

SHARP

SERVICE MANUAL



CE-150

Color graphic printer

**WWW.
PC-1500
.INFO**

SERVICE MANUAL

DPG 1301

COLOR GRAPHIC PRINTER

(CE-150 Printer)

TABLE OF CONTENTS

	Page
1. FEATURES AND OUTLINE	1
2. SPECIFICATIONS	1
3. MECHANISM AND OPERATION	4
4. DISASSEMBLY AND REASSEMBLY	8
5. REPAIR AND MAINTENANCE	16
6. OILING STANDARD	26
7. ADHESION STANDARD	27
8. CIRCUIT DIAGRAM & WIRING	28
9. PARTS GUIDE AND PARTS LIST	29

File this manual into the service manual "PC-1500 & Option"

The Color Graphic Printer DPG1301 is an X-Y plotter that uses ball-point pens as its writing instruments. Driven by stepping motors, it is capable of high accuracy plotting with a resolution of 0.2 mm and printing at a speed of 12 cps. The carriage has a four-pen capacity. Color selection is done by moving the carriage in an X axial direction, permitting random color graphic plotting. The printer has the following features:

- 1) 4-color graphic plotting
- 2) Compactness and light weight
- 3) High resolution of 0.2 mm x 216 steps
- 4) Printing of a maximum of 36 columns/line
- 5) Powered by a nickel-cadmium battery
- 6) 58 mm width plain paper can be used

1. FEATURES AND OUTLINE

The color graphic printer is an X-Y plotter that uses ball point pens.

It permits high speed printing of as many as 12 characters per second with high resolution power of 0.2mm as driven by the stepping motor.

Four pens of different colors are fitted on the carriage which permits you to draw graphics of different colors as the color is selected in moving on the X-axis.

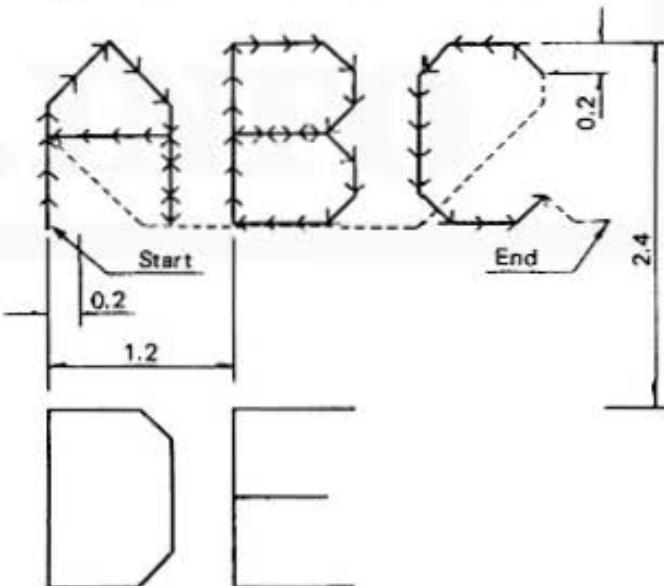
1. Four color graphic plot
2. Compact and light weight
3. High resolution of 0.2mm x 216 steps
4. Printing width of 36 columns/line at a maximum
5. Allows operation by means of NiCd battery
6. Use of ordinary paper of 58mm wide

2. SPECIFICATIONS

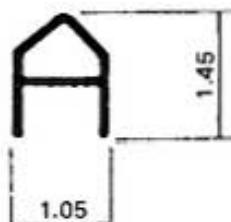
1. Model name: DPG 1301

2. Printing functions

Print method:	Ball point pen recording with four color rotary select system
Drive system:	Drum type X-Y plotter
Printing speed:	12 characters/second for specified characters (Reference)
Printing columns:	36 columns/line for specified characters 36, 18, 12, 9 columns selective
Stepping speed:	260 steps/second
Stepping distance:	0.2mm for X-axis and 0.2mm for Y-axis
Line drawing speed:	52mm/second (X and Y axis) 73mm/second for 45° direction
Character size:	One example of printed character



- 1) Character size: 1.05 x 1.45, for line width of 0.25
- 2) Character-to-character spacing: 1.2±10%
- 3) Line spacing: 2.4±10%



3. Effective range of plotting

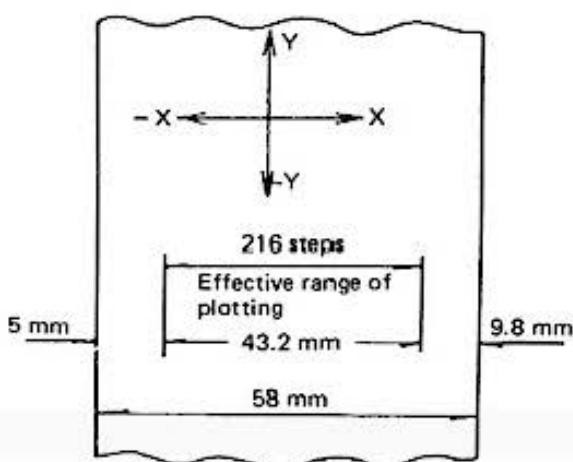
1) Plotting direction

In reference with the carriage moving direction, the rightward direction is determined to be X-axis(+) and the leftward direction is determined to be X-axis(-).

2) Effective range of plotting

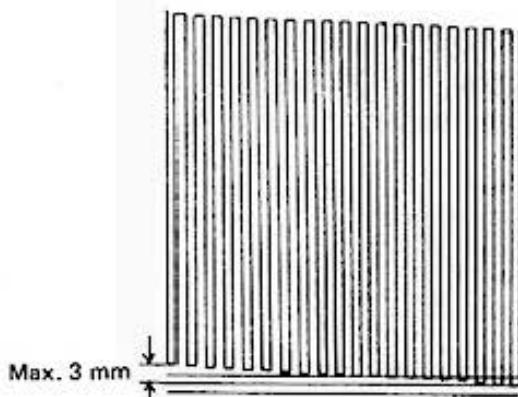
X-axis: 43.2mm, 216 steps

Y-axis: Any range as determined by software.



3) Accumulative error in Y-axis

As the paper is fed by means of friction with rubber for the Y-axis, there may arise a slight deviation, which should be within a range of $\pm 3\text{mm}$ as measured in the following manner, provided that specific paper guide is in use.



Example – Programme

```

10:GRAPH :
    GLCURSOR (8, 0)
20:FOR A=1 TO 20
30:RLINE -(0, -200
            )-(5, 0)-(0, 200
            )-(5, 0)
40:NEXT A
50:RLINE -(0, -185
            )--(-205, 0)-(0,
            -15)
60:RLINE -(210, 0)
            -(0, -15)--(-215
            , 0)
70:END

```

4. Recording paper and ball point pens

[Recording paper]

Kind: Ordinary paper

Dimensions: Paper width: 58^{+0}_{-1}mm ($2\frac{1}{4}''$)

Core size: 70mm ($2\frac{3}{4}''$), max.

Paper length: About 55m (for the core size of 70mm)

Recommended paper: High quality paper of about 45kg with thickness of 65 to 80 micron.

(1,000 sheets of 788 x 1,091 mm paper equals to 45kg.)

(Weight: equivalent to 52.3 g/m²)

[Ball point pen]

Color: Black, blue, red, green

Size: $5\phi \times 23.3$

Kind of ink: Water color

Life: 250m (820 ft.) or more

5. Electrical characteristics

5-1. Input/Output terminals

Name	No.	Circuit diagram
Color position sensor	B 1	
	A 2	
X motor	D 3	
	C 4	
	B 5	
	A 6	
Motor common	7	
Y motor	D 8	
	C 9	
	B 10	
	A 11	
Magnet (+)	B 12	
	(-) A 13	

Pen down upon current supply to the magnet in above polarity.

5-2. Magnet

1. Voltage: $4.85 \pm 0.65V$
2. Type: Self-holding magnet
3. DC resistance: $5\Omega \pm 10\% (20^\circ C)$
4. Peak current: About 1.1A ($20^\circ C, 4.85V$)
About 1.4A ($0^\circ C, 5.5V$, worst case)

5-3. Motor (260 steps/second)

	Item	X-axis	Y-axis	Condition
1	Voltage	$5.85V \pm 0.65V$		$0 \sim 50^\circ C$
2	Type	4-phase stepping motor (2 phase excitation)		
3	DC resistance	(A1) $30\Omega \pm 10\%$	$25\Omega \pm 10\%$	$20^\circ C$ (resistance per phase)
4	Peak current (per phase)	(A2) Abt. 0.16A (A3) Abt. 0.23A	Abt. 0.19A Abt. 0.27A	$20^\circ C, 4.85V$ $0^\circ C, 5.5V$, (worst case)
	Average current (per phase)	(A4) Abt. 0.12A (A5) Abt. 0.16A	Abt. 0.13A Abt. 0.18A	$20^\circ C, 4.85V$ $0^\circ C, 5.5V$, (worst case)

5-4. Power consumption

Print pattern	Scale	Voltage	Current consptn (mA)	Power consptn (W)
ACII 64 character set	S=0	4.8V	500 ~ 550	2.4 ~ 2.6
ACII 64 character set (excluding CR)	S=1	4.8V	400 ~ 450	1.9 ~ 2.2
ACII 64 character set	S=1	4.2V	340 ~ 370	1.4 ~ 1.6
ACII 64 character set	S=1	5.8V	500 ~ 580	2.9 ~ 3.4
"S" printed in 5 columns	S=1	4.8V	385	1.8
Paper feed action	—	4.8V	260	1.2
X-axis forward and backward	—	4.8V	180	0.9
45° line drawing	(L=0)	4.8V	490	2.4
45° dot drawing	(L=1)	4.8V	790	3.8

5-5. Color position sensing switch

1. Operating voltage: DC 24V, max.
2. Operating current: 100mA, max.
3. Contact resistance: 150mΩ, max.

6. Durability

No.	Item and test method	Test item	Specification
1	Life 6.5 million characters ASCII 64 character set are continuously printed in the minimum scale (S=0). At any time during the test (ie. 1, 2, 4, and 6.5 million characters), appearance, operating conditions, and print quality are tested.	1. Appearance 2. Print quality	Life: 6.5 million characters Must be good.
2	Pen life Continuous operational test is carried out in the print mode shown in Attached Drawing 4-1, with a new pen in use.	1. Ink life	Must be able to draw more than 250 meters.

3. MECHANISM AND OPERATION

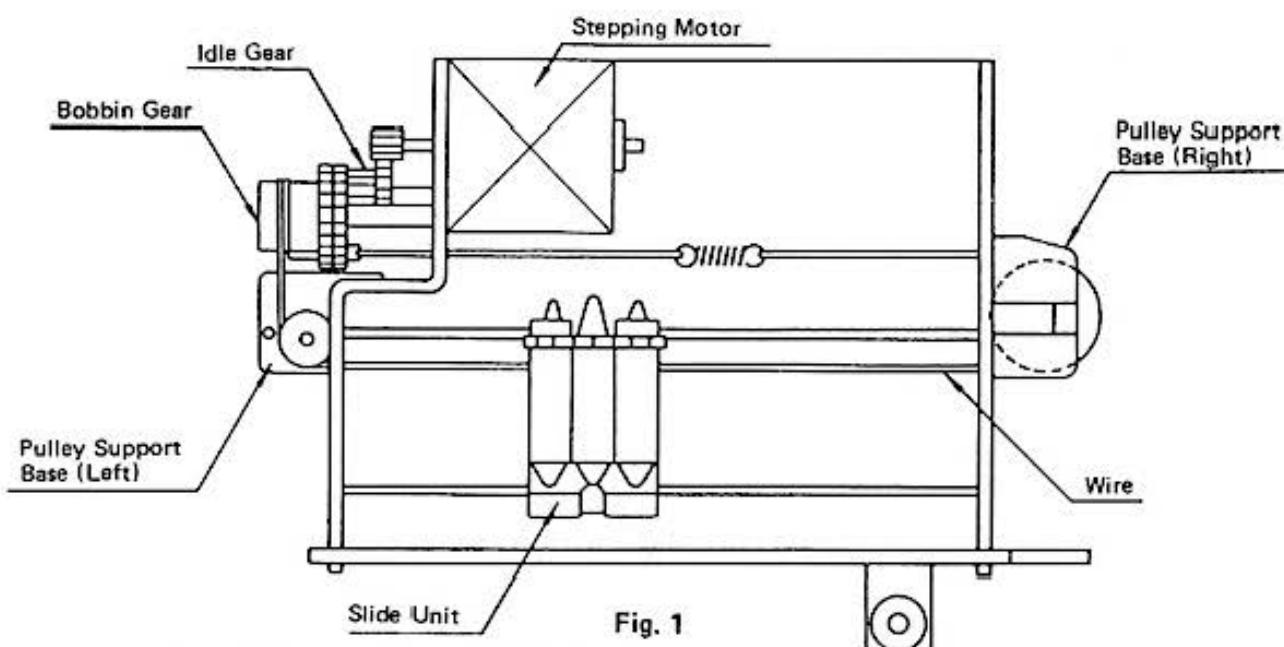
The printer roughly consists of six blocks — a frame, X-direction drive, Y-direction drive, pen drive mechanism, color change mechanism, and pen take-out mechanism sections. An explanation of these blocks will be given below.

1. Frame Section

The frame section has a side plate (right), side plate (left), holding plate, and paper guide. The various mechanisms are mounted both inside and outside. The lower edge of the frame is bent in the shape of the letter L and acts as a mounting leg.

2. X Drive Mechanism Section

The principal elements of the X-direction drive mechanism are the X stepping motor, idle gear, bobbin gear, pulley support base (left), pulley support base (right), slider unit, wire, etc.

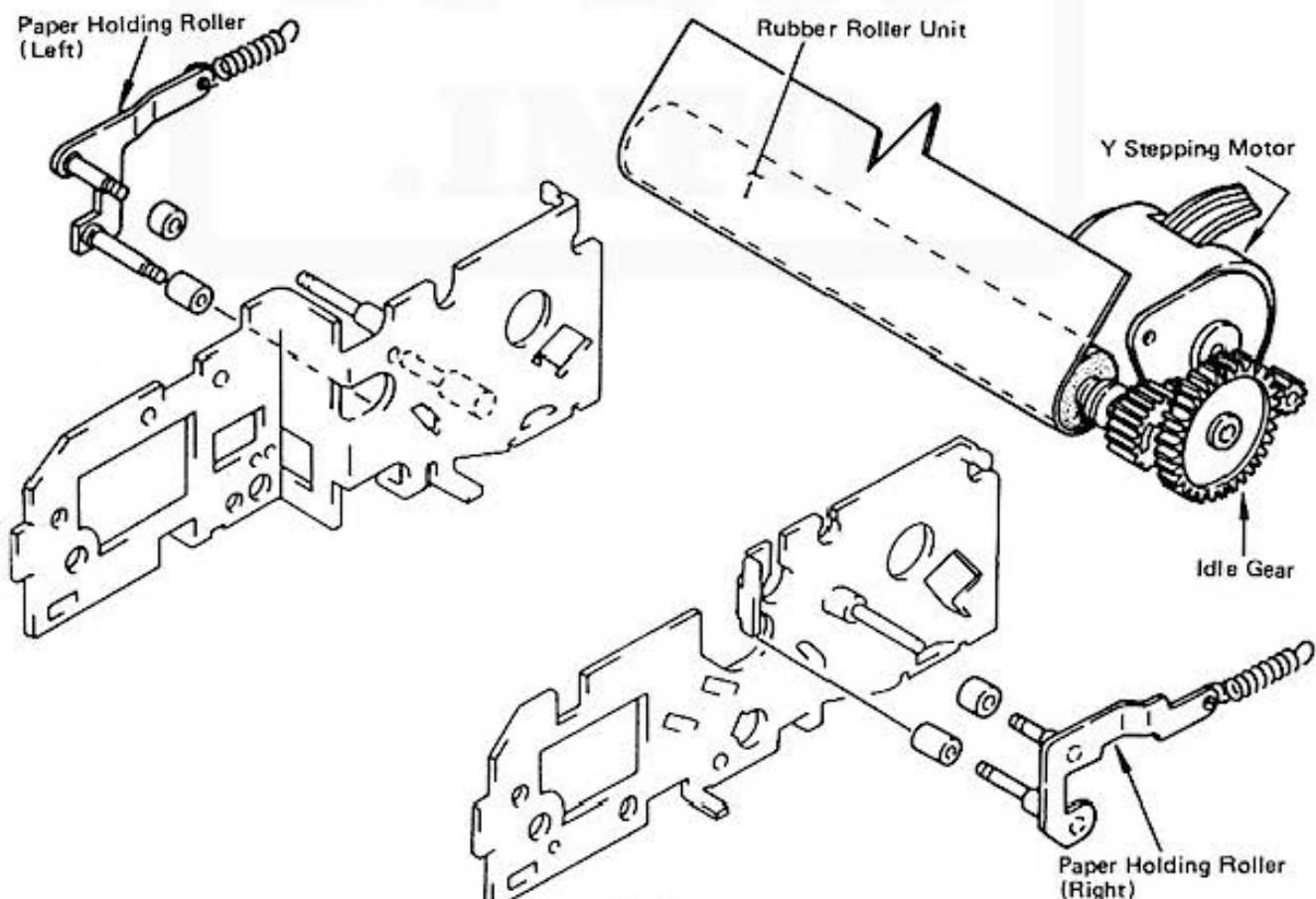


- Step Angle and Minimum Movement Pitch

The reduction ratio between the stepping motor and bobbin gear is 1:9.01, and for each stepping motor pulse ($18^\circ/360$), the slider unit, that is, pen X direction movement, is 0.2mm. The motive power is transmitted to the bobbin gear and slider unit by a wire, and the wire tension is maintained by a coil spring.

3. Y Drive Mechanism (Paper Feed Mechanism) Section

The Y-direction drive mechanism consists of the Y stepping motor, idle gear, rubber roller unit, paper holding roller (right) and paper holding roller (left). The reduction ratio between the Y stepping motor (called the Y motor below) and rubber roller gear is 1:7.86. As is the case with the X direction, the rubber roller movement per pulse of the Y motor, that is, the Y direction movement quantity of the recording paper, is 0.2mm.



4. Pen Drive Mechanism Section

The pen drive mechanism, namely the pen up-down mechanism, comprises a self-holding type electromagnet, ejection lever, roller lever, and ball-point pens. Pen up and down directions are as shown below.

- **Pen-up State**

The pen retracts when the electromagnet is energized for 5 ms against the actuator spring, and held by a permanent magnet even after the current is cut off after the initial 5 ms.

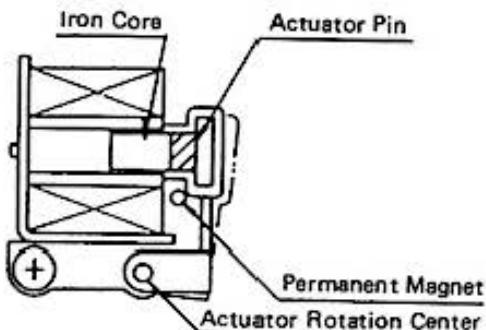


Fig. 3

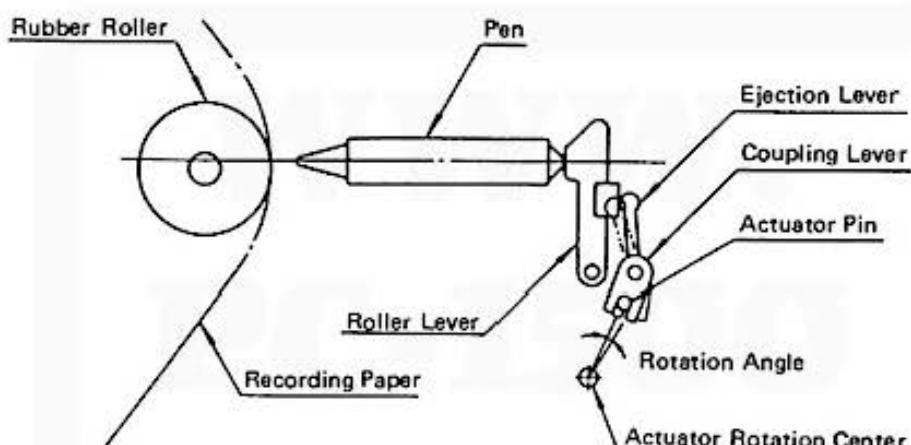


Fig. 4

- **Pen-down State**

From its pen-up state the pen descends when a current is impressed to the electromagnet for 5 ms in an opposite direction to that used for pen retraction. After 5 ms, the pen-down state will be maintained by the actuator spring force.

5. Color Change Mechanism Section

The color change mechanism section consists of the X-direction drive mechanism, a pen holder and holder stopper, both in the slider unit, and projections inside the holding plates. The operating principles are explained below.

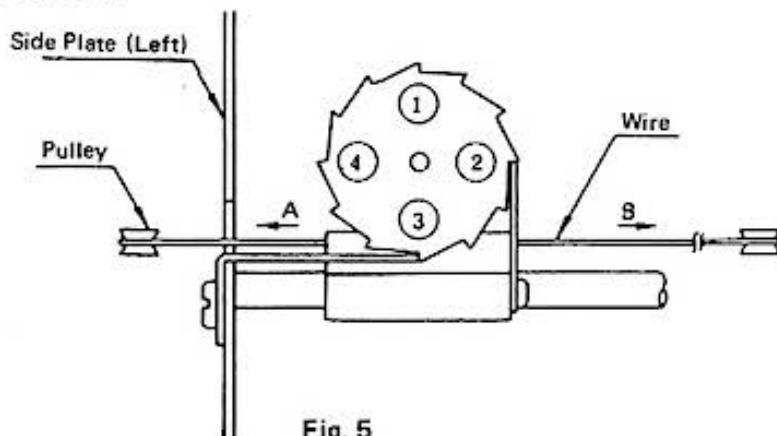


Fig. 5

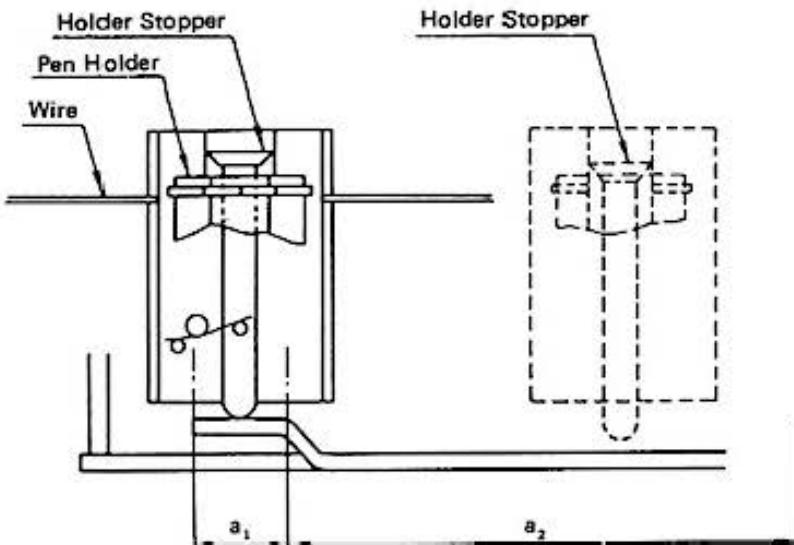


Fig. 5

To operate the color change mechanism, first, the slider is moved to the area $a \cdot 1$ in Figure 6 (45 pulses to the left from the origin). Then, the holder stopper in the slider contacts the projection on the holding plate, and the wedge section of the holder stopper slips out of the pen holder, which frees it to rotate. Next, by repeating the movement of the X motor for 30 pulses each in direction A and to the left, the pen (1) in Figure 5 changes to pen (2). The spring moves the holder stopper to the right until it returns to the origin. If then enters the pen holder groove. The pen holder is then fixed, and printing is ready.

6. Pen Ejection Mechanism

The pen ejection mechanism consists only of the pen ejection lever that is mounted on the side plate (right). The slider unit is moved fully to the side plate (right) and is stopped in order to change the pen. Push the pen ejection lever towards you, and the pen will eject.

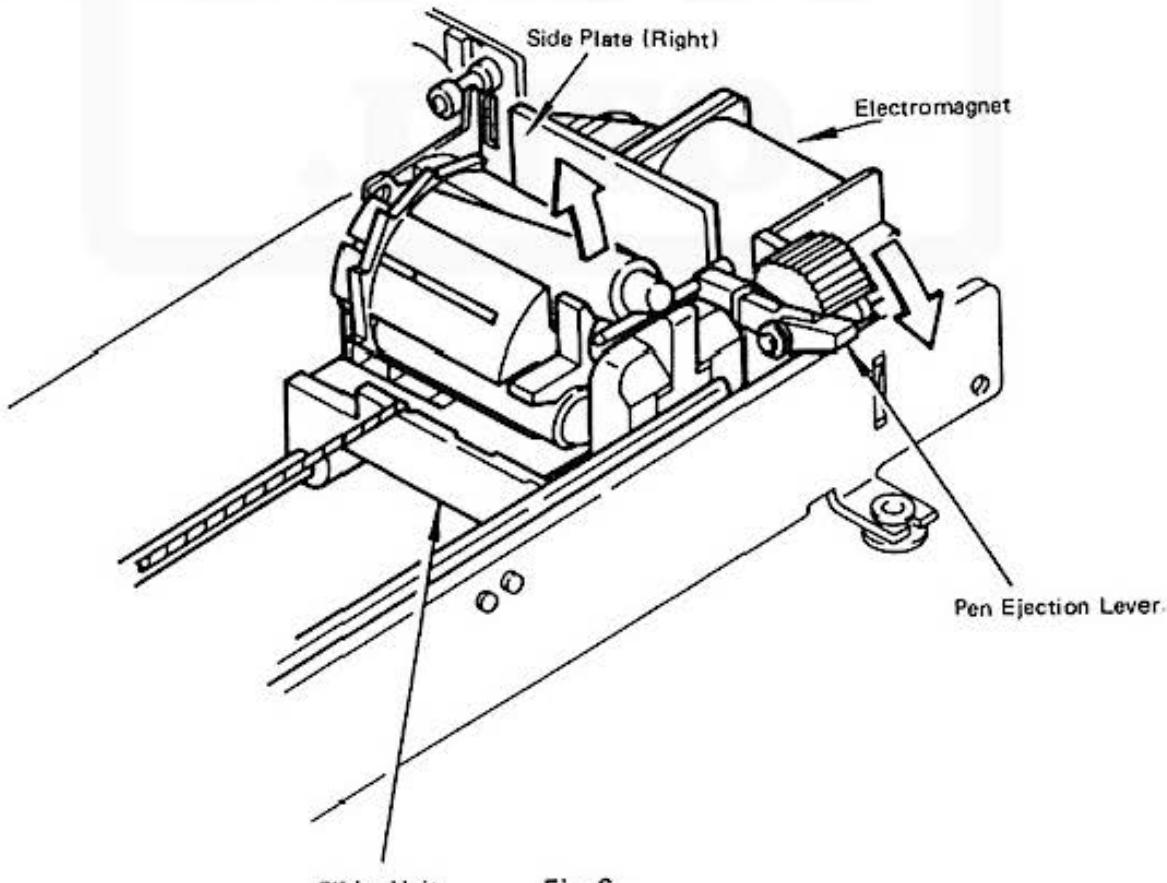
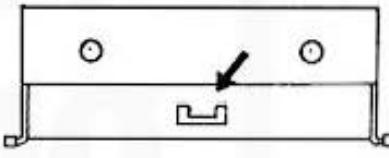


Fig. 6

4. DISASSEMBLY AND REASSEMBLY

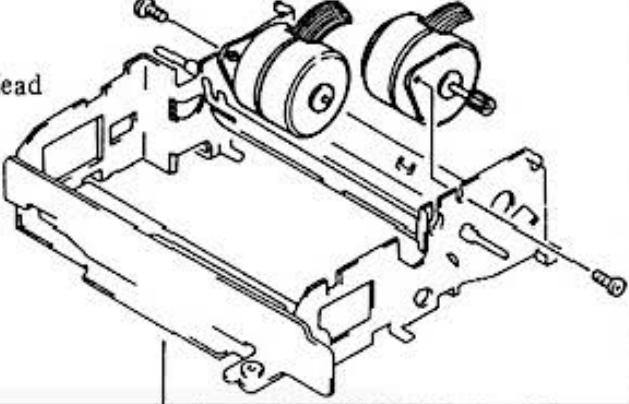
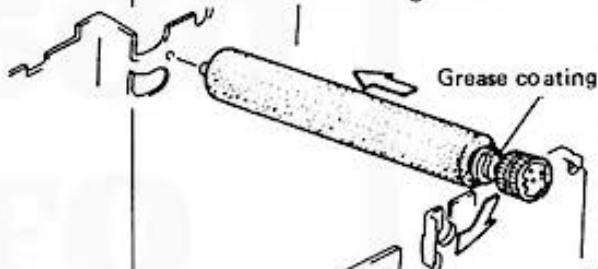
1. Disassembly

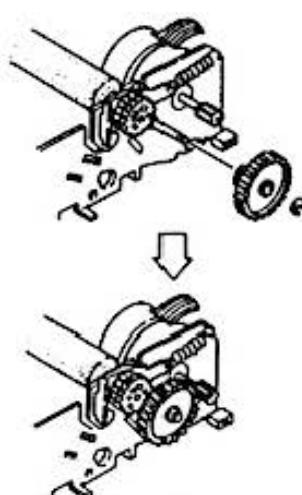
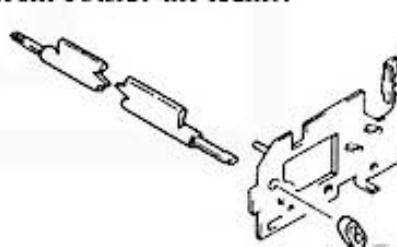
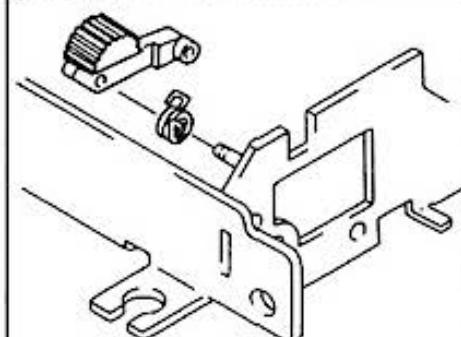
Remove the following parts from the frame in the sequence shown below.

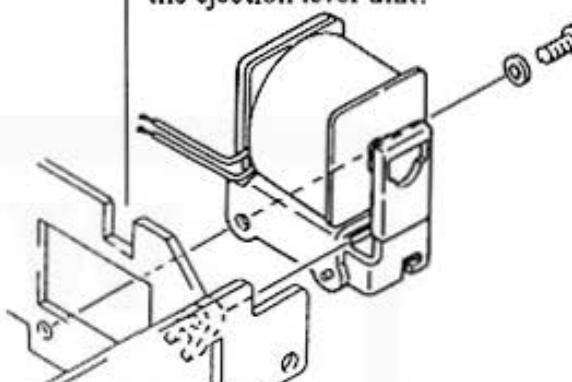
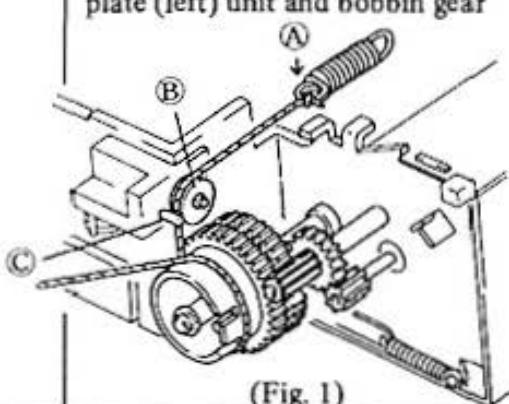
Disassembly Sequence	Part No.	Part to be Removed	Point for Disassembly
1	4-1 2-4 6-1 2-7 3-2 3-5	Electromagnet Unit Wire Unit Pen Take-out Bar Unit Pulley Support Base (Right) Unit Y Idle Gear Paper Holding Roller Support Plate (Right) Unit	<ul style="list-style-type: none"> • Disassemble after removing cross-recessed pan head machine screws (SP2 x 3) and (SP2.3 x 3), and sleeves (2-5). • Note – brass fittings may slide off. (Don't lose)
2	7-1	Motor Cover	<ul style="list-style-type: none"> • Lift up the motor cover (7-1) section covering the cross-recessed pan head machine screws (SP2.3 x 3) which hold the motor. • Remove the entire motor cover (7-1) by inserting a flat-blade screwdriver in the paper guide as shown in the diagram below. 
3	3-3	Rubber Roller Unit	<ul style="list-style-type: none"> • Push the rubber roller unit bearing to the left and remove from the right side of the rubber roller unit (3-3) as it comes off the frame unit (1-1).
4	3-1 7-4 2-3 2-2 3-4	Y Motor Unit Flat Wafer Assembly Bobbin Unit X Idle Unit Paper Holding Roller Support Plate (Left) Unit	<ul style="list-style-type: none"> • Suction solder in the junction section of the two printed circuit boards. • Note – brass fittings may slide off. (Don't lose)
5	2-1 2-6 2-8 2-9 4-5 4-2 4-3 4-7 4-6 7-2 7-3	X Motor Unit Pulley Support Base (Left) Unit Slider Shaft (A) Slider Shaft (B) Slider Unit Ejection Lever Shaft Unit Ejection Lever Color Change Reed Switch Unit Rubber Bushing Rubber Pad	<ul style="list-style-type: none"> • Push in plastic tabs from inside of frame.

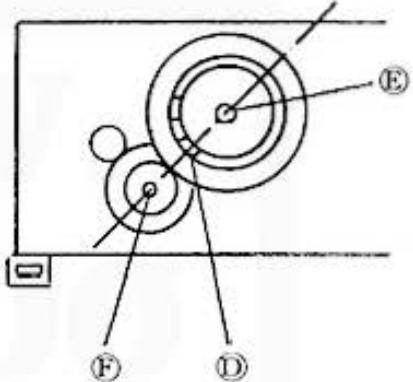
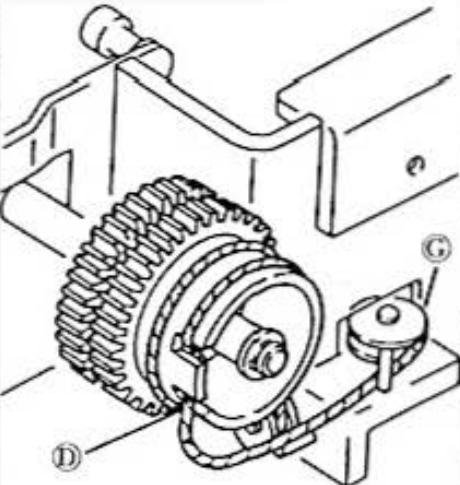
2. Reassembly

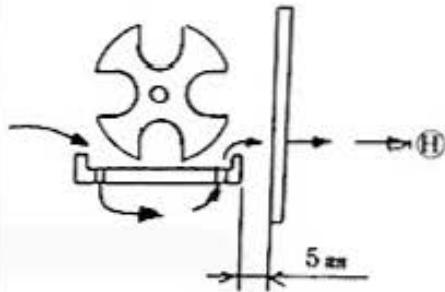
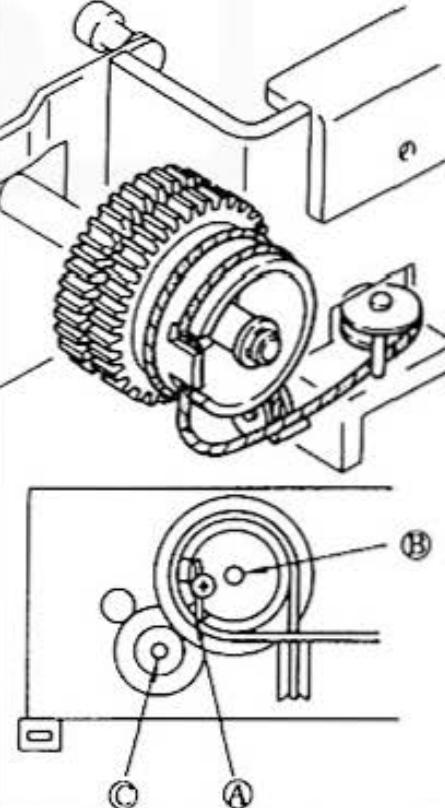
Reassembly can be completed most efficiently by referring to the reassembly sequence and precautions shown below.

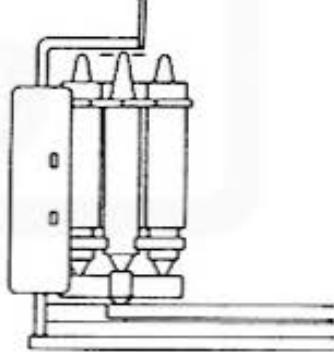
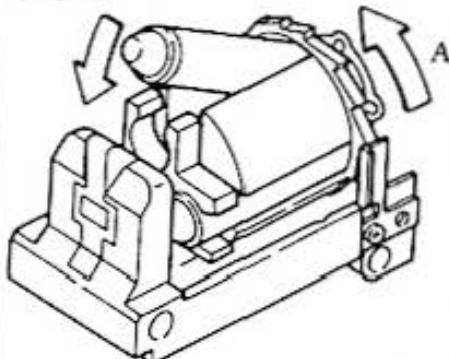
Reassembly Sequence	Parts Symbol	Parts to be Reassembled	Precautions for Reassembly
1	3-1 SP2.5 x 3 2-1 SP2.5 x 3	X Motor Unit Cross-recessed Pan Head Machine Screws Y Motor Unit Cross-recessed Pan Head Machine Screws	Screws may be coated with lock Paint. 
2	7-3 7-2 2-6 3-3 WF1.7 3-5	Rubber Pad Rubber Bushing Paper Holding Roller Support Rubber Roller Unit Plain Washer Paper Holding Roller Plate (Right) Unit	Fit the end of the rubber roller unit into the left side and push the bearing to the left side to fit the other end into the right side. 
3	2-2 RE1.5 2-3	X Idle Gear Type E Stopper Ring Bobbin Gear Unit	Move the bobbin gear by one tooth and insert the X idle gear. Recommend to engage after marking the tooth-tip and moving the gear by one tooth.

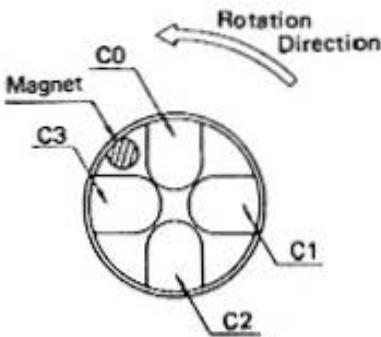
Reassembly Sequence	Parts Symbol	Parts to be Reassembled	Precautions for Reassembly
4	3-2 RE1.5	Y Idle Gear Type E Stopper Ring	<p>Insert the Y idle gear after moving the paper feed gear by one tooth.</p>  <p>Recommend to engage after marking the tooth tip and moving the gear by one tooth.</p>
5	7-1	Motor Cover	<p>Hang the motor cover by its square hole on the hook on the paper guide B on the rear of the frame unit.</p>
6	4-2 4-3 RE2	Ejection Lever Shaft Unit Ejection Lever Type E Stopper Ring	<p>Insert the ejection lever shaft unit into the frame through a bearing hole and press in the coupling lever from outside the frame.</p> 
7	6-1 6-2 RE1.2	Pen Take-out Lever Unit Pen Take-out Lever Spring Type E Stopper Ring	
8	4-7 SP1.4 x 1.6	Color Change bar. Cross-recessed Pan Head Machine Screws	Screws may be coated with lock paint.

Reassembly Sequence	Parts Symbol	Parts to be Reassembled	Precautions for Reassembly
9	4-5 2-8 2-9 RE-2	Slider Unit Slider Shaft (A) Slider Shaft (B) Type E Stop Ring	Never bring another magnet close to the slider unit magnet. If you do the magnet inside the slider unit demagnetizes causing the color detection switch to operate incorrectly.
10	2-6	Pulley Support Base (Left) Unit	
11	4-1 SP2.5 x 3 WT2.5 7-4	Electromagnet Unit Cross-recessed Pan Head Machine Screws Lock Washer Flat Wafer Assembly	<p>Hang the electromagnet unit actuator on the coupling lever on the ejection lever unit.</p>  <p>Mount so that open strokes are 0.6 mm. Clamping torque, 3.5 kg-cm. Screw lock paint coating.</p>
12	4-6 SP2 x 3	Reed Switch Unit Cross-recessed Pan Head Machine Screws	<p>Screw lock paint coating. Rotate the bobbin gear. The reed switch must actuate when the magnet at the left edge of the slider approaches the closest reed switch.</p>
13	2-7	Pulley Support Base (Right) Unit	
14	2-4 2-5	Wire Unit Sleeve	<p>Stretch wire on the pulley support plate (left) unit and bobbin gear</p>  <p>(Fig. 1)</p>

Reassembly Sequence	Parts Symbol	Parts to be Reassembled	Precautions for Reassembly
			<p>The wire must be stretched in the order described below:</p> <ol style="list-style-type: none"> 1) As shown in Fig. 1, the spring must be pressed against the side plate using a finger tip with the knot of the wire in alignment with the side plate at point (A). 2) Next, thread the leading edge of the wire through the pulley (B) and the protrusion (C). 3) As shown in Fig. 2, align the slit (D) of the bobbin gear so that it should on a line drawn between the bobbin gear shaft (E) and the idler gear shaft (F).  <p>Fig. 2</p> <ol style="list-style-type: none"> 4) Wind the wire around the bobbin gear for one and half a turn, then thread it through slits.  <p>Fig. 3</p>

Reassembly Sequence	Parts Symbol	Parts to be Reassembly	Precautions for Reassembly
			<p>5) Wind the wire around the pulley (G), then thread through the hole in the side plate.</p> <p>6) Now, set the slider unit to a distance of 5mm from the right (magnet side), then thread the wire through the hole in the slider unit as shown in Fig. 4.</p>  <p>Fig. 4 Hold the wire at the point (A) and stretch the wire towards the direction (H), then adjust the location of the slider unit so that the distance between the side plate and the slider unit should become about 5mm.</p> 

Reassembly Sequence	Parts Symbol	Parts to be Reassembled	Precautions for Reassembly
			<p>7) Apply the wire to the pulley support bracket (right) (2-7), then thread the wire through the hole in the side plate, again.</p> <p>8) Next, insert the wire into the sleeve, thread through the spring hook loop, then thread the wire through the sleeve.</p> <p>9) Pull the one of the wire (i) until the tension of 160 grams is on the spring (about 2mm of spring elongation), then set the sleeve using the long nose pliers.</p> <p>10) Check a proper movement of the slider.</p>
SP2 x 3		Phillips head, small, pan head screw	Securing of the bobbin gear with the wire.
SP2.3 x 3		Cross-recessed Pan Head Machine Screw	Adjust the relative positions of the wire and slider so that the drawing line in the X-direction will be at the center of the paper guide. Screw lock paint coating.
5-1 5-2 5-3 5-4		Ball-point Pen (Black) Ball-point Pen (Blue) Ball-point Pen (Green) Ball-point Pen (Red)	Move the slider to the left edge.  Insert the pen tip at the tip of the pen return spring and push the rear section. 

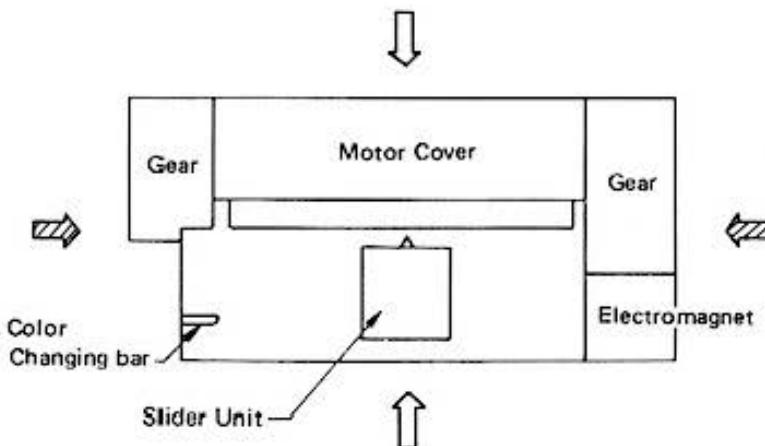
Reassembly Sequence	Parts Symbol	Parts to be Reassembled	Precautions for Reassembly
			<p>Rotate the rotary holder in the direction of arrow A and insert the pen. Mount the pen color position making the magnet for the reed switch as the reference.</p> 

5. REPAIR AND MAINTENANCE

1. Handling Precautions

1) Carrying Printer

- (1) Carry the printer by holding it in the directions shown by . Carrying the printer in the direction will cause various troubles.
- (2) The printer may be carried by holding on to the upper face of the motor cover and paper guide. However, do not apply strong pressure to it.



2) Sections where Pressure Should Not be Applied:

- (1) Do not touch the slider unit except when taking a pen out. Particularly, never apply pressure in the direction of rotation.
- (2) Do not touch the wire. The pulley may come off.
- (3) Do not touch the color changing bar. When bent, color changing cannot be accomplished.

3) Sections Not to be Touched

- (1) No shaft should be touched with bare hands.
- (2) Do not touch the pen return spring.
- (3) Do not touch the rotary holder except when the slider is positioned at the left edge of the frame and a pen is mounted.
- (4) Do not touch the slider.

4) Sections Where Magnetic Substances Should be Kept Away

- (1) Do not place a magnetic substance or powder, a permanent magnet, or an electromagnet close to the permanent magnet of the color position detector.
- (2) A strong rare earth magnet is used in the electromagnet unit.

5) Other;

Be very careful not to drop the pen or in any way bend it. When the ink is exhausted, hold by the tail plug section and shake it.

2. Maintenance

Perform repairs and maintenance as shown below to maintain the initial printer quality as long as possible.

1) Cleaning

Clean the printer and remove paper dust, dirt, etc. periodically (about every three months or after using 5 rolls of recording paper).

(Points for Cleaning)

- (1) Paper dust, dirt, etc. should be vacuumed up.
(Use an electric vacuum cleaner).
- (2) Use alcohol or benzene when removing stains. Thinner, trichloroethylene and ketone solvent may damage the plastic parts and should not be used.
- (3) Grease any places where there is no grease or where it is not sufficient. (Do not grease unless greasing is required. Do not use a lubricant except that which is specified. Refer to Item 6. Oiling Standard.)

2) Recording Paper

- (1) Use recording paper recommended in Paragraph 2-4 of the specifications.

3. Repairs

The printer has three repair levels (A, B, and C) taking into account the difficulty of after sale services and repairs. Persons in charge of repairs are asked to make repairs after comparing their own technical proficiencies and the required repair level.

1) Repair Levels

Level A: A general knowledge of the principles and construction of the printer and some degree of skill are required. Experience or a high degree of skill is not required.

Level B: A good knowledge of the principles and construction of the printer and the skill to disassemble the printer and to use measuring instruments, jigs, and tools associated with disassembly and reassembly are required. This level requires experience in making repairs.

Level C: A high degree of special knowledge regarding the principles and construction of the printer are required along with the ability to use measuring instruments, jigs, and tools associated with the printer. In-depth experience and skill in repair work are required.

2) Repair Procedure

When a fault occurs, carefully observe and check the type of failure and the conditions surrounding it. Find out the cause and make repairs after checking the location of the fault, referring to the "Repair Guide."

- (1) "Phenomenon": Determine the trouble phenomenon from this column.
- (2) "Condition": Compare the trouble with this column and verify whether it coincides.
- (3) "Cause": Causes based on the condition of the trouble are listed. Verify the cause.
Use the repair level described above when making repairs after fully considering the repair level.
- (4) "Locations and Methods of Checking":
The column lists where to check for trouble and by what method. Check according to the instructions in this column and locate the trouble.
- (5) "Repair Method": Repair the trouble according to the instructions described in this column.
If the same phenomenon or conditions exist after making repairs, check the other items in the cause column and make necessary repairs.

3) Repair Tools

- Screwdrivers (Precision Screwdrivers)
Phillips Type 4, equivalent to No. 000.
Phillips Type 5, equivalent to No. 00.
Flat-blade Type 5
- ET Holders E-Ring
ET 2
ET 1.5
ET 1.2
- Radio pliers, or reed pliers
- Tweezers
- Soldering iron

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
1. Does not draw lines	X-motor (2-1), Y-motor (3-1), and pen drive electromagnet (4-1) operate normally, but no printing is done.	Pens (5-1 to 4) have come off, or the ink is exhausted.	A	Are the pens mounted properly? Do the pens have enough ink?	Mount properly. Replace the pens.
2. Lateral lines cannot be drawn	(1) X-motor (2-1) does not rotate. (2) Abnormality is noticed regardless of whether the pen is moving up or down.	(1) X-motor lead wire is cut. (2) Idle gear (2-2) is deformed. (3) Deformation of bobbin gear unit (2-3), mis-alignment of two-piece teeth. (4) Foreign matter has accumulated between gears. (5) Low battery voltage. (6) Slider unit (4-5) does not slide properly on shaft. (1) Foreign matter has accumulated in the moving section of the slider unit. (2) Contact between pen take-out lever unit (6-1) and pen.	B B B A A B A B	Check that a normal current is impressed to each phase of the motor. Check if the X idle gear is normal. Dismount wire unit (2-4), rotate bobbin gear by hand, and check rotation state. Rotate bobbin and unit by hand and check for foreign matter. Check if battery voltage is below 4.5V. Dismount wire and move slider unit to the right and left by hand. Check that slider unit moves smoothly on the effective printing area and check for an abnormal load by slowly rotating the bobbin gear by hand. Check contact between pen take-out lever and slider unit.	Replace X-motor. Replace the X idle gear. Replace bobbin gear unit. Remove foreign matter. Recharge to regular voltage. • Replace slide unit. • Remove foreign matter if it is obstructing movement. Remove any foreign matter Replace pen take-out lever.

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
	(2) X-motor rotates, but slider unit does not move to the right or left.	(3) Wire fastening position on bobbin has moved. (4) Improper pulley rotation of pulley support base unit(right). (5) Sleeve (2-5) and frame make contact. (6) X motor unit (2-1) is operating improperly. (1) Wire has come off (2) Wire is cut. (3) Bobbin gear and wire slip. (4) Gear is damaged	B B B B A B A A	Rotate bobbin gear unit (2-3) by hand and check that slider unit moves smoothly from left edge to right edge. Dismount wire from pulley and check for smooth rotation. Check for deformation between frame unit and sleeve, as well as for other phenomena. Dismount X idle gear (2-2), slowly rotate motor gear, and check for abnormal load. Check if wire has come off the bobbin gear. Check that wire has not been cut. Confirm that wire is properly screwed on to bobbin gear. Check that X idle gear and bobbin gear are operating properly.	Restretch wire if its fastening position is wrong. Replace pulley support base unit. Repalce sleeve. Replace motor. Restretch wire properly. Replace wire unit. • Tighten the screws. • Replace the bobbin gear unit. • Replace damaged gear.
3. Insufficient Lateral Movement	(1) Operates in pen-down mode only, and not normal.	(1) Rubber roller unit (3-3) is scratched. (2) Print paper has irregularities. (3) Deformation of ejection lever shaft unit (4-2).	C A B	Slowly rotate the rubber roller by hand and check for scratches and foreign matter. Check the surface of the print paper. Check the contact between the ejection lever and ejection roller (4-4).	• Replace rubber roller unit if scratches are found. • Remove foreign matter. Use normal print paper. • Replace ejection lever shaft unit.

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
4. Drawing is done segment by segment	(1) Actuator of the electromagnet unit (4-1) is detached from the electromagnet coil.	(4) Low battery voltage (5) Contact between set-screw of return spring and small roller.	A	Check that battery voltage is not below 4.5V. Move slider and confirm contact.	Recharge to normal voltage. • Replace rotary holder. • Replace paper holding roller support plate unit.

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
	(2) Electromagnet operates normally	(1) Pen movement is slow. (2) Deformation and fatigue of pen return spring (4-5-3). (3) Paper is not winding on rubber roller properly. (4) Pen is too long.	B B A A	Check the shape of the pen Also check for rotary holder deformation and the presence of foreign matter. Inspect the pen return spring. Remove paper and check for paper guide deformation, Measure the pen length.	<ul style="list-style-type: none"> • Pen exchange. • Rotary holder exchange. • Removal of foreign matter. Repalce rotary holder set (4-5-2). <ul style="list-style-type: none"> • Return it to its normal shape using tweezers, etc. if it is only slightly deformed. • Remount properly if small roller which holds the paper has come off. • Mount a proper pen (length 23.3^{+0}_{-1}mm)
5. Color does not Change	(1) Pen moves to color change area without moving up. (2) Carriage does not move until it reaches the left edge. (3) Rotary holder does not rotate at all.	(1) Electromagnet induced pen up function is ab-normal. (1) Foreign matter has accumulated in slider section. (2) Contact between slide shaft support plate (4-5-5) and frame. (1) Fatigue and deformation of color change bar. (4-7). (2) Pen tip has come off the return spring.	B A B B	Check transmission system from electromagnet to pen drive. Check for foreign matter. Check if slider moves smoothly by rotating the bobbin gear by hand. Check for contact. Check color change bar. Check all four pens to see if they have come off.	See Phenomenon 4. Remove foreign matter and exchange slider unit. Replace slider unit. <ul style="list-style-type: none"> • Gently lift color change bar using tweezers. • Replace color change click. • Repair using tweezers. • Replace rotary holder if return spring is deformed.

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
		(3) Pen return spring is deformed. (2) Reed switch is faulty	B	Check the shape of the pen return spring.	• Replace rotary holder.
		(3) Rotary holder rotations are short by one	B	Check if reed switch actuates when magnetic flux is normal.	Replace reed switch unit (4-6).
		(1) Discrepancy in mounting of the reed switch unit. (2) Excessive rotation of rotary holder.	A	Check position of reed switch unit. Check if rotary holder excessively due to foreign matter, etc.	Replace reed switch unit. Remove foreign matter, etc.
7. Paper is not fed.	(1) Y motor (3-1) does not rotate.	(1) Y motor lead wire is cut. (2) Deformation of Y idle gear (3-2). (3) Rubber roller unit (3-3) does not rotate well. (4) Foreign matter between gears. (5) Low battery voltage. (6) Paper holding roller support plate unit (left) (3-4) does not operate smoothly. (7) Paper holding roller support plate unit (right) (3-5) does not function well.	B B B A A B B	Check if normal current is impressed to each phase of the motor. Check if Y idle gear is normal or not. Dismount Y idle gear and check rubber roller rotations. Caution rotations are heavy due to friction between rubber roller and paper guide when paper is not inserted. Slowly rotate Y idle gear by hand and check for foreign matter. Check if battery voltage is below 4.5V. Hook tweezer tips in hole of spring hook on paper holding roller support plate unit and move it up and down. Hook tweezer tips in hole of spring hook on paper holding roller support plate unit and move it up and down.	Replace X motor. Replace Y idle gear. Replace rubber roller unit. Remove foreign matter. Recharge to normal voltage. Replace paper holding roller support plate unit (left). Replace paper holding roller support plate unit (right).

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
8. Y-direction movement is insufficient.	<p>(2) Paper and rubber roller slip.</p> <p>(1) Character alignment on one line is bad, and the line rises at the right end.</p> <p>(2) Stepping error in Y-direction.</p>	<p>(1) Damage to paper holding roller (large) (3-6) and paper holding roller (small) (3-7).</p> <p>(2) Deformation of paper guide.</p> <p>(3) Foreign matter in paper guide.</p> <p>(1) Roll paper load is too heavy.</p> <p>(1) Y drive mechanism gear is damaged.</p> <p>(2) Sliding paper feed gear, by one tooth, gearing of rubber roller unit is not enough.</p> <p>(3) Rubber roller unit bearing is worn.</p> <p>(4) Low battery voltage.</p>	<p>A</p> <p>C</p> <p>A</p> <p>A</p> <p>B</p> <p>A</p> <p>B</p> <p>A</p>	<p>Check if paper holding roller is there.</p> <p>Check for paper guide deformation.</p> <p>Check for foreign matter in paper guide and for insertion of paper.</p> <p>Check that roll paper is guided smoothly into the printer.</p> <p>Check Y idle gear (3-2), rubber roller unit (3-3) gear, and Y motor unit (3-1) gear.</p> <p>Check that the two-piece tooth gear on rubber roller unit is engaging after being slid by one tooth.</p> <p>Move rubber roller unit gear up and down by hand and check for play.</p> <p>Check if battery voltage is below 4.5V.</p>	<p>Replace damaged roller.</p> <p>Replace paper guide.</p> <p>Remove foreign matter.</p> <p>Repair roll paper guide.</p> <p>Replace gears.</p> <p>Mount after setting it properly.</p> <ul style="list-style-type: none"> Replace rubber roller unit if wear is noticed. Fix by using a cyanoacrylate adhesive when there is play between the bearing and frame. Replace printer. <p>Recharge to regular voltage.</p>

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
	(3) Origin position differs after making many movements in Y direction. Origin position changes after printing a large number of characters.	(1) Rubber roller and paper slip. (2) Deformation of paper guide. (3) Roll paper is guided improperly. (4) Paper type does not match printer.	A A A	Check for stained rubber roller. Check paper guide. Check roll paper rotation and ensure that the center of the paper and the center of the printer are aligned. Check that the specified paper is used.	Wipe off rubber roller stain. • Repair paper guide if there is any deformation. • Replace printer. Repair roll paper guide. Use the specified paper.
9. Character misalignment is substantial.	(1) "F" is drawn as shown below. 	(1) Improper engagement of bobbin gear unit (2-3) after sliding by one tooth. (2) Rotary holder and slider do not lock sufficiently. (3) Play between pen return spring and pen tips. (4) Wire spring fatigue in wire unit (2-4), elongation of wire.	A B B B	Check gear engagement. Check rotary holder play by rotating it slowly by hand. Check by rotating X bobbin gear back and forth for several seconds by hand in the pen-down mode. Check for slack in wire	Mount properly. Exchange slider unit. Replace rotary holder. Replace wire unit.

Phenomenon	Condition	Cause	Level	Location and Method of Checking	Repair Method
	<p>(2) "F" is printed as shown below.</p>  <p>(3) "P" is short as shown below.</p> 	<p>(1) Pen and return spring.</p> <p>(2) Play in entire slider.</p> <p>(1) Faulty engagement of paper feed gear inside rubber roller unit after sliding by one tooth.</p> <p>(2) Substantial play in rubber roller bearing.</p>	B B A B	<p>Check by rotating Y idle gear back and forth for several seconds by hand in the pen-down mode.</p> <p>Check slider and X drive system.</p> <p>Check gear engagement.</p> <p>Check for play by moving the gear vertically.</p>	<p>Replace rotary holder.</p> <p>Exchange slider.</p> <p>Mount properly.</p> <p>Replace rubber roller unit.</p>

6. OILING STANDARD

Two types of oil are used in this printer – G488 and CRC 5-56. When oiling during disassembly and re-assembly, thoroughly clean the parts and oil in accordance with the table below.

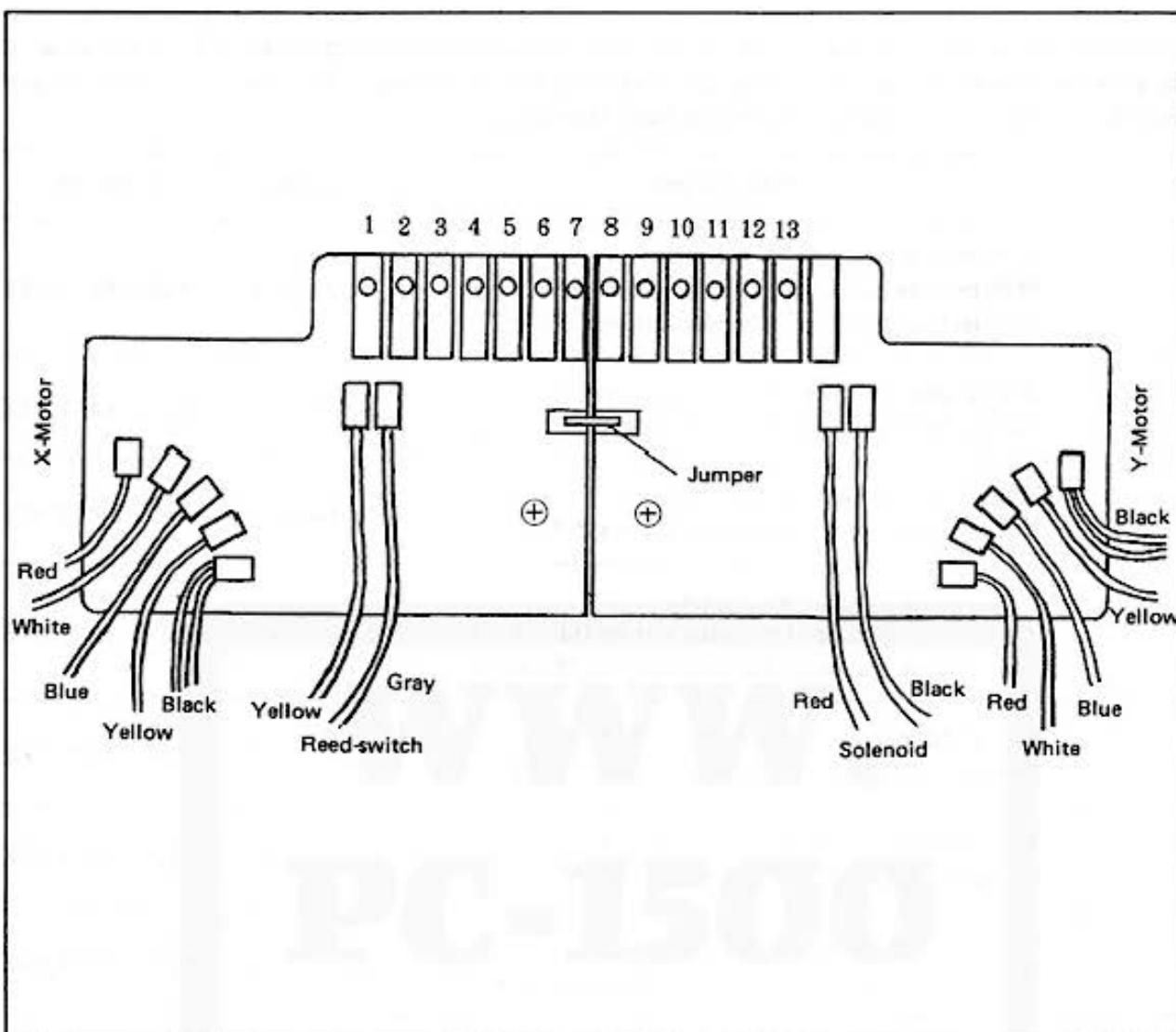
No.	Oiling Location	Oil Type	Part code
(1)	Area of contact between paper holding roller support plate (left) and side plate.	G-488	UKÖG-0108CSZZ
(2)	Contact section between paper holding roller support plate (right) and side plate.	G-488	UKÖG-0108CSZZ
(3)	Sliding sections (4 locations) between paper holding roller and roller shaft.	G-488	UKÖG-0108CSZZ
(4)	Sliding section between rubber roller unit shaft and plain washer.	G-488	UKÖG-0108CSZZ
(5)	Contact section between plain washer and side plate.	G-488	UKÖG-0108CSZZ
(6)	Sliding section between ejection lever shaft unit and slide plate.	G-488	UKÖG-0108CSZZ
(7)	Sliding section between ejection roller and slider set.	G-488	UKÖG-0108CSZZ
(8)	Tooth section of X idle gear.	G-488	UKÖG-0108CSZZ
(9)	Tooth section of Y idle gear.	G-488	UKÖG-0108CSZZ
(10)	Sliding section between holder stopper and holding plate.	G-488	UKÖG-0108CSZZ
(11)	Electromagnet unit actuator shaft.	G-488	UKÖG-0108CSZZ
(12)	Slider shaft (A)	CRC5-56	UKÖG-0098CSZZ
(13)	Slider shaft (B)	CRC5-56	UKÖG-0098CSZZ

7. ADHESION STANDARD

The table below shows points on the clamps where adhesion is to be applied to lock the screws as well as adhesion points on the printer bearings. Make sure that more than 1/4 of the screw heads are glued, but that no adhesive is present in the screw head recessions.

No.	Adhesion Point	Adhesive	Part Code
1	X motor unit set-screws in 2 places Phillips type pan head machine screws (SP2.3 x 3) Phillips type pan head machine screws (SP2 x 3)	Screw lock	UKOG-0003SCZZ
2	Bobbin gear unit wire set-screws (1 place) Phillips type pan head machine screws (SP2 x 3)	Screw lock	UKOG-0003CSZZ
3	Slider unit wire set-screws (1 place) Phillips type pan head machine screws (SP2 x 3)	Screw lock	UKOG-0003CSZZ
4	Y motor unit set-screws 2 places Phillips type pan head machine screws (SP2.3 x 3) Phillips type pan head machine screws (SP2 x 3)	Screw lock	UKOG-0003CSZZ
5	Electromagnet unit set-screws in one place Phillips type pan head machine screws (SP2.5 x 3)	Screw lock	UKOG-0003CSZZ
6	Reed switch unit set-screws in one place Phillips type pan head machine screws (SP2 x 3)	Screw lock	UKOG-0003CSZZ
7	Color change bar set-screws Phillips type pan head machine screws (SP1.4 x 1.6)	Screw lock	UKOG-0003CSZZ
8	Rubber roller unit bearing and side plate	Cyano-acrylate adhesive	UKOG-0032CSZZ

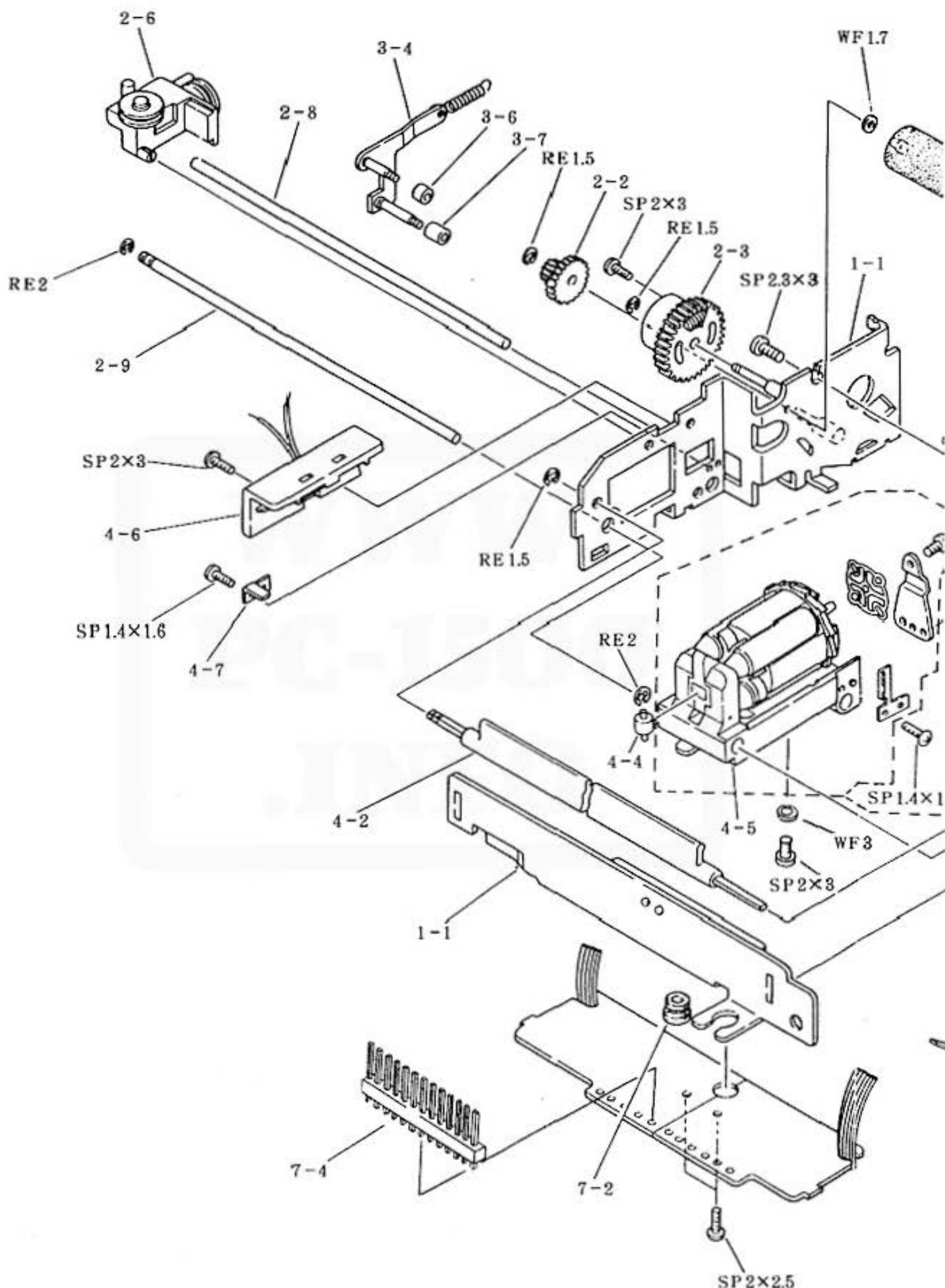
8. CIRCUIT DIAGRAM & WIRING

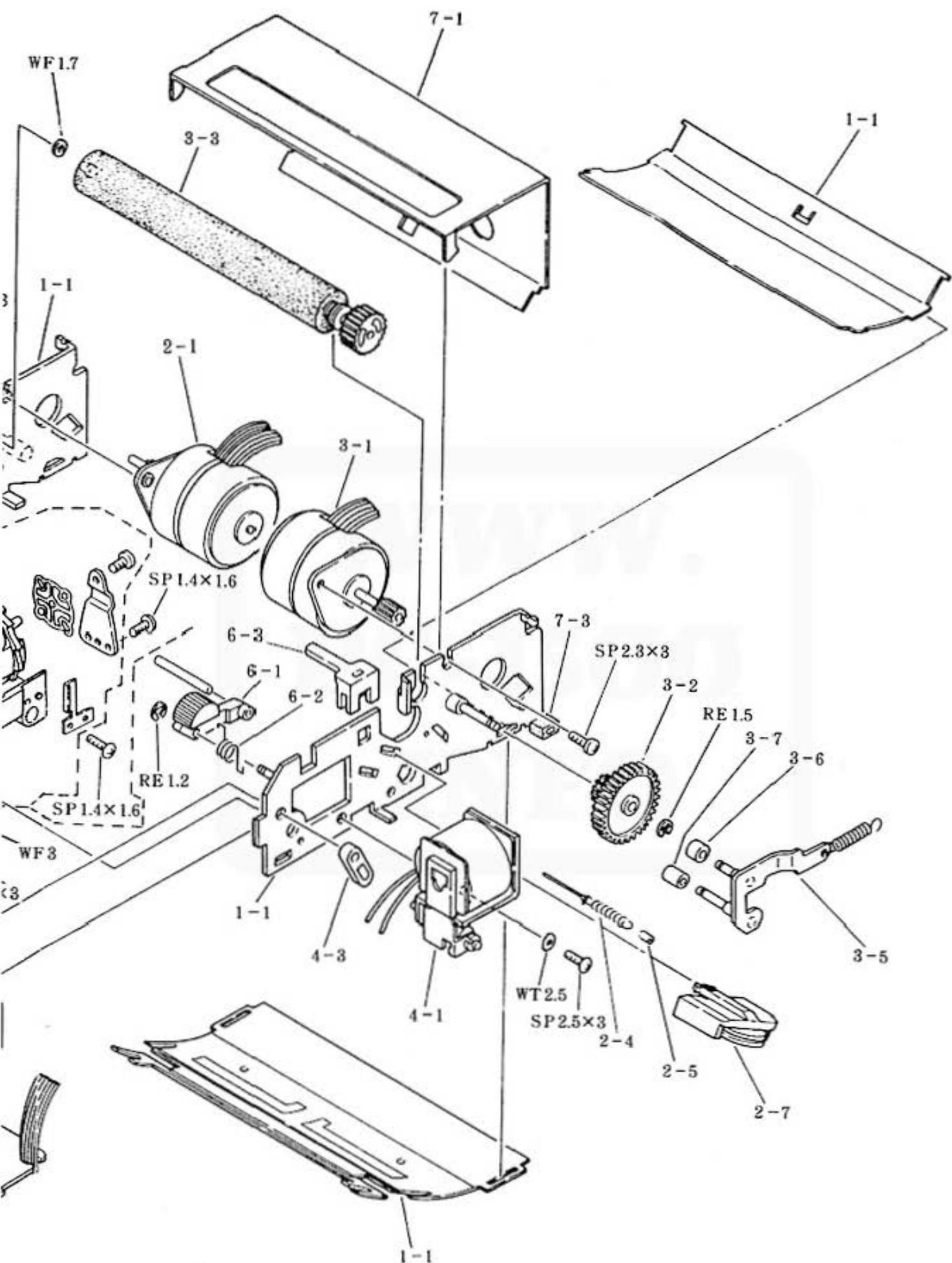


Name		No.	Circuit diagram
Color position detector	B	1	
	A	2	
X-Motor	D	3	
	C	4	
	B	5	
	A	6	
Motor common		7	
Y-Motor	D	8	
	C	9	
	B	10	
	A	11	
Solenoid (+)	B	12	
	(-)	13	

9. PARTS GUIDE AND PARTS LIST







SHARP **SERVICE MANUAL**



CE-150

Printer and cassette interface

**WWW.
PC-1500
.INFO**

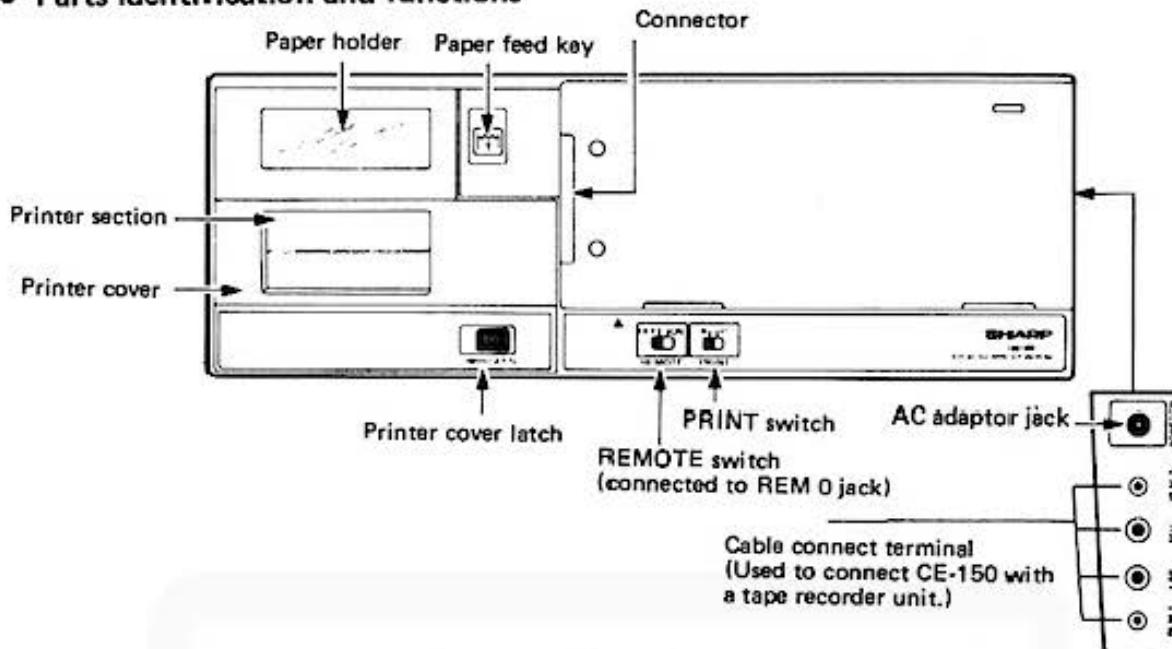
CE-150
PC-1500 Option Printer & Cassette Interface
Service Manual

TABLE OF CONTENTS

1.	Specifications	1
2.	System Block Diagram	7
3.	Circuit Description	8
4.	Circuit Daigram	11
5.	Parts and Signals Position	12
6.	Parts List & Guide	13

1. SPECIFICATIONS

- Parts identification and functions



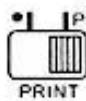
- Printer cover latch

Moving the latch lever to the OPEN position enables you to remove the printer cover.



- REMOTE switch

Moving the remote switch to the ON position allows the pocket computer to control start and stop operation of the REM 0 jack connected tape recorder. When in the OFF position, the remote control function is disabled through the REM 0 jack, and the tape recorder can then be operated manually.



- PRINT switch

Placing the switch to the P side performs automatic printout of the computing formula and results of manual calculation.

Turning the switch to the dot (·) side does not print the computing formula and results of manual calculation.

However, printout takes place with such a PRINT command as LPRINT and LINE, regardless of switch position.



- PAPER FEED key

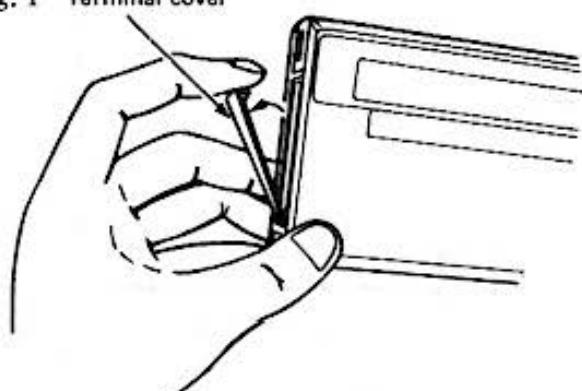
Basically, this key is used to feed paper. But, depression of this key together with the [0] key permits the printing pen exchange mode.

Also, simultaneous depression of this key with the [CL] key cancels the pen exchange mode and the machine returns to the normal operation mode.

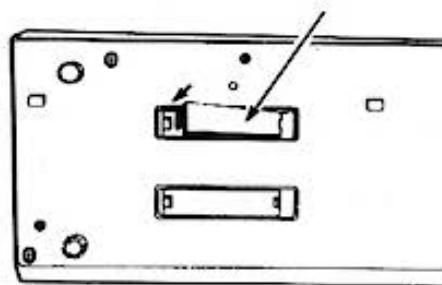
• Connection of the CE-150 with the PC-1500

- (1) Push the **OFF** key to turn power off.
- (2) Remove the terminal cover located on the left side panel of the pocket computer and place the cover in the terminal cover reserve slot located on the back panel of the CE-150 for future use.

Fig. 1 Terminal cover

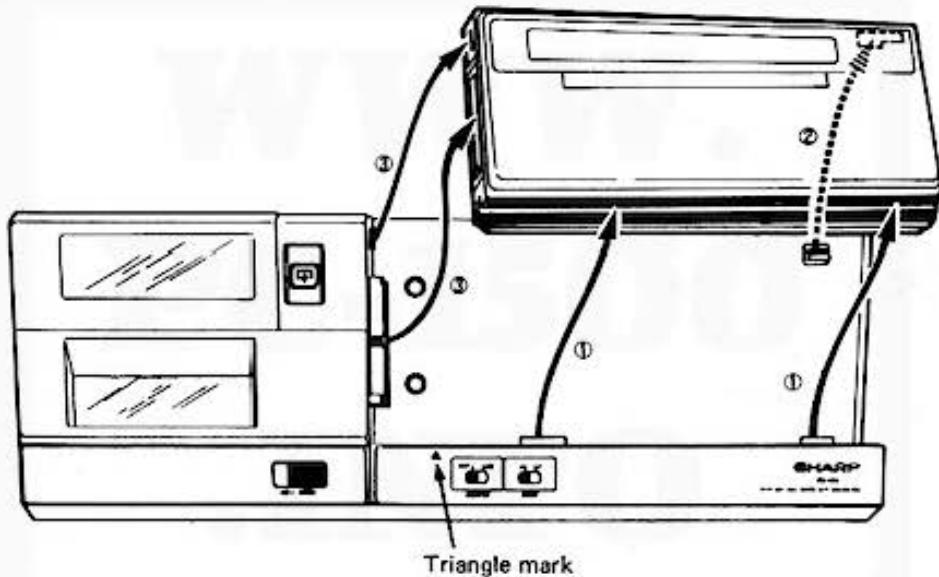


Hold the terminal cover here

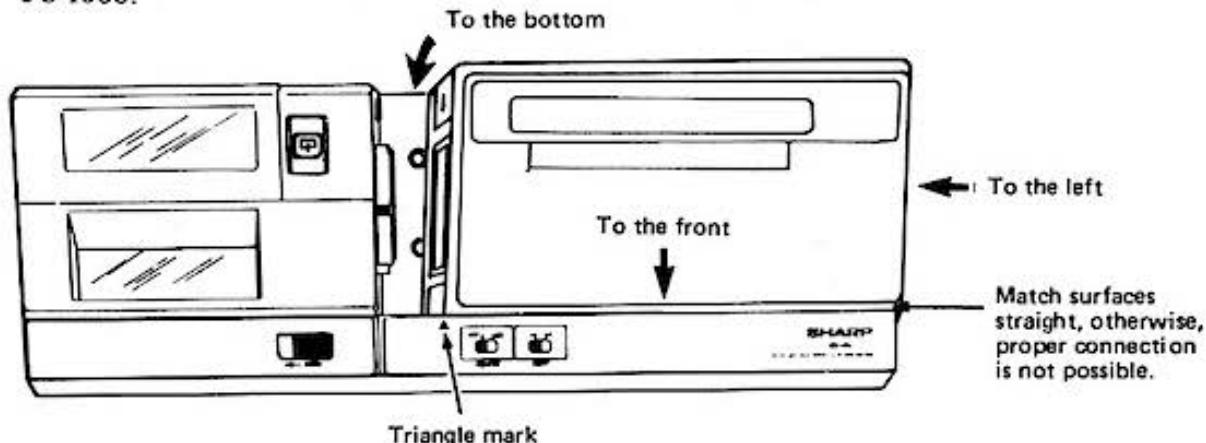


- (3) Engage studs on the CE-150 into the stud holes in the pocket computer as shown by arrowheads in order of number indicated and the procedure introduced next.

Fig. 2



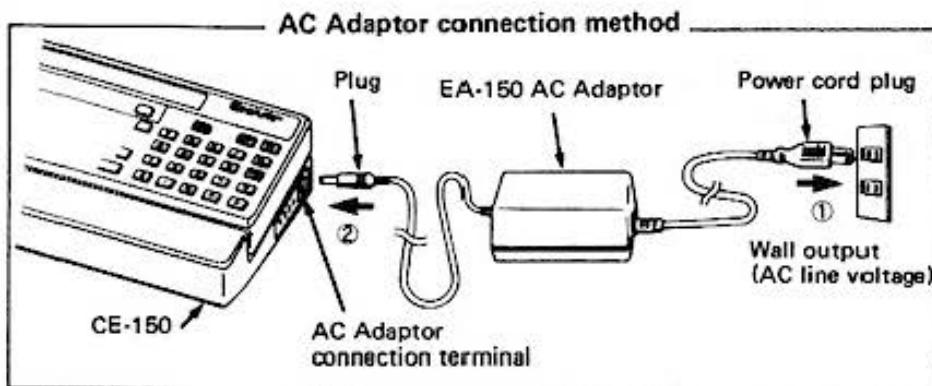
- 1) Place the PC-1500 on the CE-150 so that the triangle mark should be at the left edge of the PC-1500.



- 2) If the point 2) of Fig. 2 does not fit well, lightly move the pocket computer to left and right.
- 3) After the point 2) fits, insert the point 3) (connection terminal) next.
 - Make sure connector pins are firmly engaged and avoid forceful insertion.

- **How to charge power**

Push the **CL** key of the pocket computer and push the **OFF** key to turn off power, then make connection of the EA-150 AC Adaptor in the order of 1) and 2).



NOTE: Be sure to turn off power before the connection and disconnection of the AC Adaptor.

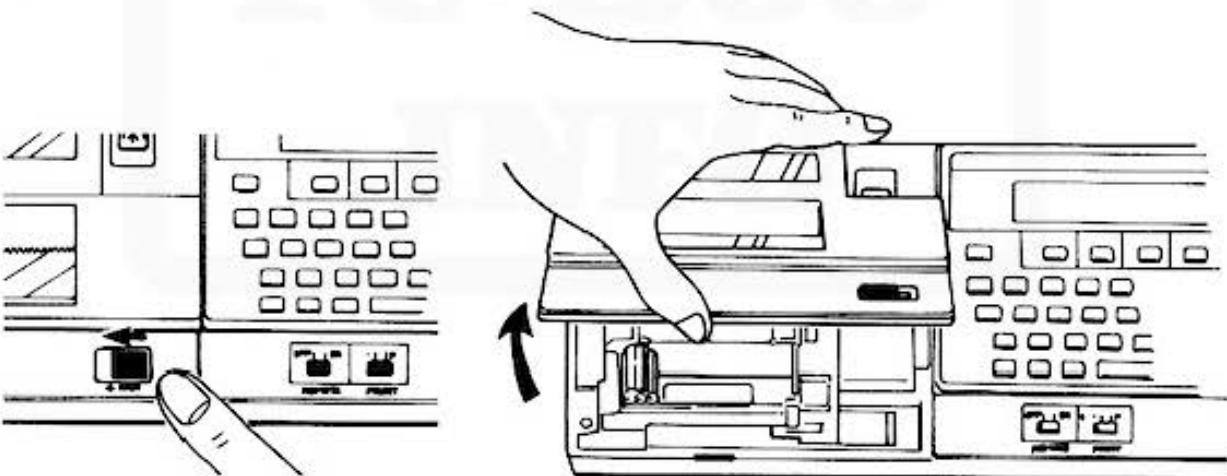
Battery recharge will be complete in about 15 hours when the AC Adaptor is connected and the power switch of the pocket computer is off.

Printing of about 1,100 characters for the size-2 print characters are permitted, when operated with the rechargeable battery supply.

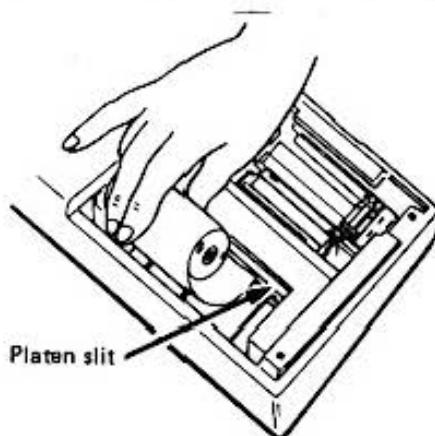
Recharge will also be complete in about 15 hours even if the pocket computer is operated, providing that the printer is kept out of operation.

- **How to change roll paper**

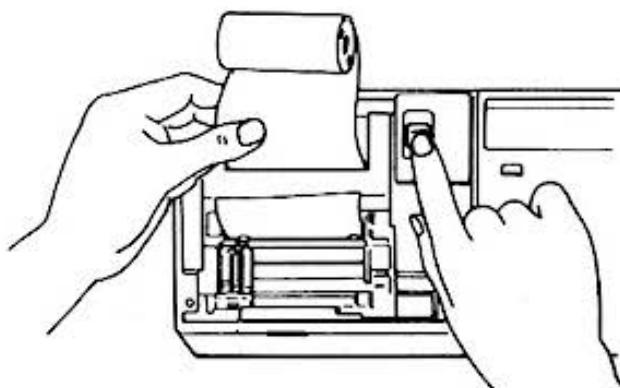
- 1) Move the printer cover latch lever towards the arrow direction to remove.



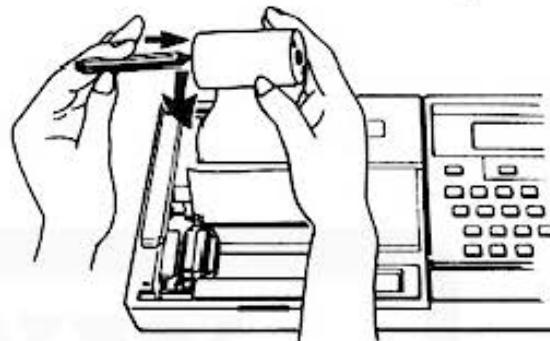
- 2) Cut the leading edge of the paper straight and insert the paper into the platen slit. Bent paper edge and wrinkles in the paper may sometimes impede proper paper insertion.



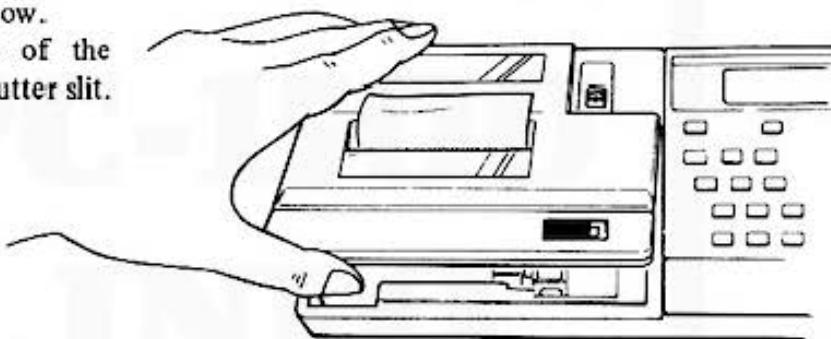
- 3) Now, turn on the power of the pocket computer by pushing the **ON** key and push the  key on the printer until the paper is advanced about 3 to 5cm from the platen. If the platen does not feed the paper properly use your fingers to help the paper advance.



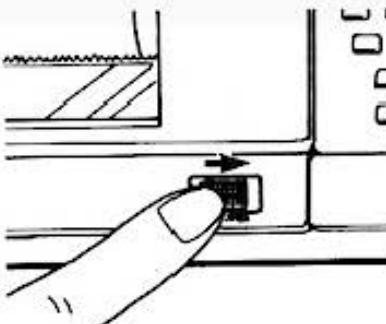
- 4) Insert the roll paper shaft into the paper roll core hole and place it into the paper holder casing.



- 5) Replace the printer cover now.
Thread the leading edge of the paper through the paper cutter slit.



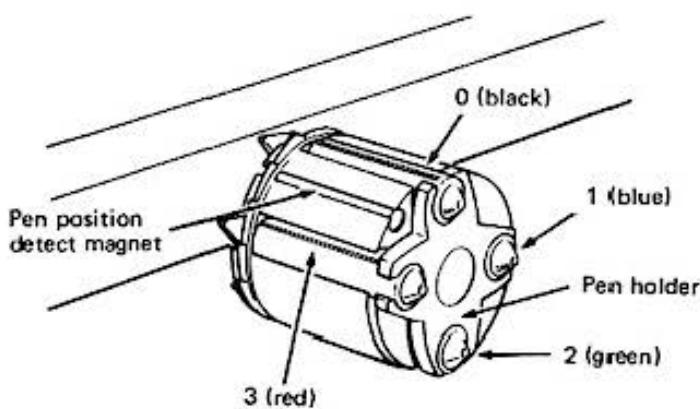
- 6) Lock the printer cover.



- To take out the paper from the printer, pull it straight out through the paper holder or paper feed side.

- How to install and exchange writing pens

Four different color ball point pens can be used on this printer. They must be arranged in position shown below.



When pen position, 0~3, is chosen by the COLOR command, it will be selected by the action of the pen position detect magnet. Pen positions are arranged in order of 0, 1, 2, 3, clockwise from the magnet position. However, the pen holder rotates counterclockwise to bring the designated pen into the upper position.

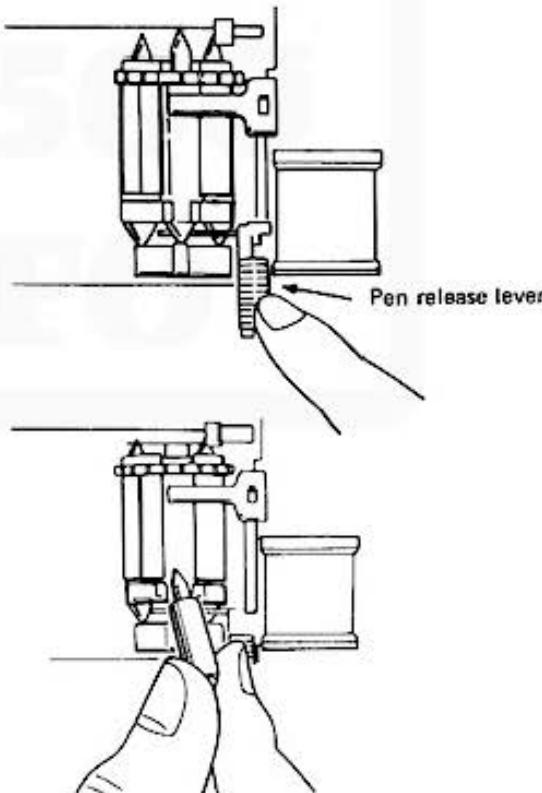
Use the following instructions to install or exchange pen(s).

1) Push the **[↑]** key on the printer while depressing the **[0]** key on the pocket computer. The printer now goes into the pen exchange mode and the pen holder moves to the left margin. It rotates to bring the first replaceable pen to the upper position, then it moves to the right margin and waits for the pen replacement there. Be sure to release your finger from the key as soon as the pen holder starts to move.

2) Push down the pen release lever to remove the pen on the upper position.

NOTE: Place your finger tip lightly in front of the pen before removing the pen, in order to prevent the pen from jumping into the printer mechanism.

3) Replace with the new pen.



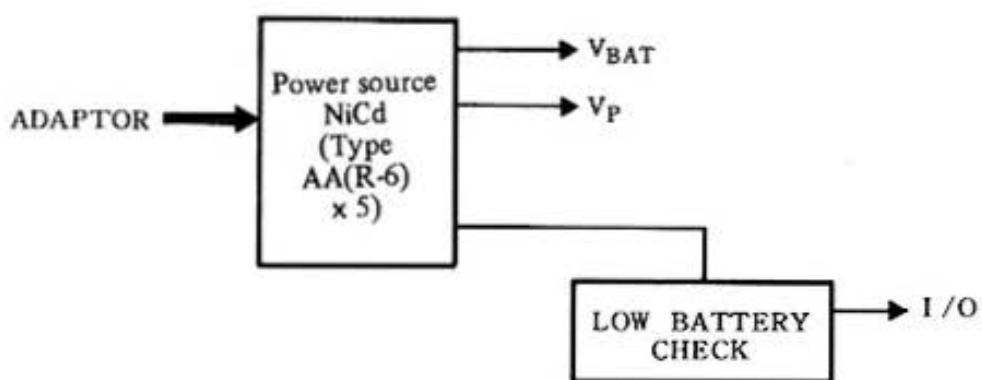
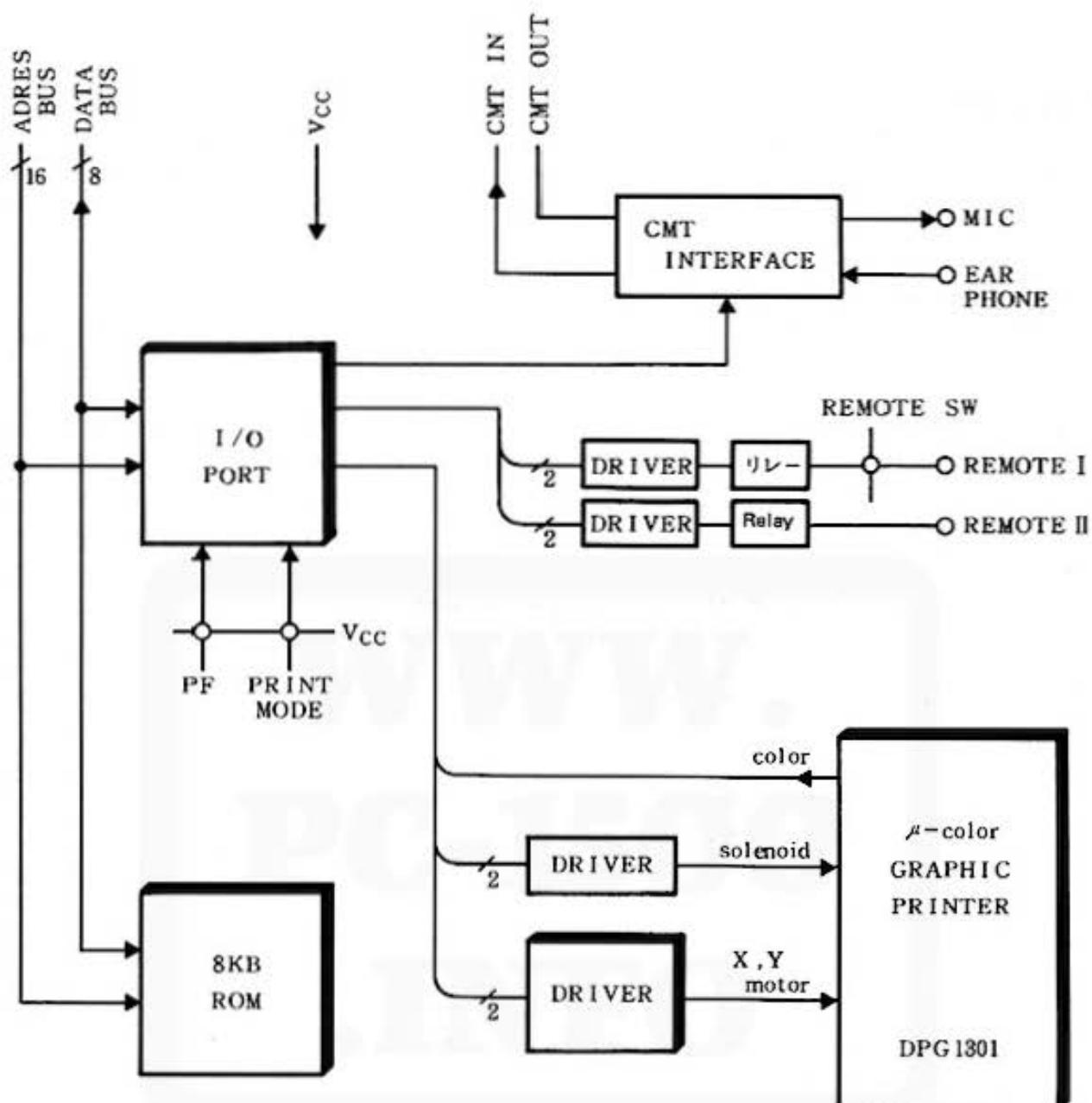
4) To remove or replace a succeeding pen, push the **[↑]** key again. This makes the pen holder move to the left margin and the pen holder rotates to the next exchanging pen position, and then starts to move to the right margin. As soon as the pen holder stops at the right margin, remove or replace the pen with the new one in accordance Steps, 2 and 3.

5) When pen replacement or installation is finished, push the **[CL]** key of the pocket computer together with the **[↑]** key on the printer. Then, the pen exchange mode is cancelled and the pen holder returns to the left margin.

• SPECIFICATIONS

Printer type:	DPG1301
Printing method:	X-Y plotter method
Print capacity:	18 digits, nominal. (Possible to select 36, 18, 12, 9, 7, 6, 5, and 4 digits operations)
Character size:	Nine variable sizes, 1.2mm x 0.8mm thru 10.8mm x 7.2mm.
Ball point pen:	EA850C (black, blue, green, red) EA850B (black only)
Printing directions:	Four directions (up, down, right, left)
Minimum pen moving distance:	0.2mm
Print speed:	11 characters/sec, max. (Print speed may vary according to character printed.)
Print paper:	Roll paper with maximum outer diameter 30mm and width of 58mm.
Power source:	Supplies as EA-1500P Rechargeable battery. EA-150 AC Adaptor
Line print power capacity:	About 1,100 lines (continuous printing of 555555555 of the character size 2 under 20°C, with a slight variation depending on operating condition).
Power consumption:	5.2W
Operating temperature:	5 to 40°C
Physical dimensions:	330(W) x 115(D) x 50(H)mm
Weight:	900g, except accessories
Accessories:	Carrying case, tape recorder connection cables (one each of three-line and one-line wires), EA-150 AC Adaptor, 3 rolls of paper, one each of black, blue, green, and red pen, name label, instruction manual.

2. SYSTEM BLOCK DIAGRAM



Referring to the block diagram, the CE-150 is the printer (DGP1301) and the CMT (cassette tape recorder interface) which consists of three blocks, the CMT interface, printer interface and power supply circuit.

- **CMT interface**

The CMT interface is the circuit that handles data transfer between the PC-1500 and the CMT and consists of a simple driver circuit (voltage level conversion circuit).

In addition, there is the CMT on/off remote controlling circuit which is under the program control of the PC-1500 and performs switching operation via the I/O port using the relay circuit.

- **Printer interface**

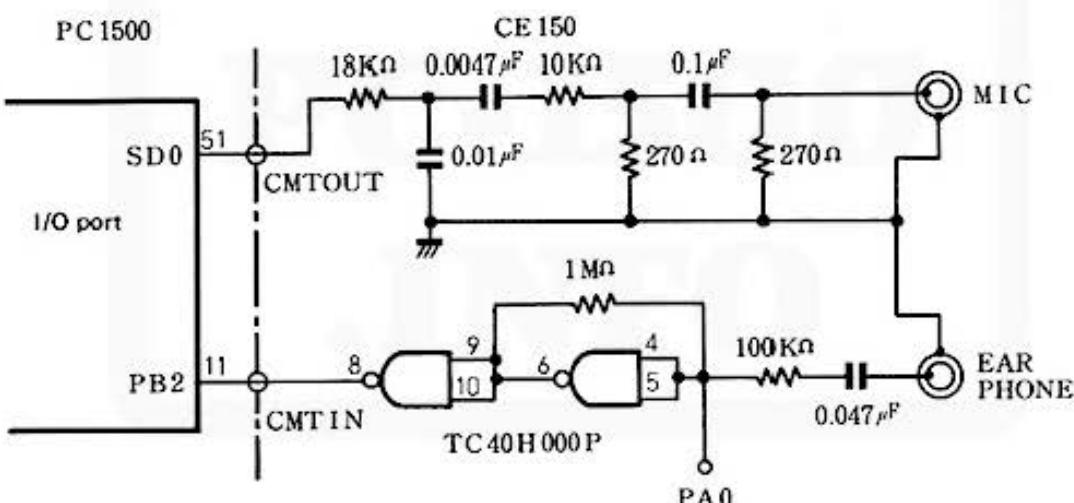
The printer interface consists of character generator ROM (LH5367-01), I/O port (LH5801), motor drive circuit, print solenoid drive circuit, and the pen color detect input signal circuit.

- **Power supply block**

A simple power stabilization circuit is provided to take care of recharge to the built-in NiCd batteries from the AC Adaptor.

3. CIRCUIT DESCRIPTION

3-1. CMT Interface circuit diagram



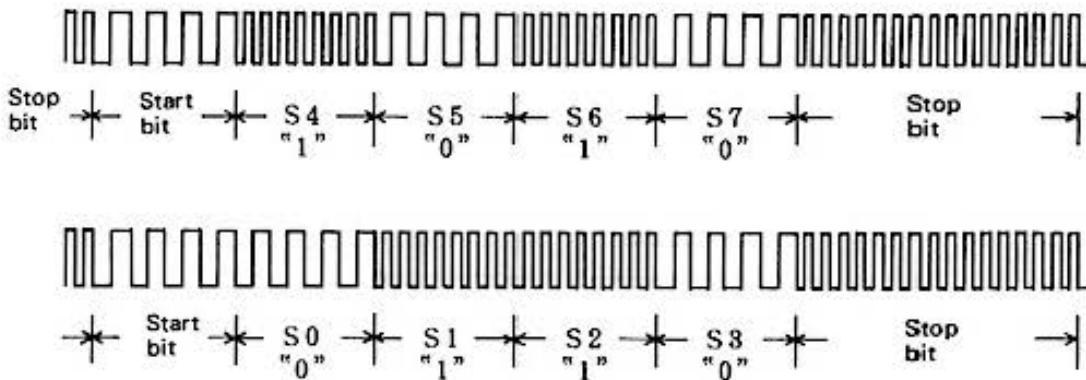
The signal CMT OUT is sent from the PC-1500 to write data on the cassette tape and the capacitor coupled interface circuit is provided to shift the voltage level before supplying the MIC jack. The read signal is the input from the EARPHONE jack and passed through the capacitor and two stages of gates to perform voltage level shift and waveform shaping and delivered to the PC-1500 as the CMT IN signal.

With respect to the CMT IN signal, the gate input to the first stage of the gate is forced to a low level by means of the PA0 signal as there is a possibility of supplying noise to the EARPHONE jack as input data to the PC-1500, except during the data read mode.

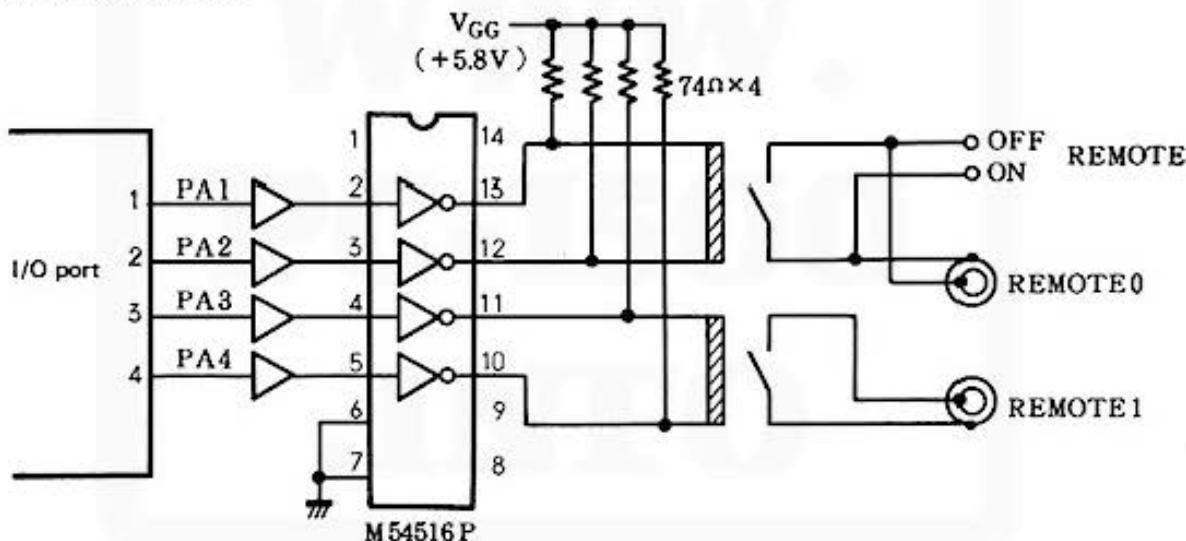
- **Recording signal**

When the contents of the start bit or data bit is "0", four pulses of 1.27kHz are recorded in a time of 3.15ms. When the contents of the data bit is "1", eight pulses of 2.54kHz are recorded in the tape in a time of 3.15ms.

Shown below is an example of the contents of S (one byte) is "0 1 0 1 0 1 1 0".

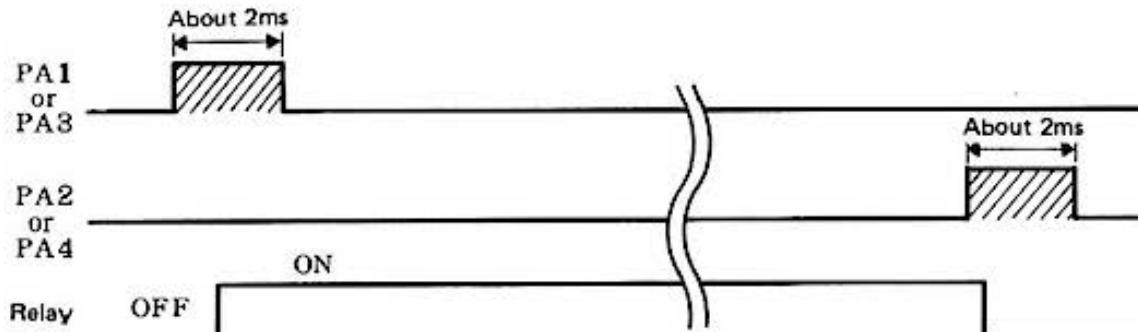


3-2. Remote circuit

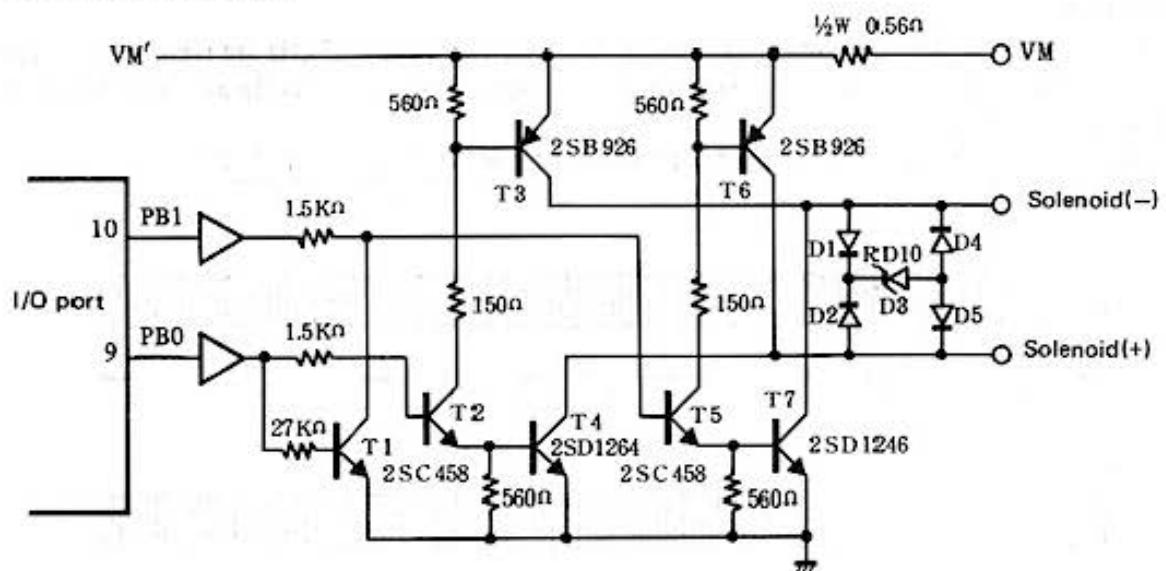


This circuit is provided to perform automatic control of CMT on/off action. Signal from the PC-1500 is used to actuate the relay via the I/O port (LH5810 or LH5811).

When the REMOVE ON/OFF switch is in the OFF position, the REMOTE 0 circuit shuts off and disables control from the PC-1500. Normally, connection is made to the REMOTE 0 jack when one unit of CMT is used and both REMOTE 0 and REMOTE 1 jacks are connected to use two units of CMT.



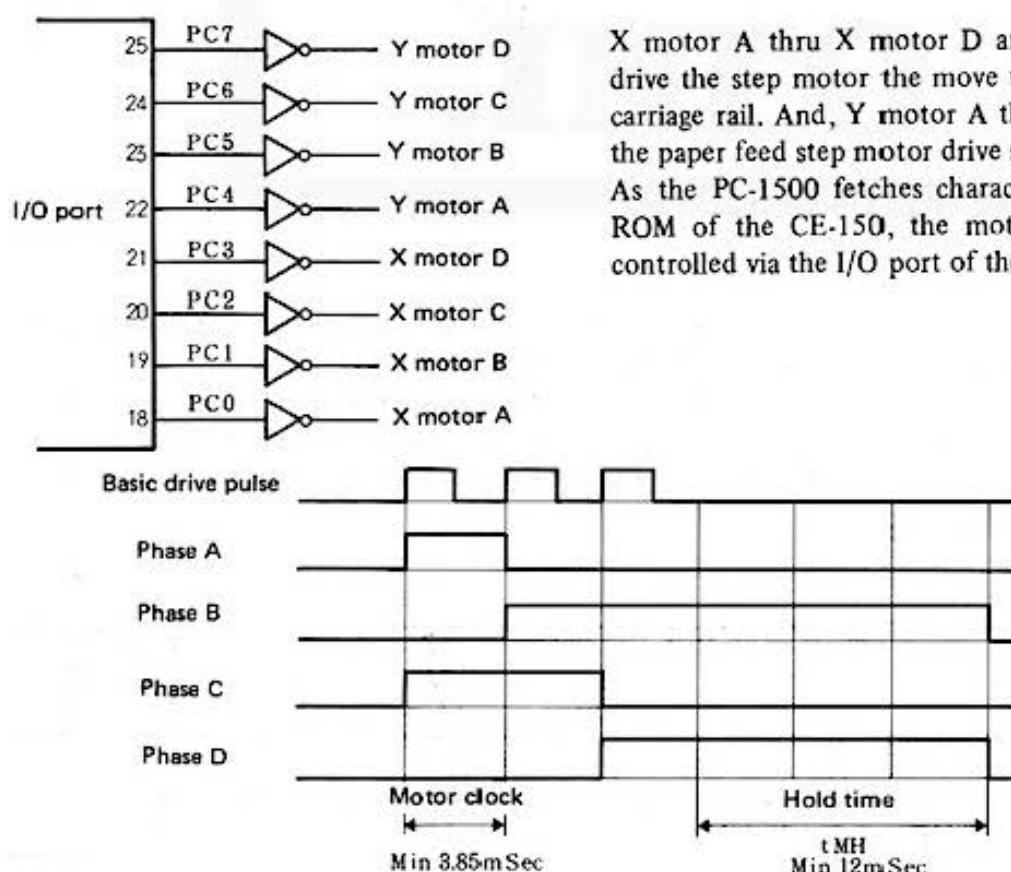
• Solenoid drive circuit



The solenoid is driven by the PEN UP signal PB0 and the PEN DOWN signal PB1 that is sent to the drive circuit via the I/O port.

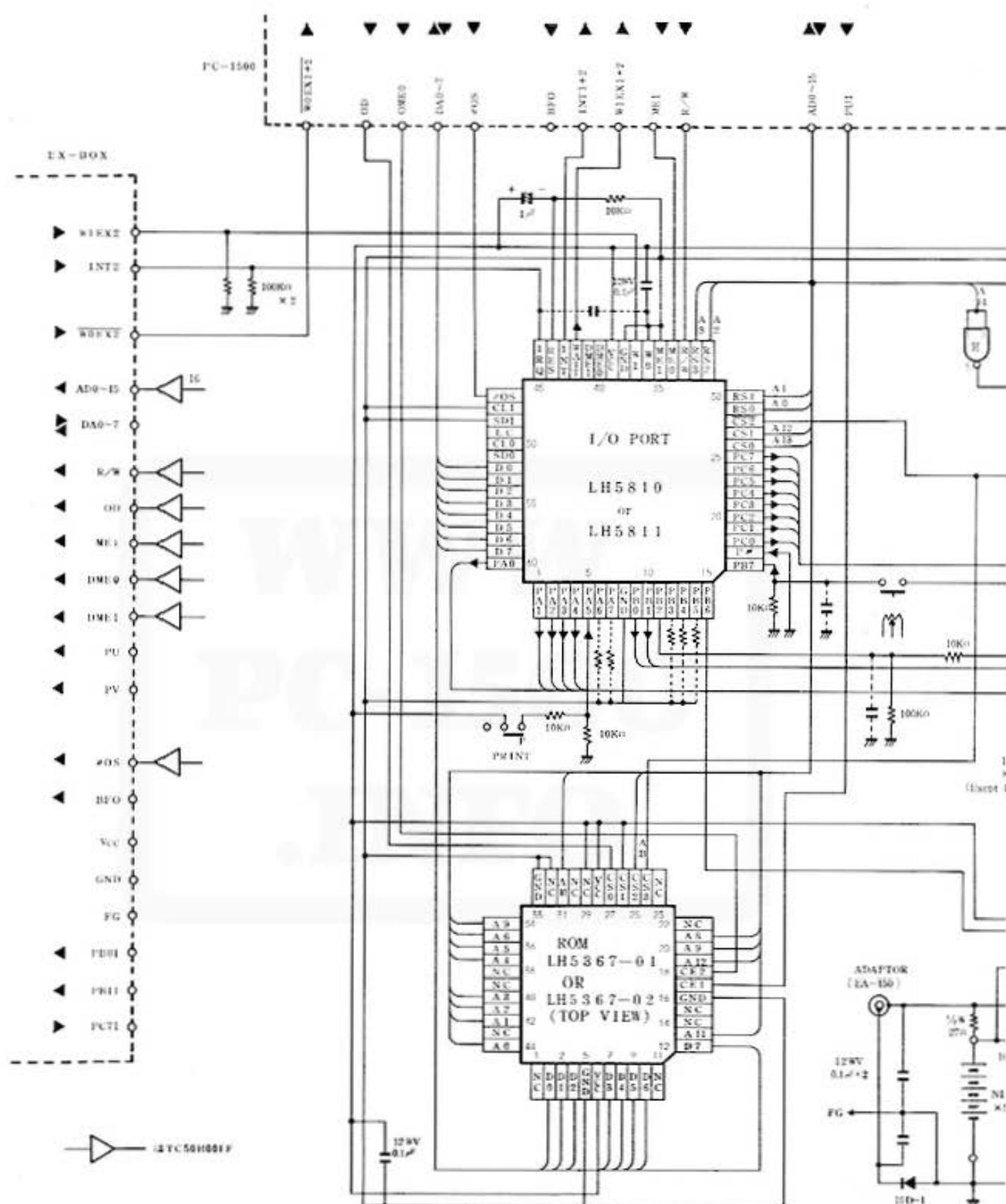
- 1) The high state of the PB1 signal, T5 and T7 actuates to turn the solenoid(–) terminal GND level. Actuation of the T5 causes T6 to actuate to turn the solenoid(+) terminal VM level (about +6V) and drive the solenoid of lower the pen to start printing.
- 2) The high state of the PB1 signal, T2 and T4 actuates to turn the solenoid(+) terminal GND level. Actuation of the T2 causes T3 to actuate to turn the solenoid(–) terminal VM level (about +6V) and drive the solenoid to lift up the pen to stop printing.
- 3) D1 thru D5 are the arc suppressor diodes for the solenoid.

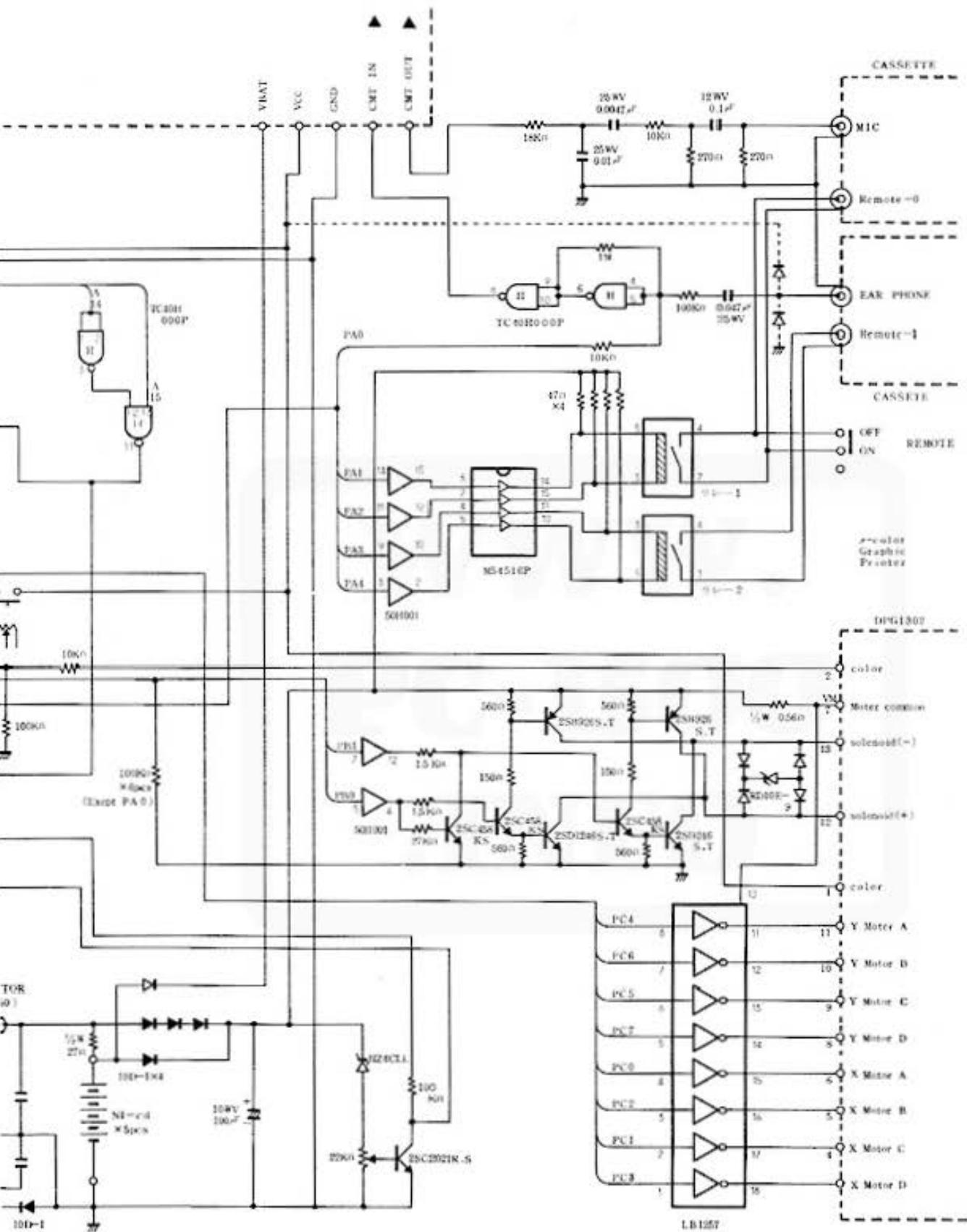
• Motor drive circuit



X motor A thru X motor D are the signals that drive the step motor the move the carriage on the carriage rail. And, Y motor A thru Y motor D are the paper feed step motor drive signals. As the PC-1500 fetches character data from the ROM of the CE-150, the motor drive action is controlled via the I/O port of the CE-150.

4. CIRCUIT DIAGRAM



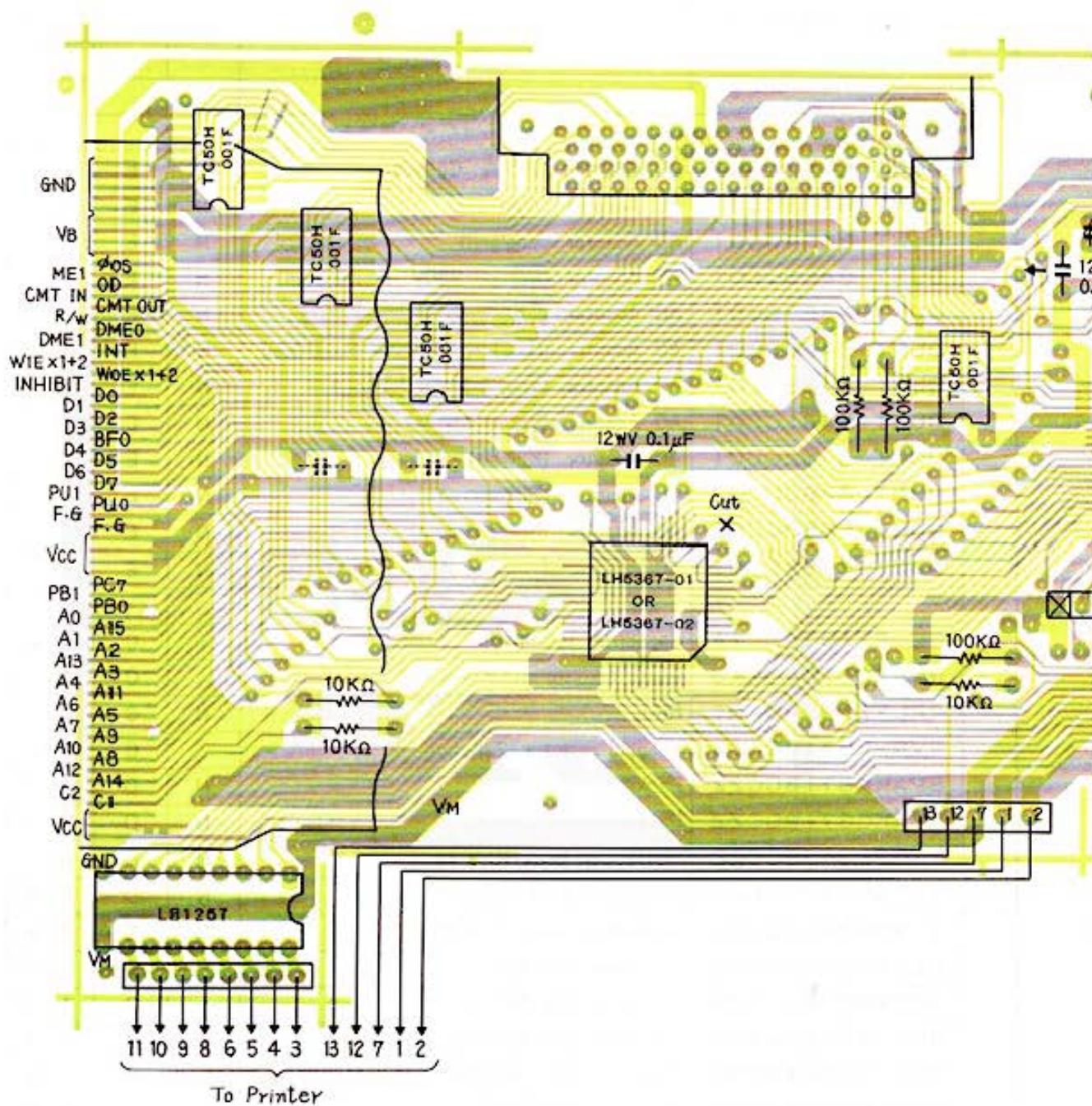


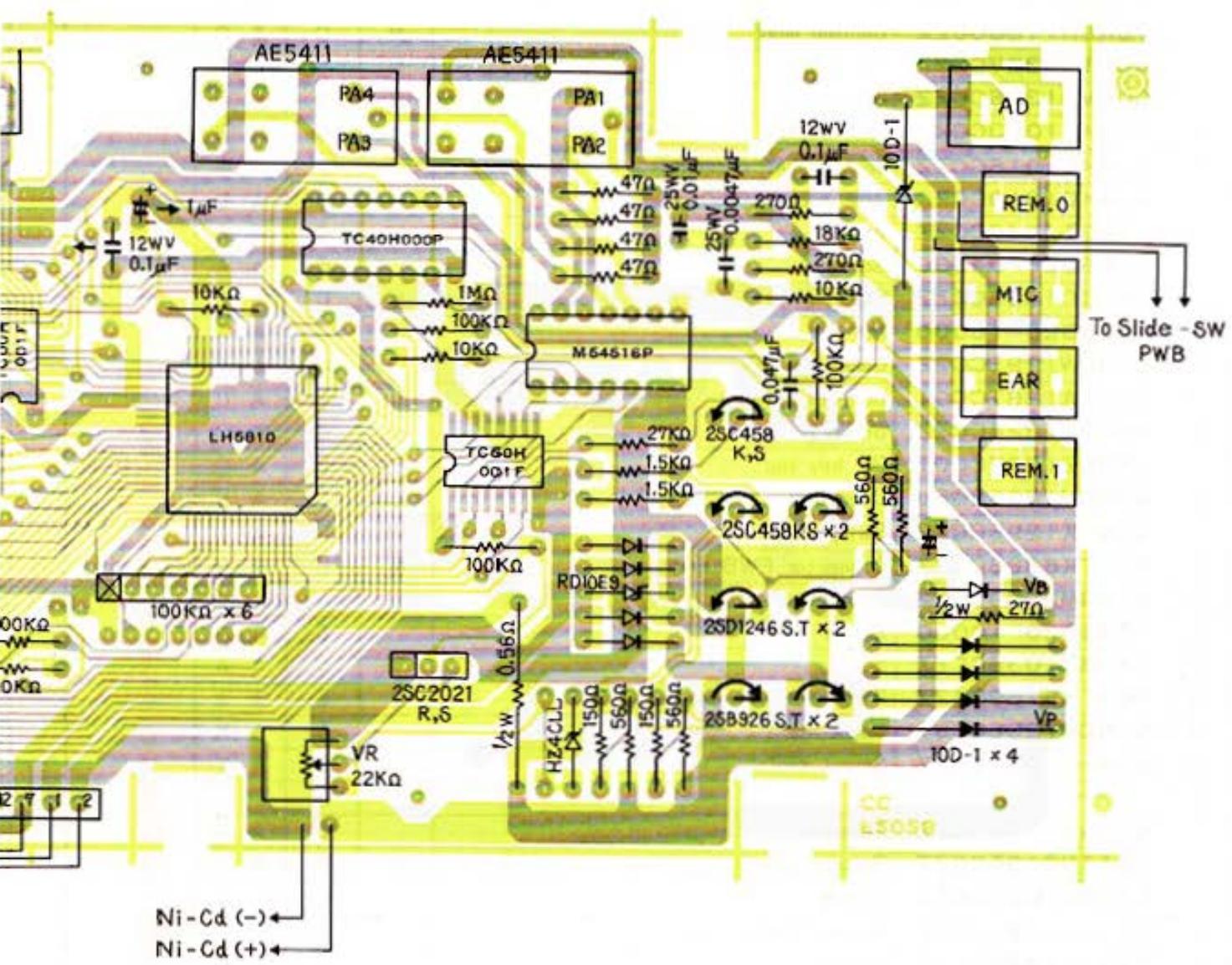
6. PARTS LIST & GUIDE

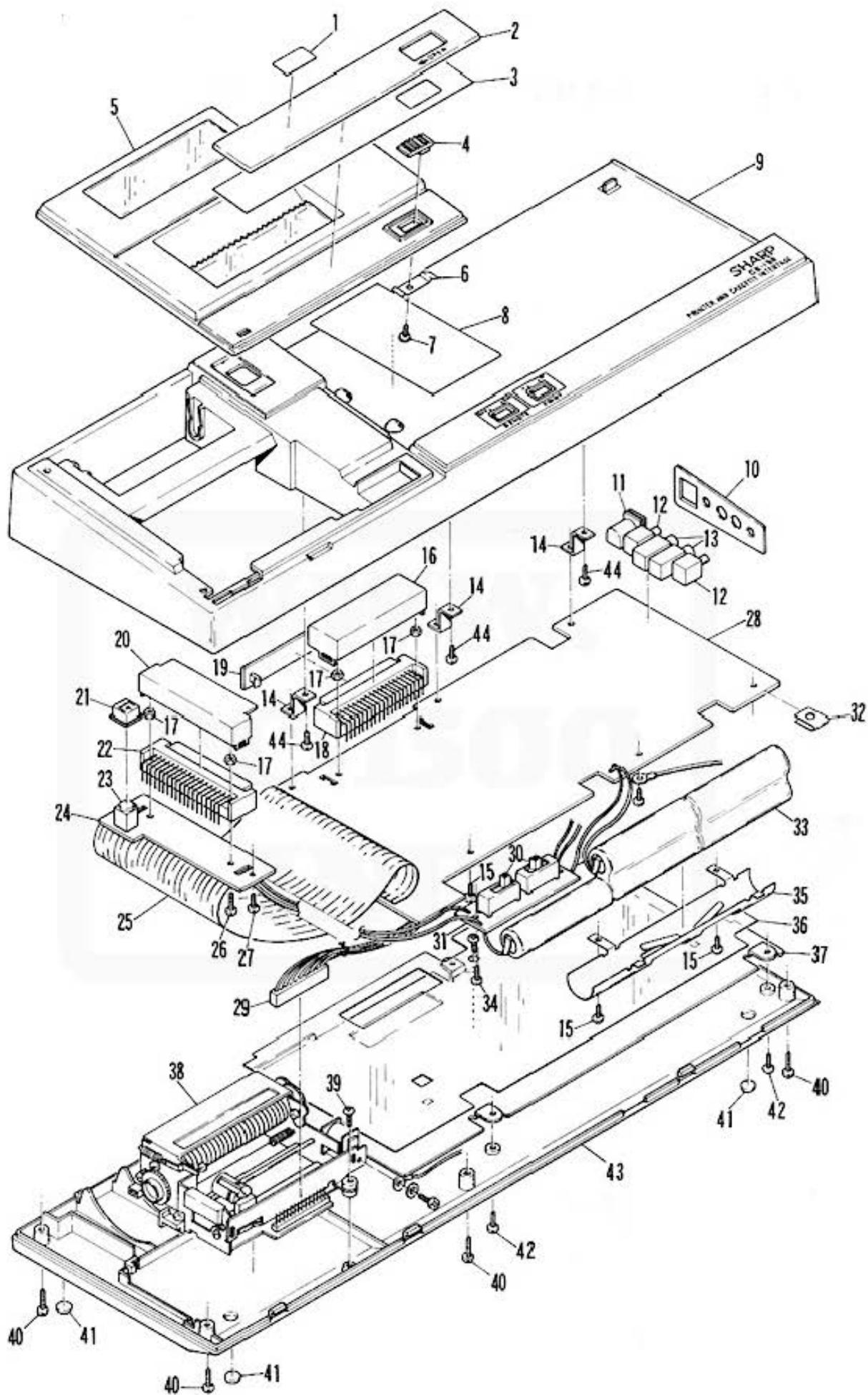
NO.	PARTS CODE	DESCRIPTION	NEW MARK	PARTS RANK	PRICE RANK
1	TLABH1628CCZZ	Pen label	N	D	A A
2	HDECA2018CCZZ	Dec. panel	N	D	A E
3	PTPEH1133CCZZ	Adhesive tape	N	D	A B
4	JBTN-1009CCZZ	Lock knob	N	C	A B
5	CCOVA1335CC0I	Printer cover	N	D	A E
6	MSPRP1189CCZZ	Lock spring	N	C	A B
7	XUPSD26P04000	Screw		C	A A
8	TCAUH1180CCZZ	Caution label	N	D	A C
9	DUNTG6387CCZZ	Top cabinet	N	D	A T
10	GC0VH1336CCZZ	Jack cover	N	D	A E
11	QJAKC1003CCZZ	Jack socket (AC adaptor)		C	A D
12	QJAKC1016CCZZ	Jack socket (Remote)		C	A C
13	QJAKC1013CCZZ	Jack socket (MiC, EAR)		C	A C
14	LANGT1448CCZZ	Angle	N	C	A B
15	XBBSD20P04000	Screw		C	A A
16	PSLDC1335CCZZ	Shield plate	N	C	A C
17	XNESD20-16000	Nut		C	A A
18	QCNCWI293CCZZ	Connector 60pin		C	A Y
19	GFTAA1267CC0I	Connector cover	N	D	A C
20	PSLDC1324CCZZ	Shield plate	N	C	A C
21	JKNBZ1737CC0I	PF key top		C	A G
22	QCNCMI295CC6J	Connector		C	A V
23	QSW-K1295CCZZ	Switch		B	A D
24	DUNTK6388CCZZ	Connector PWB unit	N	E	A Z
25	QPWBM2029CCZZ	FPC		C	A V
26	XBBSD20P10000	Screw		C	A A
27	LX-BZ1038CCZZ	Screw		C	A A
28	DUNTK5169CSZZ	PWB unit with connector PWB	N	E	※※
29	QCNCWI296CC0I	Connector		C	A K
30	QSW-S1074CCZZ	Select switch		B	A E
31	—				
32	QTANS1372CCZZ	Terminal		C	A B
33	UBATN2135CCZZ	Battery unit	N	B	A Z
34	LX-BZ1038CCZZ	Screw		C	A A
35	LFIX-1127CCZZ	Battery fixing plate		C	A E
36	PZETL1429CCZZ	Insulator		C	A D
37	QTANS1365CCZZ	Angle	N	C	A E
38	KI-OB0066CCZZ	Printer unit (DPG-1301)	N	E	B T
39	XUPSD30P06000	Screw		C	A A
40	XUPSF26P08000	Screw		C	A A
41	GLEGP1009CCZZ	Lubber foot		C	A A
42	XBBSF20P06000	Screw		C	A A
43	GCABA2605CCZZ	Cabinet bottom	N	D	A L
44	LX-BZ1116CCZZ	Screw		C	A A

NO.	PARTS CODE	DESCRIPTION	NEW MARK	PARTS RANK	PRICE RANK
	DSFTZ0480CSZZ	Paper shaft unit	N	C	A F
	QPLGJ1013CCZZ	Cassett plug B	N	C	A K
	QPLGJ1014CCZZ	Cassett plug A	N	C	A R
	TLABZ1295CCZZ	Name label			A A
	UBAGZ1292CCZZ	Hard case			A Y
	DUNT-6452CC01	AC Adaptor (U.S.A, Canada)	N	E	
	DUNT-6453CC01	AC Adaptor (Australia)	N	E	
	DUNT-6454CC01	AC Adaptor (England)	N	E	
	DUNT-6455CC01	AC Adaptor (Europe 220V Round-2pin)	N	E	
	DUNT-6457CC01	AC Adaptor (Taiwan, Philippines, Brazil) 110V/220V Flat-2pin	N	E	
	DUNT-6458CC01	AC Adaptor (Hong kong 110/220V Round-3pin)	N	E	
	DUNT-6459CC01	AC Adaptor (Europe, Middle East 110V/220V Round-2pin)	N	E	
	DUNT-6461CC01	AC Adaptor (South Africa)	N	E	
	DUNT-6462CC01	AC Adaptor (Kuwait)	N	E	
	DUNT-6463CC01	AC Adaptor (Singapore)	N	E	
	DUNT-6553CC01	AC Adaptor (Soudi-Arabia)	N	E	
	DUNT-6452CC01	AC Adaptor (120V Latin-America)	N	E	
	RC-EZ107ACC1A	Capacitor 10V 100μF		C	A B
	RMPTC6104QCKJ	Block resistor 100Kohm×6		C	A C
	RRLYZ9999QCN1	Relay	N	C	A U
	RVR-MB410QCZZ	Volatile resistor 22Kohm		C	A D
	SPAKA6718CCZZ	Packing cushion	N	D	A K
	SPAKC6719CCZZ	Packing case (U.S.A)	N	D	A K
	TINSE3439CCZZ	Instruction book (U.S.A)	N	D	A N
	TINSE3485CCZZ	Instruction book (English)	N	D	A Q
	TINSM3484CCZZ	Instruction book (E, F, G, S)	N	D	A H
	VCEAAUIHW105Q	Capacitor 50V/1μF		C	A B
	VCTYPU1EX103M	Capacitor 25V 0.01μF		C	A B
	VCTYPU1EX472M	Capacitor 25V 4700PF		C	A A
	VCTYPU1EX473M	Capacitor 25V 0.047PF		C	A B
	VCTYPU1NX104M	Capacitor 12V 0.1μF		C	A B
	VHDDS1588L1-I	Diode 1S1588L1		B	A D
	VHD10D1///-I	Diode 10D1		B	A D
	VHEHZ4CLL//I	Zener diode 4~4.4V		B	A E
	VHERD10E9//I	Zener diode 9.3~9.9V		B	A C
	VHLB1257//I	I. C. (LB1257)	N	B	A M
	VHLH536702-I	L.S.I. (LH536702)	N	B	B B

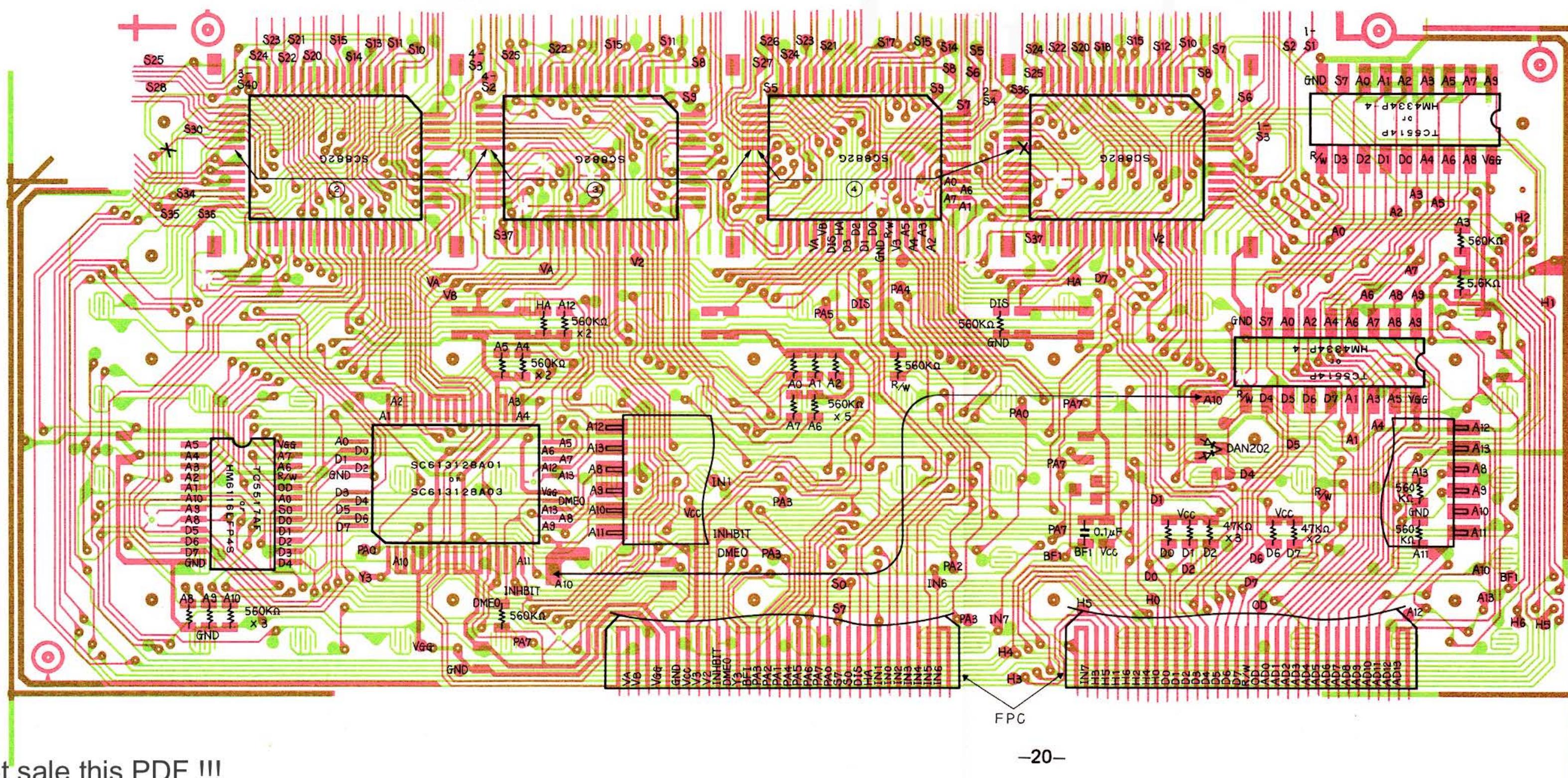
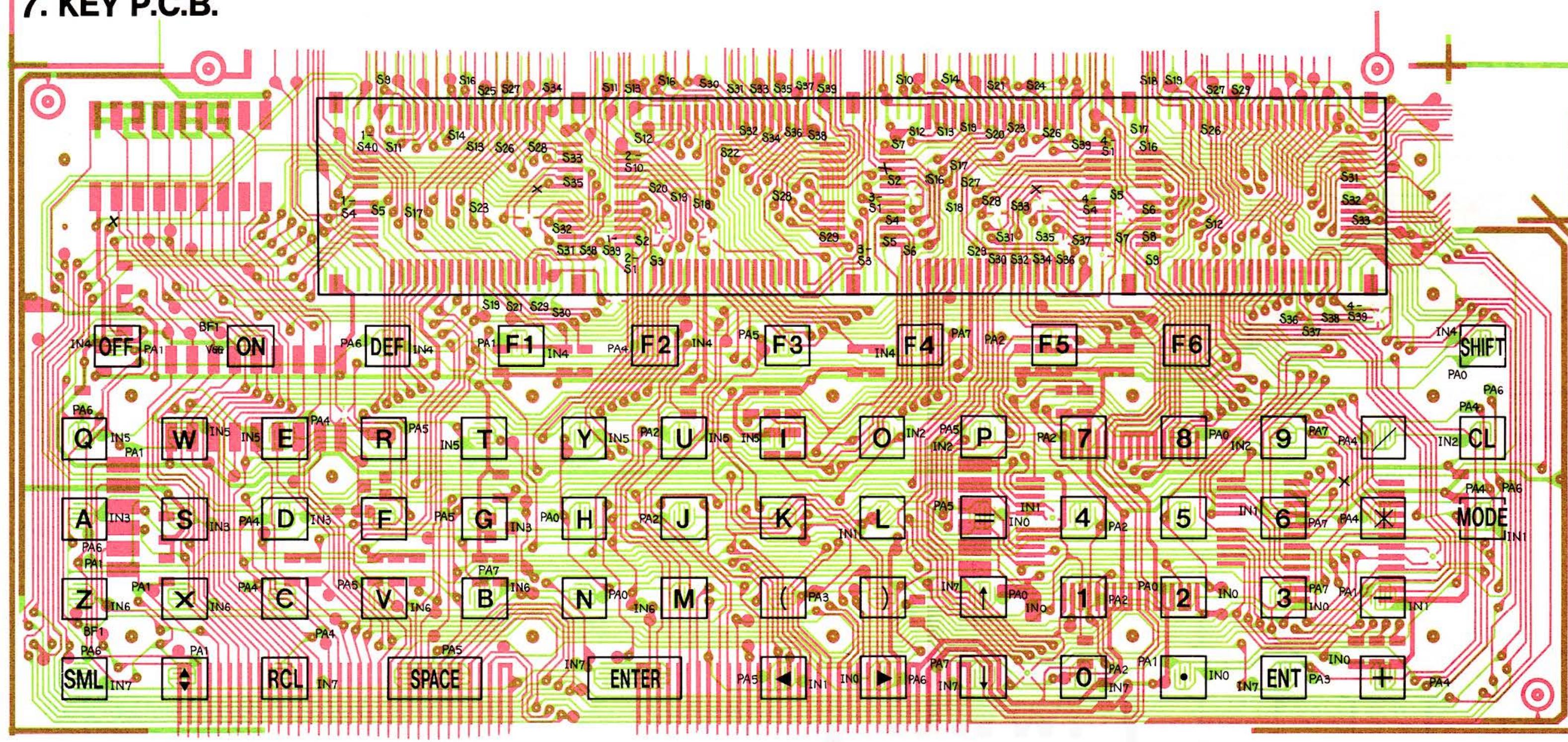
5. PARTS AND SIGNALS POSITION



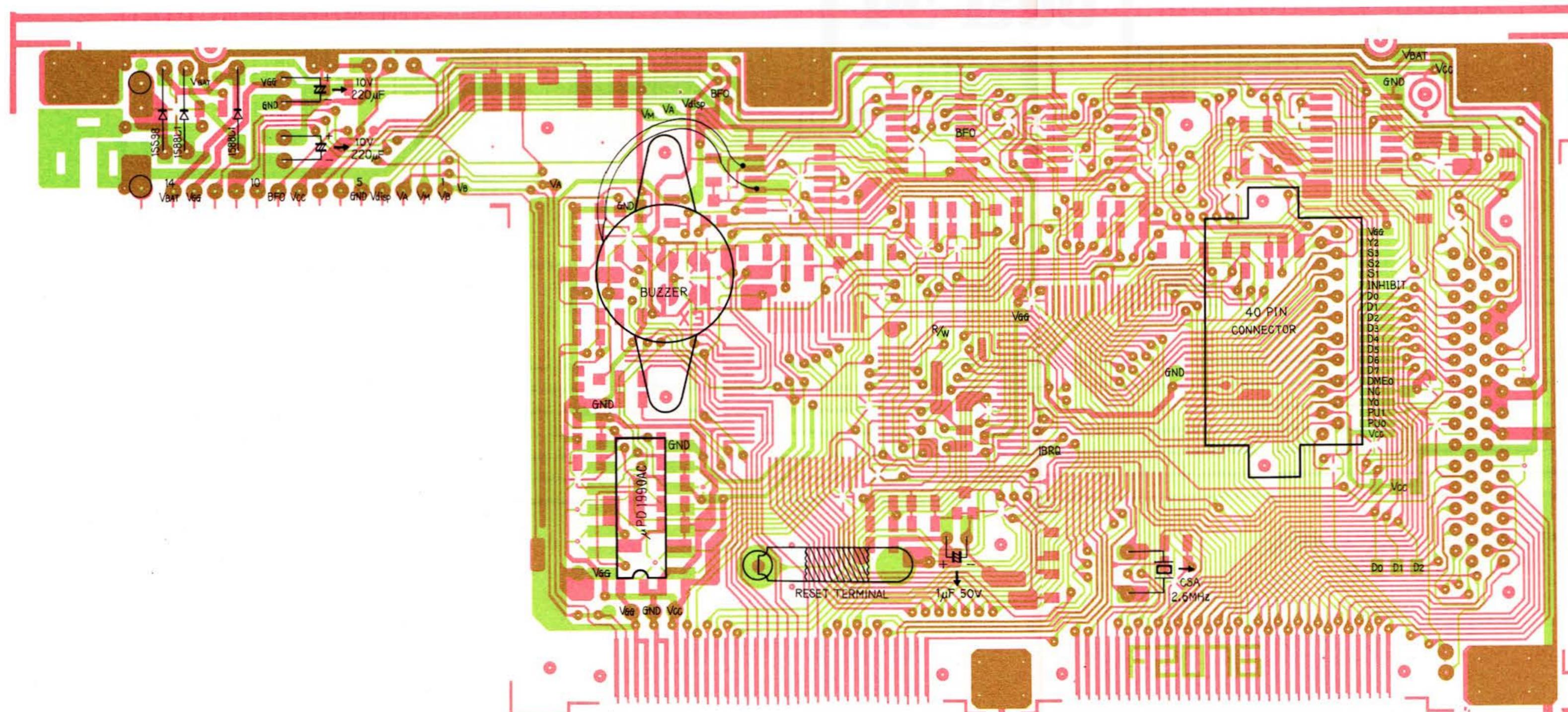
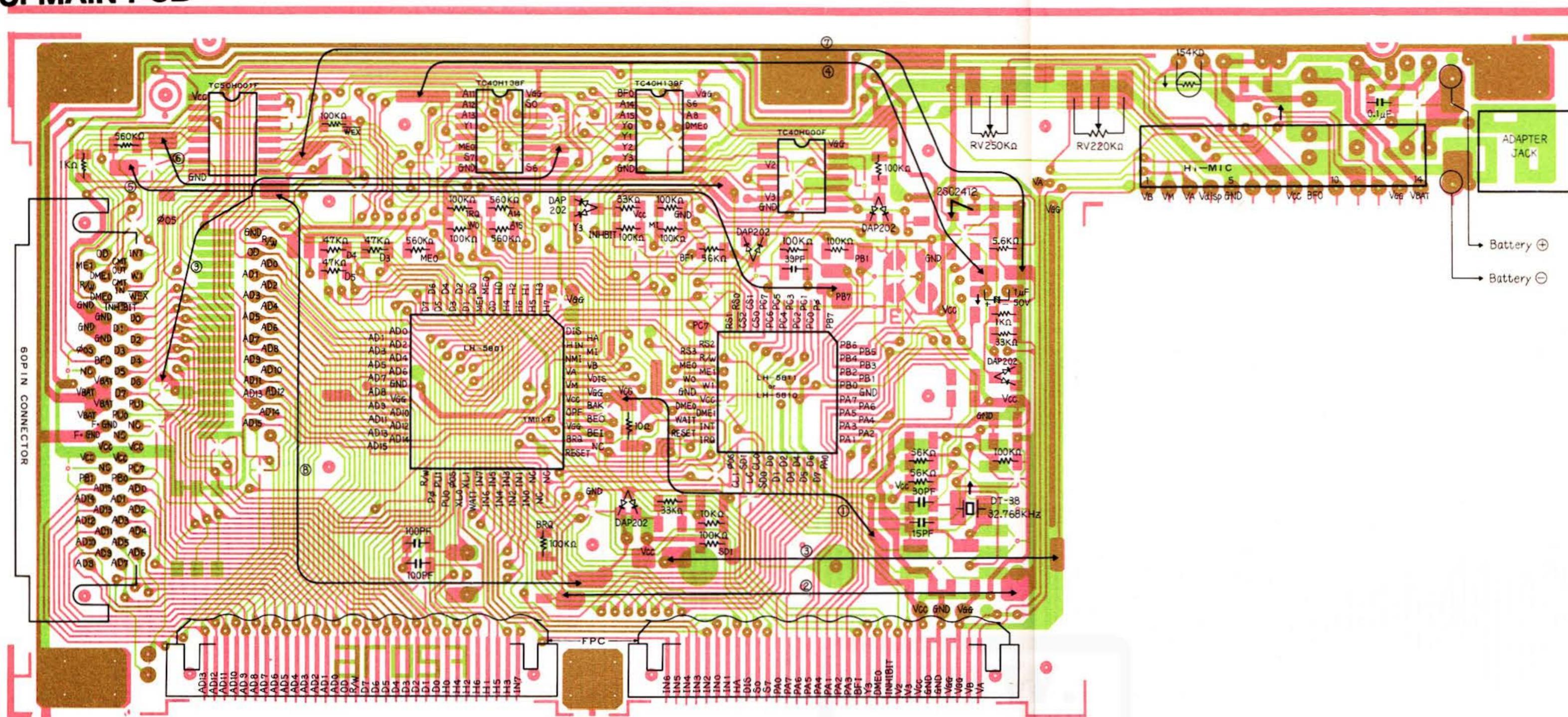




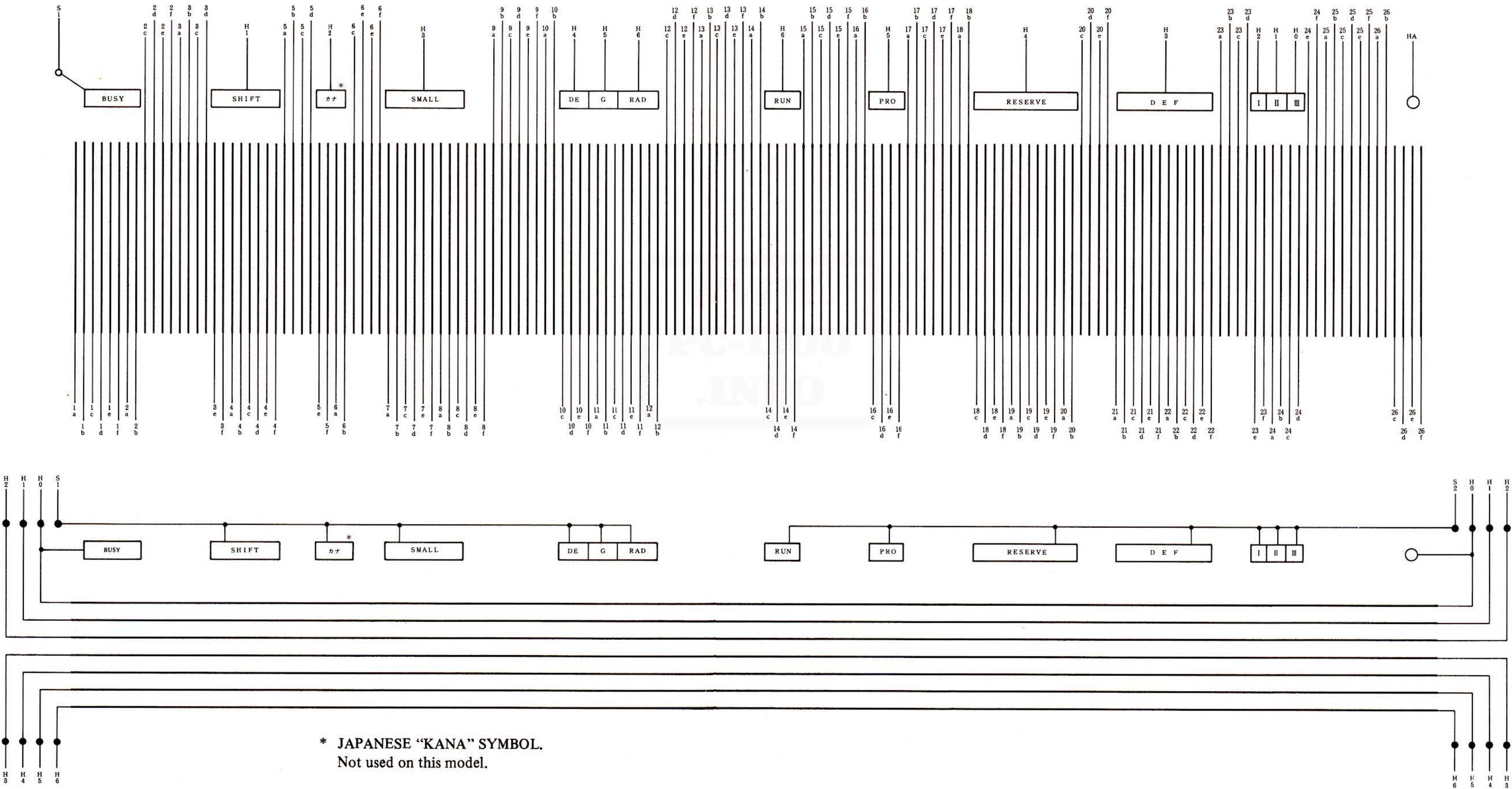
7. KEY P.C.B.



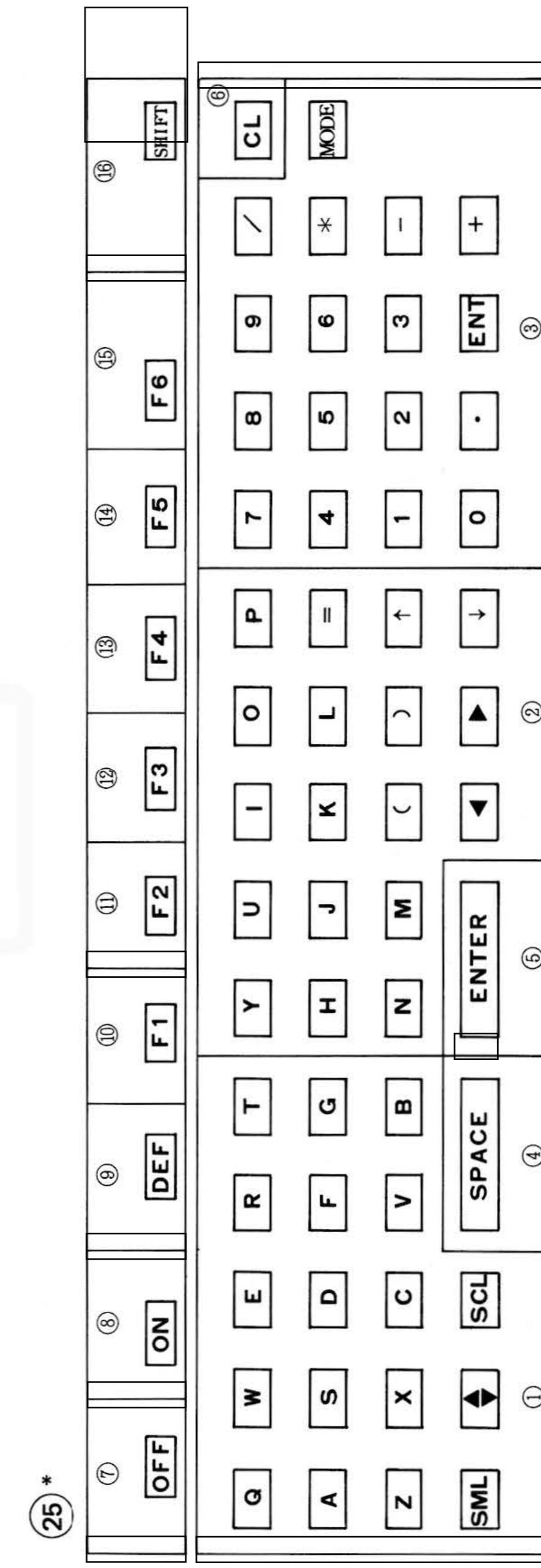
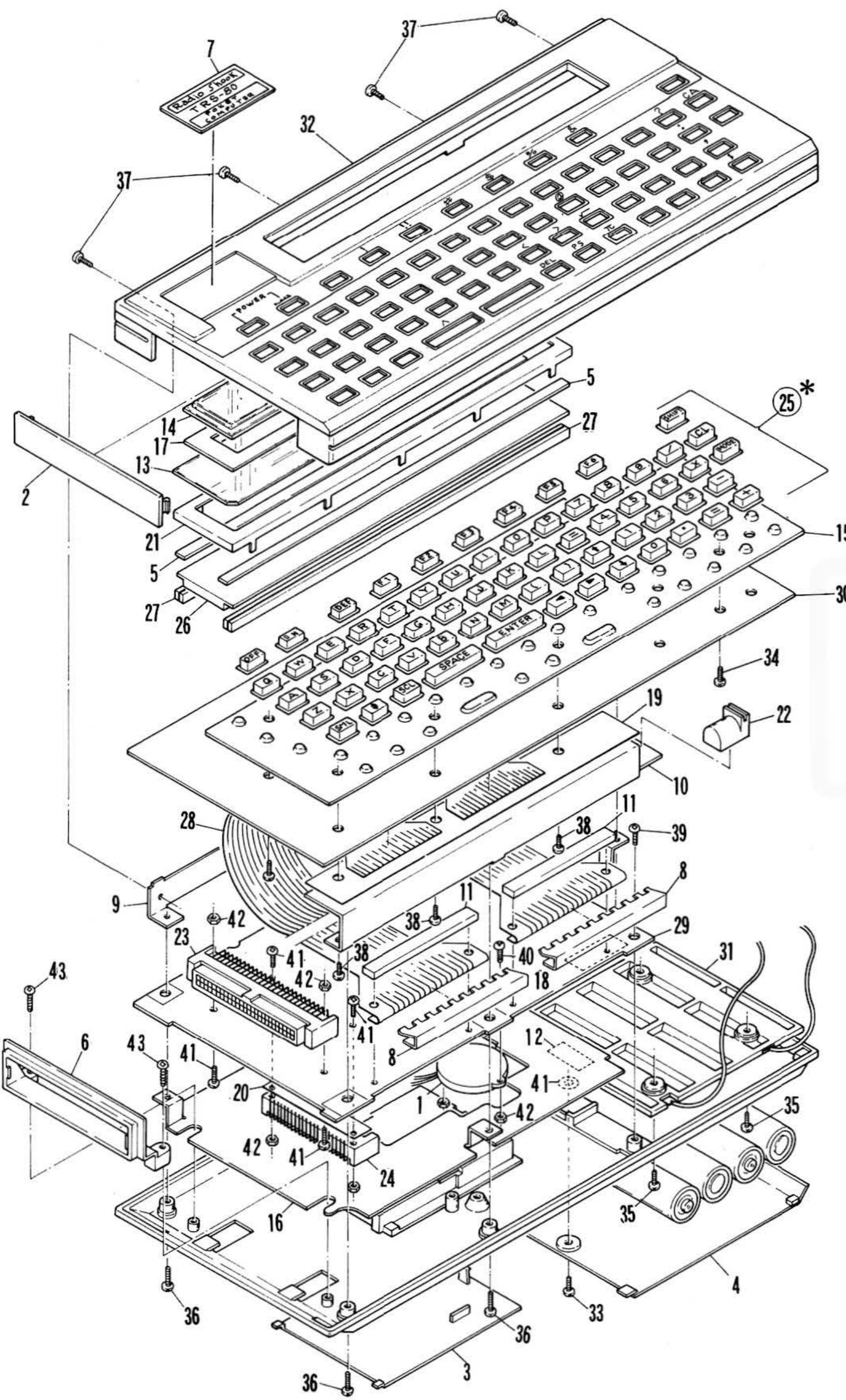
8. MAIN PCB



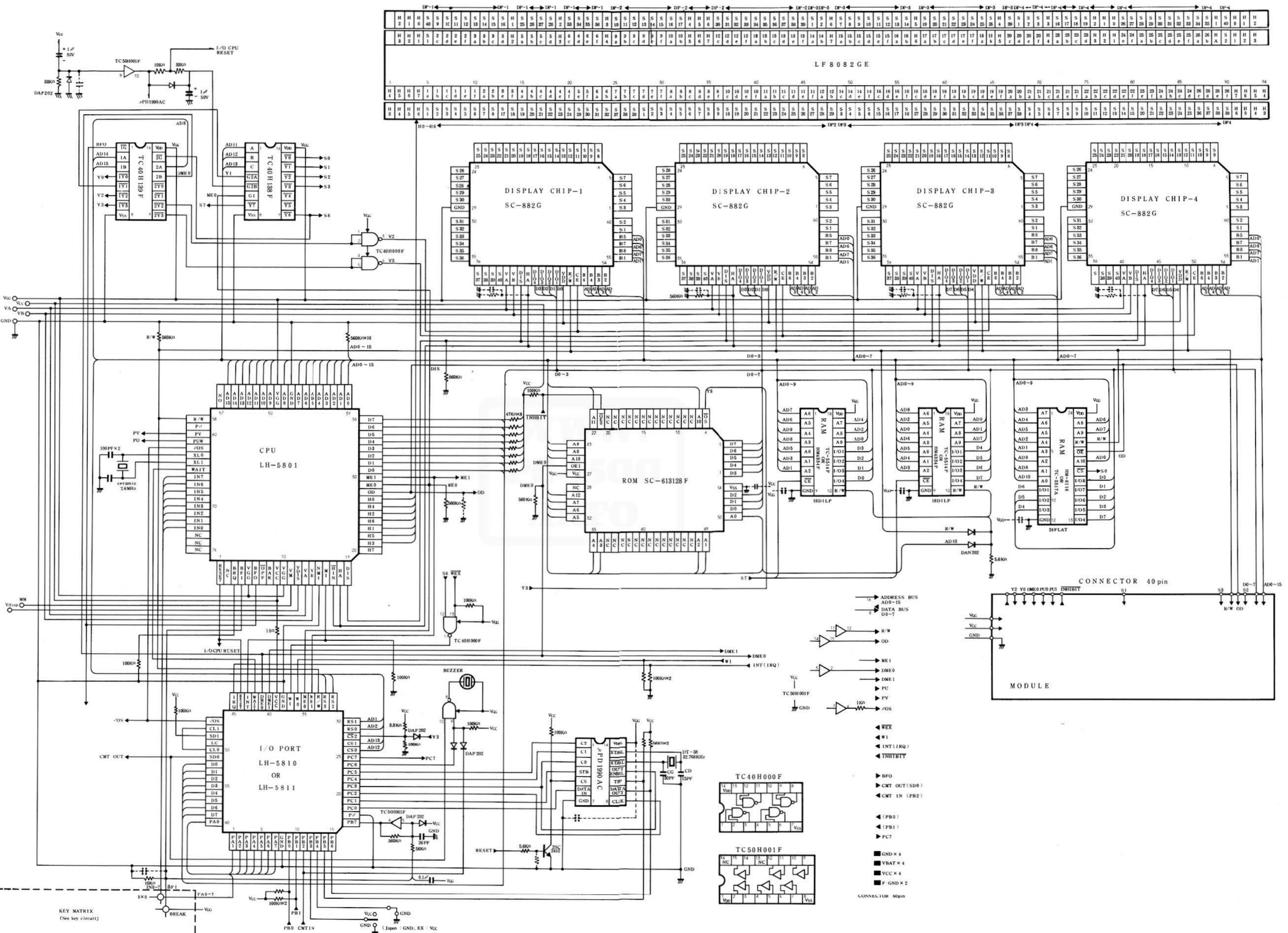
9. LCD SEGMENT & BACK PLATE SIGNAL



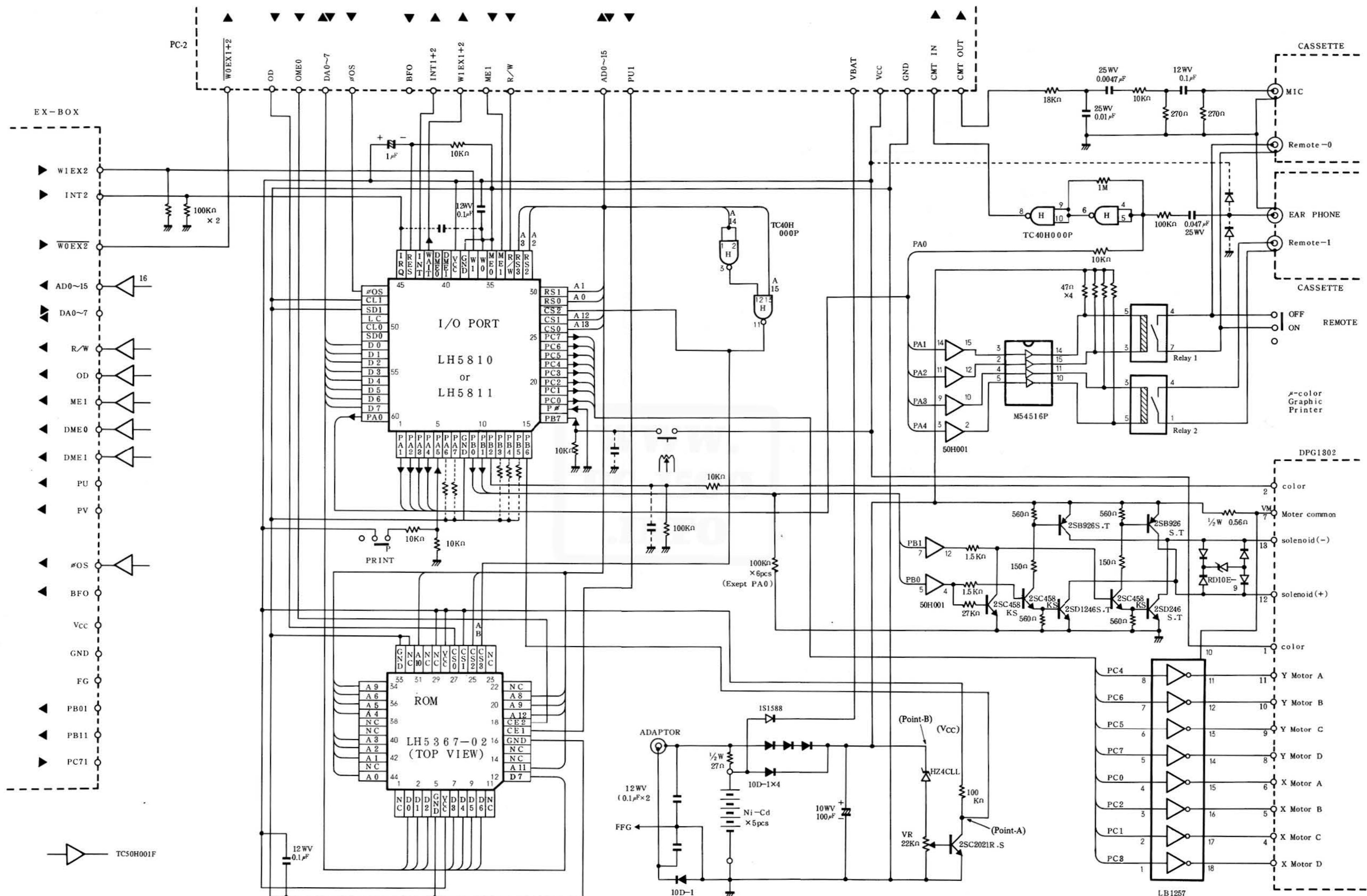
11. EXPLODED VIEW



12. MAIN CIRCUIT DIAGRAM

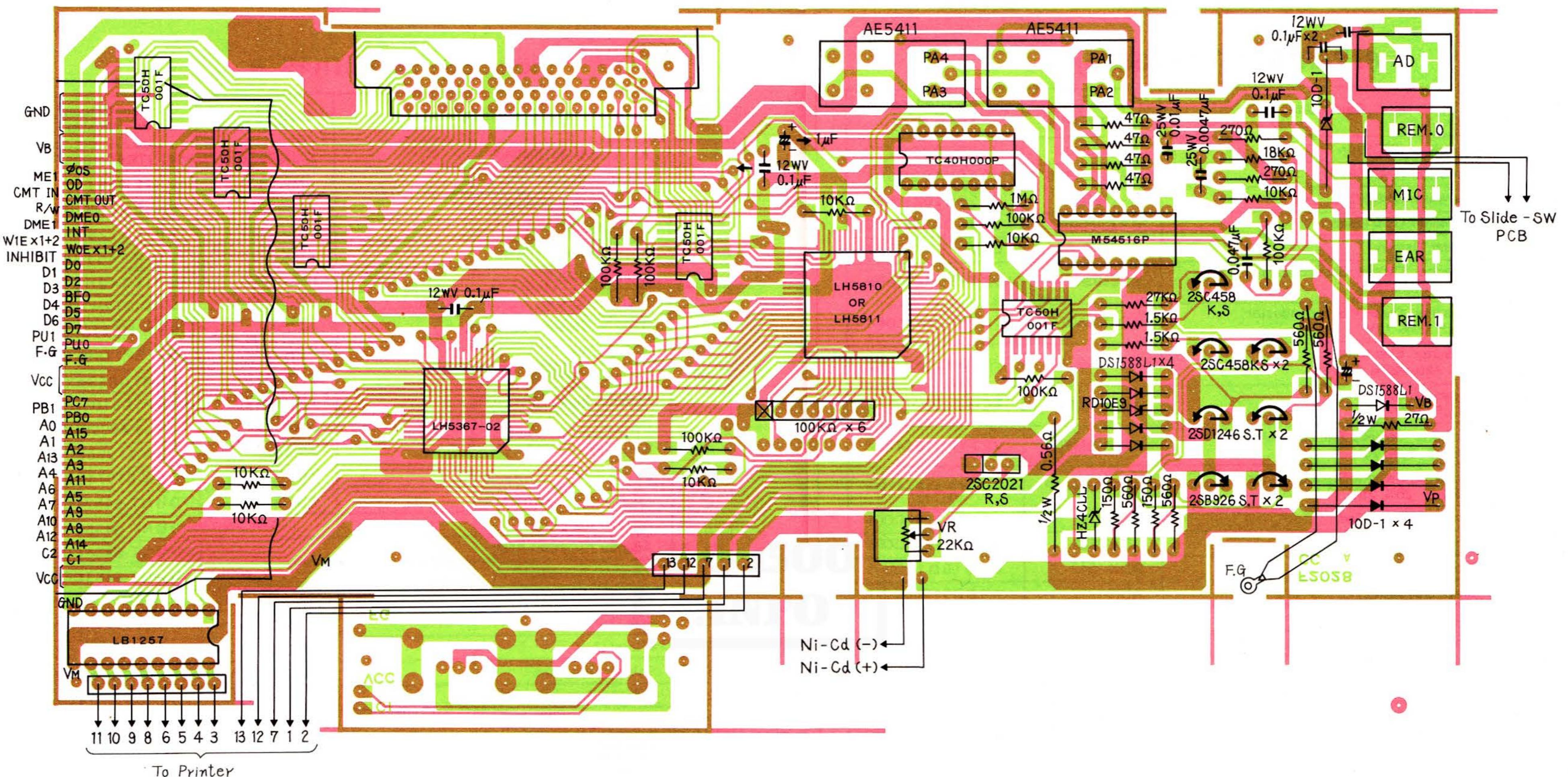


4. CIRCUIT DIAGRAM



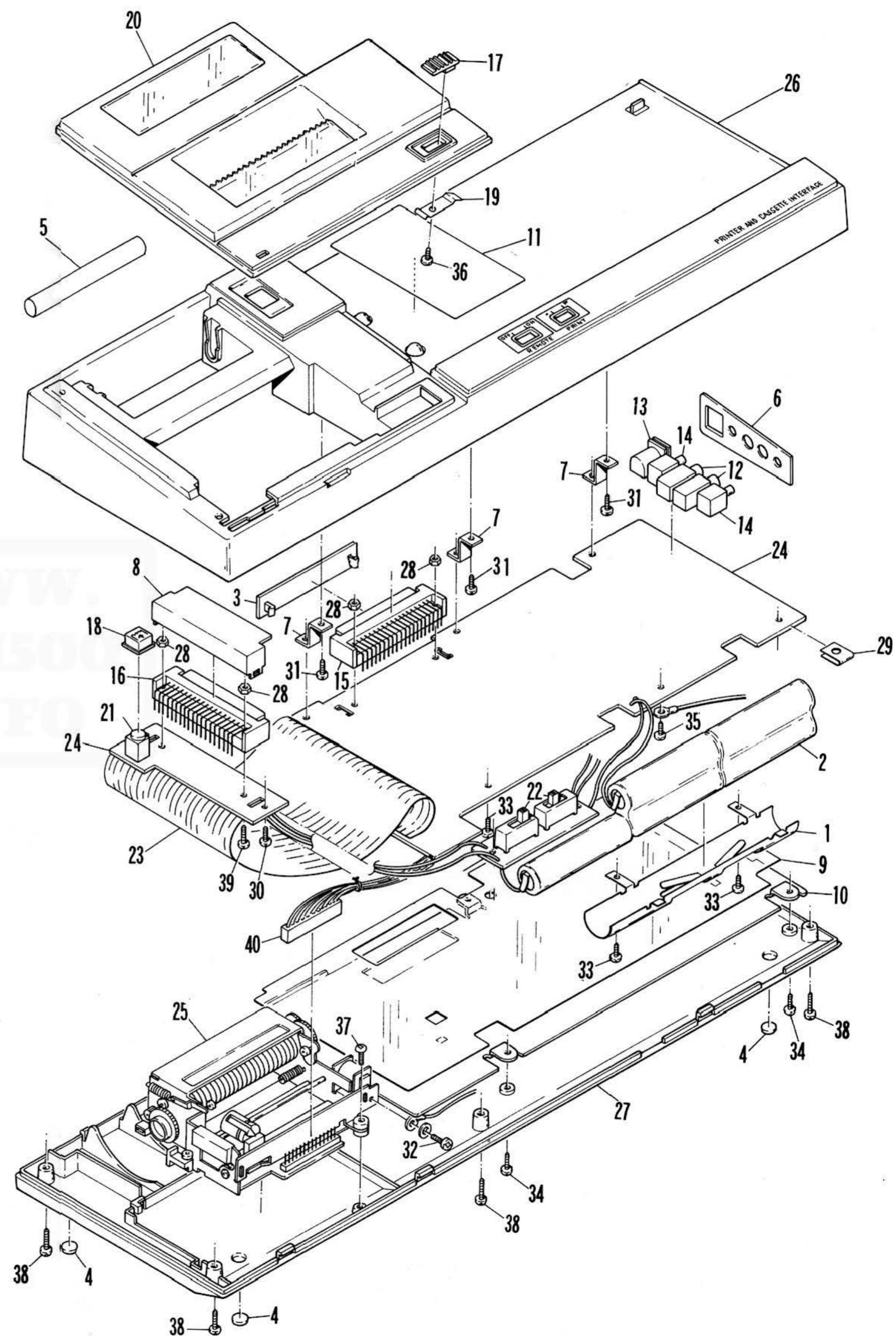
Point-A should be adjusted to become 1.0V by VR(22kΩ) with following condition
 1. Apply point-B at 4.81 ~ 4.85V, D.C
 2. Apply V_{CC} at 3.9 ~ 4.1V, D.C
 3. Temperature around 20°C

5. PARTS AND SIGNALS POSITION



7. EXPLODED VIEW

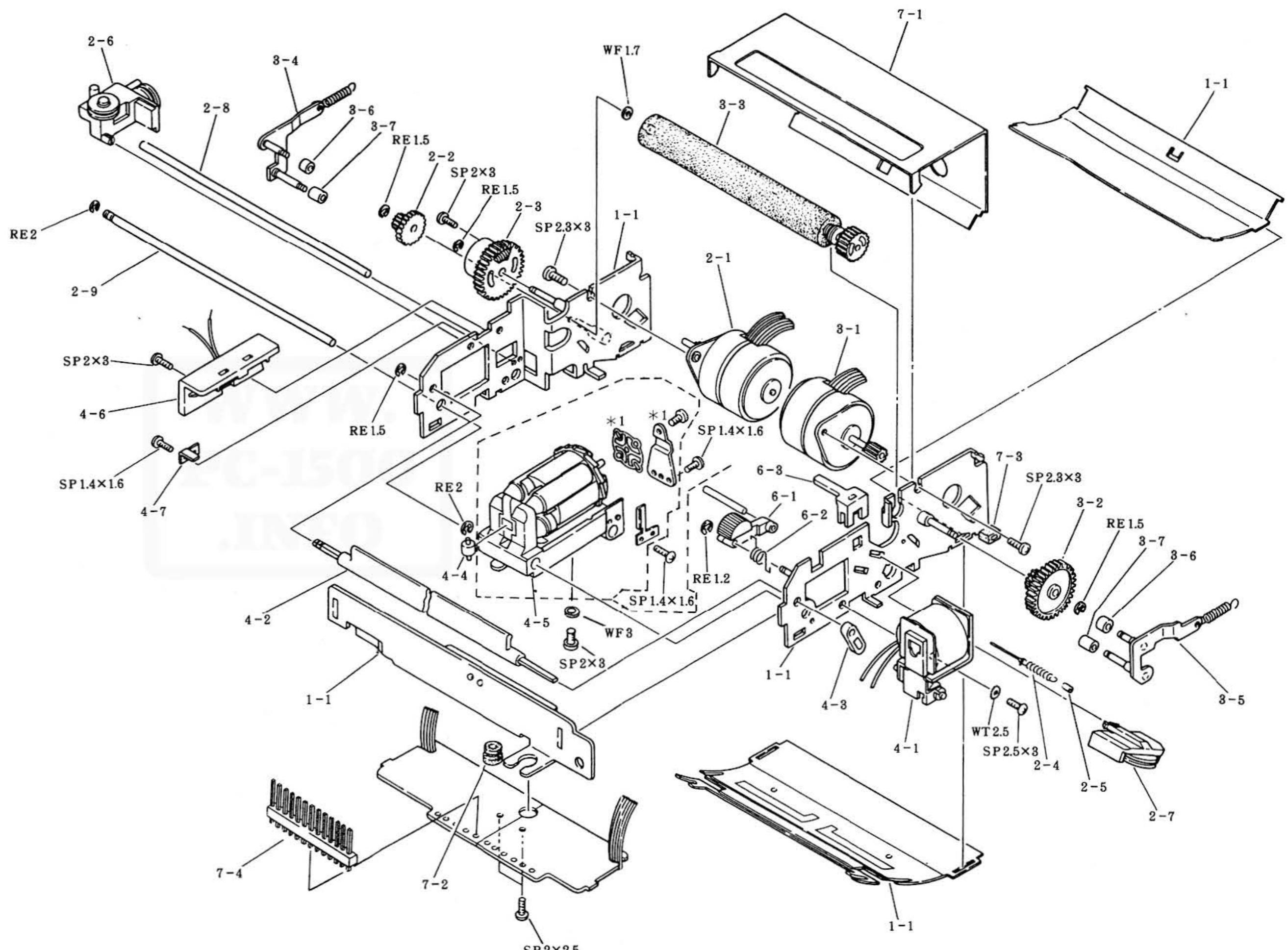
Ref. No.	Description	RS Location	Manufacturer Part Number
	Resistor, Carbon 1.5K ohm 5% 1/4W		VRD-ST2EY152J
	Resistor, Carbon 18K ohm 5% 1/4W		VRD-ST2EY183J
	Resistor, Carbon 270 ohm 5% 1/4W		VRD-2T2EY271J
	Resistor, Carbon 27K ohm 5% 1/4W		VRD-ST2EY273J
	Resistor, Carbon 47 ohm 5% 1/4W		VRD-ST2EY470J
	Resistor, Carbon 560 ohm 5% 1/4W		VRD-ST2EY561J



8. MECHANICAL PARTS LIST

Ref. No.	Description	RS Location	Manufacturer Part Number
1-1	Frame unit	AZ-6541	00PDG053////
2-1	X-motor	AM-4678	00PDG005////
2-2	X-idler gear	ARA-0395	00P07-0132-01
2-3	Bobbin gear unit	ARA-0396	00PDG013////
2-4	Wire unit	AW-2816	00PDG020////
2-5	Sleeve	AHC-1463	00P26-0001-01
2-6	Pulley unit (Left)	ARA-0397	00PDG015////
2-7	Pulley unit (Right)	ARA-0398	00PDG019////
2-8	Shaft A	AHC-1464	00P10-0335-01
2-9	Shaft B	ACH-1465	00P10-0336-01
3-1	Y-motor	AM-4679	00PDG006////
3-2	Y-idler gear	ARA-0399	00P07-0131-01
3-3	Rubber roller unit	ARA-0400	00PDG009////
3-4	Paper plate (Left)	ART-4128	00PDG010////
3-5	Paper plate (Right)	ART-4129	00PDG011////
3-6	Paper pressure roller A	ARA-0401	00P10-0297-01
3-7	Paper pressure roller B	ARA-0402	00P10-0296-01
4-1	Magnet unit	ART-4130	00PDG025////
4-2	Push lever unit	ART-4131	00PDG030////
4-3	Joint lever	AHC-1466	00P12-0147-01
4-4	Push roller	ARA-0403	00P10-0287-01
4-5	Slider head unit	AH-4483	00PDG059////
4-6	Reed SW. unit	AS-2755	00PDG063////
4-7	Pawl	ART-4132	00P19-0219-02
6-1	Pen eject lever	ART-4133	00PDG060////
6-2	Pen eject spring	ARB-7486	00P19-0214-02
6-3	Pen eject plate	AHC-1467	00P11-0062-02
7-1	Motor cover	ART-4134	00P11-0045-01
7-2	Rubber bushing	AHC-1468	00P23-0018-01
7-3	Rubber pad	AHC-1469	00P23-0017-01
7-4	Connector	AJ-7163	00P63-1002-01
	Washer (3.0)		00PCB68185///
	Washer (2.5)		00P24-0209-01
	Washer (1.7)		00P23-0025-01
	E-ring (2.0)		00P27-0003-19
	E-ring (1.5)		00P27-0002-19
	E-ring (1.2)		00P27-0001-19
	Screw (M2 x 3)		00P30-0308-00
	Screw (M2.3 x 3)		00P30-0408-00
	Screw (M2.5 x 3)		00P30-0507-00
	Screw (M1.4 x 1.6)		00P30-0904-01

9. EXPLODED VIEW



*1 PART IS INCLUDED IN 4-5

SHARP

ORDINATEUR DE POCHE

MODELE **PC-1500**

MANUEL D'INSTRUCTION



INTRODUCTION

Permettez-nous d'abord de vous remercier d'avoir choisi l'ordinateur de poche SHARP PC-1500. Nous sommes persuadés que vous aurez beaucoup de plaisir à vous servir quotidiennement de ce nouvel ami qui, quoique de petite taille, n'en est pas moins puissant. Le PC-1500 constitue l'un des modèles d'ordinateurs de poche les plus élaborés du monde. Bien qu'il ait beaucoup de points communs avec son cousin, l'ordinateur de poche SHARP PC-1211, le PC-1500 vous offre des capacités bien plus avancées, comme, par exemple:

- Un affichage LCD à points programmables de 7 à 156.
- Un générateur de sons destiné à créer des effets spéciaux par programme.
- Un jeu de caractères ASCII majuscules et minuscules.
- Fonctions scientifiques et mathématiques.
- Touches de fonction définissable par l'utilisateur.
- Une version étendue de BASIC fournissant des tableaux bidimensionnels, chaînes à longueur variable, commandes de graphismes, enchaînement de programmes et nombre d'autres caractéristiques avancées.
- Jusqu'à 4 K multiplets de RAM facultatif.
- Une Interface/Imprimante et Cassette facultative permettant le traçage X-Y en 4 couleurs, la mémorisation de données et de programmes et leur impression avec les caractères d'une taille au choix parmi les 9 disponibles.

Cet appareil est capable d'exécuter les multiples fonctions qui, il n'y a pas si longtemps encore, aurait rempli un entrepôt de tubes, de fils et d'ingénieurs. Mais il n'est pas nécessaire d'être ingénieur pour utiliser un appareil si élaboré. Bien au contraire, le PC-1500 et ce manuel ont été étudiés pour aider le débutant à accéder à cette nouvelle technologie.

Nous avons divisé ce manuel en cinq parties principales de manière à vous rendre compétent en très peu de temps. Les utilisateurs plus avancés peuvent explorer les possibilités du PC-1500 au moyen des parties "Programmation avancée", "Calculations avancées" et "Appendices".

Le langage de ce manuel est celui de la conversation et de nombreux exemples sont fournis. Mais ne vous fiez pas à nos promesses, passez tout de suite au chapitre 0 et vérifiez de par vous-même la facilité avec laquelle on peut débuter. Avant toute chose, assurez-vous que les piles ont été placées dans l'appareil. Si elles ne l'ont pas été, l'appendice B vous fournira les instructions nécessaires pour le faire.

Avant toute chose, amusez-vous bien et n'hésitez pas à expérimenter!

REMARQUES SUR LE FONCTIONNEMENT

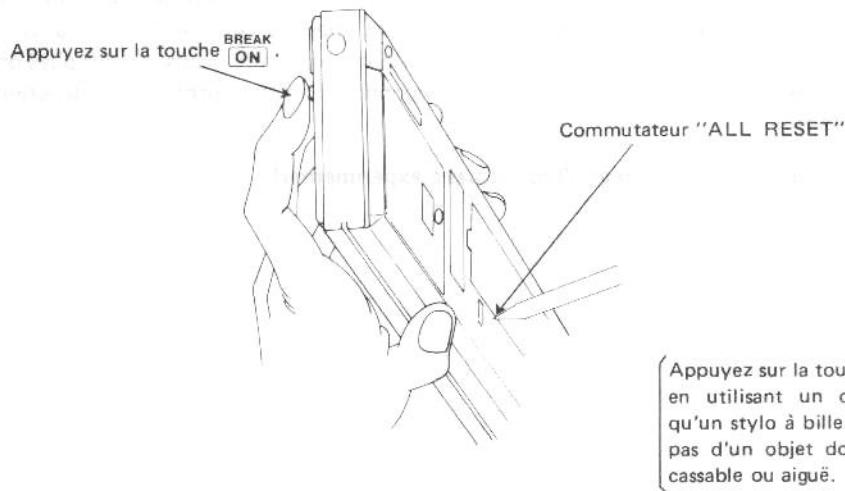
L'affichage LCD de l'ordinateur étant en verre, il est nécessaire de le manipuler avec quelques précautions. NE PLACEZ PAS le PC-1500 dans la poche arrière de vos pantalons, car vous risqueriez de l'abîmer quand vous vous asseyez.

Pour assurer une utilisation sans problèmes de votre ordinateur de poche SHARP, nous vous recommandons de:

1. Protéger l'ordinateur des changements extrêmes de températures, de l'humidité et de la poussière.
2. Nettoyer l'ordinateur avec un linge doux et sec. Evitez d'utiliser des solvants, de l'eau ou des linges mouillés.
3. Oter les piles, si vous savez que vous n'allez pas utiliser votre ordinateur pendant longtemps, afin d'éviter des dégâts provoqués par des fuites de liquide des piles.
4. Retourner l'ordinateur uniquement à un centre de réparations autorisé SHARP, quand cela s'avère nécessaire.
5. Garder ce manuel pour le cas où vous auriez besoin de vous y reporter dans le futur.

Localisation des Pannes

Des bruits extérieurs forts ou des chocs durant le fonctionnement de l'appareil peuvent rendre inopérables toutes les touches, y compris la touche **BREAK** **ON**. Au cas échéant, appuyez sur le commutateur ALL RESET situé sur le dos de l'appareil pendant environ 15 secondes tout en maintenant la touche **BREAK** **ON** en position basse.



Puis, après vous être assuré que l'affichage indique **NEWØ? : CHECK**, appuyez sur les touches **CL** **N** **E** **W** **Ø** **ENTER**. (Note: Ø=zéro, une touche numérique).

Si **NEWØ? : CHECK** n'est pas affiché, répétez les opérations précédentes. Evitez d'utiliser le commutateur ALL RESET hors des cas où les pannes décrites ci-dessus se produisent, car il effectue la destruction des programmes, et du contenu de la réserve.

SPECIFICATIONS DU PC-1500

Modèle:	Ordinateur de poche PC-1500
Nombres de chiffres de calcul:	10 chiffres (mantisse) + 2 chiffres (exposant)
Système de calcul:	Selon la formule mathématique (avec fonction déterminant les priorités)
Langage de programme:	BASIC
Capacité:	CPU: CMOS 8 bit Système ROM: 16 K multiplets Capacité de mémoire adressable (RAM): 3,5 K multiplets Système: 0,9 K multiplets Tampon d'entrée: 80 multiplets Pile: 196 multiplets Autres: Aire de l'utilisateur: 2,6 K multiplets Aire de mémoire fixe: 624 multiplets (A~Z, A\$~Z\$) Aire de données de programme de base: 1850 multiplets Aire de réserve: 188 multiplets
Calculs:	Quatre calculs arithmétiques, calcul de puissance, fonctions trigonométriques et trigonométriques inverses, fonctions logarithmiques et exponentielles, conversion d'angles, extraction de la racine carrée, fonction de signe, absolus, entiers et calculs logiques.
Fonction de mise au point:	Décalage de curseur (►◄) Insertion (INS) Elimination (DEL) Revue de ligne (↓, ↑)
Protection de la mémoire:	Unité CMOS de soutien de piles (protège les mémoires de programme, de données et de réserve)
Affichage:	Crystal liquide Largeur de 26 caractères Graphismes à points 7 x 156
Touches:	65 Touches, y compris: Alphabétiques Numériques de fonction définissable par l'utilisateur Pré-programmées
Source d'alimentation:	6,0V, c.c. 4 piles sèches de type UM-3, AA ou R6
Consommation de courant:	6,0V, c.c.: 0,13W
Durée de fonctionnement:	Environ 50 heures avec des piles sèches de type UM-3, AA ou R6
Température de fonctionnement:	0° à ~40°C
Dimensions:	Longueur: 195 mm, largeur: 86 mm, épaisseur: 25,5 mm
Poids:	Environ 375 g (avec piles)
Accessoires:	Housse, quatre piles sèches, deux gabarits de clavier, étiquette pour le nom du propriétaire et manuel d'instruction
En option:	Interface imprimante/cassette (CE-150) Module d'expansion de mémoire (CE-151 du type à prise et RAM à 4K multiplets)

TABLE DES MATIERES

	Page
Introduction	1
Remarques sur le Fonctionnement	2
Spécifications du PC-1500.	3
Table des matières	4
O. Programmation instantanée	8
A. Exemple 1	8
B. Exemple 2	10
I. Faisons connaissance	12
A. Touches ON et OFF (marche et arrêt)	12
B. Touches alphabétiques	12
C. Touches numériques	12
D. Touche SHIFT	13
E. Les minuscules et la touche SMALL	13
F. L'affichage	13
G. Le curseur et le symbole de guidage	14
H. La touche de remise à zéro	14
I. Touche ENTER	14
J. Les messages d'erreur	14
K. L'indicateur de charge des piles	15
II. Plonger dans l'eau froide	16
A. La touche MODE	16
B. Calculs simples	16
C. Calculs en séries	17
D. Calculs avec des nombres négatifs	17
E. Calculs composés	18
F. L'utilisation des parenthèses	18
G. Comparaisons logiques et fonctions logiques	19
H. Les touches et les caractéristiques de mise en page	20
H.1. Touche flèche vers la gauche/DEL (effacement)	20
H.2. Touche flèche vers la droite/INSérer	21
H.3. Fonction de rappel	22
I. Les variables	22
J. Une pause	25
RESUME	26
III. L'art mystérieux de la programmation	27
A. Qu'est-ce qu'un programme?	27
B. Comment bâtir un programme?	28
C. COMMANDES et INSTRUCTIONS	29
D. Numéros de ligne.	29
E. Touches de revue des lignes de programme	30
F. Un examen plus détaillé de certains vieux amis	30
F.1. La commande NEW	30
F.2. L'instruction LET	31
F.3. L'instruction PRINT	31

G.	L'instruction PAUSE	36
H.	L'instruction INPUT	37
I.	Raccourcis et conseils utiles.	42
	I.1. Abréviations.	42
	I.2. Instructions multiples utilisant le double-point	43
J.	La correction d' <u>erreurs</u> sur le mode PROgramme	44
K.	La commande LIST	45
L.	Plus on est de fous, plus on rit	46
	L.1. L'instruction END	46
	L.2. Numéro des lignes RUN	46
M.	Instructions de contrôle	47
N.	IF THEN	47
O.	GOTO	48
P.	FOR NEXT	54
Q.	WAIT	57
R.	READ, DATA, RESTORE	58
S.	REM	61
T.	GOSUB et RETURN	61
	Résumé des caractéristiques de mise en page sur le mode PROgramme.	63
IV.	Calculs avancés	63
A.	Notation Scientifique	63
B.	Gamme des calculs: débordement supérieur et inférieur	65
C.	Les racines, les puissances et pi.	66
D.	Les modes angulaires	68
E.	Les fonctions trigonométriques	68
	SIN, COS, TAN, ASN, ACS, ATN	
F.	Les fonctions logarithmiques	69
	LN, LOG, EXP	
G.	Conversion d'angle.	70
	DEG, DMS	
H.	Fonctions diverses	70
	ABS, INT, SGN	
V.	Programmation avancée	71
A.	Tableaux et variables indicées	71
	DIM	
B.	Plus de détails sur les chaînes des caractères	74
	B.1. Chaînes bidimensionnelles	74
	B.2. Enchaînement	75
	B.3. Comparaison de chaînes	76
C.	Fonctions	77
	C.1. ASC	77
	C.2. CHR\$	78
	C.3. INKEY\$	79
	C.4. LEN	79
	C.5. LEFT\$	80
	C.6. MID\$	80
	C.7. RIGHT\$	81
	C.8. RND	82
	C.9. RANDOM	82
	C.10. STR\$	83

C.11. STATUS	83
C.12. TIME	84
C.13. VAL	84
D. PRINT USING	85
Tableau des caractères de mise en forme	86
E. Transfert de contrôle calculé	88
ON GOTO, ON GOSUB, ON ERROR GOTO	88,89
F. Programmation de l'affichage	90
F.1. BEEP	90
F.2. CURSOR	91
F.3. CLS	93
F.4. GCURSOR	94
F.5. GPRINT	96
F.6. POINT	99
G. Détection d'erreurs	101
TRON, TROFF, touches à flèches	101
H. Nombres hexadécimaux et fonctions de BOOLE	103
H.1. Spécification hexadécimale	103
H.2. La Fonction AND (et)	103
H.3. La Fonction OR (ou)	104
H.4. La Fonction NOT	104
I. Pour arrêter l'exécution d'un programme	105
STOP, CONT	105
J. Contrôle du mode	105
LOCK, UNLOCK (Bloquer, débloquer)	105
VI. Elargissement du PC-1500	106
A. L'INTERFACE IMPRIMANTE/CASSETTE	106
A.1. Connexion de l'ordinateur sur l'interface	106
A.2. Alimentation	108
A.3. Raccordement d'un magnétophone avec l'interface	108
A.4. Chargement du papier	110
A.5. Remplacement des stylos	111
B. Utilisation du magnétophone à cassette	113
B.1. Opération du magnétophone	113
B.2. Enregistrement des programmes sur bande magnétique (CSAVE)	114
B.3. Rechargement de programmes enregistrés sur bande magnétique (CLOAD, CLOAD?).	114
B.4. Enregistrement et recharge de données sur et à partir de bandes magnétiques (PRINT#, INPUT#)	115
B.5. Mise en page de programmes sur bande magnétique (MERGE)	117
B.6. Enchaînement de programmes (CHAIN)	118
B.7. Utilisation de deux magnétophones.	119
C. Utilisation de l'imprimante	121
C.1. Spécifications de l'imprimante CE-150	121
C.2. La commande TEST	121
C.3. L'impression de calculs	122
C.4. Modes de l'imprimante	123
C.5. Listage du programme	123
C.6. Contrôle programmable de l'imprimante	125
C.6.1. CSIZE	125

C.6.2. ROTATE	125
C.6.3. COLOR	126
C.6.4. LF	126
C.6.5. LPRINT	127
C.6.6. LCURSOR	129
C.6.7. TAB	129
C.6.8. SORGN	129
C.6.9. GLCURSOR	130
C.6.10. LINE	130
C.6.11. RLINE	132
VII. Le mode RESERVE	135
A. Définition et sélection des touches de réserve	135
B. Identification des touches de réserve	136
C. Effacement des programmes de réserve	137
VIII. Démarrage de l'exécution du programme	137
A. La touche DEF	137
A.1. Passages des programmes DEFinissables	137
A.2. Mots-clés pré-affectés	138
A.3. L'instruction AREAD	139
B. Démarrage automatique de programme ARUN	140
C. Comparaison des méthodes de démarrage	140
C.1. L'aire de mémoire fixe	140
C.2. Tableau de comparaison des méthodes de démarrage de programme	141
IX. Appendices	143
A. Abréviations	144
B. Remplacement des piles du PC-1500	148
C. Séquence de collation	150
E. Messages d'erreurs	151
F. Lectures supplémentaires	157
O. Ordre d'évaluation d'expressions	157
X. Différences entre les commandes du PC-1211 et celles du PC-1500	159
Commandes particulières AU PC-1500	160
Z. Table de référence des commandes	161

Etiquette

Ecrivez votre nom sur l'étiquette ci-jointe et collez-la sur le dos de l'ordinateur.

PROGRAMMATION INSTANTANEE

(Pas besoin d'ajouter de l'eau)

Cette partie est destinée uniquement à une minorité (les auteurs inclus) dont la curiosité l'emporte sur la patience (et peut-être même le bon sens). A ceux parmi vous qui veulent absolument FAIRE quelque chose avec ce produit miraculeux de l'électronique moderne, nous allons présenter un exercice de programmation simple. (AVERTISSEMENT aux timides et aux craintifs: passez au Chapitre 1; "Faisons connaissance", pour une introduction plus tranquille et minutieuse au SHARP PC-1500.)

Avant de continuer, nous jugeons nécessaire de vous avertir qu'il est important de passer les étapes dans l'ordre donné. Contrairement à ce que beaucoup de gens pensent, les ordinateurs ne sont pas des "super-cerveaux." Ils ne peuvent pas deviner ce que vous désirez comme le peut le cerveau humain moyen. Le PC-1500 attend simplement vos ordres et les exécute.

Etes-vous prêts? Bon! On commence!

Exemple 1

Trouvez d'abord la touche "ON" située dans le coin supérieur droit du clavier. En appuyant sur cette touche, vous réveillez le génie électronique qui sommeille (ne vous attendez pas quand même à voir jaillir une colonne de fumée!). La partie affichage de l'ordinateur devrait correspondre à l'illustration ci-dessous:



Appuyez sur la touche **MODE** située à l'extrême droite jusqu'à ce que l'abréviation **PRO** apparaisse dans le coin supérieur droit de l'affichage. (Si vous avez pressé la touche trop souvent, pressez-la de nouveau jusqu'à ce que vous obteniez le résultat désiré). Le SHARP PC-1500 est maintenant prêt à accepter les séries de directives qui constituent un programme d'ordinateur.

Appuyez sur les touches suivantes dans l'ordre donné:

1 Ø A = 1 ENTER

Remarquez que, lorsque vous appuyez sur la touche **ENTER**, l'ordinateur modifie ce que vous avez entré. L'affichage devrait maintenant être comme ci-dessous:



Note: Tout au long de ce manuel nous utiliserons le symbole Ø pour le zéro de façon à le distinguer plus aisément de la lettre O.

Pour continuer, appuyez maintenant sur les touches indiquées ci-dessous. Ne vous effrayez pas de voir chaque ligne disparaître au fur et à mesure que vous écrivez la suivante.

2 Ø P A U S E A ENTER
3 Ø A = A + 2 ENTER
4 Ø G O T O 2 Ø ENTER

A ce point-ci, vous aurez terminé votre premier programme. Maintenant il s'agit de commander à l'ordinateur "d'exécuter" les ordres qu'il a emmagasinés. On appelle cette opération l'"exécution" et on l'accomplit sur le mode RUN. Appuyez de nouveau sur la touche **MODE** et les lettres RUN remplaceront les lettres PRO au sommet de l'affichage.

Une dernière étape: écrivez les lettres **R** **U** **N** et appuyez sur **ENTER**.

Félicitations! Votre premier programme BASIC est maintenant en cours d'exécution. L'ordinateur suit vos directives et dresse la liste de tous les nombres impairs positifs, dans l'ordre.

"Ah! Je suis génial!" Mais... "quand est-ce que ça va s'arrêter?"

Hé bien . . . malheureusement, ce programme ne se terminera pas, à moins que vous n'interveniez ou que les piles ne se déchargent. Revoyons le programme pour en comprendre la raison:

```
10 A = 1
20 PAUSE A
30 A = A + 2
40 GOTO 20
```

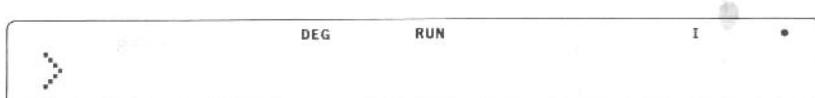
L'action de la ligne numéro 40 est de faire reproduire par l'ordinateur toutes les lignes qui suivent la ligne 20. Cela inclut, bien sûr, la ligne 40 qui, comme nous le savons déjà, ordonne à l'ordinateur de re-reproduire les lignes 20, 30 et 40 qui . . . et cela à perpétuité. Cette répétition sans fin s'appelle "une boucle" dans le jargon des ordinateurs.

Notre programme est bloqué dans une "boucle infinie" et vous, vous seulement, avez le pouvoir de mettre fin à ce tragique gaspillage de courant. Pour ce faire, appuyez sur la touche **ON**. Comme le PC-1500 est déjà en état de marche, vous avez en fait mis en branle la fonction BREAK (briser). Inutile de vous mettre à couvert! Contrairement à ce que ce mot semble indiquer, ce n'est pas une touche d'auto-destruction! Pour mettre les choses au clair, vous ne pouvez en aucun cas endommager l'ordinateur en appuyant simplement sur des touches. N'hésitez donc pas à expérimenter!

Après avoir appuyé sur la touche BREAK, un message semblable à celui ci-dessous apparaîtra sur l'affichage:



Ceci vous apprend quelle instruction était en cours d'exécution quand vous avez interrompu l'ordinateur. Appuyez à nouveau sur la touche BREAK et l'ordinateur sera prêt à accepter d'autres instructions:



Pour ceux d'entre vous qui viennent de se souvenir qu'il est presque l'heure de rentrer pour dîner, le moment est bon pour s'arrêter. (Avant de partir et pour économiser le courant des piles, appuyez sur la touche **OFF** s'il vous plaît). Les autres qui sont déjà en train de devenir des mordus de la programmation veulent probablement continuer leur éducation à l'aide du deuxième exemple. (Nous nous refusons cependant de prendre la responsabilité des conséquences d'un dîner manqué ou d'un rendez-vous oublié.)

Exemple 2

Pour débuter notre second programme, il est nécessaire d'entrer sur le mode de programmation. Pour ce faire, appuyez sur la touche MODE jusqu'à ce que les lettres PRO (abréviation de programme) remplacent les lettres RUN au sommet de l'affichage. Le PC-1500 va nous permettre maintenant d'entrer un nouveau programme ou de modifier l'ancien. Puisque notre nouveau programme ne va pas bâti sur les fondations de l'ancien, nous devons d'abord effacer les instructions de ce dernier de la mémoire de l'ordinateur. Pour accomplir cette tâche, écrivez d'abord le mot NEW sur la machine et appuyez sur la touche **ENTER**. Après un court moment le symbole de guidage > va réapparaître.

Ecrivez les caractères suivants pour entrer la première ligne du programme:

1	Ø	I	N	P	U	T	SHIFT	"	L	O	N	G	U	E	U	R	SPACE
L	I	S	T	E	SHIFT	/	SHIFT	"	SHIFT	+	A	ENTER					

Remarquez que lorsque vous appuyez sur la touche **SHIFT** avant d'appuyer sur la touche caractère au haut de laquelle est inscrit un autre caractère, le caractère du haut sera entré. La touche **SHIFT** (de décalage) permet d'utiliser la même touche pour exprimer deux fonctions différentes et c'est pour cela qu'on la nomme parfois touche de "seconde fonction". Ainsi, par exemple, dans la première ligne de notre programme (ligne 10 ci-dessus), quand on a appuyé sur la touche **SHIFT** avant d'appuyer sur la touche marquée **+**, le **:** (point-virgule) sera entré. La totalité de la ligne sera entrée dans l'ordinateur de la façon suivante:

Dans ce manuel-ci, nous illustrerons la sélection du caractère de seconde fonction à l'aide de la touche de décalage (SHIFT) et du caractère désiré. Par exemple, la ligne 10 ci-dessus va être affichée de la manière suivante:

1	Ø	I	N	P	U	T	SHIFT	"	L	O	N	G	U	E	U	R	SPACE
L	I	S	T	E	SHIFT	?	SHIFT	"	SHIFT	:	A	ENTER					

Complétez l'entrée de notre second programme en appuyant sur les touches indiquées ci-dessous:

2	Ø	I	F	A	SHIFT	<	=	Ø	G	O	T	O	9	9	ENTER	
3	Ø	F	O	R	I	=	1	T	O	A	ENTER					
4	Ø	P	A	U	S	E	I	SHIFT	,	I	*	I	ENTER			
5	Ø	N	E	X	T	I	ENTER									
9	9	B	E	E	P	2	SHIFT	:	E	N	D	ENTER				

Le PC-1500 a maintenant enregistré notre deuxième programme. Vous rappelez-vous encore quelle est l'opération suivante? Si vous répondez: "Exécuter le programme", vous êtes en voie de devenir un as de la programmation.

Retournez sur le mode RUN (Psst! utilisez la touche MODE!) et écrivez les lettres RUN. Appuyez sur **ENTER** pour commencer l'exécution de notre second programme.

Est-ce que l'ordinateur vous interroge à l'aide des caractères suivants? (s'il n'en est pas ainsi, retournez sur le mode PROgramme et re-vérifiez ce que vous avez écrit.)

LONGUEUR LISTE?...

Très bien! Notre programme demande à l'utilisateur (c'est vous) les renseignements nécessaires à l'exécution de la tâche que nous, en tant que programmeurs, lui avons ordonné d'exécuter. Rappelez-vous la première ligne de notre programme BASIC que vous avez eu la bonté d'entrer:

10 INPUT "LONGUEUR LISTE?"

L'ordinateur est actuellement en train de suivre les instructions emmagasinées dans cette ligne. Il attend maintenant que vous entrez (écriviez) certaines données.

Comme ce programme va imprimer une liste de nombres et de leurs carrés (le nombre multiplié par lui-même), l'ordinateur vous demande donc d'abord de lui indiquer combien de nombres et de carrés il doit imprimer (LONGUEUR LISTE?). L'utilisateur (C'est-à-dire vous) répond en écrivant un certain nombre et en appuyant sur (vous l'avez deviné!) la touche **ENTER**.

Ecrivez **8** et appuyez sur **ENTER**.

Regardez attentivement le défilement des nombres sur l'affichage. Le deuxième nombre de chaque paire (celui de droite) constitue le carré du premier. En tout, 8 paires vont apparaître sur l'affichage, car c'est ce nombre que vous avez demandé en répondant à l'ordinateur.

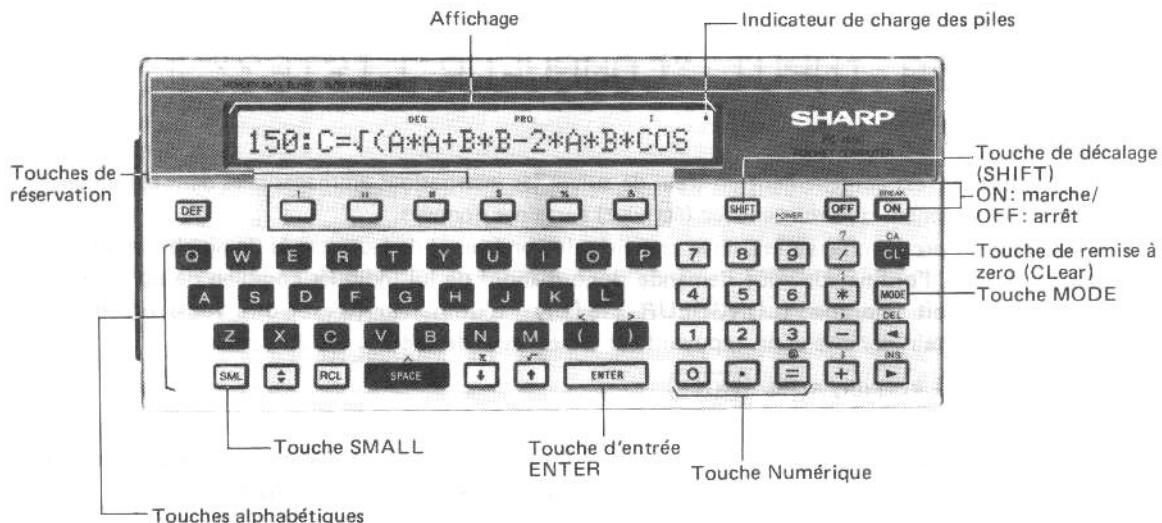
Quand le symbole de guidage réapparaît, faites passer à nouveau le programme (en écrivant RUN et en appuyant ensuite sur **ENTER**) et répondez différemment à la question "LONGUEUR LISTE?". Effectuez l'exécution du programme plusieurs fois, en changeant chaque fois la réponse. Vous devez vous rendre compte maintenant d'un des grands avantages des ordinateurs: ils peuvent répéter la même ennuyeuse opération en la variant légèrement à chaque fois en fonction de l'entrée.

Répétez une fois encore l'opération, mais entrez un zéro cette fois en réponse à la question "LONGUEUR LISTE?". Que se passe-t-il? Le programme s'arrête sans fournir de liste. Bien que cela puisse sembler bizarre, l'ordinateur ne fait que suivre vos directives.

Cette expérience nous montre l'utilité de cet outil qu'est l'ordinateur. On peut le programmer de manière à ce qu'il suive différentes séries d'instructions et traite les différentes données qu'on lui soumet. Du fait des instructions contenues dans la ligne 20, si l'utilisateur entre zéro (ou moins), l'ordinateur omettra de traiter la liste et sautera directement à la fin du programme. Actuellement, il s'est basé sur les demandes de l'utilisateur pour prendre cette décision. En tant que programmeur vous pouvez décréter quelles décisions sont possibles et quand elles sont possibles. Cela signifie que toutes les facultés de l'ordinateur sont à portée de votre main pour vous aider à résoudre vos problèmes particuliers de la manière qui vous convient le mieux.

I. FAISONS CONNAISSANCE

Après avoir déballé votre ordinateur de poche SHARP (nous l'appellerons le SHARP à partir de maintenant) et l'avoir admiré pendant un moment, vous allez probablement vous demander: "Qu'est-ce que c'est que ça? et ceci?" Regardons-le ensemble.



Nous allons décrire l'affichage dans un moment. D'abord, avant de mettre le SHARP en marche, veuillez considérer les plus importants éléments du clavier:

A. Touches ON et OFF (Marche et arrêt)

La fonction de ces touches est évidente: elles servent à mettre l'appareil en marche ou à l'arrêt. Si vous oubliez d'appuyer sur la touche OFF et que vous ne touchez pas l'ordinateur pendant plus de sept minutes, il s'éteindra automatiquement à moins qu'un programme ne soit en cours d'exécution. Remarquez le mot BREAK inscrit juste au-dessus de la touche **ON**. Cela veut dire que la touche **ON** peut servir à interrompre, briser (BREAK) l'exécution d'un programme. L'usage de cette fonction sera expliqué plus loin de manière détaillée.

B. Touches alphabétiques

Ces touches permettent à l'utilisateur (vous) de donner des instructions à l'ordinateur ou d'y entrer des données. De plus, ces touches servent aussi à désigner des "zones de stockage" à l'intérieur de la mémoire de l'ordinateur, zones dans lesquelles vous pouvez stocker des données et les en extraire à nouveau. Nous traiterons de cette utilisation dans la partie du manuel consacrée aux variables. Les touches **SHIFT** et **SML** (SMALL), qui vont être décrites plus loin, permettent d'écrire en minuscules.

C. Touches numériques et touches d'opérations arithmétiques

Elles servent à entrer les données numériques pour le calcul. Les touches **+**, **-**, *****, et **/** commandent au SHARP, respectivement, d'additionner, de soustraire, de multiplier et de diviser. La touche **E** permet l'entrée de nombres en "notation scientifique". L'utilisation de cette notation et d'autres fonctions plus avancées fait le sujet du chapitre "Calcul avancé".

D. SHIFT (Décalage)

On utilise cette touche pour la mise en action des fonctions inscrites au-dessus de nombreuses touches non-alphabétiques. Par exemple, pour écrire le double-point, appuyez d'abord sur la touche **SHIFT**, puis sur la touche avec astérisque. Quand on appuie sur une touche alphabétique après avoir pressé sur **SHIFT**, une minuscule apparaît sur l'affichage. (NOTE: Inversement, sur le mode SMALL, l'utilisation de **SHIFT** avant la touche alphabétique produira une majuscule.)

Quand on appuie sur la touche SHIFT, le mot SHIFT apparaît dans l'angle supérieur gauche de l'affichage. Ce mode n'est efficace qu'une seule fois pour la touche que l'on enfonce immédiatement après SHIFT.

On appelle "TOUCHES DE RESERVE" les six touches situées en haut du clavier, juste en dessous de l'affichage. Grâce à un usage spécial (décris plus loin) de **SHIFT**, on peut affecter certaines opérations ou commandes fréquentes à ces touches.

NOTE: Si vous appuyez par mégarde sur la touche **SHIFT**, appuyez à nouveau pour éliminer son effet.

E. Les minuscules et la touche SMALL (SML)

A l'aide de la touche **SML**, vous pouvez spécifier que vous voulez écrire en minuscules certains ou tous les caractères alphabétiques. Si vous ne spécifiez pas le mode SMALL, le SHARP choisira des majuscules chaque fois que vous appuierez sur une touche alphabétique. (On appelle cela le mode "par défaut", ce qui veut dire que si vous n'intervenez pas avec une directive différente, l'appareil décidera "par défaut"). Vous pouvez écrire individuellement des lettres en minuscules en appuyant sur la touche **SHIFT** avant d'appuyer sur la touche alphabétique, comme nous l'avons vu plus haut. On passe sur le mode SMALL en appuyant sur la touche **SML**. Dans ce mode, il suffit d'appuyer sur les touches alphabétiques pour écrire des minuscules tandis que pour les majuscules, il faut d'abord presser la touche **SHIFT**. Les lettres SMALL apparaissent sur la partie supérieure de l'affichage quand l'ordinateur fonctionne sur le mode SMALL. Une fois sur le mode SMALL, l'ordinateur y restera jusqu'à ce que vous appuyiez de nouveau sur la touche **SML**.

NOTE: Nous vous recommandons de limiter votre utilisation des minuscules pour le moment, car le SHARP ne reconnaît que les commandes en majuscules. Quand vous saurez programmer vous trouverez les minuscules très utiles.

F. L'AFFICHAGE

Appuyez sur ON. La partie "vitrée" de l'ordinateur s'appelle l'affichage. Il correspond à l'illustration suivante:



Vous devriez voir le symbole de guidage >, plusieurs mots ou abréviations et un point (qui indique que les piles sont chargées). Si les abréviations apparaissant sur l'affichage ne correspondent pas à celles de notre dessin ci-dessus, ne vous faites pas de soucis: elles changent au cours de votre utilisation du SHARP.

G. LE CURSEUR et le SYMBOLE de GUIDAGE

Le symbole de guidage ">" se trouve à l'extrême droite de l'affichage. Il est là pour vous inciter à parler au SHARP. Quand il est affiché, il vous indique que le SHARP n'a pas de projets pour l'immediat et qu'il attend vos ordres. Tapez une lettre de votre choix. Elle remplace > à la gauche de l'affichage tandis qu'à la droite de cette lettre apparaît un – (trait de soulignage) que l'on appelle "curseur". A mesure que vous appuyez sur des touches ce trait avance à travers l'affichage, indiquant chaque fois où le caractère suivant va apparaître. Tapez votre nom et suivez le mouvement du curseur.

Si vous écrivez plus de 25 caractères (la limite de ce qui peut apparaître en une fois) la ligne entière sautera vers la gauche (Faites l'essai!). Les caractères "poussés dehors" ne sont pas perdus: ils restent à l'intérieur du SHARP comme partie intégrante de la ligne écrite et cela jusqu'à un maximum de 80 caractères sur une seule ligne. Nous verrons plus tard comment faire pour "rappeler" ces caractères.

H. REMISE A ZERO

(La touche **CL** rouge dans l'angle droit supérieur du clavier)

Cette touche sert à effacer le contenu de l'affichage. Utilisez-la pour éliminer les caractères que vous venez juste d'écrire. Remarquez que le symbole de guidage vient de réapparaître pour vous avertir que l'ordinateur attend à nouveau vos ordres. La touche **CL** de remise à zéro sert aussi à éliminer une commande erronée. (Voyez plus loin la section concernant les messages d'erreur).

I. Touche ENTER

Les lettres et les nombres apparaissent sur l'affichage au fur et à mesure que vous les écrivez. Cependant le SHARP n'entrera en action que si vous lui indiquez que vous avez fini d'écrire. (C'est normal; après tout il n'est pas censé lire vos pensées!) Vous pouvez faire cela en appuyant sur la touche **ENTER**. A ce moment l'ordinateur examinera les caractères écrits pour voir si la forme est correcte. Certaines erreurs, mais pas toutes, causeront le refus de votre entrée.

N'OUBLIEZ PAS: d'appuyer sur la touche **ENTER** après chaque commande ou donnée élémentaire que vous désirez entrer dans l'appareil.

J. MESSAGES D'ERREUR

Appuyez sur les touches suivantes:

1 + 1 =

Appuyez maintenant sur **ENTER**.

La réponse devrait être affichée. Trois, n'est-ce pas? Non!? L'ordinateur vous répond: "ERROR 1"? Est-ce qu'il a un défaut de fabrication? Jamais de la vie! Vous avez fait une erreur de forme dans votre commande. "ERROR 1" est un code d'erreur qui vous révèle que vous avez effectué votre calcul de façon erronée. (Pour ceux que cela intéresse, il y a une liste complète des messages d'erreurs en appendice à la fin de ce manuel). Nous prenons la responsabilité pour la présente erreur. Plus loin nous vous montrerons quelles touches il faut utiliser pour corriger une commande erronée. Pour l'instant, contentez-vous d'éliminer le message d'erreur à l'aide de la touche **CL**.

K. INDICATEUR DE CHARGE DES PILES

Quand ce point disparaît, il est temps de remplacer les piles du SHARP. Reportez-vous à l'appendice pour les instructions nécessaires.



QUESTIONNAIRE – CHAPITRE 1

Trouvez dans la colonne B un élément correspondant à chaque élément de la colonne A. (Réponses en bas de la page).

Avertissement: La colonne B contient des réponses stupides.

A

- a) Touche SHIFT
- b) L'affichage
- c) Le Curseur
- d) Touche ENTER
- e) Touche BREAK
- f) Touche CL
- g) Le symbole de guidage
- h) La touche EXOTIQUE

B

- 1) Un messager du temps du deuxième Triumvirat.
- 2) Une touche qui nettoie la "vitre" de l'affichage et efface les résultats des calculs antérieurs.
- 3) Une touche qui choisit l'un des caractères qui ont une touche en commun.
- 4) On la cherche toujours encore.
- 5) Un caractère qui, lorsqu'il apparaît sur l'affichage, indique à l'utilisateur (ou l'utilisatrice) que l'ordinateur attend ses ordres.
- 6) Une touche qui indique à l'ordinateur que l'utilisateur (ou l'utilisatrice) a fini d'écrire.
- 7) Une touche qui interrompt une programme en cours d'exécution.
- 8) Le nom d'un poisson très rare des mers tropicales.
- 9) Le caractère qui, sur l'affichage indique où le prochain caractère va apparaître.
- 10) La "vitre" dans laquelle les données apparaissent.

Réponses: A-3, B-10, C-9, D-6, E-7, F-2, G-5, H-4

II. PLONGER DANS L'EAU FROIDE

En fait, ce chapitre est probablement mal nommé. Aprendre à se servir du PC-1500 est loin d'être aussi traumatisant qu'un saut dans l'eau froide d'une piscine. Cependant il est nécessaire de surmonter ses craintes avant d'aborder l'ordinateur. Comme nous l'avons vu avant, il est impossible d'abîmer l'ordinateur en appuyant sur la mauvaise touche. De plus, les ordinateurs n'ont jamais mordu personne.

Dans ce chapitre, nous allons explorer les caractéristiques fondamentales du SHARP, caractéristiques sur lesquelles sont basés les programmes et les calculs avancés. Prenez votre temps avec les exemples de chaque section. Une bonne compréhension des opérations de base est essentielle si vous vous proposez d'utiliser à fond les capacités de cet appareil.

Bien que nous ne le recommandions pas, si vous estimatez que vous êtes assez fort déjà, vous pouvez sauter tout de suite au résumé à la fin de ce chapitre.

A. MODE

Commençons avec une touche que nous avons passé sous silence jusqu'à présent. Tout à fait à la droite du clavier se trouve une touche très importante, marquée "MODE". Appuyez plusieurs fois de suite sur elle et remarquez à chaque fois le changement apporté aux abréviations en haut, à droite de l'affichage. On peut comparer cette opération avec le changement de vitesses dans une voiture. Chaque fois que l'on change de vitesse ou de mode, le même élément du moteur ou de l'ordinateur fonctionne différemment, bien qu'il ne semble pas avoir changé extérieurement. Le SHARP, comme la voiture, doit être sur le mode correct, si vous voulez qu'il fonctionne comme prévu. Et tout comme sur la voiture, si vous passez la mauvaise "vitesse" sur le SHARP, il vous le fera savoir très rapidement (mais sans grincements, bien sûr).

Appuyez à plusieurs reprises sur la touche MODE et vous ferez connaissance avec deux des plus importants modes du SHARP: RUN (passage, exécution) et PROgramme. Un troisième, le mode RESERVE, peut être mis en action en appuyant sur la touche SHIFT avant d'appuyer sur MODE. Nous verrons plus loin, dans d'autres chapitres de ce manuel comment chacun de ces modes apporte sa contribution au bon fonctionnement du SHARP. Pour l'instant, il vous suffit de savoir que si vous voulez vous servir du SHARP en tant que calculateur il faut que vous soyez en mode RUN. Plus tard, vous écrirez et modifierez des programmes en mode PROgramme. A l'aide du mode RESERVE vous pourrez affecter des commandes dont vous vous servez fréquemment à une seule touche de réserve. Tout cela fait l'objet d'explications en détail dans le chapitre 7.

Vous remarquerez que chaque fois que vous remettrez l'ordinateur en marche après avoir changé les piles, il se mettra de lui-même sur le mode RUN. Les autres fois, quand vous remettrez le SHARP en marche, il se rétablira sur le mode engagé au moment où il a été mis à l'arrêt.

B. CALCULS SIMPLES

Plaçons le SHARP sur le mode RUN pour faire l'essai des opérations mathématiques les plus élémentaires. Il est nécessaire d'appuyer sur la touche **[CL]** avant chaque calcul, dans le but d'effacer toute donnée antérieure qui pourrait s'ingérer dans les nouvelles opérations. Trouvez la réponse aux opérations élémentaires suivantes:

<u>Entrée</u>	<u>Affichage</u>
5 + 2 ENTER	7
5 - 2 ENTER	3
5 / 2 ENTER	2.5
5 * 2 ENTER	10

NOTE: N'écrivez pas le signe d'égalité =. Rappelez-vous que c'est au moyen de la touche ENTER que vous signalez au SHARP que vous avez fini d'écrire et que vous désirez qu'il exécute vos ordres, ou effectue un calcul.

C. CALCULS EN SERIES

Dans l'exemple suivant, on utilise directement la réponse à la première opération en passant à la deuxième opération sans appuyer sur la touche CL entre les deux.

<u>Entrée</u>	<u>Affichage</u>
161.16 - 47.50 ENTER	113.66
- 12.33	113.66 - 12.33
ENTER	101.33

Comme vous voyez, le résultat de la première opération saute à la gauche de l'affichage quand vous débutez la seconde.

NOTE: N'écrivez pas de virgules ou de \$ quand vous faites des calculs. Ces signes possèdent des sens particuliers dans le langage BASIC qui est celui du SHARP.

D'autres opérations peuvent être effectuées de la même façon. Essayez les suivantes:

<u>Entrée</u>	<u>Affichage</u>
5 + 3 ENTER	8
8 + 3 - 1 ENTER	10
10 * 3 - 1 ENTER	29
29/3 - 1 ENTER	8.666666667

D. CALCULS AVEC DES NOMBRS NEGATIFS

Prétendez que vous avez offert deux pommes à M. Laflèche votre professeur d'informatique. Il vous reste cinq pommes en stock et vous vous demandez: "Combien aurais-je maintenant de pommes si je n'avais pas été aussi généreux envers M. Laflèche?" Pour trouver la réponse, imaginez que vous annulez la soustraction que vous avez faite à votre stock de pommes, c'est-à-dire que vous soustrayez la soustraction. Tapez 5 -- 2 **ENTER** sur le SHARP. Votre stock hypothétique serait sept pommes. Essayez d'effectuer les opérations suivantes qui sont basées sur le même problème des nombres négatifs:

$5 * - 2$

$5 + - 2$

$5 / - 2$

$- 5 - 2 . 3$

$- 5 + - 2$

$- 5 / - 2$

N'OUBLIEZ PAS d'appuyer sur la touche **CL** entre chaque opération.

E. CALCULS COMPOSÉS

Il est possible d'enchaîner plusieurs opérations avant de demander une réponse au SHARP. Supposons que vous et deux amis vous vouliez partager 5 pommes deux fois par jour pendant une semaine. Combien de pommes vous faut-il pour une semaine?

5 pommes **/** 3 amis ***** 2 fois par jour ***** 7 hours

Touches utilisées

5 **/** **3** ***** **2** ***** **7** **ENTER**

Affichage

23. 33333333

(Achetez 24 pommes et 1 lapin à qui vous donnerez le tiers de pomme restant). Faites les opérations suivantes (c'est à vous d'inventer une histoire, cette fois-ci):

Entrée	Affichage
$5 * 2 - 3.675$ ENTER	6. 325
$5 / 3 * 6.2 + 7 - 47$ ENTER	-29. 66666667

F. L'UTILISATION DES PARENTHESES

Un des problèmes que nous découvrons au cours des calculs complexes est celui de la priorité. Par exemple, l'expression $5 - 3/4$ peut être interprétée comme étant: (5 moins 3) divisé par 4 et aura pour résultat 0,5; ou bien comme étant: 5 moins (3 divisé par 4) ce qui fait 4,25.

Les parenthèses, qui sont situées dans la première rangée de touches, permettent de clarifier ce genre d'ambiguité. Faites les opérations suivantes:

Entrée	Affichage
$5 - 3 / 4$ ENTER	4. 25
$5 - (3 / 4)$ ENTER	4. 25
$(5 - 3) / 4$ ENTER	0. 5

Le SHARP est prédisposé (à des priorités "par défaut") à exécuter certaines opérations avant d'autres (pour une liste complète de l'ordre dans lequel le SHARP exécute des opérations, reportez-vous à l'appendice 0). Les divisions et les multiplications sont effectuées avant les additions et les soustractions, à moins que l'on se serve des parenthèses pour modifier cet ordre. Il en résulte que le SHARP interprétera la première équation (sans parenthèses) du diagramme ci-dessus de la même façon que la seconde plutôt que de la troisième. Donc, pour être sûr que la réponse de l'ordinateur va être celle dont vous avez besoin, utilisez les parenthèses pour lui indiquer l'ordre correct des opérations.

Le SHARP est capable d'interpréter un emboîtement de parenthèse comme dans le cas ci-dessous:

Entrée	Affichage
$((6 - 4) / 2) * (((3 - 1) / 4) * 6)$ <input type="button" value="ENTER"/>	3

Les équations situées dans le noyau le plus au centre des parenthèses sont traitées avant les autres.

NOTE: Si vous avez des doutes, utilisez les parenthèses pour clarifier l'ordre de vos opérations arithmétiques.

G. COMPARAISONS LOGIQUES ET FONCTIONS LOGIQUES

Le SHARP vous permet de comparer deux valeurs ou deux équations et vous fournit le résultat de cette comparaison. Cette capacité est fondamentale pour la conception de programmes destinés à prendre des décisions. Les étudiants en mathématiques nouvelles reconnaîtront certainement cette façon de faire cela comme étant une fonction logique (ou encore "inégalité"). (Ne vous désespérez pas si vous ne vous connaissez pas en mathématiques nouvelles: c'est une nouveauté aussi pour l'auteur de ce manuel).

On peut considérer une fonction logique comme étant une comparaison qui est soit vraie soit fausse. Par exemple, il se trouve que la formulation "six divisé par trois est égal à deux" est vraie. Par contre, "six divisé par trois est plus grand que cinq" est une proposition fausse.

Les ordinateurs et les mathématiciens se servent des symboles suivants pour exprimer les différentes sortes de comparaisons:

<	moins grand que
>	plus grand que
=	égal à
<=	moins grand que OU égal à
>=	plus grand que OU égal à
<>	inégal à

Nous pouvons donc récrire les fonctions logiques du texte ci-dessus de la façon suivante: $6 / 3 = 2$ et $6 / 3 > 5$ respectivement.

Quand on présente une fonction logique au SHARP, il détermine si elle est vraie ou fausse. La pratique courante dans le domaine des ordinateurs constitue à représenter une formulation vraie par un 1 et une formulation fausse par un 0. Cette pratique vaut pour le SHARP. Par exemple si vous entrez:

6 / 3 = 2 **ENTER**

Le Sharp répondra avec un 1 (vrai) et pour:

6 / 3 > 5 **ENTER**

la réponse sera 0 (faux).

Mettez le bon jugement du SHARP à l'épreuve à l'aide des exercices suivants (assurez-vous d'être sur le mode RUN).

Entrée	Résultats
4 SHIFT > 5 ENTER	0
5 SHIFT > 4 ENTER	1
2 * 2 SHIFT < 2 * 3 ENTER	1
1 8 SHIFT < SHIFT > 1 8	0
2 = 2 ENTER	1
5 SHIFT < = 5 ENTER	1
2 5 * 2 SHIFT > = 1 0 0 / 3 + 4 ENTER	1

Peut-être que vous avez remarqué les équations peuvent être aussi complexes que nécessaires. Une seule limitation: 80 symboles sur une ligne.

Le problème suivant illustre l'utilisation pratique des inégalités (ou fonctions logiques):

Vous entrez dans une épicerie pour acheter du sucre. On le vend en sacs de 4, 8 et 12 kilos. L'argent que vous avez sur vous vous permet d'acheter soit deux sacs à 12 kilos soit trois sacs à 4 kilos et un de 8 kilos. Vous vous demandez dans quel cas vous obtiendrez le plus de sucre. Vous demandez à votre SHARP:

2 * 12 > (3 * 4) + 8

Il répond: 1 et vous achetez les 2 sacs à 12 kilos.

Essayez d'appliquer ce système à l'un des problèmes que vous rencontrez dans votre vie quotidienne.

H. LES TOUCHES ET LES CARACTERISTIQUES DE MISE EN PAGE

La plupart des êtres humains (sauf nous autres les génies) ont tendance à commettre des erreurs. Reconnaissant cette faillibilité de l'homme, les techniciens qui conçoivent le PC-1500 y ont intégré plusieurs caractéristiques qui facilitent les modifications et les corrections.

H.1. Flèche à gauche et touche DELete (d'effacement). **DEL**

Vous avez certainement dû voir déjà la touche à flèche dirigée vers la gauche, en base sur le côté droit du clavier. L'action de cette touche ressemble à celle de la touche de rappel arrière sur les machines à écrire modernes. Elle permet de retourner sur des caractères écrits antérieurement.

Ecrivez les caractères suivants, après avoir engagé le mode RUN et mis l'affichage à zéro (c-à-d que le symbole de guidage doit être visible).

L E SPACE L A T I N

L'affichage devrait être comme suit:

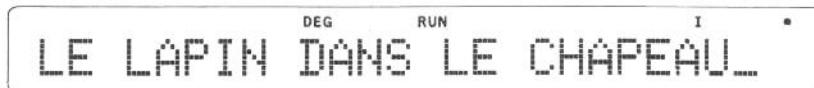


Appuyez une fois sur la touche à flèche à gauche et regardez de quelle manière le curseur change. Cette forme en "grille clignotante" du curseur vous permet de voir le caractère situé à la position présente du curseur. Appuyez plusieurs fois (ou de manière continue) sur la touche à flèche à gauche jusqu'à ce que le curseur soit situé au-dessus du T. (Si vous dépassiez le T par mégarde, ramenez le curseur en avant de nouveau à l'aide de la touche à flèche à droite. Ecrivez un P. Le P remplace le T et le curseur avance. Cela ne devrait pas vous surprendre si vous vous souvenez que le curseur indique l'endroit où le prochain caractère va apparaître.

Ecrivez les caractères qui suivent:

I N SPACE D A N S SPACE L E SPACE C H A
P E A U

L'affichage montrera:



Comme vous voyez les caractères sur lesquels on écrit à nouveau ont disparu pour toujours.

En plus de cette fonction de rappel arrière, la touche à flèche à gauche sert aussi à mettre en action la fonction d'effacement dont le symbole DEL inscrites au-dessus de la touche. Pour effacer un caractère, placez le curseur au-dessus du caractère condamné, puis, appuyez sur SHIFT et DEL .

Faisons l'essai. Ramenez le curseur en arrière sur le D et appuyez sur SHIFT et DEL cinq fois de suite.

L'affichage indique maintenant:



H.2. Flèche à droite et Touche INSert (d'insertion)

Comme nous l'arons déjà vu, la touche > à flèche à droite fait avancer le curseur sans effacer des caractères. Tout comme la touche < à flèche vers la gauche, celle-ci fera avancer plusieurs fois le curseur si vous la maintenez en position enfoncée.

Amenez le curseur au bout de la ligne. La seconde fonction de cette touche nous permet d'INSérer des caractères dans une ligne écrite. Cette caractéristique est éminemment pratique pour ceux d'entre nous qui ont tendance à oublier certaines petites choses (comme des lettres ou des mots, par exemple).

Ramenez le curseur sur le P de LAPIN. Appuyez cinq fois sur la séquence SHIFT INS et notez que les caractères en existence sautent vers la droite et sont remplacés par des caractères en forme de carrés que l'on pourrait appeler "garde-place". On peut les remplir avec de nouvelles données.

Ecrivez:

L E SPACE

Et voilà. Vous venez juste d'insérer un mot. (Je voudrais bien que ma machine à écrire puisse accomplir ce genre de prouesse!)

H.3. FONCTION DE RAPPEL

Nous nous sommes occupés jusqu'à présent uniquement des moyens de modifier des messages qui n'avaient pas encore été entrés (c'est-à-dire, après lesquels vous n'avez pas encore appuyé sur la touche **ENTER**). Une fois que vous avez appuyé sur **ENTER**, le SHARP va essayer immédiatement d'exécuter votre calcul. S'il réussit, le résultat remplacera votre équation sur l'affichage. Cette équation n'en est pas perdue pour autant. Elle peut être rappelée (ré-affichée) en appuyant soit sur la touche DEL soit sur la touche INS.

Remettez l'affichage à zéro et écrivez l'équation de votre choix. Appuyez sur **ENTER** pour obtenir le résultat. Rappelez votre équation. Remarquez que si vous voulez voir le résultat de nouveau, il faudra appuyer à nouveau sur la touche **ENTER**.

Heureusement, cette fonction de rappel agit aussi lorsque le SHARP détecte une erreur quand il évalue (essaie de comprendre) votre entrée. Elle vous permet de rappeler et de corriger l'équation erronée à l'aide des différentes méthodes que vous venez juste d'apprendre. Faites-en l'essai en entrant l'équation incorrectement énoncée qui suit:

45 * 63 / * 2 **ENTER**
↑

Quand le message d'erreur (ERROR 1) apparaît, appuyez sur l'une des touches à flèche pour rappeler l'expression. Vous devriez voir apparaître le curseur en "grille clignotante" au-dessus du deuxième astérisque (symbole de multiplication). Voilà la méthode que le SHARP utilise pour vous indiquer à quel point il a été rendu perplexe (et avec raison dans ce cas-ci). A partir de maintenant vous pouvez effectuer toutes les corrections que vous jugerez nécessaires.

I. LES VARIABLES

La capacité d'agir abstrairement par l'intermédiaire de variables constitue l'une des caractéristiques les plus formidables du SHARP. On peut se représenter les variables comme une série de petites boîtes que l'on peut remplir avec un élément unique de données tel qu'un nombre ou un nom.

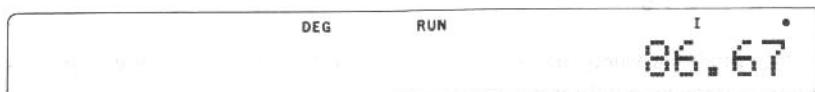
Peut-être que vous vous souvenez des variables apprises en algèbre au lycée? Vous avez dû apprendre (ou devons nous dire: on a essayé de vous apprendre) que si $5A = 30$, A doit être égal à 6 et que si $5A = 35$, A est égal à 7. Dans ce cas, le A est une variable qui contient un seul nombre (qui n'est pas toujours le même et donc "variable") que l'on appelle la "valeur" de la variable. Il est très pratique de pouvoir utiliser une lettre (comme A dans le cas ci-dessus) dans une équation à la place d'un nombre spécifique, comme l'exemple suivant nous le montrera:

Monsieur Dupont s'est décidé à acheter un jeu de clubs de golf en plastique de marque Cassevite. Le jeu comprend 5 clubs à 12F chaque, un sac (imitation cuir vert) à 21,99F et trois balles (en papier mâché avec deux couches de peinture protectrice) à 1,56F chaque. Ce jeu est en vente au magasin de sport "Le Mont-Chauve" où l'on fait une remise de 10%, mais où les frais de livraison sont de 8%. Par contre, au supermarché Multiprix, on peut se procurer le jeu avec une remise de 5% seulement, mais la livraison y est gratuite.

Pour trouver où il fera la meilleure affaire, M. Dupont décide de s'adresser à son fidèle PC-1500. Il calcule le prix de base du jeu et en fait la variable G:

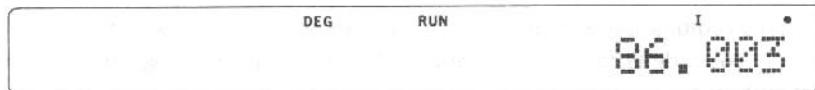
G = (5 * 12) + 21.99 + (3 * 1.56) **ENTER**

Le résultat est affiché et pourra être rappelé de la mémoire en écrivant simplement la lettre **G**, puis **ENTER**.



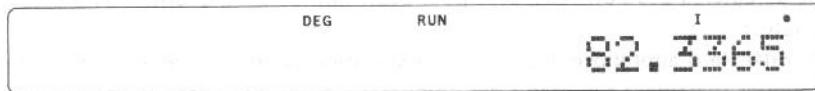
Maintenant M. Dupont calcule le prix réel du jeu au Mont-Chauve:

$G - (G * .10) + 8$ **ENTER**



puis chez Multiprix:

$G - (G * .05)$ **ENTER**



Il est évident que c'est chez Multiprix que M. Dupont va faire son achat. (Les lecteurs astucieux auront probablement remarqué que M. Dupont aurait aussi pu résoudre ce problème en se servant de la caractéristique de rappel unique au SHARP (voir Chapitre 2), bien que ce soit un peu plus compliqué)

Nous pouvons tirer plusieurs leçons de l'exemple précédent . . .

Notez que le premier calcul de M. Dupont se présentait sous la forme suivante;

Nom de la variable = expression

On appelle instruction d'affectation les instructions se présentant sous cette forme. Il est très important de ne pas confondre une instruction d'affectation avec une fonction logique. A l'encontre de l'instruction d'affectation, on n'utilise pas les fonctions logiques séparément, mais en tant qu'élément d'autres commandes de programmation.

L'instruction d'affectation ordonne au SHARP de mémoriser le résultat obtenu en calculant l'expression dans le "casier" correspondant au nom de la variable. Par la suite, utiliser le nom de la variable (G dans notre cas) équivaudra à utiliser le résultat lui-même. Prenez note aussi du fait que vous pouvez vous servir de la variable aussi souvent que nécessaire dans le même calcul.

On peut aussi se servir des variables pour d'autres tours de passe-passe. On peut, par exemple, affecter la valeur d'une variable à une autre:

$H = G$

Cette formulation passe les résultats précédents de G à H. G n'a pas changé, mais son résultat est stocké maintenant sous deux variables.

On peut aussi augmenter ou diminuer la valeur d'une variable contenant des nombres, comme l'indique l'exemple:

G = G + 5

Cette commande rappelle la valeur de G, y ajoute 5 et restocke la nouvelle valeur sous G. Cette capacité s'avère utile dans de nombreux calculs:

Le coût d'un machin est mémorisé dans la variable X. Supposons que la T.V.A. soit de 6,5%, quel est le prix d'achat du machin?

X = X * 1.065

Si nous avions voulu laisser X inchangé, nous aurions pu écrire P = X * 1.065. Et si nous avions stocké le cours de la taxe sous la variable T, nous pourrions aussi écrire X = X * T ou encore P = X * T.

Jusqu'à présent nous avons fait usage d'une lettre seulement pour une variable, ce qui met un total de 26 variables à notre disposition (A à Z). En fait, le SHARP met 950 noms de variable à notre disposition pour des nombres simples. Et comme si cela ne suffisait pas encore, l'utilisateur du SHARP peut, aux moyens de méthodes avancées, créer des variables contenant autant de nombres ou de caractères que l'on veut, avec pour seule limitation la quantité de mémoire disponible dans l'ordinateur.

Il n'est pas difficile d'apprendre le système de spécification des variables. On peut choisir les variables numériques (destinées à contenir des nombres) d'après les règles suivantes:

- Le nom peut être constitué d'une lettre de A à Z
- Le nom peut être constitué d'une lettre suivie d'un chiffre unique (de 0 à 9) ou d'une autre lettre.

Ainsi, par exemple, les noms suivants peuvent être adoptés pour des variables:

S, Q1, TX, MM, Z9, R0, E.

NOTE: Les combinaisons suivantes ne doivent pas être utilisées en tant que noms de variables du fait qu'elles ont une signification particulière dans le langage BASIC: LF, LF\$, LN, PI, TO.

Les noms de variables à caractères (contenant des caractères) suivent les mêmes règles que celles énoncées ci-dessus sauf qu'elles sont suivies du symbole \$. Ce symbole avertit le SHARP que cette variable contient des données en caractères. L'exemple suivant vous montre des noms possibles pour des variables à caractères:

T\$, P2\$, T7\$, AA\$, YR\$, X\$, ZH\$, B5\$.

NOTE: Les combinaisons suivantes ne doivent pas être utilisées en tant que noms de variables à caractères parce qu'elles font partie du vocabulaire BASIC: LF\$, IF\$, LN\$, PI\$, TO\$.

Il est important que vous compreniez que la variable A et la variable A\$ sont totalement différentes: la première représente uniquement un nombre et la seconde uniquement un caractère. Comme nous le verrons plus tard, le langage BASIC du SHARP comprend aussi des instructions concernant la conversion de caractères en nombres et de nombres en caractères. Pour mémoriser des caractères sous des variables à caractères, on se sert de notre chère amie, l'instruction d'affectation:

nom de variable à caractères = "caractères"

Voyons l'exemple suivant:

D\$ = "JEAN BART"

Rappelez maintenant le contenu de D\$ en écrivant **D SHIFT \$ ENTER**

JEAN BART

Remarquez que quoique l'espace entre les deux mots ait été mémorisé, les "guillemets" ne l'ont pas été. Les guillemets servent à "délimiter", à indiquer que l'on va mémoriser d'autres caractères. On peut mémoriser n'importe quel caractère (y compris les espaces) sauf les guillemets. On appelle ce genre de séquence de caractères encadrée par des guillemets une "chaîne" et, souvent, l'on applique aux variables à caractères le nom de "variables en chaîne". Chaque variable à caractère peut contenir jusqu'à 16 caractères. Il faut garder un oeil sur cette limite pour éviter de perdre des données. L'exercice suivant illustre cela, en régularisant involontairement la boisson de Jean. Ecrivez:

F\$ = "JEAN BOIT DU VINAIGRE"

Rappelez maintenant: **F SHIFT \$ ENTER**. Quel changement!

Une note finale à propos des variables: elles ont une mémoire d'éléphant. Des données resteront mémorisées sous une variable jusqu'à ce que:

- 1) Une autre instruction d'affectation soit exécutée pour la même variable
- 2) Une autre commande NEW ou CLEAR soit donnée
- 3) On passe un programme sur le mode RUN
- 4) On remplace les piles de l'ordinateur

La valeur de la variable reste mémorisée même si l'on met l'ordinateur hors tension. Faites l'essai: appuyez sur OFF puis ON de nouveau. Ensuite sur le mode RUN, rappelez la valeur de G en écrivant **G ENTER**. Impressionnant, n'est-ce pas? A partir du chapitre prochain vous découvrirez que les variables sont indispensables.

J. UNE PAUSE

Félicitations! Si vous avez persévééré jusqu'ici, vous avez dû assimiler les principes de base du SHARP PC-1500 suffisamment bien pour pouvoir effectuer un grand nombre d'opérations. Grâce à la souplesse du PC-1500, le lecteur pourra lui trouver une multiplicité d'usages dans son domaine d'activité propre. Quelle que soit votre application, l'heure viendra où vous voudrez apprendre à programmer pour utiliser au maximum les capacités de cet instrument étonnant. C'est à vous de choisir. A ceux dont le cœur se met à battre très vite et dont la respiration s'accélère à la pensée de programmer aussi bien qu'à ceux qui ne se sont pas encore endormis, nous allons demander de revenir en arrière et de relire le Chapitre 0 et combiner vos nouvelles connaissances avec les données qui s'y trouvent. Ensuite passez au Chapitre 3.

Les autres lecteurs qui ont besoin immédiatement de caractéristiques telles que la numération scientifique et les fonctions trigonométriques pourront passer tout de suite au chapitre "Calculs avancés".

RESUME

- 1) **Les modes** — Le SHARP PC-1500 possède trois différentes modes d'opérations: RUN (passage), PROgramme, et RESERVE. Chaque changement de mode produit un changement légèrement différent dans le fonctionnement interne, analogue au changement de vitesses sur une voiture. Le mode RUN est utilisé pour les calculs et c'est le mode sur lequel vous devez exécuter les programmes. Le mode PROgramme sert à écrire et mettre en forme (à faire des ajouts et à corriger) les programmes.
- 2) **Les calculs** — Le SHARP effectue les opérations arithmétiques d'ordre général sur le mode RUN. Appuyez sur la touche CL (de remise à zéro) avant chaque opération pour éliminer le résultat des opérations précédentes. NE PAS appuyer sur cette touche entre les opérations permet d'effectuer des calculs en série utilisant chaque fois le résultat de l'opération précédente.
- 3) **Caractères spéciaux** — Le symbole \$ et la virgule (,) ont une signification particulière dans le langage BASIC et ne doivent pas être utilisés comme partie d'un nombre dans une opération.
- 4) **Les nombres négatifs** — sont précédés du signe négatif — (moins), comme par exemple: $-5 + 2 = > -3$
- 5) **Les calculs complexes** — suivent les lois algébriques générales. Les parenthèses servent à préciser le sens d'une expression donnée. L'appendice O fournit des renseignements complets au sujet de l'ordre d'évaluation de l'expression.
- 6) **Les fonctions logiques** — Les fonctions logiques (exprimées par les symboles suivants: < moins grand que, > plus grand que, <= moins grand que ou égal à, >= plus grand que ou égal à, <> inégal à) sont jugées ainsi: 0 si elles sont fausses et 1 si elles sont vraies.
- 7) **La touche à flèche à gauche** — Cette touche sert de rappel arrière "non-destructif" du curseur. Maintenir cette touche en position enfoncée cause une répétition automatique. Le curseur endessous de ligne se métamorphose en "grille clignotante" quand on le place au-dessus d'un caractère écrit. Si l'on appuie sur la touche SHIFT avant la touche <, la fonction DEL (d'effacement) entre en action et efface le caractère au-dessus duquel s'est placé le curseur.
- 8) **La touche à flèche à droite** — Cette touche avance le curseur de façon "non-destructive". Le maintien de cette touche en position enfoncée cause une répétition automatique. Si l'on appuie sur la touche SHIFT avant la touche >, la fonction INSérer entre en action et insère un caractère "garde-place" à l'endroit où se trouve le curseur. L'on peut ensuite écrire au-dessus de ce caractère la donnée que l'on veut insérer.
- 9) **La Fonction de rappel** — Il est possible de rappeler l'équation originale, après avoir appuyé sur la touche ENTER et faire afficher le résultat du calcul, en appuyant soit sur la touche < soit sur la touche >. Il est possible alors d'apporter des modifications à l'équation puis de l'entrer à nouveau. La fonction de rappel agira aussi dans le cas où une expression non-programmée résulte en une erreur. Dans ce cas, le curseur sera placé au point où l'erreur a été découverte.
- 10) **L'utilisation des variables** sur le mode RUN augmente considérablement la puissance ordinatrice du PC-1500 et permet d'abréger des expressions complexes.
- 11) **Les instructions d'affectation** qui utilisent les formes suivantes:
nom de variable = expression
nom de variable à caractères = "caractères"
permettent le stockage d'un nombre unique ou d'une chaîne de 16 caractères au maximum, respectivement. On peut se servir de n'importe quel caractère, sauf des guillemets.
- 12) **Les noms de variables** pour les variables numériques sont constitués par:
 1. Une lettre (A à Z)
 2. Une lettre suivie d'un chiffre (0 à 9) ou d'une autre lettre.Les noms de variables à caractères suivent les mêmes règles, sauf qu'on leur ajoute à tous le suffixe \$.

- 13) **Exceptions** — Les symboles suivants constituent des exceptions et ne peuvent être utilisés comme noms de variable: **IF, LF, LN, PI, TO, IF\$, LF\$, LN\$, PI\$, TO\$**.
- 14) **Durée de vie des données** — les données contenues dans des variables sont retenues jusqu'à ce:
- 1) Qu'une commande CL ou NEW soit donnée.
 - 2) Qu'un programme soit exécuté grâce à la commande RUN.
 - 3) Qu'une autre instruction d'affectation soit effectuée pour la même variable.
 - 4) Que l'on remplace les piles de l'ordinateur.

La mise à l'arrêt (OFF) de l'ordinateur ne change en rien les valeurs mémorisées dans les variables.

III. L'ART MYSTERIEUX DE LA PROGRAMMATION

L'art de la programmation a été enveloppé dans un voile de mystère pendant si longtemps que la plupart des gens l'associent avec l'art de la magie ou encore les génies mathématiques. Le fait est qu'il n'est pas requis de savoir sortir des lapins d'un chapeau ni de savoir résoudre des équations différentielles partielles. Avant tout il vous faut de la patience, une certaine capacité de raisonnement logique, le goût du détail et surtout la volonté d'apprendre. Et une disposition à relever les défis facilite certainement les choses, car à certains moments la programmation est une véritable gageure. Mais c'en est un des charmes aussi.

En tant qu'art, la programmation demande un peu de talent, un peu de formation et beaucoup de pratique. Nous n'avons pas l'intention de vous transformer en programmeur chevronné avec ce manuel. Nous voulons vous familiariser avec les opérations de base et les concepts de l'art de programmer. Pour devenir un bon conducteur de voiture, il en faut plus que de savoir le maniement du volant et la façon de passer les vitesses. Il en est de même avec la programmation.

Il existe aujourd'hui beaucoup de livres intéressants au sujet de la programmation et nous vous recommandons vivement d'aller faire un tour à la bibliothèque et chez le marchand d'ordinateurs de votre région. Vous trouverez une liste de bons livres à propos de la programmation en général et du langage BASIC en particulier dans l'appendice F.

A. Qu'est-ce qu'un programme?

Vous serez probablement étonné de découvrir qu'un programme est tout simplement une série de directives que l'ordinateur traite séparément l'une après l'autre. Il faut simplement que ces directives soient données dans un langage que l'ordinateur "comprend". Le SHARP PC-1500 parle dans un dialecte BASIC, un langage de programmation très répandu et très populaire. Comme tous les langages, le BASIC a une grammaire et un vocabulaire particuliers qu'il faut respecter en faisant des phrases. L'ordinateur vous fera remarquer votre erreur si vous lui parlez en utilisant des mots inconnus ou dans une mauvaise syntaxe. Mais il n'est pas vraiment difficile de donner les instructions correctes au SHARP. A l'origine, on a développé le langage BASIC pour enseigner les principes de la programmation et nombreuses sont les formulations qui contiennent des mots anglais devenus internationaux et des symboles familiers.

B. Comment bâtit un programme?

Quand vous vous servez du SHARP pour programmer, vous suivez une certaine routine. Les directives qui composent un programme sont entrées sur le mode PROgramme. On appelle ces directives des "instructions" dans le langage BASIC. Pour commencer l'exécution de ces instructions, il faut tout d'abord passer sur le mode RUN puis demander au SHARP d'avancer en écrivant le mot RUN. Cela ne sera pas difficile pour vous les "experts" qui ont déjà absorbé les deux programmes du chapitre 0. Pour les autres qui essaient d'avancer à grand-peine, essayons d'entrer et de passer un programme.

Mettez-vous sur le mode PROgramme et lancez la "commande" BASIC (plus de détails plus loin sur la différence entre commandes et instructions).

N E W ENTER

Cette opération va effacer toute instruction précédente qui pourrait être dans la mémoire. Ecrivez la ligne suivante:

Listage du programme:

10 PRINT "TRES BIEN!"

Touches utilisées

1 O P R I N T SHIFT " T R E S SPACE
B I E N SHIFT ! SHIFT " ENTER

Notre programme en une ligne est achevé. Passez au mode RUN et écrivez:

R U N ENTER

Cette commande charge le SHARP de commencer à traiter les instructions (dans ce cas-ci, une seule instruction) de notre programme. Le SHARP obéit et écrit:

TRES BIEN!

sur l'affichage. (Quand vous avez fini de lire, avertissez-en le SHARP en appuyant sur ENTER.)

Si l'on désire ajouter, modifier ou effacer quelque chose de notre programme, il faut d'abord retourner sur le mode PROgramme. S'il y avait eu une erreur dans notre programme et qu'il n'ait pas été achevé avec succès, il aurait été nécessaire de revenir sur le mode PROgramme pour corriger les instructions erronées. En somme, quand on travaille sur les programmes on se place sur le mode PROgramme, tandis que l'exécution et l'essai des programmes se fait sur le mode RUN.

C. COMMANDES et INSTRUCTIONS

Vous avez peut-être noté que dans l'exemple précédent nous avons communiqué nos désirs au SHARP avec deux méthodes différentes. Immédiatement après avoir appuyé sur ENTER, les directives NEW et RUN ont été exécutées. Ce type de directive s'appelle "commande". Par contre, la directive PRINT avait été entrée en mode PROgramme. Elle était précédée par un nombre (10), et ne fut pas exécutée immédiatement. Ce type de directive est nommée "instruction". Dans un certain sens, les commandes indiquent au SHARP ce qu'il faut qu'il fasse avec les instructions. Par exemple, la commande NEW efface toutes les instructions mises de côté précédemment. Il est important de se rappeler que l'on ne peut pas utiliser les commandes à l'intérieur d'un programme, mais que les instructions sont presque toujours groupées pour former un programme.

D. Numéros de ligne

Un programme BASIC est constitué par une série de lignes numérotées qui contiennent chacune, une ou plusieurs instructions. Ces numéros de ligne servent à maintenir l'ordre correct pendant l'exécution, mais ne feront pas partie de la sortie lors du passage du programme. On peut entrer les instructions dans n'importe quel ordre, mais leur traitement sera toujours effectué dans l'ordre déterminé par les numéros de ligne, qui, cependant, comme nous le verrons plus tard, est susceptible d'être modifié.

En guise de démonstration, ajoutons une instruction de plus à notre programme en une ligne. Passons en mode PROgramme et écrivons:

Touches utilisées

5	P	R	I	N	T	SHIFT	"	A	B	S	O	L	U	M	E	N	
T		SPACE															
SHIFT		"	SHIFT	:	ENTER												

Maintenant passons le programme révisé. Que se passe-t-il? Appuyez sur **ENTER** après l'apparition de "ABSOLUMENT".

Remarquez à la fois ce que le SHARP a fait pour vous et ce que vous avez fait pour vous-même. Le SHARP a arrangé et exécuté les lignes 5 et 10 dans l'ordre correct. Il faut cependant appuyer sur ENTER entre les sorties pour pousser le SHARP de la ligne 5 à la ligne 10 et voir le résultat de cette dernière.

Bien qu'il soit possible de leur assigner n'importe quel numéro de 1 à 65279, nous recommandons vivement que vous numérotiez par intervalles de 10 (c-à-d. 10, 20, 30, . . . etc.) ce qui vous permettra d'insérer jusqu'à 9 instructions entre les instructions existantes (de 11 à 19, par exemple). Certains programmeurs recommandent un intervalle plus grand même, 20 par exemple. Quoique vous en ayez rarement besoin si vous concevez et écrivez votre programme avec soin, il vaut mieux prendre cette précaution car il est bien plus simple d'étaler les numéros de 10 en 10 que d'avoir à renommer de nombreuses instructions plus tard.

Rappelez-vous qu'il faut éviter de donner le même numéro à deux lignes différentes, car si l'on faisait ainsi, la plus ancienne serait éliminée. On peut, bien sûr, exploiter cette caractéristique pour effacer une ligne qu'on ne veut plus, rien qu'en écrivant le numéro de celle-ci et en appuyant ensuite sur **ENTER**. De cette manière, une ligne nouvelle, même vide, peut en effacer une autre plus ancienne possédant le même numéro.

Mais si par malheur on assigne involontairement le même numéro à deux lignes différentes, on se crée des problèmes. Pour démontrer cela, entrons la ligne suivante (sur le mode PROgramme, bien sûr).

Touches utilisées

1	Ø	P	R	I	N	T	SHIFT	"	H	O	R	R	I	B	L	E
SHIFT				"		ENTER										

Passons maintenant le programme de la même façon qu'avant. Il est "ABSOLUMENT HORRIBLE" de perdre une ligne, n'est-ce pas? Nous espérons que cela vous incitera à faire très attention quand vous écrivez des programmes, car une perte de ligne peut avoir de désagréables conséquences.

E. Touches de revue des lignes de programme

Vous vous demandez peut-être: "Comment puis-je me rappeler les lignes que j'ai entrées?" N'ayez crainte, intrépide programmeur! Cela a été prévu! Il est possible de revoir ce que l'on a entré grâce aux touches à flèche vers le haut **↑** et flèche vers le bas **↓**. Considérez-les comme étant des touches de revue de ligne de programme. En appuyant sur la touche appropriée (en mode PROgramme) on peut "monter" ou "descendre" le long des lignes d'un programme (une opération qui se nomme "défilement" vers le haut ou le bas).

Passez au mode PROgramme et appuyez sur la touche à flèche vers le bas pour revoir les lignes de notre programme de haut en bas. Puis appuyez sur la touche à flèche vers le haut pour revenir sur vos pas au début du programme (ligne 10). Remarquez que si vous maintenez l'une de ces deux touches en position basse, les lignes seront affichées en succession automatiquement, (ce qu'il est malheureusement difficile de discerner dans le cas de notre programme à deux lignes).

Quand vous avez atteint la ligne que vous cherchez avec ces touches de revue de ligne de programme, vous pouvez passer à la mise en forme à l'aide des touches à flèche vers la gauche et flèche vers la droite. Vous vous réjouirez probablement de découvrir que le fonctionnement de ces touches en mode PROgramme est identique à celui en mode RUN (voyez le Chapitre 2). Les fonctions DELETED (effacement) et INSERED peuvent être utilisées de la même manière qu'avant, pour mettre en forme des instructions.

NOTE: Quelles que soient les modifications que vous apportiez à une ligne de programmes, il faut toujours appuyer sur ENTER pour les faire exécuter par l'ordinateur. N'APPUYER PAS sur l'une des touches de revue de ligne pour passer à la ligne suivante ou précédente avant d'avoir appuyé sur **ENTER**, sinon la mise en forme que vous venez juste d'effectuer disparaîtra.

F. Un examen plus détaillé de certains vieux amis

Maintenant que nous savons comment entrer, exécuter et mettre en forme un programme, nous pouvons élargir notre vocabulaire d'instructions et de commandes. Mais commençons par regarder de plus près nos vieilles amies, la commande NEW et les instructions LET et PRINT.

F.1. La commande NEW

Comme nous l'avons vu dans nos exemples de programmations précédents, la commande NEW sert à effacer toutes les lignes de programme qui existent en ce moment dans la mémoire. Pour être sûr que les seules directives présentes dans la mémoire du SHARP sont celles de notre programme actuel, nous ferons usage de la commande NEW (en mode PROgramme) avant chaque programme-échantillon. Bien qu'il soit possible et même désirable de garder plusieurs programmes dans la mémoire en même temps, nous allons différer l'utilisation de cette caractéristique afin d'éviter des confusions.

En mode PROgramme, passez la commande NEW. Quel effet ont les touches à flèche vers le haut et vers le bas maintenant?

F.2. L'instruction LET

Ne vous effrayez pas si vous ne reconnaisssez pas l'instruction LET comme étant une "vieille amie". Je vous l'ai glissée en douce. En fait, l'instruction LET n'est rien d'autre que l'instruction d'affectation sous un autre déguisement. (Si vous ne reconnaissiez pas celle-ci non plus, relisez immédiatement la section sur les variables en chapitre 2!)

Au début de sa carrière, chaque instruction de langage BASIC commençait avec un mot-clé (comme PRINT, INPUT, etc.) indiquant la fonction de l'instruction. LET, qui faisait partie de ces mots-clé, signifiait qu'il fallait mémoriser une certaine valeur sous une variable. aujourd'hui on est d'accord pour dire que le mot LET n'est pas vraiment nécessaire. Il en résulte que le mot-clé LET est facultatif dans le langage BASIC du PC-1500. Ainsi, par exemple, une instruction qui mémorise le nombre 7 sous la variable S pourra être écrite d'une des deux manières suivantes:

S = 7 → ou
→ LET S = 7

Il y a une seule exception: il faut utiliser LET quand une affectation fait partie d'une instruction IF. Essayons d'expliquer cela en dépit du fait que nous n'ayons pas encore discuté de l'instruction IF (touchez du bois).

L'instruction IF permet d'écrire une directive du type suivant:

IF expression THEN instruction

Si l'instruction qui suit le THEN (alors) constitue une instruction d'affectation, il faut employer le mot-clé LET. Il en résulte une directive ayant la forme suivante:

IF expression THEN LET nom de variable = expression

NOTE: Si vous omettez le LET dans ce cas, une erreur (ERROR 19 ou un autre message d'erreur) se produira.

F.3. L'instruction PRINT

Dans le cas de la plupart des programmes que vous écrivez, les directives suivent un modèle de base. Il y aura des directives pour lire les données brutes, des directives pour traiter les données et des directives pour imprimer ou afficher les résultats. Ce modèle s'appelle le cycle "Entrée-Traitement-Sortie".

On utilise principalement l'instruction PRINT pour la sortie. Il n'est donc pas surprenant que l'instruction PRINT existe en plusieurs variantes. En général, l'instruction PRINT est suivie d'un élément ou d'une liste d'éléments destinés à être imprimés. Cela comprend des chaînes de caractères, des expressions ou des noms de variables dont les valeurs sont destinées à être imprimées. Les éléments d'une liste sont séparés par une virgule ou un point-virgule.

Entrez le programme suivant pour pratiquer l'impression d'éléments séparés (n'oubliez pas d'appuyer sur la commande NEW avant de commencer:)

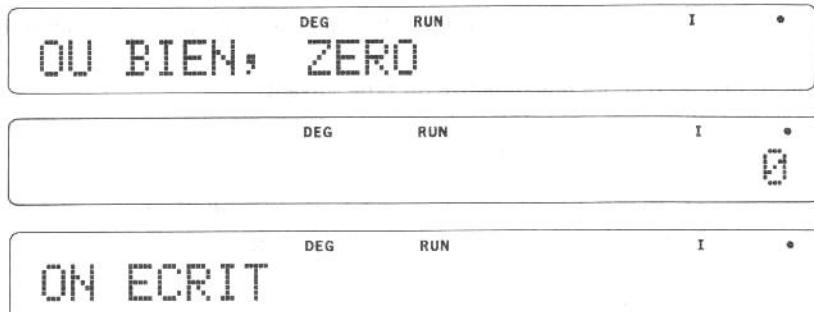
Listage du programme:

```
10 Z$ = "ON ECRIT"  
20 ZZ = 0  
30 PRINT "OU BIEN, ZERO"  
40 PRINT ZZ  
50 PRINT Z$
```

Touches utilisées:

```
1 Ø Z SHIFT $ = SHIFT " O N SPACE  
E C R I T SHIFT " ENTER  
2 Ø Z Z = Ø ENTER  
3 Ø P R I N T SHIFT " O U SPACE  
B I E N SHIFT ,  
SPACE Z E R O  
SHIFT " ENTER  
4 Ø P R I N T Z Z ENTER  
5 Ø P R I N T Z SHIFT $ ENTER
```

Après le passage de ce programme, nous devrions obtenir les trois lignes de sortie suivantes (appuyez sur **ENTER** après chaque ligne que vous avez fini de lire.)



Notre première instruction PRINT (ligne 30) affiche une chaîne de caractères. Remarquez que les guillemets ne sont pas imprimés en tant que sortie. Ils servent à délimiter ou à marquer le début ou la fin d'une série de caractères que l'on désire imprimer. On peut incorporer n'importe quel caractère dans cette série, sauf les guillemets.

Les lecteurs reconnaîtront certainement la deuxième et la troisième instruction PRINT (lignes 40 et 50) comme étant des variables. Quand on emploie un nom de variable dans une liste PRINT, la valeur de cette variable sera imprimée. Dans le cas présent, nous savions ce que les valeurs de ZZ et Z\$ seraient, puisque nous les avions spécifiées en lignes 10 et 20. Si l'on imprime une variable "vide", nous obtiendrons un zéro ou un espace vide, selon que la variable est numérique ou à caractère.

Les bons observateurs parmi les lecteurs auront certainement remarqué que l'ordinateur imprime les chaînes de caractères et les valeurs de variables à caractère en partant de la gauche de l'affichage. On dit qu'elles sont "cadrées à gauche". Par opposition, les nombres et les valeurs des variables numériques sont "cadrés à droite".

Il est possible aussi d'imprimer le résultat d'une expression contenue dans l'instruction PRINT. Le programme en une ligne suivant illustre cette possibilité;

Listage du programme:

```
10 PRINT (1982 - 1956) * 365,25
```

Touches utilisées:

1	0	P	R	I	N	T	(1	9	8	2	-	1	9	5	6)	
*	3	6	5	.	2	5	ENTER											

Comme prévu, le résultat, un nombre, est cadré à droite.

Pour obtenir le meilleur rendement possible, il vaut mieux éviter de traiter des expressions dans des instructions PRINT à moins que cette instruction (et l'expression qu'elle incorpore) ne soit exécutée qu'une seule fois au cours d'un programme.

Du fait que la plupart des programmes calculent plusieurs résultats à la fois, il arrive souvent que plusieurs éléments soient imprimés en même temps.

Dans le cas le plus simple des instructions PRINT à éléments multiples, l'affichage est divisé en deux parties. Chaque partie contient l'un des deux éléments spécifiés dans la liste d'impression. Une virgule sépare les éléments de la liste. Explorons ce format au moyen du programme suivant:

Listage du programme:

```
10 A = 2 * PI
20 PRINT "2 FOIS PI =", A
```

Touches utilisées:

1	0	A	=	2	*	P	I	ENTER		
2	0	P	R	I	N	T	SHIFT	"	2	SPACE
F	O	I	S	SPACE	P	I	=			
SHIFT	"	SHIFT	,	A	ENTER					

Passez le programme. Les nombres sont cadrés à droite et les caractères à gauche, comme dans les instructions PRINT à élément unique. Dans ce cas-ci, le cadrage (l'alignement) s'effectue dans les deux parties de l'affichage.

Modifiez la ligne 20 de la façon suivante en vous servant de vos connaissances en matière de mise en forme:

```
20 PRINT A, "= 2 FOIS PI"
```

Bien qu'il y ait plusieurs façons d'effectuer cette mise en forme, effectuons-la de la manière suivante:

MODE	↓	↑	↓	▶	▶	▶	SHIFT	INS	SHIFT	INS	A
SHIFT	,	▶	▶	SHIFT	INS	=					
▶	▶	▶	▶	▶	▶	▶	▶	▶	▶	▶	
SHIFT	DEL	SHIFT	DEL	ENTER							

La sortie de cette version modifiée devrait vous servir à identifier les deux parties de l'affichage.

Il est possible d'afficher plus de deux éléments sur la même ligne grâce à une variante de l'instruction PRINT qui emploie le point-virgule. Bâtissez et vérifiez le programme suivant:

Listage du programme:

```
10 B$ = " BE "
20 T = 2
30 PRINT T;B$; "OR NOT"; 12/3 - 2;B$
```

Touches utilisées:

```
1 Ø B SHIFT $ = SHIFT " SPACE B E SPACE
      SHIFT " ENTER

2 Ø T = 2 ENTER

3 Ø P R I N T T SHIFT ; B
      SHIFT $ SHIFT ;
      SHIFT " O R SPACE N O T SHIFT "
      SHIFT : 1 2 / 3 - 2 SHIFT ;
      B SHIFT $ ENTER
```

Le passage de cette composition électronique devrait avoir comme résultat la sortie suivante:

DEG RUN I •
2 BE OR NOT 2 BE

C'est du Shakespeare bien connu, et cet exemple montre malgré tout l'effet de l'instruction PRINT quand les éléments à imprimer sont séparés par des points-virgules. Sous ce format, les éléments sont affichés côte-à-côte avec un minimum d'espace entre eux. Cette capacité devient très utile quand on désire créer une sortie qui a l'air "naturel", c-à-d, une sortie qui tient bien ensemble.

NOTE: Si la longueur des données affichées dépasse l'espace disponible sur l'affichage (25 caractères), les éléments situés en fin de liste ne seront pas visibles.

On peut aussi employer le point-virgule complètement à la fin de l'instruction PRINT. Dans ce cas, le point-virgule servira à indiquer que la sortie affichée à ce moment doit être gardée et que, s'il y a lieu, la nouvelle sortie (dérivant de la prochaine instruction PRINT) devra rejoindre l'ancienne sur la même ligne. Comme il est plus facile de programmer que de décrire cette opération, passons tout de suite à l'essai du programme qui suit, sans oublier la commande NEW:

Listage du programme:

```
100 PRINT "MARCEL ";
110 PRINT "DUPONT"
```

Touches utilisées:

```
1 Ø Ø P R I N T SHIFT " M A R C E L  
SPACE SHIFT " SHIFT : ENTER  
1 1 Ø P R I N T SHIFT " D U P O N T  
SHIFT " ENTER
```

Maintenant exécutez-le. Comme d'habitude, appuyez sur **ENTER** après que la première instruction a fait afficher "MARCEL". A cause du point-virgule, "MARCEL" est retenu et "DUPONT" partage l'affichage avec lui. Comparez ce résultat avec l'exécution du programme suivant dans lequel nous n'employons pas le point-virgule. La différence est évidente:

Listage du programme:

```
10 PRINT "MARCEL"  
20 PRINT "DUPONT"
```

Touches utilisées:

```
1 Ø P R I N T SHIFT " M A R C E L  
SPACE SHIFT " ENTER  
2 Ø P R I N T SHIFT " D U P O N T  
SHIFT " ENTER
```

Le programme-échantillon suivant nous montre que le point-virgule peut relier autant de lignes que l'on peut placer sur l'affichage. Cependant il faut faire attention, car le SHARP ne vous avertira pas si vous commettez le péché d'imprimer trop de données sur l'affichage. C'est à celui ou à celle qui fait le programme d'éviter que cela n'arrive. Essayez le programme culturel suivant:

Listage du programme:

```
20 PRINT "DO";  
40 PRINT "RE";  
60 PRINT "MI";  
80 PRINT "FA";  
100 PRINT "SOL";  
120 PRINT "LA";  
140 PRINT "SI";  
160 PRINT "DO";
```

Toiles utilisées:

```
2 0 P R I N T SHIFT " D O SPACE  
SHIFT " SHIFT ; ENTER  
4 0 P R I N T SHIFT " R E SPACE  
SHIFT " SHIFT ; ENTER  
6 0 P R I N T SHIFT " M I SPACE  
SHIFT " SHIFT ; ENTER  
8 0 P R I N T SHIFT " F A SPACE  
SHIFT " SHIFT ; ENTER  
1 0 0 P R I N T SHIFT " S O L SPACE  
SHIFT " SHIFT ; ENTER  
1 2 0 P R I N T SHIFT " L A SPACE  
SHIFT " SHIFT ; ENTER  
1 4 0 P R I N T SHIFT " S I SPACE  
SHIFT " SHIFT ; ENTER  
1 6 0 P R I N T SHIFT " D O SHIFT "  
SHIFT ; ENTER
```

Maintenant passez le programme, en appuyant à plusieurs reprises sur la touche **ENTER** et chantez joyeusement chaque note de la gamme . . . (ça va, ça va, vous n'avez pas besoin de chanter, mais vous devez appuyer sur **ENTER** quand même).

G. L'instruction PAUSE

L'instruction PAUSE est une forme semi-automatique de l'instruction PRINT. Elle affiche les divers éléments de la liste qui lui est associée pendant une période brève et fixe. Ceci a pour effet de libérer l'utilisateur de la nécessité d'inciter le SHARP à avancer en appuyant sur ENTER. Considérez la PAUSE comme étant une instruction PRINT suivie d'un compte à rebours. Quand ce dernier est terminé, le programme continue.

Les formats de l'instruction PAUSE sont identiques à ceux de l'instruction PRINT. Toutes les techniques discutées à propos de l'instruction PRINT valent pour l'instruction PAUSE, quoique la sortie qui en résulte diffère légèrement, cela va de soi. Pour illustrer l'une des utilisations de l'instruction PAUSE, récrivons notre programme musical:

Listage du programme:

```
10 PAUSE "DO ";  
20 PAUSE "RE ";  
30 PAUSE "MI ";  
40 PAUSE "FA ";  
50 PAUSE "SOL ";  
60 PAUSE "LA ";  
70 PAUSE "SI ";  
80 PAUSE "DO";
```

Touches utilisées:

1	0	P	A	U	S	E	SHIFT	"	D	O	SPACE
		SHIFT	"	SHIFT	:	ENTER					
2	0	P	A	U	S	E	SHIFT	"	R	E	SPACE
		SHIFT	"	SHIFT	:	ENTER					
3	0	P	A	U	S	E	SHIFT	"	M	I	SPACE
		SHIFT	"	SHIFT	:	ENTER					
4	0	P	A	U	S	E	SHIFT	"	F	A	SPACE
		SHIFT	"	SHIFT	:	ENTER					
5	0	P	A	U	S	E	SHIFT	"	S	O	SPACE
		SHIFT	"	SHIFT	:	ENTER					
6	0	P	A	U	S	E	SHIFT	"	L	A	SPACE
		SHIFT	"	SHIFT	:	ENTER					
7	0	P	A	U	S	E	SHIFT	"	S	I	SPACE
		SHIFT	"	SHIFT	:	ENTER					
8	0	P	A	U	S	E	SHIFT	"	D	O	
		SHIFT	"	SHIFT	:	ENTER					

Prenez note de ce qui se passe après l'impression de la dernière "note": le programme s'achève et le symbole de guidage retourne sur l'affichage. La raison en est qu'il n'y a pas d'autres instructions qui suivent la dernière PAUSE. Nous pouvons modifier la ligne 80 de la manière suivante pour bloquer l'affichage après la dernière note:

80 PRINT "DO"

Touches utilisées:

8	0	P	R	I	N	T	SHIFT	"	D	O	
		SHIFT	"	ENTER							

Mais après tout, il n'y a pas de raison pour qu'on ne puisse mêler les instructions PRINT et PAUSE au cours de notre programme. A vous d'en faire l'essai.

H. L'instruction INPUT

A l'aide des différentes versions de l'instruction PRINT, seules ou combinées, il est possible de présenter très clairement les données à l'utilisateur. Cependant, la plupart des éléments qui seront imprimés résultent du traitement de certaines données initiales. L'utilisateur de l'ordinateur est celui qui fournit ces données initiales au programme. L'instruction qui gouverne ce processus constitue l'instruction d'INPUT.

Dans sa forme la plus simple, INPUT nom de variable, l'instruction d'INPUT écrit simplement un ? (point d'interrogation) et attend que l'utilisateur entre les données demandées. Le contenu de la demande dépend de la variable qui apparaît comme partie intégrante de la directive d'INPUT. Par exemple, si la variable est numérique, l'utilisateur devra entrer un nombre qui sera emmagasiné dans la variable.

Est-ce que vous voyez une difficulté dans ce type d'instruction d'INPUT? Bien. Comme certains d'entre vous l'ont probablement déjà réalisé, si l'utilisateur de programme n'est pas en même temps le programmeur (et même dans ce cas, ce n'est pas certain), il ne saura pas quel type de données il devra entrer quand il verra apparaître le point d'interrogation. C'est le travail du programmeur de garder l'utilisateur au courant à tout instant.

Comme exemple de programme qui néglige de faire cela, nous proposons le programme suivant:

Listage du programme:

```
10 A = 0
20 INPUT A
30 PRINT A * PI
```

Touches utilisées:

1 Ø A = Ø ENTER
2 Ø I N P U T A ENTER
3 Ø P R I N T A * P I ENTER

Imaginez maintenant le visage de l'utilisateur quand il passe ce programme . . . La première chose qui apparaît sur l'écran est un point d'interrogation. L'utilisateur tant soit peu au courant comprendra probablement qu'on lui demande d'entrer certaines données, mais il ne saura pas quelles données. Supposons même qu'il entre un nombre, au hasard, et devine juste. D'un seul coup, un numéro très long et très compliqué apparaît sur l'affichage. Qu'est-ce que cela signifie? Il appuie sur ENTER pour continuer, mais le programme s'achève. De son point de vue, toute l'opération n'est qu'une perte de temps. Pourquoi? A cause d'une piètre programmation.

La solution de ce problème est d'employer les instructions PRINT et PAUSE pour aider l'utilisateur en cours de route. Récrivons notre programme en tenant compte de cela:

Listage du programme:

```
10 A = 0
20 PAUSE "ENTREZ UN NOMBRE"
30 INPUT A
40 AP = A * PI
50 PRINT A; " FOIS PI = "; AP
```

Touches utilisées:

1 Ø A = Ø ENTER
2 Ø P A U S E SHIFT " E N T R E Z SPACE
U N SPACE N O M B R E SHIFT " ENTER
3 Ø I N P U T A ENTER
4 Ø A P = A * P I ENTER
5 Ø P R I N T A SHIFT ; SHIFT " SPACE
F O I S SPACE P I SPACE =
SPACE SHIFT " SHIFT ; A P ENTER

Cette version est beaucoup plus utile parce qu'elle "pousse" l'utilisateur à faire des entrées et parce qu'elle identifie la sortie.

Cette opération (impression d'une demande d'entrée) est tellement fréquente que l'on a créé des versions plus avancées de l'instruction INPUT qui l'incorpore. La première s'écrit avec le point-virgule:

INPUT "caractères"; nom de variable

Les lecteurs attentifs ont dû reconnaître déjà notre vieille amie la chaîne de caractères. Ces chaînes apparaîtront sur l'affichage sous forme de symbole de guidage. Le curseur remplaçant le point d'interrogation suivra sur la même ligne et le SHARP attendra l'entrée de l'utilisateur. Cette version de l'instruction INPUT nous permet de récrire l'exemple précédent:

Listage du programme:

```
10 A = 0
20 INPUT "ENTREZ UN NOMBRE "; A
30 AP = A * PI
40 PRINT A; " FOIS PI = "; AP
```

Touches utilisées:

```
1 Ø A = Ø ENTER
2 Ø I N P U T SHIFT " E N T R E Z SPACE
U N SPACE N O M B R E SPACE
SHIFT " SHIFT ; A ENTER
3 Ø A P = A * P I ENTER
4 Ø P R I N T A SHIFT ; SHIFT " SPACE
F O I S SPACE P I SPACE =
SPACE SHIFT " SHIFT ; A P ENTER
```

Au cours du passage de ce programme, notez la manière dont l'emploi de l'instruction d'INPUT diffère de l'instruction PAUSE.

La seconde version de l'opération d'inciter une entrée est presque la même que la première, sauf que la virgule remplace le point-virgule:

INPUT "caractères", nom de variable

Quand cette version de l'instruction est exécutée, la chaîne de caractères qui lui est associée est affichée et l'ordinateur attend une entrée. Cette fois, cependant, quand l'utilisateur commence à écrire, le message "d'incitation" disparaît et les données de l'utilisateur prennent sa place. Ceci permet l'entrée de données après un très long "guidage" sans déborder de l'affichage.

Remplacez le point-virgule en ligne 20 de notre programme par une virgule et repassez le programme.

Bien sûr, l'entrée de données n'est pas limitée à des nombres comme nos exemples semblaient l'indiquer jusqu'à présent. Il suffit de spécifier une variable à caractère pour entrer des caractères, comme le brillant programme déductif ci-dessous le démontre:

Listage du programme:

```
10 INPUT "ENTREZ VOTRE NOM "; L$  
20 INPUT "ENTREZ VOTRE PRENOM "; F$  
30 I$ = LEFT$(F$, 1) + ". " + LEFT$(L$, 1) + ". "  
40 PAUSE "OH,"  
50 PRINT "VOS INITIALES SONT "; I$
```

Touches utilisées:

1 Ø I N P U T SHIFT " E N T R E Z SPACE
V O T R E SPACE N O M
SPACE SHIFT " SHIFT ;
L SHIFT \$ ENTER

2 Ø I N P U T SHIFT " E N T R E Z SPACE
V O T R E SPACE P R E N O M
SPACE SHIFT "
SHIFT ; F SHIFT \$ ENTER

3 Ø I SHIFT \$ = L E F T SHIFT \$ () F
SHIFT \$ SHIFT , 1
) + SHIFT " . SHIFT " +
L E F T SHIFT \$ () L SHIFT \$
SHIFT , 1) + SHIFT " .
SHIFT " ENTER

4 Ø P A U S E SHIFT " O H SHIFT ,
SHIFT " ENTER

5 Ø P R I N T SHIFT " V O S SPACE
I N I T I A L E S SPACE S O N T SPACE
SHIFT " SHIFT ; I SHIFT \$ ENTER

P.S: Ne vous souciez pas de la ligne 30: on s'y est servi de techniques avancées que vous apprendrez plus tard.

Il est possible d'utiliser l'instruction INPUT pour rassembler et stocker plusieurs valeurs tout aussi bien qu'une seule donnée élémentaire. Pour programmer une opération de ce type, il suffit de dresser une liste des variables destinées à contenir des données pour l'intégrer dans l'instruction INPUT. Chaque variable de la liste devra être séparée des autres par une virgule.

Listage du programme:

```
10 W = 0 : X = 0 : Y = 0 : Z = 0
20 INPUT "ENTREZ 4 NOMBRES", W, X, Y, Z
30 S = W + X + Y + Z : A = S / 4
40 PAUSE "SOMME = " ; S
50 PRINT "MOYENNE " ; A
```

Touches utilisées:

```
1 Ø W = Ø SHIFT : X = Ø SHIFT : Y = Ø
SHIFT : Z = Ø ENTER
2 Ø I N P U T SHIFT " E N T R E Z SPACE
4 SPACE
N O M B R E S SHIFT " SHIFT ,
W SHIFT , X SHIFT , Y SHIFT ,
Z ENTER
3 Ø S = W + X + Y + Z SHIFT : A = S / 4
ENTER
4 Ø P A U S E SHIFT " S O M M E SPACE
= SPACE SHIFT "
SHIFT : S ENTER
5 Ø P R I N T SHIFT " M O Y E N N E SPACE
SHIFT " SHIFT : A ENTER
```

Lors de son passage, ce programme vous incitera à entrer 4 nombres. Le premier nombre que vous écrivez remplacera le symbole de guidage parce que nous avons utilisé une virgule immédiatement après la chaîne de caractères dans l'instruction INPUT. Vous devez appuyer sur **ENTER** après avoir terminé l'écriture du premier nombre. Le point d'interrogation précédera chaque nombre suivant après lesquels vous devrez appuyer chaque fois sur **ENTER**.

Si nous avions employé un point-virgule dans l'instruction INPUT, le premier nombre entré n'aurait pas remplacé le symbole de guidage, mais aurait partagé la ligne avec lui. Les autres entrées successives seraient les mêmes qu'avec la virgule. Faites-en l'essai en changeant la ligne 20 de la manière suivante:

```
20 INPUT "ENTREZ 4 NOMBRES"; W, X, Y, Z
```

Touches utilisées:

```
2 Ø I N P U T SHIFT " E N T R E Z
SPACE 4 SPACE
N O M B R E S SHIFT " SHIFT ;
W SHIFT , X SHIFT , Y SHIFT ,
Z ENTER
```

I. Raccourcis et conseils utiles

Cette partie constitue une petite récompense pour votre patience et votre assiduité. Vous préféreriez probablement de l'argent bien sonnant, mais certains des conseils et données ci-dessous valent leur pesant d'or.

I.1. Abréviations

Ceux d'entre vous dont les doigts sont un peu lents à voler d'une touche à l'autre ont dû remarquer que nos programmes-échantillons sont devenus plus longs très rapidement. Lors de la mise au point du SHARP, ses créateurs ont prévu vos difficultés et ont donné au SHARP la capacité de reconnaître des abréviations d'instructions et de commandes fréquemment utilisées.

De manière générale, une abréviation se constitue d'une ou de plusieurs lettres suivies d'un point. Le point est essentiel pour éviter la confusion avec les noms de variables. Dans l'exemple qui suit, les instructions du programme sont en abrégé:

Listage du programme:

```
15 PA. "BONJOUR, HUMAIN."
25 I. "QUEL EST VOTRE NOM?"; N$
35 P. "ENCHANTE, "; N$
```

Touches utilisées:

```
1 5 P A . SHIFT " B O N J O U R SHIFT ,
H U M A I N . SHIFT " ENTER
2 5 I . SHIFT " Q U E L SPACE E S T SPACE
V O T R E SPACE N O M SHIFT ? SHIFT "
SHIFT : N SHIFT $ ENTER
3 5 P . SHIFT " E N C H A N T E
SHIFT , SPACE
SHIFT " SHIFT :
N SHIFT $ ENTER
```

Lorsque vous appuierez sur ENTER à la fin de chaque ligne, vous noterez que le SHARP développe toutes les abréviations situées sur la ligne. La clarté qui en résulte sera énormément utile plus tard lorsque vous vérifierez s'il n'y a pas d'erreur dans le programme. Comme bénéfice supplémentaire caché, vous trouverez que l'espace nécessaire pour mémoriser les abréviations d'instructions n'est ni plus, ni moins que l'espace nécessaire pour mémoriser les instructions dans leur forme complète. La possibilité d'abréger rend simplement la programmation plus aisée pour l'utilisateur.

Cependant, dans le but d'en faciliter la lecture, nous n'emploierons pas ces abréviations dans le listage de nos programmes-échantillons. Mais cela ne vous empêche pas de vous en servir, bien sûr, quand vous entrez un programme.

Suivent les abréviations acceptables pour des commandes et des instructions que nous connaissons déjà:

PRINT	P. PR. PRI.
PAUSE	PA. PAU.
INPUT	I. IN. INP.
RUN	R.

Une liste complète d'abréviations se trouve dans un appendice à la fin du manuel.

1.2. Les instructions multiples et le double-point

Comme nous l'avons déjà dit auparavant dans la section concernant les numéros de ligne, plusieurs instructions peuvent occuper la même ligne. Le SHARP exécute ces instructions de la gauche vers la droite. Pour permettre au SHARP de distinguer la fin d'une instruction du début de la suivante, un caractère indicateur s'avère nécessaire, et l'on a choisi le double-point (:) pour remplir cette fonction. Sa fonction, dans ce contexte, est semblable à celle du point dans les phrases normales. Il indique un arrêt dans la lecture.

Plusieurs exemples de l'utilisation du double-point ont déjà apparu auparavant dans nos programmes-échantillons. En règle générale, elle prend la forme suivante:

numéro de ligne instruction1 : instruction2 : instruction3 (etc)

"Quand utiliser le double-point?" est une question de style de programmation. L'utilisation à l'aveuglette du double-point dans l'écriture de programmes très denses produit des programmes qui sont très difficiles à lire, à corriger ou à élargir. Et puisque le PC-1500 a comme avantages principaux, la faculté de s'adapter à votre personnalité et la faculté d'agir réciprocement, il est désirable que vous puissiez modifier et élargir des programmes selon vos besoins. Nous vous recommandons donc de faire un usage modéré du double-point.

Il est très profitable de grouper seulement les instructions ayant un rapport conceptuel entre elles quand on se sert du double-point. C'est-à-dire que vous devriez placer sur la même ligne uniquement les instructions destinées à effectuer une tâche particulière parmi la multitude de tâches qui forment le programme.

Comme exercice, examinez les instructions de programme suivantes qui introduisent 3 nombres, trouvez leur moyenne, calculer de combien chacun s'éloigne de cette moyenne et imprimez la somme de leurs différences:

Listage du programme:

```

10 N1 = 0 : N2 = 0 : N3 = 0
20 INPUT "ENTREZ 3 NOMBRES", N1, N2, N3
30 A = (N1 + N2 + N3) / 3
40 D1 = N1 - A : D2 = N2 - A : D3 = N3 - A
50 SD = D1 + D2 + D3
60 PRINT "SOMME DES DIFFERENCES = "; SD

```

Touche utilisées:

```
1 Ø N 1 = Ø SHIFT : N 2 = Ø SHIFT : N 3  
= Ø ENTER  
2 Ø I N P U T SHIFT " E N T R E Z SPACE 3  
SPACE N O M B R E S SHIFT " SHIFT ,  
N 1 SHIFT , N 2 SHIFT , N 3 ENTER  
3 Ø A = ( N 1 + N 2 + N 3 ) / 3 ENTER  
4 Ø D 1 = N 1 - A SHIFT : D 2 = N 2 - A  
SHIFT : D 3 = N 3 - A ENTER  
5 Ø S D = D 1 + D 2 + D 3 ENTER  
6 Ø P R I N T SHIFT " S O M M E SPACE  
D E S SPACE D I F F E R E N C E S  
= SPACE  
SHIFT " SHIFT : S D ENTER
```

Notez que chaque ligne du programme effectue une opération complète et nécessaire. La ligne 10 efface toutes les valeurs que les variables N1, N2 et N3 auraient pu contenir jusqu'à présent (une précaution destinée à prévoir le cas où l'utilisateur oublierait d'entrer les 3 nombres). La ligne 20 recueille les données de l'utilisateur. La ligne 30 fait la moyenne des nombres. La ligne 40 calcule les différences. La ligne 50 fait la somme des différences et la ligne 60 imprime le résultat.

Les instructions de chaque ligne ne correspondent pas accidentellement à la description en langage courant du programme. Bien au contraire, c'est en accord avec les principes de la bonne programmation, principes que nous essayons de vous inculquer.

A la différence de l'utilisation des abréviations, l'utilisation du double-point a un effet sur la quantité de mémoire utilisée pour stocker le programme. Voilà la raison principale de l'utilisation du double-point pour placer plusieurs instructions sur la même ligne. Chaque numéro de ligne réserve plusieurs "phases" (unité de mémoire) de programme et par conséquent, moins il y a de lignes dans un programme, moins on occupe de mémoire dans ce programme.

En conclusion, nous dirons, au sujet de l'emploi du double-point, que chaque programmeur devra garder un équilibre entre la lisibilité et la flexibilité du programme et la mémoire nécessaire pour son application.

J. La correction d'erreurs sur le mode PROgramme

Quoique les abréviations et les double-points nous facilitent l'entrée des programmes, ils ne peuvent empêcher même les meilleurs d'entre nous de faire des erreurs. Même les programmeurs professionnels ne réussissent pas à découvrir leurs propres erreurs quand ils passent en revue leurs programmes. Nous voulons dire par là que tôt ou tard vous rencontrerez une erreur pendant ou après le passage de votre programme. En général ces erreurs peuvent être corrigées facilement, si vous les considérez comme part d'une énigme à résoudre et si vous localisez le problème soigneusement. Vous serez aidé dans cette tâche par plusieurs caractéristiques du PC-1500. Quand le SHARP rencontre une instruction incorrecte, il interrompt ses opérations et vous indique, le problème avec un message laconique comme le suivant.



Dans le but d'illustrer ce cas, passons un programme dans lequel nous avons introduit délibérément une erreur:

Listage du programme:

```
25 PAUSE "TOUT EST BIEN"  
50 PRIMT "QUI FINIT BIEN"
```

Touches utilisées:

```
2 5 P A U S E SHIFT " T O U T SPACE  
E S T SPACE B I E N SHIFT " ENTER  
5 0 P R I M T SHIFT " Q U I SPACE F I N I T  
SPACE B I E N SHIFT " ENTER
```

Passons le programme. Quand le message d'erreur apparaît, appuyez sur la touche. Tant que vous appuierez sur cette touche, la ligne qui a troublé le SHARP apparaîtra sur l'affichage. La "grille clignotante" fournira PEUT-ETRE un indice de la nature du problème.

Pour corriger l'instruction erronée, appuyer sur **CL** pour quitter le programme et passez au mode PROgramme. Appuyez sur la touche, mais ne la maintenez pas en position basse. L'affichage va indiquer à nouveau la ligne erronée:

```
50 PRIMT "QUI FINIT BIEN"
```

Vous pourrez maintenant passer à la mise en forme de cette ligne en vous servant des touches INSérer, DEL (effacer) et les touches à flèches que vous connaissez déjà. N'oubliez-pas d'appuyer sur **ENTER** quand vous avez fini pour ordonner au SHARP de stocker la ligne corrigée.

K. La commande LIST

Une autre façon d'afficher une ligne définie d'un programme, consiste à employer la commande LIST dont la forme est la suivante:

LIST —————→ ou —————

→ LIST numéro de liste

Si l'on ne donne pas de numéro de ligne, la première ligne du programme sera affichée.

Si l'on spécifie un numéro de ligne, la ligne en question apparaîtra sur l'affichage. Si aucune ligne du programme n'a le numéro donné, la ligne suivante avec un numéro plus grand, apparaîtra sur l'affichage. Par exemple, dans le programme suivant:

```
15 PRINT "POLICHINELLE"  
30 PRINT "FUT UN";  
45 PRINT "CLOWN"
```

la commande LIST 40 amènera la ligne 45 sur l'affichage. La commande LIST amène la ligne 15 et la commande LIST 30, la ligne 30.

NOTE: Si vous spécifiez une ligne dont le numéro est plus grand que les numéros de ligne en existence dans le programme, vous obtiendrez le message d'erreur ERROR 11.

L. Plus on est de fous, plus on rit

Comme nous l'avons déjà signalé auparavant, il est possible de garder plus d'un programme à la fois dans la mémoire. La ruse est de donner à chaque programme sa propre gamme de numéros de ligne. Par exemple, on pourra donner à un programme les numéros de 10 à 200 tandis que l'autre recevra les numéros de ligne de 300 à 500. Méfiez-vous, bien sûr, du mélange accidentel d'instructions relevant de programmes différents (au moyen d'un numérotage incorrect), car vous courrez le risque d'obtenir des résultats bizarres.

L.1. L'instruction END

Un autre problème que l'on rencontre quand on mémorise plusieurs programmes simultanément est celui qui découle du fait que chaque programme n'est en fait qu'un groupe d'instructions numérotées qui sont exécutées dans l'ordre ascendant. Comment est-ce que le SHARP saura quand il a terminé l'exécution d'instructions d'un programme donné? La réponse est qu'il ne peut pas le deviner si on ne le lui indique pas. L'instruction dont on se sert pour indiquer à l'ordinateur qu'il est arrivé au bout d'un programme, est l'instruction END.

Jusqu'à présent nous ne nous sommes pas servis et, nous n'avons pas eu besoin de l'instruction END. Le SHARP a toujours exécuté toute les lignes de notre programme dans l'ordre ascendant jusqu'à ce qu'il arrive à la ligne finale. A ce moment, il a conclu que le programme était terminé et s'est remis en position d'attente pour notre prochaine commande. Maintenant, par contre nous voulons indiquer au SHARP qu'il doit s'arrêter d'exécuter les instructions avant de passer au programme suivant.

Entrons les lignes suivantes pour illustrer l'usage de programmes multiples:

```
10 PAUSE "QUE DIT"  
20 PAUSE "UN HOMME"  
30 PRINT "QUI SE NOIE?"  
40 END  
200 PAUSE "IL"  
210 PRINT "DISPARAIT"  
220 END
```

L.2. Numéro de ligne de RUN

Bon, maintenant que vous avez deux programmes dans la mémoire, comment allez vous les passer séparément? Ceux qui ont le goût de l'aventure ont peut-être essayé d'utiliser déjà la commande RUN ordinaire et ont découvert qu'elle convient parfaitement pour démarrer le premier programme. Voyons maintenant la procédure à suivre pour le second programme. Il va falloir que nous employions une variante de la commande RUN qui indiquera au SHARP à quelle ligne il doit commencer. C'est la commande de numéro de ligne de RUN. Pour démarrer le second programme, écrivez:

R U N 2 0 0

et voilà, c'est fait!

Cette commande, comme la plupart des commandes qui nous sont les plus utiles, peut malgré tout nous poser des problèmes. Parce qu'elle indique au SHARP l'endroit à partir duquel il devra exécuter les instructions et parce que SHARP est un serviteur si fidèle, il est possible de démarrer un programme au milieu. Ceci comme vous devez le deviner ne devrait pas être sujet à un choix.

Si on le fait, eh bien . . . les résultats ont tendance à être quelque peu extraordinaires. Faites-en l'essai avec notre premier programme en donnant la commande:

RUN 30

L'erreur est légère dans ce cas, mais imaginez ce qui se passerait dans un programme complexe!

M. Instructions de contrôle

Jusqu'à présent notre programme était constitué d'une séquence d'instructions exécutées une fois par l'ordinateur. Ceux qui ont de l'expérience à donner des instructions se rendent compte peut-être que cela ne constitue pas toujours le meilleur moyen d'accomplir une tâche. Souvent l'on donne à celui qui écoute la possibilité de choisir entre plusieurs plans. Parfois l'on "condense" des instructions avec une commande du type: "Et maintenant retracez les 3 derniers pas jusqu'à ce que vous ayez terminé."

Ces capacités ont été incorporées dans des instructions en BASIC que l'on appelle instructions de contrôle. Elles déterminent si, quand et comment s'effectuera l'exécution des autres instructions. Ces instructions de contrôle permettent de façonner des programmes très puissants et très souples. Dans les sections qui suivent nous allons examiner les IF . . . THEN, les GOTO et les GOSUB qui forment les principales instructions de contrôle du langage BASIC.

N. IF (si) et THEN (alors)

L'instruction IF offre la possibilité de faire des choix durant un programme BASIC. Equipé d'une instruction IF, le programme peut évaluer votre entrée et prendre des décisions:

Listage du programme:

```
10 PAUSE "DORMEZ VOUS?"  
20 INPUT "ESCRIVEZ OUI OU NON "; SX$  
30 IF SX$ = "OUI" THEN PRINT "OH, EXCUSEZ MOI"  
40 END
```

Touches utilisées:

```
1 0 P A U S E SHIFT " D O R M E Z SPACE  
V O U S SHIFT ?  
SHIFT " ENTER  
2 0 I N P U T SHIFT " E S C R I V E Z  
SPACE  
O U I SPACE O U SPACE N O N SPACE  
SHIFT " SHIFT : S X SHIFT $ ENTER  
3 0 I F S X SHIFT $ = SHIFT " O U I SHIFT "  
T H E N P R I N T SHIFT " O H  
SHIFT , SPACE E X C U S E Z SPACE M O I  
SHIFT " ENTER  
4 0 E N D ENTER
```

Ce programme soporifique ne produira une sortie que si vous répondez affirmativement. La forme générale de l'instruction IF, illustrée par la ligne 30, est la suivante:

IF condition THEN instruction

Pendant le passage, l'épreuve renfermée dans la proposition IF est exécutée. L'exécution ou la non-exécution de l'instruction dépend du résultat de l'épreuve. Ordinairement cette épreuve est une fonction logique et on l'appelle "condition". Souvenez-vous que les fonctions logiques sont des comparaisons qui sont soit vraies soit fausses. Si la fonction logique est vraie l'instruction sera exécutée, si elle est fausse, l'instruction sera sautée.

Dans notre programme-échantillon l'épreuve consiste à déterminer si la variable SX\$ est égale à (contient) la chaîne de caractères "F". Si elle est égale-et seulement dans ce cas-l'ordinateur ignorera l'instruction PRINT. Dans les deux cas, que l'instruction PRINT soit exécutée ou non, le SHARP passera à la ligne suivante (qui dans le cas de notre programme est la ligne 40).

Notez que nous aurions pu inverser notre épreuve en modifiant plusieurs lignes de la manière suivante:

```
30 IF SX$ = "NON" THEN END  
40 PRINT "OH, EXCUSEZ MOI"
```

Ce programme n'est pas exactement le même que l'original. Il permet à notre ordinateur de communiquer avec un utilisateur qui fait une faute de frappe ou qui ne répond pas négativement. De plus, ce programme a, en fait, deux fins: l'une déterminée par l'instruction END à la ligne 30 et la fin implicite au-delà de la ligne 40. Ce n'est pas une bonne coutume de programmer plusieurs fins dans un même programme. Cependant, notre renversement de ces instructions démontre qu'un arrangement correct des instructions et une bonne mise à l'épreuve sont nécessaires au fonctionnement correct d'un programme. Dans la prochaine section, nous aurons l'occasion de voir une troisième façon d'exprimer notre programme à l'aide de l'instruction GOTO qui résoudra les problèmes rencontrés durant la seconde.

Bien que la condition d'une instruction IF soit en général une fonction logique, elle ne l'est pas nécessairement. Dans le BASIC du PC-1500, toute expression évaluée comme nombre positif, qui n'est pas un zéro, est considérée vraie. Et inversement, toute expression évaluée comme nombre négatif, comme 5-9, ou équivalente à un zéro est considérée fausse. Ceci explique pourquoi les fonctions logiques fonctionnent comme conditions. Rappelez-vous que les fonctions logiques vraies sont représentées par un 1 et les fonctions logiques fausses par un 0. Cette méthode de représentation n'est pas standard sur tous les ordinateurs et elle peut être la cause d'obscurités dans un programme. Employez-la judicieusement.

Un autre rappel est à l'ordre du jour. Si l'instruction qui suit le THEN est une instruction d'affectation, le mot-clé DOIT être employé. Si l'on néglige de le faire, il en résultera un message d'erreur. Reportez-vous à la section sur l'instruction LET où nous avons examiné ce type de situation.

O. L'instruction GOTO

Dans la section précédente vous avez dû remarquer que nos choix étaient limités après le test dans l'instruction IF. Il nous était permis d'exécuter seulement une instruction si la fonction logique était vraie. Par souci de commodité nous voudrions exécuter plusieurs instructions et c'est l'instruction GOTO qui va nous le permettre.

L'instruction GOTO modifie la "course" de l'exécution de l'instruction. Elle ordonne au SHARP "d'aller à" ("go to" en anglais) une ligne autre que celle qui suivrait normalement et de commencer l'exécution d'instructions dans l'ordre à partir de cette ligne. Par conséquent, on saute certaines instructions complètement. Voyons cela en passant en revue le programme qui suit:

Listage du programme:

```
10 PAUSE "EVITEZ ";
20 GOTO 50
30 PRINT X * 3 / 4 + 2
40 PRINT "TOUT LIVRE AYANT ";
50 PRINT "DES ERREURS!"
60 END
```

Touches utilisées:

```
1 Ø P A U S E SHIFT " E V I T E Z
    SPACE SHIFT " SHIFT : ENTER

2 Ø G O T O 5 Ø ENTER

3 Ø P R I N T X * 3 / 4 + 2 ENTER

4 Ø P R I N T SHIFT " T O U T SPACE L I V
    R E
    SPACE A Y A N T
    SHIFT " SHIFT : ENTER

5 Ø P R I N T SHIFT " D E S SPACE
    E R R E U R S SHIFT ! SHIFT " ENTER

6 Ø E N D ENTER
```

Ordinairement l'exécution de ce programme s'effectuerait dans l'ordre des numéros de ligne du plus petit au plus grand numéro. Le GOTO entré dans la ligne 20 a pour effet de forcer le SHARP à passer tout de suite à la ligne 50 et de commencer l'exécution à partir de là. Les lignes 30 et 40 ne seront pas exécutées.

La forme générale de l'instruction GOTO est la suivante:

GOTO expression

dans laquelle:

L'expression est évaluée à un nombre qui est un numéro valide de ligne d'un programme (c'est-à-dire de 1 à 65279).

NOTE: Si l'on spécifie un numéro de ligne qui n'existe pas, on obtient un message d'erreur ERROR 11.

Si, en conséquence d'un certain choix, nous décidons de faire exécuter plusieurs instructions, nous devons nous servir d'instructions GOTO en conjonction avec l'instruction IF.

```

10 IF (si) test THEN GOTO (alors allez à) 100
20 { instructions ici sont
  : { exécutées seulement si
  : { le test est faux
}
90 GOTO 200
100 { instructions ici sont
  : { exécutées seulement si
  : { le test est vrai
}
200 { instructions ici sont
  : { toujours exécutées
}
999 END

```

La logique de ce programme est applicable dans de nombreuses situations. Il est possible d'insérer autant d'instructions que l'on veut dans chacune des sections formées par les instructions GOTO.

Comme exemple de cette structure dans la pratique, le segment de programme de compte courant qui suit, détermine si le montant d'une entrée est un dépôt (nombre positif) ou un retrait (nombre négatif). L'utilisateur entre un solde initial:

<p><u>Listage du programme:</u></p> <pre> 10 INPUT "SOLDE INITIAL?", B 20 INPUT "MONTANT?", TA 25 IF TA = 0 THEN GOTO 80 30 B = B + TA 40 IF TA < 0 THEN GOTO 70 50 PRINT "DEPOT DE \$"; TA; " INSCRIT" 60 GOTO 80 70 PRINT TA;" DOLLARS RETIRES" 80 PRINT "SOLDE FINAL = "; B 90 END </pre>

Touches utilisées:

1	0	I	N	P	U	T	SHIFT	"	S	O	L	D	E		
SPACE I N I T I A L															
SHIFT ? SHIFT " SHIFT , B ENTER															
2	0	I	N	P	U	T	SHIFT	"	M	O	N	T	A	N	T
SHIFT ? SHIFT " SHIFT , T A ENTER															
2	5	I	F	T	A	=	0	T	H	E	N	G	O	T	O
8 0 ENTER															
3	0	B	=	B	+	T	A	ENTER							

```

4 0 I F T A SHIFT < 0 T H E N
G O T O 7 0 ENTER

5 0 P R I N T SHIFT " D E P O T
SPACE D E SPACE SHIFT $
SHIFT " SHIFT ; T A SHIFT ;
SHIFT " SPACE I N S C R I T SHIFT "
ENTER

6 0 G O T O 8 0 ENTER

7 0 P R I N T T A SHIFT ; SHIFT "
SPACE D O L L A R S SPACE R E
T I R E S SHIFT " ENTER

8 0 P R I N T SHIFT " S O L D E
SPACE F I N A L = SPACE
SHIFT " SHIFT ; B ENTER

9 0 E N D ENTER

```

Comme vous le voyez, ce programme n'a de fin que si l'opération se ramène à zéro. La deuxième instruction IF illustre la structure mentionnée auparavant. Remarquez que, en plus des deux actions différentes associées avec l'instruction IF (ligne 50 et 70), il y a d'autres instructions (ligne 80 et 70) que l'ordinateur exécute sans tenir compte du résultat de la mise à l'épreuve du IF.

Maintenant nous pouvons passer à l'écriture d'une troisième version du programme de la section précédente:

Listage du programme:

```

10 PAUSE "DORMEZ VOUS?"
20 INPUT "ECRIVEZ OUI OU NON "; SX$
30 IF SX$<>"OUI" THEN GOTO 99
40 PRINT "OH, EXCUSEZ MOI"
99 END

```

Toiles utilisées:

```

1 0 P A U S E SHIFT " D O R M E Z SPACE
V O U S SHIFT ?
SHIFT " ENTER

2 0 I N P U T SHIFT " E C R I V E Z SPACE
O U I SPACE O U SPACE N O N SPACE
SHIFT " SHIFT ; S X SHIFT $ ENTER

3 0 I F S X SHIFT $ SHIFT < SHIFT >
SHIFT " O U I SHIFT " T H E N
G O T O 9 9 ENTER

```

```

4 Ø P R I N T SHIFT " O H SHIFT ,
SPACE E X C U S E Z SPACE M O I
SHIFT "
ENTER

9 9 E N D ENTER

```

Cette variante inverse l'œuvre conditionnelle en spécifiant qu'une certaine opération doit être effectuée si l'entrée faite par l'utilisateur n'est PAS égale au "OUI" ("YES"). Elle élimine de ce fait les problèmes de la deuxième version et l'on pourrait la développer dans la forme élargie de l'instruction IF décrite ci-dessus. Ce type de réarrangement d'instructions s'avère très utile en programmation. Si vous prenez la peine de faire l'expérience par vous-même, vous obtiendrez de meilleurs programmes comme récompense.

Un autre usage très commun de l'instruction GOTO consiste à lui faire répéter l'exécution d'une série d'instructions. Cette opération s'appelle un "bouclage". Le programme ci-dessous illustre un bouclage simple:

Listage du programme:

```

10 WAIT 30
20 PRINT "TEUF TEUF TEUF TEUF"
30 PRINT "      TEUF TEUF TEUF TEUF"
40 GOTO 20

```

Touches utilisées:

```

1 Ø W A I T 3 Ø ENTER
2 Ø P R I N T SHIFT " T E U F SPACE
T E U F SPACE T E U F SPACE T E U F
SHIFT " ENTER
3 Ø P R I N T SHIFT " SPACE SPACE
SPACE SPACE T E U F SPACE T E U F
SPACE T E U F SPACE T E U F SHIFT "
ENTER
4 Ø G O T O 2 Ø ENTER

```

Malheureusement ce programme continuera à faire teuf-teuf à perpétuité. (Vous pouvez l'arrêter à l'aide de la touche BREAK). C'est à nous de fournir un moyen de faire s'arrêter de tels programmes. Grâce à l'emploi d'un "compteur" avec l'instruction IF, nous pouvons réaliser cela. Le compteur est une variable sous laquelle nous notons combien de fois nous avons fait quelque chose (c'est-à-dire que nous "comptons"). Grâce à cette technique, l'instruction IF vérifie si l'ordinateur a exécuté nos instructions PRINT un nombre de fois déterminé. C'est à nous, bien sûr, de déterminer combien de fois. Choisissons 10 fois pour chaque instruction PRINT en guise de vérification (au delà de 10 cela deviendrait ennuyeux).

Servons-nous du compteur et de l'instruction IF pour écrire:

Listage du programme:

```
10 WAIT 30
15 C = 1
20 PRINT 'TEUF TEUF TEUF TEUF'
30 PRINT "      TEUF TEUF TEUR TEUF"
40 C = C + 1
50 IF C <= 10 THEN 20
60 END
```

Touches utilisées:

```
1 0 W A T 3 0 ENTER
1 5 C = 1 ENTER
2 0 P R I N T SHIFT " T E U F SPACE
T E U F SPACE T E U F SPACE T E
U F SHIFT " ENTER
3 0 P R I N T SHIFT " SPACE SPACE
SPACE SPACE T E U F SPACE T E U F
SPACE T E U F SPACE T E U F SHIFT "
ENTER
4 0 C = C + 1 ENTER
5 0 I F C SHIFT < = 1 0 T H E N
2 0 ENTER
6 0 E N D ENTER
```

Examinez le fonctionnement du compteur lors de l'exécution de chaque boucle et vous remarquerez que nous devons affecter une valeur initiale au compteur à la ligne 15, puis l'augmenter à la ligne 40 pour refléchir une exécution de plus des instructions 20 et 30.

Il y a beaucoup d'autres possibilités d'emploi des compteurs et des boucles au cours des programmes. Malheureusement l'espace nous manque ici pour les décrire. Il vous faudra vous reporter à l'un des livres listés dans l'Appendice F.

L'instruction GOTO est une commande en même temps. Son usage en tant que commande diffère naturellement de son usage en tant qu'instruction. Lancée comme commande sur le mode RUN, le GOTO débute l'exécution d'un programme d'une manière identique à la commande RUN. Nous trouverons la différence dans certains préparatifs internes qui sont faits avant l'exécution des instructions du programme. (Pour une comparaison des méthodes utilisées pour démarrer un programme, reportez-vous à la section intitulée "Commencer l'exécution d'un programme" dans le Chapitre V). A la différence de la commande RUN, la commande **GOTO n'efface pas les valeurs mémorisées dans les variables avant de commencer l'exécution d'un programme.**

Pour démarrer l'exécution d'un programme à l'aide de la commande GOTO, entrez:

GOTO numéro de ligne

où:

numéro de ligne est constitué par le numéro de la première ligne du programme destiné à être exécuté.

NOTE: Si vous spécifiez un numéro de ligne qui n'existe pas, vous obtiendrez un message d'erreur ERROR 11.

P. FOR ... NEXT

Comme nous l'avons vu dans la section précédente, il est très utile de pouvoir répéter une série d'instructions. En fait, on exploite cette faculté tellement souvent qu'on a incorporé dans le BASIC plusieurs instructions qui rendent cette opération automatique. Ce sont l'instruction FOR et son partenaire, l'instruction NEXT. Ensemble, ces deux instructions renferment une série d'instructions qui sont répétées plusieurs fois. L'instruction FOR est associée avec un compteur et comprend un test conditionnel intégré. Elle permet aussi de spécifier la valeur initiale et l'augmentation de la valeur de la variable-compteur.

La forme de toutes ces données est la suivante:

FOR variable-compteur = valeur initiale TO valeur finale STEP valeur ajoutée

où:

"Variable-compteur" constitue le nom de la variable qui sert à compter les boucles.

la valeur initiale est constituée par la valeur mémorisée sous la variable-compteur avant le premier passage par la boucle. La gamme permise pour cette valeur va de -32768 à 32767.

la valeur finale constitue le nombre employé dans le test. Si la variable-compteur contient une valeur supérieure à la valeur finale, le bouclage est terminé. La gamme permise pour ce nombre va de -32768 à 32767.

STEP valeur ajoutée constitue une proposition facultative. La valeur ajoutée indique de combien on augmente ou diminue la variable-compteur à chaque passage par la boucle. Sa gamme permise va de -32768 à 32767. Si l'on omet entièrement cette proposition, on considère que la valeur ajoutée est égale à un.

Comme cela doit paraître un peu difficile à assimiler, passons à l'observation de programmes-échantillons simples. Le premier ressemble à celui où nous avons employé le compteur. Au lieu de faire TEUF-TEUF, nous imprimons la valeur de la variable-compteur C:

```
15 FOR C = 1 TO 10  
30 PAUSE C  
50 NEXT C
```

(Pour obliger ce programme à faire "TEUF" comme auparavant, insérez simplement les instructions 10, 20 et 30 de ce programme). Remarquez que cette version est plus propre et plus concise parce qu'elle utilise moins d'instructions pour effectuer les mêmes fonctions de comptage et de bouclage que la version précédente.

Dans le cas où il y aurait toujours encore des malentendus au sujet de l'action des instructions FOR et NEXT, nous vous présentons une comparaison d'une boucle FOR ... NEXT et les instructions équivalentes:

10 FOR I = 1 TO 8	10 I = 1
20 [quelques instructions à répéter]	12 IF I > 8 THEN 100
:	20 [quelques instructions à répéter]
90 NEXT I	90 I = I + 1
100 END	92 GOTO 12
	100 END

Notre second programme-échantillon illustre la manière de programmer une "boucle générale" dont les répétitions sont contrôlées par la valeur d'une variable. Nous demandons d'abord à l'utilisateur combien de nombres il ou elle pensent entrer. Nous mémorisons ce nombre sous une variable N et nous passons la boucle par les instructions qui prennent un numéro pour le traitement. La boucle est exécutée N fois, à moins que N soit égal ou inférieur à zéro. Dans ce cas-là, nous passons à END sans traitement aucun.

Listage du programme:

```

10 N = 0 : V = 0 : T = 0 : A = 0
20 WAIT 0
30 INPUT "COMBIEN DE VALEURS? "; N
40 IF N = 0 THEN GOTO 999
50 FOR I = 1 TO N
60 CLS : CURSOR 0
70 INPUT V
80 T = T + V
90 NEXT I
100 WAIT : CLS : CURSOR 0
110 A = T / N
120 PAUSE "TOTAL = "; T
130 PRINT "MOYENNE = "; A
999 END

```

Touches utilisées:

```

1 0 N = 0 SHIFT : V = 0 SHIFT : T = 0
SHIFT : A = 0 ENTER
2 0 W A I T 0 ENTER
3 0 I N P U T SHIFT " C O M B I E N SPACE
D E SPACE V A L E U R S SHIFT ? SPACE
SHIFT " SHIFT : N ENTER
4 0 I F N = 0 T H E N G O T O
9 9 9 ENTER

```

```

5 0 F O R I = 1 T O N ENTER
6 0 C L S SHIFT : C U R S O R 0 ENTER
7 0 I N P U T V ENTER
8 0 T = T + V ENTER
9 0 N E X T I ENTER
1 0 0 W A I T SHIFT : C L S SHIFT :
C U R S O R 0 ENTER
1 1 0 A = T / N ENTER
1 2 0 P A U S E SHIFT " T O T A L
SPACE = SPACE SHIFT " SHIFT ;
T ENTER
1 3 0 P R I N T SHIFT " M O Y E N N E
SPACE = SPACE SHIFT " SHIFT ;
A ENTER
9 9 9 E N D ENTER

```

Il n'est pas nécessaire d'augmenter la variable-compteur par un, ni d'utiliser 1 comme valeur initiale. A l'aide de la proposition STEP, le programmeur peut spécifier la taille de l'augmentation (ou de la diminution). Notre programme suivant démontre cela sur le ton d'un slogan publicitaire:

Listage du programme:

```

10 WAIT 30
20 FOR HS=2 TO 8 STEP 2
30 PRINT HS;"!";
40 NEXT HS
50 WAIT 60 : CLS : CURSOR 0
60 PRINT "QUI ESTIMONS NOUS?"
70 PRINT "LE SHARP PC-1500!"
80 END

```

Touches utilisées:

```

1 0 W A I T 3 0 ENTER
2 0 F O R H S = 2 T O 8 S T E P
2 ENTER
3 0 P R I N T H S SHIFT ; SHIFT "
SHIFT ! SPACE SHIFT " SHIFT ;
ENTER
4 0 N E X T H S ENTER
5 0 W A I T 6 0 SHIFT : C L S SHIFT :
C U R S O R 0 ENTER

```

```

6 0 P R I N T SHIFT " Q U I SPACE
E S T I M O N S SPACE N O U S
SHIFT ? SHIFT " ENTER
7 0 P R I N T SHIFT " L E SPACE
S H A R P SPACE P C - 1 5 0 0
SHIFT ! SHIFT " ENTER
8 0 E N D ENTER

```

En plus du comptage dans l'ordre normal la proposition STEP permet aussi au SHARP de compter à rebours. Pour effectuer cette opération, inversez les valeurs initiales et finales et spécifiez une augmentation négative. Le programme suivant (dédié aux consommateurs du monde entier) illustre cette opération:

Le Bazar du Tapis Roulant liquide son stock à des prix extraordinaires. Toute une série de tapis ronds sont offerts à 0,99\$ le mètre carré. Le rayon des tapis varie entre 1 et de 40 mètres. Entre chaque tapis et celui qui suit en taille (en plus petit) il y a une différence de rayon de 3 mètres. M. Tournanron qui a besoin de tapis neufs, décide de calculer les prix de chaque tapis à l'aide d'un programme de son fidèle SHARP PC-1500. Il écrit le programme suivant:

```

10 FOR R = 40 TO 1 STEP -3
20 P = .99 * (PI * R ^ 2)
30 PRINT R; " LE RAYON EST $"; P
40 NEXT R
50 END

```

et trouve que les prix varient entre 4976,2\$ et 3,11\$.

Q. WAIT

L'instruction WAIT sert à modifier l'opération de l'instruction PRINT. Elle spécifie la période pendant laquelle des données affichées à l'aide de l'instruction PRINT resteront sur l'affichage.

Le format de l'instruction WAIT est comme suit:

WAIT argument

L'argument est facultatif. Si aucun argument n'est spécifié, la période de défaut est "infinie", ce qui veut dire que les données resteront affichées jusqu'à ce que l'utilisateur appuie sur **ENTER**. C'est le mode sur lequel la plupart de nos programmes opèrent.

Si l'on offre un argument, toutes les instructions PRINT qui suivent tiendront leurs données sur l'affichage pendant une période proportionnelle au numéro spécifié comme argument. Ce type d'instruction PRINT ressemble à une instruction PAUSE, excepté que la période de l'instruction PAUSE est fixée. Notez que l'instruction WAIT n'a aucun effet sur l'opération de l'instruction PAUSE.

Le numéro (ou expression qui a pour résultat un numéro) donné comme argument a une gamme permise allant de 0 à 65535. L'instruction WAIT 0 affiche les données pour une période si courte qu'on ne peut pratiquement pas la lire. Et WAIT 65535 affichera les données chaque instruction PRINT pendant à peu près 17 minutes! Plus concrètement parlant, WAIT 64 équivaut à une période d'environ une seconde et WAIT 3840, à environ une minute. Pour retourner à l'opération ordinaire de l'instruction PRINT (de façon à ce que l'ordinateur attende que l'utilisateur appuie sur ENTER), employez l'instruction WAIT sans argument.

Programme de démonstration

Ce programme sert à illustrer les effets de l'instruction WAIT sur les instructions PRINT qui la suivent. Nous allons varier ici la période de WAIT de 0 à 102 par additions de 2 durant l'impression de périodes en boucle. Le BEEP (avertissement sonore) est là uniquement pour vous aider à comprendre l'intervalle de temps car le passage sur l'affichage est difficile à capturer avec les yeux à cause de la grande vitesse.

Listage de programme:

```
10 FOR W = 0 TO 102 STEP 2
20 WAIT W
30 BEEP 1,5
40 PRINT ".";
50 NEXT W
60 END
```

Toiles utilisées:

```
1 Ø F O R W = Ø T O 1 Ø 2 S T E P
2 ENTER
3 Ø W A I T W ENTER
4 Ø B E E P 1 SHIFT , 5 ENTER
5 Ø P R I N T SHIFT " + SHIFT "
6 Ø N E X T W ENTER
7 Ø E N D ENTER
```

R. READ, DATA, RESTORE

L'utilisateur du programme n'est pas nécessairement celui qui entre toutes les données d'un programme. Souvent, les données le plus utiles sont relativement statiques, comme par exemple les tables d'impôts dans le cas d'une application dans le domaine financier, ou encore les constantes de charge dans le cas d'une application technique. On peut insérer ce type de données dans un programme et s'en servir quand on en a besoin grâce aux instructions DATA, READ et RESTORE. Ces instructions agissent de concert pour spécifier les données employées dans un programme, transférer des données à des variables et répéter l'opération si nécessaire.

L'instruction DATA est constituée du mot-clé DATA suivi d'une liste de données élémentaires comprenant des nombres en notation scientifique ou réelle et des chaînes de caractères. Les éléments de la liste sont séparés par des virgules. Des instructions DATA peuvent être placées n'importe où dans le programme, mais beaucoup de programmeurs préfèrent les grouper au début du programme, ce qui permet de les trouver plus facilement lors de la lecture du programme.

Une instruction DATA typique ressemblera à la ligne suivante:

```
10 DATA "BALZAC", 20000, "ROMANCIER", "M", 112
```

L'instruction READ est constituée du mot-clé READ suivi d'une liste de noms de variables. Ces noms sont soit des noms de variables numériques soit des noms de variables à caractères. Sur la liste, les noms de variables sont séparés par des virgules. L'instruction READ sert à faire "lire" une ou plusieurs données élémentaires d'une instruction DATA et à la/les mémoriser avec les variables associées. Ce qui suit est une instruction READ qui correspond à notre instruction DATA précédente:

```
120 READ N$, WT, C$, SX$, L
```

Chaque fois qu'une instruction READ est exécutée, le SHART exige qu'il y ait une donnée élémentaire correspondante dans une instruction DATA. Par exemple, le programme suivant produira une erreur en ligne 30 parce que toutes les données élémentaires ont été épuisées par l'instruction READ en ligne 20:

```
10 DATA 1, 2, 3  
20 READ A, B, C  
30 READ D
```

Pour corriger l'erreur, nous pouvons ajouter une donnée élémentaire à la ligne 10:

```
10 DATA 1, 2, 3, 65
```

ou nous pouvons employer une instruction DATA différente n'importe où dans le programme:

```
10 DATA 1, 2, 3  
20 READ A, B, C  
30 READ D  
40 DATA 65
```

Ceci illustre le fait que le SHARP considère toutes les instructions DATA du programme comme une liste unique de données élémentaires. Au fur et à mesure que l'ordinateur rencontre un nom de variable dans une instruction READ, il affecte la donnée élémentaire suivante de la liste à cette variable. Si le SHARP ne peut pas saisir une demande de donnée, il arrête le programme et envoie un message d'erreur. Les éléments en surplus à la fin naturelle du programme sont ignorés.

Si le type (de caractère ou numérique) de l'élément suivant ne correspond pas au type de la variable à remplir, il se produira une erreur. Un bon programmeur groupe les données élémentaires dans des instructions DATA différentes, correspondant chacune à une instruction READ du programme. Ceci est illustré dans le programme suivant qui lit trois données élémentaires quatre fois:

```
10 DATA 1, "A", 1  
20 DATA 2, "B", 3  
30 DATA 5, "C", 8  
40 DATA 13, "D", 21  
50 FOR I = 1 TO 3  
60 READ A, A$, Z  
70 T = T + A * Z  
80 NEXT I
```

Nous aurions pu écrire les lignes 10 à 40 de la manière suivante:

```
10 DATA 1, "A", 1, 2, "B", 3, 5, "C", 8, 13, "D", 21
```

ou même ainsi:

```
10 DATA 1  
20 DATA "A"  
30 DATA 1  
40 DATA 2  
  
:  
(etc)
```

Chacune de ces formes alternées cache le fait que l'instruction READ lit chaque fois les trois données élémentaires. Les formes alternées rendent aussi plus difficile la vérification des types de données élémentaires puisque le schéma: nombre, caractère, nombre n'est pas visible immédiatement.

Parfois il est désirable de réutiliser quelques unes ou toutes les valeurs des instructions DATA. L'instruction RESTORE nous permet d'effectuer cette opération. L'instruction RESTORE oblige SHARP à réutiliser des données élémentaires en commençant par la première instruction DATA du programme. De cette façon, toute instruction READ effectuée après RESTORE recevra des données élémentaires utilisées auparavant.

L'instruction RESTORE peut aussi spécifier une réutilisation plus sélective de données élémentaires. L'instruction:

RESTORE numéro de ligne

provoquera la ré-affectation de données élémentaires à partir de l'instruction DATA de la ligne spécifiée. Par exemple:

```
10 DATA .8, 4  
15 DATA 1,3,6,8E2  
100 READ X, Y  
110 READ M,N,O,P  
120 RESTORE 15  
130 READ A,B
```

A la fin de ce programme les valeurs des variables A et B seront de 1 et 3, respectivement.

En plus des numéros de ligne, on peut "étiqueter" les instruction DATA avec un caractère unique. Puis, à l'aide de l'instruction RESTORE, on pourra remettre en circulation des données élémentaires à partir de l'instruction DATA ayant une étiquette donnée. Pour un exemple d'instruction DATA étiquetée ainsi, voyez l'exemple ci-dessous:

```
20 "A" : DATA 1, -12.2,8
```

Le segment de programme qui suit rend (RESTORE = rendre, minuscule) à l'instruction DATA marquée "X":

```
10 DATA 4.2,3,1  
20 "X" : DATA -2,0,3,5  
100 READ Q,Y,Z  
110 READ MA,MB,MC,MD
```

120 RESTORE "X"

130 READ N, Z

A la fin du programme, N contient -2 et Z contient 0.

S. REM (REMARQUE)

L'instruction REM fournit la possibilité d'insérer des observations entre les instructions d'un programme. Quoique le SHARP ignore ces observations, elles sont très importantes, car elles aident d'autres utilisateurs à lire votre programme. Plus votre programme sera lisible et compréhensible, plus d'autres programmeurs voudront s'en servir et, peut-être, l'améliorer.

Pour alléger le travail de l'impression, nous avons omis de traiter des observations dans ce manuel. Nous vous recommandons de ne pas suivre notre exemple. Les remarques sont probablement de la plus haute importance quand vous imprimez, enregistrez ou mettez un programme en commun. Ne vous fiez pas à votre mémoire pour vous souvenir de la signification de chaque variable d'un programme, utilisez l'instruction REM. Si vous ne le faites pas, vous l'aurez oubliée dans six mois.

Les observations qui suivent le mot-clé REM, peuvent être insérées dans leur propre ligne ou à la fin d'une autre instruction ou série d'instructions. Il ne faut pas cependant les placer avant ou au milieu d'instructions destinées à être exécutées, parce que quand le SHARP tombe sur le mot-clé REM, il ignore le reste des caractères sur la ligne. Si l'on utilisait REM au début d'une ligne, des instructions de programmes valables seraient ignorées.

T. GOSUB et RETURN

Quand vous commencerez à projeter des programmes, vous découvrirez que vous utilisez certaines fonctions plusieurs fois au cours du même programme. Par exemple, cela pourrait consister à calculer la surface d'un cercle ou de recevoir et vérifier un nombre donné par l'utilisateur. Une telle répétition cause une duplication des instructions qui effectuent la fonction. Pour éviter une duplication inutile le programmeur peut utiliser l'instruction GOSUB.

L'instruction GOSUB permet de mettre de côté un groupe d'autres instructions employées dans plusieurs endroits du programme. On appelle ce groupe une "sous-routine" ("subroutine" en anglais d'où GOSUB). A chaque point du programme où le groupe d'instructions devrait être placé, on insère une instruction GOSUB.

Cette instruction GOSUB signale au SHARP de commencer à exécuter le groupe d'instructions mis de côté. On appelle cela "appeler une sous-routine". L'instruction GOSUB est semblable à l'instruction GOTO en ce qu'elle entraîne un changement dans le cours normal séquentiel de l'exécution. Il y a une différence cependant: avant que le SHARP commence à effectuer les instructions de la sous-routine, il "se souvient" de l'endroit où il s'était arrêté. Cela s'appelle "retourner" d'une sous-routine.

Mais comment est-ce que le SHARP discerne la fin des instructions qui forment la sous-routine? La réponse est qu'on doit l'en informer au moyen de l'instruction RETURN. Le schéma de l'instruction GOSUB est:

GOSUB numéro de ligne

numéro de ligne qui est constitué par le numéro de la première ligne de la sous-routine. Le schéma de l'instruction RETURN est simplement:

RETURN

Comme exemple pratique de la sous-routine, considérez un programme à traiter et comparez la surface de deux rectangles dont la longueur et la largeur sont données:

Listage du programme:

```
10 REM READ IN LONGUEUR ET
20 REM LARGEUR DE DEUX RECTANGLES
30 FOR I = 1 TO 2
40 PAUSE "RECTANGLE "; I; ":" "
50 INPUT "ENTREZ LONGUEUR, LARGEUR", L, W
60 GOSUB 200
70 IF I = 1 THEN LET A1 = A
80 IF I = 2 THEN LET A2 = A
90 NEXT I
100 REM PRINT AND COMPAREZ LES
110 REM AIRES DES RECTANGLES.
120 PRINT "AIRE DU RECTANGLE 1 "; A1
130 PRINT "AIRE DU RECTANGLE 2 "; A2
140 IF A1 > A2 THEN 170
150 PRINT "AIRE DE 2 EST > AIRE 1"
160 GOTO 180
170 PRINT "AIRE DE 1 EST > AIRE 2"
180 END
200 REM SOUS-ROUTINE POUR CALCULER AIRE
210 REM DU RECTANGLE, DONNE
220 REM LA LONGUEUR ET LA LARGEUR
230 A = L * W
240 RETURN
```

Notez l'emplacement de la sous-routine. Elles doivent toujours être placées après l'instruction END finale du programme principal. Ceci évite leur usage accidentel lors de l'opération ordinaire d'exécution séquentielle. Chaque sous-routine **DOIT** se terminer par une instruction RETURN.

On peut introduire n'importe quelle instruction permise dans une sous-routine et lui faire accomplir n'importe quel traitement. Les bons programmeurs BASIC conçoivent leurs programmes comme des série de pièces ou jeux de "modules". En général une sous-routine sert à transcrire un module. Le programme principal sert à contrôler l'ordre dans lequel les sous-routines sont exécutées. Pour plus de renseignements sur ce sujet, reportez-vous à l'un des livres sur la programmation avancée listé dans l'Appendice F.

Résumé des caractéristiques de mise en page sur le mode PROgramme

- 1) Numérotez les lignes de programme par intervalles d'au moins 10 unités. Cela permet d'insérer une ligne supplémentaire entre deux lignes existantes en choisissant simplement un numéro de ligne situé entre elles. Par exemple, on peut insérer entre les lignes 40 et 50 une ligne avec un numéro choisi entre 41 et 49.
- 2) Pour effacer une ligne, il suffit d'écrire son numéro et d'appuyer sur **ENTER**.
NOTE: Il faut être très prudent quand on efface une ligne car, souvent, plusieurs instructions de programme sont groupées sur une ligne.
- 3) Les touches à flèche dirigée vers le haut et les touches à flèche dirigée vers le bas peuvent être utilisées pour le défilement vers le haut ou le bas à travers le programme en cours, une ligne à la fois. On obtient une répétition automatique du mouvement si l'on maintient l'une ou l'autre touche en position basse.
- 4) On peut se servir de la commande LIST pour passer directement à une ligne donnée. La commande LIST affichera la première ligne du premier programme en mémoire. LIST N (N étant un numéro) affichera la ligne N ou s'il n'y a pas de ligne avec le numéro N, la première ligne dont le numéro est plus grand que N.
- 5) Une fois qu'une ligne est affichée, on peut se servir des touches à flèche vers la gauche et la droite pour placer le curseur en un endroit quelconque sur la ligne. Ensuite on pourra utiliser les fonctions INSérer et DEL (effacer) pour effectuer des modifications.
NOTE: Si l'on modifie une ligne, il est nécessaire d'appuyer sur ENTER avant d'afficher la ligne suivante, sinon les modifications seront annulées (ne seront pas faites).
- 6) Si l'on rencontre une erreur dans un programme pendant son exécution, la touche à flèche vers le haut "se souviendra" de la ligne où cette erreur se trouve. Pour rappeler la ligne passez sur le mode PROgramme et appuyez sur la touche à flèche vers le haut.

Note: L'aire programme sert à la mémorisation des données aussi bien que des programmes. S'il y a des chances que la somme programme-données dépasse la capacité de cette aire, entrez **65279 END** dans le programme.

IV. CALCULS AVANCES

A. Notation Scientifique

Pour entrer un nombre en notation scientifique ($A \times 10^B$), entrez d'abord en mantisse, puis appuyez sur la lettre E et entrez l'exposant.

Exemple 1: Pour entrer $6,7 \times 10^8$ au clavier:

Touches utilisées	Affichage
6 . 7	6. 7_
E	6. 7E_
8	6. 7E8_

Exemple 2: Pour entrer $-9,12 \times 10^{34}$ au clavier:

<u>Touches utilisées</u>	<u>Affichage</u>
<input type="button" value="-"/> <input type="button" value="9"/> <input type="button" value="."/> <input type="button" value="1"/> <input type="button" value="2"/>	-9. 12_
<input type="button" value="E"/>	-9. 12E_
<input type="button" value="-"/> <input type="button" value="3"/> <input type="button" value="4"/>	-9. 12E-34_

Seuls les 10 premiers chiffres de la mantisse sont importants (Voyez l'exemple). Pour un nombre plus petit que 1, mais plus grand que -1, les données sont exactes jusqu'à un maximum de 10 chiffres.

Exemple 3: Introduire 1234567898765:

<u>Touches utilisées</u>	<u>Affichage</u>
<input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> <input type="button" value="6"/> <input type="button" value="7"/> <input type="button" value="8"/>	
<input type="button" value="9"/> <input type="button" value="8"/> <input type="button" value="7"/> <input type="button" value="6"/> <input type="button" value="5"/>	1234567898765_
<input type="button" value="ENTER"/>	1. 234567898E 12

Exemple 4: Introduire 9.87654321234:

<u>Touches utilisées</u>	<u>Affichage</u>
<input type="button" value="9"/> <input type="button" value="."/> <input type="button" value="8"/> <input type="button" value="7"/> <input type="button" value="6"/> <input type="button" value="5"/> <input type="button" value="4"/>	
<input type="button" value="3"/> <input type="button" value="2"/> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/>	9. 87654321234_
<input type="button" value="ENTER"/>	9. 876543212

Exemple 5: Introduire 0.000000002345678:

<u>Touches utilisées</u>	<u>Affichage</u>
<input type="button" value="."/> <input type="button" value="0"/>	
<input type="button" value="0"/> <input type="button" value="0"/> <input type="button" value="0"/> <input type="button" value="0"/> <input type="button" value="2"/> <input type="button" value="3"/>	
<input type="button" value="4"/> <input type="button" value="5"/> <input type="button" value="6"/> <input type="button" value="7"/> <input type="button" value="8"/>	. 000000002345678_
<input type="button" value="ENTER"/>	2. 345678E-10

Exemple 6: Introduire $0.00001234567 \times 10^{24}$:

<u>Touches utilisées</u>	<u>Affichage</u>
• 0 0 0 0 1 2	
3 4 5 6 7 E 2 4	. 00001234567E24_
ENTER	1. 234567E 19

Remarquez que dans les exposants seuls les trois chiffres écrits à la fin ont de l'effet.

Exemple 7: Introduire 3×10^{123} :

<u>Touches utilisées</u>	<u>Affichage</u>
3 E 1 2 3	3E123_
ENTER	3E 23

Exemple 8: Introduire 4×10^{-3210} :

<u>Touches utilisées</u>	<u>Affichage</u>
4 E - 3 2 1 0	4E-3210_
ENTER	4E-10

B. Gamme des calculs: débordement supérieur et inférieur

La plupart des appareils opèrent à l'intérieur d'une certaine gamme de nombres. Le PC-1500 a une gamme s'étendant sur tous les nombres entre $9,999999999 \times 10^{-99}$ et $-9,999999999 \times 10^{-99}$. Si un nombre trop grand déborde des limites de cette gamme, l'ordinateur, étant incapable de le traiter, signalera ce dépassement de capacité à l'aide du message d'erreur ERROR 37. Si un nombre devient trop petit, on dit qu'il se produit un dépassement de capacité inférieur; il ne sera pas signalé par un message d'erreur ou un arrêt dans le fonctionnement. Tout nombre se situant entre -1×10^{-99} sera considéré comme égal à zéro, comme illustré dans le tableau suivant:

$-1 \times 10^{-99} \quad 1 \times 10^{-99}$				
$-9,999999999 \times 10^{-99}$		0	$9,999999999 \times 10^{-99}$	
ERREUR	GAMME DE CALCUL	CONSIDERE EGAL A ZERO	GAMME DE CALCUL	ERREUR

Exemple 1: Si vous essayez de résoudre l'équation $(5.67 \times 10^{55}) \times (8.90 \times 10^{65})$, vous provoquerez un dépassement de capacité:

Touches utilisées	Affichage
(5 • 6 7 E 5 5) *	
(8 • 9 0 E 6 5)	(5. 67E55) * (8. 90E65)_

ENTER

ERROR 37

Le message d'erreur ERROR 37 indique le dépassement de capacité de calcul.

C. Les racines, les puissances et pi

Racine carrée

Exemple 1: Pour trouver la racine carrée de 73:

Touches utilisées	Affichage
SHIFT √ 7 3	✓ 73 _
ENTER	8. 544003745

Exemple 2: Pour trouver $\sqrt[4]{256}$:

Touches utilisées	Affichage
SHIFT √ SHIFT √ 2 5 6	✓✓ 256 _
ENTER	4

Exemple 3: Pour trouver $\sqrt{3^2 + 4^2}$:

Touches utilisées	Affichage
SHIFT √ (3 * 3 + 4 * 4)	✓(3*3+4*4)_
ENTER	5

On peut aussi calculer cette équation de la manière suivante:

Exemple 4:

Touches utilisées	Affichage
SHIFT √ (3 SHIFT ^ 2 + 4 SHIFT ^ 2) ENTER	$\sqrt{(3^2+4^2)}$ 5

Puissance

La puissance ou fonction exponentielle, permet d'élever un nombre à une puissance.

Exemple 1: Calculer 4^3 ($= 4 \times 4 \times 4$):

Touches utilisées	Affichage
4 SHIFT ^ 3 ENTER	4^3 64

Exemple 2: Calculer $3^3 \cdot 2 \times 4^{-2} \cdot 4$:

Touches utilisées	Affichage
3 SHIFT ^ 3 * 2 * 4 SHIFT ^ -2 * 4 ENTER	$3^3 \cdot 2 \cdot 4^{-2} \cdot 4$ 1. 207380162

Exemple 3: Calculer $4^{(3^2)}$:

Touches utilisées	Affichage
4 SHIFT ^ 3 SHIFT ^ 2 ENTER	4^{3^2} 262144

PI π

La valeur de PI (3,141592654) est mémorisée à la fois sous le symbole PI et π comme constante fixe. On peut utiliser l'un ou l'autre de ces symboles dans les calculs où l'on a besoin de la valeur de Pi.

Par exemple, pour calculer la surface d'un tapis d'un diamètre de cinq mètres, sur le mode RUN:

Touches utilisées

P | * (5 / 2)

SHIFT ^ 2

ENTER

Affichage

$\pi * (5/2)^2 -$

19. 63495408

D. Les modes angulaires

Le PC-1500 permet de calculer des fonctions angulaires sur l'un ou l'autre des trois modes angulaires suivants:

Pour placer le PC-1500 sur le mode degré, écrivez:

DEG. ENTER

(DEG apparaîtra dans la partie supérieure de l'affichage)

Pour placer le PC-1500 sur le mode radian, écrivez:

RAD. ENTER

(RAD apparaîtra dans la partie supérieure de l'affichage)

Pour placer le PC-1500 sur le mode grade, écrivez:

GRA. ENTER

(GRAD apparaîtra dans la partie supérieure de l'affichage)

E. Les fonctions trigonométriques

Le PC-1500 fournit six fonctions trigonométriques: SIN, COS, TAN, ASN, ACS, et ATN. Chacune peut être calculée sur le mode GRAD, DEG ou RAD. L'exécution s'effectue de manière suivante:

DEG. ENTER → Place sur le mode degré.

SIN 30 ENTER 0.5 → SIN 30 en degrés

CL → Remet l'affichage à zéro.

GRA. ENTER → Place sur le mode grade.

SIN 30 ENTER 4.539904997E-01 → SIN 30 en grade

CL → Remet l'affichage à zéro.

RAD. ENTER → Place sur le mode radian.

SIN 30 ENTER -9.880316241E-01 → SIN 30 en radians

Dans les exemples ci-dessus, le sinus de 30 est calculé sur chaque mode avec trois résultats différents mais équivalents. Il est possible aussi de calculer des fonctions trigonométriques inverses de la façon suivante:

RAD. [ENTER]	→	Place sur le mode
ASN -0.5 [ENTER]	-5. 235987756E-01	Arc sinus de -.5
DEG. [ENTER]	→	Place sur le mode
ASN -0.5 [ENTER]	-30	Arc sinus de -.5
ACS (-.5 + .1) [ENTER]	113. 5781785	Arc cosinus
ATN (7/3) [ENTER]	66. 80140949	Arc tangente

F. Les fonctions logarithmiques

LN, LOG

La fonction LN calcule le logarithme naturel (base e) tandis que la fonction LOG calcule le logarithme décimal (base 10). Leur exécution se déroule de la façon suivante:

LN 7.4 [ENTER]	2. 00148
LOG 7.4 [ENTER]	8. 692317197E-01
LN 25 [ENTER]	3. 218875825
LOG 100 [ENTER]	2

EXP

La fonction inverse de LOG est constituée d'un nombre amené à la puissance 10. Par exemple:

LOG 100 [ENTER]	2
10 [SHIFT] \wedge 2 [ENTER]	100

Une fonction inverse est nécessaire parce que le logarithme naturel (LN) n'est pas basé sur une puissance 10, mais sur une puissance e.

Exemple:

LN 7.4 [ENTER]	2. 00148
EXP 2.00148 [ENTER]	7. 399999998

G. Conversion d'angle

Le PC-1500 permet d'effectuer les conversions d'angles de la forme DMS (degré, minute, seconde) à la forme DEG (degré decimal). Quand on effectue la conversion opposée-degrés décimaux en degrés, minutes et secondes- la réponse est constituée d'une partie nombre entier représentant les degrés et d'une partie fractionnaire dont les deux premiers chiffres représentent les minutes et les deux suivants, les secondes. Les chiffres allant du 5e à la fin représentent les degrés décimaux. Pour accomplir la conversion en degrés décimaux, d'un angle donné en degrés, minutes et secondes, il faut l'entrer dans l'ordinateur comme nombre entier et décimales, respectivement.

Exemple 1: Convertir 16.1932 degrés décimaux en DMS:

DMS 16.1932 **ENTER**

16. 113552

Exemple 2: Convertir 32.2513 DMS en degrés décimaux:

DEG 32.2513 **ENTER**

32. 42027778

H. Fonctions diverses

ABS (valeur absolue)

La fonction ABS établit la valeur absolue d'une valeur numérique ou d'une variable.

Exemple 1:

ABS (25 – 86) **ENTER**

61

Normalement, $25 - 86 = -61$. Pour arriver à 61, la fonction ABS prend la différence réelle des nombres.

INT (nombre entier)

La fonction INT arrondit une valeur numérique au nombre entier le plus grand qui ne dépasse pas la valeur numérique elle-même.

Exemple 1:

(25/3) + 7 **ENTER**

15. 33333333

INT (25/3) + 7 **ENTER**

15

Exemple 2:

(31.62 + 21.18) **ENTER**

52. 8

INT (31.62 + 21.18) **ENTER**

52

Dans l'exemple 2, la réponse n'arrondit pas 52,8 à 53, car 53 est plus grand que la valeur originale 52,8.

NOTE: N'oubliez pas l'ordre d'évaluation. Utilisez les parenthèses si vous désirez obtenir un nombre entier comme expression du résultat. Modifions, en guise d'illustration, l'exemple précédent en omettant toutes les parenthèses:

Exemple 3:

INT 31.62 + 21.18 [ENTER]

52. 18

L'ordinateur arrondit 31.62 au nombre entier 31 et l'ajoute à 21,18 pour obtenir 52,18 ce qui diffère légèrement du résultat du calcul précédent, n'est-ce pas?

SGN (fonction de signe)

La fonction SGN sert à établir le signe d'un nombre X, c'est à dire qu'elle indique au moyen des valeurs suivantes s'il est positif, égal à zéro ou négatif:

+1	si	$X > 0$
0	si	$X = 0$
-1	si	$X < 0$

Exemples:

5 - 10 [ENTER]

-5

SGN (5 - 10) [ENTER]

-1

12 - 4 [ENTER]

8

SGN (12 - 4) [ENTER]

1

15 - 15 [ENTER]

0

SGN (15 - 15) [ENTER]

0

V. PROGRAMMATION AVANCEE

A. Tableaux et variables indicées

Jusqu'à présent vous vous êtes servi dans nos programmes-échantillons uniquement d'un petit nombre de variables. Au fur et à mesure que vous commencerez à utiliser au maximum la capacité de traitement du PC-1500, vous découvrirez que des variables, emmagasinant un seul nombre présentent des désavantages sérieux. Si, par exemple, vous considérez un programme qui entre cinquante nombres et les arrange, vous conclurez très vite que, quoique le PC-1500 mette à votre disposition un nombre de variables plus que suffisant, il doit y avoir une façon de faire plus facile. Elle existe et elle s'appelle "variable de tableaux".

"Un tableau" est constitué tout simplement d'un groupe de zones consécutives de stockage ou "emplacements" ayant un nom commun. Chaque zone de stockage peut contenir un seul nombre ou une chaîne de caractères. Dans un tableau donné, toutes les zones de stockage doivent emmagasiner le même type de données.

Le nombre d'emplacements à l'intérieur d'un seul tableau est déterminé par les spécifications de l'utilisateur, avec 256 comme maximum. Si par exemple vous définissez un tableau numérique comme ayant 50 emplacements, cela signifie que vous pouvez stocker jusqu'à 50 nombres sous un seul nom. Si vous définissez un tableau de chaînes (qu'on appelle tableaux de caractères) vous pouvez aussi spécifier la taille des chaînes, jusqu'à un maximum de 80 caractères par chaînes. La façon de créer des chaînes de caractères de tailles différentes est décrite sous Section B. 1.

On se sert de l'instruction DIM (dimension en abrégé) pour définir un tableau. A la différence des variables à valeur unique dont nous nous sommes servi jusqu'à présent, il faut toujours "déclarer" (définir) les tableaux avant de les utiliser. La forme de l'instruction DIMension numérique est la suivante:

DIM nom de variable numérique (taille)

dans laquelle:

le nom de variable numérique est un nom de variable qui se conforme aux règles ordinaires des noms de variables numériques examinées précédemment.

la taille est le nombre d'emplacements de stockage qui doit être situé dans la gamme allant de 0 à 255. Remarquez que lorsque vous spécifiez un nombre pour la taille vous établissez un emplacement de plus que ce que vous avez spécifié.

Exemples d'instructions DIMension numériques permises:

```
DIM X (5)
DIM AA (24)
DIM Q5 (0)
```

La première instruction établit un tableau X avec 6 emplacements de stockage. La seconde établit un tableau AA avec 25 emplacements. La troisième établit un tableau avec un seul emplacement ce qui est plutôt bête-en ce qui concerne les nombres du moins-puisqu'il équivaut à déclarer une variable numérique à valeur unique.

Il est très important que vous sachiez que le SHARP voit la distinction et la séparation entre une variable de tableau X et une variable X. Le premier X représente une série d'emplacements de stockage numérique et le second une seule location différente.

Maintenant que vous savez comment créer des tableaux vous devez vous demander comment l'on indique l'emplacement de stockage. Puisque le groupe entier n'a qu'un seul nom, on indique un emplacement particulier (qu'on appelle "élément") en faisant suivre le nom du groupe par un numéro entre parenthèses. On appelle ce numéro un "indice". Pour emmagasiner par exemple, le nombre 8 dans le cinquième élément de notre tableau X (déclaré auparavant) nous écrirons:

X (4) = 8

Si l'usage du 4 vous paraît étonnant, rappelez-vous que la numérotation des éléments commence à zéro et continue jusqu'au numéro de taille déclaré dans l'instruction DIM.

Le grand avantage des tableaux consiste à fournir la possibilité d'utiliser une expression ou un nom de variable en tant qu'indice. Pour établir par exemple, une table des carrés des nombres de 0 à 9, nous pouvons écrire les instructions suivantes:

```
10 DIM SQ (9)
20 FOR I = 0 TO 9
30 SQ (I) = I * I
40 NEXT I
```

Dans cet exemple la variable I sert à choisir l'emplacement de stockage qui contiendra le résultat aussi bien qu'à calculer ce résultat.

Une forme légèrement différente de l'instruction DIM sert à déclarer un tableau de caractères.

DIM nom de variable à caractères (taille) * longueur

formule dans laquelles:

le nom de variable à caractères consiste en un nom de variables qui est conforme aux règles auxquelles sont soumises les variables à caractères ordinaires examinées précédemment.

la taille constitue le nombre d'emplacements de stockage qui doit se situer dans la gamme allant de 0 à 255. Notez que lorsque vous spécifiez un nombre pour la taille, vous établissez un emplacement de plus que ce que vous avez spécifié.

*la longueur est facultative. Quand on l'utilise, elle spécifie la longueur de chacune des chaînes qui forment le tableau. La longueur est un nombre dans une gamme allant de 1 à 80. Si l'on ne se sert pas de cette proposition, les chaînes auront par défaut une longueur de 16 caractères.

Exemples de déclarations de tableau de caractères permises:

```
DIM X$ (4)
DIM NM$ (10) * 10
DIM IN$ (1) * 80
DIM R$ (0) * 26
```

Le premier exemple crée un tableau de cinq chaînes capables chacune de mémoriser 16 caractères. La seconde instruction DIM déclare un tableau NM de onze chaînes à 10 caractères chacune. Définir explicitement la longueur de chaînes plus petites que la longueur par défaut permet d'économiser l'espace de mémorisation. Le troisième exemple déclare un tableau à deux éléments de chaînes de 80 caractères et le dernier, une seule chaîne de vingt-six caractères (Voir section B.1).

En plus des tableaux simples que nous venons juste de voir, des tableaux à deux dimensions sont possibles sur le PC-1500. Par analogie, on pourrait considérer un tableau à une dimension comme étant une liste de données arrangeées en une seule colonne, tandis qu'un tableau à deux dimensions consisterait en un tableau de données en rangées et en colonnes. Le tableau à deux dimensions se déclare à l'aide de l'instruction suivante:

DIM nom de variable numérique (rangées, colonnes)

ou

DIM nom de variable à caractères (rangées, colonnes) * longueur

dans laquelle:

les rangées spécifient le nombre de rangées dans le tableau, nombre qui doit provenir de la gamme entre 0 et 255. Remarquez que lorsque vous spécifiez le nombre de rangées, vous établissez une rangée de plus que celui que vous avez spécifié.

les colonnes spécifient le nombre de colonnes dans le tableau, nombre qui doit provenir de la gamme entre 0 et 255. Remarquez que lorsque vous spécifiez le nombre de colonnes, vous établissez une colonne de plus que celui que vous avez spécifié.

Le schéma suivant illustre les emplacements de stockage qui résultent de la déclaration DIM T (2, 3) et les indices (composés de deux nombres maintenant) qui correspondent à chaque emplacement de stockage.

	colonne 1	colonne 2	colonne 3	colonne 4
Rangée 0	T (0, 0)	T (0, 1)	T (0, 2)	T (0, 3)
Rangée 1	T (1, 0)	T (1, 1)	T (1, 2)	T (1, 3)
Rangée 2	T (2, 0)	T (2, 1)	T (2, 2)	T (2, 3)

NOTE: Des tableaux à deux dimensions peuvent occuper très rapidement l'espace de stockage. Par exemple, un tableau à 25 rangées et 35 colonnes occupe 875 emplacements!

Les tableaux sont des outils très puissants. Nous vous recommandons de lire d'autres publications pour obtenir une vue plus complète de leurs capacités et utilisations.

B. Plus de détails sur les chaînes des caractères

B.1. Etablir la DIMension des chaînes

La longueur des chaînes à caractères est limitée par défaut à seize caractères. Il est possible cependant de créer une chaîne d'une longueur de 80 caractères en établissant la dimension d'une chaîne à caractères. Il est possible aussi d'économiser l'aire de la mémorisation en diminuant la longueur de la chaîne.

L'instruction DIMension ci-dessous spécifie la longueur d'une chaîne.

DIM nom de variable (limite) * longueur

dans laquelle:

nom de variable constitue le nom du tableau de chaînes de caractères.

limite constitue l'indice maximum du tableau.

la longueur constitue la longueur de chaque chaîne du tableau.

Si l'on n'a besoin que d'une seule chaîne, il est possible de spécifier un tableau à un élément (ayant zéro pour indice) pour économiser l'aire de mémorisation. La déclaration suivante d'une chaîne à 26 caractères illustre ce cas:

DIM A\$ (0) * 26

B.2. Enchaînement

Il est possible de combiner plusieurs chaînes de caractères (ou caractères à l'intérieur de variables à caractères) pour former une seule chaîne. On appelle "enchaînement" cette action de combiner des chaînes de caractères. On exprime l'enchaînement sous la forme générale suivante:

$$\text{variable} = \frac{\text{chaîne de caractères}}{\text{variable à caractères}} + \frac{\text{chaîne de caractères}}{\text{variable à caractères}}$$

Exemple 1:

```
10 S$ = "SUPER"  
20 T$ = S$ + "MAN"  
30 PRINT T$
```

Sortie:

SUPERMAN

Dans la ligne 20, le contenu de la variable S\$ ("SUPER") a été ajouté à la chaîne "MAN". Notez que l'on n'insère pas d'espace vide pendant l'enchaînement. On peut enchaîner plusieurs chaînes pour former une seule expression, comme dans l'exemple qui suit:

Exemple 2:

```
5 WAIT 100  
10 A$ = "ANCE"  
20 B$ = "DE"  
30 C$ = "NAISS"  
40 D$ = "DATE"  
50 PRINT "ECRIVEZ VOTRE" + D$  
60 PRINT B$ + C$ + A$  
70 GOTO 50
```

Sortie:

ECRIVEZ VOTRE DATE
DE NAISSANCE

Quand on effectue des enchaînements, on se sert, pour forger les nouvelles chaînes, d'une aire temporaire de stockage de caractères. Cette aire a une capacité de stockage de 80 caractères. Si la nouvelle chaîne dépasse cette limite on obtiendra un message d'erreur ERROR 15. Ci-dessous vous trouverez une illustration de cette aire pendant un enchaînement.

Exemple 3:



4

ABCDE

5



NOTE: Le symbole représente un index de caractères interne qui contrôle la quantité de mémoire utilisée.

- 1) Au début de l'exécution, l'aire de stockage sera remise à zéro et l'index de caractères reviendra à sa position de départ.
- 2) On entre "ABC" qui occupera les trois premières positions de l'aire.
- 3) On ajoute, après "ABC", "DEFGHI" à l'aire de stockage de caractères.
- 4) La fonction LEFT\$ agit sur la chaîne "DEFGHI" pour extraire "DE" qui remplacera "DEFGHI" dans l'aire.
- 5) L'affectation est accomplie et l'aire de stockage est remise à zéro de nouveau.

B.3. Comparaison de chaînes

On peut comparer des chaînes de caractères pour déterminer quelle chaîne est "plus grande" ou moins grande que l'autre. Cette opération se base sur la séquence de collation donnée dans l'appendice C qui consiste dans l'ordre dans lequel l'ordinateur reconnaît les caractères.

Si les chaînes contiennent un nombre inégal de caractères, on "rembourre" la chaîne la plus courte avec des caractères NULS (ASCII 0). Les opérateurs valides pour la comparaison de chaîne sont:

- = Vrai si les deux chaînes sont d'une longueur égale et contiennent les mêmes caractères dans le même ordre.
- <> Vrai si les deux chaînes diffèrent en longueur, en caractères et par l'ordre de leurs caractères.
- > Vrai si les caractères de la première chaîne sont "plus grands" (arrivent plus tard dans l'ordre de leur liste) que les caractères de la seconde chaîne.
- < Vrai si les caractères de la première chaîne sont "plus petits" (arrivent plus tôt dans l'ordre de leur liste) que les caractères de la seconde chaîne.

Le format de la comparaison de chaînes est le suivant:

chaîne de caractères variable de caractères	OP	chaîne de caractères variable de caractères
--	----	--

dans lequel OP constitue l'un des opérateurs listés plus haut.

Exemples:

"MARY" > "MARI"	est vrai
"MARY" = "MARY "	est faux
"abc" <> "ABC"	est vrai
"DATA 1" < "DATA 2"	est vrai
"?" < "#"	est faux

Note: Il n'est pas possible d'utiliser les formes $A\$ \leq B\$$, $A\$ \geq B\$$ pour comparer des chaînes de caractères. Par contre, il est possible d'utiliser les formes $(A\$ < B\$) OR (A\$ = B\$)$ et $(A\$ > B\$) OR (A\$ = B\$)$ pour les comparaisons.

C. Fonctions

C.1. ASC

Il y a deux fonctions qui servent à convertir des caractères en code ASCII et à les dé-convertis de ce code. La fonction ASC convertit un caractère unique dans son code décimal ASCII. La fonction inverse CHR\$ convertit le code décimal ASCII en une chaîne d'un caractère unique.

ASC { "caractère"
 nom de variable à caract.

Dans cette fonction, l'argument consiste en une chaîne quelconque de caractères ou un nom de variable de chaîne de caractères. La valeur rendue par cette fonction constitue le code ASCII correspondant au premier caractère de la chaîne spécifiée.

Exemple 1:

```
10 LET X$ = "PATTI"
20 LET A = ASC X$
30 PRINT A
```

Sortie:

RUN	•
	80

Dans l'exemple ci-dessus, on affecte à X\$ la valeur de la chaîne de caractères "PATTI". La fonction ASC prend le premier caractère (P) et le convertit en son numéro de code ASCII (80).

Exemple 2:

```
10 PRINT ASC "K"
```

Sortie:

RUN	•
	75

ASC "K" rapporte 75 qui constitue le code ASCII de "K".

C.2. CHR\$

CHR\$ constitue la fonction complémentaire de ASC. La fonction CHR\$ prend un code décimal ASCII, de 0 à 127 et retourne la chaîne de caractères équivalente. (Note: Certains codes représentent des caractères spéciaux qu'on ne peut imprimer.)

CHR\$	code décimal ASCII variable numérique
-------	--

Exemple 1:

```
10 PRINT (CHR$ 67) + "OP"
```

Sortie:

COP	
-----	--

Exemple 2:

```
10 Z = 65
20 PRINT CHR$ Z
```

Sortie:

A	
---	--

Le premier exemple illustre l'utilisation d'un code décimal ASCII comme argument. Le résultat est enchaîné à la chaîne "OP" pour produire "COP". Le second exemple affecte une valeur numérique à la variable Z. Ensuite on se sert du nom de variable comme argument, avec le caractère "A" comme résultat.

Notre programme-échantillon suivant convertit les majuscules d'une chaîne texte en minuscules à l'aide des deux fonctions ASC et STR\$ à la fois:

Exemple 3:

```
5 WAIT 0
10 INPUT "ENTREZ MESSAGE", M$
20 FOR I = 1 TO LEN (M$)
30 T$ = MID$ (M$, I, 1)
40 L = ASC (T$)
45 IF (L < 65) OR (L > 90) THEN 60
50 T$ = CHR$ (L + 32)
60 PRINT T$;
70 NEXT I
80 WAIT : PRINT
```

C.3. La Fonction INKEY\$

Cette fonction recueille n'importe quel caractère du clavier et le stocke dans la variable spécifiée. Il n'est pas nécessaire d'appuyer sur **ENTER** car le caractère sera accepté automatiquement.

variable = INKEY\$

A moins qu'on ne se serve d'une instruction PRINT antérieure, le symbole de guidage ne sera pas affiché durant l'exécution de cette instruction. Le caractère entré n'est pas renvoyé sur l'affichage qui ne subit donc pas d'altération.

Exemple:

```
10 WAIT 0
20 A$ = INKEY$
30 IF A$ = " " THEN PRINT "AUCUNE TOUCHÉ": GOTO 20
40 PRINT A$
50 GOTO 20
```

Cette fonction accepte seulement un caractère. Si l'on entre plus d'un caractère au clavier, seul le premier passera; tous les autres seront ignorés.

C.4. La fonction LEN

Quand on manipule des caractères, il est souhaitable de connaître le nombre de caractères d'une chaîne. Pour obtenir ce renseignement on se sert de la fonction LEN qui rapportera ce nombre de caractères dans une expression spécifiée ou une variable de caractères.

LEN { "chaîne de caractères"
 nom de variable de caractères

Exemple 1:

```
10 A$ = "CATON"
20 C = LEN A$
30 PRINT C
```

Sortie:

	RUN	I	.
			5

Exemple 2:

```
10 C = LEN "CAT"
20 PRINT C
```

Sortie:

	RUN	I	.
			3

Un zéro sera rapporté si l'on sert de LEN sur une chaîne vide (c'est-à-dire qu'il n'y a rien entre les guillemets).

Exemple 3:

```
10 A = LEN " "
20 PRINT A
```

Sortie:

	RUN	I	.
		0	

C.5. La fonction LEFT\$

Il y a trois fonctions dont on se sert pour sélectionner ou extraire des sections spécifiées d'une chaîne de caractères. La fonction LEFT\$ extrait des caractères de la gauche, la fonction RIGHT\$ de la droite et MID\$ du milieu.

LEFT\$	{ ("chaîne de caractères", nombre) (nom de variable de caractères, nombre)
--------	---

L'argument "nombre" spécifie combien de caractères doivent être extraits en partant de la gauche.

Exemple 1:

```
10 A$ = LEFT$ ("DRAPEAU", 4)
20 PRINT A$
```

Sortie:

	RUN	I	.
	DRAP		

Exemple 2:

```
10 B$ = "PARACHUTE"
20 A$ = LEFT$ (B$, 4)
30 PRINT A$
```

Sortie:

	RUN	I	.
	PARA		

Dans les deux exemples, on extrait des caractères de la chaîne en partant de la gauche pour les stocker sous A\$. Si l'on écrit A\$ sur le clavier ou obtient "DRAP" et "PARA" respectivement.

C.6. La fonction MID\$

On sert de la fonction MID\$ pour extraire la portion du milieu d'une chaîne de caractères.

MID\$	{ ("chaîne de caractères", #1, #2) (variable de chaîne de caractères, #1, #2)
-------	---

Exemple 1:

```
10 A$ = "FAITES VOS JEUX"  
20 B$ = MID$ (A$, 8,3)  
30 PRINT B$
```

Sortie:

VOS	RUN	I	*
-----	-----	---	---

Exemple 2:

```
10 T$ = MID$ ("(415) 743 - 1602", 6, 3)  
20 PRINT T$
```

Sortie:

743	RUN	I	*
-----	-----	---	---

Le premier argument de cette fonction consiste en une chaîne de caractères ou une variable de chaîne de caractères. Le second argument est un nombre qui représente le premier caractère à extraire. Le troisième argument constitue le nombre total de caractères à extraire, y compris le premier.

Dans le premier exemple, "FAITES VOS JEUX" est stocké sous A\$. MID\$ extrait trois caractères en commençant par le huitième et les loge dans B\$. Si l'on imprime B\$, on trouvera qu'il contient "VOS".

Dans le deuxième exemple, une chaîne contenant un numéro de téléphone constitue le premier argument. On repère le sixième caractère ("7") et on le stocke avec les deux caractères suivants sous la variable T\$. Le résultat imprimé sera "743".

C.7. La fonction RIGHT\$

RIGHT\$ fonctionne comme LEFT\$ la différence étant que RIGHTS commence à exécuter à partir de l'extrémité inverse de la chaîne (la droite). Les arguments sont les mêmes que pour LEFT\$:

RIGHT\$ { ("chaîne de caractères, nombre")
 (variable de caractères, nombre)

L'argument "nombre" de cette fonction spécifie combien de caractères sont à extraire de la chaîne, en partant de la droite.

Exemple 1:

```
10 X$ = "READ ONLY MEMORY"  
20 Y$ = RIGHT$ (X$, 6)  
30 PRINT Y$
```

Dans ces programme, la fonction RIGHT\$ prend six caractères de l'extrémité droite de la chaîne et les stocke sous la variable Y\$. Y\$ contient maintenant "MEMORY".

C.8. La fonction RND

Il vous faudra parfois fournir des nombres aléatoires à votre programme. La fonction RND permet à l'ordinateur d'engendrer des nombres aléatoires dans une gamme de un à un nombre spécifié. (Note: la gamme commence toujours par un.)

Exemple 1:

```
10 A = RND 5
```

Dans cet exemple A pourrait avoir n'importe quelle valeur entre un et cinq, inclusivement. Si vous désirez des nombres aléatoires dans une gamme commençant par un autre nombre que 1 (par exemple, de 40 à 50) il vous faudra simuler cela en engendrant des nombres aléatoires entre 1 et 10 et en leur ajoutant une constante (39 dans notre exemple).

Exemple 2:

```
10 FOR I = 1 TO 5
20 B = 40 + RND 10
40 PRINT B;
50 NEXT I
```

Sortie:

RUN	•
42 44 47 48 42	

C.9. La fonction RANDOM

Les nombres aléatoires sont engendrés par l'intermédiaire d'une formule mathématique et nous sont accessibles au moyen de la fonction RND. Chaque fois que l'ordinateur est mis en marche, il engendre une série de nombres aléatoires. Cette liste de nombres aléatoires ne change pas, à moins que l'on utilise la fonction RANDOM. Ceci signifie qu'un programme utilisera la même série de nombres aléatoires chaque fois que l'ordinateur est mis en marche. Pour éviter cela, la fonction RANDOM recompose la "semence" utilisée par la formule pour engendrer ses nombres aléatoires.

Exemple 1:

```
10 FOR I = 1 TO 5
15 A = RND 3
20 PRINT A;
30 NEXT I
```

Sortie: 1 2 2 3 1

Exemple 2:

```
10 FOR I = 1 TO 5
15 A = RND 3
20 PRINT A;
30 NEXT I
```

Sortie: 1 2 2 3 1

Pour obtenir de vrais nombres aléatoires dans ce cas, la fonction RANDOM doit apparaître avant l'instruction RND. Cette fonction introduit une nouvelle semence dans le processus de génération de nombres aléatoires et cause ainsi la différence qui existe entre eux. Conséquemment, deux programmes que l'on passe dans des conditions identiques produisent des données de sortie différentes.

Exemple 1:

```

10 RANDOM
15 FOR I = 1 TO 5
20 A = RND 3
30 PRINT A;
40 NEXT I

```

Sortie:

3 1 2 2 3

Exemple 2:

```

10 RANDOM
15 FOR I = 1 TO 5
20 A = RND 3
30 PRINT A;
40 NEXT I

```

Sortie:

2 3 1 3 2

C.10. La fonction STR\$

STR\$ fonctionne de façon contraire à VAL. STR\$ reconvertit une variable numérique interne en sa chaîne de caractères ou expression originale.

Exemple:

```

10 INPUT "ENTREZ UN NOMBRE "; I
20 S$ = STR$ (I)
30 PAUSE "CE NOMBRE EST"
40 PRINT "LA CHAINE"; S$

```

Le programme ci-dessus accepte un nombre, forme la représentation en caractère de ce nombre et le stocke dans la variable de caractère S\$. Du fait que la représentation numérique interne ne peut pas être affichée, elle est reconvertie automatiquement en une chaîne de caractères par l'instruction PRINT.

C.11. La fonction STATUS

On utilise la fonction STATUS pour afficher combien de mémoire de programme reste disponible ou combien de mémoire le programme utilise. STATUS 0 affiche le nombre "d'étapes" de programmes qui sont encore disponibles. STATUS 1 affiche le nombre "d'étapes" déjà utilisées. Le nombre maximum "d'étapes" disponible est de 1850.

Exemple:

STATUS 0

Sortie:

RUN	I	*
1 7 7 0		

STATUS 1

Sortie:

RUN	I	*
8 1		

La directive STATUS 0 équivaut à la commande MEM du PC-1211. MEM reste valide comme commande sur le PC-1500.

C.12. La fonction TIME

Pour afficher ou régler le mois, le jour et l'heure, on se sert de la fonction TIME de la manière suivante:

Réglage: TIME = MMDDHH.MMSS **ENTER**

Affichage: TIME **ENTER**

formules dans lesquelles MM représente deux chiffres pour le mois, DD, deux chiffres pour le jour et HH, deux chiffres pour l'heure. La portion fractionnaire définit les minutes (MM) et les secondes (SS).

Quand on affiche TIME, la sortie est dans le même format que celui donné ci-dessus. On peut se servir du résultat de la fonction de la même manière qu'un nombre et l'on peut aussi s'en servir librement dans des expressions.

Le programme suivant simule une horloge. Ne manquez pas de régler l'heure avant de passer le programme.

Listage du programme:

```
10 WAIT 0
20 A$ = STR$ TIME
30 IF TIME > 99999 THEN 50
40 A$ = "0" + A$
50 M$ = LEFT$ (A$, 2)
60 D$ = MID$ (A$, 3, 2)
70 H$ = MID$ (A$, 5, 2)
80 DS$ = M$ + " / " + D$ + " / 82"
90 DS = VAL (M$ + D$ + "00")
100 PRINT DS$;
110 T = TIME - DS
120 IF T >= 1 THEN 140
130 T = T + 12
140 IF T > 23.5959 GOTO 20
150 CURSOR 18 : PRINT T
160 GOTO 110
```

C.13. La fonction VAL

VAL and STR\$ se complètent en tant que fonctions de conversion de chaînes de caractères en variables numériques et inversement de variables en chaînes. La fonction VAL convertit une chaîne contenant la représentation en caractères d'un nombre en un nombre qu'elle stocke ensuite dans une variable.

NOTE: Si l'on utilise autre chose qu'un chiffre de 0 à 9, un point décimal., un signe positif (+) ou négatif (-) ou la notation scientifique (E) au cours d'une expression, l'ordinateur l'ignorera.

Exemple 1:

10 ZS = VAL "-37"

Résultat:

ZS contient le nombre -37

Exemple 2:

10 A = VAL "237.6"

Résultat:

A contient le nombre 237,6

D. PRINT USING

L'instruction USING permet à un programmeur de contrôler strictement le format des données sur l'affichage. Ceci permet de standardiser les affichages et d'éviter les pertes de données.

La proposition USING, seule ou combinée avec une instruction PRINT ou PAUSE, définit le format de toutes les instructions PRINT ou PAUSE qui suivent jusqu'au moment la prochaine proposition USING apparaît au cours du programme.

Il se peut que plusieurs propositions USING apparaissent dans une seule instruction PRINT ou PAUSE. Dans ce cas, chacune d'entre elles définit le format à utiliser pour imprimer les variables listées jusqu'à utiliser pour imprimer les variables listées jusqu'à ce que la proposition USING suivante se présente.

On appelle "chaîne de mise en forme" la chaîne de caractères spéciaux qui sert à spécifier un format. Les caractères de la chaîne de mise en forme définissent les zones disponibles pour l'affichage de données et limitent le type de données pouvant être imprimées dans ces zones. Ce procédé est le même que celui employé par les autres langages tel que le COBOL et le PL/I.

Il est possible de stocker une chaîne de mise en forme dans une variable à chaînes. Dans ce cas le nom de la variable devra remplacer la chaîne de mise en forme dans la proposition USING. Ceci rend possible des formats multiples qui sont sélectionnés sous contrôle de programme.

Les caractères que l'on est autorisé à employer dans une chaîne de mise en forme sont examinés sommairement ci-dessous:

TABLEAU DES CARACTERES DE MISE EN FORME

<u>Caractère</u>	<u>Utilisation</u>
#	Spécifie un champ numérique. Les nombres sont cadrés à droite dans le champ. Si le champ n'est pas assez large pour contenir le nombre, un message d'erreur ERROR 36 se produit. Des zéros en tête de nombre sont convertis en espaces vides.
*	Spécifie "Asterisk Fill" (remplissage par des astérisques) des positions spécifiées d'un champ numérique, positions qui ne contiennent pas de données.
.	Provoque l'affichage d'un point décimal dans un champ numérique.
,	Utilisé au début d'un champ numérique pour spécifier l'insertion de virgules après chaque trois chiffres.
^	Utilisé dans un champ numérique pour provoquer l'affichage en notation scientifique d'un nombre.
+	Utilisé dans un champ numérique pour forcer l'impression du <u>signe</u> des données.
&	Spécifie un champ de caractères. Les caractères sont cadrés à gauche dans ce champ. Si le champ n'est pas assez large pour contenir la chaîne de données, la chaîne est coupée.

NOTE: Il faut toujours que la largeur du champ numérique soit plus grande d'un espace que la largeur des données pour faire de la place pour le signe des données. Ceci est nécessaire que vous employiez le caractère de mise en forme + ou non.

NOTE: L'utilisation de la virgule rend nécessaire l'insertion d'un # de plus pour chaque virgule dans la chaîne de mise en forme.

Exemples

X = PI Y = 1234 A\$ = "ABCDEF"

PRINT USING "# ##"; X

3

PRINT USING "+##.##"; X

+3.141

PRINT USING "###.##^"; X

3. 14E 00

PRINT USING "##^.^"; X

3.E 00

PRINT USING "*#####"; Y

**1234

PRINT USING "***##"; Y

1234

PRINT USING "&&&&&####"; A\$; Y

ABCDEF 1234

PRINT USING "&&&"; A\$

ABC

10 U\$ = "*##.##.##"

20 USING U\$

30 PRINT Y; "\$"

**1234. 00\$

PRINT X; "\$"

***** 3. 14\$

PRINT USING; A\$; X

ABCDEF 3. 141592654

PRINT USING "# ##, # ##, # ##"; 246813

246, 813

Note: Utilisez le nombre de repères # (*compris) pour la désignation de variable (entier) dans la gamme suivante:

Avec une ponctuation à 3 chiffres (,) non utilisée: Maximum 11 (y compris le signe)

Avec une ponctuation à 3 chiffres utilisée: Maximum 14 (y compris le signe)

Cet ordinateur est doté de 10 chiffres significatifs pour les nombres.

Lorsqu'un format dépassant 10 entiers est désigné par l'instruction USING et qu'un nombre dépassant 10 entiers est affiché (imprimé) par l'instruction PRINT (LPRINT), le nombre affiche (imprimé) risque d'être incorrect.

Exemple: Mode RUN

Affichage

USING "# ## ## ## ## ## # # ## #"	ENTER	→ >
PRINT 888888888888	ENTER	→ 888888888800
(LPRINT 888888888888	ENTER	→ 888888888880)

12 chiffres

E. Transfert de contrôle calculé

En plus des instructions de contrôle décrites dans le Chapitre III, le PC-1500 est doté de deux autres instructions de contrôle de grande utilité. Les instructions ON GOSUB et ON GOTO. Comme leurs noms vous l'auront fait deviner, ces instructions agissent de la même façon que les instructions GOSUB et GOTO examinées auparavant. Leur différence réside dans le fait que ON GOSUB et ON GOTO peuvent transférer le contrôle (c'est-à-dire exécuter des instructions à un emplacement différent) automatiquement. GOTO et GOSUB vont vers une instruction (ou sous-routine) parmi plusieurs autres selon la valeur d'une certaine variable numérique. C'est cette dépendance d'une variable pour le guidage qui a valu aux instructions ON le surnom d'"instructions de contrôle calculé".

Les instructions ON prennent la forme:

ON expression { GOTO ligne #1, ligne #2, ligne #3, ... (etc)
 GOSUB " " " "

L'expression qui suit le mot-clé ON doit se résoudre en un nombre entier positif, plus grand que zéro et moins grand que le nombre de numéros de ligne listés après les mots-clés GOTO et GOSUB. Quand l'ordinateur rencontre l'instruction ON durant l'exécution, il transfère le cours de l'exécution au numéro de ligne qui correspond à la valeur de l'expression.

Une instruction ON typique pourrait être comme suit:

ON TX GOSUB 100, 200, 250, 300

Dans ce cas la variable TX DOIT contenir un nombre entre 1 et 4 parce qu'il n'y a que quatre numéros de ligne listés. Un nombre autre que ces quatre produirait une erreur puisqu'il n'y a pas de numéro de ligne correspondant. Pour cette raison, il s'avère nécessaire d'inclure plusieurs épreuves (instructions IF) pour être sûr que nos expressions résultent en un nombre valide.

Les instructions ON s'avèrent très utiles pour automatiser une série de choix. Considérons, par exemple, le morceau de programme suivant qui permet à l'utilisateur de choisir une table de taux de taxation parmi plusieurs. On pourrait écrire cela comme ci-dessus si l'on n'utilisait pas l'instruction ON:

```
10 PAUSE "TAUX DES IMPOTS:"  
20 PAUSE "(1) CELIBATAIRE,"  
30 PAUSE "(2) MARIE,"  
40 PAUSE "(3) PROFESSION?"; TT  
50 IF (TT<1) OR (TT>3) GOTO 10  
60 REM CHOISISSEZ TABLEAU APPROPRIE  
70 IF TT = 1 THEN GOTO 220  
80 IF TT = 2 THEN GOTO 300  
90 IF TT = 3 THEN GOTO 450  
(etc)
```

Grâce à l'instruction ON, nous pouvons combiner les lignes 70, 80 et 90 en une seule instruction:

```
70 ON TT GOTO 220, 300, 450
```

ON ERROR GOTO

On permet à un programme de détecter quand une erreur se produit grâce à l'utilisation d'une variante de transfert contrôle. Après la détection, le programme peut exécuter des instructions qui essaient de se remettre de l'erreur. De telles instructions peuvent renseigner ou donner des instructions à l'utilisateur ou garder des données importantes.

L'instruction ON ERROR GOTO indique au SHARP où il doit se porter lorsqu'il découvre qu'une erreur s'est produite. Cette instruction prend la forme suivante:

ON ERROR GOTO numéro de ligne

dans laquelle le numéro de ligne constitue le numéro d'une ligne de programme qui contient les directives à suivre en cas d'erreur.

F. Programmation de l'affichage

L'affichage incorporé dans le PC-1500 se prête d'une manière remarquablement souple à la sortie de données. Pour permettre aux programmeurs d'utiliser à fond la puissance de cet affichage, de nouvelles instructions ont été ajoutées au langage BASIC utilisé par le PC-1500.

Pour afficher jusqu'à 26 caractères à la fois, on s'est servi de la technologie du cristal liquide. Chaque caractère du jeu dont l'ordinateur est doté occupe une matrice à points de 5 x 7. Au moyen de la commande GPRINT, les programmeurs peuvent développer et afficher leurs propres caractères.

Il est possible d'utiliser le champ entier de l'affichage comme une matrice à points de 7 x 156 pour réaliser des graphiques. Pour créer des graphiques, des figures ou des symboles spéciaux, on peut stimuler des points individuels dans l'une quelconque des 156 colonnes. La commande POINT permet de "toucher" une colonne quelconque pour découvrir quels points sont actuellement stimulés.

Un haut-parleur et un générateur de son permettent au programmeur d'ajouter une dimension sonore à l'interaction entre l'homme et la machine. Il y a 256 fréquences (de 230 Hz à environ 7 KHz) sur lesquelles on peut produire des sons. La répétition automatique d'un son et le contrôle de la durée du son sont également possible sous contrôle de programme.

F.1. BEEP

L'instruction BEEP permet au programmeur d'utiliser le générateur de sons pour créer des effets sonores durant les jeux sur ordinateur, pour avertir d'une erreur par son et pour nombre d'autres applications dans le domaine de la communication entre l'ordinateur et l'utilisateur. Le format de l'instruction BEEP pour créer des sons est le suivant:

BEEP expression1 ,expression2 ,expression3

dans lequel:

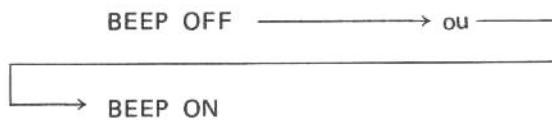
expression1 constitue le seul paramètre requis et spécifie combien de fois le son sera répété.
La gamme permise des répétitions va de 0 à 65535.

expression2 est facultative et spécifie la fréquence du ou des sons, fréquence représentée par un nombre entre 0 et 255.

expression3 est aussi facultative et spécifie la durée de chaque son, durée spécifiée par un nombre d'une gamme allant de 0 à 65279.

On peut aussi se servir de l'instruction BEEP pour mettre sous et hors tension le haut-parleur interne du PC-1500. On élimine ainsi le bruit produit par les opérations SAVE (conservation) et LOAD (chargement) de cassette.

Le format de l'instruction BEEP qui contrôle le haut-parleur interne est le suivant:



NOTE: Si l'on met le PC-1500 à l'arrêt (OFF) puis de nouveau en marche (ON), on remet le haut-parleur sous tension.

Programme de démonstration

```
10 D = 60
20 DATA 14
30 DATA 245, 1, 245, 1, 160, 1, 160, 1
40 DATA 143, 1, 143, 1, 160, 2
50 DATA 180, 1, 180, 1, 195, 1, 195, 1
60 DATA 220, 1, 220, 1, 245, 2
100 READ X
110 FOR I = 1 TO X
120 READ N, S
130 BEEP 1, N, (D * S)
140 NEXT I
150 END
```

F.2. L'instruction CURSOR

L'instruction CURSOR place le curseur sur l'un des 26 emplacements de caractères disponibles sur l'affichage. Cette instruction prend la forme suivante:

CURSOR expression d'emplacement

dans laquelle:

expression d'emplacement se réduit à un nombre de la gamme allant de 0 à 25, nombre qui spécifie l'emplacement où le curseur se rendra.

Ordinairement, on utilise la commande CURSOR pour placer le curseur avant l'impression de données. Utilisée de cette manière, elle permet au programmeur de définir sa propre séparation entre données élémentaires comme dans les instructions suivantes, par exemple:

Listage du programme:

```
10: WAIT 20
20: X = 8
30: PRINT "A"
40: CURSOR 4
50: PRINT "B"
60: CURSOR X
70: PRINT "C"
80: CURSOR ((X ^ 2) / 4) - 4
90: PRINT "D"
100: END
```

Touches utilisées:

```
1 Ø W A I T 2 Ø ENTER  
2 Ø X = 8 ENTER  
3 Ø P R I N T SHIFT " A SHIFT " ENTER  
4 Ø C U R S O R 4 ENTER  
5 Ø P R I N T SHIFT " B SHIFT " ENTER  
6 Ø C U R S O R X ENTER  
7 Ø P R I N T SHIFT " C SHIFT " ENTER  
8 Ø C U R S O R ( ( ( X SHIFT ^ 2 )  
    / 4 ) - 4 ENTER  
9 Ø P R I N T SHIFT " D SHIFT " ENTER  
1 Ø Ø E N D ENTER
```

Ce programme provoquera l'apparition des lettres A, B, C et D respectivement dans les emplacements 0, 4, 8 et 12 de l'affichage:



NOTE: Si l'on spécifie un numéro d'emplacement plus grand que 25 ou moins grand que 0, on obtiendra un message d'erreur ERROR 19.

Programme de démonstration

```
10 WAIT 0  
20 DIM A$ (0) * 13  
30 INPUT "ENTREZ VOTRE NOM", A$ (0)  
40 C = 0 : CLS  
50 FOR I = 1 TO (LEN A$ (0) - 1)  
60 CURSOR C  
70 PRINT MID$ (A$ (0), I, 1)  
80 C = C + 2  
90 NEXT I  
100 WAIT  
110 CURSOR C  
120 PRINT RIGHT$ (A$ (0), 1)  
130 END
```

Touches utilisées:

```
1 Ø W A I T Ø ENTER  
2 Ø D I M A SHIFT $ ( Ø ) * 1 3 ENTER  
3 Ø I N P U T SHIFT " E N T R E Z SPACE  
    V O T R E SPACE N O M SHIFT "  
    SHIFT , A SHIFT $ ( Ø ) ENTER  
4 Ø C = Ø SHIFT : C L S ENTER  
5 Ø F O R I = 1 T O ( L E N A SHIFT $ ( Ø )  
    - 1 ) ENTER  
6 Ø C U R S O R C ENTER  
7 Ø P R I N T M I D SHIFT $ ( A SHIFT $ ( Ø )  
    SHIFT , I SHIFT , 1 ) ENTER  
8 Ø C = C + 2 ENTER  
9 Ø N E X T I ENTER  
1 Ø Ø W A I T ENTER  
1 1 Ø C U R S O R C ENTER  
1 2 Ø P R I N T R I G H T SHIFT $  
    ( A SHIFT $ ( Ø ) SHIFT , 1 )  
    ENTER  
1 3 Ø E N D ENTER
```

F.3. L'instruction CLS (remise à zéro de l'affichage)

L'instruction CLS efface l'affichage en mettant hors tension tous les points qui le forment. On s'en sert avant d'écrire d'autres instructions pour effacer toutes les données restantes de l'affichage. La commande CLS ramène le curseur à la gauche de l'affichage. Le format est simplement CLS.

Programme de démonstration

```
10 GPRINT "7 F 7 F 7 F 7 F 7 F"  
20 CLS  
30 PRINT "NOUVELLE DONNEE"
```

Touches utilisées:

```
1 Ø G P R I N T SHIFT " 7 F 7 F 7 F  
    7 F 7 F SHIFT " ENTER  
2 Ø C L S ENTER  
3 Ø P R I N T SHIFT " N O U V E L L E SPACE  
    D O N N E E SHIFT " ENTER
```

F.4. L'instruction GCURSOR

L'instruction GCURSOR spécifie l'une des 156 colonnes de points disponibles de l'affichage comme colonne de départ pour tout affichage ultérieur de données. Le format de l'instruction GCURSOR est le suivant:

GCURSOR expression d'emplacement

dans lequel:

expression d'emplacement se réduit à un nombre de la gamme allant de 0 à 155. Ce nombre spécifie l'une des 156 colonnes à 7 points de l'affichage.

NOTE: Si une expression d'emplacement résulte en un nombre moins grand que 0 et plus grand que 155, un message d'erreur ERROR 19 apparaîtra.

On utilise généralement l'instruction GCURSOR conjointement avec l'instruction GPRINT pour réaliser un affichage de graphiques. (Nous illustrerons cette fonction avec plus de minutie dans la section suivante). On peut faire suivre l'instruction GCURSOR par d'autres types de directives puisque cette instruction n'écrit pas de données. GCURSOR indique seulement dans quelle colonne des données ultérieures seront écrites. Les lignes qui suivent illustrent cette fonction:

Listage de programme:

```
10 GCURSOR 50
20 PRINT "A"
30 GCURSOR 80
40 PRINT 26/3
```

Touches utilisées:

```
1 Ø G C U R S O R 5 Ø ENTER
2 Ø P R I N T SHIFT " A SHIFT " ENTER
3 Ø G C U R S O R 8 Ø ENTER
4 Ø P R I N T 2 6 / 3 ENTER
```

ce qui produit une sortie ayant un air normal aux emplacements 50 et 80:

		RUN	
A		8. 66666667	*

Que se passerait-il si nous changions la ligne 30 de façon à commencer à imprimer à l'emplacement GCURSOR 93? Faisons l'essai en lui substituant la ligne suivante:

```
30 GCURSOR 93
```

O surprise! La deuxième sortie a été "tronquée", coupée là où elle débordait de l'extrême de l'affichage.

Comme exemple final d'instruction GCURSOR nous vous présentons un programme plus évolué qui crée un effet insolite au moyen d'un chevauchement de caractères:

Listage du programme:

```
10 WAIT 0
20 DIM A$(0) * 10
30 A$(0) = " " : C = 0
40 INPUT "ENTREZ MESSAGE (< 10 CARA)", A$(0)
50 CLS
60 FOR I = 1 TO (LEN A$(0))
70 GCURSOR C
80 FOR J = 1 TO 3
90 GCURSOR C
100 PRINT MID$(A$(0), I, 1)
105 C = C + 3
110 NEXT J
120 C = C + 5
130 NEXT I
140 WAIT
150 GCURSOR 155
160 GPRINT "00"
170 END
```

Touches utilisées:

1 Ø W A I T Ø ENTER
2 Ø D I M A SHIFT \$ (Ø) * 1 Ø ENTER
3 Ø A SHIFT \$ (Ø) = SHIFT " SHIFT "
SHIFT : C = Ø ENTER
4 Ø I N P U T SHIFT " E N T R E Z SPACE
M E S S A G E SPACE (SHIFT < 1 Ø
SPACE C A R A) SHIFT " SHIFT ?
A SHIFT \$ (Ø) ENTER
5 Ø C L S ENTER
6 Ø F O R I = 1 T O (L E N A SHIFT \$
(Ø)) ENTER
7 Ø G C U R S O R C ENTER
8 Ø F O R J = 1 T O 3 ENTER
9 Ø G C U R S O R C ENTER
1 Ø Ø P R I N T M I D SHIFT \$ (A
SHIFT \$ (Ø)) SHIFT ? I SHIFT ?
1 Ø) ENTER
1 Ø 5 C = C + 3 ENTER
1 1 Ø N E X T J ENTER
1 2 Ø C = C + 5 ENTER

```

1 3 0 N E X T I ENTER
1 4 0 W A I T ENTER
1 5 0 G C U R S O R 1 5 5 ENTER
1 6 0 G P R I N T SHIFT " 0 0 SHIFT " ENTER
1 7 0 E N D ENTER

```

F.5. GPRINT

L'instruction GPRINT fournit un moyen de contrôle direct et programmable sur les points de l'affichage. L'instruction GPRINT est, comme nous l'avons vu plus haut, utilisée conjointement avec l'instruction GCURSOR puisqu'elle règle les points de toutes les colonnes à 7 points. L'instruction GCURSOR choisit la colonne appropriée à la modification tandis que l'instruction GPRINT manipule les points de la colonne en question. L'instruction GPRINT est aussi capable d'imprimer plusieurs colonnes attenantes de données en une seule instruction.

Pour comprendre le format de l'instruction GPRINT il est nécessaire de comprendre d'abord comment l'on contrôle les points d'une colonne. On peut spécifier le motif de points alimentés (par un courant) dans une colonne soit par un nombre décimal soit par une chaîne de caractères hexadécimale. Si l'on utilise le système décimal, on peut imaginer que chaque rangée est numérotée du haut en bas, par une puissance deux. Ceci est illustré ci-dessous:

1	-----
2	- -----
4	-- -----
8	-- -----
16	-- -----
32	-- -----
64	-- -----

Considérant qu'il y a 7 points par colonne et que l'on peut activer ou déactiver chacun d'entre eux, nous obtenons 128 motifs possibles. Pour spécifier un motif particulier, on emploie le format:

GPRINT expression de motif; expression de motif 2 . . (etc) . .

dans lequel:

l'expression de motif se réduit à un nombre de la gamme allant de 0 à 127 et spécifie le motif de points sous tension. On peut, facultativement, spécifier plusieurs expressions de motif que l'on doit séparer par un point-virgule ou une virgule. Si l'on utilise une virgule, une colonne vide existera entre chaque colonne imprimée.

Nous allons illustrer l'utilité de l'instruction GPRINT en créant un nouveau caractère: une flèche dirigée vers le haut. Dessinons d'abord ce caractère à l'intérieur d'une grille représentant les rangées et les colonnes:

1	----- * -----
2	- - * - * - - -
4	* - - - * - - - *
8	- - - - * -----
16	- - - - * -----
32	- - - - * -----
64	- - - - * -----
	1 2 3 4 5

Attention

Observez ce qui suit lors d'une correction de l'instruction PRINT dans un programme pour une instruction GPRINT:

Exemple:

Ré-écrivez complètement "PRINT" sous la forme "GPRINT".

20 PRINT A\$

Le fait d'insérer seulement "G" ne permet pas à l'ordinateur de le juger en tant que "GPRINT". (C'est considéré comme étant "G" et "PRINT".)

La même chose s'applique lorsque l'instruction CURSOR est corrigée par l'instruction GCURSOR, ou par les instructions de l'imprimante.

Il nous faudra 5 numéros séparés dans la liste expression de motif de l'instruction GPRINT parce que notre caractère s'étend sur 5 colonnes. Les numéros représentant les colonnes 1 et 5 doivent chacun spécifier un seul point dans la rangée ayant 4 comme étiquette. De même, les numéros représentant les colonnes 2 et 4 doivent chacun spécifier un seul point dans la rangée 2. Voici l'instruction finale:

GPRINT 4;2;127;2;4

La spécification de la troisième colonne (127) constitue la seule valeur numérique dont la dérivation n'est pas évidente au premier coup d'œil. 127 constitue la somme d'un point dans la première rangée (1) d'un point dans la deuxième rangée (2) d'un point dans la troisième rangée (4) et ainsi de suite. 127 consiste donc en $1 + 2 + 4 + 8 + 16 + 32 + 64$ et spécifie ainsi tous les 7 points de la colonne. On peut créer toutes sortes de motifs en spécifiant une rangée ou une somme de plusieurs rangées.

Si l'on utilise le système hexadécimal d'adressage, on divise les 7 rangées de l'affichage en un groupe inférieur de 3 rangées et un groupe supérieur de 4 rangées. Chaque groupe est numéroté de haut en bas par des puissances deux, comme indiqué ci-dessous:

1	— — — — — — —
2	— — — — — — —
4	— — — — — — —
8	— — — — — — —
1	— — — — — — —
2	— — — — — — —
4	— — — — — — —

Il est donc possible de représenter tous les motifs d'un groupe par un seul chiffre hexadécimal. La gamme de chiffres permise pour le groupe inférieur ira de 0 à 7 puisque ce groupe n'a que trois rangées. Des deux chiffres hexadécimaux requis pour spécifier toute une colonne, le premier représente le groupe inférieur et le second le groupe supérieur.

Voici la forme de l'instruction GPRINT hexadécimale:

GPRINT "chaîne hexadécimale"

dans laquelle:

la chaîne hexadécimale consiste en une chaîne de chiffres hexadécimaux, dont chaque paire spécifie le motif des points d'une colonne.

Utilisons ce format pour créer le caractère à flèche dirigée vers le haut de l'exemple précédent au moyen de l'instruction:

GPRINT "04027F0204"

Tableau de caractères hexadécimaux

1	-----	--- * ---	-----	--- * ---
2	-----	-----	--- * ---	--- * ---
4	-----	-----	-----	-----
8	-----	-----	-----	-----
	0	1	2	3
1	-----	--- * ---	-----	--- * ---
2	-----	-----	--- * ---	--- * ---
4	--- * ---	--- * ---	--- * ---	--- * ---
8	-----	-----	-----	-----
	4	5	6	7
1	-----	--- * ---	-----	--- * ---
2	-----	-----	--- * ---	--- * ---
4	-----	-----	-----	-----
8	--- * ---	--- * ---	--- * ---	--- * ---
	8	9	A	B
1	-----	--- * ---	-----	--- * ---
2	-----	-----	--- * ---	--- * ---
4	--- * ---	--- * ---	--- * ---	--- * ---
8	--- * ---	--- * ---	--- * ---	--- * ---
	C	D	E	F

Programme de démonstration

Ce programme imprime tous les motifs de points possibles, dans l'ordre, de 0 à 127.

Listage du programme:

```
10 WAIT 0 : CLS
20 FOR I = 0 TO 127
30 GCURSOR I
40 GPRINT I
50 NEXT I
60 WAIT
70 GCURSOR 155 : GPRINT "00"
80 END
```

Touches utilisées:

```
1 Ø W A I T Ø SHIFT : C L S ENTER  
2 Ø F O R I = Ø T O 1 2 7 ENTER  
3 Ø G C U R S O R I ENTER  
4 Ø G P R I N T I ENTER  
5 Ø N E X T I ENTER  
6 Ø W A I T ENTER  
7 Ø G C U R S O R 1 5 5 SHIFT :  
    G P R I N T SHIFT " Ø Ø SHIFT " ENTER  
8 Ø E N D ENTER
```

F.6. La fonction POINT

La fonction POINT rapporte un nombre qui représente le motif de points activés dans une colonne donnée. La fonction POINT permet donc de "tâter" une colonne quelconque de l'affichage sous contrôle de programme.

Ci-dessous, le format de la fonction POINT:

POINT expression d'emplacement

dans lequel:

l'expression d'emplacement se réduit à un nombre de la gamme allant de 0 à 155 et représente la colonne à sonder.

La valeur rapportée par la fonction POINT consiste en un nombre de la gamme allant de 0 à 127. L'interprétation de ce nombre est une somme de puissances 2 comme nous l'avons vu dans la section concernant GPRINT.

A titre d'illustration, supposez qu'il y ait un 1 majuscule sur l'affichage, s'étendant sur les colonnes 40 à 44:

1	— * — * — * —
2	— — — * — — —
4	— — — — * — — —
8	— — — — — * — — —
16	— — — — — — * — — —
32	— — — — — — — * — — —
64	— — * — * — * —
4	4 4 4 4 4
0	0 1 2 3 4

L'expression:

POINT 40 rapporterait 0,
POINT 41 rapporterait 65,
POINT 42 rapporterait 127.

Programme de démonstration

Le programme listé ci-dessous remplit l'affichage avec le caractère stocké dans A\$ et crée des motifs bizarre en inversant ce caractère. Ce programme utilise plusieurs instructions examinées dans le présent chapitre.

Listage du programme:

```
10 A$ = "X"
20 WAIT 0
30 Y = 5 : X = 155/Y : C = 0
40 FOR I = 1 TO X
50 GCURSOR C
60 PRINT A$
70 C = C + Y
80 NEXT I
90 FOR I = 0 TO 155
100 GCURSOR I
110 A = 127 - POINT I
120 GPRINT A
130 NEXT I
140 GOTO 90
```

Touches utilisées:

1 Ø A SHIFT \$ = SHIFT " X SHIFT " ENTER
2 Ø W A I T Ø ENTER
3 Ø Y = 5 SHIFT : X = 1 5 5 / Y
SHIFT : C = Ø ENTER
4 Ø F O R I = 1 T Ø X ENTER
5 Ø G C U R S O R C ENTER
6 Ø P R I N T A SHIFT \$ ENTER
7 Ø C = C + Y ENTER
8 Ø N E X T I ENTER
9 Ø F O R I = Ø T O 1 5 5 ENTER
1 Ø Ø G C U R S O R I ENTER
1 1 Ø A = 1 2 7 - P O I N T I ENTER
1 2 Ø G P R I N T A ENTER
1 3 Ø N E X T I ENTER
1 4 Ø G O T O 9 Ø ENTER

NOTE: Si l'on inclut la dernière ligne, le programme se trouvera dans une boucle infinie lors de son passage (la touche BREAK sert à y mettre fin). On peut changer aussi le caractère affiché en insérant un nouveau caractère dans l'affectation en ligne 10.

G. Détection d'erreurs

Aussi consciencieux que vous soyez, il vous arrivera tôt ou tard de créer un programme qui n'accomplit pas exactement ce que vous désireriez qu'il accomplisse. Les créateurs du SHARP ont prévu une méthode spéciale d'exécution de programme appelée "mode Trace" pour vous aider à identifier la source du problème. Quand on le place sur le mode Trace le PC-1500 affiche le numéro de ligne de chaque ligne de programme et s'arrête après l'exécution de cette ligne. Ceci permet de suivre la trace des instructions au fur et à mesure de leur exécution. Durant la pause de programme à la fin de l'exécution d'une ligne, vous pourrez examiner ou altérer les valeurs des variables.

La forme de l'instruction servant à débouter le mode Trace est simplement: TRON en tant que commande (sur le mode RUN); on peut aussi l'insérer en tant qu'instruction dans un programme. En tant que commande, TRON indique au SHARP qu'il faut "pister" pendant l'exécution de tous les programmes qui suivent. Après cela, on commence les programmes de la manière ordinaire, à l'aide de la commande GOTO ou RUN.

Utilisé comme instruction, TRON amorce le mode Trace seulement quand la ligne qui la contient est exécutée. Si pour une raison quelconque, on n'atteint pas cette ligne, le mode Trace restera inactif.

Une fois amorcé, le mode d'opération Trace reste efficace jusqu'à qu'il soit annulé par une directive TROFF. On peut lancer cette directive tout aussi bien en tant que commande qu'en tant qu'instruction. On peut aussi annuler le mode Trace avec la séquence-clé suivante:

SHIFT CL

En guise d'illustration de l'utilisation du mode Trace, entrez le programme suivant pour calculer la longueur de l'hypoténuse d'un triangle dont les côtés sont donnés:

Listage du programme:

```
10 INPUT A, B  
20 A = A * A : B = B * B  
30 H = √(A + B)  
40 PRINT "HYPOTENUSE = "; H
```

Sur le mode RUN, lancez la commande TRON suivie par la commande RUN. Remarquez que la commande INPUT fonctionne de manière ordinaire et affiche un point d'interrogation pour chaque valeur requise. Dès que vous avez entré deux valeurs, le numéro de ligne de l'instruction INPUT apparaît:

RUN	1	•
10 :		

Vous pouvez réexaminer la ligne tout entière en appuyant sur la touche **↑** (flèche vers le haut) et en la maintenant en position basse:

RUN	1	•
10 : INPUT A, B		

Pour continuer le programme, appuyez une fois sur la touche **↓** (flèche vers le bas). Cette opération provoque l'exécution de la ligne suivante et l'affichage de son numéro. Ensuite vous pouvez à nouveau revoir la ligne à l'aide de la touche **^** (flèche vers le haut). Vous pouvez aussi vérifier le contenu de n'importe quelle variable en imprimant son nom et en appuyant sur **ENTER**:

A **ENTER**

(dans ce cas A est une variable de programme)

RUN	•	4
-----	---	---

Il est nécessaire d'appuyer une fois pour chaque ligne sur la touche **↓** (flèche vers le bas) pour permettre l'exécution de chaque ligne, et cela jusqu'à la fin du programme. Si vous ne voulez pas continuer, l'exécution ligne par ligne ordinaire, appuyez sur ENTER pour suspendre l'exécution du programme. Si vous changez d'avis à nouveau, la commande CONT vous permettra de continuer les programmes suspendus.

Utilisons notre programme à hypoténuse pour les opérations suivantes:

Touches utilisées	Affichage
	>
T R O N	TRON_
ENTER	>
R U N	RUN_
ENTER	?
3	3_
ENTER	?
4	4_
ENTER	10:
↑	10: INPUT A, B_
↓	20:
↑	20: A=A*A : B=B*B_
A	A_
ENTER	9
B	B_
ENTER	16
↓	30:
H	H_
ENTER	5
↓	HYPOTENUSE= 5
↑	40: PRINT "HYPOTENUSE= "; H_
↓	40:
↓	>

H. Nombres hexadécimaux et fonctions de BOOLE

H.1. Nombres hexadécimaux

Le PC-1500 vous offre la possibilité d'utiliser un nombre hexadécimal (base 16) à l'intérieur d'une expression dans laquelle on peut utiliser un nombre décimal. Les nombres hexadécimaux se distinguent des nombres décimaux par l'esperluète (&) qui les précède. Les nombres de l'exemple ci-dessous constituent des formes valides de nombres hexadécimaux:

&16 &F &7ECA &08 &99A &-5B

On peut utiliser les nombres hexadécimaux dans les calculs:

10 + &A **ENTER**

RUN	I	.
		20

Ou encore dans les programmes:

Programmes:

35 GPRINT &F, 54, &3E
40 DATA 67, &7F, &2B, 12, 305

H.2. La fonction AND (et)

La fonction AND fournit un AND (et) booléen de la représentation interne de deux valeurs. Les valeurs doivent se situer dans la gamme allant de -32768 à 32767. Des nombres hors de cette gamme provoquent une erreur, ERROR 19.

<u>Exemple:</u>	<u>Résultat:</u>
10 AND &F	1 0
1 AND 0	0
-1 AND 1	1
55 AND 64	0
16 AND 63	1 6

H.3. La Fonction OR (ou)

La fonction OR effectue ou OR (ou) booléen sur les représentations internes de deux valeurs, les valeurs doivent se situer dans la gamme allant de -32768 à 32767. Des nombres hors de cette gamme provoquent une erreur, ERROR 19.

<u>Exemple:</u>	<u>Résultat:</u>
10 OR &F	15
1 OR 0	1
-1 OR 1	-1
55 OR 64	119
16 OR 63	63

H.4. La Fonction NOT

La fonction NOT rapporte le NOT (non) booléen, ou complément, de la représentation interne d'une valeur unique. Cette valeur doit se situer dans la gamme allant de -32768 à 32767. Si la valeur est hors gamme, un message d'erreur (ERROR 19) apparaîtra.

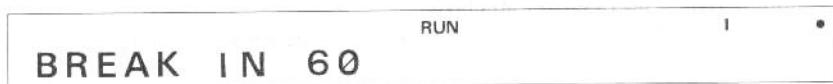
<u>Exemple:</u>	<u>Résultat:</u>
NOT Ø	-1
NOT &F	-16
NOT 55	-56
NOT 1	-2
NOT -2	1

I. Pour arrêter l'exécution d'un programme

STOP, CONT

L'instruction STOP provoque la suspension par l'ordinateur de l'exécution du programme. Quand le programme s'arrête, les valeurs des variables qui sont maintenues, sont à la disposition du programmeur qui peut les examiner et les modifier. Le programme pourra alors être remis en marche à l'endroit où il avait été suspendu, à l'aide de la commande CONT.

Quand l'ordinateur rencontre l'instruction STOP il affiche un message du type ci-dessous:



dans lequel 60 constitue le numéro de la ligne qui contient l'instruction STOP.

Appuyez sur la touche (flèche vers le haut) et maintenez-la en position basse si vous souhaitez passer en revue cette ligne.

Vous pouvez aussi passer en revue et modifier les variables lorsque le message BREAK apparaît. Par exemple:

HQ



A\$

"DEDUCT IONS"

Quand vous êtes prêts à continuer l'exécution, ramenez simplement le symbole de guidage (>) et écrivez CONT ENTER.

J. Contrôle du mode

LOCK, UNLOCK (Bloquer, débloquer)

Vous pouvez vous servir de la directive LOCK pour contrôler le mode (RUN, PROgramme ou RESERVE) sur lequel l'ordinateur fonctionne. Introduite dans un programme, elle empêche l'utilisateur de changer accidentellement le mode et d'endommager le programme. La directive LOCK rend inefficace la touche MODE et, de ce fait, "bloque" l'ordinateur sur son mode du moment.

Pour rendre son effet à la touche MODE, on se sert de la directive UNLOCK qui ramène le fonctionnement ordinaire permettant les changements de mode.

On peut se servir de LOCK et UNLOCK en tant que commande ou en tant qu'instruction. Leur forme est simplement:

LOCK

UNLOCK

VI. ELARGISSEMENT DU PC-1500

A. L'INTERFACE IMPRIMANTE/CASSETTE

L'interface imprimante/cassette constitue une option pour l'ordinateur de poche SHARP PC-1500. Cette unité permet d'effectuer le raccordement avec un ou deux magnétophones à cassette qui serviront à enregistrer des programmes et des données sur des cassettes audio standard. Ces programmes et données peuvent être "rechargés" par la suite dans le PC-1500, évitant ainsi à l'utilisateur l'ennui d'avoir à les ré-entrer au clavier.

L'utilisation de ces caractéristiques constitue le sujet des sections suivantes.

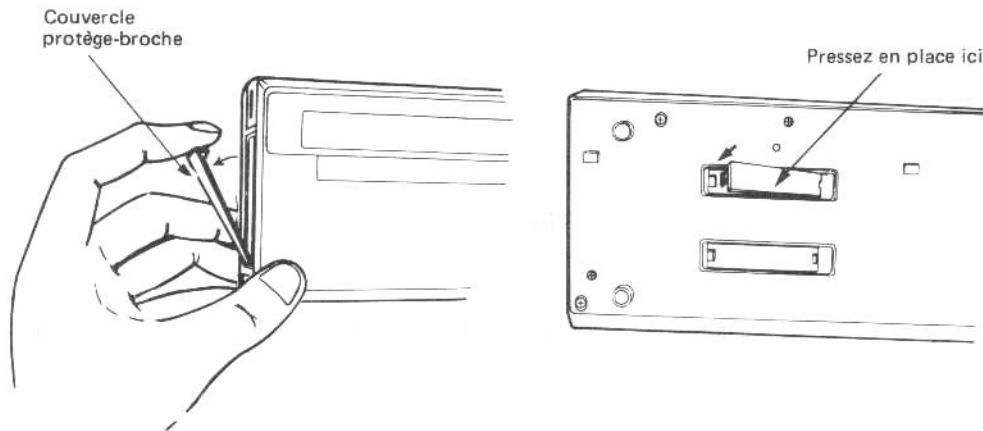
1. Connexion de l'ordinateur sur l'interface

Effectuez la connexion entre l'interface (CE-150) et l'ordinateur (PC-1500) de façon suivante:

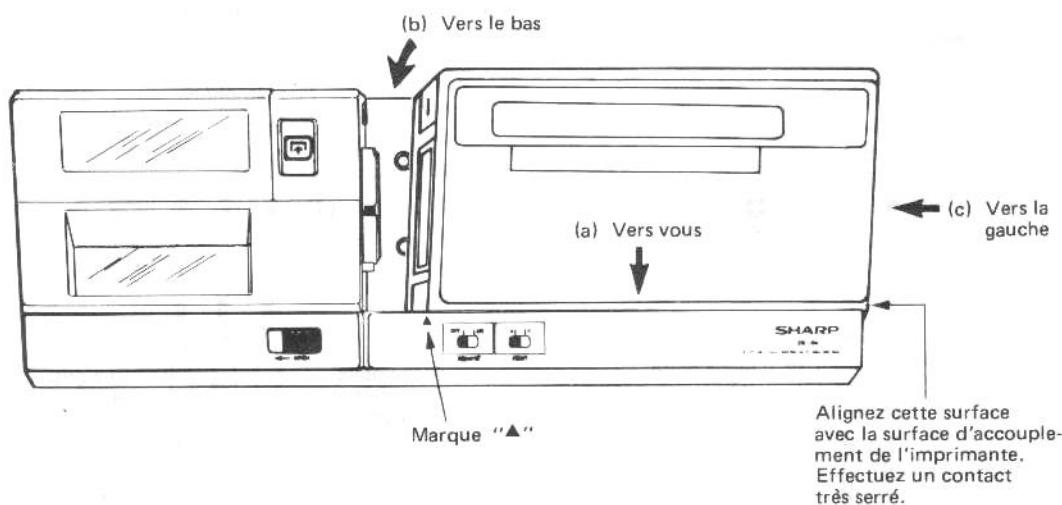
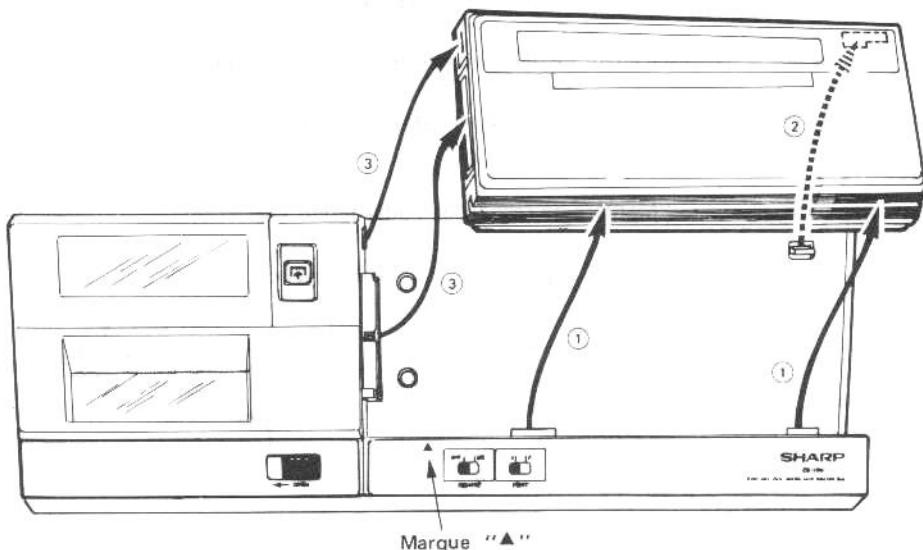
- (1) Mettez l'ordinateur hors-tension (OFF).

Note importante! Ce point est essentiel. Si l'on laisse l'ordinateur sous alimentation (ON), il se peut que toutes les touches soient mises hors fonction. Si cela se produit, appuyez sur le commutateur ALL RESET situé au dos de l'ordinateur tout en maintenant la touche **ON** en position basse.

- (2) Détachez le couvercle protège-broche située sur le côté gauche de l'ordinateur et pressez-le en place sur le dos de l'ordinateur (voir l'illustration).



- (3) Placez la tranche inférieure de l'ordinateur dans le berceau de façon à aligner les languettes-guides de l'imprimante avec les fentes correspondantes de l'ordinateur.
- (4) Déposez l'ordinateur à plat.
- (5) Faites-le glisser doucement vers la gauche de façon à ce que les broches de l'imprimante entrent dans l'ordinateur. (Voir l'illustration).



Evitez de monter en force l'ordinateur sur l'imprimante. S'il s'avère difficile d'effectuer la mise en place, déplacez l'ordinateur avec précaution, légèrement, de droite à gauche pour obtenir le contact correct des surfaces.

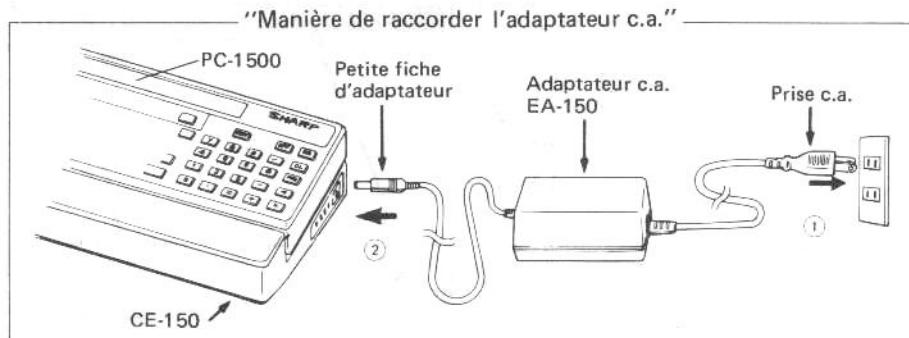
2. Alimentation

L'alimentation ou CE-150 est fournie par les piles Ni-CAD rechargeables. Il s'avère donc nécessaire de recharger les piles immédiatement après le déballage de l'appareil et, plus tard, chaque fois que l'affichage indique le message suivant. (Dans ce cas, l'imprimante est bloquée. Pour la débloquer, appuyez, dans l'ordre, sur les touches **OFF** et **ON** de l'ordinateur après avoir rechargé les piles).

(1) **ERROR 80** ou **ERROR 78**

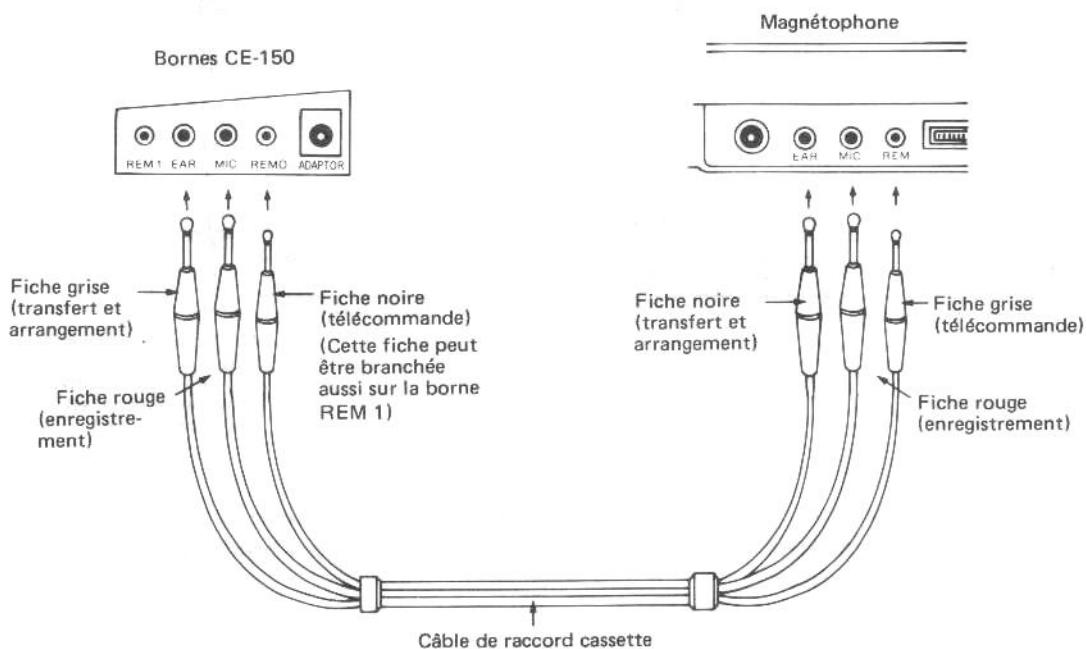
Note: Il se peut que le message d'erreur ERROR soit affiché lors du remplacement des stylos de l'imprimante.

(2) **:CHECK 6** ou **NEW Ø? : CHECK 6**



3. Raccordement d'un magnétophone avec l'interface

Après avoir monté l'ordinateur sur le CE-150, raccordez le magnétophone avec le CE-150 de la façon indiquée par le schéma ci-dessous:



Ci-dessous vous trouverez une description des conditions minimum qu'un magnétophone doit remplir pour pouvoir être branché sur le CE-150:

Article	Conditions à remplir
1. Type de magnétophone	Il est possible d'utiliser n'importe quel magnétophone à cassette, micro-cassette ou à bande ouverte tant qu'il satisfait aux conditions énoncées ci-dessous.
2. Prise d'entrée	Le magnétophone doit posséder une mini-prise d'entrée "MIC". Ne jamais utiliser la prise "AUX".
3. Impédance d'entrée	L'impédance à la prise d'entrée doit être basse (200–1000 ohm).
4. Niveau d'entrée minimum	Inférieur à 3mV ou –50dB.
5. Prise de sortie	Une mini-prise "EXT", "MONITOR", "EAR" ou un équivalent.
6. Impédance de sortie	Doit être inférieure à 10 ohm.
7. Niveau de sortie	Supérieur à 1V (sortie pratique maximum supérieure à 100mW)
8. Distorsion	Doit se situer à 15% dans une gamme allant de 2kHz à 4kHz.
9. Taux de pleurage et scintillation	0,3% au maximum
10. Autres	La vitesse du moteur du magnétophone doit rester constante.

* Au cas où la minifiche fournie avec le CE-150 ne serait pas compatible avec les prises d'entrée ou de sortie de votre magnétophone, vous devrez vous procurer un raccord d'adaptation.

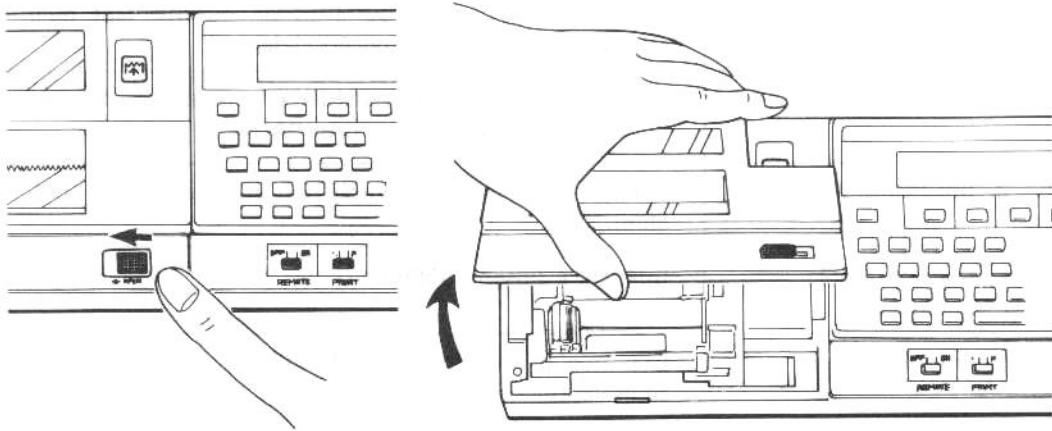
NOTE:

- Certains magnétophones ne peuvent pas être raccordés du à l'incompatibilité des spécifications. D'autres ne sont pas satisfaisants parce que de longues années d'usage ont diminué leur puissance et augmenté leurs parasites et leur distorsion.
- Précautions à prendre lors de l'utilisation d'un magnétophone
 - (1) Lors d'un transfert ou d'un arrangement, utilisez toujours le magnétophone sur lequel l'enregistrement a été effectué. Il peut arriver que le transfert ou l'arrangement soit impossible si l'on néglige de prendre cette précaution.
 - (2) Assurez-vous de la propreté des têtes du magnétophone, afin d'éviter une augmentation du niveau de distorsion ou une diminution du niveau d'enregistrement.
 - (3) Evitez d'utiliser des bandes à réponse en fréquences très basse ou encore des bandes froissées ou égratignées.

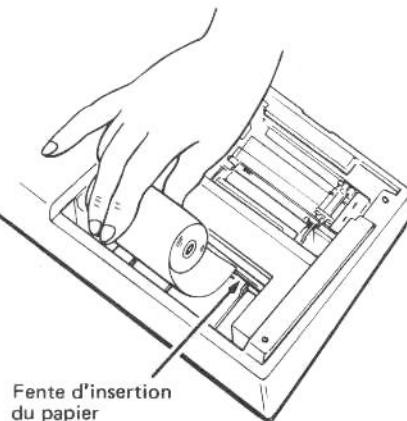
4. Chargement du papier

Pour plus de détails, reportez-vous au manuel CE-150.

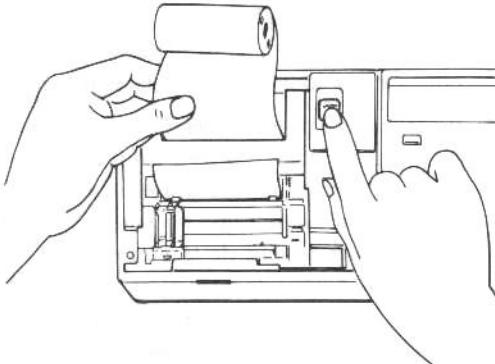
- (1) Poussez le levier de fermeture dans la direction de la flèche pour ôter le couvercle de l'imprimante.



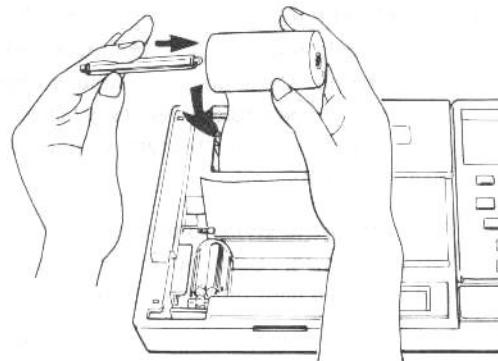
- (2) Coupez proprement l'extrémité du rouleau de papier, puis introduisez-la dans la fente d'insertion du papier. (Un pli ou un gondolage quelconque à l'extrémité du papier risque d'empêcher son insertion).



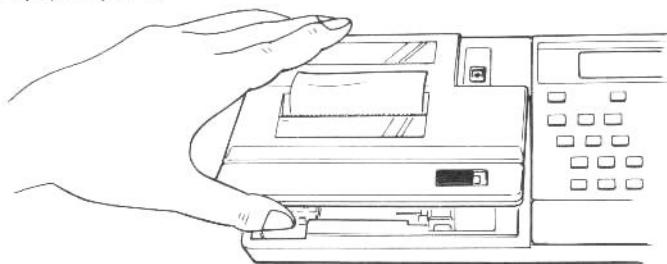
- (3) Mettez l'ordinateur en marche en appuyant sur la touche **ON**, puis sur la touche **PRINT** pour effectuer l'alimentation en papier. A ce moment, laissez le papier sortir de 3 à 5 cm de l'imprimante.



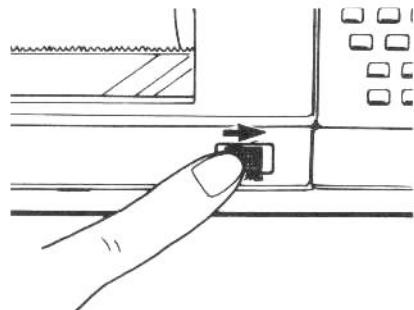
- (4) Introduisez l'arbre dans le rouleau de papier et placez l'ensemble dans le casier à papier.



- (5) Remettez le couvercle en place en faisant glisser à ce point l'extrémité du papier par la fente du coupe-papier.

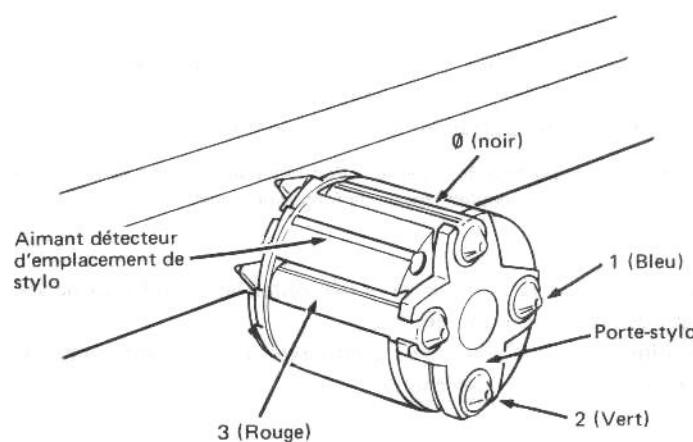


- (6) Fermez le couvercle à l'aide du levier.



5. Remplacement des stylos

Quatre stylos différents peuvent être montés sur ce dispositif. Le schéma ci-dessous indique les différents emplacements de montage de stylos. Consultez le manuel d'instruction de l'interface pour plus de détails.

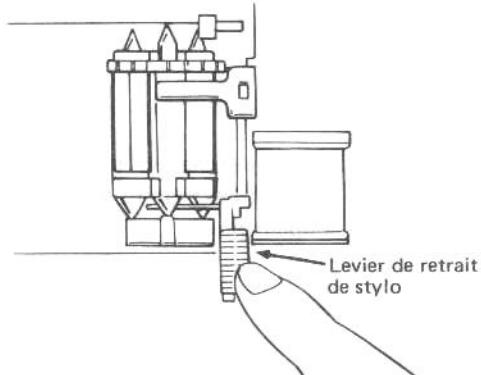


L'instruction COLOR désigne les emplacements de stylos, dans le sens des aiguilles d'une montre, à partir de l'aimant détecteur avec les numéros 0, 1, 2, et 3 comme indiqué sur la figure ci-contre. (Le porte-stylo tourne dans le sens inverse des aiguilles d'une montre pour amener le stylo choisi en haut.)

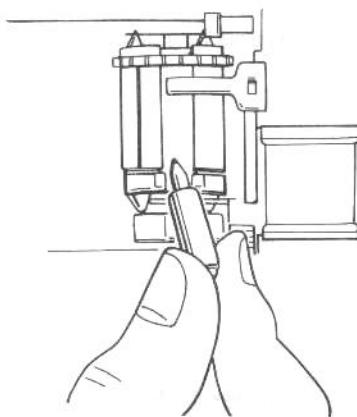
Pour monter ou remplacer les stylos, suivez les instructions ci-dessous:

- (1) Maintenez la touche **O** de l'ordinateur en position basse et appuyez sur la touche **F** de l'imprimante. Cette opération amène l'imprimante dans la position de remplacement de stylo: le porte-stylo se déplace vers la gauche et tourne. Quand le stylo situé en position supérieure a été remplacé, le porte-stylo se déplace vers la droite. (Relâchez la touche quand il commence à se déplacer).
- (2) Pour déposer le stylo, appuyez sur le levier de retrait de stylo. Cette opération fait sauter le stylo situé en haut du porte-stylo.

(Note): Pour empêcher le stylo de retomber dans le boîtier de l'imprimante, retenez-le légèrement.



- (3) Mettez en place un nouveau stylo.



- (4) Appuyez à nouveau sur la touche **F** pour monter ou démonter le stylo suivant. Le porte-stylo revient sur la gauche et tourne jusqu'à ce que le stylo suivant arrive à la position supérieure. Vous pouvez maintenant procéder à son extraction et remplacement comme décrit sous (1) et (2).
- (5) Quand le montage ou remplacement de stylo est terminé, appuyez sur la touche **F** de l'imprimante en maintenant la touche **CL** de l'ordinateur en position basse. Ceci libère l'imprimante de sa position de remplacement de stylo et le stylo revient sur la gauche.

(Note): Montez toujours quatre stylos sur le porte-stylo. Même si l'on omet simplement de monter un seul stylo, le mécanisme de changement de couleur risque de fonctionner mal.

Entretien des stylos:

Montez les stylos sur l'imprimante quand vous vous en servez et démontez-les quand vous ne vous en servez pas. Capuchonnez-les avant de les ranger dans leur étui.

Si vous les laissez en place sur l'imprimante et que vous ne vous en servez pas pendant longtemps, ou si vous les laissez à l'air libre, l'encre risque de se dessécher.

B. UTILISATION DU MAGNETOPHONE A CASSETTE

LES INSTRUCTIONS POUR L'IMPRIMANTE ET LE MAGNETOPHONE DECRITES CI-APRES NE SONT DISPONIBLES QUE SUR L'IMPRIMANTE OPTIONNELLE CE-150 (AVEC INTERFACE A CASSETTE INCORPOREE). L'ORDINATEUR N'ETANT PAS EQUIPE DE CES INSTRUCTIONS, LEUR UTILISATION N'EST POSSIBLE QUE LORSQUE L'ORDINATEUR EST CONNECTE A LA CE-150. PAR CONSEQUENT, VEILLEZ A CONNECTER L'ORDINATEUR SUR LA CE-150 POUR PROGRAMMER EN UTILISANT CES INSTRUCTIONS.

1. Opération du magnétophone

Nous vous recommandons de procéder de la manière suivante et d'utiliser un programme petit et simple, car s'il se présente un problème il sera plus facile de ré-effectuer l'opération. Entrez maintenant le programme.

Pour préparer le magnétophone pour un transfert de programme et de données, il est nécessaire de passer par les étapes suivantes:

1. Placez le commutateur "remote" (à distance) de l'interface sur OFF.
2. Introduisez une bande (ou cassette) dans le magnétophone (très important!).
3. Trouvez une portion de bande vierge. Si la bande est neuve, faites défiler la bande de guidage jusqu'à la bande magnétique proprement dite. Si votre magnétophone est doté d'un compteur, notez le numéro indiqué à cet instant: cela vous sera très utile lorsque vous cherchez à localiser le programme que vous allez enregistrer. Bon, passons à la suite . . .
4. Si votre magnétophone est doté d'un contrôle de volume automatique, placez-le en position automatique. S'il n'a qu'un contrôle manuel de volume, placez-le à mi-chemin entre le niveau moyen et le niveau maximum (c'est-à-dire au 3/4). S'il existe un contrôle de tonalité, placez-le à mi-chemin entre les aiguës et le milieu entre les aiguës et les graves. (c'est-à-dire au 3/4 en partant des graves).
5. Placez le commutateur "remote" de l'interface sur ON. Si votre magnétophone ne possède pas de "remote" (contrôle à distance), c'est-à-dire qu'il n'y a pas d'endroit où brancher l'un des câbles du réseau de raccordement, utilisez la touche "pause" pour arrêter temporairement l'enregistrement. S'il n'y a pas de touche "pause" non plus, vous rendez votre travail un peu trop difficile. Il vaudrait mieux vous procurer un autre magnétophone pour vous rendre votre vie de programmeur plus facile. Et pendant que vous y êtes, prenez-en un doté d'un contrôle à distance et d'une touche "pause".
6. Appuyez simultanément sur les deux touches RECORD et PLAY. Si votre magnétophone n'a pas de touche "pause", effectuez cette opération juste avant de procéder au transfert du programme.

Prêts? Très bien, l'aventure commence maintenant pour de bon

2. La commande CSAVE

Le compte à rebours commence. Vérifiez vos ceintures

- | | | |
|--|---------------------|-----------|
| 1. Magnétophone prêt? | oui _____ | non _____ |
| 2. Bande ou cassette dedans? | oui _____ | non _____ |
| 3. Ordinateur en marche? | oui _____ | non _____ |
| 4. Programme entré? | oui _____ | non _____ |
| 5. Raccordement effectué correctement? | oui _____ | non _____ |
| 6. Numéro du compteur de bande noté? | oui _____ | non _____ |
| 7. Etes-vous habillé? | aucun rapport _____ | |

Si vous avez répondu "NON" à l'une de ces questions, relisez la section précédente pour résoudre le problème.

Bon, maintenant une chose encore et vous serez prêt à démarrer:

En même temps que la commande d'enregistrement vous devez donner au programme un "nom de fichier" pour des raisons de référence. La longueur de ce nom de fichier ne doit pas dépasser 16 caractères. Pour enregistrer le programme avec un nom de fichier, écrivez:

CSAVE [SHIFT] " PROG.-1 [SHIFT] "

Votre programme sera enregistré sous le nom de "PROG.-1". Vous êtes libres de le nommer comme vous voulez pourvu qu'il soit facile de s'en souvenir et que sa longueur ne dépasse pas la limite de 16 caractères. La partie dépassant cette limite sera simplement ignorée. Nous vous recommandons de commencer dès maintenant un carnet de bord dans lequel vous noterez le nom des programmes, le numéro de début et de fin de l'enregistrement sur bande (si votre magnétophone est doté d'un compteur) et une brève description de la fonction du programme.

Appuyez sur la touche [ENTER]. Vous devriez entendre maintenant le son aigu d'un avertisseur et la bande devrait défiler. De plus l'indicateur "busy" (occupé) devrait s'allumer pour vous indiquer que l'ordinateur est occupé à transférer votre programme de sa mémoire à la bande. Si cela ne se produit pas, recommencez de nouveau au début de cette section. Il m'a fallu seulement 10 tentatives avant d'arriver à le faire correctement! Ne vous découragez donc pas. Si moi j'ai réussi à le faire, vous le pouvez aussi.

Dès qu'il arrive au bout du programme l'ordinateur éteint l'indicateur "busy", le magnétophone s'arrête et le symbole de guidage réapparaît sur l'affichage. Félicitations! Vous venez d'enregistrer un programme pour la première fois! Pour nous assurer cependant que tel a bien été le cas nous allons le retransférer dans la mémoire de la manière indiquée dans la section suivante.

3. La commande CLOAD

Maintenant que votre programme a été enregistré sur bande, vous voudrez certainement vérifier si le programme y est réellement. Ce qui est relativement facile à faire à l'aide de la commande CLOAD, bien sûr.

Vous allez vous demander: "Mais quel est son effet?". Ecrire simplement CLOAD?. L'ordinateur compare le programme enregistré avec celui dans sa mémoire. Si tout s'est bien passé, il affichera calmement votre nom de fichier et finira sa vérification. Si tout ne s'est pas bien passé, il affichera un message d'erreur, généralement ERROR 43. Ce message vous indique que le programme sur bande diffère de celui de la mémoire du SHARP. Effacez la portion de bande où l'erreur s'est produite et recommencez après avoir vérifié tous les raccords et augmenté légèrement le volume et la tonalité.

Ci-dessous vous trouverez les instructions concernant le rechargeement du programme dans l'ordinateur.

1. Placez le commutateur "remote" de l'interface sur OFF.
2. Rebobinez la bande jusqu'au point de départ en vous servant du compteur. (Très pratique, ce compteur, n'est-ce pas?)
3. Arrêtez le rebobinage.
4. Placez le commutateur "remote" sur ON.
5. Appuyez sur la touche PLAY.
6. Ecrivez:

CLOAD [SHIFT] "PROG.-1" [SHIFT]
et appuyez sur la touche [ENTER].

(Rappelez-vous que "PROG.-1" constitue le nom de fichier que vous avez donné à votre programme. Si vous l'avez enregistré sous un autre nom, il vous faudra utiliser ce dernier au lieu de "PROG.-1".)

7. L'indicateur "busy" s'allume maintenant et le programme est ramené dans la mémoire de l'ordinateur pour être utilisé.
8. Ecrivez RUN et à ce moment le programme que vous aviez enregistré auparavant réapparaît. Comme la copie sur bande est restée intacte, vous pouvez recharger le même programme aussi souvent que vous voudrez!

4. Les Commandes PRINT# et INPUT#

PRINT# :

Maintenant que nous avons expliqué l'utilisation de CSAVE et CLOAD, nous allons introduire deux commandes qui leur ressemblent. La commande PRINT# enregistre la valeur d'une variable ou d'une série de variables sur bande, alors que CSAVE enregistre un programme. La raison pour laquelle nous enregistrons des données, c'est que cela nous permettra de nous servir des mêmes données dans un autre programme. Par exemple, on se sert de la variable T\$ dans le programme suivant:

```
10: PRINT "QUEL EST VOTRE NOM?"  
20: INPUT T$  
30: PRINT T$
```

Si vous voulez enregistrer la valeur de T\$ pour l'utiliser dans un autre programme, vous devez lancer la commande PRINT# pour l'enregistrer sur bande. Il y a deux manières d'effectuer cette opération:

1. Manuellement
2. Par l'intermédiaire d'un programme

Note: comme le magnétophone est nécessaire pour cette opération, préparez-le en vue de recevoir des données. Si vous n'êtes pas sûr de savoir comment faire, reportez-vous à la section précédente.

1. La méthode manuelle

Plusieurs formats aux choix sont possibles:

CHOIX 1:

Après le passage d'un programme, passez sur le mode RUN et écrivez:

P R I N T SHIFT # A, B, C (puis appuyez sur ENTER)

Le magnétophone se met alors en marche et enregistre les valeurs de toutes les variables sur la bande.

CHOIX 2:

Ecrivez selon le format suivant si vous voulez identifier seulement certaines variables à enregistrer:

P R I N T SHIFT # "nom de fichier"; A, B, C

Dans cet exemple-ci, vous avez spécifié les variables A, B et C comme étant celles qui sont à enregistrer sur bande sous le nom de fichier donné.

CHOIX 3:

Vous pouvez aussi spécifier toutes les valeurs de variables apparentées en écrivant:

P R I N T SHIFT # "nom de fichier"; B (*)

Le symbole * causera l'enregistrement de toutes les variantes de "B", y compris B (1) et B lui-même.

2. Par l'intermédiaire d'un programme

Dans ce cas-ci, il suffit d'affecter un numéro de ligne à la commande PRINT# de votre programme. Vous pouvez choisir un format quelconque parmi ceux décrits ci-dessus. Quand le SHARP rencontre ce numéro de ligne il met automatiquement le magnétophone en marche pour amorcer le transfert de données sur la bande. Une fois de plus, N'HESITEZ PAS à expérimenter. Si vous rencontrez des difficultés, reportez vous aux sections précédentes. Parfois l'on oublie les choses les plus simples.

INPUT# :

Cette commande permet les mêmes formats que PRINT#. A la différence de PRINT# qui transfère des données de l'ordinateur sur la bande, INPUT# transfère des données de la bande à l'ordinateur.

Vous pouvez utiliser la commande INPUT# manuellement ou comme partie d'un programme. N'oubliez pas de préparer le magnétophone avant de vous servir de cette commande.

NOTE: Si, comme partie de votre ligne de commande INPUT#, vous spécifiez plus de variables qu'il n'en existe sur la bande, l'ordinateur affectera aux variables en excès la valeur 0. Si vous spécifiez moins de variables qu'il n'en existe sur la bande, l'ordinateur ignorera celles qui ne sont pas spécifiées.

5. Programmes de mise en forme (MERGE)

La commande MERGE vous permet de stocker en même temps deux programmes dans la mémoire de l'ordinateur. Supposons, par exemple, que la mémoire de l'ordinateur contienne le programme suivant:

```
10: PRINT "MOINS VALEUR DEDUCTIBLE"  
20: PRINT "ENTREZ METHODE?"  
30: INPUT A
```

A ce moment, vous vous souvenez tout d'un coup que vous possédez, enregistré sur bande, un morceau de programme semblable sous le nom de fichier "DEP1". Vous voulez, bien sûr, vérifier si ce programme ne contient pas des morceaux qui pourraient vous servir dans le programme que vous êtes en train de fabriquer. Le premier pas consistera à trouver la bande contenant "DEP1". Amenez la bande à l'endroit où "DEP1" commence et écrivez:

M E R G E SHIFT " D E P 1 SHIFT "

puis appuyez sur **ENTER**

L'ordinateur va maintenant charger "DEP1" dans sa mémoire en plus du programme ci-dessus. Quand "DEP1" est chargé, vous trouverez dans la mémoire quelque chose du type suivant:

```
10: PRINT "MOINS VALEUR DEDUCTIBLE"  
20: PRINT "ENTREZ VOTRE METHODE"  
30: INPUT A  
10: PRINT "MONTANT DES INTERETS"  
20: PRINT "SOMME EMPRUNTEE?"  
30: INPUT B  
:  
(etc)
```

Notez que, à la différence de la commande CLOAD, le nouveau programme n'a PAS remplacé celui en existence. Vous remarquerez aussi que vous avez des numéros de lignes en double à présent.

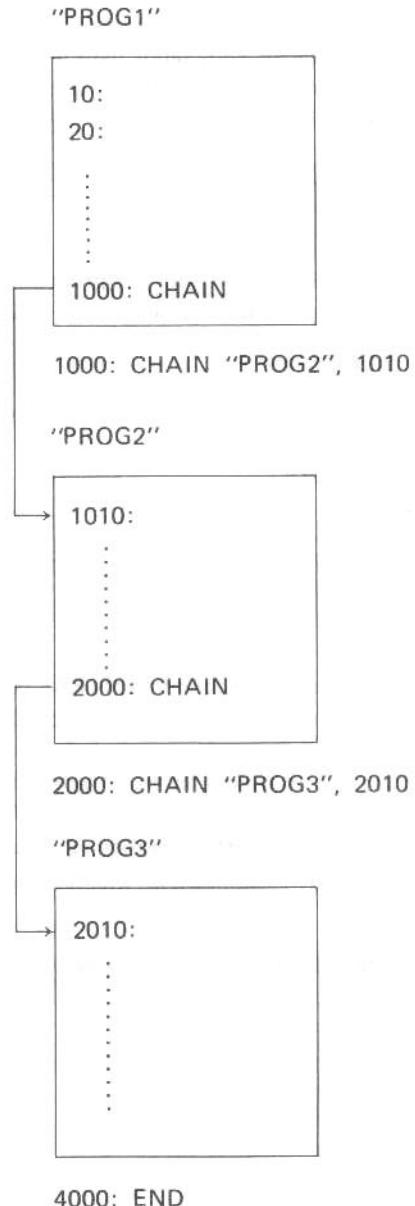
A ce moment vous réalisez que la partie de "DEP1" qui commence par la ligne 40 vous sera utile dans le programme de "Moins-valeur" que vous êtes en train d'écrire. Effacez simplement les trois premières lignes du nouveau programme (commençant par 10:PRINT "MONTANT DES INTERETS") et mettez en page le reste du nouveau programme de la manière qui vous convient le mieux. Et voilà: vous vous êtes épargné beaucoup de peine en utilisant un morceau de programme déjà fabriqué.

Une fois encore: ne vous gênez pas pour expérimenter. Essayez de fusionner d'autres sections de programme.

6. L'instruction CHAIN

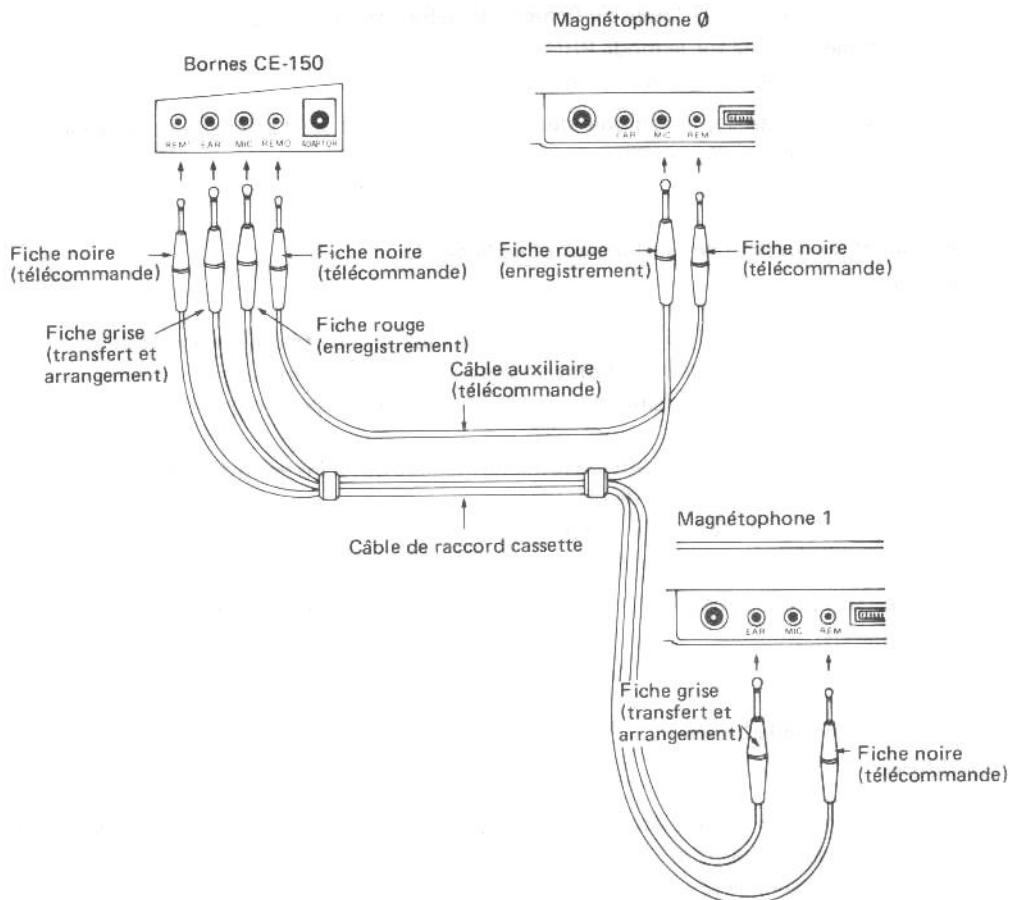
CHAIN constitue une instruction de programme. On peut l'utiliser uniquement à l'intérieur d'un programme. On ne peut pas l'utiliser manuellement comme CSAVE, CLOAD et MERGE. L'instruction CHAIN vous permet de passer un programme trop grand pour être logé tout entier dans la mémoire en une fois. Ce type de programme est coupé en morceaux à la fin de chacun desquels nous placerons une instruction CHAIN. Nous pourrons ensuite enregistrer ces morceaux sur bande.

Supposons, comme exemple, que nous ayons trois sections de programme nommées PROG1, PROG2, PROG3. Chacune se termine par une instruction CHAIN.



Quand l'ordinateur rencontre l'instruction CHAIN au cours de l'exécution, la nouvelle section est rappelée dans la mémoire et exécutée. De cette façon, toutes les sections finiront par passer.

7. Utilisation de deux magnétophones



Lorsqu'on utilise deux magnétophones, l'un sert à enregistrer et l'autre réécouter. Comme l'indique le schéma, l'interface imprimante/cassette et les deux magnétophones sont raccordés au moyen d'un câble de raccord cassette et d'un câble auxiliaire pour la télécommande.

- Le CE-150 est équipé de deux bornes de télécommande, REM 0 et REM 1; on peut se servir de l'une ou de l'autre. Durant l'opération programmée (non manuelle) cependant, le programme indique si le magnétophone est connecté à la borne REM 0 ou la borne REM 1. Leur connexion est donc à effectuer en accord avec le programme.

La manière d'utiliser le second magnétophone est expliquée dans ce chapitre.

1. Sauvegarde

Procédures:

- Ecrivez RMT OFF, puis appuyez sur **ENTER** pour régler à nouveau la deuxième fonction de télécommande (pour le contrôle de magnétophone 1 de la figure précédente).
- Placez une bande dans le magnétophone.
- Ecrivez RMT ON, puis appuyez sur **ENTER** pour régler la deuxième fonction de télécommande.
- Réglez le volume et la tonalité de la même manière que dans le cas d'un seul magnétophone. (Voir plus haut).
- Appuyez simultanément sur RECORD et PLAY.

- (6) Exécutez l'instruction ENREGISTREMENT.

Programme: CSAVE-1 "nom de fichier" [ENTER]

Données: PRINT# -1 "nom de fichier"; variable, variable, ...

(Exemple) Placez sur le mode PRO ou RUN.

CSAVE-1 "PR-1" [ENTER]

Après la sauvegarde, le symbole de guidage réapparaît sur l'affichage et la bande s'arrête.

Rebobinez-la pour l'arrangement.

2. Arrangement du contenu de l'ordinateur et de la bande

Procédures:

- (1) Ecrivez RMT OFF [ENTER] pour remettre les fonctions de télécommande à zéro.
- (2) Robobinez la bande jusqu'au point de départ à l'aide du compteur.
- (3) Entrez RMT ON, puis appuyez sur la touche [ENTER] pour régler les fonctions de télécommande.
- (4) Réglez le volume et la tonalité de la même manière que dans le cas d'un seul magnétophone.
- (5) Appuyez sur la commande de réécoute.

CLOAD?-1" nom de fichier" [ENTER]

(Exemple) Placez sur le mode PRO ou RUN. CLOAD?-1 "PR-1" [ENTER]

L'exécution est achevée quand les deux versions concordent et que le symbole de guidage réapparaît.

3. Transfert à partir de la bande

Procédures:

- (1) Entrez RMT OFF, puis appuyez sur la touche [ENTER] pour remettre les fonctions de télécommande à zéro.
- (2) Placez une bande avec enregistrement dans le magnétophone.
- (3) Ecrivez RMT ON, puis appuyez sur [ENTER] pour régler les fonctions de télécommande.
- (4) Réglez le volume et la tonalité de la même manière que dans le cas d'un seul magnétophone.
- (5) Appuyez sur la commande de réécoute.
- (6) Exécutez l'instruction TRANSFERT

Programme: CLOAD-1 "nom de fichier" [ENTER]

Données: INPUT# -1, "nom de fichier"; variable, variable, [ENTER]

(Exemple) Placez sur le mode PRO ou RUN.

CLOAD-1 "PR-1" [ENTER]

Après le transfert, le symbole de guidage réapparaît sur l'affichage.

C. UTILISATION DE L'IMPRIMANTE

NOTE: Toutes les explications et les exemples de cette section se basent sur la supposition que vous avez déjà:

- 1) Effectué correctement la connexion entre le PC-1500 et l'interface CE-150.
- 2) Fourni une source d'alimentation au CE-150 sous la forme de piles ou de l'adaptateur EA-150.
- 3) Chargé le papier dans l'imprimante.
- 4) Monté les stylos sur l'imprimante.

Si vous n'avez pas effectué ces opérations préliminaires, retournez à la section 1 de ce chapitre pour les instructions.

C.1. Spécifications de l'imprimante CE-150

Caractères par ligne:	4, 5, 6, 7, 9, 12, 18, ou 36 selon la taille de caractère choisie.
Tailles des caractères:	1,2 x 0,8 mm à 10,8 x 7,2 mm selon la taille de caractère choisie.
Vitesse d'impression:	Maximum: 11 caractères par seconde quand on imprime avec les caractères les plus petits.
Rotation:	L'impression peut se faire dans les deux sens de chacun des axes des X et des Y
Couleurs:	4 – Rouge, bleu, vert et noir
Système graphique:	Traçage de l'axe X-Y
Alimentation en papier:	Manuelle ou programmable.

C.2. La commande TEST

La première chose à faire est de tester le fonctionnement de l'imprimante CE-150. L'ordinateur et l'interface étant sur ON, écrivez:

TEST (et appuyez sur **ENTER**)

L'imprimante va maintenant dessiner 4 carrés, chacun d'une couleur différente. La couleur des carrés, de droite à gauche, correspond à celle que vous avez choisi comme étant couleur 0, 1, 2 et 3 quand vous avez monté les stylos.

C.3. Impression de calculs

Il est possible d'effectuer à l'aide de l'interface CE-150 une copie imprimée d'une série de calculs manuels exécutés sur l'ordinateur PC-1500, comme l'illustre la figure 1:

Figure 1: 12000*.065

780

780+25.56

805.56

SIN 30

0.5

TX=.065

0.065

P=20000*TX

1300

P/12

108.333333

Pour réaliser cette opération, il suffit de placer le commutateur d'impression (Print Switch) de l'interface sur la position "P".



Pour éviter une impression automatique des calculs manuels, il faut placer le commutateur d'impression sur la position “•”. Cette position du commutateur ne vous empêchera pas d'imprimer certains résultats en plaçant une commande LPRINT (voir LPRINT) au début du calcul. D'autres commandes qui déclenchent l'impression ou le dessin tels LLIST, et d'autres, fonctionnent sur ce mode.

Lors de l'impression, la couleur utilisée sera celle spécifiée auparavant. Si vous venez juste de mettre l'appareil en marche, ce sera le stylo de couleur que vous avez choisi comme correspondant à la couleur 0. Pour changer de couleur, vous devrez lancer une commande de Couleur (voir la section appropriée).

La taille des caractères utilisés pour imprimer les calculs manuels correspond à celle spécifiée auparavant. Si vous avez spécifié précédemment des caractères de taille 1 ou 2, cette taille sera celle utilisée. Si la taille spécifiée antérieurement était plus grande que 2, taille 2 sera utilisée.

L'impression automatique établit l'imprimante sur le mode TEXT. Si vous étiez sur le mode GRAPH et que vous souhaitez retourner sur ce mode, vous devez lancer la commande GRAPH. (L'explication des modes de l'imprimante se trouve dans la section suivante).

C.4. Modes de l'imprimante

L'imprimante fonctionne sur l'un de deux modes: TEXT ou GRAPH. Ils correspondent grossièrement à l'écriture à la machine et au dessin. Comme la plupart des commandes ne sont efficaces que sur un seul mode, il est important de choisir le mode correct avant de commencer à donner des instructions.

On se sert du mode TEXT pour imprimer des nombres et des caractères. Le papier à imprimer est divisé en colonnes dont le numéro est en relation avec la taille de caractère spécifiée. Des commandes de contrôle vertical et horizontal permettent de créer le format du texte.

Sur le mode GRAPH, on peut procéder à la création de différents types de figures, de courbes et de tables. Des commandes pour le dessin de lignes continues ou discontinues, à l'aide d'un système de coordonnées relatives ou directes sont à la disposition de l'utilisateur. Tous les dessins sont basés sur le schéma ordinaire de coordonnées X-Y.

Pour spécifier le mode TEXT, il suffit d'écrire l'instruction:

TEXT

Certaines commandes (dont nous parlerons plus loin) provoquent un passage automatique sur le mode TEXT.

La spécification du mode GRAPH est tout aussi simple. L'instruction:

GRAPH

amorcera ce mode et fera passer le stylo à l'extrême gauche du papier.

C.5. Listage du programme

La commande LLIST provoque l'impression du programme courant ou de morceaux de ce programme. La commande LLIST s'avère extrêmement utile durant le développement du programme parce qu'elle permet l'impression au choix de certaines parties du programme.

La forme de la commande LLIST est semblable à celle de la commande LIST. Les subtiles différences sont dues à la plus grande souplesse de la commande LLIST. Les formes de LLIST sont esquissées ci-dessous:

LLIST

- Imprime toutes les lignes de programme en existence à ce moment dans la mémoire.

LLIST expression

- Imprime seulement la ligne de programme dont le numéro est donné par l'expression.

LLIST , expression

- Imprime toutes les lignes de programme jusqu'à celle dont le numéro est donné par l'expression, y compris cette dernière:

LLIST expression,

- Imprime toutes les lignes de programme suivant celle dont le numéro est donné par l'expression, y compris celle-ci.

LLIST expression1, expression2

- Imprime les lignes de programme commençant par la ligne dont le numéro est donné par expression1 et se terminant par la ligne dont le numéro est donné par expression2. Donc, si la commande est:

LLIST 100, 150

toutes les lignes, entre 100 et 150 (s'il y en a) seront listées.

LLIST "étiquette"

- Imprime la ligne de programme contenant l'étiquette donnée.

LLIST "étiquette",

- Imprime les lignes de programme qui suivent la ligne contenant l'étiquette en commençant par celle-ci.

NOTE: Un message d'erreur ERROR 11 signalera la spécification d'une étiquette qui n'existe pas.

Lors de l'impression d'un programme, la couleur utilisée est celle qui a été spécifiée précédemment. Si vous venez juste de mettre l'appareil en marche, ce sera la couleur de stylo que vous avez choisi comme correspondant à Couleur Ø. Pour changer la couleur vous devrez lancer la commande COLOR (voir la section appropriée).

La taille des caractères utilisés pour lister un programme correspond à celle spécifiée auparavant. Si vous avez spécifié précédemment des caractères de taille 1 ou 2, cette taille sera utilisée; si la taille spécifiée antérieurement était plus grande que 2, taille 2 sera utilisée.

La commande LLIST établit l'imprimante sur le mode TEXT. Si vous étiez sur le mode GRAPH et que vous souhaitez retourner sur ce mode vous devez lancer la commande GRAPH.

Pendant le listage du programme, l'ordinateur PC-1500 essaie de cadrer les lignes de programme avec comme objectif la lisibilité. Il accomplit cela en laissant des espaces vides entre les numéros de ligne. Des numéros de ligne d'une largeur de 1 à 3 chiffres seront cadrés à droite dans un champ de 3 caractères. Des numéros imprimés dans un champ de 5 caractères:

```
10: REM LARGEUR 3
20: REM      "
300: REM     "
2001: REM LARGEUR 5
2010: REM      "
```

C.6. Contrôle programmable de l'imprimante

C.6.1. CSIZE

La commande CSIZE spécifie la taille des caractères pour toute impression ultérieure. Les neufs tailles disponibles permettent d'aller de 36 caractères par ligne imprimée à 4 caractères par ligne imprimée. La forme de la commande CSIZE est la suivante:

CSIZE expression
(sur l'un ou l'autre mode)

L'expression doit consister en un nombre allant de 1 à 9. La largeur et la hauteur des caractères pour chaque taille sont indiquées dans le tableau suivant:

Table 1:

CSIZE	1	2	3	4	5	6	7	8	9
Caractères par ligne imprimée	36	18	12	9	7	6	5	4	4
Hauteur de chaque caractère (mm)	1,2	2,4	3,6	4,8	6,0	7,2	8,4	9,6	10,8
Largeur de chaque caractère (mm)	0,8	1,6	2,4	3,6	4,0	4,8	5,6	6,4	7,2

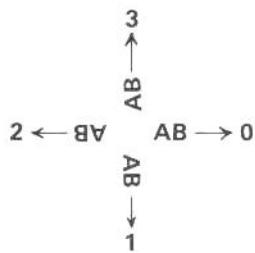
C.6.2. La commande ROTATE

Cette commande ROTATE (faire pivoter) est utilisée sur le mode GRAPH uniquement pour spécifier le sens dans lequel l'impression s'accomplit. Quatre sens sont possibles: vers le haut, vers le bas, de la gauche à la droite et de la droite à la gauche (avec les lettres à l'envers). La figure 1 illustre ces mouvements. La forme de la commande ROTATE est la suivante:

ROTATE expression
(sur le mode GRAPH uniquement)

L'expression doit consister en un nombre de 0 à 3. ROTATE 0 spécifie la façon ordinaire d'imprimer des caractères, de la droite à la gauche.

Figure 1:



C.6.3. La commande COLOR

Cette commande COLOR (Couleur) sert à spécifier le stylo que l'on va utiliser dans les impressions et les dessins ultérieurs. Si chaque emplacement de stylo contient un stylo de couleur différente, la commande COLOR peut servir à changer de couleur de plume. La forme de la commande COLOR est:

COLOR expression
(sur les deux modes)

L'expression doit consister en un nombre entier allant de 0 à 3. Chaque nombre entier représenté par le nombre varie selon l'ordre dans lequel on a monté les stylos sur les porte-stylos. On peut vérifier quel nombre correspond à quelle couleur à l'aide de la commande TEST (décrite plus haut).

Des nombres qui ne sont pas des nombres entiers mais toujours encore de 0 à 3 seront ramenés à des entiers. Tous les autres nombres résulteront en un message d'erreur ERROR 19.

Si l'on arrête puis remet en marche le PC-1500, le stylo correspondant au zéro sera sélectionné.

Sur le mode TEXT, l'exécution de la commande COLOR aura pour conséquence le remplacement du stylo à l'extrême gauche du papier. Sur le mode GRAPH, le stylo retourne sur sa position antérieure.

C.6.4. La Commande LF

La commande LF sert à faire avancer ou reculer le papier dans l'imprimante. La forme de cette commande est la suivante:

LF expression
(sur le mode TEXT uniquement)

Si l'expression consiste en un nombre positif, le papier avance du nombre de lignes spécifié par l'expression. Si le nombre est négatif, le papier reculera du nombre de lignes spécifié par la valeur absolue de ce nombre. La figure 2 illustre l'opération:

Figure 2: (sens négatif)



L'étendue réelle du mouvement du papier dépend de la taille des caractères en usage lorsque la commande LF est lancée.

Quand le papier se déplace en sens inverse (c'est-à-dire qu'il est tiré en arrière) un dispositif de comptage interne l'empêche de reculer de plus de 10,24 cm.

NOTE: N'essayez jamais d'insérer du papier durant le fonctionnement du mécanisme d'alimentation papier. Vous risqueriez d'endommager l'imprimante.

C.6.5. La commande LPRINT

La commande LPRINT est la principale commande servant à afficher un texte sur l'imprimante. Elle ressemble en nature à la commande PRINT du PC-1500 et elles ont plusieurs formes en commun. Cependant, dû aux caractéristiques supplémentaires de l'imprimante, l'effet des instructions LPRINT est plus complexe. Nous concentrerons donc notre attention sur les finesse qu'implique l'utilisation de l'instruction LPRINT.

L'examen de la commande LPRINT qui suit se base sur un fonctionnement sur le mode TEXT seulement. Quoique les formes données soient possibles sur le mode GRAPH, leur opération sera différente.

L'impression d'un élément unique reste en général la même:

LPRINT élément

format dans lequel l'élément consiste en une expression, une chaîne de caractères, un nombre ou le nom d'une variable dont le contenu est à imprimer. Comme d'habitude, les caractères sont cadrés à gauche et les nombres, cadrés à droite.

De la même manière que le curseur sur l'affichage, si le stylo n'est pas placé à l'extrême gauche du papier, l'impression commencera à partir de la position du stylo. Il est possible de changer la position du stylo à l'aide de l'instruction LCURSOR ou la proposition TAB (voyez ci-dessous).

Essayer d'imprimer un élément trop long pour une ligne à cause de la taille des caractères causera des difficultés. Si l'élément est constitué par un nombre, un message d'erreur ERROR 76 se produira. Si l'élément est constitué par une chaîne de caractères, l'impression de la chaîne continuera sur la ligne suivante.

Le souci de la taille d'un élément imprimé joue un rôle important dans l'instruction LPRINT à deux éléments, dont la forme est la suivante:

LPRINT élément1, élément2

L'utilisation de CSIZE 1 garantit l'impression sur une ligne de deux éléments numériques. Dans ce cas, les deux éléments seront cadrés de la manière ordinaire dans les deux moitiés de la ligne imprimée. Le format devient plus compliqué dans le cas d'instruction LPRINT impliquant des chaînes. Si deux éléments entrent dans la ligne, ils seront cadrés de façon ordinaire et imprimés entre deux lignes. Dans le cas de caractères de tailles plus grandes, les deux éléments sont imprimés sur deux lignes successives.

On peut aussi se servir du point-virgule dans l'instruction LPRINT. Sa fonction est à la fois d'indiquer l'espace minimum entre éléments et de grouper à la fin d'une instruction les éléments imprimés en succession sur une ligne. Dans les deux cas, si la longueur totale des éléments dépasse la capacité de la ligne imprimée, l'impression des éléments se fera sur autant de lignes successives que nécessaire. L'instruction LPRINT avec le point-virgule se présente sous la forme suivante:

LPRINT élément1 ; élément2 ; . . . (etc)

ou encore la forme:

LPRINT liste d'éléments

Parmi les formes mentionnées ci-dessus, plusieurs sont employées dans le programme de démonstration qui suit:

```
10 A$ = "ABCDEFG"  
20 B = 123456  
30 FOR I = 1 TO 3  
40 CSIZE I  
50 LPRINT A$  
60 LPRINT A$, B  
70 LPRINT A$; B  
80 LF 5  
90 NEXT I
```

Dont la sortie est:

```
ABCDEFG  
ABCDEFG  
ABCDEFG 123456
```

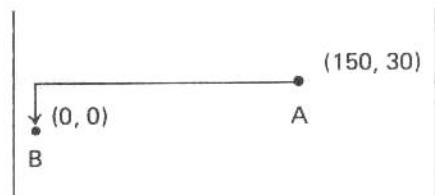
```
ABCDEFG  
ABCDEFG  
123456  
ABCDEFG 123456
```

```
ABCDEFG  
ABCDEFG  
123456  
ABCDEFG 1234  
56
```

Il est possible d'utiliser la commande LPRINT sans spécifier les éléments et cela sur le mode TEXT ou GRAPHIC:

LPRINT

Utilisée de cette manière, elle déclenchera un retour de chariot et (le passage à la ligne suivante). Si cela ne se produit pas, réglez les compteurs du mode GRAPHIC. Ainsi, dans l'exemple suivant, quoique l'on se soit servi de l'instruction LPRINT pour déplacer le stylo du point A au point B, l'imprimante se croit encore aux coordonnées (150, 30) et exécute les commandes suivantes comme s'il en était ainsi.



L'instruction LPRINT incorpore aussi une proposition USING qui fonctionne de la même manière que dans l'instruction PRINT. La proposition USING n'est permise que dans une instruction LPRINT effectuée sur le mode GRAPH.

C.6.6. L'instruction LCURSOR

L'instruction LCURSOR sert à placer le stylo sur le papier, d'une manière semblable à l'instruction CURSOR qui déplace le curseur sur l'affichage. Sa forme est la suivante:

LCURSOR position
(sur le mode TEXT uniquement)

La position de caractère sur laquelle on peut placer le stylo dépend, bien sûr, de la taille de caractère utilisée. En général, on peut placer le stylo à un espace de moins que le maximum pour la taille de caractère en question. Reportez-vous à la commande CSIZE dans ce chapitre pour la liste des largeurs de lignes imprimées pour chaque taille de caractère.

C.6.7. L'instruction TAB

L'instruction TAB est identique à LCURSOR sauf que l'on peut l'utiliser dans une instruction LPRINT. Ce type d'instruction LPRINT se présentera sous la forme suivante:

LPRINT TAB position; liste d'éléments

Les remarques faites au sujet de l'expression de la position dans le cas de LCURSOR s'appliquent aussi à TAB. Si la liste d'éléments est vide, le résultat net de l'instruction listée ci-dessus sera un passage à la ligne suivante.

C.6.8. La commande SORGN

La commande SORGN sert à établir l'origine du système de coordonnées X-Y pour les commandes d'exécution de graphiques ultérieurs. La commande SORGN établit la position présente du stylo comme l'origine. On utilisera donc, en général, cette commande immédiatement après des instructions qui placent le stylo à un point donné du papier. La forme de la commande SORGN est simplement:

SORGN
(sur le mode GRAPH seulement)

NOTE: L'imprimante CE-150 permet de spécifier la position du stylo à un point situé hors de la gamme de positions à partir desquelles il est possible de dessiner. Dans ce cas, le stylo se déplace aussi loin que possible puis son mouvement semble être "coupé". Si le stylo entre dans ce champ imaginaire et que l'on lance la commande SORGN, les instructions d'impression ou de dessin ultérieures resteront sans effet. Ce phénomène se perçoit comme une erreur du programme ou comme si l'interface était endommagée.

Le programme suivant établit le point d'origine à 100 unités plus haut et 100 unités plus à droite que la position actuelle du stylo. Il dessine ensuite un carré de 10 unités dont l'un des angles constitue le nouveau point d'origine:

```
10 GRAPH
20 LINE (0, 0) - (100, 100), 9
30 SORGN
40 LINE (0, 0) - (10, 10), 0, 0, B
50 TEXT
60 END
```

C.6.9. L'instruction GLCURSOR

L'instruction GLCURSOR place le stylo à n'importe quelle coordonnée X-Y sans laisser de trace sur le papier. La forme de l'instruction GLCURSOR est la suivante:

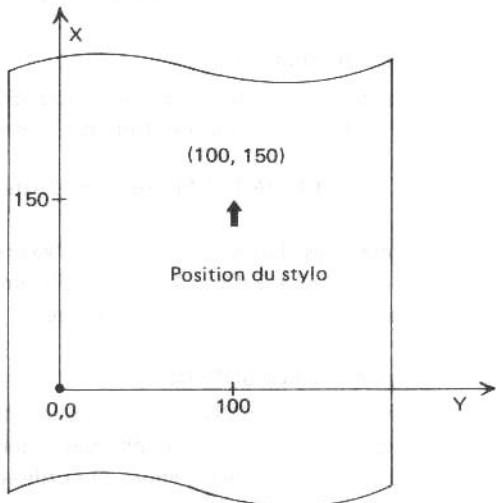
GLCURSOR (expression1, expression2)

Les deux expressions doivent consister en un nombre situé dans la gamme allant de -2048 à +2047. L'expression1 représente la distance X parcourue jusqu'au point de destination et l'expression2 la distance Y parcourue jusqu'au point de destination.

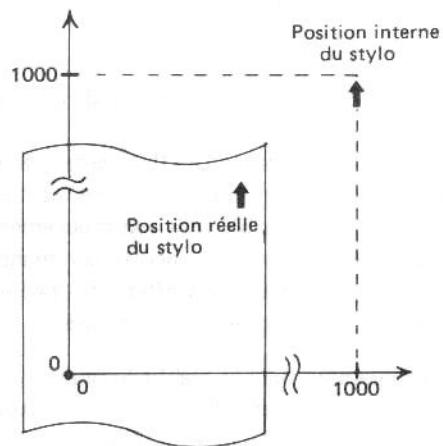
NOTE: Si le point de destination se situe hors du champ de points dans lequel le stylo peut se déplacer réellement, le stylo s'arrêtera sur le bord du papier, mais, intérieurement, le compteur qui contrôle le mouvement du stylo continue abstraitemment vers son but.

Les exemples ci-dessous illustrent l'utilisation de l'instruction GLCURSOR:

Dans l'exemple à droite le stylo se déplace vers la position (100, 150).



Dans cet exemple, le stylo est incorrectement placé dans la région "imaginaire" en position (1000, 1000). En fait, le stylo se déplace vers le bord droit et fait revenir le papier. Quand elle atteint le côté droit, elle continue vers le haut jusqu'à ce qu'elle atteigne la limite de recul de 10 cm où elle s'arrête.



C.6.10. La commande LINE

La commande LINE constitue la commande principale du mode GRAPH. Elle spécifie le mouvement du stylo d'un point à l'autre. Si le stylo descend sur le papier dans son mouvement, elle trace une ligne. La commande LINE, permet aussi de tracer des lignes discontinues avec des traits de huit longueurs différentes. La première forme de la commande LINE est la suivante:

LINE (X1, Y1) – (X2, Y2), type de ligne, couleur

La valeur des expressions X1 et Y1 détermine le point de départ de la ligne et la valeur des expressions X2 et Y2, son point de destination. Les deux valeurs de X et Y doivent se situer dans la gamme allant de -2048 à +2047. Une spécification de valeur hors de cette gamme aura une erreur pour résultat.

Les paramètres de type de ligne et de couleur sont facultatifs. Si on les omet, les valeurs en effet avant la commande seront utilisées. La couleur consiste, bien sûr, en l'une des couleurs représentés par 0, 1, 2 et 3. Le type de ligne doit être représenté par une expression consistant en un nombre de la gamme allant de 0 à 9. Le tableau 2 révèle l'effet correspondant à chaque nombre donné:

Table 2:

Valeur du type de ligne	Type de ligne qui en résulte	
0	Ligne continue	0 -----
1	Traits 0.4 mm	1
2	Traits 0.6 mm	2 -----
3	Traits 0.8 mm	3 -----
4	Traits 1.0 mm	4 -----
5	Traits 1.2 mm	5 -----
6	Traits 1.4 mm	6 -----
7	Traits 1.6 mm	7 -----
8	Traits 1.8 mm	8 -----
9	Stylo levé (pas de ligne)	9

Utilisée d'une autre façon, la commande LINE interprète les spécifications de points comme les extrémités d'une diagonale. Elle passera à l'exécution du tracé d'un carré représenté par la diagonale. Cette commande LINE se présente sous la forme suivante:

LINE (X1, X2) – (Y1, Y2), type de ligne, couleur, B

La majuscule B indique que l'on commande le dessin d'un carré. Les autres paramètres sont les mêmes que pour la forme précédente.

La dernière forme de la commande LINE permet de spécifier une multiplicité de points. Après le premier, chaque point représente le point de destination du segment de la ligne suivante qui doit être tracé. On admet que la position actuelle constitue le point de départ du segment de ligne. Cette dernière forme de la commande LINE se présente sous la forme suivante:

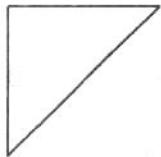
LINE (X1, Y1) – (X2, Y2) – · · · (X6, Y6), type de ligne, couleur

Les trois points servent à indiquer que l'on peut donner une série de spécifications de points, jusqu'à un maximum de six à la file. Remarquez qu'il n'est PAS permis d'utiliser le paramètre B dans cette forme de la commande. Dans l'exemple qui suit, le programme est destiné à faire tracer un triangle à l'aide de quatre segments de ligne. La ligne 15 a simplement pour fonction d'établir l'origine tandis que la ligne 20 exécute le dessin du triangle:

```

10:GRAPH
15:LINE (0, 0)-(10
0, 0),9:SORGN
20:LINE (0, 0)-(50
,50)-(-50, 50)-
(-50, -50)-(0, 0
), 0, 0
30:TEXT

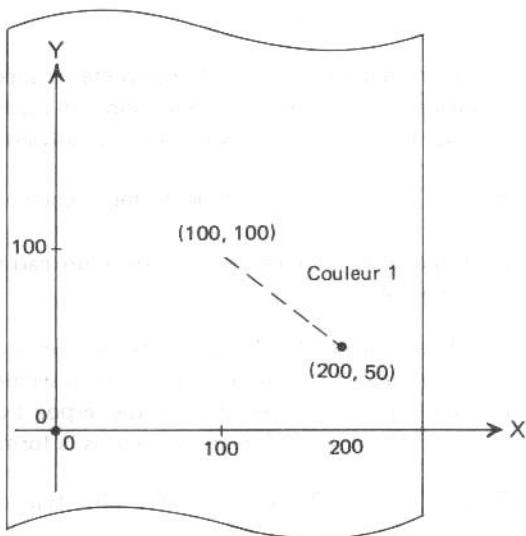
```



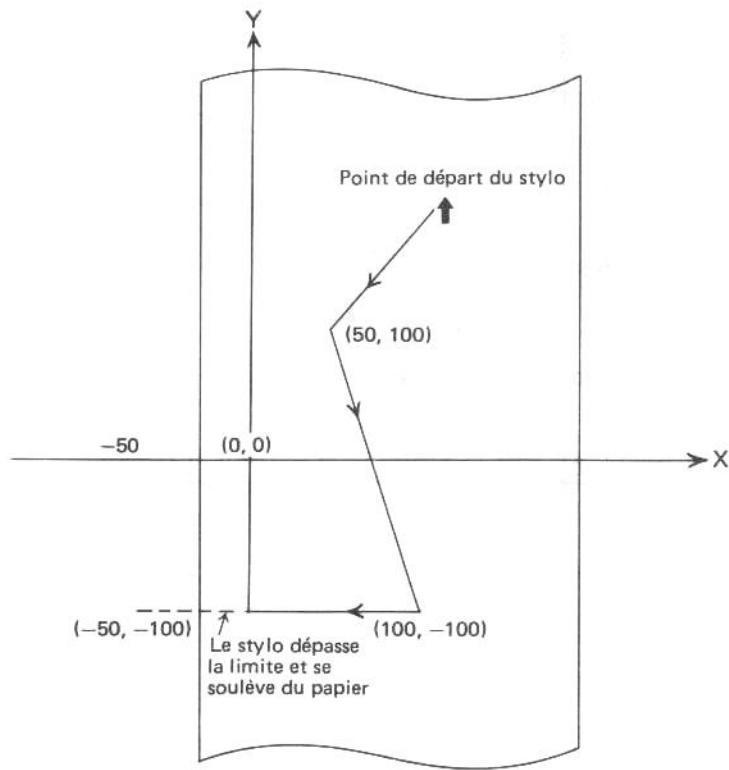
C.6.11. La commande RLINE

La commande RLINE est essentiellement la même que la commande LINE à l'exception du fait que tous les points spécifiés représentent des positions par rapport à la position actuelle du stylo plutôt que par rapport à l'origine. Les formes de commande RLINE sont les mêmes que celles de la commande LINE, à l'exception bien sûr du mot LINE à qui l'on substitue RLINE.

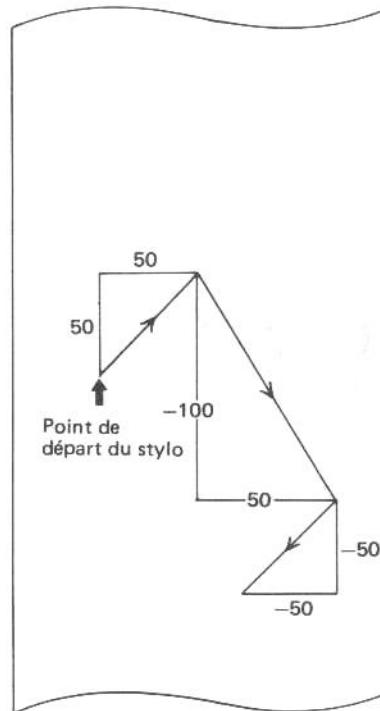
Suivent des exemples: LINE (100, 100) – (200, 50), 2, 1



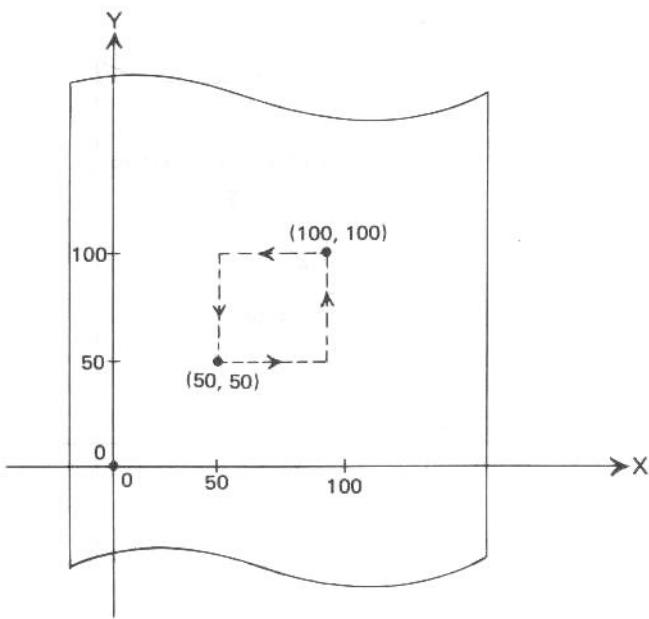
LINE – (50, 100) – (100, -100) – (-50, -100)



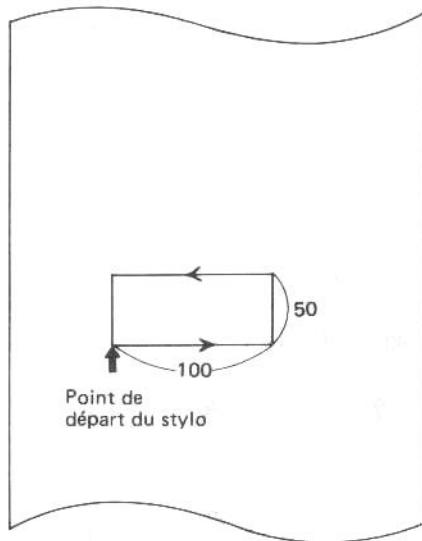
R LINE – (50, 50) – (50, -100) – (-50, -50)



LINE (50, 50) — (100, 100), 2, , B



RLINE — (100, 50) , , , B



VII. LE MODE RESERVE

A. Definition et sélection des touches de réserve

Les six touches à fonctions redéfinissable constituent une caractéristique importante du SHARP PC-1500 en ce qu'elles allègent le travail du programmeur. Ces touches de RESERVE permettent à l'utilisateur de spécifier des phrases ou des mots-clés écrits souvent, phrases et mots que l'on peut alors rappeler en appuyant sur une touche seulement. Les six touches situées dans la rangée supérieure du clavier et marquées des symboles !, ", #, \$, %, et & constituent ces touches de réserve. Il est permis d'affecter jusqu'à trois phrases ou mots-clés à chaque touche ce qui fait un total de 18 possibilités.

Le troisième mode du SHARP, le mode RESERVE est celui qui permet l'affectation de phrases aux touches. Appuyez sur les touches suivantes pour entrer le mode RESERVE:

SHIFT MODE

L'indicateur de mode à la partie supérieure de l'affichage indique maintenant RESERVE. Pour sortir de ce mode RESERVE, il suffit d'appuyer à nouveau sur la touche MODE.

Etant donné que chaque touche de réserve peut stocker jusqu'à trois phrases, il doit y avoir une méthode pour choisir celle que l'utilisateur souhaite rappeler. Cette méthode consiste à utiliser une touche située dans le coin inférieur gauche de l'affichage et marquée du symbole (↓). Cette touche que l'on appelle touche de sélection de réserve fait un choix parmi les phrases mémorisées, qui correspondent actuellement aux touches de réserve. Il est important de prendre bonne note du fait que la touche de sélection de réserve effectue un changement de correspondance pour toutes les touches de réserve à la fois. Cela signifie que la touche de sélection de réserve choisit un groupe de phrases correspondant toutes à une seule touche de réserve. Les chiffres romains (I, II et III) dans la partie supérieure de l'affichage indiquent quel groupe est choisi en ce moment.

Pour affecter une phrase à une touche de réserve, passez sur le mode RESERVE d'abord (en appuyant sur **SHIFT MODE**), puis utilisez la touche de sélection de réserve pour choisir le groupe (I, II ou III) sous lequel vous allez stocker la phrase. Appuyez ensuite sur la touche de réserve appropriée (!, ", #, \$, %, ou &). Un affichage du type suivant apparaîtra à ce moment:

RESERVE •
F 6 :

(Le nombre qui vient après le F, 6 dans l'exemple ci-dessus, représente la touche de réserve sur laquelle on a appuyé.) Quand le symbole de guidage fait son apparition, vous pouvez entrer au clavier la phrase à mémoriser. Comme exercice, écrivez ce qui suit:

R U N 1 0 0 ENTER

Cette phrase est maintenant associée avec la touche de réserve que vous avez choisi.

Faisons l'essai suivant. Appuyez sur la touche MODE pour revenir sur le mode RUN. Appuyez sur la touche de réserve utilisée dans l'exemple précédent. L'affichage indique maintenant la commande mémorisée auparavant:

RESERVE •
RUN 100_

Si vous appuyez sur la touche ENTER, l'ordinateur essaiera de passer un programme à la ligne 100. L'avantage offert par la touche de réserve consiste à ne pas avoir à écrire une commande complète chaque fois que vous désirez la lancer.

Une notation spéciale permise sur le mode RESERVE aurait pu rendre plus aisée l'opération précédente. On peut, en effet, se servir du symbole @ (à) pour représenter. La commande ENTER. Si nous avions affecté la phrase "RUN 100@" à la touche de réserve, l'exécution du programme aurait commencé dès que nous aurions appuyé sur cette touche de réserve après être revenu sur le mode RUN. Pour illustrer cela, entrons les instructions suivantes comme ligne 222:

222 BEEP 5,50 : END

Maintenant passez sur le mode RESERVE et utilisez les touches suivantes pour définir l'une des touches de réserve:

G O T O 2 2 @ ENTER

(Notez qu'il est encore nécessaire d'appuyer sur ENTER pour définir la touche de réserve elle-même.)

Revenez sur le mode RUN et appuyez sur la touche de réserve que vous venez de définir. Vous remarquerez qu'il est superflu maintenant de vous servir de ENTER après avoir appuyé sur la touche de réserve.

En fait, nous aurions pu accomplir ce bruyant exemple en affectant la phrase suivante:

BEEP 5,50@

directement à une touche de réserve. Faites-en l'essai.

B. Identification des touches de réserve

Plus vous utiliserez les touches de réserve, plus il deviendra difficile de vous rappeler quelle fonction vous avez affecté à quelle touche. Pour parer à ce problème, le PC-1500 vous permet de stocker trois chaînes de caractères (une pour chaque groupe de touches de réserve) qui identifient les fonctions des touches. Ces chaînes ressemblent aux remarques du mode PROgramme.

Ces chaînes d'identifications qu'on appelle "gabarits" sont créées sur le mode RESERVE. Passez sur ce mode et choisissez le groupe de touches de réserve approprié à l'aide de la touche de sélection de réserve. Au lieu d'appuyer sur une touche de réserve ensuite, comme vous le feriez ordinairement, écrivez un gabarit et appuyez sur ENTER. Le gabarit sera alors mémorisé en association avec le groupe.

In guise d'exemple, nous allons prétendre que nous avons affecté aux touches de réserve de un à six (dans le group 1) les noms des fonctions trigonométriques (Sinus, Cosinus, Tangente, Arc Cosinus, Arc Sinus et Arc Tangente). Pour nous rappeler quelle touche correspond à quelle fonction nous allons spécifier un gabarit. Pour effectuer cette opération, passons sur le mode RESERVE, en appuyant sur SHIFT MODE, et utilisons la touche de sélection de réserve pour choisir le groupe 1 (le un romain apparaîtra sur l'affichage. Maintenant, écrivez:

"SIN COS TAN ACS ASN ATN "

Touches utilisées:

```
SHIFT " S I N SPACE C O S SPACE  
T A N SPACE A C S SPACE A S N  
SPACE A T N SPACE  
SHIFT " ENTER
```

Vous avez maintenant défini et mémorisé le gabarit.

Revenez sur le mode RUN à l'aide de la touche **MODE**. Pour vous rappeler de la signification des touches de réserve, appuyez simplement sur la touche **RCL** (rappel) et les voilà! Appuyez une deuxième fois sur la touche **RCL** et le symbole de guidage fait sa réapparition sur l'affichage.

Il est possible de créer des gabarits pour chaque groupe de touches de réserve, gabarits, qui peuvent atteindre jusqu'à 26 caractères de longueur. Remarquez que le gabarit vous sert seulement de mémento: les mots et les lettres que vous stockez dans le gabarit n'ont pas de sens pour l'ordinateur.

C. Effacement des programmes de réserve

Comme vous devez déjà le savoir, on remet toutes les mémoires de réserve à zéro en entrant **N E W ENTER**. Remarquez cependant que cette opération doit être effectuée sur le mode RESERVE.

VIII. DEMARRAGE DE L'EXECUTION DU PROGRAMME

A. La touche DEF

La touche DEF (définir en abrégé) permet de prendre des raccourcis qui font gagner du temps.

A.1. Passer des programmes DEFinissables

La touche **DEF** fournit la troisième méthode pour commencer un programme. Comme nous l'avons vu dans la section à propos de la commande RUN, il est possible d'étiqueter un programme avec une lettre. La touche **DEF** sert à amorcer rapidement un programme étiqueté. Cette opération consiste à appuyer d'abord sur la touche **DEF**, puis sur la touche alphabétique qui correspond à l'étiquette du programme. Les seules touches alphabétiques qu'il soit permis d'utiliser de cette manière sont les suivantes:

A, S, D, F, G, H, J, K, L, Z, X, C, V,
B, N, M, SPACE et =

A titre d'exercice, entrez les instructions suivantes pour créer trois programmes étiquetés:

Listage du programme:

```
10 "Z" : GOSUB 500
20 PRINT " Z"
30 END
140 "A" : GOSUB 500
150 PRINT " A"
160 END
270 " " : GOSUB 500
280 PRINT " B"
290 END
500 CLS : PAUSE "VOUS AVEZ ENFONCE LA ";
510 RETURN
```

Essayez maintenant, sur le mode RUN, de commencer chaque programme à l'aide de la touche **DEF**. Remarquez que si vous spécifiez une lettre pour laquelle il n'existe pas de programme étiqueté correspondant, vous obtiendrez un message d'erreur ERROR 11.

A.2. Mots-clé pré-affectés

Un petit nombre de mots-clés fréquemment utilisés ont été affectés chacun, en permanence, à une touche alphabétique située dans la seconde rangée du clavier. Pour retrouver ces mots-clés, sur n'importe quel mode, il suffit d'appuyer sur la touche DEF, puis sur l'une des touches alphabétiques en question. Par exemple, pour retrouver le mot-clé USING, écrivez:

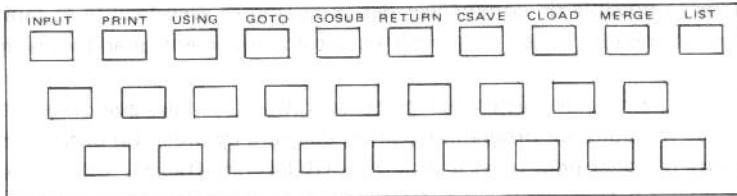
DEF E

Le tableau ci-dessous indique les mots-clés disponibles et leur touche alphabétique correspondante:

Touche alphabétique	Mot-clé
Q	INPUT
W	PRINT
E	USING
R	GOTO
T	GOSUB
Y	RETURN
U	CSAVE
I	CLOAD
O	MERGE
P	LIST

On peut se servir de ces mots-clés quand l'ordinateur est connecté à l'interface imprimante/cassette. (CE-150).

Gabarit



Deux gabarits vous sont fournis avec l'ordinateur. Utilisez-les pour identifier les fonctions affectées aux touches de définition.

A.3. L'instruction AREAD

Il est possible de donner une valeur unique aux programmes étiquetés amorcés à l'aide de la touche DEF chaque fois que l'on passe le programme sans utiliser une instruction INPUT. La lecture de la valeur est effectuée par l'instruction AREAD qui doit suivre l'étiquette du programme sur la même ligne. L'instruction AREAD se présente sous la forme suivante:

AREAD nom de variable

dans laquelle le nom de variable est un nom de variable numérique ou à caractères permis.

Pour passer une valeur à un programme qui incorpore une instruction AREAD, l'utilisateur écrit la valeur, appuie sur la touche **DEF** puis sur la touche alphabétique correspondant à l'étiquette du programme.

Comme exemple, entrez les deux programmes suivants:

Listage du programme:

```
10 "X" : AREAD TM
20 TIME = TM
30 PRINT "HEURE ACTUELLE"; TM
40 END
100 "Z" : AREAD D$
110 PRINT "LA DATE "; D$
120 END
```

Revenez sur le mode RUN et commencez le programme à l'étiquette X en écrivant au clavier le mois, le jour, l'heure et DEF X:

1 2 3 1 0 2 . 4 0 0 DEF X

Ce programme règle l'horloge du système à n'importe quelle heure si elle est spécifiée juste avant l'écriture de DEF (voir la fonction TIME).

Pour commencer le programme à l'étiquette Q, écrivez le jour de la semaine, puis DEF Q:

V E N D R E D I DEF Z

B. Amorçage automatique de programme

Il n'est pas seulement possible de commencer des programmes avec facilité et rapidité à l'aide de la touche DEF, mais encore de les faire démarrer automatiquement quand vous mettez le PC-1500 en marche.

L'instruction ARUN permet de déclencher cette opération. Il faut que cette instruction soit la toute première de la mémoire de programme, sinon elle sera ignorée. En outre plusieurs autres conditions doivent être remplies pour que l'instruction ARUN soit efficace. Il faut que le PC-1500 ait été mis à l'arrêt quand il était sur le mode RUN et qu'aucune erreur ne soit détectée quand on met le PC-1500 en marche.

Le programme suivant se sert de l'instruction ARUN pour saluer l'utilisateur de l'ordinateur:

Listage du programme:

```
10 ARUN
30 CLS
50 BEEP 5,50
70 PRINT "HE, SALUT!"
90 END
```

C. Comparaison des méthodes de démarrage

Quoique les différentes méthodes de démarrage de programme achèvent superficiellement le même résultat, leur opération interne diffère. Dans le but d'exploiter à notre avantage ces différences, il s'avère nécessaire d'examiner la mémorisation des données dans l'ordinateur. Ceci inclut aussi une étude des différentes préparations internes qu'effectue le PC-1500 avant le passage d'un programme.

C.1. L'aire de mémoire fixe

Quoique toutes les variables d'un même type s'utilisent de la même façon, elles ne reçoivent pas le même traitement à l'intérieur. Le PC-1500 comprend une aire de "mémoire fixe" avec un espace de stockage suffisant pour 26 variables numériques et 26 variables de chaînes de caractères (chaînes de 16 caractères). Par conséquent, les variables de A à Z et de A\$ à Z\$ sont affectées en permanence dans cette aire.

Toutes les autres variables, y compris les variables à noms à deux caractères sont affectés dans l'aire de mémoire principale de l'ordinateur. Elles partagent cette aire de mémoire principale avec les directives de programme, quoique les variables soient affectées à l'extrémité opposée aux directives dans la mémoire. Du fait que les directives et les données occupent la même aire, il arrive qu'elles l'encombrent totalement. Dans ce cas un erreur située entre ERROR 117 et error 181 se produit.

Il est important de réaliser que les deux aires de mémoire ne reçoivent pas le même traitement au démarrage d'un programme, comme l'explique le tableau dans la section suivante. FONDAMENTALEMENT, les variables de la mémoire fixe ne peuvent être effacées que par une instruction CL explicite. Celles stockées dans la mémoire principale sont effacées chaque fois que l'on commence un programme à l'aide de la commande RUN.

Une autre particularité de la mémoire fixe consiste en la possibilité de redéfinir les données de cette aire comme des tableaux dont le nom est le symbole @ (à) pour les variables numériques et @\$ pour les variables à chaîne. La désignation @(1) représente donc le même emplacement de stockage que la variable A et @(26) le même emplacement de stockage que la variable Z. La désignation @\$(5) s'applique au même emplacement que E\$ et la désignation @\$(20) au même emplacement que T\$. Pour des raisons évidentes, des indices supérieurs à 26 ne sont pas permis. Remarquez qu'il n'est pas nécessaire de donner une dimension à @ et @\$ avant de les utiliser.

C.2. Tableau de comparaison des méthodes de démarrage de programme

	<u>RUN</u>	<u>GOTO</u>	<u>DEF</u>
Remise à zéro de l'affichage.	Oui	Oui	Non
Le curseur revient à la 1ère colonne	O	N	N
L'intervalle WAIT est réglé à l'infini.	N	N	N
Le mode trace est altéré.	N	N	N
La mémoire fixe est remise à zéro.	N	N	N
La mémoire principale est remise à zéro.	O	N	N
FOR-NEXT, GOSUB La pile intérieure est remise à zéro.	O	O	O
ON ERROR GOTO est annulé.	O	N	N
Le compteur DATA pour l'opération de READ est rétabli.	O	N	N
Le format USING est annulé.	O	N	N

APPENDICES

A. TABLE DES ABREVIATIONS

Commandes Imprimante

COLOR	COL. COLO.	LPRINT	LP. LPR. LPRI. LPRIN.
CSIZE	CSI. CSIZ.	RLINE	RL. RLI. RLIN.
GLCURSOR	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.	ROTATE	RO. ROT. ROTA. ROTAT.
GRAPH	GRAP.	SORGN	SO. SOR. SORG.
LCURSOR	LCU. LCUR. LCURS. LCURSO.	TAB	--
LF	--	TEST	TE. TES.
LINE	LIN.	TEXT	TEX.
LLIST	LL. LLI. LLIS.		

Commandes Cassette

CHAIN	CHA. CHAI.	MERGE	MER. MERG.
CLOAD	CLO. CLOA.	PRINT #	P. # PR. # PRI. #
CLOAD?	CLO.? CLOA.?		PRIN. #
CSAVE	CS. CSA. CSAV.	RMT OFF RMT ON	RM. OF. RM. O.
INPUT #	I. # IN. # INP. # INPU. #		

Instructions

AREAD	A. AR. ARE. AREA		
ARUN	ARU.	GOSUB	GOS. GOSU.
BEEP	B. BE. BEE.	GOTO	G. GO. GOT.
CLEAR	CL. CLE. CLEA.	GPRINT	GP. GPR. GPRI. GPRIN.
CLS	--	GRAD	GR. GRA.
CURSOR	CU. CUR CURS. CURSO.	IF	--
DATA	DA. DAT.	INPUT	I. IN. INP. INPU.
DEGREE	DE. DEG. DEGR. DEGRE.	LET	LE.
DIM	D. DI.	LOCK	LOC.
END	E. EN.	NEXT	N. NE. NEX.
		ON	O.
ERROR	ER. ERR. ERRO.		
FOR	F. FO.		
GCURSOR	GCU. GCUR. GCURS. GCURSO.		

PAUSE	PA. PAU. PAUS.	STEP	STE.
PRINT	P. PR. PRI PRIN.	STOP	S ST. STO.
RADIAN	RAD. RADI. RADIA.	THEN	T. TH. THE.
RANDOM	RA. RAN. RAND. RANDO.	TRON	TR. TRO.
READ	REA.	TROFF	TROF.
		UNLOCK	UN. UNL. UNLO. UNLOC.
REM	--		
RESTORE	RES. REST. RESTO. RESTOR.	USING	U. US. USI. USIN.
RETURN	RE. RET. RETU. RETUR.	WAIT	W. WA. WAI.

Commandes

CONT	C CO. CON.	NEW	--
LIST	L. LI. LIS.	RUN	R. RU.

Fonctions

ABS	AB.	MEM	M. ME.
ACS	AC.	MID\$	MI. MID.
AND	AN.	NOT	NO.
ASC	--	OR	--
ASN	AS.		
ATN	AT.	PI	--
CHR\$	CH. CHR.	POINT	POI. POIN.
COS	--	RIGHT\$	RI. RIG. RIGH. RIGHT.
DEG	--	RND	RN.
DMS	DM.	SGN	SG.
EXP	EX.	SIN	SI.
INKEY\$	INK. INKE. INKEY.	SQR	SQ.
INT	--	STATUS	STA. STAT. STATU.
LEFT\$	LEF. LEFT.	STR\$	STR.
LEN	--	TAN	TA.
LOG	LO.	TIME	TI. TIM.
LN	--	VAL	V. VA.

B. REMPLACEMENT DES PILES DU PC-1500

Vous vous éviterez bien des problèmes si vous prenez les précautions suivantes lors du remplacement des piles:

- * Remplacez toujours les 4 piles en même temps.
- * Ne mélangez pas des piles neuves avec des piles usagées.
- * Utilisez exclusivement 4 piles sèches de type AA, R6 ou SUM-3 de 1,5V.

METHODE DE REMPLACEMENT DES PILES

Il est nécessaire de placer des piles dans l'ordinateur dès sa réception et, par la suite, de les remplacer chaque fois que l'indicateur de charge disparaît de l'affichage. Veuillez procéder de la manière suivante:

1. Appuyez sur la touche **OFF** pour mettre l'ordinateur hors tension.
2. Dévissez le couvercle du boîtier piles en vous servant d'une pièce de monnaie ou d'un petit tournevis (fig. 1).
3. Remplacez les quatre piles (fig. 2).

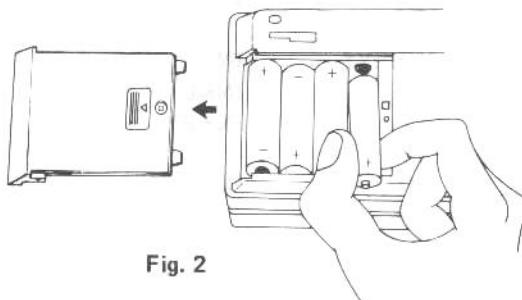
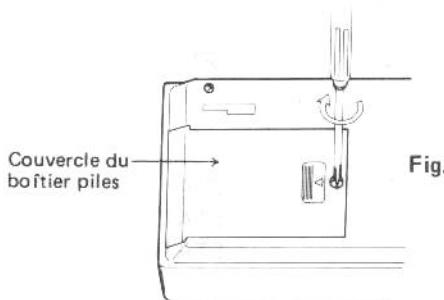
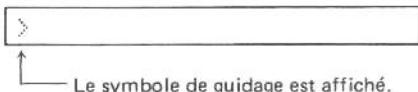


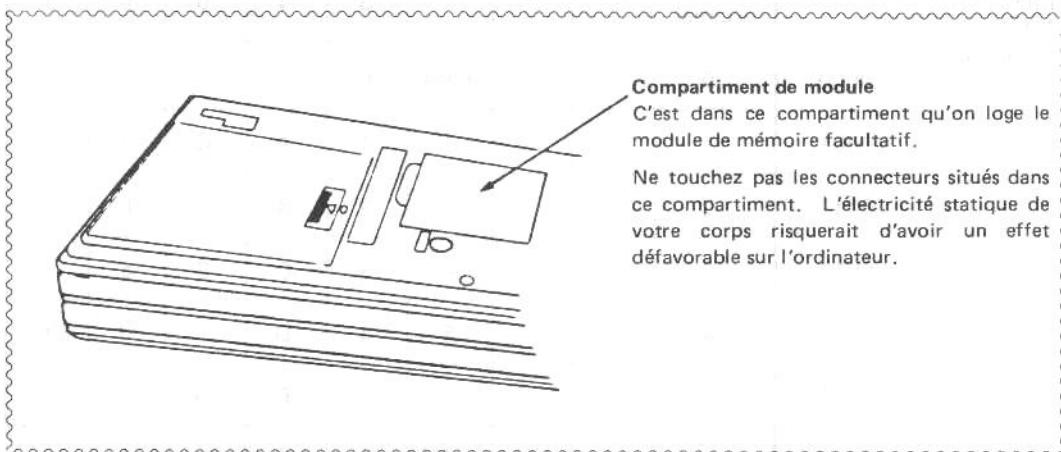
Fig. 2

4. Appuyez légèrement sur le couvercle lorsque vous remettez la vis en place.
5. Pour poursuivre, appuyez sur les touches **ON** et **CL**, écrivez NEW Ø et appuyez sur la touche **ENTER**.
6. Vérifiez l'affichage qui s'ensuit.



Si l'affichage est vide ou un symbole autre que “>” apparaît, déposez à nouveau les piles, puis remettez-les en place et vérifiez l'affichage.

- Note:
- * Otez les piles si vous savez que vous n'allez pas utiliser l'ordinateur pendant longtemps, afin d'éviter des dégâts provoqués par des fuites de liquide des piles.
 - * De même, retirez promptement une pile morte, à cause de la possibilité d'une fuite de liquide de cette pile, fuite qui risquerait d'endommager l'ordinateur.
 - * Il n'est pas possible d'utiliser les piles rechargeables sur le PC-1500.
 - * L'adaptateur c.a. EA-150 du CE-150 peut aussi être utilisé séparément du CE-150.
(Ne connectez pas le EA-150 sur l'ordinateur PC-1500 quand celui-ci est monté sur l'interface imprimante/cassette CE-150).



Compartiment de module

C'est dans ce compartiment qu'on loge le module de mémoire facultatif.

Ne touchez pas les connecteurs situés dans ce compartiment. L'électricité statique de votre corps risquerait d'avoir un effet défavorable sur l'ordinateur.

C. TABLE DES CODES DE CARACTERES ASCII

Positions supérieures →
de bits b7, b6, b5

000 001 010 011 100 101 110 111

↓
Positions
inférieures
de bits
b4, b3, b2, b1

Hexa decimal	0	1	2	3	4	5	6	7
0000 0			SPACE	0	@	P		p
0001 1			!	1	A	Q	a	q
0010 2			"	2	B	R	b	r
0011 3			#	3	C	S	c	s
0100 4			\$	4	D	T	d	t
0101 5			%	5	E	U	e	u
0110 6			&	6	F	V	f	v
0111 7			˘	7	G	W	g	w
1000 8			(8	H	X	h	x
1001 9)	9	I	Y	i	y
1010 A			*	:	J	Z	j	z
1011 B			+	;	K	√	k	{
1100 C			,	<	L	¥	l	:
1101 D			-	=	M	π	m	}
1110 E			.	>	N	^	n	~
1111 F			/	?	O	-	o	█

PC-1500 LISTE DE CODE DES ERREURS

<u>Code des Erreurs</u>	<u>Explication</u>
1.	<p>Erreur de syntaxe: Une erreur résultant d'une faute de frappe comme: d'une donnée manquante:</p> <p style="padding-left: 40px;">10: GOTO</p> <p>de commandes invalides:</p> <p style="padding-left: 40px;">10: 5A = 1 or</p> <p style="padding-left: 40px;">10: NEW</p> <p>(parce que NEW ne peut être une instruction)</p>
	<p>< Affichage > ERROR 1 IN 10</p>
2.	<p>Cette erreur se produit quand il n'y a pas de commande FOR correspondant à une commande NEXT.</p> <p>Ex. 10: FOR A = 1 TO 10 . . . 100: NEXT B</p>
	<p>< Affichage > ERROR 2 IN 100</p>
4.	<p>Cette erreur se produit quand il n'y a pas de DATA correspondant à une commande READ.</p> <p>Ex. 10: READ X, Y 20: DATA 10 30: END</p>
	<p>< Affichage > ERROR 4 IN 10</p>
5.	<p>Ce message est lancé quand une variable de tableau est déclarée sans le nom d'une variable en existence.</p> <p>Ex. 10: DIM A (10, 10) 20: DIM A (5)</p>
	<p>< Affichage > ERROR 5 IN 20</p>
6.	<p>Ce message est lancé quand une variable de tableau a été utilisée sans instruction DIN (dimension).</p> <p>Ex. 10: CLEAR 20: A (3) = 1</p>
	<p>< Affichage > ERROR 6 IN 20</p>

PC-1500 LISTE DE CODE DES ERREURS

Code des Erreurs

Explication

7. Ce message est lancé quand le nom d'une variable est inapproprié.

Ex. 10: A\$ = 10 or

10: FOR A\$ = 1 TO 10

<Affichage> **ERROR 7 IN 10**

8. Ce message est lancé quand la dimension dépasse 3 dans la déclaration d'une variable de tableau.

Ex. 10: DIM A (3, 4, 5, 6)

<Affichage> **ERROR 8 IN 10**

9. Ce message est lancé quand l'indice d'une variable de tableau est plus grand que la taille du tableau énoncée dans la commande DIM.

Ex. 10: DIM A (3)

20: A (4) = 1

<Affichage> **ERROR 9 IN 20**

10. Ce message est lancé quand il n'y a pas suffisamment de mémoire disponible pour créer plus de variables.

Ex. Touches utilisées

Affichage

MEM **ENTER**

7

AB = 10 **ENTER**

ERROR 10

11. Ce message est lancé quand la ligne spécifiée ne se trouve pas dans le programme.

Ex. 10: INPUT "X ="; X: GOTO 5

<Affichage> **ERROR 11 IN 10**

12. Ce message est lancé quand la commande USING spécifie incorrectement le format.

Ex. 100: PRINT USING "# ## A #"; 10

<Affichage> **ERROR 12 IN 100**

13. Ce message est lancé quand un programme dépasse la capacité de la mémoire de programme ou quand la spécification de la touche RESERVE dépasse la capacité de l'aire de réserve.

Ex. Touches utilisées

Affichage

MEM **ENTER**

7

15 A = A + 1 **ENTER**

ERROR 13

PC-1500 LISTE DE CODE DES ERREURS

<u>Code des Erreurs</u>	<u>Explication</u>
14.	(1) Les instructions FOR ont été logées trop profondément et la capacité de l'aire d'empilage a été outrepassée. (2) L'espace-tampon a été outrepassé durant l'analyse d'une expression.
15.	(1) Les instructions GOSUB sont logées trop profondément et la capacité de l'aire a été outrepassée. (2) Durant l'analyse d'une expression, la taille du tampon de chaîne a été dépassé par les chaînes de caractères manipulées.
16.	(1) La valeur spécifiée est supérieure à 10 E100 ou inférieure à -10 E100. Ex. 123E 99 (2) La valeur établie par hexadécimaux dépasse 65535. Ex. &1FFAB
17.	Le type de données (nombres, chaînes de caractères) n'est pas approprié à l'expression du calcul. Ex. 1 + "A" [ENTER]
18.	Le nombre des arguments n'est pas approprié à l'expression. Ex. LEFT\$ ("ABC") [ENTER] SIN (30, 60) [ENTER]
19.	La valeur numérique spécifiée est hors gamme permise. Ex. 10: DIM A (256) <Affichage> ERROR 19 IN 10
20.	Il n'y a pas de '(' suivante '@' ou '@\$' quand les variables de tableau de mémoire fixe sont spécifiées. Ex. 100: @\$ = "A" <Affichage> ERROR 20 IN 100
21.	Une variable est requise dans l'expression. Ex. 10: FOR 1 = 0 TO 10 <Affichage> ERROR 21 IN 10
22.	Ce message est lancé quand on charge le programme et qu'il n'y a pas d'espace de mémoire disponible pour le chargement.

PC-1500 LISTE DE CODE DES ERREURS

Code des Erreurs

Explication

23. Ce message est lancé quand l'heure est incorrectement réglée.
Ex. TIME = 131005.10 **[ENTER]**
26. Ce message est lancé quand la commande ne peut être exécutée sur le mode actuel.
Ex. < RUN MODE > NEW **[ENTER]**
27. Ce message est lancé quand il n'y a pas de programme correspondant à l'étiquette donnée.
Ex.

<u>Touches utilisées</u>	<u>Affichage</u>
DEF I [ENTER]	ERROR 27
28. Ce message est lancé quand une commande ou un indicatif de fonction a été inséré dans " ", ou quand on essaie de substituer des commandes INPUT ou AREAD pour des variables à caractères.
Ex. 10 INPUT A\$

<u>Touches utilisées</u>	<u>Affichage</u>
DEF W [ENTER]	ERROR 28
30. Ce message est lancé quand un numéro de ligne dépasse 65535.
Ex. 102235 A = 10 **[ENTER]**
32. Ce message est lancé quand le curseur graphique se trouve entre les colonnes 152 et 153 durant l'exécution de commandes d'entrée (l'indicatif d'entrée ne peut être affiché).
Ex. 100: GCURSOR 152
110: INPUT X
< Affichage > **ERROR 32 IN 110**

- 177 ~ 181 Durant la fabrication du programme, le programme a débordé dans l'aire des données. Il s'est produit un chevauchement des deux aires.
- 224 ~ 241 Des données d'entrée incorrectes ont été introduites durant l'exécution de commandes INPUT ou AREAD.

Ex. 10: INPUT A

<u>Touches utilisées</u>	<u>Affichage</u>
123 PRINT [ENTER]	ERROR 240

PC-1500 LISTE DE CODE DES ERREURS

<u>Code des Erreurs</u>	<u>Explication</u>
36.	Il n'est pas possible d'afficher des données ou des caractères d'après le format spécifié par les commandes USING. Ex. 10: USING "# ########.##" 20: PRINT 12345 La section du nombre entier en combinaison avec son signe a dépassé les 4 espaces de chiffres allotés.
37.	Ce message d'erreur est lancé quand, dans des calculs numériques, les résultats d'un calcul dépassent 9.999999999 E99.
38.	Ce message d'erreur est lancé quand une division a été effectuée avec 0 pour dénominateur. Ex. 10: PRINT 5/0
39.	Ce message d'erreur est lancé quand un calcul illogique est effectué: * Calcul logarithmic avec nombre négatif Ex. LN (-10) * ASN, ACS dans le cas de X = 1 Ex. ASN (1.5) ACS (100) * Racine carrée de nombres négatifs Ex. SQR (-10)

Erreurs ayant rapport avec l'enregistrement sur cassette

40. Spécifications inappropriée à l'expression.
 41. SAVE et LOAD ont été spécifiés pour l'aire ROM.
 42. Les données de fichier cassette sont trop grandes et ne peuvent être chargées.
 43. Le format des données à charger ne correspond pas au format du fichier lors de la vérification des données à l'aide de la commande CLOAD?.
 44. Une erreur CHECK SUM s'est produite.

Erreurs ayant rapport avec l'imprimante

70. (1) Le stylo est sorti du champ des coordonnées:
 $-2048 \leq X, Y \leq 2047$
(2) Le stylo sortira du champ lors de l'exécution de commandes ultérieures.

PC-1500 LISTE DE CODE DES ERREURS

<u>Code des Erreurs</u>	<u>Explication</u>
71.	(1) Le papier est retourné en arrière de plus de 10,24 cm. (2) Le papier retournera en arrière de plus de 10,24 cm lors de l'exécution de commandes à venir (sur le mode TEXT).
72.	La valeur donnée n'est pas appropriée à la valeur de TAB ou de LCUSROR.
73.	Une commande a été lancée sur le mauvais mode d'impression (GRAPH/TEXT).
74.	Le nombre de virgules (,) dans une commande LINE ou RLINE est trop grand. NOTE: Une entrée de plus de sept virgules produit une erreur. De même, si l'on omet la première virgule, plus de six virgules produiront une erreur.
76.	Concerne LPRINT: l'impression des résultats d'un calcul ne peut pas se faire sur une ligne (sur le mode TEXT).
78.	(1) On est en train de changer les stylos. (2) On n'a pas répondu à l'avertissement "LOW BATTERY" pile affaiblie. (Voir Note 1)
	Ce message est lancé quand pour l'une de ces deux raisons, des commandes visant à faire se déplacer le stylo (comme LPRINT et LINE) ne peuvent être exécutées.
79.	L'indicateur de couleur ne s'est pas allumé (Voir Note 2)
80.	Piles affaiblies. (Voir NOTE 3)

NOTES:

- (1) Si le message ERROR 78 est dû à un état affaibli des piles, mettez l'imprimante à l'arrêt (OFF). Après l'avoir rechargeé, mettez le CE-150 en marche (ON). Vous pouvez continuer à présent.
- (2) L'indicateur de couleur indique COLOR0 (couleur 0) et ne s'allume que lorsque le stylo revient sur le côté gauche. Sur cette position, il est possible d'identifier le numéro du stylo utilisé à ce moment.
- (3) Après la recharge, appuyez immédiatement sur le commutateur ON du CE-150 et continuez les opérations en cours.

F. LECTURES RECOMMANDÉES

Plusieurs éditeurs proposent des ouvrages dont la lecture pourra vous aider à aller plus loin dans le monde de l'informatique:

- SYBEX 18, rue Planchat 75020 — PARIS
- PSI: 41/51, rne Jacquard — B.P 86
 77400 — LAGNY — SUR — MARNE
- MC GRAW-HILL: 28, rue Beaunier 75014 — PARIS

O. ORDRE D'EVALUATION D'EXPRESSIONS

Les calculs sont effectués en accordance avec la hiérarchie suivante: expressions entre parenthèses ayant la plus haute priorité et opérations logiques ayant la moindre. Si deux ou plus de deux expressions de la même priorité se trouvent dans la même expression ou sous-expression, leur évaluation s'accomplit de la gauche à la droite.

- 1) Expressions entre parenthèses
- 2) Récupération des valeurs des variables, TIME, PI, MEM, INKEY\$
- 3) Fonction (SIN, COS, LOG, EXP, etc.)
- 4) Calcul de puissances (Exemple: $2A^3 = 2 * (A^3)$)
- 5) Signe Arithmétique (+, -)
- 6) Multiplication, Division (*, /)
- 7) Addition, Soustraction (+, -)
- 8) Opérateurs de comparaison (<, <=, =, >=, >, >>)
- 9) Opérateurs logiques (AND, OR, NOT)

NOTES:

— Quand on utilise un signe arithmétique et les puissances dans la même expression, la puissance est évaluée avant le signe.

Exemple: 5^4 est évalué à -625 au lieu de 625

— Les calculs entre parenthèses sont évalués d'abord. Dans les expressions à plusieurs "couches" de parenthèses, le calcul commence avec la paire de parenthèses logée le plus au centre et avance de là vers les parenthèses situées le plus à l'extérieur.

Exemple d'évaluation

$$\begin{aligned}
 & 7^2 + 3 * \sqrt{144} / \sqrt{81} + \sin(120 + 150) * -3 \\
 & 7^2 + 3 * \underbrace{\sqrt{144}}_{12} / \underbrace{\sqrt{81}}_9 + \underbrace{\sin(270)}_{-1} * -3 \\
 & \underline{7^2 + 3 * 12 / 9 +} \quad \underline{-1 * -3} \\
 & 49 + 3 * \underbrace{12 / 9}_{36} + \underbrace{-1 * -3}_3 \\
 & 49 + \underline{36 / 9 +} \quad \underline{3} \\
 & 49 + \underline{4 +} \quad \underline{3} \\
 & \underline{53 +} \quad \underline{3} \\
 & \qquad \qquad \qquad 56
 \end{aligned}$$

(GAMME DE CALCUL)

Fonctions	Gamme dynamique
y^x (y^x)	$-1 \times 10^{100} < x \log y < 100$ $\left\{ \begin{array}{l} y = 0, x \leq 0 : \text{ERROR 39} \\ y = 0, x > 0 : 0 \\ y < 0, x \neq \text{entier} : \text{ERROR 39} \end{array} \right.$
SIN x COS x TAN x	DEG: $ x < 1 \times 10^{10}$ RAD: $ x < \frac{\pi}{180} \times 10^{10}$ GRAD: $ x < \frac{10}{9} \times 10^{10}$ Dans TAN x les suivants sont exclus. DEG: $x = 90(2n-1)$ RAD: $x = \frac{\pi}{2}(2n-1)$ GRAD: $x = 100(2n-1)$ (n: entier)
SIN ^{-1}x COS ^{-1}x	$-1 \leq x \leq 1$
TAN ^{-1}x	$ x < 1 \times 10^{100}$
LN x LOG x	$1 \times 10^{-99} \leq x < 1 \times 10^{100}$
EXP x	$-1 \times 10^{100} < x \leq 230.2585092$
\sqrt{x}	$0 \leq x < 1 \times 10^{100}$

Des fonctions autres que celles ci-dessus ne peuvent être calculées que si x reste à l'intérieur de la gamme qui suit:

$1 \times 10^{-99} \leq x < 1 \times 10^{100}$ and 0	
(Ex.) \emptyset^{\emptyset} [ENTER] → ERROR 39	
\emptyset^5 [ENTER] → \emptyset	
$(-4)^{\emptyset.5}$ [ENTER] → ERROR 39	
$-4^{\emptyset.5}$ [ENTER] → -2	

- En règle générale, l'erreur des calculs fonctionnels ±1 au niveau du chiffre le plus bas d'une valeur numérique affichée (au niveau du chiffre le plus bas de la mantisse dans le cas de la notation scientifique) située dans les limites de la gamme de calcul ci-dessus.

X. DIFFÉRENCES ENTRE LES COMMANDES DU PC-1211 ET CELLES DU PC-1500

X.1 Directives communes au PC-1211 et au PC-1500

1. Fonctions

ABS
ACS
ASN
ATN
COS
DEG
DMS
EXP
INT
LOG
LN
 π (PI)
SGN
SIN
 $\sqrt{}$ (Racine carrée)
TAN
 $^{\wedge}$ (Calcul de puissance)

3. Commandes

CONT
LIST
NEW
RUN

4. Commandes cassette

CHAIN
CLOAD
CLOAD?
CSAVE
INPUT #
PRINT #

2. Instructions

AREAD
USING
CLEAR
DEGREE
END
FOR-TO-STEP
GOSUB
GOTO
GRAD
IF
INPUT
LET
MEM
NEXT
PAUSE
PRINT
RADIAN
REM
RETURN
STOP
THEN
USING

X-2 COMMANDES PARTICULIERES AU PC-1500

1. Fonctions

AND
ASC
CHR\$
INKEY\$
LEFT\$
LEN
MID\$
NOT
OR
POINT
RIGHT\$
RND
STATUS
STR\$
TIME
VAL

2. Instructions

ARUN
BEEP (pas sur le PC-1211)
CLS
CURSOR
GCURSOR
GPRINT
DATA
DIM
LOCK
ON ERROR
ON GOSUB
ON GOTO
POINT
RANDOM
READ
RESTORE
TRON
TROFF
UNLOCK
WAIT

3. Commandes

(les mêmes que le PC-1211)

CHAIN
CLOAD
CLOAD?
CSAVE
INPUT#
MERGE
PRINT#
RMT OFF
RMT ON

5. Directives imprimante

COLOR
CSIZE
GCURSOR
GLCURSOR
GPRINT
GRAPH
LCURSOR
LF
LINE
LLIST
LPRINT
RLINE
ROTATE
SORGN
TAB
TEST
TEXT

Z. TABLE DE REFERENCE DES COMMANDES

1. Fonctions

Fonction	Abréviations	Remarques
ABS	AB.	Valeur absolue
ACS	AC.	COS^{-1}
AND	AN.	exp. AND exp. (AND de Boole)
ASC	ASC	Convertit des caractères en code ASCII ASC "caractère" variable de caractères
ASN	AS.	SIN^{-1}
ATN	AT.	TAN^{-1}
CHR\$	CH. CHR.	Convertit des caractères en code ASCII
COS	COS	
DEG		Convertit des degrés, minutes et secondes en degrés décimaux.
DMS	DM.	Convertit des degrés décimaux en degrés, minutes et secondes.
EXP	EX.	e^x
INKEY\$	INK. INKE. INKEY.	variable de caractères = INKEY\$ Si l'on appuie sur une touche pendant l'exécution de la commande INKEY\$, le caractère ASCII sera lu dans la variable de caractères.
INT		Réduit une valeur à un nombre entier: INT (10/3) ENTER <Display> 3
LEFT\$	LEF. LEFT.	LEFT\$ (variable de caractères, expression numérique) Prend le nombre de caractères spécifié du côté gauche de la chaîne de caractères spécifiée.
LEN		LEN "caractère" variable de caractères Donne le nombre de caractères de la chaîne de caractères spécifiée.
LOG	LO.	$\log_{10} X$
LN		$\log_e X$
MEM	M. ME.	Affiche le restant des nombres de pas disponibles dans la mémoire. Identique à STATUS 0.

Fonction	Abréviations	Remarques
MID\$	MI. MID.	MID\$ (variable de caractères, exp1 numérique, exp2 numérique); Prend un (ou des) caractère(s) du milieu de la chaîne de caractères spécifiée.
NOT	NO.	NOT exp. [négation booléenne]
OR		exp. OR exp. [OR booléen]
π (PI)		Spécifie le rapport de la circonférence: (= 3,141592654)
POINT	POI. POIN.	POINT expression numérique Rapporte un nombre correspondant au motif de points activés de la colonne donnée.
RIGHT\$	RI. RIG. RIGH. RIGHT.	RIGHT\$ (variable de caractères, exp. numérique) Prend le nombre de caractères spécifié du coté droit de la chaîne de caractères donnée.
RND	RN.	RND expression Engendre des nombres aléatoires.
SGN	SG.	Fonction de signe
SIN	SI.	Sinus d'une expression.
$\sqrt{}$ (SQR)	SQ.	Racine carrée d'une expression.
STATUS	STA. STAT. STATU.	STATUS 0 ou STATUS 1 (0) Nombre de pas disponibles d'un programme. (1) Nombre de pas accomplis d'un programme.
STR\$	STR.	STR\$ expression numérique. Convertit des valeurs numériques en chaîne de caractères.
TAN	TA.	Tangente d'une expression.
TIME	TI. TIM.	(1) TIME = MMDDHH.MMSS (2) TIME (annonce l'heure et la date présentes) La fonction TIME établit et extrait le mois (MM), le jour (DD), l'heure (HH), la minute (MM) et la seconde (SS).
VAL (value)	V. VA.	VAL { "caractère" variable de caractères. } convertit une chaîne de caractères en valeurs numériques.
$^$		Fonction exponentielle.

2. Instructions

Instruction	Abréviations	Remarques
AREAD (lecture automatique)	A. AR. ARE. AREA.	AREAD variable Lors de l'exécution de programmes au moyen de touches DÉFinies, AREAD introduit le contenu de l'affichage dans la variable spécifiée.
ARUN (passage automatique)	ARU.	ARUN Ordonne le démarrage automatique de l'exécution d'un programme quand le PC-1500 est en marche.
BEEP	B. BE. BEE.	BEEP exp1, exp2, exp3 Commande de signal sonore. Met en marche et arrête les fonctions génératrices de signaux sonores, et spécifie le ton et la durée des dits signaux.
CLEAR	CL. CLE. CLEA.	Commande l'effacement de toutes les valeurs de variables.
CLS (clears)		Efface l'affichage.
CURSOR	CU. CUR. CURS. CURSO.	(1) CUSOR exp. ($0 \leq \text{exp.} \leq 255$) spécification de la position de départ de l'affichage. (2) CURSOR Ramène le curseur à la gauche de l'affichage.
DATA	DA. DAT.	DATA exp, ext, ... Spécifie des données à l'intérieur d'un programme. Utilisé avec une instruction READ.
DEGREE	DE. DEG. DEGR. DEGRE.	Choisit les degrés comme mode angulaire courant. [°]
DIM (dimension)	D. DI.	(1) DIM nom de variable (exp) (2) DIM nom de variable (exp) * exp3 (3) DIM nom de variable (exp1, exp2) Spécifie la taille et les dimensions d'un tableau. (): exp3:
END	E. EN.	Annonce la fin d'un programme.
FOR	F. FO.	(1) FOR variable numérique = exp1 TO exp2 Début de la boucle FOR-NEXT. Utilisée en correspondance avec la commande NEXT.
TO		(2) FOR variable numérique = exp1 TO exp2 STEP exp3
STEP	STE.	exp 1: expression initiale exp 2: valeur finale exp 3: intervalle à augmenter avec chaque boucle.

Instruction	Abréviations	Remarques
GCURSOR (graphic cursor)	GCU. GCUR. GCURS. GCURSO.	Spécifie la position du curseur graphique.
		GCURSOR expression ($0 \leq \text{expression} \leq 155$) ou (&0 $\leq \text{expression} \leq & 9B$)
GOSUB	GOS. GOSU.	GOSUB { expression "caractère" variable de caractère. } Démarre l'exécution d'un sous-programme à la ligne ou à l'étiquette spécifiée.
GOTO	G. GO. GOT.	GOTO { expression "caractère" variable de caractère. } Transfère le contrôle à la ligne ou à l'étiquette spécifiée
GPRINT (impression de graphisme)	GP. GPR. GPRI. GPRIN.	Manoeuvre l'affichage LCD pour les prophismes.
		(1) GPRINT "OO OO OO ..." (" " constituent des nombres hexadécimaux) (2) GPRINT O; O; ... (3) GPRINT & O; & O; ...
GRAD	GR. GRA.	Choisit les grades comme mode angulaire courant. ([⁹])
IF		(1) IF expression conditionnelle (THEN) expression (2) IF expression arithmétique (THEN) expression Evalue les conditions données et soit déplace l'exécution à la ligne suivante soit exécute l'expression. THEN est facultatif.
INPUT	I. IN. INP. INPU.	(1) INPUT nom de variable (2) INPUT "chaîne", variable, "chaîne", variable (3) INPUT "chaîne"; variable, "chaîne"; variable
LET	LE.	(1) LET variable numérique = expression (2) LET variable à caractères = "caractères" (3) LET variable à caractères = variable à caractères LET est facultatif, excepté à l'intérieur de commandes IF.
LOCK	LOC.	Bloque l'ordinateur sur le mode actuel.
NEXT	N. NE. NEX.	NEXT nom de variable Signale la fin de la boucle FOR-NEXT. Le nom de variable doit être le même que le nom de variable de l'instruction FOR.

Instruction	Abréviations	Remarques
ON ERROR	O. ER. ERR. ERRO.	ON ERROR GOTO expression Instruction détectrice d'erreurs
ON GOSUB	O. GOS. GOSU.	ON expression GOSUB exp1, exp2, exp3... Forme automatisée de l'instruction GOSUB. Démarrer l'exécution de l'un des sous-programme par exp1, exp2, exp3, etc. selon la valeur de la variable donnée.
ON GOTO	O. G. GO. GOT.	ON expression GOTO exp1, exp2, exp3 Forme automatisée de l'instruction GOSUB. Démarrer l'exécution de l'un des sous-programmes spécifiés par exp1, exp2, exp3, etc., selon la valeur de la variable donnée.
PAUSE	PA. PAU. PAUS.	Identique à la commande PRINT. Affiche l'information spécifiée pendant environ 0,85 secondes, puis continue le programme.
POINT	POI. POIN.	POINT expression (0 <= expression <= 155) (&0 <= expression <= &9B) Ramène un nombre représentant le motif de points activés dans la colonne donnée de l'affichage.
PRINT	P. PR. PRI. PRIN.	(1) PRINT { expression "caractère" variable de caractère. } ; (2) PRINT { expression "caractère" variable de caractère } , { expression "caractère" variable de caractère } (3) PRINT { expression "caractère" variable de caractère } ; { expression "caractère" variable de caractère } ; ; { expression "caractère" variable de caractère } ;
RADIAN	RAD. RADI. RADIA.	Choisit les radians comme mode angulaire courant.
RANDOM	RA. RAN. RAND. RANDO.	Change la semence utilisée par RND pour engendrer des nombres aléatoires.
READ	REA.	READ variable, variable2, ... etc. Entre des données des instructions DATA dans des variables spécifiées.

Instruction	Abréviations	Remarques
REM (remarque)		REM . . . information explicative . . .
		Spécifie des remarques à l'intérieur d'un programme.
RESTORE	RES. REST. RESTO.	(1) RESTORE expression Change l'ordre des données entrées par l'instruction READ. (2) RESTORE Lit les données à partir de la première instruction DATA.
RETURN	RE. RET. RETU. RETUR.	Continue l'exécution à l'instruction après l'instruction GOSUB qui a appelé le présent sous-programme.
STOP	S. ST. STO.	Commande l'arrêt de l'exécution d'un programme.
THEN	T. TH. THE.	Utilisé dans une instruction IF pour séparer une expression du test conditionnel qui détermine si l'expression sera exécutée ou non. THEN { expression "caractère" variable de caractère }
TRON (trace on)	TR. TRO.	Déclenche le mécanisme de mise au point.
TROFF (trace off)	TROF.	Arrête le mécanisme de mise au point.
UNLOCK	UN. UNL. UNLO. UNLOC.	Annule LOCK pour permettre à l'utilisateur de changer de mode.
USING	U. US. USI. USIN.	(1) USING "# # #, # # # ^" (2) USING "& & & & & &" (3) PRINT USING "Format"; (4) USING (5) PRINT USING; Spécifie le format d'informations affichées par un programme.
WAIT	W. WA. WAI.	WAIT expression ($0 \leq \text{expression} \leq 65535$) Spécifie la durée d'affichage d'une information lors de l'utilisation des commandes PRINT. Une instruction WAIT non accompagnée d'un argument annule la spécification antérieure (durée = infini).

3. Commandes

Instruction	Abréviations	Remarques
CONT (continue)	C. CO. CON.	Redémarre l'exécution d'un programme arrêtée temporairement. Efficace sur le mode RUN.
GOTO	GO. GOT.	Démarre l'exécution d'un programme à la ligne spécifiée de ce programme.
LIST	L. LI. LIS.	Effectue le listage des programmes. Efficace sur le mode PRO.
NEW		(1) NEW (2) NEW 0 Sur le mode PRO, NEW effectue la remise à zéro des variables du programme.
RUN	R. RU.	(1) RUN (2) RUN expression (3) RUN "caractère" variable de caractères Efficace sur le mode RUN.

4. Commandes Cassette

Instruction	Abréviations	Remarques
CHAIN	CHA. CHAI.	Entre un programme enregistré sur bande magnétique et exécute ce programme. (1) CHAIN "nom de fichier" (CHAIN-1 "nom de fichier") (2) CHAIN "nom de fichier", expression (CHAIN-1 "nom de fichier", expression) (3) Formes abrégées de noms de fichier de fichier de (1) et (2).
CLOAD (chargement de cassette)	CLO. CLOA.	Enregistre le contenu de la mémoire Programme ou Réserve sur bande magnétique. (1) CLOAD (CLOAD-1) (2) CLOAD "nom de fichier" (CLOAD-1 "nom de fichier")

Instruction	Abréviations	Remarques
CLAOD? (chargement de cassette)	CLO.? CLOA.?	Commande de comparaison. (Cette commande compare un programme mémorisé avec un programme enregistré sur bande magnétique. Peut aussi servir à comparer le contenu de la mémoire de réserve. (1) CLOAD? (CLOAD?-1) (2) CLOAD? "nom de fichier" (CLOAD?-1 "nom de fichier")
CSAVE (enregistrement sur cassette)	CS. CSA. CSAV.	Cette commande fait enregistrer sur bande le contenu des mémoires de programme et de réserve. (1) CASVE (CSAVE-1) (2) CSAVE "nom de fichier" (CSAVE-1 "nom de fichier")
INPUT#	I.# IN.# INP.# INPU.#	Cette commande transfère les données enregistrées sur bande dans les variables spécifiées. Sa forme est la même que celle de la commande PRINT#.
MERGE	MER. MERG.	Cette commande extrait des programmes enregistrés auparavant sur bande magnétique. A la différence de CLOAD, le programme extrait n'est pas inscrit au-dessus du programme logé dans la mémoire.
PRINT#	P.# PR.# PRI.# PRIN.#	Commande d'enregistrement de données. Cette commande enregistre sur bande les données dans le PC-1500. (1) PRINT# nom de variable, nom de variable . . . (2) PRINT# "nom de fichier", nom de variable, . . . (PRINT#-1 "nom de fichier", nom de variable, . . .)
RMT OFF	RM.OF. RMTOF.	
RMT ON	RM.O. RMTO.	

5. Commandes de l'imprimante

Instruction	Abréviations	Remarques
COLOR	COL. COLO.	Spécifie la couleur de l'impression COLOR expression COLOR expression ($0 \leq \text{expression} \leq 3$)
CSIZE (taille des caractères)	CSI. CSIZ.	Spécifie la taille des caractères imprimés. CSIZE expression ($1 \leq \text{expression} \leq 9$)

Instruction	Abréviations	Remarques
GLCURSOR (graphic cursor)	GL. GLC. GLCU. GLCURS. GLCURSO.	Déplace le stylo du point de départ à un point coordonné X-Y. Valide sur le mode GRAPH uniquement. GLCURSOR (exp1, exp2)
GRAPH (graphisme)	GRAP.	Etablit le mode sur lequel s'effectue le dessin des graphes et des illustrations.
LCURSOR (curseur de ligne)	LCU. LCUR. LCURS. LCURSO.	Déplace le stylo de l'imprimante sur l'une des positions de caractère disponibles.
LF (Alimentation lignes)		Déroule le papier du nombre de lignes indiqué par l'expression. LF expression
LINE	LIN.	Commandes de tracé de ligne (1) LINE (exp1, exp2)—(exp3, exp4) (2) LINE (exp1, exp2)—(exp3, exp4), exp5, exp6 (3) LINE (exp1, exp2)—(exp3, exp4), exp5, exp6, B exp5: spécifie le type de ligne (continue, discontinue) exp6: spécifie la couleur de la ligne. B: (4) LINE (exp1, exp2)—(exp3, exp4)—... ... (exp11, exp12)
LLIST	LL. LLI. LLIS.	Liste une ou plusieurs lignes du programme courant.
LPRINT	LP. LPR. LPRI. LPRIN.	Fait imprimer des informations spécifiées. Valide sur le mode texte uniquement. Sa forme est la même que celle des commandes PRINT.
RLINE (ligne relative)	RL. RLI. RLIN.	Trace des lignes avec des coordonnées à l'emplacement actuel du stylo. Efficace sur le mode GRAPH seulement. Sa forme est la même que celles des commandes LINE.

Instruction	Abréviations	Remarques
ROTATE (pivoter)	RO. ROT. ROTA. ROTAT.	Spécifie le sens dans lequel les caractères seront imprimés (sens d'impression). ROTATE expression (0 <= expression <= 3)
SORGN (fixation du point d'origine)	SO. SOR. SORG.	Cette commande établit l'emplacement actuel du stylo comme le nouveau point de départ (point d'origine) du stylo. Valide sur le mode GRAPH seulement.
TAB		Identique à LCURSOR, mais peut être utilisé dans des instructions LPRINT: (1) LPRINT TAB Expression; ... (2) TAB expression;
TEST	TE. TES.	Teste l'imprimante. Trace un carré de 5 mm de côté dans chaque couleur.
TEXT	TEX.	Spécification du mode TEXTE, sur lequel s'effectue l'impression des caractères et des nombres.

SHARP CORPORATION

OSAKA, JAPAN

SHARP

POCKET COMPUTER **PC-1500**
TECHNICAL REFERENCE MANUAL

SHARP CORPORATION

**WWW.
PC-1500
.INFO**



FOREWORD

Since the release of the PC-1500 on market, we have had great number of questions from users regarding the machine language of the PC-1500.

To meet with such demand from ardent users, we are now sending this text for study of the machine language of the Sharp's original design LH5801 Microprocessor and LH5811 Peripheral Control LSI in concern with the PC-1500 system. Because the text is edited on the basis of user questions, it may not support quality as a guidebook. In such an event, you are suggested to make reference to microprocessor guidebooks published on market, in addition to this text.

Your opinions and questions are welcome through our products distributor.

NOTE: Machine language program, which controls hardware directly, gives you more various functions than BASIC programs. However, you should check your machine language program enough to make no error before executing it because single wrong key operation may upset the program or occasionally make the machine break down.

Sharp Corporation assumes no liability or responsibility of any kind arising from the use of programs or program materials or any part thereof.

SPECIAL NOTICE TO PC-1500A CUSTOMERS

Because the PC-1500A provides more RAM than the PC-1500, some of the descriptions of the CHIP SELECT SIGNAL, MEMORY MAP, and 40-PIN CONNECTOR (for memory modules) require modifications to suit the PC-1500A.

A summary of the differences between the PC-1500A and the PC1500 is given in the addendum on the page 161.

Take notice of these differences when using machine language.

CONTENTS

1. Machine Language	1
2. LH5801 Microprocessor	5
2-1. Outline of LH5801	6
2-2. Internal structure	7
2-2-1. Block diagram.....	7
2-2-2. Internal registers	8
2-2-3. Status flags	8
2-2-4. CPU pin description	9
2-3. Functions	14
2-3-1. Timer	14
2-3-2. Interrupts	17
2-3-3. Reset	20
2-3-4. CPU system sequence	21
2-3-5. BF flipflop	22
2-3-6. WAIT function	23
2-4. LH5801 instructions	24
2-4-1. Outline	24
2-4-2. Add, subtract, and logical instructions	25
2-4-3. Compare and bit test	31
2-4-4. Transfer and search instructions	33
2-4-5. Block transfer and search instructions	38
2-4-6. Rotate and shift instructions	39
2-4-7. CPU control instructions	42
2-4-8. Jump instructions	45
2-4-9. Subroutine jump instructions	49
2-4-10. Return instructions	53
2-5. Command list	54
2-6. Electrical characteristics and timings	63
3. LH5810/LH5811 I/O port controller	67
3-1. Outline	68
3-2. Functions	68
3-3. Internal structure	70
3-3-1. Block diagram.....	70
3-3-2. Internal registers	70
3-3-3. Pin description	73
3-4. Functions	74
3-4-1. Operation	74
3-4-2. Wait control	75
3-4-3. Serial data input	76
3-4-4. Reset	77
3-5. Specification	78
3-5-1. I/O port controller input/output circuits	78
3-5-2. Pad layout and structure	79
3-5-3. Electrical characteristics	79

4. PC-1500 hardware description	85
4-1. PC-1500 system configuration	86
4-1-1. Outline	86
4-1-2. Block diagram.....	87
· 4-1-3. Power supplies (PC-1500, CE-150, CE-158, CE-159)	87
4-2. PC-1500.....	89
4-2-1. Outline	89
4-2-2. Block diagram.....	90
4-2-3. Chip select circuit	91
4-2-4. PC-1500 system memory map	94
4-3. Connector signals/LSI signals	102
4-3-1. 40-pin connector	102
4-3-2. 60-pin connector	103
4-3-3. LH5801 Microprocessor.....	104
4-3-4. I/O PC	105
4-4. Key matrix and key code chart	109
5. PC-1500 software	111
5-1. BASIC command related PC-1500 machine language	112
5-1-1. NEW.....	112
5-1-2. STATUS	112
5-1-3. PEEK	113
5-1-4. POKE.....	113
5-1-5. CALL	114
5-1-6. CSAVE M	114
5-1-7. CLOAD M.....	114
5-2. Internal code chart	115
5-3. Expression of variable and program	116
5-3-1. Expression of decimal number	116
5-3-2. Expression of binary number.....	116
5-3-3. Expression of character string.....	116
5-3-4. Structure of variable name	117
5-3-5. Structure of program	117
5-3-6. Structure of reserve area.....	118
5-4. System subroutines	120
5-4-1. Character functions	122
5-4-2. Arithmetic subroutines.....	127
5-4-3. Comparison	128
5-4-4. Search	129
5-4-5. Display	131
5-4-6. Printer	135
5-4-7. Cassette tape	138
5-4-8. Caution for system subroutine call	141
6. Machine language programming examples	143
6-1. Binary to hexadecimal conversion	144
6-2. Display inversion	145
6-3. Single display dot left shift	146
6-4. Single display dot right shift.....	147
6-5. Conversion of USING format expressed numerical data into character string.....	148
6-6. Power off that does not activate the printer upon power on	148

[REFERENCE]

1. Determining printing character size and direction.....	150
2. Restoration of array and two-character variable.....	150
3. Knowing the use of CE-150.....	150
4. CMT format	151
5. Circuit diagram	153
5-1. PC-1500.....	154
5-2. CE-150.....	155
5-3. CE-151.....	156
5-4. CE-153.....	156
5-5. CE-155.....	157
5-6. CE-158.....	158
5-7. CE-159.....	159
[ADDENDUM]	
Differences between the PC-1500A and the PC-1500.....	161

1

Machine Language

1. Machine language

There are many program languages for each purpose. PC-1500, for example, is designed to carry out both BASIC and machine language. BASIC is easy to use, however, execution speed is slow. On the other hand, machine language is difficult to understand but execution speed is fast.

Usually, machine language program would be written with the assemble language, which consists of mnemonic codes, and then the assemble language will be translated into machine language.

[EXAMPLE] DISPLAY REVERSE PROGRAM

1. Prepare the program with assemble language consisting of mnemonic codes.

LDI	UH,78H	prepare for assignment of the first display buffer address.
LDI	UL,4DH	
DEC	UH	advance the address
LDA	U	take data in the accumulator
EAI	FFH	take the complement
STA	U	return data into memory
LOP	06H	
CPI	UH,77H	make the loop
BCS	-0EH	
RTN		return to BASIC

2. Translate the above program into machine language. The assembler translates the assemble language into the machine language automatically according to the list. However, a short program can be translated manually. (hand assemble)

The above program can be translated as follows:

68	78	6A	4D	FD	62	25	BD
FF	2E	88	06	6C	77	93	0E
9A							

3. After the completion of the machine language program, write it in PC-1500 by using POKE instruction. And execute the program together with BASIC by CALL instruction.

Execute the following program. You can run the BASIC program with [DEF] [A] and the machine language program with [DEF] [B]. Display reverse in machine language program is faster than that of BASIC. You would know how functional the machine language program is.

EX. Write the following program after executing NEW &4100 [ENTER].

```
10 "A" WAIT 0
20 PRINT" sharp pocket computer"
30 FOR A=0 TO 155
40 GCURSOR A
50 GPRINT 255-POINT A
60 NEXT A
70 GOTO 30
80 END
100 "B" WAIT 0
110 PRINT "sharp pocket computer"
120 POKE &40C5,&68,&78,&6A,&4D,
&FD,&62,&25,&BD,&FF,&2E,&88,&06
130 POKE &40D1,&6C,&77,&93,&0E,&9A
140 CALL &40C5
150 WAIT 20:PRINT:GOTO 140
160 END
```

This manual is divided into three major sections; description of LSI (pp. 5~84: LH5801 Microprocessor & LH5810/LH5811 I/O port controller), PC-1500 hardware description (pp. 85~109), and PC-1500 software description (pp. 111~141).
If you want to know about PC-1500 system first, read from p. 86.



2

LH5801 Microprocessor

2-1. Outline of LH5801

The LH5801 Microprocessor is a CMOS static 8-bit microprocessor that features low power dissipation performance inherent to CMOS LSI and large capacity data processing. Not only that, it enables to to configure a variety of systems with a few additional chips because such as the LCD backplate signal generator, input port, external latch clock, and timer are built in the LH5801.

Features of the LH5801

- ① 8-bit parallel data processing
- ② Direct accessing of 128K bytes
- ③ Use of a 6-byte general purpose register, in addition to the accumulator, allows to comprise three pairs of 2-byte date pointers.
- ④ 9-bit timer capability
- ⑤ Three kinds of interrupts
 - Non-maskable interrupt
 - Maskable interrupt
 - Timer interrupt
- ⑥ 82 instruction set
- ⑦ WAIT function (memory access control possible)
- ⑧ Clock P_φ for input port (8-bit) and external latch
- ⑨ Memory backup function (BFI, BFO)
- ⑩ LCD backplate signal control
- ⑪ External crystal connection for clock generation
- ⑫ Reducing program steps by means of 28-kind single step vector subroutine jump

2-2. Internal Structure

2-2-1. Block diagram

Fig. 1-1 Block diagram of LH5801

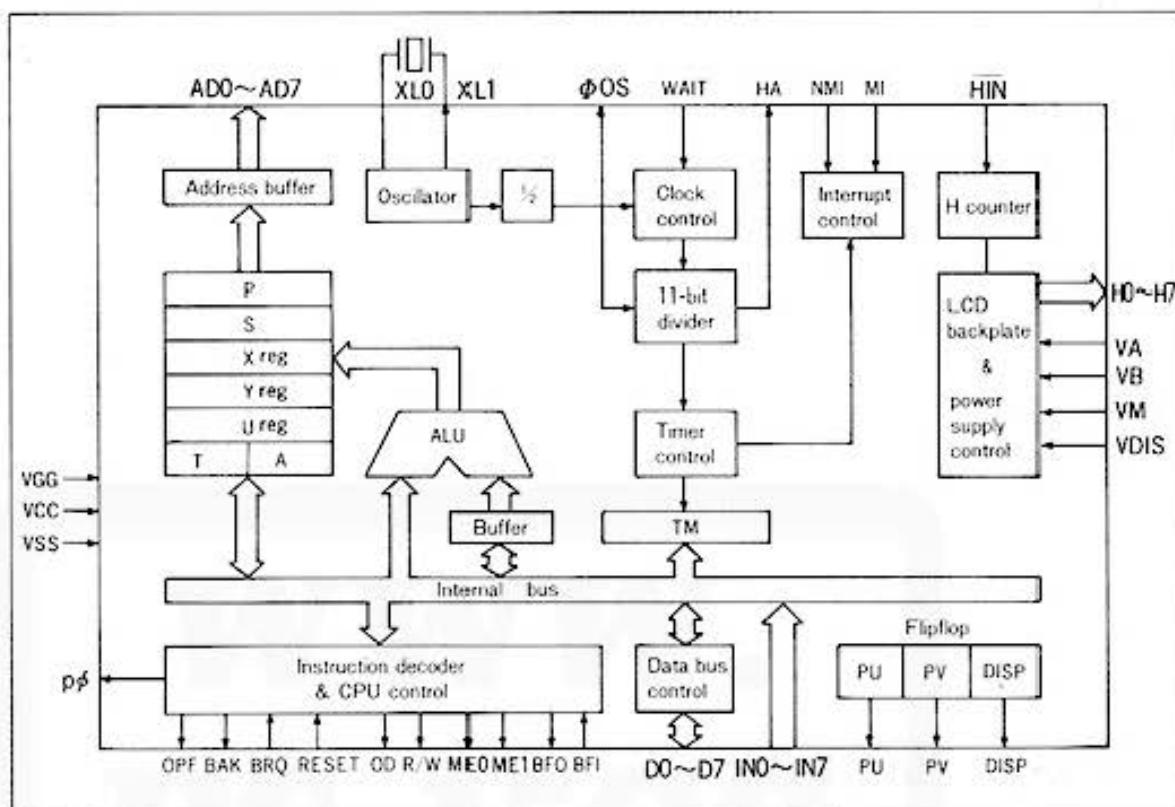
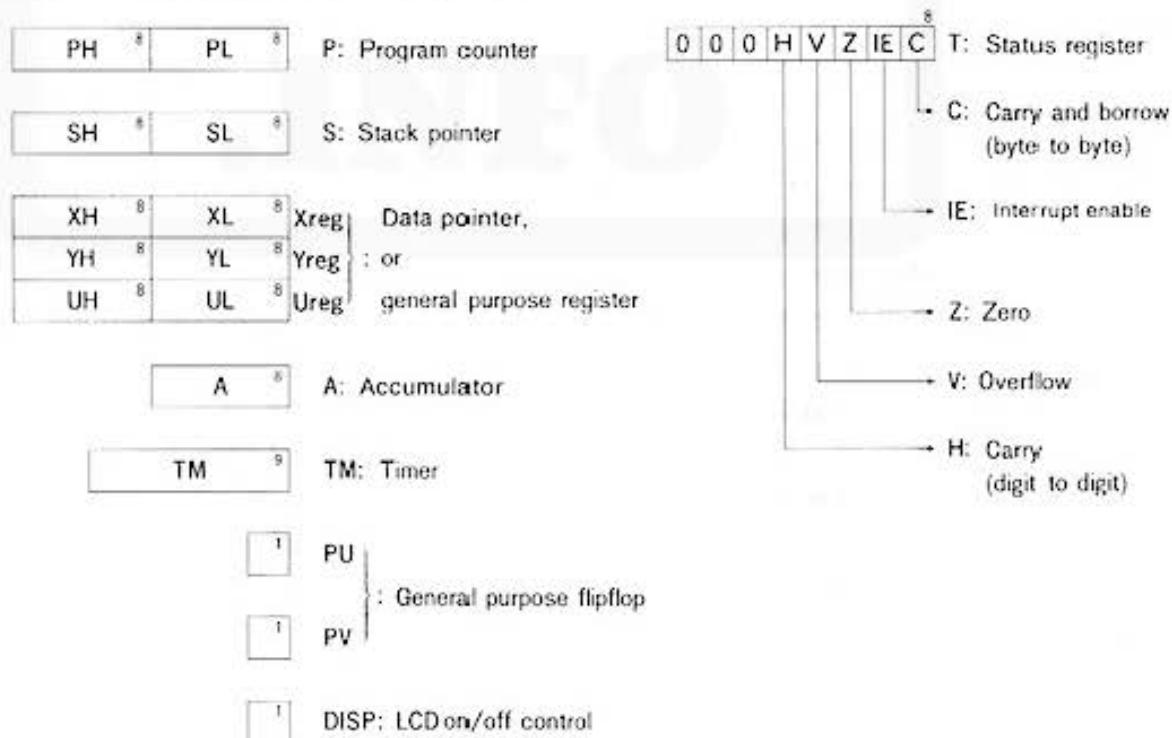


Fig. 1-2 Internal register & flipflops



2-2-2. Internal registers

Symbol	Name	Bit size	Description
P	Program counter	16	Indicates the address next to the address the CPU is now in execution. It will be incremented by "1" when the next instruction is fetched.
S	Stack pointer	16	Indicates the stack address.
A	Accumulator	8	Used for retention of operational result or for data transfer with the external memory.
Xreg	Data register	XL	XL, XH, YL, YH, UL, UH comprise independent 8-bit registers.
		XH	Also, used as 16-bit data pointers, Xreg, Yreg, and Ureg, when used in a pair.
Yreg		YL	
		YH	
Ureg		UL	
		UH	
TM	Timer counter	9	When "0" is set to the TM, it stops the counter action. When anything other than "0" is set, it puts the counter into action. When the TM turns full of "1" with the interrupt enable signal IE on, CPU executes an interrupt processing.
PU		1	General purpose flip-flop.
PV		1	
DISP		1	LCD on/off control.
T	Status register	8	Low order 5 bits represent one of five status of operational result.

2-2-3. Status flags

Status flags, C, V, H, Z, IE are contained in the 8-bit status register. In general, flags other than IE change their state after the execution of arithmetical instruction.

Status register T =

0	0	0	H	V	Z	IE	C
---	---	---	---	---	---	----	---

① Carry flag C

Carry flag C is set when there is a carry from the MSB and reset when there is no carry. For SUBTRACT, the flag is set when there is no borrow or reset when there is.

② Half carry flag H

Half carry flag H is set when there is a carry from the bit position "3" (digit-to-digit carry) and reset when there is no carry.

③ Zero flag Z

Zero flag Z is set when the operational result is zero and reset when not.

④ Overflow flag V

Overflow flag V is set or reset depending on the operational result of "C6⊕C7"; where, the carry from the bit position 6 of a single byte data is assumed to be C6 and the carry from the bit position 7 to be C7.

Single byte data

--	--	--	--	--	--	--	--

B7 B6 B5 B4 B3 B2 B1 B0

2-2-4. CPU pin description

	NC	AD15	AD14	AD13	AD12	AD11	AD10	AD9	VGG	AD8	GND	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
R/W	58																38	D7	
PΦ	59																37	D6	
PV	60																36	D5	
PU	61																35	D4	
ΦOS	62																34	D3	
XL0	63																33	D2	
XL1	64																32	D1	
WAIT	65																31	D0	
IN7	66																30	ME0	
IN6	67																29	ME1	
IN5	68																28	OD	
IN4	69																27	H0	
IN3	70																26	H1	
IN2	71																25	H2	
IN1	72																24	H3	
IN0	73																23	H4	
NC	74																22	H5	
NC	75																21	H6	
NC	76																20	H7	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
RESET	NC	BRQ	BFI	VGG	BFO	OPF	BAK	VCC	VGG	VM	VDIS	VA	VB	NMI	MI	HN	HA	DIS	

① XL0, XL1

These are external crystal connection pins. XL0 is the input pin and XL1 is the output pin. Clock frequency is divided by two inside the CPU. When the 2.6MHz crystal is connected, the CPU operates under 1.3MHz of internal machine cycle.

② AD0~AD15

Address bus. Outputs from these pins are 3-state (three output states of high, low and high impedance) and go high impedance with BRQ (Bus ReQuest). Basically, 64K bytes of memory area is supported for direct accessing, but it is made possible to access even 128K area of memory area when ME0 and ME1 are used.

③ D0~D7

Bidirectional data bus which is used to write or data to/from the external memory.

④ ME0, ME1

Memory enable signals. As memory area of 64K bytes is accessed by ME0 and another 64K bytes by ME1, it permits direct access of memory area of 128K bytes in total. Since ME0 is used for instruction fetch and stack operation, accessing by means of the program counter P and stack pointer S is limited to a maximum of 64K bytes.

As for data accessing, both memory areas covered by ME0 and ME1 can be controlled by a CPU instruction.

⑤ ϕ OS

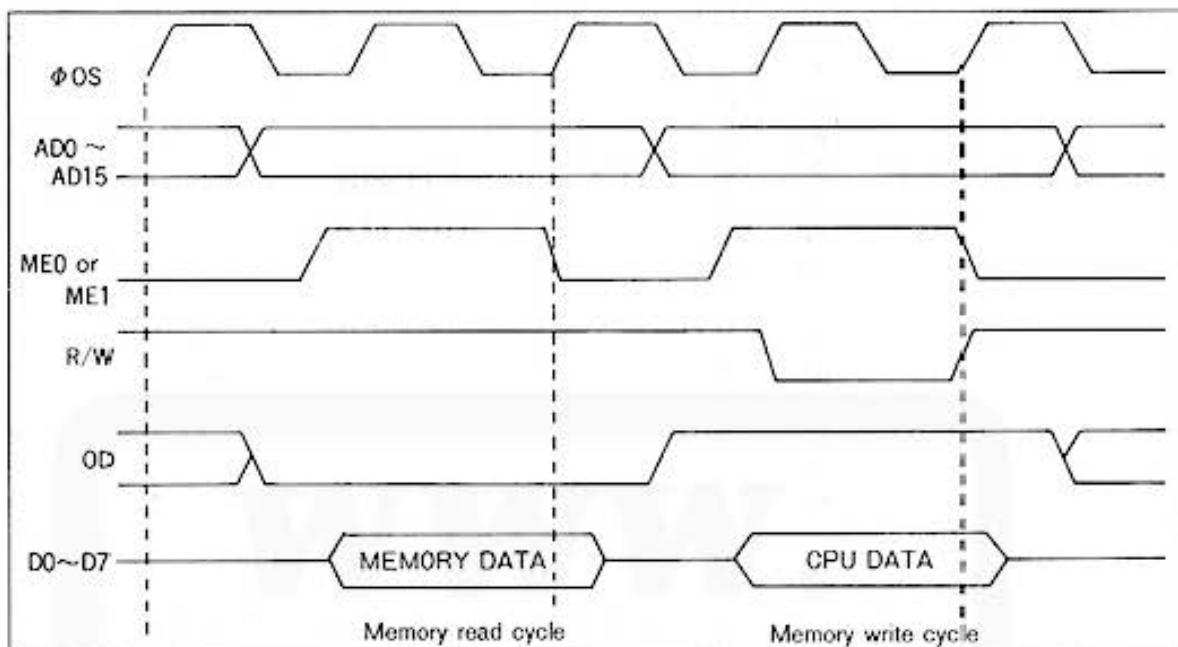
Through this line can be supplied the clock to an external system since the clock in the same phase as the CPU internal basic clock is on this line. Connection of the 2.6MHz crystal oscillator to XL0 and XL1 will supply the clock of 1.3MHz.

⑥ R/W

Memory write signal. A low on this line causes the CPU to send data on the data bus.

⑦ OD

Output disable signal. A high on this line causes the CPU to prohibit data output to the external device. It is used in writing data to the memory.

**⑧ RESET**

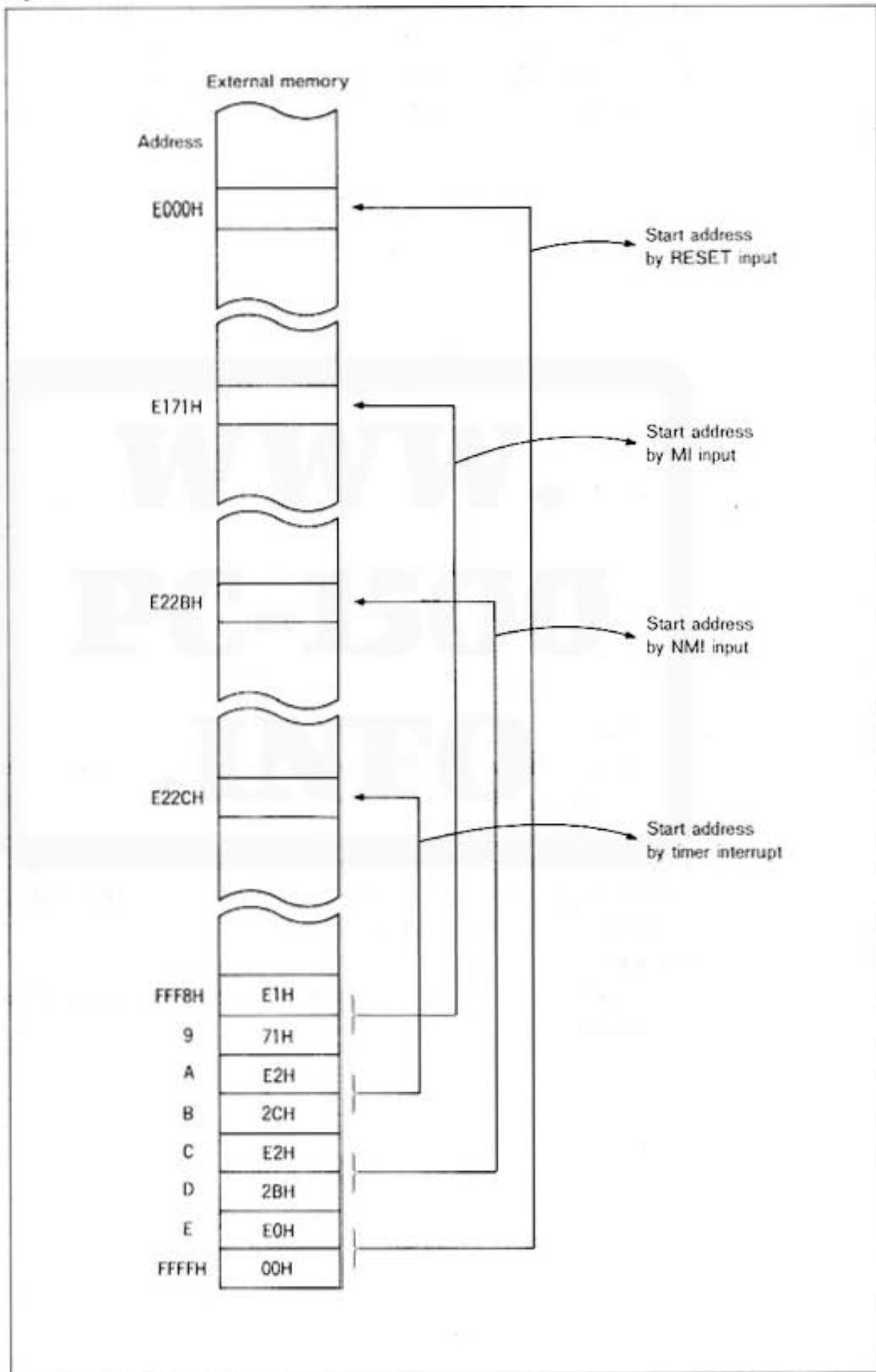
CPU reset input. High state of this signal resets the CPU and the contents of the address FFFEH is set to the register PH and the contents of the address FFFFH to the register PL. When it turns from high to low level, it starts program execution from the address of the program counter.

⑨ NMI

Non-maskable interrupt input. High state of this signal causes interrupt to the CPU, to which the CPU unconditionally responds and starts the interrupt processing routine of which high order byte address is represented by the address FFFCH and low order byte address by FFFDH.

⑩ MI

Maskable interrupt input. When the interrupt enable flag IE is active, a high on the pin M1 requests interrupt, to which the CPU starts to execute the interrupt processing routine whose high order byte address is represented by the address FFF8H and low order byte address by FFF9H.

How instruction execution address is determined against the reset and interrupt input


(11) BRQ

Bus request signal.

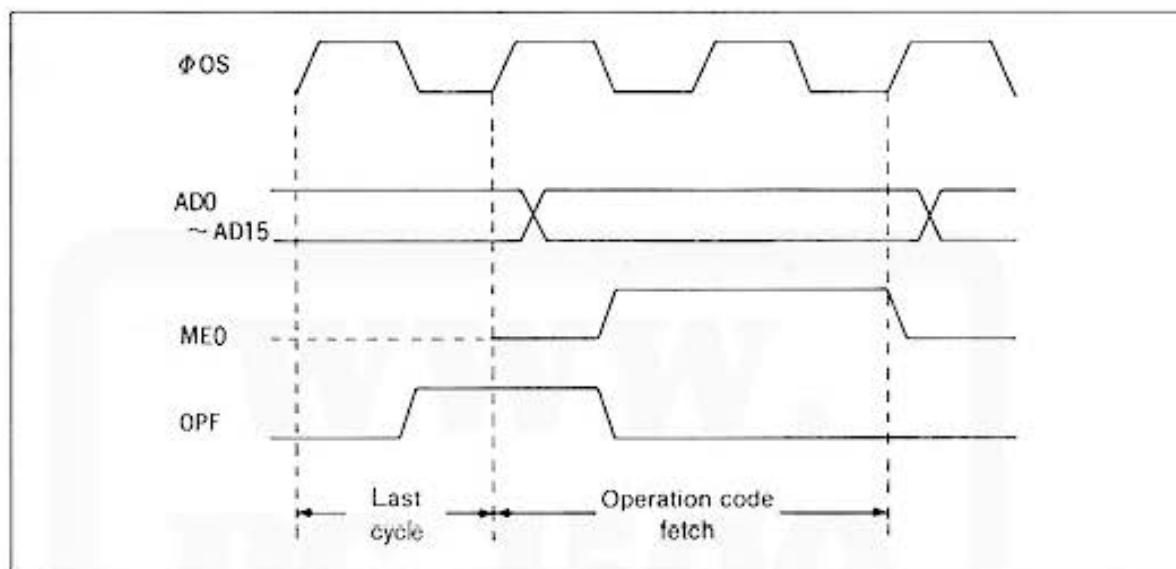
(12) BAK

Bus request acknowledge signal. When BRQ goes high, the CPU issues a high level of signal on BAK in response to it. The CPU keeps address bus (AD0~AD15), data bus (D0~D7), ME0, ME1, R/W, and OD high impedance when BAK is in high level.

(13) OPF

Operation code fetch signal which is sent out when CPU fetches operation code (instruction code).

OPF is issued only when operation code is fetched and will not be issued in fetching address data, immediate data, and second byte of the two-step instruction.

**(14) IN0~IN7**

Input port through which the CPU receives 8-bit data into the accumulator. As internal pullup resistor is used, the CPU assumes it to be high level when not connected.

(15) PΦ

External latch clock. With high level of this clock, the contents of the accumulator is sent on the data bus. Addition of IC will comprise an output port.

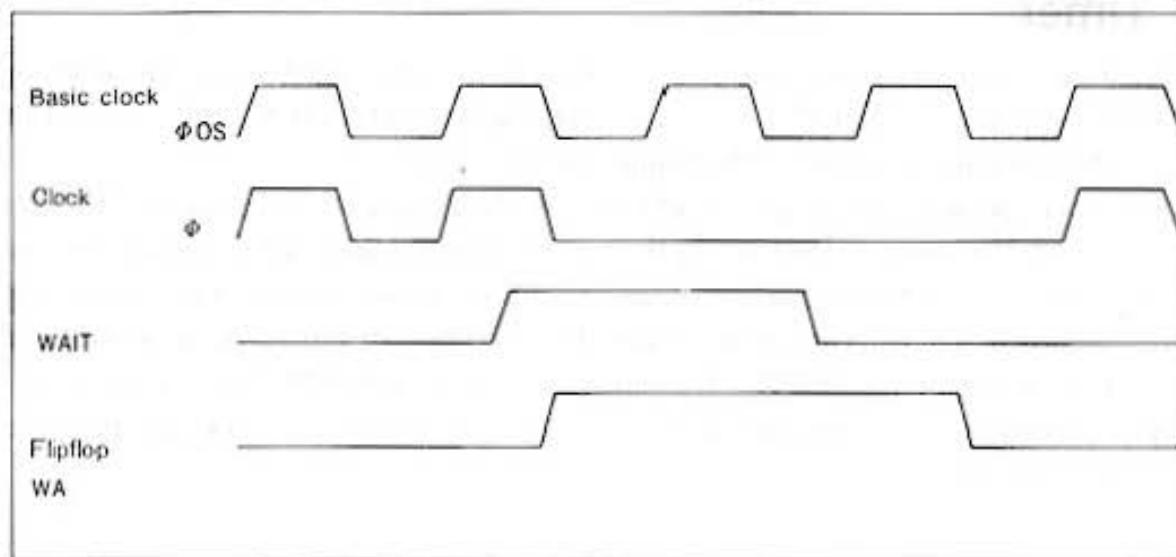
REFERENCE: ATP instruction.

(16) PU, PV

These are CPU internal flipflop outputs. PU and PV are furnished with set and reset instructions.

⑯ WAIT

CPU wait signal. A high on this line stops the clock ϕ so that the CPU halts its operation. As soon as WAIT turns low, the CPU resumes the operation.



NOTE: WA is the flipflop dedicated to WAIT, by which the WAIT input is received at the falling edge of the clock Φ_{OS} . Because the CPU operating clock ϕ stops when WA is high, it makes the CPU stopped, consequently.

⑰ H0~H7

LCD backplate signal output. As the liquid crystal display (LCD) is driven by backplate signals and segment signals, the CPU controls the backplate signals.

⑲ VA, VB, VM, VDIS

LCD backplate power supply inputs.

⑳ HIN

Input signal to the counter from which the LCD backplate signals H0~H7 are generated. Normally, connected to pin HA of the CPU.

㉑ HA

CPU internal divider output pin. It is used for the basic clock that drives the LCD that connected with HIN and the segment signal generating LSI.

㉒ DISP

LCD on/off control signal output which can be set or reset by means of instruction.

㉓ BFO, BFI

BF flipflop output (BFO) and input (BFI). The BF flipflop is reset by the OFF instruction from the CPU, and will be set when the input BFI is turned to high level.

BFO is in low level when the BF flipflop is set and in high level when reset.

As VGG is power supply to the BF flipflop, the contents of the flipflop are retained as long as VGG is in supply.

Normally, it is used for memory backup system.

2-3. Functions

2-3-1. Timer

The timer is a 9-bit polynominal counter. The counter value can be set by the AM0 or AM1 instruction of the CPU. Shown in "POLINOMINAL COUNTER" is the list of hexadecimal data of counter with decimal count number.

The timer operates continuously at all times. When the counter value reaches 1FFH, it issues interrupt request to the CPU. If IE (Interrupt Enable) flag is active at that point, the CPU jumps to the timer interrupt processing routine whose high order byte address is represented by the contents of the address FFFAH and low order byte address by the contents of the address FFFBH. The timer has to be set to 000H when it is not used. Since it counts in synchronization with the clock ϕ_F , each one cycle of ϕ_F increments the counter one step.



When the 4MHz crystal oscillator is in connection, the timer is incremented at each 32 μ sec, because ϕ_F is 31.25kHz.

"POLINOMINAL COUNTER"

COUNT NUMBER	DATA OF COUNTER								
511	1FF	459	0B7	407	1AB	355	0A7	303	1E5
510	0FF	458	05B	406	1D5	354	153	302	1F2
509	07F	457	02D	405	0EA	353	0A9	301	1F9
508	03F	456	116	404	075	352	154	300	0FC
507	01F	455	18B	403	03A	351	1AA	299	17E
506	00F	454	1C5	402	11D	350	0D5	298	1BF
505	107	453	1E2	401	08E	349	06A	297	0DF
504	183	452	0F1	400	047	348	035	296	06F
503	1C1	451	078	399	123	347	01A	295	137
502	1E0	450	13C	398	191	346	10D	294	09B
501	0F0	449	19E	397	0C8	345	186	293	04D
500	178	448	1CF	396	064	344	0C3	292	126
499	1BC	447	1E7	395	032	343	161	291	093
498	1DE	446	1F3	394	119	342	1B0	290	049
497	1EF	445	0F9	393	08C	341	1D8	289	124
496	1F7	444	07C	392	046	340	1EC	288	092
495	0FB	443	13E	391	023	339	0F6	287	149
494	07D	442	19F	390	111	338	17B	286	1A4
493	03E	441	0CF	389	088	337	0BD	285	0D2
492	11F	440	167	388	044	336	05E	284	169
491	08F	439	1B3	387	022	335	12F	283	1B4
490	147	438	0D9	386	011	334	197	282	1DA
489	1A3	437	06C	385	008	333	0CB	281	1ED
488	1D1	436	036	384	004	332	165	280	1F6
487	0E8	435	11B	383	002	331	1B2	279	1FB
486	074	434	08D	382	001	330	1D9	278	0FD
485	13A	433	146	381	100	329	0EC	277	07E
484	19D	432	0A3	380	080	328	076	276	13F
483	0CE	431	151	379	040	327	13B	275	09F
482	067	430	0A8	378	020	326	09D	274	04F
481	133	429	054	377	010	325	04E	273	127
480	099	428	12A	376	108	324	027	272	193
479	04C	427	095	375	084	323	113	271	0C9
478	026	426	04A	374	042	322	089	270	164
477	013	425	025	373	021	321	144	269	0B2
476	009	424	112	372	110	320	0A2	268	159
475	104	423	189	371	188	319	051	267	0AC
474	082	422	1C4	370	0C4	318	028	266	056
473	041	421	0E2	369	062	317	014	265	12B
472	120	420	071	368	031	316	10A	264	195
471	090	419	038	367	018	315	085	263	0CA
470	148	418	11C	366	10C	314	142	262	065
469	0A4	417	18E	365	086	313	0A1	261	132
468	052	416	0C7	364	043	312	150	260	199
467	129	415	163	363	121	311	1A8	259	0CC
466	194	414	1B1	362	190	310	0D4	258	066
465	1CA	413	0D8	361	1C8	309	16A	257	033
464	0E5	412	16C	360	0E4	308	0B5	256	019
463	172	411	0B6	359	072	307	05A	255	00C
462	1B9	410	153	358	139	306	12D	254	006
461	0DC	409	0AD	357	09C	305	196	253	003
460	16E	408	156	356	14E	304	1CB	252	101

COUNT NUMBER	DATA OF COUNTER								
251	180	199	09A	147	141	95	00E	43	01C
250	0C0	198	14D	146	1A0	94	007	42	10E
249	060	197	1A6	145	0D0	93	103	41	087
248	030	196	0D3	144	168	92	181	40	143
247	118	195	069	143	0B4	91	1C0	39	1A1
246	18C	194	134	142	15A	90	0E0	38	1D0
245	0C6	193	19A	141	1AD	89	070	37	1E8
244	063	192	1CD	140	1D6	88	138	36	0F4
243	131	191	1E6	139	1ER	87	19C	35	17A
242	098	190	0F3	138	1F5	86	1CE	34	1BD
241	14C	189	079	137	0FA	85	0E7	33	0DE
240	0A6	188	03C	136	17D	84	173	32	16F
239	053	187	11E	135	0BE	83	0B9	31	1B7
238	029	186	1BF	134	15F	82	05C	30	0DB
237	114	185	1C7	133	0AF	81	12E	29	06D
236	18A	184	1E3	132	157	80	097	28	136
235	0C5	183	1F1	131	0AB	79	04B	27	19B
234	162	182	0F8	130	155	78	125	26	0CD
233	0B1	181	17C	129	0AA	77	192	25	166
232	058	180	1BE	128	055	76	1C9	24	0B3
231	12C	179	1DF	127	02A	75	1E4	23	059
230	096	178	0EF	126	015	74	0F2	22	02C
229	14B	177	177	125	00A	73	179	21	016
228	1A5	176	0BB	124	005	72	0BC	20	10B
227	1D2	175	05D	123	102	71	15E	19	185
226	1E9	174	02E	122	081	70	1AF	18	1C2
225	1F4	173	017	121	140	69	1D7	17	0E1
224	1FA	172	00B	120	0A0	68	0EB	16	170
223	1FD	171	105	119	050	67	175	15	1B8
222	0FE	170	182	118	128	66	08A	14	1DC
221	17F	169	0C1	117	094	65	15D	13	1EE
220	0BF	168	160	116	14A	64	0AE	12	0F7
219	05F	167	0B0	115	0A5	63	057	11	07B
218	02F	166	158	114	152	62	02B	10	03D
217	117	165	1AC	113	1A9	61	115	9	01E
216	08B	164	0D6	112	1D4	60	08A	8	10F
215	145	163	16B	111	1EA	59	045	7	187
214	1A2	162	185	110	0F5	58	122	6	1C3
213	0D1	161	0DA	109	07A	57	091	5	1E1
212	068	160	16D	108	1D3	56	048	4	1F0
211	034	159	1B6	107	09E	55	024	3	1F8
210	11A	158	1DB	106	14F	54	012	2	1FC
209	18D	157	0ED	105	1A7	53	109	1	1FE
208	1C6	156	176	104	1D3	52	184	0	1FF
207	0E3	155	1BB	103	0E9	51	0C2		
206	171	154	0DD	102	174	50	061		
205	0B8	153	06E	101	1BA	49	130		
204	15C	152	037	100	1DD	48	198		
203	1AE	151	01B	99	0EE	47	1CC		
202	0D7	150	00D	98	077	46	0E6		
201	06B	149	106	97	03B	45	073		
200	135	148	083	96	01D	44	039		

2-3-2. Interrupts

There are following three kinds of interrupt for the LH5801 CPU.

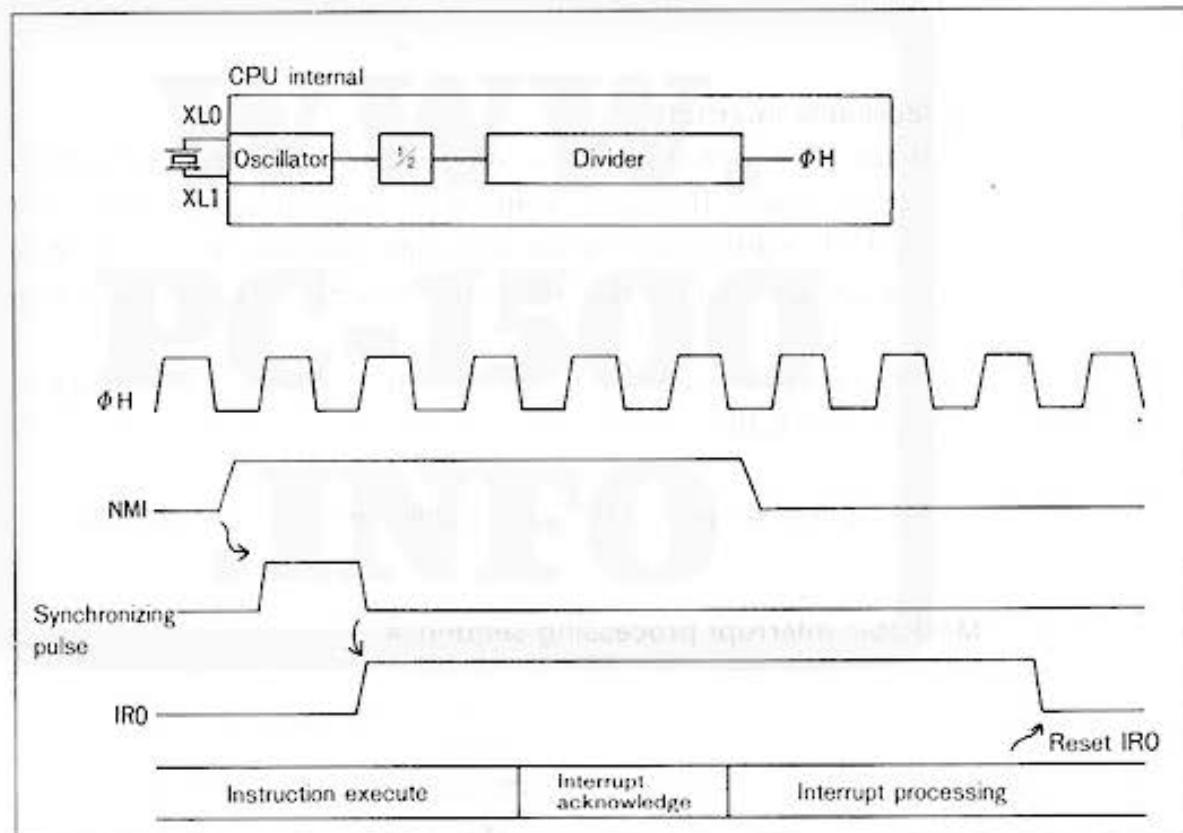
① Non-maskable interrupt

When NMI is turned from low to high level, it sets the flipflop IR0 active, upon which time interrupt is requested to the CPU, so that the CPU starts executing the interrupt processing after completion of current instruction execution.

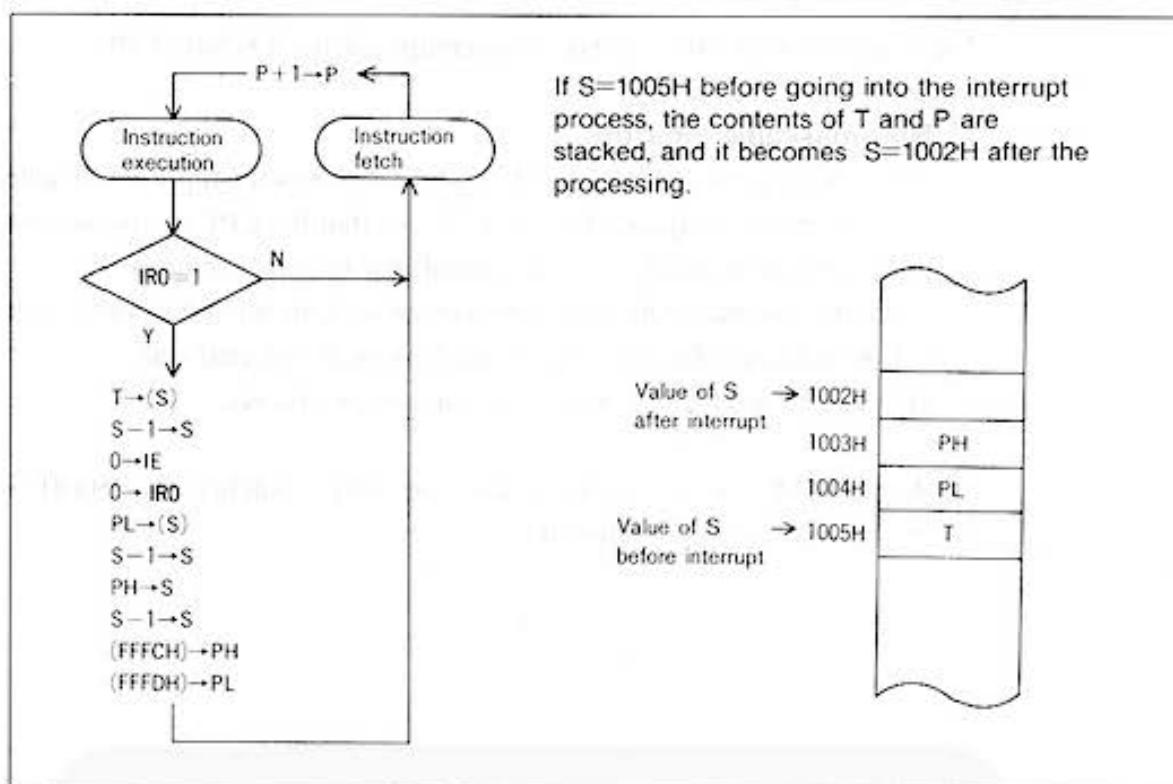
Since the non-maskable interrupt is given with the highest priority order, the interrupt will be acknowledged at once, regardless of its internal state.

IR0 will be reset in a course of the interrupt process.

As the NIM input is sampled by the clock ϕ_H , it will become 250kHz when the 4MHz crystal oscillator is connected.



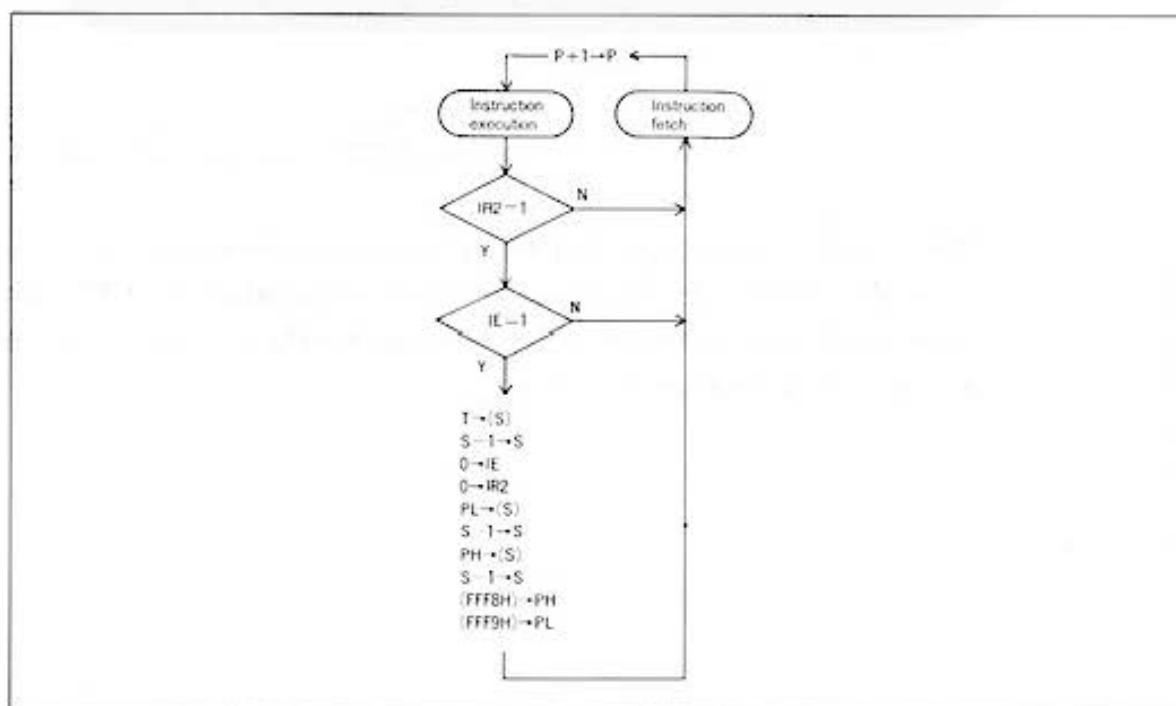
When NMI is turned high, the CPU creates synchronizing pulse at the rising edge of the clock ϕ_H , which sets IR0 active. If IR0 is active when the CPU acknowledged the interrupt after completion of execution, it goes into the interrupt routine and IR0 is reset in a course of the interrupt processing.

Non-maskable interrupt processing sequence**② Maskable interrupt**

When MI is turned from low to high level, it sets the flipflop IR2 active. If the interrupt enable IE is active at this point, interrupt is requested to the CPU, so that the CPU starts executing the interrupt processing after completion of current instruction execution. IR2 will be reset in a course of the interrupt process.

When interrupt request is issued while IE is inactive, the interrupt will be ignored even though IR2 is set.

MI input is sampled in the same manner as in the case of NMI

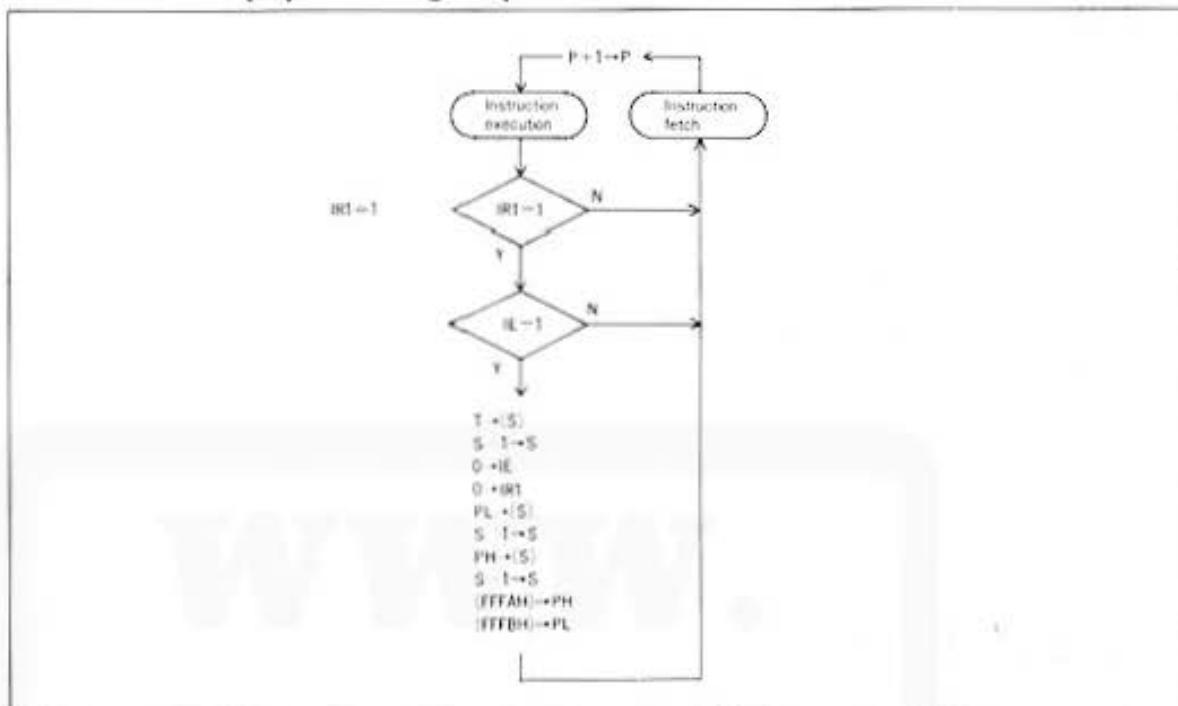
Maskable interrupt processing sequence

③ Timer interrupt

When interrupt is requested from the timer, it sets the flipflop IR1 active. If the interrupt enable IE is active at this point, the CPU starts executing the interrupt processing after completion of current instruction execution, and IR1 will be reset in a course of the interrupt process.

When interrupt is requested while IE is inactive, the interrupt will be ignored even though IR1 is set.

Timer interrupt processing sequence



Return to main routine

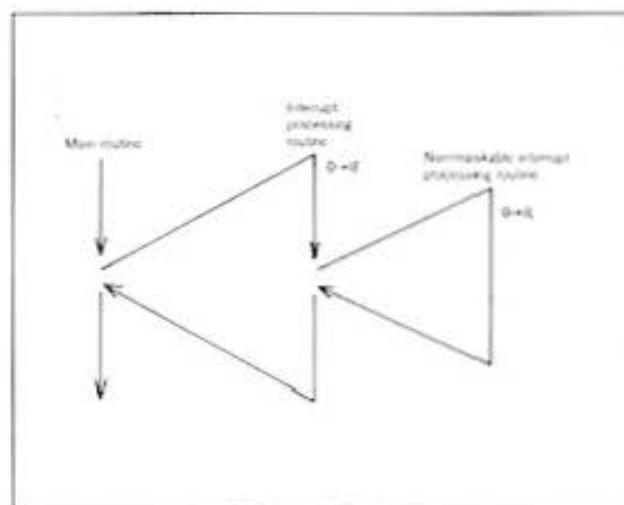
The RTI instruction is used for returning from the interrupt processing routine to the main routine.

Because the contents of the T register and the program counter are stored in the stack at the beginning of the interrupt processing routine, the contents of the T register in the stack returns by the RTI instruction. Since the interrupt enable flag IE is contained in the T register, the contents of IE immediately before the interrupt returns by the RTI instruction.

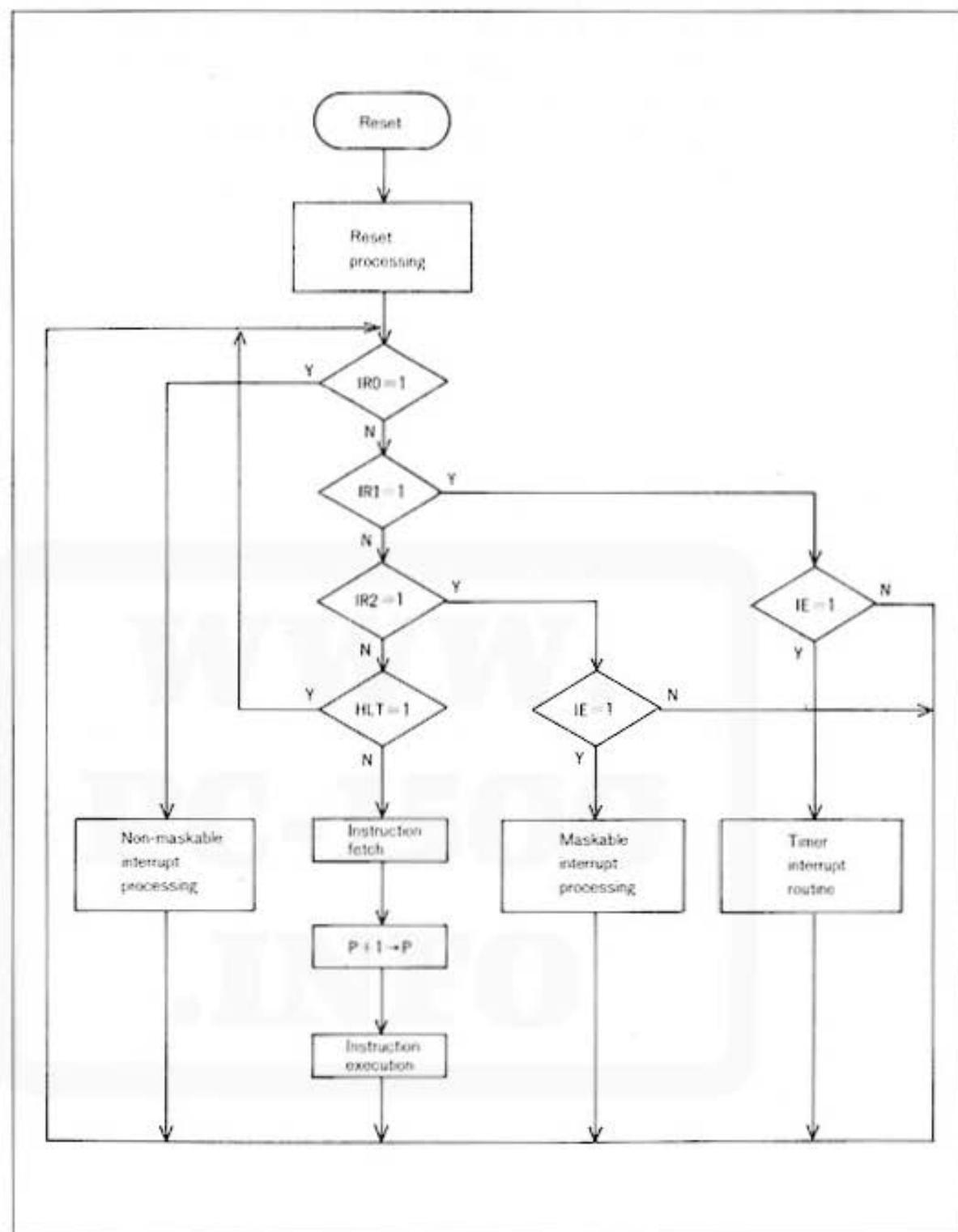
To disable maskable or timer interrupt by the main routine after returning from the interrupt processing routine, the bit in the stack corresponding to the flag IE must be reset.

Priority order of interrupts

- (1) Non-maskable interrupt responds to the interrupt request whatever the CPU internal state may be. Also, it responds in the first priority even during execution of interrupt routine by other interrupt.



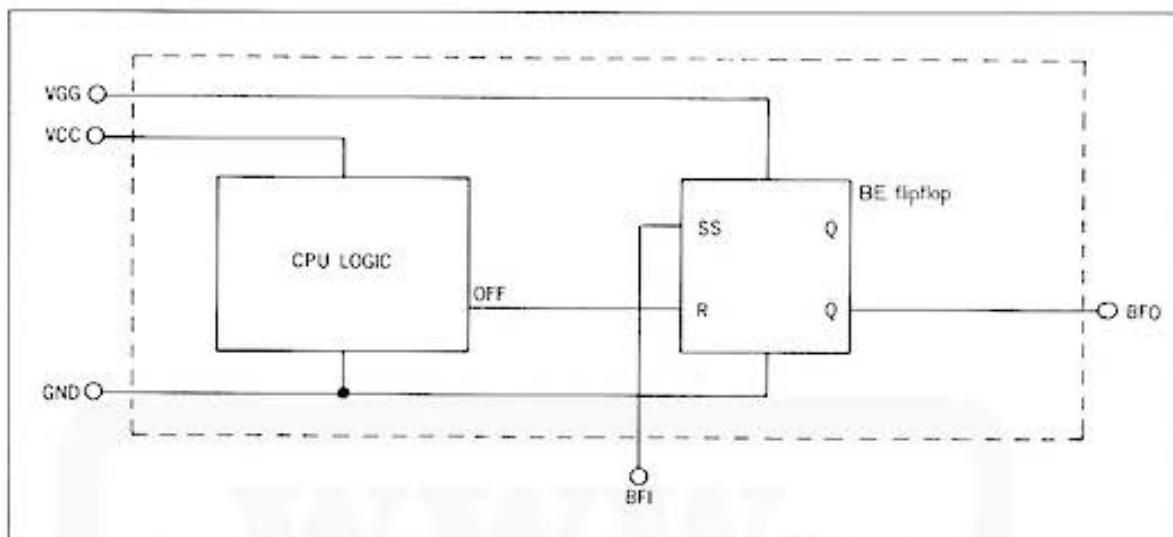
2-3-4. CPU system sequence



2-3-5. BF flipflop

The CPU has two supplies of power; VCC and VGG. As the BF flipflop is driven by VGG, the state of the BF flipflop is retained as long as VGG is in supply, even if VCC supply is out. The BF flipflop is set when the BFI input is turned from low to high level and reset when the OFF instruction is issued from the CPU.

Low state of signal is on the BF flipflop output BFO when the flipflop is active and high state of signal when inactive.



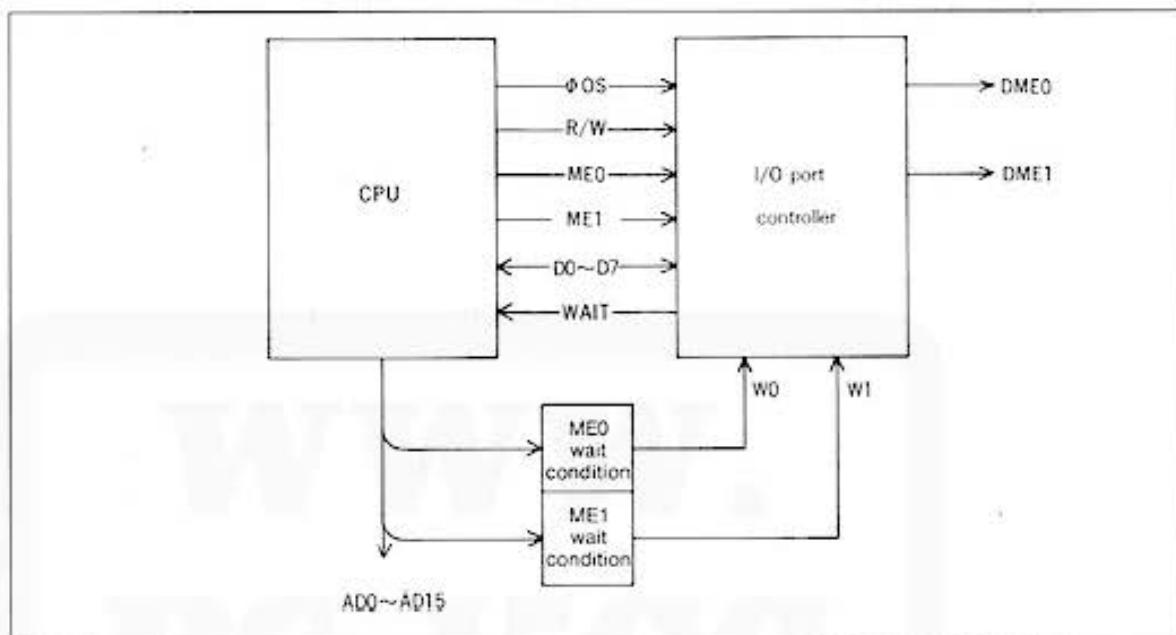
It is possible to comprise a memory backup system using this BF flipflop.

2-3-6. WAIT function (interfacing with the slow access time memory)

Because the CPU access time is very fast, the CPU must be held with the wait signal in order to access a slow access time memory.

Basically, the wait signal is created by the externally provided counter, but the I/O port controller with the programmable wait time control feature (LH8511) is used for this purpose. For more details of this function, refer to the section discussing the I/O port controller.

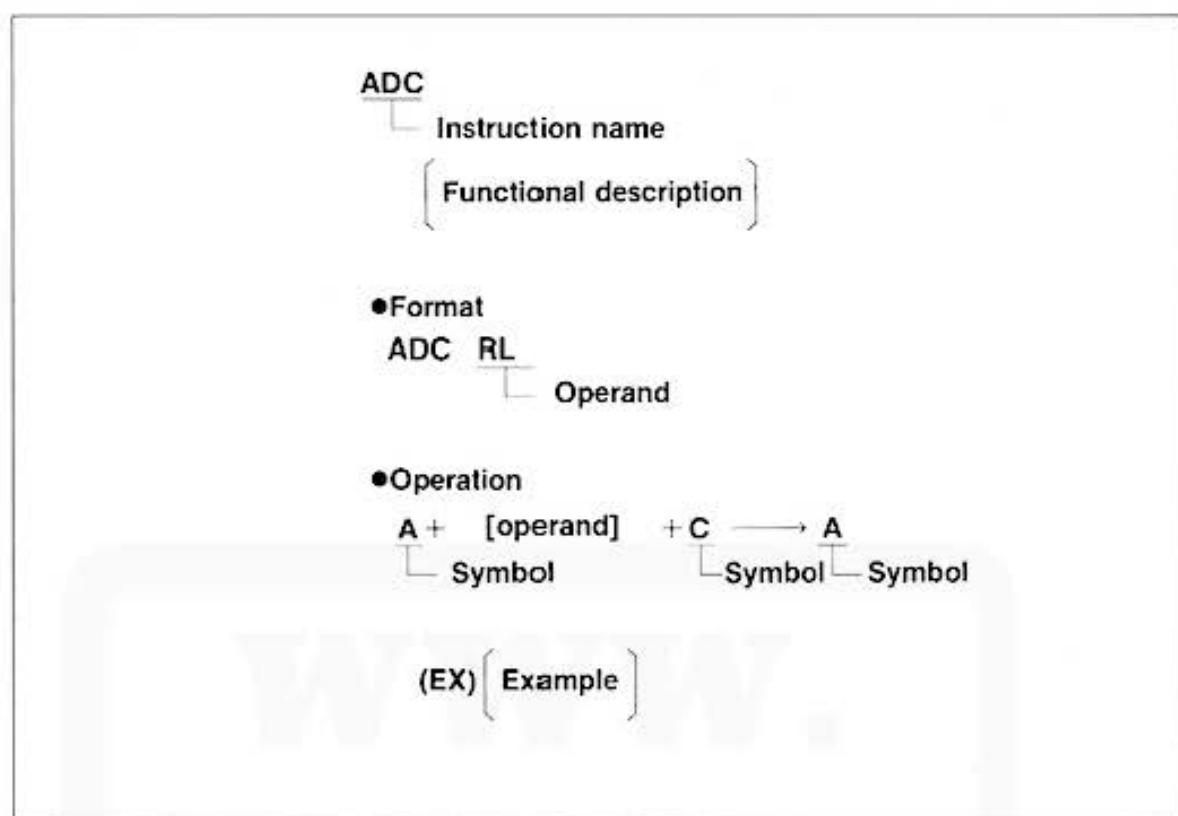
Shown in the next page is the connection of the CPU with the I/O port controller.



2-4. LH5801 instructions

2-4-1. Outline

This section deals with function of each instruction.



Symbols used in discussing operand, function, and operation

- A: Contents of the accumulator (8 bits)
- XL: Contents of the low order 8 bits of Xreg
- YL: Contents of the low order 8 bits of Yreg
- UL: Contents of the low order 8 bits of Ureg
(RL represents either of XL, YL, or UL.)
- XH: Contents of the high order 8 bits of Xreg
- YH: Contents of the high order 8 bits of Yreg
- UH: Contents of the high order 8 bits of Ureg
(RH represents either of XH, YH, or UH.)
- Xreg: Contents of the X register (16 bits)
- Yreg: Contents of the Y register (16 bits)
- Ureg: Contents of the U register (16 bits)
(Rreg represents either of Xreg, Yreg, or Ureg.)
- P: Contents of the program counter (16 bits)
- PL: Contents of low order 8 bits of the program counter
- PH: Contents of high order 8 bits of the program counter
- S: Contents of the stack pointer (16 bits)
- SL: Contents of low order 8 bits of the stack pointer
- SH: Contents of high order 8 bits of the stack pointer
- (Rreg): Contents of the memory represented by Rreg (accessed with ME0)
- #(Rreg): Contents of the memory represented by Rreg (accessed with ME1)
- (ab): Contents of the memory represented by 16 bits of ab (accessed by ME0); where a represents high order 8 bits of the address and b low order 8 bits

#(ab):	Contents of the memory represented by 16 bits of <i>ab</i> (accessed by ME1); where <i>a</i> represents high order 8 bits of the address and <i>b</i> low order 8 bits
<i>i</i> :	8-bit immediate data
<i>i, j</i> :	16-bit immediate data of which high order 8 bits are represented by <i>i</i> and low order 8 bits by <i>j</i>
C:	Carry flag
IE:	Interrupt enable flag
Z:	Zero flag
V:	Overflow flag
H:	Half carry flag
→	Data flow direction
∧	AND
∨	OR
⊕	Exclusive OR
+	ADD
-	SUBTRACT

2-4-2. Add, subtract, and logical instructions

① ADC (ADd with Carry)

Either the contents of the internal register or external memory are 8-bit added with the accumulator including carry, and its result is stored in the accumulator.
C, H, Z, and V may change.

•Format	•Operation
ADC RL	
ADC RH	
ADC (Rreg)	$A + [\text{operand}] + C \rightarrow A$
ADC #(Rreg)	
ADC (<i>ab</i>)	
ADC #(<i>ab</i>)	

(EX) ADC XL

Value of the XL register is added to the accumulator, providing "C=0".

A =	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table>	0	0	0	0	0	0	1	0	Execute	A =	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	0	1	1	0	1	0	1
0	0	0	0	0	0	1	0													
0	0	1	1	0	1	0	1													
XL =	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr></table>	0	0	1	1	0	0	1	1			C=0 , H=0 , Z=0 , V=0								
0	0	1	1	0	0	1	1													
	C=0																			

(2) ADI (ADD Immediate)

To either the accumulator or the external memory is added the immediate data. In the case of ADI to the accumulator, carry is included in the operation.

C, H, Z, and V may change.

•Format

ADI A,i

•Operation $A + i + C \rightarrow A$

ADI (Rreg),i

[operand] + i → [operand]

ADI #(Rreg),i

ADI (ab),i

ADI #(ab),i

(EX) ADI (Xreg), 20H

20H is added to the memory represented by X register.

Xreg=4F00H

Contents of 4F00H =	<table border="1"> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>	0	0	1	1	0	0	1	1	Execute	Contents of 4F00H =	<table border="1"> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>	0	1	0	1	0	0	1	1
0	0	1	1	0	0	1	1													
0	1	0	1	0	0	1	1													
			C=0 , H=0 , Z=0 , V=0																	

(3) DCA (DeCimal Add)

Decimal addition is carried out between the external memory and the accumulator including carry, and its result is stored in the accumulator.

C, H, Z, and V may change.

•Format

DCA (Rreg)

•Operation1) $A + 66H \rightarrow A$

DCA #(Rreg)

2) $A + [\text{operand}] + C \rightarrow A$

(C, H, Z, and V may change.)

3) $A + DA \rightarrow A$

Where, DA is used for compensation of decimal number which is dependent on the value of flags C and H in regard to Item 2).

C	H	DA
0	0	9AH
0	1	A0H
1	0	F0H
1	1	00H

(EX) DCA (Yreg)

Decimal addition is carried out between the memory represented by the Y register and the accumulator.

Y reg = 4700H

A	Contents of 4700H	C	After execution of DCA			Decimal addition
			C	H	A	
35H	27H	0	0	1	62H	$35 + 27 + 0 = 62$
35H	27H	1	0	1	63H	$35 + 27 + 1 = 63$
35H	67H	0	1	1	02H	$35 + 67 + 0 = 102$
35H	67H	1	1	1	03H	$35 + 67 + 1 = 103$

④ ADR (ADD Rreg)

Contents of the accumulator are added to the R register.
C, H, Z, and V may change.

•Format

ADR Rreg

•Operation

1) RL + A → RL

(C, H, Z, and V may change.)

2) RH + C → RH

(EX) ADR Xreg

Contents of the accumulator are added to X register.

A =

1	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

 XL =

0	1	0	0	1	0	1	1
---	---	---	---	---	---	---	---

XL =

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

 $\xrightarrow{\text{Execute}}$ XH =

0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

XH =

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

 C=1, H=0, Z=0, V=1**⑤ SBC (Subtract with Carry)**

The contents of the accumulator are subtracted by the internal register or external memory including \bar{C} , and its result is stored in the accumulator.

This operation may also be expressed in the following manner.

Complement of the internal register or external memory is obtained first, addition is carried out including carry, then its result is stored in the accumulator.

C, H, Z, and V may change.

•Format

SBC RL

•OperationA - [operand] - $\bar{C} \rightarrow A$

SBC RH

(C, H, Z, and V may change.)

SBC (Rreg)

or,

SBC #(Rreg)

A + [operand] + C → A

SBC (ab)

(C, H, Z, and V may change.)

SBC #(ab)

(EX) SBC XL

Contents of XL register are subtracted from the accumulator.

A =

0	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---

XL =

0	0	1	0	0	0	0	1
---	---	---	---	---	---	---	---

 $\bar{C}=0$ A =

0	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---

 $\xrightarrow{\text{Execute}}$ A =

0	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

XL =

1	1	0	1	1	1	1	0
---	---	---	---	---	---	---	---

C=1, H=1, Z=0, V=0

C=1

⑥ SBI (SuBtract Immediate)

The contents of the accumulator are subtracted by the immediate data including \bar{C} . C, H, Z, and V may change.

•Format

SBI A,i

•Operation

$$A - i - \bar{C} \rightarrow A$$

(C, H, Z, and V may change.)

or,

$$A + \bar{i} + C \rightarrow A$$

(C, H, Z, and V may change.)

(EX) SBI A, 07H

$$A = \boxed{0\ 0\ 1\ 0\ 0\ 1\ 0\ 0}$$

$$i = \boxed{0\ 0\ 0\ 0\ 0\ 1\ 1\ 1}$$

 $\bar{C}=1$ 

$$A = \boxed{0\ 0\ 1\ 0\ 0\ 1\ 0\ 0}$$

Execute

$$\longrightarrow A = \boxed{0\ 0\ 0\ 1\ 1\ 1\ 0\ 0}$$

$$\bar{i} = \boxed{1\ 1\ 1\ 1\ 1\ 0\ 0\ 0}$$

C=1, H=0, Z=0, V=0

C=0

⑦ DCS (DeCimal Subtract)

The contents of the accumulator are decimal subtracted by the external memory including \bar{C} , and its result is stored in the accumulator.

C, H, Z, and V may change.

•Format

DCS (Rreg)

•Operation

$$1) A + [\text{operand}] + C \rightarrow A$$

(C, H, Z, and V may change.)

DCS #(Rreg)

$$2) A + DA \rightarrow A$$

Where, DA is used for compensation of decimal number which is dependent on the value of flags C and H in regard to item 1).

C	H	DA
0	0	9AH
0	1	A0H
1	0	F0H
1	1	00H

(EX) DCS (Xreg)

The contents of the memory represented by the X register are decimal subtracted from the accumulator.

Xreg=4700H

A	Contents of 4700H	C	After execution of DCS			Decimal subtraction
			C	H	A	
42H	31H	1	1	1	11H	42-31-0=11
42H	31H	0	1	1	10H	42-31-1=10
23H	54H	1	0	0	69H	23-54-0=69-100
23H	54H	0	0	0	68H	23-54-1=68-100

⑧ AND

The contents of the accumulator are ANDed with the value of the external memory, and its result is stored in the accumulator.
Only the flag Z changes.

•Format

AND (Rreg)

AND #(Rreg)

AND (ab)

AND #(ab)

•Operation $A \wedge [\text{operand}] \rightarrow A$

(Z changes.)

(EX) AND (Ureg)

Ureg=4700H

A =

1	1	0	1	0	1	0	0
---	---	---	---	---	---	---	---

A =

0	1	0	0	0	1	0	0
---	---	---	---	---	---	---	---

Contents of 4700H =

0	1	1	0	1	1	0	0
---	---	---	---	---	---	---	---

Execute

Z=0

⑨ ANI (ANd Immediate)

The contents of the accumulator or external memory are ANDed with the immediate data, and its result is stored in the accumulator or the external memory.

Only the flag Z changes.

•Format

ANI A,j

ANI (Rreg),j

ANI #(Rreg),j

ANI (ab),j

ANI #(ab),j

•Operation $[\text{operand}] \wedge i \rightarrow [\text{operand}]$

(Flag Z changes.)

⑩ ORA (OR Acc)

The contents of the accumulator are ORed with the value of the external memory, and its result is stored in the accumulator.

Only the flag Z changes.

•Format

ORA (Rreg)

ORA #(Rreg)

ORA (ab)

ORA #(ab)

•Operation $A \vee [\text{operand}] \rightarrow A$

(Flag Z changes.)

(EX) ORA (Ureg)

Ureg=4700H

A =

0	1	0	1	1	0	0	0
---	---	---	---	---	---	---	---

A =

0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

Execute

Z=0

Contents of 4700H =

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

(11) ORI (OR Immediate)

The contents of the accumulator or external memory are ORed with the immediate data, and its result is stored in the accumulator or the external memory.

Only flag Z changes.

•Format

ORI A,*i*
ORI (Rreg),*i*
ORI #(Rreg),*i*
ORI (ab),*i*
ORI #(ab),*i*

•Operation

[Operand] $\vee i \rightarrow$ [operand]
(Z changes.)

(12) EOR (Exclusive OR)

The contents of the accumulator are exclusively ORed with the value of the external memory, and its result is stored in the accumulator.

Only the flag Z changes.

•Format

EOR (Rreg)
EOR #(Rreg)
EOR (ab)
EOR #(ab)

•Operation

A \oplus [operand] \rightarrow A
(Z changes.)

(EX) EOR (Xreg)

Xreg=4700H

A =	0 0 1 1 0 1 1 0	Execute	A =	0 1 0 1 1 0 1 1
Contents of 4700H =	0 1 1 0 1 1 0 1		Z=0	

(13) EAI (Exclusive or Acc and Immediate)

The contents of the accumulator are exclusively ORed with the immediate data, and its result is stored in the accumulator.

Only the flag Z changes.

•Format

EAI *i*

•Operation

A $\oplus i \rightarrow$ A
(Z changes.)

(14) INC (INCrement)

The value of the accumulator or the register is INCremented by one. In the case of the 8-bit register (A, RL, RH), it makes flags C, V, H and Z changed. In the case of the 16-bit register Rreg, no flag change takes place.

•Format

INC A
INC RL
INC RH
INC Rreg

•Operation

[Operand] + 1 \rightarrow [operand]
(C, V, H, and Z changed.)
[Operand] + 1 \rightarrow [operand]

(EX) INC XL

XL =	0 0 1 1 0 1 1 1	Execute	XL =	0 0 1 1 1 0 0 0
C=0, H=0, Z=0, V=0				

(15) DEC (DECrement)

The value of the accumulator or register is DECREMENTED by one. In the case of the 8-bit register (A, RL, RH), it makes flags C, V, H and Z changed. In the case of the 16-bit register Rreg, no flag change takes place.

•Format

DEC A

DEC RL

DEC RH

DEC Rreg

•Operation

[Operand] - 1 → [operand]

(C, V, H, and Z changed.)

[Operand] - 1 → [operand]

(EX) DEC Xreg

XL =	0 0 0 0 0 0 0 0	Execute	XL =	1 1 1 1 1 1 1 1
XH =	0 0 0 1 0 0 0 1		XH =	0 0 0 1 0 0 0 0

2-4-3. Compare and bit test**(16) CPA (ComPare Acc)**

The contents of the accumulator are compared with register or external memory, and its result is represented by flags C, V, H, and Z.

•Format

CPA RL

CPA RH

CPA (Rreg)

CPA #(Rreg)

CPA (ab)

CPA #(ab)

•Operation

A - [operand] → change in C, V, H, and Z

or

A + [operand] + 1 → change in C, V, H, and Z

Comparison	C	Z	V	H
A > [operand]	1	0	*	*
A = [operand]	1	1	*	*
A < [operand]	0	0	*	*

* : Flags V and H may change according to the condition mentioned in "Status flag", but it has no meaning with the CPA instruction.

(EX) CPA XL

A =	0 1 0 1 0 1 0 0	Execute	C=1, Z=0
XL =	0 1 0 1 0 0 0 0		

(17) CPI (ComPare Immediate)

The contents of the accumulator or register are compared with the immediate data, and its result is represented by flags C, V, H, and Z.

•Format

CPI RL,*i*
CPI RH,*i*
CPI A,*i*

•Operation

[Operand] - *i* → change in C, V, H, Z

Comparison	C	Z	V	H
[Operand]> <i>i</i>	1	0	*	*
[Operand]= <i>i</i>	1	1	*	*
[Operand]< <i>i</i>	0	0	*	*

* : Refer to "CPA instruction".

(18) BIT (test BIT)

The accumulator is ANDed with the external memory, and its result is represented by the flag Z.

•Format

BIT (Rreg)
BIT #(Rreg)
BIT (ab)
BIT #(ab)

•Operation

A \wedge [operand] → Z flag change

(EX) BIT (Xreg)	Xreg=4F00H									
A =	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	0	0	0	0	0	0	Execute → Z=1
1	0	0	0	0	0	0	0			

Contents of 4F00H =

0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

(19) BII (test Bit Immediate)

The contents of the accumulator or the external memory are ANDed with the immediate data, and its result is represented by state of the flag Z.

•Format

BII A,*i*
BII (Rreg),*i*
BII #(Rreg),*i*
BII (ab),*i*
BII #(ab),*i*

•Operation

[Operand] \wedge *i* → Z flag change

2-4-4. Transfer and search instructions

⑩ LDA (LoaD Acc)

The contents of the RL register, RH register, or the internal memory are transferred to the accumulator.

Only the flag Z changes.

•Format

LDA RL

LDA RH

LDA (Rreg)

LDA #(Rreg)

LDA (ab)

LDA #(ab)

•Operation

[Operand] → A

Flag Z

1: when [operand] = 00H

0: when [operand] ≠ 00H

(EX) LDA XL

XL =

 Execute

A =

Z=0

⑪ LDE (Load and DEcrement)

The contents of the external memory (R register) are transferred to the accumulator, then "1" is decremented from R register.

Only the flag Z changes.

•Format

LDE Rreg

•Operation

(Rreg) → A

Rreg - 1 → Rreg

Flag Z

1: when (Rreg) = 00H

0: when (Rreg) ≠ 00H

(EX) LDE Yreg

Contents of 202AH =

 Execute A =

Yreg = 202AH

Yreg = 2029H

Z=0

(2) LIN (Load and INcrement)

The contents of the external memory (R register) are transferred to the accumulator, then "1" is added to the R register.

Only the flag Z changes.

•Format

LIN Rreg

•Operation $(Rreg) \rightarrow A$ $Rreg + 1 \rightarrow Rreg$

Flag Z

1: when $(Rreg) = 00H$ 0: when $(Rreg) \neq 00H$

(EX) LIN Ureg

Contents
of 0055H =

0	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---

 A =

0	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---

 $Ureg=0055H \xrightarrow{\text{Execute}} Ureg=0056H$
 $Z=0$

(2) LDI (LoaD Immediate)

The immediate data is loaded in the accumulator, RL register, RH register, or stack pointer S.

The flag Z changes when transfer is done to the accumulator and no flag change occurs in otherwise case.

Two bytes of the immediate value are transferred in the case of the stack pointer.

•Format

① LDI A,i

•Operation $i \rightarrow A$

Flag Z

1: when $i = 00H$ 0: when $i \neq 00H$

② LDI RL,i

 $i \rightarrow RL$

LDI RH,i

 $i \rightarrow RH$

(No flag changes)

③ LDI S,i,j

 $i \rightarrow SH$ $j \rightarrow SL$

(No flag changes)

(EX) LDI XH,5AH

Immediate
value =

0	1	0	1	1	0	1	0
---	---	---	---	---	---	---	---

 $\xrightarrow{\text{Execute}}$ XH =

0	1	0	1	1	0	1	0
---	---	---	---	---	---	---	---

④ LDX (LoaD Xreg)

The contents of the R register, stack pointer, or program counter are transferred to the X register.

No change takes place in flags.

•Format

LDX Rreg

LDX S

LDX P

•Operation

[Operand] → Xreg

(EX) LDX S

S=4700H $\xrightarrow{\text{Execute}}$ Xreg=4700H**⑤ STA (STore Acc)**

The contents of the accumulator are transferred to the RL register, RH register, or the external memory.

No change takes place in flags.

•Format

STA RL

STA RH

STA (Rreg)

STA #(Rreg)

STA (ab)

STA #(ab)

•Operation

A → [operand]

(EX) STA XL

A=

 $\xrightarrow{\text{Execute}}$ XL=

⑥ SDE (Store and DEcrement)

The contents of the accumulator are transferred to the external memory (Rreg), then "1" is decremented from Rreg.

No change takes place in flags.

•Format

SDE Rreg

•Operation

A → (Rreg)

Rreg - 1 → Rreg

(EX) SDE Yreg

A=

 $\xrightarrow{\text{Execute}}$ of 4700H Contents =

Yreg=4700H $\qquad \qquad \qquad$ Yreg=46FFH

(2) SIN (Store and INcrement)

The contents of the accumulator are transferred to the external memory (Rreg), then "1" is incremented to Rreg.

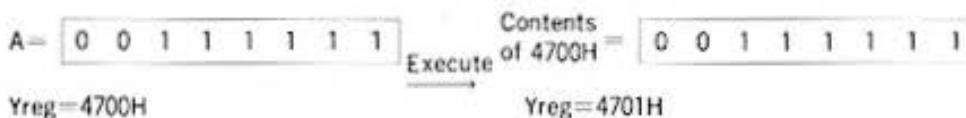
No change takes place in flags.

•Format

SIN Rreg

•Operation $A \rightarrow (Rreg)$ $Rreg + 1 \rightarrow Rreg$

(EX) SIN Yreg

**(2) STX (STore Xreg)**

The contents of Xreg are transferred to the R register, stack pointer, or program counter.

No change takes place in flags.

•Format

STX Rreg

•Operation $Xreg \rightarrow [\text{operand}]$

STX S

STX P

(2) PSH (PuSH)

The contents of the accumulator or R register are stacked in the memory specified by the stack pointer.

In the case of the accumulator, the stack pointer is decremented by one. In the case of the R register, the stack pointer is decremented by two.

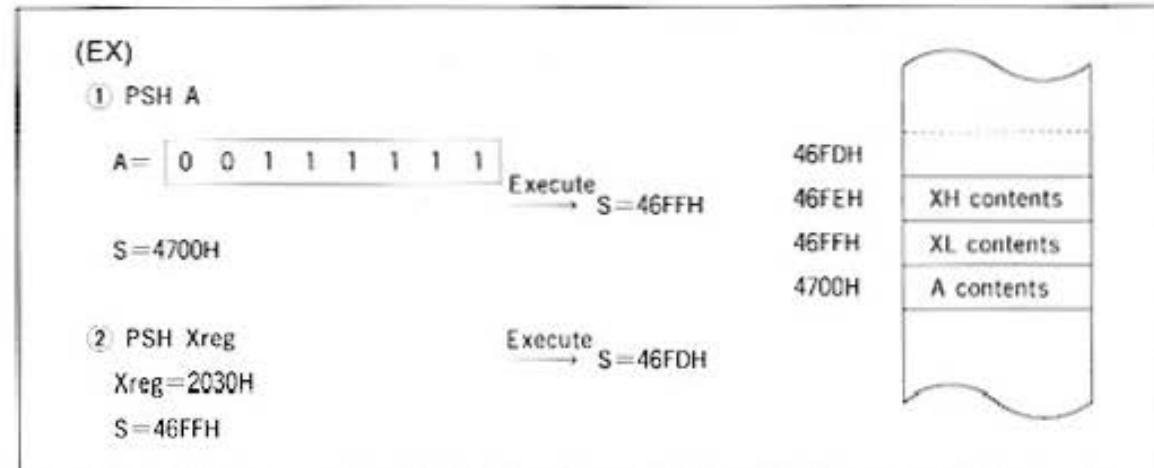
•Format

① PSH A

•Operation $A \rightarrow (S)$ $S - 1 \rightarrow S$

② PSH Rreg

 $RL \rightarrow (S)$ $S - 1 \rightarrow S$ $RH \rightarrow (S)$ $S - 1 \rightarrow S$



⑩ POP (POP)

The contents of the stack pointer transferred by the PSH instruction are returned to the accumulator or the R register.

In the case of the accumulator, the stack pointer is added by one. In the case of the R register, the stack pointer is added by two.

•Format

① POP A

•Operation

$S + 1 \rightarrow S$

$(S) \rightarrow A$

Flag Z

1: when $(S) = 00H$

0: when $(S) \neq 00H$

② POP Rreg

$S + 1 \rightarrow S$

$(S) \rightarrow RH$

$S + 1 \rightarrow S$

$(S) \rightarrow RL$

(No flag change)

(EX)

① POP Xreg

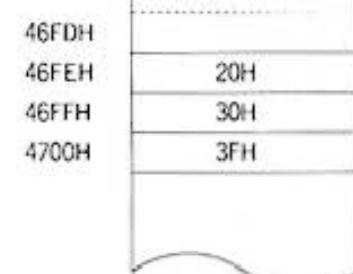
P = 46FDH → P = 46FFH

Xreg = 2030H

② POP A

P = 46FFH → P = 4700H

A = [0 0 1 1 1 1 1 1]



⑪ ATT (Acc To T)

The contents of the accumulator are transferred to the T register.

•Format

ATT

•Operation

$A \rightarrow T$

(32) TTA (T To Acc)

The contents of the T register are transferred to the accumulator.
The flag Z changes.

•Format

TTA

•Operation $T \rightarrow A$

Flag Z

1: when $T = 00H$ 0: when $T \neq 00H$ **2-4-5. Block transfer and search instructions****(33) TIN (Transfer and INcrement)**

The contents of the external memory (Xreg) are transferred to the external memory (Yreg), then both Xreg and Yreg are added by one.
No change takes place in flags.

•Format

TIN

•Operation $(Xreg) \rightarrow (Yreg)$ $Xreg + 1 \rightarrow Xreg$ $Yreg + 1 \rightarrow Yreg$

(EX) TIN

Xreg=4700H

Yreg=4800H

Xreg=4701H

Yreg=4801H

Contents of 4700H =

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Contents of 4800H =

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

(34) CIN (Compare and INcrement)

The contents of the accumulator are compared with that of the external memory (Xreg), its result is represented by flag states, then Xreg is incremented by one.

•Format

CIN

•Operation $A - (Xreg) \rightarrow$ change in C, H, Z, and V $Xreg + 1 \rightarrow Xreg$

Relation between the comparison result (A with Xreg) and flags (C, H, Z, V) is the same as in ⑯ CPA.

2-4-6. Rotate and shift instructions

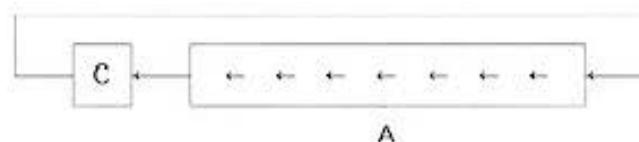
⑮ ROL (ROtate Left)

The accumulator contents are rotated left through the flag C.

•Format

ROL

•Operation



(EX) ROL

1	0 1 0 1 1 1 0 1	Execute	0	1 0 1 1 1 0 1 1
Carry	A		Carry	A

⑯ ROR (ROtate Right)

The accumulator contents are rotated right through the flag C.

•Format

ROR

•Operation



(EX) ROR

0	1 1 0 0 1 0 0 0	Execute	0	0 1 1 0 0 1 0 0
Carry	A		Carry	A

⑰ SHL (SHift Left)

The contents of the accumulator are shifted left.

•Format

SHL

•Operation



(EX) SHL

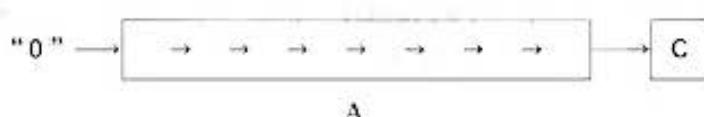
*	1 0 0 1 1 1 0 1	Execute	1	0 0 1 1 1 0 1 0
Carry	A		Carry	A

(38) SHR (SHift Right)

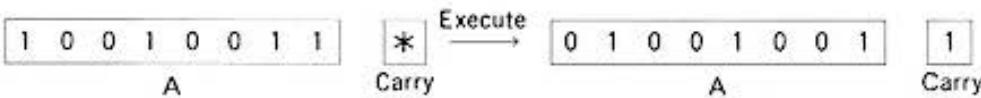
The contents of the accumulator are shifted right.

•Format

SHR

•Operation

(EX) SHR

**(39) DRL (Digit Rotate Left)**

Left rotation takes between the accumulator and the external memory (Xreg) or #(Xreg) in unit of digit (4 bits).

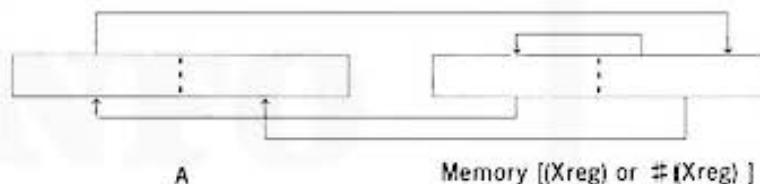
In other words, the low order 4 bits of the external memory are moved to the high order 4 bits of the external memory, the high order 4 bits of the external memory are moved to the high order 4 bits of the accumulator, the high order 4 bits of the accumulator are moved to the low order 4 bits of the external memory, and the low order 4 bits of the accumulator are moved to the low order 4 bits of the external memory at all times. No change takes place in flags.

•Format

DRL (Xreg)

•Operation

DRL #(Xreg)



(EX) DRL (Xreg)

Xreg=4700H

A = 0 1 1 0 1 1 1 1

A = 1 0 1 1 0 0 1 1

Execute

Contents of 4700H = 1 0 1 1 0 0 1 1

Contents of 4700H = 0 0 1 1 0 1 1 0

④ DRR (Digit Rotate Right)

Right rotation takes between the accumulator and the external memory (Xreg) or #(Xreg) in unit of digit (4 bits).

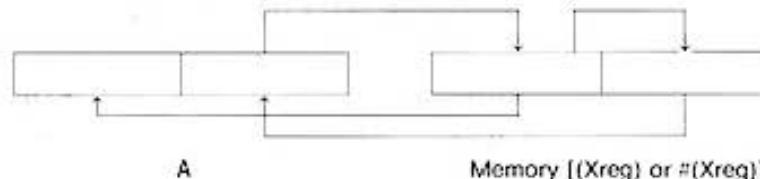
In other words, the low order 4 bits of the external memory are moved to the low order 4 bits of the accumulator, the low order 4 bits of the accumulator are moved to the high order 4 bits of the external memory, the high order 4 bits of the external memory are moved to the low order 4 bits of the external memory, and the high order 4 bits of the external memory are moved to the high order 4 bits of the accumulator memory at all times.

No change takes place in flags.

•Format

DRR (Xreg)

DRR #(Xreg)

•Operation

(EX) DRR (Xreg)

Xreg=4700H

A =

1	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

A =

0	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

Execute →

Contents of 4700H =

0	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

Contents of 4700H =

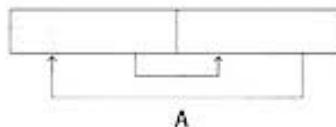
0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

⑤ AEX (EXchange Acc)

The high order 4 bits of the accumulator are swapped with the low order 4 bits.
No change takes place in flags.

•Format

AEX

•Operation

(EX) AEX

A =

1	1	0	0	1	0	1	0
---	---	---	---	---	---	---	---

 Execute → A =

1	0	1	0	1	1	0	0
---	---	---	---	---	---	---	---

2-4-7. CPU control instructions

⑫ SEC (SEt Carry)

Sets the carry flag.

No change takes place in flags.

•Format

SEC

•Operation

$I \rightarrow C$

⑬ REC (REset Carry)

Resets the carry flag.

No change takes place in flags.

•Format

REC

•Operation

$0 \rightarrow C$

⑭ CDV (Clear DiVider)

Clears the internal divider. In other words, since the CPU clock is supplied through the divider, it makes clock reset by the CDV instruction.

•Format

CDV

•Operation

$0 \rightarrow \text{divider}$

⑮ ATP (Acc To Port)

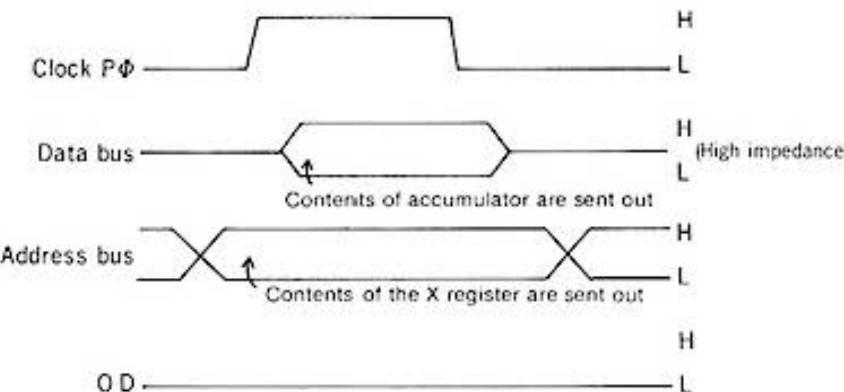
The contents of the accumulator are sent on the data bus. As the clock $P\phi$ is sent out from the CPU at this moment, it may be used for the clock of the latch IC to comprise an output port.

No change takes place in flags.

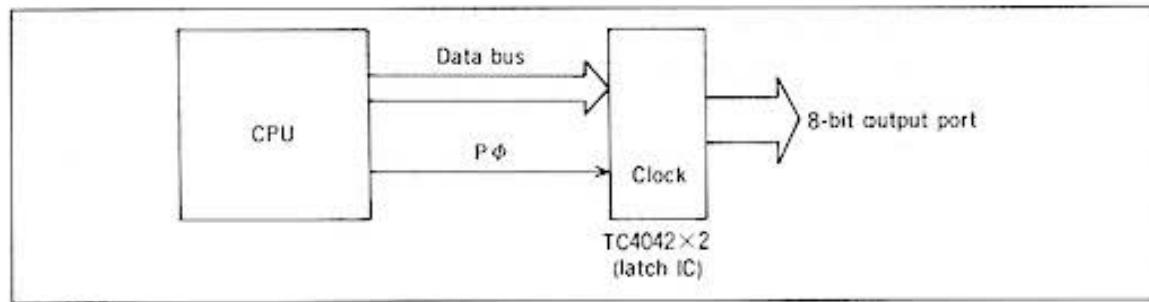
•Format

ATP

•Operation



NOTE: Though data is output with high state of OD (output disable) during memory write, OD is low state in the case of the ATP instruction.

**⑥ SPU (Set PU)**

Sets the general purpose flipflop PU.

No change takes place in flags.

•Format

SPU

•Operation

1 → PU

⑦ RPU (Reset PU)

Resets the general purpose flipflop PU.

No change takes place in flags.

•Format

RPU

•Operation

0 → PU

⑧ SPV (Set PV)

Sets the general purpose flipflop PV.

No change takes place in flags.

•Format

SPV

•Operation

1 → PV

⑨ RPV (Reset PV)

Resets the general purpose flipflop PV.

No change takes place in flags.

•Format

RPV

•Operation

0 → PV

⑩ SDP (Set DisP)

Sets the LCD on/off control flipflop DISP.

•Format

SDP

•Operation

1 → DISP

On pattern signal is generated from the CPU internal LCD backplate signal lines (H0~H7).

(31) RDP (Reset DisP)

Resets the LCD on/off control flipflop DISP.

•Format

RDP

•Operation

0 → DISP

Off pattern signal is generated from the CPU internal LCD backplate signal lines (H0~H7).

(32) ITA (In To Acc)

The contents of the input port (IN0~IN7) are transferred to the accumulator.
Only the flag Z changes.

•Format

ITA

•Operation

IN0~7 → Accumulator

(33) SIE (Set IE)

Sets the interrupt enable flag IE. After this, it becomes ready for maskable interrupt and timer interrupt acknowledge.
No change takes place in other flags.

•Format

SIE

•Operation

I → IE

(34) RIE (Reset IE)

Resets the interrupt enable flag IE. After this, maskable interrupt and timer interrupt are disabled.
No change takes place in other flags.

•Format

RIE

•Operation

0 → IE

(35) AM0 (Acc to Tm and 0)

The contents of the accumulator are transferred to the timer register (TM). Since the timer register consists of 9 bits, the accumulator contents are transferred to the low order 8 bits of the register and "0" is entered in the highest order bit.
No change takes place in other flags.

•Format

AM0

•Operation

A → TM (TM0~TM7)

0 → TM8

⑥ AM1 (Acc to Tm and 1)

Same as AM0, but "1" is entered in the highest order bit.

No change takes place in other flags.

•Format

AM1

•Operation

A → TM (TM0~TM7)

I → TM8

⑦ NOP (No Operation)**⑧ HLT (HaLT)**

Stops CPU operation. (Only the divider is in operation.)

Released from stop by interrupt.

No change takes place in flags.

⑨ OFF

BF flipflop reset instruction.

No change takes place in flags.

2-4-8. Jump instructions

⑩ JMP (JuMP)

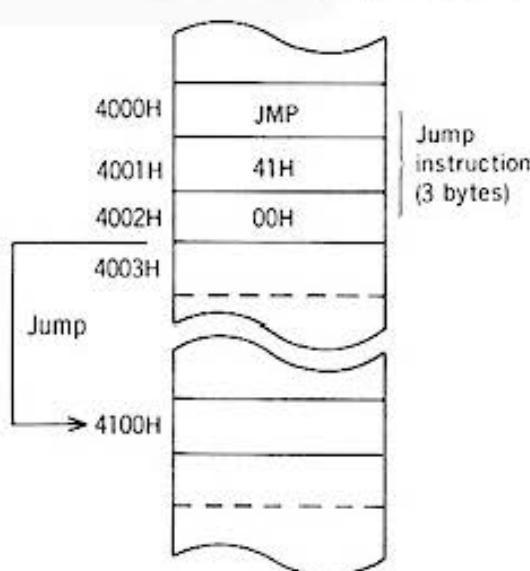
Jumps to a new program step represented by the second and third bytes of the immediate data *i, j*.

No change takes place in flags.

•FormatJMP *i, j***•Operation***i* → PH*j* → PL

(EX) JMP 41H,00H

PH=40H Execute → PH=41H
PL=03H → PL=00H



(6) BCH (BranCH)

Jumps to a new program step which is indicated by the program counter of which value is added/subtracted by the value of the immediate data i, j . It will be possible to jump within a range of $-255 < i < 255$.

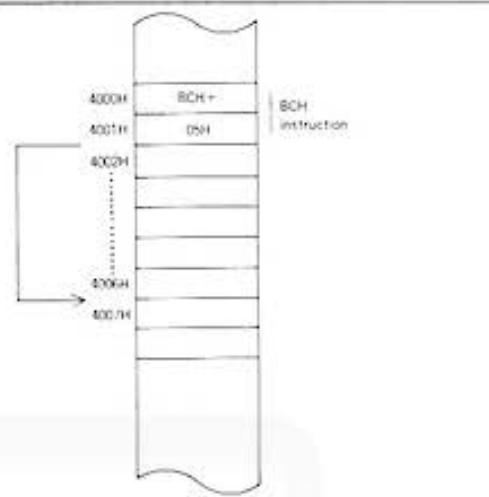
No change takes place in flags.

•FormatBCH+*i*BCH-*i***•Operation** $P + i \rightarrow P$ $P - i \rightarrow P$

(EX) BCH+5

Execute
 $P=4002H \longrightarrow P=4007H$

As soon as the BCH instruction is read, the program counter becomes "P=4002H", and after execution of the BCH instruction, the next program execution starts from the address 4007H.

**(6) BCS (Branch if C Set)**

Conditional relative address jump.

When C=1, it jumps to a program step represented by the program counter of which value is added/subtracted with the value of the immediate data.

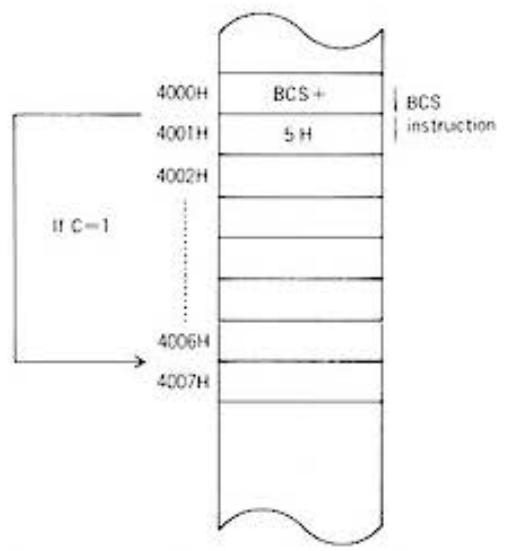
If C=0, the control proceeds directly to a next program step without causing jump.
 No change takes place in flags.

•FormatBCS+*i*BCS-*i***•Operation**
 $C = \begin{cases} 1: P + i \rightarrow P \\ 0: P \text{ not changed.} \end{cases}$
 $C = \begin{cases} 1: P - i \rightarrow P \\ 0: P \text{ not changed.} \end{cases}$

(EX) BCS+5

Execute
 $P=4002H \longrightarrow P=4007H$
 $C=1$

As soon as the BCS instruction is read, the program counter becomes "P=4002H". If C=1 at that point, next program execution starts from the address 4007H. If C=0, it starts from 4002H.



⑥③ BCR (Branch if C Reset)

If C=0, it jumps by relative address.
If C=1, it executes the next program step.
No change takes place in flags.

•Format

BCR+*i*
BCR-*i*

⑥④ BHS (Branch if H Set)

If H=1, it jumps by relative address.
If H=0, it executes the next program step.
No change takes place in flags.

•Format

BHS+*i*
BHS-*i*

⑥⑤ BHR (Branch if H Reset)

If H=0, it jumps by relative address.
If H=1, it executes the next program step.
No change takes place in flags.

•Format

BHR+*i*
BHR-*i*

⑥⑥ BZS (Branch if Z Set)

If Z=1, it jumps by relative address.
If Z=0, it executes the next program step.
No change takes place in flags.

•Format

BZS+*i*
BZS-*i*

⑥⑦ BZR (Branch if Z Reset)

If Z=0, it jumps by relative address.
If Z=1, it executes the next program step.
No change takes place in flags.

•Format

BZR+*i*
BZR-*i*

⑥⑧ BVS (Branch if V Set)

If V=1, it jumps by relative address.
If V=0, it executes the next program step.
No change takes place in flags.

•Format

BVS+*i*
BVS-*i*

(69) BVR (Branch if V Reset)

If $V=0$, it jumps by relative address.
 If $V=1$, it executes the next program step.
 No change takes place in flags.

•FormatBVR+*i*BVR-*i***(70) LOP (LOoP)**

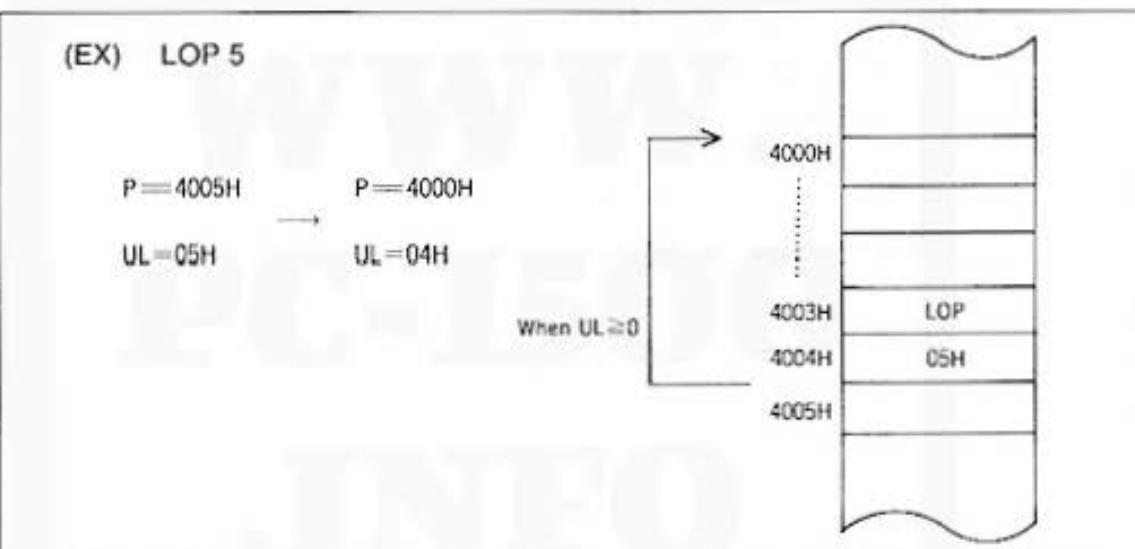
If borrow is not produced after subtracting "1" from the UL register, the program counter is subtracted by the immediate data, then it jumps to relative address for next program execution.

If there is borrow, ($UL < 0$), it proceeds to the succeeding program step.

No change takes place in flags.

•FormatLOP *i***•Operation** $UL - 1 \rightarrow UL$

$$\text{Borrow} = \begin{cases} 0: P - i \rightarrow P \\ 1: \text{To succeeding step} \end{cases}$$



2-4-9. Subroutine jump instructions

⑦ SJP (Subroutine JumP)

The contents of the program counter, which show the next program executing address, are stored in the stack pointer, then the control jumps to the subroutine address represented by i and j of the 16-bit immediate data.

No change takes place in flags.

•Format

SJP i, j

•Operation

PL \rightarrow (S)

S - 1

PH \rightarrow (S)

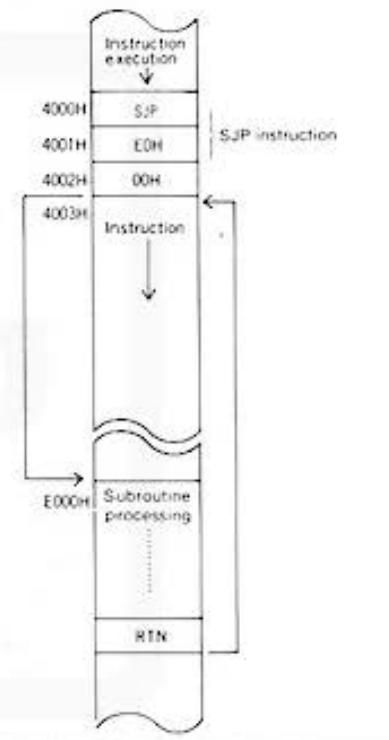
S - 1

$i \rightarrow$ PH

$j \rightarrow$ PL

(EX) SJP E0H, 00H

4003H is stored in the stack pointer by the execution of STP instruction and jumps to the subroutine of E000H. With the RTN instruction of the subroutine, the control returns to 4003H to resume previous program execution.



⑦ VEJ (VEctor subroutine Jump)

One step subroutine jump instruction that jumps to the address indicated by the two-byte vector, whose high order address byte is represented by FFH and low order address byte by the operand of the instruction.

The flag Z is reset.

There are 28 kinds of VEJ operand within two bytes range of 11000000 (COH) to 11110110 (F6H). Therefore, the vector address table contains the address area of FFC0H to FFF6H.

- Format

VEL *i*

• Operation

PL - (S)

$$S - 1 \rightarrow S$$

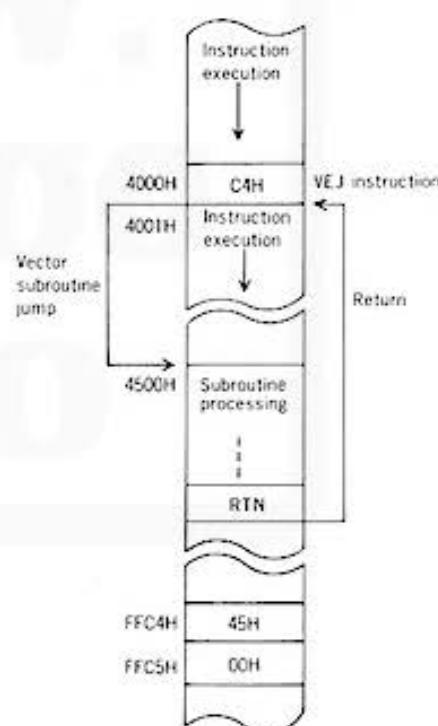
PH - (S)

$$S - 1 \rightarrow S$$

(FF00H+*i*) → PH

(EX) VELCAH

The contents of FFC4H and FFC5H are interrogated at the address 4000H and jump to the subroutine of 4500H. With the RTN instruction of the subroutine, the execution resumes from the address 4001H again. Because it permits subroutine jump by a single byte of instruction, it helps to reduce program steps, when applied to too often appearing subroutine.



⑦ VMJ (Vector 2 byte Subroutine Jump)

Jumps to the address indicated by the two-byte vector address, whose high order address byte is represented by FFH and low order address byte by the immediate data. The flag Z is reset.

Vector address table contains FF00H thru FFF6H. Immediate value *i* may take even number of 00H thru F6H.

•FormatVMJ *i***•Operation**

PL → (S)

S - 1 → S

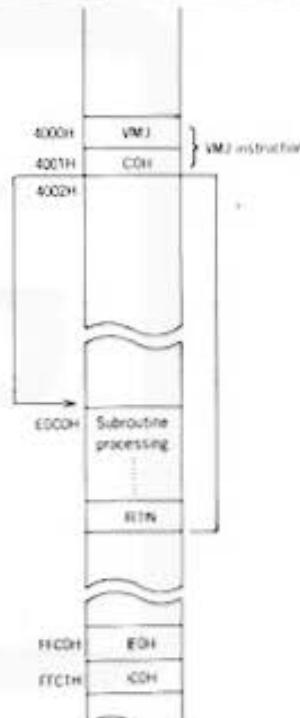
PH → (S)

S - 1 → S

(FF00H+*i*) → PH(FF00H+*i*+1) → PL

(EX) VMJ C0H

The contents of FFC0H are interrogated by the VMJ instruction and jump to the subroutine of E0C0H.
 With the RTN instruction of the subroutine, the execution resumes from the address 4002H again.
 Because it permits subroutine jump by two bytes of instruction, it helps to reduce program steps, as compared with the SJP instruction.



(4) VCS (Vector subroutine jump if C Set)

Conditional vector subroutine jump instruction.

If C=1, it performs the vector subroutine jump, the same as in the VMJ instruction.

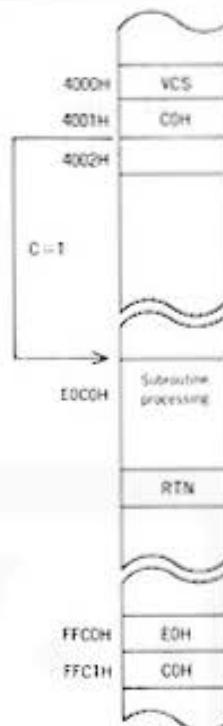
If C=0, the control proceeds to the succeeding program step.

•Format

VCS *i*

(EX) VCS C0H

The contents of C are interrogated by the VCS instruction and perform the vector subroutine jump, if C=1.
If C=0, it continues to execute the program in the address 4002H.

**(5) VCR (Vector subroutine jump if C Reset)**

If C=0, it performs the vector subroutine jump, the same as in the VMJ instruction.

If C=1, the control proceeds to the succeeding program step.

•Format

VCR *i*

(6) VHS (Vector subroutine jump if H Set)

If H=1, it performs the vector subroutine jump, the same as in the VMJ instruction.

If H=0, the control proceeds to the succeeding program step.

•Format

VHS *i*

(7) VHR (Vector subroutine jump if H Reset)

If H=0, it performs the vector subroutine jump, the same as in the VMJ instruction.

If H=1, the control proceeds to the succeeding program step.

•Format

VHR *i*

(8) VZS (Vector subroutine jump if Z Set)

If Z=1, it performs the vector subroutine jump, the same as in the VMJ instruction.

If Z=0, the control proceeds to the succeeding program step.

•Format

VZS *i*

(79) VZR (Vector subroutine jump if Z Reset)

If Z=0, it performs the vector subroutine jump, the same as in the VMJ instruction.
If Z=1, the control proceeds to the succeeding program step.

•FormatVZR *i***(80) VVS (Vector subroutine jump if V Set)**

If V=1, it performs the vector subroutine jump, the same as in the VMJ instruction.
If V=0, the control proceeds to the succeeding program step.

•FormatVVS *i*

2-4-10. Return instructions

(81) RTN (ReTurN from subroutine)

The instruction used to return from the subroutine to the main routine.
No change takes place in flags.

The previous program address is gotten from the external memory stack to be transferred to the program counter.

The next instruction will be fetched from the address indicated by the program counter.

•Format

RTN

•Operation
 $S + 1 \rightarrow S$
 $(S) \rightarrow PH$
 $S + 1 \rightarrow S$
 $(S) \rightarrow PL$
(82) RTI (ReTurn from Interrupt)

The instruction used to return from the interrupt service routine to the main routine.
After executing the same procedure as in the RTN instruction, then the contents of the T register at the time of interrupt are gotten from the external memory stack to be transferred to the T register. Flags are also set to their previous states.

•Format

RTI

•Operation
 $S + 1 \rightarrow S$
 $(S) \rightarrow PH$
 $S + 1 \rightarrow S$
 $(S) \rightarrow PL$
 $S + 1 \rightarrow S$
 $(S) \rightarrow T$

2-5. Command list

List of LH5801 Microprocessor will be shown in pages to follow. There are following nine types of commands.

Single byte command

(1)

op code

Two-byte command

(2)

1 1 1 1 1 1 0 1	op code
-----------------	---------

(3)

op code	immediate
---------	-----------

(i)

Three-byte command

(4)

1 1 1 1 1 1 0 1	op code	immediate
-----------------	---------	-----------

(i)

(5)

op code	immediate H	immediate L
---------	-------------	-------------

(i) 16 bits (j)

(6)

op code	address H	address L
---------	-----------	-----------

(a) (b)

Four-byte command

(7)

1 1 1 1 1 1 0 1	op code	address H	address L
-----------------	---------	-----------	-----------

(a) (b)

(8)

op code	address H	address L	immediate
---------	-----------	-----------	-----------

(a) (b) (i)

Five-byte command

(9)

1 1 1 1 1 1 0 1	op code	address H	address L	immediate
-----------------	---------	-----------	-----------	-----------

(a) (b) (i)

8-bit CPU command list (1)

Arithmetic/logical

MNEMONIC	SYMBOLIC OPERATION	STATUS CVH ZIE	MACHINE LANGUAGE 7 6 5 4 3 2 1 0	BYTE	CYCLE	COMMENT			
						R _L	R _H		
ADC	R _L	0000-	00R _L 0010	1	6	● R5 R4 R _L R _H R	0 0 X _L X _H X 0 1 Y _L Y _H Y 1 0 U _L U _H U 1 1 * * *		
	R _H		10R _H 0010	1	6				
	(R)		00R 0011	1	7				
	(a,b)		10100011	3	13				
	#(R)	A + # (R) + C → A	FD	2	11				
	#(a,b)	A + #(a,b) + C → A	00R 0011	4	17				
	#(a,b)	A + #(a,b) + C → A	FD 10100011						
ADI	A,i	A + i + C → A	10110011	2	7	● Address of (a,b) 15 ----- 87 ----- 0 <table border="1"><tr><td>a</td><td>b</td></tr></table> (High order) (Low order)	a	b	● (R)…MEO accessed #(R)…ME1 accessed
a	b								
(R),i	(R) + i → (R)	01R 1111	2	13					
(a,b),i	(a,b) + i → (a,b)	11101111	4	19					
#(R),i	#(R) + i → #(R)	01 R 1111	3	17					
#(a,b),i	#(a,b) + i → #(a,b)	FD 11101111	5	23					
DCA	(R)	A + (R) + C → A (BCD)	10 R 1100	1	15				
	#(R)	A + #(R) + C → A (BCD)	FD	2	19				
ADR	R _L + A → R _L		10 R 1100						
	(16-bit register operation)		FD	2	11				
	RH + 1 → RH + i / C7		11 R 1010						
SBC	R _L	A - R _L - C → A	0000-	00R _L 0000	1	6			
	R _H	A - R _H - C → A		10R _H 0000	1	6			
	(R)	A - (R) - C → A		00R 0001	1	7			
	(a,b)	A - (a,b) - C → A		10100001	3	13			
	#(R)	A - #(R) - C → A		00R 0001	2	11			
	#(a,b)	A - #(a,b) - C → A		10100001	4	17			
SBI	A,i	A - i - C → A	10110001	2	7				
DCS	(R)	A - (R) - C → A (BCD)	00R 1100	1	13				
	#(R)	A - #(R) - C → A (BCD)	FD 00R 1100	2	17				
AND	(R)	A ∧ (R) → A	----○-	00R 1001	1	7			
	(a,b)	A ∧ (a,b) → A		10101001	3	13			
	#(R)	A ∧ #(R) → A		00R 1001	2	11			
	#(a,b)	A ∧ #(a,b) → A		10101001	4	17			
ANI	A,i	A ∧ i → A	10111001	2	7				
	(R),i	(R) ∧ i → (R)	01R 1001	2	13				
	(a,b),i	(a,b) ∧ i → (a,b)	11101001	4	19				
	#(R),i	#(R) ∧ i → #(R)	01R 1001	3	17				
	#(a,b),i	#(a,b) ∧ i → #(a,b)	11101001	5	23				

8-bit CPU command list (2)

MNEMONIC	SYMBOLIC OPERATION	STATUS	MACHINE LANGUAGE												COMMENT	
			C	V	H	Z	IE	7	6	5	4	3	2	1	0	
ORA	(R) A \vee (R) \rightarrow A	----○--	0	0	R	1	0111									
	(a,b) A \vee (a,b) \rightarrow A						10101011									
	#(R) A \vee #(R) \rightarrow A						00 R 1011									
	#(a,b) A \vee #(a,b) \rightarrow A						10101011									
ORI	A,i A \vee i \rightarrow A	10111011														
	(R),i (R) \vee i \rightarrow (R)						01 R 1011									
	(a,b),i (a,b) \vee i \rightarrow (a,b)						11101011									
	#(R),i #(R) \vee i \rightarrow #(R)						01 R 1011									
#(a,b),i #(a,b) \vee i \rightarrow #(a,b)							11101011									
EOR	(R) A \oplus (R) \rightarrow A	---○---	0	0	R	1	1011									
	(a,b) A \oplus (a,b) \rightarrow A						10101101									
	#(R) A \oplus #(R) \rightarrow A						00 R 1101									
	#(a,b) A \oplus #(a,b) \rightarrow A						10101101									
EAI	i A \oplus i \rightarrow A	10111101														
INC	A A + 1 \rightarrow A	0000-	1	1	0	1	11011101									
	R _L R _L + 1 \rightarrow R _L	01 R _L 0000														
	R _H R _H + 1 \rightarrow R _H	01 R _H 0000														
	R R + 1 \rightarrow R	01 R 0100														
DEC	A A - 1 \rightarrow A	0000-	1	1	0	1	11011111									
	R _L R _L - 1 \rightarrow R _L	01 R _L 0010														
	R _H R _H - 1 \rightarrow R _H	01 R _H 0010														
	R R - 1 \rightarrow R	01 R 0110														

Compare and bit test

- CPA	R _L A - R _L	0000-	0	0	R _L	0	1110									
	R _H A - R _H		1	0	R _H	0	1110									
	(R) A - (R)		0	0	R	0	1111									
	(a,b) A - (a,b)		1	0	1001111											
	#(R) A - #(R)		0	0	R 0111											
	#(a,b) A - #(a,b)		1	0	10100111											
CPI	R _L ,i R _L - i	01 R _L 1110														
	R _H ,i R _H - i	01 R _H 1100														
	A,i A - i	10110111														
BIT	(R) A \wedge (R) \rightarrow Z	---○--	0	0	R	1	1111									
	(a,b) A \wedge (a,b) \rightarrow Z		1	0	10101111											
	#(R) A \wedge #(R) \rightarrow Z		0	0	R 1111											
	#(a,b) A \wedge #(a,b) \rightarrow Z		1	0	10101111											
BL	A,i A \wedge i \rightarrow Z	10111111														
	(R),i (R) \wedge i \rightarrow Z	01 R 1101														
	(a,b),i (a,b) \wedge i \rightarrow Z	11101101														
	#(R),i #(R) \wedge i \rightarrow Z	01 R 1101														
	#(a,b),i #(a,b) \wedge i \rightarrow Z	11101101														

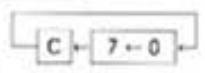
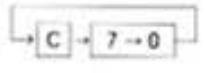
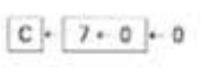
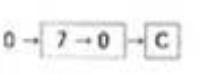
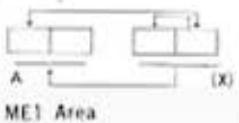
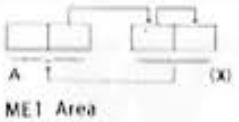
8-bit CPU command list (3)**Load and store**

MNEMONIC	SYMBOLIC OPERATION	STATUS C V H Z IE	MACHINE LANGUAGE 7 6 5 4 3 2 1 0	BYTE	CYCLE	COMMENT	
						1	5
LDA R _L	R _L → A	----○-	0 0 R _L 0 1 0 0			1	5
R _H	R _H → A		1 0 R _H 0 1 0 0			1	5
(R)	(R) → A		0 0 R 0 1 0 1			1	6
(a,b)	(a,b) → A		1 0 1 0 0 1 0 1			3	12
#(R)	#(R) → A		0 0 R 0 1 0 1 FD			2	10
#(a,b)	#(a,b) → A		1 0 1 0 0 1 0 1 FD			4	16
LDE R	(R) → A, R-1 → R		0 1 R 0 1 1 1			1	6
LIN R	(R) → A, R+1 → R		0 1 R 0 1 0 1			1	6
LDI R _{L,i}	i → R _L	-----	0 1 R _L 1 0 1 0			2	6
R _{H,i}	i → R _H		0 1 R _H 1 0 0 0			2	6
A _{,i}	i → A	----○-	1 0 1 1 0 1 0 1			2	6
S _{,i,j}	ij → S	-----	1 0 1 0 1 0 1 0			3	12
LDX R	R → X		0 0 R 1 0 0 0 FD			2	11
S	S → X		0 1 0 0 1 0 0 0 FD			2	11
P	P → X		0 1 0 1 1 0 0 0 FD			2	11
STA R _L	A → R _L	-----	0 0 R _L 1 0 1 0			1	5
R _H	A → R _H		0 0 R _H 1 0 0 0			1	5
(R)	A → (R)		0 0 R 1 1 1 0			1	6
(a,b)	A → (a,b)		1 0 1 0 1 1 1 0			3	12
#(R)	A → #(R)		0 0 R 1 1 1 0 FD			2	10
#(a,b)	A → #(a,b)		1 0 1 0 1 1 1 0 FD			4	16
SDE R	A → (R), R-1 → R		0 1 R 0 0 1 1			1	6
SIN R	A → (R), R+1 → R		0 1 R 0 0 0 1 FD			1	6
STX R	X → R		0 1 R 1 0 1 0 FD			2	11
S	X → S		0 1 0 0 1 1 1 0 FD			2	11
P	X → P		0 1 0 1 1 1 1 0 FD			2	11
PSH A	A → (S), S-1 → S		1 1 0 0 1 0 0 0 FD			2	11
R	R _L → (S), RH → (S-1), S-2 → S		1 0 R 1 0 0 0 FD			2	14
POP A	(S+1) → A, S+1 → S	----○-	1 0 0 0 1 0 1 0 FD			2	12
R	(S+1) → RH, S+2 → RL, S+2 → S	-----	0 0 R 1 0 1 0 FD			2	15
ATT	A → T (STATUS)	00000	1 1 1 0 1 1 0 0 FD			2	9
TTA	T (STATUS) → A	----○-	1 0 1 0 1 0 1 0 FD			2	9

Block transfer and search

TIN	(X) → (Y), X+1 → X Y+1 → Y	-----	1 1 1 1 0 1 0 1		1	7	
CIN	A → (X), X+1 → X	0000-	1 1 1 1 0 1 1 1		1	7	

8-bit CPU command list (4)**Rotate and shift**

MNEMONIC	S Y M B O L I C O P E R A T I O N	S T A T U S	M A C H I N E L A N G U A G E							B Y T E	C Y C L E	C O M M E N T		
			C	V	H	Z	I	E	7	6	5	4	3	2
ROL		0000							11011011			1	8	
ROR									11010001			1	9	
SHL									11011001			1	6	
SHR									11010101			1	9	
DRL	 A T (X)	-----							11010111			1	12	
DRL #	ME1 Area								11010111	FD		2	16	
DRR	 A T (X)								11010011			1	12	
DRR #	ME1 Area								11010011	FD		2	16	
AEX									11110001			1	6	

CPU control

AM0	A → TIMER(T0~T7).0 → T8	-----	11001110			2	9							
AM1	1 → T8		11011110	FD		2	9							
CDV	divider clear		10001110	FD		2	8							
ATP	A → Output port (Clock output)		11001100	FD		2	9							
SDP	I → Disp		11000001	FD		2	8							
RDP	0 → Disp		11000000	FD		2	8							
SPU	1 → PU		11100001			1	4							
RPU	0 → PU		11100011			1	4							
SPV	1 → PV		10101000			1	4							
RPV	0 → PV		10111000			1	4							
ITA	IN → A	---○---	10111010	FD		2	9							
RIE	0 → IE	----○-	10111110	FD		2	8							
SIE	I → IE	----○-	10000001	FD		2	8							
HLT		-----	10110001	FD		2	9							
OFF			01001100	FD		2	8							
NOP			00111000			1	5							
SEC	1 → C	○-----	11111011			1	4							
REC	0 → C	○-----	11111001			1	4							

8-bit CPU command list (5)**Jump**

MNEMONIC	SYMBOLIC OPERATION	STATUS C V H Z IE	MACHINE LANGUAGE 7 6 5 4 3 2 1 0	BYTE	CYCLE	COMMENT	
						8/10/11	8/12/11
JMP	$ij \rightarrow P$	-----	1 0 1 1 1 0 1 0	3	12		$s = 0; +i$
BCH	$s = 0; P \pm i \rightarrow P$		1 0 0 s 1 1 1 0	2	8	$s = 1; -i$	
	$s = 1; P \pm i \rightarrow P$				9		(Includes one more cycle)
BCS	$if C=1, P \pm i \rightarrow P$		1 0 0 s 0 0 1 1	2	8/10/11		
	$if C=0, continue$						
BCR	$if C=0, P \pm i \rightarrow P$		1 0 0 s 0 0 0 1	2	8/10/11	8/12/11	Condition
	$if C=1, continue$					0 0 0	NC: non carry
BVS	$if V=1, P \pm i \rightarrow P$		1 0 0 s 1 1 1 1	2	8/10/11	0 0 1	C: carry
	$if V=0, continue$					0 1 0	NH: non half
BVR	$if V=0, P \pm i \rightarrow P$		1 0 0 s 1 1 0 1	2	8/10/11	0 1 1	H: halt
	$if V=1, continue$					1 0 0	NZ: non zero
BHS	$if H=1, P \pm i \rightarrow P$		1 0 0 s 0 1 1 1	2	8/10/11	1 0 1	NV: zero
	$if H=0, continue$					1 1 0	NV: non overflow
BHR	$if H=0, P \pm i \rightarrow P$		1 0 0 s 0 1 0 1	2	8/10/11	1 1 1	V: overflow
	$if H=1, continue$						
BZS	$if Z=1, P \pm i \rightarrow P$		1 0 0 s 1 0 1 1	2	8/10/11		
	$if Z=0, continue$						
BZR	$if Z=0, P \pm i \rightarrow P$		1 0 0 s 1 0 0 1	2	8/10/11		
	$if Z=1, continue$						
LOP UL,i	$UL \rightarrow UL$ $if Borrow=0, P \pm i \rightarrow P$ $if Borrow=1, continue$		1 0 0 0 1 0 0 0	2	8/11		

Call

SJP	$P_L \rightarrow (S), P_H \rightarrow (S-1),$ $S-2 \rightarrow S, ij \rightarrow P$	-----	1 0 1 1 1 1 1 0	3	19	• Vector address (q)
VEJ	$P_L \rightarrow (S), PH \rightarrow (S-1)$ $S-2 \rightarrow S, (q) \rightarrow P_H, (q+1) \rightarrow P_L$	---○---	1 1 ← i → 0	1	17	VEJ: FF → qH 11/0 → qL
VCS	$if C=1, (q) \rightarrow P_H \rightarrow (S-1)$ $(q+1) \rightarrow P_L \rightarrow (S), S-2 \rightarrow S$		1 1 0 0 0 0 1 1	2	8/21	VMJ: FF → qH etc. i → qL
VCR	$if C=0,$		1 1 0 0 0 0 0 1	2	8/21	
VHS	$if H=1,$		1 1 0 0 0 1 1 1	2	8/21	
VHR	$if H=0,$		1 1 0 0 0 1 0 1	2	8/21	
VZS	$if Z=1,$		1 1 0 0 1 0 1 1	2	8/21	
VZR	$if Z=0,$		1 1 0 0 1 0 0 1	2	8/21	
VVS	$if V=1,$		1 1 0 0 1 1 1 1	2	8/21	
VMJ	$(q) \rightarrow P_H \rightarrow (S-1), S-2 \rightarrow S$ $(q+1) \rightarrow P_L \rightarrow (S)$		1 1 0 0 1 1 0 1	2	20	

Return

RTN	$(S+1) \rightarrow PH, (S+2) \rightarrow PL,$ $S+2 \rightarrow S$	-----	1 0 0 1 1 0 1 0	1	11	
RTI	$(S+1) \rightarrow PH, (S+2) \rightarrow PL$ $(S+3) \rightarrow T, S+3 \rightarrow S$	○○○○○	1 0 0 0 1 0 1 0	1	14	

NOTE: P in above list indicates a succeeding byte. For a command accompanying the immediate value, it indicates the byte that follows to the immediate value.

LH5801

MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE
ADC	XL	02	ANI	(ab)	E9 a b i	BVR	-	9D i
	YL	12		#(X)	FD 49 i		+	88 i
	UL	22		#(Y)	FD 59 i		-	9B i
	XH	82		#(U)	FD 69 i	BZR	+	89 i
	YH	92		#(ab)	FD E9 a b i		-	99 i
	UH	A2		AM0	FD CE	CDV		FD 8E
	(X)	03	AM1		FD DE		CIN	F7
	(Y)	13		ATP	FD CC		CPA	XL 06
	(U)	23	ATT		FD EC		YL	16
	(ab)	A3 a b	BCH	+	8E i		UL	26
	#(X)	FD 03		-	9E i		XH	86
	#(Y)	FD 13	BCS	+	83 i		YH	96
	#(U)	FD 23		-	93 i		UH	A6
	#(ab)	FD A3 a b	BCR	+	81 i		(X)	07
				-	91 i		(Y)	17
ADI	A	B3 i	BHS	+	87 i		(U)	27
	(X)	4F i		-	97 i		(ab)	A7 a b
	(Y)	5F i	BHR	+	85 i		#(X)	FD 07
	(U)	6F i		-	95 i		#(Y)	FD 17
	(ab)	EF a b i	BII	A	BF i		#(U)	FD 27
	#(X)	FD 4F i		(X)	4D i		#(ab)	FD A7 a b
	#(Y)	FD 5F i		(Y)	5D i	CPI	A	87 i
	#(U)	FD 6F i		(U)	6D i		XL	4E i
	#(ab)	FD EF a b i		(ab)	ED a b i		YL	5E i
ADR	X	FD CA	#(X)	FD 4D i			UL	6E i
	Y	FD DA		#(Y)	FD 5D i		XH	4C i
	U	FD EA		#(U)	FD 6D i		YH	5C i
AEX		F1	BIT	#(ab)	FD ED a b i		UH	6C i
				(X)	0F	DCA	(X)	8C
AND	(X)	09		(Y)	1F		(Y)	9C
	(Y)	19		(U)	2F		(U)	AC
	(U)	29		(ab)	AF a b		#(X)	FD 8C
	(ab)	A9 a b		#(X)	FD 0F		#(Y)	FD 9C
	#(X)	FD 09		#(Y)	FD 1F		#(U)	FD AC
	#(Y)	FD 19		#(U)	FD 2F	DCS	(X)	0C
	#(U)	FD 29		#(ab)	FD AF a b		(Y)	1C
	#(ab)	FD A9 a b	BVS	(X)	8F i		(U)	2C
ANI	A	B9 i		(Y)	9F i		#(X)	FD 0C
	(X)	49 i		(U)	8D i		#(Y)	FD 1C
	(Y)	59 i						
	(U)	69 i	BVR					

MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE
DCS	#(U)	FD 2C	LDA	UL	24	ORA	#(Y)	FD 1B
DEC	A	DF	LDA	XH	84	ORI	#(U)	FD 2B
	XL	42		YH	94		#(ab)	FD AB a b
	YL	52		UH	A4		A	BB i
	UL	62		(X)	05		(X)	4B i
	XH	FD 42		(Y)	15		(Y)	5B i
	YH	FD 52		(U)	25		(U)	6B i
	UH	FD 62		(ab)	A5 a b		(ab)	EB a b i
	X	46		#(X)	FD 05		#(X)	FD 4B
	Y	56		#(Y)	FD 15		#(Y)	FD 5B
	U	66		#(U)	FD 25		#(U)	FD 6B
DRL	(X)	D7	LDI	#(ab)	FD A5 a b	POP	#(ab)	FD EB a b i
	#(X)	FD D7		A	B5 i		A	FD 8A
DRR	(X)	D3		XL	4A i		X	FD 0A
	#(X)	FD D3		YL	5A i		Y	FD 1A
EAI		BD i		UL	6A i		U	FD 2A
EOR	(X)	0D		XH	48 i	PSH	A	FD C8
	(Y)	1D		YH	58 i		X	FD 88
	(U)	2D		UH	68 i		Y	FD 98
	(ab)	AD a b		S	AA i j		U	FD A8
	#(X)	FD 0D	LDE	X	47		RDP	FD C0
#(Y)	FD 1D	Y		57	REC		F9	
#(U)	FD 2D	U		67	RIE		FD BE	
#(ab)	FD AD a b	X		FD 08	ROL		D8	
HLT		FD B1	LDX	Y	FD 18	ROR		D1
INC	A	DD		U	FD 28		RPU	E3
	XL	40		S	FD 48		RPV	B8
	YL	50		P	FD 58		RTI	8A
	UL	60		X	45		RTN	9A
	XH	FD 40		Y	55		SBC	XL 00
INC	YH	FD 50		U	65		YL	10
	UH	FD 60	LOP	UL	88 i		UL	20
	X	44		NOP			XH	80
	Y	54	OFF		38		YH	90
	U	64			FD 4C		UH	A0
ITA		FD BA	ORA	(X)	0B	(U)	(X)	01
JMP		BA i j		(Y)	1B		(Y)	11
LDA	XL	04		(U)	2B		(U)	21
	YL	14		(ab)	AB a b		(ab)	A1 a b
				#(X)	FD 0B			

MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE
SBC	#(X)	FD 01	VEJ	TTA	FD AA	C0	C0	
	#(Y)	FD 11		VCS	C3 i			
	#(U)	FD 21		VCR	C1 i			
	#(ab)	FD A1 a b						
SBI		B1 i	C0	C2	C2	C4	C4	
SDE	X	43		C6	C6			
	Y	53	C8	C8	C8			
	U	63		CA	CA	CA	CA	
SDP		FD C1	CC	CC	CC			
SEC		FB		CE	CE	CE	CE	
SHL		D9		DO	DO			
SHR		D5		D2	D2			
SIE		FD 81	D4	D4	D4	D4	D4	
SIN	X	41		D6	D6			
	Y	51	D8	D8	D8			
	U	61		DA	DA	DA	DA	
SJP		BE i j	DC	DC	DC			
SPU		E1		DE	DE	DE	DE	
SPV		A8		EO	EO			
STA	XL	0A		E2	E2	E2	E2	
	YL	1A	E4	E4	E4			
	UL	2A		E6	E6	E6	E6	
	XH	08		E8	E8			
	YH	18	(EA)	EA	EA	EA	EA	
	UH	28		EC	EC			
	(X)	0E		EE	EE			
	(Y)	1E		FO	FO			
	(U)	2E	(F2)	F2	F2	F2	F2	
	(ab)	AE a b		F4	F4			
	#(X)	FD 0E		F6	F6			
	#(Y)	FD 1E		VMJ	CD i			
	#(U)	FD 2E	VVS			CF i	CF i	
	#(ab)	FD AE a b						
STX	X	FD 4A	V2S		CB i	CB i	CB i	
	Y	FD 5A	VZR		C9 i			
	U	FD 6A	VHR		C5 i			
	S	FD 4E	VHS		C7 i			
	P	FD 5E						
TIN		F5						

2-6. Electrical characteristics and timings

Absolute maximum ratings

Parameter	Symbol	Limits	Unit
Supply voltage	V _{CC}	-0.3 to +7	V
Input voltage	V _I	-0.3 to +7	V
Output voltage	V _{OUT}	-0.3 to +7	V
Operating temperature	T _{OPR}	0 to +40	°C
Storage temperature	T _{STG}	-55 to +150	°C

Electrical characteristics

DC characteristics

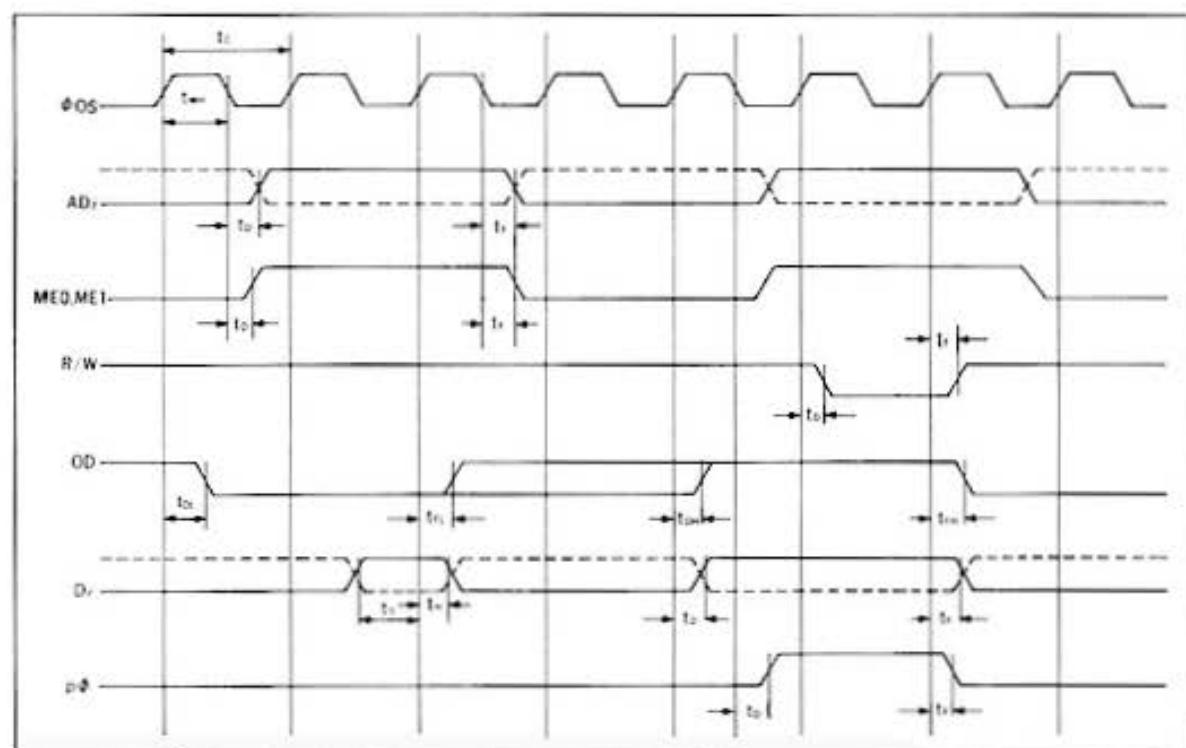
(Ta=0 to 40°C, V_{CC}=V_{GG}=4.5V ±0.5V)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Test conditions	Applicable pins
Supply current during operation	I _{CC}		7	15	mA	3.80MHz crystal in connection	
Supply current during halt	I _{SH}		4		mA		
Input voltage	V _H	V _{CC} -1.0			V		D0~7, BFI, RESET, HIN, WAIT, NMI, MI, IN0~7
	V _L		0.4		V		
Output voltage	V _{OH}	2.4			V	I _{OH} =400μA	AD0~15, OPF, BFO, R/W, OD, ME0~1, PΦ, PU, PV, ΦOS, HA, DISP, D0~7
	V _{OL}		0.4		V	I _{OL} =1.6mA	
Input current	I _{IL}		1.0		μA	V _H =V _{CC}	Input pins other than RESET, BFI
			5.0		μA		RESET, BFI
			1.0		μA	V _L =0	Input pins other than IN0~7, RESET, BFI
		30	60		μA		IN0~7
Power switch ON resistance	R _{SA}		300		Ω		VA
	R _{SB}		300		Ω		VB
LCD drive ON resistance	R _H		3.5		kΩ	H _i =V _{CC}	H0~7
	R _W		3.5		kΩ	H _i =V _W	
	R _L		5.0		kΩ	H _i =V _{DIS}	
Supply current during standby	I _{ST}		5		μA	V _{CC} =0V V _{GG} =5.5V	
3-state output leakage current	I _{OL}		1.0		μA		AD0~15, D0~7, R/W, ME0~1, OD

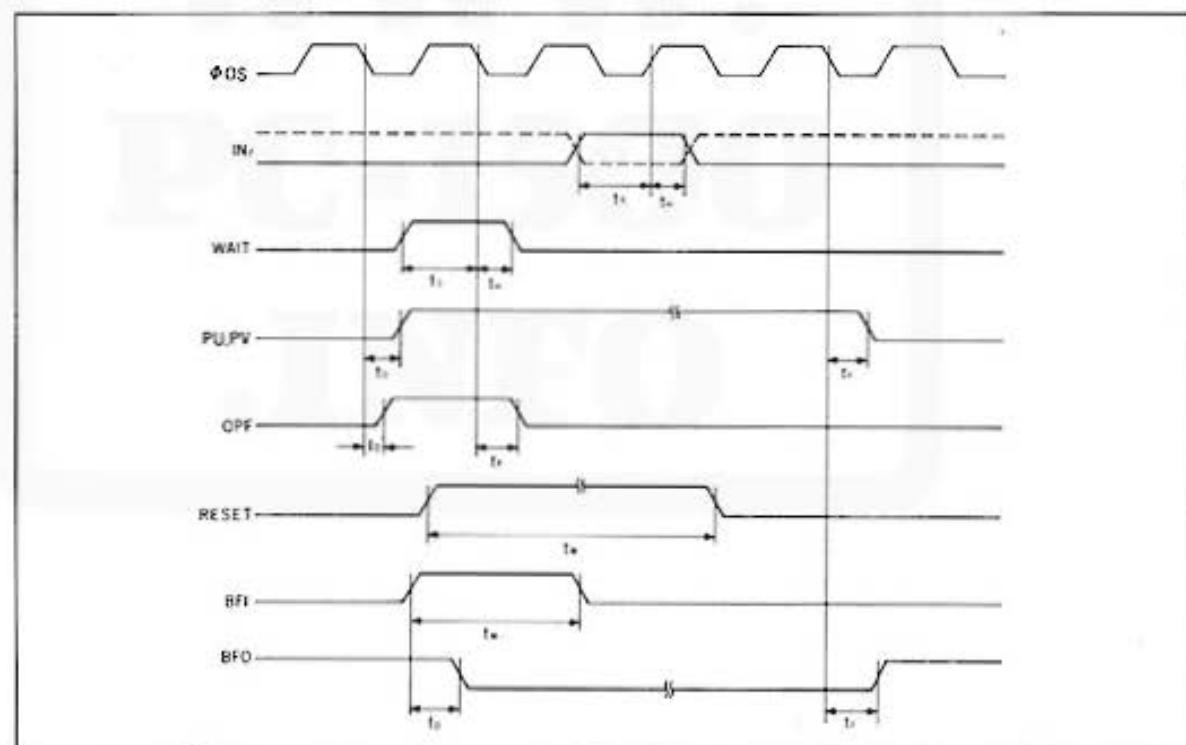
AC characteristics

Parameter	Symbol	Min.	Max.	Unit	Test conditions
Φ_{OS}	t_c	667			$C_L : 20\text{pF}$
	t_r		150		$t_r : (V_{cc} \times 0.1)\text{V} \rightarrow (V_{cc} \times 0.9)\text{V}$
	t_f		40		$t_f : (V_{cc} \times 0.9)\text{V} \rightarrow (V_{cc} \times 0.1)\text{V}$
	t_{sw}	220		nS	
AD0~15	t_b		250		$C_L : 20\text{pF}$
	t_s	80		nS	$V_{oh} : (V_{cc} \times 0.5)\text{V}$ $V_{ol} : 0.4\text{V}$
ME0,1	t_b		260		$C_L : 20\text{pF}$
	t_s	80	175	nS	$V_{oh} : (V_{cc} \times 0.5)\text{V}$ $V_{ol} : 0.4\text{V}$
OD	t_{bh}		250		$C_L : 20\text{pF}$
	t_{bi}	170			$V_{oh} : (V_{cc} \times 0.8)\text{V}$
	t_{bo}		370	nS	$V_{ol} : 0.4\text{V}$
	t_{bi}	100			
R/W	t_b	-30	50		$C_L : 20\text{pF}$
	t_s		200	nS	$V_{oh} : (V_{cc} \times 0.8)\text{V}$ $V_{ol} : 0.4\text{V}$
D0~7	t_b		600		$C_L : 20\text{pF}$
	t_s	220			$V_{oh} : (V_{cc} \times 0.8)\text{V}$
	t_{bh}	170			$V_{ol} : 0.4\text{V}$
	t_{bi}	100			
P ϕ	t_b		350		$C_L : 20\text{pF}$
	t_s		100	nS	$V_{oh} : (V_{cc} \times 0.8)\text{V}$ $V_{ol} : 0.4\text{V}$
IN0~7	t_b	190			$C_L : 20\text{pF}$
	t_s	30		nS	$V_{oh} : (V_{cc} - 1.0)\text{V}$ $V_{ol} : 0.4\text{V}$
WAIT	t_b	130			$C_L : 20\text{pF}$
	t_s	20		nS	$V_{oh} : (V_{cc} - 1.0)\text{V}$ $V_{ol} : 0.4\text{V}$
PU, PV	t_b		340		$C_L : 20\text{pF}$
	t_s	50		nS	$V_{oh} : (V_{cc} \times 0.8)\text{V}$ $V_{ol} : 0.4\text{V}$
OPF	t_b		310		$C_L : 20\text{pF}$
	t_s		190	nS	$V_{oh} : (V_{cc} \times 0.8)\text{V}$ $V_{ol} : 0.4\text{V}$
RESET	t_w	2		mS	$C_L : 20\text{pF}$
BFI	t_w	250		nS	$C_L : 20\text{pF}$
BFO	t_b		150		$C_L : 20\text{pF}$
	t_s		360	nS	$V_{oh} : (V_{cc} \times 0.8)\text{V}$ $V_{ol} : 0.4\text{V}$

Timing (1)



Timing (2)





3

LH5810/LH5811 I/O PORT CONTROLLER

3-1. Outline

The LH5810/LH5811 is a single chip CMOS static LSI that features the following functions:

- (1) two pairs of 8-bit bidirectional port
- (2) one pair of 8-bit output port
- (3) two interrupt request inputs (one of them port input)
- (4) one interrupt request output
- (5) CPU wait control
- (6) serial data transfer control

3-2. Functions

- ① Ports PA0~7 and PB0~7 can be programmed of their data flow direction in bit unit. Also, it can be accessed as one location of the memory, as seen from the CPU.
- ② Latch clock $P\phi$ can be directly given from the external source through output ports PC0~7. Also, it can be accessed as one location of the memory, as seen from the CPU.
- ③ As there are two interrupt request inputs of IRQ and PB7, interrupt request can be issued to the CPU at the rising edge of the input when the corresponding bit of the MSK register is "1". PB7 must be in the input mode before using PB7 for the interrupt input.
- ④ Since there is the CPU wait control circuit, two memory enable signals can be output to the memory of slow access time. Besides, it has two inputs of wait conditions. Up to 8 varieties of access time can be programmed.
- ⑤ It has the following functions to handle serial data transfer.

A. Serial data transmit

Serial data transfer takes place in a format of a start bit, 8 bits of data, and two stop bits. Transmission clock is selectable by means of internal and external clock select program, as well as the clock rate ($1/1, 1/2, 1/128, 1/256, 1/512, 1/1024, 1/2048$, and $1/4096$ of the basic clock).

B. Serial data receive

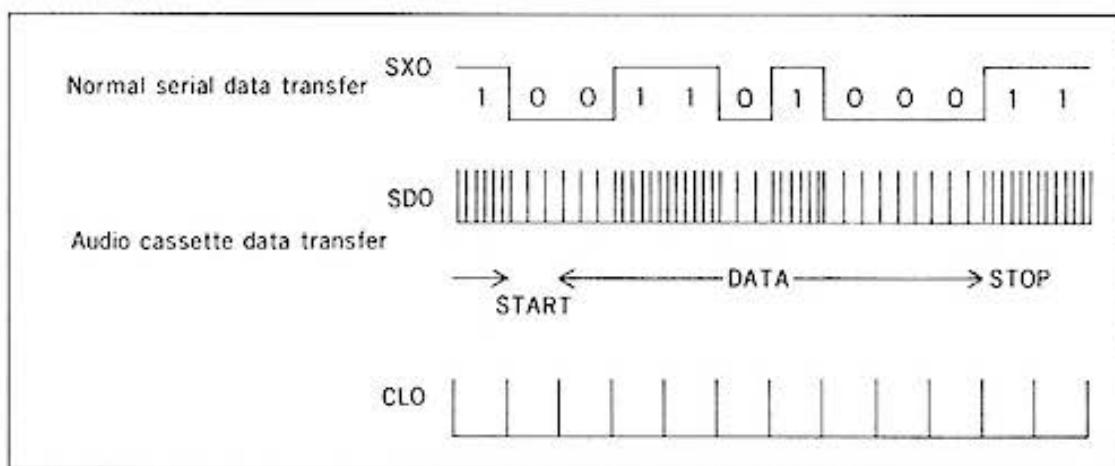
When the start bit is received in an idle state, the serial data following to it is received. After receiving the data comprised of 8 bits, it is then stored in the internal register and the interrupt request flag is set active. Receive clock is furnished from the external source which becomes the receiving clock by itself. It must be in synchronization with the serial data input.

C. Pulse waveform

It is possible to have continuous output of pulse waveform. Frequency is programmable to eight kinds of $1/1, 1/2, 1/128, 1/256, 1/512, 1/1024, 1/2048$, and $1/4096$ of the basic clock.

D. Data transfer to the audio cassette tape

Format of data transferred to the audio cassette tape consists of a start bit, 8 bits of data, and two stop bits, with the modulation signal generated from the SDO output.



Assume now that normal serial data output is to be SXO, the modulation clock to the data 1 to be FX, the modulation clock to the data 0 to be FY, and audio cassette tape data output to be SDO, then the following equation comprises:

$$SDO = SXO \cdot FX + \overline{SXO} \cdot FY$$

Whereas, FX and FY can be set independently to 1/64, 1/128, 1/256, 1/512 or 1/1024 of the basic clock by means of programming.

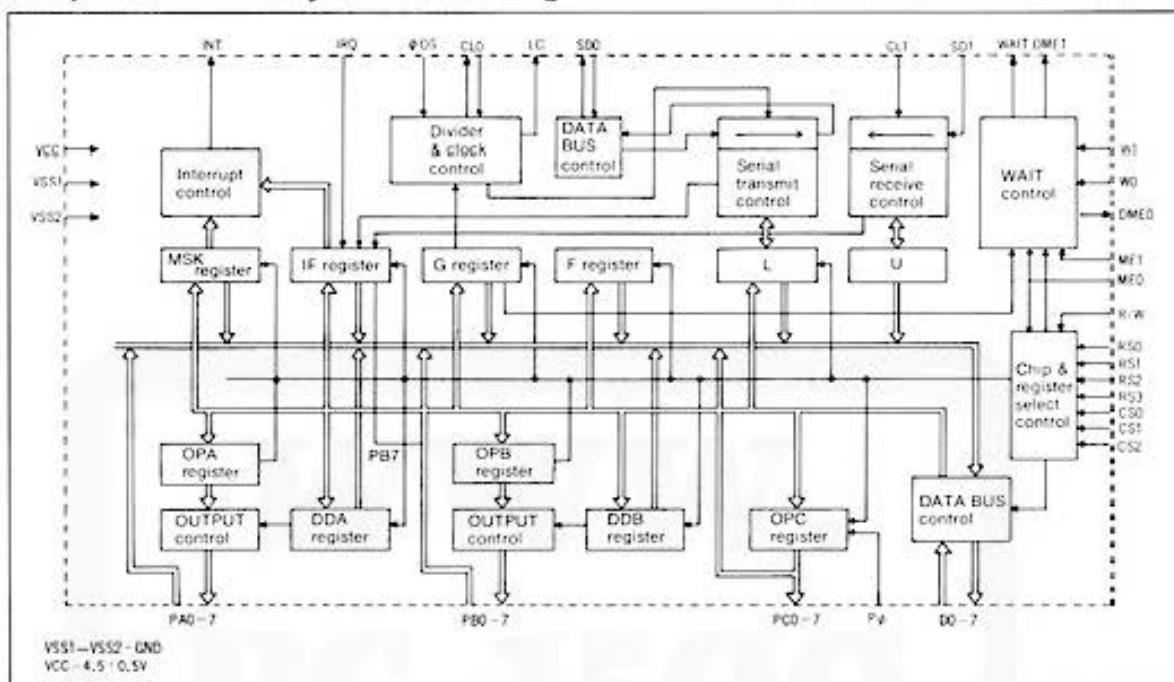
The serial transmit clock CLO can be programmed as discussed in Item A.

3-3. Internal structure

3-3-1. Block diagram

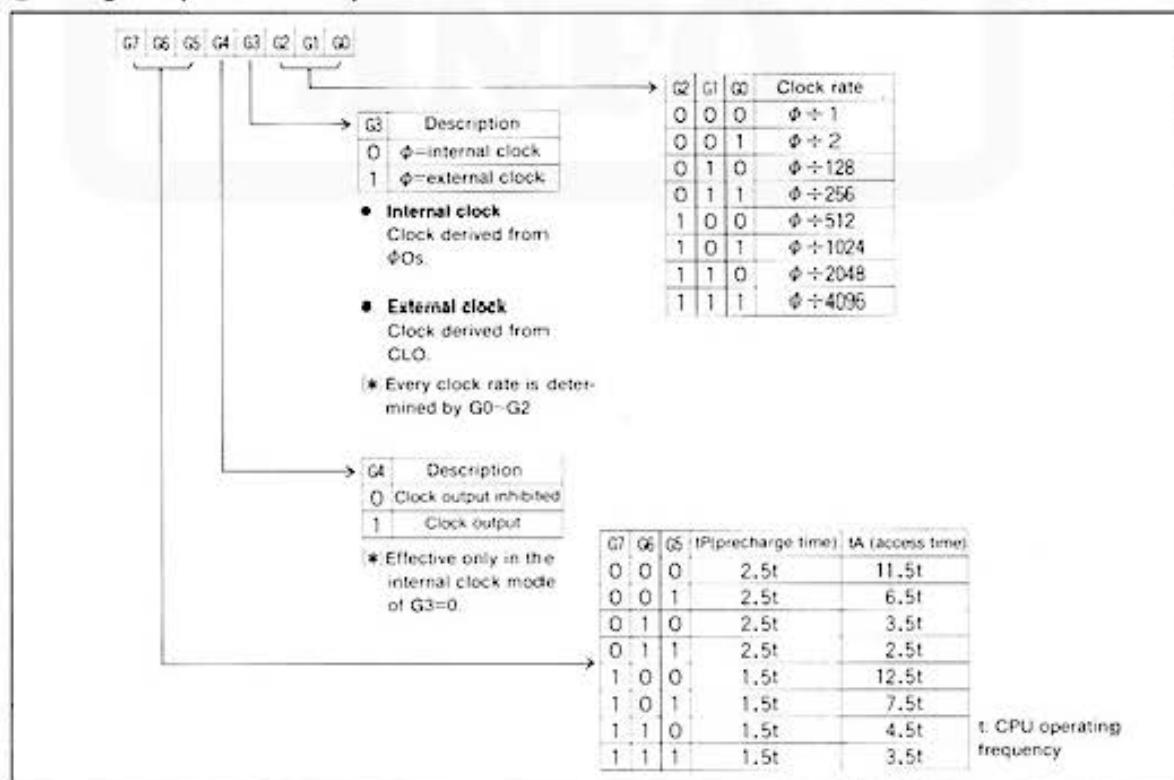
The I/O port controller consists of nine internal registers, wait controller, serial controller, and interrupt controller, and each of internal registers can be accessed as one location of the memory as seen from the CPU.

I/O port controller system block diagram



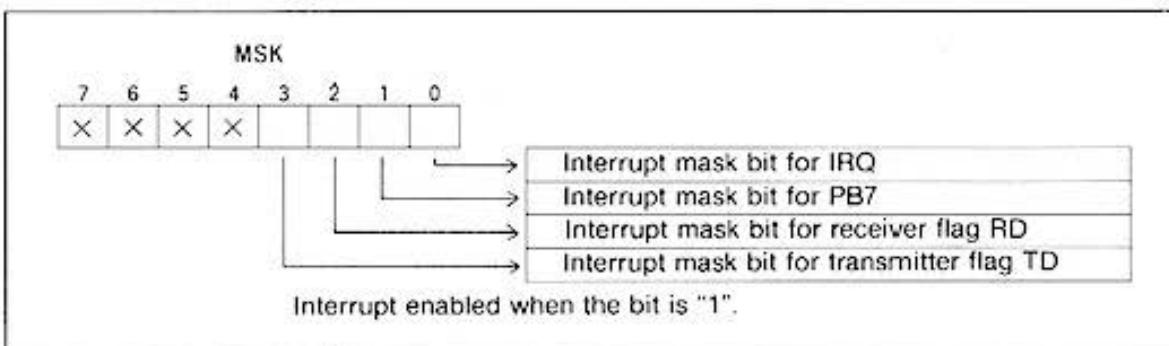
3-3-2. Internal registers

① G register (RS3~0=1001)



The G register can read/write data when register is selected (1001).

② MSK register



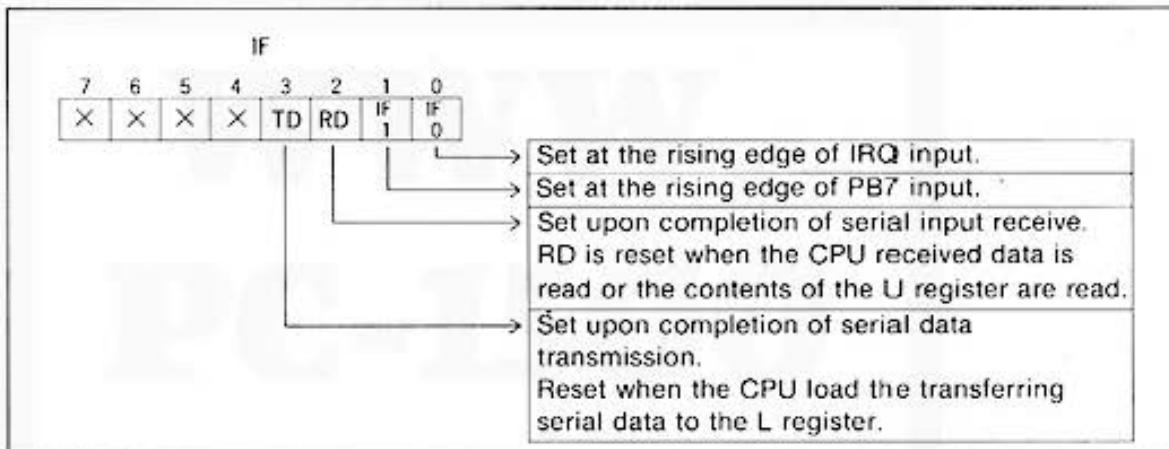
The MSK register can read/write data when register is selected (1010).

NOTE: When the contents of the MSK register are read, the contents of CL1, SD1, PB7, and IRQ are stored in high order digit positions.

**	CL1	SD1	PB7	IRQ	MSK 3	MSK 2	MSK 1	MSK 0
----	-----	-----	-----	-----	-------	-------	-------	-------

Contents of MSK read

③ IF register

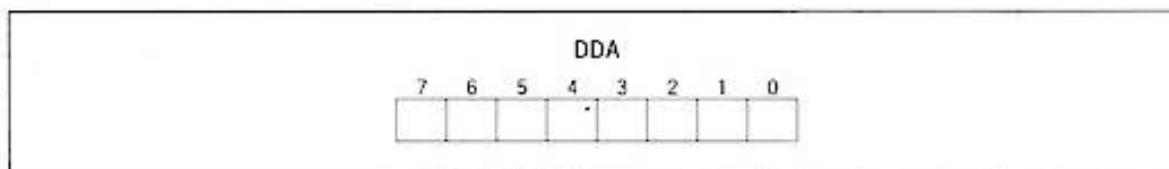


IF0 and IF1 can read/write data when register is selected (1011).

RD and TD are dedicated to read only.

NOTE: During receive of serial data, RD is reset. Term "serial data receive" means the period during which an 8-bit data is in reception, with the start bit excluded.

④ DDA register



The register used to determine the direction of the port PA.

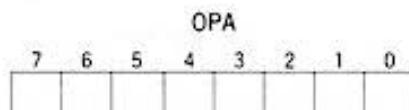
i-th bit of the DDA register

When 0	PA <i>i</i> is in the input mode.
When 1	PA <i>i</i> is in the output mode and outputs the contents of OPA <i>i</i> .

DDA can read/write when the register is selected (1100).

⑤ DDB register

The register used to determine the direction of the port PB.
 Selection of the input/output port is the same as in the DDA register.
 DDB can read/write data when the register is selected (1101).

⑥ OPA register

The OPA register is the buffer for input and output of data to the port PA.
 In the case of output, $DDAi$ must be set to "1" (output mode) and the register must be selected (1110), then the data on the bus line is loaded into $OPAi$, to be output on PAi .
 In the case of input, $DDAi$ must be set to "0" (input mode) which prohibits output from $OPAi$, then the contents of PAi are loaded into $OPAi$ and the data is sent on the bus line.

⑦ OPB register

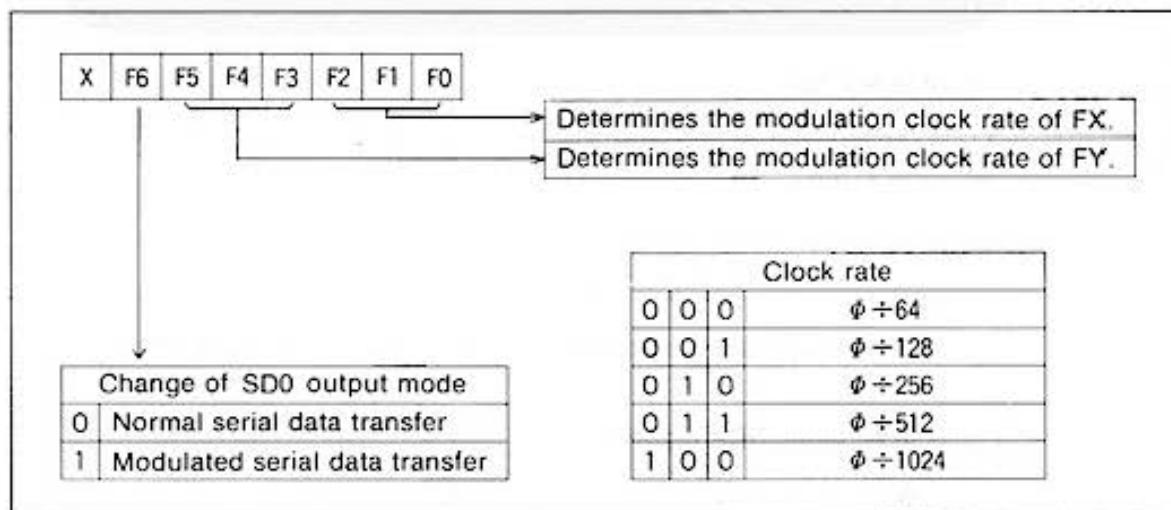
The OPB register is the buffer for input and output of data to the port PB.
 It has the same function as the OPA register.
 OPB can read/write data when the register is selected (1111).

⑧ OPC register

The OPC register can read/write data when the register is selected (1000).
 Also, the contents of the data bus can be latched to the OPC register at the falling edge of the external input clock $P\phi$.

⑨ F register

The F register can read/write data when the register is selected (0111).
 F0-2 determines the clock rate of the modulation clock FX and F3-5 the modulation clock of FY.



3-3-3. Pin description

(*): See Pad layout and structure in 3-5-2.

No.	Signal name	Function	No.	Signal name	Function
1	PA1	Port input/output	31	RS2	Register select input
2	PA2	Port input/output	32	RS3	Register select input
3	PA3	Port input/output	33	R/W	Read/write input
4	PA4	Port input/output	34	ME0	Memory enable input
5	PA5	Port input/output	35	ME1	Memory enable input
6	PA6	Port input/output	36	W0	Wait condition input
7	PA7	Port input/output	37	W1	Wait condition input
8	GND	Power source	38	GND	Power source
9	PB0	Port input/output	39	VCC	Power source
10	PB1	Port input/output	40	DME0	Memory enable output
11	PB2	Port input/output	41	DME1	Memory enable output
12	PB3	Port input/output	42	WAIT	Wait output
13	PB4	Port input/output	43	INT	Interrupt output
14	PB5	Port input/output	44	RESET	Initialize output
15	PB6	Port input/output	45	IRQ	Interrupt input
16	PB7	Port input/output, Interrupt input	46	ΦOS	Basic clock input
17	PΦ	Port PC latch clock	47	CL1	Serial receive clock input
18	PC0	Port output	48	SD1	Serial receive input
19	PC1	Port output	49	LC	Not used
20	PC2	Port output	50	CL0	Serial rec/trn clock input/output
21	PC3	Port output	51	SD0	Serial rec/trn input/output
22	PC4	Port output	52	D0	Data bus input/output
23	PC5	Port output	53	D1	Data bus input/output
24	PC6	Port output	54	D2	Data bus input/output
25	PC7	Port output	55	D3	Data bus input/output
26	CS0	Chip select input	56	D4	Data bus input/output
27	CS1	Chip select input	57	D5	Data bus input/output
28	CS2	Chip select input	58	D6	Data bus input/output
29	RS0	Register select input	59	D7	Data bus input/output
30	RS1	Register select input	60	PA0	Port input/output

3-4. Functions

3-4-1. Operation

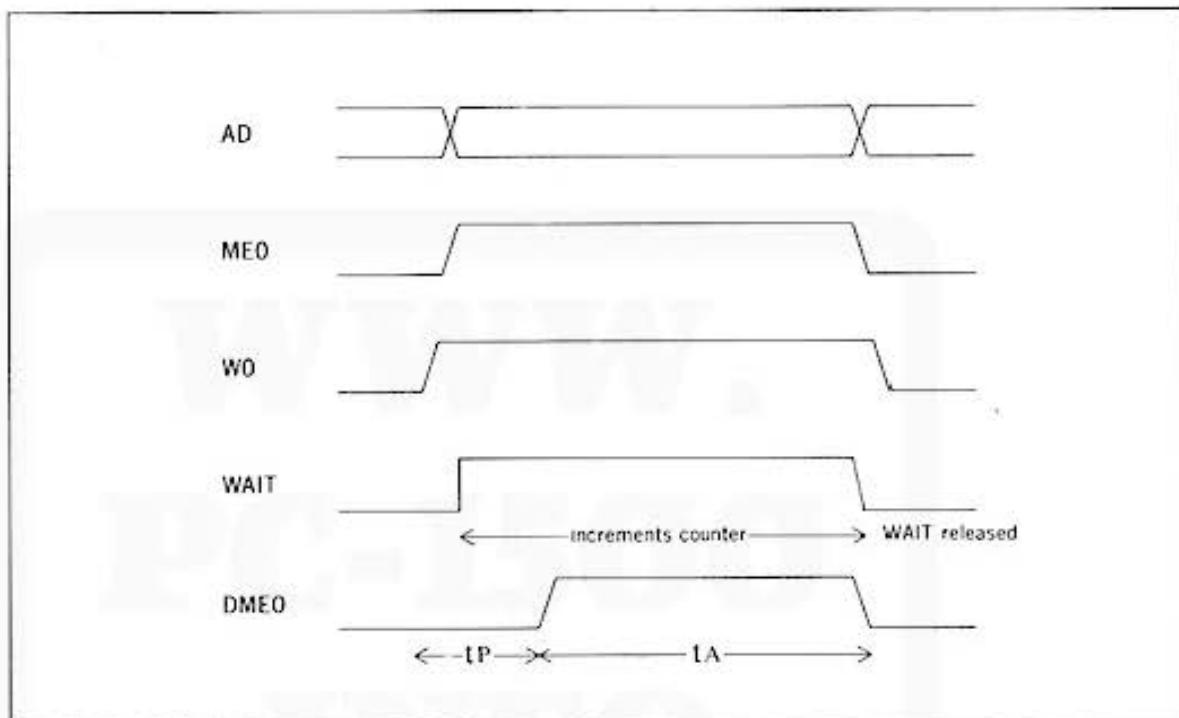
I/O port controller operation

CS2	CS1	CS0	RS3	RS2	RS1	RS0	R/W	Operation
0	1	1	0	1	0	0	0	Resets the divider internally contained in the I/O port chip.
			0	1	0	1	1	Reads the contents of the U register and sends them on the data bus. At the same time, the receive flag RD is reset.
			0	1	1	0	0	The contents of the data bus are converted into a serial data signal of start/data/stop structure. At the same time, the transmit flag TD is reset.
			0	1	1	1	0	The contents of the data bus are stored in the F register.
			1	0	0	0	0	The contents of the data bus are stored in the OPC register.
							1	The contents of the OPC register are sent on the data bus.
			1	0	0	1	0	The contents of the data bus are stored in the G register.
							1	The contents of the G register are sent on the data bus.
			1	0	1	0	0	The contents of the data bus are stored in the MSK register.
							1	The contents of the MSK register are sent on the data bus.
			1	0	1	1	0	The contents of the data bus are stored in the IF register.
							1	The contents of the IF register are sent on the data bus.
			1	1	0	0	0	The contents of the data bus are stored in the DDA register.
							1	The contents of the DDA register are sent on the data bus.
			1	1	0	1	0	The contents of the data bus are stored in the DDB register.
							1	The contents of the DDB register are sent on the data bus.
			1	1	1	0	0	The contents of the data bus are stored in the OPA register.
							1	The contents of the PA _i are sent on the data bus.
			1	1	1	1	0	The contents of the data bus are stored in the OPB register.
							1	The contents of the PB _i are sent on the data bus.

3-4-2. Wait control

① Function

Wait is a function applied in accessing a slow action memory. It makes the CPU operation temporarily halted until a complete access is done to the slow action memory. When the CPU makes access to the address where slow action memory is assigned, the condition "1" is entered to the wait condition inputs W0 and W1. W0 is the wait condition input which is applied to place wait to the memory area controlled by ME0, while W1 is the wait condition for the memory area controlled by ME1. When the wait condition is established, that is, when " $W0 \cdot ME0 + W1 \cdot ME1 = 1$ " is met, the WAIT signal is issued to the CPU. As the wait control counter is provided internally, it releases WAIT to the CPU when it is counted to the value set by the program, then the CPU proceeds to execute a next machine cycle.



Since a slow access memory usually consists of dynamic logic, it needs a precharge time (tP). Thus, the I/O port controller issues the memory enable signals DME0 and DME1 to the dynamic memory, including the precharge time. DME0 is for ME0 and DME1 is for ME1.

When the CPU accesses the I/O port controller, the signal WAIT is issued without the wait condition in W0 and W1.

② Wait time

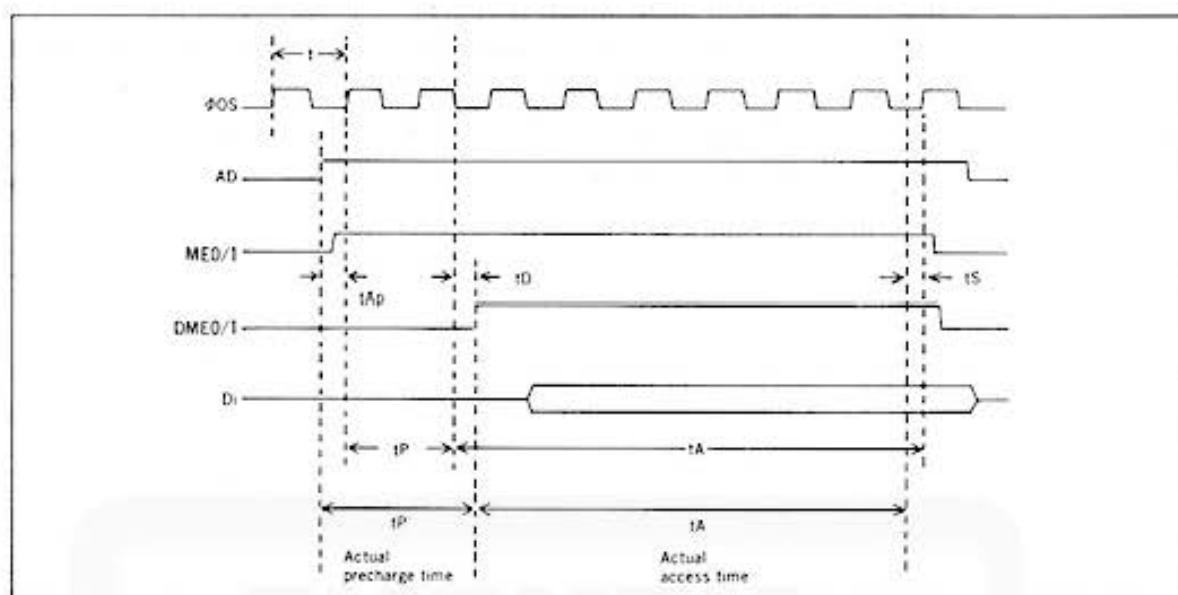
Wait time can be programmed by means of bit positions of G5, 6 and 7 of the G register. Relation of G5, 6,7 with tA and tP refer to page 70.

③ Wait time of I/O port controller itself

When the CPU accesses the I/O port controller, wait of a single CPU machine cycle is automatically applied.

If the W0 and W1 is in the wait state when the CPU accessed the I/O port controller, wait ends in one cycle.

NOTE: When there is no wait condition in W0 and W1, the same waveform as ME0 and ME1 are sent on DME0 and DME1.



tP' and tA' for actual ROM is as follows:

$$tP' = tAP + tP + tD$$

$$tA' = tA - tD - tS$$

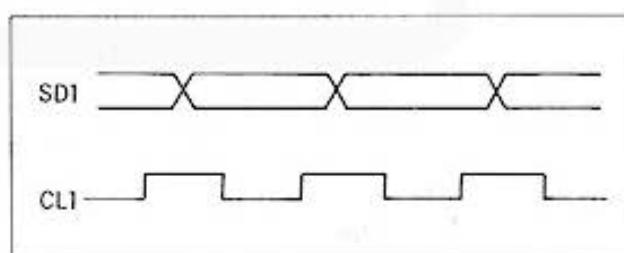
$$tS = 200\text{ns}$$

Since tAP and tD differ depending on the peripheral circuit configuration, they should be computed on the basis of load capacitance.

3-4-3. Serial data input

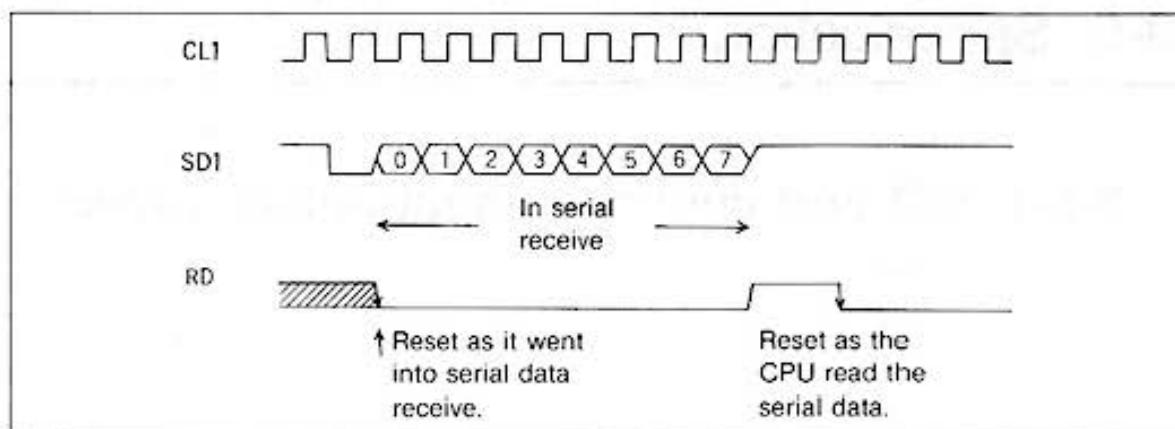
SDI is a serial data transfer input and CL1 serial data transfer clock.

The I/O port controller reads the input data at the rising edge of CL1. Serial data goes into the receiving mode when it changes from an idle state ($SDI=1$) to low state and reads data from a next clock.



When the 8-bit data is received, it sets the receive and flag RD active. If the mask bit is on at this point, interrupt request is issued to the CPU. RD will be reset upon reading the 8-bit data.

RD will be reset in a course of serial data receiving, which is the period that the 8-bit data is being received, without including the start bit.



3-4-4. Reset

When the RESET line is kept in "1" at least for three Φ_{OS} clock cycles, it causes internal reset, by which all internal registers are cleared to "0" and ports PA and PB go into the input mode. The divider, however, will not be reset.

Terminal states after the reset are as shown in Table below.

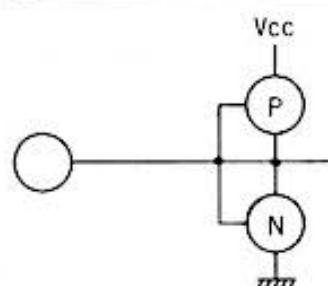
Pin name	In/Out	State after reset
D0~D7	In/Out	High impedance
PA0~PA7	"	"
PB0~PB7	"	"
PC0~PC7	Out	Low level
WAIT	"	(NOTE)
DME0	"	"
DME1	"	"
INT	"	Low level
SD0	In/Out	High level
CL0	"	Low level

NOTE: WAIT, DME0, and DME1 are output dependent of W0, ME0, W1, and ME1 and do not have any connection with reset.

3-5. Specification

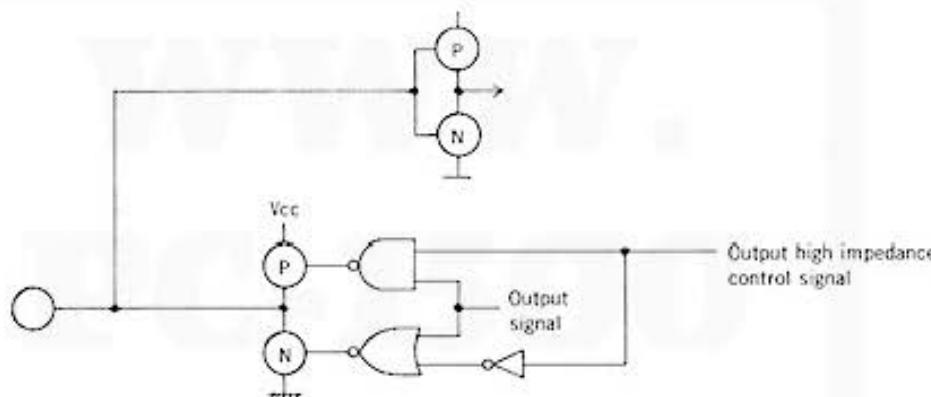
3-5-1. I/O port controller input/output circuits

① Input



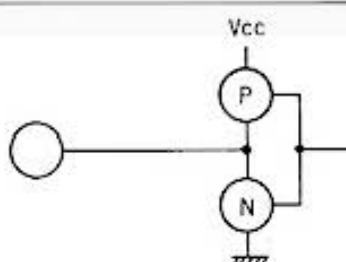
Applicable terminals : RESET, P_φ, R/W, RS0~3, CS0, CS1, CS2, ME0, ME7,
W0, W1, CL1, SD1, φOS, IRQ

② Input/output



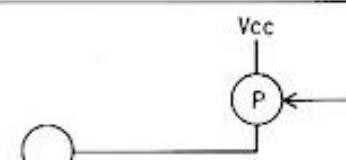
Applicable terminals : PA0~7, PB0~7, D0~7, SD0, CLO

③ Output



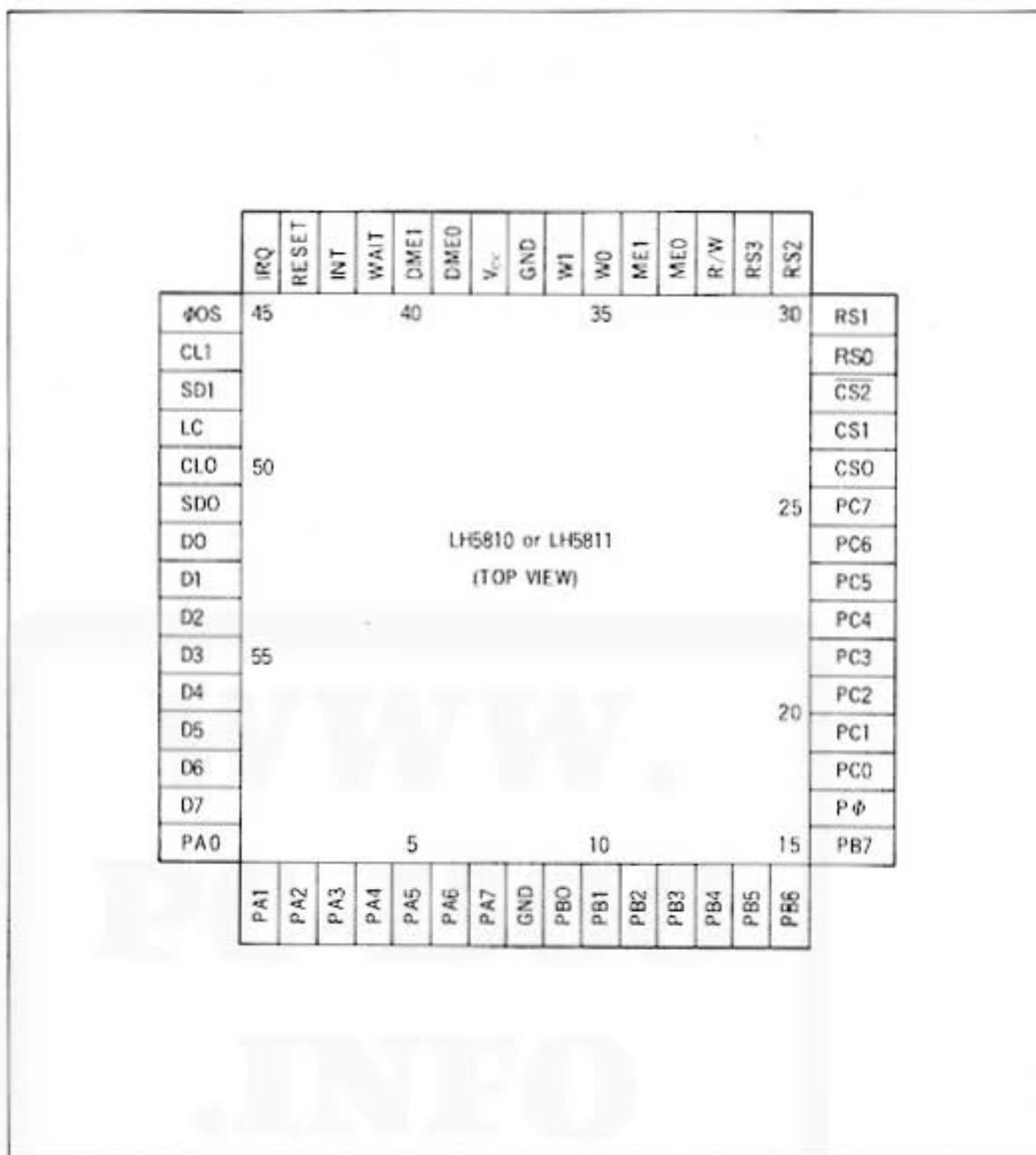
Applicable terminals : WAIT, DME0, DME1

④ Output



Applicable terminal : INT

3-5-2. Pad layout and structure



3-5-3. Electrical characteristics

① Absolute maximum ratings

Parameter	Symbol	Limits	Unit
Input apply voltage (*1)	V _{IN}	-0.3 to 6.5	V
Input apply voltage (*2)	V _{IN}	-0.3 to V _{CC} +0.3	V
Operating temperature	T _{OPR}	-5 to +55	°C
Storage temperature	T _{STG}	-55 to +150	°C

*1: Applicable to V_{CC} and with respect to GND.

*2: Applicable to other than V_{CC}, GND, and with respect to GND.

② Operating condition

Parameter	Symbol	Limits	Unit
Supply voltage	V _{CC}	4.0 to 5.0	V

① Electrical characteristics**① - 1 - DC characteristics**

$T_a = -5 \sim 55^\circ C$
 $V_{cc} = 4.0 \sim 5.0 V$

Parameter	Symbol	Limits			Unit	Test condition	Note
		Min	Typ	Max			
Input voltage	V_i	0		0.8	V		1
	V_{ih}	2.4		V_{cc}	V		
Output voltage 1	V_{o1}			0.4	V	$I_{OL}=1.6mA$	2
	V_{oh}	2.4			V	$I_{OH}=0.4mA$	
Output voltage 2	V_{oe}	1.5			V	$I_{OH}=1.5mA$	3
Output leakage current	$ I_{oL} $			1.0	μA		4
Input current	$ I_{il} $			1.0	μA	$V_n=0V$ or V_{cc}	5
Normal power dissipation	I_{cca}		0.4	0.9	mA	$\phi OS=2MHz$	6
	I_{ccb}			5	μA	$\phi OS=0V$	7

NOTES: For Note number, refer to the list below.

Note No.	Applicable terminals
1	PA0~7, PB0~7, D0~7, R/W, RS0~3, CS0, CS1, $\overline{CS2}$, ME0, ME1, W0, W1, SD1, CL1, ϕOS , SD0, CL0, IRQ, RESET.
2	PA0~7, PB0~7, PC0~7, D0~7, DME0, DME1, WAIT, CL0, INT
3	PA0~7, CL0, SD0.
4	All input terminals.
5	All output terminals.
6	The term "normal" applies to the reset state that the I/O port controller is not selected and the 2MHz clock should be supplied through the ϕOS pin. The following input terminals should be connected to 0V except $\overline{CS2}$. PA0~7, PB0~7, D0~7, CS0, CS1, $\overline{CS2}=V_{cc}$, R/W, RS0~3, P ϕ , ME0, ME1, W0, W1, CL1, SD1, IRQ.
7	Applied to the state that no clock is supplied to the ϕOS terminal in the normal condition.

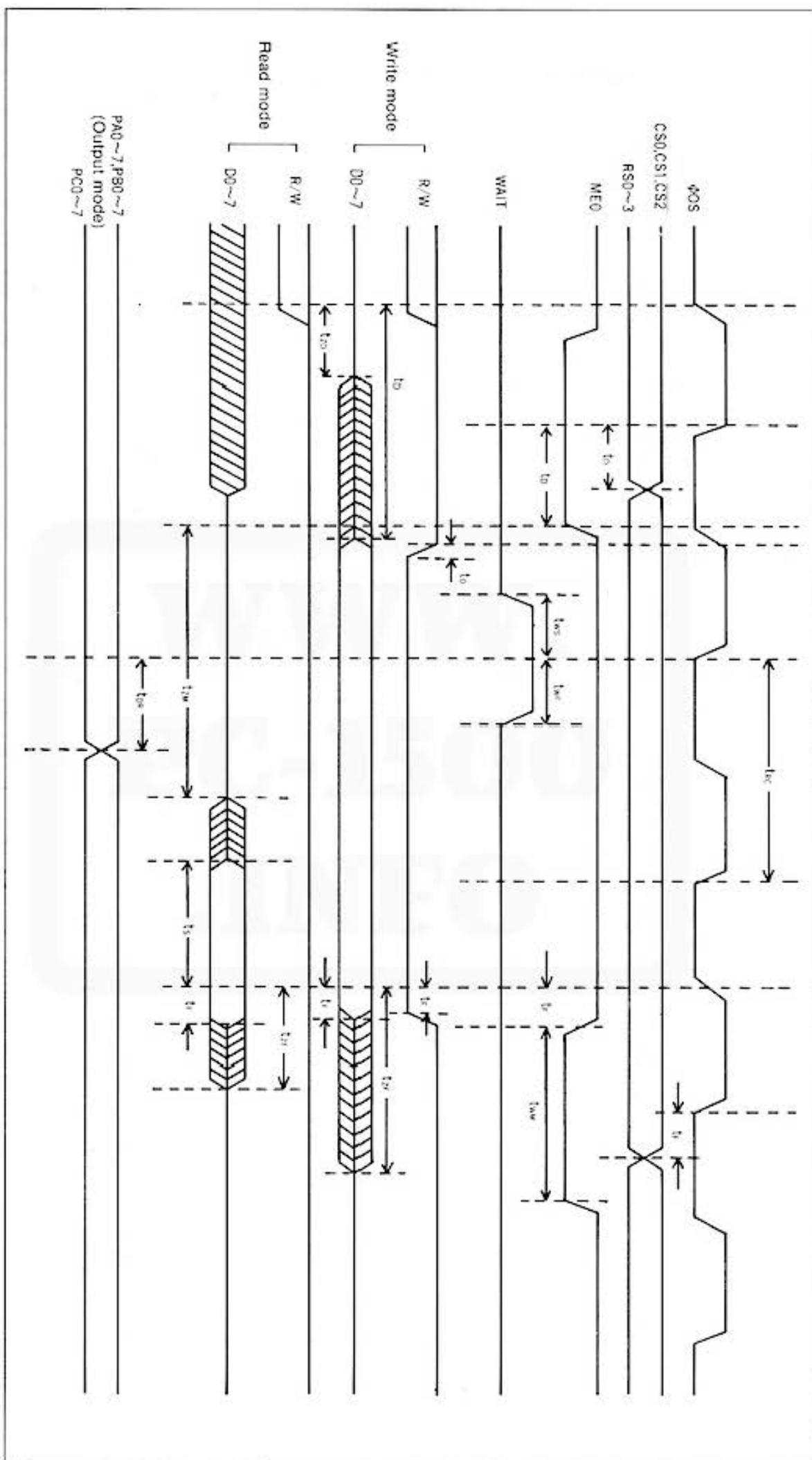
(3) - 2 - AC characteristics

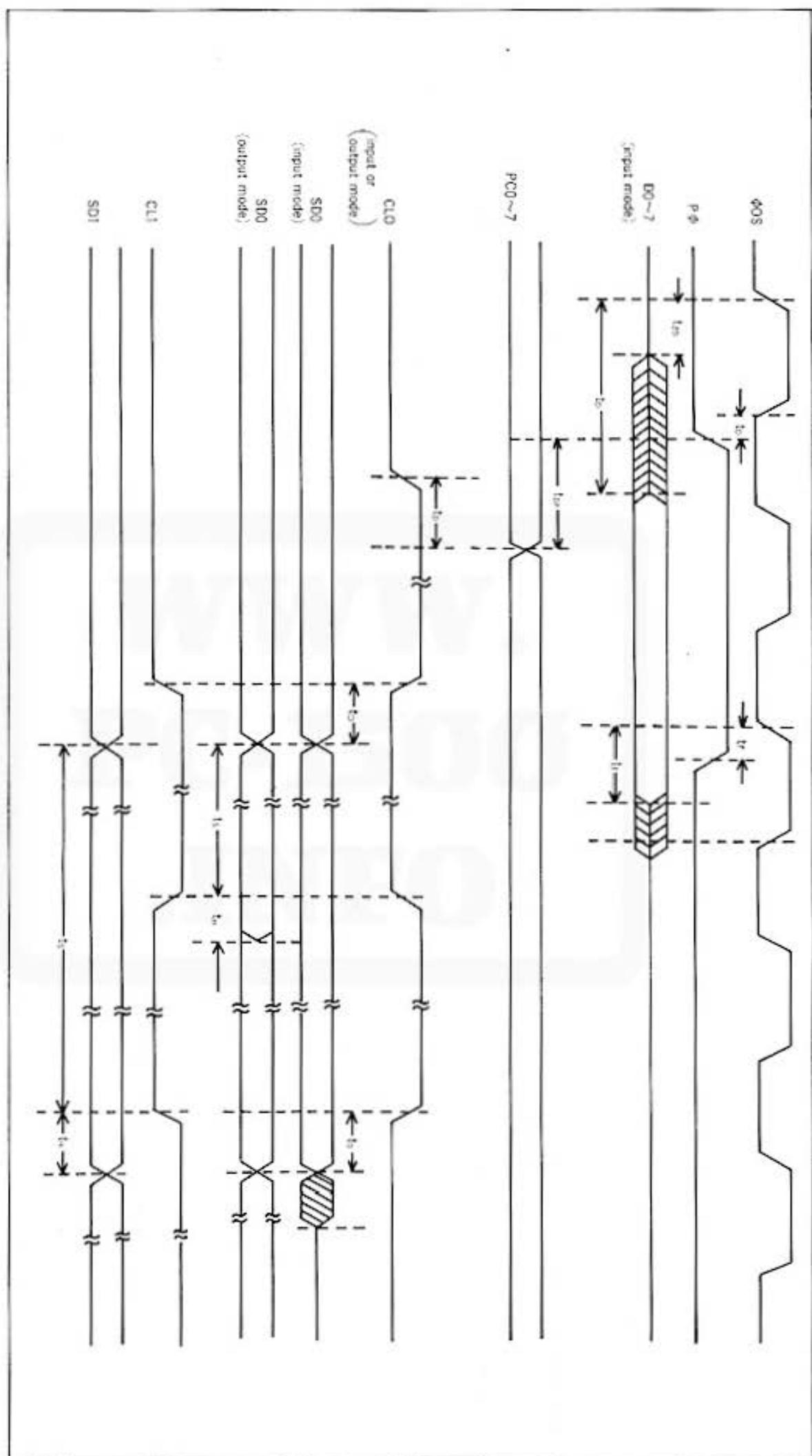
(Ta=-5 to 55°C, V_{cc}=4.0 to 5.0V)

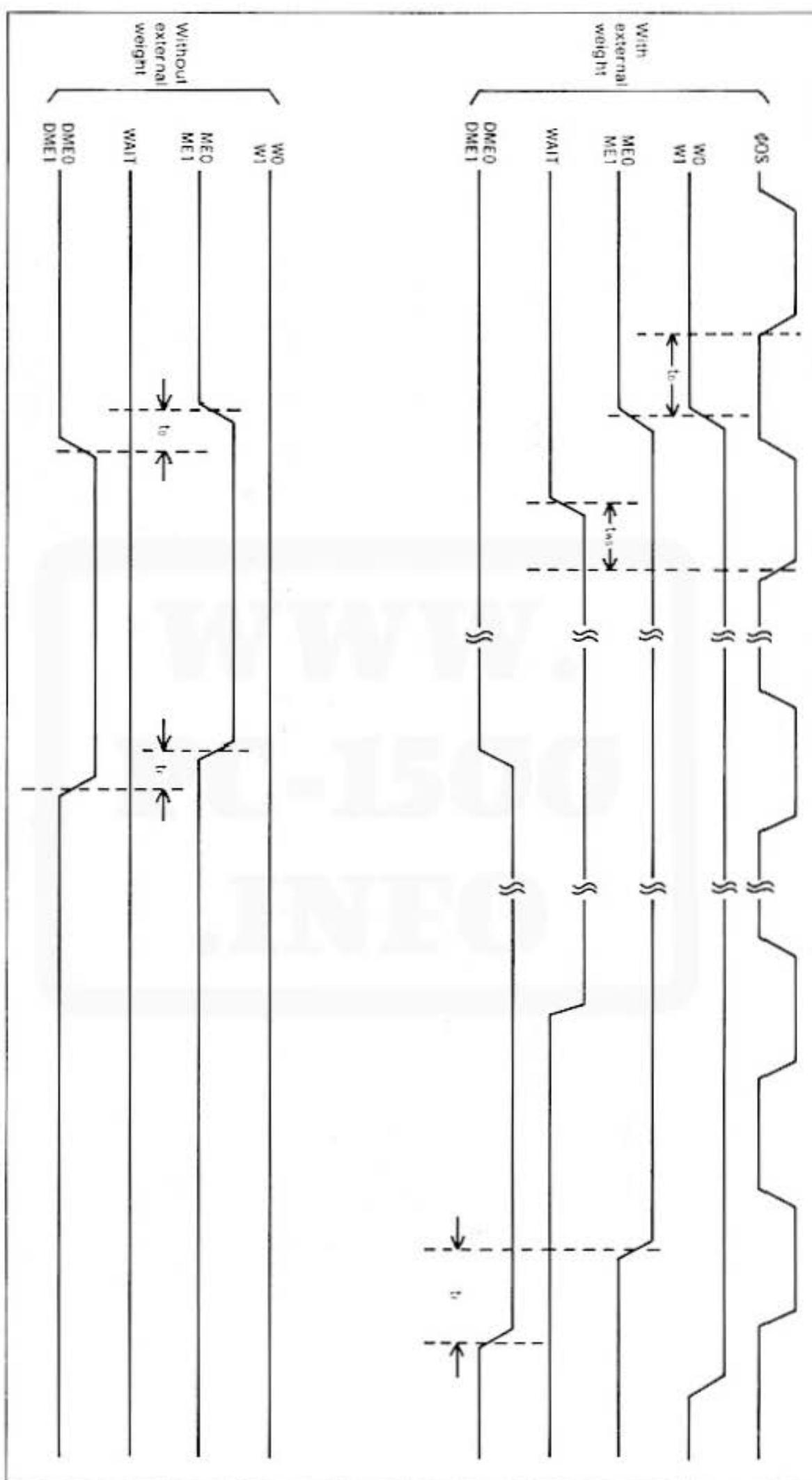
Parameter	Symbol	Limits			Unit	Test condition
		Min	Typ	Max		
CS0, CS1, CS2, RS0~3	t ₀			230	ns	
	t _r	80			ns	
ME0, ME1	t ₀			240	ns	
	t _r	80		175	ns	
	t _{sw}	250			ns	
WAIT	t _{ws}	150			ns	C _L =50pF
	t _{wr}	50		200	ns	
D0~7 (input mode)	t ₀			500	ns	V _{OH} =0.8V _{cc}
	t _r	130			ns	
	t _{zo}	100			ns	
	t _{zf}			300	ns	
D0~7 (output mode)	t _s	200			ns	
	t _r	50			ns	
	t _{zw}	150			ns	
	t _{zf}			100	ns	
R/W	t ₀	-30		50	ns	V _{OH} =0.8V _{cc}
	t _r			150	ns	
ΦOS	t _{oc}	500			ns	
	t _r			150	ns	
	t _f			40	ns	
PA0~7, PB0~7 (output mode)	t _{os}			1	μs	C _L =100pF
PC0~7 (register select)	t _{os}			1	μs	C _L =100pF
PΦ	t ₀			350	ns	V _{OH} =0.8V _{cc}
	t _r			100	ns	
PC0~7 (PΦ latch)	t _{os}			1	μs	C _L =100pF
SD0 (input mode) SD1	t _s	100			ns	
	t _r	50			ns	
SD0 (output mode)	t ₀			100	ns	C _L =100pF
DME0	t ₀			120	ns	C _L =50pF
	t _r			120	ns	
SD0	t _z			200	ns	C _L =100pF
CL0	t _r			100	ns	0.1~0.9V _{cc} C _L =50pF
	t _f			100	ns	0.9~0.1V _{cc} C _L =50pF

(*): Unless otherwise specified, the criterion shall be V_{OH}=2.0V, V_{OL}=0.4V

③ - 3 - Timings







4

PC-1500 hardware description

4-1. PC-1500 system configuration

4-1-1. Outline

Around the PC-1500 Pocket Computer, the PC-1500 system can be configured with a variety of peripherals, which include the CE-150 Color Graphic Printer, CE-151 4KB Memory Module, CE-152 Cassette Tape Recorder, CE-153 Software Board, CE-154 System Carrying Case, CE-155 8KB Memory Module, CE-158 Serial and Parallel Interface, and CE-159 Program Module.

① PC-1500 Pocket Computer

It is a pocket computer that features high speed processing with the CMOS 8-bit microprocessor in use, and it has memories of 16KB ROM and 3.5KB RAM on standard. It also permits expansion by the use of options.

Four pieces of Type AA dry batteries are used for its power source and the 7×156 dot LCD is used for the display.

② CE-150 Color Graphic Printer

It is a ball-point pen type four color printing X-Y plotter. It has 8KB ROM for the memory and is driven by five pieces of Type AA Ni-Cd batteries. It also has the cassette interface built in the unit.

③ CE-151 4KB Memory Module

It is a 4KB option memory unit in which two chips of 2KB RAM are contained.

④ CE-152 Cassette Tape Recorder

It is a cassette tape recorder dedicated for use with the PC-1500 system. It is driven by four pieces of Type AA dry batteries.

⑤ CE-153 Software Board

It has 140 keys arranged on the keyboard and each key assignment is defined by means of programming. As it does not have power supply by itself, it is supplied from the PC-1500.

⑥ CE-154 System Carrying Case

The carrying case exclusively designed for carrying of the PC-1500, CE-150, CE-152, CE-153.

⑦ CE-155 8KB Memory Module

It is an 8KB option memory unit in which four chips of 2KB RAM are contained.

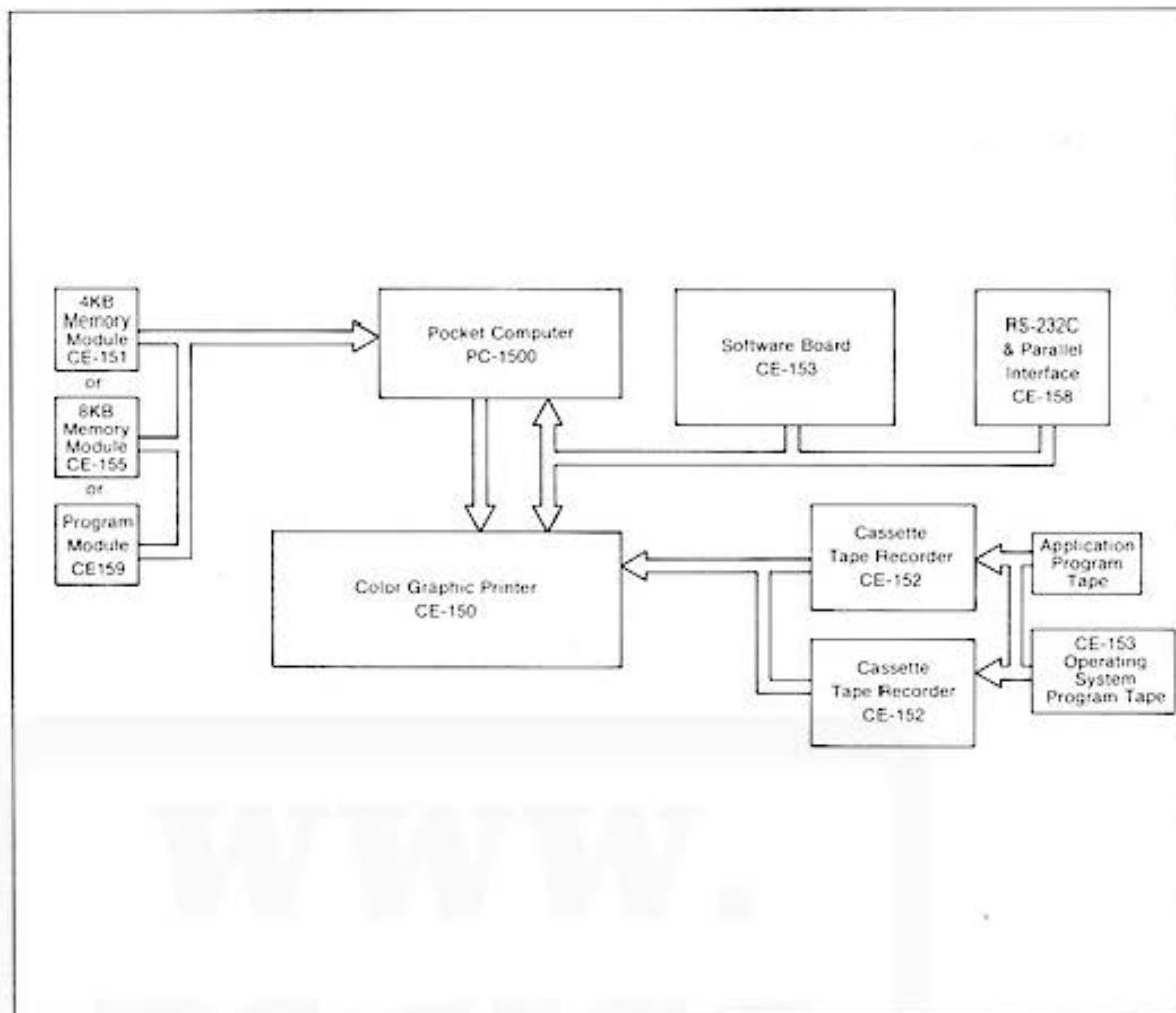
⑧ CE-158 Serial and Parallel Interface

It has the RS-232C serial interface and the Centronics parallel interface with the built-in 16KB ROM. It is driven by four pieces of Type AA Ni-Cd batteries. It will be possible to interface with a normal type printer, computer, instrument, etc.

⑨ CE-159 Program Module

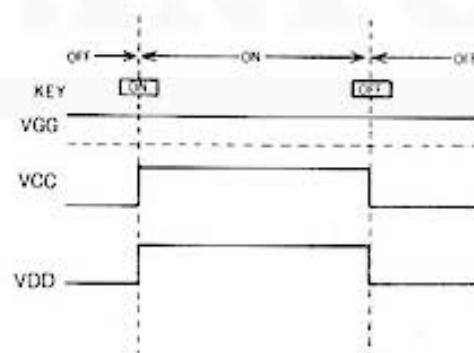
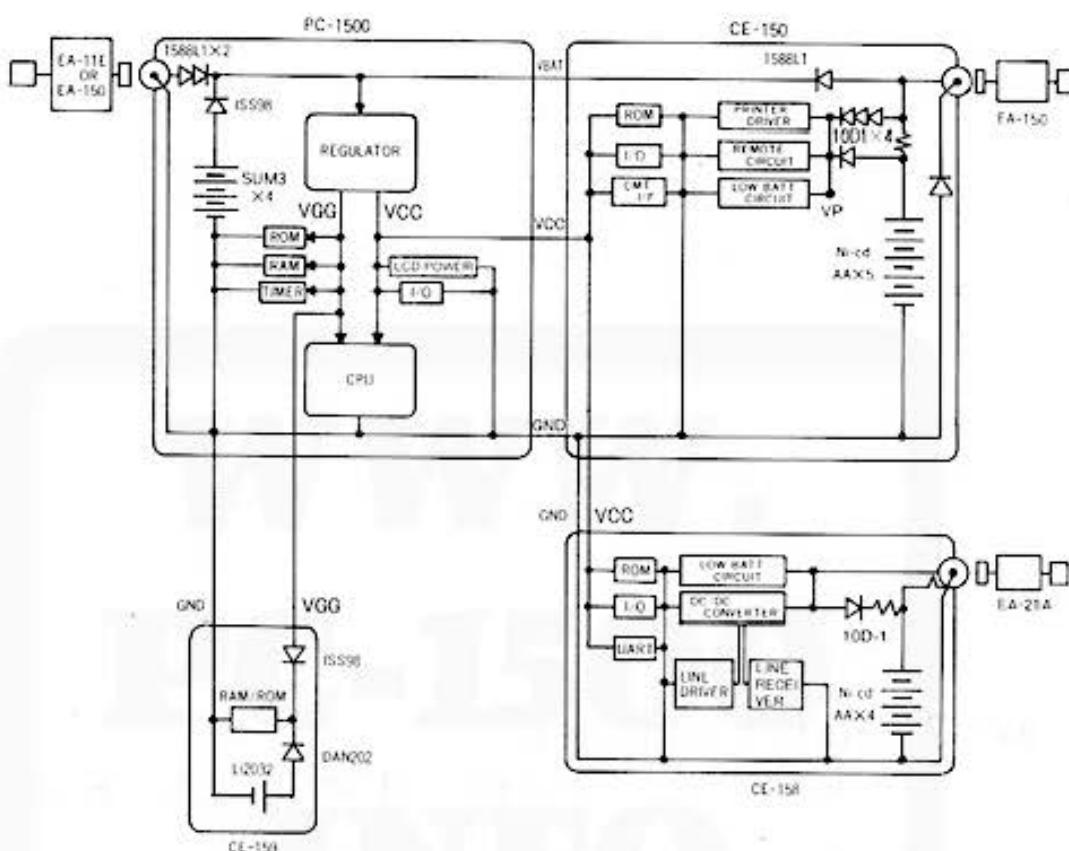
It is a detachable memory module that consists of four chips of 2KB RAM, of which area can be partially or entirely used for the read only area. The contents of the memory can be retained by means of the internal lithium battery even if it was detached from the PC-1500.

4-1-2. Block diagram



4-1-3. Power supplies (PC-1500, CE-150, CE-158, CE-159)

- ① PC-1500 and its peripheral units (CE-150, CE-158, CE-159) are driven independently by dry battery, Ni-Cd battery, or lithium battery.
When the PC-1500 is connected with peripheral, RAM, ROM, I/O PC, CMT I/F, and UART are driven by VCC or VGG supplied from the PC-1500.
- ② When the Ni-Cd battery of the CE-150 is charged capable to drive the PC-1500, power is supplied from the CE-150 to the PC-1500 and therefore the battery of the PC-1500 does not consume. In addition, it drives the driver of the printer and remote control relays.
- ③ The Ni-Cd battery of the CE-158 drives the driver and receiver of the RS-232C serial interface and the Centronics parallel interface.
- ④ When the CE-159 is in connection with the PC-1500, it is operated by VGG of the PC-1500, and its memory contents can be retained by the internal lithium battery when disconnected from the PC-1500.



NOTE: VGG is always output.

4-2. PC-1500

4-2-1. Outline

① CPU

The LH5801 CMOS 8-bit Microprocessor is used. As it is operated by the clock frequency of 2.6MHz from the crystal oscillator, its internal machine cycle is 1.3MHz.
BFI Depression of the **[ON]** key turns it to high level so that VCC is supplied from BFO.

IN0~IN7 Input port for other than the **[ON]** key.

D0~D7 Bidirectional data bus which is used to write and read data to/from the external memory.

D0: LSB

D7: MSB

AD0~AD15 Address bus.

AD0: LSB

AD15: MSB

XI0, XI1 External crystal connection terminals, through which the 2.6MHz crystal is connected.

XI0: Input

XI1: Output

② I/O PC

Either LH5810 or LH5811 is used. The same chip is used for I/O of CE-150, CE-153, and CE-158.

PA0~PA7 Key strobe output

PB7 **[ON]** key input

AD0~3, AD12~13, DM1

..... Address of I/O port is set within F000H to F00FH of the ME1 area.

③ Timer IC

μ PD1990AC is used and connected with the 32.768kHz crystal.

④ Chip select decoder

It consists of two chips of TC40H139F and TC40H138F and it is used to select chip by means of S0~4, S6, S7, $\overline{2Y2}$ and $\overline{2Y3}$. For more details, refer to "Chip select circuit".

⑤ Display chip

Because 4-bit SC882G is used, chip 1 is used in pair with chip 3 and chip 2 with chip 4. Select signal is commonly used by chip 1 and chip 3 or chip 2 and chip 4. Data bus is therefore divided into D0~D3 and D4~D7 in order to handle compatible with the 8-bit RAM.

Address is within 7600H to 77FFH of the ME0 area and is used for the fixed variable area, in addition to the display buffer.

⑥ System ROM

It is the PC-1500 system program residing ROM for which the 8-bit \times 16K SC61328F is used. Address is within C000H to FFFFH of the ME0 area.

⑦ System RAM

Because the 4-bit \times 1K bytes TC5514 is used in a pair, the data bus is divided into two of D0~D3 and D4~D7 and the select signal S7 is commonly shared so as to be compatible with the 8-bit RAM.

Address is within 7800H to 7BFFFH of the ME0 area which is used for the system memory area and for the fixed variable area.

(8) User RAM

It is the user RAM for which 8-bit × 2KB HM6116 is used. Address selected by S0 is within 4000H to 47FFH.

(9) 40-pin connector

It is the connector used for the connection of one of optional memory modules and program module, on which provided signal terminals for address bus, data bus, and chip select.

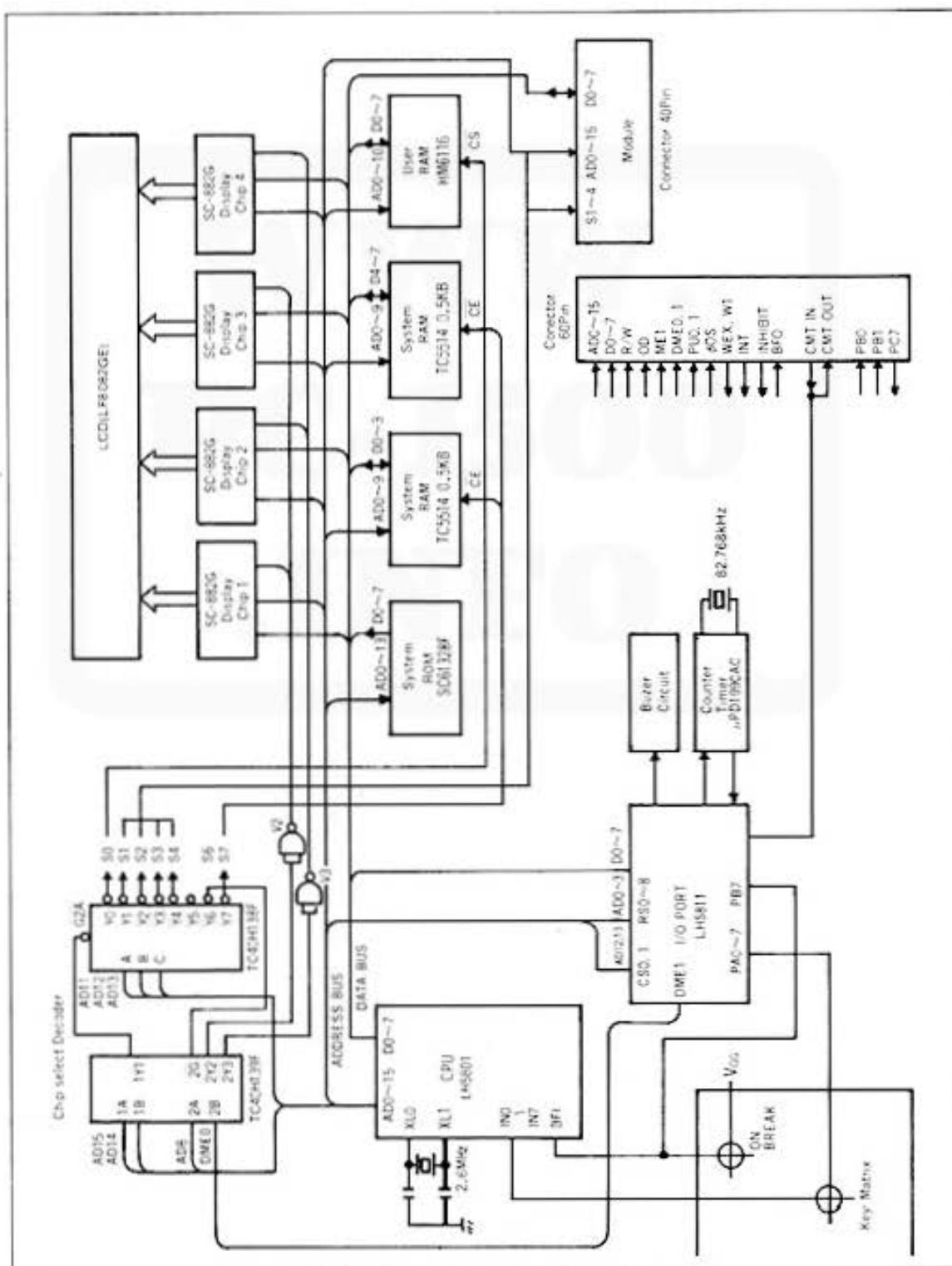
(10) 60-pin connector

It is the connector used for the connection of optional printer and interface, on which provided signal terminals of address bus and data bus for the control of peripherals.

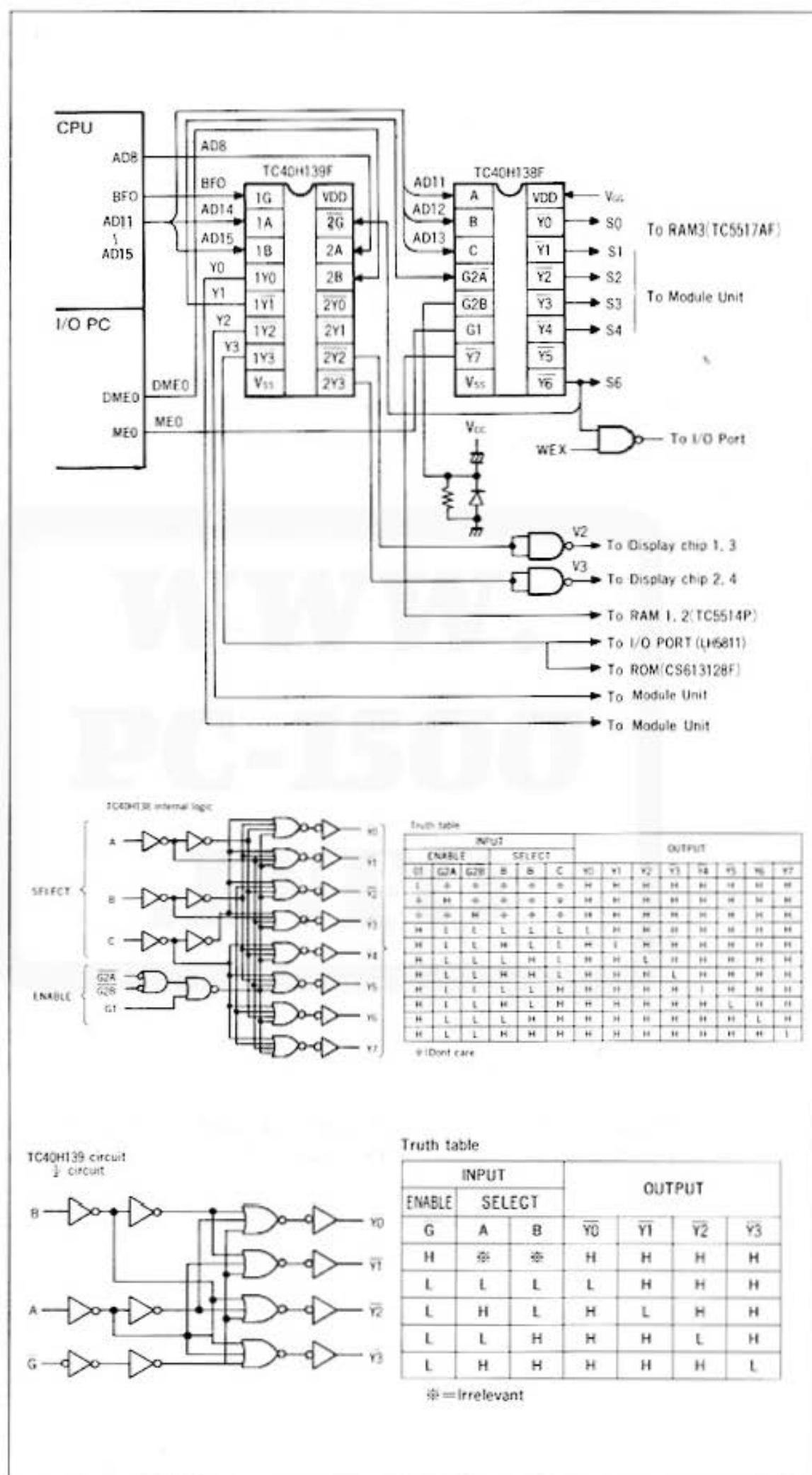
(11) LCD

The 7 × 156 dot LF8082GE multi-display is used.

4-2-2. Block diagram



4-2-3. Chip select circuit



- **$\overline{Y0} \sim \overline{Y3}$ is selected by the decoder IC (TC40H139F) when the gate signal input BFO ($\overline{1G}$) is low.**

$Y0 \dots \dots$ With low state of AD14 and AD15, the $Y0$ output goes low so as to select the $(\overline{Y0})$ user memory area of the module unit. (Address assignment of 0000H~3FFFH)

$Y1 \dots \dots$ With high state of AD14 and low state of AD15, the $Y1$ output goes low so as $(\overline{Y1})$ to select the gate ($\overline{G2A}$) of the IC (TC40H138F). (Address assignment of 4000H~7FFFH)

$Y2 \dots \dots$ With low state of AD14 and high state of AD15, the $Y2$ output goes low so as $(\overline{Y2})$ to select the optional ROM area of the module unit. (Address assignment of 8000H~BFFFH)

$Y3 \dots \dots$ With high state of AD14 and AD15, the $Y3$ output goes low so as to select $(\overline{Y3})$ the system program ROM (SC61328F) and the I/O port (LH5811 or LH5810). (Address assignment of C000H~FFFFH)

- **$S0 \sim S7$ is selected by the decoder IC (TC40H138F) when the gate signal input $ME0$ ($G1$) is high, $Y1$ ($\overline{G2A}$) low, and $\overline{G2B}$ is low (normally low).**

$S0 \dots \dots$ With all of AD11, AD12 and AD13 in low state, the $S0$ output goes low so as $(\overline{Y0})$ to select the user RAM (HM6116). (Address assignment of 4000H~47FFFH)

$S1 \dots \dots$ With high state of AD11 and low state of AD12 and AD13, the $S1$ output $(\overline{Y1})$ goes low so as to select the optional user RAM area. (Address assignment of 4800H~4FFFFH)

$S2 \dots \dots$ With low state of AD11 and AD13 and high state of AD12, the $S2$ output $(\overline{Y2})$ goes low so as to select the optional user RAM area. (Address assignment of 5000H~57FFFH)

$S3 \dots \dots$ With low state of AD11 and AD12 and high state of AD13, the $S3$ output $(\overline{Y3})$ goes low so as to select the user RAM area. (Address assignment of 5800H~5FFFFH)

$S4 \dots \dots$ With high state of AD11 and AD13 and low state of AD12, the $S4$ output $(\overline{Y4})$ goes low so as to select the user RAM area. (Address assignment of 6000H~67FFFH)

$S5 \dots \dots$ Do not use.

$S6 \dots \dots$ With low state of AD11 and high state of AD12 and AD13, the $S6$ output $(\overline{Y6})$ goes low so as to receive on the I/O port the data from the display chip RAM or interrupt input from the option. (Address assignment of 7000H~77FFFH)

$S7 \dots \dots$ With all of AD11, AD12, and AD13 in high state, the $S7$ output goes low so $(\overline{Y7})$ as to select the system RAM (TC5514) (Address assignment of 7800H~7FFFFH)

- **$\overline{2Y2}, \overline{2Y3}$ is selected by the decoder IC (TC40H139) when the $\overline{2G}$ gate becomes effective after the selection (low state) of the TC40H138F output $S6$ ($\overline{Y6}$).**

$V2 \dots \dots$ With low state of AD8 and high state of DME0, the $\overline{2Y2}$ output goes low and $(\overline{2Y2})$ makes the NAND gate output $V2$ high so as to select display chip 1 and 3. (Address assignment of 7600H~76FFFH)

$V3 \dots \dots$ With high state of AD8 and DME0, the $\overline{2Y3}$ output goes low and makes the $(\overline{2Y3})$ NAND gate output $V3$ high so as to select display chip 2 and 4. (Address assignment of 7700H~77FFFH)

Chip select signal

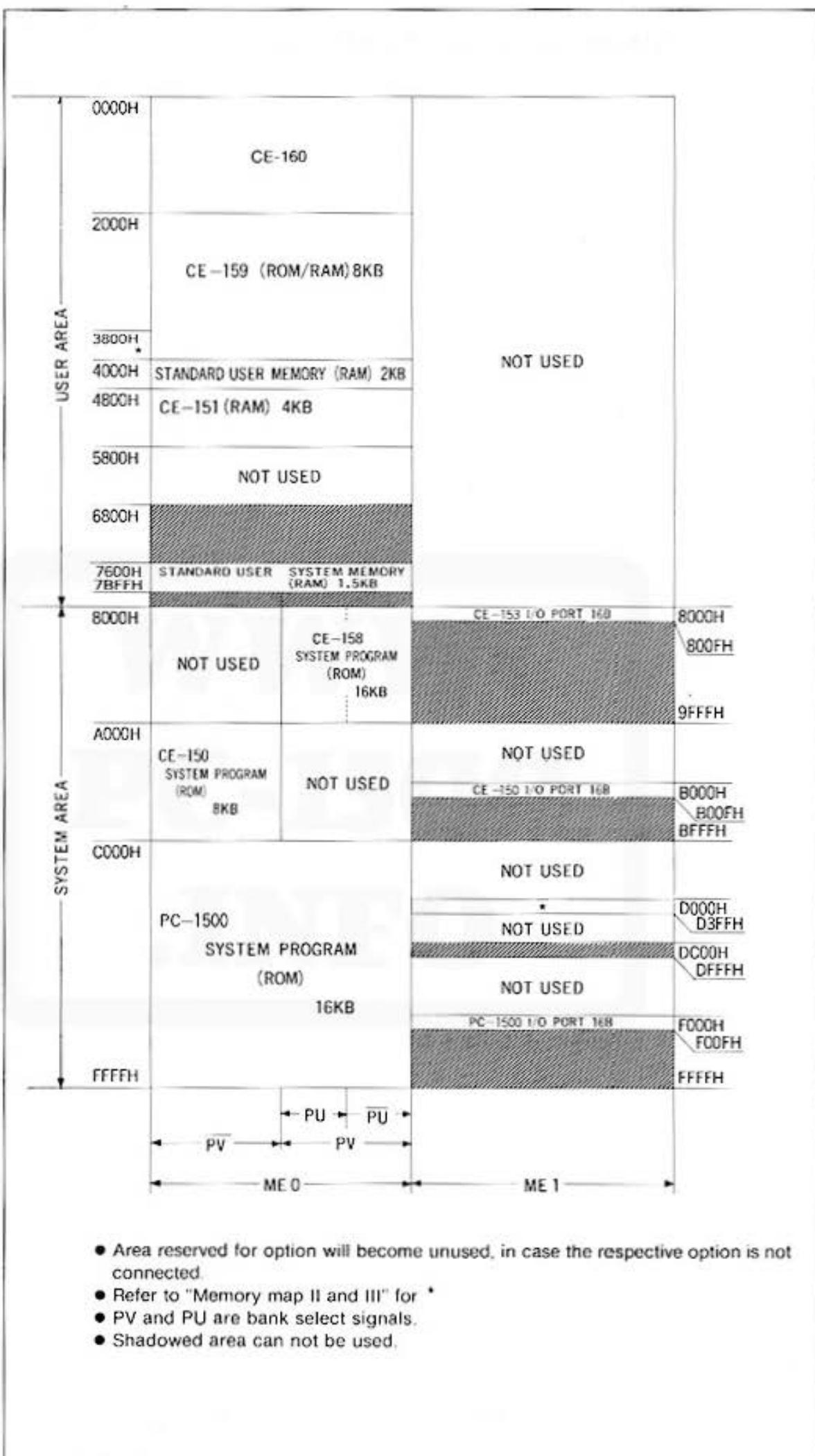
		0000H	
Y0 (<u>Y0</u>)		3FFFH	OPTIONAL USER MEMORY
Y1 (<u>Y1</u>)	S0 (<u>Y0</u>)	4000H 47FFH	STANDARD USER MEMORY
	S1 (<u>Y1</u>)	4800H 4FFFH	
	S2 (<u>Y2</u>)	5000H 57FFH	
	S3 (<u>Y3</u>)	5800H 5FFFH	OPTIONAL USER MEMORY
	S4 (<u>Y4</u>)	6000H 67FFH	
	S5 (<u>Y5</u>)	6800H 6FFFH	
	S6 (<u>Y6</u>)	7000H 75FFH	INHIBITED
	V2 (<u>2Y2</u>)	7600H 76FFH	
Y2 (<u>Y2</u>)	V3 (<u>2Y2</u>)	7700H 77FFH	STANDARD USER AND SYSTEM MEMORY
	S7 (<u>Y7</u>)	7800H 7FFFH	INHIBITED
		8000H 8FFFH	CE-150 SYSTEM PROGRAM, I/O PORT CE-153 I/O PORT CE-158 SYSTEM PROGRAM
Y3 (<u>Y3</u>)		C000H FFFFH	PC-1500 SYSTEM PROGRAM I/O PORT CE-158 I/O PORT UART

NOTE: S0~S7, V2, and
V3 are applicable only
for ME0 area.

4-2-4. PC-1500 system memory map

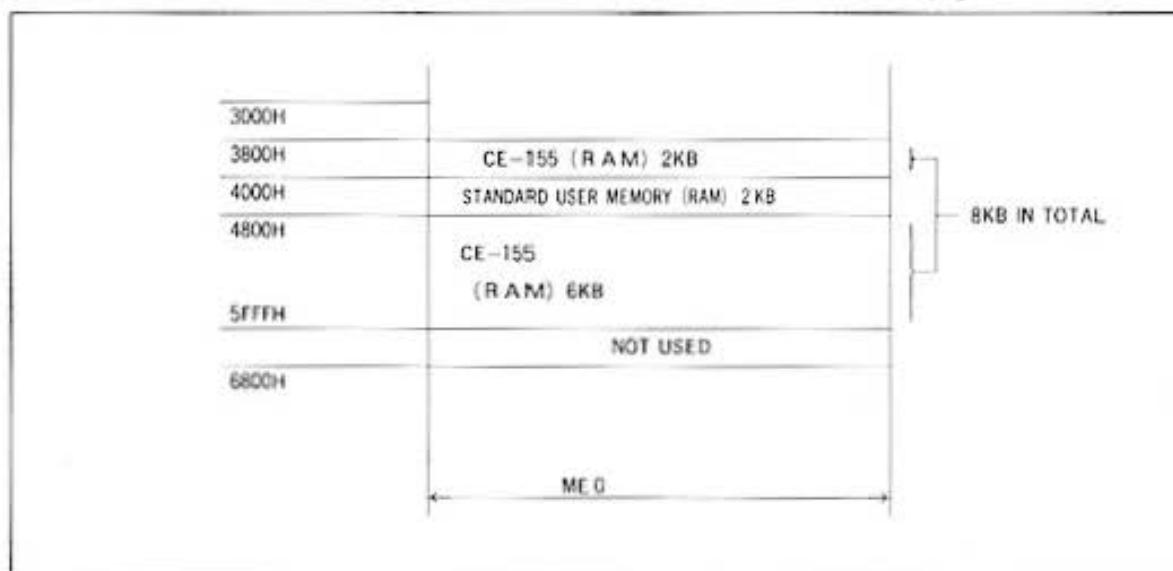
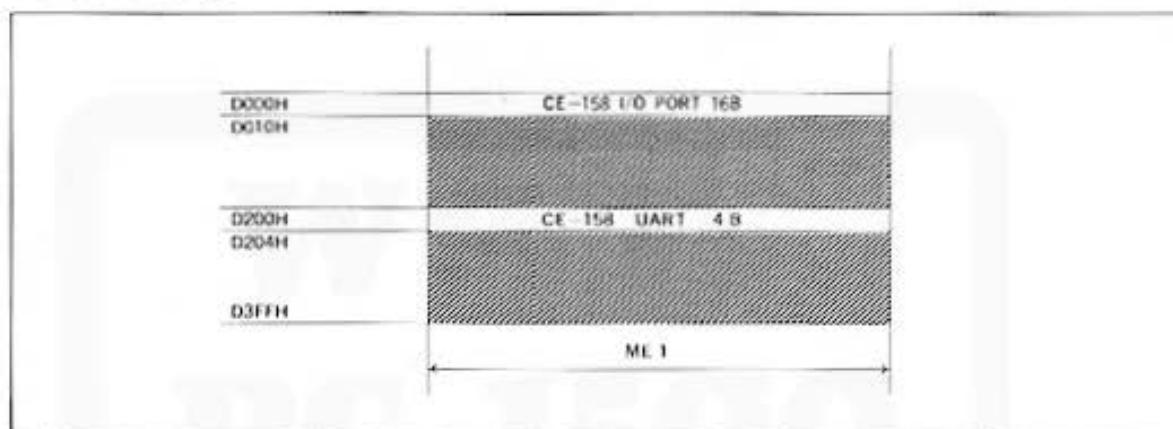
- ① In the ME0 area is contained such as the system program and user RAM area and the ME1 is used for such as I/O port.
Memory bank is assigned by PV and \overline{PV} for the ME0 area of 8000H thru BFFFH, and PU and \overline{PU} are used to assign the PV area of 8000H thru 9FFFH. Address 0000H thru 7FFFH is used for the user area and 8000H thru FFFFH for the system area.
The system area is used by the mnemonic programming system by which the CPU, I/O PC, etc. are controlled, and the user area is used to store the program written in BASIC, mnemonic programming language, and data.
With the PC-1500, ME0 memory area 4000H thru 47FFH and 7600H thru 7BFFH are used for the user memory and C000H thru FFFFH for the system program.
When the CE-150 or CE-158 is connected, the system program, I/O port, and RAM are arranged as shown in the next page.
- ② Upon power on to the PC-1500, initialization starts for peripheral units according to the system program residing in the system area, and the user area ROM and RAM are checked, then it becomes ready for a key entry.

MEMORY MAP I



MEMORY MAP II**MEMORY MAP when the CE-155 is used.**

(Note: For MEMORY MAP when the CE-161 is used, refer to page 160.)

**MEMORY MAP III****MEMORY MAP IV STANDARD USER & SYSTEM MEMORY (RAM) 1.5KB**

7600H~76FFH

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
HIGH															
LOW															
0															
2															
4															
6	DISPLAY	E	F	G	H	I	J	K	L	M	N	O			
8	BUFFER	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$			
A															
C															
E															

7700H~77FFH

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
HIGH															
LOW															
0															
2															
4															
6	DISPLAY	P	Q	R	S	T	U	V	W	X	Y	Z			
8	BUFFER	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$			
A															
C															
E															

7800H~78FFH

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
HIGH															
LOW															
0															
2															
4															
6	SYSTEM		SYSTEM		A	B	C	D							
8	STACK		MEMORY		\$	\$	\$	\$							
A															
C															
E															

7900H~79FFH

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
HIGH															
LOW															
0															
2			A	C	E	G	I	K	M	O	Q	S	U	W	Y
4															
6															
8															
A			B	D	F	H	J	L	N	P	R	T	V	X	Z
C															
E															

SYSTEM MEMORY

7A00H~7AFFH

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
HIGH																LOW
0																
2	X	Y	Y	V	S											
4																
6																
8																
A	Z	U	W													
C																
E																

BASIC STACK

7B00H~7BFFH

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
HIGH																LOW
0																
2																
4																
6																
8																
A																
C																
E																

SYSTEM MEMORY STRING BUFFER OUTPUT BUFFER INPUT BUFFER

- A~Z (7900H~79CFH) are fixed numerical variables.
- A\$~Z\$ are fixed character variables.
- The arithmetic registers X, Y, V, S, Z, U and W (7A00H~7A37H) are different from fixed numerical variables.
- BASIC stack includes stacks for FOR—NEXT, GOSUB, DATA, FUNCTION.
- See the next page for details of the display buffer.
- For system memory, refer to MEMORY MAP VII.

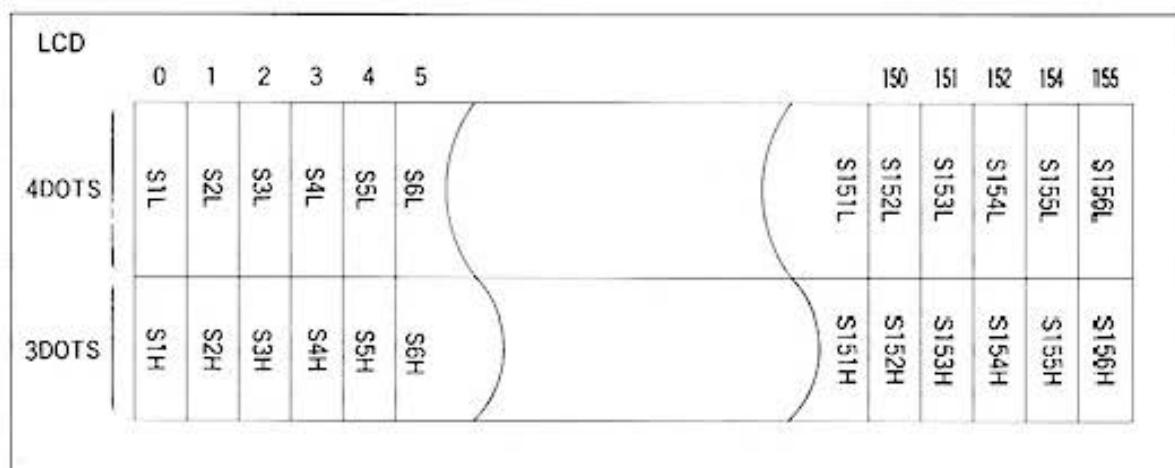
MEMORY MAP V

I/O port

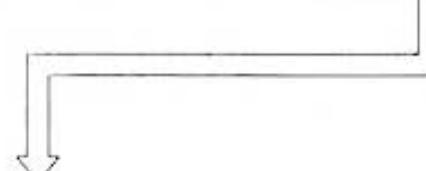
	PC-1500	CE-150	CE-153	CE-158
Do not use	F000H	B000H	8000H	D000H
Do not use	F001H	B001H	8001H	D001H
Do not use	F002H	B002H	8002H	D002H
Do not use	F003H	B003H	8003H	D003H
Divider reset	F004H	B004H	8004H	D004H
U register output	F005H	B005H	8005H	D005H
Serial transfer	F006H	B006H	8006H	D006H
Load divider to F register	F007H	B007H	8007H	D007H
Port C input/output	F008H	B008H	8008H	D008H
G register input/output	F009H	B009H	8009H	D009H
MSK register input/output	F00AH	B00AH	800AH	D00AH
IF register input/output	F00BH	B00BH	800BH	D00BH
Specify port A I/O direction	F00CH	B00CH	800CH	D00CH
Specify port B I/O direction	F00DH	B00DH	800DH	D00DH
Port A input/output	F00EH	B00EH	800EH	D00EH
Port B input/output	F00FH	B00FH	800FH	D00FH

Memory map VI**Display buffer**

Since there are 7×156 dots with 14 additional dots of symbols (DEF, I, PRO, etc.), each dot is allocated with a corresponding bit of the memory in order to control activation of dots. Address is represented by the hexadecimal notation and "H" is used to represent high state and "L" to represent low state of segment drive signals.



	760-		761-		762-		763-		764-	
	HIGH ORDER 4 BITS	LOW ORDER 4 BITS								
0	S79L	S1L	S87L	S9L	S95L	S17L	S103L	S25L	S111L	S31L
1	S79H	S1H	S87H	S9H	S95H	S17H	S103H	S25H	S111H	S31H
2	S80L	S2L								
3	S80H	S2H								
4	81	3								
5	81	3								
6	82	4								
7	82	4								
8	83	5								
9	83	5								
A	84	6								
B	84	6								
C	85	7							S117L	S39L
D	85	7							S117H	S39H
E	S86L	S8L	S94L	S16L	S102L	S24L	S110L	S30L		
F	S86H	S8H	S94H	S16H	S102H	S24H	S110H	S30H		



ADDRESS	HIGH ORDER 4 BITS				LOW ORDER 4 BITS			
764EH	DEF	I	II	III	SMALL	SML	SHIFT	BUSY
764FH	NOT USED	RUN	PRO	RESERVE	NOT USED	RAD	G	DE

	770-		771-		772-		773-		774-	
	HIGH ORDER 4 BITS	LOW ORDER 4 BITS								
0	S118L	S40L	S126L	S48L	S134L	S56L	S142L	S64L	S150L	S72L
1	S118H	S40H	S126H	S48H	S134H	S56H	S142H	S64H	S150H	S72H
2										
3										
4										
5										
6										
7										
8										
9										
A										
B										
C									S156L	S78L
D									S156H	S78H
E	S125L	S47L	S133L	S55L	S141L	S63L	S149L	S71L		
F	S125H	S47H	S133H	S55H	S141H	S63H	S149H	S71H		NOT USED

Memory Map VII

MEMORY SYSTEM DETAIL

78XX

	5	6	7	8	9	A	B
0				FOR POINTER	PREVIOUS ADDRESS H	BREAK TOP H	
1			WAIT Y/N	GOSUB POINTER	PREVIOUS ADDRESS L	BREAK TOP L	
2			WAIT CU		PREVIOUS LINE H	ERROR ADDRESS H	
3			WAIT CL		PREVIOUS LINE L	ERROR ADDRESS L	
4				STRING BUFFER POINTER	PREVIOUS TOP H	ERROR LINE H	
5			CURSOR EN		PREVIOUS TOP L	ERROR LINE L	
6			CURSOR POINTER	USING F/F	SEARCH ADDRESS H	ERROR TOP H	
7					SEARCH ADDRESS I	ERROR TOP L	
8				USING &	SEARCH LINE H	ON ERROR ADDRESS H	
9				USING M	SEARCH LINE L	ON ERROR ADDRESS L	
A				USING & F/F	SEARCH TOP H	ON ERROR LINE H	
B			RMT/BEEP	ERN	SEARCH TOP L	ON ERROR LINE L	
C				CURRENT LINE H	BREAK ADDRESS H	ON ERROR TOP H	
D			BRK CHARACTOR	TRACE	CURRENT LINE L	BREAK ADDRESS L	ON ERROR TOP L
E			CURSOR H	TRACE CONDITION	CURRENT TOP H	BREAK LINE H	DATA POINTER H
F			CURSOR L	OUTPUT BUFFER POINTER	CURRENT TOP L	BREAK LINE L	DATA POINTER L

79XX

	D	E	F	0
0	USER COUNTER XH		GRAPH/TEXT	
1	OPN DV	USER COUNTER XL		RND
2		USER COUNTER VH	ROTATE	NUMBER
3		USER COUNTER VL	COLOR	
4		SCISSORING COUNTER VH	CSIZE	
5		SCISSORING COUNTER VL		
6		ABSOLUTE POSITION X		
7		SCISSORING COUNTER XL		
8		SCISSORING COUNTER XH		
9				
A		LINE TYPE		
B		SKY LINE COUNTER		AUTO POWER OFF COUNTER U
C		UP/DOWN		AUTO POWER OFF COUNTER M
D		X MOTOR HOLD COUNTER		AUTO POWER OFF COUNTER L
E		PORT C		
F		Y MOTOR HOLD COUNTER	LOCK	

Address	Name	Function
786BH	RMT/BEEP	Remote and beep on/off pointer
7871H	WAIT Y/N	WAIT(0), WAIT0(3), WAIT1(2)
7872H	WAIT COUNTER H	WAIT counter
7873H	WAIT COUNTER L	
7874H	CURSOR ENABLE	(01H) if used, (00H) if not.
7875H	CURSOR POINTER	Cursor pointer (0~155)
787DH	BLINK CHARACTER	Character code to be blinked.
787EH	BLINK CURSOR H	Blinking cursor position (address of the display buffer)
787FH	BLINK CURSOR L	
788DH	TRACE	Trace on/off pointer
788EH	TRACE CONDITION	Status when trace on.
788FH	OUTPUT BUFFER POINTER	Output buffer pointer
7890H	FOR POINTER	FOR-NEXT stack pointer
7891H	GOSUB POINTER	GOSUB pointer
7894H	STRING BUFFER POINTER	String buffer pointer
7895H	USING F/F	Using format (presence of decimal point, comma, etc.)
7896H	USING M	Integer part of Using
7897H	USING &	Using of character string
7898H	USING m	Decimal part of Using
7899H	VARIABLE POINTER H	Variable pointer
789AH	VARIABLE POINTER L	
789BH	ERL	Error number when occurred.
789CH	CURRENT LINE H	Current line number
789DH	CURRENT LINE L	
789EH	CURRENT TOP H	Leading address of program of the current line
789FH	CURRENT TOP L	
78A0H	PREVIOUS ADDRESS H	Address of immediately preceding line
78A1H	PREVIOUS ADDRESS L	
78A2H	PREVIOUS LINE H	Line number immediately preceding
78A3H	PREVIOUS LINE L	
78A4H	PREVIOUS TOP H	Leading address of program of the line immediately preceding
78A5H	PREVIOUS TOP L	
78A6H	SEARCH ADDRESS H	Address of the line found during search
78A7H	SEARCH ADDRESS L	
78A8H	SEARCH LINE H	Line number found after search
78A9H	SEARCH LINE L	
78AAH	SEARCH TOP H	Leading address of the searched program block
78ABH	SEARCH TOP L	
78ACH	BREAK ADDRESS H	Address of breakpoint
78ADH	BREAK ADDRESS L	
78AEH	BREAK LINE H	Address of breakpoint line number
78AFH	BREAK LINE L	
78B0H	BREAK TOP H	Top address of the program block to which break is applied.
78B1H	BREAK TOP L	
78B2H	ERROR ADDRESS H	Address where error is met.
78B3H	ERROR ADDRESS L	
78B4H	ERROR LINE H	Line number where error is met.
78B5H	ERROR LINE L	

Address	Name	Function
78B6H	ERROR TOP H	Leading address of the program block in which error is met.
78B7H	ERROR TOP L	
78B8H	ON ERROR ADDRESS H	Address to which program jumps when an error is met.
78B9H	ON ERROR ADDRESS L	
78BAH	ON ERROR LINE H	Line number to which program jumps when an error is met.
78BBH	ON ERROR LINE L	
78BCH	ON ERROR TOP H	Leading address of program block in which an error is met.
78BDH	ON ERROR TOP L	
78BEH	DATA POINTER H	Pointer for data statement
78BFH	DATA POINTER L	
79D1H	OPN DV	Peripheral device select
79E0H	USER COUNTER XH	Counter by which X-coordinates of the pen are indicated.
79E1H	USER COUNTER XL	
79E2H	USER COUNTER YH	Counter by which Y-coordinates of the pen are indicated.
79E3H	USER COUNTER YL	
79E4H	SCISSORING COUNTER YH	Y-direction scissoring counter
79E5H	SCISSORING COUNTER YL	
79E6H	ABSOLUTE POSITION X	X-direction absolute point counter
78E7H	SCISSORING COUNTER XL	X-direction scissoring counter
79E8H	SCISSORING COUNTER XH	
79EAH	LINE TYPE	Kind of line
79EBH	DOT LINE COUNTER	Dot line counter
79ECH	UP/DOWN	Pen up/down position select
79EDH	X MOTOR HOLD COUNTER	X motor hold counter
79EEH	PORT C	Indicates current motor phase
79EFH	Y MOTOR HOLD COUNTER	Y motor hold counter
79F0H	GRAPH/TEXT	Printer mode select (graph=255, text=0)
79F2H	ROTATE	Printing direction select
79F3H	COLOR	Color select
79F4H	CSIZE	Printing character size select
79FFH	LOCK	Lock/unlock select
7800H	RND NUMBER	Random number
7801H	RND NUMBER	
7802H	RND NUMBER	
7803H	RND NUMBER	
7804H	RND NUMBER	
7805H	RND NUMBER	
7806H	RND NUMBER	
7807H	RND NUMBER	
780AH	AUTO P-OFF COUNTER U	Auto power off counter
780BH	AUTO P-OFF COUNTER M	
780CH	AUTO P-OFF COUNTER L	

4-3. Connector signals/LSI signals

Note that there may be a different kind of connector used for the 40 and 60 pin connector of the PC-1500 on account of product revision. The 64-pin connector on the back of the CE-150 is compatible with the 60-pin connector of the PC-1500. Either the LH5811 or LH5810 is used for the I/O PC. The LH5811 is an upgraded version of the LH5810.

4-3-1. 40-pin connector

Pin no.	Signal name	Description
1	VCC	VCC
2	PV	Chip select
3	PU	Chip select
4	Y0	Address designation of 0000H~3FFFH
5	S4	Address designation of 6000H~6700H
6	DME0	Chip select with WAIT condition in consideration (MEO area assignment)
7	D7	Data bus
8	D6	Data bus
9	D5	Data bus
10	D4	Data bus
11	D3	Data bus
12	D2	Data bus
13	D1	Data bus
14	D0	Data bus
15	INHIBIT	Prohibits ROM select of the PC-1500, when connected to GND.
16	S1	Address designation of 4800H~4FFFH
17	S2	Address designation of 5000H~57FFFH
18	S3	Address designation of 5800H~5FFFFH
19	Y2	Address designation of 8000H~BFFFH
20	VGG	VGG
21	GND	GND
22	AD15	Address bus
23	AD14	Address bus
24	AD13	Address bus
25	AD12	Address bus
26	AD11	Address bus
27	AD10	Address bus
28	AD9	Address bus
29	AD8	Address bus
30	AD7	Address bus
31	AD6	Address bus
32	AD5	Address bus
33	AD4	Address bus
34	AD3	Address bus
35	AD2	Address bus
36	AD1	Address bus
37	AD0	Address bus
38	OD	Output disable
39	R/W	Memory read/write
40	GND	GND

NOTE: S4 of No. 5 may be NC, depending on production month.

4-3-2. 60-pin connector

Pin no.	Signal name	Description
1	AD7	Address bus
2	AD6	Address bus
3	AD5	Address bus
4	AD4	Address bus
5	AD3	Address bus
6	AD2	Address bus
7	AD1	Address bus
8	AD0	Address bus
9	PB0	Not used
10	PC7	Not used
11	VCC	VCC
12	VCC	VCC
13	NC	NC
14	NC	NC
15	PV	Chip select
16	PU	Chip select
17	D7	Data bus
18	D6	Data bus
19	D5	Data bus
20	D4	Data bus
21	D3	Data bus
22	D2	Data bus
23	D1	Data bus
24	D0	Data bus
25	INHIBIT	Prohibits ROM select of the PC-1500, when connected to GND.
26	WEX	External WAIT signal
27	CMTIN	Cassette data input
28	W1	WAIT condition input
29	CMTOUT	Cassette data output
30	INT	Interrupt request to CPU
31	AD8	Address bus
32	AD9	Address bus
33	AD10	Address bus
34	AD11	Address bus
35	AD12	Address bus
36	AD13	Address bus
37	AD14	Address bus
38	AD15	Address bus
39	PB1	Not used
40	NC	NC
41	VCC	VCC
42	VCC	VCC
43	F-GND	Frame GND
44	VBAT	VBAT
45	VBAT	VBAT
46	VBAT	VBAT
47	VBAT	VBAT
48	VBAT	VBAT
49	NC	NC
50	BFO	VCC output
51	ΦOS	Clock in the same phase as the LSI internal clock
52	GND	GND
53	GND	GND
54	GND	GND
55	GND	GND
56	DME0	Chip select taking consideration of WAIT condition (ME0 area designation)
57	R/W	Memory read/write signal
58	DME1	Chip select taking consideration of WAIT condition (ME1 area designation)
59	ME1	ME1 area designation
60	OD	Output disable

NOTE: PB0 of No. 9 may be NC depending on production month.

PB1 of No. 39 may be NC depending on production month.

4-3-3. LH5801 Microprocessor

Pin no.	Signal name	Description
1	RESET	Reset input
2	NC	NC
3	BRO	Fixed to GND level
4	BFI	[x] key input signal
5	VGG	VGG
6	BFO	VCC output
7	OPF	Not used
8	BAK	Not used
9	VCC	VCC (4.0~4.7V)
10	VGG	VGG (4.0~4.7V)
11	VM	LCD power supply
12	VDIS	LCD power supply
13	VA	LCD power supply
14	VB	LCD power supply
15	NMI	GND
16	MI	Maskable interrupt input
17	HIN	LCD backplate signal
18	HA	LCD backplate signal
19	DISP	LCD on/off control
20	H7	Not used
21	H6	Backplate control
22	H5	Backplate control
23	H4	Backplate control
24	H3	Backplate control
25	H2	Backplate control
26	H1	Backplate control
27	H0	Backplate control
28	OD	Output disable
29	ME0	ME0 area designation
30	ME1	ME1 area designation
31	DO	Data bus
32	D1	Data bus
33	D2	Data bus
34	D3	Data bus
35	D4	Data bus
36	D5	Data bus
37	D6	Data bus
38	D7	Data bus
39	AD0	Address bus
40	AD1	Address bus
41	AD2	Address bus
42	AD3	Address bus
43	AD4	Address bus
44	AD5	Address bus
45	AD6	Address bus
46	AD7	Address bus
47	GND	GND
48	AD8	Address bus
49	VGG	VGG
50	AD9	Address bus
51	AD10	Address bus
52	AD11	Address bus
53	AD12	Address bus
54	AD13	Address bus
55	AD14	Address bus
56	AD15	Address bus
57	NC	NC
58	R/W	Memory read/write
59	P0	Not used
60	PV	Chip select
61	PU	Chip select
62	φOS	Clock which is in the same phase as the LSI internal clock
63	XLO	2.6MHz oscillator input
64	XL1	2.6MHz oscillator output
65	WAIT	CPU WAIT signal
66	IN7	Key input port
67	IN6	Key input port
68	IN5	Key input port
69	IN4	Key input port
70	IN3	Key input port
71	IN2	Key input port
72	IN1	Key input port
73	IN0	Key input port
74	NC	NC
75	NC	NC
76	NC	NC

4-3-4. I/O PC

When the pin No. 34 (ME0) of the I/O PC is connected with ME1 of the CPU, it will be accessed as the ME1 area.

① PC-1500 I/O PC

Pin no.	Signal name	Description
1	PA1	Key strobe
2	PA2	Key strobe
3	PA3	Key strobe
4	PA4	Key strobe
5	PA5	Key strobe
6	PA6	Key strobe
7	PA7	Key strobe
8	GND	GND
9	PB0	Not used
10	PB1	Not used
11	PB2	Serial data input from the cassette tape
12	PB3	VCC (export model), GND (domestic model)
13	PB4	GND
14	PB5	Timer control
15	PB6	Timer control
16	PB7	ON key input
17	P6	GND
18	PC0	Timer control
19	PC1	Timer control
20	PC2	Timer control
21	PC3	Timer control
22	PC4	Timer control
23	PC5	Timer control
24	PC6	Buzzer on/off control
25	PC7	Not used
26	CS0	Chip select (AD12)
27	CS1	Chip select (AD13)
28	CS2	Chip select
29	RS0	Chip select (AD0)
30	RS1	Chip select (AD1)
31	RS2	Chip select (AD2)
32	RS3	Chip select (AD3)
33	R/W	Memory read/write
34	ME0	ME0 area designation
35	ME1	ME1 area designation
36	W0	WAIT condition input
37	W1	WAIT condition input
38	GND	GND
39	VCC	VCC
40	DME0	Chip select taking consideration of WAIT condition (assignment of ME0 area).
41	DME1	Chip select taking consideration of WAIT condition (assignment of ME1 area).
42	WAIT	WAIT to CPU
43	INT	Interrupt to CPU
44	RESET	I/O port reset input
45	IRQ	Interrupt input
46	φOS	Basic clock input
47	CL1	CMT demodulation clock input
48	SD1	VCC level
49	LC	Not used
50	CL0	CMT signal
51	SD0	Serial data output to cassette tape
52	D0	Data bus
53	D1	Data bus
54	D2	Data bus
55	D3	Data bus
56	D4	Data bus
57	D5	Data bus
58	D6	Data bus
59	D7	Data bus
60	PA0	Key strobe

② CE-150 I/O PC

Pin no.	Signal name	Description
1	PA1	Remote 0 control
2	PA2	Remote 0 control
3	PA3	Remote 1 control
4	PA4	Remote 1 control
5	PA5	Manual print mode assignment
6	PA6	Not used
7	PA7	Not used
8	GND	GND
9	PB0	Pen ascending signal
10	PB1	Pen descending signal
11	PB2	Color detection
12	PB3	Not used
13	PB4	Not used
14	PB5	Not used
15	PB6	Low battery
16	PB7	Paper feed key input
17	P ₀	Not used
18	PC0	X motor control
19	PC1	X motor control
20	PC2	X motor control
21	PC3	X motor control
22	PC4	Y motor control
23	PC5	Y motor control
24	PC6	Y motor control
25	PC7	Y motor control
26	CS0	AD13 input/output
27	CS1	AD12 input/output
28	CS2	I/O port address designation
29	RS0	AD0
30	RS1	AD1
31	RS2	AD2
32	RS3	AD3
33	R/W	Memory read/write
34	ME0	ME0 area designation
35	ME1	GND
36	W0	GND
37	W1	WAIT condition output
38	GND	GND
39	VCC	VCC
40	DME0	Not used
41	DME1	Not used
42	WAIT	WAIT condition output
43	INT	Interrupt request to CPU
44	RESET	LH5811 reset
45	IRQ	Interrupt request from external source
46	ΦOS	Internal clock having the same phase as LSI
47	CL1	GND
48	SD1	GND
49	LC	Not used
50	CL0	Not used
51	SD0	Not used
52	D0	Data bus
53	D1	Data bus
54	D2	Data bus
55	D3	Data bus
56	D4	Data bus
57	D5	Data bus
58	D6	Data bus
59	D7	Data bus
60	PA0	For stable input level when CMT is not in use.

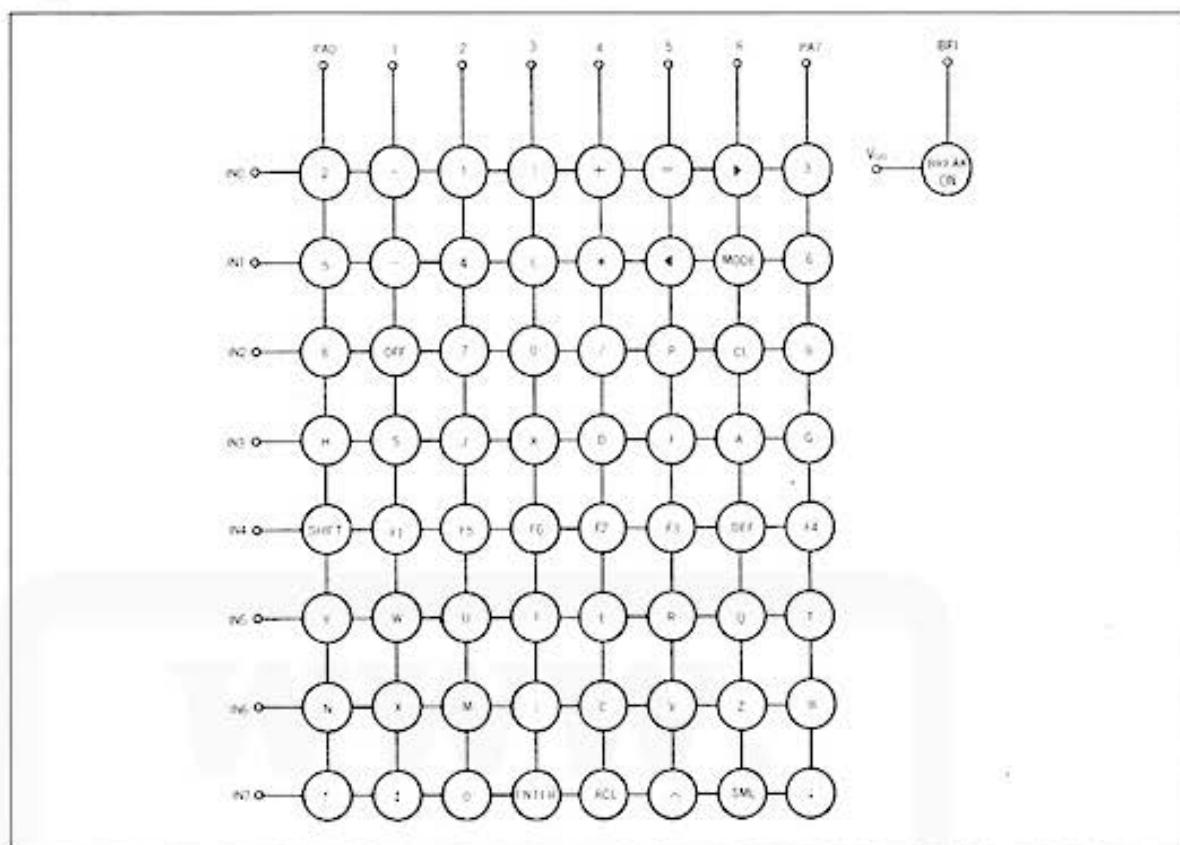
③ CE-153 I/O PC

Pin no.	Signal name	Description
1	PA1	Key input port
2	PA2	Key input port
3	PA3	Key input port
4	PA4	Key input port
5	PA5	Key input port
6	PA6	Key input port
7	PA7	Key input port
8	GND	GND
9	PB0	Key input port
10	PB1	Key input port
11	PB2	Key strobe
12	PB3	Key strobe
13	PB4	Key strobe
14	PB5	Key strobe
15	PB6	Key strobe
16	PB7	Key strobe
17	P _φ	GND
18	PC0	Key strobe
19	PC1	Key strobe
20	PC2	Key strobe
21	PC3	Key strobe
22	PC4	Key strobe
23	PC5	Key strobe
24	PC6	Key strobe
25	PC7	Key strobe
26	CS0	AD15
27	CS1	AD14
28	CS2	AD13
29	RS0	AD0
30	RS1	AD1
31	RS2	AD2
32	RS3	AD3
33	R/W	Memory read/write
34	ME0	ME0 area designation
35	ME1	GND
36	W0	GND
37	W1	GND
38	GND	GND
39	VCC	VCC (4.70±0.02V)
40	DME0	Not used
41	DME1	Not used
42	WAIT	WAIT condition
43	INT	Not used
44	RESET	Reset
45	IRQ	GND
46	ΦOS	Internal clock having the same phase as LSI.
47	CL1	GND
48	SD1	GND
49	LC	Not used
50	CL0	Not used
51	SD0	Not used
52	D0	Data bus
53	D1	Data bus
54	D2	Data bus
55	D3	Data bus
56	D4	Data bus
57	D5	Data bus
58	D6	Data bus
59	D7	Data bus
60	PA0	Key input port

④ CE-158 I/O PC

Pin no.	Signal name	Description
1	PA1	RS232C I/F send request
2	PA2	RS232C I/F ready to receive
3	PA3	RS232C I/F carrier detect
4	PA4	RS232C I/F data set ready
5	PA5	Low battery
6	PA6	Baud rate select
7	PA7	Baud rate select
8	GND	GND
9	PB0	Centronics parallel I/F DATA 2
10	PB1	Centronics parallel I/F DATA 3
11	PB2	Centronics parallel I/F DATA 4
12	PB3	Centronics parallel I/F DATA 5
13	PB4	Centronics parallel I/F DATA 6
14	PB5	Centronics parallel I/F DATA 7
15	PB6	Centronics parallel I/F DATA 8
16	PB7	Centronics parallel I/F BUSY input
17	P _φ	GND
18	PC0	Baud rate select
19	PC1	Baud rate select
20	PC2	Baud rate select
21	PC3	Baud rate select
22	PC4	Baud rate select
23	PC5	Centronics parallel I/F DATA 1
24	PC6	Centronics parallel I/F STROBE
25	PC7	Centronics parallel I/F INIT
26	CS0	VCC (4.70±0.02V)
27	CS1	VCC (4.70±0.02V)
28	CS2	GND
29	RS0	AD0
30	RS1	AD1
31	RS2	AD2
32	RS3	AD3
33	R/W	Memory read/write
34	ME0	ME0 area designation
35	ME1	GND
36	W0	GND
37	W1	GND
38	GND	GND
39	VCC	VCC (4.70±0.02V)
40	DME0	Not used
41	DME1	Not used
42	WAIT	Not used
43	INT	Interrupt request to CPU
44	RESET	LH5811 reset
45	IRQ	Interrupt request
46	ΦOS	Internal clock having the same phase as LSI.
47	CL1	GND
48	SD1	GND
49	LC	Not used
50	CL0	Not used
51	SD0	Not used
52	D0	Data bus
53	D1	Data bus
54	D2	Data bus
55	D3	Data bus
56	D4	Data bus
57	D5	Data bus
58	D6	Data bus
59	D7	Data bus
60	PA0	RS232C I/F terminal ready

4-4. Key matrix and key code chart

Key matrix**Key code chart**

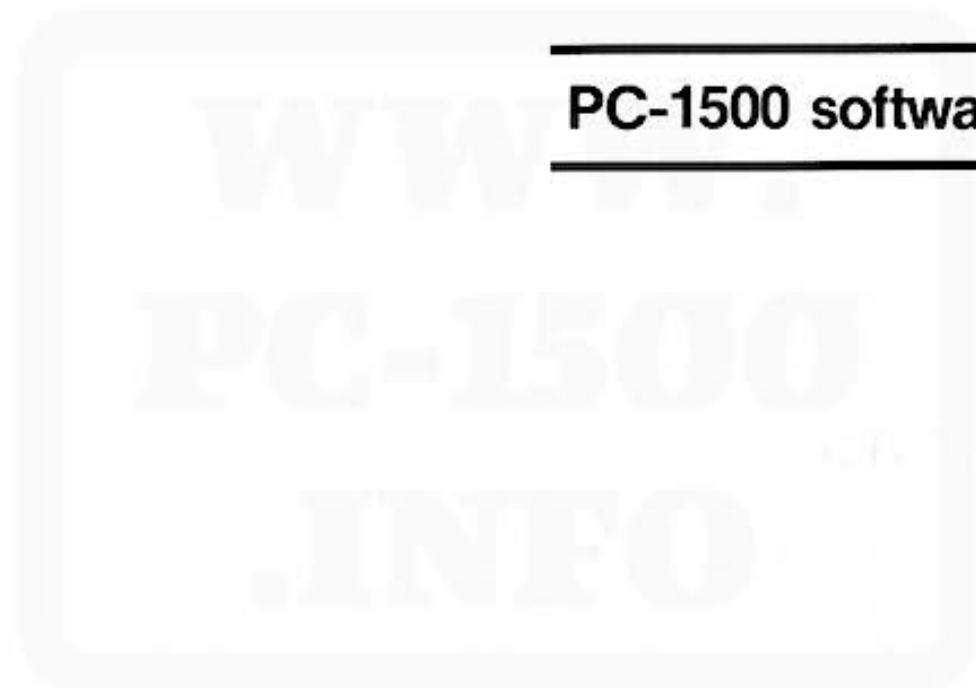
	0	1	2	3	4	5
0		SPACE	0	P		
1	SHIFT	F1		1	A	Q
2	SML	F2		2	B	R
3		F3		3	C	S
4		F4		4	D	T
5		F5		5	E	U
6		F6		6	F	V
7				7	G	W
8	◀	CL	(8	H	X
9	◆	RCL)	9	I	Y
A	↑		*		J	Z
B	↓	DEF	+		K	
C	▶				L	
D	ENTER		-	=	M	
E			.		N	
F	OFF	MODE	/		O	

NOTE: For ON key, refer to "KEY SCAN" of "SYSTEM SUBROUTINE".



5

PC-1500 software



5-1. BASIC command related PC-1500 machine language

In this section we will discuss the BASIC command related PC-1500 machine language.

5-1-1. NEW

Format (PRO mode)

NEW \leftarrow expression \downarrow

① NEW expression (=0)

Clears program and all data areas and secures the BASIC program area following to the reserve program area.

	Top of the BASIC program
Operation of the PC-1500 only	40C5H
Operation in conjunction with the CE-155	38C5H

See (System memory)

② NEW expression ($\neq 0$)

Effective when there is no user ROM area.

Clears BASIC program and all data areas and sets the BASIC program area from the address represented in the expression, which is valid within an address range of RAM area except the reserve area.

③ NEW

Clears BASIC program and all data areas without affecting the BASIC program area.

When executed in the reserve mode, it clears the reserve area.

5-1-2. STATUS

Format

Status \leftarrow expression \rightarrow

① STATUS 0

Similar to MEM, the sum of unoccupied memory size and data memory size is given in terms of bytes.

② STATUS 1

Size of the memory used for the BASIC program is given in terms of bytes.

③ STATUS 2

The last BASIC program address plus one address (next to FFH) is given.

④ STATUS 3

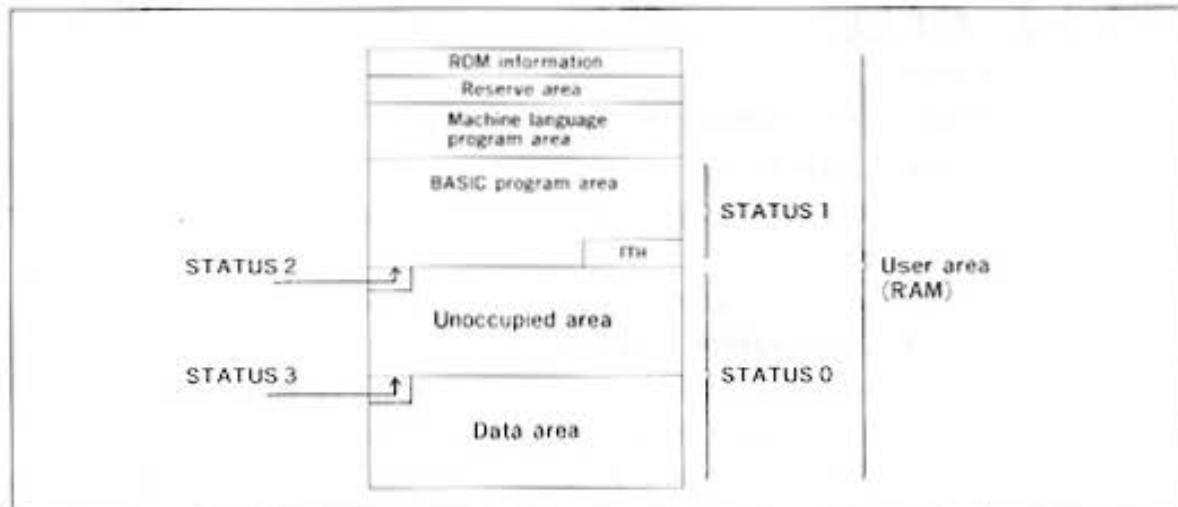
The smallest address where data stored is given.

⑤ STATUS 4

When BASIC program is in execution, the line number executed immediately before.

When program execution is at halt, the current line number.

In otherwise condition, "0". Same for STATUS 5 thru 255.



NOTE: ROM information is 8 bytes and the reserve area is 189 bytes.

5-1-3. PEEK

Format

PEEK \leftarrow # \downarrow expression →

① **PEEK expression**

Recalls the data in ME0 area whose address is represented by the expression.

② **PEEK # expression**

Recalls the data in the ME1 area whose address is represented by the expression.

5-1-4. POKE

Format

POKE \leftarrow # \downarrow expression-1 \leftarrow expression-2 \leftarrow expression-3 \dots

① **POKE expression-1, expression-2, expression-3, ...**

Starting from the ME0 area address represented by the expression-1, it begins to store successive data in order of the expression-2, expression-3, ...

② **POKE # expression-1, expression-2, expression3, ...**

Starting from the ME1 area address represented by the expression-1, it begins to store successive data in order of the expression-2, expression-3, ...

CAUTION: Since the I/O port controller is in the ME1 area for the PC-1500, care must be taken not to have a wrong use of this command, as it may possibly result in destruction.

(EX): POKE &4700, &01, &02, &03

Address	Data
4700H	01H
4701H	02H
4703H	03H

5-1-5. CALL

Format

CALL → expression ↗ , → variable ↗

① CALL expression

Machine language program starts to execute starting from address specified by the expression.

Program returns from the machine language routine by the command.

② CALL expression, variable

1) When the variable is a numerical variable (within a range of -32768 thru 32767)

1. The value of the variable is transferred to the Xreg.

2. Machine language program is executed from the starting address represented by the expression.

3. If there is a carry when returns, the value of the Xreg is transferred to the variable.

2) When the variable is a nonnumeric variable

1. The leading address of the nonnumeric variable is transferred to the Xreg and the size information of the nonnumeric variable is transferred to the accumulator.

2. Machine language program is executed from the starting address represented by the expression.

3. If there is a carry when returns, the character string whose size is indicated by the accumulator is transferred to the variable from the address represented by the Xreg.

NOTE: It will result in ERROR 7 for a two-nonnumeric variable whose variable has not been defined.

5-1-6. CSAVE M

Format

CSAVE M ↗ -1 ↗ "file name"; ↗ expression-1, expression-2 ↗ , expression 3 ↗

① CSAVE M expression-1, expression-2, expression-3

Data residing from the address represented by the expression-1 to the address represented by the expression-2 is recorded on the tape as the machine language program.

When the expression-3 is given, execution will automatically take place from the address represented by the expression-3 upon loading of the program from the tape. File name will also be recorded on the tape, when there is a file name.

② CSAVE M-1

Tape control will be set to the REMOTE-1 side.

5-1-7. CLOAD M

Format

CLOAD M ↗ -1 ↗ "file name"; ↗ expression ↗

① CLOAD M

The machine language program recorded on the tape is loaded into the same memory area as when recorded.

When the expression is given, program load will take place from the address represented by the expression.

If there have been the expression-3 during data save, program execution will automatically take place from the address represented by the expression-3 upon completion of program load. However, automatic execution will not start if the expression is given.

② CLOAD M-1

Tape control will be set to the REMOTE-1 side.

5-2. Internal code chart

With the PC-1500 system, BASIC command is converted into internal code of two bytes to be processed by the PC-1500 system.

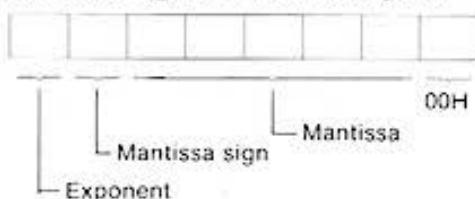
Shown next is the list of commands with corresponding internal codes.

COMMAND	INTERNAL CODE	COMMAND	INTERNAL CODE	COMMAND	INTERNAL CODE
ABS	F170H	LCURSOR	E683H	SETCOM	E882H
ACS	F174H	LEFT\$	F17AH	SETDEV	E886H
AND	F150H	LEN	F164H	SGN	F179H
AREAD	F180H	LET	F198H	SIN	F17DH
ARUN	F181H	LF	F0B6H	SORGN	E684H
ASC	F160H	LINE	F0B7H	SPACES\$	F061H
ASN	F173H	LIST	F090H	SQR	F16BH
ATN	F175H	L LIST	F0B8H	STATUS	F167H
BEEP	F182H	LN	F176H	STEP	F1ADH
BREAK	F0B3H	LOCK	F1B5H	STOP	F1ACH
CALL	F18AH	LOG	F177H	STR\$	F161H
CHAIN	F0B2H	LPRINT	F0B9H	TAB	F0BBH
CHR\$	F163H	MEM	F158H	TAN	F17FH
CLEAR	F187H	MERGE	F08FH	TERMINAL	E883H
CLOAD	F089H	MIDS\$	F17BH	TEST	F0BCH
CLS	F088H	NEW	F19BH	TEXT	E686H
COMS	E858H	NEXT	F19AH	THEN	F1AEH
CONSOLE	F0B1H	NOT	F16DH	TIME	F15BH
CONT	F183H	OFF	F19EH	TO	F1B1H
COLOR	F0B5H	ON	F19CH	TRANSMIT	E885H
COS	F17EH	OPN	F19DH	TROFF	F1B0H
CSAVE	F095H	OR	F151H	TRON	F1AFH
CSIZE	E680H	OUTSTAT	E880H	UNLOCK	F1B6H
CURSOR	F084H	PAUSE	F1A2H	USING	F085H
DATA	F18DH	PEEK	F16FH	VAL	F162H
DEG	F165H	PEEK#	F16EH	WAIT	F1B3H
DEGREE	F18CH	PI	F15DH	ZONE	F0B4H
DEV\$	E857H	POINT	F168H		
DIM	F18BH	POKE	F1A1H		
DMS	F166H	POKE#	F1A0H		
DTE	E884H	PRINT	F097H		
END	F18EH	RADIAN	F1AAH		
ERL	F053H	RANDOM	F1A8H		
ERN	F052H	READ	F1A6H		
ERROR	F1B4H	REM	F1ABH		
EXP	F178H	RESTORE	F1A7H		
FEED	F0B0H	RETURN	F199H		
FOR	F1A5H	RIGHT\$	F172H		
GCURSOR	F093H	RINKEY\$	E85AH		
GLCURSOR	E682H	RLINE	F0BAH		
GOSUB	F194H	RMT	E7A9H		
GOTO	F192H	RND	F17CH		
GPRINT	F09FH	ROTATE	E685H		
GRAD	F186H	RUN	F1A4H		
GRAPH	E681H				
IF	F196H				
INKEY\$	F15CH				
INPUT	F091H				
INSTAT	E859H				
INT	F171H				

5-3. Expression of variable and program

5-3-1. Expression of decimal number

Decimal number consists of eight bytes which are used to represent number within a range of $-9.99999999 \times 10^{99}$ to $+9.99999999 \times 10^{99}$ and are composed of the exponent part, mantissa sign and mantissa part.



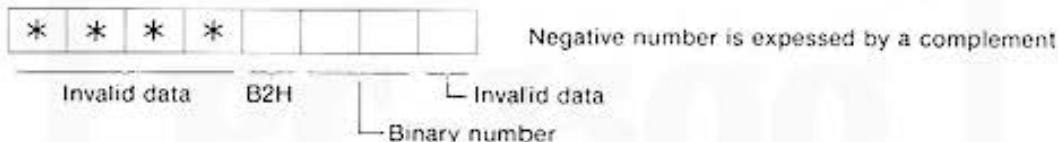
Exponent part: expressed in a binary number.
Negative number is represented by complement.
Mantissa sign: 00H (Positive)
80H (Negative)

(Example when numeric is stored from 7A00H to 7A07H)

7A00H								7A07H							
0 3 H	0 0 H	1 5 H	0 0 H	0 0 H	0 0 H	0 0 H	0 0 H	1 500							
0 0 H	0 0 H	1 2 H	3 4 H	5 6 H	0 0 H	0 0 H	0 0 H	1.23456							
F DH	0 0 H	1 2 H	3 4 H	5 6 H	7 8 H	9 0 H	0 0 H	0.00123456789012							
0 8 H	8 0 H	1 2 H	3 4 H	0 0 H	0 0 H	0 0 H	0 0 H	-1.234×10 ⁶							

5-3-2. Expression of binary number

Binary number consists of eight bytes, but five bytes are not used. It expresses a binary number within an integer range of -32768 thru +32767.



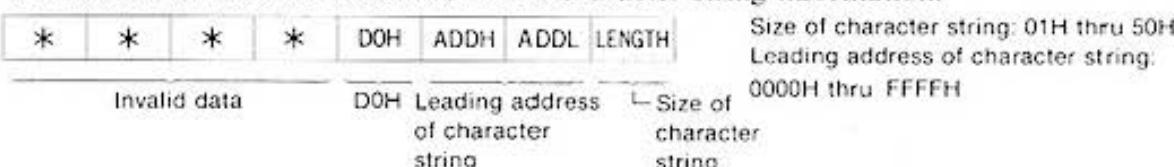
(Example when numeric is stored from 7A00H to 7A07H)

7A00H								7A07H							
*	*	*	*	B 2 H	0 5 H	D C H	*	1500							
*	*	*	*	B 2 H	F F H	F B H	*	-5							
*	*	*	*	B 2 H	7 F H	F F H	*	32767							
*	*	*	*	B 2 H	8 0 H	0 0 H	*	-32768							

NOTE: Noted with an asterisk (*) indicates invalid data.

5-3-3. Expression of character string

Character string information is composed of eight bytes (with four bytes of valid data) and it resides in the address contained in the character string information.

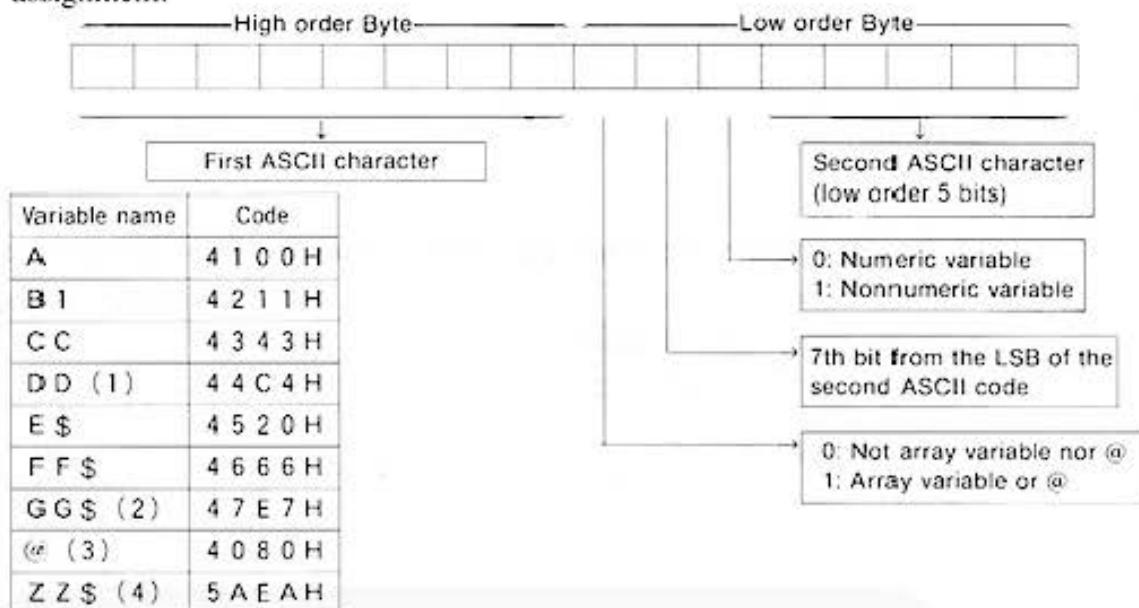


Example that the character string information resides in the arithmetic register and the character string "SHARP" resides in the string buffer.

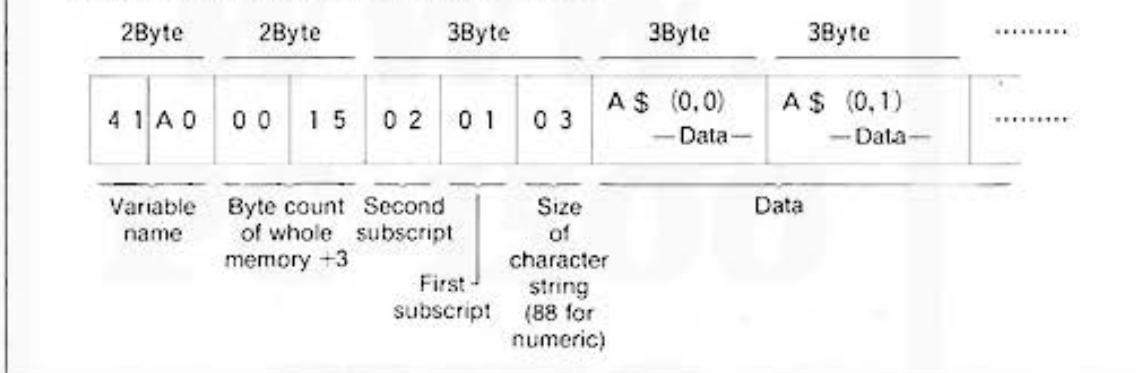
7A00H								7A07H							
*	*	*	*	D0H	7BH	10H	05H								
7B10H															
53H	48H	41H	52H	50H	S	H	A	R	P						

5-3-4. Structure of variable name

Variable name field is made of two bytes which consists of variable name composed of ASCII character, distinction of numeric and nonnumeric variable, and presence of array assignment.

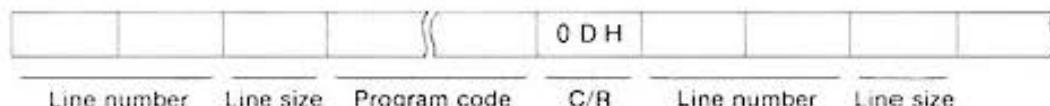


(EX) When array declared to "AS(1,2) * 3"



5-3-5. Structure of program

Each of program lines is composed of the line number, line size, program code, and end code.



```
10 PRINT A
20 END
```

The above programmed lines will be as follows (without (module)):

Address	Data
4 0 C 5 H	0 0 H
4 0 C 6 H	0 A H
4 0 C 7 H	0 4 H
4 0 C 8 H	F 0 H
4 0 C 9 H	9 7 H
4 0 C A H	4 1 H
4 0 C B H	0 D H
4 0 C C H	0 0 H
4 0 C D H	1 4 H
4 0 C E H	0 3 H
4 0 C F H	F 1 H
4 0 D 0 H	8 A H
4 0 D 1 H	0 D H
4 0 D 2 H	F F H

| 10
-----Line size
| PRINT
-----A
-----C/R
| 20
-----Line size
| END
-----C/R
-----Code to indicate end of BASIC program.

5-3-6. Structure of reserve area

① Leading address of the reserve area

System configuration	Leading address
PC-1500 ONLY	4000H
PC-1500+CE-151	4000H
PC-1500+CE-155	3800H
PC-1500+CE-159	2000H

② Reserve memory configuration (when only the PC-1500 is used)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4 0 0																
4 0 1																
4 0 2																
4 0 3																
4 0 4																
4 0 5																
4 0 6																
4 0 7																
4 0 8																
4 0 9																
4 0 A																
4 0 B																
4 0 C																
4 0 D																
4 0 E																
4 0 F																

NOTE: When NEW [NEW] is executed in the RESERVE mode, address area from 4008H to 40C4H is filled up with "00H".

③ Reserve key code

RESERVE NO	I	II	III
F 1	0 1 H	1 1 H	0 9 H
F 2	0 2 H	1 2 H	0 A H
F 3	0 3 H	1 3 H	0 B H
F 4	0 4 H	1 4 H	0 C H
F 5	0 5 H	1 5 H	0 D H
F 6	0 6 H	1 6 H	0 E H

- Reserve key contents are stored following to the key code.
- Reserve programs are stored in order of registration. In the case of re-registration, the previous program is deleted and the new program is added following to it.

④ ROM status information

ADDRESS	DESCRIPTION
1st Byte in the ROM	55H
2nd Byte in the ROM	High order one byte of the ROM top address
3rd Byte in the ROM	High order one byte of the top address of the BASIC program, assuming the ROM top address to be "0000H"
4th Byte in the ROM	Low order one byte of the top address of the BASIC program, assuming the ROM top address to be "0000H"
5th Byte in the ROM	Write the following code according to the ROM size 1KB:"04H" 2KB:"08H" 4KB:"10H" 8KB:"20H" 16KB:"40H"
6th Byte in the ROM	Undefined
7th Byte in the ROM	Undefined
8th Byte in the ROM	To inhibit "LLIST" command, write "00H" To effect "LLIST" command, write "FFH"

Note : The ROM address is 0000H~3FFFH.

(An example of registration)**Key symbol**

Reserve No. I	P R T	I N P	G T O	G S B	R E T
Reserve No. II	S I N	C O S	T A N		
Reserve No. III	R U N	G T O			

Reserve contents

Registration order		Key	Registered contents
1	I	F 1	P R I N T
2	I	F 3	G O T O
3	I	F 2	I N P U T
4	I	F 4	G O S U B
5	I	F 5	R E T U R N
6	II	F 1	S I N
7	II	F 3	T A N
8	III	F 2	G O T O @
9	III	F 1	R U N @
10	II	F 2	C O S

4 0 0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
4 0 1	20	47	4F	54	20	47	53	42	20	52	45	54	20	00	00	00	
4 0 2	00	00	20	53	49	4E	20	43	4F	53	20	54	41	4E	20	00	
4 0 3	00	00	00	00	00	00	00	00	00	00	00	00	20	52	55	4E	
4 0 4	20	47	54	4F	20	00	00	00	00	00	00	00	00	00	00	00	
4 0 5	00	00	00	00	00	00	00	01	F0	97	03	F1	92	02	F0	91	04
4 0 6	F1	94	05	F1	99	11	F1	70	13	F1	7F	0A	F1	92	40	09	
4 0 7	F1	A4	40	12	F1	7E	00	00	00	00	00	00	00	00	00	00	
4 0 8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
4 0 9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
4 0 A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
4 0 B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
4 0 C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
4 0 D																	
4 0 E																	
4 0 F																	

Characters are stored in form of character code and commands in form of internal code.
"00H" is for the end code of data.

5-4. System subroutines

System subroutines used by the machine language program will be introduced next. However, care must be taken for printer related entry address for it differs depending on the ROM version of the CE-150. The version number will be indicated in the address A800H.

Version 0	Address A800H is 44H.		
Version 1	Address A800H is BEH.		

① Character function

Combination of character	D925H	VAL	D9D7H
CHRS	D9B1H	LEN, ASC	D9DDH
STRS	D9CFH	RIGHTS, MIDS, LEFTS	D9F3H

② Arithmetic operations

Subtract	EFB6H	10 ⁿ	F1D4H	DEG	F531H
Add	EFBAH	COS	F391H	DMS	F564H
Multiply	F01AH	TAN	F39EH	ABS	F597H
Divide	F084H	SIN	F3A2H	SGN	F59DH
LN	F161H	ACS	F492H	INT	F5BEH
LOG	F165H	ATN	F496H	Power raise	F89CH
EXP	F1CBH	ASN	F49AH		

③ Compare

Numerical comparison	D0D2H	Character string comparison	D0F9H
----------------------	-------	-----------------------------	-------

④ Search

Line number search	D2EAH	Variable search	D461H
KEY scan (I)	E42CH	KEY scan (II)	E243H

⑤ Display

Auto-power-off	E33FH	One character display	ED57H
Program display	E8CAH	"n" character display	ED3BH
Graphic display	EDEFH	Cursor move after one character display	ED4DH
Hexadecimal (2 bytes → 1 byte)	ED95H	Cursor move after "n" character display	ED00H

⑥ Printer related

Color designation	A519H(A4F7H)	Pen up/down	AAE3H(AABDH)
Print	A781H(A75BH)	Motor drive	A8DDH(A8B7H)
Linefeed	A9F1H(A9CBH)	Motor off	A769H(A747H)
Paper feed	AA04H(A9DEH)	Get GRAPHIC mode ready	ABEFH(ABC6H)
Get TEXT mode ready	ACBBH(AC8FH)		

(Figures in parentheses indicate version 0.)

⑦ Cassette tape

Remote on	BF11H	Header input/output	BCE8H
Remote off	BF43H	CMT I/O control	.BBF5H
One character save	BDCCH	Create header	BBD6H
One character load	BDF0H	Transfer file	BD3CH

5-4-1. Character functions

■ Combination of character string

Combination of character string-1 and character string-2

① ENTRY PREPARATION

- Preparation of character strings

Character string information of the character string-1 is stored in the arithmetic register (7A00H~7A07H).

Character string information of the character string-2 is stored in the arithmetic register (7A10H~7A17H).

- 10H is stored in the string buffer pointer (7894H).

② ENTRY ADDRESS

D925H

- Subroutine is called in the following format:

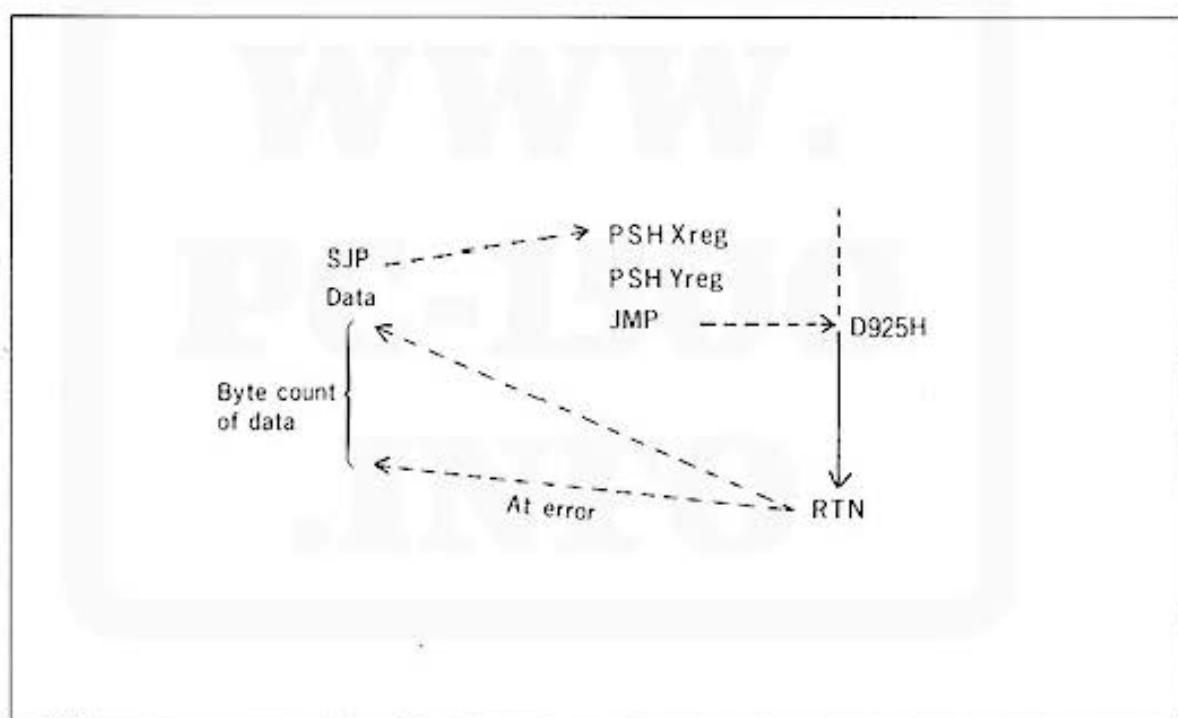
SJP (Address where PSH Xreg is written)

DB DATA (Error return address)

PSH Xreg

PSH Yreg | Write in other addresses.

JMP D925



③ EXIT STATE

- When no error

Returns to the address next to the data. Resultant character string information will then be stored in the arithmetic register X (7A00H~7A07H) and the combined character strings will be stored in the string buffer (7B10H~).

- When error

After storing the error code in the UH register, it returns to the data stored address plus one.

NOTE: For details of character string information, refer to the paragraph discussing the structure of variable and program.

■ CHRS**① ENTRY PREPARATION**

- Preparation of numeric value

An integer of 0 thru 255 is stored in the arithmetic register (7A00H~7A07H) in decimal or binary figure.

- 10H is stored in the string buffer pointer (7894H).

② ENTRY ADDRESS

D9B1H

③ EXIT STATE

- When no error

UH = 00H

Address	Contents
7A04H	C1H
7A05H	7BH
7A06H	10H
7A07H	00H or 01H
7B10H	ASCII code

When ASCII code is 00H, the contents of 7A07H become 00H or 01H when other than 00H.

- When error

UH ≠ 00H (Error code is stored in UH)

■ STRS**① ENTRY PREPARATION**

- Number to be converted is stored in the arithmetic register (7A00H~7A07H) in decimal or binary figure.
- 10H is stored in the string buffer pointer (7894H).

② ENTRY ADDRESS

D9CFH

③ EXIT STATE

- When no error
UH = 00H

Address	Data
7A04H	D0H
7A05H	Leading address of character string (high order one byte)
7A06H	Leading address of character string (low order one byte)
7A07H	Size of character string

For an actual character string, the string buffer (7B10H~7B5FH) can be used.

- When error
UH ≠ 00H (Error code will be stored in UH.)

■ VAL**① ENTRY PREPARATION**

- Character string information to be converted is stored in the arithmetic register X (7A00H~7A07H) in character string format.

② ENTRY ADDRESS

D9D7H

③ EXIT STATE

- When no error
UH = 00H
Result is stored in arithmetic register X (7A00H~7A07H) in decimal figure.
- When error
UH ≠ 00H (Error code will be stored in UH.)

■ ASC, LEN**① ENTRY PREPARATION**

- Character string information to be converted is stored in the arithmetic register X (7A00H~7A07H).

Address	Data
7A04H	D0H
7A05H	Leading address of character string (high order one byte)
7A06H	Leading address of character string (low order one byte)
7A07H	Size of character string

- Setup of function

Function	YL
ASC	60H
LEN	64H

② ENTRY ADDRESS

D9DDH

③ EXIT STATE

- When no error

UH=00H

Result is stored in arithmetic register X (7A00H ~ 7A07H) in decimal figure.

- When error

UH≠00H (UH=error code)

■ RIGHTS (Character string, numeric value-1)

LEFTS (Character string, numeric value-1)

MIDS (Character string, numeric value-2, numeric value-1)

① ENTRY PREPARATION

- Setup of function

Function	YL
RIGHTS	02H
LEFTS	7AH
MIDS	7BH

- Existence of availability for 8 bytes in the BASIC stack (7A38H~7AFFH) is checked.

- In the case of RIGHTS, LEFTS

(Data of 7890H) < (data of 7891H) - 8

- In the case of MIDS

(Data of 7890H) < (data of 7891H) - 16

Since re-write of 7890H and 7891H is not permitted, it must be avoided to call this subroutine, unless the above condition is satisfied.

- Change of the data pointer (7892H)

Function	Data
RIGHTS	(data of 7890H) + 8
LEFTS	
MIDS	(data of 7890H) + 16

- Preparation of character string

Address	Data
(Data of 7890H) + 4	D0H
(Data of 7890H) + 5	Leading address of character string (high order one byte)
(Data of 7890H) + 6	Leading address of character string (low order one byte)
(Data of 7890H) + 7	Size of character string

NOTE: High order byte of address is 7AH.

For actual character string, the string, the string buffer (7B10H~7B5FH) can be used.

- Preparation of numerical data

Numeric-1 is stored in 7A00H~7A07H in decimal or binary figure.

(In the case of MIDS, the numeric-2 is stored from the address of "data of 7890H" plus eight to the address "data of 7890H" plus fifteen.)

Stored in the above address in a decimal or binary format.

② ENTRY ADDRESS

D9F3H

③ EXIT STATE

- When no error

UH = 00H

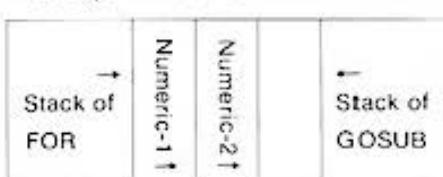
Address	Data
7A04H	D0H
7A05H	Leading address of character string (high order one byte)
7A06H	Leading address of character string (low order one byte)
7A07H	Size of character string

For an actual character string, the string buffer (7B10H~7B5FH) can be used.

- When error

UH ≠ 00H (UH is substituted with error code.)

NOTE:



Represented by the pointer (7890H) 8 bytes Represented by the pointer (7891H) 8 bytes

5-4-2. Arithmetic subroutines

① ENTRY PREPARATION

- Preparation of numeric value

Numeric value should be prepared in the arithmetic register X (7A00H~7A07H) and the arithmetic register Y (7A10H~7A17H).

(In the case of a single variable, only the arithmetic register needs to be prepared.)

② ENTRY ADDRESS

	Operation	Address
Two-variable function	Add X+Y → X	EFBAH
	Subtract X-Y → X	EFB6H
	Multiply X*Y → X	F01AH
	Divide X/Y → X	F084H
	Power raise X^Y → X	F89CH
	SQR X→X	F0E9H
	Square root LN X→X	F161H
	LOG X→X	F165H
	Logarithm EXP X→X	F1CBH
	10^X→X	F1D4H
Single-variable function	SIN X→X	F3A2H
	COS X→X	F391H
	TAN X→X	F39EH
	Inverse trigonometric ASN X→X	F49AH
	ACOS X→X	F492H
	ATN X→X	F496H
	Degree to minute and second conversion DEG X→X	F531H
	DMS X→X	F564H
	Absolute value ABS → X	F597H
	Sign SGN X→X	F59DH
	Conversion into an integer number INT X→X	F5BEH

③ EXIT STATE

- Result will be stored in the arithmetic register X (7A00H~7A07H).

5-4-3. Comparison

■ Comparison of numeric

(Numeric 1) ○ (numeric 2)

○ Indicates the operand.

① ENTRY PREPARATION

- Setup of operand

Operand	Accumulator data
<>	00H
<	01H
>	02H
=	04H
<=	05H
>=	06H

- The numeric is stored in the arithmetic register in a format of decimal figure.

Numeric 1	Arithmetic register X (7A00H~7A07H)
Numeric 2	Arithmetic register X (7A10H~7107H)

② ENTRY ADDRESS

D0D2H

③ EXIT STATE

- When the operand is established

The flag Z is reset to "0" and the arithmetic register turns to "1".

7A00H	00H	00H	10H	00H	00H	00H	00H	7A07H

- When the operand is not established

The flag Z is set to "1" and the arithmetic register turns to "0".

7A00H	00H	7A07H						

■ Comparison of character string

(Character string-1) ○ (character string-2)

○ Indicates the operand.

① ENTRY PREPARATION

- Designation of the operand

Operand	Accumulator data
<>	00H
<	01H
>	02H
-	04H

- Preparation of character string

Contents	Address	Character string-1	Character string-2
D0H		7A04H	7A14H
Leading address of the character string	(high order)	7A05H	7A15H
	(low order)	7A06H	7A16H
Size of the character string		7A07H	7A17H

For the address where the character string is stored, the string buffer (7B10H~7B5FH) can be used.

- 10H is stored in the string buffer pointer (7894H).

② ENTRY ADDRESS

D0F9H

③ EXIT STATE

- When the condition for the operand is established (Z=0)

7A00H	00H	00H	10H	00H	00H	00H	00H	7A07H
	00H	00H	10H	00H	00H	00H	00H	

- When the condition for the operand is not established (Z=1)

| 7A00H | 00H | 7A07H |
|-------|-----|-----|-----|-----|-----|-----|-----|-------|
| | 00H | |

5-4-4. Search

■ Variable address search

① ENTRY PREPARATION

- Designation of variable name
The variable name is stored in the Ureg.
- Whether array is one dimension or two dimension is stored in parameter F/F (788CH).
 - one dimension: 01H
 - two dimension: 02H
- Subscript is stored in the arithmetic register X (7A00H~7A07H).
 - When one dimension array: the first subscript
 - When two dimension array: the second subscript
- Subscript is stored in the arithmetic register Y (7A10H~7A17H).
 - When one dimension array: No need
 - When two dimension array: the first subscript

② ENTRY ADDRESS

D461H

- Subroutine must be called in the following format:

SJP D461H

DB FAH

DB DATA(Decides the address to return when an error is met.)

③ EXIT STATE

- When no error

The leading address of the variable is stored in the Ureg, and the variable name and the data size is stored in the arithmetic register X. Then, it returns to the address that follows DB DATA.

7A05H	7A06H	7A07H	
UH	UL	10001000	Numeric variable
UH	UL	0XXXXXXX	Nonnumeric variable
Leading address of variable		Size secured for the variable	

- When error

Returns to the address that data plus one is added to the "data written address" after storing the error code is stored in UH.

■ Key scan (I)**① ENTRY PREPARATION**

- None

② ENTRY ADDRESS

E42CH

③ EXIT STATE

- The key code of the key that depressed at that time is stored in the accumulator. If there is no key depression, "00H" will be stored in the accumulator.

■ Key scan (II)**① ENTRY PREPARATION**

- None

② ENTRY ADDRESS

E243H

③ EXIT STATE

- The key code of newly depressed key is stored in the accumulator. Although it does not return until a next key is depressed, it may end in auto-power-off unless a key is depressed within seven minutes. (However, the previous state resumes with depression of the key.)

When the key is pushed, PB7 of the PC-1500 IF register (#F00BH) will be set.

NOTE: So long as PB7 of the address #F00BH is set, "0EH" will be stored in the accumulator.

PB7 of the address #F00BH will be reset when "ANI #F00BH, FDH" is executed.

■ Search of program line**① ENTRY PREPARATION**

- Line number is stored in the Ureg (0001H ≤ Ureg ≤ FFFFH).

② ENTRY ADDRESS

D2EAH

- Subroutine is called in the following format:

SJP D2EAH

DB DATA

③ EXIT STATE

- When the specified line is found

Returns to the address that follows DB DATA after storing the data in SEARCH (78A6H~78ABH).

78A6H	Leading address of the line
78A7H	Leading address of the line
78A8H	Line number
78A9H	Line number
78AAH	Leading address of the line plus 3
78ABH	Leading address of the line plus 3

- When the specified line is not found

Returns to the "data written address" plus one after storing the error code in UH.

Carry	Condition
0	No specified line found, and search is made to the last of the program.
1	Found the line larger than the specified line.

5-4-5. Display

■ One character display**① ENTRY PREPARATION**

- Display start position is stored in the cursor pointer (7875H). Cursor will be effective within a range of 00H to 98H.
- Code of display character is stored in the accumulator.

② ENTRY ADDRESS

ED57H

① EXIT STATE

- No change takes place in the cursor pointer.
- Change will be met in carry depending on the cursor called.

Cursor	Carry
00H~95H	0
96H~9BH	1

- One character will be displayed on LCD.

■ Moving cursor after displaying one character**① ENTRY PREPARATION**

- Display start position is stored in the cursor pointer (7875H). Cursor will be effective within a range of 00H to 98H.
- Code of display character is stored in the accumulator.

② ENTRY ADDRESS

ED4DH

③ Change will be met in the cursor pointer.

Cursor position when called	Cursor position after return
00H~95H	Previous cursor position +6
96H~9BH	00H

- One character will be displayed on LCD.

■ Auto-power-off**① ENTRY PREPARATION**

- None

② ENTRY ADDRESS

E33FH

③ EXIT STATE

- When power is turned on once after auto-power-off, no printer initialization takes place.
- When power is turned on by means of the [ON] key once after auto-power-off, it needs to push any key, except [SHIFT], [SML], and [DEF] key, as subroutine is in execution.

Note: An example of manual operation

Example of manual operation

Key operation	Display
CALL & E33F	CALL & E33F
[ENTER]	(OFF state)
[ON]	BUSY CALL & E33F
Any key except [SHIFT], [SML], and [DEF]	>

■ "n" character display**① ENTRY PREPARATION**

- Display start position is stored in the cursor pointer (7875H).
- Size of the character string is stored in the accumulator (01H ≤ ACC ≤ 1AH).
- Leading address of the character string is stored in the Ureg (0000H ≤ Ureg ≤ FFFFH).

② ENTRY ADDRESS

ED00H

③ EXIT STATE

- Change will be met in carry.

Carry	Cursor position
0	Next to the rightmost position of the character string on display.
1	Indicates the last character on the display, in case display should end at 26th digit or exceed 26th digit.

NOTE: When display exceeds 156 dots, the contents after this dot position will be ignored.

■ Output of "n" characters from the top of LCD

"n" characters will be displayed unconditionally from the left side of LCD.

① ENTRY PREPARATION

- The leading address of the character string is stored in the Ureg (0000H ≤ Ureg ≤ FFFFH).
- Size of the character string is stored in XL (01H ≤ Xreg ≤ 1AH).

② ENTRY ADDRESS

ED3BH

③ EXIT STATE

- Change will be met in carry.

Carry	Contents
0	Character string within 25 characters.
1	Character string more than 26 characters.

■ Hexadecimal (2 byte → 1 byte)

ASCII code of two bytes is assumed to be a hexadecimal figure and is changed into numeric of one byte.

① ENTRY PREPARATION

- Leading address of the ASCII code is stored in the Xreg.

② ENTRY ADDRESS

ED95H

③ EXIT STATE

- In Xreg is stored the value of previous Xreg added with 02H.
- The one byte of the derived data is stored in the accumulator.

■ Graphic display**① ENTRY PREPARATION**

- Output pattern is stored in the accumulator.

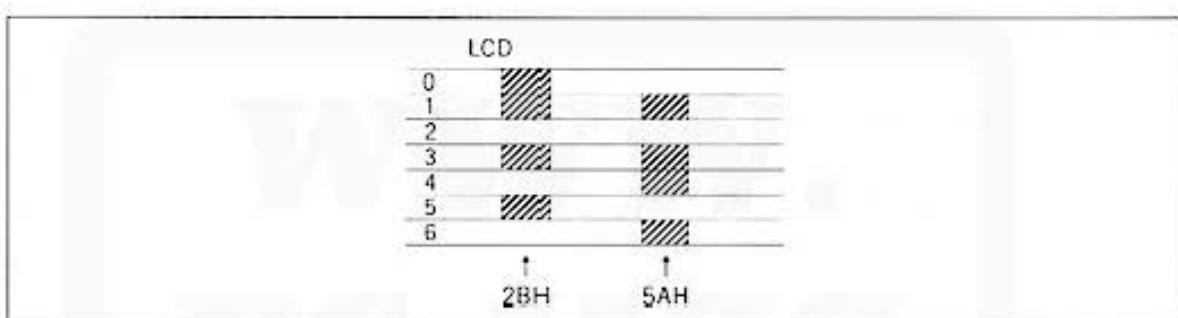
② ENTRY ADDRESS

EDEFH

③ EXIT STATE

- Contents of Xreg, Ureg, and accumulator become irrelevant.
- No change takes place in the cursor position.

(Reference)

**■ Program display****① ENTRY PREPARATION**

- A. In the case of numeric

Data is stored in the arithmetic register (7A00H~7A07H).

- B. In the case of character string or program

Data is stored in the input buffer (7BB0H~7BFFH).

Cursor position is stored in Yreg. (In the case of program, 7BH is stored in YH.)

- C. Parameter F/F (7880H) is set.

Data of 7880H	Display contents
40H	Character string is displayed according to Yreg.
00H	Character string is displayed from the top, regardless of Yreg contents.
20H	Numerical value of the arithmetic register X is displayed.
10H	Line number, space, and program are displayed from the top.
14H	Line number, colon, and program are displayed from the top.
50H	Line number and space are displayed in a middle of program according to Yreg.
54H	Line number and colon are displayed in a middle of program according to Yreg.

② ENTRY ADDRESS

E8CAH

③ EXIT STATE

- Displayed on LCD according to direction.

NOTES: 1. In the case of an internal code, the cursor must show the code position on the lower side of 2 bytes.
 2. Numeric is displayed in right justified manner and character string in left justified manner.
 3. Numeric has no concern with USING.
 4. Only 26 characters will be handled for a character string exceeding 26 characters.
 5. ODH is required at the end of character string for the input buffer.

5-4-6. Printer

■ Color designation.**① ENTRY PREPARATION**

- The specified color code (0~3) is stored in the UL.

② ENTRY ADDRESS

A519H (A4F7H for the version 0.)

③ EXIT STATE

- Return with the motor on.

(Motor off routine must be called after EXIT.)

(EX)	LDI	UL, color code
	SJP	A519H
	SJP	A769H
	RTN	

■ Motor drive**① ENTRY PREPARATION**

- Set the address pointer in the Xreg to indicate the value of relative movement, then store the value of relative movement after that address pointer.

Address shown by Xreg	ΔXH : High order 8 bits of the relative movement value towards the X direction.
Address shown by Xreg+1	ΔYH : High order 8 bits of the relative movement value towards the Y direction.
Address shown by Xreg+2	ΔXL : Low order 8 bits of the relative movement value towards the X direction.
Address shown by Xreg+2	ΔYL : Low order 8 bits of the relative movement value towards the Y direction.

② ENTRY ADDRESS

A8DDH (A8B7H for the version 0.)

③ EXIT STATE

- The motor is on. (Motor off routine must be called after EXIT.)

NOTE: Negative movement must be indicated by a complement.
Relative movement must be within a range of -32768 thru 32767.

■ Motor off

① ENTRY PREPARATION

- None

② ENTRY ADDRESS

A769H (A747H for the version 0.)

③ EXIT STATE

- Motor off

■ Pen up/down

① ENTRY PREPARATION

- Either 00H or FFH is stored in the PEN UP/DOWN F/F (79E9H).

UP	00H
DOWN	FFH

② ENTRY ADDRESS

AAE3H (AABDH for the version 0.)

③ EXIT STATE

- Pen ascends or descends then solenoid turn inactive.

■ GRAPHIC mode preparation

① ENTRY PREPARATION

- None

② ENTRY ADDRESS

ABEFH (ABC6H for the version 0.)

③ EXIT STATE

Line (79EAH)	00H
Rotate (79F2H)	00H
User counter (79E0H~79E3H)	00H

NOTES:

1. Because G/T (79F0H) is not changed, it needs to store FFH in G/T (79F0H) after return.
2. CSIZE does not change.
3. Scissoring counter does not change.

■ TEXT mode preparation**① ENTRY PREPARATION**

- None

② ENTRY ADDRESS

ACBBH (ACBFH for the version 0.)

③ EXIT STATE

Line (79EAH)	00H
G/T (79F0H)	00H
Rotate (79F2H)	00H

NOTES:

1. When the scissoring counter YH and YL (79E4H~79E5H) exceeds 0200H, it makes 79E4H turned to 01H and 79E5H to FFH. When (79E4H~79E5H) is below 01FFH, it causes no change in 79E4H and 79E5H.
2. CSIZE does not change.

■ Paper feed**① ENTRY PREPARATION**

- Paper feed count is stored in the address represented by the Xreg. (Negative number is indicated by a complement.)

② ENTRY ADDRESS

AA04H (A9DEH for the version 0.)

③ EXIT STATE

- The motor stays on. (Motor off routine must be called after EXIT.)

NOTE: Paper feed count may change after setting the CSIZE.

■ Linefeed**① ENTRY PREPARATION**

- Line kind (79EAH) must be reset to 0.
- Address area that can be destructed should be stored in Xreg. (X-10 ~ X+1 will be destructed.)

② ENTRY ADDRESS

A9F1H (A9CBH for the version 0.)

③ EXIT STATE

- The motor stays on. (Motor off routine must be called after EXIT.)

NOTES:

1. When the contents of Xreg is 7A22H, data in 7A18H thru 7A23H will be destructed.
2. Paper feed count changes after CSIZE.

■ Print**① ENTRY PREPARATION**

- 00H is stored in LINE TYPE (79EAH).
- Print code storing address must be stored in the Xreg.
(When 7A20H is stored in Xreg, it affects the data in 7A08H thru 7A37H. So, care must be exerted not to destruct address area of the Xreg represented address minus ten and plus one.)
- Print code is stored in the address represented by the Xreg.

② ENTRY ADDRESS

A781H (A75BH for the version 0.)

③ EXIT STATE

- Stays on.
- The value of 6×CSIZE will be added to the contents of CURSOR (79E6H). (Motor off routine must be called after EXIT.)

5-4-7. Cassette tape

■ REMOTE ON**① ENTRY PREPARATION**

- Sets the PARAMETER F/F (7879H).

Contents of 7879H	Remote	CMT input port
0XX0XXXX	0	Close
0XX1XXXX	1	(CMT output)
1XX0XXXX	0	Open
1XX1XXXX	1	(CMT input)

② ENTRY ADDRESS

BF11H

③ EXIT STATE

- REMOTE 0 or 1 turns active according to the contents of the PARAMETER F/F (7879H).

NOTE: This system subroutine drives the relay in the CE-150, regardless of the REMOTE switch position.

■ REMOTE OFF**① ENTRY PREPARATION**

- RMT/BEEP (786BH) must be set to control REMOTE 1.
(Preparation is not required in the case of REMOTE 0.)

Contents of 786BH	Remote
0XXXXXXXX	OFF
1XXXXXXXX	ON

② ENTRY ADDRESS

BF43H

③ EXIT STATE

- REMOTE 0 is off.
- REMOTE 1 will be in accordance with RMT/BEEP (786BH).

NOTE: This system subroutine drives the relay in the CE-150, regardless of the REMOTE switch position.

■ Save of one character**① ENTRY PREPARATION**

- Data is stored in the accumulator.

② ENTRY ADDRESS

BDCCH

③ EXIT STATE

- None.

NOTE: This system call must be executed after saving of the synchronizing header.

■ Load of one character**① ENTRY PREPARATION**

- None

② ENTRY ADDRESS

BDF0H

③ EXIT STATE

- Data has been sent in the accumulator.
- Change takes place in carry.

Carry	Condition
0	End of data read.
1	Depression of the [BREAK] key.

■ Creation of header**① ENTRY PREPARATION**

- File mode is set in the accumulator.

File mode	Contents
00H	Machine language
01H	BASIC program
02H	Reserve
04H	Data

Use another code to avoid confusion, as the file mode for other than PC-1500 is used.

- File name is stored in 7B69H~7B78H of the output buffer.

② ENTRY ADDRESS

BB6D

③ EXIT STATE

- Header is established in 7B60H~7B87H of the output buffer.

Address	Contents
7B60H~7B67H	Synchronizing header
7B68H	File mode
7B69H~7B78H	File name
7B79H~7B87H	All 00H

■ Header input/output**① ENTRY PREPARATION**

- Parameter F/F (7879H) is set.

Contents of 7879H	In/Out	Remote
0XX0XXXX	Out	0 side
0XX1XXXX		1 side
1XX0XXXX	In	0 side
1XX1XXXX		1 side

- RMT/BEEP (786BH) is set.

Contents of 786BH	BEEP
XXXXXXX0	OFF
XXXXXXX1	ON

- Header is created in the case of output.

② ENTRY ADDRESS

BCE8H

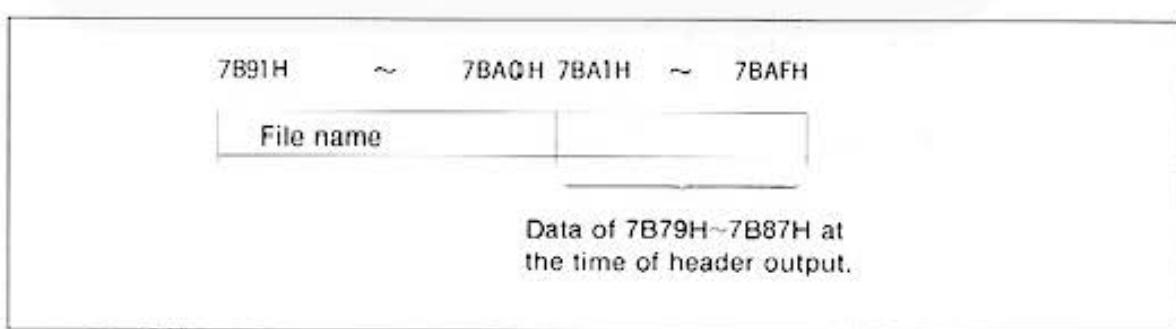
③ EXIT STATE

- Carry changes.

Carry	Condition
0	Broken in a middle.
1	Input is complete.

- In the case of input

The read data is stored in the output buffer of 7B91H~7BAFH.



- In the case of output

Stop bit modulation signal stays on.

- NOTES:
1. BEEP will be in accordance with 786BH.
 2. Paper feed operation stays prohibited for the CE-150.
 3. In the case of input, the file name coincident of the header and file mode is displayed on the LCD. In case the file name is specified, it is searched until found.

■ File transfer

① ENTRY PREPARATION

- 1) Parameter F/F (7879H) is set.

7879H	Contents
00XXXXXX	In the case of load
01XXXXXX	In the case of verification
0XXXXXXX	In the case of save

- 2) Leading address of input/output data is stored in the Xreg.

- 3) Byte count of the data minus one is stored in the Ureg.

② ENTRY ADDRESS

BD3CH

③ EXIT STATE

- In the case of save
(Carry changes)

Carry	Condition
1	Broken in a middle.
0	Input/output is complete.

■ Termination of CMT I/O control

① ENTRY PREPARATION

- Parameter F/F (7879H) is set.

Contents of 7879H	Data input/output
1XXXXXXX	Input
0XXXXXXX	Output

② ENTRY ADDRESS

BBF5H

③ EXIT STATE

- Serial port is reset.
- PAPER FEED key of the CE-150 becomes operative.
- Motor drive turns off.
- In the case of load

Change is met in carry, H, and V.

Carry	H	V	State
0	-	-	Load and verification are complete.
1	1	-	In break state.
1	0	1	Error occurrence during verification.
1	0	0	Occurrence of check sum error.

NOTE: The last transfer "address plus one" is stored in the Xreg.

5-4-8. Caution for system subroutine call

1. Printer related system call

Although the error code is put on the display when an error such as low battery occurred during execution of printer related system subroutine, it is not possible to check the error line using the [↑] key.



6

Machine language programming examples

*Machine language program discussed is assumed to start from the address 40C5H.

6-1. Binary to hexadecimal conversion

The binary number stored in the Xreg is converted into hexadecimal equivalent and stored in the fixed nonnumeric variable YS. Binary number within a range of $-32769 < a < 32768$ is applicable.

ADDRESS	MACHINE LANGUAGE		MNEMONIC
40C5	68 77		LDI UH 77H
7	6A E0		LDI UL E0H
9	84		LDA XH
A	BE 40 E0		SJP ①
D	61		SIN U
E	84		LDA XH
F	BE 40 E1		SJP ②
D2	61		SIN U
3	04		LDA XL
4	BE 40 E0		SJP ①
7	61		SIN U
8	04		LDA XL
9	BE 40 E1		SJP ②
C	61		SIN U
D	69 00		ANI U, 00H
F	9A		RTN
E0	F1	①	AEX
1	B9 0F	②	ANI A, 0FH
3	B7 0A		CPI A, 0AH
5	83 03		BCS ③
7	B3 30		ADI A, 30H
9	9A		RTN
A	B3 36	③	ADI A, 30H
C	9A		RTN

6-2. Display inversion

The current display contents are inverted.

ADDRESS	MACHINE LANGUAGE	MNEMONIC
40C5	68 78	LDI UH, 78H
7	6A 4D	LDI UL, 4DH
9	FD 62	DEC UH
B	25	LDA U
C	BD FF	EAI FFH
E	2E	STA U
F	88 06	LOP 06H
D1	6C 77	CPI UH, 77H
3	93 0E	BCS. - 0EH
5	9A	RTN

6-3. Single display dot left shift

The current display contents are shifted to the left by one dot position.

ADDRESS	MACHINE LANGUAGE		MNEMONIC
40C5	FD 88		PSH X
7	FD 98		PSH Y
9	FD A8		PSH U
B	A5 76 00		LDA 7600H
E	F1		AEX
F	B9 OF		ANI A, OFH
D1	0A		STA XL
2	A5 76 01		LDA 7601H
5	F1		AEX
6	B9 OF		ANI A, OFH
8	08		STA XH
9	68 78		LDI UH, 78H
B	FD 62	①	DEC UH
D	6A 4D		LDI UL, 4DH
F	66	②	DEC U
E0	65		LIN U
1	1A		STA YL
2	25		LDA U
3	18		STA YH
4	84		LDA XH
5	63		SDE U
6	04		LDA XL
7	2E		STA U
8	FD 18		LDX Y
A	88 0D		LOP ②
C	6C 77		CPI UH, 77H
E	93 15		BCS ①
F0	04		LDA XL
1	F1		AEX
2	AE 77 4E		STA 774EH
5	84		LDA XH
6	F1		AEX
7	AE 77 4F		STA 774FH
A	FD 2A		POP U
C	FD 1A		POP Y
E	FD 0A		POP X
4100	F9		REC
1	9A		RTN

6-4. Single display dot right shift

The current display contents are shifted to the right by one dot position.

ADDRESS	MACHINE LANGUAGE	MNEMONIC
40C5	FD 88	PSH X
7	FD 98	PSH Y
9	FD A8	PSH U
B	A5 77 4C	LDA 774CH
E	F1	AEX
F	B9 F0	ANI A, FOH
D1	0A	STA XL
2	A5 77 4D	LDA 774DH
5	F1	AEX
6	B9 F0	ANI A FOH
8	08	STA XH
9	68 75	LDI UH, 75H
B	6A FF	LDI UL, FFH
D	64	INC U
E	65	LIN U
F	1A	STA YL
E0	25	LDA U
1	18	STA YH
2	84	LDA XH
3	63	SDE U
4	04	LDA XL
5	61	SIN U
6	FD 18	LDX Y
8	6E 4D	CPI UL, 4DH
A	91 0F	BCR,-0FH
C	6C 77	CPI UH, 77H
E	91 15	BCR,-15H
F0	64	INC U
1	04	LDA XL
2	F1	AEX
3	61	SIN U
4	84	LDA XH
5	F1	AEX
6	2E	STA U
7	FD 2A	POP U
9	FD 1A	POP Y
B	FD 0A	POP X
D	F9	REC
E	9A	RTN

6-5. Conversion of USING format expressed numerical data into character string

① ENTRY PREPARATION

- Numeric data is stored in the fixed numeric variable A in decimal figure and the format is specified by means of the USING statement.
(Format is within 16 characters and no error is detected during conversion.)

② ENTRY ADDRESS 40C5H

③ EXIT STATE

- Character string is stored in the fixed nonnumeric variable AS. However, AS can be anything when in error.

④ PROGRAM

ADDRESS	MACHINE LANGUAGE	MNEMONIC
40C5	48 79	LDI XH, 79H
7	4A 00	LDI, 00H
9	58 7A	LDI YH, 7AH
B	BE F7 3F	SJP F73FH
E	B5 01	LDI A, 01H
D0	CD 96	VMJ 96H
2	DF	DEC A
3	2A	STA UL
4	58 78	LDI YH
6	5A C0	LDI YL
8	F5	TIN
9	88 03	LOP 03H
B	14	LDA YL
C	B9 0F	ANI A, 0FH
E	8B 02	BZS.+02H
E0	59 00	ANI (Y), 00H
2	9A	RTN

6-6. Power off that does not activate the printer upon power on

With the following program, the printer will not be activated when power is turned on after power was turned off, the same manner as in the case of **OFF** to **ON**.

ADDRESS	MACHINE LANGUAGE	MNEMONIC
40C5	AA 78 4F	LDI S, 784FH
8	BE CF CC	SJP CFCCH
B	BE D0 2B	SJP D02BH
E	B5 3E	LDI A, 3EH
D0	1E	STA Y
1	E9 78 8A EF	ANI 788AH, EFH
5	E9 76 4E FE	ANI 764EH, FEH
9	B5 00	LDI A, 00H
B	AE 78 80	STA 7880H
E	AE 78 9C	STA 789CH
E1	AE 78 9D	STA 789DH
4	BE E8 CA	SJP E8CAH
7	48 CA	LDI XH, CAH
9	4A 92	LDI XL, 92H
B	FD 88	PSH X
D	BA E3 3F	JMP E33FH

REFERENCE

1. Determining printing character size and direction

- Specifying printing character size
Print character size (1~9) must be stored in the character size memory (79F4H).
- Specifying printing direction
Print direction (0~3) must be stored in the print direction memory (79F2H).

2. Restoration of array and two-character variable

- Array variable and two-character variable that cleared by means of RUN operation or CLEAR command can be restored by operating the variable pointer (7899H, 789AH). Number of bytes dominated by the array variable and two-character variable should be deducted from the last address of the free area, then store it in the variable pointer.
- How to store
Assume now "x" is the number that the byte count of array and two-character variable deducted from the last address of the free area.

$$a = x / 256$$

$$b = x - 256 * a$$

Where,

a: High order two digits when x is displayed in hexadecimal figure.
b: Low order two digits when x is displayed in hexadecimal figure.
Whereas, store a in the variable pointer 7899H and b in 789AH.
- Last address of the free area

System configuration	Address
PC-1500 only	4800H
PC-1500 + CE-151	5800H
PC-1500 + CE-155	6000H
PC-1500 + CE-159	4800H

- Number of bytes used for the array and two-character variable

Numeric array variable:

7 bytes + 8 bytes * size of array

Nonnumeric array variable:

7 bytes + character length * size of array

(Character size is normally 16 characters.)

Numeric two-character variable:

7 bytes + 8 bytes

Character two-character variable:

7 bytes + 16 bytes

3. Knowing the use of CE-150

Because the system program ROM of the CE-150 resides in the CE-150, FFH will be read when a ME0 address range of A000H thru BFFFH is accessed with the CE-150 not connected to the PC-1500. If connected, the contents of A000H will fetch C0H.

NOTE: PV must be reset.

www.
PC-1500
.INFO



5. Circuit diagram

Circuitry subject to change without notice.

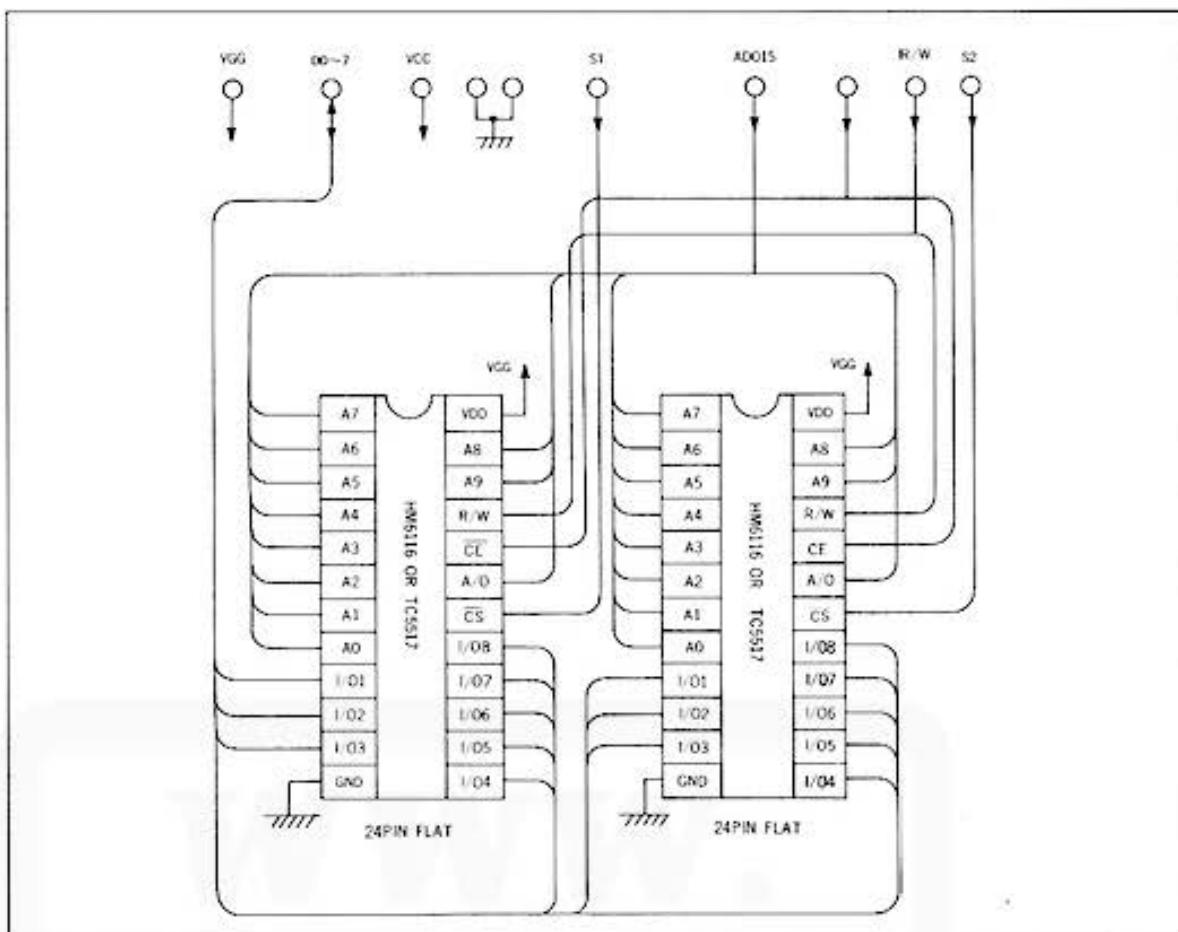




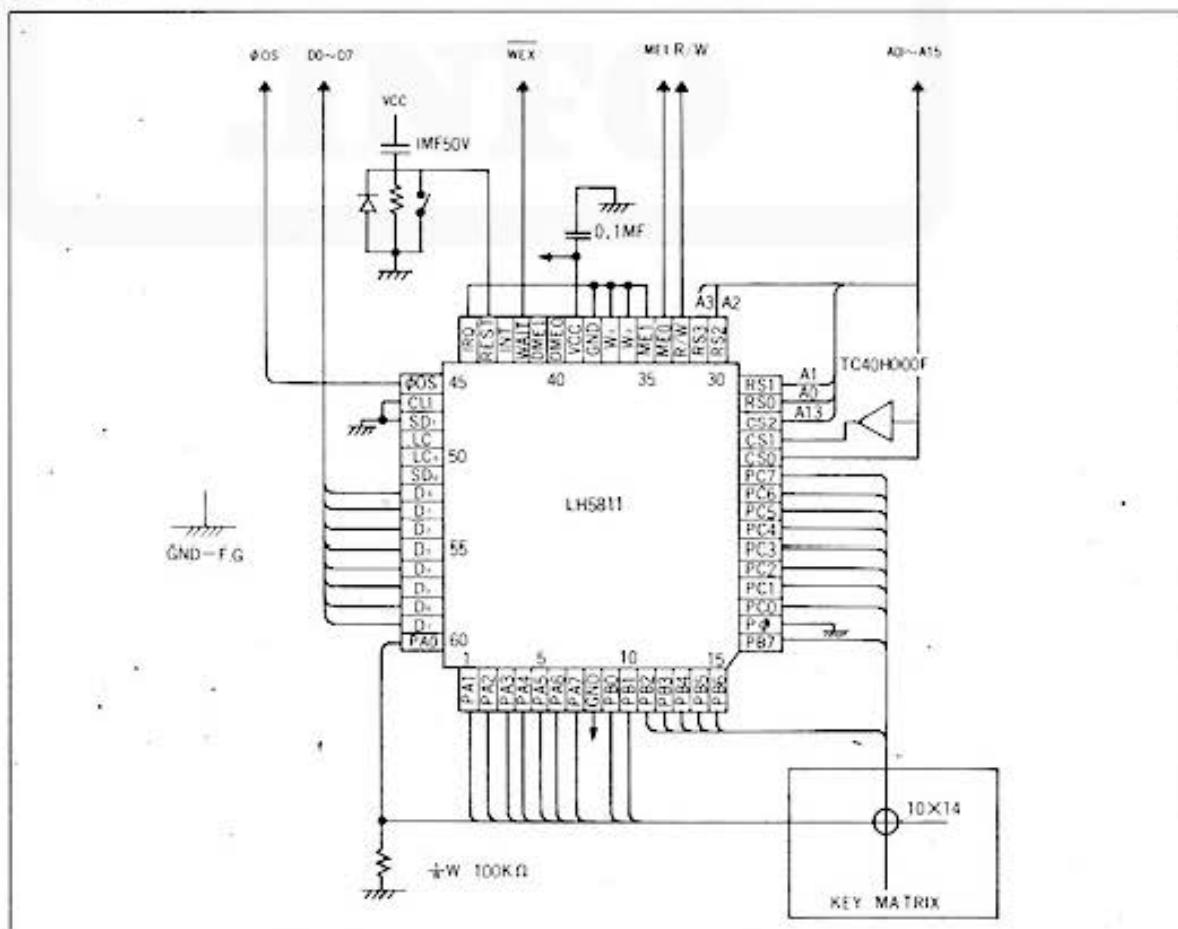


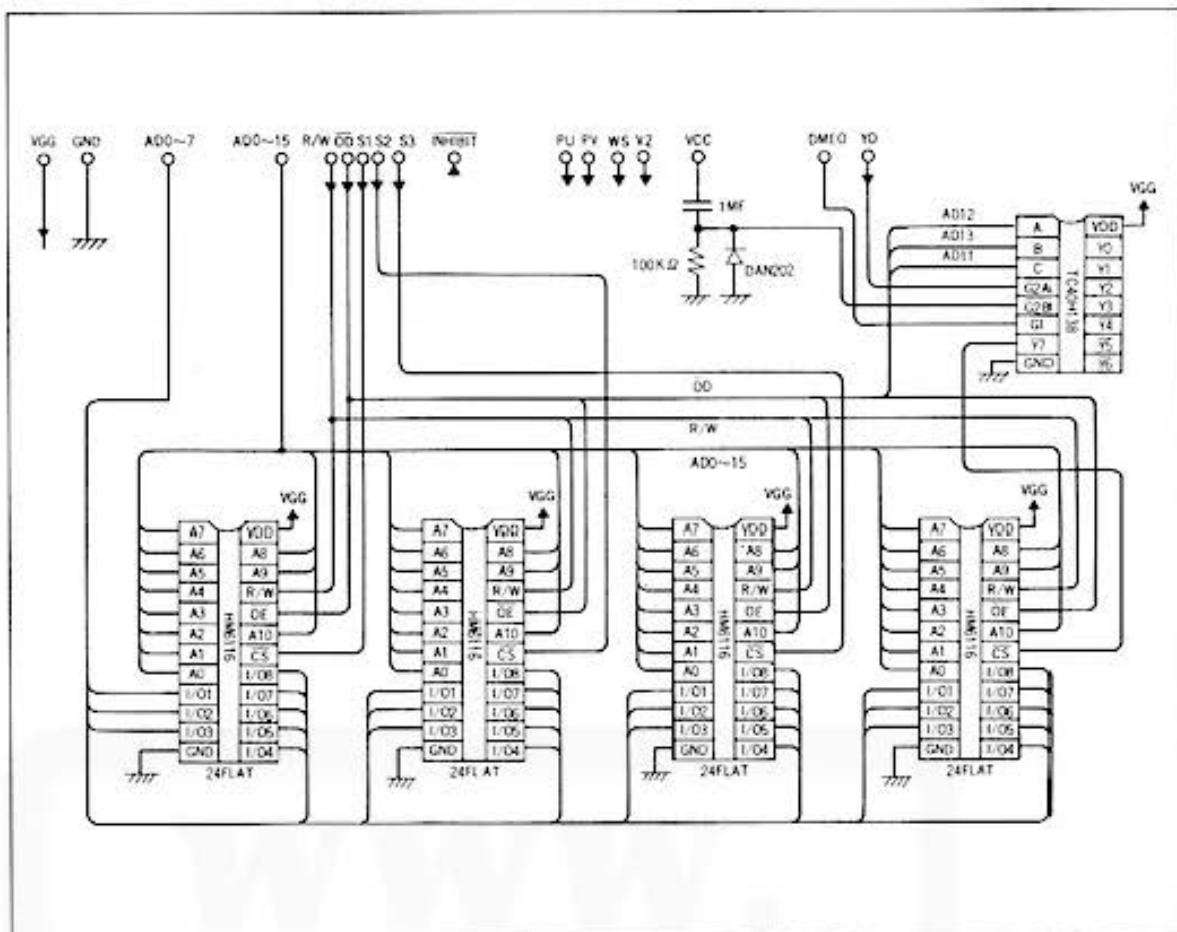


5 - 3
CE-151



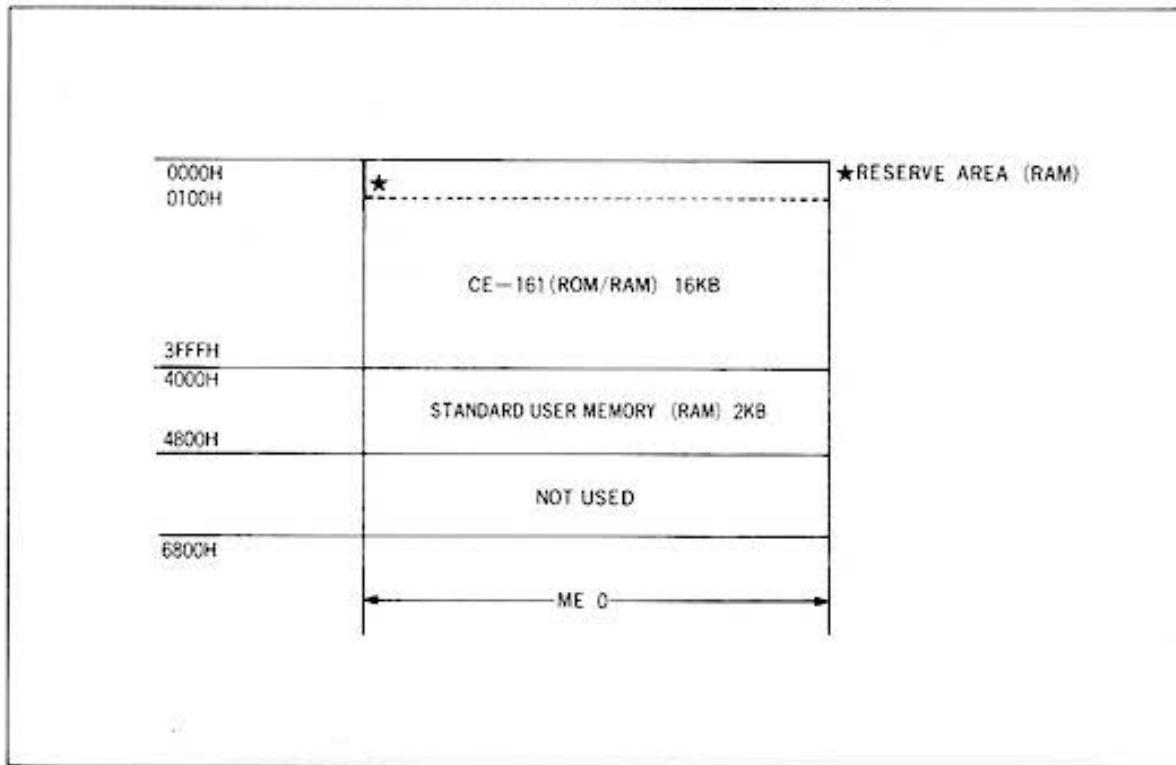
5 - 4
CE-153



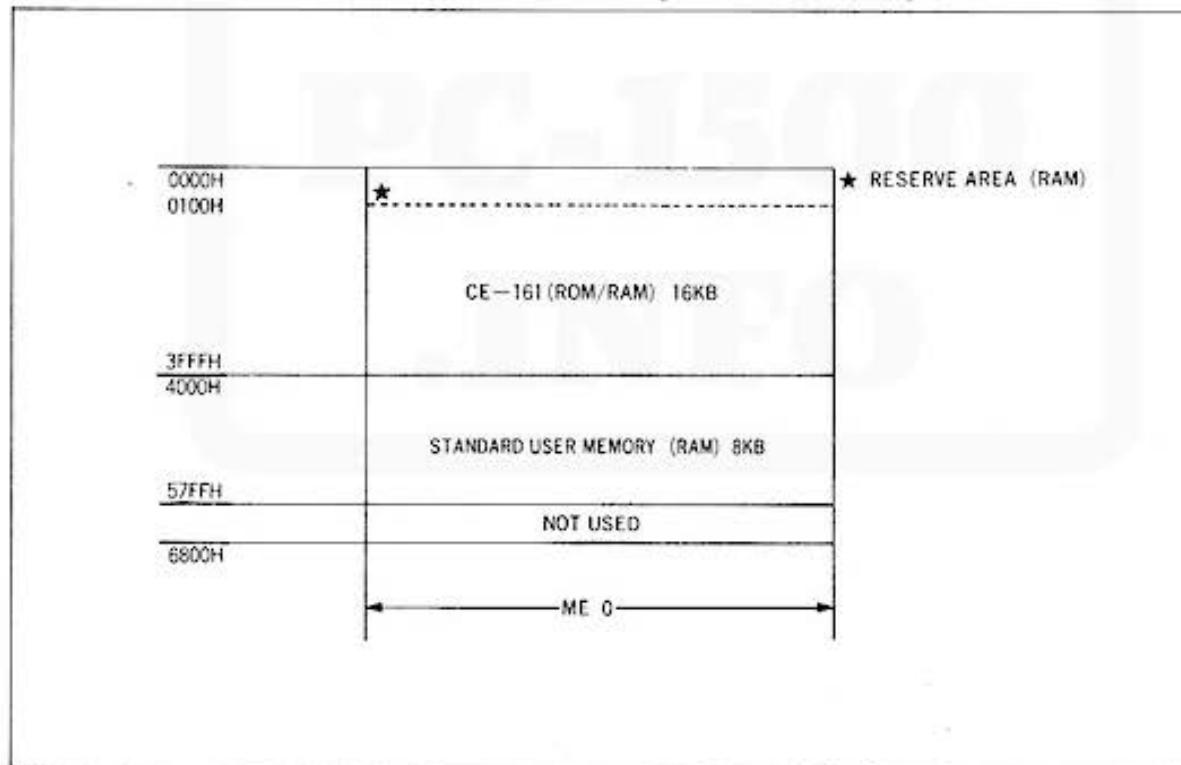




MEMORY MAP when the CE-161 is used. (FOR PC-1500)

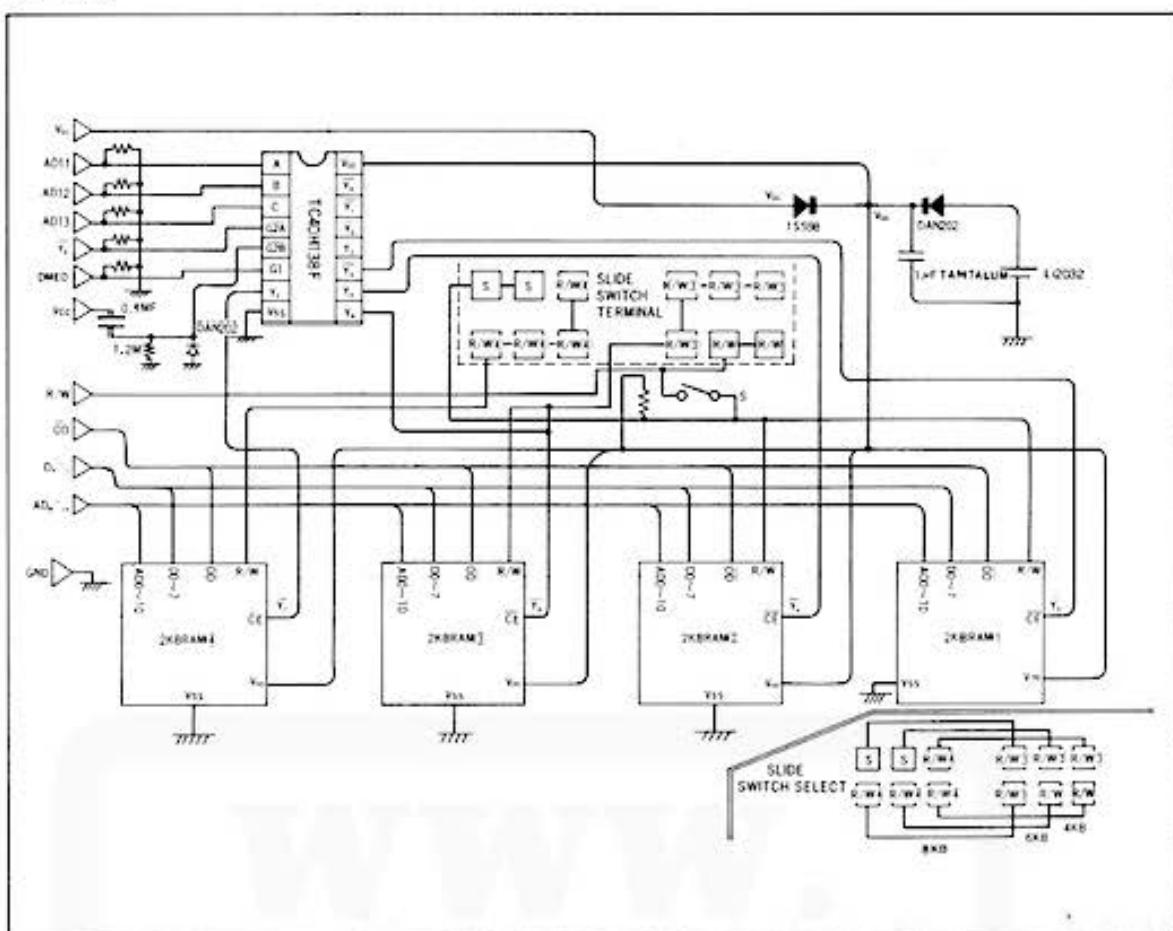


MEMORY MAP when the CE-161 is used. (FOR PC-1500A)



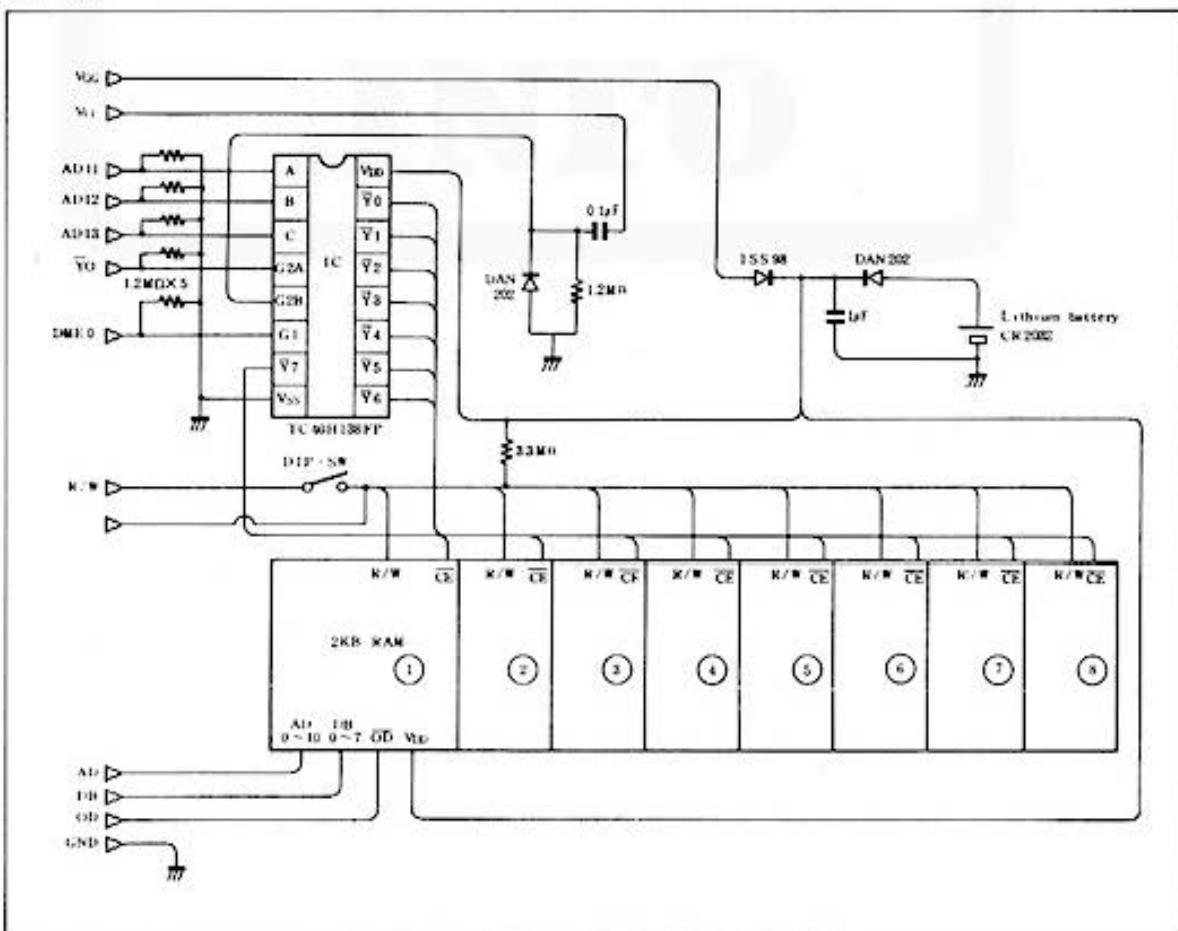
5 - 7

CE-159

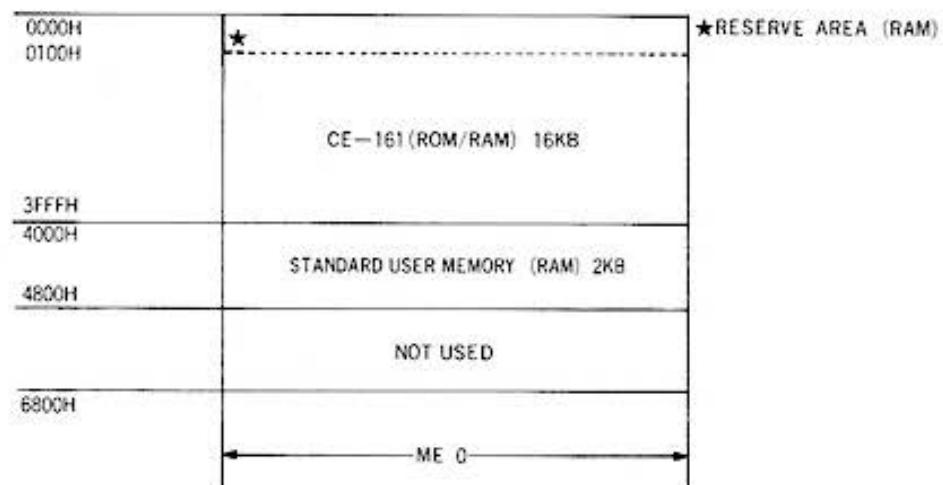


5 - 8

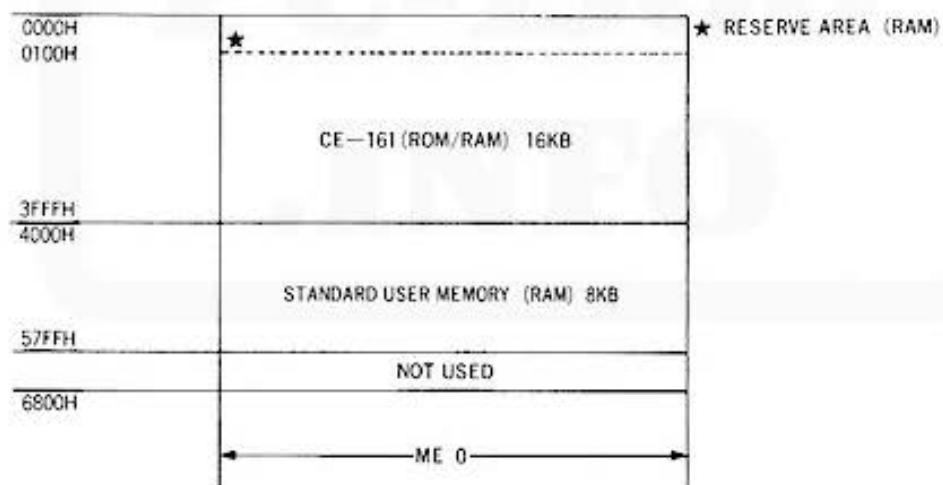
CE-161



MEMORY MAP when the CE-161 is used. (FOR PC-1500)



MEMORY MAP when the CE-161 is used. (FOR PC-1500A)



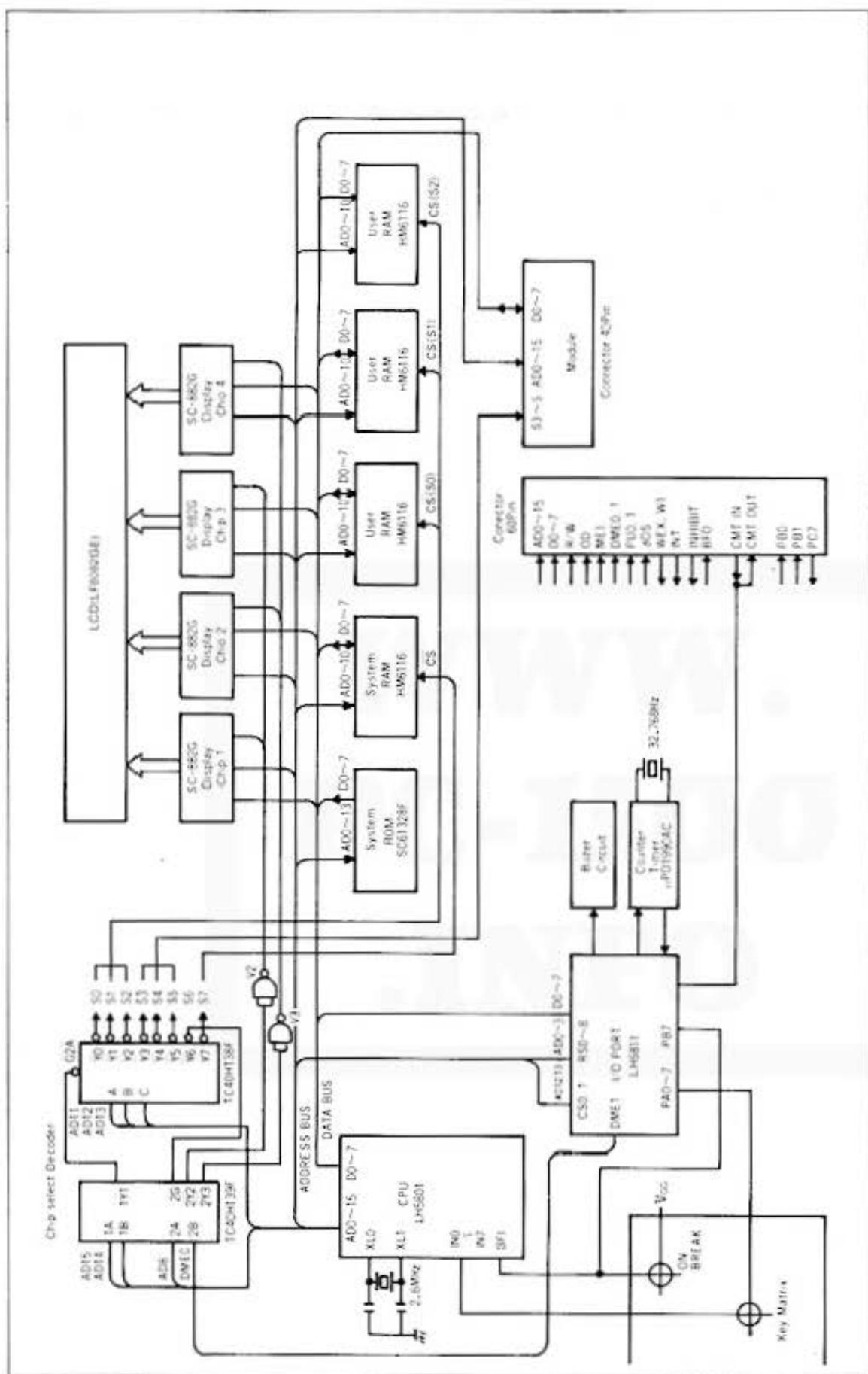
ADDENDUM

Differences between the PC-1500A and the PC-1500

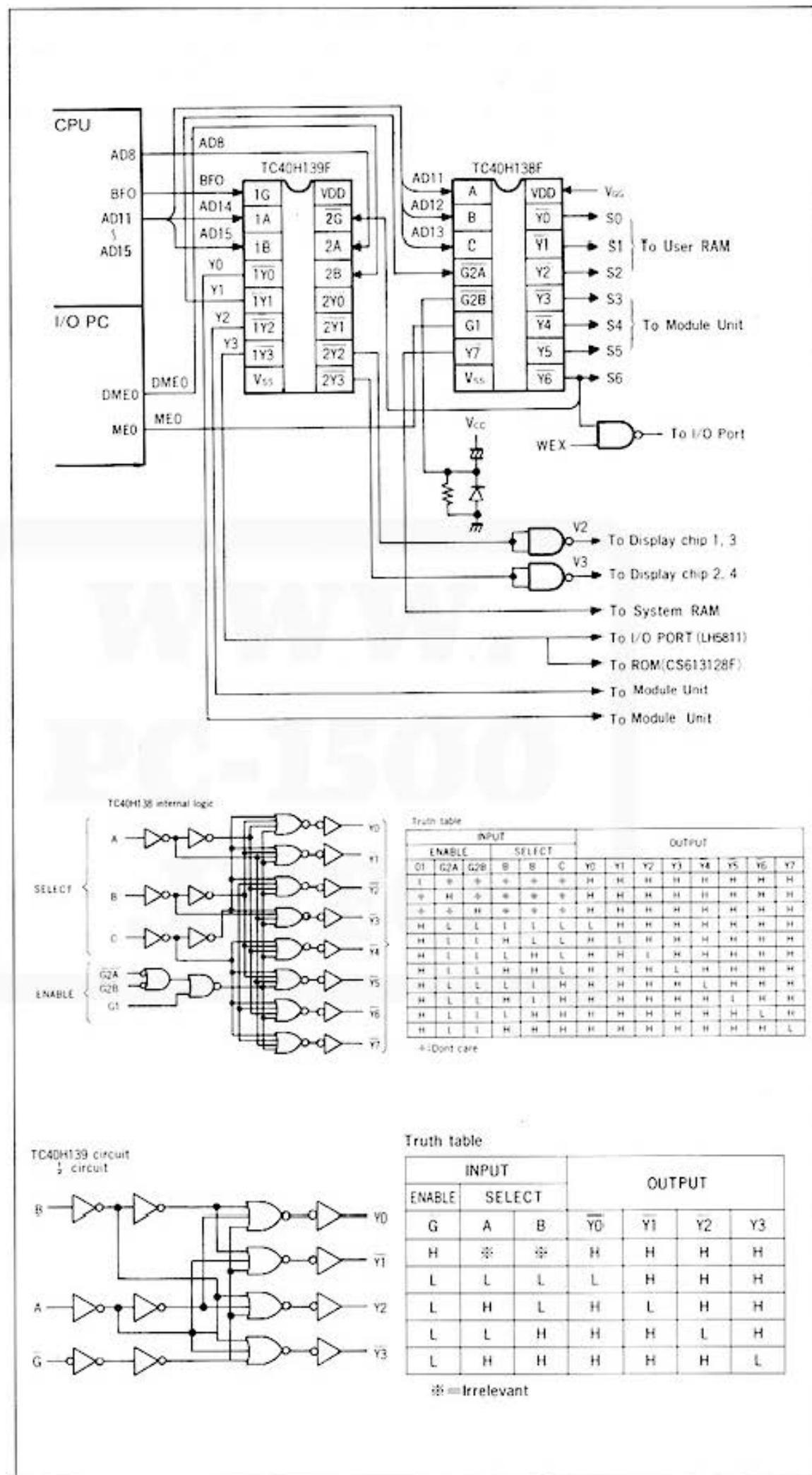
Take notice of these differences when you use the machine language for the PC-1500A.

Page	PC-1500	PC-1500A
89	<p>④ Chip select decoder ...and it is used to select chip by means of S0~4, S6, S7, $\overline{Y2}$ and $\overline{Y3}$. ...</p> <p>⑦ System RAM Because the 4-bit \times 1K bytes TC5514 is used in a pair, the data bus is divided into two of D0~D3 and D4~D7 and the select signal S7 is commonly shared so as to be compatible with the 8-bit RAM. Address is within 7800H to 7BFFFH of the ME0 area which is used for the system memory area and for the fixed variable area.</p>	<p>④ Chip select decoder ...and it is used to select chip by means of S0~7, $\overline{Y2}$ and $\overline{Y3}$. ...</p> <p>⑦ System RAM Address is within 7800H to 7FFFFH of the ME0 area which is used for the system memory area, machine language area (7C01H~7FFFH) and for the fixed variable area.</p>
90	<p>⑧ User RAM It is the user RAM for which 8-bit \times 2KB HM6116 is used. Address selected by S0 is within 4000H to 47FFH.</p> <p>4-2-2. Block diagram</p>	<p>⑧ User RAM It is the user RAM for which 8-bit \times 2KB HM6116 is used. Address selected by S0~S2 is within 4000H to 57FFH.</p> <p>4-2-2. Block diagram for the PC-1500A Refer to 4-2-2 diagram for the PC-1500A.</p>
91	4-2-3. Chip select circuit	<p>4-2-3. Chip select circuit for the PC-1500A Refer to 4-2-3 circuit for the PC-1500A.</p>

4-2-2. Block diagram for the PC-1500A



4-2-3. Chip select circuit for the PC-1500A



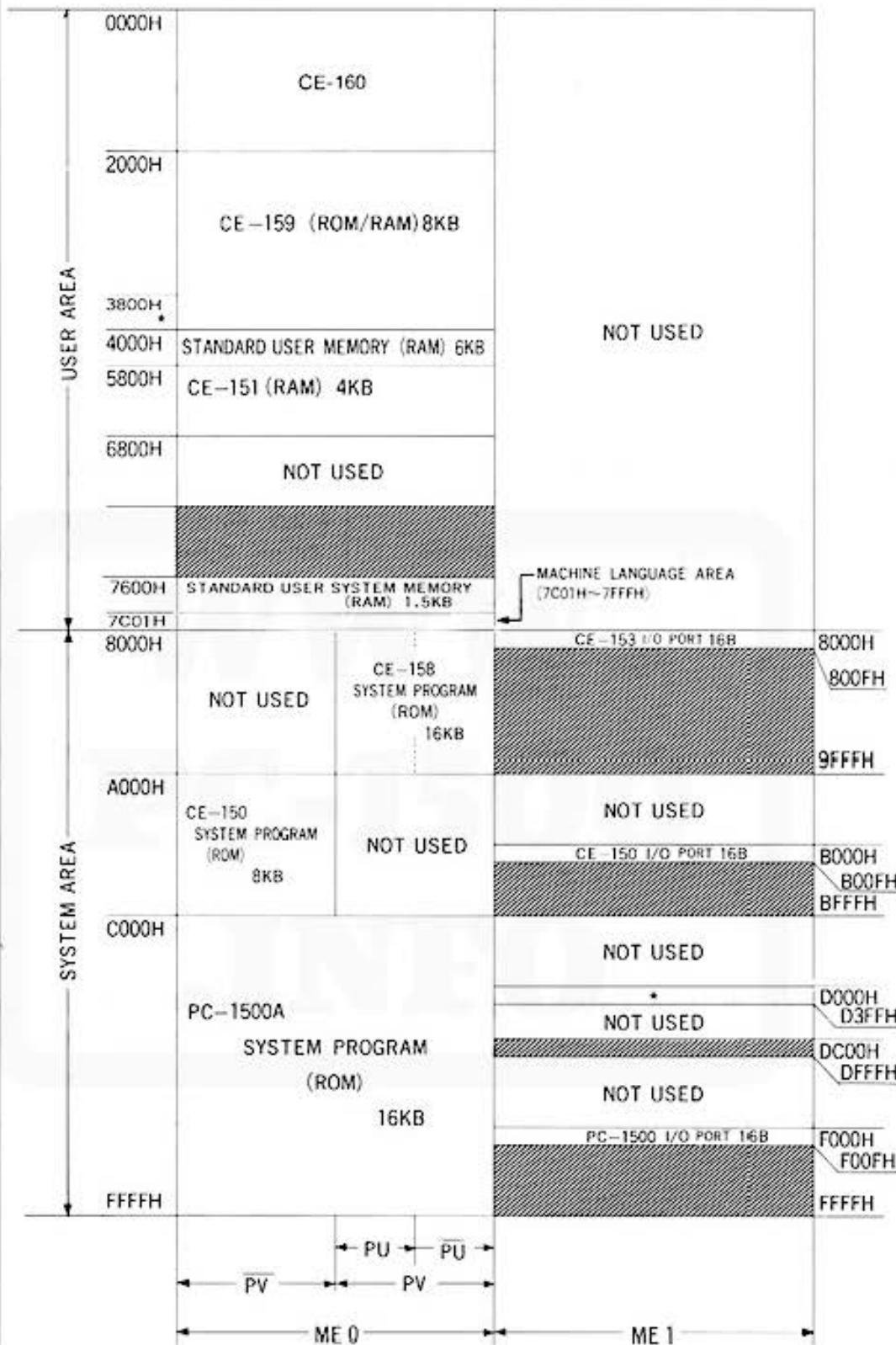
Page	PC-1500	PC-1500A
92	<ul style="list-style-type: none"> • S0~S7 is selected by the decoder IC (TC40H138F) when the gate signal input ME0 (G1) is high, Y1 ($\overline{G2A}$) low, and G2B is low (normally low). <p>S1 With high ..., low so as to ($\overline{Y1}$) select the optional user RAM area. ...</p> <p>S2 With low ..., the S2 output ($\overline{Y2}$) goes low so as to select the optional user RAM area. ...</p> <p>S3 With low ..., the S3 output ($\overline{Y3}$) goes low so as to select the user RAM area. ...</p> <p>S4 With high ..., the S4 output ($\overline{Y4}$) goes low so as to select the user RAM area. ...</p> <p>S5 Do not use</p> <p>S7 so as to select the system RAM (TC5514) ...</p>	<ul style="list-style-type: none"> • S0~S7 is selected by the decoder IC (TC40H138F) when the gate signal input ME0 (G1) is high, Y1 ($\overline{G2A}$) low, and G2B is low (normally low). <p>S1 With high ..., low so as to ($\overline{Y1}$) select the user RAM area. ...</p> <p>S2 With low ..., the S2 output ($\overline{Y2}$) goes low so as to select the user RAM area. ...</p> <p>S3 With low ..., the S3 output ($\overline{Y3}$) goes low so as to select the optional user RAM area. ...</p> <p>S4 With high ..., the S4 output ($\overline{Y4}$) goes low so as to select the optional user RAM area. ...</p> <p>S5 Optional user RAM area ($\overline{Y5}$) (Address assignment of 6800H~6FFFH)</p> <p>S7 so as to select the system RAM (HM6116) ...</p>
93	Chip select signal	Chip select signal for the PC-1500A Refer to following chart for the PC-1500A.

Chip select signal for the PC-1500A

			0000H	
Y0 (<u>Y0</u>)				OPTIONAL USER MEMORY
			3FFFH	
S0 (<u>Y0</u>)			4000H	
			47FFH	
S1 (<u>Y1</u>)			4800H	
			4FFFH	STANDARD USER MEMORY
S2 (<u>Y2</u>)			5000H	
			57FFH	
S3 (<u>Y3</u>)			5800H	
			5FFFH	
S4 (<u>Y4</u>)			6000H	
			67FFH	OPTIONAL USER MEMORY
S5 (<u>Y5</u>)			6800H	
			6FFFH	
S6 (<u>Y6</u>)			7000H	
			75FFH	INHIBITED
V2 (<u>Y6</u>)			7600H	
			76FFH	
V3 (<u>Y2</u>)			7700H	
			77FFH	STANDARD USER AND SYSTEM MEMORY
S7 (<u>Y7</u>)			7800H	
			7FFFH	MACHINE LANGUAGE AREA (7C01H~7FFFH)
Y2 (<u>Y2</u>)			8000H	CE-150 SYSTEM PROGRAM,I/O PORT
				CE-153 I/O PORT
				CE-158 SYSTEM PROGRAM
			BFFFH	
Y3 (<u>Y3</u>)			C000H	PC-1500A SYSTEM PROGRAM
				I/O PORT
				CE-158 I/O PORT
				UART
			FFFFH	

NOTE: S0~S7, V2, and V3 are applicable only for MEO area.

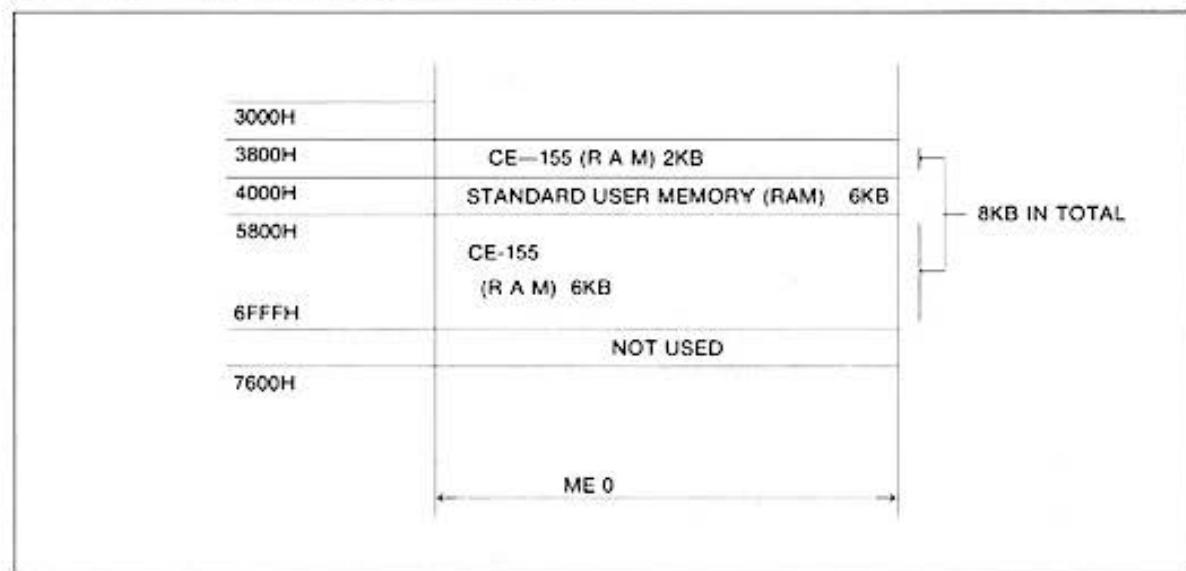
Page	PC-1500	PC-1500A
94	4-2-4. PC-1500 system memory map ① ... With the PC-1500, ME0 memory area 4000H thru 47FFH and 7600H thru 7BFFH are used for the user memory and C000H thru FFFFH for the system program. . . .	4-2-4. PC-1500A system memory map ① ... With the PC-1500A, ME0 memory area 4000H thru 57FFH and 7600H thru 7FFFH are used for the user memory and C000H thru FFFFH for the system program. . . .
95	MEMORY MAP I	MEMORY MAP I Refer to following MAP I for the PC-1500A.
96	MEMORY MAP II	MEMORY MAP II Refer to following MAP II for the PC-1500A.

MEMORY MAP I for the PC-1500A

- Area reserved for option will become unused, in case the respective option is not connected.
- Refer to "Memory map II and III" for *
- PV and PU are bank select signals.
- Shadowed area can not be used.

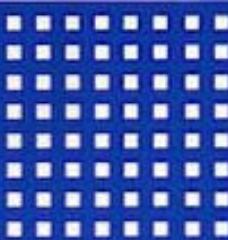
MEMORY MAP II for the PC-1500A

MEMORY MAP when the CE-155 is used.



Page	PC-1500	PC-1500A																														
102	4-3-1. 40-pin connector <table> <thead> <tr> <th>Pin no.</th> <th>Signal name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>5</td> <td>S4</td> <td>Address designation of 0000H~3FFFH</td> </tr> <tr> <td>16</td> <td>S1</td> <td>Address designation of 4800H~4FFFFH</td> </tr> <tr> <td>17</td> <td>S2</td> <td>Address designation of 5000H~57FFFH</td> </tr> <tr> <td>18</td> <td>S3</td> <td>Address designation of 5800H~5FFFFH</td> </tr> </tbody> </table>	Pin no.	Signal name	Description	5	S4	Address designation of 0000H~3FFFH	16	S1	Address designation of 4800H~4FFFFH	17	S2	Address designation of 5000H~57FFFH	18	S3	Address designation of 5800H~5FFFFH	4-3-1. 40-pin connector <table> <thead> <tr> <th>Pin no.</th> <th>Signal name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>5</td> <td>NC</td> <td>NC</td> </tr> <tr> <td>16</td> <td>S3</td> <td>Address designation of 5800H~5FFFFH</td> </tr> <tr> <td>17</td> <td>S4</td> <td>Address designation of 6000H~67FFFH</td> </tr> <tr> <td>18</td> <td>S5</td> <td>Address designation of 6800H~6FFFFH</td> </tr> </tbody> </table> <p>The above portion should be changed.</p>	Pin no.	Signal name	Description	5	NC	NC	16	S3	Address designation of 5800H~5FFFFH	17	S4	Address designation of 6000H~67FFFH	18	S5	Address designation of 6800H~6FFFFH
Pin no.	Signal name	Description																														
5	S4	Address designation of 0000H~3FFFH																														
16	S1	Address designation of 4800H~4FFFFH																														
17	S2	Address designation of 5000H~57FFFH																														
18	S3	Address designation of 5800H~5FFFFH																														
Pin no.	Signal name	Description																														
5	NC	NC																														
16	S3	Address designation of 5800H~5FFFFH																														
17	S4	Address designation of 6000H~67FFFH																														
18	S5	Address designation of 6800H~6FFFFH																														
154	5-1. PC-1500 Circuit diagram	5-1. PC-1500A Circuit Diagram For the PC-1500A, refer to the circuit diagram on the next page.																														





SHARP

SHARP CORPORATION OSAKA,JAPAN

CABLE ADDRESS: LABOMET OSAKA
TELEX No.AAB: LABOMETA J63428

Do not sale this PDF !!!