Appendices
Table of Contents for Appendices

Appendix 1

Phase-Locked Loop theory ................................................................. A1-1
Glossary of PLL terminology .............................................................. A1-1
Basic operation of a PLL ................................................................. A1-4
Loop dynamics in a locked state ..................................................... A1-6
Stability .......................................................................................... A1-13
Tracking behaviour ......................................................................... A1-15
Acquisition of lock ........................................................................ A1-17
Aided acquisition ............................................................................ A1-20

Appendix 2

PLL simulations .................................................................................. A2-1
Summary of PLL characteristics ..................................................... A2-11

Appendix 3

Recording codes .................................................................................. A3-1
Manchester and Miller codes ........................................................... A3-1
Miller² code ...................................................................................... A3-2
RLL 2,7 code ..................................................................................... A3-5
RLL Test generator board .............................................................. A3-7
## Table of Contents for Appendices

### Appendix 4

**Fixed frequency clock recovery** ........................................... A4-1

Jitter generator ............................................................................... A4-1

PDF (Probability Density Function) ............................................. A4-6

Simulation results ........................................................................... A4-9

Manchester encoded data ........................................................... A4-9

Miller encoded data .................................................................... A4-14

Miller$^2$ encoded data ................................................................. A4-18

RLL 2,7 encoded data ................................................................... A4-22

NRZ Scrambled data ..................................................................... A4-26

### Appendix 5

**Fixed frequency clock recovery - Schematic diagrams** .......... A5-1

### Appendix 6

**Fixed frequency clock recovery - Real-time DSP** ................ A6-1

DSP 320C25 Clock recovery software description ..................... A6-1

Interrupt 0 Service ....................................................................... A6-3

Interrupt 2 Service ....................................................................... A6-5

Interrupt 1 Service ....................................................................... A6-8

Experimental results .................................................................... A6-9

Source code listing ........................................................................ A6-12

Clock recovery module assembler source code ......................... A6-12

A-ii
TABLE OF CONTENTS FOR APPENDICES

Appendix 7

Adaptive tracking clock recovery ................................................................. A7-1
Simulation model .......................................................................................... A7-1
Calibration results ......................................................................................... A7-8
Simulation results ........................................................................................ A7-14
Acquisition performance overview ............................................................... A7-14
Manchester encoded data ±5% tracking range ........................................ A7-16
Manchester encoded data ±7.5% tracking range ....................................... A7-19
Manchester encoded data ±10% tracking range ....................................... A7-22
RLL3 encoded data ±5% tracking range .................................................. A7-25
RLL3 encoded data ±7.5% tracking range ................................................ A7-28
RLL7 encoded data ±5% tracking range .................................................. A7-31
Sine input ±10% tracking range ............................................................ A7-34

Appendix 8

FFT Derivation ............................................................................................... A8-1

Appendix 9

OFFT-based clock recovery ......................................................................... A9-1
Simulation model .......................................................................................... A9-1
Simulation results ........................................................................................ A9-6
OFFT - SPW custom coded block source code .......................................... A9-7

A-iii
TABLE OF CONTENTS FOR APPENDICES

<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFT.C Module</td>
<td>A9-7</td>
</tr>
<tr>
<td>OFFT.h Module</td>
<td>A9-12</td>
</tr>
<tr>
<td>C++ Source code for “twiddles.exe”</td>
<td>A9-14</td>
</tr>
<tr>
<td>Twiddles.CPP</td>
<td>A9-14</td>
</tr>
<tr>
<td>Twiddles.Q15</td>
<td>A9-16</td>
</tr>
<tr>
<td>TMS320C50 DSP source code for the OFFT</td>
<td>A9-17</td>
</tr>
<tr>
<td>OFFT DSP-macros</td>
<td>A9-18</td>
</tr>
</tbody>
</table>

Appendix 10

**Dual-processor DSP board** .......................................................... A10-1

Function of onboard links ............................................................ A10-9

Appendix 11

**Author's Published Papers** ...................................................... A11-1

{1} DSP-Based Clock Recovery for a Digital Magnetic Recording Channel.
{2} A Practical Approach to Digital Signal Processing.
{3} DSP-Based Clock Recovery Implemented in a FPGA.
{4} The Development of an Operational Computer Based Satellite Data Broadcasting System.
{5} The Development of an Operational Satellite Internet Service Provision.
{6} A Data Reply Link for Satellite TV Applications.
{7} A Novel Internet Delivery System using Asymmetric Satellite Channels.
LIST OF FIGURES FOR APPENDICES

List of Figures for Appendices

Figure A1-1 Basic phase-locked loop....................................................... A1-4
Figure A1-2 PLL Waveforms ................................................................. A1-5
Figure A1-3 Linear model of a PLL ........................................................ A1-6
Figure A1-4 Simple low-pass lag filter .................................................. A1-8
Figure A1-5 Response of a PLL to a frequency step ................................ A1-11
Figure A1-6 Lag-lead low pass filter ...................................................... A1-12
Figure A1-7 Bode plot of a second order loop with a lag-lead filter ........ A1-14
Figure A1-8 Variation of parameters during tracking ............................. A1-15
Figure A1-9 Gain reduction in PD and VCO ........................................... A1-16
Figure A1-10 Capture behaviour ............................................................ A1-18
Figure A1-11 Aided acquisition with a frequency detector .................... A1-20

Figure A2-1 SPW 2\textsuperscript{nd} Order PLL ........................................... A2-1
Figure A2-2 Simulation of acquisition with equal frequencies ............... A2-2
Figure A2-3 Response to a small change in input frequency ................ A2-3
Figure A2-4 Simulation of acquisition for different frequency offsets ...... A2-4
Figure A2-5 Simulation of acquisition and flywheel .............................. A2-5
Figure A2-6 Wide-band PLL acquisition performance for sine input ....... A2-6
Figure A2-7 Simulation - PLL with data input (BW=0.0015) ................. A2-8
Figure A2-8 Simulation model - PLL with data input ............................ A2-9
Figure A2-9 PLL Clock recovery – Schematic diagram ......................... A2-10

A-v
LIST OF FIGURES FOR APPENDICES

Figure A3-1 Miller code - Test circuit for FPGA ................................................ A3-1
Figure A3-2 Extract from Miller\(^2\) patent ....................................................... A3-2
Figure A3-3 Miller\(^2\) code - Test circuit for FPGA ........................................... A3-3
Figure A3-4 Miller\(^2\) code - Test model and simulation ................................... A3-4
Figure A3-5 RLL 2,7 code - Test circuit for FPGA ........................................... A3-5
Figure A3-6 RLL 2,7 code - Test model and simulation ....................................... A3-6
Figure A3-7 Block diagram of RLL test generator board .................................... A3-7
Figure A3-8 RLL code - Test generator board ................................................... A3-9
Figure A3-9 Schematic of RLL test generator board .......................................... A3-9
Figure A3-10 Combined codes and timing for FPGA ........................................ A3-10

Figure A4-1 Jitter generator - Switch settings ................................................... A4-1
Figure A4-2 Jitter histogram - Uniform distribution ........................................... A4-2
Figure A4-3 Gaussian distribution .................................................................... A4-4
Figure A4-4 Jitter generator - Model and test simulation ................................... A4-5
Figure A4-5 PDF Block ..................................................................................... A4-6
Figure A4-6 PDF detail and test simulation ...................................................... A4-7
Figure A4-7 Histogram analysis of jitter .......................................................... A4-8
Figure A4-8 Manchester encoded - Clock recovery model ............................... A4-9
Figure A4-9 Manchester - Acquisition and flywheel (wide zoom) ................... A4-10
Figure A4-10 Manchester - Acquisition (zoomed in) ........................................ A4-10
Figure A4-11 Manchester - Flywheel (wide zoom) .......................................... A4-11
LIST OF FIGURES FOR APPENDICES

Figure A4-12 Manchester - Flywheel (zoomed in) ............................................ A4-11
Figure A4-13 Manchester - Acquisition with applied jitter ................................... A4-12
Figure A4-14 Manchester - Flywheel with applied jitter (zoomed in) ...................... A4-12
Figure A4-15 Manchester - Jitter rejection ....................................................... A4-13
Figure A4-16 Miller encoded - Clock recovery model ........................................ A4-14
Figure A4-17 Miller - Acquisition and flywheel (wide zoom) ............................ A4-15
Figure A4-18 Miller - Acquisition (zoomed in) ............................................... A4-15
Figure A4-19 Miller - Flywheel (wide zoom) .................................................. A4-16
Figure A4-20 Miller - Flywheel (zoomed in) .................................................. A4-16
Figure A4-21 Miller - Acquisition with applied jitter ....................................... A4-17
Figure A4-22 Miller - Flywheel with applied jitter (zoomed in) ......................... A4-17
Figure A4-23 Miller$^2$ encoded - Clock recovery model .................................. A4-18
Figure A4-24 Miller$^2$ - Acquisition and flywheel (wide zoom) ......................... A4-19
Figure A4-25 Miller$^2$ - Acquisition (zoomed in) .......................................... A4-19
Figure A4-26 Miller$^2$ - Flywheel (wide zoom) .............................................. A4-20
Figure A4-27 Miller$^2$ - Flywheel (zoomed in) ................................................ A4-20
Figure A4-28 Miller$^2$ - Acquisition with applied jitter .................................. A4-21
Figure A4-29 Miller$^2$ - Flywheel with applied jitter (zoomed in) ..................... A4-21
Figure A4-30 RLL 2,7 encoded - Clock recovery model .................................. A4-22
Figure A4-31 RLL 2,7 - Acquisition and flywheel (wide zoom) ......................... A4-23
Figure A4-32 RLL 2,7 - Acquisition (zoomed in) .............................................. A4-23
Figure A4-33 RLL 2,7 - Flywheel (wide zoom) ................................................ A4-24
Figure A4-34 RLL 2,7 - Flywheel (zoomed in) ................................................. A4-24

A-vii
LIST OF FIGURES FOR APPENDICES

Figure A4-35 RLL 2,7 - Acquisition with applied jitter ........................................ A4-25
Figure A4-36 RLL 2,7 - Flywheel with applied jitter (zoomed in) ..................... A4-25
Figure A4-37 NRZ Scrambled data - Clock recovery model (2-stage filter) .... A4-26
Figure A4-38 NRZ Scrambled - Acquisition over 2-stages ............................. A4-27
Figure A4-39 NRZ Scrambled - Flywheel over 2-stages ................................. A4-28

Figure A5-1 Test generator board ................................................................. A5-1
Figure A5-2 DSP Clock recovery board ........................................................ A5-2
Figure A5-3 Test DAC board ........................................................................ A5-3
Figure A5-4 IIR Filter - FPGA design ......................................................... A5-4
Figure A5-5 IIR Filter - Sub-hierarchy 16-bit subtractor ............................... A5-4

Figure A6-1 Flow diagram - Software overview ........................................ A6-1
Figure A6-2 DSP - Software algorithm overview ....................................... A6-3
Figure A6-3 BER curve ........................................................................... A6-11

Figure A7-1 Overview of adaptive simulation model ................................. A7-1
Figure A7-2 Data generator - RLL7 ......................................................... A7-2
Figure A7-3 Data generator - Manchester encoded .................................... A7-3
Figure A7-4 Data generator - RLL3 ......................................................... A7-4
Figure A7-5 Adaptive tracking system - Simulation model ........................ A7-5
Figure A7-6 Adaptive control block - Sub hierarchy detail ........................ A7-7
Figure A7-7 Error control signal for ±5% offset ..................................... A7-8
LIST OF FIGURES FOR APPENDICES

Figure A7-8 Acquisition performance ±5% range ............................................. A7-15
Figure A7-9 Acquisition performance ±7.5% range ........................................ A7-15
Figure A7-10 Acquisition performance ±10% range ....................................... A7-15
Figure A7-11 Manchester - Clock acquisition with a +5% offset ..................... A7-16
Figure A7-12 Manchester - Flywheel decay period with a +5% offset .......... A7-16
Figure A7-13 Manchester - 80 symbol dropout with a +5% offset ................ A7-17
Figure A7-14 Manchester - 80 symbol dropout, +5% offset (zoomed in) .... A7-17
Figure A7-15 Manchester - Tracking frequency drift over a ±5% range .... A7-18
Figure A7-16 Manchester - Tracking frequency drift over a ±5% range .... A7-18
Figure A7-17 Manchester - Clock acquisition with a +7% offset ............. A7-19
Figure A7-18 Manchester - Flywheel decay period with a +7% offset .......... A7-19
Figure A7-19 Manchester - 80 symbol dropout with a +7% offset ............ A7-20
Figure A7-20 Manchester - 80 symbol dropout, +7% offset (zoomed in) .... A7-20
Figure A7-21 Manchester - Tracking frequency drift over a ±7.5% range .... A7-21
Figure A7-22 Manchester - Tracking frequency drift over a ±7.5% range .... A7-21
Figure A7-23 Manchester - Clock acquisition with a +10% offset .......... A7-22
Figure A7-24 Manchester - Flywheel decay period with a +10% offset .......... A7-22
Figure A7-25 Manchester - 80 symbol dropout with a +10% offset .......... A7-23
Figure A7-26 Manchester - 80 symbol dropout, +10% offset (zoomed in) .... A7-23
Figure A7-27 Manchester - Tracking frequency drift over a ±10% range .... A7-24
Figure A7-28 Manchester - Tracking frequency drift over a ±10% range .... A7-24
Figure A7-29 RLL3 - Clock acquisition with a +5% offset ...................... A7-25
Figure A7-30 RLL3 - Flywheel decay period with a +5% offset .............. A7-25
LIST OF FIGURES FOR APPENDICES

Figure A7-31 RLL3 - 80 symbol dropout with a +5% offset ......................... A7-26
Figure A7-32 RLL3 - 80 symbol dropout with a +5% offset (zoomed in) .......... A7-26
Figure A7-33 RLL3 - Tracking frequency drift over a ±5% range .................. A7-27
Figure A7-34 RLL3 - Tracking frequency drift over a ±5% range ................. A7-27
Figure A7-35 RLL3 - Clock acquisition with a +7% offset ......................... A7-28
Figure A7-36 RLL3 - Flywheel decay period with a +7% offset .................. A7-28
Figure A7-37 RLL3 - 80 symbol dropout with a +7% offset ....................... A7-29
Figure A7-38 RLL3 - 80 symbol dropout with a +7% offset (zoomed in) ........ A7-29
Figure A7-39 RLL3 - Tracking frequency drift over a ±7.5% range .............. A7-30
Figure A7-40 RLL3 - Tracking frequency drift over a ±7.5% range ............. A7-30
Figure A7-41 RLL7 - Clock acquisition with a +5% offset ....................... A7-31
Figure A7-42 RLL7 - Flywheel decay period with a +5% offset .................. A7-31
Figure A7-43 RLL7 - 80 symbol dropout with a +5% offset ....................... A7-32
Figure A7-44 RLL7 - 80 symbol dropout with a +5% offset (zoomed in) ........ A7-32
Figure A7-45 RLL7 - Tracking frequency drift over a ±5% range ............... A7-33
Figure A7-46 RLL7 - Tracking frequency drift over ±5% range ................. A7-33
Figure A7-47 Sine input - Adaptive tracking simulation model ................... A7-34
Figure A7-48 Sine - Adaptive filter tracking drifting frequency ................. A7-35
Figure A7-49 Sine - Tracking frequency drift over ±10% range ................. A7-36

Figure A8-1 Flow diagram for a 4-point DFT ........................................ A8-7
Figure A8-2 Standard flow diagram for a 4-point DFT ............................. A8-7

A-x
LIST OF FIGURES FOR APPENDICES

Figure A9-1 Overview of the OFFT-based adaptive simulation model ............. A9-1
Figure A9-2 OFFT-based system - Simulation model................................. A9-2
Figure A9-3 “FREQ EST” block - Sub hierarchy detail............................ A9-3
Figure A9-4 “FREQ to COEF” block - Sub hierarchy detail ....................... A9-4
Figure A9-5 Simulation of OFFT-based clock recovery ............................ A9-6

Figure A10-1 Dual-processor DSP board................................................ A10-1
Figure A10-2 Dual-processor DSP board – Block diagram......................... A10-4
Figure A10-3 Dual-processor DSP board - Schematic diagram.................... A10-5
Figure A10-4 Dual-processor DSP board - Assembly diagram..................... A10-7
Figure A10-5 Dual-processor DSP board - Memory maps ............................ A10-8
List of Tables for Appendices

Table A3-1 Baud rate generator DIL switch settings ........................................ A3-8

Table A6-1 Filter characteristics ........................................................................ A6-9
Table A6-2 Filter coefficient quantization .......................................................... A6-9
Table A6-3 Flywheel performance .................................................................... A6-10

Table A7-1 Detection filter coefficients .............................................................. A7-9
Table A7-2 Error control signal for centre bin positions ±10% range .............. A7-9
Table A7-3 Error control signal for centre bin positions ±7.5% range .............. A7-10
Table A7-4 Error control signal for centre bin positions ±5% range .............. A7-10
Table A7-5 Adaptive control threshold parameters ±10% range .................... A7-11
Table A7-6 Adaptive control threshold parameters ±7.5% range .................... A7-12
Table A7-7 Adaptive control threshold parameters ±5% range .................... A7-12
Table A7-8 Coefficient and gain parameters ..................................................... A7-13
Phase-Locked Loop theory

This appendix supplements Section 1.1.2, and reviews the basic concepts of the Phase-Locked Loop (PLL). To aid understanding of the terminology, a glossary of commonly used PLL terms are provided. A simple PLL is then analysed to consider both its static and dynamic behaviour and to establish its limitations. Basic loop equations are provided, however for more detailed analysis readers are referred to the numerous dedicated books and papers on the subject, for example Gardener [15] and Gupta [46].

Glossary of PLL terminology

**Acquisition Range** ($\Delta \omega_{\text{acq}}$): Also known as the Capture Range. The frequency range ($\Delta \omega$) over which a loop can lock to an incoming signal. Although a loop may remain locked throughout its Lock Range, it may not be able to lock at the extremes.

**Damping Factor** ($\zeta$): The standard damping constant of a second-order feedback system. In a PLL, it refers to the ability to respond quickly to an input signal frequency, without excessive overshoot. In a critically damped system $\zeta=0.707$.

**Free Running Frequency** ($\omega_{\text{FR}}$): Also know as the Centre Frequency. The frequency at which the loop operates when not locked to an input signal. Ideally this should be the centre frequency of the VCO.
**Lock Range**: The range of frequencies over which a loop remains in lock.

**Lock-In Range** ($\Delta\omega_L$): Is related to the *Acquisition Range* and describes how close a signal must be to the centre frequency before capture can occur.

**Loop Gain** ($K$): The product of the dc gains of each element in the loop (KHz/rad).

**Loop Noise Bandwidth**: Related to the *Damping* and *Natural* frequency and describes the effective bandwidth of the received signal. Noise and signal components outside this band are highly attenuated.

**Low Pass Filter** (LPF): A filter that only allows dc and low frequencies to travel around the loop. The LPF controls the *Acquisition Range* and also the out of band noise rejection characteristics. The LPF has a voltage transfer function ($F_{LPF}$), which provides voltage gain, related to the phase error.

**Natural Frequency** ($\omega_n$): The characteristic of the loop, determined mathematically by the pole positions in the complex plane. It can also be determined experimentally, as the modulation frequency at which an under-damped loop gives its maximum output, at which point the phase error swing is the greatest.
Tracking Range ($\Delta \omega_c$): Also known as the Hold-in range, is related to the Lock Range and refers to how much the loop frequency can deviate from the centre frequency.

Phase Detector Gain Factor ($K_{PD}$): The conversion factor relating the phase detector output voltage, and the phase difference between the input and the VCO signal.

Phase Detector (PD): A circuit that compares the input, with the signal from the VCO. The resulting phase error signal ($\phi_e$), is filtered to produce a dc voltage to control the VCO frequency.

Voltage Controlled Oscillator (VCO): A circuit that oscillates at a frequency controlled by a dc voltage. The VCO is related to the Centre frequency and has a limited frequency range. The VCO has gain ($K_{VCO}$) and therefore contributes to the Loop Gain.
**Basic operation of a PLL**

A phase-locked loop is a feedback system that operates on the change in phase of a nominally periodic signal. Shown in Figure A1-1 is a simple PLL, consisting of a Phase Detector (PD), a Low Pass Filter (LPF) and a Voltage Controlled Oscillator (VCO). The PD provides phase error detection between the input signal, $x(t)$ and the feedback signal from the VCO, $y(t)$.

![Figure A1-1 Basic phase-locked loop](image)

The loop is considered "phase-locked" if all the signals around the loop have settled to a steady state, a result of which, is that the VCO oscillates at a frequency equal to the input frequency, but with a phase difference of $\Delta \phi$.

Figure A1-2 shows the various signals within the PLL. The PD generates pulses whose widths are equal to the phase difference $\Delta \phi$, between the input and output signals. These pulses are low-pass filtered to produce an averaged dc voltage to control the frequency of the VCO. The VCO then continues to oscillate at a frequency so as to sustain this static phase error, thereby maintaining the same frequency as the input signal. As an insight into the tracking behaviour of a PLL, consider a small increase of the input frequency. The pulses at the PD output
become wider, which in turn produce a larger dc control voltage at the LPF output, thereby increasing the VCO frequency towards that of the input. Eventually, the VCO frequency is once again identical to the new input frequency and the settled PD pulses are averaged by the LPF, generating a slightly larger dc control voltage so as to sustain the VCO at the new frequency.

![PLL Waveforms](image-url)

**Figure A1-2 PLL Waveforms**

The loop must acquire both in terms of frequency and phase, in order to achieve lock. The PLL has a limited tracking range and if the input frequency deviates too far away from the VCO’s centre frequency, then the loop may fall out of lock. The loop will only track, if all points around the loop vary monotonically. The loop also has a limited capture range, which is usually less than the tracking range. If the PLL is initially out of lock, for example when no input has been applied, and then an input frequency is applied which is beyond the loops capture range, then the loop may never achieve lock. The loops capture range is controlled by the bandwidth of the loop filter.
Loop dynamics in a locked state

Figure A1-3 shows a linear model of a PLL in lock, along with the transfer function of each block. The input signal has a phase of $\phi_{in}(t)$ and the VCO output has a phase $\phi_{out}(t)$. Assuming that the phase detector is linear and that the PD output voltage is proportional to the phase difference, then:

$$V_{PD} = K_{PD} (\phi_{in} - \phi_{out})$$ .......................... (A1-1)

Where $K_{PD}$ is known as the phase detector gain factor and has units of volts/radian.

The phase error voltage $V_{PD}$, is filtered by the loop filter, where noise and high frequency components are suppressed. The filter also determines the loop’s dynamic performance. $F(s)$ gives the filter’s transfer function. The control voltage $V_c$ from the LPF, determines the frequency of the VCO. Deviation of the VCO from its centre frequency is $\Delta \omega = K_{VCO} V_c$, where $K_{VCO}$ is the VCO gain factor and has units of rad/sec-V. Since frequency is the derivative of phase, the VCO operation may be described as $d\phi_{out}/dt = K_{VCO} V_c$.

By taking the Laplace transform [104]:

$$L \left[ \frac{d\phi_{out}(t)}{dt} \right] = s\phi_{out}(s) = K_{VCO} V_c(s)$$ .......................... (A1-2)
Hence, the phase of the VCO output is linearly related to the integral of the control voltage. By using Laplace notation the following equations apply.

\[ V_{PD}(s) = K_{PD}[\phi_{in}(s) - \phi_{out}(s)] \]  \hspace{1cm} (A1-3)

\[ V_c(s) = F_{LPF}(s)V_{PD}(s) \]  \hspace{1cm} (A1-4)

\[ \phi_{out}(s) = \frac{K_{VCO}V_c(s)}{s} \]  \hspace{1cm} (A1-5)

Combination of these equations yield the basic loop equations.

The open-loop transfer function of the PLL, is therefore given by:

\[ H_o(s) = K_{PD}F_{LPF}(s)\frac{K_{VCO}}{s} \]  \hspace{1cm} (A1-6)

The closed-loop transfer function is given by:

\[ H(s) = \frac{\phi_{out}(s)}{\phi_{in}(s)} \]

\[ H(s) = \frac{K_{PD}K_{VCO}F_{LPF}(s)}{s + K_{PD}K_{VCO}F_{LPF}(s)} \]  \hspace{1cm} (A1-7)

The phase error transfer function is given by:

\[ H_e(s) = \frac{\phi_e(s)}{\phi_{in}(s)} \]

\[ H_e(s) = \frac{s}{s + K_{VCO}K_{PD}F_{LPF}(s)} \]

\[ H_e(s) = 1 - H(s) \]  \hspace{1cm} (A1-8)

The VCO control voltage is given by:

\[ V_c(s) = \frac{sK_{PD}F_{LPF}(s)\phi_{in}(s)}{s + K_{VCO}K_{PD}F_{LPF}(s)} = \frac{\phi_{in}(s)s}{K_{VCO}}H(s) \]  \hspace{1cm} (A1-9)

A1-7
In its simplest form, a low-pass filter is implemented as in Figure A1-4.

The magnitude of the frequency response is given by:

$$|F_{LPF}(j\omega)| = \frac{1}{\sqrt{1 + \omega^2 R^2 C^2}} \quad \text{(A1-10)}$$

And the phase is

$$\phi(j\omega) = -\tan^{-1} \omega CR \quad \text{(A1-11)}$$

The open-loop gain is given by:

$$F_{LPF}(s) = \frac{1}{1 + \frac{s}{\omega_{LPF}}} \quad \text{(A1-12)}$$

Where $\omega_{LPF} = 1/(RC)$.

The closed-loop transfer function now becomes:

$$H(s) = \frac{\frac{K_{pd} K_{VCO}}{s^2}}{\omega_{LPF} s + s + \frac{K_{pd} K_{VCO}}{\omega_{LPF}}} \quad \text{(A1-13)}$$
The system is of second order, with one pole contributed by the VCO and another by the LPF. The Loop Gain is $K = K_{PD} K_{VCO}$ and is expressed in rad/sec.

In order to understand the dynamic behaviour of the PLL, the denominator of equation (A1-13) is converted to a form used in control theory:

$$s^2 + 2\zeta \omega_n s + \omega_n^2$$

where $\zeta$ is the damping factor, and $\omega_n$ is the natural frequency of the system.

Hence, the Closed Loop Transfer Function $H(s)$, becomes:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} \quad \text{(A1-14)}$$

Where:

$$\omega_n = \sqrt{\omega_{LPF} K}, \quad \zeta = \frac{1}{2} \sqrt{\frac{\omega_{LPF}}{K}} \quad \text{and} \quad K = K_{PD} \cdot K_{VCO}$$

The damping factor is inversely proportional to the loop gain, which is an undesirable trade off, since the loop gain cannot be increased to reduce the static phase error without degrading the settling time. In a well designed second order system, the damping factor $\zeta = 0.707$, so as to provide an optimally flat frequency response. Therefore, $K$ and $\omega_{LPF}$ cannot be chosen independently.

For example, if $\zeta = 0.707$ then $K = \omega_{LPF}/2$. In a practical system, the LPF bandwidth is usually kept narrow to reduce noise and this results in a narrow capture range.

The transfer function in equation (A1-14) is that of a low-pass filter, suggesting that if the input excess phase varies slowly, then the output excess phase follows. Conversely, if the input excess phase varies rapidly, as in a frequency step, then the output phase will vary initially by only a small amount.
Therefore, if the signal, $s \rightarrow 0$, then the Closed Loop Transfer Function, $H(s) \rightarrow 1$ and the static phase shift at the input is transferred unchanged to the output. This is because for phase quantities, the presence of integration in the VCO makes the open-loop gain, approach infinity as $s \rightarrow 0$. The phase error transfer function, $H_e(s)$ is defined as $H_e(s) = \phi_e(s)/\phi_{in}(s)$ in Figure A1-3.

$$H_e(s) = 1 - H(s)$$

Substituting equation (A1-14) for $H(s)$ yields:

$$H_e(s) = \frac{s^2 + 2\zeta\omega_n s}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

(A1-15)

Now consider the loop step response, with the aid of equation (A1-14). The input excess frequency is $\Delta\omega u(t)$, where $u(t)$ is the unit step function. The output frequency then exhibits a typical step response of a second-order system, eventually settling to $\Delta\omega$ rad/s higher than its initial value.

The output phase, on the other hand is given by:

$$\phi_{out}(s) = H(s)\phi_{in}(s)$$

Substituting equation (A1-14) for $H(s)$ yields:

$$\phi_{out}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \frac{\Delta\omega}{s^2}$$

(A1-16)

Which is the response of a second-order system to a ramp input.

The phase error is given by:

$$\phi_e(s) = H_e(s)\phi_{in}(s)$$

$$\phi_e(s) = \frac{s^2 + 2\zeta\omega_n s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \frac{\Delta\omega}{s^2}$$

A1-10
Whose final value is given by:

\[ \phi_s(t = \infty) = \frac{\Delta \omega}{K} \]  

(A1-17)

Therefore, static changes in the input frequency are suppressed by a factor K, which appear as a static phase error, as shown in Figure A1-5.

![Figure A1-5 Response of a PLL to a frequency step](image)

An important drawback of the PLL considered so far, is the direct relationship between \( \zeta \), \( \omega_{LPF} \) and \( K \). For example, if the gain is increased to improve acquisition time or to reduce the static phase error, then the settling behaviour degrades.

In order to allow independent choice of \( K \) and \( \omega_{LPF} \), a zero can be added to the low-pass filter, as shown in Figure A1-6.

The magnitude of the frequency response for this lead-lag filter is given by:

\[ |F_{LPF}(j\omega)| = \sqrt{\frac{1 + \omega^2 R_2^2 C^2}{1 + \omega^2 C^2 (R_1 + R_2)^2}} \]  

(A1-18)

And the phase is \( \phi(j\omega) = \tan^{-1} \omega R_2 C - \tan^{-1} \omega C(R_1 + R_2) \)  

(A1-19)
Figure A1-6 Lag-lead low pass filter

Modifying the open loop transfer function to:

\[ F_{\text{LPF}}(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{s \cdot R_2C + 1}{s \cdot (R_1 + R_2)C + 1} \]  \hspace{1cm} (A1-20)

Hence, substituting \( F_{\text{LPF}}(s) \) into (A1-7), the PLL transfer function becomes:

\[ H(s) = \frac{K_{\text{VCO}}K_{\text{PD}}(s \tau_2 + 1)}{s^2 + s(1 + K_{\text{VCO}}K_{\text{PD}} \tau_2) + K_{\text{VCO}}K_{\text{PD}}} \]  \hspace{1cm} (A1-21)

Where \( \tau_1 = (R_1 + R_2)C \), \( \tau_2 = R_2C \) and \( \omega_{\text{pole}} = 1/((R_1 + R_2)C) \) \( \omega_{\text{zero}} = 1/(R_2C) \).

For an active filter with a very large gain, then \( H(s) \) can be rewritten:

\[ H(s) = \frac{s^2 + 2\zeta \omega_n s + \omega_n^2}{s^2 + s(2\zeta \omega_n - \omega_n^2) + \omega_n^2} \]  \hspace{1cm} (A1-22)
The damping factor is then equal to:

$$\zeta = \frac{1}{2} \sqrt{\frac{K_{\text{VCO}} K_{PD}}{\tau_1} \left( \frac{1}{\tau_2} + \frac{1}{K_{\text{VCO}} K_{PD}} \right)}$$  \hspace{1cm} \text{(A1-23)}$$

And the natural frequency is given by:

$$\omega_n = \sqrt{\frac{K_{\text{VCO}} K_{PD}}{\tau_1}}$$  \hspace{1cm} \text{(A1-24)}$$

Note however, that by adding a zero in the lag-lead filter gives two side effects:

- The -3dB bandwidth of the system also increases, which is an undesirable feature.

- The LPF attenuation of high frequencies is only $\frac{R_2}{R_1+R_2}$, which can cause instability problems. To alleviate this effect, a second capacitor can be connected across the filter output to provide another pole beyond the zero. However, this results in a slower settling time and the optimum damping value may no longer be $\zeta = 0.707$.

**Stability**

A PLL, as for any feedback system, can oscillate if its open-loop gain exceeds unity and simultaneously, its open-loop phase shift exceeds 180°. At least one of the closed-loop poles of an unstable loop will lie in the right half of the s-plane. A bode plot for a second order loop is shown in Figure A1-7.

The VCO output phase is linearly related to the integral of the control voltage. The VCO phase-shift, combined with a further phase-shift introduced by the LPF, can cause the loop to approach instability. At very low frequencies, the VCO integration is dominant, so the amplitude slope is -6dB/octive and the phase is -90°. The pole
of the loop filter introduces another lag at \( \omega = 1/C(R_1 + R_2) \). The slope then becomes 

\[ -12 \text{dB/octave} \]

and the loop phase approaches 180°.

![Diagram](image)

**Figure A1-7** Bode plot of a second order loop with a lag-lead filter

The stabilising zero in the lag-lead filter introduces a lead that causes the slope to revert to 

\[ -6 \text{dB/octave} \]

and the phase approaches -90° for high frequencies. The break in the slope occurs at \( \omega = 1/R_2 C \). The natural frequency \( \omega_n \) is the frequency at which the extension of the -12dB/octave segment crosses the unity gain coordinate. Placing the lead break at the unity gain point, yields a damping factor \( \zeta = 0.5 \). Since smaller damping is rarely wanted, the break is placed above unity gain.

Crossover of the final -6dB/octave segment occurs at a frequency \( \omega = K = 2\zeta \omega_n \).
Tracking behaviour

Figure A1-5 shows that the PLL can track a change in input frequency, but there is a limit to the tracking range. To analyse the tracking range, consider conditions when the input varies both slowly, "static tracking" and abruptly "dynamic tracking." The tracking behaviour is different for the two cases.

Starting with the VCO free-running frequency, if the input frequency varies slowly, such that the difference between $\omega_{in}$ and $\omega_{out}$ always remains much less than $\omega_{LPF}$. Then to allow tracking, the magnitude of the VCO control voltage, and hence the static phase error, must increase. The PLL tracks, as long as the three parameters plotted in Figure A1-8 vary monotonically. In other words, the edge of the tracking range is reached at the point where the slope of one characteristic falls to zero or changes sign, and this can occur in the PD or the VCO.
Shown in Figure A1-9 are examples of such behaviour. The VCO typically has a limited frequency range, beyond which the gain drops sharply. Also, the PD characteristic becomes non-monotonic for a large input phase difference, at which point the PLL fails to maintain lock. For a multiplier PD, the gain changes sign if the input phase difference deviates from its nominal value by more than 90°. Hence, the VCO output frequency can deviate from its free-running frequency by no more than:

$$\Delta \omega_f = K_{PD} \left( \sin \frac{\pi}{2} \right) K_{VCO}$$  \hspace{1cm} (A1-25)

Therefore, the static tracking range of a PLL employing a sinusoidal PD is the smaller of $K$ and half the VCO output frequency range.

Now consider the tracking behaviour of a PLL to a step change in input frequency. Consider an input frequency to a PLL that is initially locked at the VCO’s free running frequency and is then stepped by $\Delta \omega$. What is the maximum deviation of $\Delta \omega$ for which the loop can regain lock.
For any input frequency step at its input, a PLL temporarily loses lock and requires several cycles to re-acquire. However, for a small change in frequency, the loop locks again quickly, and the transient can be viewed as one of tracking, rather than locking. The key point resulting from this observation, is that the following two situations are similar:

- A loop initially locked at $\omega_{FR}$ experiences a large frequency step, $\Delta \omega$.
- A loop initially unlocked and free running must lock onto an input frequency given by $|\omega_n - \omega_{FR}| = \Delta \omega$.

In both cases, the loop must "acquire lock".

**Acquisition of lock**

The "acquisition range" or "capture range" is the maximum value of frequency deviation for which the loop can lock. Acquisition range is a critical parameter, because it trades directly with the loop bandwidth. Normally a narrow loop bandwidth is required to reject noise, however this will produce a proportionally small acquisition range, resulting in conflicting requirements.

First consider a loop that is opened at the VCO output, and the feedback signal replaced by a local oscillator, at a frequency $\omega_{FR}$. The output of the LPF is then a sinusoid at $\omega_n - \omega_{FR}$. As the sinusoid instantaneous amplitude increases, so does the VCO frequency and vice-versa, producing a frequency-modulated beat. Now if the loop is closed, the feedback signal has a time varying frequency. When the LPF output goes through a positive excursion, $\omega_{out}$ approaches $\omega_n$ and the beat period
increases. Conversely, when the LPF output becomes negative, \( \omega_{out} \) moves away from \( \omega_{in} \) and the beat period decreases.

Figure A1-10 shows the resulting waveform at the LPF output, exhibiting longer positive cycles than negative, thus carrying a positive dc component, which gradually shifts the average value of the output frequency \( \omega_{out} \) closer to input frequency \( \omega_{in} \). If \( \omega_{in} \) is sufficiently close to the VCO's free running frequency \( \omega_{FR} \), then frequency acquisition is achieved at the first peak of the beat waveform, and the PLL is locked with no "cycle slips." If however, \( \omega_{in} \) is too far away from \( \omega_{FR} \), then the beat waveform has little asymmetry and hence, not enough dc voltage is produced to drive the loop towards lock.

Figure A1-10 shows how the peak of the beat cycles gradually become more positive, and the period of each cycle slightly increases, as the average value of \( \omega_{out} \) comes closer to \( \omega_{in} \). Cycle slips can be observed only when \( \omega_{in} \) is close to the edge of the acquisition range. Also, the number of cycle slips depend upon the loops initial condition, i.e. those in the LPF as well as the initial phase of the VCO.
APPENDIX I

To estimate the acquisition range a simplified case is considered, where the LPF is approximated as:

\[ V_{LPF}(t) = K_{PD} |F_{LPF}(j\Delta \omega)| \sin(\Delta \omega t) \]

This signal modulates the VCO frequency, causing maximum deviation:

\[ (\omega_{out} - \omega_{FR})_{\text{max}} = K_{PD} K_{VCO} |F_{LPF}(j\Delta \omega)| \]

If this deviation is equal or greater than \( \Delta \omega \), then the loop locks with no cycle slips.

If \( F_{LPF}(s) \) is known, \( \Delta \omega \) can be calculated.

For a simple low pass filter:

\[ \Delta \omega_{acq} = \left[ \frac{\omega_{LPF}^2}{2} \left( -1 + \sqrt{1 + \frac{1}{4\zeta^4}} \right) \right]^{1/2} \]  \hspace{1cm} \text{(A1-26)}

Which reduces to \( \Delta \omega_{acq} \approx 0.46 \omega_{LPF} \) if \( \zeta = 0.707 \).

Most modern PLL systems usually employ additional means of frequency acquisition, which significantly increases the capture range, often removing its dependence on \( K \) and \( \omega_{LPF} \), achieving limits close to that of the VCO.
Aided acquisition

The acquisition behaviour, shown in Figure A1-10 and formulated by equation (A1-18), indicates that the capture range of a simple PLL is roughly equal to \(0.5 \omega_{LPF}\), regardless of the magnitude of \(K\). Issues such as jitter suppression impose an upper bound on \(\omega_{LPF}\), and the resulting capture range is often inadequate. Therefore, most practical PLL’s employ additional techniques to aid frequency acquisition.

![Figure A1-11 Aided acquisition with a frequency detector.](image)

Figure A1-11 shows a PLL with aided frequency acquisition. The system utilises a Frequency Detector (FD) and a second low-pass filter, \(LPF_2\), whose output is added to that of \(LPF_1\). The FD produces an output having an average dc value proportional to the difference between the input and output frequencies. If this difference is large, the PD output has a negligible dc component and the VCO is driven by the dc output from the FD with negative feedback, thereby moving the output frequency towards the input. As the frequency difference drops, the dc output of the FD decreases, whereas that of the PD increases. Hence, the frequency detection loop gradually relinquishes control of the PLL, becoming inactive when \(\omega_{in} - \omega_{out} = 0\).
Appendix 2

PLL simulations

This appendix provides simulation models and results to evaluate the performance of a 2\textsuperscript{nd} Order PLL, for use in a clock recovery circuit. This appendix concludes with a summary of PLL characteristics.

Shown in Figure A2-1 is the detail of a PLL simulation model (a standard SPW complex PLL block), consisting of a multiplier Phase Detector (PD), a Butterworth Low Pass Filter (LPF) and a Voltage Controlled Oscillator (VCO). Since, the output is dependent upon the input; a sample delay (Z\textsuperscript{-1}) is provided in the feedback loop to provide isolation, thereby avoiding a simulation error due to an unknown state.

![Figure A2-1 SPW 2\textsuperscript{nd} Order PLL](image)

The PD provides phase error detection, between the complex input signal, and the feedback signal from the VCO. The phase error is averaged by a LPF to produce a dc voltage, which in turn controls the frequency, and hence phase of the VCO.

A2-1
If the input frequency and phase is exactly the same as the VCO's free running frequency, then the phase error will be zero and the LPF will produce zero dc control voltage, in order to sustain the VCO at its nominal frequency. If however, the two frequencies are equal but a phase error exist then the loop is not locked. In this condition, the VCO frequency must temporarily change and the loop must acquire both in terms of frequency and phase.

The simulation of Figure A2-2 demonstrates this behaviour, and shows the resulting delay in acquisition, while the loop settles to the locked state. Hence, it can be observed that the PLL is a feedback system, which operates on a change in phase and not frequency.

If the input frequency is now increased slightly, phase error pulses are produced at the PD output, which are averaged by the LPF to produce a dc voltage to pull the VCO, in such a way so as to reduce the frequency error. Unless the system is critically damped, there will be some overshoot as the loop converges towards a steady state, as illustrated in Figure A2-3.
When all the signals around the loop eventually settle to a steady state, the loop is considered "phase-locked", a result of which is that the VCO frequency is equal to the input frequency. However, at this point there is a static phase difference between the VCO output and the input signal, which when averaged by the LPF, produces a constant dc voltage to the VCO, in order to sustain the increase in frequency.

In clock recovery applications, the recovered clock at the VCO output will be synchronised to the input symbols, but a phase difference will exist with respect to the input data transitions. This static phase shift varies, depending upon the frequency offset. Hence, the phase of the recovered clock moves, with respect to the input symbols and detracts from re-sampling the symbols at the optimum sample point. Jitter in the retrieved symbols, combined with sub-optimum re-timing, results in data errors. To overcome this problem, either the recovered clock or the retrieved symbols should be delayed to provide optimum re-sampling, thereby maximising the jitter detection window.
If the applied input frequency is too far from the loop's natural frequency, then the loop may never achieve lock. The simulation of Figure A2-4 (bottom trace) demonstrates failure to acquire lock. The loop's capture and tracking range is limited by the VCOs operating range, the type of PD and the loop bandwidth. In practice, the capture and tracking range of a basic PLL is limited by the loop bandwidth.

![Diagram](image)

**Figure A2-4** Simulation of acquisition for different frequency offsets

For a fixed loop bandwidth, the system may be critically damped for offset frequencies near the limits of the capture range. However, for small frequency offsets the system is under-damped and acquisition takes longer, as shown in the top three simulation traces of Figure A2-4.
For clock recovery applications, the loop bandwidth needs to be narrow for noise suppression and to provide sufficient memory to flywheel over symbol run lengths. However, this considerably increases the acquisition time and the loop hunts for long periods before settling to a stable state.

The simulation of Figure A2-5 shows how a reduced loop bandwidth results in a relatively long acquisition time. The flywheel performance, controlled by the time-constant of the loop filter, is still quite short and only fly-wheels over about 7 input cycles. It is desirable to increase the flywheel period further, however this cannot be achieved without further degrading the acquisition performance.

Hence, these simulations demonstrate the conflicting requirements of a PLL. On the one hand a wide loop bandwidth is required, to achieve rapid acquisition and a wide capture range, but conversely, a narrow loop bandwidth is desired to provide adequate noise suppression and to flywheel over data run lengths or dropouts.
The graph of Figure A2-6 provides a collection of simulation data for a wide-band PLL, to measure the acquisition performance, relative to frequency offsets from the VCOs quiescent frequency. These simulations have been performed with a sinusoidal input on a 2nd order PLL, with the following normalised loop parameters: Loop bandwidth $f_{LPF} = 0.005\text{Hz}$, VCO constant $K_{VCO} = 0.00625\text{Hz/V}$ and VCO quiescent frequency $f_{FR} = 0.0625\text{Hz}$. The sample rate is normalised to 1Hz and the nominal input frequency has $1/0.0625 = 16$ samples/cycle. The VCO constant is chosen to provide a convenient measurement of percentage offset at the LPF output, given by:

$$V_C = \frac{f_n - f_{FR}}{K_{VCO}} = \frac{\Delta f}{K_{VCO}}$$

(1-27)

Hence, a 1% frequency offset ($\Delta f = 0.063125 - 0.0625$) is represented by a VCO control voltage at the LPF output of 0.1V, and 2% by 0.2V, etc.

Figure A2-6 Wide-band PLL acquisition performance for sine input
For this wide-band PLL, more than 100 input cycles are required in order to achieve lock, while the corresponding flywheel performance is poor, decaying to zero in just 7 cycles.

One may assume that for a step input (frequency burst), the closer the input frequency is to the VCOs nominal frequency, then the faster the acquisition time. However, these results show that this is not the case, since the loop is under-damped for small frequency offsets, requiring up to 840 input cycles to achieve lock.

The previous results are for a sinusoidal input, which is an ideal periodic signal onto which the PLL can lock. For clock recovery applications, the input signal are data symbols with runs of ones and zeros, the run length of which, is dependent upon the channel code. The LPF must have sufficient memory to filter these run lengths, preventing static variations in the control voltage at the VCO input.

Simulations using Manchester encoded data required the loop-bandwidth to be reduced to 0.0015Hz, as shown in the simulation of Figure A2-7. Whilst reducing the loop-bandwidth to improve averaging of the PD output, this further degrades the capture range to 2% and increases the acquisition time, requiring 625 input symbols in order to achieve lock. For burst data applications, this would necessitate an unacceptably long preamble.
Manchester encoded data provides a short run-length between transitions. Hence, the loop filter need only contain short-term memory in order to average the PD output. However, for codes with long run-lengths, a long time-constant is required. Hence, the loop bandwidth must be reduced, further degrading the capture range to less than 1%.

These results demonstrate the undesirable trade-offs of the basic PLL clock recovery system.
Figure A2-8 Simulation model - PLL with data input.

The SPW simulation model is shown in Figure A2-8. Simulation results have revealed that the SPW standard "Random Data" block is sample rate driven and cannot be used for generating data, with frequency offsets which are not integer sub-multiples of the sample rate.
Whereas, the frequency parameter of an SPW “Signal Generator” block can be set to any rate. Hence, a PRBS generator has been developed, with a controllable run length (nominally RL=7), driven by a “Signal Generator” (square wave) block.

Also, since the standard SPW PLL block is a complex model, a Hilbert Filter (unity gain and 90° phase shift) has been used to provide an imaginary signal component for input to the PLL. To stimulate the PLL during acquisition, a preamble is applied prior to the PRBS data. The RLL PRBS data can be Manchester encoded, or fed directly to the clock recovery circuit.

Figure A2-9 PLL Clock recovery – Schematic diagram

Figure A2-9 shows a schematic diagram of a PLL clock recovery system, used to verify the simulation results. The PLL circuit interfaces directly to a test generator board, described in Appendix 3. The test generator sources test recording codes, at selectable data rates, for input to the PLL clock recovery circuit.
Summary of PLL characteristics

Following theoretical analysis of the PLL in Appendix 1 and the results presented in Appendix 2, the following key characteristics are noted:

- A PLL operates on a nominally periodic and continuous signal. Hence, in burst mode applications, a long preamble is required in order to achieve lock.

- A PLL operates on phase detection and not a frequency difference. The input frequency could be equal to the VCO frequency, but the loop may not be locked, in which case the VCO frequency must temporarily change to enable phase acquisition.

- A PLL changes dimensions around the loop. The input is converted: to phase at the PD, then to voltage by the LPF and finally back to phase by the VCO. This can make analysis difficult, compared to conventional feedback systems.

- A PLL is a feedback system, and as with any feedback system the laws of stability apply. If the open loop gain exceeds unity, while at the same time the open loop phase shift exceeds 180°, then the loop will oscillate.

- The PLL contains short-term memory in the form of a LPF and as such takes time to respond to a change in the input.

- It is desirable to keep the loop bandwidth narrow to reject high frequency noise and to flywheel over periods of lost input. However, this compromises the capture range and acquisition performance, hence there are conflicting requirements.
Fixed loop parameters designed for a critically damped system with a wide capture range, results in an under-damped system for small offsets, degrading acquisition performance for small frequency offsets.

The damping factor is inversely proportional to the loop gain. Hence, if the gain is increased to reduce the static phase error, then the loop becomes under-damped and acquisition takes longer.

If the PLL initially has no input signal, as in burst-mode applications, then the output frequency drifts to the VCO extreme (depending upon the type of PD). This can further compromise acquisition performance, when a burst-input signal is re-applied.

Depending upon the frequency offset, varying static phase error exists, with respect to the input signal. For data recovery applications, this changes the sampling point, and further processing is necessary to sample at the centre of the data eye.

To aid acquisition, a combination of frequency and phase detection can be applied. FD provides a coarse error signal to aid acquisition, relinquishing control to the PD to provide fine phase tracking control. Whilst this improves acquisition performance, the combination of loops further complicates analysis.
Appendix 3

Recording codes

This appendix contains simulation models and results, relating to the test recording codes (Manchester, Miller, Miller\(^2\) and RLL 2,7), as described in Section 1.2.2. A test generator board has been developed for real-time testing of the clock recovery system, where the test recording codes are implemented in an FPGA. The test generator board and the FPGA design are also described.

Manchester and Miller codes

Standard SPW library blocks are used for simulation, using the Manchester and Miller codes. For FPGA implementation, the Manchester code is straightforward. The Miller code is more involved and the logic test circuit is shown in Figure A3-1.

![Figure A3-1 Miller code - Test circuit for FPGA](image)
Miller\(^2\) code

The Miller\(^2\) code required development for both simulation and FPGA implementation. Simulation test results revealed an error in the original patent [32], relating to the logic design of the recommended encoder, as shown in Figure A3-2.

![Figure A3-2 Extract from Miller\(^2\) patent](image-url)
The logic diagram of Figure A3-2 fails to produce the correct coding, as shown in the supporting timing diagram. Specifically, gate 66 resets flip-flop 56 prematurely, when $\phi_1$ clock (signal A), current Bit Do (signal D) and suppressed pulse S (signal J) are all high, as indicated by the dashed line on the timing diagram. This results in a miss count of the Pz (zeros since last suppressed one) counter and failure to suppress the last one of a sequence type-C. To overcome this problem the encoder has been modified, such that gate 66 is also gated with the previous bit, D0-1, which must be a zero before a reset to the Pz counter is enabled. Logic simulation re-runs with this modification, confirmed correct Miller$^2$ encoding. The resulting Miller$^2$ test circuit for the FPGA is shown in Figure A3-3.
Figure A3-4 Miller\textsuperscript{2} code - Test model and simulation

Figure A3-4 shows the developed Miller\textsuperscript{2} sub hierarchy detail, which has been compiled and linked to a higher level symbol. SPW\textsuperscript{s} libraries do not contain JK and D type flip-flops, so these blocks have also been created from primitive gates. The bottom trace of the simulation shows the DSV bound, as per correct Miller\textsuperscript{2} encoding.
RLL 2,7 code

The RLL 2,7 code required development for both simulation and FPGA implementation. Figure A3-5 shows the RLL 2,7 logic test circuit for use in the FPGA.

Figure A3-5 RLL 2,7 code - Test circuit for FPGA

Figure A3-6 shows the SPW RLL 2,7 sub hierarchy detail, which has been linked to a higher level symbol. In this test circuit, the RLL 2,7 code is followed with an NRZI encoder, as described in Section 1.2.2.1. The simulation results confirm correct RLL 2,7 encoding.
Figure A3-6 RLL 2,7 code - Test model and simulation
RLL Test generator board.

To assist evaluation of clock recovery performance in real-time, a test generator has been developed, providing the test recording codes, as described in Section 1.2.2.

Figure A3-7 Block diagram of RLL test generator board

Figure A3-7 provides a block diagram of the board, where all four test codes have been implemented in a single Xilinx FPGA.

The test generator board contains a crystal controlled baud rate generator providing a 16 times the data rate clock. The baud rate DIL switch settings are shown in Table A3-1. The x16 clock is further divided to produce a data rate clock and is used to clock a PRBS generator, which provides a random data source for RLL encoding.
<table>
<thead>
<tr>
<th>S4</th>
<th>S3</th>
<th>S2</th>
<th>S1</th>
<th>Baud Rate</th>
<th>Actual Output (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>50</td>
<td>800</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>75</td>
<td>1,200</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>134.5</td>
<td>2,152</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>200</td>
<td>3,200</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>600</td>
<td>9,600</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>2,400</td>
<td>38,400</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>9,600</td>
<td>153,600</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>4,800</td>
<td>76,800</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>1,800</td>
<td>2,880</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>1,200</td>
<td>19,200</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>2,400</td>
<td>38,400</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>300</td>
<td>4,800</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>150</td>
<td>2,400</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>110</td>
<td>1,760</td>
</tr>
</tbody>
</table>

Table A3-1 Baud rate generator DIL switch settings

Alternatively, phase-reversal data can be selected, or data can be input from an external source. The selected data source is input to a Xilinx 3020PC68-70 FPGA, where the NRZ data are simultaneously encoded into the four test codes. An adjustable link selects one of the four codes, which is buffered off-board for connection to the clock recovery system under test.

A data dropout of between 1 and 1024 symbols can be introduced, to evaluate the flywheel performance of the clock recovery system. A binary weighted DIL switch (SW5), sets the dropout period from 1 to 128 symbols and a slider switch (SW4), sets the multiplication factor, x1 to x8. The dropout is initiated with a push button switch (SW3). A data inhibit switch (SW2), continuously inhibits encoded data, in order that the flywheel decay period of the clock recovery system can be measured.
APPENDIX 3  
RECORDING CODES

Figure A3-8 shows a photograph of the assembled printed circuit board, while Figure A3-9 provides the corresponding schematic diagram.

**Figure A3-8 RLL code - Test generator board**

![Test generator board](image)

**Figure A3-9 Schematic of RLL test generator board.**

Figure A3-10, shows the FPGA schematic diagram combining all four test codes, together with the resulting logic simulation. The NRZ data input can be compared with the encoded outputs. A zoomed-in view reveals glitches in the Manchester code. The outputs are therefore re-timed within the FPGA, with a common clock to eliminate glitches and phase noise due to routing delays.
Figure A3-10 Combined codes and timing for FPGA
Appendix 4

Fixed frequency clock recovery

This appendix supplements Chapter 2, and provides full simulation results to evaluate performance of the fixed frequency clock recovery algorithm. Clock acquisition, flywheel and jitter rejection results are presented for each of the test codes (Manchester, Miller, Miller\(^2\), RLL 2,7 and Scrambled NRZ). The developed "Jitter Generator" and "PDF" blocks, used for jitter analysis, are described. Test simulation models and results are provided to evaluate these new blocks.

Jitter generator

The jitter generator developed imposes timing errors of ±3 samples, which represents 37.5% applied jitter at a symbol rate of 16 samples/symbol. The weighting of the jitter generator can be adjusted as required, but for simulation purposes a gaussian distribution has been applied to represent typical channel noise.

![Figure A4-1 Jitter generator - Switch settings](image-url)
The jitter generator consists of a delay-line of 6 samples. The middle tap is considered as the nominal timing point, and multiplexers select either early or late samples. Random generators control the multiplexers, where the probability of zero can be weighted. Hence, the weighting can be adjusted to provide the required jitter distribution.

The ratios of the multiplexers must be carefully considered when calculating the weightings. For example, switch \( a \) must be biased, such that switches \( b \) and \( c \) are given proportionate amounts of time. The jitter generator can be divided into seven bins, as represented by the histogram in Figure A4-2. For jitter with an even distribution, each bin will contain the same values, i.e. \( \frac{1}{7} = 0.1428 \).

![Figure A4-2 Jitter histogram - Uniform distribution](image)

Switch \( x \) is always set to a 1:1 ratio, providing 50% probability of selecting early or late samples.
The bin values and corresponding switch settings are summarised by the following equations.

\[ A = (a_1 x_1) + (a_2 x_2) = (\text{early}) + (\text{late}) \]  \hspace{1cm} (A4-1)

\[ B = b_1 (1 - a_1) (1 - x_1) \]  \hspace{1cm} (A4-2)

\[ C = c_1 (1 - b_1) (1 - a_1) (1 - x_1) \]  \hspace{1cm} (A4-3)

\[ D = (1 - c_1) (1 - b_1) (1 - a_1) (1 - x_1) \]  \hspace{1cm} (A4-4)

If the bin values are known, then the equations are transposed to determine the switch settings.

\[ x_1 = x_2 = 0.5, \text{ therefore } (1 - x_1) = (1 - x_2) = 0.5 \]

\[ a_1 = A \]

\[ a_2 = 1 - a_1 \]

\[ b_2 = 1 - b_1 \]

\[ c_2 = 1 - c_1 \]

\[ d_i = c_2 = \frac{D}{(1 - b_1) (1 - a_1) (0.5)} = (1 - c_1) \]

Substituting 0.1428 (1/7) for bins A, B, C and D into the above equations, results in the following switch weightings for uniform distribution.

\[ x_1 = x_2 = 0.5, a_1 = 0.1428, a_2 = 0.8572, b_1 = 0.333, b_2 = 0.666, c_1 = 0.5, c_2 = 0.5 \]

The standard normal curve for gaussian distribution is shown in Figure A4-3, and can be represented by the equation:

\[ y = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} d \]  \hspace{1cm} (A4-5)
The probability of a randomly selected value of $z$ lying between $z = 0.5$ and $z = 1.5$ i.e. bin B, is given by:

$$P(0.5 \leq z \leq 1.5) = \int_{0.5}^{1.5} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \, dz$$  \hspace{1cm} (A4-6)

The integral can be evaluated by means of a look-up table [105] to determine the probability of $z$ lying within the histogram bands.

For the jitter generator, the curve is divided into seven histogram bins, where $z$ represents early or late jitter samples (-3 to +3). Based upon the area under the curve, the bin values are $A=0.383$, $B=0.2417$, $C=0.0606$ and $D=0.0057$.

Substituting the bin values into equations (A4-1) to (A4-4), result in the following switch weightings for a gaussian distribution.

$$x_1 = x_2 = 0.5, \quad a_1 = 0.383, \quad a_2 = 0.617, \quad b_1 = 0.783, \quad b_2 = 0.216, \quad c_1 = 0.907, \quad c_2 = 0.092$$

Values $x_j$, $a_j$, $b_j$ and $c_j$ are applied to the random generators (probability of zero parameter), which control the multiplexers in the jitter generator simulation detail, as shown in Figure A4-4.
APPENDIX 4

FIXED FREQUENCY CLOCK RECOVERY

JITTER GENERATOR PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Rate</td>
<td>1.0</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>0.40125</td>
</tr>
<tr>
<td>Jitter Control On</td>
<td>On</td>
</tr>
<tr>
<td>Jitter Weighting 0</td>
<td>0.5</td>
</tr>
<tr>
<td>Jitter Weighting 1</td>
<td>0.503</td>
</tr>
<tr>
<td>Jitter Weighting 2</td>
<td>0.703</td>
</tr>
<tr>
<td>Jitter Weighting 3</td>
<td>0.968</td>
</tr>
</tbody>
</table>

Figure A4-4 Jitter generator - Model and test simulation
PDF (Probability Density Function)

In order to determine the jitter rejection performance of the clock recovery system, it is necessary to statistically analyse the jitter in the recovered clock over a large number of symbols. SPW analysis page supports histograms, plotted as a function of amplitude. A PDF detail has therefore been developed to translate jitter information into amplitude, in order that SPW histogram analysis can be directly applied, to display the probability density function of jitter in the recovered clock.

The PDF block consists of a delay-line of 6 samples. The recovered clock is sampled into the delay-line at the sample rate (16 samples/symbol). The middle tap (A), is considered as the nominal timing point, representing zero phase error in the recovered clock. The output of the PDF block is sampled once per symbol period, at the nominal timing point. Each of the early or late taps has an edge detector, which generates a pulse of one sample duration, upon detecting a rising edge. Each tap is
translated to a corresponding amplitude level, positive for early phase errors and negative for late phase errors. An early phase error of 3 samples ($+67.5^\circ$) is translated to an amplitude of $+3$, while a late phase error of -3 samples ($-67.5^\circ$) is translated to an amplitude of $-3$. Similarly, phase errors of 1 sample ($\pm22.5^\circ$) and 2 samples ($\pm45^\circ$) are translated to $\pm1$ and $\pm2$ amplitude levels respectively.

The output of the PDF block provides a signal for the duration of the simulation with discrete amplitude levels representing jitter and hence phases errors in the recovered clock.

Figure A4-6 PDF detail and test simulation
Reference to the test simulation of Figure A4-6, shows the nominal clock overlaid. The amplitude and sign of the PDF output corresponds to the amount and direction of the phase error.

This signal is then be applied to SPWs histogram, for analysis of the PDF of phase error in the recovered clock. Figure A4-7 shows the resulting test histograms for applied jitter. This analysis has been simulated over 500,000 samples; an even longer simulation would result in a more accurate distribution. The gaussian distribution is applied to the clock recovery system and histograms produced, as illustrated in Chapter 2, Section 2.3.2.3

![Histogram analysis of jitter](image)

Figure A4-7 Histogram analysis of jitter
Simulation results

The following sections are a collection of key simulation results for the following codes: Manchester, Miller, Miller², RLL 2,7 and Scrambled NRZ (RL 15).

These results are discussed in Chapter 2 and are used to measure:

- Clock acquisition performance.
- Clock flywheel performance.
- Clock acquisition performance, with applied jitter.
- Clock flywheel performance, with applied jitter.
- Jitter rejection performance.

Manchester encoded data

![Clock Recovery System Parameters Table]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Rate</td>
<td>1.0</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>0.57129</td>
</tr>
<tr>
<td>Coefficient C</td>
<td>1.84444</td>
</tr>
<tr>
<td>Coefficient K</td>
<td>-0.39089</td>
</tr>
<tr>
<td>False in symbol</td>
<td>100.0</td>
</tr>
<tr>
<td>False in symbol</td>
<td>200.0</td>
</tr>
<tr>
<td>Filter Scale</td>
<td>45.0</td>
</tr>
</tbody>
</table>

![Clock Recovery System Diagram]

Figure A4-8 Manchester encoded - Clock recovery model
Figure A4-9 Manchester - Acquisition and flywheel (wide zoom)

Figure A4-10 Manchester - Acquisition (zoomed in)
Figure A4-11 Manchester - Flywheel (wide zoom)

Figure A4-12 Manchester - Flywheel (zoomed in)
Figure A4-13 Manchester - Acquisition with applied jitter

Figure A4-14 Manchester - Flywheel with applied jitter (zoomed in)
Figure A4-15 Manchester - Jitter rejection.

Figure A4-15 shows the PDF of jitter in the recovered clock, for an increasing filter-Q. The closer the poles are located to the unit circle in the z-plane (i.e. the closer coefficient K is to unity), then the narrower the filter bandwidth. Hence, the better the noise immunity, as indicated in the above analysis.
Miller encoded data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Rate</td>
<td>1.0</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>0.5</td>
</tr>
<tr>
<td>Coefficient C</td>
<td>0.5</td>
</tr>
<tr>
<td>Coefficient N</td>
<td>0.383</td>
</tr>
<tr>
<td>Fade Out (symbols)</td>
<td>240</td>
</tr>
<tr>
<td>Filter Scale</td>
<td>43.8</td>
</tr>
</tbody>
</table>

A standard SPW library Miller encoder block is used. This is a half-rate code and hence, a 2-stage timing conditioner is used, to extract a timing-component at the symbol rate. The fade control determines the on-off time of the data dropout, in order that the flywheel performance can be measured. This is controlled in the parameter box and is currently set to $240 - 160 = 80$ symbols. The weightings of the jitter generator are also controlled in the parameter box, which are set for a gaussian distribution. Another parameter allows the jitter generator to be switched-off, to allow noise free measurements.

Figure A4-16 Miller encoded - Clock recovery model
Figure A4-17 Miller - Acquisition and flywheel (wide zoom)

Figure A4-18 Miller - Acquisition (zoomed in)
Figure A4-19 Miller - Flywheel (wide zoom)

Figure A4-20 Miller - Flywheel (zoomed in)
APPENDIX 4

FIXED FREQUENCY CLOCK RECOVERY

Figure A4-21 Miller - Acquisition with applied jitter

Figure A4-22 Miller - Flywheel with applied jitter (zoomed in)
Miller$^2$ encoded data

A special Miller$^2$ encoder block has been developed, as described in Appendix 3. This is a half-rate code and hence, a 2-stage timing conditioner is used, to extract a timing-component at the symbol rate.

All useful parameters are included in the parameter box, which can be easily viewed and adjusted. All other parameters are calculated from these user-defined values, which then propagate down to lower level hierarchy blocks.
Figure A4-24 Miller\textsuperscript{2} - Acquisition and flywheel (wide zoom)

Figure A4-25 Miller\textsuperscript{2} - Acquisition (zoomed in)
Figure A4-26 Miller\textsuperscript{2} - Flywheel (wide zoom)

Figure A4-27 Miller\textsuperscript{2} - Flywheel (zoomed in)
**APPENDIX 4**

**FIXED FREQUENCY CLOCK RECOVERY**

**Figure A4-28 Miller\(^2\) - Acquisition with applied jitter**

**Figure A4-29 Miller\(^2\) - Flywheel with applied jitter (zoomed in)**
A special RLL 2,7 encoder block has been developed, as described in Appendix 3. This is a half-rate code and hence, a 2-stage timing conditioner is used, to extract a timing-component at the symbol rate.

All useful parameters are included in the parameter box, which can be easily viewed and adjusted. All other parameters are calculated from these user-defined values, which then propagate down to lower level hierarchy blocks.
APPENDIX 4

FIXED FREQUENCY CLOCK RECOVERY

Figure A4-31 RLL 2,7 - Acquisition and flywheel (wide zoom)

Figure A4-32 RLL 2,7 - Acquisition (zoomed in)
Figure A4-33 RLL 2,7 - Flywheel (wide zoom)

Figure A4-34 RLL 2,7 - Flywheel (zoomed in)
Figure A4-35 RLL 2,7 - Acquisition with applied jitter

Figure A4-36 RLL 2,7 - Flywheel with applied jitter (zoomed in)
NRZ Scrambled data

NRZ data has been applied to a two-stage filter system, to increase the overall flywheel performance. The data rate is the same period as the symbol rate, hence a single T/2 signal conditioner is applied.

Simulation results show that clock acquisition is still achieved at the first retrieved data transition, while flywheel performance is approximately doubled.
Figure A4-38 NRZ Scrambled - Acquisition over 2-stages
Figure A4-39 NRZ Scrambled - Flywheel over 2-stages
Appendix 5

Fixed frequency clock recovery - Schematic diagrams

This appendix contains schematic diagrams, referenced in Chapter 2.

Figures A5-1 to A5-3 relate to the test apparatus described in Section 2.4. Figures A5-4 and A5-5 contain diagrams for the IIR filter, implemented in an FPGA.

Figure A5-1 Test generator board

Figure A5-1 shows the test generator board containing a baud-rate generator, a PRBS generator and a synthesiser divider with external fine resolution, for adjusting the symbol rate frequency. Provision is made for generating phase-reversal symbols and switching the data to all-ones, to allow measurement of the IIR flywheel performance. The test-rack inter-connections to the DSP clock recovery board are also shown.
Figure A5-2 DSP Clock recovery board
Figure A5-2 shows the DSP clock recovery board, as described in Section 2.4. The 40MHz-processor clock is divided, to produce a choice of three stable sample-rate clocks. The required sample-rate clock is selected under software control and is converted to an interrupt pulse. A wait-state generator allows access to slow devices, such as the EPROMs, UART and GPIB interface. A watchdog timer is provided to reset the system in the event of a program crash. IO mapped test outputs are also provided, for connection to the Test DAC board, shown in Figure A5-3.
Figure A5-4 IIR Filter - FPGA design

Figure A5-4 shows the top-level schematic diagram for the clock recovery filter, implemented within an FPGA. While Figure A5-5, shows an example of a sub-hierarchy block, in this case, a 16-Bit ripple subtractor. The 16-Bit adder (no inverters) and 17-Bit subtractor sub-blocks are constructed in a similar manor.
Fixed frequency clock recovery - Real-time DSP

This appendix contains software information and experimental results for the real-time DSP implementation, of the fixed frequency clock recovery system.

DSP 320C25 Clock recovery software description

The software has been developed into three separate assembly language modules:

- CRSYNC Module – Clock Recovery Synchroniser algorithm.
- CRUART Module – Remote control routines via the serial interface.
- CRGPIB Module – Remote control routines via the GPIB (IEEE488) interface.

Each of which, are compiled and linked to form an executable programme in Intel-Hex format, for programming the onboard EPROMs.

Figure A6-1 Flow diagram - Software overview
APPENDIX 6  FIXED FREQUENCY CLOCK RECOVERY – REAL-TIME DSP

The flow diagram shown in Figure A6-1 provides an overview of the software execution. The CRSYNC (Clock Recovery Synchroniser) module contains the main programme control and includes the DSP initialisation and service routines. All DSP variables are stored in fast data-memory, and are pre-loaded, with the appropriate symbol-rate dependent parameters.

At switch-on, or upon application of a reset, the address is forced to the zero location in EPROM, where the reset vector is located. This forces a branch to the start of the initialisation routines. The initialisation routines sets-up the UART interface (9600 baud, no parity, 7 bits, 2 stop bits) and downloads a screen message to the terminal. The initialisation routine also reads the GPIB address from a DIL switch and configures the GPIB (IEEE488) instrumentation interface.

The power-on default status is read from a DIL switch, trimmed to 4-bits and loaded into data RAM. A series of test are then performed, to determine the default symbol rate for which clock recovery is to be applied. This causes a branch to load into data-memory the symbol rate dependant parameters, which are written as a macro. The macro variables are as follows: Centre Frequency, Number of samples/symbol, Filter-1 $K$-coefficient, Filters-1 and 2 $C$-coefficient, Filter-2 $K$-coefficient, System input scaling factor, Filter-1 input scaling factor and pre-delay length. All other symbol rate dependant parameters are calculated from this information. To complete the initialisation, the time-critical DSP routines are downloaded to the processor on-chip RAM, which is reconfigured to program memory and runs with zero wait-states. The interrupts are enabled and the processor idles in fast memory until an interrupt occurs.
Interrupt 0 Service

Upon receipt of a new sample, an interrupt is generated which causes the clock recovery algorithm to be implemented. The clock recovery algorithms are subdivided and assigned “CR” (Clock Recovery) reference numbers, which can be cross-referenced to the assembler source code.

**Figure A6-2 DSP - Software algorithm overview**

**CR20 Input Limiter and Scaler:**

The input data is read from the input latch and applied to the MSB (the sign bit) of the 16-bit data-bus, which hard limits the input to a constant value. A scaling-factor is applied, which takes into consideration the accumulative effects of algorithms CR30 and CR40, for an ideal, phase-reversal signal input.

**CR30 Delay and Multiply**

The scaled data are applied to a delay-line, which is equal to \( \frac{1}{2} \) the symbol period. This delay-line can be quite large, and is therefore implemented using indirect addressing in external RAM. The input and output of this delay are multiplied, to produce the symbol-timing frequency component.
**CR40 Filter-1**

The symbol-timing component is fed into the first of two IIR filters. The scaling at the input to CR20 has been chosen to obtain near maximum amplitude at the filter output.

**CR50 Scaler and Limiter**

The Filter 1 output is hard-limited by detecting the sign of the data. Any values that are positive are set to a positive scaled value, and similarly any negative values are set to a negative scaled value. The scaling-factor is chosen to produce near maximum amplitude at the output of Filter-2, for a phase-reversal data input to the system. In the time domain, this hard-limited output produces quite an acceptable clock. However, if the amplitude at the output of Filter-1 falls to zero, then clock synchronisation will be lost.

**CR60 Filter-2**

The Limiter/Scaler output is passed through a second IIR filter, with identical characteristics to the first. Two filters cascaded in this way, produce a high degree of noise rejection and also extend the flywheel performance at the system output. This is necessary to maintain clock recovery, during a data dropout.

**CR70 Pre-delay**

Since IIR filters are non-linear, any frequency deviation of the input data will cause a corresponding phase-shift in the recovered clock. The output of Filter-2 is delayed, such that the rising edge of the output clock re-samples the data at the centre of the symbol period. This optimises use of the jitter detection window and allows the maximum deviation of the input frequency, in both positive and negative
directions, before re-timing errors occur. This delay can be quite large and is therefore implemented using indirect addressing in external RAM.

**CR80 Quadrature Delay**

The output from CR70 is delayed by \( \frac{1}{4} \) of a symbol period to produce a second output, displaced from the first by \( \frac{1}{4} \) of a symbol period. This delay is implemented using indirect addressing in external RAM.

**CR90 Outputs**

The two derived signals from CR70 and CR80 have their MSB’s (sign bits) latched, which effectively provides hard limiting. The latches produce true and inverse outputs, providing quadrature recovered clock outputs.

**End of Interrupt**

The two filter outputs are written to the optional test DAC’s, and the signals may be viewed on an oscilloscope. All interrupts are re-enabled and the programme returns to the idle loop to await another interrupt.

**Interrupt 2 Service**

When the UART receives a character from the dumb-terminal, an interrupt is generated, which causes a branch to the UART service routines. In order that the status of the interface can be monitored, a “Rx flag” is set when receiving data, and a “Tx flag” is set when transmitting data. When an interrupt-2 signal is received, the processor status is saved to a software stack in page-0. The received ASCII character is placed into a receive-buffer, the location of which is referenced by an address vector. If the received character is valid, it is echoed-back to the terminal.
via the UART. If the character is an ASCII space, then it is removed from the receive-buffer, since it is not needed for subsequent processing. Otherwise, the buffer address vector is incremented in preparation for the next character to be received. If the received character is an ASCII carriage return, then the full message has been received and the message is processed. Otherwise, the interrupt will end and the processor status will be restored from the software stack.

The first character in the receive buffer is tested to determine the command:

A = Run from a specified program memory location.
B = Read the contents of a specified data memory location.
C = Change the contents of a specified data memory location.
D = Read the contents of a specified program memory location.
E = Change the contents of a specified program memory location.
F = Display the current clock recovery symbol rate.
G = Change the clock recovery symbol rate, displays an option menu 0 to 8.
H = Displays a Help Menu.

0 = Set clock recovery to 50 symbols/sec.
1 = Set clock recovery to 73 symbols/sec.
2 = Set clock recovery to 100 symbols/sec.
3 = Set clock recovery to 200 symbols/sec.
4 = Set clock recovery to 300 symbols/sec.
5 = Set clock recovery to 600 symbols/sec.
6 = Set clock recovery to 1200 symbols/sec.
7 = Set clock recovery to 2400 symbols/sec.
8 = Set clock recovery to 4800 symbols/sec.
9 = Error.

Option A. The specified Hex address is converted from ASCII format to true 16-bit format and an unconditional branch is executed to this address.

Option C and E. The specified Hex address is converted from ASCII format to true 16-bit format and the corresponding address located. The retrieved data is converted from 16-bit format to its equivalent ASCII Hex value and transmitted to the terminal, via the UART.

Option B and D. The specified Hex address and data is converted from ASCII format to true 16-bit format. The converted data is then stored in the address location.

Option F. The value of the data memory location containing the current symbol rate setting is read, converted into ASCII and transmitted with a message to the terminal, via the UART.

Option G. A “Change Symbol Rate” menu (displaying options 1 to 8), stored as a data array is transmitted to the terminal, via the UART.

Option H. The system monitor help screen, stored as a data array is transmitted to the terminal, via the UART.

Option 0 to 8. The ASCII number is converted to binary and stored in the data memory location, which contains the current symbol-rate setting. The appropriate
macro is then called to re-load the symbol-rate dependant parameters to this new setting. The new value of the data memory location containing the current symbol-rate setting is read, converted into ASCII and transmitted with a message to the terminal, via the UART.

**Option 9.** An error message is transmitted to the terminal.

On completion of servicing one of the above commands, the interrupt is ended and the processor status is restored from the software stack. All interrupts are re-enabled and the processor idles in fast memory, waiting for a new interrupt to occur.

**Interrupt 1 Service**

When the TMS9914A GPIB interface-chip, recognises its address and receives a character from the instrumentation bus, it generates an interrupt, which causes the GPIB routines to be implemented.

Transmit and Receive flags are set and the processor status is stored to the software stack, as for interrupt 2. The received ASCII character is placed in the same receive-buffer as used by the UART. A branch is then made to the UART routines and the command is processed, just as if it had been received from the terminal.
Experimental results

Table A6-1 summarises the measured characteristics for a single IIR filter section, where Fs is the sample rate and Ns is the number of samples/symbol period.

<table>
<thead>
<tr>
<th>Symbol Rate</th>
<th>Fs (Hz)</th>
<th>Ns</th>
<th>BW (Hz)</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>6K4</td>
<td>128</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>73</td>
<td>6K4</td>
<td>96</td>
<td>2</td>
<td>36</td>
</tr>
<tr>
<td>100</td>
<td>6K4</td>
<td>64</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td>200</td>
<td>6K4</td>
<td>32</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>300</td>
<td>38K4</td>
<td>128</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>600</td>
<td>76K8</td>
<td>128</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>1200</td>
<td>76K8</td>
<td>64</td>
<td>52</td>
<td>23</td>
</tr>
<tr>
<td>2400</td>
<td>76K8</td>
<td>32</td>
<td>105</td>
<td>22</td>
</tr>
<tr>
<td>4800</td>
<td>76K8</td>
<td>16</td>
<td>107</td>
<td>44</td>
</tr>
</tbody>
</table>

Table A6-1 Filter characteristics

Table A6-2 shows the measured optimum centre frequency, for a single IIR filter. The coefficient data values are then changed by ±1 bit and the centre frequency re-measured, to determine the effects of coefficient quantization.

<table>
<thead>
<tr>
<th>Symbol Rate</th>
<th>Optimum</th>
<th>-1 Bit</th>
<th>+1 Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50.3560</td>
<td>52.7960</td>
<td>47.7316</td>
</tr>
<tr>
<td>73</td>
<td>73.0000</td>
<td>74.8310</td>
<td>71.1690</td>
</tr>
<tr>
<td>100</td>
<td>99.4507</td>
<td>100.9155</td>
<td>98.1079</td>
</tr>
<tr>
<td>200</td>
<td>200.6204</td>
<td>201.1596</td>
<td>199.8160</td>
</tr>
<tr>
<td>300</td>
<td>302.3807</td>
<td>316.6625</td>
<td>286.9990</td>
</tr>
<tr>
<td>600</td>
<td>603.8452</td>
<td>634.3627</td>
<td>572.1679</td>
</tr>
<tr>
<td>1200</td>
<td>1194.3200</td>
<td>1208.8500</td>
<td>1178.7590</td>
</tr>
<tr>
<td>2400</td>
<td>2399.3859</td>
<td>2410.3700</td>
<td>2391.6990</td>
</tr>
<tr>
<td>4800</td>
<td>4799.8779</td>
<td>4802.3300</td>
<td>479.1923</td>
</tr>
</tbody>
</table>

Table A6-2 Filter coefficient quantization
Measurements with a logic analyser confirm the simulation results. A clock is recovered, within a symbol period, from application of the first input symbol transition.

Table A6-3 shows the number of measured symbol clocks, for which the system flywheels, when the input data are removed.

<table>
<thead>
<tr>
<th>Symbol Rate</th>
<th>Clock Flywheel for Symbol Reversals</th>
<th>Clock Flywheel for PRBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>121</td>
<td>96</td>
</tr>
<tr>
<td>73</td>
<td>84</td>
<td>80</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
<td>110</td>
</tr>
<tr>
<td>200</td>
<td>126</td>
<td>113</td>
</tr>
<tr>
<td>300</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>600</td>
<td>52</td>
<td>48</td>
</tr>
<tr>
<td>1200</td>
<td>51</td>
<td>48</td>
</tr>
<tr>
<td>2400</td>
<td>50</td>
<td>46</td>
</tr>
<tr>
<td>4800</td>
<td>64</td>
<td>60</td>
</tr>
</tbody>
</table>

**Table A6-3 Flywheel performance**

The PRBS generator contains a 16-bit shift register and therefore, the maximum run length is 15 symbols. The flywheel performance recorded for PRBS data is approximate, since the amplitude of the first filter varies depending upon the most recent run of symbols, this in turn effects the filter decay period.

Nevertheless, results show that the clock recovery system can withstand at least a 40-symbol dropout, without losing clock synchronisation. This could be further improved by increasing the filter-Q.
Bit error rate measurements have been performed using an HP 3764A Digital Transmission Analyser. These results show that zero errors occur, until the frequency of the input symbols deviated beyond the bandwidth of the filter. If the frequency deviation continues, then the change in phase of the recovered clock results in re-sampling the symbols at the edge of the data-eye. At this point, miss-timing, produces a rapid number of errors, as shown in the example of Figure A6-3. For the above example, this represents approximately a 4% change in the frequency of the applied symbol rate. The percentage change of the input symbol rate frequency is inversely proportional to the filter-Q. Therefore, if the filter-Q is increased (presently 20) to improve further the flywheel performance, then the percentage change before re-timing errors will reduce proportionally.
Source code listing

For brevity the Linker Command file, UART and GPIB assembly routines are omitted. These modules are available separately on disk. CRSYNC is listed below.

Clock recovery module assembler source code

Module 1 of 3 Main program, initialisation and clock recovery algorithms.

IDT 'CRSYNC'

**********************
* Clock Recovery *
* Main program, linked to CRUART and CRGPIB *
**********************

****************** DEFINE LINKER CROSS REFERENCE **********************
REF INITUT Uart initialization
REF INITGP GPIB initialization
REF SRINT1 Interrupt 1, Service GPIB
REF SRINT2 Interrupt 2, Service UART
DEF SYMBOL Loads symbol rate dependant parameters

****************** DEFINE DATA CONSTANTS ***********************
I MASK EQU >FFC7 Enable INTO, INT1 and INT2
FS6K EQU >0 Sample rate 6.4KHz
FS38K EQU >1 Sample rate 38.4KHz
FS76K EQU >2 Sample rate 76.8KHz
FS153K EQU >3 Sample rate 153.6KHz

************* ALGORITHM CR30 *************
SL16 EQU >10 Symbol length 16 samples/symbol
SL32 EQU >20 Symbol length 32 samples/symbol
SL64 EQU >40 Symbol length 64 samples/symbol
SL96 EQU >60 Symbol length 96 samples/symbol
SL128 EQU >80 Symbol length 128 samples/symbol
SL256 EQU >100 Symbol length 256 samples/symbol
DYSP1 EQU >0FF End of external ram

************* ALGORITHM CR40, CR60 *************
DELAY AND MULTIPLY
CLKDLY EQU >2 Delay length

************* ALGORITHM CR80 *************
QUADRATURE CLOCK DELAY
DYSP2 EQU DYSP1-128 In external ram

************* DEFINE PAGE LABELS *************
PAGE0 EQU >0 Page 0 of data mem for mem-mapped regs
B0 EQU >4 Page 4 start of internal RAM/ROM block
AIMS EQU >6 Page 6 contains constants, variables
APPENDIX 6  FIXED FREQUENCY CLOCK RECOVERY - REAL-TIME DSP

*************** DEFINE INPUT OUTPUT PORTS ***************

*** INPUTS

INPUT1 EQU >0             Port 0 is the data input
GPIADD EQU >C             Port 12 is the RD GPIB address switch
LSTAT  EQU >D             Port 13 is the local status input

*** OUTPUTS

OUT1  EQU 1               Port 1 is the output for CKs 1 and 3
OUT2  EQU 2               Port 2 is the output for CKs 2 and 4
OUT3  EQU 3               Port 3 is the data output
LOCKED EQU 7              Port 7 is the LED drive
SAMCK EQU 4               Port 4 sets the sample rate clock

*************** DEFINE ADDRESS LABELS ***************

*************** PAGE 0 ***************

TIM    EQU >2             Address of current time
PRD    EQU >3             Address of period register
IMR    EQU >4             Address of mask register
GREG   EQU >5             Address of global mem register

*************** PAGE AIMS ***************

*************** GLOBAL VARIABLES ***************

ATEMP  EQU >0              Temporary memory location
VALIN  EQU ATEMP+1         Will hold input value
SYMBOL EQU VALIN+1         Will hold operational symbol rate
SMPLRT EQU SYMBOL+1        Will hold sample rate setting

*************** PROCESS LOCATIONS ***************

*** ALGORITHM CR20

LPOS1 EQU SMPLRT+1         LIMITER/SCALER 1
Will hold positive limit value

*** ALGORITHM CR30

SYMLEN  EQU LPOS1+1        DELAY AND MULTIPLY
Will hold No. of samples/symbol
HLFSYM  EQU SYMLEN+1       Will hold 1/2 No. of samples/symbol
VDYST1  EQU HLFSYM+1       Vector for delay start location
VDYSPI  EQU VDYST1+1       Vector for delay end location
VINPT1  EQU VDYSPI+1       Vector for input location
DYIN1   EQU VINPT1+1       Will hold input value

*** ALGORITHM CR40

VF1DEN  EQU DYIN1+1        FILTER 1
F1COFK  EQU VF1DEN+1       Vector for delay end location
F1COFC  EQU F1COFK+1       Will hold Coef K
FLIN   EQU F1COFC+1        Will hold Coef C
F1DST  EQU FLIN+1          Filter input
F1ZM1   EQU F1DST+1        Vo Z-0 (delay start) also filter O/P
F1ZM2   EQU F1ZM1+1        Vo Z-1
F1DEND  EQU F1ZM2+CLKDLY    Vo Z-2
Delay output = Z-2

*** ALGORITHM CR50

LPOS2  EQU F1DEND+1        LIMITER/SCALER 2
Will hold positive limit

*** ALGORITHM CR60

VF2DEN  EQU LPOS2+1        FILTER 2
F2COFK  EQU VF2DEN+1       Vector for delay end location
F2COFC  EQU F2COFK+1       Will hold Coef K
F2IN    EQU F2COFC+1       Will hold Coef C
F2DST   EQU F2IN+1         Filter input
F2ZM1   EQU F2DST+1        Vo Z-0 (delay start) also filter O/P
F2ZM2   EQU F2ZM1+1        Vo Z-1
F2DEND  EQU F2ZM2+CLKDLY    Vo Z-2
APPENDIX 6  FIXED FREQUENCY CLOCK RECOVERY – REAL-TIME DSP

F2DEND EQU F2DST+CLKDLY  

*** ALGORITHM CR70  
VDYST2 EQU F2DEND+1  
VDYS3 EQU VDYST2+1  
VINPT2 EQU VDYS3+1  
DYOUT2 EQU VINPT2+1  
PREDLY EQU DYOUT2+1

*** ALGORITHM CR80  
QTRSYM EQU PREDLY+1  
VDYST3 EQU QTRSYM+1  
VDYS3 EQU VDYST3+1  
VINPT3 EQU VDYS3+1  
DYIN3 EQU VINPT3+1  
DYOUT3 EQU DYIN3+1

---------- LOCATIONS FOR INPUTS AND OUTPUTS OF ALGORITHMS ----------

* Locations >30 to >33 used in linked CRUART to store GPIB STATUS

************** LOCATIONS FOR INPUTS AND OUTPUTS OF ALGORITHMS **************

**'CR*** ' = Clock Recovery Algorithm Number
  * 'A'= Channel 1
  * 'B'= Channel 2
  * 'I'= Input
  * 'O'= Output

*************** ALGORITHM CR10  
CR10O EQU VALIN

*************** ALGORITHM CR20  
CR20I EQU VALIN  
CR200 EQU DYIN1

*************** ALGORITHM CR30  
CR30I EQU DYIN1  
CR300 EQU F1IN

*************** ALGORITHM CR40  
CR40I EQU F1IN  
CR400 EQU F1DST

*************** ALGORITHM CR50  
CR50I EQU F1DST  
CR500 EQU F2IN

*************** ALGORITHM CR60  
CR60I EQU F2IN  
CR600 EQU F2DST

*************** ALGORITHM CR70  
CR70I EQU F2DST  
CR700 EQU DYOUT2

*************** ALGORITHM CR80  
CR80I EQU DYOUT2  
CR800 EQU DYCUT3

*************** SET RESET VECTOR **********

A6-14
**APPENDIX 6**  
**FIXED FREQUENCY CLOCK RECOVERY – REAL-TIME DSP**

* &gt;04  
  B SRINT1  
  Address of the INT1 vector in page0  
  On interrupt branch to GPIB ISR

* &gt;05  
  B SRINT2  
  Address of the INT2 vector in page0  
  On interrupt branch to UART ISR

*************** DEFINE MACROS ***********************

*** MACRO TO DEFINE SYMBOL RATE DEPENDANT PARAMETERS TO AIMS PAGE ***
* A=Sample rate,  B=Samples/symbol,  C=Coef K F1,  D=Coef CK F1,  
* E=Coef K F2,  F=Coef CK F2,  G=Scaler 1,  H=Scaler 2
* J=Pre delay

BR $MACRO A,B,C,D,E,F,G,H,J

* CLEAR FILTER DELAYS
  ZAC
  SACL F1DST
  SACL F1ZM1
  SACL F1ZM2
  SACL F2DST
  SACL F2ZM1
  SACL F2ZM2

* SET UP SAMPLE RATE
  LALK : A.S:
  SACL SMPLRT
  OUT SMPLRT,SAMCK
  A=Selector setting  
  Store the sample rate setting  
  Set sample rate selector

* SET UP CR30
  LALK : B.S:
  SACL SYMLEN
  SFR
  SACL HLFSYM
  LALK DYSPl-1
  SUB HLFSYM
  SACL VDYST1
  SACL VINPT1
  B=Samples/symbol  
  Store the symbol length  
  Divide by 2 = Half symbol length  
  Store half symbol length  
  Location in external ram  
  Vector point to delay start address  
  Begin input point at delay start

* SET UP CR40
  LALK : C.V:
  SACL F1COFK
  LALK : D.V:
  SACL F1COFC
  FILTER 1
  C=Coef K  
  Coef K address  
  D=Coef C X K  
  Coef C address

* SET UP CR60
  LALK : E.V:
  SACL F2COFK
  LALK : F.V:
  SACL F2COFC
  FILTER 2
  E=Coef K  
  Coef K address  
  F=Coef C X K  
  Coef C address

* SET UP CR20
  LALK : G.V:
  SACL LPOS1
  LIMITER/SCALER 1
  Scale value at input  
  Store value

* SET UP CR50
  LALK : H.V:
  SACL LPOS2
  LIMITER/SCALER 2
  Scale value to F2 input  
  Store value

* SET UP CR80
  LAC HLFSYM
  SFR
  ADLK &gt;1
  SACL QTRSYM
  LAC VDYST1
  SBLK &gt;2
  SACL VDYSP3
  SUB QTRSYM
  SACL VDYST3
  SACL VINPT3
  QUADRATURE CLOCK DELAY (delay 3)  
  Half samples/symbol  
  Divide by 2 = 1/4 samples/symbol  
  Store 1/4 samples/symbol  
  Start address of delay 1  
  Leave a 2 location gap  
  End address of quadrature delay  
  Start address of quadrature delay  
  Begin with i/p point = delay start

A6-15
**APPENDIX 6  FIXED FREQUENCY CLOCK RECOVERY – REAL-TIME DSP**

* SET UP CR70
  - `LALK :J.V:`
  - `SACL PREDLY`
  - `LAC VDYST3`
  - `SBLK >2`
  - `SACL VDYSP2`
  - `SUB PREDLY`
  - `SACL VDYST2`
  - `SACL VINPT2`

  PRE CLOCK DELAY (delay 2)
  - Delay length
  - Store value
  - Start address of delay 1
  - Leave a 2 location gap
  - End address of pre delay

  Start address of pre delay
  - Begin with i/p point = delay start

**END**

*********** START OF MAIN PROGRAM ***********

START
  - `AORG >20`
  - `CNFPD`
  - `DINT`
  - `LDPK AIMS`
  - `LALK >FFFF`
  - `SACL ATEMP`
  - `OUT ATEMP,PA7`
  - `CALL INITUT`
  - `CALL INITGP`
  - `CALL INIT`
  - `LALK >0000`
  - `SACL ATEMP`
  - `OUT ATEMP,PA7`
  - `CNFP`
  - `SSXM`
  - `LOOP+BROFST`

  B0 is data memory
  - Dis.interrupts during initialization
  - Point to AIMS page
  - Disable auto reset
  - Initialise UART
  - Initialise GPIB
  - Initialise DSP
  - Renable auto reset
  - B0 is now program space
  - 2’s Complement
  - Branch to fast program

*********** THIS IS THE END OF THE MAIN PROGRAM ***********

*********** START OF INITILIZATION SUBROUTINE ***********

*** SET UP MASK REGISTER

INIT
  - `LDPK PAGE0`
  - `LRLK AR1,IMASK`
  - `SAR AR1,IMR`
  - `ZAC`
  - `SACL GREG`

  Page pointer set to 0
  - Load interrpt mask into AR1
  - All memory is local

*** INITIALIZE MODES

  - `SPM 0`
  - `SOVM`
  - `RSXM`
  - `LARP AR0`

  Set P register shift to 0
  - Set overflow mode
  - Reset sign extension mode
  - Start with ARP=AR0

*** SET UP CONSTANTS AND VARIABLES FOR AIMS PAGE

LDPK AIMS

*SET UP CR30
  - `LALK DYSPl`
  - `SACL VDYSPl`

  DELAY AND MULTIPLY
  - External ram
  - Vector points to delay end address

*SET UP CR40
  - `LALK 128*AIMS+F1DEN-1`
  - `SACL VF1DEN`

  FILTER 1
  - Absolute address
  - Vector points to delay end -1

*SET UP CR60
  - `LALK 128*AIMS+F2DEN-1`

  FILTER 2
  - Absolute address

A6-16
APPENDIX 6  FIXED FREQUENCY CLOCK RECOVERY – REAL-TIME DSP

SAACL VF2DEN
Vector points to delay end -1

*READ LOCAL STATUS
IN ATMP, LSTAT
LAC ATMP
ANDK >F
SAC SYMBOL
From DIL switch
Read local status
Accumulator = local status
Set Hi 14Bits = 0
Store symbol rate

*SET UP SYMBOL RATE DEPENDANT PARAMETERS
CALL SYMBOLRT
Determine Symbol rate

*** END OF SET UPS FOR AIMS PAGE

*** TRANSFER PROGRAM TO INTERNAL ROM
LARP AR0
LRK AR0, 128*B0
RPTK PROGL-1
BLKP FSTPRG, ++
Copy fast program section into B0

*** INITIALIZATION COMPLETE
RET
Return to main program

*************** START OF SUBROUTINE TO DETERMINE SYMBOL RATE

SYMBOLRT LARP AR1
LAR AR1, SYMBOL
ARP = AR1
AR1 = Symbol rate setting

* TEST FOR 50 SYMBOL
LARK AR0, >00
CMPR 0
BRNZ BR50
AR0 = 0
If AR1 = AR0 branch
Initialize for 50 symbols

* TEST FOR 73 SYMBOL
LARK AR0, >01
CMPR 0
BR73
AR0 = 1
If AR1 = AR0 branch
Initialize for 73 symbols

* TEST FOR 100 SYMBOL
LARK AR0, >02
CMPR 0
BR100
AR0 = 2
If AR1 = AR0 branch
Initialize for 100 symbols

* TEST FOR 200 SYMBOL
LARK AR0, >03
CMPR 0
BR200
AR0 = 3
If AR1 = AR0 branch
Initialize for 200 symbols

* TEST FOR 300 SYMBOL
LARK AR0, >04
CMPR 0
BR300
AR0 = 4
If AR1 = AR0 branch
Initialize for 300 symbols

* TEST FOR 600 SYMBOL
LARK AR0, >05
CMPR 0
BR600
AR0 = 5
If AR1 = AR0 branch
Initialize for 600 symbols

A6-17
* TEST FOR 1200 SYMBOL
LARK AR0, >06
CMPR 0
BENZ BH1200

* TEST FOR 2400 SYMBOL
LARK AR0, >07
CMPR 0
BENZ BR2400

* TEST FOR 4800 SYMBOL
LARK AR0, >08
CMPR 0
BENZ BR4800

* INVALID STATUS
B SYMBOLRT

*** CALL MACRO TO STORE SYMBOL RATE DEPENDANT PARAMETERS TO AIMS PAGE

*** SYMBOL RATE = 50
* fs=6.4KHz Ns=128 K1=1023/1024 C=1.997590 K2=1023/1024 SCL1 SCL2 PREDLY
BR50 BR FS6K,SL128,>7FE0,>1FF2,>7FE0,>1FF2,>CO,>09,>1D

*** SYMBOL RATE = 73
* fs=6.4KHz Ns=96 K1=511/512 C=1.994866 K2=511/512 SCL1 SCL2 PREDLY
BR73 BR FS6K,SL96,>7FC0,>1FE3,>7FC0,>1FE3,>149,>1A,>58

*** SYMBOL RATE = 100
* fs=6.4KHz Ns=64 K1=255/256 C=1.961570 K2=255/256 SCL1 SCL2 PREDLY
BR100 BR FS6K,SL64,>7FC0,>1FD1,>7FC0,>1FD1,>180,>24,>39

*** SYMBOL RATE = 200
* fs=6.4KHz Ns=32 K1=127/128 C=1.990369 K2=127/128 SCL1 SCL2 PREDLY
BR200 BR FS6K,SL32,>7F80,>1F52,>7F80,>1F52,>300,>90,>05

*** SYMBOL RATE = 300
* fs=38.4KHz Ns=128 K1=511/512 C=1.997590 K2=511/512 SCL1 SCL2 PREDLY
BR300 BR FS38K,SL128,>7FC0,>1FEE,>7FC0,>1FEE,>10F,>11,>12

*** SYMBOL RATE = 400
* fs=76.8KHz Ns=128 K1=255/256 C=1.990369 K2=255/256 SCL1 SCL2 PREDLY
BR400 BR FS76K,SL256,>7F00,>1F43,>7F00,>1F43,>44A,>11A,>02

*** SYMBOL RATE = 600
* fs=76.8KHz Ns=64 K1=127/128 C=1.961570 K2=127/128 SCL1 SCL2 PREDLY
BR600 BR FS76K,SL128,>7F00,>1F43,>7F00,>1F43,>44A,>11A,>02

*** SYMBOL RATE = 800
* fs=76.8KHz Ns=32 K1=127/128 C=1.847759 K2=127/128 SCL1 SCL2 PREDLY
BR800 BR FS76K,SL128,>7F00,>1D72,>7F00,>1D72,>600,>230,>02

********** THIS IS THE START OF THE TIME CRITICAL PROGRAM SECTION **********

* This time critical program section is copied into and run from on-chip RAM.
* A branch offset to run the program from on-chip memory, is calculated at
* the end of this program.
APPENDIX 6
FIXED FREQUENCY CLOCK RECOVERY – REAL-TIME DSP

FSTPRG EQU $                 Fast program section starts here
LOOP EINT
        IDLE
        B LOOP+BROFST

*** This is the start of the Digital Signal Processing ISR.

************** ALGORITHM CR20
FASTST SXF
    IN VALIN,INPUT1
    BIT VALIN,>0
    LAC LPOS1
    BBZ SAVE1+BROFST
    NEG
SAVE1 SA CL D YI N1

************** ALGORITHM CR30
LARP AR4
    LAR AR0,VDYSP1
    LAR AR4,VINPT1
    SA CL *+
    CMPR 0
    BBZ DCONT1+BROFST
DCONT1 SAR AR4,VINPT1
    LT D YIN1
    MPY *,AR1
    PAC
    SACH FliN

************** ALGORITHM CR40
LAR AR1,VF1DEN
    RPTK CLKDLY-1
    DMOV *-
    SPM 1
    LT F1ZM2
    MPY F1COFK
    SAC H AT E M P
    SPM 2
    LT F1ZM1
    MPY F1COFC
    PAC
    SUBH AT E M P
    ADD FliN,12
    SACH F1DST

************** ALGORITHM CR50
BIT F1DST,0
    LAC LPOS2
    BBZ SAVE2+BROFST
    NEG
SAVE2 SACL FZIN

************** ALGORITHM CR60
LAR AR1,VF2DEN

LIM I T E R / S C A L E R 1
Set the external flag
Store the input value
Test MSB, 0=+ve 1=-ve
Load +ve limit value
Positive, so jump past neg.
Negative, so complement -ve limit
Output location of limiter.

DELAY AND MULTIPLY (delay 1)
AR4=Pointer
AR0=Stop address
Point to input address
Store input value then
point to output address
Is pointer = stop address
If no branch
Yes, so reset pointer
Save new input point

LIMITER/SCALER 2
Test MSB, 0=+ve 1=-ve
Load +ve limit value
Positive so jump past neg
Negative, so complement -ve limit
Input to filter 2

FILTER 1
AR1 = Z-1
Shift register
X 2 on P -> A
T reg = VoZ-2
X by coef K
P -> A (X 2)
Store K(VoZ-2)

FILTER 2
AR1 = Z-1
Add input shifted into Hi bytes
Result into Z-0
Shift register

**Algorithm CR70**

```
LARP AR4
LAR AR0, VDYSPT2
LAR AR4, VINPT2
SACH *
CMPR 0
BEZ DCONT2 + BROFST
LAR AR4, VDYSP2
SACH *
```

**Pre Clock Delay** (delay 2)

```
AR4 = Pointer
AR0 = Stop address
Point to input address
Store i/p value then point to o/p
Is pointer = stop address
If no branch
Yes, so reset pointer
Save new i/p point
With pre delay o/p value
Store o/p value
```

**Algorithm CR80**

```
LAR AR0, VDYSPT3
LAR AR4, VINPT3
SACL *
CMPR 0
BEZ DCONT3 + BROFST
LAR AR4, VDYSP3
SACL *
```

**Quadrature Clock Delay** (delay 3)

```
AR0 = Stop address
Point to input address
Store i/p value then point to o/p
Is pointer = stop address
If no branch
Yes, so reset pointer
Save new i/p point
With pre delay o/p value
Store o/p value
```

```
OUT DYOUT2, 1
OUT DYOUT3, 2
OUT VALIN, 3
OUT CR400, 5
OUT CR600, 6
RXF
RET
```

```
PROGE EQU $ Fast program section ends here
```

```
**Calculate B0 address labels**
```

```
PROGL EQU PROGE-FSTPRG Fast program length
BROFST EQU > FPFO - FSTPRG Branch offset
INT0 EQU FASTST + BROFST Start of ISR in on-chip ROM
END
```

A6-20
A7 – Appendix 7

Adaptive tracking clock recovery

This appendix supplements Chapter 3, and provides full simulation results to evaluate performance of the adaptive (tracking) clock recovery algorithm.

Results are presented for a sinusoidal input and each of the run length limited test codes (Manchester, RLL3, and RLL7).

Simulation model

An overview of the simulation model is shown in Figure A7-1. Several hierarchy custom coded blocks have been created.

![Figure A7-1 Overview of adaptive simulation model](image)

Although not specified by SPW, test simulations have revealed that many of SPW's standard library blocks are quantized to the nearest sample rate integer. For simulation of the adaptive clock recovery system, the symbol rate frequency deviates by up to ±10%, corresponding to 14.545 to 17.7777 samples/symbol.
Therefore, the simulation model must operate at non-integer sample rates. For this reason SPW's standard random generator and encoders cannot be used. Instead, a special "RLL Data Generator" block has been created, which can be adjusted to provide either Manchester encoded data, or PRBS data of the required maximum run length (RLL3 and RLL7 are used). For RLL7 data, the data generator has provision for applying a short phase-reversal pre-amble at the start of a data burst, to aid clock acquisition, as shown in Figure A7-2.

![Figure A7-2 Data generator - RLL7](image)

To evaluate clock flywheel performance, the data generator output is subjected to a dropout, by holding the data for a pre-defined period. For evaluation of acquisition and dropout performance, the data generator is clocked with a fixed frequency signal generator, pre-set to the desired frequency offset. In order to simulate a drifting symbol rate frequency, the data generator is alternatively clocked from a
VCO with a ramp control voltage, such that the VCO output is linearly frequency modulated across the entire tracking range. The frequency modulation is initially inhibited to allow full acquisition of the adaptive clock recovery system, before frequency tracking is applied. The VCO quiescent frequency is set to the nominal symbol rate \( f_c = 0.0625 \text{Hz} \) and the deviation set to the maximum frequency offset \( \Delta = 0.00625 \text{V/Hz} = 10\% \). A triangular VCO control signal of \( \pm 1 \text{V} \), results in the VCO frequency linearly varying by \( \pm 10\% \) \((-1 \text{V} \equiv -10\% = 0.05625 \text{Hz}, 0 \text{V} \equiv 0\% \) to \( 0.0625 \text{Hz}, +1 \text{V} \equiv +10\% = .06875 \text{Hz})\). The amplitude of the triangular VCO control signal is then simply adjusted, to 0.75V for the \( \pm 7.5\% \) tracking range, and 0.5V for the 5% range.

![Diagram](image)

**Figure A7-3 Data generator - Manchester encoded**

For Manchester encoded data, the data generator is clocked at the bit rate (twice the symbol rate) to provide the corresponding symbol rate frequency after \( \frac{1}{2} \) rate.
encoding. Therefore, the VCO quiescent frequency is set to the nominal bit rate \( f_c = 0.03125 \text{Hz} \) and the deviation is set to the maximum frequency offset \( \Delta = 0.003125 \text{V/Hz} = 10\% \). After \( \frac{1}{2} \) rate Manchester encoding, these frequencies are translated to the standard symbol rates, as used with RLL3 and RLL7 data.

Figure A7-4 Data generator - RLL3

Figure A7-5 shows the system simulation model for RLL7 data, for a tracking range of ±5%. Similar models are used for RLL3 and Manchester encoded data, where the data generator and parameters are changed, depending upon the coding scheme and tracking range. Parameters are adjusted to place the detection filters to the desired spacing for a given tracking range, and the adaptive control block is changed to provide the required calibration to convert the error signal to frequency offset. A “delay and multiply” timing conditioner is used to generate a symbol-timing frequency component, which is then applied to the clock recovery system.

A7-4
Figure A7-5 Adaptive tracking system - Simulation model
The symbol-timing frequency component is input to the adaptive clock recovery filter and also to a fixed frequency filter, in order that the benefits of the adaptive system can be compared. The symbol-timing frequency component is provided with gain to compensate for the low-Q of the detection filters, and then low-pass filtered to remove odd harmonics. The pre-filtered signal is applied to the frequency-offset detector, which derives a frequency offset error signal. The error signal is translated to frequency-offset bins, by a custom adaptive control block, which applies the appropriate gain and coefficient to the adaptive filter.

The adaptive control block converts the error control signal into 21-frequency control bins (+10 to −10) by means of window comparators, as shown in the sub-hierarchy detail of Figure A7-6. The thresholds of the window comparators are set to ½ percent frequency offset boundaries, such that the tracking filter switches parameters, at ± ½ bin from the nominal bin centre frequency. For example, the zero bin has a lower threshold set to -0.5 and an upper threshold to +0.5, similarly the +1 bin has a lower threshold set to +0.5 and an upper threshold to +1.5, etc. If the error control signal falls within a given comparator window, the bin number is switched-on, and the corresponding coefficient and gain are applied to the tracking filter.

The calibration of the adaptive control block is related to both the tracking range and the coding scheme, calibration results of which are tabulated in Table A7-5 to Table A7-7.
Figure A7-6 Adaptive control block - Sub hierarchy detail
Calibration results

For each of the test codes, a set of open loop simulations have been performed, to measure the error control signal relative to frequency offset. The graph of Figure A7-7 provides an error control calibration curve for the ±5% tracking range. Both the detection filter spacing and the encoded run length influence the error control signal. Therefore, similar graphs may be plotted for the ±10% and 7.5% tracking ranges, from Table A7-2 and Table A7-3.

![Figure A7-7 Error control signal for ±5% offset](image)

If the tracking range is increased, then the acceptable run length of the encoded data is compromised. A tracking range of ±5% is acceptable for all three-test codes. For a tracking range of ±10% only Manchester encoded data is used, while for a tracking range of ±7.5% then a run length of 3 or less is used. For all three tracking ranges, the detection filter bandwidth is fixed (normalised BW = 0.00125327) and the corresponding coefficients $K_L = K_H = -31/32$. 

A7-8
The detection filter spacing is controlled by coefficient $C$ and is adjusted to provide the required tracking range, as shown in Table A7-1.

<table>
<thead>
<tr>
<th>Range</th>
<th>$C_L$</th>
<th>$C_H$</th>
<th>$K_L$ and $K_H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>±10%</td>
<td>+0.30794000</td>
<td>-0.30794000</td>
<td>-0.96875</td>
</tr>
<tr>
<td>±7.5%</td>
<td>+0.23137260</td>
<td>-0.23137260</td>
<td>-0.96875</td>
</tr>
<tr>
<td>±5%</td>
<td>+0.15444688</td>
<td>-0.15444688</td>
<td>-0.96875</td>
</tr>
</tbody>
</table>

Table A7-1 Detection filter coefficients

Table A7-2 to Table A7-4 tabulates the error control signal for the centre bin positions, for ±10%, ±7.5% and ±5% ranges respectively.

<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Sine</th>
<th>Filtered Pulse</th>
<th>Manch'</th>
</tr>
</thead>
<tbody>
<tr>
<td>+10</td>
<td>0.068750</td>
<td>17617</td>
<td>18585</td>
<td>13638</td>
</tr>
<tr>
<td>+9</td>
<td>0.068125</td>
<td>12189</td>
<td>12822</td>
<td>9188</td>
</tr>
<tr>
<td>+8</td>
<td>0.067500</td>
<td>7352</td>
<td>7747</td>
<td>5241</td>
</tr>
<tr>
<td>+7</td>
<td>0.066875</td>
<td>4818</td>
<td>5103</td>
<td>3232</td>
</tr>
<tr>
<td>+6</td>
<td>0.066250</td>
<td>3370</td>
<td>3571</td>
<td>2059</td>
</tr>
<tr>
<td>+5</td>
<td>0.065625</td>
<td>2416</td>
<td>2609</td>
<td>1425</td>
</tr>
<tr>
<td>+4</td>
<td>0.065000</td>
<td>1742</td>
<td>1884</td>
<td>890</td>
</tr>
<tr>
<td>+3</td>
<td>0.064375</td>
<td>1207</td>
<td>981</td>
<td>466</td>
</tr>
<tr>
<td>+2</td>
<td>0.063750</td>
<td>764</td>
<td>709</td>
<td>395</td>
</tr>
<tr>
<td>+1</td>
<td>0.063125</td>
<td>373</td>
<td>354</td>
<td>111</td>
</tr>
<tr>
<td>0</td>
<td>0.062500</td>
<td>0</td>
<td>-211</td>
<td>-230</td>
</tr>
<tr>
<td>-1</td>
<td>0.061875</td>
<td>-373</td>
<td>-377</td>
<td>-314</td>
</tr>
<tr>
<td>-2</td>
<td>0.061250</td>
<td>-764</td>
<td>-721</td>
<td>-687</td>
</tr>
<tr>
<td>-3</td>
<td>0.060625</td>
<td>-1207</td>
<td>-981</td>
<td>-818</td>
</tr>
<tr>
<td>-4</td>
<td>0.060000</td>
<td>-1742</td>
<td>-1955</td>
<td>-1301</td>
</tr>
<tr>
<td>-5</td>
<td>0.059375</td>
<td>-2416</td>
<td>-2784</td>
<td>-1680</td>
</tr>
<tr>
<td>-6</td>
<td>0.058750</td>
<td>-3370</td>
<td>-3805</td>
<td>-2692</td>
</tr>
<tr>
<td>-7</td>
<td>0.058125</td>
<td>-4818</td>
<td>-5381</td>
<td>-3809</td>
</tr>
<tr>
<td>-8</td>
<td>0.057500</td>
<td>-7352</td>
<td>-8200</td>
<td>-5647</td>
</tr>
<tr>
<td>-9</td>
<td>0.056875</td>
<td>-12189</td>
<td>-13664</td>
<td>-9431</td>
</tr>
<tr>
<td>-10</td>
<td>0.056250</td>
<td>-17617</td>
<td>-19377</td>
<td>-13299</td>
</tr>
</tbody>
</table>

Table A7-2 Error control signal for centre bin positions ±10% range
### Table A7-3 Error control signal for centre bin positions ±7.5% range

<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Sine</th>
<th>Filtered Pulse</th>
<th>Manch’</th>
<th>RLL3</th>
<th>RLL7</th>
</tr>
</thead>
<tbody>
<tr>
<td>+7</td>
<td>0.066875</td>
<td>15252</td>
<td>16216</td>
<td>11605</td>
<td>6064</td>
<td>6974</td>
</tr>
<tr>
<td>+6</td>
<td>0.066250</td>
<td>8921</td>
<td>9502</td>
<td>6698</td>
<td>3616</td>
<td>3710</td>
</tr>
<tr>
<td>+5</td>
<td>0.065625</td>
<td>5410</td>
<td>5804</td>
<td>4173</td>
<td>2351</td>
<td>2009</td>
</tr>
<tr>
<td>+4</td>
<td>0.065000</td>
<td>3506</td>
<td>3813</td>
<td>2557</td>
<td>1504</td>
<td>954</td>
</tr>
<tr>
<td>+3</td>
<td>0.064375</td>
<td>2270</td>
<td>2527</td>
<td>1489</td>
<td>942</td>
<td>369</td>
</tr>
<tr>
<td>+2</td>
<td>0.063750</td>
<td>1377</td>
<td>1261</td>
<td>682</td>
<td>490</td>
<td>152</td>
</tr>
<tr>
<td>+1</td>
<td>0.063125</td>
<td>656</td>
<td>653</td>
<td>330</td>
<td>266</td>
<td>-54</td>
</tr>
<tr>
<td>0</td>
<td>0.062500</td>
<td>0</td>
<td>-370</td>
<td>-220</td>
<td>-93</td>
<td>-350</td>
</tr>
<tr>
<td>-1</td>
<td>0.061875</td>
<td>-656</td>
<td>-661</td>
<td>-486</td>
<td>-364</td>
<td>-361</td>
</tr>
<tr>
<td>-2</td>
<td>0.061250</td>
<td>-1377</td>
<td>-1275</td>
<td>-1011</td>
<td>-671</td>
<td>-677</td>
</tr>
<tr>
<td>-3</td>
<td>0.060625</td>
<td>-2270</td>
<td>-2598</td>
<td>-1622</td>
<td>-1100</td>
<td>-957</td>
</tr>
<tr>
<td>-4</td>
<td>0.060000</td>
<td>-3506</td>
<td>-3977</td>
<td>-2715</td>
<td>-1617</td>
<td>-1667</td>
</tr>
<tr>
<td>-5</td>
<td>0.059375</td>
<td>-5410</td>
<td>-6084</td>
<td>-4432</td>
<td>-1922</td>
<td>-2269</td>
</tr>
<tr>
<td>-6</td>
<td>0.058750</td>
<td>-8921</td>
<td>-9937</td>
<td>-7394</td>
<td>-3016</td>
<td>-4234</td>
</tr>
<tr>
<td>-7</td>
<td>0.058125</td>
<td>-15251</td>
<td>-16962</td>
<td>-12350</td>
<td>-8022</td>
<td>-7690</td>
</tr>
</tbody>
</table>

### Table A7-4 Error control signal for centre bin positions ±5% range

<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Sine</th>
<th>Filtered Pulse</th>
<th>Manch’</th>
<th>RLL3</th>
<th>RLL7</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5</td>
<td>0.065625</td>
<td>16606</td>
<td>17738</td>
<td>13166</td>
<td>9695</td>
<td>7937</td>
</tr>
<tr>
<td>+4</td>
<td>0.065000</td>
<td>11047</td>
<td>11822</td>
<td>8550</td>
<td>6487</td>
<td>5104</td>
</tr>
<tr>
<td>+3</td>
<td>0.064375</td>
<td>6019</td>
<td>6581</td>
<td>4694</td>
<td>3530</td>
<td>2546</td>
</tr>
<tr>
<td>+2</td>
<td>0.063750</td>
<td>3226</td>
<td>3762</td>
<td>2309</td>
<td>1903</td>
<td>1294</td>
</tr>
<tr>
<td>+1</td>
<td>0.063125</td>
<td>1437</td>
<td>1437</td>
<td>1015</td>
<td>781</td>
<td>480</td>
</tr>
<tr>
<td>0</td>
<td>0.062500</td>
<td>0</td>
<td>-800</td>
<td>-331</td>
<td>-237</td>
<td>-250</td>
</tr>
<tr>
<td>-1</td>
<td>0.061875</td>
<td>-1437</td>
<td>-1435</td>
<td>-1247</td>
<td>-803</td>
<td>-621</td>
</tr>
<tr>
<td>-2</td>
<td>0.061250</td>
<td>-3226</td>
<td>-3808</td>
<td>-2685</td>
<td>-2010</td>
<td>-1389</td>
</tr>
<tr>
<td>-3</td>
<td>0.060625</td>
<td>-6019</td>
<td>-6741</td>
<td>-4865</td>
<td>-3657</td>
<td>-2826</td>
</tr>
<tr>
<td>-4</td>
<td>0.060000</td>
<td>-11047</td>
<td>-12171</td>
<td>-9062</td>
<td>-6748</td>
<td>-5513</td>
</tr>
<tr>
<td>-5</td>
<td>0.059375</td>
<td>-16606</td>
<td>-18443</td>
<td>-13662</td>
<td>-10179</td>
<td>-8425</td>
</tr>
</tbody>
</table>
Table A7-5 to Table A7-7 tabulates the error control signal for ½ bin boundaries, for input to the threshold comparators within the adaptive control block.

<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Sine</th>
<th>Filtered Pulse</th>
<th>Manch'</th>
</tr>
</thead>
<tbody>
<tr>
<td>+10</td>
<td>0.0687500</td>
<td>15777</td>
<td>16491</td>
<td>12500</td>
</tr>
<tr>
<td>+9.5</td>
<td>0.0684375</td>
<td>15650</td>
<td>16487</td>
<td>12075</td>
</tr>
<tr>
<td>+8.5</td>
<td>0.0678125</td>
<td>9330</td>
<td>9839</td>
<td>6927</td>
</tr>
<tr>
<td>+7.5</td>
<td>0.0671875</td>
<td>5886</td>
<td>6210</td>
<td>4204</td>
</tr>
<tr>
<td>+6.5</td>
<td>0.0665625</td>
<td>4000</td>
<td>4238</td>
<td>2672</td>
</tr>
<tr>
<td>+5.5</td>
<td>0.0659375</td>
<td>2852</td>
<td>3056</td>
<td>1604</td>
</tr>
<tr>
<td>+4.5</td>
<td>0.0653125</td>
<td>2067</td>
<td>2253</td>
<td>1159</td>
</tr>
<tr>
<td>+3.5</td>
<td>0.0646875</td>
<td>1457</td>
<td>1225</td>
<td>704</td>
</tr>
<tr>
<td>+2.5</td>
<td>0.0640625</td>
<td>965</td>
<td>924</td>
<td>453</td>
</tr>
<tr>
<td>+1.5</td>
<td>0.0643750</td>
<td>557</td>
<td>526</td>
<td>295</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0628125</td>
<td>216</td>
<td>193</td>
<td>74</td>
</tr>
<tr>
<td>-0.5</td>
<td>0.0621875</td>
<td>-216</td>
<td>-217</td>
<td>-152</td>
</tr>
<tr>
<td>-1.5</td>
<td>0.0615625</td>
<td>-557</td>
<td>-562</td>
<td>-580</td>
</tr>
<tr>
<td>-2.5</td>
<td>0.0609375</td>
<td>-965</td>
<td>-899</td>
<td>-629</td>
</tr>
<tr>
<td>-3.5</td>
<td>0.0603125</td>
<td>-1457</td>
<td>-1063</td>
<td>-696</td>
</tr>
<tr>
<td>-4.5</td>
<td>0.0596875</td>
<td>-2067</td>
<td>-2396</td>
<td>-1623</td>
</tr>
<tr>
<td>-5.5</td>
<td>0.0590625</td>
<td>-2852</td>
<td>-3231</td>
<td>-2361</td>
</tr>
<tr>
<td>-6.5</td>
<td>0.0584375</td>
<td>-4000</td>
<td>-4478</td>
<td>-3188</td>
</tr>
<tr>
<td>-7.5</td>
<td>0.0578125</td>
<td>-5886</td>
<td>-6545</td>
<td>-4515</td>
</tr>
<tr>
<td>-8.5</td>
<td>0.0571875</td>
<td>-9330</td>
<td>-10450</td>
<td>-7305</td>
</tr>
<tr>
<td>-9.5</td>
<td>0.0565625</td>
<td>-15650</td>
<td>-17575</td>
<td>-12128</td>
</tr>
<tr>
<td>-10</td>
<td>0.0559375</td>
<td>-15777</td>
<td>-17656</td>
<td>-12500</td>
</tr>
</tbody>
</table>

Table A7-5 Adaptive control threshold parameters ±10% range
<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Sine</th>
<th>Filtered Pulse</th>
<th>Manch’ RLL3</th>
<th>RLL7</th>
</tr>
</thead>
<tbody>
<tr>
<td>+7.5</td>
<td>0.0671875</td>
<td>17144</td>
<td>18329</td>
<td>13367</td>
<td>6543</td>
</tr>
<tr>
<td>+6.5</td>
<td>0.0665625</td>
<td>11848</td>
<td>12565</td>
<td>9054</td>
<td>4912</td>
</tr>
<tr>
<td>+5.5</td>
<td>0.0659375</td>
<td>6889</td>
<td>7355</td>
<td>5185</td>
<td>2792</td>
</tr>
<tr>
<td>+4.5</td>
<td>0.0653125</td>
<td>4359</td>
<td>4707</td>
<td>3337</td>
<td>1822</td>
</tr>
<tr>
<td>+3.5</td>
<td>0.0646875</td>
<td>2821</td>
<td>3158</td>
<td>2022</td>
<td>1211</td>
</tr>
<tr>
<td>+2.5</td>
<td>0.0640625</td>
<td>1772</td>
<td>1711</td>
<td>1142</td>
<td>616</td>
</tr>
<tr>
<td>+1.5</td>
<td>0.0643750</td>
<td>991</td>
<td>971</td>
<td>517</td>
<td>417</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0628125</td>
<td>379</td>
<td>292</td>
<td>332</td>
<td>31</td>
</tr>
<tr>
<td>-0.5</td>
<td>0.0621875</td>
<td>-379</td>
<td>-312</td>
<td>53</td>
<td>-196</td>
</tr>
<tr>
<td>-1.5</td>
<td>0.0615625</td>
<td>991</td>
<td>-997</td>
<td>-753</td>
<td>-529</td>
</tr>
<tr>
<td>-2.5</td>
<td>0.0609375</td>
<td>-1772</td>
<td>-1516</td>
<td>-863</td>
<td>-631</td>
</tr>
<tr>
<td>-3.5</td>
<td>0.0603125</td>
<td>-2821</td>
<td>-3278</td>
<td>-2218</td>
<td>-1503</td>
</tr>
<tr>
<td>-4.5</td>
<td>0.0596875</td>
<td>-4359</td>
<td>-4917</td>
<td>-3507</td>
<td>-1810</td>
</tr>
<tr>
<td>-5.5</td>
<td>0.0590625</td>
<td>-6889</td>
<td>-7612</td>
<td>-5543</td>
<td>-2070</td>
</tr>
<tr>
<td>-6.5</td>
<td>0.0584375</td>
<td>-11848</td>
<td>-13077</td>
<td>-9558</td>
<td>-5252</td>
</tr>
<tr>
<td>-7.5</td>
<td>0.0578125</td>
<td>-17144</td>
<td>-19206</td>
<td>-13895</td>
<td>-9839</td>
</tr>
</tbody>
</table>

Table A7-6 Adaptive control threshold parameters ±7.5% range

<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Sine</th>
<th>Filtered Pulse</th>
<th>Manch’ RLL3</th>
<th>RLL7</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5.5</td>
<td>0.0659375</td>
<td>14739</td>
<td>15808</td>
<td>11655</td>
<td>9720</td>
</tr>
<tr>
<td>+4.5</td>
<td>0.0653125</td>
<td>14510</td>
<td>15524</td>
<td>11490</td>
<td>8621</td>
</tr>
<tr>
<td>+3.5</td>
<td>0.0646875</td>
<td>8085</td>
<td>8801</td>
<td>6226</td>
<td>4766</td>
</tr>
<tr>
<td>+2.5</td>
<td>0.0640625</td>
<td>4381</td>
<td>4909</td>
<td>3440</td>
<td>2623</td>
</tr>
<tr>
<td>+1.5</td>
<td>0.0643750</td>
<td>2248</td>
<td>2266</td>
<td>1713</td>
<td>1228</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0628125</td>
<td>825</td>
<td>846</td>
<td>483</td>
<td>403</td>
</tr>
<tr>
<td>-0.5</td>
<td>0.0621875</td>
<td>-825</td>
<td>-826</td>
<td>-527</td>
<td>-443</td>
</tr>
<tr>
<td>-1.5</td>
<td>0.0615625</td>
<td>-2248</td>
<td>-2270</td>
<td>-1777</td>
<td>-1279</td>
</tr>
<tr>
<td>-2.5</td>
<td>0.0609375</td>
<td>-4381</td>
<td>-5054</td>
<td>-3612</td>
<td>-2743</td>
</tr>
<tr>
<td>-3.5</td>
<td>0.0603125</td>
<td>-8085</td>
<td>-9026</td>
<td>-6649</td>
<td>-4892</td>
</tr>
<tr>
<td>-4.5</td>
<td>0.0596875</td>
<td>-14510</td>
<td>-16003</td>
<td>-11930</td>
<td>-8909</td>
</tr>
<tr>
<td>-5.5</td>
<td>0.0590625</td>
<td>-14739</td>
<td>-16373</td>
<td>-12014</td>
<td>-10200</td>
</tr>
</tbody>
</table>

Table A7-7 Adaptive control threshold parameters ±5% range
Table A7-8 shows the corresponding coefficient and gain, which are applied to the adaptive filter for a given frequency offset.

<table>
<thead>
<tr>
<th>% Offset Bin #</th>
<th>Symbol Rate Hz</th>
<th>Coef C</th>
<th>Gain A</th>
</tr>
</thead>
<tbody>
<tr>
<td>+10</td>
<td>0.068750</td>
<td>1.812735</td>
<td>1.10204</td>
</tr>
<tr>
<td>+9</td>
<td>0.068125</td>
<td>1.816003</td>
<td>1.09090</td>
</tr>
<tr>
<td>+8</td>
<td>0.067500</td>
<td>1.819242</td>
<td>1.08000</td>
</tr>
<tr>
<td>+7</td>
<td>0.066875</td>
<td>1.822454</td>
<td>1.06931</td>
</tr>
<tr>
<td>+6</td>
<td>0.066250</td>
<td>1.825638</td>
<td>1.05882</td>
</tr>
<tr>
<td>+5</td>
<td>0.065625</td>
<td>1.828793</td>
<td>1.04854</td>
</tr>
<tr>
<td>+4</td>
<td>0.065000</td>
<td>1.831920</td>
<td>1.03846</td>
</tr>
<tr>
<td>+3</td>
<td>0.064375</td>
<td>1.835017</td>
<td>1.02857</td>
</tr>
<tr>
<td>+2</td>
<td>0.063750</td>
<td>1.838090</td>
<td>1.01886</td>
</tr>
<tr>
<td>+1</td>
<td>0.063125</td>
<td>1.841113</td>
<td>1.00934</td>
</tr>
<tr>
<td>0</td>
<td>0.062500</td>
<td>1.844146</td>
<td>1.00000</td>
</tr>
<tr>
<td>-1</td>
<td>0.061875</td>
<td>1.847132</td>
<td>0.99082</td>
</tr>
<tr>
<td>-2</td>
<td>0.061250</td>
<td>1.850089</td>
<td>0.98182</td>
</tr>
<tr>
<td>-3</td>
<td>0.060625</td>
<td>1.853017</td>
<td>0.97297</td>
</tr>
<tr>
<td>-4</td>
<td>0.060000</td>
<td>1.855917</td>
<td>0.96428</td>
</tr>
<tr>
<td>-5</td>
<td>0.059375</td>
<td>1.858788</td>
<td>0.95575</td>
</tr>
<tr>
<td>-6</td>
<td>0.058750</td>
<td>1.861631</td>
<td>0.94737</td>
</tr>
<tr>
<td>-7</td>
<td>0.058125</td>
<td>1.864445</td>
<td>0.93913</td>
</tr>
<tr>
<td>-8</td>
<td>0.057500</td>
<td>1.867230</td>
<td>0.93103</td>
</tr>
<tr>
<td>-9</td>
<td>0.056875</td>
<td>1.869986</td>
<td>0.92308</td>
</tr>
<tr>
<td>-10</td>
<td>0.056250</td>
<td>1.872714</td>
<td>0.91525</td>
</tr>
</tbody>
</table>

Table A7-8 Coefficient and gain parameters

The data from tables Table A7-5 to Table A7-8 are entered into the parameters of the adaptive control block. A separate adaptive control block is defined for each of the tracking ranges and encoded run lengths.
Simulation results

The following sections are a collection of key simulation results. Performance is measured over the following acquisition and tracking ranges relative to the coded run length:

- Manchester data: 5%, 7.5% and 10%.
- RLL3 data: 5% and 7.5%.
- RLL7 data: 5%.

These results are discussed in Chapter 3 and are used to measure:

- Clock acquisition performance.
- Clock flywheel performance.
- Frequency tracking performance.

Acquisition performance overview

Figure A7-8 to Figure A7-10 summarises a collection of simulation data to demonstrate the number of symbols taken, for the tracking filter to adapt to an input frequency offset. Since the bandwidth of the tracking filter can tolerate a ±1% deviation in frequency, successful acquisition is assumed to within 1%.

For an acquisition range of ±5%, the detection filters are positioned relatively close together, producing a symmetrical acquisition response, as shown in Figure A7-8. As the detection filter spacing is increased, then the low level aliased harmonics together with an increasing encoded run length, distorts the acquisition symmetry, as shown in Figure A7-9 and Figure A7-10.
Figure A7-8 Acquisition performance ±5% range

Figure A7-9 Acquisition performance ±7.5% range

Figure A7-10 Acquisition performance ±10% range
Manchester encoded data ±5% tracking range

**Figure A7-11** Manchester - Clock acquisition with a +5% offset

**Figure A7-12** Manchester - Flywheel decay period with a +5% offset
Figure A7-13 Manchester - 80 symbol dropout with a +5% offset

Figure A7-14 Manchester - 80 symbol dropout with a +5% offset (zoomed in)
Figure A7-15 Manchester - Tracking frequency drift over a ±5% range

Figure A7-16 Manchester - Tracking frequency drift over a ±5% range (zoomed in worst case amplitude)
Manchester encoded data ±7.5% tracking range

Figure A7-17 Manchester - Clock acquisition with a +7% offset

Figure A7-18 Manchester - Flywheel decay period with a +7% offset
Figure A7-19 Manchester - 80 symbol dropout with a +7% offset

Figure A7-20 Manchester - 80 symbol dropout with a +7% offset (zoomed in)
APPENDIX 7

ADAPTIVE TRACKING CLOCK RECOVERY

Control Voltage to VCO generating drift

Bin # for Adaptive Filter (+-7.5% Tracking Range)

Adaptive Filter

Figure A7-21 Manchester - Tracking frequency drift over a ±7.5% range

PRBS Manchester Encoded (Frequency Drifting)

Adaptive Filter

Recovered Clock from Adaptive Filter

Figure A7-22 Manchester - Tracking frequency drift over a ±7.5% range (zoomed in worst case amplitude)
Manchester encoded data ±10% tracking range

Figure A7-23 Manchester - Clock acquisition with a +10% offset

Figure A7-24 Manchester - Flywheel decay period with a +10% offset
APPENDIX 7  ADAPTIVE TRACKING CLOCK RECOVERY

Figure A7-25 Manchester - 80 symbol dropout with a +10% offset

Figure A7-26 Manchester - 80 symbol dropout with a +10% offset (zoomed in)

A7-23
Figure A7-27 Manchester - Tracking frequency drift over a ±10% range

Figure A7-28 Manchester - Tracking frequency drift over a ±10% range (zoomed in worst case amplitude)
RLL3 encoded data ±5% tracking range

![Diagram of PRBS RLL3 Data (±5% Frequency Offset)]

![Diagram of Fixed Filter]

![Diagram of Bin # for Adaptive Filter]

![Diagram of Adaptive Filter]

![Diagram of Recovered Clock from Adaptive Filter]

Figure A7-29 RLL3 - Clock acquisition with a +5% offset

![Diagram of Bin # for Adaptive Filter (+-5% Tracking Range)]

![Diagram of Tracking control decays]

![Diagram of Adaptive Filter]

![Diagram of 143 symbols]

Figure A7-30 RLL3 - Flywheel decay period with a +5% offset
Figure A7-31 RLL3 - 80 symbol dropout with a +5% offset

Figure A7-32 RLL3 - 80 symbol dropout with a +5% offset (zoomed in)
APPENDIX 7  
ADAPTIVE TRACKING CLOCK RECOVERY

Control Voltage to VCO generating drift

Bin * for Adaptive Filter (+-5% Tracking Range)

Adaptive Filter

Figure A7-33 RLL3 - Tracking frequency drift over a ±5% range

PRBS RLL3 Data (Frequency Drifting)

Adaptive Filter

Recovered Clock from Adaptive Filter

Figure A7-34 RLL3 - Tracking frequency drift over a ±5% range (zoomed in worst case amplitude)
RLL3 encoded data ±7.5% tracking range

Figure A7-35 RLL3 - Clock acquisition with a +7% offset

Figure A7-36 RLL3 - Flywheel decay period with a +7% offset
APPENDIX 7  
ADAPTIVE TRACKING CLOCK RECOVERY

Figure A7-37 RLL3 - 80 symbol dropout with a +7% offset

Figure A7-38 RLL3 - 80 symbol dropout with a +7% offset (zoomed in)
Control Voltage to VCO generating drift

Bin # for Adaptive Filter (+−7.5% Tracking Range)

Adaptive Filter

Figure A7-39 RLL3 - Tracking frequency drift over a ±7.5% range

PRBS RLL3 Data (Frequency Drifting)

Adaptive Filter

Recovered Clock from Adaptive Filter

Figure A7-40 RLL3 - Tracking frequency drift over a ±7.5% range (zoomed in worst case amplitude)
RLL7 encoded data ±5% tracking range

Figure A7-41 RLL7 - Clock acquisition with a +5% offset

Figure A7-42 RLL7 - Flywheel decay period with a +5% offset
Figure A7-43 RLL7 - 80 symbol dropout with a +5% offset

Figure A7-44 RLL7 - 80 symbol dropout with a +5% offset (zoomed in)
**APPENDIX 7**

**ADAPTIVE TRACKING CLOCK RECOVERY**

-control voltage to VCO generating drift

-bin # for Adaptive Filter (+-5% Tracking Range)

-Adaptive Filter

Figure A7-45 RLL7 - Tracking frequency drift over a ±5% range

-PRBS RLL7 Data (Frequency Drifting)

-Adaptive Filter

-Recovered Clock from Adaptive Filter

Figure A7-46 RLL7 - Tracking frequency drift over a ±5% range (zoomed in worst case amplitude)
Sine input ±10% tracking range

Tracking performance of the adaptive filter has been evaluated with a sinusoidal input over a ±10% tracking range. A sine wave is generated by a VCO, which is frequency modulated by a triangular signal connected to the VCO control input. This results in the frequency of the VCO, sweeping linearly up and down the ±10% tracking range. The resulting frequency modulated sine wave is then applied to the adaptive filter and the tracking performance analysed.

Figure A7-47 Sine input - Adaptive tracking simulation model
Figure A7-48 shows a frequency drifting at a rate of 10% per sample (s3), relative to a fixed frequency of 0.0625Hz (s2). Close examination of the adaptive filter output (s5) confirms that the filter is successfully tracking the frequency and phase of the drifting sinusoidal input. The quantized frequency bin number (s4) switches frequency bins as it follow the drifting input. The output of the adaptive filter is hard limited with a zero threshold to provide a tracking recovered clock (s6).
Figure A7-49 shows the tracking performance (wide view) of the adaptive system. Signal (s2) shows the error signal from the frequency-offset detector, which is translated into linear frequency estimation (s3) by the adaptive control block. This frequency estimation follows the modulation of the control signal to the VCO (s1), confirming successful tracking. If the frequency drift abruptly changes direction, then the tracking filter sharply drops in amplitude (s4), while the filter adapts to a change in direction of the input phase. However, close examination shows that the filter output does not fall to zero and quickly returns to full amplitude as the filter re-adapts to the change in phase.
Appendix 8

FFT Derivation

The Fourier Series states that any periodic signal can be made up from an infinite number of sine and cosine signals of varying amplitude. Hence, any arbitrary periodic waveform may be represented by a Fourier Series, given by:

\[ x(t) = a_o + \sum_{k=-\infty}^{\infty} a_k \cos(k\omega t) + \sum_{k=-\infty}^{\infty} b_k \sin(k\omega t) \]  \hspace{1cm} (A8-1)

Where \( x(t) \) represents a varying voltage with time and \( \omega = 2\pi f \) is the fundamental angular frequency. The series may be re-written using exponential notation [2].

\[ x(t) = \sum_{k=-\infty}^{\infty} C_k e^{j(k\omega t)} \]  \hspace{1cm} (A8-2)

Where \( C_k \) consist of complex phasors. This continuous time equation, can be translated into discrete form by replacing \( t \) with discrete samples \( nT \):

\[ x(n) = \sum_{k=-\infty}^{\infty} C_k e^{j(k\omega nT)} \]  \hspace{1cm} (A8-3)

Equation (A8-3) assumes a harmonically related periodic signal, where \( T_s \) is the repetition period. For non-periodic signals the number of phasors tends towards infinity, \( \omega \to \infty \), and the summation becomes an integral.

\[ x(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(\omega) \cdot e^{j(\omega nT_s)} \cdot d(\omega T_s) \]  \hspace{1cm} (A8-4)

And the reverse transform becomes:

\[ X(\omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-j(\omega nT_s)} \]  \hspace{1cm} (A8-5)
Equations (A8-4) and (A8-5) form the DFT (Discrete Fourier Transform) pair and allow discrete signals to be transformed between the time and frequency domains.

In equation (A8-3), when the phase jumps for the kth harmonic, the phase rotates by $2\pi$ ($360^\circ$) and is indistinguishable from when $k=0$. Hence, the frequency response of a discrete signal is periodic, with a period of $1/T_s$. In other words, the spectrum repeats around the sample rate frequency, which is fundamental to sampling theory.

In equation (A8-5) it is not practical to perform an infinite number of summations and therefore only a windowed number of samples are taken, denoted by $X_N(k)$.

Since the frequency scale is now digitized, $\omega$ is replaced by $k$, and the sample period is equal to $\omega_s/N$, where $N$ is the number of points in the windowed spectrum.

Equation (A8-5) is then rewritten as:

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi nk}{N}\right)}$$  \hspace{1cm} (A8-6)

Equation (A8-6) is the practical form of a DFT that is generally used, and the twiddle factor $W_N$ is defined as:

$$W_N = e^{-j\left(\frac{2\pi}{N}\right)}$$  \hspace{1cm} (A8-7)

The DFT pair can now be rewritten in their most common form:

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn}$$  \hspace{1cm} (A8-8)

And

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_N(k) \cdot W_N^{-kn}$$  \hspace{1cm} (A8-9)
The DFT is an effective method of determining the frequency spectrum of any signal, but is impractical in this standard form for implementation in DSP. Referring to equation (A8-8), a multiply-and-add function must be implemented for each of the indices n and k, hence \( N^2 \) computations must be performed. For a 1000-point DFT, \( 10^6 \) machine cycles would be required, limiting the sample rate to just 20Hz for a 50ns cycle time DSP. Fortunately, the FFT (Fast Fourier Transform) reduces the number of operations by optimising the redundancy contained within the twiddle factor, reducing the number of machine cycles to just 50,500 for a 1000-point FFT.

From equation (A8-8) it can be seen that the twiddle factor \( W_n^k \) is calculated many times and due to its periodicity contains only a limited number of values. It is the purpose of the FFT to use this redundancy to reduce the number of calculations.

The FFT uses the principle of reducing a long DFT into a series of short, but simple DFT’s. DIT (Decimation in Time) is the process of splitting the input signal \( x(n) \) into many short interleaved sequences.

Given equation (A8-8).

\[
X_n(k) = \sum_{n=0}^{N-1} x(n) \cdot W_n^{kn}
\]

Decimate \( x_n \) into odd and even sequences to give two \( N/2 \) point DFT’s:

\[
X_N(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} W_N^{2(n+1)k} \quad \ldots (A8-10)
\]

\[
X_N(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} (W_N^2)^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} (W_N^2)^{nk} \quad \ldots (A8-11)
\]
For the even term, n is replaced with 2n, while for the odd term, n is replaced with 2n+1.

Since 

\[ W_N = e^{-j\frac{2\pi}{N}} \]

then 

\[ W_N^2 = e^{-j\frac{2\pi}{N^2}} = e^{-j\frac{2\pi}{N/2}} = W_N^{N/2} \]

Then equation (A8-11) becomes:

\[ X_N(k) = \sum_{n=0}^{N-1} x_{2n} W_N^{nk} + W_N^k \sum_{n=0}^{N-1} x_{2n+1} W_N^{nk} \] \hspace{1cm} \text{(A8-12)}

Which can be written as:

\[ X_N(k) = F_{\text{even}}(k) + W_N^k F_{\text{odd}}(k) \] \hspace{1cm} \text{(A8-13)}

Where \( F_{\text{even}}(k) \) is the DFT of the even numbered points and \( F_{\text{odd}}(k) \) is the DFT of the odd numbered points.

The original DFT is now expressed in the form of two smaller DFTs of length N/2. Whereas equation (A8-8) needed evaluation over the range \( k = 0, 1, \ldots, N-1 \), equation (A8-13) only requires calculations of \( k = 0, 1, \ldots, \lceil \frac{N}{2} \rceil \), thereby halving the number of complex multiplication's. Hence, for a 1000-point DFT, the calculations have been reduced to \( 500^2 + 500^2 + 500 = 50,500 \), rather than \( 10^6 \) for the original DFT.

Decimation can continue, splitting these two sequences into two more, and so on, until the limit of this process is reached, obtaining sets of just 2-point DFTs, known as a radix-2. The complete FFT is calculated by initially performing the individual sets of 2-point DFTs, which greatly simplifies the mathematics.
However, the twiddle factors still need to be considered, which arise at every stage of decimation. A good FFT should incorporate these factors into the mathematics without imposing much processing overhead.

Consider a simple example, where \( N=4 \).

Rewriting equation (A8-8):

\[
X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn} 
\]

Substituting for \( N=4 \).

\[
X_4(k) = \sum_{n=0}^{4-1} x(n) \cdot W_4^{kn} \quad \text{(A8-14)}
\]

Decimating into odd and even sequences, from equation (A8-12):

\[
X_4(k) = \sum_{n=0}^{\frac{4-1}{2}} x_{2n} W_2^{nk} + W_4^k \sum_{n=0}^{\frac{4-1}{2}} x_{2n+1} W_2^{nk}
\]

\[
X_4(k) = \sum_{n=0}^{1} x_{2n} W_2^{nk} + W_4^k \sum_{n=0}^{1} x_{2n+1} W_2^{nk}
\]

\[
X_4(k) = [x(0) + x(2) \cdot W_2^k] + W_4^k [x(1) + x(3) \cdot W_2^k]
\]

since

\[
W_2^k = e^{-j \frac{2\pi k}{2}} = e^{-j \frac{\pi k}{2}} = W_4^{2k}
\]

Then the FFT becomes:

\[
X_4(k) = [x(0) + x(2) \cdot W_4^{2k}] + W_4^k [x(1) + x(3) \cdot W_4^{2k}]
\]
Writing out all the possible values for $k$:

$$X_4(0) = [x(0) + x(2) \cdot W_4^0] + W_4^0 [x(1) + x(3) \cdot W_4^0]$$

$$X_4(1) = [x(0) + x(2) \cdot W_4^2] + W_4^1 [x(1) + x(3) \cdot W_4^2]$$

$$X_4(2) = [x(0) + x(2) \cdot W_4^0] + W_4^2 [x(1) + x(3) \cdot W_4^0]$$

$$X_4(3) = [x(0) + x(2) \cdot W_4^2] + W_4^3 [x(1) + x(3) \cdot W_4^2]$$

Observe that:

$$W_4^4 = e^{-j\frac{2\pi}{4}} = 1 = W_4^0 \quad \text{(A8-15)}$$

and

$$W_4^6 = e^{-j\frac{3\pi}{4}} = -j = W_4^2 \quad \text{(A8-16)}$$

Also

$$W_4^1 = e^{-j\frac{\pi}{4}} = e^{-j90^\circ} = \cos(-90^\circ) + j\sin(-90^\circ)$$

$$W_4^1 = -j \quad \text{(A8-17)}$$

$$W_4^2 = (W_4^1)^2 = j^2 = -1 \quad \text{(A8-18)}$$

and

$$W_4^3 = W_4^1W_4^2 = j \quad \text{(A8-19)}$$

Hence, the twiddle factors reduce to $\pm 1$ or $\pm j$ and it becomes a simple matter to include these factors into the mathematics for the FFT.

Although only a 4-point FFT has been considered, it is clear that an FFT of any size can be reduced to simple 2-point DFTs, as long as $N$ is an integer power of 2.

The 4-point FFT can be more easily visualised with the aid of the flow diagram, as shown in Figure A8-1.
Each section in the diagram is made up of a butterfly and it is generally accepted that the intersection represents $W_N^0 = 1$ and $W_N^2 = -1$. Hence, the above flow diagram can be redrawn in the standard form, as shown in Figure A8-2. Any additional twiddle factors are then drawn on the appropriate lines, as with $-j$.

It is apparent from the diagrams that in order for the output samples to be in sequence, then the input samples need to be shuffled. Fortunately, DSP processors perform this operation automatically with zero over-head, by means of bit-reversed addressing.
To obtain the output in ascending order, the input samples must be loaded into the look-up table in the following order: n = 0,2,1,3. This is trivial for the 4-point FFT, but for a 1024-point FFT, things are rather more complicated.

Bit reversed addressing is the process of adding half the FFT length (N/2) at each address access, but performing bit carries from left to right.

This process is best visualised with an example:

Consider an 8-point FFT, containing $2^3 = 8 = 111_2$ possible memory locations and $N/2 = 4 = 100_2$.

<table>
<thead>
<tr>
<th>Physical address</th>
<th>Bit Rev address + N/2</th>
<th>Bit Rev address</th>
<th>Sample $N^0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000 start = 000</td>
<td>000</td>
<td>$x(0)$</td>
</tr>
<tr>
<td>001</td>
<td>000 + 100 = 100</td>
<td>100</td>
<td>$x(4)$</td>
</tr>
<tr>
<td>010</td>
<td>100 + 100 = 010</td>
<td>010</td>
<td>$x(2)$</td>
</tr>
<tr>
<td>011</td>
<td>010 + 100 = 110</td>
<td>110</td>
<td>$x(6)$</td>
</tr>
<tr>
<td>100</td>
<td>110 + 100 = 001</td>
<td>001</td>
<td>$x(1)$</td>
</tr>
<tr>
<td>101</td>
<td>001 + 100 = 101</td>
<td>101</td>
<td>$x(5)$</td>
</tr>
<tr>
<td>110</td>
<td>101 + 100 = 011</td>
<td>011</td>
<td>$x(3)$</td>
</tr>
<tr>
<td>111</td>
<td>011 + 100 = 111</td>
<td>111</td>
<td>$x(7)$</td>
</tr>
</tbody>
</table>

Hence, this process can be extended to any size FFT.
A9 - Appendix 9

OFFT-based clock recovery

This appendix supplements Chapter 4, and provides simulation results to evaluate performance of the OFFT-based clock recovery algorithm. The simulation model is described, and the C-language source code is provided for the developed OFFT custom-coded block. The assembler macro for the real-time DSP (TMS320C50) implementation of the OFFT is provided, together with the C-language source code for “twiddle.exe”, used to generate the twiddle factors for an OFFT.

Simulation model

An overview of the simulation model is shown in Figure A9-1. Several hierarchy custom blocks have been created. In particular, a custom-coded OFFT block has been written in the C-language, compiled and linked to a symbol, which has been used in top level simulation systems. Source code for the OFFT block is included in this appendix. Figure A9-2 shows the full SPW simulation system. Parameters in sub-hierarchy blocks are exported to this top-level system for ease of viewing an adjustment.

Figure A9-1 Overview of the OFFT-based adaptive simulation model
Figure A9-2 OFFT-based system - Simulation model
A pseudo random binary sequence is used to generate data with a maximum run length of 7 symbols. The data generator may be replaced with a signal generator, to simulate an OFFT with a sinusoidal input. A “delay and multiply” timing conditioner is used to extract a symbol-timing frequency component, for input to the adaptive clock recovery filter and the real input of the OFFT. The imaginary input to the OFFT is set to zero. The output from the OFFT is threshold detected, which returns a bin number. A guard band of 10 samples is provided to prevent detection of a spectral component at dc. Depending upon the frequency of the input symbols, the threshold may achieve one or two hits. The “FREQ EST” block is used to convert the bin numbers to a corresponding frequency estimate.

The sub hierarchy detail of the “FREQ EST” block is shown in Figure A9-3.

![Figure A9-3 “FREQ EST” block - Sub hierarchy detail](image)
The "FREQ EST" block implements equations (A9-1) to (A9-3).

If \( k_{\text{max}} < (N/2)-1 \) then \( f_{\text{est}1} = (k_{\text{max}} + 0.25) \cdot f_s/N \) Hz \hspace{1cm} (A9-1)

or \( k_{\text{max}} > (N/2)-1 \) then \( f_{\text{est}2} = (N - k_{\text{max}} - 0.25) \cdot f_s/N \) Hz \hspace{1cm} (A9-2)

or \( k_{\text{max}} < (N/2)-1 \) and \( k_{\text{max}} > (N/2)-1 \) then \( f_{\text{est}3} = \frac{f_{\text{est}1} + f_{\text{est}2}}{2} \) Hz \hspace{1cm} (A9-3)

On completion of the FFT, the returned frequency estimate is sampled and applied to the "FREQ to COEF" block, to determine the parameters necessary to reposition the adaptive clock recovery filter, to the frequency of the applied symbols.

The sub-hierarchy detail of the FREQ EST block is shown in Figure A9-4.

![Figure A9-4 "FREQ to COEF" block - Sub hierarchy detail](image)

The "FREQ to COEF" block implements equations (A9-4) and (A9-5).

Coefficient \( C(f) = \sqrt{K} \cdot 2\cos \left( \frac{2\pi f}{f_s} \right) \) ........................................ (A9-4)

Gain \( A(f) = \frac{(100 \cdot f_s) - 13}{(100 \cdot f_s) - 13} \) ........................................ (A9-5)
The coefficient and gain are applied to the adaptive filter, which snaps to the centre frequency of the input symbols. The filter is then enabled and the hard limited filter output provides the recovered clock. Performance is then identical to that of the fixed frequency clock recovery system.

Simulation results are shown in for RLL7 data, with a symbol frequency offset from the nominal (0.0625Hz) of +20%. This corresponds to a normalised absolute frequency of 0.075Hz.

Signal parameters are included in Figure A9-5 to allow cursor measurements of the following.

- Power threshold detected bin number = 947.
- Frequency estimate = 0.07495117 Hz.
- Number of OFFT points = 1024.
- Coefficient C = 1.778807.
- Gain A = 1.226184.
Simulation results

Figure A9-5 provides a full simulation of the OFFT-based clock recovery system.
OFFT - SPW custom coded block source code

The following C-language source code has been used to generate the SPW "CUSTOM CODED OFFT" block.

**OFFT.C Module**

```c
#include "spw_platform.h"
#ifdef UNIX
#include "caedata/psmithso_lib/offt/blockcode/offtu.c"
#else
#ifdef VAX_VMS
#include "[caedata.psmithso_lib.offt.blockcode)offtu.c"
#endif VAX_VMS
#endif UNIX
static char *REVISION = "2.50";
/
* Block Function: offt 
* Library: psmithso_lib 
/
/**************************************************************************//*
* FEED_THROUGH_LIST INFORMATION: 
/**************************************************************************//*
* --> FEED_THROUGH_TYPE IS NOT EDITABLE. The BDE parameter 
* screen associated with the block must be edited to change 
* the block's FEED_THROUGH_TYPE. 
* FEED_THROUGH_TYPE = ALL_FEED_THROUGH. 
/**************************************************************************//*
/**************************************************************************//*
* LINK_OPTIONS INFORMATION: 
/**************************************************************************//*
* --> The LINK_OPTIONS list is editable. It contains all the 
* libraries which the code must be linked to. Each item in 
* the list must be surrounded by double quotes and 
* separated by commas. The math library is automatically 
* linked, and does not need to be specified. The paths 
* may be specified as full paths or as paths relative to 
* the host. 
* A link option can also be specified in the form '-lx' 
* (where x is defined in the UNIX manual on 'ld' 
* IMPORTANT: The entire LINK_OPTIONS list must be deleted 
* if it doesn't contain any elements. 
* Sample LINK_OPTIONS list: 
* (Actual list should be placed below this comment block) 
* LINK_OPTIONS = { "-lm", 
* "//host/code/lib/sample.a" }; 
/**************************************************************************//*
```

A9-7
INCLUDE_DIRS INFORMATION:

--> The INCLUDE_DIRS list is editable. The list should contain all directory search paths needed to locate all the include files used by this block. It has the same format as the LINK_OPTIONS list.

IMPORTANT: The entire INCLUDE_DIRS list must be deleted if it doesn't contain any elements.

Sample INCLUDE_DIRS list:
(Actual list should be placed below this comment block)

INCLUDE_DIRS = { "/host/u/code/include", "/host/lib/dir" );

EDITABLE FUNCTIONS

--> In_offt_psmithso_lib ()
--> Ro_offt_psmithso_lib ()
--> Te_offt_psmithso_lib ()

Structure use:

Typical input value reference
local_var = *(spb_input->var_name);

**OR**
local_var = i_var_name;

Typical output value update
spb_output->var_name = local_var;

**OR**
O_var_name = local_var;

Typical parameter reference
local_var = spb_parm->var_name;

**OR**
local_var = P_var_name;

(See reference manual for further information)

Initialize Function (must be present)

--> If editing, modify only the lines within the function's opening and closing brackets.

This function is used to initialize the state structure and constant outputs of the block. It is called once for each block instance during simulation.

Function must always return either SYS_OK, SYS_TERM, or SYS_FATAL by using the return() function.
User may modify the line containing "return(SYS_OK);".

In_offt_psmithso_lib (spb_parm, spb_input, spb_output, spb_state)
STRUCT P_t_offt_psmithso_lib *spb_parm;
STRUCT It_offt_psmithso_lib *spb_input;
STRUCT Ot_offt_psmithso_lib *spb_output;
STRUCT St_offt_psmithso_lib *spb_state;
{
    double M_size;
    double M_test;
    int i;
if ( strcmp(P_FFT_dir, "forward") == 0 ) {
    S_DFT_SIGN = -1.0;
} else if ( strcmp(P_FFT_dir, "Forward") == 0 ) {
    S_DFT_SIGN = -1.0;
} else if ( strcmp(P_FFT_dir, "FORWARD") == 0 ) {
    S_DFT_SIGN = -1.0;
} else if ( strcmp(P_FFT_dir, "inverse") == 0 ) {
    S_DFT_SIGN = 1.0;
} else if ( strcmp(P_FFT_dir, "Inverse") == 0 ) {
    S_DFT_SIGN = 1.0;
} else if ( strcmp(P_FFT_dir, "INVERSE") == 0 ) {
    S_DFT_SIGN = 1.0;
} else {
    wmsgErrors ("ERROR: 'Forward/Backward' Parameter
incorrectly entered.");
    sprintf(wmsgbuf, "Instance number = %ld", spb_state->instance);
}
M_size = 0.0;
M_test = 0.0;
M_size = log(I_X_re_iovec_len)/log(2);
M_test = M_size - floor(M_size);
if ( M_test == 0.0 ) {
    S_M = (int) M_size;
} else if ( M_test != 0.0 ) {
    wmsgErrors ("ERROR: OFFT size is not a power of 2.");
    sprintf(wmsgbuf, "Instance number = %ld", spb_state->instance);
}
if ( (P_OFFSET < -0.5) || (P_OFFSET > 0.5) ) {
    wmsgErrors ("ERROR: OFFSET value outside permitted range.");
    sprintf(wmsgbuf, "Instance number = %ld", spb_state->instance);
}
S_PI = PI;
S_REAL = (double*)malloc(sizeof(double)*I_X_re_iovec_len);
S_IMAG = (double*)malloc(sizeof(double)*I_X_im_iovec_len);
for ( i = 0; i < I_X_re_iovec_len ; i++ ) {
    S_REAL[i] = 0.0;
    S_IMAG[i] = 0.0;
}
return (SYS_OK);

Run Output Function (must be present)

--> If editing, modify only the lines within the
    function's opening and closing brackets.

This function is used to update the outputs and/or state
of the block. It is called each iteration, for each
block instance during simulation.

Function must always return either SYS_OK, SYS_TERM,
or SYS_FATAL by using the return() function.
User may modify the line containing "return(SYS_OK);".

A9-9
APPENDIX 9

OFFT-BASED CLOCK RECOVERY

Ro_offt_psmithso_lib (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_offt_psmithso_lib *spb_parm;
STRUCT It_offt_psmithso_lib *spb_input;
STRUCT Ot_offt_psmithso_lib *spb_output;
STRUCT St_offt_psmithso_lib *spb_state;
{
    int p;
    int m;
    int j;
    int s;
    int q;
    int k;
    int l;

    int N1;
    int N2;
    int N3;
    double ur;
    double ui;
    double wr;
    double wi;
    double temp;

    double XT;
    double YT;

    int j1;
    int il;
    int kl;

    double tr;
    double ti;

    /* FILL UP WORKING ARRAYS FROM ACTUAL INPUTS***/
    for (p = 0; p < I_X_re_iovec_len ; p++) {
        S_REAL[p] = I_X_re[p];
        S_IMAG[p] = I_X_im[p];
    }

    /* BIT SHUFFLING INPUTS***/
    j1 = 1;
    for( il = 1; il < I_X_re_iovec_len ; il++){
        if ( il < j1 ){
            tr = S_REAL[j1-1];
            ti = S_IMAG[j1-1];
            S_REAL[j1-1] = S_REAL[il-1];
            S_IMAG[j1-1] = S_IMAG[il-1];
            S_REAL[il-1] = tr;
            S_IMAG[il-1] = ti;
            kl = (I_X_re_iovec_len/2);
            while ( kl < j1 ) {
                j1 = j1 -kl;
                kl = kl/2;
            }
        } else {
            kl = (I_X_re_iovec_len/2);
            while ( kl < j1 ) {
                j1 = j1 - kl;
                kl = kl/2;
            }
        }
    }
}

j1 = j1 + kl;

A9-10
/**RADIX 2 DIT FFT SUB_ROUTINE/**/

```c
for ( m = 1; m <= S_M; m++ )
{
    N1 = pow( 2, (m) );
    N2 = N1/2;
    ur = cos((-1.0*P_OFFSET*2*S_PI/pow(2,m)));
    ui = sin((-1.0*P_OFFSET*2*S_PI/pow(2,m)));
    wr = cos(S_PI/N2);
    wi = S_DFT_SIGN*sin(S_PI/N2);
    for ( j = 1; j <= N2; j++ )
    {
        s = j;
        while( s <= I_X_re_iovec_len )
        {
            N3 = s + N2;
            XT = S_REAL[N3-1]*ur - S_IMAG[N3-1]*ui;
            YT = S_IMAG[N3-1]*ur + S_REAL[N3-1]*ui;
            S_REAL[N3-1] = S_REAL[s-1] - XT;
            S_IMAG[N3-1] = S_IMAG[s-1] - YT;
            S_REAL[s-1] = S_REAL[s-1] + XT;
            S_IMAG[s-1] = S_IMAG[s-1] + YT;
            s = s + N1;
        }
        temp = ur*wr - ui*wi;
        ui = ui*wr + ur*wi;
        ur = temp;
    }
}
```

/****OUTPUT WORKING ARRAYS TO ACTUAL OUTPUT ARRAYS****/
/**** USING TEMPORARY WORKING ARRAYS FOR CALLS. INSTEAD OF ACTUAL OUTPUT ARRAYS ****/

```c
for ( q = 0; q < I_X_re_iovec_len ; q++ )
{
    if ( S_DFT_SIGN == 1.0 )
    {
        O_Y_re[q] = S_REAL[q];
        O_Y_im[q] = S_IMAG[q];
        O_2_power[q] = (S_REAL[q]*S_REAL[q] +
                     S_IMAG[q]*S_IMAG[q]);
    }
    else
    {
        O_Y_re[q] = S_REAL[q]/I_X_re_iovec_len;
        O_Y_im[q] = S_IMAG[q]/I_X_re_iovec_len;
        O_2_power[q] = (S_REAL[q]*S_REAL[q] +
                        S_IMAG[q]*S_IMAG[q])/(I_X_re_iovec_len*I_X_re_iovec_len);
    }
}
```

return (SYS_OK);

/*
   Termination Function (must be present)
   --> If editing, modify only the lines within the
      function's opening and closing brackets.
   * This function is used to dump the final state of the
      block. It is called once for each block instance
      during the simulation.
*/
Function must always return either SYS_OK, SYS_TERM, or SYS_FATAL by using the return() function. User may modify the line containing 'return(SYS_OK);'.

Te_offt_psmithso_lib (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_offt_psmithso_lib *spb_parm;
STRUCT It_offt_psmithso_lib *spb_input;
STRUCT Ot_offt_psmithso_lib *spb_output;
STRUCT St_offt_psmithso_lib *spb_state;
{
    free(S_REAL);
    free(S_IMAG);
    return (SYS_OK);
}

OFFT.h Module

#include "FBDEFS.h"

Block Function: offt
Library: psmithso_lib
Date: Thu Oct 31 11:43:05 1996

EDITABLE USER DEFINED STATE STRUCTURE

State Structure (User Defined, editable)

PARAMETER STRUCTURE, SIMULATOR DEFINED, UNEDITABLE

Parameter Structure, Simulator Defined, uneditable
/* Input Structure, Simulator Defined, uneditable */

```
STRUCT It_offt_psmithso_lib {
    long X_im_iovec_len;
    double *X_im;
    long X_re_iovec_len;
    double *X_re;
};
```

/** Output Structure, Simulator Defined, uneditable */

```
STRUCT Or_offt_psmithso_lib {
    long Y_im_iovec_len;
    double *Y_im;
    long Y_re_iovec_len;
    double *Y_re;
    long Z_power_iovec_len;
    double *Z_power;
};
```

The following #defines may be used to shorten references to members of the above structures.

```c
#define P_FFT_dir (spb_parm->FFT_dir)
#define P_OFFSET (spb_parm->OFFSET)
#define I_X_im_iovec_len (spb_input->X_im_iovec_len)
#define I_X_im (spb_input->X_im)
#define I_X_re_iovec_len (spb_input->X_re_iovec_len)
#define I_X_re (spb_input->X_re)
#define O_Y_im_iovec_len (spb_output->Y_im_iovec_len)
#define O_Y_im (spb_output->Y_im)
#define O_Y_re_iovec_len (spb_output->Y_re_iovec_len)
#define O_Y_re (spb_output->Y_re)
#define O_Z_power_iovec_len (spb_output->Z_power_iovec_len)
#define O_Z_power (spb_output->Z_power)
#define S_PI (spb_state->S_PI)
#define S_M (spb_state->M_STAGES)
#define S_DFT_SIGN (spb_state->IDFT_DFT)
#define S_REAL (spb_state->REAL_OUT)
#define S_IMAG (spb_state->IMAG_OUT)
```
C++ Source code for "twiddles.exe"

This section provides source code listings for the executable file "twiddles.exe" used to generate and sort the twiddle factors for an OFFT.

The executable programme requires the number of OFFT-points and offset to be entered and creates a text file (TWIDDLE.Q15), which can be called from the main DSP programme.

The TMS320C50 assembler source code should include the following lines to allow the generated file to be called.

```
FFTLEN .set N ; where N = 8 to 1024
.copy "TWIDDLE.Q15" ; automatically read the twiddles
```

Twiddles.CPP

Source code reproduced with permission of J.T. Slader.

```c
#include "stdlib.h"
#include "stdio.h"
#include "math.h"
#include "conio.h"
#define PI 3.141592654

void main(void){
    /*************************************************************************
    */
    /***************************************************************************/
    FILE *fileptr;
    char filename[80]="TWIDDLES.Q15", keypressed;
    float offset,real,imaj,invert;
    int i,j,k,index,count;
    clrscr();
    if ((fileptr=fopen(filename,"w"))==NULL){
        printf("Cannot open %s
",filename);
        getch();
        exit(1);
    }
    else{
        do{ 
            clrscr();
            printf("Enter the offset > ");
            scanf("%f",&offset);
        }while( (offset<-0.5) || (offset>0.5) );
```
APPENDIX 9

OFFT-BASED CLOCK RECOVERY

```c
do {
    clrscr();
    printf("Select FFT or IFFT
");
    printf("1) FFT
");
    printf("2) IFFT
");
    keypressed=getch();
    switch(keypressed) {
        case('1'):
            invert=1;
            break;
        case('2'):
            invert=-1;
            break;
    }
    while((keypressed!='1') && (keypressed!='2'));
    clrscr();

    fprintf(fileptr,"****************************************************\n");
    if(invert==1)
        fprintf(fileptr, "* TWIDDLE FACTORS FOR 8-POINT to 1024-POINT FFT's - Q15 FORMAT *\n");
    else
        fprintf(fileptr, "* TWIDDLE FACTORS FOR 8-POINT to 1024-POINT IFFT's - Q15 FORMAT *\n");

    fprintf(fileptr,"******************************************************************\n");
    for(i=8; i<=256; i*=2) {
        count=0;
        index=i/2;
        fprintf(fileptr,".if FFTLEN = %d\n", i);
        if(invert=1)
            fprintf(fileptr,"*!! Twiddle factors for an %d-Point FFT -\n", i);
        else
            fprintf(fileptr,"*!! Twiddle factors for an %d-Point IFFT -\n", i);

    for(j=i/2 ;j > =1;j=j /2 ){
        for(k = 0;k<(i/2);k=k+index){
            real=((2*PI)/i)*(k+(j*offset));
            freal = cos(real);
            real =cos(real)*32767;
            imag=((2*PI)/i)*(k+(j*offset));
            fimag = -sin(imaj);
            imag=invert*(-sin(imaj)*32767);
            if(k==0 && j=(i/2)){
                fprintf(fileptr, 'TWIDSTRT	.word	0%4.4xh		;W%3.3d+(%3.3d * %1.2f) R \n
', int(real) ,k,j,offset, freal);
            } else{
                fprintf(fileptr, 'TWIDSTRT	.word	0%4.4xh		;W%3.3d+(%3.3d * %1.2f) R \n
', int(real) ,k,j,offset, freal);
            } if(k==((i/2)-1) && j=1){
                fprintf(fileptr, 'TWIDEND	.word	0%4.4xh		;W%3.3d+(%3.3d * %1.2f) I \n
', int(imaj),k,j,offset, fimag);
            } else{
                fprintf(fileptr, 'TWIDEND	.word	0%4.4xh		;W%3.3d+(%3.3d * %1.2f) I \n
', int(imaj),k,j,offset, fimag);
            }
        }
    }
}
```

A9-15
APPENDIX 9

OFFT-BASED CLOCK RECOVERY

```c
} count++;
    
index=index/2;
    }
    fprintf(fileptr, " .endif\n\n\n" );
    fclose(fileptr);
    }
    printf("\nTWIDDLES.Q15 written to disk\n" );
    exit(0);

Twiddles.Q15

Example of generated twiddle factors for an 8-point OFFT.

*******************************************************************************
* TWIDDLE FACTORS FOR 8-POINT to 1024-POINT FFT's - Q15 FORMAT *
*******************************************************************************

    .if FFTLEN = 8
    *!! Twiddle factors for an 8-Point FFT - Q15 format
    *!! OFFSET = 0.250
    TWIDSTR .word 05a81h ;W000+(004 * 0.25) R 0.707107
              .word 0a57eh ;W000+(004 * 0.25) I -0.707107
              .word 07640h ;W000+(002 * 0.25) R 0.923880
              .word 0cf05h ;W000+(002 * 0.25) I -0.382683
              .word 0cf05h ;W002+(002 * 0.25) R -0.382683
              .word 089c0h ;W002+(002 * 0.25) I -0.923880
              .word 07d89h ;W000+(001 * 0.25) R 0.980785
              .word 0e708h ;W000+(001 * 0.25) I -0.195090
              .word 0471ch ;W001+(001 * 0.25) R 0.555570
              .word 09594h ;W001+(001 * 0.25) I -0.831470
              .word 0e708h ;W002+(001 * 0.25) R -0.195090
              .word 08277h ;W002+(001 * 0.25) I -0.980785
              .word 09594h ;W003+(001 * 0.25) R -0.831470
    TWIDEND .word 0b8e4h ;W003+(001 * 0.25) I -0.555570
    .endif

The full twiddle factor tables (for N = 16, 32, 64, 128, 512 and 1024) have been
omitted for brevity.
TMS320C50 DSP source code for the OFFT

There are several source code modules and macros, which have been developed for general use and are of considerable size. These include the following:

- **Link Command file.** Used to assign relocatable addresses, referenced in the assembler code, to the dual DSP processor target system.

- **Main Module.** Controls the sequence of programme execution and calls initialisation routines. Assigns workspace in memory, copies DSP code and look-up tables from slow EPROM into fast RAM. Defines algorithm specific labels and constants.

- **Housekeeping Module.** Used to configure the target system. Defines hardware specific labels and constants.

- **Uart Module.** Allows a dumb-terminal to display debug information and provides remote control of programme execution. Message macros are called to control communications with the UART and display menu messages.

- **ISR Module.** Interrupt Service routines, which contain time critical DSP code.

For brevity only the OFFT macros are included in this appendix. Since the imaginary input samples are set to zero, the first pass butterfly macro is optimised for speed, by reducing the number of complex multiplication's. The last pass macro is extended to calculate the power spectrum, with calls to subroutines to determine the peak power in each half of the spectrum.

Other source files are available separately on disk, in ASCII text format.
OFFT DSP-macros

Source code reproduced in co-operation with J.T. Slader.

OFFT -MACRO
*******************************************************************************
* Purpose: Decide which buffer to do OFFT on and set up Pointers etc... *
* If program in DARAM & Buffers in SARAM; *
*******************************************************************************
INITFFT ;MACRO - Set up pointers, increments etc..
FIRSTPASS ;MACRO - PASS 0 of an N=256 OFFT (c=0.25)
BFLY 0 ;MACRO - Bfly calculations for interim PASSES
LASTPASS ;MACRO - PASS 7 of an N=256 OFFT, calc pwr spctrm
.endm

INITFFT -MACRO
*******************************************************************************
* Purpose: Decide which buffer to do OFFT on and set up Pointers etc...
* OFFT buffer MUST be at an address range compatible with BR addressing *
* 540 cycles *
*******************************************************************************
!!! Select the start address of the samples
bit PROGSTAT,15 ;Copy bit 0 to TC (Dictates where samples are)
.if PROCESSOR=0 ;Processor A ONLY (OFFT on buffer 0 or 2)
splk #SAM2STRT.bmar ;Speculative, set BMAR with SAM2STRT (PLP for xc)
xc 2,ntc ;Next line only if OFFT on SAM1STRT (tc=0)
splk #SAM3STRT.bmar ;Set BMAR with SAM3STRT (2 * nop if tc=1)
.elseif PROCESSOR=1 ;Processor B ONLY (OFFT on buffer 1 or 3)
splk #SAM1STRT.bmar ;Speculative, set BMAR with SAM1STRT (PLP for xc)
xc 2,ntc ;Next line only if OFFT on SAM1STRT (tc=0)
splk #SAM3STRT.bmar ;Set BMAR with SAM3STRT (2 * nop if tc=1)
.endif

!!! Copy the samples to the OFFT Buffer (leave copy of samples intact)
splk #0256,indx ;Set INDX to shuffle data into real locations 2PLP
lar ar5,#OFFFTSTRT ;AR5 points to start of OFFT buffer
larp ar5 ;ARP=5
rpt #0255 ;Repeat next instruction 256 times
bldd BMAR,*BRO+ ;Samples shuffled into OFFT buffer real locations

!!! Set pointers ready for OFFT
lar ar0,#(PASSCOUNT-3) ;Initialize - Passes per FFT Counter
lar ar1,#(GROUPCOUNT-1) ;Initialize - Groups per Pass Counter
splk #GROUPCOUNT,TEMP4 ;Backup group counter
lar ar2,#(BFLYCOUNT-1) ;Initialize - B'fly per Group Counter
lar ar3,#TWIDSTRT ;Initialize - Twiddle factor pointer
splk #TWIDSTRT,cbarl ;Backup twiddle block start address in CBSR1
lar ar4,#01h ;Initialize - Twiddle Block End Offset
lar ar5,#OFFFTSTRT ;Reset AR5 to start of OFFT buffer
lar ar6,#FINC ;Initialize - P-increment/Q-offset
.endm

FIRSTPASS -macro
*******************************************************************************
* Purpose: Radix-2 DIT Butterfly Macro for first pass in OFFT, 0.5 scaling*
* Assumes a real input (All imag inputs are zero) *
* PM = 00 - No shift (Product shift Mode), SXM set *
* AR3 points to Twiddles - At exit, pointer reset to start value *
* AR5 points to P-Data - At exit, pointer incremented by 4 (handed on) *
* AR7 points to Q-Data - At exit, pointer incremented by 4 (handed on) *
* INDX register used to aid P & Q-Pointer hand on *
*******************************************************************************
OFFT-BASED CLOCK RECOVERY

APPENDIX 9

NEXT_PASS .macro

Purpose: Update OFFT pointers and parameters for the beginning of next PASS

AR0 - Pass Counter decremented (DONE ELSEWHERE)
AR1 - Group Counter halved
AR2 - B’fly Counter doubled (Modified for BRCR format)
AR3 - Set to the start of the next block of Twiddles
AR4 - Twiddle Block End Offset increased. 2(AR4+1) - 1 - AR4
AR5 - SET ELSEWHERE, offset to its start address
AR6 - P-Increment/Q-Offset doubled
AR7 - SET ELSEWHERE, (at the beginning of the B’FLY Macro)
Circular Bufferl initialized to describe the next Twiddle Block
16 cycles (Executes 6 times)

Set Circular Bufferl to describe new Twiddle Block address

!! Group Counter has been destroyed, therefore it is restored from memory

!! If B’fly Counter is derived from P-Increment, then there is no need
!! to save and restore B’fly Counter.

lamm ar6
sub #01h
samm ar2
lacc ar6,1
sacl ar6

!! Old value of AR6 - 1 is the next Twiddle Block

sub #01h
decrement ACC
samm ar4
Twiddle Block End Offset = new AR6 - 1

!! Set Circular Bufferl to describe new Twiddle Block address

lamm ar2
add #01h
add ar3
samm cbsr1

!! Convert to BANZ loop counter (sub #01 from ACCh)
APPENDIX 9
OFFT-BASED CLOCK RECOVERY

sach ar1 ;Group Counter updated
*! Circular Buffer, AR5 should already be reset to its start address
*! AR7 doesn't matter as it is set at the beginning of each B'fly Loop
.endm

BFLY .macro OPTION
*****************************************************************************
* Purpose; Radix-2 DIT Butterfly Macro for OFFT, scales by 0.5
* PM= 00 - No shift (Product shift Mode), SXM set
* AR2 controls the number of B'flies per group (In BRCR format - count-1)
* AR3 points to Twiddles - At exit, pointer incremented by 2 (handed on)
* AR5 points to P-Data - At exit, pointer incremented by 2 (handed on)
* AR7 points to Q-Data - At exit, pointer incremented by 2 (handed on)
* LAST MODIFIED 19/01/96 - Optimised
* If program in DARAM & Data in SARAM;
* PASS 3 | 24 + [32 * (10 + 4*20)] + 6 cycles
* PASS 4 | 24 + [16 * (10 + 8*20)] + 6 cycles
* PASS 5 | 24 + [ 8 * (10 + 16*20)] + 6 cycles
* PASS 6 | 24 + [ 4 * (10 + 32*20)] + 6 cycles
* PASS 7 | 24 + [ 2 * (10 + 64*20)] + 6 cycles
* 16800 cycles
*****************************************************************************

PASS_START?
.if OPTION=O
NEXT_PASS
.endif
lacl cl
sac l
add cl ar5
sacl ar7
GROUP_START?
la mr brcr
samm brcr
rptb
BFLY_LOOPEND?-1
lt *+,ar3
mpy *+,ar7
ltl *+,ar3
mpy **,ar5
mpy ***,ar2
mpy ***,ar7
mpy +++,ar
mpy +++,ar5
mpy +++,ar7
sach TEMP1
ltp +++,ar3
mpy +++,ar
mpy +++,ar5
mpy +++,ar7
sach TEMP2
lacc +++,ar5
add TEMP2,15
sach +++,ar7
sub TEMP2,16
sach +++,ar5
add TEMP1,1
sach +++,ar7
sub TEMP1,16
sach +++,ar5
add TEMP1,15
sach +++,ar7
BFLY_LOOPEND?
.ml - Repeat Block End address
.mar *O+,ar1
.mar #0FFTSTRT
.endm

A9-20
LASTPASS .macro

* Purpose: Radix-2 DIT Butterfly Macro for last pass in OFFT, 0.5 scaling
* Calculates Power Spectrum (scales by 2) stores them in imaj locations
* Also finds the largest powers in each half of the spectrum for later use
* AR1 points to Power Spectrum P entries
* AR2 points to Power Spectrum Q entries
* AR3 points to Twiddles -
* AR5 points to P-Data - At exit, reset
* AR7 points to Q-Data -
* INDX must be equal to 2!!
* If program in DARAM & Buffers in SARAM; 20 + 128*35 = 4500 cycles
* Either 15 or 17 extra cycles per subroutine call, say (16*16 + 16*18)

***************************************************************************
* Prepare pointers for the last pass
  larp ar3 ;Select Twiddle Pointer
  adrk #(FFTLEN/2) ;Set it to point to final Twiddle Block
  splk #02,indx ;Set INDX register for pointer increments of 2
  lar ar1,#POWERSTRT ;Set ARl to the start of POWER buffer
  lar ar2,#(POWERSTRT+(FFTLEN/2)) ;Set AR2 to the middle of POWER buffer
  lar ar5,#0FFTSTRT ;Reset AR5 to start of OFFT buffer
  lar ar7,#0FFTSTRT+256;Reset AR7 to middle of OFFT buffer
  larp ar7 ;Select Q- Pointer

***************************************************************************
*!! Calculate the power spectrum only in ZONE 1
*!! Prepare locations that hold peak powers so they are forced to be overwritten
*!! This in turn will force the addresses to be overwritten
*!! Only done so that the power spectrum can be displayed by DAC1
*!! call BFLYPOWER0 ;SBRT - Perform Butterflies & calculate Power (33)

***************************************************************************
*!! Set locations that hold peak powers so they are forced to be overwritten
*!! This in turn will force the addresses to be overwritten
  lacc #0fffh ;ACC = -1
  sac1 HIVAL1 ;Reset all value locations to -1
  sac1 SEVAL1 ;Address don't need to be reset as they are
  sac1 HIVAL2 ;guaranteed to be overwritten during the
  sac1 SEVAL2 ;execution of this macro

***************************************************************************
*!! Calculate the power spectrum in ZONE 2 & search for the power peaks in
*!! bins 33 to 95 and in bins 161 to 223
  splk #052,brccr ;Repeat 63 times
  rptb LP_LOOP?7-1 ;Start of Repeat Block
  lt **,ar3 ;QR " TREG
  mpy **,ar7 ;QR*WR/2 " PREGh
  ltp **,ar3 ;QI " TREG, QR*WR/2 " ACCh
  mpy **,ar7 ;QI*WI/2 " PREGh
  mpya **,ar7 ;QI*WR/2 " PREGh, [QR*WR + QI*WI]/2 " ACCh
  sach TEMP1 ;[QR*WR + QI*WI]/2 " TEMP1
  ltp **,ar3 ;QR " TREG, QI*WR/2 " PREGh
  mpy **,ar5 ;QI*WI/2 " PREGh, Hand on Twiddle-Pointer
  spac ;[QI*WR - QR*WI]/2 " ACCh
  sach TEMP2 ;[QI*WR - QR*WI]/2 " TEMP2
  lacc *,14,ar5 ;PR/4 " ACCh
  add TEMP1,15 ;[PR + (QR*WR + QI*WI)]/4 " ACCh
  sach *,1,ar7 ;[PR + (QR*WR + QI*WI)]/2 " PR location
  sub TEMP1,16 ;[PR - (QR*WR + QI*WI)]/4 " ACCh
  sach *,1,ar5 ;[PR - (QR*WR + QI*WI)]/2 " QR location
  lacc *,14,ar5 ;PI/4 " ACCh
  add TEMP2,15 ;[PI + (QI*WR - QR*WI)]/4 " ACCh
  sach *,1,ar7 ;[PI + (QI*WR - QR*WI)]/2 " PI location
  sub TEMP2,16 ;[PI - (QI*WR - QR*WI)]/4 " ACCh
  spc **,ar7 ;[PI - (QI*WR - QR*WI)]/2 " PI location
  zap ;ACC = PREG = 0
  sqra **,- ;QI*2 " PREG, 0 " ACCh
  sqra *,0,ar5 ;QR*2 " PREG, QI*2 " ACCh, hand on Q-Pointer

A9-21
This page contains a portion of a program in assembly language. The program is involved in calculating the power spectrum of a signal. Here’s a breakdown of the main sections:

### OFFT-BASED CLOCK RECOVERY

- **Calculate the power spectrum only in ZONE 3**: This instruction limits the calculation to specific zones of the signal.

- **In other words calculate the power in Bins 96 to 127 and bins 224 to 255**: The power spectrum is calculated for these specific bins to ensure accurate data representation.

- **Repeat Block End address**: This marks the end of the loop or repeat block, indicating where the process restarts for the next iteration.

### SUBROUTINE FOR LAST_PASS MACRO

- **Performs the last pass butterflies & calculates the power spectrum**: This subroutine is designed to handle the final calculations of the power spectrum.

- **Does not search for the peak powers**: This subroutine focuses on calculating the power spectrum without identifying peak values.

- **Two entry points provided**: Two different entry points are available for this subroutine, offering flexibility in its usage.

### Assembly Language Code Snippets

Here’s a sample of the code snippets from the program:

- **lt**: This command is used for loading the temporary register.
- **mpl**: This command is used for multiplying the values stored in the registers.
- **add**: This command is used for adding the values stored in the registers.
- **sub**: This command is used for subtracting the values stored in the registers.
- ** zap**: This command is used for clearing the accumulator.

These commands are part of the overall process of calculating the power spectrum, involving operations like addition, subtraction, and multiplication, which are crucial for accurate signal processing.
* SUBROUTINE FOR LAST_PASS MACRO
* Finds two highest values and addresses in the 2nd half of power spectrum
* Results stored in page 0 locations HIVAL2, HIADD2, SEVAL2 & SEADD2
* If highest value replaced, 18 cycles - including 2 from call
* If 2nd highest value replaced, 16 cycles - including 2 from call

SWAP2
add SEVAL2,15 ;(QI^2 + Q2^2) - ACCh
sacb ;(QI^2 + Q2^2) - ACCBh
sub HIVAL2,15 ;Compare highest value to current value
bcnd LP21,1t ;If greater, current value becomes highest value

LP20
lacl HIVAL2 ;Highest value - ACC
sacl SEVAL2 ;Second highest value := old highest value
lacl HIADD2 ;Highest address - ACC
sacl SEADD2 ;Second highest address := old highest address
lacl ar2 ;Current address +1 - ACC
sub #01h ;Current address - ACC
sacl HIADD2 ;Current address := highest address
retd ;Delayed return from subroutine
lacb ;Current value/2 - ACC
sach HIVAL2,1 ;Current value := current value
LP21
lacl ar2 ;Current address +1 - ACC
sub #01h ;Current address - ACC
sacl SEADD2 ;Current address := second highest address
retd ;Delayed return from subroutine
lacb ;Current value/2 - ACC
sach SEVAL2,1 ;Current value replaces second highest value

* SUBROUTINE FOR LAST_PASS MACRO
* Finds two highest values and addresses in the 1st half of power spectrum
* Results stored in page 0 locations HIVAL1, HIADD1, SEVAL1 & SEADD1
* If highest value replaced, 18 cycles - including 2 from call
* If 2nd highest value replaced, 16 cycles - including 2 from call

SWAP1
add SEVAL1,15 ;(PI^2 + PR^2) - ACCh
sacb ;(PI^2 + PR^2) - ACCBh
sub HIVAL1,15 ;Compare highest value to current value
bcnd LP11,1t ;If greater, current value becomes highest value

LP10
lacl HIVAL1 ;Highest value - ACC
sacl SEVAL1 ;Second highest value := old highest value
lacl HIADD1 ;Highest address - ACC
sacl SEADD1 ;Second highest address := old highest address
lacl ar1 ;Current address +1 - ACC
sub #01h ;Current address - ACC
sacl HIADD1 ;Current address := highest address
retd ;Delayed return from subroutine
lacb ;Current value/2 - ACC
sach HIVAL1,1 ;Highest value := current value
LP11
lacl ar1 ;Current address +1 - ACC
sub #01h ;Current address - ACC
sacl SEADD1 ;Current address := second highest address
retd ;Delayed return from subroutine
lacb ;Current value/2 - ACC
sach SEVAL1,1 ;Current value replaces second highest value

LPEND .endm
Appendix 10

Dual-processor DSP board

This appendix contains hardware information relating to the dual-processor DSP development platform, described in Chapter 4.

The hardware consist of an 8-layer, extended Eurocard printed circuit board (11 x 8.975ins), containing two TMS320C50 fixed-point digital signal processors, jointly providing more than 80 MIPS (million instructions per second). The board has been developed as a general-purpose platform for the implementation of computationally
intensive DSP algorithms, with particular consideration to the implementation of an OFFT. Each processor provides independent processing channels and can pass information between each other, either via a high speed synchronous serial port or via 2K x 16 bit words of dual-port Global RAM. Each channel has 32K x 16 of EPROM and 32K x 16 of expansion memory, for both programme space and data space. This is in addition to the processors 9K of on chip memory.

Channel-A contains a high-speed 12 bit analogue input and output, capable of operating at speeds of up to 500K samples/second. To reduce interrupt latency, channel-A’s ADC (Analogue to Digital Converter) and DAC (Digital to Analogue Converter) are buffered with 1K deep FIFOs (First in First Out memory). Hence, channel-A has been designed for block processing such as an FFT or OFFT.

For continuous block processing, channel-B can also gain access to channel-A’s input FIFO. Hence, an OFFT can be processed by one channel, while the other channel is processing input samples from the input FIFO and vicer versa, thus operating in a change over mode, such that no input samples are lost.

Channel-B also has a high speed 12 bit ADC, but has no buffering since it is intended for sample rate processing (such as filters). Channel-B contains a quad 12 bit DAC providing four independent analogue outputs. These outputs are useful for displaying on an oscilloscope, intermediate test points within the software algorithms. Both channels provide independent address decoding, digital IO with handshakes (made available at ribbon cable connectors), status LEDs, status DIL switches and UARTs (Universal Asynchronous Receiver Transmitter) for communicating with a remote dumb-terminal, via an RS232 interface.
APPENDIX 10

DUAL-PROCESSOR DSP BOARD

A BIOS (Basic Input and Output System) is programmed into the onboard EPROMs, to allow a terminal to remotely control DSP software execution and also to display debug information. A programmable timer, containing 3 independent timers, controls the sample rates. Reset logic provides a watchdog timer, which resets the system in the event of a programme crash. A standard IEEE 1149.1 (JTAG) emulator interface is also provided to allow in-circuit software debugging. Both channels have full bus expansion, brought out to ribbon cable connectors and the back-plane connectors, allowing memory-mapped expansion.

The board has been designed in modular form, such that the processing power can be increased to more than 320MIPS, by plugging in additional cards. A maximum of 8 processors can communicates with each other via a synchronous TDM (Time Division Multiplexing) serial port.

An integrated PSU (power supply unit) has also been designed, which provides +5V regulated at 3 amps (digital supply) and ±18V linear unregulated at 1 amp (regulated on board to ±12V) for the analogue supply. The board plugs into the PSU back-plane connector and is protected via a transparent cover, which hinges open for accessibility to the card allowing measurements to be made.

DSP code is written in assembly language on a text editor, which is assembled, linked (with a link command file defining the hardware memory map) and then converted to Intel Hex, for programming the onboard EPROMs. Time critical DSP code is downloaded from EPROM to RAM, which is then executed at optimum processor speed (i.e. with zero wait-states).
Figure A10.2 Dual-Processor DSP Board - Block Diagram
Figure A10-3a Dual-processor DSP board - Schematic diagram (Part a)
Figure A10-4 Dual-processor DSP board - Assembly diagram
Function of onboard links

Processor-A Digital IO

<table>
<thead>
<tr>
<th>LK1</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Port-A OUT</td>
<td>Configures C50 Data Bits 8 to 15 as outputs</td>
</tr>
<tr>
<td>b</td>
<td>Port-A IN</td>
<td>Configures C50 Data Bits 8 to 15 as inputs</td>
</tr>
<tr>
<td>c</td>
<td>Port-B OUT</td>
<td>Configures C50 Data Bits 0 to 7 as outputs</td>
</tr>
<tr>
<td>d</td>
<td>Port-B IN</td>
<td>Configures C50 Data Bits 0 to 7 as inputs</td>
</tr>
<tr>
<td>e</td>
<td>Sync IN</td>
<td>Strobe from peripheral to clock data into input latch</td>
</tr>
<tr>
<td>f</td>
<td>Async IN</td>
<td>C50 asynchronously reads data bits on input Port</td>
</tr>
<tr>
<td>g</td>
<td>GP OUT</td>
<td>Output strobe for clock output data into peripheral</td>
</tr>
<tr>
<td>h</td>
<td>Rdy</td>
<td>Flag to inform peripheral that data is ready to read</td>
</tr>
</tbody>
</table>

Processor-B Digital IO

<table>
<thead>
<tr>
<th>LK6</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Port-A OUT</td>
<td>Configures C50 Data Bits 8 to 15 as outputs</td>
</tr>
<tr>
<td>b</td>
<td>Port-A IN</td>
<td>Configures C50 Data Bits 8 to 15 as inputs</td>
</tr>
<tr>
<td>c</td>
<td>Port-B OUT</td>
<td>Configures C50 Data Bits 0 to 7 as outputs</td>
</tr>
<tr>
<td>d</td>
<td>Port-B IN</td>
<td>Configures C50 Data Bits 0 to 7 as inputs</td>
</tr>
<tr>
<td>e</td>
<td>Sync IN</td>
<td>Strobe from peripheral to clock data into input latch</td>
</tr>
<tr>
<td>f</td>
<td>Async IN</td>
<td>C50 asynchronously reads data bits on input Port</td>
</tr>
<tr>
<td>g</td>
<td>GP OUT</td>
<td>Output strobe for clock output data into peripheral</td>
</tr>
<tr>
<td>h</td>
<td>Rdy</td>
<td>Flag to inform peripheral that data is ready to read</td>
</tr>
</tbody>
</table>

Link 2A (a–d) Watchdog Timer
Link 2B (e–h) Output FIFO Control

<table>
<thead>
<tr>
<th>LK2</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>CLKO OUT 2</td>
<td>Disable watchdog reset</td>
</tr>
<tr>
<td>b</td>
<td>XFA</td>
<td>Processor-A watchdog reset</td>
</tr>
<tr>
<td>c</td>
<td>XFB</td>
<td>Processor-B watchdog reset</td>
</tr>
<tr>
<td>d</td>
<td>F/F</td>
<td>Processor A and B watchdog reset, toggles a flip flop</td>
</tr>
<tr>
<td>e</td>
<td>SmplDAC1</td>
<td>Clock DAC-1 on a single word in the output FIFO</td>
</tr>
<tr>
<td>f</td>
<td>BurstDAC1</td>
<td>Clock DAC-1 on a burst of words in the output FIFO</td>
</tr>
<tr>
<td>g</td>
<td>HF Out</td>
<td>Burst on 512 words in the output FIFO</td>
</tr>
<tr>
<td>h</td>
<td>FF Out</td>
<td>Burst on 1024 words in the output FIFO</td>
</tr>
</tbody>
</table>
Processor-A Interrupt-I Control

<table>
<thead>
<tr>
<th>LK3</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>EF In</td>
<td>Interrupt Processor-I on one word in the input FIFO</td>
</tr>
<tr>
<td>b</td>
<td>HF In</td>
<td>Interrupt Processor-I on 512 words in the input FIFO</td>
</tr>
<tr>
<td>c</td>
<td>FF In</td>
<td>Interrupt Processor-I on 1024 words in the input FIFO</td>
</tr>
</tbody>
</table>

Link 4A (a–c) ADC-1 Start of Conversion.
Link 4B (d–g) ADC-2 Start of Conversion

<table>
<thead>
<tr>
<th>LK4</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>SWSOC-A</td>
<td>Processor-A Software start of conversion to ADC-1</td>
</tr>
<tr>
<td>b</td>
<td>TIM-A</td>
<td>Processor-A internal timer: start of conversion to ADC-1</td>
</tr>
<tr>
<td>c</td>
<td>SMPL1</td>
<td>Counter Output 0: start of conversion to ADC-1</td>
</tr>
<tr>
<td>d</td>
<td>SWSOC-B</td>
<td>Processor-B Software start of conversion to ADC-2</td>
</tr>
<tr>
<td>e</td>
<td>TIM-B</td>
<td>Processor-B internal timer: start of conversion to ADC-2</td>
</tr>
<tr>
<td>f</td>
<td>SMPL2</td>
<td>Counter Output 1: start of conversion to ADC-2</td>
</tr>
<tr>
<td>g</td>
<td>A-SOC</td>
<td>ADC-1 start of conversion also starts ADC-2</td>
</tr>
</tbody>
</table>

Clock Divider

<table>
<thead>
<tr>
<th>LK5</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2M5</td>
<td>2.5MHz Clock to Counter and ADCs</td>
</tr>
<tr>
<td>b</td>
<td>5M</td>
<td>5MHz Clock to Counter and ADCs</td>
</tr>
<tr>
<td>c</td>
<td>10M</td>
<td>10Hz Clock to Counter and ADCs</td>
</tr>
<tr>
<td>d</td>
<td>20M</td>
<td>20MHz Clock to Counter and ADCs</td>
</tr>
</tbody>
</table>

Switches

<table>
<thead>
<tr>
<th>No.</th>
<th>A-Status</th>
<th>B-Status</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Off</td>
<td>Off</td>
<td>Disable Acquisition</td>
</tr>
<tr>
<td>2</td>
<td>Off</td>
<td>Off</td>
<td>Override Rx signal with stored version</td>
</tr>
<tr>
<td>3</td>
<td>Off</td>
<td>On</td>
<td>Display matched filter output on A-Out (J3)</td>
</tr>
<tr>
<td>4</td>
<td>Off</td>
<td>On</td>
<td>Display recovered clock filter output on A-Out (J3)</td>
</tr>
<tr>
<td>5–8</td>
<td>On</td>
<td>On</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Default all switches on.
## Author’s Published Papers

<table>
<thead>
<tr>
<th>No.</th>
<th>Title</th>
<th>Authors</th>
<th>Conference/Event</th>
<th>Date</th>
<th>Location</th>
<th>ISBN</th>
</tr>
</thead>
</table>
DSP-Based Clock Recovery for a Digital Magnetic Data Channel

P.M. Smithson, M. Tomlinson and T. Donnelly

Terry Donnelly, The Centre for Research in Information Storage Technology, SECEE, University of Plymouth, England, U.K.

ABSTRACT - A DSP-based clock recovery system for a digital magnetic recording channel is presented which utilises an Infinite Impulse Response (IIR) filter. Performance has been evaluated using software simulation, verified in real-time operating on a Texas Instruments TMS320C25 processor. Data coded in Manchester, Miller, Miller and 2,7 codes have been applied. In each case synchronization is achieved within the first symbol period even in the presence of jitter. The system can withstand data dropouts of 40 bit periods without losing synchronization. This can be increased when filter sections are cascaded.

I. INTRODUCTION

In data recording systems it is necessary to derive a clock which is synchronized to the retrieved data. This clock is normally extracted from the data. The modulation coding scheme used to store data is chosen to have a strong spectral component at the clock rate required. Such codes are termed "self clocking" [1-2]. Current systems usually employ a Phase Locked Loop (PLL) to recover the clock signal from the data [3]. However, loss of lock and synchronization can occur if the data synchronizing the PLL "drops out" for a period of time. Problems can also occur in regaining lock if the data signal is poor. The "dropout", together with poorer acquisition time, results in data being lost. Also, in order to ensure the PLL is in a locked condition before data are retrieved a long synchronizing sequence must be applied before data can be accurately clocked. The synchronizing sequence must be stored on the recording medium and thus occupies space which could be used for data. Digital technology has now advanced to a point where clock recovery can be realised using a bandpass digital recursive infinite impulse response filter (IIR). Because of the feedback employed, a characteristic of this form of filter is the tendency for it to become unstable and "ring". In the context of clock recovery this feature is desirable since it can be exploited to provide a good flywheel effect over periods of lost input, i.e. dropouts.

Results are presented of clock recovery from coded data using an IIR filter. Several magnetic recording codes are investigated: Manchester, Miller, Miller and 2,7 code. The results were obtained using Alta Group of Cadence Design Systems (formerly Comdisco), Signal Processing Workstation (SPW) simulation software run on a Sparc Workstation and verified in real-time using a Texas Instruments TMS320C25 fixed point digital signal processor.

II. THEORY

The clock recovery system and associated waveforms are shown in figure 1.

![Clock Recovery System with Waveforms](image)

Fig 1. Clock Recovery system with waveforms

Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible stimulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at its tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E, the recovered clock output.

When the d constraint of RLL codes [1] limits the frequency of symbol transitions (as in Miller, Miller and 2,7 codes), the stimulation to the IIR filter can be increased by implementing a two stage pre-processor. The first T/2 stage (T = symbol period) edge detects the symbol transitions and the second T/4 stage doubles the frequency for application to the IIR filter. An IIR filter is normally formed as a second order section containing both poles and zeros, this provides a centre frequency and stop-band nulls. In this application the stop-band null is of no consequence and the design can be simplified to have feedback paths only.
The architecture of the IIR filter is shown in Figure 2.

![Fig 2. Architecture of the Clock Recovery IIR Filter](image)

The filter output can be expressed in the time domain as,

\[ y(n) = x(n) + Cy(n-1) - Ky(n-2) \]  

(1)

Taking the Z transform of (1) it can be shown [4] that,

\[ H(z) = \frac{Y(z)}{X(z)} = \frac{z^2}{z^2 - Cz + K} \]  

(2)

Equation (2) is the z-domain transfer function of the filter which contains poles and a double zero at the z-plane origin.

\[ H(z) = \frac{1}{i - Cz^{-1} + Ke^{-i2\pi T/N}} \]  

(3)

Substituting \( e^{j\omega T} = e^{j2\pi f_s/Nf_s} \) for \( z \):

\[ h_n = \left(1 - Ce^{-j2\pi n/N} + Ke^{-j4\pi n/N}\right)^{-1} \]  

(4)

where \( N = 1024 \), \( n = 0, 1...N-1 \) and \( j = 0, 1...100 \) the theoretical normalized frequency response of the filter is plotted as in Figure 3.

![Fig 3. Frequency Response of the IIR Filter](image)

The poles of (2) describe the resonance and stability of the filter:

\[ z^2 - Cz + K = 0 \]  

(5)

Assuming complex poles, i.e. resonance:

\[ z = \frac{C}{2} \pm \frac{1}{2} \sqrt{4K - C^2} \]  

(6)

From (6) and analysis of the z-plane [4], \( |K| < 1 \) for a stable system but should be close to unity for a high Q.

The second filter coefficient, \( C \), can be determined from,

\[ C = \sqrt{\frac{4K}{\tan^{-1}\frac{2\pi f_s}{f_s} + 1}} \]  

(7)

Coefficient \( K \), as previously stated, is chosen to be close to unity.

Values selected for simulation are:
\( K = 255/256 = 0.99609375 \), and \( C = 1.84414 \).

A sample rate is then selected to tune the filter to the required resonant frequency:

\[ f_s = \frac{2\pi f_0}{\tan^{-1}\frac{2\pi f_s}{f_s} + 1} \]  

(8)

where \( f_0 \) is the clock rate required and \( f_s \) is the sample frequency: 32 samples / bit period was used.

The retrieved data may contain timing jitter due to noise and amplitude variations of the received signal. The IIR filter averages these timing errors, minimizing jitter in the recovered clock. To reduce sampling errors, the recovered clock is delayed so that the sampling edge retimes the data at the centre of the bit period. Jitter of \( \pm \frac{1}{2} \) a bit period can then be tolerated before data errors occur.

![Fig 4. Data Sampling](image)
III. SIMULATION RESULTS

Clock recovery performance was determined for four recording codes: Manchester, Miller, Miller¹ and 2,7 code, the results are shown in figures 5 to 7. In each case the acquisition time was measured between applying data to achieving synchronization. Also, in each case the response to a 40-bit dropout was determined.

After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronization is achieved immediately as the output is in phase with the data. At 32 samples per bit period this acquisition time represents an interval of 1/32 of one period where the clock frequency is twice the symbol interval. This short acquisition time is the same for each of the codes. In each case the clock signal is in phase with the data after the dropout. This represents a flywheel performance of 80 clock periods.

Figure 5a shows simulation results of clock acquisition when applied to Manchester encoded data. A single T/2 re-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing.

Figure 5b shows the flywheel performance over a 40-bit data dropout. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock. Zooming in to the end of the dropout, shows the recovered clock remaining in phase when data are re-applied.

Figure 6a shows simulation results of clock acquisition when applied to RLL 2,7 encoded data. A two stage pre-processor is applied, the first T/2 stage edge detects the symbol transitions and the second T/4 stage doubles the frequency and scales the simulation to the IIR filter.

---

¹ Miller code refers to a specific type of signal coding used in digital communications. In this context, it seems to be a variant of the Miller code, possibly modified or extended for specific applications.
IV. IMPLEMENTATION

Hardware and software have been developed utilizing a Texas Instruments TMS320C25 fixed point digital signal processor. Clock recovery at nine data rates ranging from 50 to 4800 bits/sec have been programmed into common hardware with selectable sample rates of 6.4KHz, 38.4KHz and 76.8KHz. A two stage cascaded IIR filter was used running with zero wait states from the processors on-chip memory. In all cases the flywheel time of the recovered clock was greater than 46 NRZ data symbols. In practice, quantisation errors occur when implementing coefficients in a 16-bit fixed point processor. These are minimised in the TMS320C25 by utilising the processor's 32 bit multiply and accumulate registers. The high and low accumulators represent the integer and fractional part of the floating point numbers.

The TMS320C25 processor provides 10 MIPS (million instructions per second), the achievable clock recovery rate can be increased proportionally with faster processors but at a higher device cost.

To accommodate clock recovery at high data rates, a parallel logic implementation must be applied either in discrete logic or an ASIC (Application Specific Integrated Circuit). To maintain programmability the clock recovery filter has been implemented in a Xilinx 3020PC68-50 (2000 gates, 50MHz) field programmable gate array, where the filter characteristics are programmed into the gate array at power-up or on-the-fly. Binary weighted coefficients and adders are employed instead of the more conventional binary multipliers, representing approximately a 50% increase in speed performance and a 50% reduction in silicon utilisation [5].

The filter has a 16 bit dynamic range and utilises 61 of 64 CLBs (Configurable Logic Blocks).

Figure 8 shows a block diagram of the filter implementation within the programmable gate array.

Figure 8. Programmable Gate Array Filter Architecture

The sample rate used was 2.59MHz providing clock recovery at a data rate of 100Kbits/sec. These values where chosen to satisfy a specific application in a satellite communication link. With the use of larger, faster devices and employing adder look ahead carry techniques, much higher data rates are achievable.
V. CONCLUSIONS

The results show superior performance in terms of acquisition time compared to a conventional Phase Locked Loop (PLL). Whereas the PLL requires a long, lead-in of synchronizing pulses before a clock synchronizes to data, with the IIR filter lock recovery system synchronization is virtually instantaneous. This feature would significantly reduce the need to store synchronizing data thus increasing the utilisation of the recording medium.

The ability to withstand long dropouts is good but PLL performance may be better in this respect if a long time constant is employed. However, in improving this aspect of a PLL the acquisition time is made worse. The ability of the IIR lock recovery system to withstand long dropouts can be improved by cascading filters, each additional filter giving a further 80 clock periods resistance. Also, the IIR filter clock recovery system can be combined with a PLL to capitalise on the benefits of both techniques.

The bandwidth of the filter controls the acceptable data rate drift, which for the simulation represents 1% tolerance. Larger tolerances can be accommodated but at a reduction in the flywheel period over dropouts. The IIR filter can be further modified with an AGC (Automatic Gain Control) in the feedback path to maximise the dynamic range of the filter output and hence the flywheel period. This will also compensate for small variations in data rate drift.

Current research involves the implementation of digital filters in VLSI employing multiplier-less coefficients. Further investigation into clock recovery at higher data rates is being pursued through the implementation of digital filters in VLSI.

VI. REFERENCES


The Satellite Communication Research Centre and the Centre for Research in Information Storage Technology, are research centres at the School of Electronic, Communication and Electrical Engineering, University of Plymouth, Drake Circus, Plymouth, Devon, United Kingdom, PL4 8AA.
A PRACTICAL APPROACH TO DIGITAL SIGNAL PROCESSING

E C Ifeachor and P M Smithson

Digital signal processing (DSP) now plays an important role in science and engineering because of the widespread use of computers to process, manipulate and store data. This makes it vital that engineering students have a good understanding of the fundamentals of the subject and a practical appreciation of the possibilities it offers. However, DSP is mathematically demanding [1,2] and this makes it difficult for students to follow the subject. This paper discusses a practical approach we have adopted in Plymouth to the teaching of DSP [3]. This has proved effective in giving the student a perceptive understanding of the fundamentals of DSP and a good appreciation of the power and versatility of DSP processors.

An important part of our teaching is a set of stand-alone DSP hardware, developed in-house, which have provided useful and inexpensive platforms for demonstrating simple DSP algorithms in real-time and to support final year degree project work. Case studies are used to explore real world problems and the interactions between DSP and other technologies, such as artificial neural networks and satellite communications.

DSP in Plymouth

DSP is a very wide subject and as such can only be taught selectively. In Plymouth, the bulk of DSP is taught in the final year of our B.Eng degree programmes. On the B.Eng Electrical & Electronic Engineering programme, the DSP course involves an in-depth study of analogue Input/Output design techniques (including sampling, quantisation, anti-aliasing/imaging filtering and oversampling techniques), discrete transforms (especially the Discrete Fourier Transform and the Fast Fourier Transform) and digital filter design, plus a selection of special topics (e.g. adaptive filters, speech processing and multi-rate processing). All the topics are illustrated with application examples to which the student can easily relate, these include biomedical engineering, digital audio and communication engineering.

Students on the Communication Engineering degree programme cover broadly similar topics, but there is more emphasis on computer simulation using SPW (Signal Processing Workstation). Assignments currently set on SPW include modulation schemes (e.g. BPSK and QPSK), channel filtering (the raised cosine filter and its effect on Inter Symbol Interference), Bit Error Rate measurements with additive noise, clock recovery and error control coding. In the near future, SPW will be covered in the second year, which is common to all engineering degree students within the School. A prerequisite for DSP in the final year, is a second year course which covers the basic concepts of signal processing and coding.

As a complement to lectures and seminars, a series of laboratory experiments are provided to enable students to gain hands-on experience and an appreciation of various signal processing techniques. An example is the Introduction to digital signal processing, which covers the principles and applications of correlation and FFT algorithms, including the detection of signals buried in noise, distance measurements, spectrum and frequency response estimation and fast convolution. Another laboratory topic is the Introduction to digital filtering which requires the students to investigate methods of calculating filter coefficients and how to implement simple digital filters in real-time. These laboratory sessions make students aware of the practical implications of real-time signal processing, such as the effects of finite wordlength.

School of Electronic, Communication and Electrical Engineering, University of Plymouth, Plymouth, UK, PL4 8AA.
A demonstration and a hands-on walk through of real-time DSP using fixed point processors (TMS320C10 and TMS320C25) are provided by the lecturer and technical staff. This covers various aspects such as, real time digital filtering, signal generation, clock recovery, the effects of finite wordlength and sampling (including \( \sin x/x \) and aliasing). Each laboratory class consists of about 14 students, working in pairs, all carrying out the same experiment. This makes it possible to schedule laboratory sessions to coincide with the subject matter of the relevant lectures. This would have been otherwise difficult if different experiments were carried out at the same time.

The assessment of laboratory work is based upon attendance and satisfactory progress, as recorded in the laboratory log. The laboratory sheet, with results and answers to tutorial questions constitute the laboratory log book. Two multiple choice computer-based tests are accessed on laboratory work and related topics, one in January and the other in March. Spaces are provided in the laboratory sheets for student to record their results. Searching tutorial questions are given, with spaces for answers, this motivates students to reflect more deeply on the outcome of their laboratory work.

**DSP Tools**

An important part of our teaching is a set of stand-alone DSP hardware, developed in house, which have provided useful and inexpensive platforms for demonstrating simple DSP algorithms in real-time and have enabled students to design and implement their own DSP algorithms. Unlike most commercially available DSP development hardware, usually inside a PC, our students have access to the hardware and can easily make measurements with an oscilloscope, logic or spectrum analyser. The hardware tools support both the laboratory work and individual student projects in the final year. In our experience, a typical student project is under funded and does not allow sufficient time to develop both hardware and software. Proven reusable DSP hardware, leaves the student more time to concentrate on DSP algorithms and their application.

The in-house hardware is a set of TMS320C10 and TMS320C25 boards. The TMS320C25 board (see Figures 1 and 2), with dual 12-bit ADCs and DACs, can interface to a PC or a remote terminal and together with commercial tools such as the Texas Instrument Software Development System (SWDS) form a powerful development platform.

![Figure 1 Block diagram of the TMS320C25 development system](image)

The SWDS allows full debugging capabilities running in real-time on our custom hardware as a target system. This includes single stepping through disassembled code, the introduction of breakpoints and the manipulation of memory and the processors internal registers. When the software has been fully developed and proven in real-time, the SWDS is removed and replaced with programmed EPROMs to produce a stand-alone DSP system.
We utilise a mix of commercial DSP cards to support DSP tasks requiring floating point operations and advanced DSP software tools. SPW is used extensively by our communication engineering students for simulation and analysis. Other DSP tools available include commercial hardware tools (TMS320C30 cards) and software tools (e.g. ILS, QE design, MISA). In addition to commercial software, in-house DSP software [3] is also provided. Assembly language programs and full circuit diagrams of our custom hardware are made available to the students.

A number of DSP projects have originated from industry or from Research Groups within the University, notably in biomedicine, communication engineering, digital audio and information storage technology. Examples of recent projects are real-time digital audio signal processing, adaptive cancellation of mains transformer noise, single-bit ADCs/DACs and fetal heart rate variability.

Case Studies

Case studies involving the design and application of DSP, form an important part of our DSP course. Our own research work in applied DSP has inspired some of the examples. These include the detection of fetal heart beats in fetal electrocardiogram (ECG) using artificial neural networks, with the input data pre-processed using digital filtering and peak detection algorithms. Another is the implementation of clock recovery in a satellite data link using DSP techniques.

We see case studies as an excellent way of getting our students to engage in extended work on real world problems. A piece of work is spread over one or two semesters which provides the opportunity to apply what has been taught and to seek new information from research papers. Two examples of our case studies are described below.
Case Study 1 - Detection of fetal heart beats from noisy electrocardiogram

Background

In fetal monitoring during labour, it is often required to analyse the changes in fetal heart rate (FHR) patterns to assess the fetal well being. A common practice is to measure the FHR from the electrical activity of the baby's heart, known as the electrocardiogram (ECG), see Figure 3. The reciprocals of successive R-to-R peaks, suitably scaled, give the heart rate pattern in beats per minute. In modern obstetrics, changes in both the heart rate pattern and in the ECG waveform itself are important in detecting fetal distress.

In the ECG waveform the most prominent part, labelled QRS, has frequency components between about 5 and 50Hz. Accurate measurement of the FHR depends on the reliable detection of the QRS complexes. Signal degradation due to baseline wander, mains interference, uterine contractions, ADC saturation, and movements of the baby or mother will lead to false detection or missed QRS complexes.

![Figure 3 An illustration of the fetal electrocardiogram](image)

\[
\text{Heart rate (beats min}^{-1}) = \frac{1}{\text{heart period (ms)}} \times 60000
\]

Task

The task is to design and implement a reliable QRS complex detection algorithm based on the multilayer perceptron (MLP) neural network and to analyse its performance. Raw fetal ECG data from the School's Research database and a program for the MLP neural net are made available.

The case study is normally carried out in groups of two students. What we expect from each group at the end of the case study are:

1. A fully trained optimum neural network (optimum in terms of network parameters and performance).
2. A short critical report (one per group), describing the QRS detection algorithm.

The report must include details of the architecture of the net and its implementation, the data pre-processing algorithms, training of the net, test/performance criteria and results. Results include details of how the network parameters (e.g. number of layers, number of nodes per layer, etc.) were determined. A brief summary of the contribution of each student group member is required (1 paragraph per member is sufficient) together with listings of any programs written.

Midway through the case study, a progress report (2 sides of A4 maximum) detailing plans and achievements to date is submitted for discussion. At the end of the case study, each group is required to present their work to the lecturer and demonstrate their neural system, using unseen test data.
To successfully complete the case study, it would normally be required to design algorithms for pre-processing the noisy data before application to the neural network (typically, a bandpass digital filter and threshold/peak detection). The students would also acquire a good appreciation of the principles of the multi-layer perceptron using the error back propagation training algorithm. Most groups complete the task successfully. Figures 4 and 5 are examples of the outcome of the case study.

Figure 4 An example of QRS detection of grade 1 (good quality) ECG

Figure 5 An example of QRS detection of grade 3 (poor quality) ECG

In the past, when the case study was a separate subject, students were required to write their own MLP programs. Now a fully working MLP program is provided.
Case Study 2 - Clock recovery for a satellite data link

Background

In data communication systems, a clock signal is normally extracted from the received data so that the data can be properly decoded. The baseband coding scheme is chosen so that the data contains a strong spectral component at the desired clock rate. Such codes are termed "self clocking". A Phase Locked Loop (PLL) may be used to recover the clock signal from the data, but loss of lock and synchronization can occur if the data "fades" for a period of time. Problems can also occur in regaining lock if the received data is poor and this results in lost data. Also, in order to ensure that the PLL is in a locked condition before data are retrieved a long synchronizing sequence must be applied before the data can be accurately clocked.

A DSP clock recovery scheme [4] exploits the feature of potential instability in an IIR (Infinite Impulse Response) filter, when the poles are close to the unit circle. In the context of clock recovery this feature is desirable since it can be exploited to provide a good flywheel effect over periods of lost input, i.e. a fade.

The clock recovery system and associated waveforms are shown in Figure 6.

![Figure 6 Clock Recovery system with waveforms](image)

Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible stimulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at it's tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E, the recovered clock output.
Task

The task is to model and simulate the DSP based clock recovery system on SPW (Signal Processing Workstation) operating on CAD workstations. Simulated Manchester data are applied and a severe data fade introduced. The clock acquisition and lock properties of the system are measured for a given set of filter coefficients. The filter coefficients are then slightly de-tuned to simulate quantisation effects and the simulation is re-run. The DSP based system performance is compared to that of a conventional analogue PLL.

Examples of simulation results are shown in Figure 7.

Figure 7a Acquisition Performance

Figure 7b Fade Performance

Figure 7a shows simulation results of clock acquisition when applied to Manchester encoded data. A single T/2 pre-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing. After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronization is achieved immediately as the output is in phase with the applied data.

Figure 7b shows the flywheel performance over a 40-bit data fade. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock. Zooming in to the end of the fade, shows the recovered clock remaining in phase when data are re-applied, this represents a flywheel performance of 80 clock periods.

The simulation results are then verified in real time with a demonstration using the TMS320C25 system. Clock recovery at nine data rates ranging from 50 to 4800bits/sec are programmed into common hardware (selectable from a menu via an RS232 terminal), with sample rates of 6.4KHz, 38.4KHz and 76.8KHz.

A two stage IIR filter section, demonstrates how the lock properties of the clock recovery system can be increased by cascading filter sections. The filter outputs are applied to DACs and viewed as analogue signals on an oscilloscope.
Conclusions

The availability of powerful DSP hardware and software tools provide the opportunity to teach the fundamentals of DSP more effectively. We find that with the increasing number of students weak in mathematics, a practical approach allows a more perceptive understanding of the subject.

Our approach succeeds in enthusing students as they participate more actively in the learning process. To work well however, this requires adequate resources and careful planning. For example, we find that our degree programmes that require extensive simulation on SPW has caused problems with student access to workstations. This is because large numbers of students are required to complete SPW assignments on a limited number of workstation nodes. Also the workstations are used for other applications such as logic, printed circuit board and VLSI design. We firmly believe that computer simulation has an important role in the teaching of DSP. It allows students to complete DSP task, which would otherwise be difficult and expensive in hardware. We expect to see an increase in the use of simulation as an analytical tool for future DSP work.

Acknowledgements

In preparing this paper, we have drawn on our course material and those of many colleagues who contribute to the teaching of DSP in the School, especially the DSP course for Communication Engineering given by Dr G. Wade. We are grateful to them all.

References

1 Rabiner L.R. and Gold B. Theory and applications of digital signal processing. Prentice Hall, 1975
DSP-BASED CLOCK RECOVERY IMPLEMENTED IN A FIELD PROGRAMMABLE GATE ARRAY

P M Smithson, M Tomlinson and T Donnelly

In data communication systems, a clock signal is normally extracted from the received data so that the data can be properly decoded. The baseband coding scheme may be chosen so that the data contains a strong spectral component at the desired clock rate. Such codes are termed "self clocking". A Phase Locked Loop (PLL) may be used to recover the clock signal from the data, but loss of lock and synchronisation can occur if the data "fades" for a period of time. Problems can also occur in regaining lock if the received data are poor and this results in lost data. Also, in order to ensure that the PLL is in a locked condition before data are received a long synchronising sequence must be applied before the data can be accurately clocked.

A DSP clock recovery scheme \(^1\) exploits the feature of a long impulse response in an IIR (Infinite Impulse Response) filter, when the poles are close to the unit circle. In the context of clock recovery this feature is desirable since it can be exploited to provide a good flywheel effect over periods of lost input, i.e. a fade.

Theory

The clock recovery system and associated waveforms are shown in Figure 1.

![Diagram of Clock Recovery System](image.png)

Figure 1 Clock Recovery system with waveforms

Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible stimulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at its tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E, the recovered clock output.

School of Electronic, Communication and Electrical Engineering, University of Plymouth, Plymouth, UK, PL4 8AA.
An IIR filter is normally formed as a second order section containing both poles and zeros, this provides a centre frequency and stop-band nulls. In this application, however, the stop-band null is of no consequence and the design can be simplified to have feedback paths only.

![Figure 2 Architecture of the Clock Recovery Filter](image)

The filter output can be expressed in the time domain as,

\[ y(n) = x(n) + Cy(n-1) - Ky(n-2) \] .......... (1)

Taking the Z transform of (1) it can be shown that \(^1,^2\)

\[ H(z) = \frac{Y(z)}{X(z)} = \frac{z^2}{z^2 - Cz + K} \] .......... (2)

Equation (2) is the z-domain transfer function of the filter which contains poles and a double zero at the z-plane origin.

It can be shown from analysis of the z-plane\(^2\), that coefficient \(K < 1\) for a stable system but should be close to unity for a high Q.

The second filter coefficient, \(C\), is determined from,

\[ C = \sqrt{\frac{4K}{(\tan^{-1}\frac{\pi f_\circ}{f_s})^2 + 1}} \] .......... (3)

Values selected are:

\[ K = \frac{255}{256} = 0.99609375, \]

\[ C = 1\frac{15}{16} = 1.9375 \]

A sample rate is then selected to tune the filter to the required resonant frequency:

\[ f_s = \frac{2\pi f_\circ}{\tan^{-1}\left|\frac{\pi}{\sqrt{C^2 - 1}}\right|} \] .......... (4)

where \(f_\circ\) is the required clock rate and \(f_s\) is the sample frequency.
Simulation Results

The DSP-based clock recovery system has been simulated on SPW (Signal Processing Worksystem) operating on CAD workstations. Simulated Manchester and RLL2,7 data are applied and a severe data fade introduced. The clock acquisition and lock properties of the system are measured for a given set of filter coefficients.

Examples of simulation results are shown in Figure 3.

![Figure 3a Acquisition Performance](image)

Figure 3a shows simulation results of clock acquisition when applied to Manchester encoded data. A single T/2 pre-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing. After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronisation is achieved immediately as the output is in phase with the applied data.

Figure-3b shows the flywheel performance over a 40-bit data fade. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock. Zooming in to the end of the fade, shows the recovered clock remaining in phase when data are reapplied, this represents a flywheel performance of 80 clock periods. This flywheel performance can be further improved by cascading IIR filter sections where each section approximately doubles the flywheel performance, i.e. the output of first filter section decays to zero before the stimulus to the second filter is affected.

The closer the poles are placed to the unit circle (higher filter Q), then the greater the flywheel effect over lost input. A high filter Q reduces the bandwidth of the bandpass filter, hence the trade-off to this increased flywheel time is a reduction in data rate drift.

The graph of figure-4 shows that the number of clock periods for which the clock flywheels, increases exponentially with the filter Q. If the acceptable data rate drift of a system is known, then the graph can be used to determine the highest permissible filter Q and hence maximise the clock flywheel period. The simulation results presented used a filter Q = 100, which represents a 1% data rate drift.
Channel noise appears as jitter in the demodulated data. The IIR filter averages phase errors in the received data minimising jitter in the recovered clock. In the simulations ±3 samples jitter with a gaussian distribution (corresponding to a standard deviation of ±25.9° phase jitter) were applied to the encoded symbols and the jitter rejection performance statistically analysed over half a million samples.

The histograms of figure-5 shows that for gaussian jitter applied to Manchester encoded data, the clock recovery filter virtually eliminates all jitter in the recovered clock. The standard deviation of the recovered clock is only 3.3°. While for the RLL2,7 code, nearly 90% of the recovered clock contains zero phase error and the standard deviation of the recovered clock is 7.1°. These graphs were obtained from simulation results over 500,000 samples with a filter Q of 100. To reduce sampling errors and to maximise the jitter detection window, the recovered clock is delayed such that the sampling edge retimes the data at the centre of the bit period.
Implementation

DSP algorithms are typically implemented using general purpose digital signal processors for moderately low sample rates. The simulation results have been verified in real time, using a Texas Instruments TMS320C25 fixed point digital signal processor. Clock recovery at nine data rates ranging from 50 to 4800bit/sec are programmed into common hardware (selectable from a menu via an RS232 terminal), with sample rates of 6.4KHz, 38.4KHz and 76.8KHz.

An ASIC may be used for high performance applications, but the high design cost and long development cycles make them unacceptable for prototypes or low to moderate volume production. Advances in programmable logic technology now offers a third alternative, namely the field programmable gate array (FPGA). Employing an FPGA to implement a DSP function allows the designer to achieve a semi-custom solution, while avoiding the high development cost and associated risk.

An FPGA consists of an array of unconfigured logic which can be configured to perform the required hardware function. The configuration programme is usually contained in a ROM which is automatically downloaded to the FPGA at power-up. The FPGA can also be reconfigured on-the-fly enabling the logic function (for example filter coefficients) to be dynamically changed.

Whilst digital signal processors have an architecture that lend themselves to digital filtering, this unfortunately is not the case with FPGAs. Hence, the common requirement of implementing filter coefficients, can be slow and require a large number of FPGA logic cells to achieve the shift and add process necessary for binary multiplication.

To achieve a high speed solution, whilst still maintaining a degree of programmability, the clock recovery IIR filter has been implemented in a Xilinx 3020PC68-50 (2000 gates, 50MHz) FPGA. To avoid using conventional binary multipliers, the filter coefficients have been implemented using binary weighted coefficients. This is achieved by bit shifting the data words and adding or subtracting to obtain the required coefficient value. Figure 6 shows the architecture of the FPGA filter. The filter has a 16 bit dynamic range and utilises 61 of 64 CLBs (Configurable Logic Blocks).

Coefficient $K$, as previously discussed, is chosen to be close to unity producing poles close to the unit circle, which provides the ringing action for the clock recovery flywheel effect. Hence, coefficient $K$ is set to 0.99609 which has a binary weighting of 255/256. This coefficient is achieved by bit shifting the 16 bit data word, 8 bits to the right providing a binary divided value of 1/256. This bit shifted value is then subtracted from the non shifted data word to provide the required coefficient of $255/256 = 0.99609$.

![Figure 6 FPGA Filter Architecture](image-url)
Similarly, coefficient $C$ is chosen to be 1.9375 which has a binary weighting of $\frac{15}{16}$. The 16 bit data word is shifted 5 bits to the right to provide $1/32$ which is subtracted from the non shifted data word to provide $31/32$. This intermediate value is then shifted to the left 1 bit, providing x2 multiplication as a new 17 bit data word, producing the required $C$ coefficient $\frac{65}{32} = \frac{15}{16} = 1.9375$.

Hence, the filter coefficients are implemented automatically at the sample rate without the need for sub-multiplication processing. The binary weightings do impose quantisation restrictions in the selection of the filter coefficients, however this restriction is offset by the choice of sample rate, which effectively fine tunes the filter to the required centre frequency.

The design of figure-6 has been implemented for clock recovery on a satellite data modem operating at a sample rate of 2.59MHz at a data rate of 100Kbits/sec. With the use of larger, faster FPGA devices and employing adder look ahead carry techniques, much higher data rates are achievable.

The technique of using binary weighted coefficients, results in approximately a 50% reduction in silicon utilisation and a 50% increase in speed performance\(^3\). The technique has also been extended to FIR filter architecture's for channel filtering.

Conclusions

The results for the IIR filter clock recovery system show superior performance, in terms of clock acquisition, over that of a conventional Phase Lock Loop (PLL). Whereas the PLL requires a long training sequence of symbols, before the clock synchronises to the data, the IIR filter clock recovery system synchronises within a symbol period of the first received transition. This feature would significantly reduce the need to transmit synchronising data, which would be desirable in burst mode applications.

The ability to withstand long fades is good with a high filter Q but this restricts the acceptable data drift, so a PLL may be better in this respect if a long time constant is used. However, in improving this aspect of a PLL the acquisition time is made worst. The ability of the IIR clock recovery system to withstand long dropouts can be improved by cascading filters, each additional filter doubles the clock flywheel performance with little degradation in acquisition performance. The rapid re-acquisition after a severe fade can reduce burst errors improving the overall BER (Bit Error Rate) performance.

The IIR filter provides good jitter rejection minimising jitter in the recovered clock. The recovered clock is delayed, such that the sampling edge of the clock re-times the received symbols at the centre of the bit period, maximising the jitter detection window. A high filter Q provides a reduced filter bandwidth and hence reduces jitter in the recovered clock. However, this again is achieved at the expense of reduced data rate drift.

References

THE DEVELOPMENT OF AN OPERATIONAL COMPUTER BASED SATELLITE DATA BROADCASTING SYSTEM

PM Smithson and M Tomlinson

University of Plymouth, UK

Abstract - The hardware and software development of a low cost, 100Kbit/sec data modem for satellite broadcasting and reception of computer disk files. Primarily developed for use as a sub carrier on a standard FM TV transmission, the system has been adapted to operate at 128Kbit/sec utilising a user data port on an MPEG compressed digital TV channel. Modulator and demodulator printed circuit cards have been developed which plug directly into IBM AT compatible PC's (Personal Computers). Applications include distance learning, electronic mail, information service, database updating and software distribution.

INTRODUCTION

The Transmit data from an IBM-PC is modulated using Differential Phase Shift Keying (DPSK) onto a 7.2MHz sub carrier and transmitted along with a standard satellite TV signal. The uplink signal is therefore the same as for normal FM TV signals except for the addition of this extra data sub carrier in a spare part of the channel spectrum.

Fig. 1 Video Baseband Spectrum

The receive (demodulator) card accepts the standard decoder video (baseband) signal from a domestic TVRO (TeleVision Receive Only) receiver. The baseband video and audio outputs from the TVRO may also be connected to a television and simultaneous TV programmes can be viewed without interference from the data modem. The receive modem card contains demodulation circuitry to extract the data sub carrier and recover the clock and data. Clock recovery has been implemented digitally as a multiplier-less recursive filter utilising a field programmable gate array. The data link is continuously active with 100Kbit/sec encrypted data, the encryption/decryption code is contained in a PLD (Programmable Logic Device), the contents of which cannot be copied, providing a first level of data security. Each Receive Board contains a PLD and ROM (Read Only Memory) programmed with a unique user ID, this provides the facility to address, from the uplink, individual users or groups for reception of the transmitted data. The integrated multitasking software (operating under Microsoft Windows V3.1) detects the presence of incoming data which is stored into the PC's memory using DMA (Direct Memory Access). The software allows users to review the received data which is automatically saved to disk. The receiver card has been designed using a digital receiver and demodulator thus eliminating alignment and drift problems normally associated with analogue electronics. Programmable logic has been extensively utilised which has enabled a high degree of circuit complexity but at a reduced board size and cost.

TRANSMIT HARDWARE

The modulator output is a 7.2MHz sub carrier which is combined with the baseband video and sound sub carriers for transmission at the satellite earth station. The 7.2MHz Intermediate Frequency can be adjusted digitally in 200KHz steps over the sub carrier range 6.6 to 9.6MHz. The complete DPSK modulator is contained on an ISA (Industrial Standard Architecture)
half card and may be plugged directly into an IBM AT compatible personal computer. The data to be transmitted is read from the computer's hard disk file into the PC's memory and then transferred to a FIFO (First In First Out) memory via the PC's I/O port. The FIFO holds 64 bytes of data and also provides parallel to serial conversion (PISO). When the FIFO has been filled, the status on the I/O port changes to flag the software to temporarily cease writing data. The contents of the FIFO are then emptied serially to the modulator. When the FIFO is empty the status flag changes and the PC continues to write data to the I/O port. Data is written to the I/O port in this way until the complete file has been transmitted. The transmitted serial data from the FIFO therefore consist of burst of 512 bits (64 bytes). Short idle periods between burst exist, while the software refills the FIFO. Due to the burst nature of the data from the FIFO and to prevent long sequences of ones and zeros in the data sequence, the data is encoded with a RLL (Run Length Limited) encryption code.

The encryption code has been implemented in a PLD (Programmable Logic Device), the security fuse of which may be programmed to prevent the device from being copied. An identical PLD device is used for decryption in the receive board and hence a low cost data security system is achieved. By selective use of codes the data encryption can be changed at will, providing a further level of data security over the user identification ROM. Since the data sub carrier system transfers binary files, a third level of data security may also be applied in software. The encrypted data is passed through a differential encoder which eliminates phase ambiguity of the received data and eliminates the need for carrier recovery in the demodulator. The output from the differential encoder is EXclusive OR'd with the synchronous 7.2MHz system clock. This mixes the baseband signal onto a 7.2MHz carrier with DPSK (Differential Phase Shift Keying) modulation at a data rate of 100K bit/sec.

The 7.2MHz I.F. is passed through a tuned bandpass filter to remove out of band frequency components. The input level to the filter may be adjusted with a variable pre-set which results in adjustment of the modulator output level.

TRANSMIT SOFTWARE

The transmit software has been developed using C++ and operates under Microsoft Windows V3.1. To aid user installation, a set-up disk has been prepared which automatically installs all the necessary files and creates a new SatLink windows group. The transmit software is opened by starting Microsoft Windows and double clicking the Transmit Satellite Dish icon in the SatLink program Group. The Transmit Form is displayed and a transmission list of computer files, assigned to different users, can be prepared or recalled from disk.

The application contains on-line help from a pull-down menu. Quick help on the SatLink transmit screen can be activated by clicking the ? toolbar icon and pointing to the required area on the Transmit Form. Commands can be activated from pull down menus or alternatively
from the appropriate toolbar icon. An information line at the bottom of the Transmit Form provides brief help on the function of each control as the mouse cursor passes over toolbar icons. The Transmit Form contains a list of users and files for transmission. A prepared list can be produced, saved to disk and later recalled for future use.

When the Transmit button is clicked, each file in the list is transmitted separated by a pre-defined delay. The transmit button changes to a Stop button and if clicked terminates the transmission and returns control to the SatLink transmit form. During transmission, the file is first checksummed and then transmitted, a bar graph provides visual indication of the transmission progress of the current file. A flashing satellite appears on the files icon indicating the transmission progress of the list, which is automatically scrolled to display the currently transmitted file. A repeat option may be selected which re-transmits the list on a cyclic basis.

A Company, Group or User is selected from the database and the selection is added to the SatLink transmission list with a different icon type for each category. Files for transmission can then be allocated to the selected users by clicking the Transmit Files toolbar icon. This opens a standard windows type File dialog box where disk and directories can be viewed and files selected.

The order of users and files in the transmission list can easily be changed by making selections with the mouse and using the cut, copy and paste toolbar icons.

Data is transmitted in packets of 60 Bytes where each packet is preceded by a 4 Byte unique word. The unique word is detected by the receive hardware and is used for byte synchronisation purposes. Transmission of a disk file is preceded by a header packet containing information about the file and users authorised for reception. Based upon the users selection displayed in the transmission list, the file and User ID information is determined and the header packet is constructed. The header packet is first transmitted followed by the disk file data. The receive software decodes the header packet and if authorised, uses the header information to prepare the receiver for reception of the data file.
RECEIVE HARDWARE

The complete DPSK demodulator and PC interface is contained on an ISA card and can be plugged directly into an IBM AT compatible personal computer. The demodulator input accepts the standard decoder video (baseband) output from a domestic satellite TVRO. The spectrum of the decoder signal is shown in Fig.1 and contains the baseband video, audio and data subcarriers. The separate video and audio outputs from the TVRO may also be connected to a television and simultaneous TV programmes can be viewed without interference from the data modem.

A 7.2MHz bandpass filter with a -3dB bandwidth of 200KHz places a window around the data subcarrier and rejects the baseband video and audio subcarriers present at the demodulator input. The filtered analogue signal is sampled and applied to a differential decoder which removes the carrier frequency and provides data symbols of the correct phase. The demodulator output is low pass filtered to remove high frequency carrier components. This filter is DC coupled to prevent ramping of the data due to combinations of ones or zeros in the scrambler sequence, which would otherwise result in closing of the data eye. The data eye is converted to logic levels by means of a comparator, the threshold of which has a transistor junction to track DC variations at the signal input.

The clock recovery circuit consists of a digital bit synchroniser to extract the data rate clock from the received data. The principle of operation of the clock recovery system is shown in Fig.12. Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C, this signal has a strong frequency component at the desired clock rate. Before application to the IIR filter, waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values are chosen to provide the highest possible stimulation whilst ensuring that the filter does not become unstable. As a result of the applied stimulus the filter "rings" at its tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E, the recovered clock output. The recovered clock can flywheel over long sequences of ones and zeros in the received data which has been minimised by the RLL encoder.
It can be shown from analysis of the z-plane, Smithson et al (1), Ifeachor and Smithson (2), that coefficient $K < 1$ for a stable system but should be close to unity for a high Q.

The second filter coefficient, $C$, is determined from,

$$C = \sqrt{\frac{4K}{\tan^2\frac{2\pi f_o}{f_s}} + 1}$$

Values selected are: $K = -\frac{255}{256} = 0.99609375$, $C = 1.9375$

A sample rate is then selected to tune the filter to the required resonant frequency:

$$f_s = \frac{2\pi f_o}{\tan^{-1}\frac{4K}{c^2}}$$

where $f_o$ is the clock rate required and $f_s$ is the sample frequency of 2.59MHz.

The digital IIR filter has been implemented in an FPGA (Field Programmable Gate Array) using binary weighted coefficients and adders, thus avoiding binary multipliers.

This representing a 50% increase in speed performance and 50% reduction in silicon utilisation, Wade et al (3).

![Fig. 14 FPGA Clock Recovery Filter](image)

The FPGA configuration programme is stored in an EPROM which also configures a second gate array for the unique word detection and the PC interface.

The output of the clock recovery filter is compared with a hardwired threshold providing indication on an LED of successful synchronisation and hence reception of received data.

It is impossible to filter out all the undesired spectral components close to the filter centre frequency and thus a small timing jitter exist on the recovered clock. To prevent clock jitter miss-timing data, the data is delayed such that the rising edge of the clock is in the centre of the data period. All digital delays are implemented in a programmable shift register sampled at 20MHz.

The retimed data and recovered clock are fed into a decryption PAL which contains the same RLL code as in the transmitter. The decryption is feed forward and is therefore self synchronising. The decryption PAL has its security fuse programmed to prevent the device from being copied. An identical PLD device is used for encryption in the transmit board providing a first level of data security.

The serial data is converted into byte wide data for loading into the PC. The 32 bit unique word is decoded and synchronises a byte counter which slices the serial data into valid bytes. A 512 bit burst counter is also initialised which inhibits any false unique words which may occur in the data sequence, thereby preventing false synchronisation corrupting the data. Unique words are stripped-off by the hardware and only valid data is applied to the I/O port.

![Fig. 15 FPGA PC Interface](image)

The receive software assigns a circular DMA (Direct Memory Access) buffer which must be contained within a DOS segment. The buffer address is determined and is used to programme the PC’s host DMA controller (8237A) address registers.

When a valid byte is available, a DMA1 request signal is asserted. The DMA controller asserts a DMA1 acknowledge signal which enables the data byte onto the PC’s data bus. The DMA controller has asserted the target data memory address onto the PC’s address bus and the byte is read into this location. The DMA acknowledge signal resets the DMA request and the single byte transfer is complete.

The 60 byte header packet is evaluated by the software and the file size is determined which is used to programme the DMA controllers word count register. Successive bytes are read into memory, each address being incremented by the DMA controller around the circular buffer until the complete file has been received. The DMA address pointer is closely followed by a disk pointer which spoils the received data to the hard disk. The DMA controller has been programmed in Auto Initialise mode which with the operation of the circular buffer overwrites old data which has already been spooled to disk, thus enabling reception of large files.
RECEIVE SOFTWARE

The receive software has been developed using C++ and operates under Microsoft Windows V3.1. To aid user installation, a set-up disk has been prepared which automatically installs all the necessary files and creates a new SatLink Windows group. The installation programme also installs a SatLink Windows device driver which can be viewed under Drivers from the Windows Control Panel.

The receive software is opened by starting Microsoft Windows and double clicking the Receive Satellite Dish icon in the SatLink program group.

An information line at the bottom of the Receive Form provides brief help on the function of each control as the mouse cursor passes over toolbar icons. The SatLink receiver is enabled by clicking the Enable button. The button changes to Disable and the message Waiting for data is displayed in the status area.

The Receive Form displays a list box which shows all files received during the current receive session. When the software detects the presence of incoming data, a message is displayed in the Status area and a bar graph indicates the reception progress of the currently received file. As files are received and saved to disk, they are automatically added to the Received Files list box which indicates the file name, the file size and the date and time received. Files already saved to disk (including previous SatLink sessions) are only updated if the received file has a later creation date or time. This feature is useful for updating databases. If the checksum option is set, then all received files saved to disk are checked for errors. During checksumming, a flashing sign appears in the files icon and if error free a tick is displayed, otherwise the file entry is removed from the list and the file is deleted from disk. An opportunity to receive the file again will occur, since data is transmitted on a cyclic basis.

A third party viewer can be assigned in the Options Form, providing integrated viewing of received files. When the filename in the SatLink list box is double clicked, the viewer is invoked with the file displayed. A typical shareware viewer displays the following file formats: Ascii, Hex, Microsoft Word, Word for Windows, Windows Write, Word Perfect, Word Perfect for Windows, Ami Pro, dBase, FoxPro, Clipper, Paradox, Excel, Lotus 123, Symphony, Quattro, Quattro Pro, BMP, ICO, PCX, GIF, TIF, ZIP, LH.

The Options Form also allows the user to assign the disk and directory for storing received files. The SatLink application is multitasking under Microsoft Windows and operates in the background receiving files, while a user can simultaneously work on another Windows application, for example word processing. An option, enables an audible 'beep' on reception of a file, alerting the user that electronic mail has been received.
A list of received files during the current receive session can be saved to disk and later recalled, to again enable integrated viewing. Because the SatLink application operates under Microsoft Windows, received graphic files can be viewed at the resolution of the Windows display device, for example, SVGA at 16.8 million colours. The received files are automatically saved to disk and can be loaded into commercial software applications. If required, the data (such as text, spreadsheets or graphics) can be modified in these applications and printed at the resolution of the Windows printer device. This can include high resolution laser and colour printers.

SATELLITE TRIALS

A series of satellite trials have been successfully completed using EUTELSAT II F3 (Superbeam 16°E, transponder 20). The uplink facility was provided by ESA’s TDS4B satellite earth station sited at Plymouth. The TV standard used was PAL and the power of the unmodulated video signal into the uplink modulator was set to 69.3dB relative to 1Watt. A threshold extension TVRO receiver was used and errors were recorded for the period of the satellite trials. The sub carrier level was 120mV providing an average error rate of $10^{-7}$ at a Carrier to Noise ratio of 14dB. The receive software carries out parity checking with CRC (Cyclic Redundancy Check) on the received files saved to disk and files in error are rejected.

The satellite earth station at Plymouth is now operational on an experimental basis, user data for broadcasting may be supplied on disk or via a telephone modem. The earth station will re-transmit the data during a routine video broadcast for reception by multiple users.

COMPRESSED DIGITAL TV ADAPTATION

With the advent of MPEG compressed digital TV, the SatLink system has been adapted to operate at 128Kbits/sec utilising the user data port of the NTL system, NTL UK (4). The sub carrier modulator and demodulator sections of the cards are by-passed and instead RS422 interfaces are provided for connections to the digital TV encoder and decoder. The SatLink software remains the same, as used in the sub carrier system.

The 128KHz transmit clock is sourced by the NTL 2000 MPEG2 Encoder at the user data port, which is connected to the transmit board via an RS422 interface. SatLink data is then clocked out of the transmit board to the encoder, for transmission with the compressed video and sound.

The NTL 2212 integrated Receiver/Decoder provides the 128Kbits/sec receive clock and data at the user data port, which is connected via an RS422 interface to the receive board. The received clock and data are applied to the decryption PAL and then to the FPGA for unique word detection and interfacing to the PC bus.

Satellite trials have been successfully completed using the NTL system on EUTELSAT II F3, during which SatLink data was received error free, even at low signal to noise ratios (8dB).
REFERENCES


4. NTL, Crawley Court, Winchester, Hants, U.K. User Handbooks: Compression Encoder NTL2000 and Integrated Receiver Decoder NTL2212

ACKNOWLEDGEMENTS

The authors would like to thank the following for their contribution to the project and hence this paper. Software programmers, A Georgousakis (Technical Institute of Athens) and D Smith (University of Plymouth). P Yarwood and the satellite uplink staff at the University of Plymouth, for their co-operation during satellite trials. The European Space Agency for providing uplink facilities and satellite transponder time and finally, the Defence and Research Agency for their encouragement and support.

The Satellite Communication Research Centre is part of the School of Electronic, Communication and Electrical Engineering, University of Plymouth, Drake Circus, Plymouth, United Kingdom, PL4 8AA.
The Development of an Operational Satellite Internet Service Provision
P.M. Smithson, J.T. Slader, D.F. Smith and M. Tomlinson
The Satellite Communications Research Centre, SECEE, University of Plymouth, England, U.K.

ABSTRACT - The development of a low cost operational satellite Internet Service Provision (SISP) is presented, which delivers the full functionality of the internet to clients, who to their geographical location may only have access to a low public telephone network. Return packets are broadcast on a satellite TV channel, at data rates up to 2Mbps. The clients receive hardware consists of a standard satellite TV receiver (analogue FM or digital MPEG-2) and a special SatLink card which plugs directly into the clients personal computer. Satellite trials have been successfully completed and the system is expected to operate commercially with effect from mid 1997.

I. INTRODUCTION
It is well known that modern domestic internet traffic is symmetric in nature, there are many more packets coming to client machines than going out. The ratio can be greater than 10:1 for clients accessing the World Wide Web (WWW). An Internet Service Provider (ISP) will often find that this imbalance leads to inefficient use of connection bandwidth.

The Satellite Internet Service Provision (SISP), allows a client's outgoing packets, which are relatively few in number, to be routed via the slow telephone network to a satellite link which has access to the internet backbone. The return packets are then delivered at high data rates using a satellite TV broadcast channel, resulting in much faster reception of complex WWW pages and other internet services. The users receive hardware consist of a standard satellite television receiver and a special SatLink demodulator card [1] which plugs directly into the clients personal computer. SatLink is an established operational satellite data broadcasting system, developed by the University of Plymouth.

The broadcast channel can use analogue television (FM TV), where data are transmitted at 90Kbps utilising a 7.74MHz audio sub carrier bandwidth slot. Alternatively, a digital MPEG-2 broadcast channel can be used where the data rate is between 128K and 2048Kbps.

The satellite internet data bandwidth is shared between all clients, but the cost is significantly lower than buying further bandwidth over a leased line. SISP was originally designed as a solution to provide an ISP in a remote country that has a low-quality Public Switched Telephone Network (PSTN), where local analogue exchanges may only offer 1200 to 9600bps to standard telephone modems.

II. SYSTEM OVERVIEW
A block diagram of the SISP system is shown in Figure 1, with an internet connection in the UK and the ISP clients located in the Middle East.

A client in the Middle East dials into the Local ISP to establish a Point to Point Protocol (PPP) connection. All traffic to and from the Local ISP is via PSTN modems. If the clients web browser requests a connection to an external machine, the IP packets are routed, via a 64Kbps leased line, to a commercial Host ISP located in the UK. Return packets are modulated and transmitted via the SISP satellite data broadcast system and received on the clients satellite TV receiver. If FM TV is used as the broadcast medium, then the standard baseband output from a domestic satellite receiver is connected to a SatLink demodulator card located within the clients PC. Alternatively, in the case of an MPEG-2 digital TV transmission, the internet data are demultiplexed from the MPEG-2 transport stream within the receiver/decoder and synchronou loss and data are connected to an MPEG version of the SatLink card, again located within the clients PC.

![Figure 1 SISP System](image-url)
the SatLink receive cards pass the data stream to a special Windows device driver, which regenerates the Internet protocol (IP) packets, completing the cycle.

The system is controlled at the Local ISP, which provides authentication and accounting information in addition to operating modem banks for up to 100 clients (additional banks can be added by networking additional PC’s).

### III. THE HOST ISP

An overview of the hardware and software for the Host ISP is shown in Figure 2.

The transmit software drivers provide an interface between the TCP/IP protocol and the SISP transmit hardware. Due to the broadcast nature of the transmission, the transmit driver contains some features that are not usual to other network packet drivers. In order to maintain maximum compatibility, the driver has been written to appear to the Windows-NT/95 operating system, as a receive-only or transmit-only Ethernet card. Although the drivers themselves appear to the Network Device Interface Specification (NDIS) layer as Ethernet cards, much of the unnecessary header information has been stripped from the transmitted data. The transmit driver operates a Proxy-Address Resolution Protocol (ARP) policy for remote clients to remove more overhead. In addition, the 12-byte Ethernet addresses have been stripped from the header, and all packets are delivered on the basis of IP address alone. However, the 4-byte error detection block is maintained in a modified form, to preserve good data integrity. There are also extensions built into the satellite protocol to allow for packet compression, forward error correction and statistical information packets, should the need arise.

The clients SatLink receive hardware is required to reassemble the received serial bit stream back into bytes, also clients must be allowed to start receiving at any time during a broadcast. To achieve this a 32-bit Unique Word (UW) is transmitted periodically to reset the synchronisation counters within the SatLink receive card. Clearly the UW must not occur in the message sequence to avoid false synchronisation, hence any unique words that appear in the data stream are remapped by the transmit driver by means of “bit stuffing”. The clients receive driver recognise when a sequence has been remapped and restores the original data.

Figure 3 provides a block diagram of the SISP transmit card.

**Figure 3 SISP Transmit Card Block Diagram**

The transmit driver writes bytes for transmission to a 4Kbyte First In First Out (FIFO) memory which also provides Parallel In Serial Out (PISO) conversion. The serial data stream is scrambled with an encryption algorithm secured within a Programmable Logic Device (PLD). The SatLink receive cards located within the Client PC’s must have an identical PLD in order to successfully decode the data. The encryption code is run length limited to break up long sequences of binary ones or zeros in the data stream, to assist clock recovery in the receiver. The data rate can be adjusted by means of onboard links to select 90Kbps (for sub carrier applications) or 128K to 2048Kbps (for MPEG-2 applications).

The transmitted data stream from the card is modulated using Differential Phase Shift Keying (DPSK) onto a 7.74MHz sub carrier and transmitted along with a standard satellite TV signal. The uplink signal is therefore the same as for normal FM TV signals except for the addition of this extra data sub carrier in a spare part of the channel spectrum. Alternatively, synchronous clock and data are available (RS422 and RS232 interfaces) for connection to an MPEG-2 Digital SNG Codec for multiplexing into an MPEG-2 transport stream.

**Figure 4 SISP Transmit Card**
IV. THE CLIENT SYSTEM

An overview of the hardware and software for the Client system is shown in Figure 5.

![Figure 5 The Client System](image)

The Clients receive (demodulator) card accepts the standard decoder video (baseband) signal from a domestic TVRO (TeleVision Receive Only) receiver.

The video and audio outputs from the TVRO may also be connected to a television and simultaneous TV programmes can be viewed without interference from the data modem.

![Figure 6 The SatLink Receive Card Block Diagram](image)

The SatLink sub carrier receive card contains a digital demodulator to extract the data sub carrier and recover the clock and data. A novel clock recovery scheme has been implemented digitally as a multiplier-less recursive filter utilising a field programmable gate array [2,3].

The hardware reassembles the data stream into valid bytes which are stored into the PC’s memory at high speed using Direct Memory Access (DMA), under control of the receive driver.

The SatLink card has been designed using a digital receiver and demodulator thus eliminating alignment and drift problems normally associated with analogue electronics. Programmable logic has been extensively utilised, enabling hardware design flexibility at reduced board size and cost.

![Figure 7 Sub Carrier SatLink Receive Card](image)

An alternative card has also been developed for MPEG-2 applications, where synchronous clock and data (RS422 interface) are received from an MPEG-2 Receiver/Decoder which demultiplexes SatLink data from the MPEG-2 transport stream. This interface is also suitable for connection to a conventional Single Channel Per Carrier (SCPC) satellite modem.

![Figure 8 MPEG-2 SatLink Receive Card](image)

The Receive Driver provides the interface between the SatLink receive hardware and the TCP/IP protocol driver. The clients receive driver recognises when a UW sequence has been remapped and removes the bit stuffing to restore Ethernet frames, passing them up to the TCP/IP protocol driver. Ethernet hardware addresses are not transmitted over the satellite link to improve efficiency, so dummy hardware addresses are generated by the receive driver. Since the transmitted data rate over the sub carrier system is relatively low (90Kbps), IP packet filtering is provided by the TCP/IP protocol. At higher data rates the SatLink card must filter IP packets to prevent the clients’ PC from being swamped by packets destined for other users, although this has not yet been implemented.
V. THE LOCAL ISP

A block diagram of the Local ISP system is shown in Fig-9.

![Local ISP System Diagram]

Figure 9 The Local ISP

The Local ISP system, operating under the UNIX operating system, is completely oblivious to the SatLink hardware. It is a standard PPP-based ISP system, where the packets from up to 100 modems are multiplexed down a leased line for connection to the host system located in the UK. Statistical information is displayed and stored providing a record of clients access times and packet usage, which may be used for counting purposes. Other extensions have been implemented using a graphical X-Windows user interface, to allow client accounts to be added or removed by the local provider. The local ISP can also provide a useful service by giving the status of the connections and other maintenance feedback, that may not come via the satellite link.

VI. SATELLITE TRIALS

A series of satellite trials have been successfully completed using EUTELSAT II F3 16°E, widebeam transponder-20, receiving on a 1 metre dish at 11.575160GHz with vertical polarisation. The uplink facility was provided by ESA's DS4B satellite earth station sited at Plymouth. The TV standard used was PAL and the power of the unmodulated video signal into the uplink modulator was set to 72dBW. A threshold extension TVRO receiver was used and errors were recorded for the period of the satellite trials. The signal level at the 7.74MHz data sub carrier at the TVRO decoder output was measured as -30dBm with a signal to noise ratio of 20dB. The adjacent audio sub carrier at 7.56MHz was measured as +3.8dB above the data sub carriers at 7.74 and 7.92MHz.

Error rate performance were monitored over a period of 3 weeks over diverse weather conditions and similar error performance was observed as for the trials. The internet service providers located in Jordan are within the 40 to 45dB contour of Hotbird's Super-Widebeam footprint, and have successfully received EBN and internet data from the data sub carrier using a 1.8 metre dish. The system is expected to operate commercially with effect from August 1997.

VII. COMMERCIAL IMPLEMENTATION

At the time of print the system was undergoing commercial installation and had been successfully installed and tested over a commercial satellite link.

The uplink is provided by BT at Madley which is located at Herefordshire in the UK. The SISP data is transmitted on a 7.74MHz sub carrier on the EBN (European Business News) TV channel. The TV channel carries six secondary sub carriers, two of which are data.

EBN is received on EUTELSAT's Hotbird I at 13°E at 11.265GHz with horizontal polarisation, and shares transponder 3 with a digital TV channel. The signal was received in the UK with 1.2 metre dishes and domestic satellite TV receivers. The signal level of the 7.74MHz data sub carrier at the TVRO decoder output was measured as -30dBm with a signal to noise ratio of 20dB. The adjacent audio sub carrier at 7.56MHz was measured as +3.8dB above the data sub carriers at 7.74 and 7.92MHz.

Error rate performance were monitored over a period of 3 weeks over diverse weather conditions and similar error performance was observed as for the trials. The internet service providers located in Jordan are within the 40 to 45dB contour of Hotbird's Super-Widebeam footprint, and have successfully received EBN and internet data from the data sub carrier using a 1.8 metre dish. The system is expected to operate commercially with effect from August 1997.

VIII. CONCLUSIONS

The Satellite Internet Service Provision (SISP) allows the low cost delivery of an internet service by satellite, which removes the background load from the usual wired connections, freeing them up for page requests and outgoing mail messages. SISP can be used to provide a high speed connection to locations where only slow local analogue exchanges exist and may also provide a lower cost solution to leasing dedicated high speed leased lines.

The system is not limited to operating while clients are connected, since multicast IP packets are received whether the PSTN modems are operating or not. Although this has not yet been implemented, it allows for e-mail signalling and on-line news spoiling. It also has some interesting possibilities for advertising in screen-saver programs.

When operating at 2Mbps, the clients terrestrial modems sending acknowledgment packets at 9.6Kbps, were found to limit the speed of reception to 160Kbps (20Kbytes/s). However the broadcast data...
A packet is shared between all clients, therefore at least 13 clients could be simultaneously on line and actively transferring data, in order to justify the extra bandwidth.

IP addresses are allocated to active modems, not to the receive cards, so because of the layering, the receive driver does not know its IP address. Hence the SatLink receive card has no choice but to pass every packet received to the operating system. This is not a problem at 90Kbps, but is a significant overhead for a 2048Kbps data stream. If the system proves to be popular, then extending the unique word to 48 bits or more, and assigning each card its own number, will allow hardware packet filtering.

IX. REFERENCES


X. ACKNOWLEDGEMENTS

The authors would like to thank Communicado Data Ltd. for commissioning this commercial project. Particular thanks to Pete Hendry and Phil Sabin for the initial concept and their contribution to the success of the project. Thanks are also extended to the University of Plymouth satellite uplink staff and ESA for their co-operation during satellite trials.

The Satellite Communication Research Centre is part of the School of Electronic, Communication and Electrical Engineering, University of Plymouth, Drake Circus, Plymouth, United Kingdom, PL4 8AA.
A Data Reply Link System for Satellite TV Applications

J.T. Slader, P.M. Smithson, and M. Tomlinson

The Satellite Communications Research Centre, SECEE, University of Plymouth, England, U.K.

ABSTRACT - A low cost satellite data reply link is presented which may be used to interactively demand the broadcast of data or video "Direct to Home". Due to the burst nature of the reply packets, rapid acquisition to the signal is necessary. Frequency detection, demodulation and clock recovery algorithms are presented which have been successfully implemented in real time using Digital Signal Processing (DSP). Signal acquisition, demodulation and clock recovery synchronisation are achieved within just 1ms (32 symbols), allowing message data to be successfully decoded for forward error correction and off-line processing.

I. INTRODUCTION

The use of MPEG-2 compressed digital TV will allow increased TV channel capacity offering many more TV services than that used by conventional analogue broadcasting. These new broadcast services "Direct to Home" include video on demand, armchair shopping facilities and many other interactive services. For user interaction, a turn link to the service provider is required. Terrestrial satellite networks offer inferior return link quality, while the cost of conventional VSAT technology is prohibitive. This project is concerned with the provision of a low cost data reply channel using a conventional TVRO antenna but with a modified feed horn, equipped with a small transmitting power amplifier. The data return signal is placed adjacent to the TV broadcast carrier and is capable of sustaining a data rate of 5Kbps, which is suitable for computer based data replies or voice communications. The forward data channel is provided by means of an operational data broadcasting system that has been developed by the University of Plymouth and features a SatLink PC based receive card [1].

The system overview is shown in Fig.1. Voice, fax or binary data files are output from the users set top unit, using serial binary data, to the outdoor unit (synthesiser, BPSK modulator and 200mW power transmitter) which is located in the antenna feed. The serial binary data stream includes configuration data for programming a Direct Digital synthesiser (DDS) to set the return link transmission frequency within the Ku band. To assist rapid acquisition, the transmitted message is preceded by a short preamble of 100 symbols, comprising of phase reversals and a Unique Word (UW). The phase reversals are used by the hub station receiver to detect the presence of a signal, provide frequency correction (demodulation) and symbol timing recovery. The UW is used to initialise (ramp-up) a 1/2 rate convolutional decoder whilst also providing decoder synchronisation.

Figure 1 Return Link Slotted ALOHA System Overview

II. THE USERS SET TOP UNIT

The prototype set top unit consists of a PC (personal computer) with a remote control keyboard and mouse. Located within the PC are two ISA (Industrial Standard Architecture) compatible satellite modem cards which operate in a multitasking environment under Microsoft Windows 95. The inbound data channel uses a SatLink receive card [1] while the return channel is provided by a Data Reply Link transmit card. A block diagram of the transmit card is shown in Fig.2.

The transmit card provides two outputs, a 70MHz I.F. for connection to a conventional VSAT and a V11 interface for connection to the Return Link outdoor unit. The transmit card has a flexible architecture based around an FPGA (Field Programmable Gate Array) allowing various signal processing options to be enabled under software control. The user's message is encrypted, where the encryption algorithm is secured within a PLD (Programmable Logic Device), providing a first level of data security.
transmit sequence is initiated under software control or alternatively in hardware with a TDMA (Time Division Multiple Access) pulse. The preamble comprising of phase reversals and a unique word, are clocked serially out of a ROM (Read Only Memory) for direct transmission. The software application writes the message data to a 4KByte FIFO (First In First Out memory) which also provides parallel serial conversion. When transmission of the preamble is most complete, data bits are clocked out of the FIFO into the FPGA which encodes symbols concatenate to the preamble transmission. The software continues to top-up the FIFO for messages larger than 4KByte until the entire message has been transmitted. When the message has propagated through the decoders and transmission is complete, the transmitter is switched off and the software application prepares for the next message to be transmitted.

Figure 4 Hub Station Receiver DSP Hardware

IV. FREQUENCY ACQUISITION

Due to oscillator tolerances and propagation over the satellite link, the received signal can deviate from the nominal 64kHz I.F. by as much as 10kHz. Frequency Acquisition is the process of detecting and estimating a carrier frequency, whereupon frequency correction (demodulation) can be applied. The term 'Frequency Correction' is used because a variable frequency local oscillator is used for demodulation.

The Fast Fourier Transform (FFT) algorithm is often used as an efficient method of evaluating the Discrete Fourier Transform (DFT). The frequency domain representation provides a convenient means of detecting and estimating a carrier frequency. At low Signal to Noise Ratios (SNRs) there are two factors that limit the performance of the FFT.

1. When carrier power is shared equally between two frequency bins the signal can be hard to detect above background noise and imposes a signal detection threshold.

2. The resolution of carrier frequency estimates are limited to half the frequency bin spacing.

The Offset Fast Fourier Transform (OFFT) offers superior signal detection and frequency acquisition performance compared to the standard FFT at low SNR [4]. Equations for the DFT and the modified Offset DFT are given below;

\[ F(k) = \sum_{n=0}^{N-1} x_n W_N^{nk} \quad k=0,1,...,N-1 \quad \ldots \quad (1) \]
offset DFT:
\[ F(k + c) = \sum_{n=0}^{N-1} x_n W_N^{n(k+c)} \quad k=0,1,...,N-1 \quad c=0.25 \quad \ldots \quad (2) \]

In practice, a standard algorithm can be used for both the FFT and OFFT since only the coefficients change.

Comparing the power spectra of the FFT and OFFT demonstrates how the OFFT can give superior signal detection performance. Fig. 5a and Fig. 5b show power spectra of a carrier applied to a 64-point FFT and 64-point OFFT (c=0.25) respectively.

The FFT power spectrum is symmetrical and shows the carrier power shared equally between two frequency bins. The OFFT power spectrum, for the same input, is not symmetrical since a frequency offset has been introduced. The majority of the power now appears in a single frequency bin. The 'worst case' at low SNR, as shown in Fig. 5a and Fig. 5b, is increased by a factor of 50% thus significantly improving the signal detection threshold.

The OFFT can also be used to derive a more accurate carrier frequency estimate than the FFT, within 0.25f_s/N and 0.5f_s/N spectively, at low SNR. Closer inspection of Fig. 5b reveals the Power Spectrum is offset by -0.25f_s/N in the range 0 to 31 and offset by 0.25f_s/N in the range k=32 to 63.

Using the following algorithm the carrier frequency can be found:

1. Find the frequency bin \( k_{\text{max}} \) containing greatest power ... (3a)
2. \( k_{\text{max}} < N/2 \quad f_{\text{carrier}} = (k_{\text{max}} + 0.25)f_s/N \quad \ldots \quad (3b) \)
3. \( k_{\text{max}} \geq N/2 \quad f_{\text{carrier}} = (N - k_{\text{max}} - 0.25)f_s/N \quad \ldots \quad (3c) \)

This case \( f_{\text{carrier}} = 16.25f_s/N \) or \( f_{\text{carrier}} = 16.75f_s/N \), both having a residual frequency error equal to half the frequency bin spacing (\( f_s/2N \)).

Residual frequency errors manifest as phase errors in the recovered data and are removed using phase tracking techniques or differential demodulation. It is desirable to minimise the phase error by generating the most accurate frequency estimate.

V. HUB STATION RECEIVER OPERATION

Transmitted messages are preceded with a 32kHz phase reversing preamble to aid frequency acquisition and symbol timing recovery. After performing a 256-point OFFT on the incoming samples the receiver examines the power spectrum. The preamble appears as two peaks separated by 32kHz and centred around the carrier frequency as shown in Fig. 6. If a suitable SNR (Signal to Noise Ratio) is exceeded, frequency acquisition is declared and the local oscillator is set using the carrier frequency estimate.

Fig. 6 Power Spectrum of the Reply Link Preamble

The input sequence \( x_i \) is frequency corrected to form the frequency corrected sample sequence \( a_i + jb_i \) as shown by equations (4) & (5);

\[ a_i + jb_i = x_i \times e^{-j2nif_{\text{correction}}} \quad \ldots \quad (4) \]
\[ a_i + jb_i = x_i \times \cos\left(\frac{2nif_{\text{correction}}}{N}\right) - jx_i \times \sin\left(\frac{2nif_{\text{correction}}}{N}\right) \quad \ldots \quad (5) \]

The maximum residual frequency error after frequency correction is equal to \( f_s/4N \) Hz. Since \( f_s = 256kHz \) and \( N = 256 \), the residual frequency error is 250Hz. At a symbol rate of 32,000 symbols per second the maximum phase error, due to the residual frequency error, is given by equation (6);

\[ \text{phase\_error} = \frac{360 \times 250}{32000} = 2.8125^\circ \quad \ldots \quad (6) \]

In keeping with the rapid acquisition and synchronisation targets, the phase reversal symbols that caused the receiver to 'acquire' are also used to begin symbol clock recovery. After frequency correction the signal is applied to the clock recovery algorithm and a stable symbol clock is produced within the first 32 symbols. Prior to Forward Error Correction (FEC) the decoder is initialised and synchronised by detecting the Unique Word at the end of the preamble. Within 1ms (32 symbols) of receiving the start of the preamble, frequency acquisition, frequency correction and symbol lock recovery are achieved. Once the end of the preamble (100 symbols) has been received forward error correction (FEC) begins.
VI. SYMBOL TIMING RECOVERY

A novel DSP based clock recovery scheme demonstrates superior performance compared to that of a PLL; particularly in terms of clock acquisition [2,3]. The scheme exploits the feature of a long impulse response in an IIR (Infinite Impulse response) filter, when the poles are close to the unit circle. In the context of clock recovery this feature is desirable since it can be exploited to provide a good flywheel effect over periods of lost input; i.e. a fade. The clock recovery system and associated waveforms are shown in Fig. 7.

![Figure 7 Clock Recovery Scheme](image)

Encoded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible simulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at its tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E; the recovered clock output.

An IIR filter is normally formed as a second order section containing both poles and zeros, this provides a centre frequency and stop-band nulls. In this application, however, the stop-band null is of no consequence and the design can be simplified to have feedback paths only.

The DSP-based clock recovery system has been simulated on Delta SPW (Signal Processing Worksystem) operating on CAD workstations. Examples of simulation results are shown in Figs. 8 & 9. A T/2 pre-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing. After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronisation is achieved immediately as the output is in phase with the applied data.

![Figure 8 Clock Recovery Acquisition Performance](image)

Fig. 9 shows the flywheel performance over a 40-bit data fade. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock.

![Figure 9 Clock Recovery Fade Performance](image)

Zooming in to the end of the fade shows the recovered clock remaining in phase when data are reapplied; this represents a flywheel performance of 80 clock periods. This flywheel performance can be further improved by cascading IIR filter sections where each section approximately doubles the flywheel performance, i.e. the output of first filter section decays to zero before the stimulus to the second filter is affected.
VII. SATELLITE TRIALS AND TEST RESULTS

A series of satellite trials have been successfully completed using EUTELSAT II F3 widebeam at 7° East (ESA’s igleasie transponder). The uplink used vertical polarisation and the downlink horizontal. Satellite reception and monitoring was provided by ESA’s TDS4B Satellite Earth Station situated at the University of Plymouth. Known data packets were transmitted by the return link terminal and received on TDS4B where the signal was down-converted to 70MHz. A series of back-to-back tests were also carried out at 70MHz I.F. The 70MHz I.F. was applied to the return link electronics where the signal was demodulated, decoded and output to a PC where the BER (Bit Error Rate) performance as measured. The BER performance is plotted in Fig. 10 and compared to theoretical results. It can be seen that for the back-to-back test there is approximately 0.7dB degradation from the theoretical, while the satellite test introduced a further 0.3dB degradation.

The packet acquisition performance is plotted in Fig. 11. For an acquisition probability of 99% or better, an Eb/N0 ratio of 6.8dB is required for back-to-back test while approximately 8dB is required to guarantee successful acquisition over the satellite.

VIII. CONCLUSIONS

The satellite data reply link described, operates at low power using the same transponder as the broadcasting service, occupying just a small part of the unused bandwidth. The system operates at low signal to noise ratios at a data rate of 16Kbps employing an efficient 1/2 rate FEC algorithm. A short synchronisation preamble of just 100 symbols is applied enabling rapid signal acquisition, symbol timing and decoder synchronisation. The system is tolerant of frequency deviation of up to 10kHz and employs differential demodulation or phase tracking to remove residual phase errors. The rapid acquisition techniques described increase the probability of successful message decoding and lend themselves to a polling or TDM type access. With an increase in TV channel capacity many more broadcast channels will become available and these will include armchair pay-on-demand “Direct to Home” services. A low cost satellite data reply link will satisfy these interactive requirements. Applications include video on demand, shopping, interactive data as well as other tele-educational markets.

A series of user trials operating application software is planned for late 1997.

IX. REFERENCES


X. ACKNOWLEDGEMENTS

The authors would like to thank the European Space Agency and Armstrong Electronics for their support during the development of this project. Thanks are also extended to the University of Plymouth satellite uplink staff for their co-operation during satellite trials.
A NOVEL INTERNET DELIVERY SYSTEM USING ASYMMETRIC SATELLITE CHANNELS


The Satellite Communications Research Centre,
SECEE, University of Plymouth, Plymouth, Devon PL4 8AA, England.

ABSTRACT - A novel two-way satellite Internet delivery system is presented. The system employs an MPEG-2 data broadcast channel to deliver data to the user and a domestic 90cm satellite antenna, equipped with a 200mW transmitting power amplifier, to provide the reply channel. The reply carriers occupy spare capacity of the satellite transponder at no extra cost to the service provider. A burst demodulator, for reception of the reply channels, has been implemented using digital signal processors. The operation and algorithms, including rapid carrier frequency acquisition and synchronisation, are discussed. Results and observations from successful satellite trials, associated with the launch of a commercial pilot-scheme within the UK, are presented.

I. INTRODUCTION

Domestic Internet traffic is asymmetric in nature. Complex Web pages generate heavy incoming data while only short request and acknowledgement packets are transmitted. This ratio often exceeds 10:1. A Satellite Internet Service Provision (SISP) system previously developed by the University of Plymouth has outgoing packets sent by a terrestrial link while incoming packets are received at higher data rates using a satellite TV broadcast channel. The result is significantly faster downloads than with a standard Public Switched Telephone Network (PSTN) connection to an Internet Service Provider (ISP). The developments outlined in this paper replace the terrestrial return link with a satellite based data reply link.

Figure 1 - Satellite Transponder Frequency Plans

The reply channels have both low power and bandwidth and occupy spare capacity of the satellite transponder; which has already been allocated to and paid for by the service provider. Figure 1 shows typical transponder frequency plans used during satellite field trials. The first is for MPEG-2 digital television and the second for analogue FM television. In both cases the reply channel carriers are placed close to the broadcast carrier without interference. Additional reply channels are added when user numbers grow and to allow different services to coexist on the same transponder. Reply channels may be prioritised so that users who elect to pay a premium will receive a faster service.

II. SYSTEM OVERVIEW

A system overview is shown in Figure 2. The Host ISP System is located at the satellite earth station. Client terminals may be placed anywhere within the footprint of the satellite. Clients send using 32kps burst transmissions on the data reply channels. These transmissions are received by burst demodulators at the satellite earth station. The transmitted packets are reassembled and transferred to the Host ISP system where they are routed onto the Internet. In the reverse direction, traffic destined for Clients is multiplexed into an MPEG-2 transport stream as user data and broadcast back over the satellite.

Figure 2 - Satellite Internet Service System Overview

A custom network device driver, on both Client and Host terminals, provides the interface between TCP/IP protocol software and the satellite hardware. The device driver also provides Ethernet emulation such that, to the operating system, the satellite hardware appears to be a standard Ethernet Network Interface Controller (NIC). This ensures compatibility with existing network applications. Each set of Client hardware contains a unique ID number so that users may be individually
addressed. System control commands are sent transparently over the satellite channels under the control of a management application. A user database and individual billing logs are maintained at the Host ISP System.

III. CLIENT RECEIVE EQUIPMENT

Broadcast data is demultiplexed from the MPEG-2 transport stream by a domestic satellite TV receiver and applied to a modified SatLink2 receive card, which is shown in Figure 3. The broadcast data channel is continuous and extra frame synchronising information is added to allow the receive cards to obtain synchronisation.

After detecting a synchronising sequence, incoming data is assembled into bytes and stored in the PC's memory using Direct Memory Access (DMA). The device driver reassembles incoming Ethernet frames and examines the destination address field. Packets with the correct destination address are passed on to the TCP/IP protocol driver while others are rejected.

IV. CLIENT TRANSMIT EQUIPMENT

Client transmissions on the data reply channels are achieved using the data reply link transmit card and outdoor unit shown in Figure 4 and Figure 5 respectively. The transmit card has a flexible architecture based around a Field Programmable Gate Array (FPGA) which allows various signal processing options to be enabled under software control. An Ethernet frame for transmission is written to a First In First Out (FIFO) buffer where it is stored until burst transmission is initiated. Once triggered the preamble, comprising of phase reversals and a unique word, is clocked serially from Read Only Memory (ROM) for direct transmission. When transmission of the preamble is almost complete, data is clocked out of the FIFO, FEC encoded and concatenated with the preamble. Once transmission is complete, at the end of the burst, the transmitter is switched off and prepared for the next burst transmission.

The transmit card provides two outputs, a 70MHz LF for connection to a conventional VSAT and a digital interface for connection to the rest of the system.

The outdoor unit is mounted on a domestic 90cm satellite antenna and contains an L-Band to 14 GHz up converter and a 200mw transmitting power amplifier. The transmit frequency is set under software control and may be changed prior to each transmission within the 14 GHz to 14.5 GHz band.
V. HOST SYSTEM TRANSMIT EQUIPMENT

For transmission over the data broadcast channel the Host PC contains the SatLink\textsuperscript{2,3} data broadcast transmit card shown in Figure 6. Transmitted frames are written to a FIFO buffer along with synchronising information. They are then encrypted and made available as synchronous clock and data for multiplexing into the MPEG-2 transport stream.

![Data Broadcast Transmit Card](image1)

Figure 6 - Data Broadcast Transmit Card

VI. HOST SYSTEM RECEIVE EQUIPMENT

Incoming burst transmissions, at 11GHz, are received on a 3.5m Earth station antenna. After down-conversion, a 64kHz I.F. is applied to the Burst Demodulator Digital Signal Processing (DSP) board where it is sampled at 256kHz by an Analogue to Digital Converter (ADC). The DSP hardware consists of a single extended 6U multilayer printed circuit board containing two 80MHz TMS320C50 fixed point digital signal processors and is shown in Figure 7.

![DSP Burst Demodulator Hardware](image2)

Figure 7 - DSP Burst Demodulator Hardware

The remaining processing is performed by DSP software algorithms. Recovered Ethernet frames are transferred to the Host PC where the outputs from several burst demodulators are multiplexed by the device driver. The frequency acquisition algorithm is computationally intensive and utilises both processors. Once frequency acquisition has been successfully achieved the processors are released, to perform modem and decoding algorithms. A single processor runs algorithms for frequency correction (demodulation), symbol timing recovery and unique word synchronisation, while the second processor provides phase tracking, forward error correction and a buffered interface to the Host PC.

VII. CARRIER FREQUENCY ACQUISITION

Due to oscillator tolerances and propagation over the satellite link, the received signal may deviate from the nominal 64kHz I.F. by several kHz. Successive received bursts arriving from different users may differ substantially in carrier centre frequency. Frequency Acquisition is the process of detecting and estimating each carrier frequency, whereupon frequency correction (demodulation) can be applied. The term 'Frequency Correction' is used because the local oscillator is tuned to the incoming signal. The Fast Fourier Transform (FFT) algorithm is often used for carrier frequency acquisition. At low Signal to Noise Ratios (SNRs) there are two factors that limit the FFT's performance;

1. When carrier power is shared equally between two frequency bins, the signal can be hard to detect above background noise.
2. The resolution of coarse carrier frequency estimates are limited to half the frequency bin spacing.

At low SNR, the Offset Fast Fourier Transform (OFFT) offers superior signal detection and frequency acquisition performance compared to that of the FFT\textsuperscript{a}. Equations for the DFT and the modified 'Offset DFT' are given below;

DFT:
\[ F(k) = \sum_{n=0}^{N-1} x_n W_N^k \quad k=0,1,...,N-1 \]  
\[ W_N = e^{-j2\pi/N} \]

Offset DFT:
\[ F(k+c) = \sum_{n=0}^{N-1} x_n W_N^{n(k+c)} \quad k=0,1,...,N-1 \]

In practice, a standard algorithm can be used for both the FFT and OFFT as only the coefficients change.
Comparing power spectra from the FFT and OFFT demonstrates how signal detection performance is improved. The FFT spectrum of Figure 8 shows the case where carrier power is shared equally between adjacent frequency bins.

![Figure 8 - 64-Point FFT Power Spectrum](image)

With the OFFT (c=0.25) a frequency offset is introduced and this situation can not occur. When carrier power is shared equally between adjacent bins in one half of the output it appears in a single bin in the other as in Figure 9.

![Figure 9 - 64-Point OFFT Power Spectrum](image)

For the FFT power spectrum, a coarse carrier frequency estimate with a resolution of $0.5f/N$ can be obtained using the following algorithm:

Find the frequency bin ($k_{max}$) with maximum power in the range $k=0$ to $k=(N/2)-1$.

If $k_{max} < N/2$, $f_{carrier} = k_{max} \times f/N$ Hz

Otherwise, $f_{carrier} = (N-k_{max}) \times f/N$ Hz (3)

For little overhead, a resolution of $0.25f/N$ can be achieved using the OFFT and the following algorithm:

Find the frequency bin ($k_{max}$) with maximum power in the range $k=0$ to $k=N-1$.

If $k_{max} < N/2$, $f_{carrier} = (k_{max}+0.25) \times f/N$ Hz

Otherwise, $f_{carrier} = (N-k_{max}+0.25) \times f/N$ Hz (4)

VIII. SYMBOL TIMING RECOVERY

A novel DSP based clock recovery scheme demonstrates superior performance compared to that of a PLL; particularly in terms of clock acquisition. The scheme exploits the feature of a long impulse response in an Infinite Impulse Response (IIR) filter, when the poles are close to the unit circle. In the context of clock recovery this feature is desirable since it can be exploited to provide low levels of clock jitter. The clock recovery system and associated waveforms are shown in Figure 10.

![Figure 10 Clock recovery Scheme](image)

Coded data symbols, A, are applied to a delay and add pre-processor where they are delayed by a half symbol period and modulo-2 added to the non-delayed data to give the signal C. This signal has a strong frequency component at the desired clock rate. Before application to the IIR filter the waveform C is converted from logic levels to positive and negative numeric values to stimulate the filter. These numeric values must be carefully chosen to provide the highest possible stimulation whilst ensuring the filter does not become unstable. As a result of the applied stimulus the filter "rings" at its tuned frequency to produce waveform D. This waveform is now hard limited back to logic levels to give waveform E; the recovered clock output.

An IIR filter is normally formed as a second order section containing both poles and zeros; this provides a centre frequency and stop-band nulls. In this application, however, the stop-band null is of no consequence and the design can be simplified to have feedback paths only. The DSP-based clock recovery system has been simulated on Alta SPW (Signal Processing Worksystem) operating on CAD workstations. Examples of simulation results are shown in Figure 11 and Figure 12.
In a T/2 pre-processor provides the scaled stimulus to the IIR filter, causing the filter to ring to near maximum dynamic range without overflowing. After the application of data there is a 2-sample delay before the output of the filter is asserted. However, synchronisation is achieved immediately as the output is in phase with the applied data.

Figure 12 shows the flywheel performance over a 40-bit data fade. The IIR filter output decays exponentially and the hard limited filter output provides an uninterrupted recovered clock. Zooming in to the end of the fade shows the recovered clock remaining in phase when data is reapplied; this represents a flywheel performance of 80 clock periods and serves to reduce clock jitter.

IX. BURST DEMODULATOR OPERATION

A diagram of the Burst Demodulator software is shown in Figure 13.

Transmitted messages are preceded with a 32kHz phase reversing preamble to aid frequency acquisition and symbol timing recovery. After performing a 256-point FFT on the incoming samples the receiver examines the power spectrum. The preamble appears as two spectral peaks separated by 32kHz and centred around the carrier frequency as shown in Figure 14.

If a suitable Signal to Noise Ratio (SNR) is exceeded for the two spectral peaks, and they possess the correct frequency separation, frequency acquisition is declared and the local oscillator is set to perform frequency correction. The input sequence \( x_i \) is frequency corrected to form the frequency corrected sample sequence \( a_i + j b_i \) as shown by equation (5);

\[
a_i + j b_i = x_i \cdot e^{-j \frac{2 \pi f_{\text{rejection}}}{N}}
\]

The maximum residual frequency error after frequency correction is equal to \( f_s / 4N \) Hz. Since \( f_s = 256 \) kHz and \( N = 256 \), the worst case residual frequency error is 250 Hz. At a symbol rate of 32,000 symbols per second the maximum phase error between symbols is given by equation (6);

\[
\text{phase error}_{\text{max}} = \frac{250 \text{Hz}}{32000 \text{Hz}} \cdot 360^\circ = 2.8125^\circ
\]
While a phase tracker can cope with this phase change from symbol to symbol, a differential demodulator is used in order to accommodate the relatively large levels of phase noise associated with the low cost frequency sources in the user equipment.

In keeping with rapid acquisition and synchronisation targets, the phase reversal symbols that caused the receiver to 'acquire' are also used to begin symbol clock recovery. After frequency correction the signal is applied to the clock recovery algorithm and a stable symbol clock is produced within the first 32 symbols. Prior to Forward Error Correction the decoder is initialised and synchronised by detecting the Unique Word at the end of the preamble. Within 32 symbols of receiving the start of the preamble, frequency acquisition, frequency correction and symbol clock recovery are achieved. Once the end of the preamble has been received forward error correction begins. Recovered frames are transferred to the Host PC for processing TCP packets.

X. DATA REPLY LINK SATELLITE TRIALS
A series of satellite trials have been successfully completed using EUTELSAT II F4 widebeam at 7° East and EUTELSAT II F3 at 16° East. The uplink used vertical polarisation and the downlink horizontal polarisation. Satellite reception and monitoring was provided by the European Space Agency's (ESA's) TDS4B Satellite Earth Station situated at the University of Plymouth. Known data packets were transmitted by the return link terminal and received on TDS4B where the signal was down-converted to 70MHz. A series of back-to-back tests were also carried out at 70MHz I.F. The 70MHz I.F. was applied to the return link receiver where the signal was demodulated, decoded and output to a PC where the Bit Error Rate (BER) performance was measured. The BER performance is plotted in Figure 15 and compared to theoretical results. It can be seen that for the back-to-back tests there is approximately 0.7dB degradation from the theoretical, while the satellite tests introduced a further 0.3dB degradation, due mostly to phase noise. The packet acquisition performance is plotted in Figure 16. For an acquisition probability of 99% or better, an $E_b/N_0$ ratio of 6.8dB is required for back-to-back tests while approximately 8dB is required to guarantee successful acquisition performance over the satellite.

![Figure 15 - Bit Error Rate Performance](image1)

![Figure 16 - Packet Acquisition Performance](image2)

XI. SYSTEM TRIALS AND SATELLITE PERFORMANCE
The Satellite Internet Delivery system has been extensively tested in a back-to-back configuration throughout development. In full satellite trials, the delays of each satellite link add a further 500ms to the round trip time. Occasional errors on the data reply channel are tolerated as TCP provides a flow control mechanism that automatically recovers from transmission errors. When performance in a back-to-back configuration was compared to the satellite performance, two main effects were observed:
1. When a Web Page is requested, there is an additional one to two second delay before data is received. The TCP connection establishment phase involves a three way handshake between the TCP modules of the client and server. Due to the increased round trip delay, the time taken to establish a connection also increases.

2. In the back-to-back mode, a near optimum transfer rate is consistently achieved. Over the satellite, the data transfer rate is sub-optimum due to the increased round trip time. Performance may be dramatically improved by increasing the TCP acknowledgement window size indicated by the client TCP module. The effect of this is to allow the server to make better use of available bandwidth by minimising idle time waiting for acknowledgements to be received.

Figure 17 shows the maximum throughput per user as a function of the TCP acknowledgement window size. The round trip time for the satellite tests in this particular case was 610ms.

Figure 17 - Data Throughput Test Results
These results were obtained by downloading a 6Mbyte test file over the satellite using the File Transfer Protocol (FTP), first with the default acknowledgement window size of 8192 bytes and then with a near maximum acknowledgement window of 61320 bytes. Using an incoming data rate of 1Mbps and an outbound rate of 16kbits, the effect of increasing the TCP acknowledgement window size can be clearly seen. The measured transfer rate increases from 11.7kbytes per second to over 40kbytes per second. It is interesting to note that with the satellite delay a single user is not able to monopolise the total available bandwidth and is limited to approximately 300kbs. In back-to-back tests, excluding the satellite, a single user can achieve a transfer rate in excess of 100kbytes per second, thus utilising the whole of the available bandwidth. This confirms that the system delays serve to reduce overall throughput for a single user.

Figure 18 shows the satellite broadcast data channel utilisation, sampled at 10 second intervals, over 30 minutes for a Web browsing session in the artificial situation where there are only two active users. Occasional periods of high utilisation can be seen while data is being transferred but there are also long periods of inactivity where each user is studying the information they have retrieved. In practice there will be many active users tending to average out the burst nature of the satellite channel utilisation. However, there will still be peaks and troughs evident.

At the time of writing, user equipment has been installed at a number of sites within the South West of England in order to obtain user feedback prior to the launch of a commercial system. A series of commercial satellite trials are planned shortly so that typical users can evaluate the system and provide additional feedback which will aid future development.
XII. CONCLUSIONS

The satellite data reply link described operates at low power and uses the same transponder as the broadcasting service; the data reply carriers occupy a small part of the satellite bandwidth assigned to the broadcast service which is typically unused bandwidth. The system operates at low signal to noise ratios with a user return data rate of 16kpbs employing an efficient 1/2 rate FEC algorithm. A short synchronisation preamble is applied enabling rapid signal acquisition, symbol timing and decoder synchronisation. The system is tolerant of frequency errors and employs either differential demodulation or phase tracking to remove residual phase errors depending upon the level of phase noise experienced. The rapid acquisition techniques described exhibit high probabilities of successful message decoding.

The data reply link has been combined with an MPEG-2 data broadcast channel to provide a novel satellite Internet delivery system. The system exploits the fact that domestic Internet traffic is asymmetric in nature. The system has particular appeal to regions without good access to the PSTN, and is useful for portable applications, as a high speed Internet link may be established anywhere within the satellite's footprint. Successful technical and user satellite trials have been conducted and commercial trials are planned for the near future.

XIII. REFERENCES


XIV. ACKNOWLEDGEMENTS

The authors would like to thank BNSC (The British National Space Council), Communicado Data Ltd., ESA (The European Space Agency) and Armstrong Electronics Ltd. for their support. Thanks are also extended to the University of Plymouth satellite uplink staff and the RATIO project staff for their assistance during satellite tests and user trials.