COMX-35

Technical manual

Table of Contents

1. General Technical Information

2. Principles of Operation

3. Expansion Guide

4. System Software Bulletine Issue I + II

COMX 35

General Technical Information

Power : unregualted DC 9V from external 4.5VA AC/DC adaptor current consumption $\leq$ 200 mA

Crystal Oscillation frequencies :

PAL models	=	8.867236 MHZ
		5.626 MHZ
NTSC models	=	7.15909 MHZ
		5.67 MHZ

Processor clock frequencies :

PAL models	=	2.813 MHZ
NTSC models	=	2.835 MHZ

Processor bus cycle frequencies :

PAL models	=	351.6 KHZ
NTSC models	=	354.4 KHZ

Memory cycle frequencies :

PAL models	=	351.6 KHZ
NTSC models	=	354.4 KHZ

RF TV output frequencies :

UHF	=	591.25 MHZ	(1mV)
VHF	=	61.25 MHZ	(1mV)

Cassette I/O baud rate (frequencies) ~ 2 KHZ

Processor : 1802A
Date transfer rate : 351.6 bytes/sec maximum
Memory capacity : 35K bytes (standard)
67K bytes (expanded)
Audio Speaker Output : ¼ watt

General

Circuit composition	:	Resistor, capacitor, inductor, diode and IC (majorty CMOS; others NMOS, TTL and linear)
Memory composition	:	16K bytes NMOS ROM 3-4 kilo bytes CMOS static RAM 32K bytes NMOS Dynamic RAM

Operational Technical Information

Dynamic memory refresh	:	by cycle steal DMA
Keyboard scanning	:	by dedicated self-scanning keyboard encoder.
Screen scanning	:	by video display controller.
Screen updating	:	buffer, interrupt driven.
Keyboard input	:	pseudo-interrupt driven.
Audio output	:	hardware generated.
Cassette I/O	:	software driven.

Table of Contents

1. General Technical Information
2. Principles of Operation
3. Expansion Guide
4. System Software Bulletine Issue I + II

COMX 35

PRINCIPLES OF OPERATION

CONTENTS

1. *Introduction*
2. *Power Supply Unit*
3. *Audio Amplifier*
4. *Reset Circuit*
5. *Keyboard Encoder*
6. *PAL/NTSC Selection*
7. *Cassette Interface*
8. *Microprocessor and Associated Circuit*

   8.1 Instruction Execution Sequence

9. *Video Interface System (VIS)*

   9.1 Writing to page memory
   9.2 Writing to character definition memory

10. *Dynamic RAM Subsystem*
11. *Interrupt and DMA Circuit Operation*
12. *Power-up Self-diagnostic Routine*

Introduction

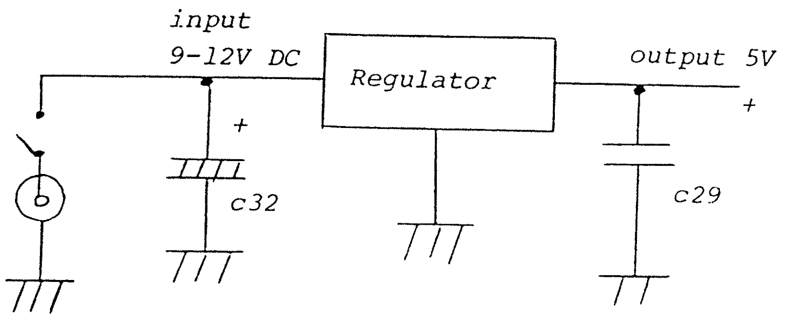
The hardware description of COMX 35 is divided into 10 sections and the operating principles of each subsystem is explained in sufficient details. Readers with fundamental understanding of any microprocessor should be able to understand the operating principle of the COMX 35 after reading this.

All service personnel should read this note before attempting to repair a COMX 35 computer.

2. *Power Supply Unit*

The COMX 35 comes with a wall-mounted AC adaptor which is rated at DC 9V(or 12V), 500MA (or 800mA). Power is fed to the COMX 35 through a standard DC power plug. Polarity is "centre-positive".

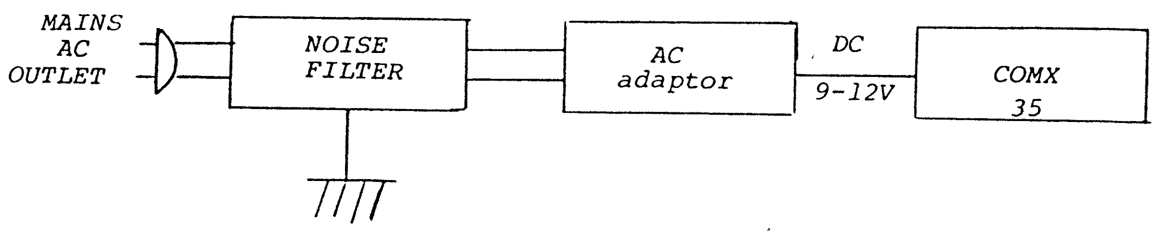
The output from the AC adaptor is fed to the audio amplifier IC and "pin C" of the expansion socket. All digital circuitries are operated under +5V DC power sourced by a 3-pin IC regulator. The DC adaptor supplies rectified unregulated DC input power to the regulator which gives a regulated 5V output. For this regulator to maintain a stable 5V output, the input must be kept at 7V or higher at all times



The two capacitors (c29,c32) are used to maintain the stability of the regulator IC. Unwanted parasitic output oscillation MAY result if these two capacitors are faulty.

NOTE : The AC adaptor must be plugged firmly into a wall socket. A loose connection WILL result in momentary (eg a few millisecond) open circuit and loss of power supply. This is the most common cause of error in the computer's operation.

Power-line noise generated by some home appliances (e.g. motor, hair dryer, air-conditioner during switch on) can also cause error in the computer's operation. Try not to plug the AC adaptor into the SAME power line shared by these home appliances. If problem still exist, install noise-filter in between the wall outlet and the AC adaptor.

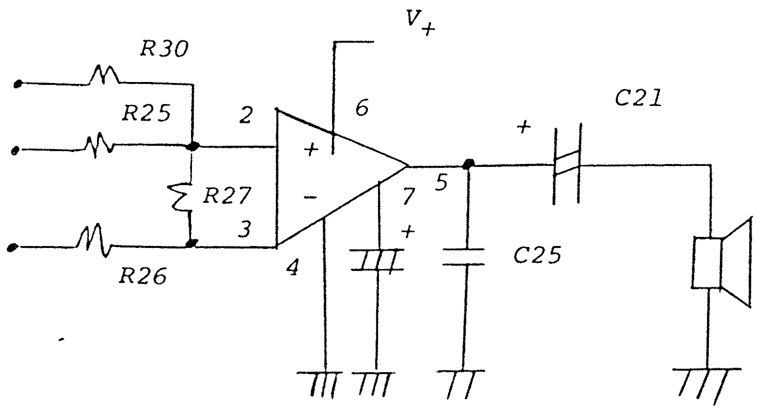


These filter can provide 30-50 dB attenuation to noise in frequency range of alout 100 KHz to 30 MHz. In Hong Kong, a 200V, 3A line filter is retailed at about US$4.00 - US$5.00.

This is the price of a bare unit, i.e. without sockets, casing, etc.

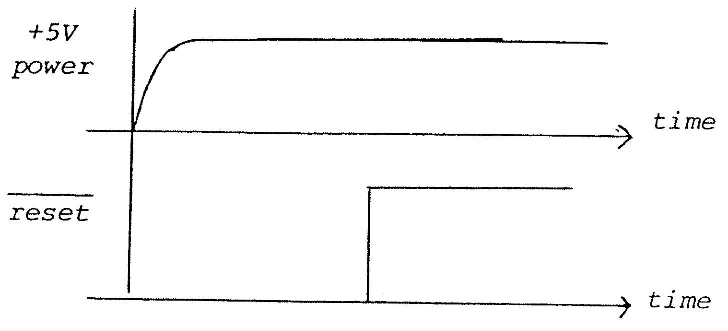
3. *Audio Amplifier*

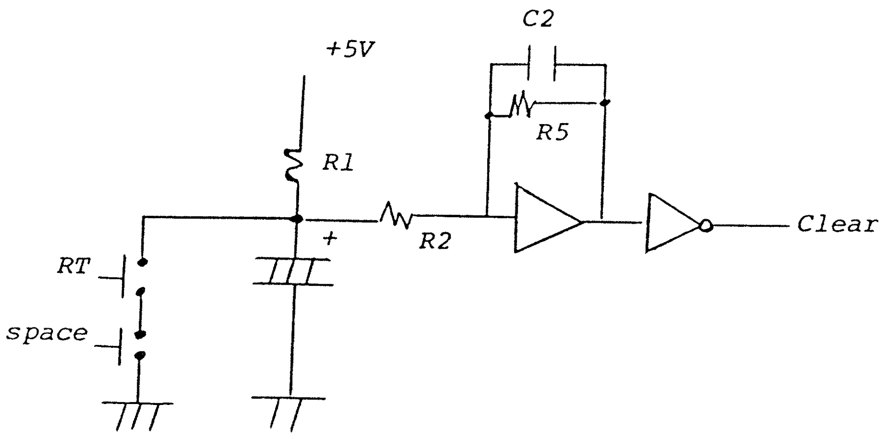
The audio section consists of one LM386 amplifier IC. Its input section consists of 3 resistors, R25-R27. This section determines the gain factor of the amplifier. Power is connected to pin 6 and is directly supplied by the AC adaptor. Pin 5 is the output pin and should stay at half the supply voltage when there is no input signal. C24 and C25 are used to eliminate parasitic oscillation. The capacitor connected to pin 7 eliminate 50/60 Hz HUM induced by power supply ripple.



4. Reset Circuit

This circuit is used to generate a sharp reset pulse at a delayed period after power up. R1 and C1 provide the necessary delay. C2 is used to speed up the "zero to one" transition and provide a sharp, glitch-free output.





5. *Keyboard Encoder*

COMX 35 use a matrix keyboard and is scanned by the hardware keyboard encoder (U4). U4 is clocked by TPB (pin 34) and will give a low output on $\overline{DA}$ (pin 33) line after a key is pressed. This is used to tall the CPU that a key has been pressed. The CPU then read this key with an "INPUT 3" instruction and a low is generated on $\overline{CS1}$, pin 21 during the read process. $\overline{RPT}$ (pin 35) is used for AUTO REPEAT function. R31 and C14 is the hardware debounce timing circuit. The capacitor is discharged on each key press. The keyboard encoder is checked (by the interrupt routine) after the completion of each TV frame (1/50 sec in PAL mode, 1/60 sec in NTSC mode).

6. *PAL/NTSC Selection*

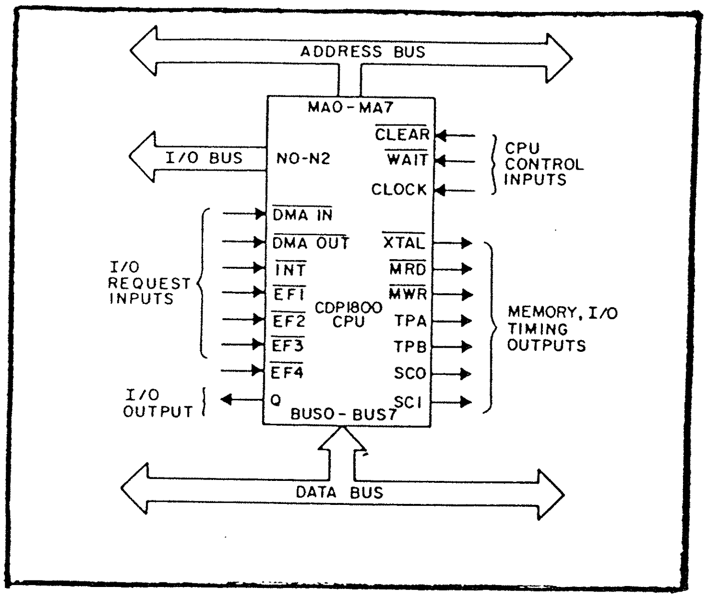
The COMX 35 computer will automatically select itself to be operated in PAL or NTSC scanning modes depending on the polarity of CR2 and pin 32 of U1.
In PAL mode, the two horizontal lines in the COMX sign (appears after power-up or reset) will become GREEN. In NTSC mode, they become YELLOW.
Pin 39 of CDP1870 should be HIGH for PAL mode. It should be LOW for NTSC mode. The system parameter stored at location 41CA (HEX) will be equal to 9 in PAL mode. It will be equal to 8 in NTSC mode.

7. *Cassette Interface*

The cassette output circuit consists of one section of U30 and its associated voltage divider. This circuit round the corners of square wave and attenuates it to provide a level suitable for feeding into the M/C input of an ordinary cassette recorder.
The cassette input section is a wave-shaping circuit; it derives a square wave signal from the analogue signal provided by an ordinary cassette recorder. It expects an input signal equal to or higher than 3V (peak-to-peak).
If C36 is open-circuited, the error rate of cassette playback will be increased.

8. Microprocessor and Associated Circuit

This is the heart and "master" of the COMX 35 computer; all other devices are under its control to perform various functions. The various pin of the CPU can be functionally grouped as shown in the diagram.



FUNCTIONS OF CPU PINS

8.1 *Instruction Execution Sequence*

In order to execute a machine language instruction, the CPU (CDP1802A) has to perform the following steps :

(a) *Send out the higher 8 bits address of the program counter through the multiplexed address bus (MAØ to MA7)*

(b) *When the higher address (A8 to A15) bits have become stable on the MAØ to MA7 lines, a HIGH pulse is sent on TPA line.. This pulse will clock the A8 to A15 into address latches (U13, U9).*

(c) *$\overline{\text{MRD}}$ line become LOW to indicate a memory READ operation.*

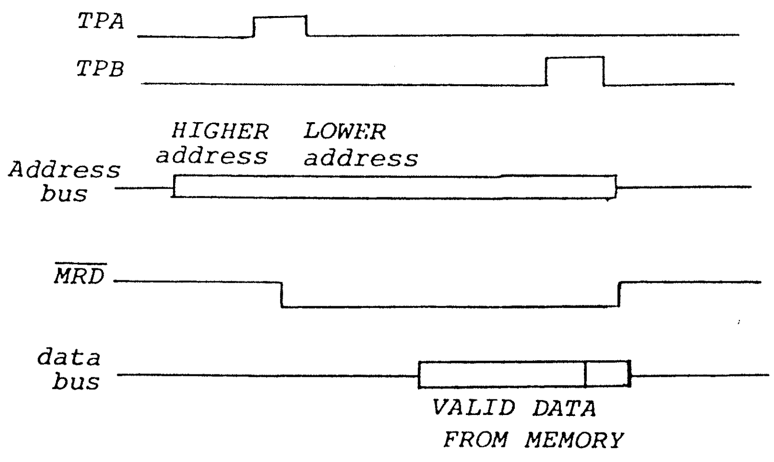
(d) *The CPU then outputs the lower address (AØ to A7) through the multiplexed address lines (MAØ to MA7).*

(e) *External address decoding circuit will detect whether the address range falls into the address range of a particular IC (e.g ROM,U21). The particular IC will be enabled if the address matches.*

(f) *The selected (address matched) IC will then send out data into the data bus of the CPU.*

(g) *The data is strobled into the CPU during the falling edge of TPB (timing pulse B).*

(h) *The CPU will then decode the machine instruction represented by the op-code just latched in and perform various task as instructed.*



MEMORY READ OPERATION

Having explain the basic operation of the CPU, we can then proceed to discuss the necessary conditions that must be satified before the CPU can function :

(a) Obviously, the CPU must have +5V power and clock signal.

(b) Clear, wait, DMA IN, DMA out and interrupt must be in normal states.

(c) The data bus must be clear from faults. No data line can be short-circuited to positive power supply or ground. They must not be short-circuited to other signal lines as well. The CPU will read in a wrong "op-code" if the data lines has fault(s). The CPU will not perform properly since it will receive "WRONG INSTRUCTIONS".

(d) The $\overline{MRD}$ line must be free from fault.

(e) The lower address line must be free from faults. Sometime, the CPU can still execute a short program when a high address line is short-circuited to ground. The power-up routine is specially written so that all or part of it will execute correctly even when there exists some minor circuit faults.

This routine also has build-in diagnostic functions. It will give useful indication as regard to the nature of fault of the COMX 35 computer.

9. Video Interface System (VIS)

This is the video and sound generating system. Moreover, it also generates several essential timing pulses for various parts of the COMX 35 computer.
The dot crystal (YZ) generate the basic timing signal for the whole COMX 35 computer. It is used to derive all timing signals (e.g horizontal and vertical sync.) in the TV signal. The dot crystal oscillates at about 5.6 MHZ, this signal is divided by 2 and outputs through pin 38 of CDP1870. This is used to clock the CPU and derive all timing pulses for memory accesses and I/O accesses.
Another important signal appears at pin 1; it will be LOW on each "display time" when the CRT controller has been programmed in on-state. This signal (50Hz for PAL, 60 Hz for NTSC mode) is used to drive the interrupt circuit and is essential for normal operation of a COMX 35.

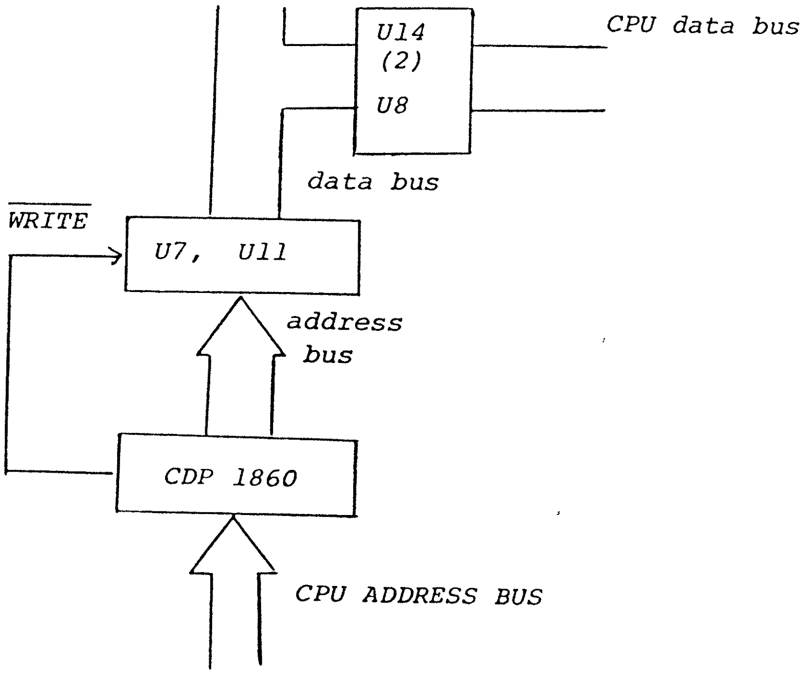
9. Video Interface System (VIS)

U7 and U11 are "page memory" which stores the ASCII code for each character position on the screen. This memory IC is 1K x 4 bit and two IC form 1K x 8 bit system. The screen has 960 character position which corresponds to 960 byte in this two ICs. Putting a character on TV screen is done by writing the ASCII code of that character into this "page memory".
PMSEL and $\overline{\text{PMWR}}$ of CDP1269 are internally decoded to response to memory access in the page memory address region (i.e F800 (HEX) to FBBF (HEX)).
U6 is called "character memory" which store the bit-by-bit character definition of each displayable character. This memory is also called "character generator". Most computer store their character bit pattern in ROM and is not user changable. The COMX 35 use RAM to store the character definition. A default character set is copied into these character definition RAM by the RESET routine. User can redefine them by the BASIC "SHAPE" command at anytime he wants.

U14(2) and U8 are bus seperators which will route the CPU data bus to the VIS subsystem during "write access" to the page memory (U7, U11). A HIGH pulse will appear at PMSEL (pin 39, CDP1869) during the access operation.

9.1 Writing to Page Memory (U7, U11)

The microprocessor address bus is decoded and fed to the address input of U7 and U11. CPU data are gated through U14(2) and U8 to reach the data input of the memory IC. Write pulse is provided through the $\overline{\text{PMWR}}$ pin of the CDP1869.



9.2 Writing To Character Definition RAM (U6)

Writing to character definition RAM is a complicate task and it involves a lot of intermediate steps. The write procedure is outlined below :

(a) *The upper 7 address bits is first written to the page memory in a way as illustrated in previous section*

(b) *The lower 4 address lines are feed through the CDP1869 to reach the address input of 46.*

(c) *"character memory access mode" are turned on by programming the CDP1870. Now, CPU data bus is connected to the data bus of U6 through internal multiplexer of the CDP1870.*

(d) *A write pulse is fed from the $\overline{\text{CMWR}}$ pin of CDP1869 to strobe the data into U6.*

10. Dynamic RAM Subsystem

This system consists mainly of eight 32K x 1 bit dynamic memory IC (U22-U29). Each IC will serve one data line.
The address inputs of dynamic RAM are multiplexed. ROW address is routed through U14(1) and U15(1) and is strobed into the internal latch of the dynamic memory by the $\overline{\text{RAS}}$ (row address strobe) pulse. ROW address is the lower 7 bit of the CPU address, i.e. AØ to A6. The higher address (A8 to A13) is first stored in the address latch (U9). A14 is provided by the jumper J1 and A7 provided by the MA7 line of CPU. A7 to A14 together forms the COLUMN ADDRESS input for the dynamic RAM. These "column address" is gated through U10 and U15(2) to reach the dynamic RAM inputs.

	CPU ADDRESS	GATED THROUGH
ROW ADDRESS	AØ to A6	U14(1), U15(1)
COLUMN ADDRESS	A7 to A14	U10 , U15(2)

10. U12 is a timing signal generator. It is clocked by signal at pin 9 and generate a series of timing pulses for the operation of the dynamic RAM and their associated buffers.

U18 is an CPU address decoder. It decodes address for dynamic RAM and ROM. It will generate enable pulse for the appropriate IC when the CPU give an address corresponds to the address range of a particular IC.

11. *Interrupt And DMA Circuit Operation*

Many home/personal computers suffers from INPUT/OUTPUT overheads. In COMX 35 this is minimized by implementing buffered interrupt-driven screen and pseudo interrupt-driven keyboard input.

In the process of generating vertical-sync for the video display, the CDP1870 provides a PREDISPLAY signal which envelops the timing period during video display scanning. When this signal is inactive (HIGH), the character memory and the page memory are free for access. If the output software waits for free accesses to these memories, CPU time will be wasted.

In COMX 35, the display output routine does not access the screen memory directly. Instead, the output characters are buffered in a FIFO memory. Interrupts are generated by the $\overline{\text{PREDISPLAY}}$ *signal (60 Hz/50Hz). The interrupt routine will flush the output characters in the FIFO memory into the page memory at the appropriate screen locations. Before returning back to the BASIC interpreter, the keyboard encoder is checked for key entries. Any keyboard input will be passed to a one-byte input memory buffer. BASIC will check this buffer whenever it proceeds to a new program line. Any 'ESC' key press will inhibt BASIC to continue onto the next line and thus breaking a BASIC program.*

The interrupt-handing hardware normally operates in the following manner :

(1) CDP1870 generates a LOW on $\overline{\text{PREDISPLAY}}$ *(Pin 1)*

(2) The signal goes to Pin 11 of U19 (FLIP-FLOP) and clocks its outputs pin 15 LOW (provided $\overline{\text{IDEN}}$ *becomes LOW after power-up. Will be explained in section 12)*

(3) CPU Pin 36 ($\overline{\text{INT}}$*) is driven low through CR1. The running program will be interrupted if the interrupt enable flag inside the CPU is set HIGH. (Note: During power-up sequence, the CPU will not respond to INTERRUPT even the* $\overline{\text{INT}}$ *line is driven LOW).*

(4) When the CPU responds to an interrupt, external hardware is notified by SCØ and SC1 (state codes). U16 decodes these state code signals and generates an interrupt acknowledge pulse on output Pin 7. This signal will drive Pin 8 of U19 and releases the $\overline{\text{INT}}$ *line.*

(5) *To prevent interrupts occurring during RESET period and power up routine. Output of U19 Pin 15 is held HIGH by a HIGH input on Pin 8. This is driven by U16 Pin 12 which is set LOW whenever*

$$\begin{array}{lll} & CLEAR & = \emptyset \\ or & \overline{IDEN} & = 1 \\ or & IACK & = 1 \end{array}$$

(6) *The above process take place in several microseconds and repeats at a rate of 50/60Hz. This is hardly seen on an oscilloscope without delay-sweep or storage. However a logic probe (pulse stretcher) may be used to detect the presence of these short pulses.*

Dynamic memory in COMX 35 is refreshed by DMA. An external counter (U20) is implemented to count on "execution cycles" and generate $\overline{\text{DMA OUT}}$ requests to the CPU. Upon acknowledging a $\overline{\text{DMA OUT}}$ request, the CPU will exercise a dummy memory read cycle at an address of value given by the DMA register. The DMA register (R(Ø)) will auto-increment after each DMA cycle.

However, R(Ø) is also used as program counter on power-up. To avoid instructions being skipped before another register R(n) is employed for program counter, U20 is held static by CLEAR at pin 6 and $\overline{\text{IDEN}}$ at Pin 9.

Output pin 13 and pin 2 of U20 should be free-running pulses during normal operation after the power up routine.

$\overline{\text{IDEN}}$ - Interrupts and DMA enable (ACTIVE LOW) signal is the master controlling signal over the interrupt and DMA supporting hardware. This signal is HIGH(inactive) during RESET and Power-up routine to suppress interrupt and DMA activity before the CPU is ready to handle them. This signal will go LOW (active) after the power-up/ RESET routine and stays LOW unless the COMX 35 is RESET again manually or powered-down.

12. *Power-up Self-Diagnostic Routine*

(ROM source listing line numbers are given for reference only)

(1) Line 41
Starting at location \$ØØØØ, six no operation instructions (NOP : \$C4) are used to allow pending DMA requests to be cleared. Even if any of these instructions are skipped, program flow will not be disturbed.

(2) Line 42-43
R(B) is set up as program counter. This is done to release R(Ø) for DMA activity afterwards.

(3) Line 49-54
Turn on high tone and stack pointer is initialized
((Note CDP1869 & CDP187Ø are accessed for the first time))

(4) Line 28-109
System perameters are initialized with default values from ROM. CPU records NTSC/PAL diode installation by checking $\overline{EF2}$ ((DRAM is being accessed here for the first time))

(5) Line 112-117
Screen memory (page memory) is initialized with \$20(SPACES).

((Page memory is accessed here for the first time))

(6) Line 119-125
Q line (pin 4 of CPU) will be set HIGH and then set low. This will clock U19(1) and $\overline{IDEN}$ will become LOW.
((A very short HIGH pulse appears on Q line. U20 begins to count and DMA action starts)

(7) Line 127-146
Character memory is initialized by copying the character definition table from ROM.
DRAM is used here. If fault exists on data lines between CDP1870 and character RAM, destorted characters will be seen after screen is turned-on at line 218.
((character memory is first accessed there))

(8) Line 112-209
Page memory is rewritten with 'COMX' logo' pattern. Different colour horizontal bars is used to distinguish between NTSC-PAL framing.

NTSC : -	YELLOW/CYAN	BAR
PAL : -	GREEN /BLUE	BAR

(9) Line 218

Screen is turned on now.

((coloured COMX logo is seen on the screen now))

(10) Line 224-235

Two tone bursts are produced with decaying amplitudes.

(11) Line 250-267

Colour of COMX logo is changed around every seven seconds. Keyboard is continuously monitored. If any key other then 'SPACE' is pressed, the system will proceed to bring up BASIC.

(12) Line 268-213

Screen is cleared first.
Display colour is changed to normal CYAN/WHITE.
Bring up BASIC

Note : - DRAM integrity is essential for start-up of BASIC.

- Interrupt circuity integrity is essential to display

```
COMX   BASIC   V1.00
READY
```

because BASIC output is interrupt driven.

- Keyboard will be read only if interrupts occurs.

Expansion Guide

For

COMX 35

EXPANSION GUIDE

Note: To be used with I/O map, memory map and expansion socket pin assignment. Knowledge of 1802 CPU is assumed.

(A) Hardware

A.1 Expansion Bus

On COMX 35 expansion bus, Pin B and Pin 8 are used by expansion. Peripheral expansion boards should support these signals or else they will not be supported by the expansion box of COMX 35.

Pin B (DS) - This signal is generated by the electronics of the expansion box. This signal should be used to enable the devices and memory on the expansion board inserted into that connector to be mapped into the I/O space and memory space.
Therefore memory and I/O decoding of any expansion board should include this signal. However, this pin is not driven on the side connector of COMX 35. In order that an expansion board will work with COMX 35 at the absence of the expansion box, this signal should be pulled up to +5V by a resistor (typically 10K ohm).

Pin 8 ($\overline{\text{DP}}$) - This signal should be generated by any expansion board whenever its devices and memory is being accessed. This signal will be detected by the expansion box electronics in controlling bi-directional data-buffers. This signal should be driven by open-collector driver or normally through a diode. Do not pull up.

A.2 Program in ROM/EPROM

On any expansion board, program in ROM should be mapped to address starting at \$CØØØ (HEXIDECIMAL). Try to avoid partial decoding. Address signals A15 and A14 are present on the expansion bus. A hex-flip-flop clocked by TPA falling edge will latch A8-A13. Address range \$CXXX can then be easily decoded using OR gates only. Remember to include DS and $\overline{\text{MRD}}$ in decoding.

A.3 *INPUT/OUTPUT*

Input/Output ports on any expansion board must not be mapped on top of built in I/O ports of COMX 35. Decoding must include DS. Input port 2 and output port 2 are recommended to be used. For data width not more than 8-bit, information should be passed through the data bus. For input data width more than 8-bits, Input ports 4-7 can be used with information passed on data bus. For output data width more than 8-bits, CPU OUTPUT Q can be used to duplicate output port 2.
16-bit output can also be achieved by using 1802 register output as in 1869. Output port 2 can be built to latch R(X) information multiplexed on the MA lines. Note that upper-byte is latched first. When 1802 outputs, memory will be read at the same time. When 1802 inputs, memory will be written at the same time. Therefore one does not have to worry about memory corruption when using OUT N for output operations. However, INP N can also be used for output if the hardware is set up correctly and the software written carefully. One can use INP N for output by means of register output through MA lines. The hardware is set up to detect INP N and latch information of R(X) on the MA lines. Unfortunately a write to memory anddressed by R(X) will take place at the same time. This is not a problem if the range of values of R(X) is restricted to point to areas of memory space which are non destructive (e.g. ROM, EPROM). This must be taken care by the driving software.

$\overline{\text{MRD}}$ should be used to distinguish between INP N and OUT N. Using $\overline{\text{MWR}}$ in controlling tri-state input buffers will give very short data window in writing memory. Output port should accept new data on signal edges. If a level-controlled latching output port is used, glitches may be present on the output-port outputs.
Memory-mapped I/O should be avoided. In cases the user insists to implement memory mapped I/O, they should overlay them on top of the ROM/EPROM on the same expansion board.
Remember to include DS in decoding.

A.4 *RAM*

Small amount of RAM can be obtained from system area of built-in RAM. Please check memory map and software section of this guide.

A.5 <u>Power</u>

Any expansion board should not draw more than 150 mA from the +5V regulated supply. Any expansion should not draw more than 250 mA from the V+ (≑9V) unregulated supply. Total draw from both supplies should not exceed 250 mA.

A.6 <u>Standard Dimension</u>

See enclosed drawing for standard size.

A.7 <u>Protection Cover</u>

Comx supplies protection cover for standard size expansion boards. Minimum delivery quantity is 1000 pc.

A.8 <u>Indicators</u>

Two LED's should be present on an expansion board.

SELECT - Indicates that the board is selected. Normally derived from the signal DS.

ACTIVE - Indicates that the board is being accessed.

(B) SOFTWARE

B.1.Ø Program ROM/EPROM Header

In order to maintain compatibility between COMX system software and software developed by add-on firms, several rules must be observed for all programs stored on add-on boards. The system will not function properly if these rules are violated.

B.1.1 Board Function Code

A '00' byte must be stored at location CØØØ (HEXIDECIMAL) for boards containing ROM/EPROM.

B.1.2 Board Identification Number

Every type of add-on board should contain an identification number. When a board selection command is issued in BASIC, the system program will sequentially enable each of the boards in the expansion box and search for the existence of a board with a particular IDENTIFICATION NUMBER. When the board is found, the system will electrically enable it and make subsequent access to it. If the board with that IDENTIFICATION NUMBER does not exist in the system, BASIC will return an error message.
The board selection command must indicate the IDENTIFICATION NUMBER of the desired board. The system software will search and access it; hence the add-on board can be plugged into any slot within the expansion box.
There are totally 8 banks in the bank-switching mechanism and labeled as bank Ø to bank 7.
The IDENTIFICATION NUMBER is one byte long and located at address CØØ1 (HEXIDECIMAL). Hence there will be 256 (0 to 255) different ID NUMBER. Address CØØØ (hexidecimal) should contain 00 to signify that an ROM/EPROM exist at that location. The system software will try to modify this byte; the system software will know that it is a ROM/EPROM if the modification fails.

B.1.3 Search Mechanism

For example, 3 add-on boards has been inserted into the expansion box. They are :

SLOT NUMBER (BANK NUMBER)	*BOARD ID NUMBER*	*BOARD TYPE*
Ø1	*Ø2*	*VOICE SYNTHESIZER*
ØØ	*Ø3*	*DISK INTERFACE*
Ø2	*Ø1*	*PRINTER*

B.1.3 *If the user wants to activate the printer in order to get a program listing, he must first find out that the ID-ENTIFICATION NUMBER of his printer interface card is equal to 1. He then issue a command : "SELECT 1".*
The system program will first electrically enable bank 0 and read the ID number at location CØØ1 (hexidecimal).
The data read will be equal to 3 and fails to match the desired board ID number. The system program then de-selects bank 0 and enable bank 1 and read the ID field again. The result is 2 and no match is obtained. The system software will sequentially enable the next bank until the desired ID number is found.

B.1.4 Board Identification Assignment Scheme

If two add-on boards happens to have same ID NUMBER, only the board located at a lower bank can be found and used. Therefore, it is very essential that boards manufactured by different manufacturers must not have same IDENTIFICATION NUMBER. Moreover, if any manufacturer expects user to install 2 or more cards of the same type, he must provide some hardware options (e.g. DIP SWITCH) that can make these two or more boards appears to have different ID NUMBERS when it is being interrogated by COMX system software.
In order to avoid ID number crash, Comx World Operations Limited has taken up the regulatory responsibility. Add-on firm should consult Comx and have an ID NUMBER assigned to each of their product. These ID number should also be shown on each of the add-on cards so that the card-users can know the ID NUMBER of their cards. Since the user will not buy two functionally identical cards and install them in one system at the same time, therefore, COMX may assign identifical ID NUMBER to all functionally identical cards manufactured by different manufacturers.
Hence, full functional description of an add-on card must be included in all ID NUMBER assignment application.

Command Entry Point Count

Address CØØ2 (HEXIDECIMAL) must contain the number of user entry points of the program stored on the ROM/EPROM.
This number is to be referenced by COMX system program.
Each user entry point usually performs a different task when "called" by user.

B.1.5 *Identification In Plain Language*

*This field is 40 bytes long. Card manufacturers should include the name of the board and its ID number in this field.
COMX DISK OPERATING SYSTEM will read this field and print out names of board in the system after booting.
This 40 bytes must be in ASCII code and will be printed as one line after booting. Unused bytes must be filled with spaces (HEX 2Ø)*

B.1.6 *Command And Execution Table*

*This table contains jump instructions and its associated English command keywords.
Each entry is 16 bytes long. The first 3 bytes will be a LONG BRANCH instruction. It is followed by the board ID NUMBER and a short English command string (in ASCII code). This string should be about 3-6 characters long; longer or shorter strings should be avoided if possible.
This command string should give the meaning of the function performed by this entry point clearly and concisely in plain or abbreviated English words.
This command and execution table starts at location C040 (HEX). Each entry is 16 bytes long and a maximum of 12 entries is allowed. The first entry should point to a subroutine that will call up the N*th *subroutine as listed in the execution table, N is a parameter number passed in the CALL command in BASIC eg. CALL (■ C040,3) will pass control to the subroutine located at C040(HEX). It will then, in turn, pass control to the subroutine 3 in the list. Allowable parameter range is 0 to 11 inclusively.
The command and execution table contains 12 entries of 16 bytes each. The table begins at memory location C040(HEX) and ends at location CØFF(HEX).*

Example : <u>COMX 35 Standard Printer Interface Driver ROM</u>

```
ADDR

C000 0001 0BFF FFFF FFFF FFFF FFFF FFFF FFFF;   ROM ID NUMBER = 1
C010 3030 3120 5354 414E 4441 5244 2D50 5249;   001 STANDARD-PRINTER
C020 4E54 4552 2049 4E54 4552 4641 4345 3A50;   INTERFACE: PARALLEL
C030 4152 414C 4C45 4CFF FFFF FFFF FFFF FFFF;
C040 C0C4 5501 5053 454C FFFF FFFF FFFF FFFF;
C050 C0C1 0301 504D 4F44 4530 FFFF FFFF FFFF;
C060 C0C1 4F01 504D 4F44 4531 FFFF FFFF FFFF;
C070 C0C1 1001 504D 4F44 4532 FFFF FFFF FFFF;
C080 C0C1 2501 504D 4F44 4533 FFFF FFFF FFFF;
C090 C0C1 3A01 504D 4F44 4534 FFFF FFFF FFFF;   COMMAND KEYWORD
C0A0 C0C1 6201 504D 4F44 4535 FFFF FFFF FFFF;   AND EXECUTION TABLE
C0B0 C0C5 0601 4455 4D4D 59FF FFFF FFFF FFFF;
C0C0 C0C5 0601 4455 4D4D 59FF FFFF FFFF FFFF;
C0D0 C0C5 0601 4455 4D4D 59FF FFFF FFFF FFFF;
C0E0 C0C4 7D01 5345 4C46 5445 5354 FFFF FFFF;
C0F0 C0C5 0601 4455 4D4D 59FF FFFF FFFF FFFF;
```

B.2.0 PROGRAMMING GUIDE LINES

B.2.1 Memory Configuration

Each add-on I/O board will contain at least one ROM OR EPROM containing its associated driver-program. A 2732 type (4K x 8 bits) EPROM is recommended as the availability and price of 2716 type EPROM is not stable nowadays. This memory size may be excessive in some applications, but there are ways to take advantages of this large EPROM/ROM space. The add-on board needs not contains its own RAM if the number of bytes required is not large. The driver program can use RAM in the COMX 35 basic unit.

B.2.2 RAM Area Allocation Scheme

Comx World Operations Ltd. has reserved some RAM area in the COMX 35 computer for add-on boards. In order to maintain compatibility between different add-on boards and between each add-on board and the system-program, add-on board manufacturers should contact COMX by writing or telex for RAM usage assignment if RAM inside the COMX 35 computer are to be used. COMX will assign some memory location to be used by each add-on board and make sure that all these add-on boards can be installed into one system and work properly. Basically, if two boards manufactured by different manufacturers happens to have identical functions, COMX may assign same memory locations to these two manufacturers since their products will not be put into one system and be operated at the same time. We will assign unique ID & RAM space to boards which are functionally different. Hence, we will ensure that these board will not interfere with each other and be able to work together in one system.

B.2.3 Usage for RAM Inside COMX 35

RAM area to be used by add-on boards is classified into 3 categories :

(A) SCRATCH PAD RAM (temporary storage, small size)

These area will be common to all add-on boards. The add-on-board program can access this area for temporary storage of data. Once the program finishs its job and return control to other program, this area may be used by other programs. Interrupt service routines will NOT change data stored in these area. This area is limited in size.

(B) *<u>PRIVATE RAM AREA</u>* *(Permanent storage,very small size)*

These memory locations is reserved for private use by different add-on board. These area are to be used for permanent data storage and storage of system-parameters. This area is very limited in size and the add-on board programmers are urged to make effective use of these RAM area. Usually, only a few bytes will be assigned to each add-on board.

(C) *<u>STACK MEMORY</u>* *(temporary storage, large size)*

This is the ONLY place where larger amount of memory space can be obtained for temporary storage. The add-on-board program will decrement the system stack pointer (CDP1802, register 2) and leave a empty 'STACK FRAME' for its usage. All memory access in the area will be referenced to the entry stack pointer value. The program will increment the stack pointer (by same amount as the decrement) before passing control back to the system program.

The program step will be shown as follows :

(a) *Disable interrupt before decrementing the stack pointer to avoid wrong interrupt sequence. This is to avoid the stack pointer being changed by interrupt service occuring during stack pointer manipulation.*

(b) *Save two registers,for example, register D and register E into system stack.*

(c) *Copy the system stack pointer into the saved registers, Register D and Register E in this example.*

(d) *Decrement Register 2 by the amount needed and enable interrupt again.*

(e) *Do the processing needed; storage access will be referenced by displacement from Register D.*

(f) *Disable interrupt.*

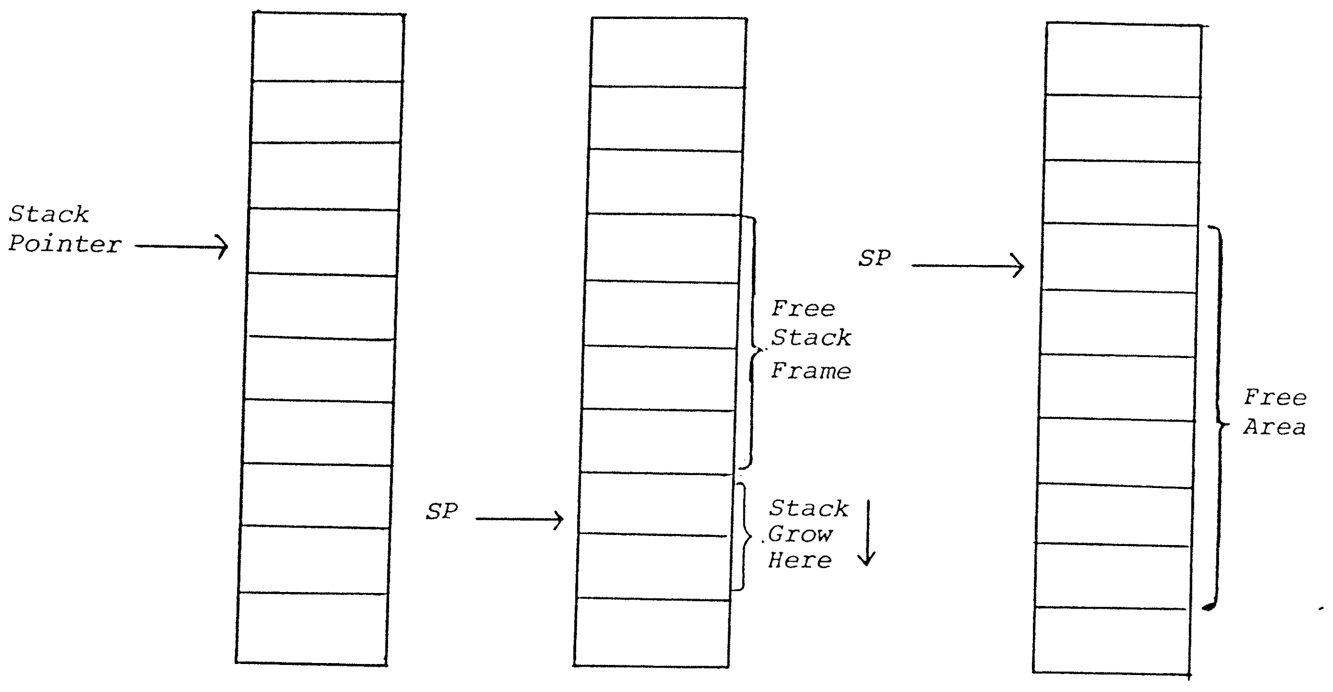
(g) *Restore system stack pointer by copying Register E into Register 2.*

(h) *Pull original contents of Register D and Register E from system stack and restore them.*

(i) *Enable interrupt.*

(j) *Pass control back to system program.*

The program can use standard call and return technique (SCRT) as the data will be stored in new stack locations after the stack pointer is decremented. However, the program must restore the stack before leaving. i.e. all memory locations in the new-stack area must be released.

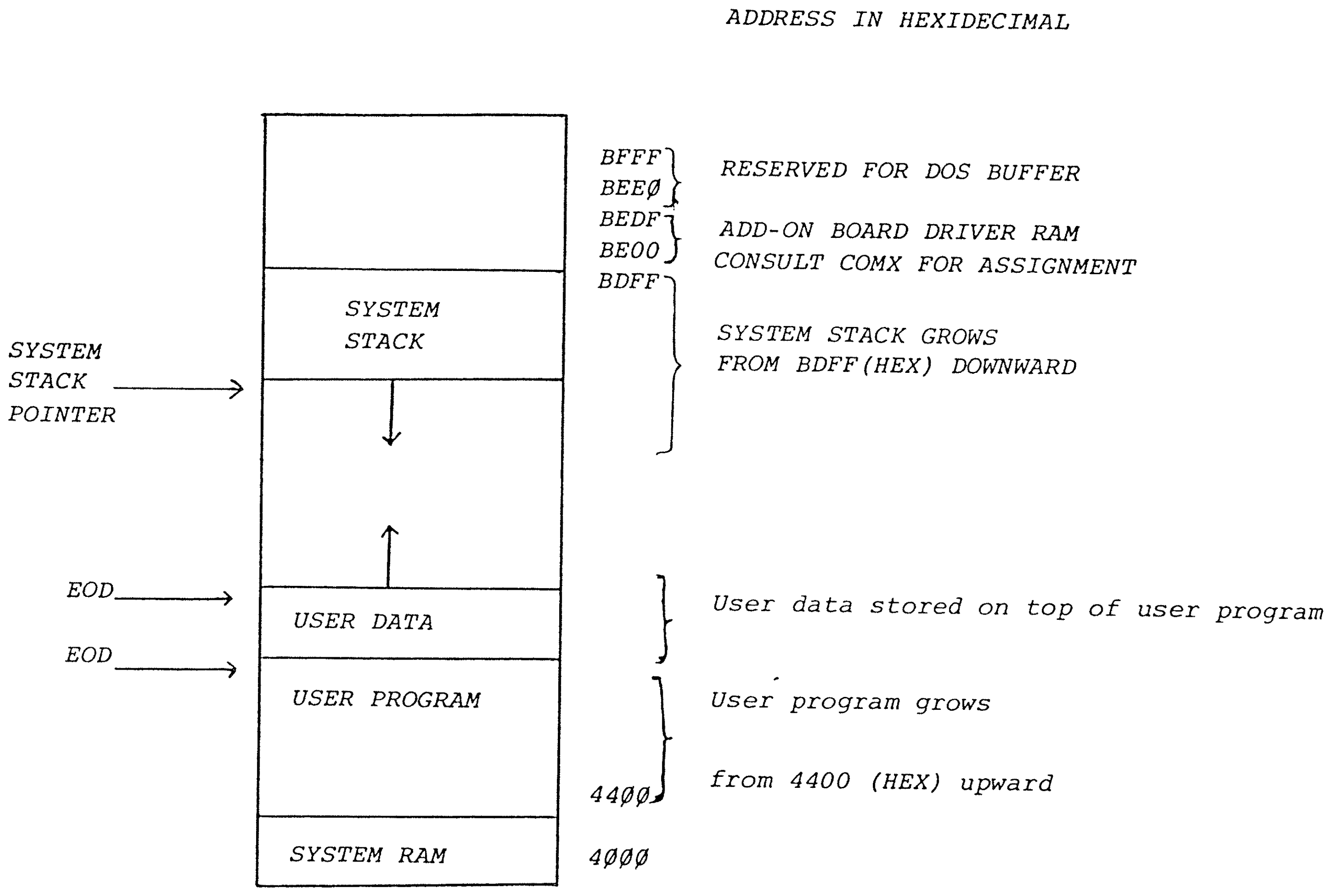


stack contents before entering this program

Decremented stack pointer

Restored stack pointer

B.2.4 MEMORY MAP OF COMX 35 RAM



B.2.4 MEMORY MAP OF COMX 35 RAM

The system stack pointer is set to BDFF (HEX) upon system reset. The system stack pointer will decrement as the system stack grow. The first statement of the user program is located at memory location 4400 (HEX) and user program's data are stored above the end of user program.
User can find out these pointers' value by typing 'EOP' and 'EOD'. The system will print out the values of the end of program pointer and the end of data pointer in hexidecimal numbers.
There is another BASIC function called 'MEM'. This function will return the value of memory left in the system. MEM is calculated by the formula :

MEM = (TOP OF SYSTEM STACK) - (271) - (EOD)

The system has reserve 271 bytes for its stack grow area and will give a negative value of MEM when the user data has grow into its reserved area.〈 MEM WARN〉 will be displayed to warn the user of this situation. The system will crash when the user data overwrite the system stack area.
If the add-on-board uses memory location in the stack area, the user's manual must give a warning to the board user. For example, if the add-on-board needs 512 bytes of memory in stack area, it must tell the user to stop entering program into the COMX 35 when the MEM value reach 512 or less. User should be advised to type "PRINT MEM" to find out the MEM value when a large (about 30K bytes long) BASIC program is entering into the COMX 35 computer. The COMX 35 has 32 kilo bytes of program memory and user program seldom has memory shortage even if the add-on-board use up some of the stack RAM. However, the add-on-board programmers are still advised to make efficient use of stack memory to preserve the effectiveness of the〈 MEM WARN〉 warning message.

B.2.5 Standard call and return technique (SCRT)

A full description of this technique is given in P.55-P.66 of RCA publication MPM-2Ø1C, user manuals for the CDP1802 COSMAC micro-processor. All CDP1802 system programmers and hardware designers are advised to obtain a copy from RCA.
In COMX 35 BASIC, Register 4 and Register 5 will always point to the CALL and RETURN routine inside the BASIC ROM.

To call a subroutine, the user simply write :

```
0000      D4      SEP     R4      --- use R4 as program counter
0001      1234    DC      SUB     --- subroutine address
0003               '              --- stored in 2 byte
                   '              --- continue execution
                   '              --- at 0003 (HEX)
                   '              --- after return from SUB
```

This calling sequence will destroy Register F high byte as it is used to store content in accumulator D. 2 bytes of stack RAM are used on each subroutine call.

To perserve RF high byte. Use this sequence :

```
9F      GHI     RF
D4      SEP     R4
1234    DC      SUB
 '       '       '
 '       '       '
 '       '       '
```

To return to the main program, the subroutine should end with a SEP R5 instruction

ie

```
                '                 --- subroutine body
                '
                '
D5        SEP         R5          --- return to calling routine
```

This will use RF (HIGH byte) to store accumulator D

To preserve it, use this routine.

ie

```
                '
                '
                '
                '
9F        GHI         RF          --- put RF.1 into D
D5        SEP         R5          --- return
```

The SCRT routines stored in COMX 35 ROM are listed below for your reference.

STANDARD CALL AND RETURN ROUTINE

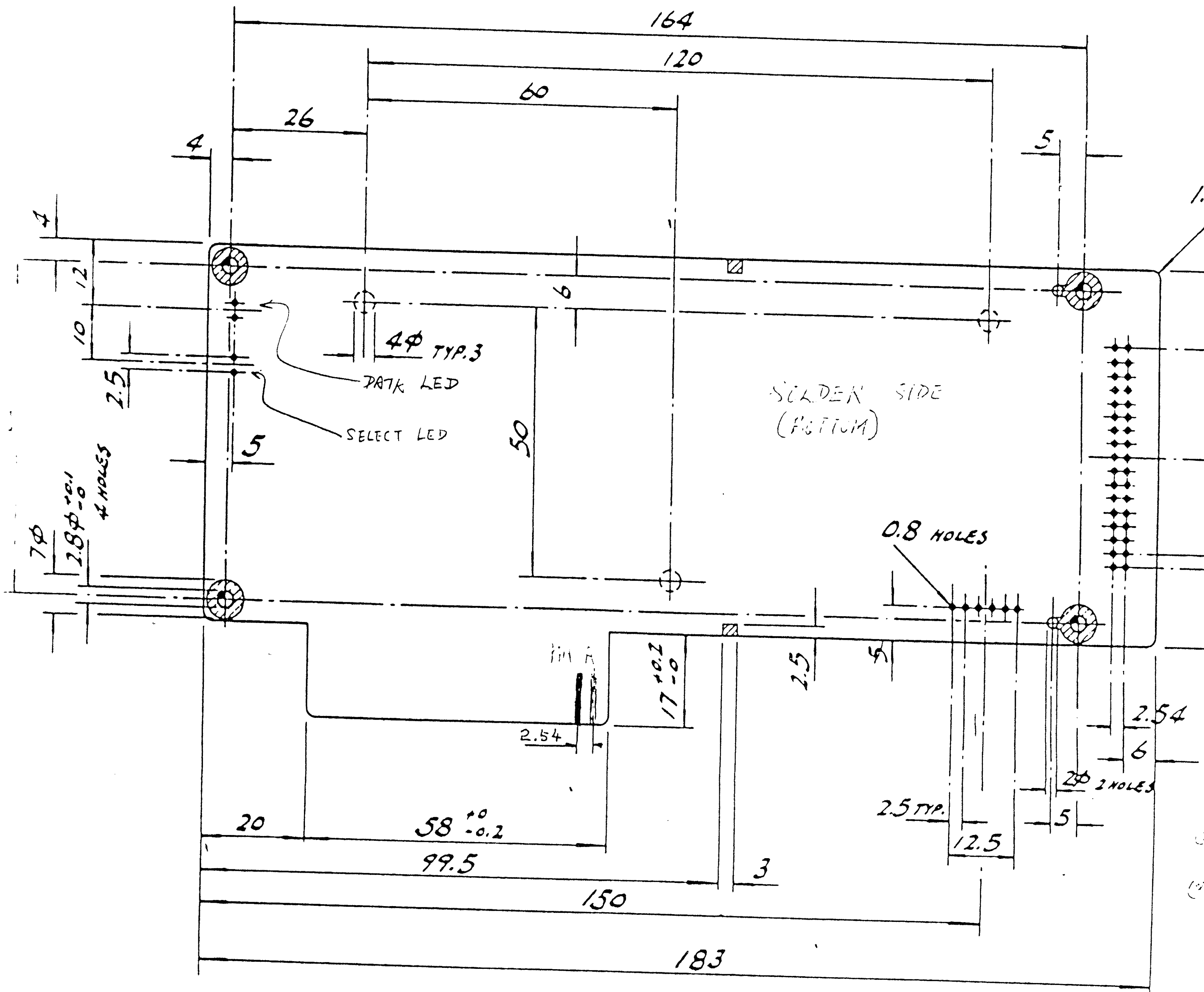
(Stored in COMX BASIC V1.00 ROM)

Note :	*STCALL*	*is pointed by Register 4*
	STRET	*is pointed by Register 5*
	WK	*represents Register F*
	SP	*represents Register 2*
	LINE	*represents Register 6*
	PC	*represents Register 3*

```
2E13  ;                  ..........................................
2E13  ;                        ..STANDARD CALL PROGRAM..
2E13  ;                        ..STCALL A
2E13  ;                        ........................
2E13  D3;                   SEP PC
2E14  BFE2;         STCALL  PHI WK; SEX SP
2E16  96738673              GHI LINK; STXD; GLO LINK; STXD
2E1A  83A693B6              GLO PC; PLO LINK; GHI PC; PHI LINK
2E1E  46B346A3              LDA LINK; PHI PC; LDA LINK; PLO PC
2E22  9F;                   GHI WK
2E23  3013;                 BR STCALL-1
2E25  ;                  ..........................................

31EA  ;
31EA  ;                  ..........................................
31EA  ;                        ..STANDARD RETURN PROGRAM..
31EA  ;                        ..STRET.A
31EA  ;                        .........................
31EA  D3;                   SEP PC
31EB  BFE2;         STRET   PHI WK; SEX SP
31ED  96B386A3;             GHI LINK; PHI PC; GLO LINK; PLO PC
31F1  12;                   INC SP
31F2  42A602B6;             LDA SP; PLO LINK; LDN SP; PHI LINK
31F6  9F;                   GHI WK
31F7  30EA;                 BR STRET-1
31F9  ;                  ..........................................
```

[illegible]
CARD [illegible]
[illegible]

UNIT : mm.



164
120
60
26
4
5
1.5R
4
12
10
2.5
5
6
4Φ TYP.3
DATA LED
SELECT LED
SOLDER SIDE
(BOTTOM)
50
40
40.64
2.54 TYP.
7Φ
2.8Φ +0.1 -0
4 HOLES
0.8 HOLES
PIN A
2.54
17 +0.2 -0
2.5
5
2.54
6
2Φ 2 HOLES
2.5 TYP.
5
12.5
20
58 +0 -0.2
99.5
3
150
183

The Technical Director
Comx World Operations Limited
5/F Wo Kee Hong Building
85-609 Castle Peak Road
Kwai Chung, N.T. Hong Kong
Tlx : 45823 CODAT HX

Applicant : ______________

COMX 35 Expansion Registration

Application Form

(1) Name of Add-On Card : ______________
Function/Application : ______________

(2) ROM/EPROM Size : ______ K bytes

(3) Number of Input Bits : ______ e.g. 8+3

(4) Number of Output Bits : ______ e.g. 16+3

(5) Q USED (Y/N) : ______ For ______
EF4 USED (Y/N) : ______ For ______

(6) RAM Requirement : Scratch Pad ______ bytes (Max. 20 bytes)
Private Status ______ bytes (Max. 8 bytes)
Stack ______ bytes

(7) Input/Output Method :

Instruction e.g. IN 1	Information Passed On e.g. MAØ - MA11	Function e.g. Output 12 bits

(8) Number of Entry Points : ______

(9) Interrupts Masked (Y/N): ______ For Periods of ______ µS/MS MAX.

(10) Power Consumption : 9V unregulated : ______ mA
+5V regulated : ______ mA

(11) Detailed Description of Operation and Programmable Configurations : ______
(Please use additional page(s))

FOR COMX USE ONLY : -	
Scratch Pad RAM Address : ______	Assigned On ______
Private Status RAM Add. : ______	By :
Stack Memory Allowance : ______	______
ID Number : ______	
Notes : ______	For and on behalf of Comx World Operations Ltd
	Ref No. ______

[illegible]

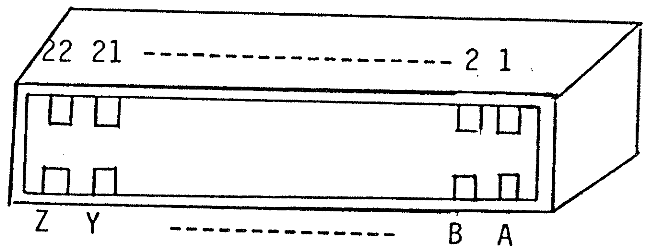
Expansion Socket

Pin Number		Function
Top	Bottom	
1		Ground
	A	Ground
2		NC (Reserve for negative supply)
	B	NC (Reserve for DEVICE SELECT SIGNAL (DS) = $\overline{\text{DS}}$
3		+5V (Regulated, 150 MA available)
	C	V+ (Unregulated ~ 9V)
4		Data bus Ø
	D	Data bus 1
5		Data bus 2
	E	Data bus 3
6		Data bus 4
	F	Data bus 5
7		Data bus 6
	H	Data bus 7
8		NC (Reserved for DEVICE PRESENT signal ($\overline{\text{DP}}$))
	J	CPU Q output (PIN 4) Cassette output signal
9		$\overline{\text{CLEAR}}$ PIN OF CPU (CDP1802A, PIN 3)
	K	$\overline{\text{MRD}}$ memory/IO READ
10		TPA (timing pulse A, CPU PIN 34)
	L	NØ CPU pin 19
11		N1 CPU pin 18
	M	N2 CPU pin 17
12		$\overline{\text{RAS}}$ row address strobe of DRAM (MSM3732 pin4)
	N	$\overline{\text{INT}}$ INTERRUPT, cpu PIN36, 10K ohm PULL UP
13		$\overline{\text{WAIT}}$, CPU PIN 2, 10K ohm PULL UP TO +5V
	P	CLOCK CPU clock, 2.8 MHZ, CDP1870 PIN 38
14		SC I CPU PIN 5
	R	SC Ø CPU PIN 6
15		$\overline{\text{EF4}}$ CPU PIN 21

[illegible]

PIN NUMBER		Function	
	S	$\overline{\text{CASE}}$	column address strobe enable
16		TPB	CPU PIN 33, timing pulse B.
	T	$\overline{\text{A15}}$	Negated address line 15. (latched)
17		$\overline{\text{MWR}}$	memory/IO write
	U	A14	latched address line 14
18		M A7	Multiplexed address 7
	V	$\overline{\text{A14}}$	Negated latched address line 14
19		M A5	Multiplexed address 5
	W	M A6	Multiplexed address 6
20		M A4	Multiplexed address 4
	X	M A3	Multiplexed address 3
21		M A2	Multiplexed address 2
	Y	$\overline{\text{EXTROM}}$	Disable internal ROM completely
22		M A1	Multiplexed address 1
	Z	M AØ	Multiplexed address Ø

PIN
NUMBER



Front view of expansion socke 1
(View from right side of COMX 35)

AW✓

Memory Map of COMX 35

Memory Location	Description
0000 - 3FFF (hexadecimal)	Build-in system ROM Contains (1) BASIC INTERPRETER (2) TV and keyboard operating system (3) Cassette operating system
4000 - 43FF	System parameter storage area.
4400 - BDFF	BASIC 'PROGRAM' and 'DATA' storage area. Top of system stack located at BDFF.
BE00 - BFFF	Reserved for disk operating system
C000 - DFFF	8 banks of 8K bytes - Used for system expansion - Program must select the desired bank before making read/write operation to this area.
E000 - EFFF	Reserved - Used for inter-bank communication - For application that cannot be fitted into one 8K byte bank.
F000 - F3FF	Reserved
F400 - F7FF	Character definition RAM area. - This area holds the dot pattern of the 128 characters. - The character can be changed using the SHAPE command in BASIC. - Each character uses 9 bytes in PAL system. - Each character uses 8 bytes in NTSC system. - This area can only be accessed by turning on the "character memory access" mode of the CDP 1869 IC. Refers to data sheet for details.

A W

Memory Location	Description
F800 - FFFF	Screen page memory - Each byte holds the character code of one display character position on the screen. - Before scrolling, F800 represents the left-top character position on the screen. F801 is the one next (to the right) to it and so on. FBBF represents the bottom-right position on the screen. - Hardware scrolling is accomplished by changing the home address register inside the CDP1869 IC. Refers to CDP1869/1870 data sheet for details. - This area is write-only and can only be accessed during the non-display period. The program may wait for non-display period and directly write to this area. e.g. `B1 $ ... check EF1 and wait for non-display` `STR R7 ... write to this area.`

Input / Output Port Assignment

INPUT	1	FREE		
	2	FREE		
	3	KEYBOARD ENCODER IC INPUT (use data bus)		
	4	FREE		
	5	FREE		
	6	FREE		
	7	FREE		
OUTPUT	1	BANK SELECT		
	2	FREE	(normally used for system I/O)	
	3	CDP 1870	CONTROL	(use data bus)
	4	CDP 1869	CONTROL	(use M/A bus of CPU)
	5	CDP 1869	CONTROL	(use M/A bus of CPU)
	6	CDP 1869	CONTROL	(use M/A bus of CPU)
	7	CDP 1869	CONTROL	(use M/A bus of CPU)

User installed I/O device can use INPUT/OUTPUT port number 2.

Bank select program

```
Machine code   Source program
                                    ... P=3, x=2 when enter to this program
   E3            SEX R3             ... X points to program counter
   61            OUT  1
   02             DC  2             ... select bank 2
   E2            SEX R2             ... reset x=2, normal stack pointer
```

In line 3, the constant will determine the bank being selected.

i.e. 1 Will select bank 1 to be the active bank
     2 Will select bank 2 and so on.

I/O MAP

I/O

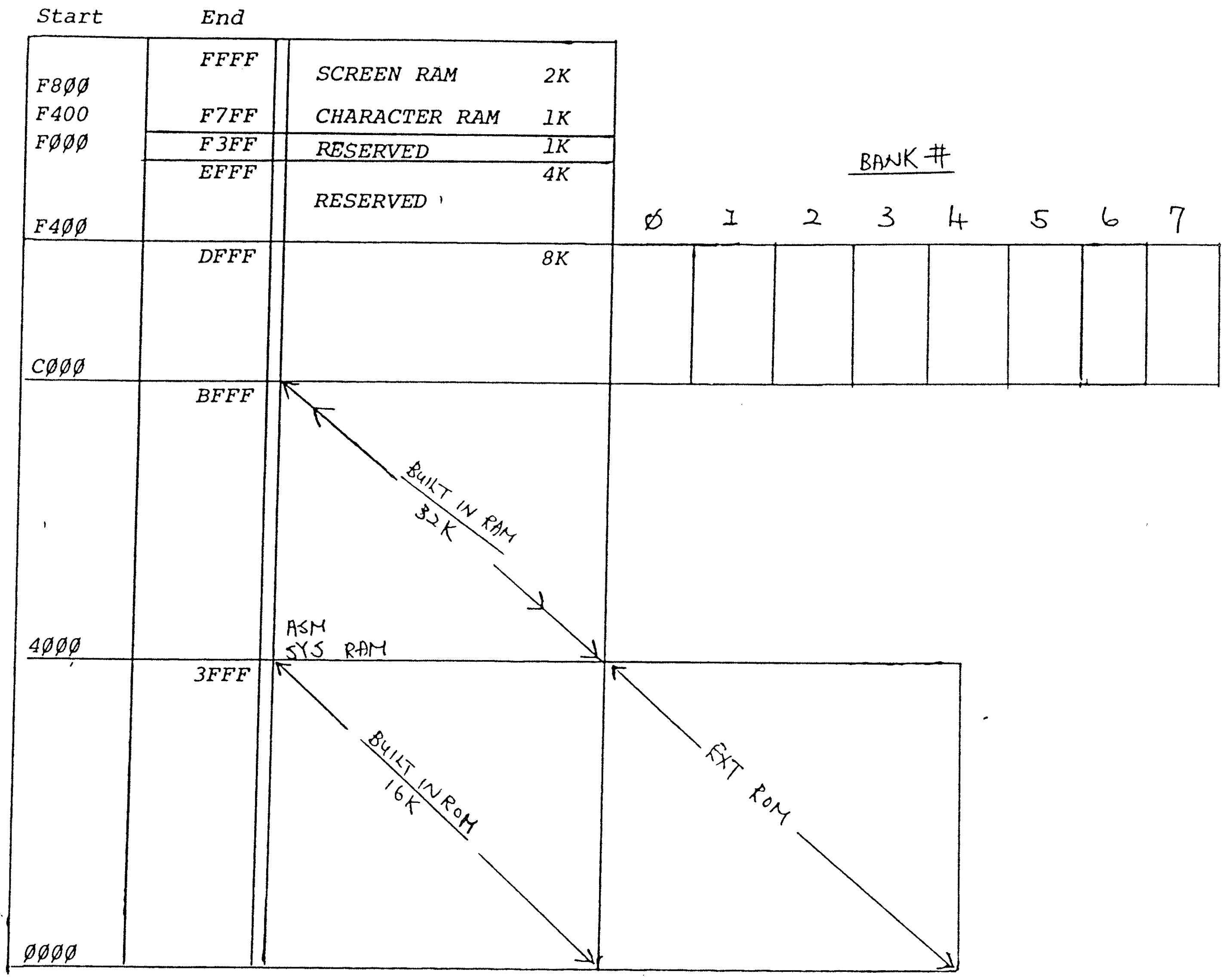
Q	- FIRST +VE GOING EDGE ENABLES DRAM REFRESH ENABLES INTERRUPT BY $\overline{\text{PREDIS}}$ - CASSETTE OUT (MIC)
$\overline{\text{EF1}}$	- PREDIS
$\overline{\text{EF2}}$	- ON POWER UP 1=PAL Ø=NTSC - AFTER TOGGLE Q = $\overline{\text{EFXB}}$
$\overline{\text{EF3}}$	- $\overline{\text{EFXA}}$
$\overline{\text{EF4}}$	- CASSETTE IN (EAR) - MAY BE OVERODE

N	IN	OUT	Use
Ø	N/A	N/A	
1		BANK	D
2			
3	1871 D	1870	D
4		1869	MA
5		1869	MA
6		1869	MA
7		1869	MA

D= Data Bus

MA= Multiplexed Address Bus

MEMORY MAP


Start
End
F800
FFFF
SCREEN RAM 2K
F400
F7FF
CHARACTER RAM 1K
F000
F3FF
RESERVED 1K
EFFF
RESERVED 4K
F400
DFFF
8K
C000
BFFF
BUILT IN RAM 32K
4000
ASM SYS RAM
3FFF
BUILT IN ROM 16K
EXT ROM
0000
BANK #
0 1 2 3 4 5 6 7

Software

System Software Bulletine Issue I

Character Size Selection

The COMX 35 computer can display characters in any one of the following FOUR formats.

1. Normal size characer - The display is 24 rows by 40 columns. 960 characters are displayed on TV screen. This is the "normal" operation mode and is the only mode supported by BASIC.

2. Horizontally expanded - The characters are twice as wide as the normal characters. Height are unchanged. The display become 24 rows by 20 columns and 480 characters are displayed. This mode is not supported by BASIC and only the upper half of the "page memory" is displayed on the TV screen. Some characters will "fall below" the TV screen and not displayed.

3. Vertically expanded - Characters are twice as high but width remain normal. The screens becomes 12 rows by 40 columns. As in mode 2, only 480 characters are displayed; others will "fall below" the screen and will only be displayed after scrolling up.

4. Vertically and horizontally expanded - This is a combination of mode 2 and 3; the characters are now twice as high and twice as wide. Only 240 (i.e ¼ of normal 960 characters) characters are displayed. Other will "fall below" the TV screen. The characters size is now 4 times as big as the normal size.

Changing Character Size

1. Horizontally Expanded Mode.

```
Turn   POKE  (■ 41CØ, PEEK (■ 41CØ) AND #7F)
  On   SCREEN ( PEEK (■ 41CØ) + 1)

Turn   POKE  (■ 41CØ, PEEK (■ 41CØ) OR #80)
 Off   SCREEN ( PEEK (■ 41CØ) + 1)
```

2. Vertically Expended Mode.

```
Turn   POKE  (■ 41C3, PEEK (■ 41C3) AND #7F)
  On   NOISE  ( PEEK (■ 41C4) +1, PEEK (■ 41C3)+1)

Turn   POKE  (■ 41C3, PEEK (■ 41C3)  OR #80)
 Off   NOISE  (PEEK  (■ 41C4) +1, PEEK (■ 41C3)+1)
```

3. Vert and Horizontally Expanded Mode.

   Do the turn ON program of both mode (1) and mode (2).

4. Return to normal mode.

   Do both the turn OFF program of mode (1) and(2).

Changing Character Size (Simplified)

1. Horizontal Expand

```
POKE   (■ 41CØ, #60)      ]  ON
SCREEN (1)

POKE   (■ 41CØ, #E0)      ]  OFF
SCREEN (1)
```

2. Vert Expand

```
POKE   (■ 41C3, Ø)        ]  ON      (PAL)
NOISE    (0,0)

POKE   (■ 41C3, #08)      ]  ON      (NTSC)
NOISE    (0,0)

POKE   (■ 41C3, #80)      ]  OFF     (PAL)
NOISE    (Ø,Ø)

POKE   (■ 41C3, #88)      ]  OFF     (NTSC)
NOISE    (Ø,Ø)
```

NOTE : TONE, SCREEN, COLOR, NOISE, MUSIC

commands issued may be scrambled and irregular response may be resulted in later operation.

Use this simplified produce in demonstration only. They will work properly if none of the above named command has been issued after the last system reset. Use the proper procedures listed on previous page for all programs.

<u>*System Software Bulletine (Issue 2)*</u>

Maxing Machine Language And BASIC Language Programs

1. *INTRODUCTION*

Machine language program can perform a given task significantly faster than the equivalent program written in BASIC language. Hence, it is desirable to write programs in machine language whenever higher execution speed is preferred. This specifically true for programs that involve movement of graphical objects on the TV screen.

However, writing a machine language program is not an easy task. It may take significantly extra effort and time when compared to the writing of an equivalent program in BASIC. Hence the most efficient way to do the job is to inter-mix programs written in these two languages. The calculation and decision making routines may be written in BASIC language so that it can be quickly written and debugged. Complex algorithms can also be implemented easier in the BASIC language. Part of the graphics, sound and colour routines may be written in assembly language to improve their execution speeds.

Programmers should use the following approach :

(A) Write the whole program in BASIC language first. Care is taken in separating the program into several functional blocks which may be replaced by machine language routines at a later stage.

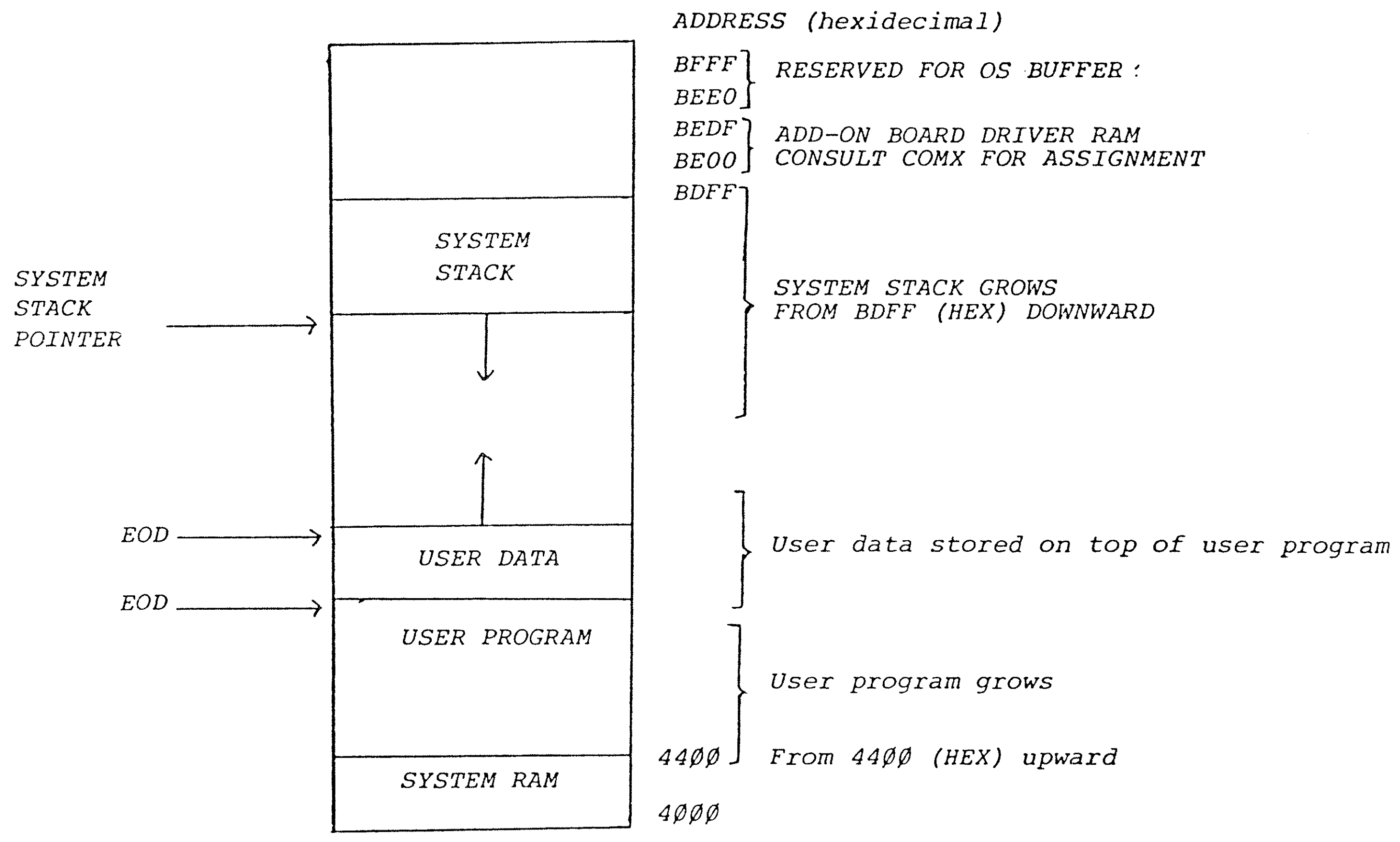
(B) When the whole program has been debugged and is running flawlessly, the program flow should be re-analysed. The speed-determinating part of the program is then identified. Usually, the speed determinating routines are the screen output routine or routines that are repeatedly executed in loops.

(C) Some routines are then replaced by machine language routines until the execution speed of the program becomes satifactory.

THIS IS A PRE-RELEASE COPY!
HAVE NOT GONE THRU PROOF-READING!
NO RESPONSIBILITY WILL BE BORN
BY COMX FOR DAMAGES INCURRED
IN THE USE OF THIS ~~[illegible]~~ BULLETINE

2. ALLOCATION OF MEMORY SPACE

The memory map of COMX 35 computer is shown below :



The COMX BASIC interpreter can accommodate machine language program and/or user defined data by storing the user's BASIC language program at a higher-than-normal location. This can be done by using the DEFUS command in BASIC. This command will redefine the "Start of User Space" to a new value as specified. For example, if 512 bytes of memory is needed, the "Start of User Space" pointer should be incremented by 512 bytes and becomes 46ØØ(HEX). This is illustrated by a sample program as shown below:

```
:DEFUS @4600

READY
:10 REM START OF USER SPACE BECOME @4600

:LIST

10 REM START OF USER SPACE BECOME @4600


READY
:CALL(@C040,7 7)

START:4600
END  :4640


4600  FF 00 37 FF FF FF FF FF  ..7.....
4608  FF FF FF FF 00 0A 24 80  ......$.
4610  20 53 54 41 52 54 20 4F   START O
4618  46 20 55 53 45 52 20 53  F USER S
4620  50 41 43 45 20 42 45 43  PACE BEC
4628  4F 4D 45 20 40 34 36 30  OME @460
4630  30 0D FF FF 03 04 0D FF  0.......
4638  FF FF FF FF FF FF FF FA  ........
4640  00 00 00 00 00 00 00 00  ........

READY
```

The BASIC interpreter keeps a system parameter called "Start of User Space" which define the memory location at which the storage of the user's BASIC program starts. The first 11 bytes of the "User space" are reserved by the system. Actual storage of user's program starts at the 12th byte in the user space.

After power-yp or reset, the "Start of User Space" is initialized to 44ØØ (HEX) as a default value. Storage of user's program start at 44ØC (HEX). This is illustrated by dumping the actual memory contents of a sample program.

Note : The memory dump routine is present on the printer interface card.

```
:LIST

10 REM FIRST BASIC LINE HERE


READY
:CALL(@C040,7)

START:4400
END  :4450


4400  00 00 2C FF FF FF FF FF   ..,.....
4408  FF FF FF FF 00 0A 19 00   ........
4410  20 46 49 52 53 54 20 42    FIRST B
4418  41 53 49 43 20 4C 49 4E   ASIC LIN
4420  45 20 48 45 52 45 0D FF   E HERE..
4428  FF 03 84 0D FF FF FF FF   ........
4430  FF FF FF FF FF FF FF FF   ........
4438  FF FF FF FF FF FF FF FB   ........
4440  00 00 00 00 00 00 00 00   ........
4448  00 00 00 00 00 00 00 00   ........
4450  00 00 00 00 00 00 00 00   ........

READY
```

LIST the program for reference

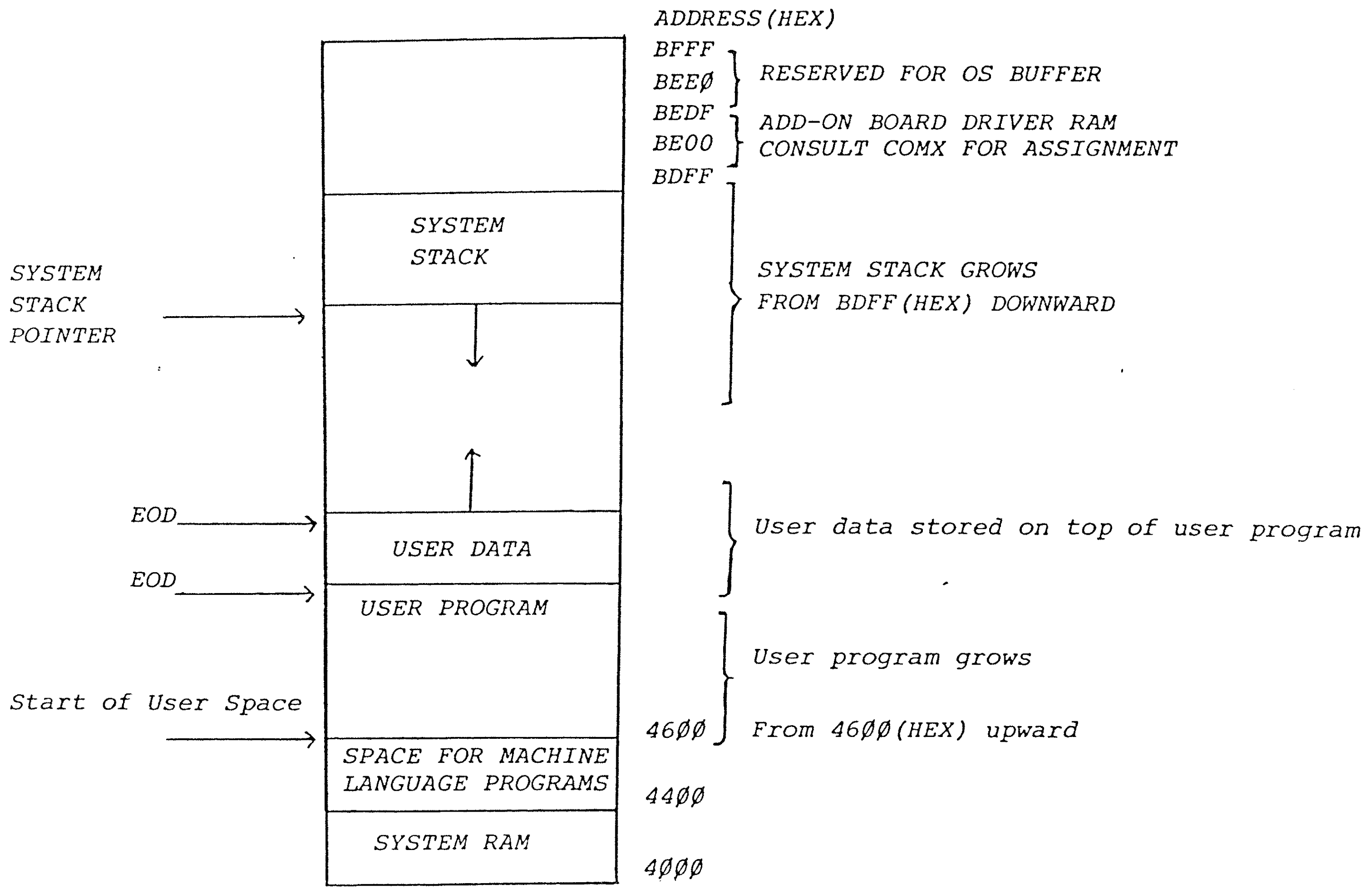
Execute the memory dump program

Memory contents are shown as 8 HEX digits pairs with the address of the first hex pair shown on the leftmost column.

ASCII character are shown on rightmost column for reference

i.e. The ASCII code for 'F' is 46
The ASCII code for 'I' is 49

The new memory map is shown below :



The cassette PLOAD and PSAVE command will load and save all memory contents between 44ØØ(HEX) and the memory location pointed by the "End of Program" pointer. It will also set all appropriate system pointers.

Hence, the machine language and BASIC program can be stored to cassette by one PSAVE command and get back from cassette by one PLOAD command. The "Start of User Space" pointer will be set automatically.

```
:DEFUS @4600

READY
:10 REM THIS PROGRAM STARTS AT @4600
```

Define new "Start of User Space"

__BEFORE__ entering the BASIC program.

The BASIC program was entered.

```
:CALL(@C10F)

MODIFY ADDRESS: 4400
4400   02   D4
4401   FF   32
4402   FF   2B
4403   FF   F4
4404   FF   F5
4405   FF   F6
4406   FF   F7
4407   FF   20
4408   FF   00
4409   FF   D5
440A   FF

ERR CODE 0
READY
:CALL(@C040,7)

START:4400
END  :4418
```

The machine language program was entered and stored at memory location 4400(HEX)

User can also use the POKE command to store machine language program in the memory.

```
4400   D4 32 2B F4 F5 F6 F7 20   .2+.....
4408   00 D5 FF FF FF FF 03 04   ........
4410   0D FF FF FF FF FF FF FF   ........
4418   FF FF FF FF FF FF FF FF   ........

READY
```

Memory dump showing the machine language program

```
:CALL(@C040,7)

START:4600
END  :4640


4600   FF 00 33 FF FF FF FF FF   ..3.....
4608   FF FF FF FF 00 0A 20 80   ........
4610   20 54 48 49 53 20 50 52    THIS PR
4618   4F 47 52 41 4D 20 53 54   OGRAM ST
4620   41 52 54 53 20 41 54 20   ARTS AT 
4628   40 34 36 30 30 0D FF FF   @4600...
4630   03 04 0D FF FF FF FF FF   ........
4638   FF FF FF FF FF FF FF FF   ........
4640   00 00 00 00 00 00 00 00   ........
```

Memory dump showing the BASIC language program

3. ADDING MACHINE LANGUAGE PROGRAM TO A NEW BASIC PROGRAM

Before the development of a new BASIC language program starts, the programmer must estimate whether machine language routine is needed in his program. Appropriate memory space MUST be allocated by the DEFUS command BEFORE he type in the first line of BASIC language program. For example, typing "DEFUS 4600" will allocate 512 bytes.

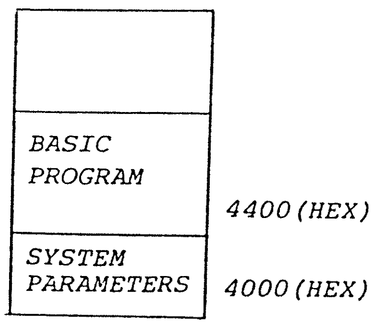
The COMX 35 has large amount of RAM and it is advisible to allocate 256 or 512 butes for machine language routines in every program. It costs you nothing other than about 10 more seconds of cassette I/O time.

4. ADDING MACHINE LANGUAGE PROGRAM TO EXISTING BASIC LANGUAGE PROGRAM

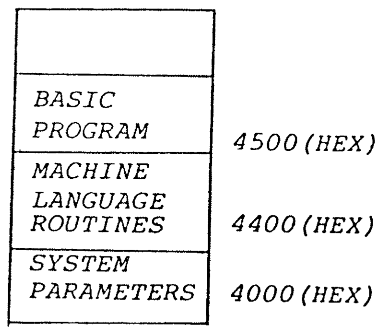
The simplest method is to obtain a hard copy of the program; define new "Start of User Space" and type in the whole program once more. However, this method is a time-consuming job especially when the program size is considerable large.

A second method is to copy the BASIC program to a new location higher up in the memory space and then change the appropriate system pointers to point to the relocated BASIC language program.

For example, we can copy the program 256 bytes upward and leave that amount of space for machine language program.



ORIGINAL BASIC PROGRAM



RELOCATED BASIC PROGRAM

4.1 System Pointers

The BASIC interpretor keeps FIVE 16-BIT system pointers to trace the location of the BASIC language program in the memory space.

Address (hexidecimal)	Functions
4281 4282	Start of User Space in memory 4281 is the HIGH byte 4282 is the LOW byte
4283 4284	End of User Space in memory This value indicate the first byte left unused by the user's BASIC program Program data storage starts at this location and extends upward. Its value is displayed (in hexidecimal) by typing the 'EOP' command 4283 is the HIGH byte 4284 is the LOW byte
4292 4293	End of array and start of string 4292 is the HIGH byte 4293 is the LOW byte
4294 4295	Start of arrays 4294 is the HIGH byte 4295 is the LOW byte
4299 429A	End of Variable This marks the first free byte left unused by the user's program and user's data. User can have free access to memory lies between this pointer and the system stack pointer (Register 2 of CPU).

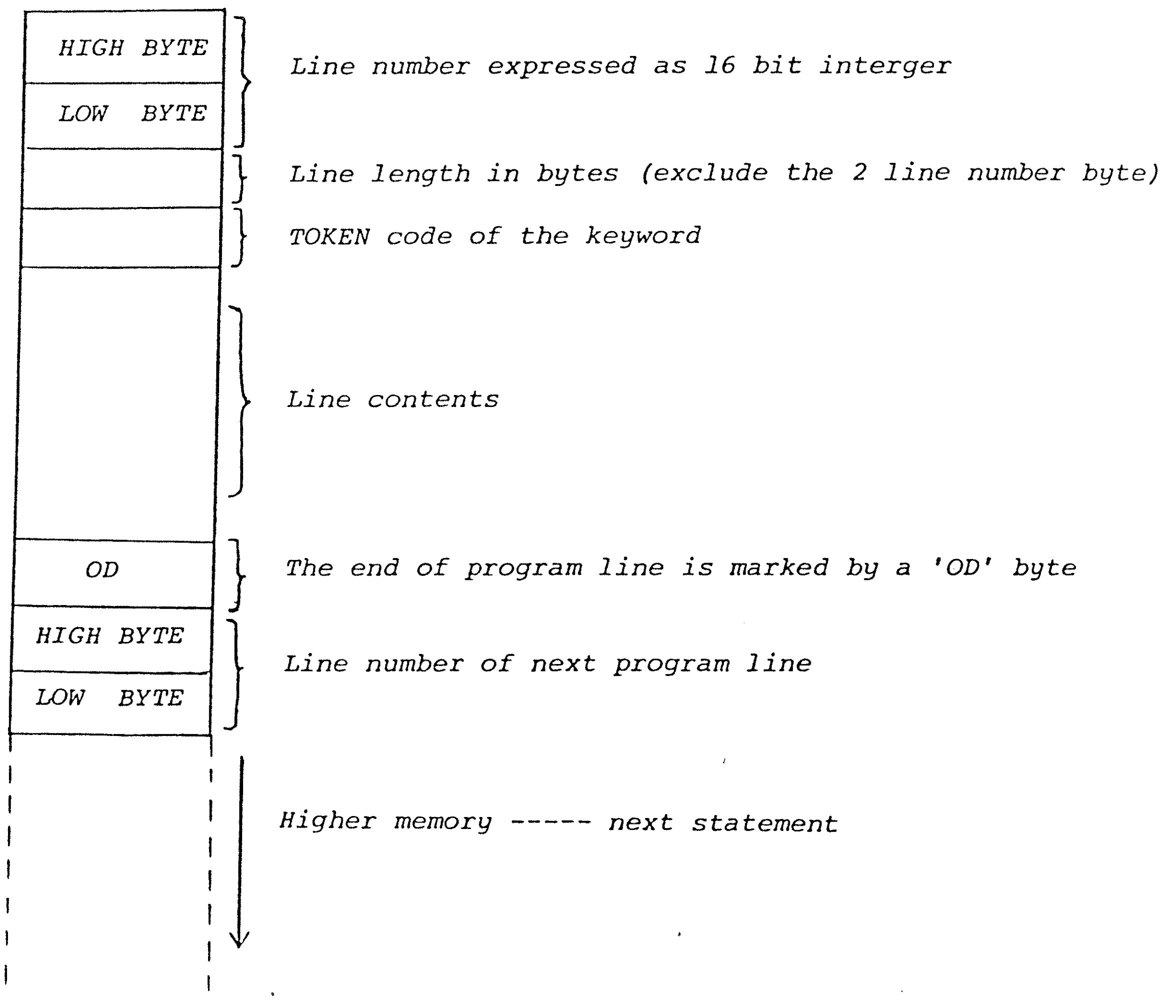
4.2 Relocation of a BASIC program using utilities available from COMX 35 Standard Printer Interface Card

In the following illustration, a sample program was typed in and stored at the default memory location. It was then decided that a machine language subroutine was needed. The BASIC program was relocated to 256 byte higher in the memory and the 5 system pointers was changed accordingly. The machine language subroutine was then entered and new lines were added to the BASIC language program to call the new machine language subroutine.

The relocation process was illustrated in a stepwise manner and explainations were given on each step.

STEP 1 : LIST the program for reference

The storage format of a BASIC language line is shown below :


HIGH BYTE
LOW BYTE
Line number expressed as 16 bit interger
Line length in bytes (exclude the 2 line number byte)
TOKEN code of the keyword
Line contents
OD
The end of program line is marked by a 'OD' byte
HIGH BYTE
LOW BYTE
Line number of next program line
Higher memory ----- next statement

```
:LIS

10 PRINT "THIS DEMOSTRATION PROGRAM WAS
STORED AT DEFAULT MEMORY LOCATION ."
20 PRINT
30 PRINT "IT STARTED AT LOCATION @4400 A
ND EXTEND UPWARD IN MEMORY SPACE ."
40 PRINT
```

List the sample program for reference

```
READY
:EOP
@44AA
READY
:EOD
@44B3
READY
:CALL(@C040,7)
```

Find out the values of "End of Progra and "End of Data" pointer to determin the memory dump range.

Execute the memory dump program

```
START:4400
END  :44FF


4400  00 00 AA FF FF 73 FF FF  ........
4408  FF FF FF FF 00 0A 46 86  ......F.
4410  CF 54 48 49 53 20 44 45  .THIS DE
4418  4D 4F 53 54 52 41 54 49  MOSTRATI
4420  4F 4E 20 50 52 4F 47 52  ON PROGR
4428  41 4D 20 57 41 53 20 53  AM WAS S
4430  54 4F 52 45 44 20 41 54  TORED AT
4438  20 44 45 46 41 55 4C 54   DEFAULT
4440  20 4D 45 4D 4F 52 59 20   MEMORY
4448  4C 4F 43 41 54 49 4F 4E  LOCATION
4450  20 2E CE 0D 00 14 03 86   .......
4458  0D 00 1E 45 86 CF 49 54  ...E..IT
4460  20 53 54 41 52 54 45 44   STARTED
4468  20 41 54 20 4C 4F 43 41   AT LOCA
4470  54 49 4F 4E 20 40 34 34  TION @44
4478  30 30 20 41 4E 44 20 45  00 AND E
4480  58 54 45 4E 44 20 55 50  XTEND UP
4488  57 41 52 44 20 49 4E 20  WARD IN
4490  4D 45 4D 4F 52 59 20 53  MEMORY S
4498  50 41 43 45 20 2E CE 0D  PACE ...
44A0  00 28 03 86 0D FF FF 03  .(......
44A8  84 0D FF FF FF FF FF FF  ........
44B0  FF FF FF FF FF FF FF FF  ........

44B8  FF FF FF FF FF FF FF FF  ........
44C0  00 00 00 00 00 00 00 00  ........
44C8  00 00 00 00 00 00 00 00  ........
44D0  00 00 00 00 00 00 00 00  ........
44D8  00 00 00 00 00 00 00 00  ........
44E0  00 00 00 00 00 00 00 00  ........
44E8  00 00 00 00 00 00 00 00  ........
44F0  00 00 00 00 00 00 00 00  ........
44F8  00 00 00 00 00 00 00 00  ........
```

The BASIC language program is stored at memory 440C(HEX) and grows upward

4400 to 440B are reserved for system use

4401 and 4402 are the program length expressed as a 16 bit integer.
(4401 is the HIGH byte)
440C and 440D are 16 bit line number
(440C is HIGH byte)
440E is the byte count of a BASIC line (excluding the line number)
440F is the TOKEN-CODE
4453 is the 'end of line' character.

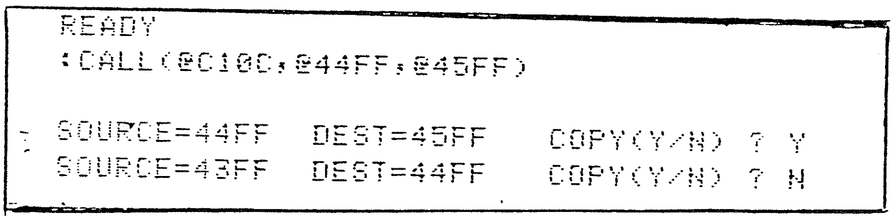
STEP 2 : *The copy program will copy 256 byte of memory at one time. The last copied byte must be 4400(HEX) and hence the source and destination address must be multiple of 256 byte. That is, the low byte must be equal to FF(HEX) and the last copied address will be ??ØØ.*

EXAMPLE

'End of Data' pointer value	*Starting Source Address*
4420	44FF
44BF	44FF
44FE	44FF
44FF	44FF
4501	45FF

In our example, EOD is equal to 44BF(HEX) and source address is equal 44FF(HEX)

Hence, memory ranging inclusively from 44FF(HEX) to 4400(HEX) will be copied.



```
READY
:CALL(@C10C;@44FF;@45FF)

SOURCE=44FF  DEST=45FF   COPY(Y/N) ? Y
SOURCE=43FF  DEST=44FF   COPY(Y/N) ? N
```

STEP 3 : The copied memory is dumped for reference.

```
READY
:CALL(@C040,7)

START:4500
END  :4550


4500  00 00 AA FF FF 73 FF FF   ........
4508  FF FF FF FF 00 0A 46 86   ......F.
4510  CF 54 48 49 53 20 44 45   .THIS DE
4518  4D 4F 53 54 52 41 54 49   MOSTRATI
4520  4F 4E 20 50 52 4F 47 52   ON PROGR
4528  41 4D 20 57 41 53 20 53   AM WAS S
4530  54 4F 52 45 44 20 41 54   TORED AT
4538  20 44 45 46 41 55 4C 54    DEFAULT
4540  20 4D 45 4D 4F 52 59 20    MEMORY
4548  4C 4F 43 41 54 49 4F 4E   LOCATION
4550  20 2E CE 0D 00 14 03 86    .......
```

STEP 4 : The 'Memory examine and modify' routine is used to change 5 system pointers to point back to the relocated BASIC program.

The BASIC program has been moved up 256 bytes and hence all these pointers must be increased by 256 (i.e 0100 hexidecimal)

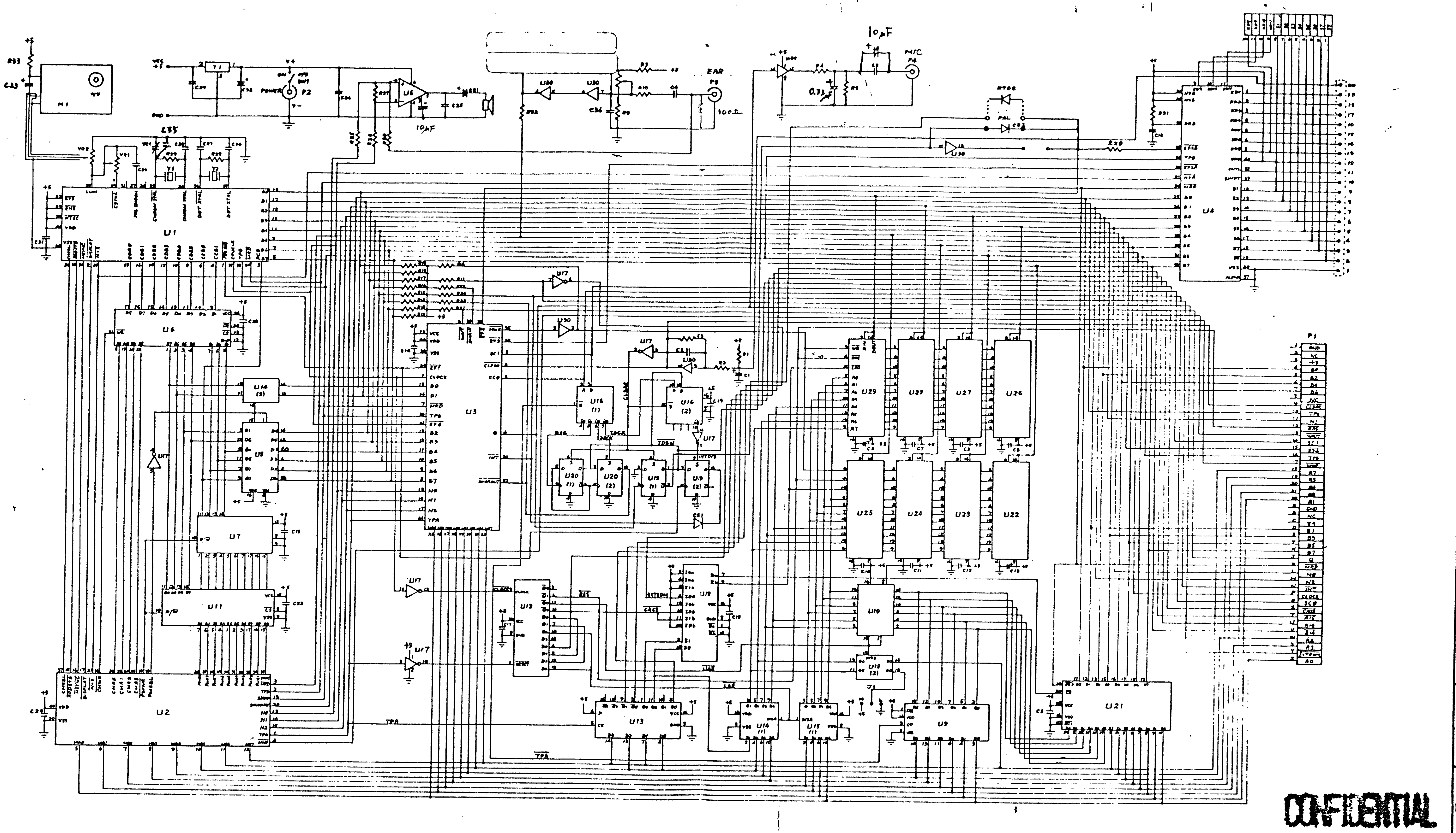
```
READY
:CALL(0C10F)

MODIFY ADDRESS: 4280
4280  86
4281  44  45          Start of User Space (HIGH Byte)
4282  0C              End of User Space   (HIGH Byte)
4283  44  45
4284  AA
4285  00
4286  28
4287  D0
4288  FF
4289  FF
428A  D4
428B  C1
428C  03
428D  D5
428E  D4
428F  C1
4290  00
4291  D5
4292  44  45          End of array/Start of string (HIGH Byte)
4293  B3
4294  44  45          Start of array (HIGH Byte)
4295  B3
4296  FF
4297  FF
4298  00
4299  44  45          End of variables  (HIGH Byte)
429A  B3
429B  06

ERR CODE 0
READY
```

STEP 5 : EOP and EOD is shown for reference
LIST will shown that the program has been relocated successfully.

Note : If the user modifies the system pointers to some wrong value, LIST may still work but the system will crash on EDITing.

New lines are then added and they are automatically appended to the end of the relocated BASIC program.

EOP and EOD are automatically incremented as the program grows.



CONFIDENTIAL

TOLERANCES UNLESS STATED DECIMAL ± ANGULAR ±	DO NOT SCALE THIS DRAWING

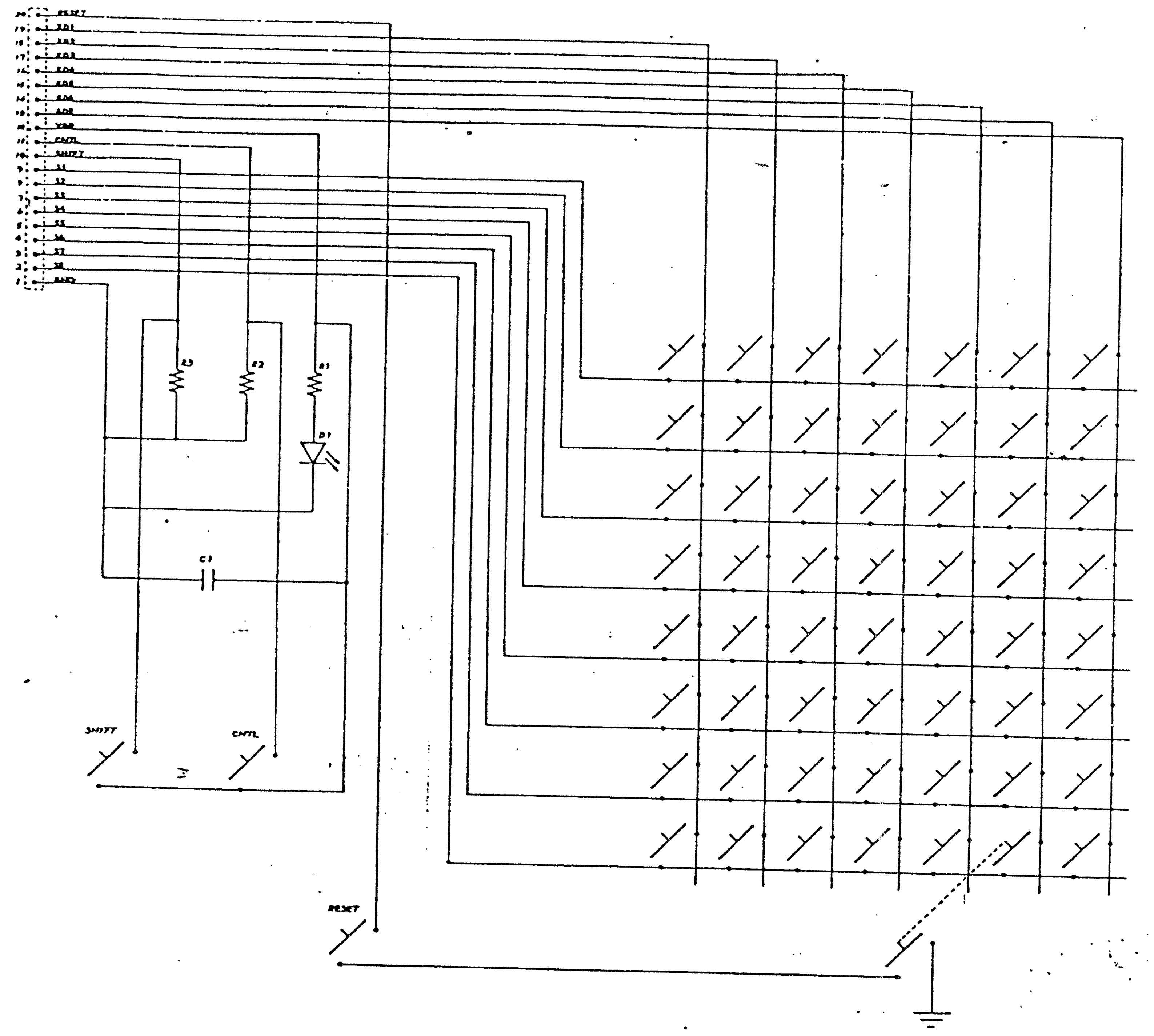
U1	CDP1870	U9	40174	U17	4049	U25	MSM3732	R1	27K	R9	75K	R17	10K		
U2	CDP1869	U10	4503	U18	74LS153	U26	MSM3732	R2	10K	R10	1K	R18	10K	R33	30Ω
U3	CDP1802	U11	2114	U19	4013	U27	MSM3732	R3	100K	R11	10K	R19	10K	R25	10K
U4	CDP1871	U12	74LS175	U20	4013	U28	MSM3732	R4	1K	R12	10K	R20	10K	R26	1K
U5	LM386	U13	4042	U21	6/9128	U29	MSM3732	R5	700Ω	R13	10K	R21	10K	R27	3K
U6	HM6116	U14	4503	U22	MSM3732	U30	4050	R6	10K	R14	10K	R22	10K	R28	22M
U7	2114	U15	4503	U23	MSM3732					R15	10K	R23	10K	R29	22M
U8	4503	U16	4555	U24	MSM3732	T1	LM341	R8	76K	R16	10K	R24	10K	R30	3M
														R31	270K
														R32	10K

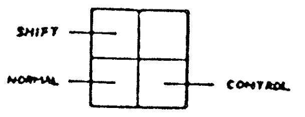
C1	1μF
C2	120PF
C4~C5	0.1μF
C21	220μF
C22~C25	0.1μF
C26~C27	30PF
C28	100PF
C29,C34	0.1μF
C30	30PF
C33	47μF 16V
CR1~CR2	IN914
VR1 VR2	10K
M1	1233236
Y1	8.867236M
Y2	5.626M
VC1	5-30PF
P1	186-412-08
C6,C13,C33	47μF/10V
C7~C12	0.1μF
C14~C20	0.1μF
VR3	100K
C3	0.1μF + 10μF
C35	20pF
C36	0.1μF

※ NOTE: J1
For 3732L INSTALL JUMPER L
For 3732M INSTALL JUMPER M

COMX world operations ltd.

SYSTEM	COMX 35		MODEL NO.	T-001
SUBSYSTEM	BASIC UNIT-MAINBOARD CIRCUIT		DRAWING NO.	T-001-MB
MATERIAL			UNIT	
FINISH			SCALE	
DESIGN	J. 10	3-1-85	DRAWN	David
CHECKED			CLASS	C
APPROVED	[illegible]	11-1-85	REV.	1





	D1		D2		D3		D4		D5		D6		D8
	DOT SP		[		40		48		50		58		CR
S1	0		8		?	00	H	08	P	10	X	18	
	!		]		41		49		51		59		ESC
S2	1		9		A	01	I	09	Q	11	Y	19	
	"		:		42		4A		52		5A		↑
S3	2		.		B	02	J	0A	R	12	Z	7A	
	#		;		43		4B		53		7B		→
S4	3		,		C	03	K	0B	S	13	+	1B	
	$		(		44		4C		54		7C		←
S5	4		<		D	04	L	0C	T	14	−	1C	
	%		^		45		4D		55		7D		↓
S6	5		=		E	05	M	0D	U	15	*	1D	
	&		)		46		4E		56		7E		DEL
S7	6		>		F	06	N	0E	V	16	/	1E	
	'		_		47		4F		57		7F		
S8	7		\		G	07	O	0F	W	17	SP	1F	

CONFIDENTIA

TOLERANCES UNLESS STATED DECIMAL ± ANGULAR ±	DO NOT SCALE THIS DRAWING

REVISIONS

R1	470 Ω
R2	10K
R3	10K
D1	LED
C1	0.1 μF

COMX world operations H

SYSTEM	COMX 351		MODEL	
			7-001	
SUBSYSTEM	BASIC UNIT—KEYBOARD CIRCUIT		DRAWING	
			7-001-K	
MATERIAL			UNIT	
FINISH			SCALE	
DESIGN	1.1o	1-1-15	DRAWN	D
CHECKED			CLASS	
APPROVED	[illegible]	12-1-13	REV.	