Our progress as a nation can be no swifter than our progress in education ... The human mind is our most valuable resource.

DIGITAL SIGNAL PROCESSING
USING THE
TMS32010

CARDIFF INSTITUTE OF HIGHER EDUCATION
FACULTY OF DESIGN & TECHNOLOGY

A. J. SHEPHERD

JULY 1992

This dissertation is submitted to the Cardiff Institute of Higher Education in fulfilment of the requirements for a research degree.
DECLARATION.

This dissertation is the product of my own investigation except where I have acknowledged my indebtedness to other sources.

The investigative material contained within has not been submitted for another award and is not being currently submitted for another award.

Candidate. . . . A. J. Shepherd.

Date. . . . July 1992

Research Supervisors
MR G GORST CIHE
MR H MORRIS CIHE
ABSTRACT.

The mathematical theory of the Fast Fourier Transform is developed from the basic Fourier series, leading to the application of the FFT to the display of a 128 line FFT on a BBC and IBM computer using a commercially available hardware and software package employing a Texas TMS320 digital signal processor.

Project Aims

The two aims of the report are:

1. To produce a coherent analysis of the theory associated with the Fast Fourier Transform (FFT) and

2. To write, develop and implement a 128 line FFT using the C3M development unit together with either a BBC or IBM P.C.

These aims are repeated on page 140 and show where they are related to the project chapters.
DIGITAL SIGNAL PROCESSING
USING THE
TMS32010

An investigation into the use of the C3M development unit to perform a 128 line Fast Fourier Transform using the Texas Instruments TMS32010 DSP chip.

Also to produce a coherent set of notes outlining the FFT theory and to produce using this theory a 128 line FFT on a BBC and an IB personal computer from the C3M unit.

A. J. SHEPHERD

Faculty of Design & Technology

## CONTENTS

**CHAPTER I.**

**INTRODUCTION.**

1.1 Raison D'Etre.  
1.2 Background Considerations.  
1.3 Historical Review.

**CHAPTER II.**

**FREQUENCY. REPRESENTATION.**

**CHAPTER III.**

**THE THEORETICAL BACKGROUND.**

3.1 Basic Fourier Series.  
3.2 Exponential Form of Fourier Series.  
3.2-1. Fourier Series Coefficients, C_k.  
3.2-2. Gibbs Phenomena.  
3.3 The Fourier Integral - or Fourier Transform.  
3.3-1. Graphical Representation of the Fourier Transform.  
3.4 Signal Power and Power Spectral Density.  
3.5 The Laplace Transform Relationship.  
3.6 The Discrete Fourier Transform (DFT)  
3.6-1. Practical Implications of the DFT.

**CHAPTER IV**

**BASIC PROGRAMMING OF THE DFT AND THE FFT.**

A. Calculation of the DFT.  
A1. Programming the DFT. In Basic  
B. Calculation of the Fast Fourier Transform.  
B1. Programming the FFT. In Basic.

**CHAPTER V.**

**THE DIGITAL SIGNAL PROCESSING CHIP (DSP).**

A. The Digital Signal Processor (DSP).  
B. Digital Signal Processor Applications.  
C. Digital Signal Processing Equipment.  
D. Types of Digital Signal Processing Chips.  
D1. The TMS32010 Digital Signal Processing Chip.

**CHAPTER VI.**

The C3M DSP UNIT.  
The C3M Digital Signal Processing Unit.
CHAPTER VII

PROGRAMMING

a. Introduction.

b. PROGRAM SETS for the BBC and the IBM PC.
   - PROGRAM SETS 1 & 2. Data Capture & Display Programs using the BBC and the IBM PC.
   - PROGRAM SET 3. Length 8 DIF FFT for the IBM PC.
   - PROGRAM SET 4. Length 128 DIF FFT for the IBM PC. 20 kHz Sample Rate.
   - PROGRAM SET 5. Length 128 DIF FFT for the IBM PC. 5 kHz Sample Rate.

c. PROGRAM SET 1
   - 1.1 Set 1 BBC Time display.
   - 1.2 Typical Set 1 display.
   - 1.3 Loading & Running the Set 1 Programs.

c. PROGRAM SET 2
   - 2.1 Set 2 IBM PC Time display.
   - 2.2 Typical Set 2 display.
   - 2.3 Loading & Running the Set 2 Programs.

c. PROGRAM SET 3
   - 3.1 Set 3 IBM PC Length 8 DIF FFT.
   - 3.5 Typical Set 3 Display.
   - 3.5 Loading & Running the Set 3 Programs.

c. PROGRAM SET 4
   - 4.1 Set 4 Length 128 FFT.
   - 4.4 Typical Set 4 displays.
   - 4.4 Loading & Running the Set 4 Programs.

CHAPTER VIII

SUMMARY FINDINGS.

General Conclusions.
Programming Conclusions.
Theoretical Conclusions.
Summary.

Appendix A

DATA ON TMS32010 DSP CHIP.

Appendix B

PROGRAM LISTINGS.

Set 1 Program listings.
The TMS32010 Program TRANBBC for the C3M board.
The BBC Basic capture & display program C3MGRA.

Set 2. Program listings.
The TMS32010 Program C3MTIME.SRC 170
The 8086/80286 Assembly program IBMTIME.COM. 173
GWBASIC display program PLOTTME.BAS 176

Set 3 Program listings. 179
The TMS32010 Length 8 DIF FFT program. 179
The IBM Assembly program FFT8IBM. 186
The GWBASIC FFT Display program. 189

Set 4 Program listings. 192
The TMS32010 length 128 FFT Program. 192
.c. Additions to the IBM PC transfer program 201
The GWBASIC FFT Display program. 202

BIBLIOGRAPHY. 207
CHAPTER I.

INTRODUCTION.

1.1 Raison D'Être.

This text is the result of three years work to unravel the complexities of the Fast Fourier Transform (FFT) and then to apply the results — using the C3M Development System to produce a real time FFT analyser capable of producing an 128 line frequency spectrum.

It was not the intention here to break any new ground but to collect, collate and clarify the theory for the FFT and to use this together with existing software — suitably modified — to program the C3M unit to perform an FFT and to display this on a host computer.

The C3M Development System used as basis for this report contains a Texas Instrument TMS32010 Digital Signal Processing chip, and was designed to be interfaced to an IBM PC — or a BBC microcomputer. It was designed and marketed by Ultra Digital Systems — the commercial wing of Liverpool University, and intended to be used to introduce students to the construction of Digital Filters.

Software was supplied with the unit to perform various filter algorithms but no provision was made for its use as an FFT analyser.

1.2 Background Considerations.

This report arises from the need for a Real Time Frequency Analyser on the HNC and HND courses in the Mechanical and Production engineering section of the Faculty of Industry, and from a desire to collate the large amount of theory associated with both the Discrete and the Fast Fourier
Transforms (DFT and FFT) and to present this theory in a unified manner.

Typical applications for a frequency analyser in the mechanical field are:

1. machine condition monitoring,
2. noise control,
3. vibration analysis,
4. signal filtering,
and 5. control system analysis.

The frequency analyser would be used to break down the above time dependant signals into their frequency components and display the amplitude of these components against frequency on a monitor screen.

There are a number of commercially available frequency analysers that would be suitable - for the areas listed above, but their cost (£10k to £15k) would be difficult to justify in the present economic climate.

A low cost alternative - reviewed here, is to buy in a Digital Signal Processing (DSP) development unit such as the C3M from Ultra Digital Systems. This unit would then need to be interfaced to a microcomputer and software written to enable it to perform a frequency analysis using the Fast Fourier Transform.

A Digital Signal Processor - or DSP, is a relatively new integrated circuit device that acts as a very fast microcomputer. It has the added facility of being able to perform large numbers of multiplications very quickly. This feature is required if a train of signals is to undergo a series of mathematical operations - such as in a Fast Fourier Transform (FFT), in order to calculate the frequency components of the signal.
To achieve the above tasks requires a knowledge of the mathematical techniques behind the Fast Fourier Transform together with a knowledge of programming both the Digital Signal Processor and the IBM PC - as well as the BBC computer.

A newcomer to the area of frequency analysis quickly finds that some authors deal well with one or two topics but sketchily with others. This was particularly so in my case and as a result a large amount of time was spent on trying to achieve a clear understanding of this complex area. Whilst there are some excellent texts on this subject (Ref. 1-3), to move from one textbook to another is not straightforward as each author has their own style and frequently use a different notation.

The primary function of this report is therefore to bring together this theory and present it in an unified manner, so that others - as well as myself - will find it easier to understand and apply.

The second part of this report is to study the possibility of implementing the above on the C3M unit and being able to use the results for the areas listed.

The material contained in the report could also form the basis of a module in an HND or HNC on Digital Signal Processing.

1.3 Historical Review.

The pioneer in the field of frequency representation was Fourier (1768-1830) (5), who in 1822 formulated the theory that any periodic event - no matter how complex, can be looked upon as a series of harmonically related simple periodic events. Fourier however would have been aware of the work by Sir Isaac Newton (6) 1671, who introduced the
word "spectrum" after observing that light split up into its constituent colours when passed through a glass prism. But it was Fourier who laid the theoretical foundations for the investigation of periodic events. Typical of these in the late 19th century was the prediction of tidal heights against the moons phases by Sir William Thompson (7) 1878, which helped give us the tide tables we use today.

More recently the technique has been used in predictive maintenance schemes, where the condition monitoring of a machine's "health" is determined using frequency analysis plots taken over a period of time (8). Also following the failure of gearboxes in helicopters servicing the North Sea rigs attention has been focused on the use of vibration analysis to predict gearbox faults well in advance of failure (9).

Following the introduction of the computer, Jim Cooley and John Tukey (10) in 1965 laid the foundations for the production of computer programs capable of calculating the frequency spectrum of a signal using the Fast Fourier Transform. Their work was based on an earlier work by Tukey (10) who in 1958 laid the groundwork for modern spectrum analysis and introduced terms such as aliasing and windowing, etc.

More recent texts include the work by Beauchamp and Yuen (11) 1979, on the use of digital methods for signal analysis which includes frequency response techniques for image enhancement, the excellent work by Burrus and Parks (12) 1985, who showed that both the Discrete Fourier Transform (DFT) as well as the Fast Fourier Transform (FFT) can be adapted for use on small computers, and the work by Andrew Batemen and Warren Yates (13), 1988, on Digital Signal Processing which includes a section on programming the Texas TMS32010 DSP chip to perform both the DFT and the FFT.

The introduction in 1982 by Texas Instruments of the worlds first purpose built Digital Signal Processing chip - the
TMS32010, offered solutions to tasks which previously required expensive multi chip solutions. The Digital Signal Processing chip is basically a fast microprocessor dedicated to array processing. It has a Harvard architecture that allows it to perform parallel execution of several tasks at once against the Von Neumann architecture found in ordinary microprocessors such as the 8086 that perform tasks sequentially. The TMS32010 DSP performs a multiply task in 1 clock cycle (0.160 microseconds) against the 118 clock cycles (11.8 microseconds) found in the 8086 microprocessor, further the 8086 is a 16 bit device whilst the TMS32010 is a 32 bit device. The TMS32010 DSP device is quoted as handling 6.4 MIPS (million instructions per second).

Applications for DSP’s cover telecommunications, automobiles, image processing, voice recognition and synthesis, signal recovery and enhanced screen displays. Speech synthesis and speech recognition was used by World of Wonders in 1989 in a child’s doll, the doll had a CMOS Texas TMS320C10 DSP chip built in which allowed the child and doll to hold a simple conversation.

The future of the DSP is rosy with new and faster devices continually being offered. Recently Analog Devices have come out with the ADSP-21msp50, a DSP that has onboard 16 bit digital to analog (D/A) and analog to digital (A/D) converters, has a 75 nano second clock time (13.3 MIPS) and is capable of working in the range (0 to 400 KHz). The device being aimed at digital mobile radio but other applications will no doubt follow which utilise the built in converters.

A criticism of the DSP has been its narrow bandwidth and dynamic range. The ADSP-21msp50 above has moved the bandwidth out to 400 KHz, whilst another recent introduction Motorola’s DSP96002 - with its floating point arithmetic, has greatly expanded its dynamic range.

The DSP96002 has a 27 MHz clock which can perform 13.5 MIPS
together with three 32 bit Arithmetic Logic Units (ALU's) which give it a 96 bit capacity.

A recent paper by McNally, Canny and Woods (15) details a fast array processor - designed as an Infinite Impulse Response filter, that can perform 40 Million samples per second. The increase in speed has been achieved by reversing the mathematical computation process to deal with the most significant operations first instead of the least significant operations first as is normally the case. If used in a DSP this technique should greatly extend the applications of the DSP to that of TV, radar and radio systems.
CHAPTER II.

FREQUENCY REPRESENTATION.

The frequency representation - or spectrum - of a periodic signal can yield information that is not obvious from an analysis of the raw time dependant data.

Fourier showed that any time varying signal such as noise or vibration can be considered to consist of a large number of pure tones - or sine waves, each of differing frequency. These sine waves are usually harmonics of the basic or fundamental frequency and can be either odd or even.

Consider the frequency spectrum (FFT diagram) shown in Fig.(2.1) of a low frequency audio signal:

This diagram was obtained using Hypersignal software to analyse a low frequency audio signal. The signal was chopped - or sampled, 256 times over a 32 millisecond period - the samples having been taken one every 0.125 millisecond. This digitised audio signal is then 'processed' using the Fast Fourier Transform method during which the sine waves that go to make up this signal can be determined. Each of
these sine waves - or harmonics, will have an amplitude or coefficient associated with it and possibly a phase relationship.

An inspection of Fig. (2.1) shows the amplitude variations of these sine wave harmonics against frequency. Only 127 lines - or harmonics, are shown, which is only half of the total 256 harmonics produced by the FFT and hence only half of the total frequency range of 8 KHz. This is due to a quirk in the FFT which produces a mirror image of the FFT about the centre frequency for single channel signals. (see pages 56 and 57).

If all the sine waves represented by these harmonics and their phases where added together then the original time dependant signal could be reconstructed. In reality as the series is limited to a finite length not all the harmonics are calculated, as a result the reconstructed wave is not perfect.

It follows then that all signals that exist in the time domain also exist in the frequency domain. That is they are two views of the same signal. This relationship is shown in Fig. (2.2) below;-

**Fig. 2.2 Time/Frequency relationship.**
The Time / Frequency relationship in Figure (2.2) shows how a time dependant signal can be sampled over a period of time (ts). A Fast Fourier Transform (FFT) performed on the time sampled data and the frequency response obtained - both magnitude and phase. This process could be reversed using the Inverse Fourier Transform (IFT) to generate the time signal from the frequency response - magnitude and phase.

If magnitude frequency plots are taken over a period of time a "Waterfall" plot can be constructed as shown in Fig. (2.3). Here a speech signal has been sampled once every ten milliseconds over a 320 millisecond period and the resulting frequency magnitude plots have been placed in order of sampling. This plot is a demonstration plot supplied with the Hypersignal software package and captured using the Capture program supplied with Microsoft's Word5. Waterfall diagrams are extremely useful in predicting the movement of resonant peaks during a machine run up or run down, or to monitor the vibration levels in a machine over a period of time in order to predict bearing or gear failure etc. in a Predictive Maintenance scheme.

The increasing cost of industrial plant and machinery
coupled with the need to avoid costly breakdowns during production runs requires that the 'health' of the plant be regularly monitored. The Digital Signal Processor in conjunction with a lap top or desk top Personal Computer now enables a comprehensive system of computer based condition monitoring to be achieved for a fraction of the cost it would have taken five years ago.

Typical prices are around £2k for the DSP based expansion card to fit in the back of the PC (Loughborough Sound Images PCS/320C25) and around £500 for an FFT based software package (Hypersignal Plus).

Such a package should enable bearing and shaft vibration levels to be monitored on a regular basis, using either a hand held vibration monitor capable of performing a 64 line FFT on the spot, by recording the vibration levels using a tape recorder for later analysis, or sending the vibration signals back to a control centre using screened or twisted wire leads.
This part covers the theoretical frequency response analysis and follows from the basic Fourier series up to the Fast Fourier Transform (FFT).

The Fourier development covered is listed below:

1. Basic Fourier Series
2. Exponential form of the Fourier Series
3. Fourier Transform or Integral and the Laplace Transform
4. Discrete Fourier Transform (DFT) and
5. Fast Fourier Transform (FFT).

All the basic stages have been included so that the development of the theory up to the FFT butterfly can be fairly easily followed.

It should be pointed out that it is not necessary to follow the steps in great detail in order to apply the FFT to a practical situation, but an appreciation of the basic theory is required if the results obtained are to be meaningfully interpreted.

3.1 Basic Fourier Series.

It has already been stated that time dependant signals—such as vibration and noise—can be represented as a series of sinusoidal terms, each term being a multiple of the basic frequency \(\omega_0\). However before the signals can be represented in this form they must be periodic and assumed to be continuous—i.e. exist for all time. In addition the signals must conform to what are known as the Dirichelet conditions \(\star\). One of these conditions is that
the magnitude of the function being considered is finite over a complete period and the second is that the signal must be single valued at all points over the period. Fortunately most time dependant signals of practical use comply with these conditions.

The basic Fourier series representation for a time dependant periodic signal having a fundamental - or lowest, frequency \( \omega_0 \) and harmonics \( k\omega_0 \) is:

\[
f(t) = A_0 + \sum_{k=1}^{\infty} \left[ A_k \cos(k\omega_0 t) + B_k \sin(k\omega_0 t) \right] \quad \ldots (1)
\]

the dc component \( A_0 \) is:

\[
A_0 = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \, dt
\]

and the Fourier coefficients \( A_k \) and \( B_k \) are found from:

\[
A_k = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos(k\omega_0 t) \, dt
\]

and .. \( B_k \) by:

\[
B_k = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin(k\omega_0 t) \, dt
\]
The fundamental frequency $\omega_0$ rad/sec is related to the periodic time $T$ by:

$$\omega_0 = 2\pi f_0 = \frac{2\pi}{T}$$

The term $k$ gives the harmonic of the fundamental frequency $\omega_0$.

The series is usually restricted to a finite number of terms $\mathbf{n}$ and then becomes:

$$f(t) = A_0 + \sum_{k=1}^{K=n} \left[ A_k \cos(k\omega_0 t) + B_k \sin(k\omega_0 t) \right] \ldots (2)$$

As both sine and cosine terms may be present at each harmonic the resulting series may possess both magnitude and phase at each frequency term - or harmonic.

The magnitude, term $M_k$ is found from:

$$M_k = \left[ A_k^2 + B_k^2 \right]^{1/2}$$

and the phase angle $\phi$ (normally a lag) from:

$$\phi = \tan^{-1}(\frac{B_k}{A_k})$$

giving an alternative expression for the Fourier series as:

$$f(t) = A_0 + \sum_{k=1}^{K=n} \left[ A_k \cos(k\omega_0 t + \phi_k) \right] \ldots (3)$$
The basic Fourier series can be represented in exponential form using the relationships:

\[
\text{Cos}(k\omega_0 t) = \frac{1}{2} \left[ e^{+jk\omega_0 t} + e^{-jk\omega_0 t} \right]
\]

and:

\[
\text{Sin}(k\omega_0 t) = \frac{1}{2} j \left[ e^{+jk\omega_0 t} - e^{-jk\omega_0 t} \right]
\]

substituting these two equations into equation (1) and rearranging gives;
\[ f(t) = A_0 + \sum_{K=1}^{K=\infty} \left[ \frac{1}{2}[A_k - jB_k]e^{i\kappa \omega_0 t} + \frac{1}{2}[A_k + jB_k]e^{-i\kappa \omega_0 t} \right] \]

if \( C_k = \frac{1}{2}[A_k - jB_k] \) and \( C_{-k} = \frac{1}{2}[A_k + jB_k] \)

this gives:

\[ f(t) = A_0 + \sum_{K=1}^{K=\infty} \left[ C_k e^{i\kappa \omega_0 t} + C_{-k} e^{-i\kappa \omega_0 t} \right] \] \quad \ldots \quad (1a)

Replacing the coefficients \( A_k \) and \( B_k \) by their original terms gives:

from \( C_k = \frac{1}{2} [A_k - jB_k] \)

\[ 2C_k = 2/T \int_0^T f(t) \cos(k\omega_0 t) \, dt - j 2/T \int_0^T f(t) \sin(k\omega_0 t) \, dt \]

and:

\[ C_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \left[ \cos(k\omega_0 t) - j \sin(k\omega_0 t) \right] \, dt \]
giving:

\[ C_k = \frac{1}{T} \int_{-\pi/2}^{\pi/2} f(t)e^{-j\omega_ot}dt \]

Similarly for \( C_{-k} \) we get:

\[ C_{-k} = \frac{1}{T} \int_{-\pi/2}^{\pi/2} f(t)e^{j\omega_ot}dt \]

The format for \( C_k \) is then seen to be similar to that for \( C_{-k} \) with \( k \) replaced by \(-k\).

Note: In some texts \( C_{-k} \) is written as \( C^* \).

The Fourier coefficient \( C_k \) then is found from:

\[ C_k = \frac{1}{T} \int_{-\pi/2}^{\pi/2} f(t)e^{-j\omega_ot}dt \ldots \quad (4) \]

where \( T \) is the fundamental signal period.

Note also that when \( k=0 \) that:

\[ C_0 = \frac{1}{T} \int_{-\pi/2}^{\pi/2} f(t)dt = A_0 \]

This leads to the exponential form of the Fourier series:
Digital Signal Processing. [Chapter 3]. page 19

\[ f(t) = \sum_{K=-\infty}^{K=\infty} C_k e^{j\omega_0 t} \quad \ldots \quad (3) \]

Note that the summation now extends from minus infinity to plus infinity in order to accommodate both \( C_k \) and its conjugate \( C_{-k} \).

In practice the limits of equation (3) are limited to a finite range \( k = \pm n \), ie a range of \( 2n+1 \) terms.

\[ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots
frequency component that equals \(+k\omega_0\) or \((2\pi f_0 t)\). Writing equation (4) again and substituting \([ E.\sin(k\omega_0 t)]\) for \(f(t)\) and noting that \(E.\sin(k\omega_0 t) = \text{Im} \{E.e^{(-i k\omega_0 t)}\}\) gives:

\[
C_k = \text{Im}\left\{ \frac{1}{T} \int_{-T/2}^{T/2} E.e^{(i k\omega_0 t)} e^{(-i k\omega_0 t)} \text{dt} \right\}
\]

the exponential term \(e^{(i k\omega_0 t)}\) rotates clockwise and cancels out the \(e^{(-i k\omega_0 t)}\) unit vector which rotates anti-clockwise leaving a finite integral ;-)

\[
C_k = \frac{1}{T} \int_{-T/2}^{T/2} E.\text{dt} = E
\]

Hence the term at the frequency \((+k\omega_0)\) can be represented by a vector of magnitude \((E)\) and having a phase angle given by the value that \(e^{(i k\omega_0 t)}\) had at \(t=0\).

Equation (4) then extracts from the signal \(f(t)\) the vectors - or coefficients, it contains for each of the frequencies \((+k\omega_0)\) for \(k=0\) to +/- infinity. Furthermore the signal \(f(t)\) can be recovered by summing these vectors from the positions they had at \(t=0\).

A diagram showing the relationships between magnitude and phase of the vectors together with their relationship to the real and imaginary components of the vectors is shown in Fig.3.1.
Each of the Fourier coefficient $C_k$ is then seen to be a complex valued spectrum component of the signal $f(t)$, having an amplitude and phase (or equivalent real and imaginary component) associated with it. The phase being the value each of the coefficients had at time $t=0$.

If a single valued real time signal is being processed - as will normally be the case, then each of the positive coefficients - or vectors, will have a matching negative coefficient having an equal magnitude but of opposite phase. Hence the imaginary components of the conjugate vectors will cancel each other out leaving only the real component.

The examples shown in the following section 3.2-1 show this for both an even and an odd single valued function.
3.2-1. Fourier Series Coefficients, $C_k$.

Once the Fourier coefficients are known the spectrum of the signal can be sketched. In addition these coefficients are widely used in the design of digital filters.

**eg.1** An even valued function ($f(t) = f(-t)$).

Consider the even valued square wave time function shown below:

![Fig. (3.2) Even Valued Function.](image)

Analytically this is given as:

$$f(t) = h \text{ for } -T/4 < t < +T/4$$

$$f(t) = 0 \text{ for } -T/2 < t < -T/4$$

and $f(t) = 0 \text{ for } T/4 < t < T/2$

Fundamental frequency $1/T$.

$$C_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j\omega kT} dt$$
ie: -

\[ C_k T = \int_{-\tau/4}^{\tau/4} (0).dt + \int_{-\tau/4}^{\tau/4} (h).e^{-j(k\omega_o)t}.dt + \int_{\tau/4}^{\tau/2} (0).dt \]

giving:

\[ C_k T = \int_{-\tau/4}^{\tau/4} (h).e^{-j(k\omega_o)t}.dt \]

\[ C_k T / h = \left[ \begin{array}{c} e^{-jk\omega_o \frac{\tau}{4}} \\ jk\omega_o \end{array} \right] \]

\[ C_k = \frac{h}{jk\omega_o T} \left[ e^{-jk\omega_o \frac{T}{4}} - e^{jk\omega_o \frac{T}{4}} \right] \]

applying the equation:

\[ 2 \sin(A) = \frac{e^{jA} - e^{-jA}}{j} \]

gives:

\[ C_k = \frac{h}{k\omega_o T} \cdot 2 \sin(k\omega_o T/4) \]

or:

\[ C_k = \frac{h/2}{k\omega_o T} \cdot \sin(k\omega_o T/4) \]

note: replacing \( k\omega_o T/4 \) by \( x \) gives:

\[ C_k = \frac{h \cdot \sin(x)}{2} \frac{1}{x} \]

This \((\sin x)/x\) type equation has the characteristic shape
shown in Fig. (3.3).

![Diagram](image)

Fig. (3.3) \( \frac{\sin x}{x} \) type plot.

Note that an even time function has produced an even frequency function - one that is wholly real, having no imaginary components.

eg.2 **An odd time function** \( f(t) = -f(-t) \).

Consider the odd valued square wave time function shown below:

![Diagram](image)

Fig. (3.4) Odd Valued function.

Period \( T \) seconds.

\[
f(t) = \begin{cases} 
  -h & -T/2 < t < 0 \\
  h & 0 < t < T/2 
\end{cases}
\]
as:-

\[ C_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-jkins} dt \]

we get for this case:-

\[ C_k = \frac{1}{T} \int_{-T/2}^{T/2} (-h) e^{-jkins} dt + \frac{1}{T} \int_{0}^{T/2} (h) e^{-jkins} dt \]

\[ C_k = -\frac{h}{jk\omega T} \left[ 2 - \left( e^{j\omega T/2} + e^{-j\omega T/2} \right) \right] \]

since \( 1/j = -j \) and as \( \cos(A) = \frac{e^{jA} + e^{-jA}}{2} \)

Then :-

\[ C_k = \frac{jh}{k\omega T/2} \left[ 1 - \cos(k\omega T/2) \right] \]

This response is shown below :-

[Graph of the spectrum of an odd function]

**Fig. (3.5) Spectrum of Odd function.**

Here an odd time function has produced an odd spectrum and one that is wholly imaginary.
Note that each coefficient \((k\omega_0)\) is a harmonic of the fundamental frequency \((\omega_0)\) and could be written as \(\omega\). Also each harmonic is associated with a sine wave that has a frequency \(\omega\) and an amplitude given by \(C_k\).

The \(k\) terms in practice are limited to a finite range such as \(+/-\ 64, +/-\ 128\), etc. The actual choice of a size for \(k\) will depend upon the FFT speed required, the computer's memory capacity, wordlength and the resolution required.

Both the above two terms \(C_k\) and \(\omega\) can be shown diagrammatically - see Fig. (3.6), and are taken from ref. 20. Here the two time functions \([f(t)]\) have been drawn and their resulting frequency plots constructed using the magnitude \([C_k]\) and the associated frequency \([\omega]\) of each of the harmonics. The Fourier series harmonics can be clearly seen. This method of representation is further outlined in the section "Graphical Representation" on page (35).

Note that if the original signal is reconstructed from the frequency plots that unless an infinite frequency series is used in the reconstruction - not realizable - then a pure square wave is not obtained. This is due to the Gibb's effect and is detailed on page 28.

For time dependant signals that are real and even the frequency plot will be real and even, whilst time dependant signals that are real and odd will have frequency plots that are imaginary and odd.
Digital Signal Processing. [Chapter 3].

\[ f(t) = \sum_{k=-\infty}^{\infty} c_k e^{j \omega_k t} \]

Fourier Series, even function

\[ f(t) = \sum_{k=-\infty}^{\infty} c_k e^{j \omega_k t} \]

Fourier Series, odd function

Fig. (3.6) Diagrammatic Representation of an Even & odd Function.
3.2-2. Gibbs Phenomena.

Sharp discontinuities in the time signal such as that in the square wave, create problems when using equation (3) to reconstruct the time signal using the inverse Fourier transformation on a limited number of terms. This shows itself as an oscillation - or ringing, which shows up at all sharp discontinuities in the reconstructed time signal. This phenomena is referred to as the Gibb's effect (17).

The effect of the Gibb's phenomena can be clearly seen when a truncated Fourier series is used to reconstruct the even valued square wave function from which it was derived:

This problem cannot be eliminated by simply taking more terms as the Fourier series does not converge uniformly at all points. A solution is to multiply the time function by a second function that acts as a window and effectively limits the time function to that of the window function.
3.3. The Fourier Integral - or Fourier Transform.

The Fourier series has so far been restricted to continuous periodic time functions and many cases arise in engineering where the signal is not periodic. Typical of these are shock pulses, impulse responses, starting transients etc.

The exponential form of the Fourier series can be modified to include time functions that are not periodic and have only a limited time duration.

From the Fourier series representation given in equation (3) we have:

\[ f(t) = \sum_{K=-\infty}^{K=\infty} C_k e^{j k \omega_0 t} \]

and from equation (4):

\[ C_k \ T = \int_{-T/2}^{T/2} f(t) e^{-j k \omega_0 t} \ dt \]

If the period T is extended to infinity then as:

\[ \omega_0 = \frac{2\pi}{T} \]

if follows that \( \omega_0 \) will tend to zero

and the term \( k \omega_0 \) can be replaced by \( \omega \).

The term \( C_k \cdot T \) is continuous and can be replaced by \( F(\omega) \) which gives:

\[ f(t) = \int_{-\infty}^{\infty} F(\omega) e^{j \omega t} \ d\omega \]
Digital Signal Processing. [Chapter 3]. page 30

\[
F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-j\omega t} \cdot dt \quad \ldots \quad (5)
\]

and from eqn (3) :-

\[
f(t) = \sum_{K=-\infty}^{\infty} F(\omega) \cdot e^{j K \omega t}
\]

replacing T by \(2\pi/\omega_0\) gives :-

\[
f(t) = \sum_{K=-\infty}^{\infty} F(\omega) \cdot \omega_0 \cdot e^{j K \omega_0 t} / 2\pi
\]

and as T tends to infinity \(\omega_0\) tends to \(d\omega\) and \(k\omega_0\) tends to \(\omega\) giving :

\[
f(t) = \int_{-\infty}^{\infty} F(\omega) \cdot e^{j \omega t} \cdot d\omega \quad \ldots \quad (6)
\]

Equation (5) is referred to as the Fourier Transform of \(f(t)\) whilst equation (6) is the Inverse Fourier Transform of \(F(\omega)\).

There is now no requirement for the signal to be periodic - only that its period is infinite, furthermore the spectrum produced is now continuous.

In practice the signal will have a finite length and the Fourier Transform taken over this finite time period leads to a truncation of the signal as shown on page 24 ;
The signal \( f(t) \) for \(-T/2 < t < T/2\)

The signal = 0 elsewhere

The effect is similar to viewing the time signal through a rectangular window as shown by the dotted lines.

The window function = 1 \(-T/2 < t < T/2\)

The window function = 0 elsewhere

The windowed time function \( f_w(t) \) is now a product of the original continuous time function \( f(t) \) and the window function \( w(t) \):

\[
f_w(t) = f(t) \times w(t)
\]

The spectrum produced by the windowed time function \( f_w(t) \) can be found from the Fourier Transform of the product \( f(t) \) and \( w(t) \):

\[
F_w(\omega) = \int_{-\infty}^{\infty} [w(\xi) \cdot f(t)] \cdot e^{-j\omega\xi} \, d\xi
\]

Or alternatively from the convolution of the two separate frequency spectra, the window spectrum \( W(\omega) \) and the signal
Digital Signal Processing. [Chapter 3]. page 32

spectrum $F(\omega)$; -

$$F_W(\omega) = \int_{-\infty}^{\infty} W(\omega) F(\omega - \omega_1) d\omega_1$$

Convolution can be visualised as reversing one of the spectra and passing it in front of the other spectra - in $(\omega_1)$ steps, and forming the product of the two as they overlap - to give the magnitude, and adding their individual phases to give the new phase.

This is shown below where a continuous time function

$$f(t) = A \cos(2\pi t/T)$$

is viewed through a rectangular window of width $T$.

For a rectangular window $W(\omega) = 2T \frac{\sin(\omega t)}{\omega t}$

this is shown below:-

and for the continuous time function $f(t) = A \cos(2\pi t/T)$ and its spectrum:-
Digital Signal Processing. [Chapter 3]. page 33

Reversing $W(\omega)$ and moving it across in front of $F(\omega)$ gives:

This produces the spectrum $F_w(\omega)$ of the windowed time function $f_w(t)$.

It is apparent from the above that $F(\omega)$ is different from $F_w(\omega)$. 
The effect of the window has been to spread out the line spectra. This spreading is termed leakage and could be reduced by lengthening the period $T$ of the window.

There are a number of different window functions to choose from. Some commonly used windows are listed below in Table (3.1) and are well covered by Brook and Wynne (19).

<table>
<thead>
<tr>
<th>Window Function</th>
<th>Fall Off Rate</th>
<th>Highest Side Lobe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular</td>
<td>20 dB/dec</td>
<td>-13 dB</td>
</tr>
<tr>
<td>Hanning</td>
<td>60 dB/dec</td>
<td>-32 dB</td>
</tr>
<tr>
<td>Hamming</td>
<td>20 dB/dec</td>
<td>-42 dB</td>
</tr>
<tr>
<td>Gaussian</td>
<td>No side lobes</td>
<td>None</td>
</tr>
</tbody>
</table>

The choice of window function and window length is a subjective one and depends upon the time function being analysed and the type of result required. The Hanning window is commonly used as it offers a good fall off rate (60 dB/decade), is easily generated and is a smoother window function than the rectangular window. The rectangular window offers only 20 dB/decade and due to its sharp edges, a discontinuity arises when the reconstituted signal samples are joined together to produce the original signal as shown below:

![Original Signal](image1)

![Reconstituted Signal](image2)

*Effect on Signal Continuity due to a rectangular window.*
The window length ($T_w$) should be as large as possible so as to keep the signal bandwidth large as:

$$\text{Bandwidth} = 1/T_w$$

The sampling or window period has introduced a periodicity into the reconstructed time signal that was not present in the original signal. In addition the sharp discontinuities present in the window give rise to a ripple on the reconstructed signal. Widening the period $T_w$ will reduce the ripple as will the use of a window such as the Hanning window which has a gradual slope off therefore being less severe than the rectangular one.

3.3-1. Graphical Representation of the Fourier Transform.

The exponential form of the Fourier transform given in Eqn. (5) was:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t} dt$$

The exponential term can be split into two parts - the real and imaginary:

$$e^{-i\omega t} = \cos(\omega t) - j \sin(\omega t)$$

and represented as a rotating complex vector having real and imaginary projections as shown in Fig. (3.14):
The Fourier Transform can then be written in terms of these real and imaginary components;-

\[ F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot \cos(\omega t) \, dt - j \int_{-\infty}^{\infty} f(t) \cdot \sin(\omega t) \, dt \]

| Real | Imaginary |

Alternatively the real and the imaginary parts of the function \( F(\omega) \) could be represented in exponential form;-

The real part being;-

\[ F(\omega)_R = \text{Re} \int_{-\infty}^{\infty} f(t) \cdot e^{-j\omega t} \, dt \]

and the imaginary part being;-

Fig. (3.14). Rotating Complex Vector.
Putting $z_{\text{real}} = F(\omega)h$

$z_{\text{imag}} = F(\omega)i$

and letting $f(t) = 1$

Produces functions that are continuous in both frequency and time and a plot of both $z_{\text{real}}$ and $z_{\text{imag}}$ against $\omega$ (on the horizontal x axis) and $t$ (on the horizontal z axis) will each produce a 3D surface with the amplitude on the vertical y axis having a maximum value of 1. A Cosine surface for $z_{\text{real}}$ and a Sinusoidal surface for $z_{\text{imag}}$.

Both these surfaces are shown Figs. (3.15) and (3.16) and were plotted using the Graph package "Derive" on an IBM PC.
Fig. (3.16) Sine surface

This graphical approach is taken a stage further by Peter Kraniauskas [20], see Figure (3.17). Here a Cosine surface has been drawn with it's time axis parallel to the time axis of a signal $f(t)$ and it's frequency axis $\omega$ perpendicular. The signal is a square wave pulse and note that the width of the cosine surface is limited to the width of the pulse - the two time axis must have the same duration.

The cosine surface has then been sliced up -parallel to the time axis, along the $\omega$ axis and at each slice the summation of the area across the slice has been calculated to produce the magnitude term and this has been plotted on the frequency axis. This is repeated for all the slices - ideally for $\omega = \pm \infty$, to give the real part of the Fourier transform.

A similar technique is used to obtain the imaginary part of the Fourier transform but this time slicing up the sine surface.

This process can also be used to obtain the inverse of the Fourier transform by slicing the surfaces up parallel to the frequency axis and projecting the summation of the areas onto the time axis, this is shown in Fig. (3.18).

Note that the resulting time signal is not sharp edged as the surface is not summed from $\omega = \pm \infty$ but over a finite frequency range.
In short the Fourier Transform can be generated by passing the time function through the exponential surfaces and the time function can be generated by passing the Fourier Transform back through the exponential surfaces and summing the result. This graphical surface representation is a good aid in understanding the forward and inverse Fourier transformation.
\[ f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega \]

\[ F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \]

Figs. (3.17) and (3.18)

The plot of \(|F(\omega)|\) against frequency is termed the Amplitude Spectrum of the signal \(f(t)\), whilst the plot of the \(\angle F(\omega)\) against frequency is termed the Phase Spectrum of \(f(t)\).

Another spectrum frequently encountered is the Power Spectrum and just as electrical power through a one ohm resistor is equal to the square of the voltage across it, then signal power is proportional to the square of the amplitude term.

**Periodic Signals.**

Just as the Fourier Transform of a periodic signal will produce a line spectrum so will the power spectrum, which is obtained by plotting the term \(|F(\omega)|^2\) against frequency.

The power spectrum shows how the power contributed by the individual discrete frequency components is distributed across the frequency spectrum. If we sum all such components then we get the total signal power - this is Parseval's theory which states that the power in each individual spectra add up to the total power of the spectrum.

*Note.* That power is used in the general sense and the square of the amplitude term need not have the units of Watts.

**Random Signals.**

If a random signal is analysed then both the Fourier Transform and the power spectrum will be continuous.

The the average power produced can be found from the mean
squared value of \( f(t) \) over the period \( T \) seconds;

\[
\text{Average Power} = \frac{1}{2T} \int_{-T}^{T} |f(t)|^2 \, dt \quad \text{(watts)}
\]

This is the average height of the area under the curve \( f(t)^2 \) over the period \( 2T \) seconds.

Parseval's Theorem gives:

\[
\int_{-\infty}^{\infty} |f(t)|^2 \, dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) F(-\omega) \, d\omega
\]

writing \( F(\omega) F(-\omega) \) as \( |F(\omega)|^2 \) dividing both sides by the period \( 2T \) and noting that the theorem can be applied to a time limited signal gives the Average Power:

\[
\frac{1}{2T} \int_{-T}^{T} |f(t)|^2 \, dt = \frac{1}{4\pi T} \int_{-\infty}^{\infty} |F(\omega)|^2 \, d\omega
\]

or the Average Power \( = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{1}{2T} \| F(\omega) \| ^2 \, d\omega \) (watts)

and the term

\[
\frac{1}{2T} \| F(\omega) \|^2
\]

is called the Power Spectral Density (PSD) and given the
symbol $S_{xx}(\omega)$ ie:

$$S_{xx}(\omega) = \frac{1}{T} \left| \frac{F(\omega)}{2} \right|^2 \text{ (volts}^2 \text{ /Hertz)}$$

The average power then is equal to the sum of the spectral density terms over the frequency range $\omega = \pm \infty$.

As the PSD is an even function and wholly real then the negative part is a mirror image of the positive portion and hence only the positive part needs to be plotted. If this is done half the power is lost and a factor of 2 needs to be introduced.

Representing the new Power term by $P(\omega)$ gives:

$$P(\omega) = 2 \cdot S_{xx}(\omega) = \frac{2}{T} \left| F(\omega) \right|^2$$

![Diagram](image)

**Fig. (3.19) Power Spectral Density.**

The term $\left| F(\omega) \right|^2$ is also sometimes called the energy spectral density, it has units of (volts)$^2$ and is commonly used in engineering.

To change Energy spectral density $\left| F(\omega) \right|^2$ to Power spectral density $S_{xx}(\omega)$ divide by the record length $T$ seconds (or multiply by the bandwidth $B$). The record length ($T$) of the time signal is therefore seen to be an important parameter in interpreting the frequency representation of a signal.
Digital Signal Processing. [Chapter 3]. page 44

Note as $F(\omega)$ is complex having both Real and Imaginary parts then: -

$$F(\omega) = \text{Real} + \text{Imaginary} = a(\omega) + jb(\omega)$$

also 

$$F(-\omega) = a(\omega) - jb(\omega)$$

hence 

$$|F(\omega) \times F(-\omega)| = |F(\omega)|^2$$

giving 

$$|F(\omega)|^2 = (a(\omega)+jb(\omega))(a(\omega)-jb(\omega))$$

hence 

$$|F(\omega)|^2 = (a(\omega)^2 + b(\omega)^2)$$

Hence Energy spectral density is the square of the real and imaginary parts of $F(\omega)$. 
3.5. The Laplace Transform Relationship.

The Laplace and Fourier transforms are closely related.

So far the Fourier transform only allows signals to be transformed that exist over all time - past, present and future, i.e. \( t = \pm \infty \), ruling out many signals of interest that only exist after \( t = 0 \).

By introducing a convergence factor this difficulty can be overcome and enable the Fourier transform to be evaluated.

The introduction of this convergence factor also leads to the relationship with the Laplace transform.

The Fourier transform was given by equation (5):-

\[
F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt
\]

One of the Dirichelet conditions mentioned on page (13) was that the magnitude of \( f(t) \) over a complete period has to be finite i.e.;-

\[
\left| \int_{-\infty}^{\infty} f(t) \, dt \right| < \infty \quad \text{or} \quad \int_{-\infty}^{\infty} |f(t)| \, dt < \infty
\]

For a unit step function \( f(t) = 0 \) for \( t < 0 \) and \( f(t) = 1 \) for \( t > 0 \)
hence:

\[
\int_{-\infty}^{\infty} |f(t)| \, dt = \int_{-\infty}^{0} 0 \, dt + \int_{0}^{\infty} 1 \, dt = \lfloor t \rfloor_0^\infty = \infty
\]

as this is not a finite value then the series does not converge and the transform is not valid.

If the unit step function is approximated by a slowly decaying exponential as shown below:

As \( f(t) = 0 \) for \( t > 0 \);

\[
\int_{-\infty}^{\infty} |f(t)| \, dt = \int_{-\infty}^{0} 1 \cdot e^{-at} \, dt
\]

which gives:

\[
\text{RHS} = \int_{0}^{\infty} 1 \cdot e^{-at} \, dt = \left[ \frac{-1 \cdot e^{-at}}{a} \right]_0^\infty
\]

\[
\text{RHS} = \int_{0}^{\infty} 1 \cdot e^{-at} \, dt = \left[ \frac{-1}{a} [1 - 0] \right]
\]
Digital Signal Processing. [Chapter 3]. page. 47

therefore: -

\[ \int_{-\infty}^{\infty} e^{-\omega t} \cdot dt = \frac{1}{a} \]

ie. a finite value.

hence the series will converge and the Fourier transform is valid.

It can be seen in the above that (a) must be greater than zero to give a finite value to the term \( \frac{1}{a} \).

In addition (a) should be very small for \( e^{-\frac{a}{t}} \) to approximate the unit step.

Hence if a function \( f(t) \) that starts after \( t=0 \) is multiplied by the term \( e^{-\frac{a}{t}} \), the series can be made to converge providing that (a) is very small.

If the Fourier Transform of the modified time signal is given by: -

\[ F^*(\omega) = \int_{0}^{\infty} f(t) \cdot e^{-\omega t} \cdot e^{-j\omega t} \cdot dt \]

Then letting \( s = a + j\omega \) and writing \( F^*(\omega) \) as \( G(s) \) gives: -

\[ G(s) = \int_{0}^{\infty} f(t) \cdot e^{-st} \cdot dt \quad \ldots \quad (9) \]

Equation (9) then is the Laplace transform of \( f(t) \) and can be written as: -
Digital Signal Processing. [Chapter 3].  page. 48

\[
L \left[ f(t) \right] = G(s) = \int_{0}^{\infty} f(t).e^{-st}.dt
\]

Now as \( s \to j\omega \) then \( G(s) \to G(j\omega) \)

Similarly \( G(j\omega) \to F(\omega) \)

where \( F(\omega) \) is the Fourier Transform.

So as \( s \to j\omega \) then :-

\[
G(j\omega) = \int_{0}^{\infty} f(t).e^{-j\omega t}.dt = F(\omega)
\]

Hence the Fourier Transform can be obtained from the Laplace transform for signals that only exist after time \( t=0 \), by putting the \( s \) terms equal to \( j\omega \) and taking the limits from 0 to infinity.

Similarly the inverse Laplace transform of \( G(s) \) yields the function \( f(t) \) :-

\[
f(t) = L^{-1 \left[ G(s) \right]}
\]

Note. The inverse Laplace transform is usually performed using a table of Laplace transforms.
eg. Consider a decaying sine wave \[ y = e^{-st} \cdot \sin(\omega t) \] ;

\[ G(s) = \int_0^\infty f(t) dt = \int_0^\infty e^{-st} \cdot \sin(\omega t) \cdot e^{-st} dt \]

Replacing \( \sin(\omega t) \) by its exponential representation ;

\[ G(s) = \int_0^\infty e^{-st} \cdot \frac{1}{2j} \left[ e^{j\omega t} - e^{-j\omega t} \right] \cdot e^{-st} dt \]

and expanding ;

\[ G(s) = \frac{1}{2j} \int_0^\infty \left[ e^{-(s+\omega) t} - e^{-(s-\omega) t} \right] dt \]

This integral reduces to ;

\[ G(s) = \frac{1}{2j} \left[ \frac{e^{-(s+\omega) t}}{s+a-j\omega} + \frac{e^{-(s-\omega) t}}{s+a+j\omega} \right]_0^\infty \]

And gives ;

\[ G(s) = \frac{\omega}{(s+a)^2 + \omega^2} \]

And as \( s \to j\omega \);
then \( G(j\omega) = \frac{\omega_0}{(j\omega + a)^2 + \omega_0^2} = \frac{\omega_0}{a^2 + \omega_0^2 - \omega^2} + j2a\omega \)

The magnitude is :-

\[ |F(\omega)| = |G(j\omega)| = \frac{\omega_0}{\sqrt{(a^2 + \omega_0^2 - \omega^2)^2 + (2a\omega)^2}} \]

and the Phase is :-

\[ \angle F(\omega) = \angle G(j\omega) = -\tan^{-1} \left[ \frac{2a\omega}{(a^2 + \omega_0^2 - \omega^2)} \right] \]

The continuous spectrum - magnitude and phase of the above expression is shown below:-

![Diagram showing the magnitude and phase of G(j\omega)](image)

Fig. (3.22) Continuous Spectrum Mag. & Phase.

It can be seen from the above expression for the magnitude that as \( \omega \) tends to infinity \( G(j\omega) \) tends to zero i.e. the series converges and hence the transform is valid.
3.6. The Discrete Fourier Transform (DFT)

The methods considered so far to produce a Fourier series have all assumed that the signal $f(t)$ is a continuous signal over the period of interest. If this series is to be used in a digital computer then a form of the Fourier series will have to be found that can accommodate a discontinuous signal.

The process of sampling is common to all digital devices and converts a continuous analog signal into a discontinuous discrete - or sampled data signal. The sampler usually takes the form of an analog to digital converter which under the control of the computer takes $N$ samples - one every $T$ seconds. $T$ being the sampling time.

Consider the sampler shown below where the input is held constant and the output is a train of pulses having unity area and spaced $T$ seconds apart:

![Continuous Signal, Sampler, Impulse Train Diagram](image)

Fig. (3.23)

Let $\delta(t)$ be a unit impulse occurring at time $t=0$ and $\delta(t-kT)$ a unit impulse occurring $kT$ seconds later.

Let $f^*(t)$ be the product of $f(t)$ and the impulse $\delta(t-kT)$:

$$f^*(t) = f(t) \times \delta(t-kT)$$
as the area of an impulse is unity then the area under $f^*(t)$ will be equal to the value of $f(t)$ at $t=kT$. This is called the sifting property and can be written:

$$f(t) \times \delta(t-kT) = f(kT)$$

If the input to the sampler is a time varying signal then as a result of the above the output will be a train of pulses whose height reflects the value of $f(t)$ at the sampling instant $kT$. This is shown below:

![Diagram of continuous signal, sampler, and pulse train](image)

If $N$ samples of $f(t)$ are taken then $f^*(t)$ can be represented by the series:

$$f^*(t) = x_0 + x_1 \delta(t) + x_2 \delta(t-T) + x_3 \delta(t-2T) + \ldots + x_{n-1} \delta(t-(n-1)T)$$

As the Laplace transform of an impulse is:

$$\mathcal{L}[\delta(t)] = 1$$

and

$$\mathcal{L}[\delta(t-kT)] = e^{-ksT}$$

then the above series can be written in the form:
Digital Signal Processing. [Chapter 3]. page 53

\[ f(t) = x_0 + x_1 e^{-sT} + x_2 e^{-s^2T} + x_3 e^{-s^3T} + \ldots + x_{n-1} e^{-s^{(n-1)}T} \]

If \( s \) is replaced by \( j\omega \) then the sum of the series gives the nth discrete Fourier coefficient \( G_n(j\omega) \):

\[ G_n(j\omega) = x_0 + x_1 e^{-j\omega T} + x_2 e^{-j\omega^2T} + \ldots + x_{n-1} e^{-j\omega^{(n-1)}T} \]

If \( N \) samples of \( f(t) \) are taken then the fundamental frequency \( \omega_0 \) will be:

\[ \omega_0 = \frac{2\pi}{(\text{total sample time})} \]

or \[ \omega_0 = \frac{2\pi}{NT} \]

and the nth discrete coefficient coincides with the nth harmonic of the fundamental frequency \( \omega_0 \);

\[ \text{i.e.} \quad \omega_n = n\omega_0 = \frac{2\pi n}{NT} \]

Similarly, the highest harmonic will be given by:

\[ \omega_N = \frac{2\pi}{T} \]

or \[ \omega_N = \frac{2\pi f_s = 2\pi}{T} \]

where \( f_s \) is the sampling frequency.

Putting the above into the expression for \( G_n(j\omega) \) gives:

\[ G_n(j\omega) = x_0 + x_1 e^{-j2\pi n/N} + x_2 e^{-j2\pi^2n/N} + \ldots + x_{n-1} e^{-j2\pi^{(n-1)}n/N} \]
It can be seen that the $N$ series will each yield $N$ frequency coefficients - or harmonics of the fundamental frequency $\omega_0$.

The above series can be written as a summation:

$$G_s(j\omega) = \sum_{k=0}^{N-1} x(k) e^{-j \frac{2\pi k \omega}{N}}$$

In keeping with Parseval's theorem the mean power of the digital spectrum should be equal to the sum of the powers of the individual discrete spectral components - or coefficients.

Hence a scaling factor of $1/N$ needs to be applied.

Writing $X_s$ in place of $G_s(j\omega)$ gives:
Equation (10) gives the discrete form of the Fourier transform and equation (11) the inverse transform:

\[
X_n = \frac{1}{N} \sum_{k=0}^{N-1} x(k) e^{-j\frac{2\pi k n}{N}} \quad \ldots (10)
\]

\[
x(k) = \frac{1}{N} \sum_{k=0}^{N-1} X_n e^{j\frac{2\pi k n}{N}} \quad \ldots (11)
\]

If \(e^{-j\frac{2\pi k}{N}}\) is written in the form \(W_k\) then equation (10) can be written in the form:

\[
X_n = \frac{1}{N} \sum_{k=0}^{N-1} x(k) W_k^{nk} \quad \ldots (12)
\]

3.6-1. Practical Implications of the DFT.

I. The DFT requires equation (12) to be repeated \(N\) times to give all the \(N\) coefficients \(X_n\), i.e., \(X_1, X_2, X_3, \ldots X_N\). Also, each equation has \(N\) terms and this results in \(N^2\) multiplications for each DFT.

II. The time series from which the DFT is being calculated is assumed to be continuous and as it is measured over a finite time period \(NT\) seconds the DFT assumes that the time signal repeats itself every \(NT\) seconds, see overpage:
Fig. (3.26) Periodicity in a DFT.

III. The sampled time signal is unlikely to have a period exactly equal to NT seconds and as a result a different portion of the signal will be included every time the N samples are taken. Thus the resulting DFT will not be identical every time. (Unless triggered from the same point every time for a periodic signal).

IV. Due to the sampling process the discrete spectrum produced by the DFT will itself be periodic, with a period Nωo radians/sec.

Consider the spectral coefficient at N+n :-

\[
X_{(n+N)} = \frac{1}{N} \sum_{K=0}^{N-1} x(k) e^{-j2\pi\frac{K(N+n)}{N}}
\]

giving :-

\[
X_{(n+N)} = 1 > \left| \sum_{K=0}^{N-1} x(k) e^{-j2\pi\frac{(1+n)}{N}} \right|
\]
Digital Signal Processing. [Chapter 3]. page. 57

i.e.;

\[ X(n+N) = \frac{1}{N} \sum_{k=0}^{N-1} x(k) \cdot e^{-j\frac{2\pi k}{N}} \cdot e^{-j\frac{2\pi kN}{N}} \]

as \( e^{-j\frac{2\pi k}{N}} = 1 \) for \( k=0,1,2,3 \) etc.

then ;-)

\[ X(n+N) = \frac{1}{N} \sum_{k=0}^{N-1} x(k) \cdot e^{-j\frac{2\pi kN}{N}} \]

this leads to \( X(n+N) = X_n \)

and hence the DFT repeats itself every N coefficients.

It can also be shown that \( X(n-N) = X_{-n} = X^*_n \) the complex conjugate of \( X_n \).

The DFT then is periodic in N both for the +ve and the -ve frequency axis. This is shown below under section V.

V. If the data is wholly real then the DFT is symmetrical about the N/2 coefficient and there is no need to calculate the coefficients above N/2. This reduces the number of calculations to be performed to \( N^2/2 \).

Note: frequency = Coefficient number x \( \omega_0 \).
VI. If frequency components higher than \((N \omega_s/2)\) (i.e. \(f_s/2\)) are present in the sampled data signal \(f*(t)\) then aliasing — or folding of the spectrum, will occur.

This arises due to the frequency limits in the spectrum being \(\pm f_s/2 \pm (\pm 1/2T)\) Hz any higher frequencies should not show up. In practice signals outside the useful bandwidth of 0 to \(f_s/2\) Hz will wrap around the folding frequencies of \(\pm f_s/2\) Hz and appear as aliases.

This effect is shown below for a square wave signal of frequency 100 Hz which includes the odd harmonics 100, 300, 500 and 700 Hz.
If a DFT is performed on this square wave taking 500 samples over a 1 second period - this gives a sampling frequency of 500 Hz and a sampling interval of 1/500 second. The ideal discrete spectrum will only show the ±100 Hz coefficient as the ±300 Hz and above are outside the ±250 Hz limit (or ±fs/2 Hz).

![Ideal Discrete Spectrum](image)

Fig. (3.29) Ideal Discrete Spectrum.

In reality unless the signal is bandlimited to exclude the higher frequencies they will wrap around the ±250 Hz frequency (±fs/2 Hz) and show up as aliases - or ghost coefficients. The ±300 and ±500 Hz harmonics being the problem ones here, the +300 Hz being +50 Hz about +250 Hz will turn up 50 Hz below i.e. at +200 Hz, and the 500 Hz being 250 Hz larger will turn up at at the origin (0 Hz). The same argument being applied to the negative coefficients. This is shown in Fig's. 30 and 31.
Fig. (3.30) Discrete Spectrum.
(Showing wrap around of higher harmonics.)

Fig. (3.31) Practical Discrete Spectrum.
(Showing Aliases.)
A. Calculation of the DFT.

This can be easily performed on a computer for any number of samples \(N\) although it is very computationally intensive - requiring \(N^2\) multiplications per DFT. (64 for a length 8 DFT)

From equation (12) we have that the DFT is given by:

\[
x(k) = \frac{1}{N} \sum_{k=0}^{N-1} x(n)W_N^{nk}
\]

where \(W_N^{nk} = e^{-j2\pi nk/N} = \cos(2\pi nk/N) - j \sin(2\pi nk/N)\)

If the \(x(t)\) is a single valued real signal then the samples \(x(k)\) will be wholly real as pointed out on page \(\text{71}\) at the end of section 3.2. If on the other hand a two channel analysis is being undertaken, or the data has been previously processed to yield a phase relationship, then the samples \(x(k)\) will be complex (having both real and imaginary components).

Hence \(x(k)\) can written in the form:

\[
x(k) = a(k) + j b(k)
\]

where \(b(k)\) may on occasions be zero. The DFT then becomes:
the Fourier transform of \( X_\alpha \) will also be complex having real \( A(n) \) and imaginary \( B(n) \) terms:

\[
X_\alpha = A(n) + jB(n) \quad \ldots \quad (13)
\]

where \( n = 0 \) to \( N-1 \)

expanding the complex DFT into the form \((a + jb)\) and equating the real and imaginary terms to that for \( A(n) \) and \( B(n) \) gives:

\[
A(n) = \frac{1}{N} \sum_{k=0}^{N-1} [a(k) \cdot \cos(2\pi nk/N) + b(k) \cdot \sin(2\pi nk/N)] \quad (14a)
\]

and:

\[
B(n) = \frac{1}{N} \sum_{k=0}^{N-1} [b(k) \cdot \cos(2\pi nk/N) - a(k) \cdot \sin(2\pi nk/N)] \quad (14b)
\]

We then need to calculate 2 terms in each part of the transform, each having \( N \) values and together with the \( N \) transform coefficients, gives \( 4N^2 \) multiplications.
It follows that if the data is wholly real then $B(n)$ will be zero and only $A(n)$ needs to be calculated. In addition, as the spectrum will be symmetrical about the $N/2$ coefficient, only the first $N/2$ terms need to be calculated, i.e. $k=0$ to $N/2$. This reduces the multiplications to $N^2$. 
Flow chart for DFT.

Fig. (4,1)
A flow chart to calculate the complex DFT is shown in Fig.(4.1) and a BBC Basic computer program for this in Fig.(4.2).

In the program a 'look up' table of the Twiddle factors, is first produced and this is stored at the end of the program in the order real part followed by imaginary part.

The twiddle factors are found from :-

\[ WR(nk) \text{ written as } WR(I,K) = \cos(2\pi I K / N) \]
\[ WI(nk) \text{ written as } WI(I,K) = \sin(2\pi I K / N) \]

the summation for \( A(n) \) being :-

\[ A(n) = a(k) \cdot WR(nk) + b(k) \cdot WI(nk) + A(n) \]

is written as :-

\[ OXR(I) = XINR(K) \cdot WR(I,K) + XINI(K) \cdot WI(I,K) + OXR(I) \]

and for \( B(N) \) :-

\[ B(n) = b(k) \cdot WR(nk) - a(k) \cdot WI(nk) + B(n) \]

being written as :-

\[ XOI(I) = XINI(K) \cdot WR(I,K) - XINR(K) \cdot WI(I,K) + XOI(I) \]

lines 160 and 170.

At line 210 the magnitude \(|X*I|^2\), referred to as MAG(I) is calculated.

A wholly real unity square wave is used as an example having a sample length of 8 and the data samples are stored under DATA at the end of the program at line 400.
The program is also included on the attached floppy disc.

As a guide to the speed of the DFT the BBC clock is used to measure the time taken in performing the DFT, hence at line 34 the TIMER is set and at 236 the elapsed time - at this point, is determined and printed.

The results are shown after the DFT listing in FIG.4.3 and give the elapsed time, the nth coefficient term number, then the real (A(n)) and the imaginary (B(n)) DFT terms together with the Power term |Xₙ|² written as MAG. The phase term has not been calculated but could easily be added as (A(N)) and (jB(N)) terms are known and :-

\[
\text{Phase} = -\tan^{-1} \left( \frac{B(N)}{A(N)} \right)
\]

A listing of the results of the DFT is shown in Fig.(4.3), and a graphical representation of the results using the Cricket Graph software is shown in Fig.(4.4).
DFT Basic Program Fig. (4.2)

10 REM .... DFT_PROG .... PROGRAM *** DATA AT LINE 410
20 INPUT " ** VALUE OF N TO BE USED **" ; N
30 DIM WR(N-1,N-1), WI(N-1,N-1), XINR(N), XINI(N)
34 V=TIMER
35 DIM OXR(N), XOI(N), MAG(N)
36 CLS: PRINT " START TIME IS ..." ; TIME$ 
40 REM ** GENERATE TWIDDLE FACTORS **
50 REM ** REAL PART * IMAGINARY *
60 FOR I=0 TO N-1 : FOR K=0 TO N-1
70 WR(I,K)=COS(2*3.142*I*K/N)
80 WI(I,K)=SIN(2*3.142*I*K/N)
90 NEXT K : NEXT I
100 REM ** CALC DFT **
110 FOR I=0 TO N-1
120 OXR(I)=0 : XOI(I)=0
130 RESTORE 400
140 FOR K=0 TO N-1
150 READ XINR(K), XINI(K)
160 OXR(I)=XINR(K)*WR(I,K)+XINI(K)*WI(I,K)+OXR(I)
170 XOI(I)=XINI(K)*WR(I,K)-XINR(K)*WI(I,K)+XOI(I)
180 NEXT K : NEXT I
190 REM ** CALC MAGNITUDE **
200 FOR I=0 TO N-1
210 MAG(I)=SQR(XOI(I)^2+OXR(I)^2)
220 NEXT I
230 REM ** PRINT RESULTS **
235 PRINT " END TIME IS ...." ; TIME$
236 PRINT " TIME IN SECONDS IS ..." ; (TIMER - V)
240 PRINT " ** RESULTS OF " ; N ; " POINT DFT ARE AS FOLLOWS ** "
250PRINTTAB(3);"N";SPC(12);"A(N)";SPC(10);"B(N)";SPC(11);"MAG"
260 FOR I=0 TO N-1
270 PRINT USING"££.££";I,OXR(I),XOI(I),MAG(I)
280 NEXT I
290 END
400 DATA 1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0
Start Time is 23:13:44
End time is 23:13:45
Time in seconds is .984375

** results of 8 point DFT are as follows **

<table>
<thead>
<tr>
<th>n</th>
<th>A(n)</th>
<th>B(n)</th>
<th>MAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>4.00</td>
<td>0.00</td>
<td>4.00</td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>-2.41</td>
<td>2.61</td>
</tr>
<tr>
<td>2.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3.00</td>
<td>1.00</td>
<td>-0.41</td>
<td>1.08</td>
</tr>
<tr>
<td>4.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>5.00</td>
<td>1.00</td>
<td>0.41</td>
<td>1.08</td>
</tr>
<tr>
<td>6.00</td>
<td>-0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>7.00</td>
<td>1.00</td>
<td>2.42</td>
<td>2.62</td>
</tr>
</tbody>
</table>

DFT Results, Fig. (4.3).
Fig. (4.4) Results of DFT

8 Point DFT Real Part A(n)

Amplitude

0 1 2 3 4 5 6 7

8 Point DFT Imag Part

Amplitude

0 1 2 3 4 5 6 7

8 Point DFT Power Plot (Mag)

Axis Label

0 1 2 3 4 5 6 7

Power (Mag)
The Fast Fourier Transform (FFT) is basically a more efficient procedure — or algorithm, for calculating the DFT. Various FFT algorithms have been produced and they are usually known by the name of their authors. The Cooley-Tukey algorithm (4) is probably the best known followed by the Goertzel (21) and Winograd (22) FFT algorithms.

From equation (12) the DFT is:

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} x(k) W_N^{nk}$$

For N=4 this gives the set of equations:

$$
X_0 = W_4^0 . x(0) + W_4^0 . x(1) + W_4^0 . x(2) + W_4^0 . x(3) \\
X_1 = W_4^1 . x(0) + W_4^1 . x(1) + W_4^1 . x(2) + W_4^1 . x(3) \\
X_2 = W_4^2 . x(0) + W_4^2 . x(1) + W_4^2 . x(2) + W_4^2 . x(3) \\
X_3 = W_4^3 . x(0) + W_4^3 . x(1) + W_4^3 . x(2) + W_4^3 . x(3)
$$

Note:— the Twiddle Factors form is $W_N^{nk}$ or $e^{-j2\pi nk/N}$ hence $W_4^{31}$ is actually $e^{-j\cdot3\cdot\pi/4}$ or $e^{-j\cdot3\cdot\pi/4}$

Writing the Twiddle Factor $W_4^{31}$ in the form W31 and omitting the N suffix gives the matrix form shown below:

For N=4:

$$
\begin{align*}
X_0 & = W0 \ W0 \ W0 \ W0 \ x \ x(0) \\
X_1 & = W10 \ W11 \ W12 \ W13 \ x(1) \\
X_2 & = W20 \ W21 \ W22 \ W23 \ x(2) \\
X_3 & = W30 \ W31 \ W32 \ W33 \ x(3)
\end{align*}
$$

The twiddle factors $W_N^{nk}$ can be simplified by forming the product $(nk)$ and cancelling with N if required. This also
shows up the symmetry that is present in the twiddle factors, for example:

\[ W_{4^{23}} = W_{4^{32}} = W_{4^6} = W_2^3 = W_4^{12} = \text{etc.} \]

The following diagram shows this symmetry for the twiddle factors in the above matrix for \( N=4 \). Note that \( W_{12} = W_4^{-1.2} = W_4^2 \) and that it is a vector of unit magnitude with a phase lag \((2/4)x360 = -180\) degrees. Similarly \( W_4^0 = W_4^4 = 1 \).

![Twiddle Factor Symmetry Diagram](image)

**Fig. (4.5) Twiddle Factor Symmetry.**

Thus the above array (for \( N=4 \)) reduces to:

\[
\begin{array}{cccc|ccc}
X_0 & = & 1 & 1 & 1 & 1 & x & x(0) \\
X_1 & & 1 & W_1 & W_2 & W_3 & x(1) \\
X_2 & & 1 & W_2 & 1 & W_2 & x(2) \\
X_3 & & 1 & W_3 & W_2 & W_1 & x(3)
\end{array}
\]

The twiddle factor array from the DFT - (equation 10) can be simplified and will always have 1's in its first row and column. As shown above the complex multiplications in the DFT have now been reduced from 16 to 8.

This reduction technique is at the heart of the FFT algorithms.

One of the first to adopt this reduction process to a computer application was Cooley and Tukey in 1965. This
has since been modified by various authors, see Burrus and Parks (1) and Bateman & Yates (2).

The basic Cooley - Tukey algorithm reduces the more complex DFT equation into a set of simpler more manageable equations. It does this by means of a mapping routine that enables the DFT to be "uncoupled" or broken into parts.

For a more detailed treatment of mapping see Burrus & Parks (1).

This technique is shown below - first for N=4 and then for N=8:

**Cooley - Tukey algorithm (For N=4).**

From the DFT equation (12):-

\[ X_n = \frac{1}{N} \sum_{k=0}^{N-1} x(k) W_N^{nk} \]

Let N have the factors r1 and r2 and let the terms n and k be replaced by the mapping terms n1,n0 and k1,k0. n and k being related to the mapping terms using the factors r1 and r2 as shown below:

\[
\begin{align*}
  n &= n1.r1 + n0 \\
  k &= k1.r2 + k0
\end{align*}
\]

The mapping used needs to be unique and needs to be able to split the DFT up.

For the case N=4 the factors are r1 and r2 = 2 and give the mapping relationships shown:
An inspection shows that $n$ and $k$ are being replaced by a natural binary series - i.e. converted into binary.

These mapping equations are better illustrated in a tabular form:— (Note. the binary representation.)

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

hence for $n=2$ , $k=3$ and $X_n = X_3$ the mapping becomes:—

from $n = 2n_1 + n_0$ and $k = 2k_1 + k_0$

i.e. $n = 2 = 2(1) + (0)$ and $k = 3 = 2(1) + (1)$

and as $X_n$ becomes $X(n_1, n_0)$

then $X_k$ becomes $X(1, 0)$

hence $n$ is replaced by $(2n_1 + n_0)$

and $k$ is replaced by $(2k_1 + k_0)$

similarly $X_3$ is replaced by $X(1, 1)$

Then writing $X_n$ as $X(n)$ and replacing $n$ and $k$ by their mapped values gives:—

\[
X(n_1, n_0) = \sum_{k_0=0}^{r_1-1} \sum_{k_1=0}^{r_2-1} x(k_1, k_0) W_n^{(2n_1 + n_0)(2k_1 + k_0)} \quad (15)
\]

Expanding the index of $W_n$ and noting that $r_1 \times r_2 = N$ ;

\[
W_n(n_1, r_1k_1r_2) = W_n(Nn_1k_1) = e^{-j\frac{2\pi}{N}n_1k_1} = e^{-j\frac{2\pi}{N}1k_1} = 1
\]
As \( r_1 \) and \( r_2 = 2 \)

\[
X(n_1,n_0) = \frac{1}{2} \sum_{k_0=0}^{\infty} \sum_{k_1=0}^{L-1} [x(k_1,k_0).W_4^{(n_0, 2k_1)}(n_0, k_0)(2n_1, 2k_1)]
\]

This gives:

\[
X(n_1,n_0) = \frac{1}{2} \sum_{k_0=0}^{\infty} \sum_{k_1=0}^{L-1} [x(k_1,k_0).W_4^{(n_0, 2k_1)}(n_0, k_0)(2n_1, k_0)]
\]

and separating the \( k \) terms gives:

\[
X(n_1,n_0) = \frac{1}{2} \sum_{k_0=0}^{\infty} \sum_{k_1=0}^{L-1} [x(k_1,k_0).W_4^{(2n_0, k_1)}] W_4^{(n_0 + 2n_1, k_0)}
\]

The DFT can now be "uncoupled" into two smaller DFT's that can be treated separately.

Letting the innermost sum be given by \( P(n_0k_0) \):

\[
P(n_0,k_0) = \sum_{k_1=0}^{L-1} [x(k_1,k_0).W_4^{(2n_0, k_1)}]
\]

which gives the first summation for \( n_0=0 \) and \( k_0=0 \) as:

\[
P(n_0,k_0) = x(k_1,k_0).W_4^{(2n_0, k_1)} + x(k_1,k_0).W_4^{(2n_0, k_1)}
\]

ie \( P(00) = x(00)W_4^{(2,0,0)} + x(10)W_4^{(2,0,1)} \)

and in matrix form writing \( W_4^{(2,0,0)} \) as \( W_{200} \), gives the array:

\[
\begin{array}{c}
\end{array}
\]
Noting that $W_{211} = W_2$ this simplifies to:

$$\begin{vmatrix} P(00) \\ P(01) \\ P(10) \\ P(11) \end{vmatrix} = \begin{vmatrix} W_{200} & 0 & W_{201} & 0 \\ 0 & W_{200} & 0 & W_{201} \\ W_{210} & 0 & W_{211} & 0 \\ 0 & W_{210} & 0 & W_{211} \end{vmatrix} \begin{vmatrix} x(00) \\ x(01) \\ x(10) \\ x(11) \end{vmatrix}$$

and can be represented using a butterfly notation:

```
        x(00)   P(00)
         |       |
        x(01)   P(01)
         |       |
        x(10)   W_4^2 P(10)
         |       |
        x(11)   W_4^2 P(11)
```

or:

$$P(11) = x(00) + x(10).W_4^2$$

finally the outer sum is put equal to $Q(n_0,n_1)$ and is evaluated from:

$$Q(n_0,n_1) = \frac{1}{4} \sum_{k_0=0}^{1} [P(n_0,k_0).W_4^{-2(n_1+n_0)}k_0]$$

Some of the summation terms are shown below and the full set expressed in matrix form.

eg. for $n_0=0,n_1=0$

$$Q(00) = P(00).W_4^{0+0}.0 + P(01).W_4^{0+0}.1$$

giving:

$$Q(00) = P(00).W_4^0.0 + P(01).W_4^0.1$$
eg. for n_0=0, n_1=1

\[ Q(01) = P(00).W_4^{2+0}.0 + P(01).W_4^{2+0}.1 \]

giving :

\[ Q(01) = P(00).W_4^{0+0} + P(01).W_4^{0+1} \]

eg. for n_0=1, n_1=0

\[ Q(10) = P(10).W_4^{0+0} + P(11).W_4^{0+1}.1 \]

giving :

\[ Q(10) = P(10).W_4^{1+0} + P(11).W_4^{1+1} \]

eg. for n_0=1, n_1=1

\[ Q(11) = P(10).W_4^{2+0} + P(11).W_4^{2+1} \]

giving :

\[ Q(11) = P(10).W_4^{2+0} + P(11).W_4^{2+1} \]

Simplify by writing \( W_4^{2+0} \) as \( W_0 \) etc. gives the matrix :

\[
\begin{align*}
Q(00) & = 1/4 & W00 & W00 & 0 & 0 & x & P(00) \\
Q(01) & & W20 & W21 & 0 & 0 & P(01) \\
Q(10) & & 0 & 0 & W10 & W11 & P(10) \\
Q(11) & & 0 & 0 & W30 & W31 & P(11)
\end{align*}
\]

Omitting the scaling term 1/4 and noting that W0 =1 this yields the butterflies shown in Fig. 4.6:

\[
\begin{align*}
x(00) & \rightarrow P(00) \\
x(01) & \rightarrow P(01) \\
x(10) & \rightarrow P(10) \rightarrow W_4^{2} \\
x(11) & \rightarrow P(11) \rightarrow W_4^{3} \\
\end{align*}
\]

\textbf{Fig. 4.6 Length 4 DIF FFT.}
The last column of Fig. 4.6 gives the Fourier coefficients $Q(n_1n_0)$.
An inspection of the $n_1n_2$ mapping gives the desired Fourier coefficients $X(N)$:

<table>
<thead>
<tr>
<th>$Q(n_0n_1)$</th>
<th>$X(n_1n_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(00)$ gives $X(00) = X(0)$ Note that if the bits $n_0$ and $n_1$ in $Q(n_0n_1)$ are reversed in value i.e. -bit reversed, that X(N) can easily be evaluated.</td>
<td></td>
</tr>
<tr>
<td>$Q(10)$ gives $X(01) = X(1)$</td>
<td></td>
</tr>
<tr>
<td>$Q(01)$ gives $X(10) = X(2)$</td>
<td></td>
</tr>
<tr>
<td>$Q(11)$ gives $X(11) = X(3)$</td>
<td></td>
</tr>
</tbody>
</table>

ie. $Q(01)$ becomes $Q(10) = X(2)$ using binary conversion.

The input values $x(k_1k_0)$ were transformed into their binary values using a normal 8,4,2,1 binary mapping - (the input map). The binary output values $Q(n_0n_1)$ needed to be unscrambled using a reversed binary sequence 1,2,4,8 - (the output map) and this results in a Decimation In Frequency FFT i.e. a DIF, FFT. If the input was coded using the output map - ie bit reversed, then the final result would be in the correct sequence according to the input map. This would give a Decimation in time FFT i.e. a DIT FFT.

The two techniques are shown later.

An inspection of array $P(nok0)$ shows that there are $N$ elements each having $r_1$ (ie 2) operations giving $N.r_1$ operations in total, whilst in array $Q(n0n1)$ there are $N.r_2$ total operations giving a total number of operations ($T$) of:

$$T = N(r_1 + r_2)$$

where $N=r_1.r_2$
in the limit \( T = N(r_1 + r_2 + \ldots + r_m) \)

where \( N = r_1 r_2 \ldots r_m \)

if all \( r \) are equal then \( N = r^m \)

giving \( m = \log_2 N \)

then as \( T = N r m \)

therefore \( T = N r \log_2 N \)

Hence the number of multiplications required to perform the FFT is given by the above equation and depends mainly on the factor - or radix \( r \).

The value of \( r \) is called the radix of the FFT.

Cooley and Tukey have shown that the use of \( r = 3 \) is usually the most efficient (4) but has only about a 6% gain over \( r = 2 \) or 4 in the number of operations \( T \). In addition the use of 2 or 4 offers advantages when used in binary computers.

Then for the case \( r = 2 \) the number of operations -calculated from \( 2N \log_2 N \), reduces to \( T = 48 \) for \( N = 8 \). Compare this with the 64 for the \( N^2 \) of the DFT, a decrease of 25% in the number of multiplications.

The speed increase is more pronounced at large values of \( N \).

A 1024 FFT taking 2,048 operations against 1,048,576 for the DFT.

Note: If the number of complex multiplications only are looked at then an inspection of the arrays in the above example shows that there are \( m \) arrays each having \( N/2 \) twiddle factors - or complex multiplications, giving a total complex \( (T_{\text{complex}}) \) multiplication of :-
The above example being a radix 2 length 4 Cooley-Tukey DIF FFT.

Equation (15) is at the heart of the Cooley-Tukey algorithm and the binary series mapping can be expressed in a more general form which includes all $m$ terms:

\[
\text{let } n = (2^{m-1}.n_{m-1}) + (2^{m-2}.n_{m-2}) + \ldots \text{ no}
\]

and \( k = (2^{m-1}.k_{m-1}) + (2^{m-2}.k_{m-2}) + \ldots \text{ k0} \)

giving:

\[
X[n(p)] = \frac{1}{N} \sum_{K0=0}^{1} \sum_{K1=0}^{1} \ldots \sum_{K(m-1)=0}^{1} [x[k(p)].W^{Ky}] \ldots (16)
\]

where \( X[n(p)] = X[n_{m-1}, n_{m-2}, \ldots n(0)] \)

for \( m=3 \) \( X[n(p)] = X[n2, n1, n0] \)

and \( x[k(p)] = x[k_{m-1}, k_{m-2}, \ldots k(0)] \)

for \( m=3 \) \( x[k(p)] = x[k2, k1, k0] \)

Now the index \( y = n.k \) but can be expanded and the factors expressed in the form :-
\[ y = y_1 \cdot y_2 \cdot y_3 \ldots Y_m \]

where \( y_1 = n_0 \cdot [2^{m-1} \cdot k(m-1)] \).

for \( m=3 \) \( y_1=n_0 \cdot [4 \cdot k2] \)

\[ y_2 = [2n1 + n0] \cdot [2^{m-2} \cdot k(m-2)] \]

for \( m=3 \) \( y_2=[2.n1+n0].[2.k1] \)

\[ y_3 = [4n2 + 2n1 + n0] \cdot [2^{m-3} \cdot k(m-3)] \]

for \( m=3 \) \( y_3=[4.n2+2.n1+n0].[k0] \)

finally \( y_m = [2^{m-1} \cdot n(m-1) + 2^{m-2} \cdot n(m-2) + \ldots n_0] \cdot k0 \)

This technique is illustrated below:-

For \( N=8 \).

eg. For a radix 2 length 8 Cooley-Tukey DIF FFT, \( m=3 \) and the three factors of the DFT are :-

\[
\begin{bmatrix}
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
X(n2,n1,n0) = L^{13} & L^{13} & L^{13}
\end{bmatrix}
\]

\[
K0=0 \quad K1=0 \quad K2=0
\]

The mapping gives :-

\[ n = 4\cdot n2+2\cdot n1+n0 \quad \text{and} \quad k = 4\cdot k2+2\cdot k1+k0 \]

and as the index \( y = n\cdot k = (4n2+2n1+n0)(4k2+2k1+k0) \)

expanding gives \( y = (n0).4k2+(2n1+n0).2k1+(4n2+2n1+n0).k0 \)
the first factor is \( y_1 = n_0.4k_2 = 4.n_0.k_2 \)

the second factor is \( y_2 = (2.n_1 + n_0).2k_1 \)

and the third factor is \( y_3 = (4n_2 + 2n_1 + n_0).k_0 \)

Taking the inner summation as:

\[
P(n_0,k_1,k_0) = \frac{1}{N} \sum_{k_2=0}^{N-1} x(k_2,k_1,k_0).W_8^y_1
\]

this gives (omitting the N suffix 8) the set of equations:

\[
P(n_0,k_1,k_0) x(k_2,k_1,k_0) W_8^y_1
\]

\[
\begin{align*}
P(0,0,0) &= x(000)W^0 + x(100)W^0 \\
P(0,0,1) &= x(001)W^0 + x(101)W^0 \\
P(0,1,0) &= x(010)W^0 + x(110)W^0 \\
P(0,1,1) &= x(011)W^0 + x(111)W^0 \\
P(1,0,0) &= x(000)W^0 + x(100)W^4 \\
P(1,0,1) &= x(001)W^0 + x(101)W^4 \\
P(1,1,0) &= x(010)W^0 + x(110)W^4 \\
P(1,1,1) &= x(011)W^0 + x(111)W^4
\end{align*}
\]

and gives the first array ;− Note \( W^0 = W_8^0 = 1 \)

and also \( W^4 = W_8^4 = -1 \)
in matrix form:-

\[
P(n0k1k0) = \begin{bmatrix}
    & T/F = W_e y_1 = W_e^{(n_0 + n_1 + 1)k_1}
\end{bmatrix} x(k2k1k0)
\]

\[
\begin{array}{c|cccccccc}
    & P(000) & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
    & P(001) & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
    & P(010) & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
    & P(011) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
    & P(100) & 1 & 0 & 0 & 0 & W4 & 0 & 0 & 0 \\
    & P(101) & 0 & 1 & 0 & 0 & 0 & W4 & 0 & 0 \\
    & P(110) & 0 & 0 & 1 & 0 & 0 & 0 & W4 & 0 \\
    & P(111) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & W4
\end{array}
\]

the inner summation is:-

\[
Q(n0,n1,k0) = \sum_{k1=0}^{1} [P(n0,k1,k0)W_e y_2]
\]

where \( y_2 = (2n1 + n0)2k1 \)

\[
giving \ the \ 2nd \ array:-
\]

\[
\begin{array}{c|cccccccc}
    & Q(000) & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
    & Q(001) & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
    & Q(010) & 1 & 0 & W4 & 0 & 0 & 0 & 0 & 0 \\
    & Q(011) & 0 & 1 & 0 & W4 & 0 & 0 & 0 & 0 \\
    & Q(100) & 0 & 0 & 0 & 0 & 1 & 0 & W2 & 0 \\
    & Q(101) & 0 & 0 & 0 & 0 & 0 & 1 & 0 & W2 \\
    & Q(110) & 0 & 0 & 0 & 0 & 1 & 0 & W6 & 0 \\
    & Q(111) & 0 & 0 & 0 & 0 & 1 & 0 & W6 & 0
\end{array}
\]
The outer summation is:

\[ R(n_0, n_1, n_2) = \frac{1}{N} \sum_{k_1=0}^{N-1} Q(n_0, n_1, k_0) W_{n_2}^{k_1} \]

giving the array:

<table>
<thead>
<tr>
<th>R(n0n1n2)</th>
<th>[ T/F = We^{4\pi i (n_2+2n_0+n_0 k_0)} ]</th>
<th>Q(n0n1k0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(000) = 1/8</td>
<td>1 1 0 0 0 0 0 0</td>
<td>x</td>
</tr>
<tr>
<td>R(001)</td>
<td>1 W4 0 0 0 0 0 0</td>
<td>Q(001)</td>
</tr>
<tr>
<td>R(010)</td>
<td>0 0 1 W2 0 0 0 0</td>
<td>Q(010)</td>
</tr>
<tr>
<td>R(011)</td>
<td>0 0 1 W6 0 0 0 0</td>
<td>Q(011)</td>
</tr>
<tr>
<td>R(100)</td>
<td>0 0 0 0 1 W1 0 0</td>
<td>Q(100)</td>
</tr>
<tr>
<td>R(101)</td>
<td>0 0 0 0 1 W5 0 0</td>
<td>Q(101)</td>
</tr>
<tr>
<td>R(110)</td>
<td>0 0 0 0 0 0 1 W3</td>
<td>Q(110)</td>
</tr>
<tr>
<td>R(111)</td>
<td>0 0 0 0 0 0 0 1 W7</td>
<td>Q(111)</td>
</tr>
</tbody>
</table>

Using the results of the above three arrays the following DIF butterfly representation can be drawn:

![Digital Signal Processing. [Chapter 4]. page. 83](image)

The last row gives the Fourier coefficients in bit reversed order i.e.: n2.n1.n0, reversed gives the corrected values n0.n1.n2;
The time samples $x(1), x(2)$ etc. can be arranged in a bit reversed order to start with which will then give the frequency coefficients in correct order at the end. The DIT butterfly shown below in Fig. 4.8 is derived from the same arrays as the last example but the time samples have been bit reversed at the start, this gives the frequency components in correct order.

With both the above methods the results are stored back in their original locations as the original data is no longer required. The result of this in place storage is a reduction in the memory storage requirements but results in a scrambling of the results (DIF) unless the input is bit reversed (DIT). FFT's are available that use correctly ordered data samples and produces correctly ordered output samples, but at least twice the storage area is required.

\[
\begin{array}{c|c|c|c|c}
\text{Norm} & \text{Reversed} & Q(n0k0k1) & P(n0n1k0) & R(n0n1n2) \\
\hline
x(000) & x(000) & 0 & 0 & 0 \\
x(001) & x(100) & 4 & 1 & 1 \\
x(010) & x(010) & 2 & 2 & 2 \\
x(011) & x(110) & 6 & 3 & 3 \\
x(100) & x(001) & 1 & 4 & 4 \\
x(101) & x(101) & 5 & 5 & 5 \\
x(110) & x(011) & 3 & 6 & 6 \\
x(111) & x(111) & 7 & 7 & 7 \\
\end{array}
\]

Fig. 4.8 Length 8. DIT FFT
B1. Programming the FFT. In Basic.

The Cooley-Tukey algorithm could be implemented directly on a computer but a number of modifications have been suggested by different authors that speed up the algorithm by reducing the number of complex multiplications.\(^{(1)}\),\(^{(2)}\),\(^{(3)}\).

The method described here is based on that by Cooley-Tukey\(^{(4)}\) and repeatedly divides the DFT until only a single pair of terms are left.

It requires a radix 2 DFT and the Twiddle Factors (TF) now lie outside the butterflies.

From the Cooley-Tukey FFT [equation 16]

\[
X(n_1,n_0) = \sum_{k_0=0}^{r1-1} \sum_{k_1=0}^{r2-1} x(k_1,k_0) W_N^{n_1 \cdot k_0} W_N^{n_0 \cdot k_1}
\]

The original mapping gave:

\[
n = n_1 \cdot r_1 + n_0 \quad \text{and} \quad k = r_2 \cdot k_1 + k_0
\]

but in order to move the Twiddle Factors (TF's) outside the butterflies the TF for the inner summation must be able to reduce to 1 or -1.

This gives the new mapping as:

\[
n = n_1 + 2n_0 \quad \text{and} \quad k = \frac{N \cdot k_1 + k_0}{2}
\]

giving \(W_N^{n \cdot k} = W_N^{(n_1 + 2n_0)} W_N^{(N \cdot k_1 / 2 + k_0)}\).
For a length 8 DFT where \(N = 2^3\) the input and output maps used to decouple the DFT are:

\[
\begin{align*}
\text{output map is:} & \quad n0 | 0 & 1 & 2 & 3 & n1 \\
& & 0 & 2 & 4 & 6 & 0 \\
& n = & 1 & 3 & 5 & 7 & 1 \\
& & & & & & \\
\text{and the input map is:} & \quad k0 | 0 & 1 & 2 & 3 & k1 \\
\ & & 0 & 1 & 2 & 3 & 0 \\
& k = & 4 & 5 & 6 & 7 & 1 \\
\end{align*}
\]

Hence \(W_8(n,k) = W_8(n1 + 2n0)(4k1 + k0)\)

\[
= W_8^4.n1.k1, W_8^n1.k0, W_8^n0.k1, W_8^2.n0.k0
\]

ie \(W_8(n,k) = W_8^4.n1.k1, W_8^n1.k0, 1, W_8^2.n0.k0\)

the DFT then becomes:

\[
X(n0,n1) = \frac{1}{8} \sum_{k0=0}^{3} \sum_{k1=0}^{1} [x(k1.k0).W_8^4.n1.k1, W_8^n1.k0, W_8^2.n0.k0]
\]

let the inner sum be given by:

\[
P_1(n1.k0) = \sum_{k1=0}^{1} [x(k1.k0).W_8^4.n1.k1]
\]
and note that as stated earlier the inner TF \( W_{\pm 1} \) has values of 1 or -1 as both \( n \) and \( k \) only have values of 0 or 1.

For the inner sum this gives the first butterfly as shown below:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x(k1k0) )</th>
<th>( \text{P1}(n1k0) )</th>
<th>( \text{TF} = W_{\pm 1k0} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00</td>
<td>( W_0 )</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>01</td>
<td>( W_0 )</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>02</td>
<td>( W_0 )</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>03</td>
<td>( W_0 )</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10</td>
<td>( W_0 )</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>11</td>
<td>( W_1 )</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>12</td>
<td>( W_2 )</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>13</td>
<td>( W_3 )</td>
</tr>
</tbody>
</table>

A closer inspection shows that four pairs are used and each pair has a TF on its lower leg of increasingly higher power:

<table>
<thead>
<tr>
<th>Pair</th>
<th>( \text{TF} = W_{\pm 1} = W_x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &amp; 4</td>
<td>( W_0 )</td>
</tr>
<tr>
<td>1 &amp; 5</td>
<td>( W_1 )</td>
</tr>
<tr>
<td>2 &amp; 6</td>
<td>( W_2 )</td>
</tr>
<tr>
<td>3 &amp; 7</td>
<td>( W_3 )</td>
</tr>
</tbody>
</table>

Writing \( \text{P2}(n1.k0) \) as the product of \( \text{P1}(n1k0) \) and the TF \( W_{\pm 1} \) gives the original FFT as:

\[
X(n0.n1) = \frac{1}{2} \sum_{k0=0}^{3} [\text{P2}(n1.k0).W_{\pm 1k0}]
\]

the summation has now been reduced to a half length FFT.

Note that \( W_{\pm 1} \) can be written as \( W_{4^{-1}0^{-1}0} \) and
that $N/2 = 3$.

Giving:

$$X(n0.n1) = \frac{1}{8} \sum_{k0=0}^{(N/2)-1} \left[ P2(n1.k0).W_{(N/2)^{n0.k0}} \right]$$

The intermediate values $P2(n1.k0)$ have values of:

00 01 02 03 and 04 in the top half

and

10 11 12 13 and 14 in the bottom half.

As $n1$ has a value of 0 in the top half DFT and 1 in the bottom half DFT then the half length FFT remaining is applied twice. First to the top half values and then to the bottom half values.

For the top half $n1=0$ and the FFT becomes:

$$X(n0) = \frac{1}{3} \sum_{k0=0}^{3} \left[ P2(k0).W_{n4^{k0}k4} \right]$$

This is identical in form to the original FFT and can be mapped and reduced in much the same way:

The mapping for the $n0$ and $k0$ for $N=4$ is:

$$n0 = n4 + 2n3 \quad \text{and} \quad k0 = 2k4 + k3$$

$$\begin{array}{c|c|c}
  n3 & 0 & 1 \\
  n4 & 0 & 1 & 2 & 3 \\
  n0= & 0 & 2 & 0 & 1 & 3 & 1 \\
  k3 & 0 & 1 & 2 & 3 & 4 \\
  k0= & 0 & 1 & 0 & 2 & 3 & 1 \\
\end{array}$$
and where \( W_{4}.n^{0}.k^{0} = W_{4}(n^{4}+2an^{3})(2k^{4}+k^{3}) \)

\[
\begin{align*}
W_{4}.n^{0}.k^{0} &= W_{4}(2.n^{4}.k^{4}), \ W_{4}(n^{4}.k^{3}), \ W_{4}(4.n^{3}.k^{4}), \ W_{4}(2.n^{3}.k^{3}) \\
W_{4}.n^{0}.k^{0} &= W_{4}(2.n^{4}.k^{4}), \ W_{4}(n^{4}.k^{3}), \ 1, \ W_{4}(2.n^{3}.k^{3})
\end{align*}
\]

giving the next stage FFT as;-

\[
X(n^{3}.n^{4}) = \frac{1}{8} \sum_{K^{3}=0}^{1} \sum_{K^{4}=0}^{1} \left[ P2(k^{4}.k^{3}), W_{4}.n^{4}.k^{4}, W_{4}(n^{4}.k^{3}), W_{4}(2.n^{3}.k^{3}) \right]
\]

Writing \( C(n^{4}.k^{3}) \) for the inner summation and noting that

\( W_{4}(2.n^{4}.k^{4}) \) only has values of 1 and -1 gives \( C(n^{4}.k^{3}) \) equal to ;- 

\[
C(n^{4}.k^{3}) = \frac{1}{2} \sum_{K^{4}=0}^{1} \left[ P2(k^{4}.k^{3}), W_{4}.n^{4}.k^{4} \right]
\]

giving the butterfly and the next TF ;-

<table>
<thead>
<tr>
<th>( P2(n^{1}.k^{0}) )</th>
<th>( P2(k^{4}.k^{3}) )</th>
<th>( C(n^{4}.k^{3}) )</th>
<th>TF ( W_{4}.n^{4}.k^{3} = W_{8}.n^{4}.k^{3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00, W0</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01, W0</td>
</tr>
<tr>
<td>02</td>
<td>10</td>
<td>10</td>
<td>10, W0</td>
</tr>
<tr>
<td>03</td>
<td>11</td>
<td>11</td>
<td>11, W2</td>
</tr>
</tbody>
</table>

Following this is the last stage.

Let \( C2(n^{4}.k^{3}) \) be the product of \( C(n^{4}.k^{3}) \) and the TF then ;-
Digital Signal Processing. [Chapter 4]. page 90

\[
X(n3.n4) = \frac{1}{8} \sum_{k3=0}^{r} [C2(n4.k3).W_{4}^{n3.k3}]
\]

this time \(n4\) has a value of 0 or 1. For the top half \(n4=0\) giving for the last stage :-

\[
X(n3) = \frac{1}{8} \sum_{k3=0}^{r} [C2(k3).W_{4}^{n3.k3}]
\]

If this is repeated for \(n4=1\) the top half of the butterfly becomes :-

- \(C2(n4.k3)\)
- \(C(n3.n4)\)

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

The complete butterfly is shown in Fig (4.9) together with the intermediate products to assist with relating the butterfly to the method outlined above.

It has been shown that the calculation of the FFT by this method is basically a repetition of the basic butterfly and multiplication by the Twiddle Factor for \(m\) stages.

Terms spaced \(N/4\) apart are used first, followed by \(N/2\) and finally adjacent pairs. The index of the TF goes up by one for every pair used in the same stage, and starts at zero again for each new stage.

The method shown is a DIF (decimation in frequency) FFT and bit reversal of the output is required to obtain the correct
value. If bit reversal of the input was used first the output would be in the correct order and this would give a DIT (decimation in time) FFT as the input would be disordered.
Fig (4.9). Radix 2, Length 8 DIF FFT (External Twiddle Factors)

\[ x(k) \rightarrow T_F^p \rightarrow W_8^p \rightarrow P_1(n1k0) \rightarrow W_8^s \rightarrow P_2(n1k0) \rightarrow W_8^p \rightarrow s_1(n4k3) \rightarrow s_2(n4k3) \rightarrow C(n3n4) \rightarrow x(n) \]

\* (NB. \( W_8^+ = W_4^x \))

I/O map

\[ \begin{array}{cccc}
0 & 0 & 1 & 2 \\
3 & 5 & 6 & 7 \\
\end{array} \]
A Basic program to implement the above radix 2, DIF, FFT has been produced and is shown in Fig.(4.11). In order to follow the programming steps a flow chart is shown in Fig.(4.10).

Again the data used is a Square wave having eight single valued samples per cycle, each having a real and imaginary part - the imaginary parts here being zero. The results are shown in Fig.(4.12). No phase shift has been calculated but could easily be added as AR(N) and AI(N) - the real and imaginary components are available.

\[
\text{Phase} = - \tan^{-1} \left( \frac{AI(N)}{AR(N)} \right)
\]

A timer was used in the program to measure the execution time and an inspection of the results from Fig.(4.10) shows that the FFT has given an 84% increase in speed over the length 8 DFT run earlier. (from 0.984 seconds down to 0.156 seconds.)

The program first inputs the value of N being used and then determines if this is a valid number ie. a factor of 2. If this is alright then the data is read in and the internal timer started.

The FFT program starts on line 140 and is detailed in the flow chart Fig.(4.8)

The butterflies are calculated on lines 390 and 400 and then multiplied by the Twiddle factors on line 430.
Fig. (4.10). DIF FFT Flow Chart.

Start FFT Routine.

A = N
B = 2π / N

D = A
A = A / 2
E = 0

C = Cos(E)
S = Sin(E)

G + G + D

F = F + 1

Calculate Butterfly

Multiply by Twiddle Factor

Does G = N
Yes
E = E + B

No

Does F = A
Yes
B = B * 2

No

Does C = M
Yes

Stop Timer and find Time taken
Calculate Magnitude and Phase
Print results Real, Imaginary
Magnitude, Phase

....... END ......
Fig. (4.11). Basic Program For DIF FFT

10 REM RADIX 2 FFT PROG . LENGTH N
20 REM N = 2^M
30 INPUT "LENGTH N ": N
40 DIM AR(N),AI(N)
50 REM *** POWER OF 2 TEST ********
60 LET M=CINT(LOG(N)/LOG(2))
70 LET Y=2^M
80 IF Y = N THEN PRINT "MUST BE A POWER OF 2": GOTO 40
90 REM **** END OF RADIX 2 TEST *****
100 REM ***** READ DATA IN TO PROG ARRAY ****
110 FOR DAT=1 TO N
120 READ AR(DAT),AI(DAT)
130 NEXT DAT
140 REM ***** END OF DATA READ *****
145 V=TIMER
150 REM ***** FFT PROG START ********
160 LET A=N : LET B=2*3.142/N
170 FOR C=1 TO M
180 LET D=A : A=A/2 : E=0
190 FOR F = 1 TO A
200 C=COS(E) : S=SIN(E)
210 FOR G=D TO N STEP D
220 H=G-D+F : J=H+A
230 GOSUB 380 : REM BUTTERFLY
240 GOSUB 420 : REM TWIDDLE_FACTOR
250 NEXT G
260 E=E+B
270 NEXT F
280 B=B*2
290 NEXT C
293 REM ** END OF FFT *****
295 V=TIMER -V :PRINT "TIME TAKEN ":V;" SECS 
298 REM **** CALC MAG AND PHASE ****
300 FOR X=1 TO N
310 M(X)=SQR(AR(X)^2 + AI(X)^2)
315 IF AR(X)=0 THEN AR(X)=.0001 : O(X)=ATN(AI(X)/AR(X))
320 NEXT X
330 REM ** PRINT RESULTS ***
335 PRINT
340 PRINT TAB(2) :" N AR(N) AI(N) MAG PHASE"
342 PRINT
345 RESTORE 500
350 FOR X=1 TO N
355 READ Y :REM ** BIT REVERSAL ***
360 PRINT
365 TAB(2);Y;SPC(2);AR(Y+1);SPC(2);AI(Y+1);SPC(2);M(Y+1);SPC(2);O(Y+1)
370 NEXT X
375 END
380 REM **** BUTTERFLY ***
390 K=AR(H)-AR(J) : L=AI(H)-AI(J)
400 AR(H)=AR(H)+AR(J) : AI(H)=AI(H)+AI(J)
410 RETURN
420 REM *** TWIDDLE FACTOR ****
Digital Signal Processing. [Chapter 4]. page 96

430 AR(J)=C*K + S*L : AI(J)=C*L + S*K
440 RETURN
450 DATA 1,0.1,0.1,0.1,0,0.0,0,0,0,0,0
460 REM DATA FOR BIT REVERSAL
500 DATA 0,4,2,6,7,3,5,1

RUN

length N ? 8
TIME TAKEN .15625 SECS

<table>
<thead>
<tr>
<th>n</th>
<th>AR(n)</th>
<th>AI(n)</th>
<th>MAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>.9995084</td>
<td>2.41407</td>
<td>2.612805</td>
</tr>
<tr>
<td>4</td>
<td>.0001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1.000204</td>
<td>.4143577</td>
<td>1.082635</td>
</tr>
<tr>
<td>7</td>
<td>.0001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1.000084</td>
<td>-.4140697</td>
<td>1.082415</td>
</tr>
<tr>
<td>5</td>
<td>.0001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1.000204</td>
<td>-2.414358</td>
<td>2.613337</td>
</tr>
</tbody>
</table>

Ok

**Fig. (4.12)** Results for Length 8 DIF FFT.
The FFT Basic program outlined above was run on a BBC computer - using unity square wave data, from which the magnitude plot Fig. (4.13) was obtained using the "Printmaster" screen dump routine.

Note: An inspection of the results table shows that the DIF FFT gives nearly identical results to that obtained from the earlier DFT (see Fig. 4.4) if rounding errors are ignored and the N coefficients are rearranged into sequence. No phase angle was obtained but could easily be obtained as explained earlier on page 63.

It is possible to speed up the FFT if the data used is solely real as the complex part can be omitted from the calculations. Also as the spectrum will be symmetrical about the N/2 point only the first N/2 points need to be calculated. For this and other techniques such as zero padding, (1), (2), and (23).
CHAPTER V.

THE DIGITAL SIGNAL PROCESSING CHIP (DSP).

A. The Digital Signal Processor (DSP).

Signal Processing is the technique that allows electronic systems to relate to the real world, i.e. to enable them to feel, see, touch, taste, smell and hear.

Digital signal processing is a branch of signal processing that confines itself to data represented in digital form. The DSP chip is a unique device designed to function as a digital signal processor and perform computationally intensive mathematical operations very quickly.

The Digital Signal Processor - or DSP chip, itself is basically an application specific integrated circuit (ASIC) that has been designed to function both as a microprocessor and a fast array processor.

Some microprocessors - such as the 6502 from Rockwell, cannot perform multiplication directly, whilst others - like Intel's 8086, are very slow at multiplication. The 16 bit 8086 microprocessor takes 60 clock cycles to perform a 16 bit by 16 bit multiplication - which converts to 15 microseconds with a 4 MHz clock. The Texas TMS32010 DSP chip on the other hand only takes 300 nanoseconds for the same multiplication, a speed increase of 50 times.

This speed deficiency in the basic microprocessor has been overcome by the use of a maths coprocessors such as the 8087 and 80287. For mathematically intensive matrix multiplications, a dedicated array processor is used.

The Digital Signal Processor (DSP) was developed to meet this array processing requirement and at the same time to function as a fast microprocessor.

The DSP comes in various shapes and sizes but is usually
found as a 40 pin DIL (dual in line) package or as a 40 pin PGA (pin grid array) unit. It has its own onboard data memory and in some cases program memory also. Many of the newer DSP's also have on board Analog to Digital and Digital to Analog converters in addition to serial input/output ports and external data and address bus connectors.

A benchmark used for DSP's is MIPS (Million Instructions Per Second), or MFLOPS (Million Floating Point Instructions Per second). This benchmark takes into account reductions in instruction times brought about by advances in calculating techniques such as that developed by McNally, McCanny and Woods (15) and improvements in manufacturing and design processes. It is therefore a more useful indicator of the speed of the DSP than a straight forward clock speed. The MFLOPS speeds of DSP's is constantly increasing and is now up in the 40 MFLOPS range (16) and (21). At the same time - as more and more applications are found for the device, the cost is coming down.

The majority of DSP's manufactured today are of the CMOS type (Complimentary Metal Oxide Semiconductor) as this gives a lower power consumption than the NMOS (N-channel Metal Oxide Semi-conductor) types. However as chip speeds increase the power consumption of CMOS chips is set to rise considerably and as complexity increases and operating speeds exceed 50 MFLOPS per second power levels could reach 10 to 15 watts for the larger chips (25).

If a floating point DSP is used - as opposed to a fixed bit length DSP, then the quantization or rounding error, which results from using a finite word length device, can be reduced to negligible values.

As an example take a fixed 8 bit number arranged as;

[1 sign bit and a 7 bit wordlength]

and working between +/- 1, this can handle values down to +/- (1/128) th. (ie. +/- 0.00781282), whereas an 8 bit floating point number arranged as;

[1 sign bit, 2 exponent bits and 5 mantissa bits]
Can handle values down to \( +/-(1/32) \times 10^{-3} \) or \( +/-0.0003125 \). The exponent here always being negative.

A comparison of fixed and floating point "quantization noise" has been studied by Weinstein and Oppenheim (2).

B Digital Signal Processor Applications.

The range of applications found for the DSP are increasing, daily, some of the current areas are:

a. Frequency representation of single channel and dual channel time signals.

b. Implementation of Finite Element Response (FIR) and Infinite Impulse Response (IIR) digital signal filters.

c. Voice coding and decoding routines for telecommunications.

d. Speech recognition and synthesizing.

e. Ordinary and Adaptive control systems.

and f. Signal recovery.

a. Fast Fourier Transform (FFT)

Chapter 4 showed how the standard Fourier series used to represent a time dependent signal in terms of its sinusoidal frequency components can be modified to deal with sampled data signals (DFT), and how the resulting series could be further speeded up to yield the Fast Fourier transform (FFT).
A 1024 line real time FFT however requires 7,097 multiplications. On a small computer such as the BBC this would take about 317 seconds, whilst a DSP such as the TMS320C50 takes only 2.5 milliseconds. The "C" signifies a CMOS chip.

An example of a 256 line FFT applied to a sampled data low frequency audio signal was shown in Chapter 2, Fig.(2.1).

The FFT opens up the use of the DSP device for real time frequency analysis in applications such as machine condition monitoring, noise, vibration, speech and pattern recognition.

b. Signal Filtering.

Signal filtering is a common application of the DSP. It enables digital signal filters to be easily produced and which have clearly defined and repeatable characteristics.

The DSP can be programmed to be used as a stand alone filter for a specific application such as an anti-aliasing filter, which requires its cut off frequency to be variable over a wide range of frequencies. The useful bandwidth for these filters being governed primarily by the conversion time of the A/D converter used.

On the other hand the DSP can be incorporated into a piece of equipment for the production of frequency spectra such as Octave and Narrow Band analysers using a range of digital filters of differing centre band frequencies. These devices simulate real time frequency analysers which up until now have relied on a bank of analog filters.

Narrow band analysers can be constructed to give a good frequency resolution and do not suffer from the drawbacks found in FFT type analysers such as windowing and aliasing. The analysers can be either constant percentage bandwidth or constant bandwidth types and are useful for noise analysis where the prospect of litigation requires the filters to
Digital Signal Processing. [Chapter 5]. page. 102

conform to International Standards such as the ISO R266 (British Standard 3593 1963) "Preferred Frequencies for Acoustic Measurement".

DSP's can also be used to construct adaptive noise filters that are capable of tracking and eliminating noise which varies in frequency and would be useful in extracting signals from noisy backgrounds.

c. **Voice coding and decoding**

Voice coding and decoding is widely used in telephone systems. The public and private PBX telephone networks use pulse code modulation to represent analog speech. This digitized data has its amplitude represented in a logarithmic form (compressed) for ease of transmission, and is then reversed (expanded) at the receiving end. DSP's are ideally suited for the implementation of compressing and expanding algorithms.

d. **Speech Recognition and Synthesizing.**

Early attempts at speech recognition relied mainly on banks of analog filters to extract the spectral information. The introduction of the DSP has greatly improved speech recognition techniques and encouraged further work. The new techniques still use spectral information but now use the Fast Fourier Transform or digital filter methods. An example of this is the use of the TMS32010 DSP chip - the subject of this report, that as stated earlier is currently being used in a doll from Worlds of Wonder (12), having both speech synthesis and voice recognition.

e. **Controller functions.**

Controller functions can readily be implemented using a DSP. They offer savings in both cost, weight, speed and
reliability. They can be adapted to suit both single and multi-input multi-output systems as well as for floppy and hard disc drive controllers, as the DSP’s fast execution time and ability to handle advanced control algorithms reduces disc memory access time. They also lend themselves to the more demanding adaptive control systems.

f. Signal recovery.

An increasing area of application is in signal recovery. Seismic, sonar and satellite signals often contain useful information that is buried in the background noise. Frequency analysing techniques exist—using DSP’s that can be used to recover the hidden information such as the Power Cepstrum analysis where the further FFT of a power FFT spectrum is undertaken, this technique highlights—or extracts components in the signal that exhibit periodicity in the original power spectrum \(^{(27)}\).

By averaging frequency spectra, points of interest can often be detected that are not obvious in a single plot. Quite often the presence of background noise masks signals of interest but as noise in random in nature averaging generally removes its masking effect.

In addition, by comparing the signals obtained from two different paths any correlation—or commonality, present can be detected. This is useful when trying to determine the path a noise or vibration signal is taking by comparing the noise level obtained from the source with that obtained with the sensor in different locations. This correlation technique can also be used for range finding or thickness measurement. A signal is sent out and later received back. the phase—or time, lag between the transmission and reception is a measure of the distance travelled by the signal and hence the range or thickness. The phase lag being measured by means of cross correlation techniques which are ideally suited to the DSP. An example of this was its use in determining the distance of the
Mariner space probe from Earth (15).

Other applications are constantly been found for this versatile chip such as pattern recognition (for robotics) (Loughborough Sound Images Ltd.), and for image compression in video cameras (Fuji Film Co.).

C Digital Signal Processing Equipment.

Typical equipment categories are ;- 

i. PC based DSP expansion boards,  
ii. PC based external DSP units,  
iii. Frequency analysing instruments using DSP's and  
iv. as a component part of a larger device.

In the first category the DSP chip is mounted on an expansion card and fits into one of the free expansion slots inside a personal computer (PC). Typical of these is the Loughborough Sound Images TMS32020 SWDS board which as its number implies uses the Texas TMS32020 DSP chip, it costs around £1700. Also in the same category is the Amplicon DAP2400 board that uses a 20 MHz Motorola 56001 DSP and costs around £1600.

Both of these boards have built in 12 bit analog to digital converters on both channels and an input sampling rate of around 230,000 samples per second. They come with software to operate them together with additional software development tools for digital filters and FFT’s etc.

In the second category the DSP is housed in an external unit but connected to the host PC via a ribbon cable and an I/O expansion port card. Typical of these is the ETL300A - supplied by AB European Marketing, which is housed in a keyboard sized unit and offers two channel processing. The FFT length in the ETL300A is limited to the type of computer
being used. The software supplied is good but advances in technology are leaving this unit behind and today an expansion card device would be a better buy.

Also in this category is the C3M DSP unit from Ultra systems which is covered in this report. The unit only has provision for a single channel input but — unlike the ETL300 unit, is a teaching and development unit. It attaches to the PC in the same way as the ETL300 via a ribbon cable. As with the ETL300 it can be used with either the BBC or IBM computers. The C3M board uses a first generation Texas TMS32010 chip but later versions offer a choice of the TMS32020 or TMS320C25 chips. The board costs around £1500 and comes complete with development software.

The third group includes the purpose built frequency analysing instruments such as the Bruel & Kjear type 2034 FFT analyser (27), which is a 2 channel, 4000 line device costing around £19,000 and the single chanel 2033 analyser costing around £16,000. Both analysers have averaging, zoom facility and alternative windowing methods. The 2034 has a wide range of additional features such as cross-correlation, coherence and system response testing.

Others in this category are the Schulemerger type 1220 FFT analyser — costing about £13,000, which is also 2 channel. Together with the cheaper but still very competitive Advantest type R9211A, a 2 channel device packed with additional features costing around £9,000.

The final category covers DSP chips that are included into other larger pieces of equipment — OEM (Other Equipment Manufacturers), to perform the DSP function. This could be as front end filters in measurement equipment or as screen drivers in high resolution CAD systems. A recent application is a hand held vibration monitor having a 150 x 100 mm LCD display (37) that shows the results of a two channel 1600 line FFT analysis and which is available from Chase Electronics Ltd. — Model No PL202.
Digital signal processors use an advanced form of the Harvard architecture - using parallel processing, that can execute a 16 x 16 bit multiplication in one instruction cycle, giving around 10 Million Instructions per Second (MIPS).

The TMS32010 is a first generation DSP chip from Texas Instruments and this is reflected in its slow cycle time of 200 nsec compared to later generations 74 nsec for the Motorola DSP96002. The TMS32010 chip does not support any direct memory access or software polling and has only a small on board data RAM memory. It is however fairly easy to program and its simplicity is useful in a teaching environment.

Given the wide variety of Digital Signal Processing (DSP) devices currently on the market a selection of one from the many is not a straightforward task.

Some DSP comparisons are shown below in table (5.1).
Table (5.1) DSP Comparisons.

<table>
<thead>
<tr>
<th>Device</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorola DSP56000</td>
<td>Inter processor communication.</td>
</tr>
<tr>
<td>Texas TMS320C30</td>
<td>Floating point multiplication.</td>
</tr>
<tr>
<td>NEC MPD77230</td>
<td>Very low cost.</td>
</tr>
<tr>
<td>Texas TMS32010</td>
<td>On board EPROM memory</td>
</tr>
<tr>
<td>Texas TMS320E15/E17</td>
<td>Compatible to TMS32010</td>
</tr>
<tr>
<td>Texas TMS32020</td>
<td>Good hardware and software support.</td>
</tr>
<tr>
<td>Analog Devices</td>
<td>High level assembler.</td>
</tr>
<tr>
<td>ADSP2100</td>
<td>Very low power (0.3 W)</td>
</tr>
<tr>
<td>AT&amp;T WE DSP16</td>
<td>Fast cycle time (50ns.)</td>
</tr>
</tbody>
</table>

Typical of the factors that will influence the choice will be cost, speed of performance, on board memory size, fixed or floating point, interfacing requirements and upward compatibility should a more powerful device be available/or be required later.

With performance the only real test is how fast will it run a specific type of program - or benchmark. Typically this could be a 1024 line real time FFT. Together with this is how much memory has it used and how many programming lines have been used.

Table (5.2) is a compilation of data that has been extracted from manufacturers data sheets and lists the main features of the TMS32010 against some more recent DSP chips together with the timings for a 1024 line FFT :-
The time quoted in the last column of the above table is for a 1024 line single channel (real) FFT. This benchmark is a useful comparison of the speed of the relevant DSP but care should be exercised as the radix of the FFT is not quoted nor the length of the program used to determine it. The DSP from Analog devices uses only 300 lines to perform a 1024 line radix 4 FFT.

The availability of a particular instruction code may also be a determining factor. This could then reduce the amount of program coding used and speed up the program execution.

The requirements of peripheral devices used will also need to be considered especially any handshaking requirements. In addition the provision of serial interfacing may be a requirement that has to be considered.

For more information on the factors affecting DSP benchmarks and their selection see the article "Considerations for selecting a DSP processor" by Analog Devices (11) mentioned in Chapter 1. This article also compares the Analog Devices Digital signal processor -the ADSP 2100A, with the Texas Instruments TMS320C25. A CMOS variant of the TMS32010 but having a cycle time of 100 nanoseconds and 544K of on board memory.
A new breed of DSP chip is now available offering 32 bit floating point arithmetic - eg. the DSP32 from AT&T. As stated earlier in this chapter, fixed point numbers can soon overflow the register size but the wider dynamic range of floating point makes this less of a problem. These devices offer far greater dynamic range than their fixed point counterparts, together with increased resolution. In addition the operating code can be reduced and often stored on the chip thus speeding up the execution time.

In addition low cost compact DSPs having on board data and program RAM memory that can be "Booted up" from cheap EPROMs are expected to be available soon (ADSP 2101 from Analog Devices) that will enable a dedicated DSP unit to be constructed using only three or four chips.

**Di. The TMS32010 Digital Signal Processing Chip.**

The pin layout of the TMS32010 is shown Appendix B, "TMS32010 Programming Data" together with a diagram of the internal architecture.

The TMS32010 was the first NMOS digital signal processor and was introduced in 1983. Since then Texas instruments has introduced the CMOS variant the TMS320C10 whilst the TMS32017 variant is provided with a co-processor interface for interfacing to 8 and 16 bit computers. Texas instruments are now producing the TMS320C50 which has 17K of on board data RAM in a CMOS chip.

Three clock speeds are available on the TMS32010 of 16,20 and 25 Mhz.

It has 144 x 16 bit words of data RAM on board and can access 4k of external program memory.

The chip only requires a single 5 volt supply and is designed to interface with an EPROM carrying the program memory.

The device is fairly easy to program having only three addressing modes - direct, indirect and immediate, and 60
Digital Signal Processing. [Chapter 5]. page 110

instructions. See Appendix B "TMS32010 Programming Data", for information on the instruction set and the mnemonics used in the assembler.

A typical programming procedure would be to write the assembly language program on a word processor such as "View" on the BBC or "Word" on an IBM and then pass this through an assembler to compile the object code required for storage in the program memory. The C3M unit as well as the Loughborough and Texas Instruments development systems come supplied with monitor and simulator programs for monitoring the program whilst running and for simulating or single stepping through a program for fault finding.

To maintain the maximum degree of accuracy the data should be converted to Q15 format. As an example consider the number 0.126:-

\[
0.126 \times 2^{15} = 4128.768 \text{ on rounding gives 4129 and in Q15 form this gives the binary representation } \]

\[
4129 = 0 \ 001,0000,0010,0001
\]

sign bit

\[
\begin{array}{c}
1 \\
32 \\
4096 \\
4128
\end{array}
\]

Note :- that \(2^{15} = 32768\)

and that \(0.126 = 4129/32768\)
CHAPTER VI

THE C3M DSP UNIT.

The C3M Digital Signal Processing Unit.

The digital signal processing unit used as a basis for this report was the "C3M Development System" from Ultra Digital Systems, the commercial arm of Liverpool University.

The main features on the C3M board are:

1. TMS32010 digital signal processor running at 20 MHz
2. 4k x 16 bit program memory RAM
3. 4k x 16 bit monitor program in EPROM.
4. 12 bit A/D converter with sample and hold. Also includes automatic conversion to 2's compliment and extension to 16 bits.
5. 12 bit D/A converter.
6. Software programmable sampling rate up to 20kHz.
7. Switchable 4th order 3.5 kHz filters on input and output.
8. 16 bit digital input and output connections.
9. 8 bit parallel bidirectional link to host computer.

and 10. Processed / unprocessed switch for evaluation of program effects.
There are three software programs supplied with the unit. The first is a Cross Assembler used to compile the TMS32010 machine code programs from the assembly text program. The second is a Monitor/Debugger program to test the compiled program using single steps — or by the use of breakpoints. The third is a Simulator program to emulate the operation of the TMS32010 processor when running the compiled program.

The C3M board uses a Texas Instruments TMS32010 digital signal processor, running at 20MHz from an external crystal oscillator. The instruction cycle time of 200 nanoseconds gives an effective timing of 5 MIPS — or 5 million instructions per second.

The board has provision for a static program memory area of 4k — made up of four 1kx12 bit (6168) static RAM's. In addition there are two 4kx8 bit (2732) EPROM's for use by the monitor — or BOOT, program but they can also be used by the programmer to create stand alone applications. The data memory of 144x16 bits is provided by the TMS32010 chip itself.

A general view of the C3M board is shown in Fig.(6.1) giving its major dimensions along with the memory map of the C3M board which is shown in Fig.(6.2).

The board uses a 12 bit Analog to Digital (A/D) converter type AD574 giving a conversion time of 25 microseconds. This in fact limits the input signal frequency to 40 kHz — half the sampling frequency. The converter is configured to accept a 5 volt bipolar signal from a BNC socket on the face of the board. In addition there is a phono socket for direct connection of line signals from audio equipment and a high impedance (50 kohm) microphone socket.

A 16 bit digital input socket is provided for direct
Digital Signal Processing. [section 6].

**Fig 6.1 C3M Unit**

**Fig. 6.2 C3M Memory Map**
connection to the data bus on the board via a 20 way IDC connector. This can be used for applications such as Logic analysis. There is also provision for a digital output via the 20 way IDC connector.

A 12 bit Digital to Analog (D/A) converter type AD7545 is provided having a settling time of 500 nanoseconds. All 12 bit inputs from the A/D converter are sign extended to 15 bits using additional components in the converter circuit.

Communication to the host computer - either a BBC or IBM computer, is via a 20 way ribbon cable. The 16 bit data word can only be sent 8 bits at a time and two strobe lines are provided to assist with this data exchange. The two strobe lines are the Dout and the Din lines and tell when a valid data signal has been written to the data bus by either the C3M board or the host computer. When a data word is passed to the C3M board the Din line causes an interrupt to be generated in the TMS32010, this causes the program to jump to the interrupt vectors at FFC and FFD in program memory.

A monitor - or BOOT - program is loaded into the two EPROM's at start up or reset. The monitor program allows for seven commands to be used for data exchange when running the monitor or simulator software programs supplied on the host computer.

As stated two EPROM's are included on the main board and offer the user the facility to store their programs and data onto these EPROM's to turn the C3M unit into a stand alone system.

An interface card is provided for use with an IBM PC which slots into one of the PC's expansion sockets. The board has a 20 pin 'D' connector - of the same type as used on the BBC for the user port. The operating software being supplied on a floppy disc.

For the BBC the operating software is in two EPROM's which need to be inserted into the ROM sockets of the BBC.
Connections are made to the C3M unit via the BBC user port. The user is required to write a separate program to run on the host computer that can poll the DOUT pin of the C3M for a low pulse, which signifies that a valid data sample is present on the data lines.

Due to the speed differences between the host computer and the faster C3M board a delay routine needs to be incorporated into the data transfer program sending data from the C3M board to the host computer. The data transfer can be speeded up by sending a signal to the C3M board along the DIN line which will cause the C3M to jump to an interrupt handling routine. This will give an increase in data transfer speed but at the cost of writing the interrupt handling routine into the TMS32010 program.

If the FFT data is to be stored — for use in a waterfall diagram for e.g. — then an additional routine would need to be written to write this information into a storage file.
CHAPTER VII

PROGRAMMING.

a. INTRODUCTION.

Five programming languages are used in this section:
1. 6502 Assembly language
2. 8086 - 8088 - 80286 Assembly Language
3. TMS32010 Assembly language
4. BBC Basic High level Language.
and 5. GW Basic High level Language

The program file extensions show the type of file. The ASM extension being used for 8086/8088/80286 assembly language text files written on a word-processor such as WORD, then loaded into a less sophisticated Text Editor such as RPED to remove any formatting instructions included by WORD at the start.

The OBJ extension is used for compiled assembly language programs, assembled using Microsoft's MASM assembler. These files were then converted to executable files having an EXE file extension and finally to a common file having a COM extension using the DOS EXE2BIN and LINK programs respectively. A COM file always loads at the same place - at 100 hex, and is limited in length to 64K bytes.

At the start all the segment registers contain the same value which is that of the Program Sector Pointer. For more information on this procedure and on programming the 8086/80286 see references (30) and (31).

The IBM GWBasic programs were written directly into the GWBasic program and have a BAS file extension.
The GWBAsic programs need to know were the actual starting point of the assembly language data capture routines are in program memory - i.e. at what segment address and at what offset. This was done by writing an extra section to the assembly language routines to store their own starting
address together with the data length at a set point in memory. The "DS" data segment register was used to point to this memory location - i.e. act as a vector, by setting the "DS" segment register to zero and storing the information at a memory offset of 100 hex.

The assembly language programmes for the BBC (6502 processor) are contained within the Basic programs filed in the "O" directory. Use being made of the built in 6502 macro assembler to compile the assembly language program at run time. The BBC macro-assembler places the assembly language program within a Basic program and does not require a separate object code program to be written unless a stand alone routine is required.

The assembly language programs for the TMS32010 were written first in text using Acorn's VIEW word processor for the BBC and WORD for the IBM PC. This restricted the length of the program for the BBC to 9.5 Kbytes - or about four A4 pages, using Mode 3. This rules out using the BBC to program an FFT for the TMS32010 on the C3M board as it took nine A4 pages for a length 8 FFT. This could be reduced by omitting the comments and compressing the code, but for longer FFT's more coefficients are required and hence a longer program would result anyway.

It is however still possible to use the BBC with the C3M board either as a fast 16 bit data logger with a maximum sampling rate of 20 kHz or as a digital filter. Although digital filters are not covered in this report a length 9 Finite Impulse Response digital filter - having a cut off frequency of 2 kHz, has been included along with the other TMS32010 programs on the attached disc.

Programming the TMS32010 is basically the same as programming the above processors. The text files have SRC file extensions whilst the compiled - or assembled - programs have OBJ file extensions. The XASM program was used to compile the TMS32010 programs and is supplied with the
In the TMS32010 there are two 16 bit auxiliary or index registers (ARPO and ARP1), a 32 bit accumulator and a 32 bit multiplication register (T). The results of the multiplication are stored in the (P) register and usually need to be transferred to the accumulator after a multiplication process.

A pointer is used to tell the processor which of the two auxiliary registers are being used.

Once the TMS32010 assembly language program has been written on a word-processor or text Editor it will need to be compiled. The compilation is achieved using the supplied program "XASM" which produces an object code program with either an "OBJ" file extension or - in the case of the BBC - files the object program in the "O" directory.

A 2nd pass listing is also produced by the cross assembler, which details the locations and values supplied by the assembler during compilation. The 2nd pass listing have "LST" extensions on the PC or are in the "T" directory on the BBC.

All the programs listed contain detailed information on the instruction being used which should enable the newcomer to follow the programming method used. For more specific programming information on the TMS32010 see Appendix A and references (31), (32) and (33).

For more detailed instructions on the C3M board programs see the handbook supplied by Ultra Digital Systems (34).

Full program listings are given in appendix A.

There are five program sets altogether not including the previous two BBC Basic programs for the DFT and the FFT, these are detailed below: -
Digital Signal Processing. [Chapter 7]. page. 118

Set 1. Data Capture & Display Program for the BBC.

Comprising:-  C3M Board Data Capture & Transfer Program.

The BBC (Basic) data collection & display program.

This set of programs produce a time dependant display on the BBC host computer of the data present at the input of the C3M board. The data length is dependant on the number of samples taken and the sample rate selected on the C3M board.

Set 2. Data Capture & Display Program for the IBM PC

Comprising:-  C3M Board Data Capture & Transfer Program

IBM PC Data collection program.

The IBM PC (GW Basic) time display program.

This set of programs produce a time based display on the IBM PC host computer of the data present at the input of the C3M board. The data length is dependant on the number of samples taken and the sample rate selected on the C3M board.

Set 3. Length 8 DIF FFT (Test Program) for the IBM PC

Length 8 FFT for the C3M Board (stored Data)

IBM PC Data collection program.

The IBM PC (GW Basic) FFT display program.

This set of programs performs a length 8 DIF FFT on the data captured by the C3M board, transfers it to the host computer - the IBM PC - and displays the result on the screen.
Set 4. Length 128 DIF FFT for the IBM PC

Comprising:
- Length 128 FFT for the C3M Board (Real Time)
- IBM PC Data collection program.
- The IBM PC (GW Basic) FFT display program.

This set of programs performs a length 128 DIF FFT on the data captured by the C3M board, transfers it to the host computer – the IBM PC – and displays the result on the screen – the sample rate being 20 kHz.

Set 5. Length 128 DIF FFT for the IBM PC

Comprising:
- As set 4 but using a sampling rate of 5 KHz instead of 20 KHz.

This set of programs performs a length 128 DIF FFT on the data captured by the C3M board, transfers it to the host computer – the IBM PC – and displays the result on the screen – the sample rate being 5 kHz.

b. PROGRAM SETS for the BBC and the IBM PC.

PROGRAM SETS 1 & 2. Data Capture & Display Programs using the BBC and the IBM PC.

Before a routine run on the C3M board can be displayed on the BBC or IBM PC it has first to be transmitted from the
Digital Signal Processing. [Chapter 7]. page 120

C3M board to the host computer. To do this the units must be able to communicate with each other and should be able to transfer data between themselves quickly, and ideally with zero transmission errors.

Program Sets 1 and 2 were written to try and meet the above requirement, that is to use the C3M unit to take in a time dependant signal, transfer it to the host computer and have the host computer display the result in graphical form with minimum error.

Once the data transfer and display routines are in place and working an FFT program can be written for use in the C3M unit. The results from the FFT can then be transferred to the host computer for display using these routines.

. PROGRAM SET 3. Length 8 DIF FFT for the IBM PC.

To test the FFT procedure a length 8 decimation in frequency (DIF) FFT was used, which used as its input the same square wave data used in the previous BASIC FFT program. The data for this being stored at the end of the TMS32010 FFT program in the C3M unit. This is accomplished in Set 3.

. PROGRAM SET 4. Length 128 DIF FFT for the IBM PC. 20 kHz Sample Rate.

Finally a length 128 DIF FFT was written and stored in the
Digital Signal Processing. [Chapter 7]. page 121

C3M unit. Real time data being sampled by the on board A/D converter, processed by the C3M board, transferred to the host computer (the IBM PC) and displayed in graphical form. This is done in Set 4.

```
.PROGRAM SET 5. Length 128 DIF FFT for the IBM PC. 5 kHz Sample Rate.
```

Set 5 is similar to Set 4 but uses the slower sampling rate of 5 kHz instead of the 20 kHz used in Set 4. This reduces the available bandwidth of the signal to be analysed.

c. Batch Files for the IBM PC.

The programs for sets 2, 3, 4 and 5 have been put on a 5.25 floppy disc (360K). In addition four Batch programs have been added to enable the user to type in only a single command to run all programs in the set, they are:

1. (Set 2) . Type in. C3MTIME.BAT
2. (Set 3) . Type in. FFT_TST.BAT
3. (Set 4) . Type in. FFT20128.BAT : (20 kHz)
and 4. (Set 5) . Type in. FFT5_128.BAT : (5 kHz)
c. PROGRAM SET 1

1.1 Set 1 BBC Time display.

This first program "TRANBBC" is the core for all subsequent TMS32010 assembly language programs. The program inputs the (N) real time data values from port 1 on the C3M board - via the internal A/D converter, and stores it in data memory in Q15 format. Negative numbers are stored in 2's complement form and sign extended to 16 bits.

The program has been padded out with text to aid those unfamiliar with TMS32010 assembly language mnemonics.

The program loads and runs from the C3M board address 200 hex. The first nine data memory location are used to store the constants used in the program and the first part of the program reads in these values from the program memory and stores in data memory on the TMS32010.

The second part reads in the data values from the C3M input port and stores in data memory on the TMS32010.

The third part sends two start bytes to the BBC (0A.0A) with a delay between each byte - to enable the much slower BEEB to keep up. (The TMS32010 is over 25 times faster).

The fourth part sends the 16 bit data values - 8 bits at a time (Low then High). to the BEEB. Again a delay is used between bytes.

The sample rate is software set to 4kHz by sending the code 6 to output port 2 on the C3M unit which is used to set an on board counter connected to the sample and hold circuit in the A/D converter.

The DOUT strobe line from the C3M unit is sent to the BBC when data is available. Following the delay period, the next data byte is sent. This delay, send, delay, send routine is continued until all 100 data words - 200 data
bytes, have been sent. The program then returns to the start and captures another data set from the input port.

The second program is "C3MGRA", a BBC BASIC program that waits for an output pulse from the C3M board and then catches 100 of the 16 bit data bytes from the C3M board. The data is sent to the screen of the BBC and displayed in graphical form.

The program checks to see if the data block is OK by comparing the first two bytes - which should both be 0A0A. The data is stored in location &2E00 upwards. Data transfer is via the 6522 VIA port - the user port. The program polls the interrupt flag on the 6522 which is set when a pulse is received on the CB2 line from the DOUT line of the C3M board.

The delay - both in the C3M program and in the BBC program, needs to be adjusted to ensure that the slower BBC has time to read the data byte before it changes and not too slow that the BBC reads the same data byte twice.

1.2 Typical Set 1 display.

A typical BBC screen plot of the set 1 programs is shown in Fig. (7.1) below first for a 500 Hz, 5 volt square wave and then in Fig. (7.2) for a 250 Hz sine wave. Both signals were sampled at 4 KHz on the C3M board. The displays were obtained using the graphics screen dump routine from "Printmaster".
There is a lack of symmetry showing in the square wave probably due to the sample frequency not being an even factor of the signal frequency, and the lack of any triggering facility to ensure the same signal start point is used every time. This is not so noticeable on the slower
250 Hz sine wave.

1.3 Loading & Running the Set 1 Programs.

The programs used are on the 40 track 5.25 BBC disc - on directory "O" and are:

1. O.TRANBBC  the C3M programs for the TMS32010
and 2. O.C3MGRA  the BBC 'Basic display program

The procedure for running the first two programs (set 1) on the BBC and C3M board is as follows:

1. Test the C3M board by loading and running the
   *MON program - stored on the eprom in the BBC, as follows:

      *MON
   A message will tell you if the board is not responding.

   Return to basic by typing *BASIC at the screen prompt.

2. Load and run the object program on the C3M board
   using the command:

      *SEND O.TRANBBC

3. Load the Basic program C3MGRA into the BBC using
   the command:

      LOAD "O.C3MGRA"

4. RUN the Basic program.

The BBC screen should show a plot of the input signal amplitude against "time" - actually the sample number.

The frequency of the input signal is limited by the sample frequency of 20 KHz to around 10 kHz - Shannon's Sampling Law.
c. PROGRAM SET 2

2.1 Set 2 IBM PC Time display.

The Set 2 programs extend the data capture and display routines developed in Set 1 to the IBM PC. The method of memory segment addressing used in Intel processors increases the complexity of both the programming and the data transfer over that used in the BBC. As a result three programs are required for use with the IBM PC over the two used with the BBC.

The three programs being:

1. a TMS32010 data capture and transfer program, (C3MTIME.OBJ)

2. an assembly language data capture program for the IBM PC. (IBMTIME.COM)

and 3. a GW Basic display program, for use in the IBM PC. (PLOTTME.BAS)

The TMS32010 does not support any form of software polling - or handshaking, other than the use of an interrupt routine. A factor remedied in its successor the TMS32020, which has a 'data ready' line that can be used for handshaking during data exchange in addition to three levels of interrupt. It was therefore decided to use two way Interrupt Polling between the C3M unit and the IBM PC instead of the one way system used with the BBC. To meet this an interrupt handling routine was required for the TMS32010 as well as the one used in the IBM PC. This was done both to speed up the data transfer, as well as to try and reduce the data transmission errors to zero.

The need to keep the time between screen updates as small as possible - so as to approach a "continuous" display, ruled out the inclusion of an error correcting routine.
The first of the Set 2 programs is "C3MTIME.SRC". This is the TMS32010 assembly language program that takes in 128 - ie (N); real time data signals from port 1. Following this it sends a start code to the IBM PC (0A0A) followed by the (N) 16 bit data bytes. The data is sent 8 bits at a time in Q15 format, with the negative numbers in 2's complement form and sign extended to 16 bits.

For example if the input was (-0.126) then:

\[-0.126 \times 2^{15} = -4129 \text{ (on rounding)}\]

and this gives the 2's complement binary representation:

\[-4129 = \begin{array}{c}
1 \\
110 \\
1101 \\
1111
\end{array} \text{ in 12 bits}

\text{sign bit}

\[\begin{array}{c}
1 \\
111 \\
1110 \\
1101 \\
1111
\end{array} \text{ sign extended to 16 bits.}\]

This data handling program "C3MTIME" is similar to that used with the BBC (TRANBBC) but as stated earlier an interrupt handling routine has been incorporated.

A delay routine is still required for the data transmission but is of a smaller duration than that used in the BBC as the Intel processors run at a faster clock rate.

The second program (IBMTIME.COM) is an 8086/80286 program that loads at 100 hex and stays resident in the PC's memory. There are two parts to this program. One part that loads the program into the PC's memory and stores the start addresss of the second part at a memory location pointed to by the Data Segment register. The other that waits for an output pulse from the C3M board on the DOUT line - to port 371H, and then reads in the 256 - ie. 2N, data bytes from port 370H.

The program checks to see if the first two bytes are the start code (0A0A), before reading in the N 16 bit data values - ie. 2N x 8 bit data bytes. The data and the start codes being stored in memory.

When first called the program loads into memory and stores
its start address (Offset and Segment locations) along with the N value in the DS data segment starting at segment location 0000 and offset 100H. This is in keeping with the requirements for a COM program. The calling program can then read the segment and offset addresses and use a CALL routine to jump to this location. This program sets aside 256 words for the storage of the 2x128 data bytes.

The C3M supplied monitor program ATMON has a program called VIEW. This allows the user to use the Monitor program to view the output from the C3M board on the screen of the IBM PC. This is useful for checking the data values being sent from the C3M unit and as an aid to debugging the IBM TIME.COM program.

The third program is a GW Basic program (PLOTTME.BAS) that first looks at the data segment (DS 0000H) and Offset (100H) to find the start address for the C3MTIME program along with the N value being used. It then uses a CALL routine to run this program, input the 128 data values from the C3M board and store them in the PC's memory. The values are then converted to denary form as positive or negative numbers and displayed on the screen in graphical form.

2.2 Typical Set 2 display.

A typical display resulting from the above programs is shown below Fig. (7.3). Here a 3 kHz square wave has been sampled at 20 kHz to obtain the 128 data samples. An inspection of the plot Fig. (7.3) shows that six samples have been obtained for each cycle showing that the actual sampling rate is nearer to 18 kHz - i.e. an equivalent analogue to digital conversion time of 55 microseconds.
Fig. (7.3) A 3 KHz Sampled Data. Square wave "Time" Signal.
2.3 Loading & Running the Set 2 Programs.

The programs used are on the 40 track 5.25 disc – on the C3M directory and are:

1. C3MTIME.OBJ the C3M programs for the TMS32010
2. IBMTIME.COM the IBM PC data transfer program
3. PLOTTME.BAS the GWbasic display program

The procedure for running the Set 2 programs on the C3M board and IBM PC is as follows:

1. Load and run the batch program C3MTIME.BAT

A message will tell you if the board is not responding.

To return to DOS after stopping the program – by typing 'S' – type 'SYSTEM' and press enter at the screen prompt.

The IBM PC screen should show a plot of the input signal amplitude against "time" – actually the sample number. The frequency of the input signal is limited by the sample frequency of 20 KHz to around 10 kHz – Shannon's Sampling Law.

d. PROGRAM SET 3

1.1 Set 3 IBM PC Length 8 DIF FFT.

Program Set 3 is the first of the TMS32010 FFT programs. It was designed to be used as a test program for the operation of the TMS32010 FFT routine and the display and transfer
programs. The test being achieved by comparing the results to those obtained earlier using the BASIC FFT program.

The TMS32010 program itself is a length 8 decimation in frequency FFT program that uses the same Cooley-Tukey routine and simple square wave data that was used in the earlier BASIC FFT program. The data being stored at the end of the TMS32010 program and read in as required.

As this is a decimation in frequency FFT the resulting FFT data has to be bit reversed before being stored in memory ready for output to the IBM PC.

The Twiddle Factors (TF) used in the FFT are added to the end of the program together with the two small formulae below, to show how they are calculated:

The Sine components are found from:

\[ TF = \sin(2\pi n/8) \] for \( n = 0 \) to \( 1 \)

and the Cosine components are found from:

\[ TF = \cos(2\pi n/8) \] for \( n = 2 \) to \( 6 \)

An inspection of the DIF FFT butterfly diagram from Fig. 4.7 shows that only the Twiddle Factors \( W^0, W^1, W^2 \) and \( W^4 \) are required. These relate to:
Digital Signal Processing. [Chapter 7]. page 132

\[
\begin{align*}
W_0 &= \cos(2\pi 0/8) + \sin(2\pi 0/8) \\
W_1 &= \cos(2\pi 1/8) + \sin(2\pi 1/8) \\
W_2 &= \cos(2\pi 2/8) + \sin(2\pi 2/8) \\
W_3 &= \cos(2\pi 3/8) + \sin(2\pi 3/8) \\
W_4 &= \cos(2\pi 4/8) + \sin(2\pi 4/8)
\end{align*}
\]

But \( \cos(2\pi 0/8) = \sin(2\pi 2/8) \)

For a length 8 FFT the TF coefficient table size would normally have 8 elements. By arranging the TF's so that the 4 cosines follow the first two sines reduces the memory requirements as the remaining 4 Sine TF's can be read in from the Cosine TF's for \( n=2 \) to \( 6 \).

At the end of the FFT and after the bit reversal routine the program goes into a loop. During the time it is in the loop it repeatedly outputs the start code OAOA, whilst waiting for an interrupt signal from the IBM PC to tell it that an FFT can be sent.

To avoid overflow of the 16 bit signed data each FFT butterfly stage is scaled down by a factor of 2 giving a reduction of 8 overall. (ie. \( 1/N \)).

Using a length 8 FFT together with the known results from Fig.4.10 enables the operation of the FFT program to be checked and any faults rectified.

When reviewing the intermediate stages of this FFT it should be remembered that the data is in Q15 format - and 2's complimented for negative numbers.

If the C3M Simulator program (SIM) is used it is possible to single step through the TMS32010 program to observe its working.

The FFT8IBM program is basically the same as that for the Set 2 IBM transfer program (IBMTIME.COM). The main difference being in the amount of data storage required, which is now 16 bytes for the length 8 FFT.
Note C3M wordlength is 16 bits.

The GWBasic program -(GRF8TST.BAS), displays the result of the length 8 FFT in graphical form on the IBM PC. Subroutines are used throughout to simplify the program structure.

No window dressing has been applied to the display, the philosophy being to keep it simple. However both Real and Imaginary as well as Magnitude only plots can be viewed by pressing the "C" key and the signal can be averaged by pressing the "A" key. Scaling is used to keep the FFT values to full size for the screen display.

If a print out of the results was required an additional print routine could be added to extract the values from the program memory arrays.

This display routine is used for all the FFT plots, the only change being in the value of N being used.

2.5 Typical Set 3 Display.

A typical set 3 display is shown below in Fig. (7.4), for a length 8 DIF FFT of a square wave. This is the same signal used in the earlier BASIC DFT and FFT and hence gives the same result.

The screen display was captured using the screen CAPTURE.COM program supplied with WORD whilst the program FFT8TST.BAS was been run on an IBM PC286.
2.5 Loading & Running the Set 3 Programs.

The programs are:

1. FFT8TST  
   The C3M program for the TMS32010.
2. FFT8IBM.COM  
   The IBM assembly program.
3. GRF8TST.BAS  
   The Basic display program.

These programs can be run from the batch program FFT_TST.BAT.

e. PROGRAM SET 4

4.1 Set 4 Length 128 FFT.

The programs in Set 4 represent attainment of the original objective to write a 128 line FFT based on the TMS32010 DSP chip that could be displayed on an IBM PC.
The Set 4 programs runs the FFT with a sampling rate for the A/D converter of 20 kHz.

An additional program set has been included that runs with a 5 kHz sampling rate. Otherwise the programs are the same.

The program -(FFT20128), is the main TMS32010 program and as such draws together all the work done in the previous TMS32010 programming sections.

The FFT and bit reversal routine used in this program and in the length 8 FFT are based on a length 128 DIF FFT given by Burrus and Parks in their book DFT/FFT and Convolution Algorithms - pages 156 to 160. (1).

The major differences between the Burrus & Parks FFT and the one used here is in the method of data input and storage as the original program obtained its data from EPROMS whilst here the data is coming from an A/D converter, in addition a different sampling rate set-up was used.

The program inputs data from an A/D converter and stores it in data memory onboard the TMS32010, it then uses this same data area to store the interim values and the final bit reversed FFT output values.

In contrast the original program by Burrus & Parks used a memory interchange routine to transfer data between the TMS32010 and external EPROM data memory.

The FFT program follows the same pattern as shown in the flow chart for the BASIC FFT in Fig.4.8.

The TMS320120 programs for Set 4 are very similar to that for Set 3 - the length 8 FFT. The main differences are:-

1. The length of the Twiddle Factor coefficient tables at the end of the program.

2. Real time data is read in from the Analog input port on the C3M unit and not from program memory.
3. The sampling rate value sent to the A/D converter circuit is different in each case. Being 20 KHz for set 4 and 5 kHz for Set 5.

4. The larger program and data memory area required.

Whilst the program -(FFT3128.COM), is basically the same as that for the Set 2 data transfer program (IBMTIME.COM) it differs in the amount of memory set aside - 256 bytes, and the N or loop counter value which is now 128 instead of 8. The changes made to the constant table at the start of the program are shown in the program listings - APPENDIX B.

The program -(FFTGRF.BAS), is similar to that for Set 3 with the exception of the N value which is now 128 and for minor changes made to the screen display to accommodate the change in the FFT number.

4.4 Typical Set 4 displays.

The display resulting from a 128 line FFT on a 7 kHz Sine wave sampled at 20 kHz is shown below. Fig.(7.5) shows the Magnitude plot and Fig.(7.6) the Real and Imaginary plots :
7 kHz Sine Wave (sample rate 20 kHz)
Fig. (7.5) Magnitude Only

Fig. (7.6) A 7 kHz Sine Wave Real & Imaginary Plots.
Again both these plots were obtained using the screen capture program CAPTURE - supplied with WORD, to obtain a screen dump of the output from the GWbasic program FFTGRAF.BAS running on an IBM PC286.

The vertical axis gives the magnitude - scaled by $1/N$ - whilst the horizontal axes shows the harmonics of the fundamental frequency - $1/$Sample period.

The Basic program has the option of viewing the output either in Magnitude form or in Real and Imaginary form.

4.4 Loading & Running the Set 4 Programs.

Both sets 4 and 5 run a length 128 DIF FFT using real time data input to the C3M board and displayed on the screen of the IBM PC in either Real and Imaginary form or as a Magnitude only plot.

Set 4 runs with the sample rate of the on board Analog to Digital Converter (ADC) running at 20 kHz. This is software programmable from within the TMS32010 program.

Set 5 runs with the onboard ADC running at 5 kHz.

The programs are:

Set 4.....

1. C320FFT
2. FFT3128.COM
3. FFTGRAF.BAS

For the 20 kHz sample rate in the C3M unit.
For the IBM PC.
For the Basic display.

and run using the batch program FFT20128.BAT.
Set 5 ....

1. C305FFT For the 5 kHz sample rate in the C3M.

2. and

3. Are as for set 4

and run using the batch program FFT5_128.BAT.
CHAPTER VIII

SUMMARY FINDINGS

General Conclusions.

The two aims of the exercise were:

1. To produce a coherent analysis of the theory associated with the Fast Fourier Transform (FFT),

and 2. To write, develop and implement a 128 line FFT using the C3M development unit together with either a BBC or IBM Personal Computer.

The first part is covered in chapters 3 and 4 of this report and takes the reader through from the basic Fourier Series to the implementation of a Radix 2 Cooley-Tukey FFT.

The second part dealing with the practical implementation of programming a length 128 FFT to run on the C3M unit is covered in chapter 7 of this report.

Production of 128 line FFT running on the C3M unit and outputting to the BBC proved to be impractical - due to the memory limitations of the BBC - and was not pursued. Although it may be possible if the program lines were condensed, the comments removed and the twiddle factors calculated each time.

Additional chapters 5 and 6 cover the C3M unit and the DSP chip respectively. Whilst chapters 1 and 2 cover the general introduction, historical review and an introduction to frequency representation.

The length 128 FFT program developed for the C3M unit and
the IBM PC worked well. The FFT transform speed, screen resolution and screen update rate proved to be adequate for frequencies up to 18 KHz.

The length 128 FFT program could be extended to give a 1024 line FFT if the data and Twiddle Factors could be taken off chip and read in from program memory - although this would be a lot slower. Alternatively another processor could be used such as the TMS32020 which has a larger data memory - 544 by 16 bit data memory. (see Ref. (35)).

The maximum signal frequency of 18 kHz that can be analysed using the C3M board and the 128 line FFT program was mainly determined by the conversion time of the A/D converter used on the C3M board (55 microseconds). A faster converter such as the 14 bit AD9014 from Analog Devices which has a conversion time of 0.1 microseconds (10 Million samples per second) would give an upper frequency - or bandwidth, of 5 MHz.

The price of the DSP chip continues to fall and its speed to increase, greatly increasing its range of applications. Meanwhile a number of integrated DSP's having onboard A/D and D/A converters are appearing on the market such as the ADSP-21msp50 - again from Analog. Meanwhile Burr-Brown have produced a dedicated dual channel D/A converter for interfacing directly to a DSP - (DSP202).

Thanks to the Digital Signal Processor the technology to create an intelligent machine which can undertake its own health checks, suggest remedial action and communicate the results is fast approaching.
Programming Conclusions.

Developing and debugging the programs took up most of the programming time and was made more difficult by the lack of any handshaking lines to aid data transfer between the C3M board and the host computer.

Five different areas needed to be programmed, each using a different language:

1. The TMS32010 Assembly language for the C3M board,
2. the 8086/80286 Assembly language for the IBM PC,
3. the 6502 Assembly language program in the BBC,
4. the BBC BASIC in the BBC,
and 5. the GWBASIC in the IBM PC.

The major problem encountered during programming was to ensure the integrity of the data transfer between the C3M and either the BBC or the IBM. For the BBC this required the data exchange to be synchronised so that data bits were not gained by reading the same bit twice or lost by being too slow.

The result of the above was that a large time delay had to be incorporated into the TMS32010 program. This gave the slower BBC time to process the previous data byte and return to watch the Dout pulse from the TMS32010, which informed it a new data byte was being sent.

With the IBM an interrupt routine was used but even here a delay in the TMS32010 is still needed - although a smaller one. In addition a small delay had to be incorporated into the 80286 assembly language program to give the data time to latch into the input port after the Dout pulse was received.

Neither of these delays affected the signal sampling speed but did alter the rate at which the screen display was updated. For the 128 line FFT the screen was updated about
once every second, which for demonstration purposes and for periodic signals is satisfactory but for non-periodic (aperiodic) and random signals as it would be too slow. The 8 bit data transfer limitation of the C3M board is the major factor preventing a faster screen update rate although for single channel signals the use of samples up to \( N/2 \) instead of \( N \) would speed this up.

If a dedicated DSP was constructed mounted on an expansion card and plugged straight into the expansion bus of the PC a very fast screen update rate could be expected. This is the method currently adopted by most DSP board manufacturers.

The necessity of using an Anti-aliasing filter — at the input of the A/D converter, to screen out all signal frequencies above half the sampling frequency has already been mentioned — section 3.6-1 part V, (page 57). The two screen printouts shown in Fig.8.1 and Fig. 8.2 show a 128 line FFT of a 500 Hz square wave sampled at 5 KHz. Fig.8.1 shows the result without the aliasing filter and Fig.(8.2) with the filter and show the clean-up produced in the FFT by using an anti-aliasing filter. The filter cut off frequency was 3.5 KHz, which is a little high for a 5 KHz sample rate.

In order to show that the FFT results obtained from the C3M based system are correct an actual print out from a Brüel & Kjær type 2033, 4000 line FFT analyser is shown in Fig.8.3. for the same 500 Hz square wave signal.

The analyser sample rate was 5 kHz and the anti-aliasing filter automatically set to 1.56 times the baseband frequency — i.e. 7.8 KHz.

It can be seen that there is a good correlation between Figs.8.2 and 8.3. The first two spectral lines at 500 and 1500 Hz — the first and third harmonics — are clearly shown on both plots and the start of the 2500 Hz spectral line is starting to show in Fig's. (8.1) and (8.2). The analyser spectrum in Fig. (8.3) also shows up the presence of the even harmonics 2, 4, 6, etc. — although of reduced but constant amplitude — which are not present in the C3M board.
analysis Fig's (8.1) and (8.3).

The C3M board has a 3.5 kHz low pass filter fitted to it which can be switched in or out as required for use as an anti-aliasing filter.

128 Line FFT - Sample rate 5kHz.
Fig. (8.1) Without Anti-aliasing Filter.
In conclusion it must be said that the C3M board could prove very useful in a higher educational establishment for investigating digital signal processing. Although not covered here the unit also worked extremely well when running digital filters.

The fact that the C3M board can be used for either the BBC or the IBM computer is a plus feature. To which must be added its ability to create stand alone programs using one or both of its onboard EPROMs. The ability to monitor logic signals whilst running is also another plus feature.

Theoretical Conclusions.

Since this report was started a number of new texts have come onto the market dealing with the FFT and pulling together large amounts of the theory. (20), (36).

The initial work of pulling together the Fourier Transform...
Full Scale Level: 120 dB
Freq. Range: 5 kHz
Weighting: HANNING
Average Mode: ON
No of Spectra: 10
Comments:
500 Hz SQUARE WAVE

**Fig. 8.3**

Record No: 1
Date: Feb 23rd 1992
Sign: A.J.S.
theory progressed well and the DFT was reasonably straightforward. However the FFT and in particular the FFT butterflies proved a major problem. Large gaps in the available texts were mainly to blame. Progress was eventually made using the paper by Cooley and Tukey (4), together with the texts by Burrus and Parks (1) and Marple (17).

The final result is a complete development of the FFT from the Fourier series to the Radix 2 FFT.

Summary.

The theoretical development together with the programming flow charts and programs developed here - together with the data and references - should be of help to others working in this area both inside and outside the Institute.

By extending the work to include the C3M development system it has been shown that this unit is capable of performing a 128 line FFT in real time.

The techniques developed here should be of use to others developing programs for DSP's and interfacing DSP's to PC's.

In closing it is hoped that the contents of this report will be of use to past and present staff and students.

AJS. 11/5/92

Cardiff Institute of Higher Education.
APPENDIX A

DATA ON TMS32010 DSP CHIP.

Additional data on the TMS32010 Digital Signal Processor follows. The data has been drawn from references (32) and (33) and thanks are due to the UK office of Texas Instruments at Bedford for permission to include this data.

For those wishing to build a prototype DSP board using the Texas TMS32020 chip see reference (2).
FIGURE 2-1 - BLOCK DIAGRAM OF THE TMS32010

NOTE
ACC = Accumulator
ARP = Auxiliary register pair 0
ARD = Auxiliary register 0
ART = Auxiliary register 1
DP = Data page pointer
PC = Program counter
P = P Register
T = T Register

Digital Signal Processing. [Appendix A]. page 148
2. ARCHITECTURE

The TMS320 family utilizes a modified Harvard architecture for speed and flexibility (see Figure 2-1). In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting a full overlap of the instruction fetch and execution. The TMS320 family’s modification of the Harvard architecture allows transfers between program and data spaces, thereby increasing the flexibility of the device. This modification permits coefficients stored in program memory to be read into the RAM, eliminating the need for a separate coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

The TMS32010 utilizes hardware to implement functions that other processors typically perform in software. For example, the TMS32010 contains a hardware multiplier to perform a multiplication in a single 200-ns cycle. There is also a hardware barrel shifter for shifting data on its way into the ALU. Finally, extra hardware has been included so that the auxiliary registers, which provide indirect data RAM addresses, can be configured in an autoincrement/decrement mode for single-cycle manipulation of data tables. This hardware-intensive approach gives the design engineer the type of power previously unavailable on a single chip.

2.1 ARCHITECTURAL OVERVIEW

Among the elements the TMS32010 microcomputers combine onto a single chip are a volatile 144 X 16-word read/write data memory, a non-volatile 1536 X 16-word program memory (TMS320M10 only), a double-precision 32-bit ALU/accumulator, a fast 200-ns multiplier, a barrel shifter for shifting data memory words into the ALU, a shifter that shifts the accumulator into the data RAM, a 16-bit data bus which can fetch instruction words from off-chip at full speed, a 4 X 12-bit stack that allows context switching, autoincrementing/decrementing registers used for indirect data addressing and loop counting, a single-vectored interrupt, and an on-chip oscillator. This section provides a description of these elements.

The generic term ‘TMS32010’ is used to refer collectively to the TMS32010 and TMS320M10.
## TABLE 3-1 - INSTRUCTION SYMBOLS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>Accumulator</td>
</tr>
<tr>
<td>AR0</td>
<td>Auxiliary register zero</td>
</tr>
<tr>
<td>AR1</td>
<td>Auxiliary register one</td>
</tr>
<tr>
<td>ARP</td>
<td>Auxiliary register pointer</td>
</tr>
<tr>
<td>D</td>
<td>Data memory address field</td>
</tr>
<tr>
<td>DATn</td>
<td>Label assigned to data memory location n</td>
</tr>
<tr>
<td>dma</td>
<td>Data memory address</td>
</tr>
<tr>
<td>DP</td>
<td>Data page pointer</td>
</tr>
<tr>
<td>I</td>
<td>Addressing mode bit</td>
</tr>
<tr>
<td>K</td>
<td>Immediate operand field</td>
</tr>
<tr>
<td>(N)</td>
<td>Contents of register &quot;N&quot; or data memory location &quot;N&quot;</td>
</tr>
<tr>
<td>&gt; nn</td>
<td>Indicates nn is a hexadecimal number. All others are assumed to be decimal values.</td>
</tr>
<tr>
<td>P</td>
<td>Product (P) register</td>
</tr>
<tr>
<td>PAn</td>
<td>Port address N (PA0 through PA7 are predefined assembler symbols equal to 0 through 7, respectively)</td>
</tr>
<tr>
<td>PC</td>
<td>Program counter</td>
</tr>
<tr>
<td>pma</td>
<td>Program memory address</td>
</tr>
<tr>
<td>PRGn</td>
<td>Label assigned to program memory location n</td>
</tr>
<tr>
<td>R</td>
<td>1-bit operand field specifying auxiliary register</td>
</tr>
<tr>
<td>S</td>
<td>4-bit left-shift code</td>
</tr>
<tr>
<td>T</td>
<td>T register</td>
</tr>
<tr>
<td>X</td>
<td>3-bit accumulator left-shift field</td>
</tr>
<tr>
<td>→</td>
<td>Is assigned to</td>
</tr>
</tbody>
</table>
TMS32010 HARDWARE SUMMARY

<table>
<thead>
<tr>
<th>COMPONENTS</th>
<th>NAME</th>
<th>SYMBOL</th>
<th>SIZE (bits)</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>SYMBOL</td>
<td>SIZE</td>
<td>DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>Program ROM</td>
<td>-</td>
<td>1536 X 16</td>
<td>On-chip masked ROM containing program code.</td>
<td></td>
</tr>
<tr>
<td>Program Counter</td>
<td>PC</td>
<td>12</td>
<td>Register containing current address of program memory.</td>
<td></td>
</tr>
<tr>
<td>Stack</td>
<td>-</td>
<td>4 X 12</td>
<td>Four 12-bit registers for saving program counter contents during subroutine and interrupt calls.</td>
<td></td>
</tr>
<tr>
<td>Data RAM</td>
<td>-</td>
<td>144 X 16</td>
<td>On-chip RAM containing data. It can be addressed both directly and indirectly. The instruction DMOVE enables the user to move the contents of a given location in RAM to the next higher location in one machine cycle. This is a very useful function in many applications, such as signal processing.</td>
<td></td>
</tr>
<tr>
<td>Data Memory Page Pointer</td>
<td>DP</td>
<td>1</td>
<td>A single-bit register containing the page address of data RAM. 1 page = 128 words. Note that the second page utilizes only the first 16 words.</td>
<td></td>
</tr>
<tr>
<td>Auxiliary Registers O and 1</td>
<td>AR</td>
<td>2 X 16</td>
<td>The eight least significant bits are used for indirect addressing of data memory. The nine least significant bits can also be configured as bidirectional counters for loop control, with options for autoincrement/decrement.</td>
<td></td>
</tr>
<tr>
<td>Auxiliary Register Pointer</td>
<td>ARP</td>
<td>1</td>
<td>A single-bit register which points to current auxiliary register.</td>
<td></td>
</tr>
<tr>
<td>Shifter</td>
<td>-</td>
<td>-</td>
<td>Two shifters are present. One left-shifts data from 0 to 15 bits on its way to the ALU; the other left-shifts the result of the accumulator either 0, 1, or 4 bits. The shifter is controlled by four bits in the opcode of arithmetic operations, and its output is always a 32-bit word. To handle two's complement arithmetic, shifted data is zero-filled, and the high-order bit is sign-extended. In addition, there are instructions that suppress sign extension.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>SYMBOL</td>
<td>SIZE (bits)</td>
<td>DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>T Register</td>
<td>T</td>
<td>16</td>
<td>Contains the multiplicand in multiply operations.</td>
<td></td>
</tr>
<tr>
<td>Multiplier</td>
<td>-</td>
<td>-</td>
<td>Multiplies two 16-bit numbers. The result is 32 bits. The multiplier is a word from the data RAM or a 13-bit immediate value in the instruction word. The immediate value is loaded right-justified and sign extended.</td>
<td></td>
</tr>
<tr>
<td>P Register</td>
<td>P</td>
<td>32</td>
<td>Contains the 32-bit product of multiply operations.</td>
<td></td>
</tr>
<tr>
<td>Arithmetic Logic Unit</td>
<td>ALU</td>
<td>32</td>
<td>Performs all arithmetic and logical functions except multiply. Logical operations are between the 16 least significant bits of the accumulator and the data memory value.</td>
<td></td>
</tr>
<tr>
<td>Accumulator</td>
<td>ACC</td>
<td>32</td>
<td>Accumulates results of ALU. Holds branch address of program memory during branch operations. Contains an overflow mode (see below).</td>
<td></td>
</tr>
<tr>
<td>Interrupt Flag Register</td>
<td>INTF</td>
<td>1</td>
<td>Used to indicate an interrupt. Automatically cleared upon grant of an interrupt.</td>
<td></td>
</tr>
<tr>
<td>Interrupt Mode Register</td>
<td>INTM</td>
<td>1</td>
<td>Used to mask the Interrupt Flag. Upon grant of an interrupt, this bit is set to one by the DINT instruction. This disables further interrupts. This register is reset by the EINT instruction.</td>
<td></td>
</tr>
<tr>
<td>Overflow Flag</td>
<td>OV</td>
<td>1</td>
<td>A one indicates an overflow in arithmetic operations. The BV (branch on overflow) instruction tests if this flag is clear and clears it. This feature allows the flexibility of overflow examination outside time-critical loops.</td>
<td></td>
</tr>
<tr>
<td>Overflow Mode</td>
<td>OVM</td>
<td>1</td>
<td>Defines whether the TMS32010 operates in the saturated or unsaturated mode during arithmetic operations. In the saturated mode, an overflow/underflow causes the accumulator to be set to its largest/smallest representative value. A logic one enables the overflow mode, and a logic zero disables it.</td>
<td></td>
</tr>
</tbody>
</table>

A.2 ADDRESSING MODES AND INSTRUCTION FORMAT (SEE SECTION 3.3)

A.3 INTERFACE AND CONTROL
Digital Signal Processing. [Appendix A]. page. 153

A.3.1 Program Control

In the microcomputer mode, the TMS32010 can access 2.5K words of program off-chip, in addition to the 1.5K words on-chip. To facilitate this ability, the program counter outputs are buffered to the address pins A11-A0. A strobe output (\( \text{MEM} \)) is generated every machine cycle to enable external memory, except when an IN, OUT, or TBLW instruction is being executed. Data from external memory is transferred to the TMS32010 via the data bus (D15-D0).

The TMS32010 suffers no performance degradation in fetching program words from off-chip memory, as long as the memory access time is approximately 100 ns.

A.3.2 Interrupts

The TMS32010 supports a single-level vectored interrupt with provisions for a full context save. A negative going edge on the \( \text{INT} \) pin generates an interrupt and sets the interrupt flag. When servicing an interrupt, the TMS32010 pushes \( \text{PC} + 1 \) onto the stack and branches to location 2. Locations 0 and 1 are reserved for \( \text{RESET} \).

TMS32010 has an interrupt mode bit which is set by the Disable Interrupt (\( \text{DINT} \)) instruction, and cleared by the Enable Interrupt (\( \text{EINT} \)) instruction. When set, this bit inhibits the TMS32010 from responding to an interrupt. Upon grant of an interrupt by the processor, the \( \text{INT} \) flag is automatically cleared, and the \( \text{INT} \) mode bit is set. This disables servicing future interrupts until \( \text{EINT} \) is executed. This configuration allows the TMS32010 to complete time-critical loops before servicing an interrupt.

This instruction set allows for the storage and recovery of all registers and status bits, except the P register. The TMS32010 also has hardware protection that prevents response to an interrupt between an MPY or MPYK instruction and the next instruction. Thus, the contents of the P register will be accumulated before the interrupt is serviced. In addition, the TMS32010 has hardware that prevents the servicing of an interrupt until the end of multicycle instructions.

A.3.3 Branch Instructions

There are a variety of branch instructions that allow testing for the following conditions:

- Auxiliary register counter portion not zero
- Overflow
- Low-level on the I/O Branch Control pin (\( \text{IO} \))
- Accumulator less than zero, less than or equal to zero, greater than zero, greater than or equal to zero, not zero, equal to zero.

A.3.4 Clock

The TMS32010 can use either its internal oscillator or an external frequency source for a clock. The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN. The frequency of CLKOUT is \( \frac{1}{4} \) the frequency of the input.
A.3.5 I/O

The TMS32010 has an external parallel 12-bit address bus and an external parallel 16-bit data bus. The I/O space is separate from the general memory space and allows direct addressing of up to eight peripherals.

Two instructions, IN and OUT, cause input and output of data to and from the TMS32010 over the data bus. These instructions contain a 3-bit port address which is multiplexed over the three least significant address lines (A2/PA2 – A0/PA0) while the remaining address lines are held at a logic zero.

Inputs or outputs are distinguished by the DEN and WE strobe. Execution of an IN instruction generates a DEN strobe which enters the data on the data bus into the data RAM. Execution of an OUT instruction outputs data from the data RAM onto the data bus and generates a WE strobe.

In addition, two instructions, TBLR (table read) and TBLW (table write), allow a transfer between data and program spaces. TBLR reads a word from program memory and transfers it to the data RAM. TBLW copies a word from the data RAM into external program memory (presumably a RAM). In both instructions, the data memory address is the instruction operand, and the program memory address is contained in the accumulator.

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>OPERATION PERFORMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN PA, A</td>
<td>(D15 through D0) − (A)</td>
</tr>
<tr>
<td></td>
<td>(PA) − (ports A2/PA2 through A0/PA0)</td>
</tr>
<tr>
<td>TBLW A</td>
<td>(PC) + 1 − top of stack</td>
</tr>
<tr>
<td></td>
<td>(ACC) − (PC) − (A11 through A0)</td>
</tr>
<tr>
<td></td>
<td>(A) − (D15 through D0)</td>
</tr>
<tr>
<td></td>
<td>Top of stack − (PC)</td>
</tr>
</tbody>
</table>

NOTE: ( ) = contents of the named location.
2. GENERAL PROGRAMMING INFORMATION

2.1 INTRODUCTION

The TMS32010 Assembly Language is a powerful set of instructions consisting of mnemonic operation codes (called mnemonics) that correspond directly to binary machine instructions. The assembly language program, as coded by the programmer, is called a source program. Before it can be executed by the computer, this source program must be processed by the assembler to obtain a machine language program. This processing of a source program is called assembling. This consists of assembling the binary values (which correspond to the mnemonic operation code) with the binary address information, to form the machine language instruction.

Assembler directives (see Section 5) control the process of making a machine language program from the assembly language program, placing data in the program, and assigning values to symbols to be used in the program.

2.2 DATA AREAS

The data manipulated by the TMS32010 is organized into four areas:

- Register areas: Two 16-bit auxiliary registers, a 1-bit auxiliary register pointer, a 32-bit T register; a 32-bit P register; an accumulator, and a 4 X 12 hardware stack area. In addition, the TMS32010 CPU has access to the 12-bit program counter (PC), the 16-bit status register (ST), and the 1-bit data page pointer (DPI).
- 1536 X 16-bit read-only-memory (ROM) program areas containing the main program and subroutines.
- 144 X 16-bit on-chip RAM data memory areas comprising data tables.
- Eight I/O Ports.

Detailed information and illustrations of these data areas are presented in Appendix A.

2.3 THE TMS32010 INSTRUCTION SET

The TMS32010 instruction set is composed of 60 instructions that provide for the input, output, manipulation, and comparison of data. The instruction set is divided into eight functional categories. They are as follows:

1) ACCUMULATOR INSTRUCTIONS: Provide a variety of ways to add, subtract, load, and store the accumulator.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>ABSOLUTE VALUE OF ACCUMULATOR</td>
</tr>
<tr>
<td>ADD</td>
<td>ADD TO ACCUMULATOR WITH SHIFT</td>
</tr>
<tr>
<td>ADDH</td>
<td>ADD TO HIGH ACCUMULATOR</td>
</tr>
<tr>
<td>ADDS</td>
<td>ADD TO ACCUMULATOR WITH NO SIGN EXTENSION</td>
</tr>
<tr>
<td>LAC</td>
<td>LOAD ACCUMULATOR WITH SHIFT</td>
</tr>
<tr>
<td>LACK</td>
<td>LOAD ACCUMULATOR IMMEDIATE</td>
</tr>
<tr>
<td>SACH</td>
<td>STORE HIGH ACCUMULATOR</td>
</tr>
</tbody>
</table>
SACL: STORE LOW ACCUMULATOR
SUB: SUBTRACT FROM ACCUMULATOR WITH SHIFT
SUBC: CONDITIONAL SUBTRACT (FOR DIVIDE)
SUBH: SUBTRACT FROM HIGH ACCUMULATOR
SUBS: SUBTRACT FROM ACCUMULATOR WITH NO EXTENSION
ZAC: ZERO ACCUMULATOR
ZALH: ZERO ACCUMULATOR AND LOAD HIGH
ZALS: ZERO ACCUMULATOR AND LOAD LOW

2) AUXILIARY REGISTER AND DATA PAGE INSTRUCTION: Load, store, modify, and compare ARs and ARP.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAR</td>
<td>LOAD AUXILIARY REGISTER</td>
</tr>
<tr>
<td>LARK</td>
<td>LOAD AUXILIARY REGISTER IMMEDIATE</td>
</tr>
<tr>
<td>LARP</td>
<td>LOAD AUXILIARY REGISTER POINTER IMMEDIATE</td>
</tr>
<tr>
<td>LDP</td>
<td>LOAD DATA PAGE MEMORY POINTER</td>
</tr>
<tr>
<td>LDPK</td>
<td>LOAD DATA MEMORY PAGE POINTER IMMEDIATE</td>
</tr>
<tr>
<td>MAR</td>
<td>MODIFY AUXILIARY REGISTER AND POINTER</td>
</tr>
<tr>
<td>SAR</td>
<td>STORE AUXILIARY REGISTER</td>
</tr>
</tbody>
</table>

3) T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS: Provide for the preparation for and execution of a multiply.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>APAC</td>
<td>ADD P REGISTER TO ACCUMULATOR</td>
</tr>
<tr>
<td>LT</td>
<td>LOAD T REGISTER</td>
</tr>
<tr>
<td>LTA</td>
<td>LOAD T REGISTER AND ACCUMULATOR PRODUCT</td>
</tr>
<tr>
<td>LTD</td>
<td>LOAD T REGISTER, ACCUMULATOR PRODUCT, AND MOVE DATA IN MEMORY FORWARD ONE LOCATION</td>
</tr>
<tr>
<td>MPY</td>
<td>MULTIPLY T REGISTER BY DATA MEMORY VALUE AND STORE THE PRODUCT IN P REGISTER</td>
</tr>
<tr>
<td>MPYK</td>
<td>MULTIPLY T REGISTER BY IMMEDIATE OPERAND AND STORE VALUE IN P REGISTER</td>
</tr>
<tr>
<td>PAC</td>
<td>LOAD ACCUMULATOR FROM P REGISTER</td>
</tr>
<tr>
<td>SPAC</td>
<td>SUBTRACT P REGISTER FROM ACCUMULATOR</td>
</tr>
</tbody>
</table>

4) BRANCH INSTRUCTIONS: Permit testing of a variety of conditions, along with subroutine calls.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>BRANCH UNCONDITIONALLY</td>
</tr>
<tr>
<td>BANZ</td>
<td>BRANCH ON AUXILIARY REGISTER NOT ZERO</td>
</tr>
<tr>
<td>BGEZ</td>
<td>BRANCH IF ACCUMULATOR &gt; OR = 0</td>
</tr>
<tr>
<td>BGZ</td>
<td>BRANCH IF ACCUMULATOR &gt; 0</td>
</tr>
<tr>
<td>BIOD</td>
<td>BRANCH ON I/O STATUS = 0</td>
</tr>
<tr>
<td>BLEZ</td>
<td>BRANCH IF ACCUMULATOR &lt; OR = 0</td>
</tr>
<tr>
<td>BLZ</td>
<td>BRANCH IF ACCUMULATOR &lt; 0</td>
</tr>
</tbody>
</table>
5) **CONTROL INSTRUCTIONS:** Affect the overflow mode, enable and disable interrupts, and store certain registers which cannot be stored by other instructions.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNZ</td>
<td>Branch if Accumulator NOT = 0</td>
</tr>
<tr>
<td>BY</td>
<td>Branch on Overflow</td>
</tr>
<tr>
<td>BZ</td>
<td>Branch if Accumulator = 0</td>
</tr>
<tr>
<td>CALL</td>
<td>Call Subroutine Indirect via Accumulator</td>
</tr>
<tr>
<td>RET</td>
<td>Return from Subroutine</td>
</tr>
</tbody>
</table>

6) **BOOLEAN OPERATIONS:** Perform logical operations between the accumulator and data memory.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>And with Low Accumulator</td>
</tr>
<tr>
<td>OR</td>
<td>Or with Low Accumulator</td>
</tr>
<tr>
<td>XOR</td>
<td>Exclusive Or with Low Accumulator</td>
</tr>
</tbody>
</table>

7) **I/O AND DATA MEMORY OPERATIONS:** Allow input/output of data to external peripherals, provide for transfer of data within data memory or between program and data memory.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMOV</td>
<td>Shift Contents of Data Memory Address Forward One Location</td>
</tr>
<tr>
<td>IN</td>
<td>Input Data from Port</td>
</tr>
<tr>
<td>OUT</td>
<td>Output Data to Port</td>
</tr>
<tr>
<td>TBLR</td>
<td>Table Read from Program Memory to Data Memory</td>
</tr>
<tr>
<td>TBLW</td>
<td>Table Write from Data Memory to Program Memory</td>
</tr>
</tbody>
</table>
### 2.13 PIN DESCRIPTIONS

Definitions of the TMS32010 pin assignments and descriptions of the function of each pin are presented in Table 2-4. Figure 2-16 illustrates the TMS32010 pin assignments.

#### TABLE 2-4 — TMS32010 PIN DESCRIPTIONS

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>PIN</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>V&lt;sub&gt;cc&lt;/sub&gt;</td>
<td>30</td>
<td>IN</td>
<td>POWER SUPPLIES</td>
</tr>
<tr>
<td>V&lt;sub&gt;ss&lt;/sub&gt;</td>
<td>10</td>
<td>IN</td>
<td>Supply voltage (+ 5 V NOM)</td>
</tr>
<tr>
<td>X&lt;sub&gt;2/CLIN&lt;/sub&gt;</td>
<td>8</td>
<td>IN</td>
<td>Clock output pin for internal oscillator (X2). Also input pin for external oscillator (CLKIN).</td>
</tr>
<tr>
<td>X&lt;sub&gt;1&lt;/sub&gt;</td>
<td>7</td>
<td>OUT</td>
<td>Crystal input pin for internal oscillator</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>6</td>
<td>OUT</td>
<td>Clock output signal. The frequency of CLKOUT is one-fourth of the oscillator input (external oscillator) or crystal frequency (internal oscillator). Duty cycle is 50 percent.</td>
</tr>
<tr>
<td>WE</td>
<td>31</td>
<td>OUT</td>
<td>CONTROL</td>
</tr>
<tr>
<td>DEN</td>
<td>32</td>
<td>OUT</td>
<td>Write Enable. When active (low), WE indicates that valid output data from the TMS32010 is available on the data bus. WE is only active during the first cycle of the OUT instruction and the second cycle of the TBLW instruction (see Section 3.4.3). MEN and DEN will always be inactive (high) when WE is active.</td>
</tr>
<tr>
<td>MEM</td>
<td>33</td>
<td>OUT</td>
<td>Data Enable. When active (low), DEN indicates that the TMS32010 is accepting data from the data bus. DEN is only active during the first cycle of the IN instruction (see Section 3.4.3). MEN and WE will always be inactive (high) when DEN is active.</td>
</tr>
<tr>
<td>MEN</td>
<td>33</td>
<td>OUT</td>
<td>Memory Enable. MEN will be active low on every machine cycle except when WE and DEN are active. MEN is a control signal generated by the TMS32010 to enable instruction fetches from program memory. MEN will be active on instructions fetched from both internal and external memory.</td>
</tr>
</tbody>
</table>
### TABLE 2.4 — TMS32010 PIN DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>PIN</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INT</strong></td>
<td>5</td>
<td>IN</td>
<td>Interrupt. The interrupt signal is generated by applying a negative-going edge to the INT pin. The edge is used to latch the interrupt flag register (INTF) until an interrupt is granted by the device. An active low level will also be sensed. (See Section 2.10.)</td>
</tr>
<tr>
<td><strong>RS</strong></td>
<td>4</td>
<td>IN</td>
<td>Interrupts. When an active low is placed on the RS pin for a minimum of five clock cycles, DEN, WE, and MEN are forced high, and the data bus (D15 through D0) is tri-state. The program counter (PC) and the address bus (A11 through AD0) are then synchronously cleared after the next complete clock cycle from the falling edge of RS. RS also disables the interrupt, clears the interrupt flag register, and leaves the overflow mode register unchanged. The TMS32010 can be held in the reset state indefinitely.</td>
</tr>
<tr>
<td><strong>BIO</strong></td>
<td>9</td>
<td>IN</td>
<td>I/O Branch Control. If BIO is active (low) upon execution of the BIOZ instruction, the device will branch to the address specified by the instruction (see Section 2.9).</td>
</tr>
<tr>
<td><strong>MC/MP</strong></td>
<td>3</td>
<td>IN</td>
<td>Microcomputer/Microprocessor Mode. A high on the MC/MP pin enables the microcomputer mode. In this mode, the user has available 1524 words of on-chip program memory. (Program memory locations 1524 through 1535 are reserved.) The microcomputer mode also allows an additional 2560 words of program memory to reside off-chip. A low on the MC/MP pin enables the microprocessor mode. In this mode, the entire memory space is external, i.e., addresses 0 through 4095. (See Section 2.3.1.)</td>
</tr>
</tbody>
</table>

#### BIDIRECTIONAL DATA BUS

- D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when WE is active (low).
Table 2-4 — TMS32010 Pin Descriptions (Concluded)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>PIN</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A11</td>
<td>27</td>
<td>OUT</td>
<td>PROGRAM MEMORY ADDRESS BUS AND PORT ADDRESS BUS</td>
</tr>
<tr>
<td>A10</td>
<td>28</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>29</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>34</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>35</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>36</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>37</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>38</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>39</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A2/PA2</td>
<td>40</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A1/PA1</td>
<td>1</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>A0/PA0</td>
<td>2</td>
<td>OUT</td>
<td></td>
</tr>
</tbody>
</table>

Program memory A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance. During execution of the IN and OUT instructions, pins A2 through A0 carry the port addresses PA2 through PA0.

Figure 2-16 — TMS32010 Pin Assignments
### 3.4.2 Instruction Set Summary

The instruction set summary in the following table consists primarily of single-cycle single-word instructions. Only infrequently used branch and I/O instructions are multicycle.

**Table 3.2 — Instruction Set Summary**

<table>
<thead>
<tr>
<th>Accumulator Instructions</th>
<th>No. Cycles</th>
<th>No. Words</th>
<th>Opcode</th>
<th>Instruction Register</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mnemonic</strong></td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ABS</strong></td>
<td>1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ADD</strong></td>
<td>1 1 0 0 0 0 ← S → I ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ADDH</strong></td>
<td>1 1 0 1 1 0 0 0 0 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ADDS</strong></td>
<td>1 1 0 1 1 0 0 0 1 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>1 1 0 1 1 1 0 0 0 1 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LAC</strong></td>
<td>1 1 0 0 1 0 ← S → I ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LACK</strong></td>
<td>1 1 0 1 1 1 1 1 1 1 1 0 ← K →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td>1 1 0 1 1 1 1 0 1 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SACH</strong></td>
<td>1 1 0 1 0 1 1 ← X → I ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SACL</strong></td>
<td>1 1 0 1 0 1 0 0 0 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUB</strong></td>
<td>1 1 0 0 0 1 ← S → I ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUBC</strong></td>
<td>1 1 0 1 1 1 0 0 1 0 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUBH</strong></td>
<td>1 1 0 1 1 0 0 0 1 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUBS</strong></td>
<td>1 1 0 1 1 0 0 0 1 1 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>XOR</strong></td>
<td>1 1 0 1 1 1 1 0 0 0 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZAC</strong></td>
<td>1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZALH</strong></td>
<td>1 1 0 1 1 1 1 6 0 0 1 0 1 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZALS</strong></td>
<td>1 1 0 1 1 0 0 1 0 0 1 0 1 ← D →</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-2 - Instruction Set Summary (Continued)

#### Auxiliary Register and Data Page Pointer Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>No. Cycles</th>
<th>No. Words</th>
<th>OPCODE Instruction Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAR</td>
<td>Load auxiliary register</td>
<td>1</td>
<td>1</td>
<td>0 0 1 1 1 0 0 R I ← D</td>
</tr>
<tr>
<td>LARK</td>
<td>Load auxiliary register</td>
<td>1</td>
<td>1</td>
<td>0 1 1 1 0 0 R ← K</td>
</tr>
<tr>
<td>LARP</td>
<td>Load auxiliary register</td>
<td>1</td>
<td>1</td>
<td>0 1 1 0 1 0 0 0 0 0 0 0 0 K</td>
</tr>
<tr>
<td>LDP</td>
<td>Load data memory page pointer immediate</td>
<td>1</td>
<td>1</td>
<td>0 1 1 0 1 1 1 1 1 ← D</td>
</tr>
<tr>
<td>LDPK</td>
<td>Load data memory page pointer immediate</td>
<td>1</td>
<td>1</td>
<td>0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 K</td>
</tr>
<tr>
<td>MAR</td>
<td>Modify auxiliary register and pointer</td>
<td>1</td>
<td>1</td>
<td>0 1 1 0 1 0 0 0 0 ← D</td>
</tr>
<tr>
<td>SAR</td>
<td>Store auxiliary register</td>
<td>1</td>
<td>1</td>
<td>0 0 1 1 0 0 0 R I ← D</td>
</tr>
</tbody>
</table>

#### Branch Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>No. Cycles</th>
<th>No. Words</th>
<th>OPCODE Instruction Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Branch unconditionally</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BANZ</td>
<td>Branch on auxiliary register not zero</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BGEZ</td>
<td>Branch if accumulator &gt; 0</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BGZ</td>
<td>Branch if accumulator &gt; 0</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BIOZ</td>
<td>Branch on $0 \geq 0$</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BLEZ</td>
<td>Branch if accumulator &lt; 0</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BLZ</td>
<td>Branch if accumulator &lt; 0</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BNZ</td>
<td>Branch if accumulator &lt; 0</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BV</td>
<td>Branch on overflow</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>BZ</td>
<td>Branch if accumulator &lt; 0</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>CALA</td>
<td>Call subroutine from</td>
<td>2</td>
<td>1</td>
<td>0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>CALL</td>
<td>Call subroutine immediately</td>
<td>2</td>
<td>2</td>
<td>1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS</td>
</tr>
<tr>
<td>RET</td>
<td>Return from subroutine</td>
<td>2</td>
<td>1</td>
<td>0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1 ← BRANCH ADDRESS</td>
</tr>
</tbody>
</table>
### Table 3-2 - Instruction Set Summary (Concluded)

#### T Register, P Register, and Multiply Instructions

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
<th>CYCLES</th>
<th>WORDS</th>
<th>OPCODE</th>
<th>INSTRUCTION REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>APAC</td>
<td>Add P register to accumulator</td>
<td>1</td>
<td>1</td>
<td>0111111100111111</td>
<td></td>
</tr>
<tr>
<td>LT</td>
<td>Load T register</td>
<td>1</td>
<td>1</td>
<td>011101101010</td>
<td></td>
</tr>
<tr>
<td>LTA</td>
<td>LTA combines LT and APAC into one instruction</td>
<td>1</td>
<td>1</td>
<td>0111011001</td>
<td></td>
</tr>
<tr>
<td>LTD</td>
<td>LTD combines LT, APAC, and DMOV into one instruction</td>
<td>1</td>
<td>1</td>
<td>110111110101</td>
<td></td>
</tr>
<tr>
<td>MPY</td>
<td>Multiply with T register, store product in P register</td>
<td>1</td>
<td>1</td>
<td>011101101101</td>
<td></td>
</tr>
<tr>
<td>MPYK</td>
<td>Multiply T register with immediate operand, store product in P register</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>PAC</td>
<td>Load accumulator from P register</td>
<td>1</td>
<td>1</td>
<td>1111111111000111</td>
<td></td>
</tr>
<tr>
<td>SPAC</td>
<td>Subtract P register from accumulator</td>
<td>1</td>
<td>1</td>
<td>0111111111101000</td>
<td></td>
</tr>
</tbody>
</table>

#### Control Instructions

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
<th>CYCLES</th>
<th>WORDS</th>
<th>OPCODE</th>
<th>INSTRUCTION REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DINT</td>
<td>Disable interrupt</td>
<td>1</td>
<td>1</td>
<td>01111111110000001</td>
<td></td>
</tr>
<tr>
<td>EINT</td>
<td>Enable interrupt</td>
<td>1</td>
<td>1</td>
<td>01111111111100000001</td>
<td></td>
</tr>
<tr>
<td>LST</td>
<td>Load status register</td>
<td>1</td>
<td>1</td>
<td>01111111111000000001</td>
<td></td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>1</td>
<td>1</td>
<td>0111111111100000000000</td>
<td></td>
</tr>
</tbody>
</table>
| POP      | Pop stack to accumulator | 2   | 1     | 0111111111010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010
APPENDIX B

PROGRAM LISTINGS.

Set.1 Program listings.

These are shown below starting with the C3M Board Data Capture & Transfer Program - "TRANBBC" followed by the BBC Basic display Program "C3MGRA".

The TMS32010 Program TRANBBC for the C3M board.

IDT 'TRANBBC'
* A TMS32010 ASSEMBLY LANGUAGE PROGRAM
* ROUTINE TO READ IN N DATA VALUES AND SEND TO BEEB
  AORG 200
  :DATA MEMORY ALLOCATION
  ONE    EQU 0
  POINT  EQU 1
  N      EQU 2 : NO OF SAMPLES 128
  BEEB   EQU 3 ; START CODE 0A0A
  HIGH   EQU 4
  LOW    EQU 5
  DELAY  EQU 6 ; TIME DELAY 100 HEX
  ADDRESS EQU 7
  RATE   EQU 8 ; SET TO 4 kHz
* END OF CONSTANT TABLE
B START     ; JUMP AROUND DATA
* DATA INITIALISATION
  DATA 1
  DATA >10 ; INPUT DATA MEMORY ADDRESS
  DATA 128 ; N = 128
  DATA >0A ; TEST CODE
  DATA 00
  DATA 00
  DATA >100 ; DELAY VALUE
  DATA 00
* TABLE DATA 06  ; SAMPLE RATE CODE
* END OF DATA VALUES
* MAIN PROGRAM START
START  CALL DATA_LOAD
       CALL ADRate
Snt    CALL GET_DATA
       CALL TELL_BEEB
       CALL SEND_DATA
       B   Snt
* END OF MAIN PROGRAM
* SUBROUTINE TO SET UP CONSTANTS IN DATA MEM
DATA_LOAD LDPK 0  ; SET DATA PAGE POINTER TO 0
       LARP 0  ; SET AUX REG POINTER TO 0
       LARK 0.8 ; SET AUX REG 0 = 8
       LACK 1 ; SET ACC = 1
       SAACL ONE  ; STORE IN LOC ONE
       LT ONE  ; GET ONE INTO T REG
       MPYK Table ; MULT T BY LOC OF Table
       PAC  ; TRANSFER TO ACC
* ACCUMULATOR NOW HOLDS ADDRESS OF TOP OF DATA
* IN PROG MEMORY AND AUX REG HOLDS LOC OF DATA MEMORY
Next TBLR *  ; TRANSFER PROG MEMORY TO DATA MEMORY
       SUB ONE
       BANZ Next  ; UNTIL DONE
       RET * END OF DATA TRANSFER ROUTINE
* SUBROUTINE TO SET SAMPLE RATE
ADRate  OUT RATE.2
       RET
* END OF ROUTINE
* SUBROUTINE TO GET DATA BYTES
GET_DATA LAR 0.POINT  ; POINT TO DATA LOC
       LAC N
Wait BIOZ Wait  ; WAIT FOR EOC IN A/D CONVERTER
       IN **+,1 ; GET 16 BIT VALUE & INC AUX REG
* VALUE INPUT FROM PORT 1 AND STORED IN DATA MEM POINTED
* TO BY AUX REG.
       SUB ONE  ; DECREMENT N
       BNZ Wait  ; UNTIL DONE
       RET
* END OF INPUT ROUTINE
* ROUTINE TO SEND START CODES 0A, 0A TO BBC
TEll_BEEB LAC DELAY.5
       OUT BEEB,BBC
Slow1  SUB ONE
       BNZ Slow1
       LAC DELAY.5
       OUT BEEB,BBC
Slow2  SUB ONE
       BNZ Slow2
       RET
* END OF ROUTINE
* ROUTINE TO SEND DATA TO BBC
SEND_DATA LARP 0
       LAR 0.POINT  ; POINT TO DATA MEM LOC
       LAC N
ArounD  SAACL ADDRESS  ; TEMP STORE FOR COUNTER
       ZAC  ; ZERO ACC
Digital Signal Processing. [Appendix B]. page 168

LAC *, 8 ; SHIFT DATA LEFT 8 BITS
SACH HIGH ; STORE HIGH BYTE
LAC * ; GET DATA AGAIN
SACL LOW ; STORE FOR LOW BYTE

*SEND DATA TO BBC
OUT LOW, BBC
LAC DELAY, 5

Loop1 SUB ONE
BNZ Loop1
OUT HIGH, BBC
LAC DELAY, 5

Loop2 SUB ONE
BNZ Loop2

*DECREMENT COUNTER
LAC ADDRESS ; GET COUNTER VALUE
SUB ONE
MAR *+ ; MODIFY AUX REG BY +1
BNZ AROUND ; UNTIL DONE
RET

* END OF DATA TRANSFER ROUTINE
* END OF PROGRAM ... AJS .. 1989

The BBC Basic capture & display program C3MGRA.

10 REM *** Data Transfer Test Program ****
20 REM *** C3M board to BBC **********
30 REM *** Displays output waveform from C3M **
40 REM *** on the screen of the BBC ********
50 REM *** BBC Prog name 0.C3MGRA *********
60 MODE 1: PORTE=&FE60 : ACMP=&FE6B : DDBR=&FE62
70 PCR=&FE6C : PR= &FE6D : IER=&FE6E
80 MEMLOC=&2500 : COUNT=202
85 REM **** PROG START ***
90 PROCaxis
100 PRINT TAB(4,4);" MEMORY TRANSFER C3M TO BBC "
105 PROCassemble
110 CALL Start
120 PROCdraw
130 GOTO 110
140 END
150 DEFPREOCaxis
160 VDU24, 300, 140; 1120; 700;
170 GCOL 0.1: GCOL 0.131 : CLG
180 STEP=(1120-140)/(COUNT/2) : HIGHT=200
190 ENDPROC
200 DEFPREOCDRAW
210 CLG
220 FOR DATA=2 TO COUNT STEP 2
230 YH=? (MEMLOC+DATA+1) ; YL=? (MEMLOC+DATA)
IF YH<128 THEN 270
250 YL=(YL-1) EOR &FF: YH= YH EOR &FF
260 Y=-(YH*256+YL): GOTO 280
270 Y=YH*256+YL
280 Y2=(Y/2048)*Hght
290 MOVE ((DATA/2)*STEP+140),350
300 DRAW ((DATA/2)*STEP+140),350+Y2
310 NEXT
320 ENDP
330 DEFPROCA
340 FOR PASS%:0 TO 2 STEP 2
350 P%=&2E00
360 I OPT PASS%
370 .START LDA&:00 /SET UP PORT B
380 STA DDrb
390 LDA&:02 /SET UP AUX CON REG
400 STA Acr /TO LATCH SIGNAL
410 /WAIT FOR PULSE ON DOUT LINE OF C3M
420 .AROUND LDX&0
430 .WAIT LDA&16
440 BIT Ifr
450 BEQ wait
460 /GET START CODE OA,OA
470 LDA Portb
480 CMP &:&01
490 BNE wait
500 STA Memloc,X
510 INX
520 CPX&2
530 BNE wait
540 /STORE RESULTS OF C3M BOARD
550 .NEXT LDA&16
560 BIT Ifr
570 BEQ next
580 LDA Portb
590 STA Memloc,X
600 INX
610 CPX&Count
620 BNE next
630 /TEST TO SEE IF 1ST 2 BYTES ARE OA,OA
640 .TEST LDX&1
650 LDA Memloc
660 CMP Memloc,X
670 BNE around
680 RTS
690 NEXT PASS%
700 ENDP
Digital Signal Processing. [Appendix B]. page 170

Set 2. Program listings.

These are shown below starting with the TMS32010 assembly language program C3MTIME.SRC, followed by the IBM PC assembly language program IBMTIME.ASM and finally by the GW Basic program PLOTTME.BAS.

The TMS32010 Program C3MTIME.SRC

```
IDT C3MTIME
: Note 1 = 07FF 0 = F800 This suits C3M board MAX and
: MIN values which are in Q15 format in prog.
: A simple data test transfer program
: Uses interrupt routine for data transfer
: to C3M board.

AORG > 200

: Data memory allocation
:
N EQU 128 ; data count
IFLAG EQU 20 ; interrupt flag byte
IBM_VALUE EQU 19 ; data location
PROGMEM EQU 18 ; program memory address
COMP EQU 17 ; code to send to host > 0A0A
STORE EQU 16 ; temp store
HIGH EQU 15
LOW EQU 14
DELAY EQU 13 ; delay for sending data to host > 100
ADDRESS EQU 12
RATE EQU 11 ; sampling rate store
XI EQU 10 ; temp data store
YI EQU 9
I EQU 8 ; memory offset
HOLDN EQU 6 ; contains value N
ONE EQU 5 ; contains value 1
ZERO EQU 4 ; contains 0
PA0 EQU 0 ; Port A0
PA1 EQU 1 ; Port A1
```
Digital Signal Processing. [Appendix B]. page 171

: BEGIN prog memory section
:

START           CALL   ADRATE
SENT            CALL   SETDATA
             CALL   INT_SETUP
             CALL   GETDATA
             CALL   SENDATA

;END OF MAIN prog LOOP
;*******************************************************************************
; START OF SET sample rate
ADRATE          LACK  >E    ;20 KHz sample rate
SAACL           RATE    ;temp DATA store
OUT              RATE,2
RET

;END OF SAMPLE RATE
;*******************************************************************************
;START OF SET DATA routine

SE TEDATA     LDPK 0
            LACK 1
            SAACL ONE
            LACK 0
            SAACL ZERO
            LT ONE
            MPYK N
            PAC
            SAACL HOLDN
            LACK >OA
            SAACL COMP
            LT ONE
            MPYK >FO
            PAC
            SAACL DELAY
            LT ONE
            MPYK TOP
            LACK >10
            APAC  ;ADD P TO ACC
            SAACL PROGMEM
            RET

;END OF SET DATA
;*******************************************************************************
;START OF GET DATA routine

GETDATA         LAC   PROGMEM
LARP 1
LAR 1,HOLDN ; DATA COUNT
MAR *- ;Aux reg = N-1

WAIT            BIOZ  WAIT
                IN   STORE,1 ;GET BYTE
                TBLW STORE ;Store
                ADD ONE ;INCREMENT STORE
                BANZ  WAIT ;Dec aux reg and branch until aux reg
IS ZERO
                RET

;END OF GET DATA routine
;*******************************************************************************
;Send start code routine

BYTE_OUT        LAC    STORE
OUT    STORE,IBM
LAC    DELAY
SLOW1
    SUB    ONE
    BNZ    SLOW1
    LAC    IFLAG
    SUB    ONE
    BNZ    BYTE_OUT
    LAC    IBM_VALUE
    SUB    STORE
    BNZ    BYTE_OUT
LAC ZERO
SACL IFLAG
RET

:*****************************************
: Start of data sending routine
SENDATA
    LARP    0
    LAC    PROGMEM  ; Point to data in program
    SACL    ADDRESS
    LAR    0, HOLDN
    MAR    -*  ; N-1
LAC ZERO
SACL IBM_VALUE
LAC ZERO
SACL I   ; Offset for address
SACL IFLAG

: Start of main data send routine
LAC COMP
SACL STORE
CALL BYTE_OUT
NEXT
    LAC    ADDRESS
ADD I    ; Add offset
TBLR XI   ; Real part
ADD ONE
TBLR YI    ; Imag part
LAC XI,8  ; High byte
SACH STORE
LACK 255
AND STORE  ; Mask off
SACL STORE
CALL BYTE_OUT

LACK 255  ; Low byte
AND XI    ; And mask off
SACL STORE
CALL BYTE_OUT

: LAC YI,8    ; High byte
SACH STORE
LACK 255    ; Mask off
AND STORE
SACL STORE
CALL BYTE_OUT
LACK 255    ; Low byte
AND YI      ; And mask off
SACL STORE
CALL BYTE_OUT
LAC I
ADD ONE,1

; SA CL I ; I = I + 2
B ANZ NEXT ; UNTIL DONE
RET

; END OF DATA SENDING
;******************************************************************************
INT_SETUP LDPK 0
DINT
LT ONE
MPYK INT_ROUTINE
PAC
SA CL ADDRESS
LT ONE
MPYK >FFD
PAC
TBLW ADDRESS ; STORE ADDRESS OF INTERRUPT
; ROUTINE AT INT VECTOR FFD
EINT
RET

;******************************************************************************
INT_ROUTINE PUSH
IN IBM_VALUE,IBM
LACK 255 ; MASK
AND IBM_VALUE
SA CL IBM_VALUE
LAC ONE
SA CL IFLAG
EINT
POP
RET

; END OF INTERRUPT HANDLER ROUTINE
;******************************************************************************
TOP END

END OF PROGRAM AJS SGIHE MECH ENG 1990

An inspection of the "GETDATA" subroutine in all the TMS32010 programs used shows that this is the fastest capture rate that can be used, no program lines can be removed.
The 8086/80286 Assembly program IBMTIME.COM.

As stated earlier this program performs two functions, one to run and store the restart address and secondly as the data transfer program which runs when "CALL"ed by the Basic program from the restart address.
TITLE IBMTIME.COM

: Machine code routine to read in data from the C3M board into PC memory.
: Called from Basic program using CALL offset
: OFFSET = (101)*256 + (100)
: SEG  = (103)*256 + (102)

; Modified to use interrupt check routine
; uses data length 128

;**********************************************************************************************
TRANSFER SEGMENT PUBLIC
ASSUME CS:TRANSFER, DS:TRANSFER
ORG 100H

START: JMP Place_vector

ATest DW 0A0AH ; START CODE
Counter DW 00  ; LOOP COUNTER
Store DW 0000 ; TEMP STORE
Value DW 256 DUP(?) ; DATA SPACE
NValue DB 128  ; N VALUE
Byte DB 00  ; TEMP VALUE STORE

DELAY PROC NEAR
    PUSH AX
    PUSH DX
    MOV AX,03H
    MOV DX,0A0H
    DEC DX
    JNZ Lop1
    DEC AX
    JNZ Lop2
    POP DX
    POP AX
    RET

DELAY ENDP

IN_PROC PROC NEAR
    MOV DX,371H ; PORT FLAG ADDRESS
    Not_set:
        IN AL,DX ; GET FLAG BYTE
        TEST AL,128 ; IS BIT 7 SET?
        JZ Not_set
        DEC DX ; SET TO DATA PORT ADDRESS
        IN AL,DX ; GET BYTE
        MOV [BX1].AL  ; AND STORE
    RET

IN_PROC ENDP

OUT_PROC PROC NEAR
    XOR AL,AL ; ZERO AL
    MOV DX,372H ; INTERFACE BOARD PORT ADDRESS
    OUT DX,AL
    MOV AL,Bite ; GET BYTE
    MOV DX,370H ; INTERFACE BOARD OUTPUT PORT
    OUT DX,AL ; SEND BYTE OUT

OUT_PROC ENDP
MOV AL, 5 ; Codes 5 and 4
MOV DX, 372H
OUT DX, AL
DEC AL ; REG FOR PC INTERFACE BOARD
OUT DX, AL
XOR AL, AL
OUT DX, AL
RET

Out_proc ENDP
;
Mainroutine PROC FAR
PUSH DS
PUSH SS
PUSH DX
PUSH CX
PUSH AX
PUSH BX
MOV AX, CS
MOV DS, AX
MOV BX, OFFSET Value
XOR AX, AX
MOV AL, NVALUE
MOV CL, 01
SHL AX, CL ; MULT BY 2 (2*N)
MOV COUNTER, AX
MOV CL, 08
;
Starter:
CALL Delay
CALL In_proc ; GET MSB
CMP AX, Atest ; IS IT 0A0A
JZ Start_up
SHL AX, CL
NOP ; CALL Delay
CALL In_proc
CMP AX, Atest
JZ Start_up
SHL AX, CL
JMP Starter
;
Start_up:
CALL Delay
MOV Byte, AL
CALL Out_proc
MOV CX, COUNTER
;
Data_in:
MOV COUNTER, CX
CALL Delay
CALL In_proc
MOV AL, [BX]
MOV Byte, AL
CALL Out_proc
INC BX
CALL In_proc
MOV AL, [BX]
MOV Byte, AL
CALL Out_proc
INC BX
Digital Signal Processing. [Appendix B]. page 177

```
MOV CX,COUNTER
LOOP Data_in
POP BX
POP AX
POP CX
POP DX
    :Restore registers
POP SS
POP DS
RET

:MainRoutine ENDP

Place_Vector: MOV AX,0000H ;Set data seg to zero
    MOV DS,AX
    MOV BX,100H ;Set offset to 100H
    MOV AX,OFFSET Main_Routine
    MOV [BX].AX ;Store start address
    MOV AX,CS
    MOV [BX+2].AX ;Store segment
    MOV AX,OFFSET Value
    MOV [BX+4].AX ;Store no of
    DATA_BYTES
    PUSH CS
    POP DS ;Get CS into DS
    MOV DX,OFFSET Finish
    INT 27H
    INT 20H ;Return to DOS

Finish: Transfer ENDS
END Start
```

GWBasic display program PLOTIME.BAS

This Basic display program is shown below and is similar to that of the FFT display programs shown later in Sets 3, 4 and 5. It requires the C3M and IBM programs (C3MTIME & IBMTIME) to be run first.

```
10 REM ** C3M Time Based screen display PLOTIME.BAS ****
20 REM ** 1990 CARDIFF INSTITUTE ***
30 REM ** A J SHEPHERD Mech/Prod Section
40 REM **********************************************
50 REM load & Run C3MTIME.SRC in C3M first
60 REM Then load & Run IBMTIME.COM on IBM PC
```
70 REM ********************************************
80 REM ** start of information outputting section ***
90 REM key off
100 DIM X(200),Y(200),XL(200),XH(200),TX(200)
110 DIM Y1(200),Y2(200)
120 SCREEN 9 : CLS : COLOR 7
130 KEY OFF
140 NV=127 :REM N value less 1
150 GOSUB 1090 : REM set up information display
160 GOSUB 400 :REM set up graph base
170 GOSUB 290 :REM set up call routine
180 GOSUB 780 : REM get fft values
190 GOSUB 980 : REM set up graph heights for the FFT
200 GOSUB 1230 :REM redraw graphs
210 GOSUB 620 :REM plot results
220 LOCATE 15,10 :BEEP
230 LOCATE 23,4:PRINT".Press.S.to STOP..."
240 GOSUB 720 :REM stop routine
250 GOTO 180
260 LOCATE 1,1
270 END
280 REM ********************************************
290 REM routine to set up parameters
300 REM to call assy lang prog C3M2MEM.COM
310 DEF SEG=0 :REM set for DS seg = 0
320 CS=PEEK(&H102)+PEEK(&H103)*256
330 OFFSET=PEEK(&H100)+PEEK(&H101)*256
340 NDATALOC=PEEK(&H104)+PEEK(&H105)*256
350 DEF SEG
360 C3M=OFFSET
370 DEF SEG=CS :REM set CS seg to value found
380 RETURN
390 REM ********************************************
400 REM sets up graph on screen
410 COLOR 14 :CLS
420 LINE (34,29)-(540,230),4,B
430 LINE(35,30)-(539,229),1,BF
460 FOR N=0 TO 190 STEP 19
470 LINE (30,225-N)-(535,225-N),0
480 NEXT
490 X=4:A=0
500 FOR N=0 TO 57 STEP 7.7
510 LOCATE 18,N+X
520 PRINT A:A=A+1
530 NEXT
540 X=0
550 FOR N=0 TO 16 STEP 2.7
560 LOCATE 17-N,1
570 PRINT USING"£.£":(-.5+X*.1):X=X+2
580 NEXT
590 LOCATE 8,70:PRINT"Time plot"
595 LOCATE 12,70:PRINT"Sample rate"
597 LOCATE 13,70:PRINT" 20 KHz "
600 RETURN
610 REM ********************************************
620 REM results plotting routine
630 COLOR 14
650 ASTEP=510/(NV+1)
660 FOR N=0 TO NV
670 PSET(35+ASTEP*N,130)
680 Y=Y1(N)*100
690 DRAW "U=Y;"
700 NEXT
710 RETURN
715 REM ***********
720 REM time delay for stop key
730 FOR N%=1 TO 100
740 A$=INKEY$
750 IF A$="S" OR A$="s" THEN 270
760 NEXT N%
780 RETURN
790 REM ** get values from c3m --
800 CALL C3M
810 XMAX=0:YMAX=0
820 DATALOC=NDATALOC
830 FOR N=0 TO NV
840 X(N)=OLDX : Y(N)=OLDY
850 XH(N)=PEEK(DATALOC):XL(N)=PEEK(DATALOC+1)
860 TX(N)=256*XH(N)+XL(N)
870 IF XH(N)<=&H7F THEN 910
880 XH(N)=XH(N) XOR &HFF: XL(N)=XL(N)XOR &HFF
890 X(N)=-(XL(N)+XH(N)*256)
900 GOTO 920
910 X(N)=XL(N)+XH(N)*256
920 DATALOC=DATALOC+2
930 IF X(N)-->XMAX THEN XMAX=X(N)
960 NEXT N
970 RETURN
975 REM *********************************************************
980 REM calc plot heights
1000 FOR N=0 TO NV
1010 X(N)=X(N)/1024
1020 Y1(N)=X(N)
1030 IF Y1(N)>1 THEN Y1(N)=1
1040 IF Y1(N)<-1 THEN Y1(N)=-1
1050 NEXT N
1080 RETURN
1090 REM Set up information page
1100 LINE (25,10)-(600,250),2,BF
1110 LINE (35,20)-(857,230),7,BF
1120 LINE (50,30)-(550,210),0,BF
1130 COLOR 14
1140 LOCATE 9,15:PRINT"... TEST PROGRAM ..."
1150 LOCATE 11,15:PRINT"This program tests the C3M set up"
1160 LOCATE 13,15:PRINT"By plotting the results as a time based plot"
1170 LOCATE 15,15:PRINT"Press any key continue"
1180 A$=INKEY$
1190 IF A$="" THEN 1180
1200 BEEP:BEEP
1210 RETURN
1230 REM *********************************
1240 REM predraw graph bases
Set 3  Program listings.

The TMS32010 Length 8 DIF FFT program.

Scaling of the intermediate and final FFT values has been achieved by dividing by two at each stage, giving a total divide of 8 — i.e. N.

```
1250 COLOR 14
1260 REM ***** magnitude *****
1270 LINE(35,30)-(539,229),1,BF
1280 FOR N=0 TO 190 STEP 19
1290 LINE (30,225-N)-(535,225-N),0
1300 NEXT
1310 RETURN
1320 END
```

```
:IDT     FFT8TST
 :2nd Test proG using N=8 and reads data input
 :1,1,1,1,0,0,0,0 from memory.
 :Uses interrupt routine to check data
 :1=07FF -1=F800 This suits C3M board max and
 :min values which are in Q15 format.
 :-ve numbers are in 2's complement form.
 :Cooley Tukey radix 2 length 8 DIF FFT
 :TMS32010 Assembly Lang. For the C3M unit
 :Single FFT butterfly complex data
 :Uses lookup table for twiddle factors
 :Scaling done in FFT /2 at each stage (=/8)
 :Real and Imaginary data in consecutive locs

AORG >200

N is size of transform N = 2**M
N   EQU  8
M    EQU  3

:Data memory allocation
IFLAG  EQU 34 ;Interrupt occurred flag
C3VALUE EQU 33 ;Data value
IBM_VALUE EQU 32 ;Data location
PROGMEM EQU 30 ;Program memory address
COMP   EQU 31 ;Code to send to host >0A0A
STORE  EQU 2 ;Temp store
```
HIGH EQU 3
LOW EQU 4
DELAY EQU 5 ; Delay for sending data to host > 100
ADDRESS EQU 6
RATE EQU 7 ; Sampling rate no 6
XI EQU 8 ; Array value X(I)
YI EQU 9 ; Array value Y(I)
XL EQU 10 ; Array value X(L)
YL EQU 11 ; Array value Y(L)
XT EQU 12 ; Temp real part
YT EQU 13 ; Temp imag part
I EQU 14 ; 1st index
L EQU 15 ; 2nd index
COS EQU 16 ; Twiddle factor real
SIN EQU 17 ; Twiddle factor imag
IA EQU 18 ; Index to twiddle factors
IE EQU 19 ; Increment to IA
HOLDN EQU 20 ; Contains value N
QUARTN EQU 21 ; Contains value N/4
N1 EQU 22 ; Increment to I
N2 EQU 23 ; Separation of I and N
J EQU 24 ; Loop counter
ONE EQU 25 ; Contains value 1
ZERO EQU 26 ; Contains 0
PA0 EQU 0 ; Port A0
PA1 EQU 1 ; Port A1
TTEST EQU 27 ; Test value 54 (36 hex)
TABLE EQU 29 ; Location of coeff table

: Begin proc memory section

START CALL AD RATE
SENT CALL SETDATA
CALL INT_SETUP
CALL GETDATA
CALL FFT
CALL BITREVERSE
CALL SENDATA

; End of main prog loop

: Start of set sample rate
AD RATE LACK 6
SACL RATE
OUT RATE, 2
RET

: End of sample rate

: Start of set data routine
SETDATA LDPK 0
LACK 1
SACL ONE
SACL IE
LACK 0
SACL ZERO
SACL XI
SACL XL
SACL YI
SACL YL
LT ONE
MPYK SINE
PAC
SACL TABLE
MPYK N
PAC
SACL HOLDN
SACL N2
LAC HOLDN, 14
SACH QUARTN
LACK > 0A
SACL COMP
LT ONE
MPYK > 100
PAC
SACL DELAY
LT ONE
MPYK TOP
LACK > 10
APAC ; ADD P TO ACC
SACL PROGMEM
LT ONE
MPYK DVALUE
PAC
SACL C3VALUE
RET

: END OF SET DATA
: *********************************
: START OF GET DATA ROUTINE
GETDATA LAC ZERO
SACL I
LARP 1
LAR 1, HOLDN : DATA COUNT
MAR *-
WAIT
ADD I
TBLR STORE ; GET BYTE
LAC PROGMEM
ADD I
TBLW STORE ; STORE IN PROGMEM
LAC I
ADD ONE
SACL I
LAC C3VALUE
ADD I
TBLR STORE
LAC PROGMEM
ADD I
TBLW STORE
LAC I
ADD ONE
SACL I
BANZ WAIT ; DEC AUX REG AND BRANCH UNTIL
: AUX REG IS ZERO
RET
; END OF GET DATA ROUTINE
;*************************************************************************************************
; SEND START CODE ROUTINE
BYTE_OUT LAC STORE
OUT STORE,IBM
LAC DELAY
SLOW1 SUB ONE
BNZ SLOW1
LAC IFLAG
SUB ONE
BNZ BYTE_OUT
LAC IBM_VALUE
SUB STORE
BNZ BYTE_OUT
LAC ZERO
SACL IFLAG
RET
; END OF BYTEOUT ROUTINE
;*************************************************************************************************
; START OF DATA SENDING ROUTINE
SENCDATA LARP 0
LAC PROGMEM :POINT TO DATA IN PROGMEM
SACL ADDRESS
LAR 0,HOLDN
MAR *- ;N-1
LAC ZERO
SACL IBM_VALUE
LAC ZERO
SACL I ;OFFSET FOR ADDRESS
SACL IFLAG
; START OF MAIN DATA SEND ROUTINE
LAC COMP
SACL STORE
CALL BYTE_OUT
NEXT LAC ADDRESS
ADD I ;Add offset
TBLR XI ;Real part
ADD ONE
TBLR YI ;Imag part
LAC XI.8 ;HIGH BYTE
SACH STORE
LACK 255
AND STORE ;MASK OFF
SACL STORE
CALL BYTE_OUT
LACK 255 ;LOW BYTE
AND XI ;And mask off
SACL STORE
CALL BYTE_OUT
LAC YI.8 ;HIGH BYTE
SACH STORE
LACK 255 ;MASK OFF
AND STORE
SACL STORE
CALL BYTE_OUT
LACK 255 ;LOW BYTE
AND YI ;And mask off
SACL STORE
CALL BYTE_OUT
LAC I
ADD ONE, I
SACL I ;I=I+2
BANZ NEXT ;UNTIL DONE
RET

: END OF DATA SENDING
: ********************************************
: START OF FFT ROUTINE
FFT
KLOOP LARP 1
LAC N2, 15
SACH N1, 1
SACH N2
ZAC
SACL IA
SACL J
LAR AR1, N2 ;AR1 CONTAINS J VALUE
MAR *- ;START AT N2-1
JLOOP
LAC TABLE ;TABLE IS FULL SIZE
ADD IA
TBLR SIN ;GET TWIDDLE FACTOR
ADD QUARTN
TBLR COS
LAC IA
ADD IE
SACL IA ;IA=IA+IE
LAC J, 1
SACL I ;I=J ADDRESS I IS TWICE
ILOOP
LAC I
ADD N2, 1 ;L=I+N2
SACL L
LAC PROGMEM ;BASE ADDRESS OF DATA
ADD I ;I VAL
TBLR XI ;READ REAL
ADD ONE
TBLR YI ;& IMAG
LAC PROGMEM
ADD L
TBLR XL
ADD ONE
TBLR YL

: COMPUTE BUTTERFLY
LAC XI
SUB XL
SACL XT ;XT=XI-XL
ADD XL. 1
SACL XI ;XI=XI+XL
LAC XI, 15 ;Scale

DOWN BY 2
SACH XI
LAC YI
SUB YL
SACL YT ;YT=YL-YL
ADD YL,1
SACL YI ; YI = YI + YL
LAC YI,15
SACH YI ; Scale down

BT 2

LT COS
MPY YT
PAC
LT SIN

MPY XT
SPAC ; YL = C*YT - S*XT
SACH YL ; Scaling done here no left

SHIFT 1.

MPY YT
PAC
LT COS
MPY XT

APAC ; XL = C*XT + S*YT
SACH XL ; Scaling done here

; Output results of the butterfly

LAC PROGMEM ; Base address of data
ADD I
TBLW XI ; Out real & Imag parts
ADD ONE
TBLW YI
LAC PROGMEM
ADD L
TBLW XL ; Output L value addr
ADD ONE
TBLW YL ; Out real & Imag parts

; Add increments for next loop

LAC I
ADD N1,1 ; I = I + 2 N1
SACL I
SUB HOLDN,1 ; I = I - 2 Hold
BLZ ILOOP ; While I + N < 2N
LAC J ; J = J + 1
ADD ONE
SACL J
BANZ JLOOP ; Loop if AR1 > 0

LAC IE,1
SACL IE ; IE = 2*IE
LARP 0
BANZ KLOOP ; Branch if AR0 > 0
RET

; Digit reverse counter for radix 2 FFT

BITREVERSE NOP
DRC2

ZAC
SACL L
SACL I
LARP 0
LAR ARO,HOLDN ; For I = 0 to N-2
MAR *-
MAR *-
SUB L ; For I < L then swap

DRLOOP
Dielital Signal Processing. [Appendix B]. page. 186

BGEZ NOSWAP

; Swap i and l values

LAC PROGMEM
ADD I
TBLR XI
ADD ONE
TBLR YI
LAC PROGMEM
ADD L
TBLR XL
ADD ONE
TBLR YL
LAC PROGMEM
ADD L
TBLW XI
ADD ONE
TBLW YI
LAC PROGMEM
ADD I
TBLW XL
ADD ONE
TBLW YL
LAC HOLDN

NOSWAP

INLOOP

SACL J ; J = N/2
LAC L
SUB J ; if L >= J
THEN

BLZ OUTL

SACL L ; L = L - J
LAC J, 15
SACH J ; J = J / 2
B INLOOP

OUTL

ADD J, 1
SACL L ; L = L + J
LAC I
ADD ONE, 1
SACL I ; Inc I
BANZ DRLOOP

: FFT Complete

WHOA

RET

:**************************

INT_SETUP

LDPK 0
DINT
LT ONE
MPYK INT_ROUTINE
PAC
SACL ADDRESS
LT ONE
MPYK >FFD
PAC
TBLW ADDRESS ; Store address of interrupt
; routine at int vector FFD
EINT
RET

:**************************

INT_ROUTINE

PUSH
IN IBM_VALUE,IBM
LACK 255 ;MASK
AND IBM_VALUE
SACL IBM_VALUE
LAC ONE
SACL IFLAG
EINT
POP
RET
:End of interrupt handler routine
:******************************************************
:Coefficient table size = 3n/4
:
:Twiddle factor data
:DATA SINE, FOR N= 0 TO 1 IN SIN(2*PI*N/8)
SINE DATA 0 ;NO 0
DATA 23170
:
:DATA COS, FOR N= 0 TO 6 IN COS(2*PI*N/8)
COSINE DATA 32767 ;NO 0 DO NOT USE 32768
DATA 23170
DATA 0
DATA -23170
DVALUE DATA >07FF
DATA >0
DATA >07FF
DATA >0
DATA >07FF
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
DATA >0
TOP END
:End of program AJE SGIHE MECH ENG. 1990.

The IBM Assembly program FFT8IBM.

TITLE FFT8IBM.ASM
Machine code routine to read in data from the C3M board into PC memory.

Called from Basic program using CALL offset

OFFSET = (101)*256 + (100)
SEG = (103)*256 + (102)

Modified to use interrupt check routine

uses fft length 8

Transfer SEGMENT PUBLIC
ASSUME CS:Transfer,DS:Transfer
ORG 100H
Start: JMP Place_vector

Atest DW 0A0Ah ; start code
Counter DB 0 ; loop counter
Store DW 0000 ; temp store
Value DW 40 DUP(?) ; data space 4*n+n
Nvalue DB 8 ; N value
Bite DB 00 ; temp value store

Delay PROC NEAR
PUSH AX
PUSH DX
MOV AX,003H
Lop2:
MOV DX,170H
Lop1:
DEC DX
JNZ Lop1
DEC AX
JNZ Lop2
POP DX
POP AX
RET

Delay ENDP

In_proc PROC NEAR
MOV DX,371H ; Port flag address
Not_set:
IN AL,DX ; get flag byte
TEST AL,128 ; Is bit 7 set?
JZ Not_set
DEC DX ; Set to data port address
IN AL,DX ; Get byte
MOV [BX],AL ; and store
RET

In_proc ENDP

Out_proc PROC NEAR
XOR AL,AL
MOV DX,372H
OUT DX,AL
Digital Signal Processing. [Appendix B]. page 189

```assembly
Mainroutine PROC FAR
    PUSH DS
    PUSH SS
    PUSH DX
    PUSH CX
    PUSH AX
    PUSH BX
    MOV AX, CS
    MOV DS, AX
    MOV BX, OFFSET Value
    XOR AX, AX
    MOV AL, NVALUE
    MOV CL, 01
    SHL AX, CL ; mult by 2 (2*N)
    MOV COUNTER, AL
    MOV CL, 08

    Starter:
        CALL Delay
        CALL In_proc ; Get MSB
        CMP AX, Atest ; Is it 0A0A
        JZ Start_up
        SHL AX, CL
        CALL Delay
        CALL In_proc
        CMP AX, Atest
        JZ Start_up
        SHL AX, CL
        JMP Starter

    Start_up:
        CALL Delay
        MOV Bite, AL
        CALL Out_proc
        MOV CL, COUNTER

    Data_in:
        MOV Counter, CL
        CALL Delay
        CALL In_proc
        MOV AL, [BX]
        MOV Bite, AL
        CALL Out_proc
        INC BX

        CALL Delay
```

Out_proc ENDP
**Digital Signal Processing. [Appendix B]. page 190**

CALL In_proc
MOV AL,[BX]
MOV Byte,AL
CALL Out_proc
INC BX

MOV CL,Counter
LOOP Data_in

POP BX
POP AX
POP CX
POP DX

;Restore registers
POP SS
POP DS
RET

Main Routine ENDP

Place_vector: MOV AX,0000H ;Set data seg to zero

MOV DS,AX
MOV BX,100H ;Set offset to 100h

MOV AX,OFFSET Main_routine
MOV [BX],AX ;Store start address

MOV AX,CS
MOV [BX+2],AX ;Store segment address

MOV AX,OFFSET Value
MOV [BX+4],AX ;Store No of data bytes

PUSH CS
POP DS ;Get CS into DS
MOV DX,OFFSET Finish
INT 27H
INT 20H ;Return to DOS

Finish:
Transfer ENDS
END Start

*The GWBasic FFT Display program.*

5 REM ** GRF8TST ***
10 REM ** LENGTH 8 FFT SCREEN DISPLAY ****
20 REM ** 1990 CARDIFF INSTITUTE ***
DIGITAL SIGNAL PROCESSING  [Appendix B], page 191

30 REM ** A J SHEPHERD Mech/Prod Section
40 REM ******************************************************
50 REM LOADS FFT8TST INTO C3M FIRST
60 REM RUNS FFT8IBM.COM FIRST
70 REM ******************************************************
80 REM ** START OF INFORMATION OUTPUTING SECTION ***
90 REM KEY OFF
100 DIM X(200), Y(200), XL(200), XH(200), TX(200), TY(200), XT(200), YL(200), YH(200)
110 DIM Y1(200), Y2(200)
120 SCREEN 9 : CLS : COLOR 7
130 KEY OFF
140 NV=7 : REM N VALUE LESS 1
150 GOSUB 1610 : REM SET UP DISPLAY MODE
155 GOSUB 370 : REM SET UP GRAPH BASE
160 GOSUB 260 : REM SET UP CALL ROUTINE
170 GOSUB 1240 : REM GET FFT VALUES
180 GOSUB 1470 : REM CALC PLOT HEIGHTS FOR THE FFT
190 GOSUB 1800 : REM REDRAW GRAPHS
200 GOSUB 900 : REM PLOT RESULTS
210 LOCATE 15,10 : BEEP
215 IF AV%=0 THEN VS="Off" ELSE VS="On"
218 IF PK=2 THEN TS="REAL & IMAG" ELSE TS="MAGN ONLY"
220 LOCATE 23,4:PRINT "Press S to STOP. C to CHANGE. A toggle AVERAGING.. A=", VS
230 GOSUB 1150 : REM STOP ROUTINE
235 IF J%=1 THEN 155 ELSE 160
240 LOCATE 1,1
250 END
255 REM ******************************************************
260 REM ROUTINE TO SET UP PARAMETERS
270 REM TO CALL ASSY LANG PROG C3M2MEM.COM
280 DEF SEG=0 : REM SET FOR DS SEG = 0
290 CS=PEEK(&H102)+PEEK(&H103)*256
300 OFFSET=PEEK(&H100)+PEEK(&H101)*256
310 NDATALOC=PEEK(&H104)+PEEK(&H105)*256
320 DEF SEG
330 C3M=OFFSET
340 DEF SEG=CS : REM SET CS SEG TO VALUE FOUND
350 RETURN
360 REM ******************************************************
370 REM SETUP GRAPH ON SCREEN
380 COLOR 14 :CLS
390 IF P%=1 THEN 730
400 LINE (35,5)-(540,129),1,BF
410 LINE (34,4)-(539,130),4,B
420 FOR N=0 TO 150 STEP 15
430 LINE (5,5+N)-(635,5+N),0
440 NEXT
450 X=4 : A=0
460 FOR N=0 TO 57 STEP 2
470 LOCATE 11,N+X : PRINT A : A=A+1
490 NEXT
500 FOR N=0 TO 8 STEP 2
510 LOCATE 10-N : PRINT N*.1
530 NEXT
540 LOCATE 1.2 : PRINT "1"
550 LOCATE 3,70:PRINT". Real.."
560 LOCATE 15,70:PRINT". Imag.."
570 REM ******************************************** NEXT ********************************************
580 LINE (35,160)-(540,279),1,BF
590 LINE (34,159)-(539,280),4,B
600 FOR N=0 TO 150 STEP 15
610 LINE (25,180+N)-(636,180+N),0
620 NEXT
630 X=4: A=0
640 FOR N=0 TO 57 STEP 7.7
650 LOCATE 22,N+X:PRINT A:A=A+1
670 NEXT
680 FOR N=0 TO 8 STEP 2
690 LOCATE 21-N: PRINT N*.1
710 NEXT
720 RETURN
725 REM ******************************************** ********************************************
730 REM MAGNITUDE ONLY PLOT
740 LINE (34,29)-(540,230),4,B
745 LINE(35,30)-(539,229),1,BF
750 FOR N=0 TO 190 STEP 19
760 LINE (30.225-N)-(535,225-N),0
770 NEXT
780 X=4: A=0
790 FOR N=0 TO 57 STEP 7.7
800 LOCATE 18,N+X
810 PRINT A:A=A+1
820 NEXT
830 X=0
840 FOR N=0 TO 16 STEP 2.7
850 LOCATE 17-N,1
860 PRINT X*.1:X=X+2
870 NEXT
875 LOCATE 8.70:PRINT"Magnitude"
880 RETURN
880 REM ******************************************** ********************************************
890 REM RESULTS PLOTTING ROUTINE
900 REM
910 COLOR 14
920 REM ***** REAL ********
930 IF P%=1 THEN 1070
940 ASTEP = 510/(NV+1)
950 FOR N=0 TO NV
960 PSET (35+ASTEP*N,67)
970 Y=Y1(N)*62: DRAW "U=Y;"
990 NEXT N
1000 REM ***** IMAG ********
1010 FOR N=0 TO NV
1020 PSET (35+ASTEP*N,220)
1030 Y=Y2(N)*60: DRAW "U=Y;"
1050 NEXT
1060 RETURN
1070 REM MAGNITUDE ONLY PLOT ****
1080 ASTEP=510/(NV+1)
1090 FOR N=0 TO NV
1100 PSET(35+ASTEP*N,222)
1110 Y=Y1(N)*180: DRAW "U=Y;"
1130 NEXT
1140 RETURN
1150 REM TIME DELAY FOR STOP KEY
1160 FOR N%=1 TO 100
1165 J%=0
1170 AS=INKEY$
1180 IF AS= "S" OR AS= "z" THEN 250
1190 IF AS= "C" OR AS= "c" THEN 1210
1195 IF AS= "A" OR AS= "a" THEN 1232
1200 NEXT N%
1205 RETURN
1210 IF P%=2 THEN P%=1 ELSE P%=2
1215 J%=1
1230 RETURN
1232 IF AV%=0 THEN AV%=1 ELSE AV%=0
1234 RETURN
1240 REM ** GET FFT VALUES FROM C3M -- USE C3M_FFT
1250 CALL C3M
1260 XMAX=0: YMAX=0: DATALOC=NDATALOC
1280 FOR N=0 TO NV
1285 X(N)=OLDX: Y(N)=OLDY
1290 XH(N)=PEEK(DATALOC): XL(N)=PEEK(DATALOC+1): YH(N)=PEEK(DATALOC+2): YL(N)=PEEK(DATALOC+3)
1300 TX(N)=256*XH(N)+XL(N): TY(N)=256*YH(N)+YL(N)
1310 IF XH(N)<=&H7F THEN 1350
1320 TXH(N)=XH(N) XOR &HFF: XL(N)=XL(N) XOR &HFF
1330 X(N)=-(XL(N)+XH(N)*256)
1340 GOTO 1360
1350 X(N)=XL(N)+XH(N)*256
1360 IF YH(N)<=&H7F THEN 1400
1370 YH(N)=YH(N) XOR &HFF: XH(N)=XH(N) XOR &HFF
1380 Y(N)=-(YL(N)+YH(N)*256)
1390 GOTO 1410
1400 Y(N)=YL(N)+YH(N)*256
1410 DATALOC=DATALOC+4
1420 IF X(N)>XMAX THEN XMAX=X(N)
1430 IF Y(N)>YMAX THEN YMAX=Y(N)
1435 IF AV%=0 THEN 1450
1436 X(N)=(X(N)+OLDX)/2 : Y(N)=(Y(N)+OLDY)/2
1450 NEXT N
1460 RETURN
1470 REM CALC PLOT HEIGHTS FOR FFT
1480 IF XMAX>YMAX THEN H%=XMAX ELSE H%=YMAX
1490 FOR N=0 TO NV
1495 X(N)=X(N)/H% : Y(N)=Y(N)/H%
1500 IF P%=2 THEN 1530
1510 Y1(N)=SQR(X(N)*2+Y(N)*2)
1525 GOTO 1550
1530 Y1(N)=X(N): Y2(N)=Y(N)
1550 IF Y1(N)>1 THEN Y1(N)=1
1555 IF Y1(N)<-1 THEN Y1(N)=-1
1560 IF Y2(N)>1 THEN Y2(N)=1
1565 IF Y2(N)<-1 THEN Y2(N)=-1
1570 NEXT N
1580 REM IF P=1 THEN BEEP
1590 P=0
1600 RETURN
1610 REM DISPLAY MODE REQUIRED
Set 4 Program listings.

The TMS32010 length 128 FFT Program.

IDT     C320FFT.SRC
; This is a 128 line DIF FFT.
;
; Note 07FF and F800 are the C3M board max and
; min values which are in Q15 format.
; The A/D converter is 12 bit and all data is extended to
; 15 bits and represented in Q15 format.
; Negative numbers are 2's complemented.
; Cooley Tukey radix 2 DIF FFT...
Digital Signal Processing. (Appendix B). page. 195

TMS32010 ASSEMBLY LANG. For the C3M unit

SINGLE FFT BUTTERFLY, COMPLEX DATA

USES LOOKUP TABLE FOR TWIDDLE FACTORS

SCALING DONE IN FFT /2 AT EACH STAGE (=/8)

REAL AND IMAGINARY DATA IN CONSECUTIVE LOC, S

USES INTERRUPT ROUTINE FOR DATA TRANSFER

to C3M BOARD.

AORG >200

N IS SIZE OF TRANSFORM N = 2**M

N EQU 128
M EQU 7

DATA MEMORY ALLOCATION

IFLAG EQU 33 ; INTERRUPT FLAG BYTE
IBM_VALUE EQU 32 ; DATA LOCATION
PROGMEM EQU 30 ; PROGRAM MEMORY ADDRESS
COMP EQU 31 ; CODE TO SEND TO HOST >0A0A
STORE EQU 2 ; TEMP STORE
HIGH EQU 3
LOW EQU 4
DELAY EQU 5 ; DELAY FOR SENDING DATA TO HOST >100
ADDRESS EQU 6
RATE EQU 7 ; SAMPLING RATE NO 6
XI EQU 8 ; ARRAY VALUE X(I)
YI EQU 9 ; ARRAY VALUE Y(I)
XL EQU 10 ; ARRAY VALUE X(L)
YL EQU 11 ; ARRAY VALUE Y(L)
XT EQU 12 ; TEMP REAL PART
YT EQU 13 ; TEMP IMAG PART
I EQU 14 ; 1ST INDEX
L EQU 15 ; 2ND INDEX
COS EQU 16 ; TWIDDLE FACTOR REAL
SIN EQU 17 ; TWIDDLE FACTOR IMAG
IA EQU 18 ; INDEX TO TWIDDLE FACTORS
IE EQU 19 ; INCREMENT TO IA
HOLDN EQU 20 ; CONTAINS VALUE N
QUARTN EQU 21 ; CONTAINS VALUE N/4
N1 EQU 22 ; INCREMENT TO I
N2 EQU 23 ; SEPARATION OF I AND N
J EQU 24 ; LOOP COUNTER
ONE EQU 25 ; CONTAINS VALUE 1
ZERO EQU 26 ; CONTAINS 0
PA0 EQU 0 ; PORT A0
PA1 EQU 1 ; PORT A1
TTEST EQU 27 ; TEST VALUE 54 (36 HEX)
TABLE EQU 29 ; LOCATION OF COEF TABLE

BEGIN PROG MEMORY SECTION

START CALL ADRATE
SENT CALL SETDATA
CALL INT_SETUP
CALL GETDATA
CALL FFT
CALL BITREVERSE
CALL SENDATA

B SENT

; LOOP

*:
END OF MAIN PROG LOOP

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

: START OF SET SAMPLE RATE
ADRATE

LACK 14

SACL RATE

OUT RATE, 2

RET

: END OF SAMPLE RATE

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

: START OF SET DATA ROUTINE

SETDATA

LDPK 0

LACK 1

SACL ONE

SACL IE

LACK 0

SACL ZERO

SACL XI

SACL XL

SACL YI

SACL YL

LT ONE

MPYK SINE

PAC

SACL TABLE

MPYK N

PAC

SACL HOLDN

SACL N2

LAC HOLDN, 14

SACH QUARTN

LACK > OA

SACL COMP

LT ONE

MPYK > FO

PAC

SACL DELAY

LT ONE

MPYK TOP

LACK > 10

APAC ; ADD P TO ACC

SACL PROGMEM

RET

: END OF SET DATA

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

**:

: START OF GET DATA ROUTINE

GETDATA

LAC PROGMEM

LARP 1

LAR 1, HOLDN ; DATA COUNT

MAR *-

WAIT

BIOZ WAIT

IN STORE.1 ; GET BYTE

TBLW STORE ; STORE IN PROGMEM

ADD ONE
** Digital Signal Processing. **

[Appendix B] page 197

```
TBLW ZERO ; Set imag part to zero
ADD ONE
BANZ WAIT ; Dec aux reg and branch until
 ; aux reg is zero
RET

; End of get data routine
; ************************************************************************
**

: Send start code routine
BYTE_OUT LAC STORE
OUT STORE,IBM
LAC DELAY

SLOW1 SUB ONE
BNZ SLOW1
LAC IFLAG
SUB ONE
BNZ BYTE_OUT
LAC IBM_VALUE
SUB STORE
BNZ BYTE_OUT
LAC ZERO
SACL IFLAG
RET

; ************************************************************************

: Start of data sending routine
SENCATA LARP 0
LAC PROGMEM ; Point to data in Progmem
SACL ADDRESS
LAR 0,HOLDN
MAR *- ; N-1
LAC ZERO
SACL IBM_VALUE
LAC ZERO
SACL I ; Offset for address
SACL IFLAG

: Start of main data send routine
LAC COMP
SACL STORE
CALL BYTE_OUT

NEXT LAC ADDRESS
ADD I ; Add offset
TBLR XI ; Real part
ADD ONE
TBLR YI ; Imag part
LAC XI,0 ; High byte
SACh STORE
LACK 255
AND STORE ; Mask off
SACh STORE
CALL BYTE_OUT

LACK 255 ; Low byte
AND XI ; And mask off
SACh STORE
CALL BYTE_OUT
```
LAC YI,B :HIGH BYTE
SACH STORE
LACK 255 ;MASK OFF
AND STORE
SACL STORE
CALL BYTE_OUT
LACK 255 ;LOW BYTE
AND YI ;AND MASK OFF
SACL STORE
CALL BYTE_OUT
LAC I
ADD ONE,1
SACL I ;I=I+2
BANZ NEXT ;UNTIL DONE
RET

;END OF DATA SENDING

;START OF FFT ROUTINE FFT
FFT LARK ARO,M-1 ;ARO CONTAINS K COUNTER
KLOOP LARP 1
  LAC N2,15
  SACH N1,1
  SACH N2
  ZAC
  SACL IA
  SACL J
  LAR AR1,N2 ;AR1 CONTAINS J VALUE
  MAR *- ;START AT N2-1
JLOOP
  LAC TABLE ;TABLE IS FULL SIZE
  ADD IA
  TBLR SIN ;GET TWIDDLE FACTOR
  ADD QUARTN
  TBLR COS
  LAC IA
  ADD IE
  SACL IA ;IA=IA+IE
  LAC J,1
  SACL I ;I=J ADDRESS I IS TWICE
  J REAL THEN IMAG
ILOOP
  LAC I
  ADD N2,1 :L=I+N2
  SACL L
  LAC PROGMEM :BASE ADDRESS OF DATA
  ADD I :I VAL
  TBLR XI ;READ

REAL
  ADD ONE
  TBLR YI ;IMAG
  LAC PROGMEM
  ADD L
  TBLR XL
  ADD ONE
  TBLR YL

: COMPUTE BUTTERFLY :
  LAC XI
  SUB XL
  SACL XT ;XT=XI-XL
ADD XL,1
SACL XI ; XI= XI+XL
LAC XI,15 ; Scale down by 2
SACH XI
LAC YI
SUB YL
SACL YT ; YT=YI-YL
ADD YL,1
SACL YI ; YI=YI+YL
LAC YI,15
SACH YI ; Scale down by 2
LT COS
MPY YT
PAC
LT SIN
MPY XT
SPAC ; YL=C*YT-
S*XT
SACH YL ; 1 left

OUT FOR SCALING

MPY YT
PAC
LT COS
MPY XT
APAC

YL=C*YT-

:S=XT+S*YT

OUT FOR SCALING

: Output results of the butterfly

LAC PROGMEM ; Base address of data
ADD I
TBLW XI ; Out real & Imag parts
ADD ONE
TBLW YI
LAC PROGMEM
ADD L
TBLW XL ; Output L value addr
ADD ONE
TBLW YL ; Out real & Imag parts

: Add increments for next loop

LAC I
ADD N1,1 ; I= I+2 N1
SACL I
SUB HOLDN,1 ; I-2HoldN
BLZ ILOOP ; while I+N<2N
LAC J ; J=J+1
ADD ONE
SACL J
BANZ JLOOP ; Loop if AR1>0
LAC IE.1
SACL IE ; IE=2*IE
LARP 0
BANZ KLOOP ; Branch if ARO>0
RET
; Digital Signal Processing. [Appendix B]. page 200

; DIGIT REVERSE COUNTER FOR RADIX 2 FFT

BITREVERSE

DRC2

  NOP

  ZAC

  SACL L

  SACL I

  LARP 0

  LAR ARO,HOLDN ; for I=0 to N-2

  MAR *-

  MAR *-

DRLOOP

  SUB L ; for I< L then swap

  BGEZ NOSWAP

; Swap I and L values

  LAC PROGMEM

  ADD I

  TBLR XI

  ADD ONE

  TBLR YI

  LAC PROGMEM

  ADD L

  TBLR XL

  ADD ONE

  TBLR YL

  LAC PROGMEM

  ADD L

  TBLW XI

  ADD ONE

  TBLW YI

  LAC PROGMEM

  ADD I

  TBLW XL

  ADD ONE

  TBLW YL

NOSWAP

  LAC HOLDN

  SACL J ; J=N

NLOOP

  LAC L

  SUB J ; if L>=J

  BLZ OUTL

  SACL L ; L=L-J

  LAC J,15

  SACH J ; J=J/2

  B INLOOP

OUTL

  ADD J,1

  SACL L ; L=L+J

  LAC I

  ADD ONE,1

  SACL I ; Inc I

  BANZ DRLOOP

; FFT Complete

WHOA

  RET

INT_SETUP

  LDPK 0

  DINT

  LT ONE

  MPYK INT_ROUTINE
Digital Signal Processing. [Appendix B], page 201

PAC
SACL ADDRESS
LT ONE
MPYK >FFD
PAC
TBLW ADDRESS ; Store address of interrupt
; routine at int vector FFD
EINT
RET

:*******************************
INT_ROUTINE  PUSH
IN IBM_VALUE,IBM
LACK 255 ; MASK
AND IBM_VALUE
SACL IBM_VALUE
LAC ONE
SACL IFLAG
EINT
POP
RET

; End of interrupt handler routine
:*******************************

: Coefficient table size = 3n/4

: Twiddle factor data
: data sine, for n = 0 to 31 in \text{SIN}(2\pi n/128)
SINE DATA 0 : NO 0
DATA 1608
DATA 3212
DATA 4808
DATA 6393
DATA 7962
DATA 9513
DATA 11040
DATA 12541
DATA 14011
DATA 15448
DATA 16848
DATA 18207
DATA 19522
DATA 20790
DATA 22007
DATA 23172
DATA 24281
DATA 25332
DATA 26321
DATA 27247
DATA 28108
DATA 28900
DATA 29623
DATA 30275
DATA 30854
DATA 31358
DATA 31787
DATA 32139
DATA 32414
DATA 32610
DATA 32728 ; No 31 \ldots 1/4 of 128
| COSINE | DATA   | COSINE | DATA   | COSINE | DATA   | COSINE | DATA   | COSINE | DATA   | COSINE | DATA   |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|        | 32728  |        | 32767  |        |        |        |        |        |        |        |        |        |
|        | 32610  |        |        |        |        |        |        |        |        |        |        |        |
|        | 32413  |        |        |        |        |        |        |        |        |        |        |        |
|        | 32138  |        |        |        |        |        |        |        |        |        |        |        |
|        | 31785  |        |        |        |        |        |        |        |        |        |        |        |
|        | 31356  |        |        |        |        |        |        |        |        |        |        |        |
|        | 30852  |        |        |        |        |        |        |        |        |        |        |        |
|        | 30273  |        |        |        |        |        |        |        |        |        |        |        |
|        | 29621  |        |        |        |        |        |        |        |        |        |        |        |
|        | 28897  |        |        |        |        |        |        |        |        |        |        |        |
|        | 28104  |        |        |        |        |        |        |        |        |        |        |        |
|        | 27244  |        |        |        |        |        |        |        |        |        |        |        |
|        | 26317  |        |        |        |        |        |        |        |        |        |        |        |
|        | 25328  |        |        |        |        |        |        |        |        |        |        |        |
|        | 24277  |        |        |        |        |        |        |        |        |        |        |        |
|        | 23168  |        |        |        |        |        |        |        |        |        |        |        |
|        | 22003  |        |        |        |        |        |        |        |        |        |        |        |
|        | 20784  |        |        |        |        |        |        |        |        |        |        |        |
|        | 19516  |        |        |        |        |        |        |        |        |        |        |        |
|        | 18201  |        |        |        |        |        |        |        |        |        |        |        |
|        | 16842  |        |        |        |        |        |        |        |        |        |        |        |
|        | 15442  |        |        |        |        |        |        |        |        |        |        |        |
|        | 14005  |        |        |        |        |        |        |        |        |        |        |        |
|        | 12535  |        |        |        |        |        |        |        |        |        |        |        |
|        | 11034  |        |        |        |        |        |        |        |        |        |        |        |
|        | 9506   |        |        |        |        |        |        |        |        |        |        |        |
|        | 7956   |        |        |        |        |        |        |        |        |        |        |        |
|        | 6386   |        |        |        |        |        |        |        |        |        |        |        |
|        | 4802   |        |        |        |        |        |        |        |        |        |        |        |
|        | 3205   |        |        |        |        |        |        |        |        |        |        |        |
|        | 1601   |        |        |        |        |        |        |        |        |        |        |        |
|        | -7     |        |        |        |        |        |        |        |        |        |        |        |
|        | -1615  |        |        |        |        |        |        |        |        |        |        |        |
|        | -3219  |        |        |        |        |        |        |        |        |        |        |        |
|        | -4816  |        |        |        |        |        |        |        |        |        |        |        |
|        | -6401  |        |        |        |        |        |        |        |        |        |        |        |
|        | -7970  |        |        |        |        |        |        |        |        |        |        |        |
|        | -9520  |        |        |        |        |        |        |        |        |        |        |        |
|        | -11047 |        |        |        |        |        |        |        |        |        |        |        |
|        | -12548 |        |        |        |        |        |        |        |        |        |        |        |
|        | -14018 |        |        |        |        |        |        |        |        |        |        |        |
|        | -15455 |        |        |        |        |        |        |        |        |        |        |        |
|        | -16854 |        |        |        |        |        |        |        |        |        |        |        |
|        | -18213 |        |        |        |        |        |        |        |        |        |        |        |
|        | -18528 |        |        |        |        |        |        |        |        |        |        |        |
|        | -20796 |        |        |        |        |        |        |        |        |        |        |        |
|        | -22013 |        |        |        |        |        |        |        |        |        |        |        |
|        | -23177 |        |        |        |        |        |        |        |        |        |        |        |
|        | -24287 |        |        |        |        |        |        |        |        |        |        |        |
|        | -25337 |        |        |        |        |        |        |        |        |        |        |        |
|        | -26326 |        |        |        |        |        |        |        |        |        |        |        |
|        | -27252 |        |        |        |        |        |        |        |        |        |        |        |
|        | -28112 |        |        |        |        |        |        |        |        |        |        |        |
|        | -28905 |        |        |        |        |        |        |        |        |        |        |        |
|        | -29627 |        |        |        |        |        |        |        |        |        |        |        |

DATA: \cos(2\pi n/128) for \( n = 0 \) to 63.
c. Additions to the IBM PC transfer program

This program is basically the same as that for the Set 2 IBM transfer program (IBMTIME.COM). The only difference being in the amount of data storage required - now 256 bytes, and the FFT length - now 128.

In order to accommodate the above changes the constant table at the start of the IBM PC transfer program for Set 4 has been altered. These changes are shown below. The remainder of the program being the same as that for Set 2.

```
TITLE FFT_128.ASM
:.................................
:Machine code routine to read in data from the
:C3M board into PC memory.
:
:Called from Basic program using CALL offset
:OFFSET = (101)*256 + (100)
:SEG = (103)*256 + (102)
:
:Modified to use INTERRUPT CHECK routine
:
:USES FFT LENGTH 128
:***********************
```
The GWBasic FFT Display program.

This is basically the same as the length 8 FFT but with a few minor changes to accommodate the longer FFT length.

5 REM ** FFTGRF **
10 REM ** 128 dif fft screen display ****
20 REM ** 1990 CARDIFF INSTITUTE ***
30 REM ** A J SHEPHERD Mech/Prod Section 
40 REM ****************************
50 REM load C305FFT into C3M first
60 REM RUN FFT3IBM.COM first
70 REM ****************************
80 REM ** start of information outputing section ***
90 REM key off
Digital Signal Processing. [Appendix B]. page 205

100 DIM X(200),Y(200),XL(200),XH(200),TX(200),TY(200),XT(200),YL(200),YH(200)
110 DIM Y1(200),Y2(200)
120 SCREEN 9 : CLS : COLOR 7
130 KEY OFF
140 NV=127 : REM N value less 1
150 GOSUB 1610 : REM set up display mode
155 GOSUB 370 : REM set up graph base
160 GOSUB 260 : REM set up call routine
170 GOSUB 1240 : REM get fft values
180 GOSUB 1470 : REM calc plot heights for the FFT
190 GOSUB 1800 : REM redraw graphs
200 GOSUB 900 : REM plot results
210 LOCATE 15,10 : BEEP
215 IF AV%=0 THEN V$="Off" ELSE V$="On"
218 IF P%=2 THEN T$="REAL & IMAG" ELSE T$="MAGN ONLY"
220 LOCATE 23,4: PRINT ".Press S. to STOP. C to CHANGE. A
toggle averaging ..A=."; V$
230 GOSUB 1150 : REM stop routine
235 IF J%=1 THEN 155 ELSE 160
240 LOCATE 1,1
250 END
255 REM *******************************************************
260 REM routine to set up parameters
270 REM to call assy lang prog C3M2MEM.COM
280 DEF SEG=0 : REM set for DS seg = 0
290 CS=PEEK(&H102)+PEEK(&H103)*256
300 OFFSET=PEEK(&H100)+PEEK(&H101)*256
310 NDATALOC=PEEK(&H104)+PEEK(&H105)*256
320 DEF SEG
330 C3M=OFFSET
340 DEF SEG=CS : REM set CS seg to value found
350 RETURN
360 REM *******************************************************
370 REM sets up graph on screen
380 COLOR 14 : CLS
390 IF P%=1 THEN 730
400 LINE (35.5)-(540,129),1,BF
410 LINE (34,4)-(539,130),4,B
420 FOR N=0 TO 150 STEP 15
430 LINE (5,5+N)-(635,5+N),0
440 NEXT
450 X=4:A=0
460 FOR N=0 TO 65 STEP 7.7
470 LOCATE 11,N+X
480 PRINT A : A=A+16
490 NEXT
500 FOR N=0 TO 8 STEP 2
510 LOCATE 10-N
520 PRINT N*.1
530 NEXT
540 LOCATE 1,2 : PRINT "1"
550 LOCATE 3,70: PRINT ".. Real .."
560 LOCATE 15,70: PRINT ".. Imag .."
570 REM *******************************************************
580 LINE (35,160)-(540,279),1,BF
590 LINE (34,159)-(539,280),4,B
600 FOR N=0 TO 150 STEP 15
610 LINE (25,180+N)-(636,180+N),0
620 NEXT
630 X=4 : A=0
640 FOR N=0 TO 65 STEP 7.7
650 LOCATE 22,N+X
660 PRINT A:A=A+16
670 NEXT
680 FOR N=0 TO 8 STEP 2
690 LOCATE 21-N
700 PRINT N*1
710 NEXT
720 RETURN
725 REM ****************************
730 REM magnitude only plot
740 LINE (34.29)-(540,230),4,B
745 LINE(35.30)-(539,229),1,BF
750 FOR N=0 TO 190 STEP 19
760 LINE (30,225-N)-(535,225-N),0
770 NEXT
780 X=4: A=0
790 FOR N=0 TO 65 STEP 7.7
800 LOCATE 18,N+X
810 PRINT A:A=A+16
820 NEXT
830 X=0
840 FOR N=0 TO 16 STEP 2.7
850 LOCATE 17-N,1
860 PRINT X*.1:X=X+2
870 NEXT
875 LOCATE 8,70;PRINT"Magnitude"
880 RETURN
890 REM ****************************
900 REM results plotting routine
910 COLOR 14
920 REM ****** REAL ******
930 IF P%=1 THEN 1070
940 ASTEP = 510/(NV+1)
950 FOR N=0 TO NV
960 PSET (35+ASTEP*N,67)
970 Y=Y1(N)*62
980 DRAW "U=Y;"
990 NEXT N
1000 REM ****** IMAG ******
1010 FOR N=0 TO NV
1020 PSET (35+ASTEP*N,220)
1030 Y=Y2(N)*60
1040 DRAW "U=Y;"
1050 NEXT
1060 RETURN
1070 REM Magnitude only plot  *****
1080 ASTEP=510/(NV+1)
1090 FOR N=0 TO NV
1100 PSET(35+ASTEP*N,222)
1110 Y=Y1(N)*180
1120 DRAW "U=Y;"
1130 NEXT
1140 RETURN
1150 REM time delay for stop key
1160 FOR N%=1 TO 100
1165 J%=0
1170 A$=INKEY$
1180 IF A$="S" OR A$="s" THEN 250
1190 IF A$="C" OR A$="c" THEN 1210
1195 IF A$="A" OR A$="a" THEN 1232
1200 NEXT N%
1205 RETURN
1210 IF P%=2 THEN P%=1 ELSE P%=2
1215 J%=1
1230 RETURN
1232 IF AV%=0 THEN AV%=1 ELSE AV%=0
1234 RETURN
1240 REM ** get fft values from c3m --use C3M_FFT
1250 CALL C3M
1260 XMAX=0; YMAX=0
1270 DATALOC=NDATALOC
1280 FOR N=0 TO NV
1285 X(N)=OLDX : Y(N)=OLDY
1290 XH(N)=PEEK(DATALOC): XL(N)=PEEK(DATALOC+1): YH(N)=PEEK(DATALOC+2): YL(N)=PEEK(DATALOC+3)
1300 TX(N)=256*XH(N)+XL(N): TY(N)=256*YH(N)+YL(N)
1310 IF XH(N)<=-&H7F THEN 1350
1320 XH(N)=XH(N) XOR &HFF: XL(N)=XL(N) XOR &HFF
1330 X(N)=-(XL(N)+XH(N)*256)
1340 GOTO 1360
1350 X(N)=XL(N)+XH(N)*256
1360 IF YH(N)<=-&H7F THEN 1400
1370 YH(N)=YH(N) XOR &HFF: YL(N)=YL(N) XOR &HFF
1380 Y(N)=-(YL(N)+YH(N)*256)
1390 GOTO 1410
1400 Y(N)=YL(N)+YH(N)*256
1410 DATALOC=DATALOC+4
1420 IF X(N)>XMAX THEN XMAX=X(N)
1430 IF Y(N)>YMAX THEN YMAX=Y(N)
1435 IF AV%=0 THEN 1450
1436 X(N)=(X(N)+OLDX)/2 : Y(N)=(Y(N)+OLDY)/2
1450 NEXT N
1460 RETURN
1470 REM calc plot heights for fft
1480 IF XMAX->YMAX THEN H%=XMAX ELSE H%=YMAX
1490 FOR N=0 TO NV
1495 X(N)=X(N)/H% : Y(N)=Y(N)/H%
1500 IF P%=2 THEN 1530
1510 Y1(N)=SQR(X(N)^2+Y(N)^2)
1525 GOTO 1550
1530 Y1(N)=X(N)
1540 Y2(N)=Y(N)
1550 IF Y1(N)>1 THEN Y1(N)=1
1555 IF Y1(N)<-1 THEN Y1(N)=-1
1560 IF Y2(N)>1 THEN Y2(N)=1
1565 IF Y2(N)<-1 THEN Y2(N)=-1
1570 NEXT N
1580 REM IF P=1 THEN BEEP
Digital Signal Processing. [Appendix B].  page. 208

1590 P=0
1600 RETURN
1610 REM Display mode required
1620 LINE (25,10)-(600,250),2,BF
1630 LINE (35,20)-(570,230),7,BF
1640 LINE (50,30)-(550,210),0,BF
1650 COLOR 14
1660 LOCATE 9,15:PRINT"... MENU..."
1670 LOCATE 11,15:PRINT"Press ..1. for Magnitude plot"
1680 LOCATE 13,15:PRINT"Press ..2. for Real & Imaginary plots"
1690 A$=INKEY$
1700 IF A$="1" OR A$="2" THEN 1720
1710 GOTO 1690
1720 P%=VAL(A$)
1730 BEEP:BEEP
1740 RETURN
1750 STOP
1800 REM ********************************************
1810 REM predraw graph bases
1820 COLOR 14
1830 IF P%=1 THEN 1950
1840 LINE (35,5)-(540,129),1,BF
1850 FOR N=0 TO 150 STEP 15
1860 LINE (5,5+N)-(635,5+N),0
1870 NEXT
1880 REM *****next ********
1890 LINE (35,180)-(540,279),1,BF
1900 FOR N=0 TO 150 STEP 15
1910 LINE (25,180+N)-(636,180+N),0
1920 NEXT
1930 RETURN
1940 REM ***** magnitude ********
1950 LINE(35,30)-(539,229),1,BF
1960 FOR N=0 TO 190 STEP 19
1970 LINE (30.225-N)-(535,225-N),0
1980 NEXT
2000 RETURN
2010 end
I. BIBLIOGRAPHY.


Digital Signal Processing. [Bibliography]. page 210


22. McClellan, J.H. and Rader, C.M. "Number theory in digital
Digital Signal Processing. [Bibliography ]. page. 211


