5 - T O N E Z V E I E N C O D E R A N A L Y S E R

BRIAN GEORGE WILSON

Thesis submitted in part fulfilment of the requirements for the Master's Diploma in Technology in the School of Electrical Engineering at the Cape Technikon.

CAPE TOWN

JANUARY 1993

DECLARATION

I declare that the contents of this thesis represents my own work and the opinions contained herein are my own and not necessarily those of the Technikon. This thesis has not been submitted before for any examination at this or any other institute.

BRIAN GEORGE WILSON

(Name of candidate)

(Signature of candidate)

DEDICATION

This thesis is dedicated to the memory of my dear parents the late Leonard Raymond Wilson and Dorothy Antoinette Wilson (nee Vollenhoven) who throughout their lives provided me with constant support and encouragement in obtaining a high level of education. It is also dedicated to my dear wife Lurinda and son Dylan for their patient, understanding and supportive manner which I continue to enjoy.

ACKNOWLEDGEMENT

I would like to thank Mr M.J. van Rensburg, Assistant City Electrical Engineer, Test and Metering Branch, Cape Town City Council and Mr Z. Bawolski from the Computer Section for making this project possible.

SUMMARY

This thesis describes the development of a 5-Tone Zentral Verband Electrotechische Industrie (ZVEI) Encoder Analyser. The 5-Tone ZVEI Encoder Analyser is used by the Radio Section of the Test and Metering Branch, which falls under the Electricity Department of the Cape Town City Council.

It assists the Quality Assurance Technician in determining whether the 5 tone ZVEI encoder, of the radio under test, is operating within the manufacturers specifications. Various manufacturers of radio equipment tender for the supply of mobile radios fitted with ZVEI tone encoders. The Radio Section are now capable of testing all the various radios and comparing the analysed ZVEI specifications of each manufacturer's radio. The results can be used to assist management in deciding which radio would be the most suitable for purchasing.

The development of the 5-Tone ZVEI Encoder Analyser involved the design and development of hardware and software. It was designed to be housed in a compact enclosure and to interface to a Motorola Communications System Analyser Model R-2001C. The RF output, from the radio under test, connects to the RF input of the Communications System Analyser. The demodulated output of the Communications System Analyser connects to the input of the 5-Tone ZVEI Encoder Analyser.

The software was designed using PLM-51 high level language to provide real-time analysis of various selective-calls (selcalls) received from the demodulated output of the Communications System Analyser. Once all 5 tones of the ZVEI selcall have been analysed the software background task is flagged and the analysed results are displayed as various MODES of display on a 16 character by 4 line dot matrix display.

The following parameters of the ZVEI selcall are analysed:

i) Frequency Digits.

ii) Frequency for each of the 5 tones.

iii) Tone Duration for each of the 5 tones.

iv) Frequency Error for the 5 tones.

v) Tone Duration Error for the 5 tones.

The design and development of the 5-Tone ZVEI Encoder Analyser was conducted at the Computer Section of the Electricity Department, Cape Town City Council.

OPSOMMING

Hierdie skripsie beskryf die ontwerp en ontwikkeling van 'n 5-Sein Zentral Verband Electrotechische Industrie (ZVEI) Enkodeerder Ontleedtoestel. Die Toets en Meet Afdeling van die Elektrisiteits Afdeling van die Kaapstadse Stadsraad het 'n 5-Sein ZVEI Enkodeerder Ontleedtoestel benodig vir gebruik in hul Radio Afdeling. Die Kwalitiiteitsversekeringstegnici moet die toestel gebruik om te verseker dat die 5 Sein ZVEI enkodeerders, van die radios wat getoets word, aan die vervaardiger se spesifikasies voldoen.

Verskeie radio vervaardigers doen aansoek om mobiele radios, toegerus met ZVEI sein enkodeerders, aan die Kaapstadse Stadsraad te verskaf. Die Radio Afdeling is nou in staat om die verskillende radio ZVEI spesifikasies te analiseer en te vergelyk met die van verskillende vervaardigers. Die ontledings word op bestuursvlak gebruik om die geskikte radio te keur, wat aangekoop moet word.

Die ontwikkeling van die 5-Sein ZVEI Enkodeerder Ontleed-toestel het die ontwerp en ontwikkeling van apparatuur sowel as programmatuur behels. Die apparatuur is ontwerp om in 'n kompakte verpakking te pas, en dien as koppelvlak na die " Motorola Communications System Analyser " Model R-2001C.

Die toets-radio se RF uittree-poort word gekoppel aan die " Communications System Analyser " se RF toevoer-poort. Die " Communications System Analyser " se gedemoduleerde uittree-poort dien as toevoer vir die 5-Sein ZVEI Enkodeerder Ontleed-toestel.

Die programmatuur is met behulp van die PLM-51 hoë vlak taal geskryf vir intydse ontleding van verskillende selektiewe oproepe ontvang, vanaf die " Communications System Analyser " se gedemoduleerde uittree-poort. Na die ontleding van al 5 seine van die ZVEI oproep word die stadium van program, wat op die agtergrond loop, gemerk en gestaak. Die ontlede resultate word vertoon as verskillende MODUSSE op 'n 16 karakter by 4 lyn punt-matriks vertoonkonsole.

Die volgende parameters van die ZVEI selektiewe oproep word ontleed:

i) Frekwensie Syfers.

ii) Frekwensie van afsonderlike 5 seine.

iii) Sein Lengte van afsonderlike 5 seine.

iv) Fout in Frekwensie van die 5 seine.

v) Fout in Sein Lengte van die 5 seine.

Die 5-Sein ZVEI Enkodeerder Ontleed-toestel is ontwikkel en ontwerp in die Rekenaar Afdeling van die Elektrisiteits Afdeling van die Kaapstadse Stadsraad.

OBJECTIVES

The objectives of this research was to accomplish the following:

i) To develop a 5-Tone ZVEI Encoder Analyser.

ii) To interface the 5-Tone ZVEI Encoder Analyser to an existing Communications System Analyser.

iii) To analyse a 5-tone ZVEI encoded signal as regards:

* The decoding of the encoded 5-Tone ZVEI Signal.
* The Frequency for each of the five tones.
* The Frequency Error for each of the five tones.
* The Tone Duration for each of the five tones.
* The Tone Duration Error for each of the five tones.

iv) To apply the PLM-51 Compiler as a development tool.

v) To decide on which Microprocessor would be the most suitable.

CONTENTS

CHAPTER 2 5-TONE ZVEI ENCODER ANALYSER PAGE

A N N E X U R E S PAGE

DESCRIPTION

LIST OF TABLES PAGE

DESCRIPTION

1. INTRODUCTION

1.1 General

Tone signalling methods and their performance recommendations have, as with the radio frequency channel specifications, grown around known device characteristics. The high cost of introducing digital transmission techniques in the past have resulted in the establishment of a number of tone signalling systems which were specified around the performance of known analog circuit elements, i.e coils and reeds.

Three companies, Motorola, GEC and Phillips, have signed an Intellectual Property Rights Agreement concerning the manufacture of tone signalling equipment which complies with the UK standard(1) for trunked private mobile radio of which the South African standard(6) forms a part.

The Zentral Verband Electrotechische Industrie (ZVEI) tone signalling system is currently being used by the Cape Town City Council. The characteristics of this tone signalling system includes the transmission of 5 sequential tones, each tone having a unique frequency representing a decimal digit. The duration for each of the 5 tones is 70 milli-seconds. This type of system enables selective calling, which is the transmission of a signal train enabling a predetermined station or a group of stations to be called exclusively.

1.2 Discussion of Signalling Systems

1.2.1 Selective-Calling Systems

Sequential tone signalling was implemented to give the mobile user a selective-call (selcall) facility and an automatic form of identification. Selective-calling has been most useful to remove the burden of continuous channel monitoring for otherwise occupied personnel.

Various sequential tone signalling methods have been introduced over the years to perform selective-calling functions. The specifications for some of these systems are shown in Table 1.1 and Table 1.2 below.

System	CCIR	DZVEI	ZVEI	EEA	EIA
Frequency range Hz	1124-2110	950-2400	1060-2600	1055-2110	459-1869
Frequencies defined	11	11	11	12	11
Frequency separation	6%	10%	10%	6%	450Hz
Tones transmitted (sequentially)	5	5	5	5	5
Address capacity	10^5	10^5	10^5	10^5	10^5
Call duration	500ms	350ms	350ms	200ms	165ms
Max calling rate/min	43	122	122	200	286
Frequency stability	±4Hz	±1.5%	±1.5%	±1%	±1Hz
Modulation index	70%	70%	70%	50-90%	70%

TABLE 1.1 Tone Signalling Systems Specifications(5)

System	Euro Signal	Pyecall (EIA)	CTCSS (EIA)	Swedish	DTMF Tones
Frequency range Hz	313.3 - 1153.1	330.5 - 928.1	67.0 - 250.3	1124-2110	697-1633
Frequencies defined	17	40	33	11	8
Frequency separation	8.5%	2.8%	3.5%	6%	10%
Tones transmitted (sequentially)	7	2	1	7	No limit 2 simul.
Address capacity	$7x10^4$	960	33	10^4	N/A
Call duration	800ms	400ms	Cont.	1.3s	Dial tm.
Max calling rate/min	75	100	N/A	40	<6
Frequency stability	±0.1%	±0.35%	±0.5%	±0.5%	±1.8%
Modulation index	90%	50-60%	10-20%	70-100%	35%

TABLE 1.2 Tone Signalling Systems Specifications(5)

The first of these tone systems was the two tone Pyecall type based on the Electronic Industries Association(5) (EIA) geometric progression of frequencies. Two different tones are selected from a group of 40 frequencies (in some systems 90 frequencies are available) and these are transmitted sequentially. Frequency separation is 2.8% and thus high 'Q' decoding circuits are used and tone bursts of more than 200ms are employed. Reliability of signalling is achieved under poor propagational conditions. Two tone signalling is very common for city-wide radio paging.

The five-tone systems have 11 frequencies to choose from (10 plus repeat tone). There is a 6% to 10% spacing between frequencies. Lower 'Q' circuits can be used so that a trade-off is made between signal to noise performance and signalling rate with a resulting relaxation of frequency stability(5).

The International Radio Consultative Committee (CCIR) and (EIA) systems(5) were not intended for encoding by commercial mobile and portable equipment, but were defined for maritime and paging applications respectively.

The Electronic Engineering Association (EEA) approach(5) has an acceptable frequency stability specification and a short call duration. This makes it suitable as a rapid tone transmission system.

Dual Tone Multi-Frequency (DTMF) is compatible with standard telephone signalling specifications. However, its performance under the relatively noisy mobile radio channel(5) is limited by the need to transmit two simultaneous tones, each with a 6dB reduced deviation. The potential for intermodulation is also a hazard although its dialling capability is achieved at very low cost. It has no advantage for polling and interrogation applications.

1.2.2 The ZVEI Choice

Zentral Verband Electrotechische Industrie (ZVEI) chose a signalling system with wider frequency separation, thus reducing the stability specification and achieving a shorter signalling time. Although this is ideal for mobile to local controlled base equipment or mobile to mobile, the higher frequencies make it unsuitable for remotely controlled schemes(5).

The repeat tone of 2600Hz is too high for 12,5kHz radio frequency channelling and conflicts with control tones used for switching and voting purposes. The ease with which the 5-tone (decimal digit) ZVEI system can be given an address change and the flexibility which a high address capacity (10^5) provides, makes it a popular choice.

1.3 Motivation

A cost effective method of enhancing the functions of the Motorola Communications System Analyser Model R-2001C was the prime motivating factor for the need to develop a 5-Tone ZVEI Encoder Analyser. The Motorola Communications System Analyser is used extensively during the QA testing of the radio transmitter and receiver stages.

An inadequacy of this model's transmitter-stage testing was verifying the specifications of the 5-Tone ZVEI Encoder installed in the Radio. This inadequacy lead to the proposal for the design and development of a 5-Tone ZVEI Encoder Analyser.

The 5 Tone ZVEI Encoder Analyser had to fulfil the following general specifications:

i) It had to be accurate.

ii) It had to be relatively cheap.

iii) It had to be menu driven and easy to operate.

iv) It had to be portable and compact so as to blend in with the QA workstation where it would be used.

1.4 Design Approach

1.4.1 Choice of Microprocessor

The Intel 87C51FA microcontroller was selected as the most suitable controlling device for the 5-Tone ZVEI Encoder Analyser for the following reasons:

i) The 87C51FA is a single chip control-orientated microcontroller.

ii) Provides a reduction in the overall chip count.

iii) Being a CMOS device, its nominal current consumption is typically 15mA @ 12MHz, thus complementing the requirements for a small, compact power supply.

iv) Hardware and software development tools were available with the Intel In-Circuit Emulator (ICE-5100), to emulate the 8051 family of microcontrollers.

1.4.2 Peripheral Hardware Selection

The peripheral hardware was researched to enable effective interfacing to the 87C51FA microcontroller. Of the range of displays, keyboard encoders, signal conditioners, tone decoders and buffers available, the components selected were best suited in terms of:

i) Functionality.

ii) Sophistication.

iii) Cost.

iv) Ease of interfacing.

v) Power requirements.

1.4.3 Need for Emulation

An In-Circuit Emulator (ICE) consists of hardware and software tools which are used during the design and development of microprocessor applications.

The features of the ICE-5100 was utilized to assist with the following:

i) Orderly and efficient debugging of target system hardware and software.

ii) Improved development time in terms of editing source code, compiling, linking, producing a Intel hex file and transferring control to the emulator processor module for testing.

1.4.4 Emulator

A complete working knowledge of the ICE-5100 was researched to assist with the hardware and software development for the 5-Tone ZVEI Encoder Analyser. The serial port of an IBM PC, XT or AT, is connected to a Controller Pod which in turn is linked to a Processor Module via a twisted pair ribbon cable. During emulation the Target Adaptor of the Processor Module replaces the microcontroller of the system currently under development.

1.4.5 5-Tone ZVEI Encoder Analyser

The following approach was used in the final design and development of the 5-Tone ZVEI Encoder Analyser:

i) Suitable hardware was researched and a schematic diagram was drawn using ORCAD.

ii) A wire-wrap prototype was built.

iii) The software was modularly designed using various algorithms and flowcharts.

iv) The hardware and software was debugged with the aid of the ICE-5100.

v) The prototype was interfaced to the Motorola Communications System Analyser and the ZVEI specifications of various mobile radios were analysed.

vi) A Printed Circuit Board (P.C.B.) was manufactured and together with keypad and display, housed in a confined enclosure.

2. 5-TONE ZVEI ENCODER ANALYSER

2.1 Initial Concepts

2.1.1 The ZVEI Standard Frequencies

Table 2.1 represents a table of the ZVEI tone frequencies, together with their corresponding 4-bit binary codes and hexadecimal characters.

ZVEI Tone Frequency (Hz)	Q3	Q2	Q1	Q0	Hexadecimal Character
2400	0	0	0	0	0
1060	0	0	0	1	1
1160	0	0	1	0	2
1270	0	0	1	1	3
1400	0	1	0	0	4
1530	0	1	0	1	5
1670	0	1	1	0	6
1830	0	1	1	1	7
2000	1	0	0	0	8
2200	1	0	0	1	9
2800	1	0	1	0	A
810	1	0	1	1	B
970	1	1	0	0	C
886	1	1	0	1	D
2600	1	1	1	0	E
No-Tone	1	1	1	1	F

Table 2.1 ZVEI tone frequencies as 4-bit binary codes(4)

Referring to Table 2.1 on page 2-1, fifteen tone frequencies are defined for the 5-Tone ZVEI sequential tone signalling system. The tone frequency separation is 10%. The number 12345 is represented by a sequential transmission of tone frequencies 1060 Hz, 1160 Hz, 1270 Hz, 1400 Hz and 1530 Hz by mobile user number 12345.

Hexadecimal character A represents a group call. This is implemented as a train of transmitted frequencies representing the numbers A, 2, 3, 4 and 5. This results in units / mobile units 02345 to 92345 being called. Hexadecimal characters B, C and D represent address suffix tones. Tone E is a repeat tone and transmission of tones representing hexadecimal characters 1, 2, 3, E and 5 results in an identification code for mobile user no. 12335.(Refer to ANNEXURE 1)

2.1.2 Preliminary Hardware and Software Research

Prior to any complex software coding, the 5-Tone ZVEI signal received from various ZVEI selective-call tone encoders / decoders was evaluated with the aid of an HP digital oscilloscope. The dynamic characteristics of the FX5070A, FX2030Z, FX003QZ, FX102LG and FX202QZK were examined.(Refer to ANNEXURE 1)

Tables 2.2 to 2.4 show the differences between some of the features and dynamic characteristics of the ZVEI selective-call tone encoders / decoders.

Device	FX5070A
Type	ZVEI Tone Encoder / Decoder
Tone Duration	63ms (min) 70ms (typ) 77ms (max)
Logic Sig. Output	Not Available
No. of Pins	32
Oscillator	Uses External RC Components

Table 2.2 Features & Dynamic Characteristics of the FX5070A(2)

Device	FX2030Z
Type	ZVEI Tone Encoder / Decoder
Tone Duration	68ms (min) 70ms (typ) 72ms (max)
Logic Sig. Output	Not Available
No. of Pins	42
Oscillator	Uses External 560kHz Resonator

Table 2.3 Features & Dynamic Characteristics of the FX2030Z(2)

Device	FX0030Z (FX102LG & FX2020ZK)
Type	ZVEI Tone Decoder
Tone Duration	Dependant on Encoder Specifications
Logic Sig. Output	Available
No. of Pins	16 (24 : 28)
Oscillator	Uses External 560khz Resonator

Table 2.4 Features & Dynamic Characteristics of the FX0030Z(2)

A thorough investigation and comparison of the dynamic characteristics and variations between the devices resulted in the selection of the FX102LG and FX202QZK for the signal front-end stage of the 5-Tone ZVEI Encoder Analyser.

This selection was made for the following reasons:

i) The FX5070A consists of a tone encoder and decoder in a single hybrid package. Hardware was only required for decoding the 5-Tone ZVEI signal.

ii) The FX2030Z is a very expensive 42 pin hybrid device incorporating a ZVEI tone encoder and decoder. The dynamic characteristics of the tone decoder(2) are superior to those of the FX5070A, however, only the FX003QZ decoder section was of importance for this particular application.

iii) The FX003QZ consists of the FX102LG and FX202QZK devices. At first the FX003QZ selective-call tone decoder appeared to meet all the requirements for the front-end stage. This was indeed so, except that there was no pin connection available on this package as a node for frequency measurement. This led to the use of the discrete, surface mount, devices of the FX102LG and FX202QZK.

Figure 2.1 on page 2-6 is a waveform showing **all five tones** of the encoded 5-Tone ZVEI selcall. The signal was received from a test unit utilizing a FX5070A tone encoder / decoder. Figure 2.2 on page 2-7 represents the **frequency change between two of the tones included** in the encoded 5-Tone ZVEI selcall as shown in Figure 2.1. In addition, the waveform shows how the tones were modulated about a sine wave.

The waveforms obtained revealed the following important characteristics:

i) The duration obtained for all 5 tones, realized after conducting numerous tests, actually ranged from 374ms to 390ms in practice. This is outside the typical specification of 350ms(2).

ii) Table 2.5 on page 2-8 represents the desired **total tone duration** for all five tones and the measured durations received from the FX5070A.

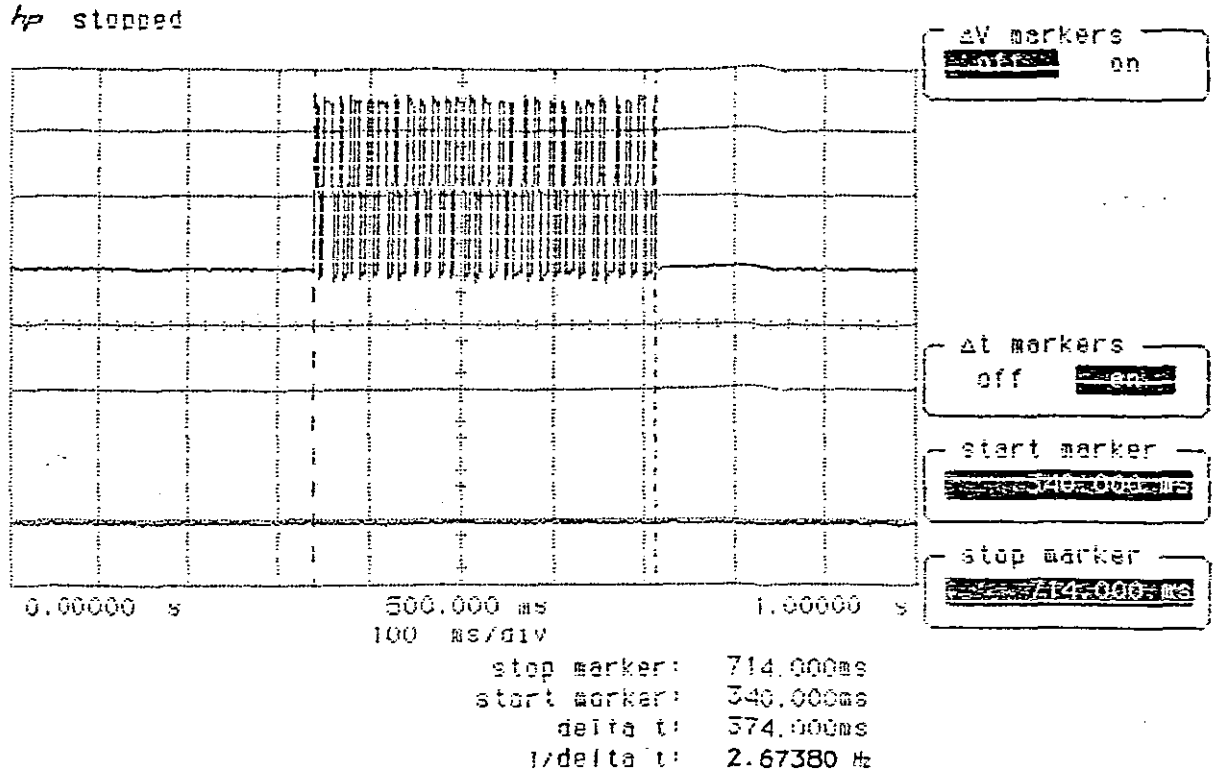


Figure 2.1 Waveform showing all five tones of the encoded 5-Tone ZVEI selcall

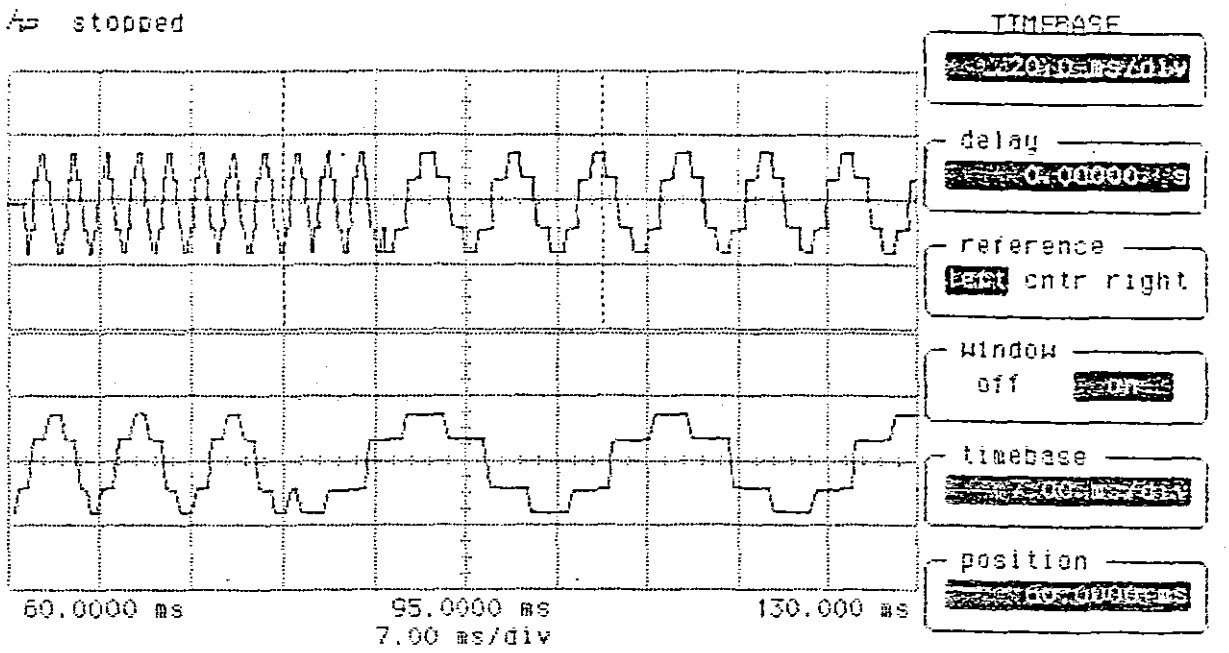


Figure 2.2 Frequency Change between two of the tones shown in Figure 2.1

Test No.	Desired Tone Duration	Tone Duration Measured
1	350ms	379ms
2	350ms	385ms
3	350ms	378ms
4	350ms	384ms
5	350ms	390ms
6	350ms	389ms
7	350ms	380ms
8	350ms	374ms
9	350ms	376ms
10	350ms	382ms

Table 2.5 Desired total Tone Duration for all five tones and measured values received from the FX5070A.

A decoded **repeatable** tone duration of typically 70ms for each of the five tones would be most desirable, however, the specifications for the FX5070A encoder allows for the generation of tones having tone durations ranging from 63ms to 77ms(2).

Figure 2.3 on page 2-9 shows the data change signals which are output by the FX202QZK when a new ZVEI tone is detected. Data change signals 1 and 6 are of significant importance.

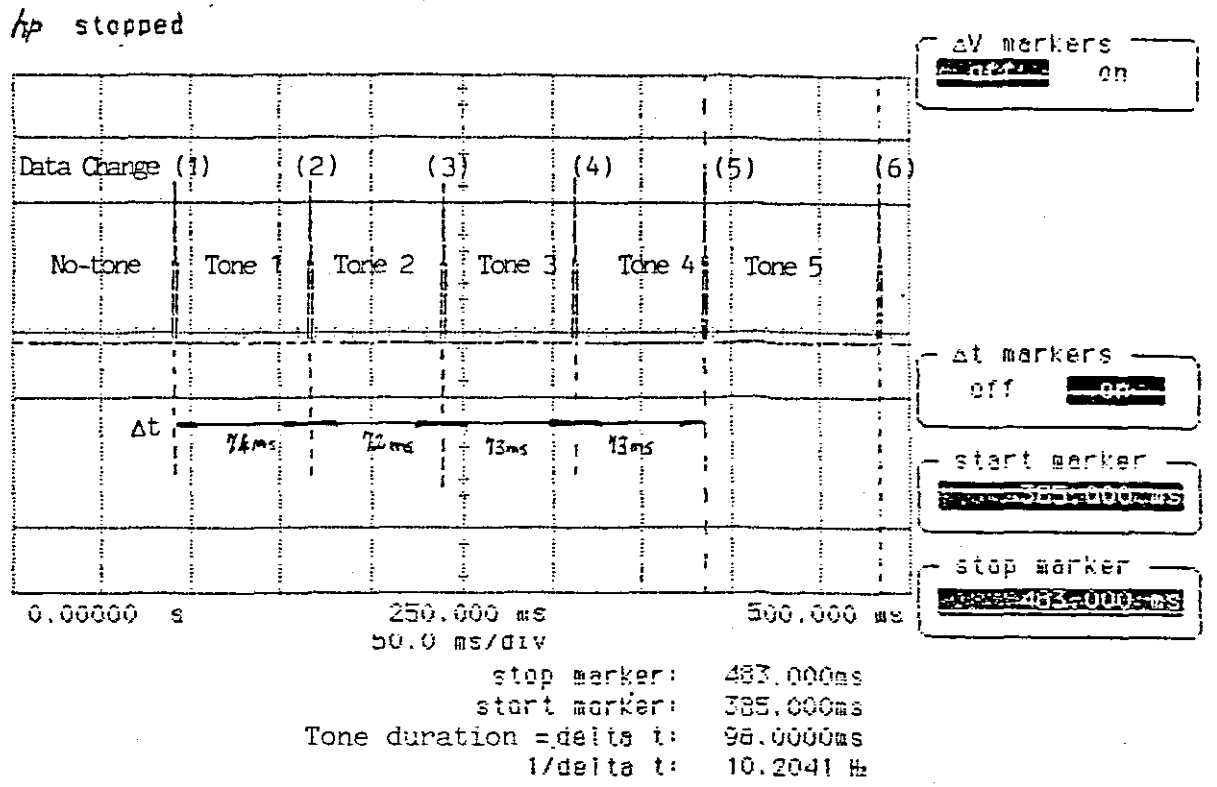


Figure 2.3 Data Change Signals output by the FX202QZK.
Tone Duration 5 includes the "no-tone" period

Data change signals 1 and 6 are generated after the decoder detects a change from "no-tone" to tone (data change signal 1) and from tone to "no-tone" (data change signal 6). The "no-tone" period is the **time taken by the decoder** to "recognise" the presence (Tone 1) or absence (Tone 5) of a **valid** ZVEI tone. Only once this condition has been established is a data change signal generated.

Figure 2.4 on page 2-11 shows the real-time relationship between tone frequencies 1 to 5 as output by the FX102LG and the generation of the data change signal for each change in tone frequency. The data change signal is generated by the FX202QZK.

The important practical characteristics of this waveform are:

i) The tone frequencies measured at the output of the FX102LG are four times that of the input frequency. (Refer to APPENDIX I for Waveforms of Tone Digit Frequencies).

ii) Data change signals 1 to 5 are activated 17ms to 19ms after each successive change in the tone frequency. An average "no-tone" period of 25ms was measured at the end of tone 5 before data change signal 6 was generated.

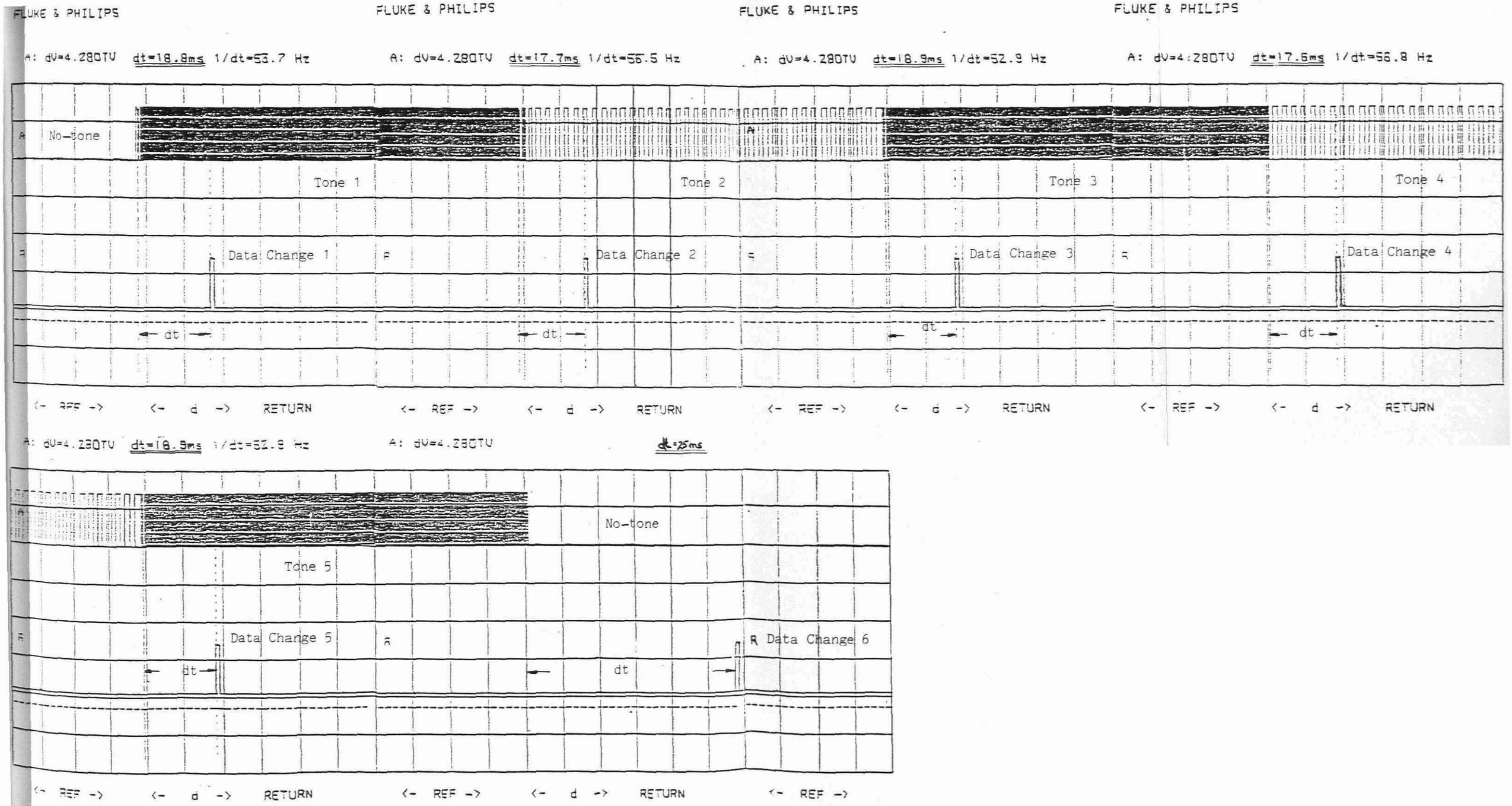


Figure 2.4 Real-time relationship between Tone Frequencies and Data Change Signals using a FX102LG and FX202QZK

A program to test the fundamental operating features of the 5-Tone ZVEI Encoder Analyser was compiled, incorporating software to test the following:

i) A routine to initialize, address and write a character to the dot-matrix display.

ii) Read a key pressed on the keypad and store its value into memory.

iii) Measure the tone duration received from a 8031 data change simulator board. See APPENDIX E, F and G for the flow charts, software listing and data change signal generated by the 8031 data change signal simulator board.

2.1.3 Initial Design Philosophy

Figure 2.5 represents a block diagram of the initial design philosophy for the 5-Tone ZVEI Encoder Analyser.

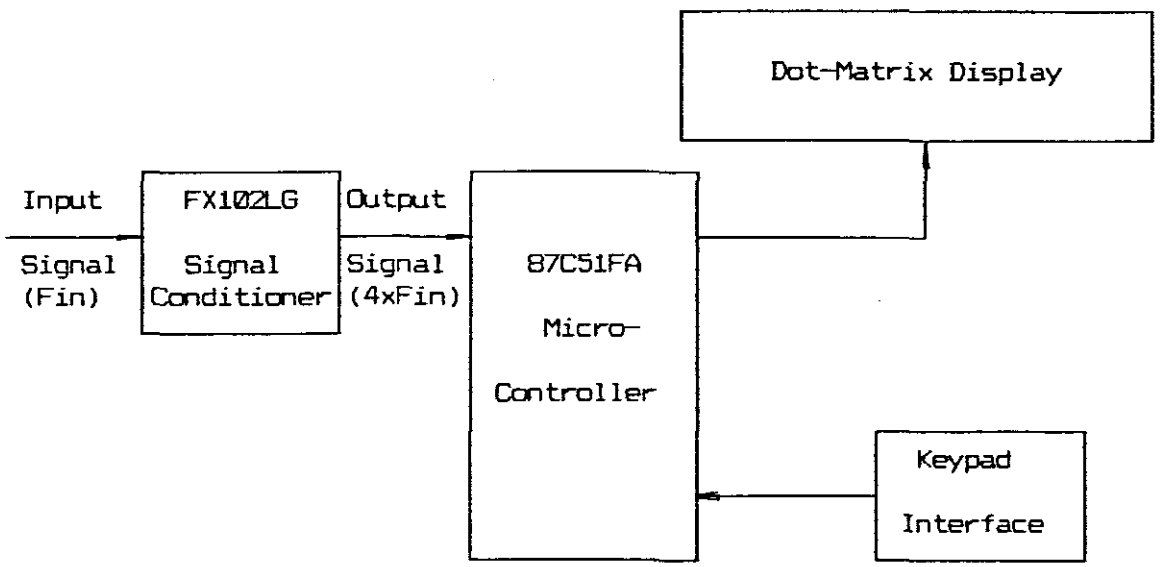


Figure 2.5 Block Diagram showing initial design philosophy

The initial design philosophy entailed the use of the FX102LG as the sole hardware device for signal conditioning, prior to being interfaced to the 87C51FA microcontroller. The FX102LG strips the noise from and "squares" any incoming tone without adjustment. It can program any ZVEI tone with a signal to noise ratio of 0dB or better(2).

The functions performed by a FX202QZK Tone Decoder were to be implemented by the control hardware and software of the microcontroller. The selcall tones, as received in real-time (see Figure 2.4 on page 2-11), would be input to the external I/O pin of the 87C51FA's Programmable Counter Array (PCA) port.

The initial design algorithm included:

i) The utilization of the 87C51FA's PCA Compare / Capture Register to compare the respective input frequencies to frequencies programmed in the on-board EPROM of the microcontroller.

ii) Software determination of the change in tone frequency, eliminating the need for a data change signal input to the microcontroller.

iii) Software determination of the 4-bit binary code representing the hexadecimal characters.

2.1.4 Final Design Technique

The final design technique utilized in the design of the 5-Tone ZVEI Encoder Analyser is largely similar to the initial design algorithm as discussed in 2.1.3 except that a FX202QZK peripheral hardware device was used to supplement the hardware to provide effective analysis of the 5-Tone ZVEI signal.

The reasons for incorporating this device are as follows:

i) The measurement of a unique frequency proved effective when utilizing the algorithm in 2.1.3, however, practical implementation of the software algorithm proved very difficult where multiple valid frequencies were to be measured and analysed.

ii) False data change signals were flagged by the software implementing the algorithm in 2.1.3. This was due to slight frequency variations being interpreted as the next tone in the selcall sequence. Software filtering, incorporating frequency change window timers, proved extremely complex for the range of frequencies to be analysed.

iii) The FX202QZK served to provide a suitable hardware device to complement the FX102LG's functionality and simplified the design of already complex software procedures without any detriment to the effective analysis of the 5-Tone ZVEI sequence. The peripheral circuitry completing the circuit diagram of the 5-Tone ZVEI Encoder Analyser was selected for reasons as discussed in Section 1.4 on page 1-5.

2.2 System Integration

2.2.1 Interface to Communications System Analyser

A subminiature earphone socket is located on the rear of the 5-Tone ZVEI Encoder Analyser. One end of a screened interface cable is terminated with a subminiature earphone plug which is inserted into the socket on the 5-Tone ZVEI Encoder Analyser. The other end of the interface cable is terminated with a BNC connector which gets connected to the "DEMOD OUT" of the Communications System Analyser.

The RF output of the radio under test is connected to the "RF IN/OUT" connector of the Communications System Analyser via a screened test cable. An interface diagram showing these connections can be found in Figure 2.6 on page 2-17.

Although suitable RF shielding has been provided in the 5-Tone ZVEI Encoder Analyser, the unit should not be placed too close to the radio under test. For ease of use and operator comfort, the radio and 5-Tone ZVEI Encoder Analyser are best located at a 30 degree angle, at opposite ends to the Communications System Analyser.

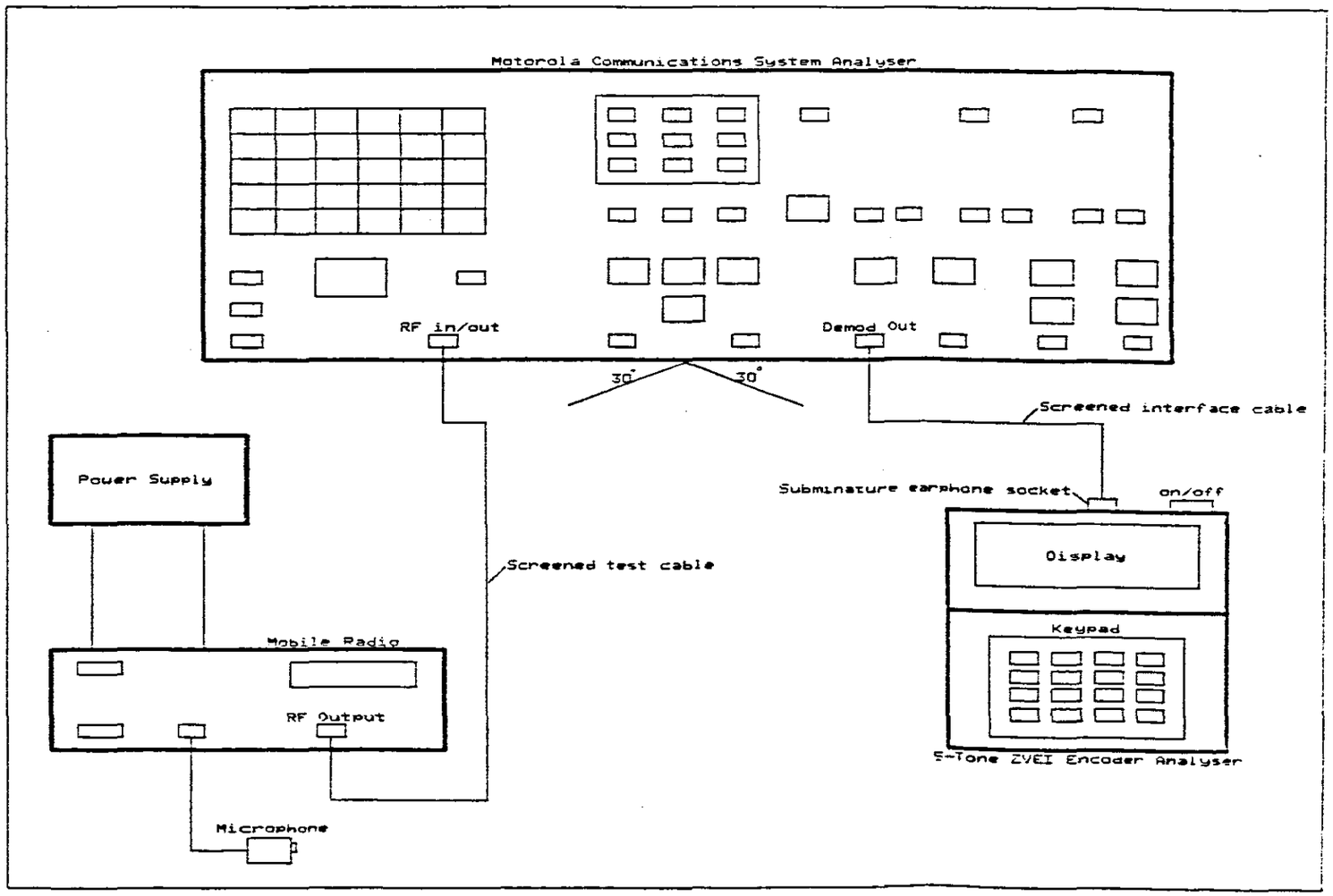


Figure 2.6 Diagram Showing Interface Connections to Motorola Communications System Analyser (Model R-2001C)

2.2.2 Communications System Analyser Settings

The Communications System Analyser is set-up to operate in the "Monitor Mode". After all interface connections have been made, as discussed in Section 2.2.1 (page 2-16), the Communications System Analyser is set-up as follows:

i) Power up the Communications Systems Analyser.

ii) Set-up the transmit frequency, via keypad, to Monitor FM 146,1500 Mhz (CCC Forestry Channel).

iii) Adjust the step attenuator of the RF Section to - 40dB.

iv) Set the Image/Dplx switch to "high".

v) Set the BW Switch to "Narrow".

vi) Turn the volume potentiometer to the mid position.

vii) Set BFO potentiometer "off/on" to "off".

viii) Set the Display LED to select "Gen / Mon Mtr".

ix) Adjust the squelch potentiometer to open the squelch.

x) Set the Function LED to "FM (mode)".

xi) Set the Function Switch to select "Pwr Mon".

The radio may now be keyed and as the five sequential tones are received, the signal level LED flashes indicating good reception of the 5-Tone ZVEI signal.

2.3 Theory of Operation

2.3.1 Hardware

Figure 2.7 on page 2-20 represents a block diagram of the 5-Tone ZVEI Encoder Analyser. The complete circuit diagram can be found in APPENDIX A page 7-1. As shown in the block diagram, the 5-Tone ZVEI Encoder Analyser consists of 4 main blocks, namely:

i) 87C51FA Microcontroller.

ii) FX102LG and FX202QZK.

iii) Dot Matrix Display.

iv) Keyboard Encoder.

The core of the 5-Tone ZVEI Encoder Analyser is the 87C51FA microcontroller. In operation, the demodulated signal from the Communications Systems Analyser is applied to the signal input of the FX102LG. Figure 2.1 on page 2-6 shows the waveform of a typical input signal.

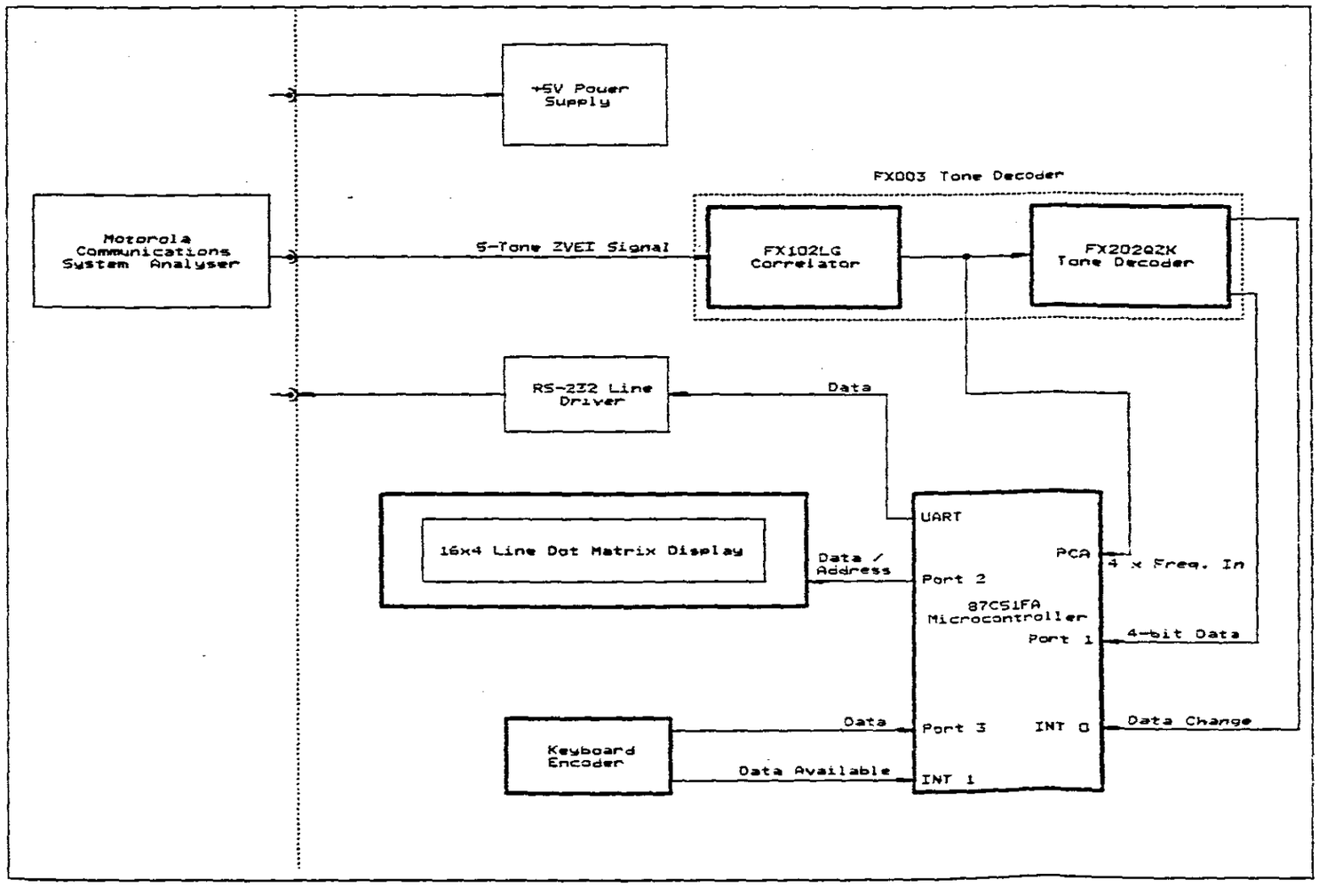


Figure 2.7. Block Diagram of the 5-Tone ZVEI Encoder Analyser

The output of the FX102LG has two destinations namely:

i) Logic signal input to the FX202QZK Tone Decoder.

ii) Buffered input signal to the PCA port of the microcontroller.

The FX202QZK, detects an input frequency falling within any of the fifteen tone channels programmed on chip and outputs the hexadecimal character in 4-bit binary code. A data change signal is also generated, indicating that a new 4-bit binary code has been latched at the data outputs of the FX202QZK (Table 2.1 page 2-1). The data change signal is also used to serve as an indication for the change in tone frequency. This indication is imperative for tone duration measurement (Figure 2.4 page 2-11).

The data change signals strobe the microcontroller which performs real-time analysis of the tone duration. The data change signal is interfaced to the first interrupt (INT 0) of the microcontroller.

The signal from the FX102LG to the microcontroller is gated to measure the frequency of each tone as it is processed during the real-time analysis of selcalls. The real-time relationship of tone-frequency change to generation of data change signal is shown in Figure 2.4 on page 2-11.

Once all the relevant parameters have been analysed by the real-time interrupts, the results are ready for displaying. An intelligent display is connected directly to the I/O ports of the microcontroller. The display contains all the necessary hardware to control the 16-character by 4-line dot matrix display.

Having initialized the display, the microcontroller sets up the Display Data (DD) address and then writes the data to the display (Refer to ANNEXURE 2 on page 17-1).

When a key on the keypad is pressed, its code is encoded by the keyboard encoder and applied, via buffers, to the I/O port pins of the microcontroller. The keyboard encoder eliminates key roll-over and also provides a debounced code at the output. This debounced code is indicated by the data available output which is buffered and interfaced to the second interrupt (INT 1) of the microcontroller.

If the interrupt has not been masked by the software, the microcontroller will read the encoded keypad data and perform the necessary display operations.

Although the uploading of the data has not been a requirement of this study, hardware for RS-232 operation has been provided for any possible software enhancements incorporating the uploading of data. An on-board power supply consisting of a mains transformer, bridge rectifier, regulator and transient voltage suppressor has been included to supply +5V to the unit. The complete P.C.B has been screened with a suitably grounded radio frequency shield.

2.3.2 Software

The control program for the 5-Tone ZVEI Encoder Analyser is contained in the 8k EPROM on board the 87C51FA. The $\overline{EA}$ pin of the microcontroller is connected to Vcc, forcing program execution from internal memory range 0000H through 1FFFH. The 87C51FA implements 256 bytes of on chip data RAM, used to store stack (upper 128 bytes, 80H to FFH) and variable values. Each of the four ports of the 87C51FA may be configured, by the software, to be used as I/O ports.

Port 3 may be configured not only as port pins, but may also serve the functions of various special features(3). This is achieved by writing a "1" into the corresponding bit latch in the port Special Function Register(3).

The control program occupies 5,2k of on-board Program Memory. This is quite a large portion of the available memory space and is mainly due to the many modes of display operation (Table 2.7 page 2-29).

The main software modules to execute are implemented as:

i) A Background Task.

ii) Real-Time Analysis of the 5-Tone ZVEI signal.

iii) Timer Ø, tone duration timer.

iv) Timer 1, interrupt to gate frequency.

v) Computation of various parameters based on data analysed during the real-time interrupts.

vi) Displaying of the analysed and computed data.

The program commences by initializing Ports, RAM, Display, Constants and Special Function Registers (SFR's). A "DO FOREVER" loop (the main program loop) is executed. The background task is executed and MODE Ø (Table 2.7 page 2-29) is displayed. The background task continues to execute until an external interrupt is received by INT Ø (Refer to APPENDIX B and C).

The following sequence of events occurs after an interrupt request is received by INT 0:

i) The current program status is saved.

ii) The microcontroller vectors to INT 0 interrupt procedure.

iii) The interrupt procedure is executed.

iv) Normal program execution continues on termination of the interrupt procedure.

The real-time analysis for each of the five tones, by INT 0 interrupt procedure, includes:

i) Software to ensure that a data change signal has been flagged.

ii) Re-initialization of Timer 1 with a 50ms gating period to measure the new tone frequency.

iii) Disabling Timer 0.

iv) Re-initialization of Timer 0 to enable correct measurement of the tone duration on the next interrupt.

v) Saving tone duration magnitudes.

vi) Starting tone duration Timer 0.

vii) Starting gating period for particular frequency in the sequence (Timer 1).

viii) Running PCA counter to count zero crossings.

ix) Checking for a repeat tone.

x) Saving the tones hexadecimal character.

When the 50ms gating period for each tone frequency elapses, the frequency count for that tone is stored. An interrupt may be serviced at any time during the background task. This task is not dependent on time as compared to the real-time INT 0, Timer 0 and Timer 1. The interrupts are serviced in the following order of priority:

i) INT 0.
ii) Timer 0.
iii) INT 1.
iv) Timer 1.
v) PCA.
vi) Serial Port.
vii) Timer 2.

On completion of the real-time analysis for the complete 5-Tone ZVEI sequence, the background task is flagged and all external interrupts are masked. The software computes values for the frequency error and tone duration error and converts the results to ASCII, ready for displaying.

2.3.2.1 Keypad

A data available output from the keyboard encoder triggers INT 1 of the microcontroller when any key is activated on the keypad. The interrupt procedure, INTRPT 1, causes the microcontroller to input one byte of information. The function of each key is shown in Table 2.6 on page 2-28.

2.3.2.2 Display

The selection of procedures controlling various modes of display depends on the hex value stored in the keyboard buffer. Refreshing and updating of the display is executed by the selected modes' procedure.

Once all parameters have been viewed, the display is cleared and returns to MODE 0. On returning to MODE 0, all variables are re-initialized and the 5-Tone ZVEI Encoder Analyser is ready to analyse further selcalls (Refer to APPENDIX B, C and D).

2.4 Operating Instructions

2.4.1 Precautions

i) Handle the unit with care. Malfunctioning would occur if the unit is dropped.

ii) Ensure that the Communications System Analyser has been properly set-up (Section 2.2.2 on page 2-18) to prevent damage to the input stage of the 5-Tone ZVEI Encoder Analyser.

iii) Ensure that the unit is located in a safe position at the QA Workstation (Section 2.2.1 on page 2-16).

iv) Do not use excessive force when pressing the keys on the keypad.

2.4.2 5-Tone ZVEI Encoder Analyser

2.4.2.1 Powering Up

Power is applied to the 5-Tone ZVEI Encoder Analyser via the illuminated rocker switch located on the rear panel of the unit.

2.4.2.2 Keypad Functions

The function of each key is shown in Table 2.6 below.

Key	Function
1 to 5	Numbers 1 to 5
A	Quit / Abandon Readings
C	Next / Previous Options Menu
D	Display MODE Options

Table 2.6 5-Tone ZVEI Encoder Analyser Keypad Functions

2.4.2.3 Display Format

The measured / calculated parameters are displayed as a screen of information. Each screen represents a different mode of operation. There are 9 different screens of information, i.e. MODE 0 to MODE 8. Table 2.7 on page 2-29 represents the various modes of operation. APPENDIX H represents the typical display formats for MODE 0 to MODE 8.

MODE	Message / Data Displayed
0	Ready for Analysis
1	Ready to View Results
2	1st Options Menu
3	2nd Options Menu
4	Decoded Digits
5	Tone Frequencies
6	Tone Frequency Errors
7	Tone Durations
8	Tone Duration Errors

Table 2.7 5-Tone ZVEI Encoder Analyser Display Modes

2.4.2.4 Display MODE Operation

On power-up, the 5-Tone ZVEI Encoder Analyser displays the MODE 0 message. After the radio's transmitter has been keyed and all 5 tones have been received, the display will automatically change to MODE 1.

All the parameters analysed by the real-time interrupt are now ready to be displayed. Selecting the D-key on the keypad results in the display changing to MODE 2. MODE 2 represents the first of two option menus. In this mode the display modes for MODE 4 to MODE 6 are selectable. By selecting the C-key on the keypad, the second option menu, MODE 3, will be displayed. Display modes 7, 8 and 0 are thus selectable.

MODE 2 to MODE 8 may be selected any number of times without losing the analysed data. If the A on the keypad is pressed while in MODE 3, the memory variables storing the analysed data will be re-initialized and the display automatically reverts to MODE Ø. The 5-Tone ZVEI Encoder Analyser will thus be ready to analyse a further selcall.

The descriptions of the various modes follow:

i) Display MODE Ø.

MODE Ø displays a message indicating that the unit is ready to analyse a 5-Tone ZVEI selcall.

ii) Display MODE 1.

This mode displays a message to the operator indicating that a selcall has been analysed and its parameters are ready for displaying.

iii) Display MODE 2 and MODE 3.

Display MODE 2 and MODE 3 are two similar option menus, each enabling controlled access to further modes of display.

iv) Display MODE 4.

This mode displays the decoded frequency digits for each of the 5 tones.

v) Display MODE 5.

Selecting MODE 5 enables the frequency for each of the 5-tones, measured during the real-time interrupt, to be displayed.

vi) Display MODE 6.

Displays the frequency error for each of the 5-tones. The error is calculated by the software as a percentage of the frequency measured by the real-time interrupt over the standard frequency for the particular tone digit.

vii) Display MODE 7.

Displays the tone duration for each of the 5-tones as measured by the real-time interrupt.

viii) Display MODE 8.

Displays the tone duration error for each of the 5-tones. The error is calculated as a percentage of the tone duration over the standard tone duration (70ms for each tone).

2.5 Maintenance

2.5.1 Calibration

The only routine maintenance required on the 5-Tone ZVEI Encoder Analyser is the verification of the 23,333kHz output frequency of the FX102LG. TP1 (APPENDIX A page 7-1) represents the test point to measure this frequency. C7, the trimming capacitor, is adjusted to preset the frequency.

The calibration procedure for the 23,333kHz output frequency is as follows:

i) Switch the 5-Tone ZVEI Encoder Analyser off.

ii) The 5-Tone ZVEI Encoder Analyser housing is made up of a top and a bottom half. The keypad and display are both attached to the top half and both connect, via ribbon cable, onto separate plugs of the P.C.B, mounted on the bottom half of the housing.

iii) Open the 5-Tone ZVEI Encoder Analyser by carefully lifting the top half, which clips off, and unplug the connectors for the keypad and display.

iv) TP1 and the tuning capacitor, C7, are both accessible through two small holes in the radio frequency shield.

v) Connect a frequency counter to TP1. Switch on the 5-Tone ZVEI Encoder Analyser and adjust C7 with a suitable tuning tool until a frequency of 23,333kHz is obtained on the frequency counter.

vi) Disconnect the frequency counter and switch off the 5-Tone ZVEI Encoder Analyser.

vii) Plug the connectors for the keypad and display into their respective sockets and fit the top cover of the housing into place. Press firmly to secure the retaining clips.

viii) Thereafter follow the Operating Instructions as discussed in Section 2.4 (page 2-27).

2.6 5-Tone ZVEI Encoder Analyser Specifications

System Components	:	5-Tone ZVEI Encoder Analyser.
Supply Voltage	:	220V~ to 250V~.
Operating Temperature	:	0 to 55 Degrees Celsius. 32 to 131 Degrees Fahrenheit.
Relative Humidity	:	To 80 percent without condensation.

Dimensions	:	5-Tone ZVEI Encoder Analyser
		length 188mm.
		width 110mm.
		max. height 90mm.
		min. height 46mm.
Mass	:	482 grams.
Worst Case Frequency Error	:	±2,2 % for 1060Hz to 2600Hz.
Tone Duration Accuracy	:	±3ms, 4 % for 70ms tones and 1060Hz to 2600Hz.
Software features provides real-time Analysis of	:	i) Decoded 5-Tone ZVEI frequency digits.
		ii) Frequency for each of the 5 ZVEI tones.
		iii) Tone Duration for each of the 5 ZVEI tones.
Computed Parameters	:	i) Frequency error for each of the 5 ZVEI tones.
		ii) Tone Duration Error for each of the 5 ZVEI tones.

3. RESULTS

3.1 Functional Tests Using HP Multifunction Synthesizer

3.1.1 Table of Test Results

Table 3.1 represents a set of practical results for a 5-tone ZVEI sequence received from an HP Multifunction Synthesizer as analysed by the 5-Tone ZVEI Encoder Analyser.

Tone Sequence Analysed :	1 1060Hz		2 1160Hz		3 1270Hz		4 1400Hz		5 1530Hz	
Test No.	Frequency Error (%)					Tone Duration Error (%)				
1	-0,4	+0,4	0,0	+0,7	-0,9	+3	-4	+4	-4	-3
2	0,0	0,0	0,0	+0,7	-0,3	+3	-4	+4	-4	-1
3	0,0	+0,4	+0,3	+0,3	-0,3	+4	-4	+4	-4	-1
4	0,0	+0,8	0,0	+0,7	0,0	+4	-4	+4	-4	-1
5	0,0	+0,4	+0,3	+0,3	0,0	+4	-3	+3	-4	-1
6	0,0	+0,4	+0,3	+1,0	-2,2	+1	-3	+4	-4	-3
7	0,0	+0,4	+0,3	+0,3	0,0	+3	-1	+3	-4	-1
8	0,0	0,0	0,0	+0,7	-2,0	+1	-1	+4	-4	-3
9	0,0	+0,4	+0,3	+0,7	-0,3	+4	-4	+4	-4	-1
10	0,0	0,0	0,0	+0,7	-2,2	+1	-4	+4	-4	-1
11	-0,4	0,0	+0,3	+1,0	-2,0	+3	-4	+4	-4	-4
12	0,0	0,0	+0,3	+0,7	0,0	+4	-4	+4	-4	-1
13	0,0	0,0	0,0	+1,0	-0,7	+3	-4	+4	-4	-1

Table 3.1 Test results of Multifunction Synthesizer as analysed by the 5-Tone ZVEI Encoder Analyser. Modes 4, 6 and 8 shown.

3.1.2 Interpretation of Test Results

Table 3.1 on page 3-1 represents a portion of the many functional tests performed on the 5-Tone ZVEI Encoder Analyser. For each of the tests, the following results were obtained:

i) The 5-Tone ZVEI Encoder Analyser performed successful tone decoding for each of the tests conducted.

ii) The frequency and hence frequency error for tones 1 to 4 were very accurately measured / computed when compared to the frequency specifications. Tone 5 yielded a most interesting result. The frequency deviation from the standard frequency and, hence, frequency error was, sometimes, slightly greater when compared to the other tones.

iii) The tone duration error for each of the 5-tones was found to be within the specifications.

iv) The tone duration error and frequency error computation algorithm proved to function accurately for each of the many tests conducted on the 5-Tone ZVEI Encoder Analyser.

3.2 Mobile Radio ZVEI Test Results

The results obtained by the 5-Tone ZVEI Encoder Analyser, having analysed the 5-Tone ZVEI signals from a Midland radio Model 70-1440 and a Motorola SYNTRX radio, are shown in Table 3.2 and Table 3.3 respectively. The results are discussed on pages 3-4 and 3-5.

Tone Sequence Analysed :	0 2400Hz		3 1270Hz		1 1060Hz	5 1530Hz		8 2000Hz	
Test No.	Frequency Error (%)					Tone Duration Error (%)			
1	-0,8	0,0	-0,5	-0,9	0,0	-5	+4	+1	0	+3
2	-0,2	0,0	0,0	+0,3	-0,3	-4	0	+3	0	+3
3	0,0	0,0	0,0	0,0	-0,3	+3	+1	+1	+1	+5
4	-0,4	0,0	-0,5	0,0	-0,3	-4	+1	0	+1	+3
5	0,0	0,0	0,0	+0,3	0,0	-4	+1	+1	+1	0
6	0,0	0,0	0,0	-0,3	0,0	-6	+1	+3	0	+4
7	-1,6	+0,4	0,0	+0,3	-0,3	-6	+1	+1	+1	+1
8	-0,6	0,0	0,0	-0,9	0,0	-7	+3	+1	+1	+1
9	-1,6	+0,4	0,0	+0,6	-0,3	-7	+1	+1	+1	0
10	+0,2	0,0	0,0	0,0	0,0	-6	+3	0	+3	0
11	-1,6	0,0	0,0	0,0	0,0	-7	+3	+3	+1	+1

Table 3.2 Analysed results of a MIDLAND radio, Model 70-1440. Modes 4, 6 and 8 shown.

Tone Sequence Analysed :	1 1060Hz		2 1160Hz		1 1060Hz	4 1400Hz		4 2600Hz	
Test No.	Frequency Error (%)					Tone Duration Error (%)			
1	+0,9	-0,4	0,0	+1,0	-0,5	-4	-3	+1	-3	+3
2	0,0	-0,4	-0,5	+0,4	-0,7	0	-4	+1	-3	+1
3	0,0	-0,4	0,0	+1,4	-1,1	0	-6	+1	-3	+1
4	0,0	-0,4	0,0	+2,8	0,0	+1	-3	+1	-6	+3
5	0,0	-0,4	+0,5	+0,4	+0,2	-1	-1	0	-3	+1
6	-0,5	0,0	-0,5	+2,8	-0,2	+1	-3	+1	-4	+1
7	0,0	-0,4	-0,5	+1,7	0,0	-1	-4	+1	-3	+1
8	-0,5	-0,4	-0,5	+0,7	0,0	0	-4	+1	-4	+1
9	0,0	0,0	0,0	+1,7	-0,7	-4	-4	+4	-4	+1
10	0,0	-0,4	0,0	+3,2	0,0	0	-4	+3	-6	+1
11	0,0	-0,4	-0,5	+2,5	-1,5	0	-1	+1	-3	+1

Table 3.3 Analysed results of a Motorola SYNTRX radio. Modes 4, 6 and 8 shown.

An explanation of the results obtained is as follows:

i) Eleven tests were performed on each of the radios.

ii) The frequency digits of the selcalls received from each radio were successfully decoded for each of the eleven tests performed.

iii) The selcall frequency / frequency-error, for each of the eleven tests performed on the Midland radio conformed to the manufactures' specifications.

iv) The first tone-duration / duration-error for each of the tests performed on the Midland radio was found to be, generally, greater than that of the Motorola radio. This phenomenon is due to a larger variation in the "no-tone" period (refer to 2.1.2, page 2-10) prior to tone 1 being received.

v) The frequency / frequency-error for each of the eleven tests performed on the Motorola radio yielded a most interesting result. The frequency errors for the forth tone was generally greater than the frequency errors for the other tones in the particular selcall sequence. Although the frequency / frequency-error for the fourth tone was generally greater, all the tone frequencies were within the frequency specifications.

vi) The tone-duration / tone-duration-error for each of the eleven tests performed on the Motorola radio conformed to the manufactures' specifications.

The results verify that differences between analysed parameters not only exists between various makes of radios, but also within the 5-Tone sequence of the selcall for the particular radio.

The software algorithms, used to obtain these effective practical results, shall be used to assist during the design of software to upgrade the Front End Processor (FEP) of the Cleansing and Traffic Departments' Radio Dispatch Systems.

4. IMPROVEMENTS ON THE 5-TONE ZVEI ENCODER ANALYSER

The 5-Tone ZVEI Encoder Analyser software could be enhanced to enable the unit to operate with selcalls received "off the air".

The software of the 5-Tone ZVEI Encoder Analyser could be re-structured to enable uploading of selcall results to a Radio Database operating on an IBM XT/AT Personal Computer. These results could then be used for further analysis and also for statistical purposes.

The program could be re-written in Franklin C, whereby a floating point library, unavailable with PLM-51, could be used for all arithmetic computations.

5. CONCLUSION

i) A 5-Tone ZVEI Encoder Analyser was successfully developed.

ii) The 5-Tone ZVEI Encoder Analyser can be interfaced to the existing Communications System Analyser and the high cost of a new Communications System Analyser facilitating ZVEI analysis was avoided.

iii) The 5-Tone ZVEI Encoder Analyser fulfils the requirements for analysing the encoded ZVEI signal.

* A display of the 5-Tone ZVEI frequency digits, tone frequencies and tone durations is obtained.
* The maximum frequency error and tone duration error obtained was +3,2% and -7% respectively.

iv) The PLM-51 Compiler was successfully applied as a development tool.

v) The 87C51FA microcontroller was found to be most suitable.

BIBLIOGRAPHY / LIST OF REFERENCES

Articles in Magazines

Ciarcia, S. 1988. **Why Microcontrollers Part 1 & 2.** Byte Magazine. August & September. Pages 239 - 247, 303 - 312.

Manuals

Hewlett-Packard. 1987. **HP Multifunction Synthesizer Basic Operation and Application.** Hewlett-Packard Company, July, Pages 3-1 to 3-19.

Intel. 1982. **PL/M-51 User's Guide.** Intel Corporation. Pages 1-1 to 16-5.

Optrex Corporation. 1989. **DMC Series Liquid Crystal Displays.** Optrex Corporation. Pages 43 - 64.

Data Books

Intel. 1990. **MCS-51 Embedded Applications.** Intel Corporation. Pages 2-1 to 2-76.

National Semiconductor. 1984. **Logic data book. Volume 1 and 2.** California. National Semiconductor Corp.

Product Information (ANNEXURE 1 and 2)

Selective-call Tone Encoder / Decoder information reproduced courtesy of UNIPLAN, official agents for Consumer Microcircuits Limited.

Display information reproduced courtesy of H.Kopp Electronics, official agents for Optrex Corporation.

References Quoted

1. British Standard. 1989. **Radio Equipment used in mobile services.** British Standard. Parts 1 & 6.

2. Consumer Microcircuits. 1990. **Product Information for Tone Encoders / Decoders.** Consumer Microcircuits Limited.

3. Intel. 1990. **Intel 8-Bit Embedded Controllers.** Intel Corporation. Pages 8-40 to 8-74.

4. Motorola. Inc. 1982. **Motorola Communications System Analyser Service Manual (R-2001C / R-2002C).** Motorola Incorporated. Pages 5-13 to 5-14.

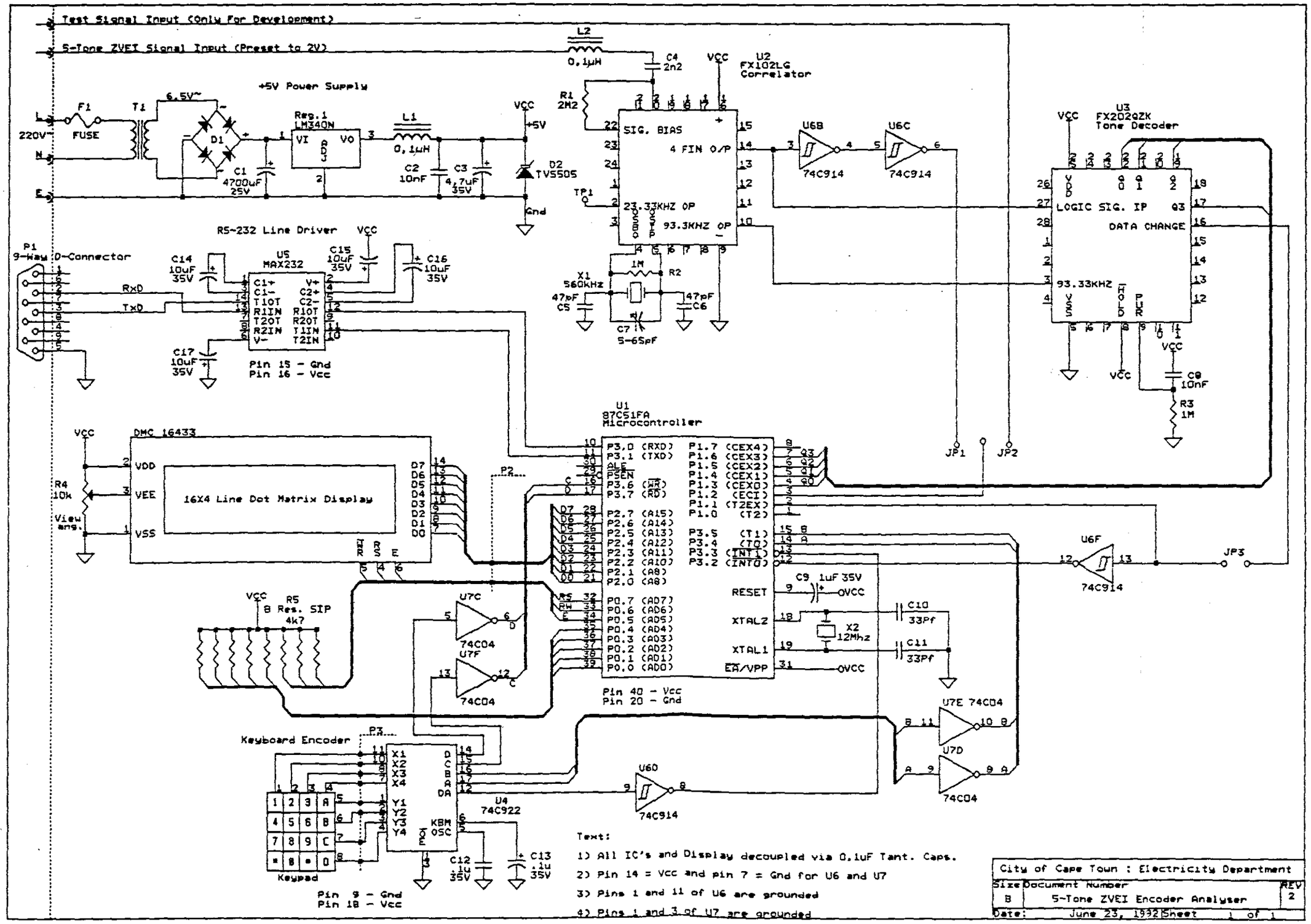
5. Sharpe, T. **Signalling Systems Today and Tomorrow.**

6. South African Standard. 1989. **S.A.B.S Specifications for Communication Signalling for use in trunked band mobile radio systems.** South African Standards.

A P P E N D I X A

5 - T O N E Z V E I E N C O D E R A N A L Y S E R

C I R C U I T D I A G R A M


Test Signal Input (Only For Development)
5-Tone ZVEI Signal Input (Preset to 2V)
+5V Power Supply
220V~
FUSE
T1
6.5V~
D1
Reg.1 LM340N
VI
ADJ
VO
+5V
L1
0,1µH
C1 4700uF 25V
C2 10nF
C3 4,7uF 35V
D2 TVS505
Gnd
L2
0,1µH
C4 2n2
VCC
U2 FX102LG Correlator
R1 2M2
SIG. BIAS
4 FIN O/P
23.33KHZ OP
93.3KHZ OP
TP1
X1 560KHz
47pF C5
47pF C6
C7 5-65pF
1M R2
U6B
U6C
74C914
74C914
U3 FX202QZK Tone Decoder
LOGIC SIG. IP
DATA CHANGE
93.33KHZ
Q0
Q1
Q2
Q3
VDD
VSS
HOLD
PWR
C8 10nF
R3 1M
JP1
JP2
RS-232 Line Driver
P1 9-Way D-Connector
RxD
TxD
U5 MAX232
C14 10uF 35V
C15 10uF 35V
C16 10uF 35V
C17 10uF 35V
Pin 15 - Gnd
Pin 16 - Vcc
U1 87C51FA Microcontroller
P3.0 (RXD)
P3.1 (TXD)
ALE
PSEN
P3.6 (WR)
P3.7 (RD)
P2.7 (A15)
P2.6 (A14)
P2.5 (A13)
P2.4 (A12)
P2.3 (A11)
P2.2 (A10)
P2.1 (A8)
P2.0 (A8)
P0.7 (AD7)
P0.6 (AD6)
P0.5 (AD5)
P0.4 (AD4)
P0.3 (AD3)
P0.2 (AD2)
P0.1 (AD1)
P0.0 (AD0)
P1.7 (CEX4)
P1.6 (CEX3)
P1.5 (CEX2)
P1.4 (CEX1)
P1.3 (CEX0)
P1.2 (ECI)
P1.1 (T2EX)
P1.0 (T2)
P3.5 (T1)
P3.4 (T0)
P3.3 (INT1)
P3.2 (INT0)
RESET
XTAL2
XTAL1
EA/VPP
C9 1uF 35V
C10 33Pf
C11 33Pf
X2 12Mhz
Pin 40 - Vcc
Pin 20 - Gnd
VCC
DMC 16433
16X4 Line Dot Matrix Display
VDD
VEE
VSS
R4 10k
View ang.
D7
D6
D5
D4
D3
D2
D1
D0
REG
R/W
E
P2
U7C 74C04
U7F 74C04
R5 8 Res. SIP 4k7
U7E 74C04
U7D
74C04
U6F
74C914
JP3
Keyboard Encoder
P3
X1
X2
X3
X4
Y1
Y2
Y3
Y4
D
C
B
A
DA
KBM OSC
EOC
U4 74C922
C12 .1u 35V
C13 .1u 35V
Keypad
Pin 9 - Gnd
Pin 18 - Vcc
U6D
74C914
Text:
1) All IC's and Display decoupled via 0,1uF Tant. Caps.
2) Pin 14 = Vcc and pin 7 = Gnd for U6 and U7
3) Pins 1 and 11 of U6 are grounded
4) Pins 1 and 3 of U7 are grounded
City of Cape Town : Electricity Department
Size B
Document Number
5-Tone ZVEI Encoder Analyser
REV 2
Date: June 23, 1992
Sheet 1 of 1

APPENDIX B

5-TONE ZVEI ENCODER ANALYSER SOFTWARE DESCRIPTION

APPENDIX B

8. 5-TONE ZVEI ENCODER ANALYSER SOFTWARE DESCRIPTION

This section contains a detailed description for each of the software procedures of the 5-Tone ZVEI Encoder Analyser.

8.1 Line Numbers : 577 to 785

Module Name : **ZVEI**

Procedure : Hardware reset (cold-start).

Description : Main DO FOREVER loop. RAM, I/O, display and SFR's are first initialized and then the DO FOREVER loop is entered. This loop consists of a background task which executes to display MODE Ø. The background task is interrupted by the real-time interrupts, INTØ, timer Ø & timer 1 which analyse the incoming selcall. Once the complete selcall has been analysed, the tone duration errors and frequency errors are computed. At this point in the program, all the results are ready for displaying.

The analysed and computed results are displayed as various modes of display. These modes of display include modes for decoded Frequency Digits, Tone Frequency, Tone Frequency Error, Tone Duration and Tone Duration Error. Once all the results have been viewed, the 5-Tone ZVEI Encoder Analyser returns to MODE Ø.

8.2 Line Numbers : 24 to 136

Module Name : **INTRPT0**

Procedure : External interrupt 0, performs real-time analysis of selcalls.

Description : This interrupt is the core of the 5-Tone ZVEI Encoder Analyser. It is activated by the data change signal from the FX202QZK tone decoder. The interrupt procedure ensures that a valid data change signal has been received and then processes the parameters for the particular tone in the 5 tone sequence. The gating window for the frequency counter is initialized for each interrupt and the PCA counter is cleared to ensure correct counts.

On the first activation of the interrupt (tone 1), timer 0 is initialized and enabled. The frequency gating window, timer 1, and the PCA counter are enabled. The tone frequency digit is read at the port pins and stored. Interrupts generated by tones 2 to 5 have similar algorithms. The instructions include stopping and re-initializing of the tone duration timer, saving the tone duration of the particular tone, clearing the variable storing the

tone duration, starting the tone duration timer for the next tone and enabling the PCA counter. The tone frequency digit is read at the port pins and stored. If the frequency digit is the same as the previous, a repeat flag is set. On the 6th and final interrupt, the duration timer is stopped, its value is stored and the variable is cleared.

Once all 5 tones have been analysed, a flag is set to enable the background task to prepare for the displaying of the results.

8.3 Line Numbers : 148 to 152

Module Name : **TIMER Ø**

Procedure : Real-time tone duration timer interrupt.

Description : Timer Ø is enabled by the main real-time interrupt, external interrupt Ø. Its function is to set up a 1 ms interrupt and increment a duration timer variable every milli-second. This timer is also disabled and cleared by INTRPTØ.

8.4 Line Numbers : 153 to 182

Module Name : **TIMER 1**

Procedure : Real-time frequency counts for each of the five frequency digits.

Description : The gating window and PCA counter (CR) are enabled in the main real-time interrupt, external interrupt 0. Once the gating period (50 ms) elapses, an interrupt is generated and the timer 1 interrupt routine is executed. The timer and counter are both disabled and the frequency count for the particular tone in the 5 tone sequence is stored for further processing by the background task.

8.5 Line Numbers : 137 to 147

Module Name : **INTRPT1**

Procedure : Keyboard interrupt, stores key pressed on the keypad.

Description : This interrupt is enabled only when the analysed results are ready to be displayed. By pressing a key on the keypad, a data available signal from the keyboard encoder triggers external interrupt 1 of the microcontroller. The port for the keyboard bits is read and stored into a keypad buffer. This keypad buffer is read during execution of the main program.

8.6 Line Numbers : 183 to 189

Module Name : **DISPLAY$SETUP**

Procedure : Power-up initialization of display (also executed by main program).

Description : The display is initialized according to the initialization instruction passed via INST. A 10 ms delay is incorporated as a dummy read of the display busy flag. The display is enabled and the initializing instruction is then written to the display. Having completed these operations, the display is disabled.

8.7 Line Numbers : 190 to 196

Module Name : DISPLAY$ADDR

Procedure : Locates address where character is to be written on the display.

Description : The instruction to enable the display and the address where data is to be written on the display is passed to this procedure. A 10 ms delay is incorporated as a dummy read of the display busy flag. The display is enabled and the DD RAM address is initialized. The display is then disabled.

8.8 Line Numbers : 197 to 206

Module Name : **DISPLAY$CHAR**

Procedure : Displays a character at a previously initialized display address.

Description : The instruction to enable the display and the character to display are passed to this procedure. The display is enabled and the character is written to the DD RAM. 10 ms delays have been incorporated as a dummy read of the display busy flag. After writing the data to the display, the display is disabled.

8.9 Line Numbers : 207 to 238

Module Name : CALC$TDUR$ERR

Procedure : Calculates the Tone Duration Error for each of the 5 tones.

Description : The tone duration is passed to this procedure and the resultant tone duration error is calculated. This procedure is executed for each of the 5 tones. The procedure is called by the main program immediately after real-time analysis of the selcall has been completed. The resultant error is rounded off and may be zero, positive or negative. The error for each of the 5 tones is shown as a percentage deviation from the standard tone duration.

8.10 Line Numbers : 247 to 253

Module Name : **SELECTDKEY$MSG**

Procedure : Initiates the displaying of MODE 1.

Description : This procedure clears the display and calls the procedure, D$KEY which performs the actual writing of characters to the display at different start address locations.

8.11 Line Numbers : 239 to 246

Module Name : **D$KEY**

Procedure : Writes display MODE 1 to the display.

Description : The display address and arrays final element, of the characters to display, are passed to the procedure. The array start address is initialized in the SELECTDKEY $MSG procedure. A do while loop writes MODE 1 to the display at the address set up by the DISPLAY$ADDR procedure until the array pointer reaches the final element of the characters to display for MODE 1.

8.12 Line Numbers : 262 to 269

Module Name : **OPTION$DISPLAY$1**

Procedure : Initiates the displaying of MODE 2.

Description : The display is cleared and the start element of the array for display MODE 2 is initialized. The address and arrays final element, of the characters to display, are passed to OPTIOIN$DONATION$1. This procedure performs the actual writing of the characters for MODE 2 to the display.

8.13 Line Numbers : 254 to 261

Module Name : **OPTION$DONATION$1**

Procedure : Writes display MODE 2 to the display.

Description : The display address and arrays final element, of the characters to display, are passed to the procedure. The display address is initialized by calling DISPLAY$ADDR. Once the address is located on the display, a do while loop writes MODE 2 to the display until the array pointer reaches the final element of the characters to display for MODE 2.

8.14 Line Numbers : 278 to 298

Module Name : **DECODED$TONE**

Procedure : Displays the frequency digits for each of the 5 tones.

Description : This procedure clears the display, writes the heading for MODE 4 and then proceeds to write the 'A-Quit' message to enable this particular mode to be exited. The display address is initialized and the frequency digits are written to the display at the relevant addresses.

8.15 Line Numbers : 270 to 277

Module Name : **QUIT$MSG**

Procedure : Displays 'A-Quit' message for the various modes of display.

Description : The display address is initialized. The position of the first character, of the array to display, is initialized. A do while loop writes the characters to the display, starting at the initialized address. This continues until the array pointer reaches the final element of the characters to display.

8.16 Line Numbers : 307 to 321

Module Name : **FREQ$HEADING**

Procedure : Writes MODE 5 heading, 'Tone Freq.(Hz)' and initiates the displaying of the frequency numbers for MODE 5.

Description : The display is cleared and the address for the heading of MODE 5 is initialized. A do while loop performs the writing of the heading for MODE 5. The address, together with the final element of the characters to display, is passed to FREQ$DONATION. This procedure performs the actual writing of the frequency numbers to the display.

8.17 Line Numbers : 299 to 306

Module Name : **FREQ$DONATION**

Procedure : This procedure writes the frequency numbers to the display.

Description : The address of the frequency number together with the final array element, to write to the display, is passed to this procedure. The initial array element is initialized in FREQ$HEADING. The address to write the characters is located on the display and a do while loop writes the frequency numbers for MODES 5 & 6.

8.18 Line Numbers : 322 to 334

Module Name : **FREQ$CALC**

Procedure : Calculates the frequency from the number of gated counts received during the real-time analysis of the 5-tone sequence.

Description : The gated counts held in the PCA counter are passed to this procedure. The counts are computed to produce a frequency. The resultant frequencies have to be divided by 4 as the FX102LG outputs a frequency 4x that of the input frequency. The frequencies are converted to ASCII and each frequency element, F$DIG1 to 4, is stored in a variable, ready for displaying.

8.19 Line Numbers : 335 to 342

Module Name : **DISPLAY$FREQ**

Procedure : Displays frequencies at the correct location on the display for MODE 5.

Description : The address where each frequency is to be displayed together with the ASCII elements comprising the frequency are passed to this procedure. DISPLAY$ADDR is called to initialize the display address for the particular frequency. DISPLAY$CHAR is called to write each ASCII frequency element, F$DIG1 to 4, to the display starting at the initialized address.

8.20 Line Numbers : 350 to 369

Module Name : **FREQERRHEADING**

Procedure : Writes MODE 6 heading, 'Frequency Error' and initiates the displaying of the frequency numbers for MODE 6.

Description : The display is cleared and the address of the heading for MODE 6 is initialized. A do while loop writes the heading for MODE 6, starting at the initialized address. The address, together with the final element of the characters to display is passed to FREQ$DONATION. In addition, the procedure writes the frequency numbers to the display at the initialized address.

8.21 Line Numbers : 370 to 415

Module Name : **CALC$FREQ$ERR**

Procedure : Calculates the frequency error for each of the tones in the 5-tone sequence.

Description : The gated frequency counts, measured during the RTI, are passed to this procedure together with the decoded frequency digit. The gated counts held in the PCA counter are computed to produce a frequency.

The standard frequency's magnitude is obtained by using the received frequency digit and then accessing the frequency from the standard frequency array. The computed frequency is compared to the corresponding array frequency. The calculated error may be zero, positive or negative and is calculated to one decimal place. The error for each of the 5 tones is shown as a percentage deviation from the standard ZVEI frequency.

8.22 Line Numbers : 416 to 424

Module Name : **SHOW$FREQ$ERR**

Procedure : Writes computed parameters for MODE 6 to the display.

Description : The address where the particular frequency error is to be written, the polarity of the frequency error and the magnitude of the frequency error is passed to this procedure. DISPLAY$ADDR is called to locate the address on the display and DISPLAY$CHAR is called to write the relevant data to the display at the initialized address. After the frequency error has been written, the procedure calls PERC$DISPLAY to display the units.

8.23 Line Numbers : 343 to 349

Module Name : **PERC$DISPLAY**

Procedure : Writes '%' character, used in MODES 6 & 8 to the display.

Description : The position of the '%' character within the array, MESSAGE$4, is passed to this procedure. A do while loop is executed which calls DISPLAY$CHAR to write this single character to the display.

8.24 Line Numbers : 433 to 440

Module Name : **OPTION$DISPLAY$2**

Procedure : Initiates the displaying of MODE 3.

Description : The display is cleared and the start element of the array for display MODE 3 is initialized. The start address and arrays final element, of the characters to display, are passed to OPTION$DONATION$2 which performs the actual writing of the characters for MODE 3 to the display.

8.25 *Line Numbers* : 425 to 432

Module Name : **OPTION$DONATION$2**

Procedure : Writes display MODE 3 to the display.

Description : The display address and arrays final element, of the characters to display, are passed to the procedure. The array start address is initialized by OPTION$DISPLAY$2. The display address is initialized by calling DISPLAY$ADDR. Once the address is located on the display, a do while loop writes MODE 3 to the display until the array pointer reaches the final element of the characters to display.

8.26 Line Numbers : 456 to 485

Module Name : **TONE$DURATIONS**

Procedure : Controls the displaying of the heading and analysed parameters for MODE 7.

Description : The display is cleared and TONE$DURAT$MSG is initialized with the display address and the arrays final element, of the characters to display. TONE$DURAT$MSG is called to write the heading for MODE 7. The tone durations, as analysed for each of the tones in the 5-tone sequence, are written to the display by passing their values to DISPLAY$CHAR.

Once the tone duration magnitudes have been displayed the units, ms, are written to the display. Finally the 'A-Quit' message is written to the display.

8.27 Line Numbers : 441 to 448

Module Name : **TONE$DURAT$MSG**

Procedure : Displays the heading for MODE 7.

Description : The display address and arrays final element, of the characters to display, are passed to the procedure. The arrays start address is initialized in TONE$DURATIONS. The display address is initialized and a do while loop writes the heading for MODE 7 to the display until the array pointer reaches the final element of the characters to display.

8.28 Line Numbers : 449 to 455

Module Name : **ms$DISPLAY**

Procedure : Displays the 'ms' units for display MODE 7.

Description : The position of the 'ms' characters within the array, MESSAGE$4, is passed to this procedure. A do while loop is entered which calls DISPLAY$CHAR to write these characters to the display.

8.29 Line Numbers : 486 to 530

Module Name : **TONE$DURATION$ERR**

Procedure : Writes display MODE 8 to the display.

Description : The display is cleared and the heading for MODE 8 is written to the display at the desired address. The displaying of the computed tone duration errors, the polarity of the errors and the units are all processed by this procedure. Once all the computed parameters have been written to the display, the 'A-Quit' message is displayed.

APPENDIX C

5-TONE ZVEI ENCODER ANALYSER

FLOW CHARTS

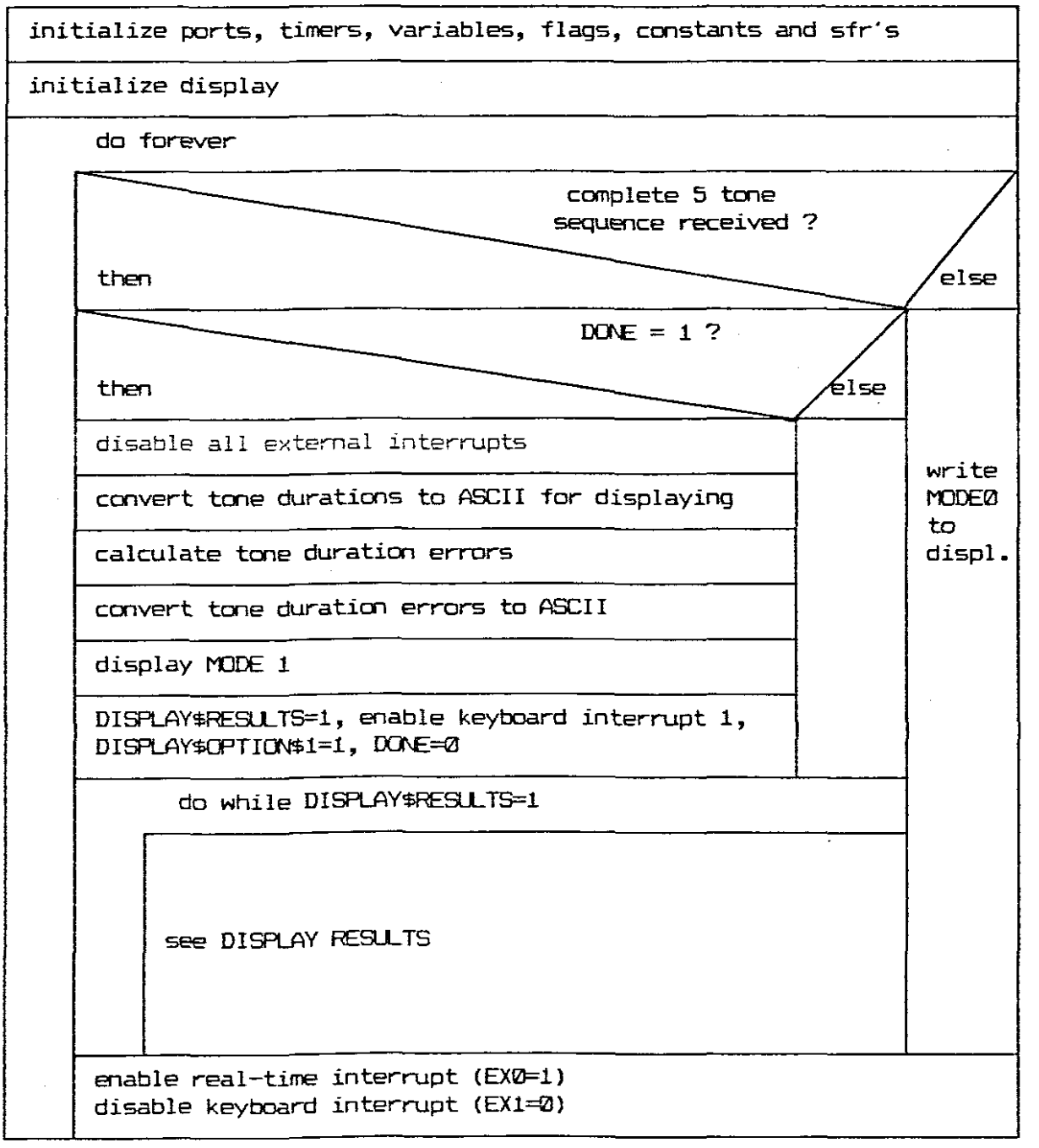
APPENDIX C

9. 5-TONE ZVEI ENCODER ANALYSER FLOW CHARTS

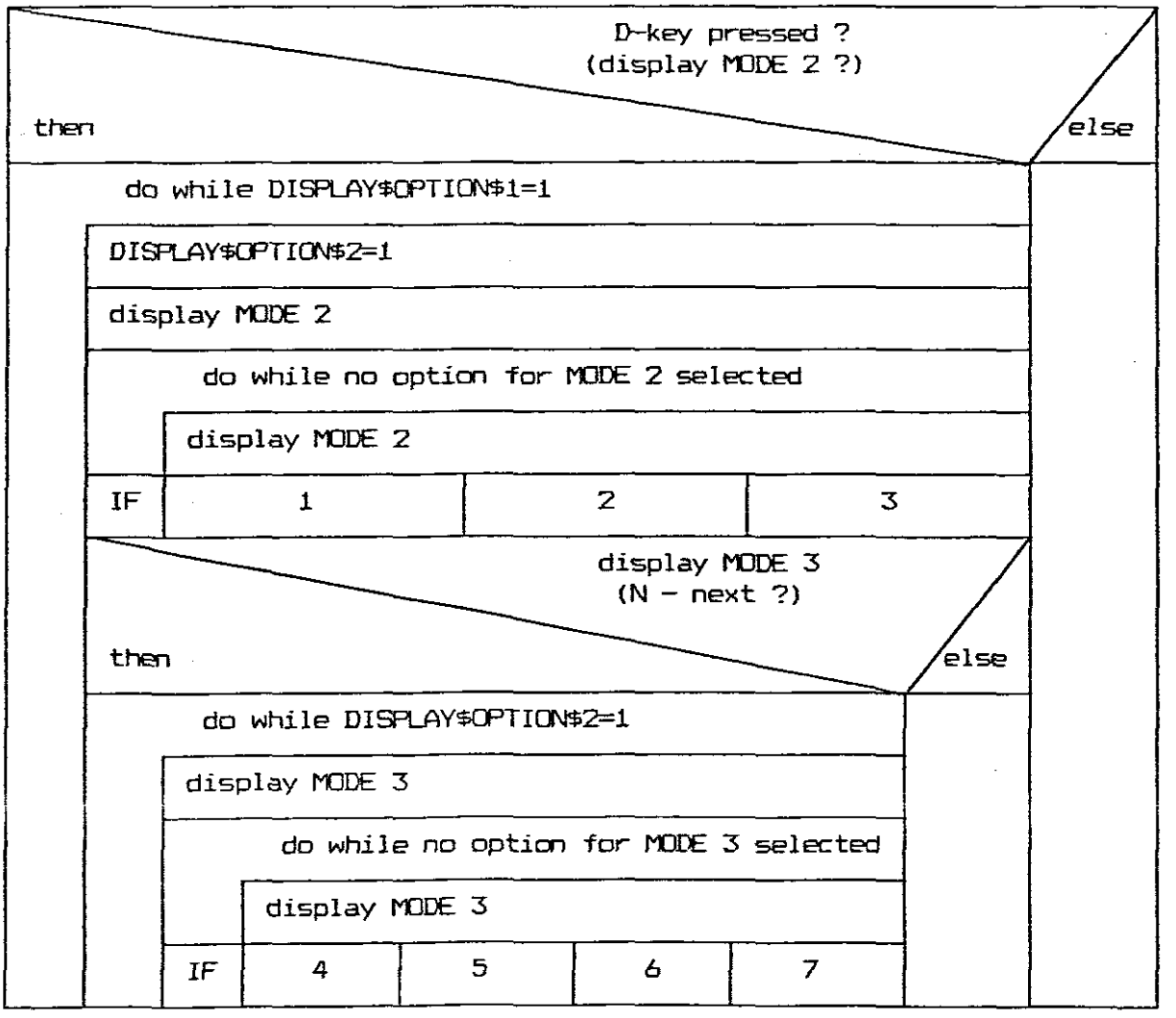
This section contains the flow charts for the software procedures of the 5-Tone ZVEI Encoder Analyser.

9.1 Main Program for the control of the 5-Tone ZVEI Encoder Analyser.

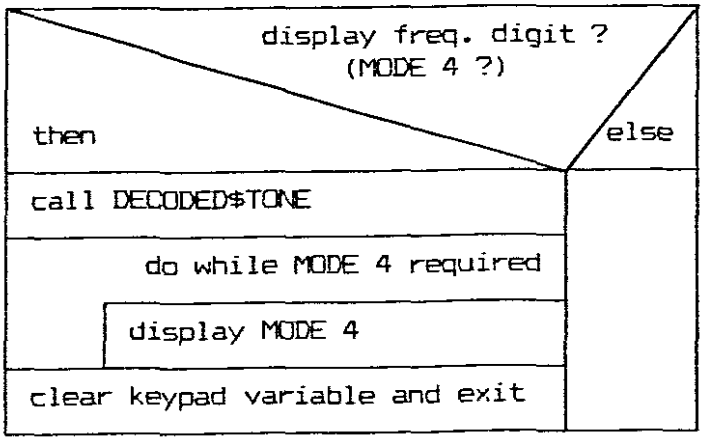
ZVEI



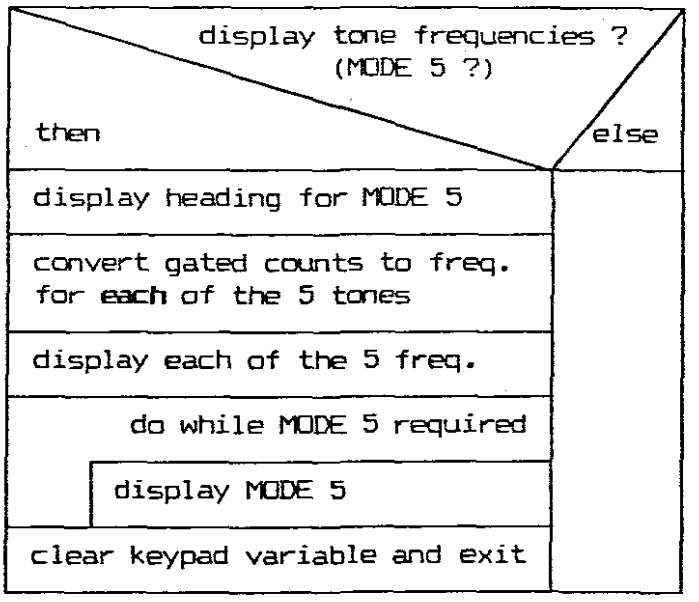
DISPLAY RESULTS



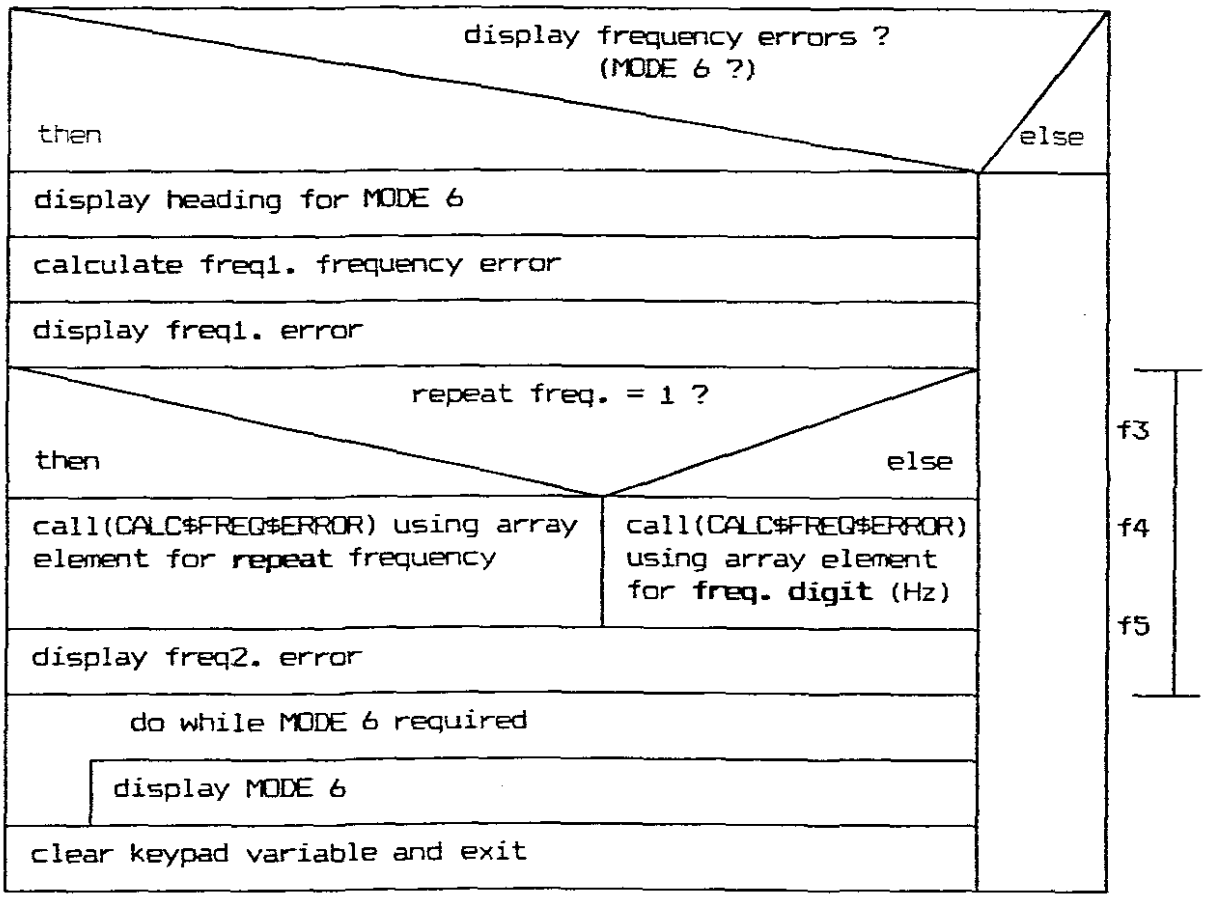
IF 1



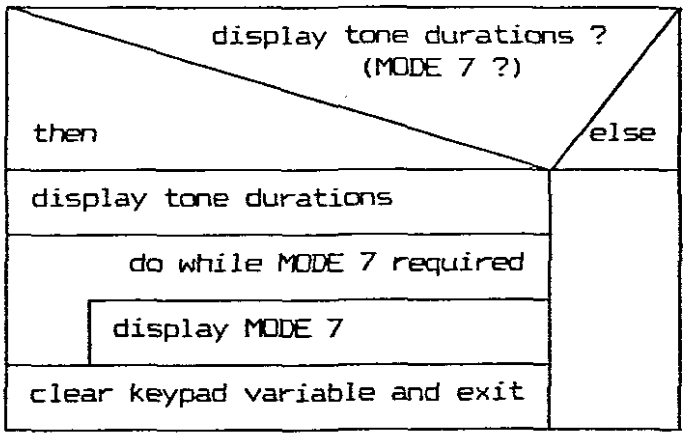
IF 2



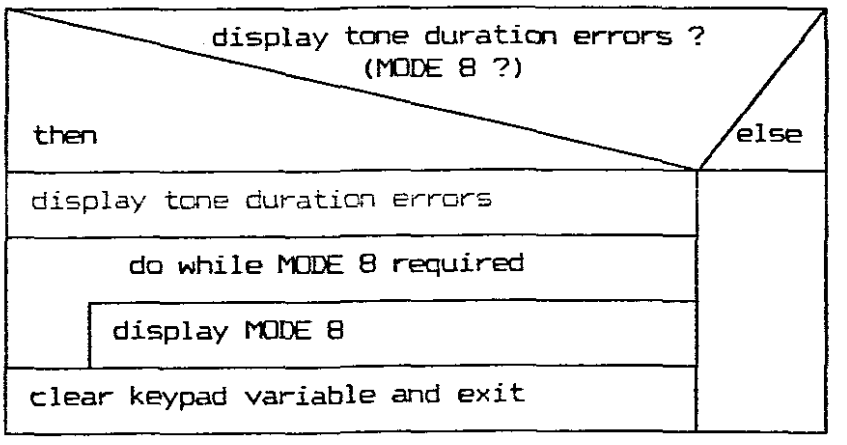
IF 3 (marking indicates repeat for f3 to f5)



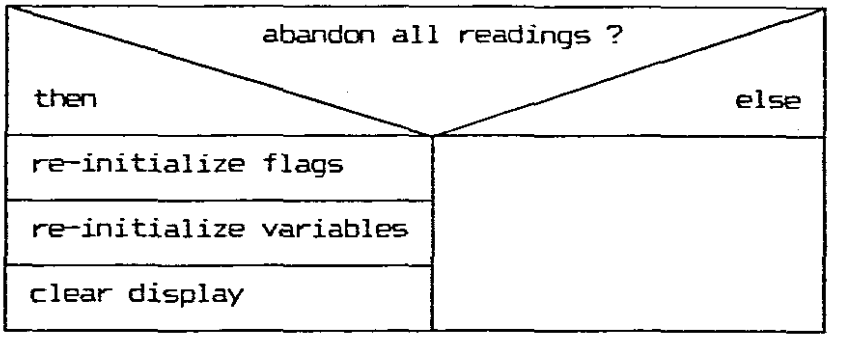
IF 4


display tone durations ?
(MODE 7 ?)
then
else
display tone durations
do while MODE 7 required
display MODE 7
clear keypad variable and exit

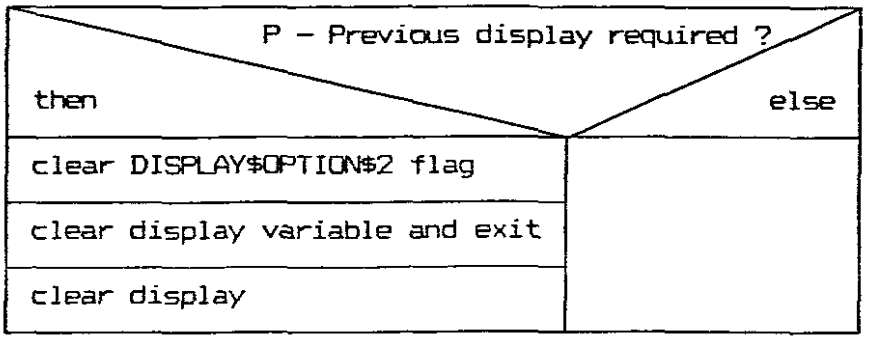
IF 5


display tone duration errors ?
(MODE 8 ?)
then
else
display tone duration errors
do while MODE 8 required
display MODE 8
clear keypad variable and exit

IF 6

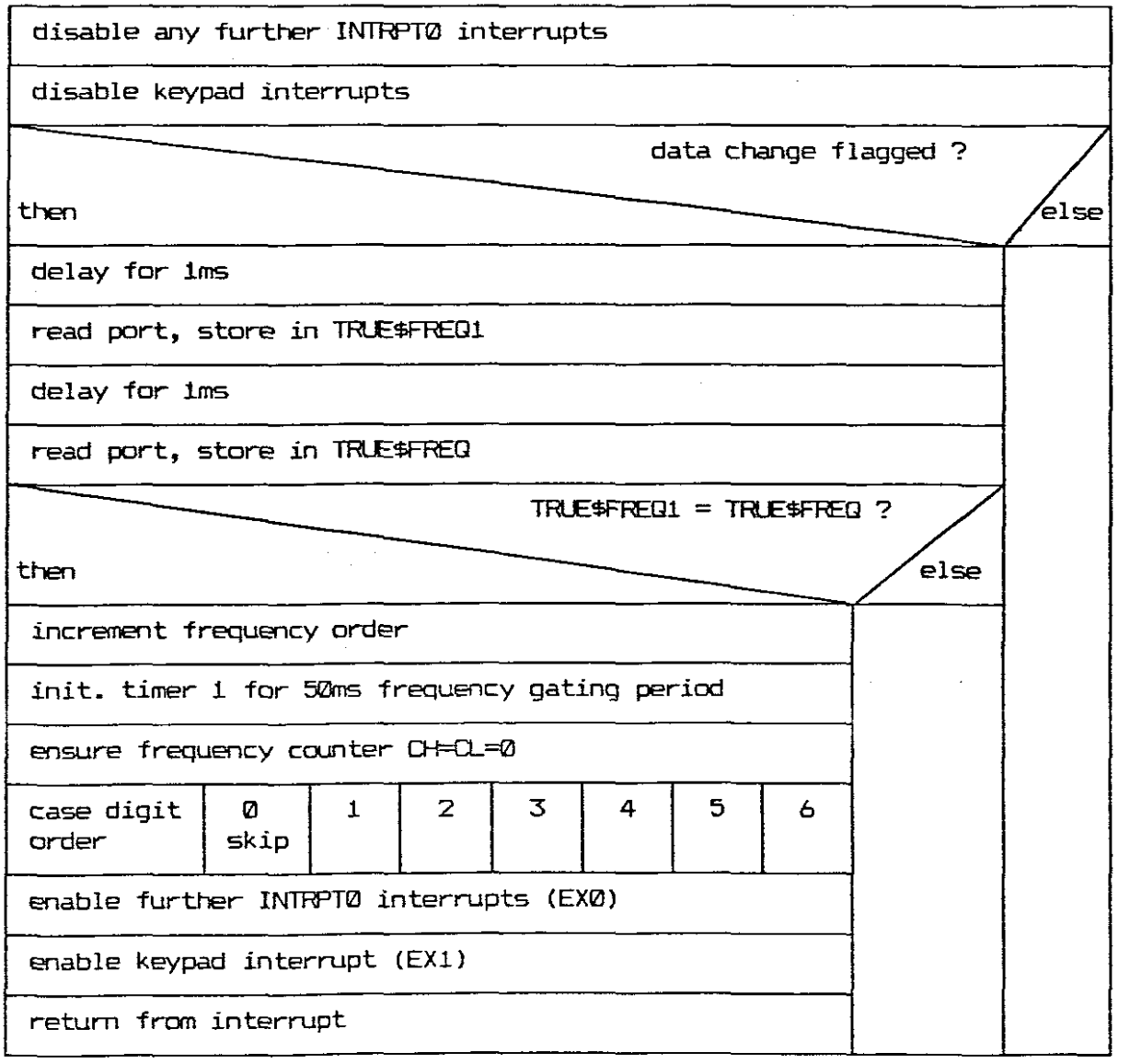

abandon all readings ?
then
else
re-initialize flags
re-initialize variables
clear display

IF 7

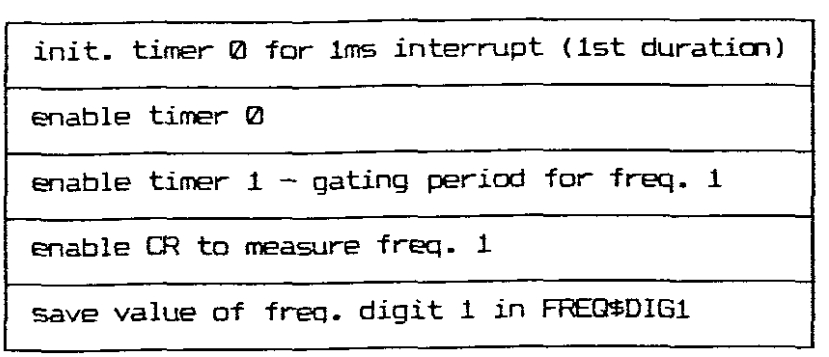

P - Previous display required ?
then
else
clear DISPLAY$OPTION%2 flag
clear display variable and exit
clear display

9.2 External interrupt 0, performs real-time selcall analysis.

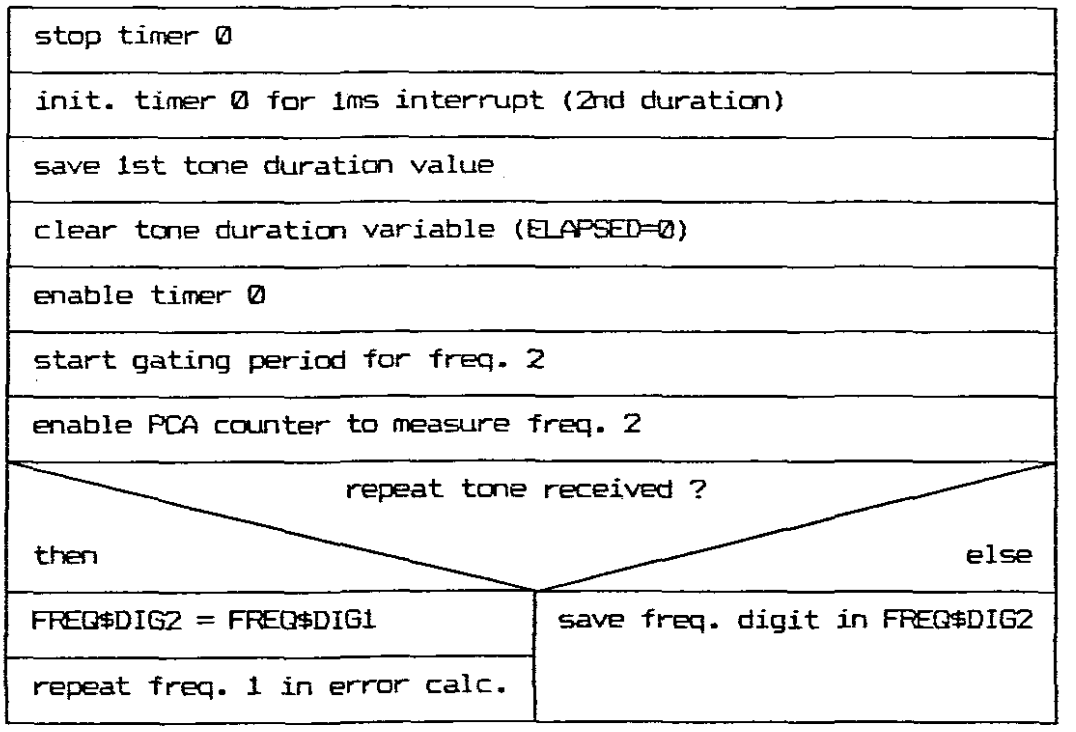
INTRPT0



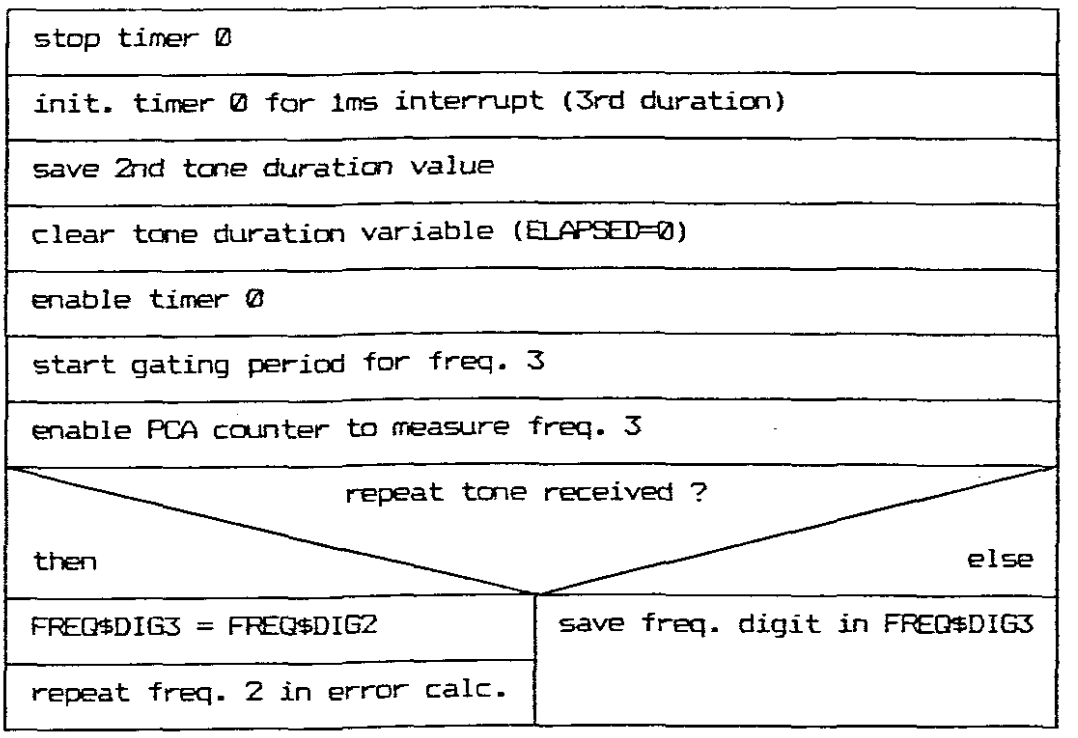
CASE 1



CASE 2


stop timer 0
init. timer 0 for 1ms interrupt (2nd duration)
save 1st tone duration value
clear tone duration variable (ELAPSED=0)
enable timer 0
start gating period for freq. 2
enable PCA counter to measure freq. 2
repeat tone received ?
then
else
FREQ$DIG2 = FREQ$DIG1
repeat freq. 1 in error calc.
save freq. digit in FREQ$DIG2

CASE 3


stop timer 0
init. timer 0 for 1ms interrupt (3rd duration)
save 2nd tone duration value
clear tone duration variable (ELAPSED=0)
enable timer 0
start gating period for freq. 3
enable PCA counter to measure freq. 3
repeat tone received ?
then
else
FREQ$DIG3 = FREQ$DIG2
repeat freq. 2 in error calc.
save freq. digit in FREQ$DIG3

CASE 4

stop timer 0
init. timer 0 for 1ms interrupt (4th duration)
save 3rd tone duration value
clear tone duration variable (ELAPSED=0)
enable timer 0
start gating period for freq. 4
enable PCA counter to measure freq. 4
repeat tone received ?

then	else
FREQ$DIG4 = FREQ$DIG3	save freq. digit in FREQ$DIG4
repeat freq. 3 in error calc.	

CASE 5

stop timer 0
init. timer 0 for 1ms interrupt (5th duration)
save 4th tone duration value
clear tone duration variable (ELAPSED=0)
enable timer 0
start gating period for freq. 5
enable PCA counter to measure freq. 5
repeat tone received ?

then	else
FREQ$DIG5 = FREQ$DIG4	save freq. digit in FREQ$DIG5
repeat freq. 4 in error calc.	

CASE 6

stop timer 0
save 5th tone duration value
clear tone duration variable (ELAPSED=0)
reset frequency no. pointer
reset received digit order flag
set DONE = 1, to display results

9.3 Real-time duration timer interrupt.

TIMER0

TF0 cleared on interrupt vectoring
increment tone duration counter
initialize timer reload value
return from interrupt

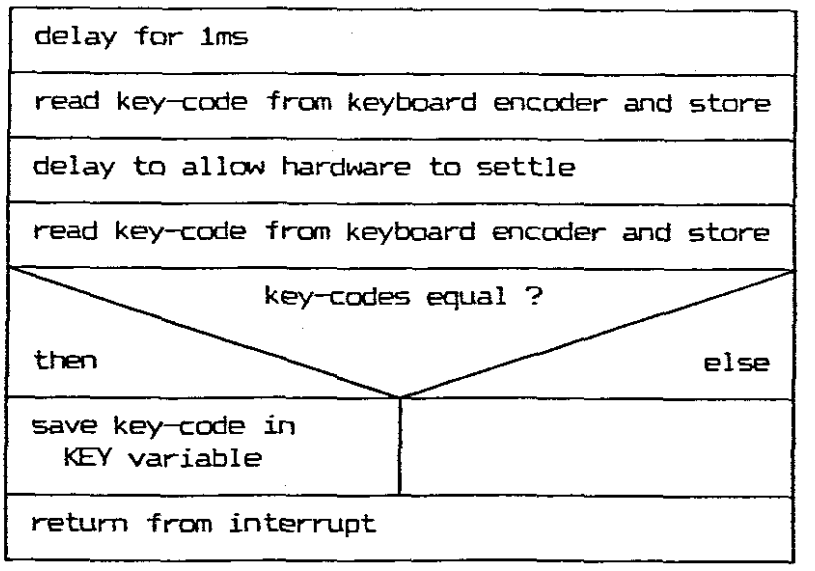
9.4 Real-time frequency count for each frequency digit.

TIMER 1

stop Capture Register counter	
stop timer 1	
increment order of received frequency	
case 0	skip
case 1	store counter value in 1st freq. variables
case 2	store counter value in 2nd freq. variables
case 3	store counter value in 3rd freq. variables
case 4	store counter value in 4th freq. variables
case 5	store counter value in 5th freq. variables
clear Capture Register counter (CH=CL=00)	
return from interrupt	

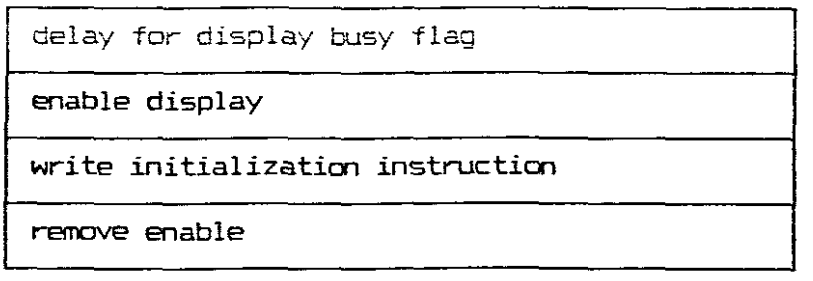
9.5 Keypad interrupt, stores key pressed on keypad.

INTRPT1



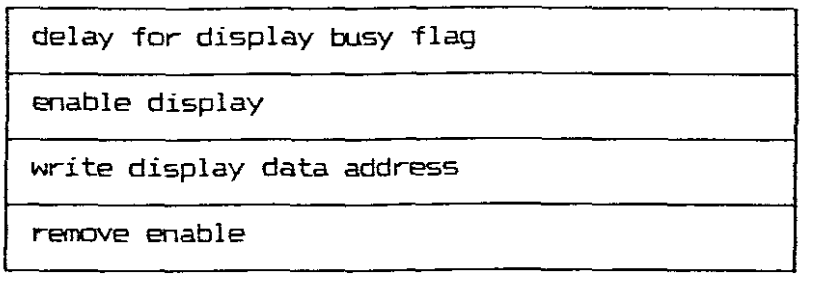
9.6 Initialization of display on power-up.

DISPLAY$SETUP



9.7 Locate address where character is to be displayed.

DISPLAY$ADDR



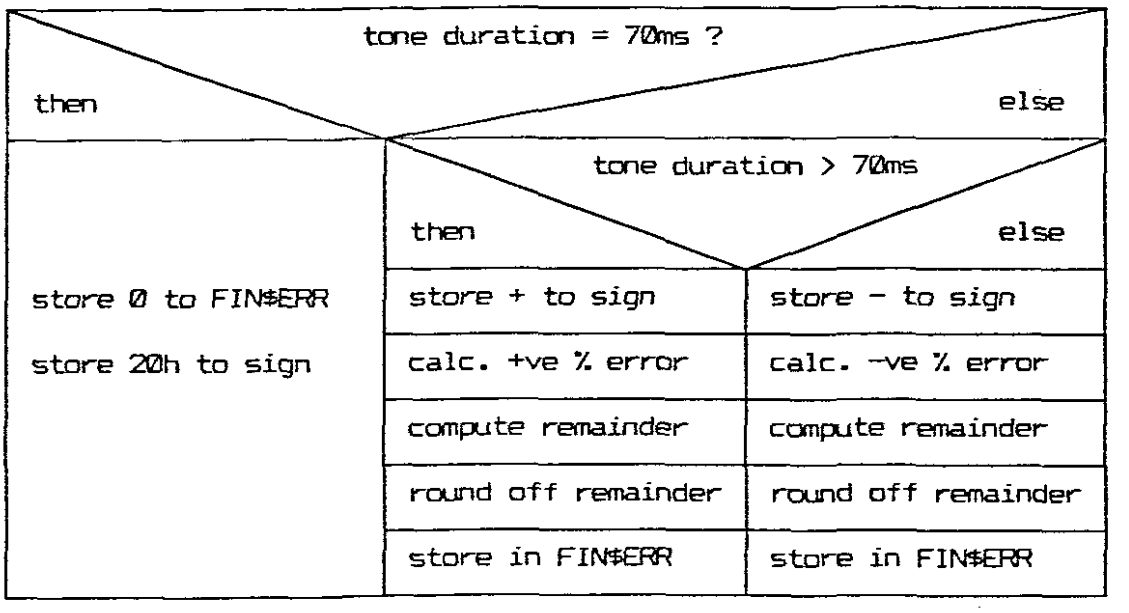
9.8 Display a character at previously defined display address.

DISPLAY$CHAR

write to display data register
delay for display busy flag
enable display
delay for display busy flag
write ASCII byte to display
delay for display busy flag
remove enable & register select

9.9 Calculate tone duration error for each of the 5 tones.

CALC$TDUR$ERR



9.10 Initiate the displaying of MODE 1.

SELECTDKEY$MSG

clear display
initialize display start address
initialize no. of chars. to write to display
call D$KEY to execute function

9.11 Write display MODE 1 to the display.

D$KEY

initialize display address
write MODE 1 to the display starting at the initialized address

9.12 Initiate the displaying of MODE 2.

OPTION$DISPLAY$1

clear display
initialize display start address
initialize no. of chars. to write to display
call OPTION$DONATION$1 to execute function

9.13 Write display MODE 2 to the display.

OPTION$DONATION$1

initialize display address
write MODE 2 to the display starting at the initialized address

9.14 Display the frequency digits for each of the 5 tones.

DECODED$TONE

clear display
initialize display address
write MODE 4 heading to the display
write QUIT$MSG to the display
init. display address for freq. digits 1 - 5
write each ASCII freq. digit to the display

9.15 'A-Quit' message for various MODES of display.

QUIT$MSG

initialize address on display
write 'A-Quit' message to the display

9.16 Write MODE 5 heading and initiate display of freq. no.'s.

FREQ$HEADING

clear display
initialize display address
write MODE 5's heading to the display
call FREQ$DONATION to write freq. no.'s
write QUIT$MSG to the display

9.17 Write frequency no.'s for MODE 5 and MODE 6.

FREQ$DONATION

initialize display address
write frequency numbers for MODE 5 & MODE 6

9.18 Calculate frequency from no. of gated counts.

FREQ$CALC

CONTR$H & CONTR$L = freq. counts from the RTI
convert LS nibble and MS nibble to words
convert gated μs counts to seconds
scale down frequency to display std. freq.
convert freq. in hex to ASCII for displaying
store freq. values in global variable

9.19 Display frequencies at initialized addresses for MODE 5.

DISPLAY$FREQ

initialize display address
display each of the 4 freq. chars for each freq.

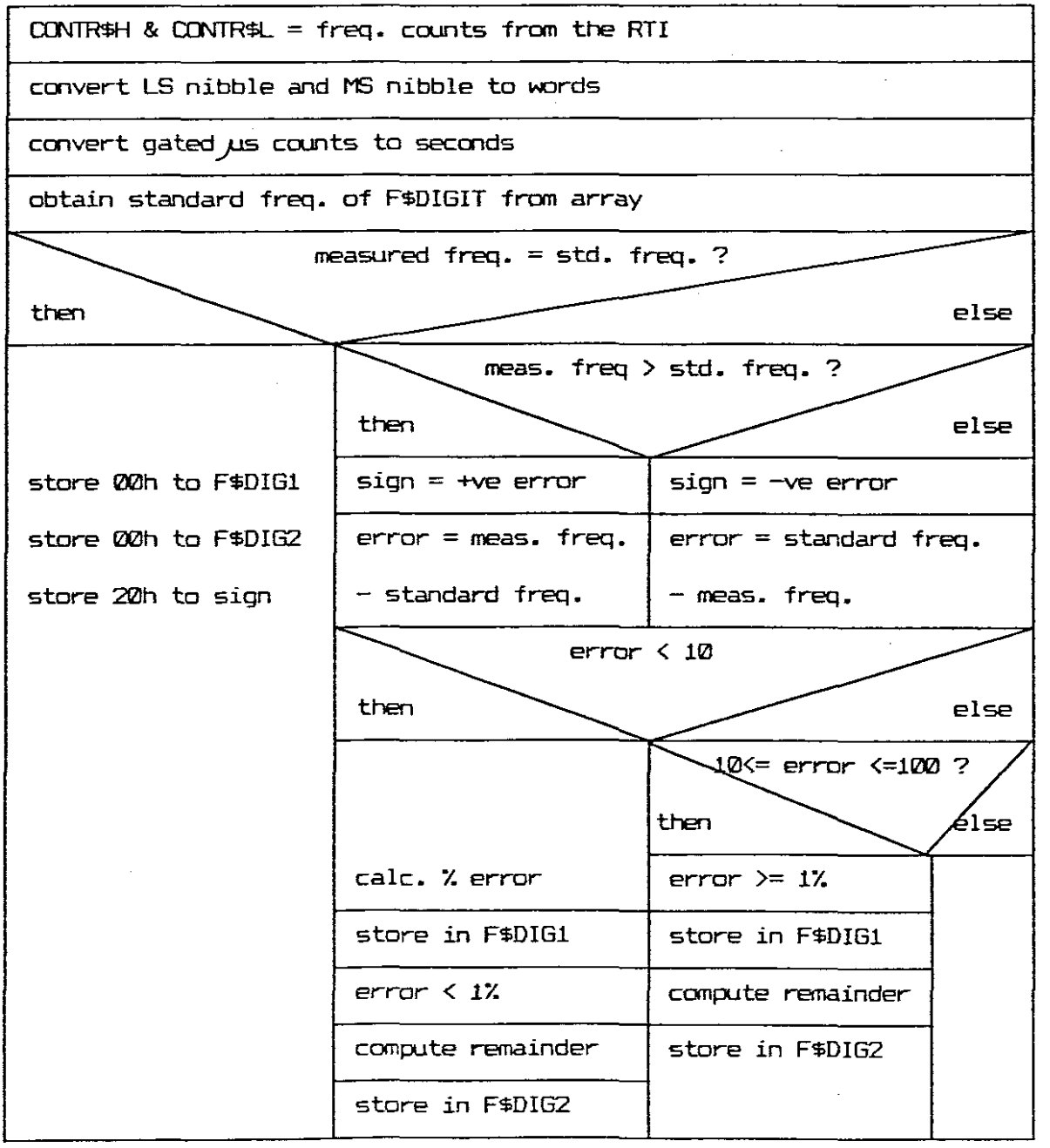
9.20 Write MODE 6 heading and initiate display of freq. no.'s.

FREQERRHEADING

clear display
initialize display address
write MODE 6's heading to the display
call FREQ$DONATION to write freq. no.'s
write QUIT$MSG to the display

9.21 Calculate frequency error for each of the 5 tones.

CALC$FREQ$ERR



9.22 Display the computed parameters for MODE 6.

SHOW$FREQ$ERR

initialize display address
display +/- error
write F$DIG1 to display
write decimal point to display
write F$DIG2 to display
call PERC$DISPLAY (write % sign to display)

9.23 Display '%' character used in MODE 6 and MODE 8.

PERC$DISPLAY

L = 31	
do while L < 32	
	write '%' character to display
	L = L + 1

9.24 Initiates the displaying of MODE 3.

OPTION$DISPLAY$2

clear display
initialize display start address
initialize no. of chars. to write to display
call OPTION$DONATION$2 to execute function

9.25 Write display MODE 3 to the display.

OPTION$DONATION$2

initialize display address
write MODE 3 to the display starting at the initialized address

9.26 Display MODE 7 heading and analysed parameters.

TONE$DURATIONS

clear the display
initialize array pointer to display MODE 7
call TONE$DURAT$MSG
write 't1=' to display
call DISPLAY$CHAR
call ms%DISPLAY
re-initialize array pointer and display tone durations 2, 3, 4 and 5
write 'A-Quit' to the display

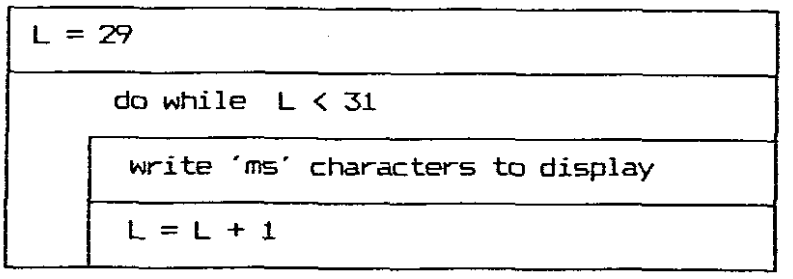
9.27 MODE 7 heading.

TONE$DURAT$MSG

initialize display address
do while MODE 7 heading not complete
write MODE 7's heading to the display

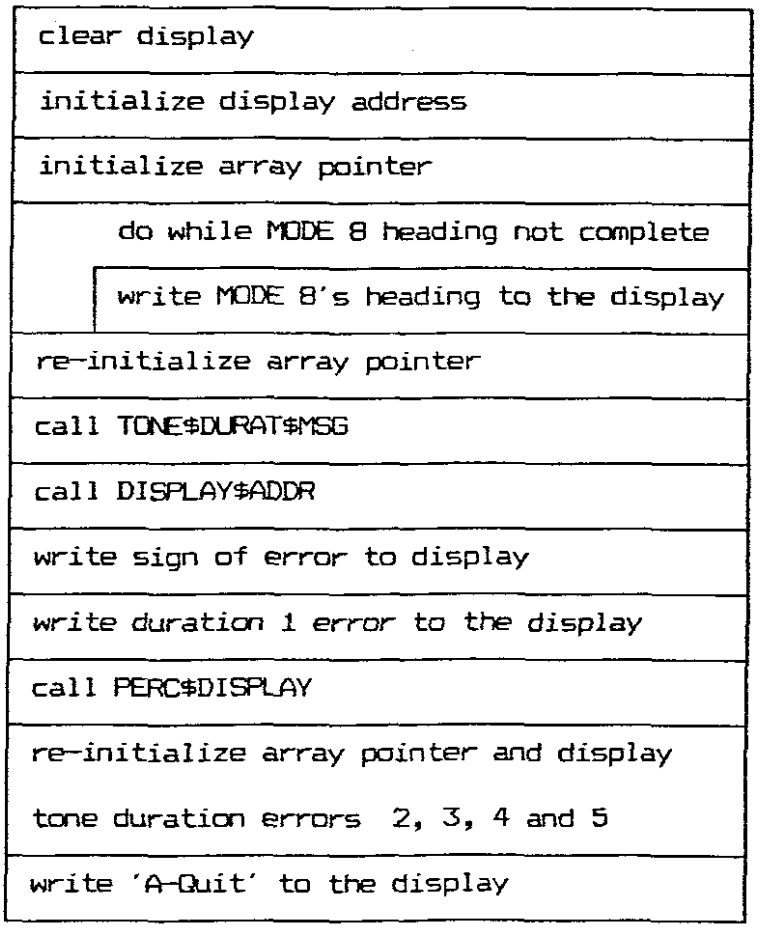
9.28 Display 'ms' character used in MODE 7.

ms$DISPLAY



9.29 Display heading and analysed parameters for MODE 8.

TONE$DURATION$ERR



APPENDIX D

5-TONE ZVEI ENCODER ANALYSER SOFTWARE LISTING

APPENDIX D

10. 5-TONE ZVEI ENCODER ANALYSER SOFTWARE LISTING

The following pages represents the software listing of the 5-Tone ZVEI Encoder Analyser PLM-51 program.

```
DOS 3.30 (038-N) PL/M-51 V1.2
COMPILER INVOKED BY:  C:\PLM\PLM51.EXE \PLM\PROJECTS\ZVEI.PLM DEBUG ROM(LARGE)


  1   1     ZVEI : do ;

                  SOFTWARE LISTING FOR THE
                5-TONE ZVEI ENCODER ANALYSER

  2   1     declare eq literally 'literally' ;

            $nolist
            $include(REG252.DCL)
            $list

                        PROGRAM VARIABLES

  6   1     declare ( L, DIG$ORDER, FREQ$DIG1, FREQ$DIG2, FREQ$DIG3, FREQ$DIG4,
                      FREQ$DIG5, T_DUR1, T_DUR2, T_DUR3, T_DUR4, T_DUR5, T_DURAT1,
                      T_DURAT2, T_DURAT3, T_DURAT4, T_DURAT5, ELAPSED, KEY )        byte main ;

  7   1     declare ( DUR$ERR$MSB1, DUR$ERR$LSB1, DUR$ERR$MSB2, DUR$ERR$LSB2,
                      DUR$ERR$MSB3, DUR$ERR$LSB3, DUR$ERR$MSB4, DUR$ERR$LSB4,
                      DUR$ERR$MSB5, DUR$ERR$LSB5, CURRENT$FREQ )                    byte main ;

  8   1     declare ( F$DIG1, F$DIG2, F$DIG3, F$DIG4 )                            byte main ;

  9   1     declare ( CONTR1$H, CONTR1$L, CONTR2$H, CONTR2$L, CONTR3$H,
                      CONTR3$L, CONTR4$H, CONTR4$L, CONTR5$H, CONTR5$L )            word main ;

 10   1     declare ( DONE, DISPLAY$RESULTS, DISPLAY$OPTION$1, DISPLAY$OPTION$2,
                      REPEAT$FREQ$1, REPEAT$FREQ$2, REPEAT$FREQ$3, REPEAT$FREQ$4 )  bit main ;


 11   1     declare ( DUR$MSB1, DUR$LSB1, DUR$MSB2, DUR$LSB2, DUR$MSB3, DUR$LSB3,
                      DUR$MSB4, DUR$LSB4, DUR$MSB5, DUR$LSB5, SIGN, FIN$ERR,
                      T$DURAT1$ERR, T$DURAT2$ERR, T$DURAT3$ERR, T$DURAT4$ERR,
                      T$DURAT5$ERR, POS$NES1, POS$NES2, POS$NES3, POS$NES4, POS$NES5 ) byte idata ;

                                                        /* ZVEI std. freq.          */
 12   1     declare STD$FREQ$TABLE (11) word constant (2400D, 1060D, 1160D, 1270D, 1400D,
                                                       1530D, 1670D, 1830D, 2000D, 2200D,
                                                       2600D) ;

 13   1     declare D_CHNG bit at (91H) reg ;          /* data change flag         */

 14   1     declare OFF         eq '0',          ON    eq '1',      /* flag equates */
                    CLEAR       eq '0',          SET   eq '1',

                    CLR$DISP    eq '01H',  FUNC$SET   eq '38H',    /* display instructions */
                    DISP$ON$OFF eq '0CH',  LCD$INST   eq '20H',
                    LCD$DATA    eq '0A0H',

                    FOREVER     eq 'while 1' ;
```

DISPLAY MODE MESSAGES

```
15  1   declare MESSAGE$1 (45) byte constant ('Ready To ReceiveIVEI Signal...Key Transmitter') ;
16  1   declare MESSAGE$2 (46) byte constant ('Signal ReceivedSelect D-Key forDisplay Options') ;
17  1   declare MESSAGE$3 (20) byte constant ('Decoded DigitsA-Quit') ;
18  1   declare MESSAGE$4 (33) byte constant ('Tone Durations:t1=t2=t3=t4=t5=ms% ') ;
19  1   declare MESSAGE$5 (14) byte constant ('Duration Error') ;
20  1   declare MESSAGE$6 (49) byte constant ('Select:-  C-Next1-Tone Digits2-Freq.3-Freq. Error') ;
21  1   declare MESSAGE$7 (64) byte constant ('Select:- C-Prev.4-Tone Durations5-Duration Error6-Quit Readings');
22  1   declare MESSAGE$8 (29) byte constant ('Tone Freq.(Hz)F1=F2=F3=F4=F5=') ;
23  1   declare MESSAGE$9 (15) byte constant ('Frequency Error') ;
```

REAL-TIME INTERRUPT PROCEDURES

```
24  2   INTRPT0: procedure interrupt 0 using 2 ;              /* ext. interrupt to process each   */
25  2      declare(TRUE$FREQ, TRUE$FREQ1) byte ;              /* of the 5-tones                   */

26  2          EX0 = off ;                                    /* disable all ext. interrupts      */
27  2          EX1 = off ;

28  3          if D_CHNG = 1 then do ;
30  3             call time(10) ;
31  3             TRUE$FREQ1 = shr((P1 and 78H),3) ;          /* ensure data change received      */
32  3             call time(10) ;
33  3             TRUE$FREQ =  shr((P1 and 78H),3) ;

34  4                 if TRUE$FREQ1 = TRUE$FREQ then do ;

36  4                    TH1 = 3CH ;                         /* 50ms timing window               */
37  4                    TL1 = 0AFH ;                        /* for freq. gating                 */
38  4                    CH = 00H ;                          /* reset freq. counter              */
39  4                    CL = 00H ;

40  4                    DIG$ORDER = DIG$ORDER + 1 ;         /* order tone sequence              */
41  5                    do case DIG$ORDER ;
42  5                       ;                                /* case 0                           */
43  6                       do ;                             /* case 1                           */
44  6                          TH0 = 0FCH ;                  /* timer reload value               */
45  6                          TL0 = 17H ;                   /* for 1st tone duration            */
46  6                          TR0 = 1 ;                     /* start tone duration timer        */
47  6                          TR1 = 1 ;                     /* start gating period for freq1.   */
48  6                          CR = 1 ;                      /* enable PCA ctr. to meas. freq1.  */
49  6                          FREQ$DIG1 = TRUE$FREQ ;       /* save freq. digit                 */
50  6                       end ;

51  6                       do ;                             /* case 2                           */
52  6                          TR0 = 0 ;                     /* stop tone duration timer         */
53  6                          TH0 = 0FCH ;                  /* timer reload value               */
54  6                          TL0 = 17H ;                   /* for 2nd tone duration            */
55  6                          T_DUR1 = ELAPSED ;            /* save 1st tone duration value     */
56  6                          ELAPSED = 00H ;               /* clear tone duration variable     */
57  6                          TR0 = 1 ;                     /* start tone duration timer        */
```

```
 58  6                  TR1 = 1 ;                  /* start gating period for freq2.      */
 59  6                  CR =  1 ;                  /* enable PCA ctr. to meas. freq2.     */

 60  7                  if TRUE$FREQ = 0EH then do ;  /* repeat tone received ?           */
 62  7                     FREQDIG2 = FREQ$DIG1 ;  /* save previous freq. digit           */
 63  7                     REPEAT$FREQ$1 = set ;   /* control flag used during freq.      */
 64  7                     end ;                   /* error calculations                  */
 65  7                  else do ;
 66  7                     FREQ$DIG2 = TRUE$FREQ ; /* save freq. digit                    */
 67  7                     end ;
 68  6               end ;

 69  6               do ;                          /* case 3                              */
 70  6                  TR0 = 0 ;                  /* stop tone duration timer            */
 71  6                  TH0 = 0FCH ;               /* timer reload value                  */
 72  6                  TL0 = 17H ;                /* for 3rd tone duration               */
 73  6                  T_DUR2 = ELAPSED ;         /* save 2nd tone duration value        */
 74  6                  ELAPSED = 00H ;            /* clear tone duration variable        */
 75  6                  TR0 = 1 ;                  /* start tone duration timer           */
 76  6                  TR1 = 1 ;                  /* start gating period for freq3.      */
 77  6                  CR = 1 ;                   /* enable PCA ctr. to meas. freq3.     */

 78  7                  if TRUE$FREQ = 0EH then do ;  /* repeat tone received ?           */
 80  7                     FREQDIG3 = FREQ$DIG2 ;  /* save previous freq. digit           */
 81  7                     REPEAT$FREQ$2 = set ;   /* control flag used during freq.      */
 82  7                     end ;                   /* error calculations                  */
 83  7                  else do ;
 84  7                      FREQ$DIG3 = TRUE$FREQ ;  /* save freq. digit                 */
 85  7                  end ;
 86  6               end ;

 87  6               do ;                          /* case 4                              */
 88  6                  TR0 = 0 ;                  /* stop tone duration timer            */
 89  6                  TH0 = 0FCH ;               /* timer reload value                  */
 90  6                  TL0 = 17H ;                /* for 4th tone duration               */
 91  6                  T_DUR3 = ELAPSED ;         /* save 3rd tone duration value        */
 92  6                  ELAPSED = 00H ;            /* clear tone duration variable        */
 93  6                  TR0 = 1 ;                  /* start tone duration timer           */
 94  6                  TR1 = 1 ;                  /* start gating period for freq4.      */
 95  6                  CR = 1 ;                   /* enable PCA ctr. to meas. freq4.     */

 96  7                  if TRUE$FREQ = 0EH then do ;  /* repeat tone received ?           */
 98  7                     FREQDIG4 = FREQ$DIG3 ;  /* save previous freq. digit           */
 99  7                     REPEAT$FREQ$3 = set ;   /* control flag used during freq.      */
100  7                     end ;                   /* error calculations                  */
101  7                  else do ;
102  7                     FREQ$DIG4 = TRUE$FREQ ;  /* save freq. digit                 */
103  7                  end ;
104  6               end ;

105  6               do ;                          /* case 5                              */
106  6                  TR0 = 0 ;                  /* stop tone duration timer            */
107  6                  TH0 = 0FCH ;               /* timer reload value                  */
108  6                  TL0 = 17H ;                /* for 5th tone duration               */
109  6                  T_DUR4 = ELAPSED ;         /* save 4th tone duration value        */
110  6                  ELAPSED = 00H ;            /* clear tone duration variable        */
111  6                  TR0 = 1 ;                  /* start tone duration timer           */
```

```
112  6                          TR1 = 1 ;                    /* start gating period for freq5.      */
113  6                          CR = 1 ;                     /* enable PCA ctr. to meas. freq5.     */

114  7                          if TRUE$FREQ = 0EH then do ; /* repeat tone received ?              */
116  7                              FREQDIG5 = FREQ$DIG4 ;   /* save previous freq. digit           */
117  7                              REPEAT$FREQ$4 = set ;    /* control flag used during freq.      */
118  7                              end ;                    /* error calculations                  */
119  7                          else do ;
120  7                              FREQ$DIG5 = TRUE$FREQ ;  /* save freq. digit                    */
121  7                              end ;
122  6                      end ;

123  6                      do ;                             /* case 6                              */
124  6                          TR0 = 0 ;                    /* stop tone duration timer            */
125  6                          T_DUR5 = ELAPSED ;           /* save 5th tone duration value        */
126  6                          ELAPSED = 00H ;              /* clear tone duration variable        */
127  6                          CURRENT$FREQ = 0H ;          /* reset freq. no. pointer             */
128  6                          DIG$ORDER = 0 ;              /* reset digit order flag              */
129  6                          DONE = 1 ;                   /* flag background task,               */
130  6                      end ;                            /* ready to display results            */
131  5                  end ;
132  4              end ;
133  3          end ;
134  2          EXI = on ;                                   /* enable all ext. interrupts          */
135  2          EX0 = on ;
136  1      end INTRPT0 ;

137  2      INTRPT1: procedure interrupt 2 using 3 ;         /* get key pressed                     */
138  2          declare (KEYIN, KEYIN1) byte ;
139  2          call time(10) ;                              /* allow hardware to settle            */
140  2          KEYIN = not(P3 and 0F0H) ;                   /* and mask off bits                   */
141  2          call time(10) ;
142  2          KEYIN1 = not(P3 and 0F0H) ;
143  3          if KEYIN = KEYIN1 then do ;
145  3              KEY = KEYIN ;                            /* key = key pressed on keypad         */
146  3          end ;
147  1      end INTRPT1 ;

148  2      TIMER0: procedure interrupt 1 using 3 ;          /* duration timer interrupt            */
149  2          ELAPSED = ELAPSED + 1 ;                      /* 1ms counter                         */
150  2          TH0 = 0FCH ;                                 /* timer reload value                  */
151  2          TL0 = 17H ;                                  /* for 1ms interrupt                   */
152  1      end TIMER0 ;

153  2      TIMER1: procedure interrupt 3 using 1 ;          /* interrupt to store gated            */
154  2          CR = 0 ;                                     /* freq. count                         */
155  2          TR1 = 0 ;                                    /* stop gating period                  */
156  2          CURRENT$FREQ = CURRENT$FREQ + 1 ;            /* freq. count for each of the         */
157  3          do case CURRENT$FREQ ;                       /* 5 freq. digits                      */
158  3              ;                                        /* case 0                              */
159  4              do ;                                     /* case1                               */
160  4                  CONTR1$H = CH ;                      /* freq1 high counter value            */
161  4                  CONTR1$L = CL ;                      /* freq1 low counter value             */
162  4              end ;

163  4              do ;                                     /* case2                               */
164  4                  CONTR2$H = CH ;                      /* freq2 high counter value            */
```

```
165  4                  CONTR2$L = CL ;                    /* freq2 low counter value          */

166  4              end ;

167  4              do ;                                   /* case3                            */
168  4                  CONTR3$H = CH ;                    /* freq3 high counter value         */
169  4                  CONTR3$L = CL ;                    /* freq3 low counter value          */
170  4              end ;

171  4              do ;                                   /* case 4                           */
172  4                  CONTR4$H = CH ;                    /* freq4 high counter value         */
173  4                  CONTR4$L = CL ;                    /* freq4 low counter value          */
174  4              end ;

175  4              do ;                                   /* case5                            */
176  4                  CONTR5$H = CH ;                    /* freq5 high counter value         */
177  4                  CONTR5$L = CL ;                    /* freq5 low counter value          */
178  4              end ;
179  3          end ;
180  2      CH = 00H ;                                     /* clear PCA counter                */
181  2      CL = 00H ;
182  1      end TIMER1 ;
```

GENERAL PROCEDURES

```
183  2      DISPLAY$SETUP : procedure(LCD,INST) ;          /* display init. on power-up        */
184  2          declare (LCD, INST) byte ;
185  2          call time(100) ;                           /* allow LCD to settle              */
186  2          P0 = P0 or LCD ;                           /* enable display                   */
187  2          P2 = INST ;                                /* instr. to initialize             */
188  2          P0 = P0 and 0DFH ;                         /* remove enable                    */
189  1      end DISPLAY$SETUP ;

190  2      DISPLAY$ADDR : procedure(LCD,ADDR) ;           /* set character address            */
191  2          declare (LCD, ADDR) byte ;
192  2          call time(100) ;                           /* allow LCD to settle              */
193  2          P0 = P0 or LCD ;                           /* enable display                   */
194  2          P2 = ADDR ;                                /* DD address                       */
195  2          P0 = P0 and not(LCD) ;                     /* remove enable                    */
196  1      end DISPLAY$ADDR ;

197  2      DISPLAY$CHAR : procedure(LCD,DATA) ;           /* displ char. at addr.             */
198  2          declare (LCD, DATA) byte ;
199  2          P0 = P0 or 80H ;                           /* write to DR                      */
200  2          call time(10) ;                            /* allow display time to settle     */
201  2          P0 = P0 or LCD ;                           /* enable display                   */
202  2          call time(10) ;                            /* allow display time to settle     */
203  2          P2 = DATA ;                                /* write ascii byte                 */
204  2          call time(10) ;                            /* allow display time to settle     */
205  2          P0 = P0 and 1FH ;                          /* remove enable and RS             */
206  1      end DISPLAY$CHAR ;

207  2      CALC$TDUR$ERR : procedure (T_DURAT) ;          /* calculate tone duration error    */
208  2                  declare (T_DURAT, SCALE ) byte ;
209  2                  declare (ERR, OVER, WORK$OVER) word ;
```

```
210   3              if T_DURAT=46H then do ;                    /* any tone duration error ?  */
212   3                  FIN$ERR = 0H ;                          /* no !!                      */
213   3                  SIGN = 20H ;                            /* no sign required           */
214   3                  end ;
215   3               else do ;
216   4                  if T_DURAT > 46H then do ;              /* yes !!                     */
218   4                     SIGN = 2BH ;                         /* +ve error                  */
219   4                     ERR = (T_DURAT - 46H)*100 ;          /* calculate % error          */
220   4                     FIN$ERR = ERR / 46H ;
221   4                     OVER = FIN$ERR * 46H ;
222   4                     WORK$OVER = (ERR - OVER) * 0AH ;     /* round off remainder        */
223   4                     SCALE = WORK$OVER / 46H ;
224   4                     if SCALE >= 5 then FIN$ERR = FIN$ERR+1 ;
226   4                     end ;
227   4                  else do ;
228   4                     SIGN = 2DH ;                         /* -ve error                  */
229   4                     ERR = (46H - T_DURAT)*100 ;          /* calculate % error          */
230   4                     FIN$ERR = ERR / 46H ;
231   4                     OVER = FIN$ERR * 46H ;
232   4                     WORK$OVER = (ERR - OVER) * 0AH ;     /* round off remainder        */
233   4                     SCALE = WORK$OVER / 46H ;
234   4                     if SCALE >= 5 then FIN$ERR = FIN$ERR+1 ;
236   4                     end ;
237   3                end ;
238   1     end CALC$TDUR$ERR ;

239   2     D$KEY : procedure(ADDRS, E$PRAM) ;                    /* display MODE 1             */
240   2        declare(ADDRS, E$PRAM) byte ;                      /* message written to display */
241   2        call DISPLAY$ADDR(LCD$INST,ADDRS) ;                /* setup display address      */
242   3        do while L < E$PRAM ;
243   3           call DISPLAY$CHAR(LCD$DATA, MESSAGE2$(L)) ;     /* display MODE1              */
244   3           L = L + 1 ;
245   3        end ;
246   1     end D$KEY ;

247   2     SELECT$D$KEY$MSG : procedure ;                        /* initiate displaying of MODE 1 */
248   2        call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;           /* clear display              */
249   2        L = 0 ;
250   2        call D$KEY( 80H,  15) ;                            /* start addr. & no. of       */
251   2        call D$KEY( 90H,  31) ;                            /* chars. to write to display */
252   2        call D$KEY( 0D0H, 46) ;
253   1     end SELECT$D$KEY$MSG ;

254   2     OPTION$DONATION$1 : procedure(ADDRS, E$PRAM) ;        /* writes chars. to display   */
255   2        declare(ADDRS, E$PRAM) byte ;                      /* based on parameters passed */
256   2        call DISPLAY$ADDR(LCD$INST,ADDRS) ;                /* init. display address      */
257   3        do while L < E$PRAM ;                              /* write chars. to display    */
258   3           call DISPLAY$CHAR(LCD$DATA, MESSAGE6$(L)) ;     /* MODE 2                     */
259   3           L = L + 1 ;
260   3        end ;
261   1     end OPTION$DONATION$1 ;

262   2     OPTION$DISPLAY$1 : procedure ;                        /* display MODE 2             */
263   2        call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;           /* clear display              */
264   2        L = 0 ;
265   2        call OPTION$DONATION$1( 80H, 16) ;                 /* start addr. & no. of       */
266   2        call OPTION$DONATION$1(0C0H, 29) ;                 /* chars. to write to display */
```

```
267  2        call OPTION$DONATION$1( 90H, 36) ;          /* passed to proc.                    */
268  2        call OPTION$DONATION$1(0D0H, 49) ;
269  1     end OPTION$DISPLAY$1 ;

270  2     QUIT$MSG : procedure ;
271  2        call DISPLAY$ADDR(LCD$INST,0DAH) ;          /* display A-Quit                     */
272  2        L = 14;
273  3        do while L < 20 ;
274  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE$3(L)) ;  /* write chars. to display       */
275  3           L = L + 1 ;
276  3        end ;
277  1     end QUIT$MSG ;

278  2     DECODED$TONE : procedure ;                     /* display decoded tones'             */
                                                         /* freq. digits                       */
279  2        call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;    /* clear display                      */
280  2        call DISPLAY$ADDR(LCD$INST,81H) ;           /* init. display address              */
281  2        L = 0 ;
282  3        do while L < 14 ;
283  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE$3(L)) ;  /* display MODE 4                */
284  3           L = L + 1 ;
285  3        end ;
286  2        call QUIT$MSG ;
287  2        call DISPLAY$ADDR(LCD$INST,92H) ;           /* init. display address              */
288  2        call DISPLAY$CHAR(LCD$DATA, (FREQ$DIG1 or 30H)) ;  /* write freq. digit 1          */
289  2        call time(10) ;                             /* allow display time to settle       */
290  2        call DISPLAY$ADDR(LCD$INST,95H) ;
291  2        call DISPLAY$CHAR(LCD$DATA, (FREQ$DIG2 or 30H)) ;  /* write freq. digit 2          */
292  2        call DISPLAY$ADDR(LCD$INST,97H) ;           /* init. display address              */
293  2        call DISPLAY$CHAR(LCD$DATA, (FREQ$DIG3 or 30H)) ;  /* write freq. digit 3          */
294  2        call DISPLAY$ADDR(LCD$INST,99H) ;           /* init. display address              */
295  2        call DISPLAY$CHAR(LCD$DATA, (FREQ$DIG4 or 30H)) ;  /* write freq. digit 4          */
296  2        call DISPLAY$ADDR(LCD$INST,9BH) ;           /* init. display address              */
297  2        call DISPLAY$CHAR(LCD$DATA, (FREQ$DIG5 or 30H)) ;  /* write freq. digit 5          */
298  1     end DECODED$TONE ;

299  2     FREQ$DONATION : procedure(ADDRS, E$PRAM) ;     /* write freq. no. for MODE's 5 & 6   */
300  2        declare(ADDRS, E$PRAM ) byte ;              /* to display                         */
301  2        call DISPLAY$ADDR(LCD$INST,ADDRS) ;         /* init. display address              */
302  3        do while L < E$PRAM ;
303  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE8$(L)) ;  /* display Fx= message           */
304  3           L = L + 1 ;
305  3        end ;
306  1     end FREQ$DONATION ;

307  2     FREQ$HEADING : procedure ;                     /* write heading for MODE 5           */
308  2        call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;    /* clear display                      */
309  2        call DISPLAY$ADDR(LCD$INST,81H) ;           /* display tone freq.(Hz)             */
310  2        L = 0 ;
311  3        do while L < 14 ;
312  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE$8(L)) ;  /* write tone freq.(Hz)          */
313  3           L = L + 1 ;                              /* to display                         */
314  3        end ;
315  2        call FREQ$DONATION(0C0H, 17) ;              /* display 'F1='                      */
316  2        call FREQ$DONATION(90H,  20) ;              /* display 'F2='                      */
317  2        call FREQ$DONATION(0D0H, 23) ;              /* display 'F3='                      */
318  2        call FREQ$DONATION(0C9H, 26) ;              /* display 'F4='                      */
```

```
 319  2          call FREQ$DONATION(99H,  29) ;                  /* display 'F5='                    */
 320  2          call QUIT$MSG ;
 321  1      end FREQ$HEADING ;

 322  2      FREQ$CALC : procedure (CONTR$H, CONTR$L ) ;         /* calc. freq. from no. of gated counts */
 323  2          declare (CONTR$H, CONTR$L, FREQ$IN, NEW$DIG1, NEW$DIG2 ) word ;
 324  2                  CONTR$L = double(CONTR$L) ;             /* nibble to byte conversion        */
 325  2                  CONTR$H = double(CONTR$H) ;
 326  2                  CONTR$H = shl(CONTR$H,8) ;              /* bit manipulation                 */
 327  2                  FREQ$IN = (CONTR$H or CONTR$L) * 05H ;  /* NB!! V.I.P COUNT * 20/4          */
                                                              /* 20 :- converts to sec.           */
 328  2                  F$DIG1 = ( FREQ$IN/3E8H ) or 30H ;      /*  4 :- freq. mult. by 4 so we     */
 329  2                  NEW$DIG1 = FREQ$IN mod 3E8H ;           /* must scale down for correct      */
                                                              /* display of std. input freq.      */
 330  2                  F$DIG2 = ( NEW$DIG1/64H ) or 30H ;      /* hex to ascii conv. for displ.    */
 331  2                  NEW$DIG2 = NEW$DIG1 mod 64H ;

 332  2                  F$DIG3 = ( NEW$DIG2/0AH ) or 30H ;      /* hex to ascii conv. for displ.    */

 333  2                  F$DIG4 = ( NEW$DIG2 mod 0AH ) or 30H ;  /* hex to ascii conv. for displ.    */
 334  1      end FREQ$CALC ;

 335  2      DISPLAY$FREQ : procedure(FREQ$ADDR, F$DIG1, F$DIG2, F$DIG3, F$DIG4) ;
 336  2          declare (FREQ$ADDR, F$DIG1, F$DIG2, F$DIG3, F$DIG4) byte ;
 337  2          call DISPLAY$ADDR(LCD$INST, FREQ$ADDR) ;        /* display freq. at correct         */
 338  2          call DISPLAY$CHAR(LCD$DATA, F$DIG1) ;           /* position on the display          */
 339  2          call DISPLAY$CHAR(LCD$DATA, F$DIG2) ;           /* for MODE 5                       */
 340  2          call DISPLAY$CHAR(LCD$DATA, F$DIG3) ;
 341  2          call DISPLAY$CHAR(LCD$DATA, F$DIG4) ;
 342  1      end DISPLAY$FREQ ;

 343  2      PERC$DISPLAY : procedure(L) ;                       /* display '%' for MODE 6           */
 344  2          declare (L) byte ;
 345  3          do while L < 32 ;
 346  3              call DISPLAY$CHAR(LCD$DATA, MESSAGE$4(L)) ; /* write '%' to display             */
 347  3              L = L + 1 ;
 348  3          end ;
 349  1      end PERC$DISPLAY ;

 350  2      FREQ$ERR$HEADING : procedure ;                      /* write heading for MODE 6         */
 351  2          call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;        /* clear display                    */
 352  2          call DISPLAY$ADDR(LCD$INST,80H) ;               /* init. display address            */
 353  2          L = 0 ;
 354  3          do while L < 15 ;
 355  3          call DISPLAY$CHAR(LCD$DATA, MESSAGE$9(L)) ;     /* write 'Frequency Error'          */
 356  3              L = L + 1 ;                                 /* heading to display               */
 357  3          end ;
 358  2          L = 14 ;
 359  2          call FREQ$DONATION(0C0H, 16) ;                  /* display 'F1='                    */
 360  2          L = 17 ;                                        /* init. display address            */
 361  2          call FREQ$DONATION(90H,  19) ;                  /* display 'F2='                    */
 362  2          L = 20 ;
 363  2          call FREQ$DONATION(0D0H, 22) ;                  /* display 'F3='                    */
 364  2          L = 23 ;
 365  2          call FREQ$DONATION(0C9H, 25) ;                  /* display 'F4='                    */
 366  2          L = 26 ;
 367  2          call FREQ$DONATION(99H,  28) ;                  /* display 'F5='                    */
```

```
368  2        call QUIT$MSG ;                          /* quit message for MODE 6          */
369  1    end FREQ$ERR$HEADING ;

370  2    CALC$FREQ$ERR : procedure(CONTR$H, CONTR$L, F$DIGIT) ;   /* calc. freq. error        */
371  2        declare (CONTR$H, CONTR$L, FREQ$IN, STD$FREQ, ERR ) word ;
372  2        declare (F$DIGIT, SCALE) byte ;
373  2            CONTR$L = double(CONTR$L) ;          /* nibble to byte conversion        */
374  2            CONTR$H = double(CONTR$H) ;
375  2            CONTR$H = shl(CONTR$H,4) ;           /* bit manipulation                 */
376  2            FREQ$IN = (CONTR$H or CONTR$L) * 05H ;   /* convert to seconds           */

377  2            STD$FREQ = STD$FREQ$TABLE(F$DIGIT) ;     /* get std. freq from array     */

378  3            if FREQ$IN = STD$FREQ then do ;      /* freq. meas. = std. ZVEI          */
380  3                SIGN = 20H ;                     /* freq. digit ?                    */
381  3                F$DIG1 = 00H ;                   /* error = 0                        */
382  3                F$DIG2 = 00H ;
383  3                end ;
384  3            else do ;
385  4                if FREQ$IN > STD$FREQ then do ;  /* freq. meas. > std. ZVEI freq. ?  */
387  4                    SIGN = 2BH ;                 /* + error                          */
388  4                    ERR = FREQ$IN - STD$FREQ ;
389  4                    end ;
390  4                else do ;
391  4                    SIGN = 2DH ;                 /* - error                          */
392  4                    ERR = STD$FREQ - FREQ$IN ;
393  4                    end ;

394  4                    if ERR < 0AH then do ;       /* error < 10 ?                     */
396  4                        ERR = (ERR*64H) ;        /* calc. error                      */
397  4                        F$DIG1 = ERR/STD$FREQ ;  /* use F$DIG1 as variables limited  */
398  4                        F$DIG2 = (ERR*0AH)/STD$FREQ ;   /* store                     */
399  4                        FREQ$IN = STD$FREQ*F$DIG2 ;     /* use FREQ$IN as variables limited */
400  4                        FREQ$IN =((ERR*0AH) - FREQ$IN)*0AH ;
401  4                        SCALE = FREQ$IN/STD$FREQ ;      /* error < 1%               */
402  4                        if SCALE >= 05H then F$DIG2 = F$DIG2 + 1 ;
404  4                        end ;
405  4                    else do ;
406  5                        if ERR >= 0AH and ERR <= 64H then do ; /* error between 10 and 100 ? */
408  5                            ERR = (ERR*64H) ;    /* error >= 1%                      */
409  5                            F$DIG1 = ERR/STD$FREQ ;   /* store                       */
410  5                            FREQ$IN=(ERR-(F$DIG1*STD$FREQ))*0AH ;
411  5                            F$DIG2 = FREQ$IN/STD$FREQ ;   /* store               */
412  5                        end ;
413  4                end ;
414  3            end ;
415  1    end CALC$FREQ$ERR ;

416  2    SHOW$FREQ$ERR : procedure (FREQ$ADDR, SIGN, F$DIG1, F$DIG2) ;
417  2        declare (FREQ$ADDR, SIGN, F$DIG1, F$DIG2) byte ;   /* display MODE 6         */
418  2            call DISPLAY$ADDR(LCD$INST, FREQ$ADDR) ;       /* init. display address  */
419  2            call DISPLAY$CHAR(LCD$DATA, SIGN) ;            /* display sign           */
420  2            call DISPLAY$CHAR(LCD$DATA, (F$DIG1 or 30H)) ; /* display error          */
421  2            call DISPLAY$CHAR(LCD$DATA, 2CH) ;             /* display decimal point  */
422  2            call DISPLAY$CHAR(LCD$DATA, (F$DIG2 or 30H)) ; /* display remainder      */
423  2            call PERC$DISPLAY(31) ;                        /* display units          */
424  1    end SHOW$FREQ$ERR ;
```

```
425  2     OPTION$DONATION$2 : procedure(ADDRS, E$PRAM) ;          /* writes chars. to display      */
426  2        declare(ADDRS, E$PRAM) byte ;                       /* based on parameters passed    */
427  2        call DISPLAY$ADDR(LCD$INST,ADDRS) ;                 /* init. display address         */
428  3        do while L < E$PRAM ;                               /* write chars. to display       */
429  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE7$(L)) ;      /* MODE 3                        */
430  3           L = L + 1 ;
431  3        end ;
432  1     end OPTION$DONATION$2 ;

433  2     OPTION$DISPLAY$2 : procedure ;                          /* display MODE 3                */
434  2        call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;            /* clear display                 */
435  2        L = 0 ;
436  2        call OPTION$DONATION$2( 80H, 16) ;                  /* start addr. & no. of          */
437  2        call OPTION$DONATION$2(0C0H, 32) ;                  /* chars. to write to display    */
438  2        call OPTION$DONATION$2( 90H, 48) ;                  /* passed to proc.               */
439  2        call OPTION$DONATION$2(0D0H, 63) ;
440  1     end OPTION$DISPLAY$2 ;

441  2     TONE$DURAT$MSG : procedure(ADDRS, E$PRAM) ;             /* display tone duration         */
442  2        declare(ADDRS, E$PRAM) byte ;                       /* heading for MODE 7            */
443  2        call DISPLAY$ADDR(LCD$INST,ADDRS) ;
444  3        do while L < E$PRAM ;
445  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE4$(L)) ;      /* write 'Tone Durations'         */
446  3           L = L + 1 ;                                      /* to display                    */
447  3        end ;
448  1     end TONE$DURAT$MSG ;

449  2     ms$DISPLAY : procedure(L) ;                             /* display 'ms' for MODE 7       */
450  2        declare (L) byte ;
451  3        do while L < 31 ;
452  3           call DISPLAY$CHAR(LCD$DATA, MESSAGE$4(L)) ;      /* write 'ms' to display         */
453  3           L = L + 1 ;
454  3        end ;
455  1     end ms$DISPLAY ;

456  2     TONE$DURATIONS : procedure ;                            /* display MODE 7                */
457  2        call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;            /* clear display                 */
458  2        L = 0 ;
459  2        call TONE$DURAT$MSG(81H, 14) ;                      /* heading for MODE 7            */

460  2        call TONE$DURAT$MSG( 0C0H, 17) ;                    /* write 't1='                   */
461  2        call DISPLAY$CHAR(LCD$DATA, DUR$MSB1) ;             /* display 1st tone duration     */
462  2        call DISPLAY$CHAR(LCD$DATA, DUR$LSB1) ;
463  2        call ms$DISPLAY(29) ;                               /* display units                 */

464  2        L = 17 ;                                            /* re-init. array pointer        */
465  2        call TONE$DURAT$MSG(90H, 20) ;                      /* write 't2='                   */
466  2        call DISPLAY$CHAR(LCD$DATA, DUR$MSB2) ;             /* display 2nd tone duration     */
467  2        call DISPLAY$CHAR(LCD$DATA, DUR$LSB2) ;
468  2        call ms$DISPLAY(29) ;                               /* display units                 */

469  2        L = 20 ;                                            /* re-init. array pointer        */
470  2        call TONE$DURAT$MSG(0D0H, 23) ;                     /* write 't3='                   */
471  2        call DISPLAY$CHAR(LCD$DATA, DUR$MSB3) ;             /* display 3rd tone duration     */
472  2        call DISPLAY$CHAR(LCD$DATA, DUR$LSB3) ;
473  2        call ms$DISPLAY(29) ;                               /* display units                 */
```

```
474  2          L = 23 ;                                  /* re-init. array pointer          */
475  2          call TONE$DURAT$MSG(0C9H, 26) ;           /* write 't4='                     */
476  2          call DISPLAY$CHAR(LCD$DATA, DUR$MSB4) ;   /* display 4th tone duration       */
477  2          call DISPLAY$CHAR(LCD$DATA, DUR$LSB4) ;
478  2          call as$DISPLAY(29) ;                     /* display units                   */

479  2          L = 26 ;                                  /* re-init. array pointer          */
480  2          call TONE$DURAT$MSG(99H, 29) ;            /* write 't5='                     */
481  2          call DISPLAY$CHAR(LCD$DATA, DUR$MSB5) ;   /* display 5th tone duration       */
482  2          call DISPLAY$CHAR(LCD$DATA, DUR$LSB5) ;
483  2          call as$DISPLAY(29) ;                     /* display units                   */
484  2          call QUIT$MSG ;                           /* quit message for MODE 7         */
485  1      end TONE$DURATIONS ;

486  2      TONE$DURATION$ERR : procedure ;               /* display MODE 8                  */
487  2          call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;  /* clear display                   */
488  2          call DISPLAY$ADDR(LCD$INST, 81H) ;        /* init. display address           */
489  2          L = 0 ;
490  3          do while L < 14 ;
491  3              call DISPLAY$CHAR(LCD$DATA, MESSAGE$5(L)) ;   /* write 'Duration Error'  */
492  3              L = L + 1 ;                           /* to display                      */
493  3          end ;

494  2          L = 14 ;                                  /* init. array pointer             */
495  2          call TONE$DURAT$MSG(0C0H, 17) ;           /* write 't1=' to display          */
496  2          call DISPLAY$ADDR(LCD$INST, 0C3H) ;       /* init. display address           */
497  2          call DISPLAY$CHAR(LCD$DATA, POS$NEG1) ;   /* display sign for t1             */
498  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$MSB1) ;   /* display 1st tone duration error */
499  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$LSB1) ;
500  2          call PERC$DISPLAY(31) ;                   /* display units                   */

501  2          L = 17 ;                                  /* re-init. array pointer          */
502  2          call TONE$DURAT$MSG(90H, 20) ;            /* write 't2=' to display          */
503  2          call DISPLAY$ADDR(LCD$INST, 93H) ;        /* init. display address           */
504  2          call DISPLAY$CHAR(LCD$DATA, POS$NEG2) ;   /* display sign for t2             */
505  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$MSB2) ;   /* display 2nd tone duration error */
506  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$LSB2) ;
507  2          call PERC$DISPLAY(31) ;                   /* display units                   */

508  2          L = 20 ;                                  /* re-init. array pointer          */
509  2          call TONE$DURAT$MSG(0D0H, 23) ;           /* write 't3=' to display          */
510  2          call DISPLAY$ADDR(LCD$INST, 0D3H) ;       /* init. display address           */
511  2          call DISPLAY$CHAR(LCD$DATA, POS$NEG3) ;   /* display sign for t3             */
512  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$MSB3) ;   /* display 3rd tone duration error */
513  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$LSB3) ;
514  2          call PERC$DISPLAY(31) ;                   /* display units                   */

515  2          L = 23 ;                                  /* re-init. array pointer          */
516  2          call TONE$DURAT$MSG(0C9H, 26) ;           /* write 't4=' to display          */
517  2          call DISPLAY$ADDR(LCD$INST, 0CCH) ;       /* init. display address           */
518  2          call DISPLAY$CHAR(LCD$DATA, POS$NEG4) ;   /* display sign for t4             */
519  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$MSB4) ;   /* display 4th tone duration error */
520  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$LSB4) ;
521  2          call PERC$DISPLAY(31) ;                   /* display units                   */

522  2          L = 26 ;                                  /* re-init. array pointer          */
```

```
523  2          call TONE$DURAT$MSG(99H, 29) ;          /* write 't5=' to display          */
524  2          call DISPLAY$ADDR(LCD$INST, 9CH) ;      /* init. display address           */
525  2          call DISPLAY$CHAR(LCD$DATA, POS$NEG5) ; /* display sign for t5             */
526  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$MSB5) ; /* display 5th tone duration error */
527  2          call DISPLAY$CHAR(LCD$DATA, DUR$ERR$LSB5) ;
528  2          call PERC$DISPLAY(31) ;                 /* display units                   */
529  2          call QUIT$MSG ;                         /* quit message for MODE 8         */
530  1      end TONE$DURATION$ERR ;
```

INITIALISATION

```
531  1      P0 = 00H ;                                  /* display = o/p, lower nibble spare  */
532  1      P1 = 0FFH;                                  /* ref. freq & sig i/p set to i/p     */
533  1      P2 = 00H ;                                  /* port 2 all o/p for disp addr. / data */
534  1      P3 = 0CH ;                                  /* INT0 & INT1 alt. func, 3.4 -> 3.7 keybd */

535  1      TMOD = 00010001B ;                          /* timer 0, 1 mode 2, auto reload     */
536  1      TCON = 00000000B ;                          /* timer 0, 1 Off, INT0 & INT1 level trig. */
537  1      SCON = 0 ;                                  /* no serial comms. implemented       */
538  1      IE   = 10001111B ;                          /* global, T0, T1, EX0 & EX1 int's. enabl. */
539  1      IP   = 01H ;                                /* INT0 higher priority T0, INT1, T1 lower */
540  1      PCON = 0 ;                                  /* power down, idle mode inhibited    */

541  1      CMOD   = 00000110B ;                        /* wdog off, external count i/p enab. CF=0 */
542  1      CCON   = 00H ;                              /* run off, comp. capt. flags clear   */
543  1      CCAPM2 = 00000000B ;                        /* disabled                           */
544  1      CCAPM4 = 01001000B ;                        /* disabled                           */

545  1      TH1 = 3CH ;                                 /* 50ms gating window                 */
546  1      TL1 = 0AFH ;
547  1      CURRENT$FREQ = 0H ;                         /* current freq. being  measured      */
548  1      CH = 00H ;                                  /* init. count reg for freq. meas.    */
549  1      CL = 00H ;

550  1      CONTR1$H = 00H ;                            /* init. freq1 high counter value     */
551  1      CONTR1$L = 00H ;                            /* init. freq1 low counter value      */
552  1      CONTR2$H = 00H ;
553  1      CONTR2$L = 00H ;
554  1      CONTR3$H = 00H ;
555  1      CONTR3$L = 00H ;
556  1      CONTR4$H = 00H ;
557  1      CONTR4$L = 00H ;
558  1      CONTR5$H = 00H ;
559  1      CONTR5$L = 00H ;

560  1      T_DUR1 = 0H ;                               /* init. tone durations               */
561  1      T_DUR2 = 0H ;
562  1      T_DUR3 = 0H ;
563  1      T_DUR4 = 0H ;
564  1      T_DUR5 = 0H ;

565  1      DIG$ORDER = 0 ;                             /* order of tone sequence             */
566  1      REPEAT$FREQ$1 = 0 ;                         /* repeat freq. flags                 */
567  1      REPEAT$FREQ$2 = 0 ;
568  1      REPEAT$FREQ$3 = 0 ;
```

```
569  1      REPEAT$FREQ$4 = 0 ;
570  1      ELAPSED = 0H ;                          /* tone time duration counter      */
571  1      DONE = 0 ;                              /* Initialize all flags            */
572  1      DISPLAY$RESULTS = 0 ;
573  1      KEY = 0FFH ;                            /* init. keypad buffer             */
```

Initialize Power-Up Condition Of Display

```
       /* display enabled, instr.(RS = 0), clear display, auto inc.    */
574  1      call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;

       /* display enabled, instr.(RS = 0), 2 lines, 5X8 Dots, 8 bit    */
575  1      call DISPLAY$SETUP(LCD$INST, FUNC$SET) ;

       /* display enabled, instr.(RS = 0), display on, curs & blink off */
576  1      call DISPLAY$SETUP(LCD$INST, DISP$ON$OFF) ;
```

M A I N P R O G R A M

```
577  2      do FOREVER ;

578  3          if not(DONE) and not(DISPLAY$RESULTS) then do ;   /* selcall rec. & ready for display ?  */
579  3              call DISPLAY$ADDR(LCD$INST, 80H) ;            /* init. display address               */
580  3              L = 0 ;
581  4              do while L < 16 ;
582  4                  call DISPLAY$CHAR(LCD$DATA, MESSAGE$1(L)) ;   /* display MODE 0                  */
583  4                  L = L + 1 ;
584  4              end ;
585  3              call DISPLAY$ADDR(LCD$INST, 0C1H) ;
586  4              do while L < 32 ;
587  4                  call DISPLAY$CHAR(LCD$DATA, MESSAGE$1(L)) ;
588  4                  L = L + 1 ;
589  3              end ;
590  3              call DISPLAY$ADDR(LCD$INST, 93H) ;
591  4              do while L < 45 ;
592  4                  call DISPLAY$CHAR(LCD$DATA, MESSAGE$1(L)) ;
593  4                  L = L + 1 ;
594  3              end ;
595  3          end ;

596  3          if DONE then do ;                             /* compensate durations for INT0 delays */
597  3              EX0 = 0 ;                                 /* & compute tone duration errors       */
598  3              EX1 = 0 ;                                 /* disable all ext. interrupts          */

599  3              T_DURAT1 = (T_DUR1+3)-84H ;               /* 4ms mean for no-tone period & osc. start*/
600  3              DUR$MSB1 = (T_DURAT1/0AH) or 30H ;        /* hex to ascii conversion              */
601  3              DUR$LSB1 = (T_DURAT1 mod 0AH) or 30H ;    /* for tone duration 1                  */

602  3              T_DURAT2 = T_DUR2+3 ;                     /* 3ms debounce period error comp.(INT0)*/
603  3              DUR$MSB2 = (T_DURAT2/0AH) or 30H ;        /* hex to ascii conversion              */
604  3              DUR$LSB2 = (T_DURAT2 mod 0AH) or 30H ;    /* for tone duration 2                  */

605  3              T_DURAT3 = T_DUR3+3 ;                     /* 3ms debounce period error comp.(INT0)*/
```

```
608  3          DUR$MSB3 = (T_DURAT3/0AH) or 30H ;          /* hex to ascii conversion          */
609  3          DUR$LSB3 = (T_DURAT3 mod 0AH) or 30H ;      /* for tone duration 3              */

610  3          T_DURAT4 = T_DUR4+3 ;                       /* 3ms debounce period error comp.(INT0) */
611  3          DUR$MSB4 = (T_DURAT4/0AH) or 30H ;          /* hex to ascii conversion          */
612  3          DUR$LSB4 = (T_DURAT4 mod 0AH) or 30H ;      /* for tone duration 4              */

613  3          T_DURAT5 = (T_DUR5+3)-19H ;                 /* 25ms mean for no-tone period     */
614  3          DUR$MSB5 = (T_DURAT5/0AH) or 30H ;          /* hex to ascii conversion          */
615  3          DUR$LSB5 = (T_DURAT5 mod 0AH) or 30H ;      /* for tone duration 5              */

616  3          call CALC$TDUR$ERR(T_DURAT1) ;              /* tone duration 1 error calculation */
617  3          T$DURAT1$ERR = FIN$ERR ;                    /* error for tone 1                 */
618  3          POS$NEG1 = SIGN ;                           /* pos. / neg. error                */
619  3          DUR$ERR$MSB1 = (T$DURAT1$ERR/0AH) or 30H ;  /* convert to ascii                 */
620  3          DUR$ERR$LSB1 = (T$DURAT1$ERR mod 0AH) or 30H ; /* for displaying                */

621  3          call CALC$TDUR$ERR(T_DURAT2) ;              /* tone duration 2 error calculation */
622  3          T$DURAT2$ERR = FIN$ERR ;                    /* error for tone 2                 */
623  3          POS$NEG2 = SIGN ;                           /* pos. / neg. error                */
624  3          DUR$ERR$MSB2 = (T$DURAT2$ERR/0AH) or 30H ;  /* convert to ascii                 */
625  3          DUR$ERR$LSB2 = (T$DURAT2$ERR mod 0AH) or 30H ; /* for displaying                */

626  3          call CALC$TDUR$ERR(T_DURAT3) ;              /* tone duration 3 error calculation */
627  3          T$DURAT3$ERR = FIN$ERR ;                    /* error for tone 3                 */
628  3          POS$NEG3 = SIGN ;                           /* pos. / neg. error                */
629  3          DUR$ERR$MSB3 = (T$DURAT3$ERR/0AH) or 30H ;  /* convert to ascii                 */
630  3          DUR$ERR$LSB3 = (T$DURAT3$ERR mod 0AH) or 30H ; /* for displaying                */

631  3          call CALC$TDUR$ERR(T_DURAT4) ;              /* tone duration 4 error calculation */
632  3          T$DURAT4$ERR = FIN$ERR ;                    /* error for tone 4                 */
633  3          POS$NEG4 = SIGN ;                           /* pos. / neg. error                */
634  3          DUR$ERR$MSB4 = (T$DURAT4$ERR/0AH) or 30H ;  /* convert to ascii                 */
635  3          DUR$ERR$LSB4 = (T$DURAT4$ERR mod 0AH) or 30H ; /* for displaying                */

636  3          call CALC$TDUR$ERR(T_DURAT5) ;              /* tone duration 5 error calculation */
637  3          T$DURAT5$ERR = FIN$ERR ;                    /* error for tone 5                 */
638  3          POS$NEG5 = SIGN ;                           /* pos. / neg. error                */
639  3          DUR$ERR$MSB5 = (T$DURAT5$ERR/0AH) or 30H ;  /* convert to ascii                 */
640  3          DUR$ERR$LSB5 = (T$DURAT5$ERR mod 0AH) or 30H ; /* for displaying                */

641  3          call SELECT$D$KEY$MSG ;                     /* display MODE 1                   */

642  3          DONE = 0 ;                                  /* reset received 5-tone sequence flag */
643  3          DISPLAY$RESULTS = set ;                     /* ready to display results         */
644  3          DISPLAY$OPTION$1 = set ;                    /* enable MODE 2                    */
645  3          EX1 = 1 ;                                   /* enable keypad interrupt          */
646  3        end ;

647  2        KEY = 00H ;                                   /* init. keypad buffer              */
648  3        do while DISPLAY$RESULTS ;                    /* display all parameters           */
649  4        if KEY = 0FFH then do ;                       /* D-key pressed ?                  */
651  5              do while DISPLAY$OPTION$1 ;             /* while MODE 2 active              */
652  5                 DISPLAY$OPTION$2 = set ;             /* enable MODE 3                    */
653  5                 call OPTION$DISPLAY$1 ;              /* display MODE 2                   */

654  5                 KEY = 00H ;                          /* clear keypad buffer              */
```

```
655  6                  do while (KEY <> 0FH and KEY <> 1FH and KEY <> 2FH and KEY <> 0BFH) ;
656  6                     KEY = 00H ;                                          /* clear keypad buffer */
657  6                  end ;

658  6                  if KEY = 0FH then do ;                                  /* display MODE 4 ?    */
660  6                        call DECODED$TONE ;                               /* yes !!              */
661  7                        do while KEY <> 3FH ;                             /* A-Quit ?            */
662  7                           KEY = 00H ;
663  7                        end ;
664  6                        KEY = 00H ;                                       /* clear keypad buffer */
665  6                  end ;

666  6                  if KEY = 1FH then do ;                                  /* display MODE 5 ?    */
668  6                     call FREQ$HEADING ;                                  /* 'Tone Freq.(Hz)'    */

669  6                     call FREQ$CALC(CONTR1$H, CONTR1$L) ;                 /* freq1, gated counts */
670  6                     call DISPLAY$FREQ (0C3H, F$DIG1, F$DIG2, F$DIG3, F$DIG4) ; /* disp. freq1. MODE 5 */

671  6                     call FREQ$CALC(CONTR2$H, CONTR2$L) ;                 /* freq2, gated counts */
672  6                     call DISPLAY$FREQ (93H, F$DIG1, F$DIG2, F$DIG3, F$DIG4) ;  /* disp. freq2. MODE 5 */

673  6                     call FREQ$CALC(CONTR3$H, CONTR3$L) ;                 /* freq3, gated counts */
674  6                     call DISPLAY$FREQ (0D3H, F$DIG1, F$DIG2, F$DIG3, F$DIG4) ; /* disp. freq3. MODE 5 */

675  6                     call FREQ$CALC(CONTR4$H, CONTR4$L) ;                 /* freq4, gated counts */
676  6                     call DISPLAY$FREQ (0CCH, F$DIG1, F$DIG2, F$DIG3, F$DIG4) ; /* disp. freq4. MODE 5 */

677  6                     call FREQ$CALC(CONTR5$H, CONTR5$L) ;                 /* freq5, gated counts */
678  6                     call DISPLAY$FREQ (9CH, F$DIG1, F$DIG2, F$DIG3, F$DIG4) ;  /* disp. freq5. MODE 5 */

679  7                     do while KEY <> 3FH ;                                /* A-Quit ?            */
680  7                        KEY = 00H ;
681  7                     end ;
682  6                     KEY = 00H ;                                          /* clear keypad buffer */
683  6                  end ;

684  6                  if KEY = 2FH then do ;                                  /* display MODE 6 ?    */
686  6                     call FREQ$ERR$HEADING ;                              /* 'Frequency Error'   */
687  6                     call CALC$FREQ$ERR(CONTR1$H, CONTR1$L, FREQ$DIG1) ;  /* calc. freq1. error  */
688  6                     call SHOW$FREQ$ERR(0C2H, SIGN, F$DIG1, F$DIG2) ;     /* freq1 error, MODE 6 */

689  7                     if REPEAT$FREQ$1 = set then do ;                     /* repeat tone ?       */
691  7                        call CALC$FREQ$ERR(CONTR2$H, CONTR2$L, 10D) ;     /* error using 2600hz  */
692  7                        end ;
693  7                     else do ;
694  7                        call CALC$FREQ$ERR(CONTR2$H, CONTR2$L, FREQ$DIG2) ;  /* calc. freq2 error   */
695  7                        end ;
696  6                     call SHOW$FREQ$ERR(92H, SIGN, F$DIG1, F$DIG2) ;      /* freq2 error, MODE 6 */

697  7                     if REPEAT$FREQ$2 = set then do ;                     /* repeat tone ?       */
699  7                        call CALC$FREQ$ERR(CONTR3$H, CONTR3$L, 10D) ;     /* error using 2600hz  */
700  7                        end ;
701  7                     else do ;
702  7                        call CALC$FREQ$ERR(CONTR3$H, CONTR3$L, FREQ$DIG3) ;  /* calc. freq3 error   */
703  7                        end ;
704  6                     call SHOW$FREQ$ERR(0D2H, SIGN, F$DIG1, F$DIG2) ;     /* freq3 error, MODE 6 */
```

```
705  7                  if REPEAT$FREQ$3 = set then do ;                    /* repeat tone        */
707  7                     call CALC$FREQ$ERR(CONTR4$H, CONTR4$L, 10D) ;    /* error using 2600hz */
708  7                     end ;
709  7                  else do ;
710  7                     call CALC$FREQ$ERR(CONTR4$H, CONTR4$L, FREQ$DIG4) ;  /* calc. freq4 error */
711  7                     end ;
712  6                  call SHOW$FREQ$ERR(0CBH, SIGN, F$DIG1, F$DIG2) ;    /* freq4 error, MODE 6 */

713  7                  if REPEAT$FREQ$4 = set then do ;                    /* repeat tone        */
715  7                     call CALC$FREQ$ERR(CONTR5$H, CONTR5$L, 10D) ;    /* error using 2600hz */
716  7                     end ;
717  7                  else do ;
718  7                     call CALC$FREQ$ERR(CONTR5$H, CONTR5$L, FREQ$DIG5) ;  /* calc. freq5 error */
719  7                     end ;
720  6                  call SHOW$FREQ$ERR(9BH, SIGN, F$DIG1, F$DIG2) ;     /* freq5 error, MODE 6 */

721  7                  do while KEY <> 3FH ;                               /* A-Quit ?           */
722  7                     KEY = 00H ;
723  7                  end ;
724  6                  KEY = 00H ;                                         /* clear keypad buffer */
725  6               end ;

726  6               if KEY = 0BFH then do ;                                /* MODE 3             */
728  7                  do while DISPLAY$OPTION$2 ;                         /* while MODE 3 active */
729  7                     call OPTION$DISPLAY$2 ;                          /* display MODE 3     */

730  7                     KEY = 00H ;                                      /* clear keypad buffer */
731  8                     do while (KEY <> 4FH and KEY <> 5FH and KEY <> 0BFH and KEY <> 6FH) ;
732  8                        KEY = 00H ;                                   /* clear keypad buffer */
733  8                     end ;

734  8                     if KEY = 4FH then do ;                           /* display MODE 7 ?   */
736  8                        call TONE$DURATIONS ;                         /* yes !!             */
737  9                        do while KEY <> 3FH ;                         /* A-Quit ?           */
738  9                           KEY = 00H ;
739  9                        end ;
740  8                        KEY = 00H ;                                   /* clear keypad buffer */
741  8                     end ;

742  8                     if KEY = 5FH then do ;                           /* display MODE 8 ?   */
744  8                        call TONE$DURATION$ERR ;                      /* yes !!             */
745  9                        do while KEY <> 3FH ;                         /* A-Quit             */
746  9                           KEY = 00H ;
747  9                        end ;
748  8                        KEY = 00H ;                                   /* clear keypad buffer */
749  8                     end ;

750  8                     if KEY = 0BFH then do ;                          /* return to MODE 2 ? */
752  8                        DISPLAY$OPTION$2 = 0 ;                        /* disable MODE 3     */
753  8                        KEY = 00H ;                                   /* clear keypad buffer */
754  8                     end ;

755  8                     if KEY = 6FH then do ;                           /* abandon  readings ? */
757  8                        CONTR1$H = 0H ;                               /* clear all variables */
758  8                        CONTR1$L = 0H ;
759  8                        CONTR2$H = 0H ;
760  8                        CONTR2$L = 0H ;
```

```
 761   8                          CONTR3$H = 0H ;
 762   8                          CONTR3$L = 0H ;
 763   8                          CONTR4$H = 0H ;
 764   8                          CONTR4$L = 0H ;
 765   8                          CONTR5$H = 0H ;
 766   8                          CONTR5$L = 0H ;
 767   8                          T_DUR1 = 00H ;
 768   8                          T_DUR2 = 00H ;
 769   8                          T_DUR3 = 00H ;
 770   8                          T_DUR4 = 00H ;
 771   8                          T_DUR5 = 00H ;
 772   8                          DISPLAY$OPTION$2 = 0 ;                  /* reset all flags     */
 773   8                          DISPLAY$OPTION$1 = 0 ;
 774   8                          DISPLAY$RESULTS = 0 ;
 775   8                          call DISPLAY$SETUP(LCD$INST, CLR$DISP) ;  /* clear display       */
 776   8                       end ;
 777   7                    end ;
 778   6                 end ;
 779   5              end ;
 780   4           end ;
 781   3        end ;
 782   2     EX0 = 1 ;                                                    /* real time int = on  */
 783   2     EX1 = 0 ;                                                    /* disable keypad int. */
 784   2     end ;
 785   1     end IVEI ;
```

WARNINGS:
   3 IS THE HIGHEST USED INTERRUPT

MODULE INFORMATION: (STATIC+OVERLAYABLE)
   CODE SIZE = 1128H 4392D
   CONSTANT SIZE = 0151H 337D
   DIRECT VARIABLE SIZE = 36H+10H 54D+ 16D
   INDIRECT VARIABLE SIZE = 16H+00H 22D+ 0D
   BIT SIZE = 08H+00H 8D+ 0D
   BIT-ADDRESSABLE SIZE = 00H+00H 0D+ 0D
   AUXILIARY VARIABLE SIZE = 0000H 0D
   MAXIMUM STACK SIZE = 0031H 49D
   REGISTER-BANK(S) USED: 0 1 2 3
   1084 LINES READ
   0 PROGRAM ERROR(S)
END OF PL/M-51 COMPILATION

APPENDIX E

SIMULATED DATA CHANGE SIGNAL FLOW CHARTS

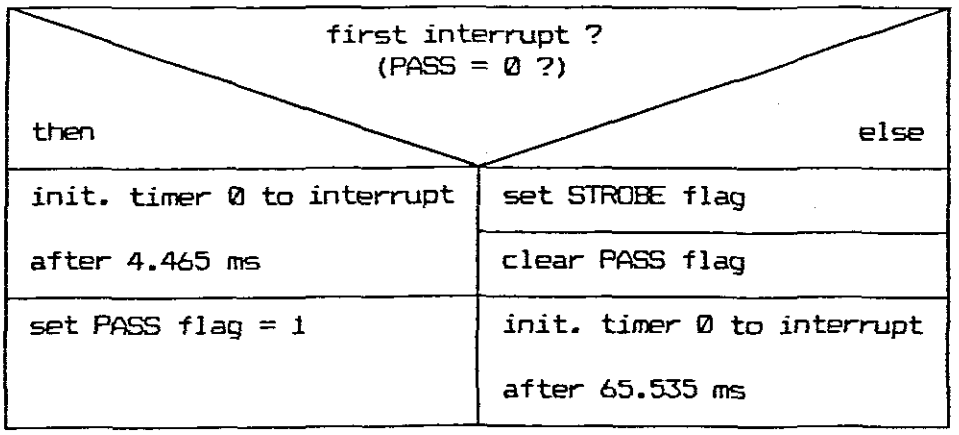
APPENDIX E

11. SIMULATED DATA CHANGE SIGNAL FLOW CHARTS

Flow diagrams for simulator software generating the data change signals on a 8031 simulator board.

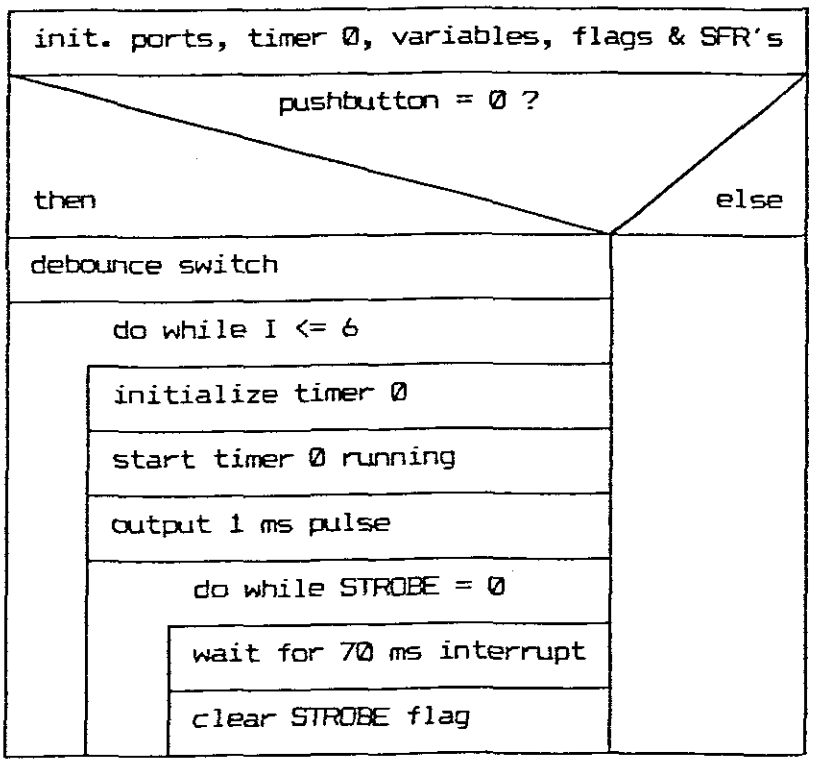
11.1 Timer interrupt to produce 70ms time delay.

TIMER0



11.2 Main program flow.

MAIN



APPENDIX F

SIMULATED DATA CHANGE SIGNAL SOFTWARE LISTING

APPENDIX F

12. SIMULATED DATA CHANGE SIGNAL SOFTWARE LISTING

The following pages represents the software listing of the simulated 1ms data change pulse generated every 70ms.

```
DOS 3.20 (038-N) PL/M-51 V1.2
COMPILER INVOKED BY:  C:\PLM\PLM51.EXE \PLM\PROJECTS\PULSESIM.PLM DEBUG ROM(LARGE)


  1   1     PULSESIM : do ;

            +-----------------------------------------------+
            | S O F T W A R E  L I S T I N G  F O R  S I M U L A T E D  1ms |
            | D A T A  C H A N G E  P U L S E  E V E R Y  70 ms  (6 OFF) |
            +-----------------------------------------------+

  2   1     declare eq literally 'literally' ;

            $nolist
            $include(REG51.DCL)
            $list

                        +-------------------------+
                        | P R O G R A M  V A R I A B L E S |
                        +-------------------------+

  6   1     declare D_CHNG bit at (90H)       reg ;
  7   1     declare PUSH$BUTTON bit at (0B0H) reg ;
  8   1     declare (PASS, STROBE)            bit main ;
  9   1     declare (I)                       byte main ;

 10   1     declare   OFF       eq '0',  ON        eq '1',
                    CLEAR     eq '0',  SET       eq '1',

                    FOREVER   eq 'while 1' ;

                        +-------------------------+
                        | R E A L - T I M E  I N T E R R U P T |
                        +-------------------------+

 11   2     TIMER0: procedure interrupt 1 using 2 ;          /* timer interrupt to produce 70ms time delay */
 12   3       if PASS = 0 then do ;
 14   3           TH0 = 0EEH ;                               /* 4.465ms + 65.535ms = 70 ms                 */
 15   3           TL0 = 8EH ;
 16   3           PASS = 1 ;                                 /* control flag                               */
 17   3       end;
 18   3       else do ;
 19   3           STROBE = 1 ;
 20   3           PASS = 0 ;
 21   3           TH0 = 0H ;                                 /* clear timer 0                              */
 22   3           TL0 = 0H ;
 23   3       end ;
 24   1     end TIMER0 ;

                        +-------------------------+
                        | I N I T I A L I S A T I O N |
                        +-------------------------+

 25   1     P0 = 00H ;
 26   1     P1 = 0EFH ;                                      /* port 1.0 = 0, o/p                          */
 27   1     P2 = 00H ;
 28   1     P3 = 0DH ;                                       /* port 3.0 = 1, i/p                          */
```

```
  29   1      TMOD = 00010001B ;                    /* timer 0 mode 2 auto reload          */
  30   1      TCON = 00000000B ;                    /* timer 0 initially off               */
  31   1      SCON = 0 ;                            /* no serial comms. implemented        */
  32   1      IE   = 10000010B ;                    /* global, T0 interrupts enabled       */
  33   1      IP   = 00H ;                          /* normal priority of interrupts       */
  34   1      PCON = 0 ;                            /* power down, idle mode inhibited     */

  35   1      TH0 = 00H ;                           /* initialize timer £0                 */
  36   1      TL0 = 00H ;

  37   1      PASS = 0 ;                            /* timer flag                          */
  38   1      STROBE = 0 ;                          /* flag to initiate data change pulse  */
  39   1      I = 0 ;                               /* data change pulse counter           */
```

MAIN PROGRAM

```
  40   2       do FOREVER ;
  41   2          if PUSH$BUTTON = 0 then ;
  43   2              call time(250) ;              /* debounce switch                     */
  44   3              if PUSH$BUTTON = 0 then do ;  /* initiate start of sequence          */
  46   3                  I = 0 ;                   /* when pushbutton is pressed          */
  47   4                  do while I <= 6 ;         /* gen. five 70ms periods              */
  48   4                      TH0 = 0H ;            /* ensure timer 0 is reset             */
  49   4                      TL0 = 0H ;
  50   4                      TR0 = 1 ;             /* start timer 0                       */
  51   4                      D_CHNG = 1 ;          /* 1ms wide pulse                      */
  52   4                      call time(10) ;
  53   4                      D_CHNG = 0 ;
  54   5                      do while STROBE = 0 ; /* wait for timer int.                 */
  55   5                      end ;
  56   4                      STROBE = 0 ;          /* reset flag for interrupt            */
  57   4                      I = I + 1 ;           /* prepair for next data change        */
  58   4                  end ;
  59   3              end ;
  60   2          end ;
  61   1      end PULSESIM ;
```

WARNINGS:

    1 IS THE HIGHEST USED INTERRUPT

```
MODULE INFORMATION:              (STATIC+OVERLAYABLE)
    CODE SIZE                    = 007AH      122D
    CONSTANT SIZE                = 0000H        0D
    DIRECT VARIABLE SIZE         =   01H+00H    1D+  0D
    INDIRECT VARIABLE SIZE       =   00H+00H    0D+  0D
    BIT SIZE                     =   02H+00H    2D+  0D
    BIT-ADDRESSABLE SIZE         =   00H+00H    0D+  0D
    AUXILIARY VARIABLE SIZE      = 0000H        0D
    MAXIMUM STACK SIZE           = 0011H       17D
    REGISTER-BANK(S) USED:         0 2
    188 LINES READ
    0 PROGRAM ERROR(S)
END OF PL/M-51 COMPILATION
```

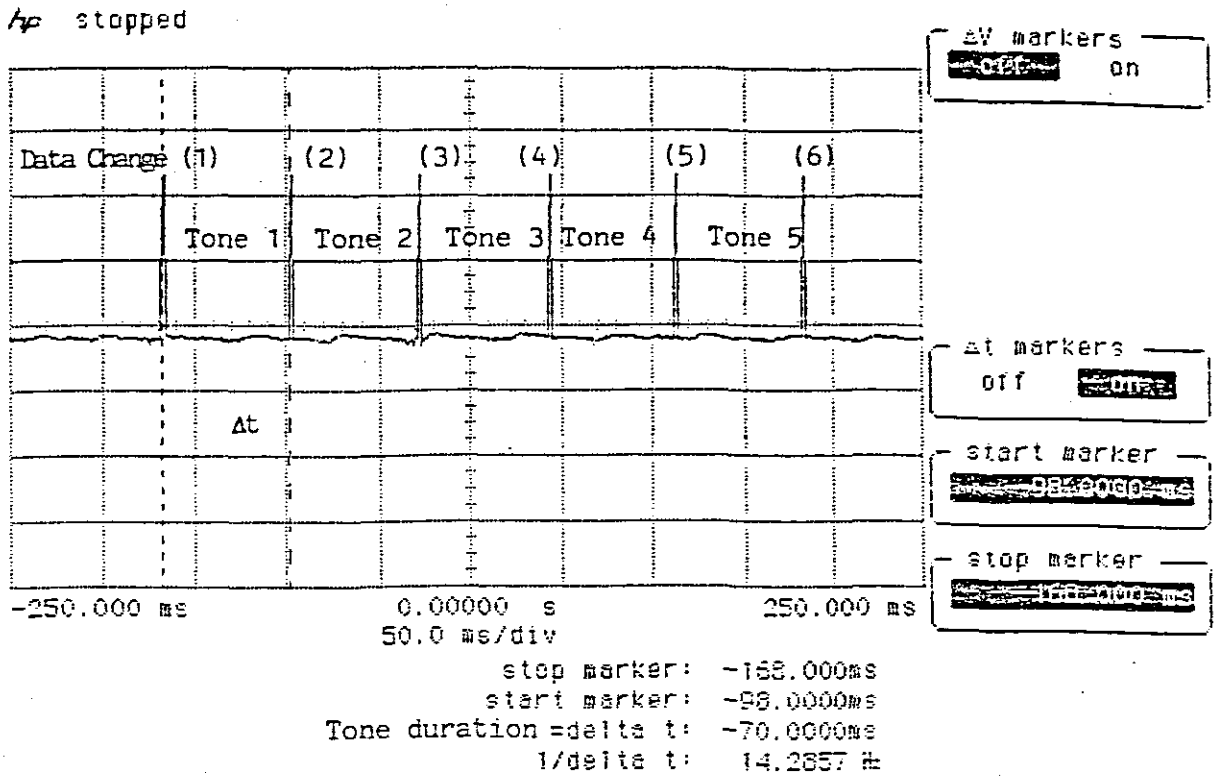
APPENDIX G

SIMULATED DATA CHANGE SIGNAL REPRESENTATION

APPENDIX G

13. SIMULATED DATA CHANGE SIGNAL REPRESENTATION

The following page represents the data change signals generated by a simulator board using a 8031 micro-controller.



Data Change signals generated by simulator board using a 8031 microcontroller

APPENDIX H

MODE 0 TO MODE 8 DISPLAY FORMAT

APPENDIX H

14. MODE 0 TO MODE 8 DISPLAY FORMAT

This section represents the various display modes which are selected and displayed on the 16-character by 4-line Dot Matrix Display of the 5-Tone ZVEI Encoder Analyser.

14.1 Display MODE 0, standby message.

R	e	a	d	y		T	o		R	e	c	e	i	v	e
	Z	V	E	I		S	i	g	n	a	l	.	.	.	
	K	e	y		T	r	a	n	s	m	i	t	t	e	r

14.2 Display MODE 1, ready to display analysed results.

S	i	g	n	a	l		R	e	c	e	i	v	e	d	
S	e	l	e	c	t		D	-	k	e	y		f	o	r
D	i	s	p	l	a	y		O	p	t	i	o	n	s	

14.3 Display MODE 2, enables display of MODE 4 to MODE 6.

S	e	l	e	c	t	:	-			C	-	M	o	r	e
1	-	T	o	n	e		D	i	g	i	t	s			
2	-	F	r	e	q	.									
3	-	F	r	e	q	.		E	r	r	o	r			

14.4 Display MODE 3, enables display of MODE 7, 8 and MODE 0.

S	e	l	e	c	t	:	-		C	-	P	r	e	v	.
4	-	T	o	n	e		D	u	r	a	t	i	o	n	s
5	-	D	u	r	a	t	i	o	n		E	r	r	o	r
A	-	Q	u	i	t		R	e	a	d	i	n	g	s	

14.5 Display MODE 4, showing typical decoded digits.

	D	e	c	o	d	e	d		D	i	g	i	t	s	
				1		2		3		4		5			
										A	-	Q	u	i	t

14.6 Display MODE 5, showing typical tone frequencies.

	T	o	n	e		F	r	e	q	.	(	H	z	)	
F	1	=	1	0	6	0			F	4	=	1	4	0	0
F	2	=	1	1	6	0			F	5	=	1	5	3	0
F	3	=	1	2	7	0				A	-	Q	u	i	t

14.7 Display **MODE 6**, showing typical frequency errors.

F	r	e	q	u	e	n	c	y		E	r	r	o	r	
F	1	+	0	,	1	%			F	4	-	1	,	2	%
F	2		0	,	0	%			F	5	+	0	,	1	%
F	3	-	0	,	5	%				A	-	Q	u	i	t

14.8 Display **MODE 7**, showing typical tone durations.

	T	o	n	e		D	u	r	a	t	i	o	n	s	
t	1	=	7	0	m	s			t	4	=	6	9	m	s
t	2	=	6	8	m	s			t	5	=	7	0	m	s
t	3	=	7	1	m	s				A	-	Q	u	i	t

14.9 Display **MODE 8**, showing typical tone duration errors.

	D	u	r	a	t	i	o	n		E	r	r	o	r	
t	1	=	+	0	3	%			t	4	=	-	0	4	%
t	2	=	-	0	1	%			t	5	=	+	0	2	%
t	3	=		0	0	%				A	-	Q	u	i	t

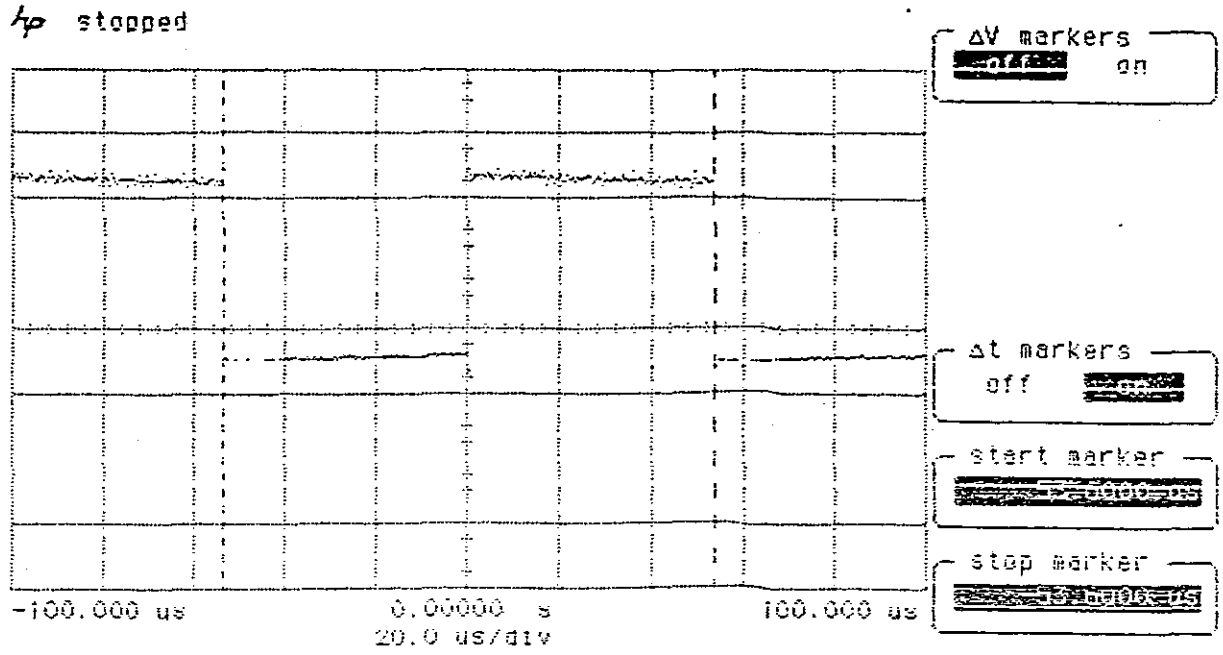
APPENDIX I

WAVEFORMS OF TONE DIGIT FREQUENCIES

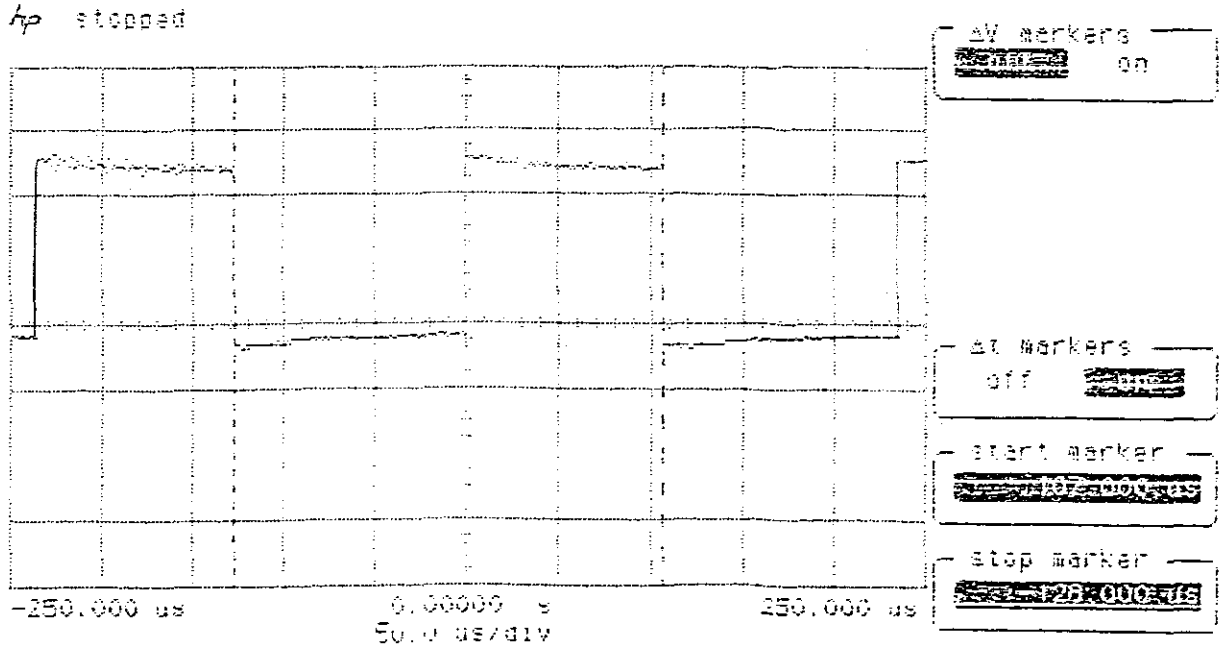
APPENDIX I

15. WAVEFORMS OF TONE DIGIT FREQUENCIES

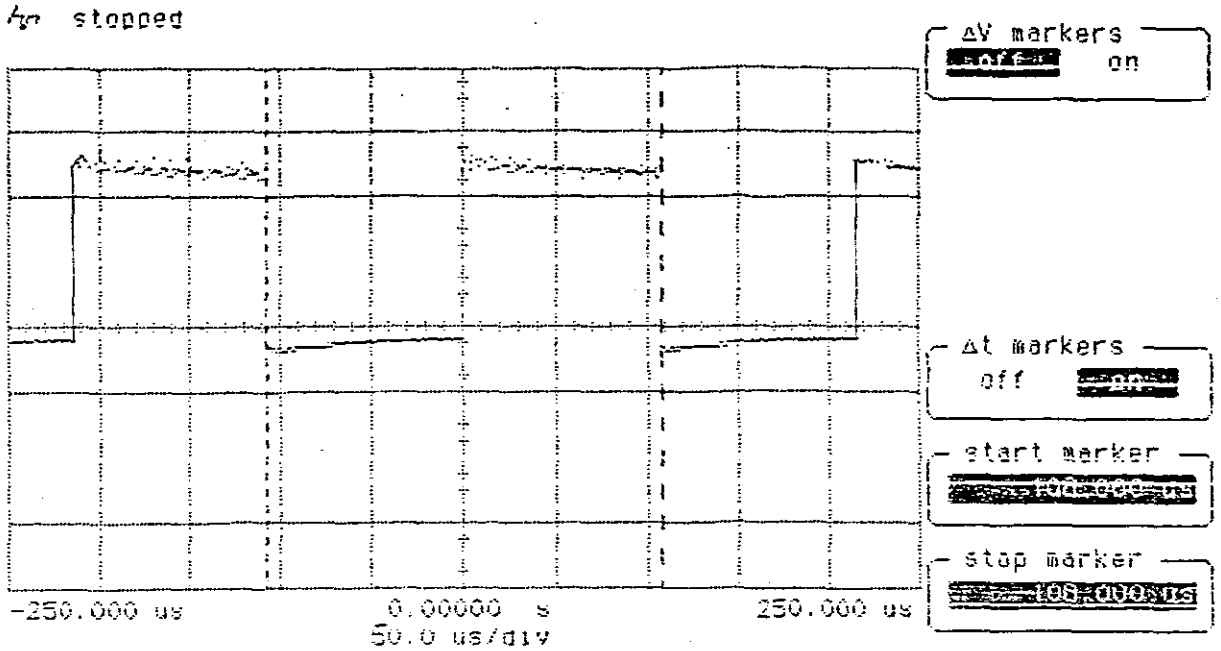
The following pages represents the waveforms of the Tone Digit Frequencies for digits 0 to 9 and the repeat tone. The waveforms were measured at the output of a FX102LG.



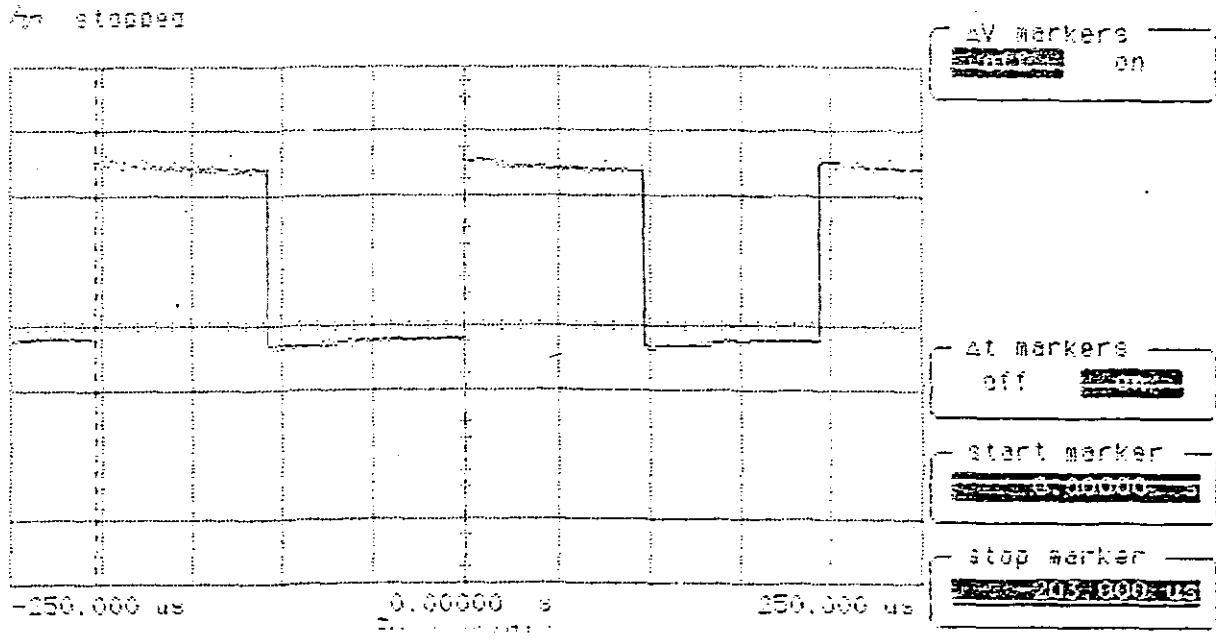
Waveform representing a Tone Digit of "0"



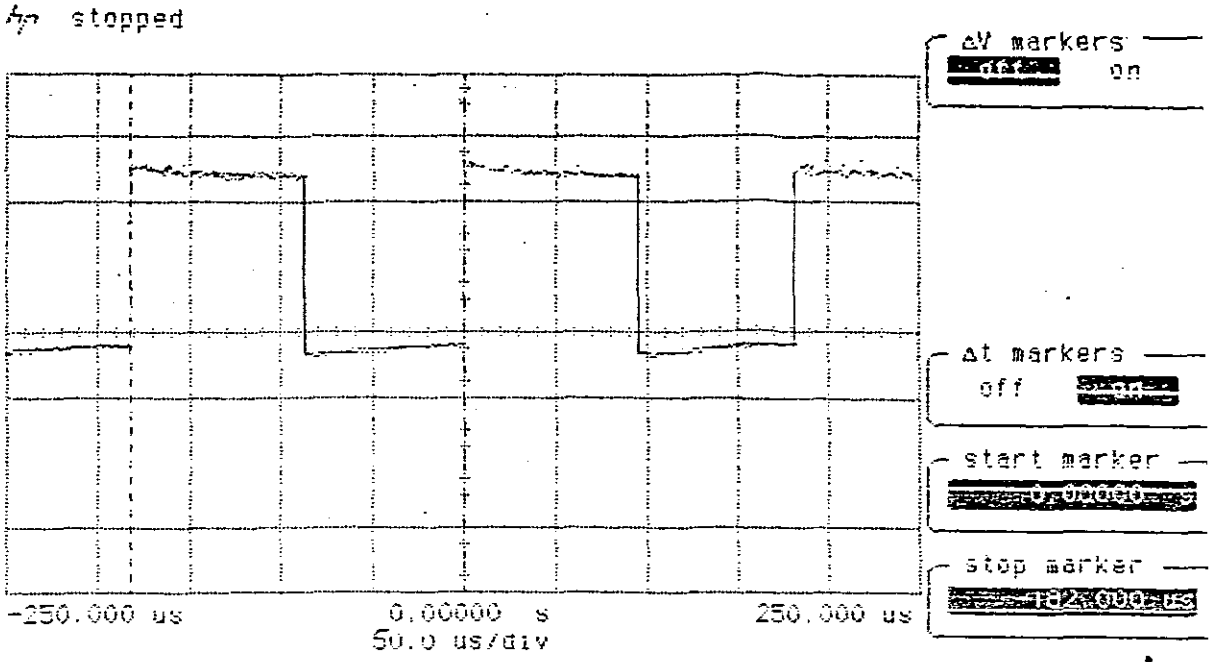
Waveform representing a Tone Digit of "1"



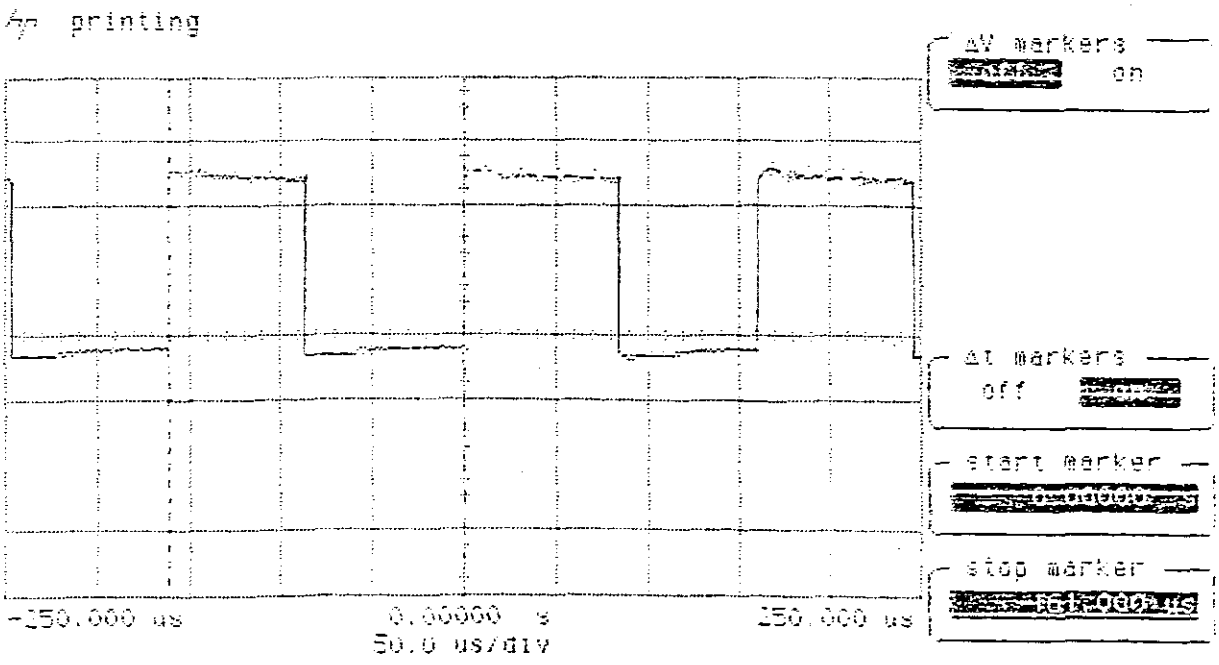
Waveform representing a Tone Digit of "2"



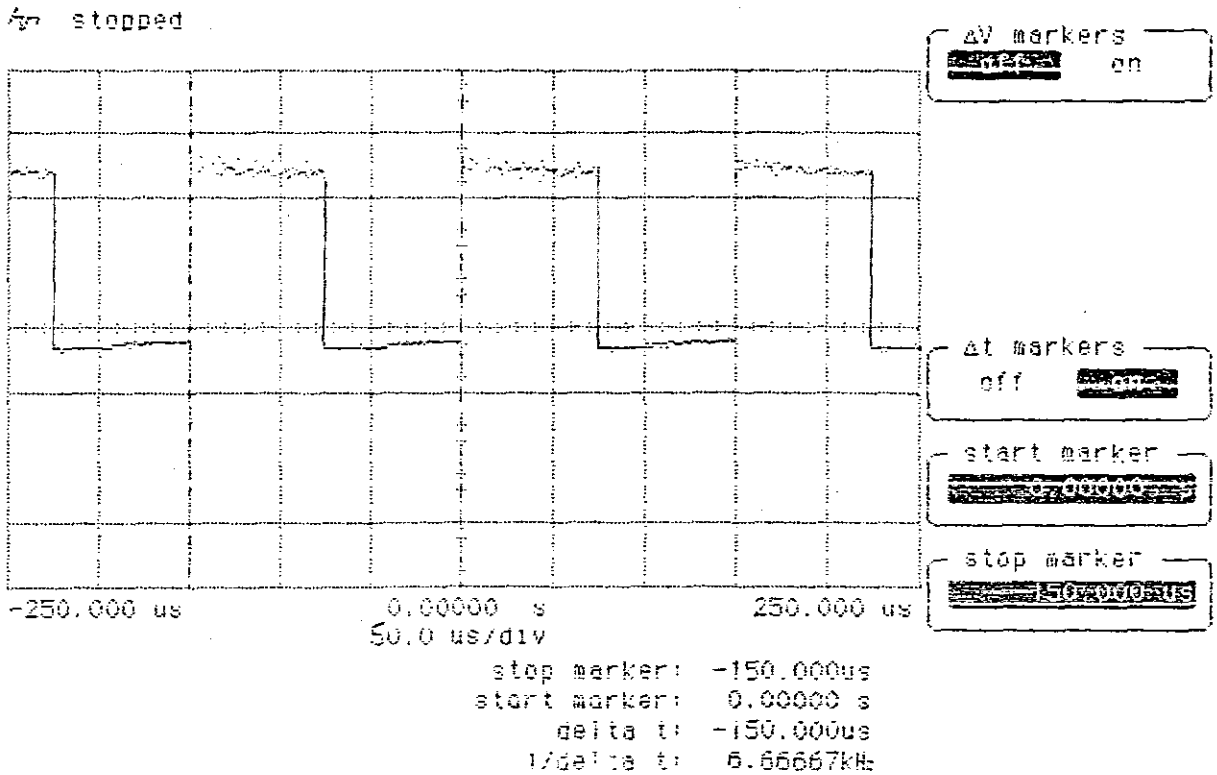
Waveform representing a Tone Digit of "3"



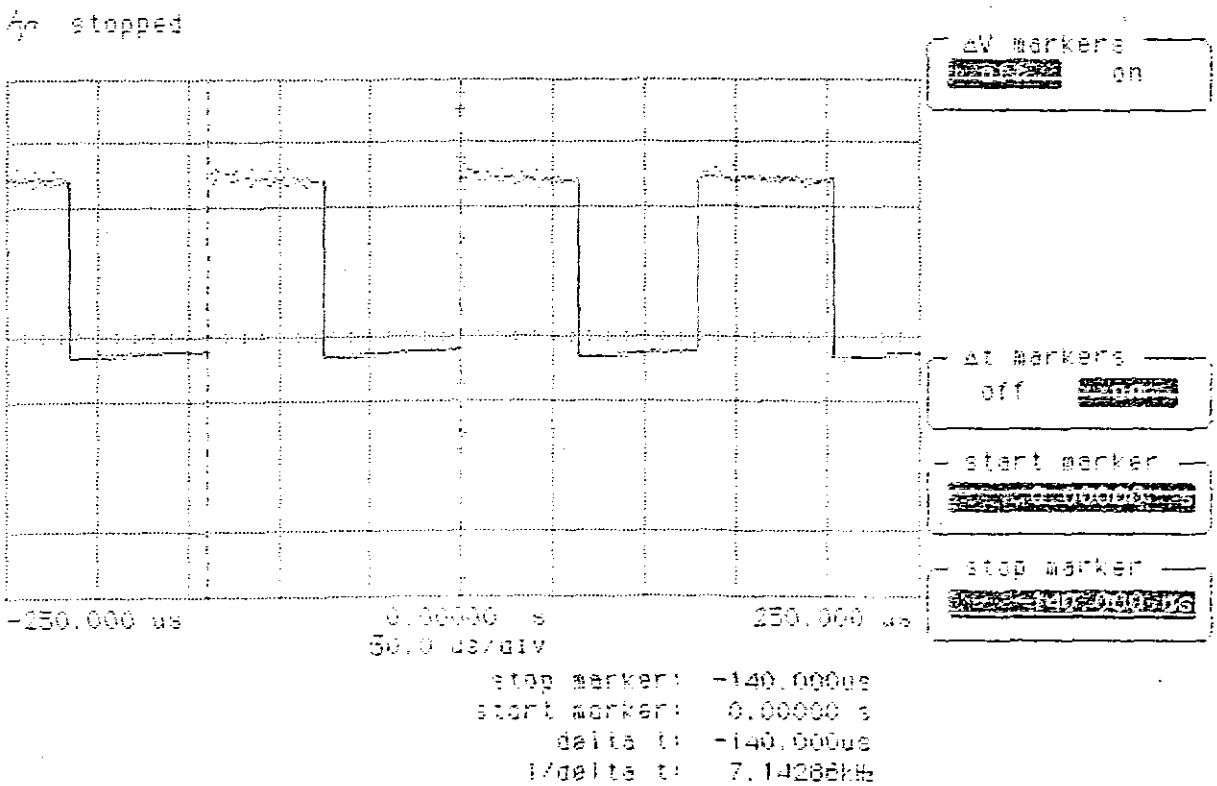
Waveform representing a Tone Digit of "4"



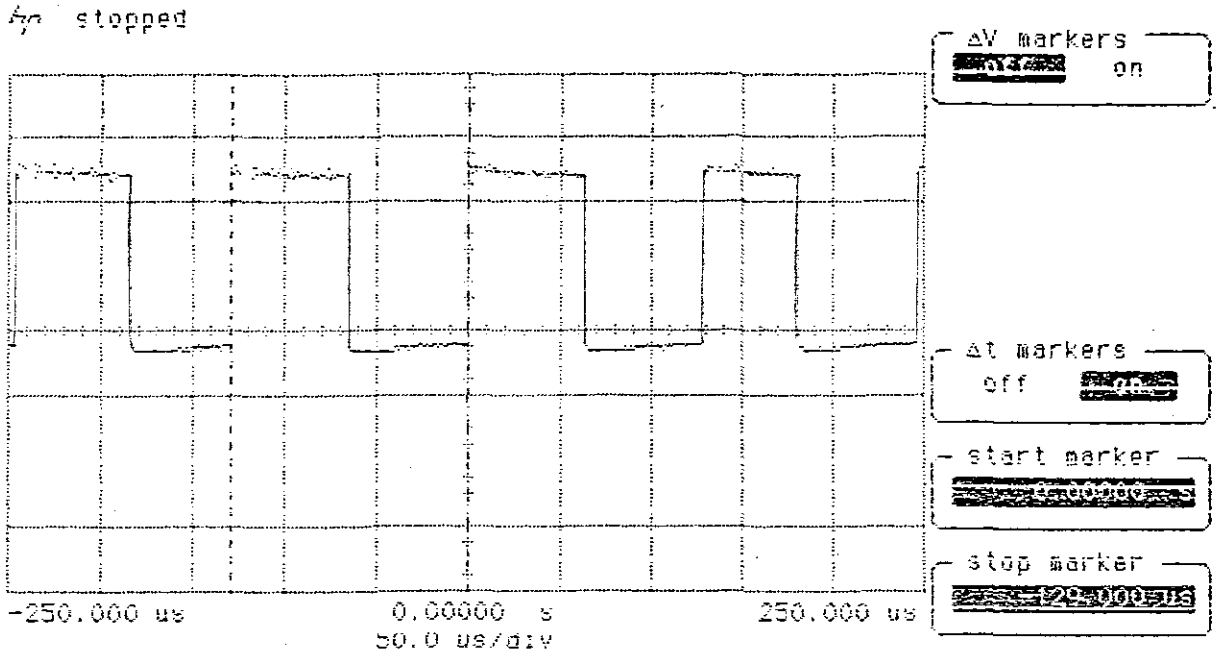
Waveform representing a Tone Digit of "5"



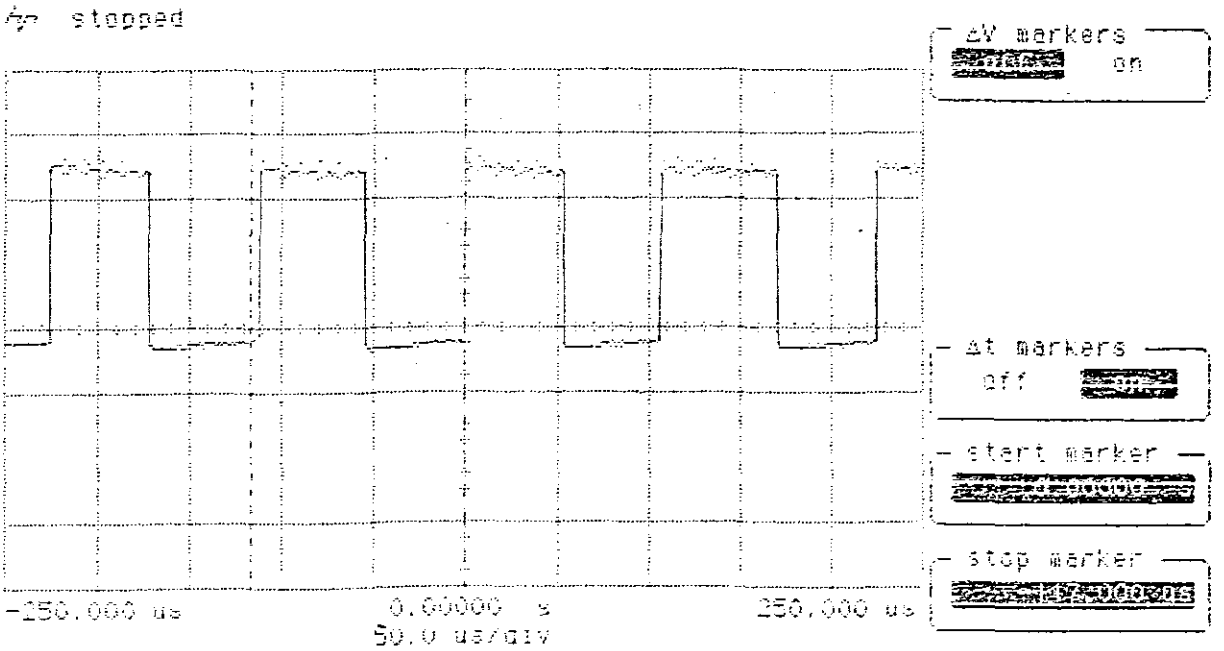
Waveform representing a Tone Digit of "6"



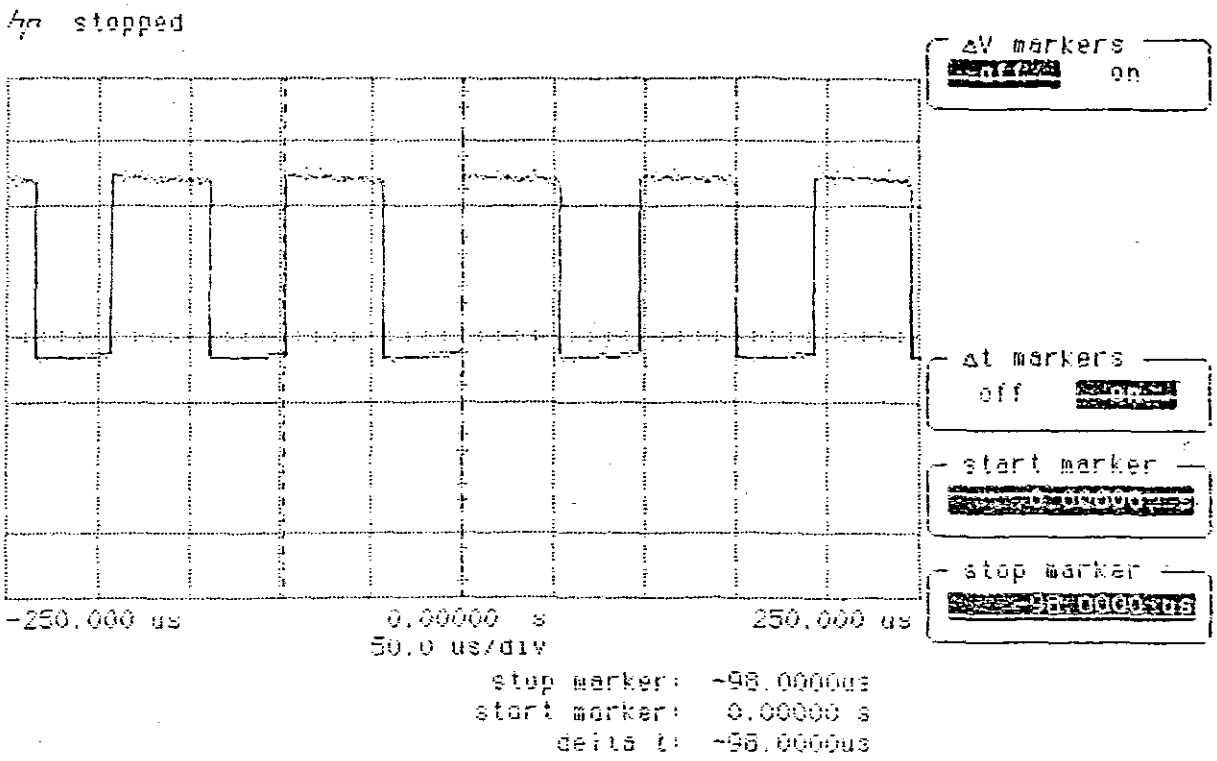
Waveform representing a Tone Digit of "7"



Waveform representing a Tone Digit of "8"



Waveform representing a Tone Digit of "9"



Waveform representing a Tone Digit "E" (Repeat Tone)

ANNEXURE 1

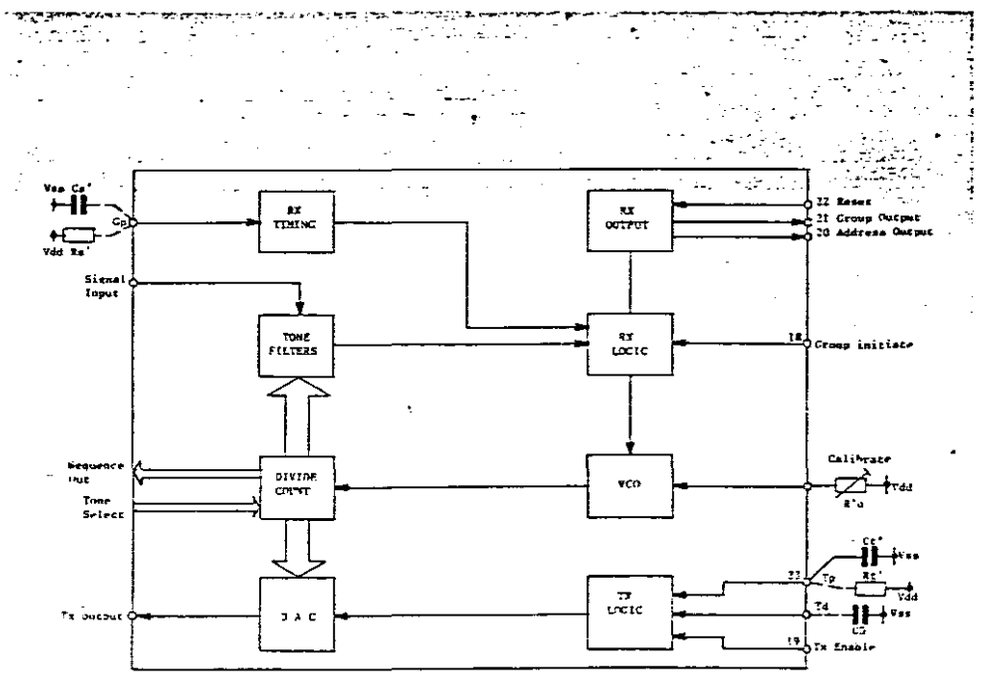
DATA SHEETS ON SELECTIVE-CALL TONE ENCODERS / DECODERS

CONSUMER MICROCIRCUITS LIMITED

PRODUCT INFORMATION

NCTIONAL SCHEMATIC

PUBLICATION D/4071/2/2



FX 4070A
FX 5070A
FX 4071A
FX 5071A

HYBRID INTEGRATED 5 TONE ENCODER/ DECODERS

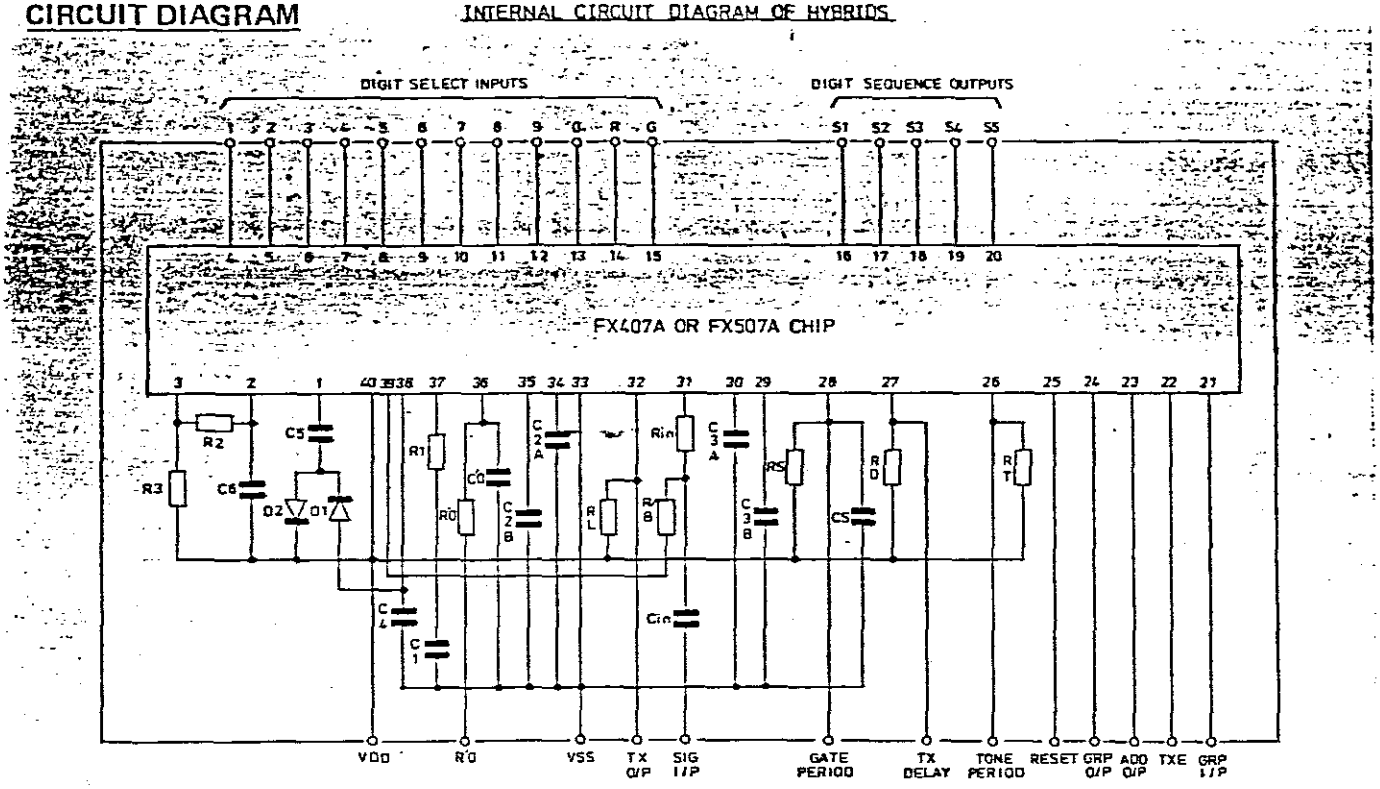
EATURES

- FULLY FUNCTIONAL 5-TONE SELECTIVE CALLING SYSTEM
- THICK FILM HYBRID CONSTRUCTION
- USES CML FX407A/FX507A CHIP
- FIXED OR VARIABLE SYSTEM TIMING
- ENCODER, DECODER AND TRANSPONDER
- CHOICE OF PACKAGE STYLE

he FX 4070A, FX 5070A, FX 4071A and FX 5071A are integrated circuits fabricated using thick film ybrid techniques and include a monolithic LSI semiconductor chip. The circuit comprises all necessary omponents to provide a 5-tone selective calling encoder/decoder which complies with the requirements of he CCIR standard (FX 4070A or FX 4071A) or the ZVEI standard (FX 5070A or FX 5071A). In all spects the operation of these devices is identical to the recommended circuit used in the application of he CML FX 407A/FX 507A integrated circuit. The hybrid circuits include passive components necessary or correct system operation, the trimmer potentiometer required to set frequency calibration is externally onnected and external components may also be used to modify internally set system timing parameters.

CIRCUIT DIAGRAM

INTERNAL CIRCUIT DIAGRAM OF HYBRIDS



HYBRID COMPONENT LIST (This information can be read in conjunction with Production Information on FX 407A)

RESISTORS			CAPACITORS			
CIRCUIT DIAGRAM REF.	VALUE	TOLERANCE	CIRCUIT DIAGRAM REF.	VALUE	TC	TOLERANCE
R1	100K	5%	C1	0.022μf	X7R	10%
R2	100K	5%	C2A=C2B	470pf	X7R	10%
R3	22K	5%	C3A=C3B	0.01μf	X7R	10%
R'O	100K	5%	C4	0.22μf	X7R	10%
R'in	330K	10%	C5	0.01μf	X7R	20%
RD	680K	10%	C6	68pf	NPO	5%
RS(FX4070A/FX4071A)	1.22M	2%	C'O	0.01μf	NPO	5%
RS(FX5070A/FX5071A)	857K	2%	C'in	0.01μf	X7R	20%
RT	355K	2%	Cs	0.22μf	X7R	20%
RL	600Ω	10%				
R'b	330K	10%				

CALIBRATION PROCEDURE

Conditions: VDD = 12.5V, TA = 20° ± 5°, no input signals.

1) Programme code 6XXXX(FX4070A/FX4071A) or 5XXXX(FX5070A/FX5071A), where 'X' is any convenient digit.
2) Apply VDD. Wait several seconds, then connect a shorting link between pins 23 and VSS.
3) Press the TX Enable button and read the frequency at the TX output. The signal is a continuous tone.
4) Adjust the R'o calibration potentiometer until the frequency is exactly 1541Hz (FX4070A/FX4071A) or 1530Hz (FX5070A/FX5071A).
5) Remove the shorting link between pins 23 and VSS. Calibration is now completed for all channels for both RX and TX.

EXTERNAL COMPONENT CONNECTIONS

1. The address and group outputs are open drain MOS transistors. This allows the outputs to be linked to give a wire – OR if required. Load resistors (10KΩ nom.) are required for these outputs from pin 20 and pin 21 to -Ve supply.

2. A 22KΩ trimmer potentiometer with a temperature coefficient <100ppm must be connected from R'O to -Ve supply (pin 32). This is used to calibrate frequency as described above.

3. The Output Reset input has an on-chip 1MΩ (nominal) resistor to VDD and a 3KΩ M.O.S. transistor to VSS which is disabled when the outputs are activated. With this input held at VSS the outputs will turn on at power-up. To reset the outputs and to ensure they are off at power-up the pin must be momentarily taken to VDD. An automatic time-out commencing from when the outputs are activated is provided with a resistor in parallel with a capacitor between pins 22 and 32. The interval T'on ≐0.65 RC seconds, and should be longer than one tone period.

4. A capacitor (CT) must be provided between pin 23 and the positive supply (VSS) to set the transmitted tone period.

   The period is given by:

   $$Tp = 0.355K\ CT \text{ seconds.}$$

   where CT is measured in μF

   K = 0.6 ± 0.03 typ. (a constant of the monolithic chip)

   For CCIR tone period = 100 mS (FX4070A or FX4071A) then CT = 0.47μF
   For ZVEI tone period = 70 mS (FX5070A or FX5071A) then CT = 0.33μF

5. Optionally:

   i) A capacitor (CD) can be connected between the TX Delay pin and the positive supply (VSS) to provide a delay prior to the start of transmission.

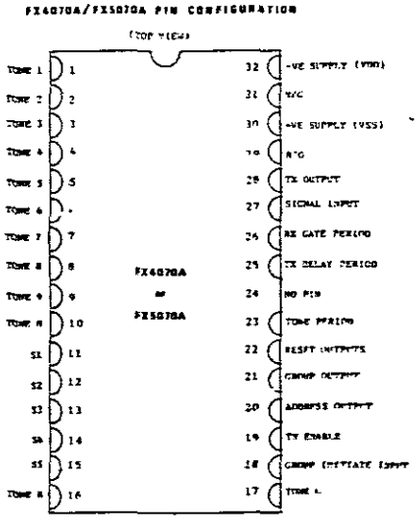
   TX delay period = 0.68K CD seconds *

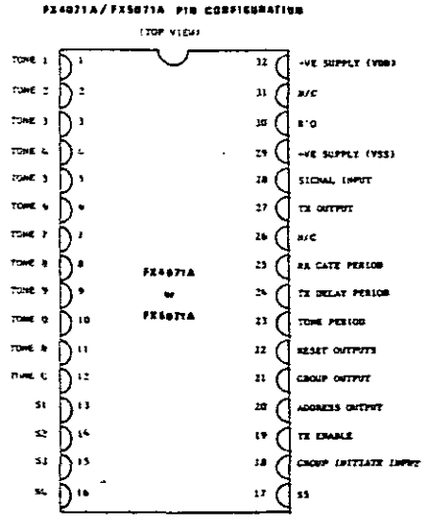
   ii) The RX gate period (internally set to 1.75 Tp) can be varied by external components connected in parallel to RS and CS.

   RX gate period = K RS' CS' seconds *

   * where K = 0.65 ± 0.033 typ. RS' and CS' are resultant values of internal and external components, CD and CS' are measured in μF
   RS' is measured in Megohms

PACKAGE DETAILS

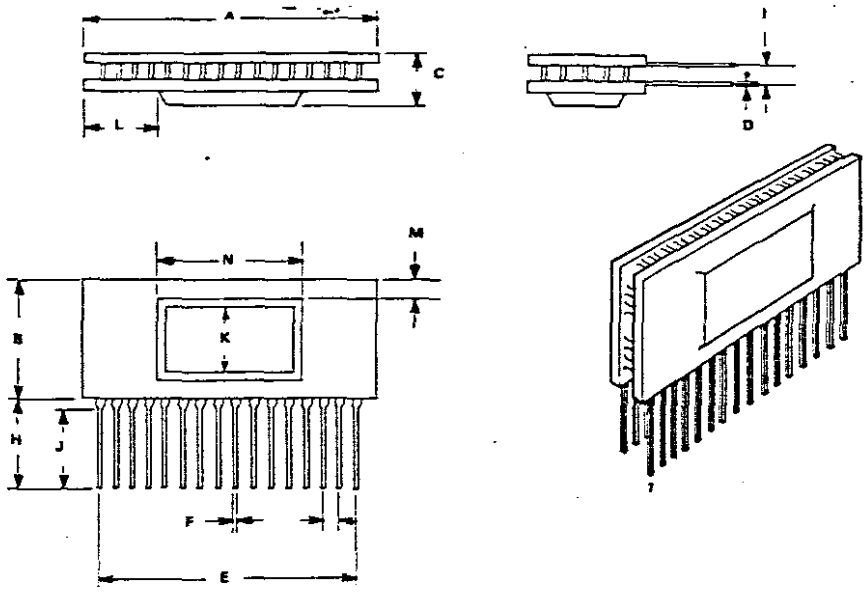




FX 4070A and FX 5070A PACKAGE

WORKING TEMPERATURE RANGE: -20°C to +60°C
STORAGE TEMPERATURE RANGE: -40°C to +85°C

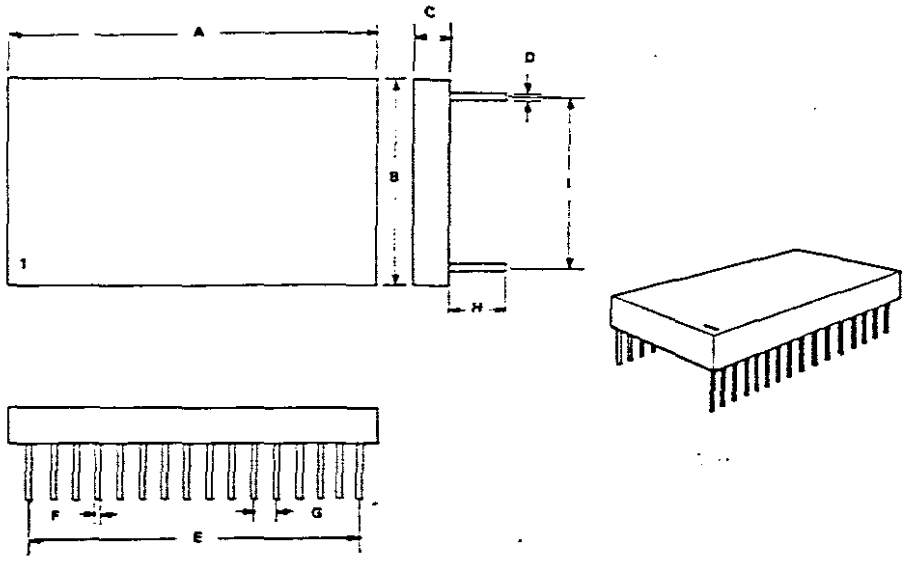
	Inches	m.m.
A	1.675	42
B	0.675	17
C	0.36	8.9
D	0.01	0.25
E	1.5	38.1
F	0.02	0.51
G	0.1	2.5
H	0.425	10.8
I	0.1	2.54
J	0.4	10.0
K	0.4	10.0
L	0.4	10.0
M	0.085	2.16
N	0.9	22.9



FX 4071A and FX 5071A PACKAGE

WORKING TEMPERATURE RANGE: -20°C to +60°C
STORAGE TEMPERATURE RANGE: -40°C to +85°C

	Inches	m.m.
A	1.77	45
B	1.18	30
C	0.2	5.1
D	0.01	0.25
E	1.5	38.1
F	0.02	0.51
G	0.1	2.54
H	0.325	8.25
I	0.9	22.9



ELECTRICAL CHARACTERISTICS

	MIN.	TYP.	MAX.	NOTE
Supply voltage	-10V		-15V	
Supply current		15mA		
Operating Temperature	-20°C		+60°C	

ELECTRICAL CHARACTERISTICS AT VDD = 12 volts & T'amb = 20°C unless otherwise specified.

Input voltage logic '1'	-8.0V			Note 1.
Input voltage logic '0'			-1.5V	
Output voltage logic '1'	-10.0V			Note 2.
Output voltage logic '0'			-1.0V	

OPERATING FREQUENCIES (fo of bandpass filters and Tx O/P) see Calibration Procedure.

Digit 1	1121Hz	1057.5Hz		
2	1200.5Hz	1163Hz		
3	1278Hz	1269Hz		
4	1357Hz	1402Hz		
5	1444Hz	1530Hz		
6 FX4070A	1541Hz	1665.5Hz	FX5070A	
7 FX4071A	1638Hz	1828Hz	FX5071A	
8	1747Hz	2001Hz		
9	1856.3Hz	2203Hz		
0	1983Hz	2403Hz		
R	2113Hz	2601Hz		
G	2401Hz	2796Hz		
Frequency stability Δf/°C			0.015%	Temperature range -20° +60°C
Frequency stability Δf/Vsupply			0.015%	Supply voltage range -10 to -15V

DECODER OPERATION

Sensitivity		50mV		r.m.s
Max. Signal Handling	700mV			r.m.s.
100% decode BW	3% 4%			FX4070A & FX4071A FX5070A & FX5071A referred to fo
0% decode BW			6% 9%	FX4070A & FX4071A FX5070A & FX5071A referred to fo
Input impedance		100KΩ		

ENCODER OPERATION

Output voltage	1V			peak to peak output
Tone period	90mS 63mS	100mS 70mS	110mS 77mS	FX4070A & FX4071A FX5070A & FX5071A with recommended value of CT
Output impedance		600Ω		

Note 1. Pins 1-10, 16-19, 22 FX4070A & FX5070A
Pins 1-12, 18, 19, 22 FX4071A & FX5071A

2. RLOAD=10KΩ Pins 11-15, 20, 21 FX4070A & FX5070A
RLOAD=10KΩ Pins 13-17, 20, 21 FX4071A & FX5071A

CODE PROGRAMMING

Programming is achieved by links between the sequence outputs and the digit select inputs. The repeat code (R) is substituted for consecutive identical digits, e.g. 39999 is coded 39R9R. Should the fifth digit be common to any other digit it must be linked via a diode as shown.

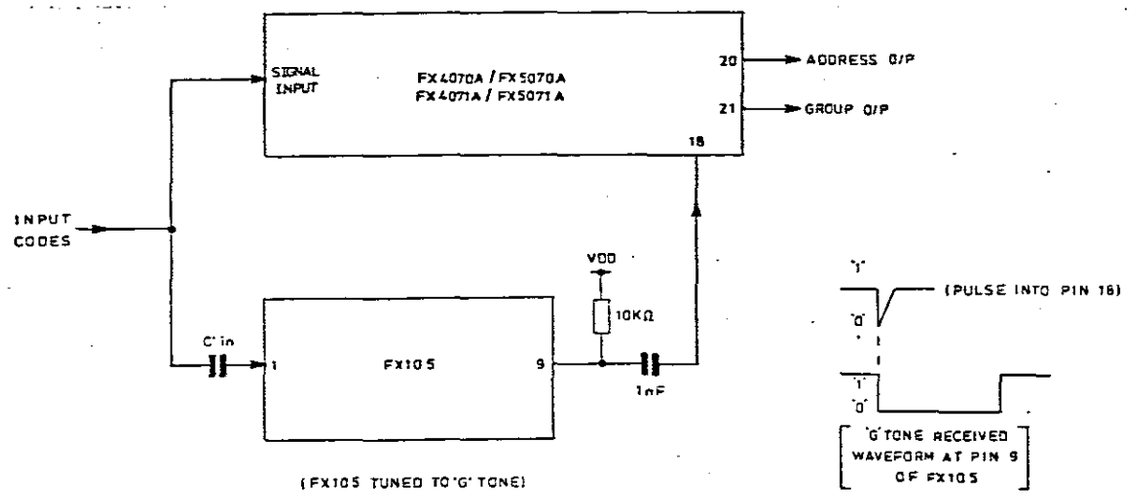
Shortened codes are achieved by linking S5 directly to the last digit.

EXAMPLES OF CODING

SEQUENCE OUTPUTS	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
PROGRAMMING LINKS	↓	↓	↓	↓	↓	↓	●	↓	┌	↓	↓	↓	●	↓	↓	↓	↓	●	┐	
DIGIT INPUTS	2	0	9	6	4	2	1	R		0	1	5	6	9		3	6	5		
PROGRAMMED ADDRESS	2	0	9	6	4	2	1	1	1	0	1	5	6	9	6	3	6	5		

CIRCUIT ARRANGEMENT

GROUP CALLING



When an input code contains a 'G' tone, the FX105 switches ON and the 1→0 edge is coupled to the Group Initiate input (pin 18) of the hybrid. This causes the hybrid to change from the Address code programmed, to an internal Group programme for the remaining digits of the input code. If the Group code is correctly completed, the Group output of the hybrid switches ON. The Group Initiate is effective only if the correct first two digits of the Address code have been received. The Group Initiate input is disabled during transmit mode, a Group call is transmitted by encoding the 'G' tone via the Digit Select inputs.

EXAMPLES OF GROUP CALLING

INPUT CODE	ACTIVATES RECEIVERS CODED:	GROUPS OF:	O/P SWITCHED
25784	25784 only	1	ADDRESS
2578G	25780 to 25789	10	GROUP
257GR	25700 to 25799	100	GROUP
25GRG	25000 to 25999	1000	GROUP

CONSUMER MICROCIRCUITS LIMITED
WHEALONROAD INDUSTRIAL ESTATE EAST
WITHAM ESSEX CM8 3TD ENGLAND
Telephone: 0376 513833

AEN Printed Limited

PRODUCT INFORMATION

PRODUCT INFORMATION

PUBLICATION D/2030/1
MARCH 1983

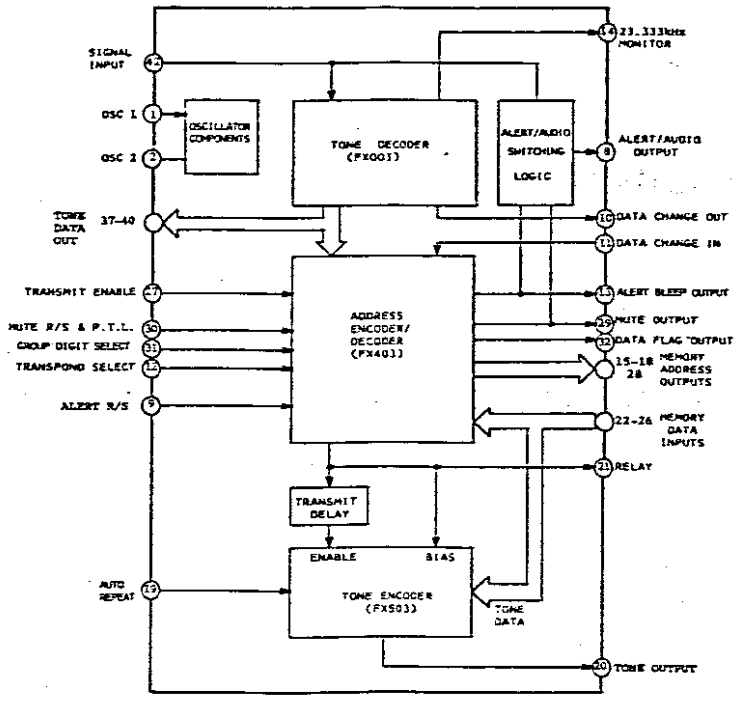


FIGURE 1 INTERNAL BLOCK DIAGRAM

FX2030C
FX2030Z
LOW POWER SELECTIVE CALLING ENCODER/ DECODER HYBRID

FEATURES

- LOW POWER CONSUMPTION
- SMALL PHYSICAL SIZE
- COMPLETE ENCODER/DECODER/TRANSPONDER
- CHOICE OF CCIR OR ZVEI TONE-SETS
- INCLUDES ALERT BLEEP GENERATOR
- INCLUDES GROUP CALL DECODING
- AUDIO SWITCHING
- FEW EXTERNAL COMPONENTS
- SIMPLE INTERFACING
- DATA DECODING

DESCRIPTION

The FX2030 is a complete selective calling encoder/ decoder for 5-tone sequential systems. It offers the advantages of small size, low power consumption, simple interfacing and high performance.

The thick film hybrid combines LSI CMOS chips with discrete resistors, capacitors and diodes to minimise the number of external components required. The FX2030C is based on the CCIR signalling frequencies and the FX2030Z is tuned to the ZVEI tone-set.

DECODER

A typical sequence of operation is illustrated in Figure 2. The FX2030 decodes a correct selective call and automatically transponds an acknowledgement in reply. The hybrid will then switch the Mute output high and generate an alert call to warn of an incoming message. Finally the incoming speech signals are switched through to the output.

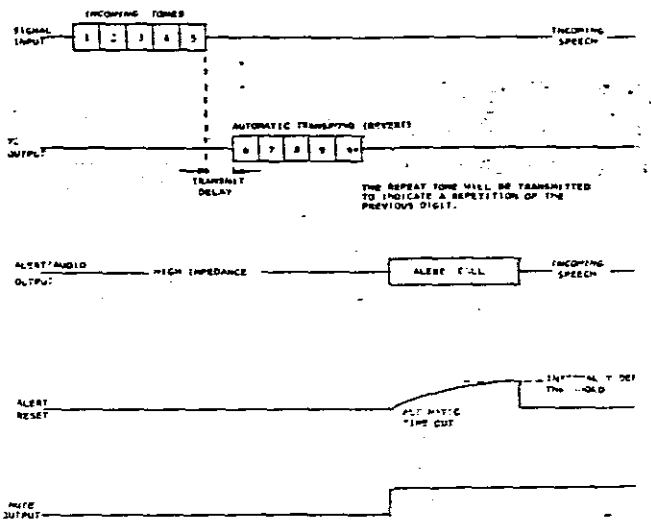


FIGURE 2 TYPICAL TRANSPONDER SEQUENCE

ENCODER

The encoder section of the FX2030 is enabled either in response to a transmit instruction (Tx) or for automatic transpond (Tp) as illustrated in Figure 2.

EXTERNAL COMPONENTS

To provide a selcall encoder/decoder function the FX2030 requires a few external components as shown in Figure 3.

560kHz Oscillator.

The receiver tone channels, encoder output frequencies and alert patterns together with system timing functions are derived from the 560kHz reference. A low cost ceramic resonator can be directly driven by the circuit together with a trimmer capacitor to adjust the frequency as shown. Pin wiring should be kept short.

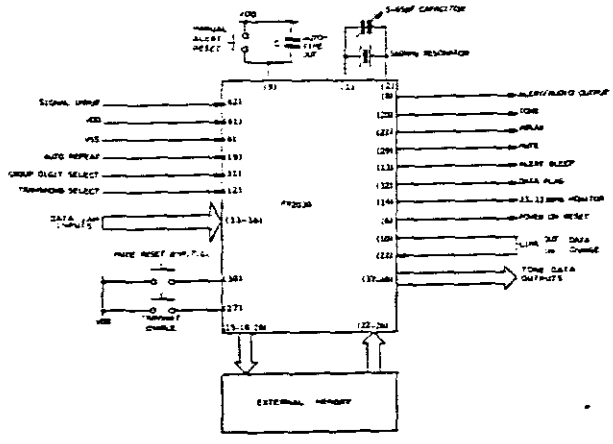


FIGURE 3 EXTERNAL COMPONENTS

The 560kHz can be measured by monitoring Pin 14 and adjusting the logic level square wave to 23.333kHz. Monitoring Pin 14 avoids directly loading the oscillator with a frequency counter. All necessary calibration is now complete.
For some applications the small frequency drifts associated with temperature and ageing effects may be unacceptable. In this case a crystal derived clock can be injected at Pin 2. Under no circumstances should power be applied to the device with no clock present. This would cause the dynamic logic to malfunction and could damage the device.

Alert Reset.

Adding a capacitor to the Alert Reset input (between pins 9 and 41) provides an automatic time out of alert signals. A N/O switch to VDD can be used for manual reset.

PIN DESIGNATIONS

Pin

1. Osc. 1
2. Osc. 2
3. Not connected
4. VSS (negative supply)
5. Not connected
6. Power on reset
7. Not connected
8. Alert/Audio output
9. Alert reset input
10. Data change output
11. Data change input
12. Transpond select input
13. Alert bleep logic output
14. 23.333kHz monitor output
15. AO memory address output
16. A1 memory address output
17. A2 memory address output
18. A4 memory address output
19. Auto repeat input
20. Tone output
21. Relay output
22. DpO(2) memory data input
23. DpO(1) memory data input
24. Dp1 memory data input
25. Dp2 memory data input
26. Dp3 memory data input
27. Transmit enable input
28. A3 memory address output
29. Mute output
30. Mute reset & P.T.L. input
31. Group digit select input
32. Data flag output
33. DjO data jam input
34. Dj1 data jam input
35. Dj2 data jam input
36. Dj3 data jam input
37. Q3 tone data output
38. Q2 tone data output
39. Q1 tone data output
40. Q0 tone data output
41. VDD (positive supply)
42. Signal input

TECHNICAL SPECIFICATION

ABSOLUTE MAXIMUM RATINGS

Exceeding the maximum rating can result in device damage. Operation of the device outside the operating limits is not implied.

SUPPLY VOLTAGE ..... −0.3V to 7.0V
INPUT VOLTAGE AT ANY PIN (ref VSS = 0V) ..... −0.3V to (VDD + 0.3)V
OUTPUT SINK/SOURCE CURRENT (TOTAL) 20mA
OPERATING TEMPERATURE RANGE ..... −20°C to +60°C
STORAGE TEMPERATURE RANGE ..... −40°C to +85°C
MAXIMUM DEVICE DISSIPATION 100mW

OPERATING LIMITS

Unless otherwise stated VDD = 5V, $T_A = 25°C$, $\phi = 560kHz$, Gaussian White Noise (band limited to 6kHz), $\Delta fo = o$, Input Tone Period { 100ms per tone CCIR / 70ms for ZVEI

CHARACTERISTICS		MIN	TYP	MAX	UNIT
STATIC CHARACTERISTICS					
Supply Volts		4.5	5.0	5.5	V
Supply Current (In receive mode)			1.5		mA
Supply Current (In transmit mode)			3.5		mA
Input Impedance Pins 2, 12, 19, 27, 31, 33-36			1		MΩ
Input Impedance Pins 9, 30			2.2		MΩ
Input logic '1' } All logic inputs		3.5			V
Input logic '0' } All logic inputs				1.5	V
Logic '1' O/P Isource = 0.1mA } Pins 10, 14-18, 29,		4.0			V
Logic '0' O/P Isink = 0.1mA } 32, 37-40				1.0	V
Logic '1' O/P Isource = 0.3mA } Pin 28		4.0			V
Logic '0' O/P Isink = 0.8μA } Pin 28				1.0	V
Logic '1' O/P Isource = 0.05mA } Pin 13, 21		4.0			V
Logic '0' O/P Isink = 0.05mA } Pin 13, 21				1.0	V
Output source current Pin 6				300	nA
DYNAMIC CHARACTERISTICS					
DECODER					
Decode Input Sensitivity		40			mVrms
Decode Bandwidth P>0.995 any tone	FX2030C	±1			
at Signal/Noise ratio >3dB	FX2030Z	±2			%
Not Decode Bandwidth	FX2030C	±3			
(P<0.03 any tone)	FX2030Z	±4.5			%
Signal/Noise Ratio for P>0.97 (any 5 tone call sequence)		0			dB
ENCODER					
Tone output emf pin 20		120	160	185	mVrms
Tone output source impedance pin 20			1		kΩ
Tone duration	FX2030C	98	100	102	ms
	FX2030Z	68	70	72	ms
Tone encode accuracy (Δf)	FX2030C			±4	Hz
	FX2030Z			±0.3	%
Transmitter lead time (Tx Delay)		150	210	290	ms
AUDIO PATH					
Input Impedance (speech path switch open circuit) pin 42		1.75			MΩ
Speech path switch impedance (i.e. 'on') between pins 42 and 8			1		kΩ
Speech path switch isolation (i.e. 'off') between pins 42 to 8			50		dB
ALERT					
Alert call frequency			2.121		kHz
Alert output level (into 1kΩ) at pin 8			275		mVpk-pk
Alert output level (into 10k) at pin 13			2.0		Vpk-pk

	MIN	TYP	MAX	UNIT
Alert time out using external capacitor C in μF on pin 9	1.57C	2.1C	2.63C	s
Leakage current of capacitor C			0.7	μA
Recommended range of external capacitor C	0.33		4.7	μF
Internal pull-up resistor pin 9	2.09	2.2	2.31	MΩ
Switchover time from Alert to incoming speech at pin 8		0.6		s
TIMINGS				
Osc. 2 input pulse width (pin 2) (using external clock)	750	893	1000	ns
Power On Reset (pin 6) input pulse width	2			ms
Alert Reset (pin 9) input pulse width	10			μs
Data Change input (pin 11) pulse width	100			μs
Transmit Enable (pin 27) input pulse width	100			μs
Transmit Enable pulse rise time			10	μs
Mute Reset & P.T.L. (pin 30) input pulse width	90			μs

SIGNALLING FREQUENCIES (in Hertz)

Tone	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
FX2030C	1981	1124	1197	1275	1358	1446	1540	1640	1747	1860	2400	930	2247	991	2110
FX2030Z	2400	1060	1160	1270	1400	1530	1670	1830	2000	2200	2800	810	970	886	2600

Table 2 shows the response of the FX2030 under various input conditions.

TABLE 2

TRANSMITTED TONES (Note 1)	TRANSPOND SELECT (Note 2)	TRANSPOND OUTPUT	ALERT OUTPUT (Note 3)	ALERT AUTO TIME OUT	AUDIO SWITCH (Note 4)	COMMENTS
12345	1	Yes	Address call	Yes	On	Address call. Transpond selected.
12345	0	No	Address call	No	Not applicable	Address call. No auto time-out. Transpond not selected.
1234A or AEAEA	X	No	Group call	Yes	On	Group call or all-call. Transpond automatically inhibited.
12345C	1	No	Address call	Yes	On	Address call. Reverses Transpond selection.
12345C	0	Yes	Address call	Yes	On	Address call. Reverses Transpond selection.
12345B6789	1	Yes	Second address/data	No	Not applicable	Addressed data (6789). No auto time out.
1234AB	X	No	Second address/data	No	Not applicable	Group call using second address. Transpond automatically inhibited.
12345D	1	Yes	None	Not applicable	Off	'silent interrogate'/mute and reset. Transpond if programmed.
AEAEAD	X	No	None	Not applicable	Off	Mute and reset all receivers. Transpond automatically inhibited.
12345CB	1	No	Second address/data	No	Not applicable	Second address call. Reverses transpond selection.
12345CD	1	No	None	Not applicable	Off	Individual mute & reset. Reverses transpond selection.
123456	X	No	None	Not applicable	Off	Incorrect address with too many digits.

Notes: 1) The address code is 12345, A is group tone, E is repeat tone, B, C and D are address suffix tones.

2) Indicates input logic state, X = don't care.

3) Refer to Fig. 5 for alert patterns and timings.

4) Defines condition of audio switch after transpond and alert sequences are completed. 'Not applicable' arises where alert does not automatically time out and must be manually reset.

DESIGN INFORMATION

Pin Descriptions

OSC. 1 - Output of a CMOS inverter. Can be used in conjunction with the Osc. 2 input to form the active element of a ceramic resonator circuit. Passive components are included on the hybrid to complete the oscillator circuit.

OSC. 2 - Input of a CMOS inverter. Alternatively a logic level 560kHz square wave can be injected at this pin with Osc. 1 left open circuit.

VSS - Negative supply input.

POWER ON RESET - provides a positive going pulse when power is applied to the hybrid. The pulse can be used to automatically initialise external circuits but must not be excessively loaded. The power supply must rise to 90% of its final value within 2ms to ensure adequate function. Alternatively an external logic '1' pulse from a low impedance source can be applied after the power supply has stabilised.

ALERT/AUDIO OUTPUT - linear output pin. The logic level alert patterns are output at this pin via an impedance of approximately 15k ohms. The transmission gate between the Signal input and Alert/Audio output is disabled when alert patterns are being generated.

When the hybrid is unmuted the analogue signals at the Signal input are available at the Alert/Audio output.

ALERT RESET INPUT - Includes internal pull-up resistor. The input has a schmitt trigger characteristic to permit the use of slow rising voltages for automatic time out applications. A discharge transistor normally holds the pin close to the negative supply and is disabled for auto time out. The alert pattern is reset when the input voltage exceeds the schmitt trigger upper limit (typically 0.66 VDD volts).

A switch connecting to the positive supply can be used for manual reset.

DATA CHANGE OUTPUT - CMOS logic output. A logic 1 pulse is output each time the tone decoder detects a change of tone. Normally connected to:

DATA CHANGE INPUT - logic input. A logic 1 input pulse causes the address decoder to compare the tone code with the data held in the external address PROM.

TRANSPOND SELECT INPUT - logic input with pull-up resistor. A logic 1 or open circuit will cause the FX2030 to automatically transpond a reply after a selective call is decoded, and a logic 0 will inhibit this feature. Address suffix tone C inverts this selection and group calls inhibit transponding.

ALERT BLEEP OUTPUT - CMOS logic output. The alert patterns are available at this pin from a source impedance of typically 5k ohms. In the quiescent state the output is at VSS.

23.333 kHz MONITOR OUTPUT - CMOS logic output. This pin outputs a square wave frequency of 23.333kHz when the input clock is 560kHz. Most applications require a calibration tolerance of ±0.1%.

A0, A1, A2, A4 MEMORY ADDRESS OUTPUTS - CMOS logic outputs used to address the external memory.

AUTO REPEAT INPUT - logic input with pull-up resistor.

An open circuit or logic 1 enables the encoder auto repeat circuitry. This detects when the repeat frequency is required in an encode sequence.

Repeater access applications may require a continuous tone of two or more tone period durations. This can be encoded by disabling the auto repeat feature with a logic 0. The memory address outputs increment as normal after each tone period.

TONE OUTPUT - linear output. The output is held at VSS in the quiescent state.

The a.c. tone output signals are generated about a mean d.c. level of typically 0.25 volts.

RELAY OUTPUT - logic output. This pin outputs a logic 1 during a transmit sequence via a typical impedance of 15k ohms.

DPO (2), DPO (1), DP1, DP2, DP3 MEMORY DATA INPUTS - logic inputs. The 4 bit selcall address codes are input to the hybrid by these pins. The least significant bits are fed separately to the tone encoder DPO (2) and address encoder chip DPO (1). This allows the encoder to be programmed to the repeat frequency (code E) if required without causing shut-down of the system.

TRANSMIT ENABLE INPUT - applying a logic 1 level pulse at this pin acts as a transmit enable request. The internal components couple a pulse through to the address encoder.

A3 MEMORY ADDRESS OUTPUT - gives a CMOS logic 1 output level when a transmit enable request is accepted. The quiescent state is a logic 0 established by a 1M ohm pull-down resistor.

MUTE OUTPUT - CMOS logic output. The mute output is set to logic 1 at the start of an alert output sequence following receipt of a selcall. It is also forced high when the Mute Reset and P.T.L. input is activated and resets on the falling edge of the input pulse. Reset of the Mute output results when suffix D is included in an incoming address sequence or when a second address/data alert is reset at the Alert Reset input.

MUTE RESET AND PUSH TO LISTEN INPUT - logic input with pull down resistor. A logic 1 at the input activates the Mute output (P.T.L. feature) and resets an alert output sequence. The negative going input edge resets the Mute output.

GROUP DIGIT SELECT INPUT - logic input with on-chip pull-down resistor. The address decoder is programmed to recognise tone A as the group digit by a logic 0 or open circuit at this pin. A logic 1 programmes tone 0 as the group tone.

DATA FLAG OUTPUT - CMOS logic output. When address suffix B is decoded the Data Flag outputs a logic 1 and resets back to a logic 0 at the end of the tone sequence.

Dj0, Dj1, Dj2, Dj3 DATA JAM INPUTS. These pins connect directly to the tone data inputs of the address decoder, and 1M ohm resistors feed from the outputs of the tone decoder. Normal operation of the device requires these pins to be open circuited. Should it be required, for example, to abort address decoding data code F can be forced on these pins from low impedance sources (max. 10k ohms). A pulse at the Data Change input strobes the data into the device.

Q0, Q1, Q2, Q3 TONE DATA OUTPUTS - CMOS logic outputs. The current state of the tone decoder is output as a 4 bit word.

VDD - positive supply input.

SIGNAL INPUT - analogue input. The selective calling and speech signals are input to the hybrid at this pin.

DECODER OPERATION

Figure 4 is a flow diagram for the FX2030 decoder.

The tone decoder in the hybrid detects all valid input tone sequences.

The decoder limits the input signal and processes it in an autocorrelator. The autocorrelator has the effect of enhancing the coherence of signalling tones while rejecting random noise. The periodic signal information is detected and passed to accurate digital divide/count circuits where fifteen tone bands are simultaneously scanned by the decoder. These are tuned to detect the tone digits 0 to 9, repeat tone, group tone and three instruction tones. The divide/count outputs are further processed to optimise the performance of the decoder.

The tone data is compared by the address decoder chip with the data stored in an external memory. If an incorrect sequence is detected the decoder aborts and remains in standby until the end of the tone sequence. The decoder includes circuitry to identify when the repeat frequency is expected in an address sequence. It automatically programmes repeat in place of the code digit held in the address memory.

Group Calling.

Completely flexible group call decoding is provided by the FX2030, without constraints on the calling structure. The examples show Tone A representing the group call frequency and E being a repeat of the previous tone digit:

Transmitted Tones	Units Called
1234A	12340-12349
12AEA	12000-12999
A2345	02345-92345
AEAEA	All units

If a group tone is included as part of a correct incoming sequence the decoder sets an internal latch. At the end of the sequence the state of the latch is inspected and transponder operation will be inhibited for group calls. This avoids the situation where multiple receivers are simultaneously keyed.

Alert.

An alert output will be generated next in the sequence. It is however inhibited for 'Silent Interrogate' (refer to Suffix Tones section) as shown by the 'NO' branch in the flow diagram of Figure 4.

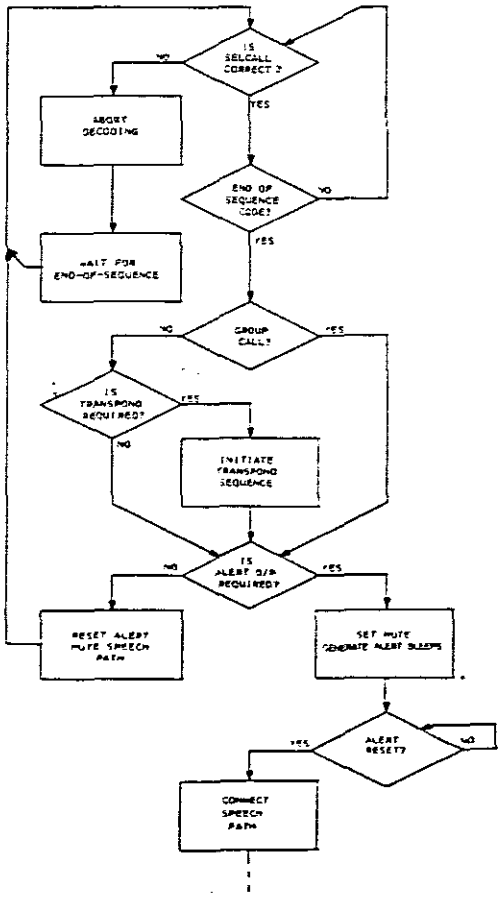


FIGURE 4 DECODER FLOW DIAGRAM

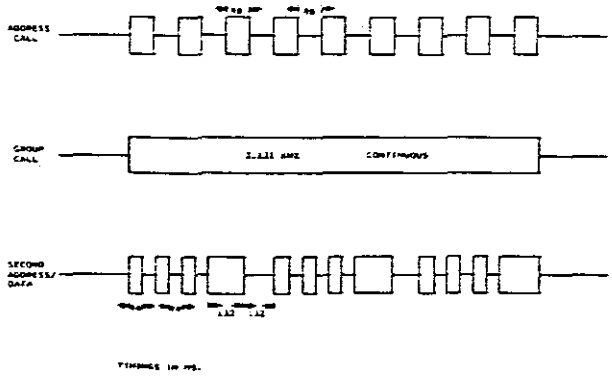


FIGURE 5 ALERT OUTPUT PATTERNS

The alert signal at the Alert/Audio output (Pin 6) is derived from the logic level square wave at Pin 13. The alert source impedance at Pin 8 is typically 15k ohms.

The 2.121kHz clock is chopped as shown in Figure 5. An even mark/space alert signals an address call and a continuous note indicates detection of a group call. A second address/data call is announced by the distinctive pattern shown.

The alert can be automatically reset by providing a capacitor 'C' on the Alert Reset input Pin 9. Automatic time out is not available for address calls where a transpond is not selected or for second address/data calls. These must be reset externally by a logic 1 pulse at Pin 9.

An alert will also be cleared by activating either the Mute Reset & P.T.L. or *Transmit Enable inputs.*

Audio Squelch

The FX2030 features an audio switch in the signal path. *This transmission gate is controlled by the decoder and* is automatically turned on after the alert has reset. The speech signals at the Signal input are now available at the Alert/Audio output.

Pin 30 is the Mute Reset & P.T.L. Input and as its name implies offers two functions. Manual defeat of the mute is provided with a high level at this pin. The 'Push to Listen' action is used to check channel occupancy.

The falling edge of the input pulse at Pin 30 is detected and used to mute the receiver.

The remote mute facility is discussed in the Suffix Tones section.

Figure 6 (i) gives the timing relationships between the Tone Data outputs and the Data Change output. The Data Change strobe causes the Memory Address outputs to increment and the critical Memory Data input timings are detailed.

Figure 6(ii) shows the timings of the Mute output, Alert Bleep Logic output and the Alert/Audio output following receipt of a correct selcall.

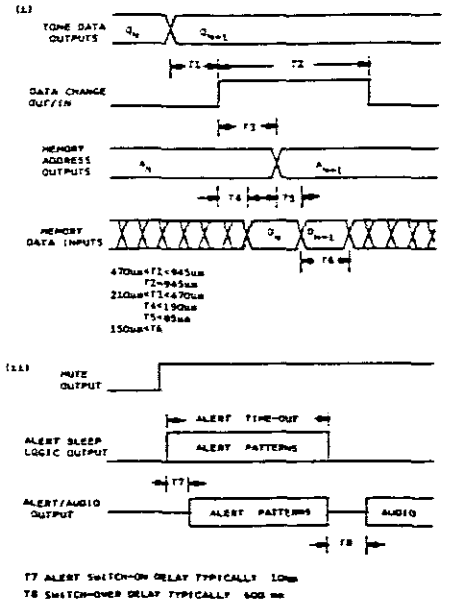


FIGURE 6 MEMORY AND ALERT OUTPUT TIMING RELATIONSHIPS

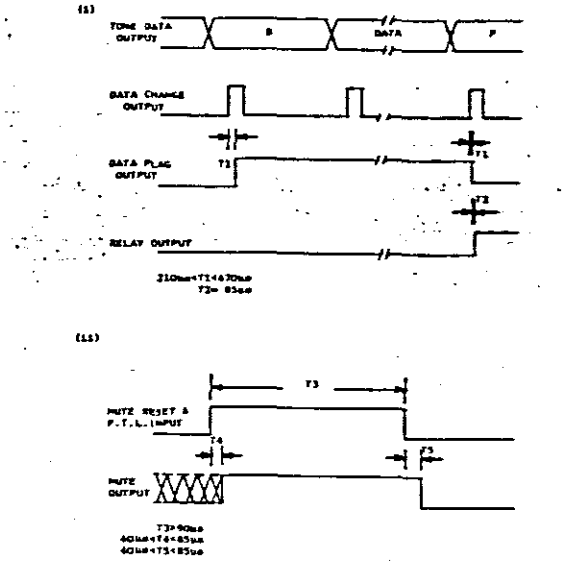


FIGURE 7 DATA AND MUTE OUTPUT TIMING RELATIONSHIPS

The Data Flag output relationships are detailed in *Figure 7 (i) and the Mute and* Mute Reset input in Figure 7 (ii).

ENCODER OPERATION

Transmit.

The hybrid can be used to transmit a code sequence as a calling or identity signal, as illustrated in the flow diagram of Figure 8.

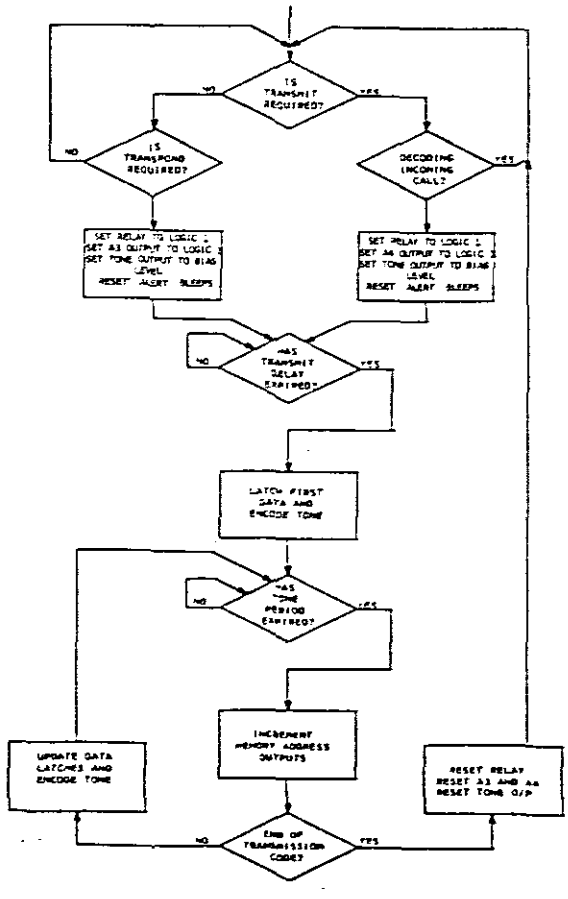


FIGURE 8 ENCODER FLOW DIAGRAM

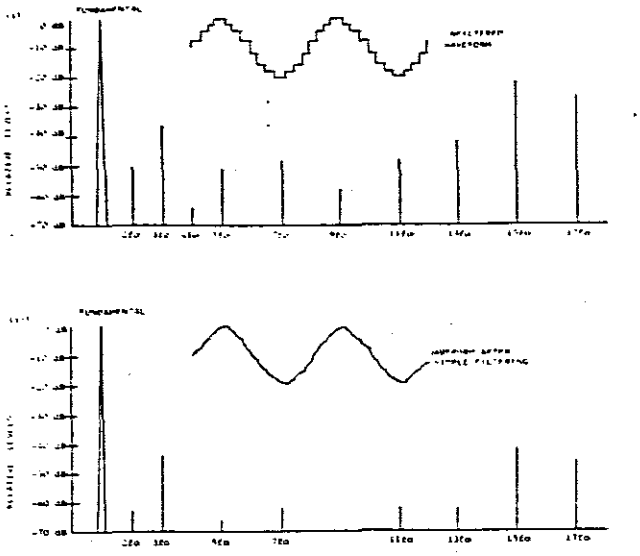


FIGURE 9 ENCODER OUTPUT WAVEFORM AND SPECTRAL ANALYSIS

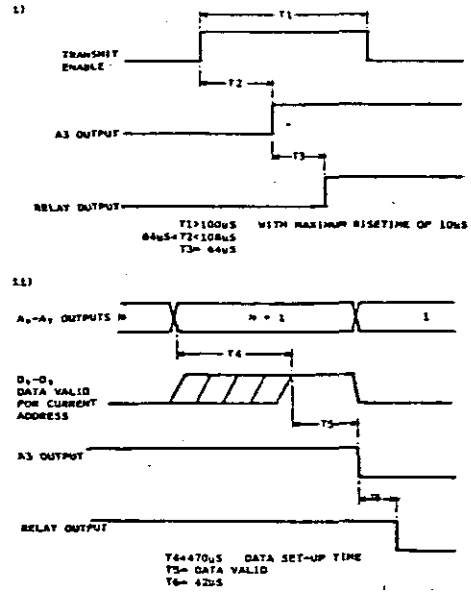


FIGURE 10 TRANSMIT ENABLE AND MEMORY TIMING RELATIONSHIPS

A positive pulse at the Transmit Enable input is coupled to the control chip (FX403) and will be accepted unless the decoder is processing an incoming selcall. The FX403 will enable the encoder chip (FX503) and set the Relay and A3 outputs high.

The Relay line is used to switch from the FX503 standby clock of 23.333kHz to a 560kHz clock used during encoding. This technique is used to reduce the overall power consumption to a minimum.

The Relay line is also used to start a delay period of typically 210ms. This transmit delay allows time to key the transmitter in a transceiver before the start of the first tone. Additionally the Tone output is set to a d.c. level of approximately 0.25 volts. The tones are generated about this mean d.c. level and the delay allows conditions to stabilise before encoding starts.

The tone data is read from the memory and automatically updated as required by the FX2030. Figure 9 (i) illustrates the tone waveform generated by the device together with the relative levels of the harmonics of the tone. The harmonic content can be further reduced if necessary by the simple filtering provided by a 0.1μF capacitor between the Tone output and VSS. This would yield the waveform and spectral response shown in Figure 9 (ii).

Each tone in the sequence is generated for an accurately timed period when the hybrid will automatically access the next memory location. The new data is inspected for the end of transmission instruction (code E) from the memory which causes the Relay and A3 outputs to be reset to logic 0. The Tone output will revert to its quiescent level (VSS) and the encoder chip is returned to the low current standby condition.

Transpond.

The transpond feature is used to automatically acknowledge correct decoding of a selective call. The reply signal can be decoded by the calling equipment to verify that contact is established.

The FX2030 is programmed into transpond mode by a logic 1 or an open circuit at the Transpond Select input (pin 12). Logic 0 on this pin will inhibit transponder operation. The encoder sequence is initiated immediately after the end of an incoming call as illustrated in Figure 2 and Figure 8. During the sequence both A4 and the Relay outputs will be at Logic 1 and will reset when the FX503 is disabled at the end of transmission. Figure 10 (i) gives the timing relationships between a pulse at the Transmit Enable input and the A3 and Relay outputs. In Figure 10 (ii) the outputs A_0-A_2 are shown addressing memory location N + 1. This contains code E which flags the end of an encode sequence.

SUFFIX TONES

The function of the FX2030 can be modified by adding extra tones after a selcall sequence. One or more of the three instruction tones B, C and D can be added to the end of a call.

Tone B.

Tone B can be used in a system in one of two ways.

Data messages can be added to selective calls by using additional tones to represent the data characters. Address and data sections of the call are separated by address suffix Tone B. The Data Flag output (pin 32) will be set high for data loading when tone B follows a valid address and remains high until the end of the tone sequence. The data is available as four bit words (hexadecimal) at the Tone Data Outputs (pins 37-40). Figure 14 shows the FX2030 interconnected to an FX313 display driver circuit.

The suffix Tone B also causes the circuit to generate a different second address/data alert output as illustrated in Figure 5. Automatic time-out of the alert is inhibited and can be reset by either the Transmit Enable or Mute Reset & P.T.L. inputs. If a selective call is transmitted with Tone B suffix but no data it can be used as a 'second address' facility.

A second address call could be used when a positive response such as 'contact the base' is required.

Tone B can be used in conjunction with C or D but must always be transmitted last in the sequence.

Tone C.

A call followed by Tone C will reverse the transpond selection programmed at Pin 12. Therefore a response can be inhibited from an equipment which normally replies or alternatively it can be requested. This feature may be employed in a system where not all equipments have the capability of decoding transponded calls. Including Tone C at the end of a transpond sequence will prevent the possibilty of transponder lock-up. This can occur where a transponded call is decoded and itself causes a further transpond.

Suffix Tone C is redundant for group calls.

Tone D.

Suffix instruction Tone D can be added to a call and results in two actions. Firstly it will both reset any alert running when the call was received, and inhibit an alert output. It will also mute the receiver audio by disabling the audio switch and reset the Mute output.

The mute and reset instruction does not affect the transponder operation of an address call so it forms a 'silent interrogate' instruction. Equally the suffix tone can be applied to an all-call to reset all the receivers on a system.

MEMORY CODING

The external memory contains the address code for the receiver, together with the transmit and transpond codes. The address digits are held as four bit (hexa-decimal) codes in the memory. Six different outputs from the hybrid could be used as address lines to the memory.

A0, A1 and A2 outputs increment in a binary count from 001 as shown below. These are used as the three least significant digits of the memory address. The A3 output goes high only for the time that a transmit instruction (from the P.T.T. button) is being encoded. Similarly A4 is set to logic 1 during a transpond sequence, while the Relay output is high whenever the hybrid is transmitting.

The interconnection chosen from the five options shown in Figure 11 will be governed by the requirements of the system.

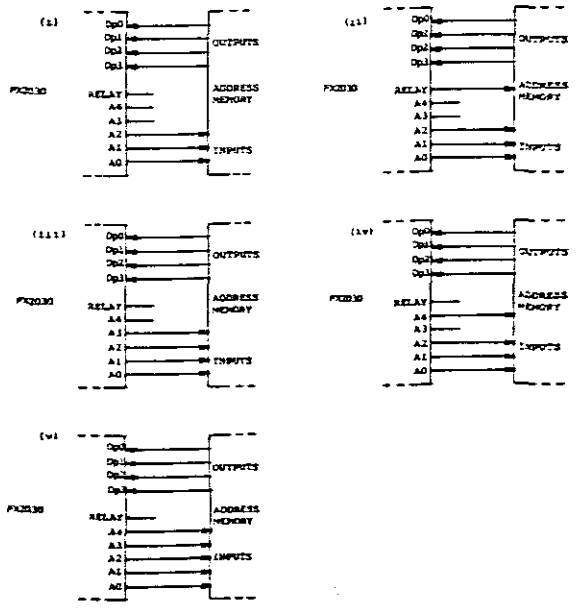


FIGURE 11 MEMORY INTERCONNECTION OPTIONS

The FX2030 fulfills the requirements of 5 tone selcall systems but is also compatible with other sequence lengths. The hybrid is instructed when the sequence is complete by code E in the memory as indicated. Therefore if for example four tones are required the fifth location would be programmed with code E as shown in example (v). Codes where two successive digits are the same do not require special programming. As illustrated in (ii) 67899 is coded as normal but the hybrid substitutes the repeat tone where necessary.

Illustration (i) is the simplest interconnection pattern. The outputs A0, A1 and A2 will provide indentical receive (Rx), transmit (Tx) and transpond (Tp) codes. An example of the codes held in the memory to give a call number 12345 would be:

Example (i) Rx code 12345
Tx code 12345
Tp code 12345

Memory location (binary)	001	010	011	100	101	110
Rx/Tx/Tp code (hexadecimal)	1	2	3	4	5	E

In (ii) the Relay output is used as an input to the memory. To programme 12345 as the receive code and 67899 for a calling number, the memory would contain:

Example (ii) Rx code 12345
Tx code 67899
Tp code 67899

Memory location (bin)	0001	0010	0011	0100	0101	0110
Rx code (hex)	1	2	3	4	5	E
Memory location (bin)	1001	1010	1011	1100	1101	1110
Tx/Tp (hex)	6	7	8	9	9	E

The interconnection pattern in (iii) utilises A3 output. It is used where the receiver code 12345 is automatically transponded but a different number 67899 is needed when the Transmit Enable button is pressed.

Example (iii) Rx code 12345
Tx code 67899
Tp code 12345

Memory location (bin)	0001	0010	0011	0100	0101	0110
Rx/Tp code (hex)	1	2	3	4	5	E
Memory location (bin)	1001	1010	1011	1100	1101	1110
Tx code (hex)	6	7	8	9	9	E

(iv) Should it be required to transmit (using Transmit Enable) the receive code as an identity, but an alternative code when transponding then interconnection (iv) is used.

Example (iv) Rx code 12345
Tx code 12345
Tp code 67899

Memory location (bin)	0001	0010	0011	0100	0101	0110
Rx/Tx code (hex)	1	2	3	4	5	E
Memory location (bin)	1001	1010	1011	1100	1101	1110
Tp code (hex)	6	7	8	9	9	E

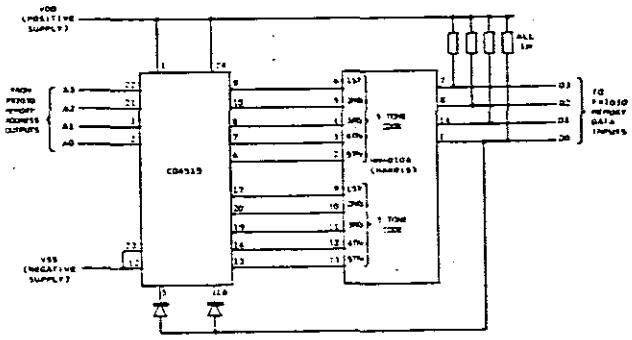


FIGURE 12 LOW POWER MEMORY

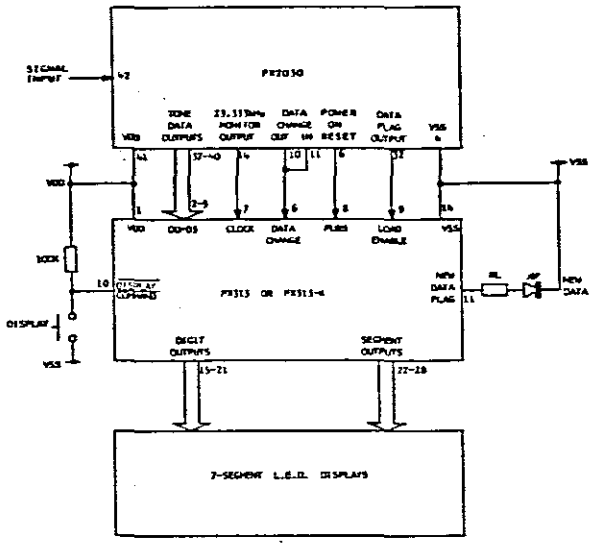


FIGURE 14 ADDITIONAL CIRCUITRY TO ADD LED DISPLAY DRIVER

(v) Connecting the FX2030 to the memory as (v) will give independent codes for receive, transmit and transpond. The five lines A0 to A4 are used to address the memory. Four tone sequential codes are illustrated.

Example (v) Rx code 1234
Tx code 6789
Tp code 4567

Memory location (bin)	00001	00010	00011	00100	00101
Rx code (hex)	1	2	3	4	E
Memory location (bin)	01001	01010	01011	01100	01101
Tx code (hex)	6	7	8	9	E
Memory location (bin)	10001	10010	10011	10100	10101
Tp code (hex)	4	5	6	7	E

Suitable memory types are fusible link PROM's or U.V. EPROM's in CMOS technology.

Applications requiring minimum possible power consumption can utilise a fusible link diode matrix and a 4-16 line decoder where space allows. The arrangement shown in Figure 12 has two independent five tone codes programmed into the matrix.

APPLICATIONS

The FX2030 is designed to provide selcall encode/decode facilities in application where both space and power consumption are prime considerations. Battery powered equipments such as hand-held radios provide an obvious example. Figure 13 shows the application of the FX2030 in a transceiver. The output from the demodulator is fed to the selcall signal input and the *Alert/Audio output is connected to the audio amplifier.* The Mute output can be used as an enable control.

A summing amplifier combines the microphone signals with the FX2030 Tone output before passing to the modulator. The transmitter section and aerial switching can be controlled by the Relay output combined with the P.T.T. button.

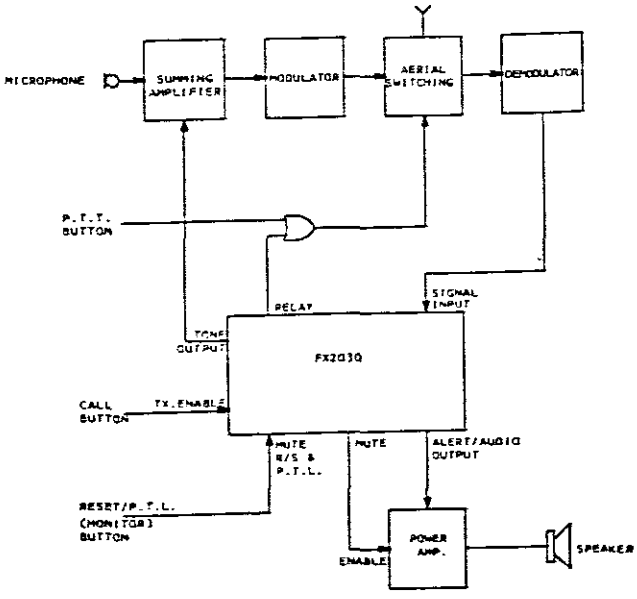


FIGURE 13 APPLICATION IN A RADIO TRANSCEIVER

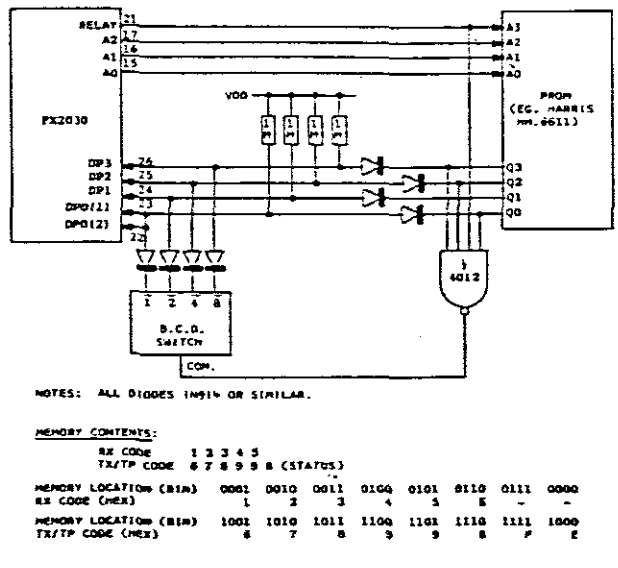


FIGURE 15 ENCODING A SINGLE STATUS DIGIT

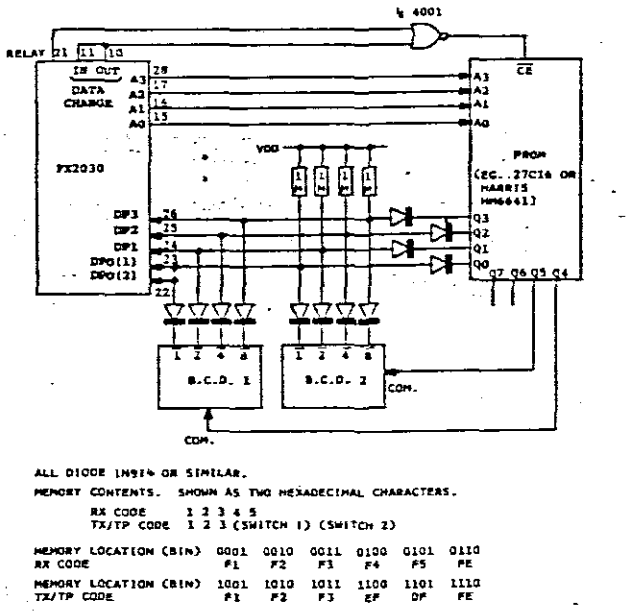


FIGURE 16 APPLICATION TO PROVIDE TWO VARIABLE TRANSMIT DIGITS

Figure 14 identifies the interconnection required to add an FX313 display/driver to a system. The LED displays data characters transmitted selectively to the hybrid.

The circuit shown in Figure 15 details the changes necessary to provide encoding of a single status digit. A high level at the FX2030 data input will be due to the effect of the pull-up resistor, while a low level is derived from forward biasing the diode.

The memory in this example has a 4-bit output, and must give a valid logic level at the FX2030 data input when the forward volt drop of the diode is included. The gate decodes the state during transmit when the memory data is F and enables the status switch. The DpO-Dp3 data inputs will then be coded via the diodes from the switch. The memory is programmed to decode 12345 as an address call number but to encode in either transmit or transpond a sequence 67899B followed by the status digit. Tone B is used in this instance to separate the 5-tone code from the status information tone to avoid aliasing selcall codes.

Figure 16 shows an efficient technique of enabling selector switches during a transmit sequence. The PROM is organised as an 8-bit output with 4 bits used to carry the hexadecimal tone codes. The 'common' input of each switch is allocated to one of the remaining memory outputs and is pulled low at the correct time by programming logic 0 at that output. These are shown as the hexadecimal codes E and D programmed into the Tx/Tp code sequence.

The circuit schematic shown is used to vary the last two digits of a 5 tone code. All receivers on the network would have the same first three digits, in this case 123XX. Up to 100 receivers could therefore be directly called using two selector switches.

The NOR gate shown in Figure 16 illustrates a technique for saving power. The FX2030 only requires to inspect the PROM data during decoding or encoding. The power dissipated at other times can be greatly reduced by disabling the memory. The chip enable or select input is therefore controlled by combining the Relay output with the Data Change output from the FX2030. The hybrid must always have logic levels applied at its inputs and these are provided by the pull-up resistors during standby.

INTERFACING AND ELECTRO-MAGNETIC COMPATIBILITY

The FX2030 requires a clock of 560kHz which is internally converted to logic level square waves. Consideration should therefore be given to possible interference problems with RF or IF circuitry caused by 560kHz or to its harmonics. Similarly the performance of the device could be impaired if high levels of radiated power were present near the high impedance offered by the Signal input.

During the time that the audio is muted the transmission gate between the input and output is disabled (pins 42 and 8). The impedance offered at the output is therefore very high. When the mute is lifted the hybrid closes the speech path with an impedance of typically 1k ohm. The changes in the loading effects on circuitry and any possible interference pick up from high impedances should be considered when interfacing the FX2030 into a system.

HANDLING PRECAUTIONS

The FX2030 contains CMOS LSI circuits which include input protection circuitry. However, precautions should be taken to prevent static discharges which can cause damage.

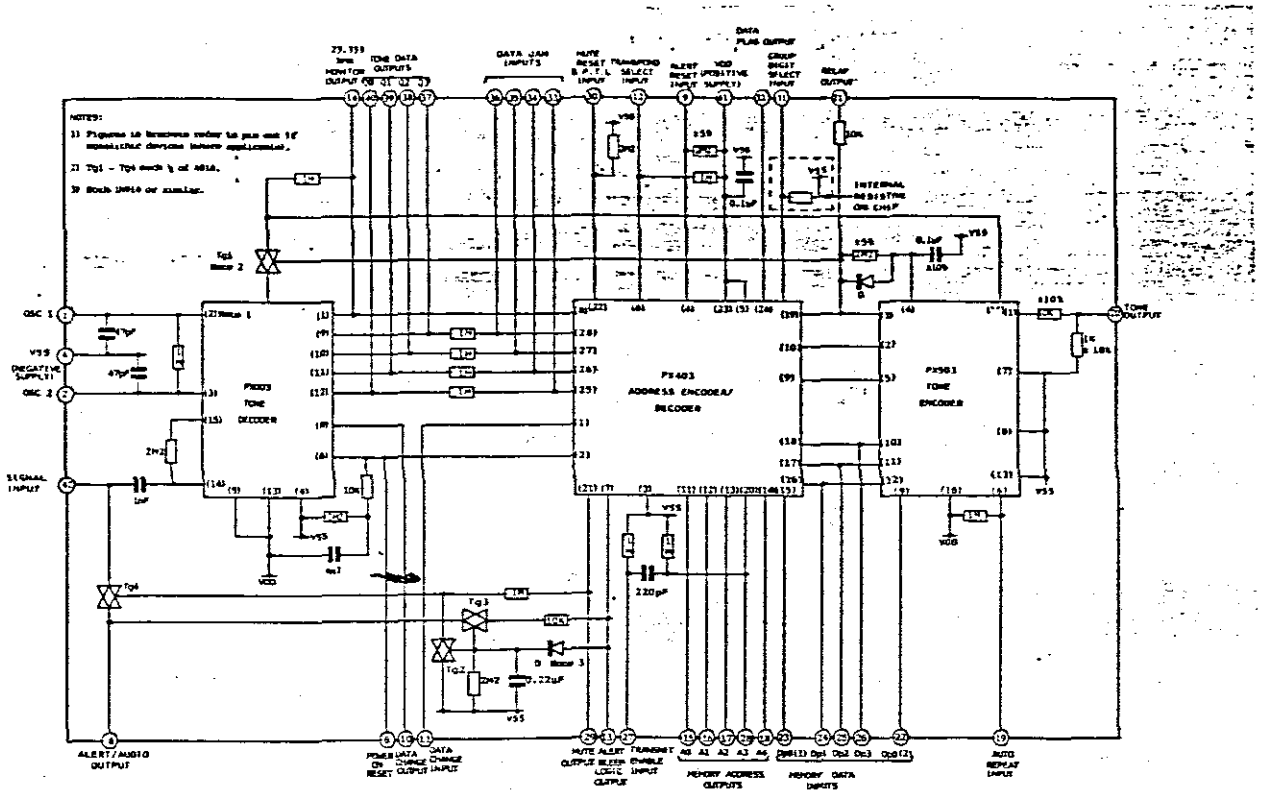


FIGURE 17 INTERNAL SCHEMATIC

MECHANICAL

The hybrid is constructed on ceramic substrates with the chips and other discrete components added. The chips are individually protected and the whole assembly is then coated to provide both mechanical and environmental protection. The package outline is given in Figure 18.

Pin 1 is identified on the package outline of Figure 18 and a '1' is printed on the top of the product.

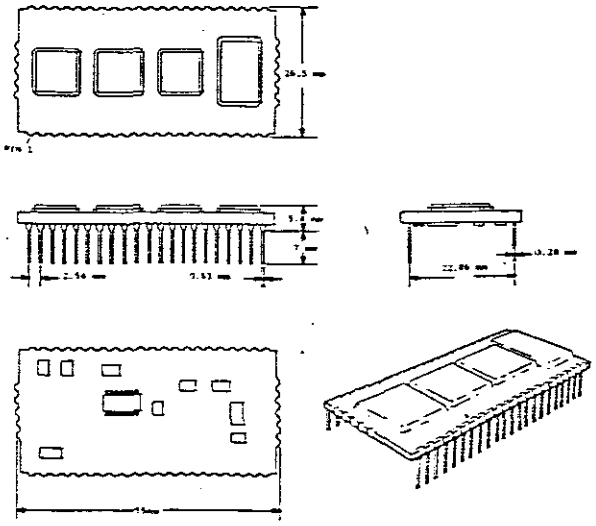


FIGURE 18 PACKAGE OUTLINE

NOTE:
CML does not assume any reponsibility for the use of any circuitry described. No circuit patent licences are implied and CML reserves the right at any time without notice to change said circuitry.

CONSUMER MICROCIRCUITS LIMITED
WHEATON ROAD INDUSTRIAL ESTATE EAST
WITHAM ESSEX CM8 3TD ENGLAND
Telex: 99383
Cables: Reaction Witham
Telephone: 0376 51388

PRODUCT INFORMATION

QTC Selective Call FX003 Tone Decoders

Publication D/003/4 February 1990

Features

- 15-Tone Selcall Decoder
- Group Call and Data Capability
- Excellent Noise Performance
- 4 Bit Data Output
- Few External Components
- CCIR, ZVEI, EEA, EIA Tone Sets
- μ Processor Compatible
- High Dynamic Range
- Low Power CMOS
- On-Chip Oscillator Uses Low-Cost Resonator

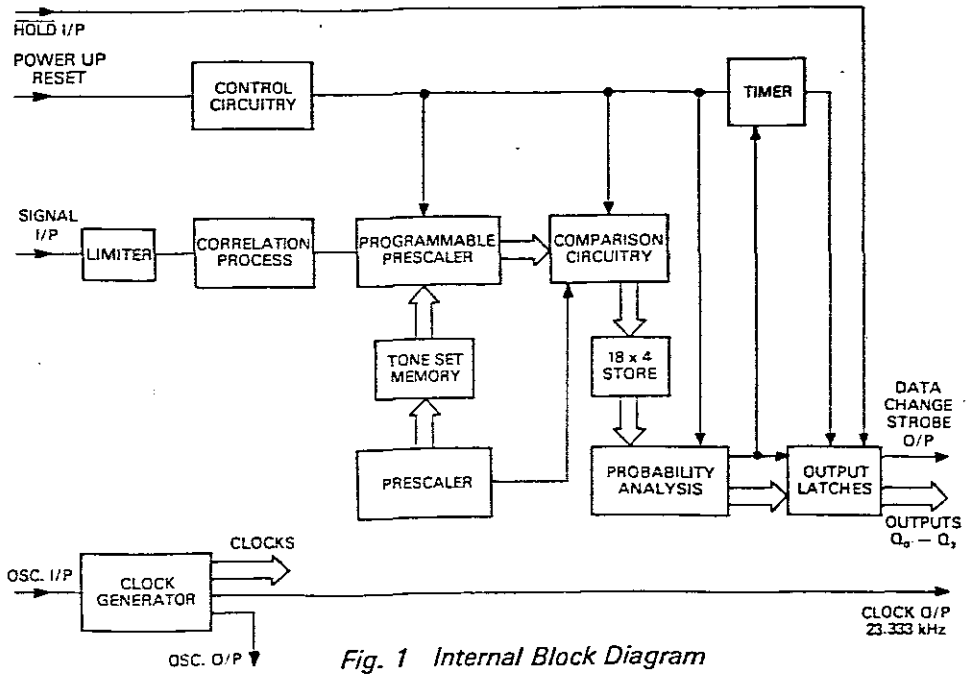


Fig. 1 Internal Block Diagram

FX003QC
FX003QZ
FX003QE
FX003QA
FX003QZS

Brief Description

The FX003 is a CMOS QTC (Quadradecimal Tone Coding) tone decoder which may be used to decode Selcall tones in accordance with CCIR, ZVEI, EEA and EIA international tone standards.

The FX003 detects an input frequency falling within any of the fifteen tone channels programmed on-chip and outputs the hexadecimal tone number in 4-bit binary code. When a tone is detected, its 4-bit code is latched at the data outputs and a Data Change is generated. Failure to qualify any tone for a continuous period of 33 ms causes the output to be set to 'Notone' (16th logic state) and a Data Change strobe to be generated.

A DATA CHANGE output signals each change in the output code and can be used with the HOLD/ACKNOWLEDGE input to establish handshake routines with microprocessors and other data processing logic.

A 'Power Up Reset' (PURS) routine ensures all internal circuitry is correctly reset when power is first applied to the device. Following 'PURS' the FX003 generates HEX 'E' (NO DATA CHANGE) which in turn is followed by a normal decode sequence.

The on-chip inverter may be used with a resonator to provide the 560 kHz master clock for the device, or an external clock may be used. A divided down buffered 23.33 kHz clock output is also provided for use with other '03 devices and trimming of the 560 kHz resonator.

The FX003 is available in a number of pin compatible versions, each version programmed in accordance with the frequencies and bandwidths of a specified QTC toneset.

Pin Description (See Figure 2)			Function
D.I.L. FX003*	Chip Carrier FX102K (see Note A)	Chip Carrier FX202*K (see Note A)	*QC, QZ, QE, QA, QZS
1	2	—	**23.333 kHz Clock O/P:** A 23.333 kHz buffered squarewave logic output directly derived from the oscillator frequency (nominally 560.0 kHz). May be used for auxiliary functions e.g. 560 kHz resonator trimming,external timing of received tone periods and for other '03 family products.
2	4	—	**$\overline{\text{Xtal}}$:** Output from on-chip inverter.
3	6	—	**Xtal/Clock:** Input to on-chip inverter may be used in conjunction with $\overline{\text{Xtal}}$ O/P and 560 kHz resonator or as a buffered input for an external clock (nominally 560.0 kHz).
4	10	5	**VSS:** Negative supply.
5	—	8	**$\overline{\text{Hold}}$ I/P:** Active when at VSS. If hold is taken to VSS when the input tone changes it latches the next data change pulse at logic 1 until the $\overline{\text{Hold}}$ is returned to VDD. This facilitates Interrupt/Handshake routines for microprocessors when used in conjunction with the Data Change O/P. Tie to VDD if not used.
6	—	9	**Power Up Reset:** A logic 1 level of at least 1 ms duration is required at this pin to reset internal circuitry on power-up. For slow-rising power supplies increase the time constant of the components shown accordingly.
7 & 16	1 3 5 7 8 9 11 12 14 15 16 18 20 21 22 23 25 27	1 2 3 6 7 10 11 13 14 15 18 20 23 24 26 28	**Internally Connected/Open Circuit:** Should be left open circuit.
8	—	16	**Data Change:** A pulse is generated at this pin shortly after detection of a tone and new data being presented at the Q_0 — Q_3 outputs *(see Figure 5 Timing Diagram).*
9	—	22	Q_3 | **Data Outputs:** A 4-bit word which
10			Q_2 | is output after a successful decode
11			Q_1 | and represents the Hex code for the
12			Q_0 | decoded tone frequency.
13	19	25	**VDD:** Positive Supply.
14	24	—	**Signal Input:** Audio selcall tones are a.c. coupled to this pin via a capacitor. D.C. bias of the internal high gain limiter is set up by connecting this pin via a resistor to the bias pin.
15	26	—	**Signal Bias:** These pins should not be loaded with any other circuitry.
—	—	3	93.333 kHz Osc I/P ⎫ FX102K/FX202*K: Interchip connections.
—	13	—	93.333 kHz Osc O/P. ⎬ These pins should not be loaded with any external
—	—	27	Logic Signal I/P. ⎪ circuitry.
—	17	—	Logic Signal O/P ⎭ FX003*D.I.L: internally connected.

Note A:
*FX102K and FX202*K are sold as a pair, and represent the same circuit function as the FX003* D.I.L. device.*

EXTERNAL COMPONENT CONNECTIONS

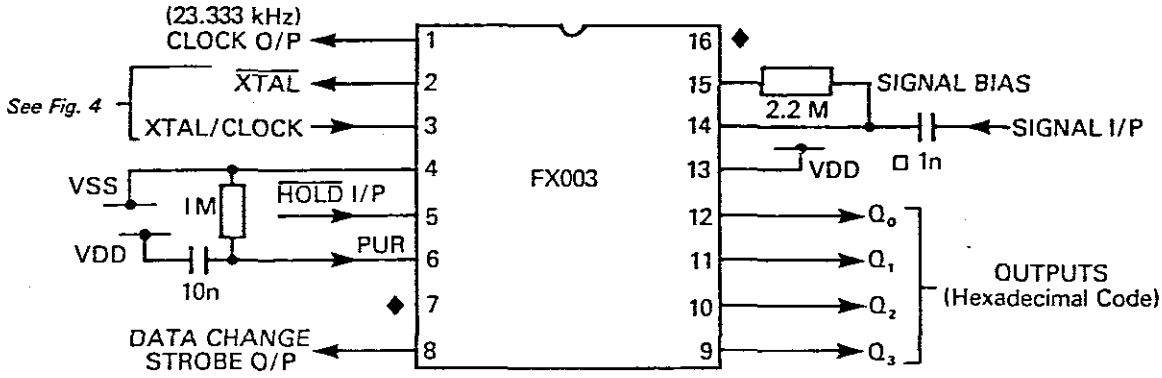


Fig. 2 Dual-In-Line

* *QC, QZ, QE, QA, QZS*
◆ *No connection. Do not tie.*
□ *1n recommended value for C, Z, E and ZS versions.*
2.2n recommended value for the A version.

INTERCONNECTION OF FX102K AND FX202*K AND EXTERNAL COMPONENTS

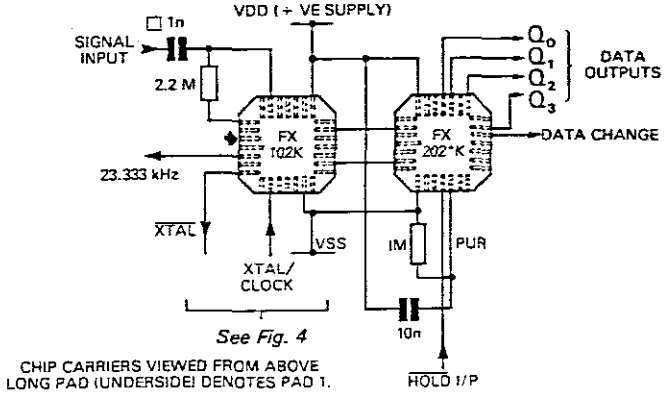


Fig. 3 Chip Carrier

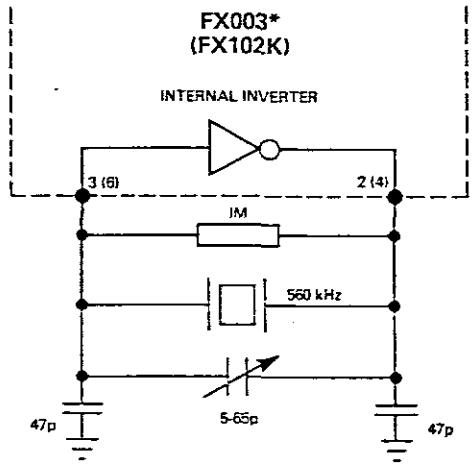


Fig. 4 560 kHz Resonator Circuit

Character Tone Table

Tone Frequencies (f_0) in Hz

003QA (EIA)	003QC (CCIR)	003QE (EEA)	003QZ (ZVEI)	003QZS (ZVEI-S)	Output Code Q_3	Q_2	Q_1	Q_0	QTC Format Character
600	1981	1981	2400	2400	0	0	0	0	0
741	1124	1124	1060	1060	0	0	0	1	1
882	1197	1197	1160	1160	0	0	1	0	2
1023	1275	1275	1270	1270	0	0	1	1	3
1164	1358	1358	1400	1400	0	1	0	0	4
1305	1446	1446	1530	1530	0	1	0	1	5
1446	1540	1540	1670	1670	0	1	1	0	6
1587	1640	1640	1830	1830	0	1	1	1	7
1728	1747	1747	2000	2000	1	0	0	0	8
1869	1860	1860	2200	2200	1	0	0	1	9
2151	2400	1055	2800	886	1	0	1	0	A
2433	930	930	810	810	1	0	1	1	B
2010	2247	2247	970	740	1	1	0	0	C
2292	991	991	886	680	1	1	0	1	D
459	2110	2110	2600	970	1	1	1	0	E
NOTONE	NOTONE	NOTONE	NOTONE	NOTONE	1	1	1	1	F

Specification

Absolute Maximum Ratings

Exceeding the maximum rating can result in device damage. Operation of the device outside the operating limits is not implied.

Supply voltage		−0.3V to 7.0V
Input voltage at any pin (ref VSS = OV)		−0.3V to (VDD + 0.3V)
Output sink/source current (total)		20mA
Operating temperature range:	FX003*	−30°C to +85°C
	FX102K/FX202K*	
Storage temperature:	FX003*	−55°C to 125°C
	FX102K/FX202*	
Maximum device dissipation		100mW

*QC, QZ, QE, QA, QZS

Operating Limits

VDD = 5V, T_A = 25°C, ϕ = 560kHz, $\Delta f\ \phi$ = 0.
All characteristics measured using the standard test circuit with the following test parameters, and is valid for all tones unless otherwise stated:—

Characteristic	See Note	Min	Typ	Max	Unit
Static Characteristics					
Supply voltage (VSS = OV)		3.3	5.0	5.5	V
Supply current			500		μA
Logic '1' output I source = 0.1 mA	1	4.5			V
Logic '0' output I sink = 0.1 mA	1			0.5	V
Logic '1' input Level	2	3.5			V
Logic 'O' input level	2			1.5	V
Dynamic Characteristics					
Signal input range	3	0.1		VDD	Vpk-pk
Decode Bandwidth (P⩾0.995)	4				
QA	4a	20			±Hz
QC	4b	1			±%
QE	4c	1			±%
QZ/QZS	4d	2			±%
Not-decode bandwidth (P⩽0.03)					
QA	5			60	±Hz
QC	5			3	±%
QE	5			3	±%
QZ/QZS	5			4.5	±%
Noise response rate (hours per F→ $\bar{F}$→ F single character response with no input tone).					
QA	6		0.15		Hour
QC	6		40.0		Hour
QE	6		40.0		Hour
QZ/QZS	6		1.0		Hour
Decode response time:					
Notone to tone (F→ $\bar{F}$)	7	20	25	Tp	ms
Tone to notone, T_f ($\bar{F}$→ F)	7	33		53	ms
Min. intertone gap for 'F'	8	15		28	ms

Notes

1. *Relates to output pins 1, 8, 9-12.*
2. *Relates to input pins 5 and 6.*
3. *A.C. coupled, sine/squarewave.*
4. *With minimum tone period (Tp) specified for toneset. P = decode probability:*
   (a), (c) SNR 3 dB
   (b), (d) SNR 0dB
5. *All conditions of input SNR and amplitude with maximum Tp specified for toneset.*
6. *Gaussian input noise, bandwidth 6kHz, maximum input level corresponds to 1-digit code falsing rate. $\bar{F}$ = random single character.*
7. *Delay from change of input (tone applied/removed) to change at Q_0–Q_3 outputs (see fig. 5).*
8. *Included in T_2. Minimum tone gap requirement for 'notone' recognition. Outputs = F after delay. (see fig. 5).*

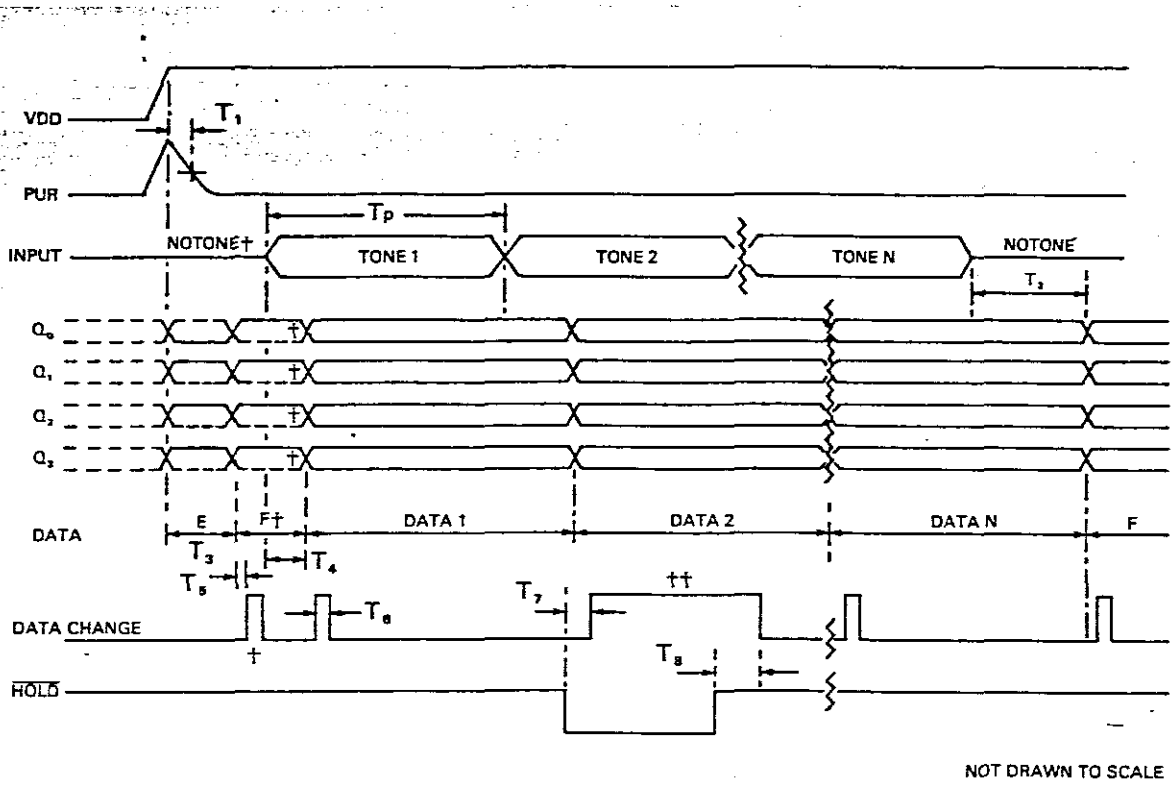


Fig. 5 FX003 Timing Diagram (See References)* ** QC, QZ, QE, QA, QZS*

Typical Performance

References:

T_1 Logic 1, > 2 ms
T_2 > 33 ms & < 50 ms
T_3 33 ms (DATA E)
T_4 20 ms minimum (Tp MAXIMUM)
T_5 0.5 ms — 1.0 ms (DATA CHANGE)
T_6 1.0 ms (DATA CHANGE PULSE DURATION)
T_7 > 50 µs
T_8 < 120 µs

† *$Q_0 - Q_3$ will represent the input frequency present during and after PUR (shown as 'F' (Notone) in this example).*

†† *After application of $\overline{HOLD}$ the next Data Change pulse will stay high until $\overline{HOLD}$ is removed according to timing shown.*

Package Outlines

The ceramic dual-in-line package of the FX003 is shown in *Figure 6* and the chip carrier version shown in *Figure 7.* For the D.I.L. package, the pins number counter-clockwise (top view) from 1 with reference to a notch as a guidance. For the chip carrier package, pins number counter-clockwise (viewed from above) from the long pad (pad 1).

Handling Precautions

The FX003 is a CMOS LSI circuit which includes input protection. However, precautions should be taken to prevent static discharges which can cause damage.

Fig. 6 FX003 D.I.L. Package*

	inches	mm.
A	0.80	20.32
B	0.59	15.00
C	0.77	19.56
D	0.11	2.79
E	0.01	0.25
F	0.30	7.62
G	0.70	17.78
H	0.10	2.54
I	0.018	0.46
J	0.155	3.94

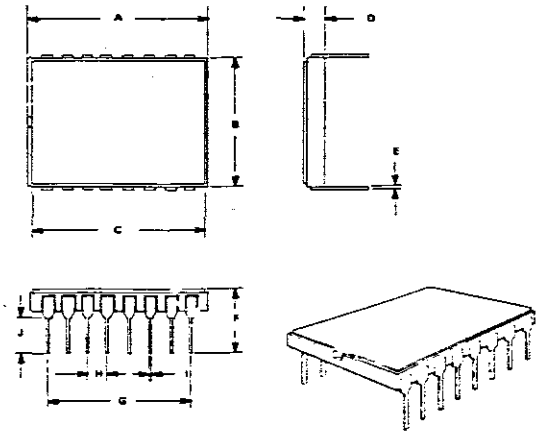
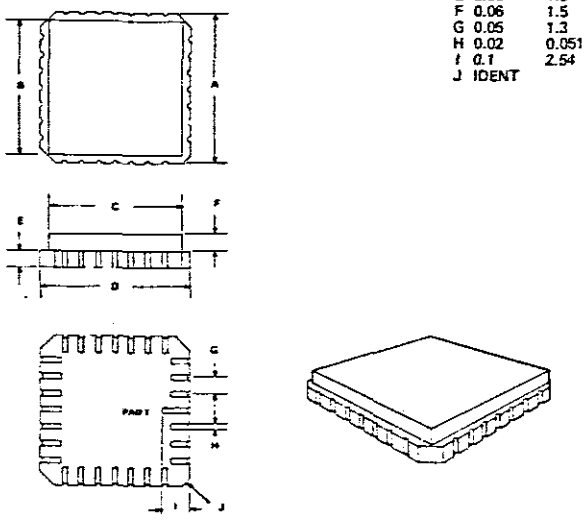


*Fig. 7 FX102K/202*K Chip Carrier Package*

	inches	mm.
A	0.45	11.4
B	0.41	10.4
C	0.41	10.4
D	0.45	11.4
E	0.06	1.5
F	0.06	1.5
G	0.05	1.3
H	0.02	0.051
I	0.1	2.54
J	IDENT	



* *QC, QZ, QE, QA, QZS*

Ordering Information

FX003*	16-pin Ceramic D.I.L.
FX102K	28-pad Ceramic Chip Carrier
FX202*K	28-pad Ceramic Chip Carrier

*** VERSIONS**

- **QC : CCIR**
- **QZ : ZVEI**
- **QE : EEA**
- **QA : EIA**
- **QZS : Suppressed ZVEI**

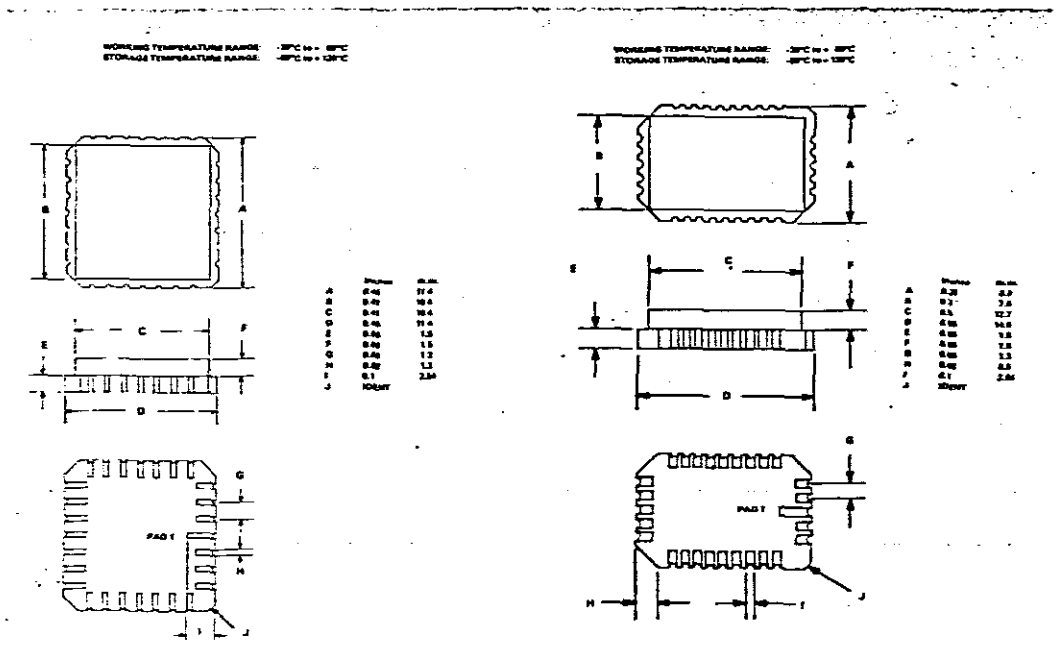
Note: *FX102K & FX202*K are available in pairs only.*

CONSUMER MICROCIRCUITS LIMITED
WHEATON ROAD · INDUSTRIAL ESTATE EAST
WITHAM · ESSEX CM8 3TD · ENGLAND
Telephone: 0376 513833
Telex: 99382 CMICRO G
Telefax: 0376 518247

PRODUCT INFORMATION

PRODUCT INFORMATION

PUBLICATION – D/FX '03-K/1 JANUARY 1982



FX102·K
FX202C·K
FX202Z·K
FX103·K
FX313·K
FX403·K
FX503C·K
FX503Z·K

FEATURES

* SMALL PHYSICAL SIZE
* C SUFFIX DEVICES TUNED TO CCIR TONESET
* Z SUFFIX DEVICES TUNED TO ZVEI TONESET
* WIDE TEMPERATURE RANGE
* HERMETICALLY SEALED PACKAGE
* LOW POWER CMOS PROCESS

GENERAL DESCRIPTION

FX003 PRODUCTS PACKAGED IN LEADLESS CHIP CARRIERS.

Products from the FX'03 range are available packaged in leadless chip carriers. This form of packaging can result in a greatly increased packing density.

The ceramic leadless chip carrier provides a high level of package hermeticity and wide operating and storage temperature ranges.

DESCRIPTION

The circuit function and electrical specification of leadless chip carrier products are identical to the dual-in-line product information with the following exceptions:

a) The FX003 tone decoder comprises two discrete LSI chips which are each packaged in a chip carrier. Both packages are required for a complete tone decoder, and are available to decode the CCIR or the ZVEI range of signalling tones. The product codes for a CCIR decoder are an FX102-K together with an FX202QC-K and for a ZVEI device are FX102-K and FX202QZ-K. The interconnection and external components of the two chip carriers are given in diagram 2.

The supply current taken by the FX102 and FX202 combined is typically 1200μA at 5 volts supply.

b) All chip carrier products are specified for operation over supply voltage range 4.5V to 6V.

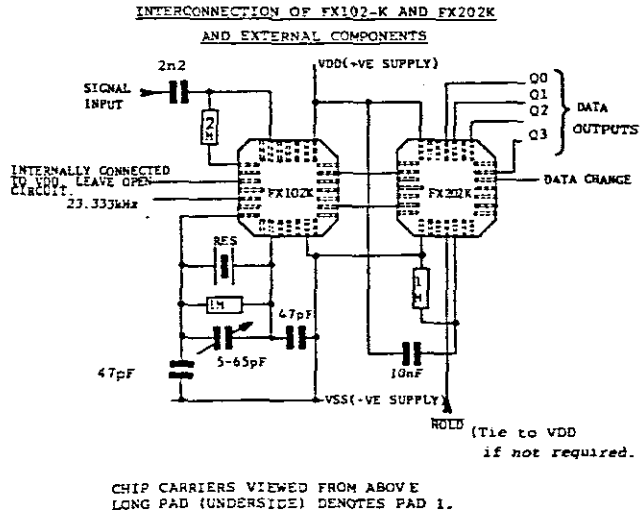


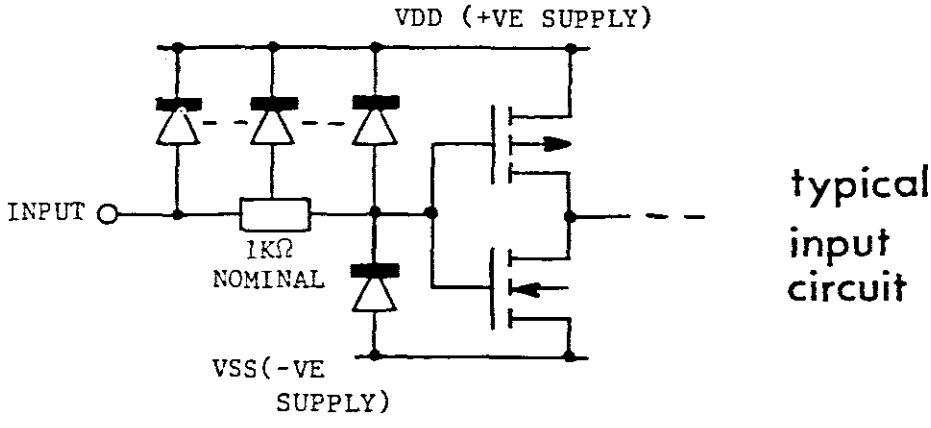
FIG.2

Methods of Attachment.

Leadless chip carriers may be attached by reflow soldering techniques. A typical reflow temperature would be 210°C and the molten solder tends to align the chip carrier over the solder pads on the substrate. The ideal substrate material is ceramic because it has a coefficient of linear expansion very closely matched to that of the carrier. Epoxy glass printed circuit boards can be used but the unmatched expansion coefficients lead to strain on the solder joints during cooling.

Handling Considerations.

Care should be taken when handling any CMOS product to minimise the possibility of the static electrical discharge into the input pins.



During storage the devices should be protected from static charges by packaging in conductive foam or storage in metal trays.

PIN DATA

PAD	FX102-K	FX202QC-K FX202QZ-K	FX103-K	FX313-K	FX403-K	FX503C-K FX503Z-K
1	N/C	N/C	Data Chng.I/P	VDD (+)	Data Chng.I/P	Tone O/P
2	23.333kHz I/P	N/C	PUR I/P	D0 I/P	PUR I/P	Tp O/P
3	N/C	93.333kHz I/P	Vss (-)	D1 I/P	Vss (-)	O/P Bias I/P
4	Osc. Bias	N/C	Alert R/S I/P	D2 I/P	Audio R/S I/P	N/C
5	N/C	Vss(-)	$\overline{\text{Dec.Abort}}$ I/P	D3 I/P	$\overline{\text{Decode}}$ I/P	Enable I/P
6	Osc. I/P	N/C	Preamble I/P	Data Chng.I/P	Tp. mode I/P	N/C
7	N/C	N/C	Alert O/P	23.333kHz I/P	Audio O/P	N/C
8	N/C	$\overline{\text{Hold}}$ I/P	23.333kHz I/P	PUR I/P	23.333kHz I/P	Data Chng.I/P
9	N/C	PUR I/P	$\overline{\text{Qp}}$ O/P	Load En. I/P	Inc.Add. I/P	Auto R I/P
10	Vss (-)	N/C	$\overline{\text{Q1}}$ O/P	$\overline{\text{Display}}$ I/P	Load O/P	N/C
11	N/C	N/C	$\overline{\text{Q2}}$ O/P	New Data O/P	Q0 O/P	N/C
12	N/C	N/C	$\overline{\text{Q3}}$ O/P	B Invalid I/P	Q1 O/P	N/C
13	93.333kHz O/P	N/C	$\overline{\text{Q4}}$ O/P	Overflow O/P	Q2 O/P	N/C
14	N/C	N/C	$\overline{\text{Q5}}$ O/P	Vss (-)	Q3 O/P	XTC/$\overline{\text{QTC}}$ I/P
15	N/C	N/C	D10 I/P	Digit 1 O/P	Dp0 I/P	Vss (-)
16	N/C	Data Chng.O/P	D11 I/P	Digit 2 O/P	Dp1 I/P	N/C
17	Logic sig.O/P	Q3 O/P	D12 I/P	Digit 3 O/P	Dp2 I/P	D0 I/P
18	N/C	N/C	D13 I/P	Digit 4 O/P	Dp3 I/P	D1 I/P
19	Vdd (+)	Q2 O/P	Group 'O'I/P	Digit 5 O/P	Tx Relay O/P	D2 I/P
20	N/C	N/C	A.C.I. I/P	Digit 6 O/P	Tx Enable I/P	D3 I/P
21	N/C	Q1 O/P	Mute O/P	Digit 7 O/P	Mute O/P	N/C
22	N/C	Q0 O/P	Mute R/S I/P	Segment g O/P	Mute R/S I/P	N/C
23	N/C	N/C	Vdd (+)	Segment f O/P	Vdd (+)	N/C
24	Signal I/P	N/C	Data Flag O/P	Segment e O/P	Data Flag O/P	$\overline{\text{Character}}$ I/
25	N/C	Vdd (+)	D0 I/P	Segment d O/P	D0 I/P	N/C
26	Signal Bias	N/C	D1 I/P	Segment c O/P	D1 I/P	Clock I/P
27	N/C	Logic sig.I/P	D2 I/P	Segment b O/P	D2 I/P	$\overline{\text{Clock}}$ O/P
28	Internal Vdd	N/C	D3 I/P	Segment a O/P	D3 I/P	Vdd (+)

ANNEXURE 2

5-TONE ZVEI ENCODER ANALYSER DISPLAY DOCUMENTATION

DMC16433 (16 characters × 4 lines) ● Display Fonts 5 × 8 Dots ● 1/16 Duty Drive

ABSOLUTE MAXIMUM RATINGS

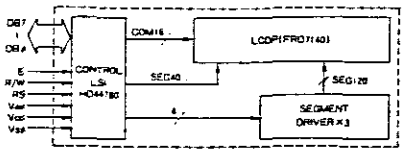
Item	Symbol	Test Condition	Standard Value min.	Standard Value max.	Unit
Supply Voltage for Logic	$V_{CC}-V_{SS}$	Ta=25°C	−0.3	7	V
Supply Voltage for LCD Drive	$V_{CC}-Vee$	Ta=25°C	$V_{CC}-13.5$	$V_{CC}+0.3$	V
Input Voltage	V_I	Ta=25°C	−0.3	$V_{CC}+0.3$	V
Operating Temperature	Topr	——	0	+50	°C
Storage Temperature	Tstg	——	−20	+70	°C

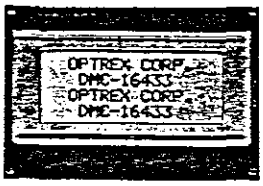
ELECTRICAL CHARACTERISTICS

Item	Symbol	Test Condition	Standard Value min.	Standard Value typ.	Standard Value max.	Unit
Input "High" Voltage	V_{IH}	——	2.2	——	V_{CC}	V
Input "Low" Voltage	V_{IL}	——	——	——	0.6	V
Output "High" Voltage	V_{OH}	$-I_{OH}$=0.205mA	2.4	——	——	V
Output "Low" Voltage	V_{OL}	I_{OL}=1.2mA	——	——	0.4	V
Supply Current	I_{CC}	V_{CC}=5.0V	——	2.4	4.0	mA

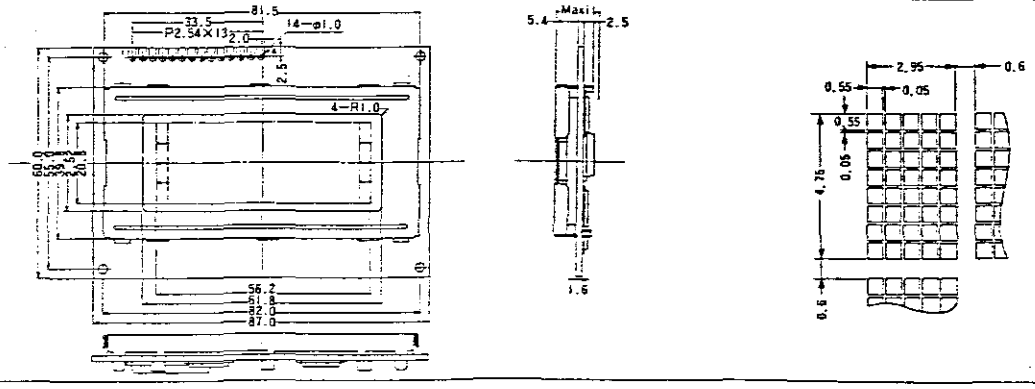
*V_{CC}=5.0V ± 5%, Ta=25°C

Block diagram





External dimensions



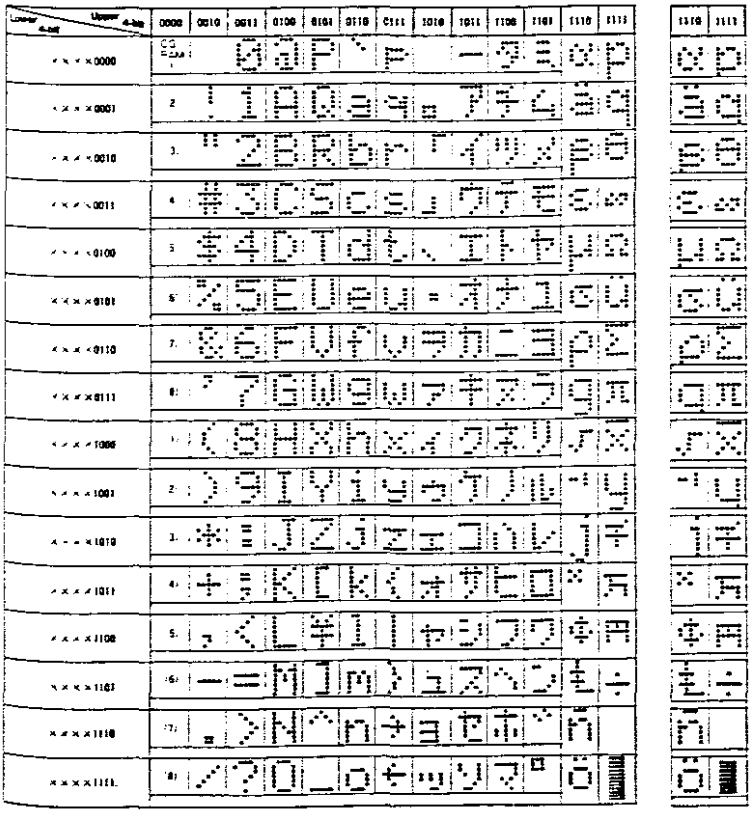
DMC Series

INSTRUCTIONS

Instruction	Code										Description	Execute Time(max.)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position(Address 0).	1.64mS
Cursor At Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position(Address 0). Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operati These operations are performed during data write and read.	40μS
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display(D) cursor ON/OFF (C), and blink of cursor position charact character(B).	40μS
Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DDRAM contents.	40μS
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length(DL) number of display lines(L) and character font(F).	40μS
CGRAM Address Set	0	0	0	1	A_{CG}						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40μS
DDRAM Address Set	0	0	1	A_{DD}							Sets the DDRAM address. DDRAM data is sent and received after this setting	40μS
Busy Flag/ Address Read	0	1	BF	AC							Reads Busy flag(FB) indicating internal operation is being performed and reads address counter contents.	0μS
CGRAM/DDRAM Data Write	1	0	W_{RITE} D_{ATA}								Writes data into DDRAM or CGRAM.	40μS
CGRAM/DDRAM Data Read	1	1	R_{EAD} D_{ATA}								Reads data from DDRAM or CGRAM.	40μS

Code	Description	Execute Time(max.)
I/D=1:Increment I/D=0:Decrement S=1:With display shift S/C=1:Display shift S/C=0:Cursor movement R/L=1:Shift to the right R/L=0:Shift to the left DL=1:8-bit DL=0:4-bit N=1:1/16Duty N=0:1/8Duty, 1/11Duty F=1:5×10dots F=0:5×7dots BF=1:Internal operation is being performed BF=0:Instruction acceptable	DDRAM:Display Data RAM CGRAM:Character Generator RAM ACG :CGRAM Address ADD :DDRAM Address Corresponds to cursor address. AC :Address Counter, used for both DDRAM and CGRAM * :Invalid	fcp or fosc=250kHz However, when frequency changes, execution time also changes Ex. When fcp or fosc=270kHz. $40\mu S \times \frac{250}{270} = 37\mu S$

FONT TABLE(5×11Dots) (5×8Dots)



※CGRAM:Character pattern area can be rewritten by program.

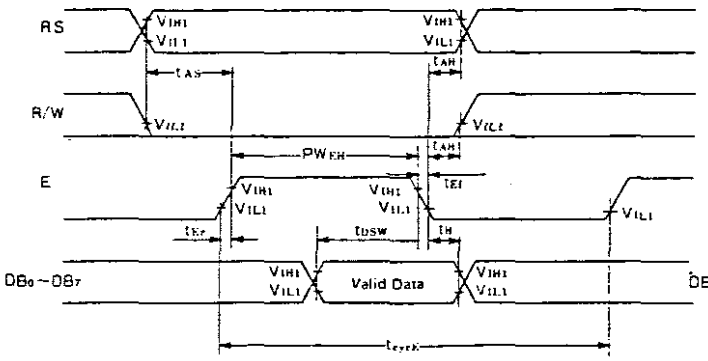
TIMING CHART

Item	Symbol	Measuring Condition	Standard Value			Unit
			min.	typ.	max.	
Enable Cycle Time	T_{CYCE}	Figs.1, 2	1000	—	—	nS
Enable Pulse Width, High Level	PW_{EH}	Figs.1, 2	450	—	—	nS
Enable Rise and Decay Time	t_{Er}, t_{Ef}	Figs.1, 2	—	—	25	nS
Address Setup Time, RS, R/W–E	t_{AS}	Figs.1, 2	140	—	—	nS
Data Delay Time	t_{DDR}	Fig.2	—	—	320	nS
Data Setup Time	t_{DSW}	Fig.1	195	—	—	nS
Data Hold Time	t_H	Fig.1	10	—	—	nS
Data Hold Time	t_{DHR}	Fig.2	20	—	—	nS
Address Hold Time	t_{AH}	Figs.1, 2	10	—	—	nS

※V_{CC}=5.0V±10%, GND=0V, Ta=−20~+75°C

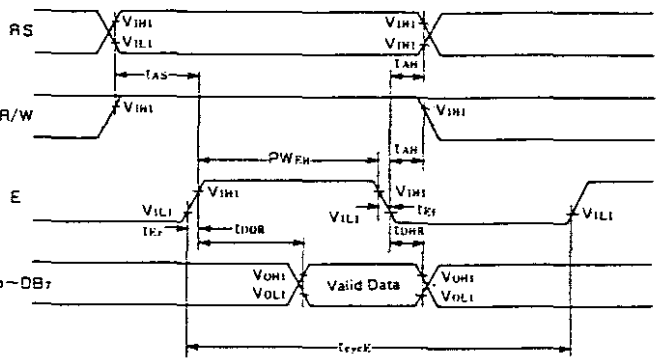
(In case control LSI is HD44780)

FIG. 1 WRITE OPERATION



(Write Data from MPU to MODULE)

FIG. 2 READ OPERATION



(Read Data from MODULE TO MPU)

PIN ASSIGNMENT

Pin No.	Symbol	Level	Function	
1	V_{SS}	—	Power Supply	0V(GND)
2	V_{CC}	—		+5V
3	Vee	—		for Liquid Crystal Drive
4	RS	H/L	Register H:Data Input Select L:Instruction Input	
5	R/W	H/L	H:Data Read(Module→MPU) L:Data Write(Module←MPU)	
6	E	H, H→L	Enable Signal(No pull-up Resistor)	
7	DB0	H/L	Data Bus Line	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

■In the data bus line, data transfer is performed two times by the 4-bit or one time by the 8-bit in order to interface with 4-bit or 8-bit MPU.

■In case interface data length is 4-bit, the data is transferred by using only four buses of DB4−DB7 and the buses of DB0−DB3 are not used. The data transfer to MPU is completed by transferring the data of 4-bits twice. Transfer of upper four bits and low four bits is performed in sequence.

■In case interface data length is 8-bit, data transfer is performed by using eight buses of DB0~DB7.

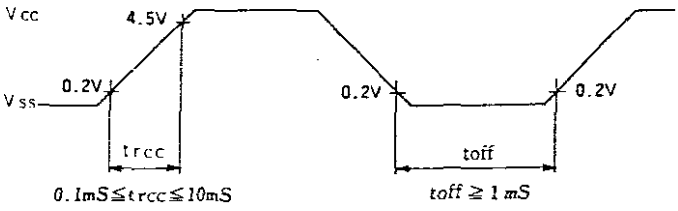
Note:Please refer p. 63–64 for pin assignment of DMC40457 and DMC40401N.

POWER SUPPLY RESET

※In case control LSI is HD44780

The internal reset circuit will not be correctly operated, when the following power supply condition is not satisfied. In this case, please perform initial setting according to the instruction.

Item	Symbol	Measuring Condition	Standard Value			Unit
			min.	typ.	max.	
Power Supply Rise Time	trcc	——	0.1	—	10	mS
Power Supply OFF Time	toff	——	1	—	—	mS



$0.1mS \leqq trcc \leqq 10mS$ $\quad toff \geqq 1\,mS$

Note: The item toff defines the time when the power supply is off, when the power supply shuts down mementarily or repeats on-off state.

RESET FUNCTION

● Initializing by Internal Reset Circuit

The HD44780 automatically initializes (resets) when power is turned on using the internal reset circuit. The following instructions are executed in initialization. The busy flag (BF) is kept in busy state until initialization ends. (BF=1) The busy state is 10ms after Vcc rises to 4.5V.

(1) Display clear

(2) Function ser

DL=1 : 8bit long interface data

DL=0: 4bit F=0 : 5×7dot character font

N=1: 1/16 Duty

N=0: 1/8 Duty, 1/11 Duty

(3) Display ON/OFF control

D=0 : Display OFF C=0 : Cursor OFF B=0 : Blink OFF

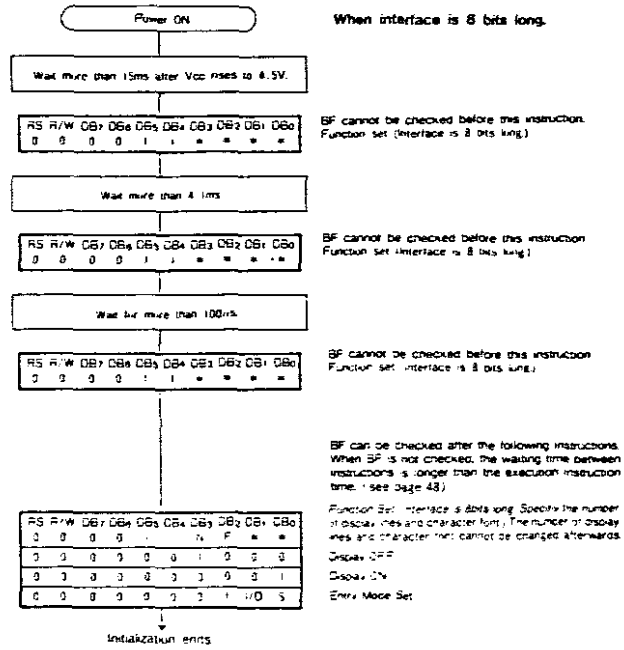
(4) Entry mode set

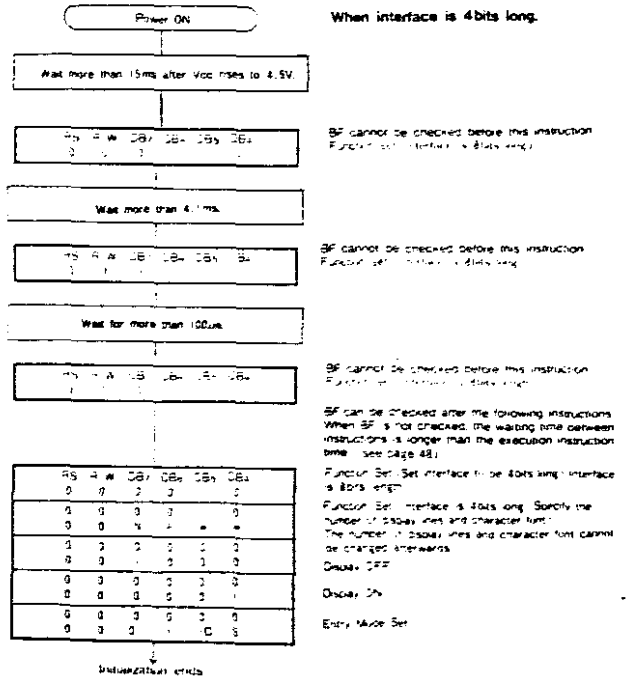
I/D=1 : +1(increment) S=0 : No shift

Note : When conditions in "Power Supply Conditions Using Internal Reset Circuit" are not met, the internal reset circuit with not operate normally and initialization will not be performed. In this case initialize by MPU according to "Initializing by Instruction".

● Initializing by Instruction

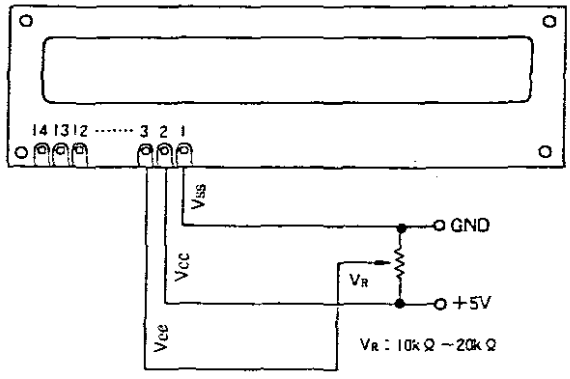
If the power supply conditions for correctly operating the internal reset circuit are not met, initialization by instruction is required. Use the following procedure for initialization.



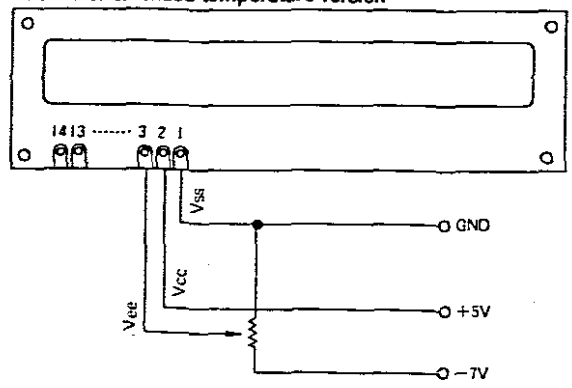


EXAMPLE OF POWER SUPPLY

DMC Module (Standards)



In case of extended temperature version



*NOTE:When the voltage of Vee is different from the recommended voltage, the viewing angle may be changed.

● Examples of Temperature Compensation Circuits for Extended Temp Type.(Only for reference)

(A) 1/8Duty~1/4Bias

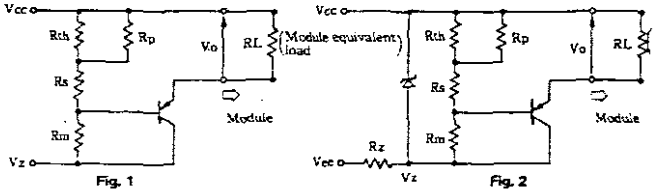
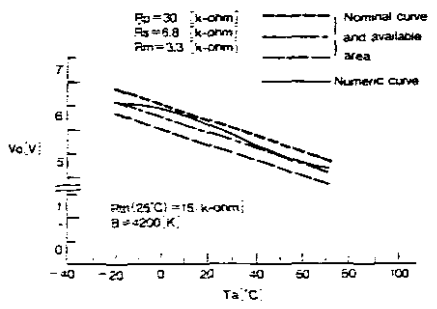


Fig. 1 Fig. 2

Thermistor:Rth(25℃) = 15[k-ohm], B = 4200[K]
Resistors :Rp = 30[k-ohm], Rs = 6.8[k-ohm], Rm = 3.3[k-ohm]
Transistor :PNP Type
Vcc : +5V, Vss : 0v (Logic Supply)
Vz : −8[V] (−7.8 to −8.2[V])
Vee < Vz[V], Rz = (Vz − Vee)/5[k-ohm]



Ta[℃]	Vo[V]
−20	6.56
−10	6.50
0	6.40
10	6.26
20	6.09
30	5.88
40	5.67
50	5.47
60	5.29
70	5.15

(B) 1/16Duty~1/5Bias

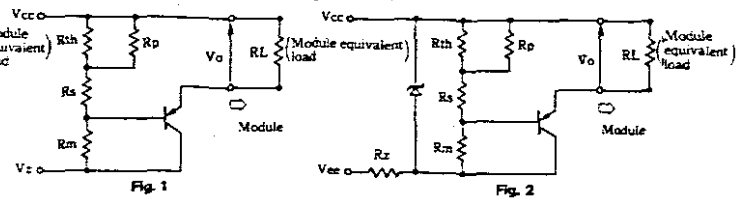
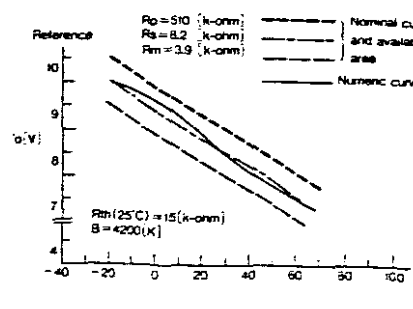


Fig. 1 Fig. 2

Thermistor:Rth(25℃) = 15[k-ohm], B = 4200[K]
Resistors :Rp = 510[k-ohm], Rs = 8.2[k-ohm], Rm = 3.9[k-ohm]
Transistor :PNP Type
Vcc : +5V, Vss : 0v (Logic Supply)
Vz : −11[V] (−10.725 to −11.275[V])
Vee < Vz[V], Rz = (Vz − Vee)/5[k-ohm]



Ta[℃]	Vo[V]
−20	10.01
−10	9.84
0	9.60
10	9.28
20	8.89
30	8.49
40	8.11
50	7.79
60	7.53
70	7.33

*Specifications are subject to change without notice