PORTABLE COMPUTERS FOR REAL-TIME SIGNAL PROCESSING: EEG ANALYSIS AS A CASE STUDY

(VOLUME II)

R.A.COMLEY

Thesis presented for the degree of
Doctor of Philosophy
of the City University, London

1978

INTRODUCTION

The work recorded in Volume I represents the initial phase of a proposed three stage programme concerned with the development of the Roving Slave Processor (RSP) concept. This being the case, much of the work involved during the first phase will be required for the second and third phases and so may need occasional maintenance and, more importantly, may require future additions and modifications. As a result, it was considered that the development work should be recorded in sufficient detail so as to facilitate any such future work and, further, that the records should not only be made available in the form of user documents but should also be placed together in a more permanent binding in order to minimize the possibilities of loss or damage. The result is this second volume, in which the technical details of both the hardware and software, developed for the RSP, are recorded.

Since each of the chapters is intended to form the basis of individual documents, repetition of some points has been unavoidable, although it has been kept to a minimum.

CONTENTS

Chapter 1

Gipop Link

1. Technical Description

2. User Guide

1.1 Technical Description

1.1.1 Introduction

The Gipop (general input-output) link is basically a sixteen-bit bi-directional driver-receiver unit which is attached to the Gipop highway of the FM1600B computer (Fig. 1.1). The Gipop unit is a peripheral of the FM1600B, and is well documented elsewhere. The link and associated programs are designed to make the Gipop unit easy to use.

The Gipop link is intended primarily for use as an interface between the FM1600B and the F100 microcomputer system, but may be easily adapted for general use by the provision of a suitable control program, and some minor changes to the subroutines described in this document.

In addition to the sixteen-bit bi-directional data highway, three latched outputs are supplied for control purposes and a handshake pair included to provide a comprehensive interface facility.

The circuit has been designed in such a way that the control word may be latched from the Gipop highway before the first request is issued via the handshake lines. In this manner the control word is stable at the interface before any data transfers are initiated (see timing diagrams).

1.1.2 Design Considerations

The Gipop highway from the FM1600B comprises two sixteen-bit parallel data highways (data in and data out), and additional inputs and outputs are therefore required to provide the handshake and control word latching functions.

A relay output (Relay 1) is used to perform the latching functions and a second relay (Relay 2) and the External Program Interrupt line form the handshake pair. An output from the 'S' statisiser of the FM1600B is also taken out to the link where it is used to reset the link hardware prior to the

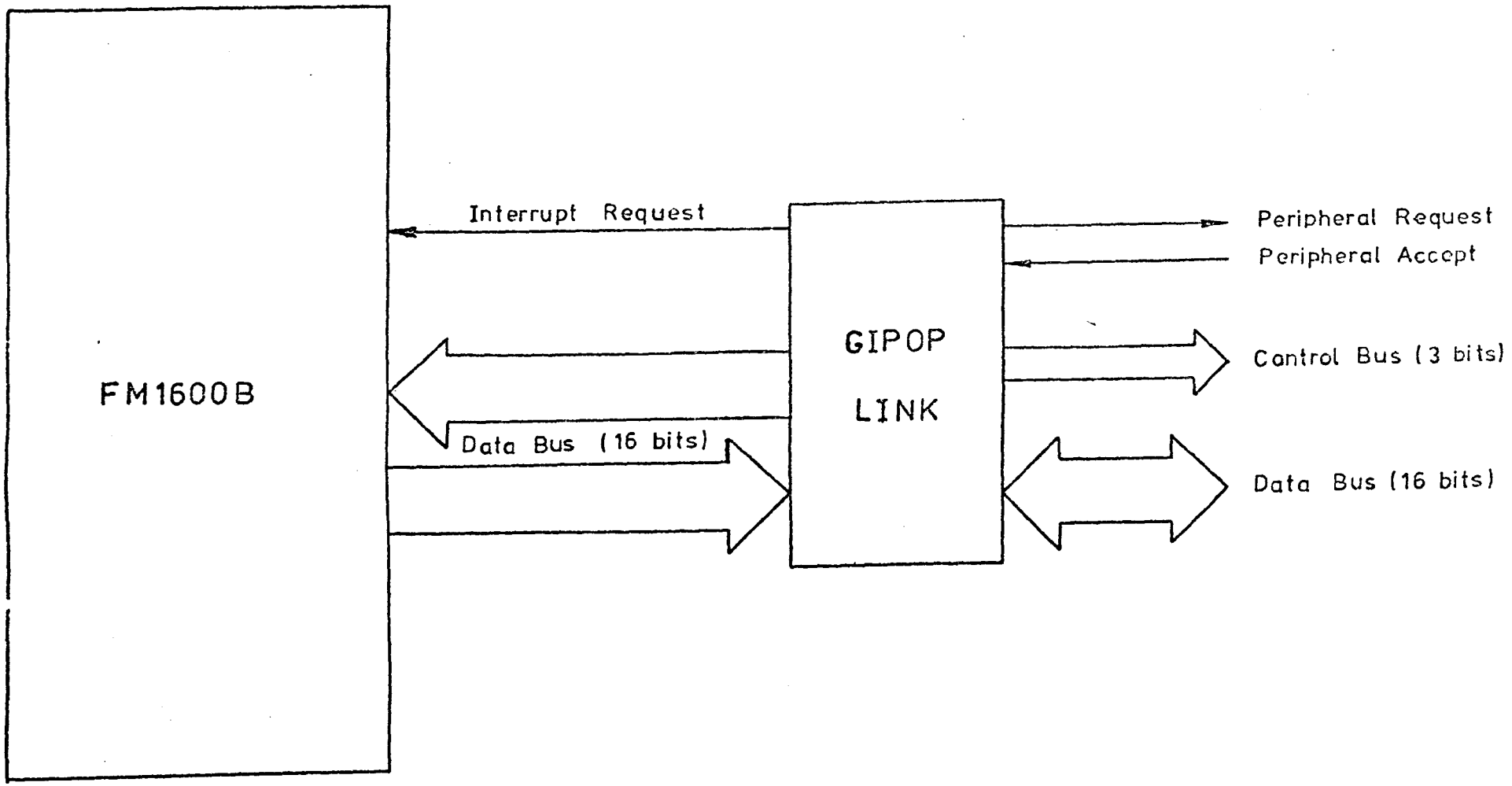


Figure 1.1 Block diagram of Gipop link system.

initiation of any operations.

A second reset input is also provided so that the user may reset the link hardware manually (e.g. under peripheral control, so the link is reset when the peripheral is reset).

1.1.3 Circuit Description

1.1.3.1 Control Latch

This circuit consists of a four-bit latch and associated buffers, into which the lower four bits of the Gipop highway are latched. Three outputs are provided for the user. These are presented as open-collector outputs and must be pulled-up to V_{CC} via a suitable resistor (330Ω) on the peripheral board. The recommended receiving circuit is as shown in Figure 1.2.

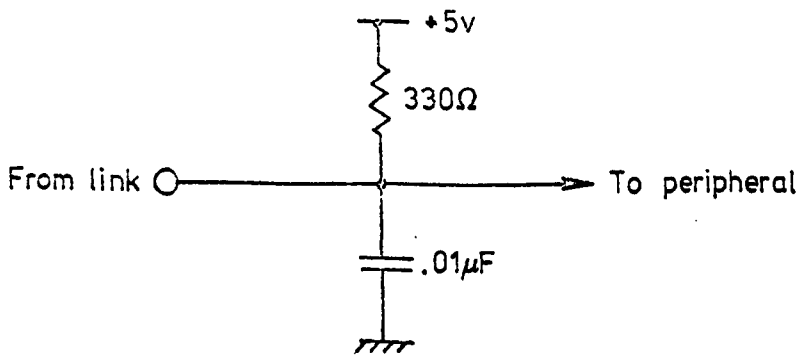


Figure 1.2 Recommended pull-up circuit.

Pull-up resistors are provided on the link board (R47, R48 and R49) but they are disconnected from the V_{CC} supply. This supply may be connected by the user if desired, but the use of this pull-up arrangement is not recommended.

The remaining latched output is used for internal control, and sets the direction of data transfer (read or write).

Note, the inputs to the latches are taken from the FM1600B side of the (data out) Gipop highway and as a result, the read-write control output has no effect on them.

1.1.3.2 Edge-Detector Circuit

The clock input to the latches is provided by the Relay 1 input via an edge-detector circuit. This is of a standard design comprising two D-type flip-flops and a NAND gate, as shown in Figure 1.3.

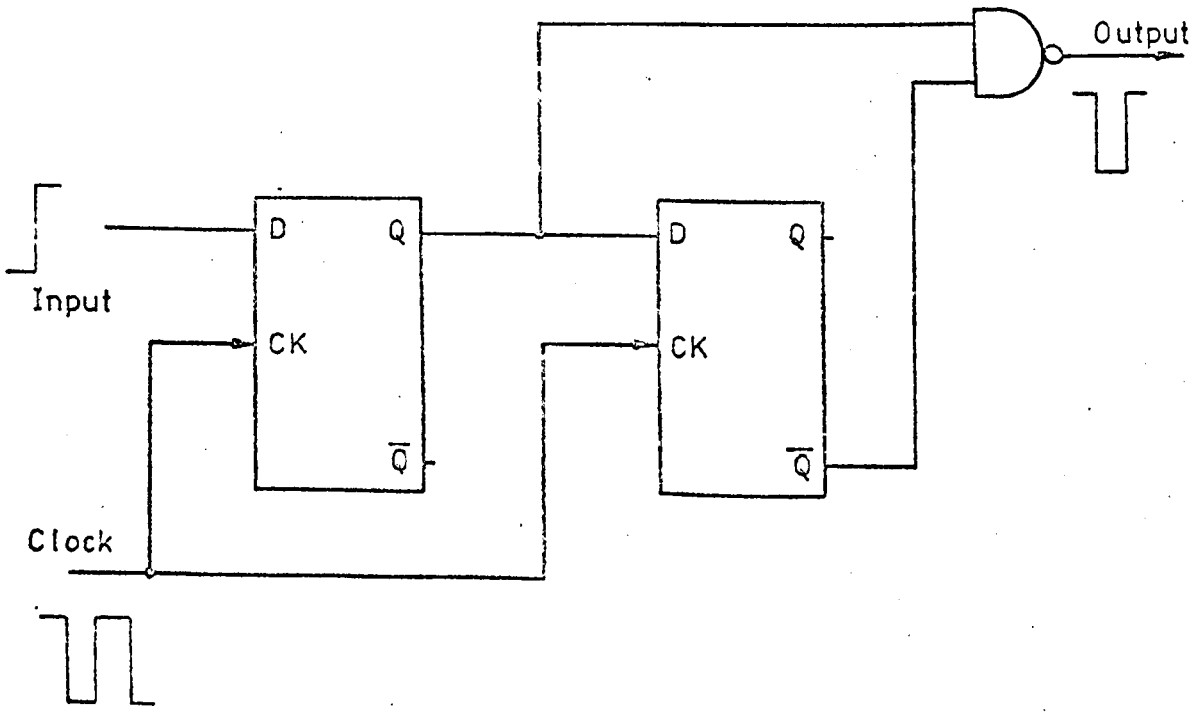


Figure 1.3 Edge-detector circuit.

1.1.3.3 Relay Inputs

A low-pass filter circuit with a 1.2ms (approx.) time constant is included to de-bounce both relay inputs. These are followed by Schmitt gates to 'square-up' the inputs before they are applied to the rest of the circuitry (Fig. 1.4).

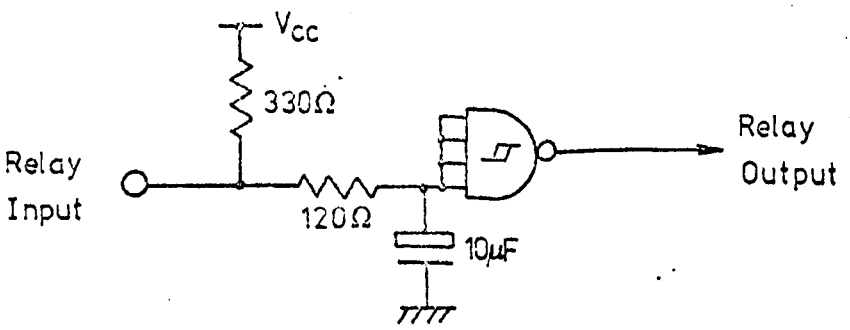


Figure 1.4 Relay de-bounce circuit.

Relay 1 is used to provide the clock input for the control latch and Relay 2 to generate the Peripheral Request ($\overline{\text{PeRq}}$) line of the handshake pair.

When set, the relay input (to the link) is connected to ground and when clear the input is floating.

1.1.3.4 Reset Input

Two reset inputs are provided, one under software control via the FM1600B and the other a manual input which must be taken to ground to effect a reset.

Upon start-up, the Gipop outputs will be in some indeterminate state, with the result that Relay 2 may be set thus generating a permanent peripheral request. This situation would not be rectified until the first Gipop scan were performed, with the result that an interrupt error would be generated immediately.

To overcome this problem a latch is used to provide an enable control for the $\overline{\text{PeRq}}$ line. This latch may be cleared by either reset input and is not set until after the control word has been latched. Note, this requires that Relay 1 be set at least once before the Relay 2 input can have any affect on the circuit.

1.1.3.5 Oscillator Circuit

A small oscillator circuit, running at $\simeq$ 30kHz, is included to provide the clock input to the control latches, via the edge-detector circuit. The circuit is as shown in Figure 1.5.

1.1.3.6 Bi-Directional Driver-Receivers

Sixteen bi-directional driver-receiver circuits are provided, each comprising two open-collector gates and two pull-up resistors (see Figs. 1.7 and 1.8). The highway from

the FM1600B is pulled-up on the link board (R0 - R15) and also the highway to the user (R20 - R35). Hence, the user need not provide any pull-up resistors on a peripheral board designed to operate with the link.

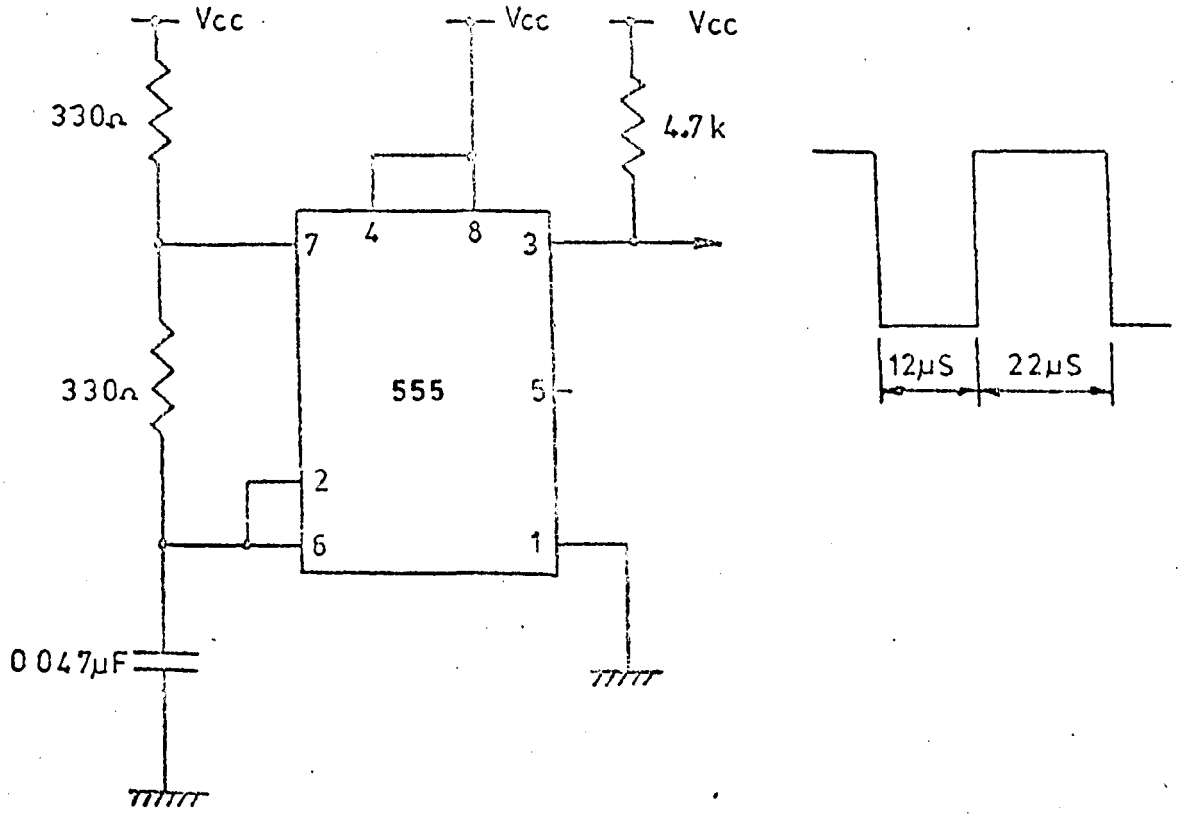


Figure 1.5 Oscillator circuit and timing waveform.

1.1.3.7 Power Supply

The link is provided with a self contained 5V @ 1A power supply unit of conventional design (see Fig. 1.19). Note, the 0V supply is separated from earth so as to avoid earth loop problems with the FM1600B and other peripheral circuits.

1.1.3.8 Power Supply Considerations

The FM1600B uses a 4.5V supply voltage and no input should be allowed to go above this value. The TTL circuits in the link, however, require a 5V supply and so care must be taken in the interconnection of the two systems.

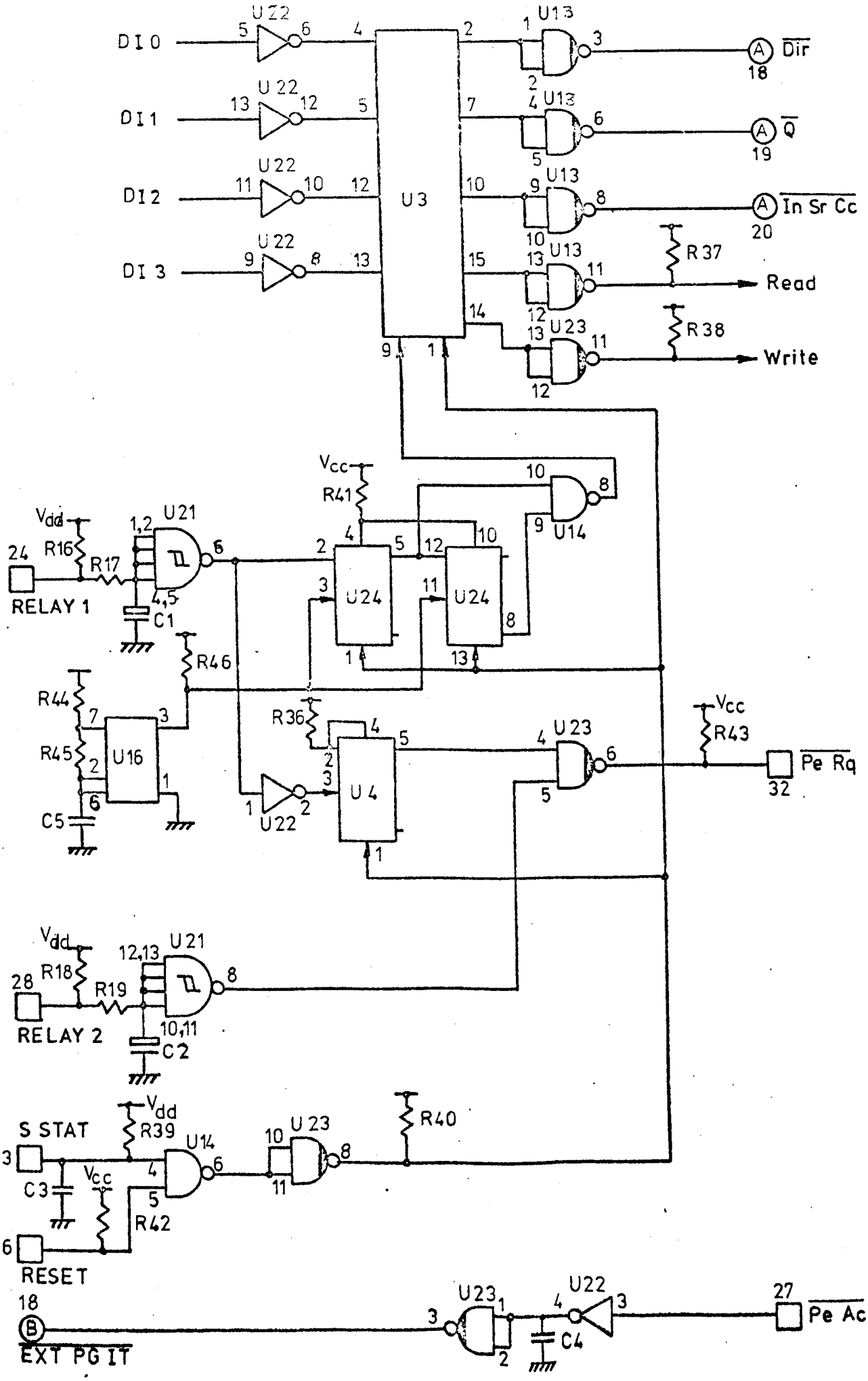


Figure 1.6 Overall circuit diagram for control section.

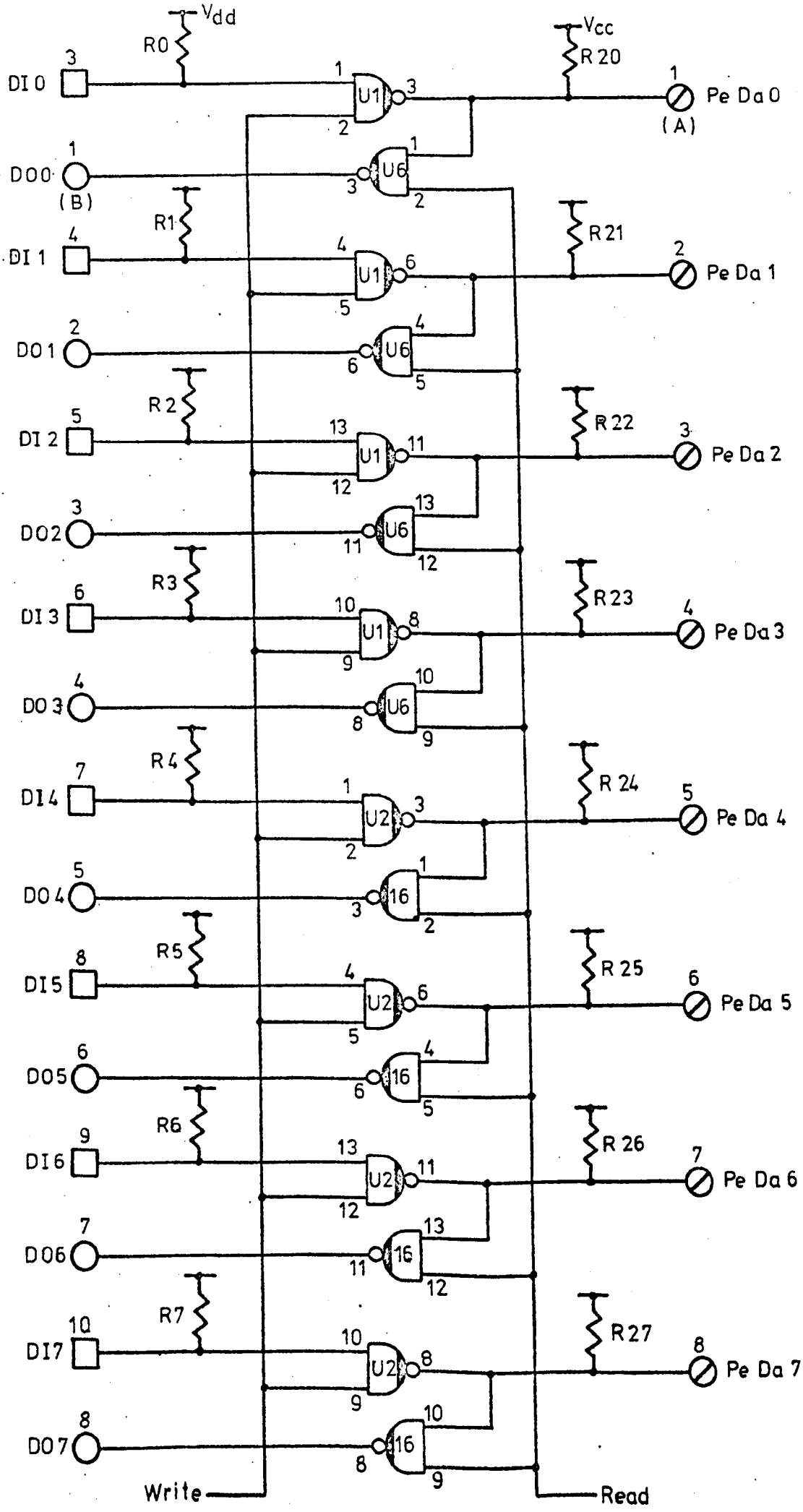


Figure 1.7 Bi-directional drivers/receivers 0 - 7.

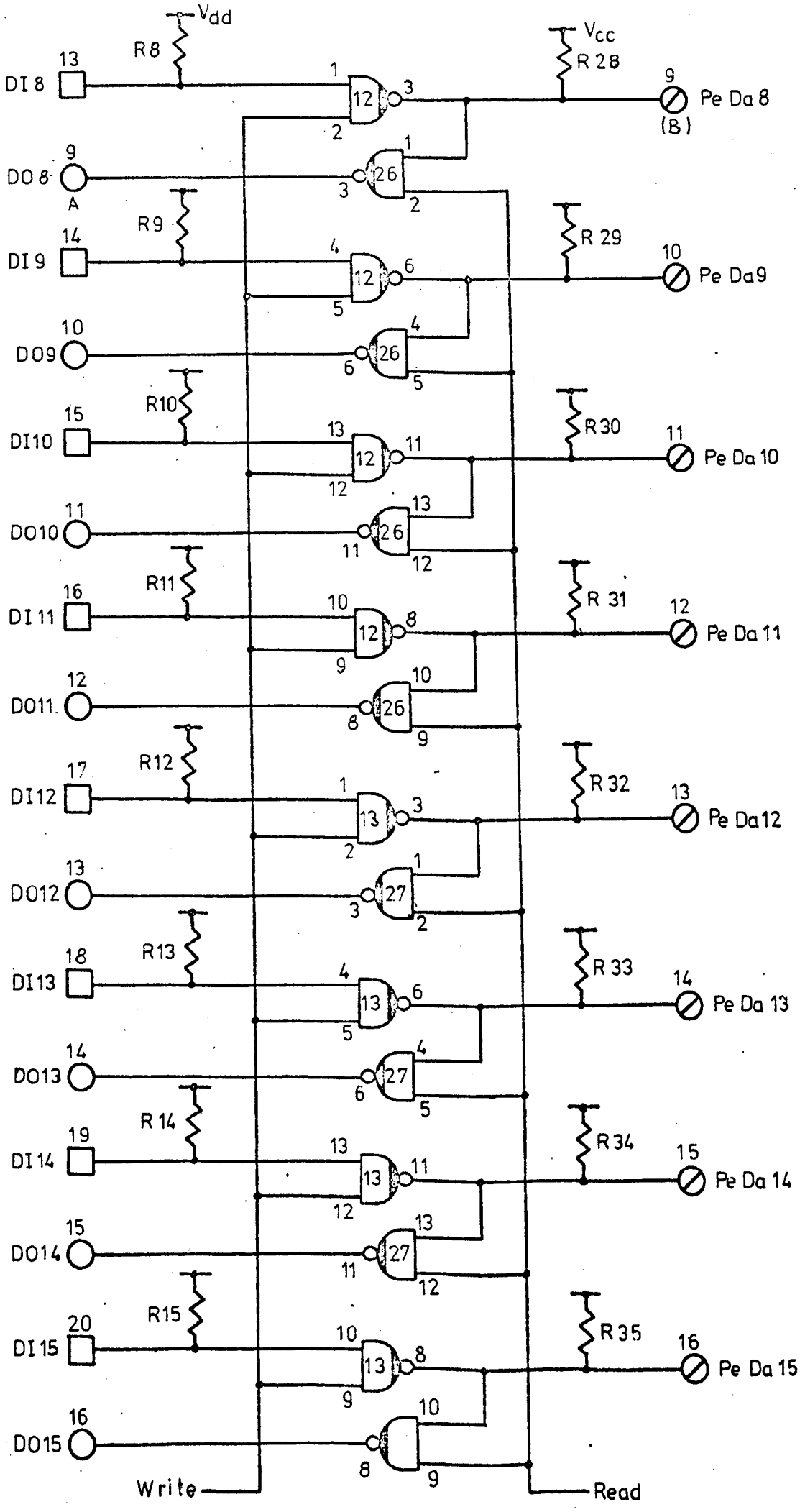


Figure 1.8 Bi-directional drivers/receivers 8 - 15.

A 4.2V supply (V_{dd}) is generated from the 5V supply (V_{cc}) of the link by the use of a diode (see Fig. 1.19) and this is used on any lines to or from the FM1600B requiring a pull-up voltage.

1.1.3.9 Mechanical Construction

All of the circuitry needed for the link is housed in a small aluminium box mounted on the laboratory wall in the Level 4 laboratory of the Electrical and Electronic Engineering Department, The City University, London. For mechanical details see Figures 1.20 to 1.22.

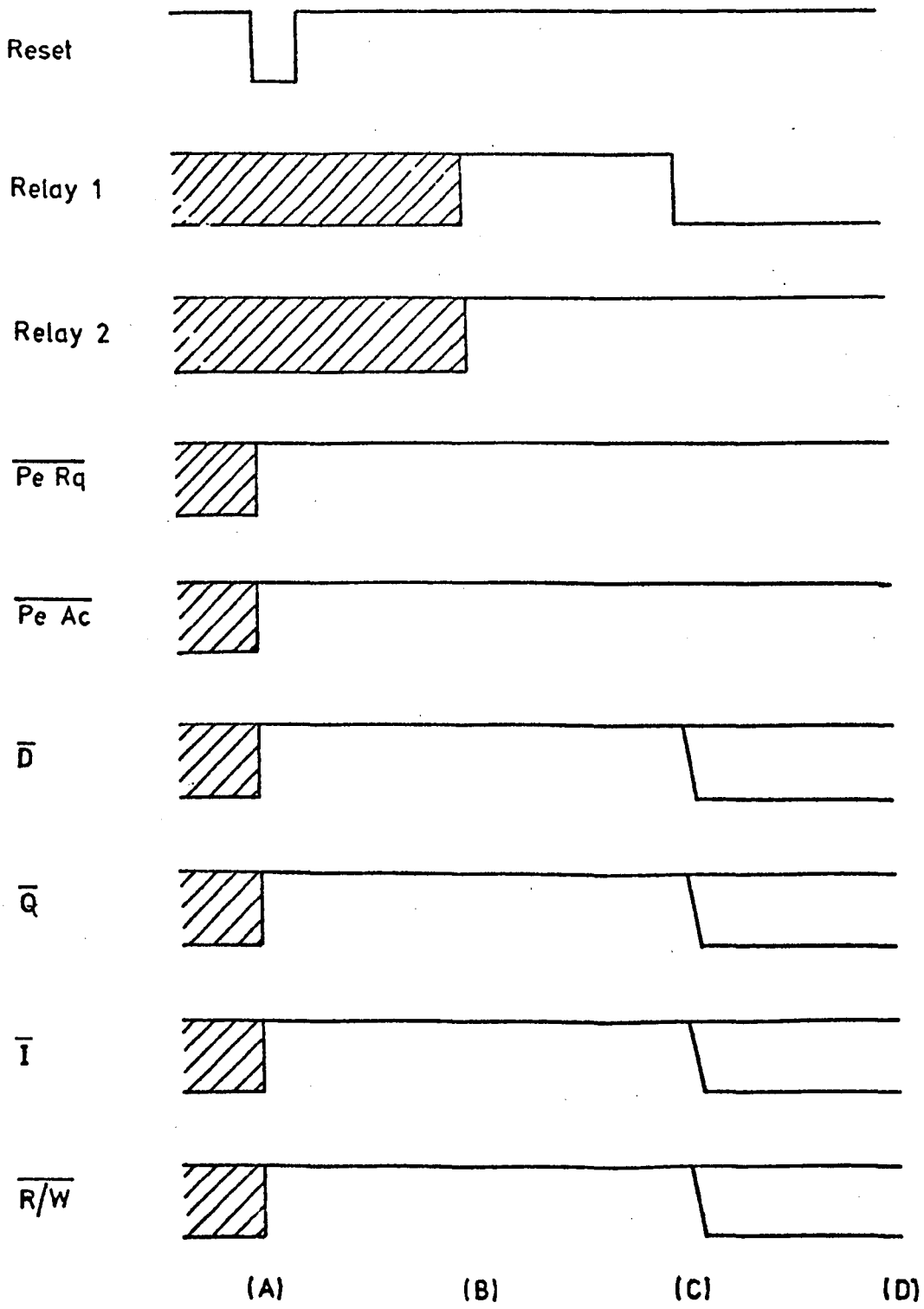
1.1.4 Circuit Operation

The start of any sequence should be to reset the link hardware, and this can be achieved by taking either reset input low. The data output word and relay control word should now be cleared and a scan initiated so as to set all of the link inputs to a known state.

Once this has been done the required code word should be loaded into the Gipop data output word location (see S8090), Relay 1 set and a second scan initiated. This causes the code word to be latched into the control word latch and enables the $\overline{\text{PeRq}}$ output. The system is now ready to commence read-write operations (see Fig. 1.9).

For a read operation bit three of the control word must be low and for a write, it must be high. This signal is then inverted and used to control the bi-directional latches.

If the link is set for a read operation, then the relay control word should be loaded to set Relay 2 and positive-going external program interrupts should be enabled (n.b. the FM1600B uses a negative logic convention). A scan is now initiated with Relay 2 causing a peripheral request to be issued. The FM1600B then waits for the peripheral accept signal, which causes a program interrupt. The FM1600B checks



(A) Reset hardware

(B) Initiate scan to reset relays and data output

(C) Initiate scan to set Relay 1 and latch control word

(D) System ready for use

Figure 1.9 Link set-up sequence.

that the interrupt was valid and, if correct, reads the data input word from the Gipop data area. Note, end of scan interrupts are ignored.

The relay control word must now be loaded to clear Relay 2 and negative-going external program interrupts enabled. A second scan is initiated which resets the peripheral request line and again the FM1600B waits for the peripheral accept signal via the interrupt input. If the interrupt is valid then the system is ready for the next transfer. This sequence continues until all words have been transferred (see Fig. 1.10).

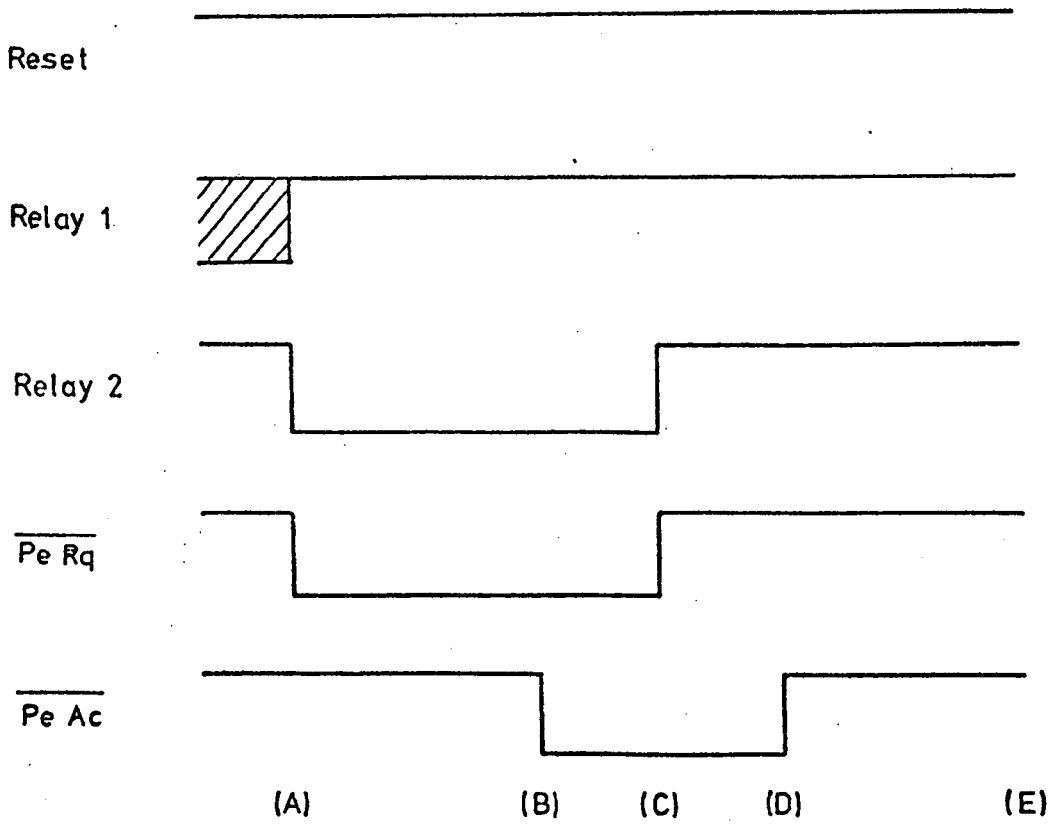
For a write operation the sequence of events is very similar to that of the read operation except that now the data output word in the Gipop data area must be loaded before a peripheral request is issued. Also, the edges of the peripheral request and accept waveforms have different meanings (see Fig. 1.11).

1.1.5 Operation with the F100 System

The link was designed primarily for use as an interface between the FM1600B and F100 systems. The following explanation of this link-up, although specific to the F100 system, can be considered as a typical example of the use of the Gipop link.

The F100 system is connected to the link via a 25-way Cannon connector and a 25-way twisted-pair cable, approximately 1m in length. This cable is connected to the Gipop buffer board of the F100 which is described in a separate document (see Chap. II.2).

A complete set of programs has been written to control the FM1600B and F100 link-up (see later). These enable the user to specify the type of operation required (i.e. read or write), a start address of a data area in the F100's store, a start address of a data area in the FM1600B's store



(A) Initiate scan to set relay 2 and generate peripheral request

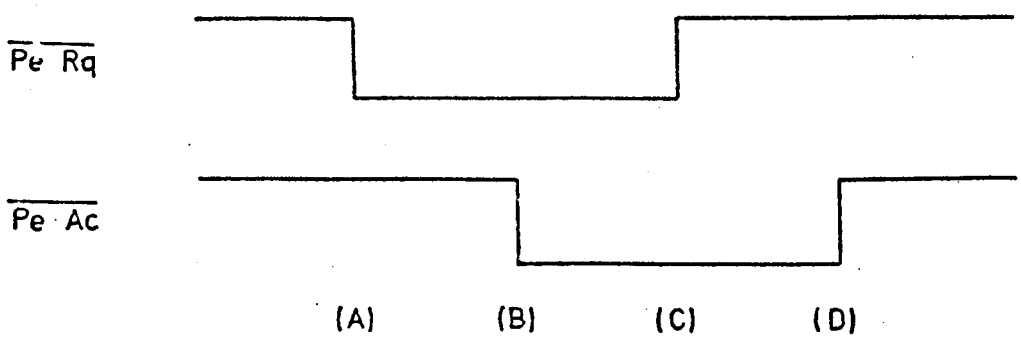
(B) Peripheral accept generates Ext. Pg It; data on bus

(C) Initiate scan to clear Relay 2 and $\overline{\text{Pe Rq}}$

(D) Peripheral accept causes Ext. Pg It; end of cycle

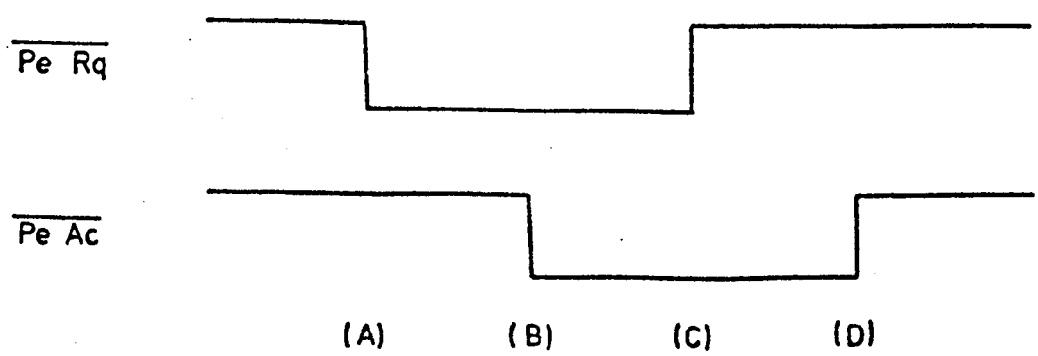
(E) Ready for next read.

Figure 1.10 Read cycle.



(A) Transfer request
(B) Transfer accepted
(C) Data on bus
(D) Data accepted

(a) Read Cycle



(A) Transfer request
(B) Data on bus
(C) Data accepted
(D) Data removed from bus

(b) Write Cycle

Figure 1.11 Peripheral request and accept handshake lines.

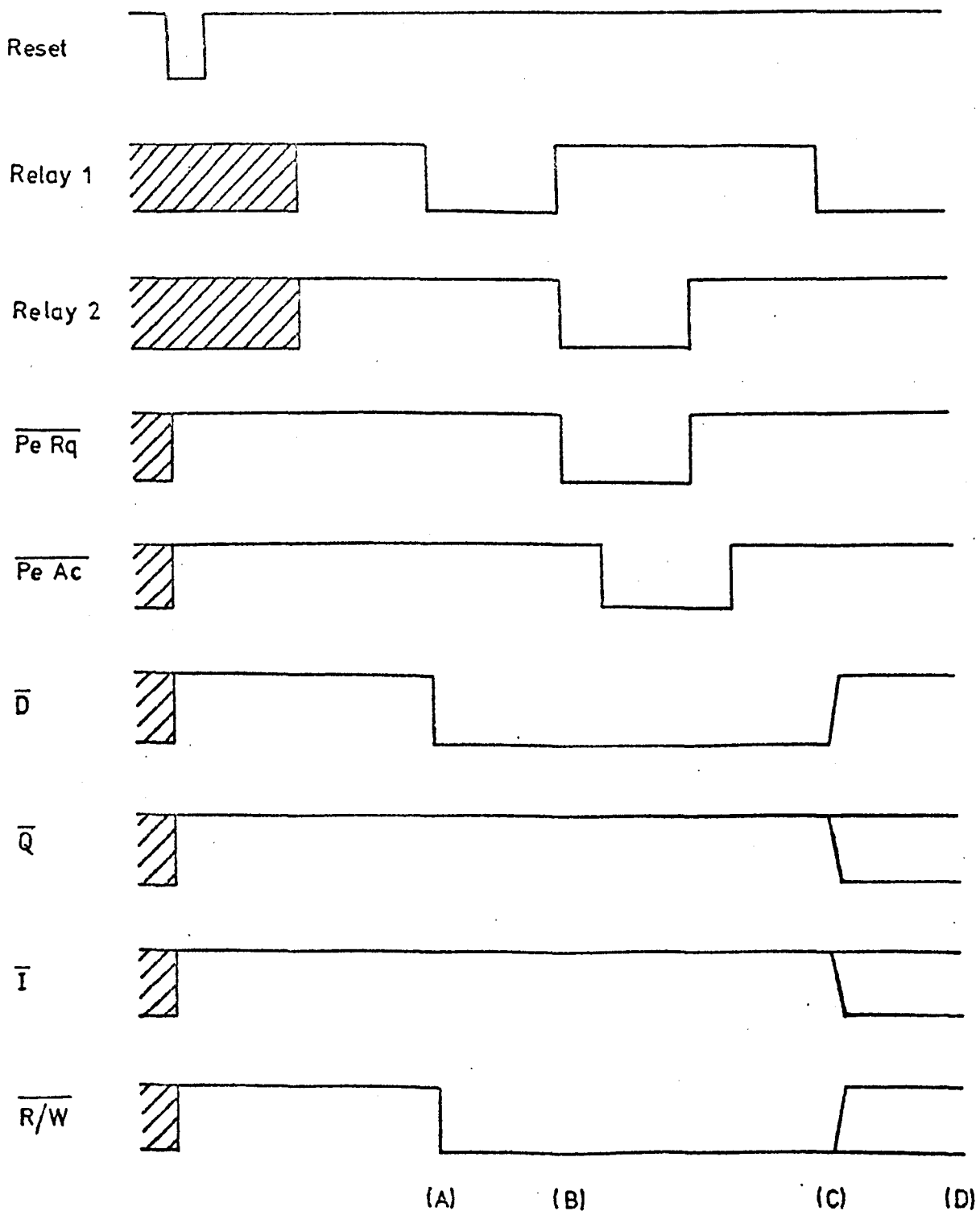
and the length of the block to be transferred. This is normally performed under teletype control, the user receiving appropriate prompts. When these three parameters have been entered the system is ready for use.

The link set-up procedure is as explained earlier in the circuit operation description (sect. 1.1.4). In addition to the link set-up, the F100 must also be set-up, which requires that the F100 address be written into an address register. To do this, a code word corresponding to the 'load address counter' code word for the F100 is latched into the control latch and a transfer initiated to write the start address to the F100. The control word specified by the user (i.e. read or write) is now transferred to the control latch and both the link and the F100 are now set-up and ready for use. The timing waveforms for the complete set-up sequence are as shown in Figure 1.12.

The read and write operations are performed in exactly the same manner as described in the circuit operation section. These transfers continue until the number of transfers specified by the 'length' input have been completed.

A complete set of program listings and flow diagrams is included for this example as most of them will be required for any application using the Gipop link; only the control and set-up (S8080) programs need be changed.

The Gipop-F100 delay routine (S8085) is used to set a software delay of 10ms (approx.) necessary to let the relay outputs settle before the program can proceed. This program is self explanatory as is the Gipop Data Area subroutine (S8090) and no flow diagrams are included.



(A) Latch 'load address counter' code word.

(B) Load F100 address to address counter.

(C) Latch F100 control word.

(D) System ready for read/write operations.

Figure 1.12 F100 set-up sequence.

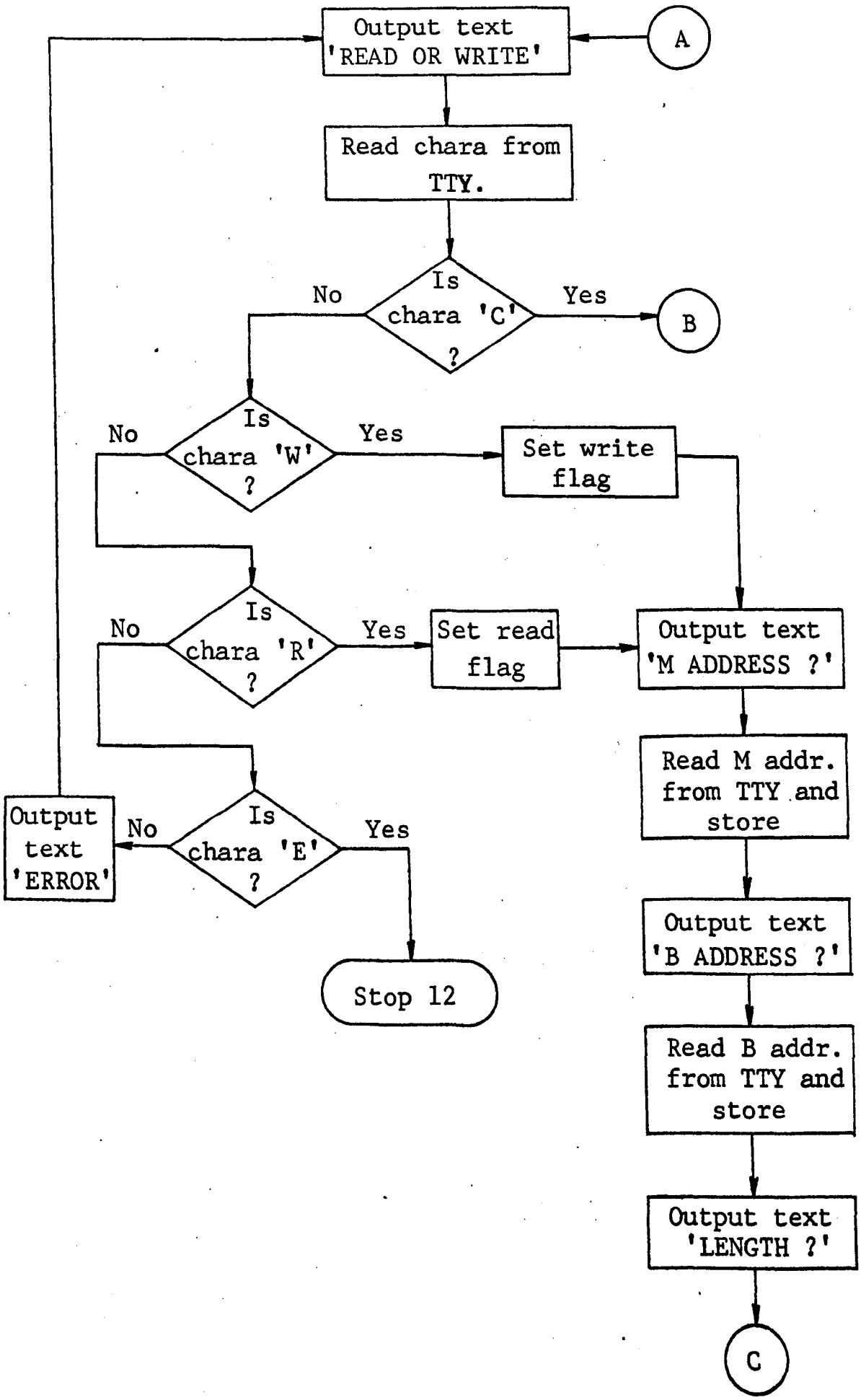


Figure 1.13(a) Control program for Gipop-F100 interface.

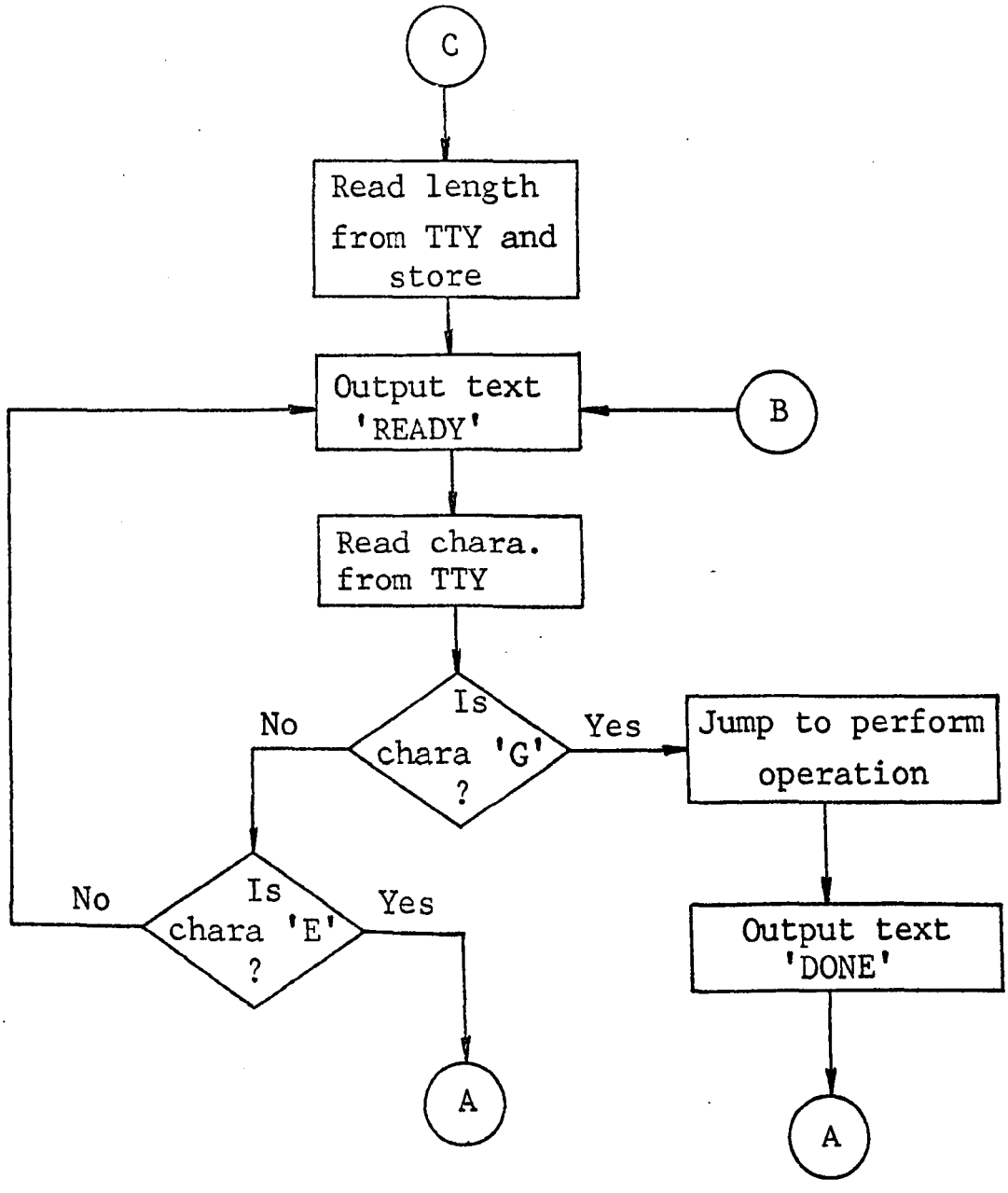


Figure 1.13(b) Control program for Gipop-F100 interface (cont.)

```
N
CONTROL PROGRAM FOR GIPOP/F100 INTERFACE - R.A.C. 1/7/77;

U

[1]V19=6
   TEXT 1P 1E
   READ OR WRITE?
   V19=5
  ←S555,1

  ←3,V21#VNO=0
   +195                    'C-CONTINUE'
   V2=14
  ←2,V21#VNO=0
   +215                    'W-WRITE'
   V2=5
  ←2,V21#VNO=0
   +210                    'R-READ'
  ←5,V21#VNO=0
   +197                    'E-END'
   V19=6
   TEXT 1P 1E
   ERROR
  ←1

[2]V19=6
   TEXT 1P 1E
   M ADDRESS?
   V19=5
   V23=0
  ←S100
   V11=V21-1               'V11=M ADDR'

   V19=6
   TEXT 1P 1E
   B ADDRESS?
   V19=5
   V23=0
  ←S100
   V12=V21                 'V12=B ADDR'

   V19=6
   TEXT 1P 1E
   LENGTH?
   V19=5
   V23=0
  ←S100
   V14=V21                 'V14=LENGTH'

[3]V19=6
   TEXT 1P 1E
   READY
   V19=5
  ←S555,1                  'INPUT G(GO) OR E(ERROR)'
  ←4,V21#VNO=0
   +71                     'G'
  ←1,V21#VNO=0
   +197                    'E'
```

```
  ←3

[4]←S8080
   V19=6
   TEXT 1P 1E
   DONE
  ←1

[5]V19=4
   V22=12
  ←S555.6                    'STOP 12'
  ←1

END

A
```

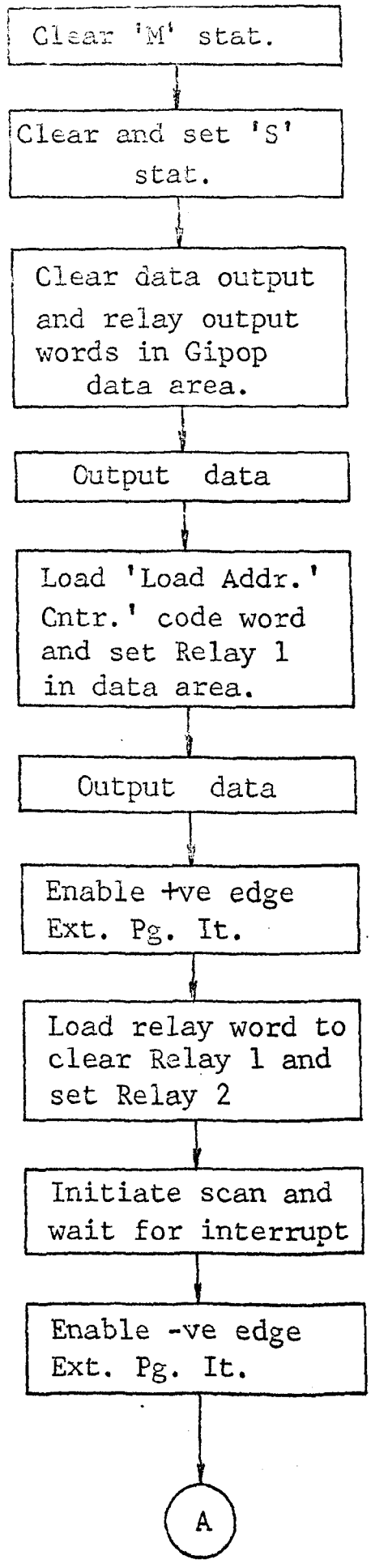


Figure 1.14(a) Gipop-F100 set-up routine (S8080).

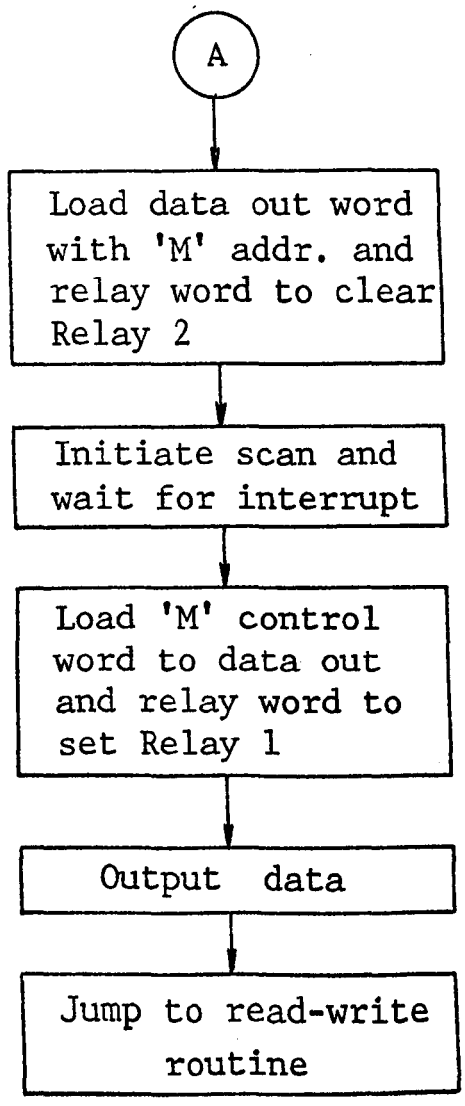


Figure 1.14(b) Gipop-F100 set-up routine (cont.).

```
N
S8080 GIPOP/F100M SET-UP ROUTINE - R.A.C. 1/7/77;

U

    VN3=V10,N3-1
    VN3=N1,N3-1
    VN3=N2,N3-1

    QM320=0
    QS320=0
    QS320#0                    'RESET HARDWARE'

    N1=VN0
    AS8090,2
    VN1=0                      'CLEAR DATA OUTPUT WORD'
    VN1=0,N1+1                 'CLEAR RELAY OUTPUTS'
    V10=0
    V10[1]#0                   'SET INTERRUPT FLAG'
   ←S8084                      'OUTPUT DATA'

   ←S8085                      'WAIT FOR RELAYS TO SETTLE'

    N1=VN0
    AS8090,2
    VN1=9                      'DATA=LOAD ADDR CNTR CODE WD'
    VN1=1,N1+1                 'SET RELAY 1'

    V10[1]#0
   ←S8084                      'OUTPUT DATA'

   ←S8085

    V10[2]=0                   'ENABLE +VE EDGE EXT PG IT'

    N1=VN0
    AS8090,3
    VN1=2                      'CLEAR RELAY 1;SET RELAY 2'
    V10[1]=0
   ←S8084                      'INITIATE SCAN'

   ←S8085

    V10[2]#0                   'ENABLE -VE EDGE EXT PG IT'

    N1=VN0
    AS8090,2
    VN1=V11                    'VN1=M ADDR'
    VN1=0,N1+1                 'CLEAR RELAY 2'
    V10[1]=0
   ←S8084                      'INITIATE SCAN'

   ←S8085

    N1=VN0
    AS8090,2
    VN1=V2                     'VN1=M CONTROL WORD'

    VN1=1,N1+1                 'SET RELAY 1'
```

```
 V10[1]#0                    'RESET FLAG'
←S8084                       'OUTPUT CODE WORD'

←S8085

←S8088                       'JUMP TO READ/WRITE ROUTINE'

 N2=VN3
 V0=V0,N3+1
 N1=VN3
 V0=V0,N3+1
 V10=VN3,N3+1

←L
END

S
8080
GIPOP/F100M SET-UP ROUTINE - R.A.C. 1/7/77;
```

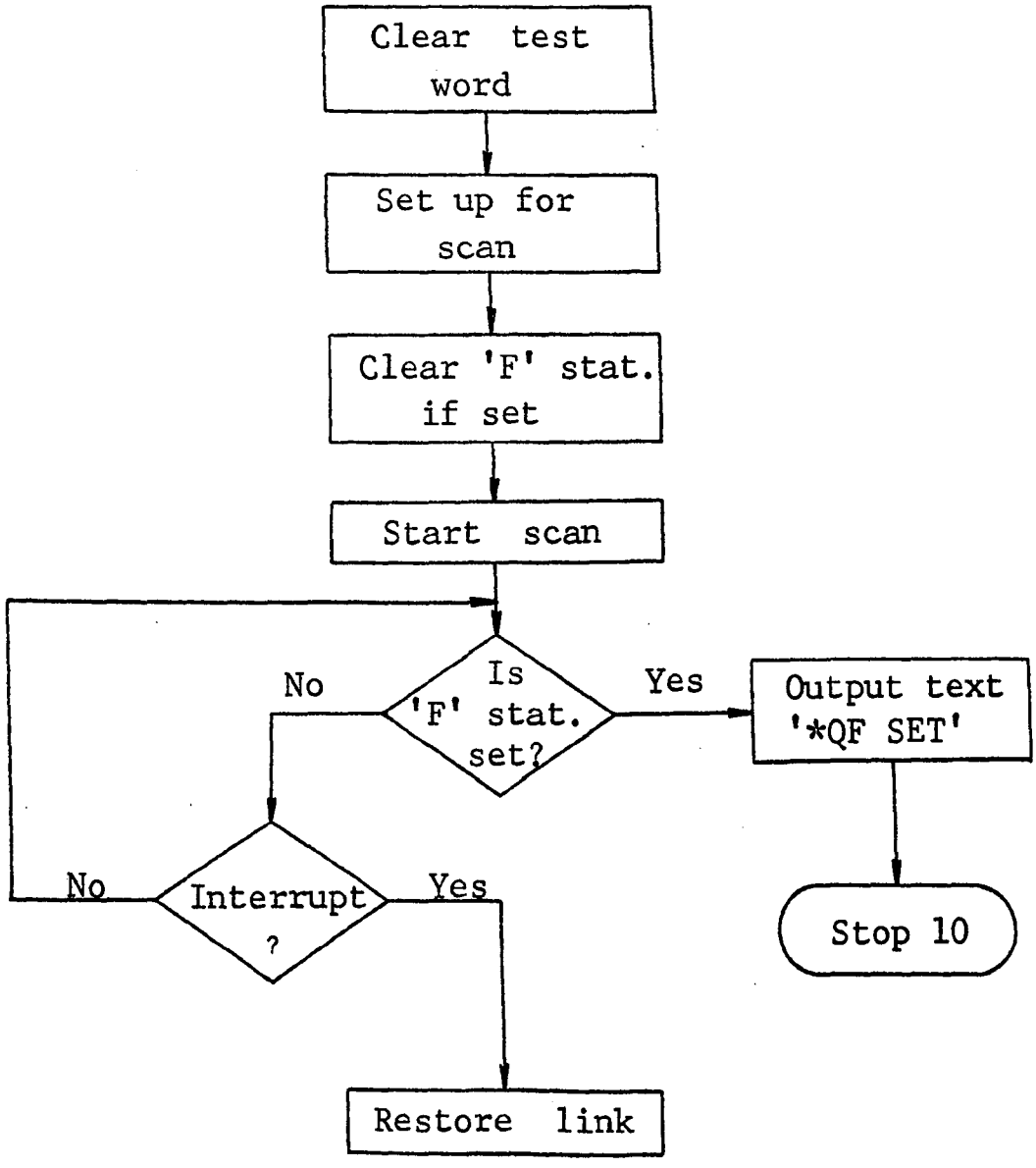


Figure 1.15 Initiate scan and wait for interrupt (S8084).

```
N
S8084 INITIATE SCAN AND WAIT FOR INTRRUPT - R.A.C. 1/7/77;

U

   VN3=N1,N3-1
   N1=VN0
   AS8090
   VN1=0                        'CLEAR TEST WORD'

   N1=VN0
   IB320,DP
   VN1=VN0
   AS8090                       'SET UP SCAN'

   VN1=VN0,N1+1
   AS8089                       'SET UP INT ADDR'

   QF320=0                      'CLEAR F STAT IF SET'

   QM320#0                      'INITIATE SCAN'
[0]←1,QF320#0                   'JUMP TO DRUID STOP IF F STAT SET'
  ←0,V10[1]=0                   'WAIT FOR INTERRUPT'
   N1=VN3
   V0=V0,N3+1
  ←L

[1]V19=6
   TEXT 1P 1E
   * QF SET
   V19=4
   V22=10
[10]←S555,6
  ←10
END

S
8084
INITIATE SCAN AND WAIT FOR INTERRUPT - R.A.C. 1/7/77;
```

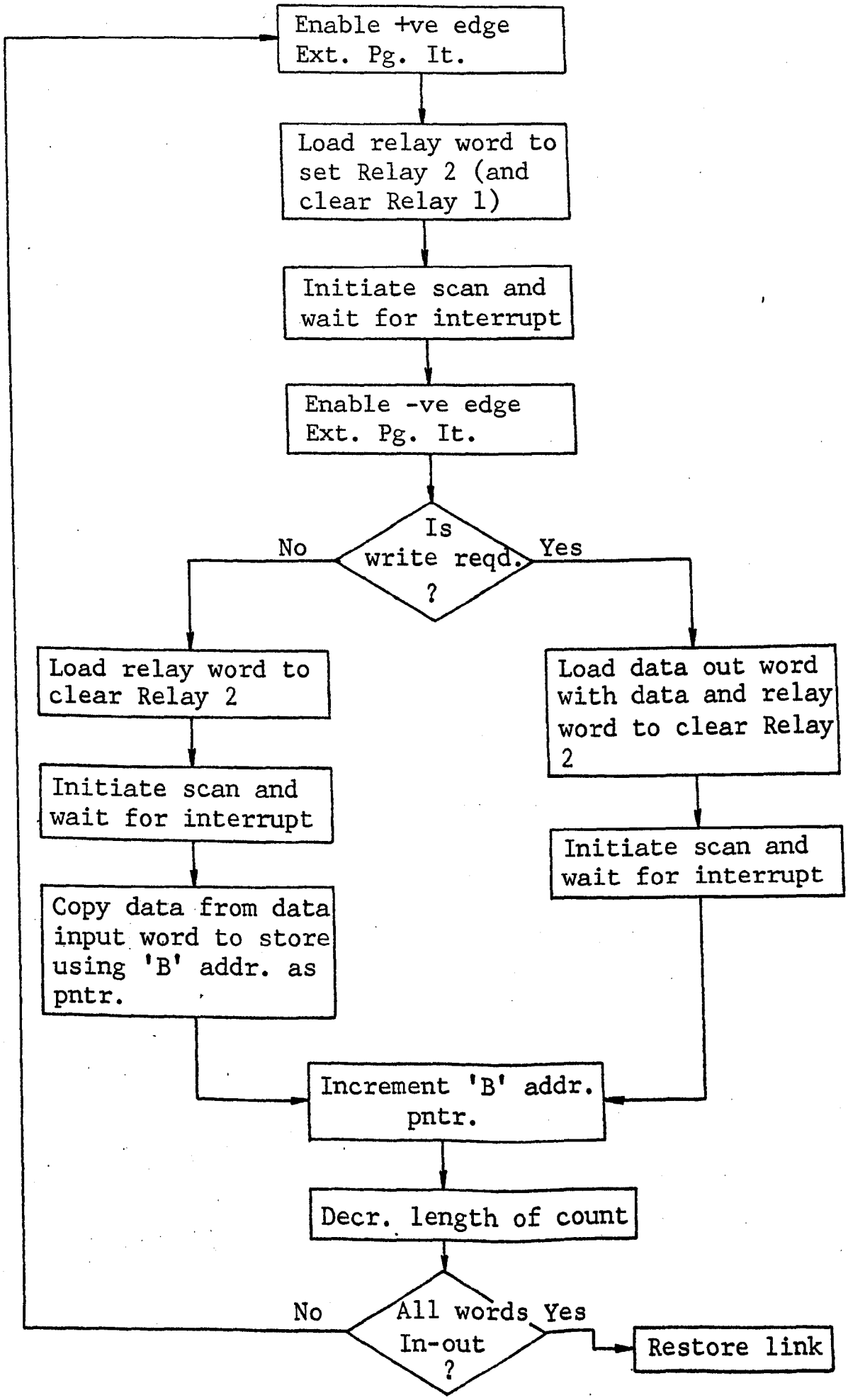


Figure 1.16 Gipop-F100 read-write routine (S8088).

```
N
S8088 GIPOP/F100M READ/WRITE ROUTINE - R.A.C. 1/7/77;

U
   VN3=V8,N3-1
   VN3=V12,N3-1
   VN3=V14,N3-1
   VN3=N1,N3-1
   VN3=N2,N3-1
   V14=V14-1
[1]V10[2]=0                    'ENABLE +VE EXT PG IT'

   N1=VN0
   AS8090,3
   VN1=2                       'CLEAR RELAY 1;SET RELAY 2'
   V10=0
  ←S8084                       'INITIATE SCAN'
  ←S8085

   V10[2]#0                    'ENABLE -VE EXT PG IT'
  ←2,V2-10>0                   'JUMP IF WRITE REQUIRED'

   N1=VN0
   AS 8090,3
   VN1=0                       'CLEAR RELAY 2'
   V10[1]=0
  ←S8084                       'INITIATE SCAN'
  ←S8085

   N1=VN0
   AS8090,1
   V8=VN1                      'V8=INPUT DATA'
   N1=V12                      'N1=B ADDR'
   VN1=V8                      'DATA STORED'
  ←3

[2]N1=VN0
   AS8090,2
   N2=V12,VN1=VN2&VN2          'VN1=OUTPUT DATA'
   VN1=0,N1+1                  'CLEAR RELAY 2'
   V10[1]=0
  ←S8084                       'INITIATE SCAN'
  ←S8085

[3]V12=V12+1                   'INCREMENT B DATA POINTER'
  ←1,V14=V14-1>0               'CHECK FOR ALL WORDS OUTPUT'

   N2=VN3
   V0=V0,N3+1
   N1=VN3
   V0=V0,N3+1
   V14=VN3,N3+1
   V12=VN3,N3+1
   V8=VN3,N3+1

  ←L
END

S
8088
GIPOP/F100M READ/WRITE ROUTINE - R.A.C. 1/7/77;
```

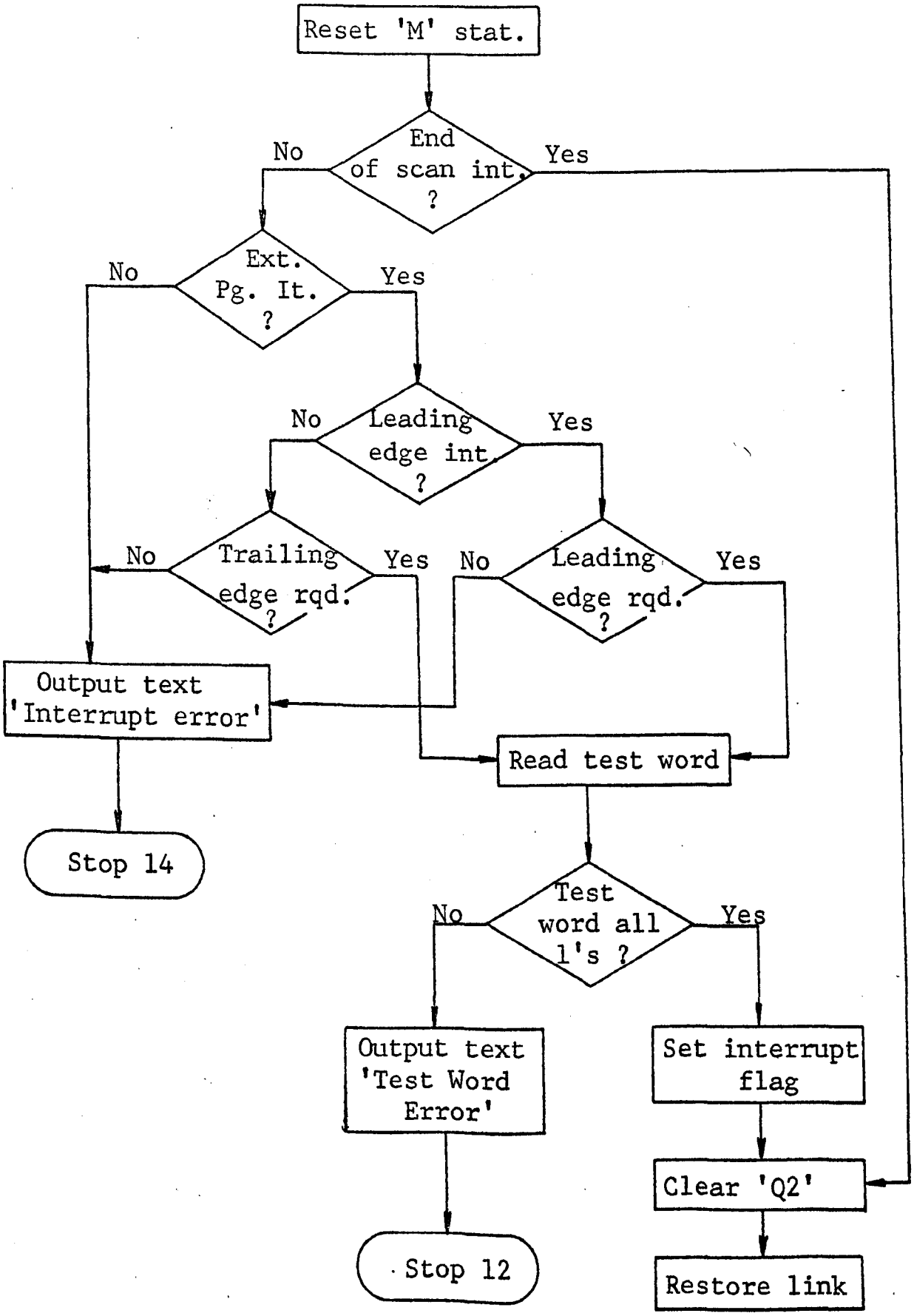


Figure 1.17 Gipop-F100 interrupt routine (S8089)

```
N
S8089 GIPOP/F100M INTERRUPT ROUTINE - R.A.C. 1/7/77;

U

   VN3=V19,N3-1
   VN3=N1,N3-1

   QM320=0                          'RESET M STAT'

   N1=VN0
   IB320,SW
  ←3,VN1[0]#0                       'END OF SCAN INTERRUPT?'
  ←5,VN1[7]#0                       'EXT PG IT ?'
[7]Q2=0
   V19=6
   TEXT 1P 1E
   * INTERRUPT ERROR
   V19=4
   V22=14
  ←S555,6                           'STOP 14'

[5]N1=VN0
   IB320,SW
  ←6,VN1[6]#0
  ←7,V10[2]#0                       'TRAILING EDGE INT. REQD.?'
[8]N1=VN0
   AS8090
  ←2,VN1#VN0#0
   OCT 00177777                     'CHECK TEST WORD ALL ONES'
   V10[1]#0                         'SET INT. FLAG'

[3]Q2=0
  ←4

[6]←7,V10[2]=0                      'LEADING EDGE INT. REQD.?'
  ←8

[2]Q2=0
   V19=6
   TEXT 1P 1E
   * TEST WORD ERROR
   V19=4
   V22=12
  ←S555,6                           'STOP 12'

[4]N1=VN3
   V0=V0,N3+1
   V19=VN3,N3+1
←L
END

S
8089
GIPOP/F100 INTERRUPT ROUTINE - R.A.C. 1/7/77;
```

```
N
S8085 GIPOP/F100M DELAY ROUTINE - R.A.C. 1/7/77;

U

   VN3=V9,N3-1
   V9=VN0
   OCT 02000                '10MS DELAY APPROX'
[1]←1,V9=V9-1>0             'LOOP TILL COUNT = ZERO'
   V9=VN3,N3+1
  ←L

END

S
8085
GIPOP/F100 DELAY ROUTINE - R.A.C. 1/7/77;
```

```
N
S8090 GIPOP DATA AREA - R.A.C. 1/7/77;

U

+0          'TEST WORD'
+0          'DATA INPUT WORD'
+0          'DATA OUTPUT WORD'
+0          'RELAY CONTROL WORD'
+0          'SWITCH INPUT WORD'
END

S
8090
GIPOP DATA AREA; - R.A.C. 1/7/77;
```

1.1.6 Miscellaneous Details

1.1.6.1 Components List

Resistors

All 330Ω ¼w ±5% except:-

R17,R19	120Ω ¼w ±5%
R36,R41,R42,R46,R50	4.7k ¼w ±5%

Capacitors

C1,C2	10μF elect. 25v
C3,C4	2.2nF ceramic
C5	.047μF polyester 150v
C6,C7,C8	0.1μF ceramic
C9	4700μF elect. 25v
C10	0.47μF polyester

Semiconductors

D1	Si diode OA200
D2 - D5	Si diode OA200
D6	Si diode 1N4004
REG1	5v regulator 7805

Integrated Circuits

U1,U2,U5,U11,U12,U13,U15,U23,U25,U26	SN7438
U3	SN74175
U4,U24	SN7474
U6	ZN344E
U14	SN7400
U16	NE555
U21	SN7413
U22	SN7404

Miscellaneous

Tr 1 - Mains transformer 4.5v @ 2.2A, 4.5v @ 2.2A

Fs 1 - Fuse (2A anti-surge) + holder

Sk 1 - 25-way Cannon connector

Sk 2 - 3 pin mains socket (Bulgin)

Sw 1 - Min-toggle DPDT

Lp 1 - 6v sub-min indicator (amber)

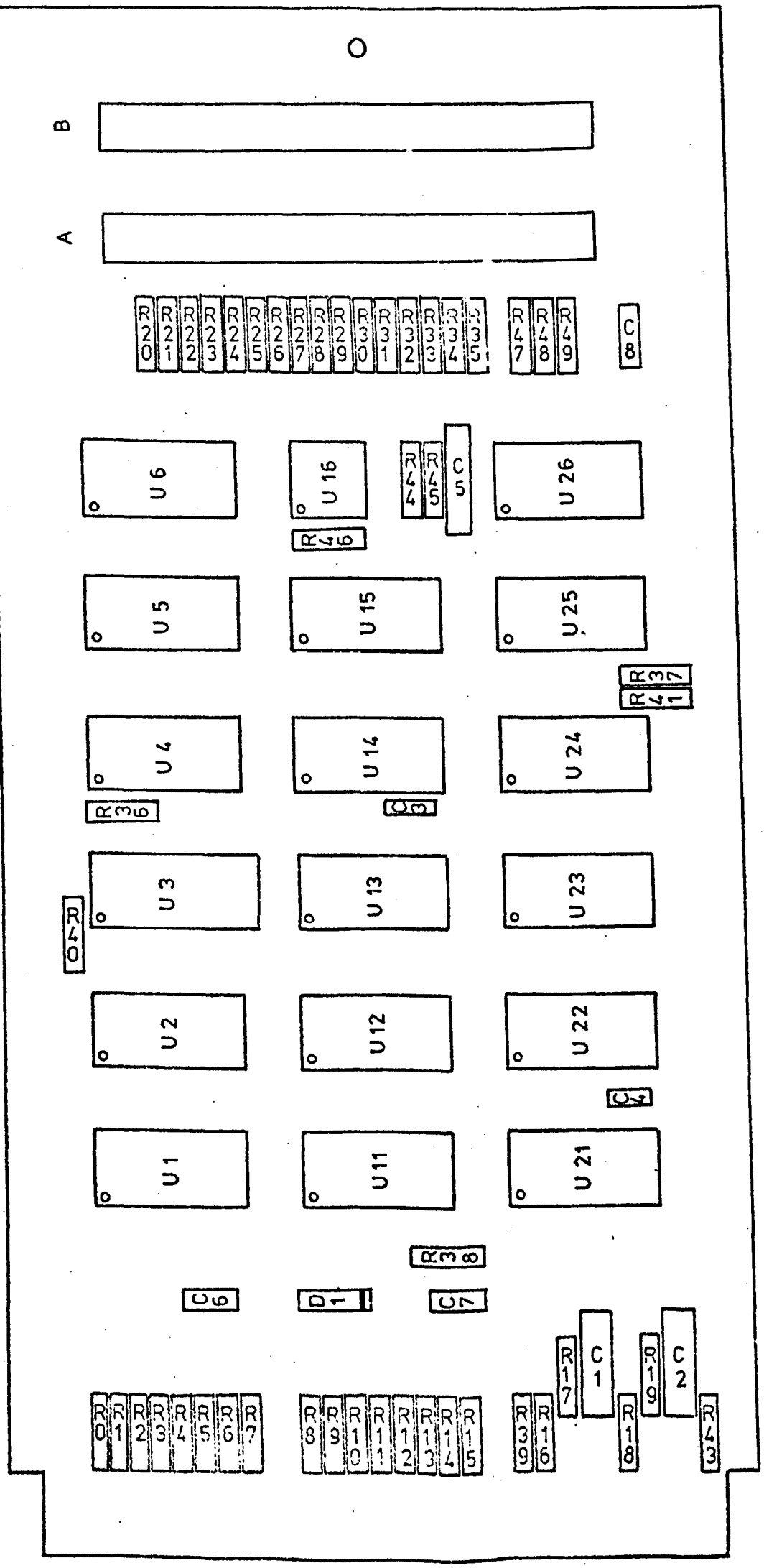


Figure 1.18 Component layout.

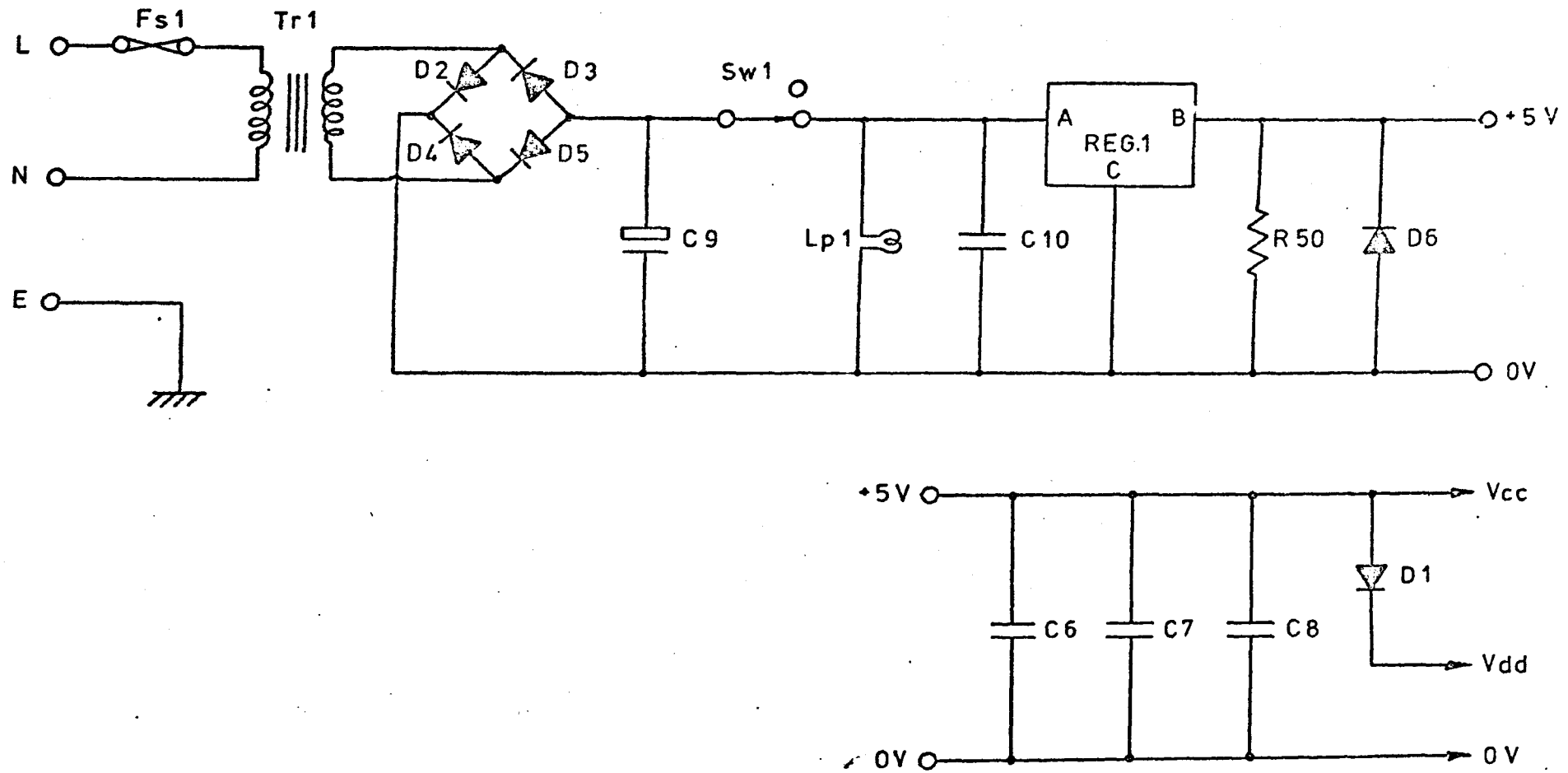


Figure 1.19 Power supply.

E.C.1 - 32-way edge connector
(A) - 20-way printed circuit connector
(B) - 20-way printed circuit connector
H/S 1 - 4°C/W heatsink
1 × R.S. Type C stripboard
2 × Tag strips

1.1.7 Connection Lists

1.1.7.1 Cannon connector

1	Pe Da 0
2	Pe Da 1
3	Pe Da 2
4	Pe Da 3
5	Pe Da 4
6	Pe Da 5
7	Pe Da 6
8	Pe Da 7
9	Pe Da 8
10	Pe Da 9
11	Pe Da 10
12	Pe Da 11
13	Pe Da 12
14	GND
15	Pe Da 13
16	Pe Da 14
17	Pe Da 15
18	$\overline{\text{Pe Rq}}$
19	$\overline{\text{Dir In}}$
20	$\overline{\text{Q}}$
21	In Sr Cc
22	$\overline{\text{Pe Ac}}$
23	$\overline{\text{Reset}}$ (Ext.)
24	$\overline{\text{Enable}}$ (GND)
25	GND

1.1.7.2 Board edge connector

1	-
2	Vcc
3	DI 0
4	DI 1
5	DI 2
6	DI 3
7	DI 4
8	DI 5
9	DI 6
10	DI 7
11	GND
12	Vcc
13	DI 8
14	DI 9
15	DI 10
16	DI 11
17	DI 12
18	DI 13
19	DI 14
20	DI 15
21	GND
22	Vcc
23	'S' stat.
24	Relay 1
25	-
26	$\overline{\text{Reset}}$ (Ext.)
27	$\overline{\text{Pe Ac}}$
28	Relay 2
29	-
30	-
31	GND
32	$\overline{\text{Pe Rq}}$

(Note: DI means data input from FM1600B)

1.1.7.3 Printed circuit connectors

	Connector (A)		Connector (B)
1	Pe Da 0	1	DO 0
2	Pe Da 1	2	DO 1
3	Pe Da 2	3	DO 2
4	Pe Da 3	4	DO 3
5	Pe Da 4	5	DO 4
6	Pe Da 5	6	DO 5
7	Pe Da 6	7	DO 6
8	Pe Da 7	8	DO 7
9	Pe Da 8	9	DO 8
10	Pe Da 9	10	DO 9
11	Pe Da 10	11	DO 10
12	Pe Da 11	12	DO 11
13	Pe Da 12	13	DO 12
14	Pe Da 13	14	DO 13
15	Pe Da 14	15	DO 14
16	Pe Da 15	16	DO 15
17	-	17	-
18	$\overline{\text{Dir In}}$	18	Ext. Pg. It.
19	$\overline{Q}$	19	-
20	In Sr Cc	20	-

(Note: DO means data output to the FM1600B)

1.1.7.4 Tag-strip connectors

	Tag-strip 1		Tag-strip 2
1	Pe Da 0	1	DO 0
2	Pe Da 1	2	DO 1
3	Pe Da 2	3	DO 2
4	Pe Da 3	4	DO 3
5	Pe Da 4	5	DO 4
6	Pe Da 5	6	DO 5
7	Pe Da 6	7	DO 6
8	Pe Da 7	8	DO 7
9	Pe Da 8	9	DO 8
10	Pe Da 9	10	DO 9
11	Pe Da 10	11	DO 10
12	Pe Da 11	12	DO 11
13	Pe Da 12	13	DO 12
14	Pe Da 13	14	DO 13
15	Pe Da 14	15	DO 14
16	Pe Da 15	16	DO 15
17	-	17	-
18	$\overline{\text{Pe Rq}}$	18	Ext. Pg. It.
19	$\overline{\text{Dir In}}$	19	-
20	$\overline{Q}$	20	-
21	In Sr Cc	21	-
22	$\overline{\text{Pe Ac}}$	22	-
23	$\overline{\text{Reset}}$ (Ext.)	23	-
24	-	24	-
25	-	25	-
26	-	26	-
27	-	27	-
28	-	28	-

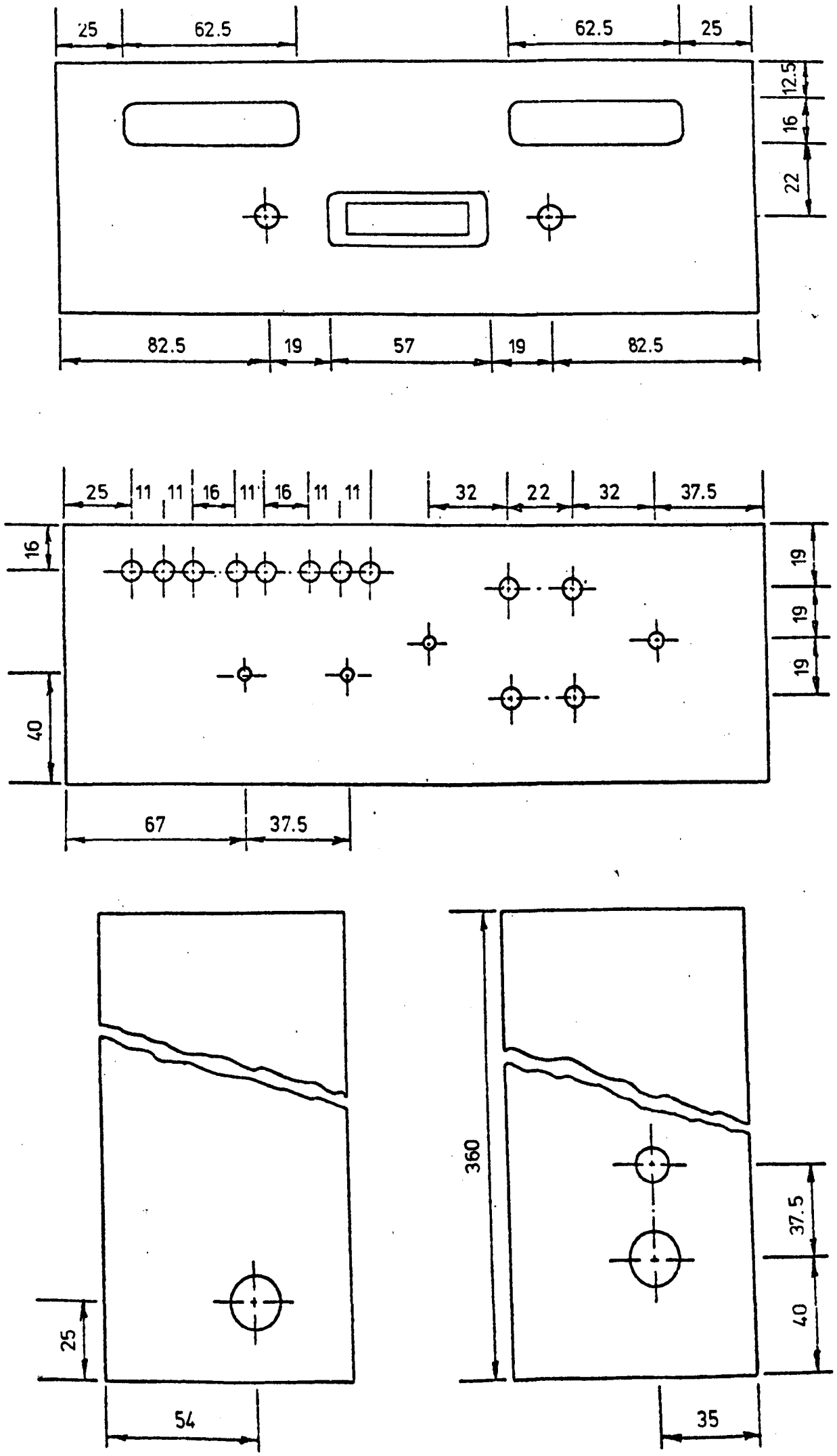


Figure 1.20 Case - main dimensions.

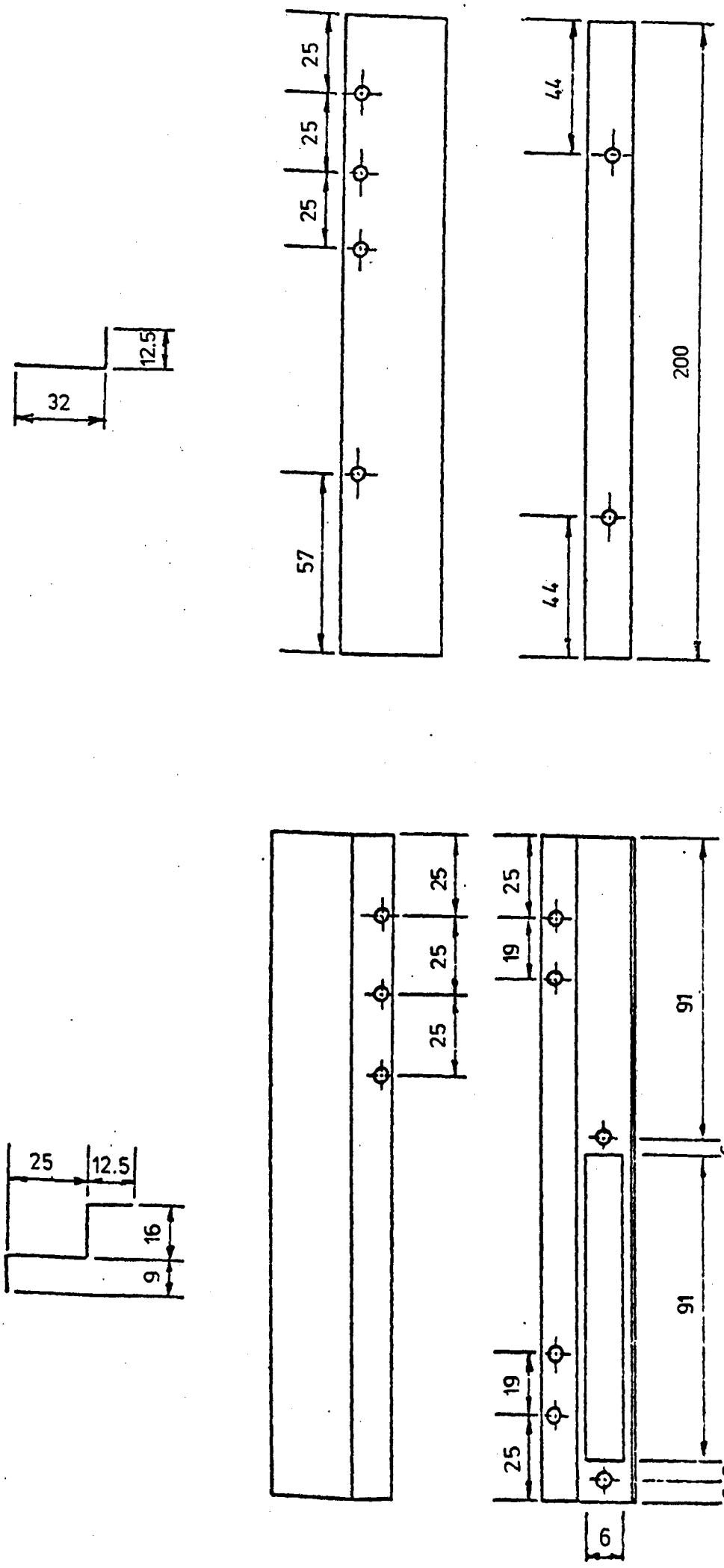


Figure 1.21 Case - inner support dimensions.

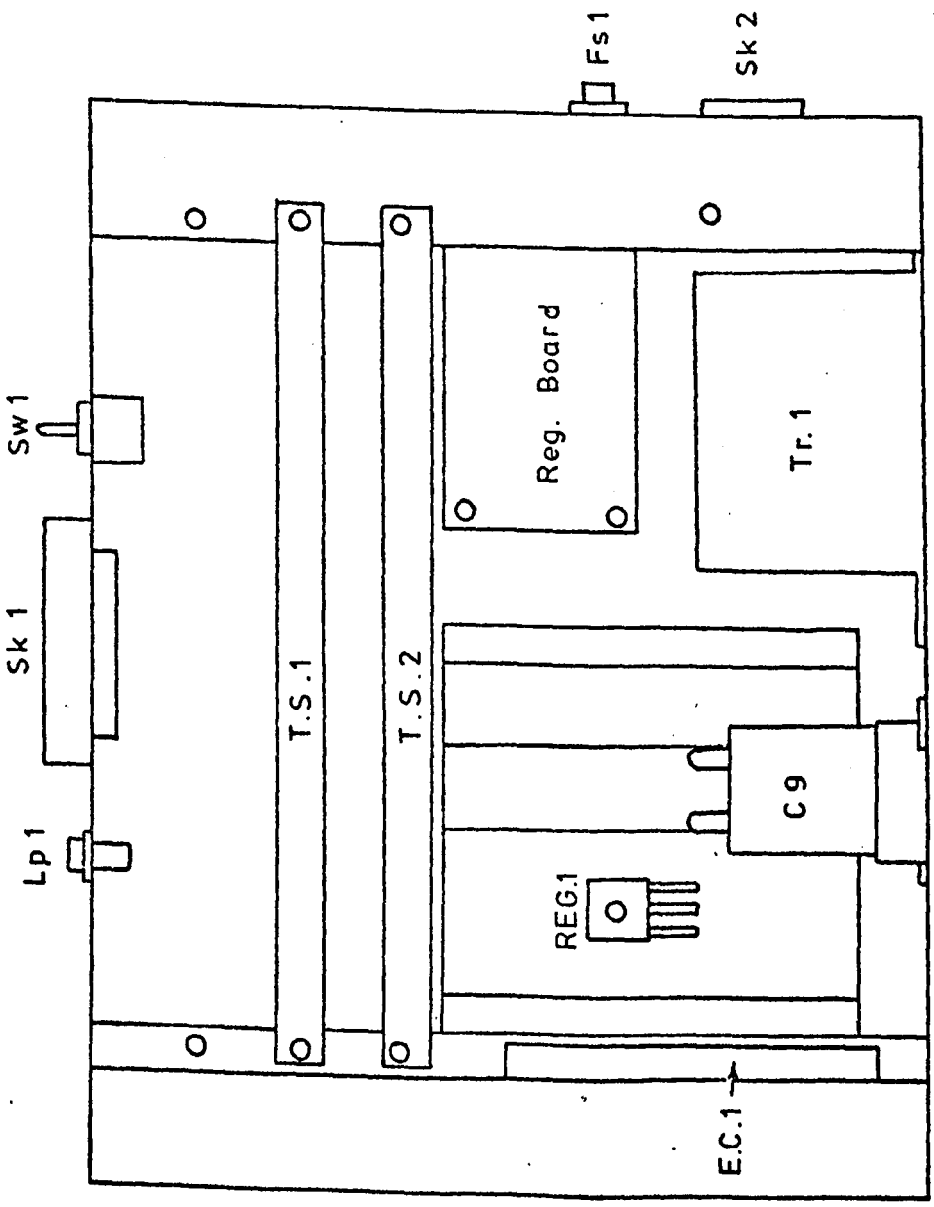


Figure 1.22 Case - internal layout.

1.2 User Guide

1.2.1 Introduction

The Gipop (general input-output) link is basically a sixteen-bit bi-directional driver-receiver unit attached to the Gipop highway of the FM1600B computer. The Gipop unit is a peripheral of the FM1600B and is well documented elsewhere [1]. The link and associated programs are designed to make the Gipop unit easy to use.

The Gipop link is intended primarily for use as an interface between the FM1600B and the F100 microcomputer system, but may be easily adapted for general use by the provision of a suitable control program, and some minor changes to the subroutines described in this document.

In addition to the sixteen-bit bi-directional data highway, three latched outputs are supplied for control purposes and a handshake pair included to provide a comprehensive interface facility.

The circuit has been designed in such a way that the control word may be latched from the Gipop highway before the first request is issued via the handshake lines. In this manner, the control word is stable at the interface before any data transfers are initiated.

1.2.2 Electrical Specifications

All inputs and outputs to and from the link are TTL compatible. The use of open-collector drivers is recommended to drive the highway between the user peripheral and the link. The data highway is pulled-up by the link (via 330Ω to 5v) and so no pull-up resistors need be provided by the user.

The control lines are driven by open-collector gates and should be pulled-up by the user via a suitable circuit (see Technical Description). Pull-up resistors are provided on the link board but are not connected to the Vcc supply. They may be connected if desired but their use is not

recommended [2].

The operating speed is limited to half the maximum scan rate available for the Gipop unit and the settling time required by the Gipop link circuits. This results in a maximum transfer rate of ≈70Hz or one transfer per 15 ms.

1.2.3 Gipop Set-Up

The following is a brief description of the required configuration of the Gipop unit when used with the link.

The data area for the link (S8090) is arranged in the following order:-

0	Test word
1	Data input word
2	Data output word
3	Relay control word
4	Switch input word

Performing the scan in this order ensures that the data input and output words are stable before any peripheral activity is instigated via the relay outputs. The switch input word is included for the sake of completeness and to accommodate any future developments.

The order of the boards in the Gipop half-shelf module of the FM1600B should be as given in Table 1.2.1.

Two switches are included on the back of the Gipop half-shelf (in the back of the FM1600B), one a rotary switch (FIN WD) and the other a toggle (SCAN). The FIN WD switch should be set to the 3 position (4 if switch input required) and the SCAN switch to the INT position.

Numerous jump leads are required on the Gipop patch board (at side of FM1600B); the complete set-up for this is as shown in Figure 1.2.1.

1.2.4 Program Requirements

A complete set of subroutines has been written to operate the Gipop link. The user need only write a control program and a set-up routine for the peripheral attached to the link. For examples of these programs and complete explanations of the operating routines the reader is referred to Ref. 2 .

The basic set-up requirements of the operating routines is as follows:-

V2 = code word (i.e. control word output)

V11 = peripheral address

V12 = FM1600B start address

V14 = number of transfers

Note, at the end of a transfer sequence these registers are returned to the user in an uncorrupted state. However, during the excution of the transfers V12 is incremented from its initial starting point and V14 is decremented to zero. Once at zero, the registers are restored and control returned to the user program. The programs are designed for block transfer operation and if the FM1600B address is required to remain constant then block transfers of length one must be performed (or the read-write routine (S8088) modified).

The following subroutines are available to the user:-

S8080 - Set-up routine

No. of words - 50

Entry point - 0

Registers destroyed - none

Subroutines called - S8084, S8085, S8090

This routine performs all the necessary set-up operations required for the use of the link with the F100 system.

S8084 - Initiate scan and wait for interrupt

No. of words - 28

Entry point - 0

Registers destroyed - V10, V19, V22

Subroutines called - S8089, S555, S12

This routine initiates the Gipop scan and loops on a flag in V10, waiting for an interrupt.

S8085 - Delay routine

No. of words - 6

Entry point - 0

Registers destroyed - none

Subroutines called - none

This routine generates a 10ms (approx.) delay by means of a count-down loop, necessary to allow the Gipop relay outputs to settle before the next operation is started.

S8088 - Read-write routine

No. of words - 45

Entry point - 0

Registers destroyed - V10

Subroutines called - S8084, S8085, S8090

This routine performs the data transfer portion of the scan.

S8089 - Interrupt routine

No. of words - 55

Entry point - 0

Registers destroyed - V10, V22

Subroutines called - S8090, S555, S12

This routine checks for valid interrupts and that the Gipop test word has been transferred correctly.

S8090 - Gipop data area

No. of words - 5

Entry point - N.A.

Registers destroyed - none

Subroutines called - none

This is a five word data area used by the Gipop unit during data transfers. The area is arranged as follows:-

Word 0 - Test word

1 - Data input word

2 - Data output word

3 - Relay control word

4 - Switch input word

1.2.5 Mechanical Details

The Gipop link is contained in a small aluminium box attached to the laboratory wall in the level 4 laboratory of the Electrical and Electronic Engineering Department, The City University, London.

The link is provided with a self-contained power supply in which 0V and GND are isolated. A switch and indicator are included with the supply.

Connection to the link is by means of a 25-way Cannon connector mounted on the top of the case. The connections for this socket are given in the Technical Description, section 1.1.7.1.

Position	Board No.	Function
1	189	Control board
2	020	Control board
3	845	Control board
4	079	Voltage input interface
5	079	Voltage input interface
6	023	Voltage output interface
7	022	Relay output interface
8 - 12	-	Not used

Table 1.2.1 Board order in the Gipop half-shelf module.

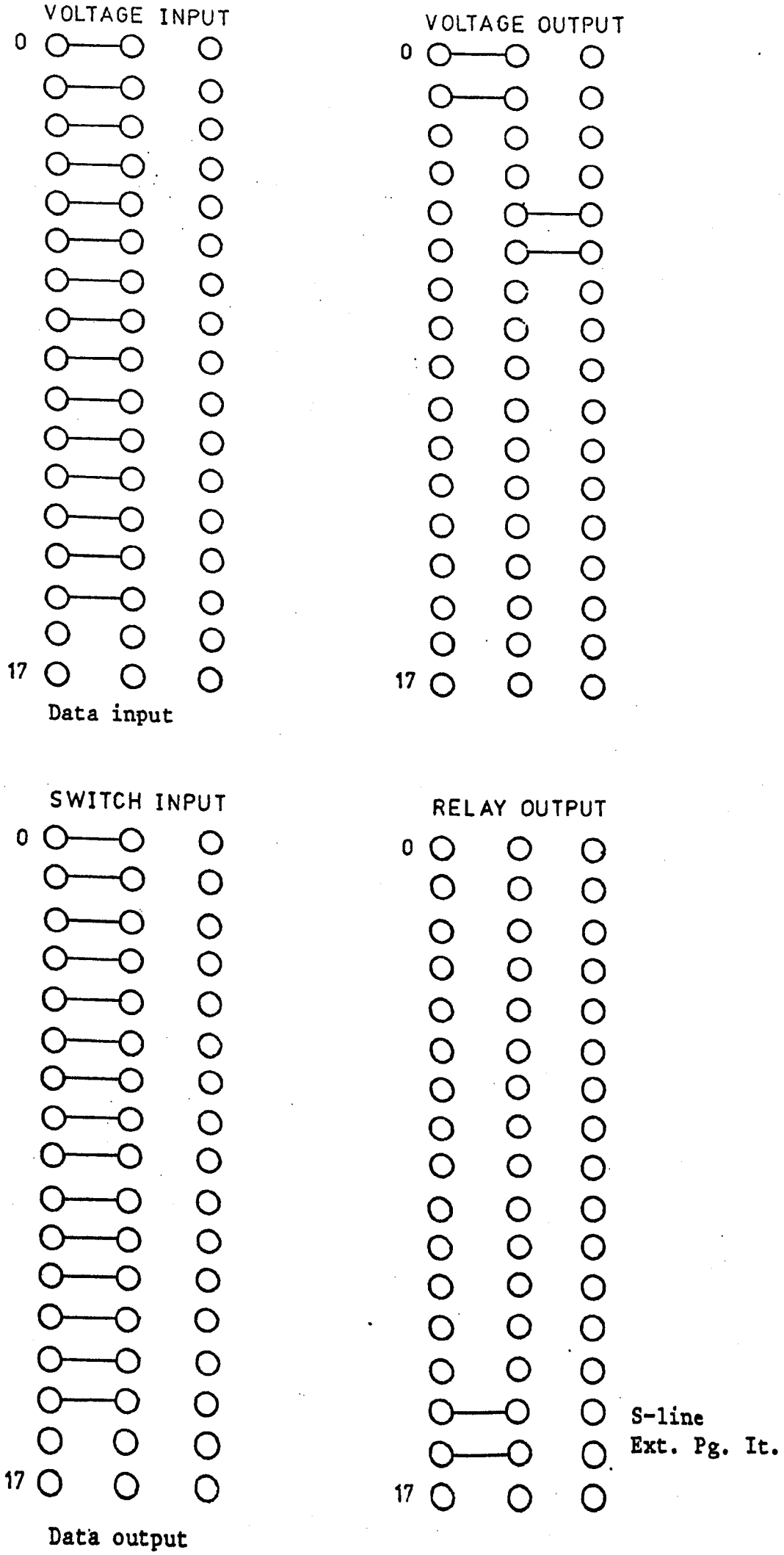


Figure 1.2.1 Gipop patch board connections.

1.2.6 Use of the link in the FM1600B-F100 interface

This section explains the use of the control program and set-up requirements of the Gipop link when used with the F100M microcomputer system as a peripheral.

The link is connected to the F100M via the Gipop buffer board [3][4] which is in turn connected to channel 1 of the F100M's input-output controller (IOC). The peripheral address is entered into a register in the IOC as the start address of a block of F100M store. Note, a pre-increment is performed on this register and so the user must set V11 to the start address minus 1. Before the address register may be loaded it must first be accessed via the F100M's front panel, to initialize the circuits from their start-up settings. To do this the user must access location 32763 (all handswitches set except 2 and 15); the data loaded is irrelevant. The F100M and link are now ready for data transfers.

The control program requests the various parameters from the user via the teletype by a series of prompts. The first of these is 'Read or Write' to which the user replies:-

R - Read

W - Write

C - Continue (with the settings from the previous run)

E - End (jump to Stop 12).

The 'C' command jumps the control program straight to the 'Ready' state, by-passing the rest of the set-up procedure. This can be extremely useful for test purposes.

If any input, other than the above four, is entered, 'Error' is output on the teletype and the 'Read or Write' prompt repeated.

Note, all characters or character strings are terminated with a space.

Next, the 'M Address', 'B Address' and 'Length' are

requested and the corresponding values are entered in decimal.

'Ready' is now output, to which the user may reply:-

G - Go

E - Error. Return to 'Read or write' prompt to re-enter parameters.

If neither of the above characters are entered the 'Ready' prompt is repeated.

When 'G' is entered the requested operation is performed and when completed, 'Done' is output and the 'Read or Write' prompt re-issued.

1.2.7 References

1. Gipop Manuals, Ferranti Ltd., Oct. '71 + updates.
2. Gipop link - Technical Description, Internal report, R. A. Comley, July '77.
3. Gipop Buffer - Technical Description, Internal report, R. A. Comley, July '77.
4. Gipop Buffer - User Guide, Internal report, R. A. Comley, July '77.

Chapter 2

Gipop Buffer

1. Technical Description

2. User Guide

2.1 Technical Description

2.1.1 Introduction

The Gipop (general input-output) buffer is a general purpose data channel for the F100M microcomputer system. It is basically a sixteen-bit bi-directional input-output buffer designed, primarily, for use in the FM1600B and F100M interface, but may be of general use whenever a slow peripheral must be interfaced to the F100M system (see Fig. 2.1).

The sixteen input-output lines are latched on the buffer board which relieves the user of the timing restrictions imposed by the F100M. The user may select the handshake response time by the use of two monostable circuits. At present, it is set to 2ms (approx.) for use with the Gipop link and FM1600B.

The buffer is connected into channel 1 of the input-output controller (IOC) of the F100M system and all data transfers take place under direct memory access (DMA) control. For this purpose, three control lines ($\overline{\text{Dir}}$, $\overline{\text{Q}}$, In Sr Cc) and a peripheral request-accept handshake pair are provided. The three control inputs are enabled via a fourth input ($\overline{\text{Enable}}$) which is active low (0v). This ensures that when not in use (i.e. Cannon plug disconnected) the board is disabled.

All input-output between the Gipop buffer and peripheral devices is via a 25-way twisted-pair cable and a 25-way Cannon connector.

2.1.2 Design Considerations

The fundamental design consideration of the Gipop buffer was to provide an asynchronous interface between the F100M and a peripheral, and in particular for the interface between the F100M and the FM1600B. The F100M, however, is constrained by a 'time-out' on peripheral transfers (50μs max.) and hence the need for the buffer board. The Gipop

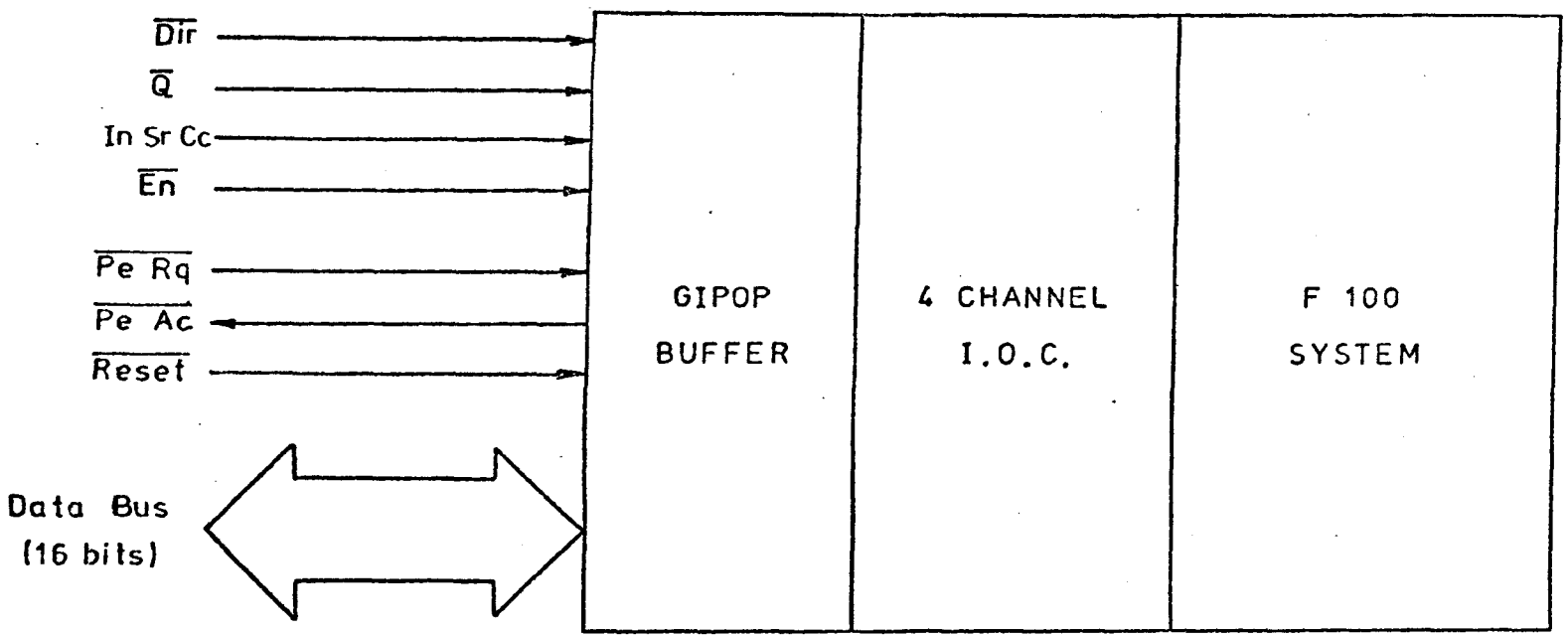


Figure 2.1 Block diagram of Gipop buffer system.

buffer permits asynchronous transfers at both the F100M and peripheral interfaces.

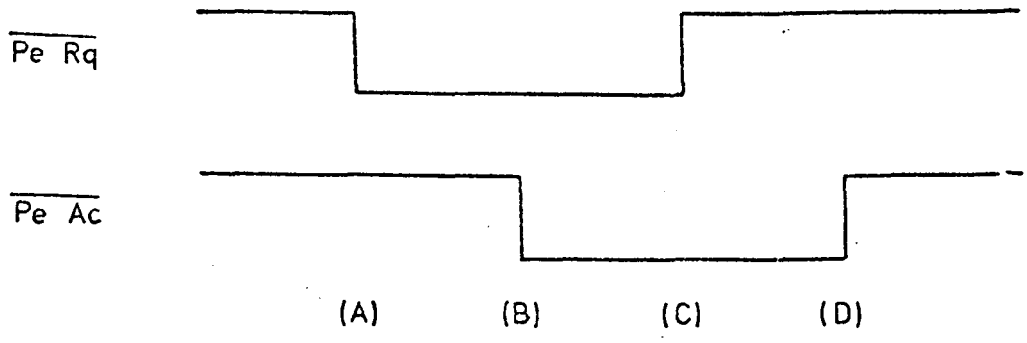
The basic form of the handshake at either interface is as shown in Figure 2.2, but the time scale for a peripheral and the F100M will generally be very different. The complete handshake sequence for both the F100M and a peripheral will be as shown in Figure 2.3.

For a read cycle, the attention request ($\overline{\text{AtRq}}$) (or peripheral request ($\overline{\text{PeRq}}$) from the peripheral) causes an immediate peripheral request to the F100M. Upon completion of the F100M read cycle (i.e. $\overline{\text{PeAc}}$ returned high) the requested data are available and, after some time interval T_1, attention accept ($\overline{\text{AtAc}}$) is issued. The time interval T_1 is governed by the response time of the peripheral and set by a monostable. After attention accept has been issued, the peripheral then completes its handshake sequence with the buffer board.

For a write cycle, the rising edge of $\overline{\text{AtRq}}$ must be used to issue $\overline{\text{PeRq}}$ to the F100M to ensure that the data to be written are available at the start of the F100M cycle. Again, a time delay, T_2, should be included to ensure that $\overline{\text{AtAc}}$ is not cleared too quickly after $\overline{\text{AtRq}}$ has been reset.

From an inspection of Figures 2.2 and 2.3 it is apparent that some form of latch is required to hold the input-output data during transfers, as in both the read and write cycles the transfers to or from the F100M take place with the data bus in some indeterminate state. To overcome this problem, a bi-directional latching system was included in the bi-directional driving circuitry (Figs. 2.14 - 2.17).

The complete control circuit to decode and generate the necessary timing signals is given in Figure 2.13.



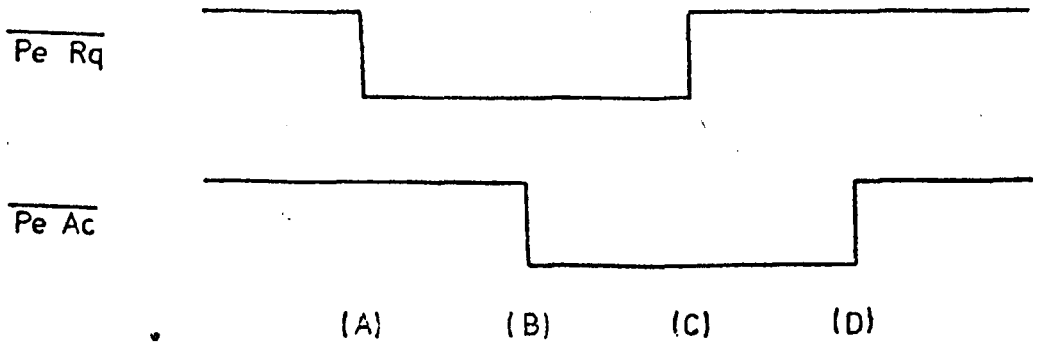
(A) Transfer request

(B) Transfer accept : data on bus

(C) Data accepted

(D) Data removed : end of cycle

(a) Read cycle.



(A) Transfer request : data on bus

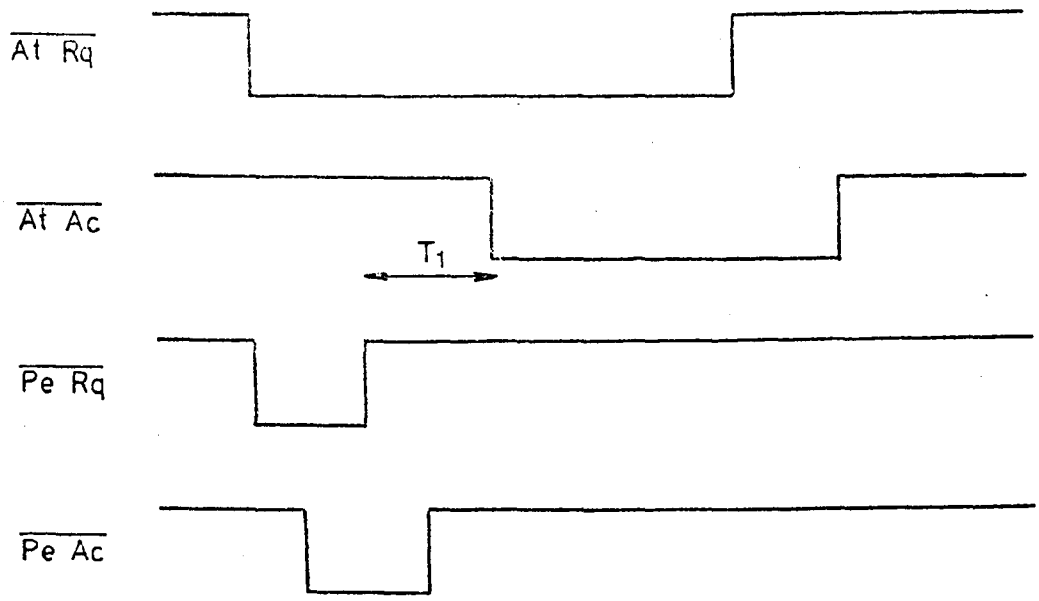
(B) Transfer accept : data accepted

(C) Data removed

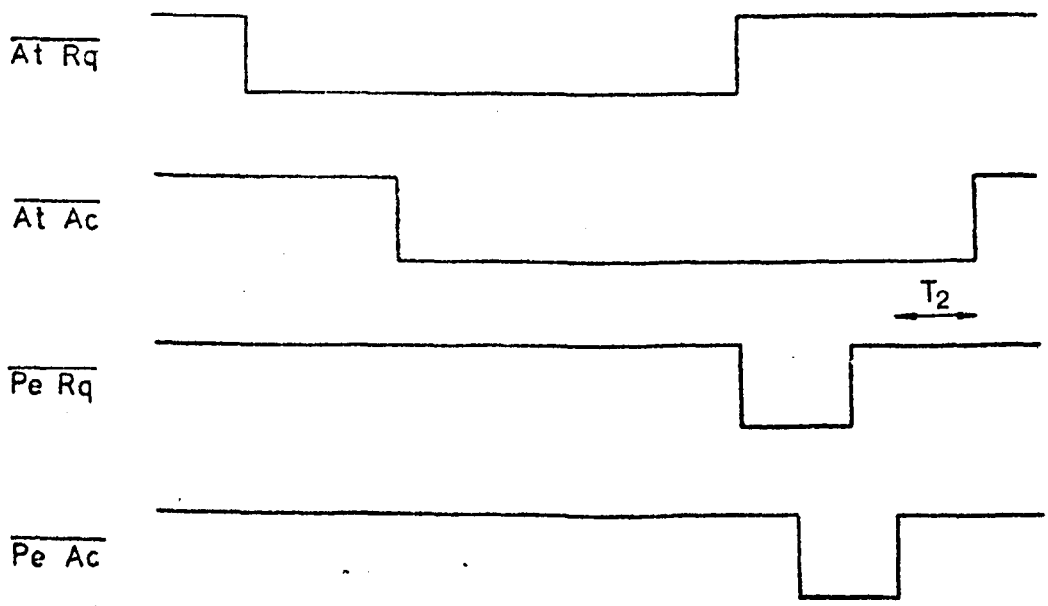
(D) End of cycle

(b) Write cycle.

Figure 2.2 Handshake timing waveforms.



(a) Read cycle.



(b) Write cycle.

Figure 2.3 Peripheral and F100M relative timing waveforms.

2.1.3 Circuit Design

2.1.3.1 Control inputs

Three control inputs are required to specify the type of DMA operation required (see Table 2.1) and these are received and buffered as shown in Figure 2.4. An enable input is also provided so that the buffer board is effectively disabled when unplugged from the peripheral.

2.1.3.2 Read-write control

This is decoded from two of the three control inputs and is used to control data direction and set the handshake sequence within the Gipop buffer circuits. The circuit is as shown in Figure 2.5.

2.1.3.3 Peripheral request circuit

The route taken by attention request to D-type flip-flop 2 and thereby to the peripheral request output is determined by the state of the read-write input applied to NAND 1. This input is low for a read operation and high for a write. Hence, for a read operation $\overline{\text{AtRq}}$ goes via flip-flop 1 to clock flip-flop 2 and set $\overline{\text{PeRq}}$ on the first edge (trailing) of $\overline{\text{AtRq}}$. For a write operation, $\overline{\text{AtRq}}$ goes via NAND gates 1 and 2 but no signal reaches flip-flop 2 until $\overline{\text{AtRq}}$ returns to the high state (see Figs. 2.6 and 2.7).

Flip-flop 1 is cleared at the end of each transfer by the clear 1 signal (see later) and flip-flop 2 by the peripheral accept signal. The main reset input is also taken to this input to ensure that $\overline{\text{PeRq}}$ is cleared when the F100M is reset.

2.1.3.4 Peripheral accept

To ensure that the peripheral request signal is not

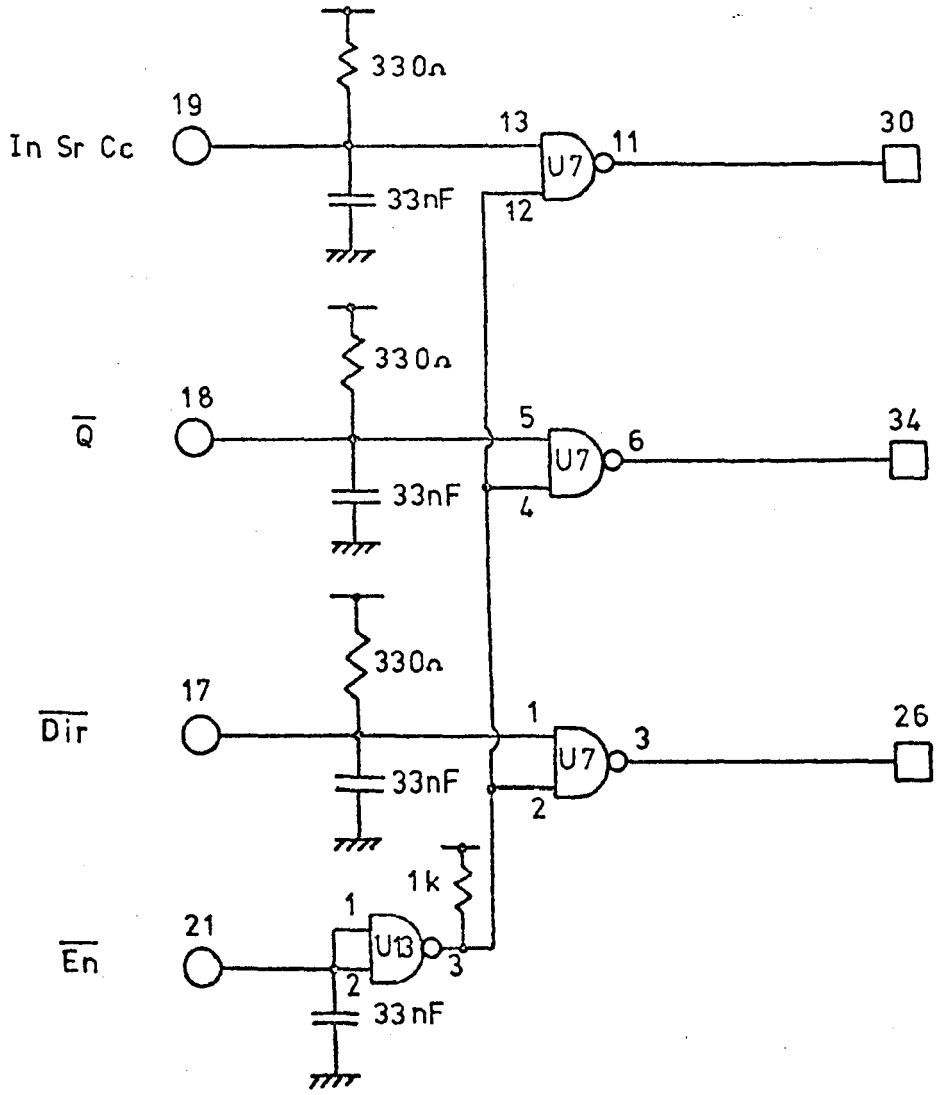


Figure 2.4 Control inputs.

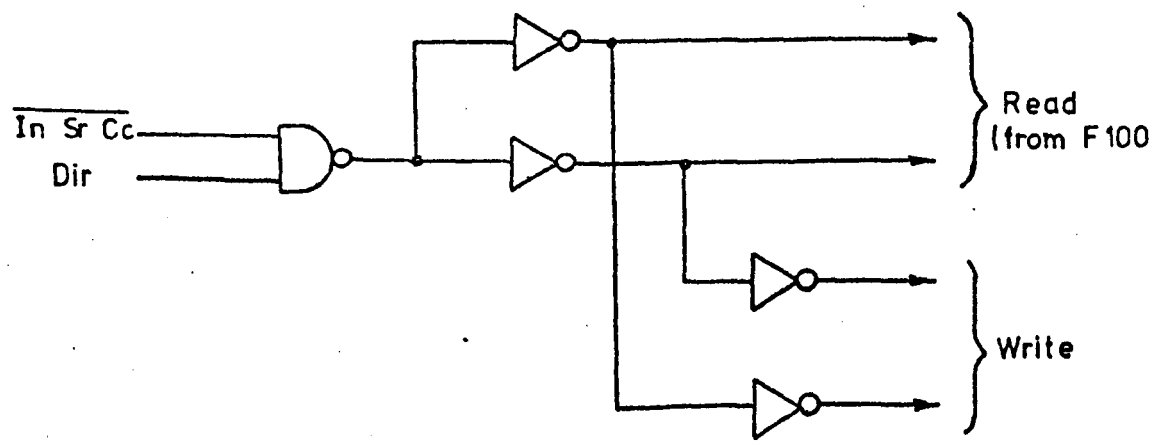


Figure 2.5 Read-write control.

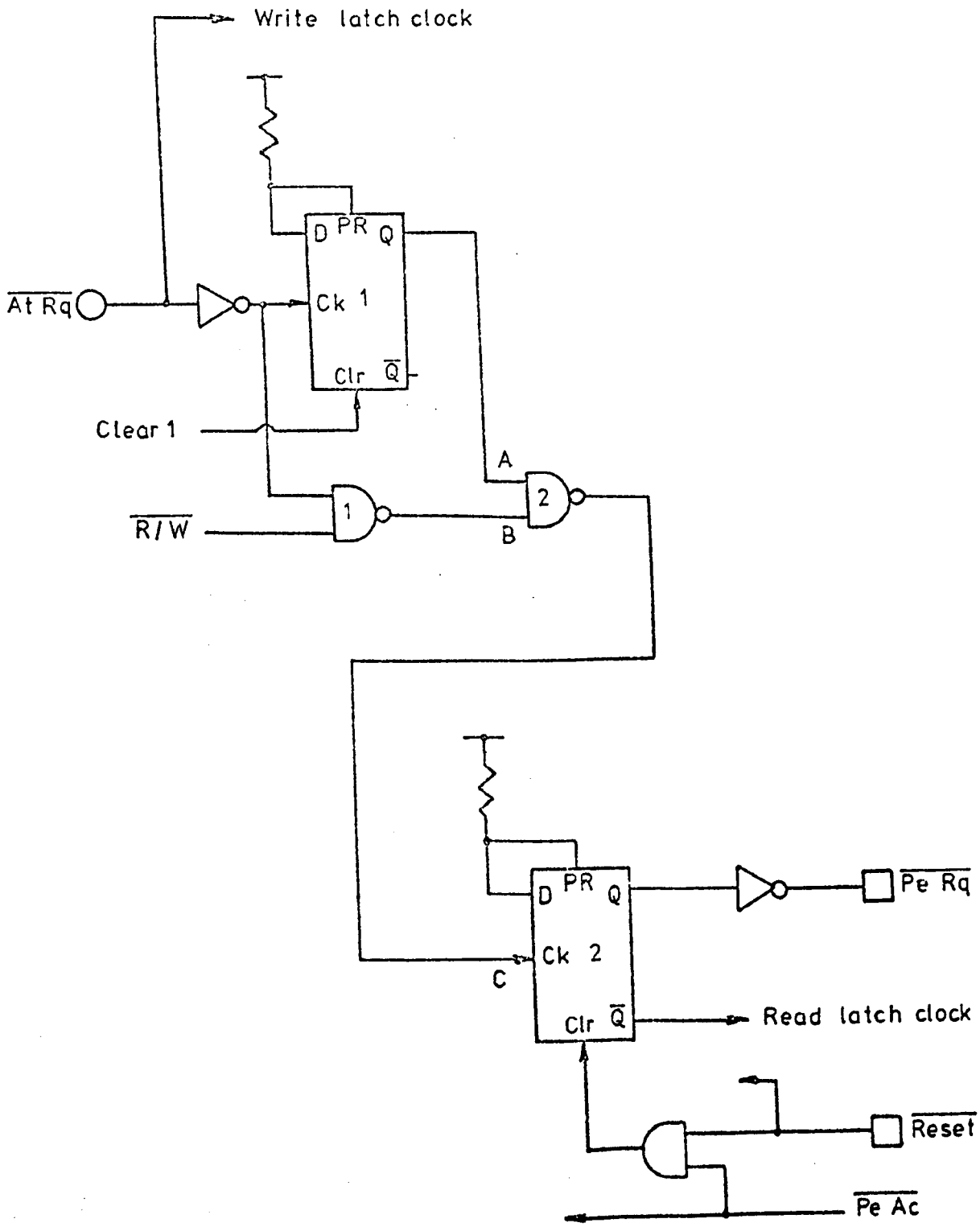
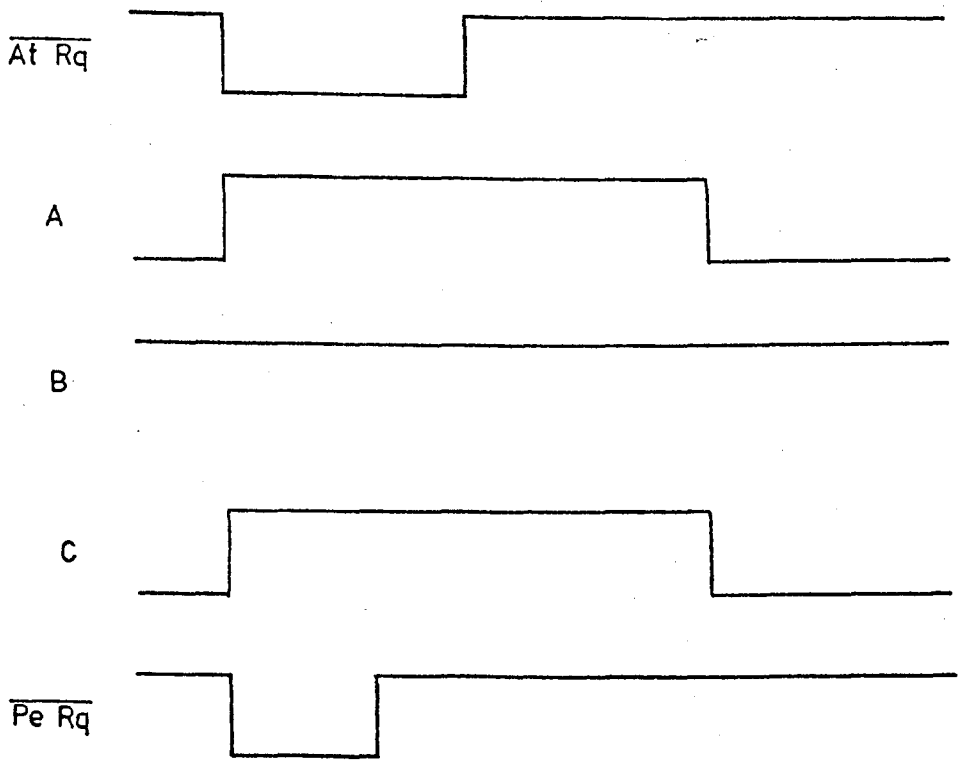
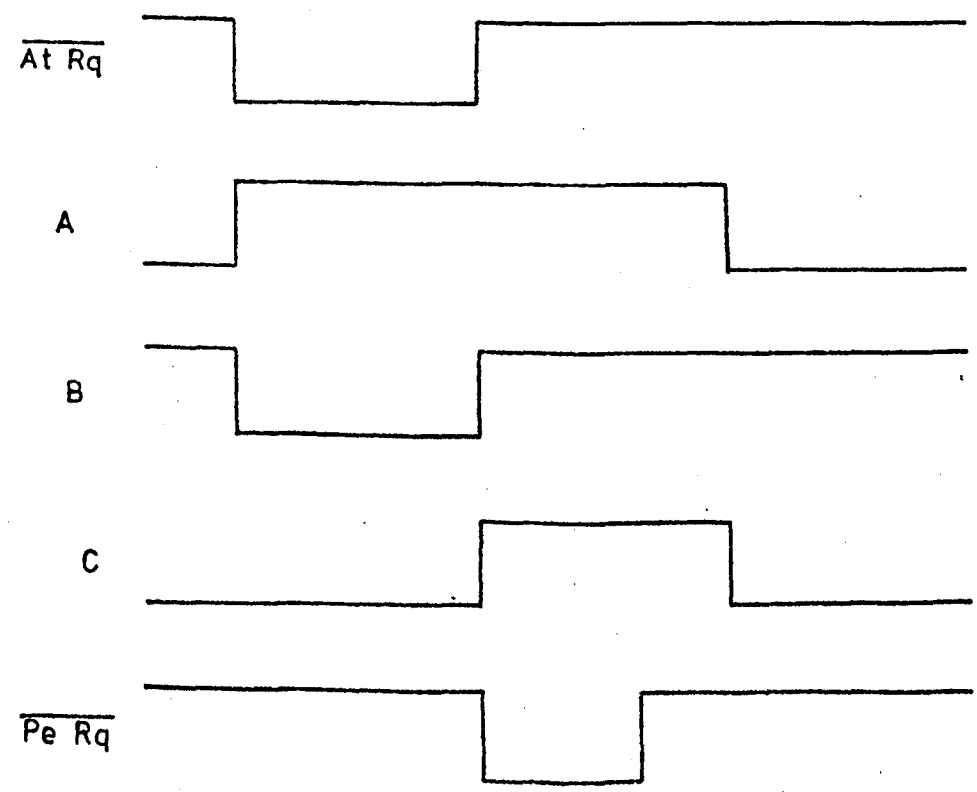


Figure 2.6 Peripheral request circuit.



(a) Read cycle.



(b) Write cycle.

Figure 2.7 Timing waveforms for peripheral request circuit.

cleared too quickly upon receipt of the peripheral accept signal, a monostable is included. This delays the $\overline{\text{PeAc}}$ signal by 0.2μs (approx.) (Fig. 2.8).

2.1.3.5 Attention accept circuit

The mode of operation of this circuit is determined by the state of the read-write input. For a read operation the $\overline{\text{PeAc}}$ input causes a trigger signal to be generated which is applied to a monostable input (Figs. 2.9 and 2.10). This is set to give a delay of 2ms (approx.) after which time a D-type flip-flop is clocked to generate the attention accept signal. For a write operation the trailing edge of the $\overline{\text{AtRq}}$ input causes the trigger pulse to the monostable to be generated, thereby issuing attention accept before peripheral request is issued (or peripheral accept received).

Attention accept is reset via the clear 1 input to the flip-flop.

2.1.3.6 Clear 1 circuit

This circuit is used to reset the attention request and accept flip-flops and its mode of operation is also controlled by the state of the read-write circuit. For a read cycle, the $\overline{\text{AtRq}}$ input generates the trigger input to a monostable set to give 2ms (approx.) delay. The output from this monostable then causes the clear 1 signal to be generated (see Figs. 2.11 and 2.12). For a write cycle, the $\overline{\text{PeAc}}$ input causes the clear 1 signal to be generated. This ensures that clear 1 is generated at the correct time for both read and write cycles. The external reset (from the F100M) is also gated into the clear 1 circuit to ensure that all flip-flops are reset when the processor is reset.

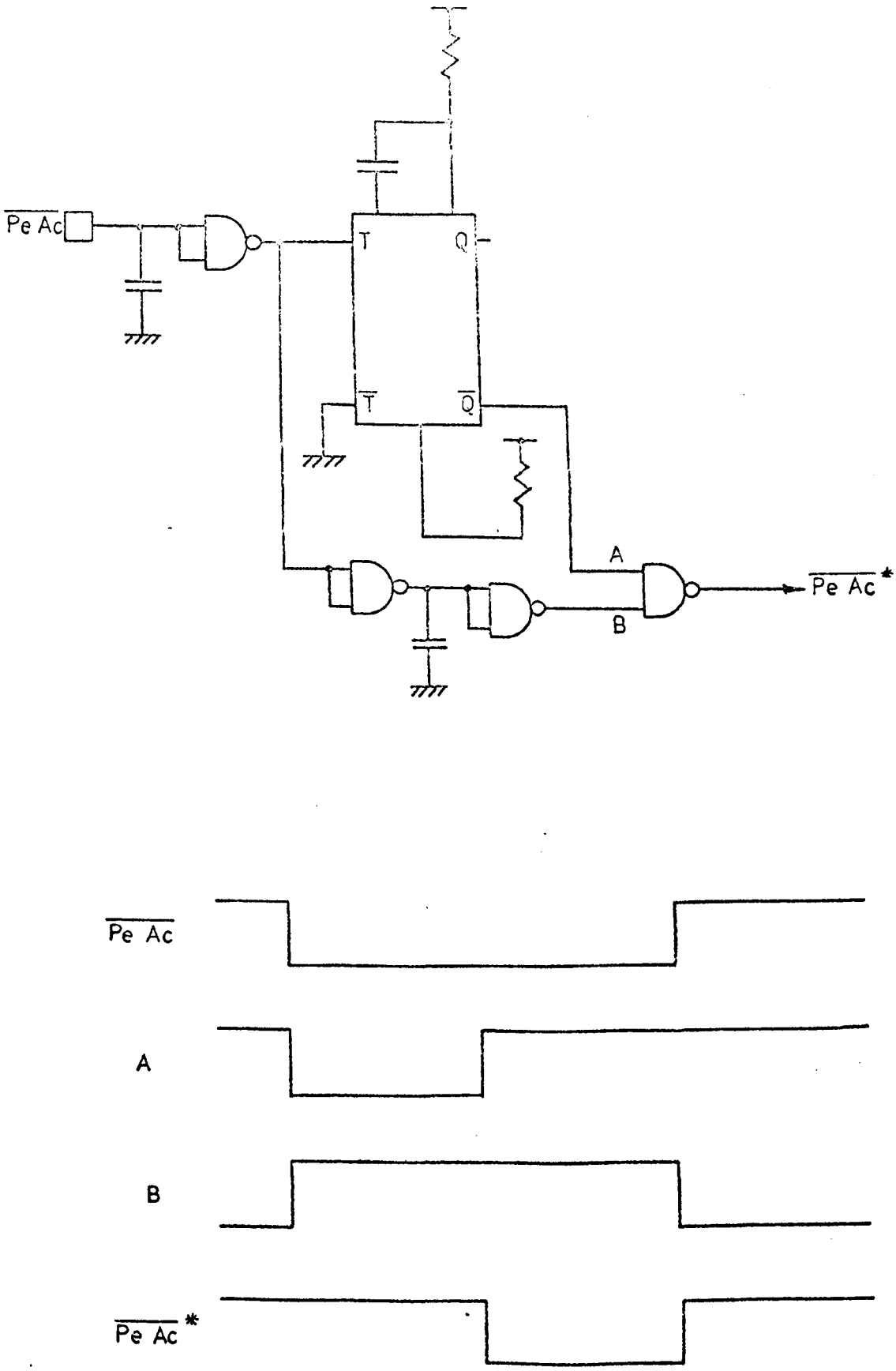


Figure 2.8 Peripheral accept delay circuit and timing waveforms.

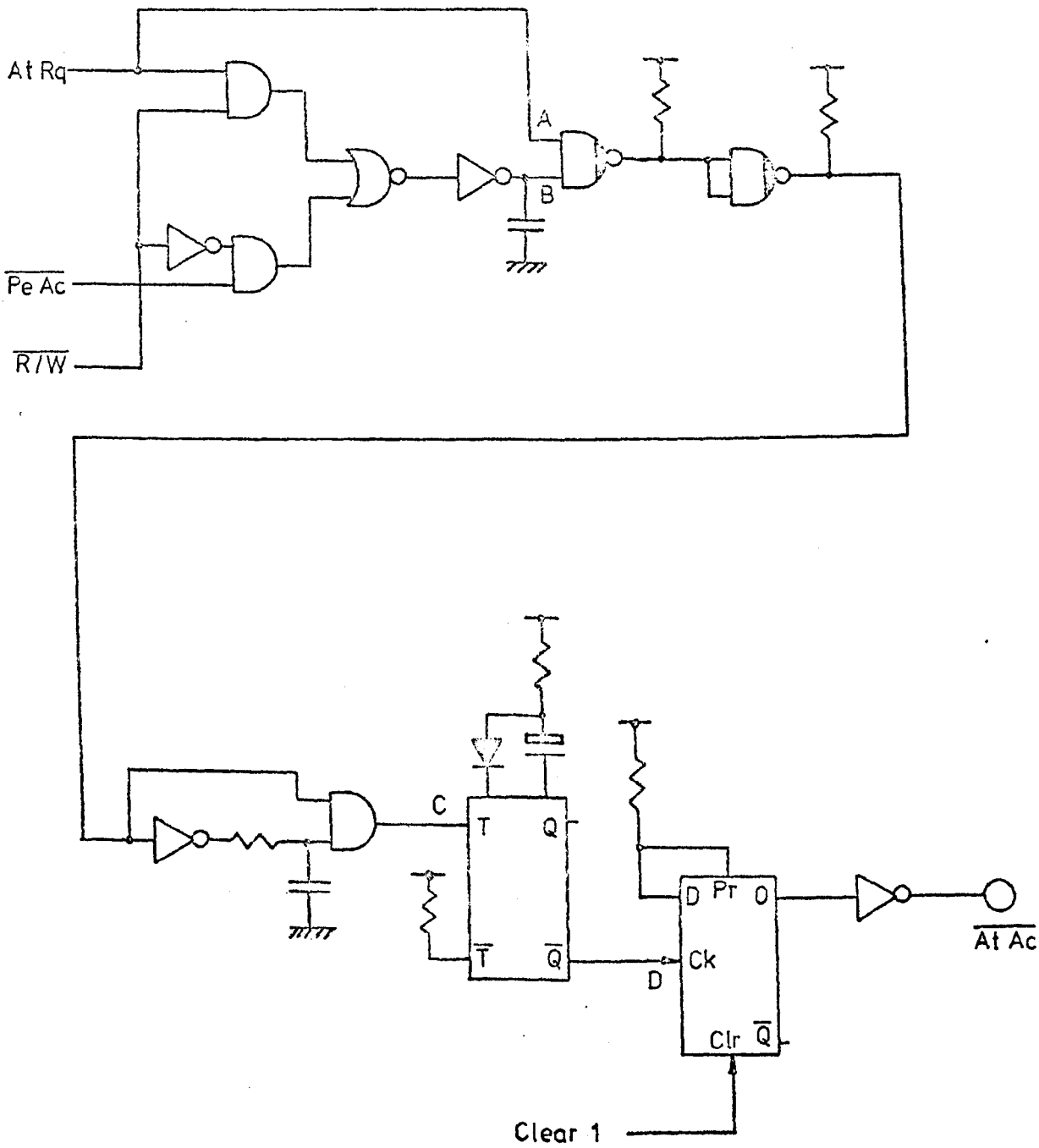
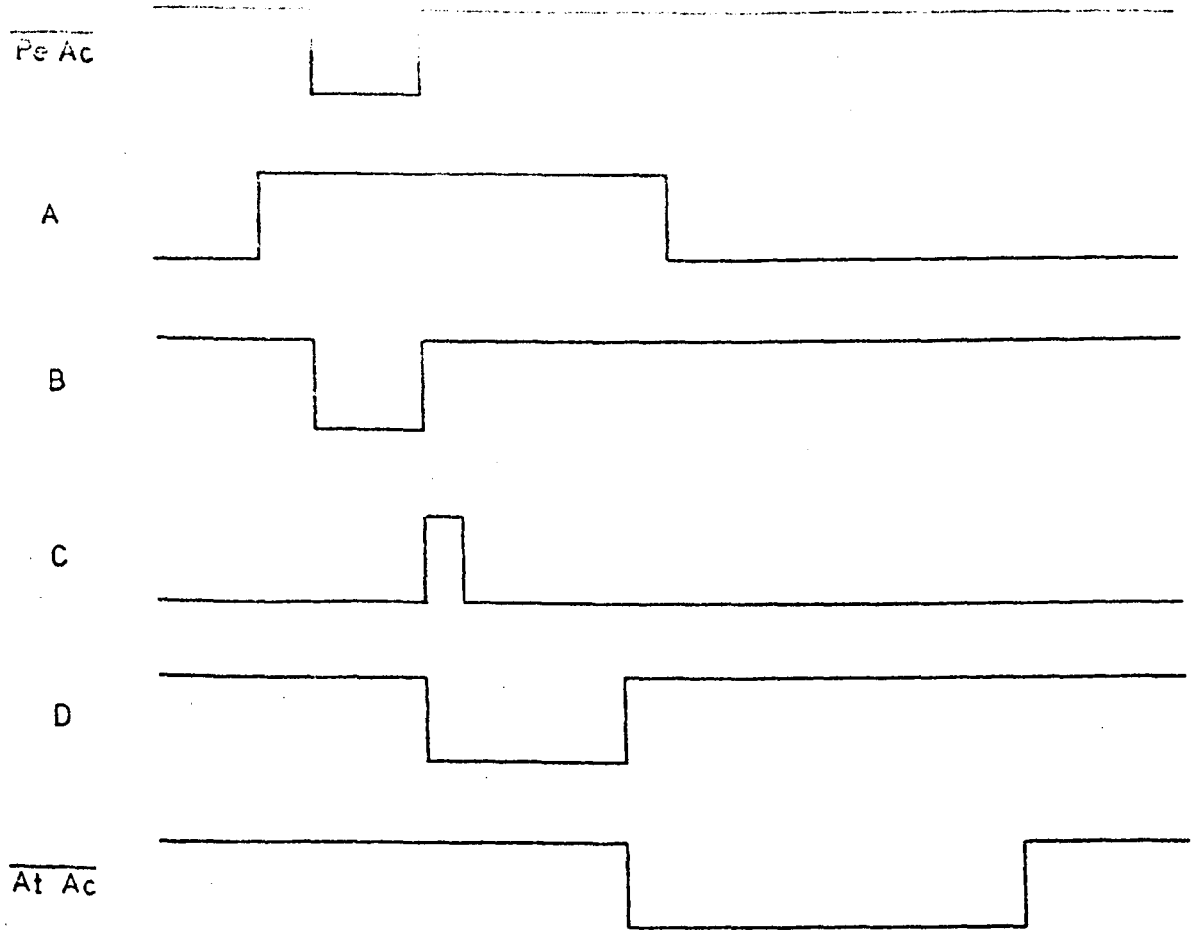
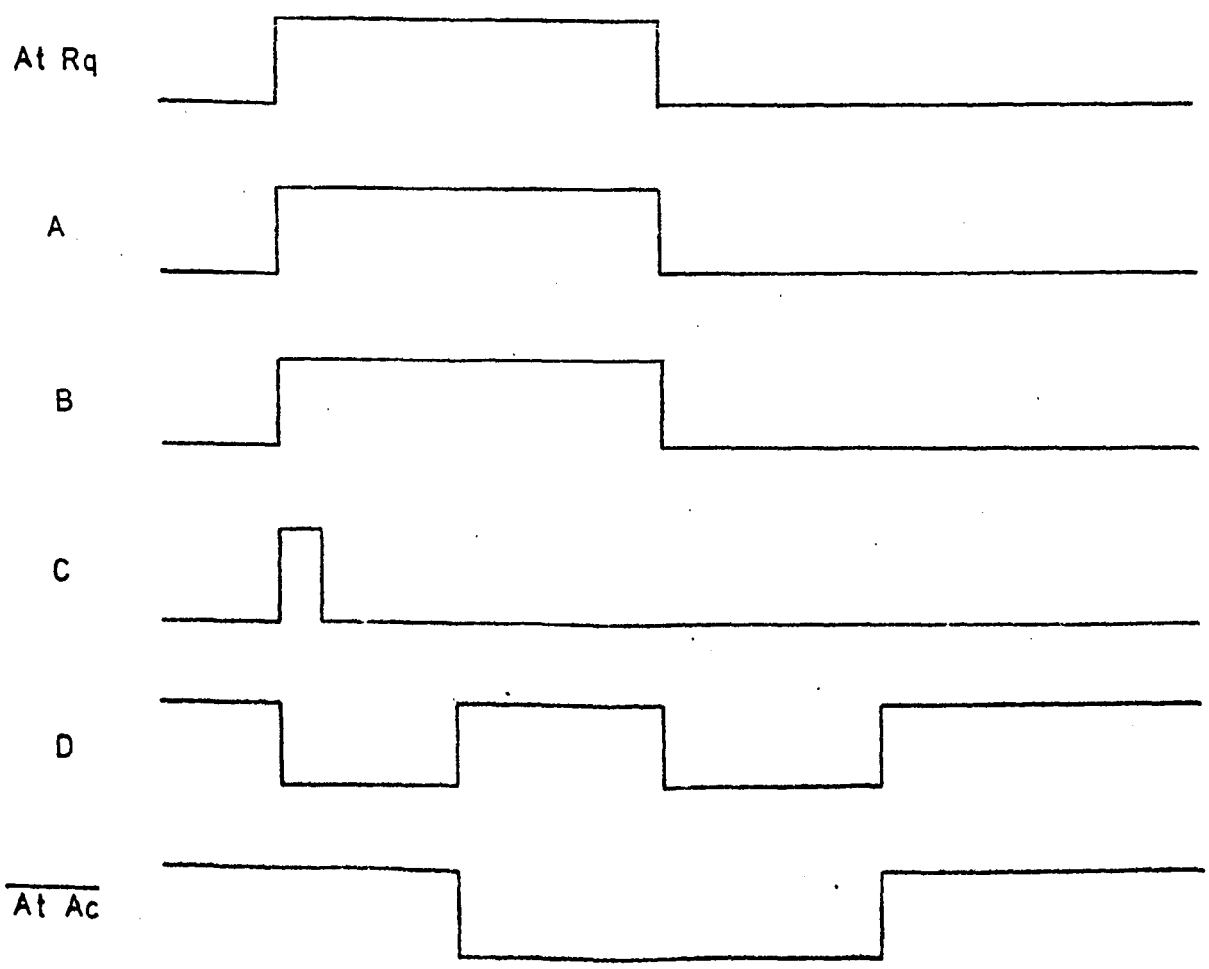


Figure 2.9 Attention accept circuit.



a) Read cycle.



b) Write cycle

Figure 2.10 Timing waveforms for attention accept circuit.

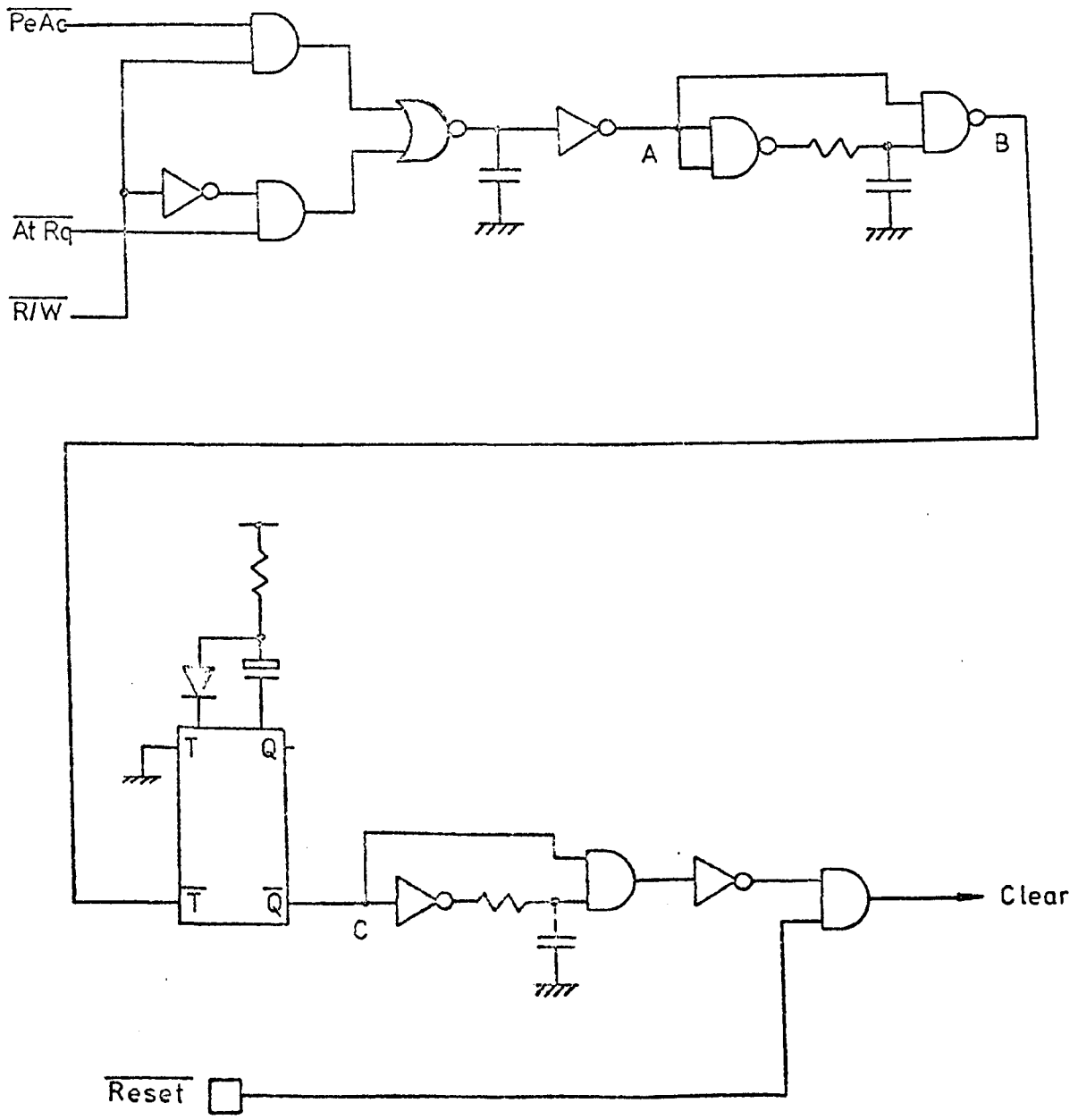
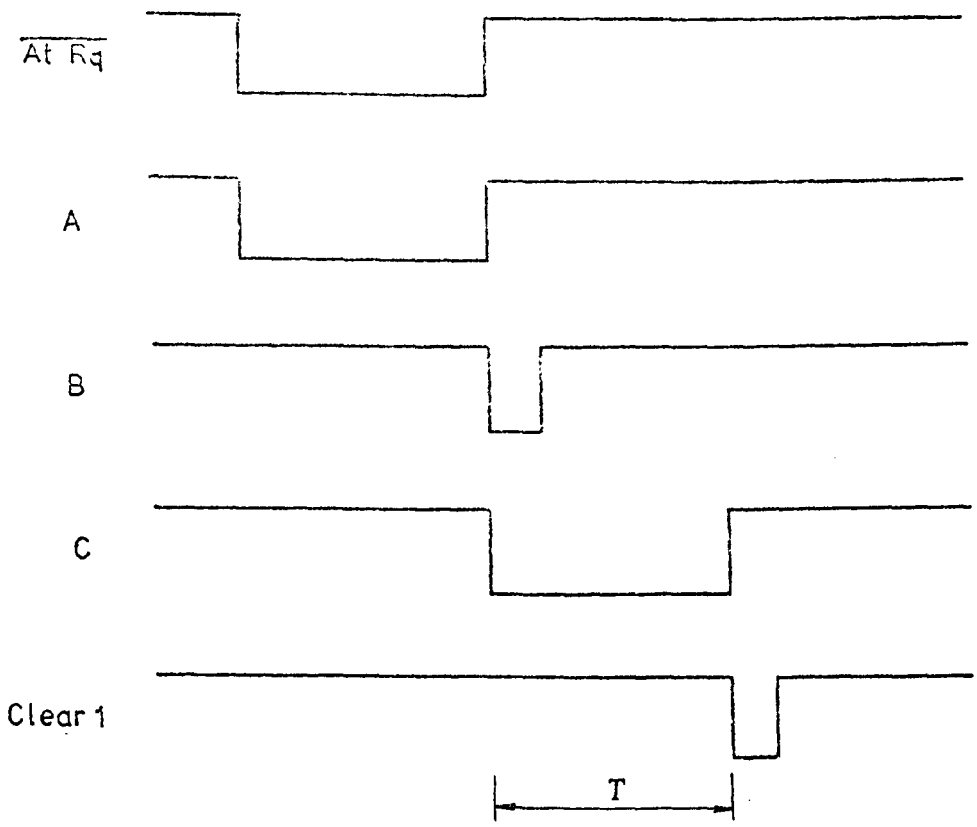
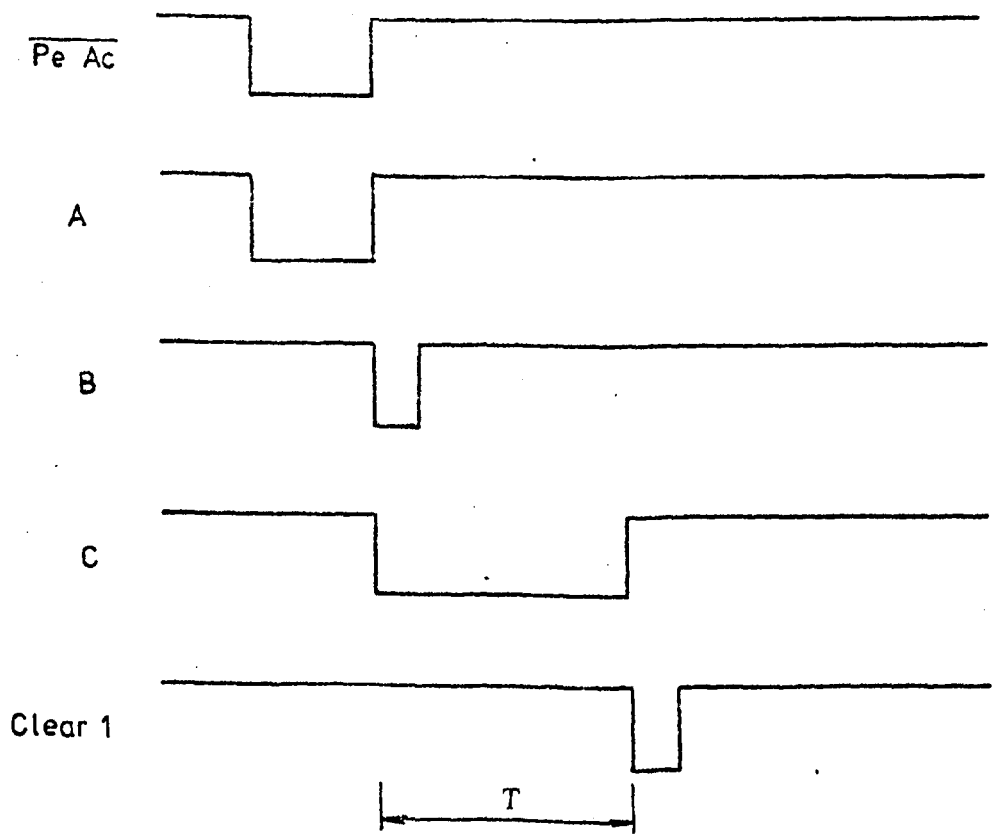


Figure 2.11 Clear 1 circuit.



(a) Read cycle.



(b) Write cycle.

Figure 2.12 Timing waveforms for clear 1 circuit.

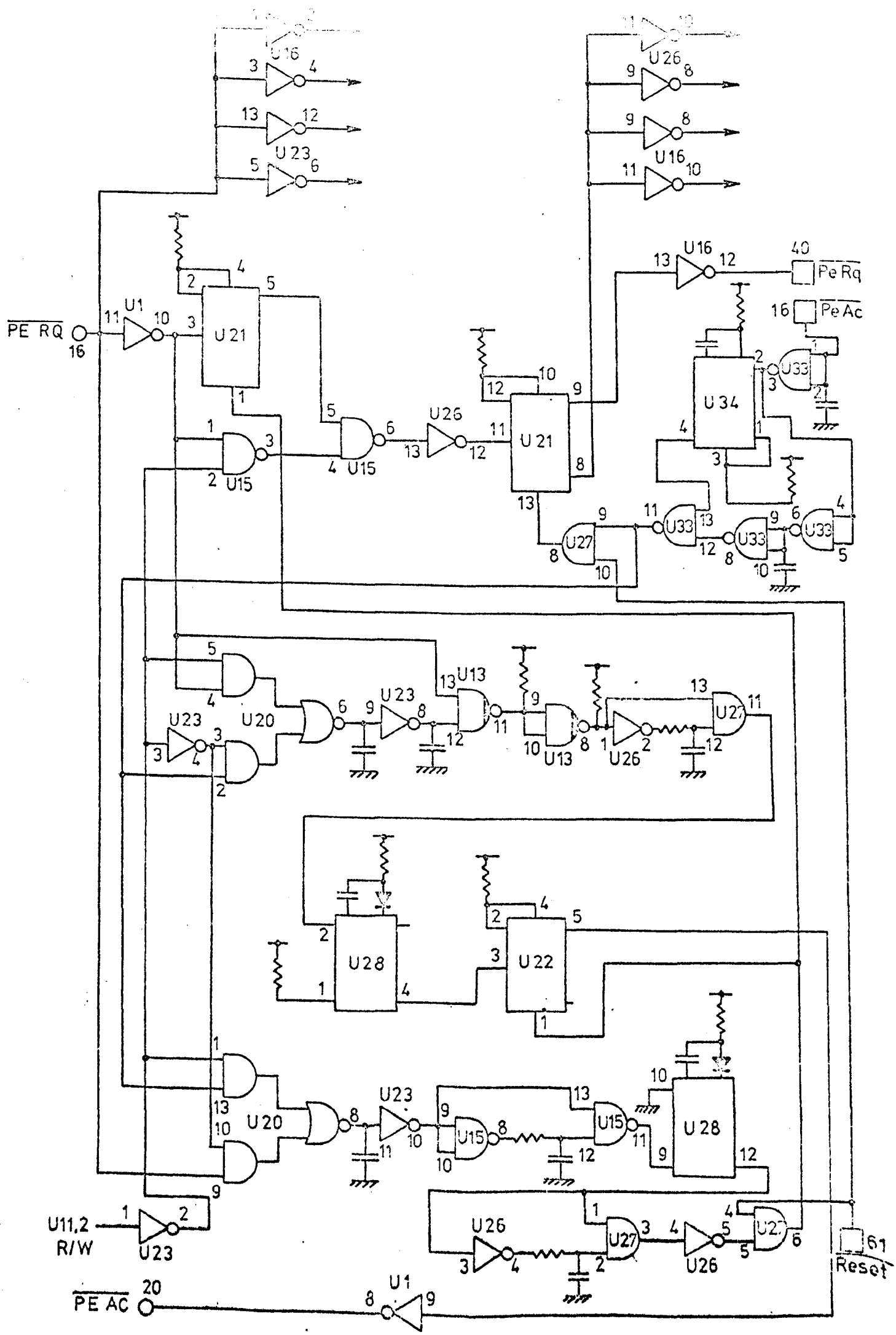


Figure 2.13 Complete buffer control circuit.

2.1.3.7 Latch circuits

Four, eight-bit latch circuits are used in the bi-directional highway, for temporary storage of data. These are divided into two, sixteen-bit groups, one for read operations and one for write. The clock input to the read latches is taken from the peripheral request flip-flop and the clock for the write latches is taken directly from the attention request input.

Both sets of latches are clocked during every operation, irrespective of the direction of data transfer.

2.1.3.8 Bi-directional highway drivers

The sixteen-bit bi-directional highway is driven by open-collector gates in both directions. The gates are not pulled-up, but a set of pull-up resistors is provided on the peripheral side of the highway, should they be required. When not in use, the V_{CC} supply should be disconnected. Latches are also included in the highway for temporary storage of data (see Figs. 2.14 - 2.17).

2.1.3.9 Power supply considerations

The Gipop buffer requires a 5v @ 1A (approx.) supply which it draws from the F100M power supplies. The supply is decoupled to 0v via five 0.1μF ceramic capacitors, distributed over the board.

2.1.3.10 Other considerations

The program controlled handshake lines ($\overline{JPe}$ and $\overline{KPe}$ [4]) are linked on the board as the processor requires their use during the set-up proceedure (see User Guide [1]). This requires that the handshake paths be completed.

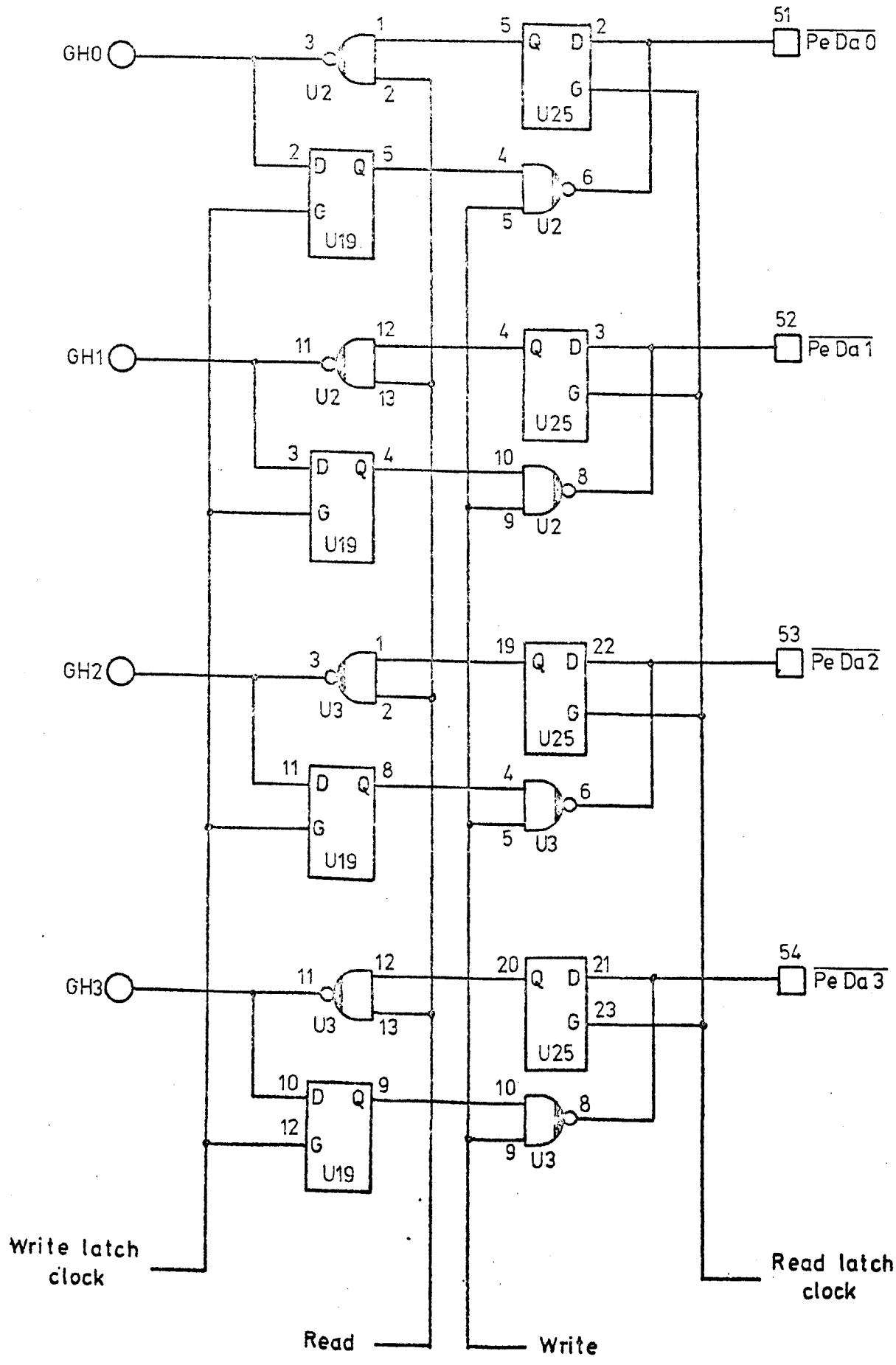


Figure 2.14 Bi-directional, latched data highway 0 - 3.

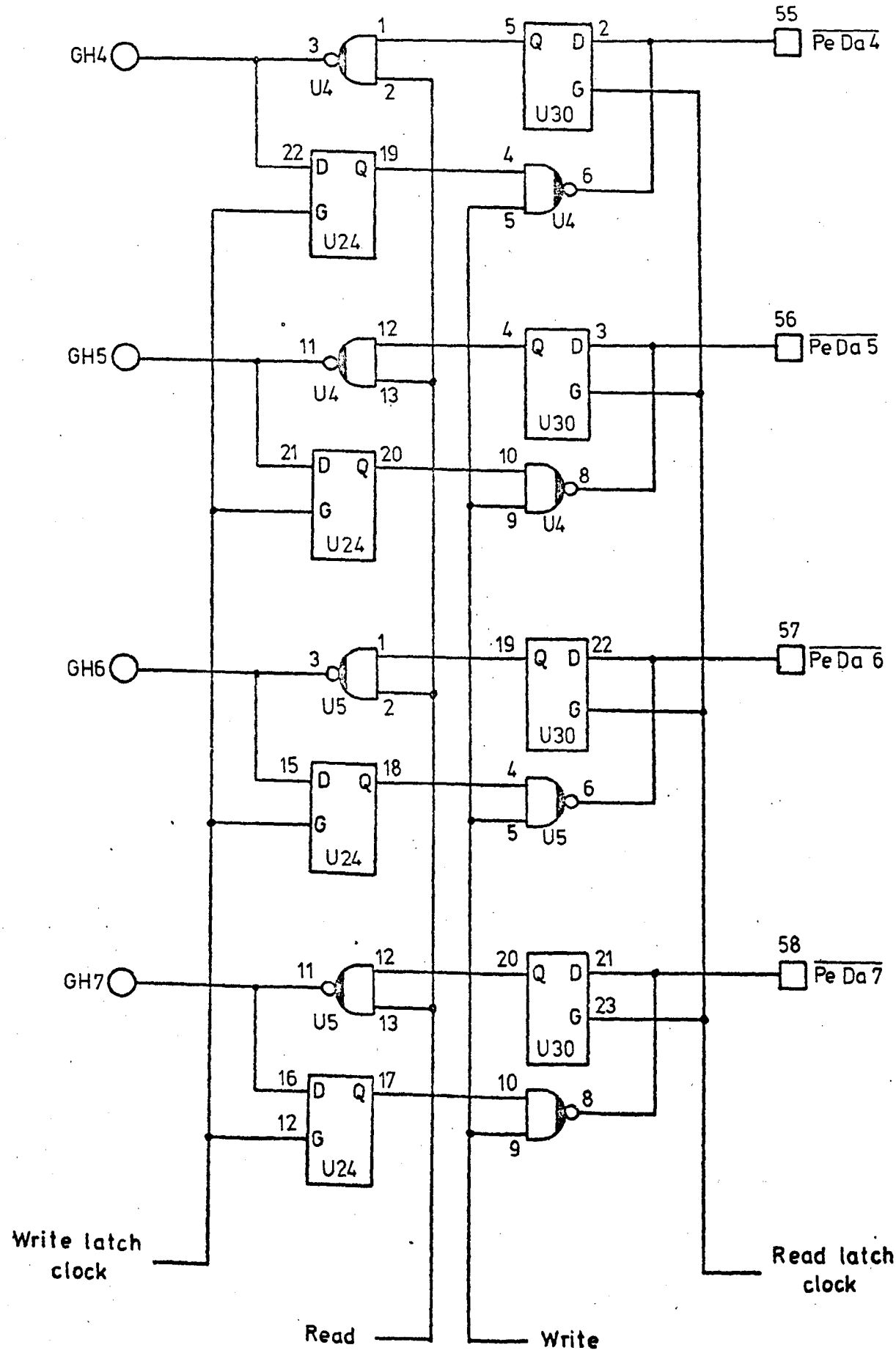


Figure 2.15 Bi-directional, latched data highway 4 - 7.

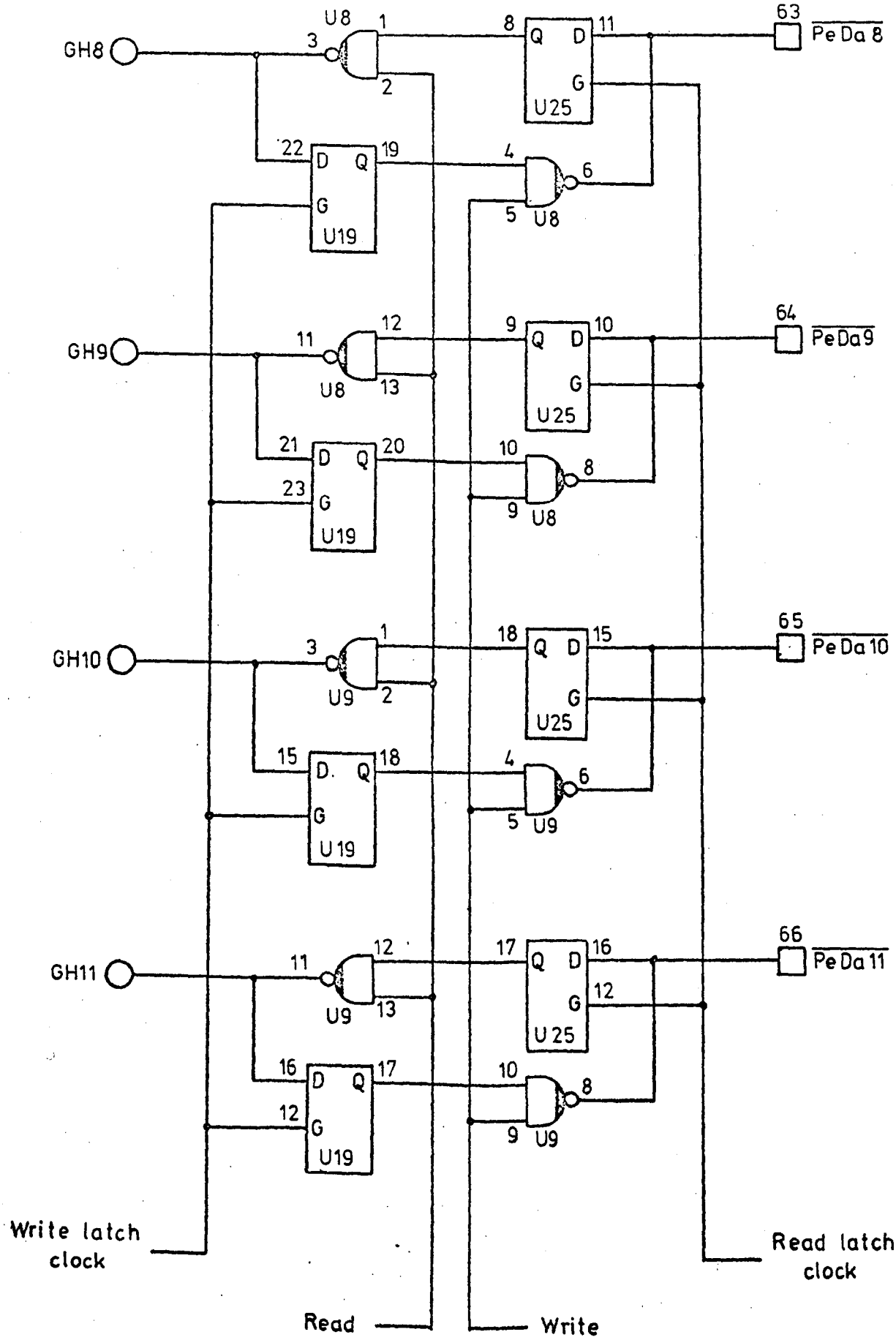


Figure 2.16 Bi-directional, latched data highway 8 - 11.

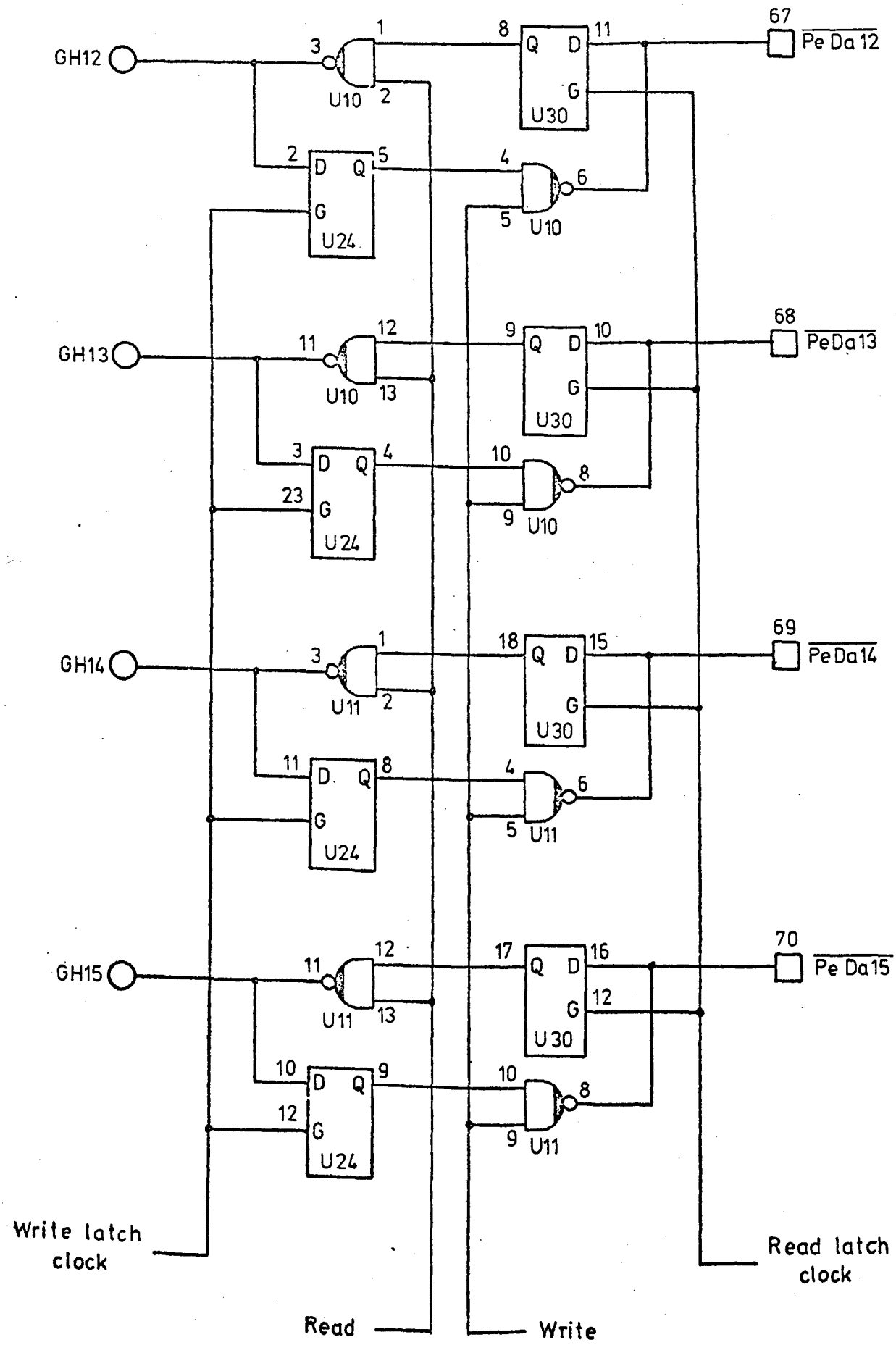


Figure 2.17 Bi-directional, latched data highway 12 - 15.

2.1.4 Miscellaneous Details

2.1.4.1 Component list

Resistors

All ¼w ±5%

R1, R2	330Ω
R3, R4, R18 - R33	220Ω
R5, R6, R9, R10, R13, R14	1k
R7, R8, R11	180Ω
R12, R16, R17	10k
R15	4.7k

Capacitors

C1, C2, C3, C12	1000pF ceramic
C4, C13, C14, C16	2.2nF ceramic
C5	68pF polystyrene
C6, C7	0.68μF elect. 64v
C8 - C11, C15	33nF ceramic

Semiconductors

D1, D2 Si diodes 1N4148

Integrated circuits

U1, U16, U23, U26	SN7404
U2, U3, U4, U5, U8, U9, U10, U11	SN7438
U7, U15, U33	SN7400
U13	SN7403
U19, U24, U25, U30	SN74100
U20	SN7451
U21, U22	SN7474
U27	SN7408
U28, U34	SN74123

Miscellaneous

4 x 24 pin d.i.l. sockets

2 x 16 pin d.i.l. sockets

1 x 25-way Cannon connector

1 x F100M 'Panic Plate' (p.c.b.) (designed and built at The City University by R. A. Comley).

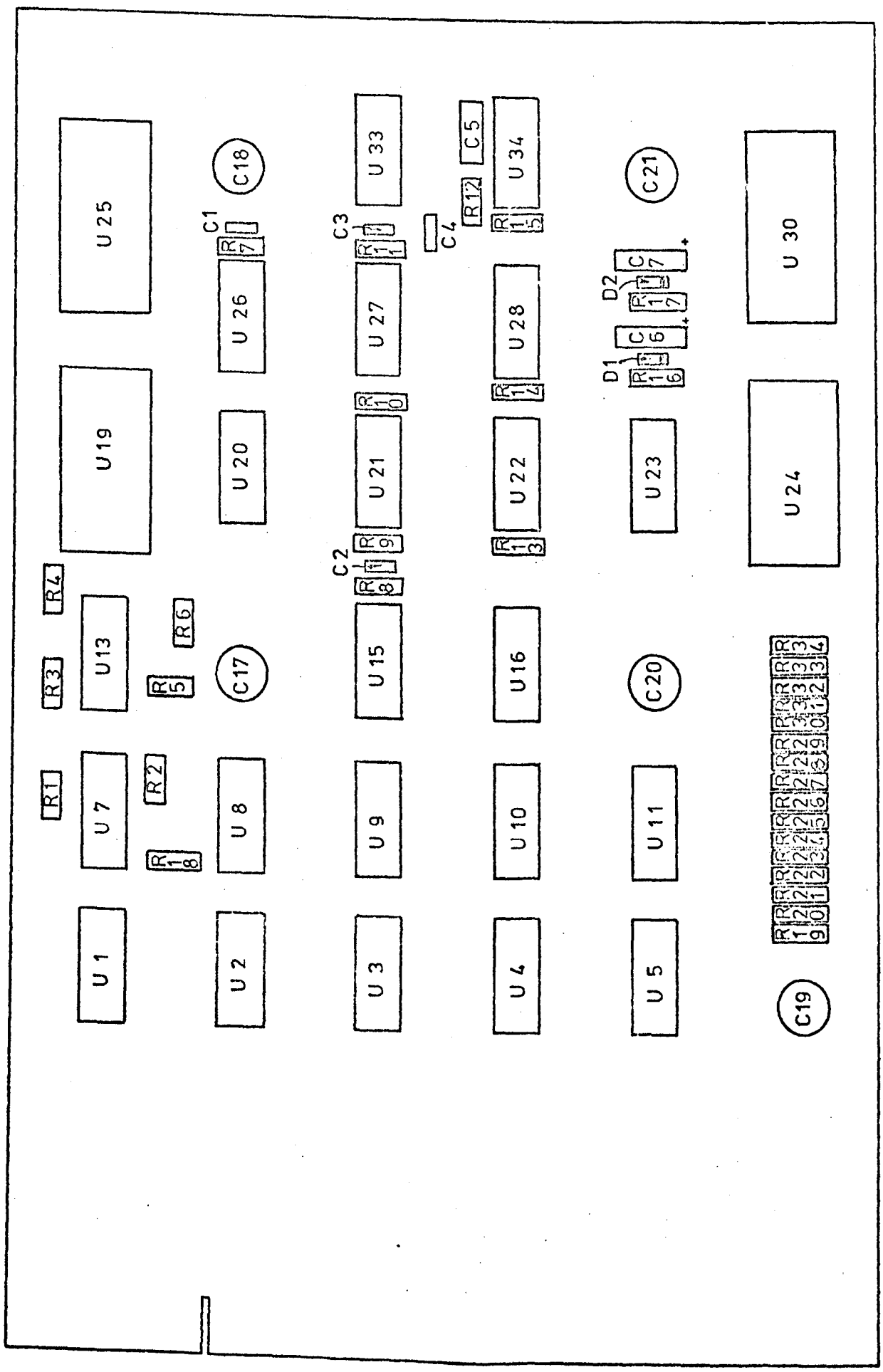


Figure 2.18 Component layout.

2.1.5 Connection lists

2.1.5.1 Cannon (peripheral) connector

1	Pe Da 0
2	Pe Da 1
3	Pe Da 2
4	Pe Da 3
5	Pe Da 4
6	Pe Da 5
7	Pe Da 6
8	Pe Da 7
9	Pe Da 8
10	Pe Da 9
11	Pe Da 10
12	Pe Da 11
13	Pe Da 12
14	Pe Da 13
15	Pe Da 14
16	Pe Da 15
17	$\overline{\text{At Rq}}$
18	$\overline{\text{Dir}}$
19	$\overline{\text{Q}}$
20	In Sr Cc
21	$\overline{\text{At Ac}}$
22	$\overline{\text{Enable}}$
23	-
24	-
25	-

2.1.5.2 F100M edge connector

	1	2	
	3	4	
	5	6	
	7	8	
	9	10	
0V	11	12	
+5V	13	14	
	15	16	$\overline{\text{Pe Ac 1}}$
	17	18	
	19	20	$\overline{\text{J Pe 1}}$
	21	22	
0V	23	24	
	25	26	Dir In 1
	27	28	
	29	30	$\overline{\text{In Sr Cc 1}}$
	31	32	
	33	34	Q1
0V	35	36	
	37	38	
	39	40	$\overline{\text{Pe Rq 1}}$
	41	42	
	43	44	
	45	46	
0V	47	48	
+5V	49	50	$\overline{\text{In Ti Out}}$
$\overline{\text{Pe Da 0}}$	51	52	$\overline{\text{Pe Da 1}}$
$\overline{\text{Pe Da 2}}$	53	54	$\overline{\text{Pe Da 3}}$
$\overline{\text{Pe Da 4}}$	55	56	$\overline{\text{Pe Da 5}}$
$\overline{\text{Pe Da 6}}$	57	58	$\overline{\text{Pe Da 7}}$
0V	59	60	$\overline{\text{K Pe 1}}$
$\overline{\text{Reset}}$	61	62	
$\overline{\text{Pe Da 8}}$	63	64	$\overline{\text{Pe Da 9}}$
$\overline{\text{Pe Da 10}}$	65	66	$\overline{\text{Pe Da 11}}$
$\overline{\text{Pe Da 12}}$	67	68	$\overline{\text{Pe Da 13}}$
$\overline{\text{Pe Da 14}}$	69	70	$\overline{\text{Pe Da 15}}$
0V	71	72	
	73	74	
	75	76	
	77	78	
	79	80	

In Sr Cc	$\overline{Q}$	$\overline{Dir}$	
0	0	0	Program interrupt
0	0	1	Load address counter
0	1	0	Load program counter
0	1	1	-
1	0	0	Read-modify-write
1	0	1	DMA read
1	1	0	DMA write
1	1	1	-

Table 2.1 Control line settings.

2.1.6 References

1. Gipop Buffer - Technical Description, Internal report, R. A. Comley, July '77.
2. F100M Documentation (technical notes), Ferranti Ltd., 1975 - 1977.
3. Design Specification for the Input-Output Controller (TN/32), Ferranti Ltd., June 1975.
4. HSM 150 - F100L Hardware Manual, Ferranti Ltd., 1977

2.2 User Guide

2.2.1 Introduction

The Gipop (general input-output) buffer is a general purpose data channel for the F100M microcomputer system. It is basically a sixteen-bit, bi-directional input-output buffer designed, primarily, for use in the FM1600B and F100M interface, but may be of general use wherever a slow peripheral must be interfaced to the F100M system.

The sixteen input-output lines are latched on the buffer board so as to relieve the user of the timing restrictions imposed by the F100M. The user may select the handshake response time by the use of two monostable circuits. At present, it is set to 2ms (approx.) for use with the Gipop link [2][3] and FM1600B.

The buffer is connected into channel 1 of the input-output controller (IOC) of the F100M system [4] and all data transfers take place under direct memory access (DMA) control. For this purpose three control inputs ($\overline{\text{Dir}}$, $\overline{\text{Q}}$, In Sr Cc) and a peripheral request-accept handshake pair are provided. The three control inputs are enabled via a fourth input ($\overline{\text{Enable}}$) which is active low (0v). This ensures that when not in use (i.e. Cannon plug disconnected) the board is disabled.

All input-output between the Gipop buffer and peripheral device is via a 25-way twisted-pair cable and a 25-way Cannon connector.

2.2.2 Electrical Specifications

All inputs and outputs of the Gipop buffer are TTL compatible. The use of open-collector drivers is recommended to drive all inputs to the buffer board. The data highway uses open-collector drivers (max. current sink 40mA) which are not pulled-up on the buffer board. A set of pull-up

resistors is provided, however, and may be connected to the V_{cc} supply if required. The three control inputs are pulled-up to V_{cc} via 330Ω resistors.

The enable input to the control lines should be tied to 0v on the peripheral so as to enable the buffer board when plugged in.

Reset on the processor is connected into the buffer board, hence a processor reset causes a buffer reset.

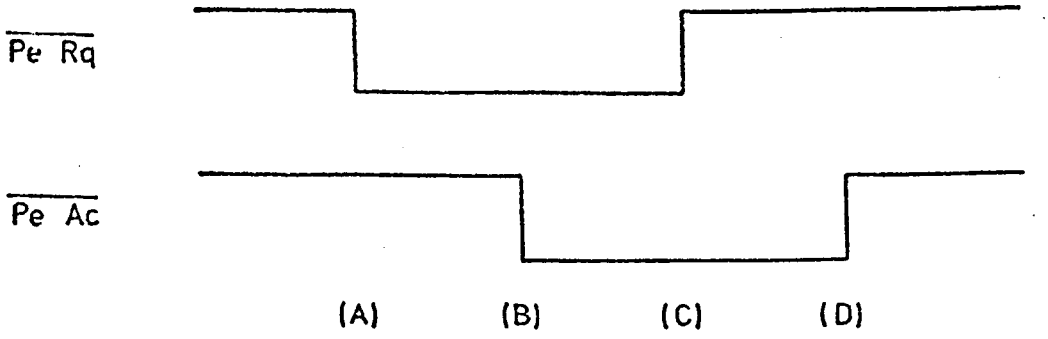
The operating speed of the buffer is controlled by the settings of two monostable circuits which are, at present, set to give 2ms (approx.) delay. This sets a maximum transfer rate of 250Hz (approx.). The delay may be reduced, or omitted completely, if desired, to speed up the rate of transfer for different peripherals.

2.2.3 Set-up requirements

The user must observe the handshake protocol given in Figure 2.2.1 to ensure correct operation of the buffer circuits.

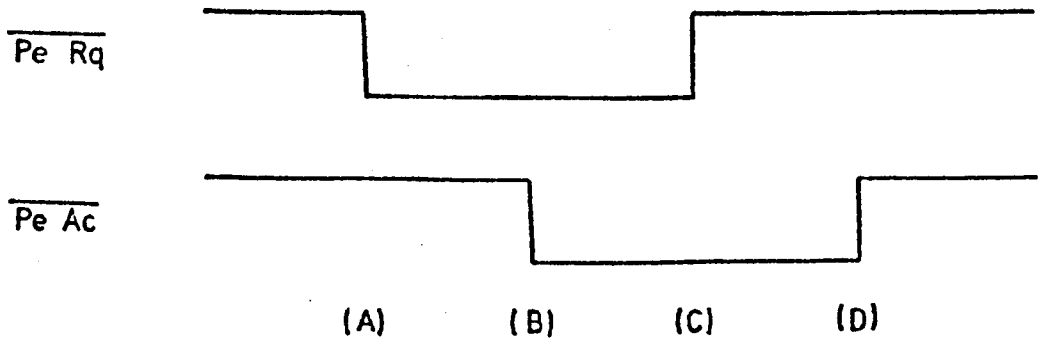
As transfers take place under DMA control, the control word must be stable before the first transfer request is issued and must remain stable throughout the transfer. The operations defined by the control inputs are as given in Table 2.1 (see Technical Description [1]).

Before any transfers can occur, the address counter of the F100M's IOC must be accessed; this sets-up the circuits from their random settings after power-up. The address of the counter for channel 1, used by the Gipop buffer, is 32763 (all handswitches set except 2 and 15) and data must be written to this location. The value of the written data is irrelevant. Note, the address counter performs a pre-increment before every read or write operation, and so the start address loaded into the counter should be one less than the required start address.



(A) Transfer request
(B) Transfer accepted
(C) Data on bus
(D) Data accepted

(a) Read Cycle



(A) Transfer request
(B) Data on bus
(C) Data accepted
(D) Data removed from bus

(b) Write Cycle

Figure 2.2.1 Handshake timing waveforms.

For futher details on the use of the IOC see Refs. 3 and 4 .

2.2.4 Mechanical details

The Gipop buffer is constructed on a single 233 x 169mm 'breadboard' designed specifically for random logic layouts. Connection to the board is via a 25-way twisted-pair cable and a Cannon connector, the connections for which are given in section 2.1.5.1 (Technical Description).

2.2.5 References

1. Gipop Buffer - Technical Description, Internal report, R. A. Comley, July 1977.
2. Gipop Link - Technical Description, Internal report, R. A. Comley, July 1977.
3. Gipop Link - User Guide, Internal report, R. A. Comley, July 1977.
4. Design Specification for the Input-Output Controller (TN/32), Ferranti Ltd., June 1975.

Chapter 3

Non - Volatile Semiconductor Store

3.1 Introduction

The memory system to be described is a 4k x16 bit memory constructed from 4k x 1 bit NMOS dynamic storage devices and interfaced to a Ferranti F100M microprocessor. Texas TMS4030 RAMs are used which were, at the commencement of the design, the best devices available, but have since been superseded by numerous other memory types. However, the design criteria employed are applicable to any non-volatile storage system which utilizes dynamic semiconductor devices.

Re-chargeable batteries are proposed as a means of providing power when the mains supply is disconnected, to maintain the memory packages and any essential support circuits. In this manner the memory can be made effectively non-volatile and it is hoped that a standby period of at least one day (24 hrs.) may be achieved before the batteries require re-charging.

The design and construction of the memory system have not been finalized and so no actual performance figures are available.

3.2 General Design Considerations

Dynamic rather than static memory devices are used as they exhibit lower standby power requirements, despite the need for periodic refresh cycles. The average power drain during standby for both types of device are shown in Figure 3.1.

In order for the dynamic device to maintain its advantage, the power requirements of the refresh circuits must be minimized. This can be achieved by the use of a combination of CMOS and discrete circuits, to reduce the standby power. To reduce the operating power, all line capacitance effects must be minimized. A suitable memory device should be chosen (the TMS4030 features very low input capacitance figures:see Table 3.1) and the number of signals

switched into these capacitive loads, minimized. For the address inputs, the use of a Grey code appears to offer the optimum performance (see Vol. I - sect. 4.4).

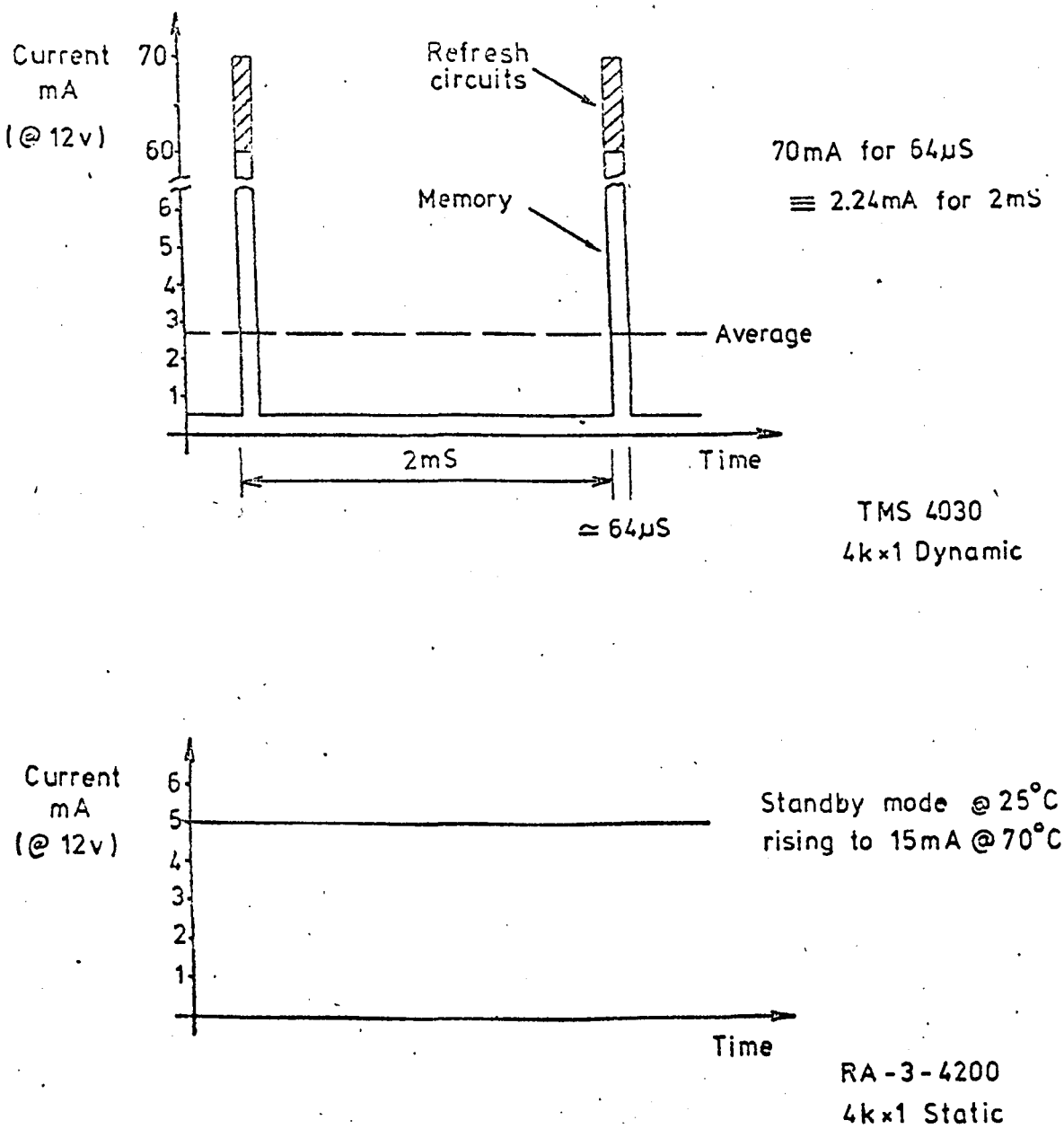


Figure 3.1 Comparison of power requirements for dynamic and static memory devices.

All non-essential circuits are turned off during the standby mode and, further, must not interfere with the circuits which are still active whilst in this state. Low-power Schottky circuits are useful in this respect since their outputs are clamped to 0V when no power is supplied; other device outputs go into a high impedance state and so may 'float' to an indeterminate level which may result in the generation of spurious signals, unless clamped (pulled-up).

The other major consideration for the non-volatile store is one of power supplies and voltage regulation. Any losses associated with the voltage regulation stages involved with the battery supply must be minimized in order to provide the maximum standby period (see Vol.I - sect. 4.6).

4096 × 1 Organization
300ns Maximum Access Time
470ns Maximum Cycle Time
710ns Maximum Read/Modify/Write Cycle Time
Full TTL Compatibility on all Inputs (No pull-up resistors needed)
Registers for Address and Chip Select Provided on Chip
Three-state Output Buffers
Full TTL Output Capability (Fan-out of 2)
Low Capacitance Inputs - 7pF max. (except Clock)
Single Low Capacitance Clock - 27pF max.
Low power Dissipation
  400mW Operating
  2mW Standby
Standard Power Supplies
  +12V @ 60mA max. Operating
    @ 0.5mA max. Standby
  +5V @ 1mA max.
  -3V @ 100 A max.
N-Channel Technology
22-Pin Dual-in-Line Package

Table 3.1 Characteristics of TMS4030 Random Access Memory.

3.3 The Texas TMS4030

The TMS4030 is a 4096 x 1 bit dynamic random access memory (RAM) fabricated in the NMOS technology and organized internally as a 64 x 64 array. A total of twelve address lines is provided, six column and six row, to specify the access of any individual memory cell. The device is housed in a 22-pin dual-in-line package with TTL compatibility on all inputs except the clock (C.E.). The main characteristics are listed in Table 3.1.

Three modes of operation are possible, read, write and read-modify-write. The timing requirements for the control signals involved in each of these operations are given in Figure 3.2.

A refresh operation is required every 2ms if data are not to be lost. A refresh is accomplished by a complete cycle of the sixty-four combinations of the row address inputs (A0 - A5). The chip should be placed in the read mode and need not be selected during refresh, which can help reduce the power requirements in the standby mode.

3.4 Design and Constructional Details

The complete RAM is built-up from six functional units (Fig. 3.3). Only the refresh circuits, some of the drivers and the RAMs themselves are powered during the standby mode, all others being turned off.

The RAM packages are configured in a 16 x 1 array, with all common inputs connected in parallel. This can have important consequences on the input loads presented to the RAM drivers, which may become highly capacitive.

3.4.1 Control circuit

A control circuit is required to provide the following functions:-

i) generate all necessary clocking waveforms, i.e.

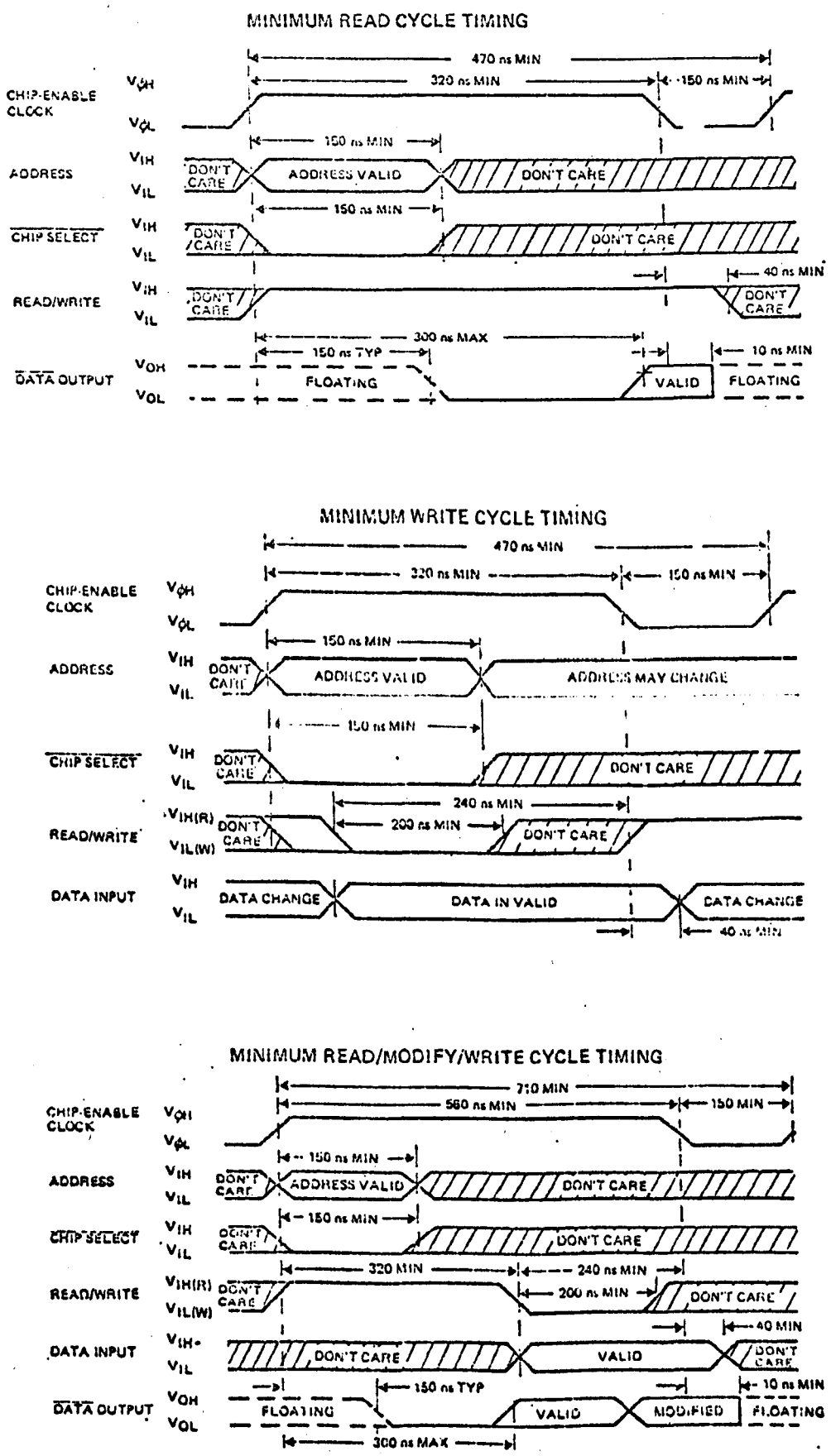


Figure 3.2 Minimum cycle times for the three operating modes of the TMS4030 RAM.

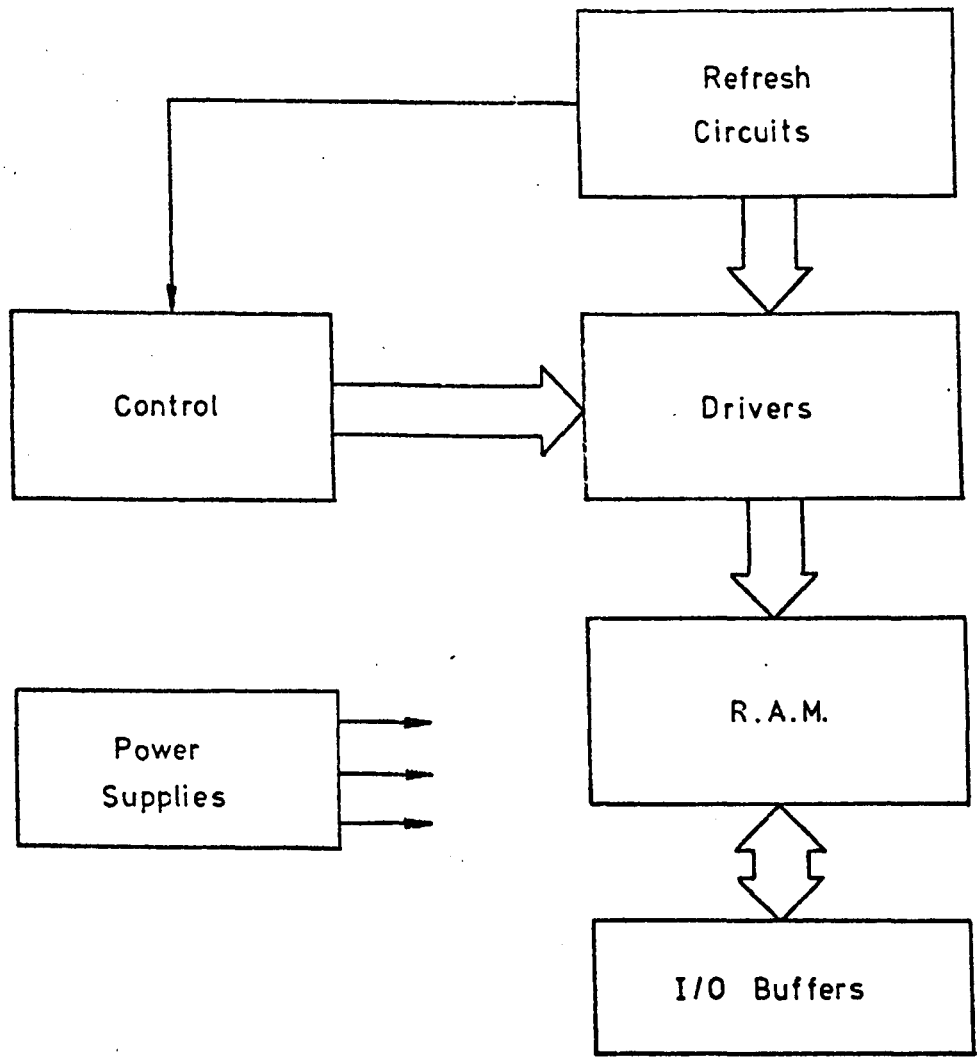


Figure 3.3 Functional units of the non-volatile RAM.

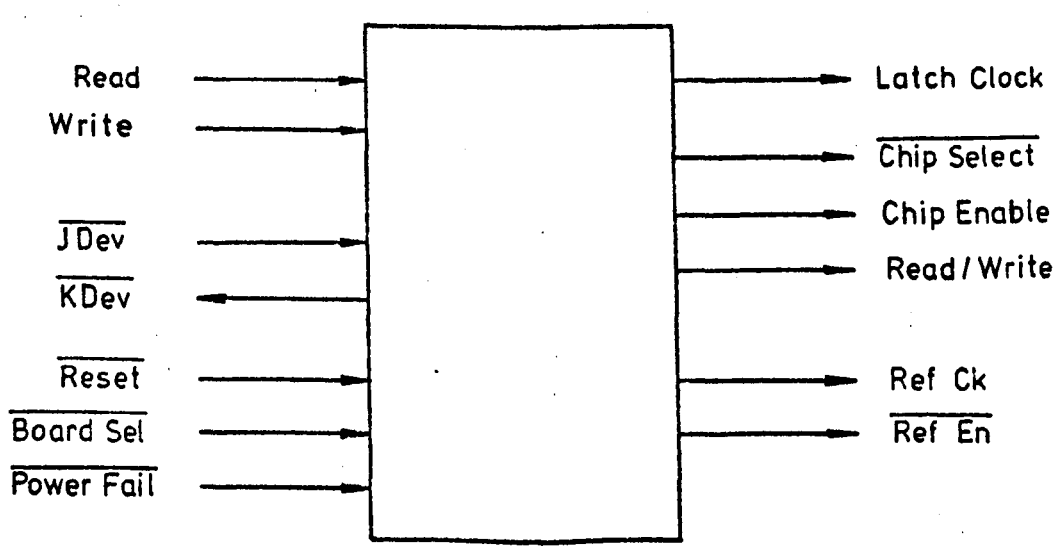


Figure 3.4 Block diagram of control circuit.

chip enable, chip select, read-write, etc.

ii) perform 'handshake' operations

iii) decode operating mode, i.e. read, write or read-modify-write

iv) control refresh operations during the operating mode

v) facilitate other functions such as reset, board select, time-out, etc.

The complete control circuit is shown in block diagram form in Figure 3.4.

i) Clocking waveforms.

The clocking waveforms required by the RAM are given in section 3.3 (Fig. 3.2). The read-modify-write operation cannot be implemented directly as shown in Figure 3.2.c with the F100 system, since the period between the read and write cycles would exceed the maximum cycle time for the memory (2μs). Hence, the read-modify-write cycle must be implemented as a read followed by a write cycle.

The generation of the read and write cycle waveforms is somewhat complicated for the F100 system due to its asynchronous data transfer method (Vol.I - sect. 4.2). One solution is to use monostable circuits, triggered by the handshake signals, to generate the delays required for the clocking waveforms. This method suffers from problems of long term stability (component aging effects), spurious triggering and accuracy. If the memory is to be operated close to its maximum frequency, the clocking waveforms must be controlled to very close tolerances, which is not practically possible with the monostable system.

A more satisfactory solution is to use an oscillator circuit and some form of feedback shift-register; in this case a two stage shift-register will suffice (Fig. 3.5). The clock input should be generated from a crystal-controlled oscillator, to provide a suitable accuracy, and of 160ns

	C.E.	$\overline{C.S.}$
Reset	0	1
	1	0
One cycle	1	1
	0	1
Next cycle	1	0

$(C.E.)_{n+1}$

CS \ CE	0	1
0	1	1
1	1	0

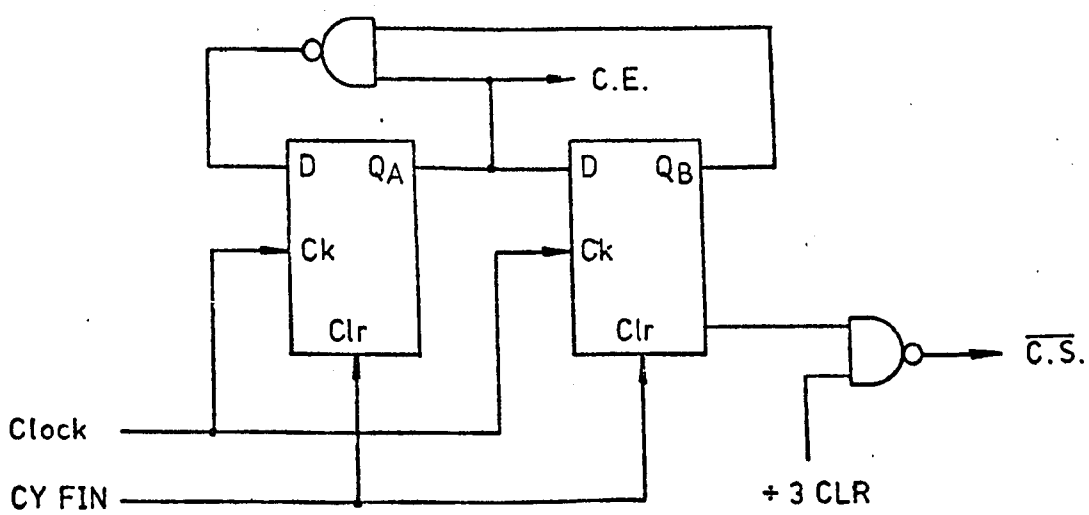
$(C.E.)_{n+1} = \overline{(C.E.)}_n + \overline{(C.S.)}_n$

$\overline{(C.S.)}_{n+1}$

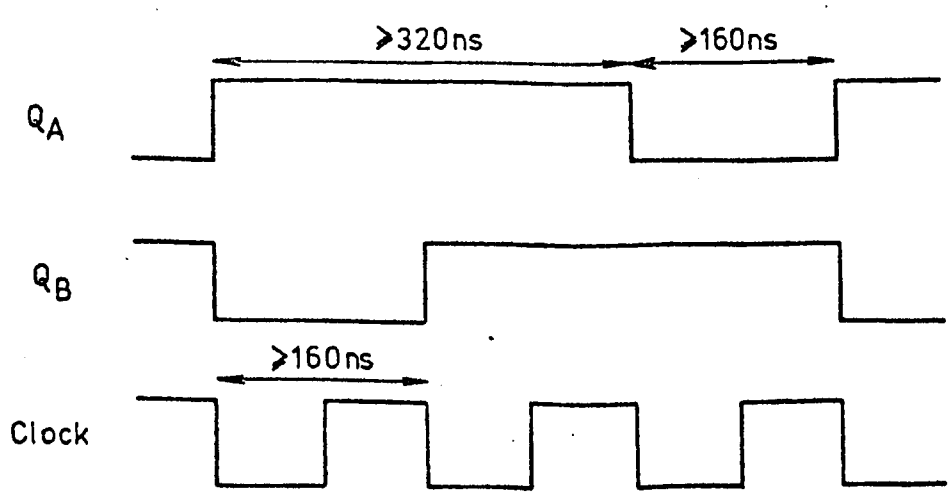
CS \ CE	0	1
0	0	1
1	0	1

$\overline{(C.S.)}_{n+1} = (C.E.)_n$

a) Required feedback function



b) Circuit to generate functions given in (a)



c) Clocking waveforms

Figure 3.5 Generation of C.E. and $\overline{C.S.}$ clocking waveforms.

minimum period to ensure correct operation. This will increase the minimum cycle time by 10ns, to 480ns which is not considered to be an unreasonable overhead.

The read-write waveform can be simply derived from the chip enable (C.E.) output (Fig. 3.6).

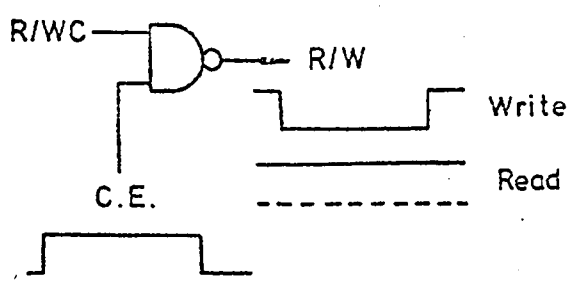


Figure 3.6 Generation of R/W waveform.

The clock input for the feedback generator may be derived from either an oscillator circuit constructed specifically for the purpose, or from the main system clock. A separate oscillator has the advantage that it may be started and stopped in synchronization with the read-write requests, via the handshake signals. This enables the memory to be used at its maximum frequency, but in this case the solution must be rejected on size and weight grounds (these must be minimized for the RSP application). As a result, the main system clock is used as the clock source.

For the F100M, an 8MHz clock is required and for the F100L, a 16MHz clock. Hence, if a 16MHz clock is assumed, the memory will be compatible with both systems (since the F100M's clock may be easily obtained from a 16MHz source). The period for a 16MHz input is 62.5ns and so a divide-by-three circuit will be required to generate the clock input for the shift-register (Fig. 3.7). This results in a minimum cycle time of 562.5ns.

The clock to the feedback shift-register must be started and stopped in such way that spurious or part main-clock cycles are not output. A set-reset flip-flop is used to

provide this function, via the divider's clear input. The flip-flop is set by a start cycle (ST CYL) command and cleared at the end of the requested operation (Fig. 3.8).

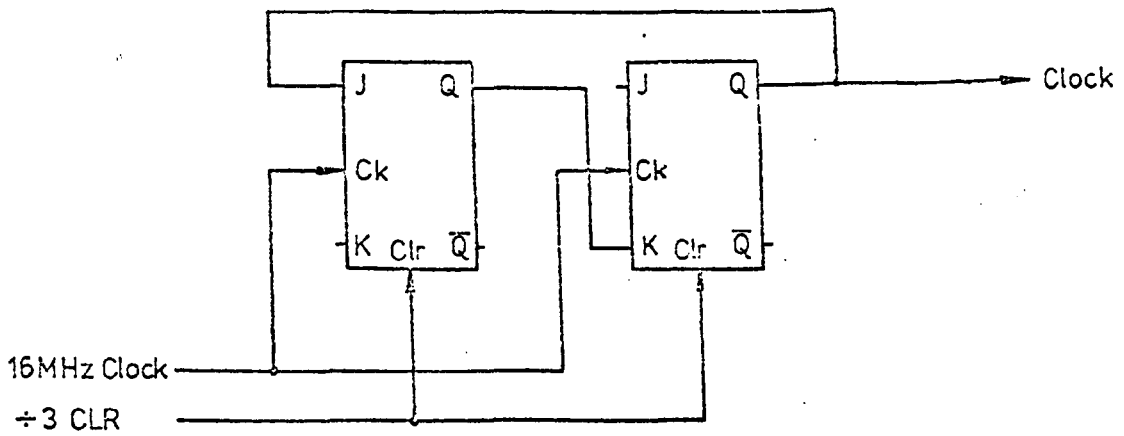


Figure 3.7 Divide-by-three circuit.

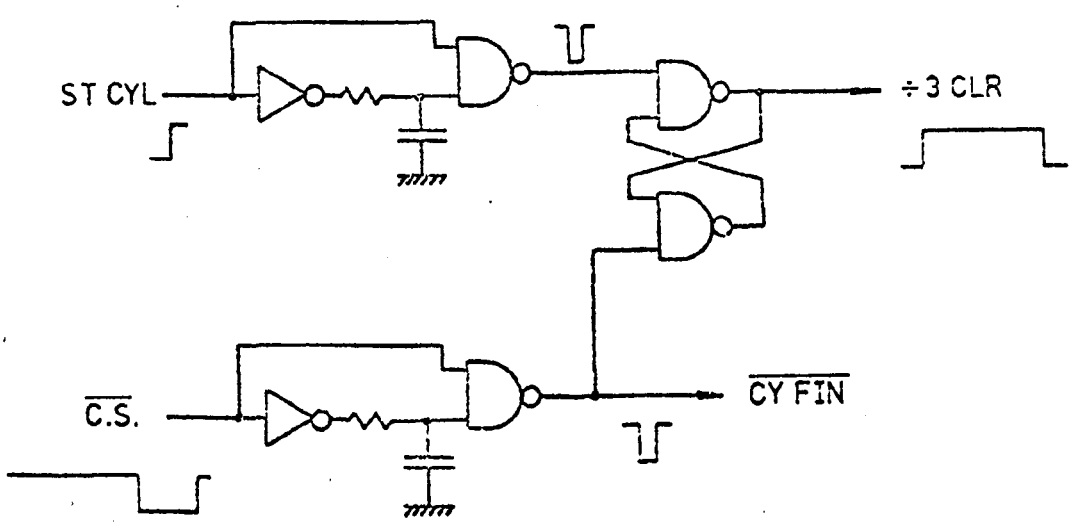


Figure 3.8 Clear signal for divide-by-three circuit.

The ST CYL signal is a function of both the read-write mode and JDev handshake line and is delt with in sub-section (ii). The cycle finish ($\overline{\text{CY FIN}}$) signal is generated from the chip select ($\overline{\text{C.S.}}$) signal of the feedback shift-register. The use of these signals results in the desired start-stop characteristics (Fig. 3.9).

The divider and clock generator circuits must also be

enabled during refresh operations. This requires that the ST CYL command be generated from the refresh enable circuit and is discussed in sub-section (iv).

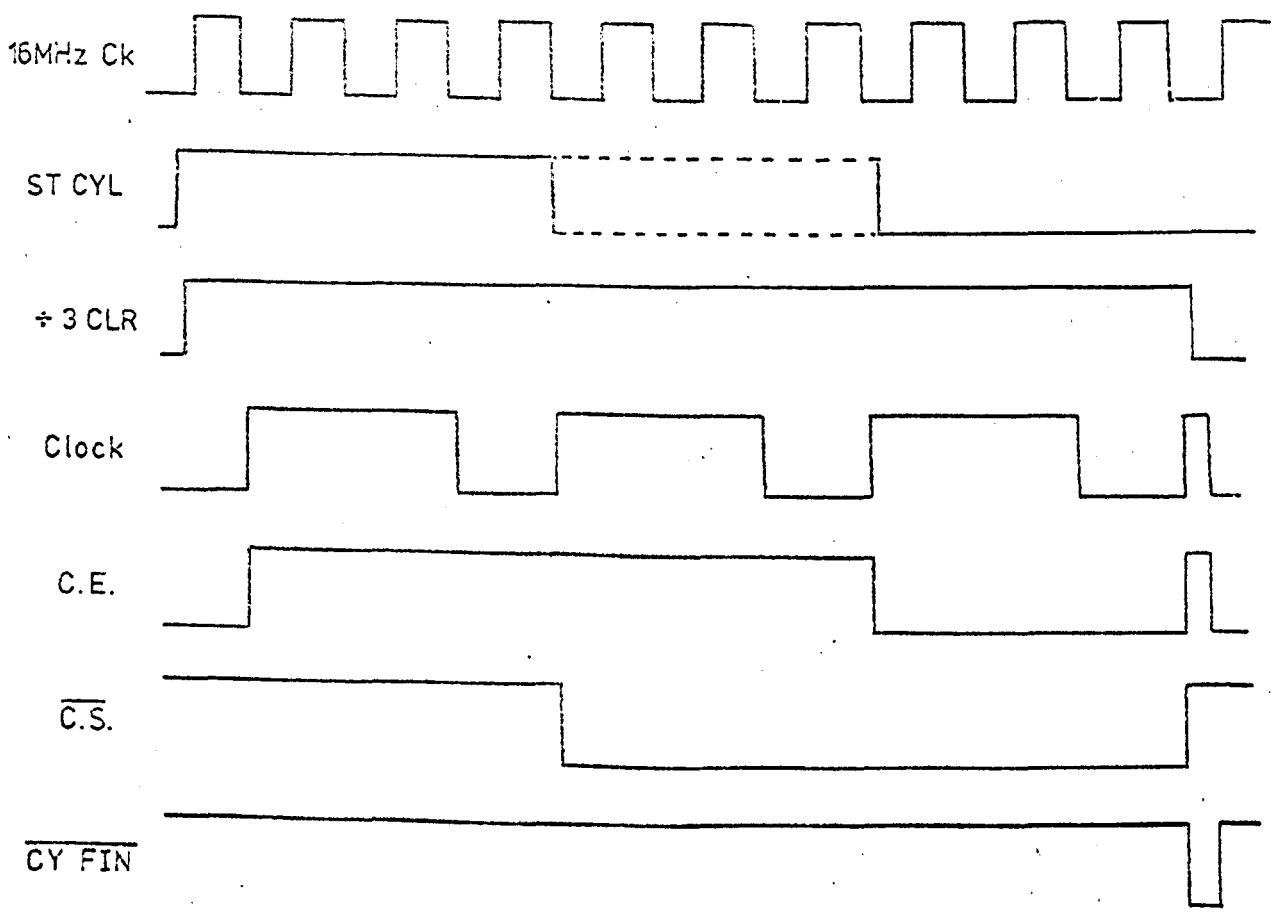


Figure 3.9 Complete clocking sequence for one cycle.

The final clocking waveform required is for the bi-directional input-output data latch, which is included to help eliminate any interface timing problems and decrease effective cycle times, as seen by the CPU (see sub-section (ii)). A positive-going clock pulse is required, to strobe data into the latch, at different positions in the cycle, depending upon whether a read or write operation has been selected. It is required on the rising edge of $\overline{C.S.}$ for a write cycle and the trailing edge of C.E. for a read (see Fig. 3.11). A suitable circuit to perform this operation is as shown in Figure 3.10.

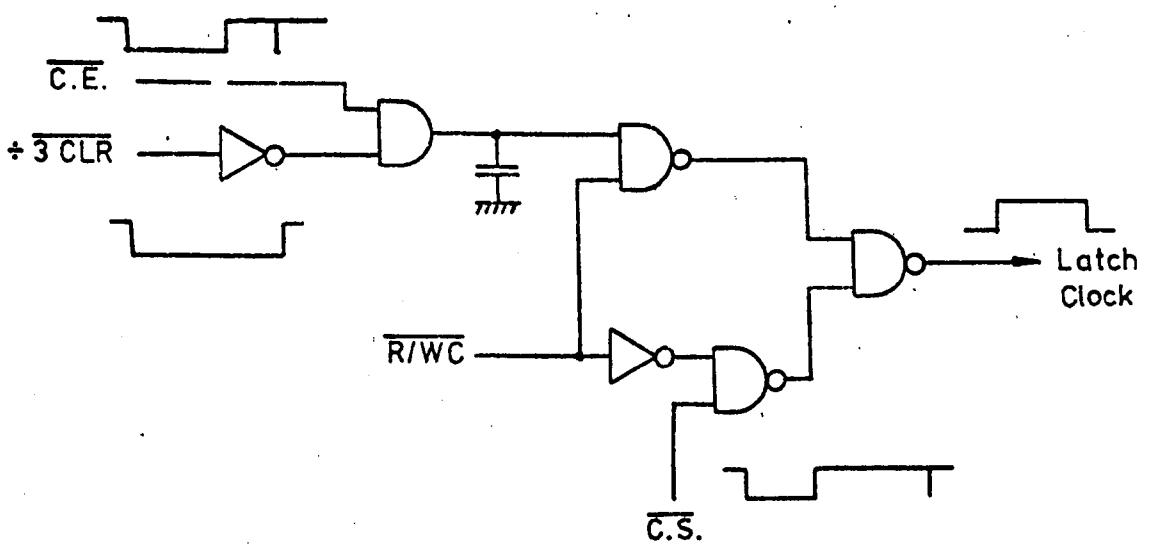


Figure 3.10 Latch clock generator circuit.

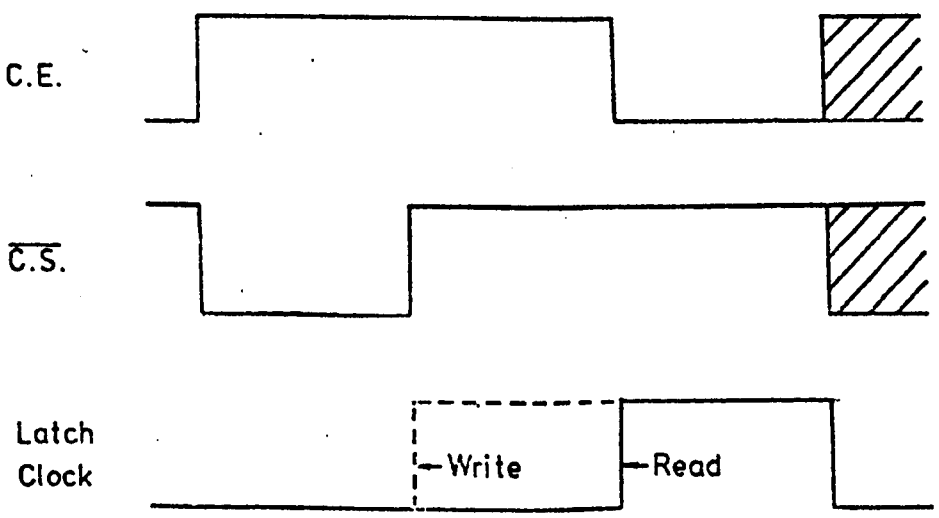
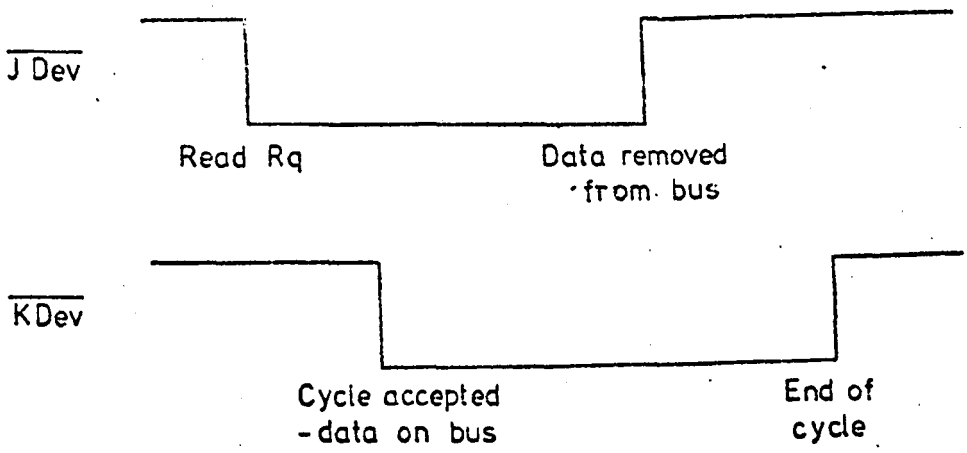


Figure 3.11 Timing diagram for latch clock circuit.

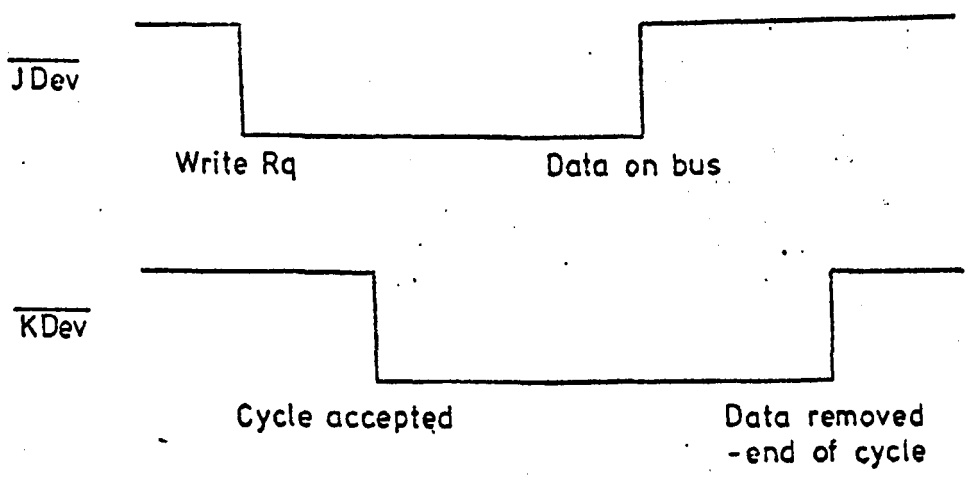
ii) Handshake Controls

The F100 system transfers data on an asynchronous handshake basis, using two control lines as shown in Figure 3.12.

For a read request, the $\overline{\text{JDev}}$ input initiates a read cycle on its trailing edge and when data becomes available, the $\overline{\text{KDev}}$ output may be taken low. The $\overline{\text{KDev}}$ output must then remain low until the $\overline{\text{JDev}}$ input is cleared (Fig. 3.13.a). For a write request, the start of the write cycle is delayed until the rising edge of $\overline{\text{JDev}}$ is received, indicating that



a) Read cycle



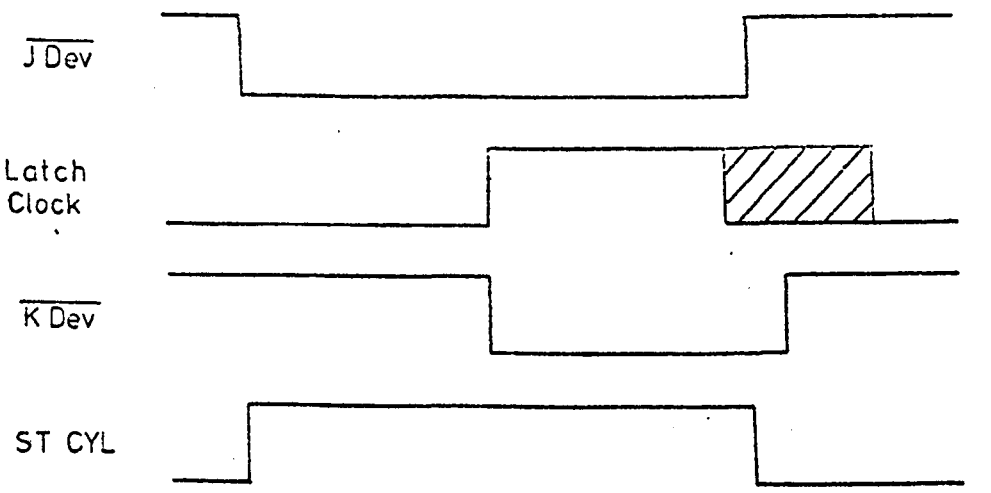
b) Write cycle

Figure 3.12 Handshake timing diagrams for the F100 system.

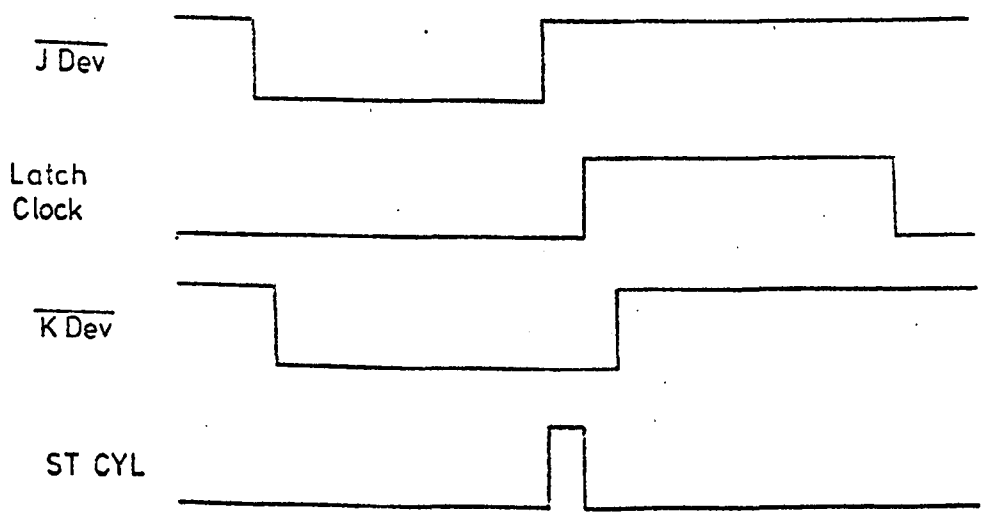
the data to be written are available on the bus. The $\overline{\text{KDev}}$ output is taken low as soon as the falling edge of $\overline{\text{JDev}}$ is received, unless a previous operation is still in progress, and is not cleared until the latch clock (see sect. 3.4.4) indicates that the data have been latched (Fig. 3.13.b). Note, the rising edge of $\overline{\text{C.S.}}$ generates the latch clock signal in the write mode and so both data and address may be changed at this stage, although the write cycle has not been completed.

Operation of the memory in this manner results in a considerable reduction in the effective cycle time of the memory, particularly for the write mode; it does however require that the next $\overline{\text{JDev}}$ input must be inhibited until the

full cycle time has elapsed. Various other inputs are also used to delay or inhibit the $\overline{\text{JDev}}$ signal as shown in Figure 3.14.



a) Read cycle



b) Write cycle

Figure 3.13 Read and write handshake sequences.

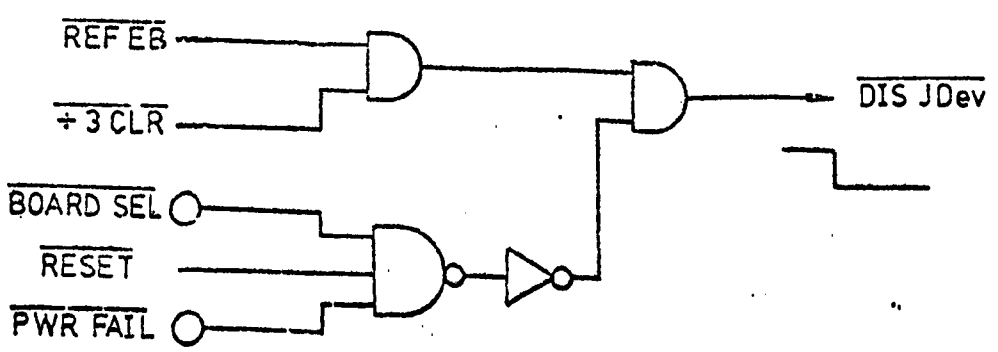


Figure 3.14 Disable inputs for $\overline{\text{JDev}}$ signal.

The use of the ÷3CLR input to provide the 'full cycle delay' requires that a set-reset latch is included in the $\overline{\text{JDev}}$ input circuit, as shown in Figure 3.15.

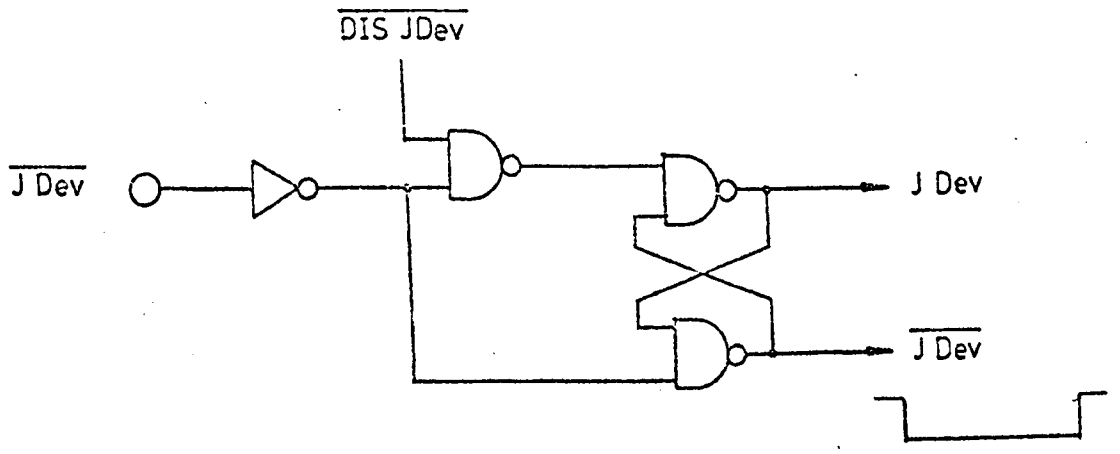


Figure 3.15 $\overline{\text{JDev}}$ input circuit.

From Figure 3.13.a it can be seen that for a read cycle the $\overline{\text{KDev}}$ output should be set by the rising edge of the latch clock input and cleared on the rising edge of $\overline{\text{JDev}}$; for a write cycle, the $\overline{\text{JDev}}$ input sets $\overline{\text{KDev}}$ and the latch clock should clear it. A suitable circuit is as shown in Figure 3.16.

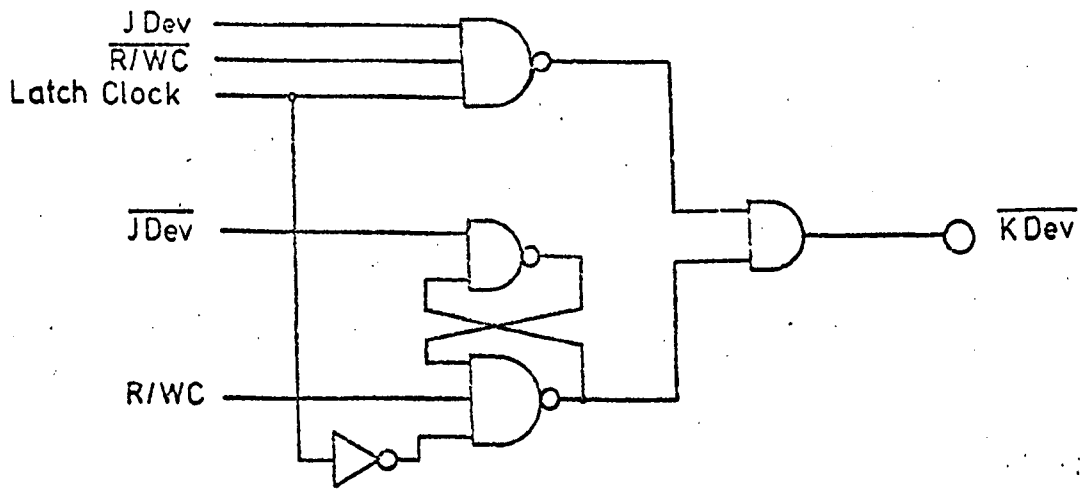


Figure 3.16 $\overline{\text{KDev}}$ handshake circuit.

In addition to the control of the $\overline{\text{KDev}}$ output, the $\overline{\text{JDev}}$ input must also generate a start cycle (ST CYL) signal to start the divide-by-three circuit (see Fig. 3.7) . This is generated directly from the $\overline{\text{JDev}}$ input and is controlled by

the setting of the read-write line (see Figs. 3.17 and 18).

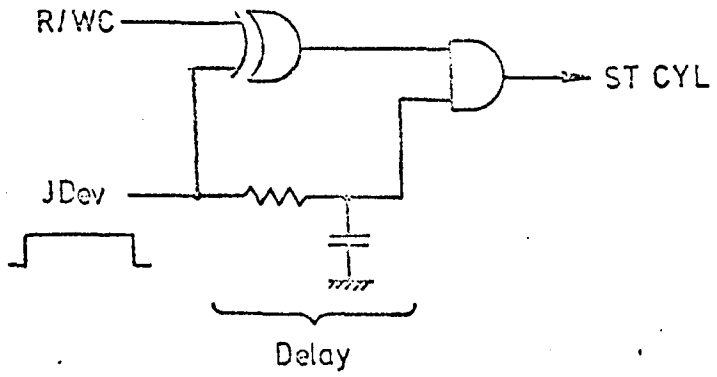


Figure 3.17 Generation of the start cycle command.

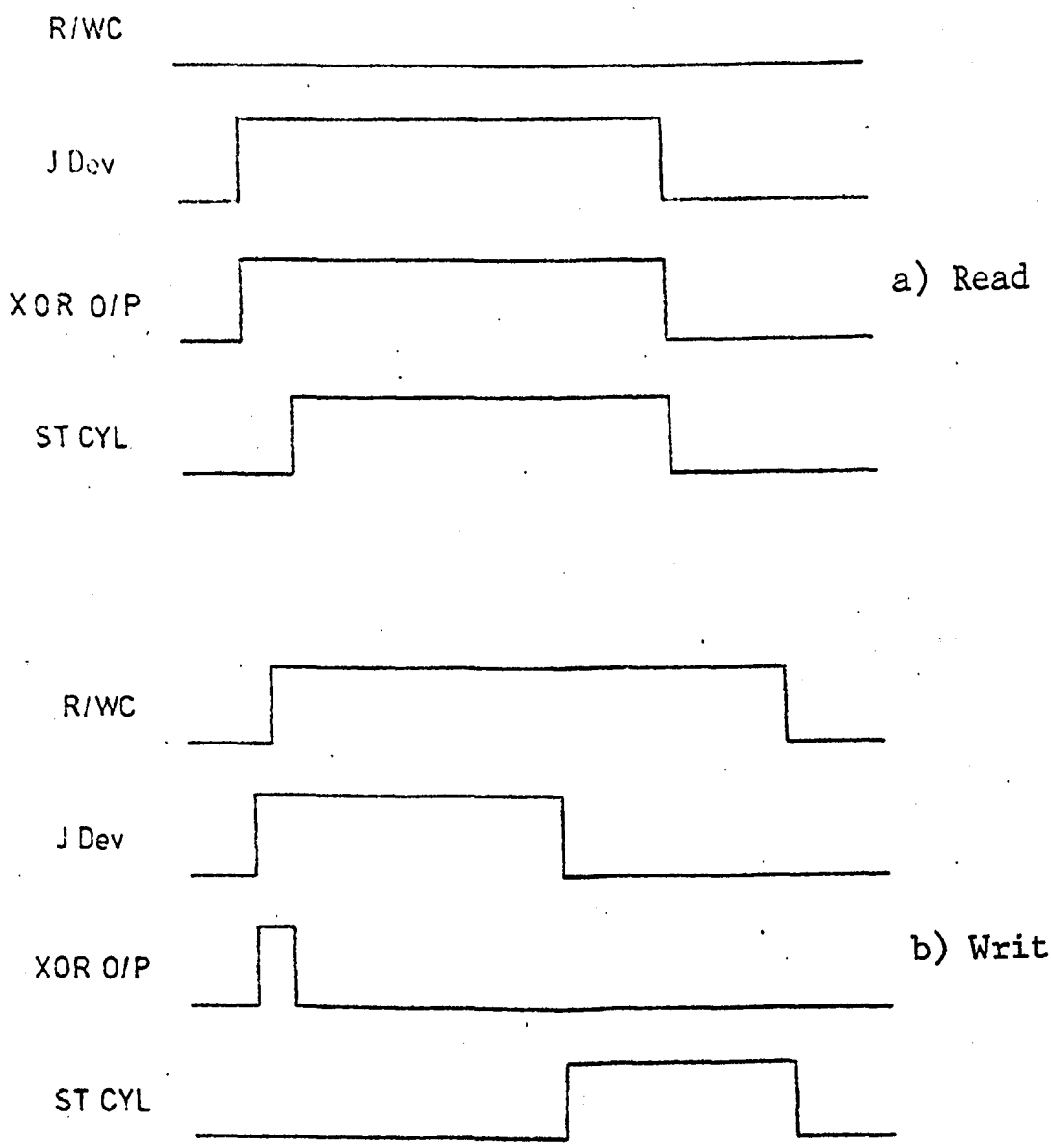


Figure 3.18 Start cycle timing.

iii) Mode Select

Two control lines are set by the F100 system to indicate the required mode of operation for the memory:-

$\overline{RD}$	$\overline{WR}$	Function
1	1	None (set to read)
0	1	Read
1	0	Write
0	0	Read-modify-write

For normal read or write operations, the $\overline{RD}$ signal may be used to provide the read-write (R/W) signal, necessary for the other circuits, since for a read, $\overline{RD}$ = 0 and for a write, $\overline{RD}$ = 1, as required. The read-modify-write (RMW) case is a little more complicated, however.

The maximum allowable cycle time for the RAM is 2μs and so the RMW cycle must be performed as two separate operations; a read followed by a write. The handshake lines for the memory are, in fact, provided by a special two-port interface [1] which produces separate read and write operations for the RMW cycle and so all that is required is to ensure that the R/W line is set to the appropriate level for the two portions of the sequence.

A two-stage counter is required, which is only enabled during RMW cycles and resets itself at the end of the write cycle (Fig. 3.19). This ensures correct operation for consecutive RMW cycles.

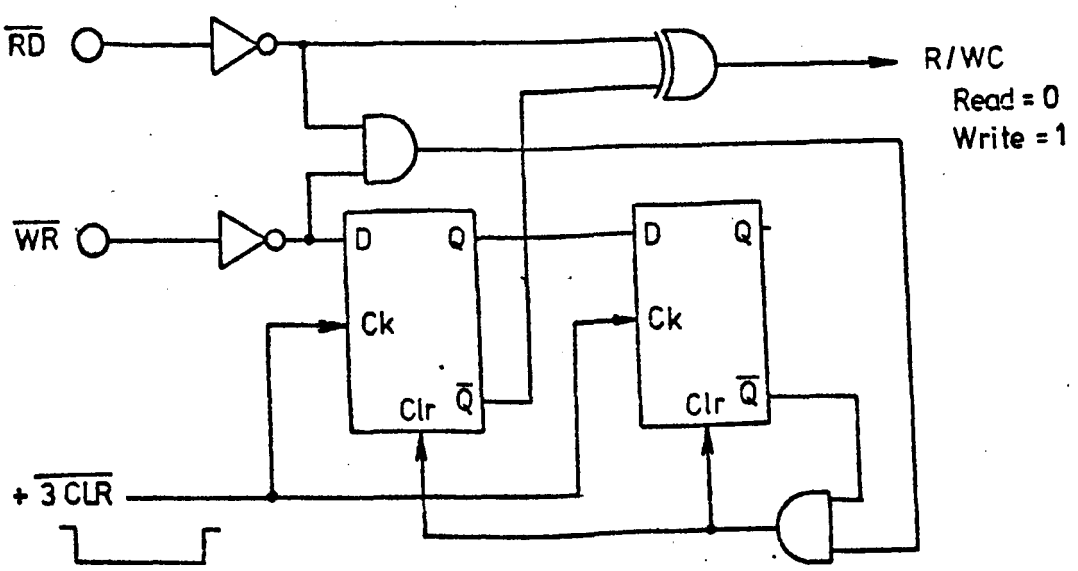


Figure 3.19 Mode select circuit.

iv) Refresh Control

The function of this section of the circuit is to generate periodic refresh cycles. In the operating mode, the refresh operations are to be dispersed throughout the entire 2ms refresh period. Since sixty-four refresh cycles are required for a complete refresh of the memory, a cycle is required every 2/64μs or 31.25μs. The refresh operation is performed in this manner to avoid the possibility of a sixty-four cycle delay in the acceptance of an external read or write request. This could occur, since the refresh operation must take precedence over normal read-write cycles which must hence be disabled during the refresh period.

The basis of the circuit is a monostable which sets a refresh request ($\overline{\text{REF RQ}}$) signal. Upon completion of any operation, should one be in progress, the refresh enable ($\overline{\text{REF EB}}$) output is set, which places the current refresh address on the address lines of the RAM. The $\overline{\text{REF RQ}}$ signal may be cleared at this stage and so $\overline{\text{REF EB}}$ may be used directly to re-trigger the monostable. The $\overline{\text{REF EB}}$ signal is also used to generate the start cycle (ST CYL) command, thereby initiating the refresh cycle (Figs. 3.20 and 21). The $\overline{\text{C.S.}}$ and latch clock outputs are disabled during the refresh cycle.

The monostable delay is set to 28μs, to allow for possible inaccuracies in the monostable and timing components and the fact that the refresh request may be delayed by up to one cycle time.

v) Other Functions

The additional control inputs have been discussed in connection with the handshake controls (see sub-sect. (ii)) and will not be repeated here.

Although a maximum cycle time of 2μs exists for the TMS4030 RAM, no time-out circuits are required since, by the

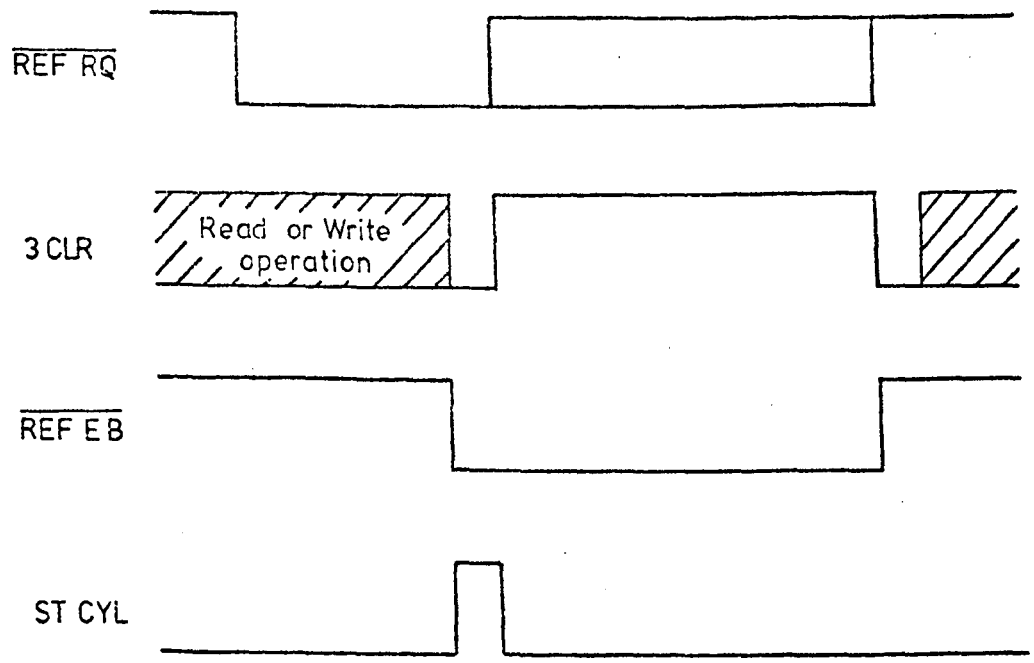


Figure 3.20 Refresh cycle timing sequence.

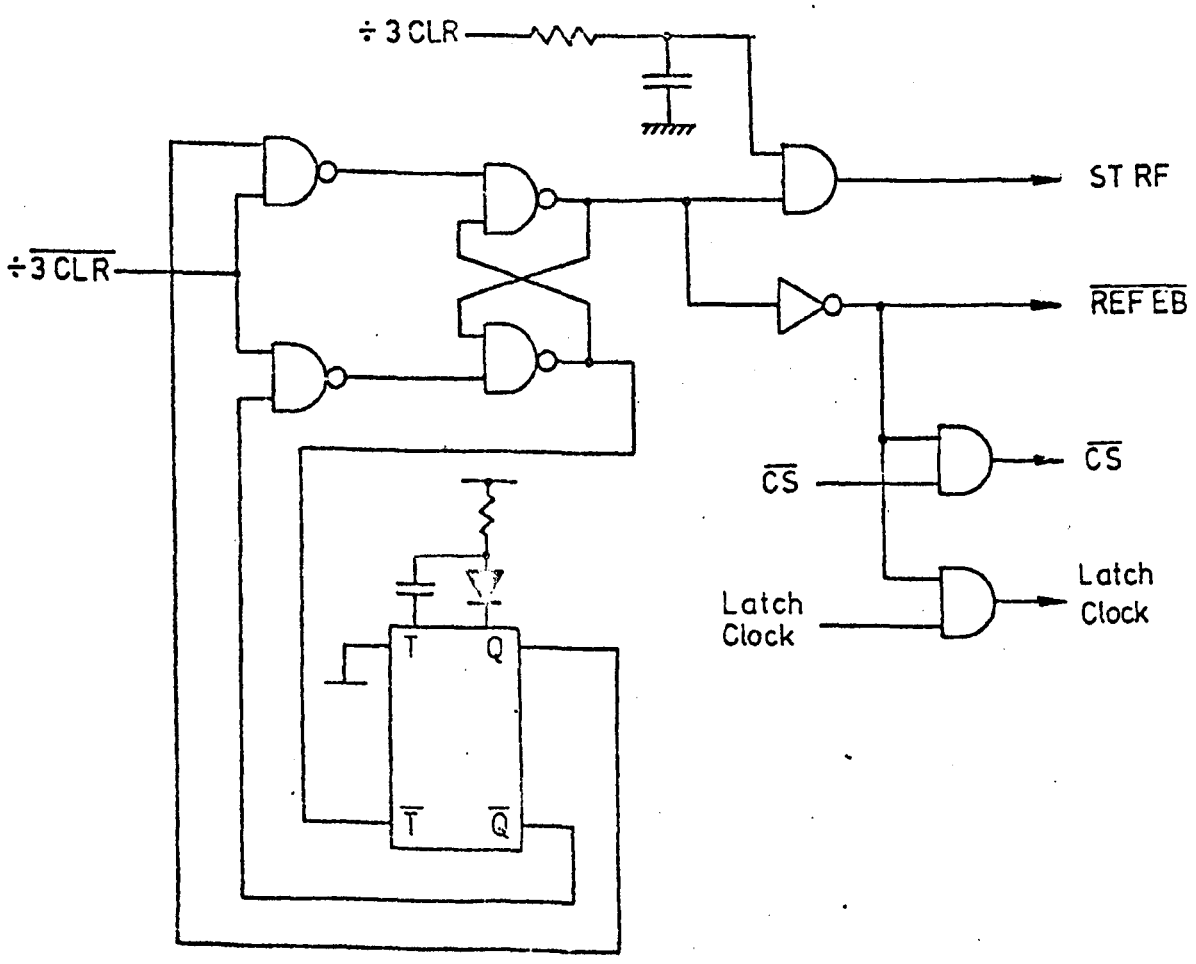


Figure 3.21 Refresh control circuit.

use of a latched input and output, the memory cycle length is independent of the cycle time of the accessing device. The clock input must not, however, be allowed to fall below 5MHz, otherwise the 2μs cycle time will be exceeded unless the clock input circuit is modified.

3.4.2 Refresh Circuits

The refresh circuits will be required for both the operating and standby modes of operation and as a result, power consumption must be kept to a minimum. The circuits involved in the refresh operation are constructed from CMOS devices, except for the drivers (see sect. 3.4.3) which use low-power schottky (LS) gates.

For the standby mode, the refresh is to be performed as one complete, sixty-four cycle operation, instead of the dispersed mode used during normal working. This is not only more convenient, but should also help to conserve power, since the memory requires considerably more power during a refresh cycle than it does during standby. If the refresh were performed as sixty-four separate cycles dispersed throughout the 2ms refresh period it would appear, to the power supply, that the memory were being turned on and off sixty-four times. If, however, the refresh is performed as one burst of sixty-four cycles, the increased power consumption during switch-on (due to capacitive effects) occurs only once.

The main element of the refresh circuit is an eight-bit binary counter of which six outputs are converted to a Grey code (see Vol.I - sect. 4.4) and taken to the six row address inputs (Fig. 3.22).

During normal operation, the refresh clock input is generated in the control section, and increments the refresh address once every 30μs (approx.). The control circuits, however, are turned off during standby, as is the system clock, and so a small low-power oscillator is used to provide

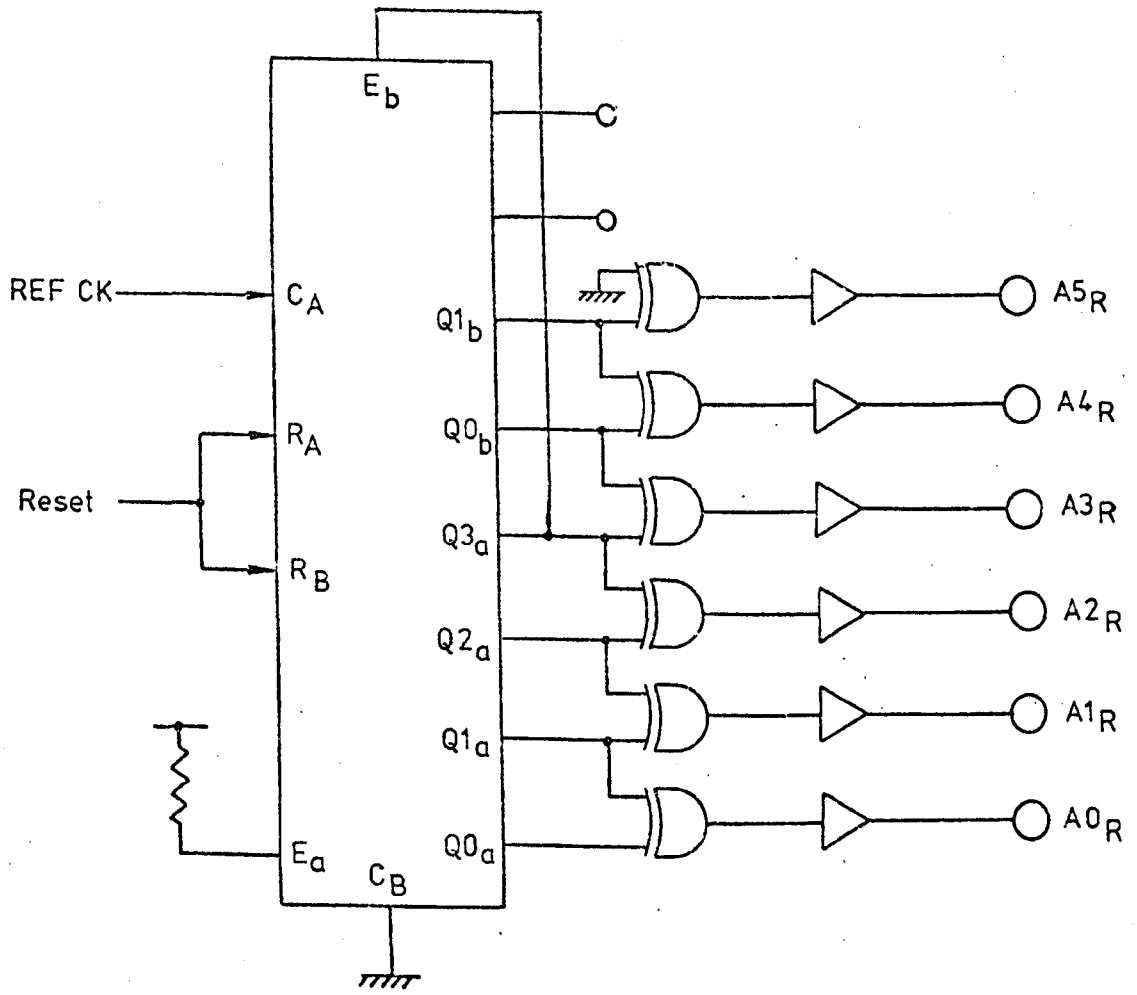


Figure 3.22 Refresh address counter with Grey code output.

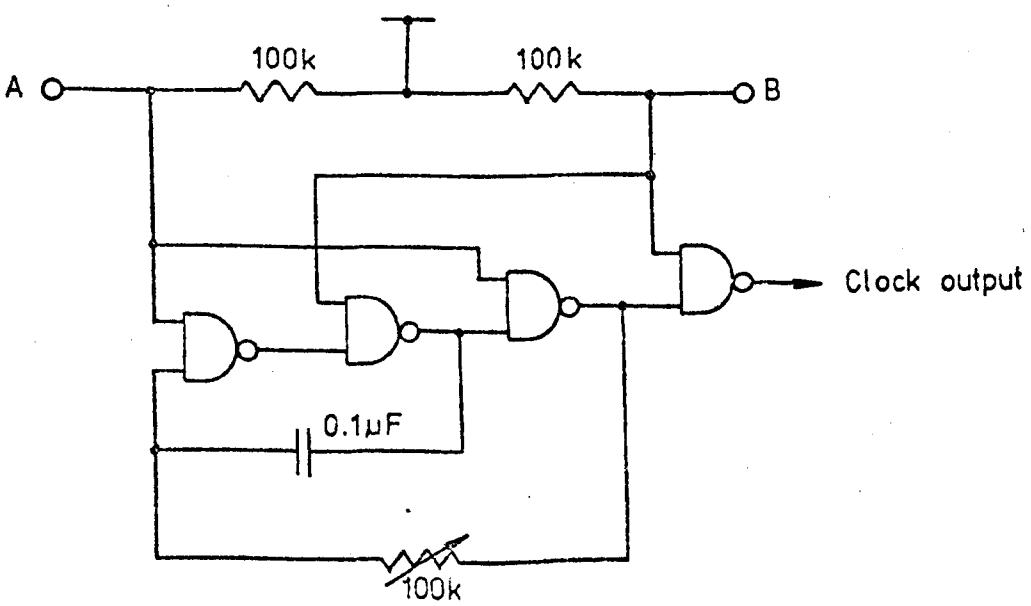


Figure 3.23 Low-power refresh oscillator.

the count input for the refresh address counter (Fig. 3.23).

The application of a logic zero to point A causes the oscillator to be halted with the output in a low state and for point B, the output is left in a high state. Hence, the circuit provides a very useful and programmable clock source.

The main chip enable - chip select generator will also be turned off during standby and so a duplicate circuit must be provided. Only chip enable is needed in the standby mode, since the RAM need not be selected during refresh, which reduces the generator to a divide-by-three circuit (Fig. 3.24).

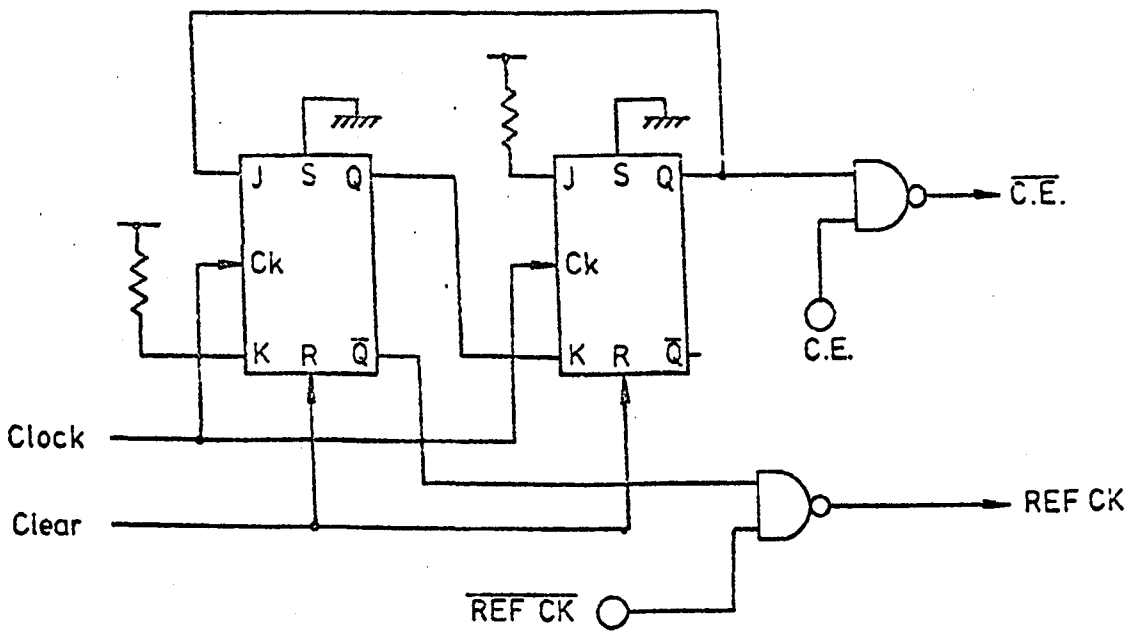


Figure 3.24 $\overline{\text{C.E.}}$ and REF CK generator circuit for standby mode.

A start-stop, or reset, input is required for both the oscillator and chip enable circuit and this can be best supplied via a retriggerable monostable circuit. The monostable is set to give a 1.9ms delay (approx.) and may be triggered from the Q_{2b} output of the refresh address counter (see Fig. 3.22). During normal operation, Q_{2b} provides the reset signal directly, the monostable being disabled, but during standby the monostable output is used (Fig. 3.25).

A problem arises with this system when the standby circuits first assume control of the refresh operation, since in general the address counter will be set to some non-zero count, Qn. If the count is merely completed and Q_{2b} allowed

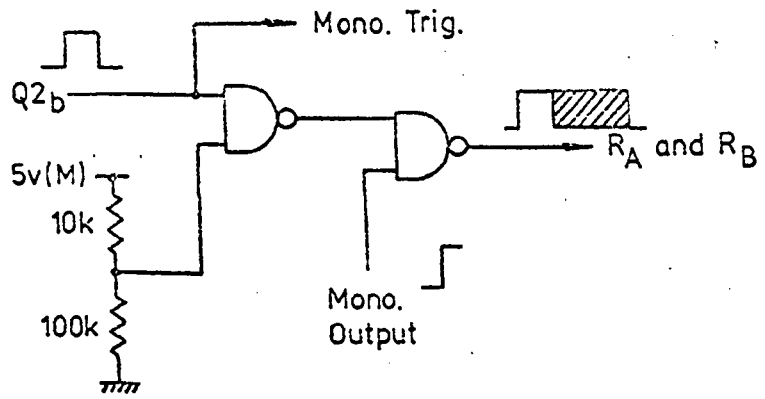


Figure 3.25 End of refresh cycle detection circuit.

to trigger the monostable, then the locations specified by addresses 0 - Q_n will not be refreshed until the end of a 1.9ms delay, which may exceed their refresh period. Hence, the first Q_{2b} output after power-down must not be allowed to trigger the monostable, i.e. the refresh operation in progress is completed and is then followed by a complete sixty-four cycle refresh. This can be achieved with the use of the circuit shown in Figure 3.26. The complete reset circuit is now as shown in Figure 3.27.

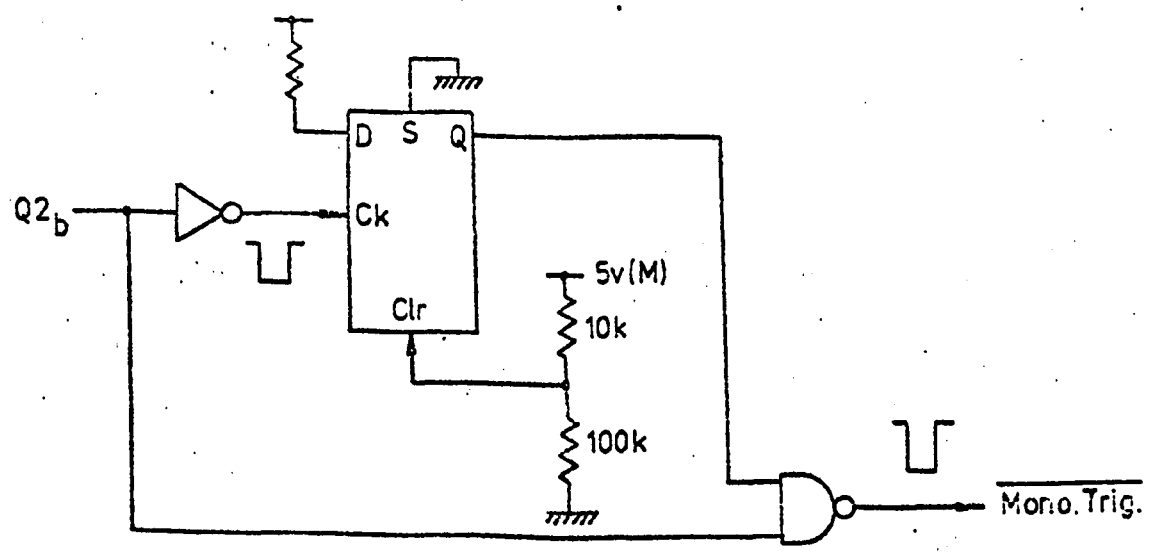


Figure 3.26 Modified monostable trigger circuit to ensure correct start-up from main power-fail.

All unused inputs to the RAMs must be pulled-up to the main and standby (M-S) supply, which can be easily achieved with the use of resistors, since the outputs of the standard logic devices, used in the control circuits, will be placed

in a high-impedance, floating state when main power is removed.

The complete standby refresh circuit is shown in Figure 3.28.

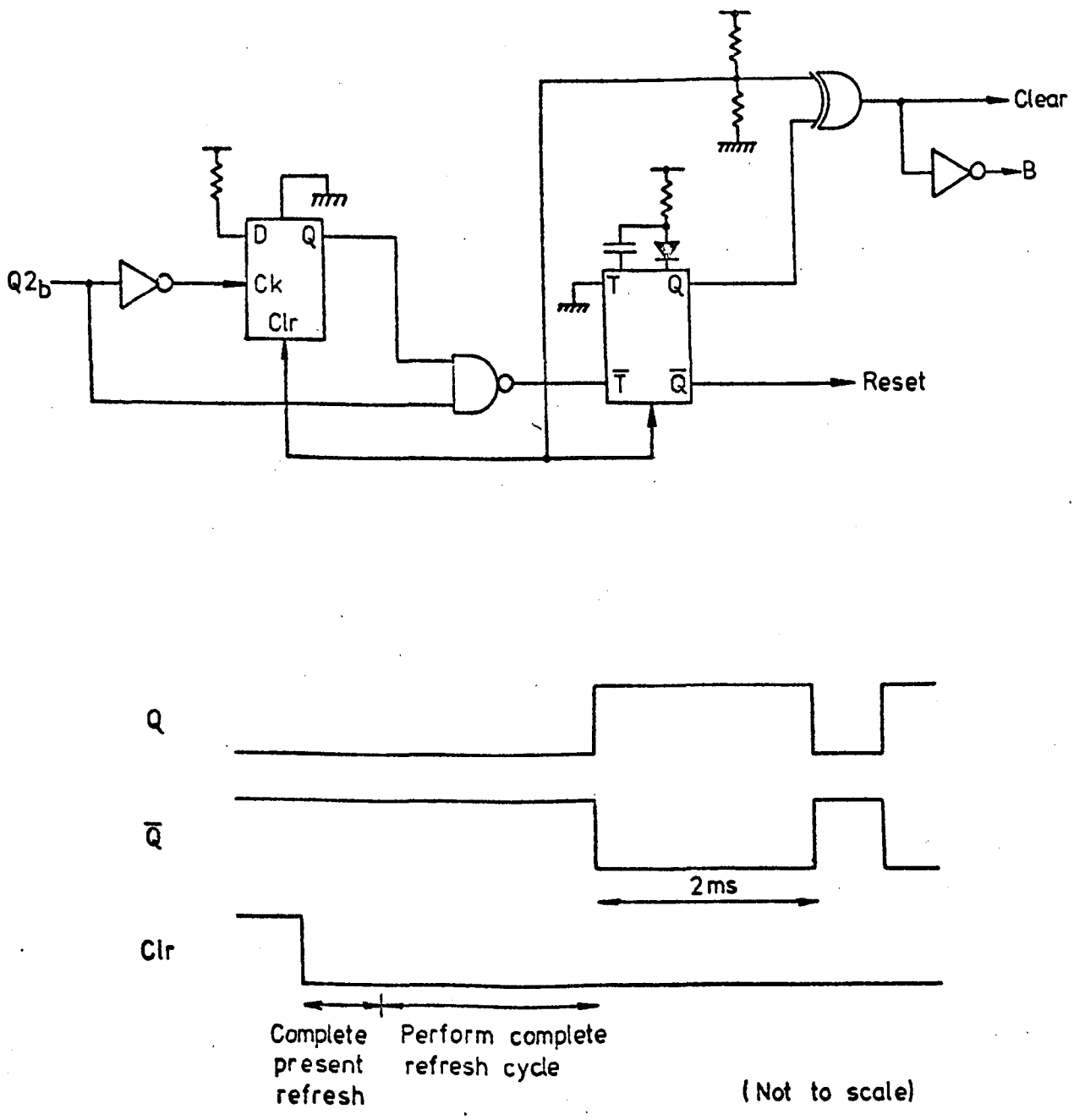


Figure 3.27 Complete refresh reset circuit.

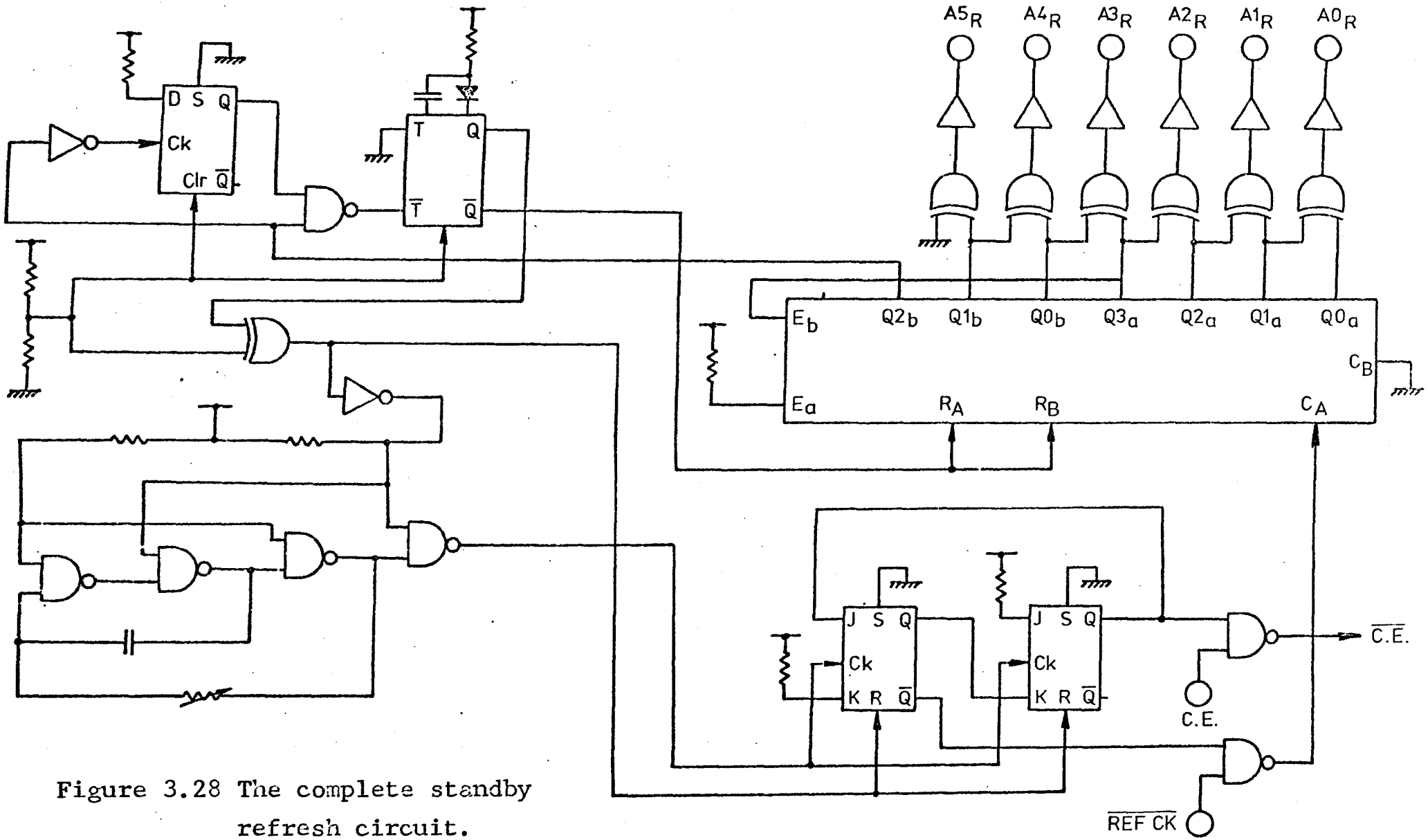


Figure 3.28 The complete standby refresh circuit.

3.4.3 Driver Circuits

Sixteen loads are placed in parallel on the input lines to the memory array and driver circuits are hence required to switch the twelve address inputs and three control lines, chip enable (C.E.), chip select ($\overline{C.S.}$) and read-write (R/W). The C.E. and six row address inputs (A0 - A5) are required in both the operating and standby modes. Further, an address multiplexer is required for each row address input so that operating and refresh addresses may be placed on the lines as required.

The six column address inputs (A6 - A11) and $\overline{C.S.}$ input drivers are implemented as single driver gates (Fig. 3.29).

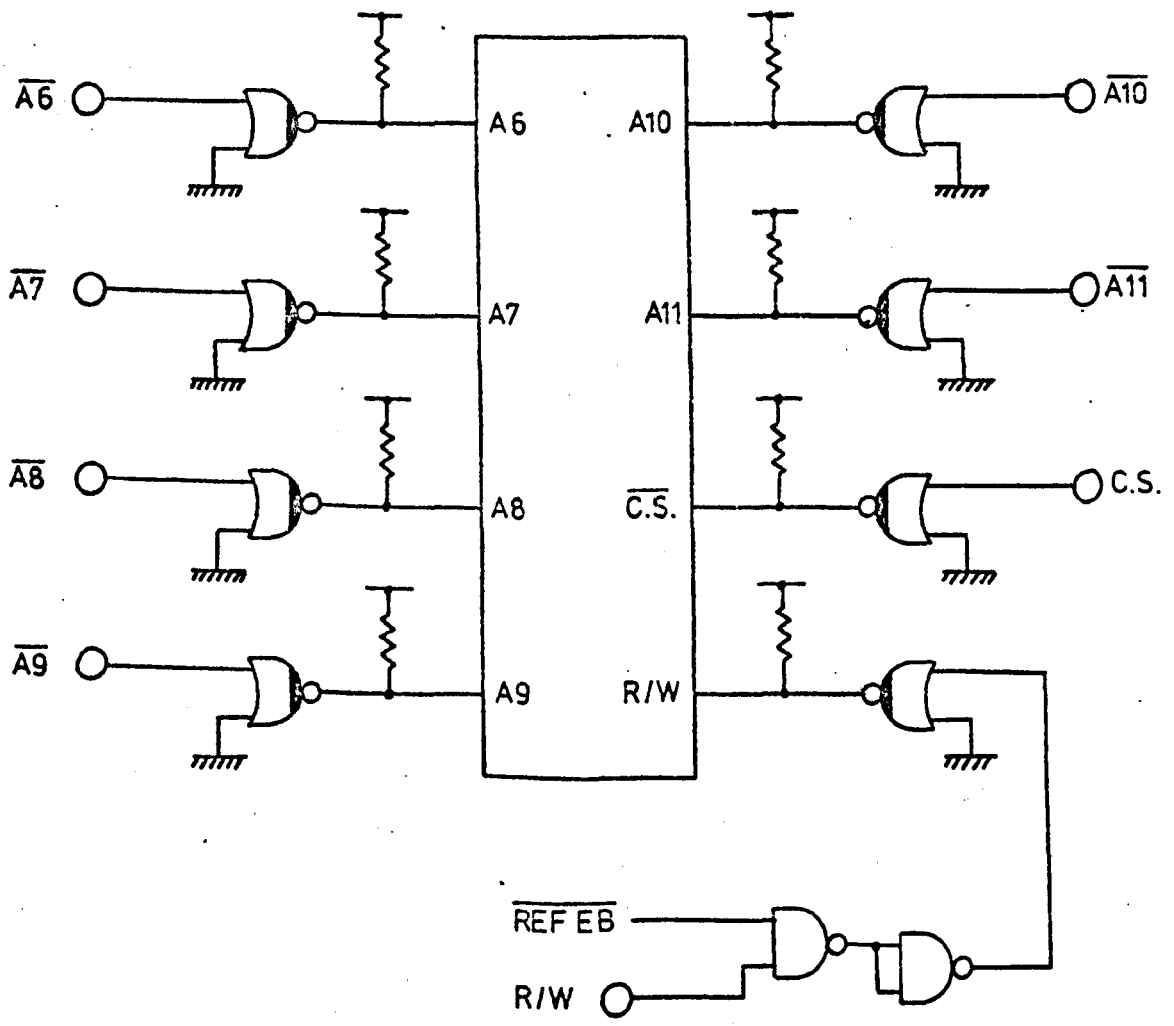


Figure 3.29 Chip select, read-write and column address line drivers.

The R/W input is similarly implemented but includes additional gates to ensure that the memory is always in the read mode when a refresh operation is in progress. All pull-up resistors for these inputs are connected to the main and standby supplies so that the inputs are pulled high during standby; this minimizes the internal power consumption of the memory chips. The driver gates are all turned off during standby, which allows their outputs to float.

The basic multiplexer-driver circuit for the row address inputs is as shown in Figure 3.30. In order to provide a lower power driver for the standby mode, without impairing the operating performance of the basic circuit (which must be kept as fast as possible) the modification shown in Figure 3.31 is used.

During standby, only the LS gates and a pull-up to V_S via a 1k resistor remain active. Diode D1 is included to prevent power drain from the V_S supply into the turned-off V_M supply. For normal operation, the output from the LS driver gate is held in the high state and diode D2 is included to prevent damage to this gate, as would occur in this mode if the main driver placed the address bus in the low state (Fig. 3.32). A germanium diode is used to minimize the forward voltage drop when the LS gate is in use.

A 120pF load was connected to the address driver output and the switching waveforms shown in Figures 3.33 and 34 obtained.

The chip enable (C.E.) input is the only non-TTL compatible input and involves the largest line capacitance load (430pF max.). A special driver circuit has been designed to overcome these problems and is as shown in Figure 3.35.

The complementary emitter-follower stage (Q2 and Q3) provide a very high capacitive drive capability and transistor Q1 provides a buffer for the LS gate. A test load of 470pF was placed on the C.E. output line and the switching waveforms shown in Figure 3.36 obtained.

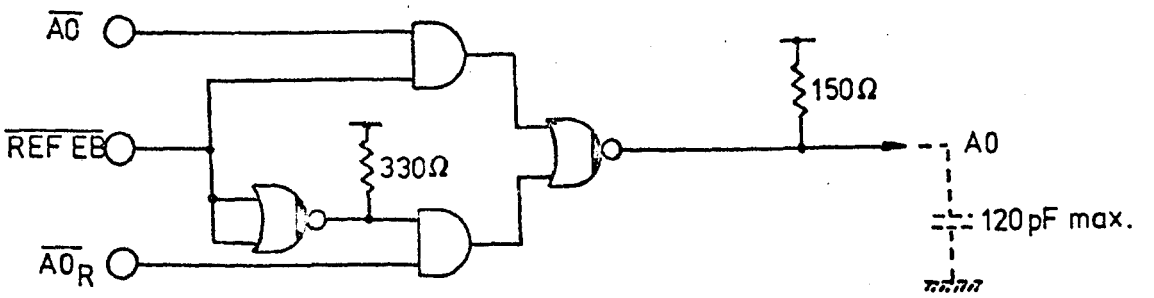


Figure 3.30 Basic row address driver.

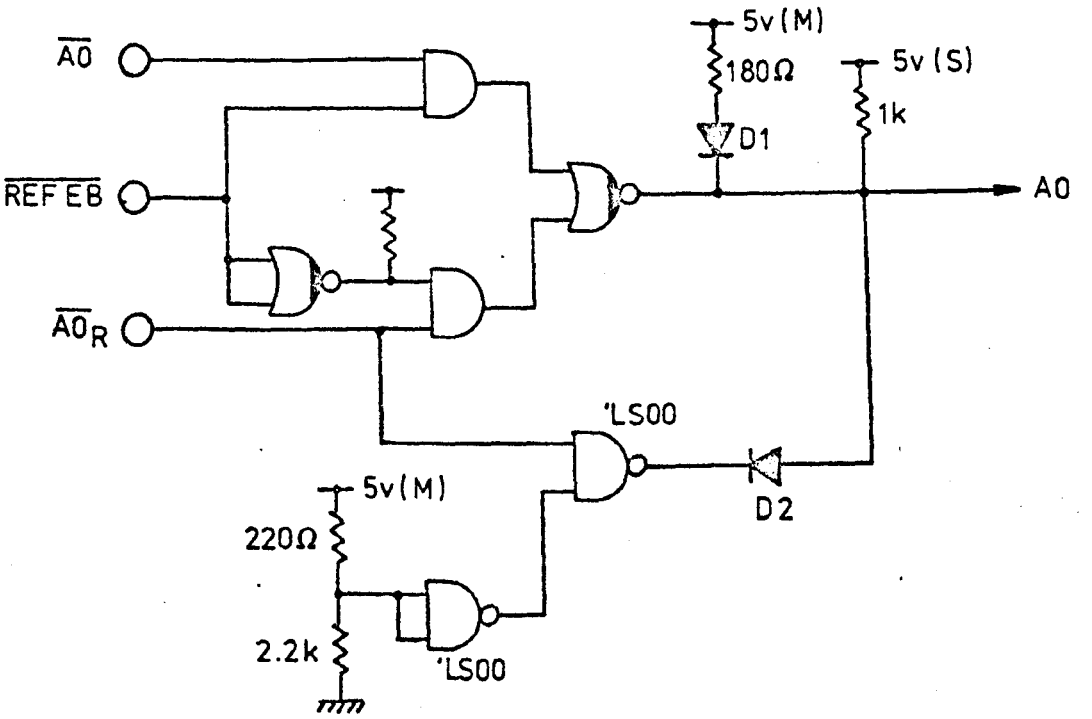


Figure 3.31 Modified row address driver.

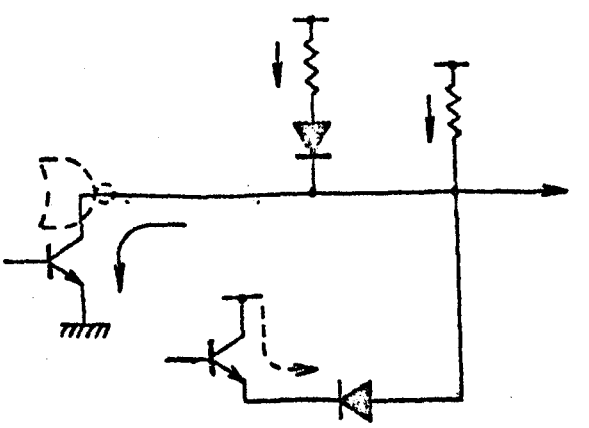


Figure 3.32 Diode protection of the low-power schottky gate.

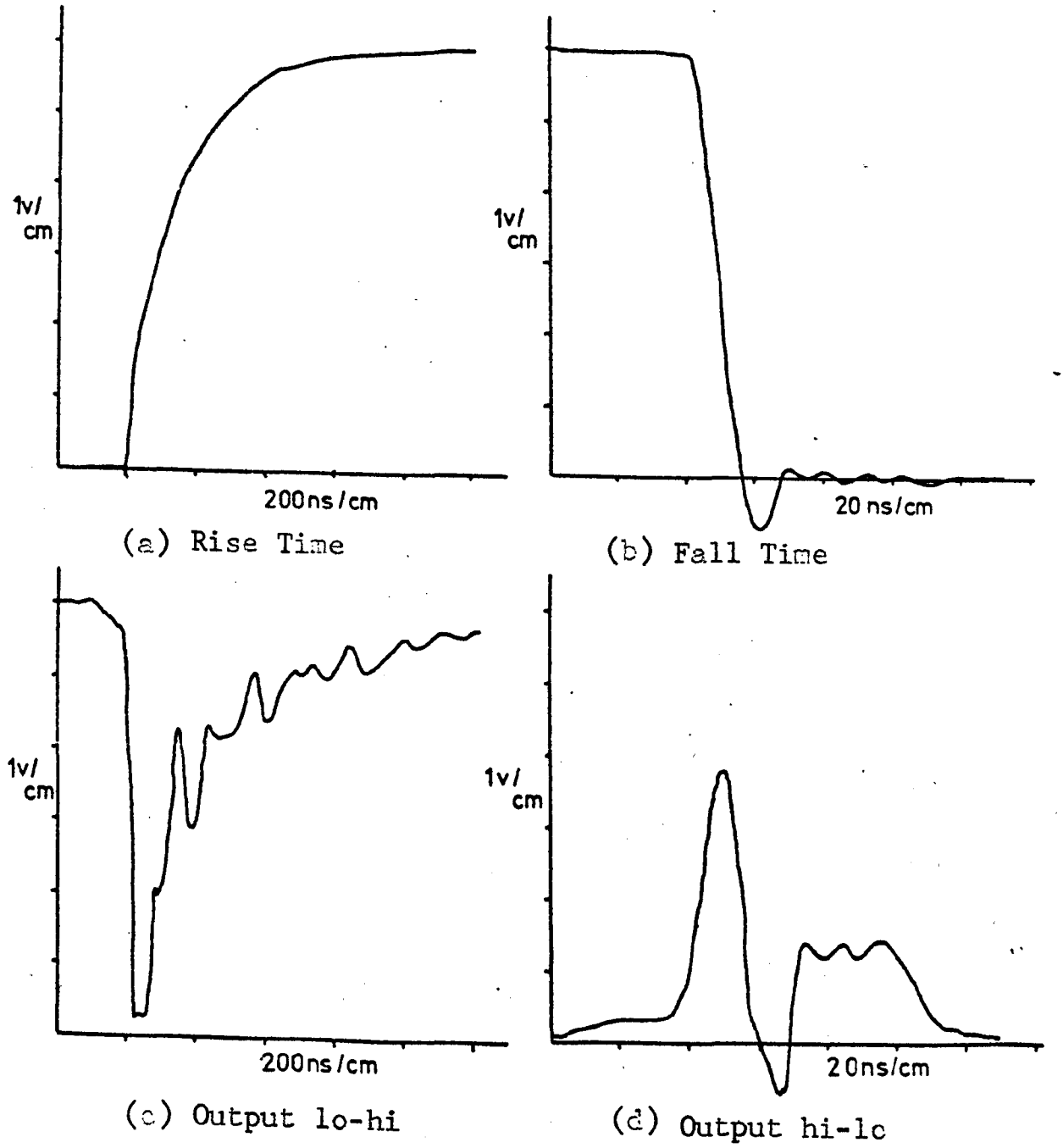
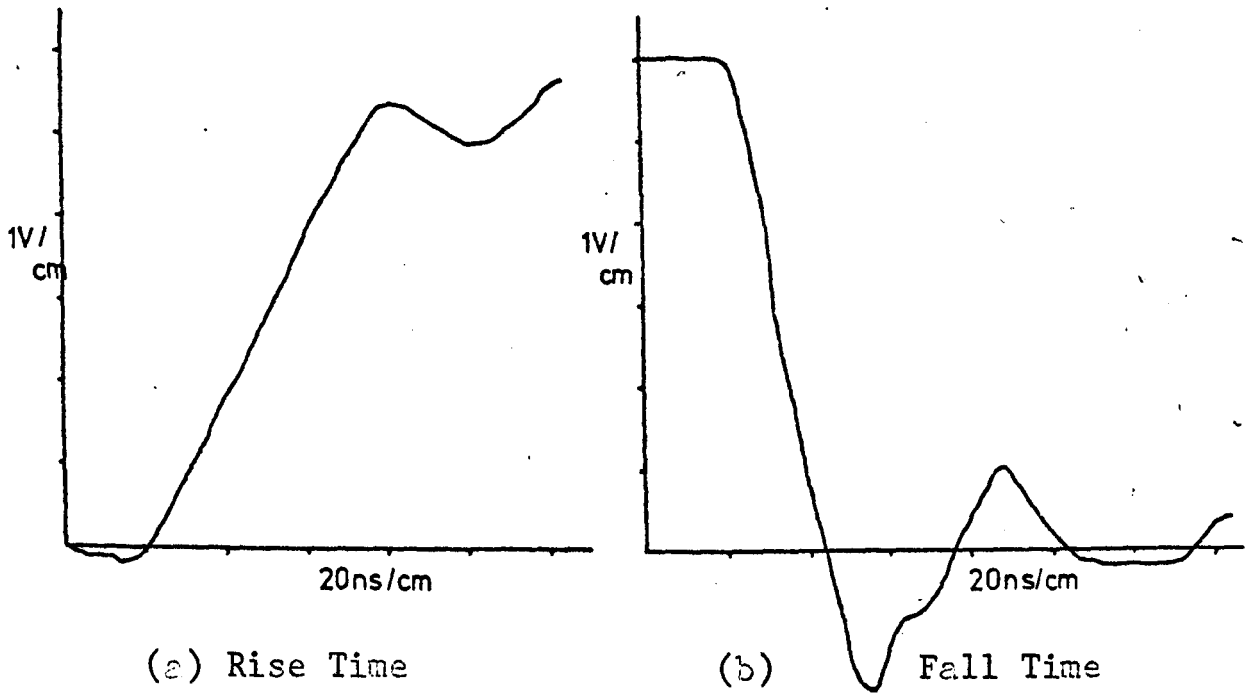
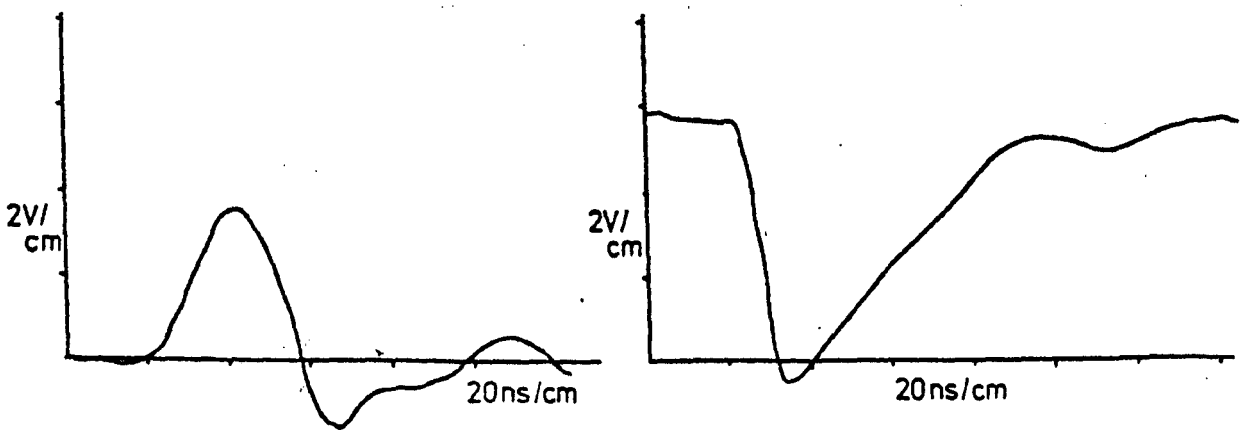


Figure 3.33. Rise and fall and input-to-output delay for the row address drivers with main power.



(a) Rise Time (b) Fall Time



(c) Output hi-lo (d) Output lo-hi

Figure 3.34. Rise and fall and input-to-output delay for the row address drivers with standby power.

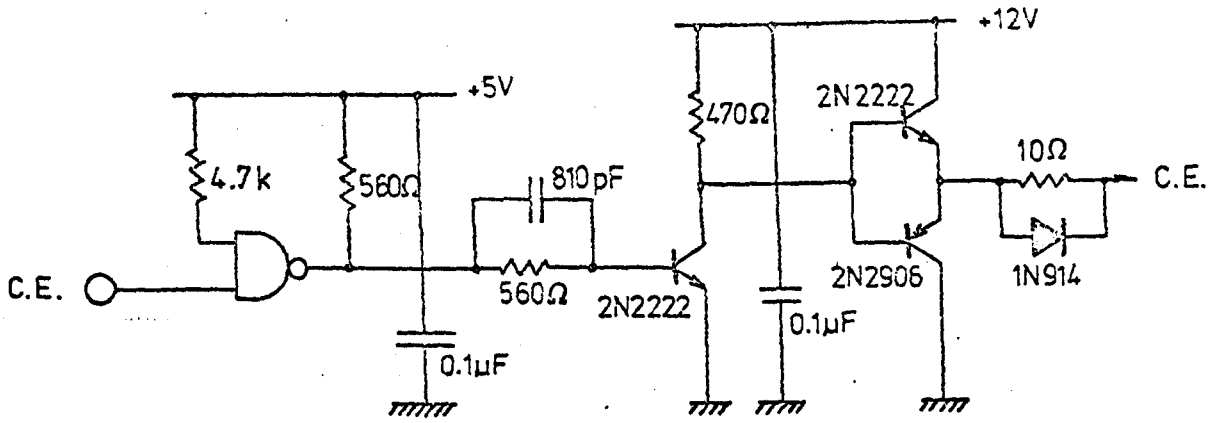


Figure 3.35 Chip enable driver circuit.

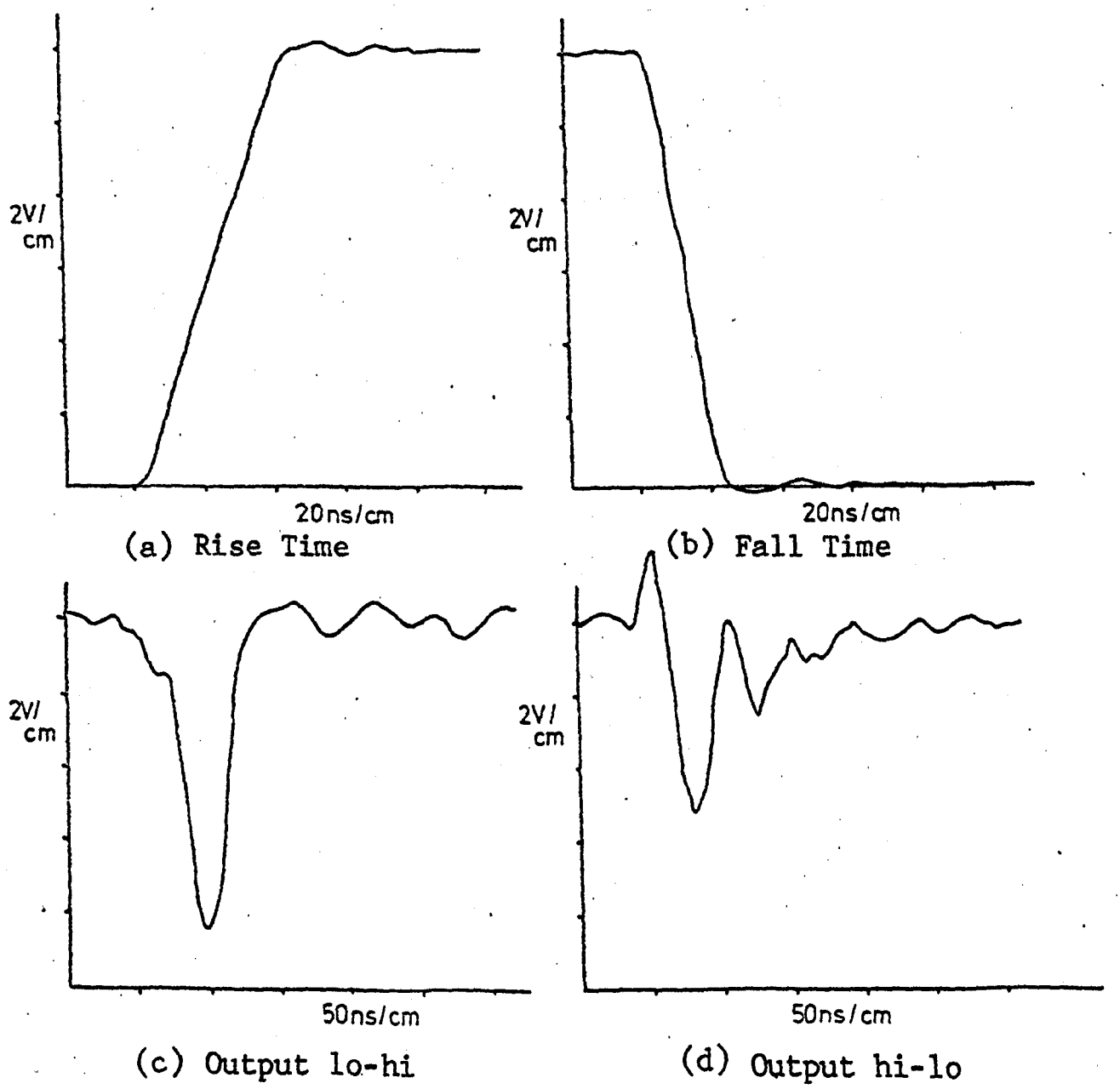


Figure 3.36 Rise and fall and input-to-output delay for the chip enable (C.E.) driver.

3.4.4 Input - Output Buffers

The TMS4030 memory package has separate data input and output lines and these are combined to give a single bi-directional data bus to the F100 microprocessor. A bi-directional latch is included in the data highway which helps eliminate any interface timing problems and offers a very short write cycle time; the write cycle time is only governed by the need to maintain the address for the duration of the chip select signal (150ns min.) since the input data are latched. The sixteen input-output stages are as shown in Figures 3.37 to 40.

3.4.5 Power Supply Considerations

The fact that some circuits are only to be powered during normal operation and then turned-off during standby and others are to be powered at all times, imposes certain problems for the layout of the power supply. To provide separate standby and main supplies requires that two supply systems are carried around the memory circuit board. The requirement for different supply voltages (+12V, +5V and -3V) further aggravates the situation and imposes considerable additional problems for the standby supply.

As discussed earlier, a battery is to be used to provide the standby supply and the problem immediately arises as to whether separate batteries should be used to provide the different supplies or one battery used to generate all the supplies. This problem has still to be resolved, but at present the single battery solution is favoured. The proposed system is as shown in Figure 3.41.

When mains power is applied, both the main and standby supplies are provided by the main power source: note that the main supply voltage levels will need to be the required output voltage plus the forward voltage drop across the isolating diodes (approx. 1V). In this mode the batteries

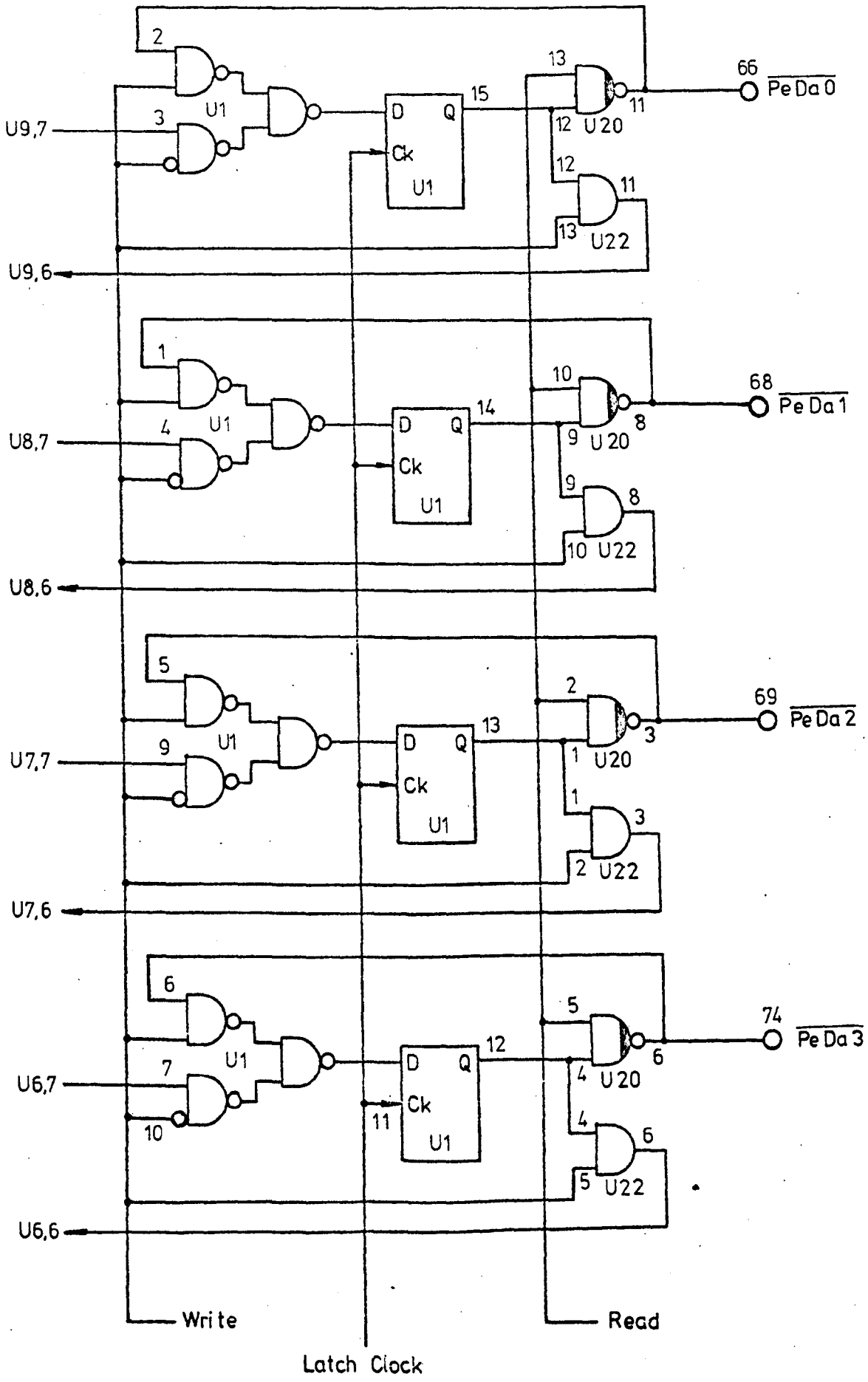


Figure 3.37 Input - Output Buffers 0 - 3.

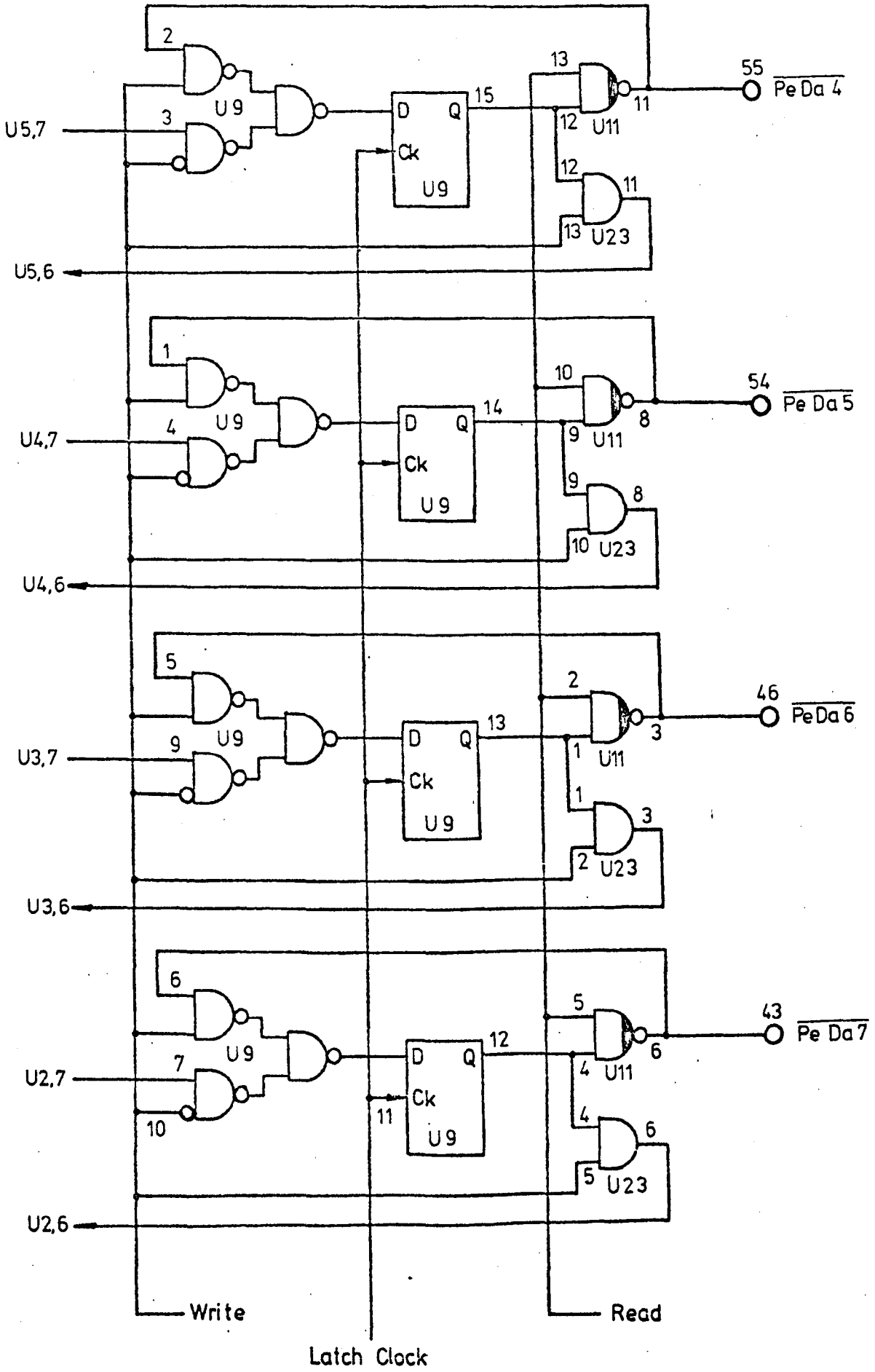


Figure 3.38 Input - Output Buffers 4 - 7.

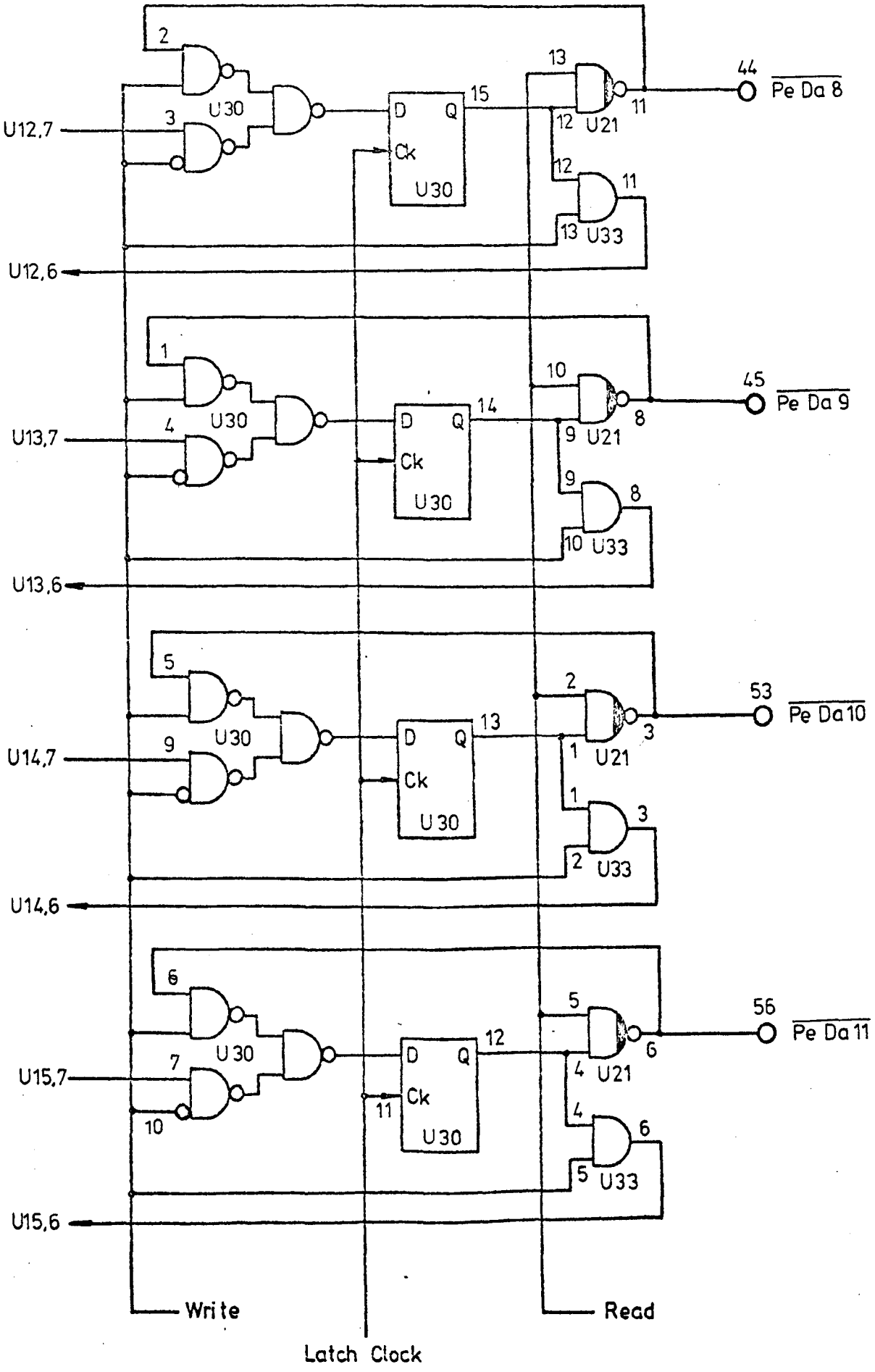


Figure 3.39 Input - Output Buffers 8 - 11.

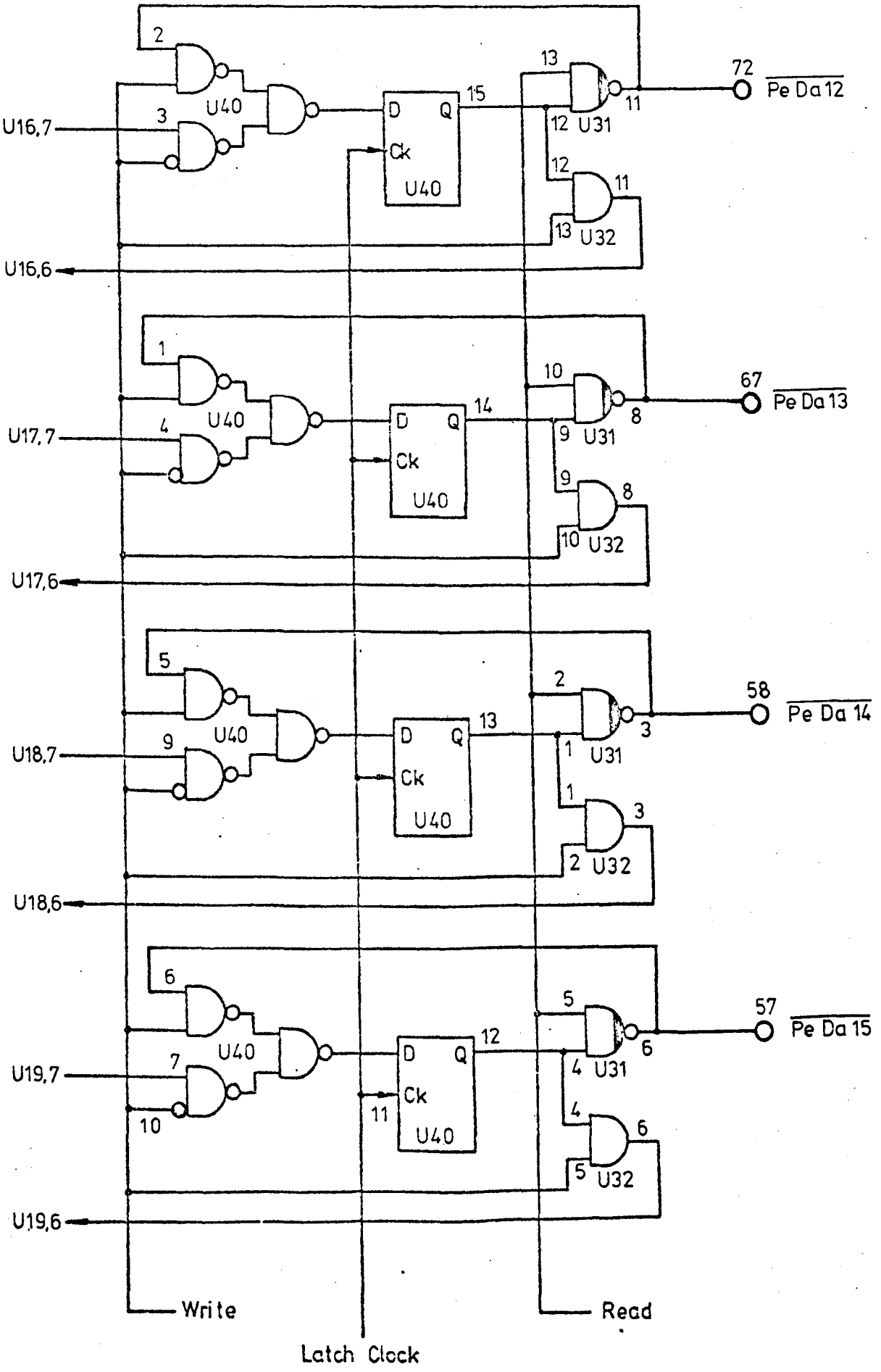


Figure 3.40 Input - Output Buffers 12 - 15.

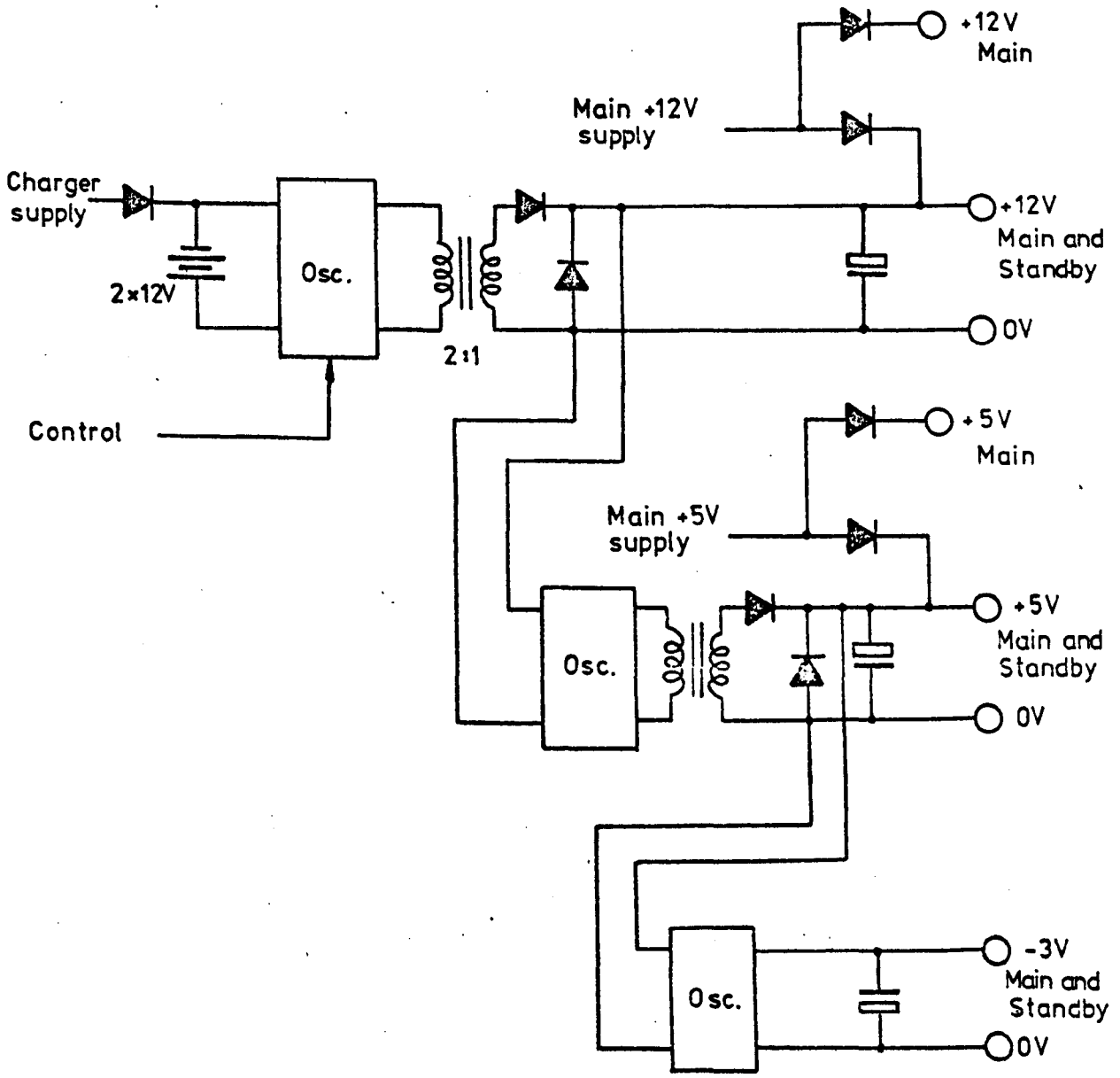


Figure 3.41 Proposed main-standby power supply circuit.

are being recharged from the main supply via the recharge input and the standby supplies are disabled by the control input, which turns off the oscillator circuit. The control signal is connected to a power-fail circuit so that when mains power is lost, the oscillator is enabled and the battery takes over automatically. If the power-fail circuit is connected to the mains input side of the main supply, a period of several milliseconds will be available in which to turn on the battery supply, owing to the charge storage afforded by the smoothing capacitors of the main supply.

The -3V supply is required as a substrate bias for the memory chips and hence very little current is required (1.6mA max.). This supply is needed for both the main and standby periods and is generated from the +5V main-standby supply. A small oscillator is constructed from four low-power open-collector gates and its output fed to a half-wave rectifier circuit (Fig. 3.42). The supply is capable of providing -3V at 2mA (approx.) and is hence sufficient for this application.

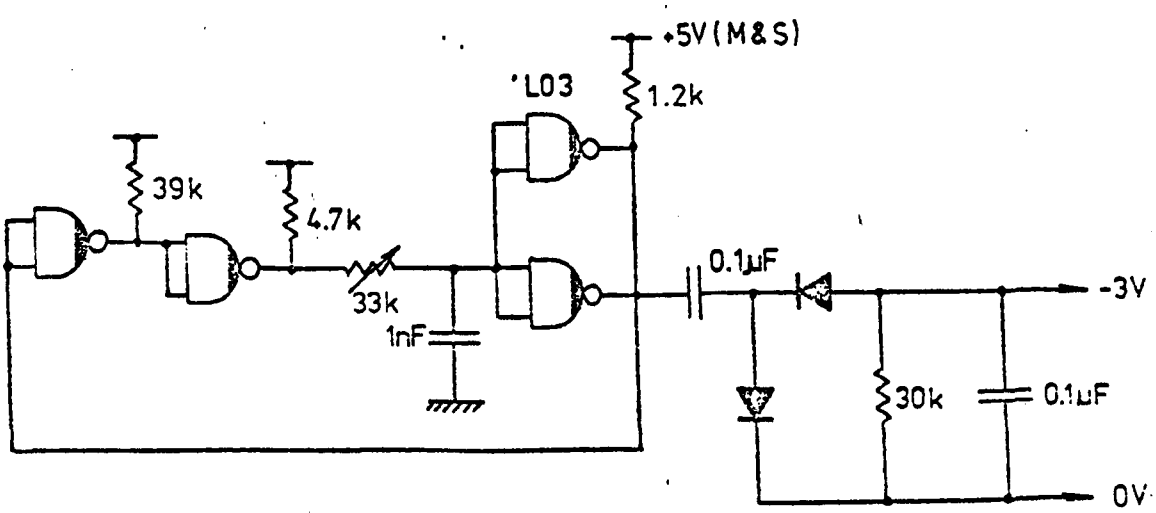


Figure 3.42 Oscillator circuit for -3V supply generation.

The other oscillator circuits are still to be designed.

The layout of supply voltage rails and decoupling capacitors is also very important for dynamic memories due to the large peak currents involved during memory access and refresh operations. The +12V supply is the major supply for

the TMS4030 memory, with +5V only required to give TTL compatibility on the input and output stages. Hence, special precautions have been taken with the +12V supply, with a large distribution bus provided on one side of the board which is connected to each of the package +12V inputs and to smaller distribution systems on the other side of the board to help minimize the length of the component leads of the decoupling capacitors. A 0.1μF ceramic capacitor is provided for each memory package, connected in close proximity to their respective packages.

An essential element of the power supply system is a power fail indicator. If, during the standby period, the standby supply falls below 11.4V, the minimum permissible supply voltage, even for a very short period, data may be lost from the store. As a result, a power fail indicator is required to inform the user of the possible loss of data during the standby period. Since the circuit is required during standby, the power consumption must be kept to a minimum.

The circuit to perform the operation consists of two functional units, an 11.4V Schmitt trigger and a set-reset flip-flop. A Schmitt trigger, with very low quiescent power requirements, can be constructed from two CMOS gates and six discrete components (Fig. 3.43). The potentiometer (47k) is set so that, if the +12V supply falls below 11.4V, the voltage at point A falls to a logic zero and gate 1 turns on. The output from gate 2 is hence taken low and the feedback resistor maintains this state until the +12V supply voltage is raised to approximately +15V, or the +5V main supply is reconnected. Diode D1 ensures that the Schmitt is always reset to a high output condition when main power is applied.

The output from the Schmitt is taken to a 'biased' set-reset circuit, again constructed from CMOS gates to minimize quiescent power requirements (Fig. 3.44). Only the set-reset

flip-flop is on during standby, the power fail indicator being powered from the main supply. This reduces the power consumption of the circuit to an absolute minimum since the CMOS circuits consume very little power when not switching: the circuit will switch only once at most during a standby period.

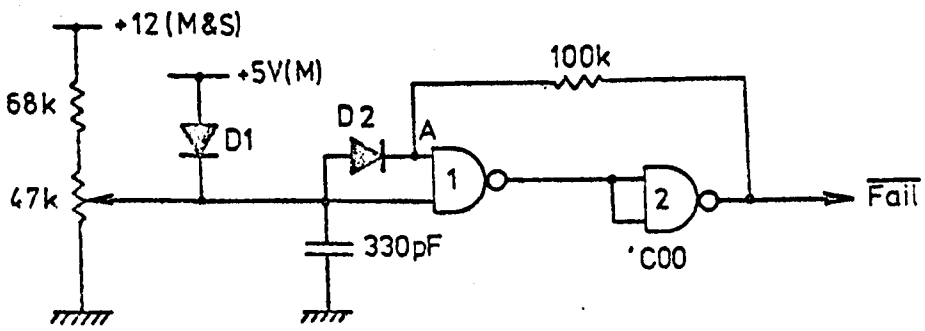


Figure 3.43 Low power Schmitt trigger circuit.

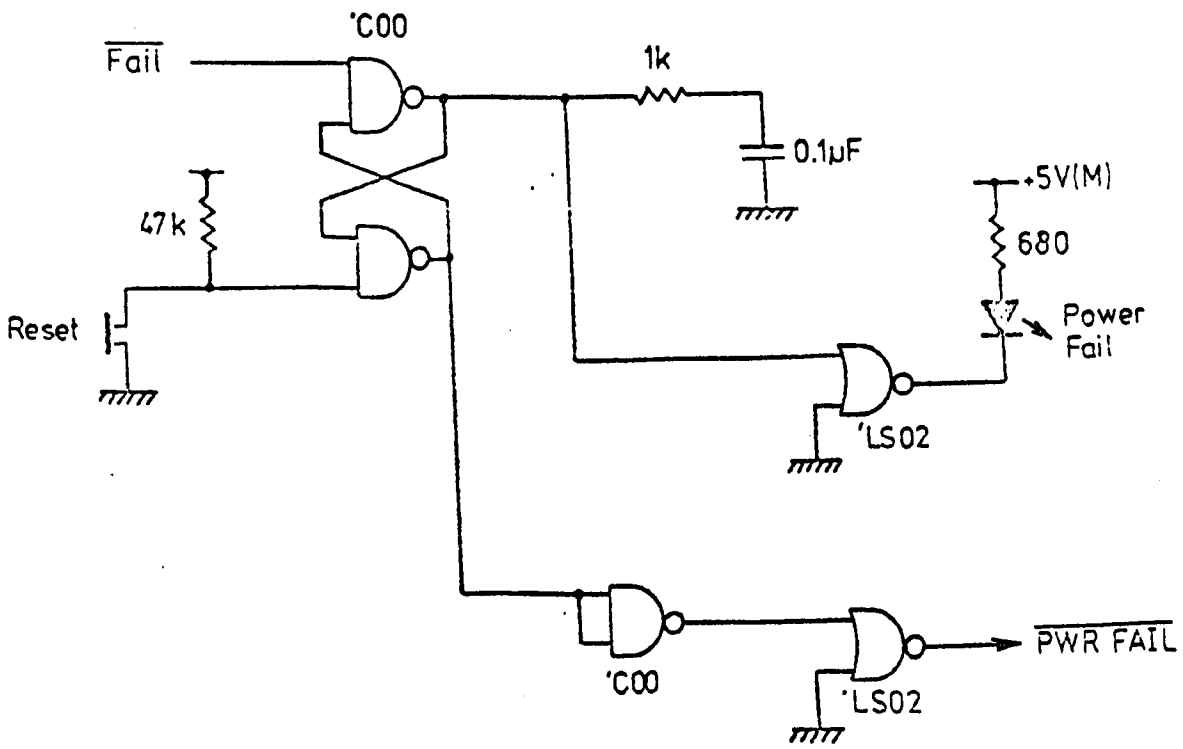


Figure 3.44 Power fail latch and indicator circuit.

If, at any time during standby, the +12V supply has fallen below 11.4V, the Schmitt sets the flip-flop which will remain set until an external reset is applied. When

main power is restored, the LED will be turned on to indicate the occurrence of the power fail. The RC load (bias) placed on the flip-flop ensures that even if the standby power fails completely and the power fail circuits are turned off, when the main supply is restored a true power fail indication will be given.

An output is also taken from the flip-flop to the control circuits to prevent use of the memory until reset is issued. Note this is not the main reset but is provided by a micro-switch, mounted on the RAM board. This configuration ensures that the user cannot fail to notice that a power fail has occurred.

3.5 Miscellaneous Details

The RAM circuits have been constructed on two circuit boards, designed specifically for the purpose. The RAM chips, input-output buffers, address drivers and multiplexers, C.E., C.S. and R/W drivers are all contained on the main circuit board with the control and refresh circuits constructed on a smaller 'pick-a-back' board. The pick-a-back board is attached to the main board by four 6B.A. fixing screws and insulating stand-off spacers. The layout of the components on these two boards is as shown in Figures 3.45 and 46. Note, the circuit design has not been finalized and so the layouts given are subject to possible alterations, particularly on the pick-a-back board (Fig. 3.46).

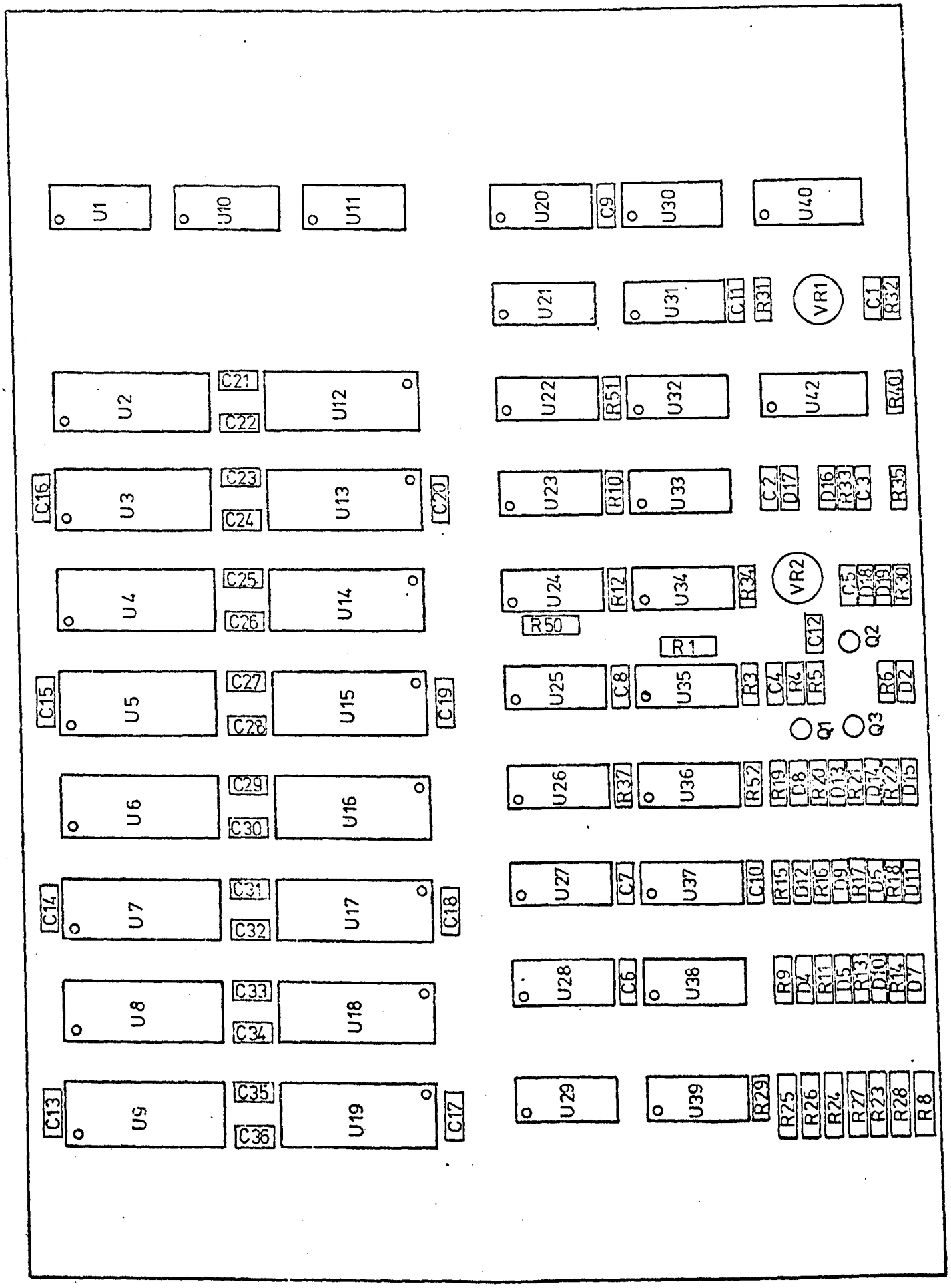


Figure 3.45 Main circuit board layout for the 4k x 16 non-volatile RAM.

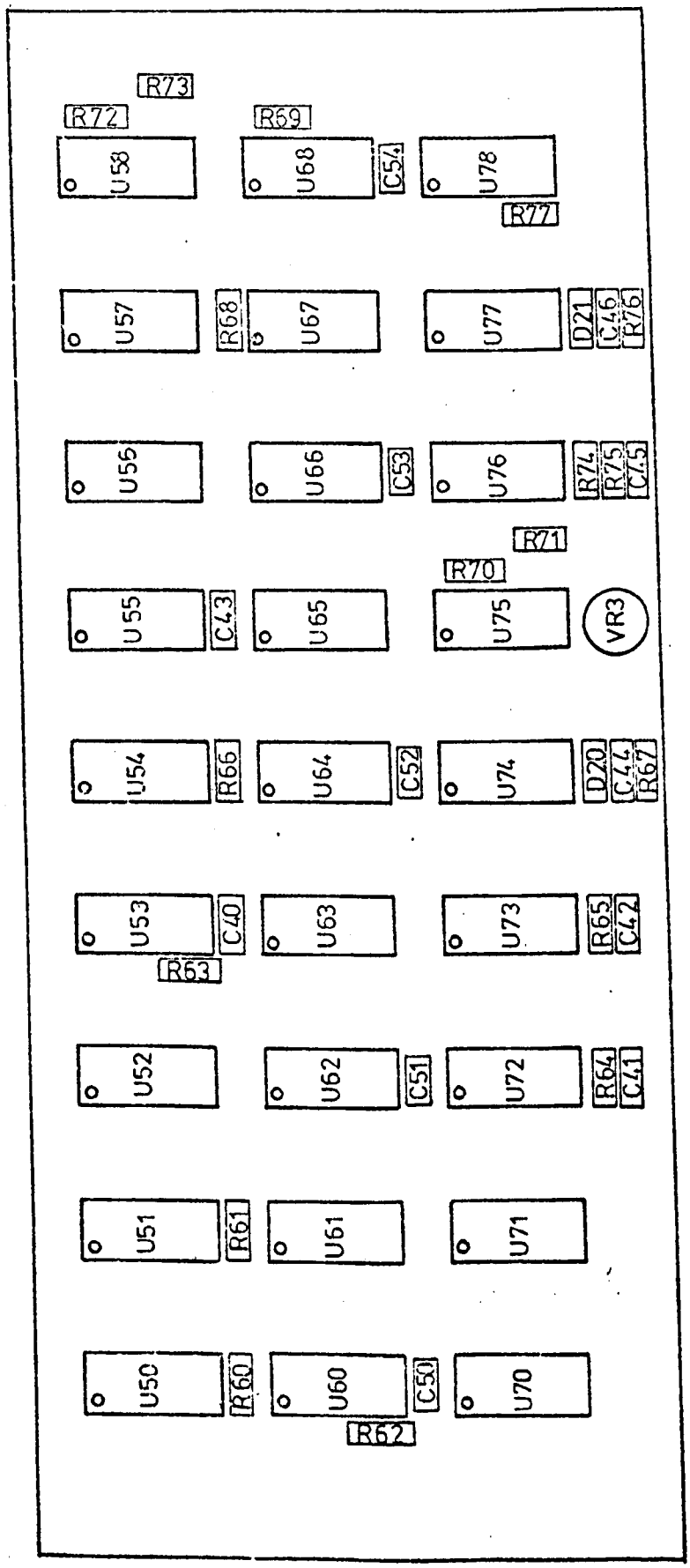


Figure 3.46 Pick-a-back board layout for the 4k × 16 non-volatile RAM.

3.5.1 Component list

Resistors

All ¼w 5%

R1,R11,R14,R16,R18,R20,R22,R37	1k
R3,R4,R34	680Ω
R5,R52	470Ω
R6	10Ω
R8,R23 - R29,R50,R51	150Ω
R9,R13,R15,R17,R19,R21	180Ω
R10,R63,R64,R65,R66	220Ω
R30,R68,R69,R70,R72,R77	10k
R31,R35,R60,R61,R62	4.7k
R32	1.2k
R33	33k
R40	39k
R67,R76	20k
R71,R73,R74,R75	100k
VR1,VR2	4.7k preset
VR3	100k preset

Capacitors

C1,C5,C41,C43	220pF ceramic
C2,C3,C6 - C36,C40,C45,C50 - C54	0.1μF ceramic
C4	1000pF ceramic
C44	4.7nF ceramic
C46	0.33μF tantalum

Semiconductors

D2,D4,D6,D8,D10,D12,D14,D16,D19 - D21	1N914 Silicon diode
D5,D7,D9,D11,D13,D15	OA47 Germanium diode
Q1,Q2	2N2222
Q3	2N2906

Integrated circuits

U1,U10,U30,U40	SN74298
U2 - U9,U12 - U19	TMS4030
U11,U20,U21,U24,U31	SN7403
U22,U23,U28,U32,U33,U36,U38,U55,U64	SN7408
U25	SN74LS02
U26,U27,U29,U39	SN7433
U34,U66,U75,U76	MC4011B
U35,U37	SN74LS00
U42	SN74L03
U50,U51	SN7474
U52,U54,U62,U63,U65,U71,U72	SN7400
U53	SN7486
U56	MC4010B
U57,U58	MC4030B
U60	SN7473
U61	SN7410
U67	MC4520B
U68	MC4027B
U70,U73	SN7404
U74	SN74123
U77	MC4528B
U78	MC4013B

3.5.2 Connection lists

a) F100M Edge Connector.

	1	2	
	3	4	
	5	6	
	7	8	
	9	10	
0V	11	12	
+5V	13	14	
	15	16	
	17	18	
	19	20	
	21	22	
0V	23	24	
+5V (S)	25	26	
	27	28	
	29	30	
	31	32	
	33	34	
0V	35	36	
+5V (S)	37	38	
	39	40	
	41	42	
$\overline{\text{Pe Da 7}}$	43	44	$\overline{\text{Pe Da 8}}$
$\overline{\text{Pe Da 9}}$	45	46	$\overline{\text{Pe Da 6}}$
0V	47	48	St Ad 4
+5V	49	50	St Ad 5
St Ad 11	51	52	St Ad 10
$\overline{\text{Pe Da 10}}$	53	54	$\overline{\text{Pe Da 5}}$
$\overline{\text{Pe Da 4}}$	55	56	$\overline{\text{Pe Da 11}}$
$\overline{\text{Pe Da 15}}$	57	58	$\overline{\text{Pe Da 14}}$
0V	59	60	
+12V	61	62	St Ad 1
St Ad 6	63	64	St Ad 0
St Ad 7	65	66	$\overline{\text{Pe Da 0}}$
$\overline{\text{Pe Da 13}}$	67	68	$\overline{\text{Pe Da 1}}$
$\overline{\text{Pe Da 2}}$	69	70	
0V	71	72	$\overline{\text{Pe Da 12}}$
+12V	73	74	$\overline{\text{Pe Da 3}}$
St Ad 3	75	76	St Ad 2
St Ad 8	77	78	St Ad 9
	79	80	

b) Pick-a-back board connector.

	1	2	$\overline{RD}$
$\overline{A_{0R}}$	3	4	$\overline{WR}$
$\overline{A_{1R}}$	5	6	$\overline{REF\ EB}$
$\overline{A_{2R}}$	7	8	R/W
0V	9	10	LATCH CK
+5V(M)	11	12	$\overline{FAIL}$
$\overline{A_{3R}}$	13	14	
$\overline{A_{4R}}$	15	16	$\overline{C.S.}$
$\overline{A_{5R}}$	17	18	
0V	19	20	$\overline{JDev}$
+5V(S)	21	22	$\overline{KDev}$
	23	24	
	25	26	C.E.
0V	27	28	
SYSTEM CK	29	30	$\overline{RESET}$
	31	32	$\overline{BOARD\ SEL}$
	33	34	$\overline{PWR\ FAIL}$

3.6 References

1. Two-port memory interface, Private communication, R. Young, The City University.

Chapter 4

High-Level Definer

1. Introduction
2. The High-Level Definer
3. Library Update Routine
4. Module Preparation
5. Module Generation
6. Module Library
7. References

4.1 Introduction

The programs to be described are all designed to run on a Ferranti FM1600B computer via the SC Interface [1] under the control of the 'DRUID' operating system [2].

Only a very rudimentary knowledge of the operating system is required to run the High-Level Definer. The preparation of modules, however, will require a greater understanding of the operating system and a knowledge of the programming language 'FIXPAC' [3] will also be necessary.

At present, the system is designed to produce assembly language programs for the CP1600 microprocessor but only very minor changes are required in the High-Level Definer code to make it suitable for any other assembly language. The other routines are all completely general purpose, except of course the module library.

4.2 The High-Level Definer

4.2.1 Program description

The High-Level Definer is a two-pass program designed to simplify the task of producing efficient real-time programs. The basis of the system is a backing library which contains 'standard modules' available to the user via 'high-level calls' [5]. Parameters may also be specified in the calls and these serve to specify the 'segments' of the module required by that call.

The user writes his program in the assembly language of the machine on which it is to run and may insert high-level calls within this program which are then decoded and an appropriate block of code added from the backing library, held on disc.

The standard blocks may be called via two modes, in one the module name is prefixed by an @ character and in the other by a # character. These must appear as the first

character in the line, leading spaces are not permissible. Further, if parameters are specified these must be contained within brackets, e.g. (U,8), and separated from the module name by at least one space. Each parameter in the string should be separated by a comma.

An @ prefix will cause the High-Level Definer to insert the requested module as a macro. A # prefix will cause the requested module to be inserted as either a macro or a subroutine, the distinction being resolved by the number of calls made to a particular module from one master program. Note, different parameters may be specified in these calls and it is the responsibility of the control segment of the module to decode them and then to generate a suitable subroutine. The checking of parameters is also the responsibility of the module's control segment (see section 4.4 - Module preparation).

In pass 1 all high-level calls are resolved and, if prefixed with a #, copied complete with parameters to a data area (S4400) reserved for the call table. This is performed by the definer, which reads the first character of each line and checks for a # or @ character. If neither of these appears then the reader is advanced to the next line (by a read to <NL>).

If a # character is found then the module name and parameters are copied into the call table. The format of the record in this table is:-

NAME <SP><P1,P2,.....Pn><NL> ...

Syntax checking is performed on the high-level calls during this pass, to check for correct formatting.

When pass 1 has been completed a prompt of STOP 8 is output. The user program should then be reloaded and RUN entered via the teletype. This will cause pass 2 to commence.

If an error is encountered, one of the following stops will be generated:-

STOP 20 - (error

STOP 21 - terminator error

STOP 22 -) error

During pass 2, the user program is again scanned for high-level calls. This, as pass 1, works on a line basis. If the first character of a line is an @, the module name and any parameters are copied to a temporary data area. The module library is then searched to find the requested module which is output immediately, in place of the high-level call. If the first character is a #, the module name and parameters are again copied to the temporary storage area. Now, however, the call table, set up in pass 1, is searched and a flag number set against the requested module name each time it is encountered. When the end of the call table is reached the number of calls made to the module is checked and, if more than one, a subroutine call to the specified module name is inserted into the output code in place of the high-level call. The Definer then proceeds to the next instruction in the source program. If only one call has been made to the module it is output as a macro.

When the end of the user program is reached the call table is scanned and any outstanding subroutines are output. Blank tape is output between each subroutine.

When all of the code has been output, a STOP 12 prompt is given on stream 5. If the High-Level Definer is required for use again, the user must 'log-out' of the system and then 'log-in' again. This is necessary to close the disc files opened during the previous run before they are re-opened in the next. If this operation is not performed an error stop (STOP 10 21 29*) will occur at the start of pass 2.

If a requested module is not found in the module library its name is output and the run abandoned ending with a STOP 7.

4.2.2 Modification for other languages

The program listings given are for the High-Level Definer as used with the CP1600 assembly language. To modify the program for use with any other language, three small changes are required. These are all changes to text type statements and do not involve any logical alterations.

The first is the text required for a subroutine call. For the CP1600 the following format is required:-

```
JSR R5,<name><CR><NL>
```

and is delt with by the block of code:-

```
[55] V19=2
     TEXT
     JSR R5,
     N2=VNO
     AL 100
[32] V20=VN2,N2+1
     ←31,V20=VNO=0
     +160
     ←S555,0
     ←32
[31] TEXT 1E
     <CR>
```

The other text associated with subroutine calls is the header to the subroutine code. These are specified as relocatable and a text output of:-

```
REL <name><CR><NL>
```

is output. The code for this is:-

```
[50] ...
     .
     ←57,L
     REL<SP>
     .
[37] V20=VNO
     +141
```

```
←S555,0          'OUTPUT <CR>'
 V20=VNO
 +10
←S555,0          'OUTPUT <NL>'
 .
```

The final parts of the program that will require modification are the sections that add terminating characters. For CP1600 assembly code a <FF> character is output at the end of the master program and at the end of each subroutine. The code used for these terminators is:-

```
[23] V19=2
     V20=VNO
     +12
    ←S555,0          'OUTPUT <FF> CHARA'
```

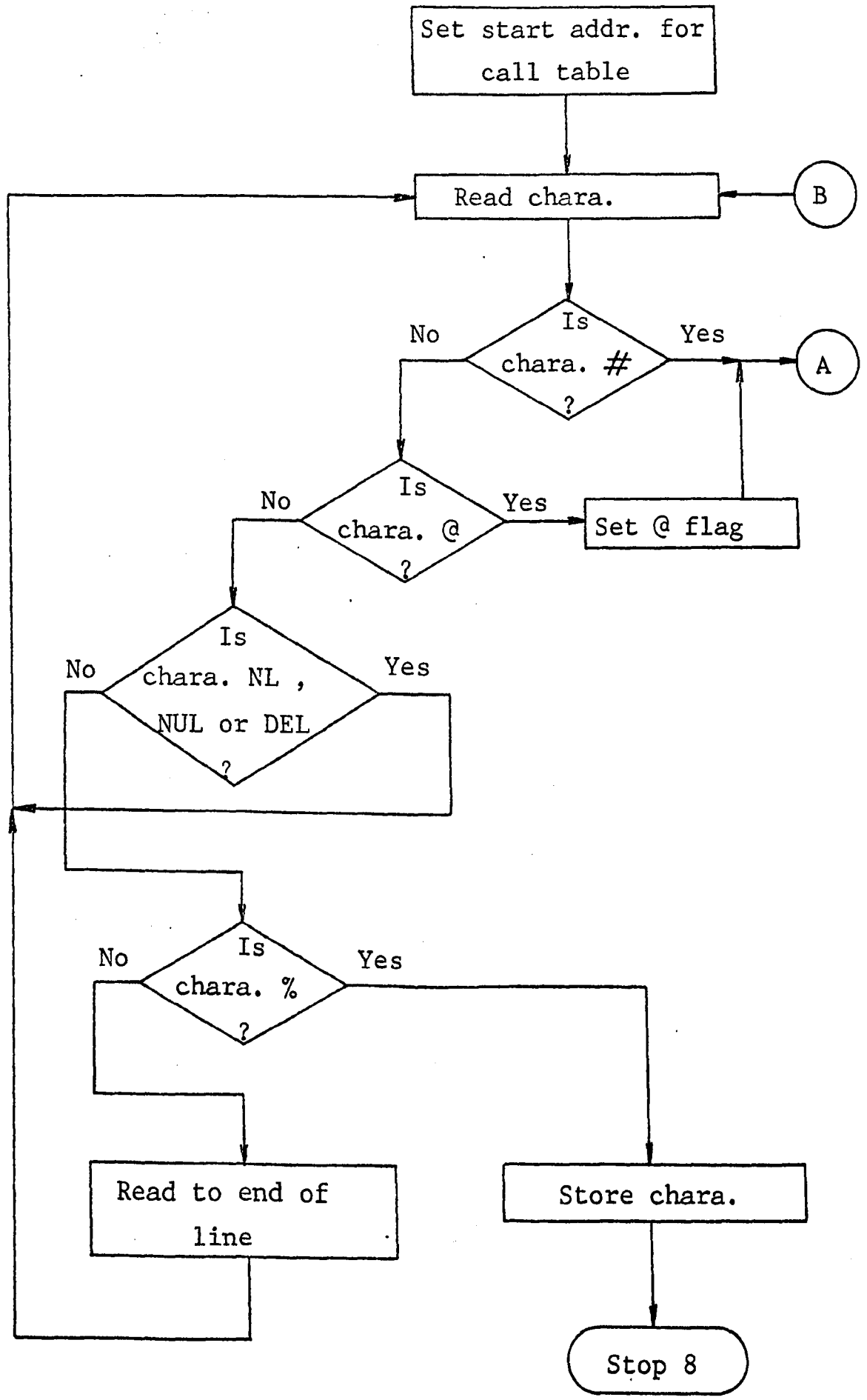
for the end of the master program and:-

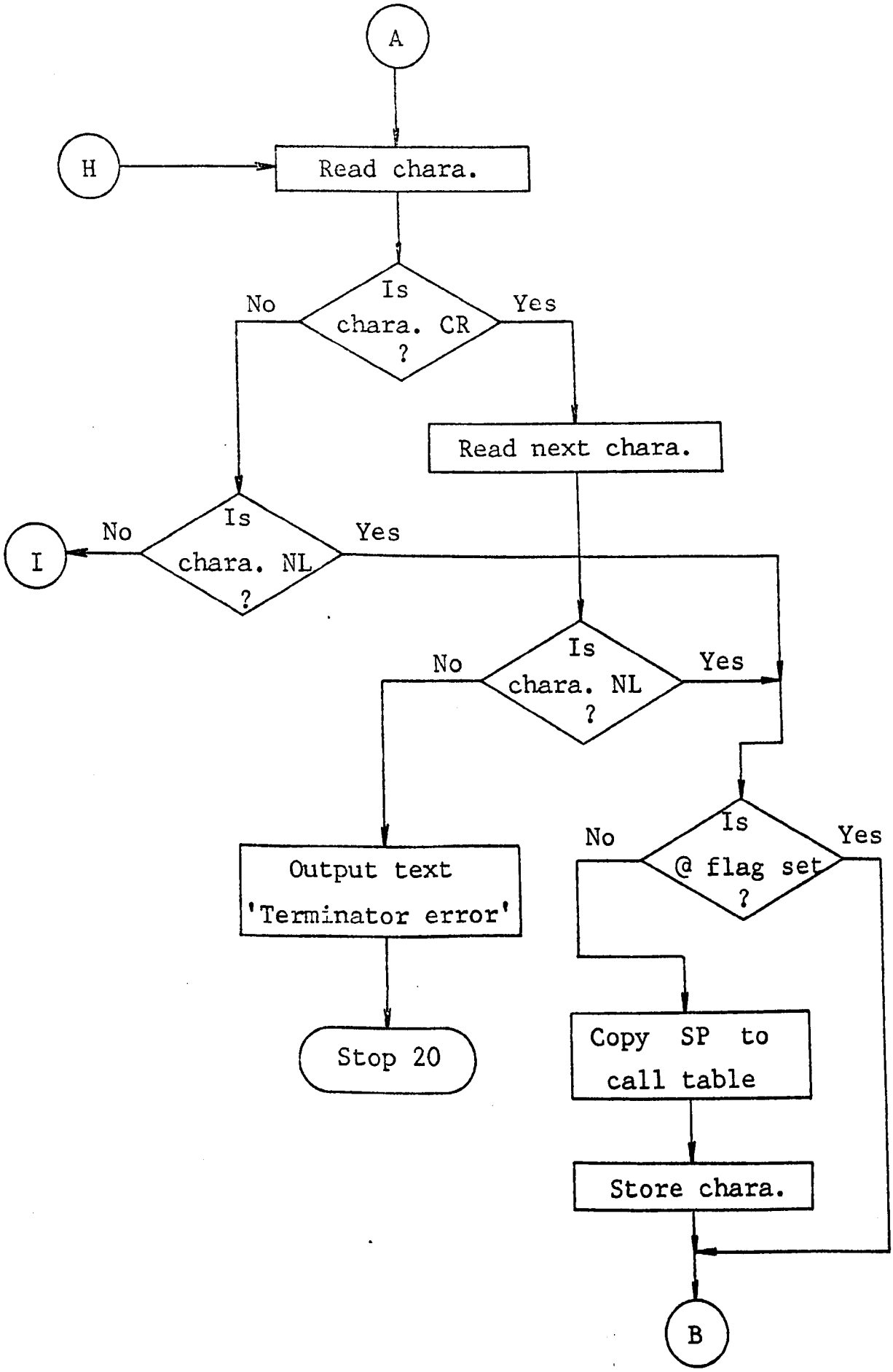
```
[67]←41,L            'OUTPUT CODE'
     V19=2
     V20=VNO
     +12
    ←S555,0          'OUTPUT <FF> CHARA'
```

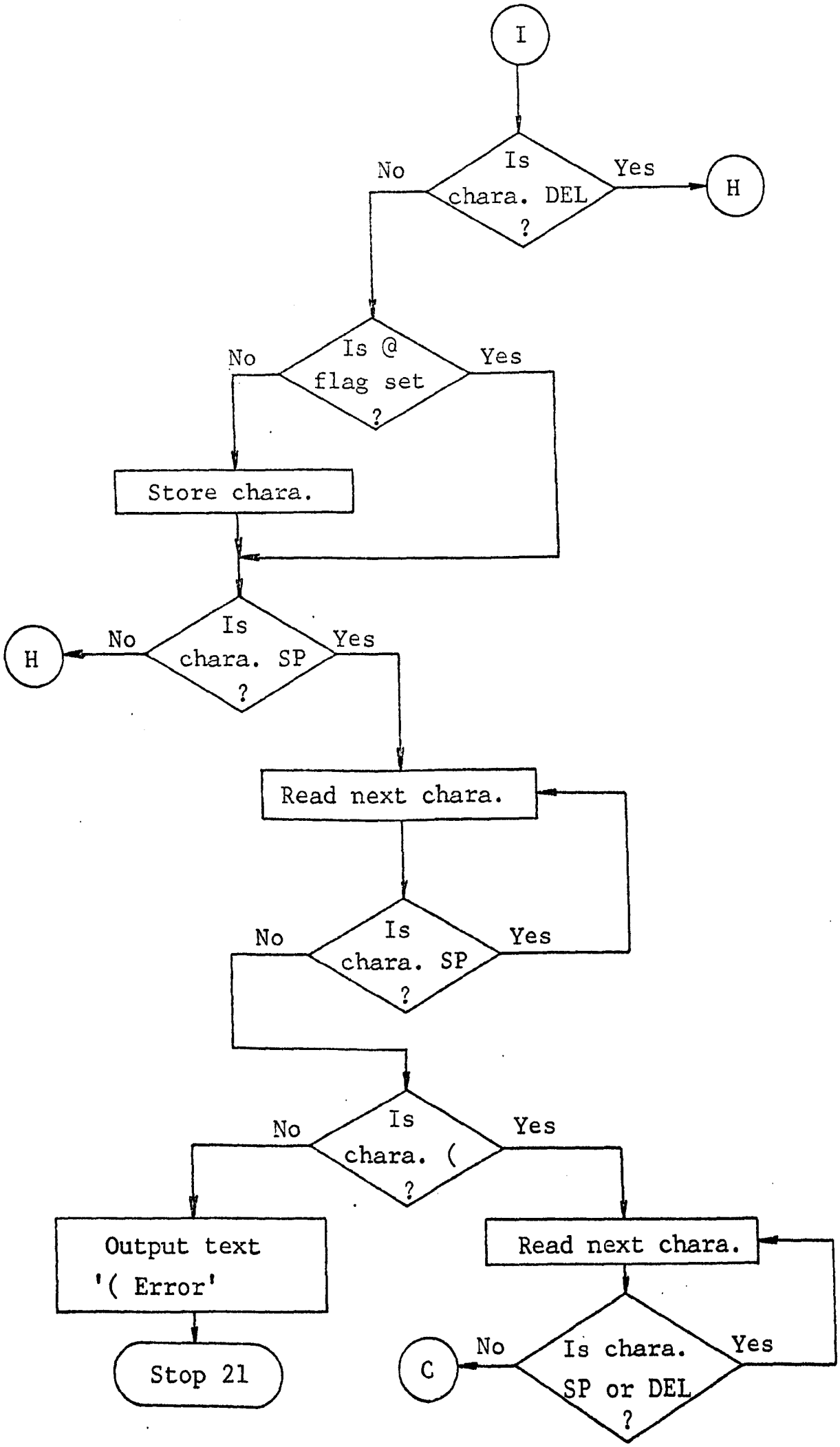
for each subroutine.

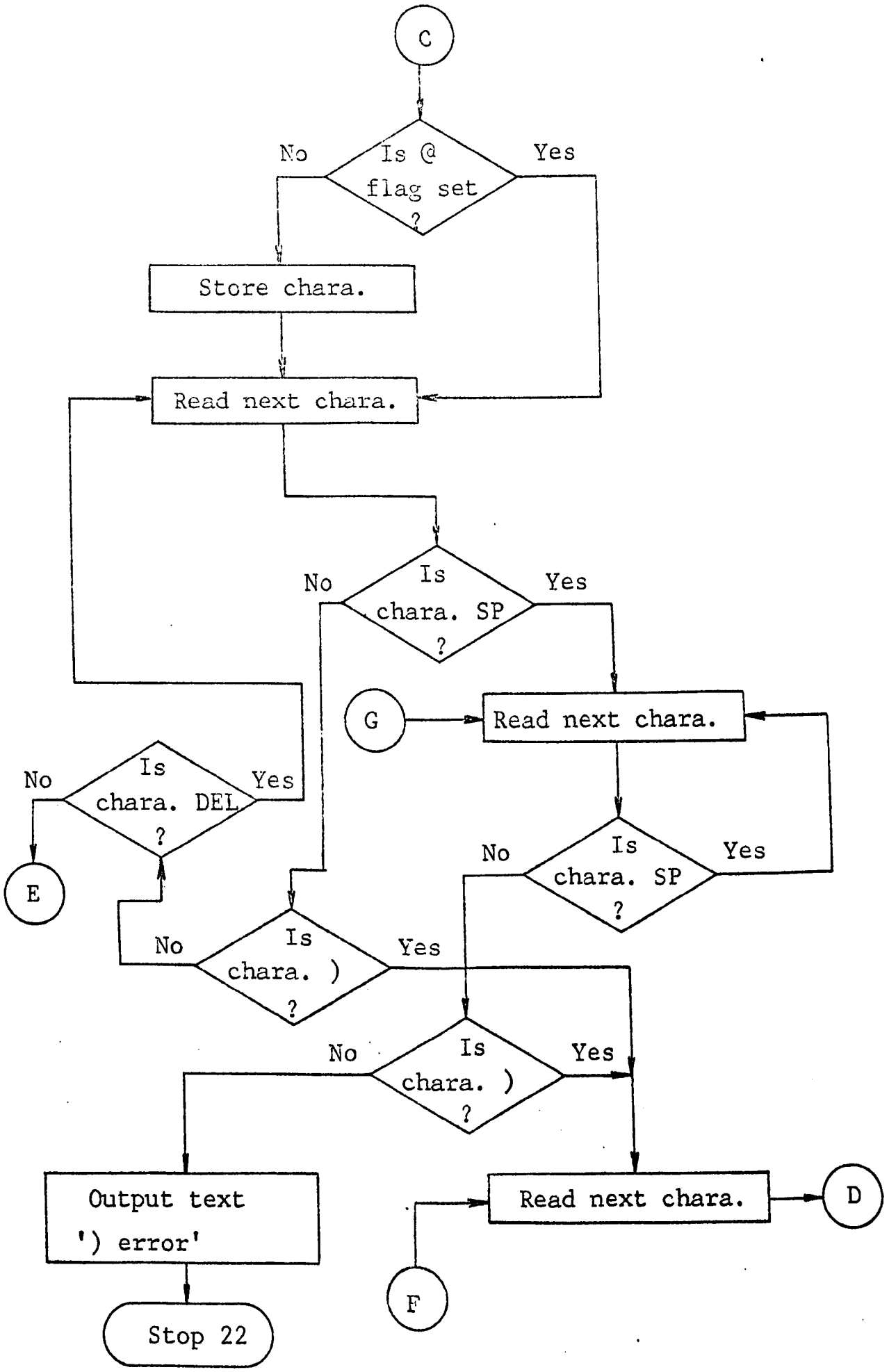
A complete set of flow-diagrams and program listings for both passes of the High-Level Definer are given in the following pages.

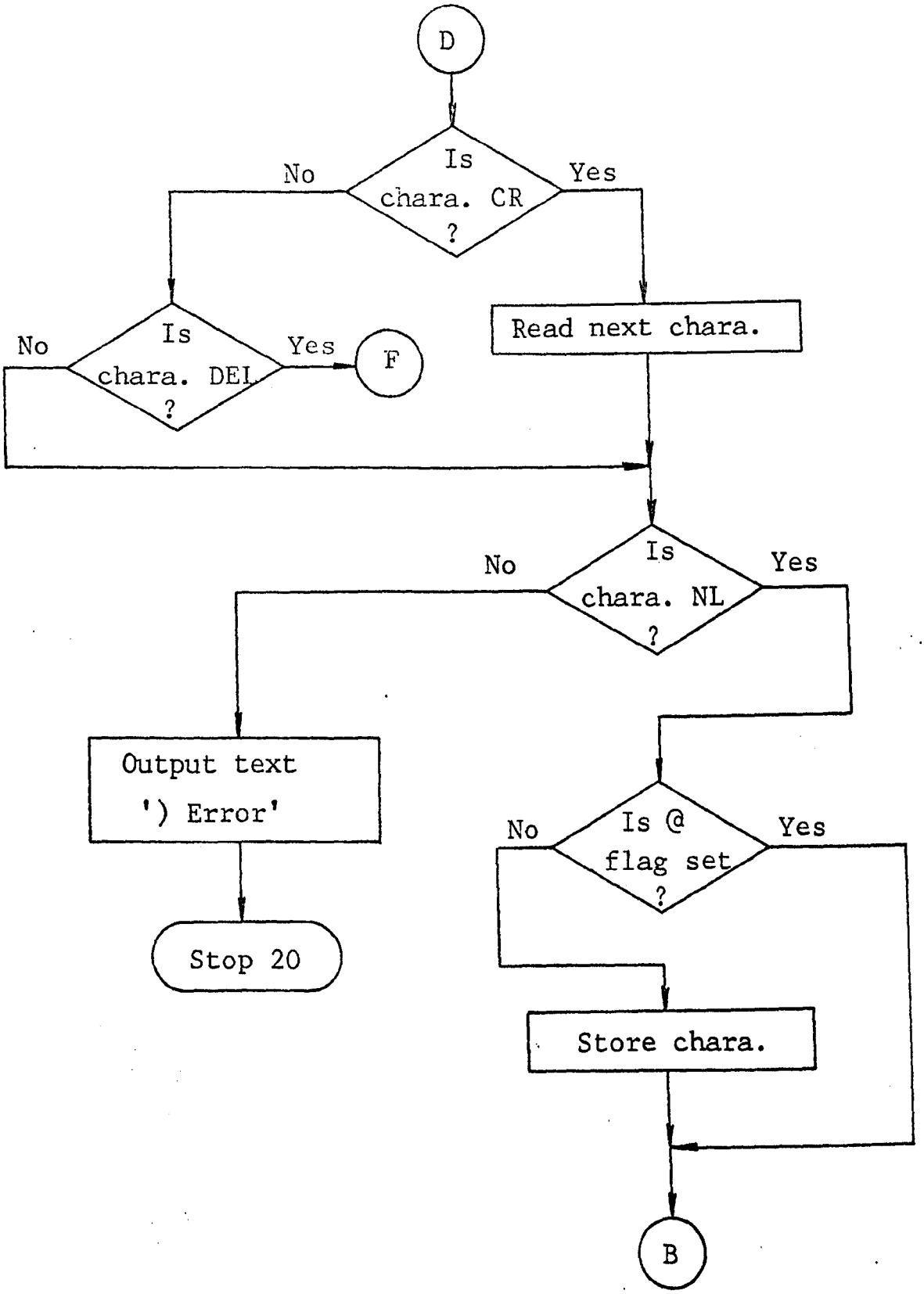
4.2.3 Pass 1 details

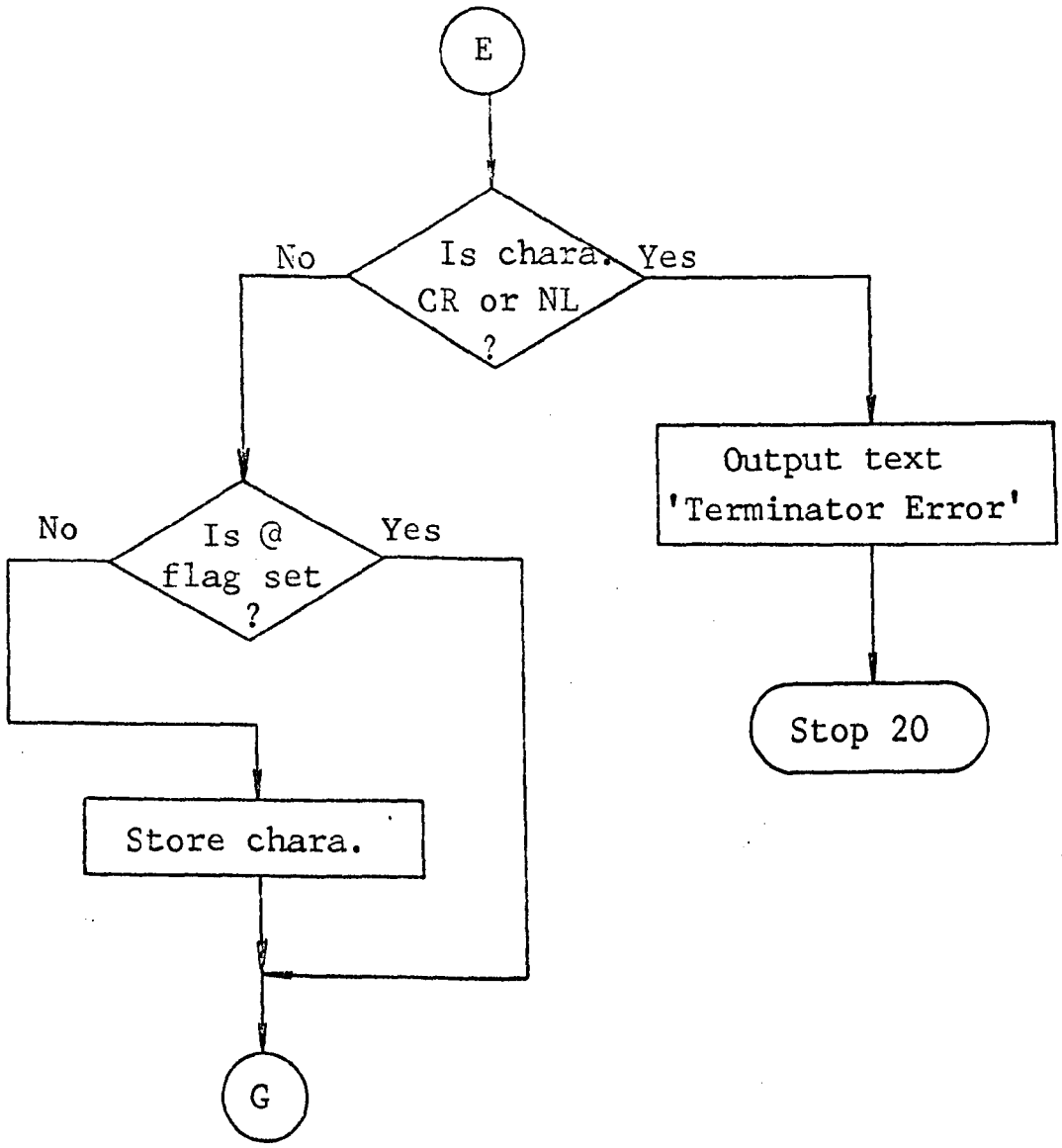



A
H
Read chara.
Is chara. CR ?
No
Yes
Read next chara.
Is chara. NL ?
No
Yes
I
Is chara. NL ?
No
Yes
Output text 'Terminator error'
Stop 20
Is @ flag set ?
No
Yes
Copy SP to call table
Store chara.
B


I
Is chara. DEL ?
No
Yes
H
Is @ flag set ?
No
Yes
Store chara.
Is chara. SP ?
No
Yes
H
Read next chara.
Is chara. SP ?
No
Yes
Is chara. (?
No
Yes
Output text '(Error'
Stop 21
Read next chara.
Is chara. SP or DEL ?
No
Yes
C


C
Is @ flag set ?
No
Yes
Store chara.
Read next chara.
Is chara. SP ?
No
Yes
G
Read next chara.
Is chara. DEL ?
No
Yes
E
Is chara. SP ?
No
Yes
Is chara.) ?
No
Yes
Is chara.) ?
No
Yes
Output text ') error'
Stop 22
Read next chara.
D
F


D
Is chara. CR ?
No
Yes
Read next chara.
Is chara. DEL ?
No
Yes
F
Is chara. NL ?
No
Yes
Output text ') Error'
Stop 20
Is @ flag set ?
No
Yes
Store chara.
B

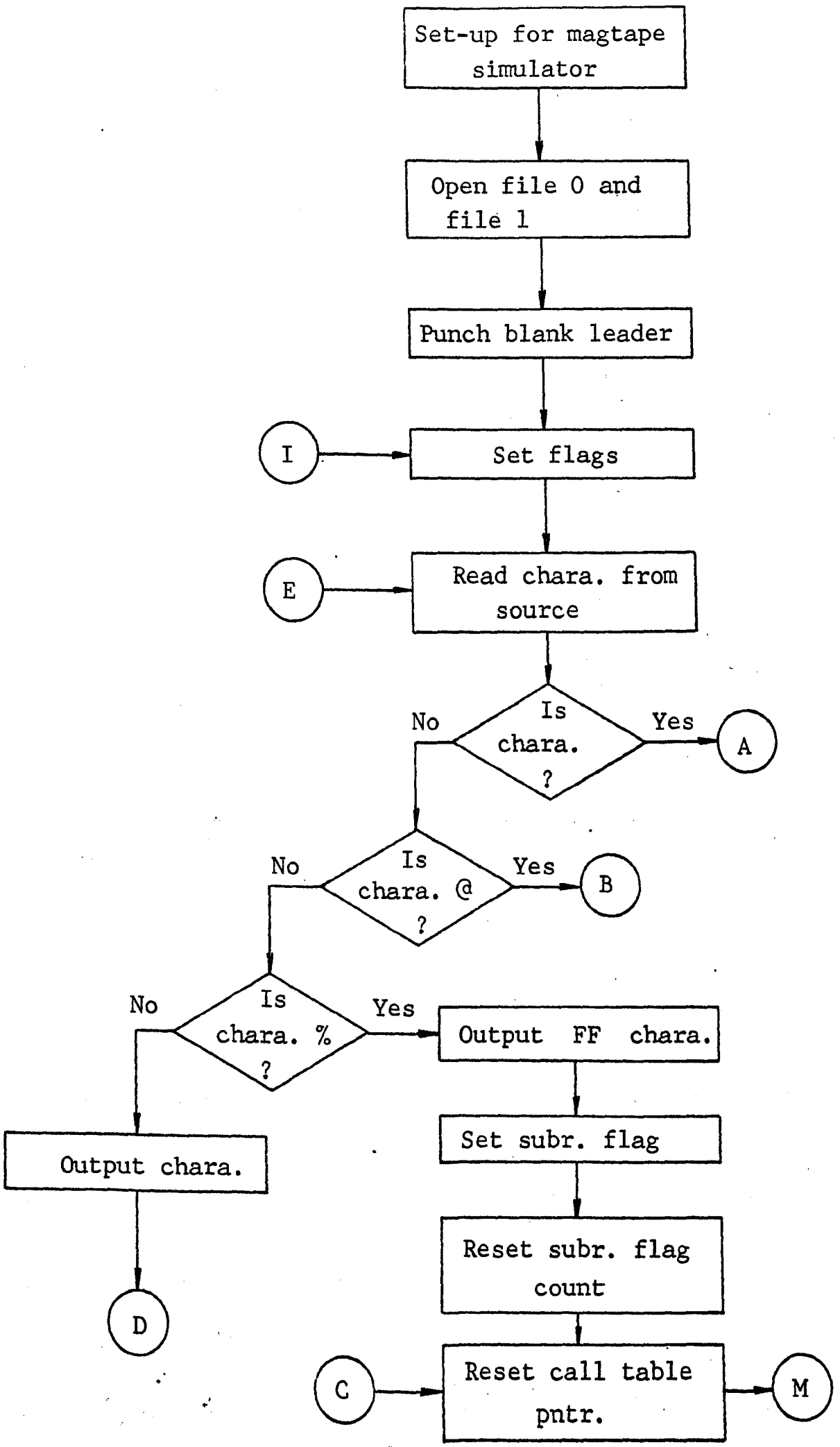

E
Is chara. CR or NL ?
No
Yes
Output text 'Terminator Error'
Stop 20
Is @ flag set ?
No
Yes
Store chara.
G

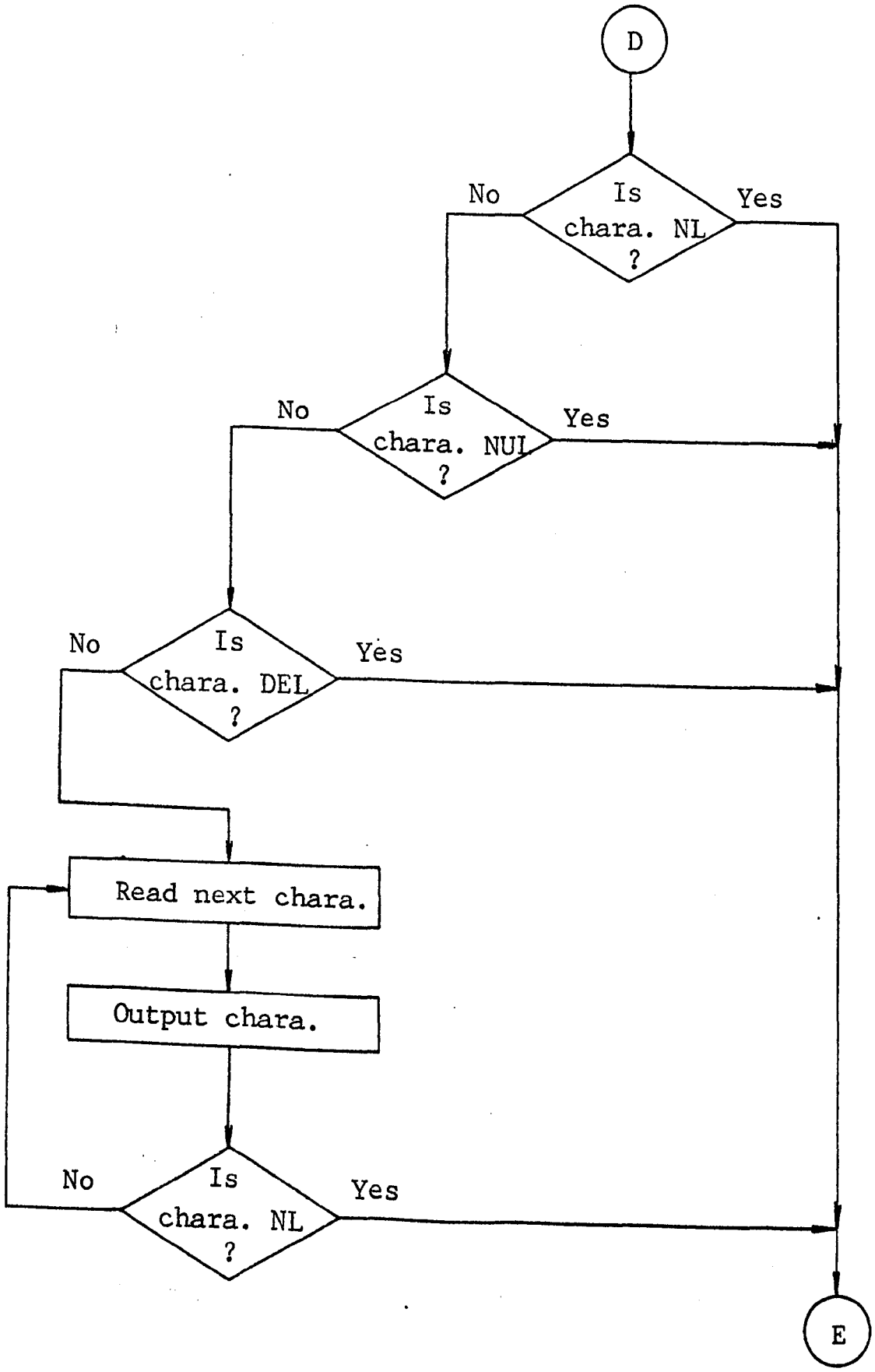
```
N
HIGH-LEVEL DEFINER - R.A.C. 16/6/77;

U
[0] N1=VN0                      'SET LABEL TABLE BASE ADDR'
    AS 4400,-1

[1] V17=0                       'CLEAF @ FLAG'
    V19=3
   ←S555,1                      'READ CHARA'
   ←4,V21#VN0=0                 'IS CHARA #'
    +163
   ←74,V21#VN0=0                'IS CHARA @'
    +192
   ←1,V21#VN0=0                 'IS CHARA <NL>'
    +10
   ←1,V21#VN0=0                 'IS CHARA <NUL>'
    +0
   ←1,V21#VN0=0                 'IS CHARA <DEL>'
    +255
   ←3,V21#VN0=0                 'IS CHARA %'
    +165
[2]←S555,1
   ←2,V21#VN0#0                 'IS CHARA <NL>'
    +10
   ←1

[3]VN1=VN0,N1+1                 'STORE %'
    +165
    V19=3
    V22=8
   ←S555,2                      'STOP 8'
   ←10                          'PASS 2'

[74]V17=31                      'SET @ FLAG'
[4] V19=3
   ←S555,1
   ←70,V21#VN0#0                'IS CHARA <CR>'
    +141
   ←S555,1                      'READ NEXT CHARA'
   ←75,V21#VN0#0                'IS CHARA <NL>'
    +10
[76]←84,V17#0                   'IS @ FLAG SET'
    VN1=VN0,N1+1                'STORE <SP>'
    +160
    VN1=V21,N1+1                'STORE CHARA'
[84]←1

[70]←69,V21#VN0#0               'IS CHARA <NL>'
    +10
   ←76
[69]←4,V21#VN0=0                'IS CHARA <DEL>'
    +255
   ←85,V17#0                    'IS @ FLAG SET'
    VN1=V21,N1+1                'STORE CHARA'
[85]←4,V21#VN0#0                'IS CHARA <SP>'
    +160

[6]←S555,1
```

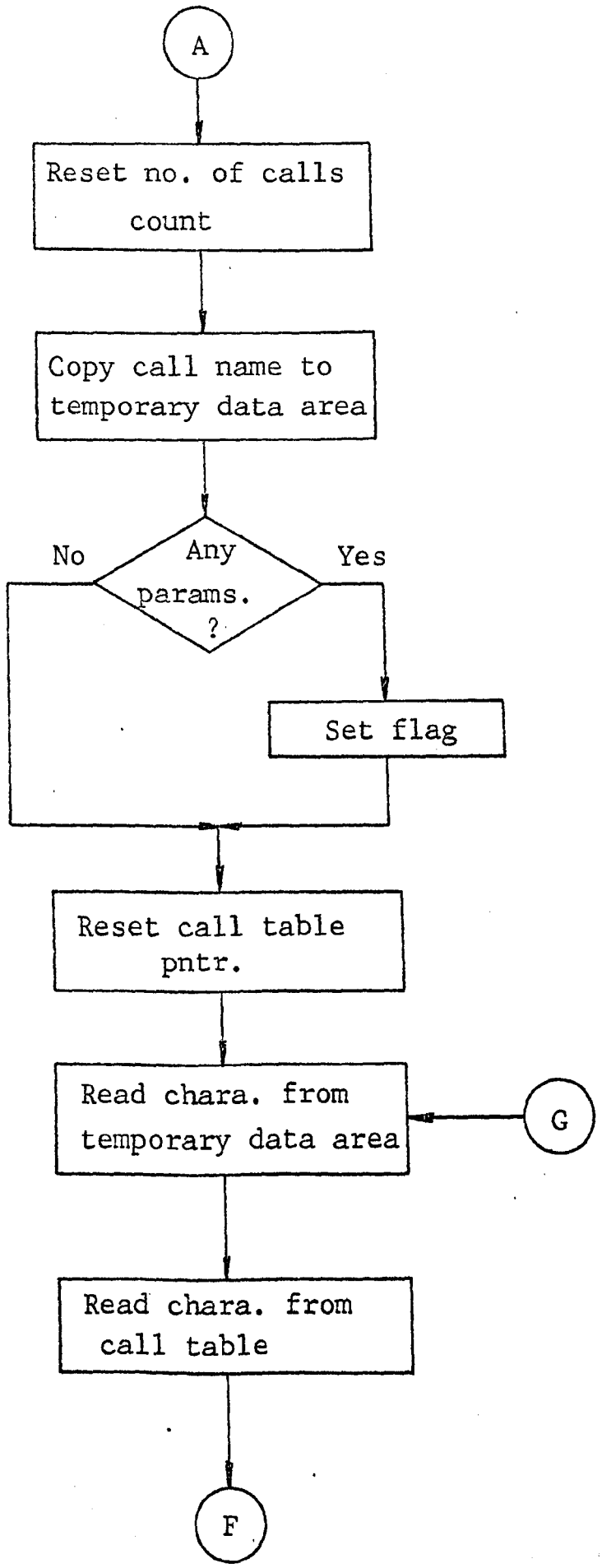
```
    ←6,V21#VN0=0                'IS CHARA <SP>
     +160
    ←77,V21#VN0#0               'IS CHARA (
     +40

[7]←S555,1
    ←7,V21#VN0=0                'IS CHARA <SP>
     +160
    ←7,V21#VN0=0                'IS CHARA <DEL>
     +255
    ←8,V17#0                    'IS @ FLAG SET
     VN1=V21,N1+1               'STORE CHARA

[8] ←S555,1
    ←9,V21#VN0=0                'IS CHARA <SP>
     +160
    ←79,V21#VN0=0               'IS CHARA )
     +169
    ←8,V21#VN0=0                'IS CHARA <DEL>
     +255
    ←78,V21#VN0=0               'IS CHARA <CR>
     +141
    ←78,V21#VN0=0               'IS CHARA <NL>
     +10
    ←8,V17#0
     VN1=V21,N1+1               'STORE CHARA
    ←8

[9]←S555,1
    ←9,V21#VN0=0                'IS CHARA <SP>
     +160
    ←78,V21#VN0#0               'IS CHARA )
     +169
[79]←S555,1
    ←82,V21#VN0#0               'IS CHARA <CR>
     +141
    ←S555,1
[83]←75,V21#VN0#0               'IS CHARA <NL>
     +10
    ←1,V17#0
     VN1=V21,N1+1               'STORE CHARA
    ←1

[82]←79,V21#VN0=0               'IS CHARA <DEL>
     +255
    ←83

[75]V19=4
    TEXT 1P 1E
TERMINATOR ERROR
     V22=20
    ←S555,6                     'STOP 20
    ←75

[77]V19=4
    TEXT 1P 1E
( ERROR
     V22=21
    ←S555,6                     'STOP 21
    ←77
```

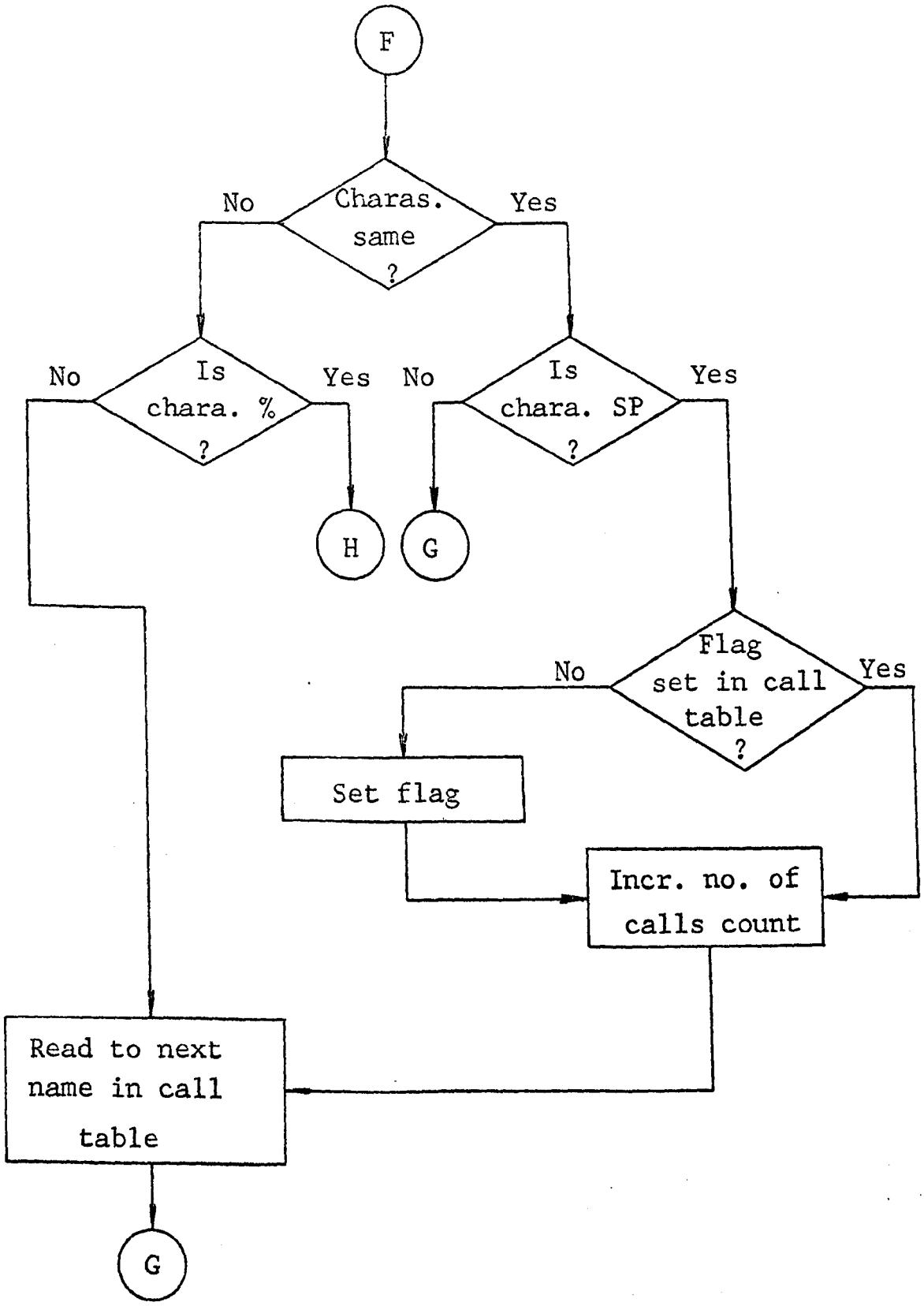
```
[78]V19=4
    TEXT 1P 1E
) ERROR
    V22=22
   ←S555.6                    'STOP 22'
   ←78
```

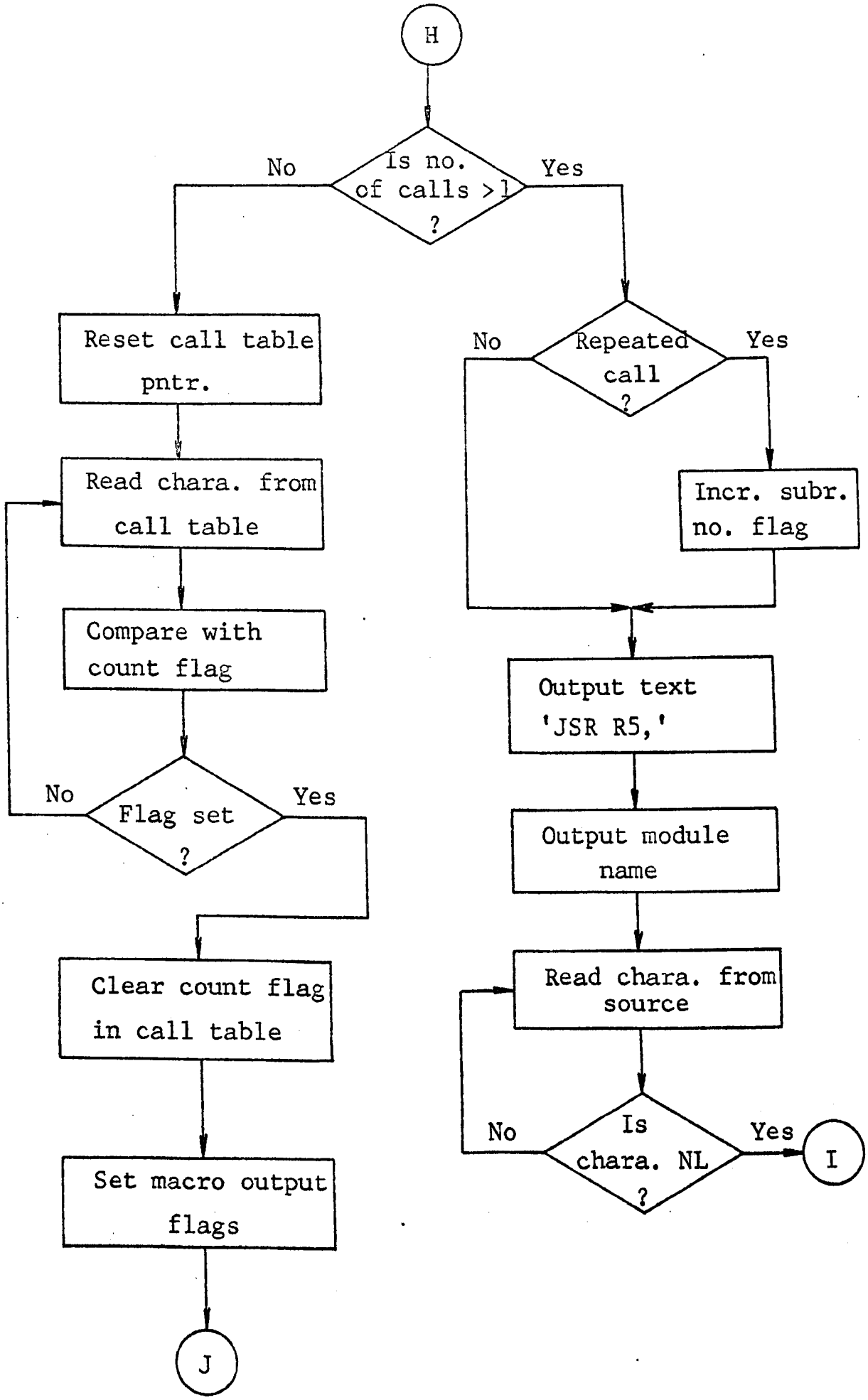
4.2.4 Pass 2 details

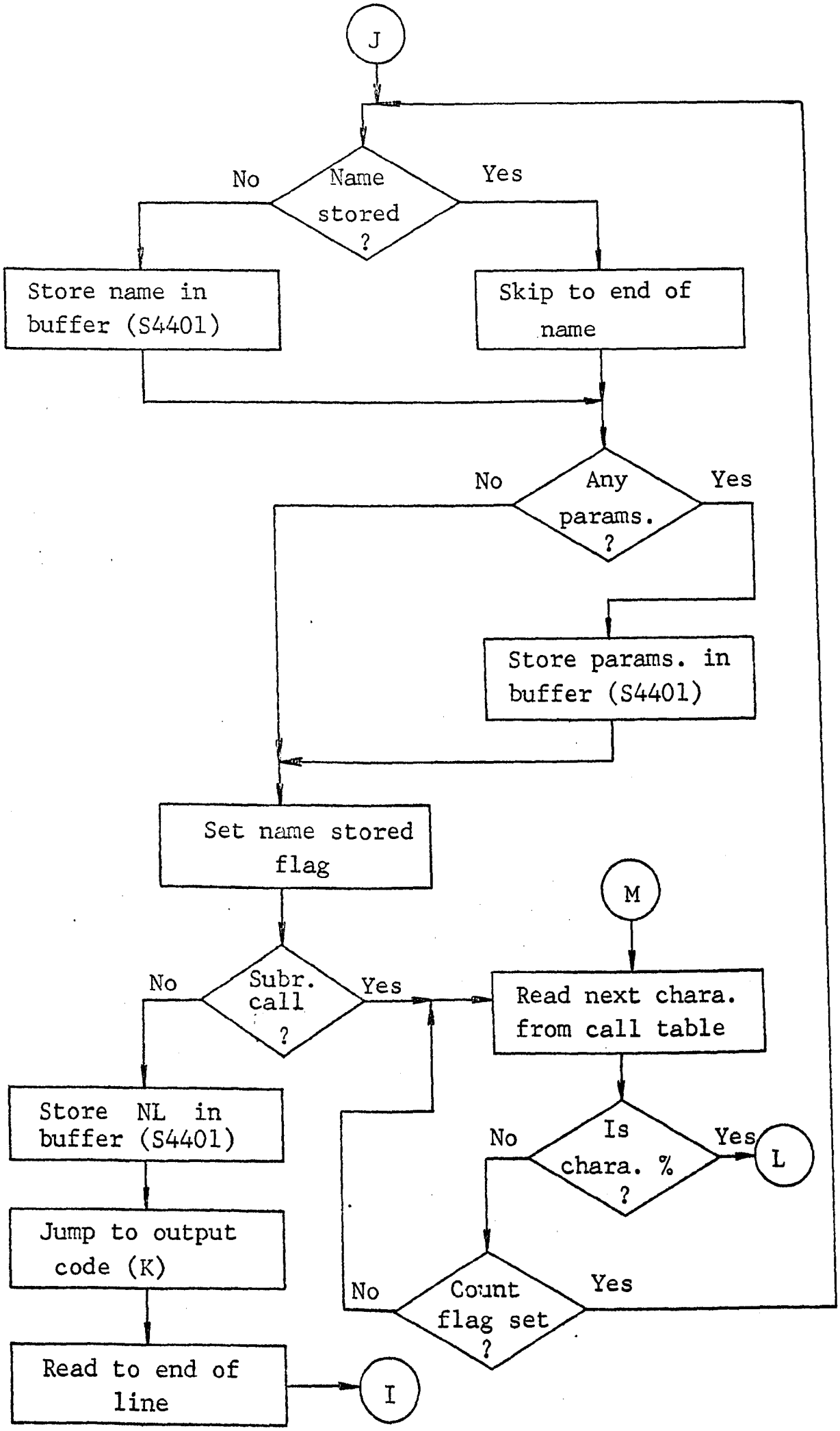


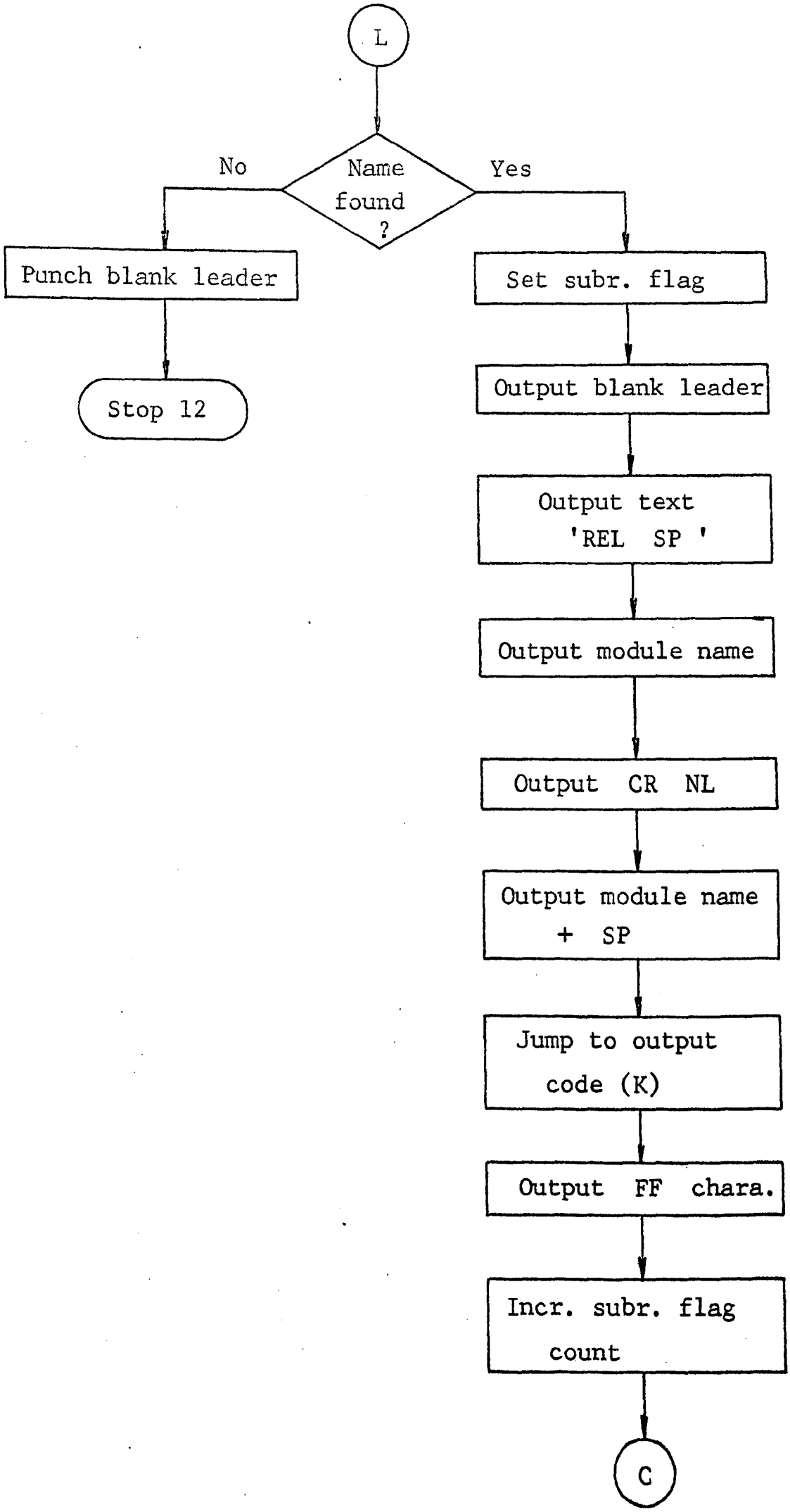

D
Is chara. NL ?
No
Yes
Is chara. NUL ?
No
Yes
Is chara. DEL ?
No
Yes
Read next chara.
Output chara.
Is chara. NL ?
No
Yes
E

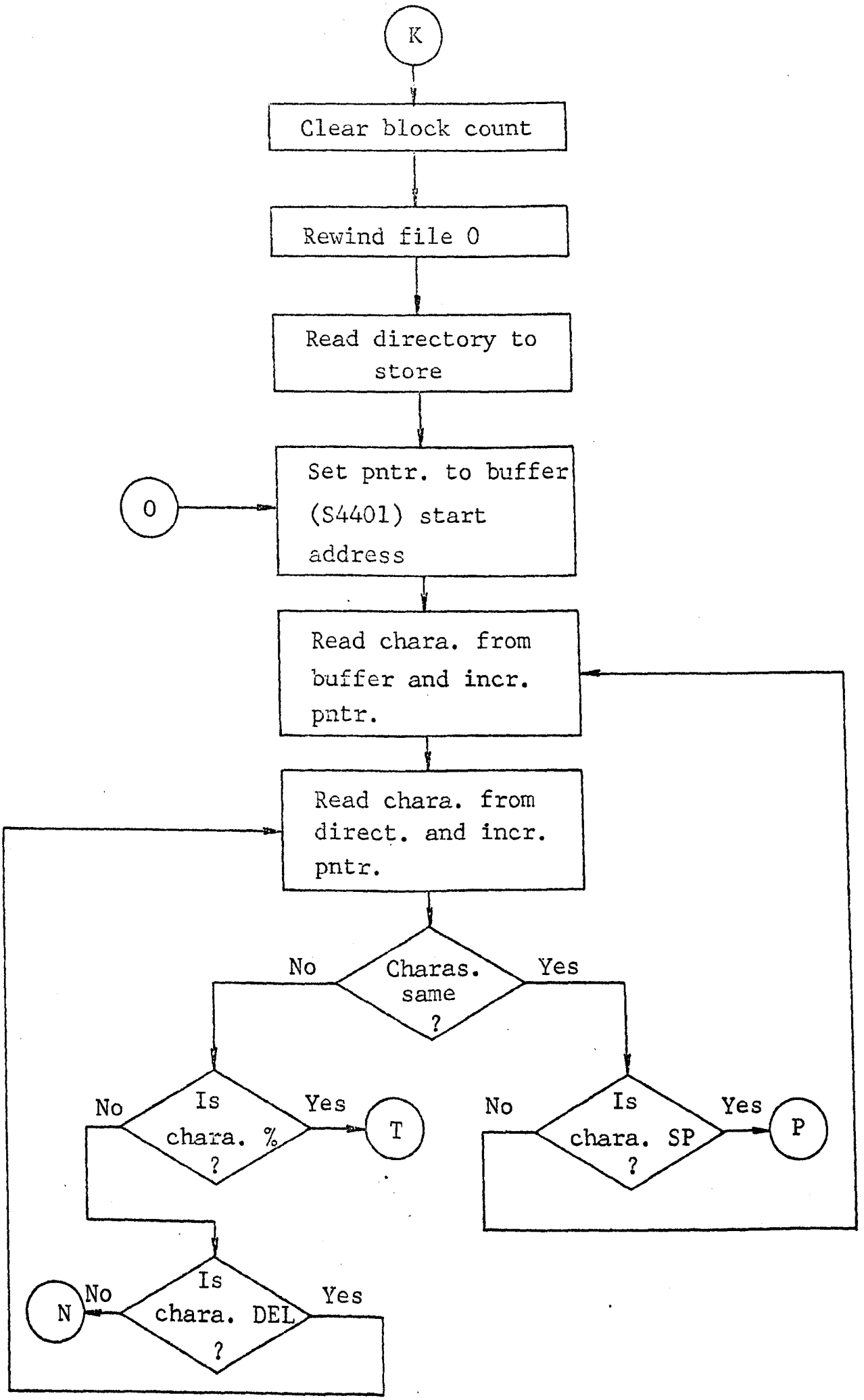

A
Reset no. of calls count
Copy call name to temporary data area
Any params. ?
No
Yes
Set flag
Reset call table pntr.
Read chara. from temporary data area
G
Read chara. from call table
F

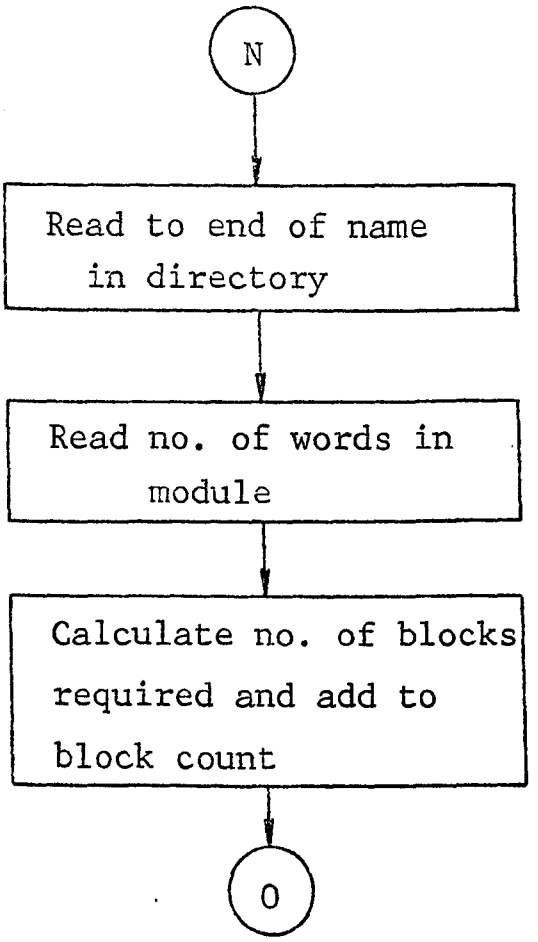

F
Charas. same ?
No
Yes
Is chara. % ?
No
Yes
Is chara. SP ?
No
Yes
H
G
Flag set in call table ?
No
Yes
Set flag
Incr. no. of calls count
Read to next name in call table
G

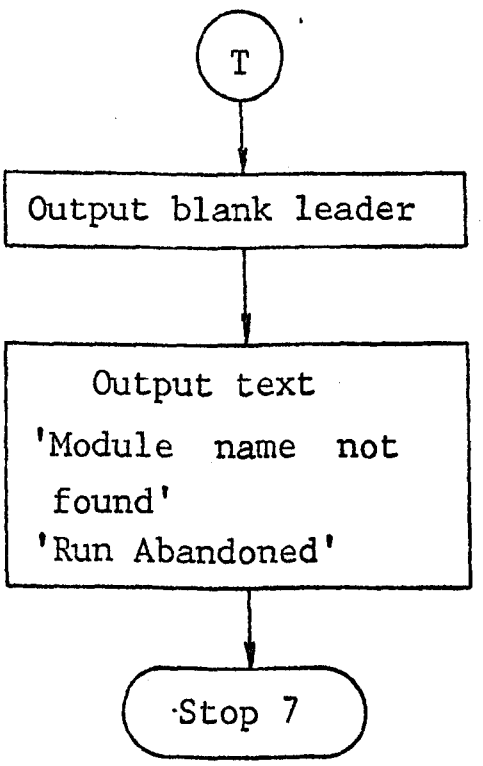

H
Is no. of calls >1 ?
No
Yes
Reset call table pntr.
Read chara. from call table
Compare with count flag
Flag set ?
No
Yes
Clear count flag in call table
Set macro output flags
J
Repeated call ?
No
Yes
Incr. subr. no. flag
Output text 'JSR R5,'
Output module name
Read chara. from source
Is chara. NL ?
No
Yes
I

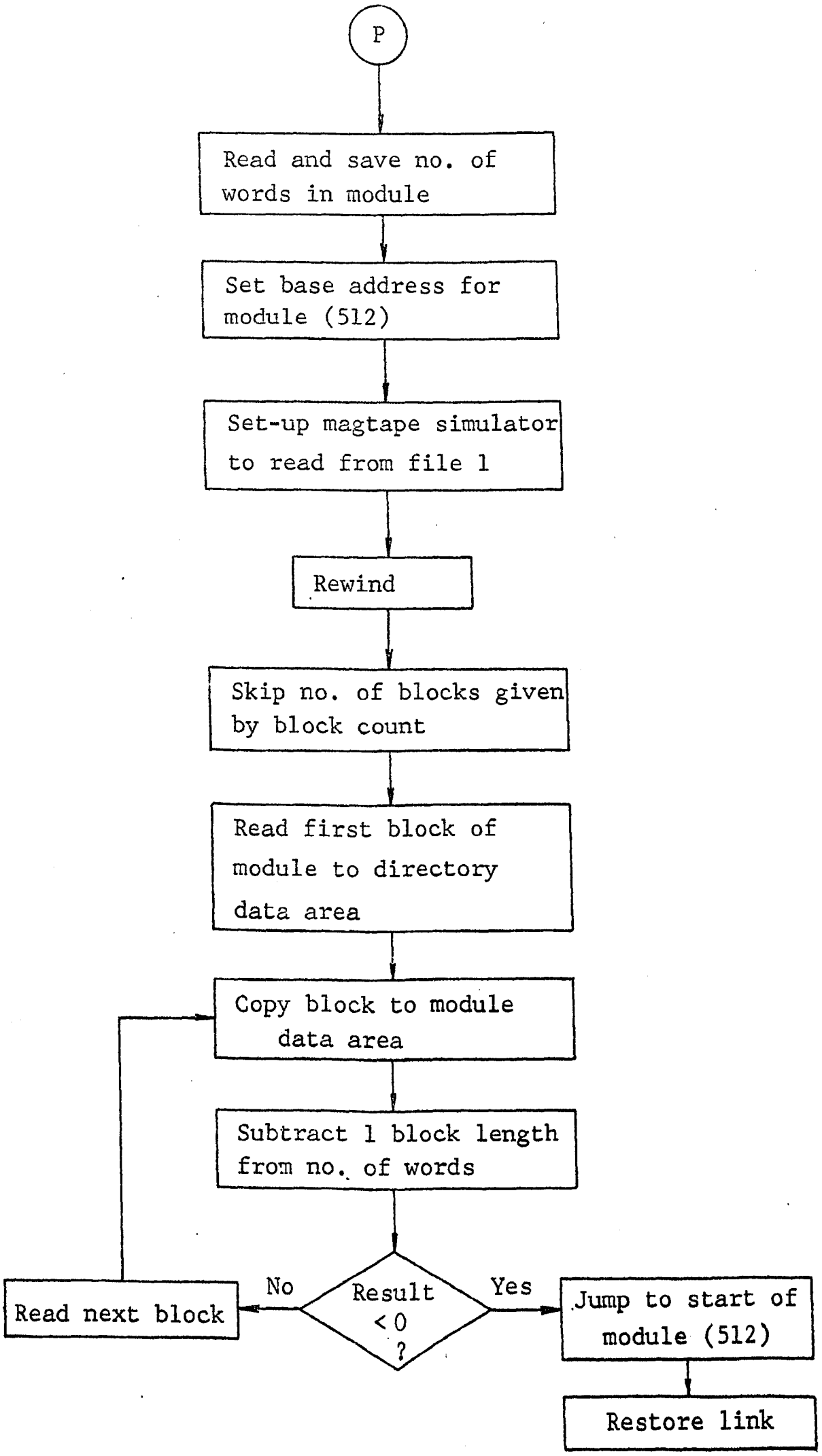

J
Name stored ?
No
Yes
Store name in buffer (S4401)
Skip to end of name
Any params. ?
No
Yes
Store params. in buffer (S4401)
Set name stored flag
M
Subr. call ?
No
Yes
Read next chara. from call table
Store NL in buffer (S4401)
Is chara. % ?
No
Yes
L
Jump to output code (K)
Count flag set ?
No
Yes
Read to end of line
I

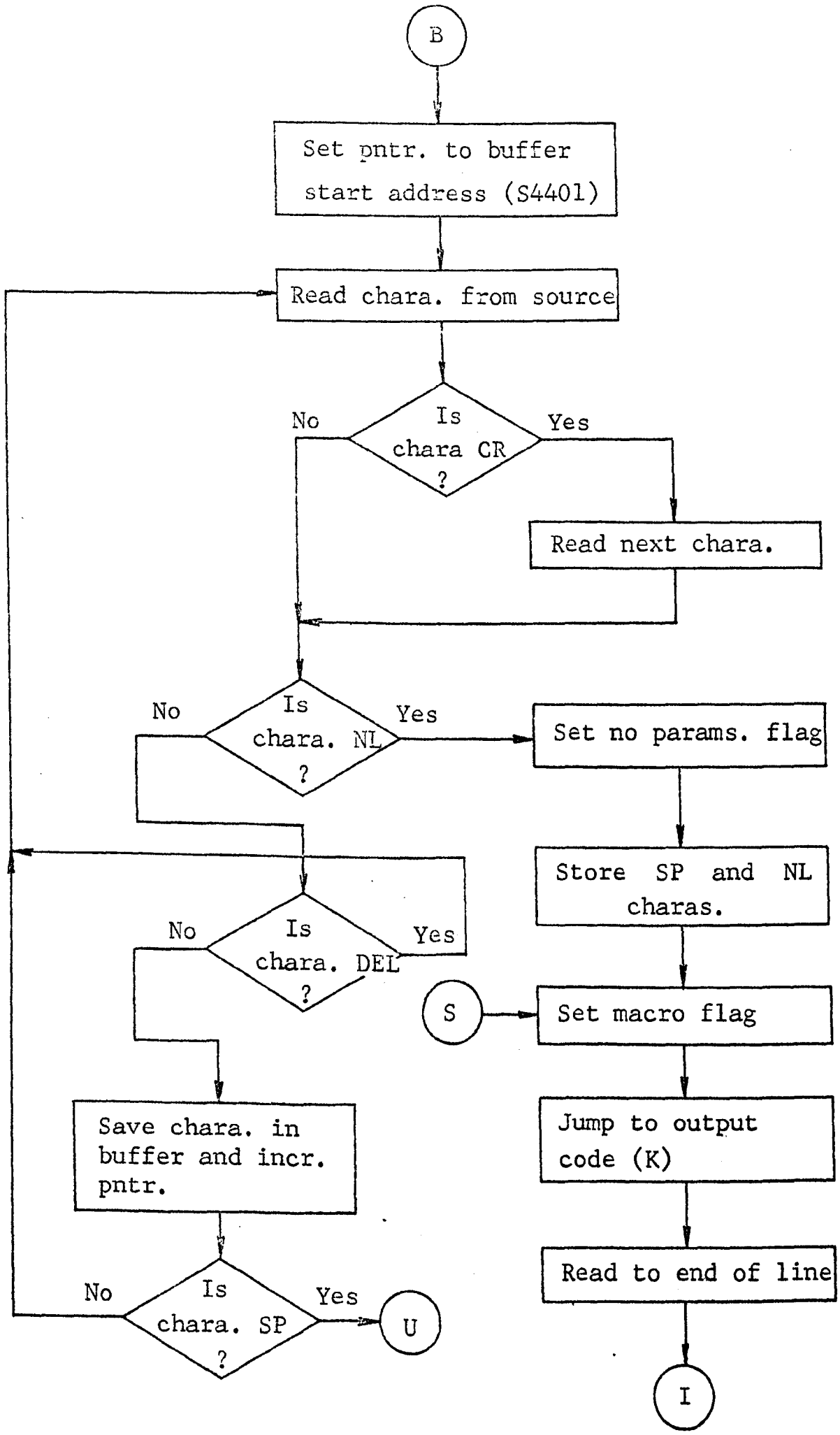

L
Name found ?
No
Yes
Punch blank leader
Stop 12
Set subr. flag
Output blank leader
Output text 'REL SP '
Output module name
Output CR NL
Output module name + SP
Jump to output code (K)
Output FF chara.
Incr. subr. flag count
C

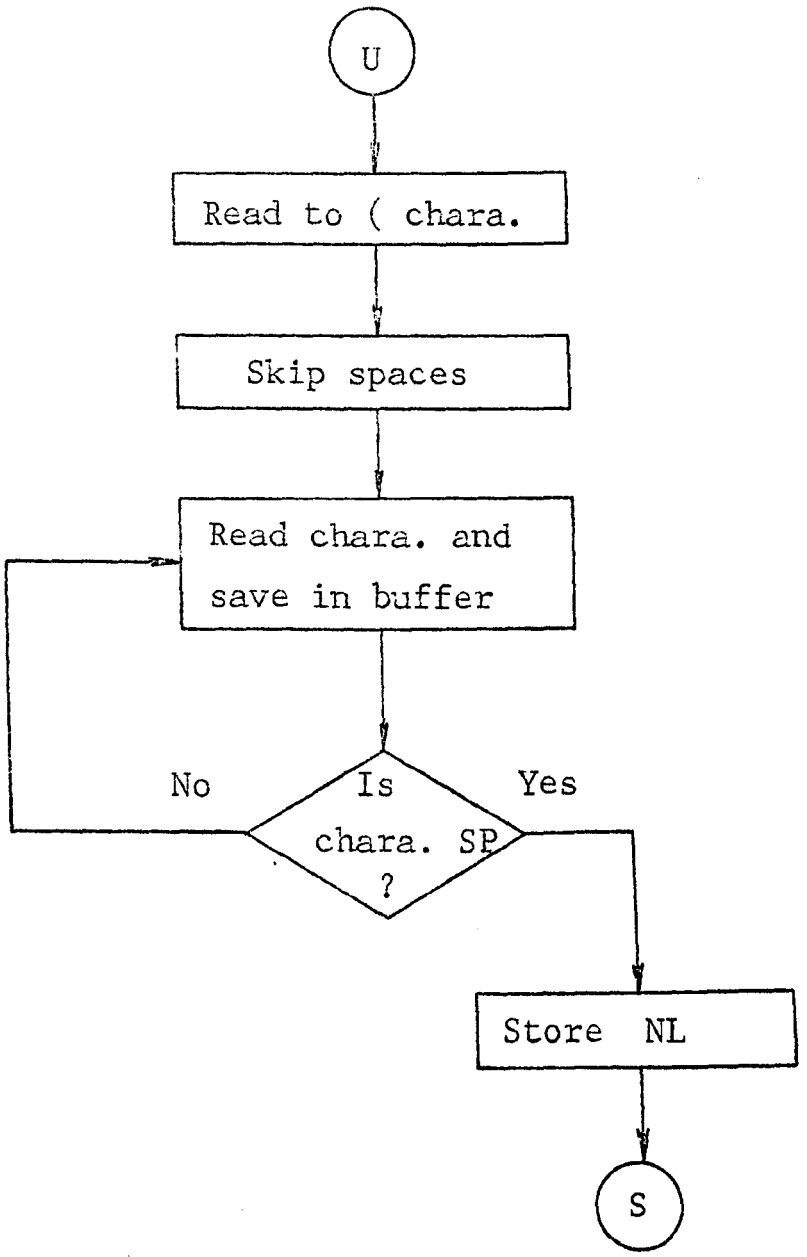

K
Clear block count
Rewind file 0
Read directory to store
O
Set pntr. to buffer (S4401) start address
Read chara. from buffer and incr. pntr.
Read chara. from direct. and incr. pntr.
Charas. same ?
No
Yes
Is chara. % ?
No
Yes
T
Is chara. SP ?
No
Yes
P
Is chara. DEL ?
N
No
Yes


N
Read to end of name in directory
Read no. of words in module
Calculate no. of blocks required and add to block count
O


T
Output blank leader
Output text
'Module name not found'
'Run Abandoned'
Stop 7


P
Read and save no. of words in module
Set base address for module (512)
Set-up magtape simulator to read from file 1
Rewind
Skip no. of blocks given by block count
Read first block of module to directory data area
Copy block to module data area
Subtract 1 block length from no. of words
Result < 0 ?
No
Yes
Read next block
Jump to start of module (512)
Restore link


B
Set pntr. to buffer start address (S4401)
Read chara. from source
Is chara CR ?
No
Yes
Read next chara.
Is chara. NL ?
No
Yes
Set no params. flag
Store SP and NL charas.
Is chara. DEL ?
No
Yes
S
Set macro flag
Jump to output code (K)
Save chara. in buffer and incr. pntr.
Read to end of line
Is chara. SP ?
No
Yes
U
I


U
Read to (chara.
Skip spaces
Read chara. and save in buffer
No
Is chara. SP ?
Yes
Store NL
S

```
     'PASS 2'

[10]N1=VN0                    'SET UP FILE MAP FOR MTS'
     AS 4571,0
     VN1=1                    '1 BLOCK IN FILE 0'
     V0=V[N1+2]
     VN1=VN0
     +500                     '500 BLOCKS IN FILE 1'
     V20=0                    'SET UP FOR MAGTAPE SIMULATOR'
     V21=V20
     N1=VN0
     AS 4403

    ←S4570,0                  'OPEN FILE 0'

     V20=1
     V21=V20
    ←S4570,0                  'OPEN FILE 1'

    ←57,L                     'PUNCH BLANK LEADER'

     V9=VN0,N2-1              'SET SUBR FLAG COUNT'
     OCT 1000
[19]N2=VN0                    '10 WORD DATA AREA'
     AL 100
     V10=0                    'SET FLAGS'
     V17=V10

[21]V19=3
    ←S555,1                   'READ SOURCE'
     V20=V21
    ←24,V20#VN0=0             'IS CHARA #'
    +163
    ←60,V20#VN0=0             'IS CHARA 0'
     +192
    ←23,V20#VN0=0             'IS CHARA %'
     +165                     'END OF SOURCE'
     V19=2
    ←S555,0                   'OUTPUT CHARA'
    ←21,V20#VN0=0             'IS CHARA <NL>'
     +10
    ←21,V20#VN0=0             'IS CHARA <NUL>'
     +0
    ←21,V20#VN0=0             'IS CHARA <DEL>'
     +255

[22]V19=3
    ←S555,1                   'READ NEXT CHARA'
     V20=V21
     V19=2
    ←S555,0                   'OUTPUT CHARA'
    ←22,V20#VN0#0             'IS CHARA <NL>'
     +10
    ←21                       'READ NEXT INSTR'

[24]V8=0                      'RESET NO OF CALLS COUNT'
[28]←S555,1                   'READ LABEL'
    ←72,V21#VN0#0             'IS CHARA <CR>'
     +141
```

```
   ←S555,1
[72]←73,V21#VNO#0       'IS CHARA <NL>'
    +10
    V21=VNO             'LOAD <SPACE>'
    +160
    V17=1

[73]←28,V21#VNO=0       'IS CHARA <DEL>'
    +255
    VN2=V21,N2+1        'STORE CHARA'
   ←28,V21#VNO#0        'IS CHARA <SPACE>'
    +160

[39]N1=VNO
    AS 4400
[34]N2=VNO              'RESET DATA  PNTR'
    AL 100
    V5=N1,N2+1          'SAVE N1'
[35]V6=VN2,N2+1
    V7=VN1,N1+1         'V7=CHARA FROM LABEL TABLE'
    V7=V7&VNO
    OCT 777
   ←25,V6#V7=0          'CHARAS SAME ?'
   ←29,V7#VNO=0         'IS CHARA %'
    +165                'END OF LABEL TABLE'

[33]V7=VN1,N1+1
   ←33,V7#VNO#0         'READ TO NEXT LABEL'
    +10
   ←34

[25]←35,V6#VNO#0        'IS CHARA <SPACE>'
    +160
    N1=V5
    V7=VN1
   ←51,V7&VNO#0         'FLAG ALREADY SET?'
    OCT 77777000
    V10=1
    VN1=VN1#V9          'SET FLAG'
[51]V8=V8+1             'INC NO OF CALLS'
   ←33                  'CONTINUE SEARCH'

[29]←30,V8-2<0
   ←55,V10=0            'REPEATED SUBR'
    V9=V9+VNO           'INC SUBR NO FLAG'
    OCT 1000

[55]V19=2
    TEXT
 JSR R5,
    N2=VNO              'RESET PNTR'
    AL 100
[32]V20=VN2,N2+1
   ←31,V20#VNO=0        'IS CHARA <SPACE>'
    +160
   ←S555,0              'NO - OUTPUT CHARA'
   ←32
[31]TEXT 1E
```

```
[56]V19=3
   ←19,V17#0
   ←S555,1                    'READ TO NEXT <NL>'
   ←56,V21#VNO#0
    +10
   ←19

[30]N1=VNO                    'OUTPUT AS MACRO'
    AS 4400                   'BLANK DATA SUBR'
[43]V7=VN1,N1+1
    V7=V7&VNO                 'CLEAR BOTTOM BITS'
    OCT 777000
   ←43,V7#V9#0                'SEARCH TABLE FOR FLAG'
    V0=V0,N1-1
    VN1=VN1#V9                'CLEAR FLAG'
    V11=0                     'MACRO OUTPUT FLAG'
    V5=0,N1+1
   ←46                        'OUTPUT CODE'

[60]N1=VNO                    'SET PNTR'
    AS 4401,-1
[64]←S555,1                   'READ NEXT CHARA'
   ←71,V21#VNO#0              'IS CHARA <CR>'
    +141
   ←S555,1                    'READ NEXT CHARA'
[71]←68,V21#VNO#0             'IS CHARA <NL>'
    +10
    V21=VNO                   'LOAD <SPACE>'
    +160
    V17=1                     'SET NO PARAMS FLAG'
    VN1=V21,N1+1              'STORE CHARA'
   ←65
[68]←64,V21#VNO=0             'IS CHARA <DEL>'
    +255
    VN1=V21,N1+1
   ←64,V21#VNO#0              'IS CHARA <SPACE>'
    +160
[61]←S555,1
   ←61,V21#VNO#0              'IS CHARA ('
    +40
[62]←S555,1
   ←62,V21#VNO=0              'SKIP <SPACES>'
    +160
    VN1=V21,N1+1              'SAVE CHARA'
[63]←S555,1
    VN1=V21,N1+1
   ←63,V21#VNO#0              'IS CHARA <SPACE>'
    +160
    VN1=VNO                   'LOAD <NL>'
    +10
[65]V11=0                     'SET MACRO FLAG'
   ←41,L                      'OUTPUT CODE'
   ←56                        'READ TO <NL>'

[23]V19=2
    V20=VNO
    +12
   ←S555,0                    'OUTPUT <FF> CHARA'

    V11=1
```

```
     V9=VN0                    'RESET SUBR FLAG'
     OCT 1000
[48]N1=VN0                     'RESET LABEL TABLE PNTR'
     AS 4400
     V5=0
[45]V7=VN1,N1+1
    ←50,V7#VN0=0               'IS CHARA %'
     +165
     V7=V7&VN0
     OCT 777000
    ←46,V7#V9=0                'LABEL FOUND'
    ←45
[50]←47,V5=0                   'LABEL NOT FOUND'
     V11=1                     'SUBR OUTPUT FLAG'
     VN2=VN0,N2+1              'OUTPUT <NL>'
     +10
    ←57,L                      'OUTPUT BLANK LEADER'
     TEXT
 REL
     N1=VN0                    'OUTPUT MODULE NAME'
     AS 4401
     V19=2
[42]V20=VN1,N1+1
    ←37,V20#VN0=0              'IS CHARA <SPACE>'
     +160
    ←S555,0                    'OUTPUT CHARA'
    ←42

[37]V20=VN0
     +141
    ←S555,0                    'OUTPUT <CR>'
     V20=VN0
     +10
    ←S555,0                    'OUTPUT <NL>'
     N1=VN0                    'SET PNTR TO LABEL NAME'
     AS 4401
[38]V20=VN1,N1+1               'OUTPUT LABEL NAME'
    ←40,V20#VN0=0              'IS CHARA <NL>'
     +10
    ←S555,0
    ←38,V20#VN0#0              'IS CHARA <SPACE>'
     +160
    ←67
[40]V20=VN0                    'OUTPUT <SPACE>'
     +160
    ←S555,0
[67]←41,L                      'OUTPUT CODE'
     V19=2
     V20=VN0
     +12
    ←S555,0                    'OUTPUT <FF> CHARA'

     V9=V9+VN0                 'INC SUBR NO FLAG'
     OCT 1000
    ←48                        'SEARCH FOR NEXT LABEL'

[46]←52,V5#0
     V0=V0,N1-1
     N2=VN0
     AS 4401
```

```
     VN2=VN1,N1+1
     VN2=VN2&VN0                    'REMOVE FLAG'
     OCT 777
     V0=V0,N2+1
[49]VN2=VN1,N1+1
    ←53,VN2#VN0=0                   'IS CHARA <SPACE>'
     +160
     V0=V0,N2+1
    ←49

[52]VN2=VN1,N1+1
    ←52,VN2#VN0#0                   'SKIP TO SPACE'
     +160

[53]V0=V0,N2+1
     VN2=VN1,N1+1
    ←53,VN2#VN0#0                   'IS CHARA <NL>'
     +10
     VN2=VN0                        'STORE <SPACE>'
    +160
     V5=V5+1
    ←45,V11#0
     VN2=VN0,N2+1                   'OUTPUT <NL>'
     +10
    ←41,L                           'OUTPUT MACRO'
    ←56

[47]←57,L                           'PUNCH BLANK LEADER'

     V19=4
     V22=12
[26]←S555,6                         'STOP 12'
    ←26

[58]←57,L                           'PUNCH BLANK TAPE'
     V19=4
     TEXT 1P
MODULE
     N1=VN0                         'SET PNTR TO LABEL NAME'
     AS 4401
     V19=4
[59]V20=VN1,N1+1
    ←S555,0                         'OUTPUT LABEL NAME'
    ←59,V20#VN0#0                   'IS CHARA <SP>'
     +160
     TEXT
NOT FOUND
     TEXT 1P
RUN ABANDONED
[44]V19=4
     V22=7
    ←S555,6                         'STOP 7'
    ←44

[41]V15=0                           'SET BLOCK PNTR TO START OF LIB'
     V20=0                          'V20=FILE NO.'
     V21=V20
     N1=VN0                         'N1=10 WORD B.D.A.'
     AS 4403
```

```
    ←S4570,5                    'REWIND'
     V22=VN0                    'V22=N.O.W.'
     +1000

    ←S4570,3                    'READ DIRECT.'
     N1=VN0                     'N1=S.A. OF DIRECT.'
     AS 4404,4
[13]N2=VN0                      'SET TO READ LABEL'
     AS 4401

[14]V7=VN1,N1+1                 'READ CHARAS'
[20]V6=VN2,N2+1
    ←11,V7#V6=0                 'CHARAS SAME'
    ←58,V7#VN0=0                'END OF DIRECT.'
     +165
    ←20,V6#VN0=0                'IS CHARA <DEL>'
     +255

[12]V7=VN1,N1+1                 'READ TO NEXT NAME'
    ←12,V7#VN0#0
     +160
     V7=VN1                     'V7 = N.O.W.'
[36]V6=VN1,N1+1                 'ADVANCE PNTR TO NEXT NAME'
    ←36,V6#VN0#0
     +10
     V16=VN0
     +991
     V15=V15+1                  'INCR BLOCK COUNT'
     V10=V7-V16
    ←13,V10[23]#0               'NEGATIVE RESULT?'
[27]V15=V15+1
     V10=V10+1
    ←27,V10=V10-V16>0           'CONTINUE TO INCR BLOCK COUNT'
    ←13

[11]←14,V7#VN0#0                'IS CHARA <SPACE>'
     +160
     V7=VN1                     'V7=N.O.W.'
     N2=VN0                     'N2=S.A. OF MODULE AREA'
     +512
     V12=N2                     'SAVE ADDR'

     N1=VN0                     'SET UP FOR MTS'
     AS 4403
     V20=1
     V21=V20
     V22=VN0
     +1000

    ←S4570,5                    'REWIND'

     V21=V15                    'V21 = BLOCK NO.'
    ←S4570,6                    'FIND AND COPY BLOCK TO DIR. AREA'

[18]N1=VN0                      'N1=S.A. OF BLOCK'
     AS 4404
     V16=VN0                    'V16=BLOCK LENGTH'
     +989
```

```
[15]VN2=VN1,N1+1                'COPY BLOCK'
   ←16,V16=V16-1<0
    V0=V0,N2+1
   ←15

[16]←17,V7=V7-VN0<0             'MODULE > 1 BLOCK LENGTH'
    +990
    N1=VN0
    AS 4403
    V20=1                       'SET UP FOR MTS'
    V21=V20
    V22=VN0
    +1000

   ←S4570,3                     'READ NEXT BLOCK'

    V12=V12+VN0
    +990
    N2=V12                      'SET CONTINUE ADDR FOR BLOCK'
   ←18

[17]OCT 70001000                'JUMP TO CONTROL SEG. OF MODULE'
   ←L

[57]V19=2                       'PUNCH BLANK LEADER'
    V16=VN0
    +120
[54]V20=0
   ←S555,0
   ←54,V16=V16-1>0
   ←L

[100]+0
     +0
     +0
     +0
     +0
     +0
     +0
     +0
     +0
     +0

END

A
```

4.3 Library Update Routine

4.3.1 Program description

This routine is designed to make the task of adding modules to the High-Level Definer comparatively easy. With the use of this routine a module, in relocatable binary (RLB) form, may be automatically added to the end of the library. The complete RLB output should be input since the REF3 listing is used by the routine to calculate the length of block to be copied to disc. Note, this means that the RLB output from the assembly stage may be written directly to a disc file and this may then act as the input for the update routine, the REF1 list being skipped automatically.

Before the module is copied to the library, the user is requested to update the directory via the prompt:-

ENTER MODULE NAME -

which is output on stream 4. The name (of not more than SIX characters) should be entered via the teletype and the string terminated with a space or new line. Note, any characters input at this stage will become the name associated with the module. If a mistake is made a control+B command should be input before a terminating character is entered. This will cause a . to be output and reset the program ready to receive the module name again, in full.

The entry of the terminating character for the module name will cause the program to add the new name to the directory and then copy the module to the library. Upon completion, the user is prompted with the command:-

```
UPDATE COMPLETE
PLEASE LOGOUT TO CLOSE FILE
STOP 1
```

The user should now logout to complete the update operation and safeguard the new data.

Note, the Library Update Routine overwrites part of the operating system which, under normal conditions, is

reset before the 'update complete' prompt is given. If any stop other than 'stop 1' arises then the operating system should be reloaded.

4.3.2 Technical Information

The format of the directory is:-

<SP><No. of blocks><SP><NL><Name 1><SP><No. of words><NL>
.....<Name N><SP><No. of words><NL><%>

The directory is on file 0 of pack 6 and is 990 words in length.

The library is organized as 500 blocks of 990 words each, and is on file 1 of pack 6. A module is copied into the smallest integral number of successive blocks needed to accommodate it.

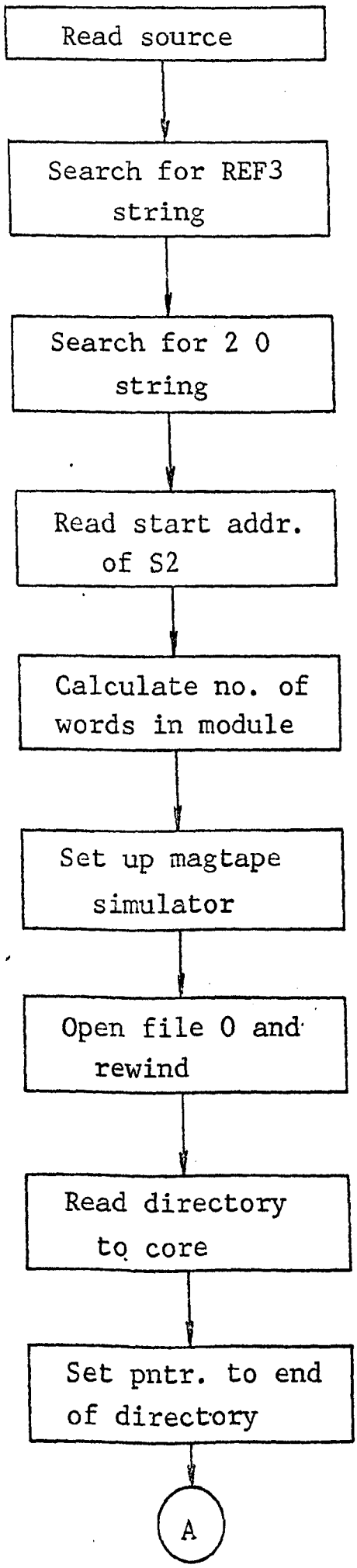
To update the library, the directory is first read to store and a record made of the current number of blocks. The new module name is then added to the end of the directory, overwriting the % terminator which is then replaced at the end of the new name. The number of blocks is then updated to include those required for the new module. The whole directory is then copied back to file 0.

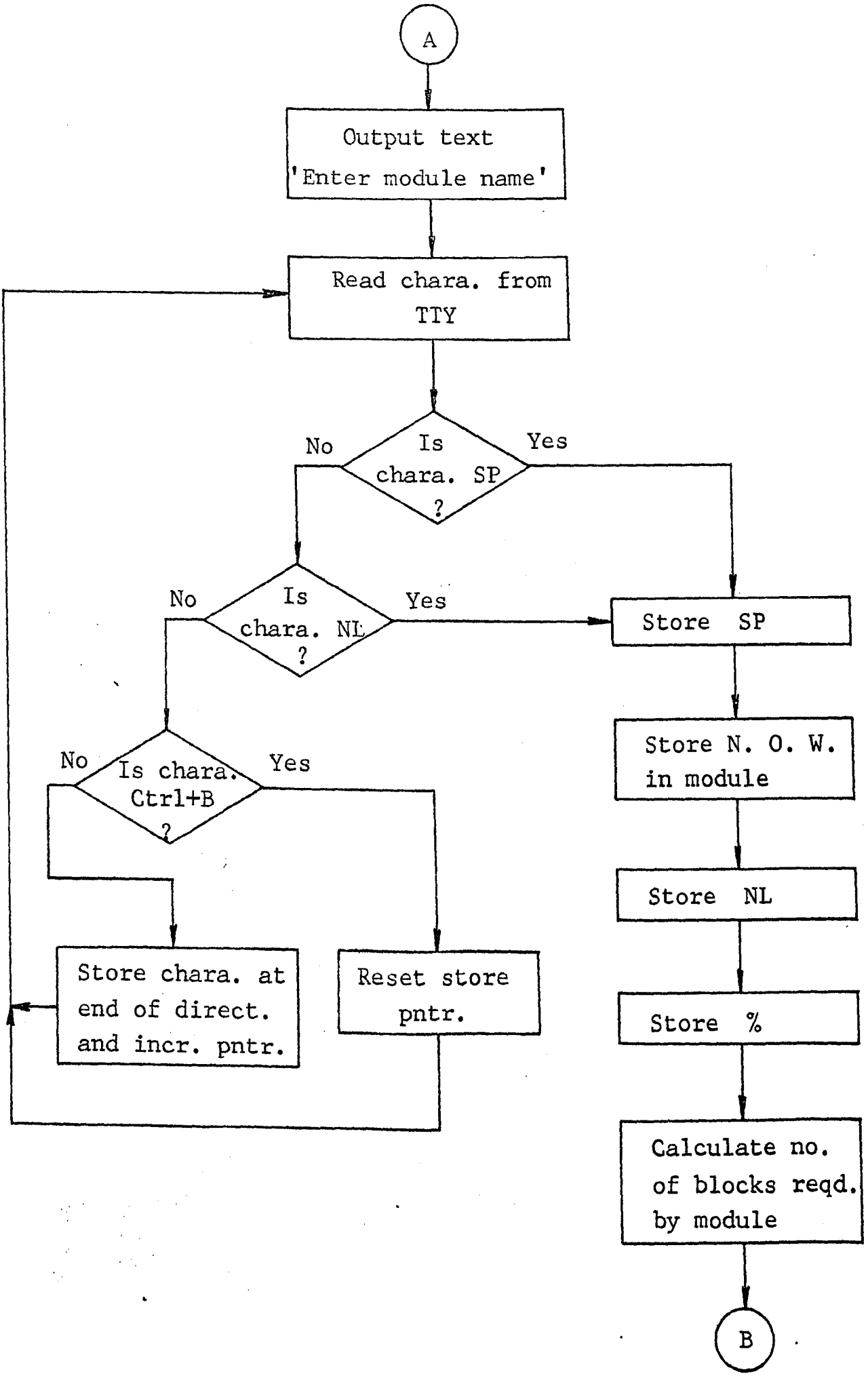
File 1 is then opened and the number of blocks recorded, before the directory was updated, are skipped. The new module is then copied to the appropriate number of blocks at the end of the library.

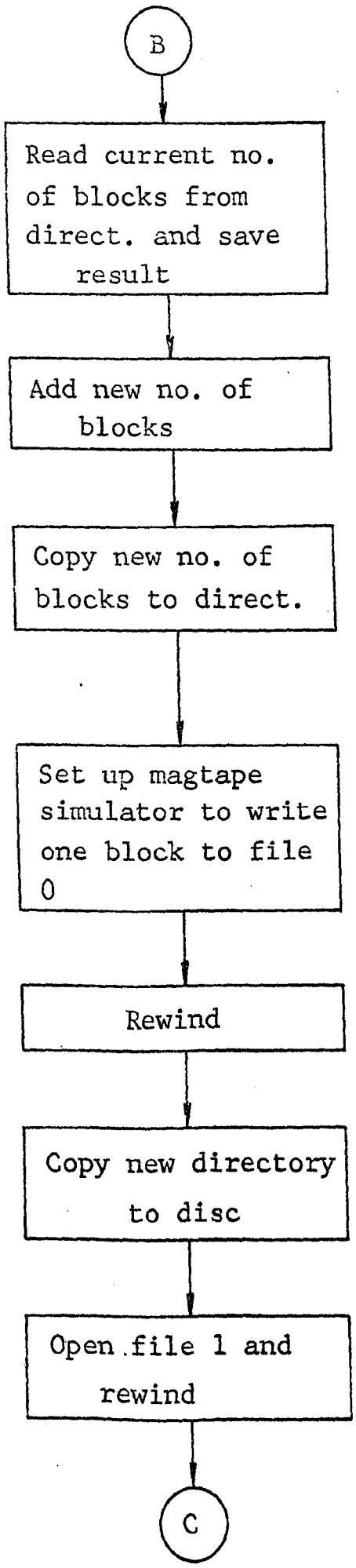
The magtape simulator routines [4] are used for all file and block handling.

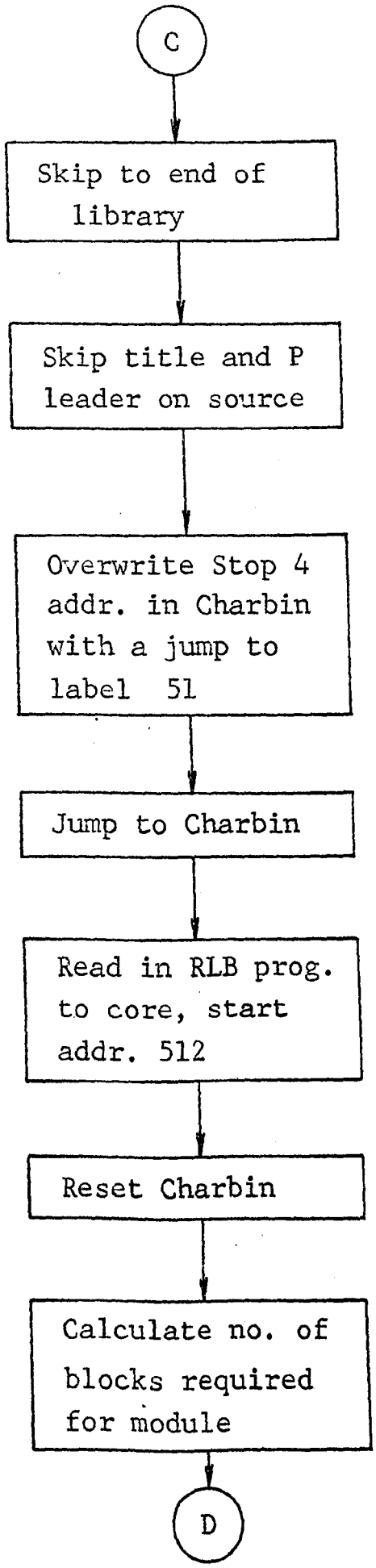
Note, the module to be copied is read into store via 'Charbin' starting at location 512. As a result, the Library Update Routine must be written to an address above the final address required by the module (given by the address of S2 in the REF3 list). A start address of 10000 is recommended.

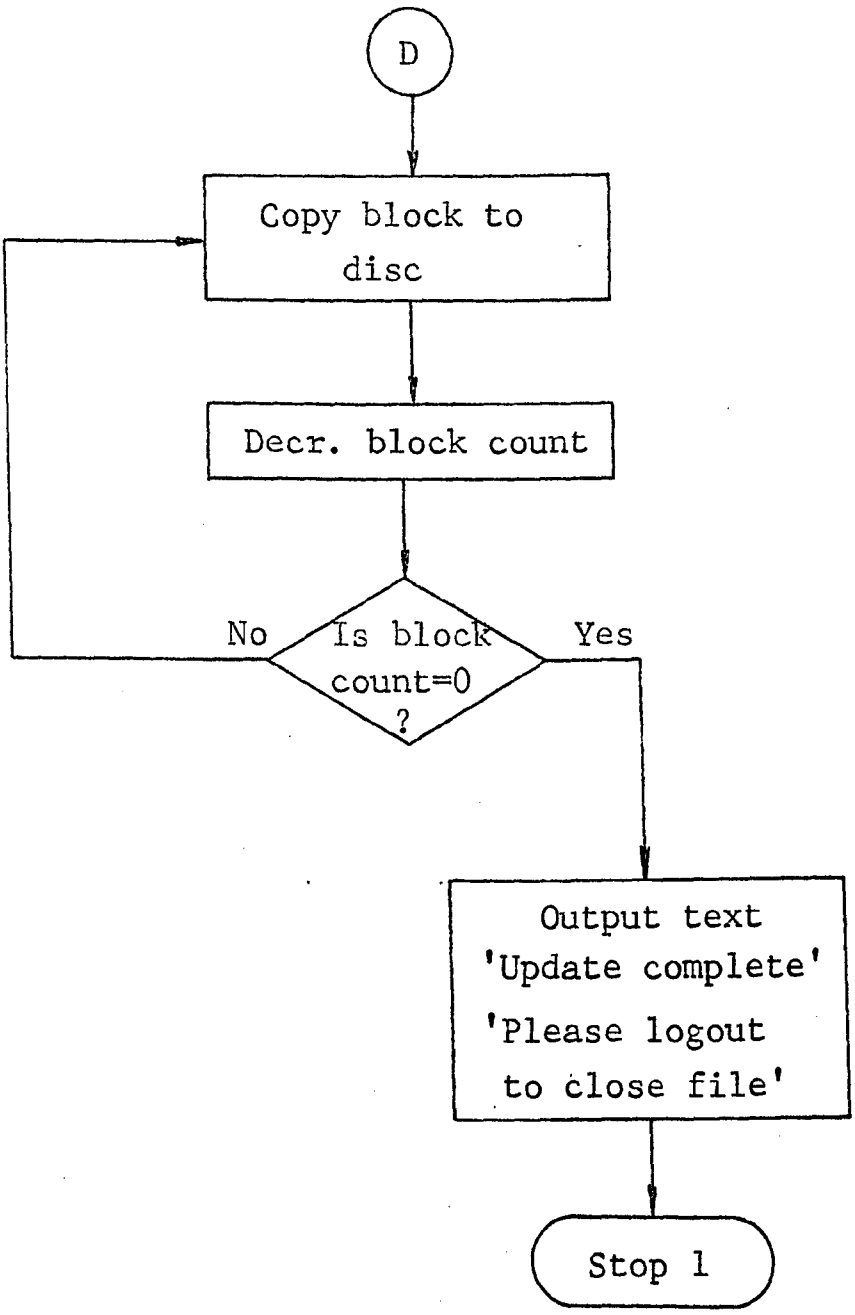
4.3.3 Program details




A
Output text 'Enter module name'
Read chara. from TTY
Is chara. SP ?
No
Yes
Is chara. NL ?
No
Yes
Store SP
Store N. O. W. in module
Store NL
Store %
Calculate no. of blocks reqd. by module
B
Is chara. Ctrl+B ?
No
Yes
Store chara. at end of direct. and incr. pntr.
Reset store pntr.


B
Read current no. of blocks from direct. and save result
Add new no. of blocks
Copy new no. of blocks to direct.
Set up magtape simulator to write one block to file 0
Rewind
Copy new directory to disc
Open file 1 and rewind
C


C
Skip to end of library
Skip title and P leader on source
Overwrite Stop 4 addr. in Charbin with a jump to label 51
Jump to Charbin
Read in RLB prog. to core, start addr. 512
Reset Charbin
Calculate no. of blocks required for module
D


D
Copy block to disc
Decr. block count
Is block count=0 ?
No
Yes
Output text 'Update complete' 'Please logout to close file'
Stop 1

```
N
LIBRARY UPDATE ROUTINE - R.A.C. 25/7/77;

U
[0] V19=3
    ←S555,1
    ←0,V21#VNO=0              'SKIP LEADING SPACES'
     +160
     N1=VNO
     AL 4

[1] V6=VN1,N1+1               'READ CHARA FROM TABLE'
    ←3,V6=0                   'END OF TABLE'
    ←2,V21#V6#0               'CHARAS SAME'
    ←S555,1
    ←1

[2] ←S555,1
    ←2,V21#VNO#0              'READ TO NEXT LINE'
     +10
    ←0

[4] +210                      'R'
     +197                     'E'
     +198                     'F'
     +51                      '3'
     +0

[3] ←S555,1                   'READ REF 3 LIST'
    ←3,V21#VNO=0              'SKIP SPACES'
     +160
     N1=VNO
     AL 10

[5] V6=VN1,N1+1               'READ CHARA FROM TABLE'
    ←7,V6=0                   'END OF TABLE'
    ←6,V21#V6#0               'CHARAS SAME'
    ←S555,1
    ←5

[6]←S555,1
    ←6,V21#VNO#0              'READ TO NEXT LINE'
     +10
    ←3

[7]←S555,1
    ←7,V21#VNO=0              'SKIP SPACES'
     +160
     VN1=V21                  'SAVE M.S. CHARA'
[8]←S555,1
    ←9,V21#VNO=0              'ALL CHARAS READ'
     +160
     VN1=V21,N1+1             'SAVE CHARA'
    ←8

[9] V9=N1-VNO                 'V9=NO. OF CHARAS'
     AL 11
     V7=VN1,N1-1              'V7=L.S. CHARA'
```

```
     V7=V7&VN0
     OCT 17
     V10=10                    'SET SCALE FACTOR'

[12]←16,V9=V9-1<0              'ALL CHARAS READ'
     V8=VN1,N1-1
     V8=V8&VN0                 'V8=NEXT CHARA'
     OCT 17
     V8=V8*V10,I               'SCALE'
     V7=V7+V8                  'ADD TO TOTAL'
     V10=V10*10,I              'ADVANCE SCALE FACTOR'
    ←12

[10]+178                       '2'
     +160                      '<SP>'
     +160                      '<SP>'
     +48                       '0'
     +0

[11]+0
     +0
     +0
     +0
     +0
     +0
     +0
     +0

[16]N1=VN0                     'SET UP FILE MAP FOR MTS'
     AS 4571,0
     VN1=1
     V0=V[N1+2]                'FILE 1 MAP AREA'
     VN1=VN0
     +500

     V20=0                     'V20=FILE NO.'
     V21=V20
     N1=VN0                    'N1=S.A. OF 10 WORD MAP AREA'
     AS 4403

    ←S4570,0                   'OPEN FILE 0'

    ←S4570,5                   'REWIND'

     V20=0                     'RESET PARAMS'
     V21=V20
     N1=VN0                    'RESET PNTR'
     AS 4403
     V22=VN0                   'V22=N.O.W.'
     +1000

    ←S4570,3                   'READ DIRECTORY TO CORE'

     N1=VN0                    'N1=START OF DIRECT'
     AS 4404,3
[20]V6=VN1,N1+1                'READ CHARA'
    ←20,V6#VN0#0               'IS CHARA <NL>'
     +10
     V6=VN1                    'READ NEXT CHARA'
    ←20,V6#VN0#0               'IS CHARA %'
     +165
```

```
    V4=N1               'SAVE ADDR'
[24]V19=4
    TEXT 1P
ENTER MODULE NAME -

[21]V19=5
   ←S555,1              'READ CHARA'
   ←22,V21#VN0=0        'IS CHARA <SPACE>'
    +160
   ←22,V21#VN0=0        'IS CHARA <NL>'
    +10
   ←23,V21#VN0=0        'IS CHARA <CTRL+B>'
    +130
    VN1=V21             'STORE CHARA'
    V0=V0,N1+1
   ←21                  'CONTINUE'

[23]V19=4
    V20=VN0
    +46
   ←S555,0              'OUTPUT .'
    N1=V4               'RESET PNTR'
   ←21

[22]VN1=VN0             'STORE <SPACE>'
    +160
    V7=V7-VN0
    +511
    VN1=V7,N1+1         'STORE N.O.W. IN MODULE'
    VN1=VN0,N1+1        'STORE <NL>'
    +10
    VN1=VN0,N1+1        'STORE %'
    +165

    V9=VN0              'V9 = BLOCK LENGTH'
    +990
    V12=1               'SET BLOCK COUNT'
    V10=V7-V9           'V10 = PROG - BLOCK LENGTH'
[41]←40,V10[23]#0       'PROG > 1 BLOCK ?'
    V12=V12+1           'INCR NO OF BLOCKS'
    V10=V10-V9
   ←41

[40]N1=VN0
    AS 4404,1
    V11=VN1             'SAVE NO. OF BLOCKS'
    V5=VN1+V12
    VN1=V5              'UPDATE NO. OF BLOCKS IN DIRECT'

    V20=0               'RESET FILE NO.'
    V21=V20
   ←S4570,5             'REWIND'

    V22=VN0
    +1000
    N1=VN0              'RESET PNTR'
    AS 4403

   ←S4570,2             'COPY NEW DIRECT TO DISC'
```

```
     V20=1                    'SET TO FILE 1'
     V21=V20
    ←S4570,0                  'OPEN FILE 1'
    ←S4570,5                  'REWIND'
     V21=V11                  'V21=NO. OF BLOCKS'
    ←S4570,7                  'SKIP TO END OF LIB'

     V19=3                    'SKIP TITLE AND P LEADER ON SOURCE'
[35]←S555,1
    ←35,V21#VNO#0             'IS CHARA ; ?'
     +187

    ←S555,1                   'SKIP 3 CHARAS'
    ←S555,1
[36]←S555,1
    ←36,V21#VNO#0             'IS CHARA <NL> ?'
     +10

     N1=VNO                   'STOP 4 ADDR IN CHARBIN'
     +308

     V22=VNO
     OCT 70400000             'FIDS 7004 0 0 0'
     V22=V22+VNO
     AL 51                    'V22 = JUMP TO LABEL 51'

     VN1=V22                  'OVERWRITE CHARBIN'
     V22=3                    'V22 = STREAM NO.'
     V23=VNO                  'V23=RELOCATING ADDR'
     +512

     OCT 70400356             'FIDS 7004 0 7 14 (238)'

[51]N1=VNO                    'OVERWRITTEN ADDR IN CHARBIN'
     +308
     V22=VNO
     OCT 03054004             'FIDS 0110 22 0 4'
     VN1=V22                  'RESET CHARBIN'

     V8=0                     'ZERO BLOCK COUNT'
     N2=VNO
     +512                     'N2=S.A. OF MODULE IN CORE'

[33]N1=VNO
     AS 4404
     V9=VNO                   'V9=BLOCK LENGTH'
     +989

[15]VN1=VN2,N2+1              'COPY BLOCK TO S4404 AREA'
     V0=V0,N1+1
    ←15,V9=V9-1>0

[31]←25,V7=V7-VNO<0           'MODULE LENGTH>BLOCK LENGTH'
     +990
     V8=V8+1                  'INC BLOCK COUNT'
[25]V22=VNO
     +1000
     N1=VNO                   'N1=S.A. OF MAP AREA'
```

```
    AS4403
    V20=1                    'RESET FILE NO.'
    V21=V20

   ←S4570,2                  'COPY BLOCK TO DISC'

   ←33,V8=V8-1>0
    V19=4
    TEXT 1P 1E
UPDATE COMPLETE
    TEXT 1P 1E
PLEASE LOGOUT TO CLOSE FILE
    V22=1
   ←S555,6                   'STOP 1'

END

A
```

4.4 Module Preparation

For the purpose of the High-Level Definer, a module is a self-contained program, written in the programming language of a master computer and used to generate blocks of assembly language code for another (slave) computer, or microprocessor. These modules are all held in a module library (file 1, pack 6) and are assigned names (see Library Update Routine, sect. 4.3) in such a way that they may be called via a high-level call, using the appropriate name. At present, the master computer is a Ferranti FM1600B minicomputer and the modules are written in its assembly language, FIXPAC. As a result, a working knowledge of both the operating system and assembly language of this machine will be required by any user who wishes to write new modules for addition to the Definer's library.

When the required module has been written, as a master program, it should be assembled in the normal manner and a relocatable binary (RLB) version obtained, complete with a reference three (REF3) list (a list of all subroutines called, and their relative addresses). The REF3 list is used by the Library Update Routine to calculate the number of blocks required by the module when copied to the definer library.

Modules will, in general, be sub-divided into a number of smaller functional units, or segments, each responsible for the generation of a specific block of code which, when linked together, go to make up the complete block of assembly code requested by the high-level call. Further, a control segment will be required for most applications to decode any parameters specified in the call.

The line containing the high-level call is copied to a buffer area, start address +11800 (assuming the High-Level Definer to be loaded from location 10000) and V11 specifies whether the module is to be output as a macro or

subroutine:-

V11 = 1 - subroutine output

V11 = 0 - macro output

Note, for a subroutine output the following text is output before control is passed to the module:-

REL <SP> NAME <CR><NL>

NAME <SP>

Control is then passed to the module via a link-storing jump to location 512, where the module will be loaded from the backing library when called.

In order to avoid accidental corruption of any registers required by the High-Level Definer, all registers used in a module should be stored on entry and restored on exit. A 'dummy' call to S2 is also required, which generates a blank data subroutine of unspecified length, assembled at the end of the user program after all other subroutines, and extending to the top of store. The start address of S2 is used by the Library Update Routine to give the final address of the complete module in order to calculate its length for addition to the library.

As an illustrative example, consider the multiply module. The calling string for this module is of the form:-

or @MULTI <SP>(U,8)

The characters contained within the brackets define the form the routine is to take, in this case an unsigned, eight-bit multiply.

The first segment of the module is a control segment which loads the calling line from location +11800 and decodes the parameters, checking for any errors. Various flags are set at this stage which are used during the output of the text to determine which of the other segments are required. The sequence is as shown in Figure 4.4.1 and for this example segments (B)(D) and (F) would be output.

Program listings for the multiply module can be found in section 4.6.2.

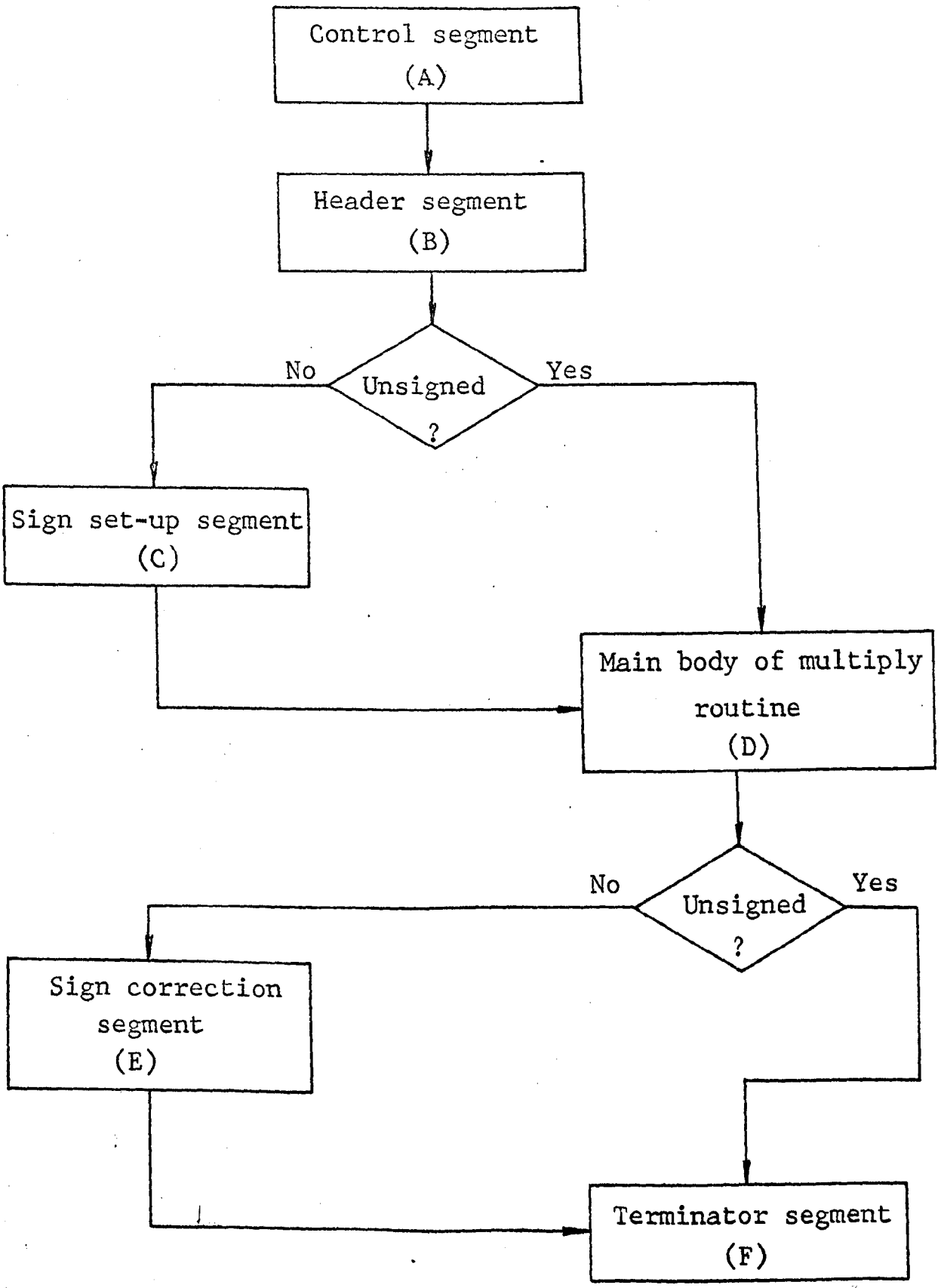


Figure 4.4.1 Control segment for multiply routine.

4.5 Module Generator

4.5.1 Program description

This is an optional pre-pass routine designed to assist in the preparation of modules for use with the High-Level Definer. It relieves the user of the task of inserting text statements within a block of assembly code. Instead, a text begin (TEXT B) statement is inserted at the beginning of the block and a text finish (TEXT F) at the end.

The format of the TEXT B and TEXT F statements is:-

TEXT <SP> B <NL>

Leading spaces are NOT permissible. If they are inserted, the TEXT B and TEXT F commands will be transferred directly to the output. No TEXT 1E statements will be inserted if the error occurs in the TEXT B command. If the error is in the TEXT F command the effect is to output this and all subsequent commands as part of the block of assembly code. This will continue until the first valid TEXT F command is encountered. Leading deletes ARE permissible.

The module generator reads in the source code on stream 3 and outputs the object code on stream 2 with all the necessary TEXT 1E statements inserted (i.e. in a format suitable for the FM1600B). A <NL><%> character string is needed to terminate the program.

Various messages may be output on stream 4, during the excution of the program:-

(i) TEXT B ERROR
STOP 11

An error has been encountered in a TEXT B command. A RUN command causes the erroneous statement to be output and the run continued.

(ii) TEXT F ERROR
STOP 11

An error has been encountered in a TEXT F command. A

RUN command causes the erroneous statement to be output and the run continued.

(iii) ERROR RUN

An error has been encountered during the course of the run and the error flag set. A RUN command causes the program to be reset and a new pass initiated.

(iv) RUN COMPLETE
STOP 1

The run has been successfully completed. A RUN command causes the program to be reset and a new pass initiated.

4.5.2 Example

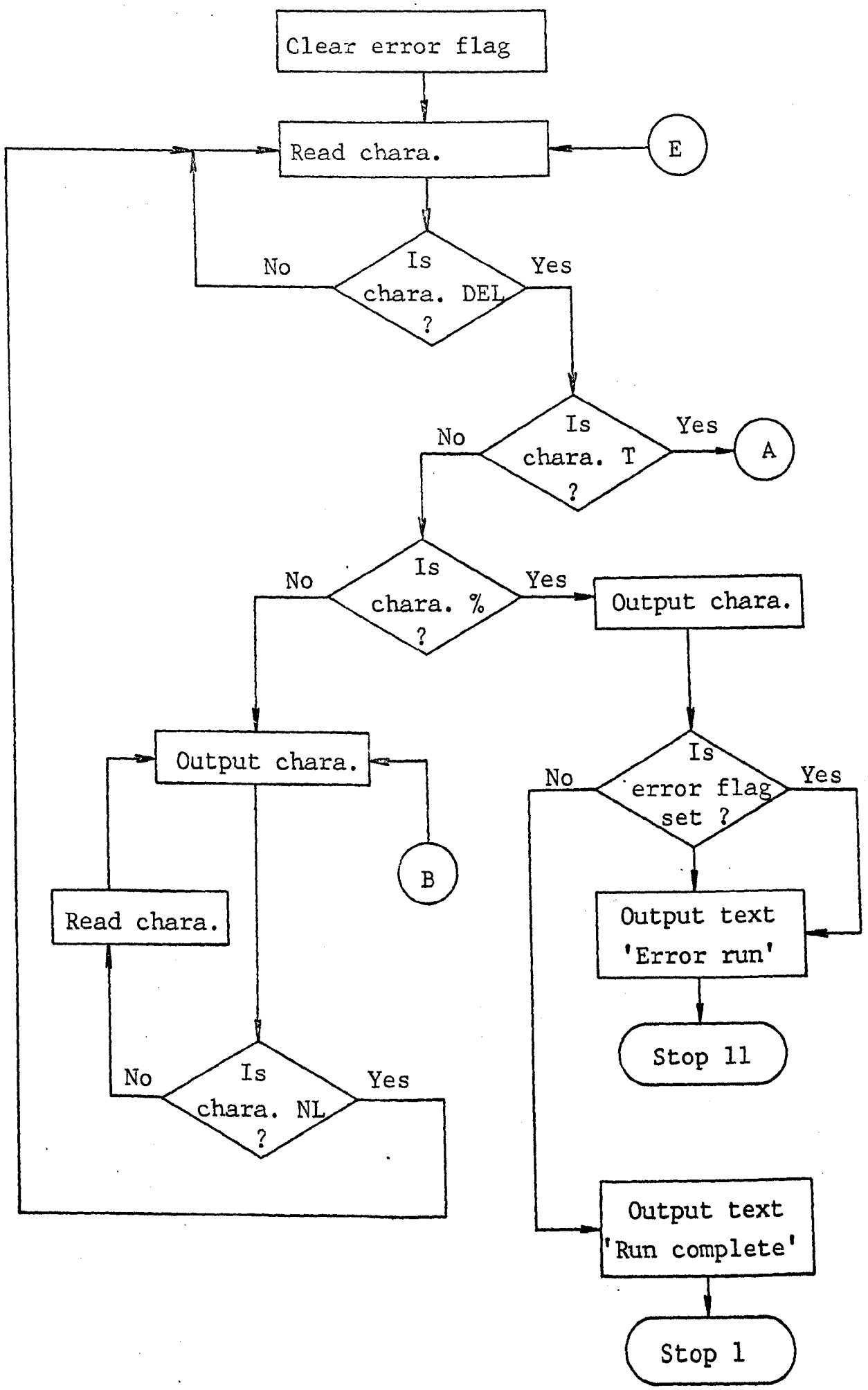
Input code:-

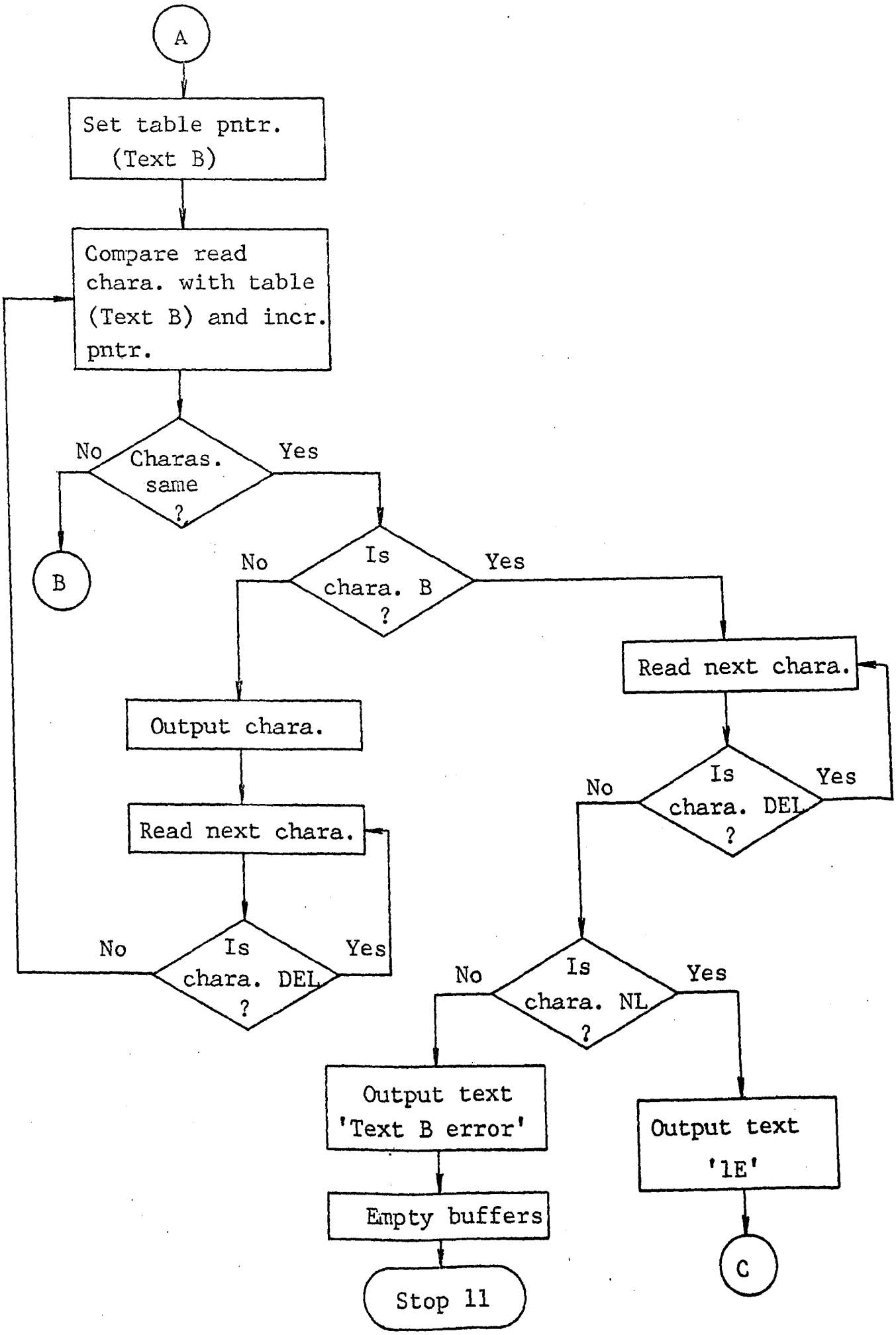
```
[14] TEXT 1E
     PSHR R2
    ←7,V6=0
TEXT B
     TSTR R1
     BPL £MULIO
     NEGR R0
TEXT F
```

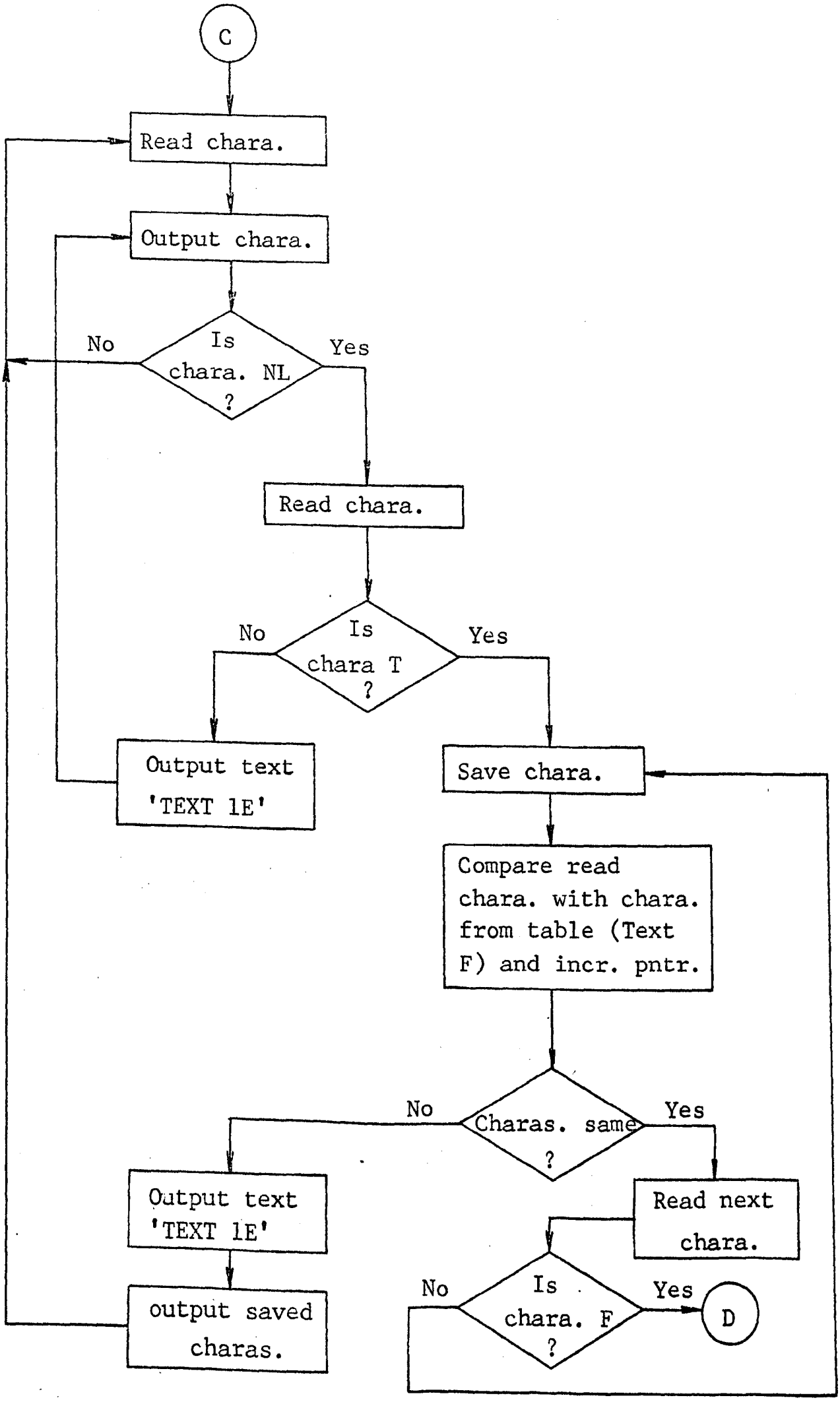
Output code:-

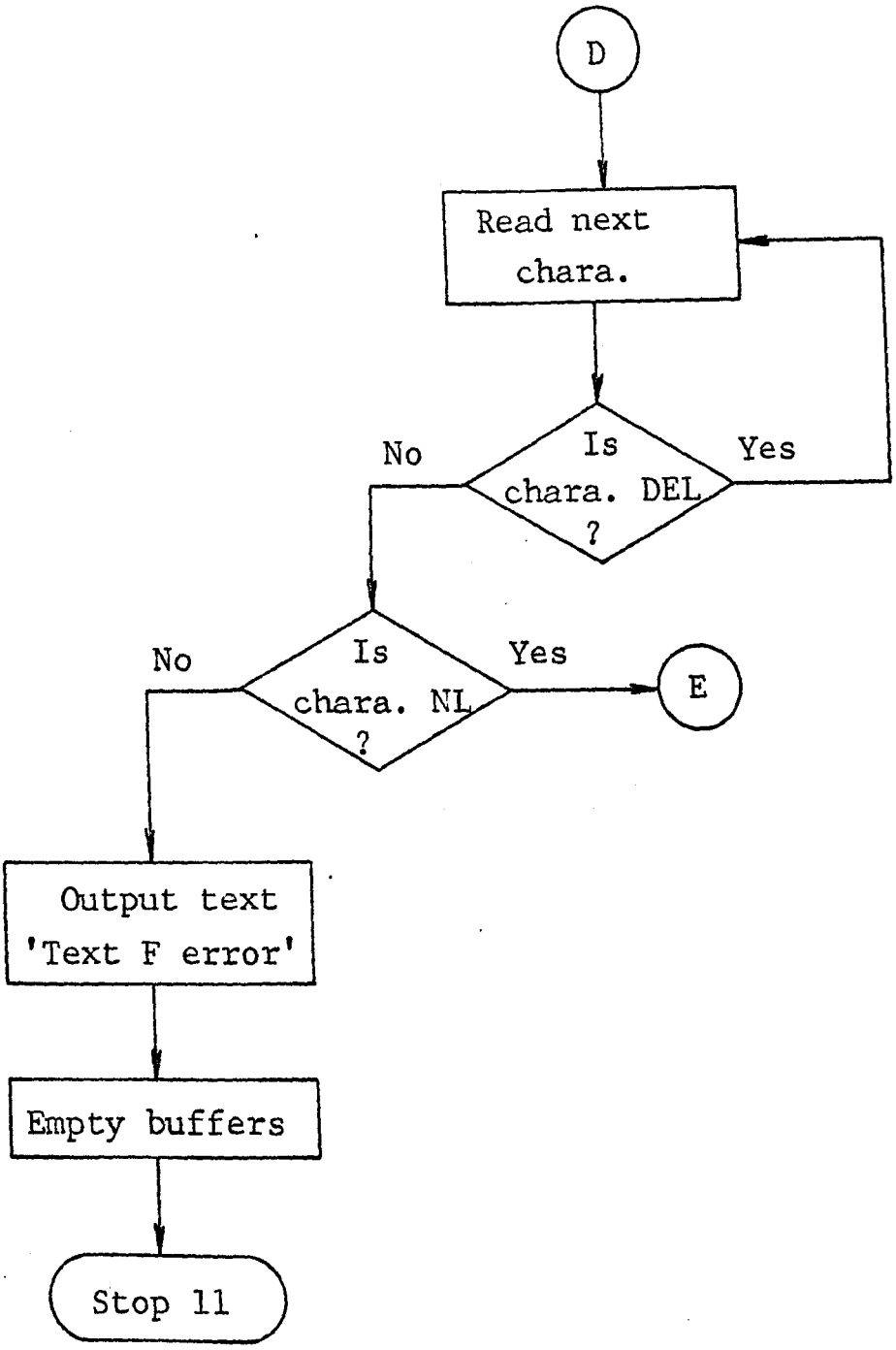
```
[14] TEXT 1E
     PSHR R2
    ←7,V6=0
TEXT 1E
     TSTR R1
TEXT 1E
     BPL £MULIO
TEXT 1E
     NEGR R0
```

4.5.3 Program details




A
Set table pntr. (Text B)
Compare read chara. with table (Text B) and incr. pntr.
Charas. same ?
No
Yes
B
Is chara. B ?
No
Yes
Output chara.
Read next chara.
Is chara. DEL ?
No
Yes
Read next chara.
Is chara. DEL ?
No
Yes
Is chara. NL ?
No
Yes
Output text 'Text B error'
Empty buffers
Stop 11
Output text '1E'
C


C
Read chara.
Output chara.
Is chara. NL ?
No
Yes
Read chara.
Is chara T ?
No
Yes
Output text 'TEXT 1E'
Save chara.
Compare read chara. with chara. from table (Text F) and incr. pntr.
Charas. same ?
No
Yes
Output text 'TEXT 1E'
Read next chara.
output saved charas.
Is chara. F ?
No
Yes
D


D
Read next chara.
Is chara. DEL ?
No
Yes
Is chara. NL ?
No
Yes
E
Output text 'Text F error'
Empty buffers
Stop 11

```
N
MODULE GENERATOR - R.A.C. 12/6/77;

U
[5] V7=0                    'RESET ERROR FLAG'

[0] V19=3
    ←S555,1
    ←1,V21#VNO=0            'IS CHARA T'
     +212
    ←2,V21#VNO=0            'IS CHARA %'
     +165
    ←0,V21#VNO=0            'IS CHARA <DEL>'
     +255

[3] V20=V21
     V19=2
    ←S555,0                 'OUTPUT LINE'
    ←0,V21#VNO=0            'IS CHARA <NL>'
     +10
     V19=3
    ←S555,1                 'READ NEXT CHARA'
    ←3

[1] N1=VNO
     AL 10                  'TEXT B TABLE'
[7] V6=VN1,N1+1
    ←4,V6#V21=0             'CHARAS SAME'
    ←3

[4]←6,V21#VNO=0             'IS CHARA B'
     +66
     V19=2
     V20=V21
    ←S555,0                 'OUTPUT CHARA'
     V19=3
[24]←S555,1                 'READ NEXT CHARA'
    ←24,V21#VNO=0           'IS CHARA <DEL>'
     +255
    ←7

[6] V19=3
    ←S555,1
    ←6,V21#VNO=0            'IS CHARA <DEL>'
     +255
    ←8,V21#VNO#0            'IS CHARA <NL>'
     +10
     V19=2
     TEXT 1E
1E
    ←23

[11]V19=3
    ←S555,1
    ←11,V21#VNO=0           'IS CHARA <DEL>'
     +255
    ←9,V21#VNO=0            'IS CHARA T'
     +212
```

```
    V19=2
    TEXT 1E
 TEXT 1E
[12]V20=V21
    V19=2
   ←S555,0                  'OUTPUT LINE'
   ←11,V21#VN0=0
    +10
[23]V19=3
   ←S555,1
   ←12

[9] N1=VN0
    AL 13
    N2=VN0
    AL 14

[15]V6=VN1,N1+1
    VN2=V6,N2+1
   ←16,V6#V21=0             'CHARAS SAME'

    VN2=VN0,N2+1            'STORE %'
    +165
    N2=VN0
    AL 14
    V19=2                   'RESET PNTR'
    V0=V0,N2+1
    TEXT 1E
 TEXT 1E
[18]V20=VN2,N2+1
   ←17,V20#VN0=0            'IS CHARA %'
    +165
   ←S555,0                  'OUTPUT CHARA'
   ←18

[16]V19=3
   ←S555,1
   ←16,V21#VN0=0            'IS CHARA <DEL>'
    +255
   ←15,V6#VN0#0             'IS CHARA F'
    +198
   ←19,V21#VN0#0            'IS CHARA <NL>'
    +10
   ←0

[17]V19=3
   ←S555,1                  'READ NEXT CHARA'
   ←12                      'CONTINUE TEXT OUTPUT'

[8] V19=4
    TEXT 1P 1E
TEXT B ERROR
   ←20
[19]V19=4
    TEXT 1P 1E
TEXT F ERROR
[20]V19=3
    V22=11
   ←S555,2                  'STOP 11'
```

```
     V7=1
    ←23                    'CONTINUE FROM ERROR'

[2] V19=2
     V20=V21
    ←S555,0                'OUTPUT %'
     V19=4
    ←21,V7≠0
     TEXT 1P 1E
RUN COMPLETE
     V22=1
    ←22                    'STOP 1'

[21]TEXT 1P 1E
ERROR RUN
     V22=11                'STOP 11'
[22]V19=3
    ←S555,2
    ←5                     'RESTART'

[10]+212                   'T'
     +197                  'E'
     +216                  'X'
     +212                  'T'
     +160
     +66                   'B'

[13]+212                   'T'
     +197                  'E'
     +216                  'X'
     +212                  'T'
     +160
     +198                  'F'

[14]+0
     +0
     +0
     +0
     +0
     +0
     +0
     +0
     +0

END

A
```

4.6 Module Library

Two sample modules are included as a guide to the general format of modules for use with the High-Level Definer.

4.6.1 Define Registers Module

This module may be used to generate the code needed to specify the eight internal registers of the CP1600. It should normally be called as a macro at the beginning of the user program.

Specification:-

No. of words	771
Format	relocatable binary
Pack No.	6
File No.	1
Block No.	1

```
N
DEFINE REGISTERS MODULE (DEFREG) - R.A.C. 20/7/77

U
     V19=2
     TEXT 1E
R0 = 0
     TEXT 1E
R1 = 1
     TEXT 1E
R2 = 2
     TEXT 1E
R3 = 3
     TEXT 1E
R4 = 4
     TEXT 1E
R5 = 5
     TEXT 1E
SP = 6
     TEXT 1E
PC = 7

     ←L

     V0=VN0
     AS 2
END

A
```

4.6.2 Multiply Module

This module is capable of generating a multiply routine in CP1600 assembly code. Any number of bits, up to sixteen, may be specified with sign handling, if required. The options are defined by two parameters appearing in the order:-

(No. of bits, Signed (S) or Unsigned (U))

The calling sequence is:-

(or @) MULTI <SP> (8,U)><CR <NL>

for an eight-bit, unsigned multiply routine.

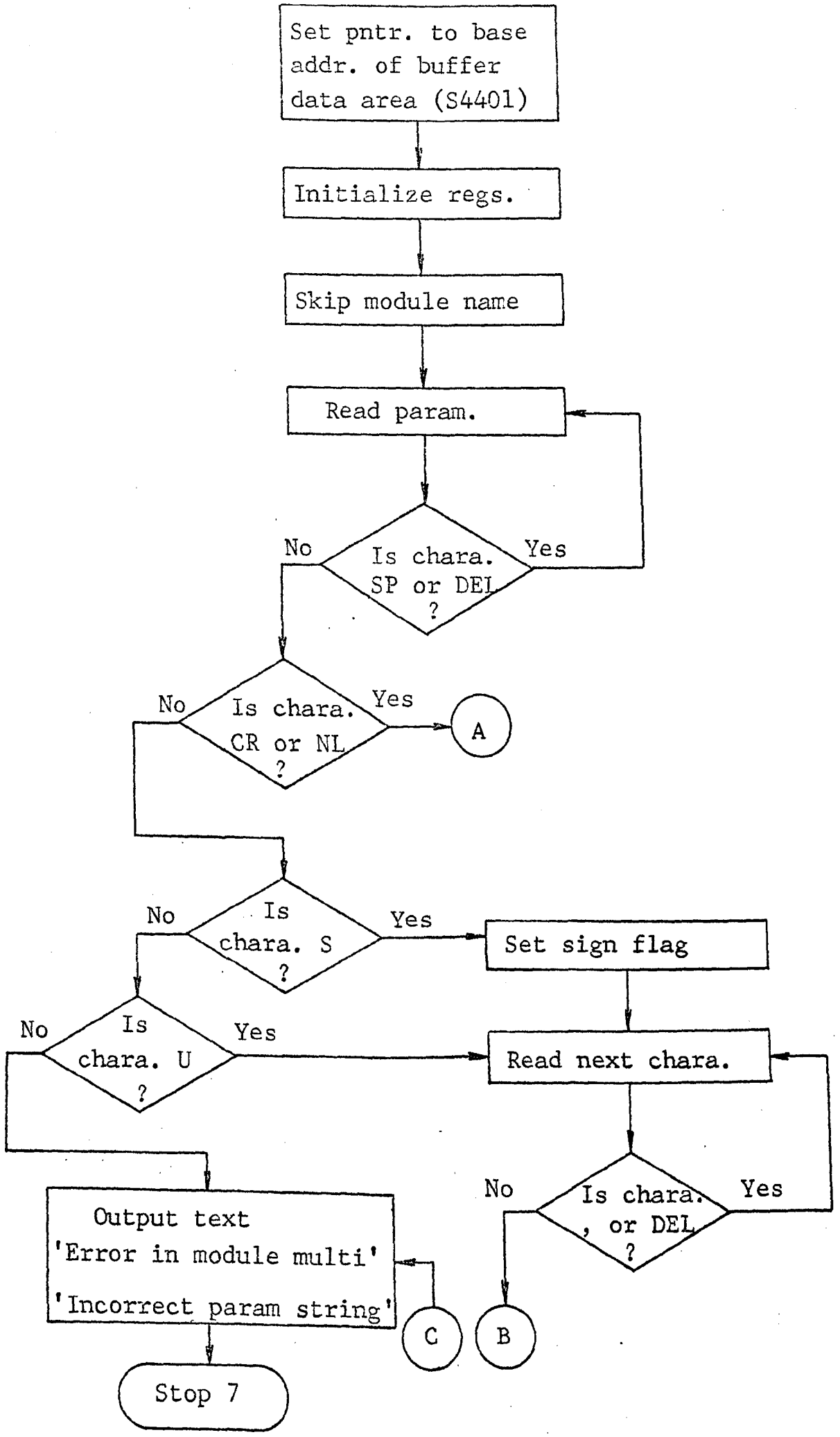
If an incorrect parameter string is specified an error stop will occur:-

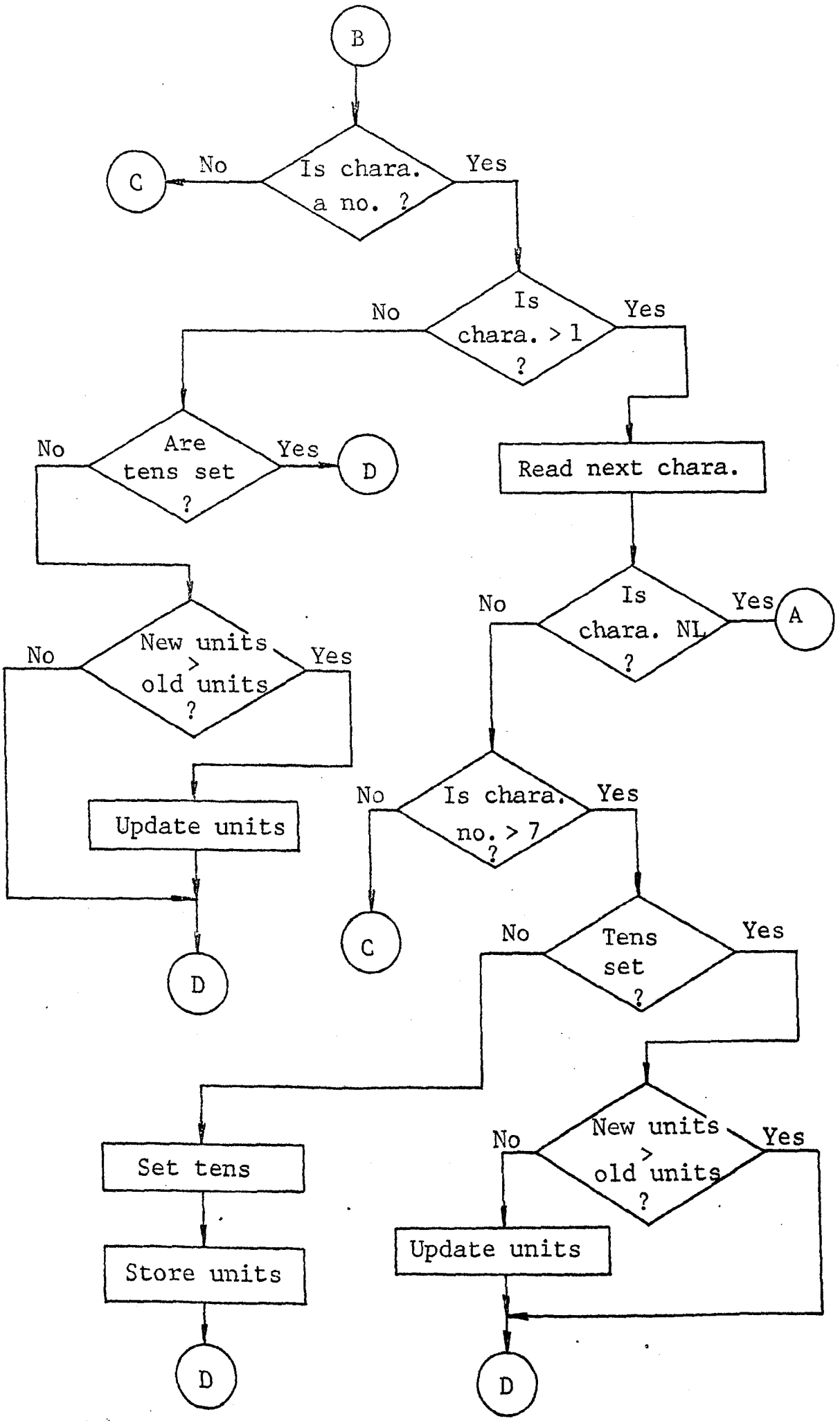
```
ERROR IN MODULE MULTI
INCORRECT PARAM STRING
STOP 7
```

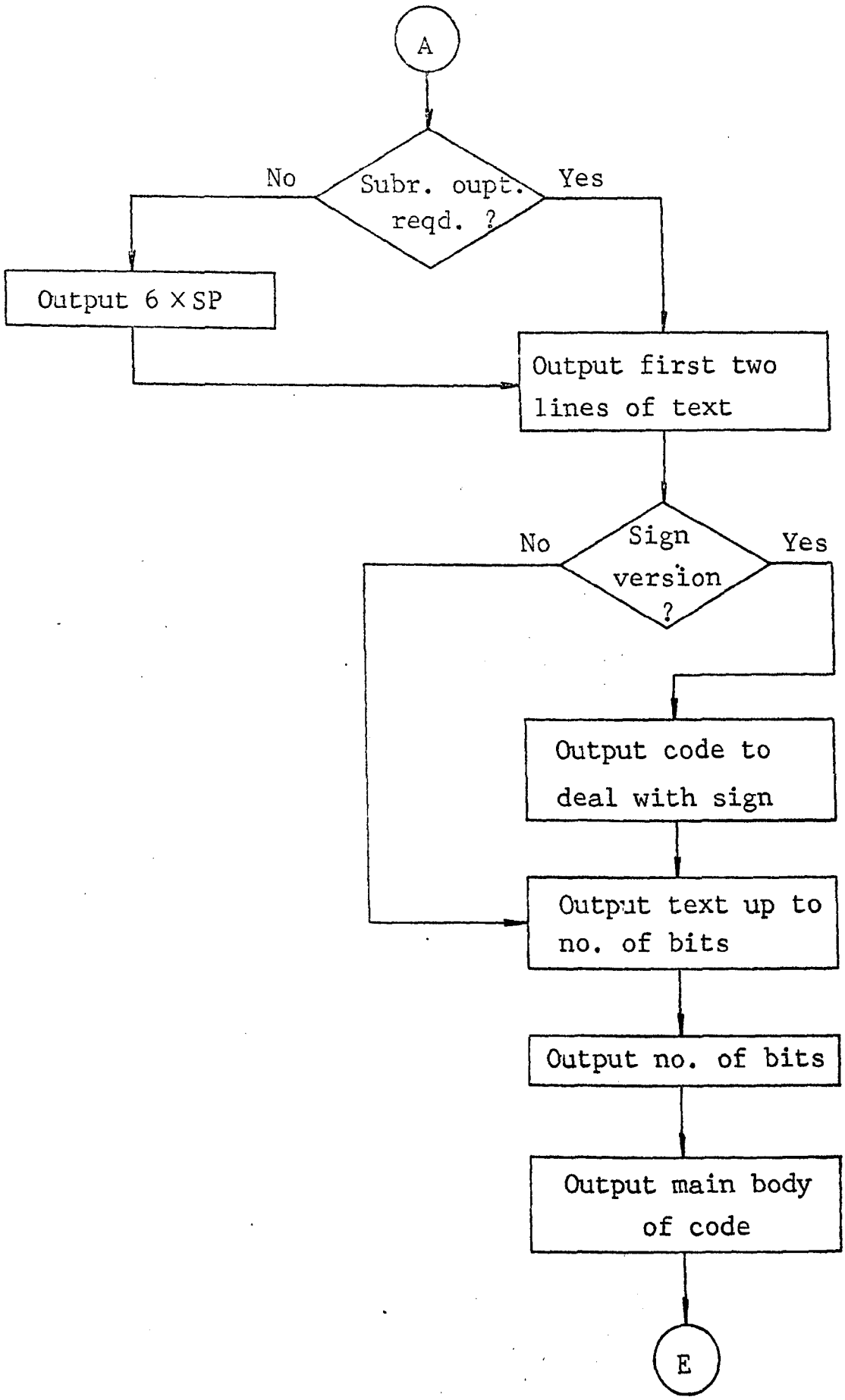
Specification:-

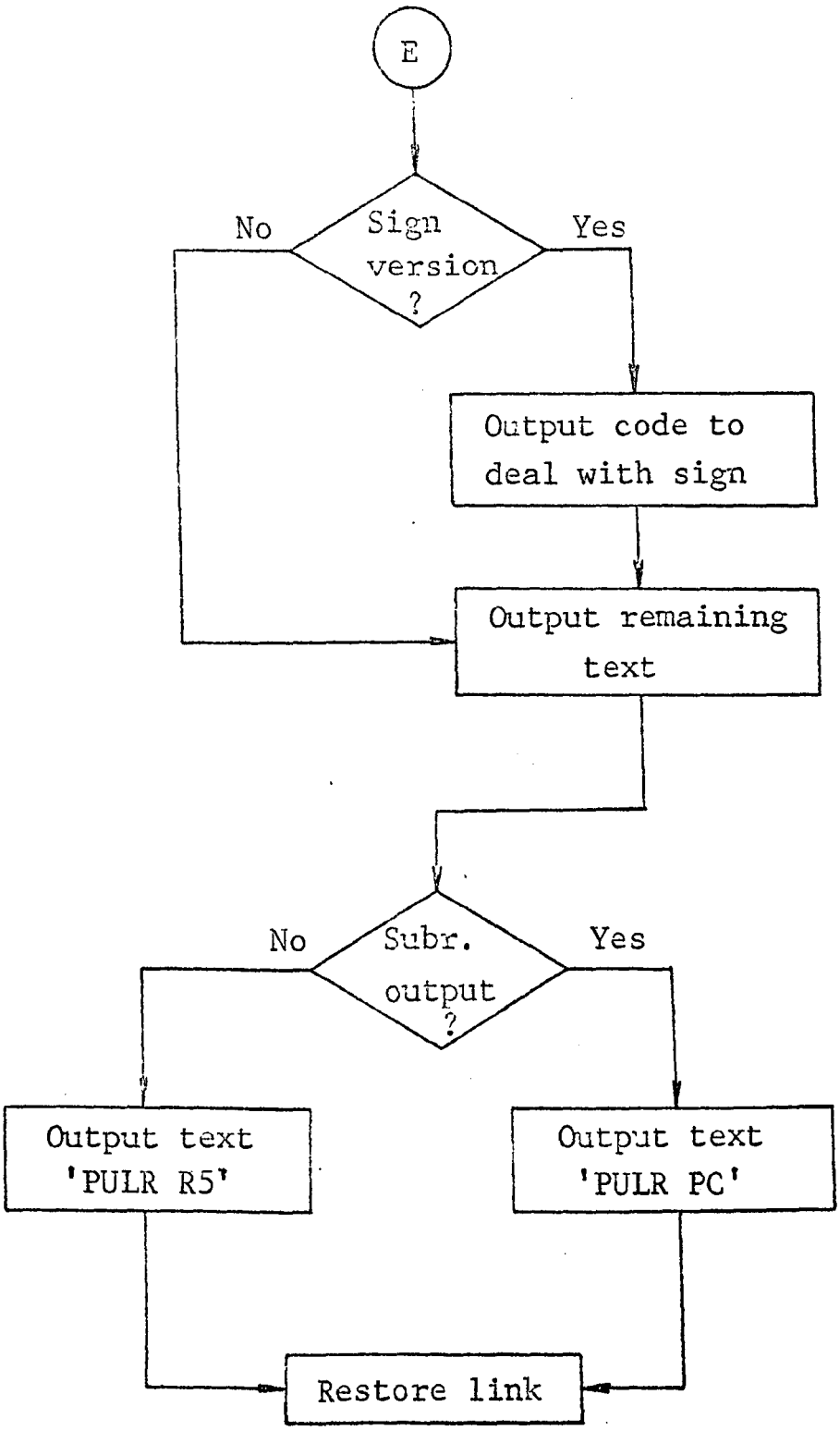
No. of words (including all necessary subroutines)	1190
Format	relocatable binary
Pack No.	6
File No.	1
Block No.	2 and 3

4.6.2.1 Program details




B
Is chara. a no. ?
No
Yes
C
Is chara. > 1 ?
No
Yes
Are tens set ?
No
Yes
D
Read next chara.
New units > old units ?
No
Yes
Update units
D
Is chara. NL ?
No
Yes
A
Is chara. no. > 7 ?
No
Yes
C
Tens set ?
No
Yes
Set tens
Store units
D
New units > old units ?
No
Yes
Update units
D


A
Subr. oupt. reqd. ?
No
Yes
Output 6 × SP
Output first two lines of text
Sign version ?
No
Yes
Output code to deal with sign
Output text up to no. of bits
Output no. of bits
Output main body of code
E


E
Sign version ?
No
Yes
Output code to deal with sign
Output remaining text
Subr. output ?
No
Yes
Output text 'PULR R5'
Output text 'PULR PC'
Restore link

```
N
MULTIPLY MODULE (MULTI) - R.A.C. 3/8/77;

U
     STK VN3,9,6
     VN3=N1,N3-1              'SAVE REGS'

     N1=VN0
     +11800                   'N1=BASE OF LABEL TABLE'
     V5=0
     V6=0
     V4=0                     'INITIALISE REGS'

[0]  V7=VN1,N1+1
    ←0,V7#VN0#0               'IS CHARA <SPACE>'
     +160
[1]  V7=VN1,N1+1              'READ PARAM'
    ←1,V7#VN0=0               'IS CHARA <SPACE>'
     +160
    ←6,V7#VN0=0               'IS CHARA <NL>'
    +10
    ←2,V7#VN0=0               'IS CHARA S'
     +83
    ←3,V7#VN0=0               'IS CHARA U'
     +85
    ←1,V7#VN0=0               'IS CHARA <DEL>'
     +255

[4]  V19=4
     TEXT 1P 1E
ERROR IN MODULE MULTI
     TEXT 1P 1E
INCORRECT PARAM STRING
     V22=7
    ←S555,6                   'STOP 7'

[2]  V6=1                     'SET SIGN FLAG'
[3]  V7=VN1,N1+1
    ←3,V7#VN0=0               'IS CHARA ,'
     +172
    ←3,V7#VN0=0               'IS CHARA <DEL>'
     +255
     V8=V7&VN0
     OCT 17
    ←4,V8-10>0                'CHECK FOR NO. INPUT'
    ←4,V8-1<0

    ←8,V8-1#0                 'IS CHARA 1'
     V7=VN1,N1+1
    ←6,V7#VN0=0               'IS NEXT CHARA <NL>'
     +10

     V8=V7&VN0
     OCT 17
    ←4,V8-7>0                 'CHECK FOR NO. IN < 7'
    ←10,V8-1>0
    ←4,V8#0
[10]←15,V4-1=0                'TENS ALREADY SET'
     V4=1
```

```
    V5=V8                    'STORE UNITS'
   ←1
[15]←1,V8-V5<0               'NEW DIGIT>OLD DIGIT'
    V5=V8
   ←1                        'CONTINUE'

[8]←1,V4-1=0                 'IS UPPER DIGIT 1'
[9]←1,V8-V5<0                'NEW DIGIT>OLD DIGIT'
    V5=V8
   ←1                        'CONTINUE'

[6] V19=2
   ←14,V11#0                 'SUBROUTINE OUTPUT ?'
    V7=5
[5] V20=VN0                  'OUTPUT 5 SPACES'
    +160
   ←S555,0
   ←5,V7=V7-1>0
[14]TEXT 1E
 PSHR R5        ;SAVE R5
    TEXT 1E
       PSHR R2        ;SAVE R2

   ←7,V6-1#0                 'SIGN BIT SET'

    TEXT 1E
       TSTR R1        ;CHK MULTIPLIER SIGN
    TEXT 1E
       BPL £MUL10     ;POSITIVE, OK
    TEXT 1E
       NEGR R0        ;NEGATE MULTIPLICAND
    TEXT 1E
       BOV £MUL13
    TEXT 1E
       NEGR R1        ;AND MULTIPLIER
    TEXT 1E
       BOV £MUL14

[7] TEXT 1E
£MUL10 MOVR R0,R2     ;MULTIPLICAND TO R2
    TEXT 1E
       CLRR R0        ;INIT PARTIAL PRODUCT
    TEXT
       MVII .

   ←11,V4=0                  'ANY TENS'
    V4=10
[11]V20=V4+V5                'V20=NO. OF BITS'
    V20=V20+1
    V23=VN0
    +224
   ←S1                       'OUTPUT NO. OF BITS'

    TEXT 1E
,R5    ;LOOP COUNT
    TEXT 1E
£MUL11 SARC R0        ;SHFT MS PART PROD
    TEXT 1E
£MUL12 RRC R1,1       ;SHFT LS PART PROD
    TEXT 1E
```

```
      DECR R5          ;DECR LOOP COUNT
    TEXT 1E
      BZE £MUL15       ;DONE, EXIT
    TEXT 1E
      BNC £MUL11       ;CHK SHFTED OUT BIT
    TEXT 1E
      ADDR R2,R0       ;ADD MULTIPLICAND TO P.P.
    TEXT 1E
      BNOV £MUL11      ;CHK OVERFLOW
    TEXT 1E
      RRC R0,1         ;SHFT IN CARRY
    TEXT 1E
      B £MUL12

  ←16,V6=0              'SIGN FLAG SET'

    TEXT 1E
£MUL13 MOVR R1,R0      ;LARGEST NEG NUMBER
    TEXT 1E
      NEGR R0
    TEXT 1E
£MUL14 CLRR R1

[16]TEXT 1E
£MUL15 PULR R2         ;RESTORE R2
  ←21,V11=0             'MACRO OUTPUT'
    TEXT 1E
      PULR PC          ;EXIT
  ←20
[21]TEXT 1E
      PULR R5          ;RESTORE R5

[20]N1=VN3
    V0=V0,N3+1
    LDK VN3,9,6         'RESTORE REGS'
  ←L

    V0=VN0
    AS 2

END

A
```

4.7 References

1. Software Control Module, User Information Document (Edition 2), SCM UID 60-58-258, Ferranti Ltd., 1973.

2. Druid Program Development System, User Information, UID 59, Ferranti Ltd., 1974.

3. Fixpac Programmers Manual, Ferranti Ltd., 1971.

4. A User Guide to the Magtape Simulator Control Program, Internal Report, Buffam C. J., June 1976.

5. The Software Realization of Instruments, International Microcomputers/Minicomputers/Microprocessors '77, Geneva. Proceedings published by IPC Science and Technology Press, pp41-47, Comley R. A. and Hewish T. R., 1977.

Chapter 5

High - Speed Reader Interface

5.1 Introduction

The circuit described is designed to act as an interface between the GIMINI microcomputer high-speed reader channel and an ADDMASTER high-speed paper tape reader. The GIMINI channel is designed to interface directly with a 350 character per sec. reader and so must be modified to facilitate the link with the ADDMASTER reader, which runs at 150 characters per sec..

5.2 Circuit Description

The drive left (DRV LFT) signal from the GIMINI is used to generate a 0.5ms clock pulse for the reader by use of a monostable (see Fig. 5.2). The clock (CK) output is taken via a second monostable to provide the high-speed reader data ready (HSR DATA RDY) signal approximately 6ms later.

The high-speed reader ready (HSR READY) signal is taken via an inverter to the reader to make it of the correct polarity.

The complete interface circuit is as shown in Figure 5.1. The circuit is laid-out as shown in Figure 5.3 and is constructed on a small piece of vero-board which is attached to the edge-connector of the high-speed reader.

5.3 Miscellaneous Details

5.3.1 Component list

Resistors

All ¼w ± 5%

R1	22Ω
R2,R6	2.2k
R3	180Ω
R4,R5	20k
R7	220Ω

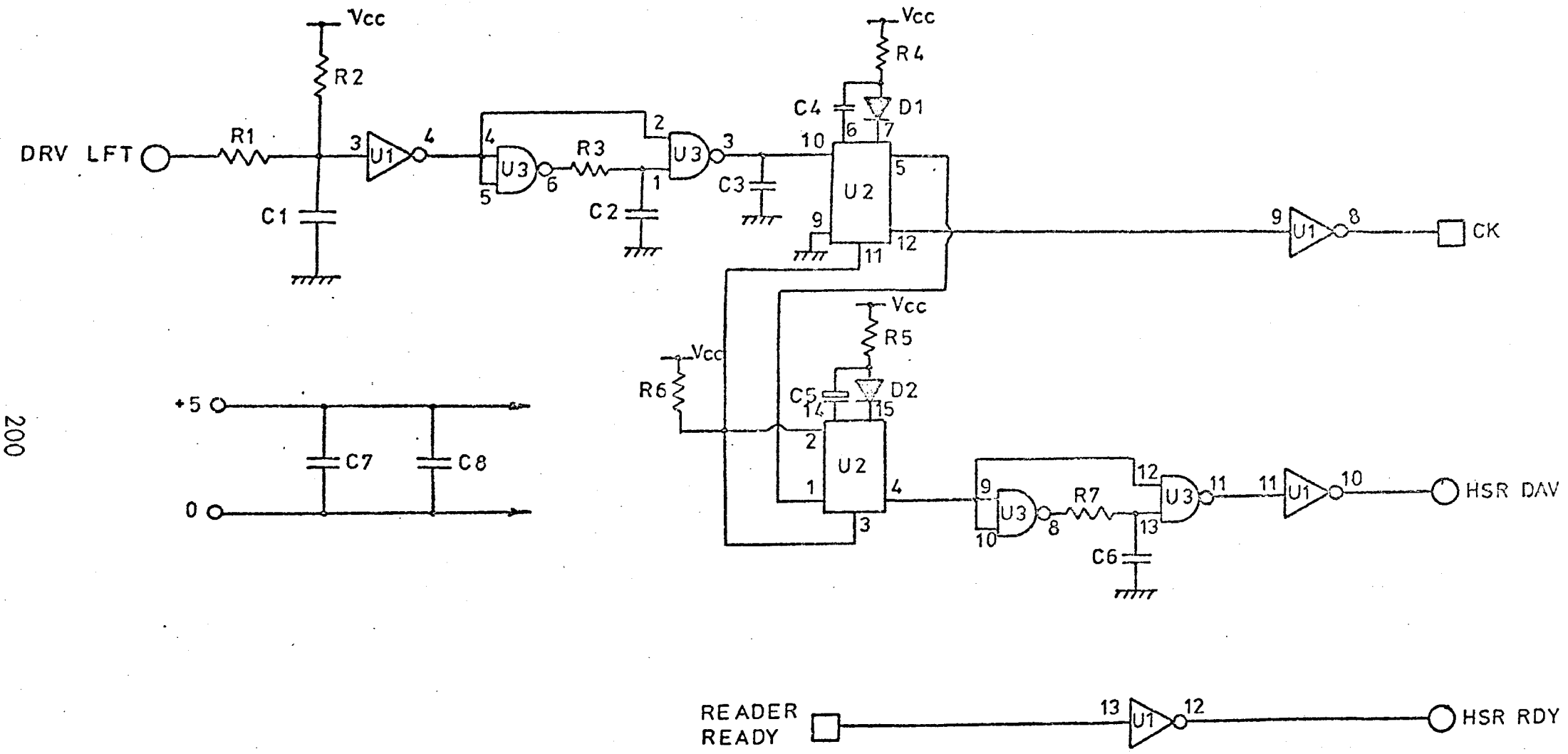


Figure 5.1 High-speed reader interface circuit.

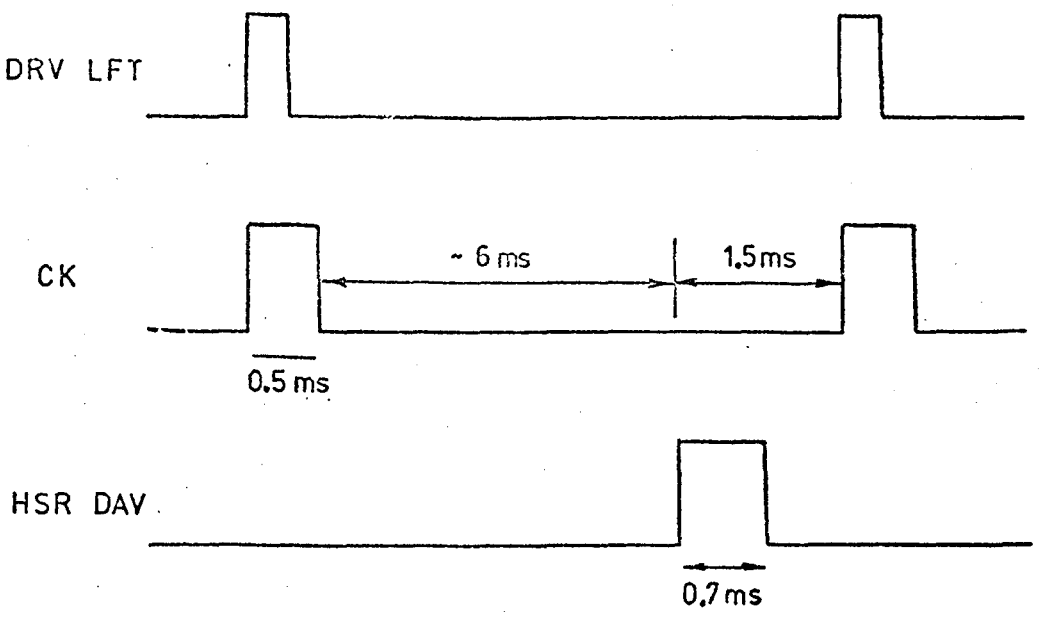


Figure 5.2 Timing waveforms.

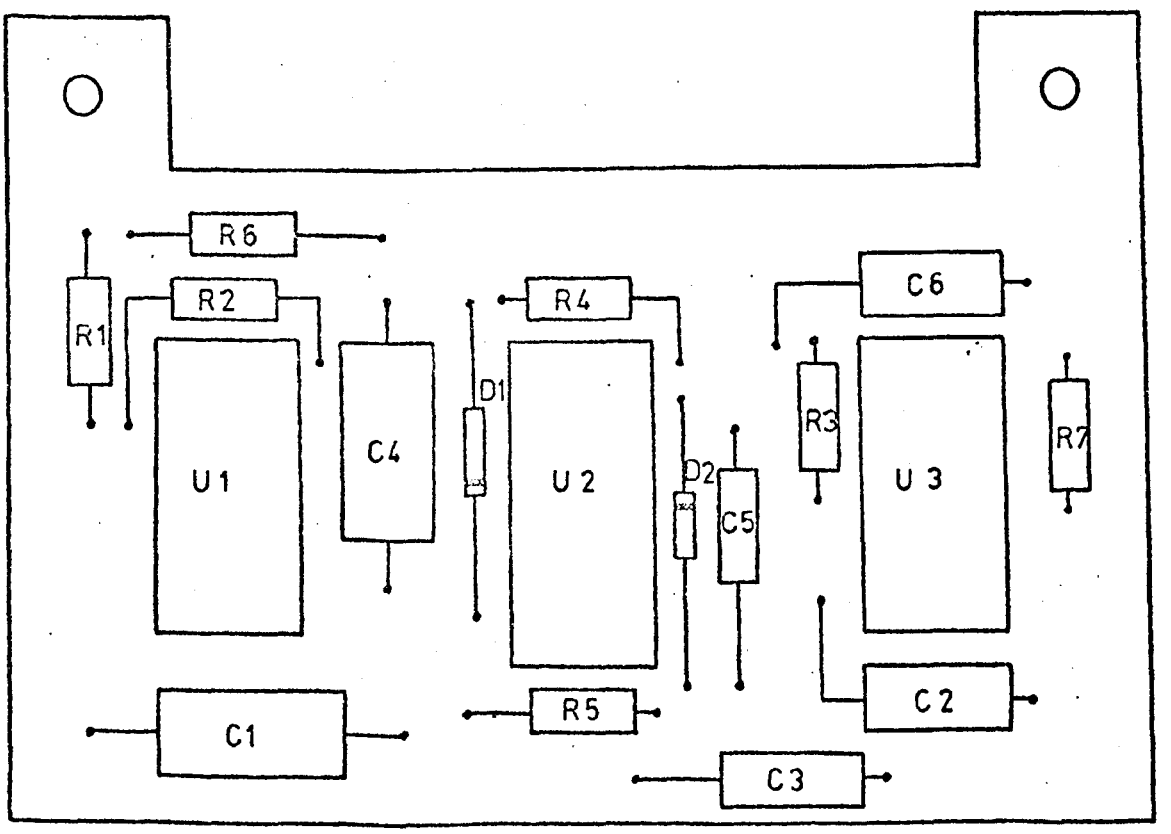


Figure 5.3 Component layout.

Capacitors

C1,C4	0.1μF polyester
C2,C6	2.2μF elect. 63V
C3	0.1μF polystyrene
C5	1.25μF elect. 25V
C7,C8	0.1μF ceramic

Semiconductors

D1,D2	1N4148	Si diodes

Integrated circuits

U1	SN7404
U2	SN74123
U3	SN7400

5.3.2 Connection list

Addmaster	Gimini	Meaning
1	B25	HSR 0
2	B27	HSR 1
3	B28	HSR 2
4	B29	HSR 3
5	B30	HSR 4
6	B31	HSR 5
7	B33	HSR 6
8	B34	HSR 7
9	GND	GND
10	Link to L	LED Power
A	-	SPKT
B	(via interface board to B23)	Reader Ready
C	-	Remote Start
D	-	Remote Off
E	-	Ck Enable
F	-	Fwd/Rev
H	(via interface board to B24)	Clock
J	+5V	+5v
K	GND	Gnd
L	+12V	LED Power

5.3.3 Signal definitions

DRV LFT - Drive left

Gimini output

Meaning - advance tape one character.

CK - Clock

Reader input

Meaning - advance tape one character.

HSR DAV - High-speed reader data available

Gimini input

Meaning - last character read from tape is stable on the reader data highway.

HSR RDY - High-speed reader ready

Gimini input

Meaning - reader ready for operation. This signal is the inverse of the READER READY output from the high-speed reader.

READER READY

Reader output

Meaning - reader is ready for next operation.

Chapter 6

Analogue-to-Digital and Digital-to-Analogue Converter Channels

6.1 Introduction

The circuits of an analogue input-output channel for use with a Gimini microcomputer [1] are described. Both circuits make use of the Ferranti ZN425E analogue-to-digital (A-D) and digital-to-analogue (D-A) converter as the basis of an eight-bit A-D and D-A channel. For the A-D converter, inputs in the range of 0 - 2.55V in 10mV steps can be accommodated with a conversion accuracy of ±5mV, and a conversion time of 22μs (approx.). The D-A converter is provided with a polarity switch, via a ninth bit (bit 15), and can hence give an output of -2.55V to +2.55V in 10mV steps to an accuracy of ±5mV with a minimum conversion time of 4μs (approx.).

Although, in this case, the circuits are for use with a Gimini microcomputer, they form the basis of a general purpose eight-bit analogue channel, which may be interfaced with any microcomputer.

6.2 Circuit Description

6.2.1 Function select

The mode of operation of the A-D/D-A channel is determined by a select input and the settings of the three low-order address lines. The select input is controlled by the thirteen high-order address lines, which must be set to OCT16773 in order to select the analogue channels. The functions specified by the three low-order addresses are:-

OCT16773	- 0	Reset
	- 2	A-D data strobe
	- 4	D-A data strobe.

The complete select circuit is given in Figure 6.1.

6.2.2 Reset

The analogue circuits may be cleared via two inputs,

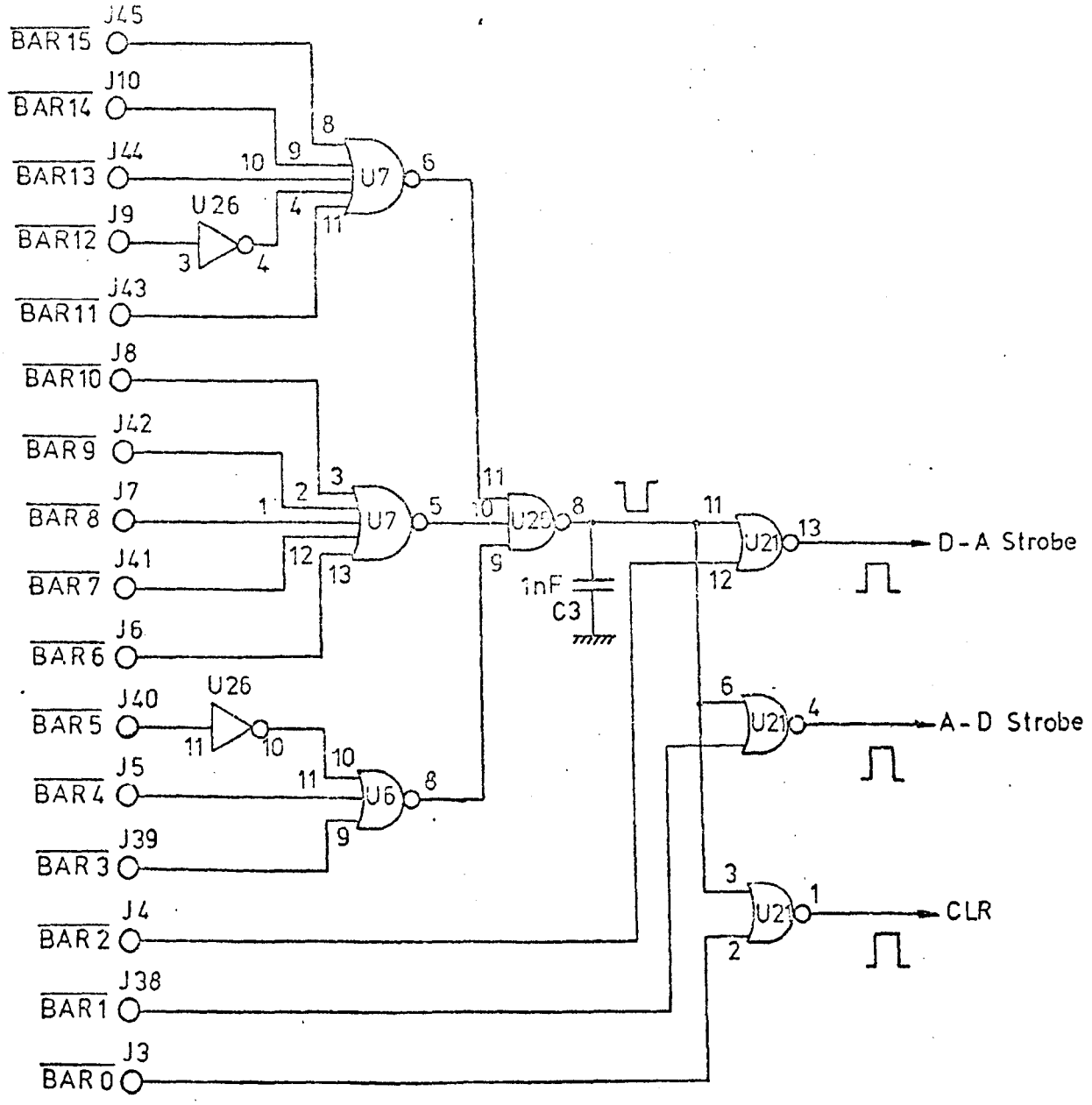


Figure 6.1 Function select control circuit.

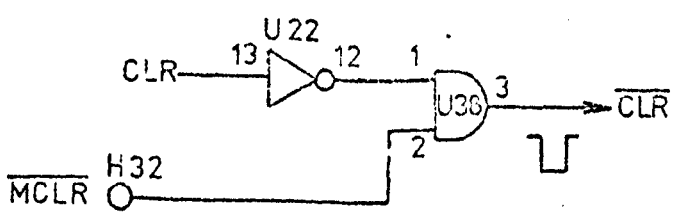


Figure 6.2 Clear circuit.

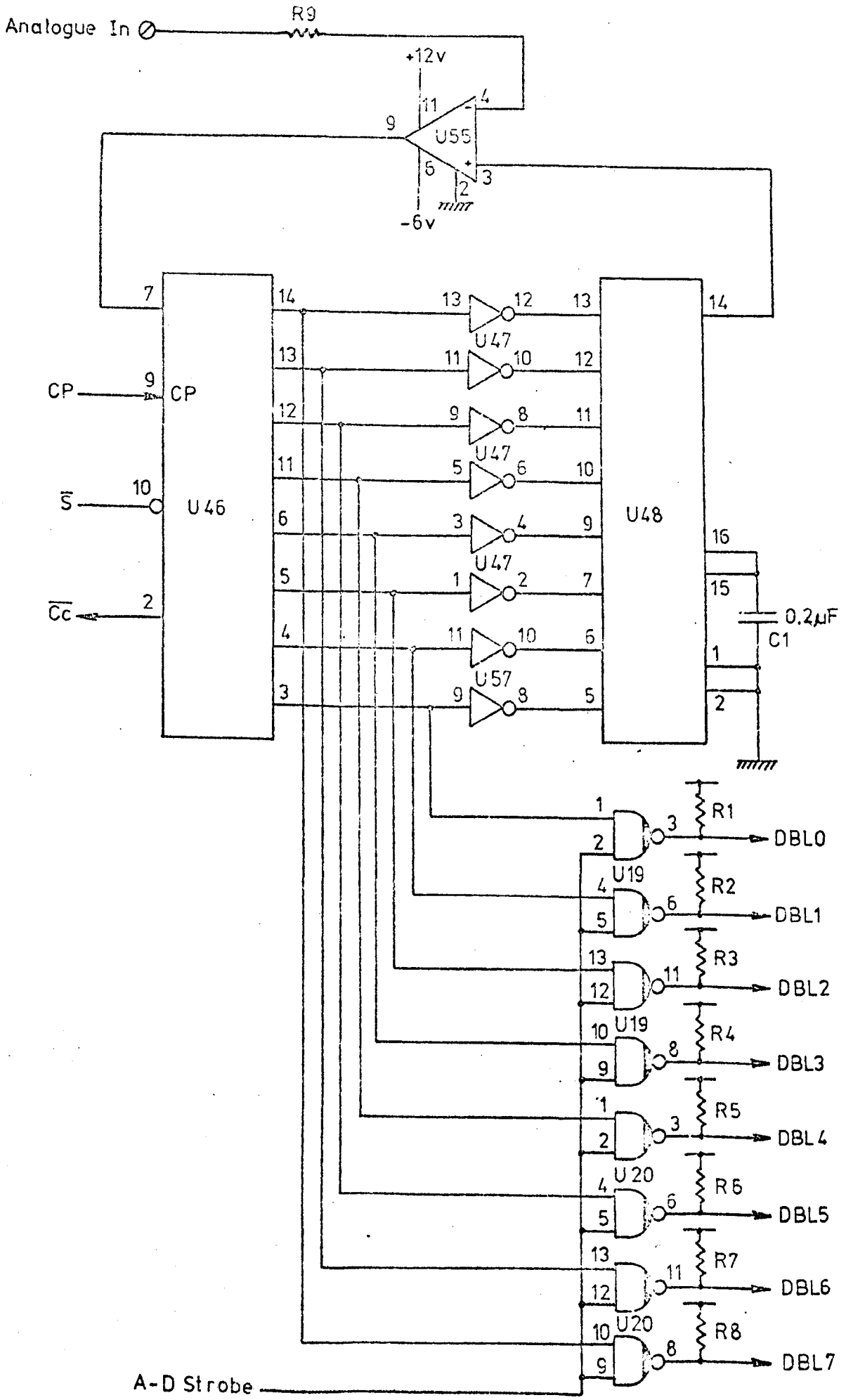


Figure 6.3 Successive approximation type A-D converter.

one under software control which involves the access of location OCT167730 and the other from the master clear (MCLR) switch on the front-panel of the Gimini microcomputer (Fig. 6.2).

6.2.3 Analogue-to-digital converter

The ZN425E circuit is capable of performing the A-D conversion alone but since it depends on a counter-type conversion technique, the operating speed is low (0.5ms min.). A considerable improvement can be achieved by the use of a successive approximation register in a successive approximation type A-D converter (Fig. 6.3). A minimum conversion time of 16μs is possible with this circuit.

The digital data are placed on the Gimini data bus by use of the A-D strobe control line. The strobe control passes the data to a set of bi-directional highway drivers, and thence to the Gimini highway. The bi-directional drivers are also used by the cassette interface[2] and so details are not reproduced here. To enable the bi-directional drivers, the A-D strobe control signal and data-to-bus ($\overline{DTB}$) signal from the Gimini are used (Fig. 6.4).

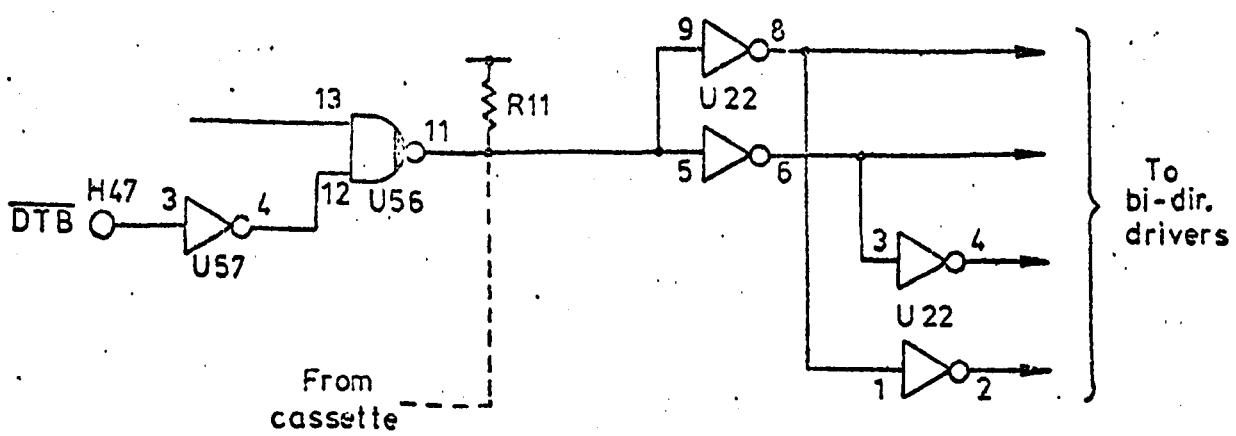


Figure 6.4 Bi-directional driver control.

6.2.4 Analogue-to-digital converter control

As well as the highway driver control, a clock input (CP) and start conversion ($\overline{S}$) signal are required by the successive approximation register (SAR). A conversion complete ($\overline{Cc}$) is also provided to indicate that the SAR has completed one conversion cycle. These signals are related as shown in Figure 6.5.

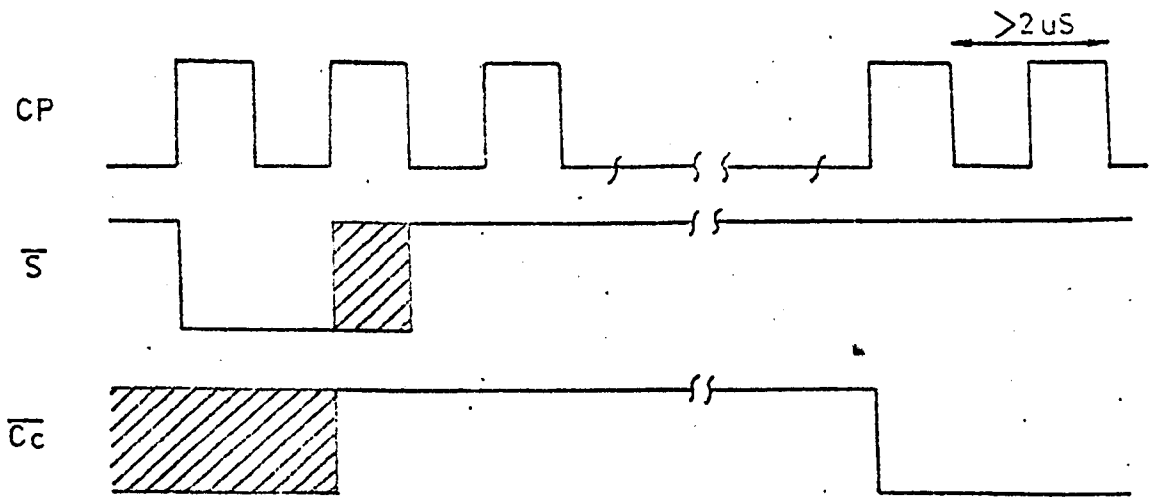


Figure 6.5 Timing diagrams for clock, start and conversion complete signals.

The clock input is generated quite conveniently from the clock-four ($\overline{CK4}$) signal from the Gimini system. The basic period of the $\overline{CK4}$ signal is 700ns (approx.) and a divide-by-four circuit is included to ensure that the CP signal is greater than 2μs in length (Fig. 6.6).

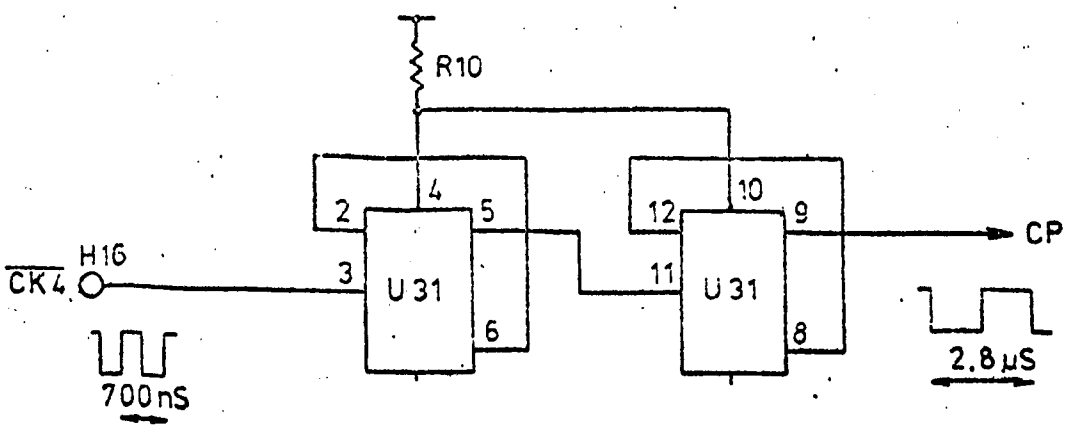


Figure 6.6 Clock generation circuit.

The A-D strobe signal is used to set the start conversion ($\overline{S}$) input which is then cleared by the rising edge of the conversion complete ($\overline{Cc}$) signal (Fig. 6.7).

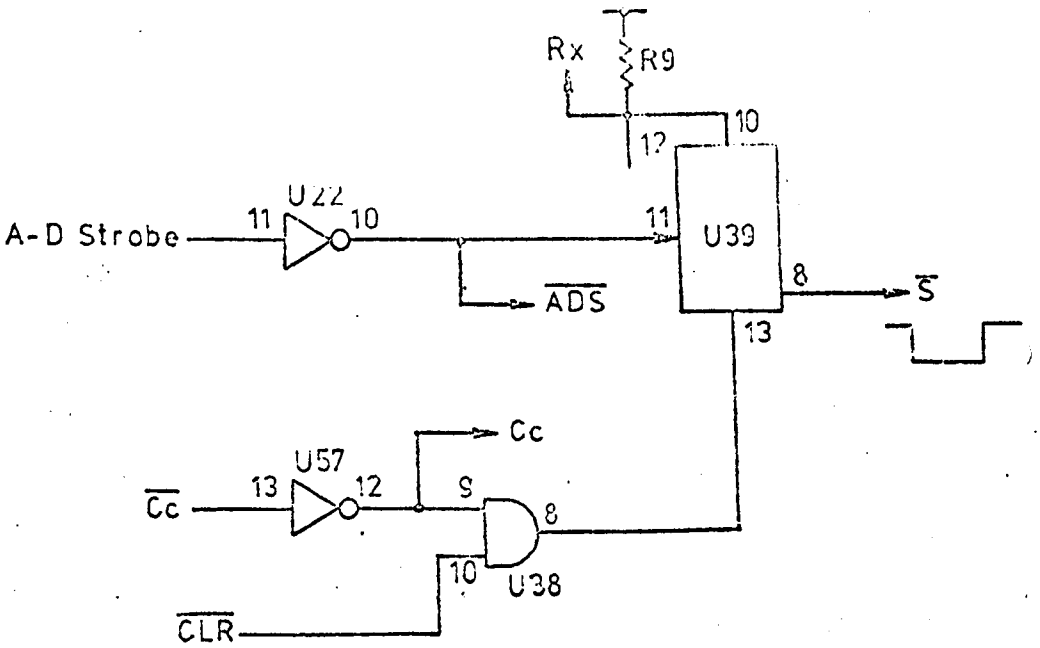


Figure 6.7 Start conversion signal.

An input is also provided so that the user may select the sampling frequency of the A-D converter via an external oscillator (e.g. from a real-time clock). The circuit is basically a two-stage synchronous counter which is clocked by the sampling input and cleared every time a sample is requested from the A-D converter (Fig. 6.8). The output from the first stage is input to the Gimini via the branch external 0 ($\overline{ECB0}$) input. To ensure that the data available ($\overline{ADC\ DAV}$) signal is not issued during a conversion, as may arise under the conditions shown in Figure 6.9, the conversion complete signal is used to hold $\overline{ADC\ DAV}$ high during this period. The data available signal remains valid until the next read data from the A-D converter command clears the counter. If, however, the next sampling clock pulse is received before the A-D converter data has been read, the second stage of the counter is set with the result that an LED is turned on, which indicates that the sampling frequency is too high for the program loop, currently under execution. The circuit given in Figure 6.8 may be modified so that the second

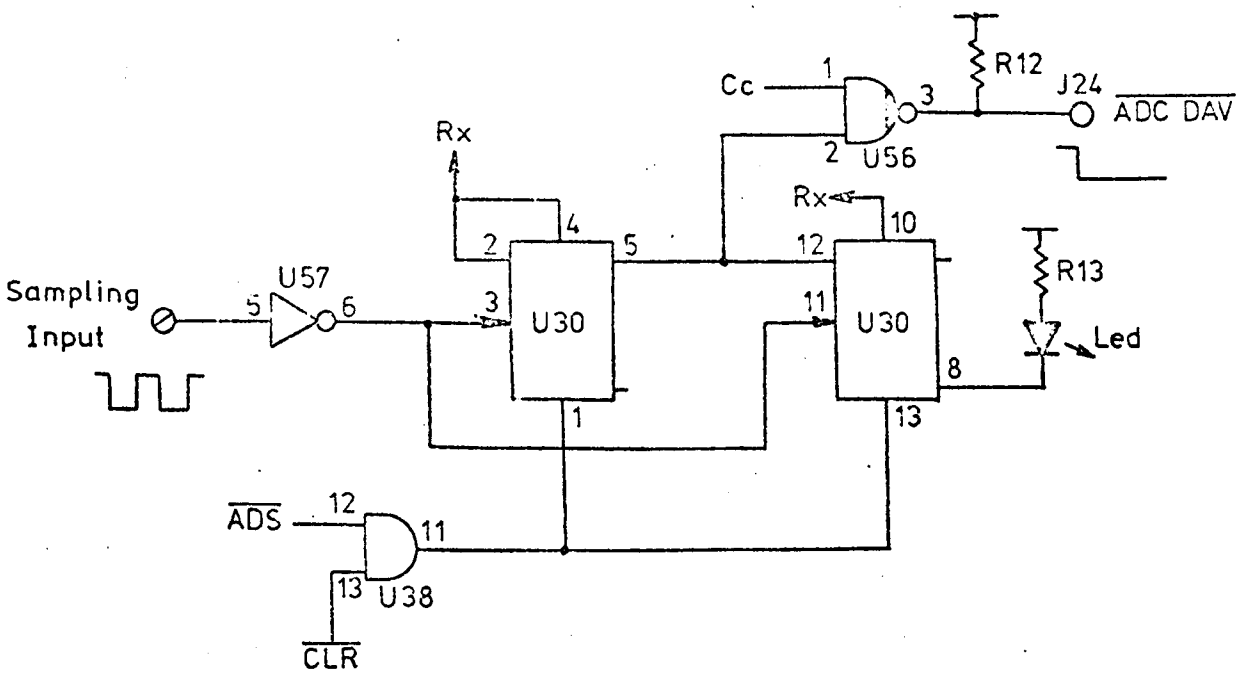


Figure 6.8 External sampling input - data available circuit.

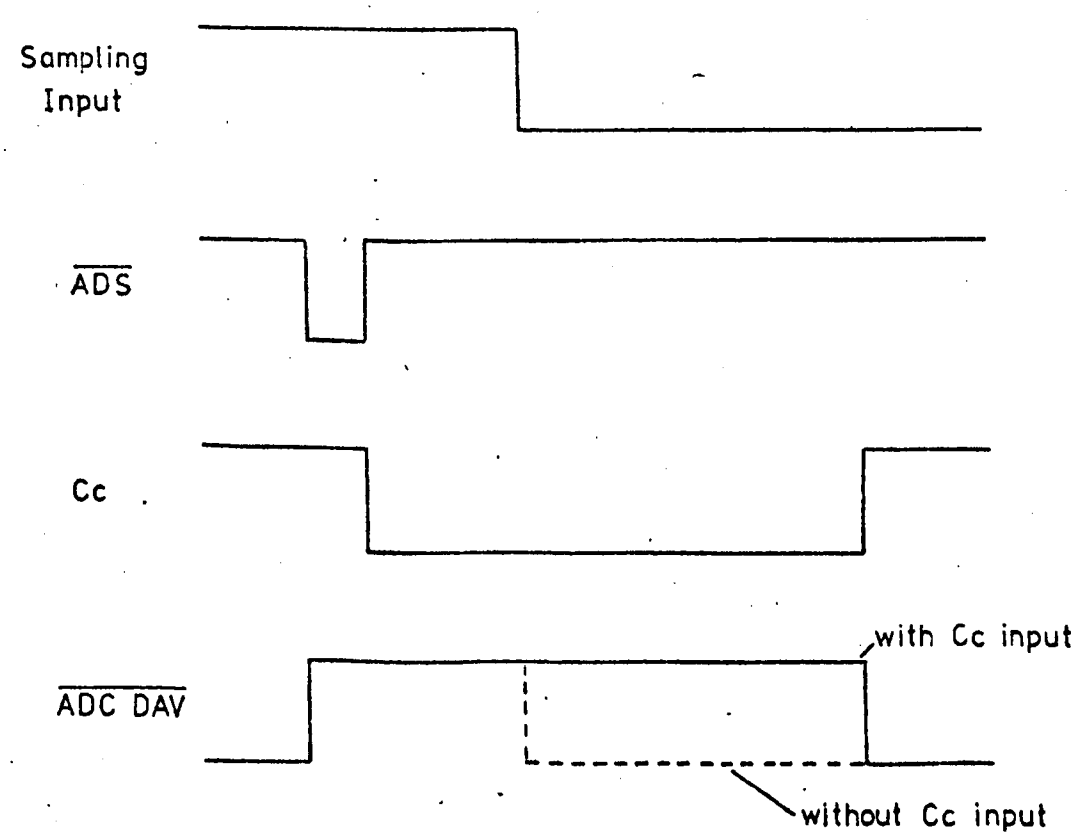


Figure 6.9 Use of conversion complete signal to ensure correct data available output.

stage of the counter is only cleared by the $\overline{CLR}$ signal. This will mean that, once set, the LED will remain on until a reset command is issued. Further, the output may also be used to generate an interrupt to the CPU should this be desired.

In use, the data read from the A-D converter are placed in the low-byte position of the Gimini data word; bits 8 - 15 are all set high. A typical program to read from the A-D converter is given in Appendix 6.A, which shows how the data available signal may be used.

6.2.5 Digital-to-analogue converter

The basic D-A converter is much simpler than the A-D converter and comprises an eight-bit latch and a second ZN425E converter package (Fig. 6.10). The digital data to be converted to analogue form for output are placed in the low-byte position and written to location OCT 167734. This causes a D-A strobe command to be issued which is gated with the data write strobe signal ($\overline{DWS}$) from the Gimini, to clock the data into the D-A data latch. The analogue signal described by the digital data word is then presented at the output of the D-A converter.

To provide a bi-polar output, the circuit shown in Figure 6.11 is used. The analogue output is fed to the input of a unity gain, non-inverting amplifier which acts as a buffer, and then to a second unity gain amplifier which may be set to either an inverting or non-inverting mode by means of transistor Q1. When a logic 0 is applied to point P, Q1 is turned off and the amplifier behaves as a unity gain, non-inverting amplifier. When a logic 1 level is applied to point P, Q1 is turned on, thereby shorting the positive amplifier input to ground. The circuit then acts as a unity gain, inverting amplifier. The polarity control bit (P) is the MSB of the Gimini data word (DBL15).

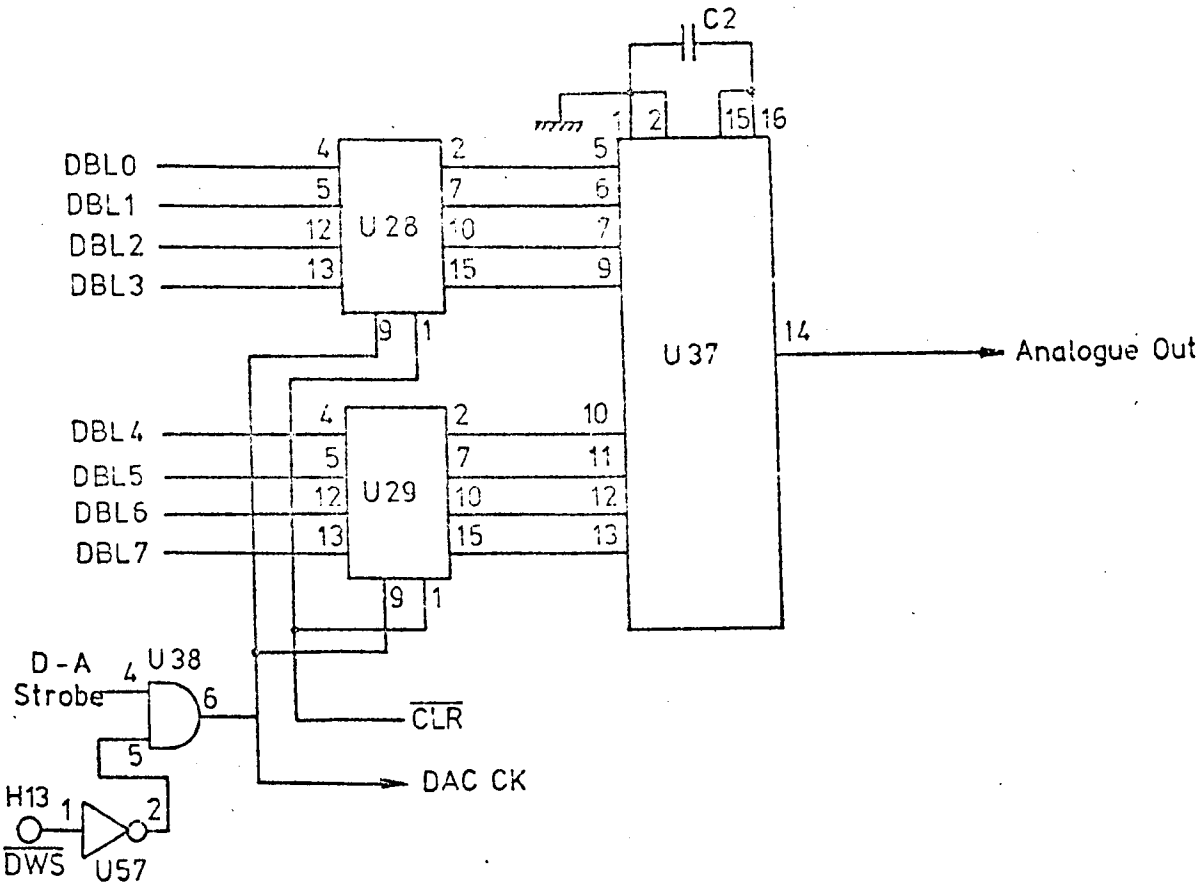


Figure 6.10 Digital-to-analogue converter circuit.

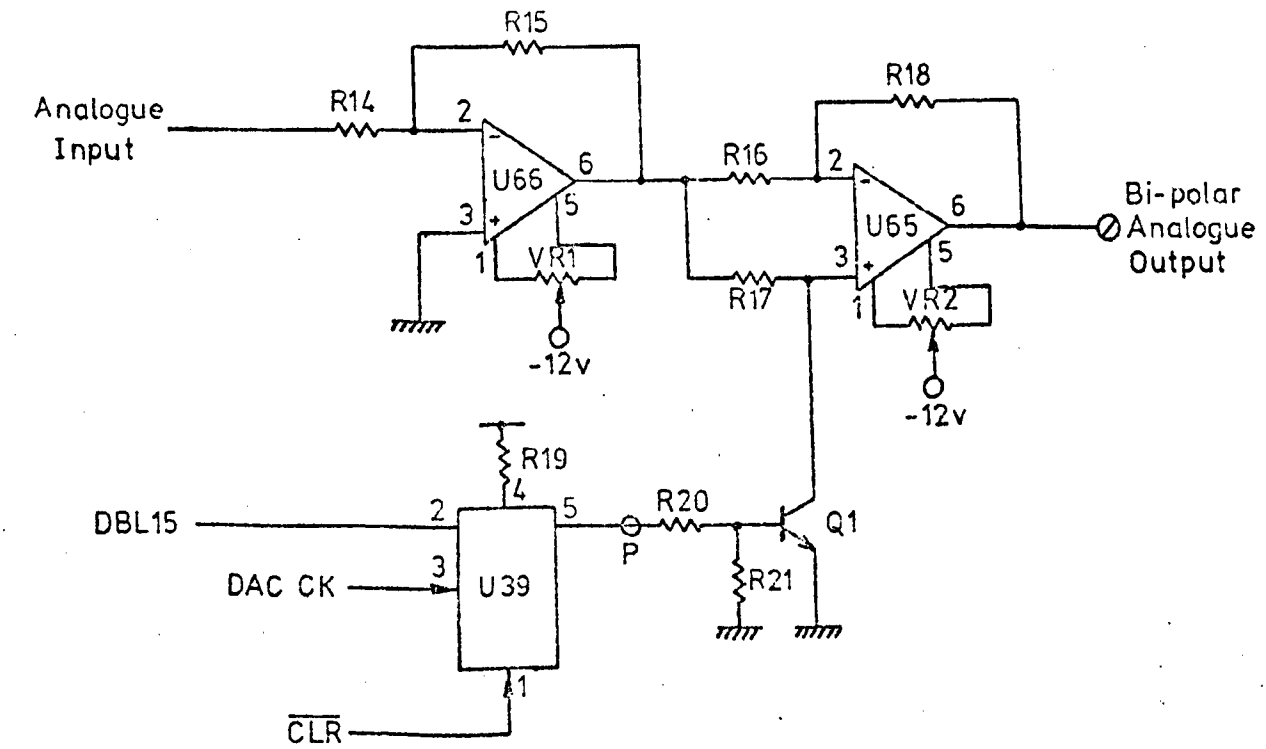


Figure 6.11 Polarity switching circuit.

The zero offset voltage is set by potentiometers VR1 and VR2, which are adjusted to give a zero output voltage for a digital zero input to the D-A converter.

6.2.6 Control bit

An additional output line is provided (DBL8) which may be used to provide a control or marker output. The ninth bit of the Gimini data word is strobed into a latch at the same time as the eight-bits plus sign for the D-A converter are strobed into the D-A latch. The output is buffered by a single transistor (Fig. 6.12).

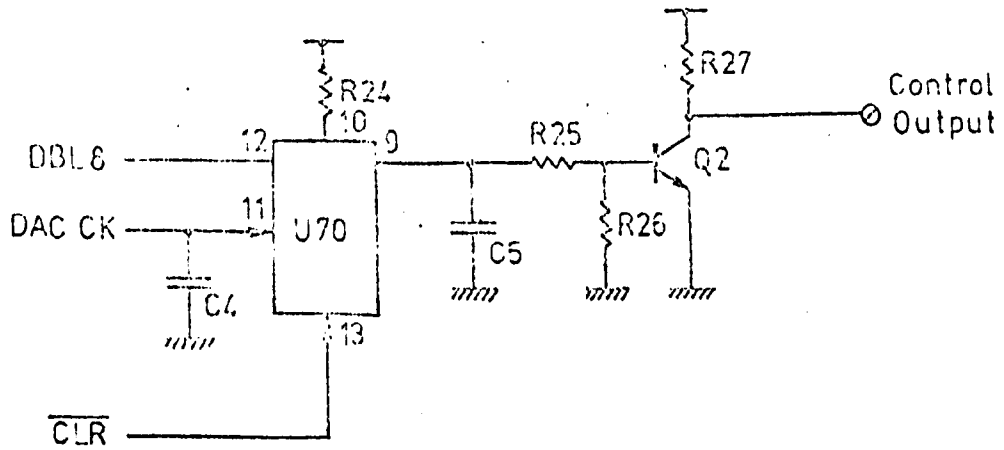


Figure 6.12 Control-bit output.

6.2.7 Power supply

The A-D and D-A converter circuits require +12V, +5V and -12V supplies, all of which are available from the Gimini system. A supply of -6V is required by the comparator circuit and this is derived from the -12V supply (Fig. 6.13).

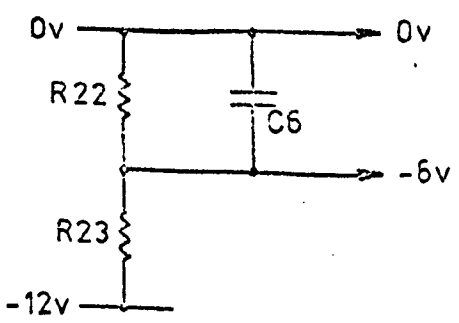


Figure 6.13 Generation of -6V supply.

6.3 Miscellaneous Details

The A-D and D-A converters are constructed on a 235mm x248mm 'breadboard' designed specifically for use with the Gimini system for random logic layouts. The component layout is as shown in Figure 6.14. The cassette interface [2] is constructed on the same circuit board and some circuits are common to both channels. The position occupied by the analogue channels on the board and the common circuits are shown in Figure 6.15.

6.3.1 Components list

Resistors	
R1 - R9	220Ω
R10,R12	1k
R11	470Ω
R13	680Ω
R14 - R19	10k
R20,R21	3.9k
R22,R23	330Ω
All ¼w ± 5%	
Capacitors	
C1,C2	2 x 0.1μF ceramic
C3 - C5	1nF ceramic
C6 - C9	0.1μF ceramic
Semiconductors	
Q1	2N2222
Q2	ZTX 300
LED	General purpose L.E.D.
Integrated circuits	
U6	SN7427
U7	SN74S260
U19,U20,U56	SN7403
U21	SN7402
U22,U26,U47,U57	SN7404

U25	SN7410
U28,U29	SN74175
U30,U31,U39,U70	SN7474
U37,U48	ZN425E
U38	SN7408
U46	Am2502
U55	A710
U65,U66	SN72741P

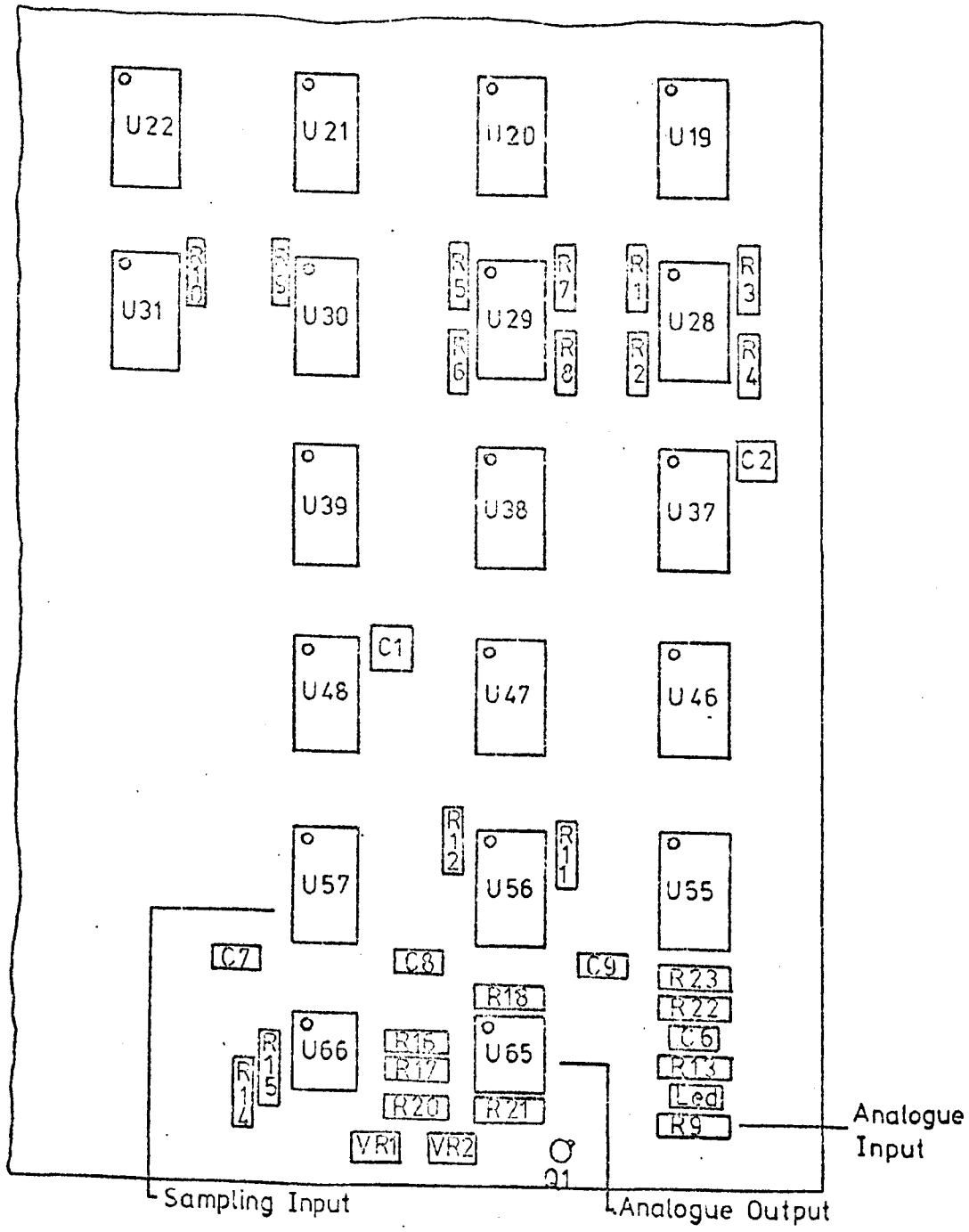


Figure 6.14 A-D and D-A converter component layout.

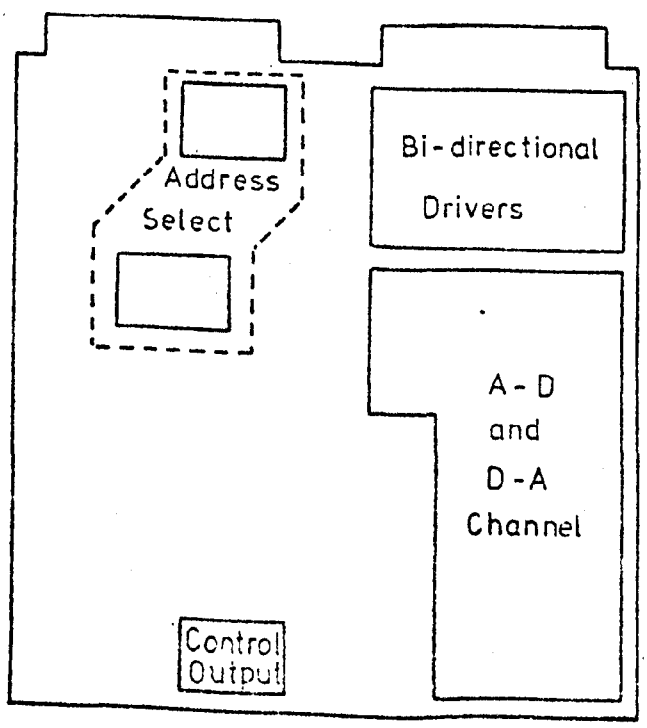


Figure 6.15 Location of A-D and D-A channels on the Gimini circuit board.

6.3.2 Gimini Edge Connector

J

+5V	36	1	+5V
	37	2	
$\overline{DBL1}$	38	3	$\overline{DBL0}$
$\overline{DBL3}$	39	4	$\overline{DBL2}$
$\overline{DBL5}$	40	5	$\overline{DBL4}$
$\overline{DBL7}$	41	6	$\overline{DBL6}$
$\overline{DBL9}$	42	7	$\overline{DBL8}$
$\overline{DBL11}$	43	8	$\overline{DBL10}$
$\overline{DBL13}$	44	9	$\overline{DBL12}$
$\overline{DBL15}$	45	10	$\overline{DBL14}$
	46	11	
HW2	47	12	HW1
HW4	48	13	HW3
	49	14	
	50	15	
BDRDY	51	16	
+12V	52	17	+12V
+12V	53	18	+12V
	54	19	
	55	20	
	56	21	$\overline{DISBAR}$
	57	22	
	58	23	
	59	24	$\overline{EBCO}$
	60	25	
	61	26	
	62	27	
	63	28	
	64	29	
	65	30	
	66	31	
	67	32	
	68	33	
	69	34	
GND	70	35	GND

H

+5V	36	1	+5V
	37	2	
$\overline{BADR1}$	38	3	$\overline{BADR0}$
$\overline{BADR3}$	39	4	$\overline{BADR2}$
$\overline{BADR5}$	40	5	$\overline{BADR4}$
$\overline{BADR7}$	41	6	$\overline{BADR6}$
$\overline{BADR9}$	42	7	$\overline{BADR8}$
$\overline{BADR11}$	43	8	$\overline{BADR10}$
$\overline{BADR13}$	44	9	$\overline{BADR12}$
$\overline{BADR15}$	45	10	$\overline{BADR14}$
$\overline{LWBT}$	46	11	$\overline{HGBT}$
$\overline{DTB}$	47	12	$\overline{DW}$
$\overline{ADAR}$	48	13	$\overline{DWS}$
$\overline{IAB}$	49	14	$\overline{INTAK}$
$\overline{NACT}$	50	15	$\overline{BAR}$
$\overline{TCI}$	51	16	$\overline{CK4}$
-12V	52	17	-12V
-12V	53	18	-12V
	54	19	
$\overline{INTRM}$	55	20	$\overline{INTR}$
$\overline{IPRI}$	56	21	$\overline{IPRO}$
IMSKI	57	22	IMSKO
$\overline{BUSRO}$	58	23	$\overline{BUSAK}$
$\overline{BAKI}$	59	24	$\overline{BAKO}$
BMSKI	60	25	BMSKO
	61	26	
	62	27	
	63	28	
	64	29	
	65	30	
	66	31	
$\overline{HALT}$	67	32	$\overline{MCLR}$
	68	33	$\overline{DISCC}$
	69	34	$\overline{DISDTB}$
GND	70	35	GND

6.4 Appendix 6.A

```
 REL £ADCIP
;;;;;
;
;   EXAMPLE PROGRAM TO INPUT 1024 SAMPLES FROM THE
;   ANALOGUE-TO-DIGITAL CONVERTER AND STORE IN A
;   BUFFER AREA.
;   BUFFER S.A. = 1024 (OCT 2000)
;
;;;;;
R0 = 0
R1 = 1
R2 = 2
R3 = 3
R4 = 4
R5 = 5
SP = 6
PC = 7
BUFF = 2000

 GLOB £ADCIP
      MVII BUFF,R5          ;LOAD BUFFER S.A.
      CLRR R0
      MVO R0,167730         ;RESET ADC
WAIT  BEXT INPT,0           ;SAMPLE READY?
      B WAIT
INPT  MVI 167732,R0         ;INPUT ADC SAMPLE
      ANDI 377,R0           ;CLEAR TOP BITS
      MVO@ R0,R5            ;SAVE RESULT AND INC. PNTR.
      CMPI R5,BUFF+2000     ;BUFFER FULL?
      BLE WAIT
      HLT                   ;STOP
      NOP
      END
```

6.5 References

1. Series 1600 Microprocessor System - Documentation, General Instrument Microelectronics Corp., Dec. 1976.
2. Cassette Unit Interface - Circuit Description, Internal Report, R. A. Comley, Dec. 1977.

Chapter 7

Cassette Storage Unit

1. Technical Description

2. User Guide

7.1 Technical Description

7.1.1 Introduction

The circuits required to interface a Racal P72 digital cassette deck to a Gimini microcomputer system are described. The P72 is a general purpose, incremental and continuous digital data recorder and reproducer and uses standard (3.81mm) 'Phillips' type cassettes. It is supplied with all of the control and formatting logic necessary to control the motor and transfer data to and from tape.

Control of the deck is accomplished via eleven TTL compatible control inputs, nine of which are used in this interface. Data are transferred as eight-bit parallel bytes with a ninth bit available in certain modes of operation (see User Guide [1]). A clock input is provided to initiate the transfer of data and a data available output indicates the availability of data during read operations.

In addition to the control and data lines, nine status outputs are provided, seven of which are made available to the user via the Gimini system.

Provision is made for the extension of the cassette system to a complete self-contained, general purpose, line-local peripheral. For this purpose, a comprehensive front panel has been provided which is, at present, used very little.

7.1.2 Circuit Description

7.1.2.1 Function select

The function to be performed by the interface circuits is controlled by the select input and three, low-order address lines. The select input is controlled by the thirteen high-order address lines, which must be set to OCT16774 in order to select the interface. The functions specified by the three lower address lines are:-

OCT16774 - 0 Reset

- 1 Read status
- 2 Write control word
- 3 Write data
- 4 Read request
- 5 Read data
- 6 Not used
- 7 Not used

A three-to-eight line decoder is used to perform the decoding operation, as shown in Figure 7.1.1.

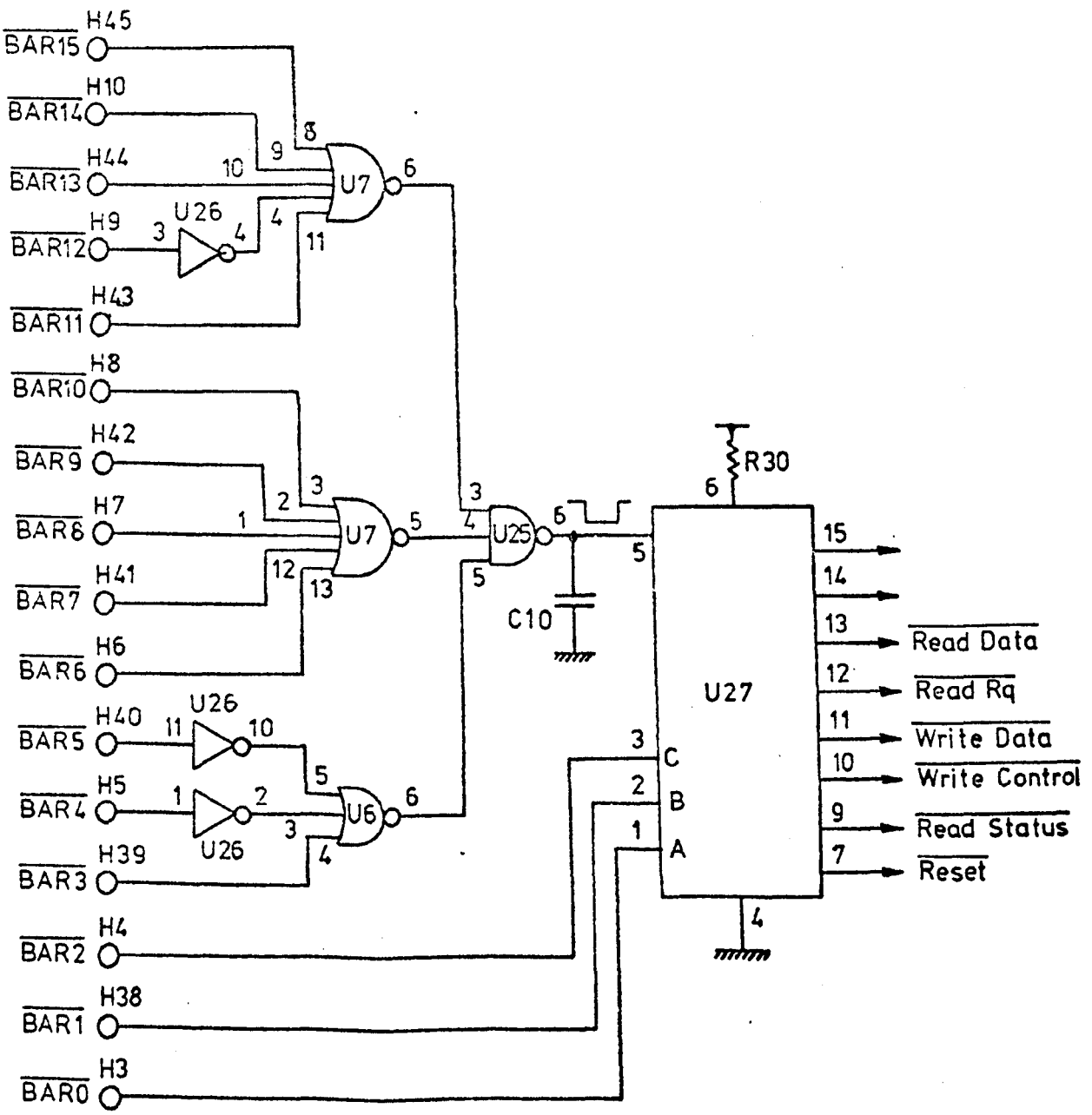


Figure 7.1.1 Function select control circuit.

7.1.2.2 Reset circuit

The cassette interface may be reset via three different inputs. Upon receipt of a reset command from any of these inputs, the control word is set to select a low-speed incremental read operation, the data in the data-out latches are cleared and a reset signal ($\overline{BRST}$) is sent to the cassette (Fig. 7.1.2).

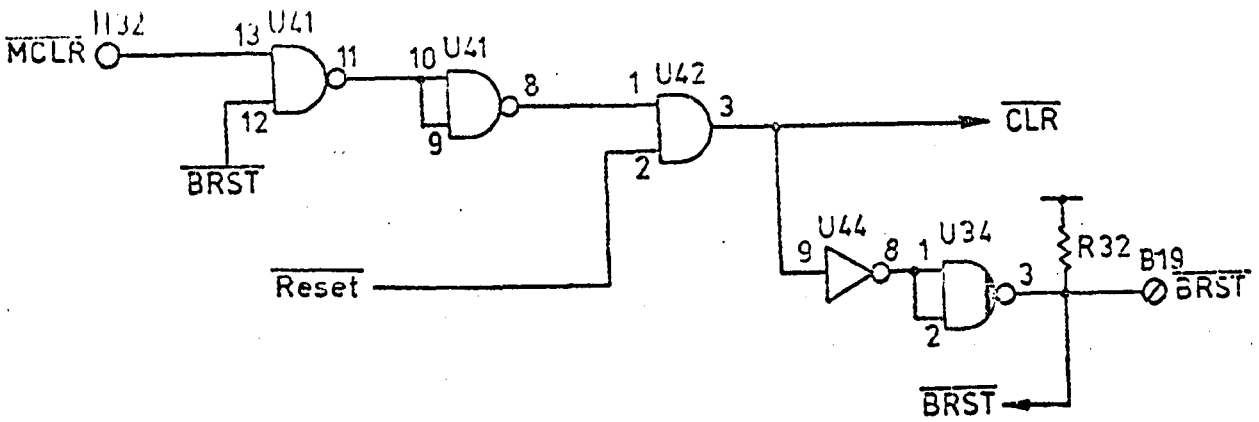


Figure 7.1.2 Reset circuit.

7.1.2.3 Read status

Six inputs are provided which serve to specify the status of the cassette deck. (For a definition of each of the status inputs, see User Guide [1]). The status word is gated onto the six low-order data bus lines by the data-to-bus signal ($\overline{DTB}$) from the Gimini when read status is selected (Fig. 7.1.3).

7.1.2.4 Control word

The operation of the cassette deck is controlled by eight control inputs. These must remain stable at the cassette interface and so must be latched from the Gimini data bus at the appropriate time. The control word is latched by the data write strobe ($\overline{DWS}$) from the Gimini when the write control word line is selected (Fig. 7.1.4). The lower eight-bits of the data bus should be set to the required control word. (For an explanation of each of the

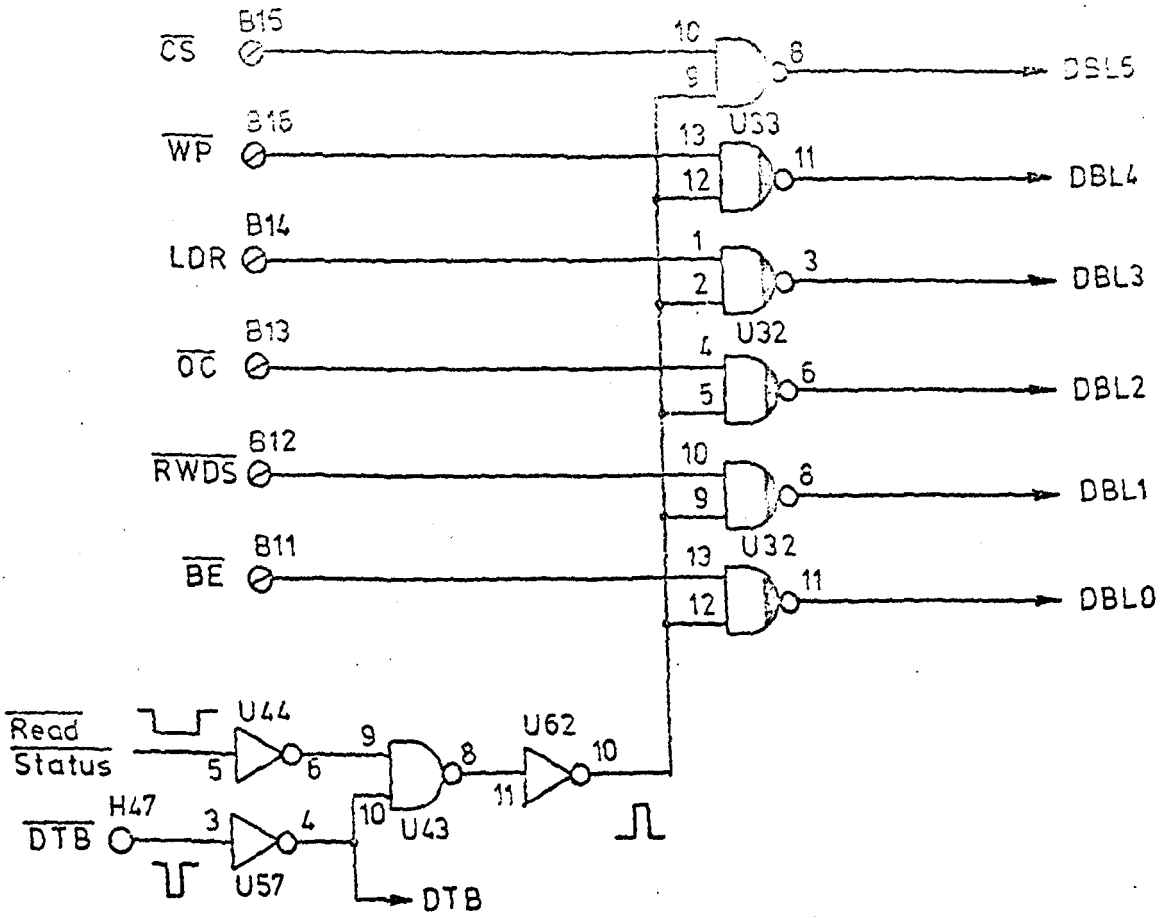


Figure 7.1.3 Status word input.

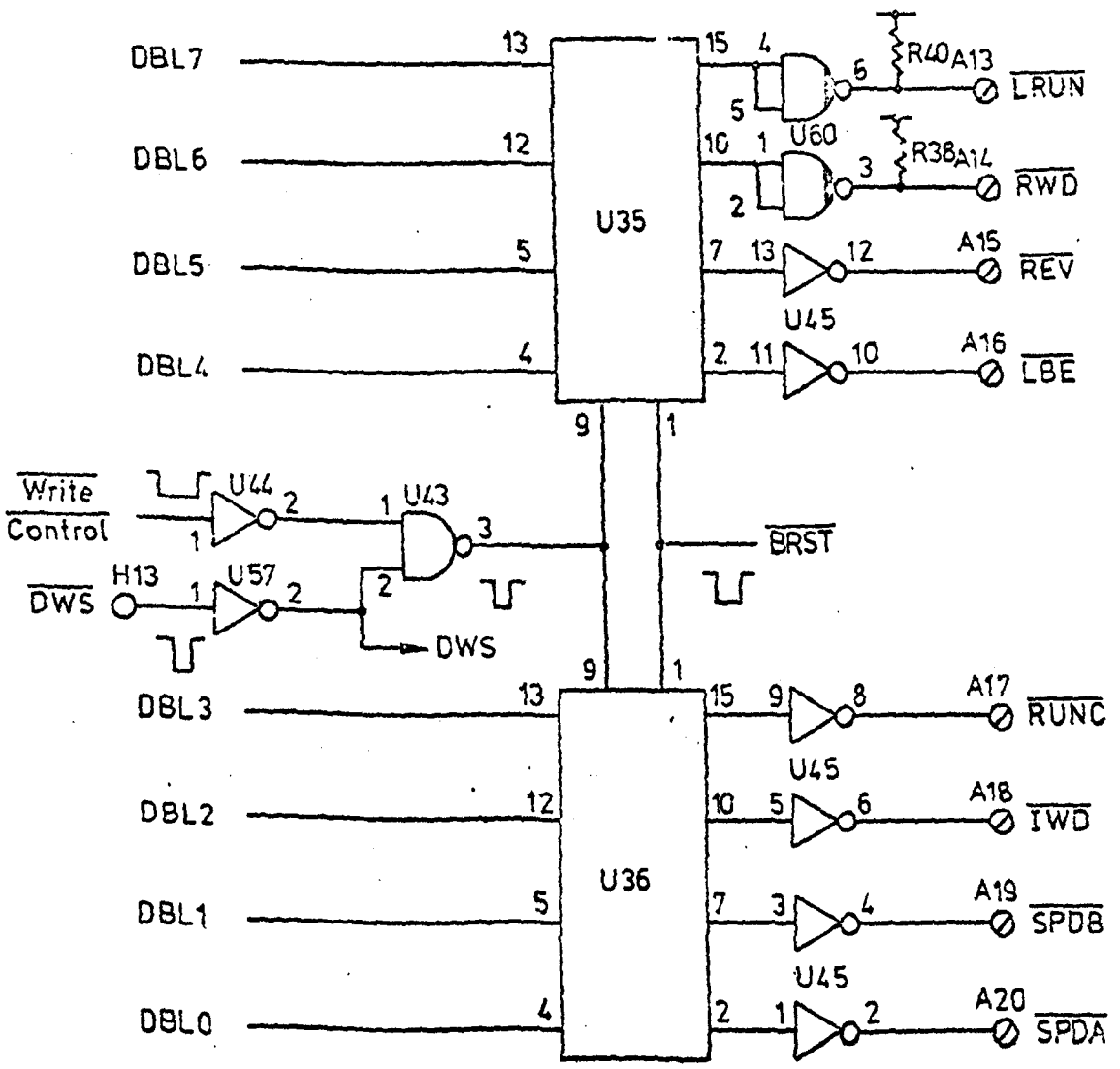


Figure 7.1.4 Control word latch.

control bits, see the User Guide [1]).

7.1.2.5 Read-write delay

The P72 cassette deck requires that any change in the read-write input ($\overline{IWD}$) be accompanied by a 40ms delay to allow the head current to become established (for a write) or decay (for a read). A monostable, set to give this delay and triggered by either edge of the $\overline{IWD}$ signal, is used to provide this function (Fig. 7.1.5). The output from the monostable is gated into the read-write handshake lines so as to inhibit operations during the 40ms delay period.

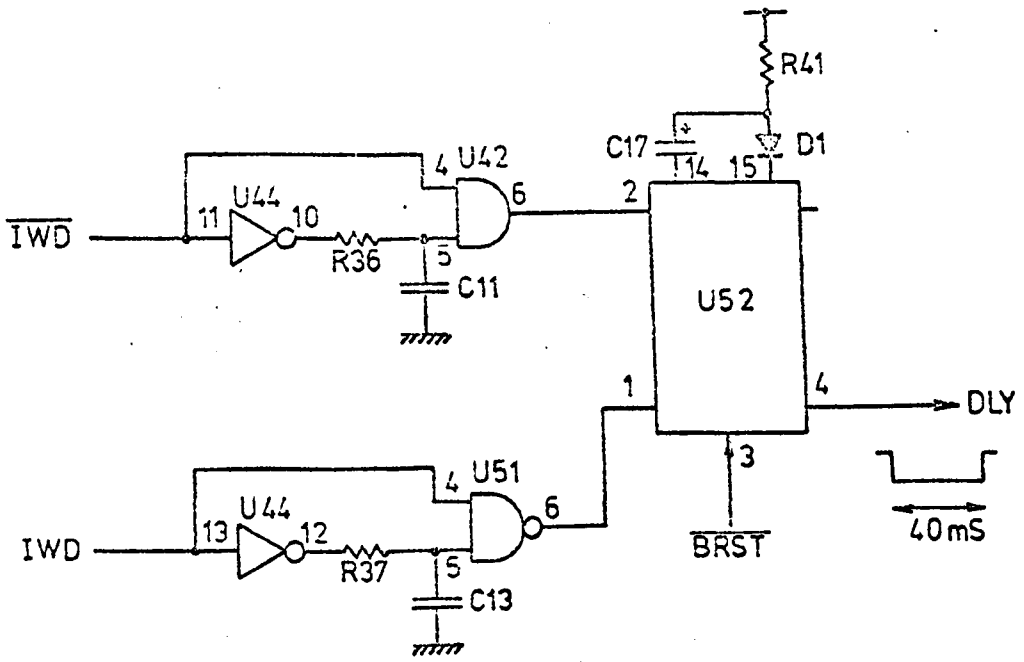


Figure 7.1.5 Read-write delay circuit.

7.1.2.6 Read-write handshake

All read and write operations are performed by the use of three handshake lines. One output, issue byte strobe ($\overline{IBS}$), is used to initiate either read or write operations. This is basically a clock output, the negative-going edge of which instigates the sequence of events necessary for a read or write operation. The delay input is used to prevent the issue of a byte strobe during the 40ms delay period.

During read operations, the data available input (IDAV) indicates that the requested byte is stable at the cassette

output. For write operations, the ready output ($\overline{IRDY}$) indicates the readiness of the deck to receive the next data byte. Timing diagrams for both sequences are given in Figure 7.1.6.

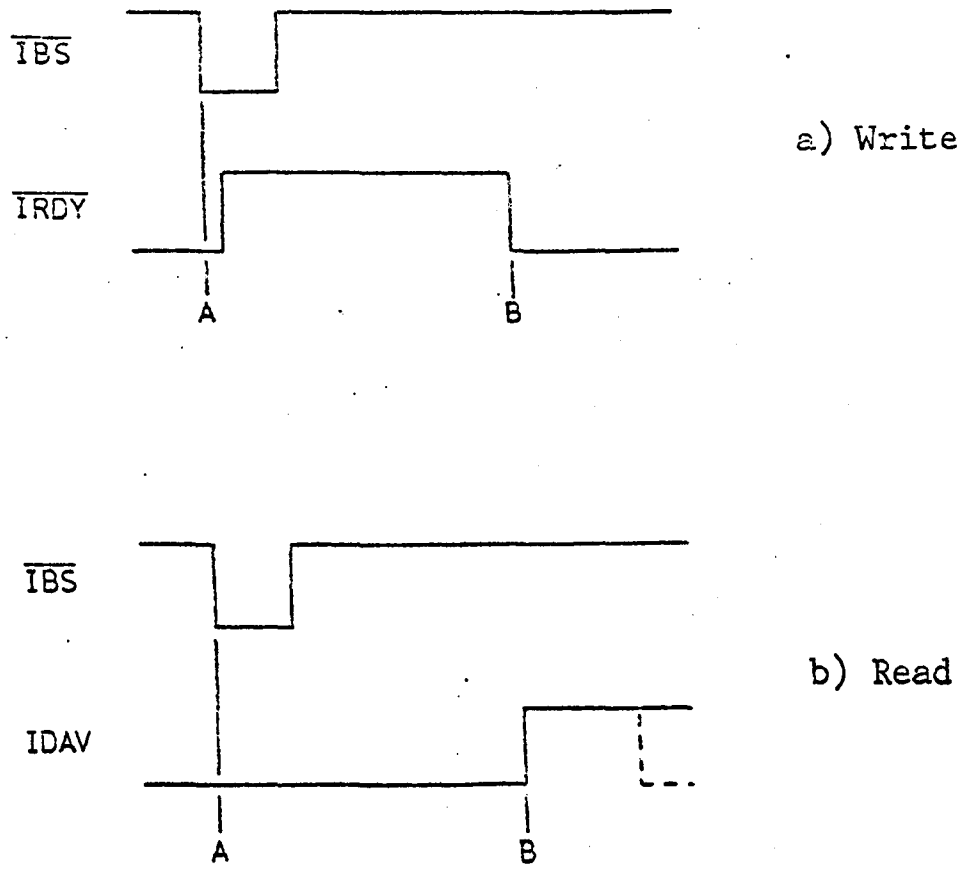


Figure 7.1.6 Read-write timing diagrams.

In some modes of operation, a 'drop through' delay is required to allow data to ripple through a first-in, first-out (FIFO) buffer (internal to the P72) before the ready ($\overline{IRDY}$) line is taken high. In order to avoid any ambiguities, a delay is generated which is of a greater duration than that required by the cassette deck, so that $\overline{IRDY}$ is set by the trailing edge of $\overline{IBS}$ and remains set until $\overline{IRDY}$ returns low (Fig. 7.1.7).

Both IDAV and $\overline{IRDY}$ are input to the Gimini system via the branch external inputs [2]. Inputs 1 and 2 are used for this purpose and provision is made to disable the inputs from the cassette, should they be required by another peripheral. At present a pull-up to V_{CC} via a 2.2k resistor is used, which permanently enables the cassette lines, but a

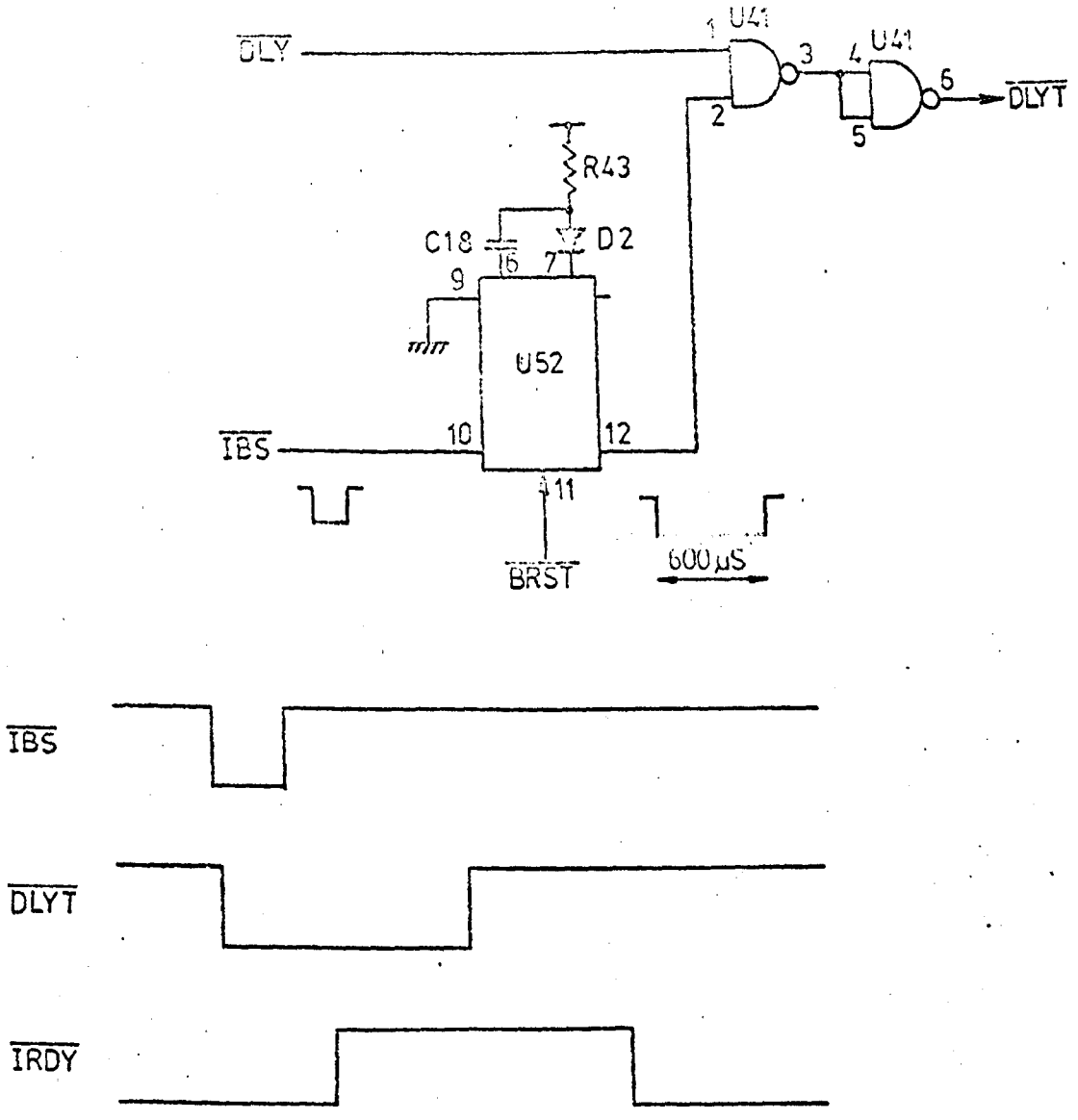


Figure 7.1.7 'Drop through delay' for low-speed incremental and continuous write operations.

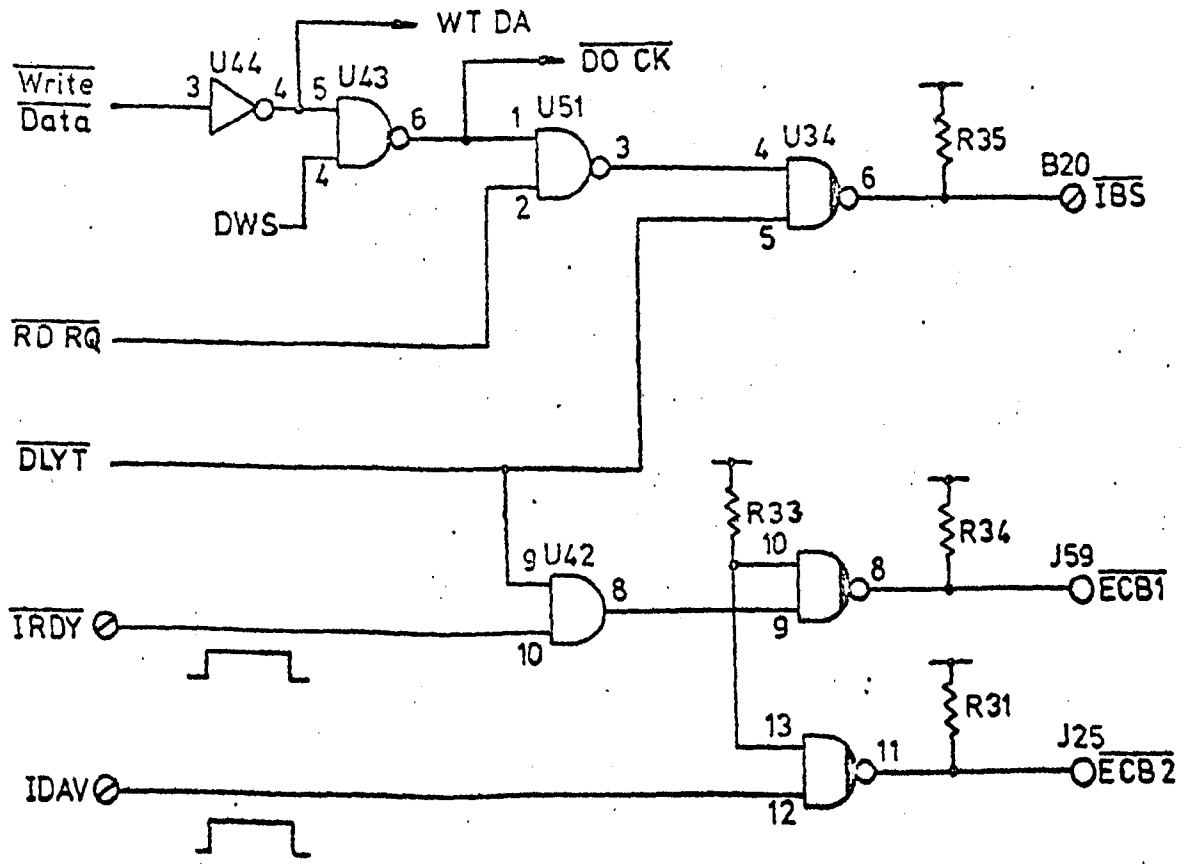


Figure 7.1.8 Handshake lines.

method of enable via a single gate could be provided, if required.

The delay output is gated into the ready line so that busy is indicated during the 40ms delay period. The complete handshake circuits are given in Figure 7.1.8.

7.1.2.7 Data out (write to cassette)

The data to be written must be stable at the cassette inputs before a byte strobe is issued and must remain stable until busy is indicated ($\overline{IRDY}$=1). It is therefore necessary to latch the data. The data write ($\overline{DW}$ from Gimini) can be

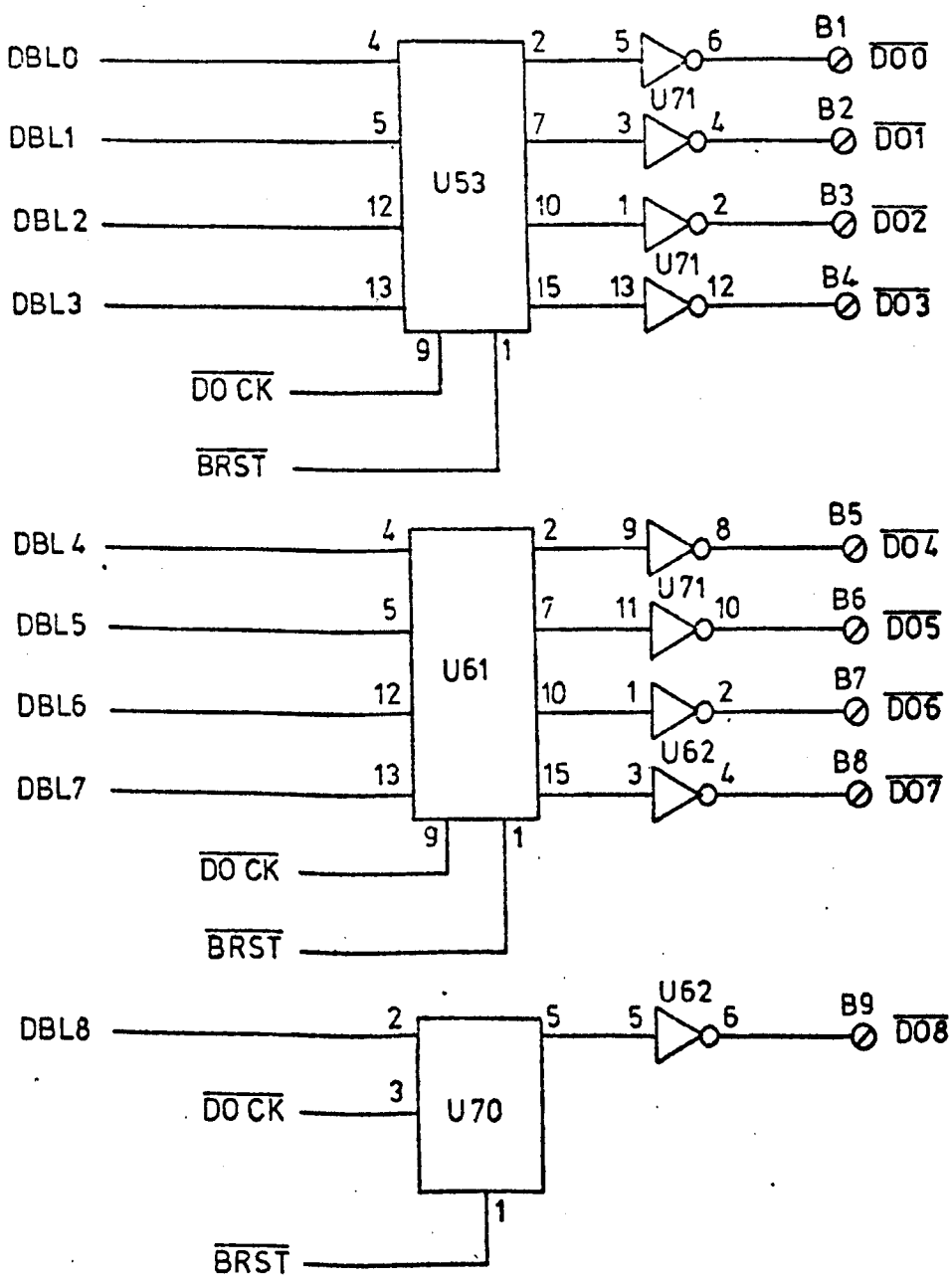


Figure 7.1.9 Data out latch.

used conveniently, to perform this operation since it occurs 50ns (approx.) before the data write strobe ($\overline{DWS}$) signal which is used to issue the byte strobe command. The latch circuit is given in Figure 7.1.9.

7.1.2.8 Data in (read from cassette)

The rising edge of IDAV indicates that the requested byte is available on the cassette data output lines. The data may be strobed onto the Gimini highway at any time after the occurrence of the IDAV edge. The data-to-bus ($\overline{DTB}$ from Gimini) is used to issue the strobe pulse when read data is selected. The data in circuit is as shown in Figure 7.1.10.

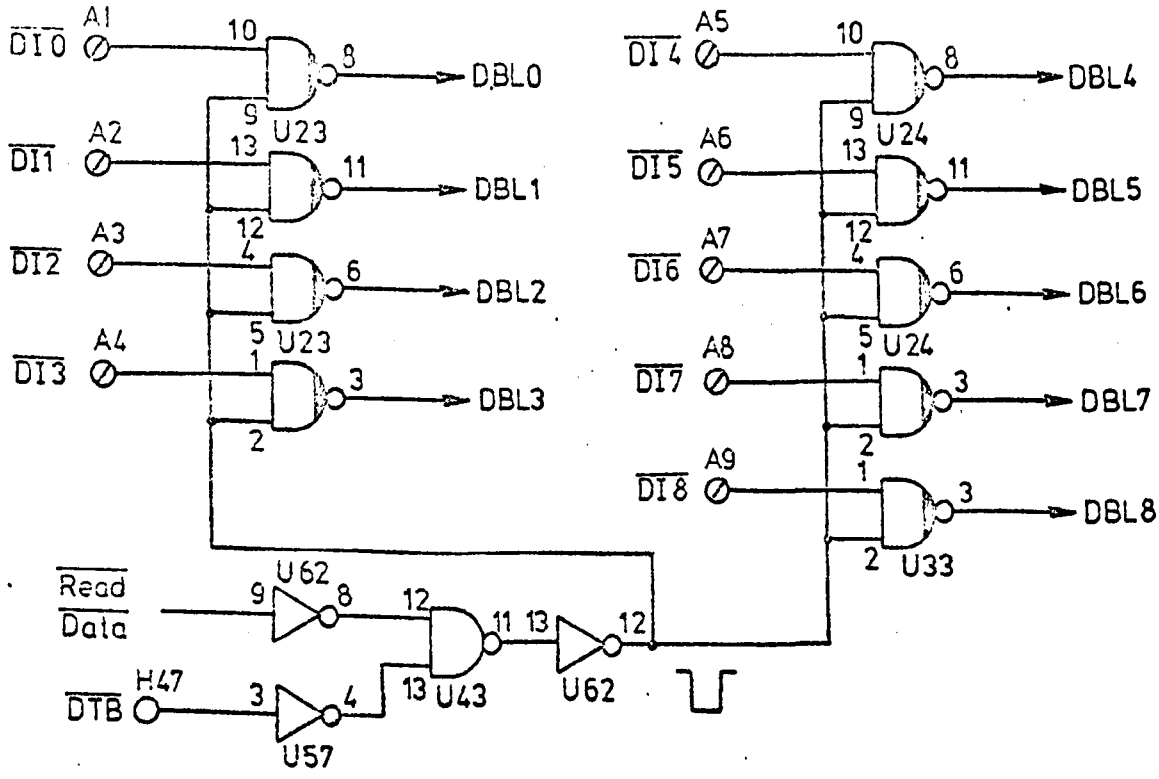


Figure 7.1.10 Data in circuit.

7.1.2.9 Bi-directional driver-receivers

A set of sixteen bi-directional driver-receivers is included to act as a buffer between the cassette interface circuits and the Gimini highway (Fig. 7.1.11). These are of a standard design and are included so that the circuit board containing the interface circuits presents only a single

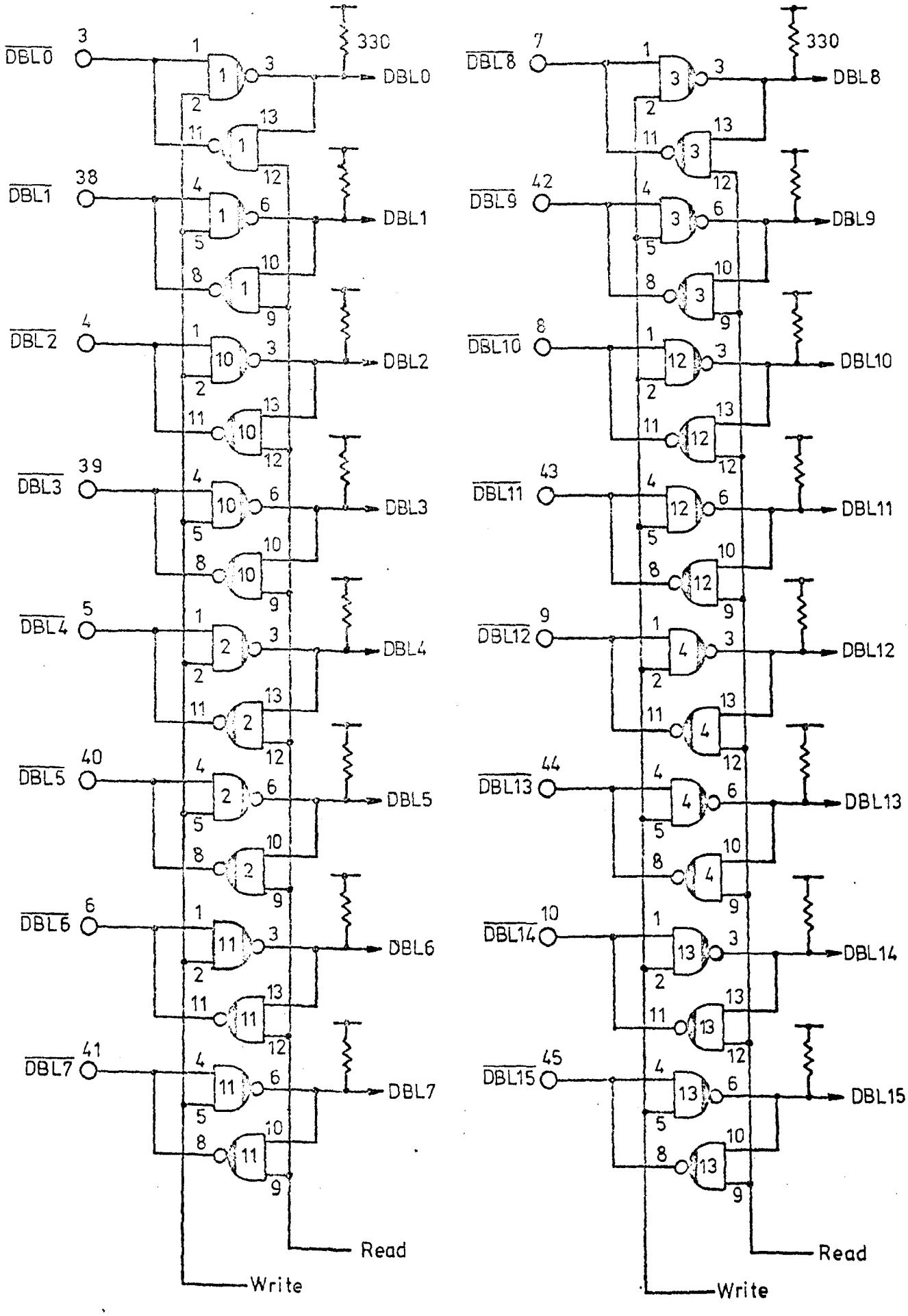


Figure 7.1.11 Bi-directional driver-receivers.

load to the Gimini bus on the receive side and to provide the necessary drive capability on the driver side. (Note, the analogue-to-digital and digital-to-analogue converter circuits (see Chapter II-6) also use the bi-directional driver-receivers).

7.1.2.10 Front-panel control and indicators

Three functions may be controlled from the front-panel, at present; reset, rewind and leader run, which is used to advance the cassette from clear leader. In addition, seven LEDs are connected to give a visual indication of the deck status (Fig. 7.1.12). The buffers, necessary to drive the indicators, are mounted on a small board housed in the cassette unit.

7.1.2.11 Power supply

The cassette unit requires three supply voltages, all of which are available in the Gimini system. All three supplies are switched by a single-pole, illuminated switch. The switch is used to power the coil of a four-pole change-over relay which connects the supplies simultaneously (Fig. 7.1.13).

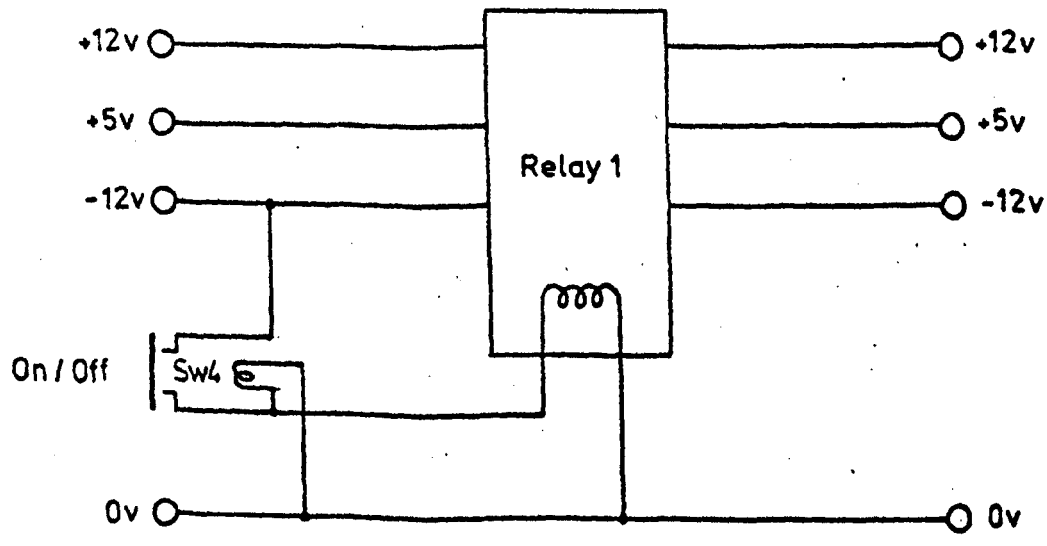


Figure 7.1.13 Power supply switch.

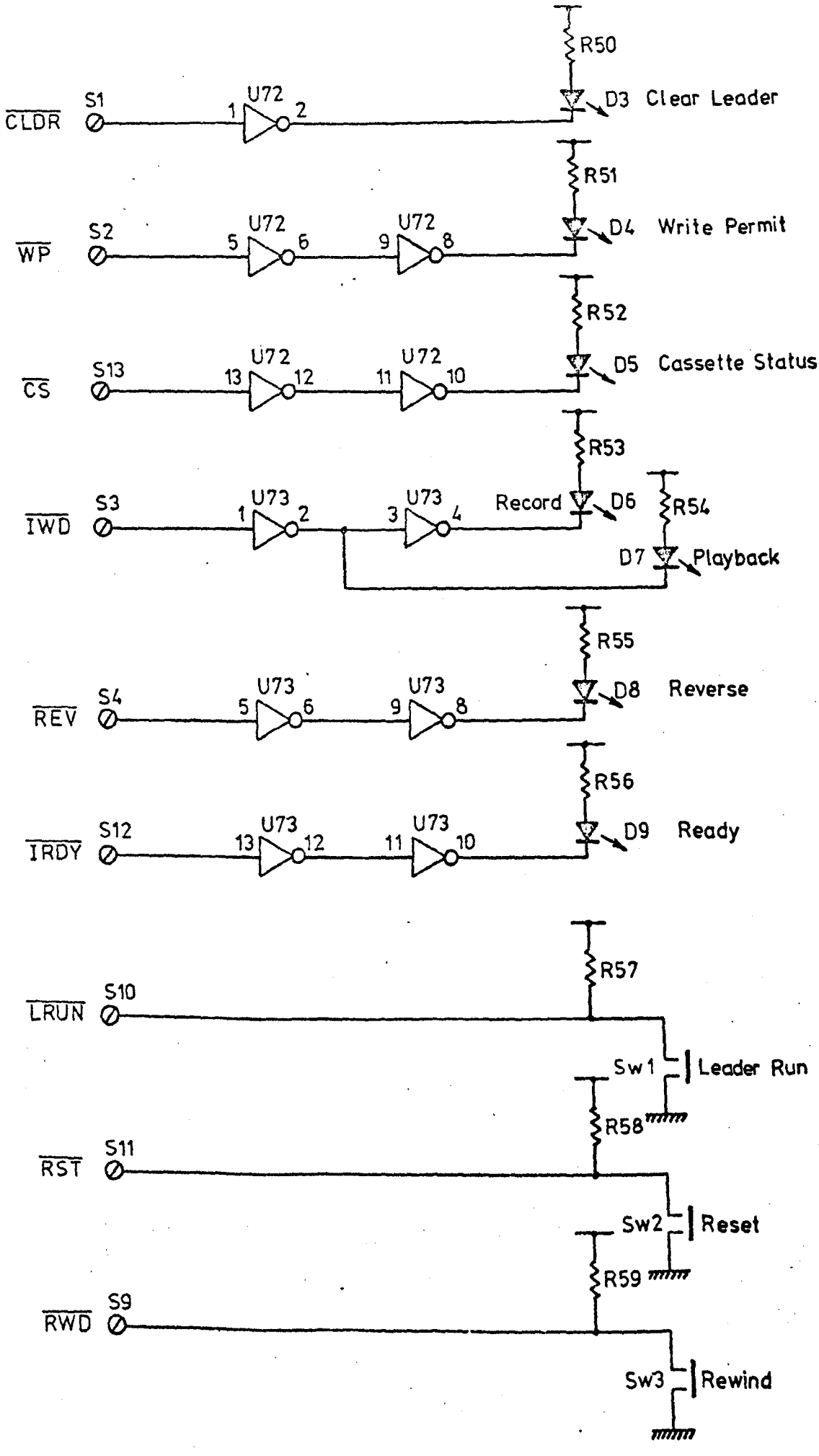


Figure 7.1.12 Front-panel control and indicators.

7.1.3 Miscellaneous Details

The cassette unit interface circuits are constructed on a 235mm × 248mm 'breadboard' designed specifically for use with the Gimini system for random logic layouts. Circuits other than the interface are constructed on the same board (A-D and D-A converter channels - see Vol. II-6) and only the components relevant to this application are shown (Fig. 7.1.13).

The control and indicator circuits and power supply relay are mounted on small pieces of vero-board, housed in the cassette unit (Fig. 7.1.14).

7.1.3.1 Components list

Resistors

All ¼w 5%

R24 - R27	4.7k
R30, R33	2.2k
R31, R32, R34, R35, R38, R40, R42	330Ω
R36, R37	180Ω
R39	3.9k
R41	22k

Capacitors

C10	1000pF ceramic
C11	220pF ceramic
C12, C14 - C16	0.1μF ceramic
C17	6.8μF elect. 10V

Semiconductors

D1, D2	1N914 Silicon diode
Q2	ZTX300

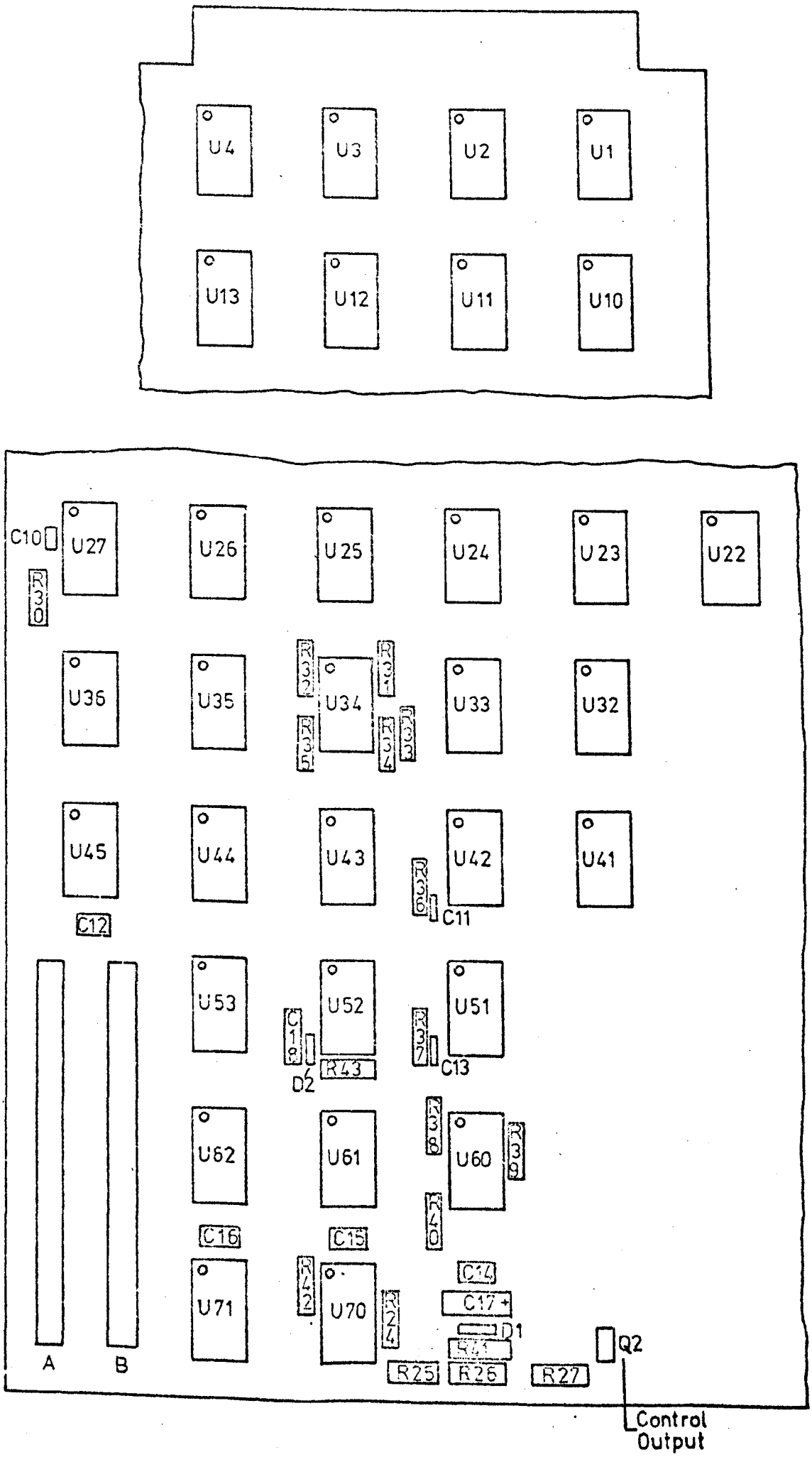


Figure 7.1.13 Cassette interface component layout.

Integrated circuits

U1 - U4,U10 - U13	SN7438
U22,U26,U44,U45,U62,U71	SN7404
U23,U24,U32,U33,U34,U60	SN7403
U25	SN7410
U27	SN74S138
U35,U36,U53,U61	SN74175
U41,U43,U51	SN7400
U42	SN7408
U52	SN74123
U70	SN7474

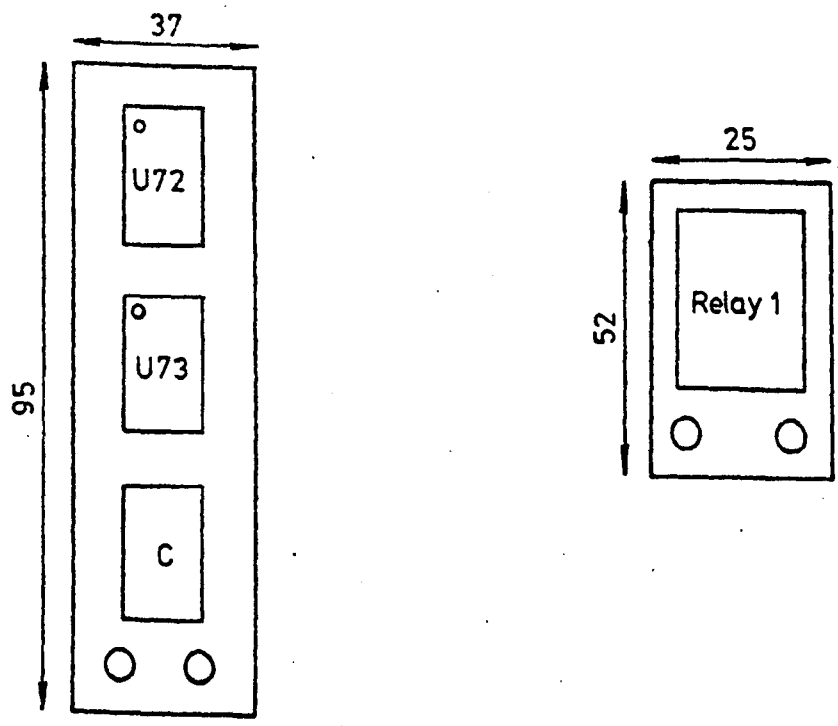


Figure 7.1.14 Component layout for control and indicator circuits and power supply relay.

7.1.3.2 Connection lists

a) Gimini Edge Connector.

J

+5V	36	1	+5V
	37	2	
$\overline{DBL1}$	38	3	$\overline{DBL0}$
$\overline{DBL3}$	39	4	$\overline{DBL2}$
$\overline{DBL5}$	40	5	$\overline{DBL4}$
$\overline{DBL7}$	41	6	$\overline{DBL6}$
$\overline{DBL9}$	42	7	$\overline{DBL8}$
$\overline{DBL11}$	43	8	$\overline{DBL10}$
$\overline{DBL13}$	44	9	$\overline{DBL12}$
$\overline{DBL15}$	45	10	$\overline{DBL14}$
	46	11	
HW2	47	12	HW1
HW4	48	13	HW3
	49	14	
	50	15	
BDRDY	51	16	
+12V	52	17	+12V
+12V	53	18	+12V
	54	19	
	55	20	
	56	21	$\overline{DISBAR}$
	57	22	
	58	23	
$\overline{EBC1}$	59	24	$\overline{EBC0}$
	60	25	$\overline{EBC2}$
	61	26	
	62	27	
	63	28	
	64	29	
	65	30	
	66	31	
	67	32	
	68	33	
	69	34	
GND	70	35	GND

H

+5V	36	1	+5V
	37	2	
$\overline{BADR1}$	38	3	$\overline{BADR0}$
$\overline{BADR3}$	39	4	$\overline{BADR2}$
$\overline{BADR5}$	40	5	$\overline{BADR4}$
$\overline{BADR7}$	41	6	$\overline{BADR6}$
$\overline{BADR9}$	42	7	$\overline{BADR8}$
$\overline{BADR11}$	43	8	$\overline{BADR10}$
$\overline{BADR13}$	44	9	$\overline{BADR12}$
$\overline{BADR15}$	45	10	$\overline{BADR14}$
$\overline{LWBT}$	46	11	$\overline{HGBT}$
$\overline{DTB}$	47	12	$\overline{DW}$
$\overline{ADAR}$	48	13	$\overline{DWS}$
$\overline{IAB}$	49	14	$\overline{INTAK}$
$\overline{NACT}$	50	15	$\overline{BAR}$
$\overline{TCI}$	51	16	$\overline{CK4}$
-12V	52	17	-12V
-12V	53	18	-12V
	54	19	
$\overline{INTRM}$	55	20	$\overline{INTR}$
$\overline{IPRI}$	56	21	$\overline{IPRO}$
IMSKI	57	22	IMSKO
$\overline{BUSRO}$	58	23	$\overline{BUSAK}$
$\overline{BAKI}$	59	24	$\overline{BAKO}$
BMSKI	60	25	BMSKO
	61	26	
	62	27	
	63	28	
	64	29	
	65	30	
	66	31	
$\overline{HALT}$	67	32	$\overline{MCLR}$
	68	33	$\overline{DISCC}$
	69	34	$\overline{DISDTB}$
GND	70	35	GND

b) P.c.b. Connector (A)

c) P.c.b. Connector (B)

	A	B
1	$\overline{DI0}$	$\overline{DO0}$
2	$\overline{DI1}$	$\overline{DO1}$
3	$\overline{DI2}$	$\overline{DO2}$
4	$\overline{DI3}$	$\overline{DO3}$
5	$\overline{DI4}$	$\overline{DO4}$
6	$\overline{DI5}$	$\overline{DO5}$
7	$\overline{DI6}$	$\overline{DO6}$
8	$\overline{DI7}$	$\overline{DO7}$
9	$\overline{DI8}$	$\overline{DO8}$
10	-	-
11	-	$\overline{BE}$
12	-	$\overline{RWDS}$
13	$\overline{LRUN}$	$\overline{OC}$
14	$\overline{RWD}$	LDR
15	$\overline{REV}$	$\overline{CS}$
16	$\overline{LBE}$	$\overline{WP}$
17	$\overline{RUNC}$	IDAV
18	$\overline{IWD}$	$\overline{IRDY}$
19	$\overline{SPDB}$	$\overline{BRST}$
20	$\overline{SPDA}$	$\overline{IBS}$

d) P.c.b. Connector (C)

1	$\overline{CLDR}$	10	$\overline{LRUN}$
2	$\overline{WP}$	11	$\overline{RST}$
3	$\overline{IWD}$	12	$\overline{IRDY}$
4	$\overline{REV}$	13	$\overline{CS}$
5	-	14	+5V
6	-		
7	GND		
8	-		
9	$\overline{RWD}$		

e) Cassette Edge Connector

INPUT/OUTPUT INTERFACE PIN CONNECTIONS

Pin No.	Signal			Direction	Signal Type	
A	LGND	Logic Ground				
B	$\overline{CS}$	Cassette Status	D	Output	Level	
C	$\overline{IWD}$	Write Data Enable		Input	Level	
D	$\overline{IBS}$	Byte Strobe		Input	Pulse	0.5 µs min, 10 ms max
E	$\overline{IR7}$	Data In 7 (Buffer)		Input	Level	
F	$\overline{IR6}$	Data In 6 (Buffer)		Input	Level	
H	$\overline{IR5}$	Data In 5 (Buffer)		Input	Level	
J	$\overline{DI4}$	Data In 4		Input	Level	Bypasses Buffer
K	$\overline{IR0}$	Data In Ø (Buffer)		Input	Level	Least significant bit
L	$\overline{IR1}$	Data In 1 (Buffer)		Input	Level	
M	$\overline{IR2}$	Data In 2 (Buffer)		Input	Level	
N	$\overline{IR3}$	Data In 3 (Buffer)		Input	Level	
P	$\overline{PO0}$	Data Out Ø (Buffer)	O	Output	Level	Least significant bit
R	$\overline{PO1}$	Data Out 1 (Buffer)	O	Output	Level	
S	$\overline{PO8}$	Data Out 8 (Buffer)	O	Output	Level	Most significant bit
T	$\overline{DO4}$	Data Out 4	D	Output	Level	Bypasses Buffer
U	$\overline{PO2}$	Data Out 2 (Buffer)	O	Output	Level	
V	$\overline{PO7}$	Data Out 7 (Buffer)	O	Output	Level	
W	$\overline{PO6}$	Data Out 6 (Buffer)	O	Output	Level	
X	$\overline{PO5}$	Data Out 5 (Buffer)	O	Output	Level	
Y	+5V	Logic Power				
Z	-12V	Motor and Signal Power				
AA	$\overline{RWD}$	Rewind		Input	Pulse	0.5 µs min, Level if on clear leader.
BB	$\overline{REV}$	Reverse		Input	Level	
CC	$\overline{LBE}$	Load Buffer Enable		Input	Level	
1	CGND	Chassis Ground				
2	MGND	Motor Ground				
3	SGND	Signal Ground				
4						
5	$\overline{SPDA}$	Speed Control A		Input	Level	
6	+12V	Motor and Signal Power				
7	MO	Marker Out	D	Output	Level	*
8	CHR	Character		Input	Level	*
9	$\overline{WP}$	Write Permit	D	Output	Level	
10	$\overline{IR8}$	Data In 8 (Buffer)		Input	Level	Most significant bit
11	DC	Data Clock	T	Output	Pulse	*
12	$\overline{RWDS}$	Rewind Status	D	Output	Level	
13	$\overline{BRST}$	Reset		Input	Pulse	
14						
15	$\overline{RUNC}$	Run Continuous		Input	Level	
16	$\overline{IRDY}$	Ready	D	Output	Level	
17	DO	Serial Data Out	D	Output	Level	*
18	$\overline{MD}$	Marker Delete		Input	Level	*
19	IDAV	Data Available	D	Output	Level	
20	$\overline{PO3}$	Data Out 3 (Buffer)	O	Output	Level	
21	$\overline{LRUN}$	Leader Run		Input	Level	
22	OC	Operation Complete	D	Output	Level	
23	LDR	Clear Leader	D	Output	Level	
24	SPDB	Speed Control B		Input	Level	
25	BE	Buffer Empty	D	Output	Level	

* Used only in special application. Consult Racal-Thermionic for details of operation.

T signifies active pull up TTL output

O signifies open collector TTL output

D signifies DTL output

7.1.3.3 Mechanical details

The P72 cassette drive mechanism and associated drive and control circuits are built into a 482mm rack-mountable unit, complete with a comprehensive front-panel (Figs. 7.1.15 and 7.1.16). Little use is made of the front-panel at present. It is intended to form the basis of the next phase of the design, a self-contained, line-local cassette storage unit.

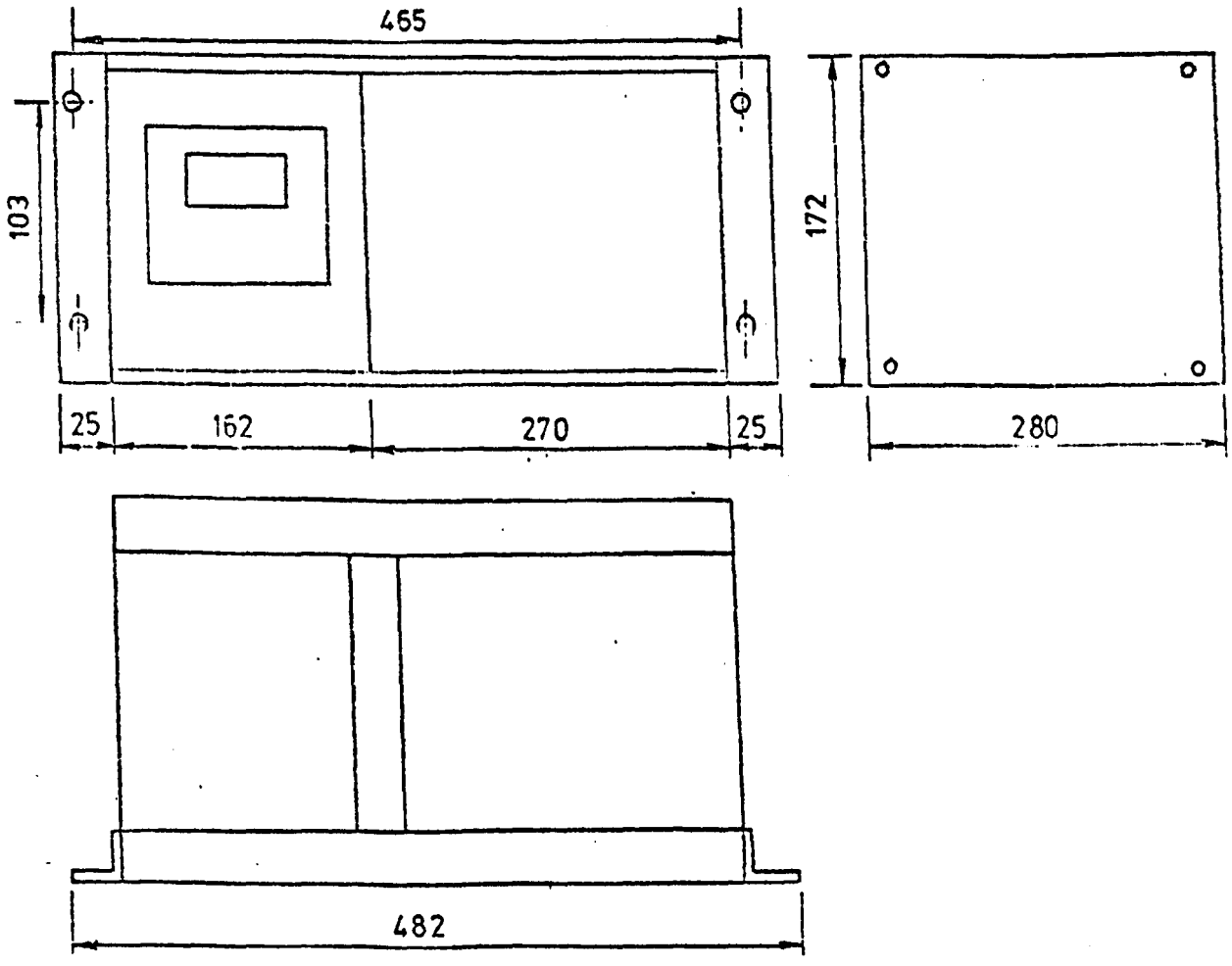


Figure 7.1.15 Overall case dimensions.

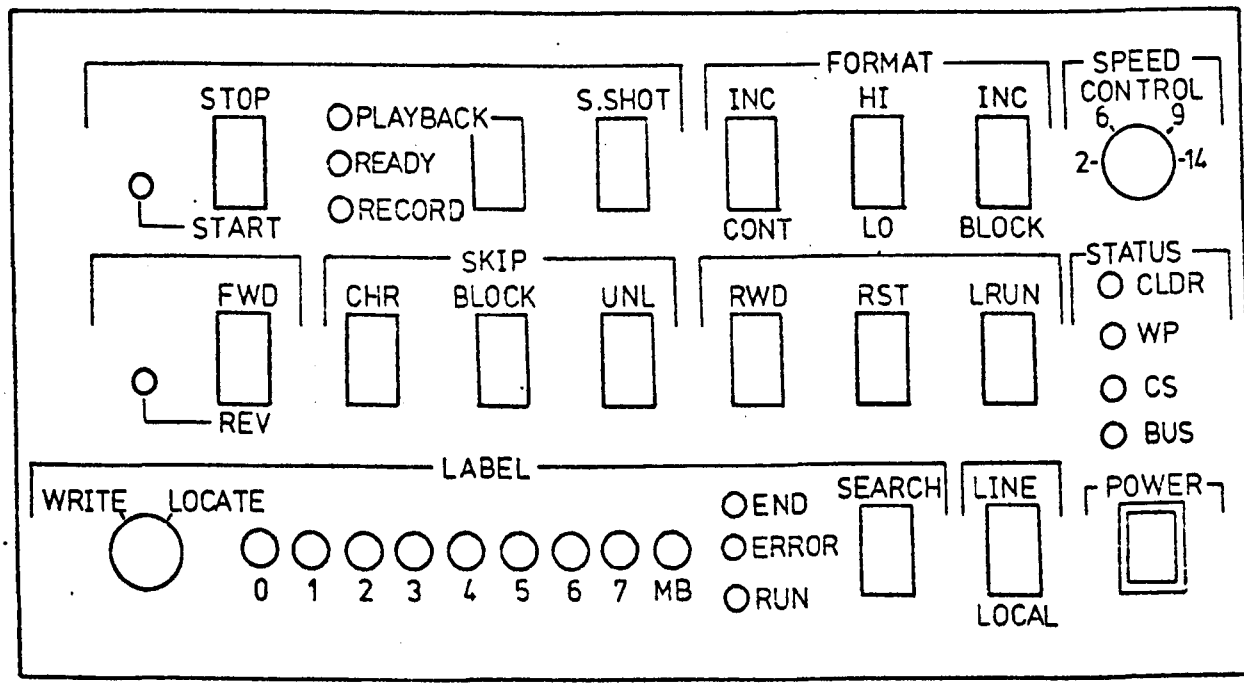


Figure 7.1.16 Front-panel layout.

7.1.4 References

1. Cassette User Guide, Internal Report, R. A. Comley, Dec. 1977.
2. Series 1600 Microprocessor System - Documentation, General Instrument Microelectronics, Dec. 1976.
3. Digideck - Model P72, Technical Handbook, Racal Thermionic Ltd., Jan. 1975.

7.2 User Guide

7.2.1 Introduction

The Racal P72 is a general purpose digital data storage unit which uses standard computer grade 3.81mm cassette tapes.

To the user, the P72 appears to read and write data incrementally at any rate up to 350 bytes per sec.. Operation at rates of up to 40 bytes per sec. is truly incremental, i.e. the tape moves in discrete steps. At higher data rates, data are stored in a thirteen character first in-first out (FIFO) buffer and the tape is moved continuously, its speed being servo controlled and a function of the number of characters stored in the buffer.

The P72 may also be used in a continuous mode at any one of four predefined tape speeds, 2, 6, 9 or 14 ins. per sec., giving a maximum data transfer rate of 630 bytes per sec..

Data are transferred at the interface as eight-bit parallel bytes. A ninth bit is also available at incremental rates below 40 bytes per sec. and in continuous modes with the FIFO buffer disabled.

7.2.2 Status indicators

a) Front-panel

Seven status signals are displayed on the front-panel via LEDs. The meaning attached to each of the indicators is:-

CLDR - Clear leader.

When illuminated, this indicates that clear leader, at either the beginning or end of tape is detected. Only the LRUN and RWD signals will be honoured in this state.

W.P. - Write permit

When illuminated, this indicates that the write tab

is in place on the cassette and that write-erase operations may be executed. Note that during rewind, the writing or erasure of data is inhibited and the indicator is hence turned off.

C.S. - Cassette status.

When illuminated, this indicates that a cassette is in place, the cassette holder is closed and that the deck is in an operational condition.

REV - Reverse.

When illuminated, this indicates that the cassette is set for reverse motion of the tape.

Playback-Record.

When illuminated, these indicate the mode of operation of the cassette deck.

Ready.

When illuminated, this indicates that the cassette is ready to accept the next read or write strobe. A busy indication (LED off) will be displayed under the following conditions:-

i) Reset
ii) Cassette rewinding
iii) No cassette loaded
iv) Write command issued to a cassette which has no write permit tab
v) During write operation, with the FIFO buffer enabled, whilst a byte is being loaded into the buffer
vi) During write operations, with the buffer disabled, whilst a character is being transferred onto tape.

Under all other conditions, ready will be illuminated.

b) Gimini

Six status signals are taken to the Gimini, and may be strobed onto the data bus with a MVI@ 167741,Rx command. The

status signals are:-

$\overline{BE}$ - Buffer empty	DBL0
$\overline{RWDS}$ - Rewind status	DBL1
$\overline{OC}$ - Operation complete	DBL2
LDR - Clear leader	DBL3
$\overline{WP}$ - Write permit	DBL4
$\overline{CS}$ - Cassette status	DBL5

Three of these status conditions have been defined in the previous section (7.2.2.a). A logic one on the $\overline{WP}$ or $\overline{CS}$ input and a logic zero on the LDR input, correspond to the LED illuminated state.

The meaning associated with the other three inputs is:-

$\overline{BE}$ - Buffer empty.

A logic one on this line indicates that the FIFO buffer is empty.

$\overline{RWDS}$ - Rewind status.

A logic one on this line indicates that the cassette is being rewound.

$\overline{OC}$ - Operation complete.

A logic one on this line, when the deck is operating incrementally, indicates that the FIFO buffer is empty and the tape is stationary. In the read mode, with the buffer enabled, thirteen bytes will be contained in the buffer. Note, this status signal is meaningless if LRUN is applied and does not take account of the stop time required when operating in continuous modes (see sect. 7.2.4).

7.2.3 Front-panel control

The user is provided with control of three functions via the front-panel switches; LRUN, RWD and RST. These may, of course, be controlled by the Gimini via software commands.

LRUN - Leader run.

This command is used to advance the tape off clear

leader at the beginning of tape (BOT). All operations except LRUN and RWD are inhibited whilst clear leader is sensed (indicated by the clear leader status).

If clear leader is sensed at BOT then LRUN should be issued and maintained until CLDR status goes to a logic one. Any speed setting may be used for this operation.

At end of tape (EOT), a rewind command should be issued as reverse operation is inhibited on clear leader.

RWD - Rewind.

A rewind may be initiated from any point on tape. It is the only way to reverse from clear leader at EOT. The rewind command must be maintained for as long as clear leader is sensed but may be removed as soon as magnetic tape is detected. An internal latch then maintains the rewind command until clear leader is again sensed at BOT. A rewind command may only be cancelled by a reset command. To avoid possible loss or erasure of data, rewind should only be issued with the cassette in a read mode.

BRST - Reset.

A reset command causes both the cassette logic and the Gimini interface to be reset, with the following results:-

i) the control word latch is cleared (see later) with the result that a low-speed incremental read is selected

ii) the data contained in any other of the interface latches are cleared

iii) the FIFO buffer is cleared

iv) if a rewind is in progress it is cancelled

v) if a read strobe has not been honoured (i.e. an attempt has been made to read from an unrecorded cassette) it is cancelled, with the result that the tape will stop.

7.2.4 Modes of operation

Two basic modes exist for the reading or writing of data, incremental and continuous. For incremental operations, two options are available, high speed (buffer enabled) and low speed (buffer disabled). In the incremental mode, the speed control settings, $\overline{SPDA}$ and $\overline{SPDB}$, have no meaning and should be held low.

For continuous operations the tape is run at a constant speed, determined by one of four predefined tape speeds, and data must be supplied (write) or accepted (read) at the rate set by the P72. The four speed settings are:-

$\overline{SPDA}$	$\overline{SPDB}$	Speed (ips)
0	0	2
1	0	6
0	1	9
1	1	14

Before any read or write operations are instigated, a group of eight control lines should be set to specify the desired operation, as given in Table 7.2.2. To set the control word, it should be placed in the low-byte position of the Gimini data word and written to location OCT167742 (see Table 7.2.3),the control word latch.

7.2.4.1 Write operations

i) Incremental - High speed.

In this mode, eight-bit data may be transferred at rates of up to 350 characters per sec.. The data are written to the FIFO buffer, internal to the P72, and the tape speed is adjusted automatically to match the speed of the incoming data. Typically, the buffer will contain several bytes at any given time and to terminate a high speed incremental write operation, after the final write command, the operation complete ($\overline{OC}$) signal should be checked to ensure that the

buffer is empty.

To perform a high speed incremental write operation, after status has been checked, the control word should be set as indicated in Table 7.2.2 and written to location OCT167742. External branch input 1 [3] should then be tested to check for the deck's readiness to accept data. When the data are ready, the first data word should be placed in the low-byte position and written to location OCT167743 and external branch input 1 again checked for ready. When ready, the next data word may be transferred and so on until all words have been transmitted. The operation complete ($\overline{OC}$) signal should then be awaited before program execution is resumed.

A program listing for a typical high speed incremental write operation is given in Appendix 7.A.

ii) Incremental - Low speed.

In this mode, the FIFO buffer is disabled and the ninth data bit is available, but the data input rate is restricted to a maximum of 40 bytes per sec.. The control word should be set as indicated in Table 7.2.2 and the same sequence as performed in the high speed incremental mode, used to transfer the data.

iii) Continuous.

This mode is similar to the low speed incremental mode with the buffer disabled and all nine data bits available, except that now, data rates of up to 630 bytes per sec. are possible. No account is taken of the start-stop delay of the tape in this mode. The user should allow an appropriate delay, as given in Table 7.2.1, which is measured from the time at which the run-continuous ($\overline{RUNC}$) bit is set or cleared in the control word latch.

Two recording formats are possible in the continuous mode, one which results in a record which may be read either incrementally or continuously (incremental structure), the other

provides data which may only be read continuously (block structure). For the incremental structure, slower data rates are used which results in the generation of an 'inter-character gap', equivalent to that which is generated in a high speed incremental write operation. For the block structure, the intercharacter gap is removed, thereby increasing the packing density and operating speed of the cassette.

The maximum data rates for each structure at the four speed settings are given in Table 7.2.1.

$\overline{SPDA}$	$\overline{SPDB}$	Speed (ips)	Start-stop delay (ms)	Data rate (bytes /sec) Structure	
				Inc.	Block
0	0	2	10	50	90
1	0	6	20	150	270
0	1	9	30	225	410
1	1	14	60	350	630

Table 7.2.1 Data rates and start-stop delays for various speed settings.

7.2.4.2 Read operations

i) Incremental - High speed.

As for the write mode, eight-bit data may be transferred at rates up to 350 characters per sec.. Again, the FIFO buffer is used to adjust, automatically, the speed of the tape deck to match the data rate from the cassette. When a high speed incremental read operation is terminated, the tape continues in motion, decelerating, and finally stops when the buffer is full (i.e. the buffer contains thirteen bytes). Operation complete is indicated when the tape has stopped and the buffer is full.

To perform a high speed incremental read operation, after status has been checked, the control word should be set as indicated in Table 7.2.2 and written to location

OCT167742. External branch input 1 should then be tested to ensure that the deck is ready. When ready, the first read request is issued by an access of location OCT167744 and then a wait on external branch input 2 implemented to await the data available (IDAV) signal. When data available is indicated, the data may be read from location OCT167745 into the low-byte of the processor word; bits 8 - 15 are set to logic one. The next read request may then be issued and the sequence repeated until all words have been transferred. The operation complete signal should then be awaited before the program is continued. Note, the FIFO buffer will contain thirteen bytes at this stage.

A program listing for a typical high speed incremental read operation is given in Appendix 7.B

ii) Incremental - Low speed.

This mode gives true incremental read operation at rates of up to 40 characters per sec., i.e. no data are held in the buffer and so all nine bits are available. The control word should be set as indicated in Table 7.2.2 and the same sequence as performed in the high speed incremental read mode, used to transfer the data.

iii) Continuous.

The continuous read mode may be used to read or 'search' data which has been written in either incremental or block format, at speeds of up to 630 characters per sec.. When a particular character is sought, there may be no interblock gaps in which to stop, however after stopping and clearing the buffer with reset, one incremental read operation (low speed) is sufficient to resynchronize with the recorded data.

No account is taken of the start-stop delay of the tape and these should be included as for the continuous write operations (see sect. 7.2.4.1 (iii)).

Operation in this mode is similar to the low speed incremental read with a read request command issued and when

data available is indicated (via external branch input 2), all nine data lines may be sampled. After reading, and within 500μs of the data available indication, a read request must be issued to clear the buffer output for the next character, unless no more read operations are required. In this event, the run-continuous command should be cancelled and a sufficient stop time allowed, before program execution is resumed.

7.2.5 Care of equipment and cassette handling procedures

To ensure the optimum performance from the cassette unit the following instructions should be observed:-

i) Do not leave cassettes unprotected. They should be either mounted in the deck or enclosed in approved storage cases.

ii) New cassettes should be run-in by several full passes of the tape at a high operating speed. This should be done with the tape guides and heads in contact with the tape. The run-in removes loose oxide particles and polishes the tape.

iii) Tape heads and guides should be cleaned frequently (at least every eight operating hours) to remove oxide accumulations.

iv) The first and last 30cms of tape should not be used as these sections are most subject to wear, wrinkling and contamination.

v) The tape should always be rewound to clear leader before the cassette is removed from the deck.

vi) The cassette should be discarded after 1000 passes (approx.) or earlier if permanent errors begin to appear.

COMMAND STRUCTURE

SIGNAL	PIN No.	INCREMENTAL READ FORWARD LOW SPEED	INCREMENTAL READ FORWARD HIGH SPEED	INCREMENTAL WRITE FORWARD LOW SPEED	INCREMENTAL WRITE FORWARD HIGH SPEED	INCREMENTAL READ REVERSE LOW SPEED	INCREMENTAL READ REVERSE HIGH SPEED	CONTINUOUS WRITE FORWARD	CONTINUOUS READ FORWARD	CONTINUOUS READ REVERSE	REWIND	WRITE FORWARD FROM LEADER	READ FORWARD FROM LEADER
		COMMAND											
$\overline{RUNC}$	15	HI	HI	HI	HI	HI	HI	LO	LO	LO	HI	HI	HI
$\overline{LRUN}$	21	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI	LO	LO
$\overline{IWD}$ *	C	HI	HI	LO	LO	HI	HI	LO	HI	HI	HI	LO	HI
$\overline{LBE}$	CC	HI	LO	HI	LO	HI	LO	HI	HI	HI	HI	HI	HI
$\overline{SPDA}$	5	HI	HI	HI	HI	HI	HI	X	X	X	HI	X	X
$\overline{SPDB}$	24	HI	HI	HI	HI	HI	HI	X	X	X	HI	X	X
$\overline{REV}$	BB	HI	HI	HI	HI	LO	LO	HI	HI	LO	HI	HI	HI
$\overline{RWD}$	AA	HI	HI	HI	HI	HI	HI	HI	HI	HI	LO	HI	HI
$\overline{BRST}$	13	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI
$\overline{CS}$	B	LO	LO	LO	LO	LO	LO	LO	LO	LO	LO	LO	LO
$\overline{WP}$	9	X	X	LO	LO	X	X	LO	X	X	X	LO	X
LDR	23	LO	LO	LO	LO	LO	LO	LO	LO	LO	X	X	X
$\overline{RWDS}$	12	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI	HI
$\overline{IRDY}$	16	LO	LO	LO	LO	LO	LO	LO	LO	LO	X	LO	LO
$\overline{OC}$	22	LO	LO	LO	LO	LO	LO	LO	LO	LO	LO	X	X

HI = +2.4 to +5Volt

LO = 0V to +0.4Volt

X = May be either HI or LO

* A change in $\overline{IWD}$, either from write to read or read to write must be followed by a delay of at least 40 ms before any command causing tape motion is issued ($\overline{IBS}$, $\overline{RUNC}$, $\overline{LRUN}$, $\overline{RWD}$)

Table 7.2.2 Control word settings.

OCT16774	- 0	Reset
	- 1	Read status
	- 2	Write control word
	- 3	Write data
	- 4	Read request
	- 5	Read data
	- 6	Not used
	- 7	Not used

Table 7.2.3 Address allocation.

Bit		
0	Buffer empty	$\overline{BE}$
1	Rewind status	$\overline{RWDS}$
2	Operation complete	$\overline{OC}$
3	Clear leader	LDR
4	Write permit	$\overline{WP}$
5	Cassette status	$\overline{CS}$

Table 7.2.4 Status word.

7.2.6 References

1. Cassette Storage Unit - Circuit Description, Internal report, R. A. Comley, Dec. 1977.
2. Digideck - Model P72, Technical Handbook, Racal Thermionic Ltd., Jan. 1975.
3. Series 1600 Microprocessor System Documentation, General Instrument Microelectronics, Dec. 1976.

Appendix 7.A

```
 REL $WRTDA
;;;;
;
;       WRITE DATA TO CASSETTE
;       HI-SPEED INCREMENTAL
;       200 BYTES PER SEC.
;
;;;;
R0 = 0
R1 = 1
R2 = 2
R3 = 3
R4 = 4
R5 = 5
SP = 6
PC = 7
  GLOB $WRTDA
$WRTDA MVII 24,R0          ;SET FOR HI-SPEED INC. WRITE
       MVO R0,167742       ;WRITE CONTROL WORD
SWAT   BEXT CONT,1         ;ALLOW START-UP DELAY
       B SWAT
CONT   MVI 167732,R0       ;READ FROM ADC
       ANDI 377,R0         ;CLEAR TOP BITS
       MVO R0,167743       ;WRITE DATA TO CASSETTE
WAIT   BEXT CONT,0         ;WAIT FOR REAL-TIME CLOCK
       B WAIT
       END
```

Appendix 7.B

```
 REL $RDDTA
;;;;
;
;       READ DATA FROM CASSETTE
;       HI-SPEED INCREMENTAL
;       200 BYTES PER SEC.
;
;;;;
R0 = 0
R1 = 1
R2 = 2
R3 = 3
R4 = 4
R5 = 5
SP = 6
PC = 7
  GLOB $RDDTA
$RDDTA MVII 20,R0          ;SET FOR HI-SPEED INC. READ
       MVO R0,167742       ;WRITE CONTROL WORD
SWAT   BEXT CONT,1         ;ALLOW START-UP DELAY
       B SWAT
CONT   MVO R0,167744       ;ISSUE READ REQUEST
RWAT   BEXT READ,2         ;WAIT FOR IDAV
       B RWAT
READ   MVI 167745,R0       ;READ BYTE FROM CASSETTE
       ANDI 377,R0         ;CLEAR TOP BITS
       MVO R0,167734       ;OUTPUT VIA DAC
WAIT   BEXT CONT,0         ;WAIT FOR REAL-TIME CLOCK
       B WAIT
       END
```

Chapter 8

12v - 0 - 12v Stabilized Power Supply

8.1 General Description

Circuit and constructional details of a 12V - 0 - 12V, stabilized power supply with an output capability of 3A + 3A, are described.

The supply is completely protected against over-voltage and reverse polarity outputs. The over-voltage protection is provided by means of conventional 'crow-bar' circuits, constructed from a zener diode, resistor and thyristor (SCR). The reverse polarity protection is provided by a single diode, which is capable of withstanding the full output current.

The supply can tolerate an indefinite short-circuit output, but overheating (and hence thermal breakdown) may occur if the condition exists for a considerable period.

Note: the supply case is connected to earth but the 0V output is completely isolated.

8.2 Miscellaneous Details

The complete circuit diagram and mechanical construction details are given in the following pages.

The components used in the construction of the supply are:-

R10,R11,R20,R21	0.33Ω 2.5w wirewound resis.
R12,R22	68Ω ½w ±10% resistors
R13,R23	470Ω ½w ±10% resistors
R14,R24	4.7kΩ ½w ±10% resistors
C10,C20	10,000μF Elect. 40V
C11,C21	0.47μF polyester
C12,C22	10μF Elect. 25V
B10,B20	Bridge Rec. - type REC46
D10,D20	Si Diode - I_f = 26A
SCR10,SCR20	Thyristor - type BTY79-400R
Z10,Z20	13V ±5% Zener diode - BZY88

Tr10,Tr20	BC461
Tr11,Tr21	PNP3055
REG10,REG20	12V Regulator - type MVR-12V
T1	Mains Transformer - 15V @ 3.3A, 15V @ 3.3A
1 500mA Fuse	
2 3A Fuse	
1 Barrier strip	
5 Heatsinks - 2.1°C/W	

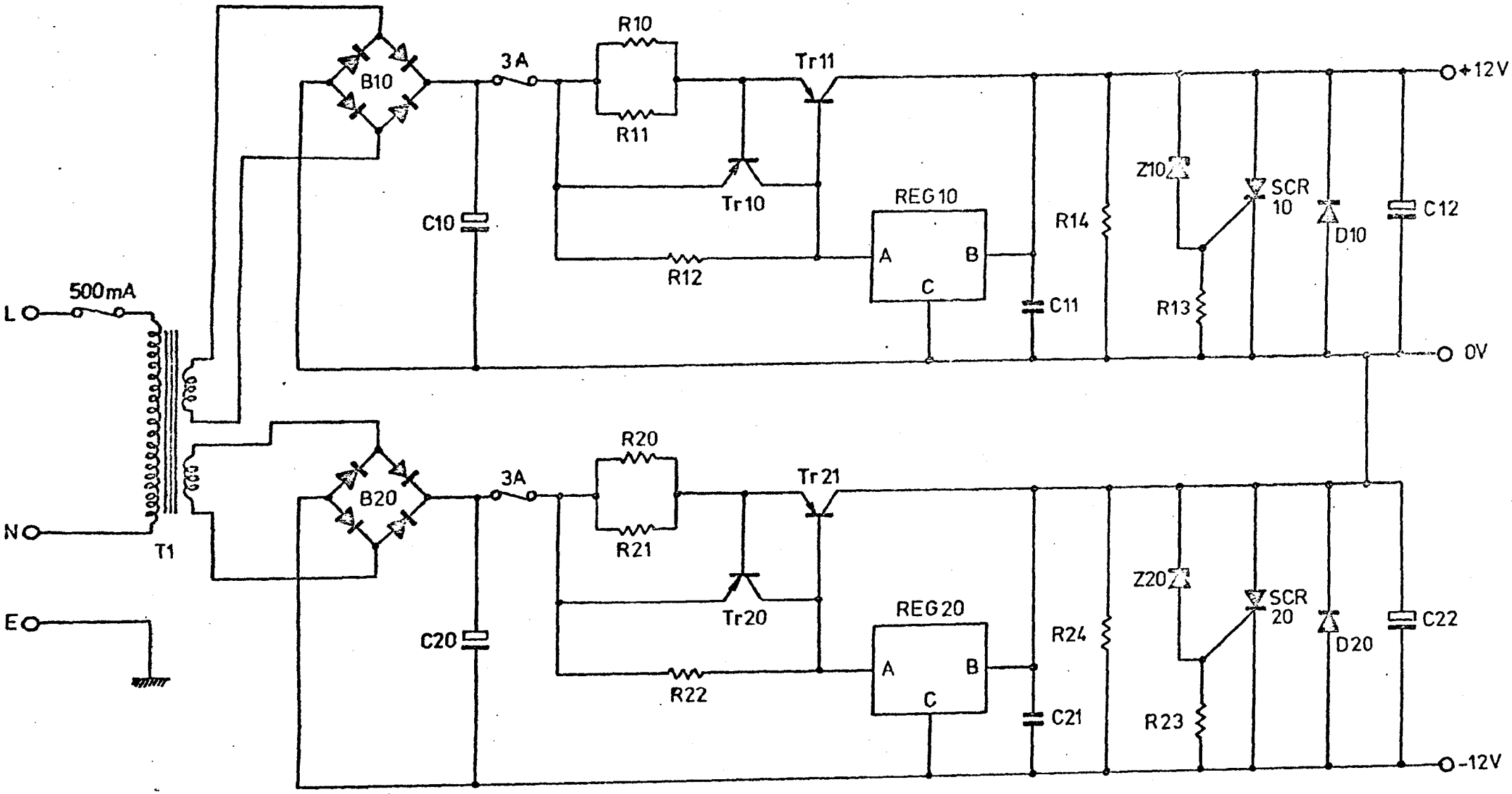


Figure 8.1 Circuit diagram for the 12V - 0 - 12V power supply.

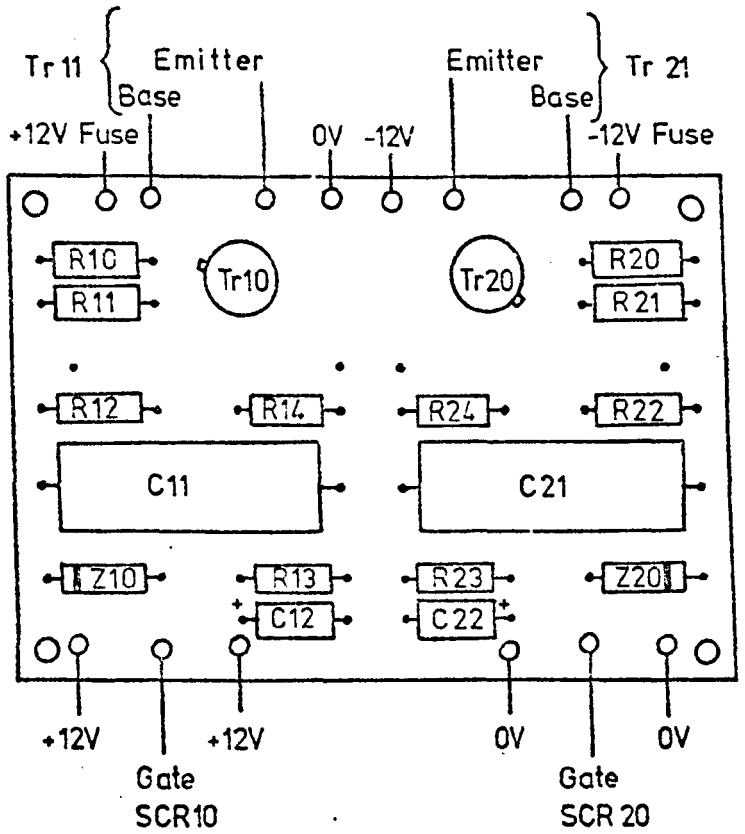
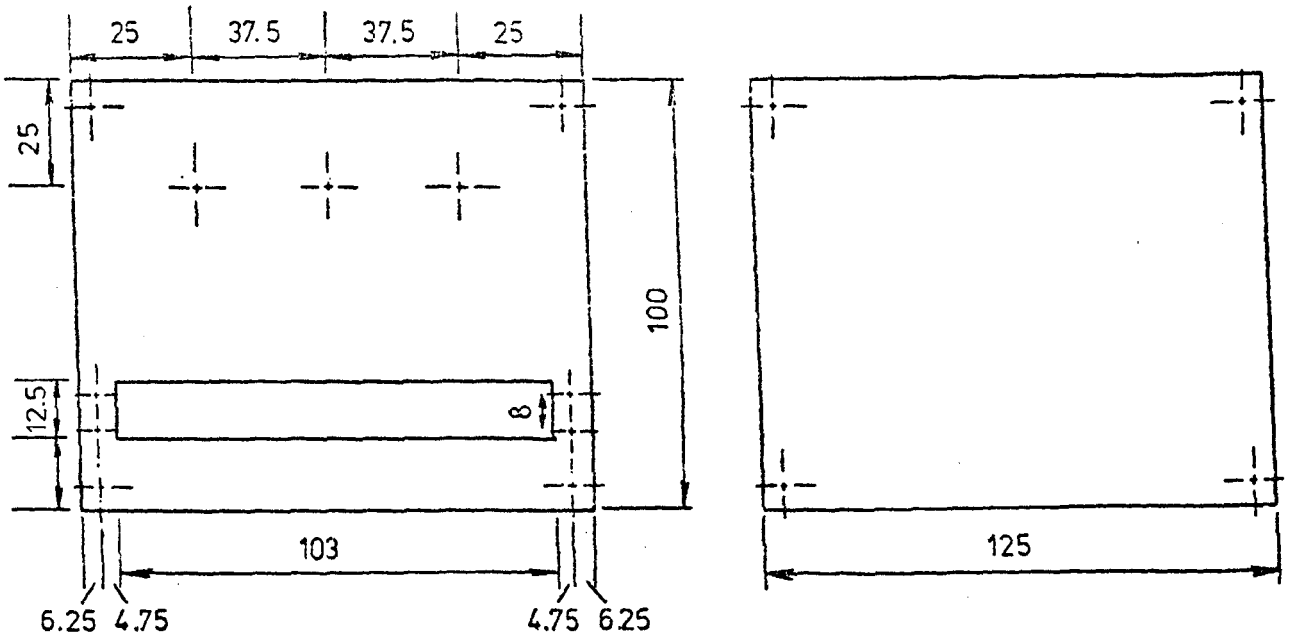
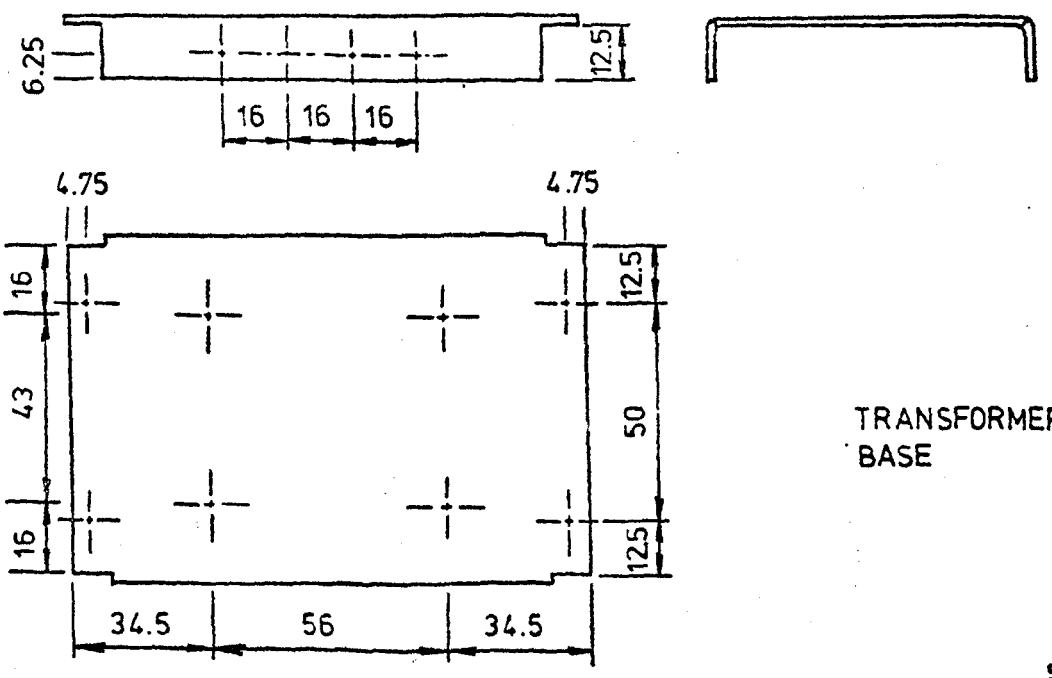


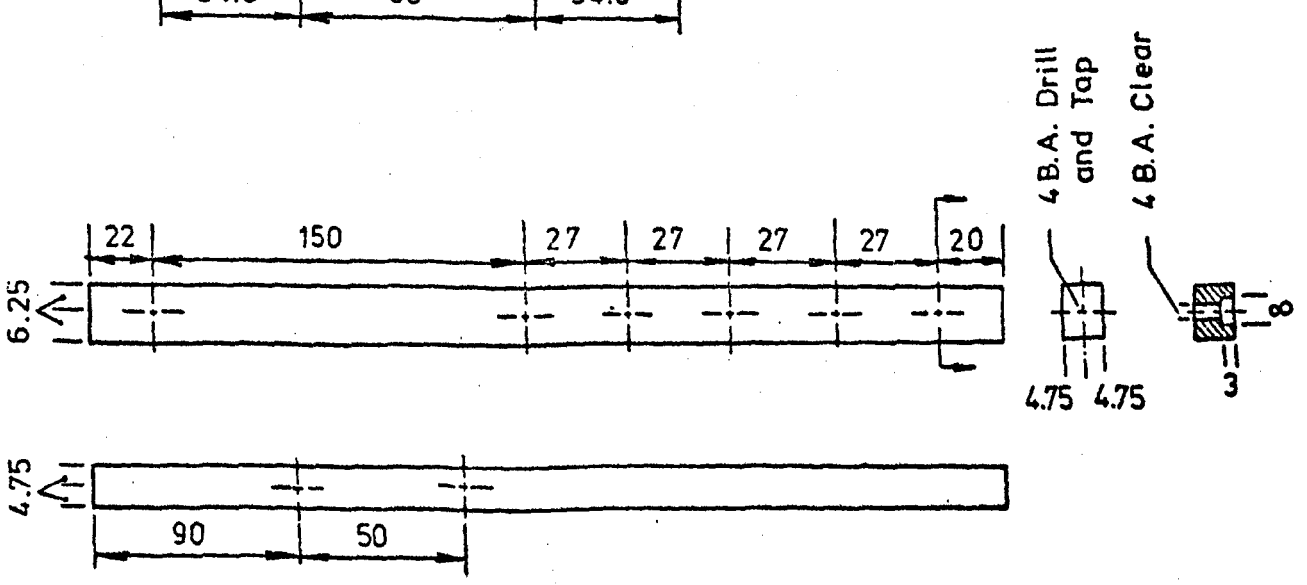
Figure 8.2 Circuit board layout.



FRONT PLATE

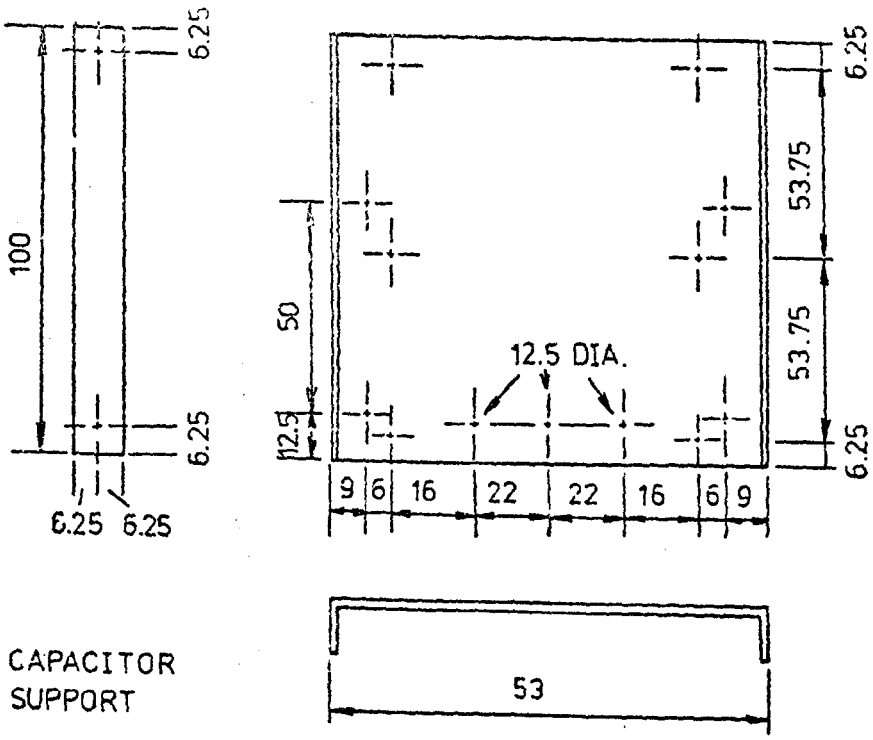


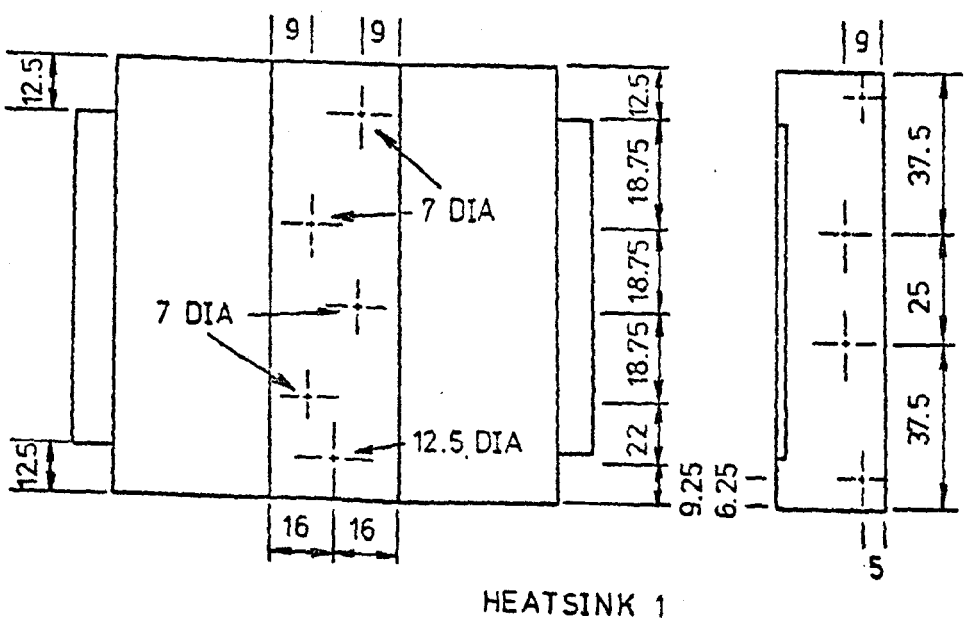
TRANSFORMER BASE



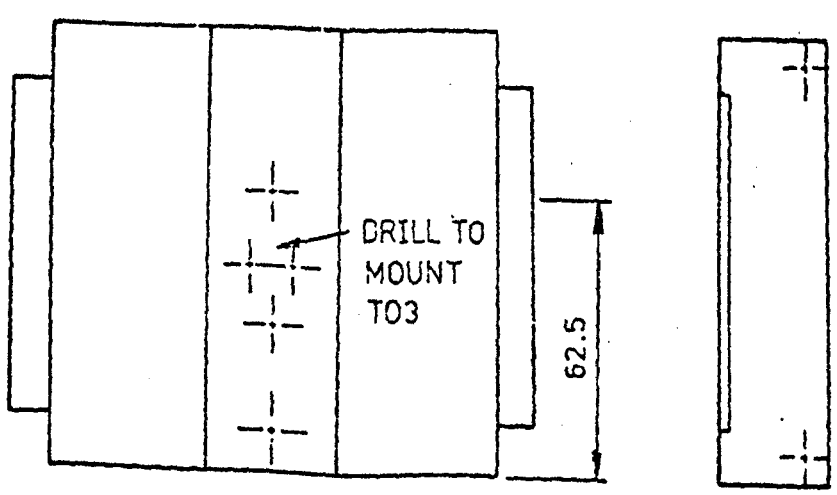
SIDE RAILS

Figure 8.3 Mechanical details.





HEATSINK 1



HEATSINKS 2-5

Figure 8.3 Mechanical details (cont.).

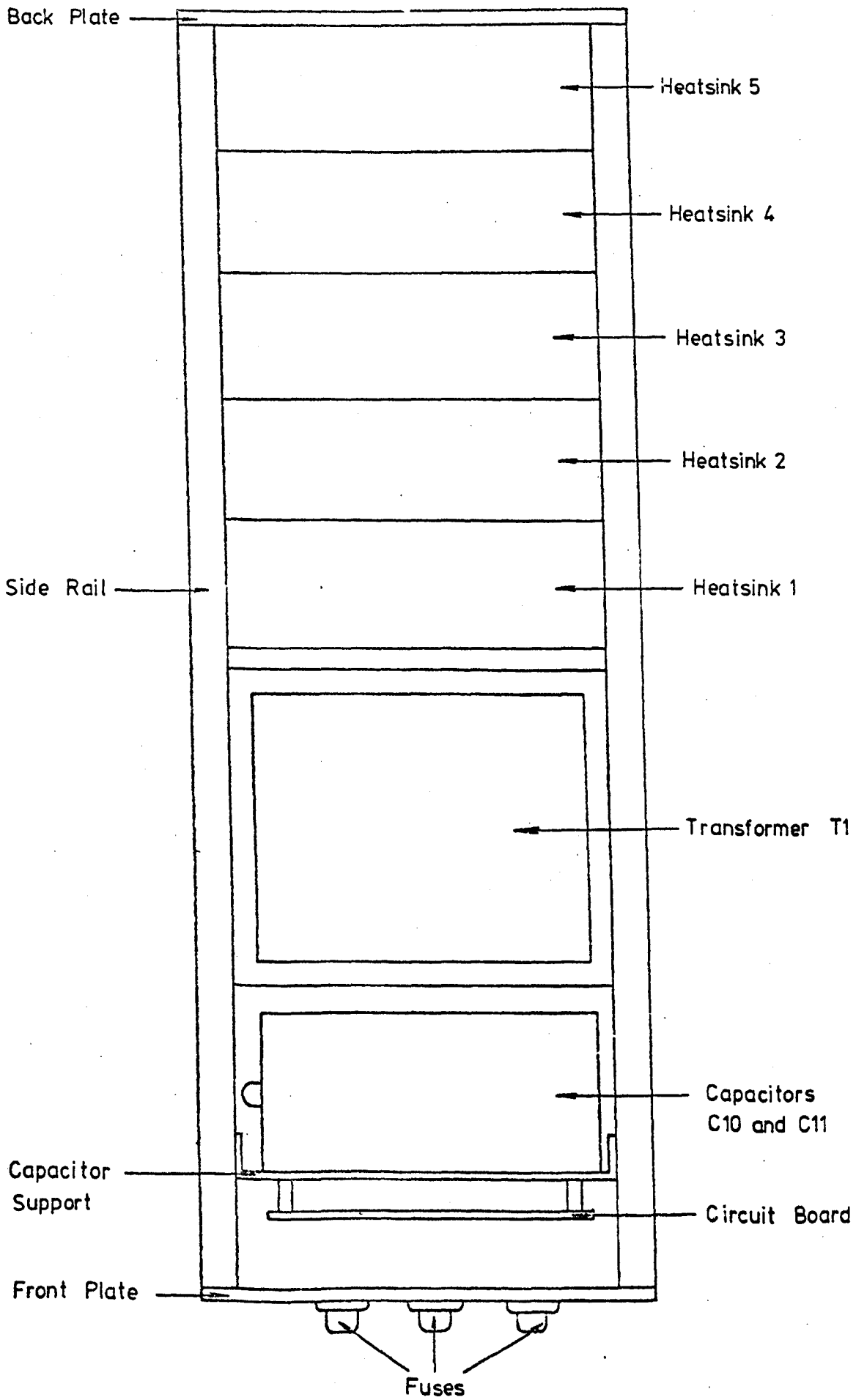


Figure 8.4 The assembled power supply (from above).

Chapter 9

Miscellaneous Details

1. Mobile Rack Units

9.1 Mobile Rack Units

Both prototype RSPs are mounted in 482mm. (19 ins.) rack units, which are equipped with castors so as to facilitate easy movement of the systems around the laboratory. These have been designed and constructed specifically for the RSPs and working drawings together with the major dimensions are given in the next few pages. All dimensions are in mm. and a scale of one-eighth full-size (approx.) has been used.

Details of the fixing brackets have not been included as these are specific to the equipment mounted in the rack and are hence variable.

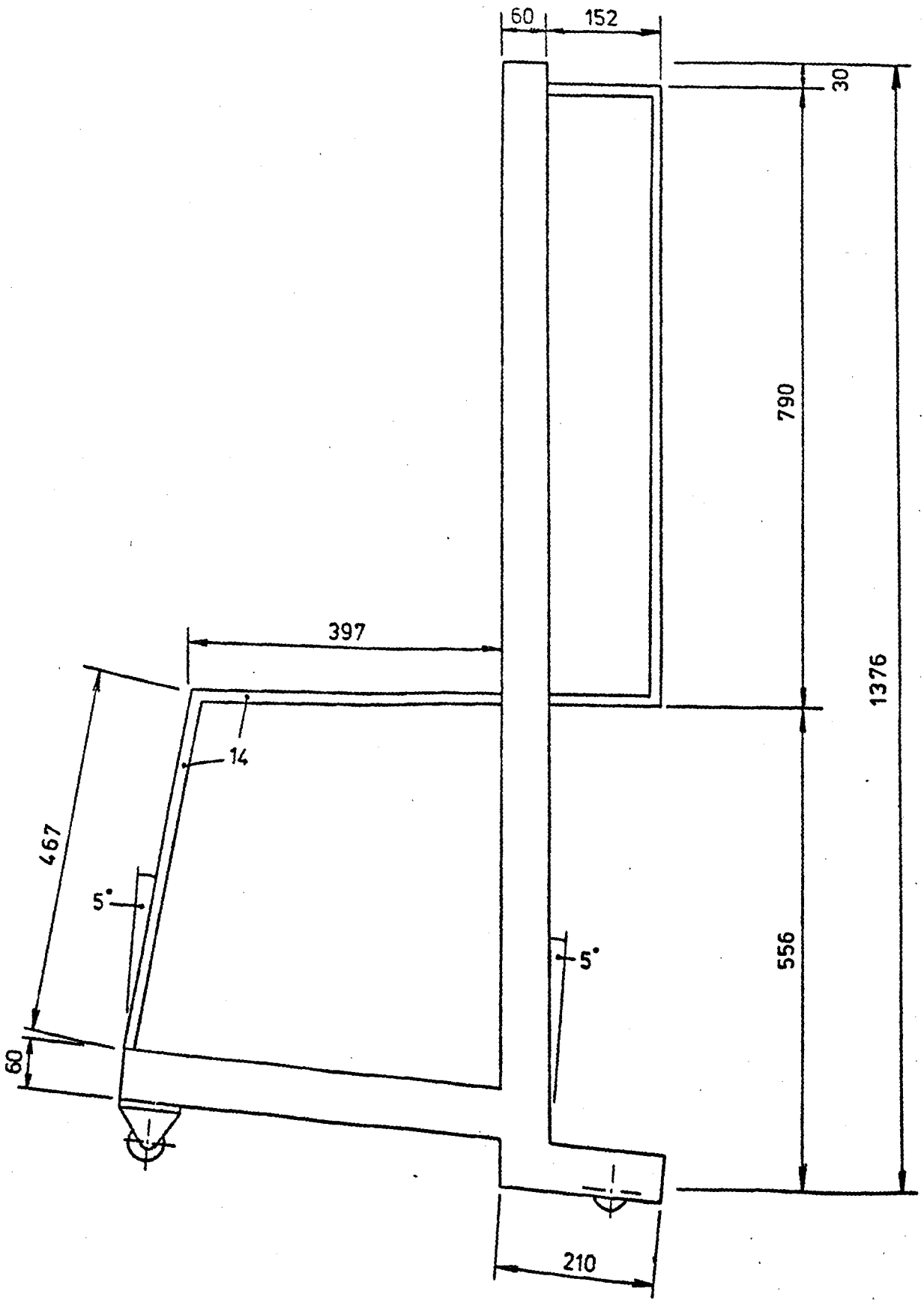


Figure 9.1 Mobile rack units - main dimensions.

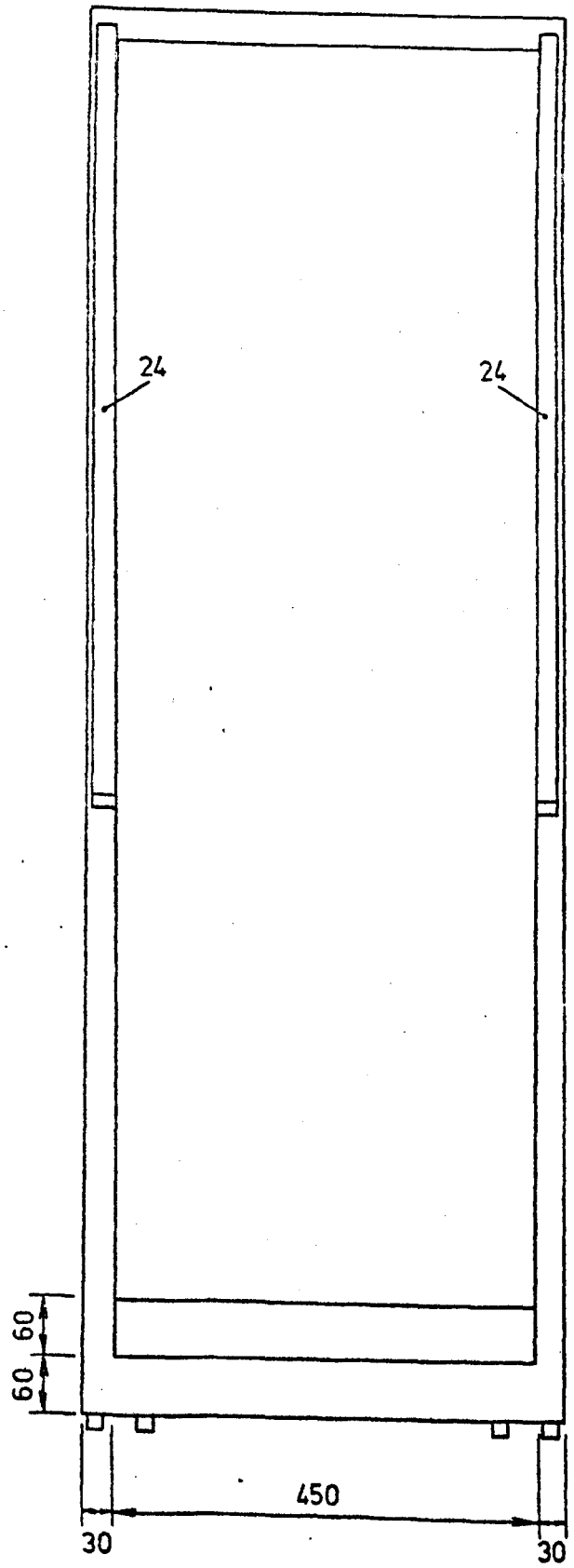
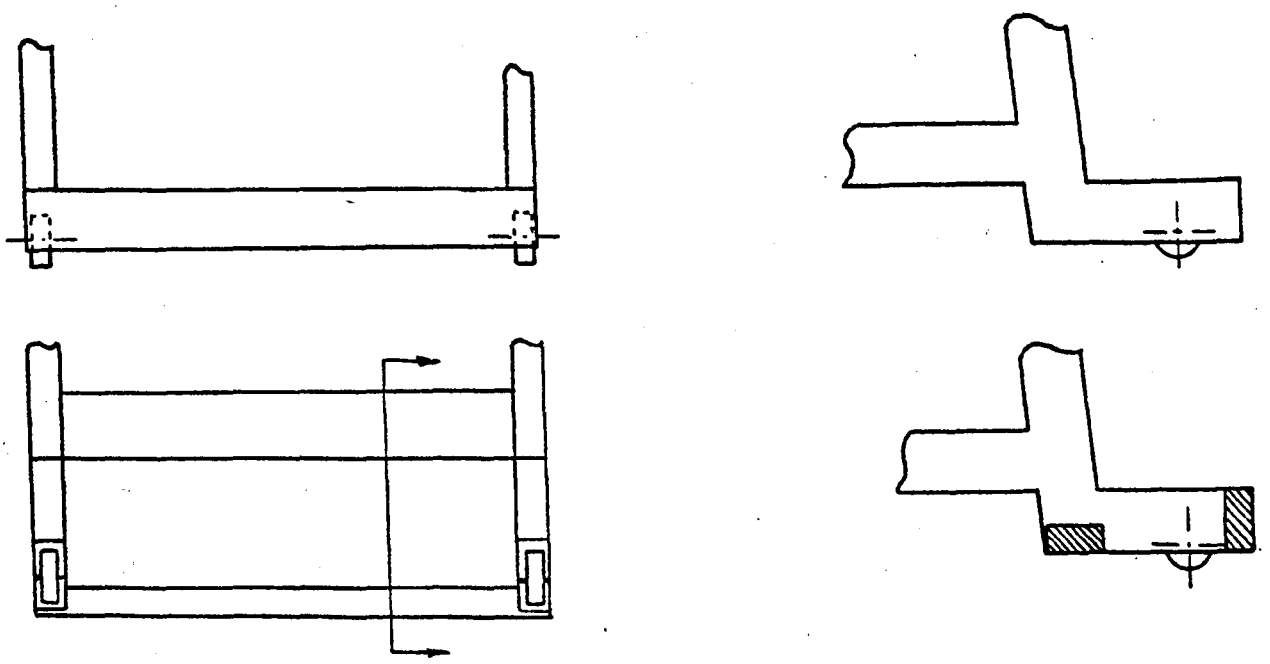
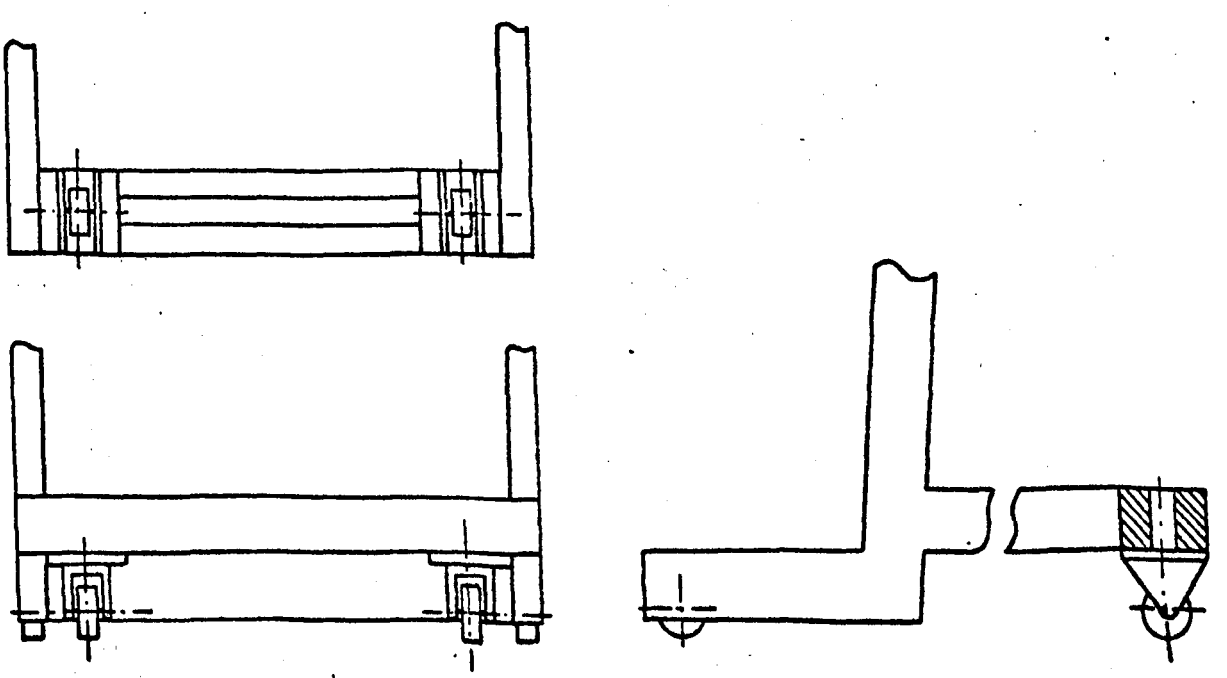


Figure 9.1 Mobile rack units - main dimensions (cont.).

Figure 9.2 Castor details.





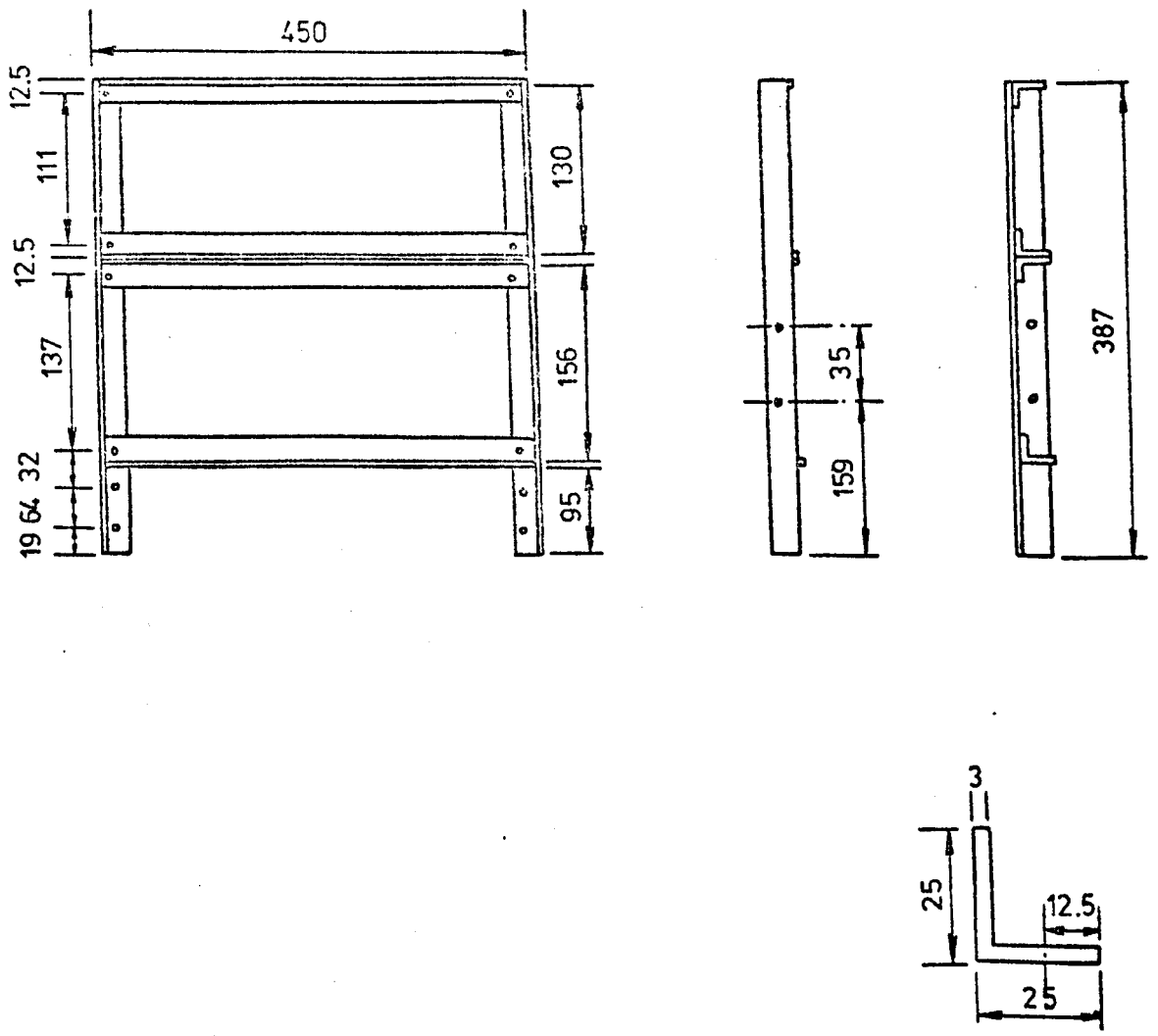


Figure 9.3 Power supply bracket - main dimensions.