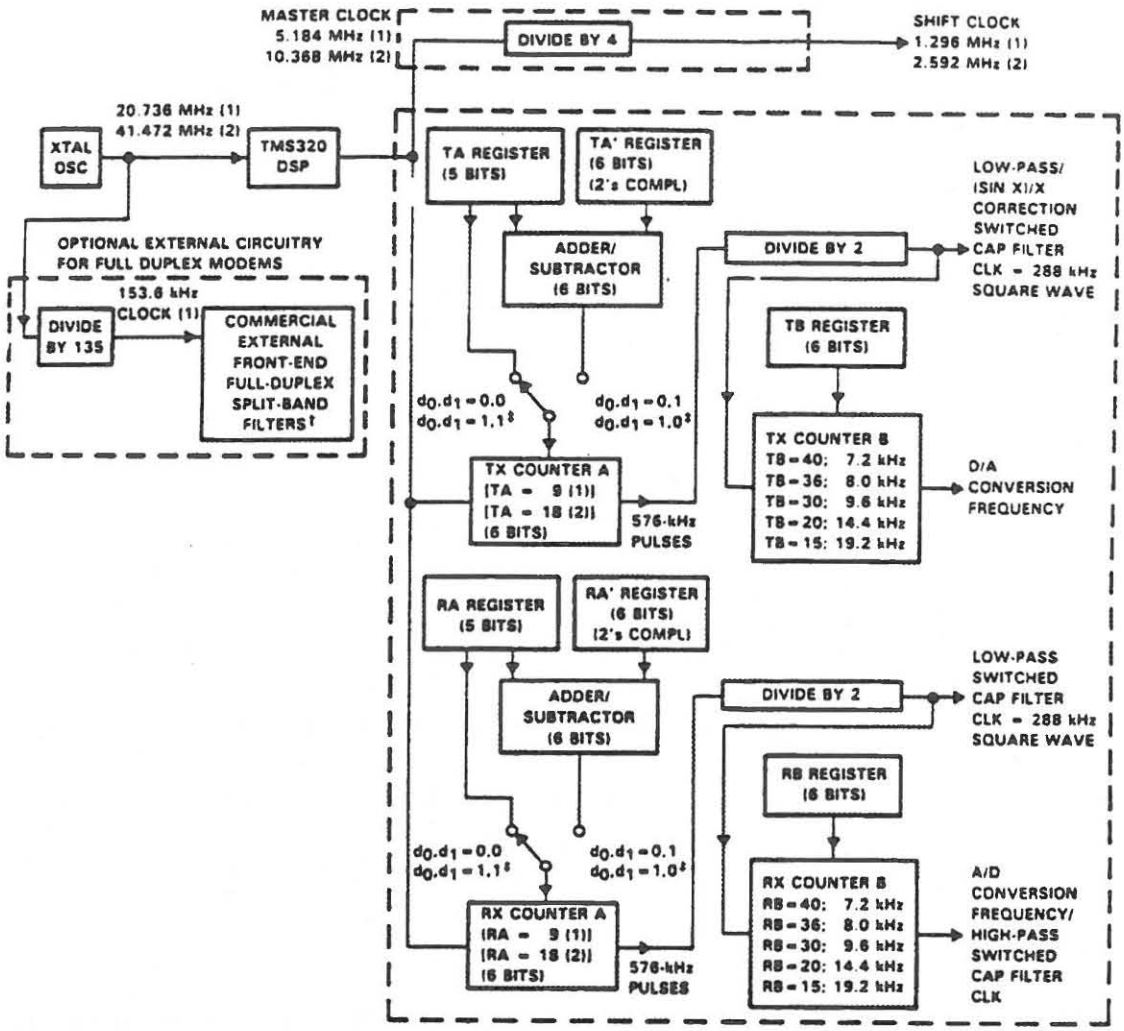


Figure 6.2: Internal timing configuration of TLC32044I chip [64, p. 9].

However, this value could not be attained. The anti-aliasing filter was programmed for a cut-off frequency of 6,8 kHz and provided an attenuation of approximately 68 dB at frequencies above the Nyquist value [64, p. 29]. This was provided for by loading corresponding control words into registers A and B (see Figure 6.2) of the TLC32044I. These values were derived from [64, p. 10] as:

Assume Register B (CB) = 27

Then: $SCF = 27 \times f_s$

$= 518{,}4\ \text{kHz}.$

where SCF is the SCF clock frequency of the TLC32044I

$f_s = 19{,}2\ \text{kHz}.$

Counter A (CA) should now be equal to

$$CA = \frac{Master}{2 \times SCF}$$

$$= \frac{5{,}184 \times 10^6}{2 \times 518{,}4 \times 10^3}$$

$$= 5$$

where: *Master* is the master clock frequency of the TLC32044I.

6.3 LAYOUT OF COMMUNICATION SYSTEM

The DSP boards fit into IBM-compatible personal computers. For this reason, the basic layout as shown in Figure 6.3 was required.

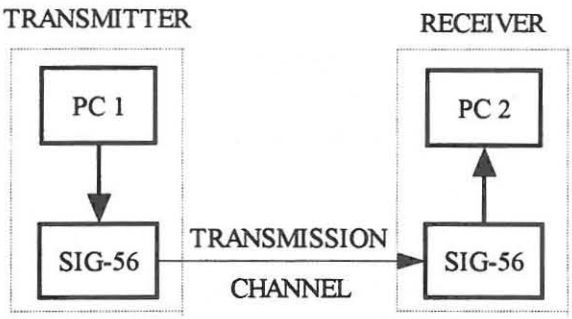


Figure 6.3: Basic layout of data transmission system.

Unfortunately, it was not possible to programme the receive SIG-56 such that the receiver could operate in real time.[2] The total amount of processing required by the carrier and the clock recovery circuits was too time-consuming for that. Therefore, it was decided to execute the receiver code in a number of phases. This process can be described as follows:

1. The transmitter transmits a data file using the *%TX4.PAS*[3] and *%TX4.ASM* programmes. This data file is sampled and stored in the host computer by making use of the *SAMPLE.PAS and SAMPLE.ASM* programmes.
2. A set of three programmes - controlled by the Pascal programme *%RXV.PAS* - process the sampled data as follows:
    - The carrier signal is recovered by an assembly language programme *@CARX2.ASM*. Two carrier signals, in phase with the carriers of the I- and Q-channels, are stored in the receiver host computer as data files.
    - The two carrier files, as well as the sampled data file, are processed by an assembly language programme *@DENX2.ASM*. The transmitted file is demodulated and the 7PRS signals of the I- and Q-channels are stored in the secondary storage of the receiver host computer.
    - The two demodulated sequences are decoded into binary strings by an assembly language programme *@CLK.ASM*. This programme

[2] Detailed descriptions of the transmitter and receiver software are given in Chapter 7.
[3] PAS and .EXE suffixes are used to identify programmes written in Pascal - irrespective of whether such a programme is compiled or not. The same principle is used with respect to assembly language programmes, in which case .ASM and .LOD are used interchangeably.

also uses the I-channel 7PRS signal to recover the clock signal for both the I and Q channels. The decoded data strings are interleaved and the resultant data file is written into the secondary storage of the receiver host computer.

The operation in time of the transmitting and receiver sampling software is shown schematically in Figure 6.4.

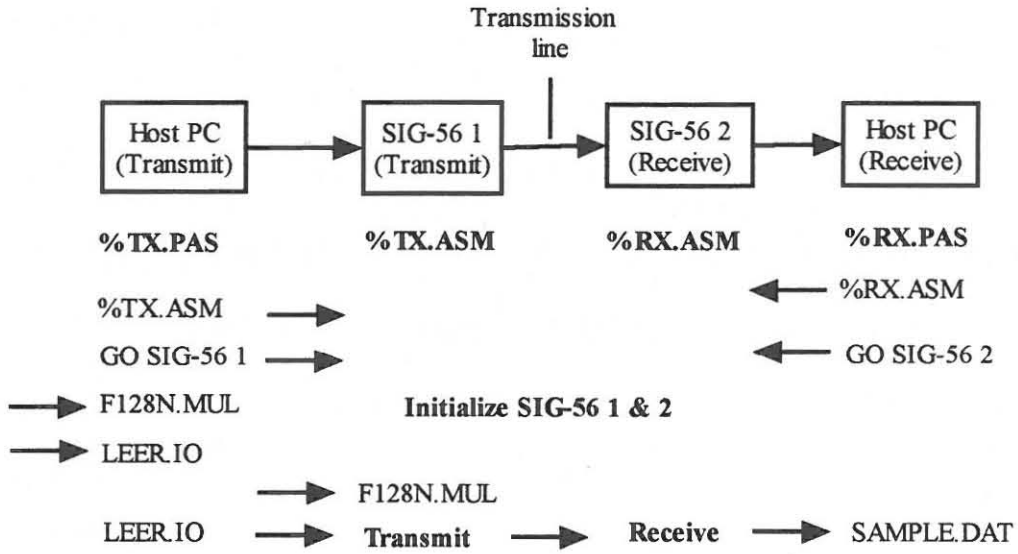


Figure 6.4: Basic operation of transmitter and receiver sampling software.

This process can be described as follows:

1. The transmitter (*%TX4.PAS*) and the receiver (*SAMPLE.PAS*) Pascal programmes are loaded into the respective host PCs.
2. The transmitter (*%TX4.ASM*) and the receiver (*SAMPLE.ASM*) assembly language routines are transferred from the host PCs' secondary storage to the respective SIG-56 boards. The assembly language programmes, *%TX4.ASM* and

SAMPLE.ASM, are initialised in the transmitter and the receiver SIG-56 boards respectively.

3. The coefficients of a pulse shaping filter are transferred from secondary storage of the transmitter host PC to the RAM of the transmitter SIG-56.
4. The data file (*LEER.IO*) which has to be transmitted is loaded from the transmitter secondary storage into the transmitter PC host.
5. The receiver sampling software is executed and the receiver is initialised. No samples are taken as yet.
6. The transmitter is activated and data is transferred to the transmitter SIG-56, encoded, modulated and transmitted. Before transmission of actual data is begun, a short learning sequence is transmitted, followed immediately by the data sequence. During transmission of the learning sequence, the receiver SIG-56 is activated and data is sampled and saved as a data file for later processing by the actual receiver software.

6.4 GENERAL CHARACTERISTICS OF SOFTWARE

Although DSP processors are extremely fast, especially with respect to the execution of correlation and convolution routines, the SIG-56 board was too slow to allow development of the transmitter and the receiver software in a high level language. Instead, all the relevant DSP code was written in assembly language, whilst the host programmes were written in Turbo Pascal[4] Version 6.0. A rather complicated process, to ensure the optimal operation of the software, was followed during development of

[4] Turbo Pascal is a trademark of Borland International.

the software. In this manner, the correct operation - at the maximum possible execution speed - of the software could be ensured.

6.4.1 DEVELOPMENT PROCEDURE OF SOFTWARE

The following procedure was followed during development of the software:

1. As far as possible, the operation of the transmitter, as well as the receiver, was simulated using MathCAD 6.0 before any code was written. In this way, the expected operation of code could be ascertained and evaluated before coding commenced.
2. Assembly language programmes, similar to the simulations referred to above, were developed. These programmes were executed on a DSP56001 simulator and the correct operation thereof verified.
3. The simulated programmes were adapted to interface the SIG-56 board with the relevant host computer and to enable the execution of the simulated routines with the DSP processor.
4. Whenever possible to execute the required routines in real time, the programmes in 3 above were adapted once more to do this. Even though everything possible was done to optimize the execution time of code, it was unfortunately not possible to do the recovery of both the carrier and clock signal in real time. The transmitter and sampling software operate in real time. The sampling, recovery of the clock signal and decoding of a transmitted baseband 7PRS signal were also implemented in real time.

The validity of step 2 above is obvious from Figure 6.5, which shows the spectral contents of the clock recovery circuit output before bandpass filtering. The trace at the top represents the output of the actual software during clock recovery in real-time, whilst the bottom trace shows the expected output as determined by the DSP56001 simulator programme.

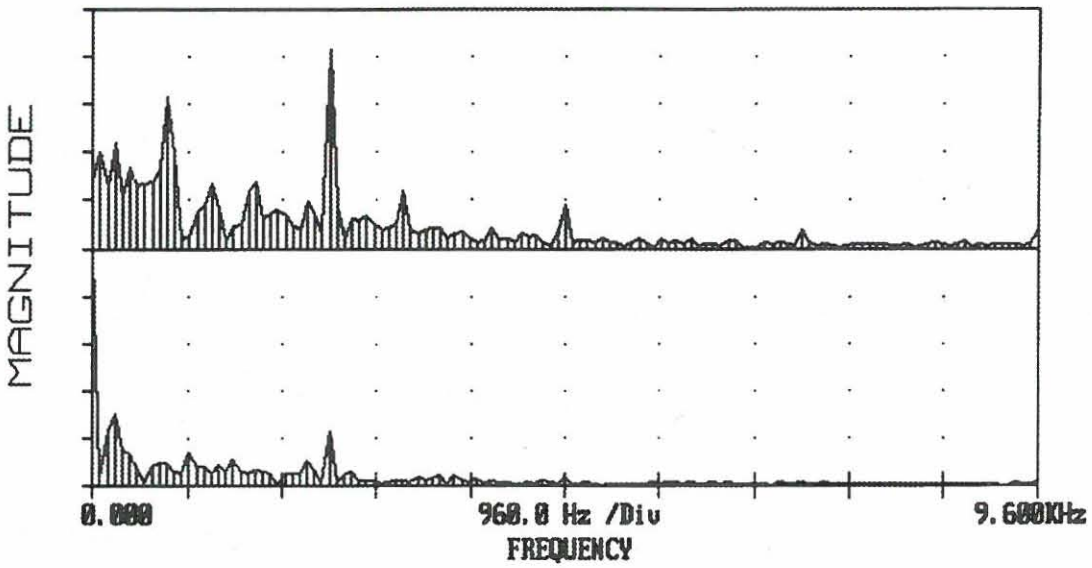


Figure 6.5: Measured (top) and simulated (bottom) frequency spectrum of output of clock recovery circuit.

6.4.2 SAMPLING RATE

The software, as developed, had to be a compromise between the requirements of the transmitted signal - as defined in Paragraph 4.2 - and the capabilities of the SIG-56 boards. With the transmitted signal occupying the frequency range of DC to approximately 6.8 kHz[5], a minimum theoretical sampling frequency of 13.6 kHz was required [58, p. 50 and 25, p. 171] to enable the recovery of the transmitted signal and

[5] See Figure 4.6.

prevent aliasing distortion. For this reason, and in order to facilitate the required anti-aliasing filter characteristic[6], a sampling rate of 19.2 kilohertz was decided upon. This is also the maximum possible sampling frequency attainable with the SIG-56 board [44, p. 11].

6.4.3 EXECUTION SPEED LIMITATIONS

The DSP processor used in the project is virtually obsolete and comparatively slow. Due to the large amount of processing required to realise the receiver in particular, and the relatively inefficient code generated by the available C-compiler, it was decided to write the software in assembly language.

6.5 SIMULATIONS

The simulations of both the transmitter and the receiver are described in relation to their respective block diagrams. In this manner, it will be easier to identify the subcircuits which were not simulated. The simulations improved the researcher's understanding of the underlying principles and provided an easy means for the evaluation of the relative merits of utilising different non-linearities as part of the clock recovery system.

[6] See Paragraph 6.3.3 for details of the anti-aliasing filter.

6.5.1 SIMULATION OF TRANSMITTER

A simulation of the primary elements of the transmitter is shown in Appendix A, whilst a number of typical simulated waveforms are shown and described below.

Figure 6.6 shows a block diagram of the transmitter. A pseudo random data sequence is generated and encoded into 7PRS format. This sequence is lowpass filtered using a pulse-shaping filter.[7]

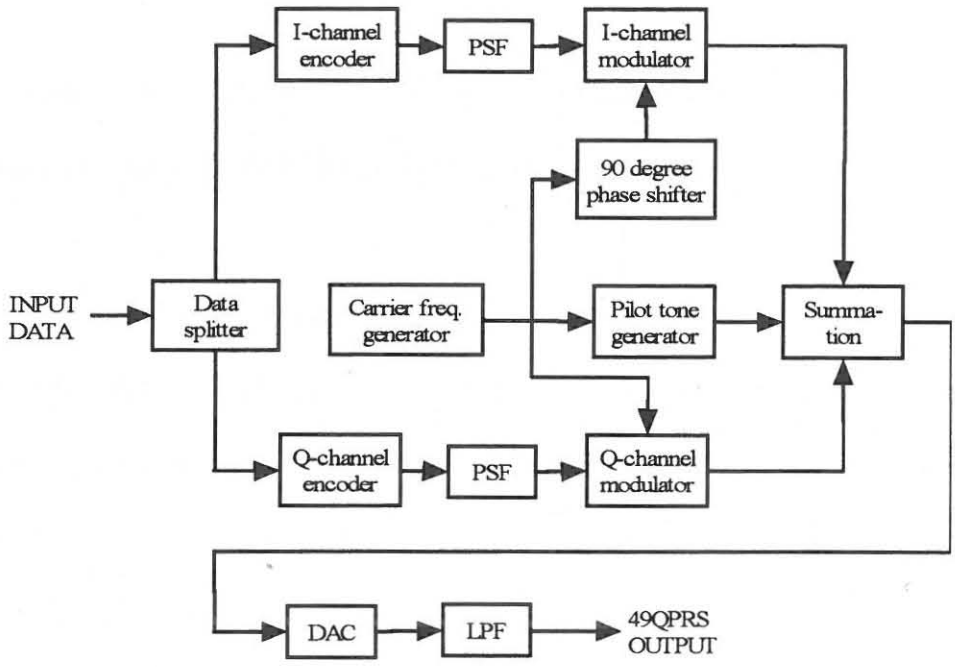


Figure 6.6: Block diagram of transmitter.

Both the unfiltered (top) and filtered (bottom) versions of a short sequence of the encoded 7PRS signal are shown in Figure 6.7.

[7] See Expression (2.12) and Figure 2.4(a).

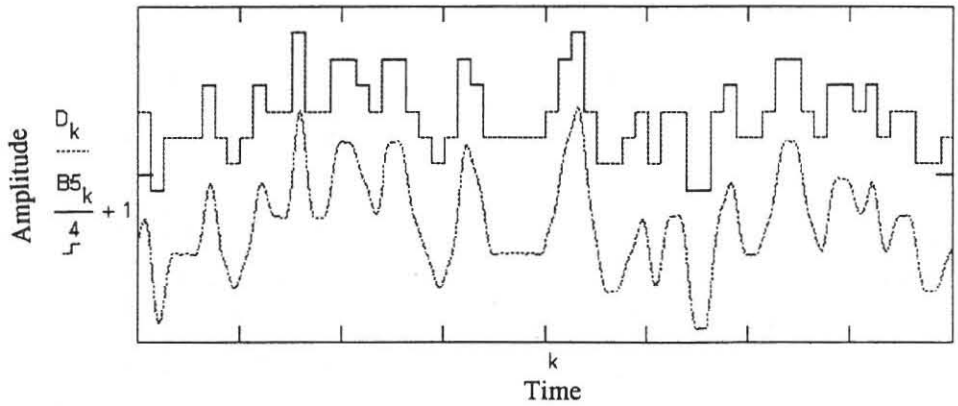
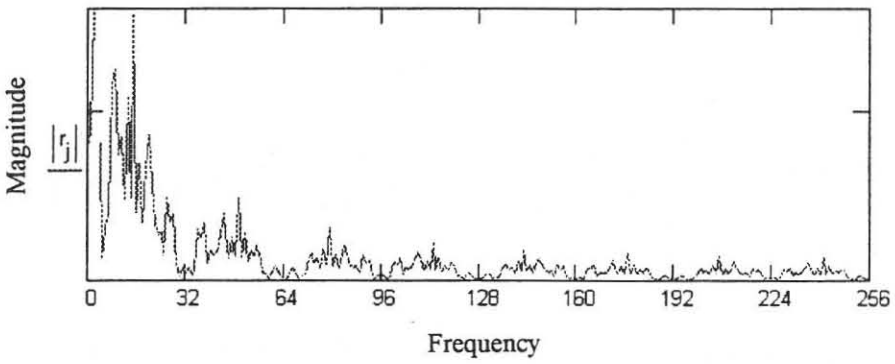


Figure 6.7: Typical unfiltered (top trace) and PSF filtered (bottom trace) 7PRS signals.

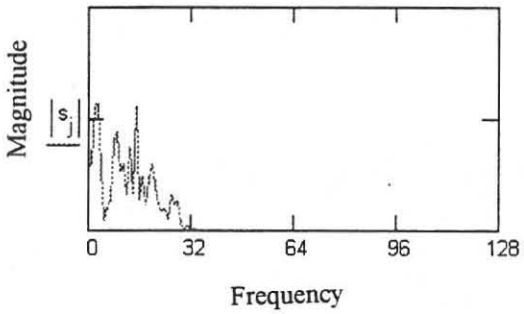
The frequency spectrums of both signals are shown in Figure 6.8 - as well as that of the baseband signal filtered with an ordinary raised cosine filter.[8]

Two of these PSF-filtered signals are modulated in quadrature, the modulated signals summed and the 49QPRS signal is transmitted. Typical waveforms of the I-channel and the modulated 7PRS signals are shown in Figure 6.9. In order to limit the memory requirements of the computer simulating the signals, the same filtered baseband signal was modulated onto both the sine and the cosine carrier signals. Due to this, the 49QPRS signal is not shown.

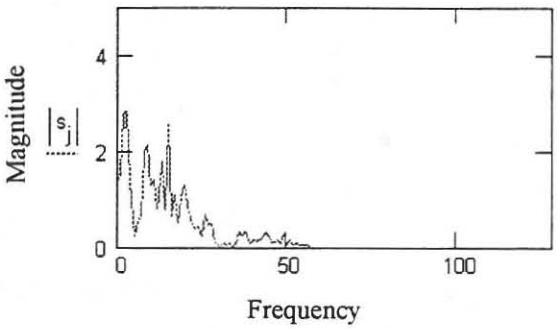
[8] See also Figure 2.4.



(a)



(b)



(c)

Figure 6.8: Frequency spectrum of (a) unfiltered, (b) RC-filtered and (c) PSF-filtered 7PRS signal.

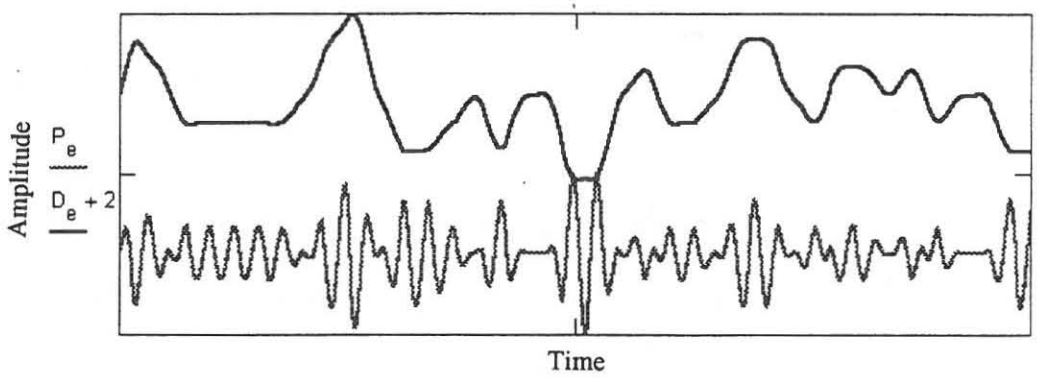


Figure 6.9: Simulated unmodulated (top) and modulated (bottom) 7PRS signal.

6.5.2 SIMULATION OF CLOCK RECOVERY SCHEME

This is considered in terms of a block diagram of the receiver - as shown in Figure 6.10.

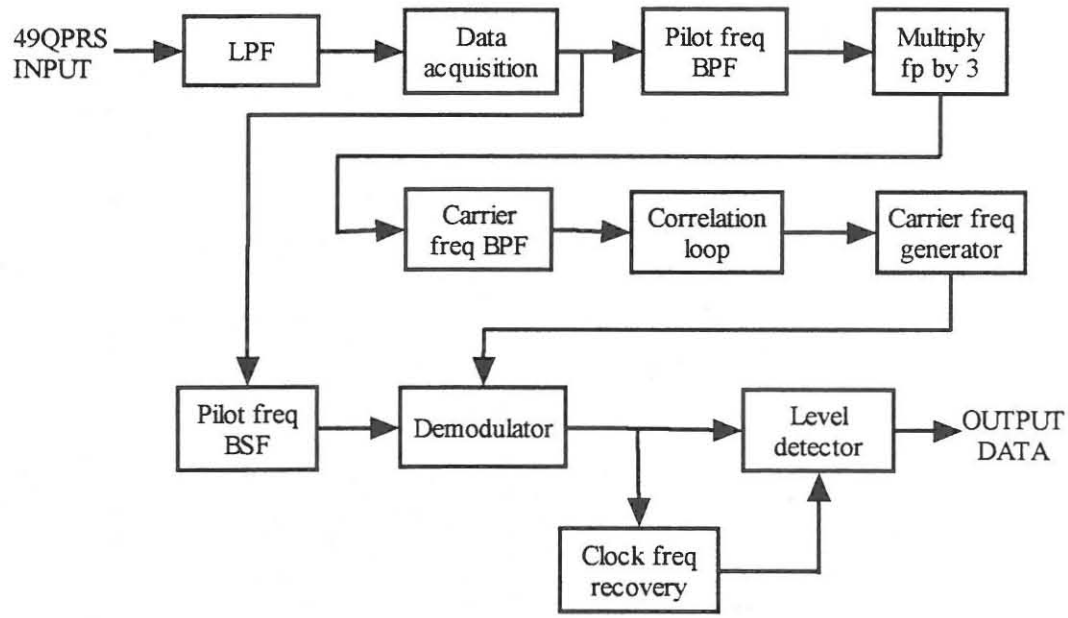


Figure 6.10: Block diagram of receiver.

Part of a simulation of the clock recovery scheme developed during the course of the project is shown in Appendix B, whilst a number of typical simulated waveforms are shown and described below.

The demodulated 7PRS signal is passed through a pre-filter as described in Paragraphs 5.4.4.1 and 5.4.4.2. The output of this filter can be described as:

$$y(n) = x(n) - x(n - \tfrac{T}{2}) \qquad (6.1)$$

where $x(n)$ = the n-th sample of an unmodulated 7PRS signal

T = symbol period of 7PRS signal

This output is connected to an AVR and the output of this circuit can be described by:

$$z(n) = |y(n)| \qquad (6.2)$$

Figure 6.11 shows the signal defined in Expression 6.2.

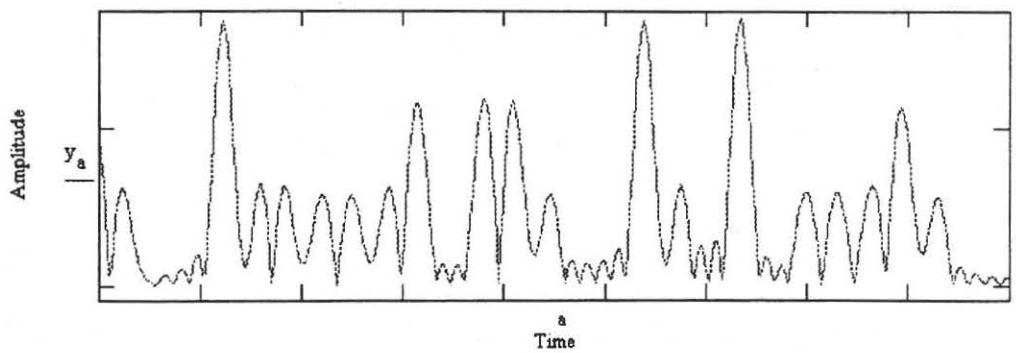


Figure 6.11: Simulated output of absolute value rectifier.

The frequency content of this signal was determined and the characteristic shown in Figure 6.12 was obtained.

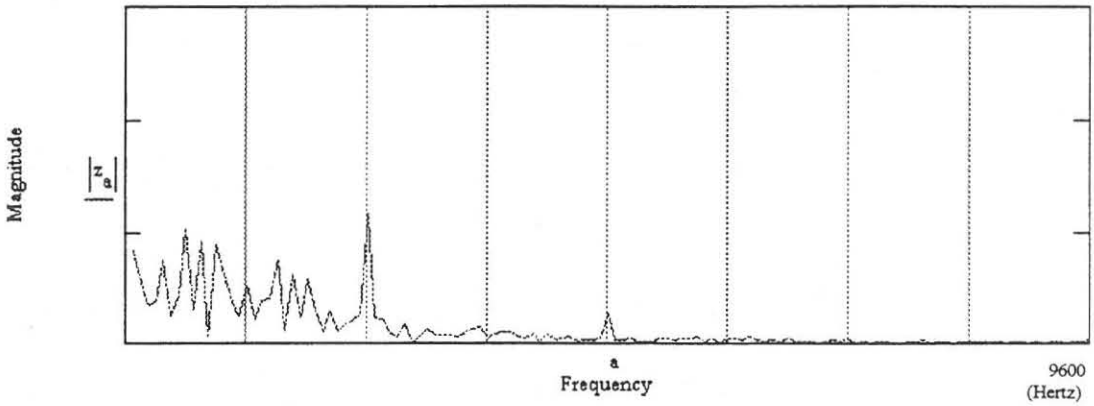


Figure 6.12: Frequency components of rectified signal.

The spectral line corresponding with the second vertical line from the left corresponds with the required clock frequency (of 2 400 Hz). From this figure, it is clear that a substantial frequency component at the correct frequency results, if the so-called AVR-LP technique of clock recovery is used.

The signal as depicted in Figure 6.11 is bandpass filtered to give a sine wave at the clock frequency. A lowpass and a highpass filter were simulated, so that their combined response might approach a characteristic attainable by DSP techniques. The output of the simulated filter combination is shown in Figure 6.13.

Unfortunately, the modulated, recovered clock signal shows a substantial variation in amplitude. However, it was found experimentally, that this does not cause a loss of synchronisation.

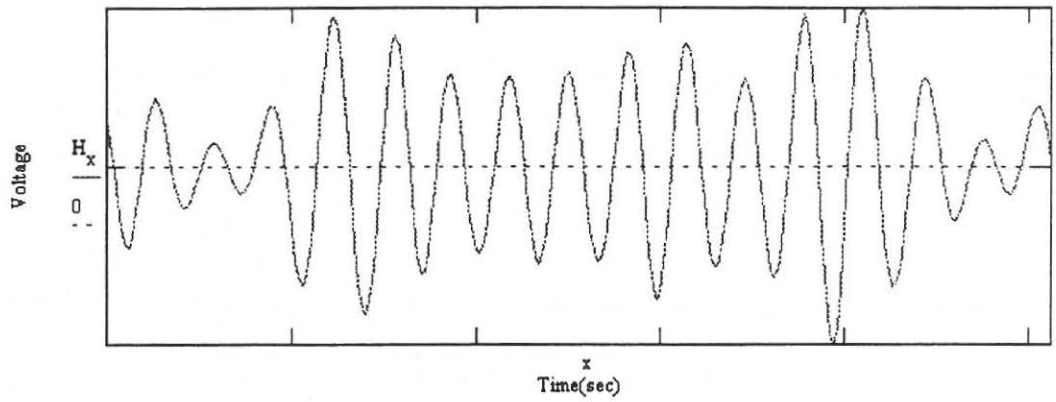


Figure 6.13: Simulated clock frequency output of bandpass filter.

This signal was regarded as sufficient to be utilised in a DSP data communications system and the proposed configurations of both the transmitter and the receiver were converted into code - in a number of different phases, as explained in Paragraph 6.4.1.

6.6 SUMMARY

The expected operation of the proposed transmitter was simulated successfully. From this, the researcher obtained valuable insight into the characteristics of the transmitter. Using the simulated output of the transmitter, a number of timing recovery techniques, where use is made of different non-linearities, were simulated. This proved an invaluable aid to quickly evaluate the expected performance of each of these techniques - and to optimize the relative performance of each by varying different parameters. During the eventual development of the DSP code required to realise both the transmitter and the receiver, these simulations were used - to good effect - to assist in debugging code.

An important product of the simulations was that the relative importance of the value for the excess bandwidth could be investigated for the RC and PS filters. A large value for α appeared to be less important during simulations using the PSF [28, p. 398].

Due to limitations in the simulation software, the carrier recovery system was not simulated by making use of MathCAD. However, during development of the transmitter and the receiver software, intermediate results of the processing of the DSP code were written into the host PC secondary storage. These files were then evaluated by making use of MathCAD, as well as Hypersignal-Plus[9] software.

The simulations, as described, were optimised with respect to the required characteristics of the carrier and the clock recovery techniques. Assembly language code was then developed to realise the simulations.

[9] Hypersignal software is a product of Hyperception and requires software made by Metagraphics Software Corporation to function correctly.

CHAPTER 7

SOFTWARE

The software developed during the course of the project can be divided into three main categories:

1. Transmitter software.
2. Receiver software.
3. Supplementary software.

Each of these is described in this chapter.

7.1 TRANSMITTER SOFTWARE

The advantage of writing an algorithm in a high-level language rather than in assembly language is that fine-tuning and modifications are greatly simplified during development and performance testing [6, p. 44]. However, due to speed constraints inherent in the SIG-56, the DSP code was written in assembly language. The control software, which is executed on the host PC is in Pascal.

As described in Chapter 6, the transmitter software consists of two programmes, *%TX4.PAS* and *%TX4.ASM*. The first controls the transmitter host PC during transmission

of data, whilst the second is executed on the transmitter SIG-56 board. These programmes appear as Appendix F in this document and will now be described in turn.

7.1.1 %TX4.PAS

The following sequence is followed during execution of the *%TX4.PAS* programme:

1. *%TX4.ASM* is read from the host secondary storage and transferred to the SIG-56.
2. This programme is initialised in the SIG-56.
3. The coefficients of the PSF[1] are read from the host secondary storage into the host PC.
4. Data to be transmitted is read into the host PC.
5. The filter coefficients are transferred to the SIG-56.
6. The data is transferred, one longinteger value at a time, on request from the SIG-56, to the DSP board for processing and transmission.

7.1.2 %TX4.ASM

This is the assembly language routine executed by the SIG-56 board during transmission of data. Figure 7.1 shows the flowchart of this routine. Before the data is accessed from

[1] See Paragraph 7.1.3 and Appendix E

the host PC, a short learning sequence is transmitted. This is immediately followed by transmission of the data file. Each longinteger value, transferred from the host PC, is transmitted as six hexadecimal values, each of which is encoded into two dibits, one for the I-Channel and one for the Q-Channel. Eight PS filtered values are transmitted during each symbol period.

7.1.3 COEFFICIENTS OF PULSE SHAPING FILTER

These coefficients are calculated with a supplementary programme, as listed in Appendix E. The operator is prompted to provide the following parameters of the PSF to be designed:

1. The number of symbols over which the filter is to be active. This value is a function of the excess bandwidth.
2. The number of points per symbol. The *%TX4.ASM* routine makes provision for eight values per symbol.
3. The excess bandwidth - with a value of between zero and one.

The programme calculates values for RC and PS filters according to the specifications provided and writes these values to data files. The file containing the PSF coefficients is accessed before commencement of a data transmission cycle.

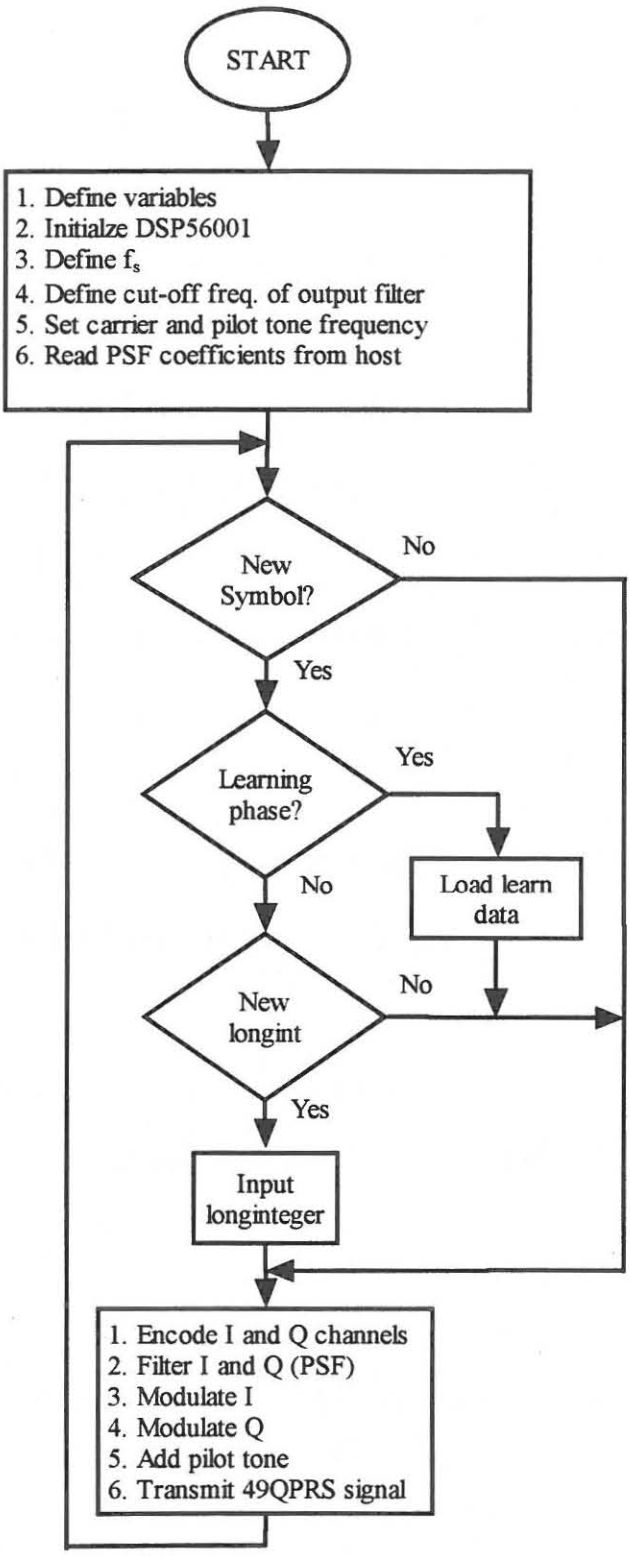


Figure 7.1: Flowchart of *%TX4. ASM*.

7.2 RECEIVER SOFTWARE

As described in Chapter 6, the execution speed of the SIG-56 board is such that the receiver software was found to be too slow to operate in real time. Sampling software was developed for this reason. This software, consisting of *SAMPLE.PAS,* executed on the host PC, and *SAMPLE.ASM*, executed on the SIG-56, sample the transmitted sequence at a frequency of 19 200 Hz. *SAMPLE.ASM* makes provision for anti-aliasing filtering at approximately 6 800 Hz[2]. No other processing takes place at this time. The sampled values are stored in the secondary storage of the host PC, for eventual processing by the receiver software.

One Pascal and three assembly language programmes are used to process the sampled data file, as summarised in Table 7.1.

Table 7.1: Operation of receiver processing software.

PASCAL	ASSEMBLY LANGUAGE	FUNCTION
%RXV		Controls execution of assembly language routines
	@CARX2	Recovers synchronised carrier signals
	@DENX2	Demodulates 49QPRS signal using recovered carriers
	@CLK	Recovers clock signal and decodes I-and Q-Channels

[2] See Paragraph 6.2.3 for calculations defining filters.

7.2.1 CARRIER RECOVERY

Both the in-phase and quadrature carriers are regenerated by *@CARX2.ASM*. A flowchart of this routine is shown in Figure 7.2. The pilot tone is bandpass filtered at 1 200 Hz. This frequency is multiplied by a factor of three and the output is bandpass filtered at 3 600 Hz. This signal is at the carrier frequency, with a phase dependent on the phase of the transmitter carrier. The correct phase of a locally generated carrier signal is now derived from this signal by the repeated calculation of the cross-correlation between the received and regenerated signals.

A correlation process, similar to the half-split method commonly used in faultfinding, is followed to determine the correct phase of this signal. During the first correlation calculation the correct position on the sinetable, to within an angle of 180°, is determined - simply by noting the sign of the correlation value. A negative value indicates a required phase shift of at least 180°. Following the next correlation calculation, the phase is adjusted by 90°, in the direction determined by the calculated value. This process is repeated and smaller phase adjustments are made in each loop. Loveday [38, p. 169] states that if this process were to be fully utilised, with a sinetable of 256 values, the mean number of phase adjustments may be calculated as:

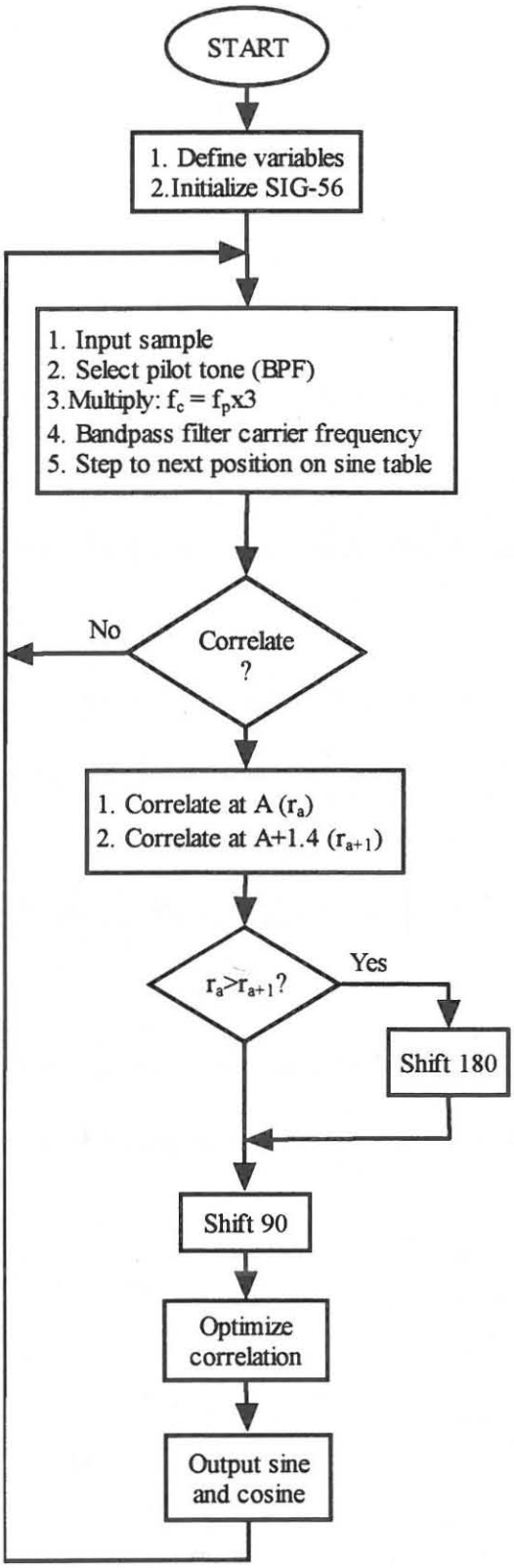


Figure 7.2: Flowchart of carrier recovery routine.

$$C = 3.32 \log_{10} 256 \qquad (7.1)$$

$$= 8$$

where C = Mean number of phase adjustments,

However, this procedure was followed only for phase adjustments down to 11.25°. From this point onwards, synchronisation was optimised by a series of adjustments of 1.4° - corresponding to one position on the sinetable - each. The flowchart shown in Figure 7.3 describes this process.

As soon as the phase synchronisation has been optimised, the same relative positions on the sinetable are maintained for the next eleven samples, after which the process is repeated. These values (positions on the sinetable) are written to a data file used by the *@DENX2.ASM* software for the demodulation of the 49QPRS signal.

The process described above resulted in very fast acquisition of the correct phase. During a long period of comprehensive testing a maximum of 14 correlation cycles were required to acquire synchronisation.

7.2.2 DEMODULATION

The demodulation routine is very straightforward and its execution is summarised below:

1. The variables are defined.
2. The DSP processor is initialised.
3. Sampled values of the 49QPRS signal, as well as the synchronised carrier signals, are read from the host PC.
4. The PRS signal is passed through a bandstop filter - centred at the pilot tone frequency. In this manner the pilot tone, which might have caused distortion in the demodulated outputs, is removed.
5. The PRS signal is multiplied with the (1) sine and (2) cosine carriers.
6. Both multiplied signals are lowpass filtered and the I and Q baseband signals are written to a data file on the host PC.

This programme was evaluated with and without the pilot tone bandstop filter and no distortion of the baseband signals could be identified - presumably due to the frequency of the pilot tone.[3]

[3] See Paragraph 4.3.5.1 for motivation of pilot tone frequency used.

7.2.3 CLOCK RECOVERY AND DECODING

The clock signal is derived from one baseband 7PRS channel by means of *@CLK.ASM*. The recovered clock is used to decode the I and Q 7PRS signals and the outputs are saved in a data file in the host PC. An assembly language routine, *TIME3.ASM*, identical to *@DENX2.ASM*, except that it receives a transmitted 7PRS baseband signal, recovers the clock and decodes the data in real time, is described below. The flowchart of this routine is shown in Figure 7.4.

The sampled 7PRS signal is passed through a pre-filter[4] before the absolute values are derived. These values are passed through a bandpass filter centred at 2 400 Hz. This signal corresponds with the symbol rate and is used to time the decoding procedure. For reasons described in Paragraph 3.5, the decision levels are not separated by the same amount. The relative values as listed in Table 7.2 were determined experimentally to ensure the best performance by the decoding circuit.

[4] See Paragraph 5.4.4.1 for description of the pre-filter.

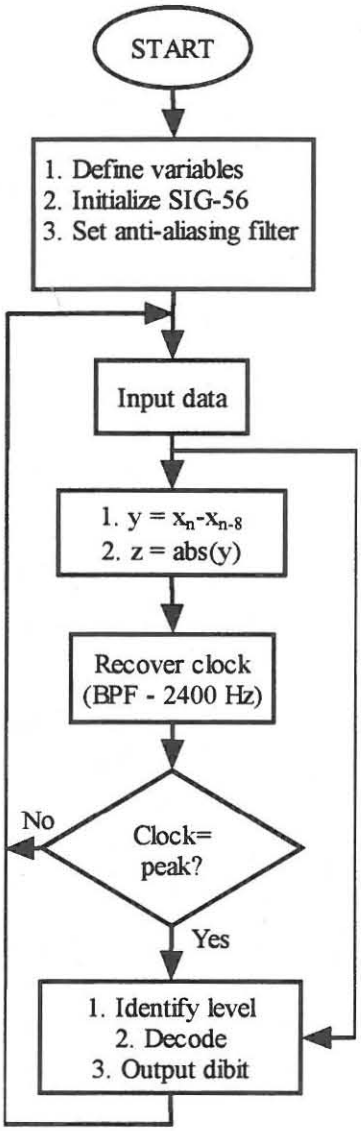


Figure 7.4: Flowchart of clock recovery routine.

Table 7.2: DECISION LEVELS

EYE NUMBERS	SPACING BETWEEN SUCCESSIVE DECISION LEVELS (%)
1 to 2	19.3
2 to 3	20.6
3 to 4	20.2
4 to 5	20.6
5 to 6	19.3

7.3 DESIGN OF DSP FILTERS

The different filters referred to in the preceding paragraphs were designed using Hypersignal Plus software. The basic specifications of the filter are supplied by the user and the coefficients and response of the filter are calculated. These coefficients are used in the actual code.

One of the problems relating to FIR filter design is the estimation of the number of taps required to realise a particular frequency response [15, p. 165]. Direct truncation of the series leads to the Gibbs phenomenon, which manifests itself as a fixed percentage of overshoot and ripple before and after an approximation of the ideal LPF. This effect can be alleviated by using a window to modify the filter coefficients [47, p. 88]. Hypersignal

Plus software enables the designer to evaluate the performance of the filter as designed before coding begins [23, p. 79] and the effect of different numbers of coefficients, as well as the effect of using a particular window, can be considered by the designer.

Appendix C shows a typical output of the software - in this case the pilot tone bandpass filter described in Paragraph 7.2.1. The frequency response of the filter can also be evaluated before coding begins. Figure 7.4 shows this response of the pilot tone bandpass filter. All the filters in the receiver were designed and programmed in this manner.

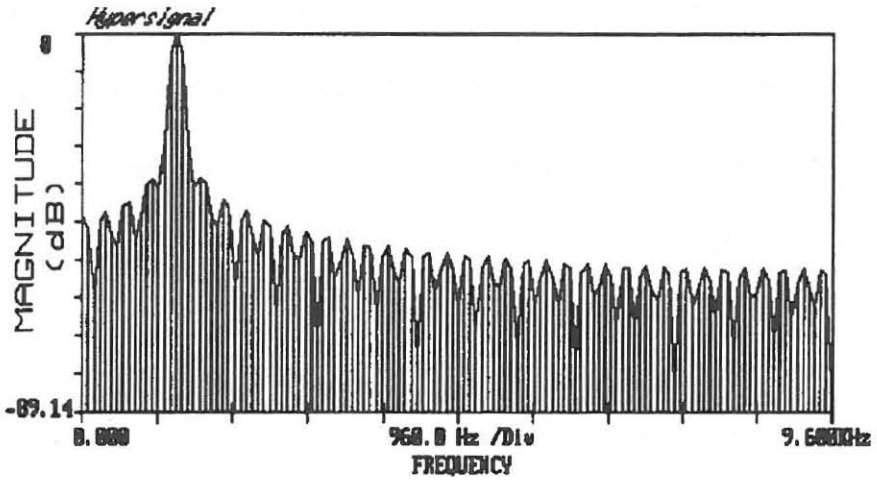


Figure 7.4: Frequency response of pilot tone bandpass filter.

7.4 TESTING AND DEBUGGING

Testing should not be confused with debugging. Testing ensures correct results under all conditions, whereas debugging are the steps taken to correct errors becoming apparent

during testing. The two major classifications of software testing are functional and structional testing. Functional testing is conducted without any consideration of the internal structure of the programme. A major weakness of functional testing is that there is no way to be sure that testing is complete. The aim of structural testing is to ensure that the entire programme code has been executed during testing [22, pp. 14-15]. Another way of evaluating the testing performance is by determining:

1. The percentage of code executed.
2. The percentage of number of branches tested in both directions.

According to these standards, the programmes developed during the course of the project have been thoroughly tested and debugged. Care has been taken that test cases were selected in such a manner as to verify the programme performance under all possible input conditions.

7.5 SUMMARY

Programmes have been developed to realise a 49QPRS data transmitter. This code includes a PSF. The transmitter transmits data at a rate of 9 600 bits/second. Pascal has been used for the programme executed on the host PC, whilst assembly language was used for the DSP code. A programme has also been developed to calculate the coefficients of the PSF, implemented in the transmitter. The total PSF response is obtained by a single

filter, since execution of the receiver's code was expected to be much more time consuming.

A group of programmes are required to realise the receiver, since the required amount of processing could not be managed in the available time. The transmitted signal is sampled during transmission, and afterwards processed in three phases:

1. Recovery of the carrier signal.
2. Demodulation of the received 49QPRS signal.
3. Recovery of the clock signal and decoding of the data.

The programmes were tested thoroughly and debugged where necessary.

CHAPTER 8

MEASUREMENTS AND RESULTS

The experimental results of the project are described below. After describing the experimental layout, the performance of both the data transmitter and the receiver are discussed. These results were, where possible, also compared with the expected, simulated values as described in Chapter 6.

8.1 INTRODUCTION

Figure 8.1 shows the experimental set-up used during evaluation of the carrier and clock recovery techniques as investigated during execution of the project.

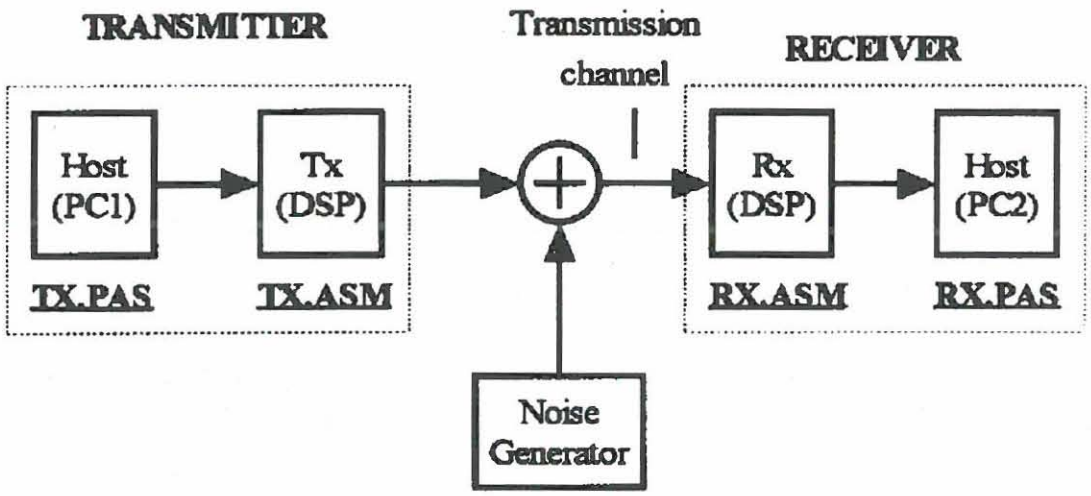


Figure 8.1: Experimental set-up during evaluation.

Before discussing the experimental procedures adopted to evaluate the operation of the recovery circuits, the design and operation of the noise generator and summation circuit shown in Figure 8.2 are described.

8.2 NOISE GENERATOR AND SUMMATION CIRCUIT

Figure 8.2 shows a circuit diagram of the wideband noise generator and summation circuit.

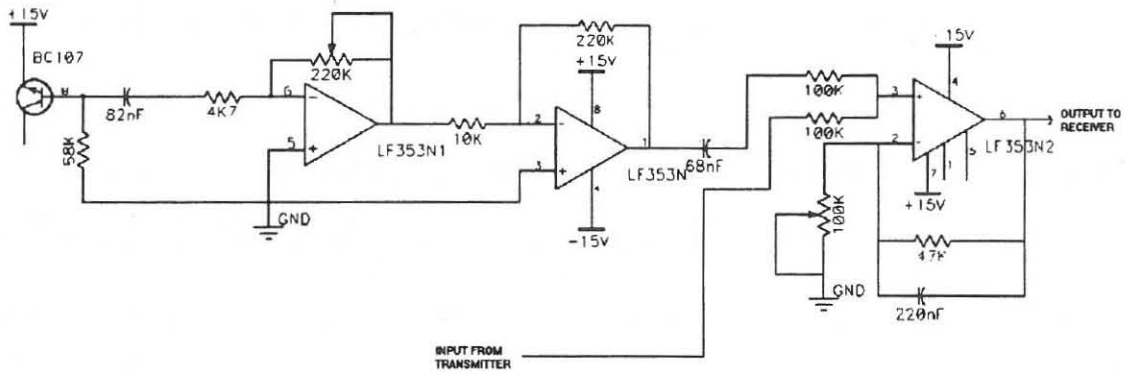


Figure 8.2: Circuit diagram of wideband noise generator and summation circuit.

The frequency spectrum of the output of the circuit in Figure 8.2 is shown in Figure 8.3. The amount of noise added to the transmitter output can be controlled accurately by adjustment of the multi-turn feedback potentiometer on the noise amplifier (LF353N1 in Figure 8.2).

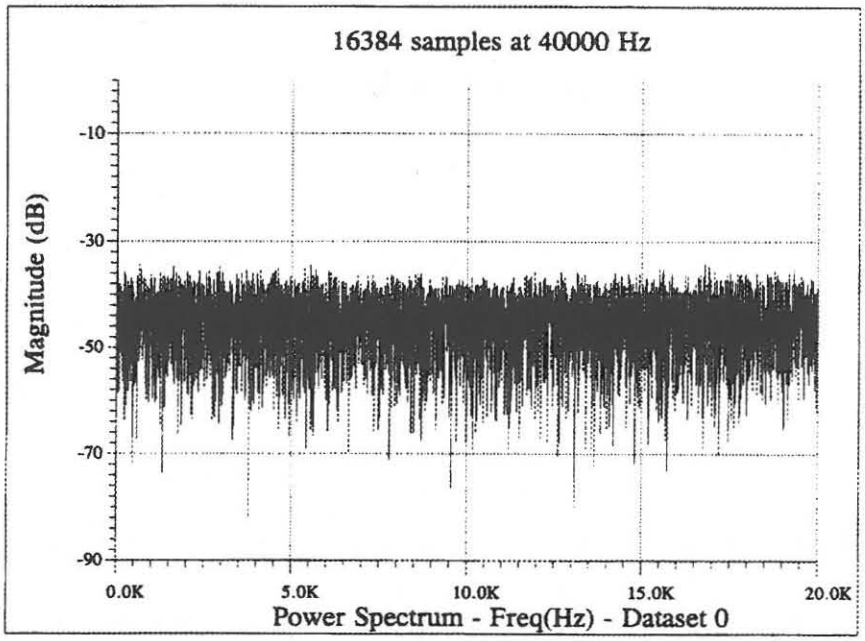


Figure 8.3: Frequency spectrum of noise generator and summation circuit output.

During signal to noise measurements, the noise level was determined by disconnecting the transmitted signal and sampling only the noise signal. An assembly routine was coded to calculate the mean-squared value of the noise for a large number of samples. The level of the signal with noise and without noise was determined in a similar manner, and these values were used to calculate the relevant SNR during evaluation of the system's performance.

8.3 EVALUATION OF TRANSMITTER

The following performance criteria of the transmitter were evaluated:

1. Output waveform of unmodulated 7PRS signal.

2. Eye diagram of PSF-filtered 7PRS signal.
3. Output waveform and frequency response of modulated 49QPRS signal.

The experimental results of each of the above performance criteria are given below:

8.3.1 CHARACTERISTICS OF UNMODULATED 7PRS SIGNAL

These signals were measured by making use of a PC-30D analogue-to-digital conversion card, at very high sampling rates - as shown in the figures below. Figure 8.4 shows the unmodulated output signal of the I-channel during the learning phase.

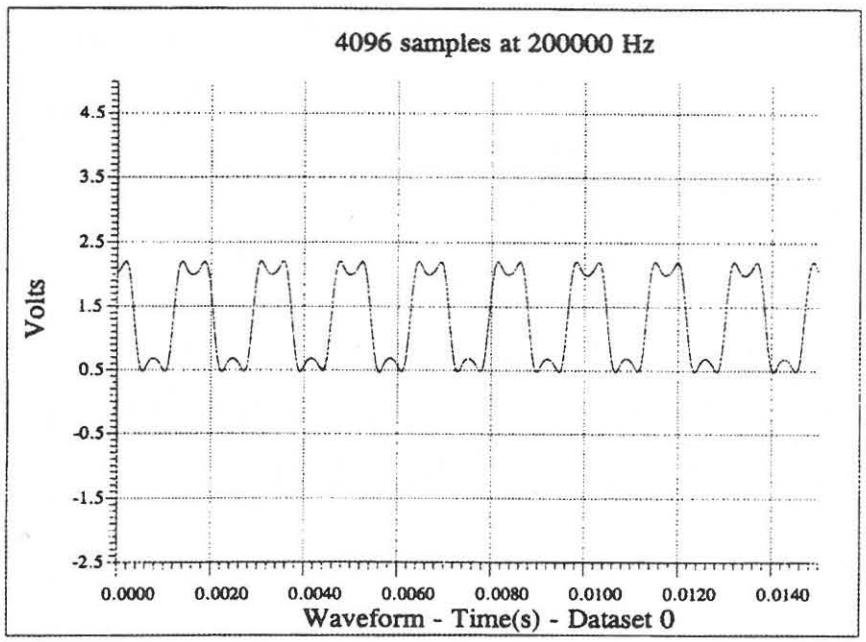


Figure 8.4: Filtered, unmodulated 7PRS signal during learning phase.

From the figure, the different levels are easily discernible. The correct encoding during the learning phase can also be verified by considering the training data. In this case, the following sequence, as summarised in Table 8.1 was anticipated.

Table 8.1: Data during learning phase of I-channel.

Time (k)	-1	0	1	2	3	4	5	6	7	8	9	10
a_k		0	0	3	3	0	0	3	3	0	0	3
b_k	0	0	0	3	0	0	0	3	0	0	0	3
c_k		0	0	3	3	0	0	3	3	0	0	3
d_k		0	0	3	3	0	0	3	3	0	0	3

where: a_k = k-th dibit
c_k = k-th 7PRS symbol
d_k = k-th decoded dibit

The unmodulated output during transmission of a pseudo random data sequence was also measured. Figure 8.5 shows the output under these conditions. The seven levels of the baseband 7PRS signal are easily discernible. The correctness of the coding process was verified by comparing the baseband output of the transmitter for known data sequences with hand-encoded values. This process was executed for a large number of data sequences. The output was also compared with values obtained by means of MathCAD simulations.

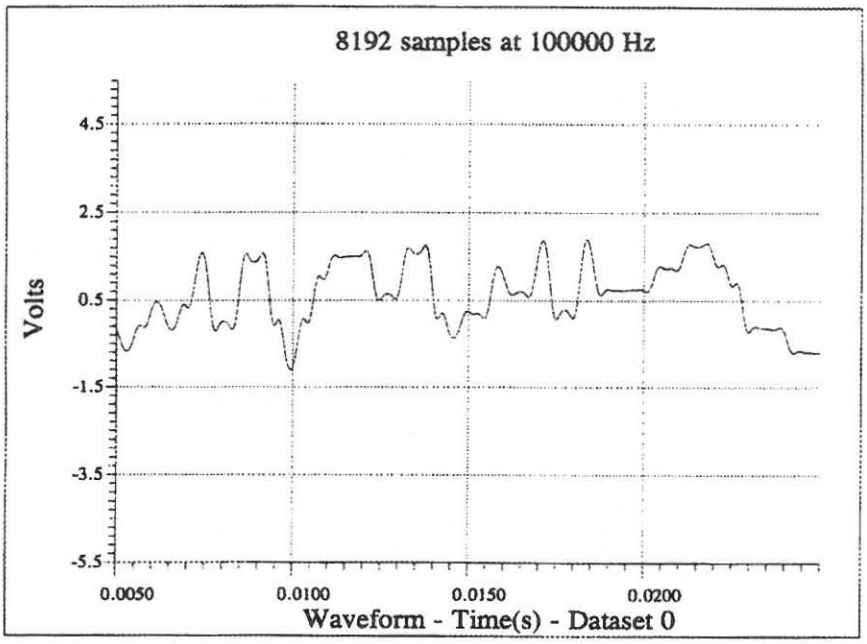


Figure 8.5: Unmodulated 7PRS signal during transmission of a pseudo random data sequence.

These signals were also measured as follows:

1. A baseband signal was transmitted, sampled by the DSP receiver (operating in the baseband mode) and written to a secondary storage file on the receiver host PC.

2. This signal was then evaluated using the Hypersignal-Plus software, or in some cases with MathCAD 6.0.

Similar results to those already shown were obtained. This process was followed to verify that the receiver SIG-56 card, in fact, samples the same signal measured as the transmitter

output. If that was not the case, the performance of the receiver could never have been expected to be satisfactory.

8.3.2 EYE DIAGRAM OF TRANSMITTED 7PRS SIGNAL

Due to the lack of an instrument capable of displaying the eye diagram of the transmitted signal, software was developed which made it possible to ascertain the eye diagram of any sampled signal in the project. This software appears as Appendix D in this document.

Using the stored SIG-56 sampled values discussed above, the eye diagram of the transmitted signal was determined. The eye diagram for a sampled pseudo random data sequence is shown in Figure 8.6.

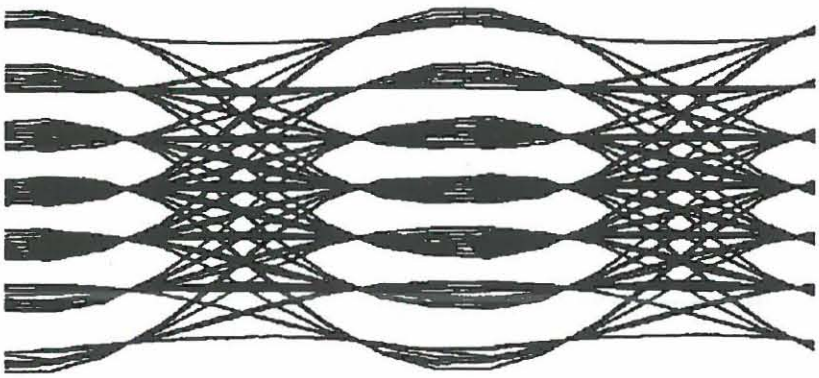


Figure 8.6: Eye diagram of sampled baseband 7PRS signal.

The relative wide eye opening of the PSF-filtered, and therefore, the relatively lenient specifications required from the clock recovery circuit, is obvious from this figure. It is also clear that, at the optimum sampling instants, no substantial ISI is present in the signal.

8.3.3 CHARACTERISTICS OF MODULATED 7PRS SIGNAL

A similar procedure was followed to evaluate the modulated output of the transmitter as described above with respect to the unmodulated output. The following measurements were made on the modulated signal:

1. The waveform of a single modulated data channel during the learning phase.
2. The waveform of the modulated 49QPRS signal.
3. The spectral composition of a modulated 49QPRS signal with the pilot tone.

Figures 8.7 to 8.9 show these measurements.

From these figures it is clear that the output of the transmitter is indeed a Class I encoded 49QPRS signal with a carrier frequency of 3 600 Hz and a pilot tone of 1 200 Hz. This signal was then used to evaluate the clock and carrier recovery techniques developed during the course of the project.

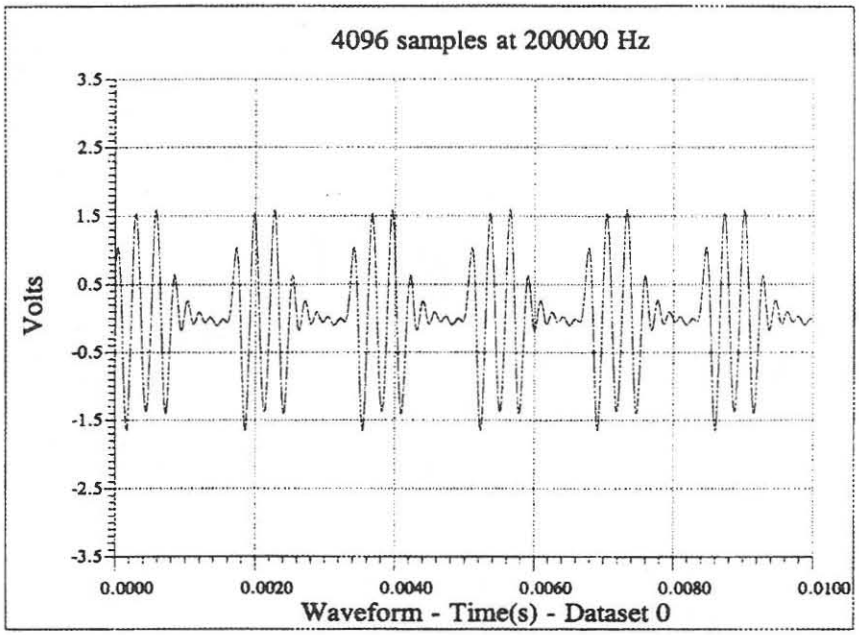


Figure 8.7: Modulated 7PRS signal during learning phase.

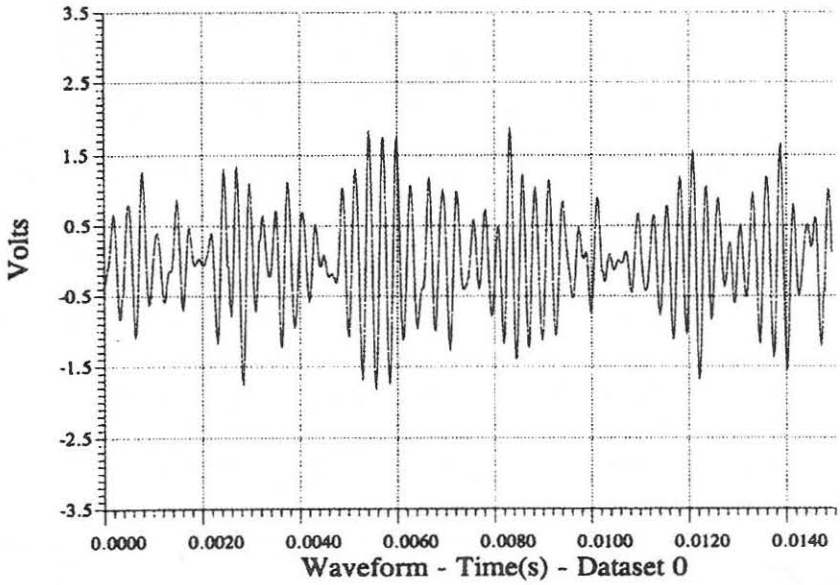


Figure 8.8: Modulated 49QPRS signal.

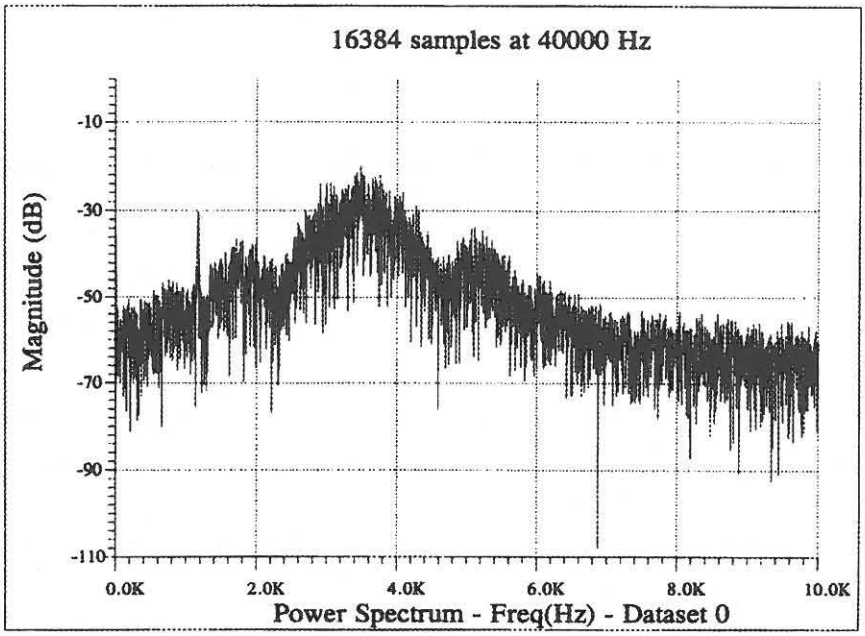


Figure 8.9: Frequency content of modulated 49QPRS signal with pilot tone.

8.4 EVALUATION OF RECEIVER

Since the aim of the project was to investigate clock and carrier frequency recovery techniques, and not the development of a commercially viable modem, no measurements regarding the probability of receiving data in error were conducted. Rather, the acquisition time and tracking characteristics of the proposed recovery techniques were evaluated by the transmission and reception of pseudo random data files, processing of the received signal and the writing of the outputs to files. These files were then examined by means of MathCAD 6.0 and Hypersignal Plus 3.0 software.

8.4.1 PERFORMANCE OF CARRIER FREQUENCY RECOVERY TECHNIQUE

The performance of the carrier recovery scheme was evaluated with respect to three main parameters, viz.:

1. The acquisition time.
2. The effect of the level of the pilot tone on the tracking performance of the recovery system. It was also important to identify the minimum level of the pilot tone required to maintain synchronisation.
3. The performance of the carrier recovery scheme at different SNRs. From this, the relative noise immunity of the scheme could be ascertained.

With respect to the evaluation of the effect of different levels of pilot tone, the following procedure was followed:

1. The pilot tone was adjusted to a predetermined level and a data sequence was transmitted and sampled using the *sample.pas* and *sample.asm* set of programmes. The sample data was stored as a *sample.dat* file.

2. The *rxv.pas* programme was executed on the *sample.dat* data file. The relative position in the sinetable is derived (and saved as a data file) during this process.

The above process was repeated a number of times for each of six different levels of pilot tone and the data files were processed. The basic acquisition characteristics of the carrier recovery scheme is demonstrated in Figure 8.10 which shows a typical waveform before, during and after acquisition. Initially, the local and remote oscillators are unsynchronised, but after deriving twelve values of the recovered carrier frequency signal, it is correlated with the locally generated signal and a major phase adjustment is made. After this, the two signals remain in synchronisation with only minor phase adjustments every eleventh sample, if necessary. The instantaneous value of the locally generated signal is determined by the position on the built-in sinetable of the DSP processor.

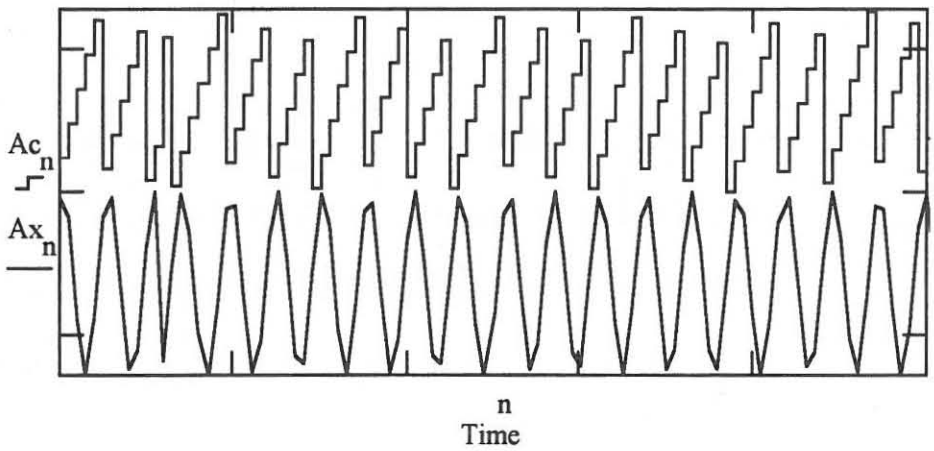


Figure 8.10: Position on sinetable and output waveform before, during and after acquisition of synchronisation.

Figure 8.11 shows a typical acquisition and tracking response for a pseudo random data sequence. The first adjustment, after twelve samples (at a sampling frequency of 19 200 Hz), represents a major phase change - approximately 124 degrees in the figure. This is

followed by a series of small adjustments, none of which exceed 3 degrees - representing a relative shift of two positions on the sinetable.

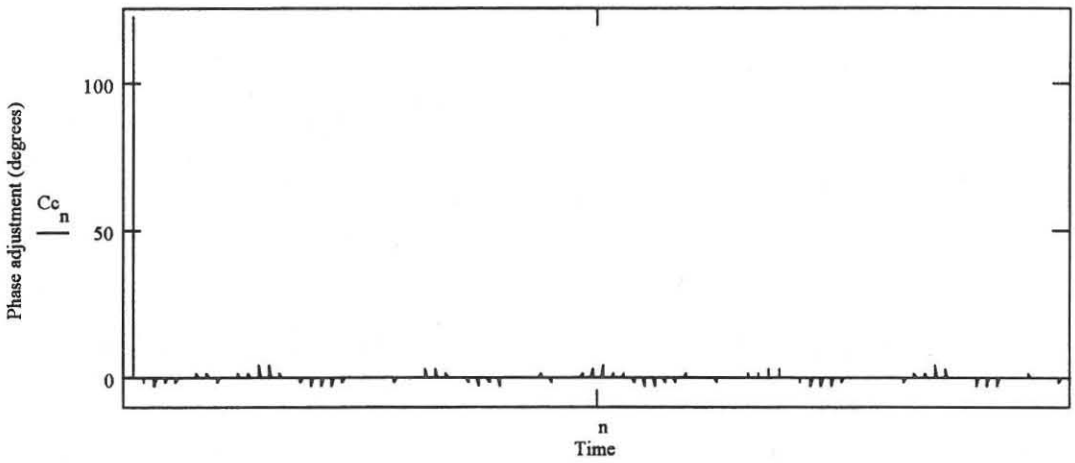


Figure 8.11: Typical acquisition and tracking response of carrier recovery circuit.

It is therefore clear that acquisition takes place only after twelve samples, i.e. 625 μs, from the beginning of reception. This was verified by monitoring this process during the transmission of a large number of data sequences. Once acquisition had taken place, the receiver oscillator was able to track the transmitter very accurately.

The average size of the phase adjustment was found to be related to the level of the pilot tone transmitted. Figure 8.12 shows the average phase adjustment for different levels of pilot tone. From this, it is obvious that a lower pilot tone level results in poorer tracking. With a pilot tone of less than -35 dB relative to the peak data level, there is a sudden increase in the extent to which the phase angle needs to be adjusted to maintain synchronisation.

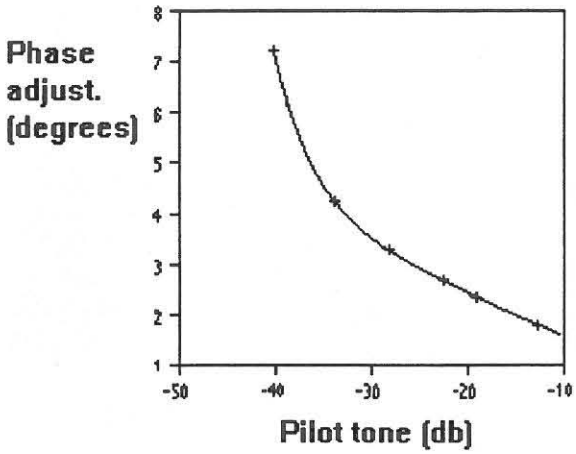


Figure 8.12: Average size of phase adjustment relative to the level of pilot tone.[1]

From the figure, it is obvious that at a pilot tone of -30 dB relative to the data level, the average phase error is approximately 3 degrees. A phase error in the local oscillator causes a variable gain factor in the output signal[2], that is proportional to the cosine of the phase error. A tracking error limited to 3 degrees limits this variable gain factor to less than 0.14%, whilst an error of 10 degrees causes a gain factor of approximately 1.5%. Considering this, use of pilot tones at levels of less than -30 dB are not recommended.

The distribution of the phase adjustments also depends on the pilot tone levels. For lower levels of tone, the distributions are less concentrated around a phase shift of zero degrees. Figures 8.13(a) to 8.13(f) show typical distributions for different pilot tone levels. In each case, the frequency at which each value of phase adjustment occurred, was recorded.

[1] All the levels of the pilot tone referred to in the discussion should be regarded as relative to the peak modulated signal level.
[2] See Paragraph 4.3 and Expression 4.3.

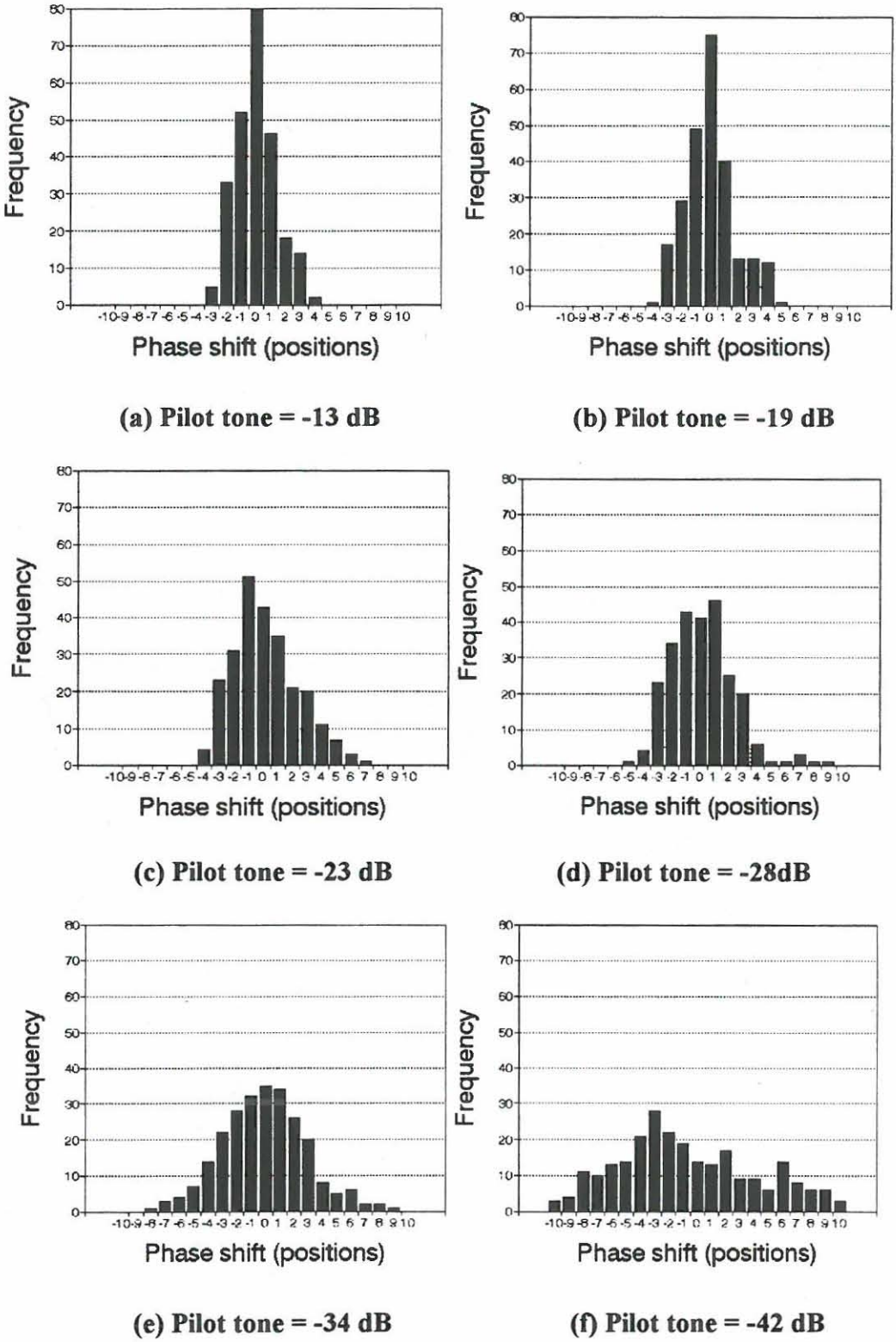


Figure 8.13: Distributions of tracking performance at different levels of pilot tone.

Figure 8.13 shows a gradual deterioration from the ideal distribution characteristic with a decrease in the level of the pilot tone. This is true for pilot tones with a level as low as -34 dB. However, at -42 dB the distribution is seriously degraded, as was to be expected from Figure 8.12.

The distributions shown in Figure 8.13(a) to (f) are all suffering from skewness. Values between 0.3 and 0.54 were measured in this regard. This points to a deviation from a normal distribution [50, p. 97]. Figure 8.14 shows the normal quantile plot for the tracking characteristic with a pilot tone of -23 dB. One can gather from this figure - which is typical of the characteristics measured in this regard - that this deviation is not excessive.

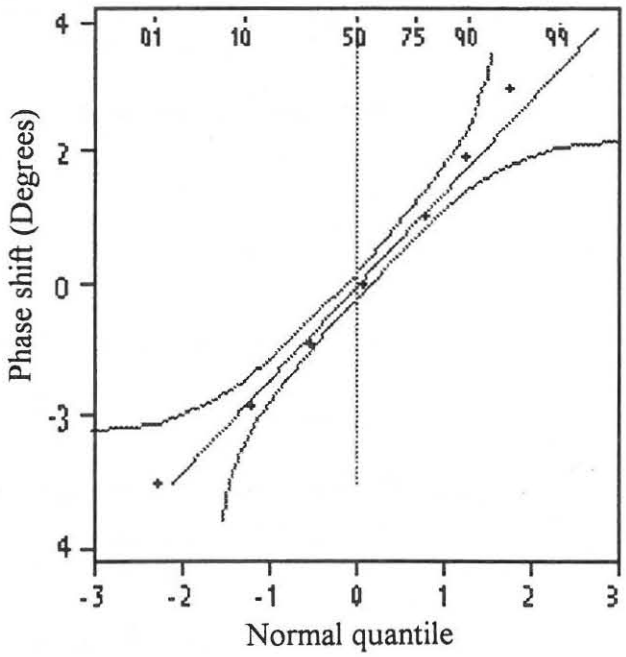


Figure 8.14: Normal quantile plot of tracking characteristics for a pilot tone of -23 dB relative to the peak modulated signal level.

The plotted points in Figure 8.14 represent the distribution of the tracking characteristic. This obviously follows the straight line, which represents a normal distribution, closely.

The spectral purity of the recovered carrier wave depends on the quality of tracking, and also deteriorates with a decrease in the amplitude of the transmitted pilot tone. Figure 8.15(a) and (b) show this characteristic for pilot tones of -19 dB and -34 dB respectively. The same characteristics for higher levels of pilot tone are considerably better.

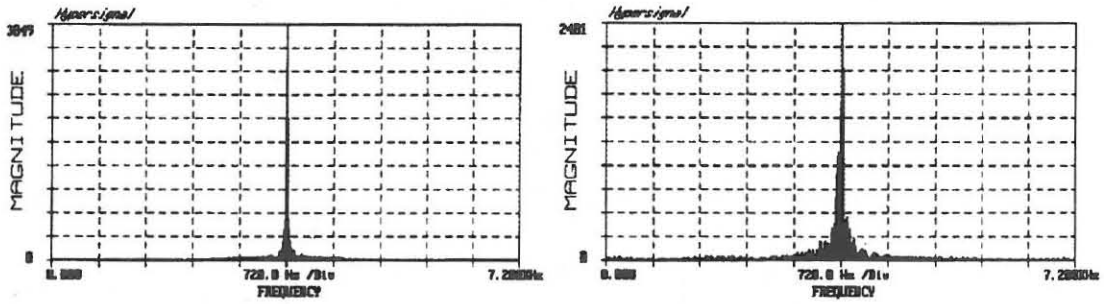


Figure 8.15: Spectral purity of recovered carrier wave with a pilot tone of (a) -19 dB and (b) -34 dB.

The acquisition and tracking characteristics of the carrier recovery scheme were also evaluated at different SNRs - with the pilot tone level at a fixed level of -14 dB. Figure 8.16 shows the average phase shift required after acquisition at different signal-to-noise ratios. From this it would appear that there is no substantial decrease in the quality of tracking with an in increase in noise.

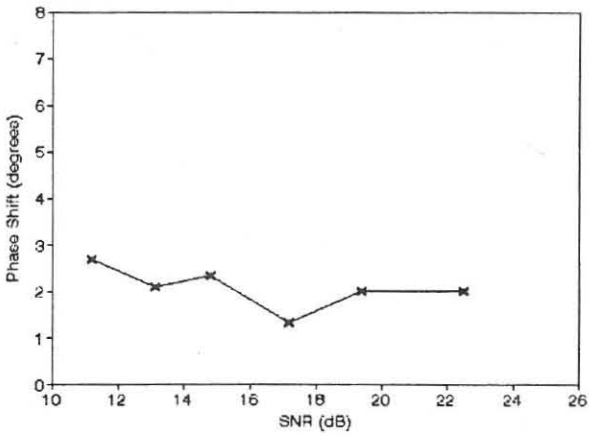


Figure 8.16: Plot of SNR against phase shift.

Figure 8.16 shows that with the pilot tone at an acceptably high level (higher than -35 dB), the carrier recovery system operated satisfactorily, even when the signal was severely degraded by noise. Only a very limited deterioration in the tracking performance of the system is observed with a decrease in SNR and this figure basically corresponds with the results shown in Figure 8.12.

8.4.2 PERFORMANCE OF CLOCK RECOVERY TECHNIQUE

The performance of the clock recovery technique was evaluated with respect to the following parameters, viz.:

1. Ability of clock recovery circuit to derive a timing signal of acceptable amplitude.
2. Timing accuracy of timing signal.

3. Constellation of demodulated data signal.

The following technique was followed during the evaluation of the proposed timing recovery technique:

1. A baseband 7PRS signal was transmitted, the timing signal recovered and the received data decoded in real time. The data was written to a data file and evaluated using Mathcad 6.0 and Hypersignal Plus..
2. The clock recovery software was integrated with the clock recovery and demodulation software. A 49QPRS signal was sampled with the *sample.pas* and *sample.asm* software, processed with the combined receiver software and the output written to a data file. The data file was then evaluated as described in (1) above.

The clock recovery circuit delivered a sine wave output of acceptable amplitude. Unfortunately, as can be seen in Figure 8.17, it showed some variation in amplitude, but not to any great extent. This situation might have been improved by using more taps on the output FIR filter in the clock recovery stage, but this would have led to a longer period of "fading in" of the sine wave output.

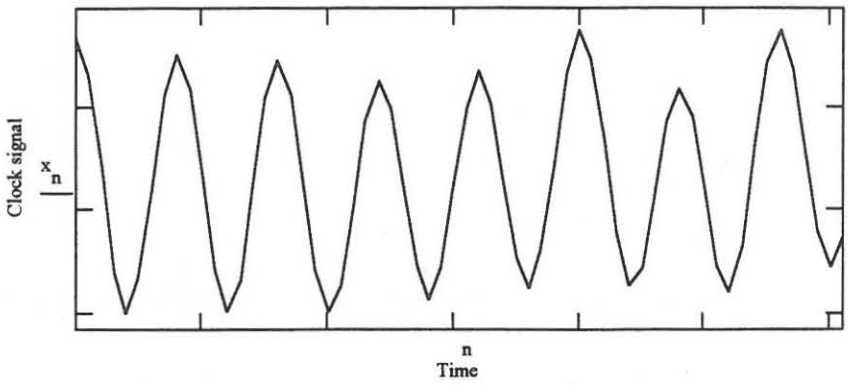


Figure 8.17: Sine wave output of clock recovery circuit.

This phenomenon is shown in Figure 8.18, where the time required by the clock signal to reach its final amplitude, can be seen.

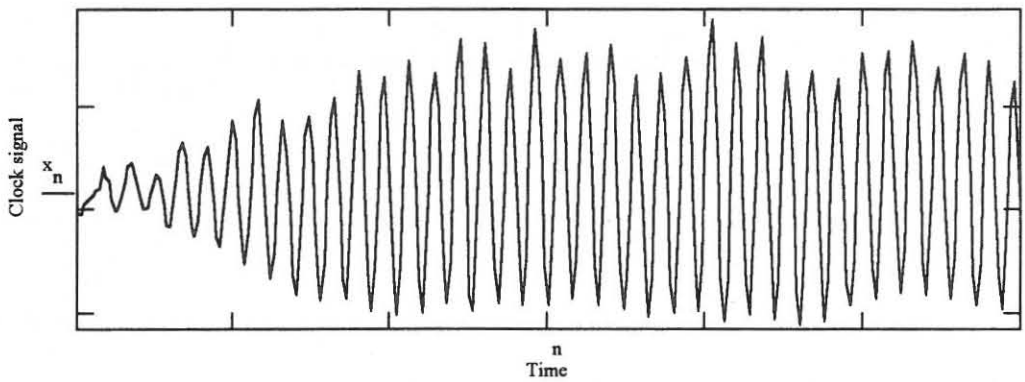


Figure 8.18: Fading in and variation in amplitude of the recovered clock signal.

The frequency of the clock signal is 2 400 Hz. This was verified by investigation of the signal, as well as by determining the cross-correlation between this signal and a simulated signal. This process was repeated for different lengths of signal, ranging from one cycle (i.e. eight samples) to eight cycles. Correlation coefficients of between 0,9 and 0,99 were

obtained when correlated with a standard 2 400 Hz sine wave of the correct phase (see Table 8.2). The output can therefore be considered to be a fair approximation of a sine wave at the required frequency.

By means of auto-correlation, the stability of the recovered signal was determined and correlation coefficients of between 0,97 and 0,988 were obtained. This indicated good stability with little jitter [28, p. 400].

Table 8.2: Correlation coefficients between recovered and simulated clock signals.

	Data Length		
	8 Symbols	**2 Symbols**	**1 Symbol**
Cross-correlation	0,900	0,988	0.989
Auto-correlation	0,979	0,985	0,988

It should be borne in mind that the signals were correlated at a limited number of different phase angles, which had a limiting effect on the maximum correlation coefficient values obtained. A typical output of the calculated correlation coefficients is shown in Figure 8.19. In this case, a zero degree phase shift - corresponding with the left hand side of the figure - indicated no phase difference between the simulated, ideal waveform and the recovered clock signal.

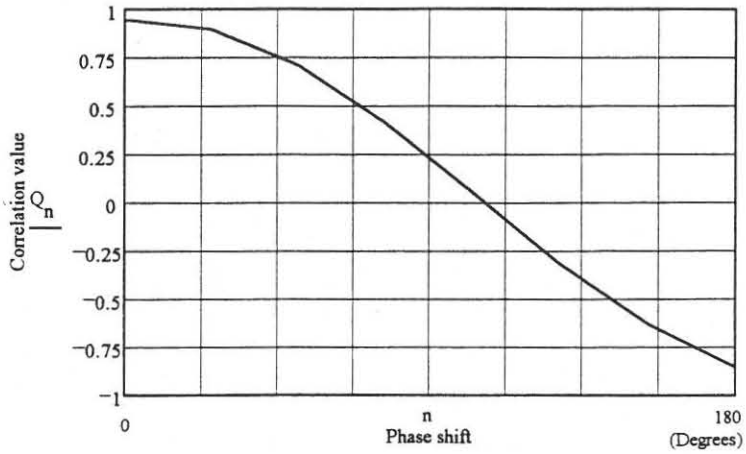


Figure 8.19: Cross-correlation coefficient between recovered clock signal and reference signal.

As shown in Figure 5.6, a peak detector detects the moment the recovered clock signal reaches its peak positive value. Figure 8.20 shows the AVR output (A_n), clock signal (B_n) and peak detector output (C_n) - from the top to the bottom of the figure. Immediately following the peak of the sine wave output, the peak detector delivers a pulse, identifying the optimum timing instant.

The relative timing between the peak detector output and the data symbols is shown in Figure 8.21.

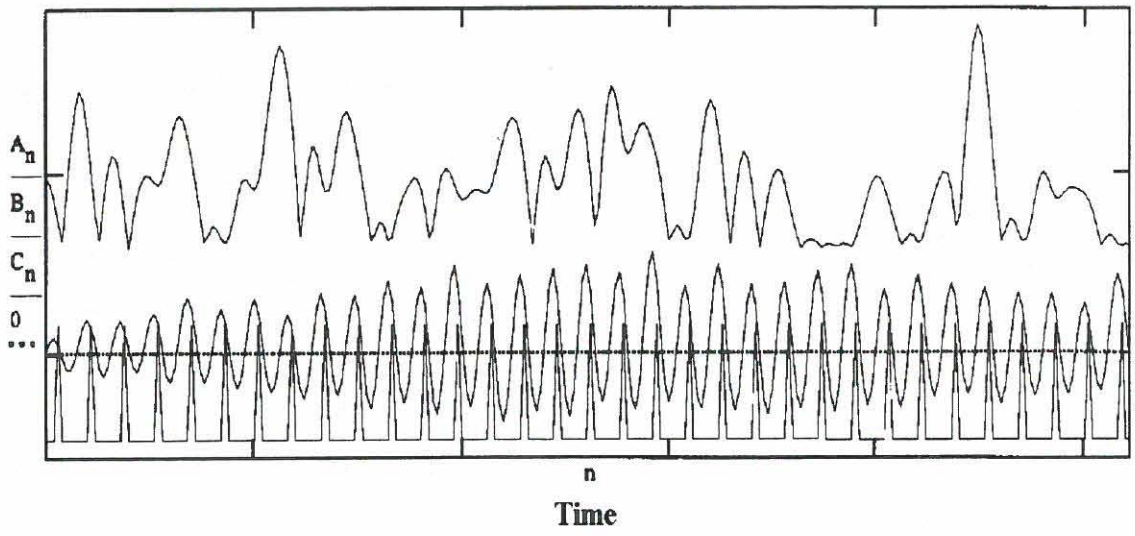


Figure 8.20: AVR output, clock signal and peak detector output.

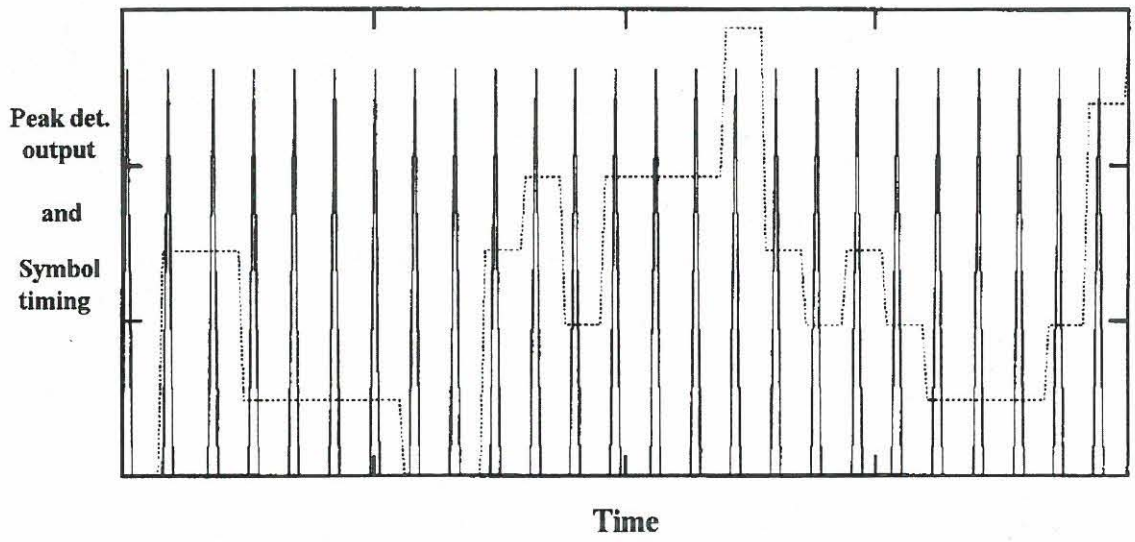


Figure 8.21: Relative timing of peak detector output and data symbols.

From Figure 8.21, it is clear that there is a fixed timing relationship between the peak detector output and the symbol timing.

Baseband 7PRS data sequences, as well as modulated 49QPRS signals, were transmitted and received by the receiver software. Figure 8.22 shows typical levels of the received 7PRS signals, as were recovered by making use of the recovered clock signal. The slight variation in the values of each level indicates the presence of a limited amount of ISI. By varying a variable in the clock recovery code, the actual timing instant can be optimised. Careful consideration of these levels enabled the setting of optimised decision levels in the decoding code.

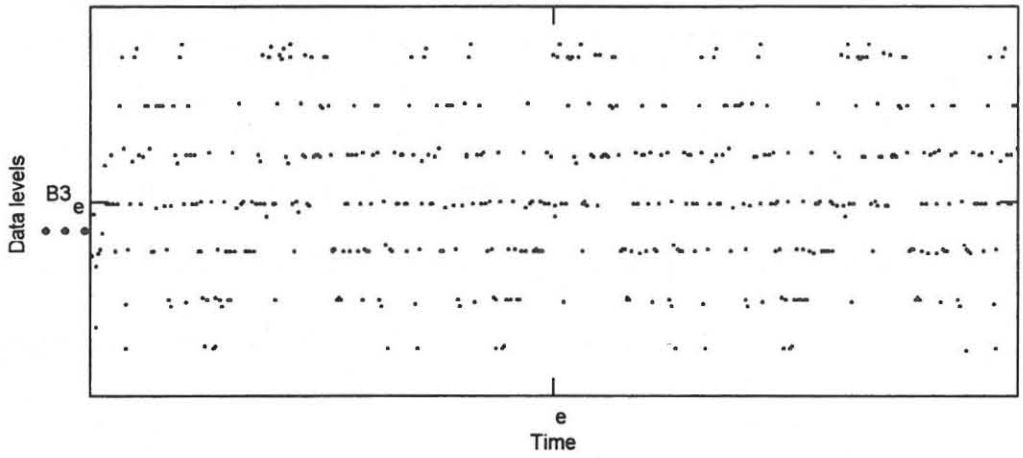


Figure 8.22: Levels of received 7PRS code.

The recovered signal constellation of the I- and Q-channels for a pseudo random data sequence is shown in Figure 8.23. Again, a moderate amount of ISI is in evidence, but the different levels are easily discernible.

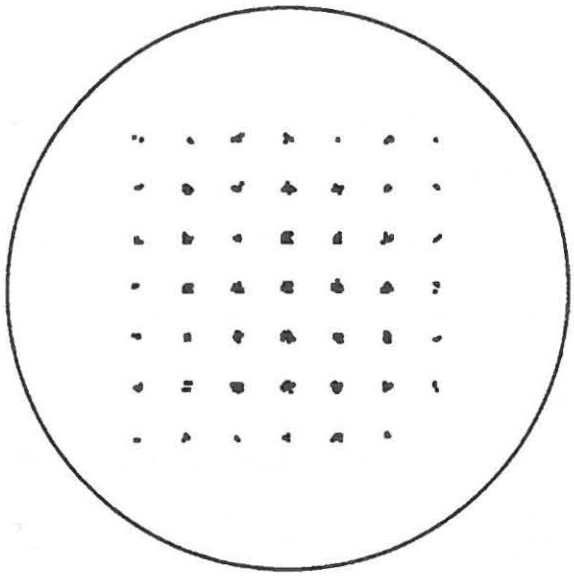


Figure 8.23: Signal constellation of 7PRS I- and Q-channels.

The levels, as shown in Figure 8.23, were decoded into dibits, resulting in dibit values as shown in Figure 8.24. This figure shows the I-channel during transmission of the learning sequence and the first data symbols.

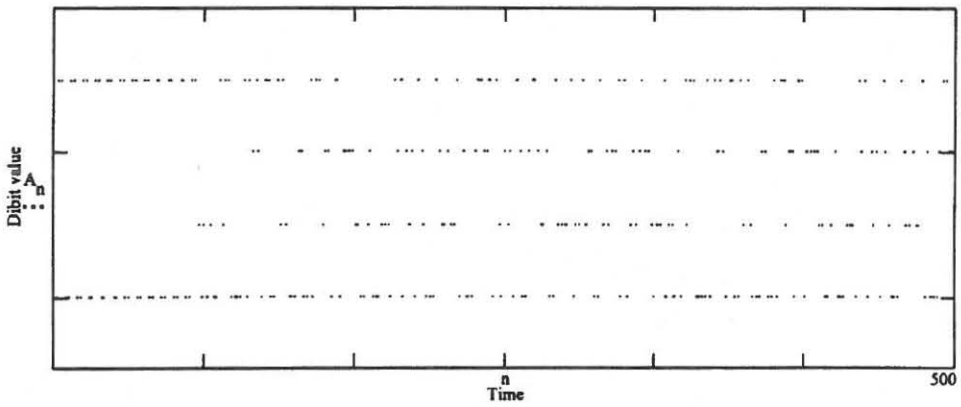


Figure 8.24: Decoded dibits of I-channel during the learning period and immediately afterwards.

8.5 SUMMARY

This concludes the evaluation of the carrier and clock recovery schemes as developed - and implemented in DSP - during the course of the project. By analysing the results as measured it was possible to come to a number of conclusions regarding the performance of the two synchronisation techniques as proposed. These conclusions are discussed in the next chapter.

CHAPTER 9

CONCLUSIONS

9.1 INTRODUCTION

Considering the evaluation of the transmission system as developed during the course of the project, a number of definite conclusions regarding the performance of the transmitter, as well as the carrier and clock recovery techniques, can be made.

After a thorough study of existing carrier and clock recovery techniques, and in particular closed-loop systems, it was decided to implement a carrier recovery system by making use of a pilot tone.

The exact operation of many non-linearities used in clock recovery systems has never been analysed mathematically. Due to this, extended simulations of a number of possible clock recovery techniques - making use of different non-linearities - were done in MathCAD. The simulations indicated that excellent results could be obtained from a clock recovery system which comprises a limited amount of processing before the baseband signal was passed through an AVR.

Through the simulations, it was also determined that making use of a pulse shaping filter, derived from the standard raised cosine response, resulted in a substantial increase in the

magnitude of a spectral component at the symbol frequency in the output of the AVR. This filter has a wider bandwidth than a standard RC filter, but still better results were obtained with the PSF than with the RCF at identical values of excess bandwidth.

9.2 TRANSMITTER

The correct operation of the data splitters, PS filters and PRS encoders in the transmitter was verified by inspection of intermediate results during simulations, and execution of the assembly language code. The frequency and relative phases of the pilot tone, in-phase carrier and quadrature carrier signals were also monitored in both the time and frequency domains. In each case, satisfactory results were obtained.

Software was developed to display the eye diagram of the transmitted baseband PRS signal. This diagram showed a wide eye opening with essentially no ISI at two specific instants during each symbol. This was an indication of a good phase response by the PSF.

The power spectrum of the 49QPRS signal has a spectral null at 1 200 Hz, a frequency corresponding to one third of that of the carrier signal. This was an ideal situation, which facilitated the easy implementation of the carrier recovery system when DSP techniques were used. The frequency spectrum of the modulated 49QPRS signal was therefore conducive to the use of a pilot tone in the carrier recovery scheme.

9.3 RECEIVER

The performance of the receiver was evaluated with respect to the clock and carrier recovery techniques, but not with respect to its BER.

9.3.1 REAL TIME IMPLEMENTATION OF RECEIVER

Transmitting a baseband 7PRS signal at a data rate of 4 800 bits per second, with a sampling rate of 19 200 samples per second, the baseband receiver was capable of recovering the clock signal in real time. The recovered clock was used to decode the sampled baseband signal, and the dibit values of the decoded 7PRS signal were written to a data file on the receiver host computer. Similarly, a carrier recovery circuit was capable of recovering the carrier in real time.

However, the execution speed of the SIG-56 board is of such a nature that the total communication system could not be implemented to operate in real time. In an effort to optimize the speed of the system, all the relevant code was written in assembly language, but even then, the required speed could not be attained. Accordingly, sampling software was used to sample the transmitted 49QPRS signal. The sampled values were written to a data file for eventual processing by the actual receiver code.

9.3.2 CARRIER RECOVERY USING A PILOT TONE

The carrier recovery scheme was evaluated relative to the following parameters:

1. Acquisition time.
2. The effect of the level of the pilot tone.
3. The noise immunity of the carrier recovery scheme.

The acquisition time was evaluated with the level of pilot tone at different values between -10 dB and -42 dB relative to that of the modulated signal. It was found that, irrespective of the level of pilot tone, phase synchronism was obtained within 625 μs after the beginning of reception. The same acquisition time was measured for any initial phase error.

Once synchronised, the receiver tracked the transmitter carrier closely and only very moderate phase adjustments were required to maintain synchronisation. However, it was found that the tracking was dependent on the level of pilot tone and that the phase adjustments became excessive with a pilot tone below approximately -35 dB relative to the peak modulated signal level. Use of such low levels of pilot tone should therefore be avoided.

Since carrier synchronisation involves the calculation of the cross-correlation between a regenerated signal and a series of values read from a built-in sinetable, special care was taken to maximise the execution speed of this part of the code.

The noise immunity of the system - with the pilot tone at a fixed level of -13 dB - was also determined. A large number of measurements were made at different SNRs, but no conclusive cut-off point with respect to this characteristic could be identified.

9.3.3 CLOCK RECOVERY MAKING USE OF AN AVR AND LIMITED PRE-PROCESSING

The clock recovery scheme was evaluated relative to the following parameters:

1. The ability of the clock recovery circuit to derive a timing signal of acceptable amplitude at all times.
2. The timing accuracy of the recovered clock signal.
3. The signal constellation of the demodulated data signal.

It was found that the recovered clock signal takes approximately 4,25 ms after the beginning of reception to reach the final amplitude. This is acceptable since the duration of the learning phases of such transmission systems normally far exceeds this value. Unfortunately there was a substantial variation in the amplitude of the clock signal. This

characteristic could probably have been improved by use of more taps in the FIR filter on the output of the AVR, but since the clock signal is not used for demodulation purposes, and since this would have increased the fading in time of this signal, it was decided not to increase the length of the filter.

Unfortunately, the jitter of the clock signal could not be measured directly, but by cross-correlating the recovered signal with an ideal, simulated clock signal, the properties of the recovered signal were evaluated. This was repeated by auto-correlating different sections of the recovered signal. The measurements indicated a stable clock signal with little jitter.

The relative timing between the timing signal and the symbol periods were also determined and satisfactory results were obtained. This was verified by inspection of the instantaneous values of the received 7PRS signals and again, very satisfactory results were obtained. Very little ISI was present in the received data signal.

9.4 GENERAL

From the above, it is obvious that both the proposed carrier and clock recovery techniques perform very well - as long as the level of the pilot tone is not too low. It is conceivable that characteristics of the PSF filter can still be improved upon by optimisation, making use of genetic algorithms [45, p. 67 and p. 68]. This might result in a decreased bandwidth, which in turn, might enable the use of a carrier frequency of only 2 400 Hz.

Making use of genetic algorithms in the design of the filter should not only result improved frequency characteristics, but also in the minimisation of computional complexity [59, p. 234]. Due to time constrains, it was not possible to evaluate this experimentally.

With the PSF, a 2 400 Hz carrier frequency cannot be attained since the lower sideband of the modulated signal would extend into the negative frequency range, causing severe distortion. Alternatively, ways to demodulate analytic signals successfully can also be investigated.

APPENDIX A

SIMULATION: RC AND PS FILTERS

The following is a MathCAD simulation of the filtering of a 7PRS signal using both RC and PS filters. From this it is possible to evaluate the relative frequency responses of the filters, as well as the pulse shape of a filtered 7PRS signal.

GENERATE 7PRS SIGNAL

$k := 128$ --Generation of pseudo random data

$l := 1 .. k$

$B1_l := \Phi(\mathrm{rnd}(1) - 0.5)$

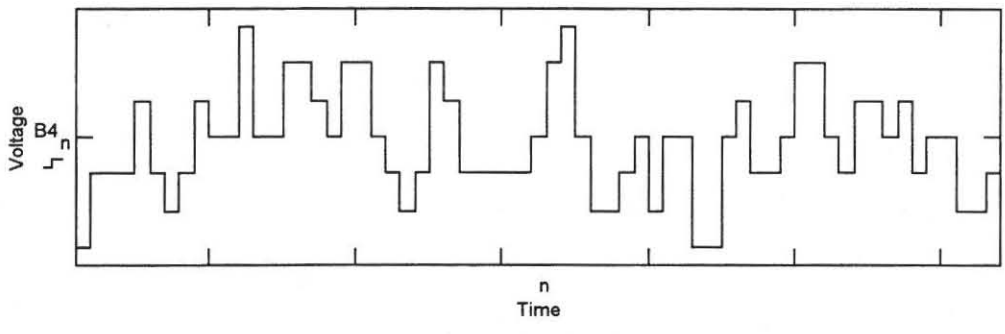
$n := 1 .. \frac{k}{2}$ --Compilation of dibits (0 to 3)

$B2_n := B1_{2 \cdot n - 1} \cdot 2 + B1_{2 \cdot n}$

$B3_1 := 0$

$B3_n := B2_n - B3_{n-1} + \Phi\left[-\left(B2_n - B3_{n-1} + 0.5\right)\right] \cdot 4$ --Precoding

$B4_1 := 0 \quad B4_n := B3_n + B3_{n-1} - 3$ --7PRS signal



7PRS SIGNAL

DERIVE SAMPLED VALUES OF 7PRS SIGNAL:

$z := 8$ --8 Samples per bit

$I := 0, 8 .. \left(\frac{k-1}{2}\right) \cdot z$

$B5_I := B4_{\frac{I}{8}}$

$B5_{I+1} := B5_I$

$B5_{I+2} := B5_I$

$B5_{I+3} := B5_I$

$B5_{I+4} := B5_I$

$B5_{I+5} := B5_I$

$B5_{I+6} := B5_I$

$B5_{I+7} := B5_I$

SIMULATION OF RAISED COSINE FILTER

$b := \frac{k}{2}$

$f := \frac{1}{2}$

$T := \frac{1}{f}$

$W := \frac{\pi}{T}$

$\alpha := 0.2$

$a := b \cdot z - 1$

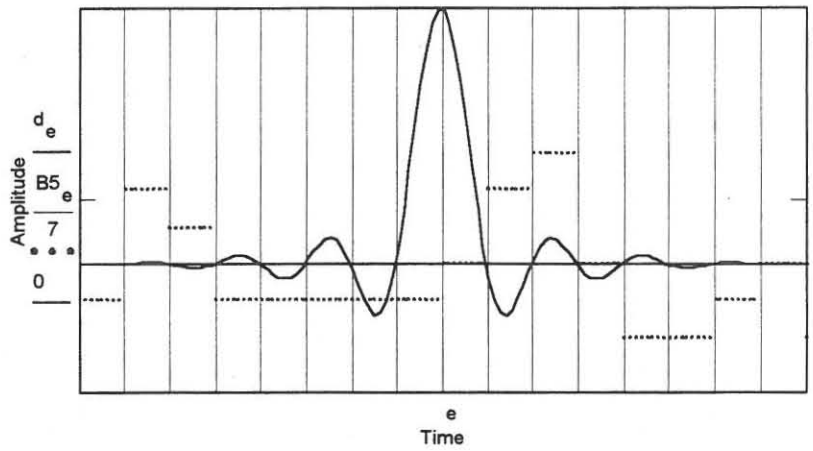
$e := 0 .. a$ --Note: f=sampling freq, W=π/T

$c := \frac{b \cdot T}{a}$

$t_e := \left(e \cdot c - \frac{c \cdot a}{2}\right)$ --Relative sampling instant

$$d_e := \left(\frac{\sin\left(W \cdot t_e\right)}{W \cdot t_e}\right) \cdot \left[\frac{\cos\left(\alpha \cdot W \cdot t_e\right)}{1 - \left(2 \cdot \alpha \cdot W \cdot \frac{t_e}{\pi}\right)^2}\right]$$

--Time response of RC filter



STEP RESPONSE OF RAISED COSINE FILTER

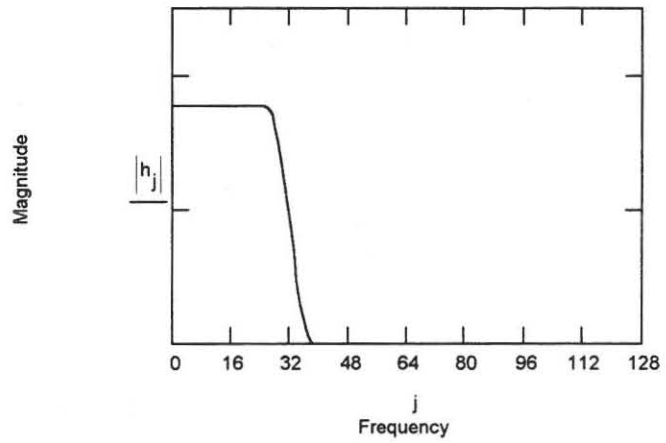
$h := \text{fft}(d)$ --Convert RC filter response into

$i := \text{last}(h)$ frequency domain.

$j := 0 .. i$

$r := \text{fft}(B5)$

$s_j := h_j \cdot r_j$ --Filtering of 7PRS signal



FREQ. RESPONSE - RC FILTER

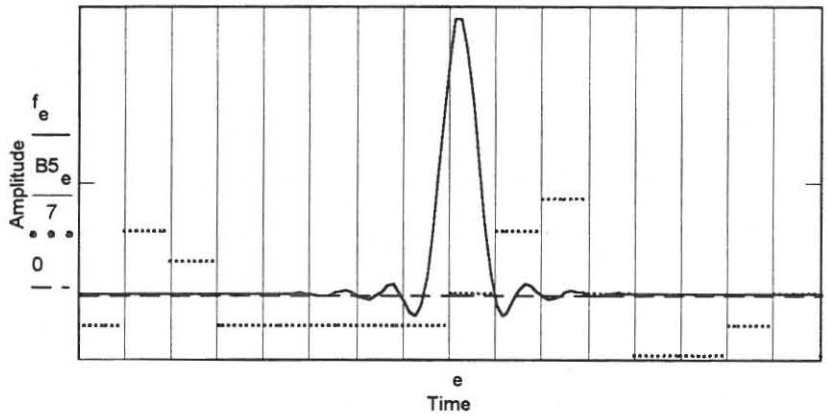
SIMULATION OF PSF FILTER AND FILTERING OF 7PRS SIGNAL

$e := 4 .. a$

$u_e := t_e$

$$f_e := \left(\frac{\sin\left(2\cdot W\cdot u_e\right)}{2\cdot W\cdot u_e}\right)\cdot\left[\frac{\cos\left(2\cdot\alpha\cdot W\cdot u_e\right)}{1-\left(4\cdot\alpha\cdot W\cdot\frac{u_e}{\pi}\right)^2}\right]+\left(\frac{\sin\left(2\cdot W\cdot u_{e-4}\right)}{2\cdot W\cdot u_{e-4}}\right)\cdot\left[\frac{\cos\left(2\cdot\alpha\cdot W\cdot u_{e-4}\right)}{1-\left(\frac{4\cdot\alpha\cdot W\cdot u_{e-4}}{\pi}\right)^2}\right]$$

--This is the time response of the PSF.

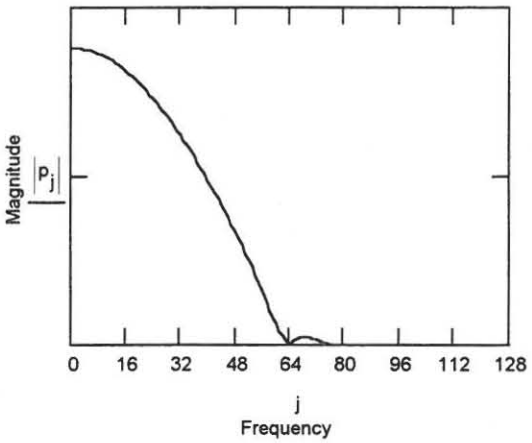


STEP RESPONSE - PSF FILTER

$p := fft(f)$ --Convert PSF response into frequency domain.

$r2 := fft(B5)$

$s_j := p_j \cdot r2_j$



FREQ. RESPONSE OF PSF FILTER

From the frequency responses of the RC and PS filters, as shown above, it is obvious that a PSF occupies a bandwidth considerably in excess of that of the RC filter, however its bandwidth requirements is significantly less than that of a 2W RC filter.

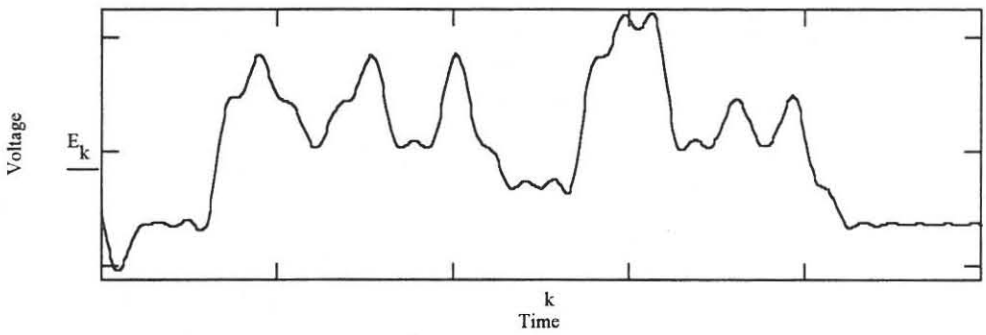
APPENDIX B

SIMULATION: REGENERATION OF TIMING SIGNAL

This is a MathCAD simulation of the AVR-LP clock recovery technique developed during execution of this project. A PSF filtered 7PRS signal is used as input (E_k below).

$D_k := D_{k+\left(\frac{b \cdot z}{2}\right)}$

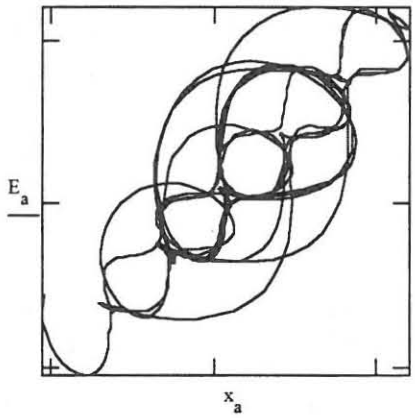
$E_k := 2 \cdot D_k$ --Demodulated 7PRS Signal



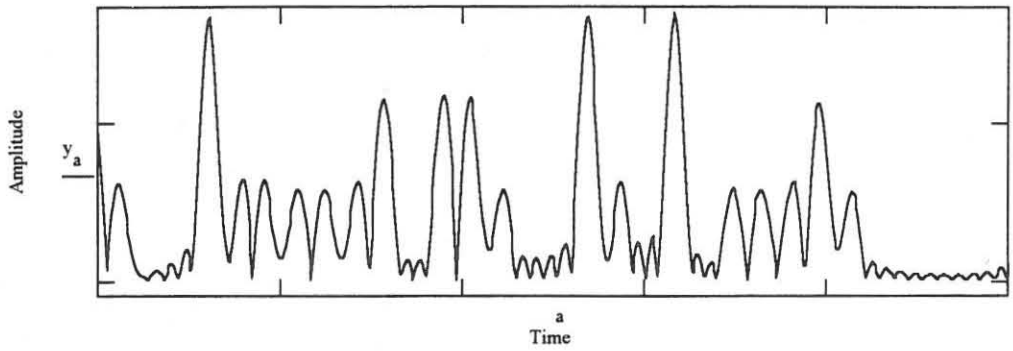
$a := 0 .. 1023$

$x_a := E_{a+8}$

$y_a := |x_a - E_a|$



PREPROSESSING FILTER OUTPUT



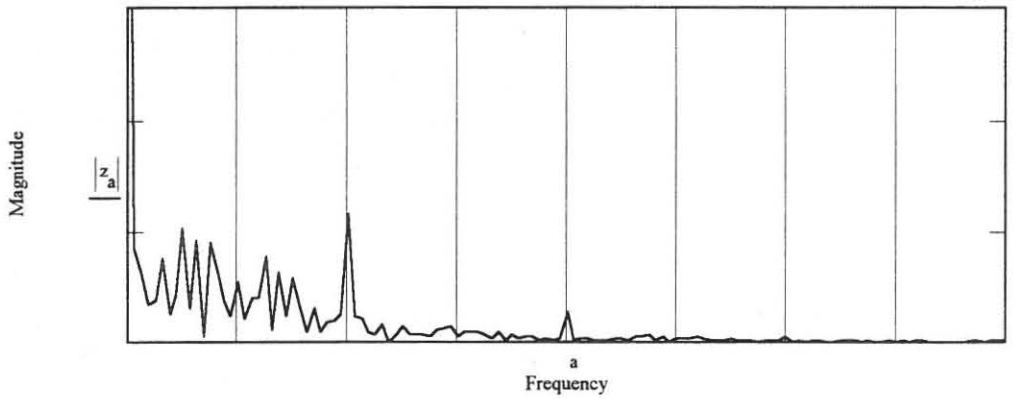
OUTPUT OF AVR

$a := 0..511$

$b_a := y_a$

$z := fft(b)$

$a := 0..128$



FREQ. RESPONSE - PROCESSED DATA

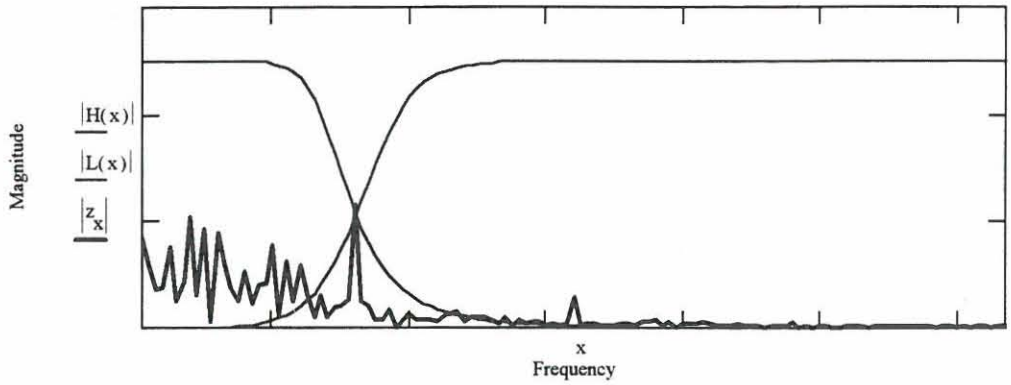
The above frequency response includes a significant spectral component at the fundamental frequency of the clock signal. This component can be bandpass filtered and used to drive the clock circuit.

$x := 1..256$

$$H(x) := \frac{1}{\sqrt{1+\left(\frac{2\cdot\pi\cdot 5.8}{x}\right)^{2\cdot 6}}}\cdot 5$$

--Bandpass filtering of processed data

$$L(x) := \frac{1}{\sqrt{1+\left(\frac{x}{2\cdot\pi\cdot 4.5}\right)^{2\cdot 6}}}\cdot 5$$



BANDPASS FILTERING - CLOCK FREQ.

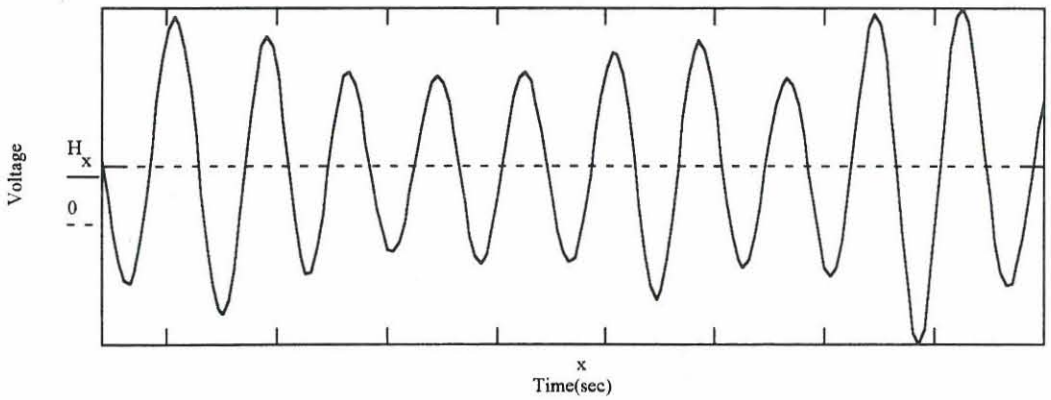
The combined response of the filters simulated above enables the recovery of the clock signal.

$R_x := z_x \cdot L(x) \cdot H(x)$

$x := 0..256$

$H := ifft(R)$

$p := 0..511$



CLOCK FREQUENCY

Unfortunately the recovered clock signal has a moderate amount of variation in amplitude. This would normally cause jitter. However, since the use of a PSF results in a very wide eye opening this characteristic is quite acceptable.

APPENDIX C

PILOT TONE BANDPASS FILTER

Since the filter has a length of 180 taps, only the specifications thereof, as well as the first 10 coefficient values, are given. This data is in the form as prepared by Hypersignal Plus during the design of the filter.

In designing the filter a compromise had to be reached between an even better frequency characteristic and an excessively long filter. A length of 180 taps provided an attenuation of 25dB (which was considered to be acceptable), with a bandwidth of 200 Hz.

BP

180
1.92000000000000E+0004
FILENAME: pilot.FIR
DESIGN: kw
SAMPLING FREQUENCY (Hz): 19200
CENTER FREQUENCY (Hz): 1200
BANDWIDTH (Hz): 50
TRANSITION BANDWIDTH1 (Hz): 100
TRANSITION BANDWIDTH2 (Hz): 100
STOPBAND ATTENUATION (dB): 25
FIR LENGTH/IIR ORDER: 180
RIPPLE (dB)/STEIGLITZ ERR: 1

FILTER TYPE (LP/HP/BP/BS): BP

CONSTANT MULTIPLIER C:

1.00000000000000E+0000

NUMBER OF NUMERATOR (ZERO) FACTORS:

1

ORDER OF ZERO FACTOR (SECTION) 1:

180

A[0] (z** -0)=

-1.71204515438793E-0003

A[1] (z** -1)=

-2.13867291471384E-0003

A[2] (z** -2)=

-2.26153242605626E-0003

A[3] (z** -3)=

-2.02449307876836E-0003

A[4] (z** -4)=

-1.42644964496455E-0003

A[5] (z** -5)=

-5.27508673975278E-0004

A[6] (z** -6)=

5.54824562387685E-0004

A[7] (z** -7)=

1.65979381602659E-0003

A[8] (z** -8)=

2.60641853548277E-0003

A[9] (z** -9)=

3.22224671938326E-0003

The values of the coefficients were adjusted before integration into the assembly language programme.

APPENDIX D

PROGRAMME: EYE DIAGRAM

This programme provides an easy means to display the eye diagram of a baseband 7PRS signal as transmitted by the DSP transmitter. It was important to determine the amount of ISI caused by the PSF and this measurement could be made without any problems in this manner. The received signal is, without any processing, written as a text file which is then processed using this software.

```
PROGRAM PlotEyeDiagram;

{*************************************************************************
 * This programme plots  an  eye diagram of a data signal by using a text file as input. *
 * The input file must have an amplitude in excess of 2000 units.                        *
 *************************************************************************}


USES
 Crt,Dos,Graph;

VAR
 I        : Integer;                  {For loop counter var}
 Data     : Array[1..4096] of Real;       {Data array of input signal}
 Plot     : Array[1..4096] of Integer;    {Converted data for plot}
 Points   : Integer;                  {Number of points - input sig.}
 WindowS  : Integer;                    {Window size}
 Samples  : Integer;                   {Sample time}
 Lines    : integer;                  {Number of lines to plot}
 Del      : Integer;                  {Delay time var}
 ColourChar : Char;                     {Colour decision char}
 FileName : String;                    {Input file name}
```

```
PROCEDURE InitGraphics;
{*************************************************************************
 * Initializes the screen for graphics mode                              *
 *************************************************************************}

VAR
  Driver, Mode, ErrCode, LowMode, HighMode : Integer;

BEGIN
  Driver := Detect;
  InitGraph(Driver,Mode,'c:\tp');
  ErrCode := GraphResult;
  If ErrCode <> grOK then
  Begin
    ClrScr;
    Writeln(^g);
    Writeln('Graphics ERROR : ',GraphErrorMsg(ErrCode));
    Readln;
    Halt(1);
  End;
  GetModeRange(Driver,LowMode,Highmode);
  SetGraphMode(Highmode);
End;

FUNCTION Num2Str(i:longint) : String;
{*************************************************************************
 * Function to convert numbers to strings for output of integers in graphics mode.  *
 * ***********************************************************************}

VAR
  S : String;

BEGIN
  Str(i,S);
  Num2Str := S
```

```
End;


PROCEDURE Plotit;
{*************************************************************************
 * Plots the data collected by the LoadData procedure                    *
 *************************************************************************}

VAR
 X       : Integer;              {X-axes pointer}
 Loop    : Integer;               {Loop var}
 Counter : Integer;                {Counter var}
 XJump   : Integer;                {Sample jump var}
 Colour  : Integer;               {Colour code var}

BEGIN
 Counter := 1;
 InitGraphics;
 SetTextStyle(1,0,3);
 SetColor(White);
 OutTextXY(220,5,'EYE DIAGRAM');
 SetColor(Yellow);
 SetTextStyle(1,0,1);
 OutTextXY(400,11,'File : '+Filename);
 SetTextStyle(2,0,5);
 OutTextXY(10,455,'Points : '+Num2Str(Points));
 OutTextXY(130,455,'Window : '+Num2Str(WindowS));
 OutTextXY(240,455,'Samples p/s : '+Num2Str(Samples));
 OutTextXY(390,455,'Lines : '+Num2Str(Lines));
 OutTextXY(500,455,'Delay : '+Num2Str(Del));
 SetColor(White);
 MoveTo(0,240);
 PutPixel(1,Plot[1],White);
 XJump := 640 div WindowS;
 X := 1;
 For i:= 2 to WindowS do
```

```
Begin
  X := X+XJump;
  Delay(Del);
  LineTo(X,Plot[i]);
End;
Colour := 0;
For Loop:= 1 to Lines-1 do
Begin
  Colour := Colour + 1;
  If Colour > 15 then
    Colour := 1;
  X := 0;
  If UpCase(ColourChar) = 'J' then
    SetColor(Colour);
  Counter := Counter + Samples;
  Moveto(0,0);
  For i:= Counter to Counter+WindowS-1 do
  Begin
    LineTo(X,Plot[i]);
    Delay(Del);
    X := X + XJump;
  End;
End;
SetColor(Red);
Line(640,240,0,240);
Moveto(0,0);
Lineto(GetMaxX,0);
Lineto(GetmaxX,GetMaxY);
Lineto(0,GetmaxY);
Lineto(0,0);
Readln;
End;
```

```
PROCEDURE LoadData;
{************************************************************************
 * Procedure to input the signal data from a file.  The input values are also scaled to *
 * fit onto the screen.                                                                 *
 ************************************************************************}

VAR
 F          : Text;                  {Text file pointer}
 MaxValue   : Real;                    {Max malue of input signal}
 ScaleVal   : Real;                   {Scale value for input signal}
 Resolution : Integer;                 {Size of screen pointer}
 RealVal    : Integer;
 YVal       : Integer;

BEGIN
 MaxValue := 0;
 ClrScr;
 Write('Enter the data file         : ');
 Rcadln(FileName);
 Write('Number of points ?          : ');
 Readln(Points);
 Write('Windows Size ? (1..640)     : ');
 Readln(WindowS);
 Write('Samples per symbol ?        : ');
 Readln(Samples);
 Write('Number of lines ? (1..',(Points div Samples)-8,')   : ');
 Readln(Lines);
 Write('Enter draw delay            : ');
 Readln(Del);
 Writeln('Multiple colours ? (J/N) ');
 ColourChar := Readkey;

 Assign(F,FileName);
 Reset(F);
 For I := 1 to Points do
 Begin
```

```
  Readln(F,Data[i]);
  If ABS(Data[i]) > MaxValue then
    MaxValue := ABS(Data[i]);
 End;
 Close(F);
 Writeln;
 Resolution := 200;
 ScaleVal   := MaxValue/Resolution;
 If MaxValue > Resolution then
 Begin
  For i := 1 to Points do
  Begin
    Data[i] := Data[i]/ScaleVal;
  End;
 End;
 For i := 1 to Points do
 Begin
  If Data[i] >= 0 then
  Begin
    RealVal := 200;
    YVal    := 40;
    While RealVal <> Round(Data[i]) do
    Begin
      RealVal := RealVal - 1;
      YVal    := YVal    + 1;
    End;
    Plot[i] := YVal;
  End
  Else
    Plot[i] := Round(ABS(Data[i])) + 240;
 End;
End;


Begin
Repeat
```

```
 LoadData;
 Plotit;
 CloseGraph;
Until keypressed;
End.
```

APPENDIX E

PROGRAMME: PSF COEFFICIENTS

This programme calculates three sets of coefficients for filters according to specifications submitted by the user. The filters are as follows:

- A standard raised cosine filter.
- A 2W raised cosine filter.
- A PSF filter.

The operator is prompted for the following characteristics:

- The symbol width of the filter.
- The number of points per symbol.
- The excess bandwidth, α.
- A factor by which the calculated coefficients are multiplied.
- The name of the file in which the coefficients are to be saved.

```
PROGRAM RCF_Generator;
{*********************************************************************
 * This programme generates the coefficients of a raised cosine      *
 * filter.  The coefficients are also converted into different formats to realize *
 * different filter characteristics. The following files are saved:  *
 *      <FILENAME>.NOR : Normal X-point RC filter with a length of Y-symbols *
 *      <FILENAME>.HLF : Every second coefficient of the above filter *
 *      <FILENAME>.MUL : PS filter with X-points and length Y/2-symbols *
 *********************************************************************}

USES
  Crt, Dos;

TYPE
  Pointer  = Array[0..1024] of Real;
```

```
Pointer2 = Array[0..1024] of Integer;

PROCEDURE Gen_Raised_Cosine;
VAR
 Value      : Pointer;          {Calculated real filter value}
 NewValue   : Pointer2;         {Integer filter value}
 TwoFil1    : Pointer2;
 TwoFil2    : Pointer2;
 Amount_P   : LongInt;          {Number of points}
 Mul        : Integer;          {Multiply factor}
 SymbolW    : Integer;          {Symbol width}
 Points_P_S : Integer;          {Points per symbo0l}
 Skip       : Integer;          {Skip for every second point}
 TempInt    : Integer;          {Temperary integer}
 i          : Integer;          {Counter integer}
 Alpha      : Real;             {Alpha value for filter}
 Step       : Real;             {Step size}
 FileName   : String;           {Output file name}
 OutFile    : Text;             {Output file pointer}


BEGIN
 ClrScr;
 Write('Enter the symbol width of the filter : ');
 Readln(SymbolW);
 Write('Enter the points per symbol        : ');
 Readln(Points_P_S);
 Write('Enter the value of alpha           : ');
 Readln(Alpha);
 Write('Enter the multiply factor          : ');
 Readln(Mul);
 Write('Enter the output file name         : ');
 Readln(FileName);
 Writeln; Writeln;
 Writeln('Please wait...');
 Writeln; Writeln;
```

```
Step      := pi/Points_P_S;
Amount_P  := Points_P_S * SymbolW div 2;
Value[0]  := 1;
For i:= 1 to Amount_P do
  Value[i] := (Sin(i*Step)/(i*Step))
            *(cos(Alpha*i*Step)/(1-sqr(2*Alpha*i*Step/pi)));
Assign(OutFile,FileName+'.nor');
ReWrite(OutFile);
For i := Amount_P downto 0 do
  Writeln(OutFile,Round(Value[i]*mul));
For i := 1 to Amount_P-1 do
  Writeln(OutFile,Round(Value[i]*mul));
Close(OutFile);
Assign(OutFile,Filename+'.nor');
Reset(OutFile);
Skip := 0;
For i:= 1 to Amount_P*2 do
  Readln(OutFile,NewValue[i]);
Close(OutFile);
Assign(OutFile,Filename+'.hlf');
ReWrite(OutFile);
For i:= 1 to Amount_P do
Begin
  Skip := Skip + 2;
  TwoFil1[i] := NewValue[Skip];
  Writeln(OutFile,TwoFil1[i]);
End;
Close(OutFile);
For i:= 1 to Amount_P do
  TwoFil2[i+4] := TwoFil1[i];
For i:= 1 to 4 do
  TwoFil2[i] := 0;
Assign(OutFile,Filename+'.mul');
ReWrite(OutFile);
For i:= 1 to Amount_P do
Begin
```

```
    TempInt := TwoFil1[i] + TwoFil2[i];
    Writeln(OutFile,TempInt);
  End;
  Close(OutFile);
  Writeln;
  Writeln(Filename+'.NOR',' : Normal filter with ',Points_P_S,' points and ',SymbolW,' symbols');
  Writeln(Filename+'.HLF',' : Every second point of the filter');
  Writeln(Filename+'.MUL',' : PSF ',Points_P_S,' point filter with ',SymbolW div 2,' symbols');
  Readln;
End;

BEGIN
  Gen_Raised_Cosine;
END.
```

APPENDIX F

PROGRAMME: DATA TRANSMITTER

The data transmitter software consists of two programmes, viz. %TX4.PAS and %TX4.ASM. These programmes are listed below. The basic function of the programmes is described in the first lines of each programme.

%TX4.PAS

```
PROGRAM Transmit_Data_TO_AIC;

{*************************************************************************
 * This programme reads the coefficients of a pulse shaping filter and a long integer    *
 * data file (to be transmitted) into memory. The %TX4.ASM file is then activated and *
 * data is transmitted through the DSP card's AIC. Any of the following signals can be  *
 * made to appear on the output - by editing the %TX4.ASM program:                      *
 *          - I-Channel baseband                                                        *
 *          - Q-Channel baseband                                                        *
 *          - Modulated I-Channel                                                       *
 *          - Modulated Q-Channel                                                       *
 *          - The 49-QPRS signal                                                        *
 *          - Pilot tone                                                                *
 *************************************************************************}

USES
 Crt, DSPlib4;

VAR
 datain              : array[1..511] of longint;
 filterin            : array[1..256] of longint;
 baseaddress, timeout    : word;
 Infile              : text;
 i,a                 : Integer;
 Length              : Integer;
 temp                : Longint;
```

```
 DataFile          : Text;

BEGIN
 timeout    := 10000;                {Time before error is shown}
 baseaddress := $2b0;                {Base address of the DSP card}
 DSP_Reset(BaseAddress);
 DSP_Load(BaseAddress,'%tx4');          {Load %tx4.asm into SIG-56 board}
 DSP_Go(BaseAddress);                 {Start the DSP card}

 Assign(Infile,'e:\f128n.mul');         {PS Filter file name}
 Reset(Infile);
 For i := 1 to 128 do
  Readln(Infile,filterin[i]);          {Read filter coefficients into array}
 Close(Infile);

 ClrScr;
 Write('Loading data ');
 Assign(DataFile,'e:\lint.io');
 Reset(DataFile);
 Length:=500;
 For i:= 1 to Length do               {Read data data into array}
  Read(DataFile,datain[i]);
 Close(DataFile);

 For i := 1 to 128 do
 Begin
  DSP_WriteFlag(baseaddress,timeout);      {Check if DSP card is ready}
  DSP_WriteLongInt(baseaddress,filterin[i]); {Write filter coefficients to DSP board}
 End;

 Writeln; Writeln;
 Writeln('Transmitting...');
 For i := 1 to 3000 do                {6 times length of data file}
 Begin
  DSP_WriteFlag(baseaddress,timeout);      {Check if DSP card is ready}
  DSP_WriteLongInt(baseaddress,datain[i]);  {Write data to DSP card}
```

```
End;
Writeln('.....END OF TRANSMISSION');

END.
```

%TX4.ASM

```
;*************************************************************************
;*          REAL-TIME 49QPRS TRANSMITTER - TO THE AIC                    *
;*************************************************************************
;* PROGRAMME :  Transmitter of a 49QPRS signal.                          *
;*          Reads long integer values from the host PC and converts these values into *
;*          binary. The binary values are then encoded into two 7PRS signals (an I    *
;*          and Q-channel). These two channels are then filtered by a PSF and         *
;*          modulated in quadrature. The modulated channels are summated to give a    *
;*          49QPRS signal. A pilot tone is added before transmission.                 *
;*          Transmission of data is preceded by a learning phase.                     *
;*************************************************************************

;*************************************************************************
;* USE OF MEMORY AND INTERNAL REGISTERS**                                *
;*                                                                       *
;;*     r0    -     Pulse shaping filter coefficient table              *
;*                 Memory              : x:$1000..$1080                  *
;*                 Base address        : $1000                           *
;*                 Modifier value  m0  : $7f or 127                      *
;*                 Offset value    n0  : 8                               *
;*                                                                       *
;*     r4    -     Circular buffer 7PRS data file                        *
;*                 Memory              : y:$0000..$0020                  *
;*                 Base address        : $0000                           *
;*                 Modifier value  m0  : $1f or 31                       *
;*                 Offset value    n0  : Not used                        *
;*                                                                       *
;*     r2    -     Sine look-up table                                    *
```

```
;*              Memory              : y:$0100..$0200                    *
;*              Base address        : $0100                             *
;*              Modifier value  m2  : $ff or 255                        *
;*              Offset value    n0  : $30 or 48                         *
;*                                                                      *
;*    r6    -   Cosine look-up table file                               *
;*              Memory              : y:$0100..$0200                    *
;*              BaseAddress         : $0140                             *
;*              Modifier value  m6  : $ff or 255                        *
;*              Offset value    n6  : $30 or 48                         *
;*                                                                      *
;*                                                                      *
;* The following address registers were used as counters :              *
;*                                                                      *
;*    r1    -   Pointer for pilot tone.                                 *
;*              Length of jump (n1) : 16                                *
;*                                                                      *
;*    r5    -   Counter for long integer read                           *
;*              Number of counts    : 24                                *
;*                                                                      *
;*    r7    -   Counter for symbol calculation                          *
;*              Number of counts    : 8                                 *
;*                                                                      *
;*    r3    -   Learning phase pointer                                  *
;*                                                                      *
;**********************************************************************

                opt     fc,mu,s,w,mex


buffmod     equ     31          ;7PRS circular data buffer modules
buffleng    equ     32          ;7PRS circular data buffer length
lumod       equ     127         ;Filter look-up table  modules
luleng      equ     128         ;Filter look-up table length
lupps       equ     8           ;Filter points per symbol
symbols     equ     16          ;Number of symbols
```

```
tempI       equ    $050          ;Temporary channel I value
tempQ       equ    $051          ;Temporary channel Q value
store       equ    $052          ;Temporary binary value
mask        equ    $053          ;Saves the AIC mask value
datain      equ    $054          ;Stores the input longint

HSR         equ    $ffe9         ;Host status register
HDR         equ    $ffeb         ;Host data register

CRA         equ    $ffec         ;Control register A
CRB         equ    $ffed         ;Control register B
PCC         equ    $ffe1         ;Port C control register
PCDDR       equ    $ffe3         ;Port C direction register
TX          equ    $ffef         ;Transmit memory location
TEST        equ    $ffee         ;DAC Ready register


            org    p:$0000
            jmp    AICSetup

            org    p:$0080

AICSetup
;*************************************************************************
;* Set the analogue interface circuit (AIC) for a sampling frequency of 19.2 kHz      *
;* by loading B counter with the value of 27.  To get a switched cap filter (SCF)     *
;* frequency of 518 kHz counter A should be set to 5. O/P filter f0 = +-6,66kHz.      *
;*                                                                                    *
;* Calculations :                                                                     *
;*                        Master Clock Freq.      5,184MHz                            *
;*  SCF Clock Freq. = ----------------------- = ------------ = 518,4kHz               *
;*                          2 X Counter A           2 X 5                             *
;*                                                                                    *
;*                       SCF Clock Frek.     518,4kHz                                 *
;*  Sampling Freq.  = ------------------ = --------------- =      19,2kHz              *
;*                          Counter B              27                                 *
```

```
;*********************************************************************

                ori    #%00000100,omr   ;Enable onboard ROM
                movep  #$4000,x:CRA     ;Setup SSI control register A
                movep  #$3A00,x:CRB             ;Setup SSI control register B

                movep  #$0001ff,x:PCC   ;Setup Port C control register
                movep  #$010e,x:PCDDR   ;Setup Data direction register
                clr    a
                move   a,x:$ffff        ;Disable interrupts
                movep  #$ffff1e,y:$ffe0 ;Set AIC master freq

                move   #$fffc00,b1      ;Setup AIC mask for 14 bit output
                move   b1,x:mask
                clr    b


;*** Primary transmission for Timer A

                move   #$ffffff,a1      ;Maximum value for a
priTA
                btst   #6,x:TEST        ;Check if DAC ready
                jcc    priTA            ;If not goto priTA
                movep  a1,x:TX          ;Transmit max value $ffffff


;*** Secondary transmission for Timer A

                move   #$0a0000,b1      ;Set TA = 5
secTA
                btst   #6,x:TEST        ;Check if DAC ready
                jcc    secTA            ;If not goto secTA
                movep  b1,x:TX          ;Transmit setup value TA=5


;*** Primary transmission for Timer B
```

```
priTB
                btst    #6,x:TEST          ;Check if DAC ready
                jcc     priTB              ;If not goto priTB
                movep   a1,x:TX            ;Transmit max value $ffffff


;*** Secondary transmission for Timer B

                move    #$360200,b1        ;TB = 27 for 19200 Hz sampling
secTB
                btst    #6,x:TEST          ;Test if DAC ready
                jcc     secTB              ;If not goto secTB
                movep   b1,x:TX            ;Transmit setup value TB=27


;*********************************************************************
;* Initialize the DSP :                                              *
;*     1) Setup the address register, offset register and the modifier registers *
;*     2) Resetting of temporary memory locations                    *
;*     3) Set the carrier frequency to 3 600 Hz                      *
;*        This is achieved as follows:                               *
;*                                                                   *
;*     Sampling Freq.      19200 Hz                                  *
;*     ----------------- = ----------- = 5,33                        *
;*      Carrier Freq.        3600 Hz                                 *
;*                                                                   *
;*     Number of points of sine table      256                       *
;*     ---------------------------------- = ------ =      48        *
;*              5,33                         5,33                    *
;*                                                                   *
;*                                                                   *
;*     This means that every 48-th point of the sine table should be used in a *
;*     circular buffer format.                                       *
;*                                                                   *
;*      4) Clearing the 7PRS circular buffer                         *
```

```
;*      5) Reading the filter coefficients into DSP memory                    *
;************************************************************************

                movec   #6,omr          ;Enable onboard ROMs

;*** 1) Setup address register, offset register and modifier files

                move    #$100,r1        ;Set counter for pilot tone
                move    #$10,n1         ;Set length of jump (freq)
                move    #$ff,m1         ;Set length of sine table
                move    #5,r5           ;Set counter for longint read
                move    #0,r7           ;Clear symbol counter
                move    #0,r3           ;Clear learning phase register
                move    #0,r4           ;7PRS Circular buffer base address
                move    #buffmod,m4     ;Set circular buffer modifier value
                move    #$1000,r0       ;Filter coefficient circular buffer base address
                move    #lumod,m0       ;Set circular buffer modifier value
                move    #lupps,n0       ;Set offset jump value

;*** 2) Clear temporary memory locations

                move    #0,x0
                move    x0,y:tempI      ;Clear y:tempI
                move    x0,y:tempQ      ;Clear y:tempQ
                move    x0,y:store      ;Clear y:store

;*** 3) Setup carrier frequency of 3600 Hz

                move    #$100,r2        ;First position on sine table
                move    #48,n2          ;Offset 48 for frequency of 3600 Hz
                move    #$ff,m2         ;Define last position of circular buffer
                move    #$140,r6        ;90 degree phase shift for cosine
                move    #48,n6          ;Offset 48 for frequency of 3600 Hz
                move    #$ff,m6         ;Modifier value for circular buffer
```

```
;*** 4) Clear 7PRS circular buffer

                move    #0,y0           ;Clear y0
                do      #buffleng,clear ;Repeat #buffleng (32) times
                move    y0,y:(r4)+
clear


;*** 5) Read the filter coefficients to memory x:$1000..1080

                do      #luleng,endin   ;Repeat #luleng (128) times
filin
                btst    #0,x:HSR        ;Check if host is ready
                jcc     filin           ;If not goto filin
                move    x:HDR,a         ;Move filter coef. to acc. a
                move    a,x:(r0)+       ;Store acc. a in circular buffer
endin
                move    #$1000,r0       ;Restore r0 base value

                clr     a               ;Clear acc. a
                move    a,x0            ;Clear register x0
                move    a,y0            ;Clear register y0


calc
;*********************************************************************
;* This routine filters the two 7PRS signals with the PSF.           *
;*                                                                   *
;* Explanation of the two MAC routines :                             *
;*    mac  x1,x0,a y:(r4)+,x0                                        *
;*      a = x1 * x0 + a.  A new value is loaded into x0 from y:(r4). *
;*      r4 is increased by 1.                                        *
;*                                                                   *
;*    mac  x1,y0,b x:(r0)+n0,x1   y:(r4)+,y0                         *
;*      b = x1 * y0 + b.  A new value is loaded into x1, with a      *
;*      offset value of 8, from x:(r0).  y0 is loaded with a new     *
```

```
;*     value while r4 is increased by 1.                                    *
;*                                                                          *
;*   Where,                                                                 *
;*     x1 = filter coefficients                                             *
;*     x0 = 7PRS I channel data                                             *
;*     y0 = 7PRS Q channel data                                             *
;*                                                                          *
;*     Thus, the I and Q channel data are multiplied with the same          *
;*     filter coefficient.  The I & Q buffer pointer is incremented         *
;*     by 1 while the filter pointer is increased by 8.                     *
;**************************************************************************

                clr   a     x:(r0)+n0,x1
                clr   b                     ;Clear register b

                do    #symbols,point        ;Repeat #symbol (16) times
                mac   x1,x0,a       y:(r4)+,x0
                mac   x1,y0,b       x:(r0)+n0,x1   y:(r4)+,y0
point
                move  x:(r0)-n0,x1          ;Decreases the value of r0 by 8
                move  x:(r0)+,x1            ;Increases the value of r0 by 1


;**************************************************************************
;* Output signal to the AIC                                               *
;* Multiply the filtered data of the I en Q channels with the sine and cosine carriers.  *
;* The filtered data and carriers are scaled to the correct values.       *
;**************************************************************************

;*** I-CHANNEL
                asr   a                     ;Move register A right.  This
                asr   a                     ;is done to prevent overflow.
                asr   a
                asr   a
                asr   a
                move  a0,a1                 ;Copy a0 into a1
```

```
                move    a0,x1           ;Transfer a0 to x1


;*** Output the I-channel baseband signal to the AIC

                do      #11,SinShift    ;Scale acc. a for output
                asl     a
SinShift
                move    x:mask,y1       ;Load x0 with mask
                and     y1,a            ;Mask acc. a with x0 ($fffc00)
;outIAIC
;               btst    #6,x:TEST       ;Test if AIC ready
;               jcc     outIAIC         ;If not goto outIAIC
;               movep   a,x:TX          ;Else output acc. a.

                move    y:(r2)+n2,a     ;Load next point of sine wave (3600Hz)
                do      #12,Shift1      ;Scale the sine wave down
                asr     a
Shift1
                move    a1,y1           ;Load y1 with a1
                mpy     x1,y1,a         ;Modulate the I-channel
                asr     a               ;Divide by 2
                move    a0,a1           ;Save value in a1


;*** Output the modulated I-channel to the AIC

                move    x:mask,y1
                and     y1,a
;outICAIC
;               btst    #6,x:TEST       ;Test if AIC ready
;               jcc     outICAIC        ;If not goto outICAIC
;               movep   a,x:TX          ;Else output acc. a


;*** Q-CHANNEL
```

```
                asr   b                 ;Move acc. a right.  This
                asr   b                 ;is done to prevent overflow.
                asr   b
                asr   b
                asr   b
                move  b0,b1             ;Copy b0 into b1
                move  b0,x1             ;Transfer b0 to x1


;*** Output the Q-channel baseband signal to the AIC

                do    #11,CosShift
                asl   b
CosShift
                move  x:mask,y1
                and   y1,b
;outQAIC
;               btst  #6,x:TEST         ;Test if AIC ready
;               jcc   outQAIC           ;If not goto outQAIC
;               movep b,x:TX            ;Else output b

                move  y:(r6)+n6,b       ;Load the next cosine value (3600Hz)
                do    #12,Shift2        ;Scale the cosine wave down
                asr   b
Shift2
                move  b1,y1
                mpy   x1,y1,b
                asr   b
                move  b0,b1


;*** Output the modulated Q-channel to the AIC

                move  x:mask,y1
                and   y1,b
;outQCAIC
```

```
;               btst    #6,x:TEST       ;Test if AIC ready
;               jcc     outQCAIC        ;If not goto outQCAIC
;               movep   b,x:TX          ;Else output b

                add     b,a             ;I + Q-modulated signal = 49-QPRS

;***ADD PILOT TONE

                clr     b
                move    y:(r1)+n1,b     ;Read pilot tone voltage from sine table
                rep     #$5
                asr     b
                add     b,a             ;Add pilot tone

;*** Output the 49 QPRS signal to the AIC

                move    x:mask,y1
                and     y1,a
out49QPRS
                btst    #6,x:TEST       ;Test if AIC ready
                jcc     out49QPRS       ;If not goto out49QPRS
                move    a,x:TX          ;Else output a


                jsr     sym             ;Look if next symbols should be calculated
                jmp     calc            ;Filter next symbols

sym
;************************************************************************
;* Check if new symbols (I & Q) should be calculated.                   *
;* Because there are 8 samples per symbol, a new symbol should be       *
;* calculated every 8-th filtering cycle.                               *
;************************************************************************

                move    x:(r7)+,a       ;Dummy move to inc r7
                move    r7,x1           ;Load x1 with the value of r7
```

```
                move    #7,a1           ;Set maximum count (0-7)
                and     x1,a
                jseq    Code            ;If x1 = a then goto Code
                rts                     ;else return to output subroutine

Code
;*********************************************************************
;* Learning phase data loaded                                        *
;*********************************************************************

                move    x:(r3)+,a       ;Dummy move - incr r3
                move    r3,x1
                move    #>$c0,a         ;192 ($c0) baud learning phase
                cmp     x1,a
                jgt     learn                   ;Jump to learn

;*********************************************************************
;* Encode the data and store in the buffer                           *
;* Data are now encoded - and a new lonint is read if necessary      *
;*********************************************************************

                move    x:(r5)+,a       ;Dummy move - incr r5
                move    r5,x1
                move    #6,a1
                cmp     x1,a
                jseq    read            ;If r5=$6 then read new longint

encode
;*********************************************************************
;* Precode and code data into 7 PRS format - I-channel into x0       *
;* and Q-channel into y0                                             *
;*********************************************************************

                move    #0,r7           ;Reset the CalcSym counter
                move    #$1000,r0       ;Reset the filter file
                clr     a
```

```
clr    b
move   x:datain,a0          ;Longint into a0
asl    a                    ;a = 2 MSB of longint
asl    a
move   #>3,x0               ;Load x0 with 3
and    x0,a                 ;a(k) of I in a1
move   a0,b0
asl    b
asl    b
move   b0,x:datain
and    x0,b
move   #>4,y0               ;Load y0 with 4
add    y0,a y:tempI,x1      ;a=a+4 and load x1 with b(k-1) of I
add    y0,b y:tempQ,y1      ;b=b+4 and load y1 with b(k-1) of Q
sub    x1,a                 ;b(k)=a(k)-b(k-1) of I
sub    y1,b                 ;b(k)=a(k)-b(k-1) of Q
and    x0,a
and    x0,b a1,y:tempI      ;Store b(k) of I
add    x1,a b1,y:tempQ      ;y(k)=b(k)+b(k+1) and store b(k) of Q
add    y1,b                 ;y(k)=b(k)+b(k+1) of Q
sub    x0,a                 ;Shift I-channel level
sub    x0,b                 ;Shift Q-channel level
move   y:(r4)-,x0           ;Dec r4 and get new value for x0
move   b1,y:(r4)-           ;Dec r4 and Save 7PRS Q value
move   b1,y0                ;Save 7PRS Q value in y0
move   a1,y:(r4)            ;Save 7PRS I value in y:(r4);
move   a1,x0                ;Save 7PRS I value in x0
rts                         ;CalcSym


read
;*********************************************************************
;* Reads a new long integer from the host.                           *
;*********************************************************************

        btst   #0,x:HSR     ;Check if host is ready
```

```
                jcc    read                    ;If not goto Read
                move   x:HDR,a          ;Move long integer to register a
                move   a,x:datain       ;Store the long integer in x:datain
                move   #0,r5            ;Reset the long integer counter
                rts

learn
;*********************************************************************
;* Transmit learning phase (I=0,0,3,3,0... and Q=0,2,0,2,0,2...                *
;*********************************************************************

                clr    b      #$200000,y1
                jclr   #0,x1,learnq
                or     y1,b            ;Set Q=2

learnq
                move   #$C00000,y1
                jclr   #1,x1,learni
                or     y1,b
learni
                move   b1,x:datain
                jmp    encode

                end
```

APPENDIX G

PROGRAMME: CARRIER RECOVERY

```
;*************************************************************
;* This programme reads the sampled values of the transmitted signal      *
;* (stored as SAMPLE.DAT) and recovers the in-phase and quadrature        *
;* carrier signals. These signals are written to a file on the receiver host PC *
;* for further processing.                                                *
;*************************************************************

;************ FILTER 2 (1200 Hz)*********************************
;* This is a BPF which recovers the pilot tone from the transmitted signal. *
;*                                                                        *
;* Length       => 180                                                    *
;* Memory (Coef)=> x:$400..$4b4                                           *
;* Memory (Data)=> y:$400..$4b4                                           *
;* Pointer      => r6,m6,r5, m5 - Not Free                                *
;*************************************************************

;*************FILTER 3 (3600 Hz)*****************************
;* This is a BPF, centred at 3600 Hz. It recovers the carrier frequency.  *
;*                                                                        *
;* Length       => 80                                                     *
;* Memory (Coef)=> x:$500..$54f                                           *
;* Memory (Data)=> y:$500..$54f                                           *
;* Pointer      => r7, m7 - Not free                                      *
;*************************************************************

;**********CORRELATION COEFFICIENT*************************
;* Data length  => 12 samples          (m3=#$b)                           *
;* Memory       => x:$600              (r3=$600)                          *
;* Sine table   => y:$100-$1ff         (r4=$100, m4=$ff)                  *
;* Stepsize for 3k6 => n4=#$30                                            *
;* Maximize cor => x:$700              (r2=$700, m2=$1)                   *
```

```
;* Memory (stack)=>x:$702 to $704          (r2, n2, m2)                    *
;************************************************************

;**************SINETABLE OUTPUT*************************
;* Register     => r2=$100 to $1ff                                        *
;* Memory (stack)=> x:$705 to $707 (r2, n2, m2)                           *
;************************************************************

;------------------------------------------------------------------------

            opt     fc,mu,s,w,mex

Filt_Len2       EQU  180            ;Length of 1200 Hz filter
Filt_Len3       EQU  80             ;Length of 3600 Hz filter
Shift_Len       EQU  9
Shift_Len2      EQU  $6
Shift_Len3      EQU  $6

HSR             EQU  $ffe9          ;Host status register
HDR             EQU  $ffeb          ;Host data register

        ORG     X:$400

Coef2   ; coefficients for 1200 Hz pilot filter are stored in reverse order,
;         with h[N-1] first, h[0] last

        DC -144,-179,-190,-170,-120,-44
        DC 47,139,219,270,283,251
        DC 175,64,-67,-199,-309,-379
        DC -394,-347,-240,-87,91,267
        DC 413,503,519,454,313,113
        DC -116,-341,-525,-636,-653,-568
        DC -389,-140,144,419,641,773
        DC 790,684,467,167,-171,-495
        DC -755,-907,-923,-796,-541,-193
        DC 196,567,860,1029,1044,897
```

```
        DC 607,216,-218,-629,-952,-1134
        DC -1146,-981,-662,-234,236,678
        DC 1023,1215,1223,1043,701,248
        DC -249,-712,-1070,-1267,-1271,-1080
        DC -724,-255,255,727,1089,1285
        DC 1285,1089,727,255,-255,-724
        DC -1080,-1271,-1267,-1070,-712,-249
        DC 248,701,1043,1223,1215,1023
        DC 678,236,-234,-662,-981,-1146
        DC -1134,-952,-629,-218,216,607
        DC 897,1044,1029,860,567,196
        DC -193,-541,-796,-923,-907,-755
        DC -495,-171,167,467,684,790
        DC 773,641,419,144,-140,-389
        DC -568,-653,-636,-525,-341,-116
        DC 113,313,454,519,503,413
        DC 267,91,-87,-240,-347,-394
        DC -379,-309,-199,-67,64,175
        DC 251,283,270,219,139,47
        DC -44,-120,-170,-190,-179,-144

        ORG     X:$500

Coef3   ; coefficients for 3600 Hz filter are stored in reverse order,
;       with h[N-1] first, h[0] last

        DC -1081,258,1315,756,-767,-1372
        DC -277,1194,1209,-287,-1460,-836
        DC 845,1507,303,-1302,-1314,311
        DC 1576,900,-906,-1611,-323,1384
        DC 1392,-328,-1659,-944,948,1681
        DC 335,-1434,-1439,338,1705,967
        DC -969,-1712,-341,1453,1453,-341
        DC -1712,-969,967,1705,338,-1439
        DC -1434,335,1681,948,-944,-1659
        DC -328,1392,1384,-323,-1611,-906
```

```
        DC 900,1576,311,-1314,-1302,303
        DC 1507,845,-836,-1460,-287,1209
        DC 1194,-277,-1372,-767,756,1315
        DC 258,-1081

            org    p:$0000
            jmp    InitValues

            org    p:$0800
InitValues
            ori    #%00000100,omr          ;Enable sine table
            move   #>$60,r1
            move   #>$2,m1
            move   #$400,r6
            move   #Filt_Len2-1,m6
            move   #$500,r5
            move   #Filt_Len3-1,m5
            move   #$500,r7
            move   #Filt_Len3-1,m7
            move   #>$600,r3
            move   #>$b,m3
            move   #$100,r4
            move   #$ff,m4
            move   #$30,n4
            move   #$100,r2
            move   r2,x:$705
            move   #>$30,n2
            move   #>$ff,m2
            move   m2,x:$707
            move   #>$700,x0
            move   x0,x:$702
            move   #>$1,x0
            move   x0,x:$704

;CLEAR CIRCULAR BUFFERS
            move   #0,x0
```

```
        do      #180,stopclr
        move    x0,y:(r6)+
stopclr

;INPUT DATA

        do      #1024,Spt

;CALCULATE CARRIER

data_in
        btst    #0,x:HSR                ;Check if host is ready
        jcc     data_in                 ;If not, go to data_in
        move    x:HDR,a                 ;Input data to acc. a
;       move    y:$800,a
        move    a,y:(r6)+

;Filter 2 - 1200 Hz bandpass (pilot tone)

Filt2
    MOVE    #Coef2,R0
    MOVE    #Filt_Len2-1,M6             ;Use modulo arithmetic for circular buffer
    NOP
    CLR     A       X:(R0)+,X0    Y:(R6)+,Y0
;   init A to zero      get last coeff, get last delay element

    DO      #Filt_Len2,_endFIR2         ;Execute filter loop

    MAC     X0,Y0,A     X:(R0)+,X0    Y:(R6)+,Y0
;   A = A + coeff*delay  get next coeff  get next delay element

_endFIR2
    DO      #Shift_Len,_Shift2
    ASL     A                           ;Adjust to correct amplitude
_Shift2
```

```
;out_P                                  ;Transfer pilot tone to host PC
;       btst    #1,x:HSR                ;Test if host is ready
;       jcc     out_P                   ;If not goto out_P
;       move    a,x:HDR                 ;Transmit pilot to host
;       move    a,y:$700

        move    x:(r1)+,b               ;Multiply pilot frequency with 3
        move    #>$62,x0
        move    r1,b
        sub     x0,b
        jmi     filt
        move    a,y:(r5)+
filt
        move    y:(r6)-,a               ;Dummy move - dec r6

; Start of filter 3 - 3600 Hz bandpass

Filt3   MOVE    #Coef3,R0
        MOVE    #Filt_Len3-1,M7         ;Use modulo arithmetic for circular buffer
        NOP
        NOP
        CLR     A       X:(R0)+,X0      Y:(R7)+,Y0
;       init A to zero  get last coeff  get last delay element

        DO      #Filt_Len3,_endFIR3

        MAC     X0,Y0,A X:(R0)+,X0      Y:(R7)+,Y0
;       A = A + coeff*delay  get next coeff  get next delay element

_endFIR3
;
        DO      #Shift_Len,_Shift3
        ASL     A                       ;Adjust to correct amplitude
_Shift3

;out_Car                                ;Output of unsynchronized carrier signal
```

```
;       btst    #1,x:HSR                ;Test if host is ready
;       jcc     out_Car                 ;If not goto out_Car
;       move    a,x:HDR                 ;Transmit carrier to host
;       move    a,y:$701

        move    a,x:(r3)+
        move    #$40,n2
        nop
        move    y:(r2)+n2,a             ;Position table pointer for sin
;       rep     #$6
;       asr     a
out_S                                   ;Output synchronized sine position on table
        btst    #1,x:HSR                ;Test if host is ready
        jcc     out_S                   ;If not goto out_S
        move    a,x:HDR                 ;Transmit sin to host
;       move    a,y:$702
        move    #$f0,n2
        nop
        move    y:(r2)+n2,a
;       rep     #$6
;       asr     a
        move    #$30,n2
        move    r2,a
out_C                                   ;Output cos position on sin table
        btst    #1,x:HSR                ;Test if host is ready
        jcc     out_C                   ;If not goto out_C
        move    a,x:HDR                 ;Transmit cos to host

;BEGINNING OF CORRELATION LOOP

        move    r3,b1
        move    #$0,b2
        move    #$601,x0
        sub     x0,b
        jsmi    Corr
```

```
Spt                                     ;Read next data value

;CALCULATE CORRELATION AT ANGLE A

Corr

;Push and pop r2,n2, m2
        move    r2,x:$705
        move    m2,x:$707
        move    x:$702,r2
        move    x:$704,m2
        nop
        move    r4,x:(r2)+
        CLR     A               X:(R3)+,X0      Y:(R4)+n4,Y0
;       init A to zero      get last coeff get last delay element

        DO      #$c,_endLoop                    ;Correlate recovered signal with current position on table

        MAC     X0,Y0,A     X:(R3)+,X0      Y:(R4)+n4,Y0
;       A = A + x1*x2   Next sine value  Next sine table value

_endLoop

        DO      #Shift_Len2,_Shift
        ASR     A                               ;Align to correct Q-notation
_Shift
        move    a,x:(r2)+


;CALCULATE CORRELATION AT ANGLE A+ (+1.41 degrees)

        move    r4,a
        move    #>$70,n4
        move    x:(r3)-,a
        move    x:(r4)-n4,a
        move    #$30,n4
```

```
        move    x:(r4)+,a
        move    r4,x:(r2)+
Corr2
        CLR     A       X:(R3)+,X0      Y:(R4)+n4,Y0
;       init A to zero  get last coeff  get last delay element

        DO      #$c,_endLoop2           ;Correlate recovered signal with (current position+1) on table

        MAC     X0,Y0,A X:(R3)+,X0      Y:(R4)+n4,Y0
;       A = A + x1*x2   Next sine value Next sine table value

_endLoop2

        DO      #Shift_Len3,_Shift2
        ASR     A                       ;Align to correct Q-notation
_Shift2

        move    x:(r3)-,b               ;Dummy move (dec r3)
        move    x:(r4)-,b
        move    x:(r2)+,x0
        cmp     x0,a
        jgt     Adjust180               ;Add 180 degree if rho(x)>rho(x+1) - (Error > 180 degrees)
        neg     a       x:(r2),r4
        move    #$80,n4
        nop
        move    x:(r4)+n4,b
        move    #$30,n4
;       nop
        move    r4,x:(r2)

Adjust180

        move    #>$0,y1
        add     y1,a
        jgt     Adjust90                ;Add 90 degrees if rho(x)<0
        move    x:(r2),r4
```

```
        move    #>$40,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #$30,n4
Adjust90
        move    x:(r2),r4
        move    #>$bf,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #>$30,n4

        jsr     Correl                  ;Accurate to within 45 degrees
        jpl     TooHigh
        move    x:(r2),r4
        move    #>$20,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #>$30,n4
TooHighR

        jsr     Correl                  ;Accurate to within 22.5 degrees
        jpl     TooHigh2
        move    x:(r2),r4
        move    #>$10,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #>$30,n4

TooHigh2R
        jsr     Correl                  ;Accurate to within 11.25 degrees
        jpl     TooHigh3
        move    x:(r2),r4
```

```
        move    #>$8,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #>$30,n4

TooHigh3R
        jsr     Correl
        jpl     TooHigh4

TooHigh4R

Loop
        jsr     Correl
        jpl     Output                  ;Correct phase angle determined - exit from correlation loop
        move    x:(r2),r4
        move    #>$1,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #>$30,n4
        jmp     Loop

;Spt

Correl                                  ;Correlation with one position shifted on sine table
        move    x:(r2)+,r4
        move    #$600,r3
        nop
        CLR     A           X:(r3)+,X0      Y:(R4)+n4,Y0
;       init A to zero      get last coeff  get last delay element

        DO      #$c,_endCor

        MAC     X0,Y0,A     X:(R3)+,X0      Y:(R4)+n4,Y0
;       A = A + x1*x2       Next sine value Next sine table value
```

```
_endCor

    DO    #Shift_Len2,_Shift2
    ASR   A                          ;Align to correct Q-notation
_Shift2

    move  a,x:(r2)+
    rts

TooHigh
    move  x:(r2),r4
    move  #>$e0,n4
    nop
    move  x:(r4)+n4,b
    move  r4,x:(r2)
    move  #>$30,n4
    jmp   TooHighR

TooHigh2
    move  x:(r2),r4
    move  #>$f0,n4
    nop
    move  x:(r4)+n4,b
    move  r4,x:(r2)
    move  #>$30,n4
    jmp   TooHigh2R

TooHigh3
    move  x:(r2),r4
    move  #>$f8,n4
    nop
    move  x:(r4)+n4,b
    move  r4,x:(r2)
    move  #>$30,n4
    jmp   TooHigh3R
```

```
TooHigh4
        move    x:(r2),r4
        move    #>$f9,n4
        nop
        move    x:(r4)+n4,b
        move    r4,x:(r2)
        move    #>$30,n4
        jmp     TooHigh4R

Output
        move    x:(r2),r4
        move    #>$c3,n4        ;Determine final phase of carrier waves - compensate for
        nop                     ;Delay through pilot BSF in DENX2.ASM.
        move    x:(r4)+n4,b
        move    #$30,n4
        move    r4,x:(r2)
        move    r4,x:$705

;Push r2, n2,m2
        move    r2,x:$702
        move    m2,x:$704
        move    x:$705,r2
        move    x:$707,m2
        rts
        nop
        nop

        end
```

APPENDIX H

PROGRAMME: CLOCK RECOVERY

```
;*********************************************************************
;* This programme recovers a clock signal from a baseband 7PRS signal.          *
;* The PRS signal is decoded and the data is written to a data file on the host PC.  *
;*********************************************************************


;**************** FILTER COEF ****************
;* Length      => 118                                  *
;* Memory      => x:$200..$280                         *
;* Pointer     => r0                                   *
;*********************************************

;*************** DATA IN FILTER***************
;* Length      => 118                                  *
;* Memory      => y:$200..$280                         *
;* Pointer     => r4,m4 - Not Free                     *
;*********************************************

;*********** DELAY BUFFER ******************
;* Length      => 9                                    *
;* Memory      => x:$300..$308                         *
;* Pointer     => r3, m3, n3 - Not Free                *
;*********************************************

;************* LEVEL DETECTOR **************
;* Length      => 6 values                             *
;* Memory      => x:$290 ..$295                        *
;* Pointer     => r1 - Not Free                        *
;*********************************************

;---------------------------------------------------------------------
```

```
            opt    fc,mu,s,w,mex

Filt_Len    EQU    118          ;Length of filter
Shift_Len   EQU    9            ;Shift value
Learn       EQU    $053
Temp        EQU    $052         ;Previous sample sine value
Tel         EQU    $051         ;Pointer for sample
mask        EQU    $050         ;Saves the AIC mask value
HSR         EQU    $ffe9        ;Host status register
HDR         EQU    $ffeb        ;Host data register
CRA         EQU    $ffec        ;Control register A
CRB         EQU    $ffed        ;Control register B
PCC         EQU    $ffe1        ;Port C control register
PCD         EQU    $ffe3        ;Port C direction register
RX          EQU    $ffef        ;Receive register
TX          EQU    $ffeb        ;Data to host
TEST        EQU    $ffee        ;AIC ready register
WS          EQU    $fffe        ;Wait state register
INT         EQU    $ffff        ;Interrupt priority register


            ORG    X:$200

Coef    ; coefficients are stored in reverse order,
;       with h[N-1] first, h[0] last

            DC -528,536,406,298,164,16,-95,-124,-59,57,152,164
            DC 80,-56,-164,-179,-86,64,182,196,93,-71,-199,-212
            DC -99,78,215,228,105,-85,-230,-242,-110,91,243,254
            DC 114,-97,-256,-265,-117,103,266,274,119,-108,-275,-281
            DC -121,112,281,285,121,-116,-286,-288,-120,119,288,288
            DC 119,-120,-288,-286,-116,121,285,281,112,-121,-281,-275
            DC -108,119,274,266,103,-117,-265,-256,-97,114,254,243
            DC 91,-110,-242,-230,-85,105,228,215,78,-99,-212,-199
            DC -71,93,196,182,64,-86,-179,-164,-56,80,164,152
```

```
        DC 57,-59,-124,-95,16,164,298,406,536,-528

        org    p:$0000
        jmp    begin

        org    p:$0080

begin
        movep  #0,x:WS                  ;Set zero wait states
        ori    #%00000100,omr           ;Enable onboard ROM


; *** Set-up SSI control registers

        move   #$4000,a                 ;Set-up value for SSI register A
        movep  a,x:CRA                  ;Set-up SSI control register A
        move   #$3A00,a                 ;Set-up value for SSI register B
        movep  a,x:CRB                  ;Set-up SSI control register B
        movep  #$FFFF1E,y:$FFE0         ;Set AIC master freq.

;*** Set-up Port C control registers

        move   #$01FF,a                 ;Set-up value for Port C register
        movep  a,x:PCC                  ;Set-up Port C control register
        move   #$010E,a                 ;Set-up value for direction register
        movep  a,x:PCD                  ;Set-up Data direction register

;*** Set-up the interrupts and mask values

        move   #$0000,a                 ;Disable interrupts
        move   a,x:INT                  ;Set-up interrupt priority

        move   #$1000,r0

        move   #$FFFC00,b1              ;Set-up mask value
        move   b1,x:mask                ;Save mask value
```

```
;*** SET-UP THE AIC

        move    #$ffffff,b              ;Value to inform AIC that it is to be
                                        ;programmed.

;*** Primary transmission for AIC control register

priC
        btst    #6,x:TEST               ;Check if DAC ready
        jcc     priC                    ;If not goto priC
        movep   b,x:RX                  ;Else transmit programming value

;*** Secondary transmission for AIC control register

        move    #$00C300,a              ;Set-up AIC control register value
secC
        btst    #6,x:TEST               ;Check if DAC ready
        jcc     secC                    ;If not goto secC
        movep   a,x:RX                  ;Else transmit value

;*** Primary transmission for Counter A

priTA
        btst    #6,x:TEST               ;Test if DAC ready
        jcc     priTA                   ;If not goto priTA
        movep   b,x:RX                  ;Else transmit programming value

;*** Secondary transmission for Counter A

        move    #$0a1400,a              ;Set-up counter A

secTA
        btst    #6,x:TEST               ;Test if DAC ready
```

```
        jcc    secTA                ;If not goto secTA
        movep  a,x:RX               ;Else transmit sampling freq. value

;*** Primary transmission for Counter B

priTB
        btst   #6,x:TEST            ;Test if DAC ready
        jcc    priTB                ;If not goto priTB
        movep  b,x:RX               ;Else transmit programming value

;*** Secondary transmission for Counter B

        move   #$366e00,a           ;Sampling freq. of AIC (19200 Hz)
secTB
        btst   #6,x:TEST            ;Test if DAC ready
        jcc    secTB                ;If not goto secTB
        movep  a,x:RX               ;Else transmit sampling freq. value

;*** Filter settling time

        move   x:mask,x0            ;load x0 with mask
        do     #255,sloop           ;Repeat 255 times
datai   btst   #6,x:TEST            ;Test if DAC ready
        jcc    datai                ;If not goto datai
        and    x0,b                 ;Mask register b
        movep  b,x:RX               ;Transmit register b
        move   #0,b                 ;Clear register b
sloop

;InitValues
                move   #$2000,r2
                movec  #6,omr       ;Enable onboard ROMS
                move   #0,x0
                move   x0,x:Temp
                move   x0,x:Tel
                move   #$300,r3
```

```
                move    #8,m3
                move    #$200,r4
                move    #>$fa0,a        ;Set decision levels (0)
                move    a,x:$290
                move    #$276,x0        ;(1)
                sub     x0,a
                move    a,x:$291
                move    #$29e,x0        ;(2)
                sub     x0,a
                move    a,x:$292
                move    #$294,x0        ;(3)
                sub     x0,a
                move    a,x:$293
                move    #$29e,x0        ;(4)
                sub     x0,a
                move    a,x:$294
                move    #$276,x0        ;(5)
                sub     x0,a
                move    a,x:$295        ;(6)
                MOVE    #Filt_Len-1,M4 ;use modulo arithmetic for circular buffer

;CLEAR CIRCULAR BUFFERS
                move    #0,x0
                do      #118,stopclr
                move    x0,y:(r4)+
stopclr
                move    #$200,r4

;CLEAR X(N)-X(N-8) DELAY BUFFER
                do      #9,res
                move    x0,x:(r3)+
res
                move    #$300,r3
                do      #4095,Spt

;INPUT DATA
```

```
din
                btst   #6,x:TEST          ;Input data through ACIA
                jcc    din
                movep  x:RX,a             ;Save data in acc. a
                movep  #$0,x:RX           ;Acknowledge data received
                do     #10,dev
                asr    a
dev
                move   a,x:(r2)+          ;Save raw data to x:(r2)+
                move   a,x:(r3)+
                move   x:(r3),b

;CALCULATE TIME RECOVERY SIGNAL
                sub    b,a    #$1c0,y0    ;Calculate x(n) - x(n-8)
                abs    a                  ;Take absolute value
;           sub    y0,a                   ;Offset (by $1c0)
;           move   a,x:(r2)+              ;Write processed data to x:(r2)+
                move   a,y:(r4)+          ;Save value in Sipho buffer

;Filtering routine
        MOVE   #Coef,R0                   ;Beginning of filtering routine
        NOP
        CLR    A        X:(R0)+,X0       Y:(R4)+,Y0
;    init A to zero      get last coeff   get last delay element
        DO     #Filt_Len,_endFIR
        MAC    X0,Y0,A  X:(R0)+,X0       Y:(R4)+,Y0
;    A = A + coeff*delay  get next coeff  get next delay element
_endFIR

        DO     #Shift_Len,_Shift
        ASL    A                          ; align to correct Q-notation
_Shift
;       move   #>5000,b
;       add    b,a
;       move   a,x:(r2)+                  ;Write sine wave (clock signal) to x:(r2)+
```

```
;Start of peak detector

        move    x:Temp,y1           ;Load xs(n-1) into y1
        move    a,x:Temp            ;Save xs(n) into x:Temp

        move    x:Tel,b             ;Beginning of TEL=0 decision
        move    #0,x1
        cmp     x1,b
        jeq     Top                 ;Jump to Top since TEL=0
        cmp     y1,a                ;Determine if xs(n)>xs(n-1)
        jsgt    ClearTel            ;Jump to Cleartel since xs(n)>Xs(n-1)

Output                              ;Output peak pulse o/p to host PC
;       move    b,x:(r2)+           ;Write peak value to x:(r2)+
        move    y:(r4)-,a           ;Dummy move (dec r4)

Spt

        move    #$2000,r2

Uitt                                ;Download decoded data to host

        btst    #1,x:HSR
        jcc     Uitt
        move    x:(r2)+,a
        move    a,x:TX
        jmp     Uitt

Top                                 ;Determine whether xs(n)<xs(n-1)
        cmp     y1,a
        jslt    Sample              ;Jump to TEL=1 since xs(n)<xs(n-1)
        jmp     Output              ;Jump to peak pulse o/p

ClearTel
        move    #0,y1               ;Set TEL=0
        move    y1,x:Tel
```

```
        rts


Sample                          ;Set TEL=1, sample I and Q channels
        move    #>1,y1
        move    y1,x:Tel
        move    #>6,n1
        nop
        move    x:(r3)+n3,b
        move    x:(r3)-n3,b
;       move    b,x:(r2)+
        move    #$960,x0
        add     x0,b
        move    #$290,r1
        nop
        move    x:(r1)+,x0      ;Deciding which level
        cmp     x0,b
        jgt     Value0
        move    x:(r1)+,x0
        cmp     x0,b
        jgt     Value1
        move    x:(r1)+,x0
        cmp     x0,b
        jgt     Value2
        move    x:(r1)+,x0
        cmp     x0,b
        jgt     Value3
        move    x:(r1)+,x0
        cmp     x0,b
        jgt     Value0
        move    x:(r1)+,x0
        cmp     x0,b
        jgt     Value1
        move    #>2,b
Value                           ;Read dibit value to host PC
;       move    b,x:(r2)+
```

```
        rts

Value0
        move    #>0,b
        jmp     Value
Value1
        move    #>1,b
        jmp     Value
Value2
        move    #>2,b
        jmp     Value
Value3
        move    #>3,b
        jmp     Value

        end
```

REFERENCES

1. **Bateman, A. and Yates, W.** Digital Signal Processing Design. Londen, Pitman. 1988.

2. **Bellamy, J.** Digital Telephony. New York, John Wiley and Sons. 1991.

3. **Bic, J.C., Duponteil, D. and Imbeaux, J.C.** Elements of digital Communications. Chichester, John Wiley and Sons. 1991.

4. **Bingham, J.A.C.** The Theory and Practice of Modem Design. Palo Alto, John Wiley and Sons. 1988.

5. **Bissell, C.C. and Chapman, D.A.** Digital Signal Transmission. New York, Cambridge University Press. 1992.

6. **Bohm, B.** DSP Implementation and Performance Testing of the Massey-Hodgart MSK Demodulator. Unpublished M Eng Thesis, University of Cape Town. 1995.

7. **CCITT Blue Book** Data Communication Over the Telephone Network, Volume VIII, Series V Recommendations. Geneva, IXth Plenary Assembly, Melbourne, 14-25 November 1988. 1989.

8. **Chevillat, P.R. and Ungerboeck, G.** Optimum FIR Transmitter and Receiver Filters for Data Transmission over Band-Limited Channels. IEEE Transactions on Communications, vol. com-30, no. 8. August 1982, pp. 1909-1915.

9. **Courtney, T.D.** The Development of a Novel Modem Structure for Connection of Rural to Diginet. Unpublished M Eng Thesis, University of Cape Town. 1990.

10. **Crouch, S.E.C. and Jedwab, J** Coding in 100VG-AngLAN. Hewlett Packard Journal, vol. 46, no. 4. August 1995. pp. 27-32.

11. **D'Andrea, N.A. and Luise, M.** Design and Analysis of a Jitter-Free Clock Recovery Scheme for QAM Systems. IEEE Transactions of Communications, vol. 41, no. 9. September 1993. pp. 1296-1299.

12. **D'Andrea, N. A. and Mengali, U.** A Simulation Study of Clock Recovery in QPSK and 9QPRS Systems. IEEE Transactions on Communications, vol. com-33, no. 10. October 1985. pp. 1139-1142.

13. **Fliege, N.J.** Multirate Digital Signal Processing, Chichester, John Wiley and Sons. 1994.

14. **Franks, L.E. and Bubrouski, J.P.** Statistical Properties of Timing Jitter in a PAM Timing Recovery Scheme. IEEE Transactions on Communications, vol. com-22, no. 7. July 1974. pp. 913-920.

15. **Frerking, M.E.** Digital Signal Processing in Communication Systems, New York, Van Nostrand Reinhold. **1994.**

16. **Gibson, J.D.** Principles of Digital and Analog Communications, 2nd ed. New York, Macmillan Publishing Company. 1993.

17. **Gitlin, R.D., Hayes, J.F. and Weinstein, S.B.** Data Communications Principles. New York, Plenum Press. 1992.

18. **Hagiwara, M. and Nakagawa, M.** New DSP Type Phase Synchronizer with the Method of Least Squares. Proceedings of International Conference on Acoustics, Speech and Signal Processing, vol. 3. 1988. pp.1882-1885.

19. **Haykin, S.** Communication Systems. 2nd ed. New York, John Wiley and Sons. 1983.

20. **Haykin, S.** Digital Communications, New York, John Wiley and Sons. 1988.

21. **Haykin, S.** An Intoduction to Analog and Digital Communications. New York, John Wiley and Sons. 1989.

22. **Herrington, D.E., Nichols, P.A. and Lipp, R.D.** Software Verification Using Branch Analysis, Hewlett Packard Journal, vol. 38, no. 6. June 1987. pp. 13-22.

23. **Hypersignal Users Manual.** Dallas, Hyperception. 1989.

24. **Ifeachor, E.C. and Jervis, B.W.** Digital Signal Processing: A Practical Approach. Worthingham, Addison-Wesley. 1993.

25. **Jackson, L.B.** Signals, Systems and Transforms, New York, Addison-Wesley. 1991.

26. **Jeruchim, M.C., Balaban, P. and Shanmugan, K.S.** Simulation of Communication Systems. New York, Plenum Press. 1992.

27. **Johnson, J.R.** Introduction to Digital Signal Processing. Englewood Cliffs, Prentice-Hall. 1989.

28. **Jordaan, G.D., Du Plooy, C.E. and Braun, R.M.** A Clock Recovery Scheme for 7PRS Signals, IEEE Africon Conference, vol. 1. September 1996. pp. 397-401.

29. **Kabal, P. and Pasupathy, S.** Partial Response Signalling. IEEE Transactions on Communications, vol. com-23, no. 9. September 1975. pp. 921-934.

30. **Killen, B.K.** Digital Communications with fiber optics and satelite applications, Englewood Cliffs, Prentice-Hall. 1988.

31. **Kim, D.Y.** Lower-Bound Eye Widths of Minimum-Bandwidth Systems, IEEE Transactions on Communications, vol. 43, no. 2. February 1995. pp. 1235-1239.

32. **Kim, D.Y. and Kim, J-K.** A Condition for Stable Minimum-Bandwidth Line Codes, IEEE Transactions on Communications, vol. com.-33, no.2. February 1985. pp.152-157.

33. **Kobayashi, H.** Simultaneous Adaptive Estimation and Decision Algorithm for Carrier Modulated Data Transmission Systems, IEEE Transactions on Communications Technology, vol com.-19, no. 3. June 1971. pp. 268-280.

34. **Lafrance, P.** Fundamental Concepts in Communication. Englewood-Cliffs, Prentice-Hall. 1990.

35. **Lathi, B.P.** Modern Digital and Analog Communication Systems, 2nd ed. Philidelphia, Holt, Rinehart and Winston, Inc. 1989.

36. **Lee, E.A. and Messerschmitt, D.G.** Digital Communication. Boston, Kluver Academic Publishers. 1988.

37. **Liuse, M. and Reggiannini, R.** Carrier frequency recovery in All-Digital Modems for Burst-Mode Transmission, IEEE Transactions on Communications, vol. 43 no. 2, February 1995. pp. 1169-1178.

38. **Loveday, G.C.** Electronic Testing and Fault Diagnosis. Bath, Pitman. 1980.

39. **Meyr, H. and Ascheid, G.** Synchronization in Digital Communications, vol 1. New York, John Wiley and Sons. 1990.

40. **Motorola** DSP56000/DSP56001 Digital Signal Processor User's Manual, Phoenix, Motorola Inc. 1990.

41. **Panayiri, E. and Bar-Ness, E.Y.** A New Approach for Evaluating the Performance of a Symbol Timing Recovery System Employing a General Type of Nonlinearity, IEEE Transactions on Communications, vol. 44, no. 1, January 1996. pp. 29-33.

42. **Paul, C.R.** Analysis of Linear Circuits. New York, McGraw-Hill.1989.

43. **Peebles, P.Z.** Digital Communication Systems. Englewood-Cliffs, Prentice-Hall. 1987.

44. **Peralex** Peralex SIG-56 Reference Manual. Kalk Bay, Peralex Electronic Development CC. 1992.

45. **Perry, M.** Genetic Algorithm Homes In On FIR Filter That Meets Specific Requirements, Personal Engineering. January 1995. pp. 67 69.

46. **Proakis, J.G.** Digital Communications, 2nd ed. McGraw-Hill, New York. 1989.

47. **Rabiner, L.R. and Gold, B.** Theory and Applications of Digital Signal Processing, Englewood-Cliffs, Prentice-Hall. 1975.

48. **Ridout, P.N.** A New Method of Timing-Signal Recovery for (1,0,-1) Multiple-Response Signals, Electronic Letters, vol. 14, no. 12. 8 June 1978. pp.354-355.

49. **Roden, M.S.** Digital Communication System Design, Englewood Cliffs, Prentice-Hall. 1988.

50. **Sal, J. and Lehman, A.** JMP Start Statistics: A guide to Statistics and Data Analysis Using JMP IN Software. Belmont, SAS Institute. 1996.

51. **Schwartz, M.** Information, Transmission, Modulation and Noise, 4th ed. New York, McGraw-Hill. 1990.

52. **Scott, K.E. and Olasz, E.B.** Simultaneous Clock Phase and Frequency Offset Estimation, IEEE Transactions on Communications, vol. 43, no. 7. July 1995. pp. 2263-2270.

53. **Sheers, K. A.** HP OpenView Event Correlation Services, Hewlett-Packard Journal, vol. 47, no. 5. October 1996. pp. 31-42.

54. **Sinnema, W. and McGovern, T.** Digital, Analog and Data Communication, 2 nd ed. Englewood-Cliffs, Prentice-Hall. 1986.

55. **Sklar, B.** Digital Communications: Fundamentals and Applications. Englewood-Cliffs, Prentice-Hall. 1988.

56. **Stark, H., Tuteur, F.B. and Anderson, J.B.** Modern Electrical Communications: Analog, Digital and Optical systems, 2 nd ed. Englewood-Cliffs, Prentice-Hall. 1988.

57. **Stremler, F.G.** Introduction to Communication Systems, 3rd ed. Reading, Addison-Wesley, 1990.

58. **Strum, R.D. and Kirk, D.E.** First Principles of Discrete Systems And Digital Signal Processing. Reading, Addison-Wesley.

59. **Suckley, D.** Genetic Algorithm in the Design of FIR Filters, IEE Proceedings-G, vol. 138, no. 2, April 1991. pp. 235- 238.

60. **Takasaki, Y.** Timing Extraction in Baseband Pulse Transmission, IEEE Transactions on Communications, vol. com-20, no.5. October 1972, pp.877-884.

61. **Takasaki, Y.** Digital Transmission Design and Jitter Analysis, Boston, Artech House. 1991.

62. **Taub, H. and Schilling, D.L.** Principles of Communication Systems, 2 nd ed. Singapore, McGraw-Hill. 1986.

63. Terrestrial Digital Microwave Communications, Ed. by **Ivonek, F.** Norwood, Artech House. 1989.

64. **Texas Instruments** Application Notes: TLC32044I, TLC32044C Voise-band Analog Interface Circuits. Dallas. 1988.

65. **Tomasi, W.** Advanced Electronic Communications Systems. Englewood-Cliffs, Prentice-Hall. 1987.

66. **Trischitta, P.R. and Varna, E.L.** Jitter in Digital Transmission Systems. Norwood, Artech House. 1989.

67. **Trondle, K. and Soder, G.** Optimization of Digital Transmission Systems, Boston, Artech House. 1987.