DEVELOPMENT OF A TWO ELEMENT CORRELATING RADIO TELESCOPE INTERFEROMETER

Master of Engineering: Electrical

Thesis

by

David James Callaghan

DEVELOPMENT OF A TWO ELEMENT CORRELATING RADIO TELESCOPE INTERFEROMETER

Submitted to the Department of Electronic Engineering in the Faculty of Engineering and The Built Environment at Durban University of Technology, in fulfilment of the academic requirements for the Degree

Master of Engineering: Electrical

by

David James Callaghan

March 2015

_____ _____

Mr SD MacPherson Date

Supervisor

_____ _____

Dr O Sokoya Date

Co-supervisor

DECLARATION

I, David James Callaghan, hereby declare that the contents of this thesis, entitled **DEVELOPMENT OF A TWO ELEMENT CORRELATING RADIO TELESCOPE INTERFEROMETER**, is a true reflection of my own work, that all sources of information have been appropriately referenced and that this thesis has not been submitted, in whole or part, for a degree to any other University or Institution.

_____ _____

DJ Callaghan Date

Student Number: 18604709

Acknowledgements

When first embarking on this research project, the task, while very interesting and inspiring, seemed more than a little overwhelming. It was clear from the outset that this was not going to be a lone endeavour; in fact this research project would require input from many people, companies and organizations. While formal thanks and acknowledgement is extended to those mentioned below, there were so many others who gave help, advice, guidance and support along the way.

Firstly a huge debt of gratitude to my supervisor, Mr Stuart MacPherson, who maintained a steady belief in the original purpose of this research project even though my own resolve waivered several times over the years. Thank you Stuart for your patience, guidance and wisdom throughout; the original purpose simply would not have been achieved without this. Secondly to Mr Gary van Vuuren who, together with Mr Stuart MacPherson, jointly heads the Radio Astronomy Technology (RAT) Centre at the Durban University of Technology (DUT); thank you for your encouragement and support throughout.

A special word of thanks to Xilinx for their support, interest and generous donations of equipment, software tools and free training. It was through a Xilinx University Programme (XUP) sponsored training workshop that I had the pleasure of meeting Professor Bob Stewart and Louise Crockett of Steepest Ascent Ltd and the University of Strathclyde, Scotland. Thank you Bob and Louise for a superb DSP on FPGA workshop and for your continued help and support afterwards; I would not have succeeded with the FPGA design without this.

Thank you to Professor Girish Beharee of the University of Mauritius for your help, advice and support during your visits to the RAT centre at DUT and the many emails that followed. Thank you also to Francois Kapp for generously allowing me to spend a week at the SKA Africa laboratories in Cape Town and for the help and advice from Jason Manley and all the staff there. Thank you to Mentor Graphics for the trial version of PADS PCB design that was used to design the interface PCB and to NXP for helping with information about the TDA18272 silicon tuner IC.

Abstract

A two element correlating radio telescope interferometer is the fundamental building block of modern radio telescope aperture synthesis arrays. Early radio telescopes consisted of a single antenna, usually a dish antenna. Larger and larger antennas were constructed in order to improve the resolution of the measurement of the direction and extent of radio frequency radiation coming from the sky. Telescope resolution is fundamentally limited by the ratio of the telescope aperture to the wavelength of the received radiation. For single element radio telescopes to approach the resolution of their optical telescope counterparts, they would need to be impractically large. Mathematical analysis of correlating two element radio telescope interferometers shows that very large aperture radio telescopes can be synthesized from a number of two element interferometers.

An array of two element correlating radio telescope interferometers can be used to produce a synthesized aperture equal to the largest distance between two receiving antennas in the array. Telescope arrays thus enable very high resolution since the angular resolution of a telescope is proportional to the wavelength of the received signal divided by the aperture diameter. A spread of separation distances between antenna pairs is required to produce a complete image of the radiating sources in the field of view. Modern digital signal processing techniques can be used to provide cost effective performance and flexibility in two element correlating radio telescope interferometer design.

The aim of this research project was to design and construct a two element correlating radio telescope interferometer using modern digital signal processing techniques and hardware. The

relevant theory has been investigated together with suitable hardware and software platforms and tools used to produce such a system. The two element correlating radio telescope interferometer produced, will be used as a platform for further investigative research into its design, performance and application.

The outcome of this research project was the successful completion of a working two element correlating radio telescope interferometer. The development process has been analysed and carefully documented. Some fringe measurements for a simple single frequency radiating point source have been taken and these measurements have been analysed according to theoretical expectation. Potential for further research, using the two element correlating radio telescope interferometer produced, has been identified and discussed.

Contents

List of figures

List of tables

List of annexures

Glossary of terms

ADC: Analogue to digital converter. An electronic device that converts a continuous analogue input voltage to a discrete binary digital number.

AGC: Automatic gain control. The process whereby an electronic circuit automatically adjusts the gain of an amplifier in order to keep the output signal level within acceptable limits as the input signal level fluctuates.

algorithm: The logic method used to achieve desired processing of data.

architecture: The design principles of a fixed set of digital circuits that are used to process data.

aperture: The EM wave receiving area of a radio antenna or optical instrument.

ASIC: Application specific integrated circuit. A circuit, custom designed for a specific application, that is integrated onto one silicon chip.

azimuth: The horizontal angle between some reference direction (such as true North) and the direction in which some instrument is pointing.

bandwidth: The range of signal frequencies that a circuit or system will process. The highest frequency minus the lowest frequency.

baseline: The distance between the two receiving elements in a two element radio or optical interferometer.

BEE: Berkeley emulation engine. An FPGA based development board.

BGA: Ball grid array. A matrix of conducting bumps on the underside of an IC used to connect the internal circuits to tracks on a PCB.

C: A high level microprocessor or microcontroller programming language.

clip: To limit the voltage at some point in a circuit to some level. Also refers to imposing a numerical limit on a register in a digital signal processor.

component: An electronic device connected onto a PCB by soldering.

configuration: The pattern of digital circuit interconnections in an FPGA.

conversion: The process whereby a circuit or device converts data in one form to another form.

conversion delay: The time taken for a circuit or device to perform a data conversion.

correlator: A circuit or device that performs the mathematical function of correlation on two input data streams.

DAC: Digital to analogue converter. An electronic device that converts a discrete binary digital number to a discrete analogue voltage.

daughterboard: A secondary PCB that plugs onto the main PCB in an electronic instrument.

DB9: A common nine pin connector for cable connecting devices to a PC.

DDR: Double data rate. A fast modern RAM technology that transfers data to or from memory on both the rising and falling edges of the master clock. This allows double the data transfer speed compared to older RAM technology, where data is transferred on either the falling or rising edge of the master clock.

delay: A digital circuit where the digital number at the output equals the digital number that was at the input some number of clock cycles back.

digitize: The process of converting an analogue quantity to a binary digital number.

DIP: Dual in-line package. An IC with two rows of conducting pins protruding from opposite sides. The pins are bent 90 degrees so that they can mount into holes in a circuit board to connect the internal circuitry of the IC to tracks on the circuit board.

DSP: Digital signal processing. The methods used to process digitized analogue input signals to achieve some desired output.

EEPROM: Electrically erasable programmable read-only memory. Non-volatile (does not lose the data contents when power is removed) digital memory that can be erased and programmed repeatedly by an electrical signal.

elevation: The angle from a perfectly flat horizon that an instrument is pointing up into the sky.

EM: Electromagnetic. Refers to wave propagation resulting from the interaction of changing electric and magnetic fields.

emulate: When a programmable digital device is programmed to function in the same way as some other digital device.

enable: A digital technique used to allow a selected digital result to appear on a digital circuit output.

field of view: The region of sky that a telescope is processing at some particular point in time.

firmware: A programmed sequence of instructions that is fixed in a microcontroller in non-volatile memory until the device is reprogrammed.

flow control: The digital techniques used to synchronize the transfer of data from one device to another.

FPGA: Field programmable gate array. An integrated circuit consisting of a very large number of basic digital circuits (gates) that can be repeatedly programmed by the user (in the field) to be connected together as desired.

frequency divider: A digital circuit that has an output rectangular voltage waveform with a frequency determined by dividing the frequency of the input rectangular voltage waveform by some integer.

fringe: An oscillating pattern of maximum and minimum light intensities or voltage levels or values of a mathematical function or digital signal.

FSPL: Free space path loss. The ratio by which the power of a radiated radio wave signal is reduced when received at some distant location with no absorption or reflections in the space in between (free space).

gain: The ratio by which the power of an amplifier circuit output signal is greater than the power of the input signal.

gate: A basic digital circuit that performs Boolean logic on its inputs.

gateware: The design information that defines how the gates in an FPGA should be interconnected to perform the desired processing of data.

Gerber files: A standard data file format for communicating tracks, pads and drill hole information for PCB manufacturing.

GPIO: General purpose input or output. A digital connection pin that can be used for general (software/firmware/gateware defined) input or output.

GPU: Graphical processing unit. A custom IC processor designed to handle fast graphical processing in PC applications.

ground plane: A large copper area on a PCB that is connected to the ground potential. The large copper area ensures a low impedance connection to ground for various components mounted on the PCB.

GUI: Graphical user interface. A general IT term referring to graphical features of a software application provided for a user to enter information into and obtain information from the software application.

HDL: Hardware description language. A high level language used to define the logic functionality of complex digital logic ICs. Commonly used to define gateware for FPGAs.

IC: Integrated circuit or integrator-comb. A collection of circuits integrated onto one silicon chip or a type of digital filter in DSP, respectively.

IF: Intermediate frequency. The frequency of the signal at an intermediate stage of signal processing in a superheterodyne RF receiver circuit. A much lower frequency than the input RF signal.

IIC: Inter-IC. An NXP (Philips) standard protocol for serial bus communications between ICs on a PCB, sometimes called I^2C.

interrupt: A technique used in microprocessors and microcontrollers to handle input data to a process. The timing of the input data is usually not known and so the input device interrupts (temporarily halts) the process running in the processor when data is ready for input.

interrupt latency: The time delay between an interrupt request signal (IRQ) from an input device and the time that the processor starts to process the input data.

I/O: Input or output. Digital connections to a device for input or output data.

IP: Intellectual property. Refers to ownership of software/firmware/gateware.

IRQ: A signal from a data input device used to interrupt a process running on a microprocessor or microcontroller.

ISM: Industrial, scientific and medical. Refers to licence free radio communication frequency bands for experimental use.

ISR: Interrupt service routine. The sequence of instructions that is executed in a microprocessor or microcontroller after an interrupt request (IRQ) signal is received.

interface: A circuit or software/firmware/gateware that is used to transfer data from one circuit or device to another.

Impact: A Xilinx PC software tool used to program devices on a PCB using a JTAG connecting cable.

ISE: Integrated software environment. A collection of Xilinx PC software development tools for FPGA gateware design, in one GUI application.

JTAG: Joint test action group. An IEEE standard protocol for communication between a PC and digital ICs on a PCB for the purpose of testing the PCB interconnections. It includes a standard protocol for programming the ICs.

layout: The design of the size shape and placement of component pads and interconnecting tracks on a PCB.

local oscillator: A circuit in a superheterodyne RF receiver that provides a local sinusoidal voltage waveform that is mixed (multiplied) with the received RF signal voltage waveform to produce an intermediate frequency (IF) signal.

LPDA: Log periodic dipole array. A type of antenna consisting of an array of dipoles of increasing length.

master clock: A rectangular voltage waveform applied to a circuit or device that provides the basis for timing and synchronizing of all the sequential logic.

microcontroller: A sub-class of microprocessor, custom designed for low level control functionality, with all the required digital circuitry integrated on one IC.

microprocessor: A complete digital processor integrated on one IC.

migration: The process of implementing software/firmware/gateware on a different device to the one used during its initial development.

motherboard: The main PCB in an electronic instrument.

MSB: Most significant bit. The bit with the highest weight in a binary number.

multiplexer: A digital circuit that places data from one of several sources onto a common output.

noise: An unwanted random waveform that combines with a wanted signal waveform.

noise temperature: The temperature of black body radiator or electrical resistor that would produce thermal noise power equal to the noise being quantified.

open source: Software/firmware/gateware for which the source design information is freely available.

pad: A region of copper on a PCB to which a pin of a device is soldered.

parallel: Usually refers to data transfer between devices where there is a separate connection for each bit of the binary number transferred. Also refers to multiple processors or processes working simultaneously on part of a whole process.

PC: Personal computer. A type of computer modelled on the historic IBM PC.

PCB: Printed circuit board. A fibreglass board with copper pads and interconnecting tracks bonded to its surface/s that is manufactured (printed) from a layout design of the pads and tracks.

PIC: Peripheral interface controller. The historical name for the Microchip range of microcontrollers.

pins: Conducting tabs on an electronic component that are soldered onto pads on a PCB to connect the internal circuitry of the component to the PCB track interconnections.

poll: Repeated testing of data input connections to see if the data input device is ready to input data.

QDR: Quad data rate. A fast modern RAM technology that doubles the speed of DDR by having separate ports and separate clock signals for writing data to memory and reading data from memory.

RAM: Random access memory. Historical name for volatile digital memory. Volatile meaning that it loses its data when power is removed from the memory.

ray: A construction line drawn to show the direction of propagation (travel) of an EM wave.

receiver: A circuit used to frequency select, amplify and demodulate an RF signal received on a radio antenna.

register: A set of digital circuits used to hold the bits of a binary number during processing.

regulator: An IC that provides a constant output voltage while the supply voltage varies.

resolution: The smallest unit of measured quantity that any measuring instrument can measure. In the case of telescopes, it refers to the smallest unit of sky area that a telescope can discriminate from adjacent units of sky area.

RF: Radio frequency. A frequency in the Radio band of the EM spectrum. Usually refers to the propagation and electronic processing of Radio band EM wave communications.

RFI: Radio frequency interference. Unwanted man-made EM radiation that is received by some instrument and interferes with the processing of the desired signal.

RHINO: Reconfigurable Hardware Interface for computatioN and radiO. An FPGA based development board.

ROACH: Reconfigurable open architecture computing hardware. An FPGA based development board.

route: A copper track that provides a circuit interconnection between two component pins on a PCB. Also a verb that refers to the process of defining the path of the copper track interconnection.

RS232: An Electronic Industries Alliance (EIA), Radio Sector (RS) protocol standard for serial data communication.

RX: An abbreviation used for receive or receiver.

sample: A snapshot value of a time varying voltage at some instant in time. Usually represented as a binary digital number and usually taken at regular time intervals called the sample period.

sample rate: The rate at which samples of a time varying voltage are taken.

schematic: The graphical technique of drawing symbols to represent electronic components or circuits and straight lines to represent the interconnections between them.

SDR: Software defined radio. Modern radio communication equipment that uses technology that allows the equipment to be reconfigured for different applications, using software.

sensitivity: The smallest level of input signal to which a piece of equipment can respond.

serial: Describing a binary digital process that occurs one bit at a time.

shift: When all the bits of a binary number in a register move to the adjacent position (left or right) in the register. The last bit of the register usually becomes a serial output bit while some serial input bit occupies the first bit position in the register.

silicon tuner: A frequency adjustable radio receiver integrated in one IC.

SKA: Square kilometre array. A modern radio astronomy quest to build a radio telescope array with resolution and sensitivity equal to a single element radio telescope with an aperture of one square kilometre.

software: A sequence of microprocessor instructions that is stored in a mass storage device such as a disc drive and loaded into RAM when it is required to be executed (run) by the microprocessor.

surface mount: An electronic component package with pins that are designed to be soldered onto the same surface of a PCB as the one on which the component is placed (mounted). This type of component mounting does not require holes to be drilled through the PCB for the component pins.

synthesis: The process in FPGA gateware development where the design logic is translated into the required interconnection of the digital circuits within the target FPGA device.

terminal: A piece of equipment consisting of a keyboard and a screen, used to enter data into a computer and get data out. The computer is a separate and remote device to which the terminal is connected.

terminal emulator: Application software for a PC that makes the PC operate as a terminal.

track: A thin strip of copper on the surface of a PCB that provides the circuit interconnection between component pins.

transit correlator: A simple radio telescope correlator that requires the observed source to pass overhead in order to obtain a fringe function.

truth table: A table showing the action or Boolean logic output that results for each Boolean logic input to a digital circuit.

tuner: A frequency adjustable (tuneable) radio receiver.

TVRX2: A two channel tuner daughterboard for the USRP2 that is tuneable across the VHF and UHF (TV channel) range of RF input frequencies.

TX: An abbreviation used for transmit or transmitter.

UHF: Ultra high frequency. A sub-band of frequencies (300 MHz – 3 GHz) within the Radio band.

USRP2: Universal software defined radio peripheral 2. A second generation FPGA based development board.

verilog: A hardware description language.

VHDL: VHSIC hardware description language.

VHF: Very high frequency. A sub-band of frequencies (30 MHz – 300 MHz) within the Radio band.

VHSIC: Very high speed integrated circuit.

via: A hole drilled through a PCB for the purpose of connecting a track on one surface (layer) to a track on the opposite surface (layer).

VSYNC: Vertical sync. A wide rectangular pulse in a TV signal that is used to synchronize the start of a frame of a TV picture in TV receiver equipment.

zenith: The point in the sky directly above a corresponding point of reference on the Earth surface.

Chapter 1

Introduction to concepts

1.1 Introduction

Astronomy is the study of the Universe and until 1930, almost all knowledge of the Universe was restricted to what we can see in the sky with our eyes [1]. This information is brought to us from distant objects in the Universe by electromagnetic (EM) waves in the Visible band of the EM spectrum. The Visible band is a very narrow band of frequencies or wavelengths as shown in Figure 1.1 [2].

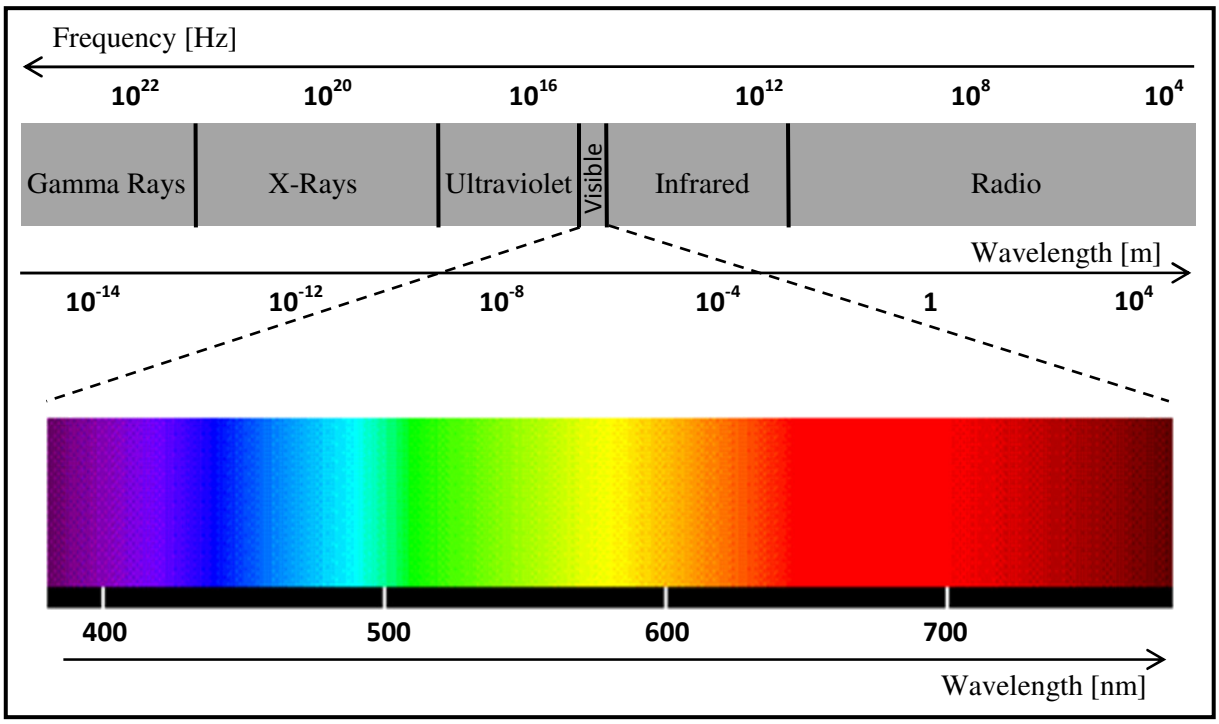


Figure 1.1: The electromagnetic spectrum in frequency and wavelength with the Visible band expanded.

In 1931, while working for Bell Laboratories, Karl Jansky began investigating the source of external radio frequency noise that hampered radio communication systems. He found, using highly directive antenna arrays, that a source of some external radio frequency noise was the

sky and in fact a specific region of the sky [3]. The Sun was later identified as a source of radio frequency noise at higher frequencies in the Radio band but Jansky, and a few years later Grote Reber, found the Milky Way region of the sky to be a strong source at lower frequencies [1].

The work of Jansky and Reber began a quest to discover the source and nature of Radio band EM radiation coming from the sky. The sources turn out to be very distant galaxies, supernovas, quasars and pulsars [2] and what began as communication systems research very soon turned into a branch of astronomy, called radio astronomy.

1.2 Electromagnetic waves

EM waves are time varying electric and magnetic fields that propagate through space at approximately 3 x 10^8 m.s^{-1}. If the space, through which an EM wave propagates, is occupied by some matter, then the speed of propagation slows down by a factor dependant on the type and density of the matter as well as the frequency of the EM wave. The matter or absence of matter through which the EM wave propagates is called the medium of propagation. When there is no matter, the medium is called free space or vacuum [4], [5].

The electric and magnetic fields of an EM wave vary in a periodic fashion, repeating the pattern of variation as it propagates through space. The distance covered by the EM wave for a complete cycle of the pattern of variation is called the wavelength. The time rate at which the EM wave repeats the cycle (in cycles per second or Hertz) is called the frequency of the EM wave. The wavelength of an EM wave is related to the frequency and speed of propagation by the relationship [5], [6]:

$$c = \lambda \nu \qquad 1.1$$

c – speed of propagation [$m.s^{-1}$]
λ – wavelength [m]
v – frequency [Hz]

The frequency of an EM wave is determined by the physical mechanism that produced it while the wavelength is determined by the frequency and the speed of propagation. Since the speed of propagation varies depending on the medium of propagation, the wavelength will also vary depending on the medium, becoming shorter when the wave propagation slows down. The electric and magnetic fields of the EM wave are represented by vectors denoted **E** and **B** respectively [4], [6].

The magnitude and direction of the electric and magnetic field components of an EM wave may vary in both magnitude and direction but the two vectors are always perpendicular to each other. The direction of propagation is always perpendicular to the plane of the electric and magnetic fields and this defines a third vector called the Poynting vector denoted **S**. EM waves fall into a class of waves called transverse waves since the oscillating quantities are always perpendicular to the direction of propagation [4], [6].

The Poynting vector **S** is also known as the energy flux density vector and is given by [4], [6]:

$$\mathbf{S} = \frac{1}{\mu_0}(\mathbf{E} \times \mathbf{B}) \qquad 1.2$$

The energy flux density magnitude gives the amount of energy per unit of surface area crossing a surface perpendicular to the direction of propagation, per unit of time. The power crossing any surface S is therefore given by [4]:

$$P = \int_S \mathbf{S}.\,d\mathbf{a}$$

It is very often more practical to work with the Fourier transform of the energy flux density **S**. This provides a spectral flux density function also commonly denoted **S** and given in units of $W.m^{-2}.Hz^{-1}$. The power spectral density (P_ν) crossing a surface is then given by:

$$P_\nu = \int_S \mathbf{S}.\,d\mathbf{a} \qquad 1.4$$

In radio astronomy, the received EM wave originates from many sources in the sky. Since it is these sources that we wish to identify and quantify it is convenient to consider the received wave as a sum of contributions from radiating sources or segments of an extended radiating source in the sky. For this approach, the sky is divided into solid angle segments dΩ and the contribution of flux from each segment is called the sky brightness B. This sky brightness is spectrally decomposed into a spectral brightness function B_ν as a function of the polar coordinates θ and φ of the sky segment. θ is the horizontal (compass bearing) angle of a sky segment and φ is the elevation angle. Integrating this brightness function over solid angle and over frequency gives the total received source flux S from the sky [7]. The vector notation is often dropped in favour of considering a receiving surface normal to the incoming wavefront.

$$S = \iiint B_\nu(\theta,\phi)d^2\Omega d\nu \qquad 1.5$$

To determine the received power, the source flux must be integrated over the receiving area normal to the incoming flux, as in equations 1.3 and 1.4.

1.3 Electromagnetic radiation

EM radiation is the process that initiates EM waves and refers to the transfer of energy, by EM waves, from a source to the surroundings. The basic prerequisite for producing EM waves is an accelerating electrical charge. This, as we know, can be artificially produced by electrical or electronic devices that produce alternating current flow in electrical conductors.

In the case of man-made Radio band radiation, the radiating conductors are called antennas. The alternation of direction of current flow, guarantees an acceleration of charge. The frequency of alternation of direction of current flow directly determines the frequency of the cyclic variation of electric and magnetic fields in the resultant EM wave [4].

In addition to man-made EM radiation, there is an abundance of naturally produced EM radiation. The most common natural physical mechanism causing accelerating electrical charges is thermal motion of atoms or molecules. The regular collisions of atoms or molecules cause changes of direction and speed which provide the required acceleration of charges to produce EM radiation. The nature of the acceleration of charges due to thermal motion is random and produces a continuous spectrum of frequencies of EM radiation [5], [6].

The spectral intensity or brightness (B_ν) of thermal EM radiation from an ideal black body radiator is distributed over the continuous spectrum of frequencies of the EM waves produced. This distribution of intensity or brightness of radiated energy versus frequency is called Planck's distribution and is determined by Planck's formula, given in equation 1.6, which is derived from the quantum-statistical nature of thermal motion of atoms [5], [8].

$$B_\nu = \frac{2h\nu^3}{c^2}\left(\frac{1}{e^{\frac{h\nu}{kT}} - 1}\right) \qquad [Wm^{-2}.Hz^{-1}.sr^{-1}] \qquad 1.6$$

h	–	Planck's constant	k	–	Boltzmann's constant
v	–	frequency	T	–	temperature [K]
c	–	speed of light			

Planck's formula shows that the intensity of thermal radiated EM wave energy will peak at a frequency determined by the temperature of the radiating object. It turns out that the Sun's thermal radiation intensity peaks in a frequency band very close to the Visible band of the EM

spectrum. This can be seen in the distribution of brightness over frequency versus temperature for thermal radiating bodies shown in Figure 1.2.

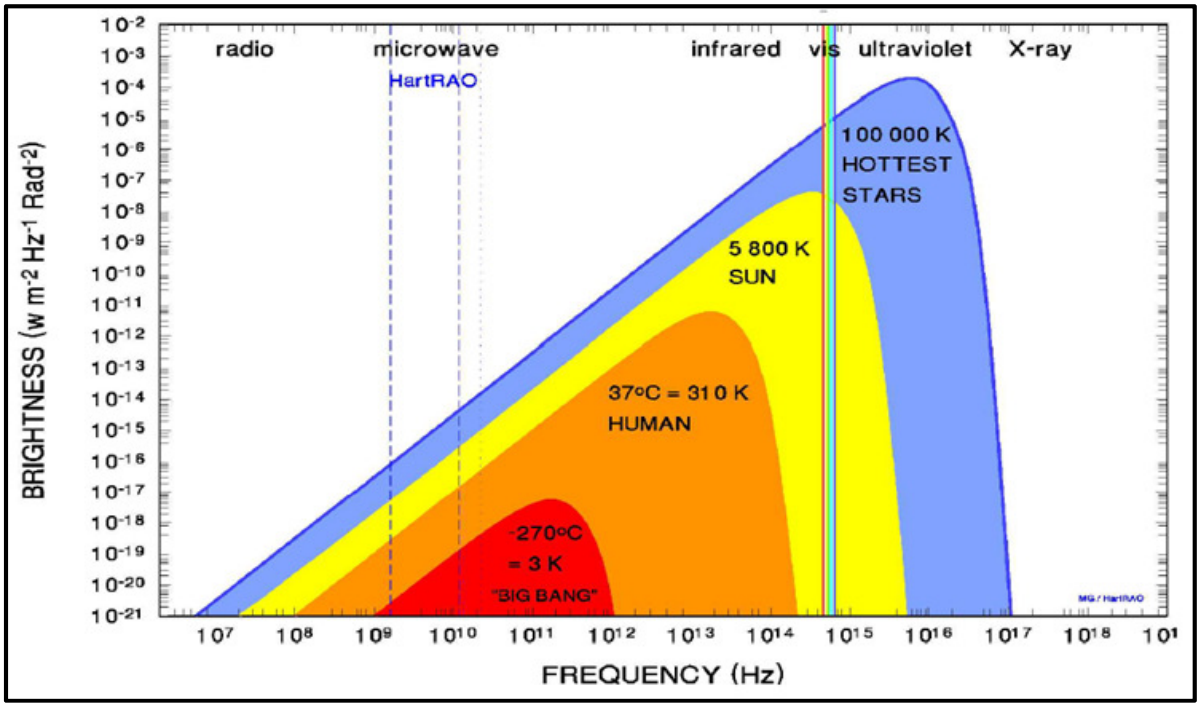


Figure 1.2: From [8], courtesy Dr MJ Gaylard. Distribution of brightness over frequency for blackbody radiation from solid objects of the same angular size, at different temperature, with the Visible band superimposed.

For most measurements in the Radio band, the Rayleigh-Jeans approximation applies since the frequency of interest is well below the peaks shown in Figure 1.2. This applies when $h\nu/kT \ll 1$ and equation 1.6 becomes [7]:

$$B_\nu \approx \frac{2kT\nu^2}{c^2} = \frac{2kT}{\lambda^2} \qquad 1.7$$

This approximation holds for the straight line sections in Figure 1.2, well before the maximum. The graph is a straight line even though the function goes as ν^2 because it is plotted on a logarithmic frequency scale. Equation 1.8 (derived from equation 1.7) is often used to convert source brightness into an equivalent temperature called the brightness temperature T_b:

$$T_b = \frac{c^2}{2k\nu^2} B_\nu = \frac{\lambda^2}{2k} B_\nu \qquad 1.8$$

While visible objects in the cosmos mainly radiate EM waves due to the thermal energy of the object, there are objects that are not visible but radiate strongly in the Radio band of the EM spectrum. The mechanism of radiation for these objects is not thermal. Examples of non-thermal radiating bodies are radio galaxies, quasars and pulsars [2].

1.4 Electromagnetic wave propagation through the Atmosphere

The chemical properties of matter determine the extent to which that matter absorbs and reflects EM waves. The balance of wave energy that is not absorbed or reflected by the matter, passes through the matter. The Earth's atmosphere is a complicated system consisting of layers with different chemical composition and properties. In total however, there exists two distinct bands of frequencies in the EM spectrum for which the Earth's atmosphere is largely transparent, as can be seen in Figure 1.3 [7]. The one that we are most familiar with is the band called the Visible band, since this allows us to see objects in the cosmos, such as the Sun, the Moon and the stars. The transparency of the Atmosphere in this band is called the Optical Window. There is however another wider band, in terms of frequency octaves or decades, for which the Atmosphere is transparent, in the Radio band of the EM spectrum. This transparency is called the Radio Window [7].

An advantage of observations of the cosmos in the Radio band is that it does not suffer as much from high background radiation during the day which makes observations in the Visible band impractical. Even when the region of observation is far away from the Sun, atmospheric scattering of the Sun's visible light makes the entire Atmosphere appear light and blue in colour during the day [6]. This background sky radiation masks out weaker light from distant

stars. In addition, clouds are not transparent to visible light but are transparent to the lower frequency end of the Radio band. Thus observations of the cosmos in cloudy weather are possible in the Radio band but often hampered in the Visible band [2].

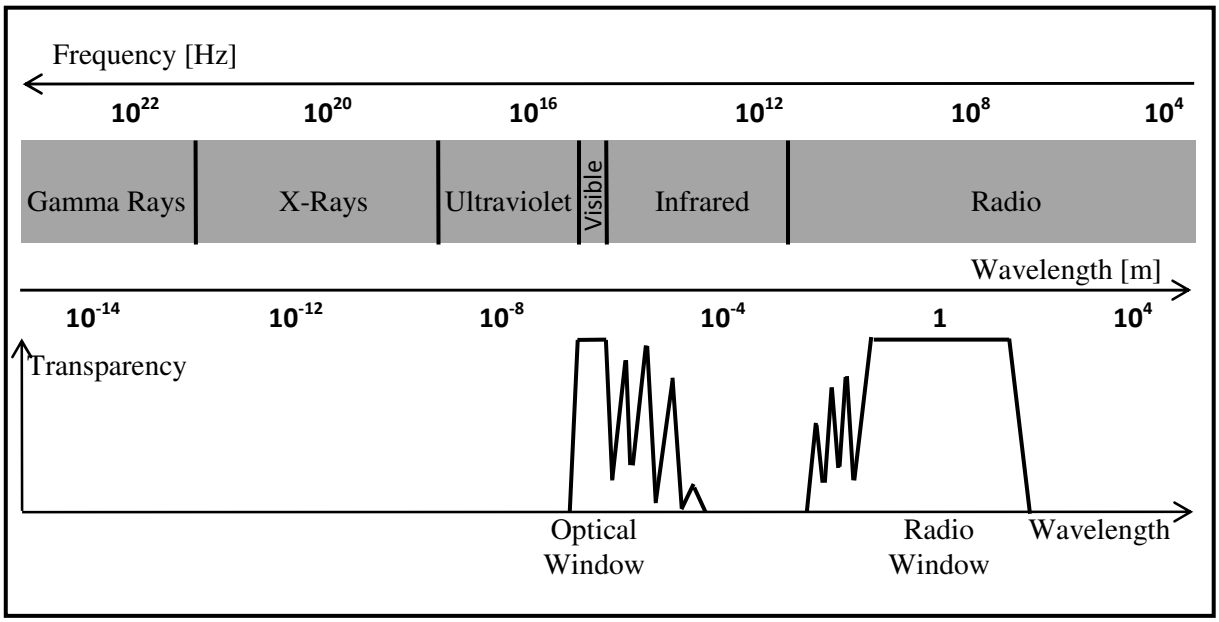


Figure 1.3: Transparency of the atmosphere versus radiation wavelength showing the existence of two distinct transparency windows and their relationship to the bands of the electromagnetic spectrum.

1.5 Telescope resolution

Telescopes are instruments used to observe objects in the cosmos. The cosmos appears to an observer on the Earth's surface as a celestial hemisphere with objects having an elevation above the Earth's horizon and an azimuth angle according to the compass direction of the observed object relative to the observation position. An observed object in the cosmos occupies a portion of the observed celestial hemisphere. This portion is best described as a solid angle segment in units of steradians (sr). A complete sphere has 4π steradians of solid angle therefore a hemisphere has 2π steradians of solid angle and so on. The angular diameter of the smallest segment of sky (celestial hemisphere) that any telescope can distinguish from an adjacent segment of the sky of equal angular diameter is called the resolution of a telescope ($\Delta\theta$), measured in linear angle of radians, being the angular diameter of a circular segment of the celestial hemisphere. The fundamental diffraction limit resolution of any

telescope, optical or radio, results from interference fringes surrounding the outline of the image of observed objects and is determined by the ratio between the wavelength of the received EM waves and the aperture or receiving area of the telescope [1], [5]. The relationship is expressed by:

$$\Delta\theta = \frac{1{,}22\,\lambda}{d} \qquad [\text{rad}] \qquad 1.9$$

λ – wavelength of the received wave
d – diameter of the receiving aperture

The wavelength of an EM wave in the Radio Window, at say 21 cm Hydrogen line wavelength, compared with ~600 nm mid-band visible light wavelength, is a wavelength ratio of ~350 000. A single element 21 cm wavelength radio telescope would need, according to the fundamental diffraction limit resolution, approximately 350 000 times the aperture diameter of an optical telescope for comparable resolution. The largest aperture optical telescope is the Gran Telescopio Canarias (GTC) on the Canary Islands with a reflecting aperture mirror of 10,4 m diameter. A single element 21 cm radio telescope with comparable fundamental diffraction limit resolution would need to have approximately 3 600 km aperture diameter. Most single element radio telescopes use parabolic dish reflectors and the diameter of the dish is analogous to the diameter of the reflecting mirror in optical telescopes. Constructing a dish of 3 600 km diameter is not practical. The largest steerable parabolic dish radio telescopes are approximately 100 m diameter of which there are two. One is located at the National Radio Astronomy Observatory (NRAO) at Green Bank, West Virginia in the USA. The other, built some time before Green Bank is at Max Planck Institut für Radioastronomie at Effelsberg in Germany [7]. These ‘goliaths’ require special stress relief adjustments when they are steered in order to maintain the correct parabolic surface shape. A larger single parabolic dish radio antenna has been constructed near Arecibo on the island of Puerto Rico. The diameter is approximately 300 m but the dish is not steerable. Some beam

steerability is obtained by moving the receiving equipment located at the focal point. The parabolic surface has been modified to minimize the error over the limited steerable range of the focal equipment [7].

1.6 Telescope sensitivity

An increase in aperture also leads to an increase in sensitivity because the flux collecting area is larger and more power is received according to equations 1.3 and 1.4. Sensitivity is the smallest amount of radiation intensity that a telescope can detect. This radiated energy is also distributed across the spectrum of EM wave frequencies. In general, radio sources in the sky are fairly weak due to the large distances between Earth and the radiating sources. A commonly used unit of radiation flux intensity for astronomical Radio band measurements is the Jansky (Jy) [7].

$$1\ \text{Jy} = 10^{-26}\ \text{W.m}^{-2}.\text{Hz}^{-1} \tag{1.10}$$

The sensitivity of radio telescopes is limited by the amount of unwanted noise power added to the wanted signal power from the observed field of view. The wanted signal power needs to be significantly larger than the unwanted noise power in order to discriminate wanted signal from noise. This leads to a performance specification called signal to noise ratio (SNR or S/N). A large part of the unwanted noise power is internal electrical noise produced in the radio telescope equipment itself. A radio telescope will consist of a number of cascaded signal amplification stages and the ratio of SNR at the input of each stage to the SNR at the output of each stage is called the noise factor (f). A low noise factor (close to one) implies little added noise from the amplifier stage itself and therefore little reduction of the SNR from the input to the output of the amplifier stage. Furthermore, analysis of noise factor for cascaded amplifier

systems shows that the overall system noise factor for N cascaded amplifier stages with the noise factor of stage x being f_x and the gain of stage x being g_x is given by [1], [9]:

$$f_{sys} = f_1 + \frac{f_2 - 1}{g_1} + \frac{f_3 - 1}{g_1 g_2} + \cdots \frac{f_N - 1}{g_1 g_2 \cdots g_{N-1}} \quad 1.11$$

The first term on the right of equation 1.11 is the dominant term because f_1 is not divided by any amplifier stage gain. Thus the overall system noise factor is optimally minimised by minimising the noise factor of the first amplifier stage. This is minimised by minimising the electrical noise added in the first amplifier stage [1], [9]. The first amplifier stage in a radio telescope must therefore be a low noise amplifier (LNA). This amplifier is usually cooled to almost absolute zero (-273°C) by liquid nitrogen in order to minimize added internal thermal electrical noise [1]. In correlating telescope arrays, the SNR and hence the sensitivity, can be improved by increasing the correlation integration time. This is discussed further in section 7.3.6. For example, the beam sensitivity of the Long Wavelength Array (LWA) is ~5 Jy for one second of integration time while a sensitivity of ~10 mJy can be achieved from the complete system with an integration time of one hour [10].

There are other factors that contribute to noise, such as background sky radiation and radiation received from other sources that are outside the desired field of view. The problem of radiation received from an unwanted direction arises because real antennas have receiving lobes other than the primary lobe or beam. The objective with antenna design is to achieve high sensitivity to radiation in the direction of the primary lobe compared to radiation received in the direction of other receiving lobes of the antenna, called side lobes (and a back lobe in some cases). Figure 1.4 shows a typical antenna radiation sensitivity pattern.

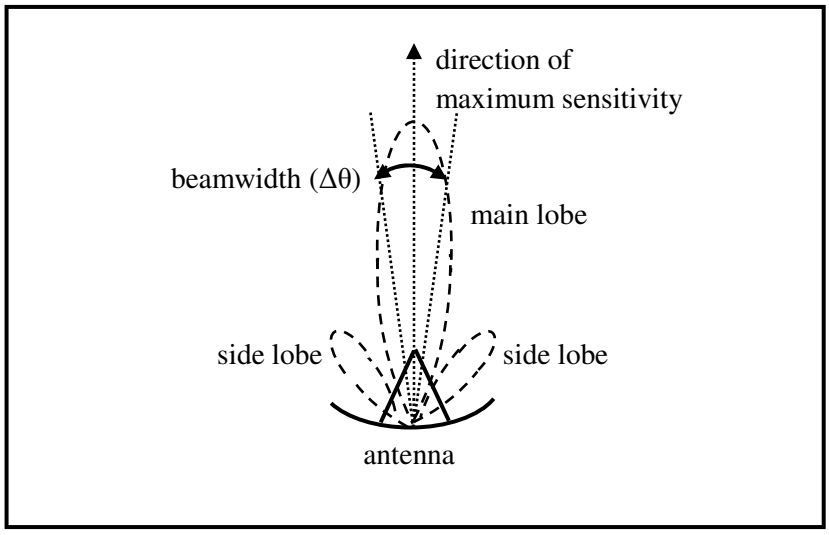


Figure 1.4: Typical antenna radiation sensitivity pattern showing a main lobe, the beamwidth and typical side lobes.

Antenna beamwidth $\Delta\theta$ is the angle between the directions where the sensitivity of the antenna is half that of the direction of maximum sensitivity and is given by equation 1.9. Antenna sensitivity patterns are usually plotted in dB with 0 dB reference in the direction of maximum sensitivity. The beamwidth limits are then at -3 dB sensitivity. A larger aperture antenna, which implies a larger diameter dish in the case of dish antennas, gives rise to a narrower beamwidth, as can be seen from equation 1.9 [9].

1.7 Aperture Synthesis

As discussed in section 1.5, for a radio telescope to have a resolution comparable to a large optical telescope, the required aperture is physically impractical to construct as a single element. Fortunately, unlike optical detection, the full EM wave information can be preserved in electrical form at radio frequencies, allowing coherent processing of the received EM wave. Coherent processing of the signals received at two spatially separated radio antennas can be used as a basis for synthesizing a very large effective receiving aperture. The coherent

processing used, is correlation. In simple terms, correlation is interference by multiplication of the two received signals, followed by an averaging filter.

The correlation of the signals from two receivers separated by a distance (d), called the baseline, provides one sample of the Fourier transform of the spatial intensity distribution in the field of view [11]. Long baselines give high spatial frequency components or fine image detail and short baselines give low spatial frequency components or coarse image detail. A two dimensional image of the sky requires Fourier components in two perpendicular coordinates. These two coordinates, when corrected for the direction that the elements are pointing, are known as the **u** and **v** coordinates. Ideally the number and spacing of baseline pairs should result in as many evenly spaced Fourier components as possible while minimising baseline spacing redundancies [11]. The number of Fourier components required may be limited by symmetries in the shape of the radiating source in the field of view and the required image resolution. Achieving the optimum number and spacing of Fourier components is called filling the **uv** plane [7]. This provides a complete set of evenly spaced samples, up to some resolution limit, of the Fourier transform of the sky intensity distribution. This can be inverse transformed to an image. Often, due to various limitations, a very limited number of baselines are constructed, however various interpolation and image refinement techniques can be used to produce reasonable images with relatively few baselines [7], [11].

Chapter 2

Context and scope of the research project

2.1 Introduction

This research project forms part of the work of the Radio Astronomy Technology (RAT) centre research group in the department of Electronic Engineering at the Durban University of Technology (DUT). The RAT centre began with the construction of a single element 5 m parabolic dish radio telescope, seen in Figure 2.1. The receiving electronics mounted at the focal point was designed to operate at the Hydrogen line frequency of 1,42 GHz. This radio telescope is called Indlebe, which is the Zulu word for ear. Indlebe remains the flagship project of the RAT centre.

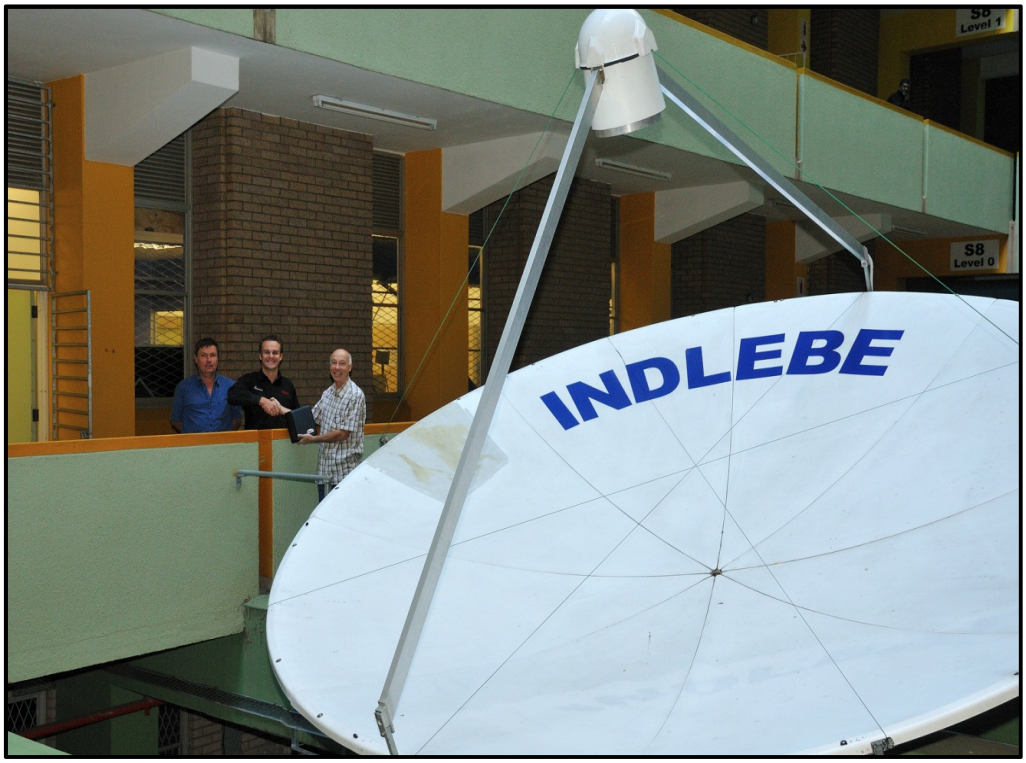


Figure 2.1: Durban University of Technology's Indlebe Radio Telescope antenna consisting of a 5m diameter parabolic dish reflector and a prime focus feed horn.

In keeping with international trends in radio astronomy and especially in light of South Africa's successful bid to host a very large radio telescope array called the Square Kilometre Array (SKA), the group's activities have extended to working with multi-element telescope arrays.

2.2 The aim and purpose of the research project

The aim of this research project was to develop an experimental two element correlating radio telescope interferometer for the purpose of entry level experimentation. The purpose was capacity building; firstly, knowledge and human capital development, and secondly, experimental equipment development. To this end, a 'grass roots' approach was taken. A two element correlating radio telescope interferometer appropriate for experimental work at the DUT RAT centre was developed from first principles. The process was investigative and experimental and not intended to produce new 'cutting edge' technological solutions but to investigate and experiment within the budget and current capability limits of the RAT centre.

In line with the discipline of engineering, a key focus of the project was to design and build (engineer) a piece of equipment. In addition to this focus there is a cross disciplinary element involving the integration of some knowledge of physics and astronomy. Table 2.1 shows the knowledge fields applied in this research project from the two disciplines identified.

Table 2.1: Knowledge fields applied in this research project.

Engineering	Physics and Astronomy
Radio frequency communication systems	EM radiation and wave propagation
Digital electronic systems and circuits	Interferometry
Microprocessors and microcontrollers	Telescopes and radio telescopes
Field programmable gate arrays	
Digital signal processing	
Printed circuit board design	
C language programming	

Design is a key component of the discipline of engineering. Without a key focus on design, the study ceases to be engineering. Measurement and observation are key components of the discipline of astronomy. The focus on experimental equipment development in this research project, combines the application of engineering design, centred around FPGA based digital logic, with an investigation of interferometry measurement and observation techniques in radio astronomy.

2.3 Astronomy context

Radio astronomy provides a set of measurement and observation tools within the discipline of astronomy. Optical astronomy makes use of optical telescopes to examine EM radiation from the cosmos that lies within the narrow band of visible EM radiation from around 700 nm (red) to around 400 nm (violet) wavelength, as can be seen in Figure 1.1. The information gathered, is used to identify objects, estimate their size, mass, chemical composition, motion, and distance from Earth and from each other [3].

A radio antenna can be viewed as analogous to an optical telescope when used to measure the intensity distribution of a band of EM radiation as a function of solid angle of the sky. This

intensity distribution is used to identify objects or groups of objects or clouds of gaseous matter in the cosmos. The angular resolution of the instrument used to measure the intensity distribution determines the ability to discriminate detail about the composition of very distant sources in the cosmos. As discussed in section 1.5, the angular resolution of a radio antenna is determined by the physical size of the antenna or radiation collecting diameter relative to the wavelength of radiation being detected. The shortest wavelength of EM radiation in the Radio Window (~1 mm) is more than three orders of magnitude greater than wavelengths in the Optical Window [3]. This means that radio antennas used as telescopes need to be very large compared to their optical counterparts if they are to achieve comparable angular resolution. It is not physically practical to construct an antenna of this size.

Interferometry and radio telescope arrays can be used to improve the angular resolution of a radio telescope compared to single element radio telescopes. The resolution of an array is proportional to the received wavelength divided by the largest distance between any two elements of the array [1]. For a single element radio telescope, the resolution or beamwidth, is proportional to the received wavelength divided by the aperture of the receiving antenna [9]. The distance between two receiving antennas in a radio telescope array can be made much larger than the aperture of any single element radio telescope antenna resulting in much better resolution. Surface radio telescope arrays have already achieved angular resolutions of at least an order of magnitude better than the best surface optical telescope resolution [12]. Very high resolution radio telescopes are often required to be very sensitive and hence are plagued by local and system induced noise as they attempt to resolve more and more detailed information from, often very weak and very distant, sources of EM radiation [7].

2.4 Electronics engineering context

Modern digital signal processing (DSP) techniques involving hardware and software are being used to improve the correlation process used in radio telescope arrays. A key hardware component that is gaining increasing popularity in radio astronomy is the field programmable gate array (FPGA). This silicon integrated circuit (IC) contains a very large amount of digital logic circuitry. These logic circuits consist of a large number of basic logic circuit elements called gates that can be interconnected (configured) in an almost infinite number of ways. The configuration details, called gateware, are programmed into the device. Most FPGA's have volatile configuration which means that the configuration is lost when the power to the device is removed. The device must be reconfigured every time it is powered up. This allows orders of magnitude more configurable digital logic to be implemented in the available silicon than could be achieved with non-volatile configuration logic. A microprocessor by contrast has fixed digital logic configuration called architecture. The processing flexibility is achieved by compiling a sequence of software instructions that manipulate data, using the fixed logic architecture, in a repetitive sequential manner.

FPGAs are popular for DSP applications because the logic circuitry can be configured to be optimized for the given application. This usually involves extensive parallel processing, unlike microprocessors that perform sequential processing. The parallel processes can be tailored to meet the exact needs of the task resulting in very efficient use of the hardware resources. Many logic and arithmetic processes are executed simultaneously at very fast hardware speeds, unlike microprocessors where logic and arithmetic processes involve a sequence of program instructions with only one instruction executed at a time. Some parallel processing and dedicated hardware resources are often integrated into modern microprocessor ICs. An interesting example is a specific class of microprocessor called Graphical Processing Units (GPUs). GPUs are very commonly used for implementing fast graphical processes in

PCs. GPUs are also used in experimental radio telescope arrays but since they are not designed for this purpose their effectiveness is limited by the fixed hardware architecture. Designing FPGA gateware requires a fundamental understanding of digital logic circuitry and the way in which digital logic circuits are combined to perform combinational digital logic functions. GPUs on the other hand are more closely related to microprocessors and programmed sequential digital logic.

While FPGAs are excellent for implementing DSP processes, they are very clumsy for implementing the more conventional digital processing tasks such as data input and output interfacing, particularly interfacing to a PC, because there is no fixed hardware configuration to take care of this. The task of PC interfacing is best performed by a microprocessor or microcontroller since it does not require extensive parallel processing or very fast hardware logic speed and microprocessors and microcontrollers usually have convenient PC interfacing hardware included in their architecture.

2.5 Scope of the research project

The aim of this research project was to produce a functional two element correlating radio telescope interferometer for the DUT RAT centre. As much software and hardware as possible was developed from scratch in order to achieve this purpose. Some hardware and software resources have been developed by open source software communities for use in high-end operational radio telescopes. There is however little open source resources available for entry level radio telescope experimental work. It was decided that the Universal Software Defined Radio Peripheral (USRP2) platform would be the best fit for the required application. Other digital hardware platforms were investigated for resource and performance comparison but all were out of the reach of the available budget for this research project. The USRP2 platform has a Xilinx Spartan3 FPGA and a dual 100 Msample/s analog to digital converter

(ADC) on the motherboard. There are a variety of plug-in radio receiver daughterboards available for this platform. Xilinx FPGA gateware configuration development tools were donated by Xilinx to support the planned development.

The required experimental outcome was to produce correlating interferometer fringes from a pair of UHF band receiving radio antennas. The results for some radiating test source at various radio frequencies in the UHF band and for short baseline distance would be recorded and analysed according to theoretical expectation. Instrumental phase offset for the signals of the baseline pair would be carefully characterized and documented. Techniques for calibration of gain and instrumental phase offset would be investigated.

Chapter 3

Summary of mathematical processes

3.1 Introduction

In this chapter, the key mathematical processes involved in radio telescope arrays are identified and discussed. Full mathematical derivation of all formulae is not given, however some derivation steps have been included to illustrate some limitations and give insight into how the mathematics may be applied.

The fundamental problem of radio telescopes compared to optical telescopes is poor resolution. In order to achieve resolution comparable to optical telescopes, radio telescopes need very large apertures. Achieving this with a single element receiver is not practical. The solution is to exploit the ability of electronic circuitry to process the full wave information of the received signal. The basic processing element used, is a two element correlating interferometer. A two element correlating interferometer performs the mathematical function of cross-correlation on an incoming Radio band EM wave, received and converted to a coherent electrical signal at two spatially separate locations. The distance (d) between the two receivers is called the baseline distance of the receiving pair [7], [12].

3.2 Interference

Interference occurs when two waves occupying the same region of a medium combine [5], [6]. The combination may be the sum of the amplitude of the waves or it may be the product. An Excel spreadsheet simulation was used to analyse the resultant wave function when two sinusoidal wave functions are combined, firstly by sum and secondly by product. See annexure C for interference by product spreadsheet details.

3.2.1 Interference by sum

In Figure 3.1, two waves of the same amplitude and slightly different frequency are added. The slight frequency difference results in 360 degrees of phase shift between the interfering wave functions over ten cycles. When the two waves are in phase, there is constructive interference and the resultant sum is twice the amplitude of one of the source waves. When the two waves are 180 degrees out of phase, the resultant sum is zero. This is complete destructive interference.

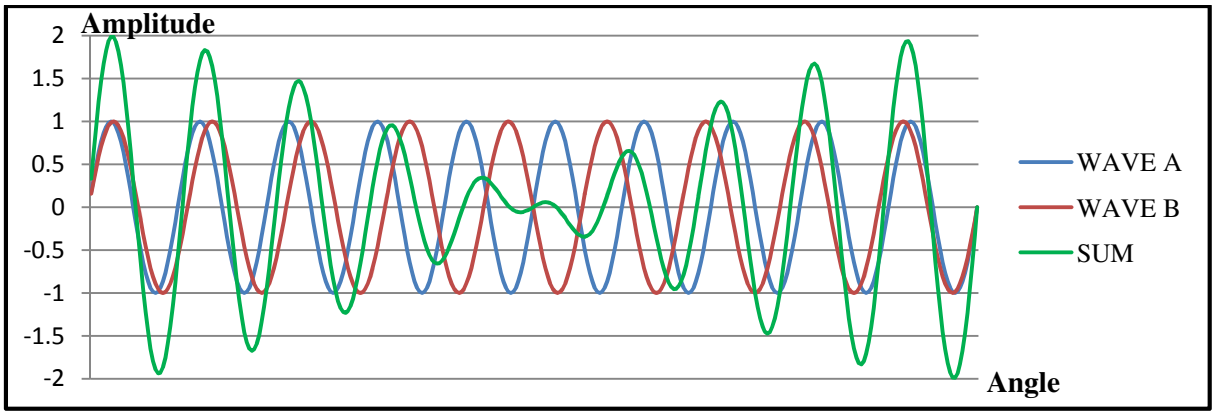


Figure 3.1: Interference by sum: An Excel simulation plot showing interference by sum for two sinusoidal wave functions that are close in frequency. The simulation shows the familiar beat frequency effect that arises in the sum function, resulting from constructive and destructive interference.

3.2.2 Interference by product

In Figure 3.2, two waves of the same amplitude and slightly different frequency are multiplied. The resultant product has been scaled by a factor of 0,5 to make the graph clearer. In frequency component terms, the product has two components. A high frequency component equal to the sum of the frequencies of wave A and B, and a low frequency component equal to the difference between the frequencies of wave A and B. This arises from the trigonometric identity for the product of two sinusoidal functions:

$$\sin(X)\sin(Y) = \frac{1}{2}[\cos(X - Y) - \cos(X + Y)] \qquad 3.1$$

Computing a moving average of suitable length (discussed further in sections 3.8 and 7.3.6), of the product function will filter out the high frequency component and leaves only the low frequency component. The low frequency component tracks the difference in phase between the two waves A and B, as can be seen in Figure 3.2. The moving average is zero when the difference in phase between waves A and B is plus or minus 90 degrees. It is at maximum when A is in phase with B and at minimum when A is 180 degrees out of phase with B.

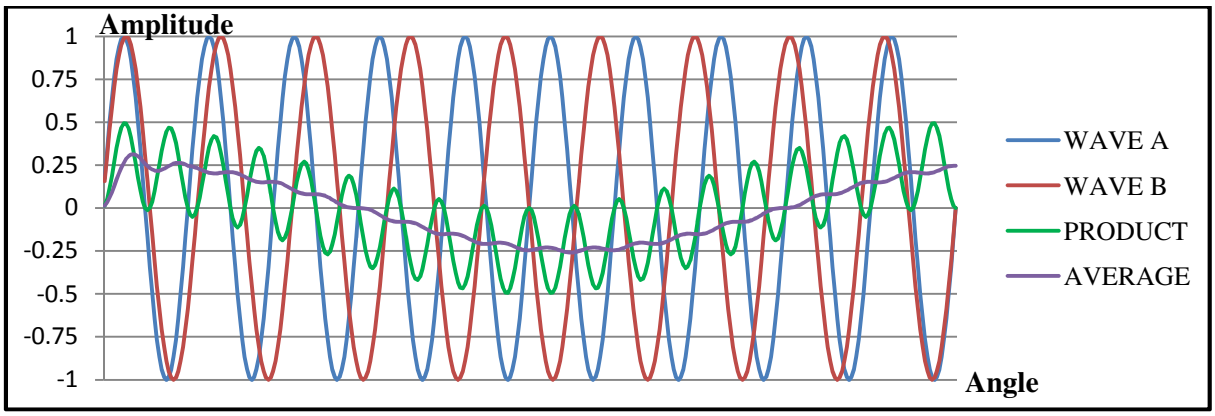


Figure 3.2: Interference by product: An Excel simulation plot showing interference by product for two sinusoidal wave functions that are close in frequency. The simulation shows the sum and difference frequency components that arise in the product function and the attenuation of the sum frequency component by the application of a moving average filter process.

3.3 Correlation

The average, in Figure 3.2, is the correlation of wave A and wave B over a limited integration interval. In Figure 3.2, the phase difference between the interfering wave functions is made to vary a complete 360 degrees over ten cycles of wave A, in order to illustrate the complete range of the correlation function. Correlation is the process that is used in radio telescope interferometers. The varying phase difference between two signals is obtained by a varying path difference between the source and two receivers that varies as the Earth rotates. This phase variation is very slow, in terms of change of phase lag per cycle of the source waves, compared to the Excel spreadsheet analysis above, allowing a much longer integration interval to be used relative to the period of the wave functions being correlated. This will

result in a much smoother correlation function compared to the average shown in Figure 3.2 (see simulation test results in section 8.3).

3.4 Interference fringes

A good illustration of interference fringes is Young's Double Slit Interferometer experiment [5], illustrated in Figure 3.3, which was used by Young to demonstrate the wave nature of light. Two slits in a screen near a light source generate two point sources of coherent light. Coherent meaning they are in phase and have the same wave shape at the slit. When the light from the two slits reaches a target screen some distance away, it forms a pattern of light and dark bands, called fringes.

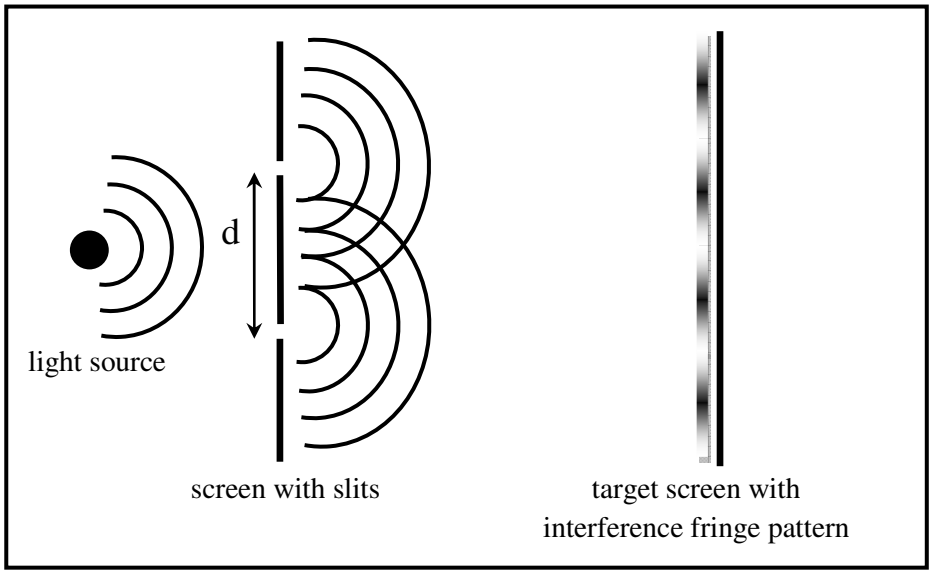


Figure 3.3: Young's double slit interferometer experiment.

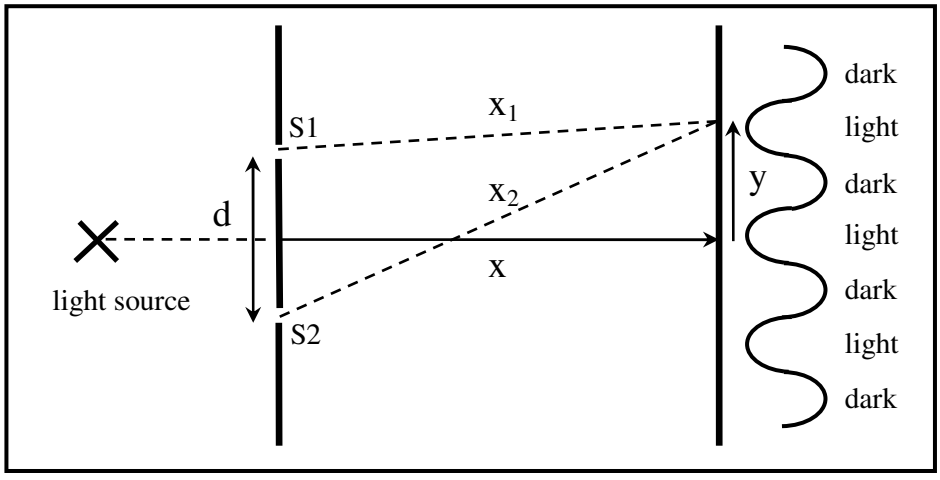


Figure 3.4: Young's double slit interferometer experiment geometry.

Figure 3.4 shows how the geometry of the experiment gives rise to the interference fringes on the target screen. Differences in path length between slit 1 and slit 2 and some point on the target screen result in the waves from slit 1 and slit 2 arriving completely in phase at some points and completely out of phase at other points on the target screen. The resultant constructive and destructive interference produces light and dark bands or fringes respectively.

3.5 The fringe visibility function

When incoming radiation from a source in the cosmos is received at two spatially separated receivers, as shown in Figure 3.5, correlation of the coherent electrical signals produces an interference fringe function which represent a component of the Fourier transform of the spatial intensity distribution of the source in the field of view [11], [12]. The mathematics is formally described by the van Cittert-Zernike theorem shown in equation 3.2 [12], [13]:

For two antennas positioned at vector positions $\mathbf{r_1}$ and $\mathbf{r_2}$ respectively, the spatial coherence function (also called the visibility function) of the source intensity distribution ($V(\mathbf{r_1},\mathbf{r_2})$) is equal to the cross-correlation of the received electric fields at $\mathbf{r_1}$ and $\mathbf{r_2}$.

$$V(\mathbf{r_1}, \mathbf{r_2}) = \langle \mathbf{E}(\mathbf{r_1})\mathbf{E}^*(\mathbf{r_2}) \rangle \qquad 3.2$$

* – denotes complex conjugate

$\langle\ \rangle$ – denotes expected value or average

Closely spaced receivers produce low spatial frequency components of the source intensity distribution over the field of view. This is coarse image detail. Widely spaced receivers produce high spatial frequency components of the source intensity distribution over the field of view. This is fine image detail.

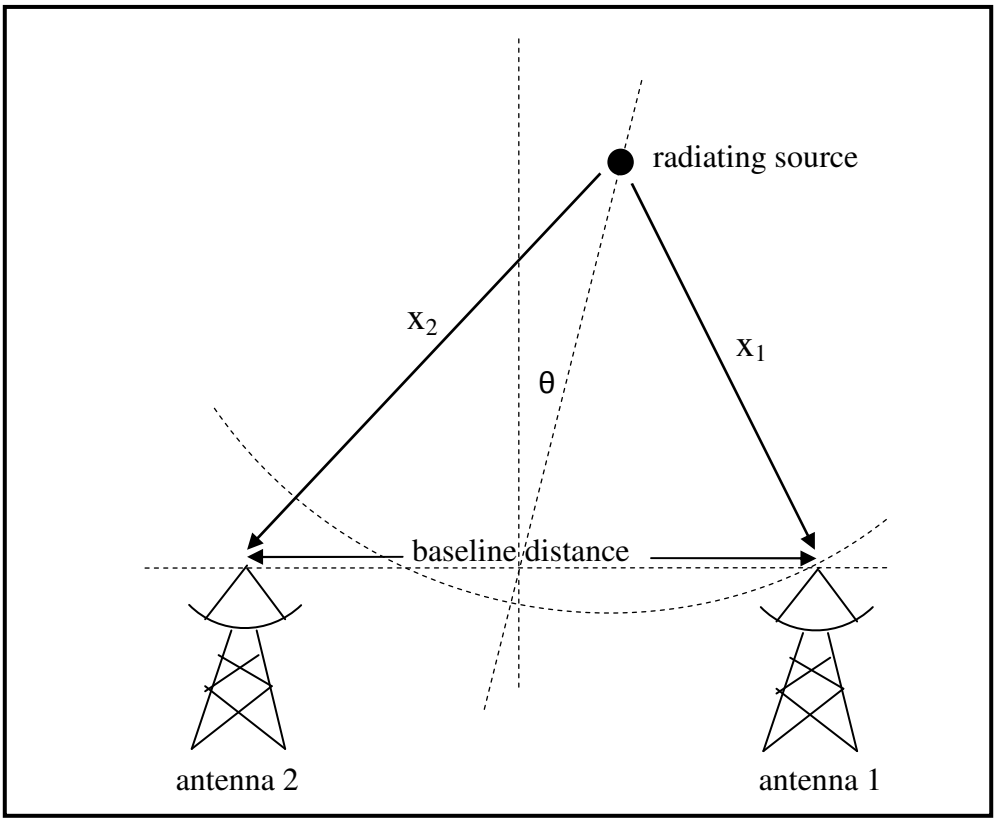


Figure 3.5: Antenna baseline pair showing the basic geometry of the baseline distance and the path length difference between some radiating source and each of the two antennas. Spherical wavefront and corresponding radial rays are shown but for a very distant radiating source, a plane wavefront approximation is used, as shown in Figure 3.7.

The definition of fringe visibility comes from optical interferometry and fringe pattern analysis used in the Michelson interferometer method of star angular size measurement [5]. The Michelson fringe visibility V_M is a measure of the difference in intensity between the light bands and the dark bands of an optical fringe pattern and is given by [5], [12]:

$$V_M = \frac{intensity\ of\ maxima - intensity\ of\ minima}{intensity\ of\ maxima + intensity\ of\ minima} \qquad 3.3$$

The van Cittert-Zernike theorem (equation 3.2) shows that the cross-correlation of two time based signals representing the coherent detection of the EM wave from a radiating source received at two locations separated by a distance d, produces the Fourier components of the spatial intensity distribution of the radiating source [12], [13]. The cross-correlation produces

a fringe oscillation function, the amplitude and phase of which define a Fourier component of the spatial intensity distribution. The fringes, or Fourier components, are sinusoidal functions with amplitude and phase. The amplitude and phase can be represented by a single complex number resulting in a complex visibility function which is a complex valued function of the separation distance (d) between the two receivers. The separation distance is called the baseline.

In practice, a limited number of samples of the complex visibility function are obtained by a finite number of different baselines [11]. The complete complex visibility function can be estimated by interpolation between a finite number of samples [12]. The inverse Fourier transform of the complex visibility function is the spatial intensity distribution, which is an image of the source. For simple source geometries, a complete inverse Fourier transform may not be necessary and the required information about the geometry and intensity distribution of the source may be inferred from the visibility function samples directly. Figure 3.6 illustrates the relationship between fringe visibility and baseline distance for a source of simple circular geometry. The visibility function has a sin(x)/x form. The optical fringes of Figure 3.6 are obtained using equipment called the Michelson stellar interferometer which combines the light received from a source at two lenses separated by a baseline distance (d) [14]. The sign change in the values of the visibility function indicate a fringe phase reversal as can be seen in Figure 3.6. The complex visibility function provides both the amplitude and the phase of the fringe visibility simultaneously. Note that the fringe spacing decreases with increasing baseline. The diameter of a circular source may be calculated from the baseline distance at which the first null in fringe visibility occurs [14]. This is analogous to the relationship between a rectangular pulse and its Fourier transform, of sin(x)/x form.

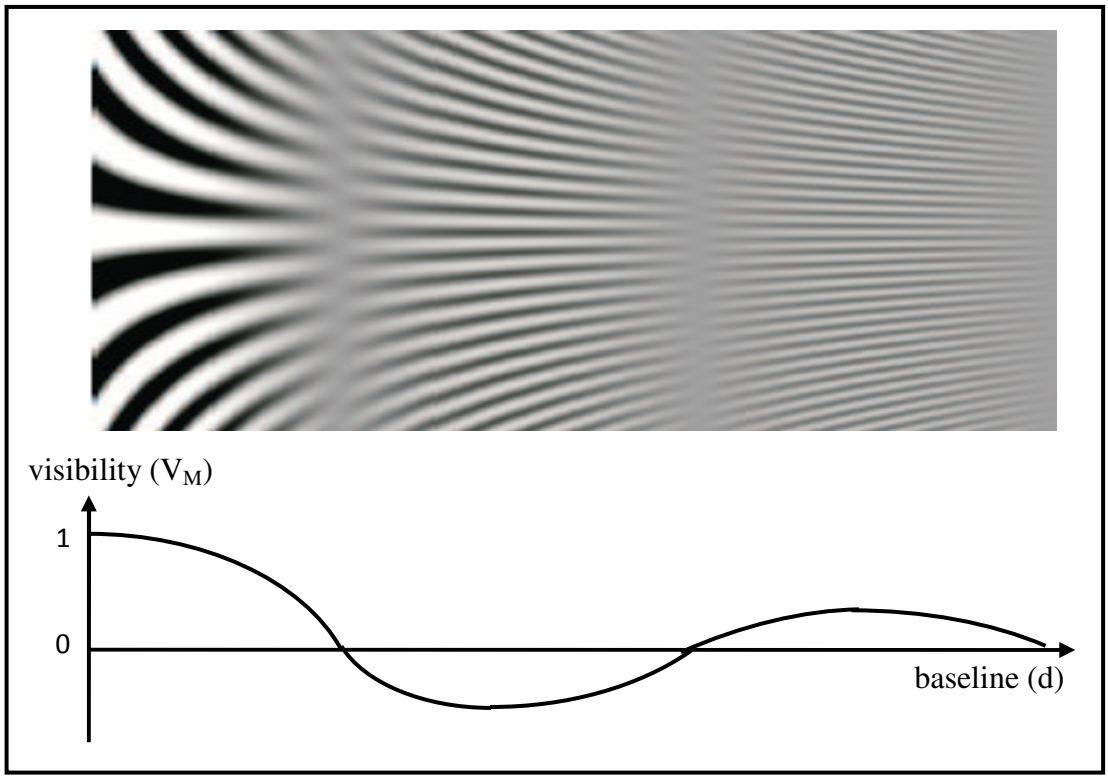


Figure 3.6: Fringe visibility versus baseline for a source of circular geometry showing the sin(x)/x visibility function shape and corresponding fringe visibility and phase reversal.

3.6 Mathematical cross-correlation

Mathematically, cross-correlation is a transformation of two input functions of a common variable to an output function of another variable. In the case of radio telescope signals, the two input functions are voltage functions of time. Since both input signals represent the same source (some radiating portion of the sky) but measured at different physical positions, the signals will have the same shape but may be offset in time relative to each other, dependent on the angular position of the source relative to the two receivers.

The cross-correlation of two functions f(t) and g(t) of time produces an output function ($r(\tau)$) which is a function of the time lag (τ) between the two functions f(t) and g(t). For each value of time lag, the integral of the product of the two functions f(t) and g(t-τ) produces one value of the output cross-correlation function.

$$r(\tau) = \int_{-\infty}^{+\infty} f(t)g(t-\tau)\,dt \qquad 3.4$$

In the case of a two element radio telescope interferometer, the lag between the two received signals is obtained by the path difference between the observed source and the two receivers. The arrival of the EM wave, travelling at the speed of light, at one receiver will be offset by a time lag relative to the other receiver. The lag is proportional to the angular position of the source relative to the baseline of the receivers and to the baseline distance between the receivers. Considering each resolvable element of radiating source in the field of view, each element will lie at a different angle relative to the baseline of the receivers. The signals received at the receivers due to each element will therefore have different lags. This relationship between the spatial distribution of radiation across the field of view and the correlation of the signals from two spatially separate receivers is described mathematically in terms of the quasi-sinusoidal output obtained from the correlation of the two baseline signals. The amplitude and phase of the quasi-sinusoidal output of the correlation forms a sample of a function called the fringe visibility function. The fringe visibility function is a function of the baseline distance between the two receivers. So another transformation has transformed the correlation function of signal lag to a fringe visibility function of baseline distance. Each sample of the fringe visibility function, that is the amplitude and phase of the fringe oscillation function, for each baseline distance, constitutes a component of the spatial Fourier transform of the sky brightness distribution in the field of view. The fringe visibility function is determined in each of two perpendicular coordinates and a two dimensional inverse Fourier transform of these will produce a two dimensional spatial image of the sky brightness.

There are many practical limitations and approximations that apply when implementing an electronic system according to the mathematical theory described above. This research project investigated some of the practical limitations and approximations involved in obtaining a

single value of a fringe visibility function. That is, a fringe amplitude and phase for a single baseline distance. In practical radio telescopes, visibilities for many baselines are simultaneously measured by combining signals from pair combinations of a large number of telescope elements [11]. This process aims to produce as close to a complete two dimensional visibility function as possible. The technique is called aperture synthesis. Ideally, as many evenly spaced samples of the visibility function is needed, in each of two perpendicular axes, as discussed in section 1.7. This poses a significant digital processing challenge and large-N radio telescope arrays require computing capacities of 10^{13} to 10^{14} ops/second or more [10], [15], [16]. In many current radio telescope arrays, only a relatively small number of baselines are implemented and this and other practical limitations and approximations are the subject of aperture synthesis and image formation algorithms [7], [11], [12], [17]. This is outside the scope of this study which aims to investigate the correlation of a single baseline pair.

3.7 Baseline geometry and path difference delay

In Figure 3.7, the incoming EM wavefront is taken to be a plane wavefront. This approximation holds true since the receiving aperture (the baseline) will always be very small compared to the very distant (astronomical) sources. For closer sources, the spherical shape of the wavefront may need to be considered. The wavefront rays in Figure 3.7 point in the direction of propagation of the approximated plane wave, normal to the plane wavefront surface.

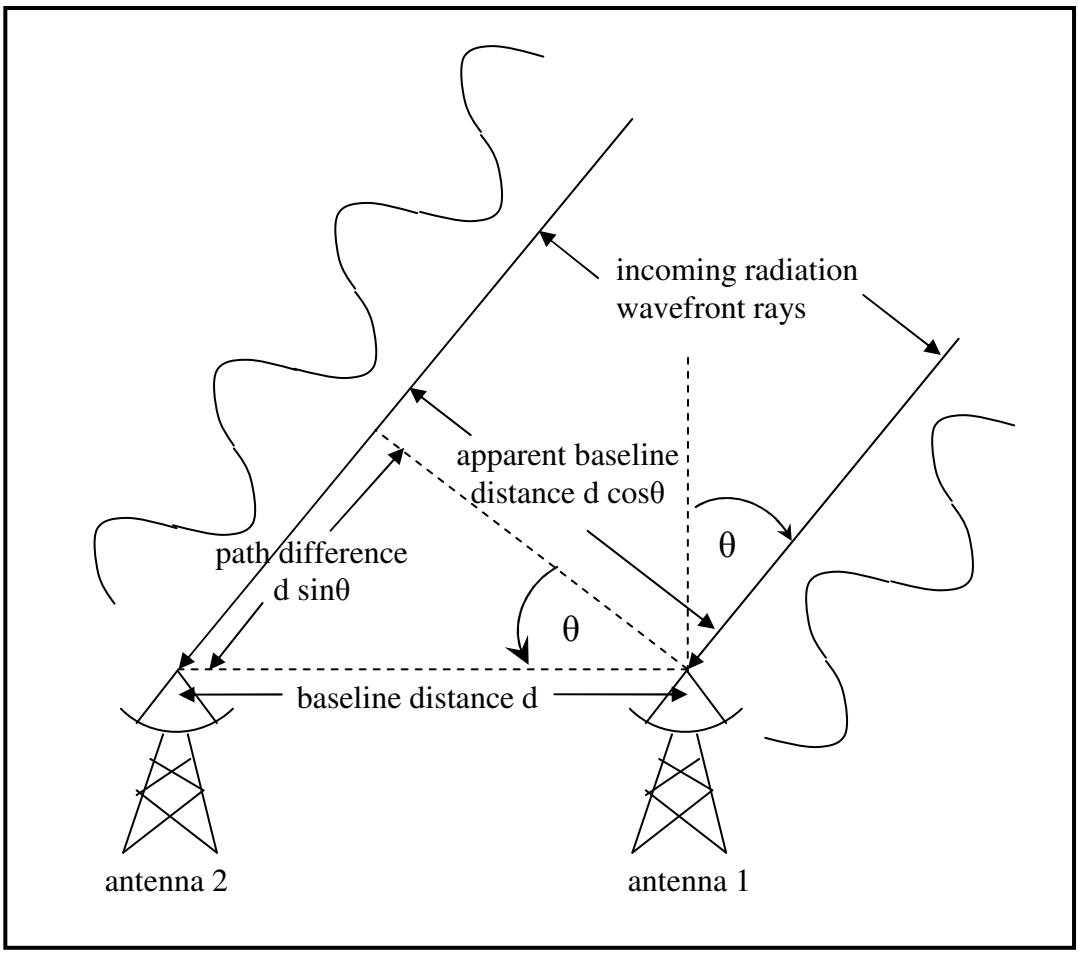


Figure 3.7: Two element interferometer geometry illustrating the relationship between baseline distance d, source angle θ and the resultant radiation path length difference for a very distant radiation source with corresponding plane wave front (parallel rays) assumption. The apparent baseline distance for fringe measurements of sources not close to zenith is shown.

As can be seen in Figure 3.7, there is a path difference related to the baseline distance and the angle of the source position in the sky, relative to zenith. Figure 3.7 is a sectional view and the angle of the source is represented in only one dimension. The full reality is a two dimensional sky but this is usually decomposed into two perpendicular sets of baselines in coordinates called **u** and **v**.

The path difference (d sin(θ)) can be expressed as a geometric time delay (τ_g) using the wavefront speed, the speed of light. The speed of light in air is only slightly slower than the speed in a vacuum which is denoted by the symbol c.

$$\tau_g = \frac{d \sin(\theta)}{c} \qquad 3.5$$

The amount by which EM wave propagation slows down through the Atmosphere is medium dependent, as explained in section 1.2. Since the Atmosphere is a dynamic mixture of gases, an EM wavefront may experience different Atmospheric delays when arriving at different antennas. The fluctuations in phase difference caused by the EM wave passing through different atmospheric compositions in travelling to one receiver compared with the other, becomes more problematic the longer the baseline. This is the topic of atmospheric phase coherence [11] which is beyond the scope of this study.

3.8 Obtaining the fringe visibility by cross-correlation

Referring to the diagram of Figure 3.7, in this research project a sample of the fringe visibility function of the source is obtained by correlation of the signals received at antenna 1 and antenna 2, separated by a distance d. This is achieved by implementing two digital processes in an FPGA. The first is multiplication of the signal samples from antenna 1 with the signal samples from antenna 2. This element of correlation is seen in equation 3.4 as the product of f(t) and g(t-τ). The second process is a moving average filter that finds the average of a finite number of the product terms. The filter implements the integration seen in equation 3.4. The average is found by adding a finite number of product terms and dividing by the number of terms. The FPGA output is a finite interval approximation of the correlation of the two received signals for a specific value of lag determined by τ_g. The process is continually repeated each time there is a new sample of the two signals. As time passes, the source angle changes due to Earth rotation, thereby changing the lag or phase difference between the signals. The FPGA correlator output therefore gives a complete correlation of the two signals over time. The varying lag is provided by the varying path difference as the source moves across the sky.

Analysing the output of the correlator (F) for a single frequency sine wave of frequency (v) received at antennas 1 and 2 of Figure 3.7 we have, before filtering [12]:

$$\begin{aligned} F &= \sin(2\pi\nu t)\sin\left(2\pi\nu t - 2\pi\nu\tau_g\right) \\ &= \sin(2\pi\nu t)\left(\sin(2\pi\nu t)\cos\left(2\pi\nu\tau_g\right) - \cos(2\pi\nu t)\sin\left(2\pi\nu\tau_g\right)\right) \\ &= \frac{1}{2}\left(\left(1 - \cos(4\pi\nu t)\right)\cos\left(2\pi\nu\tau_g\right) - \sin(4\pi\nu t)\sin\left(2\pi\nu\tau_g\right)\right) \\ &= \frac{1}{2}\left(\cos\left(2\pi\nu\tau_g\right) - \cos(4\pi\nu t)\cos\left(2\pi\nu\tau_g\right) - \sin(4\pi\nu t)\sin\left(2\pi\nu\tau_g\right)\right) \end{aligned} \quad 3.6$$

The second and third terms of equation 3.6 are filtered out by the moving average filter. Ignoring the scale factor of a half we are left, after filtering, with [12]:

$$F = \cos\left(2\pi\nu\tau_g\right) \quad 3.7$$

Substituting equation 3.5 into equation 3.7 we get:

$$F = \cos\left(2\pi\nu\frac{d\sin(\theta)}{c}\right) = \cos\left(\frac{2\pi\, d\sin(\theta)}{\lambda}\right) = \cos\left(\frac{2\pi\, d\, l}{\lambda}\right) \quad 3.8$$

Where:

$$l = \sin(\theta) \quad 3.9$$

Sin(θ) is approximately equal to θ for small θ and so the correlator output is a quasi-sinusoidal fringe; a cosine function of θ which varies slowly as the Earth rotates.

Equation 3.8 is valid for observing sources that transit directly overhead with the baseline antennas pointing straight up to zenith. In most cases however the desired field of view is at some elevation other than zenith and it is not necessary for the source to pass overhead before

fringes can be obtained. In this case the baseline distance is resolved into the distance d cos(θ) which is the apparent spacing of the baseline relative to an offset field of view, as shown in Figure 3.7. The geometric time delay τ_g is then cancelled out with delay adjustment within the baseline pair receiver electronics and the baseline pair then responds only to small Δθ of the source intensity distribution within the field of view. The apparent baseline distance is then also converted into wavelength units at the wavelength of observation and a new quantity *u* is defined, given by [12]:

$$u = \frac{\mathrm{d}\cos\theta}{\lambda} \tag{3.10}$$

Equation 3.9 becomes:

$$l = \sin\Delta\theta \tag{3.11}$$

Equation 3.8 becomes:

$$\mathrm{F} = \cos(2\pi u l) \tag{3.12}$$

In a 2-dimensional radio telescope array, each element of sky intensity in the field of view is resolved into two perpendicular dimensions called *l*, as in equation 3.11, and *m*. *l* and *m* are the sine of the source angle resolved in two perpendicular coordinates according to two perpendicular apparent baselines called *u* and *v* [12].

Equation 3.12 is the function defining the fringe obtained by baseline *u*. The amplitude and phase of this fringe gives one sample of the fringe visibility function V(*u*). Obtaining fringes at different baseline distances (*u*) gives other samples of the visibility function V(*u*). When fringes are obtained in two dimensions *u* and *v*, the result is a visibility function of two variables V(*u*,*v*).

Chapter 4

Survey of available hardware and software platforms

4.1 Introduction

Digital technology has advanced to the point where digital techniques are replacing most of the processing that was previously achieved with analogue circuits, reducing the cost and development time thus making solutions accessible in shorter times and with smaller budgets [15], [16].

Real time digital cross-correlation at radio frequencies in radio telescope arrays is efficiently achieved using an FPGA. The FPGA gives the advantage of digital hardware speed and extensive parallel processing at digital gate level. This makes efficient high speed real time DSP, such as radio frequency digital correlation, possible. Only a dedicated ASIC could achieve better processing speed and silicon area efficiency but field programmability and reconfigurability are sacrificed. These features of FPGA's are essential in the fast evolving and highly developmental field of digital correlators for radio telescopes [15], [16].

FPGA's are gaining widespread use as a versatile digital signal processing platform. The Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) that began at the University of California, Berkeley has been a major contributor and coordinator of developments in the field of radio astronomy, particularly DSP on FPGA [13], [15], [16] [18], [19].

The development of FPGA based hardware platforms is a very specialized and time consuming business. Typically FPGA's use ball grid array (BGA) technology for bonding of the IC connections (pins) with the PCB. Pin counts are in the several hundreds and IC package sizes are getting smaller. The precision required for PCB tracking design, manufacture and assembly is now well out of reach of all but a few highly specialized PCB manufacture and assembly companies. This has led to the availability of a number of general purpose FPGA development platforms [15], [16], [20], [21]. These hardware platforms are also extensively used as prototype platforms for ASIC development where the advantages of field programmability and reconfigurability during prototype development dramatically reduce the development cost and turnaround time [21], [22].

4.2 Hardware platforms

The BEE (Berkeley Emulation Engine) board developed at the University of California Berkeley, targets high end professional development of complex FPGA based systems. ROACH (Reconfigurable Open Architecture Computing Hardware), developed by SKA in South Africa has aimed at being more affordable with more modest resources but still capable of advanced high speed DSP. RHINO (Reconfigurable Hardware Interface for computatioN and radiO), developed at the University of Cape Town in South Africa has targeted low cost and medium DSP performance. The USRP (Universal Software defined Radio Peripheral) board from Ettus Research was developed for lower-end software defined radio (SDR) work in universities and other research environments and has only basic DSP capability [21]. The FPGA on the USRP1 is an Altera Cyclone whereas on the newer version USRP2, a Xilinx Spartan3 FPGA is used. The BEE, ROACH and RHINO boards all use Xilinx FPGAs.

Table 4.1 shows a comparison of various resources of the hardware platforms discussed above. The USRP2 provides on-board ADC resources for two RF input channels whereas the other three development boards provide plug-in slots for external ADC boards [21]. By comparison to the other three platforms shown in table 4.1, the USRP2's performance and resources are much lower. The performance and resources available on the USRP2 board was however sufficient for the intended research project. Because the USRP2 has a more modest FPGA and an onboard ADC compared to the other hardware platforms described in table 4.1, this hardware platform would be more cost effective for this research project and would require less development time to master. The USRP2 was therefore chosen as the hardware platform for this project.

Table 4.1: Comparison of the resources of some fpga dsp development boards.

BOARD → / **RESOURCE ↓**	**USRP2**	**RHINO**	**ROACH**	**BEE4W**
FPGA	SPARTAN3	SPARTAN6	VIRTEX5	4 x VIRTEX6
external memory	none	512MB DDR3	2 x 4 Mb QDR2 512MB DDR2	4 x 32GB DDR3
microprocessor	none	TI SITARA	PowerPC	none
external memory	none	256MB DDR2	512MB DDR2	none
ADC	on board	plug in FMC	plug in Z-DOK	plug in FMC
channels/precision	dual 14 bit	up to 4 ch/ 14 bit	Up to 4 ch/14 bit	4 x 2ch. typ. 14 bit
speed	100 Msps	typ. 1 Gsps	typ. 1 Gsps	typ 1 Gsps
approx. cost	$1200	$1700	$5300	$25000

Key factors in assessing the resources of FPGA development boards for more advanced DSP are the availability of fast external memory for the FPGA based DSP processes and the inclusion of a suitable onboard microprocessor. There should also be sufficient external memory resources to support an operating system for the microprocessor that will facilitate communication with the board and provide an interface to the FPGA for initial configuration, on-the-fly configuration changes as well as the control of input and output data. The USRP2 has neither fast external memory nor an onboard microprocessor and is therefore limited to fairly low level DSP functionality. The absence of an onboard microprocessor for the USRP2 was solved in this research project by developing a simple PIC microcontroller based plug-in daughterboard. It would have been possible to implement a simple microprocessor in the FPGA gateware but this requires special development tools and uses up high speed FPGA digital resources that are best used to implement the required DSP. The FPGA based microprocessor would use approximately 33% of the USRP2 FPGA resources [21]. A standard microprocessor or microcontroller that is well documented and comes with a high level of software and development support is therefore better for implementing routine connectivity, communication, data flow and control functions.

4.2.1 The USRP2

The Universal Software Defined Radio Peripheral 2 produced by Ettus Research is the second generation of this hardware platform for software defined radio (SDR). Software defined radio (SDR) is the collective name for modern programmable digital hardware that is replacing analogue hardware in radio communication systems. The programmability allows a radio communication system configuration to be determined by ‘software’ and not by the hardware as was previously the case with analogue systems. This allows a system to be reconfigured for different applications using the same hardware. The hardware becomes a

platform on which to implement a variety of radio communication systems. The FPGA plays a central role in achieving reconfigurable hardware for fast DSP.

The USRP2 has very limited resources compared to other FPGA based hardware platforms but has the advantage that all essential resources are on-board. A selection of plug-in RF daughterboards is available for different front-end RF frequency bands and for various combinations of receivers and transmitters. The RF front-end daughterboards together with onboard dual ADC makes the USRP2 easier to use and quicker to get up and running for first time users compared to the other FPGA development boards surveyed in table 4.1. This was an important consideration in choosing the USRP2 for this research project as there was no history of use of such equipment at DUT to draw upon.

4.3 Software platforms

The hardware platforms discussed in section 4.2 are general purpose DSP hardware platforms. They are not limited to any particular DSP application and the ROACH board is the only one designed to target radio astronomy DSP. The total global demand for radio astronomy DSP is very small and is insufficient to make proprietary software platform development for radio astronomy viable. There is however a lot of open source software available for the various hardware platforms discussed in section 4.2. The GNU radio open source software development community have developed a number of applications for software defined radio (SDR) that targets the USRP1 and to a lesser extent the USRP2. These applications are aimed at modulation techniques in narrowband radio communication systems and are not suitable for radio astronomy where the focus is on digital correlation of wideband noise sources. The Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) is an open source software and gateware development community that develops radio astronomy gateware applications targeting the BEE and ROACH boards [13], [16], [18], [19], [20].

These applications are aimed at high-end operational radio telescope developments and the approach taken to implement correlation on FPGA is to first Fourier transform the input signals (this process usually requires at least one entire FPGA which is called the F-engine) and then to cross multiply all the frequency components of each input channel (this again requires at least one entire FPGA which is called the X-engine). This process, called FX correlation, is equivalent to time domain correlation. CASPER has developed a number of these FX correlator gateware applications that target the high end FPGA hardware platforms such as the ROACH and the BEE [15], [16], [18], [20].

Since the time domain correlation approach taken in this research project requires fairly simple DSP processes and there were no suitable open source applications for the chosen hardware platform, it was decided that the FPGA gateware should be developed from scratch. This would require basic competence in the use of Xilinx synthesis tools and the implementation of simple DSP processes on a Xilinx FPGA. An introductory Xilinx DSP on FPGA workshop was attended and the extensive workbook of tutorial exercises and DSP theory notes provided in this course was a valuable resource and reference for much of the work that follows [22].

4.3.1 FPGA synthesis tools

Modern FPGA's have a very large amount of fast digital resources which can be configured in many ways. The very high level of digital circuit integration is achieved using a fabrication process that requires volatile chip configurability. The FPGA internal circuitry can be configured to be connected in a certain desired way but when power is removed the configuration is lost. The FPGA must be reconfigured each time it is powered up. The

configuration information is therefore usually stored in an external EEPROM and loaded into the FPGA on power-up. Development of FPGA configurations, called gateware, requires development tools that are provided by the FPGA manufacturer or some third party gateware development tool provider. As with most programmable devices, a choice of device manufacturer tends to lock development into a subset of available development tools. FPGA configurations can be designed in high level hardware description languages such as VHDL or Verilog but these must be converted to a configuration file for the target FPGA by a process called synthesis, which is similar to compiling of high level microprocessor software. The synthesis tools are specific to a particular FPGA manufacturer.

The basic synthesis tool for Xilinx FPGA gateware development is called Integrated System Environment (ISE). As with development tools for many other types of programmable devices, ISE is an integration of a number of individual software tools into one GUI environment. ISE can accept source logic design information in a number of different formats. The source logic design information is then used to create a configuration file (.bit) for the target FPGA. There are usually many ways to implement the desired logic in the FPGA and together with user constraint information in a .UCF file, Xilinx Synthesis Technology (XST), which is part of ISE, will generate an optimized FPGA silicon configuration (.bit) file. The final step is to configure the target FPGA by running Impact, which is also part of ISE. Impact is a JTAG standard programming tool that identifies JTAG programmable devices on the target board and allows the user to select which .bit configurations should be loaded into which programmable device on the target board [22], [23].

ISE also provides a simulation tool as well as logic timing analysis and silicon usage analysis tools. For this research project the initial logic design was done using a graphical block schematic FPGA gateware design capture tool called System Generator. System Generator is

a Xilinx FPGA design tool plug-in for Matlab-Simulink. It provides libraries of logic blocks that can be placed and connected within Simulink. Input signals and output monitor blocks can be added to allow simulation of the design in Simulink. One of the blocks in the Simulink schematic is always the System Generator block. Clicking on this block allows the user to select between a VHDL or Verilog format (.vhd or .v) design output and to generate this hardware description language code from the block schematic design. The .vhd or .v file must then be opened in ISE as the starting point of the synthesis process for the target FPGA [20], [22], [24].

All the tools discussed above and their features were more than sufficient for the simple DSP on FPGA design required for this research project. Xilinx generously donated a full system version of ISE 13.1 to DUT for the purpose of this research project. A full Matlab-Simulink software package with Xilinx System Generator plug-in was purchased from Mathworks. For more complex DSP on FPGA design projects, the Xilinx Embedded Development Kit (EDK) is often used. EDK consists of Xilinx Platform Studio (XPS) and a suite of IP cores, including PowerPC microprocessor and some operating systems. IP cores, such as microprocessors and Ethernet controllers are pre-designed FPGA configurations used to implement fully designed logic devices in the target FPGA. In addition to purchased IP cores, a design house may have its own reusable FPGA design blocks that are simply implemented in new designs. Xilinx Platform Studio facilitates the integration of new configuration design and pre-designed configuration blocks into a target FPGA. There are also some third party FPGA synthesis tools available that have similar features to those described above with advanced project management tools added in. Examples of third party synthesis tools are Synplify produced by Synopsis and Precision from Mentor Graphics.

Chapter 5

System overview

5.1 Introduction

In this chapter, the description of the development of a two element correlating radio telescope interferometer begins with a system block diagram. This block diagram presents the plan to meet the overall system requirement for an end-to-end two element correlating radio telescope interferometer for transit correlation. An overview of the proposed system structure, the system component interconnections and specifications are discussed.

Modern radio receivers (tuners) use a superheterodyne signal process to select a certain smaller bandwidth of input RF signal from an available larger RF bandwidth. The required smaller bandwidth of RF signal is shifted in frequency to a much lower and fixed centre frequency called the intermediate frequency (IF). Sharp roll-off band-pass filtering is more easily implemented at a fixed and lower frequency [9]. In broadcast FM radio receivers, for example, the available RF bandwidth is from 88 MHz to 108 MHz. A selected small portion of this RF band (a single broadcast radio station) is shifted to an IF of 10,7 MHz for sharp roll-off band-pass filtering, further amplification and demodulation [9]. The radio receiver used in this project (TVRX2) has an available RF bandwidth of 50 MHz to 860 MHz. The IF centre frequency is at 5 MHz and the IF band-pass filtering has a bandwidth of 10 MHz.

The advantage of using the TVRX2 receiver in this project is that band selection and band-pass filtering is accomplished for two input RF signals by two dedicated ICs (TDA18272). This leaves only multiplication and moving average filtering (correlation) to be performed by the limited processing capacity of the FPGA on the USRP2 hardware platform.

5.2 System block diagram

As discussed in chapter 4, the USRP2 FPGA development hardware for software defined radio (SDR) was chosen as the hardware platform. A block diagram of the complete two element correlating radio telescope interferometer is shown in Figure 5.1.

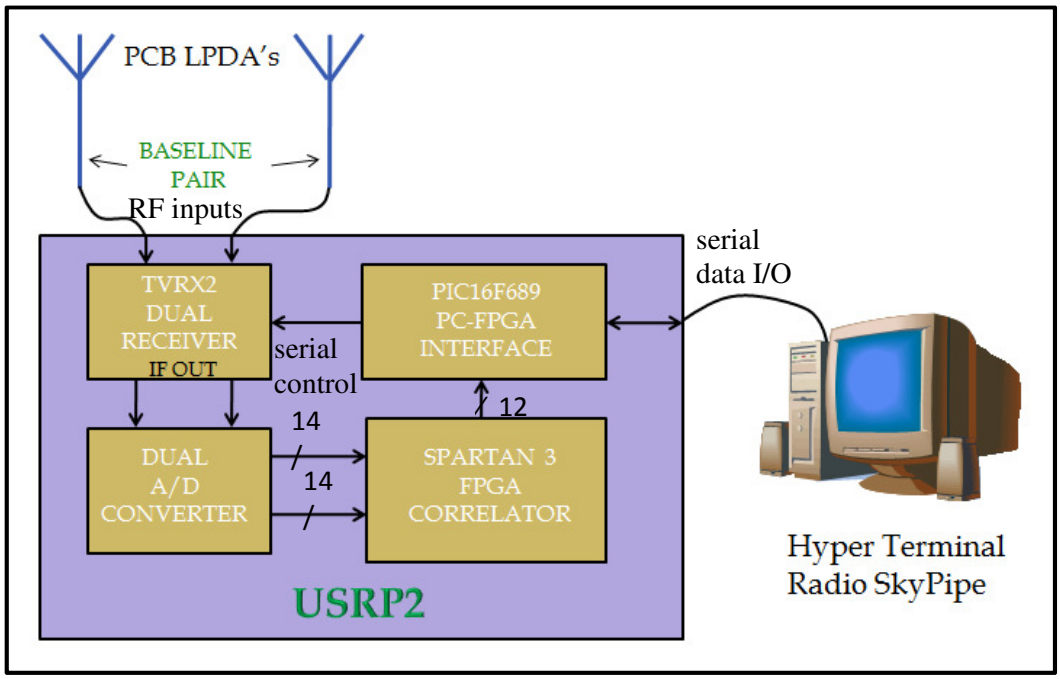


Figure 5.1: The complete two element interferometer system block diagram.

Two broadband log periodic dipole array (LPDA) antennas were used as the baseline antenna pair. LPDA antennas are broadband and moderately directional. The antennas used have a centre frequency of 700 MHz and a bandwidth of 600 MHz which allowed for receiver tuning across the range of frequencies from 400 MHz to 1 GHz without needing to change antennas. The signals from the antennas were cabled to the USRP2 unit which, as seen in the block diagram of Figure 5.1, incorporates four of the system functional blocks. The USRP2 contains the front-end receivers, the analogue to digital converters, the FPGA digital correlator and a PC-FPGA interface for data output and system configuration and control.

5.3 TVRX2 receiver daughterboard

The TVRX2 daughterboard is a dual receiver daughterboard accessory provided by Ettus Research. It has two NXP TDA18272 silicon tuner ICs on the board. These ICs are designed for receiving TV signals from 50 MHz to 860 MHz. The tuning and other adjustments for these receivers are achieved by sending a sequence of serial digital codes on an IIC bus interface. These digital codes are provided in this research project by a PIC microcontroller based interface daughterboard that was designed and constructed as part of this project. The TDA18272 silicon tuners have an IF output signal that is adjustable in frequency and bandwidth. For this system, an IF centre frequency of 5 MHz and bandwidth of 10 MHz was selected, which is the highest available bandwidth. For communication systems, the bandwidth of signal that is processed is kept as small as possible to minimize received noise power which in turn improves sensitivity but for radio telescope systems a wider bandwidth gives improved sensitivity as will be seen in equation 7.10 in section 7.3.6.

5.4 Analogue to digital converter

The analogue IF signals from the TVRX2 daughterboard are digitized by a dual, 14-bit, 100 Msamples/s, analogue to digital converter (ADC). A Linear Technology LTC2284 dual ADC IC is used that is situated on the motherboard of the USRP2. The parallel 14 bit digital outputs from the two ADC channels are routed to the Xilinx Spartan3 FPGA which is also on the USRP2 motherboard. The ADC is clocked at 100 MHz from an Analog Devices AD9510 clock distribution IC. The same 100 MHz clock is the master clock for the Spartan3 FPGA.

5.5 FPGA Correlator

The FPGA gateware configuration was designed from scratch as part of this research project using Xilinx System Generator plug-in for Matlab-Simulink. This provides a block schematic

design environment with a library of logic blocks. System Generator generates HDL code from the block schematic design. An FPGA configuration was synthesised from the HDL code using Xilinx ISE 13.1 [22], [23], [24].

The FPGA is configured as a correlator. The correlation is performed in the time domain as a multiplier followed by a moving average filter. The filter provides the integration as explained in section 3.8. The lag is provided by the geometric time delay between the antenna baseline pair, as explained in section 3.7. Some channel delay offset adjustment has been included in the FPGA correlator design. This is intended to correct for instrumental channel phase offset and not to compensate for geometric delay (τ_g) when observing sources that are not at zenith, as discussed in section 3.8. The problem and possible mitigation of unpredictable phase offset in the TDA18272 silicon tuner is discussed later in sections 9.7, 9.8 and 10.5.

5.6 Interface daughterboard

As part of this research project, the interface daughterboard was designed and constructed and the microcontroller firmware was designed and written in C. The correlator output is read from the FPGA by a Microchip PIC microcontroller on the interface daughterboard. The interface daughterboard interfaces the FPGA correlator and the TVRX2 front-end receiver with two programs running on a PC. Hyper Terminal is a terminal emulator program and this is used to allow text commands to be entered at the PC for configuring the TVRX2 receivers and controlling the transfer of data from the FPGA. Radio SkyPipe is a PC chart recorder and data storage program specifically designed for radio astronomy data capture. Radio SkyPipe is designed to accept 12-bit data samples. Since the dual ADC on the USRP2 motherboard provides 14-bit samples, the two least significant bits will be discarded in the FPGA correlator after correlation.

Chapter 6

Detailed hardware design

6.1 Introduction

The Universal Software Defined Radio Peripheral (USRP2) was used as the starting point for the hardware design. Various front-end tuner daughterboards are available from Ettus Research that allows for designs at different frequency sub-bands within the Radio band. The TVRX2 is a dual channel VHF/UHF receiver daughterboard. This was convenient as two receivers were required for this research project and the USRP2 has only one slot for receiver daughterboard connection. The tuners are silicon tuners that are configured and tuned using IIC serial data commands. This allowed full control of the two element radio telescope interferometer from a PC via the interface daughterboard, which was specifically designed, constructed and programmed for this purpose as part of this research project.

The USRP2 motherboard has, amongst other devices, a Xilinx Spartan3 FPGA, a dual 14-bit 100 Msamples/s ADC and connectors to attach two daughterboards. One daughterboard connector is routed to the ADC on the motherboard and can accommodate a choice of receiver daughterboards. The other daughterboard connector is routed to a DAC on the motherboard to accommodate a choice of transmitter daughterboards. Since this project required only receivers, the transmitter daughterboard connector was spare and the interface daughterboard was therefore designed to plug into this spare slot. The connections to the DAC on the motherboard are not used but there are also 16 general purpose digital I/O (GPIO) connections that connect straight to the FPGA. These 16 GPIO connections were used to interface the PC to the FPGA. Figure 6.1 shows the USRP2 motherboard with the features described above.

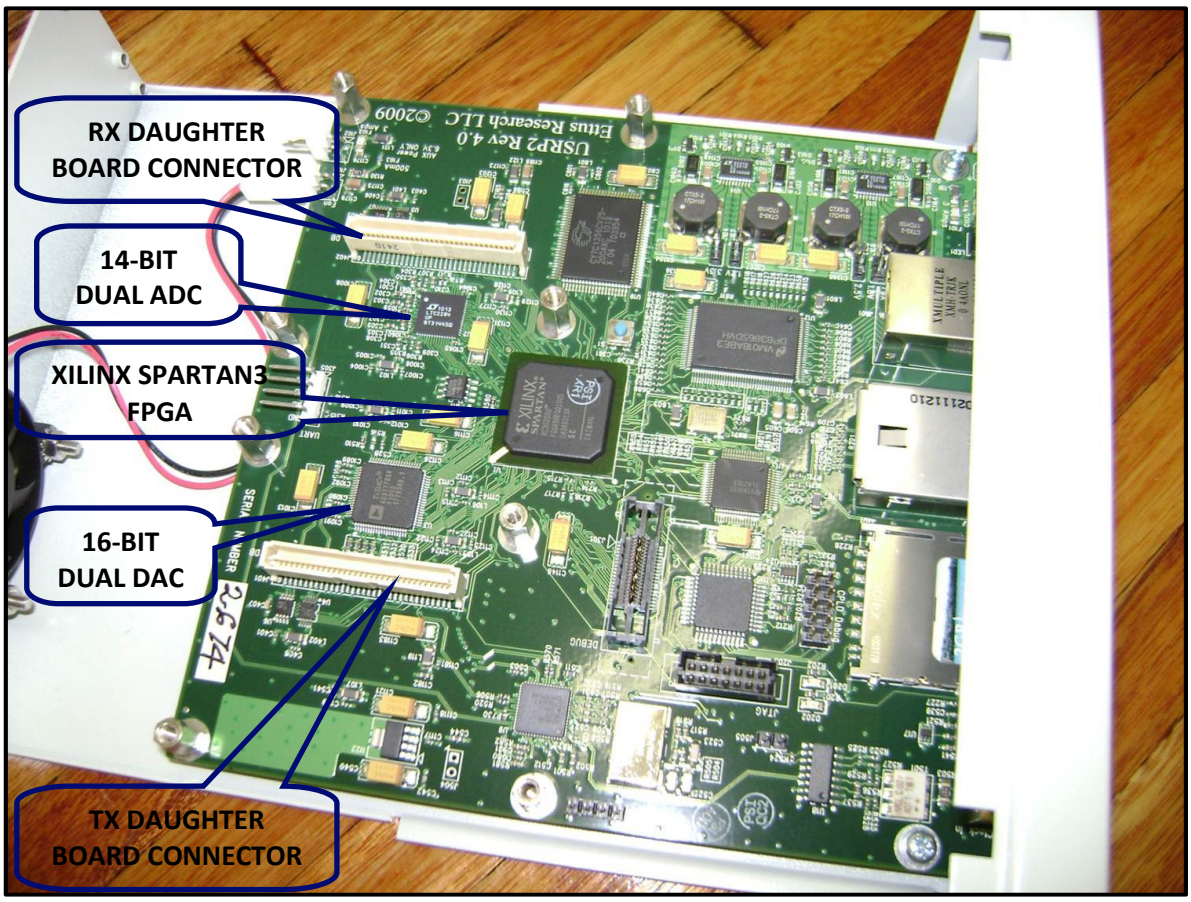


Figure 6.1: The USRP2 hardware platform motherboard showing the location and interconnections between the daughterboard connectors, the ADC, the DAC and the FPGA.

6.2 Interface daughterboard

The interface daughterboard design constitutes the whole of the hardware design for this research project. The functionality of the interface daughterboard is software defined in the PIC microcontroller firmware on the interface board.

6.2.1 Schematic circuit design

The circuit schematic and PCB layout design were drawn using a trial version of Mentor Graphics PADS PCB design software. The USRP2 motherboard provides 16 general purpose digital I/O (GPIO) connections from the daughterboard connector directly to I/O pins of the FPGA. These 16 GPIO connections were allocated to data exchange and flow control between

the FPGA and Radio SkyPipe PC software. Eight were allocated for data and five for flow control. The remaining three were used for the IIC serial data (SDA) connection, the vertical sync control (VSYNC) and the interrupt request pin (IRQ) on the TDA18272 ICs. These three connections can be seen on the far right hand side of Figure 7.3. The five flow control pin connections can be seen in the bottom left of Figure 7.1. These five connections are also used to route serial communications to the TDA18272 silicon tuners on the TVRX2 daughterboard and to two auxiliary serial DACs that are used to provide DC gain adjustment voltages to the IF stage of the TDA18272 tuners. The eight data connections can be seen in the bottom right of Figure 7.1. The 16 GPIO connections to the FPGA on the USRP2 motherboard could be reconfigured to control internal FPGA processes or to connect to external devices on the USRP2 motherboard simply by changing the gateware configuration of the FPGA.

Figure 6.2 is a complete schematic circuit design capture for the interface daughterboard. The voltage supply from the USRP2 motherboard to the interface daughterboard is an unregulated 6 V. This is regulated to 5 V and 3,3 V by two voltage regulators. The 5 V source provides power to a MAX232 IC for serial data communication with Hyper Terminal on the PC. The 3,3 V source provides power to the PIC microcontroller. The PIC microcontroller allows a voltage supply range of 2 – 5,5 V [25], [26]. The 3,3 V supply ensures that the PIC I/O voltage is compatible with the FPGA I/O voltage of 3,3 V [27]. The FPGA I/O voltage was initially thought to be 2,5 V and the PIC microcontroller on the prototype interface board was first provided with a 2,5 V supply. Investigation of consequent erratic behaviour and ‘hanging up’ of the PIC microcontroller eventually led to the discovery that while the FPGA core is supplied with a 2,5 V supply, the FPGA I/O banks are provided with a 3,3 V supply. This separate I/O supply voltage is a feature of Xilinx FPGAs that allows for easy I/O interfacing [27], [28].

The interface with Radio SkyPipe uses the parallel printer port of the PC. Some voltage level shifting resistors were included to reduce the voltage on the parallel printer port pins from 5 V to 3,3 V to be compatible with the 3,3 V I/O voltage of the PIC. It was found, however, that there is significant voltage drop on the approximately two metre long cable from the PC parallel port and the resistor dividers were causing a logic high voltage to be too low, leading to occasional logic level errors. The divider resistors R8 and R9, shown in Figure 6.2, were removed and the logic high voltage level was then found to be 3,6 V at the USRP2 end of the parallel printer cable, with no divider. The excess voltage (>3,3 V) is not a problem because there are still the series resistors R3 and R4 (shown in Figure 6.2) and all PIC microcontroller I/O pins have clipping diodes [25], [26]. This will result in the excess voltage being clipped to 4 V (3,3 + 0,7 V) at the PIC I/O pin and any additional excess voltage will be dropped across the series resistors R3 and R4. A problem could arise if the cable connection from the PC parallel printer port is too long resulting in the I/O logic high voltage at the PIC microcontroller end being too low. This would cause data errors in the plots on Radio SkyPipe. A maximum cable length of two metres is therefore recommended.

Transistor Q1, seen in Figure 6.2, converts the 3,3 V logic voltage of the SkyPipe serial data output from the PIC to 5 V, for compatibility with the parallel printer port of the PC. The transistor inverts the serial bits that are sent to Radio SkyPipe on this parallel printer port pin. The PIC microcontroller software inverts the serial data logic that is output to Radio SkyPipe to compensate for this.

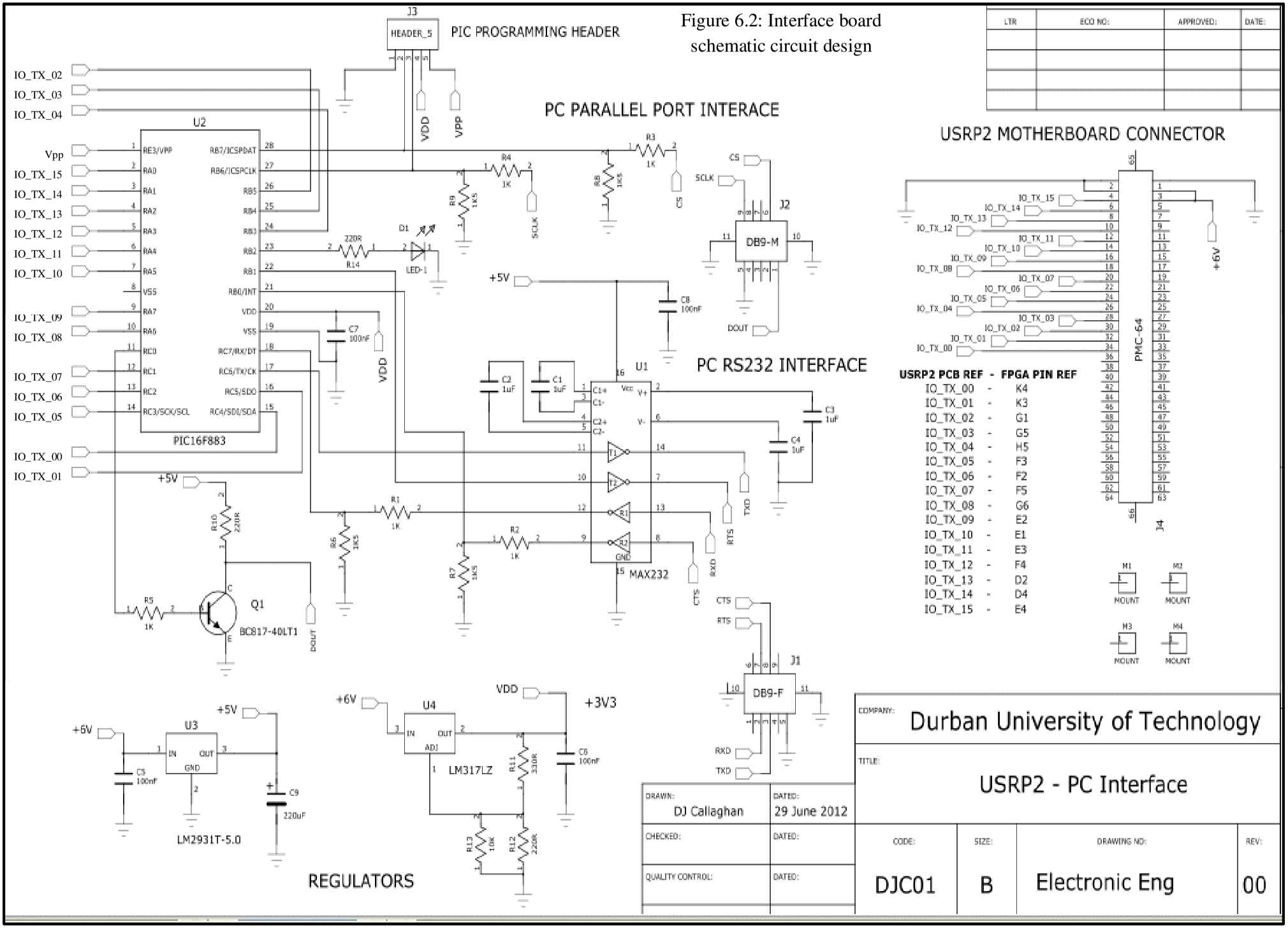


Figure 6.2: Interface board schematic circuit design

6.2.2 Prototype construction and testing

A prototype of the interface board circuit was constructed on project board and tested. This prototype was used both for verifying that the circuit design would perform as required, as well as for continuing with the development of the FPGA correlator and PIC microcontroller interface firmware while the interface board PCB was being manufactured. A PIC16F689 microcontroller was used in the prototype since a DIP package of this microcontroller, needed for construction on project board, was on hand. For the final PCB, a PIC16F882 surface mount device, also on hand, was used. The PIC16F882 has more I/O than the PIC16F689 which allowed the data interface to be increased from six bits to eight bits. This simplified the microcontroller firmware design because all the registers in the PIC 16 series microcontrollers are eight bit registers [25], [26]. The FPGA configuration could be easily and quickly adapted to accommodate this change in interface design between the prototype and final interface board. Ease of reconfiguration of FPGAs is one of the key features of these devices.

Figure 6.3 shows the prototype interface board temporarily mounted onto the TVRX2 daughterboard to facilitate connection to the FPGA. A spare TVRX2 daughterboard was modified for this purpose. The spare daughterboard connector on the motherboard, where the final interface board would be mounted, is seen to the left of the prototype interface board. The other prototype board on the right was used to apply firstly variable DC voltage levels onto the two ADC channels and then audio signals from an audio signal generator. The variable DC levels, adjusted using the preset potentiometers seen on the prototype board, were used to verify that the ADC on the USRP2 motherboard was correctly configured and that the FPGA was reading the ADC samples correctly. The SkyPipe chart recorder trace line was seen to move up and down as the preset potentiometers were adjusted, confirming that the ADC digital data input to the FPGA was functioning correctly.

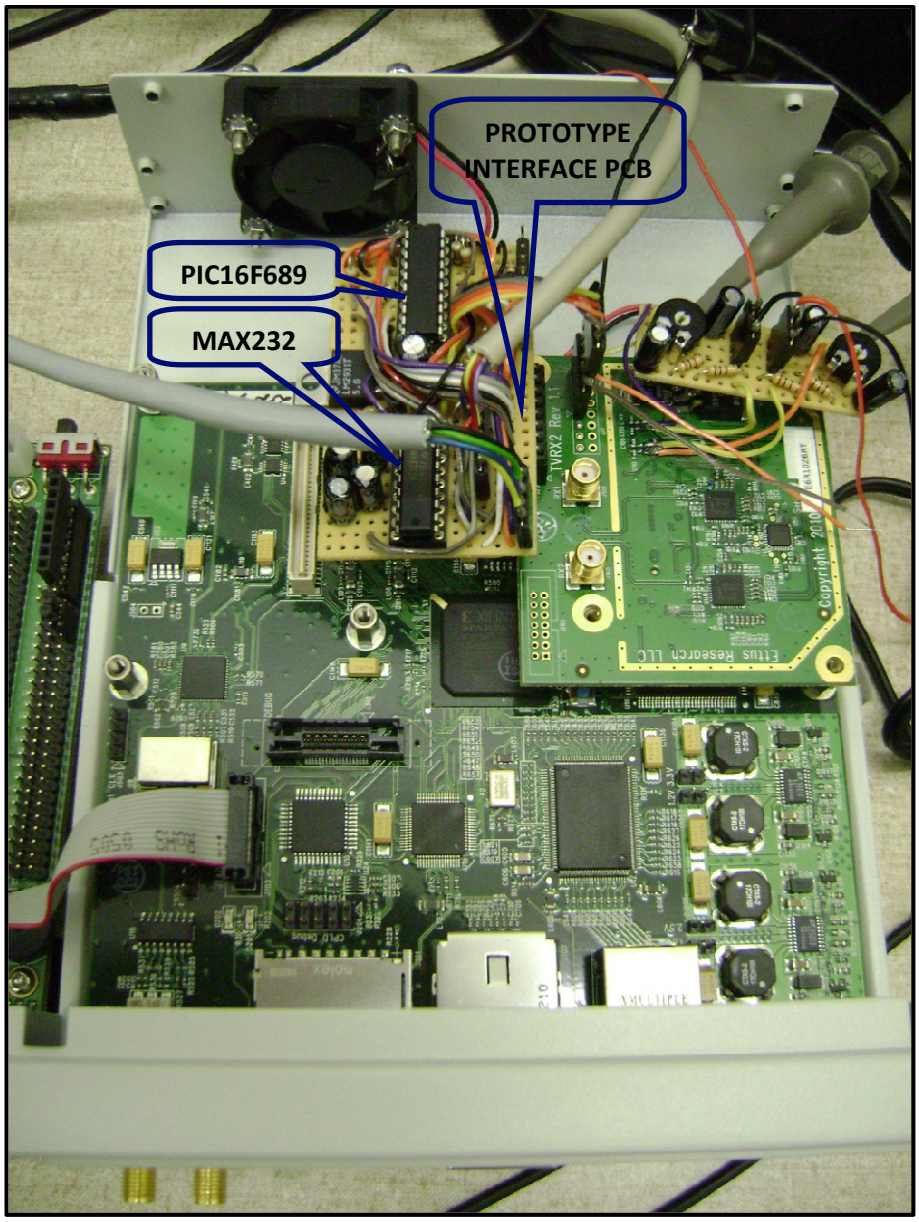


Figure 6.3: Prototype interface board construction showing the use of a modified TVRX2 daughterboard in order to access connections on the RX daughterboard connector on the USRP2 motherboard.

6.2.3 PCB layout design

The interface board consists of a PIC 16F882 microcontroller, a MAX232 serial interface IC, two voltage regulators, a 64-pin connector to fit the daughterboard connector on the USRP2 motherboard and two DB9 connectors for two interface cables to connect to a PC. An LED was included as a board status indicator. All components are surface mount except the 5 V

and 3,3 V regulators, the two DB9 connectors and the programming pin header. A populated and unpopulated interface board are shown in Figure 6.4.

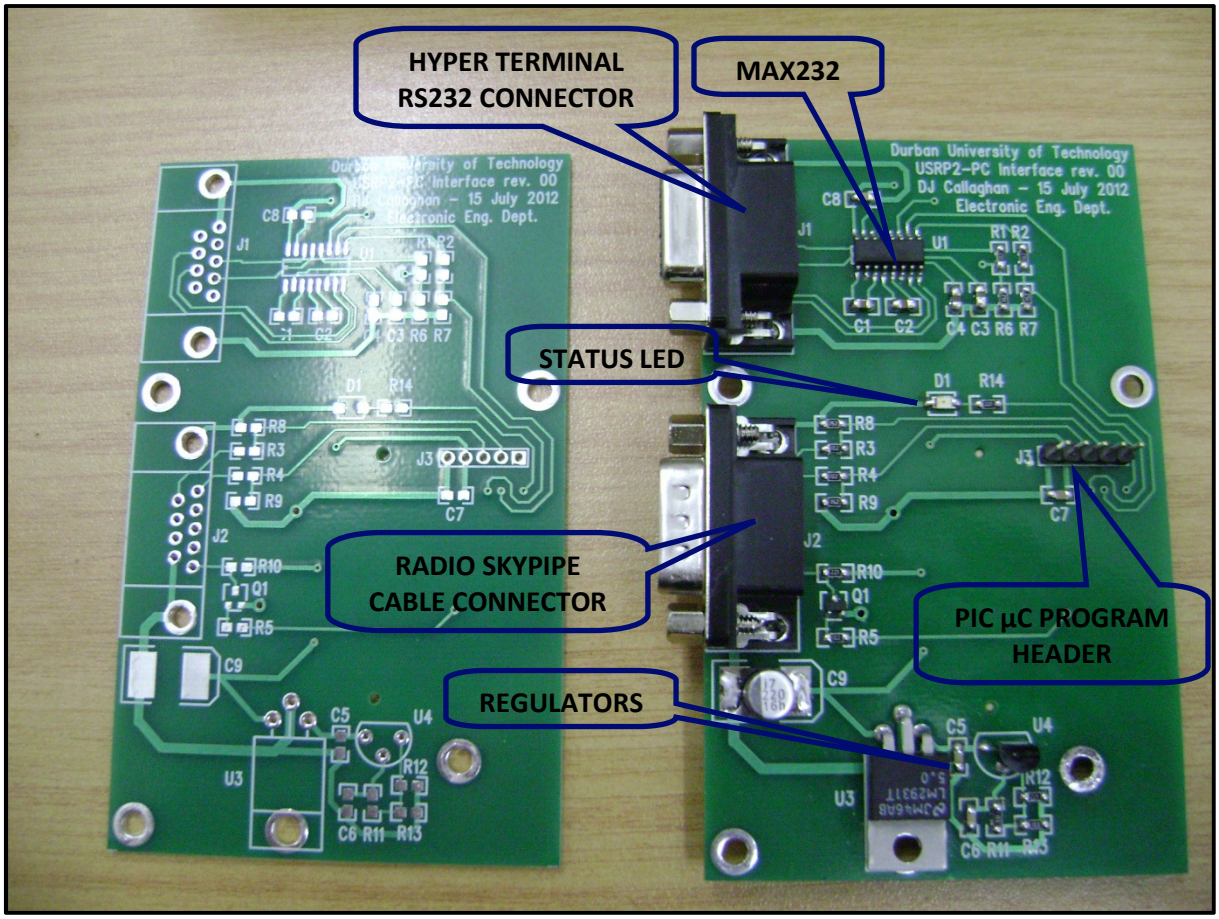


Figure 6.4: Unpopulated and populated interface PCB showing the construction steps, the simplicity and structure of this sub-system product, the component identification and the board identification.

The USRP2 motherboard has four mounting posts for each of its two daughterboards. Pads were placed on the interface board to mark the position of drill holes to align with the four mounting posts. The position of the daughterboard connector, relative to the mounting holes was then critical and had to align with the daughterboard connector socket on the USRP2 motherboard. In the absence of a detailed engineering drawing showing the position of the daughterboard connector relative to the mounting posts, this position was reverse engineered by printing the outline of the designed interface PCB mounting pads and daughterboard connector onto transparency and placing this onto the USRP2 motherboard to check for correct alignment of the mounting pads and daughterboard connector. Several small

adjustments to the position of the daughterboard connector relative to the mounting pads were required to achieve proper alignment.

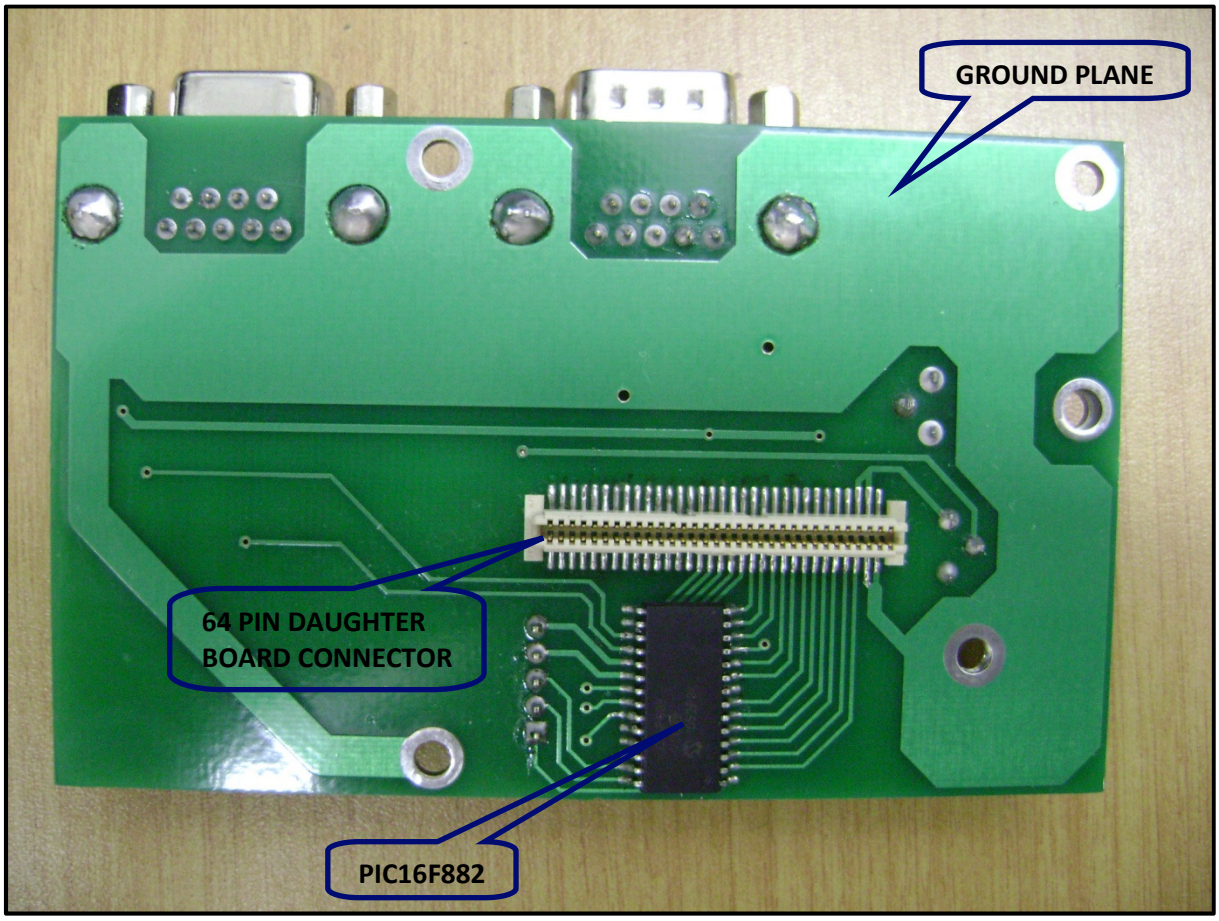


Figure 6.5: Interface PCB (bottom view) showing the location and extent of the ground plane (a short distance from all components requiring a low impedance ground connection) and the placement of the PIC microcontroller for ease of track routing to the interface board connector.

The placement of the DB9 connectors was chosen to facilitate a convenient side connection for the two PC connection cables, as seen in Figure 6.6. The balance of the component and track placement was flexible and chosen to allow a sufficiently large ground plane and to minimise track routing. The PIC microcontroller was placed on the bottom side of the PCB. This minimised the number of vias required since the 64-pin daughterboard connector, to which most of the PIC microcontroller pins are connected, had to be surface mounted on the bottom side to facilitate connection to the USRP2 motherboard. There are 16 track

connections directly from the microcontroller to the daughterboard connector. The rest of the components were placed on the top side of the interface board as seen in Figure 6.4.

6.2.4 PCB population and final assembly

All components were hand soldered onto the PCB and the board was tested, firstly on its own with a separate power supply and test cables and finally connected into the USRP2 daughterboard socket as shown in Figure 6.6. Firmware was written for the PIC microcontroller to provide test data that could be displayed on the Radio SkyPipe chart recorder without requiring the interface board to be connected to the USRP2 motherboard. Details of the PIC microcontroller firmware design are given in chapter 7.

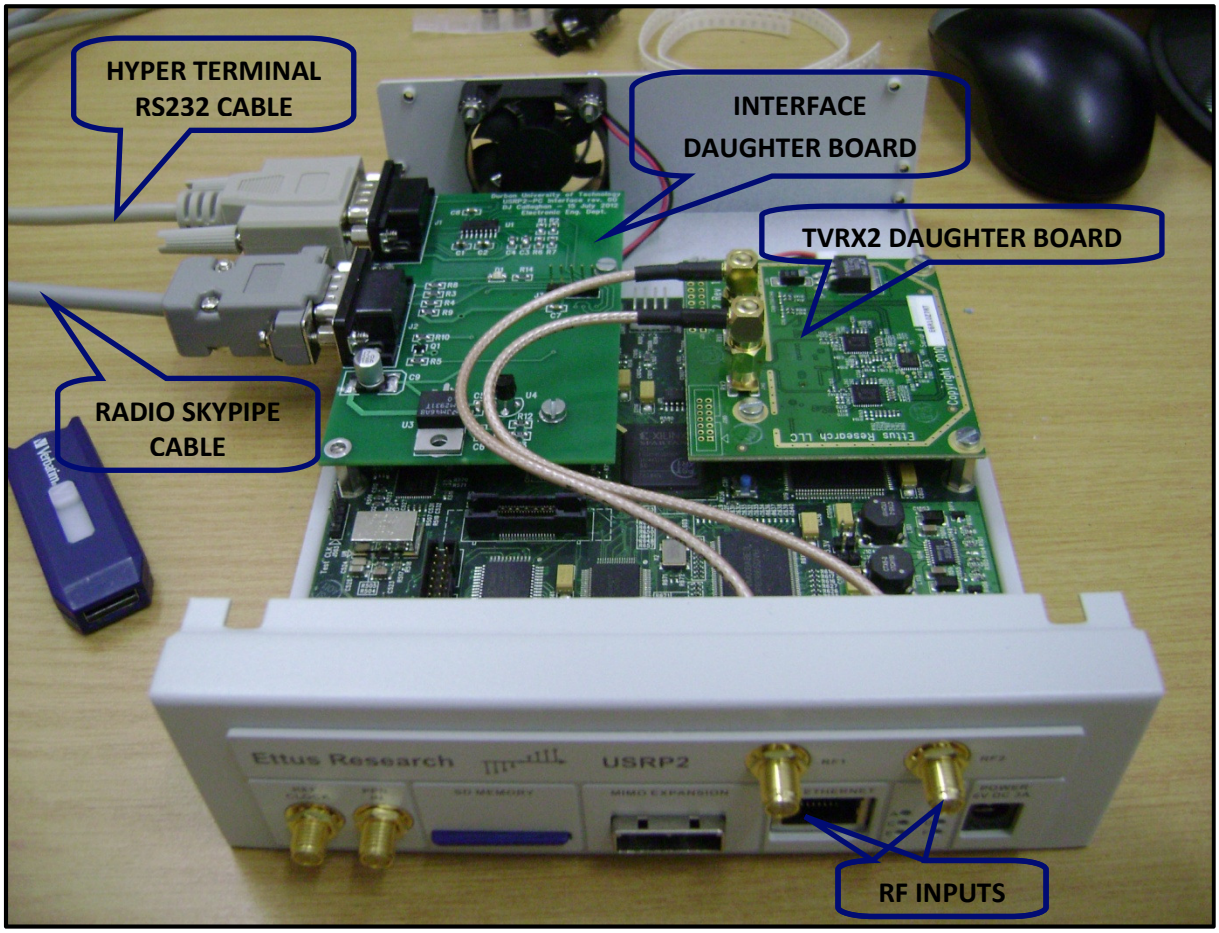


Figure 6.6: The final two element interferometer hardware platform assembly showing the inserted interface daughterboard, its physical compatibility with the complete USRP2 system and ease of access for testing and programming.

Chapter 7

Detailed system software design

7.1 Introduction

The digital correlator for the radio telescope interferometer was implemented in the Xilinx Spartan3 FPGA on the motherboard of the USRP2. The input data comes from the dual 14-bit 100 Msamples/s ADC that samples the analogue IF outputs of the TVRX2 dual receiver daughterboard. The two digital signal input channels to the FPGA correlator are called channels X and Y in this project. Samples from channels X and Y are multiplied using a hardware multiplier and consecutive products are averaged by a 2048 point integrator comb filter implemented in the FPGA. The filter output is the digital cross-correlation of the two TVRX2 IF signals for one value of lag, as discussed in section 3.8. The lag value varies as the Earth rotates, because of changing geometric path length difference between the source and the two spatially separated receiving antennas, as discussed in section 3.7. The FPGA correlator output is made available in eight bit parallel format on FPGA output pins which are read by the PIC microcontroller on the interface daughterboard and serially communicated to Radio SkyPipe chart recorder software on a PC. Radio SkyPipe chart recorder is designed to plot relatively slowly varying values read from a 12-bit serial ADC IC. The PIC microcontroller was therefore programmed to emulate this serial ADC IC.

The software design for this research project therefore consisted of two parts. Xilinx Spartan3 FPGA gateware design for the digital correlator and Microchip PIC16F882 microcontroller embedded firmware design for the PC interface. The FPGA gateware was designed using Xilinx System Generator block schematic design tool. System Generator is a plug-in to Matlab-Simulink and provides a library of FPGA logic function blocks for schematic design.

One of the blocks is a System Generator block that must be included in the Simulink block schematic design. The design may be simulated using Simulink and when ready to implement, the design is compiled to HDL code by double clicking on the System Generator block and clicking on generate [22], [24].

The PIC microcontroller embedded firmware was written in C programming language. This is standard for embedded firmware and facilitates easy migration to other microcontroller devices and makes program updates and modification easier. The C program was compiled using PICC Lite which is a free C Compiler for 16 series PIC microcontrollers.

7.2 Transit correlator

The FPGA digital correlator process produces one sample of the cross-correlation function for a specific lag value τ, the time lag between the two signals received at the two antennas. As the Earth rotates, the time lag, or phase, between the EM wave received at one antenna relative to the other changes due to path difference geometry. A longer interval of integration gives a result closer to the theoretical correlation function, which requires an infinite integration interval. Since the lag term τ is varying with time due to Earth rotation, the response of the correlator to changing lag is reduced by an integration interval that is too long. If there is no rotation of the source relative to the receiver pair, the correlator output will be constant. As the source passes overhead due to Earth rotation, the correlator output will produce the full range of the correlation function $r(\tau)$ which is the fringe oscillation function of baseline described by equation 3.8 in section 3.8.

For radio telescope arrays that are used to observe sources at some elevation other than zenith, the geometric time delay is usually cancelled out with delay electronics in the receivers as discussed in section 3.8. The reason for this is that for wider bandwidth signals, the coherence time is shorter and fringes may not be obtained if the geometric delay is too long [7]. A wider bandwidth of observation is often required to improve the equipment sensitivity, which is discussed further in section 7.3.6.

7.3 FPGA Gateware

The heart of the FPGA gateware is the digital correlator. The correlation is computed directly according to equation 7.1, the discrete equivalent of equation 3.4, by multiplying samples from the two TVRX2 tuner IF outputs together and computing a moving average of these products. The average is the digital equivalent of integration over a finite interval and the moving average tracks the changes in the correlation function value as the lag term increases with Earth rotation.

$$r(\tau) = \frac{1}{n}\sum_{i=1}^{n} f(i)g(i-\tau) \qquad 7.1$$

7.3.1 Setting the FPGA sampling rate

The sample rate for digitizing the IF output from the silicon tuners on the TVRX2 board must be at least twice the bandwidth of the IF signal, according to the Nyquist sampling theorem [9]. The IF bandwidth was set to 10 MHz, which is the highest bandwidth setting for the TDA18272 silicon tuners. A sample rate of 25 Msamples/s was therefore chosen. This is set in the input blocks of the System Generator block schematic design for the FPGA correlator. The FPGA master clock of 100 MHz is specified in the System Generator block and this leads

to the automatic inclusion of a divide by four frequency divider when the design is generated by System Generator. Since the ADC is then running at four times the FPGA sample rate, the FPGA is reading every fourth sample from the ADC. One advantage of not having the FPGA sample rate too much higher than the Nyquist rate is that a higher integration time is achieved with the number of sample points in the averaging process. This can be seen in equation 7.9. Another is less power consumption and heating in the FPGA due to a slower clocking speed.

7.3.2 Simulink hierarchical design

The FPGA gateware design uses a feature of Simulink called sub-system design. This creates a hierarchical layered design [22]. Figure 7.1 shows the top layer of the design. The moving average filter design is shown as a block in this layer. Double clicking on this block opens up the detail of this sub-system design, which is shown in Figure 7.2. Similarly, the microcontroller interface logic design is defined in the top layer as a sub-system design. Double clicking on this sub-system will open up the sub-system design details shown in Figure 7.3.

7.3.3 Channel phase offset adjustment

The channel phase offset adjustment logic shown in Figure 7.1 provides up to 16 samples of phase shift in either the X or Y channel input. This feature was included to compensate for unpredictable channel phase offset that occurs in the TDA18272 silicon tuners on the TVRX2 daughterboard. The phase offset can be adjusted using microcontroller firmware by first selecting the channel for which the phase offset must be adjusted. This is done by pressing X or Y in Hyper Terminal. Then the three select logic inputs discussed in section 7.3.7 must be set to **"**output phase offset**"** according to the truth table in Figure 7.3. Finally the phase offset

counter in the FPGA is enabled by setting the enable bit also defined in the truth table in Figure 7.3. The channel phase offset will increment once every millisecond, clocked by the X or Y phase counters shown in Figure 7.1. The phase offset in samples from 0 to 15 can be read from the data input lines on PORTA of the PIC microcontroller. When the phase offset reaches the desired number of samples, the enable bit must be cleared. The microcontroller firmware for this was not implemented due to time constraints but the menu option was put in place in line 318 of the C program listing of Annexure A and an empty C program function called “setphase” in line 1005 [29]. Recommendations for channel phase offset adjustment are discussed in sections 9.8 and 10.5.

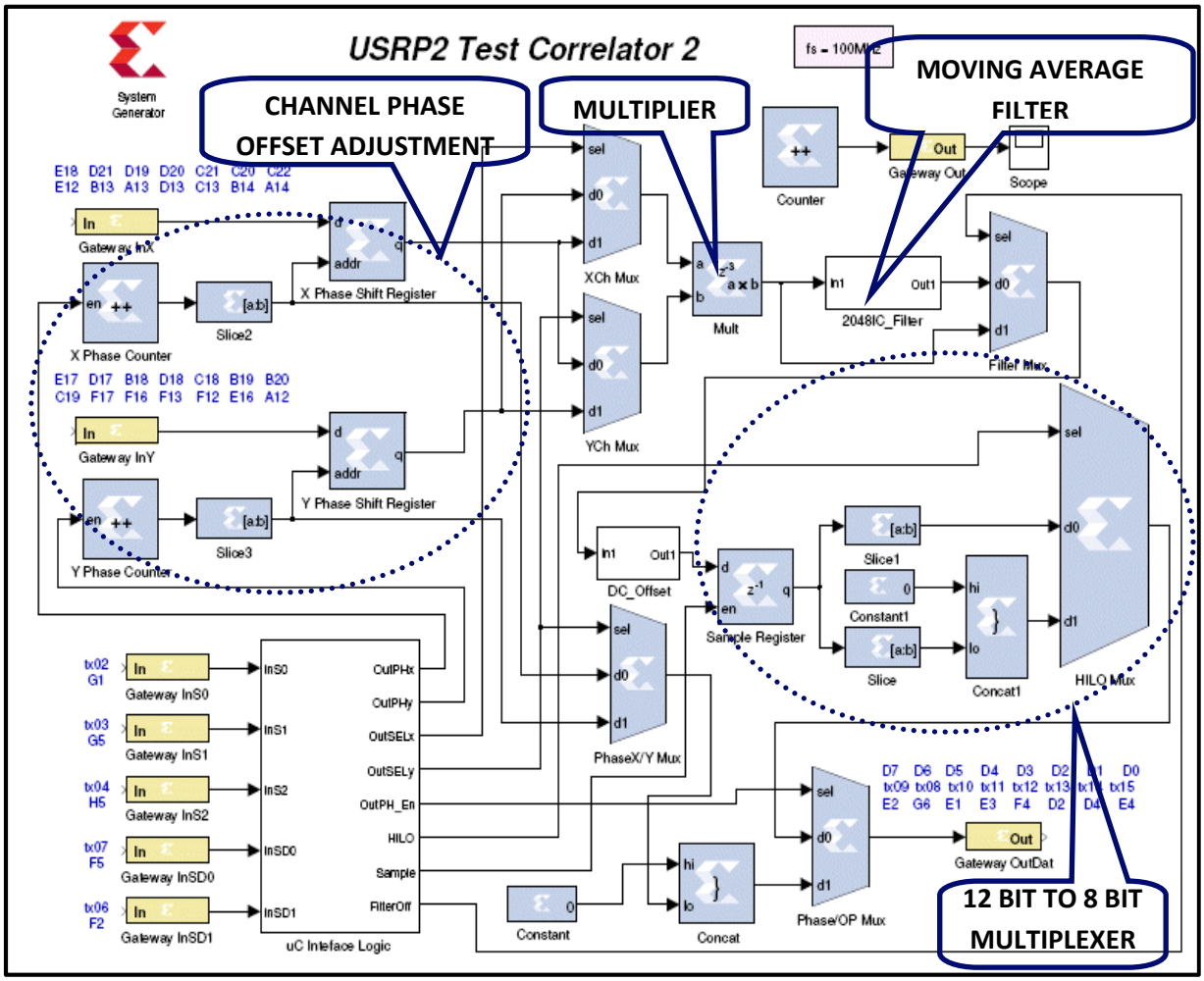


Figure 7.1: USRP2 FPGA correlator block schematic diagram design using Simulink and Xilinx System Generator. The multiplier and moving average filter blocks are identified.

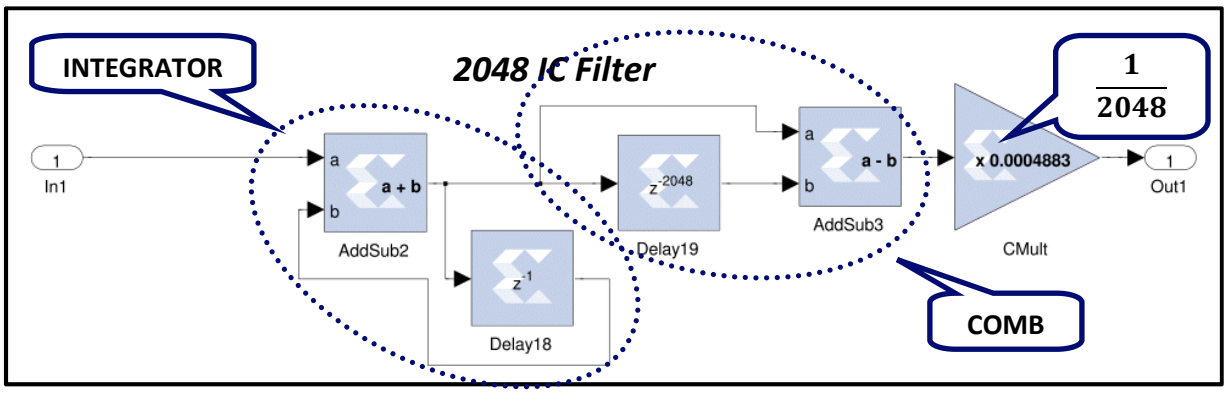


Figure 7.2: FPGA moving average filter block schematic diagram. A sub-system of the correlator block schematic design of Figure 7.1. The required moving average filter is implemented as an integrator-comb filter as shown.

7.3.4 FPGA multipliers

Multipliers can be implemented in several different ways in an FPGA. Most FPGA's provide a limited number of hardware multipliers. These are intended for use when the latency (number of clock cycles required to complete a process) needs to be very low [22]. For some multiplications in DSP, one of the numbers in the multiplication is a constant. These are called constant coefficient multiplications and are sometimes implemented in FPGA fabric rather than using a hardware multiplier. When designing the logic in System Generator, a multiplier and a constant coefficient multiplier are represented by different schematic blocks, as can be seen in Figures 7.1 and 7.2. Right clicking on any block placed in a design allows the designer to specify how the block should be implemented in the target FPGA [22]. If the designer does not specify this then the synthesis tool will attempt to make the best use of the available FPGA resources when implementing the required logic.

The multiplier in the FPGA correlator for this research project was specified in System Generator to be a hardware multiplier. There are 40 hardware multipliers available in the Spartan3 FPGA. Each hardware multiplier can multiply two 18-bit numbers and produce a

36-bit result [27]. In this research project, the ADC produces 14-bit samples and so the width of the hardware multiplier was adequate.

7.3.5 Integrator-comb filter

The FPGA correlator shown in Figure 7.1 consists of a multiplier block configured as a hardware multiplier and a 2048 sample moving average filter implemented as an integrator-comb (IC) filter, shown in Figure 7.2. The IC filter implementation is very gateware efficient, requiring only two adders compared with 2048 adders required for a direct moving average implementation [22]. The constant coefficient multiplier of the IC filter divides by 2048 and since this is the eleventh power of two, this is implemented by simply shifting the bits eleven places to the right. This is implemented by the synthesis tool in the routing of bits from the second adder to the filter output and requires no gateware. A mathematical proof that an integrator-comb filter is equivalent to a moving average filter is as follows [22]:

The z domain transfer function for an eight sample moving average filter is given by:

$$\mathrm{H(z)} = \frac{1}{8}(1 + \mathrm{z}^{-1} + \mathrm{z}^{-2} + \mathrm{z}^{-3} + \mathrm{z}^{-4} + \mathrm{z}^{-5} + \mathrm{z}^{-6} + \mathrm{z}^{-7}) \qquad 7.3$$

The integrator output in Figure 7.2 is given by:

$$Y(z) = X(z) + z^{-1}Y(z) \qquad 7.4$$

$$Y(z)(1 - z^{-1}) = X(\,z) \qquad 7.5$$

And so the integrator transfer function is:

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{1}{1 - \mathrm{z}^{-1}} \qquad 7.6$$

An eight sample integrator-comb filter subtracts from the current integrator output, the integrator output eight samples back and then divides by the number of samples in the average and so the full integrator-comb transfer function is:

$$H(z) = \frac{1}{(1 - z^{-1})}(1 - z^{-8})\frac{1}{8} = \frac{1}{8}\frac{(1 - z^{-8})}{(1 - z^{-1})} \qquad 7.7$$

Furthermore:

$$(1 - z^{-1})(1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + z^{-5} + z^{-6} + z^{-7}) = (1 - z^{-8}) \qquad 7.8$$

Substituting the left side of equation 7.8 into equation 7.7 we see that the transfer function of the integrator comb filter is identical to that of the moving average filter given in equation 7.3. The same proof holds for any number of sample points.

7.3.6 Adjusting the integration time

There are limited resources on the Spartan3 FPGA and the longest moving average that could be obtained was 2048 samples. This was limited by the available resources on the target FPGA to implement the 2048 sample delay labelled Delay19 in Figure 7.2. Since the sample rate (τ_S) is 25 Msamples/s, this gives an integration time (τ_I) of:

$$\tau_I = 2048 \times \tau_S = 2048 \times \frac{1}{25 \times 10^6} = 81{,}92\ \mu s \qquad 7.9$$

As a starting point, the integration time should be much greater than the inverse of the bandwidth to smooth out the modulating effect that the bandwidth has on the fringe amplitude [12]. For this research project the bandwidth was 10 MHz and so the integration time should be much greater than 0,1 µs. To avoid a reduction in fringe sensitivity, the integration time should be much less that the fringe period [12], which was about 20 seconds for the two metre baseline test of chapter 9, shown in Figure 9.6. Care must be taken not to make the integration

time too long, bearing in mind that the fringe period decreases with increasing baseline distance, as can be seen from equation 3.8 and also seen in the optical fringe diagram of Figure 3.6.

A longer integration time can be used to overcome the effect of receiver system noise temperature (T_{sys}) on the sensitivity of the two receivers to source noise temperature (ΔT). The system sensitivity, after correlation, is inversely proportional to the square root of the bandwidth times the integration time [1], [17]:

$$\Delta T \propto \frac{T_{sys}}{\sqrt{\Delta\nu\tau_I}} \qquad 7.10$$

The Spartan3 FPGA does not have sufficient resources to increase the integration time to the next allowable quantity for the IC filter of 4096 samples. Much higher integration times could however be achieved by further averaging of the FPGA correlator data output in the PIC microcontroller on the interface board. The microcontroller need only read a sample from the FPGA correlator output every 81,92 µs and compute a moving average of these samples. This sample rate is well within the reach of the processing speed of the microcontroller. So in other words, the microcontroller computes an average of the 81,92 µs interval averages. This is equivalent to multiplying the interval of the FPGA IC filter average because:

$$\frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{m}\sum_{i=1}^{m} a_{(j-1)m+i}\right) = \frac{1}{n\times m}\sum_{i=1}^{n\times m} a_i \qquad 7.11$$

The advantage of extending the integration time in this way is that expensive FPGA fast digital logic resources are not wasted performing processes that could be achieved with lower speed logic. The FPGA acts as the front-end fast logic processor and once the required processing speed is within the reach of microcontroller firmware, the integration time can

then be increased as much as required in the microcontroller firmware. The FPGA integration time will set the increment of adjustment for the total integration time. This potential was identified but not implemented in this project due to time constraints and so the integration time is fixed at 81,92 µs for this research project.

7.3.7 Microcontroller interface logic

Although there are a lot of blocks in the microcontroller interface logic design shown in Figure 7.3, it consists of repeated simple enable logic. The microcontroller provides data routing information to the FPGA via three select logic inputs (S2 S1 S0). Two serial data connections (SD1 SD0) are then routed to various parts of the FPGA allowing the microcontroller to set up various FPGA configuration options or to route serial communications to other devices on the USRP2 motherboard. A truth table showing the logic details for each selection option is shown on the diagram of Figure 7.3 and in table 7.1.

Table 7.1: Microcontroller-FPGA interface control function truth table.

S2	**S1**	**S0**	**SD1**	**SD0**	**FUNCTION**
0	SELECT		DAT	CLK	ROUTE SERIAL SETUP TO DEVICES
1	0	0	X/Y	EN	SELECT X times X or Y times Y
1	0	1		EN	OUTPUT PHASE OFFSET
1	1	0	sample	HILO	OUTPUT CORRELATOR DATA
1	1	1		EN	SELECT CORRELATE – X times Y

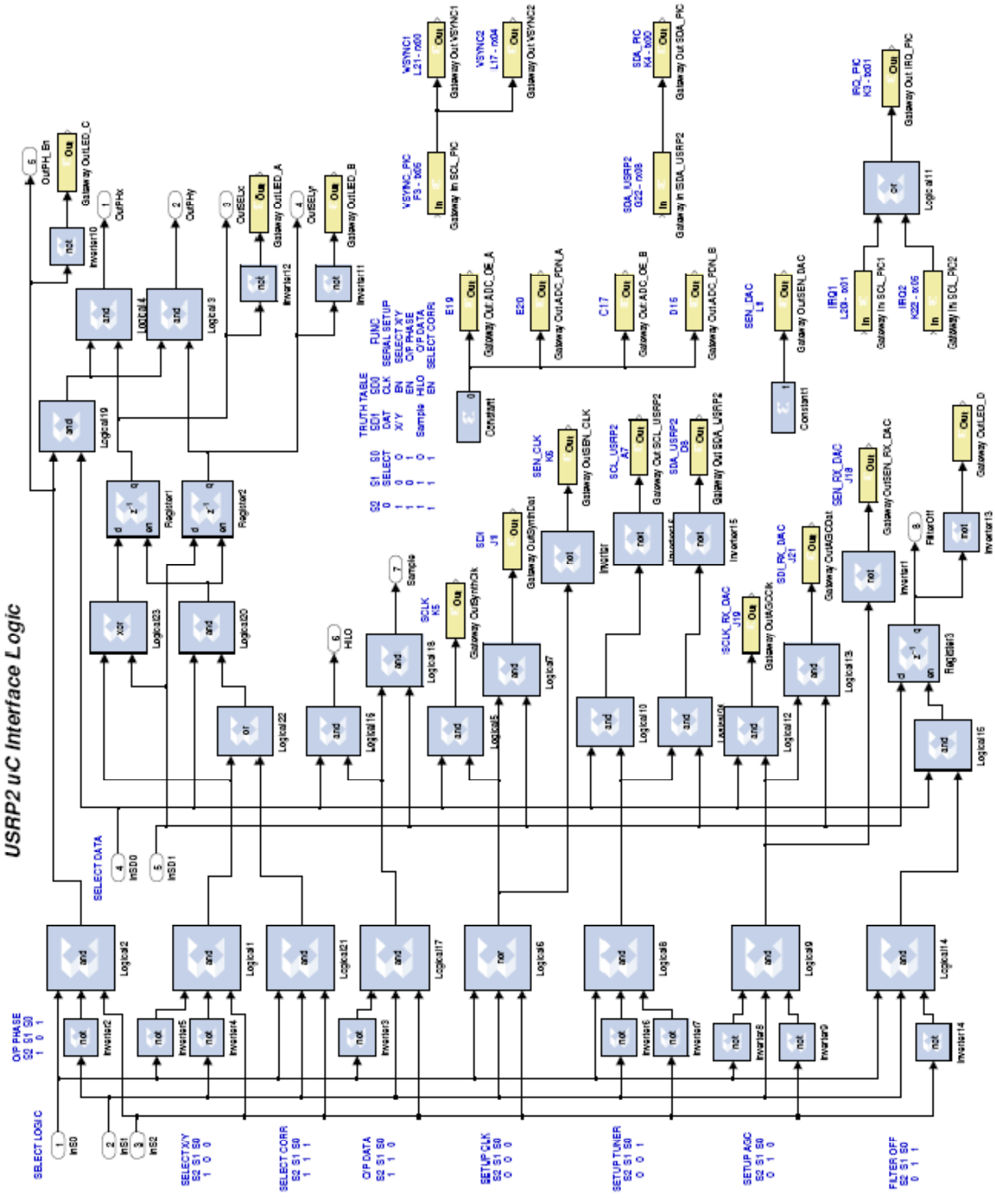


Figure 7.3: FPGA microcontroller interface logic schematic diagram. A sub-system of the correlator block schematic diagram of Figure 7.1.

7.4 Microcontroller Firmware

Simplified flowcharts of the interface board PIC microcontroller firmware are shown in Figures 7.4 and 7.5. A complete C program listing is included in Annexure A [29]. The main program polls the RS232 serial data from Hyper Terminal, waiting for some setup command while the interrupt service routine (ISR) reads the correlator output data from the FPGA and communicates this serially to Radio SkyPipe chart recorder software on the PC.

7.4.1 Interfacing with Radio SkyPipe

The PIC microcontroller on the interface board emulates a MAX187 ADC. A sample of the FPGA correlator output is held in the microcontroller memory until the enable signal is received from Radio SkyPipe. This signal triggers the interrupt service routine (ISR) shown in Figure 7.5. The ISR shifts the 12-bit correlator data out to the serial data pin allocated on the PC parallel printer port. The data transfer is synchronized with a clock signal provided by Radio SkyPipe on another parallel printer port pin. Three parallel printer port pins are used that are called clock (SCLK), data (DOUT) and enable (CS). The C program details can be seen in lines 193 to 248 of the C program listing in Annexure A. Radio SkyPipe expects to be communicating with a hardware ADC and so the PIC ISR needed to have very low latency and very fast firmware to keep the serial correlator data output in proper synchronization with the clock signal provided by Radio SkyPipe [29].

The correlator output data is truncated to 12 bits from the original 14-bit data from the dual ADC on the USRP2 motherboard. The reason for this is that the PC chart recorder software, Radio SkyPipe, requires 12-bit input data. Radio SkyPipe software is designed to interface with a MAX187 ADC IC that provides 12-bit sample data output serially. The MAX187 IC

requires an active low enable that initiates a conversion. After a conversion delay (t_{conv}) of 8,5 µs, a clock signal must be provided to synchronize the output of the twelve data bits [30]. Radio SkyPipe uses specific pins on the PC parallel printer port to provide the enable signal, the clock and to read the serial data from a MAX187 ADC, as described above.

Emulating a MAX187 IC is achieved by an ISR with very short latency (~8,5 µs) and fast parallel to serial data conversion synchronized to the clock signal received from Radio SkyPipe. In order to achieve very low interrupt latency with C programming language [29], the data read from the FPGA correlator is pipelined. In other words, each time the ISR is triggered by the Chip Select (CS) enable signal from Radio SkyPipe, the data transferred to Radio SkyPipe is the data that was read from the FPGA on the previous interrupt. The data is thus ready for transfer in microcontroller memory. After transferring twelve bits of data to Radio SkyPipe, the ISR reads the next correlator sample from the FPGA and stores this in microcontroller memory ready for the next Radio SkyPipe interrupt. During testing, it was found that Radio SkyPipe would periodically miss the first bit of data from the microcontroller due to interrupt latency. Since this is the MSB, this might result in a large error on the SkyPipe chart recorder which showed as a large chart spike up or down. This was solved by having the first bit ready on DOUT before the interrupt occurs. Therefore, as can be seen in the flowchart of Figure 7.5, no data is shifted on the first pulse of SCLK. Each data bit is read by SkyPipe on the rising edge of SCLK and should the interrupt latency result in the first rising edge being missed by the ISR, the bit is already in place and the PIC firmware proceeds to the “SCLK HI?” decision that waits for the end of the first SCLK pulse.

Not shown in the flowchart of Figure 7.5, the ISR can be configured to provide test data to test the interface with Radio SkyPipe without needing to be connected to the motherboard of

the USRP2. This facility is activated by pressing one of the test data command characters U/D/Z/M described in table 7.2. The test data facility is useful for tracking down system malfunctions. This will be discussed further in section 8.2.

7.4.2 Interfacing with Hyper Terminal

The PIC microcontroller also communicates, using a standard RS232 serial cable [25], [26], with Hyper Terminal, which is a serial terminal emulator program that runs on a PC. ASCII text commands can be entered in Hyper Terminal to configure the FPGA and to start and stop the acquisition of data from the FPGA. A flow chart of the command menu can be seen in Figure 7.4 and the C program details in lines 251 to 337 of Annexure A [29]. A simple command menu is communicated from the PIC microcontroller to Hyper Terminal. Most commands require the user to press a single upper case alphabetical character key on the keyboard. Some commands require this to be followed by a numerical input. The first command, “I”, performs the initialization of the TVRX2 daughterboard tuners. This could not be automatically executed by the microcontroller on power up since the FPGA has to be configured first. The FPGA is manually configured from the PC using a Xilinx configuration tool called Impact. Impact uses a JTAG cable connection to the USRP2 motherboard to install the configuration on the FPGA. Once this has been done, The FPGA can route serial communication using IIC from the PIC microcontroller on the interface card to the TDA18272 silicon tuners on the TVRX2 daughterboard. This is initiated by entering the command “I” in Hyper Terminal. Table 7.2 is a list of all the single character command options.

The purpose of the X times X and Y times Y data source options in table 7.2 is to output a flat line on Radio SkyPipe chart recorder, corresponding to the mean square voltage or power

received on the selected channel. The multiplier computes the square and the moving average filter determines the mean. The chart recorder output can be used to adjust the X channel and Y channel IF gains until the output level of the TVRX2 IF signal output on each channel is equal and calibrated. A calibration signal source will be required for this. Calibration is discussed further in chapter 9.

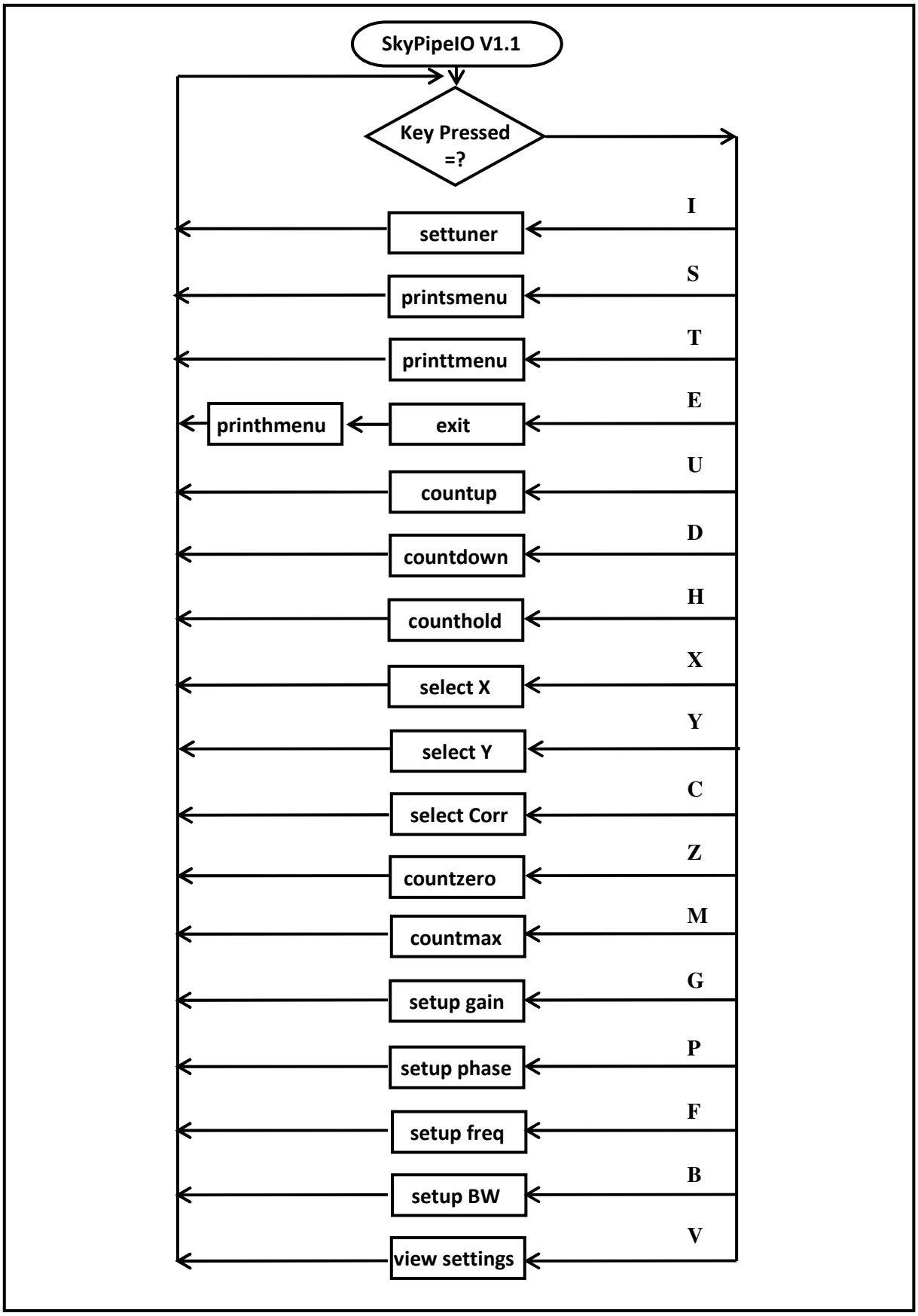


Figure 7.4: Microcontroller firmware flowchart – main loop

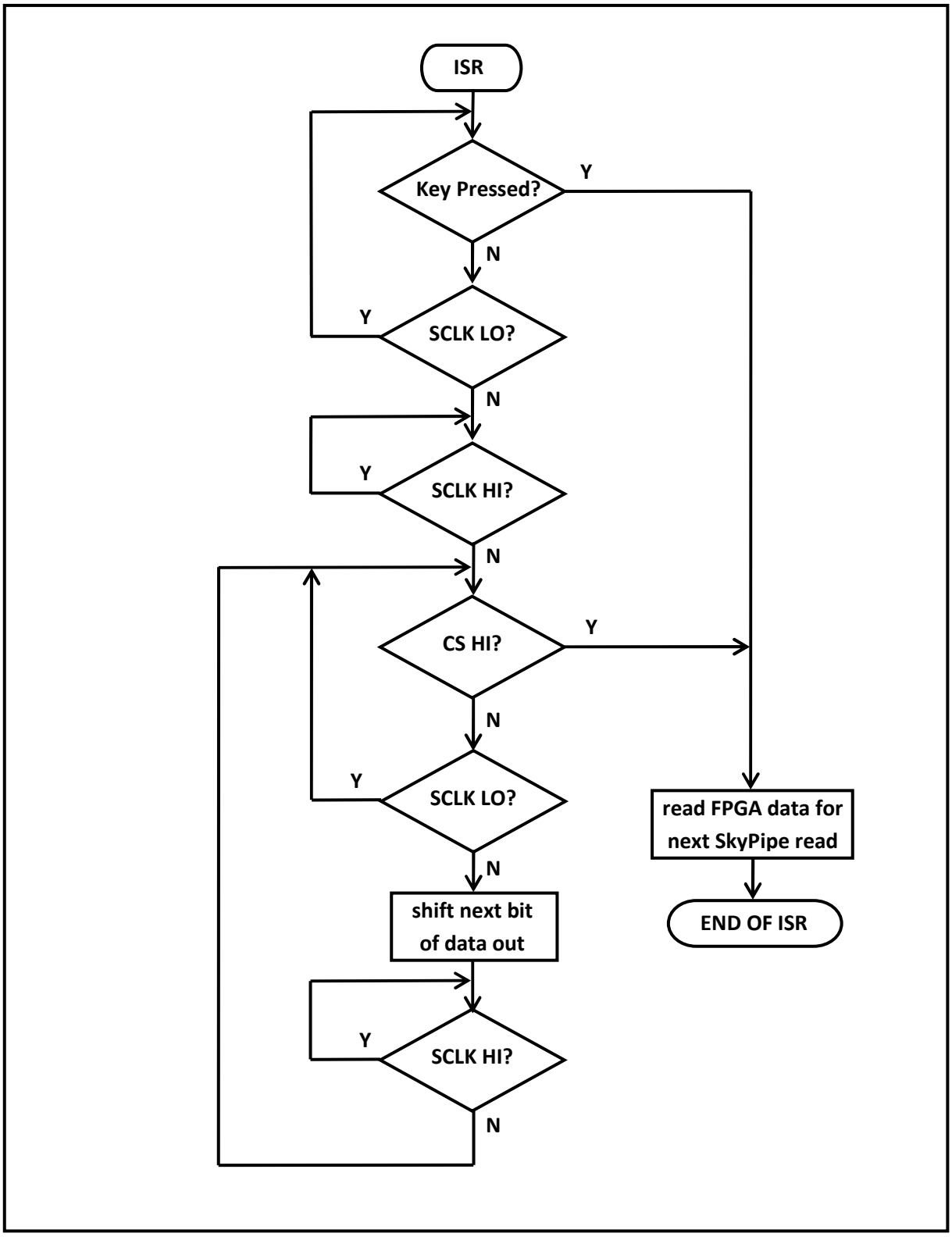


Figure 7.5: Microcontroller firmware flowchart – interrupt service routine showing the logic and timing applied to the Radio SkyPipe synchronous data transfer control signals (SCLK and CS) whereby the microcontroller firmware emulates a MAX187 ADC IC. The interrupt is initiated by a transition from high to low on the chip select (CS) control line.

Table 7.2: Interface board command options. These single character commands are entered using Hyper Terminal, a terminal emulator program running on the PC.

I -	Initialize	-	Initializes TVRX2 silicon tuners
S -	Source	-	Displays a menu of data source options
T -	Test	-	Displays a menu of test data options
E -	Exit	-	Stops reading data from FPGA correlator
U -	Up count	-	Test data up counter
D -	Down count	-	Test data down counter
H -	Hold count	-	Test data hold counter value
X -	Select X	-	Select X times X input to moving average filter
Y -	Select Y	-	Select Y times Y input to moving average filter
C -	Select Correlate	-	Select X times Y input to moving average filter
Z -	Zero	-	Zero test data counter
M -	Maximum	-	Set test data counter to maximum
P -	Phase	-	Set phase offset between channels X and Y
F -	Frequency	-	Set tuner centre frequency
B -	Bandwidth	-	Set tuner IF bandwidth
V -	View settings	-	View frequency, phase and bandwidth settings

7.4.3 Setting up the TVRX2 daughterboard tuners

The TDA18272 tuners on the TVRX2 daughterboard are initialized by the PIC microcontroller firmware to 100 MHz centre frequency and 10 MHz IF bandwidth. A common setup error that occurred during development was forgetting to install the FPGA configuration using Impact. The PIC microcontroller firmware is then unable to communicate with the TDA18272 silicon tuners, since this communication is routed through the FPGA as described in section 7.3.7, and a timeout message is displayed on Hyper Terminal. Details of this can be seen in lines 810 to 820 in the “settuner” function of the C program listing in Annexure A [29]. Although a control signal from the silicon tuners, as seen in the bottom right-hand corner of Figure 7.3, is called IRQ (interrupt request), it is not treated by the PIC

microcontroller as an interrupt but is polled in lines 810 to 820 of the C program listing. The IRQ signals from both tuners on the TVRX2 board are OR'ed together in the FPGA to generate one IRQ signal. The microcontroller initializes one TDA18272 tuner completely and then the other one. The purpose of the IRQ signal is to cause the microcontroller to wait for the TDA18272 to complete a state change before continuing with the rest of the TDA18272 setup sequence [31]. The TDA18272 data sheet showing the state diagram and associated IIC communication sequence is given in Annexure B. The complete sequence of addresses and data for initializing the tuners is stored in the EEPROM memory of the PIC microcontroller. This can be seen in lines 27 to 37 in the C program listing of Annexure A [29]. In total, 35 bytes of data are sent to various register addresses in the TDA18272 [31]. A frequency change is a subset of the initialization sequence and this is performed when the set frequency command "F" is entered at Hyper Terminal. Details of this can be seen in lines 966 to 1003 in the "setfreq" function of the C program listing in Annexure A [29]. Only seven bytes are sent for a frequency change [31], starting from byte 28 of the initialization sequence stored in the PIC EEPROM.

Chapter 8

System sub-assembly and final assembly testing

8.1 Introduction

To contain the number of variables and unknowns, the development of a complex system consisting of several hardware and software sub-systems should be undertaken in a modular and systematic way. The complete system development strategy needs to provide for the independent development and testing of each sub-system. Only after some verification testing has been successfully completed on each sub-system independently, should final system integration and final assembly testing be attempted. Development means a continuous process of improvement. The development of each sub-system should involve a systematic process of manageable changes to some fully tested starting platform with thorough testing to verify the expected improvement after each set of manageable changes. In this way, any observed system malfunction is traceable to a manageable set of changes in a particular sub-system.

The contents and associated development activities of the preceding chapters show that the development of a two element correlating radio telescope interferometer was sufficiently complex to illustrate the importance of a modular and systematic development strategy to a successful outcome.

8.2 Testing the interface with Radio SkyPipe and Hyper Terminal

A prototype interface board was built and test code written for the PIC16F689 microcontroller on the prototype interface board. The objective was to develop and fully test the communication of data between the PC and the prototype interface board. For these tests, the

interface board was not connected to the USRP2 but was connected via two cables to the PC alone and powered from a laboratory power supply. A test data facility was programmed into the microcontroller that could provide output data to Radio SkyPipe that would count up or down, hold its count value, be zeroed or set to maximum value. The microcontroller test data facility was written so that the various test counter options could be selected by entering commands at Hyper Terminal on the PC. A PC screen shot of these tests is shown in Figure 8.1. The test commands are issued by pressing a single upper case character key. The microcontroller displays a confirmation message on the Hyper Terminal screen. These sub-system tests revealed a problem with the interrupt latency in the microcontroller emulation of a MAX187 ADC IC. As discussed in section 7.4.1, Radio SkyPipe was occasionally missing the first (MSB) bit of the 12-bit serial data communication from the interface board. This occasionally resulted in large spikes going up or down from the desired plot output which indicated an MSB error in the data transfer from the interface board to Radio SkyPipe. This was solved by having the first bit of output data ready before the interrupt signal from Radio SkyPipe occurred, by pipelining the data read from the FPGA correlator.

The value of identifying this problem in a sub-system test was that the cause was limited to data transfer problems between the interface board and Radio SkyPipe. Neither the FPGA correlator nor the front-end receiver sub-systems was in question and full confidence in the interface board sub-system was established before proceeding with other sub-system testing. Once the prototype interface board was completely tested, a schematic design was captured and PCB layout was designed using Mentor Graphics PADS PCB design software. The final PCB Gerber files were sent out for PCB manufacture while testing proceeded to the next sub-system; the FPGA correlator.

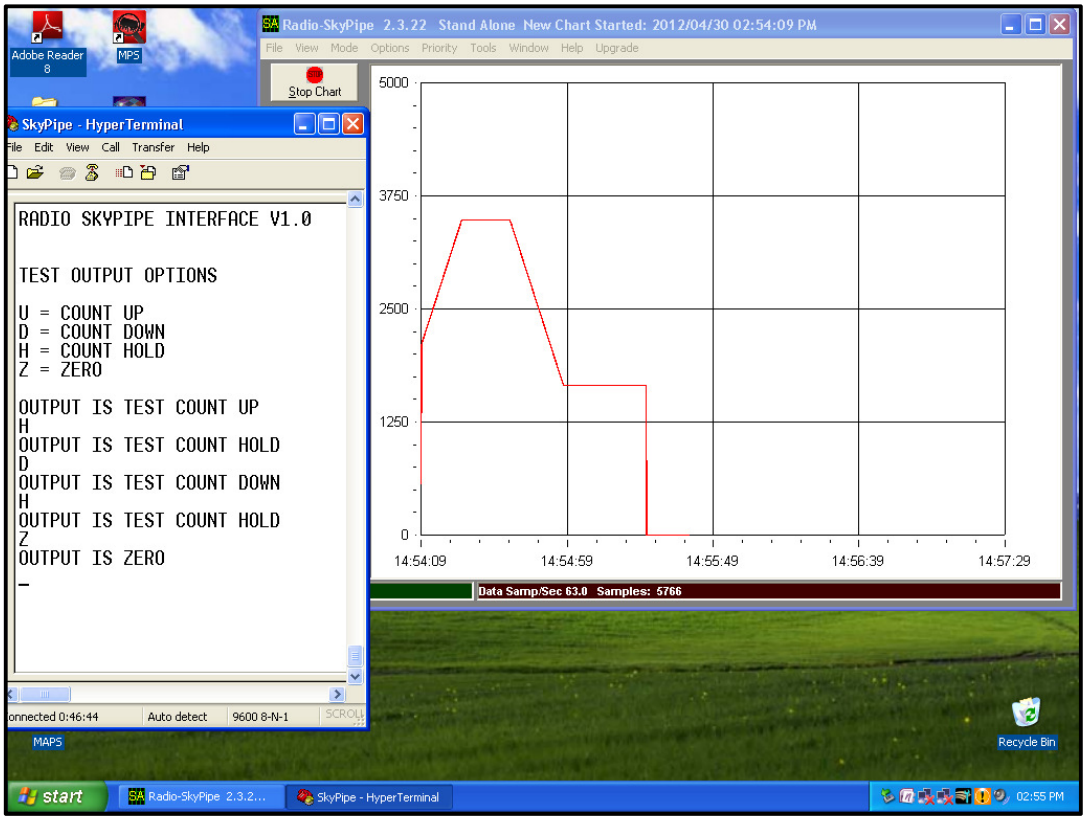


Figure 8.1: PC screen shot of interface board testing showing the use of the built-in firmware test routines (count up, count hold, count down and zero). Each test routine is initiated by pressing a single character (such as D) in the Hyper Terminal window on the left. The Radio SkyPipe chart recorder then traces a straight line going down. The test options default to counting up.

8.3 Testing the FPGA correlator

The FPGA correlator sub-system was first implemented and tested on a Xilinx University Program (XUP) Virtex5 development board and was tested at audio frequencies [28], [32]. This eliminated the possibility of hardware design errors and any system malfunction was traceable to the FPGA configuration design. Testing at audio frequencies meant that standard laboratory equipment could be used and that wiring, cabling and impedance matching would not be a consideration. For these audio frequency tests, the FPGA sample rate was set to 48,828 kHz resulting in an integration time from equation 7.9 of:

$$\tau_I = 2048 \times \tau_S = 2048 \times \frac{1}{48828} = 41{,}94 \text{ ms} \qquad 8.1$$

Figure 8.2 shows the XUPV5 development board on the left at the point of migration to the USRP2 with the prototype interface board, shown on the right. Figure 8.3 shows the standard laboratory equipment used for testing at audio frequencies. The top instrument is a 100 MHz digitizing oscilloscope and the bottom instrument is a 25 MHz dual arbitrary waveform generator. The waveform generator was set to output two 1 kHz sine waves, however the frequency of the channel shown at the top of the screen was set a little bit higher, to 1.00001 kHz. The oscilloscope was triggered using the top sine wave which therefore appeared stationary on the oscilloscope screen while the bottom sine wave was seen to continually shift in phase relative to the top sine wave. The slight frequency offset was used to simulate the continuously increasing phase difference that occurs in baseline pair receivers, due to Earth rotation. For the two sine wave input signals that were used, the correlator should produce a sine wave output with a period equal to the inverse of the difference in frequency between the two input sine waves, as shown in the waveforms of Figure 3.2.

$$T_{\text{correlator o/p}} = \frac{1}{\Delta\nu} = \frac{1}{1{,}00001 \times 10^3 - 1 \times 10^3} = \frac{1}{1 \times 10^{-2}} = 100 \text{ s} \qquad 8.2$$

The correlator output is shown in Figure 8.3 and has a full wave rectified shape. The reason for this is that Radio SkyPipe will not accept negative digital values and so the negative half cycle was flipped in the FPGA using an absolute value conversion block that can be seen in the centre of Figure 8.4. This was later replaced with a constant (DC) offset conversion so that the normal sine wave cycle shape could be seen on Radio SkyPipe with all digital values being positive. This DC offset block can be seen in the centre of Figure 7.1. The full sine wave cycle in Figure 8.5 can be seen to have a period of 100 s. The time scale of the chart recorder graph shown in Figure 8.5 is actual time of day.

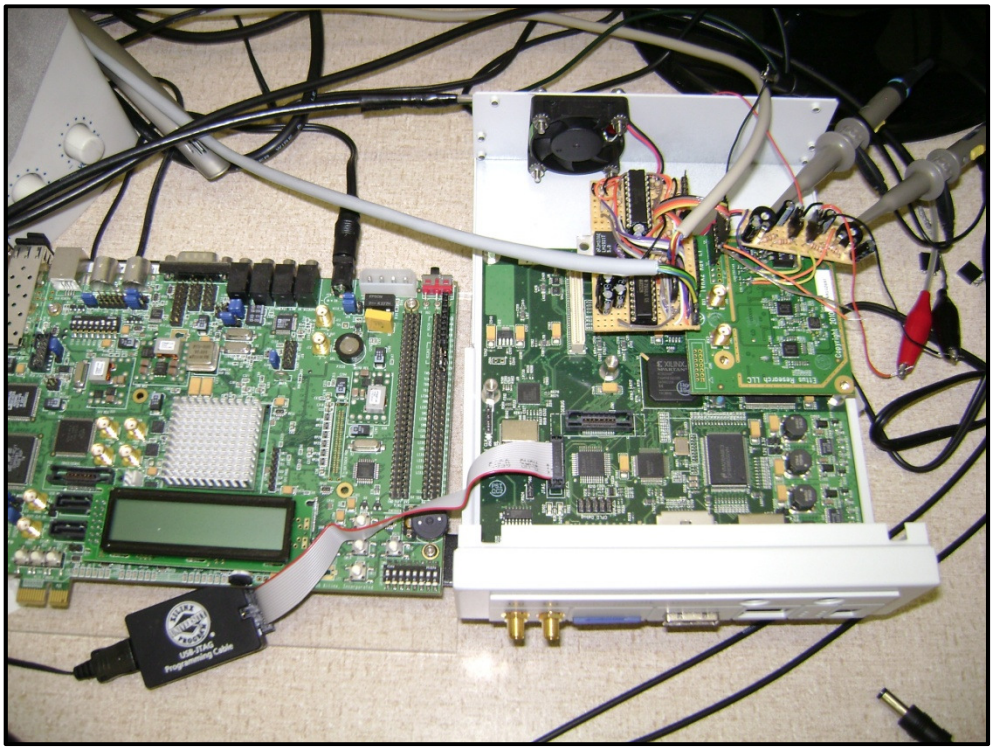


Figure 8.2: Hardware platforms for FPGA correlator testing showing the point of migration from testing on the Xilinx XUPV5 FPGA development board (on the left) to the USRP2 target system.

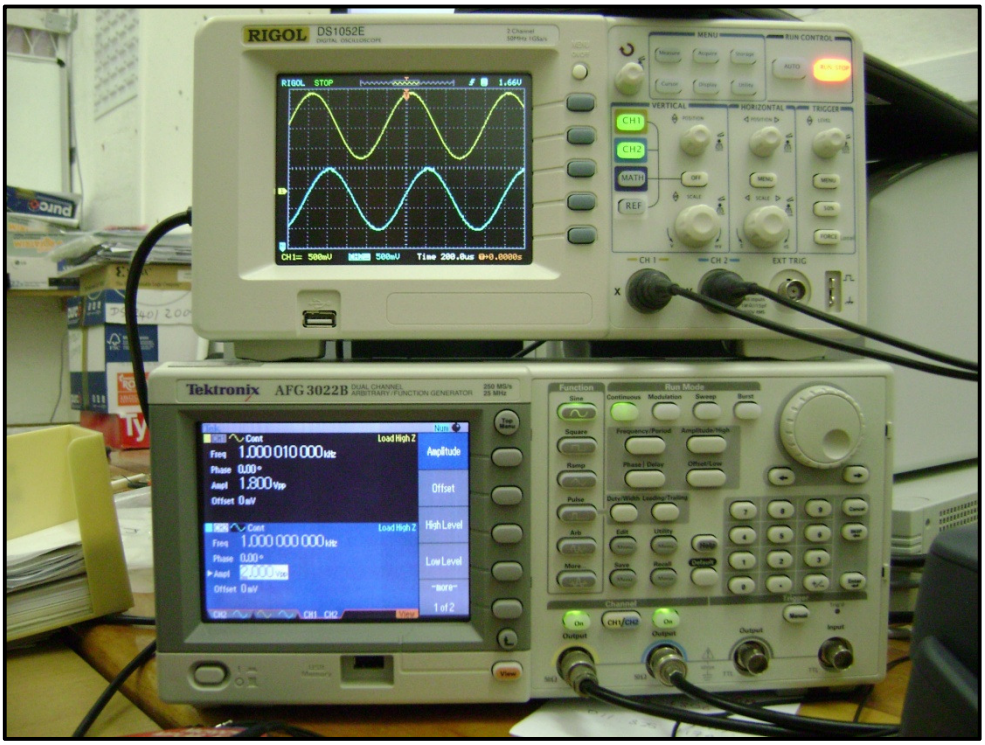


Figure 8.3: FPGA correlator testing with audio frequency test equipment showing the use of standard low frequency laboratory equipment (a function generator and an oscilloscope) to inject into the FPGA correlator, two 1 kHz sine waves with continuously shifting phase relationship.

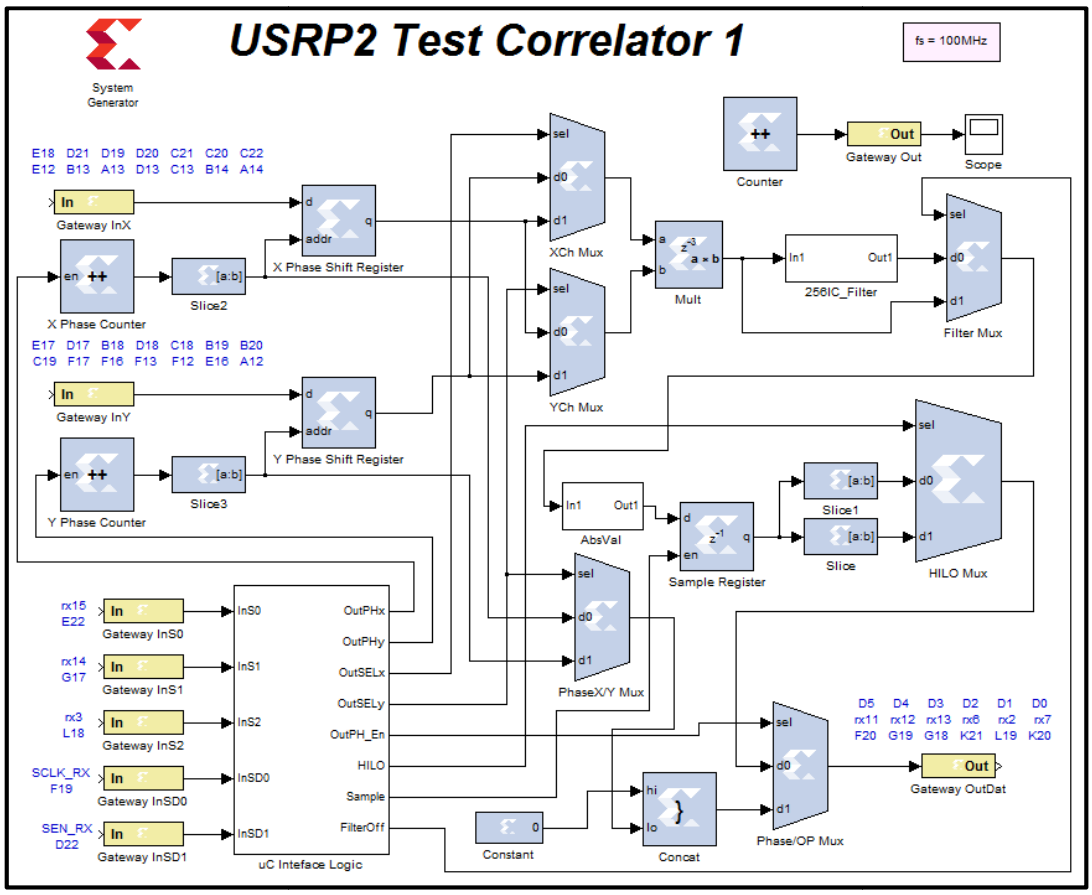


Figure 8.4: USRP2 FPGA test correlator 1 schematic diagram. An early version of the FPGA correlator used during the audio frequency tests.

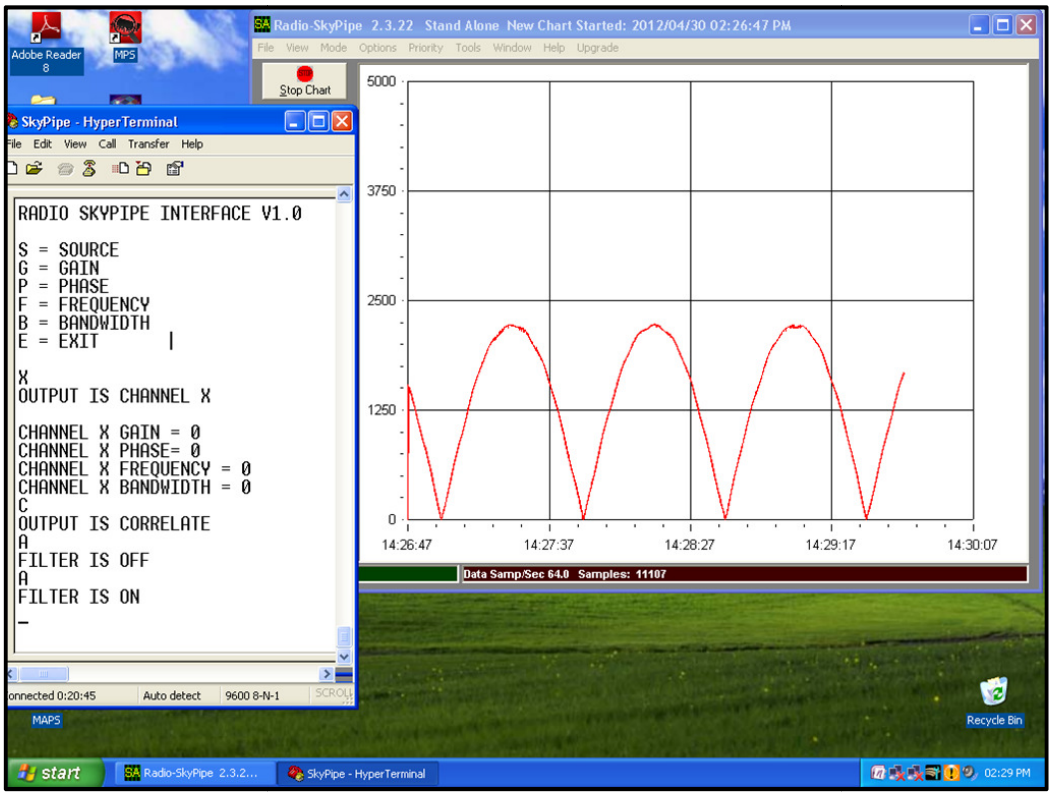


Figure 8.5: Test result for FPGA correlator for 1 kHz sinusoidal inputs. This early version of the correlator used an absolute value process to flip the negative half cycle of the correlation output.

Once the FPGA correlator sub-system had been fully tested with 1 kHz audio test signals, the sub-system was re-tested while increasing the signal frequency in decades up to 10 MHz. This was done so that full confidence in the FPGA correlator could be established before proceeding to develop the RF front-end sub-system. The RF front-end TVRX2 dual tuner daughterboard produces a 5 MHz centre frequency, 10 MHz bandwidth IF signal output. Therefore an IF output signal frequency range of 0 – 10 MHz.

8.4 Testing the TVRX2 daughterboard tuners and the final assembly

After the interface board sub-system and the FPGA correlator sub-system had been tested, the final assembly was completed and tested. Any problems uncovered in this testing could be traced to the RF front-end sub-system, since all other sub-systems had been thoroughly tested. The RF front-end sub-system consists of the TVRX2 daughterboard and the setting up of the TDA18272 silicon tuner IC's on this daughterboard. The final hardware assembly can be seen in Figure 6.6.

The initial test consisted of injecting a 100 MHz, -50 dBm sine wave signal into each of the RF inputs and activating the TDA18272 initialization sequence programmed into the interface board microcontroller. Oscilloscope probes were placed on the analogue IF output signals of the TDA18272 tuners and after editing the PIC C code for the setting of the IF gain, a 5 MHz sine wave IF signal was seen on the oscilloscope. The next test was to attach antennas to the RF inputs of the TVRX2 daughterboard and to see what signals could be received off-air. Figure 8.6 shows the test equipment setup for this. In this picture, the sine wave injection from the RF signal generators had been replaced by two LPDA antennas, one of which can be seen in the picture. The receivers were tuned to 570 MHz and the analogue IF output signals

from the TDA18272 silicon tuners, seen on the digitizing oscilloscope in Figure 8.6, is the off-air DSTV Walker Mobile signal down-converted to 5 MHz. The amplitudes and phase difference of the two received IF signals could be clearly seen to change as the antennas were rotated on a jig that was constructed for this purpose. The correlator output was also seen to rise and fall as the jig was rotated and remained constant while the jig remained stationary, as expected. The correlator output was seen to be very sensitive to the phase difference between the two received signals, changing rapidly as the phase difference changed and remaining constant when the phase difference remained constant. There was also some, but significantly less, change in the correlator output as the amplitude of the two input signals changed. In order to obtain recognizable results, the antenna jig seen in Figure 8.6 was further developed to facilitate slow constant speed rotation that would simulate Earth rotation and allow for laboratory measurements and calibration of the system. The details of this development and the measurements obtained are presented in chapter 9.

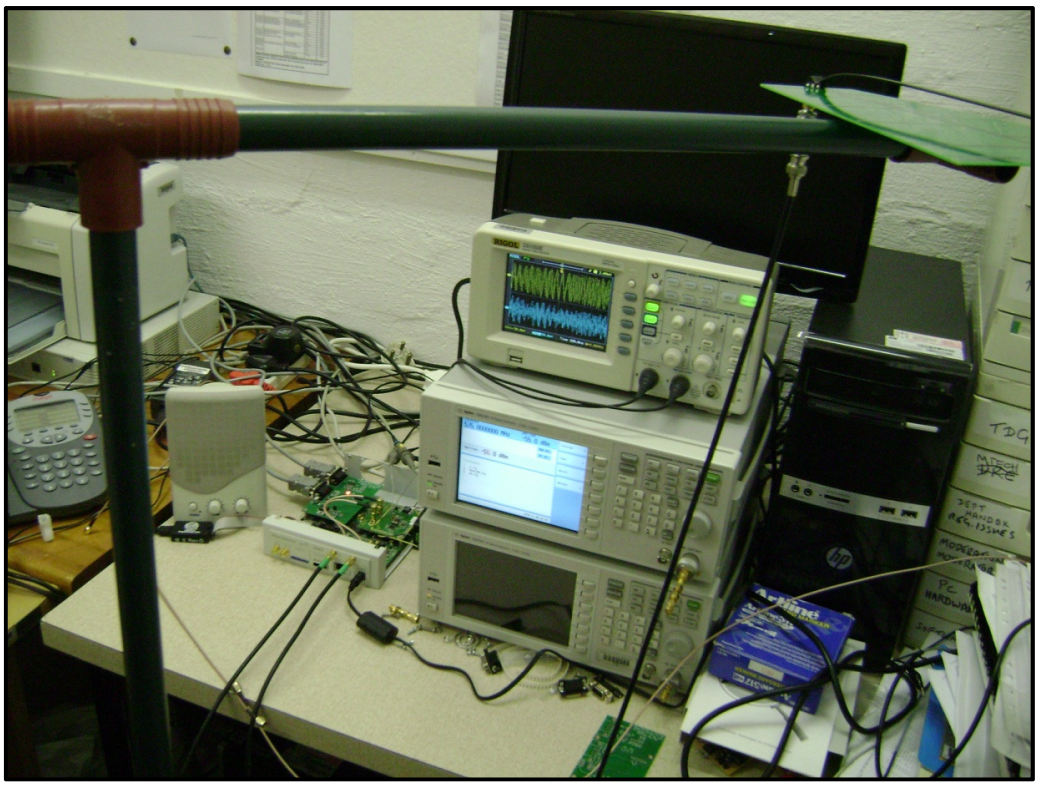


Figure 8.6: RF front-end sub-system test setup. Audio test equipment is first replaced by two phase locked RF frequency generators to provide input to the correlator and later replaced by off-air signals from two LPDA antennas. One of which can be seen in the photograph.

Chapter 9

Comparative results and calibration

9.1 Introduction

As illustrated in Figure 3.2, for a single frequency point radiating source, the two element correlating interferometer essentially measures the phase difference between the signals received at two spatially separated points. This phase difference will vary as the point source rotates across the field of view, as shown in Figures 3.5 and 3.7. As discussed in chapter 3, the correlation of the two received signals produces a fringe oscillation function. The amplitude and phase of the fringe oscillation function constitute a sample of the complex visibility function of baseline distance. An inverse Fourier transform of the visibility function is an image of the radiating sources in the field of view. The two element correlating radio telescope interferometer produces one sample of the complex visibility function for each value of baseline distance [11], [12], [13].

A laboratory RF signal generator connected to an LPDA antenna at some distance from the baseline receiver pair was chosen as a simple test radiating source to obtain fringes from the completed two element correlating interferometer. The advantage of the LPDA antenna over other antennas is that it is broadband, allowing for tests at different frequencies without needing to replace the antenna. For these measurements, a very short baseline distance was used and rotating the baseline antenna pair relative to a fixed point radiating source was easier than rotating a point radiating source around the fixed baseline. A rotating baseline pair calibration jig was constructed to do this with a baseline distance of two metres.

9.2 Rotating baseline pair calibration jig

The rotating baseline pair calibration jig was constructed using an old swivel chair base and some PVC irrigation riser pipes and fittings as shown in Figure 9.1. The baseline pair in Figure 9.1 has a baseline distance of two metres. A PVC waste pipe end cap was fixed at the base of the upright rotator to provide a pulley for a belt drive to rotate the baseline pair at a constant angular speed. Figure 9.2 shows the addition of an old electric clock mechanism and office elastic band belt drive. Three office elastic bands were bonded together with rubber cement to solve a problem of dead time followed by a slow acceleration that occurred after switch on as a single elastic band belt stretched to overcome the jig rotational inertia. The triple laminated belt drive provided a tighter coupling between the clock mechanism and the upright rotator which in turn provided fast acceleration to the required constant speed of rotation, analogous to the rotation of the Earth. The LED's on the rotator electronics, which can be seen in Figure 9.2, indicate the rotational speed setting. There is a potentiometer above the smoothing capacitor which could be used to adjust the rotational speed setting. A PIC16F689 microcontroller was used to determine the potentiometer position, switch on the corresponding LED and provide the electric clock mechanism with a drive signal according to the speed setting. The required drive signal is an AC square wave of around 30 V(pp). The frequency of the square wave, in the order of 10 – 100 Hz, drives a solenoid in the clock mechanism back and forth which rotates the first gear of the clock mechanism gear chain.

Figure 9.1: Rotating baseline pair calibration jig showing two PCB LPDA antennas mounted two metres apart on a rotating jig.

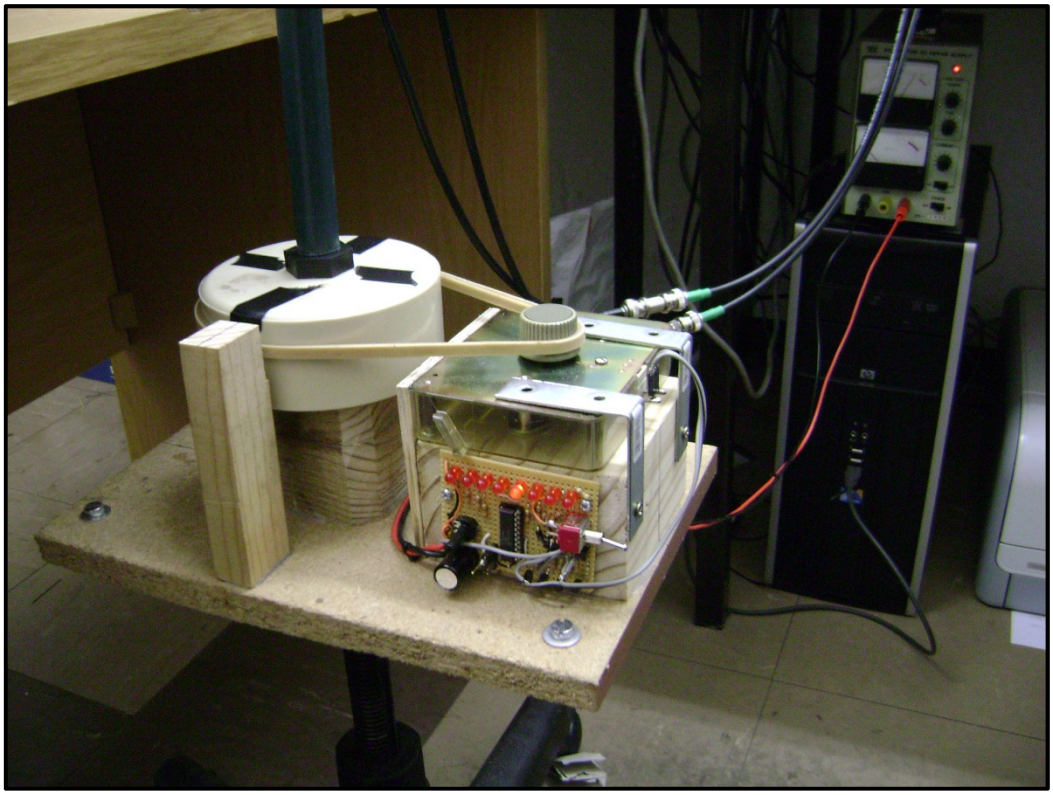


Figure 9.2: Rotating baseline pair belt drive showing the electric clock mechanism, the variable speed electronics, and the office elastic band belt drive onto the upright rotator. This construction was used to obtain suitably slow rotation of the baseline antenna pair for system test purposes.

9.3 Final fringe measurement results

The Radio SkyPipe PC chart recorder window was maximised to full screen and the chart grid settings were adjusted to obtain a better graph compared to Figures 8.1 and 8.5. The vertical scale, which is the digital value from the correlator, was adjusted to have a horizontal line at intervals of 500 and tick marks at sub-intervals of 100. The midpoint value of the FPGA correlator output is 2048 and the maximum value is 4096. This is offset by a constant, as discussed in section 8.3, in order to avoid negative digital values that are not accommodated by the Radio SkyPipe chart recorder software. A digital output of 2048 from the FPGA therefore corresponds to a correlator output of zero.

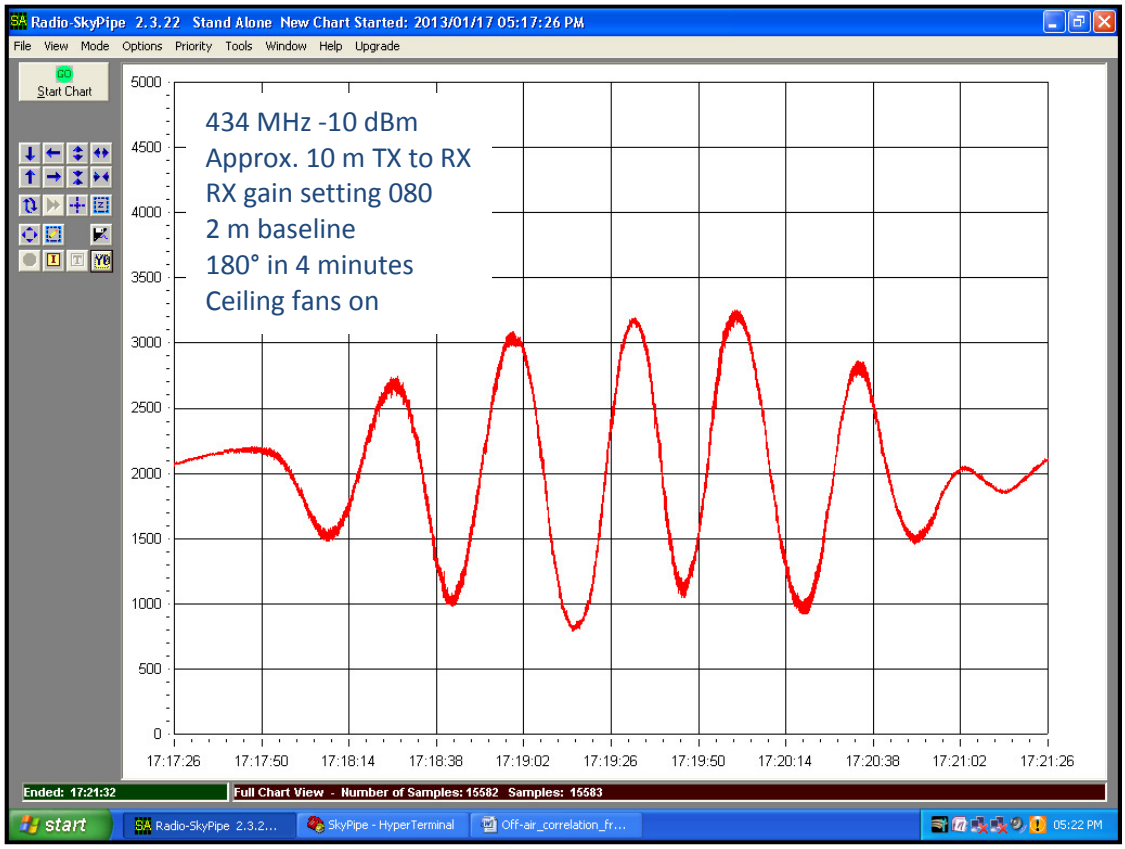


Figure 9.3: Radio SkyPipe interference fringe plot 1 showing the noise introduced by the rotating laboratory ceiling fans.

An unexpected outcome of the results as seen in Figures 9.3 and 9.4 is the effect of the laboratory ceiling fans on the correlator output. The fuzziness or noise on the fringe result of Figure 9.3 occurred when the laboratory ceiling fans were on. This noise was completely absent, as seen in Figure 9.4, when the same measurement was taken with exactly the same instrument settings and only minutes later but with the ceiling fans off and stationary. Repeated testing showed that movement of physical objects between the radiating source and the receiving baseline antenna pair was a significant source of measurement noise.

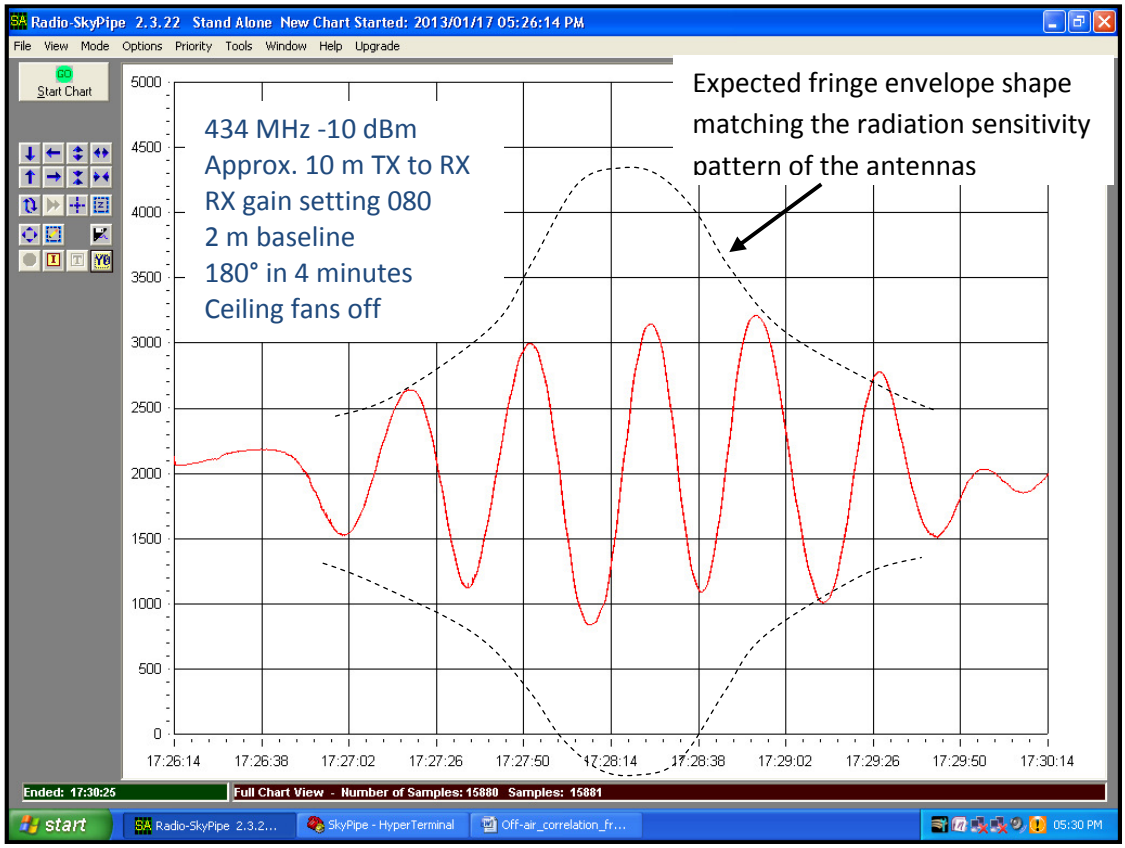


Figure 9.4: Radio SkyPipe interference fringe plot 2 showing the absence of the noise seen in Figure 9.3 when the ceiling fans are off and stationary and the effect of RF stage AGC on the fringe amplitude envelope. The repeatability of the correlation fringe shape is evident by comparison with the plot of Figure 9.3.

The radiating source frequency was restricted to the ISM licence free bands at 434 MHz and 868 MHz and the transmitter power level was kept very low at -10 dBm. The tests were performed inside the laboratory with two dry wall partitions between the radiating source and

the receiving baseline pair. The presence of reflecting surfaces and obstructions in the physical setup was far from ideal however some positive and useful results were obtained. The same setup was taken onto the roof to attempt to get some free field measurements but the rotating baseline receiver pair jig could not stand up to the gusty wind and there were high levels of RFI in the UHF band, which rendered the results unusable.

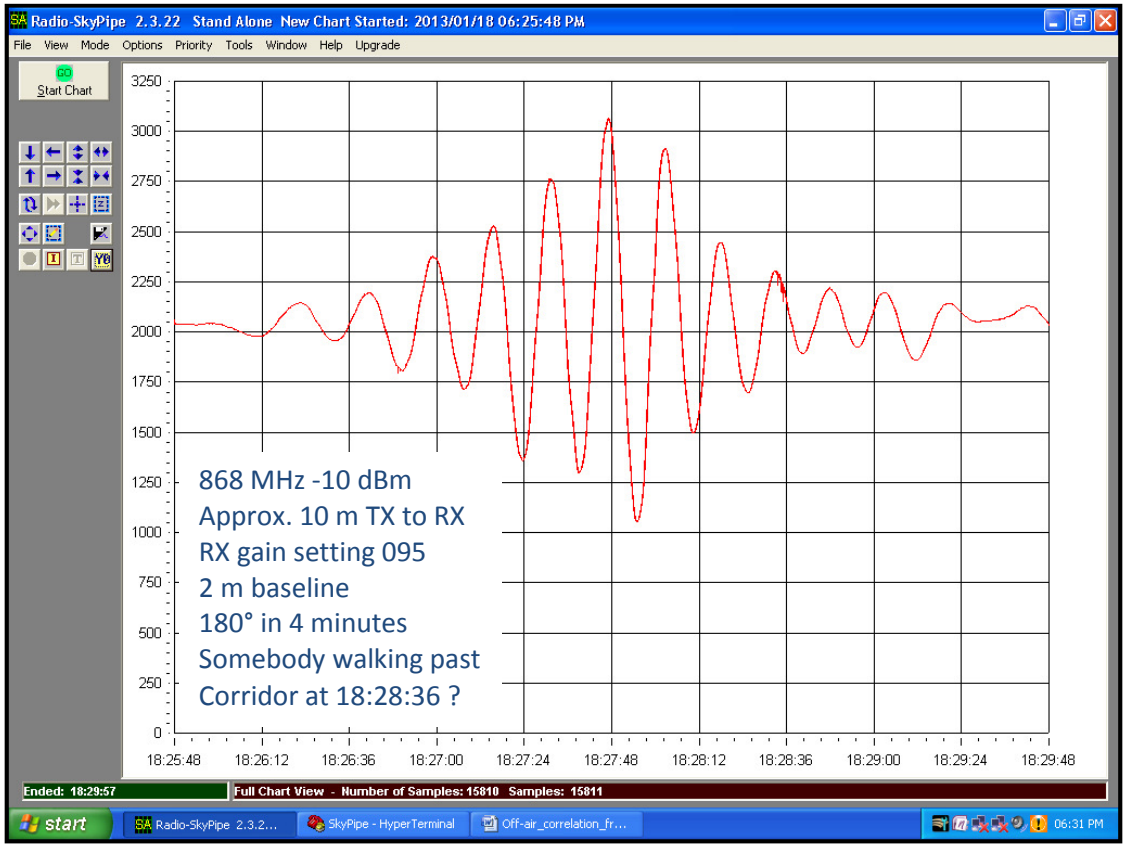


Figure 9.5: Radio SkyPipe interference fringe plot 3 showing the correct fringe amplitude envelope shape as well as the expected reduced fringe spacing compared to Figure 9.4. An isolated noise component is identified as a person walking past nearby the test set up.

The effect of the beam pattern of the LPDA antennas can be clearly seen in the 868 MHz plots of Figures 9.5 and 9.6. If the received signal strength is too high then RF stage AGC may cause a reduction of the overall receiver gain during a plot of the fringe function. This may happen as the rotating jig approaches the direction of maximum received signal strength, resulting from the radiation pattern of the antennas. The receiver gain may then automatically drop in this region, resulting in a flattening of the beam pattern. Evidence of this can be seen

in the reduced amplitude of the fringe cycle lying between times 17:28:14 and 17:28:48 in Figure 9.4. Lower frequency radiation has lower free space path loss (FSPL) and since the radiating source signal level was the same for both the 434 MHz and the 868 MHz tests, the signal level received for the 434 MHz tests would have been higher [33].

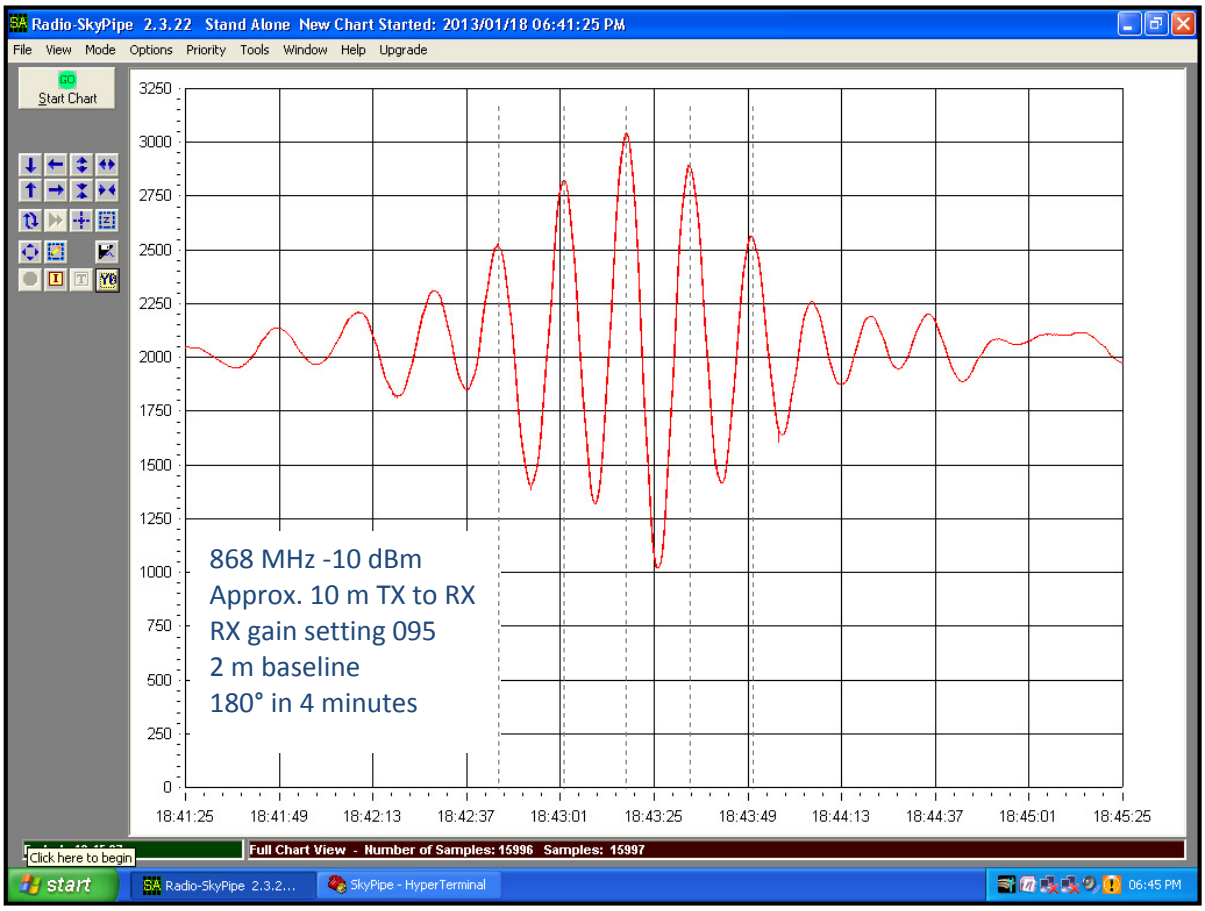


Figure 9.6: Radio SkyPipe interference fringe plot 4 showing the repeatability of the fringe shape by comparison with Figure 9.5. The isolated noise component is now absent. The fringe period is marked for verification of the fringe spacing according to the theory presented in chapter 3.

The free space path loss in dB ($FSPL_{dB}$) with distance d in kilometres and frequency f in megahertz is given by [33]:

$$FSPL_{dB} = 32{,}45 + 20\log_{10}d + 20\log_{10}f \qquad 9.1$$

So for the 434 MHz and 868 MHz tests conducted, the received signal strength under free field conditions would have been:

$$P_{RX}(434)_{dBm} = P_{TX}(434)_{dBm} - FSPL(434)_{dB} + G(TX\ ant)_{dB} + G(RX\ ant)_{dB} \quad 9.2$$

$$= -10\ dBm - (32{,}45 + 20\log_{10}0{,}01 + 20\log_{10}434)\ dB + 6\ dB + 6\ dB$$

$$= -10\ dBm - (32{,}45 - 40 + 52{,}75)\ dB + 12\ dB = -43{,}20\ dBm$$

$$P_{RX}(868)_{dBm} = P_{TX}(868)_{dBm} - FSPL(868)_{dB} + G(TX\ ant)_{dB} + G(RX\ ant)_{dB} \quad 9.3$$

$$= -10\ dBm - (32{,}45 + 20\log_{10}0{,}01 + 20\log_{10}868)\ dB + 6\ dB + 6\ dB$$

$$= -10\ dBm - (32{,}45 - 40 + 58{,}77)\ dB + 12\ dB = -49{,}22\ dBm$$

9.4 Channel gain adjustment and calibration

In order to focus on the effect of phase difference on the correlator output, the analysis of section 3.8 considers the correlation of two phase offset sine waves of unity amplitude and the same frequency. The effect of scaling factors in the amplitudes of the two input sine waves is ignored. The amplitude of the correlator output is however also dependant on the amplitudes of the input signals and the gain of each channel. The Visibility function is obtained from the amplitude and phase of the correlator fringe output as a function of baseline distance. Therefore careful calibration of input signal amplitude scaling factors is required. Each fringe measurement may take up to an hour and different baseline samples may be taken on different days. The channel gains need to be kept constant during a fringe measurement and from one fringe measurement to another, in order to prevent fringe amplitude errors.

The gain setting values shown in Figures 9.3 to 9.6 are the digital number sent to the DAC that connects to the IF gain adjustment input on the TDA18272 silicon tuner. The input number range is 0 to 255 and the DAC output voltage range is 0 V to 3,3 V. According to Figure 19 on page 46 of the TDA18272 data sheet (attached as Annexure B) the gain adjustment is obtained by applying voltages in the range 0,6 V to 1,6 V [31]. This corresponds

to DAC digital input quantities of 46 to 123. The gain adjustment sensitivity according to Figure 19 [31] is 30 dB/V and so the gain increment per digital value increment is:

$$\Delta G = 30\ \text{dB/V} \times \Delta V = 30\ \text{dB/V} \times \frac{(1{,}6\ \text{V} - 0{,}6\ \text{V})}{(123 - 46)} = 0{,}4\ \text{dB} \qquad 9.4$$

The PIC microcontroller firmware could be adjusted to give finer IF gain adjustment increments which in turn will facilitate better calibration resolution. The C program details can be seen in function “setgain” in lines 886 to 911 of the C program listing in Annexure A [29].

Referring to page 37 of the TDA18272 silicon tuner specifications in Annexure B, the RF stage amplifier gain automatically adjusts from about -28 dB to about +86 dB. The IF stage amplifier adds up to 30 dB of gain according to the DC voltage applied to the IF gain adjustment pin. The TDA18272 has an RF AGC synchronization feature that should only allow RF AGC gain changes during a VSYNC pulse applied to the VSYNC pin of the TDA18272 [31]. The VSYNC pins are connected together in the FPGA and connected to port pin RC3 of the PIC microcontroller, as can be seen in Figures 6.2 and 7.3. Line 135 of the C program listing of Annexure A shows the declaration of VSYNC, which is configured as an output in line 344 and set to zero in line 348 [29]. No further software for VSYNC was implemented and so this should disable any RF AGC gain changes. It is suspected, as discussed in section 9.3, that the RF AGC is not disabled. This requires further investigation.

9.5 Fringe spacing

The fringe measurements taken at 868 MHz, shown in Figures 9.5 and 9.6, show the relationship between the wavelength of the radiated EM wave, the baseline distance and the fringe spacing. Although the same baseline distance is used, the baseline distance at 868 MHz

is double the number of wavelengths compared to the 434 MHz measurements. This gives rise to double the number of fringe cycles or half the fringe spacing as seen in Figure 9.5 compared to Figure 9.4.

In the plots of Figures 9.5 and 9.6, the rotational speed of the baseline pair jig is approximately 180 degrees of rotation in four minutes:

$$\frac{\theta}{t} = \frac{180^\circ}{240\ s} = 0{,}75^\circ/s \qquad 9.5$$

From Figure 9.6, the period of the fringe oscillation is approximately 15 s which, from equation 9.5, corresponds to 11,25 degrees of rotation of the 2 m baseline. According to equation 3.8, the fringe function has a maximum at $l = 0$ and again at $l = \lambda/d$. Furthermore, $l = \sin\theta \approx \theta$ for small θ. From Figure 3.7, θ is the angle between the normal to the baseline and the direction of the radiating source. From equation 1.1, the wavelength for the EM wave that produced the fringe of Figure 9.6 is:

$$\lambda = \frac{c}{\nu} = \frac{3 \times 10^8}{868 \times 10^6} = 345{,}6\ mm \qquad 9.6$$

This gives a fringe spacing of:

$$\Delta l \approx \Delta\theta = \frac{\lambda}{d} = \frac{345{,}6}{2000} = 0{,}1728\ rad = 9{,}9\ degrees \qquad 9.7$$

The difference between the actual fringe spacing measurement of 11,25 degrees and the calculated theoretical fringe spacing of 9,9 degrees could be attributed to the fact that the formulae derived in section 3.8 are based on very distant sources relative to the baseline distance. This leads to the assumption, as shown in Figure 3.7, that the EM wave arriving at the baseline receiver pair is approximately a plane wave. In the laboratory tests of Figures 9.3 to 9.6, the source was a mere ten metres away for a two metre baseline and so the plane

incident wave assumption does not hold. Also, the baseline distance of two metres is only approximate since the LPDA antennas were temporarily attached to the rotating jig using cable ties. A more permanent fixing was intended that would allow the antennas to be twisted around into either horizontal or vertical polarization, but time did not permit this.

Compared to measurement of real cosmic sources using Earth rotation, the speed of rotation of the test jig was much faster than Earth rotation but the baseline (2 m) was very short. The rotating test jig took approximately four minutes to rotate through 180 degrees, whereas Earth takes 12 hours, which is 180 times longer. So comparable fringe spacing, measured in time, would only occur with a real cosmic point source for a baseline spacing of ~360 m. Up to 100 m baseline spacing will result in a fringe spacing of greater than a minute for real cosmic point sources measured at the same centre frequency as the test result shown in Figure 9.6.

9.6 Sources of measurement noise

For the measurements of Figures 9.5 and 9.6, the ceiling fans remained off. It was found however that the movement of people in the vicinity was a significant source of noise in the resultant plots. Movement of people inside the laboratory between the radiating source and the baseline receivers rendered the results virtually unusable. Measurements were generally taken at night, especially in university term time when there were people in the laboratories during the day. In the measurement of Figure 9.5, it is believed that the observed disturbance of the plot at 18:28:36 was due to a person walking along the corridor, outside the laboratory where the test measurement was conducted, but between the radiating source and the receiving baseline antennas. This moving person would have provided a time varying multi-path signal combination between the direct and reflected waves which would have produced a

small time varying phase difference, changing at the speed of movement of the person walking past.

When real cosmic sources are measured, sources of measurement noise will include man-made radiation from terrestrial and satellite sources, as well as natural cosmic sources outside the field of view that are radiating into side lobes of the receiving antennas, as discussed in section 1.6. The thermal noise generated inside the receiver electronics is less significant than for single element radio telescopes since it is not coherent for the two receivers and its effect should reduce as the integration time is increased, as discussed in section 7.3.6.

9.7 Instrumental phase offset measurements

As discussed in section 9.1, the two element correlating interferometer measures the phase difference between the signals received at the two receivers. Any phase difference between the signals of the two channels introduced by the measuring instrument, known as instrumental phase offset, especially when unpredictable, poses a problem. Unfortunately, the TDA18272 silicon tuner introduces an unpredictable phase difference between the RF signal received at the antenna and the downshifted IF signal output. The method used to verify this was to inject two RF sine wave signals at the same frequency into the TDA18272 tuners from two laboratory signal generators that are locked in phase. The phase difference between the IF output signals was measured and is recorded in table 9.1. Trial one and trial two were conducted on different days with the equipment power completely removed in between.

The unpredictable phase difference between the two IF output signals from the TVRX2 tuner board occurs because the local oscillator required for the heterodyne down conversion of

signal frequency from the RF input to the IF output is provided by an on-chip phase locked loop synthesized internal oscillator [31]. There are two TDA18272 tuner ICs on the TVRX2 daughterboard, each with its own internal synthesized local oscillator. Each time the tuner is initialized and each time the tuned frequency is changed, the phases of the two local oscillators will be randomly different leading to a random difference in phase between the RF signal received at the antenna and the IF output signal. Ideally there should be a common local oscillator for the two tuners but this is not possible with the TDA18272 tuners.

Table 9.1: TVRX2 IF output phase difference trials

Trial 1			**Trial 2**		
RF Frequency	**IF phase difference**		**RF Frequency**	**IF phase difference**	
[MHz]	[ns]	[degrees]	[MHz]	[ns]	[degrees]
100	-34,4	-61,92	100	90,0	162,00
120	-44,0	-79,20	120	89,0	160,20
140	6,4	11,52	140	20,8	37,44
160	34,4	61,92	160	73,0	131,40
180	-82,4	-148,32	180	69,0	124,20
200	-21,6	-38,88	200	76,0	136,80
220	-94,4	-169,92	220	40,8	73,44
240	62,4	112,32	240	27,2	48,96
260	-58,4	-105,12	260	-85,6	-154,08
280	74,0	133,20	280	11,2	20,16
300	-99,2	-178,56	300	56,0	100,80
320	99,0	178,20	320	-21,6	-38,88
340	-28,0	-50,40	340	18,4	33,12
360	-85,6	-154,08	360	2,4	4,32
380	-52,8	-95,04	380	-13,6	-24,48
400	92,0	165,60	400	82,0	147,60
420	-48,8	-87,84	420	-8,8	-15,84
440	-32,0	-57,60	440	50,4	90,72
460	-13,6	-24,48	460	31,2	56,16
480	70,0	126,00	480	-84,8	-152,64
500	-84,0	-151,20	500	39,2	70,56
520	42,4	76,32	520	80,0	144,00
540	-96,8	-174,24	540	13,6	24,48
560	-11,2	-20,16	560	4,0	7,20
580	20,0	36,00	580	50,4	90,72
600	-4,8	-8,64	600	2,4	4,32
620	-11,2	-20,16	620	-91,2	-164,16
640	-82,4	-148,32	640	35,2	63,36
660	7,2	12,96	660	69,0	124,2
680	-28,0	-50,40	680	-80,0	-144,00

9.8 Discussion of results

The angular sensitivity of the two element correlating radio telescope interferometer, like all telescopes, is proportional to the ratio between the wavelength of the received EM wave and the receiving aperture (in this case the baseline distance d), as can be seen from equation 9.7. The two metre baseline used for testing and calibrating the completed radio telescope interferometer is very short. The potential for improved angular resolution compared to some single element radio telescope will be achieved at baseline distances much greater than the aperture of the single element radio telescope.

The quantities required to obtain samples of the visibility function, as described in section 3.8, are the amplitude and phase of the fringe function versus baseline distance. Both of these measurements require equipment calibration to overcome uncertainty and changes in channel gain and instrumental phase offset. A suggested calibration technique would be to inject the same calibrated broadband noise source into both receiving channels. The reason for choosing a broadband noise source as opposed to a sine wave is that calibration could be performed at any frequency within the tuning range of the receivers. The next step would be to select channel X and Y in turn, as discussed in section 7.4.2 and adjust the IF stage gain until the flat line plot seen on Radio SkyPipe is at a predetermined reference level. The final calibration step would be to select correlate by pressing “C” in Hyper Terminal and adjust the FPGA channel phase offset until the Radio SkyPipe plot is as close to a predetermined maximum as possible. The correlator output should be at a maximum because the same noise source is connected to both channels and the two sources are therefore coherent and in phase.

Care must be taken to prevent the RF stage AGC from adjusting the RF stage gain during a set of measurements. The simplest technique to achieve this would be to ensure that the received signal strength is very low resulting in the RF stage AGC always being at maximum RF gain. From the results obtained above and the estimates of the received signal strength given by the free field loss calculations in equations 9.2 and 9.3, it is recommended that the maximum received signal strength at the RF inputs to the USRP2 should be less than -50 dBm in order to avoid AGC gain changes during a fringe measurement or between calibration and fringe measurement.

The unpredictable phase difference introduced by the TDA18272 silicon tuners can be minimised by the calibration technique described above. In order to get better phase offset adjustment resolution, the sample rate should be increased. The disadvantage of increasing the sample rate though, is that the integration time will be reduced as discussed in section 7.3.6. In any case, after the best adjustment possible, the remaining phase offset can be measured using the difference between the expected and the actual Radio SkyPipe plot level for the correlation of the injected calibration noise source. The difference in correlation level seen on the Radio SkyPipe plot will be a measure of the remaining phase difference between the IF output signals.

Chapter 10

Conclusion and recommendations

10.1 Introduction

A two element correlating radio telescope interferometer has been developed using a platform that is suitable for practical entry level experimentation and investigation of radio telescope arrays. This qualitative research project has focussed on the technological aspects of developing a working system. The working system produced, is intended to provide a platform for further quantitative research.

10.2 Signal to noise ratio

The two metre baseline test jig and measurement results were used to produce credible fringe results for a single frequency point source radiator. It was found that physical movement of objects in the space between the source radiator and the baseline pair receiver was a significant source of measurement noise. It is recommended that experimentation be done with longer integration times to establish how this may overcome some of the unwanted measurement noise without impairing the wanted measurement. The current integration time of 81,92 µs could be increased in multiples by further averaging in the microcontroller on the interface board as discussed in section 7.3.6. The FPGA based integration time could then also be decreased by increasing the FPGA sample rate. This would improve the resolution of the phase offset adjustment as discussed in section 9.8. Various antenna types and gains should be tested to establish the effectiveness of the correlation process in overcoming internal system noise as well as unwanted signals received in antenna side lobes and to experiment with system gain calibration procedures.

10.3 Angular resolution

The two metre baseline test jig provided a means to verify the proper functioning of the two element correlating radio telescope interferometer produced. This baseline distance is however very short and the single frequency point source is limited in usefulness to the verification and calibration of the system. Real cosmic sources should be measured using a pair of suitably high gain dish antennas. The simplest way to obtain a number of different baseline measurements is to construct an array of several dishes. The spacing between the dishes should be chosen to give the highest number of different baseline distances when selecting different pairs within the group [11]. Three dishes can provide three different baseline distances while four dishes can provide six different baseline distances, for dishes arranged in a straight line from East to West. The East to West straight line arrangement is recommended to preserve the simplicity of the transit correlator that has been developed. Figure 10.1 shows example spacing for four antennas and the different baseline distances that could be obtained.

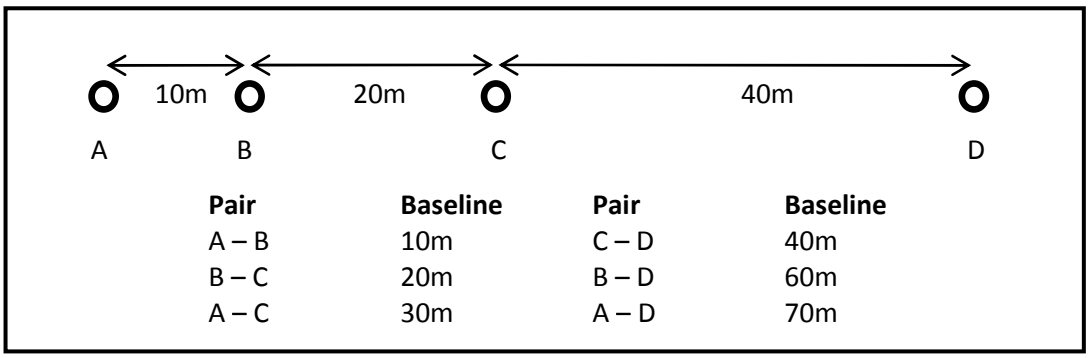


Figure 10.1: Suggested antenna pair spacing for a straight line array configuration of four elements.

10.4 Gain settings

The IF gain setting increment should be reduced to improve gain calibration resolution, as discussed in section 9.4. The system should be calibrated by switching the RF inputs to a

calibrated noise source as discussed in section 9.8. For weak cosmic sources, suitable high gain antennas will be required with low noise amplifiers near the antennas. Care must be taken not to have the signal level input at the USRP2 too high as this may result in AGC gain changes during a measurement. This is discussed in section 9.8. The calibration of system gain should be repeated regularly to compensate for system gain changes with temperature and time. Measurements and analysis of these system gain changes should be undertaken to establish the best calibration repetition interval and procedure.

10.5 Instrumental phase offset

The unpredictable random phase offset that occurs in the TDA18272 silicon tuner IC poses a calibration problem. This phase offset will need to be determined and compensated for, each time the equipment is initialized or an RF frequency change is made. The calibration procedure described in section 9.8 could be used to solve this. An increase in FPGA sample rate will improve the resolution of the FPGA phase offset correction and the remaining phase offset could be measured as described in section 9.8. An increase in FPGA sample rate will result in a reduction in the FPGA integration time but this could be overcome by increasing the integration time using microcontroller firmware as described in section 7.3.6. There are other factors contributing to instrumental phase offset such as differences in RF cable lengths and dielectric constant. The methods used to overcome the random phase offset in the TDA18272 silicon tuner ICs would at the same time compensate for other instrumental phase offset. Measurement and analysis of instrumental phase offset should be undertaken and compensation methods investigated for the different system configurations discussed in section 10.3.

10.6 Summing up

A working two element correlating radio telescope interferometer has been developed for entry level experimental work. The correlator and other useful calibration features have been implemented using DSP on an FPGA as well as microcontroller firmware. The proper functioning of the complete end-to-end system has been verified and some test measurements of fringe functions for a single frequency radiating source were conducted. The results have been analysed and the potential for further quantitative and qualitative research using the equipment developed, including further development of the system, have been identified. The research project objective, to develop a working two element correlating radio telescope interferometer that will provide a platform for further qualitative and quantitative research, has been achieved.

References

[1] K. Rohlfs , T. L. Wilson , *Tools of Radio Astronomy*, 4th ed., London, Springer, 2006.

[2] M. Zeilik, *Astronomy : the Evolving Universe*, 9th ed., Cambridge, UK, Cambridge University Press, 2002.

[3] D. F. Miller, *Basics of Radio Astronomy for the Goldstone-Apple Valley Radio Telescope*, Jet Propulsion Laboratory, California Institute of Technology, USA, 1998. [Online]. Available: http://www.jpl.nasa.gov/radioastronomy

[4] D. J. Griffiths, *Introduction to Electrodynamics*, 3rd ed., Upper Saddle River, New Jersey, USA, Prentice Hall, 1999.

[5] F. A. Jenkins, H. E. White, *Fundamentals of Optics*, 4th ed., New York, McGraw-Hill, 1976.

[6] F. W. Sears, M. W. Zemansky, H. D. Young, *University Physics*,5th ed., London, Addison Wesley, 1976.

[7] B. F. Burke, F. Graham-Smith, *An Introduction to Radio Astronomy*, 2nd ed., Cambridge, UK, Cambridge University Press, 2002.

[8] M. J. Gaylard, "Practical Radio Astronomy a hi-math introduction," tutorial notes, Hartebeeshoek Radio Astronomy Observatory, South Africa, 2006.

[9] P. H. Young, *Electronic Communication Techniques*, 5th ed., Upper Saddle River, New Jersey, USA, Prentice Hall, 2004.

[10] G. B. Taylor et al., "First Light for the First Station of the Long Wavelength Array," *Journal of Astronomical Instrumentation*, Vol. 1, Issue 1, 2012. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S2251171712500043

[11] E. Keto, "Hierarchical Configurations for Cross-correlation Interferometers with Many Elements," *Journal of Astronomical Instrumentation*, Vol. 1, Issue 1, 2012. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S2251171712500079

[12] A. R. Thompson, J. M. Moran, G. W. Swenson Jr., *Interferometry and Synthesis in Radio Astronomy*, 2nd ed., New York, Wiley-VCH, 2004.

[13] J. Jansen van Rensburg, "The design of a two element correlation interferometer operating at L-band," MSc (Eng.) thesis, Dept. Elect. & Electron. Eng., Stellenbosch University, South Africa, 2012.

[14] A. Glindemann, "Introduction to Spatial Interferometry," tutorial notes, European Southern Observatory, Garching, Germany. [Online]. Available: http://www.eso.org/sci/facilities/paranal/telescopes/vlti/tuto/index.html

[15] A. Parsons et al., “Digital Instrumentation for the Radio Astronomy Community,” *Astro-ph.IM*, arXiv:0904.1181v1, Cornell University Library, 2009. [Online]. Available: http://arxiv.org/abs/0904.1181

[16] A. Parsons et al., “PetaOp/Second FPGA Signal Processing for SETI and Radio Astronomy,” Proc. Asilomar Conference, 2006. [Online]. Available: http://www.eecs.berkeley.edu/papers/asilomar_2006.pdf

[17] N. Jackson, “Principles of Interferometry,” lecture notes, University of Manchester, Jodrell Bank Observatory, UK. [Online]. Available: http://wwwog.arcetri.astro.it/Workshops/elba06/SCI_LEC/Jackson.pdf

[18] A. Parsons et al., “A New Approach to Radio Astronomy Signal Processing: Packet Switched, FPGA-based, Upgradeable, Modular Hardware and Reuseable, Platform-Independent Signal Processing Libraries,” Proc. General Assembly of the International Union of Radio Science, 2005. [Online]. Available: https://casper.berkeley.edu/wiki/Papers

[19] P.L. McMahon, “Adventures in Radio Astronomy Instrumentation and Signal Processing,” MSc (Eng.) thesis, Dept. Elect. Eng., University of Cape Town, South Africa, 2008.

[20] A. Parsons et al., “A Scalable Correlator Architecture Based on Modular FPGA Hardware, Reuseable Gateware, and Data Packetization,” *Astro-ph.IM*, arXiv:0809.2266v3, Cornell University Library, 2009. [Online]. Available: http://arxiv.org/abs/0809.2266

[21] S. Scott, “RHINO: Reconfigurable Hardware Interface for computatioN and radiO,” MSc (Eng.) thesis, Dept. Elect. Eng., University of Cape Town, South Africa, 2011.

[22] R. Stewart, L. Crockett, “Xilinx DSP for FPGA Primer,” tutorial workbook Ver.6.3/27/11, University of Strathclyde and Steepest Ascent Ltd, Scotland, 2011.

[23] *ISE In-Depth Tutorial*, UG695 (v13.1), Xilinx, 2011.

[24] *System Generator for DSP User Guide*, UG639 (v13.4), Xilinx, 2012.

[25] *PIC16F631/677/685/687/689/690 Data Sheet*, DS41262E, Microchip, 2008.

[26] *PIC16F882/883/884/886/887 Data Sheet*, DS41291F, Microchip, 2009.

[27] *Spartan3 Generation FPGA User Guide*, UG331 (v1.7), Xilinx, 2010.

[28] *Virtex-5 FPGA User Guide*, UG190 (v5.3), Xilinx, 2010.

[29] *HI-TECH C® for PIC10/12/16 User’s Guide*, DS51865B, Microchip, 2010.

[30] *MAX187/MAX189*, data sheet 19-0196, Rev 0, Maxim, 1993.

[31] *TDA18272 Product data sheet*, Rev 2, NXP, 2010.

[32] *ML505, ML506, ML507 Evaluation Platform User Guide*, UG347 (v3.1.1), Xilinx, 2009.

[33] M. M. da Silva, *Multimedia Communications and Networking*, Boca Raton, Florida, USA, CRC Press, 2012.

ANNEXURE A

Interface daughterboard PIC microcontroller C code list

```
#include<htc.h>
#include<string.h>

/* SKYPIPE INTERFACE CODE

    C code for Microchip PIC16F883 microcontroller
    8 MHz internal RC Oscillator
    Interfaces USRP2 on-board XILINX SPARTAN 3 FPGA to HyperTerminal and SkyPipe on a PC
    USRP2 is Ettus Research -  Universal Software-defined-radio Peripheral V2
    HyperTerminal is an ASCII terminal emulator PC app
    SkyPipe is a 12 bit data chart recorder PC app for Radio Astronomy

    Written by Dave Callaghan
    Durban University of Technology
    Ver 1.1   (ver 1.0 is assembly language test code)
    3rd September 2012
*/

// CONFIGURATION BIT SETTINGS
//    MCLR internal, PWRON reset enabled, WDT Off , Internal Osc 8 MHz, CP on, Brown out on

__CONFIG(4);
__CONFIG(0);

// DATA STORED IN THE 256 BYTE EEPROM

__EEPROM_DATA(0x0A,0x06,0x36,0x24,0x2E,0x0E,0x11,0x0A);          // tuner setup register addresses
__EEPROM_DATA(0x19,0x1A,0x0C,0x14,0x14,0x06,0x06,0x14);
__EEPROM_DATA(0x15,0x12,0x13,0x23,0x0C,0x0D,0x0E,0x1B);
__EEPROM_DATA(0x0F,0x10,0x11,0x06,0x14,0x0A,0x16,0x17);
__EEPROM_DATA(0x18,0x19,0x1A,0,0,0,0,0);

__EEPROM_DATA(0x9F,0x06,0x0C,0x49,0x40,0xFF,0x4A,0x9F);          // tuner setup register data
__EEPROM_DATA(0x3B,0x01,0x09,0x03,0x43,0x06,0x00,0x43);
__EEPROM_DATA(0x64,0x00,0x03,0x03,0x00,0x0F,0x21,0x60);
__EEPROM_DATA(0x01,0x01,0x01,0x00,0x43,0x9F,0x01,0x86);
__EEPROM_DATA(0xA0,0x41,0x01,0,0,0,0,0);

__EEPROM_DATA('R','a','d','i','o',' ','S','k');       // text messages (starting at address 80D)
__EEPROM_DATA('y','P','i','p','e',' ','I','n');       // (88)
__EEPROM_DATA('t','e','r','f','a','c','e',' ');       // (96)
__EEPROM_DATA('V','1','.','1',0,'C','H','A');         // (104)
__EEPROM_DATA('N','N','E','L',' ',0,' ','=');         // (112)
__EEPROM_DATA(' ',0,'T','E','S','T',' ',0);               // (120)
__EEPROM_DATA('E','X','I','T',0,'O','U','T');         // (128)
__EEPROM_DATA('P','U','T',' ','I','S',' ',0);         // (136)
__EEPROM_DATA('C','O','U','N','T',0,' ','U');             // (144)
__EEPROM_DATA('P',0,' ','D','O','W','N',0);               // (152)
__EEPROM_DATA(' ','M','A','X','I','M','U','M');           // (160)
__EEPROM_DATA(0,' ','H','O','L','D',0,'C');               // (168)
__EEPROM_DATA('O','R','R','E','L','A','T','E');           // (176)
__EEPROM_DATA(0,' ','Z','E','R','O',0,'S');               // (184)
__EEPROM_DATA('O','U','R','C','E',0,'G','A');             // (192)
__EEPROM_DATA('I','N',0,'P','H','A','S','E');         // (200)
__EEPROM_DATA(0,'F','R','E','Q','U','E','N');             // (208)
__EEPROM_DATA('C','Y',0,'B','A','N','D','W');             // (216)
__EEPROM_DATA('I','D','T','H',0,'V','I','E');         // (224)
__EEPROM_DATA('W',' ','S','E','T','T','I','N');           // (232)
__EEPROM_DATA('G','S','E','R','R','O','R',0);             // (240)
__EEPROM_DATA('O','K',0,' ','M','H','z',0);               // (248)

// CONSTANTS
// text message addresses
#define skypipe 80
#define channel  109
#define equals 118
#define test 122
#define exitm 128
#define output 133
#define countm 144
#define up 150
#define down 154
#define max 160
#define holdm 169
```

```
75  #define corr 175
76  #define zero 185
77  #define sourcem 191
78  #define gainm 198
79  #define phasem 203
80  #define freqm 209
81  #define bwm 219
82  #define view 229
83  #define error 242
84  #define ok 248
85  #define MHz 251
86
87  // offsets for storing settings in array called settings
88
89  #define gainX 0
90  #define phaseX 3
91  #define bwX 6
92  #define freqXY 9
93  #define gainY 12
94  #define phaseY 15
95  #define bwY 18
96
97  // ASCII control character codes
98
99  #define LF 10
100 #define FF 12
101 #define CR 13
102
103 // IIC addresses for communicating with tuners on the TVRX2 daughterboard
104
105 #define tuner1 0xC6
106 #define tuner2 0xC0
107
108 // IO pins connected to Radio SkyPipe communication cable
109
110 #define CS RB7              // Enable input from SkyPipe
111 #define SCLK RB6            // sync clock input from SkyPipe
112 #define DOUT RC0            // Serial data output to SkyPipe
113
114 // IO pin connected to status LED on interface board
115
116 #define LED RB2
117
118 // IO pins for select (S2 S1 S0) and serial data (SD1 SD0) connections to FPGA
119
120 #define S0 RB3
121 #define S1 RB4
122 #define S2 RB5
123 #define SD0 RC1
124 #define SD1 RC2
125
126 // IO pins for IIC comms with tuners on TVRX2 tuner daughterboard
127
128 #define SCL RC1
129 #define SDA RC2
130 #define SDAin RC4     // bidirectional IO for IIC not supported by System Generator FPGA design tool
131
132 // IO pins for control signal inputs from tuners on TVRX2 daughterboard
133
134 #define IRQ RC5
135 #define VSYNC RC3
136
137 // C FUNCTION DECLARATIONS
138
139 void init() ;
140 void typem() ;
141 void typeCHR(unsigned char CHR) ;
142 void typeNUM(unsigned char setdigit) ;
143 void newline() ;
144 void newpage() ;
145 void printhmenu() ;
146 void printsmenu() ;
147 void printtmenu() ;
148 void viewsettings() ;
```

```
149  void selectXY() ;
150  void selectCorr() ;
151  void selectTest() ;
152  void selectIP();
153  void setup(unsigned char pmsg) ;
154  void countup() ;
155  void countdown() ;
156  void counthold() ;
157  void countzero() ;
158  void countmax() ;
159  void printmsg(unsigned char pmsg, unsigned char newl) ;
160  void getnum() ;
161  void delay1() ;
162  void delay2() ;
163  void delay3() ;
164  void delay4() ;
165  void setgain() ;
166  void setfreq() ;
167  void setphase() ;
168  void setbw() ;
169  unsigned char BCD (unsigned char setptr);
170  unsigned short long BCD3 (unsigned char setptr);
171  void settuner() ;
172  void sendbyte() ;
173  void sendIIC(unsigned char bytes) ;
174  void IICbyte() ;
175  void timeout() ;
176  void timeoutm();
177
178  // GLOBAL VARIABLE DECLARATIONS
179
180  unsigned char tcnt,temp,temp1,new,number,key,source,setting,flashcnt,flashrate,msg[22];
181  unsigned char digit,device,sbyte,sdata,sptr,freq1,freq2,freq3,settings[21] ;
182  bit new1,new2,typech,upcnt,hold,num,rdfpga,tuneron,tout,toutf,gainf,phasef,freqf,bwf ;
183  unsigned int count,counts,dummy ;
184
185  // INTERRUPT SERVICE ROUTINE FOR COMMUNICATION WITH RADIO SKYPIPE
186  /*   Interrupt is triggered by high to low on CS line. Skypipe provides clock on SCLK line. PIC sends serial
187       data on DOUT line. Data is pipelined to minimize interrupt latency. SkyPipe expects to be communication
188       with hardware ADC IC MAX187 so very little delay +-5us between enable and data clocking.
189       Data sent to SkyPipe is data read from fpga on the previous interrupt. First bit of next data is ready on
190       serial data line after each interrupt. Next interrupt starts by sending the second bit of data.
191  */
192
193  void interrupt handler (void)
194
195  {
196       GIE = 0;                                  // disable interrupts while transferring data
197       while (!SCLK)                             // wait for first SCLK pulse - ADC conversion delay
198       {
199            if (RCIF) break;                     // break out if command received from Hyper Terminal
200       }
201       while (SCLK)                              // wait for falling edge of first SCLK pulse
202       {
203       }
204       while (!CS && !RCIF)                      // break out if enable signal from SkyPipe goes high
205       {                                         // or command received from Hyper Terminal
206            count <<=1;                          // rotate data for SkyPipe left one bit - send second bit of data
207            while (!SCLK && !CS)                 // wait for rising edge of SCLK
208            {
209            }
210            if (count & 0x8000) DOUT = 0;        // send next bit of data - DOUT is inverted by a driver transistor
211            else DOUT = 1;
212            while (SCLK)                         // wait for falling edge of SCLK
213            {
214            }
215       }                                         // end data transfer of data stored during previous interrupt
216
217       if (!rdfpga)                              // this ISR either reads data from FPGA or generates test data
218       {                                         // generate test data - up or down count or hold count
219            if (!hold)
220            {
221                 if (upcnt) ++counts;            // prepare test data for next interrupt
222                 else --counts;
```

```
223             }
224         }
225         else                                    // read correlator data from FPGA and store it for the next interrupt
226         {
227             SD1 = 1;                            // sends sample data pulse to fpga which latches correlator data
228             delay4();                           // short delay
229             SD1 = 0;
230             delay4();
231             counts = PORTA;                     // read low byte of correlator data into 16 bit variable (counts)
232             SD0 = 1;                            // sends select high byte signal to fpga
233             delay4();
234             count = PORTA;                      // read high byte of correlator data into 16 bit variable (count)
235             SD0 = 0;                            // restore HILO signal for next read
236             count <<=8;                         // shift high byte data to upper 8 bits of 16 bit variable (count)
237             counts = counts | count;            // combine low byte and high byte data
238
239         }
240         count = counts;                         // transfer data to count
241         count <<=4;                             // shift 4 bits left because SkyPipe takes only 12 bits
242         if (count & 0x8000) DOUT = 0;           // set serial data output pin to the first bit of the next data interrupt
243         else DOUT = 1;
244         ++flashcnt;                             // flash status LED on interface board to show data is being tranferred
245         if (flashcnt & flashrate) LED = 1;
246         else LED = 0;
247         RBIF = 0;
248     }
249
250
251     main(void)                                  // main routine -  top level Hyper Terminal command loop
252     {
253         init();                                 // initialization
254         for(;;)
255         {
256             if (RCIF)                           // command key pressed in Hyper Terminal
257             {
258                 key = RCREG ;                   // fetch ASCII character code from RS232 receive register
259                 if (key >= '0' && key <= '9') num=1;  // set num flag if key pressed is numeric
260                 else num = 0;
261                 if (!num) number = 0;           // number of digits in number must be set to zero if not a number entry
262                 switch (key)
263                 {
264                     case 'I':                   // I - Initialize tuners
265                     if (!GIE) settuner();
266                     else printmsg(error,3);
267                     break;
268                     case 'S':                   // S - Source selection menu request
269                     printsmenu();
270                     break;
271                     case 'T':                   // T - Test data menu request
272                     rdfpga = 0;
273                     printtmenu();
274                     break;
275                     case 'E':                   // E - Exit fpga data aquisition
276                     rdfpga = 0;
277                     temp = source;
278                     source = 'C';
279                     selectIP();
280                     source = temp;
281                     GIE = 0;
282                     LED = 1;
283                     printhmenu();               // print header menu
284                     break ;
285                     case 'U':
286                     countup();                  // generate up counting test data
287                     break;
288                     case 'D':                   // generate down counting test data
289                     countdown();
290                     break;
291                     case 'H':                   // generate constant value test data (hold counting)
292                     counthold();
293                     break;
294                     case 'X':                   // select source X - output is X times X so mean square power on ch X
295                     source = key;
296                     selectXY();
```

```
297                     selectIP();
298                     break;
299                     case 'Y':                   // select source Y - output is Y times Y so mean square power on ch Y
300                     source = key;
301                     selectXY();
302                     selectIP();
303                     break;
304                     case 'C':                       // select source X and Y - output is X times Y correlation
305                     source = key;
306                     selectCorr();
307                     selectIP();
308                     break;
309                     case 'Z':                   // zero the test data counter
310                     countzero();
311                     break;
312                     case 'M':                       // set test data counter to maximum value
313                     countmax();
314                     break;
315                     case 'G':                       // display message for numeric entry for gain
316                     setup(gainm);
317                     break;
318                     case 'P':                   // display message for numeric entry for phase correction
319                     setup(phasem);
320                     break;
321                     case 'F':                   // display message for numeric entry for tuner frequency selection
322                     setup(freqm);
323                     break;
324                     case 'B':                   // display message for numeric entry for tuner IF bandwidth
325                     setup(bwm);
326                     break;
327                     case 'V':                   // display the settings for gain, phase, frequency and bandwidth
328                     viewsettings();
329                     break;
330                     default:
331                     if (num) getnum();              // if numeric key pressed then get the number
332                     else printmsg(error,3);         // key pressed is none of the above menu options so display error message
333                     break;
334                 }
335             }
336         }
337 }
338
339 void init(void)
340 {
341     unsigned char i;
342     IRCF0 = 1;                                  // set internal RC osc frequency select bit 0 to select 8 MHz internal osc
343     TRISB = 0XC1;                               // RB1(RTS),RB2(LED),RB3(S0),RB4(S1),RB5(S2) output. Rest innput
344     TRISC = 0XF0;                               // RC0(DOUT),RC1(SD0),RC2(SD1),RC3(VSYNC) output. Rest input
345     ANSEL = 0;                                  // set all PIC IO to digital
346     ANSELH = 0;
347     PORTB = 0;
348     PORTC = 0;
349     SPBRG = 12;                                 // set RS232 Baud rate to 9600 bits/s
350     TXEN = 1;                                   // enable transmit on USART
351     SPEN = 1;                                   // enable USART
352     CREN = 1;                                   // enable receive on USART
353     IOCB7 = 1;
354     RBIF = 0;                                   // clear portb interrupt flag
355     RBIE = 1;                                   // enable interrupts from CS signal (RB7) on SkyPipe cable
356     LED = 1;                                    // flash status LED on interface board 3 times to confirm software OK
357     delay1();
358     LED = 0;
359     printhmenu();                               // display header menu on Hyper Terminal
360     LED = 1;
361     delay1();
362     LED = 0;
363     delay1();
364     LED = 1;
365     delay1();
366     LED = 0;
367     delay1();
368     LED = 1;
369     DOUT = 1;                                   // start serial o/p data to SkyPipe in low state (transistor inverts)
370     S0 = 0;                                     // (S2 S1 S0 : 1 1 0) select O/P fpga data on fpga uC interface
```

```
371         S1 = 1;
372         S2 = 1;
373         SD0 = 0;
374         SD1 = 0;
375         freq1 = 0x01;                       // set initial frequency setting to 100 MHz - see tuner datasheet
376         freq2 = 0x86;                       // 24 bit binary number in kHz - 100 000 kHz
377         freq3 = 0xA0;
378         rdfpga = 0;
379         count = 0;
380         counts = 0;
381         flashcnt = 0;
382         flashrate = 0;
383         number=0;
384         num=0;
385         new1 = 0;                           // flags for newline before and after displaying a message on Hyper Terminal
386         new2 = 1;
387         for (i=0 ; i<21; i++) settings[i] = '0' ;       // clear settings array
388         settings[freqXY] = '1';             // set initial frequency to 100 MHz
389         settings[gainX+1] = '8';            // set initial gain = 080
390         settings[gainY+1] = '8';
391         source = 'X';                       // set initial fpga signal source to X times X
392     }
393
394     void printmsg(unsigned char pmsg, unsigned char newl)
395     {                                       // display the message at EEPROM addr (pmsg) on Hyper Terminal
396     unsigned char nchar;
397
398     if (newl & 1) newline();
399     if (newl & 4) newpage();
400     typech = 1;
401     while (typech)
402         {
403         nchar = eeprom_read(pmsg);
404         if (nchar == 0)                     // a zero indicates end of message string
405             {
406             typech = 0;
407             break;
408             }
409         while (!TXIF)                       // wait for TX buffer clear flag
410             {
411             }
412         TXREG = nchar;
413         ++pmsg;
414         }
415     if (newl & 2) newline();
416     if (newl & 8) newline();
417     }
418
419     void typem(void)
420     {                                       // display a message stored in array (msg)
421         unsigned char i;
422         if (new1) newline();
423         for (i=0 ; i<22; i++)
424         {
425             if (msg[i] == 0) break;
426             while (!TXIF)                   // wait for TX buffer clear flag
427             {
428             }
429             TXREG = msg[i];
430         }
431         if (new2) newline();
432         new1 = 0;
433         new2 = 1;
434     }
435
436     void newpage(void)                      // clear Hyper Terminal screen
437     {
438         while (!TXIF)                       // wait for TX buffer clear flag
439         {
440         }
441         TXREG = FF ;
442     }
443
444     void newline(void)                      // start new line on Hyper Terminal
```

```
445 {
446     while (!TXIF)                       // wait for TX buffer clear flag
447     {
448     }
449     TXREG = CR ;
450     while (!TXIF)                       // wait for TX buffer clear flag
451     {
452     }
453     TXREG = LF ;
454 }
455
456 void typeCHR(unsigned char CHR)         // display a single character on Hyper Terminal
457 {
458     while (!TXIF)                       // wait for TX buffer clear flag
459     {
460     }
461     TXREG = CHR ;
462 }
463
464 void typeNUM(unsigned char setdigit)    // display a 3 digit number stored in settings array
465 {
466     unsigned char i;
467     for (i = 3 ; i--;)
468     {
469         while (!TXIF)                   // wait for TX buffer clear flag
470         {
471         }
472         TXREG = settings[setdigit] ;
473         ++setdigit ;
474     }
475 }
476
477 void printsmenu(void)                   // display source select menu on Hyper Terminal
478 {
479 //    newpage();
480     printmsg(skypipe,14);
481     temp = 'X';
482     typeCHR(temp);
483     printmsg(equals,0);
484     printmsg(channel,0);
485     typeCHR(temp);
486     newline();
487     temp = 'Y';
488     typeCHR(temp);
489     printmsg(equals,0);
490     printmsg(channel,0);
491     typeCHR(temp);
492     newline();
493     typeCHR('V');
494     printmsg(equals,0);
495     printmsg(view,2);
496     typeCHR('C');
497     printmsg(equals,0);
498     printmsg(corr,2);
499     typeCHR('T');
500     printmsg(equals,0);
501     printmsg(test,0);
502     printmsg(countm,2);
503     typeCHR('E');
504     printmsg(equals,0);
505     printmsg(exitm,2);
506     switch (source)
507     {
508         case 'X':
509         selectXY();                     // display msg "OUTPUT IS CHANNEL X"
510         break;
511         case 'Y':
512         selectXY();                     // display msg "OUTPUT IS CHANNEL Y"
513         break;
514         case 'C':
515         selectCorr();                   // display msg "OUTPUT IS CORRELATE"
516         break;
517         case 'T':
518         selectTest();                   // display msg "OUTPUT IS TEST"
```

```
519             break;
520         }
521 }
522
523
524 void printtmenu(void)                       // display test data menu on Hyper Terminal
525 {
526 //    newpage();
527       printmsg(skypipe,14);
528       source = key ;
529       newline();
530       typeCHR('U') ;
531       printmsg(equals,0);
532       printmsg(countm,0);
533       printmsg(up,2);
534       typeCHR('D') ;
535       printmsg(equals,0);
536       printmsg(countm,0);
537       printmsg(down,2);
538       typeCHR('H') ;
539       printmsg(equals,0);
540       printmsg(countm,0);
541       printmsg(holdm,2);
542       typeCHR('M') ;
543       printmsg(equals,0);
544       printmsg(countm,0);
545       printmsg(max,2);
546       typeCHR('Z') ;
547       printmsg(equals,0);
548       printmsg(countm,0);
549       printmsg(zero,2);
550       newline();
551 }
552
553 void printhmenu (void)                      // display header menu on Hyper Terminal
554 {
555 //    newpage();
556       printmsg(skypipe,14);
557       typeCHR('S');
558       printmsg(equals,0);
559       printmsg(sourcem,2);
560       typeCHR('G');
561       printmsg(equals,0);
562       printmsg(gainm,2);
563       typeCHR('P');
564       printmsg(equals,0);
565       printmsg(phasem,2);
566       typeCHR('F');
567       printmsg(equals,0);
568       printmsg(freqm,2);
569       typeCHR('B');
570       printmsg(equals,0);
571       printmsg(bwm,2);
572       typeCHR('E');
573       printmsg(equals,0);
574       printmsg(exitm,2);
575       newline();
576 }
577
578
579 void viewsettings (void)                    // display frequency, gain, phase and BW settings on Hyper Terminal
580 {
581 unsigned char i,j;
582 temp = source;
583 source = 'X';
584 j = 0;
585 setup(freqm);
586 typeNUM(freqXY);
587 printmsg(MHz,2);
588 for (i = 2; i--;)
589       {
590       setup(gainm);
591       typeNUM(gainX+j);
592       setup(phasem);
```

```
593         typeNUM(phaseX+j);
594         setup(bwm);
595         typeNUM(bwX+j);
596         newline();
597         ++source;
598         j = 12;
599         }
600   source = temp;
601   number = 0;
602   }
603
604   void selectXY (void)                    // display msg "OUTPUT IS CHANNEL X or Y" on Hyper Terminal
605   {
606   printmsg(output,1);
607   printmsg(channel,0);
608   typeCHR(source);
609   newline();
610   rdfpga = 1;
611   }
612
613   void selectCorr (void)                      // display msg "OUTPUT IS CORRELATE" on Hyper Terminal
614   {
615   printmsg(output,1);
616   printmsg(corr,2);
617   rdfpga = 1;
618   }
619
620   void selectTest (void)                      // display msg "OUTPUT IS TEST" on Hyper Terminal
621   {
622   new1 = 1;
623   printmsg(output,0);
624   printmsg(test,2);
625   }
626
627
628   void selectIP (void)                        // set select inputs on fpga uC interface for selected data source
629   {
630   SD0 = 0;
631   SD1 = 0;
632   S2 = 1;
633   S1 = 0;
634   S0 = 0;
635   if (source == 'C')                  // data source is X times Y - correlate
636   {
637         S1 = 1;
638         S0 = 1;
639   if (rdfpga)     SD1 = 1;
640   }
641   else
642   {
643         if (source == 'X') SD1 = 0;         // data source is X times X
644         if (source == 'Y') SD1 = 1;         // data source is Y times Y
645   }
646   delay4();
647   SD0 = 1;
648   delay4();
649   SD0 = 0;
650   delay4();
651   SD1 = 0;
652   S1 = 1;
653   S0 = 0;
654   flashrate = 0x10;
655   GIE = 1;
656   }
657
658   void setup(unsigned char pmsg)      // display a prompt message on Hyper Terminal for settings entry
659   {
660   setting = key;
661   if (pmsg != freqm)
662         {
663         printmsg(channel,1);
664         typeCHR(source);
665         typeCHR(' ');
666         printmsg(pmsg,0);
```

```
667         }
668     else printmsg(pmsg,1);
669     printmsg(equals,0);
670     number =3;                              // 3 digit numeric entry to follow
671     }
672
673     void countup (void)                     // set data source to up count test data generated in PIC uC
674     {
675         upcnt = 1;
676         hold = 0;
677         source = 'T';
678         flashrate = 0x20;
679         new1 = 1;
680         printmsg(output,1);
681         printmsg(countm,0);
682         printmsg(up,2);
683         rdfpga = 0;
684         GIE = 1;
685     }
686
687     void countdown (void)                   // set data source to down count test data generated in PIC uC
688     {
689         upcnt = 0;
690         hold = 0;
691         source = 'T';
692         flashrate = 0x40;
693         new1 = 1;
694         printmsg(output,1);
695         printmsg(countm,0);
696         printmsg(down,2);
697         rdfpga = 0;
698         GIE = 1;
699     }
700
701     void counthold (void)                   // set data source to hold count test data generated in PIC uC
702     {
703         hold = 1;
704         flashrate = 0x80;
705         source = 'T';
706         new1 = 1;
707         printmsg(output,1);
708         printmsg(countm,0);
709         printmsg(holdm,2);
710         rdfpga = 0;
711         GIE = 1;
712     }
713
714     void countzero (void)                   // zero the test data counter
715     {
716         counts = 0;
717         count = 0;
718         source = 'T';
719         new1 = 1;
720         printmsg(output,0);
721         printmsg(countm,0);
722         printmsg(zero,2);
723     }
724
725     void countmax (void)                    // set the test data counter to maximum value
726     {
727         counts = 0x0fff;
728         count = 0x0fff;
729         source = 'T';
730         new1 = 1;
731         printmsg(output,0);
732         printmsg(countm,0);
733         printmsg(max,2);
734     }
735
736
737     void getnum (void)                      // read a 3 digit setting typed at Hyper Terminal and store
738     {
739         if (!number) printmsg(error,3);
740         else
```

```
741         {
742             if (number==3)
743             {
744                 switch (setting)                // set index (digit) to correct place in settings array to store number
745                 {
746                     case 'G':
747                     if (source == 'X') digit = gainX;
748                     else digit = gainY ;
749                     gainf = 1;
750                     break;
751                     case 'P':
752                     if (source == 'X') digit = phaseX;
753                     else digit = phaseY;
754                     phasef = 1;
755                     break;
756                     case 'F':
757                     digit = freqXY;
758                     freqf = 1;
759                     break;
760                     case 'B':
761                     if (source == 'X') digit = bwX ;
762                     else digit = bwY ;
763                     bwf = 1;
764                     break;
765                 }
766             }
767             settings[digit] = key;
768             ++digit;
769             --number;
770             if (!number)                        // after third digit typed set the requested parameter
771             {
772                 if (gainf) setgain();
773                 if (freqf) setfreq();
774                 if (phasef) setphase();
775                 if (bwf) setbw();
776             printmsg(ok,3);                     // display ok message on Hyper Terminal to confirm setting
777 
778             }
779         }
780 }
781 
782 
783 void settuner (void)                            // initial setup of tuners
784 {
785 if (tuneron) printmsg(error,3);
786 else
787         {
788         temp = source;
789         source = 'X';                           // setup initial gain settings on tuner IF stages
790         setgain();
791         source = 'Y';
792         setgain();
793         source = temp;
794         SD0 = 0;
795         SD1 = 0;
796         S0 = 1;                                 // (S2 S1 S0 : 0 0 1) select inputs for fpga uC interface to setup tuner
797         S1 = 0;
798         S2 = 0;
799         new1 = 1;
800         strcpy(msg,"INITIALIZING TUNERS") ;     // display "INITIALIZING TUNERS" on Hyper Terminal
801         typem();
802 
803         sptr = 0;                               // point to first byte of tuner addr and data in EEPROM (0A:9F) - clear IRQ2
804         device = tuner2;
805         sendIIC(1);                             // send one byte from EEPROM to tuner 2
806 
807         device = tuner1;
808         sendIIC(9);                             // send next 9 bytes from EEPROM to tuner 1
809 
810         tout = 1;
811         while (!IRQ)                            // wait for IRQ signal from tuner 1
812         {
813         timeout();                              // timeout if IRQ signal not received
814         if (toutf)
```

```
815             {
816             timeoutm();                     // display timeout msg on Hyper Terminal
817             break;
818             }
819         }
820
821         sendIIC(4);                         // send next 4 bytes from EEPROM to tuner 1
822         delay3();
823         sendIIC(13);                        // send next 13 bytes from EEPROM to tuner 1
824         delay3();
825         sendIIC(8);                         // send next 8 bytes from EEPROM to tuner 1
826
827         tout = 1;
828         while (!IRQ)                    // wait for IRQ signal from tuner 1
829         {
830         timeout();                          // timeout if IRQ signal not received
831         if (toutf)
832             {
833             timeoutm();                     // display timeout msg on Hyper Terminal
834             break;
835             }
836         }
837
838         sptr = 0;                       // point to first byte of tuner addr and data in EEPROM (0A:9F) - clear IRQ1
839         sendIIC(1);                         // send one byte from EEPROM to tuner 1
840         delay3();
841         device = tuner2;
842         sendIIC(9);                         // send next 9 bytes from EEPROM to tuner 2
843
844         tout = 1;
845         while (!IRQ)                    // wait for IRQ signal from tuner 2
846         {
847         timeout();                          // timeout if IRQ signal not received
848         if (toutf)
849             {
850             timeoutm();                     // display timeout msg on Hyper Terminal
851             break;
852             }
853         }
854
855         sendIIC(4);                         // send next 4 bytes from EEPROM to tuner 2
856         delay3();
857         sendIIC(13);                        // send next 13 bytes from EEPROM to tuner 2
858         delay3();
859         sendIIC(8);                         // send next 8 bytes from EEPROM to tuner 2
860
861         tout = 1;
862         while (!IRQ)                    // wait for IRQ signal from tuner 2
863         {
864         timeout();                          // timeout if IRQ signal not received
865         if (toutf)
866             {
867             timeoutm();                     // display timeout msg on Hyper Terminal
868             break;
869             }
870         }
871
872         sptr = 0;                       // point to first byte of tuner addr and data in EEPROM (0A:9F) - clear IRQ2
873         sendIIC(1);                         // send one byte from EEPROM to tuner 2
874
875         S0 = 0;                             // (S2 S1 S0 : 1 1 0) select inputs for fpga uC interface to O/P fpga data
876         S1 = 1;
877         S2 = 1;
878         printmsg(freqm,1);                  // display tuner frequency setting confirmation msg on Hyper Terminal
879         printmsg(equals,0);
880         typeNUM(freqXY);
881         printmsg(MHz,2);
882         tuneron=1;                          // set tuner on flag to prevent tuners being initialized again
883         }
884 }
885
886 void setgain (void)                         // setup a DC voltage for IF gain setting on tuners
887 {                                           // sends serial number to serial DACs on USRP2 motherboard using SPI
888 unsigned char gain;
```

```
889  gainf = 0;
890  SD0 = 0;
891  SD1 = 0;
892  S0 = 0;                                    // (S2 S1 S0 : 0 1 0) select inputs for fpga uC interface to set tuner gain
893  S1 = 1;
894  S2 = 0;
895  if (source == 'X')
896       {
897       sdata = 0x19;                         // set gain X - address of X gain DAC
898       gain = BCD(gainX);
899       }
900  else
901       {
902       sdata = 0x18;                         // set gain Y - address of Y gain DAC
903       gain = BCD(gainY);
904       }
905  sendbyte();
906  sdata = gain;                          // send upper 8 bits of gain DAC on SPI
907  sendbyte();
908  sdata = 0x00;                              // send lower 8 bits of gain DAC on SPI
909  sendbyte();
910  S2 = 1;                                    // (S2 S1 S0 : 1 1 0) select inputs for fpga uC interface to O/P fpga data
911  }
912
913  unsigned char BCD (unsigned char setptr)    // 3 character digits to binary number conversion (255 maximum)
914  {
915  unsigned char binary,cntr;
916
917  binary = 0;
918  cntr = settings[setptr] - 0x30;            // ASCII to numeric conversion 100's digit
919  while (cntr)
920       {
921       binary = binary + 100;
922       --cntr;
923       }
924  ++setptr;
925  cntr = settings[setptr] - 0x30;            // ASCII to numeric conversion 10's digit
926  while (cntr)
927       {
928       binary = binary + 10;
929       --cntr;
930       }
931  ++setptr;
932  binary = binary + settings[setptr] - 0x30;      // ASCII to numeric conversion 1's digit
933  return (binary);
934  }
935
936  unsigned short long BCD3 (unsigned char setptr)
937  {                                               // 3 character digits to binary number X 1000 conversion
938  unsigned char cntr;                        // tuner freq setting is in kHz
939  unsigned short long binary;                // but this program only allows freq settings in MHz
940
941  binary = 0;
942  cntr = settings[setptr] - 0x30;            // ASCII to numeric conversion 100's digit
943  while (cntr)
944       {
945       binary = binary + 100000;
946       --cntr;
947       }
948  ++setptr;
949  cntr = settings[setptr] - 0x30;            // ASCII to numeric conversion 10's digit
950  while (cntr)
951       {
952       binary = binary + 10000;
953       --cntr;
954       }
955  ++setptr;
956  cntr = settings[setptr] - 0x30;            // ASCII to numeric conversion 1's digit
957  while (cntr)
958       {
959       binary = binary + 1000;
960       --cntr;
961       }
962  return (binary);
```

```
963  }
964
965
966  void setfreq (void)                    // change frequency of tuners
967  {
968  unsigned short long frequency;
969
970  freqf = 0;                             // clear the change frequency flag
971  frequency = BCD3(freqXY);              // convert 3 char digits (MHz) to binary and multiply by 1000 (kHz)
972  freq3 = frequency;                     // split 24 bit binary number (frequency) into 3 eight bit numbers for comms
973  frequency>>=8;
974  freq2 = frequency;
975  frequency>>=8;
976  freq1 = frequency;
977  if (tuneron)
978      {
979      SD0 = 0;
980      SD1 = 0;
981      S0 = 1;                            // (S2 S1 S0 : 0 0 1) select inputs for fpga uC interface to setup tuner;
982      S1 = 0;
983      S2 = 0;
984
985      device = tuner1;
986      sptr = 28;                         // point to 28th byte of tuner addr and data in EEPROM (14:43)
987      sendIIC(7);                        // send 7 bytes of data from EEPROM to tuner 1
988
989
990      device = tuner2;
991      sptr = 28;                         // point to 28th byte of tuner addr and data in EEPROM (14:43)
992      sendIIC(7);                        // send 7 bytes of data from EEPROM to tuner 2
993
994
995      printmsg(freqm,1);
996      printmsg(equals,0);
997      typeNUM(freqXY);
998      printmsg(MHz,2);
999      S0 = 0;                            // (S2 S1 S0 : 1 1 0) select inputs for fpga uC interface to O/P fpga data ;
1000     S1 = 1;
1001     S2 = 1;
1002     }
1003 }
1004
1005 void setphase (void)                   // not implemented at present
1006 {
1007 phasef = 0;
1008 }
1009
1010 void setbw (void)                      // not implemented at present
1011 {
1012 bwf = 0;
1013 }
1014
1015 void sendIIC (unsigned char bytes)     // send (bytes) bytes on IIC comms to tuners
1016 {
1017 unsigned char i,addr;
1018
1019 GIE = 0;
1020 for (i=0 ; i<bytes; i++)
1021     {
1022     SDA = 1;                           // start IIC
1023     delay4();
1024     SCL = 1;
1025     delay4();
1026
1027     sbyte = device;                    // send tuner address on IIC
1028     IICbyte();
1029     sbyte = eeprom_read(sptr);         // send tuner register address
1030     addr = sbyte;
1031     IICbyte();
1032     sbyte = eeprom_read(sptr+40);      // send tuner register data
1033     if (addr == 0x16) sbyte = freq1;
1034     if (addr == 0x17) sbyte = freq2;
1035     if (addr == 0x18) sbyte = freq3;
1036     IICbyte();
```

```
1037        ++sptr;
1038
1039        delay4();                                   // stop IIC
1040        SDA = 1;
1041        delay4();
1042        SCL = 0;
1043        delay4();
1044        SDA = 0;
1045        delay4();
1046        }
1047  GIE = 1;
1048  }
1049
1050  void IICbyte (void)                               // send one byte on IIC to tuners
1051  {
1052  unsigned char i;
1053  for(i = 0 ; i<8; i++ )
1054        {
1055        if (sbyte & 0x80) SDA = 0;
1056        else SDA = 1;
1057        sbyte <<= 1;
1058        delay4();
1059        SCL = 0;
1060        delay4();
1061        SCL = 1;
1062        }
1063  delay4();
1064  SDA = 0;                                          // release SDA
1065  delay4();
1066  SCL = 0;                                          // clock ack bit from slave
1067  delay4();
1068  SCL = 1;
1069  delay4();
1070  }
1071
1072
1073  void sendbyte (void)                            // send one byte on SPI to tuner gain DACs
1074  {
1075  unsigned char i;
1076
1077  for(i = 0 ; i<8; i++ )
1078        {
1079        if (sdata & 0x80) SD1 = 1;
1080        else SD1 = 0;
1081        sdata <<= 1;
1082        delay4();
1083        SD0 = 1;
1084        delay4();
1085        SD0 = 0;
1086        }
1087  SD1 = 0;
1088  }
1089
1090  void timeout (void)                               // timeout for receiving IRQ signal from tuners
1091  {
1092  if (tout) tcnt = 50;
1093  tout = 0;
1094  toutf = 0;
1095  delay2();
1096  --tcnt;
1097  if (tcnt==0) toutf = 1;
1098  }
1099
1100  void timeoutm (void)                            // timeout message display on Hyper Terminal
1101  {
1102  new1 = 1;
1103  strcpy(msg,"TIMEOUT") ;
1104  typem();
1105  }
1106
1107
1108  void delay1 (void)                                // approx 200 ms delay
1109  {
1110  unsigned char i,j ;
```

```
1111
1112 for(i = 255 ; --i ;)
1113                 {for(j = 255 ; --j ;)
1114                 continue;
1115                 }
1116 }
1117
1118 void delay2 (void)                          // approx 10 ms delay
1119 {
1120 unsigned char i,j ;
1121
1122 for(i = 10 ; --i ;)
1123                 {for(j = 255 ; --j ;)
1124                 continue;
1125                 }
1126 }
1127
1128 void delay3 (void)                          // approx 50 us delay
1129 {
1130 unsigned char j ;
1131
1132 for(j = 16 ; --j ;)   continue;
1133 }
1134
1135 void delay4 (void)                          // approx 10 us delay
1136 {
1137 unsigned char j ;
1138
1139 for(j = 4 ; --j ;)    continue; delay
1140 }
```

ANNEXURE B

TDA18272 silicon tuner data sheet extracts

TDA18272HN

Hybrid (analog and digital) Silicon Tuner for terrestrial and cable TV reception

Rev. 2 — 7 July 2010 Product data sheet

1. General description

The TDA18272HN is a Silicon Tuner designed for terrestrial and cable TV reception for both analog and digital signals. TDA18272HN/M (Master) is to be used as a stand-alone tuner IC or Master in dual tuner application. TDA18272HN/S (Slave) is only to be used as Slave Silicon Tuner in dual tuner application.

The TDA18272HN supports all analog and digital TV standards and delivers a LOW IF (LIF) signal to a demodulator for analog TV and/or a channel demodulator for digital TV.

2. Features and benefits

- Fully integrated IF selectivity; eliminating the need for external SAW filters
- Worldwide multistandard terrestrial and cable
- Fully integrated oscillators
- Alignment free
- Single 3.3 V supply voltage
- Integrated wideband gain control
- Crystal oscillator output buffer (16 MHz) for single crystal applications
- I^2C-bus interface compatible with 3.3 V microcontrollers
- Slave tuner output function to drive second (slave) Silicon Tuner
- Easy programming
- 5 ms tuning time
- LIF channel center frequency output ranging from 3 MHz to 5 MHz
- 1.7 MHz, 6 MHz, 7 MHz, 8 MHz and 10 MHz channel bandwidths
- Ready for DVB-T2
- RoHS compliant

3. Applications

- Hybrid (analog and digital) STB, TV, PCTV
- Analog (PAL, SECAM, NTSC) and digital (DVB-T/T2/C/H, DTMB, ATSC, ISDB-T) standards supported
- Targeted specification (based on channel decoder or demodulator capabilities):
  - Digital terrestrial ATSC A74 compliance (US)
  - NorDig cable (EU)
  - C-BOOK (Cable, EU)
  - CENELEC EN55020 (EU)

- NorDig 2.0 (EU TV)
- E-BOOK and D-BOOK
- ARIB STD-B21 for ISDB-T
- OCUR (US)

4. Quick reference data

Table 1. Quick reference data

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{RF}	RF frequency	full range of RF input	42	-	870	MHz
NF_{tun}	tuner noise figure	75 Ω source; maximum gain	-	5.0	5.6	dB
φ_{jit}	phase jitter	UHF; integrated from 250 Hz to 4 MHz	-	0.4	0.6	degree
α_{image}	image rejection	worst case for image rejection and 4 MHz IF frequency for levels above −50 dBm	57.5	63	-	dB
ICP_{1dB}	1 dB input compression point	at tuner input and minimum gain	124	-	-	dBμV

5. Ordering information

Table 2. Ordering information

Type number	Package		
	Name	Description	Version
TDA18272HN/M/C1[1]	HVQFN40	plastic thermal enhanced very thin quad flat package; no leads; 40 terminals; body 6 × 6 × 0.85 mm	SOT618-1
TDA18272HN/S/C1[2]			

[1] M for master.

[2] S for slave.

6. Block diagram

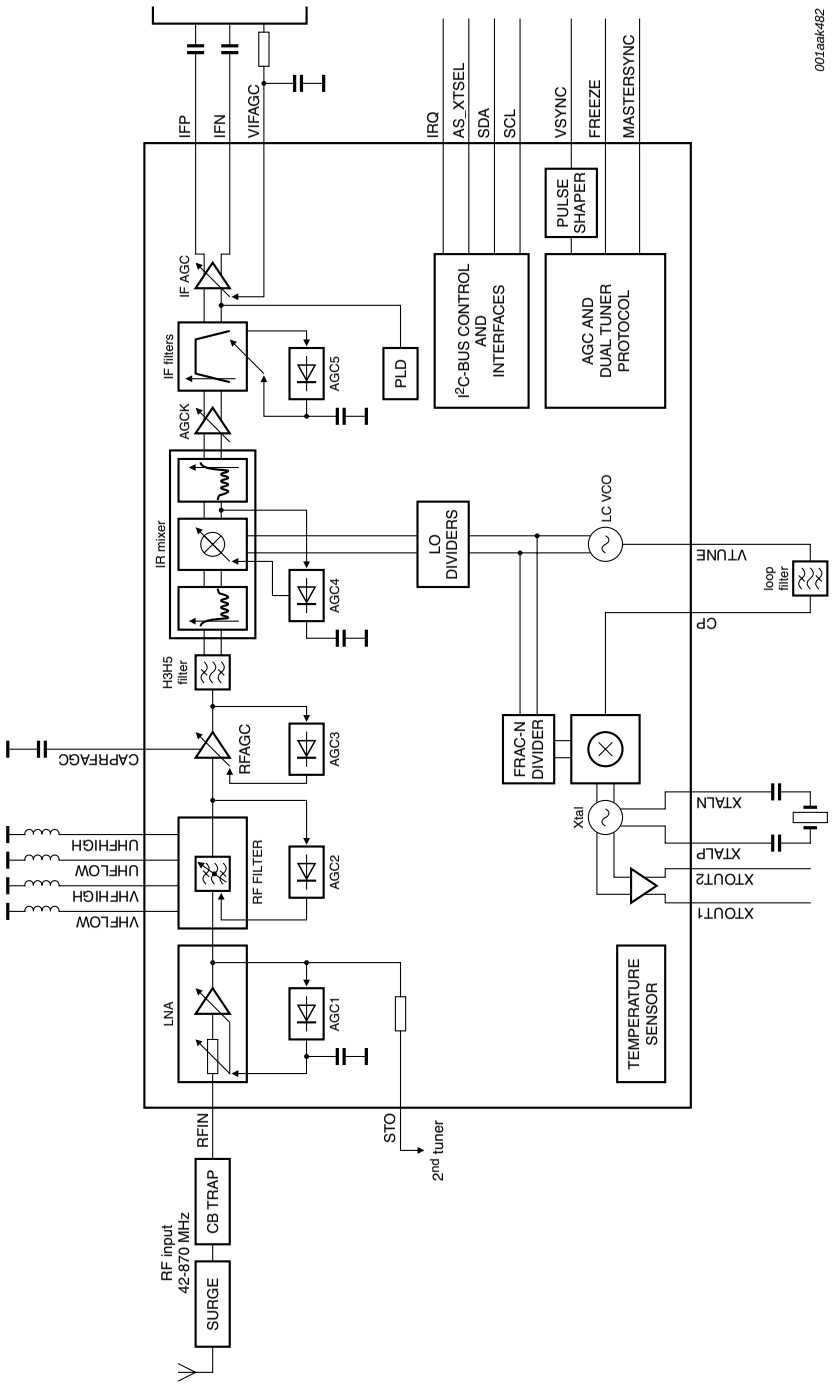


Fig 1. Block diagram TDA18272HN/M

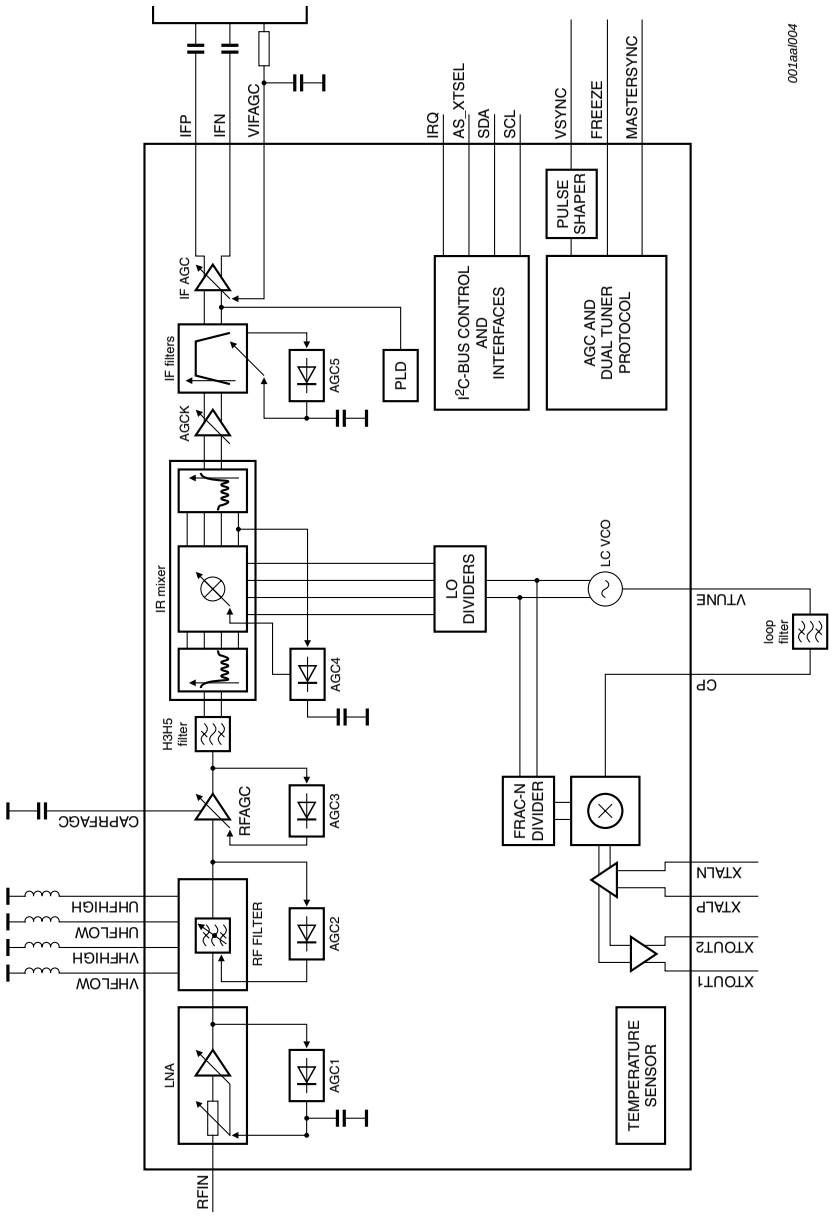


Fig 2. Block diagram TDA18272HN/S

7. Pinning information

7.1 Pinning

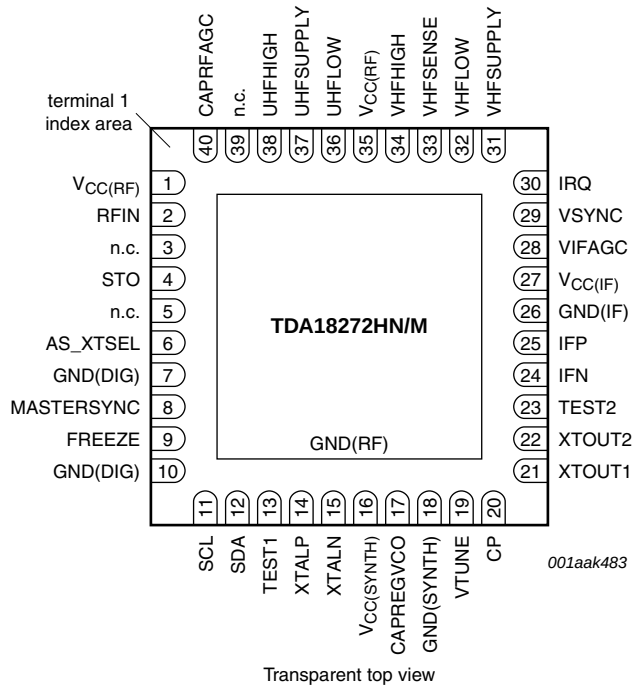


Fig 3. TDA18272HN/M master pin configuration

7.2 Pin description

Table 3. Master pin description

Symbol	Pin	Description
$V_{CC(RF)}$	1	RF supply voltage
RFIN	2	unbalanced RF input
n.c.	3	not connected
STO	4	slave tuner output
n.c.	5	not connected
AS_XTSEL	6	I^2C-bus address and XTOUT level selection input
GND(DIG)	7	digital ground supply voltage
MASTERSYNC	8	synchronization signal for dual tuner applications; do not connect for single tuner applications
FREEZE	9	synchronization signal for dual tuner applications; do not connect for single tuner applications
GND(DIG)	10	digital ground supply voltage
SCL	11	I^2C-bus clock input
SDA	12	I^2C-bus data input/output
TEST1	13	test input 1; connect to ground for operation mode

Table 3. Master pin description *…continued*

Symbol	Pin	Description
XTALP	14	crystal oscillator positive input
XTALN	15	crystal oscillator negative input
$V_{CC(SYNTH)}$	16	synthesizer supply voltage
CAPREGVCO	17	VCO regulator filtering input
GND(SYNTH)	18	synthesizer ground
VTUNE	19	VCO tuning voltage input
CP	20	charge pump output
XTOUT1	21	crystal oscillator buffer output 1
XTOUT2	22	crystal oscillator buffer output 2
TEST2	23	test input 2; connect to ground for operation mode
IFN	24	IF negative output
IFP	25	IF positive output
GND(IF)	26	IF ground
$V_{CC(IF)}$	27	IF supply voltage
VIFAGC	28	IF gain control input
VSYNC	29	AGCs synchronization input
IRQ	30	interrupt request output
VHFSUPPLY	31	RF filter VHF supply input
VHFLOW	32	RF filter VHF LOW input
VHFSENSE	33	RF filter VHF sense
VHFHIGH	34	RF filter VHF HIGH input
$V_{CC(RF)}$	35	RF filter supply voltage
UHFLOW	36	RF filter UHF LOW input
UHFSUPPLY	37	RF filter UHF supply input
UHFHIGH	38	RF filter UHF HIGH input
n.c.	39	not connected
CAPRFAGC	40	RF AGC filtering
GND(RF)	die pad	RF ground

7.3 Pinning

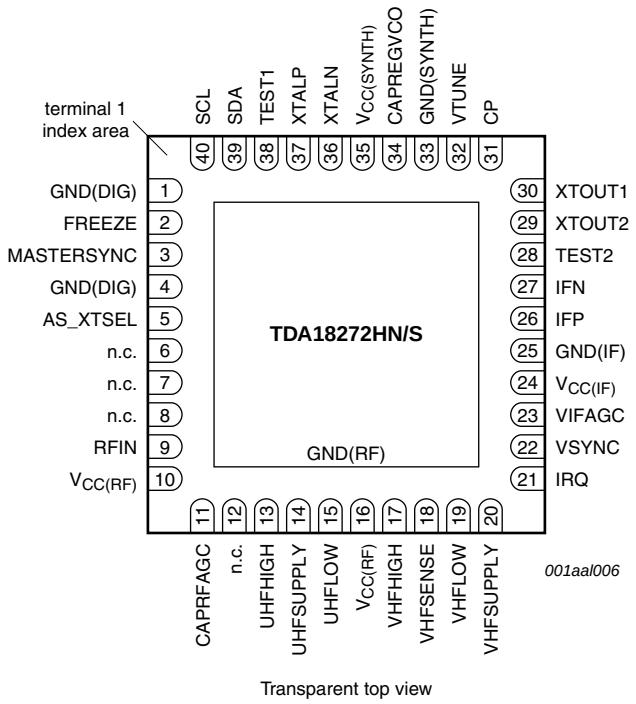


Fig 4. TDA18272HN/S slave pin configuration

7.4 Pin description

Table 4. Slave pin description

Symbol	Pin	Description
GND(DIG)	1	digital ground supply voltage
FREEZE	2	synchronization signal for dual tuner applications
MASTERSYNC	3	synchronization signal for dual tuner applications
GND(DIG)	4	digital ground supply voltage
AS_XTSEL	5	I^2C-bus address and XTOUT level selection input
n.c.	6	not connected
n.c.	7	not connected
n.c.	8	not connected
RFIN	9	unbalanced RF input
$V_{CC(RF)}$	10	RF supply voltage
CAPRFAGC	11	RF AGC filtering
n.c.	12	not connected
UHFHIGH	13	RF filter UHF high input
UHFSUPPLY	14	RF filter UHF supply input
UHFLOW	15	RF filter UHF LOW input
$V_{CC(RF)}$	16	RF filter supply voltage
VHFHIGH	17	RF filter VHF high input

Table 4. Slave pin description *...continued*

Symbol	Pin	Description
VHFSENSE	18	RF filter VHF sense
VHFLOW	19	RF filter VHF LOW input
VHFSUPPLY	20	RF filter VHF supply input
IRQ	21	interrupt request output
VSYNC	22	AGCs synchronization input
VIFAGC	23	IF gain control input
$V_{CC(IF)}$	24	IF supply voltage
GND(IF)	25	IF ground
IFP	26	IF positive output
IFN	27	IF negative output
TEST2	28	Test input 2; connect to ground for operation mode
XTOUT2	29	crystal oscillator buffer output 2
XTOUT1	30	crystal oscillator buffer output 1
CP	31	charge pump output
VTUNE	32	VCO tuning voltage input
GND(SYNTH)	33	synthesizer ground
CAPREGVCO	34	VCO regulator filtering input
$V_{CC(SYNTH)}$	35	synthesizer supply voltage
XTALN	36	crystal oscillator negative input
XTALP	37	crystal oscillator positive input
TEST1	38	Test input 1; connect to ground for operation mode
SDA	39	I^2C-bus data input/output
SCL	40	I^2C-bus clock input
GND(RF)	die pad	RF ground

8. Functional description

The Silicon Tuner is based on single down-conversion and LIF architecture that allows full integration of band-pass selectivity and eliminates the need for external SAW filters.

In the single and dual tuner applications, the RF input signal is applied to the master tuner. In the dual stream with splitter application, the RF input signal is fed to the input splitter and then sent to the master and slave tuners input (pin RFIN). Low Noise Amplifier (LNA) amplifies the signal and an alignment free RF tuned filter protects the rest of the tuner function against strong unwanted signals.

The LIF concept needs complex signals that highly suppress the N + 1 image channel thanks to image rejection calibration. A complex filter and a IF filter perform the IF selectivity. The IF filter depends on IF frequency choice and channel bandwidth. The IF filter is built with a IF Low-Pass Filter (LPF), a IF notch filter and a programmable IF High-Pass Filter (HPF) for more flexibility on IF frequency selection. The IF notch filter when activated, suppress the residual adjacent N – 1 sound carrier.

Continuous gain control is performed after the RF filters and the IF selectivity. Stepped AGC is available at all stages (LNA, RF filter, mixer and IF LPF) in order to optimize the tuner signal-to-noise ratio. Internal broadband level detectors control gain settings of all stepped AGC and the RF AGC amplifier. The steps in the different stages are automatically compensated in IF with AGCK to keep a constant IF output level. The demodulator controls the gain of the IF AGC amplifier to take advantage of the full ADC dynamic range.

A single LC-VCO operating at 8 GHz for master and 7 GHz for slave is used within a FRAC-N phase lock-loop to generate the LO frequency. A crystal oscillator differential input provides the clock reference signal. This signal can be provided to a slave tuner for the dual stream applications or demodulators through the crystal output buffer.

All the programming is performed via I^2C-bus transceiver. An embedded test tone generator is used for automatic calibration at Power-On-Reset (POR).

The power level indicator can be used to indicate the RF input signal strengths of the received channel.

8.1 Dual tuner mode

The TDA18272HN product family includes TDA18272HN/M (Master) and TDA18272HN/S (Slave).

The TDA18272HN/M is the Master tuner. Dual tuner applications require a specific TDA18272HN/S Slave tuner.

Two configurations are available for building dual tuner applications:

- either providing RF signal to RFIN pin of TDA18272HN/S from the STO pin of TDA18272HN/M through a 100 Ω resistor, see Figure 23 “Dual stream tuner application diagram recommended”. MASTERSYNC pins of TDA18272HN/M and TDA18272HN/S are connected together. In addition, FREEZE pins of TDA18272HN/M and TDA18272HN/S are connected together.
- using an external splitter to provide RF signal to both pins RFIN of TDA18272HN/M and TDA18272HN/S, see Figure 24 “Dual stream tuner application diagram with external splitter”. The MASTERSYNC pins of TDA18272HN/M and TDA18272HN/S are left open. The FREEZE pin of TDA18272HN/M is left open and the FREEZE pin of TDA18272HN/S is connected to 3.3 V through a 100 kΩ resistor.

The TDA18272HN/S has a clock buffer instead of a built-in crystal reference oscillator. In the dual tuner configurations, pins XTALP and XTALN of the slave are connected to pins XTOUT1 and XTOUT2 of the master through two 4.7 nF capacitors (see Figure 23 “Dual stream tuner application diagram recommended” and Figure 24 “Dual stream tuner application diagram with external splitter”).

TDA18272HN/S cannot be used alone.

8.2 RF filter

The RF filter block is an alignment free tunable Band-Pass Filter (BPF). At power up, a self-calibration is performed which compensates for the external and internal components frequency spread. The center frequency is automatically tuned to the frequency set using the I^2C-bus to suppress any unwanted interference across the broadband spectrum.

8.3 Crystal output mode

Pins XTOUT1 and XTOUT2 provide a symmetrical sine waveform which drives the channel demodulator and/or IF demodulator. The load on these outputs must be identical to ensure optimum performance matching. If only one crystal output is used, the unused output must be loaded with the same capacitance value. The XTOUT output level can be set to either 400 mV (p-p) or 800 mV (p-p) single ended, refer Table 31 “Pin AS_XTSEL decoding”.

8.4 AGC description

The tuner gain is composed of different variable gain stages spread according to the block diagram (see Figure 1 “Block diagram TDA18272HN/M” and Figure 2 “Block diagram TDA18272HN/S”). Using the different detectors at different stages, the gain is distributed to offer best trade-off between linearity and noise. Continuity gain variation is made in 3 dB steps. The AGCK stage compensates the 3 dB steps to ensure gain linearity.

The tuner gain is externally controlled via IFAGC voltage level applied to pin VIFAGC. The RF gain is set automatically, based on the defined AGCn_TOP values.

The different stages gain values are then a combination of the following input parameters:

- Input signal
- TOP values set via I^2C-bus
- IFAGC voltage level

The programmable Take Over Points (TOP) provide the optimal noise versus linearity trade-off during reception. The TOP values are carefully selected so that they do not overload the following stages or have too weak signal-to-noise ratio. They correspond to thresholds where the gain distribution changes inside the tuner.

In order to avoid instability of gain chain while working around thresholds, a hysteresis is implemented to avoid gain toggling. This is the reason why there are different values for TOP-up and/or TOP-down. Its main purpose is to make sure gain switch occurs to prevent signal distortion along the gain chain.

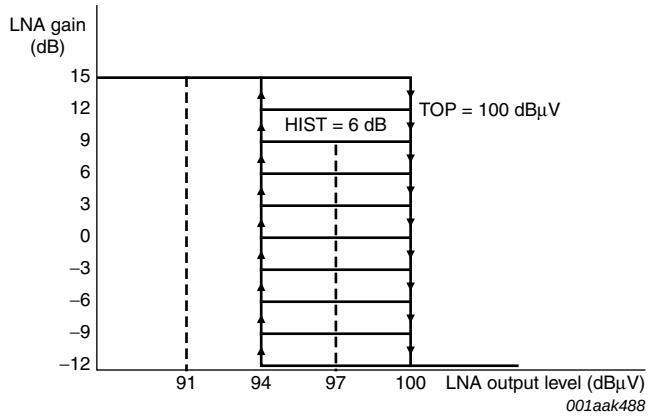


Fig 5. AGC1 TOP description example

The TOP values tuner settings are key for all tuner performance.

Table 5. AGC number/block correspondence

AGC number	Corresponding AGC block	Comment
AGC1	LNA AGC	
AGC2	RF Filter AGC	Not described, handled internally by tuner
AGC3	RF AGC	
AGC4	Mixer AGC	
AGC5	LPF AGC	

8.5 Harmonic 3 and harmonic 5 filter (H3H5)

In addition to the RF tuned filters, a H3H5 filter is implemented to prevent broadband down-conversion with third and fifth LO harmonics for off-air reception in case of presence of strong interferer.

8.6 Low-pass filter (LPF)

The programmable LPF avoids aliasing of demodulators Analog-to-Digital converters. In addition, it suppresses the remaining signals. The programming enables receive signal bandwidth of 1.7 MHz, 6 MHz, 7 MHz, 8 MHz and 10 MHz.

8.7 High-pass filter (HPF)

The HPF contributes to residual adjacent (N + 1) channel suppression after image rejection removal. It is intended mainly for digital reception.

8.8 Notch filter

This block has been implemented in the IF filter to optionally provide additional robustness against adjacent analog channels. It reduces the adjacent channel sound carrier level to prevent overloading of IF output stage. The notch frequency is tracked with LPF settings.

8.9 VSYNC pulse shape

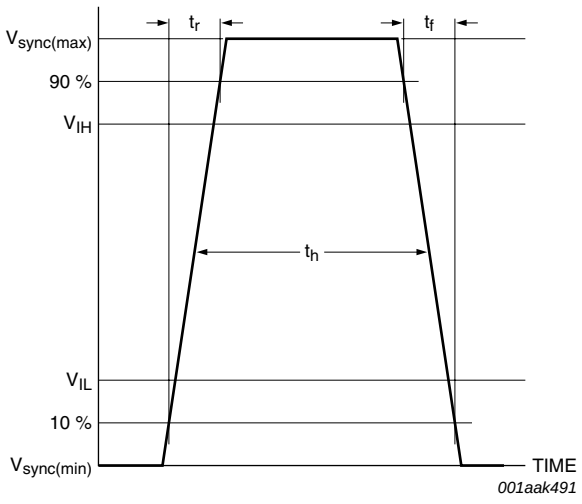


Fig 6. VSYNC pulse shape

Table 6. VSYNC signal characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{sync(max)}$	maximum synchronization voltage		-	-	Min(3.6 V, V_{CC} + 0.3 V)	V
$V_{sync(min)}$	minimum synchronization voltage		−0.3	-	-	V
V_{IL}	LOW-level input voltage		-	-	$0.3V_{CC}$	V
V_{IH}	HIGH-level input voltage		$0.7V_{CC}$	-	-	V
t_h	hold time	pulse shaper:				
		disabled	50	-	-	µs
		enabled	0.2	-	100	µs

VSYNC signal provided to the tuner allows tuner gain changes only during vertical synchronization to prevent picture quality degradation during analog reception.

A specific pulse shaper function is available to reshape the signal inside the tuner. The VSYNC signal duration must be according to Table 6.

If the pulse shaper is enabled, as long as t_h exceeds 0.2 µs, the pulse shaper reshapes the VSYNC signal. Should the signal at VSYNC pin be maintained longer than t_h max (100 µs) in Table 6, it would lead to unwanted tuner gain changes that could be visible on the picture during analog reception.

8.10 IR mixer

The LIF concept needs complex signals that highly suppress the N + 1 image channel thanks to image rejection calibration.

8.11 LO generation

A single LC-VCO operating at 7 GHz or 8 GHz is used within a FRAC-N phase lock-loop to generate the LO frequency. A crystal oscillator differential input provides the clock reference signal. This signal is provided to a slave tuner for dual stream applications or to a demodulator through the crystal output buffer.

8.12 Temperature sensor

The temperature sensor indicates the junction temperature of the IC via I^2C-bus for soldering check. Refer to Section 9.1.2 for detailed description and operation.

8.13 Power Level Detector (PLD)

The power level detector can be used to indicate the RF input signal strengths of the received channel via I^2C-bus. Refer to Section 9.1.4 for detailed description and operation.

8.14 I²C-bus transceiver

The TDA18272HN is controlled via the two-wire I²C-bus. There is one device address (7-bit). The read or write mode is selected with the R/W bit. To be able to have more than one tuner in an I²C-bus system, one of two possible addresses is selected depending on the voltage applied to address selection pin AS_XTSEL see Table 31 “Pin AS_XTSEL decoding”.

9. Control interface

9.1 Register table description

Table 7. Register table description

Address (hex)	Name[1]	Bit							
		7	6	5	4	3	2	1	0
00	ID_byte_1	Master_ Slave	Ident[14:8]						
01	ID_byte_2	Ident[7:0]							
02	ID_byte_3	Major_rev[3:0]				Minor_rev[3:0]			
03	Thermo_byte_1	-	TM_D[6:0]						
04	Thermo_byte_2	-							TM_ON
05	Power_state_byte_1	-						POR	LO_Lock
06	Power_state_byte_2	-				SM	SM_PLL	SM_LNA	0
07	Input_Power_Level_byte	-	Power_Level[6:0]						
08	IRQ_status	IRQ_status	-						
09	IRQ_enable	1	-	0					
0A	IRQ_clear	IRQ_clear	-	0					
0B	IRQ_set	0	-	0					
0C	AGC1_byte_1	0	1	-		AGC1_TOP[3:0]			
0D	AGC2_byte_1	-			0	1	1	1	1
0E	AGCK_byte_1	0		1	Pulse_Shaper _disable	0		AGCK_mode[1:0]	
0F	RF_AGC_byte_1	PD_RFAGC _Adapt	RFAGC_Adapt_TOP[1:0]		1	RF_Atten_ 3dB	AGC3_TOP[2:0]		
10	IR_MIXER_byte_1	-				AGC4_TOP[3:0]			
11	AGC5_byte_1	-	0		AGC5_HPF	AGC5_TOP[3:0]			
12	IF_AGC_byte	-					IF_Level[2:0]		
13	IF_byte_1	IF_HP_Fc[1:0]		IF_Notch	LP_FC_Offset[1:0]		LP_Fc[2:0]		
14	Reference_byte	0	Digital_Clock	-	0	-		XTout[1:0]	
15	IF_Frequency_byte	IF_Freq[7:0]							
16	RF_Frequency_byte_1	-				RF_Freq[19:16]			
17	RF_Frequency_byte_2	RF_Freq[15:8]							

Table 7. Register table description *…continued*

Address (hex)	Name[1]	Bit							
		7	6	5	4	3	2	1	0
18	RF_Frequency_byte_3	RF_Freq[7:0]							
19	MSM_byte_1	POWER_Meas	RF_CAL_AV	RF_CAL	IR_CAL[1:0]		0	RC_CAL	Calc_PLL
1A	MSM_byte_2	-						0	MSM_Launch
1B	PSM_byte_1	0		VHFIII			0		
1C	DCC_byte_1	0				-			
1D	FLO_Max_byte	-					0		
1E	IR_Cal_byte_1	0							
1F	IR_Cal_byte_2	1	0						
20	IR_Cal_byte_3	-					0		
21	IR_Cal_byte_4	-					0		
22	Vsync_Mgt_byte	0							1
23	IR_MIXER_byte_2	0			-			HI_Pass	DC_NOTCH
24	AGC1_byte_2	0				1	0		1
25	AGC5_byte_2	0		-		0	-	0	1
26	RF_Cal_byte_1	0							
27	RF_Cal_byte_2	0							
28	RF_Cal_byte_3	0							
29	RF_Cal_byte_4	0							
2A	RF_Cal_byte_5	0							
2B	RF_Cal_byte_6	0							
2C	RF_Filter_byte_1	0				1	0	1	1
2D	RF_Filter_byte_2	0							
2E	RF_Filter_byte_3	0							
2F	RF_Band_Pass_Filter_byte	0	-				1	1	0
30	CP_Current_byte	-	1	0			1	1	1
31	AGC_Det_Out_byte	X	X	X	X	X	X	X	X
32	RF_AGC_Gain_byte_1	-		RF_FILTER_GAIN[1:0]		LNA_GAIN[3:0]			
33	RF_AGC_Gain_byte_2	-					TOP_Agc3_read[2:0]		
34	IF_AGC_Gain_byte	-			LPF_GAIN[1:0]		IR_MIXER[2:0]		

Table 7. Register table description *…continued*

Address (hex)	Name[1]	Bit							
		7	6	5	4	3	2	1	0
35	Power_byte_1	X	X	X	X	X	X	X	X
36	Power_byte_2	-	-	0	X	1	1	1	0
37	Misc_byte_1	1	1	0	0	1	0	0	IRQ_Polarity
38	rfcal_log_1	X	X	X	X	X	X	X	X
39	rfcal_log_2	X	X	X	X	X	X	X	X
3A	rfcal_log_3	X	X	X	X	X	X	X	X
3B	rfcal_log_4	X	X	X	X	X	X	X	X
3C	rfcal_log_5	X	X	X	X	X	X	X	X
3D	rfcal_log_6	X	X	X	X	X	X	X	X
3E	rfcal_log_7	X	X	X	X	X	X	X	X
3F	rfcal_log_8	X	X	X	X	X	X	X	X
40	rfcal_log_9	X	X	X	X	X	X	X	X
41	rfcal_log_10	X	X	X	X	X	X	X	X
42	rfcal_log_11	X	X	X	X	X	X	X	X
43	rfcal_log_12	X	X	X	X	X	X	X	X
50	-	FORBIDDEN ACCESS							
67	-	FORBIDDEN ACCESS							
FE	-	FORBIDDEN ACCESS							
FF	-	FORBIDDEN ACCESS							

[1] The settings optimization is bound to channel decoder or demodulator choice and has a high impact on the tuner performances within system environment. Refer to Application Note for optimal settings.

Remark:

- The register tables are identical for TDA18272HN/M and TDA18272HN/S but not all functionality is available on slave. Refer to Figure 1 “Block diagram TDA18272HN/M” and Figure 2 “Block diagram TDA18272HN/S”.
- The values in Table 7 must be written as described for operation mode of the tuner.
- X means the tuner provides the value 0 or 1.
- - means the value is undefined. No internal bit corresponds to this address.

9.1.1 Device type address ID

Table 8. ID byte bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
00	ID_byte_1	7	Master_Slave	R	0	slave
					1	master
		6 to 0	Ident[14:8]	R	18272	type number information
01	ID_byte_2	7 to 0	Ident[7:0]			
02	ID_byte_3	7 to 4	Major_rev[3:0]	R	1	current major releases
		3 to 0	Minor_rev[3:0]	R	1	current minor releases

9.1.2 Temperature sensor

Table 9. Temperature sensor bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
03	Thermo_byte_1	6 to 0	TM_D[6:0]	R	-	a junction temperature measurement ranging from 22 °C to 127 °C is indicated through these bits
04	Thermo_byte_2	0	TM_ON	W		temperature sensor ON or OFF
					0	temperature sensor switched OFF
					1	temperature sensor switched ON

Remark: The temperature sensor value is updated each time a read is performed on the byte Thermo_byte_1, if TM_ON is set to 1. Otherwise, temperature sensor value is not updated.

9.1.3 Power state

Table 10. Power state bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
05	Power_state_byte_1	1	POR	R	-	detects when the tuner supply voltage went below POR threshold voltage. The tuner is then reset to its original settings (tuner not initialized).
					0	once it has been read
					1	the POR occurred on the tuner
		0	LO_Lock	R		LO lock flag
					0	PLL unlocked
					1	PLL locked
06	Power_state_byte_2	3	SM	R/W	[1]	Sleep mode control bits
		2	SM_PLL	R/W	[1]	
		1	SM_LNA	R/W	[1]	

[1] See Table 11.

Table 11. Mode selection

SM[1]	SM_PLL[1]	SM_LNA[1]	Mode
0	0	0	operation mode = ON
1	0	0	Standby mode with LNA and PLL ON
1[2]	1[2]	0[2]	Standby mode with LNA ON and PLL OFF
1	1	1	Standby mode with LNA and PLL OFF

[1] All others values are forbidden.

[2] Default value at POR.

When LNA is OFF, STO output is not active (no slave tuner option supported).

9.1.4 Power level detector

Table 12. Power level detector bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
07	Input_Power_Level_byte	6 to 0	Power_Level[6:0]	R	-	The power value is in the range from 40 dBμV (RMS) to 110 dBμV (RMS). Outside this range, Power_Level value[1] is computed as follows
					0	power < 40 dBμV (RMS)
					1	power ≥ 40 dBμV (RMS)
					...	...
					126	power ≤ 110 dBμV (RMS)
					127	power > 110 dBμV (RMS)

[1] Power_Level value is updated only if requested via triggering of MSM_byte_1 and MSM_byte_2.

The power level is measured only on request. It is performed with the bytes MSM_byte_1 (address: 19) and MSM_byte_2 (address: 1A):

- Set MSM_byte_1 (value: 80) to indicate a power measurement is required
- Set MSM_byte_2 (value: 01) to trig the measurement
- Then read byte Input_Power_Level_byte (address: 07) for result

Remark: This hardware feature purpose is to ease channel search when no picture is displayed on screen only. Software alternative is available for channel power measurement without picture impact.

9.1.5 IRQ

Table 13. IRQ bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
08	IRQ_status	7	IRQ_status	R/W	0	IRQ_clear is set to 1
					1	all calibration sequences selected by MSM_byte_1 and launched by MSM_byte_2 are completed
0A	IRQ_clear	7	IRQ_clear	R/W	0	no action
					1	drops the bit IRQ_status

Remark: A level changed is generated on IRQ pin that reflects the IRQ_status bit. The polarity of the IRQ pin is set with IRQ_Polarity bit (address: 37). In operation mode, the IRQ status raised at the end of the calibration sequence selected with MSM_byte_1 and MSM_byte_2 and at each programming of a new channel.

9.1.6 AGC and Take Over Points (TOP)

Table 14. AGC and Take Over Points bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
0C	AGC1_byte_1	3 to 0	AGC1_TOP[3:0]	R/W	[1]	set the TOP of the LNA detection loop (AGC1) according to the reception standard or the system settings
0E	AGCK_byte_1	4	Pulse_Shaper_disable	R/W	0	the pulse signal applied on VSYNC pin is directly transmitted to the digital part
					1	the pulse duration is internally increased to 500 μs
		1 to 0	AGCK_mode[1:0]	R/W	[2]	control AGC modes according to required reception mode. It sets the time reference for AGC changes
0F	RF_AGC_byte_1	7	PD_RFAGC_Adapt	R/W		RF AGC adapts algorithm power-down
					0	ON
					1	OFF
		6 to 5	RFAGC_Adapt_TOP[1:0]	R/W	-	“AGC3 Adapt” algorithm decreases the AGC3 TOP for low LPF gains. LPF gain is set to low value by AGC5 detector loop in the ACI (N) and (N – 1) reception cases. Decreasing the AGC3 TOP then limits signal carrier to noise ratio degradation caused by distortion in the RF stages. At opposite, LPF gain is set to high value by AGC5 detector loop in the ACI (N – X) and (N + X) reception cases. Increasing the AGC3 TOP then limits signal carrier to noise ratio degradation caused by noise from RF AGC stage. PD_RFAGC_Adapt disables this algorithm. RFAGC_Adapt_TOP[1:0] enables the low AGC3 TOP value, the high one is enabled via the AGC3 TOP field.
		3	RF_Atten_3dB	R/W		Adds 3 dB attenuation out of RF AGC
					0	OFF
					1	ON
		2 to 0	AGC3_TOP[2:0]	R/W	[3]	sets the RF AGC, MIXER and LPF blocks TOP. These bits must be set according to the required reception standard and performances
10	IR_MIXER_byte_1	3 to 0	AGC4_TOP[3:0]	R/W	[4]	sets the RF AGC, MIXER and LPF blocks TOP. These bits must be set according to the required reception standard and performances

Table 14. AGC and Take Over Points bit descriptions *...continued*

Address	Register	Bit	Symbol	Access	Value	Description
11	AGC5_byte_1	4	AGC5_HPF	R/W		turns HPF in AGC5 detection loop
					0	OFF
					1	ON
		3 to 0	AGC5_TOP[3:0]	R/W	[5]	sets the RF AGC, MIXER and LPF blocks TOP. These bits must be set according to the required reception standard and performances
12	IF_AGC_byte	2 to 0	IF_Level[2:0]	R/W	[6]	sets the tuner required maximum output level. This enables internal computation of the best linearity to noise ratio based on required output level

[1] See Table 16.

[2] See Table 15.

[3] See Table 17.

[4] See Table 18.

[5] See Table 19.

[6] See Table 20.

Table 15. AGCK Mode values

AGCK_mode[1:0] (hex)[1]	Recommended for
1	Analog TV
2	Digital TV, FM

[1] All others values are forbidden.

Table 16. AGC1 TOP values

AGC1_TOP[3:0] (hex)[1]	AGC1 TOP Down (dBμV)	AGC1 TOP Up (dBμV)
0	95	89
5	99	93
A	100	94
F	101	95

[1] All others values are forbidden.

Table 17. AGC3 TOP values

AGC3_TOP[2:0] (hex)[1]	AGC3 TOP (dBμV)
1	96
3	100
4	102
5	104

[1] All others values are forbidden.

Table 18. AGC4 TOP values

AGC4_TOP[3:0] (hex)[1]	AGC4 TOP Down (dBμV)	AGC4 TOP Up (dBμV)
1	105	100
4	107	102
6	108	103
8	109	104
B	110	105
E	112	107

[1] All others values are forbidden.

Table 19. AGC5 TOP values

AGC5_TOP[3:0] (hex)[1]	AGC5 TOP Down (dBμV)	AGC5 TOP Up (dBμV)
1	105	100
4	107	102
6	108	103
8	109	104
B	110	105
E	112	107

[1] All others values are forbidden.

Table 20. Tuner output level

IF_Level[2:0] (hex)	Output level (V (p-p) differential)[1]	Minimum gain (dB)	Maximum gain (dB)
7	0.5	−12	18
6	0.6	−10.3	19.7
5	0.7	−9	21
4	0.85	−7.5	22.5
3	0.8	−8	22
2	1	−6	24
1	1.25	−4	26
0	2	0	30

[1] Output level depends on standard and ADC headroom.

9.1.7 IF Filtering

Table 21. IF Filtering bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
13	IF_byte_1	7 to 6	IF_HP_Fc[1:0]	R/W		selects the IF HPF cut-off frequency. The high-pass frequency must be set according to the required reception standard. High-pass frequency:
					00	0.4 MHz
					01	0.85 MHz
					10	1 MHz
					11	1.5 MHz
		5	IF_Notch	R/W		enables or disables a notch filter embedded for adjacent N − 1 sound carrier suppression. The notch frequency depends on LP_Fc[2:0].
					0	OFF
					1	ON
		4 to 3	LP_FC_Offset[1:0]	R/W		enables offset to LPF cut-off frequency providing further adjacent channel rejection
					00	0
					01	−4 %
					10	−8 %
					11	forbidden
		2 to 0	LP_Fc[2:0]	R/W	[1]	selects the IF LPF cut-off frequency. It must be set according to required reception standard.

[1] See Table 22.

Table 22. Low Pass Filter bits descriptions

LP_Fc[2:0][1]	LP cut-off frequency (MHz)	IF notch frequency (MHz)
100	1.7	-
000	6	6.5
001	7	7.25
010	8	8.25
011	10	-

[1] All others values are forbidden.

9.1.8 XTOUT

Table 23. XTOUT bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
14	Reference_byte	6	Digital_Clock	R/W		spreads digital clock power to improve tuner EMC behavior
					0	OFF
					1	ON
		1 to 0	XTout[1:0]	R/W		provides 16 MHz reference signal on the XTOUT1 and XTOUT2 pins. XTOUT mode:
					00	no signal
					01	forbidden
					10	forbidden
					11	16 MHz

9.1.9 IF and RF frequency

Table 24. IF and RF frequency bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
15	IF_Frequency_byte	7 to 0	IF_Freq[7:0]	R/W	-	sets the tuner to the required IF frequency by 50 kHz steps. For example, to set the IF frequency to 4 MHz, IF_Freq[7:0] value must be 4000 : 50 = 80
16	RF_Frequency_byte_1	3 to 0	RF_Freq[19:16]	R/W	-	sets the required RF frequency expressed in kHz
17	RF_Frequency_byte_2	7 to 0	RF_Freq[15:8]	R/W	-	
18	RF_Frequency_byte_3	7 to 0	RF_Freq[7:0]	R/W	-	

9.1.10 Calibration controls

Table 25. Calibration control bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
19	MSM_byte_1	7	POWER_Meas	R/W	-	these bits must be set according to the programming flowchart (see Figure 7) to control the calibration and calculation state machines embedded in the chip
		6	RF_CAL_AV	R/W	-	
		5	RF_CAL	R/W	-	
		4 to 3	IR_CAL[1:0]	R/W	-	
		1	RC_CAL	R/W	-	
		0	Calc_PLL	R/W	-	
1A	MSM_byte_2	0	MSM_Launch	R/W	-	

9.1.11 IF filtering options

Table 26. IR Mixer bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
23	IR_MIXER_byte_2	1	HI_Pass	R/W	-	controls the high-pass frequency filter
					0	OFF
					1	ON
		0	DC_NOTCH	R/W		controls a DC notch in the IR mixer
					0	OFF
					1	ON

9.1.12 Gain values

Table 27. AGC bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
32	RF_AGC_Gain_byte_1	5 to 4	RF_FILTER_GAIN[1:0]	R		RF FILTER gain value
					00	−11 dB
					01	−8 dB
					10	−5 dB
					11	−2 dB
		3 to 0	LNA_GAIN[3:0]	R	[1]	LNA gain value
					0000	−12 dB
					0001	−9 dB
					0010	−6 dB
					0011	−3 dB
					0100	0 dB
					0101	3 dB
					0110	6 dB
					0111	9 dB
					1000	12 dB
					1001	15 dB
33	RF_AGC_Gain_byte_2	2 to 0	TOP_Agc3_read[2:0]	R		gives the AGC3 TOP value
					000	94 dBµV (RMS)
					001	96 dBµV (RMS)
					010	98 dBµV (RMS)
					011	100 dBµV (RMS)
					100	102 dBµV (RMS)
					101	104 dBµV (RMS)
					110	106 dBµV (RMS)
					111	107 dBµV (RMS)

Table 27. AGC bit descriptions *...continued*

Address	Register	Bit	Symbol	Access	Value	Description
34	IF_AGC_Gain_byte	4 to 3	LPF_GAIN[1:0]	R		LPF gain value
					00	0 dB
					01	3 dB
					10	6 dB
					11	9 dB
		2 to 0	IR_MIXER[2:0]	R	[1]	IR MIXER gain value
					000	2 dB
					001	5 dB
					010	8 dB
					011	11 dB
					100	14 dB

[1] Other values are forbidden.

9.1.13 IRQ polarity

Table 28. IRQ polarity bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
37	Misc_byte_1	0	IRQ_Polarity	R/W		selects the IRQ pin polarity. IRQ pin output voltage when IRQ raised:
					0	V_{CC}
					1	0

9.1.14 rfcal_log

These bytes provide the outcome of the RF filter calibration. It can be used as an indicator regarding RF filter robustness implementation on PCB.

Table 29. rfcal_log bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
38	rfcal_log_1	7 to 0		R	-	provides RF filter calibration results according to the following convention: Bit [7] are set to 1 in case of calibration error Bit [6:0] is a signed number indicating the number of switch capacitors
39	rfcal_log_2	7 to 0		R	-	
3A	rfcal_log_3	7 to 0		R	-	
3B	rfcal_log_4	7 to 0		R	-	
3C	rfcal_log_5	7 to 0		R	-	
3D	rfcal_log_6	7 to 0		R	-	
3E	rfcal_log_7	7 to 0		R	-	
3F	rfcal_log_8	7 to 0		R	-	
40	rfcal_log_9	7 to 0		R	-	
41	rfcal_log_10	7 to 0		R	-	
42	rfcal_log_11	7 to 0		R	-	
43	rfcal_log_12	7 to 0		R	-	

9.1.15 Forbidden

Table 30. Forbidden bit descriptions

Address	Register	Bit	Symbol	Access	Value	Description
50 to 67	-	7 to 0	-	-	-	do not write or read these bytes. Any modification of these bits can lead to performance degradation.
FE	-	7 to 0	-	-	-	
FF	-	7 to 0	-	-	-	

9.2 Tuner programming sequences with IRQ

In the Figure 7,

- Transition 0 (initialization):
  - write Power_state_byte_2 (address: 06): wakes the tuner up
  - write Power_byte_2 (address 36; value 0C): post POR register update
  - write AGC1_byte_2 (address 24; value 49): post POR register update
  - write RF_Filter_byte_3 (address 2E; value 40): post POR register update
  - write AGCK_byte_1 (address 0E; value FF): post POR register update
  - write AGC5_byte_1 (address 11; value 4A): post POR register update
  - write IRQ_clear (address 0A; value 9F): makes sure the IRQ status is reset
  - write MSM_byte_1 (address 19; value 3B): selects which calibrations and/or calculations to perform
  - write MSM_byte_2 (address 1A; value 01): launches tuner calibration and/or calculation
  - read IRQ_status (address 08; value BF): IRQ is generated once operations are completed
  - write FLO_Max_byte (address 1D; value 0A) (slave only)
  - write AGC1_byte_1 (address 0C; value 09)
  - write Reference_byte (address 14; value 03)
  - write Reference_byte (address 14): sets clock mode
  - write Power_state_byte_2 (address 06): sets the tuner in Standby mode
- Transition 1 (standard selection):
  - write Power_state_byte_2 (address 06; value 00): wakes the tuner up
  - write Reference_byte (address 14; value 43)
  - write IF_Frequency_byte (address 15)
  - write IF_AGC_byte (address 12)
  - write IF_byte_1 (address 13)
  - write IR_MIXER_byte_2 (address 23)
  - write AGC1_byte_1 (address 0C)
  - write AGC2_byte_1 (address 0D)
  - write AGCK_byte_1 (address 0E)
  - write PSM_byte_1 (address 1B; value 60)
  - write RF_AGC_byte_1 (address 0F)

- write IR_MIXER_byte_1 (address 10)
- write AGC5_byte_1 (address 11)
- These settings depend on received TV standard (standard and demodulator dependant). Register values depend on the standard received.

- Transition 2 (first frequency selection after standard change):
  - write Power_state_byte_2 (address 06): wakes the tuner up
  - write Reference_byte (address 14): sets clock mode
  - write IRQ_clear (address 0A; value 9F): makes sure the IRQ status is reset
  - write RF_Frequency_byte_1 (address 16)
  - write RF_Frequency_byte_2 (address 17)
  - write RF_Frequency_byte_3 (address 18): sets the tuner to the required RF frequency
  - write MSM_byte_1 (address 19; value 41)
  - write MSM_byte_2 (address 1A; value 01): RF filters tuning, PLL locking
  - Tunes the settings that depend on the RF input frequency, expressed in kHz, RF filters tuning, PLL locking
  - read IRQ_status (address 08; value BF): IRQ raised once operations are completed
- Transition 3 (standby):
  - write Reference_byte (address 14): sets clock mode
  - Power_state_byte_2 (address 06)
  - Sets the tuner into one of the available Standby modes, keeping the digital data unchanged
- Transition 4 (frequency selection within same standard):
  - write Reference_byte (address 14; value 43)
  - write Reference_byte (address 14): sets clock mode
  - write IRQ_clear (address 0A; value 9F): makes sure the IRQ status is reset
  - RF_Frequency_byte_1 (address 16)
  - RF_Frequency_byte_2 (address 17)
  - RF_Frequency_byte_3 (address 18): sets the tuner to the required RF frequency
  - MSM_byte_1 (address 19; value 41)
  - MSM_byte_2 (address 1A; value 01): RF filters tuning, PLL locking
  - Tunes the settings that depend on the RF input frequency, expressed in kHz, RF filters tuning, PLL locking
  - read IRQ_status (address 08; value BF): IRQ raised once operations are completed

Remark: a transition action can be launched only after having acknowledged the IRQ of the previous state if applicable and clear the IRQ_clear byte.

Remark: IRQ can be polled in I^2C-bus register table or checked on dedicated pin.

Remark: more details are available in user programming guide.

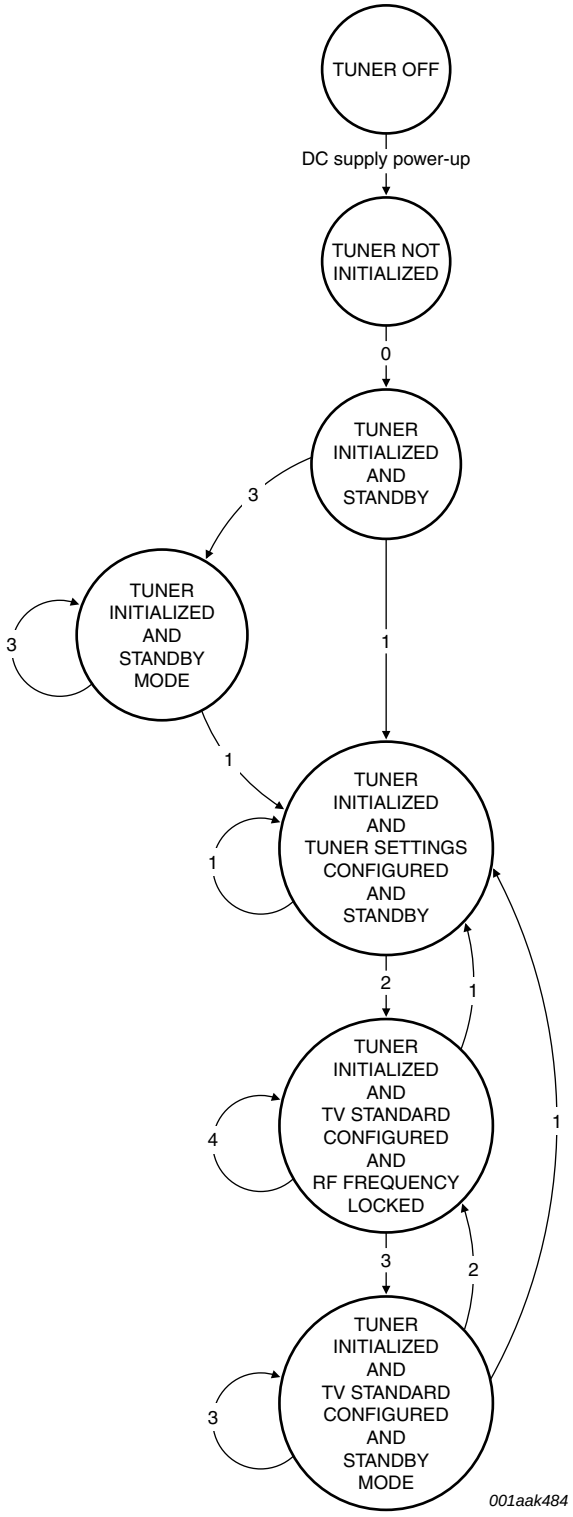


Fig 7. Tuner programming sequence with IRQ

9.3 Channel change programming required parameters

9.3.1 Same reception mode

The new channel to be programmed is within the same standard as the previous one (same channel demodulator).

To be programmed:

- RF frequency
- MSM byte

9.3.2 Different reception mode

The new channel to be programmed is from a different standard compared to the previous one (different channel demodulator).

To be programmed:

- AGC TOP
- IF frequency
- IF output level
- IF bandwidth
- RF frequency
- MSM byte

10. Hardware settings

10.1 XTOUT output level and I²C-bus address

Table 31. Pin AS_XTSEL decoding

Pin AS_XTSEL	Tuner write address (hex)	Tuner read address (hex)	Tuner status
0 V to 0.1 × V_{CC}	C0	C1	XTOUT 400 mV (p-p); single ended
0.2 × V_{CC} to 0.3 × V_{CC}	C0	C1	XTOUT 800 mV (p-p); single ended
0.4 × V_{CC} to 0.6 × V_{CC}	C6	C7	XTOUT 400 mV (p-p); single ended
0.9 × V_{CC} to V_{CC}	C6	C7	XTOUT 800 mV (p-p); single ended

Remark: sets the XTOUT output level from the master tuner to 400 mV (p-p) level in dual applications.

13. Thermal characteristics

Table 35. Thermal characteristics

Symbol	Parameter	Conditions	Typ	Unit
$R_{th(j-a)}$	thermal resistance from junction to ambient	according to JEDEC specification 4L board with 9 thermal vias[1][2][3]	31.4	K/W

[1] Measured in free air as defined by JEDEC standard.

[2] These values are given for information only. The thermal resistance depends strongly on the nature and design of the PCB used in the application. The thermal resistance given corresponds to the value that can be measured on a multilayer PCB (4 layers) as defined by JEDEC standard.

[3] The junction temperature influences strongly the reliability of an IC. The PCB used in the application contributes in a large part to the overall thermal characteristic. It must therefore be ensured that the junction temperature of the IC never exceeds $T_{j(max)}$ = 125 °C at the maximum ambient temperature.

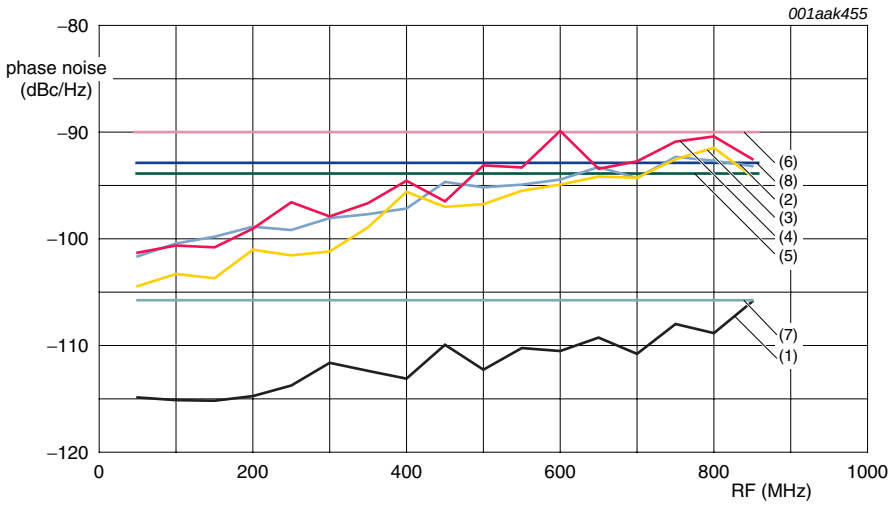
14. Characteristics

Table 36. General characteristics for TV reception (RF input to IF output)

T_{amb} = 25 °C; V_{CC} = 3.3 V; IF output level option 1 V (p-p); IF output load of 1 kΩ/1 pF; AGC1 TOP: 95 dBμV/89 dBμV; AGC3 TOP: 96 dBμV; AGC4 TOP: 105 dBμV/100 dBμV; AGC5 TOP: 105 dBμV/100 dBμV; IF_Level[2:0]: −6 dB/24 dB; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{CC}	supply voltage			3.13	3.30	3.47	V
$V_{O(p-p)(max)}$	maximum peak-to-peak output voltage	differential IF output		0.5	-	2	V
I_{CC}	supply current	operation mode		240	280	320	mA
		Standby mode:					
		crystal oscillator, STO output ON (default at POR)		35	45	55	mA
		only crystal oscillator ON		10	14	20	mA
f_{RF}	RF frequency	full range of RF input		42	-	870	MHz
		analog TV reception; picture		43.25	-	863.25	MHz
		center of channel		45	-	866	MHz
f_{lo}	local oscillator frequency	for respective IF frequency		50	-	870.25	MHz
VSWR	voltage standing wave ratio	RF input; 75 Ω nominal impedance, level below 95 dBμV		-	2.6	3	-
NF_{tun}	tuner noise figure	75 Ω source; maximum gain		-	5.0	5.6	dB
		75 Ω source; 60 dBμV condition	[1]	-	-	6.6	dB
$G_{v(max)}$	maximum voltage gain	all bands		82	86	-	dB
$G_{v(min)}$	minimum voltage gain	all bands		-	−28	−24	dB
ΔG_{rsd}	residual gain variation	in case of LNA AGC gain variation	[2]	-	-	0.8	dB
$\Delta G_{AGC(tun)}$	tuner AGC gain range			-	115	-	dB
$\Delta G_{AGC(IF)}$	IF AGC gain range	range measured between 0 V to 2 V	[3]	29	30	31	dB
ICP_{1dB}	1 dB input compression point	at tuner input and minimum gain		124	-	-	dBμV

14.2 Phase noise curves



(1) 100 kHz
(2) 10 kHz
(3) 1 kHz
(4) 250 Hz
(5) Typ_1 kHz
(6) Typ_250 Hz
(7) Typ_100 kHz
(8) Typ_10 kHz

Fig 18. Phase noise

14.3 IF AGC gain versus IF AGC voltage

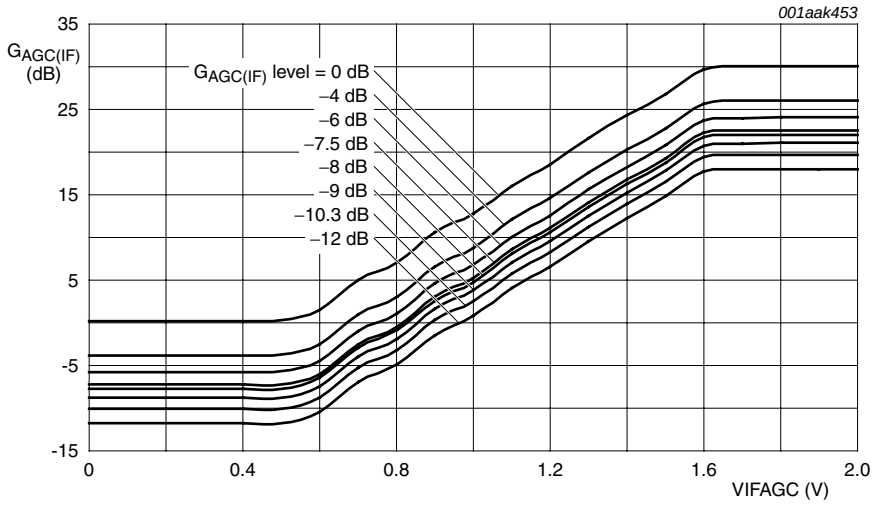


Fig 19. IF AGC gain versus IF AGC voltage

ANNEXURE C

Excel spreadsheet simulation for interference by product

	A	B	C	D	E	F	G
1	**Excel spreadsheet simulation for interference by product and correlation**						
2							
3	The formulae given for columns B to E are copied down to the bottom. For column F the moving average						
4	formula initially keeps the low end fixed while the upper end of the average grows until a full cycle of the						
5	input waveforms is included. From 370 degrees onwards, both the lower end and upper end of the moving						
6	average shift to give a sliding average window. The formula changes for column F are shown in column G.						
7							
8	**Angle (degrees)**	**Angle (radians)**	**WAVE A**	**WAVE B**	**PRODUCT**	**AVERAGE**	
9		**PI()*A10/180**	**SIN(B10)**	**SIN(B10-B10/10)**	**C10*D10/2**		
10	10	0.174532925	0.173648178	0.156434465	0.01358228	0.01358228	**E10**
11	20	0.34906585	0.342020143	0.309016994	0.05284502	0.033213649	**AVERAGE(E$10:E11)**
12	30	0.523598776	0.5	0.4539905	0.11349762	0.059974974	
13	40	0.698131701	0.64278761	0.587785252	0.18891054	0.092208865	
14	50	0.872664626	0.766044443	0.707106781	0.27083761	0.127934614	
15	60	1.047197551	0.866025404	0.809016994	0.35031463	0.164997951	
16	70	1.221730476	0.939692621	0.891006524	0.41863613	0.201231976	
17	80	1.396263402	0.984807753	0.951056516	0.46830392	0.234615969	
18	90	1.570796327	1	0.987688341	0.49384417	0.263419102	
19	100	1.745329252	0.984807753	1	0.49240388	0.28631758	
20	110	1.919862177	0.939692621	0.987688341	0.46406172	0.302476138	
21	120	2.094395102	0.866025404	0.951056516	0.41181955	0.311588089	
22	130	2.268928028	0.766044443	0.891006524	0.3412753	0.313871721	
23	140	2.443460953	0.64278761	0.809016994	0.26001305	0.310024673	
24	150	2.617993878	0.5	0.707106781	0.1767767	0.301141474	
25	160	2.792526803	0.342020143	0.587785252	0.1005172	0.288602457	
26	170	2.967059728	0.173648178	0.4539905	0.03941731	0.273944507	
27	180	3.141592654	1.22515E-16	0.309016994	1.893E-17	0.258725368	
28	190	3.316125579	-0.17364818	0.156434465	-0.01358228	0.244393387	
29	200	3.490658504	-0.34202014	1.22515E-16	-2.0951E-17	0.232173717	
30	210	3.665191429	-0.5	-0.156434465	0.03910862	0.222980141	
31	220	3.839724354	-0.64278761	-0.309016994	0.09931615	0.21735905	
32	230	4.01425728	-0.76604444	-0.4539905	0.17388845	0.215469024	
33	240	4.188790205	-0.8660254	-0.587785252	0.25451848	0.217096085	
34	250	4.36332313	-0.93969262	-0.707106781	0.33223151	0.221701502	
35	260	4.537856055	-0.98480775	-0.809016994	0.3983631	0.228496179	
36	270	4.71238898	-1	-0.891006524	0.44550326	0.236533478	
37	280	4.886921906	-0.98480775	-0.951056516	0.46830392	0.244810994	
38	290	5.061454831	-0.93969262	-0.987688341	0.46406172	0.252371364	
39	300	5.235987756	-0.8660254	-1	0.4330127	0.258392742	
40	310	5.410520681	-0.76604444	-0.987688341	0.37830658	0.26226093	
41	320	5.585053606	-0.64278761	-0.951056516	0.30566367	0.263617266	
42	330	5.759586532	-0.5	-0.891006524	0.22275163	0.262378913	
43	340	5.934119457	-0.34202014	-0.809016994	0.13835005	0.258731006	
44	350	6.108652382	-0.17364818	-0.707106781	0.0613939	0.253092803	
45	360	6.283185307	-2.4503E-16	-0.587785252	7.2012E-17	0.246062447	
46	370	6.457718232	0.173648178	-0.4539905	-0.03941731	0.244590236	**AVERAGE(E11:E46)**
47	380	6.632251158	0.342020143	-0.309016994	-0.05284502	0.241654402	
48	390	6.806784083	0.5	-0.156434465	-0.03910862	0.23741534	
49	400	6.981317008	0.64278761	-2.4503E-16	-7.8751E-17	0.232167825	
50	410	7.155849933	0.766044443	0.156434465	0.05991788	0.226308943	
51	420	7.330382858	0.866025404	0.309016994	0.13380828	0.220294878	
52	430	7.504915784	0.939692621	0.4539905	0.21330576	0.214591257	
53	440	7.679448709	0.984807753	0.587785252	0.28942774	0.209622474	
54	450	7.853981634	1	0.707106781	0.35355339	0.205725508	
55	460	8.028514559	0.984807753	0.809016994	0.3983631	0.203113264	
56	470	8.203047484	0.939692621	0.891006524	0.41863613	0.201851442	
57	480	8.37758041	0.866025404	0.951056516	0.41181955	0.201851442	

	A	B	C	D	E	F	G
58	490	8.552113335	0.766044443	0.987688341	0.37830658	0.202880089	
59	500	8.72664626	0.64278761	1	0.3213938	0.20458511	
60	510	8.901179185	0.5	0.987688341	0.24692209	0.206533593	
61	520	9.07571211	0.342020143	0.951056516	0.16264024	0.208259233	
62	530	9.250245036	0.173648178	0.891006524	0.07736083	0.20931322	
63	540	9.424777961	3.67545E-16	0.809016994	1.4867E-16	0.20931322	
64	550	9.599310886	-0.17364818	0.707106781	-0.0613939	0.207985119	
65	560	9.773843811	-0.34202014	0.587785252	-0.1005172	0.205192975	
66	570	9.948376736	-0.5	0.4539905	-0.11349762	0.200953912	
67	580	10.12290966	-0.64278761	0.309016994	-0.09931615	0.195436349	
68	590	10.29744259	-0.76604444	0.156434465	-0.05991788	0.188941728	
69	600	10.47197551	-0.8660254	3.67545E-16	-1.5915E-16	0.181871771	
70	610	10.64650844	-0.93969262	-0.156434465	0.07350016	0.174684789	
71	620	10.82104136	-0.98480775	-0.309016994	0.15216117	0.167845846	
72	630	10.99557429	-1	-0.4539905	0.22699525	0.161776179	
73	640	11.17010721	-0.98480775	-0.587785252	0.28942774	0.156807396	
74	650	11.34464014	-0.93969262	-0.707106781	0.33223151	0.153145446	
75	660	11.51917306	-0.8660254	-0.809016994	0.35031463	0.150848277	
76	670	11.69370599	-0.76604444	-0.891006524	0.3412753	0.14981963	
77	680	11.86823891	-0.64278761	-0.951056516	0.30566367	0.14981963	
78	690	12.04277184	-0.5	-0.987688341	0.24692209	0.150491032	
79	700	12.21730476	-0.34202014	-1	0.17101007	0.151398255	
80	710	12.39183769	-0.17364818	-0.987688341	0.08575514	0.152074956	
81	720	12.56637061	-4.9006E-16	-0.951056516	2.3304E-16	0.152074956	
82	730	12.74090354	0.173648178	-0.891006524	-0.07736083	0.151020969	
83	740	12.91543646	0.342020143	-0.809016994	-0.13835005	0.148645829	
84	750	13.08996939	0.5	-0.707106781	-0.1767767	0.144821716	
85	760	13.26450232	0.64278761	-0.587785252	-0.18891054	0.139574201	
86	770	13.43903524	0.766044443	-0.4539905	-0.17388845	0.133079581	
87	780	13.61356817	0.866025404	-0.309016994	-0.13380828	0.125645787	
88	790	13.78810109	0.939692621	-0.156434465	-0.07350016	0.117678956	
89	800	13.96263402	0.984807753	-4.90059E-16	-2.4131E-16	0.109639297	
90	810	14.13716694	1	0.156434465	0.07821723	0.10199107	
91	820	14.31169987	0.984807753	0.309016994	0.15216117	0.095152128	
92	830	14.48623279	0.939692621	0.4539905	0.21330576	0.089448506	
93	840	14.66076572	0.866025404	0.587785252	0.25451848	0.085079032	
94	850	14.83529864	0.766044443	0.707106781	0.27083761	0.082093783	
95	860	15.00983157	0.64278761	0.809016994	0.26001305	0.080388762	
96	870	15.18436449	0.5	0.891006524	0.22275163	0.07971736	
97	880	15.35889742	0.342020143	0.951056516	0.16264024	0.07971736	
98	890	15.53343034	0.173648178	0.987688341	0.08575514	0.079950536	
99	900	15.70796327	6.12574E-16	1	3.0629E-16	0.079950536	
100	910	15.88249619	-0.17364818	0.987688341	-0.08575514	0.079273835	
101	920	16.05702912	-0.34202014	0.951056516	-0.16264024	0.077548194	
102	930	16.23156204	-0.5	0.891006524	-0.22275163	0.074513361	
103	940	16.40609497	-0.64278761	0.809016994	-0.26001305	0.070049558	
104	950	16.58062789	-0.76604444	0.707106781	-0.27083761	0.064190677	
105	960	16.75516082	-0.8660254	0.587785252	-0.25451848	0.057120719	
106	970	16.92969374	-0.93969262	0.4539905	-0.21330576	0.049153888	
107	980	17.10422667	-0.98480775	0.309016994	-0.15216117	0.04070049	
108	990	17.27875959	-1	0.156434465	-0.07821723	0.032222365	
109	1000	17.45329252	-0.98480775	2.38893E-15	-1.1763E-15	0.024182706	
110	1010	17.62782545	-0.93969262	-0.156434465	0.07350016	0.016995724	
111	1020	17.80235837	-0.8660254	-0.309016994	0.13380828	0.010981658	
112	1030	17.9768913	-0.76604444	-0.4539905	0.17388845	0.006332024	
113	1040	18.15142422	-0.64278761	-0.587785252	0.18891054	0.003088881	
114	1050	18.32595715	-0.5	-0.707106781	0.1767767	0.001140398	

	A	B	C	D	E	F	G
115	1060	18.50049007	-0.34202014	-0.809016994	0.13835005	0.000233175	
116	1070	18.675023	-0.17364818	-0.891006524	0.07736083	-9.2133E-17	
117	1080	18.84955592	-7.3509E-16	-0.951056516	3.4956E-16	-8.8591E-17	
118	1090	19.02408885	0.173648178	-0.987688341	-0.08575514	-0.00023318	
119	1100	19.19862177	0.342020143	-1	-0.17101007	-0.0011404	
120	1110	19.3731547	0.5	-0.987688341	-0.24692209	-0.00308888	
121	1120	19.54768762	0.64278761	-0.951056516	-0.30566367	-0.00633202	
122	1130	19.72222055	0.766044443	-0.891006524	-0.3412753	-0.01098166	
123	1140	19.89675347	0.866025404	-0.809016994	-0.35031463	-0.01699572	
124	1150	20.0712864	0.939692621	-0.707106781	-0.33223151	-0.02418271	
125	1160	20.24581932	0.984807753	-0.587785252	-0.28942774	-0.03222237	
126	1170	20.42035225	1	-0.4539905	-0.22699525	-0.04070049	
127	1180	20.59488517	0.984807753	-0.309016994	-0.15216117	-0.04915389	
128	1190	20.7694181	0.939692621	-0.156434465	-0.07350016	-0.05712072	
129	1200	20.94395102	0.866025404	-7.35089E-16	-3.183E-16	-0.06419068	
130	1210	21.11848395	0.766044443	0.156434465	0.05991788	-0.07004956	
131	1220	21.29301687	0.64278761	0.309016994	0.09931615	-0.07451336	
132	1230	21.4675498	0.5	0.4539905	0.11349762	-0.07754819	
133	1240	21.64208272	0.342020143	0.587785252	0.1005172	-0.07927383	
134	1250	21.81661565	0.173648178	0.707106781	0.0613939	-0.07995054	
135	1260	21.99114858	8.57604E-16	0.809016994	3.4691E-16	-0.07995054	
136	1270	22.1656815	-0.17364818	0.891006524	-0.07736083	-0.07971736	
137	1280	22.34021443	-0.34202014	0.951056516	-0.16264024	-0.07971736	
138	1290	22.51474735	-0.5	0.987688341	-0.24692209	-0.08038876	
139	1300	22.68928028	-0.64278761	1	-0.3213938	-0.08209378	
140	1310	22.8638132	-0.76604444	0.987688341	-0.37830658	-0.08507903	
141	1320	23.03834613	-0.8660254	0.951056516	-0.41181955	-0.08944851	
142	1330	23.21287905	-0.93969262	0.891006524	-0.41863613	-0.09515213	
143	1340	23.38741198	-0.98480775	0.809016994	-0.3983631	-0.10199107	
144	1350	23.5619449	-1	0.707106781	-0.35355339	-0.1096393	
145	1360	23.73647783	-0.98480775	0.587785252	-0.28942774	-0.11767896	
146	1370	23.91101075	-0.93969262	0.4539905	-0.21330576	-0.12564579	
147	1380	24.08554368	-0.8660254	0.309016994	-0.13380828	-0.13307958	
148	1390	24.2600766	-0.76604444	0.156434465	-0.05991788	-0.1395742	
149	1400	24.43460953	-0.64278761	4.41032E-15	-1.4174E-15	-0.14482172	
150	1410	24.60914245	-0.5	-0.156434465	0.03910862	-0.14864583	
151	1420	24.78367538	-0.34202014	-0.309016994	0.05284502	-0.15102097	
152	1430	24.9582083	-0.17364818	-0.4539905	0.03941731	-0.15207496	
153	1440	25.13274123	-9.8012E-16	-0.587785252	2.8805E-16	-0.15207496	
154	1450	25.30727415	0.173648178	-0.707106781	-0.0613939	-0.15139825	
155	1460	25.48180708	0.342020143	-0.809016994	-0.13835005	-0.15049103	
156	1470	25.65634	0.5	-0.891006524	-0.22275163	-0.14981963	
157	1480	25.83087293	0.64278761	-0.951056516	-0.30566367	-0.14981963	
158	1490	26.00540585	0.766044443	-0.987688341	-0.37830658	-0.15084828	
159	1500	26.17993878	0.866025404	-1	-0.4330127	-0.15314545	
160	1510	26.35447171	0.939692621	-0.987688341	-0.46406172	-0.1568074	
161	1520	26.52900463	0.984807753	-0.951056516	-0.46830392	-0.16177618	
162	1530	26.70353756	1	-0.891006524	-0.44550326	-0.16784585	
163	1540	26.87807048	0.984807753	-0.809016994	-0.3983631	-0.17468479	
164	1550	27.05260341	0.939692621	-0.707106781	-0.33223151	-0.18187177	
165	1560	27.22713633	0.866025404	-0.587785252	-0.25451848	-0.18894173	
166	1570	27.40166926	0.766044443	-0.4539905	-0.17388845	-0.19543635	
167	1580	27.57620218	0.64278761	-0.309016994	-0.09931615	-0.20095391	
168	1590	27.75073511	0.5	-0.156434465	-0.03910862	-0.20519297	
169	1600	27.92526803	0.342020143	-9.80119E-16	-1.6761E-16	-0.20798512	
170	1610	28.09980096	0.173648178	0.156434465	0.01358228	-0.20931322	
171	1620	28.27433388	4.65535E-15	0.309016994	7.1929E-16	-0.20931322	

	A	B	C	D	E	F	G
172	1630	28.44886681	-0.17364818	0.4539905	-0.03941731	-0.20825923	
173	1640	28.62339973	-0.34202014	0.587785252	-0.1005172	-0.20653359	
174	1650	28.79793266	-0.5	0.707106781	-0.1767767	-0.20458511	
175	1660	28.97246558	-0.64278761	0.809016994	-0.26001305	-0.20288009	
176	1670	29.14699851	-0.76604444	0.891006524	-0.3412753	-0.20185144	
177	1680	29.32153143	-0.8660254	0.951056516	-0.41181955	-0.20185144	
178	1690	29.49606436	-0.93969262	0.987688341	-0.46406172	-0.20311326	
179	1700	29.67059728	-0.98480775	1	-0.49240388	-0.20572551	
180	1710	29.84513021	-1	0.987688341	-0.49384417	-0.20962247	
181	1720	30.01966313	-0.98480775	0.951056516	-0.46830392	-0.21459126	
182	1730	30.19419606	-0.93969262	0.891006524	-0.41863613	-0.22029488	
183	1740	30.36872898	-0.8660254	0.809016994	-0.35031463	-0.22630894	
184	1750	30.54326191	-0.76604444	0.707106781	-0.27083761	-0.23216782	
185	1760	30.71779484	-0.64278761	0.587785252	-0.18891054	-0.23741534	
186	1770	30.89232776	-0.5	0.4539905	-0.11349762	-0.2416544	
187	1780	31.06686069	-0.34202014	0.309016994	-0.05284502	-0.24459024	
188	1790	31.24139361	-0.17364818	0.156434465	-0.01358228	-0.24606245	
189	1800	31.41592654	-1.2251E-15	1.10263E-15	-6.7544E-31	-0.24606245	
190	1810	31.59045946	0.173648178	-0.156434465	-0.01358228	-0.24473435	
191	1820	31.76499239	0.342020143	-0.309016994	-0.05284502	-0.24235921	
192	1830	31.93952531	0.5	-0.4539905	-0.11349762	-0.23932437	
193	1840	32.11405824	0.64278761	-0.587785252	-0.18891054	-0.23608123	
194	1850	32.28859116	0.766044443	-0.707106781	-0.27083761	-0.23309598	
195	1860	32.46312409	0.866025404	-0.809016994	-0.35031463	-0.23079881	
196	1870	32.63765701	0.939692621	-0.891006524	-0.41863613	-0.22953699	
197	1880	32.81218994	0.984807753	-0.951056516	-0.46830392	-0.22953699	
198	1890	32.98672286	1	-0.987688341	-0.49384417	-0.23087979	
199	1900	33.16125579	0.984807753	-1	-0.49240388	-0.23349204	
200	1910	33.33578871	0.939692621	-0.987688341	-0.46406172	-0.23715399	
201	1920	33.51032164	0.866025404	-0.951056516	-0.41181955	-0.24152346	
202	1930	33.68485456	0.766044443	-0.891006524	-0.3412753	-0.2461731	
203	1940	33.85938749	0.64278761	-0.809016994	-0.26001305	-0.2506369	
204	1950	34.03392041	0.5	-0.707106781	-0.1767767	-0.25446101	
205	1960	34.20845334	0.342020143	-0.587785252	-0.1005172	-0.25725316	
206	1970	34.38298626	0.173648178	-0.4539905	-0.03941731	-0.25872537	
207	1980	34.55751919	-2.2051E-15	-0.309016994	3.407E-16	-0.25872537	
208	1990	34.73205211	-0.17364818	-0.156434465	0.01358228	-0.25725316	
209	2000	34.90658504	-0.34202014	-4.77786E-15	8.1706E-16	-0.25446101	
210	2010	35.08111797	-0.5	0.156434465	-0.03910862	-0.2506369	
211	2020	35.25565089	-0.64278761	0.309016994	-0.09931615	-0.2461731	
212	2030	35.43018382	-0.76604444	0.4539905	-0.17388845	-0.24152346	
213	2040	35.60471674	-0.8660254	0.587785252	-0.25451848	-0.23715399	
214	2050	35.77924967	-0.93969262	0.707106781	-0.33223151	-0.23349204	
215	2060	35.95378259	-0.98480775	0.809016994	-0.3983631	-0.23087979	
216	2070	36.12831552	-1	0.891006524	-0.44550326	-0.22953699	
217	2080	36.30284844	-0.98480775	0.951056516	-0.46830392	-0.22953699	
218	2090	36.47738137	-0.93969262	0.987688341	-0.46406172	-0.23079881	
219	2100	36.65191429	-0.8660254	1	-0.4330127	-0.23309598	
220	2110	36.82644722	-0.76604444	0.987688341	-0.37830658	-0.23608123	
221	2120	37.00098014	-0.64278761	0.951056516	-0.30566367	-0.23932437	
222	2130	37.17551307	-0.5	0.891006524	-0.22275163	-0.24235921	
223	2140	37.35004599	-0.34202014	0.809016994	-0.13835005	-0.24473435	
224	2150	37.52457892	-0.17364818	0.707106781	-0.0613939	-0.24606245	
225	2160	37.69911184	-1.4702E-15	0.587785252	-4.3207E-16	-0.24606245	
226	2170	37.87364477	0.173648178	0.4539905	0.03941731	-0.24459024	
227	2180	38.04817769	0.342020143	0.309016994	0.05284502	-0.2416544	
228	2190	38.22271062	0.5	0.156434465	0.03910862	-0.23741534	

	A	B	C	D	E	F	G
229	2200	38.39724354	0.64278761	4.90038E-15	1.575E-15	-0.23216782	
230	2210	38.57177647	0.766044443	-0.156434465	-0.05991788	-0.22630894	
231	2220	38.74630939	0.866025404	-0.309016994	-0.13380828	-0.22029488	
232	2230	38.92084232	0.939692621	-0.4539905	-0.21330576	-0.21459126	
233	2240	39.09537524	0.984807753	-0.587785252	-0.28942774	-0.20962247	
234	2250	39.26990817	1	-0.707106781	-0.35355339	-0.20572551	
235	2260	39.4444411	0.984807753	-0.809016994	-0.3983631	-0.20311326	
236	2270	39.61897402	0.939692621	-0.891006524	-0.41863613	-0.20185144	
237	2280	39.79350695	0.866025404	-0.951056516	-0.41181955	-0.20185144	
238	2290	39.96803987	0.766044443	-0.987688341	-0.37830658	-0.20288009	
239	2300	40.1425728	0.64278761	-1	-0.3213938	-0.20458511	
240	2310	40.31710572	0.5	-0.987688341	-0.24692209	-0.20653359	
241	2320	40.49163865	0.342020143	-0.951056516	-0.16264024	-0.20825923	
242	2330	40.66617157	0.173648178	-0.891006524	-0.07736083	-0.20931322	
243	2340	40.8407045	-1.96E-15	-0.809016994	7.9285E-16	-0.20931322	
244	2350	41.01523742	-0.17364818	-0.707106781	0.0613939	-0.20798512	
245	2360	41.18977035	-0.34202014	-0.587785252	0.1005172	-0.20519297	
246	2370	41.36430327	-0.5	-0.4539905	0.11349762	-0.20095391	
247	2380	41.5388362	-0.64278761	-0.309016994	0.09931615	-0.19543635	
248	2390	41.71336912	-0.76604444	-0.156434465	0.05991788	-0.18894173	
249	2400	41.88790205	-0.8660254	-1.47018E-15	6.3661E-16	-0.18187177	
250	2410	42.06243497	-0.93969262	0.156434465	-0.07350016	-0.17468479	
251	2420	42.2369679	-0.98480775	0.309016994	-0.15216117	-0.16784585	
252	2430	42.41150082	-1	0.4539905	-0.22699525	-0.16177618	
253	2440	42.58603375	-0.98480775	0.587785252	-0.28942774	-0.1568074	
254	2450	42.76056667	-0.93969262	0.707106781	-0.33223151	-0.15314545	
255	2460	42.9350996	-0.8660254	0.809016994	-0.35031463	-0.15084828	
256	2470	43.10963252	-0.76604444	0.891006524	-0.3412753	-0.14981963	
257	2480	43.28416545	-0.64278761	0.951056516	-0.30566367	-0.14981963	
258	2490	43.45869837	-0.5	0.987688341	-0.24692209	-0.15049103	
259	2500	43.6332313	-0.34202014	1	-0.17101007	-0.15139825	
260	2510	43.80776423	-0.17364818	0.987688341	-0.08575514	-0.15207496	
261	2520	43.98229715	-1.7152E-15	0.951056516	-8.1563E-16	-0.15207496	
262	2530	44.15683008	0.173648178	0.891006524	0.07736083	-0.15102097	
263	2540	44.331363	0.342020143	0.809016994	0.13835005	-0.14864583	
264	2550	44.50589593	0.5	0.707106781	0.1767767	-0.14482172	
265	2560	44.68042885	0.64278761	0.587785252	0.18891054	-0.1395742	
266	2570	44.85496178	0.766044443	0.4539905	0.17388845	-0.13307958	
267	2580	45.0294947	0.866025404	0.309016994	0.13380828	-0.12564579	
268	2590	45.20402763	0.939692621	0.156434465	0.07350016	-0.11767896	
269	2600	45.37856055	0.984807753	5.14541E-15	2.5336E-15	-0.1096393	
270	2610	45.55309348	1	-0.156434465	-0.07821723	-0.10199107	
271	2620	45.7276264	0.984807753	-0.309016994	-0.15216117	-0.09515213	
272	2630	45.90215933	0.939692621	-0.4539905	-0.21330576	-0.08944851	
273	2640	46.07669225	0.866025404	-0.587785252	-0.25451848	-0.08507903	
274	2650	46.25122518	0.766044443	-0.707106781	-0.27083761	-0.08209378	
275	2660	46.4257581	0.64278761	-0.809016994	-0.26001305	-0.08038876	
276	2670	46.60029103	0.5	-0.891006524	-0.22275163	-0.07971736	
277	2680	46.77482395	0.342020143	-0.951056516	-0.16264024	-0.07971736	
278	2690	46.94935688	0.173648178	-0.987688341	-0.08575514	-0.07995054	
279	2700	47.1238898	-1.715E-15	-1	8.575E-16	-0.07995054	
280	2710	47.29842273	-0.17364818	-0.987688341	0.08575514	-0.07927383	
281	2720	47.47295565	-0.34202014	-0.951056516	0.16264024	-0.07754819	
282	2730	47.64748858	-0.5	-0.891006524	0.22275163	-0.07451336	
283	2740	47.8220215	-0.64278761	-0.809016994	0.26001305	-0.07004956	
284	2750	47.99655443	-0.76604444	-0.707106781	0.27083761	-0.06419068	
285	2760	48.17108736	-0.8660254	-0.587785252	0.25451848	-0.05712072	

	A	B	C	D	E	F	G
286	2770	48.34562028	-0.93969262	-0.4539905	0.21330576	-0.04915389	
287	2780	48.52015321	-0.98480775	-0.309016994	0.15216117	-0.04070049	
288	2790	48.69468613	-1	-0.156434465	0.07821723	-0.03222237	
289	2800	48.86921906	-0.98480775	-8.82064E-15	4.3433E-15	-0.02418271	
290	2810	49.04375198	-0.93969262	0.156434465	-0.07350016	-0.01699572	
291	2820	49.21828491	-0.8660254	0.309016994	-0.13380828	-0.01098166	
292	2830	49.39281783	-0.76604444	0.4539905	-0.17388845	-0.00633202	
293	2840	49.56735076	-0.64278761	0.587785252	-0.18891054	-0.00308888	
294	2850	49.74188368	-0.5	0.707106781	-0.1767767	-0.0011404	
295	2860	49.91641661	-0.34202014	0.809016994	-0.13835005	-0.00023318	
296	2870	50.09094953	-0.17364818	0.891006524	-0.07736083	1.65377E-16	
297	2880	50.26548246	-1.9602E-15	0.951056516	-9.3215E-16	1.64156E-16	
298	2890	50.44001538	0.173648178	0.987688341	0.08575514	0.000233175	
299	2900	50.61454831	0.342020143	1	0.17101007	0.001140398	
300	2910	50.78908123	0.5	0.987688341	0.24692209	0.003088881	
301	2920	50.96361416	0.64278761	0.951056516	0.30566367	0.006332024	
302	2930	51.13814708	0.766044443	0.891006524	0.3412753	0.010981658	
303	2940	51.31268001	0.866025404	0.809016994	0.35031463	0.016995724	
304	2950	51.48721293	0.939692621	0.707106781	0.33223151	0.024182706	
305	2960	51.66174586	0.984807753	0.587785252	0.28942774	0.032222365	
306	2970	51.83627878	1	0.4539905	0.22699525	0.04070049	
307	2980	52.01081171	0.984807753	0.309016994	0.15216117	0.049153888	
308	2990	52.18534463	0.939692621	0.156434465	0.07350016	0.057120719	
309	3000	52.35987756	0.866025404	5.39044E-15	2.3341E-15	0.064190677	
310	3010	52.53441049	0.766044443	-0.156434465	-0.05991788	0.070049558	
311	3020	52.70894341	0.64278761	-0.309016994	-0.09931615	0.074513361	
312	3030	52.88347634	0.5	-0.4539905	-0.11349762	0.077548194	
313	3040	53.05800926	0.342020143	-0.587785252	-0.1005172	0.079273835	
314	3050	53.23254219	0.173648178	-0.707106781	-0.0613939	0.079950536	
315	3060	53.40707511	5.63547E-15	-0.809016994	-2.2796E-15	0.079950536	
316	3070	53.58160804	-0.17364818	-0.891006524	0.07736083	0.07971736	
317	3080	53.75614096	-0.34202014	-0.951056516	0.16264024	0.07971736	
318	3090	53.93067389	-0.5	-0.987688341	0.24692209	0.080388762	
319	3100	54.10520681	-0.64278761	-1	0.3213938	0.082093783	
320	3110	54.27973974	-0.76604444	-0.987688341	0.37830658	0.085079032	
321	3120	54.45427266	-0.8660254	-0.951056516	0.41181955	0.089448506	
322	3130	54.62880559	-0.93969262	-0.891006524	0.41863613	0.095152128	
323	3140	54.80333851	-0.98480775	-0.809016994	0.3983631	0.10199107	
324	3150	54.97787144	-1	-0.707106781	0.35355339	0.109639297	
325	3160	55.15240436	-0.98480775	-0.587785252	0.28942774	0.117678956	
326	3170	55.32693729	-0.93969262	-0.4539905	0.21330576	0.125645787	
327	3180	55.50147021	-0.8660254	-0.309016994	0.13380828	0.133079581	
328	3190	55.67600314	-0.76604444	-0.156434465	0.05991788	0.139574201	
329	3200	55.85053606	-0.64278761	-1.96024E-15	6.3001E-16	0.144821716	
330	3210	56.02506899	-0.5	0.156434465	-0.03910862	0.148645829	
331	3220	56.19960191	-0.34202014	0.309016994	-0.05284502	0.151020969	
332	3230	56.37413484	-0.17364818	0.4539905	-0.03941731	0.152074956	
333	3240	56.54866776	-9.3107E-15	0.587785252	-2.7363E-15	0.152074956	
334	3250	56.72320069	0.173648178	0.707106781	0.0613939	0.151398255	
335	3260	56.89773362	0.342020143	0.809016994	0.13835005	0.150491032	
336	3270	57.07226654	0.5	0.891006524	0.22275163	0.14981963	
337	3280	57.24679947	0.64278761	0.951056516	0.30566367	0.14981963	
338	3290	57.42133239	0.766044443	0.987688341	0.37830658	0.150848277	
339	3300	57.59586532	0.866025404	1	0.4330127	0.153145446	
340	3310	57.77039824	0.939692621	0.987688341	0.46406172	0.156807396	
341	3320	57.94493117	0.984807753	0.951056516	0.46830392	0.161776179	
342	3330	58.11946409	1	0.891006524	0.44550326	0.167845846	

	A	B	C	D	E	F	G
343	3340	58.29399702	0.984807753	0.809016994	0.3983631	0.174684789	
344	3350	58.46852994	0.939692621	0.707106781	0.33223151	0.181871771	
345	3360	58.64306287	0.866025404	0.587785252	0.25451848	0.188941728	
346	3370	58.81759579	0.766044443	0.4539905	0.17388845	0.195436349	
347	3380	58.99212872	0.64278761	0.309016994	0.09931615	0.200953912	
348	3390	59.16666164	0.5	0.156434465	0.03910862	0.205192975	
349	3400	59.34119457	0.342020143	5.63547E-15	9.6372E-16	0.207985119	
350	3410	59.51572749	0.173648178	-0.156434465	-0.01358228	0.20931322	
351	3420	59.69026042	-1.2249E-15	-0.309016994	1.8926E-16	0.20931322	
352	3430	59.86479334	-0.17364818	-0.4539905	0.03941731	0.208259233	
353	3440	60.03932627	-0.34202014	-0.587785252	0.1005172	0.206533593	
354	3450	60.21385919	-0.5	-0.707106781	0.1767767	0.20458511	
355	3460	60.38839212	-0.64278761	-0.809016994	0.26001305	0.202880089	
356	3470	60.56292504	-0.76604444	-0.891006524	0.3412753	0.201851442	
357	3480	60.73745797	-0.8660254	-0.951056516	0.41181955	0.201851442	
358	3490	60.91199089	-0.93969262	-0.987688341	0.46406172	0.203113264	
359	3500	61.08652382	-0.98480775	-1	0.49240388	0.205725508	
360	3510	61.26105675	-1	-0.987688341	0.49384417	0.209622474	
361	3520	61.43558967	-0.98480775	-0.951056516	0.46830392	0.214591257	
362	3530	61.6101226	-0.93969262	-0.891006524	0.41863613	0.220294878	
363	3540	61.78465552	-0.8660254	-0.809016994	0.35031463	0.226308943	
364	3550	61.95918845	-0.76604444	-0.707106781	0.27083761	0.232167825	
365	3560	62.13372137	-0.64278761	-0.587785252	0.18891054	0.23741534	
366	3570	62.3082543	-0.5	-0.4539905	0.11349762	0.241654402	
367	3580	62.48278722	-0.34202014	-0.309016994	0.05284502	0.244590236	
368	3590	62.65732015	-0.17364818	-0.156434465	0.01358228	0.246062447	
369	3600	62.83185307	-2.4503E-15	-2.20527E-15	2.7018E-30	0.246062447	