CHROM-1 AND CHROM-1AT

Keithley MetraByte Corporation

A Subsidiary of Keithley Instruments, Inc.
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

Part Number: 24886

First Printing: May 1986

by

Keithley MetraByte Corporation
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

WARNING

Keithley MetraByte Corporation assumes no liability for damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support or critical applications.

Table of Contents

Section 1

INTRODUCTION

1.1 CROM-1 HARDWARE DESCRIPTION

MetraByte's CROM-1 A/D board for chromatography and precision voltage measurement uses a voltage to frequency (V/F) converter and counter to obtain very high resolution and integral accuracy. The board functions as a true electronic chromatograph integrator. The V/F converter has a scaling of approximately 100,000 counts/sec. for a full scale input and a typical integral linearity of 0.005% (1 part in 20,000). The input ranges of +10v, +5v, +2v and +1v are software selectable and the input is floating, completely isolated from the computer through the use of optical isolators. Also through software, the V/F converter can be switched to either of 2 inputs as well as a ground (zeroing) reference and a +1.0000v calibration reference. This provides a means of performing zero and full scale calibration of the V/F to eliminate temperature and time drifts. A block diagram is shown below in Fig. 1.1

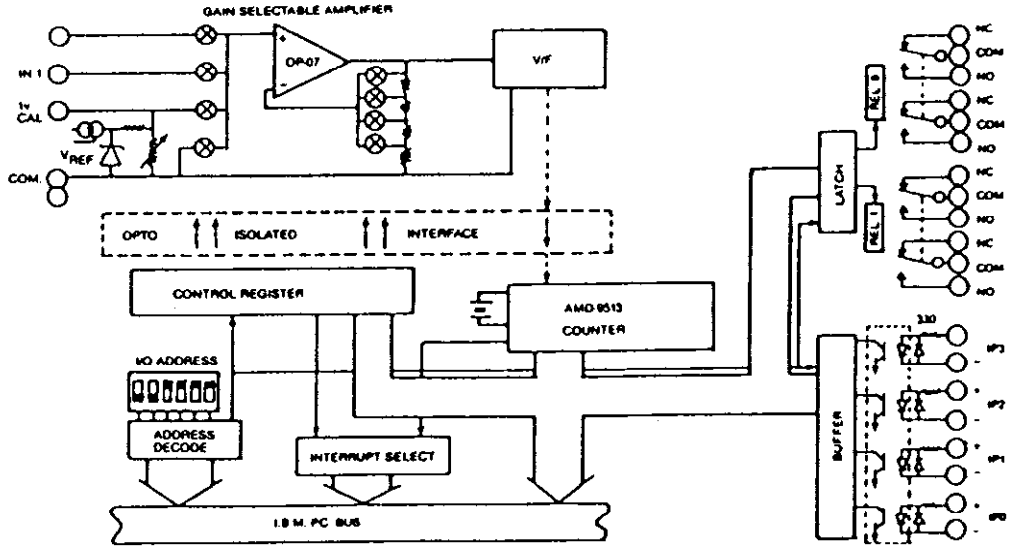


Fig. 1.1 CROM-1 Block Diagram

The CROM-1 includes a complex counter device, Advanced Micro Devices AMD-9513, which contains five 16 bit counters. Counters 1 and 2 are cascaded to form a 32 bit counter that accumulates the pulses from the V/F. When running at 100KHz, a 32 bit counter will accumulate pulses for almost 12 hours before overflowing. For increased counter capacity, it is also possible to cascade counters 3 & 4 to counters 1 & 2 using software commands only, otherwise these counters are normally unused. Counter 5 is used to generate periodic interrupts. Under software control it can be connected to a precision xtal oscillator source and decade scaler providing 1MHz, 100KHz, 10KHz, 1KHz and 100Hz frequencies. Any of these frequencies can be further divided by any integer in the range 2 - 65,535 loaded into counter 5. The output of counter 5 in turn can generate a hardware interrupt on any of the PC expansion bus interrupt levels 2 thru 7. The active interrupt level (2-7) is selected by software. The interrupt service routine can simultaneously latch counters 1 & 2 without disturbing the count in process, so that it is possible to store the counts at equal intervals of time and so measure the voltage during the interval and the integrated volt-seconds between any pair of intervals. This feature is useful for determining the area under the peaks of the chromatogram to very high precision.

The resolution obtained from the CROM-1 is dependent on the integrating interval and range as follows:-

	INTEGRATING INTERVAL			
Range	0.01 sec.	0.1 sec.	1 sec.	10 sec.
1v	1mV	100uV	10uV	1uV
2v	2mV	200uV	20uV	2uV
5v	5mV	500uV	50uV	5uV
10v	10mV	1mV	100uV	10uV

Apart from providing continuous integration of the input signal, the V/F also integrates input noise whether it be 50/60Hz line noise or detector noise. This characteristic is a useful feature dispensing with the need to provide filtering of the input. Line noise will be integrated to zero and completely rejected over any even multiple of the line period e.g. 100 mS or 1 sec. In addition the optically isolated input avoids any generation of errors through ground loops.

Once in the measurement mode, the CROM-1 is essentially a single channel input as the signal is continuously connected to the V/F. A second input is provided on the CROM-1 so that it is possible to share one CROM-1 between two signal sources but **not** at the same time. If you wish to perform simultaneous measurements on two or more channels, you can install several CROM-1 boards in the same computer, limited primarily by the availability of expansion slots. Each CROM-1 needs to be set to a different I/O address selected by the on-board DIP switch.

To facilitate starting and stopping of the chromatograph or other external process synchronization, the board includes 4 opto-isolated digital inputs and 2 double pole double throw relay

outputs. The output contacts are rated at 1A at 28v D.C. or 0.5A at 120v A.C.(resistive). These I/O lines are addressed through separate ports.All the digital I/O connections to the board are electrically isolated from the computer to improve safety and noise performance.

By changing component values, it is also possible to run the V/F at any rate from 10KHz/F.S. to 1MHz/F.S (standard is 100KHz/F.S. - higher frequencies tend to degrade integral linearity). It is also possible to provide other ranges (including low-level) and bipolar (+/-) inputs. For these modifications, please contact MetraByte. Apart from its uses in chromatography, the CROM-1 can be used in many other precision and slow high precision measurement applications e.g. direct interface to a thermocouple etc., our applications engineering staff will be glad to provide information and assistance.

The CROM-1 is a 9 inch "3/4 slot" board suitable for installation in computers with reduced length expansion slots such as the Tandy 1000 as well as standard IBM PC/XT/AT and compatible machines.

1.2 CROM-1 SOFTWARE DESCRIPTION

The CROM-1 can be used with several different types of software. Included in the price and supplied with the board is MetraByte's standard utility package which is directed towards the needs of users who wish to perform their own programming. This includes a BASIC callable machine language driver CROM.BIN, a fully commented assembly language source listing for the driver (CROM.ASM), a programming example EX.BAS and a simple menu driven logging program (CROM.BAS & CROM.EXE). In addition, the object file (CROM.OBJ) is provided to allow the use of compiled BASICA such as the IBM Basic Compiler or Microsoft Quick Basic. This software is documented in this manual.

For the user who wishes to avoid programming, MetraByte offers two optional menu driven data acquisition packages developed for the CROM-1 by Laboratory Technologies Corporation. The first and more expensive option is the general purpose Labtech Notebook that may be supplemented with Labtech Chrom for chromatography analysis, and the second is Labtech CHROM+ which is specifically designed for chromatography analysis only. If you are interested in the capabilities of Labtech Notebook, call or mail MetraByte for a free demonstration disk. The features of these products are detailed in our catalog, a brief guide follows:-

LABTECH NOTEBOOK This is a general purpose data acquisition and control package that may be used with many other MetraByte boards as well as CROM-1. Through user menus, it allows you to control triggering,

sampling, graphing, filing and analysis of data. Analysis is through built in functions including curve fitting and fast Fourier transforms. It also allows acquisition of data while running another program and links to Lotus 1-2-3 for additional analysis and graphing.

LABTECH CHROM — This is an automatic analysis package that is used together with Labtech Notebook. It performs an automatic analysis of the chromatogram peaks, reporting area, retention time and height, displaying, printing or storing this data to disk.

LABTECH CHROM+ — This is a lower cost package for chromatography only. It performs the functions of Labtech Notebook and Labtech Chrom for the CROM-1 but does not include the general data acquisition and analysis capabilities of Labtech Notebook.

Section 2

INSTALLATION

2.1 BACKING UP THE DISK

The software supplied with CROM-1 is in DOS 1.10 double sided (320K) format which can be read by DOS versions 1.1, 2.0, 2.1, 3.0 and 3.1. It is advisable to make a back up copy before using the software, although if for any reason you lose the software, MetraByte will always provide a free replacement. The easiest way to copy the original to any other disk formatted under any other revision of DOS is to insert the disk in your A drive and from DOS enter:-

COPY A:*.* B: (or other destination drive specifier)

2.2 HARDWARE INSTALLATION

The CROM-1 board uses 4 consecutive address locations in I/O space. Some I/O address locations will already be occupied by internal I/O and other peripheral cards, so to provide flexibility in avoiding conflict with these devices the base I/O address can be set by the Base Address D.I.P. switch to be on a 4 bit boundary anywhere in the I.B.M. P.C. decoded I/O space. This I/O address space extends from decimal 512-1023 (Hex 200-3FF) which is many times larger than is ever likely to be fully occupied. Summarising the usual I/O address assignments from data in the "IBM Technical Reference Manual":-

ADDRESS(Hex)	DEVICE	ADDRESS(Hex)	DEVICE
000-1FF	Internal system	378-37F	LPT1:
1F0-1F8	Hard disk (PC/AT)	380-38C	SDLC comm.
200-20F	Game	380-389	Binary comm. 2
210-217	Expansion unit	3A0-3A9	Binary comm. 1
220-24F	Reserved	3B0-3BF	Mono dsp/LPT1:
278-27F	Reserved	3C0-3CF	Reserved
2F0-2F7	LPT2:	3D0-3DF	Color graphics
2F8-2FF	COM2:	3E0-3E7	Reserved
300-31F	Prototype card	3F0-3F7	Floppy disk
320-32F	Hard disk (PC/XT)	3F8-3FF	COM1:

This covers the standard I/O options, but if you have other I/O peripherals e.g. battery backed up clocks, special graphics boards, prototype cards etc. they will also be sharing I/O address space. Memory addressing is separate from I/O addressing so you need not be concerned with a possible conflict with any add-on memory. If multiple CROM-1 boards are installed in the same computer, then each one must be set to a different base I/O address.

The CROM-1 is shipped with its DIP switch set for a base I/O address of Hex 300. This is usually a good default value, and may not need to be altered. If you want to check or change the setting before you install the board in your computer, insert the software disk in your floppy drive and enter:-

A> INSTALL

This runs a self-explanatory program (INSTALL.EXE) that will give you a pictorial view of the base address switch setting on the CROM-1. After entering your choice of address, simply set the switch the way you see it on the screen and press <ESC> to exit back to DOS. You will also see warning messages of settings which could possibly conflict with standard IBM peripheral devices **if you have them installed.** If you receive a warning for a device that is not in your computer, it can safely be ignored. These cautions apply strictly for IBM standard devices (although the same mapping is followed by most compatibles) and may not be totally foolproof as far as non-IBM peripherals are concerned. If your CROM-1 does not appear to work correctly, or interferes in some way with other devices on your computer e.g. disk drives etc. or your computer fails to boot up as normal, remove the CROM-1 and try a different I/O address. Once you have set the base I/O address, make a note of its value as you will need to provide it in the initializing or configuration sections of programs.

To install the CROM-1 inside your computer, start by removing the board from its protective electrostatic packaging. It is a good precaution to discharge any electrostatic charge you may have accumulated by touching the metal frame of your computer just before you plug the board into it. TURN OFF THE POWER on your computer and remove the case (See IBM "Guide to Operations" for your model of computer if you are not already expert at this maneuver). Remove a vacant back plate by undoing the screw at the top and plug the CROM-1 in and then replace the screw and secure the backplate. The CROM-1 will fit in any of the regular full depth slots of the IBM PC/XT/AT or "three-quarter" slots such as those in the Tandy 1000.

Remember, always TURN OFF THE POWER whenever installing or removing any peripheral board. Failing to observe this precaution can cause costly damage to the electronics of your computer and/or the CROM-1 board. If for any reason you later remove the CROM-1 board, MetraByte recommends that you retain the special electrostatically shielded packaging and use it for storage and shipping.

Section 3

CROM-1 HARDWARE

3.1 I/O ADDRESS MAP & REGISTER DATA FORMAT

The information in this section is directed towards the needs of programmers and with the exception of Section 3.2 may be skipped if you are using menu driven software such as Labtech or CHROM+.

The chromatography board uses 4 consecutive I/O addresses. The base I/O address is fully settable by means of a D.I.P. switch on the board (this is the only user settable component - see Installation). If required more than one chromatography board can be used in a single computer provided that their I/O addresses and operating interrupt levels are different. The I/O address map is as follows:-

I/O ADDRESS	FUNCTION	
	Read	Write
Base address + 0	Counter data	Counter data
+ 1	Counter status	Counter control
+ 2	Main control	Main control
+ 3	Aux. inputs	Relay outputs

Each register has functions as follows:-

Counter data & counter control/status

The locations at Base Address +0 & +1 correspond to the standard AMD-9513 address locations. Refer to the 9513 data sheet for information. The 9513 has many internal registers and commands which are addressed indirectly through the 9513 counter control register. A brief guide to the 9513 is given in Appendix A.

Main control register

This is a single byte read/write register at Base Address + 2 that controls selection of the V/F input source and range, hardware interrupt level and interrupt enable:-

Data bus bit

D7	D6 D5 D4	D3 D2	D1 D0
Int. enable	Int. level	Source	Range
0 - disabled	000 - inactive	00 - In 0	00 - 10v
1 - enabled	001 - inactive	01 - In 1	01 - 5v
	010 - 2	10 - 1v cal.	10 - 2v
	011 - 3	11 - zero	11 - 1v
	100 - 4		
	101 - 5		
	110 - 6		
	111 - 7		

Note that the main control register is automatically cleared on reset (power up) of the computer. Also when changing ranges, the zero and full scale should be re-calibrated as these are not consistent from range to range. In operation a range that suits the input signal source should be chosen and subsequent measurements performed on that fixed range. The wide dynamic range of the V/F conversion technique makes gain switching unnecessary.

Digital inputs

This is a 4 bit read only register at Base Address + 3 which returns the state of isolated digital inputs IP0-3:-

Data bus bit

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	IP3	IP2	IP1	IP0

An energized input will return as a logic 1. The input circuit of the optoisolated inputs is shown in Fig. 3.1 Note that the input is polarity sensitive and will respond to voltages in the range of 3 to 12 volts D.C.. Higher input voltages may be handled by adding an external current limiting resistor in series with R to limit the circuit current to 20mA.

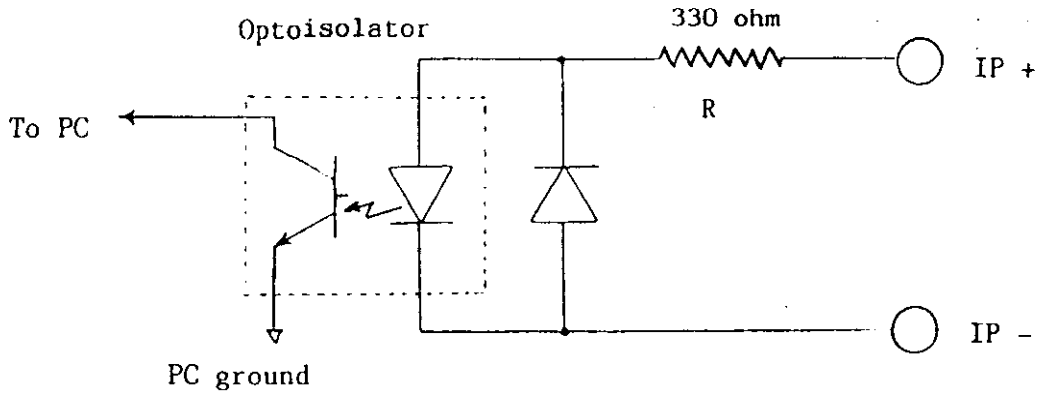


Fig. 3.1 Isolated Logical Input Circuit of IP0-IP3

Relay outputs

The 2 control relays are operated by bits D0 & D1 of a write only register at BASE Address + 3:-

Data bus bit

D7	D6	D5	D4	D3	D2	D1	D0.
x	x	x	x	x	x	REL 1	REL 0

A logic 1 corresponds to an energized relay, bits D2-D7 are irrelevant. The relay register is cleared on reset (power up) de-energizing both relays. The output connections of the relays are shown in Fig. 3.2. Relays are rated at 1A at 28v D.C. and 0.5A at 120v A.C. (resistive).

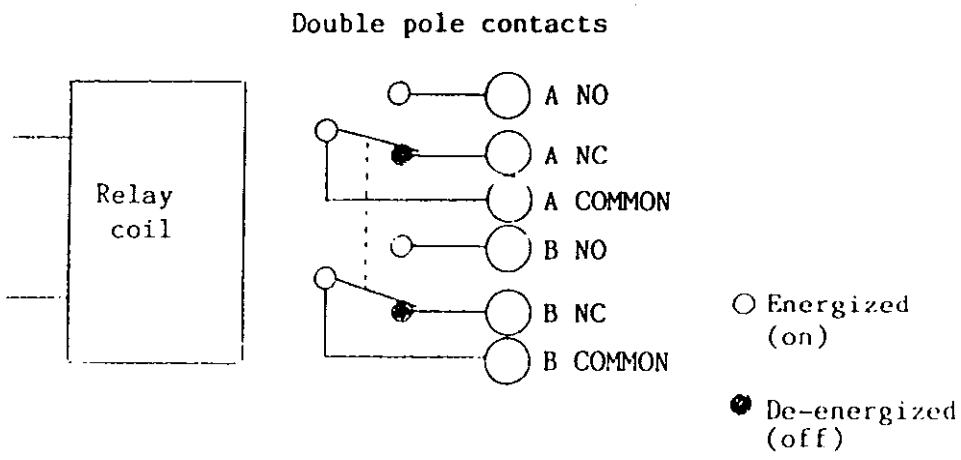
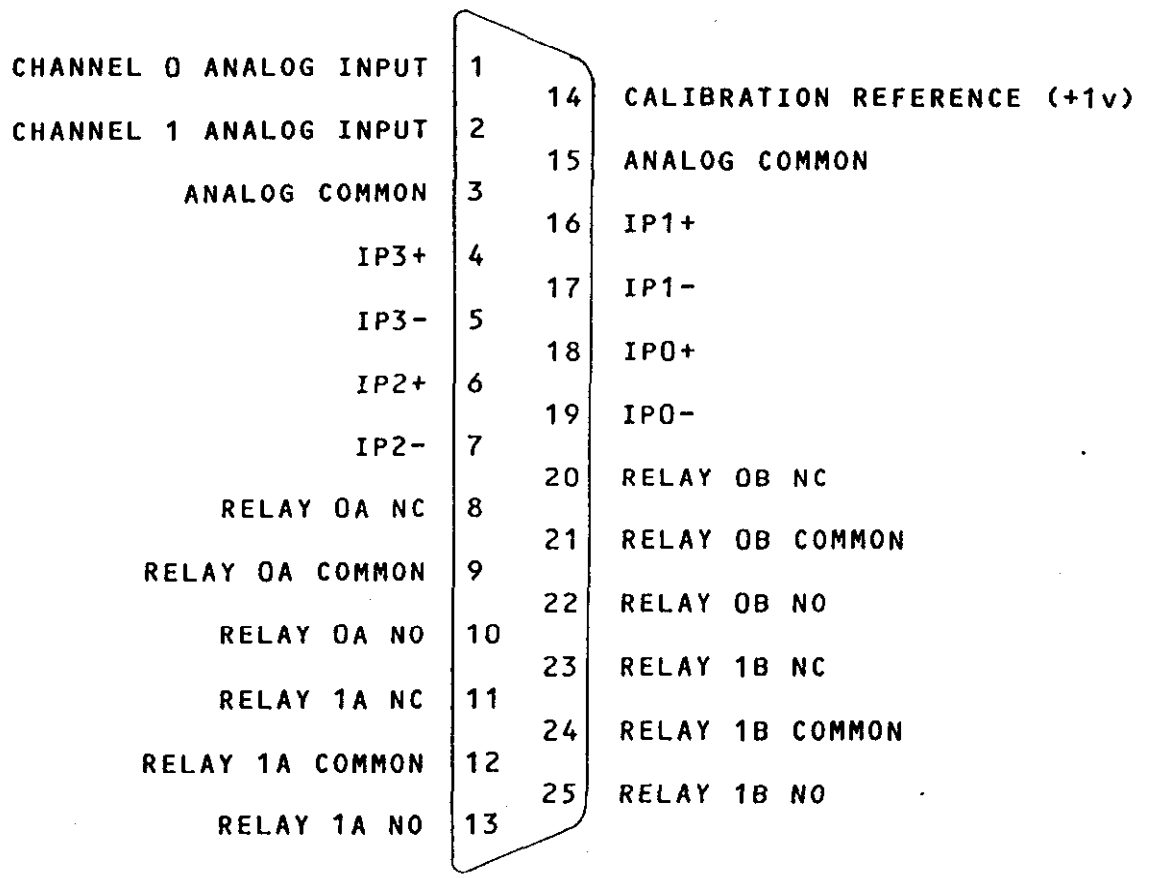


Fig. 3.2 Control Relay Connections

3.2 CONNECTING UP CROM-1

All connections are made to a 25 pin D type female connector that projects through the back mounting plate of the board. A 25 pin D type male connector should be used to make connections (MetraByte part number SMC-25). The pin assignments are as follows:-



NO = Normally open (de-energized)
NC = Normally closed (de-energized)

Fig. 3.3 CROM-1 Connector (Rear View)

Two wires are usually all that is required to connect to your signal source. In a noisy environment, shielded coaxial cable is recommended wired as shown in Fig. 3.4. If a compatible screw terminal connector board is required, use MetraByte model STA-U with K-1800 cable.

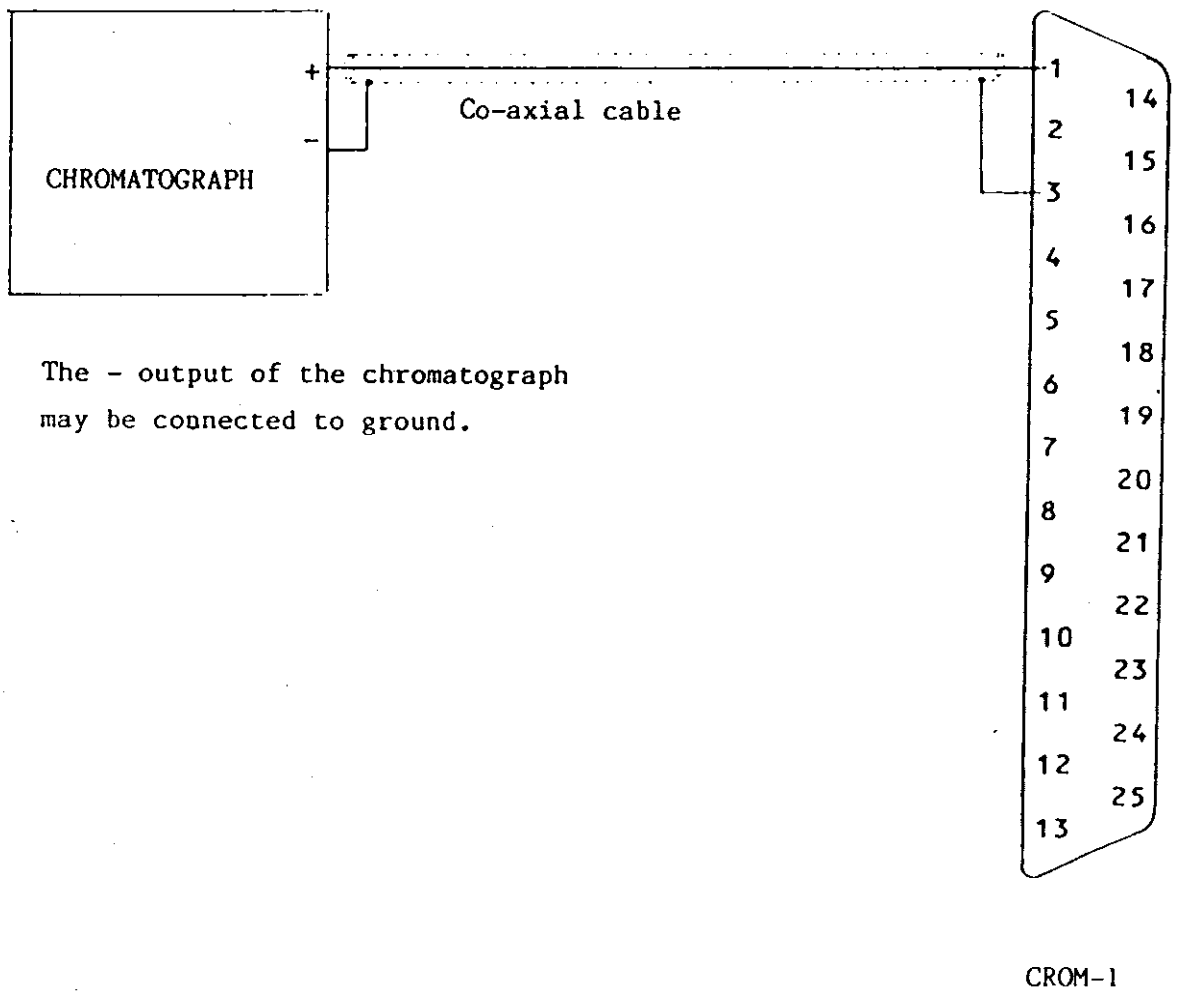


Fig. 3.4 Analog Signal Source Wiring

Section 4

PROGRAMMING

4.1 PROGRAMMING CROM-1

This section provides information to programmers who wish to use the MetraByte utility software supplied with CROM-1. If you intend to operate with a high level menu driven data acquisition package such as Labtech Notebook or CHROM+, you can skip this section and refer to the instructions included in the Labtech documentation instead.

At the lowest level, CROM-1 is programmed using I/O input and output instructions such as BASIC's INP (port) and OUT port,Y functions or assembly language IN AL,DX and OUT DX,AL. Most other languages (with the exception of Fortran) have equivalent instructions. For Microsoft Fortran, MetraByte can supply a library that includes both I/O and memory access routines.

Programming directly with I/O commands involves formatting data and dealing with absolute I/O addresses. Although not demanding, this can require many lines of code and necessitates an understanding of the devices, data format and architecture of the CROM-1. Also many languages, such as BASIC, do not support the writing of interrupt routines, and this is a major obstacle as interrupts are very effectively utilised in the operation of the CROM-1. To simplify program generation in BASIC, a special I/O driver routine "CROM.BIN" is included in the CROM-1 software package. Apart from providing an interrupt service routine, the driver also includes a 10,000 point FIFO (first in - first out) buffer to hold data which may then be accessed asynchronously by a BASIC program using a single line CALL statement. This lets you process data in the program e.g. graph, file, analyze or print it at any speed without affecting the sampling operation of the CROM-1. Each time the CALL is entered, one data item is removed from the buffer for processing by the program. On the first entry to the CALL, the driver also looks after initialization of the CROM-1 (this is described in more detail in Section 4.5).

The object file of the driver, CROM.OBJ is included for linking when compiling your BASICA program using the IBM Basic Compiler or Microsoft Quick Basic (see Section 4.10). Also for assembly language programmers and those wishing to interface to other

languages, the fully commented source code, CROM.ASM, of the driver is provided (see Section 4.12). This can be modified and re-assembled as required, though MetraBytecan only support inquiries concerning operation of the driver in its unmodified form.

4.2 LOADING THE MACHINE LANGUAGE CALL ROUTINE "CROM.BIN"

In order to make use of the CALL routine "CROM.BIN", it must first be loaded into memory. You must avoid loading it over any part of memory that is being used by the main body of your program or DOS or programs which use high memory such as "RAM disks". If you do collide with another program, your computer will usually hang up although sometimes the results can be more peculiar. Often you will need to turn the power off and restore it to re-boot the machine, the usual Ctrl-Alt-Delete reset may fail to do anything. This may sound ominous, but apart from the frustration, no damage will ever result!

Since CROM.BIN uses about 41 Kbytes of memory (40K is the FIFO buffer), it is best loaded outside BASIC's workspace. A typical loading sequence is as follows:-

```
xx100 DEF SEG = &H3000   'segment of memory to load link
                          (choose an empty area e.g. @ 192K)
xx110 BLOAD "CROM.BIN",0  'load driver
 . .
Continue program
```

The above initializing steps will be the same for any interpreted BASIC program. A more comprehensive example is provided on the disk as EX.BAS. Note that the DEF SEG = &H3000 statement in line 100 specifies the load location for the CROM.BIN driver. All subsequent CALL's will occur to the last DEF SEG address, so if you add other DEF SEG's in your program, remember to precede your CALL's to CROM-1 with the same DEF SEG = &H3000 that you used to load the link (see CALL and DEF SEG in your BASIC reference manual). Finding a place to load CROM.BIN is seldom much of a problem now that most PC's are equipped with at least 256K of memory. The following explanation provides some insight into the process of choosing a memory location for the driver and what to do if memory is in short supply.

DOS occupies the bottom of memory, the amount of memory required being dependent on the version (it grows as each new revision adds extra features!). The simplified memory map in Fig 4.1 overleaf shows what happens after booting up BASICA.

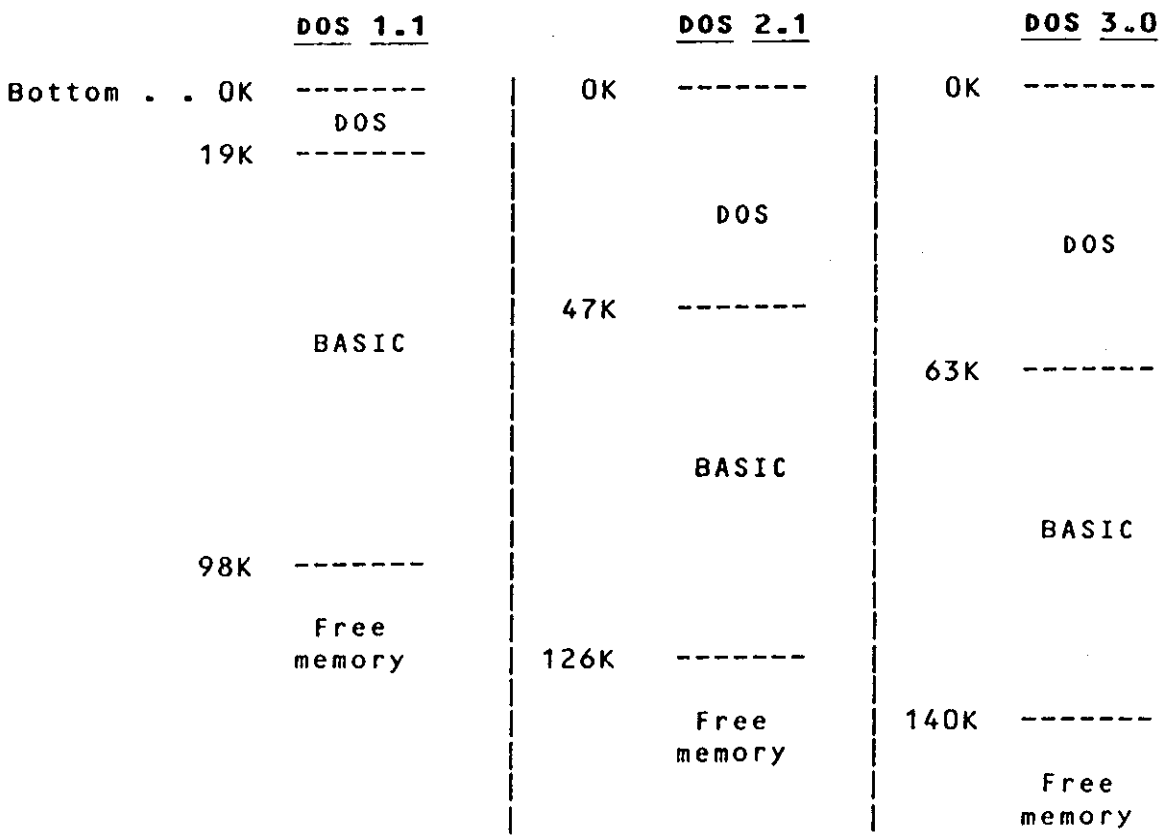


Fig. 4.1 Memory map with DOS and BASIC installed

CROM.BIN should be loaded somewhere in the free memory area so that it does not interfere with either BASIC or DOS. This would be above 98K (&H1880) for DOS 1.1, 126K (&H1F80) for DOS 2.1 or 140K (&H2300) for DOS 3.0. If you have 256K (&H4000) or more of memory, then loading the link at DEF SEG = &H2800 or &H3000 is a good solution for all versions of DOS. One further small detail is that if you are using a PC compatible which does not have BASIC in ROM like the IBM, then BASIC is usually loaded as an .EXE file from the top of memory down. This is likely to fill up to 64K of the top segment of memory. Some virtual disks or print spoolers will do the same. Also if you are accustomed to using DOS resident programs such as Borland's Sidekick etc. be aware that these will push the loading floor of BASIC up and require a compensating increase in the location of CROM.BIN.

If you are memory limited, or you have so much resident stuff that there is no longer 64K left for BASIC to load in, then BASIC will attempt to make the most of what it can find. Instead of getting the message when BASIC has loaded:-

```
The IBM Personal Computer Basic
Version A2.0 Copyright IBM Corp. 1981, 1982, 1983
60865 Bytes free
Ok
```

You may only get 49345 bytes free (or something less than 60000 bytes) for example. In this case make a note of what space BASIC has found. You can then contract this space further using the CLEAR function and load the link at the end of BASIC. This is more complicated, but just as effective.

Let's suppose we get the message 52000 bytes free. CROM.BIN will use 41K bytes, which does not leave much for BASIC's workspace (11K) but it may in some cases get us by. Our initializing code would now be:-

```
xx100 CLEAR, 11000          'contracts BASIC workspace
```

Next we need to find out where BASIC has loaded in memory, add 11000 to it and load CROM.BIN just after the end of BASIC workspace. Memory locations &H510 and &H511 always contain BASIC's load segment:-

```
xx110 DEF SEG = 0           'set up to read &H510 and &H511
xx120 LS = 256*PEEK(&H511)+PEEK(&H510)  'load segment
xx130 SG = LS + 11000/16    'remember segment addresses are on
                             16 byte (paragraph) boundaries
xx140 DEF SEG = SG          'set up to load link
xx150 BLOAD "CROM.BIN",0    'load link
 . .
Proceed with your program as before
```

As CROM.BIN is a large file, the advantages of contracting BASIC's workspace are minimal. Generally the simplest recourse is to add more memory to your computer, an alternative is to reduce the size of the FIFO buffer by re-assembling CROM.ASM.

4.3 STRUCTURE OF THE CALL STATEMENT

If you are new to using CALL statements, this explanation may assist you in understanding how the CALL transfers execution to the machine language (binary) driver routine (also see CALL in your Basic Reference Manual). Prior to entering the CALL, the DEF SEG = SG statement sets the segment address at which the CALL subroutine is located. The CALL statement for the CROM.BIN driver must be of the form:-

```
xxxxx  CALL CROM (A, BUF%, CTRL%(0), FLAG%)
```

Let us examine the parameters after CALL one by one:-

CROM - In interpreted BASIC this is a variable that specifies the offset of the start of our routine from the segment defined in the last DEF SEG statement. In our case its

value is always set to zero (CROM = 0).
In compiled BASIC (and most other compiled languages) CROM has a different significance - it is the name of the external routine that the linker will look for.

A - This is a single precision (4 byte) real variable, that passes one item of count data from the FIFO buffer.

BUF% - This is an integer variable that returns the number of valid data items in the buffer. When BUF% = 0, the buffer is empty (variable A will also return zero).

CTRL%(7) - This is a 6 element integer array that specifies the operating conditions of the CROM-1 as follows:-

CTRL%(0) - selects input channel
0 = CH0
1 = CH1
2 = Calibrate
3 = Zero

CTRL%(1) - selects full scale range
0 = 10v
1 = 5v
2 = 2v
3 = 1v

CTRL%(2) - selects interrupt rate in mS
e.g. CTRL%(2) = 1000
generates an interrupt every second

CTRL%(3) - Specifies number of samples required

CTRL%(4) - selects interrupt level, value 2-7.
If CTRL%(3) <> 2 thru 7= 0, interrupts will be disabled.

CTRL%(5) - specifies Base I/O address (as set on DIP switch) e.g. CTRL%(4) = &H300

CTRL%(6) - returns interrupt status
1 = interrupts active
0 = interrupts disabled

FLAG% - Returns error code if any of the specifying CTRL%(*) are out of range (see Section 4.7)

The four variables within brackets are known as the CALL parameters. On executing the CALL, the addresses of the variables (pointers) are passed in the sequence written to BASIC's stack. The CALL routine unloads these pointers from the stack and uses them to locate the variables in BASIC's data space so data can be exchanged with them. Three important format requirements must be met:-

1. - The CALL parameters are positional. The subroutine knows nothing of the names of the variables, just their locations from the order of their pointers on the stack. If you write:-

   xxxxx CALL CROM (BUF%, A, CTRL%(0), FLAG%)

   you will mix up the CALL routine, since it will interpret BUF% as the count data, and A as the buffer data etc. Also the variables will now be the wrong types which is more serious. **The parameters must always be written in the correct order:-**

   (count data, buffer, control data, errors)

2. - The CALL routine expects its parameters to be of correct type and will write and read to the variables on this assumption:-

   (real, integer, integer array, integer)

   If you slip up and use the wrong variable types in the CALL parameters, the routine will not function correctly and may hang up the program.

3. - You **cannot** perform any arithmetic functions within the parameter list brackets of the CALL statement. There can only be a list of variables. Also you are not allowed to replace variables by constants.

Apart from these restrictions, you can name the variables what you want, the names in the examples are just convenient mnemonics. Strictly, you should declare the variables before executing the CALL. If you do not, the simple variables will be declared by default on execution, but array variable obviously cannot be dimensioned by default and must be dimensioned before the CALL to pass data correctly if used as a CALL parameter. In the case of the integer array, the first element CTRL%(0) should be specified as the data variable so that the CALL routine can locate all other data items in the array correctly.

4.4 INTERRUPTS

Variable CTRL%(4) specifies the hardware interrupt level that CROM-1 will use on the expansion bus to generate interrupts. The PC's 8259 interrupt controller can prioritize 8 different hardware interrupts. Level 0 is the highest priority and is used by the internal timer which generates an interrupt about 18 times/sec. This is used by the BIOS and DOS to provide the system time and date. Level 1 is used by the keyboard to signal that a key has been

pressed and invoke a keyboard handling routine. Both levels 0 and 1 are internal to the PC and not available on the expansion bus. A few of the remaining levels 2 thru 7 are usually available. The interrupt levels are assigned to the standard peripherals as follows:-

Level 2: Reserved (used for cascade input on PC/AT only)
Level 3: Used by COM2: serial port if installed
Level 4: Used by COM1: serial port if installed
Level 5: Used by LPT2: or hard disk if installed
Level 6: Used by floppy disk drive adapter
Level 7: Used by LPT1:

On standard PC and PC/XT models any of levels 2 thru 5 are good choices if the corresponding peripheral device is not installed e.g. if you have no second serial port, you can use level 3. The PC/AT has an expanded interrupt structure and level 2 is used to cascade a second 8259 interrupt controller to provide an additional 8 levels, however these are only available on the 16 bit portion of the PC/AT expansion bus connectors and are not available to CROM-1. For PC/AT's, this narrows down the choice to levels 3 thru 5.

The higher level interrupts will get serviced ahead of the lower levels. If no higher level interrupt is pending and interrupts are enabled, an interrupt will normally be serviced within a few microseconds of its generation. If a higher level interrupt collides with one from the CROM-1, it can delay servicing of the CROM-1 interrupt. The main culprit here is the timer interrupt on level 0, it can occasionally delay the CROM-1 interrupt by 30-40 microseconds, enough to increase the count by 2 or 3 count/sec.s at full scale. The next interval will be shorter by an equal amount, so that over a long period it has no effect, but from interval to interval the latency or variation in delay in servicing the interrupt can introduce some small variation in the latching interval. This is not usually a significant problem, and for ultimate precision the jitter can be practically eliminated by disabling the timer interrupt through the 8259 mask register, and refraining from using the keyboard or COM: ports while gathering data. Usually this is unnecessary as the error introduced amounts to 0.003% or less.

Counter 5 on the CROM-1 is initialized with a 1KHz clock input and divides by the integer specified in CTRL%(2). This generates a precise periodic interrupt when bit 7 of the CROM-1 control register is set. The interrupt service routine in the driver latches counters 1 & 2, changes the 32 bit integer data to a 4 byte single precision floating point real and places the data on the top of the FIFO buffer. This continues until interrupts are disabled by one of the 3 following conditions:-

1. The number of samples specified in CTRL%(3) is reached.

or:- 2. The buffer fills to capacity (9,999 data items).

or:- 3. You abort operation by entering the CALL with CTRL%(4) = 0.

CTRL%(6) provides information on the status of the interrupts. As long as interrupts are active, CTRL%(6) is set to 1. As soon as any of the above termination conditions are met, it is set to 0 to indicate that interrupts have been disabled. Even though the interrupt is disabled and further data sampling has ceased, you may continue unloading the FIFO buffer by accessing the CALL. This action is demonstrated in EX.BAS.

4.5 INITIALIZATION

The first time the CALL is executed, no data is returned, but the CROM-1 is initialized. Initialiazing performs the following operations:-

1. Sets the master mode register of the AMD9513A counter
2. Sets counter 1, 2 and 5 mode registers
3. Zeroes (resets) counters 1 & 2
4. Loads counter 5 for specified interrupt rate
5. Installs interrupt handler
6. Enables interrupt (if CTRL%(4) = 0, disables interrupt)

Subsequent entries to the CALL with the setup parameters, i.e. CTRL%(0 thru 5), unchanged will return data from the FIFO buffer in variable A. If however any of the parameters in CTRL%(0 thru 5) are altered, the driver will perform a re-initialization of CROM-1 on a subsequent CALL and also clear the buffer.

4.6 FIFO BUFFER OPERATION

In operation, counter 5 generates a periodic interrupt. The interrupt service routine performs the following operations:-

1. Latches counters 1 & 2 "on the fly". The counters are latched at the same instant, this does not disturb the count in process.

2. Reads the 32 bit integer from the latches, turns the data into 4 byte floating point single precision format, and puts it on top of the FIFO buffer, incrementing the top of buffer pointer.

Entering the CALL takes the data item from the bottom of the buffer and returns the accumulated count in variable A. The bottom of buffer

pointer is incremented. BUF% returns the difference between the pointers and indicates the amount of data left in the buffer. The buffer is circular and can contain up to 10,000 data items. If you were sampling at 1 interrupt/second this would be sufficient to hold nearly 3 hours of data, or 16 minutes at 10 samples/sec. In practice the buffer is being continually unloaded and data processed, although you may not be able to keep up with the rate at which data is being acquired. For example, if you are filling the buffer at 10 samples/sec and unloading and processing data at 3 samples/sec. the buffer is effectively being filled at 7 samples/sec. and will take nearly 24 minutes to overflow.

If the buffer becomes completely filled, further interrupts are disabled so that data in the buffer is conserved. This salvages as much data as possible before suspension of operation, but it is your responsibility to avoid this condition by correct choice of sample rate, duration and simultaneous processing demands.

If the buffer is empty, BUF% will be returned zero (and A will also be zero) indicating that there is no available data. A simple routine method of sensing the presence of data and continuing with your program is:-

```
xxx00 CALL CROM (A, BUF%, CTRL%(0), FLAG%)
xxx10 IF BUF% = 0 AND CTRL%(6) = 0 THEN END  'all data read out
xxx20 IF BUF% = 0 GOTO xxx00  'wait for data to be added to FIFO
 . .
 . .   (process data any way you want)
 . .
yyyyy GOTO xxx00   'get next data point
```

Note that the value in A is the cumulative count. To obtain the change in count in a sampling interval between interrupts, the previous value of A must be subtracted from it.

Changing the capacity of the buffer can only be accomplished by re-assembling CROM.ASM. Change the LEN_BUF equate to 4 times the number of data items desired (4 bytes per item). The driver is designed to operate within a 64K segment, and the buffer length cannot exceed 15,000 items without major modifications to the driver.

4.7 ERROR CODES

Some value checking is performed on entry data and returned in FLAG%. This stops you setting up the CALL with interrupt level 9, sample rate -6 etc. or other incorrect parameters. It is not possible to start making readings if an error condition exists, so you should check for errors after initializing.

Error code #	Problem
1	CTRL%(0) - channel data, <0 or >3
2	CTRL%(1) - range data, <0 or >3
3	CTRL%(2) - interrupt rate <4mS
4	CTRL%(4) - interrupt level, <2 or >7 (0 terminates interruptsif active)
5	CTRL%(5) - base address, <&H100 or >&H3FC

Checking for errors is easily performed after a CALL:-

```
xxx00 CALL CROM (A, BUF%, CTRL%(0))

xxx10 IF CTRL%(5)<>0 THEN PRINT "Error number ";CTRL%(5):STOP
```

4.8 ZEROING AND CALIBRATION

V/F converters exhibit zero and gain drift with time and temperature. To minimise this problem, the CROM-1 can be internally connected to zero and 1v calibration references. The V/F is slightly offset at its zero, so that instead of producing a zero count for zero volts in, it will return about 2000 - 3000 counts/sec. (about 2% of full scale). The count returned for 1v input will depend on the range selected, on a 1v range where this corresponds to full scale, about 90,000 - 110,000 counts/sec. is typical.

Performing a zero/calibration before starting to sample data will return the calibration constant in counts per volt per second. This calibration step corrects for all gain and zero dritts. Over a long measurement run, it can also be repeated at the end of sampling and the start and end values averaged. The data must then be post processed to correct it. Usually this more elaborate method is unnecessary and only justified over very long sampling durations. An example of a pre-measurement calibration routine is in EX.BAS.

4.9 EXAMPLE BASIC PROGRAMS

Two example BASICA programs are provided on the disk. The first EX.BAS is a simple program to illustrate the operation of the CALL. It is commented and can be listed to provide further information. It shows you how to load CROM.BIN, initialize the

CROM-1, auto-calibrate CROM-1 and then read out data for 6 seconds at 10 samples/sec. It also illustrates the action of the FIFO buffer. In practice, sampling would take place for a much longer duration, this example is of short duration to demonstrate how things work - you can easily alter it.

The second program CROM.BAS is a more complicated BASICA program that performs most of the tasks that interfacing to a chromatograph will require in the form of a menu driven program. It will optionally, generate an ASCII comma delimited .PRN data file (suitable for Lotus 1-2-3, use IMPORT/NUMBERS), and plot the data on the screen as it is acquired. A faster running version has been compiled with Microsoft Quick Basic, this is CROM.EXE. The source, CROM.BAS, can be modified to requirements and used as an interpreted program or re-compiled. This program and its operation are more fully described in Appendix A.

4.10 COMPILING A BASIC PROGRAM

After you have your program running in interpreted BASIC, you may find that you need to increase its execution speed. Interpreted BASIC requires several milliseconds to execute each statement, and a sizable program particularly with extensive analysis or data processing loops can take a good part of a second to execute. One of the simplest ways to speed things up is to compile your program using the IBM Basic Compiler, or Microsoft Basic Compiler or Quick Basic Compiler. This will usually lead to an improvement in speed by a factor of 3 to 30 times depending on the type of operations in the program.

Before you charge into compiling your measurement program, spend some time familiarizing yourself with the operation and capabilities of the Basic Compiler, try out a few examples and get used to some of the differences from interpreted BASIC e.g. you must declare all arrays prior to use etc. When you are ready proceed as follows:-

1. Take your interpreted BASIC program and delete all lines that load CROM.BIN e.g. the DEF SEG and BLOAD.

2. Store your program in ASCII form (SAVE "yourprog.bas",A).

3. Compile it (BASCOM yourprog) with whatever compiler switches you require. Fix any errors before proceeding.

4. Next run the linker (it is best to use the one supplied with your compiler rather than DOS).

```
A> LINK yourprog.obj + CROM.OBJ

IBM Personal Computer Linker
Version 1.10 (C)Copyright IBM Corp. 1982

Run File [MYPROG.EXE]
List File [NULL.MAP]
Libraries [.LIB]
```

At the end of the linking session, you will have a DOS executable file MYPROG.EXE.

If you receive an "Unresolved External" error message from the linker, it is most likely that you have either omitted or misnamed the CROM.OBJ declaration when prompted for the files to link, or that the calls do not read CALL CROM (you must use the name CROM in the call).

4.11 MULTIPLE CROM-1's IN ONE SYSTEM

What if you wish to operate more than one CROM-1 in a system? To avoid conflicts, each CROM-1 must have a different base address and for simultaneous operation be connected to a different interrupt level. Each board must also be assigned its own CALL routine. To do this start by loading the CROM.BIN routine at different locations in memory:-

```
xxx10 SG1 = &H3000
xxx20 SG2 = &H4000
xxx30 DEF SEG = SG1
xxx40 BLOAD "CROM.BIN",0
xxx50 DEF SEG = SG2
xxx60 BLOAD "CROM.BIN",0
```

Now the CALL appropriate to each board can be entered as required. Note that each CALL is preceded by a DEF SEG appropriate to that board:-

```
yyy10 DEF SEG = SG1
yyy20 CALL CROM (A, BUFA%, CTRLA%(0))
yyy30 DEF SEG = SG2
yyy40 CALL CROM (B, BUFB%, CTRLB%(0))  'etc.
```

4.12 ASSEMBLY LANGUAGE PROGRAMMING

The source code, CROM.ASM, for the driver is also supplied on disk. This can be printed out (use the Ctrl-PrtSc and TYPE CROM.ASM) or examined and altered with any text editor e.g. Wordstar. It is heavily commented and self explanatory.

If you wish to link this with another assembly language program, you will need to modify the entry and exit modules which are designed for the CALL procedure used with BASIC. The initializing, interrupt, and FIFO buffer handling routines are self-contained and communicate only with the local variables defined in the source, making it easy to lift them out unchanged.

Interfacing to calls from C, Pascal and Fortran can also be accomplished by altering the entry and exit routines as these languages usually pass pointers consisting of segment:offset addresses on the stack (unlike BASIC which passes offsets only).

The steps involved in creating a BASIC loadable .BIN driver are described in HOWTO.DOC. Basic tacks a header onto a BLOADable file, the method of using the linker, EXE2BIN, debug and the Basic interpreter achieve this are detailed in this file. Use TYPE HOWTO.DOC and Ctrl-NumLock to read it or list on the printer with Ctrl-PrtSc.

Section 5

CALIBRATION

5.1 CALIBRATION

Since the CROM-1 is normally autocalibrated to eliminate zero and gain drifts, there is only one user adjustment for calibration. On the top of the board is a single multi-turn trimpot (marked R14) that adjust the internal 1v calibration reference. To calibrate this reference, connect a digital voltmeter from the calibration reference output (pin 14 of the rear connector) to analog common (pin 3). Simply adjust the trimpot for a reading of +1.0000v. After calibrating the internal reference, EX.BAS or CROM.EXE can be run to check operation.

5.2 USER REPLACEABLE PARTS

Some of the components that interface to the outside world through the rear connector are socketed so that you can replace them if damaged by overloads, static etc. The board layout is shown in Fig. 5.1.

Digital input lines, IP0-IP3, pass through a quad optoisolator marked U11. To test digital inputs, use DEBUG or Basic and read the input port at Base Address + 3 e.g. PRINT INP(BASE +3). With the inputs the following readings should be obtained:-

Inputs	Reading
All off	0
IP0 on	1
IP1 on	2
IP2 on	4
IP3 on	8

Combinations of inputs on will produce the sum corresponding to the energized inputs e.g. IP1 and IP3 on will return 10 etc. If there is a fault and none of the inputs appear to work right, check that the base address setting of the board is correct. If so, or if one

or more of the inputs are blown, replace U11 with a Siemens ILQ-2 or ILQ-74 opto-isolator.

If any problems are encountered with the analog inputs, CH0 or CH1, you can replace solid state multiplexer U8 (Harris Semiconductor HI3-508A-5. Likewise if the V/F (Analog Devices AD650JN) or counter (Advanced Micro Devices AM9513APC) give problems they can be replaced.

The output relays (Fujitsu type FBR244D005/02CS) can be damaged by overload of the contacts. They are soldered into the board but can be replaced if carefully removed. The relays can be tested by performing an OUT BASE+3, 1 or 2 using Basic or Debug.

If you prefer to have MetraByte service the board, simply call Customer Service at (617)-880-3000.

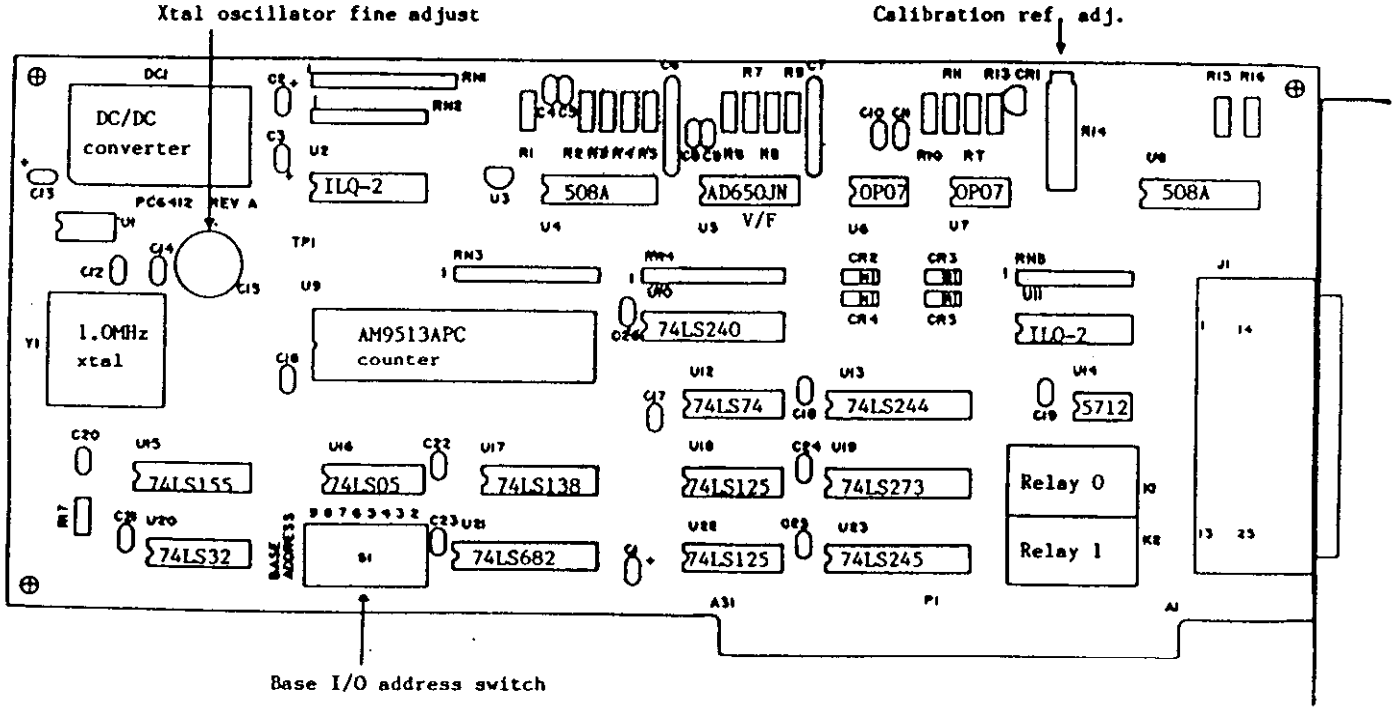


Fig. 5.1 CROM-1 Board Layout

Section 6

SPECIFICATIONS

6.1 ELECTRICAL

+5v power consumption	:	900mA typical / 1A max. (with relays energized)
-5v, +12v, -12v power	:	Not used
Input ranges	:	1v, 2v, 5v & 10v unipolar software selectable. Contact MetraByte for other ranges.
Input impedance	:	100Megohm min.
Input bias current	:	+/- 6nA max
Input channels:	:	2
Input common mode voltage range	:	300v min. analog common to computer frame.
Input overvoltage	:	30v continuous channel input to analog common.
Resolution	:	Controlled by sampling time 1 part in 100,000 per second. typ.
Minimum resolvable signal input	:	1 microvolt (1v range, 10 second sample)
Linearity	:	0.005% typ., 0.02% max.
Scaling	:	100,000 (+/-20%) counts/sec for full scale input.
Calibration reference stabilty	:	20 ppm/deg. C 0.002% per year.

Calibration reference voltage	:	+1.0000 v
I/O address	:	Can be set on any 4 bit boundary from Hex 0 to Hex 3FC
PC bus loading	:	1 74LS TTL load on all inputs
Interrupts	:	Software selectable on any level 2-7 (only active when interrupts enabled, tri-state otherwise)
User adjustments & calibration	:	1 - calibration reference (recommended calibration interval 3 - 6 months)

6.2 DIGITAL INPUTS

Number	:	4
Type	:	Opto-isolated, 3-12v non-latched Approx 330 ohm input resistance

6.3 RELAY OUTPUTS

Number	:	2
Contacts	:	Double pole, double throw 2 Form C, gold overlay silver
Contact rating:	:	1A at 28v D.C. 0.5A at 120v A.C. resistive
Life	:	Mechanical 10,000,000 operations min. Electrical 100,000 operations @ full load.
Operate/release time	:	10 milliseconds max.

6.4 MECHANICAL & ENVIRONMENTAL

Card size	:	9" long x 3.9" high
Weight	:	6.5 oz. (185 gm)
Operating temperature range	:	0 to 50 deg. C.
Storage temperature range	:	-40 to 100 deg. C.
Humidity	:	0 - 90% non-condensing

Appendix A

USING CROM.EXE

A.1 CROM.EXE DESCRIPTION OF OPERATION

CROM.EXE is a compiled BASIC program that handles data acquisition with the CROM-1 in a menu driven form. To run CROM.EXE, simply type CROM after the DOS prompt. After loading has completed, you will see a menu with the currently selected option highlighted:-:-

RUN CONFIG EXIT

Use the cursor keys and Enter to change the option, or simply enter the first letter, R, C or E to run the option.

On the first use, select the CONFIG option. This will display a secondary menu that controls the configuration of the CROM-1 as follows:-

Sample rate - readings per second: .1 .2 .5 1 2 5 10

Run duration: 1

Input channel: CH0 CH1

Full scale range: 1v 2v 5v 10v

Trigger code to start on (0-15): 0

Relay 0 start: off on

Relay 0 end: off on

Relay 1 start: off on

Relay 1 end: off on

CROM-1 board I/O address: &H300

CROM-1 board interrupt level (2-7): 2

Use the cursor keys and/or enter to select the options which are

indicated by a blinking highlight. Some options such as reading rate, channel, range and relay state are predesignated, others require entry of a numerical value. When you have made your selection, hit ESCAPE, this will generate an ASCII file CROM1.CFG that stores your current selections. On running CROM.EXE another time, these configuration settings will be recalled. You can in theory alter or generate CROM1.CFG with a text editor, but this is not recommended as syntax checking and error reporting in CROM are minimal. Use CROM.EXE itself to change the contents of CROM1.CFG.

Some explanation of the choices and their effects on operation are provided below.

SAMPLE RATE

There is a choice ranging from 1 sample every 10 seconds to 10 samples/second which encompasses most of the rates likely to be used in chromatography although the CROM-1 can be usee at rates that are both faster and slower than the selections. Choose a rate that is consistent with your needs. The following tradeoffs are involved:-

1. The V/F produces about 100,000 counts/second at full scale on any range. The choice of sample rate affects resolution e.g. 10 samples per second amounts to 10,000 counts in the sampling interval or a resolution of 0.01%, whereas a 10 second sampling interval would provide 0.0001% resolution. The accuracy of the integral count is unaffected by sampling interval and depends on the non-linearity of the V/F, typically 0.005%.

2. Due to the design of the CROM.BIN driver, the total number of samples in a run must be less than 65,535:-
   Run Duration (secs) * Sample rate < 65,535
   If this condition is not satisfied, an error will be generated on selecting RUN and you will be returned to the CONFIG menu.

3. Since data is stored in ASCII format which is a wasteful way to use disk space (Lotus needs ASCII files) each sample takes 35 bytes of storage. A 10,000 sample run would need 350K of disk space - be prepared for this!

RUN DURATION

This is always entered as a whole number of minutes. There is no restriction on duration except as mentioned above. If you are not sure what duration is needed choose a number that is a lot longer than you are likely to need, you can always abandon a run by hitting ESCAPE from the keyboard.

INPUT CHANNEL

You have a choice of 2 input sources on the CROM-1 connector. Note that a source is permanently selected during a run, you cannot switch between the channels during the run. This allows you to share a CROM-1 between two sources so that one can be measured while the other is being set up etc.

FULL SCALE RANGE

Choose an appropriate full scale range for your detector. This also selects the plotting scale on the display. The CROM-1 has about a 30% overange capability above the full scale selected. For low level and bipolar detectors, it is possible to modify the CROM-1 hardware for different input ranges - contact MetraByte.

TRIGGER CODE

After autocalibrating, RUN will pause before starting the acquisition of data to check digital inputs IP0-IP3. The trigger code specifies which inputs must be energized before the run will proceed e.g. trigger code 5 would require IP0 and IP2 to be energized. If this condition is not satisfied, the program waits in a loop which can be broken out of by hitting any key on the keyboard. If you want to manually start the run from the keyboard then select any non-zero trigger code. For automatic starting, select an appropriate code and connect your external trigger signal(s) to the inputs.

RELAY ON/OFF

The state of the relay outputs at the beginning at the beginning and end of the run can be specified here.

BOARD ADDRESS

The address supplied here corresponds to the CROM-1 base address DIP switch setting as set up in installation. It should be in the range 200-3FC Hex. The entry may be decimal or hexadecimal (&H---) format.

INTERRUPT LEVEL

Specify which of the expansion bus hardware interrupt levels 2-7 that you wish to use (see Section 4.4).

RUN is the business part of CROM. After selecting this option, you will have to answer the following questions:-

Do you want to plot data on screen (Y/N)? Y

Do you want to print data on printer (Y/N)? Y

Do you want to dump data to file (Y/N)? Y

File name (default extension is .PRN): MYFILE.DAT

If you have a color graphics or other adapter capable of bit mapped graphics, you can answer yes to the first question to obtain a real time plot of the data on screen as it is taken. If you have a monochrome adapter or do not wish to plot data, answering no will give a display of each point in the form:-

Time Voltage Cumulative Volt-seconds

To determine the area under any section of the chromatogram, select two bounding points and subtract the cumulative volt seconds.

Answering yes to the second question will produce a printout on your line printer (LPT1:). Set your printer up at the top of a page. The speed of the printer may delay the processing of points so that the plotting will fall behind real time. The actual data is being taken in real time and the relays will operate correctly at the end of the run duration even though the processing of data is still taking place. This will make good use of the FIFO buffer in the driver. A print spooler can also be used to minimise the printer delay, or you can print a data file directly after the run.

The third question asks if you wish to send data to a disk file as it is taken. If you answer yes, you are further prompted for a file name. If you do provide an extension (.---) it will automatically default to .PRN which is what Lotus requires for an import file. The file generated is a sequential ASCII text file with the data comma delimited. Each sample contains 3 items of data:-

Time Voltage Cumulative volt-seconds (area)

A single sample requires 35 bytes of storage on disk.

RUN may be aborted before its normal end duration by hitting the ESCAPE key on the keyboard. This returns you to the main menu.

EXIT is self explanatory. Selecting this option will return you to DOS.

A.2 RECOMPILING CROM.EXE

The BASIC source CROM.BAS supplied on the disk has been compiled to produce CROM.EXE. You can run CROM.BAS using the BASIC interpreter for testing purposes. The interpreted version will be unable to plot in real time above about 2 samples/sec. and a more serious problem occurs with the disk, printer and plotting enabled. BASICA has a nasty characteristic of disabling interrupts for short periods and this will produce artifacts in the data and plot. For the purposes of testing and debugging, these are minor problems. You are free to modify the program, change file format, introduce new sample speeds, get rid of certain prompts etc. and generally alter and embellish any way you want, there are infinite possibilities.

Once your program is debugged, delete line 510 or whatever line loads the CROM.BIN driver, save the file in ASCII form (,A) and follow the instructions detailed in Section 4.10. The Microsoft Quick Basic compiler is both an economical and powerful way of compiling your BASICA programs for CROM.EXE. You can also use any of the IBM Basic Compilers.

Appendix B

AMD-9513 COUNTER DESCRIPTION

B.1 INTRODUCTORY 9513 DESCRIPTION & PROGRAMMING SEQUENCE

The AMD-9513A counter counter used in the CROM-1 is a complex device containing five 16 bit up/down counters, a crystal oscillator and binary/decade scaler and a variety of gating and control logic. Apart from a master mode register that controls overall operation of the AMD-9513 and oscillator scaler, each counter has associated with it a mode register, a hold register and a load register. Before CROM-1 can be used, its configuration must be set up by writing to the various mode and hold registers in the device. Normally this initialization will be performed by the CROM.BIN driver and be invisible to the user but this appendix provides an introduction to the AMD-9513 for those wishing to write their own drivers etc. For additional information on the Advanced Micro Devices AMD-9513A, users should refer to the manufacturer's data sheet.

All data transfers to the 9513 timer-counter are performed through 2 I/O ports. The port at the Base address is used for data transfer e.g. loading and reading counters and counter mode registers. The port at Base address + 1 is used for addressing, command, control and status purposes. Since there are many internal registers in the 9513 an indirect system of accessing them is used via an internal data pointer register which in turn is reached through the command register. The command register also performs other functions such as loading and enabling the counters and latching their contents etc. The various legal command register codes are listed on the next page.

Those codes that reference counter operations use a linear select S5 thru S1. Only the counters with the appropriate S bit set are affected. This is a powerful feature since it allows simultaneous loading, latching, enabling etc. of any combination of the 9513 internal counters.

COMMAND CODE								FUNCTION
C7	C6	C5	C4	C3	C2	C1	C0	
0	0	0	E2	E1	G4	G2	G1	Load data pointer register with E & G
0	0	1	S5	S4	S3	S2	S1	Arm counting for selected counters (S=1)
0	1	0	S5	S4	S3	S2	S1	Load source into specified counter
0	1	1	S5	S4	S3	S2	S1	Load & arm specified counters
1	0	0	S5	S4	S3	S2	S1	Disarm & save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save selected counters in hold registers
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	0	N4	N2	N1	Clear output bit N (001 <= N <= 101)
1	1	1	0	1	N4	N2	N1	Set output bit N (001 <= N <= 101)
1	1	1	1	0	N4	N2	N1	Step counter N (001 <= N <= 101)
1	1	1	0	0	0	0	0	Enable data pointer sequencing (clear MM14)
1	1	1	0	0	1	1	0	Gate FOUT on (clear MM12)
1	1	1	0	0	1	1	1	Enter 8 bit bus mode (clear MM13)
1	1	1	0	1	0	0	0	Disable data pointer sequencing (set MM14)
1	1	1	0	1	1	1	0	Gate FOUT off (set MM12)
1	1	1	0	1	1	1	1	Enter 16 bit bus mode (set MM13)
1	1	1	1	1	1	1	1	Master reset

Note that there is the following logical structure to command codes:-

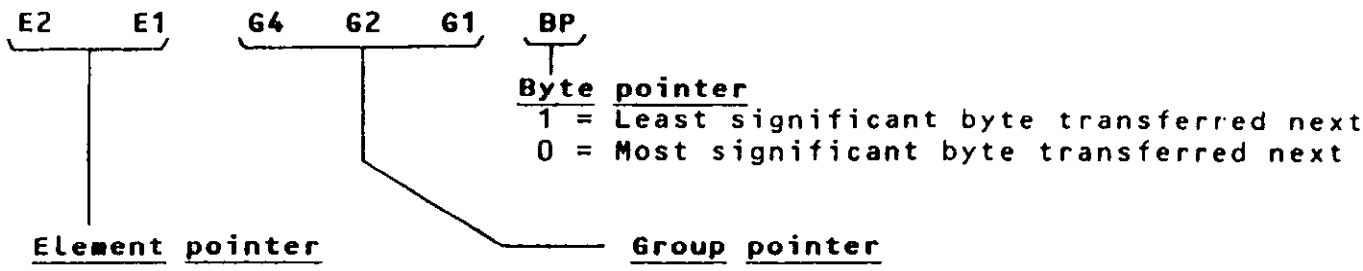
All codes beginning with 000 - Reference data pointer register

Codes from 001 to 110 - Reference counter operations

Codes beginning with 111 and ending in 001 thru 101 - Perform single bit counter functions

Codes beginning with 111 and ending in 000 or 110 thru 111 - Perform master control functions (all these functions can also be activated by writing the master mode register)

Returning to command codes that commence with 000. These codes select the internal registers according to the E and G fields which set the internal data pointer register. The 9513 has one master mode register which controls the operation of all the counters and the scaler. This must be set in the initialization sequence of your program. In addition, each counter has its own mode, load and hold registers. These registers are accessed through the data port at the Base address after setting the internal data pointer register to address the desired register. The data pointer register data format is detailed on the next page.



Counter group E2,E1:-
- 00 - Mode register
- 01 - Load register
- 10 - Hold register
- 11 - Hold register/ hold cycle increment

\+

G4,G2,G1:-
- 000 - Illegal
- 001 - Counter group 1
- 010 - Counter group 2
- 011 - Counter group 3
- 100 - Counter group 4
- 101 - Counter group 5
- 110 - Illegal

Control group E2,E1:-
- 00 - Alarm register 1
- 01 - Alarm register 2
- 10 - Master mode register
- 11 - Status register/ no increment

\+

G4,G2,G1:-
- 111 - always for control group

The data pointer consists of a 2 bit element pointer (E), a 3 bit group pointer (G) and a 1 bit byte pointer (B). The byte pointer bit indicates which byte of a 16 bit register is to be transferred on the next access therough the data port. Whenever the data pointer is loaded the byte pointer (B) is set to 1, indicating a least significant byte of data is expected next. With an 8 bit data bus as used on the IBM P.C., the byte pointer toggles following each 8 bit data transfer (master mode bit MM13=0). The element and group pointers are used to select which internal register is to be accessible through the data port. Although the element and group pointers in the data pointer register cannot be read, the byte pointer is available as a bit in the status register.

Random access to any internal location can be accomplished by loading the data pointer (through BASE ADDRESS +1) and then reading or writing to the location through the data port (at BASE ADDRESS) as appropriate. The counter registers are all 16 bit and after loading the pointer the data is transferred in low byte / high byte sequence. The following example shows loading counter 3 load register (using BASIC):-

```
xxx10 OUT BASE + 1, &H13     'write 000 10 011 to command reg.
xxx20 OUT BASE, 0            'low byte = 0
xxx30 OUT BASE, &H80         'high byte = 128
                             'register loaded with 32,768
```

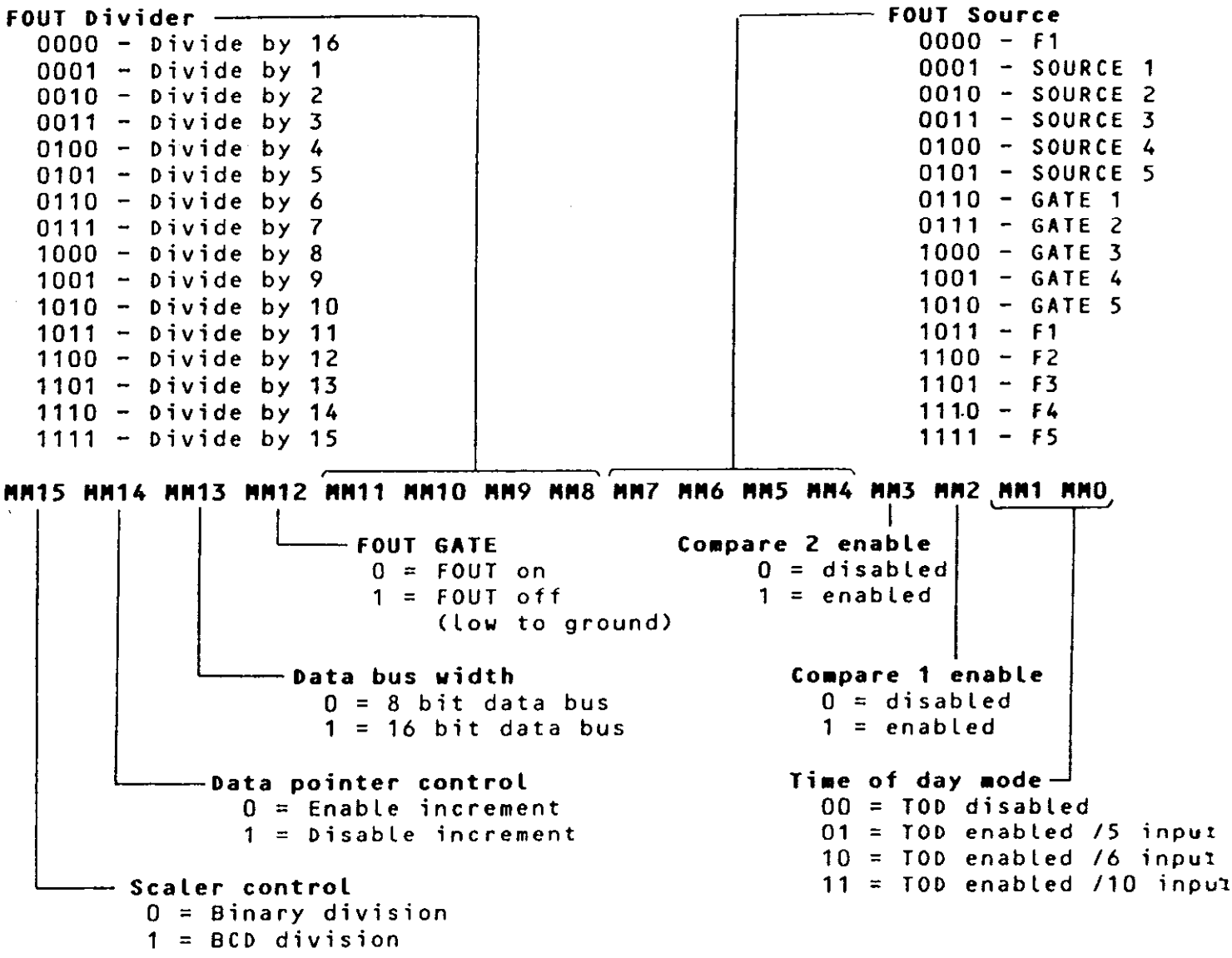
In many programs you will find a pattern of loading the counter mode

register, load register and hold register in sequence or setting alarm register 1, alarm register 2 and the master mode register. The element pointers are arranged to auto-increment on each 2 byte data transfer if master mode bit 14 (MM14) = 0. This saves writing to the command register between items of data and depending on your preferences is a feature that you may wish to utilise in the interests of brevity of code, or ignore in the interests of avoiding confusion.

In general most programs will consist of an initialization section that will set the overall operation of the 9513 through the master mode register and then proceed to set each counter operating configuration through their individual mode registers and finally load initial data into the counters through the load or hold registers. Following the initialization, the counters are usually enabled using the command register, possibly latched and read using the command and hold registers etc. or disabled, re-loaded and re-enabled etc. Most of the "heavy" work in the programming is in the initialization and subsequent reading and writing operations are much simpler.

B.2 MASTER MODE REGISTER

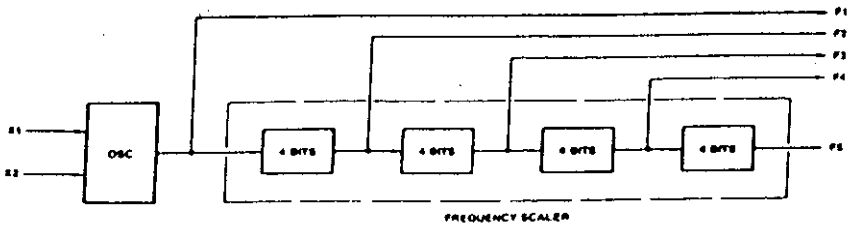
The master mode register controls the overall operation of the 9513 and should be the first register initialized by your program. The register is 16 bits:-



MM15 selects the dividers for the 4 counters in the xtal oscillator scaler. The scaler stages can divide by either 10 or 16 (BCD or binary) according to whether MM15 is 1 or 0. The fundamental xtal frequency F1 (1MHz) and each of the scaler outputs F2,F3,F4 & F5 can be routed to any of the counters and the FOUT divider by software control. For instance with MM15=1 (BCD) the frequencies will be:-

F1	F2	F3	F4	F5
1MHz	100KHz	10KHz	1Khz	100Hz

The structure of the oscillator scaler is shown below:-



Frequency	BCD Scaling MM15 = 1	Binary Scaling MM15 = 0
F1	OSC	OSC
F2	F1 ÷ 10	F1 ÷ 16
F3	F1 ÷ 100	F1 ÷ 256
F4	F1 ÷ 1,000	F1 ÷ 4,096
F5	F1 ÷ 10,000	F1 ÷ 65,536

MM14 selects automatic incrementing of the data pointer register. MM14 can also be individually controlled via the command register.

MM13 selects the data bus width and for IBM P.C. operation should always be zero (8 bit bus). MM13 can also be individually controlled by the command register.

MM12 controls operation of FOUT (pin 30 on the CTM-05). When MM12 is low, FOUT is enabled. When MM12 is high, FOUT is at a logic low (note this is not a tristate output). MM12 can also be individually controlled via the command register.

MM11 through MM8 set the divider modulus for the FOUT divider (not to be confused with ther oscillator scaler). This is a 4 bit divider counter ahead of the FOUT output. Any modulus from 1 to 16 is possible.

MM7 through MM4 set the input source of the FOUT divider. This can be any of the oscillator scaler outputs F1-F5, any of the counter gate inputs GATE 1 - 5, or any of the external source inputs SOURCE 1 - 5. Truly flexible!

MM3 and MM2 set the comparison modes for counters 2 and 1. If these bits are set, the comparator outputs are substituted for the normal counter outputs on Counter Out 1 and 2 (pins 35 & 34). The comparator output will be active high if the output control field of the counter mode register is 001 or 010 and active low for a code of 101. Once the compare output is true, it will remain so until the count changes and the comparison therefore goes false.

Finally MM1 and MM0 set the optional "time of day" mode for counters 1 & 2. When both these bits are zero, counters 1 & 2 operate in exactly the same way as allthe other counters. For other

combinations of these bits, the counter division ratios are set so that the most significant byte of counter 2 is hours, the less significant byte is minutes and the most significant byte of counter 1 is seconds. The least significant byte section of counter 1 becomes a pre-scaler in this mode and can divide by 50,60 or 100 for 50Hz, 60Hz or 100Hz (xtal) input frequencies.

After performing a master reset (OUT BASE +1, &HFF), the master mode word of the CROM-1 is normally set to C100 Hex.

B.3 COUNTER MODE REGISTERS

Each counter has its own mode register controls to control its operation. The counter mode registers should be initialized after the master mode register. Each register is 16 bits:-

Count Source Selection

- 0XXXX - Count on rising edge
- X0000 - Output of counter (n-1)
- X0001 - SOURCE 1
- X0010 - SOURCE 2
- X0011 - SOURCE 3
- X0100 - SOURCE 4
- X0101 - SOURCE 5
- X0110 - GATE 1
- X0111 - GATE 2
- X1000 - GATE 3
- X1001 - GATE 4
- X1010 - GATE 5
- X1011 - F1
- X1100 - F2
- X1101 - F3
- X1110 - F4
- X1111 - F5

Count Control

- 0XXXX - Disable special gate
- XOXXX - Reload from load
- X1XXX - Reload from load or h
- XX0XX - Count once
- XX1XX - Count repetitively
- XXX0X - Binary count
- XXX1X - BCD count
- XXXX0 - Count down
- XXXX1 - Count up

CM15 CM14 CM13 CM12 CM11 CM10 CM9 CM8 CM7 CM6 CM5 CM4 CM3 CM2 CM1 CM0

Gating Control

- 000 - No gating
- 001 - Active high level TCN-1
- 010 - Active high level GATE N+1
- 011 - Active high level GATE N-1
- 100 - Active high level GATE N
- 101 - Active low level GATE N
- 110 - Active high edge GATE N
- 111 - Active low edge GATE N

Output Control

- 000 - Inactive, output low
- 001 - Active high TC pulse
- 010 - TC toggled
- 011 - Illegal
- 100 - Inactive high impedance
- 101 - Active low TC pulse
- 110 - Illegal
- 111 - Illegal

CM13-15 control the effect of the GATE inputs on the selected counter. The gate input can be disabled (000) or enabled in a variety of ways. The counter can be gated for counting from the

previous counter (TCN-1 = Terminal Count of Counter - 1) e.g. Counter 3 could be gated by the output of Counter 2. Alternatively, the counter can be gated from its own gate input (GATE N) or adjacent gate inputs (GATE N-1 or N+1). This last configuration allows 2 or 3 adjacent counters to share the same gate control input provided the gate is level triggered. If only the counter's own gate input is used, it may be level triggered (active high or low) or edge triggered (positive or negative).

CM8-12 control the clock input source for the counter. You can select whether you count on the positive or negative input edge and select any of the SOURCE inputs, GATE inputs or xtal scaler outputs (F1-F5). Note this lets you connect several counters to the same source or a standard frequency input just through software! For cascading counters, you can connect to the terminal count output of the next lower counter e.g. for 32 or 48 bit counters etc.

CM3-7 control how the counter will operate. Essentially each bit performs a specific function.

CM0-2 control the terminal count output characteristics. It may be permanently low, high impedance, active low pulse, active high pulse or toggled on terminal count.

Users of the AT computer should note that all ports are 8 bits (one byte) wide and should perform byte oriented read/write operations rather than word (16 bit) operations. When performing consecutive byte transfers to the same I/O port on the AT (this a common occurrence with the 9513 architecture), the IBM P.C. AT Technical Reference Manual recommends the following coding in assembly language. This is required to allow sufficient recovery time for the AT I/O circuits, see section 9.8 of the manual.:-

```
        OUT IO_ADDR,AL      'write low byte
        JMP NEXT            'delay
NEXT:   MOV AL,AH           'fetch high byte
        OUT IO_ADDR,AL      'write high byte
```

Also the earlier non-A version of the AMD9513 (marked AM9513PC instead of AM9513APC) will not work reliably in the PC/AT due to access time limitations. Counters 1 & 2 on the CROM-1 are cascaded as a 32 bit up counter with the signal from the V/F applied to Source 1. The output of Counter 1 is hardware connected to Source 2 which is made the input of Counter 2. Strictly this is not necessary, as the same effect can be achieved in software by selecting the TCN-1 mode. If you wish, counter 3 & 4 which are unused can be cascaded to counters 1 & 2 using the TCN-1 selection for the counter mode words. This would form a 64 bit counter, sufficient to count 100KHz for nearly 6 million years without overflowing! Counter 1 mode word is normally set to 012D Hex and counter 2 mode word to 002D Hex. Counters 3 & 4 mode words are not set.

Counter 5 is used to generate interrupts from the crystal clock on the 9513. Its input is connected to F4 on the crystal scaler

which provides a 1KHz signal. Counter 5 is selected to be a square wave (output toggle) count down divide by N mode using mode word 0E22 Hex.

Once the mode registers are set, the counters are loaded and enabled. Counters are loaded via their corresponding hold registers, so counters 1 & 2 are reset by loading their hold registers to zero, and counter 5 is set to the correct division ratio by loading its hold register with N/2. After the hold registers are initialized, the counterswill still not be enabled until a Load & Arm (enable) command is issued.

Subsequently as each interrupt is processed, a counter latching command is issued to counters 1 & 2. This does not disturb the counting operation but simultaneously latches the contents of counters 1 & 2 into their hold registers so that the data can later be read.

Following the initialization section of CROM.ASM together with the information in this appendix will give you a good idea of how the AMD-9513 is used in the CROM-1.

Index