AN INTERFACE CARD FOR MULTILINGUAL PROCESSING

SUBMITTED BY :

TAHERI SAIFEE

GUIDE :

Dr. P. C. SAXENA

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI

DECEMBER 1989

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF TECHNOLOGY IN COMPUTER SCIENCE AND TECHNOLOGY OF JAWAHARLAL NEHRU UNIVERSITY , NEW DELHI - 110 067 .

Nitel

CERTIFICATE

This is to certify that the project entitled AN INTERFACE CARD FOR MULTILINGUAL PROCESSING has been successfully completed by Taheri Saifee under the guidance of Mr. Deepak Verma and Dr. P. C. Saxena , at NITEL , Bhopal .

This work has not been submitted to any other institution or university for the award of any degree .

Verma

DEEPAK VERMA
Asstt. Manager,
R&D , NITEL ,
BHOPAL .

Dr. P. C. SAXENA
Asoct. Prof. ,
SC&SS , JNU ,
NEW DELHI .

Prof. N.P. MUKHERJEE
Dean
SC & SS, JNU
NEW DELHI - 110 067.

ACKNOWLEDGEMENTS

I wish to use this opportunity for expressing my gratitude to Mr. Navin Krishnan , MD , NITEL , Bhopal , for allowing me to work on this project and providing me with all the facilities available at NITEL .

The words lose their worth in expressing my indebtness to Dr. P. C. Saxena , my elite guide for his invaluable guidance and encouragemment .

I would like to thank Mr. Deepak Verma , Assisstant Manager R&D , NITEL and Mr. R. P. Garg , Engineer R&D , NITEL for their guidance in design and developmment of this card .

I want to thank my friends Shailesh , Atul and Sandeep for their help and the discussions we had on the topics of our mutual interest .

TAHERI SAIFEE

ABSTRACT

The increasing use of computers in various fields of business has opened up various venues of research in computer use . India being a country with diverse languages needs incorporating multilingual facilities on computers at the earliest . Here a design for a card is proposed and has been tested successfully for printing in Hindi . Slight modifications will result in multilingual word - processor .

CONTENTS

CERTIFICATE

ACKNOWLEDGEMENTS

ABSTRACT

1. INTRODUCTION .... 1.

2. ARCHITECTURE

   2.1 INTRODUCTION .... 4.

   2.2 SELECTION OF COMPONENTS.................... 4.

   2.3 COMPONENTS .... 6.

   2.4 HARDWARE DESCRIPTION .... 7.

3. PROCESSING

   3.1 INTRODUCTION .... 10.

   3.2 SOFTWARE STRUCTURE .... 10.

   3.3 EXCEPTION PROCESSING .... 10.

   3.4 DECODING .... 11.

   3.5 CHARACTER GENERATOR .... 12.

   3.6 CHARACTER GENERATION .... 13.

   3.7 RUNNING THE SOFTWARE .... 13.

4. POSSIBLE DEVELOPMENTS .... 14.

APPENDICES

   A. 8035 .... 15.

   B. 2732 .... 17.

   C. 6264 .... 18.

REFERENCES .... 19.

1. INTRODUCTION

Today computer is not a machine known only to a select few i.e. the scientists , the engineers and the related persons . Even a man of the streets knows what a computer is and what its utilities are . It has become a household item in the developed countries. In our country too , everybody is aware of computers and its capabilities . Though it is still not a household item , but it is widely being used in business and industry .

The use of computers is not only for research work , statistical analysis or business data-processing but for word-processing too . The secretarial job has been made much easier by using wordprocessing machines i.e. word-processors , which are essentially computers.

India is a vast country , with diverse languages. So it becomes utmost necessary that computer be able to understand various languages . But the language on computers is English or English-like only . Also it is difficult , though not impossible to crry out processing in languages other than English . Work is going on in this area . In the mean time , the utmost important thing is that word-processing be carried out in various languages .

The project I have undertaken is a small step in this direction . Many people are working on similar ideas and coming out with various products . The card designed by me is one of them .

The card basically is an interface card for printer . It is designed for news services like BHASHA and VARTA , which are Hindi services . The card acts as an interface for teleprinter line and a printer . The message is transmitted in Baudat code and is printed in Hindi .

Actually the newspapers have dedicated telex lines for news agencies like UNI , PTI , VARTA , BHASHA ,etc. There is no use of connecting teleprinter on these lines because they are costly and very noisy . Also they are used as incoming lines only . So it is better to use a printer which is very less noisy and less expensive too . So a printer with an interface card is the best alternative . The card takes in the codes transmitted on the telex lines and convert them to suitable codes for printer to print .

In case English is the language , the character generator of the printer is sufficient . The card only takes in the codes and convert the voltage levels from +60 V - -60 V to 5 V - 0 V . But in case of Hindi services a character generator on the card is needed and the printer is to be used in the graphics mode .

The interface card designed has a character generator for Hindi alphabets . At present Baudat codes are being used for Hindi transmission . So the software has been written keeping the 5-bit code in mind . Later when IASCII (Indian ASCII) will be used , it can be easily modified to accomodate this 8-bit code .

The design of this card can also be used for multilingual word-processing on computers . How can this be done will be explained in the chapters that follow . This card can be said to be the foundation stone for multilingual processing .

2 . ARCHITECTURE

2.1 INTRODUCTION

The basic function of the interface card is to convert the voltage levels of the telex line to that of the digital circuit . This is so because the telex line levels are +60 V to -60 V whereas those for the digital circuit are +5 V to 0 V . Then the need is for receiving the data , storing them , simultaneously processing them and then output them to the printer . This is the problemm definition .

2.2 SELECTION OF COMPONENTS

For converting the voltage levels , the module needed is shown in FIG. 1 . The MCT2E is the opto - isolator / coupler . Its basic work is to separate the two grounds . Here it seperates those for telex lines and that of the circuit . It helps change the voltage level to that for digital circuit . Its working is based on the opical principles .

For receiving and decoding the code and for character generator , the configuration neede is - a CPU , memory and I/O device . For this sort of work any

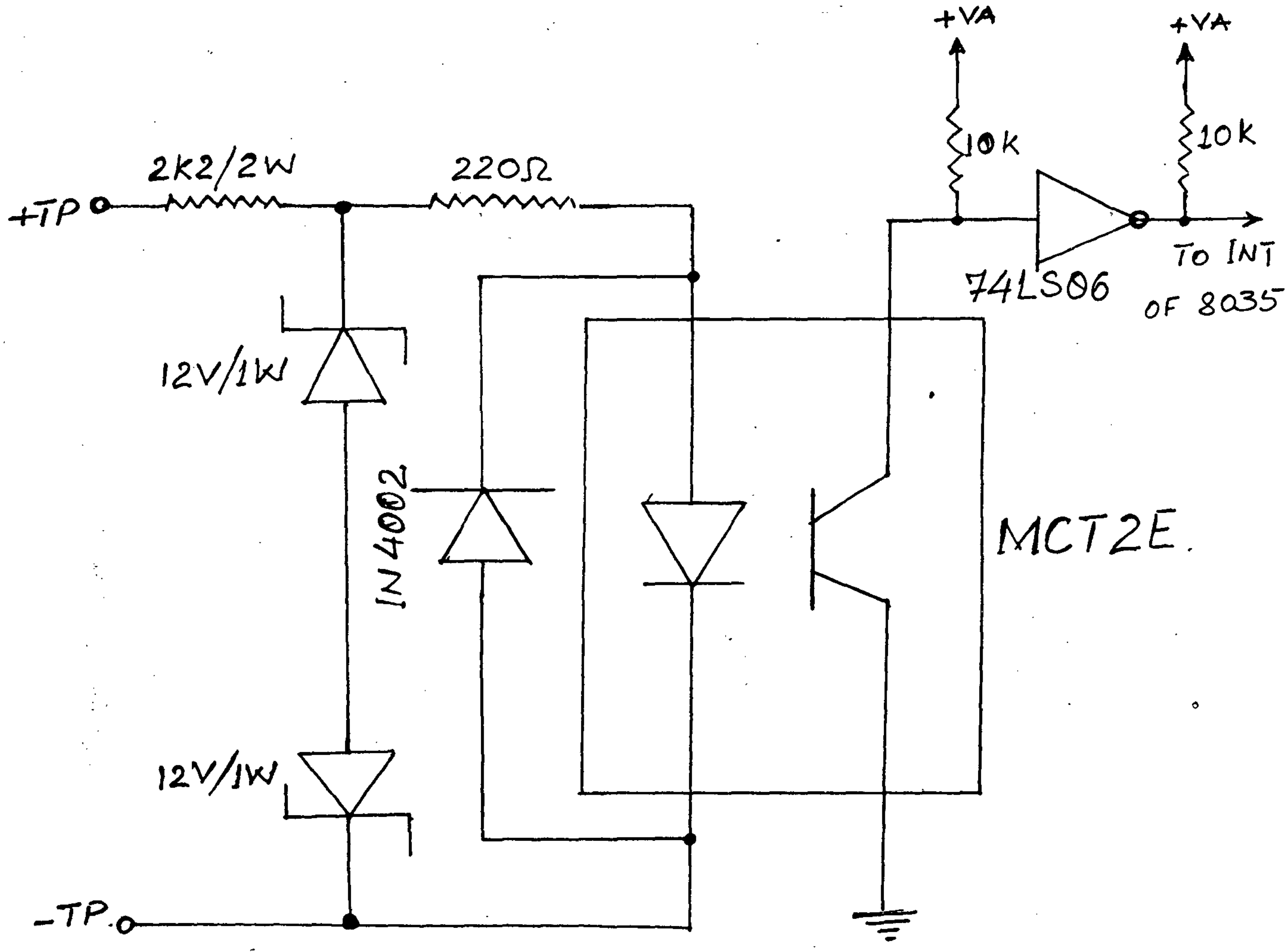

+VA
+VA
10k
10k
2K2/2W
220Ω
+TP
74LS06
To INT
OF 8035
12V/1W
IN4002
MCT2E.
12V/1W
-TP.

FIGURE - 1.

8-bit microprocessor may be used . 8-bit because , the code transmitted is 5-bit and so the processing will not exceed 8 bits at all . The memory needed will be RAM and ROM . ROM will have the program for receiving , decoding and transmitting the data and RAM will be needed for temporary storage during processing . Instead of ROM it is better to use EPROM because the program can be fused into it when desired and the character generator can be used as desired , like Hindi in one case , in another it might be some other ; while the programm remains the same .

In this application the microprocessor will be a disadvantage because alongwith a microprocessor it will be necessary to use some I/O devices as well . Like if we use 8085 microprocessor , then we will have to use either 8155/56 or 8255 as I/O device . Instead it will be much better to use an intelligent controller i.e. a microcontroller .

A microcontroller has a CPU , input-output ports , RAM and may also have ROM/EPROM . So for a dedicated purpose it is better to use a microcontroller . The microcontroller used here is of MCS-48 family from INTEL . It is 8035 , H-MOS Single - Component 8-bit

Microcomputer . (Refer APPENDIX A for details .)

MEMORY

For program and character generator storage a ROM is needed . More precisely an Erasable Programmable ROM is needed . 2732 is chosen . It is a 32k (4k x 8) UV EPROM . 32k is sufficient for our purpose . (Refer APPENDIX B for details .)

For RAM , **6264** , a 8k Byte RAM is selected . (Refer APPENDIX C for details .)

2.3 COMPONENTS

The components needed are :

S.NO.	NAME	DESCRIPTION	QTY
1.	MCS-8035	Single-Component 8-bit Microcomputer	1
2.	2732	32k (4k x 8) UV EPROM	1
3.	6264	64k (8k x 8) Integrated RAM	1
4.	MCT2E	Opto-isolator / coupler	1
5.	1489	Line - driver	1
6.	74LS373	Octal TRI - STATE Transparent D - Latches	2
7.	74LS244	Octal TRI - STATE Buffers / Line drivers / receivers	1

8.	74LS74	Dual D Positive - Edge - Triggered flip - flops with Preset and Clear	1
9.	74LS32	Quad 2 - input OR gates	1
10.	74LS06	Hex inverters with Open Collector outputs	1
11.	74LS04	Hex inverters	2
12.	74LS00	Quad 2 - input NAND gates	1
13.	DIP Switch	_	1
14.	20 Pin CONN.	_	1
15.	6 Pin CONN.	_	1
16.	5 Pin CONN.	_	1
17.	3 Pin CONN.	_	1
18.	Oscillator	4 MHz Peizo Crystal	1

2.4 HARDWARE DESCRIPTION

For making telex line levels compatible to digital levels opto - isolator is used .

Since the address and data bus of 8035 are multiplexed , a latch is needed for demultiplexing . The lines DB0 - DB7 are used by both 2732 and 6264 . P20 - P23 are to be used only by 2732 , while P24 - P27 are to be used by 6264 only . So while 2732 is being addressed P24 - P27 is 0000 and while 6264 is being addressed then P20 - P23 is 0000 . This fixes the address for 2732 and 6264 .

The logic used for generating the enabling signals for the 3732 and 6264 is clear from the FIG. 2 . When 2732 is to be used , the CE and RD for 6264 are false and when 6264 is to be used then OE and CS for 2732 are false .

The DIP Switch is used while testing the circuit , using a microprocessor or a computer . During testing 1489 is used which receives data and transmits to 8035 . The DIP Switch settings invoke the corresponding test programs .

For communicating with the printer a 20 pin connector is used . A latch is used to latch in the data being given by port 2 of 8035 (P0 - P7) .

The architecture is based on exception - processing mode . As soon as the START bit is obtained the INT pin of 8035 goes low and activates the exception routine . This routine takes in data and stores it in 6264 and waits for more data . In case the INT line remains high after STOP bit of 1 1/2 bits then the exception is over and 8035 returns to normal mode . The data is taken up and 16 bytes of character

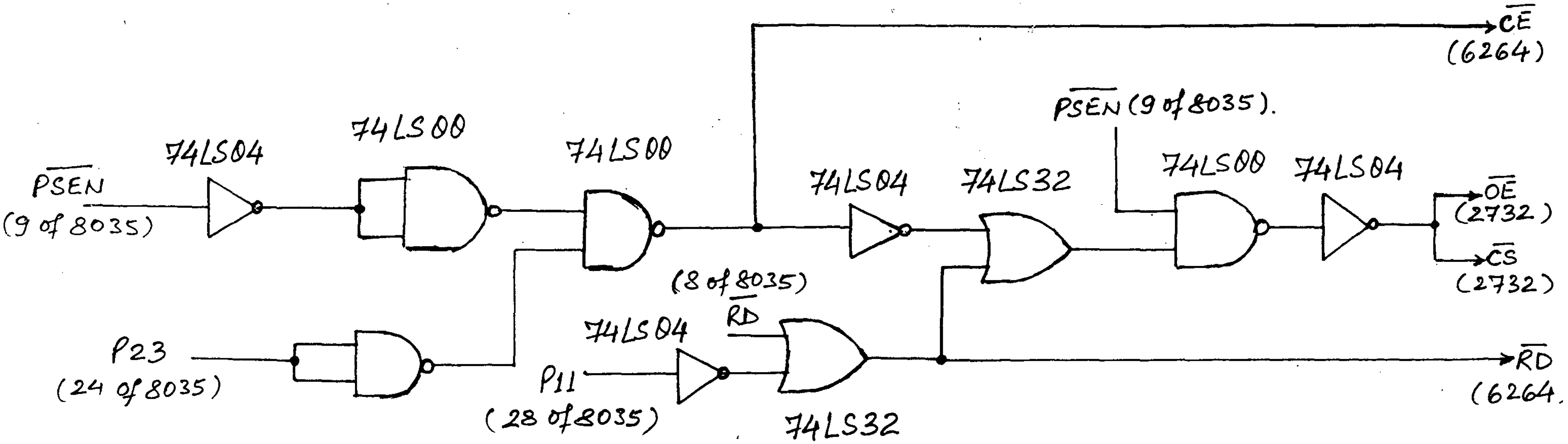
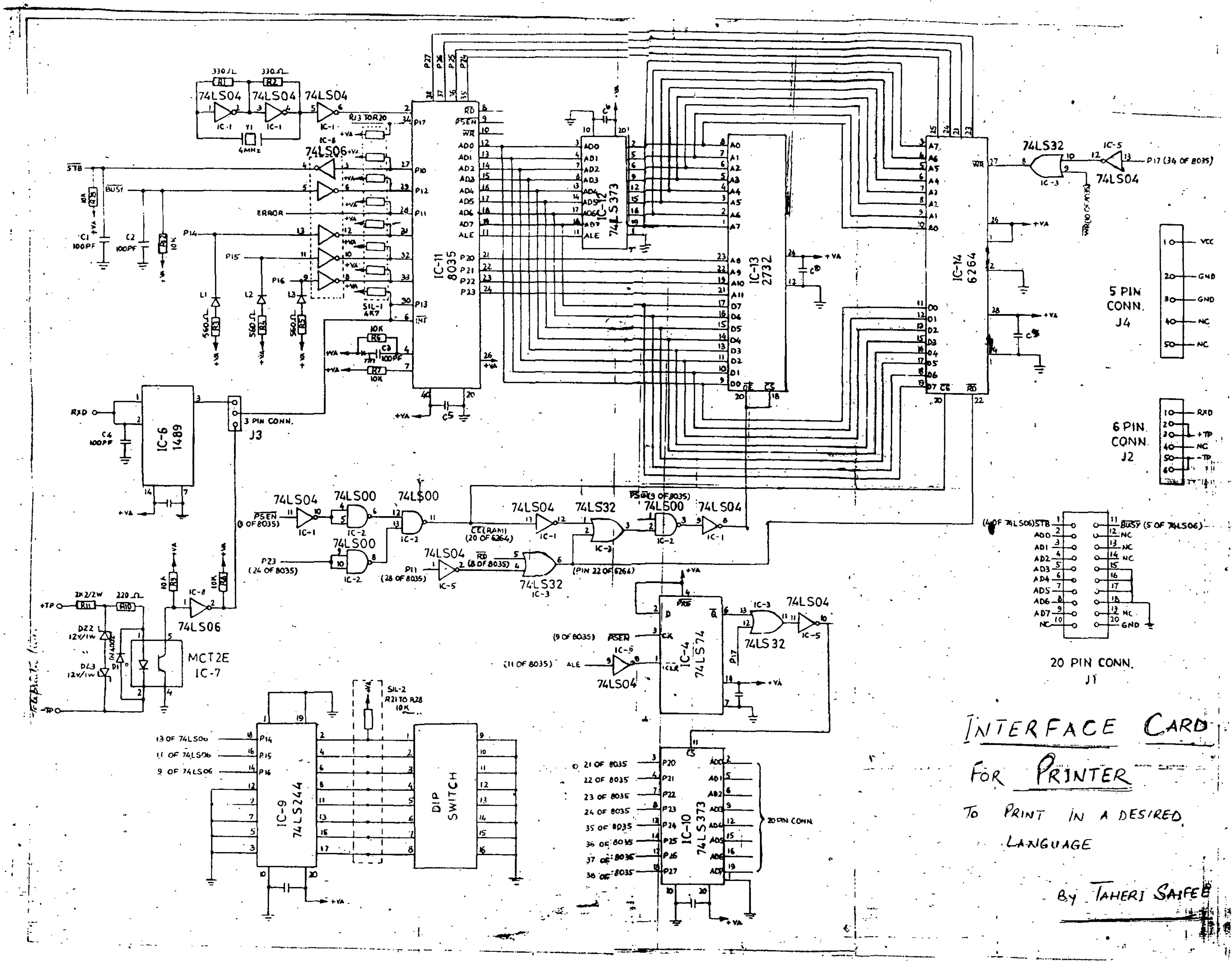

$\overline{PSEN}$
(9 of 8035)
74LS04
74LS00
74LS00
P23
(24 of 8035)
$\overline{CE}$
(6264)
74LS04
74LS32
$\overline{PSEN}$ (9 of 8035).
74LS00
74LS04
$\overline{OE}$
(2732)
$\overline{CS}$
(2732)
(8 of 8035)
$\overline{RD}$
74LS04
P11
(28 of 8035)
74LS32
$\overline{RD}$
(6264.

FIGURE-2



INTERFACE CARD FOR PRINTER

TO PRINT IN A DESIRED LANGUAGE

By TAHERI SAIFEE

3 . PROCESSING

3.1 INTRODUCTION

The input to the card is the Baudat code on telex line . The output needed is the 16 byte word for the corresponding character . This output is for printer which then prints it in dot - matrix mode . The software takes in the data and processes it to find the corresponding character and outputs it to the printer , which then prints it .

3.2 SOFTWARE STRUCTURE

The input of data is done in interrupt driven mode . The rate of transmission is 50 bauds . So after every time interval equal to 50 baud - rate , the incoming data drives the interrupt pin low and the data is taken in . During this the register bank is switched and the data is taken in and stored and then the controller returns to its work suspended before exception processing .

3.3 EXCEPTION PROCESSING

As soon as the START bit comes the INT line goes low and exception routine starts . The data is taken in through the INT pin . After taking in 5 bits the STOP bit arrives which is 1 1/2 bits long and is a high . If INT remains high then the exception routine is over and the controller returns to normal processing .

The data input is stored at a prticular location in RAM . The next data is stored at the consecutive location . This is the exception routine whose work is to input the data and store it .

3.4 DECODING

The main loop of the programm checks to see if there is any data to be processed . If not then it re - enters the check loop , otherwise processes the data obtained .

The data received is first checked to see if it is a Figure - shift (FS) or a Letter - shift (LS) .If it is a LS then the corresponding code is accepted to be an alphabet and if it is a FS then the code is accepted to be a figure like matra , halant , or numeral , etc .

If LS is received then the next data is taken up from the next location and the corresponding character is taken up from the character generator and kept to be output .

If FS is received then the next data is taken up from the corresponding location , its character word taken up from the character generator and ORed with those of the previous character .

So the scheme is as follows :

First FS or LS is determined . If ls then the code is used as an index for character generator for characters and the sixteen bytes obtained are kept in a variable . Then the next data is checked . If it is LS then the previous sixteen bytes are sent to the printer .

If FS is obtained then the code is used as an index for character generator for figures . the corresponding sixteen bytes are taken and ORed with the previous sixteen bytes . Then they are stored and the next data is examined similarly . Until LS is obtained ORing takes place .

3.5 CHARACTER GENERATOR

The Baudat code is a 5 - bit code . So a maximmum of 32 characters can be coded . But using FS and LS there can be atmost 62 characters i.e. 31 letters and 31 figures .

The letters are : अ, क, ख, ग, घ, च, छ, ज, झ, ट, ठ, ड, ढ, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व, श, स, ह, क्ष, ज्ञ.
The figures are : ा, े, ै, ो, ौ, ि, ी, ', :, ।, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, ट.

For printing numerals , first LS is sent then 00 , then FS and then the code for numeral .

For printing words like मा , first LS then code for म then FS then code for ा . The letters like इ, ई, ए, ऐ are written as िअ, अी, अे, अै.

3.6 CHARACTER GENERATION

The characters are generated in a matrix of 16 x 8 i.e. 16 rows by 8 columns . And while printing the printer is used in a compressed mode .

The formation of some of the characters is shown in the adjoining figures .

3.7 RUNNING THE SOFTWARE

As soon as the supply is switched on the programm starts . Initially a loop is executed which checks if any data has been received . Exception routine takes in the data and normmal processing decodes and outputs to the printer .

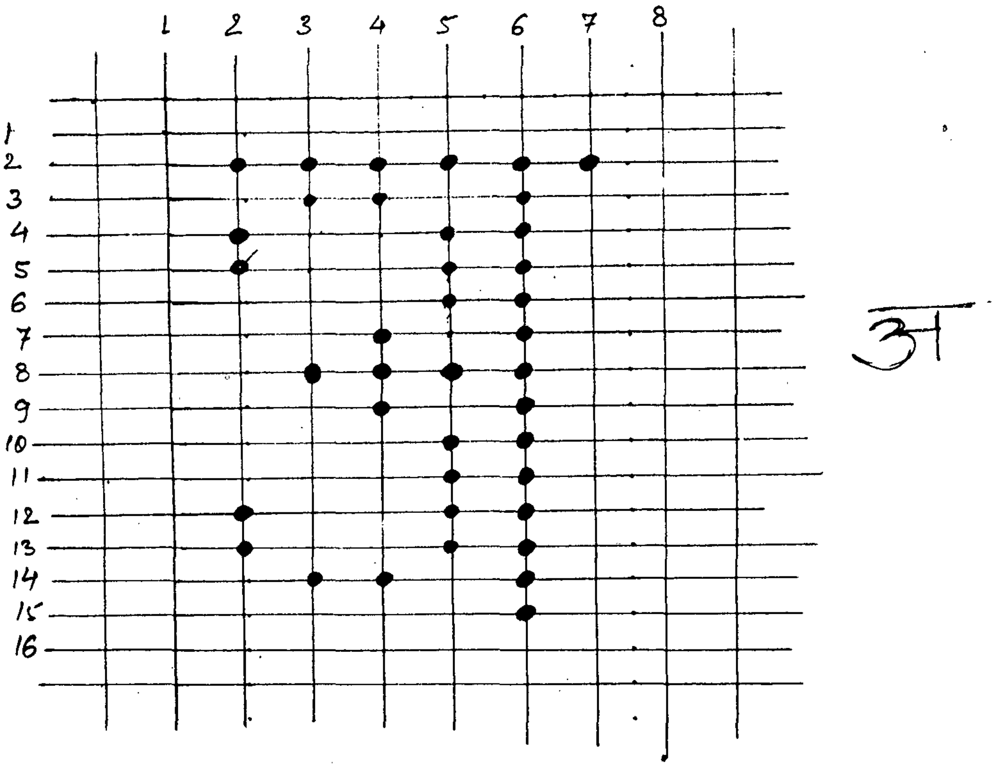


FIGURE -3.

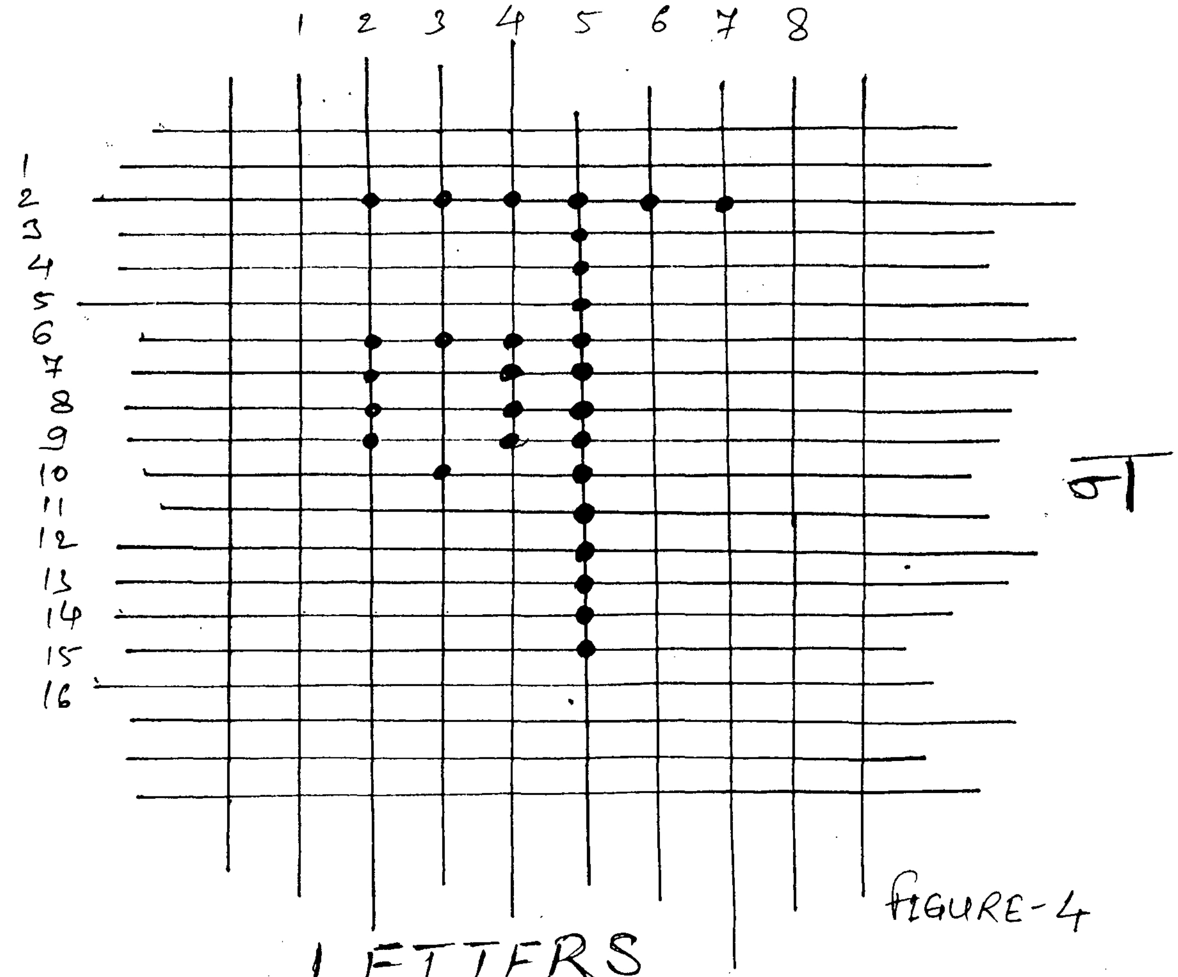


FIGURE-4

LETTERS

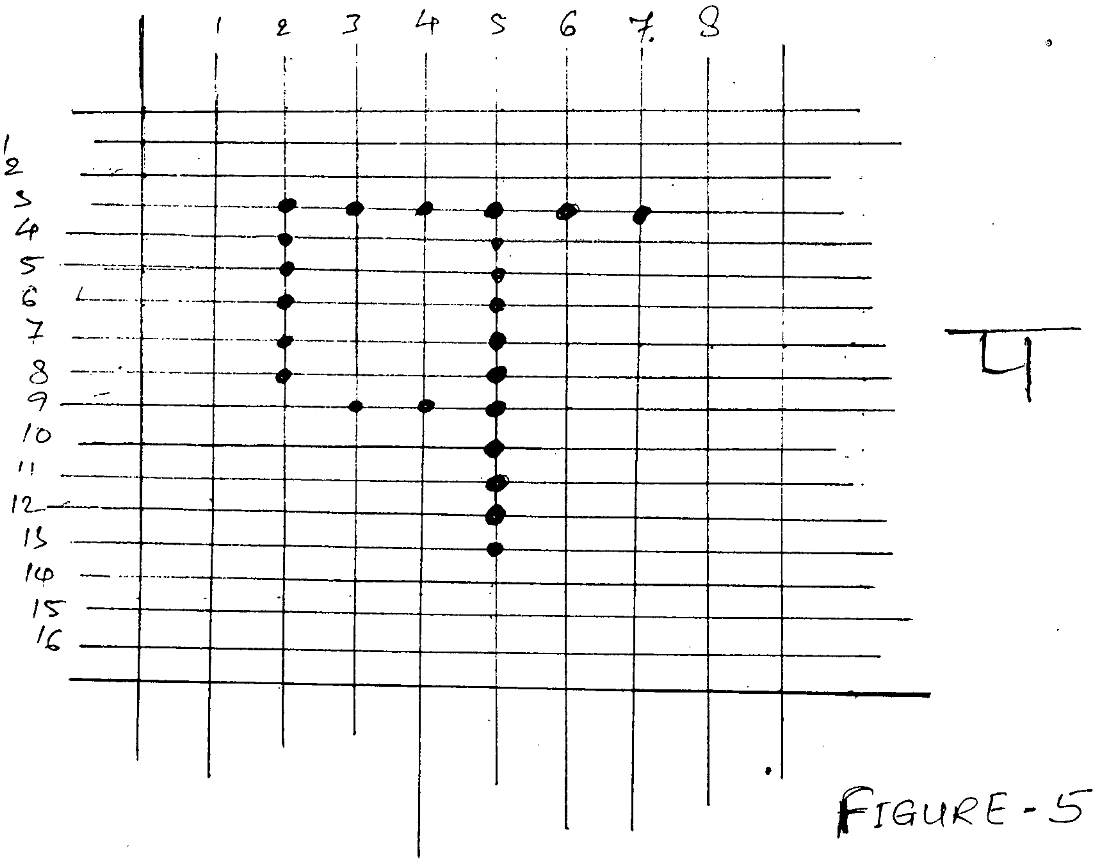


FIGURE-5

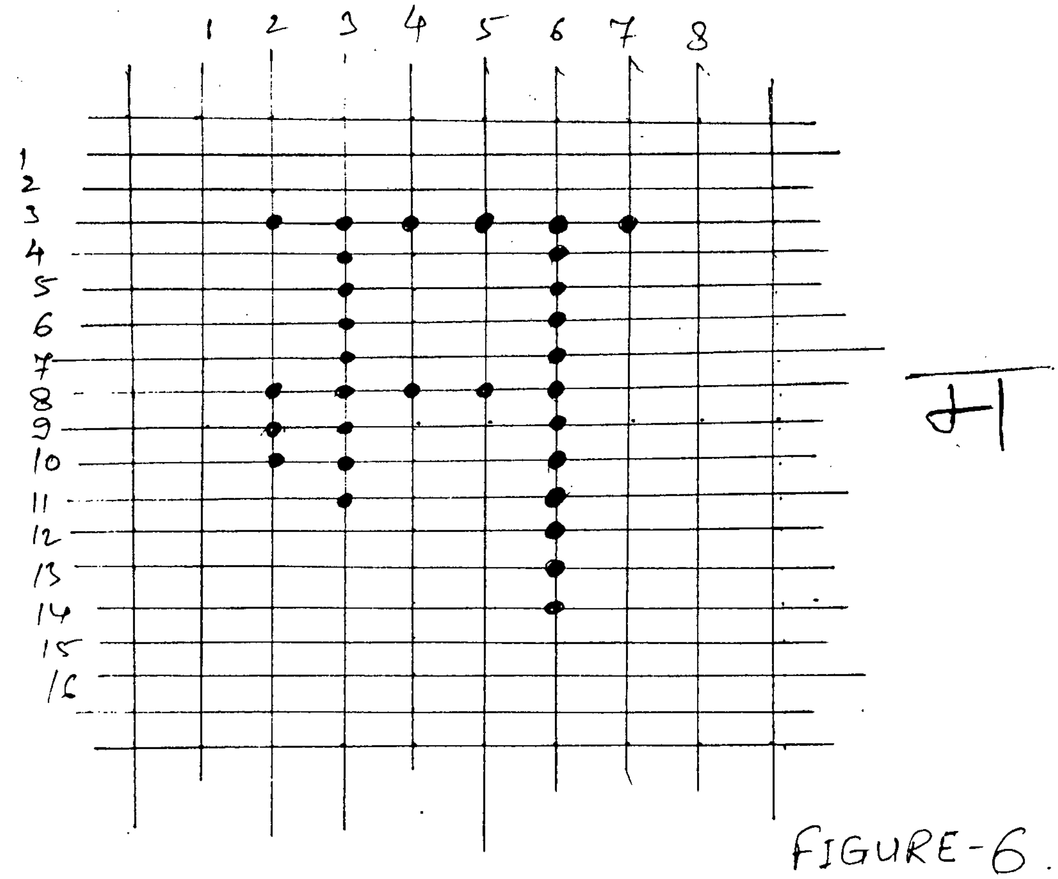
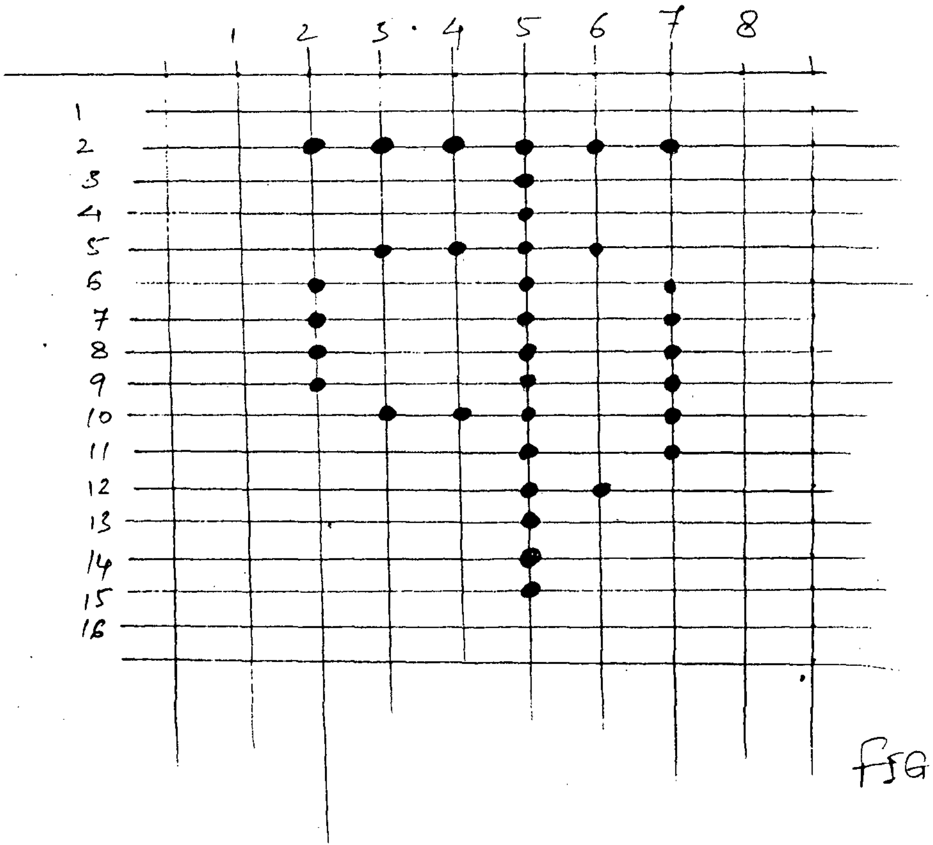


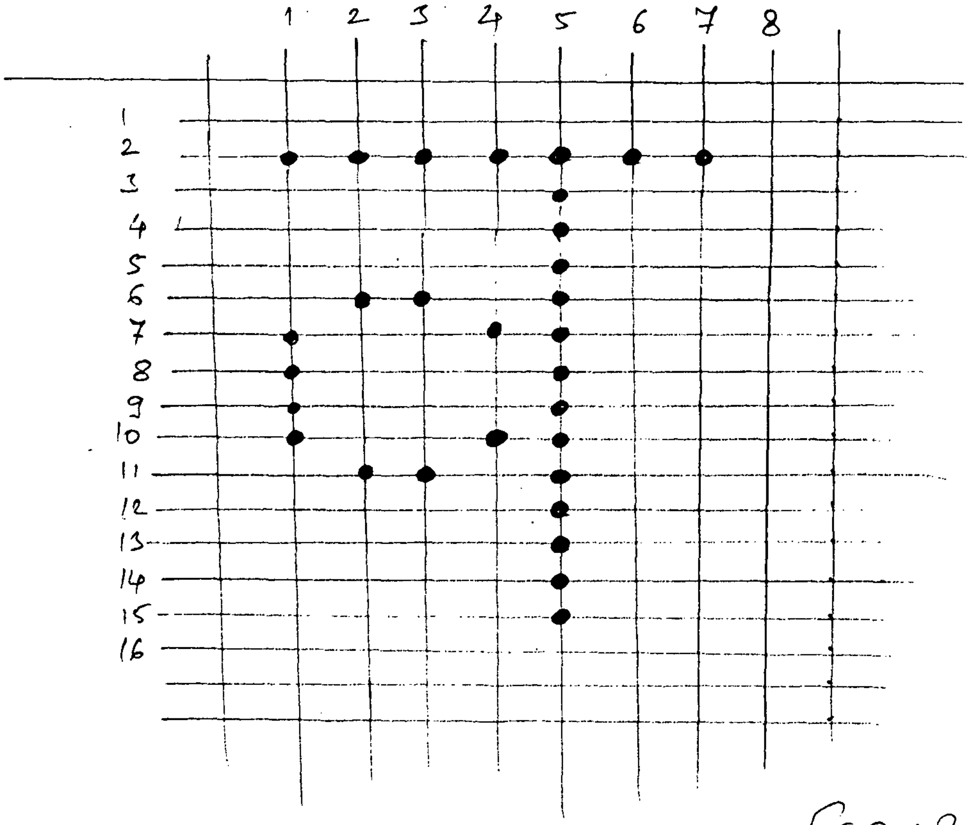
FIGURE-6.

LETTERS



क

FIGURE-7



व

FIGURE-8

LETTERS

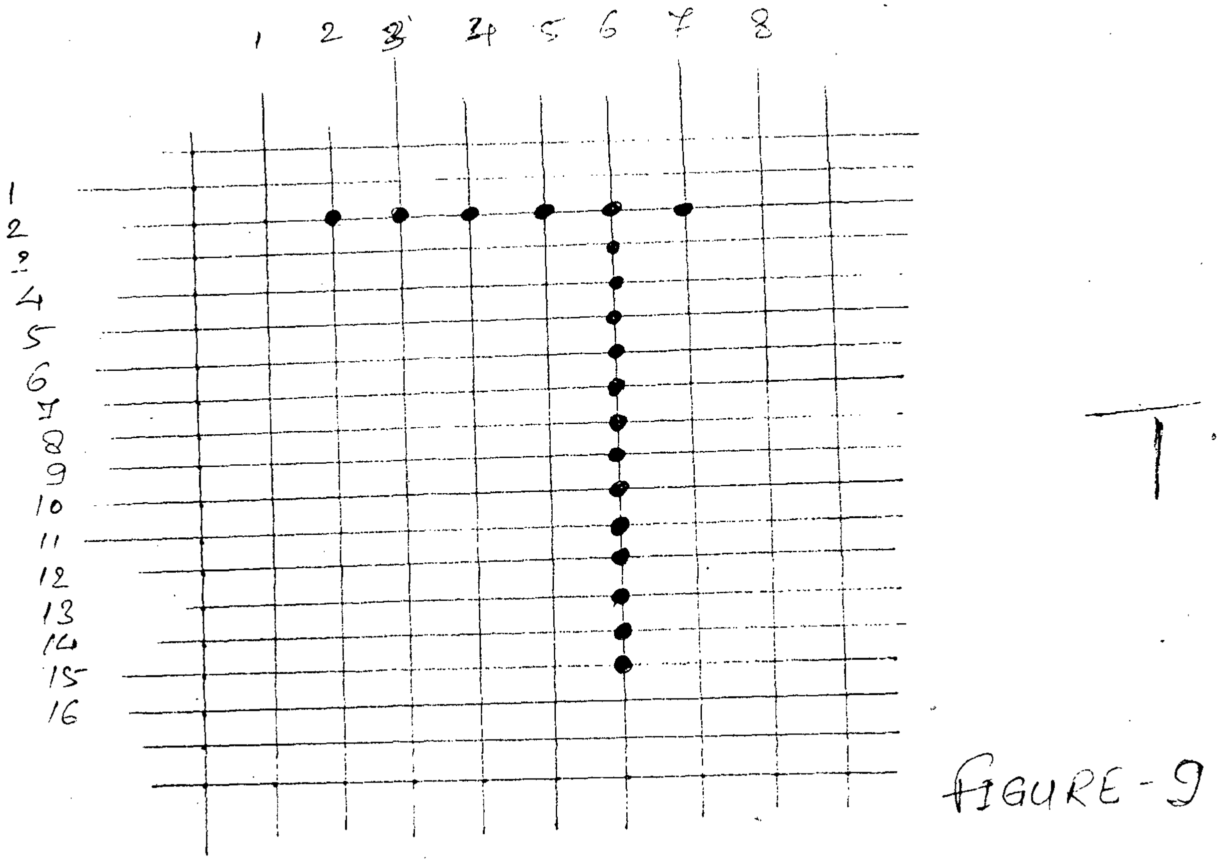


FIGURE-9

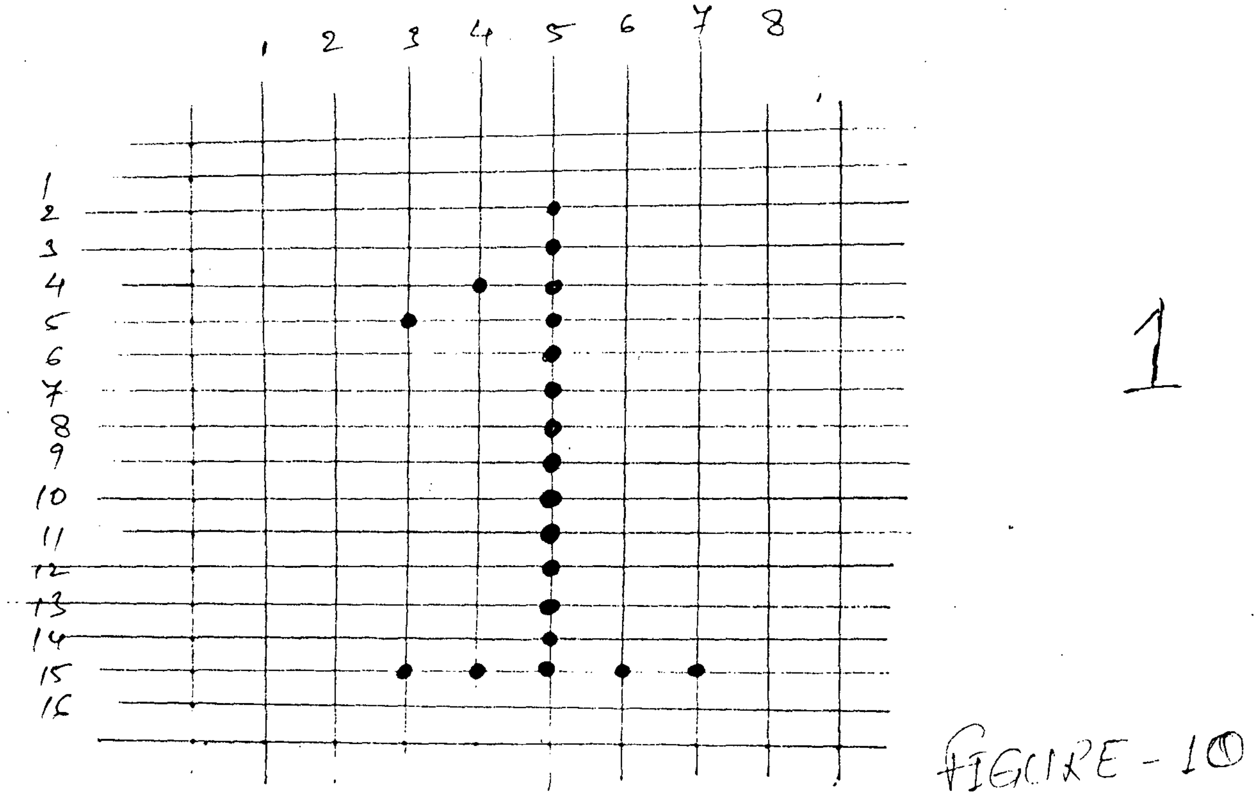


FIGURE-10

FIGURES

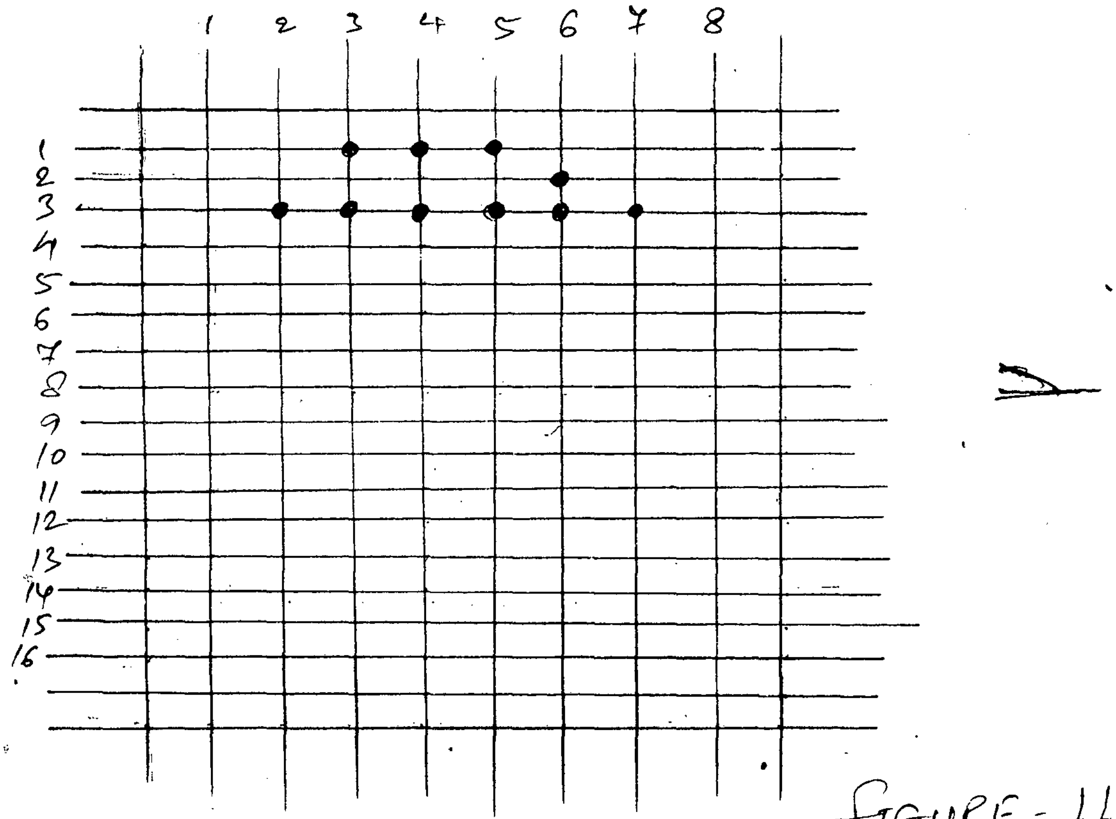


FIGURE - 11

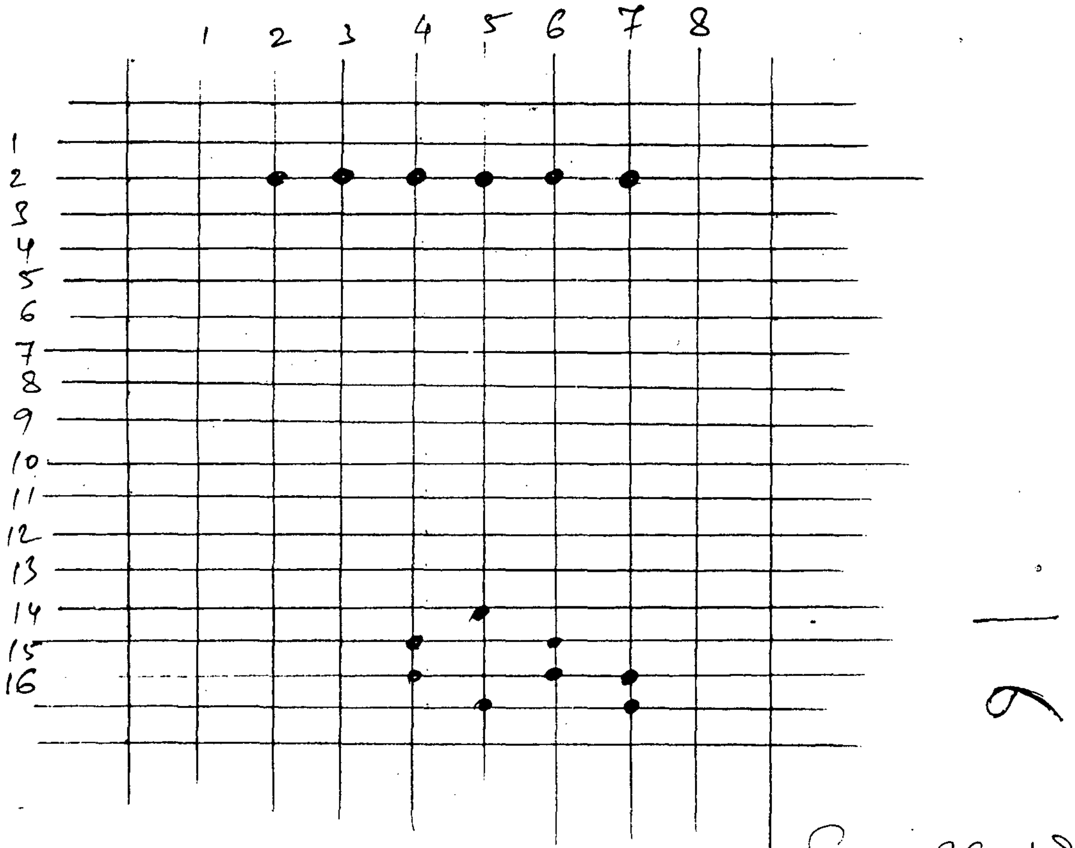


FIGURE - 12

FIGURES

4 . POSSIBLE DEVELOPMENTS

This card has been designed keeping in view the needs of the news agencies and the newspaper offices . The card can be attached to a printer and put on telex line for BHASHA and VARTA . It has been tested on BHASHA service and is working efficiently .

There are vast possibilities of developing various multilingual systemms using this . Change the character generator and you can use it for a different language .

For multilingual word - processing on computers use this . Write keyboard driver to change the function of keys and write a program to handle CRT so that the character generator of the card may be used .

Working on similar lines various methods can be developed to use this multilingual card.

APPENDIX A

8035

H - MOS SINGLE - COMPONENT 8 - BIT MICROCOMPUTER

FEATURES :

High performance H - MOS II

Interval timer / Event counter

Two single level interrupts

Single 5 - volt supply

Over 96 instructions ; 90 % single byte

Reduced poiwer consumption

Compatible with 8085 / 8080 peripherals

Easily expandable memory

Up to 1.36 usec instruction cycle

PIN DESCRIPTION :

SYMBOL	PIN NO.	FUNCTION
Vss	20	Circuit ground potential
Vdd	26	+5V during normal operation , Low power standby pin
Vcc	40	Main power supply , +5V during operation
PROG	25	Output strobe for 8243 I/O expander
P10 - P17	27-34	8 - bit quasi - bidirectional port

P20 - P23	21-24	- do -
P24 - P27	35-38	
DB0 - DB7	12-19	True bidirectional port
T0	1	Input pin testable using the conditional transfer instruction
T1	39	Input pin testable using JT1 and JNT1
INT	6	Interrupt input , initiates an interrupt when enabled
RD	8	Output strobe activated during bus read
RESET	4	Input which is used to initialize the processor
WR	10	Output strobe during a bus write
ALE	11	Address latch enable
PSEN	9	Program store enable . Occurs only during fetch to external program memory
SS	5	Single step input
EA	7	External access input
XTAL1	2	One side of crystal input for internal oscillator
XTAL2	3	Other side of crystal input

APPENDIX B

2732

32k (4k x 8) UV EPROM

FEATURES :

200ns Maximum Access Time HMOS - E Technology

Compatible with High - Speed 8MHz iAPx186 Zero wait state

Two line control

Compatible with 12MHz 8051 Family

Industry standard pinout .... JEDEC Approved

Low standby current .... 30mA maximum

±10 % Vcc tolerance available

Intelligent Identifier mode

TTL compatible

This chip has a separate output control OE . So bus contention in microprocessor systems is eliminated . The standby mode reduces power consumption without increasing access time .

APPENDIX C

6264

64k (8k x 8) INTEGRATED RAM

FEATURES :

Low cost high volume HMOS Technology

High density one transistor cell

Single 5V ±10 % supply

Proven HMOS reliability

Low active current

2764 EPROM compatible pinout

Two line bus control

JEDEC standard 28 - pin site

Low standby current (20 mA)

REFERENCES :

1. Microcontroller Handbook - INTEL
2. Memory components Handbook - INTEL
3. LOGIC Databook Volume - II - National Semiconductor Corporation

```
;************************* Header ********************************;
;               Date                    :       3Øth Sep'89
;               Last Updation    :      3Øth Sep'89
;               Program                 :       temp.asm
;               Purpose                 :       This is copy of program lmdl16.asm
;                                                       and created for temp use
;                                                       and to study the program
;               Status                  :       #all program segment which are in
;                                                        comment are removed
;*****************************************************************;

;****************** Main Memory Locations *********************;
;
;       2ØH             :
;       21H             :
;       22H             :
;       23H             :
;       24H             :
;       25H             :
;       26H             :
;
;       4ØH-6ØH  :      buffer(32 locations)
;
;       7ØH             :
;       71H             :
;       72H             :
;       73H             :
;       74H             :
;       75H             :
;       76H             :
;
;       8ØH             :
;       81H             :
;       82H             :
;       83H             :
;       84H             :
;       85H             :
;       86H             :       (86H)<-#AAH  when char is being rcvd and
;                               (86H)<-#ØØH  when STOP bit has been rcvd
;
;       87H             :
;       88H             :
;
;       AØH             :
;       A1H             :
;       A2H             :
;
;       2ØH             :
;       21H             :
;       22H             :
;
;*****************************************************************;


; COPY OF LMDL15.ASM PRINTS ALL CHRS WITH ALL LF INTRPT
;       RECOGNISION     IS REMOVED..
                ORG     ØØØØH
                JMP     MAIN                    ;main program loop
                ORG     ØØØ3H
                JMP     EXT                     ;external interrupt
                ORG     ØØØ7
                JMP     INTL                    ;timer interrupt
MAIN:
                SEL     MBØ
                SEL     RBØ
                CALL    SET_FLAG
                CALL    SET_FLAG1
                MOV     RØ,#86H
                CALL    PORT_INIT
                MOV     A,#AAH              ; SET R5 TO AAH
                MOVX    @RØ,A
                MOV     A,#8EH
                OUTL    P1,A
                EN      I
SELF:
                MOV     RØ,#81H             ; BUFFER SWITCHING PTR
                CALL    PORT_INIT
                MOVX    A,@RØ
                JZ      SELF
;PRINTING STARTS HERE..
HALF_PRN1:
                MOV     RØ,#70H
                CALL    PORT_INIT
                MOVX    A,@RØ
                MOV     R4,A
                MOV     RØ,#71H
                CALL    PORT_INIT
                MOVX    A,@RØ
                JZ      SNGL_PO
                MOV     R4,#FFH
SNGL_PO:
                MOV     R7,#1BH             ; ESC K  select single density bit image printing.
                CALL    BK2
                MOV     R7,#4BH
                CALL    BK2
                MOV     A,R4
                MOV     R7,A
                CALL    BK2
                MOV     R7,#ØØH
                CALL    BK2
                MOV     RØ,#ØØH
CARY_ON:
                MOV     A,R4
                MOV     R5,A
GBCK1:
                MOV     A,RØ
                MOV     R7,A
                MOV     RØ,#82H
                CALL    PORT_INIT
                MOVX    A,@RØ
                JZ      UBFRP1
                MOV     A,#FEH
                OUTL    P1,A
                MOV     A,#BØH
                OUTL    P2,A
                JMP     COMNP1
UBFRP1:
                MOV     R7,#24H
                CALL    BK2
                MOV     A,#FEH
                OUTL    P1,A
                MOV     A,#5ØH
                OUTL    P2,A
COMNP1:
                MOV     A,R7
                MOV     RØ,A
                MOVX    A,@RØ
                MOV     R7,A
                CALL    BK2
                INC     RØ
                DEC     R5
                MOV     A,R5
                JNZ     GBCK1
                MOV     R1,#71H
                CALL    PORT_INIT
                MOVX    A,@R1
                JNZ     SKIP2
```

```
                    MOV      [illegible]
                    MOV      A,R0
                    MOV      R7,A
                    MOV      R0,#83H
                    CALL     PORT_INIT
                    MOVX     A,@R0
                    JZ.      UBFR3
                    MOV      A,#FEH
                    OUTL     P1,A
                    MOV      A,#D0H
                    OUTL     P2,A
                    JMP      COMN3
 UBFR3:
                    MOV      A,#FEH
                    OUTL     P1,A
                    MOV      A,#70H
                    OUTL     P2,A
 COMN3:
                    MOV      A,R7
                    MOV      R0,A
                    MOV A,R2
                    MOVX     @R1,A
                    INC      R0
                    INC      R1
                    DEC      R4
                    MOV      A,R4
                    JZ       SKIP4
                    JMP      COPY6    ;JNZ     COPY6
 SKIP4:
                    MOV      A,R1
                    MOV      R7,A
                    MOV      R1,#25H
                    CALL     PORT_INIT
                    MOV      A,R7
                    MOVX     @R1,A
  DUMY3:
                    CALL     PORT_INIT
                    MOV      R1,#20H ; 20H HAS A ACTUAL ADRESS WHERE TO STORE CHR..
                    MOVX     A,@R1
                    MOV      R1,A
                    CPL      A
                    JZ       DUM4
                    JMP SEM
 DUM4:
                    MOV      R1,#21H
                    CALL     PORT_INIT
                    MOVX     A,@R1
                    MOV      R1,A
                    CPL      A
                    NOP
                    NOP
                    NOP
                    NOP
                    JNZ      SKIP6
                    JMP      THRD_PG
 SKIP6:
                    MOV      R4,#08H
 COPY2:
                    CALL     PORT_INIT
                    MOVX     A,@R0
                    MOV      R2,A
                    MOV      A,R0
                    MOV      R7,A
                    MOV      R0,#83H
                    CALL     PORT_INIT
                    MOVX     A,@R0
                    JZ       LBFR1
                    MOV      A,#FEH
                    OUTL     P1,A
                    MOV      A,#20H
                    OUTL     P2,A
                    MOV      A,R7
```

```
                 MOV       A,R1
                 JNZ       SAME3
                 MOV       A,#FFH
SAME3:
                 MOV       R7,A
                 MOV       R1,#24H
                 CALL      PORT_INIT
                 MOV       A,R7
                 MOVX      @R1,A
                 JMP       DUMY3
SEM1:
                 MOV       RØ,#4ØH
                 MOV       R4,#Ø8H
COPY4:
                 CALL PORT_INIT
                 MOVX      A,@RØ
                 MOV R2,A
                 MOV       A,RØ
                 MOV       R7,A
                 MOV       RØ,#83H
                 CALL      PORT_INIT
                 MOVX      A,@RØ
                 JZ        UBFR1
                 MOV       A,#FEH
                 OUTL      P1,A
                 MOV       A,#BØH
                 OUTL      P2,A
                 JMP       COMN1
UBFR1:
                 MOV       R7,23H
                 CALL      BK2
                 MOV       A,#FEH
                 OUTL      P1,A
                 MOV       A,#5ØH
                 OUTL      P2,A
COMN1:
                 MOV       A,R7
                 MOV       RØ,A
                 MOV A,R2
                 MOVX      @R1,A
                 MOV       R7,A
                 CALL      BK2
                 INC       RØ
                 INC       R1
                 DEC       R4
                 MOV       A,R4
                 JNZ       COPY4
                 MOV       A,R1
                 JNZ       SAME2
                 MOV       A,#FFH
SAME2:
                 MOV       R7,A
                 MOV       R1,#23H
                 CALL      PORT_INIT
                 MOV       A,R7
                 MOVX      @R1,A
                 JMP                 DUMY3
THRD_RMPG:
                 MOV       R1,#24H
                 CALL      PORT_INIT
                 MOV       A,#FFH
                 MOVX      @R1,A
                 MOV       R1,#25H
                 CALL      PORT_INIT
                 MOVX      A,@R1
                 MOV       R1,A
                 MOV       RØ,#4ØH
                 MOV       R4,#Ø8H
COPY6:
                 CALL PORT_INIT
                 MOVX      A,@RØ
```

```
                                MOV       R1,A
 ;ADJUST THE PORT LINE TO GET THE CG CHRS.
                                MOV       R4,#16
                                MOV       A,R7
                                SWAP      A
                                ANL       A,#ØFH
                                ADD       A,R5
                                MOV       R5,A
 BCK:
                                MOV       A,R5
                                OUTL      P2,A
                                MOVX      A,@R1
                                MOV       R7,A
                                CALL      PORT_INIT
                                MOV       A,R7
                                MOVX      @RØ,A
                                INC       RØ
                                INC       R1
                                DEC       R4
                                MOV       A,R4
                                JNZ       BCK
                                DEC       RØ
                                RET
  HERE:
                                MOV       RØ,#23H
                                CALL      PORT_INIT
                                MOVX      A,@RØ
                                MOV       R1,A
                                CPL       A
                                JZ        SKIP11
                                JMP       SEM1
 SKIP11:
                                MOV       R1,#24H
                                CALL      PORT_INIT
                                MOVX      A,@R1
                                MOV       R1,A
                                CPL       A
                                NOP
                                JNZ       SKIP12
                                JMP       THRD_RMPG
 SKIP12:
                                MOV       RØ,#4ØH
                                MOV       R4,#Ø8H
 COPY5:
                                CALL      PORT_INIT
                                MOVX      A,@RØ
                                MOV       R2,A
                                MOV       A,RØ
                                MOV       R7,A
                                MOV       RØ,#83H
                                CALL      PORT_INIT
                                MOVX      A,@RØ
                                JZ        UBFR2
                                MOV       A,#FEH
                                OUTL      P1,A
                                MOV       A,#CØH
                                OUTL      P2,A
                                JMP       COMN2
 UBFR2:
                                MOV       A,#FEH
                                OUTL      P1,A
                                MOV       A,#6ØH
                                OUTL      P2,A
 COMN2:
                                MOV       A,R7
                                MOV       RØ,A
                                MOV       A,R2
                                MOVX      @R1,A
                                INC       RØ
                                INC       R1
                                DEC       R4
                                MOV       A,R4
```

```
                CALL      GET_FL
                MOV       R0,#A1H
                CALL      PORT_INIT
                MOVX      A,@R0
                CPL       A
                ADD       A,#01H
                MOV       R7,A
                ADD       A,#01H   ; TO BE CHANGE THIS LIKE ADD A,#02
                NOP
                NOP
                JNZ       ORING    ;                                       JZ ORING
                JMP       PRINT_PRV        ;DATA TO BE TAKEN FROM QG AFTER PRINTING.
ORING:
                MOV       A,R7     ;IT IS TO BE INITIALISED TO 00 FROM 40HTO 50H
                ADD       A,#02H
                JZ        SCND_CHR
                JMP       PRINT_PRV
SCND_CHR:
                MOV       A,R5
                MOV       R7,A
                CALL      GET_FL
                MOV       R0,#50H
                CALL      FIRST
                CALL      ORDATA
                JMP       RDY1
PRINT_PRV:
                CALL      HERE
                MOV       R0,#80H
                CALL      PORT_INIT
                MOVX      A,@R0
                MOV       R7,A
                CALL      GET_FL
                MOV       R0,#40H
                CALL      FIRST
                MOV       R0,#A2H
                CALL      PORT_INIT
                MOVX      A,@R0
                MOV       R2,A
DUM9:
                INC       R2
                MOV       R0,#A2H
                CALL      PORT_INIT
                MOV       A,R2
                MOVX      @R0,A
                JMP       RDY1
RDY:
                MOV       R0,#81H                     ; TO PRINT
                CALL      PORT_INIT
                MOV       A,#FFH
                MOVX      @R0,A
                INC       R0
                INC       R0
                CALL      PORT_INIT
                MOVX      A,@R0
                MOV       R1,A
                CALL      PORT_INIT
                MOV       A,R1
                CPL       A
                MOVX      @R0,A
RDY1:
                MOV       R0,#84H
                CALL      PORT_INIT
                MOV       A,#AAH
                MOVX      @R0,A
                CALL      POP
                EN I
                SEL       RB0
                RETR
FIRST:
                MOV       A,R7
                ANL       A,#0FH
                SWAP      A
```

```
DUMY1:
                MOV     R7,#ØAH
LSFS:
                MOV     RØ,#AØH          ; STORES THE LS OR FS CODE INDICATION.
                CALL    PORT_INIT
                MOV     A,R7
                MOVX    @RØ,A
                JMP     RDY1
SWP:
                MOV     A,R7
                CPL     A
                ADD     A,#Ø1H
                ADD     A,#Ø2H
                JNZ     DUMY2
                CALL    HERE
                CALL    GET_VALUE
                JMP     RDY
DUMY2:
                MOV     A,R7
                MOV     R5,A
                MOV     RØ,#A1H
                CALL    PORT_INIT
                MOVX    A,@RØ
                CPL     A
                ADD     A,#Ø1H
                JZ      SKIP3
                JMP     OTHER    ;JNZ    OTHER
SKIP3:
                CALL    COMP_R
                MOV     RØ,#A1H
                CALL    PORT_INIT
                MOVX    A,@RØ
                CPL     A
                ADD     A,#Ø1H
                MOV     R7,A
                ADD     A,#Ø1H
                JNZ     SECOND
                MOV     A,R5
                MOV     R7,A
                CALL    GET_FL
                MOV     RØ,#4ØH
                CALL    FIRST    ;A2 CAN BE INCREMENTED HERE..
                MOV     RØ,#A2H
                CALL    PORT_INIT
                MOVX    A,@RØ
                MOV     R2,A
                INC     R2
                CALL    PORT_INIT
                MOV     A,R2
                MOVX    @RØ,A
                JMP     RDY1
SECOND:
                MOV     A,R7
                ADD     A,#Ø2H
                JZ      SKIP25
                JMP     THIRD
SKIP25:
                MOV     A,R5
                MOV     R7,A
                CALL    GET_FL
                MOV     RØ,#5ØH
                CALL    FIRST
                CALL    ORDATA
                JMP     RDY1
THIRD:
                MOV     A,R5
                MOV     R7,A
                CALL    GET_FL
                MOV     RØ,#4ØH
                CALL    FIRST
                JMP     RDY1
OTHER:
```

```
                JNZ        SET_CY
                MOV        R2,#00H
                MOV        A,R7
                RL         A
                ADD        A,R2
                MOV        R7,A
                JMP        CONTNU
SET_CY:
                MOV        R2,#01H
                MOV        A,R7
                RL         A
                ADD        A,R2
                MOV        R7,A
CONTNU:
                MOV        A,#FEH
                OUTL       P1,A
                MOV        A,#90
                MOV        T,A
                EN         TCNTI
                STRT       T
                DEC        R6
                MOV        A,R6
                JZ         STPBT
                CALL       POP
                SEL        RB0
                RETR
CONTNU1:
STPBT:
                DIS        TCNTI
                STOP       TCNT
                MOV        R0,#86H
                CALL       PORT_INIT
                MOV        A,#00H
                MOVX       @R0,A
                MOV        A,#80H   ;GIVES DELAY AFTER LAST DATA BIT TO DETECT STRTBT
                MOV        T,A
                EN         TCNTI
                STRT       T
                CALL POP
                SEL        RB0
                RETR


;*** For (86H)<-#AAH & (80H)<-R7  R7 contains rcvd char
BACK:
                DIS        TCNTI
                STOP       TCNT
                MOV        R0,#86H
                CALL       PORT_INIT
                MOV        A,#AAH
                MOVX       @R0,A
                MOV        R0,#80H
                CALL       PORT_INIT
                MOV        A,R7
                MOVX       @R0,A

BK3:     MOV    A,R7
                CPL        A
                ADD        A,#01H
; CHECK CR AND LF AT THIS POINT.......
                ADD        A,#0EH            ; check LS
                JNZ        FS
                MOV        R7,#08H
                JMP        LSFS
FS:
                MOV        A,R7
                CPL        A
                ADD        A,#01H
                ADD        A,#1BH
                NOP
                JZ         DUMY1
                JMP        SWP                      ;JNZ     SWP
```

```
;     (88H) <- Acc & (87H) <- PSW      when enter in routine .   ;
;        Acc <- (88H) & PSW <- (87H)      when exit from routine    ;
;*****************************************************************;


EXT:     SEL RB1
                 DIS       I
                 DIS       TCNTI
                 CALL      PUSH                    ; saves (88H)<-A, (87H)<-PSW
                 STOP      TCNT
                 MOV       R6,#07H         ; R6=no of bits in a char
                 MOV       R7,#ØØH         ; R7=char being rcvd #initialized
                 MOV       A,#FEH
                 OUTL      P1,A
                 MOV       A,#Ø5           ; after ext int delay for sampling next bit
                 MOV       T,A
                 EN        TCNTI
                 STRT      T
                 MOV       RØ,#8AH         ; TO CHECK EN I FROM EXT..
                 CALL      PORT_INIT
                 MOV       A,#FFH
                 MOVX      @RØ,A
                 CALL      POP                     ; restores A<-(88H), PSW<-(87H)
                 SEL       RBØ
                 RETR

PUSH:
                 MOV       R4,A
                 MOV       A,PSW
                 MOV       R2,A
                 MOV RØ,#87H
                 CALL PORT_INIT
                 MOV A,R2
                 MOVX @RØ,A
                 INC       RØ
                 CALL      PORT_INIT
                 MOV       A,R4
                 MOVX      @RØ,A
                 RET
POP:
                 MOV RØ,#87H
                 CALL PORT_INIT
                 MOVX A,@RØ
                 MOV       R2,A
                 INC       RØ
                 CALL      PORT_INIT
                 MOVX      A,@RØ
                 MOV R4,A
                 MOV       A,R2
                 MOV       PSW,A
                 MOV A,R4
                 RET
;******************* Timer Interrupt Routine ********************;
;
;*****************************************************************;


INTL:
                 SEL       RB1
                 DIS       TCNTI
                 STOP      TCNT
                 CALL PUSH
SKIP10: MOV      RØ,#86H
                 CALL      PORT_INIT
                 MOVX      A,@RØ
                 JNZ       SKIP9           ;if (86H)=#AAH then rcving char
                                                   ;otherwise rcvd first bit of char

                 JMP       BACK

SKIP9:
                 IN        A,P1
                 ANL       A,#Ø8H          ; Ø8 BAUDAT LEFT SHIFT IS NOT NEEDED.
```

```
                CALL    PORT_INIT
                MOVX    A,@RØ
                JNZ     SKIP27
                EN      I
                CALL    PORT_INIT
                MOV     A,#ØØH
                MOVX    @RØ,A
SKIP27:
                JMP     SELF
SET_FLAG:
                MOV     R3,#Ø6H
                MOV     RØ,#2ØH         ; HERE LOCS..2ØH..26H
INIT3:
                CALL    PORT_INIT
                MOV     A,#ØØH
                MOVX    @RØ,A
                INC     RØ
                DEC     R3
                MOV     A,R3
                JNZ     INIT3
                MOV     RØ,#7ØH         ; GET_VALUE LOCS....7ØH..74H
                MOV     R3,#Ø5H
INIT4:
                CALL    PORT_INIT
                MOV     A,#ØØH
                MOVX    @RØ,A
                INC     RØ
                DEC     R3
                MOV     A,R3
                JNZ     INIT4
                MOV     RØ,#AØH ; AØ..LSFS,A1..CHR TYPE,A2..NO.OF CHR
                MOV     R3,#Ø4H
INIT2:
                CALL    PORT_INIT
                MOV     A,#ØØH
                MOVX    @RØ,A
                INC     RØ
                DEC     R3
                MOV     A,R3
                JNZ     INIT2
                MOV     RØ,#4ØH         ;BUFFER
                MOV     R3,#32
INIT1:
                CALL    PORT_INIT
                MOV     A,#ØØH
                MOVX    @RØ,A
                INC     RØ
                DEC     R3
                MOV     A,R3
                JNZ     INIT1                   ; 81-ØØ ..NOT TO PRINT..
                RET
SET_FLAG1:                                      ;81 KEEPS THE TRACK OF PRINTING OF EITHER EOFF
                MOV RØ,#81H                     ; 82-TO SELECT PRINTING. 83- TO
                MOV R3,#Ø5H
;       SELECT WRITING
INIT9:
                CALL PORT_INIT
                MOV A,#ØØH
                MOVX @RØ,A
                INC RØ
                DEC R3
                MOV A,R3
                JNZ INIT9
                RET


;***************** External Interrupt Routine *****************;
;       This routine initializes:
;           R6 <- #Ø7h    (no of bits in a char)
;           R7 <- #ØØH    (init. R7 contains char being rcvd )
;           T  <- #Ø5h    (dealy for sampling first bit of char)
;
.
```

```
                CALL      PORT_INIT
                MOVX      A,@RØ
                JZ        LBFRP3
                MOV       A,#FEH
                OUTL      P1,A
                MOV       A,#AØH
                OUTL      P2,A
                MOV       A,R7
                MOV       RØ,A
                JMP       COMNLP3
LBFRP3:

                MOV       A,#FEH
                OUTL      P1,A
                MOV       A,#4ØH
                OUTL      P2,A
COMNLP3:
                MOVX      A,@RØ
                MOV       R7,A
                CALL      BK2
                INC       RØ
                DEC       R5
                MOV       A,R5
                JNZ       GBCK6
; GIVES THE NORMAL CR LF..
CRLF:
                MOV       RØ,#7ØH
                MOV       R3,#Ø3H

INIT5:
                CALL      PORT_INIT
                MOV       A,#ØØH
                MOVX      @RØ,A
                INC       RØ
                DEC       R3
                MOV       A,R3
                JNZ       INIT5
                MOV       R7,#ØDH
                CALL      BK2
                MOV       R7,#1BH
                CALL      BK2
                MOV       R7,#33H
                CALL      BK2
                MOV       R7,#24H
                CALL      BK2
                MOV       R7,#ØAH
                CALL      BK2
                MOV       RØ,#AØH
                CALL      PORT_INIT
                MOVX      A,@RØ
                MOV       R7,A
                CALL      SET_FLAG
                MOV       RØ,#AØH
                CALL      PORT_INIT
                MOV       A,R7
                MOVX      @RØ,A
                MOV       RØ,#86H
                CALL      PORT_INIT
                MOV       A,#AAH
                MOVX      @RØ,A
                MOV       RØ,#81H
                CALL      PORT_INIT
                MOV       A,#ØØH
                MOVX      @RØ,A
                INC       RØ
                CALL      PORT_INIT
                MOVX      A,@RØ
                MOV       R1,A
                CALL      PORT_INIT
                MOV       A,R1
                CPL       A
                MOVX      @RØ,A
                MOV       RØ,#8AH
```

```
            MOV     R4,A
            JNZ     SKIP5
            JMP     CRLF
SKIP5:
            MOV     R1,#72H
            CALL    PORT_INIT
            MOVX    A,@R1
            JZ      SKPP      ;CRLF    ; TO BE CHANCED....
            MOV     R4,#FFH
SKPP:
            MOV     R7,#1BH
            CALL    BK2
            MOV     R7,#4BH
            CALL    BK2
            MOV     A,R4
            MOV     R7,A
            CALL    BK2
            MOV     R7,#ØØH
            CALL    BK2
            MOV     RØ,#ØØH
            MOV     RØ,#ØØH
            MOV     A,R4
            MOV     R5,A
GBCK4:
            MOV     A,RØ
            MOV     R7,A
            MOV     RØ,#82H
            CALL    PORT_INIT
            MOVX    A,@RØ
            JZ      LBFRP2
            MOV     A,#FEH
            OUTL    P1,A
            MOV     A,#7ØH
            OUTL    P2,A
            MOV     A,R7
            MOV     RØ,A
            JMP     COMNLP2
LBFRP2:
            MOV     A,#FEH
            OUTL    P1,A
            MOV     A,#3ØH
            OUTL    P2,A
COMNLP2:
            MOVX    A,@RØ
            MOV     R7,A
            CALL    BK2
            INC     RØ
            DEC     R5
            MOV     A,R5
            JNZ     GBCK4
            MOV     R1,#72H
            CALL    PORT_INIT
            MOVX    A,@R1
            JZ      CRLF
            MOV     R4,A
            MOV     R7,#1BH
            CALL    BK2
            MOV     R7,#4BH
            CALL    BK2
            MOV     A,R4
            MOV     R7,A
            CALL    BK2
            MOV     R7,#ØØH
            CALL    BK2
            MOV     RØ,#ØØH
            MOV     RØ,#ØØH
            MOV     A,R4
            MOV     R5,A
GBCK6:
            MOV     A,RØ
            MOV     R7,A
            MOV     RØ,#[illegible]H
            OUTL    P1,A
            MOV     A,#70H
            OUTL    P2,A
COMNP3:
            MOV     A,R7
            MOV     RØ,A
            MOVX    A,@RØ
            MOV     R7,A
            CALL    BK2
            INC     RØ
            DEC     R5
            MOV     A,R5
            JNZ     GBCK5
CARY_ON2:
            MOV     R1,#71H
            CALL    PORT_INIT
            MOVX    A,@R1
            JZ      CARY_ON3
            MOV     R4,#FFH
CARY_ON3:
            MOV     R7,#ØDH
            CALL    BK2
            MOV     R7,#1BH       ;ESC 3      set n/216 inch line spacing   n 0-255
            CALL    BK2
            MOV     R7,#33H
            CALL    BK2
            MOV     R7,#17H
            CALL    BK2
            MOV     R7,#ØAH
            CALL    BK2
            MOV     R7,#1BH       ;ESC K
            CALL    BK2
            MOV     R7,#4BH
            CALL    BK2
            MOV     A,R4
            MOV     R7,A
            CALL    BK2
            MOV     R7,#ØØH
            CALL    BK2
            MOV     RØ,#ØØH
            MOV     A,R4
            MOV     R5,A
GBCK2:
            MOV     A,RØ
            MOV     R7,A
            MOV     RØ,#82H
            CALL    PORT_INIT
            MOVX    A,@RØ
            JZ      LBFRP1
            MOV     A,#FEH
            OUTL    P1,A
            MOV     A,#80H
            OUTL    P2,A
            MOV     A,R7
            MOV     RØ,A
            JMP     COMNLP1
LBFRP1:
            MOV     A,#FEH
            OUTL    P1,A
            MOV     A,#2ØH
            OUTL    P2,A
COMNLP1:
            MOVX    A,@RØ
            MOV     R7,A
            CALL    BK2
            INC     RØ
            DEC     R5
            MOV     A,R5
            JNZ     GBCK2
            MOV     R1,#71H
            CALL    PORT_INIT
            MOVX    A,@R1
```

```
                JMP     CARY_ON2            ;JZ     CARY_ON2
SKIP2:
                MOV     R4,A
                MOV     R5,A
                MOV     R7,#1BH
                CALL    BK2
                MOV     R7,#4BH
                CALL    BK2
                MOV     A,R4
                MOV     R7,A
                CALL    BK2
                MOV     R7,#00H
                CALL    BK2
                MOV     R0,#00H
; TO BE ADDED FROM HERE...........//////.
GBCK3:
                MOV     A,R0
                MOV     R7,A
                MOV     R0,#82H
                CALL    PORT_INIT
                MOVX    A,@R0
                JZ      UBFRP2
                MOV     A,#FEH
                OUTL    P1,A
                MOV     A,#C0H
                OUTL    P2,A
                JMP     COMNP2
UBFRP2:
                MOV     A,#FEH
                OUTL    P1,A
                MOV     A,#60H
                OUTL    P2,A
COMNP2:
                MOV     A,R7
                MOV     R0,A
                MOVX    A,@R0
                MOV     R7,A
                CALL    BK2
                INC     R0
                DEC     R5
                MOV     A,R5
                JNZ     GBCK3
                MOV     R1,#72H
                CALL    PORT_INIT
                MOVX    A,@R1
                JZ      CARY_ON2
                MOV     R4,A      ;;;;
                MOV     R5,A
                MOV     R7,#1BH
                CALL    BK2
                MOV     R7,#4BH
                CALL    BK2
                MOV     A,R4
                MOV     R7,A
                CALL    BK2
                MOV     R7,#00H
                CALL    BK2
                MOV     R0,#00H
GBCK5:
                MOV     A,R0
                MOV     R7,A
                MOV     R0,#82H
                CALL    PORT_INIT
                MOVX    A,@R0
                JZ      UBFRP3
                MOV     A,#FEH
                OUTL    P1,A
                MOV     A,#D0H
                OUTL    P2,A
                JMP     COMNP3
UBFRP3:
                MOV     A,#FEH
```

```
               MOVX      @R1,A
               RET
 THRD_FG:
               MOV       R1,#21H
               CALL      PORT_INIT
               MOV       A,#FFH
               MOVX      @R1,A
               MOV       R1,#22H
               CALL      PORT_INIT
               MOVX      A,@R1
               MOV       R1,A
               MOV       R4,#Ø8H
 COPY3:
               CALL PORT_INIT
               MOVX      A,@RØ
               MOV R2,A
               MOV       A,RØ
               MOV       R7,A
               MOV       RØ,#83H
               CALL      PORT_INIT
               MOVX      A,@RØ
               JZ        LBFR3
               MOV       A,#FEH
               OUTL      P1,A
               MOV       A,#AØH
               OUTL      P2,A
               MOV       A,R7
               MOV       RØ,A
               JMP       COMNL3
 LBFR3:
               MOV       A,#FEH
               OUTL      P1,A
               MOV       A,#4ØH
               OUTL      P2,A
 COMNL3:
               MOV A,R2
               MOVX      @R1,A
               INC       RØ
               INC       R1
               DEC       R4
               MOV       A,R4
               JZ        SKIP1
               JMP       COPY3
 SKIP1:
               MOV       A,R1
               MOV       R7,A
               MOV       R1,#22H
               CALL      PORT_INIT
               MOV       A,R7
               MOVX      @R1,A
               RET
 GET_VALUE:
               MOV       RØ,#A2H
               CALL      PORT_INIT
               MOVX      A,@RØ
               MOV       R2,A
               CPL       A
               ADD       A,#Ø1H
               ADD       A,#32
               ANL A,#8ØH
               JNZ       SKIP26
               JMP       AD N2
 SKIP26:
               MOV       RØ,#7ØH
               CALL      PORT_INIT
               MOV       A,#FFH
               MOVX      @RØ,A
               MOV       A,#32
               CPL       A
               ADD       A,#Ø1H
               ADD       A,R2
               MOV       R2,A
```

```
                        CPL         A
                        ADD         A,#Ø1H
                        ADD         A,#32
                        JZ          ADIN4
                        ANL         A,#8ØH
                        JZ          ADIN4
                        MOV.        RØ,#71H
                        CALL        PORT_INIT
                        MOV         A,#FFH
                        MOVX        @RØ,A
                        MOV         A,#32
                        CPL         A
                        ADD         A,#Ø1H
                        ADD         A,R2
                        MOV         R2,A
                        MOV         R4,A
                        MOV         R3,#Ø7H
 ADIN5:
                        MOV         A,R2
                        ADD         A,R4
                        MOV         R4,A
                        DEC         R3
                        MOV         A,R3
                        JNZ         ADIN5
                        MOV         RØ,#72H
                        CALL        PORT_INIT
                        MOV         A,R4
                        JNZ PRPER2
                        MOV A,#FFH
 PRPER2:
                        MOVX        @RØ,A
                        RET
 ADIN4:
                        MOV         A,R2
                        MOV         R4,A
                        MOV         R3,#Ø7H
 ; CALCULATES THE REMAINING BYTES REQRD..
 ADTN3:
                        MOV         A,R2
                        ADD         A,R4
                        MOV         R4,A
                        DEC         R3
                        MOV         A,R3
                        JNZ         ADTN3
                        MOV         RØ,#71H
                        CALL        PORT_INIT
                        MOV         A,R4
                        NOP
                        JNZ PRPER
                        MOV A,#FFH
 PRPER:     MOVX        @RØ,A
                        RET
 ADTN2:
                        MOV         A,R2
                        MOV         R4,A
                        MOV         R3,#Ø7H
 ADTN1:
                        MOV         A,R2
                        ADD         A,R4
                        MOV         R4,A
                        DEC         R3
                        MOV         A,R3
                        JNZ         ADTN1
                        MOV         RØ,#7ØH
                        CALL        PORT_INIT
                        MOV         A,R4
                        NOP
                        JNZ PRPER1
                        MOV A,#FFH
 PRPER1:    MOVX        @RØ,A
                        RET
 COMP_R:
```

```
                MOV     RØ,#AØH
                CALL    PORT_INIT
                MOVX    A,@RØ
                MOV     R7,A
                CPL     A
                ADD     A,#Ø1H
                ADD     A,#Ø8H
                NOP
                NOP
                NOP
                NOP
                JZ      LETTER
                JMP     FIGURE
LETTER:
                MOV     RØ,#8ØH
                CALL    PORT_INIT
                MOVX    A,@RØ
                CPL     A
                ADD     A,#Ø1H
                MOV     R7,A
LOOP:
                MOV     R3,#ØØH
                ADD     A,#ØØH
                JNZ     ONE
                JMP     CHNO1
ONE:
                MOV     A,R3
                ADD     A,#Ø5
                MOV     R3,A
                MOV     A,R7
                ADD     A,R3
                NOP
                JNZ     TWO
                JMP     CHNO1
TWO:
                INC     R3
                MOV     A,R7
                ADD     A,R3
                JNZ     THREE
                JMP     CHNO1
THREE:
                INC     R3
                MOV     A,R7
                ADD     A,R3
                NOP
                NOP
                NOP
                JNZ     FOUR
                JMP     CHNO1
FOUR:
                MOV     A,R3
                ADD     A,#Ø2H
                MOV     R3,A
LOOP1:
                MOV     A,R7
                ADD     A,R3
                JNZ     FIVE
                JMP     CHNO1
FIVE:
                INC     R3
                CALL    CHK_CHR
                ADD     A,#ØEH
                JZ      TEN
                JMP             LOOP1
TEN:
                INC     R3
LOOP2:
                MOV     A,R7
                ADD     A,R3
                JNZ     SEVEN
                JMP     CHNO1
SEVEN:
```

```
                CALL     CHK_CHR
                ADD      A,#14H
                JNZ      LOOP2
                INC      R3
LOOP3:
                MOV      A,R7
                ADD      A,R3
                JNZ      EIGHT
                JMP      CHNO1
EIGHT:
                INC      R3
                CALL     CHK_CHR
                ADD      A,#1BH
                JNZ      LOOP3
                INC      R3
                INC      R3
                MOV      A,R7
                ADD      A,R3
                JNZ      NINE
                JMP      CHNO1
NINE:
                INC      R3
                MOV      A,R7
                ADD      A,R3
                JZ       CHNO1
                JMP LOOP4
CHNO1:
                MOV      RØ,#A1H
                CALL     PORT_INIT
                MOV      A,#Ø1H
                MOVX     @RØ,A
                RET      ;JMP
LOOP4:
                MOV      R3,#Ø1H
LOOP5:
                MOV      A,R7
                    ADD      A,R3
                    JZ       CHNO2
                    MOV      R3,#Ø3H
                    MOV      A,R7
                    ADD      A,R3
                    JZ       CHNO2
                    MOV      R3,#Ø8H
                    MOV      A,R7
                    ADD      A,R3
                    JZ       CHNO2
                    MOV      R3,#14H
                    MOV      A,R7
                    ADD      A,R3
                    JZ       CHNO2
                    MOV      R3,#1CH
                    MOV      A,R7
                    ADD      A,R3
                    JZ       CHNO2
                    MOV      R3,#1FH
                    MOV      A,R7
                    ADD      A,R3
                    JZ       CHNO2
                    JMP      CHNO3
    CHNO2:
                    MOV      RØ,#A1H
                    CALL     PORT_INIT
                    MOV      A,#Ø2H
                    MOVX     @RØ,A
                    RET                 ;JMP
    CHNO3:
                    MOV      RØ,#A1H
                    CALL     PORT_INIT
                    MOV      A,#Ø3H
                    MOVX     @RØ,A
                    RET                 ;JMP
                MOV      RØ,#8ØH
                CALL     PORT_INIT
                MOVX     A,@RØ
                CPL      A
                ADD      A,#Ø1H
                MOV      R7,A
                MOV      R3,#Ø5H
                ADD      A,R3
                JNZ      DUMY5
                JMP      CHNO1
DUMY5:
                MOV      R3,#Ø9H
                MOV      A,R7
                ADD      A,R3
                JNZ      DUMY4
                JMP      CHNO1
DUMY4:
                MOV      R3,#ØCH
                MOV      A,R7
                ADD      A,R3
                JNZ      DUM11
                JMP      CHNO1
DUM11:
                MOV      R3,#11H
                MOV      A,R7
                ADD      A,R3
                NOP
                NOP
                NOP
                NOP
                NOP
                JNZ      DUM8
                JMP      CHNO1
DUM8:
                MOV      R3,#13H
                MOV      A,R7
                ADD      A,R3
                JNZ      DUM1Ø
                JMP      CHNO1
DUM1Ø:
                MOV      R3,#17H
                MOV      A,R7
                ADD      A,R3
                JNZ DUM6
                JMP CHNO1
DUM6:           MOV      R3,#19H
                MOV      A,R7
                ADD      A,R3
                NOP
                NOP
                NOP
                JNZ  DUM5
                JMP      CHNO1
DUM5:           INC      R3
                MOV      A,R7
                ADD      A,R3
                NOP
                NOP
                NOP
                JNZ      DUM1
                JMP  CHNO1
DUM1:     INC   R3
                INC      R3
                INC      R3
                MOV      A,R7
                ADD      A,R3
                NOP
                NOP
                NOP
                JZ       JM_OVER
                JMP      DUM2      ;JNZ DU  2
                JMP       CHNO1
DUM2:    INC    R3
                MOV       A,R7
                ADD       A,R3
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                JZ        JMP_CH
                JMP  DUM3
JMP_CH:
                JMP  CHNO1
DUM3:    JMP    LOOP4
CHK_CHR:
                MOV       A,R3
                CPL       A
                ADD       A,#Ø1H
                RET
ORDATA:
                MOV       R4,#16
                MOV       RØ,#4ØH
                MOV       R1,#5ØH
PRIT:
                CALL      PORT_INIT
                MOVX      A,@RØ
                MOV       R7,A
                MOVX      A,@R1
                ORL       A,R7
                MOVX      @RØ,A
                INC       RØ
                INC       R1
                DEC       R4
                MOV       A,R4
                JZ        SKIP7
                JMP       PRIT      ;JNZ     PRIT
SKIP7:
                RET
GET_FL:
                MOV       RØ,#AØH
                CALL      PORT_INIT
                MOVX      A,@RØ
                MOV       R5,A
                RET
BK2:
                IN        A,P1
                JB2       OKY
                JMP       BK2
OKY:
                CALL      PORT_INIT
                MOV       A,#ØAH
                OUTL      P1,A
                MOV       A,R7
                OUTL      P2,A
                MOV       A,#ØBH
                OUTL      P1,A
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
```

```
            NOP
            NOP
            NOF
            MOV       A,#FEH
            OUTL      F1,A
            RET
FORT_INIT:
            MO        ABJEFE  F1,

            OQV       ,,#ØH
            UUT¶≈P2,
            R‖
            NND┐;;EDDS HE E--------
```