National Defence Défense nationale

DEFENSIO PER SCIENTIAM

MULTICHANNEL M-ARY FREQUENCY-SHIFT-KEYING BLOCK DEMODULATOR IMPLEMENTATION (U)

by

Caroline Tom and Dr. Zaiqing Meng

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
REPORT NO. 1307

Canadä

November 1996
Ottawa

National Defence Défense nationale

MULTICHANNEL M-ARY FREQUENCY-SHIFT-KEYING BLOCK DEMODULATOR IMPLEMENTATION (U)

by

Caroline Tom
MILSATCOM Group
Space Systems and Technology Section

and

Dr. Zaiqing Meng
Royal Military College (Kingston)

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
REPORT NO. 1307

PROJECT
5CA11

November 1996
Ottawa

ABSTRACT

The Military Satellite Communications (MILSATCOM) group at Defence Research Establishment Ottawa (DREO) is performing in-house satellite communications uplink synchronization experiments. A critical part of the experiment is the block demodulation of the multi-channel frequency-hopped waveform. This report analyses the Fast Fourier transform (FFT)-based demodulator implemented by Royal Military College (RMC) Kingston. This system is capable of demodulating a 32 channel, 8-ary frequency-shift-keying (FSK) waveform. The implementation consists of analog-to-digital (A/D) conversion, FFT computation, and high speed input/output (I/O) interface processes using commercial off-the-shelf boards. A description of each of the boards is presented. The operational data flow of the system is examined. For the software implementation, a decimation-in-time approach is used in the digital signal processing (DSP) board to compute the discrete Fourier transform (DFT). A brief description of decimation-in-time algorithms is included. Key components of the assembly language program are outlined. A user's guide detailing the setup requirements for proper operation of the system is included with those required for the DREO application. Steps required for modification to the software are briefly discussed.

RÉSUMÉ

Le groupe MILSATCOM (communications par satellite militaire) au Centre de Recherches pour la Défense à Ottawa (CRDO) exécute les essais de synchronisation de communication par satellite. Une partie critique des essais est la démodulation de la forme d'ondes du système à voies de communication multiple et à spectre étalé à saut de fréquence. Ce rapport analyse le système basé sur les transformées de Fourier rapides qui a été réalisé par le Collège Militaire Royal (CMR) à Kingston. Ce système démodule les signaux transmis par déplacement de huit fréquences sur 32 voies de communication. La réalisation comprend le processus de la conversion analogique-numérique du signal, du calcul des transformées de Fourier rapides, et l'interface d'entrée-sortie des données à haute-vitesse, tout en utilisant des produits commerciaux. Ce rapport présente une description de ces produits. Le fonctionnement du système est examiné. Pour la réalisation du logiciel, les transformées de Fourier rapides discrètes sont calculées par la méthode de décomposition en temps. Une description brève des algorithmes de décomposition en temps est incluse. Les aspects importants du programme écrit en language d'assemblage sont presentés. Un guide d'utilisation est compris dans ce rapport. Ce guide expose en détail les conditions pour la fonctionnement propre du système incluant celles requises pour son usage au CRDO. Les mesures pour modifier le logiciel sont brièvement décrites.

EXECUTIVE SUMMARY

In 1990, Defence Research Establishment Ottawa (DREO) tasked Royal Military College (RMC) Kingston to investigate the feasibility of implementing a Fast Fourier transform (FFT)-based solution to demodulate multiple channels of a frequency-shift-keying (FSK) system. This report is an analysis of the RMC system, describing the theory behind the operation of the system, how it is being used in the current DREO experiments, and how to modify the system for future experiments.

The principle of the FFT-based demodulator for the 32-channel 8-ary FSK system is to perform an FFT over the entire bandwidth of interest and select the output frequency bin in each channel containing the most energy to determine the transmitted symbols. The demodulator system was implemented using commercial off-the-shelf boards developed by Catalina Research Inc. The VME-based implementation consists of an analog-to-digital (A/D) board, a digital signal processing (DSP) board, and a high speed input/output (I/O) interface board. The high speed I/O board was added to the system to bypass the VME-bus data transfer bottleneck so that the output of the system could be provided in real-time. The entire system is contained in a VME chassis with a system controller which uses a 486 processor.

The operation of the system begins with baseband in-phase (I) and quadrature (Q) signals which are provided to the input of the A/D board. The A/D board digitizes the signals and stores the samples in memory. The stored samples are then transferred to the DSP board via the high speed I/O board. Once the FFT computations are completed, the results are again transferred through the high speed I/O board to the front panel.

The Military Satellite Communications (MILSATCOM) group at DREO will be using the system as an A/D-FFT processor in upcoming satellite uplink synchronization trials. As a result, the output of the system has been modified to be the complex FFT frequency bins. The demodulation function will be performed separately.

A user's guide is included which outlines the setup requirements for proper operation of the system. These include the input signals required, the hardware configuration and jumper settings, and software initialization settings.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 Introduction

A common method of demodulating an M-ary frequency-shift-keying (MFSK) signal is based on performing a Fast Fourier Transform (FFT) over the bandwidth of interest. The received symbol is determined by selecting the frequency bin containing the most energy. At first glance, the task described appears to be straightforward. However, in many practical applications a communications system may consist of multiple channels of FSK symbols being transmitted. In such cases, computational speed in order to obtain the demodulated information in a timely manner may need to be considered.

In 1990, Defence Research Establishment Ottawa (DREO) tasked Royal Military College (RMC) Kingston to investigate the feasibility of implementing an FFT-based solution to demodulate multiple channels of a frequency-shift-keying (FSK) system. This demodulator would support the in-house research in spread spectrum communications over a processing extremely high frequency (EHF) satellite. Of particular interest was the investigation of an FFT-based system as an alternative to a surface acoustic wave (SAW)-based system. Preliminary work at Royal Military College (RMC) Kingston included theoretical and simulated studies which are documented in [2,3]. An initial implementation of the MFSK demodulator using simulated data [3] identified some deficiencies in the speed of the input/output (I/O) data transfer. The deficiencies did not allow the demodulator to support the required system throughput. A high speed I/O interface board was later integrated into the system which resolved the data transfer problem.

The Military Satellite Communications (MILSATCOM) group at DREO will integrate this system into the upcoming in-house satellite uplink synchronization trials. It should be noted that for the DREO application, the system will be used as an analog-to-digital (A/D)-FFT processor only, and not contain the logic to select which symbols were transmitted. That part of the demodulator is contained in a separate subsystem and will be described in a future report.

This report analyses the system delivered to DREO, including the work to meet the strict timing constraints of the application and the modifications to support the in-house experiment. A review of the problem which was examined by RMC is provided. A description of the revised hardware and software implementation is given with consideration to the flow of data between the various subsystems. System requirements and specification are identified for proper operation of the demodulator in the upcoming DREO in-house EHF satellite communications (SATCOM) uplink synchronization experiments. Some guidelines on how to modify software parameters which may be required for the trials are also provided.

2.0 Task outline and theoretical considerations

The proposed demodulator processes 32 channels of an M-ary FSK (M=8) system with a frequency separation between FSK tones of Δf = 36 kHz. The equation for the transmitted signal in such a system can be expressed as follows [2]:

$$s_{32}(t) = \sum_{n=1}^{32} A_n \cos(2\pi\,(f_n + i_n\,\Delta f\,t)) \qquad 0 \le t \le T_{symbol} \qquad i_n = 0,\ldots(M-1) \tag{1}$$

where

$s_{32}(t)$ = transmitted signal in a 32 channel system
n = channel number
A_n = amplitude of FSK tone in channel n
f_n = base frequency of channel n
i_n = FSK tone number for channel n
t = variable time
T_{symbol} = FSK symbol period = 55.56 µs

The minimum bandwidth requirement for each channel is $BW_{ch} = 8 \times \Delta f$ = 288 kHz and for the 32-channel system is $BW_{32ch} = 32 \times BW_{ch}$ = 9.216 MHz as shown in Fig.2.1.

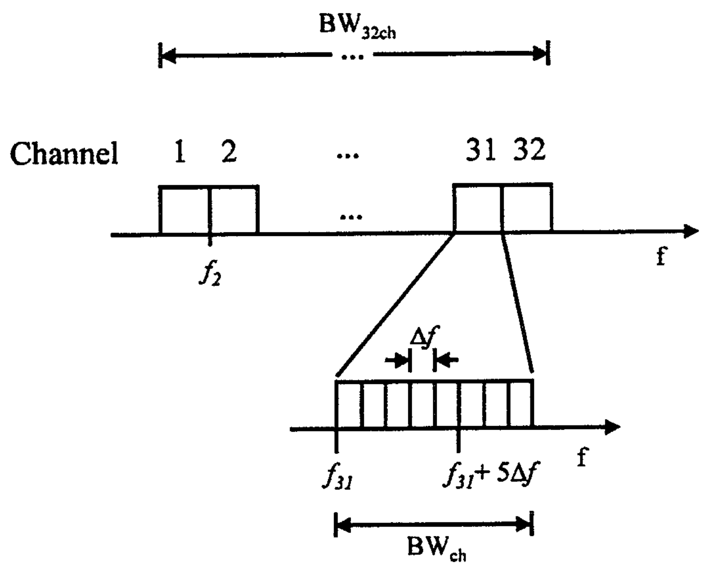


Fig.2.1. Frequency representation of 32-channel 8-ary FSK system

For a 32-channel system such as shown in Fig.2.1, one possible method to determine the demodulated information, is to perform an FFT over the entire bandwidth of interest and select the appropriate channel and frequency bin with the most energy. After a preliminary study of the problem carried out by RMC [2], an implementation for the demodulator was proposed. In the proposal, a received signal of the 32-channel system is bandpass filtered according to BW_{32ch} and translated via quadrature detection into in-phase (I) and quadrature (Q) components. The I and Q signals are sampled at $f_s = 9.216$ MHz (BW_{32ch}) thereby satisfying Nyquists' Sampling Theorem [4]. The proposed FFT-based demodulator then performs an FFT on the sampled signals. The transmitted FSK symbol(s) can subsequently be detected by selecting the bin in the appropriate channel(s) with the most energy. A block diagram illustrating the demodulation process is given in Fig.2.2.

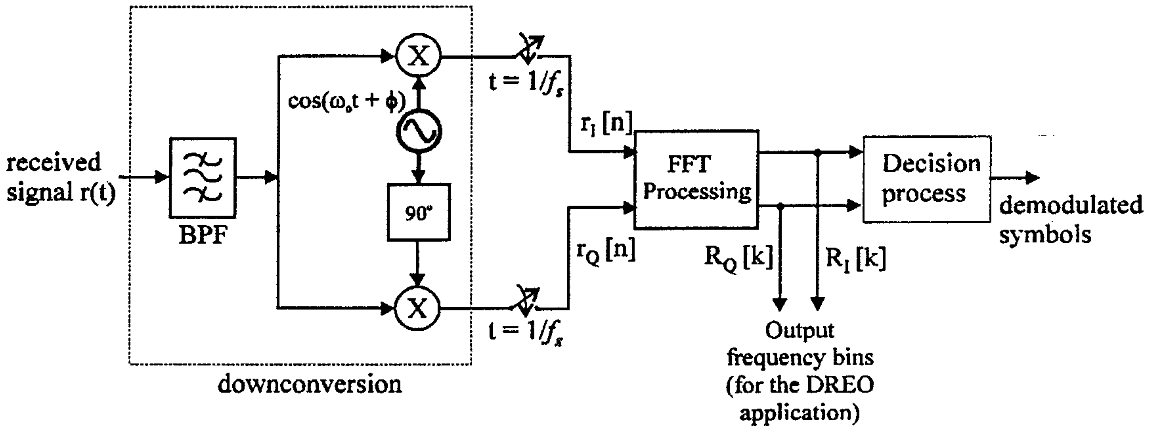


Fig.2.2. Block diagram of proposed demodulation process

The application of the demodulator in support of DREO's in-house R&D work involves the processing of multiple channels of frequency-hopped signals employing FSK modulation. One FSK symbol is to be transmitted per hop in each channel. For the DREO application, the required system output is simply the FFT output frequency bins in I and Q. Subsequent processing to demodulate the signal is to be performed elsewhere. In order to obtain the processed information on a hop basis, the data throughput of each stage of the system must be less than one hop period (approximately 56 μs). For the implementation, it is assumed that the I and Q signals are available from the downconverter. This report covers the sampling, analog-to-digital (A/D) conversion, and the FFT processing components identified above. In addition, a high speed I/O interface board integrated to meet the data throughput requirement is also described.

3.0 Implementation details

This section provides a description of the implementation of an FFT-based demodulator as proposed by RMC [3] and includes both hardware and software details. The demodulator used commercial off-the-shelf products developed by Catalina Research Incorporated (CRI). In section 3.1, the hardware for the demodulator and its key features are described, as well as the overall operation and data flow. Section 3.2 discusses the software implementation of the FFT algorithm for the demodulator.

3.1 System description and operation

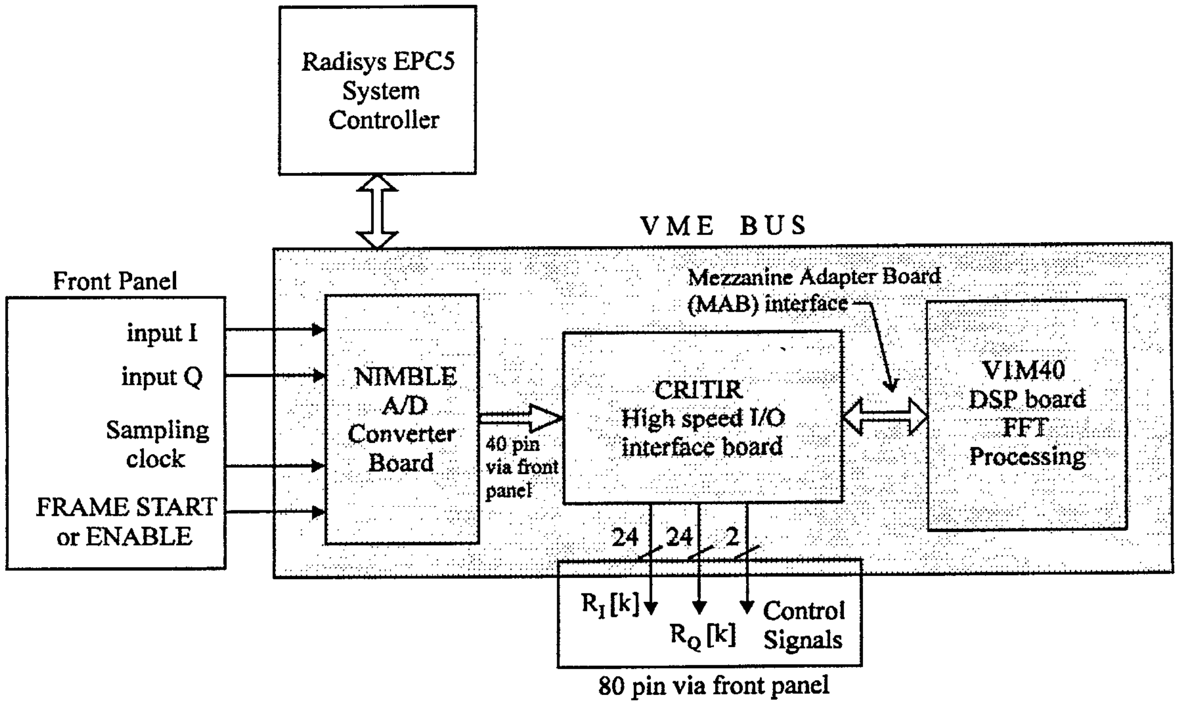

Radisys EPC5 System Controller
VME BUS
Front Panel
input I
input Q
Sampling clock
FRAME START or ENABLE
NIMBLE A/D Converter Board
40 pin via front panel
Mezzanine Adapter Board (MAB) interface
CRITIR High speed I/O interface board
VIM40 DSP board FFT Processing
24
24
2
$R_I[k]$
$R_Q[k]$
Control Signals
80 pin via front panel

Fig.3.1. Overall system block diagram of FFT-based demodulator

An overall system block diagram of the FFT-based demodulator implementation is given in Fig.3.1. A CRI board, called 'NIMBLE' performs the A/D conversion of an input signal. The CRI NIMBLE board consists of a digital mother board and an A/D daughter card. The daughter card used in this implementation is a dual channel 10 MHz-12 bit card which performs the A/D conversion function. The digital motherboard houses four synchronous FIFOs (two per channel) of up to 8 kwords per channel. The NIMBLE board is designed in the form of a single slot 6U VME card, programmable via the VME bus with software driver routines provided by CRI. Further details on the configuration and setup of the NIMBLE board are given in Section 4.0 and in the NIMBLE User's Guide supplied by CRI. [5].

Another CRI board called 'V1M40', based on SHARP's LH9124/LH9320 DSP technology, is employed as an FFT processor, the core of the demodulator. The V1M40 is a VME version of the high speed digital signal processing board containing the SHARP LH9124/LH9320 floating-point DSP chip set. A block diagram of the V1M40 architecture is included in Fig.3.2. The use of the LH9320 address generators makes the addressing of the FFT arrays during the computations very efficient. The V1M40 design incorporates a high speed I/O bus which is available to transfer data arrays into and out of the V1M40 board at rates of up to 40 MHz without being limited by the host VME bus transfer speeds. This feature is exploited for this implementation, along with a high speed I/O interface card, to allow for processing of the received signal on a hop basis. The DSP high speed I/O bus consists of 48-bit data, 24 bits I and 24 bits Q. Control signals are available for control of the V1M40 I/O bus. Specific configuration and setup issues for the V1M40 board are addressed in Section 4.0 and in the 1M40 User's Guide supplied by CRI. [6].

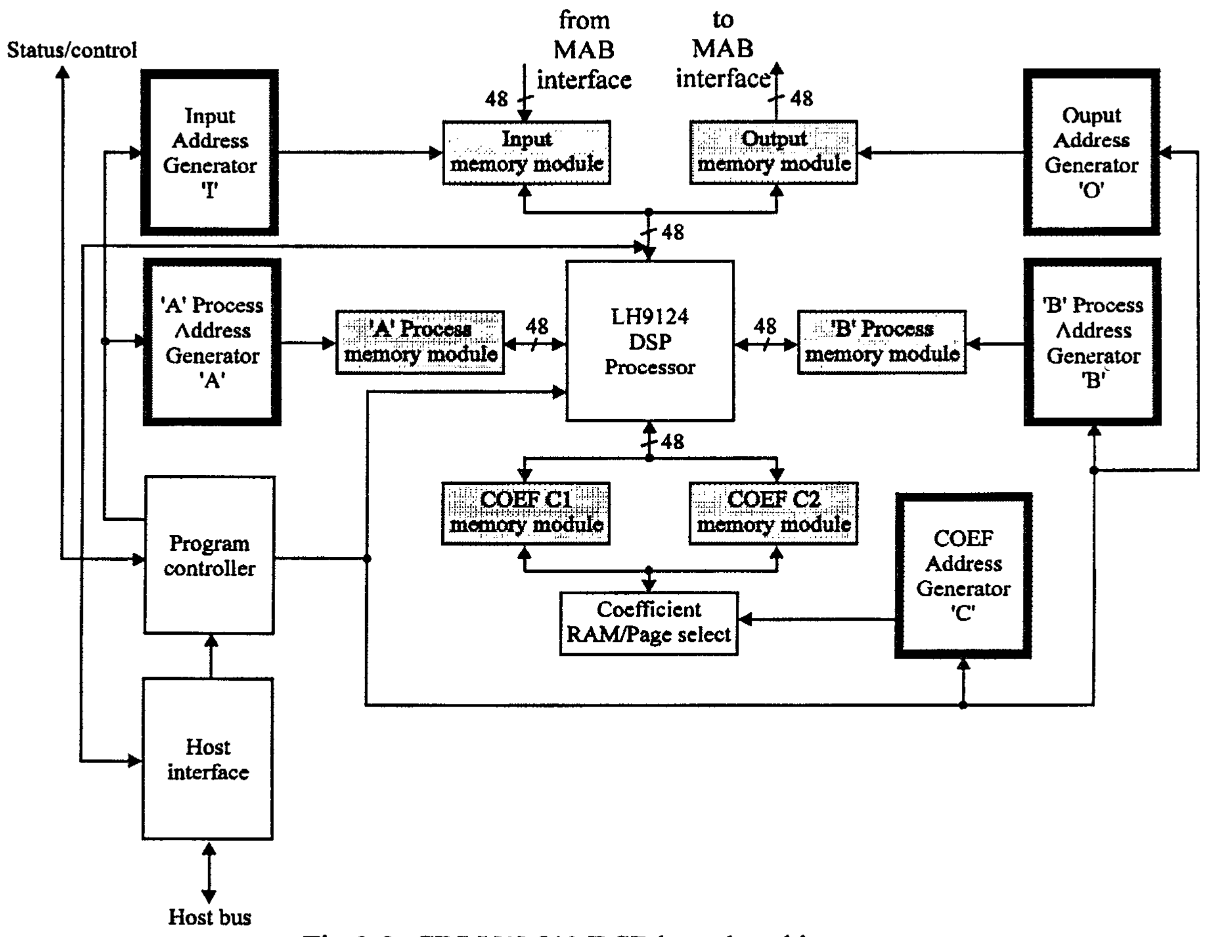

Status/control
from MAB interface
to MAB interface
48
48
Input Address Generator 'I'
Input memory module
Output memory module
Ouput Address Generator 'O'
48
'A' Process Address Generator 'A'
'A' Process memory module
48
LH9124 DSP Processor
48
'B' Process memory module
'B' Process Address Generator 'B'
48
COEF C1 memory module
COEF C2 memory module
COEF Address Generator 'C'
Program controller
Coefficient RAM/Page select
Host interface
Host bus

Fig.3.2. CRI V1M40 DSP board architecture

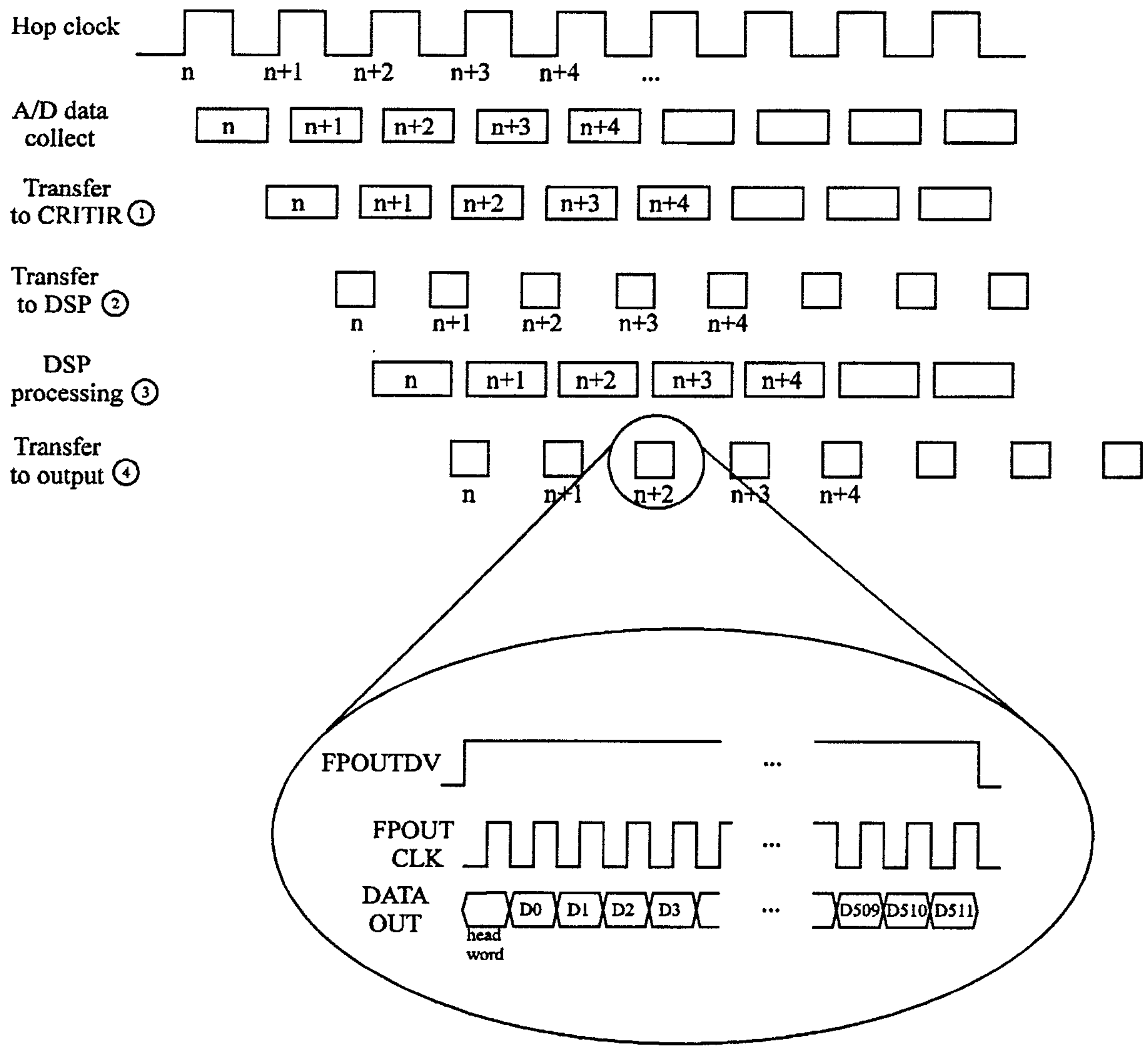
The CRI 'CRITIR' board is used for inter-board high speed parallel interfacing. It was developed to serve as an off-the-shelf high speed I/O interface for the V1M40 processing board. Although the V1M40 can process data utilizing the VME bus as the input and output medium, the VME bus bandwidth cannot sustain the I/O rate capabilities of the V1M40 DSP board as described in the previous paragraph. The CRITIR board allows the VME bottleneck to be bypassed by transferring input and output data from the V1M40 high speed I/O connectors. The CRITIR board also interfaces to the NIMBLE A/D board. This enables a direct hook up of the NIMBLE board to the CRITIR board, again avoiding the data transfer problem posed by the VME bus. Additional information on the configuration and setup of the CRITIR board is provided in Section 4.0 and the CRI 'CRITIR' User's Guide [7].

The entire system is hosted in a VME chassis controlled by a VME controller, RadiSys EPC-5 (486), which is accompanied by a hard drive, a floppy drive, a mouse, a keyboard and a VGA monitor.

Operational control of the A/D-FFT system is effected by the use of external trigger signals, as indicated in Fig.3.1. In this implementation, the user is required to provide an appropriate external sampling clock (conversion clock) for the A/D operation. In addition, a signal called 'FRAME START' is required which enables/disables the collection of data. By using 'FRAME START', the user can effectively select specific intervals during which demodulation is required. At the output of the demodulator, 24-bit I and Q results are synchronized with control signals generated by the CRITIR board.

Samples of the I and Q channels of the received signal are taken and stored by the NIMBLE board, and then transferred into the high-speed parallel interface of the V1M40 DSP board via the CRITIR interface board. When the FFT of the data has been completed, the results are transferred to a parallel output port through the CRITIR I/O board again. For the DREO application, the output of the system is chosen to be the FFT components of the received signal in I and Q. Subsequent processing to determine the FSK symbol transmitted is done elsewhere. Fig.3.3 illustrates an example of the sequence of events for this demodulator.

The software used by the V1M40 DSP board to compute the FFT of the received signal is written in SHARP LH9124/LH9320 assembly language. The compiled program is loaded into V1M40 program memory and invoked by a system management program. The system management program, written in C, provides the system hardware initialization and user interface. Custom routines for system initialization have been provided by CRI. A further description of the assembly language and system management programs is given in Section 3.2.



① Data transfer from Nimble FIFO to CRITIR FIFO
(512 samples @ 9.216 MHz ≈ 56μs)

② Data transfer from CRITIR FIFO to DSP memory
(512 samples @ 20MHz ≈ 26μs)

③ DSP processing - 512-point FFT (Radix-2, Radix-16, Radix-16 computation)

④ Data transfer from DSP memory to front panel
(512 samples @ 20MHz ≈ 26μs)

Fig.3.3. System operation flow diagram

3.2 Software implementation

3.2.1 FFT processing algorithm

3.2.1.1 Background of FFT algorithm used in DSP board

The LH9124 DSP is used to perform the discrete Fourier transform (DFT) computations for the demodulator. These computations are carried out using a decimation-in-time (DIT) approach. The equation for the DIT algorithm to compute an N-point DFT can be written as follows [8]:

$$X[k] = \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{rk} + \sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{rk} \qquad k = 0,1,\ldots,N-1 \tag{2}$$

where

$X[k] =$ k^{th} output frequency bin
$x[2r]$ = even-numbered time sequence samples
$x[2r+1]$ = odd-number time sequence samples
$W_{N/2}^{rk}$ = twiddle factor

Equation (2) effectively breaks the original N-point DFT into two (N/2)-point DFT computations. Similarly, the (N/2)-point DFTs can each be broken down into 2 (N/4)-point DFTs. For N-points, this process can be continued until left with a series to two-point transforms, called "butterflies" (i.e. $\nu = \log_2 N$ times). An example of a DIT FFT is given for N=8 in Fig.3.4.

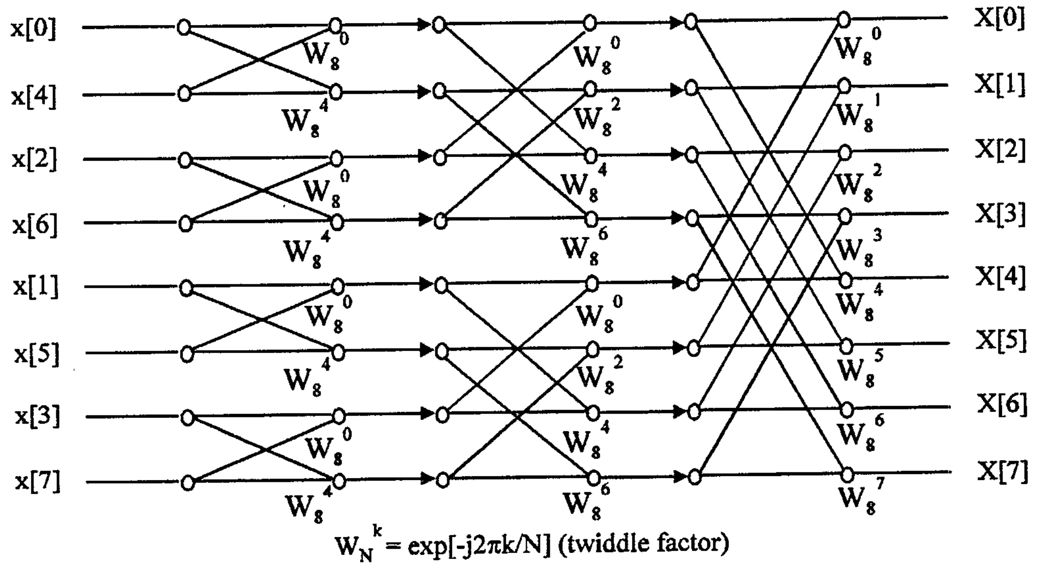


Fig.3.4. Flow diagram of decimation-in-time DFT

Further simplification of the flow diagram can be obtained by considering that the general butterfly computation is of the form

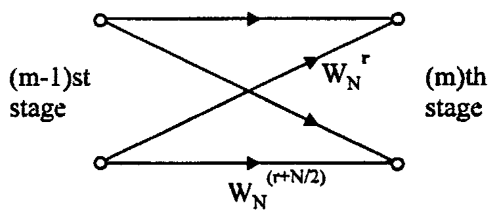


Fig.3.5. General representation of butterfly computation

and

$$W_N^{N/2} = -1$$
$$W_N^{r+N/2} = W_N^{N/2}\, W_N^r = -W_N^r$$

so that the flow diagram can be redrawn as shown in Fig.3.6.

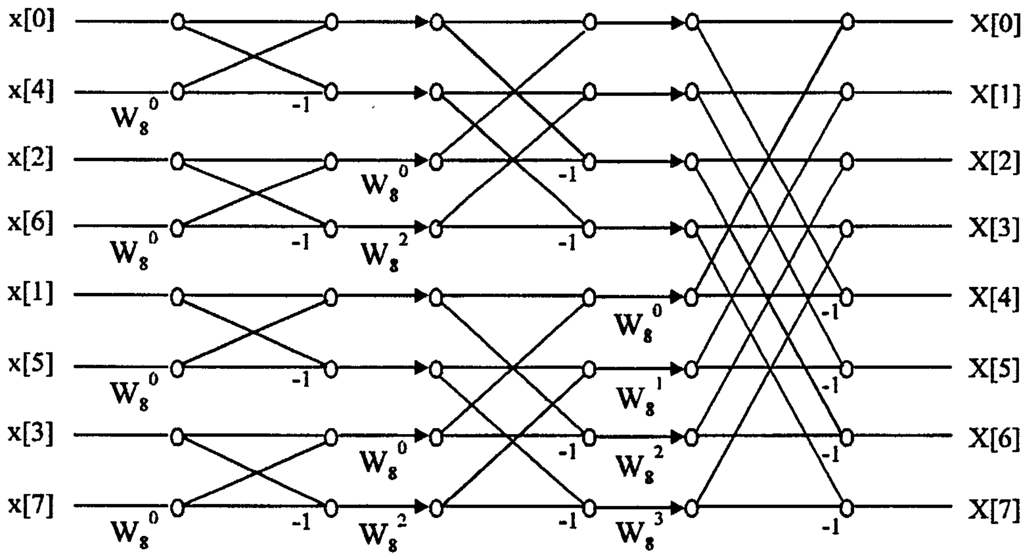


Fig.3.6. Flow diagram of simplified decimation-in-time DFT

Fig.3.4 and Fig.3.6 illustrate the case where N=8 has been decomposed into $N = 8 = 2^v = R^v$, where R is the radix. As such, this is called a Radix-2 algorithm. Again, $\nu = 3$ refers to the number of stages or passes in the algorithm. However, it may be that N is decomposed into other radices. For example, if $N = 256$, it can be broken down into 2^8, 4^4, or 16^2. The latter two are Radix-4 and Radix-16 operations respectively. The LH9124 DSP chip supports Radix-2, Radix-4, Radix-8, Radix-16 algorithms [9].

In the DREO application, the DFT is required for sequences of 512 complex samples. Each sequence is broken down as $N = 512 = 2x16x16$ so that the 512-point DFT is computed by computing a Radix-2 DFT, followed by two Radix-16 DFTs.

3.2.1.2 Assembly language program

The assembly code for computing a complex 512-point FFT is stored in a file called "fft512.asm" which resides on the hard drive of the system controller. A copy of the program listing is included in Appendix A. The program is written as a loop which continuously accepts input data from the CRITIR board, processes the various stages of the FFT and outputs data via the CRITIR board again as mentioned in Section 3.1.

The FFT processing is divided into stages according to the decomposition of the 512-sample sequence for the DIT algorithm. In this implementation, the 512-point sequence is decomposed as N=512=2x16x16 so that a complex 512-point FFT is computed in three stages consisting of Radix-2, Radix-16, and Radix-16 algorithms respectively. The V1M40 architecture (see Fig.3.2) allows for the processing of these stages to be pipelined through the various memory modules as follows:

a) input data in "Q" memory module
b) 1st pass, Radix-2 computation, data moves from "Q" to "A" memory module
c) 2nd pass, Radix-16 computation, data moves from "A" to "B" memory module
d) 3rd pass, Radix-16 computation, data moves from "B" to "Q" memory module

A general flow diagram of the assembly routine is given in Fig.3.7. The program is basically separated into two parts: initialization of the five address generators (I,O,A,B,C); and performing the FFT.

In this implementation, the following parameters for each address generator are set (refer to Appendix A and [10]):

a) set to internal memory operation
b) initialize program memory registers (total 32) including:
    (i) LATENCY register
    (ii) PCSTART register (points to first address patten to be executed in program memory
    (iii) MODE register
    (iv) N register (length of address sequence)
    (v) ADRSTART register (starting address in the sequence)
    (vi) ADRINC register (address increment)
    (vii) ADRLENGTH (number of points in sequence)
    (viii) DIGREV register
    (ix) MEMSIZE register
    (x) PCEND register (points to last address pattern to be generated)

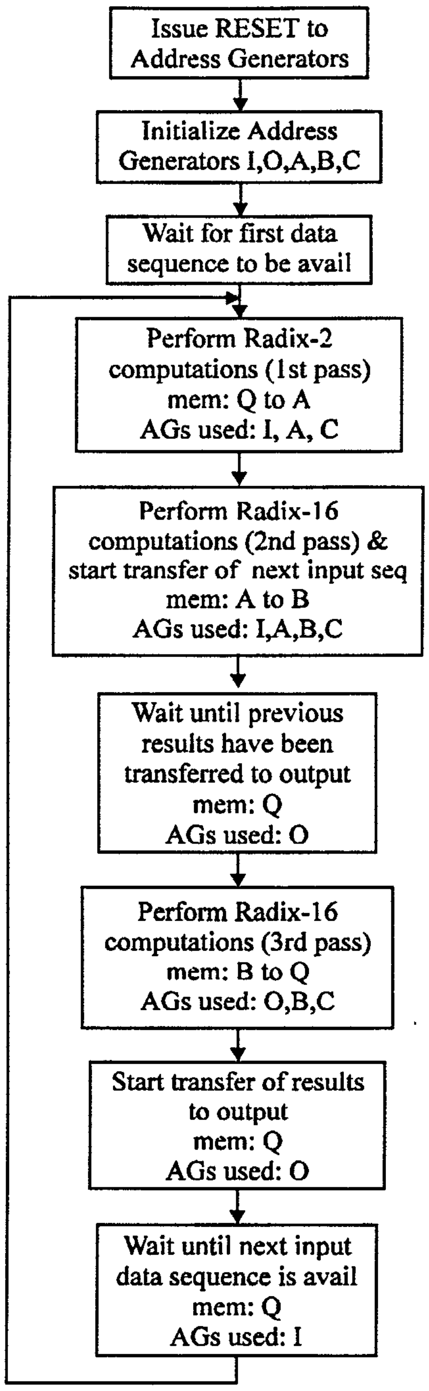

Issue RESET to Address Generators
Initialize Address Generators I,O,A,B,C
Wait for first data sequence to be avail
Perform Radix-2 computations (1st pass) mem: Q to A AGs used: I, A, C
Perform Radix-16 computations (2nd pass) & start transfer of next input seq mem: A to B AGs used: I,A,B,C
Wait until previous results have been transferred to output mem: Q AGs used: O
Perform Radix-16 computations (3rd pass) mem: B to Q AGs used: O,B,C
Start transfer of results to output mem: Q AGs used: O
Wait until next input data sequence is avail mem: Q AGs used: I

Fig.3.7. Flow diagram of DSP assembly routine

The initialization of memory registers is carried out using two instructions where the first instruction points to the address of the memory register and the second is the 8-bit value to be stored. The general format is as follows:

$$I\_AG\_PROC\ A = 1,\ DB = 0$$
$$I\_AG\_PROC\ A = 0,\ DB = 71$$

The first instruction points to the address of the memory register, with A=1 indicating that the value in DB (in hexadecimal) is an address. The second instruction stores the value of DB in that memory register, with A=0 to indicate that the value in DB is data. The result of these two instructions is to store 71h at address 0 of the "I" address generator. It follows that to load a 24-bit value, these two instructions are executed three times for three contiguous address locations. An example of a 24-bit load is the initialization of the N register for the "I" address generator shown in Appendix A.

The second part of the program relates to the computations for the 512-point complex FFT. As described above, the algorithm is divided into three stages or passes. The general format of each pass is:

DSP_ENA RQWA, BFLY2:
DSP_ENB:
START I A C:
WAIT I A C:

The *DSP_ENA* and *DSP_ENB* instructions initialize the DSP for the specified mathematical operation (in the above example, it is *BFLY2* - Radix-2 FFT). *RQWA* indicates the direction of data flow (in this case, read from "Q" memory module and write to "A" process memory module). Finally, *START* and *WAIT* instructions, along with the address generators specified, initiates the process and pauses until the computations for the mathematical operation have been completed, respectively.

It should be noted that once the computations for the first sequence of 512 samples is initiated, the input of subsequent data samples and output of results overlap the stages the FFT computations as illustrated in Fig.3.3.

3.2.2 System management program

The system management program for the FFT-based demodulator is written in C. It is stored as "rmciq.c" on the hard drive. A copy of the program listing is included in Appendix B. The system management program initializes the three CRI boards and downloads the assembly language program to the DSP program memory and the coefficients file into DSP memory. All functions performed by the system management program are executed by subroutines which have been supplied by CRI. The reader is referred to the user manual for each of the CRI boards for further details on the subroutines. The subroutines are stored on the hard drive of the demodulator system.

4.0 User's Guide - System Setup

The following section outlines the physical requirements and steps necessary to properly operate the demodulator system for the DREO application. Further information on the setup may be obtained directly from documentation supplied by CRI [5,6,7].

4.1 Hardware configuration

4.1.1 NIMBLE A/D board

The NIMBLE A/D board is manufactured in the VME 6U 160mm standard Eurocard format and is shown in Fig.4.1. The A/D conversion is performed by a daughter board mounted directly on the VME card. The following paragraphs describe the external signals, hardware (jumper) settings, and software settings that are required for the current implementation of the board.

4.1.1.1 NIMBLE external signals

All the external signals and connections required for the NIMBLE board are accessible from the front panel of the board and are described in the following paragraphs. Specific requirements for the DREO application of the A/D-FFT demodulator system for upcoming EHF SATCOM uplink synchronization trials are also included. Further information on these signals may be found in [5].

4.1.1.1.1 CONVERT CLOCK

The A/D conversion of the input signal takes place on the rising edge of the CONVERT CLOCK. This signal is identified on the front panel as "CLK" and requires an SMA connector with 75Ω input impedance. A 50% duty cycle, TTL-level clock signal is required with a maximum frequency of 10 MHz. For the DREO application, a 9.216 MHz signal will be used.

4.1.1.1.2 /SAMPLE GATE

The /SAMPLE GATE signal allows the user to control the WRITE ENABLE signal of the FIFOs on the A/D board. It is indicated on the front panel of the A/D board as "GATE". For the DREO application, this signal has been disabled by setting bit '3' in CONTROL REGISTER ONE of the A/D board to a '1' via the system management program [refer to Appendix B]. By doing so, this allows only FRAME START to affect the sampling of the analog input signal.

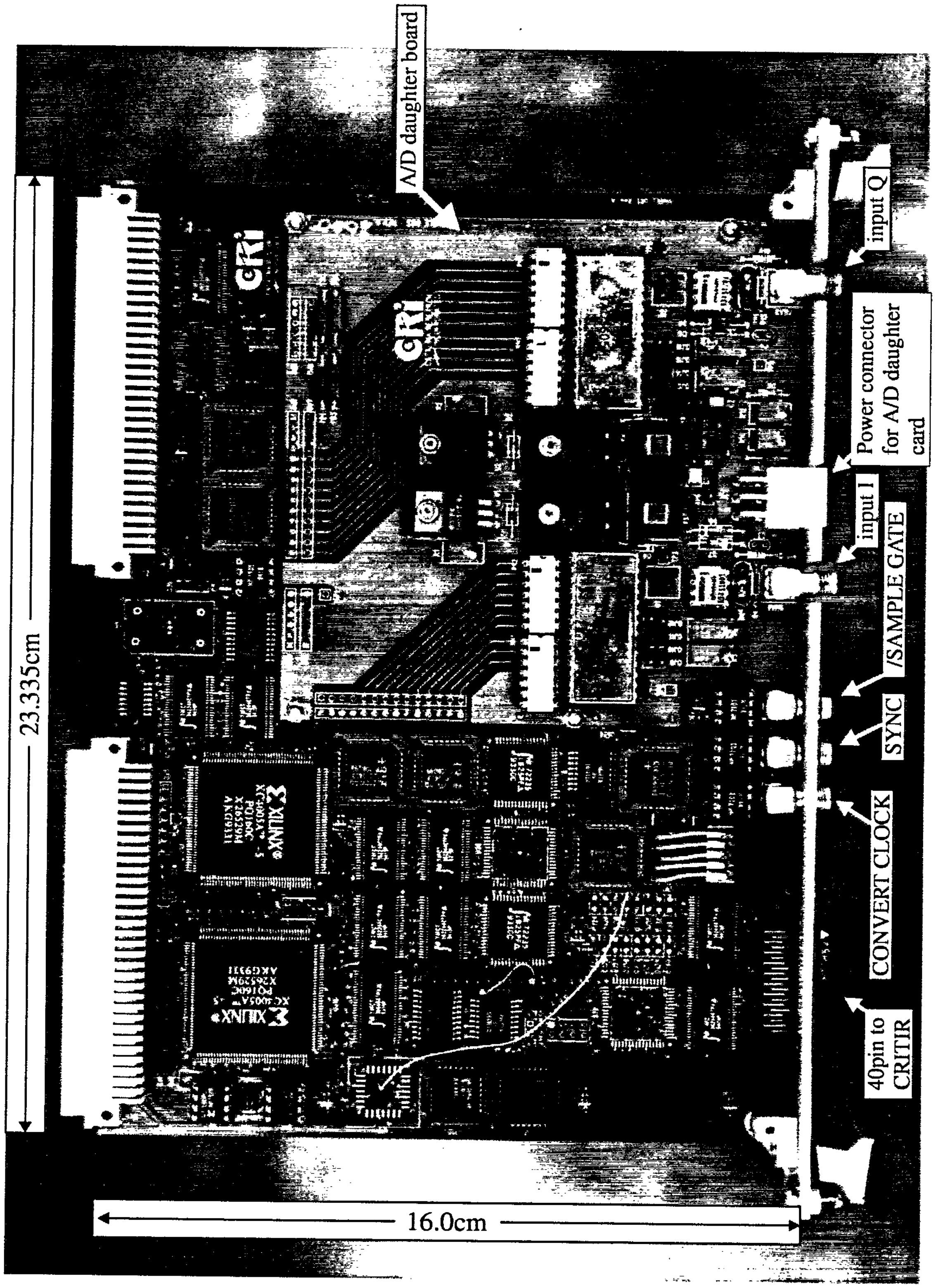


Fig.4.1. CRI NIMBLE A/D board (top view)

4.1.1.1.3 FRAME START

The A/D counter is loaded with the number of samples to be collected by the A/D board on the rising edge of the CONVERT CLOCK signal after the rising edge of FRAME START. In addition to initializing the A/D counter, the FRAME START signal serves as the A/D FIFO WRITE ENABLE signal after a three cycle CONVERT CLOCK delay. The FRAME START signal is identified on the front panel by the SMA connector with the label "SYNC". It has a 75Ω input impedance. An active high, TTL-level signal is required. For the DREO application, a 16kHz square wave signal is supplied with an 88% duty cycle (i.e. in one cycle, the signal is 'high' for 55.6μs and 'low' for 6.9μs).

4.1.1.1.4 Input signals I and Q

The two input signals are found on the front panel with the SMA connectors marked "CH1" and "CH2" for "I" and "Q" respectively. These signals also have 75Ω input impedance. The input signals are specified to have a maximum voltage of ±1 volt.

4.1.1.1.5 Power supply for A/D daughter card

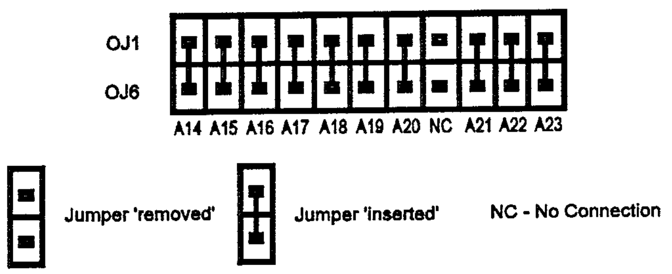
±12 volts is required for the A/D daughter card via a front panel connection using the connector specified in [5].

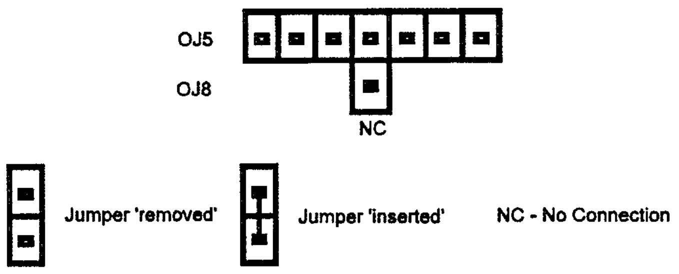
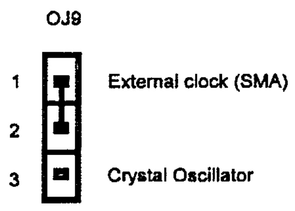
4.1.1.1.6 High speed parallel connection between NIMBLE and CRITIR

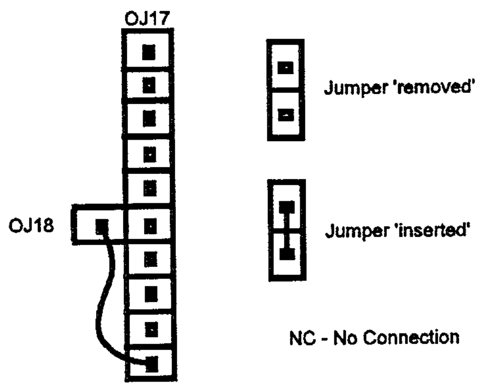
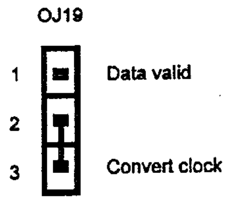
A 40-pin connector with high density ribbon cable is used to connect the NIMBLE board to the CRITIR I/O board. The pinout of the connector is given in Section 4.2.1.

4.1.1.2 NIMBLE hardware settings on the digital motherboard

The NIMBLE board is configured via jumpers for the DREO application as follows:

OJ1, OJ6	VMEbus Base Address Selection. Jumper inserted signifies a 'LO' and jumper removed signifies a 'HIGH'  OJ1 OJ6 A14 A15 A16 A17 A18 A19 A20 NC A21 A22 A23 Jumper 'removed' Jumper 'inserted' NC - No Connection

OJ2, OJ4	Reserved. Jumper removed.
OJ3	VME IACKOUT* Signal: Jumper removed.
OJ5, OJ8	VME Interrupt Level Selection. Jumper removed.  OJ5 OJ8 NC Jumper 'removed' Jumper 'inserted' NC - No Connection
OJ9	Clock Source Selection.  OJ9 1 2 3 External clock (SMA) Crystal Oscillator
OJ10	Output data format. Jumper inserted for the DREO application. (2s complement)
OJ12, OJ13, OJ14, OJ15, OJ29	Interconnections for A/D daughter card. Jumpers are factory set.
OJ16	Selects A16 addressing (jumper 'inserted') or A24 addressing (jumper 'removed'). Jumper is inserted for the DREO application.

OJ17, OJ18	CVT CLK Delay Selection. Jumper is inserted for a 5ns delay (default) for the DREO application.  OJ17 OJ18 Jumper 'removed' Jumper 'inserted' NC - No Connection
OJ19	CONVERT CLOCK/DATA VALID from A/D select. Jumper is inserted for the DREO application as follows: 
OJ20 to OJ41 (except OJ36, OJ37)	FIFO configuration jumpers. Jumpers are factory set.
OJ36	External Ground/Digital ground. Jumper is inserted for DREO application indicating external ground is tied to digital ground.
OJ37	+5 volts external/ +5 volts digital. Jumper is inserted for DREO application indicating external +5 volts is tied to +5 volts digital.

4.1.1.3 NIMBLE hardware settings (A/D daughter card)

The NIMBLE A/D daughter card is configured via jumpers for the DREO application as follows:

OJ1, OJ6	Jumpers are factory set.
OJ2, OJ7	Jumpers are factory set.
OJ3, OJ8	Jumpers are factory set.
OJ4, OJ5	Jumpers are factory set.
OJ9	Jumper is removed.
OJ10	Jumper is inserted.
OJ11	Jumper is inserted.
OJ12	Jumper is removed.
OJ13	Jumper inserted.
OJ14	Jumper is removed.
OJ15	Jumper is inserted.
OJ16	Jumper is removed.

4.1.1.4 NIMBLE software settings

The NIMBLE A/D board is configured in software by an ASCII text configuration file (included in Appendix C) and by initializing registers in the system management program (see Appendix B). For the DREO application, the NIMBLE board is set to operate with dual channel write to FIFO 1 and FIFO 2 and dual channel continuous read via the WRITE MODE register and READ MODE register respectively. Further details on these registers can be obtained from [5].

4.2 CRITIR I/O interface board

The CRITIR I/O interface board acts as the common area for the A/D-FFT system and is shown in Fig.4.2 and Fig.4.3. The CRITIR board receives the data collected by the NIMBLE A/D board via the 40-pin ribbon connector on the front panel. The board then transfers the data to the V1M40 DSP board, which is connected by a Mezzanine Adapter Board (MAB) interface, for FFT processing. Once the processing is completed, the CRITIR receives the results from the V1M40 DSP board and transfers the data at high speed through the 80-pin output connector on the front panel for custom applications. The following section describes the physical connections that are required.

4.2.1 CRITIR hardware connections

The interface between the CRITIR and the NIMBLE boards is a 40 pin connector on the front panel using high density ribbon cable. The pinout of the connector is shown in Fig.4.4. The output interface of the CRITIR board is an 80 pin connector on the front panel using high density ribbon cable. The pinout of the connector is shown in Fig.4.5.

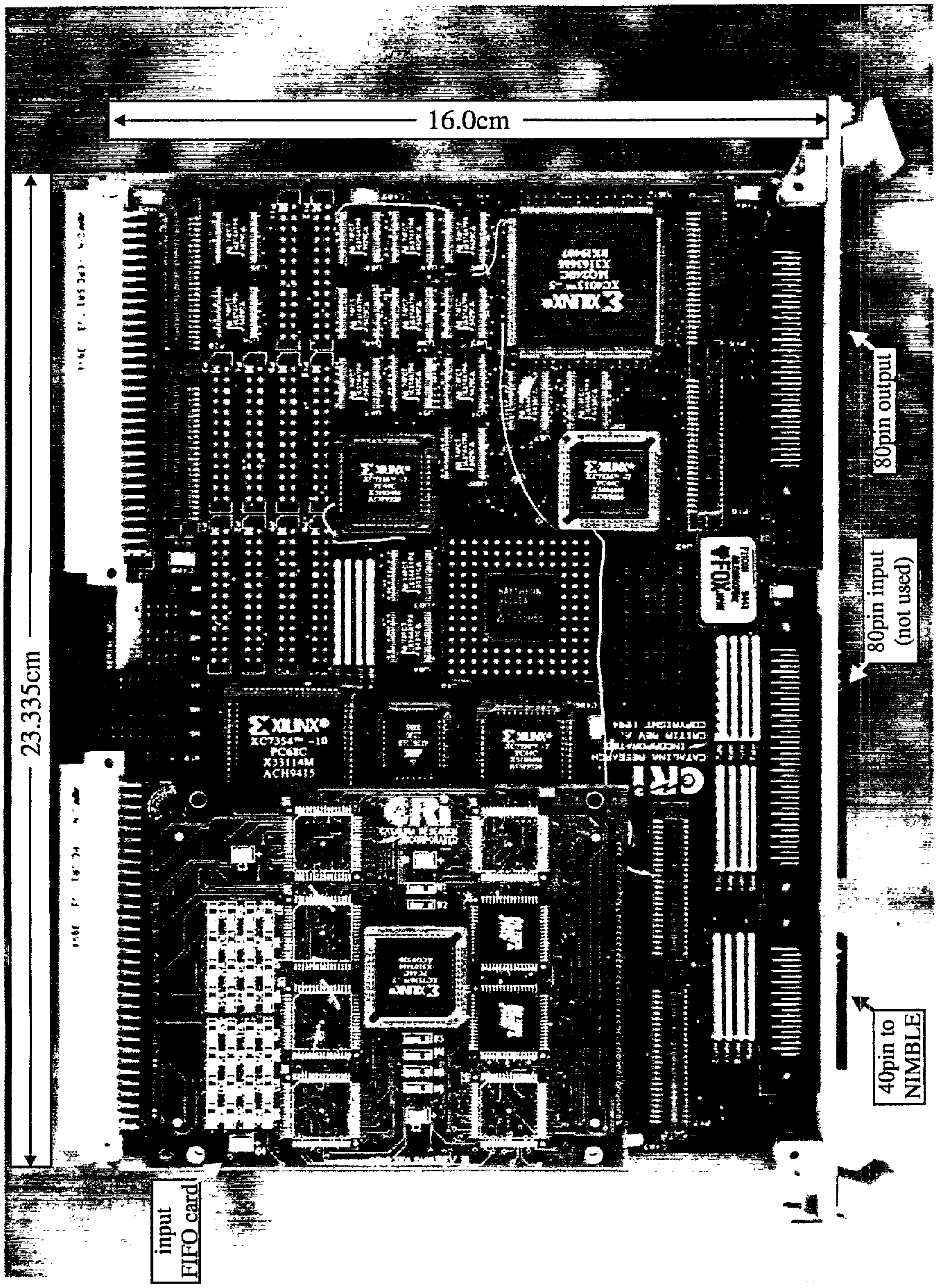


Fig.4.2. CRI CRITIR high speed interface board (top view)

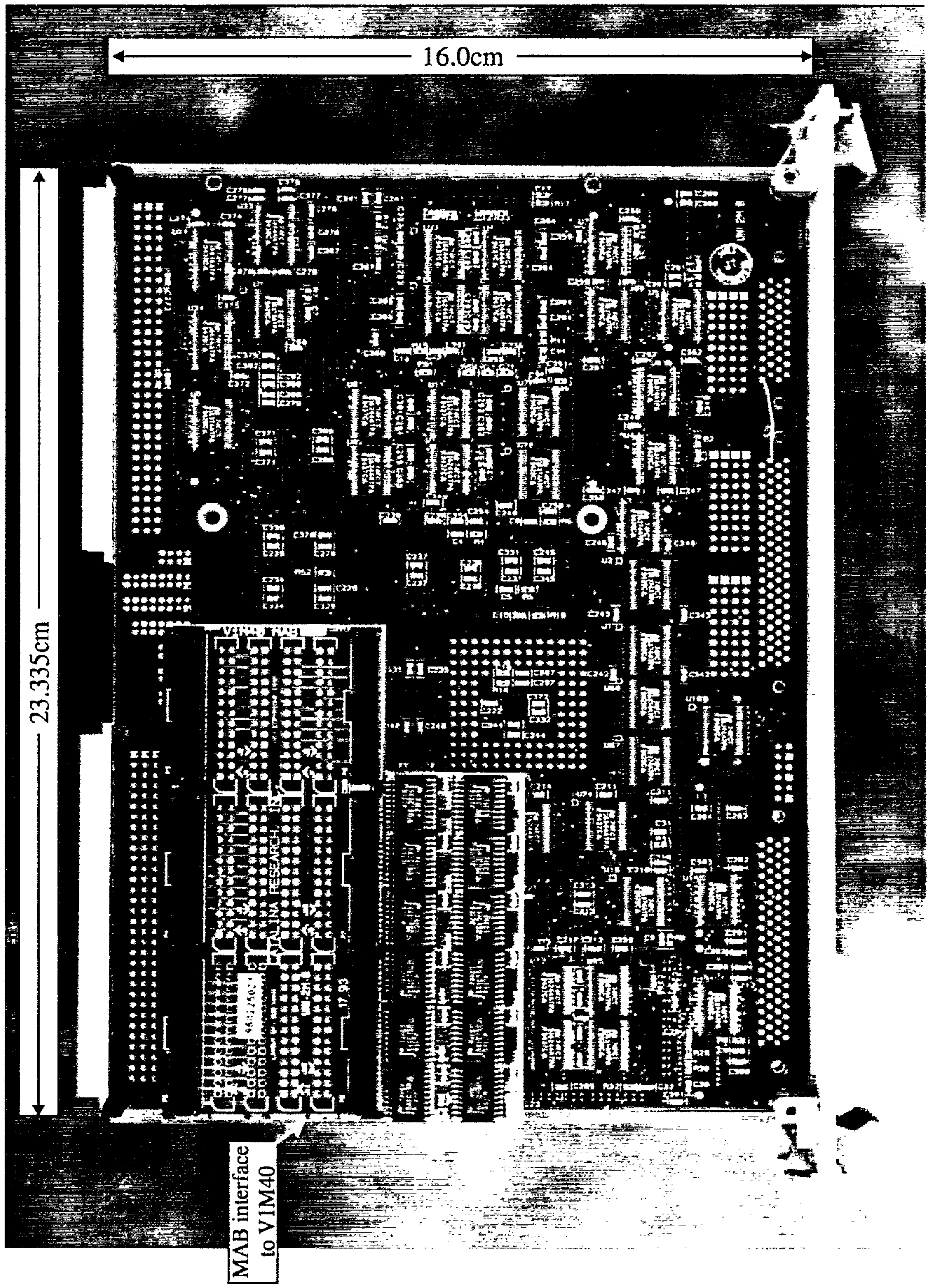


Fig.4.3. CRI CRITIR high speed interface board (bottom view)

SIGNAL NAME	PIN #	SIGNAL NAME	PIN #
DATA_VALID	1A	ADQ2	11A
	1B	ADQ3	11B
ADI0	2A	ADQ4	12A
ADI1	2B	ADQ5	12B
ADI2	3A	ADQ6	13A
ADI3	3B	ADQ7	13B
ADI4	4A	ADQ8	14A
ADI5	4B	ADQ9	14B
ADI6	5A	ADQ10	15A
ADI7	5B	ADQ11	15B
ADI8	6A	ADQ12	16A
ADI9	6B	ADQ13	16B
ADI10	7A	ADQ14	17A
ADI11	7B	ADQ15	17B
ADI12	8A		18A
ADI13	8B	ADINPCLK	18B
ADI14	9A		19A
ADI15	9B	ADVSYNC	19B
ADQ0	10A		20A
ADQ1	10B		20B

Fig.4.4. CRITIR - input 40-pin connector (interface to NIMBLE A/D board)

SIGNAL NAME	PIN #	SIGNAL NAME	PIN #
Sheild Ground #1	1A	DQ13	21A
Shield Ground #2	1B	DQ14	21B
DI0	2A	DQ15	22A
DI1	2B	DQ16	22B
DI2	3A	DQ17	23A
DI3	3B	DQ18	23B
DI4	4A	DQ19	24A
DI5	4B	DQ20	24B
DI6	5A	DQ21	25A
DI7	5B	DQ22	25B
DI8	6A	DQ23	26A
DI9	6B	GND	26B
DI10	7A	BFPI0	27A
DI11	7B	BFPI1	27B
DI12	8A	BFPI2	28A
DI13	8B	BFPI3	28B
DI14	9A	BFPI4	29A
DI15	9B	BFPI5	29B
DI16	10A	GND	30A
DI17	10B	DSFI0	30B
DI18	11A	DSFI1	31A
DI19	11B	DSFI2	31B
DI20	12A	Reserved	32A
DI21	12B	Reserved	32B
DI22	13A	Reserved	33A
DI23	13B	GND	33B
GND	14A	CLK	34A
DQ0	14B	GND	34B
DQ1	15A	VSYNC	35A
DQ2	15B	GND	35B
DQ3	16A	DATAVALID	36A
DQ4	16B	GND	36B
DQ5	17A	Reserved	37A
DQ6	17B	GND	37B
DQ7	18A	Spare 1	38A
DQ8	18B	GND	38B
DQ9	19A	Spare 2	39A
DQ10	19B	GND	39B
DQ11	20A	Spare 3	40A
DQ12	20B	GND	40B

Fig.4.5. CRITIR - OUTPUT 80-pin connector (interface to custom interface board)

4.2.2 CRITIR I/O interface board jumper settings

The jumpers on the CRITIR I/O interface board are used to to set up the VME base address location and address modifier type. More information on these jumpers settings is available from [7]. For the DREO application, the CRITIR I/O interface is configured as follows:

W1, W10	Select VME master mode. Currently not used in the DREO application.
W5, W8	Jumpers are factory set.
W2	VME address mode. Jumper 'inserted' indicates in A24 mode. Jumper 'removed' indicates A32 mode. For the DREO application, the jumper is inserted.
W6	VME address bit. For the DREO application, all eight jumpers are inserted.
W4	VME interrupt level. For the DREO application, all seven jumpers are removed.

4.2.3 CRITIR I/O interface board software settings

As with the NIMBLE A/D board, parameters of the CRITIR I/O interface board are initialized in the system configuration file included in Appendix C. These include the board address modifier, the input FIFO width and depth.

4.3 V1M40 DSP board

The V1M40 board is manufactured in the VME 6U 160mm standard Eurocard format (Fig.4.6). In this implementation, the V1M40 DSP board and the CRITIR board are interconnected via a MAB interface as described in Section 4.2 and assembled as one double VME-slot wide module. The V1M40 board consists of one LH9124 FFT processor, five LH9320 address generators and six memory modules (In, Out, A, B, C0 or C1) of 16K 48-bit words each . The memory modules are accessed through four different ports as follows: 'Q' port for both 'In' and 'Out'; 'A' port for memory module 'A; 'B' port for memory module 'B'; and 'C' port for both C0 and C1 memory modules. The address lines for each memory module are driven by a LH9320 address generator, with the exception of coefficient memories (C0 and C1) which share a single address generator.

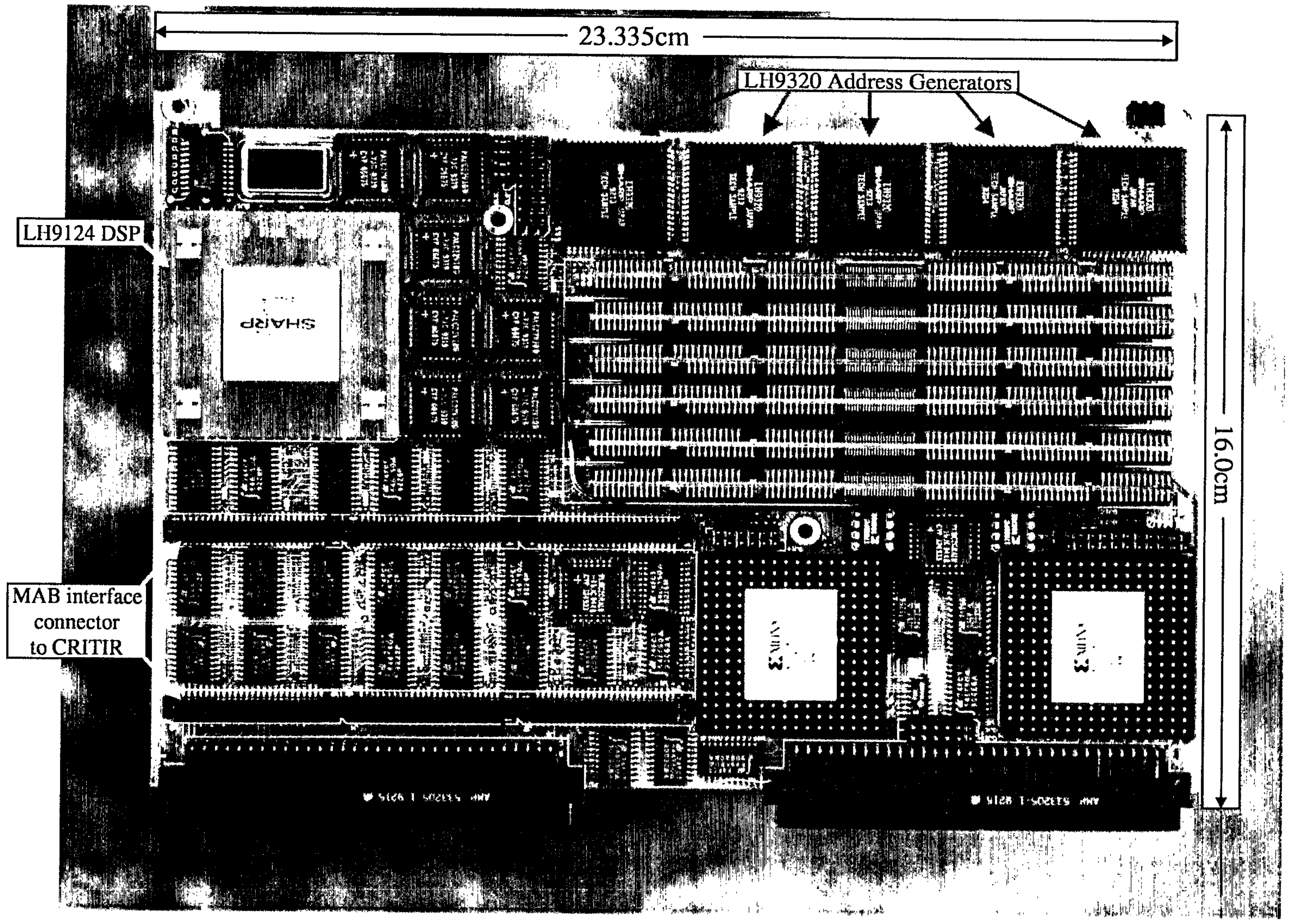
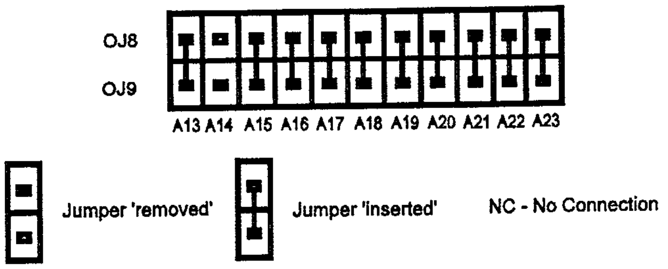


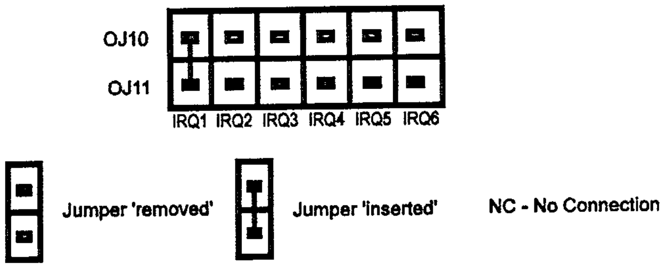
Fig.4.6. CRI V1M40 digital signal processor board (top view)

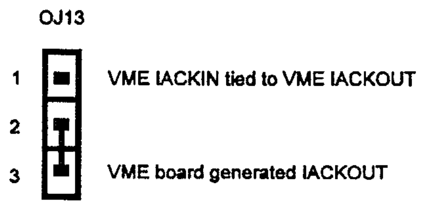
4.3.1 V1M40 DSP board hardware settings

The V1M40 board is configured via jumpers in order to select the base address location, the address modifier type, the coefficient memory paging, the VME interrupt ID and the VME interrupt level. Further details of these variables including a schematic of jumper locations on the board are found in [6]. The settings for the DREO application are as follows:

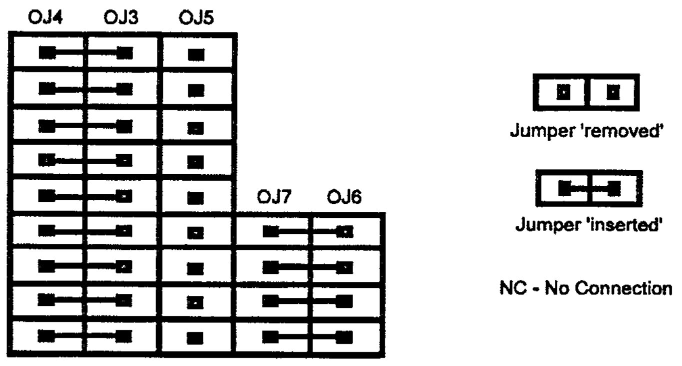
OJ8, OJ9	V1M40 Board Base address location. A jumper that is 'inserted' denotes a 'LOW' whereas one that is 'removed' denotes a 'HIGH'. The jumper settings for the DREO application are:  OJ8 OJ9 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 Jumper 'removed' Jumper 'inserted' NC - No Connection

OJ12	V1M40 board address modifier. Jumper 'inserted' indicates the board responds to VME A16 addresses. Jumper 'removed' indicates the board responds to VME A24 addresses. For the DREO application, the jumper is removed.

OJ10, OJ11	V1M40 interrupt addressing. For the DREO application, the jumper for IRQ1 is inserted as shown below:  OJ10 OJ11 IRQ1 IRQ2 IRQ3 IRQ4 IRQ5 IRQ6 Jumper 'removed' Jumper 'inserted' NC - No Connection

OJ13	VME Interrupt acknowledge. In the DREO application, the jumper has been set for a VME board generated interrupt acknowledge as follows:  OJ13 1 2 3 VME IACKIN tied to VME IACKOUT VME board generated IACKOUT

OJ14, OJ15, OJ19, OJ20	VME interrupt ID. For the DREO application, all jumpers are inserted.

OJ3, OJ4, OJ5, OJ6, OJ7	V1M40 Coefficient Paging Jumpers. The settings for the DREO application are as follows:  OJ4 OJ3 OJ5 OJ7 OJ6 Jumper 'removed' Jumper 'inserted' NC - No Connection

OJ1, OJ2	Jumpers are factory set.

OJ16, OJ17, OJ18	Jumpers are factory set.

4.3.2 V1M40 DSP board software settings

The DSP board uses the same configuration file as the NIMBLE A/D board and the CRITIR I/O interface board to initialize various parameters for proper operation. (see Appendix C). Once these parameters are read by the subroutines in the system management program, the DSP algorithm is downloaded into DSP program memory. For the DREO application, a program called 'fft512.pml' is transferred to DSP program memory.

4.4 Radisys EPC-5 system controller

The EPC-5 is a PC/AT-compatible embedded CPU module featuring:

- 33MHz Intel 486 DX processor
- 4 Mbytes of DRAM
- Keyboard interface
- COM1 & COM2
- LPT1
- Award 486 BIOS
- VMEBus interface
- EXM expansion interface

The EPC-5 has been designed for VMEBus 6U specification. It provides direct VMEBus communication to all three VMEBus address spaces (A32, A24, A16). An SVGA card and an IDE controller are added to the EPC-5 through the EXM expansion interface.

The EPC-5 is configured as a slot-1 controller. The BIOS is set up as:

Diskette Drive A	1.4M 3.5 inch
Diskette Drive B	none
Fixed Disk Drive C	AT
Fixed Disk Drive D	none
Bus Priority	Pri 0
Bus Release Method	ROR(VME)
Slot 1 Arbitration	Priority
VXI Register Base (ULA)	FE00(F8)
Slave Memory Offset	Disabled

EXM Menu

Slot	ID	OB1	OB2
0	F6	07	00
1	ED	01	00
2	FF	00	00
3	FF	00	00
4	FF	00	00
5	FF	00	00

Fixed Disk Menu

Fixed Disk Drive C	AT

Type 49 115 Mbyte 981 Cyls 6 Heads 40 Sectors Landing Zone 1023
Precompensation None

Fixed Disk Drive D	none

4.5 VME chassis

The VME Chassis is a 21-slot 500W AC powered 6U chassis. In this implementaion, the space from slot 1 to slot 3 is occupied by a floppy drive and a hard drive. A 10 VME socket backplane occupies slots 4 to 13 in which the VME boards can be inserted. All 21 slots are equipped with a slot guide. Slots 14 to 21 are not used currently but are available for backplane expansions. DC power supplies are available at +5V and ±12V.

The entire system is shown in Fig.4.7 to Fig.4.10.

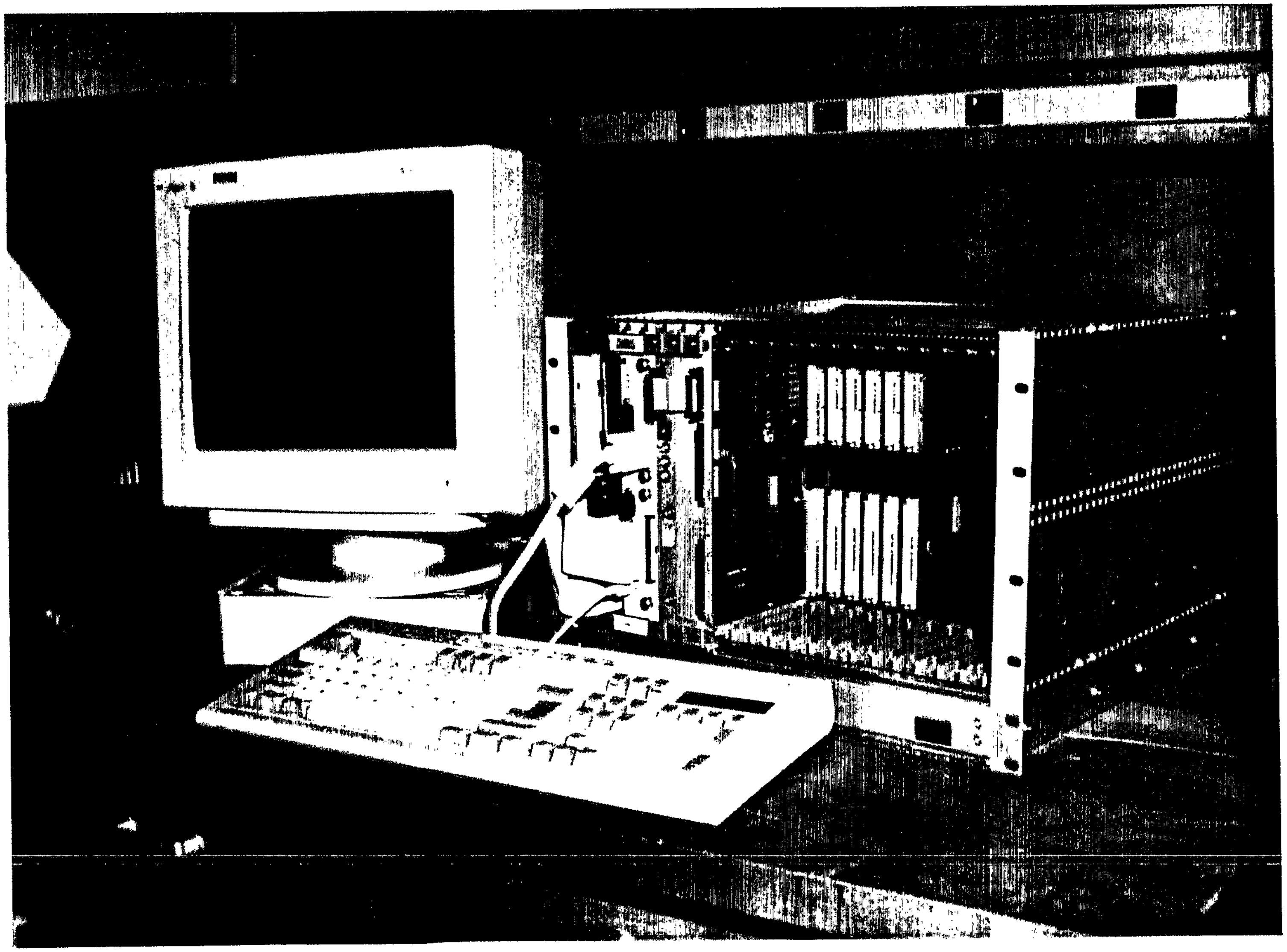


Fig.4.7. Complete implementation of FFT-based demodulator

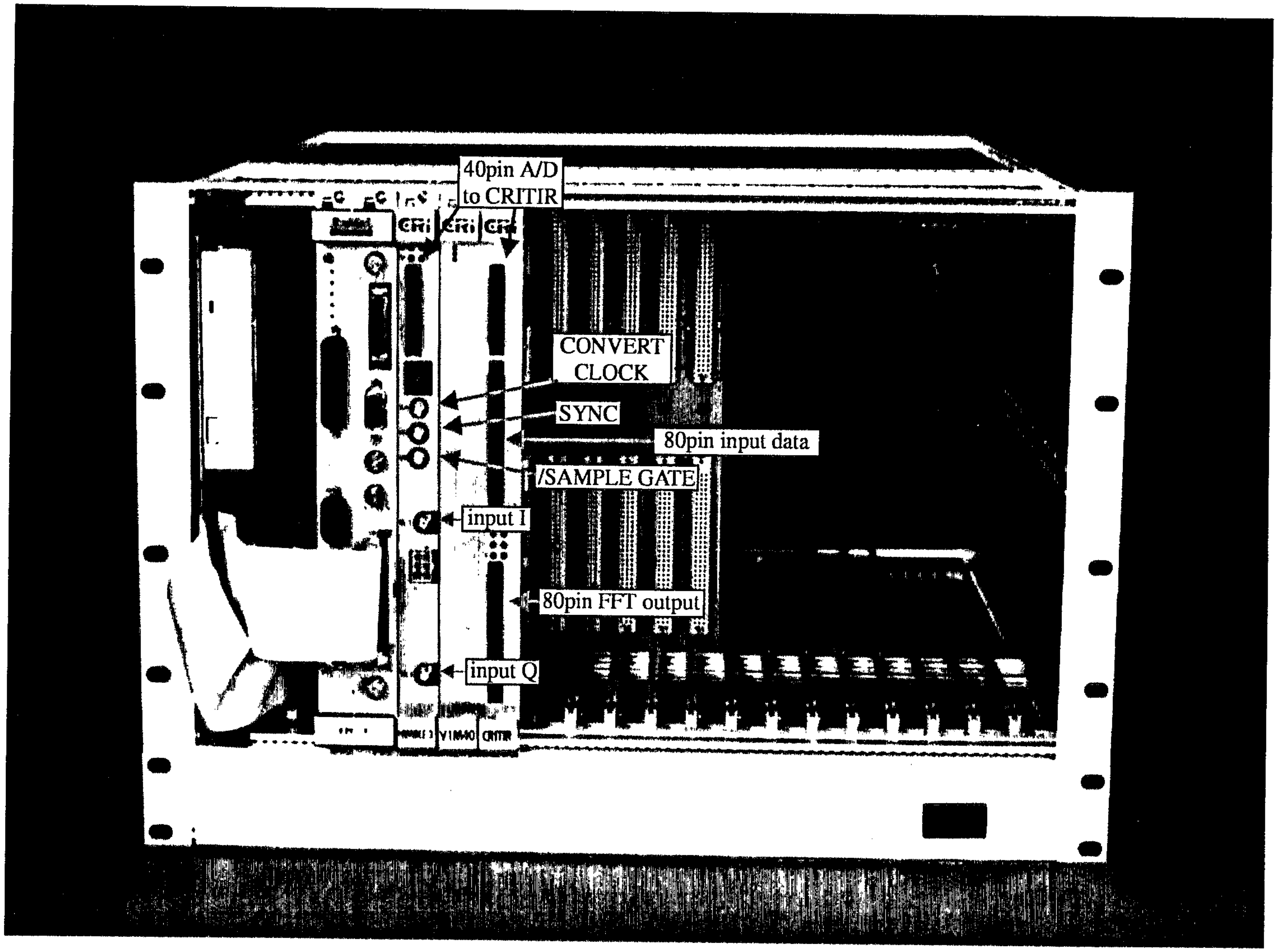


Fig.4.8. FFT-based demodulator chassis (front view)

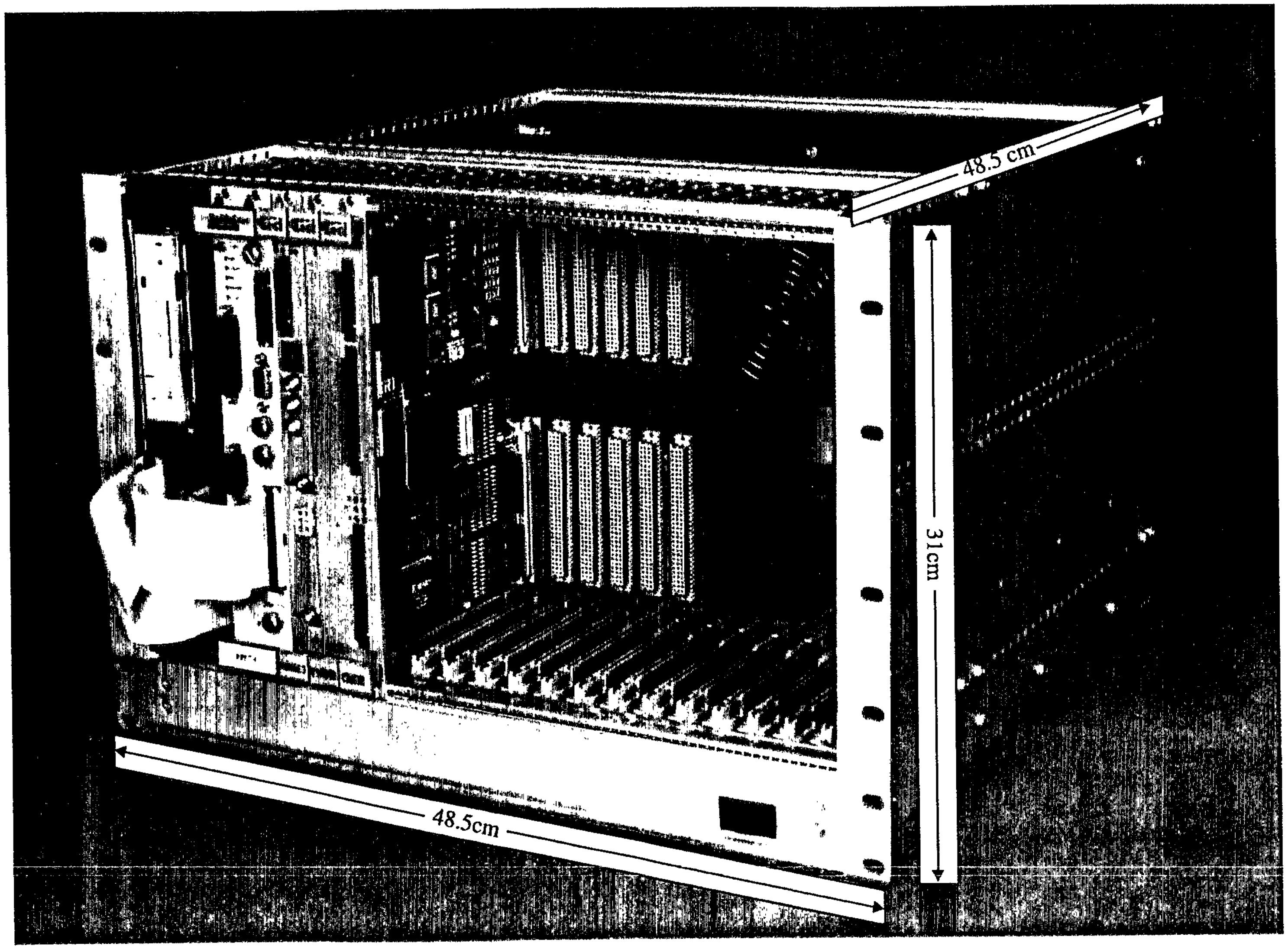


Fig.4.9. FFT-based demodulator chassis (side view)

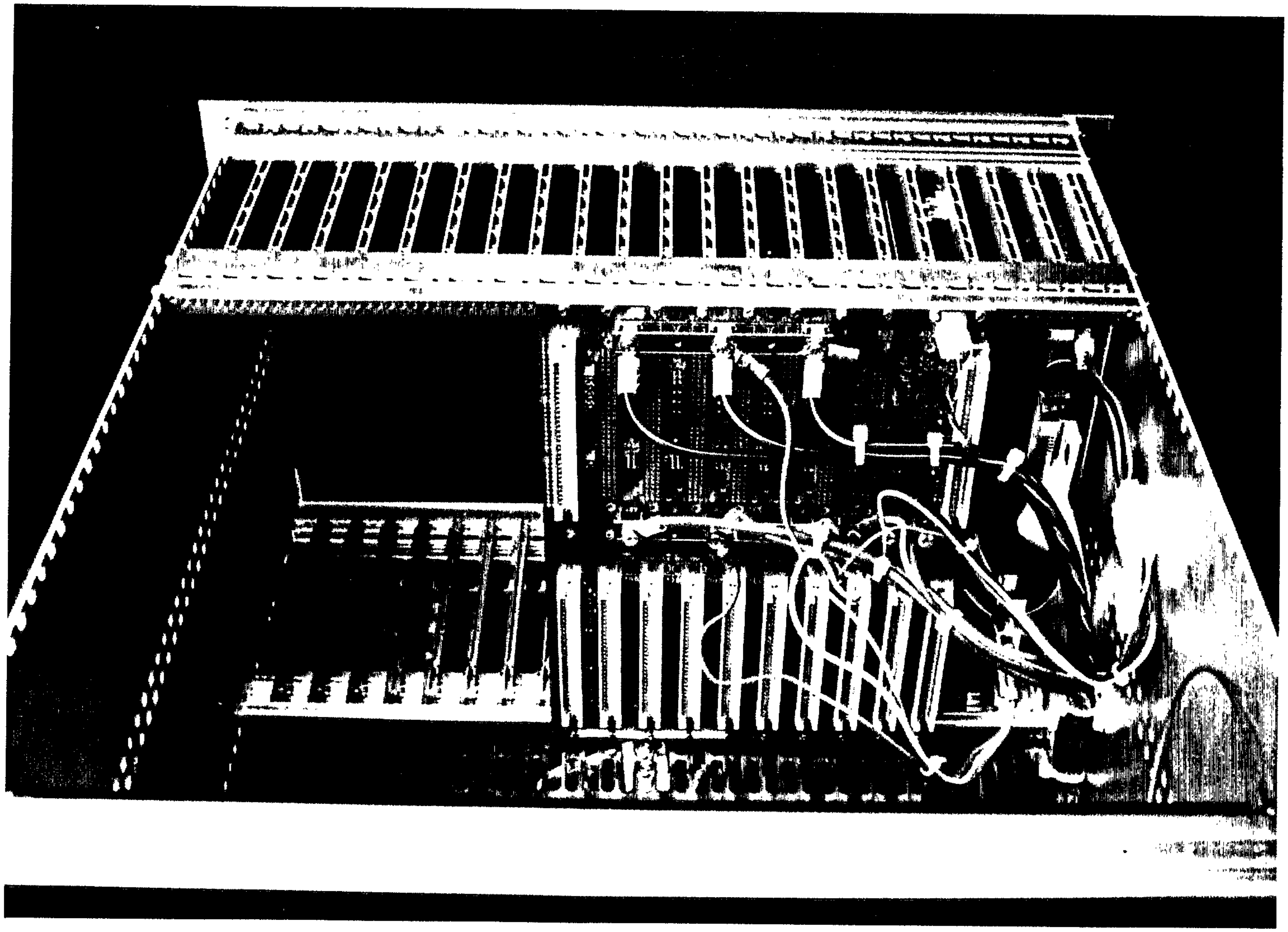
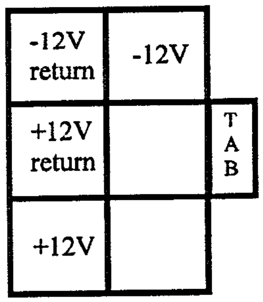


Fig.4.10. FFT-based demodulator chassis (back view)

4.6 Operator's Guide

In order to use the demodulator as an A/D-FFT processor for the DREO application, ensure that the following connections are made and commands executed.

Step	Instruction
1	The 'I' and 'Q' channels from the downconverter are connected via SMA connectors to the inputs 'CH1' and 'CH2' respectively, found on the front panel of the NIMBLE A/D board. The input impedance is 75Ω. The input signals should be within ± 1V.
2	A sampling clock signal should be connected via an SMA connector to the 'CLK' input located on the front panel of the NIMBLE A/D board. This signal should be a continuous clock with 50% duty cycle preferred. The clock input terminal requires a TTL level signal with an input impedence of 75Ω.
3	A signal to enable the collection of data by the A/D board should be provided in the '/FRAME START' input. A TTL level signal is required with 75Ω input impedance. For the DREO application, this signal is equivalent to a gated pulse to synchronize the A/D collection to the hop interval.
4	Connect the DC power for the A/D converters on the NIMBLE board to ±12V according to the following: 
5	Connect the NIMBLE A/D board output (J3) to the CRITIR input with a high density 40-pin ribbon cable.
6	Connect the CRITIR output (J5) via an 80-pin high density ribbon cable to the subsequent buffer board fabricated for the DREO application.
7	Connect the peripherals for EPC-5 including a floppy drive, a harddrive, an SVGA monitor, a keyboard, and a mouse (optional).
8	Turn on the main power switch located on the bottom front of the chassis. At DOS prompt, for front panel FFT output, enter: c:>**cd \rmc\cmr** [press return] c:\rmc\cmr:> **rmciq** [press return] For VMEBus FFT output, enter instead: c:> **cd \rmc\cmr** [press return] c:\rmc\cmr:> **rmcdb** [press return]

5.0 Making modifications to the software - Guidelines

The requirements to redirect the FFT output data to the screen or to change the number of points for the FFT were identified for the DREO application. The following sections outline the necessary steps required to carry out the modifications. All involve making changes to either the system management program or the assembly language program or both.

5.1 Redirecting the FFT output data

In this implementation, two modes of operation were provided for the proposed demodulator. The user can select whether the output of the demodulator is transferred to the front panel of the CRITIR board or whether the output data is passed to the VME bus to a file and/or displayed on the screen. The latter is especially useful for testing purposes. The principle change is the use of a different subroutine in the system management program. The modifications are already provided in the form of two system management programs called 'rmciq.c' and 'rmcdb.c' which are stored on the hard drive of the system. In 'rmciq.c', the FFT output is directed to the front panel of the CRITIR high speed I/O interface board, whereas in 'rmcdb.c', the FFT output is directed to the VMEBus for access by the VME controller.

5.2 Changing the number of FFT points

If the number of samples to be processed is changed, a number of modifications must also be made to the assembly program, the system management program and the coefficients file. The following paragraphs discuss the principle changes required.

5.2.1 Modifications to the DSP program for a different *N*

As described in Section 3.2.1.2, the computations for the FFT are divided into stages which depend on how the number of samples, *N*, is decomposed. Recall that for *N*=512 in this implementation, the result was a three stage process (Radix-2, Radix-16, Radix16). It is relatively simple to change the number of points by specifying a different LH9124 radix instruction for each of the stages. However, if the number of points is changed such that the number of stages is altered in the algorithm, the process becomes more involved . Due to the pipelined architecture of the V1M40 processing, any additional stages are implemented by extra transfers between processes and memory modules 'A' and 'B'. For example, for a four stage process, the following data flow may occur in the V1M40:

stage 1:	Radix-R1	Q to A	(from input memory to A process memory)
stage 2:	Radix-R2	A to B	(from A memory module to B memory module)
stage 3:	Radix-R3	B to A	(from B memory module to A memory module)
stage 4:	Radix-R4	A to Q	(from A memory module to output memory)

Thus, the process goes back and forth between A and B until the last stage where data is transferred to the output memory module.

It should be noted that if the number of stages is increased to compute the FFT for a different *N*, the pipeline process illustrated in Fig.3.3 is affected such that the DSP processing time is extended. The implication of the extended processing time is a delay in the transfer of the next sequence of data samples to the DSP which may result in a bottleneck at the DSP.

A change in the number of FFT points also affects the initialization of the memory registers for each of the address generators. Parameters include the ADRLENGTH, N register, and MEMSIZE, which have to be loaded with the new *N*.

5.2.2 Modifications to the system management program

The changes required for the system management program are fairly straightforward when the number of FFT points is changed. The first is to assign the new value to the variable 'NUMPTS' in the program, which is used to initialize the NIMBLE board. If the changes to the DSP program are stored as a different file, the filename of the program that is downloaded to the DSP program memory must be updated. Finally, a different FFT coefficients file may have to be transferred to DSP memory. Several files are already included on the hard drive which are ready to be used.

6.0 Summary

An FFT-based block demodulator was implemented by RMC for DREO to demodulate a 32-channel frequency-hopped 8-ary FSK signal. The system was implemented using commercial, off-the-shelf boards developed by CRI. The demodulator system is VME-based and consists of a NIMBLE A/D board, a V1M40 DSP board, and a CRITIR high speed I/O board. The latter was added to mitigate a data transfer problem posed by the VME bus bottleneck so that results of the FFT can be provided in real-time.

In the DREO implementation of the demodulator, a complex baseband signal is digitized and stored. An FFT is then performed on the digitized samples. The initial implementation of the demodulator contained further processing to select which symbols were transmitted. However, for upcoming DREO satellite uplink synchronization trials, further processing within the demodulator was disabled and the output of the system is the FFT bin components (I and Q). Subsequent processing will be performed elsewhere to detect the transmitted symbol. The output of the modified system can be directed to the 80-pin connector on the front panel or to the VME bus and displayed on the screen. The latter is useful for testing purposes.

The V1M40 DSP board uses a DIT algorithm to compute the complex FFT. Furthermore, mixed radix processing is used. The V1M40 architecture results in processing of the FFT in stages in a pipelined fashion and makes use of address generators for efficient array addressing. Currently, an assembly language program is implemented to compute a complex 512-point DFT in three stages. After the computations for the first sequence is completed, the input of subsequent data points and output of results overlap the DFT computations.

A user's guide outlining the hardware and software configuration and settings is provided. All the software routines have been included with the demodulator system to enable changes in the system to be made in the future if required. A brief description of the steps required to change the number of FFT points to be processed is given. It is noted that if the number of points for the DFT is altered, the times at which the next sequence of data samples can be transferred in and the results transferred out are affected thereby altering the pipeline process of the DSP.

REFERENCES

1. Hindson, W.D., "*Digital 32-channel FSK Demodulator: Preliminary study*", DREO Memorandum, April 1990.
2. Couture, F., and Chan, Y.T., "*Performance Analysis of a 32 MFSK Demodulator*", Royal Military College of Canada, Department of Electrical and Computer Engineering, May 1992.
3. Couture, F., Chan, Y.T., and Ferland, G., "*Implementation of a 32-channel FFT-based FSK Demodulator*", Royal Military College of Canada, Department of Electrical and Computer Engineering, March 1993.`
4. Proakis, J.G., *Digital Communications*, 2nd Edition, McGraw-Hill Inc., N.Y., 1989.
5. "*CRI NIMBLE Analog-to-Digital Converter Board User's Guide*", Catalina Research Inc., May 1995.
6. "*1M40 User's Guide*", Catalina Research Inc., August 1994.
7. "*CRITIR User's Guide*", Catalina Research Inc., June 1994.
8. Oppenheim, A.V., and Schafer, R.W., *Discrete-time Signal Processing*, Prentice-Hall, N.J., 1989.
9. "*LH9124 Digital Signal Processor User's Guide*", SHARP, 1992.
10. "*LH9320 Address Generator User's Guide*", SHARP, 1990.

LIST OF SYMBOLS/ABBREVIATIONS

A_n	Amplitude of FSK tone in channel *n*
A/D	Analog-to-digital
BPF	Bandpass filter
BW_{ch}	Channel bandwidth
BW_{32ch}	32 channel bandwidth
CRI	Catalina Research Inc
CRITIR	CRI CRITIR high speed I/O interface board
DFT	Discrete Fourier transform
DIT	Decimation-in-time
DREO	Defence Research Establishment Ottawa
DSP	Digital signal processor
EHF	Extremely high frequency
f_n	Base frequency of channel n
f_s	Sampling frequency
FFT	Fast Fourier transform
FIFO	First-in first-out memory
FSK	Frequency-shift-keying
i_n	FSK tone number for channel *n*
I	In-phase component
I/O	input/output
kHz	kiloHertz
MAB	Mezzanine adapter board
MFSK	M-ary frequency shift keying
MILSATCOM	Military satellite communications
MHz	MegaHertz
n	Channel number
N	Number of FFT points
NIMBLE	CRI NIMBLE A/D board
Q	Quadrature component
$r_I[n]$	Sequence of the in-phase component of r(t)
$r_Q[n]$	Sequence of the quadrature component of r(t)
r(t)	Received signal
R	Radix number
$R_I[k]$	Sequence of the in-phase component of the DFT of r(t)
$R_Q[k]$	Sequence of the quadrature component of the DFT of r(t)
RMC	Royal Military College
SATCOM	Satellite communications
SAW	Surface acoustic wave
$s_{32}(t)$	transmitted signal in a 32 channel system
t	Variable time
T_{symbol}	FSK symbol period

V	Volts
VME	Versamodule-Europe
V1M40	CRI V1M40 DSP board
W	Watts
W_N^{kn}	Twiddle factor in FFT computation
$x[n]$	Time sampled sequence
$X[k]$	Frequency component of DFT
μs	microsecond
Δf	Frequency separation between FSK tones
ν	Number of stages in FFT algorithm
ϕ	Phase of oscillator frequency
ω_o	Reference oscillator frequency
Ω	Ohms

APPENDIX A

APPENDIX A

Assembly Program for LH9124/LH9320

The following program performs a 512-point complex FFT with a rectangular time domain window used in the demodulator. The FFT algorithm implemented for the DREO application is a mixed-Radix algorithm consisting of the following sequence: Radix-2/Radix-16/Radix-16.

```
;Program Listing
;============

RESET A,B,C,I,O,BF:            ;Initialize the Address Generators (AGs)
                               ;Reset all AG'S, BLK FLOATING POINT and PGM
                               ;COUNTER
                               ;1FF RESETS ALL
;PROGRAM MEMORY STORED ADDRESSES = AG HAS INTERNAL PROGRAM

;PROM INITIALIZE 'O' AG        ;CLEARS MODE REGISTER, INTERNAL
;-------------------------
O_AG_PROC A=2,  DB=00:

; PROGRAM MEMORY LOAD
;----------------------------
O_AG_PROC A=1,  DB=00:         ;RBWQ  SET AG ADDRESS EQUAL TO PROG. ADD0
O_AG_PROC A=0,  DB=51:         ;SET INSTRUCTION TO MXB2161

O_AG_PROC A=1,  DB=01:         ;OUTPUT  SET AG ADDRESS EQUAL TO PROG.
O_AG_PROC A=0,  DB=72:         ;ADD1
                               ;SET INSTRUCTION TO INC
; AG LATENCY REGISTER LOAD
;--------------------------------
O_AG_PROC A=1,  DB=20:         ;RBWQ  LATENCY REGISTER ADD FOR 0 INST.
O_AG_PROC A=0,  DB=45:         ;69 LATENCY, WRITE AG

O_AG_PROC A=1,  DB=21:         ;OUTPUT  LATENCY REGISTER ADD FOR 1 INST.
O_AG_PROC A=0,  DB=80:         ;0 LATENCY, READ AG

;PCSTART REGISTER LOAD
;----------------------------

O_AG_PROC A=1, DB=A0:          ;ADDRESS OF PCSTART
O_AG_PROC A=0, DB=00:          ;START PGM ADDRESS = 0

;MODE REGISTER LOAD
;------------------------

O_AG_PROC A=1, DB=A2:          ;ADDRESS OF MODE REG.
O_AG_PROC A=0, DB=08:          ;SET TC DELAY BIT IN MODE REG
```

```
; N LOADING                        ;length of address sequence
;-----------------
O_AG_PROC A=1, DB=AC:              ;ADDRESS OF LSB OF N REG.
O_AG_PROC A=0, DB=00:              ;SET 512 ADD LENGTH
O_AG_PROC A=1, DB=AD:              ;ADDRESS OF MIDDLE OF N REG.
O_AG_PROC A=0, DB=02:              ;SET 512 ADD LENGTH
O_AG_PROC A=1, DB=AE:              ;ADDRESS OF MSB OF N REG.
O_AG_PROC A=0, DB=00:              ;SET 512 ADD LENGTH

; ADDRSTART REGISTER
;----------------------------
O_AG_PROC A=1, DB=C1:              ;ADDRESS OF LSB OF ADDRSTART REG.
O_AG_PROC A=0, DB=00:              ;SET 0 ADDRSTART
O_AG_PROC A=1, DB=C2:              ;ADDRESS OF MIDDLE OF ADDRSTART REG.
O_AG_PROC A=0, DB=00:              ;SET 0 ADDRSTART
O_AG_PROC A=1, DB=C3:              ;ADDRESS OF MSB OF ADDRSTART REG.
O_AG_PROC A=0, DB=00:              ;SET 0 ADDRSTART

; ADDRINC REGISTER
;-------------------------
O_AG_PROC A=1, DB=C7:              ;ADDRESS OF LSB OF ADDRINC REG.
O_AG_PROC A=0, DB=01:              ;SET 1 ADDRINC
O_AG_PROC A=1, DB=C8:              ;ADDRESS OF MIDDLE OF ADDRINC REG.
O_AG_PROC A=0, DB=00:              ;SET 1 ADDRINC
O_AG_PROC A=1, DB=C9:              ;ADDRESS OF MSB OF ADDRINC REG.
O_AG_PROC A=0, DB=00:              ;SET 1 ADDRINC

; ADDRLEN REGISTER
;-----------------------------
O_AG_PROC A=1, DB=C4:              ;ADDRESS OF LSB OF ADDRLEN REG.
O_AG_PROC A=0, DB=00:              ;SET 512 ADDRLEN
O_AG_PROC A=1, DB=C5:              ;ADDRESS OF MIDDLE OF ADDRLEN REG.
O_AG_PROC A=0, DB=02:              ;SET 512 ADDRLEN
O_AG_PROC A=1, DB=C6:              ;ADDRESS OF MSB OF ADDRLEN REG.
O_AG_PROC A=0, DB=00:              ;SET 512 ADDRLEN

; PC END REGISTER
;------------------------
O_AG_PROC A=1, DB=D3:              ;PROGRAM SIZE REGISTER
O_AG_PROC A=0, DB=01:

;PROGRAM MEMORY STORED ADDRESSES = AG HAS INTERNAL PROGRAM;

;PROM INITIALIZE 'I' AG
;--------------------------
I_AG_PROC A=2, DB=00:              ;CLEARS MODE REGISTER, INTERNAL

; PROGRAM MEMORY LOAD
;-------------------------------
```

```
I_AG_PROC A=1,  DB=00:          ;INPUT  SET AG ADDRESS EQUAL TO PROG. ADD0
I_AG_PROC A=0,  DB=72:          ;SET INSTRUCTION TO INC
I_AG_PROC A=1,  DB=01:          ;RQWA  SET AG ADDRESS EQUAL TO PROG. ADD1
I_AG_PROC A=0,  DB=71:          ;SET INSTRUCTION TO RBF0

; AG LATENCY REGISTER LOAD
;---------------------------------------
I_AG_PROC A=1,  DB=20:          ;INPUT  LATENCY REGISTER ADD FOR 0
I_AG_PROC A=0,  DB=00:          ;0 LATENCY, WRITE AG INST.
I_AG_PROC A=1,  DB=21:          ;RQWA  LATENCY REGISTER ADD FOR 1 INST.
I_AG_PROC A=0,  DB=80:          ;0 LATENCY, READ AG

;PCSTART REGISTER LOAD
;----------------------------------

I_AG_PROC A=1, DB=A0:           ;ADDRESS OF PCSTART
I_AG_PROC A=0, DB=00:           ;START PGM ADDRESS = 0

;MODE REGISTER LOAD
;------------------------------

I_AG_PROC A=1, DB=A2:           ;ADDRESS OF MODE REG.
I_AG_PROC A=0, DB=08:           ;SET TC DELAY BIT IN MODE REG

; N LOADING
;-------------
I_AG_PROC A=1, DB=AC:           ;ADDRESS OF LSB OF N REG.
I_AG_PROC A=0, DB=00:           ;SET 512 ADD LENGTH
I_AG_PROC A=1, DB=AD:           ;ADDRESS OF MIDDLE OF N REG.
I_AG_PROC A=0, DB=02:           ;SET 512 ADD LENGTH
I_AG_PROC A=1, DB=AE:           ;ADDRESS OF MSB OF N REG.
I_AG_PROC A=0, DB=00:           ;SET 512 ADD LENGTH

; ADDRSTART REGISTER
;---------------------------
I_AG_PROC A=1, DB=C1:           ;ADDRESS OF LSB OF ADDRSTART REG.
I_AG_PROC A=0, DB=00:           ;SET 0 ADDRSTART
I_AG_PROC A=1, DB=C2:           ;ADDRESS OF MIDDLE OF ADDRSTART REG.
I_AG_PROC A=0, DB=00:           ;SET 0 ADDRSTART
I_AG_PROC A=1, DB=C3:           ;ADDRESS OF MSB OF ADDRSTART REG.
I_AG_PROC A=0, DB=00:           ;SET 0 ADDRSTART

; ADDRINC REGISTER
;-----------------------
I_AG_PROC A=1, DB=C7:           ;ADDRESS OF LSB OF ADDRINC REG.
I_AG_PROC A=0, DB=01:           ;SET 1 ADDRINC
I_AG_PROC A=1, DB=C8:           ;ADDRESS OF MIDDLE OF ADDRINC REG.
I_AG_PROC A=0, DB=00:           ;SET 1 ADDRINC
I_AG_PROC A=1, DB=C9:           ;ADDRESS OF MSB OF ADDRINC REG.
I_AG_PROC A=0, DB=00:           ;SET 1 ADDRINC
```

```
; ADDRLEN REGISTER
;--------------------------
I_AG_PROC A=1, DB=C4:          ;ADDRESS OF LSB OF ADDRLEN REG.
I_AG_PROC A=0, DB=00:          ;SET 512 ADDRLEN
I_AG_PROC A=1, DB=C5:          ;ADDRESS OF MIDDLE OF ADDRLEN REG.
I_AG_PROC A=0, DB=02:          ;SET 512 ADDRLEN
I_AG_PROC A=1, DB=C6:          ;ADDRESS OF MSB OF ADDRLEN REG.
I_AG_PROC A=0, DB=00:          ;SET 512 ADDRLEN

; DIGIT REVERSE REGISTER
;---------------------------------
I_AG_PROC A=1, DB=D5:          ;ADDRESS OF LSB OF DIGREV REG.
I_AG_PROC A=0, DB=FF:          ;SET FF DIG LOW
I_AG_PROC A=1, DB=D6:          ;ADDRESS OF MIDDLE OF DIGREV REG.
I_AG_PROC A=0, DB=00:          ;SET FF DIG MID
I_AG_PROC A=1, DB=D7:          ;ADDRESS OF MSB OF DIGREV REG.
I_AG_PROC A=0, DB=00:          ;SET FF DIG HIGH

; PC END REGISTER
;------------------------
I_AG_PROC A=1, DB=D3:                  ;PROGRAM SIZE REGISTER
I_AG_PROC A=0, DB=01:

;PROGRAM MEMORY STORED ADDRESSES = AG HAS INTERNAL PROGRAM

;PROM INITIALIZE 'A' AG
;----------------------------
A_AG_PROC A=2,  DB=00:                 ;CLEARS MODE REGISTER, INTERNAL

; PROGRAM MEMORY LOAD
;--------------------------------------
A_AG_PROC A=1,  DB=00:         ;RQWA  SET AG ADDRESS EQUAL TO PROG. ADD0
A_AG_PROC A=0,  DB=01:         ;SET INSTRUCTION TO BF20
A_AG_PROC A=1,  DB=01:         ;RAWB  SET AG ADDRESS EQUAL TO PROG. ADD1
A_AG_PROC A=0,  DB=50:         ;SET INSTRUCTION TO MXB2160

; AG LATENCY REGISTER LOAD
;-------------------------------------------
A_AG_PROC A=1,  DB=20:         ;RQWA  LATENCY REGISTER ADD FOR 0 INST.
A_AG_PROC A=0,  DB=13:         ;19 LATENCY, WRITE AG
A_AG_PROC A=1,  DB=21:         ;RAWB  LATENCY REGISTER ADD FOR 1 INST.
A_AG_PROC A=0,  DB=80:         ;0 LATENCY, READ AG

;PCSTART REGISTER LOAD
;-----------------------------------

A_AG_PROC A=1, DB=A0:          ;ADDRESS OF PCSTART
A_AG_PROC A=0, DB=00:          ;START PGM ADDRESS = 0

;MODE REGISTER LOAD
;--------------------------------
```

```
A_AG_PROC A=1, DB=A2:          ;ADDRESS OF MODE REG.
A_AG_PROC A=0, DB=08:          ;SET TC DELAY BIT IN MODE REG

; N LOADING
;--------------
A_AG_PROC A=1, DB=AC:          ;ADDRESS OF LSB OF N REG.
A_AG_PROC A=0, DB=00:          ;SET 512 ADD LENGTH
A_AG_PROC A=1, DB=AD:          ;ADDRESS OF MIDDLE OF N REG.
A_AG_PROC A=0, DB=02:          ;SET 512 ADD LENGTH
A_AG_PROC A=1, DB=AE:          ;ADDRESS OF MSB OF N REG.
A_AG_PROC A=0, DB=00:          ;SET 512 ADD LENGTH

; PC END REGISTER
;-----------------------
A_AG_PROC A=1, DB=D3:          ;PROGRAM SIZE REGISTER
A_AG_PROC A=0, DB=01:

;PROGRAM MEMORY STORED ADDRESSES = AG HAS INTERNAL PROGRAM;

;PROM INITIALIZE 'B' AG
;----------------------------
B_AG_PROC A=2,  DB=00:         ;CLEARS MODE REGISTER, INTERNAL

; PROGRAM MEMORY LOAD
;--------------------------------
B_AG_PROC A=1,  DB=00:         ;RAWB  SET AG ADDRESS EQUAL TO PROG. ADD0
B_AG_PROC A=0,  DB=50:         ;SET INSTRUCTION TO MXB2160
B_AG_PROC A=1,  DB=01:         ;RBWQ  SET AG ADDRESS EQUAL TO PROG. ADD1
B_AG_PROC A=0,  DB=51:         ;SET INSTRUCTION TO MXB2161

; AG LATENCY REGISTER LOAD
;-------------------------------------
B_AG_PROC A=1,  DB=20:         ;RAWB  LATENCY REGISTER ADD FOR 0 INST.
B_AG_PROC A=0,  DB=44:         ;68 LATENCY, WRITE AG
B_AG_PROC A=1,  DB=21:         ;RBWQ  LATENCY REGISTER ADD FOR 1 INST.
B_AG_PROC A=0,  DB=80:         ;0 LATENCY, READ AG

;PCSTART REGISTER LOAD
;-----------------------------------

B_AG_PROC A=1, DB=A0:          ;ADDRESS OF PCSTART
B_AG_PROC A=0, DB=00:          ;START PGM ADDRESS = 0

;MODE REGISTER LOAD
;-------------------------------

B_AG_PROC A=1, DB=A2:          ;ADDRESS OF MODE REG.
B_AG_PROC A=0, DB=08:          ;SET TC DELAY BIT IN MODE REG
```

```
; N LOADING
;--------------
B_AG_PROC A=1, DB=AC:           ;ADDRESS OF LSB OF N REG.
B_AG_PROC A=0, DB=00:           ;SET 512 ADD LENGTH
B_AG_PROC A=1, DB=AD:           ;ADDRESS OF MIDDLE OF N REG.
B_AG_PROC A=0, DB=02:           ;SET 512 ADD LENGTH
B_AG_PROC A=1, DB=AE:           ;ADDRESS OF MSB OF N REG.
B_AG_PROC A=0, DB=00:           ;SET 512 ADD LENGTH

; PC END REGISTER
;---------------------------
B_AG_PROC A=1, DB=D3:           ;PROGRAM SIZE REGISTER
B_AG_PROC A=0, DB=01:

;PROGRAM MEMORY STORED ADDRESSES = AG HAS INTERNAL PROGRAM

;PROM INITIALIZE 'C' AG
;----------------------------
C_AG_PROC A=2,  DB=00:          ;CLEARS MODE REGISTER, INTERNAL

; PROGRAM MEMORY LOAD
;-------------------------------------
C_AG_PROC A=1,  DB=00:          ;RQWA  SET AG ADDRESS EQUAL TO PROG. ADD0
C_AG_PROC A=0,  DB=24:          ;SET INSTRUCTION TO TF20
C_AG_PROC A=1,  DB=01:          ;RAWB  SET AG ADDRESS EQUAL TO PROG. ADD1
C_AG_PROC A=0,  DB=65:          ;SET INSTRUCTION TO MXT2160
C_AG_PROC A=1,  DB=02:          ;RBWQ  SET AG ADDRESS EQUAL TO PROG. ADD2
C_AG_PROC A=0,  DB=66:          ;SET INSTRUCTION TO MXT2161

; AG LATENCY REGISTER LOAD
;-----------------------------------------
C_AG_PROC A=1,  DB=20:          ;RQWA  LATENCY REGISTER ADD FOR 0 INST.
C_AG_PROC A=0,  DB=81:          ;1 LATENCY, READ AG
C_AG_PROC A=1,  DB=21:          ;RAWB  LATENCY REGISTER ADD FOR 1 INST.
C_AG_PROC A=0,  DB=90:          ;16 LATENCY, READ AG
C_AG_PROC A=1,  DB=22:          ;RBWQ  LATENCY REGISTER ADD FOR 2 INST.
C_AG_PROC A=0,  DB=90:          ;16 LATENCY, READ AG

;PCSTART REGISTER LOAD
;-----------------------------------

C_AG_PROC A=1, DB=A0:           ;ADDRESS OF PCSTART
C_AG_PROC A=0, DB=00:           ;START PGM ADDRESS = 0

;MODE REGISTER LOAD
;-------------------------------

C_AG_PROC A=1, DB=A2:           ;ADDRESS OF MODE REG.
C_AG_PROC A=0, DB=08:           ;SET TC DELAY BIT IN MODE REG

; N LOADING
```

```
;-----------------
C_AG_PROC A=1, DB=AC:          ;ADDRESS OF LSB OF N REG.
C_AG_PROC A=0, DB=00:          ;SET 512 ADD LENGTH
C_AG_PROC A=1, DB=AD:          ;ADDRESS OF MIDDLE OF N REG.
C_AG_PROC A=0, DB=02:          ;SET 512 ADD LENGTH
C_AG_PROC A=1, DB=AE:          ;ADDRESS OF MSB OF N REG.
C_AG_PROC A=0, DB=00:          ;SET 512 ADD LENGTH

; MEM SIZE REGISTER
;--------------------------
C_AG_PROC A=1, DB=D8:          ;ADDRESS OF LSB OF MEMSIZE REG.
C_AG_PROC A=0, DB=00:          ;SET 512 MEM SIZE
C_AG_PROC A=1, DB=D9:          ;ADDRESS OF MIDDLE OF MEMSIZE REG.
C_AG_PROC A=0, DB=02:          ;SET 512 MEM SIZE
C_AG_PROC A=1, DB=DA:          ;ADDRESS OF MSB OF MEMSIZE REG.
C_AG_PROC A=0, DB=00:          ;SET 512 MEM SIZE

; PC END REGISTER
;-----------------------
C_AG_PROC A=1, DB=D3:          ;PROGRAM SIZE REGISTER
C_AG_PROC A=0, DB=02:

DSP_ENA RAWB, MOVD:            ;SET DF FOR EXT I/O
START I:                       ;Start I prior to LOOP
WAIT I:

JMP_LOC 0:                     ;begin loop

COEF_SEL C1,PAGE=0:            ;SELECT window MEMORY
DATA_CTL PGM_SCALE = 4:

DSP_ENA RQWA, BFLY2:           ;DF=1, FC=01 INSTRUCTION
DSP_ENB:                       ;WRITE ALL 0 TO DCX, CSFI
START I A C :
WAIT I A C :

DATA_CTL PGM_SCALE:

DSP_ENA RAWB, BFLY16:          ;DF=1, FC=01 INSTRUCTION
DSP_ENB:                       ;WRITE ALL 0 TO DCX, CSFI
START I A B C :
WAIT A B C :

WAIT O:                        ;wait for last output pass to be completed

DSP_ENA RBWQ, BFLY16:          ;DF=1, FC=01 INSTRUCTION
DSP_ENB:                       ;WRITE ALL 0 TO DCX, CSFI
START O B C :
WAIT O B C :

RESET BF:                      ;latch bfp to output & reset
```

```
JMP 0:                          ;jump to begin of loop
                                ;3 instruction pipeline delay
DSP_ENA RAWB, MOVD:             ;SET DF FOR EXT I/O
START O:                        ;start output pass
WAIT I:                         ;wait for new input
```

In addition to the FFT program, a coefficient file (and a weighting file if there is a time domain window) is (are) needed for Fourier Transform. A sample listing of a coefficient file (stored as 'cd512.iql') for a 512-point FFT is shown below. The first column is the real part of the coefficient which is one cycle of cosine wave in 512 samples and, the second column is the imaginary part, a cycle of sinusoid wave. Values are given as 24-bit fixed point integers. Other coefficient files and time domain windowing files are available and are stored on the hard drive of the demodulator system.

```
;Sample of 'cd512.iql'

7FFFFF 000000                   ;Coefficient 0
7FFD88 FE6DE3
7FF621 FCDBD6
7FE9CB FB49E7
7FD887 F9B827
7FC255 F826A5
7FA736 F69570
7F872B F50498
7F6236 F3742D
7F3857 F1E43E
7F0991 F054D9                   ;Coefficient 10
7ED5E5 EEC610
7E9D55 ED37F0
7E5FE4 EBAA8A
7E1D93 EA1DEC
7DD666 E89227
.
.
.
```

APPENDIX B

APPENDIX B

System Management Program

The system management program for the A/D-FFT demodulator system is written in C. There are two system management programs available called 'rmciq.c' and 'rmcdb.c'. The first program directs the FFT output to the front panel of the CRITIR. The second program displays the FFT results on the screen. The purpose of both programs is to initialize the hardware, download the executable program to the target board and start the system execution. The program is executed on the VMEBus controller, Radisys EPC-5.

The following program ('rmciq.c') directs the FFT Output to Critir Front Panel

```
/*******************************************************************************/
/*                RMC IQ MODE PROGRAM                          */
/*******************************************************************************/
# include <math.h>
# include <stdlib.h>
# include <stdio.h>
# include "oldhost.h"
# include "host.h"
# include "1m40.h"
# include "critir.h"
# include "nim.h"
# include "readcfg.h"
# include  "cri_err.h"

void main()
{
  long          fpacked;
  short         c_addr_modifier, v_addr_modifier, n_addr_modifier,error;
  long          n_addr_cfg, c_addr_cfg, v_addr_cfg;
  unsigned      Swap;
  long          numpts;

  long          nim_rd_mode, nimmode;
  long          nim_wr_mode, nim_capt_mode;

  char *system_cfg_filename = "system.cfg";
  SystemConfigStruct sconfig;

  nim_rd_mode = 3;                          /*Rd_mode(2:0)=3 dual channel read
                                            continuous*/
  nim_wr_mode = 7;                          /*Wr_mode(3:0)=7 dual channel write
                                            FIFO 1 and FIFO 2*/
  nim_capt_mode = 1;
  fpacked = 0;
  Swap = 0;

  numpts = 512;                             /* nimble frame capture length */
```

```
/* read the system config file */

if (ReadSystemConfig(system_cfg_filename, &sconfig, stderr))

  {
    fprintf(stderr, "fatal errors in reading system config file\n");
    exit (1);
  }

if (M40Initialise(&sconfig, 1))
  {
    fprintf(stderr, "%s\n", CriReportError());
    PrintSystemConfig(stderr, &sconfig);
    exit(1);
  }

/* Initialise the critir boards */

if(CritInitialise(&sconfig, 1))
  {
    fprintf(stderr, "%s\n", CriReportError());
    PrintSystemConfig(stderr, &sconfig);
    exit(1);
  }

if(NIMInitialise(&sconfig, 1))
  {
    fprintf(stderr, "%s\n", CriReportError());
    PrintSystemConfig(stderr, &sconfig);
    exit(1);
  }

c_addr_cfg = sconfig.critir_address[0];
c_addr_modifier = sconfig.critir_addrmod[0];
v_addr_cfg = sconfig.v1m40_address[0];
v_addr_modifier = sconfig.v1m40_addrmod[0];
n_addr_cfg = sconfig.nimble_address[0];
n_addr_modifier = sconfig.nimble_addrmod[0];

printf( "v_addr_cfg: %08lx, v_addr_modifier is: %04hx\n", v_addr_cfg, v_addr_modifier );
printf( "n_addr_cfg: %08lx, n_addr_modifier is: %04hx\n", n_addr_cfg, n_addr_modifier );
printf( "c_addr_cfg: %08lx, c_addr_modifier is: %04hx\n", c_addr_cfg, c_addr_modifier );

HostInit( v_addr_cfg, v_addr_modifier );          /* set up to talk to the 1m40 */

wr_file_pgm( "fft512.pml", 0 );                   /* download the control program to
                                                  v1m40 */

wr_file_mem(C1, 512, 512,0,1,"cd512.iql",0);      /* write coeficient files to v1m50 c
                                                  process memories */
                                                  /* write twiddle factors to  C1 memory */
```

```
board_reset();                                  /* reset the v1m40 */

HostInit( n_addr_cfg, n_addr_modifier );        /* set up to talk to the nimble */
nim_board_reset();

nim_frame_start(numpts);                        /* load nimble frame counter with the
                                                frame size */

nimmode = nim_wr_mode + (nim_rd_mode << 4);     /* set nimble up for dual channel */
nim_set_mode((short)nimmode);                   /* write (fifo 1 & 2) read continuous */

HostInit( c_addr_cfg, c_addr_modifier );        /* set up to talk to the critir */

CritBdReset();                                  /* reset the critir board */

CritSelFrontPanelAsDest();                      /* select front panel output */

CritSelIQOut();                                 /* select i/q output format */

CritSelAllOut();                                /* select all data (no thresholding ) */

CritSelADInput();                               /* select nimble a/d as input for critir */

CritSelBurstIn();                               /* select burst mode processing  (input
                                                fifos reset each frame ) */

CritSelIQUnPack();                              /* select unpacked input format to v1m40
                                                (i/q) */

CritSelIQUnSwap();                              /* do not swap i/q channels */

CritOutReset();                                 /* reset critir output fifos */

CritInpReset();                                 /* reset critir input fifos */

HostInit( v_addr_cfg, v_addr_modifier );        /* set up to talk to the v1m50 */

cntrl_start(0);                                 /* start the v1m50 */

HostInit( n_addr_cfg, n_addr_modifier );        /* set up to talk to the nimble */

nim_capture_mode(nim_capt_mode);                /* set up nimble to frame capture mode
                                                (this will start processing) */

exit( 0 );
}
```

APPENDIX B

The following program ('rmcdb.c') directs the FFT Output to VMEBus for Display

```
/***********************************************************************/
/*                 RMC  VME READ                                     */
/***********************************************************************/


# include <math.h>
# include <stdlib.h>
# include <stdio.h>
# include <graph.h>
# include "oldhost.h"
# include "host.h"
# include "1m40.h"
# include "critir.h"
# include "nim.h"
# include "readcfg.h"
#include  "cri_err.h"

long     inp_array[1026];

float    xdata1[4096];
float    xdata2[4096];

void main()
{
  long          fpacked;
  short         c_addr_modifier,v_addr_modifier,n_addr_modifier,error;
  long          n_addr_cfg,c_addr_cfg,v_addr_cfg;
  unsigned      Swap;
  long          numpts, num_points_read,flag_iq,index;
  long          tmag;
  short         tphase;

  long          nim_rd_mode,nimmode;
  long          nim_wr_mode, nim_capt_mode;

  int           i, j, loop;
  long          buf;

  char *system_cfg_filename = "system.cfg";
  SystemConfigStruct sconfig;

  loop=10;
  nim_rd_mode = 3;
  nim_wr_mode = 7;
  nim_capt_mode = 1;
  fpacked = 0;
  Swap = 0;

  numpts = 512;                                /* nimble frame capture length */
```

```
/* read the system config file */

if (ReadSystemConfig(system_cfg_filename, &sconfig, stderr))
  {
    fprintf(stderr, "fatal errors in reading system config file\n");
    exit (1);
  }

if(M40Initialise(&sconfig, 1))
  {
    fprintf(stderr, "%s\n", CriReportError());
    PrintSystemConfig(stderr, &sconfig);
    exit(1);
  }

/* Initialise the critir boards */

if(CritInitialise(&sconfig, 1))
  {
    fprintf(stderr, "%s\n", CriReportError());
    PrintSystemConfig(stderr, &sconfig);
    exit(1);
  }

if(NIMInitialise(&sconfig, 1))
  {
    fprintf(stderr, "%s\n", CriReportError());
    PrintSystemConfig(stderr, &sconfig);
    exit(1);
  }

c_addr_cfg = sconfig.critir_address[0];
c_addr_modifier = sconfig.critir_addrmod[0];
v_addr_cfg = sconfig.v1m40_address[0];
v_addr_modifier = sconfig.v1m40_addrmod[0];
n_addr_cfg = sconfig.nimble_address[0];
n_addr_modifier = sconfig.nimble_addrmod[0];

printf( "v_addr_cfg: %08lx, v_addr_modifier is: %04hx\n", v_addr_cfg, v_addr_modifier );
printf( "n_addr_cfg: %08lx, n_addr_modifier is: %04hx\n", n_addr_cfg, n_addr_modifier );
printf( "c_addr_cfg: %08lx, c_addr_modifier is: %04hx\n", c_addr_cfg, c_addr_modifier );

HostInit( v_addr_cfg, v_addr_modifier );              /* set up to talk to the 1m40 */

wr_file_pgm( "fft512.pml", 0 );                       /* download the control program to
                                                      v1m40 */
wr_file_mem(C1, 512, 512,0,1,"cd512.iql",0);          /* write coeficient files to v1m50 c
                                                      process memories */
                                                      /* write twiddle factors to  C1 memory */

wr_file_mem(C2, 512, 512,0,1,"win.iql",0);            /* write win coeficient file to c2 page 0 */
                                                      /*  coeficient are all 0x7FFFFF 000000
```

```
                                              (mult by 1, 0  = no window) */

board_reset();                                /* reset the v1m40 */

HostInit( n_addr_cfg, n_addr_modifier );      /* set up to talk to the nimble */

nim_frame_start(numpts);                      /* load nimble frame counter with the */
nim_board_reset();                            /* frame size */

nimmode = nim_wr_mode + (nim_rd_mode << 4);   /* set nimble up for dual channel write
                                              (fifo 1 & 2) read continuous */

HostInit( c_addr_cfg, c_addr_modifier );      /* set up to talk to the critir */

HostInit( c_addr_cfg, c_addr_modifier );      /* initialize reg to zero */
CritWrReg( CRIT_CTL_REG1, 0x0);

CritBdReset();                                /* reset the critir board */

CritSelVMEAsDest();                           /* select VMEBus output */
CritSelIQOut();                               /* select i/q output format */

CritSelAllOut();                              /* select all data (no thresholding ) */

/* CritSelThreshOut(); (used for test purposes only)
   CritSetThreshOut(10,5); (used for test purposes
   only*/

CritSelADInput();                             /* select nimble a/d as input for critir */

CritSelBurstIn();                             /* select burst mode processing  (input
                                              fifos reset each frame ) */

CritSelIQUnPack();                            /* select unpacked input format to v1m40
                                              (i/q) */

CritSelIQUnSwap();                            /* do not swap i/q channels */

CritOutReset();                               /* reset critir output fifos */

CritInpReset();                               /* reset critir input fifos */

HostInit( v_addr_cfg, v_addr_modifier );      /* set up to talk to the v1m50 */
HostInit( v_addr_cfg, v_addr_modifier );
cntrl_start( 0 );
HostInit( n_addr_cfg, n_addr_modifier );
nim_set_mode((short)nimmode);

nim_capture_mode(nim_capt_mode);              /* enable frame start sampling */
flag_iq = 1;
numpts = 512;

_setvideomode(_VRES16COLOR);

HostInit( c_addr_cfg, c_addr_modifier );                /* set up to talk to the critir */

while(1) {
   CritRdNeFifo((numpts+1), inp_array );

   _clearscreen(_GCLEARSCREEN);
   printf("\n                          Complex 512 points FFT");
   _setcolor(7);
   _moveto(50,40);
   _lineto(563,40);
   _lineto(563,470);
   _lineto(50,470);
   _lineto(50,40);
   _setcolor(11);
   _moveto(50,470);

for (i=0; i<2*513; i+=2)
{
    if (flag_iq==0 )
       index = inp_array[i]&0xFFFF;
    if (flag_iq==1 )
       index = ((inp_array[i]&0x000000FF)<<8) + (inp_array[i+1]&0x000000FF);
    if (index > (numpts-1))
    {
       printf("Error: index is out of range %ld\n", index);
       exit(1);
    }
    if(flag_iq == 0 )
    {
         tmag  = inp_array[i] >> 16;
         xdata1[i] = tmag;
         tphase = inp_array[i+1]&0x0000FFFF;
         xdata2[i] = tphase;
         xdata2[i] = 180.0 * (xdata2[i] / 32768 );
    }
    if (flag_iq == 1 )
    {
         xdata1[i] = (inp_array[i]/256);
         xdata2[i] = (inp_array[i+1]/256);
    }
    _lineto(i/2+50, 470-(int)(sqrt((float)(xdata1[i]*xdata1[i]+xdata2[i]*xdata2[i]))/30000.0));

    }
  }
  exit( 0 );
}
```

APPENDIX C

System Configuration file

The system configuration file contains the hardware configuration information required by the CRI board utility/driver software. This file should be placed in the directory where the executable files, i.e. 'rmciq.exe' or 'rmcdb.exe', are stored. With current hardware settings, the system configuration file (called "system.cfg") is as follows:

```
; SYSTEM CONFIGURATION FILE
; VAR NAME              BOARD #  VALUE

HOST_TYPE                   RADISYS

V1M40_BOARDS              1              NUMBER OF V1M40 BOARDS
V1M40_ADDRESS             0    0X4000    VME ADDRESS FOR V1M40
V1M40_ADDRMOD             0    A24       VME ADDRESS MODIFIER FOR V1M40
V1M40_MEMSIZE             0    0X4000    V1M40 A,B,I,O MEMORY SIZE
V1M40_C1_SIZE             0    0X4000    V1M40 C1 MEMORY SIZE
V1M40_C2_SIZE             0    0X4000    V1M40 C2 MEMORY SIZE
V1M40_COEF_PAGES          0    1         V1M40 NUMBER OF COEF PAGES
V1M40_COEF_PAGE_SIZE      0    0X4000    V1M40 COEF PAGE SIZE
V1M40_COEF_PAGE_STEP      0    1         V1M40 COEF PAGE STEP

NIMBLE_BOARDS             1              NUMBER OF NIMBLE BOARDS
NIMBLE_ADDRESS            0    0X0000    VME ADDRESS FOR NIMBLE
NIMBLE_ADDRMOD            0    A16       VME ADDRESS MODIFIER FOR NIMBLE
NIMBLE_FIFO_DEPTH         0    0X100     NIMBLE FIFO DEPTH
NIMBLE_NUM_CHANNELS       0    2         2 NIMBLE A/D CHANNELS
NIMBLE_OFFSET_BINARY      0    FALSE     NIMBLE OFFSET BINARY INPUT
NIMBLE_TWOS_COMPLEMENT 0       TRUE      NIMBLE 2S COMPLEMENT INPUT

CRITIR_BOARDS             1              NUMBER OF CRITIR BOARDS
CRITIR_ADDRESS            0    0X0000    VME ADDRESS FOR CRITIR
CRITIR_ADDRMOD            0    A24       VME ADDRESS MODIFIER FOR CRITIR
CRITIR_INPUT_FIFO_WIDTH   0    16        CRITIR INPUT FIFO WIDTH
CRITIR_INPUT_FIFO_DEPTH   0    256       CRITIR INPUT FIFO DEPTH
CRITIR_INPUT_OVERLAP      0    false     HARDWARE DEPENDENT FUNCTION
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. **ORIGINATOR** (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence Research Establishment Ottawa Ottawa, Ontario K1A 0Z4	2. **SECURITY CLASSIFICATION** (overall security classification of the document including special warning terms if applicable) **UNCLASSIFIED**

3. **TITLE** (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Multichannel M-ary Frequency-shift-keying Block Demodulator Implementation (U)

4. **AUTHORS** (Last name, first name, middle initial)

Tom, Caroline and Meng, Zaiqing

5. **DATE OF PUBLICATION** (month and year of publication of document) December 1996	6a. **NO. OF PAGES** (total containing information. Include Annexes, Appendices, etc.) 68	6b. **NO. OF REFS** (total cited in document) 10

7. **DESCRIPTIVE NOTES** (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

DREO Report

8. **SPONSORING ACTIVITY** (the name of the department project office or laboratory sponsoring the research and development. Include the address.

SST, Defence Research Establishment Ottawa
Ottawa, Ontario, K1A 0Z4

9a. **PROJECT OR GRANT NO.** (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant) 5ca11	9b. **CONTRACT NO.** (if appropriate, the applicable number under which the document was written)
10a. **ORIGINATOR'S DOCUMENT NUMBER** (the official document number by which the document is identified by the originating activity. This number must be unique to this document.) DREO REPORT 1307	10b. **OTHER DOCUMENT NOS.** (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. **DOCUMENT AVAILABILITY** (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. **DOCUMENT ANNOUNCEMENT** (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). however, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Unlimited Announcement

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13. **ABSTRACT** (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

The Military Satellite Communications (MILSATCOM) group at Defence Research Establishment Ottawa (DREO) is performing in-house satellite communications uplink synchronization experiments. A critical part of the experiment is the block demodulation of the multi-channel frequency-hopped waveform. This report analyses the Fast Fourier transform (FFT)-based demodulator implemented by Royal Military College (RMC) Kingston. This system is capable of demodulating a 32 channel, 8-ary frequency-shift-keying (FSK) waveform. The implementation consists of analog-to-digital (A/D) conversion, FFT computation, and high speed input/output (I/O) interface processes using commercial off-the-shelf boards. A description of each of the boards is presented. The operational data flow of the system is examined. For the software implementation, a decimation-in-time approach is used in the digital signal processing (DSP) board to compute the discrete Fourier transform (DFT). A brief description of decimation-in-time algorithms is included. Key components of the assembly language program are outlined. A user's guide detailing the setup requirements for proper operation of the system is included with those required for the DREO application. Steps required for modification to the software are briefly discussed.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

digital demodulator
Fast Fourier Transform
FSK

S00767