Digital beacon receiver for propagation experiments: optimization and testing of a Digital Frequency-Locked Loop/Phase-Locked Loop

Aalberts, D.J.

Award date:
1995

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Report of graduation work:

Digital Beacon Receiver
for propagation experiments

Optimization and testing of a Digital
Frequency-Locked Loop/
Phase-Locked Loop

by D.J. Aalberts

Coach : Ir. J. Dijk
Supervisor : Prof. Dr. Ir. G. Brussaard
Period : February 1995 - December 1995

The Faculty of Electrical Engineering of Eindhoven University of Technology does not accept any responsibility regarding the contents of graduation reports.
Summary

Propagation experiments at Eindhoven University are carried out using beacon signals of satellites. The equipment used to receive the beacon signals consists of an analog PLL and a digital detector, based on a Z80 microprocessor. The processing power of this processor is poor. Therefore the telecommunications division started the development of a digital receiver, based on a Digital Signal Processor (DSP). The receiver consists of a digital Frequency-Locked Loop / Phase-Locked Loop and a Decimator/Detector.

The digital Frequency-Locked Loop / Phase-Locked Loop (FLL/PLL) was already built in a previous project but this design was reconsidered. The hardware is designed using a Texas Instruments TMS 320C25 DSP. The digital FLL is used to realize that the incoming analog signals are mixed to a fixed intermediate frequency. At this intermediate frequency, subsampling of the signal is performed and the All Digital PLL is used to generate coherent in-phase and quadrature components derived from of the received harmonic beacon signals. This is necessary to perform coherent detection of the received signals.

Unfortunately, the first prototype receiver did not function very well. The main problems encountered in the software of the FLL/PLL.

The purpose of this study for graduation is to optimize and measure the characteristics of the FLL/PLL. All parts of the software were studied and tested. The general setup of the software proved to be good. However, several bugs were found and debugged in the implementation.

For characterization of the FLL/PLL several measurements have been executed. The FLL is tested and proved to be able to track a beacon signal of the "ITALSAT" satellite with a C/N ratio at the input of the FLL/PLL of at least -16.5 dB. The PLL is tested and will lock on a received beacon signal with a C/N ratio at the input of at least -14.5 dB. These figures compare very good with respect to measured figures of analog Phase-Locked Loops of proven design.
# Contents

Summary ............................................................... i

Contents ............................................................. ii

Symbols ............................................................... iv

Abbreviations ......................................................... vi

1 Introduction ......................................................... 1
   1.1 Propagation research at Eindhoven University of Technology 1
   1.2 The problem definition ........................................ 3

2 The Frequency-Locked Loop/Phase-Locked Loop .................. 4
   2.1 The Frequency-Locked Loop ................................. 5
   2.2 The Phase-Locked Loop ...................................... 5
   2.3 The coherent detection of the signals to be measured ...... 6
   2.4 Communication between the different devices .............. 8

3 Hardware for the digital FLL/PLL ............................... 10
   3.1 The Digital Signal Processor .............................. 10
   3.2 Memory ....................................................... 11
   3.3 The waitstate generator ................................... 12
   3.4 The I/O section .............................................. 12

4 Frequency Tracking with the FLL ................................ 14
   4.1 The Fourier Transform ...................................... 14
   4.2 The Fast Fourier Transform ............................... 15
   4.3 Window functions to be used .............................. 16
   4.4 Choosing the number of samples for the FFT ............. 18

5 Digital Phase-Locked Loops ...................................... 20
   5.1 A conventional analog PLL ............................... 20
   5.2 The All Digital Phase-Locked Loop ....................... 21
   5.3 Measurement results of the ADPLL ......................... 24

D.J. Aalberts .......................... ii
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Communication between the different devices</td>
<td>32</td>
</tr>
<tr>
<td>6.1</td>
<td>The communication between the FLL/PLL and the PC</td>
<td>32</td>
</tr>
<tr>
<td>6.2</td>
<td>Solutions for error correction</td>
<td>34</td>
</tr>
<tr>
<td>6.3</td>
<td>The communication between the FLL and the Decimator/Detector</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Implementation of the FLL/PLL software</td>
<td>36</td>
</tr>
<tr>
<td>7.1</td>
<td>Function description of and improvements made in the FLL/PLL programs</td>
<td>36</td>
</tr>
<tr>
<td>7.2</td>
<td>Unsolved problems</td>
<td>39</td>
</tr>
<tr>
<td>7.3</td>
<td>Memory allocation</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions and recommendations</td>
<td>40</td>
</tr>
<tr>
<td>8.1</td>
<td>Conclusions</td>
<td>40</td>
</tr>
<tr>
<td>8.2</td>
<td>Recommendations</td>
<td>41</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>42</td>
</tr>
</tbody>
</table>

Appendices

<table>
<thead>
<tr>
<th>Letter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>The 12.288 kHz oscillator</td>
<td>44</td>
</tr>
<tr>
<td>B</td>
<td>The calculation of the constants for the FFT</td>
<td>45</td>
</tr>
<tr>
<td>C</td>
<td>The (sub)sampling theorem</td>
<td>47</td>
</tr>
<tr>
<td>D</td>
<td>The software for the TMS320C25</td>
<td>48</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Unit</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>$\Delta f$</td>
<td>bin size</td>
<td>[Hz]</td>
</tr>
<tr>
<td>$\varepsilon_{ss}$</td>
<td>Steady state error</td>
<td>-</td>
</tr>
<tr>
<td>$\theta$</td>
<td>angle between $0^\circ$ and $90^\circ$</td>
<td>[degrees]</td>
</tr>
<tr>
<td>$\theta_{co-x}$</td>
<td>angle between $x_{co}$ and $x_x$</td>
<td>[degrees]</td>
</tr>
<tr>
<td>$\theta_o$</td>
<td>detection angle</td>
<td>[degrees]</td>
</tr>
<tr>
<td>$\xi$</td>
<td>damping ratio of the PLL-filter</td>
<td>-</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Angle</td>
<td>[degrees]</td>
</tr>
<tr>
<td>$\phi_d$</td>
<td>phase angle difference</td>
<td>[rad]</td>
</tr>
<tr>
<td>$\phi_i$</td>
<td>Input phase angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>$\phi_o$</td>
<td>Output phase angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Angular frequency</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>Harmonic angular frequency</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$\omega_0$</td>
<td>Angular centre frequency at the output</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$\omega_i$</td>
<td>Angular centre frequency at the input</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$\omega_n$</td>
<td>Natural angular frequency of the PLL loopfilter</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$A_{co}$, $A_x$</td>
<td>co-polar amplitude, cross-polar amplitude</td>
<td>-</td>
</tr>
<tr>
<td>$B_{0o}$, $B_1$, $B_2$</td>
<td>Beacon Signals of 12.5, 20 and 30 GHz</td>
<td>-</td>
</tr>
<tr>
<td>BW</td>
<td>Bandwidth</td>
<td>[Hz]</td>
</tr>
<tr>
<td>$C_0$</td>
<td>DVCO Constant (free running angular frequency)</td>
<td>-</td>
</tr>
<tr>
<td>$C_1$</td>
<td>Constant used in the ADPLL</td>
<td>-</td>
</tr>
<tr>
<td>$C_2$</td>
<td>Constant used in the ADPLL</td>
<td>[rad/s^2]</td>
</tr>
<tr>
<td>$f$</td>
<td>Oscillator frequency</td>
<td>[Hz]</td>
</tr>
<tr>
<td>$f_s$</td>
<td>Sample frequency</td>
<td>[Hz]</td>
</tr>
<tr>
<td>$G$</td>
<td>Gain</td>
<td>-</td>
</tr>
<tr>
<td>$h(n)$</td>
<td>discrete impulse response</td>
<td>-</td>
</tr>
<tr>
<td>$H_{CL}$</td>
<td>Closed Loop Transfer function</td>
<td>-</td>
</tr>
<tr>
<td>$H_{OL}$</td>
<td>Open Loop Transfer function</td>
<td>-</td>
</tr>
<tr>
<td>$I_{co}$, $I_x$</td>
<td>In phase, co-polar and x-polar</td>
<td>-</td>
</tr>
<tr>
<td>$k$</td>
<td>Sample counter</td>
<td>-</td>
</tr>
<tr>
<td>$M$</td>
<td>$^2\log N$</td>
<td>-</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of FFT inputs</td>
<td>-</td>
</tr>
<tr>
<td>$n_{co}$, $n_x$</td>
<td>Noise amplitude</td>
<td>-</td>
</tr>
<tr>
<td>$Q_{co}$, $I_{co}$</td>
<td>Quadrature, co-polar and x-polar</td>
<td>-</td>
</tr>
<tr>
<td>$s$</td>
<td>Laplace Transform variable</td>
<td>-</td>
</tr>
<tr>
<td>$t$</td>
<td>Time</td>
<td>[s]</td>
</tr>
<tr>
<td>$T$</td>
<td>time between samples</td>
<td>[s]</td>
</tr>
<tr>
<td>$t_l$</td>
<td>Clock signal low-time</td>
<td>[sec]</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>$W_N$</td>
<td>Twiddle Factor</td>
<td></td>
</tr>
<tr>
<td>$X(\omega)$</td>
<td>Fourier transformed input signal</td>
<td></td>
</tr>
<tr>
<td>$X(k), X_1(k), X_{11}(k), X_{12}(k)$</td>
<td>Discrete Fourier transform of $x(t)$</td>
<td></td>
</tr>
<tr>
<td>$x(nt)$</td>
<td>Sampled signal</td>
<td></td>
</tr>
<tr>
<td>$x(t)$</td>
<td>Analog signal</td>
<td></td>
</tr>
<tr>
<td>$x_{co}, x_x$</td>
<td>co-polar input signal</td>
<td></td>
</tr>
<tr>
<td>$X_i(k)$</td>
<td>Delayed input signal</td>
<td></td>
</tr>
<tr>
<td>$x_i(t)$</td>
<td>Input signal</td>
<td></td>
</tr>
<tr>
<td>$x_o(t)$</td>
<td>Output signal</td>
<td></td>
</tr>
<tr>
<td>$Y_i(k)$</td>
<td>Hilbert transformed input signal</td>
<td></td>
</tr>
<tr>
<td>$z$</td>
<td>$z$-transform variable</td>
<td></td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>A/D, D/A</td>
<td>Analog/Digital, Digital/Analog</td>
<td></td>
</tr>
<tr>
<td>BW</td>
<td>Bandwidth</td>
<td></td>
</tr>
<tr>
<td>C/N</td>
<td>Carrier to Noise ratio</td>
<td></td>
</tr>
<tr>
<td>CL</td>
<td>Closed Loop</td>
<td></td>
</tr>
<tr>
<td>Co,X</td>
<td>Co-polar, Cross-polar</td>
<td></td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
<td></td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
<td></td>
</tr>
<tr>
<td>DPLL, ADPLL</td>
<td>(All) Digital Phase-Locked Loop</td>
<td></td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
<td></td>
</tr>
<tr>
<td>DUT</td>
<td>Delft University of Technology</td>
<td></td>
</tr>
<tr>
<td>DVC0</td>
<td>Digital Voltage Controlled Oscillator</td>
<td></td>
</tr>
<tr>
<td>EOT</td>
<td>End Of Transmission</td>
<td></td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electrically Erasable Programmable ROM</td>
<td></td>
</tr>
<tr>
<td>EUT</td>
<td>Eindhoven University of Technology</td>
<td></td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
<td></td>
</tr>
<tr>
<td>FIFO</td>
<td>First In, First Out</td>
<td></td>
</tr>
<tr>
<td>FLL</td>
<td>Frequency-Locked Loop</td>
<td></td>
</tr>
<tr>
<td>I/Q</td>
<td>In Phase/Quadrature</td>
<td></td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>Intermediate Frequency</td>
<td></td>
</tr>
<tr>
<td>I/O</td>
<td>Input Output</td>
<td></td>
</tr>
<tr>
<td>IRQ</td>
<td>Interrupt ReQuest</td>
<td></td>
</tr>
<tr>
<td>L-ADPLL</td>
<td>Linear All Digital Phase-Locked Loop</td>
<td></td>
</tr>
<tr>
<td>LEN</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>LF</td>
<td>Loop Filter</td>
<td></td>
</tr>
<tr>
<td>LNB</td>
<td>Low Noise Block converter</td>
<td></td>
</tr>
<tr>
<td>LO</td>
<td>Local Oscillator</td>
<td></td>
</tr>
<tr>
<td>OL</td>
<td>Open Loop</td>
<td></td>
</tr>
<tr>
<td>OTS</td>
<td>Orbital Test Satellite</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>Phase Detector</td>
<td></td>
</tr>
<tr>
<td>PLL</td>
<td>Phase-Locked Loop</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
<td></td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
<td></td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
<td></td>
</tr>
<tr>
<td>SOH</td>
<td>Start of Header</td>
<td></td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage Controlled Oscillator</td>
<td></td>
</tr>
</tbody>
</table>
Introduction

1.1 Propagation research at Eindhoven University of Technology

Research on radiowave propagation forms an important contribution to the design of telecommunications applications. Eindhoven University of Technology (EUT) and PTT Research started propagation experiments with the Orbital Test Satellite (OTS), which carries a propagation beacon and transponder. Extensive tests were carried out at 11 and 14 GHz. The Olympus satellite, which carries three beacon transmitters operating at 12.5, 20 and 30 GHz, was launched and commissioned in 1989. This provides the opportunity to characterize the atmospheric transmission channel. EUT carries out the complete set of measurements possible with the Olympus beacons. Radiometers at all three frequencies and meteorological instruments are used at the earth station. The comparison of the 12.5 GHz, 20 and 30 GHz results will give insight in the frequency dependence of the propagation effects. These results will be applied to the prediction of signal impairments on satellite earth links [10].

The beacon receiver is divided into five different functional parts as shown in Figure 1.1. The first stage is the antenna which is capable of receiving three beacon signals simultaneously. The Co- and Cross-polar components of these three frequencies are provided at six separate antenna ports.

D.J. Aalberts
The Low Noise Block (LNB) downconverts the received signals to 10 MHz. The converted signals are band filtered and amplified. The Phase-Locked Loop (PLL) will maintain the input frequency at a stable 125 kHz by measuring the Phase Difference in a Phase Detector (PD). The signal representing the phase difference controls a Voltage Controlled Oscillator (VCO), which is used for mixing to the 10 MHz signal. With the stable 125 kHz signal the digital detector determines the amplitude of the Co- and Cross-polar signals and phasedifference between them.

![Figure 1.1 Block diagram of the receiver](image)

Currently, analog receivers and digital detectors are used to process the converted signals. The analog receiver splits the incoming co- and cross-polar signals to in-phase and quadrature components. The digital detector, developed by PTT Research, determines the amplitude and the phase of the incoming satellite signals. The detector was built using a Z80 microprocessor. Unfortunately, this processor does not have enough processing power to process complicated algorithms; consequently, the algorithms have been simplified.

The Telecommunications Division started to design a new detector, using the TMS320C25 Digital Signal Processor (DSP). This processor has more processing power; therefore, and filter characteristics and communication possibilities were improved. In 1991 this resulted in a dual channel digital detector with a serial communications link to a PC [11].

The next step was the design of a digital Phase-Locked Loop (PLL), combined with a frequency tracking facility, indicated as the Frequency-Locked Loop (FLL). The detector is improved by adding a decimator, which reduces the input data. The whole path from receiver to data acquisition is digital in this setup. In this way the main disadvantage of analog receivers, i.e. the non-constant behaviour in time, will be avoided.

The FLL/PLL was implemented using the same DSP. However, a new board was designed for both the FLL/PLL and the Decimator/Detector. The two boards are based on the same architecture. Some differences exist in the number of A/D and D/A converters, communication possibilities etc.
1.2 The problem definition

The software for the FLL/PLL was designed and implemented by Op den Camp [1]. Strik [2] implemented the software for the Decimator/Detector. Together they wrote several parts of the programs. Unfortunately the software for the FLL/PLL did not function very well. Franken [9] has improved the FLL/PLL-software. However, this software still did not function satisfactorily when Franken graduated.

The purpose of this study for graduation is to optimize and characterize the FLL/PLL. All parts of the software were studied and tested. The general design of the software proved to be good. However, several bugs were found. For characterization of the FLL/PLL several measurements have been done.

This report describes the FLL/PLL hard- and software, the improvements made and the measurements done. Chapter 2 describes the functional configuration of the FLL/PLL. In Chapter 3 the FLL/PLL hardware is described. Chapter 4 deals with algorithms used for frequency tracking. In Chapter 5 the PLL is described. The configuration and algorithms used for the communication are described in Chapter 6. In Chapter 7 general configuration problems, programming problems etc. are described. Finally, conclusions and some recommendations are presented in Chapter 8.
The FLL/PLL software developed at the Telecommunications Division of EUT consists of four functional blocks [1] as shown in Figure 2.1:
- The Frequency-Locked Loop, which determines the carrier frequency of the co-polar signal;
- the Phase-Locked Loop, which has the possibility to recover the carrier of the co-polar signal with a poor Signal to Noise ratio;
- the coherent detector, which resolves the in-phase and quadrature components from the co- and crosspolar input signals with the use of the recovered carrier;
- a communications module, which is able to send data to the Decimator/Detector and to the data logger PC.

![Figure 2.1 Block diagram of the EUT digital FLL/PLL satellite receiver](image)

Optimization and testing of a digital FLL/PLL
2.1 The Frequency-Locked Loop

The Frequency-Locked Loop maintains the frequency of the input signal constant by determining the frequency of the carrier, calculating the error between the frequency of the input signal and the frequency of the output signal and correcting the difference in frequency with a Voltage Controlled Oscillator (VCO).

![Diagram of Frequency-Locked Loop](image)

**Figure 2.2 The Frequency Locked Loop**

The carrier frequency is found by calculating the discrete frequency spectrum of the input signal. The frequency bin with the highest signal power should be the carrier frequency. Calculating the difference between the desired frequency and the measured frequency will result in an error signal. The error signal controls the Voltage Controlled Oscillator (VCO). The signal generated by the VCO is mixed with the input signal.

Op den Camp [1] has formulated several constraints for the FLL:
- maintain a carrier frequency of 124.516 kHz with the use of a VCO;
- output voltage of -5V to 5V to control VCO;
- possibility to transmit the data that represents the frequency spectrum to the PC;
- possibility to transmit the FLL-status to the Decimator/Detector.

2.2 The Phase-Locked Loop

The function of a Phase-Locked Loop is to recover the carrier from a signal with a poor Carrier/Noise (C/N) ratio. In this application the PLL is implemented using a Digital Signal Processor. The implemented algorithm is the digital representation of the analog version. The algorithm is described in Chapter 4.
Op den Camp [1] has formulated several constraints for the PLL:
- An optimum natural frequency of 17 Hz, which equals a noise bandwidth of 56 Hz. This results in an input Signal to Noise ratio that may reduce to 25 dB/Hz before loss of lock occurs;
- Constant loop bandwidth because an adaptively controlled loop bandwidth results in additional software complexity;
- Possibility to transmit the Lock-Status to the Decimator/Detector.

2.3 Coherent detection of the signals to be measured

Coherent detection is used to determine the in-phase and quadrature component of the copolar and crosspolar signals. Detection is achieved by mixing the input signal with the recovered carrier, which is the VCO output signal, shifted 90° in phase [1].

The Co-polar signal $x_{co}$ is represented by:

$$ x_{co}(t) = A_{co} \cos(\omega t + \theta_{osc}(t)) + n_{co}(t) \cos(\omega t) - n_{co}(t) \sin(\omega t) \tag{2.1} $$

The Cross-polar signal $x_x$ is represented by:

$$ x_x(t) = [A_x + n_{xc}(t)] \cos(\omega t + \theta_{co-x}) - n_{xx}(t) \sin(\omega t + \theta_{co-x}) \tag{2.2} $$
The signal received at the co-polar input is given in (2.1). Assuming that the satellite and ground station local oscillator are ideal, the co-polar input signal is represented by [1]:

\[ x_{co}(t) = [A_{co} + n_{co,c}(t)]\cos(\omega_{t}t) - n_{co,s}(t)\sin(\omega_{t}t) \]  

(2.3)

Applying coherent detection to equation 2.3 and removing the high frequency components will result in an in-phase and quadrature-component:

\[ I_{co}(t) = [A_{co} + n_{co,c}(t)]\cos(\omega_{t}t) - n_{co,s}(t)\sin(\omega_{t}t) \]

\[ \Rightarrow [A_{co} + n_{co,c}(t)]\cos(\theta_{0}(t)) + n_{co,s}(t)\sin(\theta_{0}(t)) \]

(2.4)

\[ Q_{co}(t) = [A_{co} + n_{co,c}(t)]\sin(\omega_{t}t) - n_{co,s}(t)\cos(\omega_{t}t) \]

\[ \Rightarrow n_{co,s}(t)\cos(\theta_{0}(t)) - [A_{co} + n_{co,s}(t)]\sin(\theta_{0}(t)) \]

In case of ideal detection, \( \theta_{0}(t)=0 \), equation 2.4 will reduce to:

\[ I_{co}(t) = A_{co} + n_{co,c}(t) \]  

(2.5)

\[ Q_{co}(t) = n_{co,s}(t) \]

Averaging equation 2.5 will result in:

\[ <I_{co}(t)> = A_{co} \]

(2.6)

\[ <Q_{co}(t)> = 0 \]

Demodulation of the co-polar signal results in an in-phase and quadrature component. The in-phase component will be equal to the amplitude of the co-polar input signal.

Assuming the satellite and ground station local oscillator are ideal, the cross-polar signal (2.2) can also be demodulated using coherent detection, which results in:

\[ I_{x}(t) = [A_{x} + n_{x,c}(t)]\cos(\omega_{t}t + \theta_{co-x}) - n_{x,s}(t)\sin(\omega_{t}t + \theta_{co-x}) \cdot 2\cos(\omega_{t}t + \theta_{0}(t)) \]

\[ \Rightarrow [A_{x} + n_{x,c}(t)]\cos(\theta_{co-x} - \theta_{0}(t)) - n_{x,s}(t)\sin(\theta_{co-x} + \theta_{0}(t)) \]

(2.7)

\[ Q_{x}(t) = [A_{x} + n_{x,c}(t)]\sin(\omega_{t}t + \theta_{co-x}) - n_{x,s}(t)\cos(\omega_{t}t + \theta_{co-x}) \cdot 2\sin(\omega_{t}t + \theta_{0}(t)) \]

\[ \Rightarrow [A_{x} + n_{x,c}(t)]\sin(\theta_{co-x} - \theta_{0}(t)) - n_{x,s}(t)\cos(\theta_{co-x} + \theta_{0}(t)) \]

In case of ideal detection, \( \theta_{0}(t)=0 \), (2.7) will reduce to:

\[ I_{x}(t) = [A_{x} + n_{x,c}(t)]\cos(\theta_{co-x}) - n_{x,s}(t)\sin(\theta_{co-x}) \]

(2.8)

\[ Q_{x}(t) = [A_{x} + n_{x,c}(t)]\sin(\theta_{co-x}) + n_{x,s}(t)\cos(\theta_{co-x}) \]

Averaging Equation 2.8 will result in:

\[ <I_{x}(t)> = A_{x}\cos(\theta_{co-x}) \]

(2.9)

\[ <Q_{x}(t)> = A_{x}\sin(\theta_{co-x}) \]

Both signals are a measure of the phase difference between the co- and cross-polar.

The phase difference can be determined by:

---

D.J. Aalberts
\[ \theta_{\text{co-x}} = \arctan \frac{Q_x}{I_x} = \arctan \frac{A_x \sin \theta_{\text{co-x}}}{A_x \cos \theta_{\text{co-x}}} \quad (2.10) \]

Moreover, it is possible to calculate \( A_x \) from the detected co- and cross-polar by true envelope detection:

\[ A_x = \sqrt{\langle I_x(t) \rangle^2 + \langle Q_x(t) \rangle^2} = A_x \left[ \sin^2 \theta_{\text{co-x}} + \cos^2 \theta_{\text{co-x}} \right] \quad (2.11) \]

### 2.4 Communication between the different devices

There must be a communication link between the FLL/PLL detector and the decimator/detector board. Data which have to be transmitted consists of the in-phase and quadrature components. The communication link is based on a First In, First Out (FIFO) principle. All data are sent to the other board without handshaking or error correction. This is done because the transfer rate must be high.

Additional data have to be sent to a PC via a parallel data link. Data to be transmitted are the frequency spectrum calculated by the FLL and the status of the FLL. The PC-link is based on a Centronics interface. The redefined Centronics interface operates in a master slave configuration where the PC is the master. It is possible to connect 8 slaves. The principles of the communication between the different devices will be extensively discussed in Chapter 6.
The hardware of the FLL/PLL is designed by Op den Camp [1] and Strik [2]. The global design consists of a Digital Signal Processor (DSP), memory, a waitstate generator, an I/O-section and a reset and oscillator circuit. A block diagram is shown in Figure 3.1.

Figure 3.1 Hardware block diagram of the FLL/PLL

3.1 The Digital Signal Processor

The used DSP is a Texas Instruments TMS320C25-50. This DSP has the following characteristics:
- 50 MHz clock frequency,
- 544 words of internal RAM,
- 64k words addressable RAM and 64k words addressable ROM,
- 16 bit address and data bus,
- 16 bit address and data bus,
- 3 external interrupts,
- 40 MIPS.

3.2 Memory

The total memory space of a TMS320C25 DSP is divided into 64 k words program memory and 64 k words data memory. The program memory contains the software, the data memory contains the variables.

In this hardware design the program memory is divided into 32 k words EEPROM for code storage and 32 k words of RAM to hold the program code while executing. The program code copies itself into RAM where program execution is continued at triple speed. The data memory is divided into on-chip and external RAM. The address space of the on-chip RAM is 1024 words of which 544 words are available for data memory, divided into 3 blocks B0, B1 and B2. The other words are for internal use and not available for the user.

![Memory Configuration Diagram](image)

**Figure 3.2 Program and Data memory configuration**

The memory configuration used is shown in Figure 3.2. To address and select the different chips, an address decoder is used which uses the signals *PS* (Program Memory Select), *DS* (Data Memory Select) and A15 (address line 15). *PS* is low when program memory is selected and *DS* is low when Data memory is selected. The truth table of the address decoder is shown in Table 3.1.

---

1 *X should be implemented as NOT(X), where X is the variable mentioned after the asterisk.
Table 3.1 The truth table of the address decoder

<table>
<thead>
<tr>
<th>A15</th>
<th>DS</th>
<th>PS</th>
<th>active output</th>
<th>selected memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>Y0</td>
<td>none</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>H</td>
<td>Y1</td>
<td>none</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
<td>Y2</td>
<td>EPROM</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>H</td>
<td>Y3</td>
<td>none</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td>Y4</td>
<td>none</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>H</td>
<td>Y5</td>
<td>Data RAM</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>L</td>
<td>Y6</td>
<td>Program RAM</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
<td>Y7</td>
<td>none</td>
</tr>
</tbody>
</table>

3.3 The waitstate generator

The waitstate generator controls the instruction cycle length. If a device (memory, I/O) is acting slow, the processor is halted for one or two clock cycles. In this way the slower acting device is able to finish the task that is being processed. All I/O operations and EPROM reads are implemented with two waitstates, RAM access is performed without waitstates.

The waitstate generator is implemented using two J-K flip-flops. Whenever EEPROM Select or I/O is chosen, two extra clock cycles are needed before READY goes high and the processor is able to continue its task.

3.4 The Input/Output-section

The Input/Output-section (I/O) consists of a selector, two A/D-converters, a D/A converter, a micro switch block and several I/O buses for the parallel and the FIFO communication. The selector selects 1 of the 7 I/O sections. The signals used are A0, A1, Read/*Write, *I/O Space and *STROBE. The first three signals are used to select a device. The other signals are used for timing. The selected devices are shown in Table 3.2.

Table 3.2 The selected devices

<table>
<thead>
<tr>
<th>Port address</th>
<th>IN-operation</th>
<th>OUT-operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Read A/D converter 1</td>
<td>Send to FIFO</td>
</tr>
<tr>
<td>1</td>
<td>Read A/D converter 2</td>
<td>Send to parallel port</td>
</tr>
<tr>
<td>2</td>
<td>Read parallel port</td>
<td>Write to D/A converter</td>
</tr>
<tr>
<td>3</td>
<td>Read data from dip-switches</td>
<td>-</td>
</tr>
</tbody>
</table>

The A/D Converter

A/D-converters are used to convert an analog input signal to a digital value, suitable for further processing by the DSP. In this application two AD779 A/D converters are used.
The specifications of these converters configured for this application are:
- Output: 14 bit, 2s complement
- Input: -5V to +5V, sample and hold
- Conversion time: 100k conversions per second

No extra chips are needed to apply this device in a DSP-board because all necessary hardware like voltage references, amplifiers etc. are implemented on chip. A high pass filter ($\tau^1=340$ Hz) is implemented to filter the input signal.

The D/A Converter

D/A converters are used to transform digital signals into analog signals. One D/A converter (AD569) is implemented on the board. The specifications of the converter configured for this application are:
- Input: 16 bit
- Output: -5V to +5V
- Settling time: 3µs

Latches etc. are built in the chip. The only extra device needed is an external voltage reference. In this application an AD588 is used. A second order low pass filter is designed on the printed circuit board. This filter has not been implemented.

The First In, First Out buffer

For communication with the Decimator/Detector board a FIFO communication bus is used. The data is sent to the other board and stored in a buffer, waiting for further processing. The FIFO communication is discussed in Chapter 6.

Dip switches

The FLUPLL board is equipped with a block of 8 dip switches. The board has a communication bus address specified by the setting of 3 dip switches. One extra dip switch is used to enable or disable the communication with the acquisition computer. Four dip switches are for future use. The use of the switches is shown in Table 3.3:

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parallel address LSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Parallel address</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Parallel address MSB</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Parallel I/O enable</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.3 The function of the dipswitches

To avoid unnecessary interrupts of the running program, addressing of the device is checked by hardware. The address specified by the acquisition computer and the FLUPLL board, specified by switches 1, 2 and 3, are compared with hardware. If both addresses are found to be equal, an interrupt will be generated.
4 Frequency Tracking with the FLL

The frequency of the carrier will change in time because of Doppler shift, frequency drift and oscillator phase noise. Keeping track on the carrier frequency is important as the whole system is tuned for one carrier frequency. Tracking is done by calculating the frequency spectrum with a Discrete Fourier Transform. The frequency bin with the highest signal power should be the carrier frequency. The difference between the desired frequency and the measured carrier frequency is used to control the voltage controlled oscillator (VCO).

4.1 The Fourier Transform

The Fourier transform converts information between the time domain and the frequency domain. The Fourier transform of an analog signal \( x(t) \) is defined as:

\[
X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt
\]  

(4.1)

where \( X(\omega) \) is a complex function. The integral exists if \( x(t) \) is piece wise continuous and if the absolute value of \( x(t) \) can be integrated.

In practice the analog signal \( x(t) \) is sampled at regular time intervals. For \( N \) samples \( x(t) \) is represented by \( x(nT) \), where \( n \) is the sample number and \( T \) is the sample period.

By assuming \( x(t)=0 \) for \( t<0 \) and for \( t>(N-1)T \) and by replacing \( x(t) \) by \( x(nT) \), \( \omega \) by \( k\Omega \) \((0<k<N)\) and \( t \) by \( nT \), Formula 4.1 becomes

\[
X(k) = \sum_{n=0}^{N-1} x(nT)e^{-j2\pi kn/N}
\]  

(4.2)

where \( \Omega=2\pi/NT \) is the fundamental frequency and \( X(k) \) is understood to represent \( X(k\Omega) \). Equation (4.2) is generally known as the Discrete Fourier Transform (DFT). Rewriting equation (4.2) gives:

\[
X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} = \sum_{n=0}^{N-1} x(n)W_N^{nk}
\]  

(4.3)
\( W_N = e^{j\frac{2\pi}{N}} \) is known as the twiddle factor. It can be seen as a \( N \)-th part of the unity circle in the complex plane.

### 4.2 The Fast Fourier Transform

Equation 4.3 is an \( N \)-point Discrete Fourier Transform (DFT). Because of the large quantity of multiplications and additions (Order \( N^2 \)), this method is not generally used. If \( N \) is a power of 2, a more sophisticated way of calculating a Fourier transform can be performed by decomposing equation 4.3 into two DFTs of length \( N/2 \) by splitting the input samples into even and odd samples.

\[
X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} = \sum_{n=0}^{N/2-1} x(2n)W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1)W_N^{(2n+1)k} \tag{4.4}
\]

Identifying the even samples \( x_1(m) \) and the odd samples \( x_2(m) \), results in

\[
X(k) = \sum_{m=0}^{N/2-1} x_1(m)W_N^{mk} + W_N^{k/2} \sum_{m=0}^{N/2-1} x_2(m)W_N^{mk} \tag{4.5}
\]

where \( W_N = W_{N/2} \). Equation 4.5 may be written as follows:

\[
X_1(k) = X_{11}(k) + W_N^kX_{12}(k) \tag{4.6}
\]

By comparing equation 4.6 with equation 4.4 it is seen that \( X_{11} \) and \( X_{12} \) are reduced to \( N/2 \)-point DFTs. Repeating this rewriting, eventually results in \( N/2 \) 2-point DFTs. The 2 point DFT is a basic operation, called butterfly because of its shape in graphical representation (Figure 4.3).

![Figure 4.3 Representation of the Decimation in Time (DIT) radix 2 butterfly.](image)

The formulas to perform a 2-point DFT are:

\[
X(k) = X_1(k) + W_N^kX_2(k)
\]

\[
X(k+N/2) = X_1(k) + W_N^{k+N/2}X_2(k) \tag{4.7}
\]

This butterfly can be used to recompose the Fourier Transform of \( N \) discrete signals. A decomposition is called a stage, and the total numbers of stages is given by \( M = \log_2 N \).
Calculating a FFT results in \( N \) values which represent the power of the signal \( x(t) \) in a frequency interval which is called a bin. The size of one bin is:

\[
\Delta f = \frac{1}{NT} = \frac{f_s}{N}
\]  

(4.8)

The spectrum will cover a frequency interval of \( \Delta f \cdot N = f_s \) Hz.

The input data in Figure 4.4 are placed in bit reversed order, e.g. the sample \( x(1) \) is placed on the location of sample \( x(4) \) because 4 (binary representation: 100) is the bit reversed number of 1 (binary representation: 001).

### 4.3 Window functions to be used

A run of \( N \) samples is used for a FFT. When this run is not adapted at the beginning and at the end, the sharp edges in the time domain will return as high frequency components in the frequency domain. Sharp edges are the result of multiplying an infinite run of samples by a square window function in time. The high frequency components will result in leakage of information to other bins in the frequency domain. For this reason sharp edges in the window function should be avoided.

One of the most common solutions to solve the problem of sharp edges is the Hamming window:

\[
h(n) = 0.54 - 0.46 \cos(2\pi/N), \quad n=0,1,2,\ldots,N-1
\]

(4.9)

Almost the same result can be obtained by using the Hanning window:

\[
h(n) = 0.5 - 0.5 \cos(2\pi n/N), \quad n=0,1,2,\ldots,N-1
\]

(4.10)

The time and frequency response of the Hamming, Hanning and square-shaped windows are shown in Figure 4.5–4.10.
In these figures can be seen that a square window has worse characteristics than a Hanning...
or Hamming window. A Hanning window has lower first lobes, but these remain at the same level. This window is chosen because the side lobes of the Hamming window decrease with increasing frequency distance from the center frequency.

### 4.4 Choosing the number of samples for the FFT.

Choosing a number of samples for the FFT is done by trading off different contributions to the carrier phase drift. Choosing a large N will implement a large sample acquisition time and a larger calculation time. Moreover, the Doppler shift may eventually cause the carrier to shift through several bins. However, the frequency bins will be smaller. An optimum N must be determined, taking these factors into consideration. Table 4.5 shows the various contributions to the carrier frequency drift for the different beacons.

#### Table 4.5 Contributions to carrier frequency drift

<table>
<thead>
<tr>
<th>Beacon</th>
<th>B₀</th>
<th>B₁</th>
<th>B₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doppler Shift</td>
<td>0.0013</td>
<td>0.0022</td>
<td>0.0031</td>
</tr>
<tr>
<td>Satellite &amp; LO freq.</td>
<td>0.17</td>
<td>0.29</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Using a sample rate of 12.288 kHz and different number of samples the Table 4.6 can be drawn up for beacon B₀:

#### Table 4.6 Effect of the number of samples to the acquisition time and frequency drift

<table>
<thead>
<tr>
<th>Bin size</th>
<th>12</th>
<th>6</th>
<th>3</th>
<th>1.5</th>
<th>Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample acquisition time</td>
<td>83</td>
<td>166</td>
<td>333</td>
<td>670</td>
<td>ms</td>
</tr>
<tr>
<td>Total drift over acquisition time</td>
<td>0.014</td>
<td>0.028</td>
<td>0.057</td>
<td>0.114</td>
<td>Hz</td>
</tr>
</tbody>
</table>

The frequency drift of the signal is small compared to the bin size. The bin size is important, because a stable input frequency will result in a better performance of the PLL. The smallest possible bin size should be the best. However, at a sample rate of 12.288 kHz, this would result in 8192 samples, leading to a huge number of multiplications and high memory usage. Changing the sample rate to 6144 Hz will result in 4096 samples. By reducing the sample rate, the covered part of the spectrum will decrease.

Using a bin size of 3 Hz and a sample rate of 6144 Hz will result in an equilibrium between calculation time and accuracy of the FLL.

While testing the programs with the development system, the available memory is limited to 4000h. This results in a maximum number of samples of 1024.
5.1 A conventional analog PLL

A Phase-Locked Loop (PLL) is a special kind of feedback control system that keeps the phase difference between the input and output signal as small as possible. A basic analog PLL looks like Figure 5.1. It consists of a phase detector (PD), a loop filter (LF) and a voltage controlled oscillator (VCO).

![Figure 5.1 A basic analog PLL](image)

The analog phase detector compares the phase \( \phi_i(t) \) of a periodic input signal with the phase \( \phi_o(t) \) of the VCO output signal \( x_o(t) \). The output of the phase detector is a signal which is proportional to the phase difference between the two phase detector input harmonic signals. The output of the PD is filtered by the loop filter. The loop filter output signal will adjust the frequency of the VCO in a direction that reduces the phase difference between the input signal and the VCO output signal. The basic principles of a PLL have been extensively discussed in Gardner [5] and in Op Den Camp [1]

The advantage of a digital PLL (DPLL) are the parameters that can easily be controlled. Basically DPLLs can be divided into two categories:
* Those which are built with gates, counters, registers etc. These are available in IC's such as 4046 and LM565.
* Those which uses an A/D converter and an algorithm running on a microprocessor. To implement a PLL on a DSP it has to be implemented in software.

Two different algorithms are discussed in Op Den Camp [1]. Concluded is that the best PLL for this application is a Linear All Digital Phase-Locked Loop. This PLL is the
digital equivalent of the model of the analog Phase-Locked Loop.

5.2 The All Digital Phase-Locked Loop

At first, Digital Phase-Locked Loops were derived from analog PLL by replacing the different components by their digital equivalent. In this way some disadvantages of the analog PLL remained present. The most important disadvantages are:
- Generation of high order frequency components by the phase detector
- non-linear behaviour
- time-variant behaviour

Hagiwara [3] proposed a Linear All Digital Phase-Locked Loop (L-ADPLL), based on a zero-order loop filter which results in a first-order PLL. In 1989 Shayan and Le-Ngoc [4] proposed a second-order L-ADPLL, shown in Figure 5.2. The implemented loop filter is a first-order filter which results in a second-order PLL. A second-order PLL has better characteristics.

![L-ADPLL proposed by Shayan](image)

The digitized input signal is split and one part transformed by a Hilbert transform (H(z)), which shifts all frequencies $90^\circ$. Because a Hilbert transform takes some time, the other part has to be delayed by $\Delta T$.

Assume $x_i(t)$ is defined as:

$$x_i(t) = A_i \sin(\omega_i t + \theta_i(t))$$  \hspace{1cm} (5.1)

The sampled (and delayed) signal $X_i(k)$ will be:

$$X_i(k) = A_i \sin[\omega_i kT + \theta_i(kT)]$$  \hspace{1cm} (5.2)

and the Hilbert transformed signal $Y_i(k)$ will be

$$Y_i(k) = A_i \cos[\omega_i kT + \theta_i(kT)]$$  \hspace{1cm} (5.3)

From (5.1) and (5.2) the input signal argument $\phi_i(k)$ can be calculated:

$$\phi_i(k) = \tan^{-1} \frac{X_i(k)}{Y_i(k)} = [\omega_i kT + \theta_i(kT)] \mod 2\pi$$  \hspace{1cm} (5.4)

The phase detector subtracts $\phi_o(k)$ from $\phi_i(k)$. The Difference is called $\phi_d(k)$. Since $\phi_i(k)$
and $\phi_o(k)$ are periodic functions, the subtraction does not always produce the right phase difference. Correction is performed by the mod [-\pi, \pi] block.

Figure 5.3 shows two situations, a phase lead (situation I) and a phase lag (situation II)

![Graphs showing phase lead and phase lag](image)

**Figure 5.3  Behaviour of the mod[-\pi, \pi] circuit.**

The transfer function of the first-order loop filter is:

\[
H_{LF}(z) = \frac{C_1}{z-1} + C_2
\]  

(5.5)

The transfer function of the DVCO is:

\[
H_{DVCO}(z) = \frac{1}{z-1}
\]  

(5.6)

The open loop transfer function is:

\[
H_{OL}(z) = H_{LF}(z) \cdot H_{DVCO}(z) = \frac{C_2(z-1) + C_1}{(z-1)^2}
\]  

(5.7)

Thus the Closed loop transfer function $H(z)$ is:

\[
H(z) = \frac{\phi_o}{\phi_i} = \frac{H_{OL}(z)}{1+H_{OL}(z)} - \frac{C_2(z-1) + C_1}{(z-1)^2 + C_2(z-1) + C_1}
\]  

(5.8)

This is a second order transfer function of the Linear A-DPLL.

Using the forward difference method ($z-1$) can be replaced by $sT$ [12]. Using this method Equation 5.8 can be transformed to the transfer function of an analog second order PLL:

---

22 Optimization and testing of a digital FLL/PLL
If \( f_n \ll f_s \), \( C_1 \) and \( C_2 \) can be calculated from 5.8 and 5.9:

\[
C_1 = 2\xi \omega_n T \\
C_2 = \omega_n^2 T^2
\]  

(5.10)

The steady state error for different inputs can be calculated using the final value theorem [12]:

\[
\epsilon_{ss} = \lim_{z \to 1} (1-z) (1-H(z)) \phi_i(z)
\]  

(5.11)

The z-transforms of the different inputs are:

\[
\phi_i(z) = \frac{z}{z-1} \Delta_i \quad \text{Phase step of } \Delta_i \text{ rad}
\]

\[
\phi_i(z) = \frac{zT}{(z-1)^2} \Delta_2 \quad \text{Phase ramp of } \Delta_2 \text{ rad/s}
\]

\[
\phi_i(z) = \frac{zT^2}{(z-1)^3} \Delta_3 \quad \text{Phase acceleration of } \Delta_3 \text{ rad/s}^2
\]

Using equation 5.11 to calculate the steady state error gives respectively:

\[
\epsilon_{ss} = 0 \quad \text{Phase step of } \Delta_i \text{ rad}
\]

\[
\epsilon_{ss} = 0 \quad \text{Phase ramp of } \Delta_2 \text{ rad/s}
\]

\[
\epsilon_{ss} = \frac{\Delta_3}{\omega_n^2} \quad \text{Phase acceleration of } \Delta_3 \text{ rad/s}^2
\]

These results corresponds to the steady state errors of an analog second order PLL.

Because the steady state error of a phase ramp (frequency step) is 0, the hold-in-range of the PLL will be large. This can be verified by calculating the DC Loop gain:

\[
G_{DC\text{loop}} = \lim_{z \to 1} H_{OL}(z) = \lim_{z \to 1} \frac{C_2(z-1) + C_1}{(z-1)^2} \to \infty
\]  

(5.12)

For a linear phase detector the hold-in-range is proportional to the DC Loop Gain.
Stability can be reached by placing the poles of the closed loop inside the unit circle in the z-plane. The poles of $H_{CL}$ are:

$$z_{1,2} = \frac{2-C_2 \pm \sqrt{(C_2-2)^2-4(C_1-C_2+1)}}{2}$$  \hspace{1cm} (5.13)

Positioning of $z_{1,2}$ inside the unit circle gives:

$$2C_2-4 < C_1 < C_2, \quad C_1 > 0$$  \hspace{1cm} (5.14)

These constants can be calculated with Formula 5.10. Using $\zeta=0.707$ and $\omega_n=17.2\pi$ rad/s (17 Hz) will result in $C_1=75.56\cdot10^{-6}$ and $C_2=12.29\cdot10^{-3}$. These values appear to meet the constraints from 5.14.

The free-running frequency should be 124.952 kHz. Using the subsampling theorem, this results in a free running frequency of 1536 Hz. This frequency is represented by $C_0$ and is determined by the number of samples per period (8 samples/period).

$$C_0 = \frac{2^{16}}{f_s/1536}$$  \hspace{1cm} (5.15)

### 5.3 Measurement results of the ADPLL

Several characteristics of the ADPLL are measured using the HP-3325A oscillator. Measurements with noisy signals are carried out using the satellite signals of the beacon receiver at 40 GHz.

**Measurements without noise**

The response of the phase error $\phi_e$ is simulated with MatLab. Using an input step of 100Hz resulted in Figure 5.4. The phase error is reduced to zero in 13 $\mu$s. The maximum phase error is 31 degrees. The step response is measured and matches the simulated step.

*Figure 5.4 Phase error $\phi_e$ as a result of a frequency step of 100 Hz, using $\zeta=0.707$ and $f_n=100$ Hz*
response. Assuming the PLL is locked, coherent detection is performed correctly. The data shown in Figure 5.5 is measured by varying the amplitude of the co-polar input signal. As shown in Equation 2.6, $I_c$ should be proportional to the amplitude of the co-polar input signal, $Q_c$ should equal 0. The high frequency components, which are the result of the multiplications in the coherent detection procedures, are not removed in the FLL/PLL software because a digital filter is not implemented. A Low-Pass RC filter ($\tau = 550$Hz) will remove the high frequency components after the D/A-converter has converted the digital output to an analog signal.

![Figure 5.5 Ic and Qc as a function of the amplitude of the Co-polar input signal](image)

As shown in Figure 5.5, $I_c$ is linear with the co-polar input signal.

Varying the phase angle between the co- and cross-polar input signal results in varying $I_x$ and $Q_x$ signals. The phase angle between the co- and crosspolar is determined (formula 2.8 and 2.9) by the calculation of the arctangent of the quotient of the $I$ and $Q$ signals. The result is shown in Figure 5.6.
As shown in Figure 5.6, the phase angle equals the input phase difference. The output range is limited by the arctangent circuit to $[-90^\circ, 90^\circ]$.

**Measurements with noise**

For measurements with noise the ITALSAT-satellite beacon signal received at 40 GHz is used because the noisy signal is difficult to simulate. The S/N ratio is dependent on rain attenuation etc. It is also possible to change the S/N ratio manually by using an attenuator in the wave guide.

---

1 Attention should be payed to unwanted noise in the system. Unwanted noise in the digital FLL/PLL is added by the digital part to the analog part. Moreover, the FLL/PLL and the PC-clock signal add noise at a high level to the analog PLL receivers. Keeping a distance of about 3 meter is a solution to the last problem. The first problem might be solved by redesigning the digital FLL/PLL printed circuit board.
The 40 GHz signal is down converted to 10 MHz. This signal is used as input for the analog PLL, which down converts the co-polar to 125 kHz. The analog PLL is used because a VCO for use with the FLL was not available.

Kamperman [13] added an amplifier and a bandpass filter to the analog PLL. The amplifier adapts the 125 kHz signal amplitude to the input range (-5V/5V) of the A/D converter. The bandpass filter prevents aliasing. The transfer characteristic of the amplifier/bandpass filter in the down converter is shown in Figure 5.8.

![Amplifier/Bandpass Transfer Characteristic](image)

**Figure 5.8** The amplitude transfer-characteristic of the amplifier / bandpass filter section

The power spectrum of the 10 MHz signal with noise looks like Figure 5.9. The mainlobe is the noise which is mainly generated by the amplifiers and local oscillators. The smaller lobe on top of the main lobe is the received satellite signal. Whenever the satellite signal is attenuated by rain etc., the small lobe will shrink. The main lobe will remain the same because this lobe is the result of noise generated by the receiversystem.

![Power Spectrum](image)

**Figure 5.9** An impression of the shape of the powerspectrum at 10 MHz of a satellite signal

The C/N ratio of the signal at 10 MHz is determined using a power meter. The power meter is able to determine the signal power with noise of the powerspectrum shown in
Figure 5.9. When the satellite signal is removed using a piece of absorbing material mounted on the antenna, only the noise will remain.

Table 5.1 The measured power densities

<table>
<thead>
<tr>
<th>Signal</th>
<th>Power [dBm]</th>
<th>Power [W]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S+N)</td>
<td>-3.55</td>
<td>441·10^-6</td>
</tr>
<tr>
<td>(N)</td>
<td>-5.3</td>
<td>295·10^-6</td>
</tr>
</tbody>
</table>

This will result in a C/N ratio of:

\[
\frac{C}{N} = \left(\frac{(S+N)-(N)}{(N)}\right) = 0.5 \pm 3\text{dB}
\] (5.16)

The C/kT, which is constant in the whole system, is determined using the noise bandwidth of the noise signal at 10 MHz. The noise bandwidth by the bandpass filter in the 10 MHz IF-strip. The amplitude-transfer function of the filter is shown in Figure 5.10.

The noise bandwidth of the bandpass filter is calculated using the following Formula [13]:

\[
B_N = \frac{\int_{0}^{\infty} |H(f)|^2 df}{|H(f)|_{\text{max}}^2}
\] (5.17)

Unfortunately only the transfer characteristic is known, so Formula 5.17 has to be evaluated numerically, which results in a noise bandwidth of 44.3 kHz.
Using this bandwidth the \( C/kT \) can be calculated:

\[
\frac{C}{kT} = \frac{C}{N} \cdot B_N = 0.5 \cdot 44.3 \text{ kHz} \approx 43.5 \text{ dBHz}
\quad (5.18)
\]

The noise bandwidth of the bandpass filter/attenuator is also evaluated numerically. The noise bandwidth of this filter is 1530 Hz. Using the \( C/kT \) and the noise bandwidth of the bandpass filter/attenuator, the \( C/N \) at the input of the digital FLL/PLL can be calculated:

\[
\frac{C}{N} = \frac{C}{kT} \cdot \frac{1}{B_N} \approx 43.5 \text{ dBHz} - 10 \cdot \log(1530) = 11.6 \text{ dB}
\]

The analog PLL satellite receiver, which is currently used for the propagation experiments, will be used as a reference. The output of these receivers is registrated on a chart. Calibration is done using a HP-synthesizer which generates a 10 MHz signal. This signal from the synthesizer is added to the 10 MHz signal as shown in Figure 5.11. In this way the synthesizer is able to simulate the satellite carrier, assuming that the noise is only added by the circuits after the attenuator in the wave guide. The received satellite signals with the dish antenna are removed with a piece of absorbing material.

![Figure 5.11 The setup of the receiver for calibration experiments](image)

The chart can be calibrated by changing the amplitude of the simulated satellite signal with the synthesizer.

The dynamic inputrange of the DPLL is calculated using the resolution of the A/D converter:

\[
\text{Dynamic inputrange} = 10 \cdot \log(2^{14}) = 42.1 \text{ dB}
\quad (5.20)
\]

Due to crosstalk and non-linearity, the resolution of the A/D converter is smaller. The linearity is determined using the output of the analog PLL as a reference. This resulted in the following characteristic.

From Figure 5.12 it is shown that the characteristic of the digital FLL/PLL remains linear from a \( C/N \) ratio at the input of the digital FLL/PLL of 11.6 to -10.5 dB. At lower \( C/N \) ratios the analog PLL compresses the signal.

If the input frequency is varied without changing the input level, the detected \( I_c \) will change. This should not happen as shown in equation (2.4) and (2.5). An explanation is the non-ideal transfer function of the amplifier/bandpass filter in the 10 MHz IF-strip.
non-ideal transfer function of the amplifier/bandpass filter in the 10 MHz IF-strip.

Figure 5.12 The linearity of the FLL/PLL calibrated using the analog PLL

The noise behaviour of the ADPLL is determined by measuring the phase error after the loopfilter as a function of the C/N ratio at the input of the digital FLL/PLL. For simulating a worse signal to noise ratio, an attenuator in the waveguide is used.

Figure 5.13 Phase angle $\phi_e$ as function of the input C/N ratio

The phase error is an indication for the lock status. A phase error of 9 degrees is an indication the loop might be out of lock. Using this criterion, the loop falls out of lock when the C/N is about -15 dB. The analog PLL goes out of lock at -12 dB.
Communication between the different devices

The FLL/PLL is able to communicate with a Personal Computer and with the decimator/detector. For communication with the PC, a modified Centronics interface is used. Transmitting data to the Decimator/detector is done with a single direction communication buffer.

6.1 The communication between the FLL/PLL and the PC

The parallel communication is based on a standard Centronics parallel interface. This interface is a standard device on every personal computer. The Centronics interface consists of 12 digital outputs and 5 digital inputs. The Centronics interface on a PC has a 25 pin female sub-D connector. The pinning shown in Figure 6.1

<table>
<thead>
<tr>
<th>pin</th>
<th>I/O</th>
<th>original function</th>
<th>pin</th>
<th>I/O</th>
<th>original function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>Strobe</td>
<td>10</td>
<td>I</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td>Data 0</td>
<td>11</td>
<td>I</td>
<td>Busy</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>Data 1</td>
<td>12</td>
<td>I</td>
<td>Out of paper</td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>Data 2</td>
<td>13</td>
<td>I</td>
<td>Select</td>
</tr>
<tr>
<td>5</td>
<td>O</td>
<td>Data 3</td>
<td>14</td>
<td>O</td>
<td>Auto Feed</td>
</tr>
<tr>
<td>6</td>
<td>O</td>
<td>Data 4</td>
<td>15</td>
<td>I</td>
<td>Error</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>Data 5</td>
<td>16</td>
<td>O</td>
<td>Reset</td>
</tr>
<tr>
<td>8</td>
<td>O</td>
<td>Data 6</td>
<td>17</td>
<td>O</td>
<td>Select</td>
</tr>
<tr>
<td>9</td>
<td>O</td>
<td>Data 7</td>
<td>18-25</td>
<td></td>
<td>Ground</td>
</tr>
</tbody>
</table>

The 12 available output lines are used for communication from the PC to the device. Three lines are used to select a device. One line is used to generate an interrupt for the device and eight lines are used to transmit data. The five available input lines are used to receive
and eight lines are used to transmit data. The five available input lines are used to receive
data from the device. One line is used to get an interrupt from the device, the other 4 lines
are used to receive data. The redefined Centronics port is shown in Figure 6.2.
Redefining the Centronics interface in this way will create the possibility for the master
(the PC) to communicate with 8 different slaves.

**Table 6.2 Pinout of the redefined Centronics interface.**

<table>
<thead>
<tr>
<th>pin</th>
<th>I/O function</th>
<th>pin</th>
<th>I/O function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>10</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>IRQ on device</td>
<td></td>
<td>IRQ on PC</td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td>11</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Data Out 0</td>
<td></td>
<td>Data In 3</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>12</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Data Out 1</td>
<td></td>
<td>Data In 2</td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>13</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Data Out 2</td>
<td></td>
<td>Data In 1</td>
</tr>
<tr>
<td>5</td>
<td>O</td>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Data Out 3</td>
<td></td>
<td>Device Select 0</td>
</tr>
<tr>
<td>6</td>
<td>O</td>
<td>15</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Data Out 4</td>
<td></td>
<td>Data In 0</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>16</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Data Out 5</td>
<td></td>
<td>Device Select 1</td>
</tr>
<tr>
<td>8</td>
<td>O</td>
<td>17</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Data Out 6</td>
<td></td>
<td>Device Select 2</td>
</tr>
<tr>
<td>9</td>
<td>O</td>
<td>18-25</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Data Out 7</td>
<td></td>
<td>Ground</td>
</tr>
</tbody>
</table>

**The communication protocol**

Besides a good hardware definition of the communication system, a protocol has to be
designed. The used protocol is the result of the following constraints:
- Minimum processor time occupation
- ASCII commands and answers must be possible
- transmission error detection and recovery

Minimum processor occupation is achieved by using interrupt-driven procedures. By using
Cyclic Redundancy Check (CRC) for error detection and timers for time-out detection,
errors can be detected and recovered.

The designed communication packet sent by the PC has the following structure:

```
SOH  LEN  D₁  D₂ ... D_{len-4}  CRCₙ  CRC₁  EOT
```

The meaning of the fields are:
- **SOH**: Start of Header
- **LEN**: Length of the packet in bytes (not including SOH)
- **D₁,...,Dₙ**: Data fields
- **CRCₙ, CRC₁**: 16 bit CRC-word from SOH to D_{len-4}
- **EOT**: End of Transmission

Every character transmitted by the PC is coded as an 8 bit wide word.
A Communication packet send by a device has the following structure:

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN₂</th>
<th>LEN₁</th>
<th>LEN₀</th>
<th>( D_{1h} )</th>
<th>( D_{1l} )</th>
<th>( D_{2h} )</th>
<th>( D_{2l} )</th>
<th>...</th>
<th>( D_{LEN₂} )</th>
<th>( D_{LEN₁} )</th>
<th>CRC₂</th>
<th>CRC₁</th>
<th>CRC₀</th>
<th>EOT</th>
</tr>
</thead>
</table>

The meaning of the fields are:
SOH: Start of Header
LEN₂, LEN₁: Length (\( \text{LEN₂} 	imes 256 + \text{LEN₁} \times 16 + \text{LEN₀} \)) of the packet in nibbles (not including SOH)
\( D_{1h}, D_{1l}, ..., D_{nh}, D_{nl} \): Low and High data fields
CRC₂, CRC₁: 16 bit CRC-word from SOH to \( D_{\text{LEN}_2-8} \)
EOT: End of Transmission
Every character transmitted is 4 bit wide.

### 6.2 Solutions for Error Detecting

Error detection and recovery is done by using time-outs and Cyclic Redundancy Check (CRC) on both the master and the slave. Using a Time-Out signal will prevent a device waiting for an answer. Cyclic redundancy check is one of the most common error detecting codes. It is based on the use of polynomials. At both sides a generator polynomial has been agreed upon. The algorithm used in this program is based on a lookup table and was published in Byte [8].
6.3 The communication between the FLL/PLL and the decimator/detector

The First In, First Out communication passes the in-phase and quadrature demodulated signals of both channels from the FLL/PLL to the decimator/detector. This one way communication is based on a FIFO buffer. The data are transmitted in parallel. Handshaking is done with a clock and ready signal.

The specification of the FIFO data format is as follows:

<table>
<thead>
<tr>
<th>word</th>
<th>bit 0</th>
<th>bit 1</th>
<th>bit 2</th>
<th>bit 3</th>
<th>bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$I_x$ bit 12</td>
<td>$I_x$ bit 13</td>
<td>$I_x$ bit 14</td>
<td>$I_x$ bit 15</td>
<td>status data available</td>
</tr>
<tr>
<td>2</td>
<td>$I_x$ bit 8</td>
<td>$I_x$ bit 9</td>
<td>$I_x$ bit 10</td>
<td>$I_x$ bit 11</td>
<td>status PLL</td>
</tr>
<tr>
<td>3</td>
<td>$I_x$ bit 4</td>
<td>$I_x$ bit 5</td>
<td>$I_x$ bit 6</td>
<td>$I_x$ bit 7</td>
<td>status FLL</td>
</tr>
<tr>
<td>4</td>
<td>$I_x$ bit 0</td>
<td>$I_x$ bit 1</td>
<td>$I_x$ bit 2</td>
<td>$I_x$ bit 3</td>
<td>undefined</td>
</tr>
<tr>
<td>5</td>
<td>$Q_x$ bit 12</td>
<td>$Q_x$ bit 13</td>
<td>$Q_x$ bit 14</td>
<td>$Q_x$ bit 15</td>
<td>undefined</td>
</tr>
<tr>
<td>6</td>
<td>$Q_x$ bit 8</td>
<td>$Q_x$ bit 9</td>
<td>$Q_x$ bit 10</td>
<td>$Q_x$ bit 11</td>
<td>undefined</td>
</tr>
<tr>
<td>7</td>
<td>$Q_x$ bit 4</td>
<td>$Q_x$ bit 5</td>
<td>$Q_x$ bit 6</td>
<td>$Q_x$ bit 7</td>
<td>undefined</td>
</tr>
<tr>
<td>8</td>
<td>$Q_x$ bit 0</td>
<td>$Q_x$ bit 1</td>
<td>$Q_x$ bit 2</td>
<td>$Q_x$ bit 3</td>
<td>undefined</td>
</tr>
<tr>
<td>9</td>
<td>$I_x$ bit 12</td>
<td>$I_x$ bit 13</td>
<td>$I_x$ bit 14</td>
<td>$I_x$ bit 15</td>
<td>undefined</td>
</tr>
<tr>
<td>10</td>
<td>$I_x$ bit 8</td>
<td>$I_x$ bit 9</td>
<td>$I_x$ bit 10</td>
<td>$I_x$ bit 11</td>
<td>undefined</td>
</tr>
<tr>
<td>11</td>
<td>$I_x$ bit 4</td>
<td>$I_x$ bit 5</td>
<td>$I_x$ bit 6</td>
<td>$I_x$ bit 7</td>
<td>undefined</td>
</tr>
<tr>
<td>12</td>
<td>$I_x$ bit 0</td>
<td>$I_x$ bit 1</td>
<td>$I_x$ bit 2</td>
<td>$I_x$ bit 3</td>
<td>undefined</td>
</tr>
<tr>
<td>13</td>
<td>$Q_x$ bit 12</td>
<td>$Q_x$ bit 13</td>
<td>$Q_x$ bit 14</td>
<td>$Q_x$ bit 15</td>
<td>undefined</td>
</tr>
<tr>
<td>14</td>
<td>$Q_x$ bit 8</td>
<td>$Q_x$ bit 9</td>
<td>$Q_x$ bit 10</td>
<td>$Q_x$ bit 11</td>
<td>undefined</td>
</tr>
<tr>
<td>15</td>
<td>$Q_x$ bit 4</td>
<td>$Q_x$ bit 5</td>
<td>$Q_x$ bit 6</td>
<td>$Q_x$ bit 7</td>
<td>undefined</td>
</tr>
<tr>
<td>16</td>
<td>$Q_x$ bit 0</td>
<td>$Q_x$ bit 1</td>
<td>$Q_x$ bit 2</td>
<td>$Q_x$ bit 3</td>
<td>undefined</td>
</tr>
</tbody>
</table>

The hardware at the transmitter side consists of a 74F245 output buffer. At the receiver, a First In, First Out (FIFO) buffer (74F225) is used. This buffer is 5 bit wide and is able to contain 16 words. The use of the FIFO eliminates the need for special synchronisation because the FLL/PLL is able to fill the FIFO independent of what the decimator/detector is doing. The Decimator/Detector receives an interrupt from the FIFO if the buffer is full (all data are available).
7

Implementation of the FLL/PLL software

7.1 Function description of and improvements made in the FLL/PLL program

The function description of the procedures and the improvements made in the program of Franken [9] are discussed in this paragraph. The FLL/PLL program consists of a main loop program, 2 interrupt driven procedures, a parallel command interpreter and the FLL algorithm. When not executing any (interrupt)procedure, the main loop program is executed.

The improvements are discussed in the following chapters, using the flow diagrams of the FLL/PLL program.

The interrupt 0 routine

This routine is called on every interrupt 0. The A/D converters, which are initiated by the 12.288kHz clock, generate this interrupt when the analog signals are converted. The samples are read and adapted to Q15 format. The PLL procedures are called for. Every sample is evaluated by the PLL algorithm.

The PLL algorithm that is used is an exact implementation of the block diagram shown in Figure 5.2. An ideal Hilbert transform will shift every frequency component 90°, however, an ideal Hilbert transform takes an infinite number of samples. To prevent this, a finite Hilbert transform is implemented which uses 35 samples. This will result in a delay equal to 17 samples. A table in memory is used for storage of the Hilbert constants. The time delay is compensated with a delay of the Xj(k) signal. The two signals, which represent a sine Xj(k) and a cosine Yj(k), are divided. The result is an index to the arctangent table, which is only present from 0° to 45°. Values from 45° to 90° are calculated by its mirror value. By using the signs of X(k) and Y(k), angles in other quadrants can be calculated (table 7.1).
The result of the calculation represents a phase angle. The Phase detector is implemented by subtracting the desired angle from the measured angle. This implementation will result in a linear phasedetector characteristic.

The loop filter is implemented using a normal digital first-order filter with two constants $C_1$ and $C_2$. These constants are normalized to 16 bits by multiplying them with $2^{16}$.

The Digital Voltage Controlled Oscillator adds $C_0$ to the filtered signal. The calculation of the cosine of the output angle $\phi_o$ is done using a cosine-table. The sine of the phase angle $\phi_o$ can also be calculated with the cosine-table.

The interrupt routine also collects the samples for use with the FFT algorithm and stores every n-th (n=2) sample in bit reversed order in the memory. If a memory bank is full, a flag is set and another memorybank is filled. There are two memory banks for the storage of the samples.

Changes made to the interrupt 0 procedures and the called procedures are:

- The sequence of the PLL procedures is changed. The sequence was: output routines, Hilberttransfrom, delaylines, demodulation, lowpass filters, phasethdetection, Phase-Locked Loop algorithm and is changed to: delaylines, Hilberttransform, phase detection, Phase-Locked Loop algorithm, demodulation, low pass filter, output routines.
- An extra procedure for testing is added (OUTPUT2). This procedure transfers the values of a specified memory location to the D/A converter. This procedure is not shown in the flowchart because it is for testing purposes only. In the finite program the D/A converter should be used for VCO control.
- The possibility to change the sample rate of the data stored in the FFT memory bank is added.
- A PLL lock detection is added. The absolute value of the phase error is added to a register. Every 1024 samples this register is read and compared with a threshold value. If this value is exceeded, the PLL is said to be out of lock. The PLL is reset. The interrupt 0 procedure is placed in block BO. This improves the performance of execution.
The Main loop

If the program is started, all variables are initialized. The main program consists of a loop which controls whether the FFT-memory bank is full and whether a parallel command is received. If the FFT-memory bank is full the FFT algorithm is called, the spectrum of the signal is calculated and the carrier is detected. If a parallel command is received the command interpreter is called.

The FFT algorithm consists of two separate parts. In the first part the memory banks are filled, as described in the previous chapter. In the second part of the FFT algorithm is started, all data are collected. In this case the memory bank is full and the calculation of the FFT will start. The FFT algorithm is implemented using the Butterfly algorithm [7] described in Chapter 4.2. A procedure which calls for the butterfly routine arranges the decomposition of the data.

Calculation of the FFT is done in place. Extra memory locations are required for the imaginary part of the calculated results.

When finished calculating the FFT, all values are squared and stored in a separate memory bank which represents the frequency spectrum. While squaring, the largest value and its location are searched. The location of this value determines the frequency of the carrier.

Dependent on the type of VCO, the calculated carrier frequency can be the input for a control loop with P or PI characteristic and with a desired nominal frequency of 124.519 kHz. A control loop for use with the VCO is not yet implemented.

Changes made to the main loop and the called procedures are:
- The Decimation in Time(DIT) Radix-2 Butterfly is replaced by the butterfly algorithm from [7].
- The procedures to calculate the spectrum and to search the carrier are optimized by rewriting several parts.
- The parallel command interpreter is replaced by the interpreter designed by Kooistra [6] because this procedure is more flexible.

Figure 7.2 The flowchart of the mainloop

The interrupt 2 routine

This procedure is called when a parallel command is sent or received. A status variable is used. At first the PAR_status variable is 0, during receiving PAR_status equals 1 and after the successful completion of PAR_status equals 2. During transmission PAR_status equals 3, as initialized by the procedure transmit. After transmission PAR_status is set to zero.

There are no changes made to this procedures.
General

Some improvements are made to the variable declaration and look-up tables. The tables, which remain constant while debugging the code, are stored in a separate file which is linked with the source code. This improves readability and compilation time. The variables are sorted and placed in one memory block. Due to this improvement the memory usage is more efficient and the usage of memory pages is prevented.

7.2 Unsolved problems

When separated, the software parts (FLL/communication and PLL) function correctly. When the parts of the software are combined, the FFT is not correctly calculated. This is the result of lack of processing power. When the sample frequency is decreased to 5 kHz, the combined software parts function correctly. The solution might be found in two possible explanations of the problem:

- The development system is slowing down the execution of the program. This problem can be solved using the FLL/PLL without development system by programming the EPROMs with the FLL/PLL software.
- The interrupt 0 procedure is too large. By optimizing the code or by increasing the processing power by using a faster version of the TMS320C25, this problem can be solved.

7.3 Memory Allocation

As shown in Chapter 3, the RAM of the TMS320C25 is divided into on-chip and off-chip memory. On-chip memory is addressed by Oh-3FFh, off-chip is addressed by 400h-2000h. 544 words of the 1000 addressable words are available to the user. These words are divided into three memory blocks: B0 (200h-2FFh), B1(300h-3FFh) and B2(60h-7Fh). Other memory blocks are for internal use by the processor. It is possible to execute a procedure from block B0. The advantage of this configuration is the fast execution of the procedure placed in this block, disadvantage is the loss of 100h memory space.

In the FLL/PLL software, block B0 is configured as program memory. The INTO routines copied into block B0 when the program is started. Block B1 is used for storing variables. The total number of variables is 88 (58h) and will fit into the lower half of block B1 (300h-37Fh). The upper half (380h-3FFh) is used for the three delay lines and the Hilbert transform constants.

Block B2 is used for the stack. The function of the stack is saving variables and the processor state when executing an interrupt.

The memory allocation is defined in a .CMD file (Appendix D). The assembler source code contains only names which represent the memory allocations. The command file is used by the linker to connect these names to memory locations.
8 Conclusions and recommendations

8.1 Conclusions

- The Frequency-Locked Loop for a digital beacon receiver is implemented using a 1024-point FFT. The reduced size of the number of points is the result of the lack of memory in the DSP development system.

- The Phase-Locked Loop is implemented using an All-Digital Phase-Locked Loop. The characteristics of this type of Phase-Locked Loop proved to be superior to those of the analog counterpart. Using $f_n=100$ Hz and $\zeta=0.707$ results in a hold-in range of 2500 Hz and a lock-in range of 600 Hz. The C/N ratio at the input of the FLL/PLL may decrease to -14.5 dB before loss of lock occurs.

- Communication between the PC and the FLL/PLL was tested and improved. Another communication procedure allows free programming of the command set. Using the current setup, data representing the discrete spectrum of the signal are sent every second to the PC.

- Coherent detection of the co- and crosspolar signals results in the in-phase and quadrature signals. It is possible to measure the phase difference between the co and crosspolar signal and the amplitudes of these signals.

- An accurate oscillator has been built, which produces the sampling clock.

- The software has been annotated, in order to improve accessibility.
8.2 Recommendations

- The exact characteristics of the FLL and PLL have to be determined using input signals with a well defined Carrier to Noise ratio.

- A computer program has to be written which logs the data. This program must be able to use the DCF-77 signal for time alignment.

- The interrupt routine calculating the PLL has to be optimized or a faster processor has to be used, to accomplish that the separate parts of the software cooperate. When functioning, this software has to be programmed in an EPROM.

- A voltage controlled oscillator has to be designed and built for converting the 10 MHz signal down to 124.516 kHz

- Modern personal computers have high processing power. Using a 16-bit A/D-converter card for sampling the data, a fast computer and a C program may result in a simple and well organized system, which combines all advantages of a digital implementation of the receiver.
References

[1] Camp, Op Den, R.H.C.M.  
*Digital Beacon receiver for propagation experiments, Part I: Design and implementation of an FLL/PLL for coherent detection.*  
Eindhoven University of Technology, Department of electrical Engineering, Graduation Report, October 1992

[2] Strik, M.  
*Digital Beacon receiver for propagation experiments, Part II: A Decimator/Detector for Phase and Amplitude Measurements*  
Eindhoven University of Technology, Department of electrical Engineering, Graduation Report, October 1992

*DSP-type first order digital phase-locked loop using linear phase detector.*  
Electronics and Communications in Japan, Part 1, Vol. 69, No. 6, 1986

*All digital phase-locked loop: concepts, design and applications.*  

*Phaselock Techniques.*  
New York, John Wiley & Sons, 1979

*On the development of a spread spectrum transceiver based on a digital signal processor.*  
Eindhoven University of Technology, IVO, IVO-report, 1995
[7] Digital Signal Processing Applications with the TMS320 Family
Theory, Algorithms and Implementations.
Texas Instruments.

[8] Le Van, J.
A fast CRC

Digital beacon receiver for propagation experiments:
Design, testing and optimization of a Frequency-Locked Loop / Phase-Locked Loop
Eindhoven University of Technology, Department of electrical Engineering,
Graduation Report, April 1994

[10] Propagation experiments in the Netherlands in the 12,20 and 30 GHz band with the
Olympus-satellite.
PTT research Dr. Neher laboratories, Delft University of Technology, Eindhoven
University of Technology.

Fase- en amplitude metingen met behulp van een digitale signaal processor
Eindhoven University of Technology, Department of electrical Engineering,
Graduation Report, July 1991

[12] Enden, A.W.M. van den, N.A.M. Verhoeckx
Digitale Signaalbewerking
Amerongen, Delta Press BV, 1987

[13] Shanmugam, K. Sam
Digital and analog communication systems
John Wiley and Sons, 1985
The clock signal of the A/D converters need to be very accurate to ensure that further calculations are accurate.
The demands on the clock signal are:
- Square wave TTL-signal
- Frequency = 12.288 kHz ± 0.09Hz
- \( T_l = 5 \) µs.

Crystal oscillators are very suitable for high accuracy clock signals. Unfortunately oscillators at the desired frequency are not available. To obtain the desired frequency a frequency divider should be used. Several are available, but IC 74HCT4059 is ideal because all natural dividers can be implemented. A suitable crystal oscillator is 19.6608 MHz, which can be divided by 1600, which results in 12288Hz.
The duty cycle however is 50%. To reach the demands this should be adapted by a one-shot multivibrator 74HCT4538. The low time is adaptable with a RC-combination and can be calculated using \( 0.7 R C \). C is chosen 680 pF, so R should be 5147 \( \Omega \). To adjust the timing, a potentiometer of 10K is used.

![Figure 7.3 The 12.288 kHz oscillator](image)
Appendix

The calculation of the FFT constants

B.1 Hanning Window factors

The Hanning window is calculated using Formula B.1.

\[ h(n) = 0.5 + 0.5 \times \cos\left(\frac{2\pi n}{N}\right) \]  \hspace{1cm} (B.1)

The Quick Basic program listed below will calculate these factors and will generate a table which contains all these factors. The table can be copied into the TMS320 program.

```
'******************************************************************************
'** HAMMING.BAS   Written in Quick Basic
'** D.J. Aalberts  april 1995
'******************************************************************************
'** calculates the Hamming Factors for a N-point FFT
'** for use with the FLL/PLL program for the TMS320C25
'** The file generated by this program has to be copied
'** FLL/PLL software
'******************************************************************************
'** initialisation
INPUT "N=", N
teller = 0
PI = 3.141592654#
'** Open file Hamming.dat
OPEN "hamming.dat" FOR OUTPUT AS 1
'** calculation of N Hamming Factors
PRINT "Calculation started"
PRINT #1, " .word ";
FOR i = (-N / 2) TO (N / 2 - 1)
h = .5 + .5 * COS(2 * PI * (i + .5) / N)
h = INT(h * ((2 ^ 15) - 1))
IF teller = 10 THEN PRINT #1, : teller = 0: PRINT #1, " .word ";
PRINT #1, USING "####"; h;
IF teller <> 9 THEN PRINT #1, "; "; PRINT "; ";
teller = teller + 1
NEXT i
PRINT "Calculation finished"
'** end of calculation, Close file
CLOSE (1)
```
B.2 Twiddle factors

To calculate the twiddle factors the following Formula is used:

\[ W_N^k = e^{-j2\pi k/N} = \cos(2\pi k/N) + j\sin(2\pi k/N) \]  \hspace{1cm} (B.2)

This formula is implemented in the Quick Basic program listed below. It generates a table TWIDDLE.DAT which can be copied into the software. The tables is filled with \( N/2 \) real and imaginary parts of equation B.1.

```basic
'******************************************************************************
'* TWIDDLE.BAS    Written in Quick Basic
'* D.J. Aalberts   april 1995
'******************************************************************************
'* calculates the Twiddle Factors for a N-point FFT
'* for use with the FLL/PLL program for the TMS320C25
'* The file generated by this program has to be copied
'* into the FLL/PLL software
'******************************************************************************
'* initialization
' PI = 3.1415927#
' INPUT "N =", N
' Q15 = (2 ^ 15) - 1
'* Open file Hanning.dat
' OPEN "TWIDDLE.DAT" FOR OUTPUT AS 1
'* calculation of N Twiddle factors
' PRINT #1, " .word ";
' FOR k = 0 TO N / 2
'   X = 2 * PI * (k / N)
'   RE = INT(COS(X) * Q15 + .5)
'   IM = INT(SIN(X) * Q15 + .5)
'   PRINT #1, USING "#######"; RE; ", ";
'   PRINT #1, USING "#######"; IM;
'   IF INT((k + 1) / 5) = (k + 1) / 5 THEN
'     PRINT #1, : PRINT #1, " .word ";
'   ELSE PRINT #1, ", ";
' NEXT k
'* End of calculation, close file
' CLOSE (1)
```

Optimization and testing of a digital FLL/PLL
Appendix

The (sub)sampling theorem

Analog signals have to be sampled at certain moments to convert these signals to a digital value. The Nyquist theorem states that a signal with frequency $f$ should be sampled with a sample frequency $2f$. Signals with a frequency spectrum with frequency components higher then $f$ should be filtered by a lowpass filter. If the sample rate is lower than $2f$, the higher frequency components will return in the baseband. This effect is called aliasing. If a signal is bandlimited, sampling at a lower frequency as stated by the Nyquist theorem is possible. This is called subsampling.

\[
\frac{mf_s}{2} < f < \frac{(m+1)f_s}{2}, \quad m \in \mathbb{N} \tag{B.1}
\]

The principle of aliasing is used in the subsampling theorem. The bandpassed signal is copied to the baseband.

![Diagram showing subsampling](image)

**Figure B.1 Subsampling at 12.288 kHz of the 124.952 kHz signal**

The subsampling theorem is used in the FLL/PLL hardware. A carrier at 124.516 kHz is sampled at 12.288 kHz. The carrier will appear in the baseband at 1636 Hz. Attention should be payed to the fact that a shift of 100 Hz of the carrier will result in a shift of the subsampled signal of 100 Hz. This principle is used in the Frequency Tracking Algorithm.
Appendix

The programs for the TMS320C25

A program for the TMS320C25 consists of a file with extension .ASM contains the code and a file with extension .CMD is needed by the compiler to place all data block in the appropriate memory blocks.
/*

Linker Command File

*/

TMSPRG.OBJ
TABLES.OBJ
-o TMSPRG.out
-m TMSPRG.MAP

MEMORY
{
  PAGE 0 : VECTORS : origin = 00H, length = 020H
  CODE : origin = 020H, length = 03FE0H
  ProgBO : ORIGIN = OFF00H, LENGTH = 100H

  PAGE 1 : RAMB2 : origin = 060H, length = 020H
  RAMB1P6 : origin = 0300H, length = 080H
  RAMB1P7 : origin = 0380H, length = O80H
  PARRAM : origin = 400H, length = 700H
  FFTRAM : origin = 1000H, length = 1000H
}

SECTIONS
{
  vectors : > VECTORS PAGE = 0
  .text : > CODE PAGE = 0
  Strings : > CODE PAGE = 0
  tables : > CODE PAGE = 0
  }

  TABLES.OBJ
  }

  ProgBO : LOAD = CODE RUN=OFF00H
  page6 : > RAMB1P6 PAGE = 1
  plIRAM : > RAMB1P7 PAGE = 1
  global : > RAMB2 PAGE = 1
  parRAM : > PARRAM PAGE = 1
  fftram : > FFTRAM PAGE = 1
}
Variables declaration

* Int 0/sample variables

C .usect "page6",1 ;Co-Polar input samples
X .usect "page6",1 ;X-Polar input samples
CBP .usect "page6",1 ;C BandPass (Q15)
XBP .usect "page6",1 ;X Bandpass (Q15)
TEMP .usect "page6",1 ;
EVNODD .usect "page6",1 ;Determines if the input sample is even/odd
AR1BCKUP .usect "page6",1 ;Next storage adress in the Interrupt handler

* FFT variables

FFTPOWER .usect "page6",1 ;FFT power
N .usect "page6",1 ;number of FFT input points
NMIN1 .usect "page6",1 ;number of FFT input points - 1
NBY2 .usect "page6",1 ;number of FFT input points / 2
NBY2MIN1 .usect "page6",1 ;number of FFT input points / 2 - 1

* ERF .usect "page6",1 ;FFT ERF factor
ERFMIN1 .usect "page6",1 ;ERF ERF factor - 1

DATTDRY .usect "page6",1 ;if enough samples collected -> DATTDRY = 1

INFBNK .usect "page6",1 ;number of current input bank
FFTBNK .usect "page6",1 ;start address of current FFT bank
FFTFINP .usect "page6",1 ;input sample to FFT routine
NOSAMP .usect "page6",1 ;number of samples currently read
RANCOF .usect "page6",1 ;hanning multiplication coefficient

* TDWINC .usect "page6",1 ;twiddle exponent incrementation factor
TWDSTG .usect "page6",1 ;twiddle stage counter
TWDTPNP1 .usect "page6",1 ;address of first twiddle input data
TWDTPNP2 .usect "page6",1 ;address of second twiddle input data
TWDINS .usect "page6",1 ;real twiddle exponent factor
TWDCCOS .usect "page6",1 ;imaginary twiddle exponent factor

* MAXPOW1 .usect "page6",1 ;Lower byte of the Maximum power
MAXPOW2 .usect "page6",1 ;Higher byte of the maximum power
MAXPADR .usect "page6",1 ;Address of the maximum power
MAXQFDS .usect "page6",1 ;Offset of the addr which contains maxpower
FREQC1 .usect "page6",1 ;Scaled Maximum frequency
FPPCOUNTER .usect "page6",1 ;Number of FFT-Calculation

* Parallel port variables

AR1VAR .usect "page6",1 ;Memory location to save AR1
VAR1 .usect "page6",1 ;Multi purpose variable
VAR2 .usect "page6",1 ;Multi purpose variable
DATA .usect "page6",1 ;
TRDATA .usect "page6",1 ;
SINCR .usect "page6",1 ;
PAR_ZeroCnt .usect "page6",1 ;
PAR_Status .usect "page6",1 ;

* Private variables

Mon Dec 11 12:58:52 1995 PLL05.ASM
Mon Dec 11 12:58:52 1995 PLL05.ASM

PAR_CRC .usect "page6",1
PAR_ReccRC .usect "page6",1
PAR_ReccLen .usect "page6",1
PAR_ReccCnt .usect "page6",1
PAR_TmLen .usect "page6",1
PAR_TmCnt .usect "page6",1
PAR_Retry .usect "page6",1
Temp1 .usect "page6",1
Temp2 .usect "page6",1
Help1 .usect "page6",1
Help2 .usect "page6",1

* PLL variables

CD .usect "page6",1 ; del co-polar sample after bandp filter
XD .usect "page6",1 ; del x-polar sample after bandp filter
HD .usect "page6",1 ; co-polar with hilb delay
CSHIFT .usect "page6",1 ; 90 degrees shift co-pol sample after bpf
CDelay .usect "page6",1 ; delay line counter
MDelay .usect "page6",1 ; Hilbert delay line counter
NUMER .usect "page6",1 ; numerator
DENOM .usect "page6",1 ; denominator
QUOT .usect "page6",1 ; quotient
PH10 .usect "page6",1 ; primary phase
PH11 .usect "page6",1 ; PLL input phase
PHICO .usect "page6",1 ; VCO output phase
PHINF .usect "page6",1 ; phase detector output phase
PH1LP .usect "page6",1 ; loop filter output phase
CO .usect "page6",1 ; VCO ref freq phase accumulation constant
C1 .usect "page6",1 ; loop filter coefficient 1
C2 .usect "page6",1 ; loop filter coefficient 2
LPREG .usect "page6",1 ; loop filter register (bit 0-15)
LPREGH .usect "page6",1 ; loop filter register (bit 16-31)
VCOreg .usect "page6",1 ; VCO phase accumulator
INDEX .usect "page6",1 ; sine table index
IDEN .usect "page6",1 ; in-phase demodulation signal
QDEN .usect "page6",1 ; quadrature demodulation signal
CI .usect "page6",1 ; co-polar in-phase demodulated signal
CQ .usect "page6",1 ; co-polar quadrature demodulated signal
XI .usect "page6",1 ; x-polar in-phase demodulated signal
XQ .usect "page6",1 ; x-polar quadrature demodulated signal
ONE .usect "page6",1
SIGNEXT .usect "page6",1
AR4BCUP .usect "page6",1
AR5BCUP .usect "page6",1
AR6BCUP .usect "page6",1
DVAR .usect "page6",1 ; First Adress Co-polar Delay Line
; First Adress Co-polar Delay Line
; First Adress X-Polar delay line?
; Adres variabele die naar de DA
; converter geschen worde
INV_FLAG .usect "page6",1 ; flag: 0 DA signal niet complemen
SHIFT .usect "page6",1 ; bepaal da DA converter
AR4BCUP .usect "page6",1 ; tijdelijk copy AR0
LOCKSTAT .usect "page6",1 ; lock status
LOCKCTR .usect "page6",1 ; Lock counter Counts number of samples
LOCK .usect "page6",1 ; lock
LOCKCRIT .usect "page6",1 ; lock criterium
DELAYC .usect "parRAM",NDelay ; Co-polar delay line buffer
DELAYX .usect "parRAM",NDelay ; x-polar delay line buffer
DELAYY .usect "parRAM",NDelay ; Co-polar delay line buffer
HTFONCH .usect "parRAM",HilDb ; Hilbert coefficient in onchip RAM

* POST detection LPF variables, Memory Req: 50h

...
**Parallel port possible commands**

- .asect Strings, 400h
- .label ParStrBeg
- CopyStrToAnswer
  - This macro copies a measurement reply message for PC in AnswerStr with format as used by Transmit. Possible messages are defined in "Strings" and must have an even length. Afterwards Temp2 contains the string length so far.

RestoreEnv $MACRO
  - Restore all four levels of the hardware stack
    RPTK */
    PSHD */
- Restore low P register
  MAR */ ;skip T register
  LT * ;114 -> TR
  MPYK 1 ;TR -> PRL
- Restore T register
  LT */ ;113 -> TR
  MAR */ ;skip low P register
- Restore high P register
  LPH */ ;115 -> PH
- Restore accumulator
  ZALS */ ;116 -> ACCL
  ADDH */ ;117 -> ACCH
- Restore status registers
  LST */ ;118 -> ST0
  LST1 */ ;119 -> ST1. AR? = 120
- Restore complete
  EINT
  RET
$ENDM

LptWrtC $MACRO
  - Sends 4 bit character (called a nibble) to the parallel communications output port 1h. If the device is selected by the PC then the character will be put on the centronics bus. Both macros perform the same, however:
    - LptWrtA expects the character to be already in the accumulator
    - LptWrtC expects a constant as parameter
- This macro is used in:
  - the interrupt service routine PAR_Int2
  - the procedure Transmit
- In Transmit it only sends the SOH to start a communication. The macro does not yet generate an IRQ-on-PC signal. However it does load the data nibble with bit 4 in IRQ-on-PC set to 1 (active) into the variable TRDATA and it (re)enables/starts the timer with period WAITLEN = 1875 = 0.15 msec (at 12.5 MIPS). Later on after the first deliberate time-out of the timer TRDATA is then put to the centronics bus and will generate IRQ-on-PC. The period WAITLEN is probably necessary to ensure that the data nibble signal values have settled on the centronics bus. (See also the PAR_Tint interrupt service routine.)
.word GetSpecStr
IDStr .word Sayhelloback
ResetStr .word Reset ; Check HELLO
GetSpecStr .word PCSPEC

* *** Parallel port possible answers ***
Hello .string 0, 41, "Hello, I am a digital PLL or PLL version 1.2"
NotInitStr .string 0, 40, "This function is not initiated by the PC"
Spectrum .string 0, 6, "Data2 *
UnknwCmdStr .string 0, 15, "Unknown command"
Ok .string 0, 2, "Ok"
.label ParStrEnd

**************************************************************************

* Interrupts
**************************************************************************

.sect "vectors"
B INIT ;Reset
B INTU ;EOC AD converters
EINT RET ;Not used, future use: DCF time alignment
EINT RET ;Par Interrupt ;Parlel interrupt
.space 16*16
B Timer ; Timer interrupt
EINT RET ;Not used
EINT RET ;Not Used
EINT RET ;Not used

**************************************************************************

* Program initialisation
**************************************************************************

* Initialize system
**************************************************************************

.text
INIT .sect
DINT ;no interrupts possible
CNFD
LDPK 0 ;datapage = 0
SPM 0 ;no shifts at outputs of P-register
VCEN
ROVM ;reset overflow mode

* Clear memory
**************************************************************************

ZAC LAXP AR1
LALP AR1, RAMBO

* Initialize PLL
**************************************************************************

ZAC SACL VCOREG ;reset VCO register
SACL PRIVCO ;reset PRIVCO
SACL HDELAY
SACL CDELAY ;reset delay line counter
LALK 2C2Bh ;load CO with VCO rest frequency phase
LALK 0C0 ;accumulation: 16384 (90 degrees)
LALK 171 ;load C1 with 15 and C2 with 1422 for
LALK 4738 ;zeta = 0.707 and Bn = 100 Hz
LALK 4C2 ;AR5BCKUP = 1st addr of co-p delay line

**************************************************************************

* Initialize stackpointer
**************************************************************************

LRLK AR7,07CH ; use block 82 as stack, top address 7CH

**************************************************************************

* Initialize interrupt mask register
**************************************************************************

LAC IMR ;copy register to accumulator
LALK 0FFC5H ;reset bit 0, 1, 2, 3, 4 and 5
SACL IMR ;copy accumulator to register
LDPK 6 ;datapage = 6

**************************************************************************

* Initialize FFT constants
**************************************************************************

ZAC LALK FFTPOWER ;set power of FFT to FFTPOWER
SACL FFTPOW
LALK 1,FFTPOWER ;calculate number of FFT input points N
SACL N ;and store
SACL NMIN1 ;calculate N-1
SACL NBY2 ;load CO with VCO rest frequency phase
SACL FFTCOUNT

**************************************************************************

* Initialize auxiliary registers
**************************************************************************

LRLK AR1, FFTBANK0
SAR AR1,AR1BCKUP

**************************************************************************

* Initialize input bank start address and FFT bank start address
**************************************************************************

ZAC SACL INPBKN ;current input bank is bank 0
LALK FFTBK0 ;set FFT bank start addr to FFTBANK1
SACL FFTBK

**************************************************************************

* Initialize STC
**************************************************************************

ZAC SACL CDELAY
SACL C1
LALK 4C2
SACL C2
LALK DELAYC
SACL AR5BCKUP ;AR5BCKUP = 1st addr of co-p delay line
**Initialize LPF**

- LALK AR6BCKUP; AR6BCKUP = addr of x-p delay line
- LALK DELAYH
- LALK AR4BCKUP; AR4BCKUP = addr of co-p delay line
- LALK RAMO
- LALK 560h
- SACL LOCKCRIT; init lockcriterium

**Initialize bit reversal counter**

- \[ \text{SACL EVNODD} \] \; reset even/odd counter

**Copy Int0 routine into Block B0**

- \[ \text{Hilbert coefficients in Block B0} \]

**Initialize parallel port**

- \[ \text{LARK AR1, PAR_ANSWER} \] \; copy parallel port strings
- \[ \text{RPTK ParStrEnd-ParStrBeg-1} \]
- \[ \text{BLKP ParStrBeg, **} \]
- \[ \text{IN VAR1, PAR} \] \; clear port D-PF and read dipsw
- \[ \text{LDPK 0} \]
- \[ \text{LALK TMOUTL} \]
- \[ \text{SACL TIM} \]
- \[ \text{SACL PRD} \]
- \[ \text{LDPK 6} \]

**End of initialize**

- \[ \text{EINT} \] \; interrupts allowed

---

**Main program**

- \[ \text{MAIN \; ZALS DATRDY} \]
- \[ \text{BZ MAIN_L1} \] \; FFT memory bank full?
- \[ \text{ZAC DATRDY} \]
- \[ \text{CALL FFT} \]
- \[ \text{CALL POWER} \]
- \[ \text{CALL OUTPUT} \]
- \[ \text{VCO control output routine} \]
- \[ \text{ipar command received?} \]

**Interrupt routines**

- \[ \text{ZAC \; EVNODD} \]
- \[ \text{NEW SAMPLE is arrived} \]
- \[ \text{This block is placed in internal RAM, Address FF00} \]
- \[ \text{X Samples are sorted out and stored in the appropriate BANK for FFT} \]
- \[ \text{X and C samples are stored in a delay line} \]

---

**Interrupt 0**

- \[ \text{New sample is arrived} \]
- \[ \text{This block is placed in internal RAM, Address FF00} \]
- \[ \text{X Samples are sorted out and stored in the appropriate BANK for FFT} \]
- \[ \text{X and C samples are stored in a delay line} \]

---

**Initilize parallel port**

- \[ \text{LARK AR1, PAR_ANSWER} \] \; copy parallel port strings
- \[ \text{RPTK ParStrEnd-ParStrBeg-1} \]
- \[ \text{BLKP ParStrBeg, **} \]
- \[ \text{IN VAR1, PAR} \] \; clear port D-PF and read dipsw
- \[ \text{LDPK 0} \]
- \[ \text{LALK TMOUTL} \]
- \[ \text{SACL TIM} \]
- \[ \text{SACL PRD} \]
- \[ \text{LDPK 6} \]

**End of initialize**

- \[ \text{EINT} \] \; interrupts allowed

---

**Main program**

- \[ \text{MAIN \; ZALS DATRDY} \]
- \[ \text{BZ MAIN_L1} \] \; FFT memory bank full?
- \[ \text{ZAC DATRDY} \]
- \[ \text{CALL FFT} \]
- \[ \text{CALL POWER} \]
- \[ \text{CALL OUTPUT} \]
- \[ \text{VCO control output routine} \]
- \[ \text{ipar command received?} \]
Interrupt service routine for INT2 from the parallel communications port. For a description of the protocol and data formats see [2].

- PAR_L0 up to PAR_L9 for receiving data from the PC. At the first beginning PAR_Status = 0, during the receiving PAR_Status = 1 and after the successful completion PAR_Status = 2.

- PAR_L10 up to PAR_L14 and PAR_EndInt for transmitting data to the PC (via conditional branches in PAR_L1 and PAR_L2.). During a transmission PAR_Status = 3 as initialized by the routine Transmit. After completion, whether successful or not. PAR_Status = 0.

Auxiliary registers:
- ARP points to AR1 in the whole routine.
- AR1 is restored by means of AR1VAR and contains the current position in the area where received data bytes must be stored (PAR_RecData) or where the to be transmitted data nibbles are stored (PAR_TrmData).

PAR Interrupt SaveEnv

text

----------------------------- --- PAR Interrupt interrupt service routine (INT2) -------------------

Interrupt service routine for INT2 from the parallel communications port. For a description of the protocol and data formats see [2].

- PAR_L0 up to PAR_L9 for receiving data from the PC. At the first beginning PAR_Status = 0, during the receiving PAR_Status = 1 and after the successful completion PAR_Status = 2.

- PAR_L10 up to PAR_L14 and PAR_EndInt for transmitting data to the PC (via conditional branches in PAR_L1 and PAR_L2.). During a transmission PAR_Status = 3 as initialized by the routine Transmit. After completion, whether successful or not. PAR_Status = 0.

Auxiliary registers:
- ARP points to AR1 in the whole routine.
- AR1 is restored by means of AR1VAR and contains the current position in the area where received data bytes must be stored (PAR_RecData) or where the to be transmitted data nibbles are stored (PAR_TrmData).

PAR Interrupt SaveEnv

LDLK 0 ; Switch to register space
LAC IPR
ANDX OFFFh ; Disable parallel port and timer interrupt
SAI IMR
INT ; Enable DCF and ADC int dep on prev status
LDLK 6 ; Return to main program
LAR AR1, AR1VAR
RAR AR1, A1VAR
IN DATA, PA2 ; Get Data from parallel interface
**PAR_Tint interrupt service routine (TINT)**

- The timer is used to detect parallel communications port time-outs and can have two periods:
  - **WAITLEN=0.15 msec**, this period is started by the LptWrt macro and defines the time that a nibble is put onto the centronics bus before an IRQ-on-PC is issued.
  - **TMOUTL=5.2 msec**, this period is enabled in TI_LO and states the time within which the PC should react on IRQ-on-PC.

- The timer is enabled or remains enabled in:
  - LptWrt with **WAITLEN**
  - TI_LO with **TMOUTL**

- The timer is disabled in:
  - PAR_Tint when PAR_Status=0 and the last EOT was transmitted in transmit mode
  - TI_L1 when the PC has not reacted on IRQ-onPC

- The timer interrupt is reallocatable by means of the macro RelocInVec. For the parallel communications protocol the timer is allocated in the following way:

---

**Timer**

- **SaveEnv**
- **SaveEnv**
- **saveEnv**

---
Subroutines

FFT
Performs in-place N-point radix-2 DIT FFT transform
-> Data should be in bit-reversed order

ENDFFT

CAR

ENDFFT

LAC

ZAC

ADAK

SAACL

ADDK

SAACL

BZ

ZAC

SAACL

PAR_Status

PAR_ResetTrm

PAR_Status

RstRecEnv

RestRecEnv

...
This routine determines the co-polar signal argument. The newest input samples should be in memory locations CBP and XBP. The output (delayed) samples will be in memory locations CB and XD when returned. The number of delays is specified by the constant NDELAY. The variable CDELAY is used as counter and should be initialized to 0.

```
DELAY
LARP AR5, AR5BCKUP ;auxiliary register points to AR5
LAC *;get hilbert TIME delayed samples
SACL HD;sample HD from delay line
LAC CBP;co-polar sample into buffer
SACL *,0, AR5;into delay line, AR7 is active
LAC HEDELAY;increment HWAIT by 1
SACL HEDELAY
SUBR TDELAY;compare HWAIT and TMWAIT
BLZ WAIT1;brach if argument full
SACL HEDELAY
SUBR TDELAY;reset register AR7
WAIT1 SAR AR5, AR5BCKUP
RET
```

**HILBERT TRANSFORMER**

This FIR Hilbert transformer uses on-chip memory block B0 (page 5): 2E7-2FF for storage of delay-tap data samples. The newest input should be in memory location CBP when called.

```
HILFLT
LODPK 6 ;get input for hilbert transf
LAC CBP
SOVM
LODPK 7 ;
SACL HTFINP ;
LARP AR4 ;AR4 is active
LARK AR4, RAM31END-1 ;extra word for shift.
MDPK 0 ;reset P-register
ZAC ;reset accumulator
RTPK LHLIBL-1 ;repeat [LHLIB] times
MCD0 0FD00H+HILBERT, *;mult/acc, shift data word in
APAC ;block BL and decrement AR4
LDPPK 6 ;final addition of P-reg to acc
SACH CSROWN, 1 ;scale and store y
RET
```

```
POWER
ZAC SACL MAXPOWL ;reset MAXPOWL
SACL MAXPOWH ;reset MAXPOWH
LAC FFTBK1 ;calc start address for power calculations
SACL Temp1
LAR AR4, Temp1 ;set AR4 to start address for power calc
LAR AR1, NBY2MINI ;set AR1 to N/2-1
LRLK AR5, FFTBK1 ;set AR5 to power spectrum storage address
LARP AR4 ;AR4 is active
POWBIN ZAC ;ACC:=POW-MAXPOW
MPYK 0 ;P:=0
SQR2 *.5 ;P:=Re-2, ACC+P=0
SQR2 *,1 ;P:=Im-2, ACC+P=Re-2
APAC ;ACC+P=Re-2-Im2
SACL *;store calculated power of bin in FFTBK2
SACL *;end of loop
SUBH MAXPOWH ;ACC:=POW-MAXPOW
SUB MAXPOWL ;MAXPOWL
BLE2 NEXTBIN ;if POW-MAXPOW then goto NEXTBIN
ZAC ;ACC:=POW-MAXPOW then go to NEXTBIN
```

**POWER**

This routine determines the co-polar signal argument. The newest input samples should be in memory locations CBP and XBP. The output (delayed) samples will be in memory locations CB and XD when returned. The number of delays is specified by the constant NDELAY. The variable CDELAY is used as counter and should be initialized to 0.

```
ADJPOW
SBRK 2 ;get input for hilbert transf
SAR AR5, MAXPADR ;store maximum power bin address
LAC *;store new maximum power high
ZAC ;This accumulator
SACL MAXPOWL ;store new maximum power low
NEXTBIN
LARP AR1 ;ar5
SANZ POWBIN, *.4 ;if power of < N/2 bins calc -> to POWBIN
ENDPOW
LAC MAXPADR ;load accumulator with max power bin addr
SBLK FFTBK2 ;calculate offset from DC
SFR ;divide by 2
SACL FREQOFFS ;and store result in FREQOFFS
SACL FREQUENCY ;and store result in FREQUENCY
RET ;back to main
```

```
HTFLT (Hilbert transform filter)
* This FIR Hilbert transformer uses on-chip memory block B0 (page 5):
* 2E7-2FF for storage of delay-tap data samples. The newest input
* should be in memory location CBP when called.

```
```
HTFILT
LODPK 6 ;get input for hilbert transf
LAC CBP
SOVM
LODPK 7 ;
SACL HTFINP ;
LARP AR4 ;AR4 is active
LARK AR4, RAM31END-1 ;extra word for shift.
MDPK 0 ;reset P-register
ZAC ;reset accumulator
RTPK LHLIBL-1 ;repeat [LHLIB] times
MCD0 0FD00H+HILBERT, *;mult/acc, shift data word in
APAC ;block BL and decrement AR4
LDPPK 6 ;final addition of P-reg to acc
SACH CSROWN, 1 ;scale and store y
RET
```
samples should be in memory locations CD and CSHIFT. The calculated argument will be in memory location PHI IN. The procedure uses the variable PHIO as temporary helpvariable.

••

PLL
This routine generates an argument PHIVCO that is phase locked to the input signal argument PHIIN.

DEMO0
This routine generates an argument PHIVCO that is phase locked to the input signal argument PHIIN.

PLL2
This routine generates an argument PHIVCO that is phase locked to the input signal argument PHIIN.

DEMOD
Demodulate the I Q signals from the delayed input lines
**PLLO5.ASM**

```asm
BIT PHIVCO, 0 ; test sign-bit of PHIVCO
BBZ DEMOD2 ; if PHIVCO => 0, branch to DEMOD2
BIT PHIVCO, 1 ; test if PHIVCO => 16384
BBNZ DEMOD1 ; if -16384 < PHIVCO < 0, branch to DEMOD1
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ; Acc = Acc/2; Shift Right
SFR ; ...
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
```

**PLLO5.ASM**

```asm
BIT PHIVCO, 0 ; test sign-bit of PHIVCO
BBZ DEMOD2 ; if PHIVCO => 0, branch to DEMOD2
BIT PHIVCO, 1 ; test if PHIVCO => 16384
BBNZ DEMOD1 ; if -16384 < PHIVCO < 0, branch to DEMOD1
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ; Acc = Acc/2; Shift Right
SFR ; ...
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR QDEMOD ; read sine
LAC QDEMOD ; calculate quadrature demodulation value and
NEG ; store result in QDEMOD
SACL QDEMOD ; and store result in QDEMOD
B DEMOD4 ; branch to DEMOD4
ANDX 16383 ; calc index of sine table (delete 2 MSBs)
SACL INDEX ; and store result in INDEX
SFR ;
ADLK SINTBL ; add start address of sine table
TBLR IDEMOD ; read sine
LAC IDEMOD ; calculate in-phase demodulation value and
NEG ; store result in IDEMOD
SACL IDEMOD ; index of sine table
LAC INDEX ; calculate index of sine table
SBLX 16384 ;
NEG ;
SFR ;
SFR ;
```

```
This procedure starts a packet transmission from the device to the PC by sending the first character SOH a nibble). After that the whole rest of the packet is transmitted and taken care of by the interrupt service routine PAR_Int and PAR_Tint. First however Transmit fills the packet array PAR_TrmData with an answer string.

Auxiliary registers:
- AR0 points to the to be transmitted answer string of which the first word contains the length in bytes, see for examples the section "Strings".
- AR1 points to the packet storage area PAR_TrmData.
- AR2 counts the number of bytes in the string.

Labels:
- TRM_L1 and TRM_L2 from a loop that stores the high byte, respectively the low byte of the words in the string as nibbles (two nibbles per byte) in the packet.
- TRM_L3 at the end the SOH is put to port 1h with LptWrtC and PAR_Status=1, PAR_Retry=0, PAR_TrmLen-length of packet, and PAR_TrmCnt=1. It is not necessary to initialize AR1VAR, because this will be done in PAR_L10.

Remark:
When a packet transmission does not complete successfully a retry can be performed in the following cases:
- When another nibble then AACK was received from the PC on SOH, then two additional retries can be done by immediately retransmitting the SOH in PAR_L11. For even more retries the program branches to PAR_RetryTrm.
- The part PAR_RetryTrm restarts a packet transmission all over again by making PAR_TrmCnt=1, resetting AR1 and sending the SOH. This can happen up to two times, otherwise PAR_Status=0 and the transmission is aborted.
- PAR_L10, when no correct AACK was received on SOH.
- PAR_L12, when no correct DACK was received on LEN2-0, D1h..Dlen-4h.
- Just as PAR_RetryTrm the part TI_ReturnTmr in the PAR_Tint interrupt service routine restarts a packet transmission all over again when a time-out of TMOUTL = 5.2 msec (at 12.5 MIPS) occurs. This indicates that no data was received from the PC.
- PAR_CRC,12
- PAR_CRC,4
- PAR_CRC
- PAR_CRC
- PAR_CRC,4
- PAR_CRC
- PAR_CRC,4
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CRC
- PAR_CR...
Algorithm with lookup table from BYTE magazine November 1987 page 339

IC_L1
LAR AR2,*,AR2 ; AR2 points to command ID
LAC * ; Subroutine address
SAACL Temp2
LAR AR1,*,AR1 ; Number of characters to check
MAR * ,AR2 ; AR1 = number of char to check - 1
LRLK AR0,PAR_BecData
LAC *,AR0 ; AR2 points to known command char
SAC Temp1 ; Save high byte
LAC *,AR1 ; AR0 points to received character
SUB Temp1 ; compare with known character
BNZ PC_L4 ; equal?
BANZ PC_L3,*,AR2 ; equal yes, more characters to check?
LARP AR0
LAC Temp2 ; command found
BACC ; goto subroutine

PC_L3
LAC *,AR0 ; AR2 points to known command char
ANDK OFFH ; get low byte
SUB *,AR1 ; compare with received character
BNZ PC_L4 ; equal?
BANZ PC_L2,*,AR2 ; equal yes, more characters to check?
LARP AR0
LAC Temp2 ; Command found
BACC ; goto subroutine

PC_L4
LARP AR4
BANZ PC_L1,*,AR3
LRLK AR0,UnknwCmdStr ; Command not recognized!
CALL Transmit ; Transmit "Unknown command"
B MAIN

PCSPEC DINT
CopyStrToAnswer Spectrum
SACR AR1,Help1 ;Help1=startposition
LRLK AR0,FFTBANK2
LRLK AR0,2 ;Increment factor
LARK AR6,255
LAC FREQUENCY
SACL ** ,0.4 ;:frequency in eeste word
PCSPEC 1 LAC * ,0 ;:Hoogste word power 0
ADD * ,0 ;:Hoogste word power 1
ADD * ,0;0.1 ;:Hoogste word power 2
ADD SACL ** ,2.6 ;:gemiddelde truc
BANZ PCSPEC_L1,*,.4
SACR AR1,Help2 ;help2=eindpositie
larp 1
LAC Help2
SUB Help1 ;Length of the data
ADD Temp2 ; De al bewaarde lengte
LRLK AR0,AnswerStr ;
LARP 0 ;Store Length field
SACL * ; CALL Transmit
EINT ;initialize transmitting
B MAIN

Sayhelloback
Say hello back "Hello, I'm a digital FLL-PLL"

Sayhelloback LRLK AR0,Hello
CALL Transmit
B MAIN

Reset
LRLK AR0,OK
CALL Transmit

RS_L1
LAC PAR_Status
BNZ RS_L1
RPTX 255
LAC 300h ;Do nothing, give timer chance to generate
;last interrupt of "OK" transmission
B 0000h

Answers and actions taken on received command

Reset
Complete reboot is executed after transmission of "OK"
Same as pressing reset button except that FIFO is not reset by hardware