1

PM-1 3 1/2"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395 rue Wellington
Ottawa (Ontario)
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

**Investigation and Implementation of a
Computationally-efficient Algorithm for
Cyclic Spectral Analysis**

by

Caroline Tom, B.Eng.

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of

Master of Engineering

Faculty of Engineering
Department of Electronics
Carleton University
Ottawa, Ontario

March 1995

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395 rue Wellington
Ottawa (Ontario)
K1A 0N4

Author: [illegible]

Author: [illegible]

THE AUTHOR HAS GRANTED AN
IRREVOCABLE NON-EXCLUSIVE
LICENCE ALLOWING THE NATIONAL
LIBRARY OF CANADA TO
REPRODUCE, LOAN, DISTRIBUTE OR
SELL COPIES OF HIS/HER THESIS BY
ANY MEANS AND IN ANY FORM OR
FORMAT, MAKING THIS THESIS
AVAILABLE TO INTERESTED
PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE
IRREVOCABLE ET NON EXCLUSIVE
PERMETTANT A LA BIBLIOTHEQUE
NATIONALE DU CANADA DE
REPRODUIRE, PRETER, DISTRIBUER
OU VENDRE DES COPIES DE SA
THESE DE QUELQUE MANIERE ET
SOUS QUELQUE FORME QUE CE SOIT
POUR METTRE DES EXEMPLAIRES DE
CETTE THESE A LA DISPOSITION DES
PERSONNE INTERESSEES

THE AUTHOR RETAINS OWNERSHIP
OF THE COPYRIGHT IN HIS/HER
THESIS. NEITHER THE THESIS NOR
SUBSTANTIAL EXTRACTS FROM IT
MAY BE PRINTED OR OTHERWISE
REPRODUCED WITHOUT HIS/HER
PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE
DU DROIT D'AUTEUR QUI PROTEGE
SA THESE. NI LA THESE NI DES
EXTRAITS SUBSTANTIELS DE CELLE-
CI NE DOIVENT ETRE IMPRIMES OU
AUTREMENT REPRODUITS SANS SON
AUTORISATION.

ISBN 0-612-03037-7

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

U·M·I

SUBJECT TERM

SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0906
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communications 0459
Theater 0465

Psychology 0525
Reading 0535
Religion 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Theory of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
General 0289
Ancient 0290
Linguistics 0291
Modern 0401
Literature 0294
General 0295
Classical 0297
Comparative 0298
Medieval 0316
Modern 0591
African 0305
American 0352
Asian 0355
Canadian (English) 0593
Canadian (French) 0711
English 0312
Germanic 0315
Latin American 0313
Middle Eastern 0314
Romance 0370
Slavic and East European 0372

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion 0318
General 0321
Biblical Studies 0319
Clergy 0320
History of 0322
Philosophy of 0469
Theology 0323

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
Archaeology 0326
Cultural 0327
Physical 0310
Business Administration 0272
General 0770
Accounting 0454
Banking 0338
Management 0385
Marketing 0501
Canadian Studies 0503
Economics 0505
General 0508
Agricultural 0509
Commerce Business 0510
Finance 0511
History 0358
Labor 0366
Theory 0351
Folklore 0578
Geography 0383
Gerontology 0386
History 0485
General 0537

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science 0615
General 0616
International Law and Relations 0617
Public Administration 0814
Recreation 0452
Social Work 0626
Sociology 0627
General 0938
Criminology and Penology 0631
Ethnic and Racial Studies 0628
Individual and Family Studies 0629
Industrial and Labor Relations 0630
Public and Social Welfare 0700
Social Structure and Development 0344
Theory and Methods 0709
Transportation 0999
Urban and Regional Planning 0453
Women's Studies 0537

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0272
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Elementary 0277
Finance 0519
Guidance and Counseling 0680
Health 0745
Higher 0520
History of 0278
Home Economics 0521
Industrial 0279
Language and Literature 0280
Mathematics 0522
Music 0998
Philosophy of 0523
Physical 0523

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
General 0285
Agronomy 0475
Animal Culture and Nutrition 0476
Animal Pathology 0359
Food Science and Technology 0478
Forestry and Wildlife 0479
Plant Culture 0480
Plant Pathology 0817
Plant Physiology 0777
Range Management 0746
Wood Technology 0306
Biology 0287
General 0308
Anatomy 0309
Biostatistics 0370
Botany 0329
Cell 0353
Ecology 0369
Entomology 0793
Genetics 0410
Immunology 0307
Microbiology 0317
Molecular 0416
Neuroscience 0433
Oceanography 0821
Physiology 0778
Radiation 0472
Veterinary Science 0786
Zoology 0780
Biophysics 0425
General 0998
Medical 0998

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
General 0300
Audiology 0992
Chemotherapy 0567
Dentistry 0350
Education 0769
Hospital Management 0758
Human Development 0982
Immunology 0564
Medicine and Surgery 0347
Mental Health 0569
Nursing 0570
Nutrition 0380
Obstetrics and Gynecology 0354
Occupational Health and Therapy 0381
Ophthalmology 0571
Pathology 0419
Pharmacology 0572
Pharmacy 0382
Physical Therapy 0573
Public Health 0574
Radiology 0575
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry 0485
General 0749
Agricultural 0486
Analytical 0487
Biochemistry 0488
Inorganic 0738
Nuclear 0490
Organic 0491
Pharmaceutical 0494
Physical 0495
Polymer 0754
Radiation 0405
Mathematics 0605
Physics 0986
General 0606
Acoustics 0608
Astronomy and Astrophysics 0748
Atmospheric Science 0607
Atomic 0798
Electronics and Electricity 0759
Elementary Particles and High Energy 0609
Fluid and Plasma 0610
Molecular 0752
Nuclear 0756
Optics 0611
Radiation 0463
Solid State 0463
Statistics 0346
Applied Sciences 0984

Engineering 0537
General 0538
Aerospace 0539
Agricultural 0540
Automotive 0541
Biomedical 0542
Chemical 0543
Civil 0544
Electronics and Electrical 0348
Heat and Thermodynamics 0545
Hydraulic 0546
Industrial 0547
Marine 0794
Materials Science 0548
Mechanical 0743
Metallurgy 0551
Mining 0552
Nuclear 0549
Packaging 0765
Petroleum 0554
Sanitary and Municipal System Science 0790
Geotechnology 0796
Operations Research 0795
Plastics Technology 0994
Textile Technology 0621

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0453
Social 0621

Carleton University
Ottawa, Canada K1S 5B6

Thesis contains black & white
photographs &/or explanatory tables
which when microfilmed may lose their
significance. The hardcopy of the
thesis is available upon request from
Carleton University Library.

The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis

**Investigation and Implementation of a
Computationally-Efficient Algorithm for
Cyclic Spectral Analysis**

submitted by Caroline Tom, B.Eng

in partial fulfilment of the requirements for
the degree of Master of Engineering

Thesis Supervisor

Chair, Department of Electronics

Carleton University

March 1995

ABSTRACT

The suitability of cyclic spectral analysis for signal detection and identification is investigated. An overview of the theoretical background leading to the cyclic autocorrelation and cyclic spectral density equations is presented. Brute force and computationally-efficient approaches for estimating the cyclic spectrum are examined. Of particular interest in this thesis is the computationally-efficient FFT accumulation method (FAM) for computing cyclic spectrum estimates. Key features of the algorithm which reduce the number of computations are demonstrated. A cyclic spectrum analyser that processes digitally-modulated signals, computes the cyclic spectrum estimates, and displays the results is designed, implemented, and tested. The FAM algorithm is implemented on a DSP board containing a TMS320c51 processor. Cyclic spectra for phase-shift-keyed (PSK), frequency-hopped frequency-shift-keyed (FH/FSK), FH/PSK, and burst PSK waveforms are computed and displayed. Theoretical and experimental results are compared and discussed.

ACKNOWLEDGEMENTS

The author wishes to acknowledge her supervisor, Mr. W.R. Seed, for his encouragement and guidance throughout this thesis project. The constructive and timely suggestions offered in many discussions have brought forth a profitable exchange of ideas. His dedication and efforts are truly appreciated.

Sincere thanks is also expressed to her supervisor, Professor J.S. Wight, for his direction and thoughtful advice. His enthusiasm and interest in the progress of this thesis and in its final outcome has been inspiring.

The MILSATCOM Group of the Space Systems and Technology Section at the Defence Research Establishment Ottawa provided the facilities and necessary equipment for the completion of the thesis work. The valuable contributions of Mr. R. Burrill and Mr. P. Richer to facilitate the hardware realization is recognized.

The support of the Prototype Design Office at the Communications Research Centre in the layout and fabrication of the data collection board is acknowledged.

The author is also grateful to the Department of National Defence for granting her an Educational Leave of Absence in order to complete this thesis. The support of the Director Civilian Personnel (Materiel) 7 (DCP(Mat) 7) in the administrative matters of the Educational Leave is much appreciated.

TABLE OF CONTENTS

	Page
ACCEPTANCE SHEET	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS	v
LIST OF SYMBOLS	viii
LIST OF ABBREVIATIONS	xi
LIST OF FIGURES	xii
LIST OF TABLES	xv
1.0 Introduction	
1.1 Introduction	1
1.2 Thesis Objectives	2
1.3 Thesis Outline	3
2.0 Cyclic Spectral Analysis Concepts	
2.1 Introduction	5
2.2 Spectral Representation of Signals	5
2.2.1 Spectral Content	5
2.2.2 Fourier Transform Properties	6
2.2.3 Power Spectral Density	7
2.3 Statistical Representation of Signals	8
2.3.1 Statistical Properties	8
2.3.2 Ergodicity	9
2.4 Periodicity in Signals	10
2.4.1 Cyclic Autocorrelation	11
2.4.2 Cyclic Spectral Density	12
2.5 Cyclic Spectrum	14
2.6 Cyclic Spectrum Equations for Common Signal Waveforms..	16
3.0 Cyclic Spectrum Estimation	
3.1 Introduction	25
3.2 Finite-time Cyclic Spectrum Estimate	26
3.3 Discrete-time Cyclic Spectrum Estimate	30
3.4 Cyclic Spectrum Estimation Algorithms	32

3.4.1 Brute Force Method	32
3.4.2 FFT Accumulation Method (FAM)	36
3.4.2.1 Partitioning the Bifrequency Plane	37
3.4.2.2 Product Sequence Decimation	40
3.4.2.3 Computing a Block of Cyclic Spectrum Estimates	41
3.4.2.4 FAM Process (Summary)	46
4.0 Cyclic Spectrum Analyzer Design Considerations and Implementation	
4.1 Introduction	48
4.2 Hardware Implementation	50
4.2.1 Signal Generation	50
4.2.2 Receiver/Downconverter	50
4.2.3 Baseband Processing/Digitization	51
4.2.3.1 A/D Converter	52
4.2.3.2 LPF and Buffer Amplifier	56
4.2.4 Data Collection and Timing	60
4.2.4.1 Data Latch and FIFO Memory	60
4.2.4.2 Clock Source Selection	61
4.2.4.3 Timing Signals Generation	61
4.2.4.4 Timing Signals Control	67
4.2.5 FIFO/DSP Interface	68
4.2.5.1 DSP Board	68
4.2.5.2 Data Transfer - Read Signal Generation ..	69
4.2.5.3 Status Flags Monitoring	71
4.2.6 DSP/PC Handshaking	73
4.3 Software Implementation	74
4.3.1 FAM Computation Overview	74
4.3.1.1 Data Pre-processing	76
4.3.1.2 Frequency Components Computation	77
4.3.1.3 Cyclic Spectrum Estimates Computation ..	81
4.3.2 Hardware/User Interface	85
4.3.2.1 Variables and Indices Transfer	87
4.3.2.2 Window Effects Compensation	89
4.3.3 Graphical Analysis	90
5.0 FAM Algorithm Testing and Results	
5.1 Introduction	91
5.2 Test Procedure	91
5.3 Experimental Results and Discussion	93
5.3.1 Verification of FAM Algorithm Implementation ...	93
5.3.2 Cyclic Spectra for Common Digital Modulations ..	95
5.3.2.1 BPSK Waveform	95
5.3.2.2 QPSK Waveform	97

5.3.2.3 FH/MFSK Waveform	99
5.3.2.4 Burst PSK and FH/PSK Waveforms	100
6.0 Summary and Recommendations	
6.1 Introduction	103
6.2 Thesis Summary	103
6.3 FAM Algorithm Summary of Results	105
6.4 Areas for Future Work	106
REFERENCES	108
APPENDICES	
A A/D Board Circuit Diagram	110
B Timing Diagrams for the A/D Board	112
C Software Listings	119

LIST OF SYMBOLS

a	Quantization step
B	Bandwidth
C	Capacitor value in LPF/Buffer amplifier circuit
C_{ext}	External capacitance for 74LS221
$\frac{E_b}{N_o}$	Bit-energy-to-noise-spectral-density ratio
f	Temporal frequency
f_g	Sampling frequency of product sequence
f_{res}	Frequency resolution (of Fourier transform)
f_s	Sampling frequency of input sequence, $x[n]$
G	Gain of buffer amplifier
J	Number of discrete frequency bins averaged
k	Discrete temporal frequency
k_{res}	Frequency resolution of discrete-time Fourier transform
L	Decimation or overlap parameter
m	Time hop number
M	Total number of hops
mV	Millivolts
mW	Milliwatts
$m_x(t)$	Mean or Expected value
M_x^a	Fourier coefficient
n	Sample index
N	Length of sequence $x[n]$
N'	Length of short-time Fourier transform

N_p	Noise power level
N_q	Quantization noise power
$p_{a,b}[m]$	Product sequence for channel pair region (a,b)
P	Length of product sequence ($P=M$)
r	Number of bits on ADC
$Rect(t/T)$	Rectangular function
R_b	Data bit rate
R_{ext}	External resistance for 74LS221
R_{in}	Input resistance for AD9005A
R_{term}	Terminated resistance for LPF/Buffer amplifier circuit
R_1, R_2, R_3, R_4	Resistor values for LPF/Buffer amplifier circuit
$R_x(t_1, t_2)$	Autocorrelation function
$R_x(\tau)$	Autocorrelation function for cyclostationary process
$R_x^a(\tau)$	Cyclic autocorrelation function
S	Signal power level
$Sinc(x)$	Sinc function, $Sinc(x) = \frac{\sin(\pi x)}{(\pi x)}$
$S_x(f)$	Power spectral density
$S_{x_T}(f)$	Periodogram
$S_x^a(f)$	Cyclic spectral density or cyclic spectrum
$S_{x_T}^a(f)$	Cyclic periodogram
$S_{x_T}^a(t, f)_{\Delta t}$	Time-smoothed cyclic periodogram
$S_{x_{\Delta t}}^a(t, f)_{\Delta f} \frac{1}{T}$	Frequency-smoothed cyclic periodogram
$S_{x_N}^Y(n, k)$	Discrete cyclic periodogram
$S_{x_N}^Y(n, k)_N$	Discrete time-smoothed cyclic periodogram
$S_{x_N}^Y(n, k) \frac{1}{J}$	Discrete frequency-smoothed cyclic periodogram
t_{pulse}	Pulse width for 74LS221

t_{res}	Temporal resolution (of Fourier transform)
V	Volts
$V_{in_{AD}}$	Input voltage of AD9005A ADC
$V_{o_{AD844}}$	Output voltage of AD844 LPF/Buffer amplifier
W	Watts
$x(t)$	Continuous-time signal
$X(f)$	Fourier transform of $x(t)$
$X_N(n,k)$	Discrete finite-length (N') Fourier transform
$X_T(t,f)$	Time-dependent finite-time Fourier transform
Z	Number of cyclic spectrum estimates on bifrequency plane
α	Cycle frequency
β	6.0dB bandwidth in terms of frequency bins
γ	Discrete cycle frequency
Δf	Frequency resolution of cyclic spectrum estimate
Δk	Frequency resolution of discrete cyclic spectrum estimate
Δt	Finite length of signal, $x(t)$
$\Delta \alpha$	Cycle frequency resolution of cyclic spectrum estimate
$\Delta \gamma$	Cycle frequency resolution of discrete-time cyclic spectrum estimate
ϵ	Quantization noise error
$\Lambda(t/T)$	Triangular function

LIST OF ABBREVIATIONS

A/D	Analog-to-Digital
ADC	Analog-to-Digital Converter
ASCII	American Standard Code for Information Interchange
AWGN	Additive White Gaussian Noise
BPSK	Binary-Phase-Shift-Keying
Burst PSK	Burst Phase-Shift-Keying
CW	Continuous-wave
dB	Decibels
dBm	Decibels relative to mW
dBq	Decibels relative to a Quantization Unit
DIF	Decimation-In-Frequency
DSP	Digital Signal Processor
FAM	FFT Accumulation Method
FFT	Fast Fourier Transform
FH	Frequency-Hopped
FH/FSK	Frequency-Hopped/Frequency-Shift-Keying
FH/MFSK	Frequency-Hopped/M-ary Frequency-Shift-Keying
FIFO	First-In-First-Out
IF	Intermediate Frequency
kbps	Kilobits-per-second
kHz	Kilohertz
ksps	Kilosymbols-per-second
kΩ	Kilo-ohms
LPF	Low Pass Filter
Mbps	Megabits-per-second
MHz	Megahertz
ms	Milliseconds
nF	Nanofarads
ns	Nanoseconds
pF	Picofarads
PSK	Phase-Shift-Keying
QPSK	Quaternary-Phase-Shift-Keying
qu	Quantization Unit
SNR	Signal-to-Noise Ratio
RAM	Random Access Memory
RF	Radio Frequency
Ω	Ohms
μs	Microseconds

LIST OF FIGURES

Chapter I:

- 2.1 (a) Bifrequency Plane (b) Region of Support for Lowpass Signal (c) Region of Support for Bandpass Signal (d) Region of Support for Highpass Signal

Chapter III

- 3.1 Cyclic spectrum estimation via time-smoothing (a) Short-time Fourier transforms over Δt (b) Sequence of frequency products from short-time Fourier transforms
- 3.2 Cyclic spectrum estimation via frequency-smoothing
- 3.3 Cyclic spectrum estimate (a) For a particular (k_0, γ_0) (b) For all possible combinations (k_0, γ_0) for $N'=8$
- 3.4 Outline of channel pair regions when $k_{res}=f_r/N'$
- 3.5 Channel pair region description (a) Frequency components separated by γ_0 (b) γ dimension (c) k dimension (d) Channel pair region dimensions
- 3.6 Product window bandwidth in γ of cyclic spectrum estimate sample
- 3.7 Resolving γ -bandwidth into P components
- 3.8 Gaps in bifrequency plane as a result of considering estimates $\pm k_{res}/2$ from the channel pair region centre
- 3.9 Overlapping channel pair regions (a) with Hamming window (b) Channel pair region covered by estimates $\pm k_{res}/2$ from centre

Chapter IV

- 4.1 Experimental setup
- 4.2 Signal generation and receiver block diagram
- 4.3 Baseband I/Q A/D Board

- 4.4 A/D voltage, quantization units, and dynamic range scales for positive signals
- 4.5 Signal dynamic range mapped into ADC dynamic range
- 4.6 LPF/Buffer amplifier circuit
- 4.7 Final configuration of LPF/Buffer amplifier circuit
- 4.8 Clock source selection
- 4.9 Timing diagram for digitization and storage operation
- 4.10 ENCODE and CLKEN signals generation
- 4.11 FIFO Write signal generation
- 4.12 Timing signals control
- 4.13 Timing diagram for READ operation
- 4.14 FIFO Read signals and buffer enable
- 4.15 Timing diagram for reading FIFO flags
- 4.16 Hardware for receiver/downconverter and A/D board
- 4.17 Flowchart of assembler program
- 4.18 DIF Radix-2 FFT algorithm (a) Flow diagram for 8-pt FFT (b) Fortran subroutine used to derive assembler code
- 4.19 Frequency bins normalization
- 4.20 Resolving cyclic spectrum components in the channel pair region
- 4.21 Channel pair regions for positive γ (a) Order in which channel pair regions are computed (b) Order in which channel pair regions are stored in memory
- 4.22 Flowchart of hardware/user interface program

- 4.23 (a) Bifrequency plane partitioned into "slices" with non-overlapping channel pair regions (b) Portion of one "slice" with overlapping channel pair regions (data taper with Hamming window)
- 4.24 Frequency product window

Chapter V

- 5.1 Cyclic spectrum for two CW signals (a) CW signals in separate frequency bins (b) Separate frequency bins - front view (c) Separate frequency bins - side view (d) CW signals in same frequency bin (e) Same frequency bin - front view (f) Same frequency bin - side view
- 5.2 Cyclic spectrum for a BPSK signal in AWGN (a) $E_b/N_o > 30\text{dB}$, $R_b = 128\text{kbps}$ (b) $E_b/N_o = 10\text{dB}$, $R_b = 128\text{kbps}$ (c) $E_b/N_o = 0\text{dB}$, $R_b = 128\text{kbps}$ (d) $E_b/N_o > 30\text{dB}$, $R_b = 1.024\text{Mbps}$ (e) $E_b/N_o = 10\text{dB}$, $R_b = 1.024\text{Mbps}$ (f) $E_b/N_o = 0\text{dB}$, $R_b = 1.024\text{Mbps}$
- 5.3 Cyclic spectrum for two BPSK signals (a) BPSK signal 1 alone (b) BPSK signal 2 alone (c) Combined BPSK signals
- 5.4 Cyclic spectrum for BPSK and QPSK waveforms (a) Real BPSK signal, $R_b = 1.024\text{Mbps}$ (b) Real QPSK signal, $R_s = 512\text{kpsps}$
- 5.5 Cyclic spectrum for a FH/MFSK signal (a) Entire cyclic spectrum for positive α (b) Partial spectrum for positive α (c) Profile of cyclic spectrum for FH/MFSK signal
- 5.6 Cyclic spectrum for a burst PSK waveform (a) Direct view of cyclic spectrum for positive α (b) Side view of cyclic spectrum for positive α
- 5.7 Cyclic spectrum for a FH/PSK waveform (a) Direct view of cyclic spectrum for positive α (b) Side view of cyclic spectrum for positive α

LIST OF TABLES

Chapter III

- 3.1 Number of complex multiplies using the time-smoothed brute force method**
- 3.2 Number of complex multiplies using the frequency-smoothed brute force method**
- 3.3 Number of complex multiplies using the FAM algorithm**

Chapter V

- 5.1 Test scenarios for FAM algorithm**

CHAPTER I

INTRODUCTION

1.1 Introduction

The recent development of the theory for cyclostationary (cyclic) spectral analysis has generated much interest in its application in traditional signal processing tasks [1-3]. In communications systems, these tasks may include signal detection and identification or parameter estimation. The underlying principle of cyclic spectral analysis involves the treatment of signals under examination as cyclostationary rather than conventional stationary processes. A cyclostationary signal has statistical parameters that vary periodically in time. These parameters provide additional information about the signal which may be extracted and used to improve system performance.

In digital communications, operations such as sampling, modulation of either amplitude, phase, or frequency, or features such as sinusoidal carriers of signals represent periodic transformations unique to each signal. These periodic attributes suggest that a cyclostationary model would be a more suitable representation of the signal. By using cyclic spectral analysis, these attributes are potentially revealed in the form of non-zero correlation where the separation of frequency components is related to the period of cyclostationarity.

Cyclic spectral analysis is described by the cyclic autocorrelation and cyclic spectral density equations. These equations are extensions of the conventional

autocorrelation and power spectral density equations used in stationary signal analysis. The cyclic autocorrelation function incorporates aspects of square law detection and delay-multiply feature detection. The cyclic spectral density (cyclic spectrum) is the frequency representation of the cyclic autocorrelation. It consists of two frequency variables, temporal and cyclic, which can be displayed on a bifrequency plane. The bifrequency plane allows a range of frequencies to be examined and is convenient for analysis of multiple signals within the same band.

The major drawback of employing cyclic spectral analysis techniques is the processing requirement, making it difficult to use in applications where a near real-time response is required. As a result, the development and study of cyclic spectral analysis concepts has been paralleled by a comparable effort in developing computationally-efficient algorithms for computing the cyclic spectrum [4,5]. Cyclic spectrum estimation algorithms are based on time-smoothing or frequency-smoothing approaches analogous to power spectrum estimation techniques [3] and are introduced in [1,4]. Of particular interest in this thesis is the FFT Accumulation Method (FAM), first described in [5]. It is a time-smoothing algorithm which takes advantage of the efficiency of the Fast Fourier Transform (FFT) to compute cyclic spectrum estimates over a large portion of the bifrequency plane. Decimation and windowing operations also contribute to the computational efficiency and are examined in this thesis.

1.2 Thesis Objectives

The objective of this thesis is to analyse, implement and evaluate a computationally-efficient algorithm for cyclic spectral analysis. In addition, an

understanding of system and hardware design considerations with respect to RF signal processing, data collection, and digital data processing is obtained. The manner in which these goals are achieved is described in the following paragraph.

An overview of the theory which forms the basis of cyclic spectral analysis is first presented. This leads to the development of equations for the cyclic autocorrelation and the cyclic spectral density. An understanding of concepts relating to techniques for estimating the cyclic spectrum is then established. An analysis of these techniques is carried out to illustrate the extensive computational requirements. A detailed examination of the FAM algorithm is subsequently presented to gain an appreciation of the source of computational savings. The FAM algorithm is implemented on a DSP board to compute the cyclic spectrum. The algorithm is tested using digitally-modulated waveforms, with and without additive white gaussian noise (AWGN). Additional verification is also provided with simulated input data. Upon examination of the output, the suitability of cyclic spectrum estimation for signal detection and identification is assessed.

1.3 Thesis Outline

A review of the equations and properties used in conventional spectral analysis is presented in Chapter 2. The concept of periodicity in signals is described in terms of its effect in the frequency domain. The equation which describes this effect leads to the introduction of equations for cyclic spectral analysis. The equations are shown to be generalizations or extensions of those used in conventional spectral analysis.

Chapter 3 introduces equations for time-smoothed and frequency-smoothed

cyclic spectrum estimates. An example of a brute force implementation for both methods is presented to highlight the extensive computational requirements. Subsequently, for the time-smoothed estimate, the computationally-efficient FAM algorithm is described. Key points include the use of decimation and an FFT-based routine to reduce the number of computations.

Chapter 4 describes the implementation of a cyclic spectrum analyser based on the computationally-efficient algorithm introduced in Chapter 3. Key components include signal generation, received signal processing, baseband signal processing and data collection, cyclic spectrum estimates computations, and hardware/user interface.

Experimental results obtained from the cyclic spectrum analyser are presented in Chapter 5. Cyclic spectra are shown for common digital modulation schemes over a range of bit-energy-to-noise-spectral-density ratios (E_b/N_o) and data rates (R_b). Experimental and theoretical results are compared and discussed.

Chapter 6 provides a summary of the theoretical and implementation aspects of the thesis. General conclusions are given based on the cyclic spectrum analyser performance for the parameters implemented. Suggestions for future areas of study relating to this thesis project are included.

CHAPTER II

CYCLIC SPECTRAL ANALYSIS CONCEPTS

2.1 Introduction

The development of cyclic spectral analysis equations can be interpreted as an extension of conventional spectral analysis properties and concepts. This chapter reviews the conventional equations for the statistical and spectral representations of signals and their classifications. The concept of periodicity in signals is examined. A common characteristic is observed between the equation that reflects periodicity and the conventional autocorrelation function for cyclostationary processes. This characteristic leads to the definition of the cyclic autocorrelation and cyclic spectral density equations.

2.2 Spectral Representation of Signals

2.2.1 Spectral Content

A key aspect of signal analysis involves the representation of time signals or processes as a weighted sum of exponential or sinusoidal components to observe characteristics which discern one signal from another. The spectral content, also known as the frequency spectrum, of a signal displays the amplitude of each exponential component as a function of the sinusoidal frequency. It is obtained from the Fourier transform of the corresponding time signal, $x(t)$. The Fourier transform of $x(t)$ is defined as [6],

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-i2\pi ft} dt \quad (1)$$

where f is the frequency variable and $i = (-1)^{1/2}$. The functions $x(t)$ and $X(f)$, together, form a Fourier transform pair denoted as follows,

$$x(t) \xleftrightarrow{\mathcal{F}} X(f) \quad (2)$$

2.2.2 Fourier Transform Properties

It is useful, at this point, to review four properties of the Fourier transform that relate time-domain operations to those in the frequency domain. These four properties facilitate the introduction of cyclic spectral analysis principles as they relate to those for conventional spectral analysis.

The Time-Shifting Property reveals that if a signal, $x(t)$, is shifted in time by some delay, T_d , then its Fourier transform, $X(f)$, is multiplied by an exponential that is a function of the delay [6],

$$x(t-T_d) \xleftrightarrow{\mathcal{F}} e^{-i2\pi f T_d} X(f) \quad (3)$$

This can be verified by replacing t by $(t-T_d)$ in (1). The exponential term resulting from the substitution allows the evaluation of the frequency spectrum at different instances in time.

If a signal, $x(t)$, is multiplied with an exponential, $e^{i2\pi f_o t}$, the Fourier transform of the new signal is the Fourier transform of $x(t)$, frequency-shifted by a factor of f_o . This is referred to as the Frequency-Shifting Property of the Fourier transform [6],

$$e^{i2\pi f_o t} x(t) \xleftrightarrow{\mathcal{F}} X(f-f_o) \quad (4)$$

Time-Shifting and Frequency-Shifting are dual properties; a delay or shift in one domain results in a multiplication by an exponential in the other domain.

When a signal, $x(t)$, is time-reversed (i.e. $x(-t)$), the effect on the Fourier transform, $X(f)$, is referred to as the Time-Reversal Property of Fourier transforms and is given by [6]

$$x(-t) \xrightarrow{\mathcal{F}} X^*(f) \quad (5)$$

where the asterisk denotes the complex conjugate. This can be verified by substituting $-t$ for t in (1).

If two signals, $x_1(t)$ and $x_2(t)$, with Fourier transforms $X_1(f)$ and $X_2(f)$ respectively, are convolved, the Fourier transform of the convolved signal is obtained by multiplying $X_1(f)$ and $X_2(f)$. This is referred to as the Convolution Theorem [6],

$$x_1(t) \otimes x_2(t) \xrightarrow{\mathcal{F}} X_1(f)X_2(f) \quad (6a)$$

where $\otimes$ represents the convolution operation. Furthermore, if $x_2(t)$ is time-reversed, the following Fourier transform pair is obtained by using (5),

$$x_1(t) \otimes x_2(-t) \xrightarrow{\mathcal{F}} X_1(f)X_2^*(f) \quad (6b)$$

2.2.3 Power Spectral Density

Another practical representation for signal analysis is the power spectral density. As the name implies, the power spectral density reveals the frequency distribution of signal power and is defined [7] for a signal, $x(t)$, as

$$S_x(f) \triangleq \lim_{T \rightarrow \infty} \frac{|X_T(f)|^2}{T} \quad (7)$$

where $X_T(f)$ is the finite-time Fourier transform. In the limit, $X_T(f)$ approaches $X(f)$ defined in (1). The squared magnitude of $X_T(f)$ normalized by T is defined as the Periodogram, denoted $S_x(f)$, and was initially developed to search for periodicities in signals [1,7].

2.3 Statistical Representation of Signals

2.3.1 Statistical Properties

An alternate description of random signals or processes is based on properties obtained from statistical or ensemble averages. The statistical properties used include the Expected Value or Mean, and the Autocorrelation, denoted $m_x(t)$ and $R_x(t_1, t_2)$ respectively. They are defined [8] as

$$m_x(t) \triangleq \mathcal{E}[x(t)] = \int_{-\infty}^{\infty} x g(x) dx \quad (8)$$

$$R_x(t_1, t_2) = \mathcal{E}[x(t_1)x(t_2)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 g(x_1, x_2; t_1, t_2) dx_1 dx_2 \quad (9)$$

where $\mathcal{E}[\]$ represents the expected value or ensemble average.

The characteristics of $m_x(t)$ and $R_x(t_1, t_2)$ provide necessary and sufficient conditions for certain classifications of signals. A signal, $x(t)$, is classified as being Stationary if its statistical properties are invariant to a shift in time [8]. Furthermore, it is called Wide-Sense Stationary if the mean is constant and the autocorrelation is dependent only on the time difference, $\tau = (t_1 - t_2)$. Hence,

$$m_x(t) = \mathcal{E}[x(t)] = C \quad (10)$$

$$R_x(t_1, t_2) = \mathcal{E}[x(t_1)x(t_2)] = \mathcal{E}[x(t)x(t-\tau)] = R_x(\tau) \quad (11)$$

The autocorrelation function can also be written in symmetrical form with a change of variable,

$$R_x(\tau) = \mathcal{E} \left[x\left(t + \frac{\tau}{2}\right) x\left(t - \frac{\tau}{2}\right) \right] \quad (12)$$

For a stationary process, it can be shown [7] that the link between the spectral and statistical representations of signals is given by the Wiener relation as follows,

$$S_x(f) \triangleq \lim_{T \rightarrow \infty} \frac{|X_T(f)|^2}{T} = \mathcal{F}[R_x(\tau)] \quad (13)$$

A process with a time-varying mean and autocorrelation is classified as Non-stationary. Between the two types of processes are those which have periodically-varying mean and autocorrelation.

$$m_x(t) = m_x(t + nT_o) = \mathcal{E}[x(t + nT_o)] \quad (14)$$

$$R_x(t, \tau) = R_x(t + nT_o, \tau) = \mathcal{E} \left[x\left(t + nT_o + \frac{\tau}{2}\right) x\left(t + nT_o - \frac{\tau}{2}\right) \right] \quad (15)$$

A process satisfying these conditions is said to be Cyclostationary [8].

2.3.2 Ergodicity Property

In many practical applications, a random process may be represented by only one sample function, $x(t)$, from which the ensemble averages are derived. Obtaining the statistical properties from this one sample function is possible if the random process is assumed to be Ergodic [1,7,9]. If a random process is ergodic, an ensemble average can be determined by a corresponding time-average. For a stationary process, this implies,

$$m_x(t) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) dt \quad (16)$$

$$R_x(\tau) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t + \frac{\tau}{2}) x(t - \frac{\tau}{2}) dt \quad (17)$$

Since the mean and autocorrelation functions which characterize a cyclostationary process depend on time, ergodicity cannot be applied to simplify the analysis. However, it is shown in [1,3] that by applying the technique of synchronized averaging, the concept of "cycloergodicity" is developed and results in the replacement of the ensemble averages with the following,

$$m_x(t+nT_o) = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N x(t+nT_o) \quad (18)$$

$$R_x(t+nT_o, \tau) = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N x(t+nT_o + \frac{\tau}{2}) x(t+nT_o - \frac{\tau}{2}) \quad (19)$$

2.4 Periodicity in signals

If a signal, $x(t)$, is periodic with period T_o in time, it can be represented by a Fourier series expansion [6] as follows,

$$x(t) = \sum_{n=-\infty}^{\infty} M_x^{n/T_o} e^{i2\pi n t / T_o} \quad (20)$$

If $\alpha = \frac{n}{T_o}$, the Fourier coefficient, M_x^α , represents the spectral component in the power spectral density at $f = \alpha$ [2] and can be obtained as follows,

$$M_x^\alpha = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) e^{-i2\pi \alpha t} dt \neq 0 \quad (21)$$

A signal with non-zero Fourier coefficients is termed a first-order periodic signal. There may be signals with periodic parameters that are not reflected as non-zero spectral components as specified in (21). However, by applying an N^{th} order non-linear transformation to the signal [1,2], it may be possible to convert these

hidden parameters into first-order periodicities. This would provide further information about the signal to facilitate its identification or detection. The focus of this thesis is on signals whose periodicities can be observed as being first-order following a quadratic transformation of the signal. As the transformed signal of interest is of the second-order, periodicities that are revealed are appropriately termed second-order periodicities.

2.4.1 Cyclic Autocorrelation

In order to convert a second-order periodicity into a first-order periodicity, an appropriate second-order transformation of the signal is required. A common configuration used in current receiver/detection schemes is the delay-multiply circuit. A received signal is multiplied with a delayed version of itself, and then integrated over the time interval for which the product exists. For communications signals, periodic characteristics, such as the symbol rate, can be obtained from the reciprocal of the delay which results in a non-zero value when integrated over time. This second-order transformation,

$$y(t) = x(t)x(t-\tau)$$

is known as the delay-product of $x(t)$ and is considered for examining second-order periodicities. If the delay-product of $x(t)$, for some delay, τ , generates a non-zero spectral component at some frequency, α , then $x(t)$ is considered to be second-order periodic. As a result,

$$\begin{aligned}
M_y^\alpha &\triangleq \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} y(t) e^{-i2\pi\alpha t} dt \neq 0 \\
&= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) x(t-\tau) e^{-i2\pi\alpha t} dt \neq 0
\end{aligned} \tag{22}$$

Rewriting $x(t)x(t-\tau)$ as a symmetric delay product and considering the case where $x(t)$ is a complex signal, (22) becomes,

$$R_x^\alpha(\tau) \triangleq M_y^\alpha = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t + \frac{\tau}{2}) x^*(t - \frac{\tau}{2}) e^{-i2\pi\alpha t} dt \tag{23}$$

It is observed that for $\alpha = 0$, (23) is the conventional autocorrelation of $x(t)$ which was defined for a stationary ergodic process in Section 2.3.2. For $\alpha \neq 0$, the exponential term, $e^{-i2\pi\alpha t}$, identifies the location of the spectral line, and hence, the periodic or cyclic feature of the delay-product of $x(t)$. Consequently, this provides another means for observing the symbol or data rates. The Fourier coefficient,

M_y^α , can be renamed the Cyclic Autocorrelation Function and is denoted $R_x^\alpha(\tau)$.

It is observed in [1,3] that (23) can also be interpreted as the Fourier coefficient in the Fourier series expansion of the periodic autocorrelation function for a cyclostationary random process (19). This interpretation suggests that the representation of a signal with second-order periodicity is directly related to equations for cyclostationary (cyclic) analysis.

2.4.2 Cyclic Spectral Density

As for conventional spectral analysis, it is useful to establish a link between time- and frequency-domain operations. The Wiener relation [12] established that the conventional autocorrelation function and the power spectral density were a

Fourier transform pair. The same relation can be extended to the cyclic autocorrelation function to define the Cyclic Spectral Density [1,2]. The modified Fourier transform pair generalizes the Wiener relation to include cyclostationary processes and is thus called the Cyclic Wiener Relation. Therefore,

$$\begin{aligned}
 S_x^\alpha(f) &= \mathcal{F}[R_x^\alpha(\tau)] \\
 &= \int_{-\infty}^{\infty} R_x^\alpha(\tau) e^{-i2\pi f\tau} d\tau \\
 &= \int_{-\infty}^{\infty} \left[\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t + \frac{\tau}{2}) x^*(t - \frac{\tau}{2}) e^{-i2\pi\alpha t} dt \right] e^{-i2\pi f\tau} d\tau \quad (24)
 \end{aligned}$$

Multiplying (24) by $e^{-i\pi\alpha(\frac{\tau}{2} - \frac{\tau}{2})} = e^0 = 1$, and separating $e^{-i2\pi\alpha t}$ into $e^{-i\pi\alpha t} e^{-i\pi\alpha t}$ gives,

$$S_x^\alpha(f) = \int_{-\infty}^{\infty} \left[\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t + \frac{\tau}{2}) e^{-i\pi\alpha(t + \frac{\tau}{2})} \right] \left[x(t - \frac{\tau}{2}) e^{i\pi\alpha(t - \frac{\tau}{2})} \right]^* dt \right] e^{-i2\pi f\tau} d\tau \quad (25)$$

Applying (4) and (6a) to (25),

$$S_x^\alpha(f) = \lim_{T \rightarrow \infty} \frac{X_T(f + \frac{\alpha}{2}) X_T^*(f - \frac{\alpha}{2})}{T} \quad (26)$$

where

$$\frac{X_T(f + \frac{\alpha}{2}) X_T^*(f - \frac{\alpha}{2})}{T}$$

is the Cyclic Periodogram denoted $S_{x_T}^\alpha(f)$. The cyclic periodogram is a generalization of the conventional periodogram (let $\alpha = 0$) used to estimate the power spectral density (Section 2.2.3). For a finite T , the cyclic periodogram represents an estimate of the ideal cyclic spectrum.

2.5 Cyclic Spectrum

The cyclic spectral density (or cyclic spectrum) consists of two variables, f and α . These variables are called temporal frequency and cycle frequency, respectively. They may be represented on a two-dimensional plane referred to as the (f, α) plane or the bifrequency plane as shown in Figure 2.1a. The magnitude of the cyclic component at coordinates (f_o, α_o) reflects the correlation of two spectral components separated by α_o , centred about f_o . Furthermore, for $\alpha = 0$, the cyclic spectral density reduces to the power spectral density.

The range of values of f and α for which $S_x^\alpha(f)$ exists, is defined as the Region of Support. In practice, a filter is used to limit the bandwidth of the received signal and hence defines boundaries for the region of support. The region of support for low-, high-, and bandpass filtered signals is illustrated in Figure 2.1(b)-(d).

Figure 2.1 (a) Bifrequency plane (b) Region of support for lowpass signal (c) Region of support for bandpass signal (d) Region of support for highpass signal

2.6 Cyclic Spectrum for Common Signal Waveforms

Example 1 - Consider $y(t)$ as the sum of two complex continuous-wave (CW) signals

$$y(t) = e^{-i2\pi f_1 t} + e^{-i2\pi f_2 t}$$

The cyclic spectrum of $y(t)$ is found by taking the Fourier transform of the cyclic autocorrelation function. The cyclic autocorrelation function of $y(t)$ is expressed as follows,

$$R_y^\alpha(\tau) \triangleq \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} y(t + \frac{\tau}{2}) y^*(t - \frac{\tau}{2}) e^{-i2\pi \alpha t} dt$$

Substituting for $y(t)$ and separating terms,

$$\begin{aligned} R_y^\alpha(\tau) = & e^{-i2\pi f_1 \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-i2\pi \alpha t} dt + e^{-i2\pi \left(\frac{f_1 + f_2}{2}\right) \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-i2\pi [\alpha - (f_1 - f_2)] t} dt \\ & + e^{-i2\pi \left(\frac{f_1 + f_2}{2}\right) \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-i2\pi [\alpha + (f_1 - f_2)] t} dt + e^{-i2\pi f_2 \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-i2\pi \alpha t} dt \end{aligned}$$

Hence,

$$R_y^\alpha(\tau) = \begin{cases} e^{-i2\pi f_1 \tau} + e^{-i2\pi f_2 \tau}, & \alpha = 0 \\ e^{-i2\pi \left(\frac{f_1 + f_2}{2}\right) \tau}, & \alpha = \pm(f_1 - f_2) \\ 0, & \text{otherwise} \end{cases}$$

Taking the Fourier transform of $R_y^\alpha(\tau)$ gives,

$$S_y^{\alpha}(f) = \mathcal{F}[R_y^{\alpha}(\tau)] = \begin{cases} \delta(f+f_1) + \delta(f+f_2), & \alpha = 0 \\ \delta\left(f + \left(\frac{f_1+f_2}{2}\right)\right), & \alpha = \pm(f_1-f_2) \\ 0, & \text{otherwise} \end{cases}$$

Example 2 - Consider $y(t)$ as a single hop of a Frequency-Hopped¹ M-ary Frequency-Shift Keyed (FH/MFSK) signal (complex)

$$y(t) = p(t-t_o) e^{-i(2\pi f_i t + \phi_o)}, \quad i = 0, 1, \dots, M-1$$

where

$$p(t) = \text{Rect}(t/T_h) = \begin{cases} 1, & |t| \leq T_h/2 \\ 0, & \text{otherwise} \end{cases}$$

is the rectangular function. Without loss of generality, let $t_o = 0$, $\phi_o = 0$ resulting in

$$y(t) = p(t) e^{-i2\pi f_i t}$$

The cyclic autocorrelation of $y(t)$ is

$$\begin{aligned} R_y^{\alpha}(\tau) &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} y(t + \frac{\tau}{2}) y^*(t - \frac{\tau}{2}) e^{-i2\pi \alpha t} dt \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2}) e^{-i2\pi f_i(t + \frac{\tau}{2})} p^*(t - \frac{\tau}{2}) e^{i2\pi f_i(t - \frac{\tau}{2})} e^{-i2\pi \alpha t} dt \\ &= e^{-i2\pi f_i \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2}) p^*(t - \frac{\tau}{2}) e^{-i2\pi \alpha t} dt \end{aligned}$$

¹Frequency-hopping is a spread spectrum modulation technique which is based on a periodic change in carrier frequency (hop rate).

First, consider the autocorrelation function of the rectangular pulse, $p(t)$,

$$\begin{aligned} R_p(\tau) &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2}) p^*(t - \frac{\tau}{2}) dt \\ &= \frac{1}{T} \Lambda(\tau/T_h) \end{aligned}$$

where $\Lambda(\tau/T_h)$ is the triangular function described by

$$\Lambda(\tau/T_h) = \begin{cases} 1 - \frac{|\tau|}{T_h} , & -T_h \leq \tau \leq T_h \\ 0 , & \text{otherwise} \end{cases}$$

As a result, the cyclic autocorrelation of $y(t)$ becomes

$$R_y^\alpha(\tau) = \begin{cases} \frac{1}{T} \Lambda(\tau/T_h) e^{-i2\pi f_i \tau} , & \alpha = 0 \\ 0 , & \text{otherwise} \end{cases}$$

Noting that the Fourier transform of the triangular function is the

$\text{Sinc}^2(x) \triangleq \left(\frac{\sin(\pi x)}{(\pi x)} \right)^2$ function,

$$\begin{aligned} S_y^\alpha(f) &= \mathcal{F}[R_y^\alpha(\tau)] \\ &= \begin{cases} \frac{1}{T} T_h^2 \text{Sinc}^2((f+f_i)T_h) , & \alpha = 0 \\ 0 , & \text{otherwise} \end{cases} \end{aligned}$$

As $T \rightarrow \infty$, $R_p(\tau) \rightarrow 0$ due to the normalization by the integration period, T .

Similarly, $R_y^\alpha(\tau)$ and $S_y^\alpha(f)$ also tend to zero for very large T as $y(t)$ is non-zero

over a finite interval. Therefore, as the observation time increases, the cyclic spectrum for one hop of FH/MFSK tends to zero for all f and α . This observation is obvious considering the fact that $y(t)$ is not a cyclostationary signal. Conversely, for finite T , the cyclic autocorrelation and cyclic spectral density for FH/MFSK will be non-zero.

Example 3 - Now, consider $y(t)$ consisting of multiple FH/MFSK hops

The cyclic spectrum for multiple FH/MFSK hops is obtained by first considering the case of two hops. To simplify the analysis, initial starting phases of $\phi_1, \phi_2 = 0$ were assumed, as well as arbitrary values for t_1, t_2, f_1 , and f_2 .

$$y(t) = p(t-t_1)e^{-i2\pi f_1 t} + p(t-t_2)e^{-i2\pi f_2 t}$$

where, as defined in the previous example,

$$p(t) = \text{Rect}(t/T_h)$$

The cyclic autocorrelation of $y(t)$ is

$$\begin{aligned} R_y(\tau) &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} y(t+\frac{\tau}{2}) y^*(t-\frac{\tau}{2}) e^{-i2\pi \alpha t} dt \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} \left[p(t+\frac{\tau}{2}-t_1) e^{-i2\pi f_1(t+\frac{\tau}{2})} + p(t+\frac{\tau}{2}-t_2) e^{-i2\pi f_2(t+\frac{\tau}{2})} \right] \\ &\quad \left[p^*(t-\frac{\tau}{2}-t_1) e^{i2\pi f_1(t-\frac{\tau}{2})} + p^*(t-\frac{\tau}{2}-t_2) e^{i2\pi f_2(t-\frac{\tau}{2})} \right] e^{-i2\pi \alpha t} dt \end{aligned}$$

By multiplying and separating terms,

$$\begin{aligned}
 R_y^{\alpha}(\tau) = & e^{-i2\pi f_1 \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2} - t_1) p^*(t - \frac{\tau}{2} - t_1) e^{-i2\pi \alpha t} dt \\
 & + e^{-i2\pi \left(\frac{f_1 + f_2}{2}\right) \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2} - t_1) p^*(t - \frac{\tau}{2} - t_2) e^{-i2\pi [\alpha + (f_1 - f_2)]t} dt \\
 & + e^{-i2\pi \left(\frac{f_1 - f_2}{2}\right) \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2} - t_2) p^*(t - \frac{\tau}{2} - t_1) e^{-i2\pi [\alpha - (f_1 - f_2)]t} dt \\
 & + e^{-i2\pi f_2 \tau} \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} p(t + \frac{\tau}{2} - t_2) p^*(t - \frac{\tau}{2} - t_2) e^{-i2\pi \alpha t} dt
 \end{aligned}$$

which reduces to

$$R_y^{\alpha}(\tau) = \begin{cases} \frac{1}{T} \Lambda(\tau/T_h) [e^{-i2\pi f_1 \tau} + e^{-i2\pi f_2 \tau}], & \alpha = 0 \\ \frac{1}{T} \Lambda((\tau - \Delta t)/T_h) e^{-i2\pi \left(\frac{f_1 + f_2}{2}\right) \tau}, & \alpha = \pm(f_1 - f_2) \\ & \Delta t = t_1 - t_2 \\ 0, & \text{otherwise} \end{cases}$$

Subsequently, the Fourier transform of $R_y^{\alpha}(\tau)$ gives $S_y^{\alpha}(f)$. Using the nomenclature from the previous example,

$$S_y^\alpha(f) = \mathcal{F}[R_y^\alpha(\tau)]$$

$$= \begin{cases} \frac{1}{T} T_h^2 [\text{Sinc}^2((f+f_1)T_h) + \text{Sinc}^2((f+f_2)T_h)] , & \alpha = 0 \\ \frac{1}{T} T_h^2 \text{Sinc}^2\left(\frac{(f_1+f_2)T_h}{2}\right) e^{-i2\pi\left(\frac{f_1+f_2}{2}\right)\Delta t} , & \alpha = \pm(f_1-f_2) \\ & \Delta t = t_1 - t_2 \\ 0 , & \text{otherwise} \end{cases}$$

The cyclic spectrum is thus defined along $\alpha = 0$ (i.e., power spectral density) and for α equal to the difference between the carrier frequencies of each hop. However, as in the previous example, when $R_p(\tau)$ is normalized over the integration period, it tends to zero as the integration period goes to infinity. As a result, the cyclic spectrum again tends to zero for all f and α .

In the case of multiple hops, the cyclic spectrum can be inferred from the two hop case; the cyclic spectrum components exist at the difference between all combinations of carrier frequencies.

Example 4 - Consider $y(t)$ to be a burst binary-phase-shift-keyed (BPSK) signal

$$y(t) = \sum_m a_m p(t - mT_h) \sum_n b_n p(t - nT_b) e^{-i2\pi f_c t}$$

with

$$\begin{aligned}
a_m &: \{0,1\} \\
b_n &: \{-1,1\} \\
T_b &= \text{bit period} \\
T_h &= T_{burst} = N_b T_b
\end{aligned}$$

where N_b is the number of bits per burst. In effect, $y(t)$ is a BPSK signal that is transmitted at a particular carrier frequency and at random integer multiples of the burst period. In order to find its cyclic spectrum, it is more convenient to express $y(t)$ in terms of its exponential Fourier series [7]. Furthermore, since $y(t)$ can be interpreted as a periodic waveform, it need only be described over one period for an exponential Fourier series to be defined. Considering $y(t)$ over one burst period, $T_h = T_{burst}$, a new function, $y_{T_h}(t)$, is obtained. This new function is equivalent to a time-limited BPSK signal. The exponential Fourier series is written as

$$y_{T_h}(t) = \sum_{n=-\infty}^{\infty} Y_n e^{i2\pi n f_h t}$$

where

$$Y_n = \frac{1}{T_h} \int_{-\frac{T_h}{2}}^{\frac{T_h}{2}} y_{T_h}(t) e^{-i2\pi n f_h t} dt$$

are the Fourier coefficients of $y_{T_h}(t)$, and the burst rate is denoted as

$$f_h = \frac{1}{T_h}$$

The spectrum of $y_{T_h}(t)$ can be found by taking the Fourier transform of both sides,

$$\begin{aligned}\mathcal{F}[y_{T_h}(t)] &= Y_{T_h}(f) = \mathcal{F}\left[\sum_{n=-\infty}^{\infty} Y_n e^{i2\pi n f t}\right] \\ &= \sum_{n=-\infty}^{\infty} Y_n \delta(f - n f_h)\end{aligned}$$

Since $y_{T_h}(t)$ reduces to a time-limited BPSK signal, the envelope of the spectrum of $y_{T_h}(t)$ also resembles that of the BPSK signal. The spectrum envelope for a BPSK signal is defined by the *Sinc*² function with nulls at integer multiples of the data rate. The Fourier coefficients in $Y_{T_h}(f)$ hence trace the *Sinc*² function with discrete components at integer multiples of $\frac{1}{T_h}$, which relates to the periodic nature of the signal.

By expressing $y(t)$ as an exponential Fourier Series, it is more convenient to use (26) directly for the cyclic spectrum. As a result,

$$\begin{aligned}S_y^\alpha(f) &\simeq \frac{Y_T\left(f + \frac{\alpha}{2}\right) Y_T^*\left(f - \frac{\alpha}{2}\right)}{T} \\ &\simeq \frac{Y_{T_h}\left(f + \frac{\alpha}{2}\right) Y_{T_h}^*\left(f - \frac{\alpha}{2}\right)}{T_h}\end{aligned}$$

Substituting for $Y_{T_h}(f)$ found above,

$$\begin{aligned}
S_y^{\alpha}(f) &= \frac{1}{T_h} \sum_n Y_n \delta\left(f + \frac{\alpha}{2} - nf_h\right) \sum_m Y_m^* \delta\left(f - \frac{\alpha}{2} - mf_h\right) \\
&= \frac{1}{T_h} \sum_{n,m} Y_n Y_m^* \delta\left(f + \frac{\alpha}{2} - nf_h\right) \delta\left(f - \frac{\alpha}{2} - mf_h\right)
\end{aligned}$$

which is non-zero only for $\alpha = nf_h = \frac{n}{T_h}$ with $n = 0, 1, 2, \dots$. The resulting spectrum consists of a series of delta functions spaced by the burst rate and whose envelope is defined by the spectrum envelope of the time-limited BPSK signal.

Drawing upon the analysis of a burst BPSK signal, the same approach can be applied to obtain a similar result for one hop of a frequency-hopped BPSK (FH/BPSK) signal. This follows because one hop of a FH/BPSK can be viewed as equivalent to $y_{T_h}(t)$ from the burst PSK example. The analysis to obtain the cyclic spectrum expression for a FH/BPSK waveform is thus the same as for the burst PSK waveform. It is noted however, that the burst PSK waveform exists for all T , and considering it over $T_{burst} = T_h$ helps to simplify the analysis. The single hop of the FH/BPSK signal only exists over T_h . Consequently, the cyclic autocorrelation and cyclic spectral density for the FH/BPSK waveform include the $\frac{1}{T}$ term and approach zero as $T \rightarrow \infty$.

CHAPTER III

CYCLIC SPECTRAL ESTIMATION

3.1 Introduction

The vast majority of signals under consideration are defined over a finite time interval. As a result, the cyclic spectral density, defined in (26), cannot be measured exactly. An equation for the finite-time case provides an estimate of the cyclic spectral density and is presented in this chapter. It is observed that the cyclic spectrum estimate equation is equivalent to a spectral correlation function. The equation is also extended to its discrete-time equivalent to allow processing of digitized signals.

Methods for computing the cyclic spectrum estimates are based on time-smoothing and frequency-smoothing techniques [3-5]. A brute force approach for computing a time-smoothed cyclic spectral estimate is first presented. An analysis of the number of complex multiplies required to cover the bifrequency plane illustrates the computational overhead of this approach. A brute force method for the frequency-smoothed estimate is also described. The computation of a time-smoothed estimate is investigated further in the detailed description of the computationally-efficient FFT accumulation method (FAM). The description consolidates key concepts which enable a reduction in the number of computations. The FAM algorithm is implemented in this thesis to compute the cyclic spectrum for several digitally-modulated signals. Details of the implementation are discussed in Chapter IV.

3.2 Finite-time Cyclic Spectrum Estimate

For a signal of length Δt , it is shown [1,4] that the cyclic spectral density can be estimated by the time-smoothed cyclic periodogram, denoted $S_{x_T}^\alpha(t, f)_{\Delta t}$,

$$\begin{aligned} S_x^\alpha(f) &\approx S_{x_T}^\alpha(t, f)_{\Delta t} = \frac{1}{\Delta t} \int_{t-\frac{\Delta t}{2}}^{t+\frac{\Delta t}{2}} S_{x_T}(u, f) du \\ &= \frac{1}{\Delta t} \int_{t-\frac{\Delta t}{2}}^{t+\frac{\Delta t}{2}} \frac{1}{T} X_T(u, f + \frac{\alpha}{2}) X_T^*(u, f - \frac{\alpha}{2}) du \end{aligned} \quad (27)$$

where $X_T(t, f)$, is the time-dependent finite-time (short-time) Fourier transform of $x(t)$ given by,

$$X_T(t, f) \triangleq \int_{t-\frac{T}{2}}^{t+\frac{T}{2}} x(u) e^{-i2\pi fu} du \quad (28)$$

Figure 3.1 illustrates this process graphically for an arbitrary (f_o, α_o) . For some signal $x(t)$, frequency components are evaluated over a time window T along the entire observation interval Δt . At each window, two components, centred about some f_o and separated by some α_o are multiplied and the resulting sequence of products is integrated over time. For all values of α , (27) can be interpreted as the correlation of two components, separated by α , centred about f . Hence, the cyclic spectrum estimate equation is also referred to as the Spectral Correlation Function.

The window of time over which $X_T(t, f)$ is performed gives a temporal resolution capability of $t_{res} = T$. From the Fourier transformation property, the spectral resolution capability, f_{res} , of $X_T(t, f)$ is approximately $\frac{1}{T}$. When the two spectral components are multiplied together in (27), the spectral resolution

Figure 3.1 Cyclic spectrum estimation via time-smoothing (a) Short-time Fourier transform over Δt (b) Sequence of frequency products from short-time Fourier transforms

capability of $S_{x_T}^{\alpha}(t, f)$, and subsequently the estimate, $S_{x_T}^{\alpha}(t, f)_{\Delta t}$, is the same as for $X_T(t, f)$ (i.e. $\Delta f = f_{res} = \frac{1}{T}$).

An estimate of the cyclic spectrum can also be made by frequency-smoothing the cyclic periodogram (i.e. integrating with respect to frequency). Referring to Figure 3.2, the short-time FFT is extended to cover the entire interval, Δt , so that only one is performed to obtain very fine resolution frequency components $\left(f_{res} = \frac{1}{\Delta t}\right)$. The cyclic periodogram is again computed using the finer resolution components. These components produce a cyclic periodogram with equally fine spectral resolution. The spectral correlation for a particular f_o and α_o is obtained by integrating values of the cyclic periodogram evaluated for α_o over a range of frequencies, centred about f_o . The equation for the frequency-smoothed cyclic periodogram is,

$$\begin{aligned} S_{x_{\Delta t}}^{\alpha}(t, f)_{\Delta f = \frac{1}{T}} &= \frac{1}{\Delta f} \int_{f - \frac{\Delta f}{2}}^{f + \frac{\Delta f}{2}} \frac{1}{\Delta t} X_{\Delta t}(t, \Omega + \frac{\alpha}{2}) X_{\Delta t}^*(t, \Omega - \frac{\alpha}{2}) d\Omega \\ &= \frac{1}{\Delta f} \int_{f - \frac{\Delta f}{2}}^{f + \frac{\Delta f}{2}} S_{x_{\Delta t}}^{\alpha}(t, \Omega) d\Omega \end{aligned} \quad (29)$$

Using this method, the frequency resolution of the estimate, Δf , depends on the range of frequencies over which the integration is performed. As observed in (29), Δf can be chosen so that the frequency resolution for both time- and frequency-smoothed estimates are the same. It is shown in [1], that as $\Delta t \rightarrow \infty$, the time- and frequency-smoothed cyclic periodograms are equal to each other.

In order for (27) and/or (29) to produce reliable estimates, a substantial amount of smoothing must be carried out to reduce the random effects of the signal.

Figure 3.2 Cyclic spectrum estimation via frequency-smoothing

From Grenander's Uncertainty Condition [1], this requires that,

$$\Delta t \Delta f \gg 1 \quad (30)$$

This implies that for time-smoothing, the interval over which the product sequence is integrated (Δt) must greatly exceed the temporal resolution capability (since $\Delta f = \frac{1}{T}$). Likewise, with frequency-smoothing, the range of frequencies being integrated (Δf) must greatly exceed the spectral resolution of the cyclic

periodogram (since $\Delta t = \frac{1}{f_{res}}$). In addition, it is shown [4] that the cycle frequency resolution of the estimate is $\Delta \alpha = \frac{1}{\Delta t}$. Therefore, in the bifrequency plane, estimates have finer resolution in cycle frequency than in temporal frequency. This suggests that the cyclic spectrum consists of a significant number of estimates requiring a correspondingly large number of computations.

3.3 Discrete-time Cyclic Spectrum Estimate

In practice, the finite-time signal $x(t)$ is often represented by a sequence of discrete-time samples. As a result, the equation developed for estimating the cyclic spectral density is extended to its discrete-time equivalent. To distinguish between the continuous- and discrete-time cases, the discrete temporal and cycle frequencies are represented by k and γ respectively.

For the time-smoothing method, consider a sequence, $x[n]$, of total length N . The discrete spectral correlation represents the correlation of two frequency bins, separated by γ , centred about k . Frequency bin components of $x[n]$ are computed over a window of N' samples via a discrete short-time Fourier transform which is defined,

$$X_{N'}(n, k) \triangleq \sum_{n=0}^{N'-1} x[n] e^{\frac{-j2\pi kn}{N'}} \quad (31)$$

The discrete finite cyclic periodogram becomes,

$$S_{x_{N'}}^{\gamma}(n, k) = \frac{1}{N'} X_{N'}(n, k + \frac{\gamma}{2}) X_{N'}^*(n, k - \frac{\gamma}{2}) \quad (32)$$

which leads to the equation for the time-smoothed discrete spectral correlation function,

$$S_{x_{N'}}^{\gamma}(n, k)_N = \frac{1}{N} \sum_n \frac{1}{N'} X_{N'}(n, k + \frac{\gamma}{2}) X_{N'}^*(n, k - \frac{\gamma}{2}) \quad (33)$$

The same procedure can be followed to obtain the discrete spectral correlation for the frequency-smoothing method giving,

$$S_{x_N}^{\gamma}(n,k)_{1/J} = \frac{1}{J} \sum_{j=-\frac{(J-1)}{2}}^{\frac{(J-1)}{2}} \left[\frac{1}{N} X_N(n, k + \frac{\gamma}{2} + j) X_N^*(n, k - \frac{\gamma}{2} + j) \right] \quad (34)$$

where an N -point FFT over the entire sequence $x[n]$ provides the finer resolution frequency components and J is the number of frequency products that are averaged.

J may be selected to be N' so that in both time- and frequency-smoothing methods, the discrete frequency and cycle frequency resolutions for the estimate are $\Delta k = \frac{1}{N'}$ and $\Delta \gamma = \frac{1}{N}$, respectively.

The choice between time- and frequency-smoothing approaches depends on whether estimates are required for the entire bifrequency plane or only for a moderate number of cycle frequencies [5]. In general, time-smoothing algorithms are considered more suitable where the entire bifrequency plane is of interest. Frequency-smoothing techniques are preferable when examining a small portion of the cyclic spectrum.

As in the finite-time case, the reliability of the discrete spectral correlation estimate is considered. Extending Grenander's Uncertainty Condition, the reliability of the discrete estimate now depends on,

$$\frac{N}{N'} \gg 1 \quad (35)$$

3.4 Cyclic Spectrum Estimation Algorithms

3.4.1 Brute Force Method

A brute force approach for computing the time-smoothed cyclic spectral estimate is first considered. An implementation using this technique is presented in [3] which can be referred to for further detail. The discrete spectral correlation equation is modified as,

$$S_{x_{N'}}^{\gamma}(n,k)_N = \frac{1}{M} \sum_{m=0}^{M-1} \frac{1}{N'} \overline{X_{N'}(n+mL, k + \frac{\gamma}{2})} X_{N'}^*(n+mL, k - \frac{\gamma}{2}) \quad (36)$$

where, using the time-shift property of the Fourier transform,

$$\begin{aligned} \overline{X_{N'}(n+mL, k)} &= \sum_{n=0}^{N'-1} x[n+mL] e^{\frac{-i2\pi k(n+mL)}{N'}} \\ &= \left[\sum_{n=0}^{N'-1} x[n+mL] e^{\frac{-i2\pi kn}{N'}} \right] e^{\frac{-i2\pi kmL}{N'}} \\ &= X_{N'}(n+mL, k) e^{\frac{-i2\pi kmL}{N'}} \end{aligned} \quad (37)$$

Inserting (37) into (36), the spectral correlation estimate becomes,

$$S_{x_{N'}}^{\gamma}(n,k)_N = \frac{1}{MN'} \sum_{m=0}^{M-1} \left[X_{N'}(n+mL, k + \frac{\gamma}{2}) X_{N'}^*(n+mL, k - \frac{\gamma}{2}) \right] e^{\frac{-i2\pi \gamma mL}{N'}} \quad (38)$$

With the addition of a "time-shift" parameter, mL , an N' -point FFT is taken every L samples instead of every sample. This can be visualized as a window "hopping" every L samples along the entire sequence $x[n]$ for a total of M time hops. The effect caused by the selection of L on the computation of the cyclic spectrum is discussed in the section describing the FAM algorithm.

The implementation of (38) in [3] begins with the computation of the series of N' -point or short-time FFTs. Subsequently, frequency-shifted components are complex-multiplied for every short-time FFT and averaged over the total number of time hops that are required to cover the sequence. The term, $e^{-i2\pi\gamma mL/N'}$, is generated from the "time-shift" of the data window and preserves the phase coherence between successive time hops. These steps must be performed for every value of k and γ in the band of interest.

Recall from Section 3.3, that the discrete temporal and cycle frequency resolutions for the estimate are given by $\frac{1}{N'}$ and $\frac{1}{N}$ respectively. Consequently, the estimate for some (k_o, γ_o) will occupy a finite area on the cyclic spectrum as shown in Figure 3.3a. The entire cyclic spectrum is computed by selecting all combinations of (k_o, γ_o) within the region of support, as shown in Figure 3.3b. The row-like coverage occurs because the values of γ are in increments of one temporal frequency bin (larger than the cyclic frequency resolution). Consequently, a cyclic component located outside of these strips will not be observed.

To address the problem of gaps in the bifrequency plane, [3] proposes to zeropad the N' points of the short-time FFT so that the frequency bin spacing is equal to the cycle frequency resolution at each hop. Therefore, a $(B=2N)$ -point FFT is performed every L samples on the sequence, $x[n]$, producing B spectral components spaced $\frac{1}{B}$ apart. The B points comprise the original N' points with $(B - N')$ additional zeroes. The spectral correlation estimate is rewritten as,

$$S_{x_{N'}}^{\gamma}(n, k)_N = \frac{1}{MB} \sum_{m=0}^{M-1} \left[X_B(n+mL, k+\frac{\gamma}{2}) X_B^*(n+mL, k-\frac{\gamma}{2}) \right] e^{\frac{-i2\pi\gamma mL}{N'}} \quad (39)$$

Figure 3.3 Cyclic spectrum estimate (a) For a particular (k_o, γ_o) (b) For all possible combinations (k_o, γ_o) for $N' = 8$

Having to perform a B -point FFT on every time hop to eliminate the occurrence of gaps in the cyclic spectrum increases the number of computations required to estimate the cyclic spectrum. The number of complex multiplies performed is used to measure the computational cost of the algorithm. For a sequence, $x[n]$, of length N samples, and N' samples for the short-time FFT, a total of $Z = 2N(N')$ estimates is required to cover the entire bifrequency plane. Table 3.1 lists the number of complex multiplies for all Z estimates using the time-smoothed brute force method.

$B = 2N$ -point FFTs	$M(B/2)\log_2 B$
Bin by Bin Multiplication	$Z(M)$
Time-shift Factor	$Z(M)$

Table 3.1 Number of Complex Multiplies Using the Time-Smoothed Brute Force Method

In Section 3.3, the frequency-smoothed cyclic spectral estimate, (34), was defined as,

$$S_{x_N}^Y(n, k)_{1/J} = \frac{1}{J} \sum_{j=-\frac{(J-1)}{2}}^{\frac{(J-1)}{2}} \left[\frac{1}{N} X_N(n, k + \frac{Y}{2} + j) X_N^*(n, k - \frac{Y}{2} + j) \right]$$

Using the same brute force approach to implement (34), a single FFT is performed on the entire sequence to resolve $x[n]$ into its spectral components. Similar to the time-smoothed estimate, the sequence must be zeropadded to $B = 2N$ in order to avoid gaps in the cyclic spectrum. A window of J frequency bins is captured about each of the two spectral components to be correlated. The frequency components included in one window are complex-multiplied with the corresponding components in the other. These J complex products are subsequently averaged to produce the cyclic spectral estimate. The frequency resolution of the estimate, Δk , is now dependent on the length of the frequency window, J , and is equal to J times the resolution of the spectral components.

If the frequency resolution for the frequency-smoothed estimate is set to be equal to that of the time-smoothed estimate (i.e. $\frac{J}{B} = \frac{1}{N'}$), the same number of estimates is obtained for the bifrequency plane. The number of complex multiplies required to compute the same Z estimates of the cyclic spectrum using the

frequency-smoothing approach is given in Table 3.2.

$B = 2N$ -point FFT	$(B/2)\log_2 B$
Bin by Bin Multiplication	$Z(J)$

Table 3.2 Number of Complex Multiplies Using the Frequency-Smoothed Brute Force Method

3.4.2 FFT Accumulation Method

The FFT accumulation method (FAM) [5] is based on the time-smoothing approach and is derived from the equation for the time-smoothed cyclic periodogram (33). Sampling theory and signal processing concepts are applied to reduce the number of computations required to compute the cyclic spectrum estimates. It is noted that some of the considerations discussed in this section are included in the brute force time-smoothing approach for calculating the cyclic spectrum estimates. However, they are consolidated here to better illustrate the reduction in the number of computations.

The strategy of the FAM algorithm is to divide the entire bifrequency plane into smaller areas so that cyclic spectrum estimates are computed a block at a time. A general description of the FAM procedure is given here. It is directly followed by a more detailed examination of the considerations for each step of the process.

Spectral components of a sequence, $x[n]$, are computed using a short-time (N' -point) FFT. Two components are multiplied to provide a sample of a cyclic

spectrum estimate for a finite area on the bifrequency plane. A sequence of samples for a particular area can be obtained by performing a series of consecutive short-time FFTs over the whole length, N , of the input sequence and multiplying the same two components for each short-time FFT. Instead of computing an average of the samples, an FFT is performed on the second sequence to further resolve the estimates along the cycle frequency axis. The second FFT generates a block of cyclic spectrum estimates; one of these (i.e. the DC component) would include the estimate obtained by time-averaging. Estimates over the entire cyclic spectrum can be obtained by following this procedure for all combinations of spectral components pairs from the short-time FFTs.

3.4.2.1 Partitioning the Bifrequency Plane

Consider a sequence of data obtained from sampling a signal at a frequency f_s . If an N' -point FFT is performed, N' spectral components are computed at intervals of $\frac{f_s}{N'}$ equally spaced about the bandwidth ($-\frac{f_s}{2}$ to $\frac{f_s}{2}$). Each component has a frequency resolution or bin width of k_{res} , proportional to $\frac{f_s}{N'}$. Subsequent correlation of two frequency components, a distance γ_o apart centred about some k_o , produces a sample of a cyclic spectral estimate at (k_o, γ_o) on the bifrequency plane. As the spectral components have finite bandwidth, the sample of the cyclic spectrum estimate at (k_o, γ_o) will represent a finite area on the bifrequency plane. This area is defined as a channel pair region. A bifrequency plane with the channel pair regions outlined for the case $k_{res} = \frac{f_s}{N'}$ is shown in Figure 3.4. For an N' -point FFT, $(N')^2$ possible channel pair regions can be defined. The grid of channel pair regions and thus samples, is formed over the entire bifrequency plane by correlating all possible combinations of frequency bin pairs.

Figure 3.4 Outline of channel pair region, when $k_{res} = f_c/N$

The dimensions of the channel pair regions can be inferred by drawing an analogy between the spectral correlation of two frequency components and the convolution of two unit pulse functions of equal length. Referring to Figure 3.5a, it is assumed that the centre of the frequency bin is used as the point of reference for the spectral correlation. The distance between the two centres of the bins is represented by the discrete cycle frequency, γ , and is analogous to the delay variable in convolution. As the two spectral components have finite bandwidth, their correlation (or convolution) will exist only for a finite range of cycle frequency (or delays). In this example, the range, also referred to as the product window bandwidth, is equal to twice the spectral resolution, k_{res} , as shown in Figure 3.5b.

Figure 3.5 Channel pair region description (a) Frequency components separated by γ_o (b) γ dimension (c) k dimension (d) Channel pair region dimensions

The dimension of the channel pair region along the k -axis is related to the amount of spectral overlap between the two components for a particular value of γ_o . In effect, it indicates the amount of frequency bin energy which contributes to the correlation at a particular γ_o . The overlap is maximized when the two components are aligned. However, as the distance between the spectral components increases and the two bins "slide" away from each other, the amount of overlap decreases (see Figure 3.5c) and reduces to zero at the limits of the range of γ . Thus, the temporal frequency dimension is non-uniform over the range of γ defined for the channel pair region. The dimensions and shape of the channel pair region are shown in Figure 3.5d. It is noted that the sidelobes of each frequency component and consequently channel pair region, are omitted to simplify Figures 3.5a-d.

3.4.2.2 Product Sequence Decimation

For every N' -point FFT performed, a partitioned bifrequency plane, as shown in Figure 3.4, can be formed. If an N' -point FFT is performed on an input sequence of length N at every sample, a second sequence is created for each channel pair region. This new sequence (also of length N) has a sampling frequency, f_g , equal to the original input sequence, (i.e. $f_g = f_s = \frac{1}{T_s}$). Recall from the definition of the channel pair region, that these samples have a bandwidth in γ of $2k_{res}$. From sampling theory [10], it is established that a signal can be reconstructed accurately from sampled data as long as the samples are collected at a minimum rate of the total bandwidth of the signal. Therefore, from Figure 3.6, f_g can be reduced to $2k_{res}$ without compromising any information in γ needed to reconstruct the second sequence. A reduction in f_g means that there are fewer short-time FFTs and consequently fewer samples in the second sequence.

Figure 3.6 Product window bandwidth in γ of cyclic spectrum estimate sample

Therefore, the second sampling frequency, f_g , can be reduced by a factor L such that,

$$\text{new } f_g = \frac{f_s}{L} \geq 2k_{res} \quad (40)$$

By decimating the second sequence by a factor L , an N' -point FFT need only be performed every L samples instead of every sample. The length of the second sequence is subsequently reduced from $P=N$ to $P=\left(\frac{N-N'}{L} + 1\right)$. It is noted that P is equivalent to the number of time hops, M , defined for the brute force time-smoothed method.

3.4.2.3 Computing a Block of Cyclic Spectrum Estimates

In the brute force method, a time-average using the samples from the second sequence is taken to obtain a cyclic spectrum estimate. However, this only provides one estimate for a small area on the bifrequency plane that is defined by the discrete temporal and cycle frequency resolutions. Furthermore, simple time-averaging leaves gaps in the cyclic spectrum unless the short-time FFTs are zeropadded as discussed in Section 3.4.1. In the FAM, a P -point FFT is performed to further resolve cyclic

spectral components along the γ -axis in the channel pair region. The result is the simultaneous calculation of a block of cyclic spectrum estimates in the bifrequency plane using only two "coarse" temporal frequency components.

The resolution of the bandwidth in γ into finer cyclic spectral components is shown in Figure 3.7. As the frequency resolution, k_{res} is not constant throughout the channel pair region, the ratio of the frequency resolution to the cycle frequency resolution varies. The reliability condition (35) requires the resolution in cycle frequency to be much finer than in temporal frequency, suggesting that estimates toward the top and bottom of the channel pair region are less reliable. To address this problem, one approach involves taking only those resolved cyclic spectrum components that are the equivalent in γ of $\pm \frac{k_{res}}{2}$ away from the centre of the channel pair region [5]. However, from Figure 3.8, taking only some of the cyclic spectrum estimates will again leave gaps in the bifrequency plane.

Figure 3.7 Resolving γ -bandwidth into P components

Figure 3.8 Gaps in bifrequency plane as a result of considering estimates $\pm k_{res}/2$ from the channel pair region centre

Data tapering windows are used in FFT analysis to attenuate sidelobe levels and hence reduce spectral leakage effects. In conjunction with a decrease in the sidelobe levels is an increase in the width of the main lobe. This latter effect is used to address the problem of gaps in the bifrequency plane.

Up to this point, the ideal case for the frequency component resolution $\left(k_{res} = \frac{f_s}{N'}\right)$ and subsequently the channel pair region dimensions has been considered. The width or resolution of the output frequency bins is more accurately defined as $k_{res} = \beta \frac{f_s}{N'}$, where β commonly refers to the 3.0dB or 6.0dB bandwidth.

β is given in terms of the number of frequency bins and generally ranges between 1.0 to 2.0 in value depending on the type of window used (the 6.0dB bandwidth for a Hamming window is $\beta = 1.81$) [11]. For any value of $\beta > 1.0$, the resolution of

the frequency component bin increases. As a result, the dimensions of the channel pair region will also increase and produce a partitioned bifrequency plane with overlapping channel pair regions. Figure 3.9 illustrates some of the overlapping regions when data is tapered with a Hamming window. The advantage of this effect is that the overlap eliminates any gaps when considering estimates that are $\pm \frac{k_{res}}{2}$ away from the centre of the channel pair region.

Figure 3.9 Overlapping channel pair regions (a) with Hamming window (b) Channel pair region covered by estimates $\pm k_{res}/2$ from centre

The data taper consideration is included to obtain an appropriate value for the decimation factor or overlap parameter, L . In the worst case, a data taper window with $\beta = 2.0$ is used so that the frequency resolution, k_{res} , becomes

$$k_{res} = (2.0) \frac{f_s}{N'} \quad (41)$$

Inserting k_{res} into (40) gives

$$\frac{f_s}{L} \geq 2(2.0) \frac{f_s}{N'} \quad (42)$$

In terms of the decimation factor, (42) reduces to,

$$L \leq \frac{N'}{4} \quad (43)$$

In other words, the number of computations can be reduced by decreasing the overlap of the N' -point FFTs to $\frac{3}{4}N'$. If a data taper window with $\beta = 2.0$ is used and

$L = \frac{N'}{4}$, the area of the channel pair region that meets estimate reliability requirements $\left(\pm \frac{k_{res}}{2} \right)$ is represented by exactly half of the output bins from the second FFT. In addition, these output bins extend over the channel pair region such that no gaps are observed.

The data taper and the limit imposed on the decimation factor (and hence the sampling frequency of the (second) product sequence) prove to be significant in reducing an effect called Cycle Leakage [1,3,5]. Cycle leakage refers to the introduction of false characteristics in the cyclic spectrum as a result of energy which has "leaked" into other frequency bins from the sidelobes. Subsequent correlation of these other bins may produce appreciable values that are mistaken as cyclic features. The data taper minimizes this leakage by suppressing the sidelobes. Decimation of the second sequence may also result in cycle leakage. It is known that the frequency representation of a sampled signal is replicated at integer multiples of the sampling

frequency. As the sampling frequency is reduced via decimation, the replicas move closer to one another. If the sampling frequency is too small and aliasing occurs, it may cause cyclic components to be observed at incorrect cycle frequencies.

3.4.2.4 FAM Process Summary

The considerations described above form the basis for the FAM algorithm. The FAM algorithm is described by,

$$S_{x_{N'}}^{\gamma_i + q\Delta\gamma}(n, k_j)_N = \sum_m \left[X_{N'}(n+mL, k_j + \frac{\gamma_i}{2}) X_{N'}^*(n+mL, k_j - \frac{\gamma_i}{2}) e^{\frac{-i2\pi\gamma_i mL}{N'}} \right] e^{\frac{-i2\pi mq}{P}} \quad (44)$$

where,

$$X_{N'}(r, k) = \sum_{a=0}^{N'-1} w[a] x[a+r] e^{\frac{-i2\pi ka}{N'}} \quad , w[a] = \text{window coefficients} \quad (45)$$

An N' -point window is "hopped" over a sequence of data in blocks of $L = \frac{N'}{4}$ samples at a time. At every L samples, a data taper window is applied and an N' -point FFT is performed to obtain the spectral components of the input sequence. Two spectral components, a distance of γ_o apart, centred about k_o , are multiplied together for every N' -point FFT to form a complex product sequence of length P . Each sample of this second sequence is multiplied by the appropriate time-shift exponential, $e^{-i2\pi\gamma mL/N'}$. This sequence is transformed by a P -point FFT to obtain cyclic spectrum estimates for the channel pair region associated with the coordinates (k_o, γ_o) . Of the P output bins, half are stored as reliable estimates of the cyclic spectrum. The number of complex multiplies for the FAM algorithm is listed in Table 3.3.

Data Taper	$P(N')$
N' -point FFTs	$P(N'/2)\log_2 N'$
Bin by Bin Multiplication	$P(N')^2$
Time-shift Exponential	$P(N')^2$
P -point FFT	$[(N')^2](P/2)\log_2 P$

Table 3.3 Number of Complex Multiplies Using the FAM Algorithm

CHAPTER IV

CYCLIC SPECTRUM ANALYZER

DESIGN CONSIDERATIONS AND IMPLEMENTATION

4.1 Introduction

The purpose of this chapter is to present the components of the experimental setup that provide a practical scenario for testing the implementation of the FAM algorithm. Implementation details are discussed and supported via calculations where appropriate. The key components in the experimental setup are as follows: signal generation, received signal processing, baseband signal processing and data collection, cyclic spectrum estimate computation, and hardware/user interface.

An overview of the experimental setup is illustrated in the block diagram in Figure 4.1.

Figure 4.1 Experimental Setup

A variety of signal waveforms having modulation schemes commonly found in

digital communications are generated and combined to aid in evaluating the FAM algorithm. These IF test signals represent a typical RF front-end output of a communications receiver.

The receiver/downconverter filters and downconverts the IF signals to baseband for subsequent signal processing. Both in-phase (I) and quadrature (Q) components are obtained to facilitate analysis of complex waveforms.

An analog-to-digital (A/D) board is designed and fabricated to digitize the analog waveform and perform high speed data collection. The process is controlled by a DSP board and on-board circuitry.

During the A/D operation, the DSP monitors the status flags of the A/D board. When it observes that data collection has been completed, the sampled data is read into DSP memory. The DSP performs the FAM computations to obtain the cyclic spectrum estimates.

The DSP board is contained in a PC which loads the DSP program for the FAM computations. The results of these computations are formatted and transferred by the PC to an ASCII file for subsequent graphical analysis.

The implementation of the DSP and PC programs is discussed in the software implementation section. Considerations leading to the choice of parameters and/or methodology used in the programs are examined. A software routine was also developed to inject user-provided data into the DSP memory as an aid to debugging and evaluating the FAM algorithm.

4.2 Hardware Implementation

4.2.1 Signal Generation

A block diagram of the components that generate the test signals is included in Figure 4.2. As mentioned in Section 2.4.1, the principal signal characteristics that are potentially revealed using cyclic spectral analysis include: carrier components, data/symbol rates, and hop rates. Accordingly, continuous wave (CW), binary-phase-shift-keying (BPSK), quaternary-phase-shift-keying (QPSK), and frequency-hopped (FH) signal waveforms were selected as signal sources. A programmable noise generator is also included as a signal source to allow a range of E_b/N_o to be tested. The centre frequency for all generated signal waveforms is 70 MHz. A range of E_b/N_o between -10dB and 10dB, and R_b between 64kbps and 2Mbps are proposed for testing.

4.2.2 Receiver/Downconverter

A block diagram of the receiver/downconverter components is included in Figure 4.2. Measured insertion losses of the hardware components are provided to enable a comparison of power levels at key points. A monitor port is also provided at the output of the bandpass filter to facilitate the comparison.

The bandpass filter effectively limits the bandwidth of the communications channel of interest. For this implementation, filters with bandwidths of 1MHz and 5MHz centred about 70MHz are considered for the calculation of the cyclic spectrum.

The bandpass filter output signal is translated via quadrature detection to

baseband after being mixed with a 70MHz reference. The resulting quadrature signals are transferred to an A/D board to be processed as described below. The desired signal and noise power levels presented to the A/D board are governed by the dynamic range of the A/D converter (ADC) and the range of E_b/N_o and R_b .

Figure 4.2 Signal generation and receiver block diagram

4.2.3 Baseband Processing/Digitization

An A/D board, designed and fabricated for this thesis, is used to digitize the baseband quadrature signals and store the data. The block diagram in Figure 4.3 illustrates the general A/D board operations: filtering, signal amplification, A/D conversion, data latch, storage of digitized samples, and timing signals generation. The A/D board operates at clock rates up to 10MHz and processes both I and Q channels simultaneously.

Figure 4.3 Baseband I/Q A/D Board

The circuit diagram of the A/D board is included in Appendix A. The A/D board is a double-sided printed circuit board. East-west tracks run on one side of the board (component side) while north-south tracks run on the other (solder side). In the board layout, an effort is made to isolate analog and digital components to minimize coupling of digital signals and digital ground onto the relatively small amplitude analog signals.

Considerations relating the parameters of the buffer amplifier and ADC to the choice of signal and noise power levels at the input of the A/D board are expanded upon in the following sections. The ADC characteristics define the A/D board operating limits and are presented first.

4.2.3.1 A/D Converter

The AD9005^A is an ADC that has a digitization range of $\pm 1.024\text{V}$ at

sampling rates up to 10 MHz. The ADC uses binary offset coding to digitize a bipolar input signal so that 1 bit is used to represent the sign bit and 11 bits represent the magnitude.

As mentioned in Section 4.2.2, the ADC's dynamic range affects the noise power level that is required to accommodate the desired range of E_b/N_o and R_b to be tested. The calculation of the noise power level is facilitated by establishing the relationship between voltage level, quantization units, and dynamic range for the AD9005A. Figure 4.4 maps the signal magnitude range (in volts) to the quantization units of the ADC and the quantization level (in dBQu).

Figure 4.4 A/D voltage, quantization units, and dynamic range scales for positive signals

The relationship between voltage level and quantization units is defined as follows,

$$\frac{1.024V}{2048 \text{ qu}} = 0.0005 \frac{V}{qu} = 0.5 \frac{mV}{qu}$$

If $1qu = 0.0005V = 0dBqu$, the relationship between the units of dBqu and dBm, assuming a 50Ω load, can be established as follows,

$$\begin{aligned} 0 \text{ dBqu} &= 10 \times \log \left[\left(\frac{(0.0005V)^2}{50\Omega} \right) \times 10^3 \frac{mW}{W} \right] \\ &= -53.01dBm \end{aligned}$$

An appropriate mapping of expected signal and noise power levels is required to avoid operating the ADC near either of the input range limits. As a result, it is desirable, in practice, to map the input signal range in the middle of the A/D dynamic range. Recall that the proposed range of E_b/N_o and R_b for the test signals are,

$$-10dB < \frac{E_b}{N_o} < 10dB$$

$$64kbps < R_b < 2Mbps$$

The dynamic range of the input signal can be determined by considering the signal-to-noise ratios (SNRs) of the two extreme cases of E_b/N_o , R_b , and bandwidth, B .

CASE 1: $E_b/N_o = -10dB$, $R_b = 64kbps$, $B = 5MHz$

$$\begin{aligned} SNR &= \frac{E_b R_b}{N_o B} = -10dB + 10 \log \left[\frac{(64 \times 10^3)}{(5 \times 10^6)} \right] \\ &= -28.93dB \end{aligned}$$

CASE 2: $E_b/N_o = 10dB$, $R_b = 1Mbps$, $B = 1MHz$

$$\begin{aligned} SNR &= \frac{E_b R_b}{N_o B} = 10dB + 10 \log \left[\frac{(1 \times 10^6)}{(1 \times 10^6)} \right] \\ &= 10dB \end{aligned}$$

The required dynamic range of the input signals is the difference between the SNRs in CASE 1 and 2.

$$\begin{aligned}\text{Signal Dynamic Range} &= 10\text{dB} - (-28.93\text{dB}) \\ &= 38.93\text{dB}\end{aligned}$$

The signal dynamic range can then be mapped into the middle of the ADC's dynamic range as illustrated in Figure 4.5.

Figure 4.5 Signal dynamic range mapped into ADC dynamic range

Figure 4.5 shows the noise power level, N_p , 23dB below 66dBm. Recall, from Figure 4.4, that 66dBm represents 1.024V. Using the relationship established between dBm and dBm, the noise power level is

$$N_p = 66\text{dBqu} - 23\text{dB}$$

56

$$= 43\text{dBqu}$$

$$= -10\text{dBm}$$

This value assumes that the signal dynamic range can be mapped in the middle of the ADC dynamic range. The selection of N_p must also ensure that the minimum detectable signal is larger than the quantization noise power, N_q . As shown in [12,13], the quantization error, ϵ , for a uniform quantizer can be characterized by the uniform probability density function,

$$f(\epsilon) = \begin{cases} \frac{1}{a} & -\frac{a}{2} \leq \epsilon \leq \frac{a}{2} \\ 0 & \text{otherwise} \end{cases}$$

where $a=2^{-r}$ is the quantization step for an r -bit quantizer. N_q is the mean square value of ϵ ,

$$\begin{aligned} N_q &= E[\epsilon^2] \\ &= \frac{a^2}{12} \text{qu} \\ &= 10\log \frac{2^{-2r}}{12} \text{dBqu} \\ &= -77.0\text{dBqu} \end{aligned}$$

From the dBqu-dBm relationship derived above, ($0\text{dBqu} = -53.01\text{dBm}$), N_q is extremely small relative to the lowest input signal level (13dBqu). Thus, the optimum use of the ADC dynamic range and the minimum detectable signal requirements are both satisfied with a noise power level, N_p , of -10.0dBm.

4.2.3.2 LPF and Buffer Amplifier

The mixer outputs for the quadrature signal components consist of signals at

baseband and twice the IF frequency. A buffer amplifier combined with a first-order LPF (RC) is included at the input of the A/D board to filter out the higher frequency components. The level of amplification is obtained by evaluating the noise power expected at the input of the A/D board and using the noise power level calculated in the previous section for the ADC. The configuration of the LPF/buffer amplifier, given by the data sheets, is shown in Figure 4.6

Figure 4.6 LPF/Buffer Amplifier Circuit

A board input impedance of 50Ω is required to match the mixer output. The terminated resistance, R_{term} , at the input of the buffer amplifier is specified as the parallel connection of R_1 and R_3 . Thus, a value of 100Ω is chosen for R_1 and R_3 so that,

$$R_{term} = R_1 // R_3 = \frac{R_1 R_3}{R_1 + R_3} = \frac{100\Omega \times 100\Omega}{200\Omega} = 50\Omega$$

The range of values expected at the input and output of the buffer amplifier stage must be considered in order to establish the gain required. To examine the level at the input, consider the top end of the signal dynamic range. From the previous section, the maximum SNR occurs for $E_b/N_o = 10\text{dB}$, $R_b = 1\text{Mbps}$,

and $B = 1\text{MHz}$.

$$\begin{aligned}
 SNR &= \left(\frac{E_b}{N_o} \right)_{\max} \left(\frac{R_{b\max}}{B} \right) \\
 &= 10\text{dB} + 10\log \left[\frac{1 \times 10^6}{1 \times 10^6} \right] \\
 &= 10\text{dB} \\
 &= 10 \text{ (linear)}
 \end{aligned}$$

which means that,

$$S \simeq 10N_p$$

If it is assumed that a maximum signal-plus-noise ($S+N_p$) level of 0dBm (1mW) is to be generated, the noise level can be determined according to the above relation,

$$\begin{aligned}
 S+N_p &= 1\text{mW} \\
 10N_p+N_p &= 1\text{mW} \\
 N_p &= 0.09\text{mW} \\
 &= -10.41\text{dBm}
 \end{aligned}$$

If a total loss of approximately 5dB is estimated between the signal generator and the input of the A/D board, the noise power level at the input of the A/D board is $-10.41\text{dBm}-5\text{dB} = -15.41\text{dBm}$. Recall from Section 4.2.3.1, that the noise power level at the input of the ADC was determined to be -10.0dBm . This implies that an overall gain of 5.41dB (≈ 2) is required between the input of the A/D board and the input of the ADC.

At the buffer amplifier output, consideration must be given to the maximum input voltage range supported by the ADC. While the input voltage range that is digitized by the A/D converter is $\pm 1.024\text{V}$, a range of $\pm 1.5\text{V}$ is selected as the

cutoff level to protect the A/D converter from amplifier saturation and/or large input signals. However, the maximum output voltage rating of the AD844 is $\pm 12\text{V}$ when powered by $\pm 15\text{V}$ supplies as is the case for the A/D board. In the event of a transient AD844 output of 12V , which may occur at power-up, for example, an attenuation by a factor of 8 ($12\text{V} \div 1.5\text{V} = 8$) is required to ensure that the voltage at the input of the A/D converter does not exceed 1.5V . As a result, a voltage divider is included in the buffer amplifier circuit as shown in Figure 4.7. The voltage divider is implemented simply with an additional resistor between the output of the AD844 and the input the A/D converter. With an input resistance, R_{in} , of $1\text{k}\Omega$ for the A/D converter, the value of R_4 can be determined as follows,

$$\frac{V_{o \text{ AD844}}}{V_{in \text{ A/D}}} = \frac{R_{in} + R_4}{R_{in}} = 8$$

$$R_4 = 8R_{in} - R_{in}$$

$$R_4 = 7\text{k}\Omega$$

With an overall gain of 2 required between the input and output of the buffer amplifier stage, and a divide-by-8 voltage divider at the output of the AD844, the actual gain required from the AD844 is 16. The magnitude of the AD844 gain is given by $G = \frac{R_2}{R_1}$ so that

$$R_2 = GR_1 = 16(100\Omega) = 1.6\text{k}\Omega$$

As mentioned earlier, a LPF/buffer amplifier is also required to filter out the signal that is replicated at twice the IF frequency during downconversion. Since a sharp roll-off is not needed to eliminate the higher frequency version, a simple first order LPF (RC) can be implemented. The value of the capacitor, $C = 1200\text{pF}$, was selected to provide a corner of approximately 5MHz .

The final configuration of the LPF/buffer amplifier stage is shown in Figure 4.7.

Figure 4.7 Final configuration of LPF/Buffer amplifier circuit

4.2.4 Data Collection and Timing

4.2.4.1 Data Latch and FIFO Memory

Three edge-triggered flip flops are implemented to hold the valid ADC 12-bit output for the storage operation. There are four 9-bit First-In-First-Out (FIFO) memory chips on board to collect and save up to 4k I and Q digitized samples of the received signal. These FIFOs must be reset to initialize the pointers that point to the next memory location to be accessed during a read or write operation. Half- and Full-Flags are provided by the FIFO to indicate the FIFO status.

4.2.4.2 Clock Source Selection

There are two available clock sources for the A/D board operation: an on-board crystal oscillator and an external clock source. A circuit is included that enables the user to select the desired clock source. A 74LS157 2-to-1 multiplexer provides a SEL (select) pin which, depending on whether 5V or 0V is present at the SEL pin, passes one of the input clock signals to the output. The selection of 5V or 0V is realized by a jumper and is shown in Figure 4.8. The SEL pin of the 74LS157 is presented with 5V (less the voltage drop across the resistor) when the jumper is removed and 0V when it is inserted. An output enable (/OE) is also available to enable or disable the 2-to-1 multiplexer, thereby disabling the clock signal. The circuitry required for the /OE signal is discussed in Section 4.2.4.4.

Figure 4.8 Clock source selection

4.2.4.3 Timing Signals Generation

A timing diagram illustrating the overall A/D board operation is found in Figure 4.9. There are three important timing signals involved in digitizing an input

Figure 4.9 Timing diagram for digitization and storage operation

analog signal: the A/D conversion (ENCODE), the data latch (374 CLKEN), and the FIFO write (/W). These timing signals are generated to meet data setup and hold times for clock rates between 1MHz and 10MHz. Timing diagrams recorded by a logic analyzer for key events in the A/D board operation are included in Appendix B.

The A/D operation begins on the rising edge of the ENCODE signal, after which the internal track-and-hold (T/H) of the AD9005A enters the hold (H) mode. The ENCODE signal is held high for a minimum of 25ns to ensure that the AD9005A is in the hold state before the conversion takes place. The digitized sample appears on the output data lines after a three clock cycle delay.

For this thesis the ENCODE signal is generated by using a 74LS221 monostable multivibrator as shown in Figure 4.10. With the proper selection of component values for R_{ext} and C_{ext} , the required pulse width is generated upon receipt of a rising edge at the input B according to the following equation,

$$\begin{aligned} t_{pulse} &= \ln(2)R_{ext}C_{ext} \\ &\approx 0.7R_{ext}C_{ext} \end{aligned}$$

Figure 4.10 ENCODE and CLKEN signals generation

Using values of $5K\Omega$ and $10pF$ for R_{ext} and C_{ext} respectively, a rising edge at input B generates a $35ns$ wide pulse at the Q output.

Data latches are included on the A/D board to ensure that all ADC output data lines are valid before the value is stored. The data latches are implemented using three 74LS374 edge-triggered flip flops which are activated on the rising edge of a CLKEN signal. When a rising edge is supplied to the 74LS374, the value of the input data lines is latched to the output data lines until another rising edge is received.

The main consideration for the CLKEN signal generation is the proximity of the rising edge to the transition of the data lines at the output of the AD9005A. Data sheets state that the valid AD9005A output may appear a maximum of $100ns$ after the rising edge of the ENCODE. If a $10MHz$ clock is used ($100ns$ cycle time) to generate the ENCODE signal and the CLKEN assumes the same rising edge, it is possible that the output of the AD9005A may be changing at the same time as the activation of the data latch. Therefore, the same signal cannot be used for both the ENCODE and CLKEN. Consideration must also be given to the situation where the delay for a valid AD9005A output is less than $100ns$. The typical rating for the delay is given by the data sheets as $90ns$. As this value is close to the maximum delay, it is unlikely that the AD9005A output delay from the rising edge of the ENCODE signal is less than $45-50ns$. Hence, a CLKEN rising edge that is generated between the rising edge of the ENCODE signal and $45ns$ after that edge will properly latch the AD9005A data lines.

The 74LS221 circuit in Figure 4.10 for the generation of the ENCODE signal also provides an inverted output ($/Q$) with a pulse width of $35ns$. Since the rising

edge occurs at the end of the pulse (i.e. 35ns after the rising edge of ENCODE) this signal can be used to meet both constraints of the positioning of the CLKEN rising edge.

Once the data lines are latched, the sample can be written to the FIFO. The write operation of the FIFO begins when the write signal (/W) goes low and remains low for a minimum of 25ns. However, the actual storage of the data sample occurs on the rising edge of the write cycle. The write signal remains high for a minimum of 10ns before the next write operation can be initiated, resulting in a 35ns minimum write cycle. The 35ns minimum write cycle is easily accommodated by the A/D board since the maximum clock rate is 10MHz giving a 100ns cycle time.

A potentially suitable write signal is already available from either the Q or /Q output of the 74LS221 as the 35ns pulse width meets both minimum time requirements for high and low levels specified above for the FIFO /W. For a valid write operation FIFO input data lines must meet setup and hold times about the rising edge of the write cycle. Therefore, the location of the 74LS221 Q and /Q rising edges with respect to the transition of the 74LS374 output data lines must be examined to determine if either can be used as the FIFO /W.

The transition of the 74LS374 output data lines occurs a maximum of 28ns after the rising edge of the CLKEN. Since no minimum delay is given, a 0ns minimum delay must be assumed which would result in 74LS374 output data lines changing on the CLKEN rising edge. If the /Q output of the 74LS221, which corresponds to the CLKEN signal, is used for the FIFO /W, the data setup time for the FIFO before the rising edge may not be met. The 74LS221 Q output, however,

provides a rising edge between the CLKEN rising edges. When the shortest cycle time (100ns for 10MHz clock rate) is considered, the Q output of the 74LS221 generates a rising edge approximately 65ns after the rising edge of the CLKEN. This is well after the 28ns maximum delay for a valid 74LS374 output and ensures that the data lines are not changing when the storage operation occurs.

As shown in Figure 4.9, the first four samples stored on the FIFO are invalid due to the pipeline delay of the AD9005A conversion process. The inclusion of these four samples is not significant, however, considering that up to 2K samples may be written to the FIFO.

Although the selection of the 74LS221 Q output meets the specifications for a valid write operation, there remains an additional consideration with respect to the status of the FIFO /W. The FIFO /W signal must be high when the A/D board is being reset. If the 74LS221 Q output is directly connected to the FIFO /W input, this requirement is not satisfied because during a master reset, the clocks are disabled and the Q output of the 74LS221 is low. Instead, a 74LS157 (2-to-1 multiplexer) (see Figure 4.11) is implemented to provide one of two possible signals to the FIFO /W input: the 74LS221 Q output (during data collection process) and 5V during a master reset or idle period. The selection of the FIFO /W signal is determined by the output of another monostable output, discussed in the next section.

Figure 4.11 FIFO Write signal generation

4.2.4.4 Timing Signals Control

Before the data collection process can begin, the A/D board must be initialized so that the FIFOs are empty and the timing signals are enabled. The data collection process is initiated by a master reset cycle from the DSP. The only requirement during a master reset is that the read (/R) and write (/W) signals of the FIFO must be held high. When the FIFOs are full and the FIFO full flag is activated, the A/D collection process is halted and all timing signals are disabled. The A/D board then waits for the DSP board to read the data.

The hardware realization for the timing signals control is shown in Figure 4.12. On the rising edge of the master reset (input B), a pulse is generated at the outputs of the 74LS221 monostable. Since a transition from high to low is needed, the /Q output can be used as the /OE of the 74LS157 to allow the selected clock source through. In addition, it can be used as the SEL of the other 74LS157 to release the FIFO /W from being held at 5V. When the FIFOs are full, the FIFO full flag (connected to the CLR pin of the 74LS221) clears the outputs and returns the /Q output to a high level. As a result, the clock source is disabled and the FIFO /W is once again pulled high.

Figure 4.12 Timing signals control

If for any reason, a FIFO full flag is not received the 74LS221 times out after a period determined by the selection of R_{ext} and C_{ext} . Since the A/D board must be able to collect a maximum of 2048 samples at a minimum rate of 1MHz, a minimum of 2.048ms is required for the timing signals control pulse width. Using the same equation to obtain the pulse width generated by the 74LS221, values of 50k Ω and 68nF are selected for R_{ext} and C_{ext} respectively giving

$$\begin{aligned} t_{pulse} &\approx 0.7(50k\Omega)(68nF) \\ &\approx 2.38ms \end{aligned}$$

4.2.5 FIFO/DSP Interface

4.2.5.1 DSP Board

A PC/c5x DSP board, developed by Loughborough Sound Images Inc, is used to implement the FAM algorithm. The cyclic spectrum estimates are computed by a Texas Instruments TMS320c51 fixed-point DSP chip which is contained on the DSP board. In this implementation, the 50-pin connector of the DSP board (DSPLINK) provides a hardware interface to the A/D board that includes access to its address and data lines, I/O signals (/IOE, RNW), and flags. The DSP board is contained in an IBM-compatible PC and interfaces to the PC through I/O and memory-mapped (status and control) registers. In addition, a dual-port (DP) RAM is included on the DSP board and can be accessed by either the DSP or the PC without interrupting the DSP operation. If the same location in the DP memory is accessed simultaneously by the DSP and PC, arbitration is provided and the access cycle for either the DSP or PC is lengthened, for example, via wait states.

4.2.5.2 Data Transfer - Read Signal Generation

When the DSP is monitoring the FIFO flags or is ready to read the data, it generates an I/O read cycle by pulling the /IOE (input-output enable) line low for a minimum of 120ns as shown in Figure 4.13. The address lines are valid at least 10ns before the /IOE goes low and remain valid at least 10ns after the /IOE goes high. The data or flags from the FIFO must be valid and present on the data bus lines at least 30ns before the rising edge of the DSP /IOE signal.

Address lines, /IOE, and RNW signals are brought onto the A/D board through a 74LS245 data buffer. They are decoded using a 74LS138 3-to-8 decoder to generate the required signals to retrieve the I and Q data from the FIFO. As shown in Figure 4.14, three address lines are used as input signals to determine the appropriate port or output line. The 74LS138 is enabled by the RNW and /IOE signals. The /IOE signal undergoes the greatest amount of delay in the 74LS138 and becomes the critical signal for the timing requirements of the read signal and FIFO data valid. When a FIFO read (/R) is generated from the 74LS138, valid FIFO output data is present no later than 25ns after the falling edge. The data lines pass through data buffers and are presented on the data bus lines of the DSP. If maximum delays are assumed for all component outputs, Figure 4.13 shows that the data is ready on the bus lines at least 39ns before the rising edge of the DSP /IOE. This fully satisfies the 30ns minimum requirement in the DSP specifications.

Figure 4.13 Timing diagram for READ operation

Figure 4.14 FIFO Read signals and buffer enable

4.2.5.3 Status Flags Monitoring

After the DSP initiates a master reset to start the data collection process, it waits until the FIFOs are full before reading any data into its own memory. As a result, the DSP continually monitors the FIFO flags for the activation of the FIFO full flag during the A/D operation. As described above, a read cycle is initiated after which the /IOE signal goes low. The relevant data lines (in this case, FIFO flags) must be valid on the data bus at least 30ns before the /IOE goes high. The FIFO flags are buffered through two 74LS245s, the first of which is enabled by one of the 74LS138 decoder output ports. A maximum propagation delay of 12ns is given for the data buffer. Again, assuming worst case delays, FIFO flags are valid on the data bus at least 52ns before the rising edge of the DSP /IOE as shown in Figure 4.15, thereby satisfying the required data setup time for a DSP read operation.

Figure 4.15 Timing diagram for reading FIFO flags

Figure 4.16 is a photograph of the hardware for the receiver/downconverter and A/D Board portions of the experimental setup.

Figure 4.16 Hardware for receiver/downconverter and A/D board

4.2.6 DSP/PC Handshaking

An IBM-compatible 486 PC is used to house the PC/c5x DSP board. The PC loads the DSP program and resets the DSP board to start the FAM computations. When the DSP indicates that the computations have been completed, the PC formats and transfers the results of the FAM computations into an ASCII file. The ASCII file is used by PV-Wave, a 3-dimensional graphics software package, to plot the cyclic spectrum estimates for analysis.

4.3 Software Implementation

4.3.1 FAM Computation Overview

The assembler program developed for the FAM computations is divided into three modules: data pre-processing, computation of frequency components, and computation of cyclic spectrum estimates. A flow chart showing the main functions of the assembler program according to these three modules is given in Figure 4.17. A source code listing which includes all invoked subroutines is included in Appendix C.

At the start of the assembler program for the FAM computations, the following parameters are initialized: the length of the input sequence (N), the length of the short-time FFT window (N'), and the overlap parameter $\left(L = \frac{N'}{4}\right)$. Due to the memory limitations of the DSP board, a value of $N = 1024$ samples of I and Q components is selected for analysis. The length of the short-time FFT is chosen to be $N' = 32$ to satisfy the estimate reliability factor requirement discussed in Chapter 3 $\left(\frac{N}{N'} = 32\right)$. The overlap parameter is, thus, $L = 8$.

A master reset of the A/D board is initiated by the DSP. When the DSP receives a signal indicating that the FIFOs are full, the data collected on the A/D board is read into DSP board memory. Samples are read in as 12-bit signed numbers using offset binary coding and are converted to 2's complement representation. The input sequence is then zeropadded so that the length of the second sequence, P , will be a power-of-2 to facilitate the use of an FFT Radix-2 routine.

Figure 4.17 Flowchart of assembler program

A series of overlapping short-time FFTs is performed to obtain the spectral components of the input sequence. The output of each short-time FFT is stored in an array. All frequency components are normalized to 16 bits to reduce the memory requirements when products are subsequently formed from these frequency components and stored.

Complex product sequences are formed by multiplying two spectral components together over all the short-time FFTs or time hops. A second FFT is performed on this sequence to obtain the cyclic spectrum estimates for a particular value of temporal frequency and range of cycle frequencies. The estimates are then stored in DSP memory. This process is repeated for all possible combinations of spectral components or channel pair regions. The DSP then sends a signal to the PC to indicate that the cyclic spectrum estimates are ready to be transferred.

4.3.1.1 Data Pre-processing

A master reset cycle is initiated by the DSP to reset the FIFO memory flags and set up the timing circuits for data collection. When data collection has been initiated, the DSP continually reads the FIFO status until the FULL flag is activated. The DSP then transfers the data from the FIFOs to DSP memory.

As mentioned in the previous section, digitized data is represented using 12-bit offset binary coding. Since the DSP operations are based on 2's complement data, the digitized data must be converted. The value 2048 (800h) is thus subtracted from each sample read into the DSP before being stored.

In Section 3.4.2.2, it was determined that with the decimation of the input sequence, the length of the second sequence, P , is calculated as follows,

$$P = \left(\frac{N-N'}{L} + 1 \right)$$

Since an FFT will be performed on the second sequence to obtain the cyclic spectrum estimates, it is useful to have P as a power-of-2. To ensure a power-of-2 length for the second sequence, the above equation is evaluated and the result is rounded up to the nearest power-of-2. The number of zeroes that must be added to the input sequence to obtain this rounded value of P is then computed. The number of zeroes is then doubled to account for both the I and Q components and added to the end of the input sequence.

4.3.1.2 Frequency Components Computation

The frequency components of the input sequence that are used to obtain cyclic spectrum estimates are computed and stored in DSP memory. A pointer is hopped through the input sequence at intervals of the overlap parameter, $L = 8$, to locate the first sample of each short-time FFT sequence. A counter is used to identify the time hop number and count up to the number of time hops required to cover the entire input sequence. The number of hops required to cover the sequence is equal to the length of the second sequence, P , which was previously calculated. At each time hop, N' complex samples are moved into a section of the DSP data memory that is used as an FFT work area and then tapered with a Hamming window [11]. As discussed in Chapter 3, the application of a data taper window reduces the sidelobes and eliminates the occurrence of gaps in the bifrequency plane. The Hamming window coefficients, which were calculated beforehand, are stored as a table in the

DSP program memory. A. N' -point FFT is then performed on the tapered sequence.

A general purpose FFT subroutine was implemented to perform arbitrary 2^M length FFTs as determined by certain variables. These variables include the size of the FFT, the number of stages, the spacing between twiddle factors provided in a table in DSP program memory, and the location of the FFT input data. The FFT routine used is a decimation-in-frequency (DIF) Radix-2 algorithm [10]. An example of the flow diagram for an 8-pt FFT is shown in Figure 4.18a. The result of the DIF Radix-2 algorithm consists of frequency bins in bit-reversed order. In this implementation, a routine is included to rearrange the frequency bins into normal order before storing them to allow for a simpler indexing scheme in subsequent operations. Figure 4.18b contains the Fortran subroutine for the DIF algorithm from which the assembler code was derived.

After the frequency components are computed, a second sequence is created by multiplying two frequency components over all the time hops. When the TMS320C51 DSP chip performs a multiplication of two 16-bit numbers, the result, stored in the P (product) register, is a 32-bit number. Retaining all 32 bits of the P register increases addressing complexity, increases computation time and requires significantly larger memory for subsequent processing and data storage. Instead, a strategy is adopted to store only a 16-bit word for a product and to do so with minimal degradation in accuracy.

2 of/de 2

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

(a)

SUBROUTINE DIFFFT(X,NU)

COMPLEX X(1024),U,W,T

N=2**NU

PI=3.14159265358979

DO 20 L=1,NU

LE=2**(NU+1-L)

! F1=LE/2

U=(1.,0.)

W=CMPLX(COS(PI/FLOAT(LE1)), -SIN(PI/FLOAT(LE1)))

DO 20 J=1,LE1

DO 10 I=J,N,LE

IP=I+LE1

T=X(I)+X(P)

X(IP)=(X(I)-X(P))*U

10 X(I)=T

20 U=U*W

(b)

Figure 4.18 DIF Radix-2 FFT algorithm (a) Flow diagram for 8-pt FFT (b) Fortran Subroutine used to derive assembler code

The approach begins by recognizing that the data transferred from the FIFOs and subsequently converted to 2's complement representation forms a 16-bit word with 11 data bits and sign extension. Since the FFT routine performs a divide-by-2 following each butterfly stage the values of the output frequency bins is at most 11 bits long. With a limit of 11 bits of data, the multiplication of two frequency bins will yield a product with a maximum of 22 bits of data. The TMS320c51 uses 16-bit words; consequently, either the least significant 7 bits are dropped to yield a 16-bit word or two 16-bit words are stored with the disadvantages described above.

The assembler program includes a subroutine to normalize the array of frequency bin components by searching the array for the largest value and determining the appropriate shift required to normalize this maximum value to 16 bits. Since the frequency magnitudes are at most 11 bits long, the minimum shift is 4 bits. The same shift is applied to all frequency components. When the normalized frequency components are complex-multiplied the product is "shifted" to the left as shown in Figure 4.19 and only the upper 16 bits are stored. In the worst case, when the value of both frequency components is 11 bits long, the product, including the sign bit, is 23 bits long. The truncation of the product register will generate a loss of 7 bits of accuracy. The loss implies that components which are 96dB below the peak value can be omitted. Furthermore, from Section 4.2.3.1, the signal dynamic range is mapped in the middle of the ADC dynamic range so that samples, and hence the frequency components, will be coded in less than 11 bits. As a result, less than 7 bits of accuracy is lost during the truncation of the product register. Thus, by normalizing the frequency bins, the memory requirements for the cyclic spectral estimates are reduced with minimal loss of accuracy.

Figure 4.19 Frequency bins normalization

4.3.1.3 Cyclic Spectrum Estimates Computation

The final component of the FAM algorithm is the computation of the cyclic spectrum estimates. Since the bifrequency plane is symmetric about the $\gamma = 0$ axis, it is only necessary to compute the cyclic spectrum estimates for either the positive or negative γ 's. As described in Section 3.4.2.3, cyclic spectrum estimates are computed in blocks designated by channel pair region. The estimates for a particular channel pair region are obtained from the Fourier transform of a product sequence,

formed by complex-multiplying two spectral components from the input sequence for every short-time FFT. Each channel pair region is assigned two indices, (a,b) , which identify the two spectral components being multiplied to form the product sequence.

$$\begin{aligned} &\text{Product sequence for} \\ &\text{channel pair region } (a,b) = p_{a,b}[m] = X_N(n+mL,a)X_N^*(n+mL,b) \end{aligned}$$

The samples of the product sequence, in fact, represent estimates of the discrete cyclic periodogram for a particular k_o and γ_o as defined in (33). Recall, that k_o and γ_o are the discrete temporal and cycle frequencies, respectively. They are related to a and b as follows,

$$\begin{aligned} a &= k_o + \frac{\gamma_o}{2} \\ b &= k_o - \frac{\gamma_o}{2} \end{aligned}$$

Recall from Section 3.4.1 that an exponential factor resulting from the time-shift property of the Fourier transform was introduced. The exponential term is required to preserve the coherence of the spectral components between successive short-time FFT hops. However, instead of applying the exponential factor to each of the frequency bins before the complex-multiplication, it is more computationally efficient to apply it after the product is formed,

$$\begin{aligned} &\overline{X_N(n+mL,k+\frac{\gamma}{2})} \overline{X_N^*(n+mL,k-\frac{\gamma}{2})} \\ &= X_N(n+mL,k+\frac{\gamma}{2}) e^{\frac{-i2\pi(k+\frac{\gamma}{2})mL}{N'}} X_N^*(n+mL,k-\frac{\gamma}{2}) e^{\frac{i2\pi(k-\frac{\gamma}{2})mL}{N'}} \\ &= X_N(n+mL,k+\frac{\gamma}{2}) X_N^*(n+mL,k-\frac{\gamma}{2}) e^{\frac{-i2\pi\gamma mL}{N'}} \end{aligned}$$

Delaying the phase adjustment, $e^{-i2\pi\gamma mL/N'}$, to this point allows time-independent FFTs to be used and saves a complex-multiplication operation.

When the product sequence, including the exponential factor, is generated, a P -point FFT is performed with respect to the discrete cycle frequency, γ . The same general purpose FFT routine that was used for the N' -point FFTs can be used to compute the P -point FFT with the appropriate variables initialized. Recall that zeropadding the input sequence ensures that P is a power-of-2. The output of the P -point FFT provides fine resolution in the cyclic frequency domain. The arrangement of the P output bins over the channel pair region is illustrated in Figure 4.20.

Section 3.4.2.3 described that in order to maintain statistical reliability, only estimates situated the equivalent in γ of $\pm \frac{\Delta k}{2}$ from the centre of the channel pair region are considered valid. This corresponds to $\frac{P}{2}$ output bins for an overlap parameter of $L = \frac{N'}{4}$ and a data taper window of $\beta = 2.0$. In this implementation, a Hamming window ($\beta = 1.81$) is used to taper the input data. As Figure 4.20 shows, the slightly smaller channel pair region causes the consideration of $\frac{P}{2}$ output bins to deviate slightly from the $\pm \frac{\Delta k}{2}$ boundary for statistical reliability. However, the deviation is acceptable in view of the simplification of the routine when dealing with $\frac{P}{2}$ bins instead of, say, $\left(\frac{P}{2} - 3\right)$ bins.

The $\frac{P}{2}$ output bins about the centre of the channel pair region in Figure 4.20 correspond to bins $\left(\frac{3}{4}P\right)$ to $(P-1)$ and bins 0 to $\left(\frac{P}{4} - 1\right)$. At the output of the P -point FFT, the bins are again in bit-reversed order. Rather than rearrange all the output bins just to save half in the end, a routine was developed to rearrange and

store only the top and bottom quarter of the bins.

Figure 4.20 Resolving cyclic spectrum components in the channel pair region

The above procedure is repeated for all channel pair regions to obtain the cyclic spectrum estimates for the entire bifrequency plane. Due to the manner in which the frequency components are stored and the implementation of the index loops, the order of the channel pair regions for which estimates are computed is as

illustrated in Figure 4.21. The indices are stored in an array to be used later by the C-program to locate the cyclic spectrum estimates. This is discussed in the next section.

Figure 4.21 Channel pair regions for positive γ (a) Order in which channel pair regions are computed (b) Order in which channel pair regions are stored in memory

4.3.2 Hardware/User Interface

The hardware/user interface is provided through a PC-based program written in C. A copy of the source code is included in Appendix C. The main functions of the program are outlined in the flowchart in Figure 4.22. When the C-program is executed, the compiled assembler code is first loaded onto the DSP board and run. The loading and execution of the assembler program is carried out by subroutines which have been provided by the developer of the DSP board, Loughborough Sound

Images, Inc.

Figure 4.22 Flowchart of hardware/user interface program

The PC continually monitors a status register located in the DSP's DP RAM until it is set by the DSP indicating the completion of the FAM computations. Due to

the restrictions on the size of the 2-D array that can be defined, the cyclic spectrum estimates must be transferred to the PC in blocks. A set of variables and indices is read into an array defined in the PC program. These variables and indices are used to determine the location of the block of cyclic spectrum estimates to be transferred. The estimates are transferred to an intermediate 2-D array which represents a "slice" of values on the bifrequency plane. The estimates are also multiplied by a compensation factor to reduce the amplitude distortion caused by the data taper window. This is discussed further in Section 4.3.2.3. When the intermediate array is filled with the appropriate estimates, it is written into an ASCII file and the process is repeated until all the cyclic spectrum estimates have been transferred.

4.3.2.1 Variables and Indices Transfer

After the completion of the FAM computations, the PC reads two variables stored by the DSP in the DP area: the lengths of the first and second FFTs, N' and P , respectively. These variables are used to determine the number of channel pair regions in the bifrequency plane as well the coordinates of the final 2-D table of cyclic spectrum estimates stored in the ASCII file.

Due to the order in which the estimates for each channel pair region are stored in contiguous memory (see Figure 4.21), a routine was developed to map the estimates into a more convenient format for graphical analysis. An array containing the indices of the channel pair regions and the discrete cycle frequency corresponding to the centre of the channel pair regions is transferred from DSP memory. These values are used to locate the particular block of estimates to be transferred.

Figure 4.23 (a) Bifrequency plane partitioned into "slices" with non-overlapping channel pair regions (b) Portion of one "slice" with overlapping channel pair regions (data taper with Hamming window)

As a result of the restrictions on the size of the 2-D array that can be derived in the C-program, an iterative process is required to transfer the estimates to the PC. The bifrequency plane is partitioned into "slices" as shown in Figure 4.23. Each slice covers the entire temporal frequency range and one increment of the discrete cycle

frequency. Consequently, a smaller 2-D array can be defined and a loop can be implemented to transfer the estimates one slice at a time. Referring to Figure 4.23(a), a "slice" is composed of estimates from adjacent channel pair regions, alternating between the top and bottom halves. When a Hamming window is used, causing the channel pair regions to overlap, the estimates from each channel pair region used to build the "slice" are shown in Figure 4.23(b). It is observed that by specifying the slice to cover one increment of the discrete cycle frequency, the estimates transferred are consistent with those which satisfy the reliability condition. At every iteration, the 2-D array is initialized with zeroes to ensure that regions outside the triangular-shaped bifrequency plane are represented by zeroes. Cyclic spectrum estimates are then retrieved using the indices array and stored in the appropriate location in the 2-D array.

4.3.2.2 Window Effects Compensation

Figure 4.24 illustrates the process of multiplying two frequency components to create a product window. When the product window is further resolved into finer cyclic components, the bins located away from the centre of the channel pair region (i.e. the peak of the product window) are attenuated by a factor of the data taper spectral coefficients squared. To compensate for this effect, the data taper spectral coefficients are computed, squared, and stored by the PC in an array. As the block of estimates from a particular channel pair region are transferred to the 2-D array, each estimate is multiplied by the reciprocal of the corresponding coefficient.

Figure 4.24 Frequency product window

4.3.3 Graphical Analysis

The ASCII file generated by the PC which contains the cyclic spectrum estimates are transferred to a SUN workstation for graphical analysis. The cyclic spectrum estimates are displayed using a 3D graphics software package called PV-Wave.

CHAPTER V

FAM ALGORITHM TESTING AND RESULTS

5.1 Introduction

The experimental setup described in Chapter IV was built and used to test the FAM algorithm which was implemented on a DSP board. Comments about the test procedure are included, along with a table outlining the digital modulation schemes considered. The results obtained from the cyclic spectrum analyser for both experimental and simulated input data are presented. Experimental and theoretical results are compared and discussed. Observations on the suitability of the FAM algorithm for signal detection are made where appropriate.

5.2 Test Procedure

An outline of the digital modulation schemes and range of E_b/N_o used to test the implementation of the FAM algorithm is given in Table 5.1. Simulated signal waveforms were used to verify the implementation of the algorithm and for cases where multiple signal generators were required to produce the test scenario of interest.

For the signal-plus-noise cases, the parameter E_b/N_o is used. The value of N_o is obtained by dividing the noise power level (N_p), calculated in Section 4.2.3.1, by the bandpass filter bandwidth (5MHz or 1MHz). The noise generator is

1. CW	a. Two in one frequency bin (simulated)
	b. Two in separate frequency bins (simulated)
2. BPSK	a. Single BPSK signal (simulated)
	b. Two BPSK signals (simulated)
	c. One complex BPSK signal (experimental)
	d. One real BPSK signal (experimental)
	e. BPSK+AWGN, $E_b/N_o = 10dB$
	f. BPSK+AWGN, $E_b/N_o = 0dB$
3. QPSK	a. One real QPSK signal (experimental)
4. FH and Burst	a. FH/MFSK (experimental)
	b. FH/PSK (experimental)
	c. Burst PSK (experimental)

Table 5.1 Test Scenarios for FAM Algorithm

adjusted until the desired noise spectral density is observed in the middle of the bandwidth on the spectrum analyzer (Figure 4.2). At the same monitor port E_b is found by measuring the equivalent carrier power of the signal on the spectrum analyzer and dividing by R_b for the BPSK examples. For the frequency-hopped and burst signals, the peak-to-peak voltage was measured on the oscilloscope, converted into power terms, and divided by the data rate to obtain E_b .

As indicated in Section 4.3.1, the FAM algorithm was implemented with the following parameters: length of input sequence, $N = 1024$; length of short-time FFT

window, $N' = 32$; and overlap parameter, $L = 8$. It is noted that for the presentation and discussion of the discrete cyclic spectrum estimates obtained from the FAM computations, f and α are used to refer to the temporal and cycle frequency respectively. In addition, the results of the FAM computations are presented only for positive values of the cycle frequency as the cyclic spectrum is symmetric about the $\alpha = 0$ axis.

5.3 Experimental Results and Discussion

5.3.1 Verification of FAM Algorithm Implementation

To verify the implementation of the FAM algorithm on the DSP board, computer-generated input data representing the sum of two continuous wave (CW) signals was used. The results are shown in Figure 5.1. In the first instance (Figure 5.1 (a)-(c)), the carrier frequencies of the two signals are located in separate FFT output frequency bins. Spectral correlation is observed at a cycle frequency equal to the difference between the two carrier frequencies, centred in temporal frequency about the two CW signals. This is consistent with the result obtained from Example 1 in Section 2.6. Figure 5.1(d)-(f) show the cyclic spectrum for two CW signals which are in the same FFT output frequency bin. If the power spectrum (along the $\alpha = 0$ axis) is considered, it only reveals that a signal is present in the band of interest. However, the cyclic spectrum reveals that within the frequency bin, there exists two signals, separated in frequency by a value α , corresponding to the location of the cyclic component. This illustrates the potential of cyclostationary analysis for detecting the presence of two or more signals where it is difficult using the conventional power spectrum.

Figure 5.1 Cyclic Spectrum for two CW signals (a) CW signals in separate frequency bins (b) Separate frequency bins - front view (c) Separate frequency bins - side view (d) CW signals in same frequency bin (e) Same frequency bin - front view (f) Same frequency bin - side view

5.3.2 Cyclic Spectra for Common Digital Modulations

5.3.2.1 BPSK Waveform

The experimental results for a complex BPSK input signal are shown in Figure 5.2. Using a sampling frequency of 6.4MHz, with FAM parameters N and N' equal to 1024 and 32 respectively, temporal and cycle frequency resolutions of 200kHz and 6.25kHz are obtained. As shown in [1], the expected cyclic component on the cyclic spectrum is the bit rate of the signal. Both a high and low data rate were chosen to investigate their effects on the cyclic spectrum.

In the case of a low bit rate (Figure 5.2(a)), the cyclic spectrum shows no distinguishable component. Accordingly, when AWGN is added, as shown in Figure 5.2(b) and 5.2(c), it becomes difficult to assess whether any signal is present in the band of interest. For a higher data rate BPSK waveform, the cyclic component appears at a value of α equal to the data rate (Figure 5.2(d)-(f)). Only when $E_b/N_0 = 0\text{dB}$ does the cyclic component become concealed by the correlation of the frequency components from the bandlimited noise signal. The results for the two data rates differ because fewer bits are observed over the same observation interval for the low R_b case due to the corresponding longer bit period. With a small number of bits, the BPSK signal appears as a complex CW signal which exhibits no spectral correlation. Furthermore, the absence of any recognizable cyclic component in Figure 5.2(b) and 5.2(c) is a result of the relatively small SNR within the frequency bin for the values of N and N' that were implemented in the FAM algorithm. Conversely, for the higher bit rate, an increase in the effective SNR coupled with the increase in the number of bits acquired allows the cyclic component to be extracted. This example demonstrates the impact of the values of N and

Figure 5.2 Cyclic spectrum for a BPSK signal in AWGN (a) $E_b/N_o > 30\text{dB}$, $R_b = 128\text{kbps}$ (b) $E_b/N_o = 10\text{dB}$, $R_b = 128\text{kbps}$ (c) $E_b/N_o = 0\text{dB}$, $R_b = 128\text{kbps}$ (d) $E_b/N_o > 30\text{dB}$, $R_b = 1.024\text{Mbps}$ (e) $E_b/N_o = 10\text{dB}$, $R_b = 1.024\text{Mbps}$ (f) $E_b/N_o = 0\text{dB}$, $R_b = 1.024\text{Mbps}$

N' on the cyclic spectrum estimates.

Taking into account the factors described above, it was of interest to determine whether two BPSK signals with different data rates could be detected using the FAM algorithm. The input data for the two BPSK signal case was simulated with two different rates. The carrier frequencies of both signals were selected to be very close together, making it difficult to confirm the presence of the two signals using the power spectrum alone. The results of the FAM computations for the two signals alone and for both BPSK signals are shown in Figure 5.3. The cyclic spectrum of the two BPSK signal case shows that the two data rates are identifiable. Furthermore, they are centred about different temporal frequencies, thereby demonstrating the association of the data rate with the carrier frequency of the signal.

5.3.2.2 QPSK Waveform

The cyclic spectrum for both a real (I component only) BPSK signal ($R_b = 1.024 \text{ Mbps}$) and real QPSK signal (symbol rate, $R_s = 512 \text{ kbps}$) are computed and compared in Figure 5.4(a)-(b). The first noticeable feature of the cyclic spectrum is the width of the main lobe along the $\alpha = 0$ line. The narrower lobe shown for the QPSK input waveform illustrates the spectral efficiency of this modulation scheme by transmitting two bits simultaneously on quadrature carriers. It is also observed that in the QPSK example, there is no spectral correlation associated with the carrier frequency, f_c , located at $f=0$, $\alpha = 2f_c$, unlike the BPSK signal. The cyclic component is cancelled out by the in-phase and quadrature components in the QPSK waveform. This effect allows a BPSK signal to be differentiated from a QPSK signal. The non-zero values surrounding the area is related to the variance of the estimate as

a result of $\frac{N}{N'}$ not being very large.

Figure 5.3 Cyclic Spectrum for two BPSK signals (a) BPSK Signal 1 alone (b) BPSK Signal 2 alone (c) Combined BPSK signals

Figure 5.4 Cyclic Spectrum for BPSK and QPSK waveforms (a) Real BPSK signal, $R_b = 1.024 \text{ Mbps}$ (b) Real QPSK signal, $R_s = 512 \text{ kbps}$

5.3.2.3 FH/MFSK Waveform

The cyclic spectrum for a FH/MFSK signal is shown in Figure 5.5. For this case, a 1MHz bandpass filter was used. The sampling frequency was reduced to 1.024MHz, giving temporal and cycle frequency resolutions of 32kHz and 1kHz respectively for the implemented FAM algorithm parameters. Both I and Q components of the downconverted signal are collected and used in the FAM computations.

In Figure 5.5(a), several frequency hops are shown to be present in the form of distinct peaks at different carrier frequencies along the $\alpha = 0$ axis. In the bifrequency plane, it is observed that no spectral correlation occurs between the carrier frequencies of the hops. This result deviates from the theoretical result (Example 3 in Section 2.6) which indicated that correlation occurs between frequency hops at a cycle frequency equal to the difference between their carrier frequencies. The difference between the theoretical and experimental results can be attributed to the manner in which the FAM algorithm computes the cyclic spectrum estimates. For the FH/MFSK signal, the frequency hops observed in Figure 5.5(a) are generated by a single source and hence occur at different time intervals. Consequently, two or more frequency hops are unlikely to be present together over many short-time FFTs in the FAM computation. As a result, when a product sequence is formed for the second FFT, the only non-negligible products are those centred about the individual hops. With no apparent correlation exhibited between any two frequency hops in Figure 5.5, it is likely that the interval between hops is larger than the time span of a single short-time FFT. Although this appears to be a weakness in the FAM method for computing cyclic spectrum estimates, the characteristic observed in fact provides

additional information about the signal(s) present in the frequency band. It may be used to distinguish between CW signals and FH or short-duration signals.

Another feature of the cyclic spectrum in Figure 5.5(b) is the definite pattern observed for the frequency hops. By viewing the cyclic spectrum from the side (Figure 5.5(c)), it is observed that the nulls for a hop occur at the hop rate. What the profile of the cyclic spectrum reveals is the outline of the $T_h^2 \text{Sinc}^2(fT_h)$ function obtained by taking the Fourier transform of the autocorrelation of the $\text{Rect}(t/T_h)$ function which defines the FH/MFSK waveform (Example 2 in Section 2.6).

5.3.2.4 Burst PSK and FH/PSK Waveforms

The results of the FAM computations for a burst PSK waveform are shown in Figure 5.6. Although the view of the cyclic spectrum in Figure 5.6(a) does not reveal any distinct cyclic features, the profile identifies discrete components at integral multiples of the hop rate (Figure 5.6(b)). It is also observed that the discrete components diminish in magnitude as α increases. This effect is the result of the superposition of the Sinc^2 function associated with the rectangular pulse for the bit period as described in Example 4 in Section 2.6.

The FH/PSK signal is similar to the burst PSK signal with the exception that the frequency hops are located at random frequencies in the hopping bandwidth. The cyclic spectrum for the FH/PSK waveform is shown in Figure 5.7. Like the burst PSK example, the cyclic features of the spectrum are only evident when it is viewed from the side. Figure 5.7(b) reveals the similarity in the general shape of the spectrum (profile) between the frequency-hopped and burst cases. In addition, the

separation between the peak cyclic components is also approximately at the hop rate. However, these components appear "noisier" than for the burst PSK case due to the fact that they are obtained from a single frequency hop.

(a)

(b)

(c)

Figure 5.5 Cyclic Spectrum for a FH/MFSK signal (a) Entire cyclic spectrum for positive α (b) Partial spectrum for positive α (c) Profile of cyclic spectrum for FH/MFSK signal

Figure 5.6 Cyclic spectrum for a burst PSK waveform (a) Direct view of cyclic spectrum for positive α (b) Side view of cyclic spectrum for positive α

Figure 5.7 Cyclic spectrum for a FH/PSK waveform (a) Direct view of cyclic spectrum for positive α (b) Side view of cyclic spectrum for positive α

CHAPTER VI

SUMMARY AND RECOMMENDATIONS

6.1 Introduction

For this thesis, an analysis, implementation, and evaluation of a computationally-efficient algorithm for cyclic spectral analysis was carried out. The objective of this chapter is to provide a summary of each of these components. General remarks and conclusions pertaining to cyclic spectral analysis and implementation concepts are first presented. A performance summary of the computationally-efficient FAM algorithm is then provided in Section 6.3. Related areas of study are proposed in the final section.

6.2 Thesis Summary

An overview of cyclic spectral analysis concepts, developed in Chapter II, establishes a link between a signal exhibiting second-order periodicity and a cyclostationary process. The cyclic autocorrelation and cyclic spectral density functions, defined in Sections 2.4.1 and 2.4.2 respectively, are generalizations of the conventional autocorrelation and power spectral density used in stationary signal analysis. The cyclic autocorrelation function includes aspects of square-law detection and delay-multiply feature detection to exploit periodic components unique to a signal.

Techniques for estimating the cyclic spectrum are derived from time-smoothing and frequency-smoothing approaches that were introduced for power spectrum

estimation (Chapter III). The resulting cyclic spectrum estimate equation can be interpreted as a spectral correlation function. In order for estimates to be reliable, Grenander's Uncertainty Condition implies that a fair amount of smoothing is required to reduce the effects of random fluctuations of the signal.

An analysis of a brute force approach for computing cyclic spectrum estimates in Section 3.4.1 illustrates the considerable computational requirements. These requirements are further emphasized by the fact that zeropadding is needed to eliminate the occurrence of gaps in the bifrequency plane and thus the possibility of a cyclic component being missed.

The FAM algorithm is a computationally-efficient time-smoothing algorithm for estimating the cyclic spectrum and is described in Section 3.4.2. The algorithm effectively divides the bifrequency plane into smaller areas called channel pair regions and computes the cyclic spectrum estimates one block at a time. A block of estimates is obtained by performing an FFT on a sequence of products formed by correlating two frequency components over the length of the input sequence. To meet the requirements for estimate reliability, consideration must be given to the length of the input sequence, N , and the short-time FFT, N' . Specifically, the ratio of N to N' must be much greater than 1. The problem of gaps in the cyclic spectrum as noted in the brute force method is alleviated by using a data taper window on the short-time FFT input data.

6.3 FAM Algorithm Summary of Results

A hardware and software implementation was realized to evaluate the FAM algorithm for computing cyclic spectrum estimates. Common digital modulation schemes were selected to test the algorithm.

The results from the FAM computations confirm the potential of cyclic spectral analysis for detecting and identifying signals in a frequency band. Cyclic spectra presented in Chapter V were in general agreement with theoretical examples provided in Chapter II as well as with results in [1-2,14-16]. Examples where the symbol and frequency hop rates could be extracted from the cyclic spectrum estimates were shown. Furthermore, the association of the bit rate with a particular temporal frequency (i.e. carrier) in the two BPSK signal example illustrates that multiple signals can be separated and identified.

It was observed in the FH/MFSK example that a frequency-hopped (short-duration) signal could be distinguished from a CW signal when looking at the cyclic spectrum. This was due to the way in which the estimates are computed in the FAM algorithm.

Limitations of the algorithm due to the finite observation time were noted in some test cases. As an example, the short observation time caused the low R_b BPSK signal to resemble a complex CW signal which exhibits no spectral correlation at any cycle frequency. This discrepancy can be overcome by increasing the number of input data samples in the analysis. However, the improvement is achieved at the expense of computation time and memory requirements.

6.4 Areas for Future Work

In this thesis, an analysis of both time-smoothing and frequency-smoothing methods for computing the cyclic spectrum estimates was presented. It was mentioned that time-smoothing algorithms were more suited to cases where estimates for the entire cyclic spectrum was required. Conversely, a frequency-smoothing algorithm was preferable if only a relatively small portion of the bifrequency plane was of interest. It would be of interest to investigate the possibility of a hybrid-smoothing algorithm which combines both time- and frequency-smoothing techniques to see if any further computational efficiencies could be achieved. In situations where a general search is being carried out, a hybrid algorithm may be useful in reducing the number of estimates to be computed.

The ability for real-time processing of a received signal and the identification or characterization of any signal(s) present in the frequency band of interest is an important task performed in communications systems. The tremendous computational requirements of cyclic spectral analysis algorithms impedes this ability greatly. This was observed for the single DSP processor implementation described in this thesis, where the processing time required to collect approximately $156\mu\text{s}$ (1024 samples at 6.4MHz), compute the cyclic spectrum estimates, and format the results into a usable format was nearly 30 seconds. Proposals to implement these algorithms using more parallel architectures to distribute processing load and reduce processing time are encouraging, but still warrant further study. As an example, [5] proposed to implement multiple processors to compute the cyclic spectrum estimates by channel pair region. This would, of course, require a significant increase in hardware and software complexity.

The concepts and algorithms discussed in this thesis were based on second order cyclostationarity as described in Chapter II. It follows that higher order (third or fourth order) statistics may also provide information about a signal which could be used for detection and identification. Efficient algorithms would also be required to meet the increased complexity of these equations.

A Radix-2 FFT algorithm was used in this thesis to compute both short-time and product sequence FFTs. Recall from Section 3.4.2.3 that in the second FFT, only half of the output frequency bins were considered valid. An investigation into other algorithms for computing the Fourier transform may reveal additional computational savings for the reduced number of frequency bins.

REFERENCES

1. Gardner, W.A., *Statistical Spectral Analysis: A Non-probabilistic Theory*, Prentice-Hall, N.J., 1987.
2. Gardner, W.A., "Exploitation of Spectral Redundancy in Cyclostationary Signals", *IEEE SP Magazine*, April 1991, pp. 14-36.
3. Mudry, A.H., "The Exploitation of Spectral Correlation for the Purpose of Signal Search and Detection", DSTO Research Note ERL-0572-RN, Salisbury, Australia, September 1991.
4. Gardner, W.A., "Measurement of Spectral Correlation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 5, October 1986, pp. 1111-1123.
5. Roberts, R.S., Brown, W.A., and Loomis Jr., H.H., "Computationally Efficient Algorithms for Cyclic Spectral Analysis", *IEEE SP Magazine*, April 1991, pp. 38-49.
6. Kamen, E., *Introduction to Signals and Systems*, MacMillan Publishing Company, N.Y., 1987.
7. Stremler, F.G., *Introduction to Communications Systems*, 2nd Edition, Addison-Wesley Publishing Company, Reading, Ma, 1982.
8. Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, 3rd Edition, McGraw-Hill Inc., N.Y., 1991.
9. Peebles Jr., P.Z., *Probability, Random Variables, and Random Signal Principles*, 2nd Edition, McGraw-Hill Inc., N.Y., 1987.
10. Oppenheim, A.V., and Schafer, R.W., *Discrete-time Signal Processing*, Prentice-Hall, N.J., 1989.
11. Harris, F.J., "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform", *Proceedings of the IEEE*, vol. 66, no. 1, January 1978, pp.51-82.
12. Carlson, A.B., *Communications Systems: An Introduction to Signals and Noise in Electrical Communication*, 3rd Edition, McGraw-Hill Inc., N.Y., 1986.

13. Proakis, J.G., *Digital Communications*, 2nd Edition, McGraw-Hill Inc., N.Y., 1989.

Additional References

14. Brown, W.A., and Loomis Jr., H.H., "Digital Implementations of Spectral Correlation Analyzers", *IEEE Transactions on Signal Processing*, vol. 41, no. 2, February 1993, pp. 703-720.
15. Gardner, W.A., "Spectral Correlation of Modulated Signals: Part I - Analog Modulation", *IEEE Transactions on Communications*, vol. COM-35, no. 6, June 1987, pp. 584-594.
16. Gardner, W.A., Brown, W.A., and Chen, C.K., "Spectral Correlation of Modulated Signals: Part II - Digital Modulation", *IEEE Transaction on Communications*, vol. COM-35, no. 6, June 1987, pp. 595-601.
17. Gardner, W.A., "The Spectral Correlation Theory of Cyclostationary Time-Series", *Signal Processing*, vol. 11, no. 1, July 1986, pp.13-36.
18. *SPECTRUM Signal Processing Inc. PC/C5x Board User's Guide*, Document no. 500-00045, Version 1.05, February 1993.
19. *Texas Instruments' TMS320C5x User's Guide*, 2547301-9721 revision D, January 1993.

DSP READING FIFO FLAGS DURING DATA CONVERSION AND COLLECTION PROCESS

1. On falling edge of **/IOE**, **ENFLAG** (buffer enable for FIFO flags) enabled after delay of 20ns.
2. On falling edge of **ENFLAG**, data ready at buffer after $t_{PLH} = 30$ ns. Data ready on bus on rising edge of **/IOE**.
3. Data collection in process.
4. On falling edge of **FF_FIF**, timing circuit is disabled (**/OE** goes HIGH), and data collection is stopped (**/FIF_W** goes HIGH and stays HIGH). Delays shown on next timing diagram.
5. On second falling edge of **/IOE**, FIFO FULL flag valid on bus after $t_{PLH} = 50$ ns. Data ready on bus on rising edge of **/IOE**.

IQ AD Board Circuit Diagram (CRC W.O. no. 94-266)

Timing diagrams for the A/D Board
(obtained from HP1650B Logic Analyser)

MASTER RESET

1. On the falling edge of the MRESET signal, FIFO EMPTY and FULL flags are reset to LOW and HIGH respectively. $t_{PH} = 30\text{ns}$.
2. On the rising edge of the MRESET signal, timing circuits are enabled (CLKON). $t_{PH} = 30\text{ns}$.
3. On the rising edge of the MRESET signal, three timing signals in operation (ADC ENCODE, Latch 374CLK, FIFO /WRITE). No delay observed between ENCODE and 374CLK edges (Q and /Q of 74LS221). $t_{PH} = 10\text{ns}$ delay between ENCODE and /WRITE edges.
4. On first valid write on FIFO, EMPTY flag goes high 10ns after rising edge of WRITE signal.

A/D CONVERSION AND DATA COLLECTION OPERATION

1. On rising edge of ENCODE, ADC data out valid after $t_{PD}=90\text{ns}$.
2. On rising edge of CLK374 (Latch), data setup time $t_u=120\text{ns}$, data hold time $t_h=40\text{ns}$.
3. On rising edge of CLK374 (Latch), 374 data out valid after $t_{PHL}=20\text{ns}$.
4. On rising edge of /WRITE (FIFO), data setup time $t_u=100\text{ns}$, data hold time $t_h=60\text{ns}$.

DSP READING FIFO FLAGS DURING DATA CONVERSION AND COLLECTION PROCESS

1. On falling edge of `/IOE`, `ENFLAG` (buffer enable for FIFO flags) enabled after delay of 20ns.
2. On falling edge of `ENFLAG`, data ready at buffer after $t_{PIH} = 30\text{ns}$. Data ready on bus on rising edge of `/IOE`.
3. Data collection in process.
4. On falling edge of `FF_FIF`, timing circuit is disabled (`/OE` goes HIGH), and data collection is stopped (`/FIF_W` goes HIGH and stays HIGH). Delays shown on next timing diagram.
5. On second falling edge of `/IOE`, FIFO FULL flag valid on bus after $t_{PI} = 50\text{ns}$. Data ready on bus on rising edge of `/IOE`.

FIFO FULL

1. Timing signals active.
2. On last FIFO /WRITE (falling edge), /FFLAG (FIFO FULL flag) goes low after $t_{pf}=20\text{ns}$.
3. On falling edge of /FFLAG, timing circuit is disabled (/CLKON goes HIGH) after $t_{pf}=30\text{ns}$.
4. On rising edge of /CLKON, FIFO /WRITE goes HIGH after $t_{pf}=20\text{ns}$.

BETWEEN LAST FIFO WRITE AND FIRST DSP READ

1. FIFO /WRITE active until FIFO FULL flag goes LOW. For timing see FIFO FULL event.
2. On second /IOE falling edge, FIFO /READ is activated. Timing for FIFO Read is shown on next timing diagram.

DSP READING FIFO

1. On falling edge of /IOE, FIFO /READ goes LOW after $t_{PD} = 30ns$.
2. On falling edge of FIFO /READ, data out valid after $t_{PD} = 20ns$.
3. When FIFO data out valid, data presented on bus after $t_{PD} = 10ns$. Data ready on rising edge of /IOE.

Software Listings

```

*****
*           Program Name:      SPECTRM.ASM          *
*           Author:           C. Tom               *
*           Date:             06 Jul 1994          *
*           Description:       This program calculates estimates for *
*                               the cyclic spectrum using the FFT   *
*                               Accumulation Method (FAM).          *
*****

```

```

        .mmregs                ;defines global symbolic names for
                                ;TMS memory-mapped registers

```

```

*****
*           Initialisation and declaration of variables          *
*           Reservation of data memory                          *
*****

```

```

***** Initialised variables *****

```

```

***** The following block of variables are either chosen or determined *****
***** at the start *****

```

```

N_SAMP      .set      1024      ;no. of complex pairs of data
N_SAMP_minus_1 .set      1023
FFT1_LGTH   .set      32       ;N'=32, **N.B. Check FFT_DATA in .bss**
M_FFT1      .set      5        ;M=5 stages (32-pt FFT)
OVLAP_PAR   .set      8        ;OVLAP_PAR = L = N'/4
N_BINS_minus_1 .set      8191   ;(2*N'*P - 1)= (2x32x128 - 1)

```

```

PAGE0      .set      0
PAGE2      .set      2
zero       .set      0
one        .set      1
three      .set      3        ;3 bit rev is 1/2 + 1/4 = 3/4
SP_TWD128  .set      2
SP_TWD32   .set      8
SIN2COS    .set      64      ;distance between sin and cos values in table
                                ;N.B. only valid for current sin/cos table
COD_OFF    .set      2048     ;coding offset for A/D data
                                ;to convert to 2's compl
set_ndx_mask .set      4      ;mask 0000 0000 0000 0100
clr_ndx_mask .set      0fffh  ;mask 1111 1111 1111 1011
FLAGVAL    .set      1234h   ;flag value to PC to signal end of processing
PORT0      .set      0E000h  ;Addr of Port 0 in DSPLNK
PORT6      .set      0E006h  ;Addr of Port 6 in DSPLNK
PORT7      .set      0E007h  ;Addr of Port 7 in DSPLNK

```

```

***** Reserve Space in Data Memory *****

```

```

        .bss      L,1        ;L = OVLAP_PAR
        .bss      COUNTER,1   ;counter for MORE loop
        .bss      HOP_NUMBER,1 ;loop counter for MKSEQ2
        .bss      FLAGS,1     ;FLAGBITS from A/D board
        .bss      END_DATA,1  ;contains addr of last "raw" data sample
        .bss      NUM_FFT,1   ;no. of short time FFTs
        .bss      REMAINDER,1 ;raw data samp remaining with NUM_FFT
        .bss      FFT2_LGTH,1 ;length of second sequence (P sequence)
        .bss      M_FFT2,1    ;no. of stages for P-length FFT
        .bss      XTRASEQ2,1   ;no. of xtra samples req'd for P sequence
        .bss      XTRASEQ1,1   ;no. of xtra samples req'd for N sequence
        .bss      TOT_ZERO,1
        .bss      ODD,1
        .bss      ZEROB4,1     ;no. of zeroes to add before N sequence (I,Q)
        .bss      ZEROAFT,1    ;no. of zeroes to add after N sequence (I,Q)
        .bss      MV_CNT,1     ;no. of array elements to move
        .bss      SEQSTART,1   ;contains addr where next N' samples start
        .bss      NXT_BIN,1    ;contains addr of next available bin in FFT1_BINS
        .bss      IDX_BIN1,1   ;index of first freq bin to be multiplied

```

```

.bss    IDX_BIN2,1      ;index of 2nd freq bin to be multiplied
.bss    ADDR_B1,1      ;addr of first freq bin in FFT1_BINS
.bss    ADDR_B2,1      ;addr of 2nd freq bin in FFT1_BINS
.bss    CMPR_REG,1     ;holds value of Aux Reg Compare Reg
.bss    TMP1,1         ;used in FFT subroutine
.bss    TMPQ,1         ;used in FFT subroutine
.bss    NXTIWD,1       ;twiddle factor increment
.bss    FFTSIZ,1
.bss    NO_BFLY,1      ;no. of butterflies per stage in FFT
.bss    STAGES,1       ;no. of stages in FFT
.bss    DAT_LOC,1      ;addr of data to be FFTd
.bss    DATSEP,1       ;sep betw input data points for FFT
.bss    BFLY_CNT,1
.bss    SINADDR,1      ;addr pointer to sin table
.bss    OFFSET,1       ;offset from base address to input sample of interest
.bss    SINVAL,1       ;value of sin
.bss    COSVAL,1       ;value of cos
.bss    COEFF,1        ;value of hamming coefficient
.bss    SWPCTR,1       ;counter for bit reversal routine
.bss    BITRV_IDX,1    ;bit reverse index
.bss    NORM_IDX,1     ;normal index
.bss    XFERI,1        ;temp hold of I to be swapped
.bss    XFERQ,1        ;temp hold of Q to be swapped
.bss    SWPADDR,1      ;destination addr of swap
.bss    QUART_N,1      ;N/4
.bss    THREE_QUART_N,1 ; (3/4)N
.bss    BOTTOM_ADDR,1   ;addr of bin (3/4)N
.bss    MAXVAL,1       ;current max value of block
.bss    SHIFT_minus_1,1 ;shift to bring max value to 16 bits
.bss    TMPIH,1
.bss    TMPIL,1
.bss    TMPQH,1
.bss    TMPQL,1
.bss    IQ_PTR,1       ;IQ pointer
.bss    NXT_EST,1      ;addr of next available loc to store est
.bss    MAG_I,1        ;magn of I
.bss    MAG_Q,1        ;magn of Q
.bss    ADDR_SEQ2,1
.bss    PHASEINCR,1    ;for CORR_PHAS, value of phase factor table
.bss    N_PRIM,1
.bss    NPRIM_BY_2,1
.bss    IDX_B1,1
.bss    IDX_B2,1
.bss    HOLD_ADDR,1
.bss    ALPHA,1
.bss    M_TIMES_L,1
.bss    VALUE,1        ;TEMP VARIABLE FOR TEST (FAKEDAT)
.bss    DP_ADDR,1      ;temp variable for DP addr
.bss    XMEM,1         ;temp variable for INDICES array
.bss    FFT_DATA,64    ;32-pt (complex) FFT data array

```

* Initialisation of sections *

```

RESET      .sect    "I_VECTOR"      ;in .CMD file, load into prog mem loc 0h
            B        MAIN

            .sect    "ARRAYS"
FFT1_BINS   .set     3200h           ;short time FFT output bins for all "hops"
INDICES     .set     7800h           ;area to store channel pair indices, alphas

            .sect    "SEQ_2"
SEQ2        .set     0800h           ;work area for 2nd in-place FFT
RAW_DAT     .set     1000h           ;raw data read in from A/D board

            .sect    "RESULT"
SPEC_EST    .set     8000h           ;cyclic spectral estimates array

```

```

        .sect    "DP_Area"
SIGNAL    .set    3000h        ;addr of signal flag for PC
VARS      .set    3001h        ;variables to be passed back to PC

        .text
*****
*          Initialise C5x          *
*****

MAIN      CLRC     OVM          ;reset overflow mode
          CLRC     CNF          ;configure DARAM to data space
          LDP      #PAGE0       ;load data memory page pointer
          OPL      #set_ndx_mask,PMST ;set NDX bit to separate ARO,INDX,ARCR
          LDP      #PAGE2       ;adj data memory page pointer for .bss
          LACC     #OVLAP_PAR    ;store OVLAP_PAR in data memory, L
          SACL     L
          LACC     #FFT1_LGTH
          SUB      #one
          SACL     SWPCTR        ;init cntr for bit rev routine in MORE loop
          LACC     #FFT1_BINS    ;initialise NXT_BIN to point to next
          SACL     NXT_BIN       ;available freq bin
          LACC     #VARS
          SACL     DP_ADDR       ;initialise DP_ADDR to point to next
                                   ;available position in DP area

          LACC     #INDICES
          SACL     XMEM
          LACC     #SPEC_EST
          SACL     NXT_EST       ;initialise NXT_EST to point to next
                                   ;available cyclic estimate array location

*****
*          MASTER RESET OF FIFOs          *
*****

          CLRC     XF           ;FIFO reset active low (min 25 ns)
          NOP
          SETC     XF           ;release FIFO reset

*****
*          C5x waits until FIFOs from A/D board are filled up with raw data.          *
*****

WAITFF    IN       FLAGS,PORTO  ;read FIFO flags from port addr 0 (DSPLNK)
          APL      #0030h,FLAGS  ;check full flag bits
          BCND     WAITFF,NTC    ;keep waiting if FF bits still high

*****
*          FIFOs full, read in data          *
*****

          CALL     DATAIN       ;read in data from A/D
          CALL     ZEROPAD       ;to determine whether zeropadding needed

*****
*          First set of computations, short time FFTs.          *
*          It calls on other subroutines to:          *
*          a) move block of data into N' array          *
*          b) data taper the N' samples          *
*          c) perform the N'=32 point FFT          *
*          d) store output freq bins into memory          *
*****

          MAR      *,AR7
          LAR      AR7,FFT2_LGTH
          MAR      *.
MORE      SAR      AR7,COUNTER
          CALL     DATAMV        ;move data into N' array
          CALL     TAPERA        ;data taper N' samples, Hamming window
          CALL     FFT1_INIT

```

```

CALL    FFT
CALL    BITREV          ;bit rev routine for all bins (I,Q,I,Q,...)
CALL    SAVBINS         ;save FFT output bins
MAR     *,AR7
LAR     AR7,COUNTER
BANZ    MORE

*****
*               Modify data to avoid overflow               *
*****

CALL    FINDMAX
CALL    DO_SHFT

*****
*               Second set of computations, spectral estimate *
*****
CALL    SPECTR

*****
*               Transfer lengths of 1st and 2nd FFT to Dual Port Memory *
*****
MAR     *,AR7
LAR     AR7,DP_ADDR
LACC    #FFT1_LGTH
SACL    *+
LACC    FFT2_LGTH
SACL    *+
LACC    M_FFT2
SACL    *+
LACC    END_DATA
SACL    *+
SAR     AR7,DP_ADDR

*****
*               c5x finished computations. Send signal to PC *
*****
MAR     *,ARO           ;ARP <- 0
LAR     ARO,#SIGNAL     ;ARO <- Addr of signal flag in DP Memory
LACC    #FLAGVAL
SACL    *               ;send flag to DP Memory

STDBY    NOP
B        STDBY          ;C5x waits here

*****
*               SUBROUTINES SECTION                         *
*               Description of each subroutine provided at the end *
*****

*****
*               DATAIN                                     *
*****

DATAIN    MAR     *,ARO           ;ARP <- 0
          LAR     ARO,#RAW_DAT   ;ARO<- points to addr of data array
          LACC    #N_SAMP_minus_1 ;ACC <- (N-1)
          SAMM    BRCR          ;Block repeat count register <- ACC
          RPTB    END_BLK-1
          IN      *,PORT6       ;read I from port addr 6 (DSPLNK)
          LACC    *             ;ACC <- I
          SUB     #COD_OFF       ;ACC <- I - COD_OFF, COD_OFF=2048
          SACL    *+            ;store 2's compl I, move pointer
          IN      *,PORT7       ;read Q from port addr 7 (DSPLNK)
          LACC    *             ;ACC <- Q
          SUB     #COD_OFF       ;ACC <- Q - COD_OFF, COD_OFF=2048
          SACL    *+            ;store 2's compl Q, move pointer

END_BLK    MAR     *-           ;readjust pointer ARO

```


[illegible]

```

LAR    ARO,#FFT_DATA    ;ARO <- dest addr
LACC   SEQSTART
SAMM   BMAR             ;Block move memory register <- srce addr
RPT    #63              ;move 64 elements (32 I,Q)
BLDD   BMAR,*+          ;on rpt, BMAR incr automatically
ADD    #OVLAP_PAR,1     ;move pointer to start of next FFT "hop"
                        ;ACC should still contain SEQSTART
                        ;shift of 1 for I,Q

SACL   SEQSTART
RET

```

```

*****
*                               TAPERA                               *
*****

```

```

TAPERA    SPM    1          ;shift product by 1
          MAR    *,AR1
          LAR    ARO,#FFT_DATA ;ARO <- points to data to be tapered
          LAR    AR1,#FFT1_LGTH ;AR1 <- counter (N'-1)
          MAR    *-,ARO
          LACC   #HAM_32     ;ACC <- addr of hamming coefficient
NEXTA     TBLR   COEFF       ;load in coeff
          ADD    #one        ;ACC <- points to next coeff
          LT     *           ;T <- data(I)
          MPY    COEFF        ;P <- data(I) * coeff (x2 with shift)
          SPH    *+          ;data(I) <- P, move to data(Q)
          LT     *           ;T <- data(Q)
          MPY    COEFF        ;P <- data(Q) * coeff (x2 with shift)
          SPH    *+,AR1      ;data(Q) <- P, move to data(I), ARP <- 1
          BANZ   NEXTA,*-,ARO
          RET

```

```

*****
*                               FFT1_INIT                           *
*****

```

```

FFT1_INIT LACC   #SP_TWD32    ;ACC <- sp betw twid factors in table
          SACL   NXTTWD      ;store as next twid factor increment
          LACC   #FFT1_LGTH
          SACL   FFTSIZ
          SACL   NO_BFLY
          LACC   #M_FFT1     ;ACC <- no. of stages for 16-pt FFT = 4
          SACL   STAGES      ;store no. of stages
          LACC   #FFT_DATA
          SACL   DAT_LOC     ;init addr of data for FFT
          RET

```

```

*****
*                               FFT                                *
*****

```

```

FFT       CLRC   OVM         ;reset overflow mode
          SETC   SXM         ;set sign extension mode
          SPM    1          ;shift product by 1
          LDP    #PAGE0      ;change data page pointer
          APL    #clr_ndx_mask,PMST ;clr ndx bit
          LDP    #PAGE2      ;restore data page pointer for .bss
          MAR    *,AR4
          LAR    AR4,STAGES   ;AR4 <- STAGLP counter (DO 20 L=1,M)
          MAR    *-,AR5
          STAGLP LACC   NO_BFLY,15 ;ACC <- NO_BFLY (N initially)
          SACH   DATSEP,1     ;store sep betw two input val (x2 I,Q)
          SACH   NO_BFLY     ;store actual no. of butterflies = N/2
          ZAC
          SACL   BFLY_CNT     ;store butterfly count for each stage
          LAR    AR5,NO_BFLY  ;AR5 <- BFLYLP counter (DO 20 J=1,LE1)
          MAR    *-,AR6      ;AR5 <- AR5-1, ARP <- 6
          LACC   #SINE
          SACL   SINADDR      ;init pointer to sin table

```

```

BFLYLP      LACC  BFLY_CNT,1
CALC_BFY    SACL  OFFSET      ;offset from base addr to input pt of interest
            LAR   ARO,OFFSET    ;ARO = INDX <- OFFSET for indirect addr
            LAR   AR6,DAT_LOC   ;AR6 <- points to loc of data
            MAR   *0+          ;point to AI
            LAR   ARO,DATSEP    ;ARO = INDX <- DATSEP for indirect addr
            LACC  *0+,15        ;ACC <- 1/2 AI, point to BI
            SUB   *,15          ;ACCH <- 1/2(AI-BI)
            SACH  TMP1         ;TMP1 <- 1/2(AI-BI)
            ADD   *0-,16        ;ACCH <- 1/2(AI+BI), point to AI
            SACH  *+           ;AI <- 1/2(AI+BI), point to AQ
            LACC  *0+,15        ;ACC <- 1/2 AQ, point to BQ
            SUB   *,15          ;ACCH <- 1/2(AQ-BQ)
            SACH  TMPQ         ;TMPQ <- 1/2(AQ-BQ)
            ADD   *0-,16        ;ACCH <- 1/2(AQ+BQ), point to AQ
            SACH  *0+          ;AQ <- 1/2(AQ+BQ), point to BQ
            LACC  SINADDR
            TBLR  SINVAL      ;read in sin value
            ADD   #SIN2COS
            TBLR  COSVAL      ;read in cos value
            LT    COSVAL      ;T <- COS
            MPY   TMPQ        ;P <- 1/2(AQ-BQ) * COS
            LTP   SINVAL      ;T <- SIN, ACC <- P
            MPY   TMP1        ;P <- 1/2(AI-BI) * SIN
            SPAC   *          ;ACC <- 1/2(AQ-BQ)*COS - 1/2(AI-BI)*SIN
            SACH  *          ;store BQ, point to BI
            MPY   TMPQ        ;P <- 1/2(AQ-BQ) * SIN
            LTP   COSVAL      ;T <- COS, ACC <- P
            MPY   TMP1        ;P <- 1/2(AI-BI) * COS
            APAC   *          ;ACC <- 1/2(AI-BI)*COS + 1/2(AQ-BQ)*SIN
            SACH  *          ;store BI
            LACC  OFFSET      ;compute new offset from base addr to
            ADD   DATSEP,1    ;input val of interest
            SACL  OFFSET
            SUB   FFTSIZ,1    ;ACC <- J + DATSEP*2 - FFTSIZ*2
            BCND  CALC_BFY,LT ;if ACC<0, cont CALC_BFY with same twid
            LACC  SINADDR     ;else change twid factor
            ADD   NXTTWD
            SACL  SINADDR
            LACC  BFLY_CNT
            ADD   #one        ;incr butterfly count
            SACL  BFLY_CNT
            MAR   *,AR5
            BANZ  BFLYLP,*-,AR6
            LACC  NXTTWD,1
            SACL  NXTTWD      ;NXTTWD <- 2*NXTTWD to prepare for following
                                ;stages where twid incr is every 2nd
            MAR   *,AR4
            BANZ  STAGLP,*-,AR5
            LDP   #PAGE0      ;change data page pointer
            OPL   #set_ndx_mask,PMST ;set ndx bit
            LDP   #PAGE2      ;restore data page pointer for .bss
            RET

```

```

*****
*                               BITREV                               *
*****

```

```

BITREV      LACC  FFTSIZ
            SFR
            SAMM  INDX        ;INDX <- FFTSIZ/2
            LAR   AR4,#zero    ;AR4 <- Normal order counter
            LAR   AR5,#zero    ;AR5 <- Bit-reversed order counter
            LAR   AR6,DAT_LOC   ;AR6 <- Addr of FFT bins, swap src loc
            LAR   AR7,SWPCTR    ;AR7 <- swap counter, with minus 1 already
            MAR   *,AR4
            SAR   AR5,BITRV_IDX ;store bit rev index
            LACC  BITRV_IDX
            SAMM  ARCR         ;ARCR <- AR5

```

```

NOP                                ;to resolve pipeline conflict with
NOP                                ; loading of ARCR
CMPR      1                        ;is AR4 < ARCR=AR5 ?
BCND      NO_SWP,NTC              ;if not, no swap necessary
CALL      SWAP
MAR        *,AR4
NO_SWP    MAR      *,AR5          ;incr AR4 (normal counter)
MAR        *BRO+,AR6             ;incr AR5 (bit-reverse counter)
RPT        #1
MAR        *+                    ;incr AR6 by 2 because of I,Q (swap src loc)
MAR        *,AR7                ;ARP <- 7
BANZ      REORDER,*-,AR4
RET

```

```

*****
*                                SWAP                                *
*****

```

```

SWAP      MAR        *,AR6
LACC      *+          ;ACC <- cont in loc pointed to by AR6 (I)
                        ;AR6 points to (Q)
SACL      XFERI       ;store in temp transfer area
LACC      *-,0,ARO    ;ACC <- cont in loc pointed to be AR6 (Q)
                        ;AR6 points back to (I), ARP <- 0
SACL      XFERQ       ;store in temp transfer area
LACC      DAT_LOC     ;locate swap partner
ADD       BITRV_IDX,1 ;ACC <- DAT_LOC + (2*BITRV_IDX)
SACL      SWPADDR     ;store swap dest addr
LAR       ARO,SWPADDR ;Swap data
LACC      *+,0,AR6    ;ACC <- cont in loc pointed to by ARO (I)
                        ;ARO points to (Q), ARP <- 6
SACL      *+,0,ARO    ;move I to src loc
                        ;AR6 points to (Q), ARP <- 0
LACC      *-,0,AR6    ;ACC <- cont in loc pointed to by ARO (Q)
                        ;ARO points back to (I), ARP <- 6
SACL      *-,0,ARO    ;move Q to src loc
                        ;AR6 points back to (I), ARP <- 0
LACC      XFERI
SACL      *+          ;store XFERI in swap dest loc (I)
LACC      XFERQ
SACL      *           ;store XFERQ in swap dest loc (Q)
RET

```

```

*****
*                                SAVBINS                            *
*****

```

```

SAVBINS   MAR        *,ARO
LAR       ARO,NXT_BIN ;ARO <- points to next bin location
LACC      #FFT_DATA
SAMM      BMAR        ;BMAR <- points to FFT output samples
RPT        #63
BLDD      FMAR,*+     ;move output to bins array
SAF       ARO,NXT_BIN ;no readjustment of pointer necessary
RET

```

```

*****
*                                FINDMAX                            *
*****

```

```

FINDMAX   LDP        #PAGE0      ;for access to BRCR
SPLK      #N_BINS_minus_1,BRCR
LDP       #PAGE2              ;reset for .bss variables
MAR       *,ARO               ;ARP <- 0
LAR       ARO,#FFT1_BINS      ;ARO <- pointing to first freq bin sample
ZAC       ;ACC <- init with min value of 0 (magnitude)
SACB      ;ACCB <- ACC
RPTB      ENDB-1
LACC      *+                  ;ACC <- freq bin, ARO points to next bin

```

```

                ABS                ;absolute value of ACC, reset carry bit
                CRGT               ;ACC > ACCR ? if yes, set carry bit
                SACL               ;store maximum value
                MAXVAL
                LACC               ;store max value in high accumulator
                MAXVAL,16
                MAR                ;15-bit norm, AR1 <- stores shift
                *,AR1
                LAR                ;init shift counter (0)
                AR1,#zero
                RPT                ;know at least a shift of 4 (12 bit no.)
                #14
                NORM              ;stops shifting when first significant mag
                *+                ;is found, performing NOP's for the remainder
                NOP               ;of the repeat loops
                NOP
                MAR                *-
                SAR                AR1,SHIFT_minus_1 ;store shift to be applied prior to
                RET              ;multiplying to form product seq

```

```

*****
*                                DO_SHIFT                                *
*****

```

```

DO_SHIFT      LAR      ARO,#FFT1_BINS ;ARO <- points to first freq bin sample
              LAR      AR3,#N_BINS_minus_1;AR3 <- bin counter
              MAR      *,ARO
BINLP         LACC     *
              RPT      SHIFT_minus_1
              SFL
              SACL     *+,0,AR3
              BANZ     BINLP,*-,ARO
              RET

```

```

*****
*                                SPECTR                                *
*****

```

```

SPECTR        LACC     #FFT1_LGTH,15
              SACH     N_PRIM,1
              SACH     NPRIM_BY_2
              SUB      #one,15
              SACH     CMPR_REG,1 ;store ARCR for later use (N'-1)
              LACC     FFT2_LGTH,14
              SACH     QUART_N
              SUB      #one,14
              SACH     SWPCTR      ;BITREV cntr -> first (N/4) bins (incl bin 0)
              ADD      #one,14    ;restore QUART_N
              ADD      FFT2_LGTH,15
              SACH     THREE_QUART_N
              LAR      AR1,#zero ;AR1 <- IDX_BIN1 counter
B1_LOOP       SAR      AR1,IDX_BIN1
              LACC     #FFT1_BINS ;calc addr of bin 1
              ADD      IDX_BIN1,1 ;ADDR_B1 = #FFT1_BINS + (2*IDX_BIN1)
              SACL     ADDR_B1    ;      = 1 of bin 1
              LAR      AR2,IDX_BIN1 ;initialize IDX_BIN2 counter
B2_LOOP       SAR      AR2,IDX_BIN2
              LACC     #FFT1_BINS ;calc addr of bin 2
              ADD      IDX_BIN2,1 ;ADDR_B2 = #FFT1_BINS + (2*IDX_BIN2)
              SACL     ADDR_B2    ;      = 1 of bin 2
              MAR      *,AR6
              LAR      AR6,XMEM
CHECK_B1      LACC     IDX_BIN1
              SUB      NPRIM_BY_2
              BCND     POS_B1,LT
NEG_B1        SUB      NPRIM_BY_2 ;convert to "neg" freq
              B        SAV_B1
POS_B1        ADD      NPRIM_BY_2 ;restore IDX_BIN1
              SAV_B1    *+        ;save in XMEM
              SACL     IDX_B1
CHECK_B2      LACC     IDX_BIN2
              SUB      NPRIM_BY_2
              BCND     POS_B2,LT
NEG_B2        SUB      NPRIM_BY_2 ;convert to "neg" freq
              B        SAV_B2

```



```
*****
*                                     PRODUCT                                     *
*****
```

```
PRODUCT      SPM      1
LAR          AR4,ADDR_B1      ;AR4 <- points to I1
LAR          AR5,ADDR_B2      ;AR5 <- points to I2
LAR          AR6,ADDR_SEQ2    ;AR6 <- points to next addr in SEQ2
MAR          *,AR5
LT           *,AR4            ;T <- I2, AR5 points to Q2, ARP <-4
MPY          **               ;P <- I1*I2, AR4 points to Q1
SPH          TMPH            ;TMPH = (I1*I2)
SPL          TMPH            ; store hi and lo to keep accuracy til +/-
MPY          *,AR5            ;P <- I2*Q1, ARP <- 5
SPH          TMPQ            ;TMPQ = (I2*Q1)
SPL          TMPQ            ; store hi and lo to keep accuracy til +/-
LT           *,AR4            ;T <- Q2, ARP <-4
LACC         TMPH,16          ;ACC <- (I1*I2)
ADDS         TMPH            ;suppress sign extension mode
MPY          *,AR6            ;P <- Q1*Q2, AR4 points to I1, ARP <- 6
APAC         **               ;ACC <- (I1*I2 + Q1*Q2)
SACH         *,0,AR4          ;store new I in SEQ2, next loc in SEQ2
LACC         TMPQ,16          ;ACC <- 1/2 .I2*Q1)
ADDS         TMPQ
MPY          *,AR6            ;P <- Q2*I1
SPAC         **               ;ACC <- 1/2 (I2*Q1 - I1*Q2)
SACH         **               ;store new Q in SEQ2 array, point back to new I
                                ;to prepare for phase correction
SAR          AR6,ADDR_SEQ2    ;TEMPORARY FOR CHECK
RET
```

```
*****
*                                     CORR_PHAS                               *
*****
```

```
CORR_PHAS    SPM      0
LACC         #SP_TWD32
SACL         NXTTWD          ;initialize NXTTWD
LT           HOP_NUMBER      ;let hop number = m
MPY          L               ;P <- m*L
SPL          M_TIMES_L
LT           ALPHA
MPY          M_TIMES_L       ;P <- alpha*m*L
PAC          **               ;ACC <- P
RPT          #15
SUBC         N_PRIM          ;alpha*m*L/N'
SACH         PHASEINCR       ;remainder is the phase increment
LT           PHASEINCR       ;adjust because reading from table of 256
MPY          NXTTWD
SPL          PHASEINCR
LACC         #SINE           ;locate phase correction factor in table
ADD          PHASEINCR
TBLR        SINVAL
ADD          #SIN2COS        ;from FFT1_INIT
TBLR        COSVAL
SPM          1
MAR          *,AR6           ;AR6 already pointing to I to be phase corrected
LAR          AR6,ADDR_SEQ2   ;TEMPORARY FOR CHECK
LACC         **             ;save current value of I in tmp variable
SACL         TMPH            ;point to Q
LT           SINVAL
MPY          **               ;P <- Q * sin, point to I
LTP          COSVAL          ;ACC <- P=Q*sin, T <- cosval
MPY          *               ;P <- I * cos
APAC         **               ;ACC <- I*cos + Q*sin
SACH         **             ;store new I, point to Q
MPY          *               ;P <- Q * cos
LTP          SINVAL          ;ACC <- P=Q*cos, T <- sinval
MPY          **             ;P <- I * sin
```

```

SPAC                                ;ACC <- Q*cos - I*sin
SACH                                ;store new Q, AR6 pointing to next loc
SAR                                ;update ADDR_SEQ2
RET

*****
*                                FFT2_INIT                                *
*****

FFT2_INIT    LACC    #SP_TWD128
SACL    NXTTWO    ;store as next twid factor incr
LACC    FFT2_LGTH
SACL    FFTSIZ    ;init size of FFT = P
SACL    NO_BFLY    ;init no. of butterflies = P
LACC    M_FFT2    ;ACC <- no. of stages, FFT lgth-P
SACL    STAGES
LACC    #SEQ2
SACL    DAT_LOC    ;init addr of data for FFT
RET

*****
*                                BREV_2                                *
*****

BREV_2    LACC    FFTSIZ    ;may not have to do this
SFR
SAMM    INDX
LACC    DAT_LOC
ADD    THREE_QUART_N,1 ;shift of 1 for I,Q
SACL    BOTTOM_ADDR
MAR    *,AR7
LAR    AR4,THREE_QUART_N ;AR4 <- Norm counter
LAR    AR5,#three    ;AR5 <- Bit rev counter
LAR    AR6,BOTTOM_ADDR ;starting addr of bottom half
LAR    AR7,QUART_N    ;AR7 <- counter
MAR    *-,AR3
ARRANGE    SAR    AR4,NORM_IDX
SAR    AR5,BITRV_IDX
LACC    BITRV_IDX
SAMM    ARCR    ;ARCR <- bit rev counter
LAR    AR3,QUART_N
NOP    ;to resolve pipeline conflict
CMPR    2    ;is N/4 > bit rev counter
BCND    GO_ON,TC    ;if yes, already switched
LAR    AR3,THREE_QUART_N
CMPR    2    ;is (3/4)*N > bit rev counter
BCND    REPLACE,TC    ;if yes, just overwrite with new value
LAR    AR3,NORM_IDX
CMPR    1    ;is Norm counter < Bit rev counter
BCND    GO_ON,NTC    ;if not, no swap or already swapped
CALL    SWAP
B    GO_ON
REPLACE    MAR    *,ARO
LACC    DAT_LOC
ADD    BITRV_IDX,1
SACL    SWPADDR
LAR    ARO,SWPADDR
LACC    *+,0,AR6    ;transfer I
SACL    *+,0,ARO
LACC    *-,0,AR6    ;transfer Q
SACL    *-,
GO_ON    MAR    *,AR4
MAR    *+,AR5    ;update norm counter
MAR    *BRO+,AR6    ;update bit rev counter
RPT    #1
MAR    *+    ;update addr
MAR    *,AR7
BANZ    ARRANGE,*-,AR3
RET

```



```
*****
*                               SAVEST                               *
*****
```

```
SAVEST      LACC    #SEQ2      ;determine addr of I pointer
BOTTOM_HALF ADD     THREE_QUART_N,1 ;Addr = #SEQ2 + 2*(3/4)lgth_seq2
            SACL    IQ_PTR
            LAR     AR4,IQ_PTR    ;AR4 <- IQ_PTR
            LAR     AR5,QUART_N   ;AR5 <- counter for number of estimates
            LAR     AR6,NXT_EST   ;AR6 <- addr of next avail SPEC_EST loc
            MAR     *,AR5
            MAR     *-           ;initialize MAGN_B counter
MAGN_B      CALL    MAGLP
            MAR     *,AR5
            BANZ    MAGN_B
TOP_HALF    LACC    #SEQ2
            SACL    IQ_PTR
            LAR     AR4,IQ_PTR    ;AR4 <- IQ_PTR
            LAR     AR5,QUART_N   ;AR5 <- MAGN_T counter
            MAR     *,AR5
            MAR     *-
MAGN_T      CALL    MAGLP
            MAR     *,AR5
            BANZ    MAGN_T
            SAR     AR6,NXT_EST
            RET
```

```
*****
*                               MAGLP                               *
*****
```

```
MAGLP      MAR     *,AR4
            LACC    *+           ;ACC <- I, point to Q
            ABS     ;take absolute value
            SACL    MAG_I
            LACC    *+,0,AR6     ;ACC <- Q, point to next I, ARP <- 6
            ABS     ;take absolute value
            SACL    MAG_Q
            SUB     ;ACC <- MAG_Q - MAG_I
            BCND    QG1,GT
IGQ         LACC    MAG_I         ;I > Q, check if I > 4Q
            SUB     MAG_Q,2
            BCND    IG4Q,GT
IL4Q        LACC    MAG_I,16      ;I < 4Q
            SUB     MAG_I,13      ;MAG = I - (1/8)I + (1/2)Q
            ADD     MAG_Q,15
            B       STORMAG
IG4Q        LACC    MAG_I,16      ;I > 4Q
            ADD     MAG_Q,13      ;MAG = I + (1/8)Q
            B       STORMAG
QG1         LACC    MAG_Q         ;Q > I, check if Q > 4I
            SUB     MAG_I,2
            BCND    QG4I,GT
QL4I        LACC    MAG_Q,16      ;Q < 4I
            SUB     MAG_Q,13      ;MAG = Q - (1/8)Q + (1/2)I
            ADD     MAG_I,15
            B       STORMAG
QG4I        LACC    MAG_Q,16      ;Q > 4I
            ADD     MAG_I,13      ;MAG = Q + (1/8)I
STORMAG     SACL    *+           ;store magn in spec est array
            ;point to next spec est loc
            RET
```

```
.data
```

```
*****
*                               Sine/Cosine Coefficients Table: length 5N/4                               *
*                               instead of 3N/4 because using the same table for                               *
*                               phase correction routine.                               *
*****
```

```

*           Before, 3N/4 was all that was needed because the           *
*           second half of the sine table is equal to the first        *
*           half of the cosine table.                                  *
*           Here, N is the length of the longest FFT.                  *
*           Currently, N=256                                           *
*****

```

```

SINE      .word 0h                ;0
          .word 0324h
          .word 0647h
          .word 096Ah
          .word 0C8Bh
          .word 0FABh
          .word 012C8h
          .word 015E2h
          .word 018F8h
          .word 01C0Bh
          .word 01F19h            ;10
          .word 02223h
          .word 02528h
          .word 02826h
          .word 02B1Fh
          .word 02E11h
          .word 030FBh
          .word 033DEh
          .word 036BAh
          .word 0398Ch
          .word 03C56h            ;20
          .word 03F17h
          .word 041CEh
          .word 0447Ah
          .word 0471Ch
          .word 049B4h
          .word 04C3Fh
          .word 04EBFh
          .word 05133h
          .word 0539Bh
          .word 055F5h            ;30
          .word 05842h
          .word 05A82h
          .word 05CB4h
          .word 05ED7h
          .word 060ECh
          .word 062F2h
          .word 064E8h
          .word 066CFh
          .word 068A6h
          .word 06A6Dh            ;40
          .word 06C24h
          .word 06DCAh
          .word 06F5Fh
          .word 070E2h
          .word 07255h
          .word 073B5h
          .word 07504h
          .word 07641h
          .word 0776Ch
          .word 07884h            ;50
          .word 0798Ah
          .word 07A7Dh
          .word 07B5Dh
          .word 07C29h
          .word 07CE3h
          .word 07D8Ah
          .word 07E1Dh
          .word 07E9Dh
          .word 07F09h
          .word 07F62h            ;60
          .word 07FA7h

```

	.word	07FD8h	
	.word	07FF6h	
COSINE	.word	07FFFh	
	.word	07FF6h	
	.word	07FD8h	
	.word	07FA7h	
	.word	07F62h	
	.word	07F09h	
	.word	07E90h	;70
	.word	07E10h	
	.word	07D8Ah	
	.word	07CE3h	
	.word	07C29h	
	.word	07B5Dh	
	.word	07A7Dh	
	.word	0798Ah	
	.word	07884h	
	.word	0776Ch	
	.word	07641h	;80
	.word	07504h	
	.word	073B5h	
	.word	07255h	
	.word	070E2h	
	.word	06F5Fh	
	.word	06DCAh	
	.word	06C24h	
	.word	06A6Dh	
	.word	068A6h	
	.word	066CFh	;90
	.word	064E8h	
	.word	062F2h	
	.word	060ECh	
	.word	05ED7h	
	.word	05C84h	
	.word	05A82h	
	.word	05842h	
	.word	055F5h	
	.word	05398h	
	.word	05133h	;100
	.word	04EBFh	
	.word	04C3Fh	
	.word	049B4h	
	.word	0471Ch	
	.word	0447Ah	
	.word	041CEh	
	.word	03F17h	
	.word	03C56h	
	.word	0398Ch	
	.word	0368Ah	;110
	.word	033DEh	
	.word	030FBh	
	.word	02E11h	
	.word	02B1Fh	
	.word	02826h	
	.word	02528h	
	.word	02223h	
	.word	01F19h	
	.word	01C03h	
	.word	01811h	;120
	.word	015E2h	
	.word	012C8h	
	.word	0FABh	
	.word	0C88h	
	.word	096Ah	
	.word	0647h	
	.word	0324h	
	.word	0h	
	.word	0FCDCh	
	.word	0F9B9h	;130

```

.word 0F696h
.word 0F375h
.word 0F055h
.word 0ED38h
.word 0EA1Eh
.word 0E708h
.word 0E3F5h
.word 0E0E7h
.word 0DDDDh
.word 0DAD8h      ;140
.word 0D7DAh
.word 0D4E1h
.word 0D1EFh
.word 0CF05h
.word 0CC22h
.word 0C946h
.word 0C674h
.word 0C3AAh
.word 0C0E9h
.word 0BE32h      ;150
.word 0BB86h
.word 0BBE4h
.word 0B64Ch
.word 0B3C1h
.word 0B141h
.word 0AECDh
.word 0AC65h
.word 0AA0Bh
.word 0A7BEh
.word 0A57Eh      ;160
.word 0A34Ch
.word 0A129h
.word 09F14h
.word 09D0Eh
.word 09B18h
.word 09931h
.word 0975Ah
.word 09593h
.word 093DCh
.word 09236h      ;170
.word 090A1h
.word 08F1Eh
.word 08DABh
.word 08C4Bh
.word 08AFCh
.word 089BFh
.word 08894h
.word 0877Ch
.word 08676h
.word 08583h      ;180
.word 084A3h
.word 083D7h
.word 0831Dh
.word 08276h
.word 081E3h
.word 08163h
.word 080F7h
.word 0809Eh
.word 08059h
.word 08028h      ;190
.word 0800Ah
.word 08000h
.word 0800Ah
.word 08028h
.word 08059h
.word 0809Eh
.word 080F7h
.word 08163h
.word 081E3h

```

```

.word 08276h          ;200
.word 08310h
.word 083D7h
.word 084A3h
.word 08583h
.word 08676h
.word 0877Ch
.word 08894h
.word 0898Fh
.word 08AFCh
.word 08C48h          ;210
.word 08DA8h
.word 08F1Eh
.word 090A1h
.word 09236h
.word 093DCh
.word 09593h
.word 0975Ah
.word 09931h
.word 09B18h
.word 09D0Eh          ;220
.word 09F14h
.word 0A129h
.word 0A34Ch
.word 0A57Eh
.word 0A7BEh
.word 0AA08h
.word 0AC65h
.word 0AEC Dh
.word 0B141h
.word 0B3C1h          ;230
.word 0B64Ch
.word 0B8E4h
.word 0BB86h
.word 0BE32h
.word 0C0E9h
.word 0C3AAh
.word 0C674h
.word 0C946h
.word 0CC22h
.word 0CF05h          ;240
.word 0D1EFh
.word 0D4E1h
.word 0D7DAh
.word 0DAD8h
.word 0DDDDh
.word 0E0E7h
.word 0E3F5h
.word 0E708h
.word 0EA1Eh
.word 0ED38h          ;250
.word 0F055h
.word 0F375h
.word 0F696h
.word 0F9B9h
.word 0FCDCh
.word 0h
.word 0324h
.word 0647h
.word 096Ah
.word 0C88h          ;260
.word 0FABh
.word 012C8h
.word 015E2h
.word 018F8h
.word 01C08h
.word 01F19h
.word 02223h
.word 02528h

```

```

.word 02826h
.word 0281Fh      ;270
.word 02E11h
.word 030FBh
.word 033DEh
.word 0368Ah
.word 0398Ch
.word 03C56h
.word 03F17h
.word 041CEh
.word 0447Ah
.word 0471Ch      ;280
.word 049B4h
.word 04C3Fh
.word 04EBFh
.word 05133h
.word 05398h
.word 055F5h
.word 05842h
.word 05A82h
.word 05CB4h
.word 05ED7h      ;290
.word 060ECh
.word 062F2h
.word 064E8h
.word 066CFh
.word 068A6h
.word 06A6Dh
.word 06C24h
.word 06DCAh
.word 06F5Fh
.word 070E2h      ;300
.word 07255h
.word 07385h
.word 07504h
.word 07641h
.word 0776Ch
.word 07884h
.word 0798Ah
.word 07A7Dh
.word 07B5Dh
.word 07C29h      ;310
.word 07CE3h
.word 07D8Ah
.word 07E1Dh
.word 07E9Dh
.word 07F09h
.word 07F62h
.word 07FA7h
.word 07FD8h
.word 07FF6h

```

```

*****
*          Hamming Coefficients Table (32 entries)          *
*****

```

```

HAM_32      .word 0A86h
             .word 0CC6h
             .word 01131h
             .word 0179Ah
             .word 01FC4h
             .word 0295Dh
             .word 03407h
             .word 03F59h
             .word 04AE4h
             .word 05636h
             .word 060E0h
             .word 06A79h
             .word 072A2h

```

```
.word 0790ch
.word 07D76h
.word 07FB7h
.word 07FB7h
.word 07D76h
.word 0790ch
.word 072A2h
.word 06A79h
.word 060E0h
.word 05636h
.word 04AE4h
.word 03F59h
.word 03407h
.word 0295Dh
.word 01FC4h
.word 0179Ah
.word 01131h
.word 0CC6h
.word 0A86h

.end
```



```

*****
*****
*
*                               DATAMV
*
*   This subroutine moves a block of data into an N' array (working
*   area) for the short time FFT calculations. Data is moved
*   sequentially so that it is in complex pairs I,Q,I,Q,...
*
*****

*****
*
*                               TAPERA
*
*   This subroutine data tapers N' samples with a Hamming window
*   before the FFT is performed. The coefficients for the Hamming
*   window are stored in a table in program memory.
*
*****

*****
*
*                               FFT1_INIT
*
*   This subroutine initialises parameters for an N' point FFT.
*   The parameters initialised are as follows:
*
*       NXTTWD - distance between twiddle factors in
*               sin/cos table
*       FFTSIZ - length of the FFT (N' in this case)
*       NO_BFLY - number of butterflies in the first stage
*       STAGES - number of stages for FFT
*       DAT_LOC - addr of the data sequence to be FFT'd
*
*****

*****
*
*                               FFT
*
*   This subroutine performs the FFT of a sequence using a Radix-2,
*   decimation-in-frequency, and in-place algorithm. The algorithm
*   is based on the following algorithm:
*
*   SUBROUTINE DIFFFT(X,NU)
*
*       COMPLEX X(1024),U,W,T
*       N=2**NU
*       PI=3.14159265358979
*
*       DO 20 L=1,NU
*       LE=2**(NU+1-L)
*       LE1=LE/2
*       U=(1.0,0.0)
*       W=CMPLX(COS(PI/FLOAT(LE1)), -SIN(PI/FLOAT(LE1)))
*       DO 20 J=1,LE1
*       DO 10 I=J,N,LE
*       IP=I+LE1
*       T=X(I)+X(P)
*       X(IP)=(X(I)-X(P))*U
*   10   X(I)=T
*   20   U=U*W
*
*       Ref: Oppenheim, A.V., and Schaffer, R.W., Discrete-time Signal
*       Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.
*
*   At the end of this subroutine, data (frequency bins) are paired
*   (I,Q) in bit-reversed order.
*
*****

*****
*
*                               BITREV
*

```

```

* This subroutine unscrambles the output bins of the FFT subroutine.
* This routine eliminates the need to reserve two extra data memory
* arrays for the bit-reversed data (one for the short time FFT, and
* one for the second sequence FFT). Data comes in I,Q,I,Q,... in
* bit reversed mode and is transformed into an I,Q,I,Q,... sequence
* in normal order (i.e. I0,Q0,I1,Q1,...I(n-1),Q(n-1)).

```

Basic procedure:

```

    Initialize two counters, one which counts in normal order,
    one in bit reversed order. At each increment, if the
    norm ctr < bit rev ctr, then bins need to be swapped.
    The normal count and bit rev count represent the "location"
    from the base addr of the array of the desired bins.

```

(e.g. N=16)

Bin	Norm Count	Bitrev count	SWAP?
0	0	0	N
8	1	8	Y
4	2	4	Y
12	3	12	Y
2	4	2	N (already swapped)
10	5	10	Y
6	6	6	N
14	7	14	Y
1	8	1	N (already swapped)
9	9	9	N
5	10	5	N (already swapped)
13	11	13	Y
3	12	3	N (already swapped)
11	13	11	N (already swapped)
7	14	7	N (already swapped)
15	15	15	N

***** SWAP

```

* This subroutine swaps two bins (I,Q) around.

```

***** SAVBINS

```

* This subroutine stores the FFT output (frequency) bins from the
* short time FFT into an array. Note that FFT output bins for all
* the short time FFTs are stored in order to be able to normalize
* frequency bin values and to form all possible combinations of the
* second sequence (explained in SPECTR).

```

***** FINDMAX

```

* This subroutine searches through the frequency bins array to
* find the maximum (in magnitude). Using the NORM instruction,
* the exponent of the maximum magnitude is found. The corresponding
* shift required to normalize this number to 15 bits is calculated.

```

Ref: Example 7-7, TMS320C5x User's Guide, p. 7-10
 Example 3, TMS320C5x User's Guide, p. 4-159

***** DO_SHIFT

```

* This subroutine applies the shift calculated in FINDMAX to all
* frequency bins. With all the frequency bins normalized,

```

* the whole P register (32 bits) need not be stored when forming *
* the second sequence (The second sequence is formed by complex *
* multiplying two frequency bins together for the series of short *
* time FFT output bins). This allows the samples of the second *
* sequence to be kept as 16 bit numbers (including a sign bit). *

SPECTR

This subroutine calculates estimates of the cyclic spectrum for the channel pair regions in the bifrequency plane. The channel pair regions represent all possible combinations of frequency bins multiplied together from each short time FFT to form the second sequence. An FFT is performed on the second sequence. The resulting FFT output bins represent the cyclic spectral estimates for a particular channel pair region. The estimates of interest are stored in an array for subsequent transfer to the PC.

Indexing for the channel pair regions is as follows:

Case: N' bins, $N' = 4$

where $(.,.) = (\text{first bin}, \text{second bin})$

Basic Procedure:

```

for INDX_BIN1 := 0 to (N'-1)
  calculate addr of bin 1
  for INDX_BIN2 := INDX_BIN1 to (N'-1)
    calculate addr of bin2
    /for SLICE := 0 to (P-1)      (*P is lgth of seq2)
      /  new I := I1I2 + Q1Q2
      <  store new I
      \  new Q := -I1Q2 + I2Q1
      \  store new Q
      \  phase correction for new I and Q
    end (for SLICE)
    perform P-point FFT
    store P/2 output values of interest
  end (for INDX_BIN2)
end (for INDX_BIN1)

```

Other calculations performed in this subroutine include converting the indices (0,...,N'-1) into ((-N'/2+1)...,0,...(N'/2-1)). These converted indices were stored in an array to be transferred to the PC to simplify mapping of data into an ASCII file for printing.

Note that due to the indexing scheme used, the subroutine also verifies that the correct bin is complex conjugated before being multiplied.

PRODUCT

This subroutine multiplies two complex frequency bins and stores the complex result in P sequence array (work area for 2nd FFT). Recall from subroutine FINDMAX and DO SHFT. frequency bins have

```

*      normalized in order that the high P register, alone, could be      *
*      stored as the result, therefore limiting samples to 16 bits for the *
*      second FFT calculation. (Truncation)                                *
*                                                                           *
*      (BIN 1) x (BIN 2)^          (^ is complex conjugate)              *
*      new I = (I1I2 + Q1Q2)                                              *
*      new Q = (-I1Q2 + I2Q1)                                             *
*****
*                                                                           *
*      CORR_PHAS                                                           *
*                                                                           *
*      This subroutine applies a phase correction to the complex product.  *
*      This is done in order to maintain coherence from sample to sample  *
*      in the second sequence.                                           *
*                                                                           *
*      The phase correction term is calculated as follows:                *
*                                                                           *
*      phase correction = exp[-2*pi*alpha*mL/N']                          *
*                                                                           *
*      where,                                                             *
*      alpha   - distance separating the two frequency bins being        *
*                multiplied                                              *
*      m       - sample number of second sequence (hop number)          *
*      L       - overlap parameter                                       *
*      N'      - length of short time FFT                                *
*                                                                           *
*      The sine and cosine values for the phase correction are obtained    *
*      from the sin/cos table in program memory.                         *
*                                                                           *
*      Therefore,                                                         *
*                                                                           *
*      (I + jQ)*exp[-x]          , x = 2*pi*alpha*mL/N'                 *
*      new I = Icos(x) + Qsin(x)                                         *
*      new Q = -Isin(x) + Qcos(x)                                         *
*****
*                                                                           *
*      FFT2_INIT                                                           *
*                                                                           *
*      This subroutine initialises parameters for a P point FFT.         *
*      The parameters initialised are as follows:                         *
*                                                                           *
*      NXTTWD - distance between twiddle factors in                      *
*                sin/cos table                                           *
*      FFTSIZ - length of the FFT (P in this case)                       *
*      NO_BFLY - number of butterflies in the first stage               *
*      STAGES - number of stages for FFT                                  *
*      DAT_LOC - addr of the data sequence to be FFT'd                  *
*****
*                                                                           *
*      BREV_2                                                               *
*                                                                           *
*      This subroutine bit reverses the last P/4 complex bins (I,Q) of   *
*      the output of an P-point FFT. The key thing is the initialisation *
*      of the normal and bit reversed counters with (3/4)P and 3         *
*      respectively. Note that the bit reversed pattern of 3 will always *
*      be (3/4)N.                                                         *
*                                                                           *
*      Basic procedure:                                                    *
*                                                                           *
*      The top P/4 bins will already have been reordered. Therefore,     *
*      some of the bottom P/4 bins may already be correct. If the bit   *
*      rev counter is < N/4, bins have already been swapped. If the     *
*      bit rev counter is > N/4 and < (3/4)N, all that is required is   *
*      to replace the bin the normal counter is pointing to, with the    *
*      bin the bit rev counter is pointing to because the data in the

```

```

*      former is no longer needed. Otherwise, the normal and bit rev      *
*      counters are compared. The corresponding bins are swapped only      *
*      if the normal counter is smaller than the bit rev counter.          *
*****
*****
*
*      SAVEST
*
*      This subroutine computes the magnitude of the spectral estimates
*      and stores them in a spectral estimate array that will be used to
*      plot the bifrequency plane. The values are stored such that they
*      can be read into PC memory in order, i.e. estimates for bottom half
*      of bins calculated first, then top half. Channel pair region
*      defined bottom up by freq bins as follows:
*
*      (P/2+1),..., (P-1), 0, 1, ..., (P/2 - 1)
*****
*****
*
*      MAGLP
*
*      The subroutine uses a two-line approximation for calculating the
*      magnitude derived in the reference listed below.
*
*      Ref: Adams, W.T., and Brady, J., "Magnitude Approximations
*      for Microprocessor Implementation", IEEE Micro, October 1973
*      pp. 27-31.
*
*      This subroutine calculates the magnitude according to the following:
*
*
*      I < Q          I > Q
*      ----          ----
*      4I < Q        4I > Q          I > 4Q          I < 4Q
*      ----          ----          ----          ----
*
*      * MAG =  Q+(1/8)I      Q-(1/8)Q+(1/2)I      I+(1/8)Q      I-(1/8)I+(1/2)Q
*****

```

```

/*****
/*      Program Name:  BIFRQ.C
/*      Author:      C. Tom
/*      Date:        06 Jul 1994
*****/

#include <stdio.h>
#include "tic5x.h"          /*TMS Development Library Header File*/

#define BASEIO      0x290    /*I/O Base Addr - default*/
#define BASEDPMEM   0xd000   /*Dual Port Memory Addr - default*/
#define BASEADDR_EST 0x8000  /*Base addr of spectral estimates in TMS memory*/

#define freq_bins   61       /*(N-1)+(N-2) = 2N-3*/
                             /*where N = length of 1st FFT*/
#define alpha_bins  32       /*P/4, P = length of 2nd FFT*/
#define alpha_range 992      /*alpha_bins * (N'-1)*/
#define N           32       /*N = length of 1st FFT*/
#define M           64       /*M = # Hamm coef per BIN for correction*/
                             /* = P/2*/

double real_zero;
double pi=3.141592654;
double X_coord[freq_bins],Y_coord[alpha_range];
double freq,smallest_f,alpha_incr;
double m, k, k_incr;
double r_d1,i_d1,r_d2,i_d2,r_d3,i_d3;
double r_w, i_w, mag_w, norm_w;
double estimate,correction;
double Hamm_f[33];          /*#elements = (fft2_lgth/4 + 1)*/

int zero=0;
int a,b,c,i,j;
int err,state;
int fft1_lgth,fft2_lgth;
int fft2stages,end_data;
int no_cp_regions,est_per_region;
int freq_2D;
int idx_alpha[1488];        /*#channel pair reg * 3*/
int tms_addr;
int values[32];             /*P/4 estimates*/
int Bi_f_slice[freq_bins][alpha_bins];

FILE *fp,*rp,*np;          /*file pointer*/

void main(void);
void Reset_DP(void);
void Wait_for_c5x(void);
void Retrieve_vars(void);
void Corr_win_atten(void);
void Read_indices(void);
void Open_file(void);
void Transfer_estimates(void);
void Init_2D_Array(void);
void Get_Data(int z);
void Forward_Inv_Hamm(int row);
void Backward_Inv_Hamm(int row);
void Write_to_file(int slice);
void Message(void);

/*****      CODE      *****/

void main(void)
/*****/
{
    fft1_lgth = 0;
    fft2_lgth = 0;

    /**LOAD AND RUN .ASM Program**/

```

```

Select_Board(BASEIO, BASEDPMEM);      /*Initialise c5x*/
Assert_Reset();                       /*Reset DSP Board*/
Assert_Hold();                        /*Put DSP processor into Hold mode*/
Reset_DP();
err = Load_Object_File("SPECTRM.OUT"); /*Load c5x program*/
switch (err) {
case 1:
    printf("\nFile SPECTRM.OUT not found\n");
    exit(1);
    break;
case 2:
case 3:
case 7: /*Other fatal error codes*/
    printf("\nError loading SPECTRM.OUT on DSP memory\n");
    exit(1);
    break;
} /*end switch (err)*/

    /**c5x PROGRAM LOADED SUCCESSFULLY**/

printf("SPECTRM.OUT has been loaded onto c5x.\n");
Release_Hold();
Release_Reset();                     /*start executing prog from loc P:0*/
printf("Executing .ASM program.\n");
Wait_for_c5x();                      /*Wait for c5x finish signal flag*/
printf("c5x program completed. Ready to read data.\n");
Retrieve_vars();
no_cp_regions = fft1_lgth*(fft1_lgth-1)/2;
est_per_region = (fft2_lgth/2);

    /**FILL IN "X" COORDINATES (i.e. "first" row of array in ASCII file)**/

smallest_f = (-1.0)*((double)(fft1_lgth/2)-1.0);
for (i=0; i< freq_bins; i++){
    X_coord[i] = smallest_f + (0.5*(double)(i));
} /*end for i*/

    /**FILL IN "Y" COORDINATES (i.e. "first" column of array in ASCII file)**/

alpha_incr = (4.0/(double)(fft2_lgth));      /*actually (fft1_lgth/L)/fft2_lgth*/
                                              /*normalized so that f(s)/N' = 1.0*/
for (i=0; i< alpha_range; i++){
    Y_coord[i] = (double)(i)*alpha_incr;
} /*end for i*/

    /**CREATE TABLE FOR CORRECTING ATTENUATION OF SPECTRAL ESTIMATES**/
    /**DUE TO HAMMING WINDOW IN f DOMAIN**/

Corr_win_atten();

    /**READ ARRAY FROM TMS WITH CHANNEL PAIR REGION INDICES AND ALPHAS**/

Read_indices();

    /**OPEN ASCII FILE TO STORE SPECTRAL ESTIMATES**/

Open_file();

    /**DETERMINE MAPPING OF ESTIMATES INTO 2D ARRAY THAT REPRESENTS**/
    /**ONE "SLICE" OF BIFREQUENCY PLANE**/

Transfer_estimates();
fclose(fp);
Message();
} /*end of MAIN*/

/***** SUBROUTINES *****/

void Reset_DP()

```

```

/*****/
{
    Put_Int('D', 0x3000, 0);          /*reset SIGNAL to PC*/
    Put_Int('D', 0x3001, 0);          /*reset loc holding FFT1_LGTH*/
    Put_Int('D', 0x3002, 0);          /*reset loc holding FFT2_LGTH*/
    Put_Int('D', 0x3003, 0);
    Put_Int('D', 0x3004, 0);
    Put_Int('D', 0x3005, 0);
}

void Wait_for_c5x()
/*****/
{
    int    flag=0;
    while (flag != 0x1234) {
        Get_Block_DP_RAM(0, 1, &flag);
    } /* end while (flag)*/
}

void Retrieve_vars()
/*****/
{
    Get_Block_DP_RAM(1, 1, &fft1_lgth);
    Get_Block_DP_RAM(2, 1, &fft2_lgth);
    Get_Block_DP_RAM(3, 1, &fft2stages);
    Get_Block_DP_RAM(4, 1, &end_data);
}

void Corr_win_atten()
/*****/

/*****/
/*Have to determine attenuation caused by Hamming Window according to*/
/*the following function:*/
/*
/*      W(theta)=0.54*D(theta) + 0.5*(0.46)*[D(theta-(2*pi/N'))
/*                                     + D(theta+(2*pi/N'))]
/*
/*      where N'= fft1_lgth
/*      D(theta) = exp[j(theta/2)]*sin[N'*(theta/2)]
/*               -----
/*               sin[theta/2]
/*
/*      D(.) = Dirichlet function
/*      let D1 = D(theta)
/*      D2 = D(theta-(2*pi/N'))
/*      D3 = D(theta+(2*pi/N'))
/*      theta = 2*pi*i/(N*M)      i=0,1,...,M
/*      M = no of coeff for one BIN
/*
/*****/
{
    k_incr = 2*pi/(double)(N*M);
    Hamming_f[0] = 1.0;
    if ((rp = fopen("Hamming", "w")) == NULL) {
        printf("Cannot open file named hamming.\n");
        exit(1);
    } /*end if rp*/
    for (i=1; i< ((M/2)+1); i++) {
        k = k_incr*i;
        r_d1 = cos(k/2)*sin((double)N*k/2)/sin(k/2);
        i_d1 = sin(k/2)*sin((double)N*k/2)/sin(k/2);
        m = k - (2*pi/(double)N);
        r_d2 = cos(m/2)*sin((double)N*m/2)/sin(m/2);
        i_d2 = sin(m/2)*sin((double)N*m/2)/sin(m/2);
        m = k + (2*pi/(double)N);
        r_d3 = cos(m/2)*sin((double)N*m/2)/sin(m/2);
        i_d3 = sin(m/2)*sin((double)N*m/2)/sin(m/2);
        r_w = 0.54*(r_d1) + 0.23*(r_d2 + r_d3);
        i_w = 0.54*(i_d1) + 0.23*(i_d2 + i_d3);
        mag_w = sqrt((r_w*r_w)+(i_w*i_w));
    }
}

```



```

        norm_w = mag_w/(0.54*(double)N);
        Hamm_f[i] = norm_w*norm_w;
    } /*end for i*/
    for (i=0; i<33; i++){
        fprintf(rp,"%7.6f\t\t;%i\n",Hamm_f[i],i);
    } /*end for i*/
    fclose(rp);
}

void Read_indices()
/******/
{
    Assert_Hold();
    state = Get_Block_16('D', 0x7800, 1488, &idx_alpha[0]);
    switch (state) {
        case 0:
            printf("\nData transfer from DSP to PC failed.\n");
            exit(1);
            break;
        case 1:
            printf("\nData block transfer from 0x7800 to PC successful.");
    } /*end switch (state)*/
    Release_Hold();
    if ((np = fopen("INDICES","w")) == NULL){
        printf("Cannot open file named INDICES.\n");
        exit(1);
    } /*end if np*/
    for (i=0; i<no_cp_regions; i++){
        fprintf(np,"%i\t%i\t%i\n",idx_alpha[(3*i)],idx_alpha[(3*i)+1],idx_alpha[(3*i)+2]);
    } /*end for i*/
    fclose(np);
}

void Open_file()
/******/
{
    if ((fp = fopen("fa_spec1","w")) == NULL){
        printf("\nCannot open file named fa_spec1.\n");
        exit(1);
    } /*end if fp*/
    fprintf(fp,"%i\t%i\n",alpha_range, freq_bins);
    real_zero = (double)zero;
    fprintf(fp,"%6.4f\t",real_zero);
    for (b=0; b<freq_bins; b++){
        fprintf(fp,"%3.1f\t",X_coord[b]);
    } /*end for b*/
    fprintf(fp,"\n");
    printf("\nHave opened ASCII file for spectral estimates.\n");
}

void Transfer_estimates()
/******/
{
    for (a=0; a<(fft1_lgth-1); a++){
        Init_2D_array(); /*Clear memory of 2D array*/

        /**FIND ALL CHANNEL PAIR REGIONS WITH ALPHA = a AND ALPHA = a+1**)*/

        for (i=0; i<no_cp_regions; i++){
            if (idx_alpha[(3*i)+2] == a)
            {
                /**STORE 2nd HALF OF ESTIMATES IN CP REGION**/

                tms_addr = BASEADDR_EST+(est_per_region*i)+(fft2_lgth/4);
                Get_Data(tms_addr);

                /**DETERMINE FREQ (i.e. row indx for 2D array**)*/

                freq = (double)(idx_alpha[(3*i)]+idx_alpha[(3*i)+1])/2.0;

```

```

        freq_2D = (int)((freq-smallest_f)*2.0);
        Forward_Inv_Hamm(freq_2D);
    }
    else if (idx_alpha[(3*i)+2] == (a+1))
    {
        /**STORE 1st HALF OF ESTIMATES IN CP REGION**/

        tms_addr = BASEADDR_EST+(est_per_region*i);
        Get_Data(tms_addr);

        /**DETERMINE FREQ (i.e. row indx for 2D array)**/

        freq = (double)(idx_alpha[(3*i)]+idx_alpha[(3*i)+1])/2.0;
        freq_2D = (int)((freq-smallest_f)*2.0);
        Backward_Inv_Hamm(freq_2D);
    }
    } /*end for i*/
    Write_to_file(a);
} /* end for a*/
}

void Init_2D_Array()
/******/
{
    for (i=0; i< freq_bins; i++){
        for (j=0; j< alpha_bins; j++){
            Bi_f_slice[i][j] = zero;
        } /*end for j*/
    } /*end for i*/
}

void Get_Data(int tms_loc)
/******/
{
    Assert_Hold();
    state = Get_Block_16('D', tms_loc, (fft2_lgth/4), &values[0]);
    switch (state) {
        case 0:
            printf("\nData transfer from DSP to PC failed.\n");
            exit(1);
            break;
        case 1:
            printf("\nTransferring data from %Xh.\n",tms_loc);
    } /*end switch (state)*/
    Release_Hold();
}

void Forward_Inv_Hamm(int row_idx)
/******/
{
    /**TRANSFER ESTIMATES TO 2D ARRAY**/

    for (j=0; j< alpha_bins; j++){
        correction = Hamm_f[j];
        estimate = (double)values[j]/correction;
        Bi_f_slice[row_idx][j] = (int)estimate;
    } /*end for j*/
}

void Backward_Inv_Hamm(int row_idx)
/******/
{
    /**TRANSFER ESTIMATES TO 2D ARRAY**/

    for (j=0; j< na_bins; j++){
        correction = Hamm_f[alpha_bins-j];
        estimate = (double)values[j]/correction;
        Bi_f_slice[row_idx][j] = (int)estimate;
    } /*end for j*/
}

```

```

}

void Write_to_file(int slice)
/******/
{
    for (b=0; b< alpha_bins; b++){
        fprintf(fp,"%5.4f\t",Y_coord[(slice*alpha_bins)+b]);
        for (c=0; c< freq_bins; c++){
            fprintf(fp,"%i\t",Bi_f_slice[c][b]);
        } /*end for c*/
        fprintf(fp,"\n");
    } /*end for b*/
}

void Message()
/*****/
{
    printf("\nProgram completed.\n");
    printf("Cyclic spectral data in fa_spec1.\n");
    printf("\nFFT1_LGTH:\t%i",fft1_lgth);
    printf("\nFFT2_LGTH:\t%i",fft2_lgth);
    printf("\nnfft2stages:\t%i",fft2stages);
    printf("\nend_data:\t%i",end_data);
    printf("\nend_data+1:\t%i",(end_data+1));
}

```

END

1 7 - 1 | 1 - 9 5

FIN