PERMISSIONS

# WANG

# INSTRUCTION PACKAGE

# Wang 500 and 600

# Operation Manual

Prepared as a

Faculty Development Project

For

Rock Valley College



By

Gene A. Streitmatter

Associate Professor, Electronics

August, 1976

Rockford, Illinois

# PREFACE

The Rock Valley College Technology Division has two types of Wang calculators available for use: the 500 series and the 600 series.

The purpose of this paper is to provide a condensed set of operating instructions to enable a person to become proficient on the Wang calculator in the least amount of time. A person wishing to learn the proper operation of both machines would have to study and absorb the information contained in nine books totaling over one thousand pages which are provided by Wang Labs. This would consume many many hours of valuable time which an instructor can profitably use in other endeavors.

Hopefully, this manual will save the instructor an appreciable amount of the time normally needed to master these machines.

I would like to thank the following Rock Valley students who contributed greatly to the production of this manual by offering constructive criticism and serving as co-authors for some sections: David Dahlen, James Dunlea, George Fulop and Drew Reid.

Gene A. Streitmatter

Rockford, Illinois

August, 1976

CONTENTS

I. INTRODUCTION TO MACHINE

A. Turn-on Procedure

B. Keyboard Description

1. Data entry

2. Scratch register

3. Math and algebra

4. Programming keys

5. Upper registers

6. Mode switches

7. Special function switches

8. Tape switches

C. Auxiliary Equipment

1. Auxiliary equipment 500

2. Auxiliary equipment 600

# INTRODUCTION TO MACHINE

The Wang calculators are extremely capable machines. The Rock Valley College Technology Division has two types available for use: the 500 series and the 600 series. The two machines have many similar operating procedures and features. It should be made clear at this point that the 600 series is capable of all the operations performed by the 500 plus many additional features.

Some valuable statistics about the two calculators follows in a condensed form.

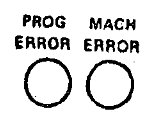
| | WANG 500 | WANG 600 |
|---|---|---|
| Type of Display | Nixie | Segmented |
| Number of Digits | 12 | 12 |
| Scientific Notation Limit | ±99 | ±99 |
| Printout Capability | Yes<br>Auto<br>Manual | Yes<br>Auto<br>Manual |
| Mathematical Operation Similar on Both Machines | Scientific or Floating Decimal Point | |
| Language | Keyboard | Keyboard |
| Tape Cassette | Yes | Yes |
| Number of Programming Steps | 319 | 823 |
| Number of Storage Registers | 40 | 103 |
| Keyboard Registers | 16 | 16 |
| Type of Address | Direct | Direct/Inidrect |

| | WANG 500 | WANG 600 |
|---|---|---|
| Type of Peripheral Equipment Available | Card Reader<br>Classroom Display | Card Reader<br>Plotter<br>ROM Pak<br>Typewriter I/O |
| General Library Available | Yes | Yes |
| Multiple Function Keys | Yes | Yes |
| Overflow | Flashing Display | |
| Self Loading | No | Yes |

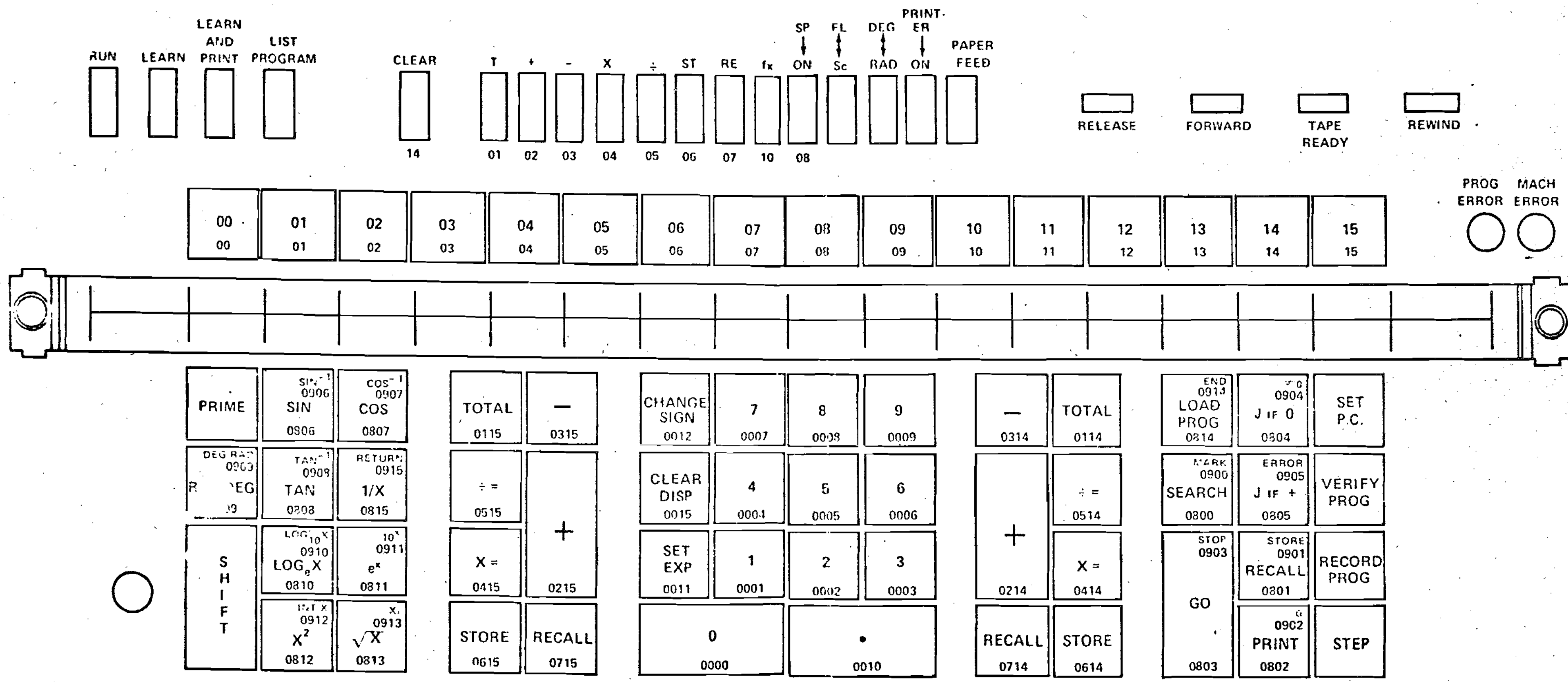
The first eleven sections of this manual will deal only with the Wang 500 and its use. The last nine will deal with the 600 and its equipment features.

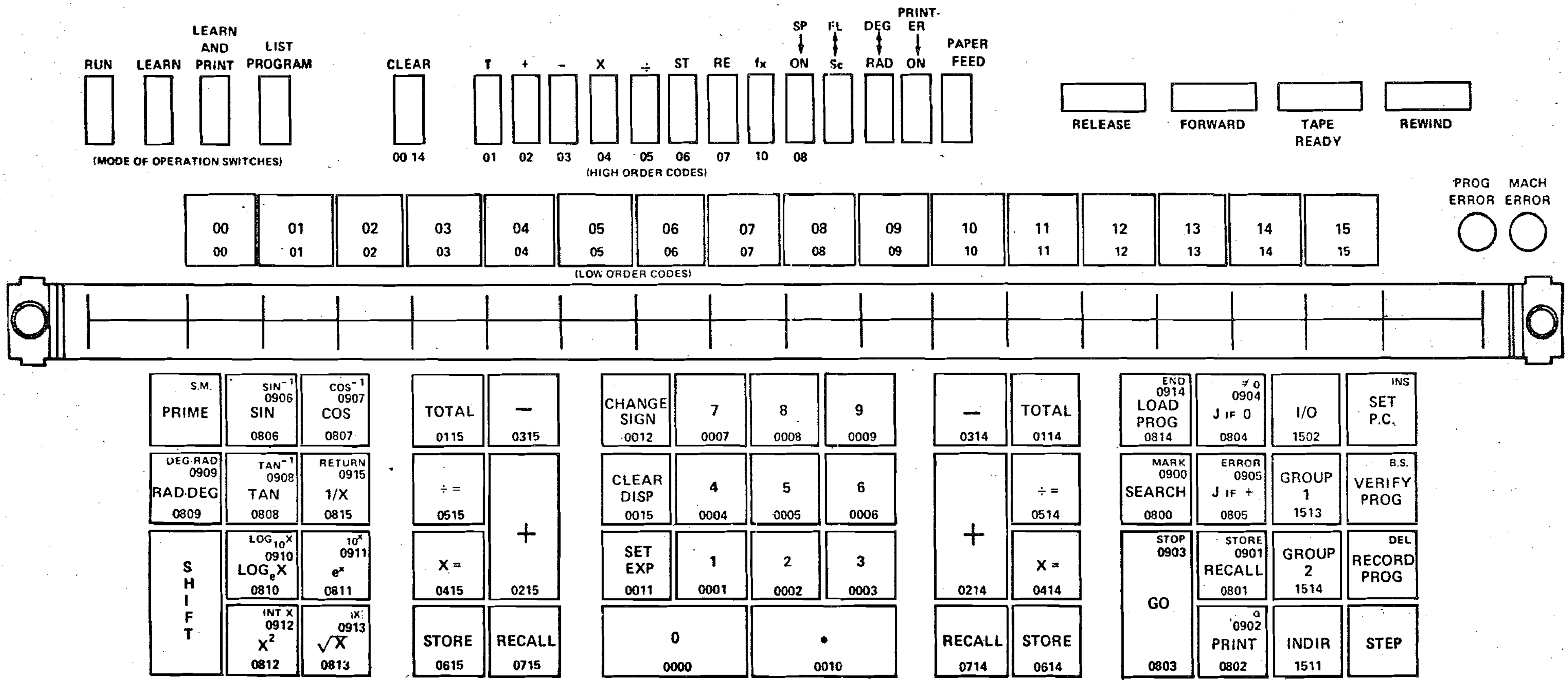
Turn-on Procedure

Both machines are turned on in the same manner. As you sit facing the machine, the power switch is in the back on the left. Simply reach around the back with the left hand near the bottom and tip the switch in. As the machine turns on, it can be seen that the program error light and the machine error light are on and the display is indicating a calculation overflow which is a flashing display.



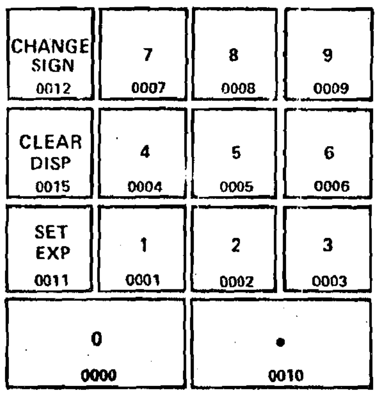

The correction for this situation is to depress the PRIME and CLEAR keys. Both error lights should turn off and the display should stop flashing. The machine is now ready for use.


RUN
LEARN
LEARN AND PRINT
LIST PROGRAM
CLEAR
14
T + - X ÷ ST RE fx
SP ON
FL Sc
DEG RAD
PRINTER ON
PAPER FEED
01 02 03 04 05 06 07 10 08
RELEASE
FORWARD
TAPE READY
REWIND
PROG ERROR
MACH ERROR
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
PRIME
SIN 0806
COS 0807
TAN 0808
1/X 0815
SHIFT
LOG<sub>e</sub>X 0810
e^x 0811
X^2 0812
√X 0813
TOTAL 0115
— 0315
÷ = 0515
X = 0415
+ 0215
STORE 0615
RECALL 0715
CHANGE SIGN 0012
CLEAR DISP 0015
SET EXP 0011
7 0007
8 0008
9 0009
4 0004
5 0005
6 0006
1 0001
2 0002
3 0003
0 0000
• 0010
— 0314
TOTAL 0114
+ 0214
÷ = 0514
X = 0414
RECALL 0714
STORE 0614
END 0914 LOAD PROG 0814
0904 J IF 0 0804
SET P.C.
MARK 0900 SEARCH 0800
ERROR 0905 J IF + 0805
VERIFY PROG
STOP 0903 GO 0803
STORE 0901 RECALL 0801
RECORD PROG
0902 PRINT 0802
STEP


RUN
LEARN
LEARN AND PRINT
LIST PROGRAM
(MODE OF OPERATION SWITCHES)
CLEAR
00 14
T + − X ÷ ST RE fx
SP ON
FL Sc
DEG RAD
PRINTER ON
PAPER FEED
01 02 03 04 05 06 07 10 08
(HIGH ORDER CODES)
RELEASE
FORWARD
TAPE READY
REWIND
PROG ERROR
MACH ERROR
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
(LOW ORDER CODES)
S.M. PRIME
SIN⁻¹ 0906 SIN 0806
COS⁻¹ 0907 COS 0807
DEG-RAD 0909 RAD-DEG 0809
TAN⁻¹ 0908 TAN 0808
RETURN 0915 1/X 0815
SHIFT
LOG10X 0910 LOGeX 0810
10x 0911 ex 0811
INT X 0912 X2 0812
IXI 0913 √X 0813
TOTAL 0115
− 0315
÷ = 0515
+ 0215
X = 0415
STORE 0615
RECALL 0715
CHANGE SIGN 0012
7 0007
8 0008
9 0009
CLEAR DISP 0015
4 0004
5 0005
6 0006
SET EXP 0011
1 0001
2 0002
3 0003
0 0000
• 0010
− 0314
TOTAL 0114
+ 0214
÷ = 0514
X = 0414
RECALL 0714
STORE 0614
END 0914 LOAD PROG 0814
≠ 0 0904 J IF 0 0804
I/O 1502
INS SET P.C.
MARK 0900 SEARCH 0800
ERROR 0905 J IF + 0805
GROUP 1 1513
B.S. VERIFY PROG
STOP 0903 GO 0803
STORE 0901 RECALL 0801
GROUP 2 1514
DEL RECORD PROG
Q 0902 PRINT 0802
INDIR 1511
STEP

Keyboard Description

1. If you are sitting in front of the machine, you should refer to the keyboard. If not, you should refer to the diagram included here.



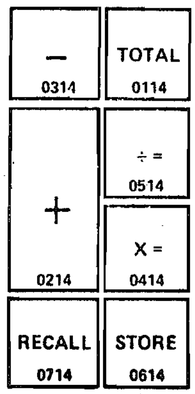

The keys shown at the left will be referred to as the data entry keys from now on. They are similar to any standard calculator or adding machine and are used in the same manner. There are three keys shown here that are different from those found on an adding machine: CHANGE SIGN, CLEAR DISPLAY, and SET EXP. The change sign key changes the algebraic sign of the displayed number or exponent. {If the set exp key is depressed prior to the change sign, the sign of the exponent will be changed rather than the sign of the number itself.}

The set exponent key activates the scientific notation capability of the machine. When depressed, two additional digital displays and a sign display are made available and the power of 10 may be used in expressing the value. The power of ten must be expressed in two digits which establish a limit of $10^{+99}$ to $10^{-99}$. Therefore, in order to express $3.576 \times 10^{-5}$, you would key the following sequence: 3, ., 5, 7, 6, Set Exp, chage sign, 0, 5.

The clear display key is used to erase the displayed value if an error is made in entering the number.

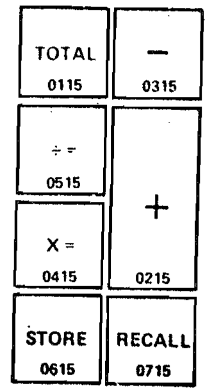
DO NOT USE $\boxed{\text{PRIME}}$ OR USE THE RED $\boxed{\text{CLEAR}}$ BUTTON FOR THIS PURPOSE.

2.



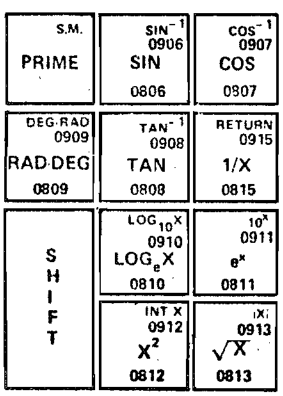

LEFT REG
{15}

The keys at the left and right of the Data Entry Keys are the arithmetic controls for the Left and Right SCRATCH REGISTERS. The Left Register is Reg 15, the Right Register is Reg 14. They will be called Scratch Registers when both are being referred to and by Left and Right if specific reference is made to one or the other. The specific use of these keys will be discussed in the section on arithmetic operations.




RIGHT REG
{14}

3.



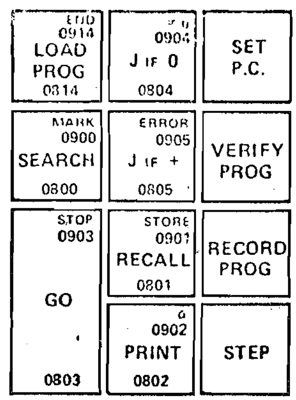

The keys shown here will be referred to in the future as the Math keys as a group or by their function in a specific case. Notice that each of these keys except PRIME and SHIFT have a second function imprinted in the upper right hand corner. These functions can be performed by using the Shift key in conjunction with the Function key desired.

For example, in order to find $L_n$ of 30, enter 30 on the display and depress the $LOG_eX$ key. However, to get the $Log_{10}$ 30, enter 30 on the display and depress Shift, $Log_{10}X$

{in the upper case}. The Shift key gives access to the upper case functions in a way similar to the shift to capital in typing. Once used, however, the machine automatically returns to the lower case functions and does not stay in the upper case. The Prime key will eliminate an overflow indication {flashing display} and the program error indicator.

THE PRIME KEY ALSO RESETS THE PROGRAM COUNTER TO STEP "0" SO UNTIL PROGRAMMING OPERATIONS HAVE BEEN DISCUSSED, USE IT SPARINGLY.

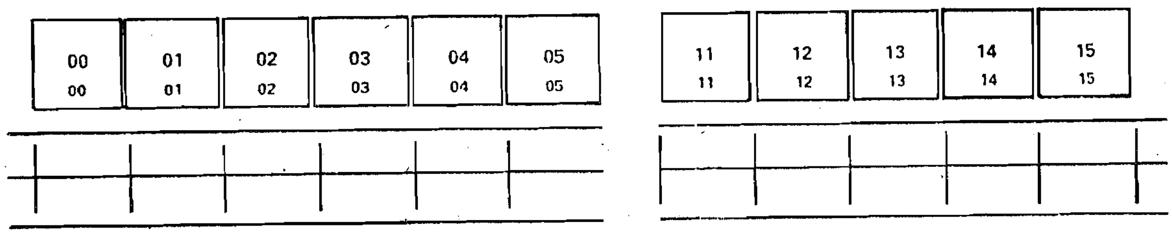
4. All these keys, with the exception of the Recall and Print key, are used exlusively when dealing with programming and will therefore be referred to as the programming keys. The Recall key is used in conjunction with the Upper Registers and the special function switches to store data away in what will be called substorage and will be discussed along with the special function switches.




The Print key may be used any time a permanent record of the display is desired. {The Printer On switch at the top must be on for the printer to operate under any circumstances.} The Print key is also used for one other operation and that is to generate an $\alpha$ {alpha} code. This code, in turn, is used in several ways in programming. For example, α 00 generates π {Pi} on the display register. To get the value of π , key Shift-Print-Register {00} and pi will appear on the display.

NOTE: On the 600, the f{x} special function key must also be depressed.

5.



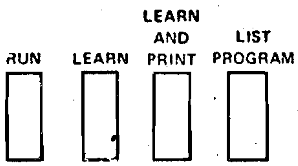

UPPER REGISTERS

Registers 00 through 15

These keys give the operator direct access to the first sixteen Data Storage Registers which will be designated the Upper Registers. They are used basically for three functions: {1} access to Storage Register, {2} generation of low order portions of programming codes, {3} decimal placement and movement in printing and display respectively. Function {1} will be discussed in the arithmetic section and {2} and {3} in their appropriate places in the programming section.

Notice the strip of tape below the keys laid out in blocks. These blocks can be labeled for use by the operator to facilitate and remember Register data such as what is stored in each one by symbol or what program it calls. In general, the tape is used to record register assignments and program numbers.

6.



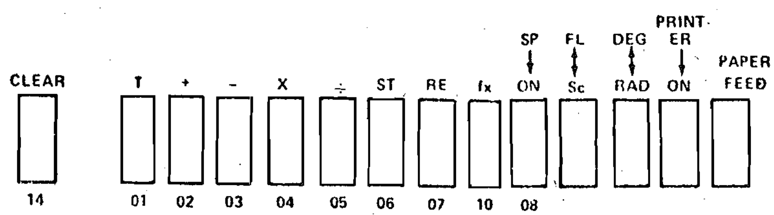

These switches are referred to as the Mode Switches. They determine the way in which the machine is to be used. The run mode is the mode in which the machine will most commonly be operated. In this mode, the machine will operate as a standard desktop calculator and execute previously programmed operations when called upon to do so.

In both the Learn, and Learn and Print modes, the machine will "remember" or store away in memory the sequence of keystrokes made while in this mode. It might rightly be called the programming mode. The Learn and Print mode additionally gives you a written record of keystrokes as the program is developed.

The List Program mode will list out the keystrokes in memory from the step where it is, at the beginning until the end of program code is encountered.

These modes will be extensively discussed in the programming section.

7.



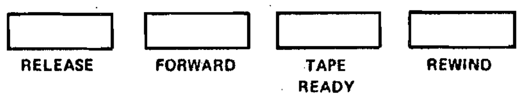

These keys, located in the top center of the keyboard, will be referred to as the function switches and the clear switch.

The Clear switch will clear all data stored in the upper registers and scratch registers but not the display.

All of these switches are multiple function switches and are used to accomplish basically four things: {1} provide arithmetic functions for the upper registers, {2} generate high order codes for programming, {3} determine alpha labeling, {4} determine display mode.

The Printer and Paper Feed switches are only used in manual operation. That is, the printer cannot be turned on from the keyboard and the paper can be advanced manually any time without affecting the programming or operation of the machine.

8.




The Tape Control switches are similar to those found on any normal tape cassette. RELEASE opens the cassette's compartment for insertion or removal of program tapes. FORWARD and REWIND are self-explanatory and need no comment. The TAPE READY switch means that the cassette drive motors are ready to respond to keyboard or programmed instructions. The different ways in which the tape can be used will be discussed in the section on tape storage of data and programs.

<u>Auxiliary Equipment</u>

1. The Wang 500 calculators were purchased with the following equipment features:

1) Rotational drum printer

2) Classroom display

3) Mark sense card reader

4) Cassette tape storage capability.

2. The Wang 600 has all of the above features and additionally incorporates:

1) Capacity for a ROM (read only memory)

2) Flatbed plotter

3) Capacity for an input-output writer

4) Capacity for peripheral memory

## II. BASIC ARITHMETIC

A. Arithmetic Key

B. Scratch Registers

C. Store, Recall, Total

D. Equation Handling {Simple}

E. Upper Register Technique

# BASIC ARITHMETIC

The one common error that most people make when first starting to use the Wang is in not beginning instructions to the machine in proper sequences. The $\boxed{\text{STORE}}$ key seems to be the culprit. For example, in order to divide 4 by 2, the key strokes would be as follows: $\boxed{4}$, $\boxed{\text{STORE}}$, $\boxed{2}$, $\boxed{\div}$.

Following is an analysis of what has just been done. First, a number was generated by keying the 4. On the Wang, keying a number does not automatically put it into the machine's operating registers. The number, when keyed, only exists at one place in machine--on the display. You must now give that number to the machine for further arithmetic operations. This is done by pressing the store key. When $\boxed{\text{STORE}}$ is keyed, the machine will take what is on the display at that time and write it into the memory register being keyed. If anything was in that register previously, it is automatically erased when the store command is generated. Another number (2) was then generated on the display. Again, it exists only in the display. When you key the $\boxed{\div}$, you are generating the arithmetic instruction [divide the number on the display into this register.] The machine does this and automatically re-stores the answer number back into the register. (Later, when programming the machine, it will save you steps if you remember that the machine always stores the answers from arithmetic instructions back in the register.} You do not have to store the answer to prevent it from being lost.

Arithmetic Key

The operator must always keep in mind that, when he generates an arithmetic instruction, the machine will use the number on the display, operate on the register keyed and store the answer of the operation in the register keyed. This answer is then available for further operations. In verbal form, the arithmetic keys generate the following instructions:

+ add the display to register keyed
- subtract the display from register keyed
÷ divide the display into register keyed
x multiply the display by register keyed

Remember, the answer is always stored in the register keyed.

Scratch Registers

Both the Wang 500 and 600 have sixteen registers available for arithmetic operations: Registers $\boxed{00}$ → $\boxed{15}$. Registers 14 and 15, however, can be addressed and operated from two places on the keyboard. One method of addressing these registers is by using the Blue keys at the left and right of the Data Entry keys. {Right = Reg 14; Left = Reg 15} These two registers, commonly called Scratch Registers, may also be operated using the function selectors at the top of the keyboard in combination with the upper register keys $\boxed{14}$ and $\boxed{15}$.

Store, Recall, Total

The $\boxed{\text{STORE}}$ key in the scratch registers will store whatever is on the display into the register for which the key has been depressed. Very much like a tape recorder, it also erases anything previously stored in the register.

The $\boxed{\text{RECALL}}$ key duplicates on the display whatever is in the register at the time the key is depressed. It is important to remember that using the recall key does not empty the register; the number is still in the register memory.

The $\boxed{\text{TOTAL}}$ key removes whatever is in the register and transfers the number to the display. It must be remembered that the number is removed from the register and the register then contains "0."

The decision as to whether Recall or Total should be used hinges on two considerations: (1) Does that number need to remain in memory? (2) Does the operator need that register available for further use?

An example of when the Recall and Total keys would be used follows:

$$A \ (B/A) + A = X$$

Notice: A is going to be used three times.

PROCEDURE:

Key value B

B is now in Left Register

| | |
|---|---|
| Key store left | |
| Key value A | |
| Key store right | A now is in two places: on the display and in the Right Register |
| Key ÷ left | Display is divided into Left Register. The value B/A is in the Left Register. |
| Key recall right | Brings A to the display and also leaves it in Right Register for further use. |
| Key {x} left | Display times Left. A {B/A} now in Left Register |
| Key total right | Removes A from Right Register |
| Key + left | Adds display to Left Register A {B/A} + A now in Left Register and Right Register has been cleared for future use. |

The completes the introduction to the Scratch Register arithmetic keys and what they do.

Equation Handling {Simple}

Below are examples of how to handle some simple equations.

--Example 1--

$$\frac{AB}{A + B}$$

| Procedure {correct} | Procedure {incorrect} |
|---|---|
| Key value A | Key value A |
| Store left | Store left |
| Store right | Key value B |
| Key value B | {+} left |
| {x} left | Key value A |
| Reg value B | Store right |
| {+} right | Key value B |
| {:} left | {x} right |

Recall left
{÷} right

Notice that the correct procedure is two steps shorter. Both procedures generate a correct answer. However, the shorter procedure will be the more desirable one later when minimum steps and most efficient handling are essential. {You should always strive for the most effeicient procedure.}

Two more examples would be:

--Example 2--

A + B = C

PROCEDURE:

Key A
Store left
Key B
+ L
Answer on display

--Example 3--

3{A+2} - B{C+3}

PROCEDURE:

Key A
Store left
Key 2
+ left
Key 3
x L
Key C
Store right
Key 3
+ right
Key B
x right
- left
Answer on display

It is good to practice <u>now</u> for the programming section <u>later</u>. You should practice writing out the procedural steps of several simple equations provided in the supplement.

An attempt to solve problems using only the two scratch registers will lead to the discovery that there are many problems which cannot be handled at this point. Therefore, it becomes necessary to explain how the upper register can be utilized in conjunction with the special function switches.

to expand your ability to handle problems.

Upper Register Technique

It may already be evident to you that the arithmetic keys of the scratch registers are the same commands as the first seven special function switches on the left end. To do arithmetic with the upper register, two key strokes are required. The switch sets up the machine command for what it is to do and keying one of the registers tells it on which register it is to operate. By using one switch and one register for each operational step, the same operations as done with the scratch keys can be accomplished.

Example: Add A + B in the 03 register.

| | | |
|---|---|---|
| Key A<br>STORE SWITCH | 03 | Puts A in Register 3 |
| Key B<br>+ SWITCH | 03 | Adds B to Register 3 |

You must remember that doing arithmetic in the upper register requires two key strokes (one step).

(1) a special function switch (what to do)

(2) a register (where to do it)

You should now practice the problems provided for this section in the supplement.

The upper registers and scratch register may be used separately, together or in any number of combinations.

Following is a procedure to solve six problems at once for six different values of x.

$$\frac{3x + 4}{6}$$

First, depress the $\boxed{\text{STORE SWITCH}}$. Then key the values of x, one after the other, and depress the register into which each one is to go as follows:

Key 3 → $\boxed{03}$

Key 5.5 → $\boxed{02}$

Key 6 → $\boxed{01}$

Key 4 → $\boxed{06}$

Key 2 → $\boxed{04}$

Key .5 → $\boxed{05}$

Each number is now stored in its appropriate register.

Now depress the x (multiply) SWITCH, then Key 3 and the registers one after the other: Key 3 → 01, Key 3 → 02, Key 3 → 03, etc. Each register has just been multiplied by 3.

Depress (+) switch, key 4 → 01, Key 4 → 02, etc. Depress (÷) switch, Key 6 → 01, Key 6 → 02, etc.

All six problems have now been solved and the answers are in register 1 through 6. To see what they are, depress the Recall switch and key each register in turn and the answers will appear on the display.

The use of the registers (scratch and upper) to solve problems is only limited by your imagination.

## III. SPECIAL FUNCTION KEYS

A. Meaning of Codes

B. Other Switches

1. fL - SC
2. Deg, Rad
3. Printer
4. Paper Feed

C. Substorage Access

1. Memory layout
2. Addressing registers

D. Alpha Codes

# SPECIAL FUNCTION KEYS

## Meaning of Codes

Every key on the Wang 500 and 600, with the exception of the modes switches and the tape control, generate operational codes. These codes are generated in four digits {xx yy}. The first two digits {xx} are high order functional commands. They, in essence, tell the machine what to do {store, add, total, etc.}. The second two digits {yy} determine where the operation is to be accomplished {Reg. 1,2,3,etc.}.

The following is a listing of the machine codes and the switches, keys and registers necessary to generate those codes.

Special function switches generate high order codes.

| | |
|---|---|
| 01 yy - Total | 09 yy - SP Total |
| 02 yy - + | 10 yy - SP + |
| 03 yy - - | 11 yy - SP - |
| 04 yy - x | 12 yy - SP x |
| 05 yy - ÷ | 13 yy - SP ÷ |
| 06 yy - $S_T$ | 14 yy - SP ST |
| 07 yy - Re | 15 yy - SP Re |
| 08 yy - SP | |

Register keys generate low order codes.

| | |
|---|---|
| xx 00 - 00 | xx 08 - 08 |
| xx 01 - 01 | xx 09 - 09 |
| xx 02 - 02 | xx 10 - 10 |
| xx 03 - 03 | xx 11 - 11 |
| xx 04 - 04 | xx 12 - 12 |
| xx 05 - 05 | xx 13 - 13 |
| xx 06 - 06 | xx 14 - 14 |
| xx 07 - 07 | xx 15 - 15 |

Using these switches and keys in combination allows you to generate any code you wish.

The next three pages are a listing of the codes and their meaning to the machine. The first page is a layout of all the instructional codes you use when programming. You should be aware, however, that there are other codes that can be generated using the Print key and the Shift key to change the code meaning. The second page is a listing of the alphanumeric codes that the machine can print and their meaning. The third page is a code listing of codes generated by the keyboard keys.

# PROGRAM CODES

| CODE | KEY | CODE | KEY | CODE | KEY | CODE | KEY |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 0015 | CLD | 0801 | RECALL | 0901 | STORE |
| 0001 | 1 | 0114 | TOTAL R | 0802 | PRINT | 0902 | α |
| 0002 | 2 | 0115 | TOTAL L | 0803 | GO | 0903 | STOP |
| 0003 | 3 | 0214 | + R | 0804 | J if 0 | 0904 | J if ≠ 0 |
| 0004 | 4 | 0215 | + L | 0805 | J if + | 0905 | J if ERROR |
| 0005 | 5 | 0314 | − R | 0806 | SIN | 0906 | $SIN^{-1}$ |
| 0006 | 6 | 0315 | − L | 0807 | COS | 0907 | $COS^{-1}$ |
| 0007 | 7 | 0414 | X R | 0808 | TAN | 0908 | $TAN^{-1}$ |
| 0008 | 8 | 0415 | X L | 0809 | RAD − DEG | 0909 | DEG − RAD |
| 0009 | 9 | 0514 | ÷ R | 0810 | $Log_e X$ | 0910 | $Log_{10} X$ |
| 0010 | • | 0515 | ÷ L | 0811 | $e^x$ | 0911 | $10^x$ |
| 0011 | SET EXP | 0614 | ST R | 0812 | $x^2$ | 0912 | INT X |
| 0012 | CHS | 0615 | ST L | 0813 | $\sqrt{X}$ | 0913 | \|X\| |
| 0013 | SEARCH | 0714 | RE R | 0814 | LOAD PROG | 0914 | END |
| | & STOP | 0715 | RE L | 0815 | 1/X | 0915 | RETURN |
| 0014 | CLEAR | 0800 | SEARCH | 0900 | MARK | | |

**Decimal Shifting Left •**

| CODE | KEY |
|---|---|
| 1101 | $10^{-1}$ |
| 1102 | $10^{-2}$ |
| 1103 | $10^{-3}$ |
| 1104 | $10^{-4}$ |
| 1105 | $10^{-5}$ |
| 1106 | $10^{-6}$ |
| 1107 | $10^{-7}$ |
| 1108 | $10^{-8}$ |
| 1109 | $10^{-9}$ |
| 1110 | $10^{-10}$ |
| 1111 | $10^{-11}$ |
| 1112 | $10^{-12}$ |
| 1113 | $10^{-13}$ |
| 1114 | $10^{-14}$ |
| 1115 | $10^{-15}$ |

**Decimal Shifting Right •**

| CODE | KEY |
|---|---|
| 1001 | $10^{1}$ |
| 1002 | $10^{2}$ |
| 1003 | $10^{3}$ |
| 1004 | $10^{4}$ |
| 1005 | $10^{5}$ |
| 1006 | $10^{6}$ |
| 1007 | $10^{7}$ |
| 1008 | $10^{8}$ |
| 1009 | $10^{9}$ |
| 1010 | $10^{10}$ |
| 1011 | $10^{11}$ |
| 1012 | $10^{12}$ |
| 1013 | $10^{13}$ |
| 1014 | $10^{14}$ |
| 1015 | $10^{15}$ |

**Register Keys †**

| CODE | KEY |
|---|---|
| XX00 | 00 |
| XX01 | 01 |
| XX02 | 02 |
| XX03 | 03 |
| XX04 | 04 |
| XX05 | 05 |
| XX06 | 06 |
| XX07 | 07 |
| XX08 | 08 |
| XX09 | 09 |
| XX10 | 10 |
| XX11 | 11 |
| XX12 | 12 |
| XX13 | 13 |
| XX14 | 14 |
| XX15 | 15 |

**Selector Switches †**

| CODE | SWITCH |
|---|---|
| 01YY | TOTAL |
| 02YY | + |
| 03YY | − |
| 04YY | X |
| 05YY | ÷ |
| 06YY | STORE |
| 07YY | RECALL |
| 08YY | SP |
| 10YY | f(x) |
| 11YY | * F(x) |
| 14YY | SP & ST |

* Shift

**Alpha Characters ••**

| CODE | SWITCH |
|---|---|
| 00YY | All up |
| 01YY | SP up T down |
| 02YY | SP up + down |
| 03YY | SP up − down |
| 04YY | SP up X down |
| 05YY | SP up ÷ down |
| 06YY | SP up ST down |
| 07YY | SP up RE down |
| 08YY | SP down |
| 09YY | SP down T down |
| 10YY | SP down + down |
| 11YY | SP down − down |
| 12YY | SP down X down |
| 13YY | SP down ÷ down |
| 14YY | SP down ST down |
| 15YY | SP down RE down |

**Decimal Setting ••**

| CODE | KEY |
|---|---|
| XX00 | 00 |
| XX01 | 01 |
| XX02 | 02 |
| XX03 | 03 |
| XX04 | 04 |
| XX05 | 05 |
| XX06 | 06 |
| XX07 | 07 |
| XX08 | 08 |
| XX09 | 09 |
| XX10 | 10 FP |
| XX11 | 11 SN |
| 0015 | CLD PF |

**Miscellaneous**

| CODE | KEY | |
|---|---|---|
| 0902 | α | π |
| 1000 | 00 | |
| 0902 | α | PAUSE |
| 0903 | STOP | |

•Must be preceded by an α command.

†Selector Switches (high order) Register Keys (low order).

••Must be preceded by a Print command.

••Alpha character (high order) Decimal Setting (low order).

## PRINTING CODES

All command preceded by a print command
Require two additional strokes, a switch (High Order) and a
register (low order).

| High Order Code | Switch Position | Meaning | | Low Order Code | Key Reg | Meaning |
|---|---|---|---|---|---|---|
| 00 | all up | Label | X | 00 | 00 | 0 decimal |
| 01 | T | Label | Y | 01 | 01 | 1 decimal |
| 02 | + | " | Z | 02 | 02 | 2 " |
| 03 | - | " | A | 03 | 03 | 3 " |
| 04 | x | " | B | 04 | 04 | 4 " |
| 05 | ÷ | " | C | 05 | 05 | 5 " |
| 06 | ST | " | D | 06 | 06 | 6 " |
| 07 | Re | " | E | 07 | 07 | 7 " |
| 08 | (SP down) | " | F | 08 | 08 | 8 " |
| 09 | SP T | " | G | 09 | 09 | 9 " |
| 10 | SP + | " | H | 10 | 10 | 10 floating |
| 11 | SP - | " | I | 11 | 11 | 11 scientific |
| 12 | SP x | " | J | | | |
| 13 | SP ÷ | " | K | | | |
| 14 | SP ST | " | L | | | |
| 15 | SP Re | " | M | 15 | 15 | 15 paper feed one line |

## REGISTER KEY CODES

| | | | |
|---|---|---|---|
| 0018 | 01 | 15 | [illegible] |
| 0019 | 03 | 15 | −15 |
| 0020 | 05 | 15 | ÷15 |
| 0021 | 02 | 15 | +15 |
| 0022 | 04 | 15 | ×15 |
| 0023 | 06 | 15 | [illegible] |
| 0024 | 07 | 15 | E15 |

## MATH GROUP KEY CODES

| | | | |
|---|---|---|---|
| 0000 | 03 | 06 | * [illegible] |
| 0001 | 03 | 07 | * [illegible] |
| 0002 | 03 | 08 | * [illegible] |
| 0003 | 05 | 15 | * 1/x |
| 0004 | 05 | 10 | * [illegible] |
| 0005 | 05 | 11 | * $e^x$ |
| 0006 | 05 | 12 | * $x^2$ |
| 0007 | 05 | 13 | * √x |
| 0008 | 03 | 09 | * [illegible] |
| 0009 | 09 | 09 | * [illegible] |
| 0010 | 09 | 06 | * [illegible] |
| 0011 | 09 | 07 | * [illegible] |
| 0012 | 09 | 08 | * [illegible] |
| 0013 | 09 | 15 | * [illegible] |
| 0014 | 09 | 10 | * [illegible] |
| 0015 | 09 | 11 | * [illegible] |
| 0016 | 09 | 12 | * I |
| 0017 | 09 | 13 | * [illegible] |

## DATA ENTRY KEY CODES

| | | | |
|---|---|---|---|
| 0025 | 00 | 10 | E10 |
| 0026 | 00 | 00 | E0 |
| 0027 | 00 | 01 | E1 |
| 0028 | 00 | 02 | E2 |
| 0029 | 00 | 03 | E3 |
| 0030 | 00 | 04 | E4 |
| 0031 | 00 | 05 | E5 |
| 0032 | 00 | 06 | E6 |
| 0033 | 00 | 07 | E7 |
| 0034 | 00 | 08 | E8 |
| 0035 | 00 | 09 | E9 |
| 0036 | 00 | 11 | E11 |
| 0037 | 00 | 15 | E15 |
| 0038 | 00 | 12 | E12 |

## PROGRAMING GROUP KEY CODES

| | | | |
|---|---|---|---|
| 0000 | 08 | 14 | * [illegible] |
| 0001 | 08 | 00 | * S |
| 0002 | 08 | 03 | * [illegible] |
| 0003 | 08 | 04 | * [illegible] |
| 0004 | 08 | 05 | * [illegible] |
| 0005 | 08 | 01 | * [illegible] |
| 0006 | 08 | 02 | * [illegible] |
| 0007 | 09 | 03 | * [illegible] |
| 0008 | 09 | 00 | * [illegible] |
| 0009 | 09 | 14 | * [illegible] |
| 0010 | 09 | 04 | * [illegible] |
| 0011 | 09 | 05 | * [illegible] |
| 0012 | 09 | 01 | * ST |
| 0013 | 09 | 02 | * α |

## REGISTER KEY CODES

| | | | |
|---|---|---|---|
| 0 0 1 8 | 0 1 | 1 5 | T 15 |
| 0 0 1 9 | 0 3 | 1 5 | - 15 |
| 0 0 2 0 | 0 5 | 1 5 | ÷ 15 |
| 0 0 2 1 | 0 2 | 1 5 | + 15 |
| 0 0 2 2 | 0 4 | 1 5 | × 15 |
| 0 0 2 3 | 0 6 | 1 5 | ST 15 |
| 0 0 2 4 | 0 7 | 1 5 | RE 15 |

## MATH GROUP KEY CODES

| | | | |
|---|---|---|---|
| 0 0 0 0 | 0 8 | 0 6 | * SN |
| 0 0 0 1 | 0 8 | 0 7 | * CS |
| 0 0 0 2 | 0 8 | 0 8 | * TN |
| 0 0 0 3 | 0 8 | 1 5 | * $\frac{1}{x}$ |
| 0 0 0 4 | 0 8 | 1 0 | * $L_e$ |
| 0 0 0 5 | 0 8 | 1 1 | * $e^x$ |
| 0 0 0 6 | 0 8 | 1 2 | * $x^2$ |
| 0 0 0 7 | 0 8 | 1 3 | * $\sqrt{x}$ |
| 0 0 0 8 | 0 8 | 0 9 | * RD |
| 0 0 0 9 | 0 9 | 0 9 | * DR |
| 0 0 1 0 | 0 9 | 0 6 | * $S^{-1}$ |
| 0 0 1 1 | 0 9 | 0 7 | * $C^{-1}$ |
| 0 0 1 2 | 0 9 | 0 8 | * $T^{-1}$ |
| 0 0 1 3 | 0 9 | 1 5 | * RT |
| 0 0 1 4 | 0 9 | 1 0 | * LG |
| 0 0 1 5 | 0 9 | 1 1 | * $10^x$ |
| 0 0 1 6 | 0 9 | 1 2 | * I |
| 0 0 1 7 | 0 9 | 1 3 | * [X] |

## DATA KEY CODES

| | | | |
|---|---|---|---|
| 0 0 2 5 | 0 0 | 1 0 | E10 |
| 0 0 2 6 | 0 0 | 0 0 | E0 |
| 0 0 2 7 | 0 0 | 0 1 | E1 |
| 0 0 2 8 | 0 0 | 0 2 | E2 |
| 0 0 2 9 | 0 0 | 0 3 | E3 |
| 0 0 3 0 | 0 0 | 0 4 | E4 |
| 0 0 3 1 | 0 0 | 0 5 | E5 |
| 0 0 3 2 | 0 0 | 0 6 | E6 |
| 0 0 3 3 | 0 0 | 0 7 | E7 |
| 0 0 3 4 | 0 0 | 0 8 | E8 |
| 0 0 3 5 | 0 0 | 0 9 | E9 |
| 0 0 3 6 | 0 0 | 1 1 | E11 |
| 0 0 3 7 | 0 0 | 1 5 | E15 |
| 0 0 3 8 | 0 0 | 1 2 | E12 |

## PROGRAMMING GROUP KEY CODES

| | | | |
|---|---|---|---|
| 0 0 0 0 | 0 8 | 1 4 | * LP |
| 0 0 0 1 | 0 8 | 0 0 | * S |
| 0 0 0 2 | 0 8 | 0 3 | * $G_0$ |
| 0 0 0 3 | 0 8 | 0 4 | * $J_0$ |
| 0 0 0 4 | 0 8 | 0 5 | * $J_+$ |
| 0 0 0 5 | 0 8 | 0 1 | * RL |
| 0 0 0 6 | 0 8 | 0 2 | * W |
| 0 0 0 7 | 0 9 | 0 3 | * SP |
| 0 0 0 8 | 0 9 | 0 0 | * M |
| 0 0 0 9 | 0 9 | 1 4 | * EP |
| 0 0 1 0 | 0 9 | 0 4 | * $J_\emptyset$ |
| 0 0 1 1 | 0 9 | 0 5 | * $J_E$ |
| 0 0 1 2 | 0 9 | 0 1 | * ST |
| 0 0 1 3 | 0 9 | 0 2 | * $\alpha$ |

facsimile of page 30 - original page illegible

Having looked through the preceding pages, you have probably noticed that the codes generated by the regular keyboard keys can be duplicated by some combination of the special function keys and the register keys.

Later, in the sections on printing and programming, you will find that the sequence of key strokes will also determine what is to be printed and how it is to be printed.

Other Switches

1. fl-sc: In the up position, the machine will give and receive numbers in standard decimal form and the answer will be displayed in floating decimal up to ten digits. Beyond that, it will always automatically go to scientific notation.

When the switch is depressed, the machine will always give and receive numbers in scientific form. For example:

Key 4567.4
    STORE
The machine will store $4.5674 \times 10^3$
    4.567400000 +03

All answers and operations will be handled in standard scientific notation.

2. Rad ÷ Deg: When handling trigometric operations, the machine will interpret the display in degrees when the switch is up and in radians when the switch is down. You should interpret any answers given by the machine in like manner.

NOTE: The switch position is not programmable.

3. Printer on: Depressing the printer on switch activates

the printer drum and readies the machine for printing.

4. Paper feed: This advances the paper by one printed row for each stroke of the key.

The {fl-sc, Rad - Deg, and printer on} keys are all two touch keys. That is, the first touch engages, the second touch releases them. All ohters are released when another key is depressed {similar to a car radio}.

Substorage Access

1. The Wang 500 provides, in addition to the sixteen upper storage registers, forty storage registers in its memory circuits which can be directly addressed for simple storage. These registers will be referred to as substorage registers. They can be used for storage by keying a particular sequence of keys for each register. Each storage register has an address just as you have a home address. When you wish to store a number into or take a number out of the registers, you must use this address. The address is composed of a command and a location.

2. The Command

Notice in the green programming keys a RECALL-STORE KEY. This is the substorage command key. It tells the machine you are going to put a number in {STORE} or take it out {RECALL}. To recall, simply touch the key. To store, key Shift/Recall {Store}, This procedure has used one step of programming as you will later discover. Then key the address. The address is composed of two key strokes: a special function switch and a register key. The addresses for the registers are as follows:

| SELECTION SWITCH | REGISTER KEY | | REG NO. |
|---|---|---|---|
| T | 00 | addresses | 16 |
| T | 01 | " | 17 |
| T | 02 | " | 18 |
| | ↓ | | ↓ |
| T | 15 | " | 31 |
| + | 00 | " | 32 |
| + | 01 | " | 33 |
| + | 02 | " | 34 |
| | ↓ | | ↓ |
| + | 15 | " | 47 |
| - | 00 | " | 48 |
| - | 01 | " | 49 |
| | ↓ | | ↓ |
| - | 07 | " | 55 |

EXAMPLE: Store the number 3574 in Reg. 15.

Key the number
Shift Store {Green key}
[T] Selector switch
[00] Register key

The number is now located in the register for recall any time it is needed. To bring the number back, you touch Recall {green key} and the address T{00} and the number will be brought to the display.

Alpha Codes

There are two other codes that can be used directly from the keyboard without programming that allow some flexibility of operation.

Decimal Shifting: The decimal point of any number can be moved left and right on the machine without affecting the sequence of digits displayed.

With the f {x} switch down:

Shift → Alpha → $\boxed{\text{Reg}}$ # tells the machine to shift decimal right

Shift → Alpha → shift → $\boxed{\text{Reg}}$ # tells the machine to shift left.

The register # selected tells how many digist to shift.

Calling π : The number π {Pi} may be called up out of the machine's memory by keying {shift → Alpha → $\boxed{00}$ }.

These operations may be used at your discretion when needed.

SUMMARY OF ALPHA OPERATIONS

Shifting Decimal Point to the Right or Left

| Shifting Right | | Shifting Left | |
|---|---|---|---|
| f{x} down<br>SHIFT<br>α<br>01 | 1 place | f{x} down<br>SHIFT<br>α<br>SHIFT<br>01 | 1 place |
| SHIFT<br>α<br>02 | 2 places | SHIFT<br>α<br>SHIFT<br>03 | 2 places |
| SHIFT<br>α<br>03<br>.<br>. | 3 places | SHIFT<br>α<br>SHIFT<br>03<br>. | 3 places |
| SHIFT<br>15 | 15 places | SHIFT<br>SHIFT<br>15 | 15 places |

Calling π

| | |
|---|---|
| (SHIFT)<br>α<br>00 | Brings π to the display |

Alpha commands are normally two step procedures and simply alert the machine for the next code to be handled differently than normal.

## IV. MATH GROUP

A. Use of Shift Key

B. Trig Keys

C. Log Keys

D. Rad-Deg

E. Reciprocal

F. $10^x$

G. $x^2$

H. $\sqrt{x}$ ? $|x|$

I. INT. x

Rad-Deg

By using the Rad to Deg key, the calculator will interpret whatever is on the display as degrees of rotation and convert it to radians of rotation. With the use of the shift key, you will generate the second function of Deg to Rad. The calculator will now interpret the display as radians and convert it to degree measurement. These keys are used in conjunction with a second Deg to Rad switch. When the switch is up, all numbers are interpreted as degrees; when the switch is down, all values are interpreted as radians.

Reciprocal

Use of this key will take the number on the display and divide it into 1 (reciprocal).

$10^x$

To generate this function, you use the second function of the $e^x$ key. To do this, you press the shift key and then $e^x$. This will raise 10 to the power of the number on the display.

$x^2$

The $x^2$ key takes the number on the display and multiplies it by itself or squares it.

$\sqrt{x}$, $|x|$

The $\sqrt{x}$ key takes the displayed number and finds the square root. The second function of this key is $|x|$ which takes the absolute value of the number on the display (makes all numbers positive}.

INT x

The INT x function is the second of the $x^2$ key. This key truncates or subtracts the decimal part of the number on the display. Following is a brief program to demonstrate one use of this key.

```
SHIFT MARK      This program will convert a value of degrees
   0            with its decimal counterpart into Degrees,
Re 00           Minutes and Seconds. It will also round off
ST L            the seconds' value using .5 as the base for
SHIFT INT x     comparison.
PRINT
- 10   ---->    {-switch, and register 10}
                {labels A}    {scientific notation}
- L
60
XL
SHIFT INT x
PRINT
x 10            {labels B}
- L
60
x L
SHIFT INT x
ST R
- L
- 1
x L
.5
+ L
J if +
1               If decimal is greater than .5, the seconds'
+ R             value will be increased by 1.
Re R
PRINT
÷ 10            {labels c, in scientific notation}
SHIFT END
```

For practice you should attempt the problems for this section in the supplement. Try writing out the key strokes for each problem prior to doing them on the machine.

## V. PRINTER CAPABILITIES

A. Introduction

1. Switches {On, Paper Feed}

2. Key -- {Print}

3. Mode {Learn Print} {List Program}

B. Automatic Printing

C. Manual Decimal Setting

D. Trace Capability

1. Program trace

2. Manual trace

# PRINTER CAPABILITIES

## Introduction

The Wang 500 is supplied with a rotating drum line printer which is very useful in several ways. There are basically four switches and one key used in connection with the operation of the drum printer.

1. PRINT-ON-SWITCH: This switch must be in the depressed position for the printer to operate in any manner.

PAPER FEED-SWITCH: Advances the paper by one line for each stroke of the switch.

2. PRINT, KEY: Causes whatever is in the display window to be printed on the paper tape.

3. LEARN-PRINT MODE SWITCH: When this switch is depressed, the calculator will automatically make a hard copy listing of the programming steps being entered by the programmer as he enters them.

LIST PROGRAM-MODE SWITCH: When this switch is depressed and the $\boxed{\text{Go}}$ is keyed, the machine will list out the program steps in its memory from the present pc position in groups of 100. This may be stopped at any time by touching the step key. This feature is very useful in debugging and listing programs for record keeping.

## Automatic Printing

You may instruct the calculator to print automatically all functional operations being processed. The instructions are as follows:

| auto print "on" | auto print "off" |
|---|---|
| shift | shift |
| α | α |
| print | shift |
| | α |

Two different printing operations can occur in this mode: single or double cycle printing. For addition, subtraction, storing, recalling and total, only a single entry will be printed: either the number being entered or the number being extracted. A running total will not be entered. For all other operations, a double entry is printed: the number and the operation being performed and the result of the operation.

Manual Decimal Setting {Manual Control}

If a specific decimal setting is desired for manual print operations, the unit can be instructed to print in floating decimal, scientific notation, or up to nine designated places. THE DECIMAL POINT SETTING FOR MANUAL PRINTING ALSO APPLIES FOR AUTOMATIC PRINTING AS WELL.

To condition the unit to read and print the desired decimal setting:

Set PC
For the 500, index a 3 digit number
For the 600, index a 4 digit number
The last digit determines the decimal setting
  0 = floating
  1 = scientific
  2 - 9 = number of digits following decimal

The instructions from this point on apply only to the 600 calculator.

Trace Capability

1. If a programmer could open his calculator and watch his program being executed step by step, he would be able to correct his programming problems in half the time. Since he could see an error when it first occured, most of the problem of fixing a programming error would be eliminated. While it is not practical with today's high-speed electronic machines, the Wang 600 offers the next best thing--a step-by-step printout of each program operation showing all intermediate results. The program logic becomes clear as well since all MARK SEARCH and Subroutine branchings are shown as they occur in the normal flow of the program. This allows the programmer to make on-the-spot changes and modification in a program.

The method of initiating a Program Step Trace is by means of a two-step ALPHA command, which programs control of the Column Printer. The two- step ALPHA {$\alpha$} $\text{Log}_e X$ {or $\text{Log}_{10} X$} signals the TRACE-ON, while ALPHA {$\alpha$} $e^x$ {or $10^x$} turns the Program TRACE-OFF. Output of the Printer includes a printout of the window contents, and the program symbol representing the program step being executed at the time of the printout.

2. The Step Trace feature can be initiated from the keyboard as well as by a program. Suppose you had a program in memory which you wanted to Step Trace through its entire operation. All you would have to do is turn the Trace Mode

ON and then key the normal operating instructions. Depress the "PRINTER ON" Switch,

1. Key PRIME
2. Key ALPHA {α}, $Log_e X$ {or $Log_{10} X$}--{turns TRACE ON}
3. Key in the data needed by the program {if the normal program instructions require it}, and key GO.

RESULTS: The Printer will trace the entire program. If a "TRACE OFF" Command {ALPHA {α} $e^x$ or ALPHA {α} $10^x$} is found within the program however, the Trace will end at that point.

## VI. INTRODUCTION TO PROGRAMMING

A. Definition of and Reasons for Programming

B. Developing Programs

1. Common features

2. Flow diagrams

3. Saving steps

C. Utilizing a Program

D. Debugging

E. Record Keeping

# INTRODUCTION TO PROGRAMMING

## Definition of and Reasons for Programming

So far, we have been using the Wang as a calculator. However, most of the Wang's usefulness lies in its ability to be programmed. This allows the Wang to calculate automatically by utilizing a program stored in its memory, making the Wang more like a computer.

The program consists of a complete set of instructions which enable the calculator to process the values given it to obtain the desired results. The program can eliminate all the time-consuming and repetitive keystrokes that would be performed manually for each set of keystrokes.

Thus programs:

1. Save time
2. Save work
3. Add to the versatility of the machine

## Developing Programs

1. In every program, there are certain features that are common.

The first item in a program must be some sort of identification. Just like defferent telephones ring when different phone numbers are dialed, each program has a different "number" called a fug. These identifications are necessary in more complex situations where several programs are stored simultaneously in the memory. On the Wang, any programmable keyboard function can serve as an identification. To distinguish the function as an identification, it is preceded by the "Mark" command which "marks" the program. This is sometimes referred to as the index. The last item in a program is the "End Program" command. This tells the Wang to stop calculating. This must absolutely be the final step after all calculations are made or the Wang will stop before completion of the program.

The program itself consists of the instructions given the Wang to do the necessary calculations. These are the same commands you would use if the calculations were being performed manually. Programs should be versatile enough so any values can be plugged in. Therefore a register may be recalled in place of a value (unless it is a constant in the formula). The register then contains the value of the number and can be readily changed.

Let's consider the example:

$$A = \pi r^2$$

This formula is used to find the area of a circle. The variable would be "r" in this example. If we assign it a register (for instance 00), we can give it any value. The value of "$\pi$" itself should be inserted in the program since it is a constant.

The program needs a mark flag, in the example "1." So the sequence of instructions given the Wang would be:

| INSTRUCTION | OPERATION |
|---|---|
| MARK 1 | |
| RECALL 00 | {radius} |
| $x^2$ | $r^2$ |
| ST L | $r^2$ in left register |
| 3.14 | $\pi$ |
| x L | $\pi r^2$ |
| END PROGRAM | $\pi r^2$ {ANSWER ON DISPLAY} |

This is essentially the program to find the area of a circle. Although it may be simple, even the most complex programs consist of smaller programs like these.

The program must not contain any illegal maneuvers or directions, such as dividing by zero or finding the tangent of 90 degrees, or any other undefined functions.

These are the basic rules to programming:

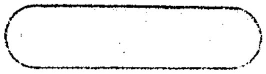
1. Every program must start with a Mark "x" command.
2. Every program must have an End Program command.
3. The program can be used for any set of values.
4. The program should not produce any undefined values.

2. To write a program, it is necessary to plan the sequence of operations that the calculator is to perform. The relationships between the operations may become complex and difficult to keep clearly in mind. For this reason, programmers usually sketch out an outline of the program. The outline, called a flow diagram, gives the programmer a visual representation of the relationships involved.

A set of symbols are used to show the computer operations, and lines and arrows are used to show the flow of

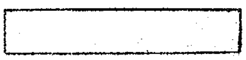
Logic between them.

The following symbols are used in a flow diagram.



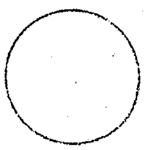
An oval box indicates a starting or stopping operation.

Arrows indicate the direction of flow through the diagram. Every connecting line should have an arrow on it.



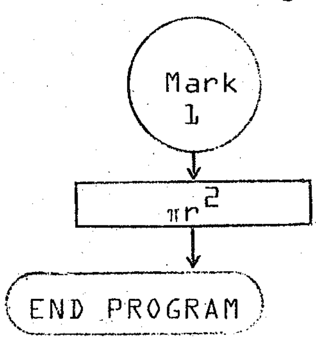
A rectangular box indicates any processing operation except a decision.

A diamond indicates a decision. The lines leaving the decision box indicate the possible paths the program can follow.



A large circle indicates a connection in a program where a number may be indexed.

An example of the use of these symbols follows. This is the flow diagram for the example used earlier {$A = \pi r^2$}




This is the "mark" or index of the program.

This is the body of the program which does the processing.

This is the "End Program" command.

3. Although this flow chart shows a simple program, a flow chart could be invaluable in a more complex program. The program can be mapped out to show the flow through it, and steps can be saved by determining the most effecient way to design the program. These methods will be discussed in the sections concerning looping and subroutines.

Utilizing a Program

When doing the actual programming, it is important to remember that the process is the same as it would be if it were being done under manual control.

1. Depress the learn mode button.
2. Set the Program Counter at the step where the first program command will be stored. In most cases, it will be at step 000 if no other programs are in memory. The program counter is set by depressing the SET PC key followed by a three digit number indexed by using the Data Entry keys. If you want to start at step 000, depress the Prime key. This sets the P.C. at 000.
3. Introduce the program via the sequence of key-strokes the program requires. Do not forget the "Mark" command or the "End Program" command.
4. The program is now stored in the memory and can be executed by switching to the Run Mode.
5. Any values required should be stored in the proper registers.
6. To run the program, depress the Search command followed by the mark "Flag" at the beginning of the program.

Debugging

If it is necessary to alter or "debug" a program, a listing of the program stored in the memory can be retreived by pressing the list program switch and keying the go button.

By putting the Wang in Learn mode and using the step key, the display will show the same as the paper printout, minus the alphanumeric translation which is found on the right side of the paper.

| 000<br>Step No. | 00 00<br>Function being<br>Processed | ☐<br>Alphanumeric<br>Translation |
|---|---|---|
| DISPLAY ONLY | | PAPER ONLY |

The listing or display will show a set of codes which tell what the program does. The step no. is indicated, and then the two-part code. A set of pull-out cards on the bottom of the calculator tell what the codes mean. The listing can be compared with the original program to see if there are any discrepancies.

With the machine in Run mode, the flow of values through the program can be monitored by using the step key. If the programmer has an idea of what the values should be at each particular step, he can determine where his error is.

If a step needs to be changed, the Wang should be put in learn mode. Set the program counter to the step to be cahnged and key in the new step. Steps may be deleted on the Wang 500 by replacing them with GO commands. Extra steps may be inserted or deleted on the Wang 600.

Record Keeping

It is usually desirable to keep a record of a program once it is written in case it is necessary to use the program in the future. Forms such as those of the following pages are ideal since they are set up to facilitate programming on the Wang.

Record keeping and methods of storing programs will be covered in greater detail in later sections.

# VII. SUBROUTINES

A. Reason for Subroutines

B. Use within Mainlines

C. Subroutine Codes

D. Subroutine Considerations

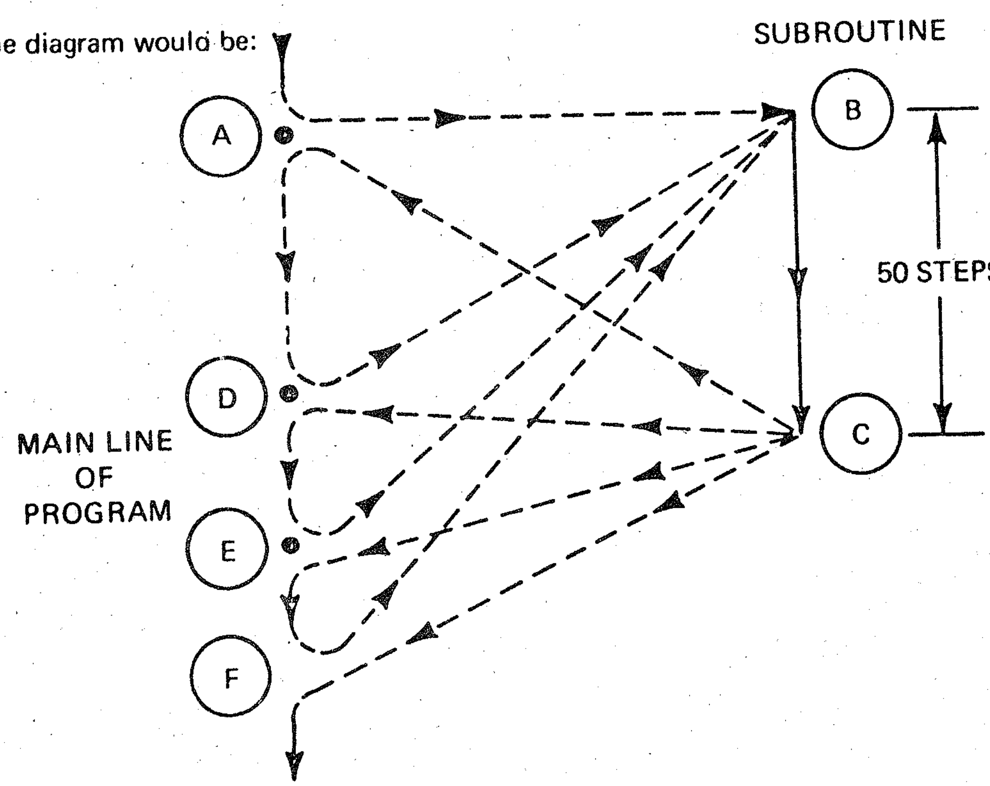
# SUBROUTINES

Subroutines are programs within a main program. Many times a program will contain many subroutines. The subroutines are called upon by the main program, whenever necessary, to perform a series of calculations. After these calculations are performed, the subroutine then returns control to the main program.

## Reasons for Subroutines

Let's say you have a program that uses a fifty step calculation four times. Instead of writing this program four times, and using 200 program steps, it is written as a subroutine just once, taking fifty program steps, and is called upon by the main program when needed as shown in the example below.

The diagram would be:



## Use within Mainlines

How do you write subroutines in a program? Let's say you have a program and at Step 10 you want to go to a subroutine, do the calculations, and return to your mainline. It would look like this:

| Step | Command | Code | Step | Command | Code |
|---|---|---|---|---|---|
| 0009 | | | 0040 | Mark | 0090 |
| 0010 | Search 1013 | 1013 | 0041 | 1013 | 1013 |
| 0011 | | | 0042 | . | . |
| | | | 0043 | . | . |
| | | | 0044 | . | . |
| | | | 0045 | Return | 0915 |

The command in the mainline used to search for the subroutine is the subroutine command itself. The machine searches for the subroutine f{13}, represented by a MARK command and then the subroutine command. When the machine finds the subroutine, it runs through it and then returns to the mainline program by means of a RETURN command. You may use subroutines within subroutines up to five deep.

Most subroutines are written in memory prior to the mainline program.

## Subroutine Coding

Most programs you write will use no more than a few subroutines. However, to assure an adequate number, almost every key on the Keyboard can be used to address a subroutine. The basic set of thirty-two subroutine codes is given on the following page.

| CODE | SYMBOL | CODE | SYMBOL |
|---|---|---|---|
| 10 00 | f(00) | 11 00 | F(00) |
| 10 01 | f(01) | 11 01 | F(01) |
| 10 02 | f(02) | 11 02 | F(02) |
| 10 03 | f(03) | 11 03 | F(03) |
| 10 04 | f(04) | 11 04 | F(04) |
| 10 05 | f(05) | 11 05 | F(05) |
| 10 06 | f(06) | 11 06 | F(06) |
| 10 07 | f(07) | 11 07 | F(07) |
| 10 08 | f(08) | 11 08 | F(08) |
| 10 09 | f(09) | 11 09 | F(09) |
| 10 10 | f(10) | 11 10 | F(10) |
| 10 11 | f(11) | 11 11 | F(11) |
| 10 12 | f(12) | 11 12 | F(12) |
| 10 13 | f(13) | 11 13 | F(13) |
| 10 14 | f(14) | 11 14 | F(14) |
| 10 15 | f(15) | 11 15 | F(15) |

To generate the subroutine codes, the f{x} switch must by down and the register key is pushed for the subroutine number. For example, $f_x$ down and $\boxed{04}$ will generate 10 04. Shift, $f_x$ down and $\boxed{04}$ will generate 11 04.

Subroutine Considerations

When using subroutines, remember that their main purpose is to reduce the number of steps required to accomplish a particular operation. If this is not accomplished, the program would be better done without the subroutine.

When using various subroutine levels, be very careful that the requirements of the program do not cause the subroutines to exceed five levels. If this happens, the calculator will not be able to find and return to its previous position. The machine is only capable of remembering five prior step positions and would therefore lose track of its "train of thought."

## VIII. PROGRAMMED PRINTING

A. When to Print

B. How to Print

C. Labeling

D. Paper Feeding

# PROGRAMMED PRINTING

## When to Print

Printing instructions in a program can be used to obtain a permanent record of a problem and to save time if the equation or program has multiple answers. In addition to printing the answers of a program, you may want to have the program include instructions to print the values in the input registers. This will give you a record of the variables used as well as the results.

If you want your calculated number to be printed in a specific order other than the order calculated, you can store the numbers in registers and recall them in the order desired. This can be a help if you are using a looping program which will be used for graphing.

For programmed printing, you must also remember to manually turn the printer on. (Always remember to turn printer off before turning main calculator off.)

## How to Print

Programmed printing involves two steps of program and will print the number on the display previous to the print command. The first step is the actual print command key producing the code 0802. The second tells the calculator how to label the previous number (using high order codes) and also how to print it numerically. This means to print it in scientific notation, floating, or with fixed decimal point (using low order codes). If you forget to include the second step of the printing procedure, the calculator will interpret the next program step as the code for how to print and cause program error. The chart on what keys to use for labeling and printing the numbers numerically can be found on the following page.

Labeling

When you label in a program, some thought should be given to what letter will best represent the answer. An example would be to label a current by using I which is denoted by the high order code 11 and generated by using the {SP} and {-} switches in the down position.

Paper Feeding

During your program, you may want to separate groups of printed answers for easier identification. This is accomplished by using a pre-programmed paper feed step. This procedure contains two program steps the same as any other printing routine. The first is the print command and the second is register 15. The switch positions do not matter in this case.

## PRINTING CODES

All command preceded by a print command
Require two additional strokes, a switch (High Order) and a register (low order).

| High Order Code | Switch Position | Meaning | Low Order Code | Key Reg | Meaning |
|---|---|---|---|---|---|
| 00 | all up | Label X | 00 | 00 | 0 decimal |
| 01 | T | Label Y | 01 | 01 | 1 decimal |
| 02 | + | " Z | 02 | 02 | 2 " |
| 03 | - | " A | 03 | 03 | 3 " |
| 04 | x | " B | 04 | 04 | 4 " |
| 05 | ÷ | " C | 05 | 05 | 5 " |
| 06 | ST | " D | 06 | 06 | 6 " |
| 07 | Re | " E | 07 | 07 | 7 " |
| 08 | (SP down) | " F | 08 | 08 | 8 " |
| 09 | SP T | " G | 09 | 09 | 9 " |
| 10 | SP + | " H | 10 | 10 | 10 floating |
| 11 | SP - | " I | 11 | 11 | 11 scientific |
| 12 | SP x | " J | | | |
| 13 | SP ÷ | " K | | | |
| 14 | SP ST | " L | | | |
| 15 | SP Re | " M | 15 | 15 | 15 paper feed one line |

## IX. LOOPS AND DECISION KEYS

A. Decision Key Functions

B. Use of Loops

C. Incrimenting Routines

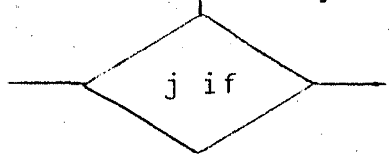
# LOOPS AND DECISION KEYS

## Decision Key Functions

The Wang is capable of marking 4 {four} decisions {Display Recognitions}:

Jump if +
Jump if 0
Jump if error {shift function}
Jump if ≠ 0 {shift function}

These "decision" keys compare what is on the display to a preset condition. If that condition is met, the calculator skips the next two steps of programming and carries out the third instructional step after the jump command. If the condition is not met, the calculator continues as it normally would. The flow chart symbol for a decision is




The actual operation that occurs is as follows:

| Step No. | ↓ | |
|---|---|---|
| 010 | -L | |
| 011 | j if 0 ---> | compare display |
| 012 | ☒ | {instructions for con- |
| 013 | ☒ ---> | ditions not being met} |
| 014 | Print ---> | continue instructions if the conditions are met |
| | ↓ | |

The two steps that are used when the condition is not met are usually used to create a loop or direct the calculator to a different program or subroutine.

A loop is created when you tell the program to search for its won mark code. You may direct the machine to another completely separate program by "Search" "Program Code" which requires two steps. You may also call for a subroutine {1 step} and the machine will branch to the subroutine and come come back as has been previously discussed. Because only one program step is used to call for a subroutine, you must fill the leftover step with a GO command.

The two steps may also be used in any manner appropriate to the program being written.

<u>Use of Loops</u>

A loop differs from a subroutine in that a loop sets up a repeating or cycling operation that continues until the condition desired is met while a subroutine is usually a completely separate set of operations. Since subroutines have already been discussed, the following is an example of setting up a looping routine.

Step No.

```
000 ------------ Mark
001 ------------ 1
001 ------------ E2
003 ------------ +L
004 ------------ ST Rt
005 ------------ α      ½ sec
006 ------------ STOP   pause
007 ------------ -Re 00
008 ------------ -Rt
009 ------------┌j if +
010 ------------│Search
011 ------------│1
012 ------------└EP
```

This program will count by 2's form the beginning value {stored in the left register} until it exceeds the ending value {stored in 00} at which time it will jump to the End Program.

<u>Incrimenting Routines</u>

The example above is a form of incrimenting routine. An incrimenting routine is a program loop that counts up or down by a value that is predetermined by the program. The count being generated can then be used in another program.

EXAMPLE: Incriment from 10,000 to 100,000 by 500's and use each incriment in a subroutine $\{f_1\}$. Store the first value in $\boxed{00}$, store the last value in $\boxed{01}$, store incriment in $\boxed{02}$.

```
Mark
 1
Re00
{f1}
Re02
+00
STL
Re01
-L
j if +
Search
 1
Re01
{f1}
EP
```

This method of incrimenting may be used or any number of other possible forms can be used by the programmer to accomplish just the conditions he wishes to create.

Try some of the problems in the supplement that apply to this section.

## X. PROGRAM STORAGE

A. Cards

1. Programming

2. Loading

B. Tape

1. Recording

2. Loading

C. Bootstrap Routine

# PROGRAM STORAGE

## Cards

1. The card input is another way of inputing the program codes into the calculator.

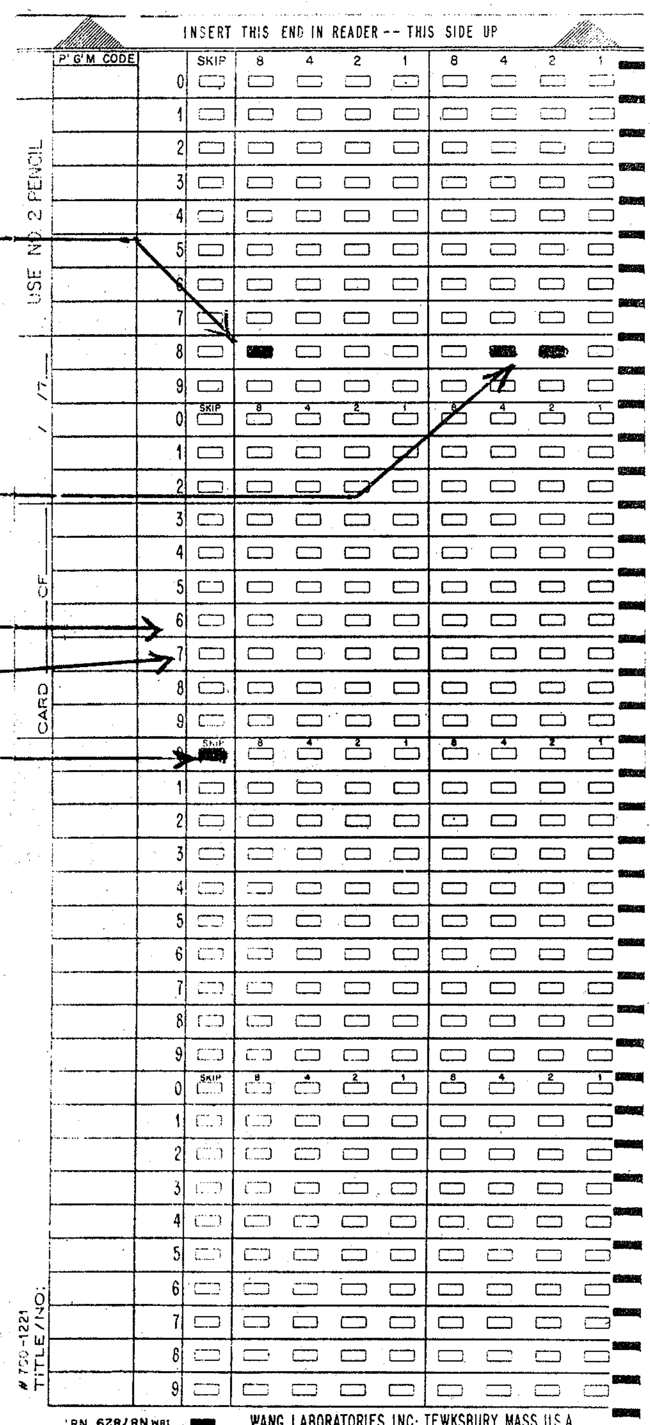
On your cards, you have your high order and low order codes. To program the cards, you simply color in the appropriate squares. For example:

To enter code for SIN{0806} you first color in the box labeled 8, the high order part {08xx}.

Then you color in the 4 box and the 2 box, the sum of which is the low order code {xx06}.

Try to fill in this code. Store reg. 03 {0603} and recall reg. 15 {0715}.

The skip box is there so that if, for any reason, you wish to skip that line, just fill in the box.



2. To enter the card into the calculator make the necessary attachments.

1. PRIME
2. DEPRESS LEARN MODE
3. INSERT CARD INTO CARD READER (pencil marks up, arrows first)

The card will automatically be pulled through the card reader. The calculator display will now show that it has read forty steps (minus skip marks). If it does not read this, PRIME and start over.

If two cards are entered consecutively, the calculator display will read that it is on step 80. You can keep doing this until you run out of memory. In most cases, for the Wang 500's, it takes 312 steps.

Program storage on cards of this type is especially good for numerous not too lengthly programs.

Tape

1. The Wang also has the capability of storing programs on tape. Let's suppose you have a program in memory and you wish to store it on tape. Follow this procedure:

1. Place calculator into RUN MODE.
2. Insert tape cassette.
3. Rewind tape.
4. Engage TAPE READY switch.
5. a. If you wish to start recording from step 000,

depress PRIME and key RECORD PROGRAM.*

5. b. If you wish to start recording from a step, i.e. 138, simply set P.C. {program counter} to step 138 and key RECORD program.*

2. How to load program from tape into memory

1. Place calculator into RUN MODE.

2. Insert tape and rewind.

3. Depress the tape ready switch.

4. a. If you wish to load from step 000 and on

   Key PRIME

   Key LOAD PROGRAM

   b. If you wish to load starting at some other step in memory

   Set PC {to that step}

   Key LOAD PROGRAM

Bootstrap Routine

A Bootstrap routine is a program which lets you choose and load any one program from a tape. All you do is enter the number of its position on the tape into the calculator.

Following is one way to do this.

The first block of your tape must be the index and contain these steps:

0000 ST13

---

*There must be an END PROGRAM {EP} command at the end of your program for the calculator will record up to and includeing the EP and STOP.

```
0001    Mark
0002     0
0003    LP
0004    EP
```

All other program blocks must have these steps preceeding the actual program itself:

```
Steps for      Steps for      Steps for      Steps for
1st block      2nd block      3rd block      4th block

1              2              3              4
stl            stl            stl            stl
Re 13          Re 13          Re 13          Re 13
-L             -L             -L             -L
j if +         j if +         j if +         j if +
search         search         search         search
0              0              0              0
*****************ACTUAL PROGRAM**********************
EP             EP             EP             EP
```

These seven steps do the comparing between the numbers you put into the calculator and the actual numbers of the program block.

Let's say that you have loaded the Index block into the calculator. The memory now looks like this:

```
0000    ST13
0001    Mark
0002     0
0003    LP
0004    EP
```

And now you wish to load the second program. With the Wang in run mode, enter 2 and key GO. Calculation will start at step 0000 and continue. When the load program command is encountered, the tape will start running and load in the next block program. These steps will be loaded from the step after the load program command and on. So the memory now looks like this:

```
0   st13
1   M
2   0
3   LP
4   1
5   STL
6   Re13
    -L
    j if +
    s
    0
    ∿
    ∿
    ∿
    EP
```

When the tape drive stops, the calculation will resume operation from step 0004. The next Bootstrap step will compare the number that you entered with the number of that block.

In this case, the number that we entered was 2 and the number of the block is 1. The math compares these numbers {1 - 2 = -1}. When the jump if positive {j if +} command is encountered, the program will not jump because the answer is negative. Instead, it will do the next two steps and search for the 0 mark.

Now we are back to step 0001. When the load program command is encountered again, the next program block will be loaded in. Then again, the number we entered will be compared to the block number {this time, 2}.

After the comparison {2 - 2 = 0} the j if + command will jump the program over the Search 0 command and execute the rest of your program.

## XI. TAPE FEATURES

- A. Introduction
- B. Program Transfers
  - 1. Record program
  - 2. Load program
  - 3. Alpha loading
    - a. Loading
    - b. Tape positioning
- C. Data Transfer
  - 1. Recording data
  - 2. Loading data

# TAPE FEATURES

## Introduction

The Wang 500 and 600 are both supplied with cassette tape drive mechanisms. These greatly enhance the versatility of the machines. Programs can be "chained" or sequenced into the machine allowing for the development of programs which exceed the capability of the machine in terms of program steps.

The "Bootstrapping" technique used on the Wang 500 has already been discussed and will not be presented again here. It should be stated, however, that the same technique may be used with the Wang 600 as well.

## Program Transfers

1. Recording a program from memory to tape and vice-versa are quite simple and very straight forward.

Record Program

a) Place in run mode
b) Rewind tape
c) Tape ready
d) If you wish to start recording from step 0000, prime the machine. If, however, the block of core you wish to record starts at some position other than 0000, you should set P.C. to the desired step before recording.
e) Key record program (orange key). All steps of programming from the initial setting position of the PC up to and including the END PROG command will be recorded on tape.

2. Load Program

a) Place in run mode
b) Rewind tape
c) Tape ready
d) Set PC to the step where you wish to start

loading into memory. If you wish to start at 0000, "prime." If the tape block you wish to load is locate on tape at, for example, the third position on the tape, you would depress the load program three times as follows.

1} Set PC to starting step
2} Load program {When tape stops, the first tape block has been loaded and the program counter has been reset to the initial setting.}
3} Load program again--Second block is loaded and PC is reset.
4} Load program one last time and the third block is now loaded into the memory starting at the step you specified.

This method of loading may be used either on the 500 or 600. However, the 600 has a tape feature which eliminates this somewhat time-consuming procedure. It is called alpha loading.

3. Alpha Loading {tape block search and load} lets you automatically skip over a designated number of blocks of programs on tape and load the desired block without following the procedure stated above.

a. Suppose the same situation exists as was previously set up. That is, you have a program written in the third position on a tape and you would like this to be loaded into memory starting at step 257. Proceed as follows:

a} Place in run mode
b} Rewind tape
c} Tape ready
d} Set P.C. at 0257
e} Key 2. {The number of blocks you wish to skip over}
f} Key {shift, alpha, load program}

The first two tape blocks will be skipped over and the third block will be loaded into memory starting at step 257 and continuing until an end program is encountered.

This procedure for loading memory from tape may be programmed and used for sequential loading and chaining and save a tremendous amount of time spent searching for a particular program.

b. Another technique using alpha load which you will find very convenient is one in which you position the tape for recording a program that has been written in memory.

Suppose you have written a program into memory that you wish to put on tape in the fifth block position:

a) Place in run mode
b) Rewind tape
c) Tape ready
d) Set P.C. to last program step (0823)
e) Key 3
f) Key (shift, alpha, load)
The calculator will skip the first three blocks on tape and attempt to load the fourth program into the last 8 program steps (which it cannot do). This causes a program error light (Tape stops at E.P. of fourth block).
g) Clear the error (prime).
h) Set PC to the beginning position of the program to be recorded.
i) Key record program.

This results in the program in memory being loaded into position 5 on the tape.

Data Transfer

Data that is stored in memory may be transferred to tape for permanent or temporary storage. Also, data that has been stored on tape may be transferred into memory registers using the alpha store and alpha recall commands. Alpha store pulls data out of memory registers and stores it on tape; alpha recall pulls data off the tape and puts it into the memory registers.

1. Suppose you had data in registers 00 through 26 that was to be used over and over in several programs. It would be very convenient to have this data on tape rather than having to load it manually for each program.

a) Place in run mode.
b) Rewind tape
c) Tape ready
d) Prime
e) Key the last register you wish to record (in this case, 26)
f) Key (alpha, store).

When the tape stops, the contents of the registers will have been transferred to tape in reverse order, that is, 26 first and 00 last.

2. Any time you wish to recall this data back into memory from tape, you follow the same procedure except that you use (alpha, recall) instead of alpha, store. That is:

a) Place in run mode.
b) Rewind tape
c) Tape ready
d) Prime
e) Key first register to be loaded (26)
f) Alpha, recall

The data will be transferred into memory beginning with register 26 and continuing until all data has been loaded.

NOTE: Alpha store may be used to store as many registers as you wish. However, once stored on tape, the entire block must be recalled. If this is not done, you will receive a program error indication.

## XII. INTRODUCTION TO THE PLOTTER

A. Switches and Controls

B. Bed Layout

C. Scaling

D. Operation

1. Plotting

2. Printing

3. I/O Writing

# INTRODUCTION TO THE PLOTTER

## Switches and Controls

The Model 612 Flatbed Plotter is connected to the Wang 600 calculator by connecting the input cable to the typewriter output receptacle. There are several switches and controls on the front of the plotter. They are as follows:

Controls {zero reference}

| | | |
|---|---|---|
| Check | {Push button} | Brings pen to Zero position |
| X | {Knob} | Controls Horizontal position |
| Y | {Knob} | Controls Vertical position |
| {Scale adj} | | |
| Check | {Push button} | Moves pen to upper right position |
| X | {Knob} | Controls Horizontal position |
| Y | {Knob} | Controls Vertical position |

Switches

| | |
|---|---|
| Power | on - off |
| Select | Plotter - typewriter |
| Chart | Hold - release {Hold paper} |
| Pen | Up - locks pen in up position |
| | Down - allows calculator control |

## Bed Layout

The plotting surface of the plotter has a maximum chart capability of 10" x 15" paper. Any smaller size paper may be used and the scale may be adjusted to it. Whatever the size paper scaled for, the area scaled will be automatically divided into 999 vertical incriments and 999 horizontal units. Once the plotting area is scaled, all plotting and alphanumeric writing are done by incriments and do not depend upon actual size of area scaled. You should also remember that scaling adjustments affect the size of the area being scaled while zero reference adjustments move the entire area to new positions. {The plotting proportions do not change.}

## Scaling

Scaling the area to be plotted is done as follows:

1. Press zero reference check
2. Adjust X and Y to {lower left} zero point
3. Press scaling check
4. Adjust X and Y to extreme upper right point.

Area of plot is now scaled.

Remember: To change size, make scaling adjustments.
To change position, make zero reference adjustments.

Condensed plotter instructions:

Turn plotter on -- load paper
Set reference point
Set scaling point

NOTE: The scaled area will always be divided into 999 parts for both X and Y axes no matter what the actual size in inches.

Maximum scale size is 10" x 15"

One unit ≈ 1/999 of axis size.
Ex: 5" horizontal scale
one horizontal unit ≈ .005"

Once the area is scaled, moving the reference point moves the whole area.
The proportions do not change.

## Operation

There are basically three ways in which the plotter may operate:

1. Plotting mode

2. Alphanumeric printing

3. I/O printing

1. All plotting commands are generated using a combination

of selector switches {high order} and storage registers {low order}.

There are basically six plotting commands:

| | Code | Switch | Register |
|---|---|---|---|
| Plot | 0502 | : | 02 |
| Advance {without plot} | 0503 | : | 03 |
| Return {to zero} | 0511 | : | 11 |
| Pen up | 0103 | T | 03 |
| Pen down | 0102 | T | 02 |
| Move {1 incriment} | 0402 | X | 02 |

All of these command codes are preceded by an α code and followed by an end alpha command {α - 0902} {end α - 0202}.

When plotting or advancing the incriment amounts are stored prior to the command.

X amount stored in 00 {in incriments}

Y amount stored in 01 {in incriments}

X and Y values exceeding 999 incriments cannot be plotted by the plotter.

CONDENSED MANUAL PLOTTING OF LINES

Y coordinate is always in 00 Reg    Y in incriments
X coordinate is always in 01 Reg    X in incriments

There are three basic plotting commands:

1. Plot a line
2. Advance {no plot}
3. Return to zero

These are accomplished as follows:

1. Plot a line

| Key stroke | Instruction | Code Number |
|---|---|---|
| α | Alpha | 0902 |
| : 02 | Plot ΛXΛY | 0502 |
| + 02 | End alpha | 0202 |

| Key stroke | Instruction | Code Number |
|---|---|---|

2. Advance

| | | |
|---|---|---|
| α | Alpha | 0902 |
| : 03 | Advance ΔXΔY | 0503 |
| + 02 | End alpha | 0202 |

3. Return "0"

| | | |
|---|---|---|
| α | Alpha | 0902 |
| : 11 | Return "0" | 0511 |
| + 02 | End alpha | 0202 |

No value in 00 or 01 should exceed 999.

2. Printing with the plotter is done by setting character size, setting the character spacing and then generating α printing codes.

For alphanumberic printing, the plotter requires three bits of information for it to write:

Character size {once}
Character spacing {once}
Numeric code for each character

Setting size:

| | | | |
|---|---|---|---|
| 1. | Enter size {1 to 15} | | |
| 2. | Store in 01 | 0601 | St 01 |
| 3. | Alpha | 0902 | Shift print {α} |
| 4. | Set size | 0508 | : 08 |
| 5. | End alpha | 0202 | + 02 |

Set spacing:

| | | | |
|---|---|---|---|
| 1. | Enter horizontal spacing {in incriments} | 0601 | St 01 |
| 2. | Store in 01 | | |
| 3. | Enter vertical spacing {in incriments} | 0600 | St 00 |
| 4. | Store in 00 | | |
| 5. | Alpha | 0902 | Shift α |
| 6. | Set spacing | 0510 | : 10 |
| 7. | End alpha | 0202 | + 02 |
| 8. | Open and generate alphanumeric codes. | | |

NOTE: Restrictions

Sizes are multiples of single numeric size. Each numeric is 10 units high by 9 units wide. Size 1 allows 99 vertical units and 110 horizontal units.

A table of possible units follows.

| | Number of Vertical Characters | Number of Horizontal Characters |
|---|---|---|
| Size 1 | 100 | 110 |
| Size 2 | 50 | 55 |
| Size 3 | 33 | 36 |
| Size 4 | 25 | 27.5 |
| Size 5 | 20 | 22 |
| Size 6 | 17 | 19 |
| Size 7 | 14 | 16 |
| Size 8 | 12 | 13 |
| Size 9 | 11 | 12 |
| Size 10 | 10 | 11 |
| Size 11 | 9 | 10 |
| Size 12 | 8 | 9 |
| Size 13 | 8 | 9 |
| Size 14 | 7 | 8 |
| Size 15 | 6 | 7 |

Spacing is from center to center.

Vertical spacing should be no less than 10 X character size.

Horizontal spacing should be no less than 13 X character size.

The 13 multiple allows for space between characters.

Don't forget to close α at the end of a printing block.

The next two pages are a listing of the keys, switches, and other codes which will generate the alphanumeric characters.

2. Character list {Table of Alpha Characters}

| CHARACTER | SWITCH TO BE DEPRESSED | KEY TO PRESS |
|---|---|---|
| A | T | 12 |
| B | + | 00 |
| C | + | 12 |
| D | + | 13 |
| E | + | 05 |
| F | | CLEAR {red switch} |
| G | | CLEAR DISPLAY |
| H | + | 01 |
| I | T | 04 |
| J | | 7 |
| K | + | 04 |
| L | – | 15 |
| M | T | 15 |
| N | + | 06 |
| O | T | 09 |
| P | | 5 |
| Q | | 4 |
| R | T | 13 |
| S | T | 01 |
| T | + | 07 |
| U | + | 14 |
| V | T | 14 |
| W | T | 00 |
| X | + | 15 |
| Y | | 1 |
| Z | – | 07 |
| . | T | 06 |
| SPACE | | 2 |
| 0 | – | 01 |
| 1 | + | 09 |
| 2 | – | 06 |
| 3 | – | 14 |
| 4 | – | 09 |
| 5 | – | 05 |
| 6 | – | 04 |
| 7 | – | 13 |
| 8 | – | 12 |
| 9 | – | 00 |
| { | | SET EXP |
| } | | 8 |
| : | ALL UP | 13 |
| , | | CHANGE SIGN |
| ? | | 9 |
| = | | . |
| + | | 6 |
| – | | 0 |
| ' | T | 05 |
| / | | 3 |

One precaution must be taken here: if you ask the plotter to print when it is located immediately beside eitherthe left edge or bottom edge of the plotting area, the characters will be distorted. Therefore, before actually generating the character codes, you should position the pen away from both edges by 1/2 a spacing value or more. You will also find that the generation of characters is somewhat frustrating because of the time consumed in cross referencing and pushing all the character combinations. A technique for using the input/output writer has been developed whereby you can simply type in the letters and they will be duplicated on the plotter.

3. There is another way in which the plotter may be used. It can be instructed to print out the results of calculations located on the display. All procedures concerning scaling, spacing, sizing and positioning were previously discussed and still apply in this case.

A two step command is required for printout of the display value. The first is the I/O command {1502} which alerts the calculator to the need for display printout. The second command is called the format command. Using a combination of a switch and a register key, you indicate the number of digits before and after the decimal point. The selector switch determines the digits to the left of the decimal and the register indicates the positions to the right. There is a maximum number of digits of nine to the left and nine to the right.

A listing of switches and registers follows.

| Selector Switch | Print Positions |
|---|---|
| All Switches UP | 0 |
| "T" Switch DOWN | 1 |
| "+" Switch DOWN | 2 |
| "–" Switch DOWN | 3 |
| "X" Switch DOWN | 4 |
| "÷" Switch DOWN | 5 |
| "St" Switch DOWN | 6 |
| "Re" Switch DOWN | 7 |
| "Sp" Switch DOWN | 8 |
| "Sp" and "T" DOWN | 9 |

decimal point

•

| Function Key | Print Positions |
|---|---|
| $\boxed{00}$ | Suppresses decimal point |
| $\boxed{01}$ | 1 |
| $\boxed{02}$ | 2 |
| $\boxed{03}$ | 3 |
| $\boxed{04}$ | 4 |
| $\boxed{05}$ | 5 |
| $\boxed{06}$ | 6 |
| $\boxed{07}$ | 7 |
| $\boxed{08}$ | 8 |
| $\boxed{09}$ | 9 |

The plotter will ignore "0" (leading zeroes) positions to the left of the decimal and space over one position for each zero. The plotter may also be instructed to space over without printing again by using a two step command.

```
I/O      1502
SP XX    12 XX
```

The low order of the second step determines the number of spaces the pen will move over.

One last note: if you "underformat" that is, if you do not provide for enough positions to the left of the decimal point, the plotter will automatically print the answer in full scientific notation.

CONDENSED I/O WRITING

All sizing, spacing, and scaling rules apply as before.

1. Key I/O
2. Give format    Switches                  Register
                  Digits before decimal     Digits after decimal

Spacing command    12XX        SP X REG

Try some of the examples and problems in the supplement.

# XIII. PLOTTER UTILITY PACKAGE

A. Introduction

B. List of Options

1. Types of scales
2. Types of graphs
3. Types of axes

C. Loading the Options

D. Program Conbinations

1. Pie chart
2. Line graph
3. Bar graph
4. Point graph
5. Linear regression
6. Math function
7. Alpha labeling

E. Drawing and Numbering the Axes

1. Option 12
2. Option 13
3. Option 14
4. Option 15

F. Alpha Labeling

1. Procedure
2. Character list

G. General Procedures and Considerations

H. Summary Sheets: Procedure and Notes on Plotting

I. Math Function Usage

# PLOTTER UTILITY PACKAGE

## Introduction

Wang Laboratories provides a software package of the Wang 600-14 calculator for use with the 612 plotter. This utility package allows the operator selection of many different combinations of fifteen plotting options. These options are grouped into three types:

1. Types of scaling
2. Type of graph
3. Type of axis

The calculator may plot points entered or points calculated. Calculated point plotting is done by using the math function procedure. Since it is the method most used in electronics, it will receive the msot discussion here.

## List of Options

1. Types of scales

[01] X Linear Y Linear
No restrictions

[02] X Linear Y Logarithmic

[03] X Logarthmic Y Linear

[04] X Logarithmic Y Logarithmic
{In choosing the logarithmic option, the operator restricts himself to values which may be neither "0" or negative. An attempt to plot either a "0" or a negative value will result in a program error.}

[05] Polar coordinates
X = redius Y = θ angle
{Calculated or entered data points are limited to the above restrictions. The radius designates

the distance from the origin and the angle is measured counter-clockwise from the x axis.}

2. Types of graphs

| 06 | Pie chart {When doing this graph, NO other options are needed. That is, you need not specify type of scaling and type of axis as they have no meaning in a pie chart.}

| 07 | Point to point line graph

| 08 | Vertical bar chart

| 09 | Point graph

| 10 | Linear regression

| 11 | Math function {This option requires that a subroutine labeled 1101 be written and that this subroutine loads X in register 1 and Y in register 0.}

3. Types of axes

| 12 | X and Y both positive

| 13 | X and Y positive or negative

| 14 | Horizontal lines

| 15 | Hatched fill.

If you desire alpha labeling after the plot is finished, you may key "shift | 01 |." This will give you the option to write on the graph. You may write anything at any point in any size letters. Writing is restricted to horizontal. If you wish to write vertically, you should pass on the alpha labeling and use the standard alphanumeric plotting procedure.

Loading the Options

To select and load the program option proceed as follows:

To select and load the program option proceed as follows:

Insert cassette--Rewind--Tape ready--Run mode--prime--load program--verify.

Display should read 1444.

With f{x} depressed, key $\boxed{\text{00}}$ and then select options desired.

The procedure is used each time you wish to program a particular type of plot. Once done, however, the plotting capability is loaded in the machine and need only be used according to the restriction inherent in each set of options. This procedure needs to be repeated only if the operator wishes to change the options selected.

In the following discussion, the steps specified above will be designated by the words LOAD OPTIONS and then the desired options will be specified. Options are selected by keying registers specified. When finished selecting options, key $\boxed{\text{GO}}$ and options will be loaded automatically.

When loading options, always specify:

scaling first
graph second
axis third

Alpha loading is always the last option.

NOTE: When selecting options--

> ONLY <u>ONE</u> SCALING OPTION, <u>ONE</u> GRAPH OPTION, AND <u>ONE</u> AXIS OPTION MAY BE CHOSEN AT A TIME.

Scaling is a mathematical procedure for the calculator which tells it how to lay out the graph on the area to be plotted. You must later be careful that you obey the restrictions of the scaling method you have chosen. {These have already been specified.}

Axis selection tells the calculator how you wish to draw the axis and is restricted to the type of scaling you have chosen. For example, you would not choose option $\boxed{13}$ with option $\boxed{04}$ or $\boxed{05}$, polar coordinates do not have a Y axis and therefore 14-15 axis labeling method would not be chosen, and so on.

<u>Program Combinations</u>

1. Pie chart: Load options $\boxed{06}$.

2. For line graph of observed data points,

| Load option: | Scaling | Graph | Axis | |
|---|---|---|---|---|
| | 01 | 07 | 12 | |
| Select one | 02 | | 13 | Select one |
| | 03 | | 14 | |
| | 04 | | 15 | |
| | 05 | {No axis selection} | | |

3. Bar graph:

| Load options | Scaling | Graph | Axis |
|---|---|---|---|
| | 01 | 08 | 12 |
| | 02 | | 13 |
| | 03 | | 14 |
| | 04 | | 15 |
| | 05 Scaling NOT usable with bar chart. | | |

4. Point graph:

| Load options | Scaling | Graph | Axis |
|---|---|---|---|
| | 01 | 09 | 12 |
| | 02 | | 13 |

| | | | |
|---|---|---|---|
| | $\boxed{03}$ | | $\boxed{14}$ |
| | $\boxed{04}$ | | $\boxed{15}$ |
| | $\boxed{05}$ No axis selection | | |

5. Linear regression:

| Load options | Scaling | Graph | Axis |
|---|---|---|---|
| | $\boxed{01}$ | $\boxed{09}$ | $\boxed{12}$ |
| | | | $\boxed{13}$ |

Use of $\boxed{02}$ $\boxed{03}$ $\boxed{04}$ $\boxed{05}$ $\boxed{14}$ and $\boxed{15}$ not appropriate

6. Math function:

| Load options | Scaling | Graph | Axis |
|---|---|---|---|
| | $\boxed{01}$ | $\boxed{11}$ | $\boxed{12}$ |
| | $\boxed{02}$ | | $\boxed{13}$ |
| | $\boxed{03}$ | | $\boxed{14}$ |
| | $\boxed{04}$ | | $\boxed{15}$ |
| | $\boxed{05}$ | | |

7. Alpha labeling may not always be compatible with the graph chosen. Therefore, if you wish to label, you will have to scale the calculator after plotting is completed. The pie chart is a good example of this problem.

<u>Drawing and Numbering the Axes</u>

1. Option $\boxed{12}$

Key $\boxed{03}$--X hash spacing--GO--{hashes drawn}
Y hash spacing--GO {hashes drawn}

Key $\boxed{04}$--first X--GO--last X--GO--Incriment X--
GO--{X axis numbered}
--first Y--GO--last Y--GO--Incriment Y--GO--
{Y axis numbered}

Numbering is in whole numbers; decimal parts not printed.

2. Option [13]

Using this option requires a scaling which includes "0." Therefore, since logrithmic scaling cannot be done with either "0" or negative values, these two options are mutually exclusive.

The procedure for this option is the same as for option [12].

3. Option [14]

Key [03]--X hash spacing--GO--{hashes drawn}
--Y line spacing--GO--{lines drawn}

Key [04]--first X--GO--last Y--go--frequency Y--
GO--{Y printed}

Numbering done in whole numbers only.

4. Option [15]

Note: Data must be entered and scaled prior to axis labeling and drawing. Also, data should be entered, if possible, in a sequence of increasing X values. Xmin must be smallestX and Xmax must be largest X used when scaling.

Key [03]--vertical line spacing--GO--{vertical lines drawn}
--Horizontal line spacing--GO--{horizontal lines drawn}

Key [04]--first X--GO--last X--GO--X incriment--
GO--{X axis numbered}
--first Y--GO--last Y--GO--Y incriment--GO--{Y axis numbered}

Alpha Labeling

1. Note: Alpha labeling requires prior scaling.

Key [05]--X starting point--GO--Y starting point--GO--
{pen moves to start}

| NOTE: STARTING POINT POSITION MUST BE SPECIFIED IN THE SAME COORDINATE MANNER AS THE SCALING SCHEME CHOSEN. |
| --- |

2. Character list {Table of Alpha Characters}

| CHARACTER | SWITCH TO BE DEPRESSED | KEY TO PRESS |
|---|---|---|
| A | T | 12 |
| B | + | 00 |
| C | + | 12 |
| D | + | 13 |
| E | + | 05 |
| F | | CLEAR {red switch} |
| G | | CLEAR DISPLAY |
| H | + | 01 |
| I | T | 04 |
| J | | 7 |
| K | + | 04 |
| L | - | 15 |
| M | T | 15 |
| N | + | 06 |
| O | T | 09 |
| P | | 5 |
| Q | | 4 |
| R | T | 13 |
| S | T | 01 |
| T | + | 07 |
| U | + | 14 |
| V | T | 14 |
| W | T | 00 |
| X | + | 15 |
| Y | | 1 |
| Z | - | 07 |
| . | T | 06 |
| SPACE | | 2 |
| 0 | - | 01 |
| 1 | + | 09 |
| 2 | — | 06 |
| 3 | — | 14 |
| 4 | — | 09 |
| 5 | — | 05 |
| 6 | — | 04 |
| 7 | — | 13 |
| 8 | — | 12 |
| 9 | — | 00 |
| { | | SET EXP |
| } | | 8 |
| : | ALL UP | 13 |
| , | | CHANGE SIGN |
| ? | | 9 |
| = | | . |
| + | | 6 |
| - | | 0 |
| ' | T | 05 |
| / | | 3 |

## General Procedures and Considerations

In general, most of the plots can be done by following these steps in order:

1. Key $\boxed{00}$ and enter data.
2. Key $\boxed{01}$ and scale the calculator.
3. Key $\boxed{02}$ and plot the graph.
4. Key $\boxed{03}$ and draw the axes.
5. Key $\boxed{04}$ and number the axes.
6. Key $\boxed{05}$ and alpha label.

The pie chart is a program by itself. However, if alpha labeling is desired, then it will be necessary to choose the scaling with X and Y both linear as well as the alpha labeling option. After the pie chart itself is drawn, scale the calculator by touching key $\boxed{01}$ and then locate the point to print according to how it was scaled.

Do not use polar coordinates with any type of axes, with the bar chart or with linear regression. Locate the point to start alpha labeling a polar plot in terms of r and θ.

The axes cannot be numbered with decimal fractions.

With the hatched axes, X-min should be the minimum value of X and X-max should be the maximum value of X.

Do not use any logarithmic type of scaling with either linear regression or the hatched axes.

The following is a one page summary of procedure and options.

## Summary Sheets: Procedure and Notes on Plotting

Loading the Program

| | | | |
|---|---|---|---|
| Turn on Machine | Display | | |
| Clear - Prime - Run Mode | | | |
| Insert Cassette | | | |
| Rewind - Tape ready | | | |
| Load Program | | | |
| Verify | 1444.0000000 | | |
| Key ———————→ | f(x) | Reg 00 | X linear Y linear |
| Choose scale type | f(x) | Reg 01 | X linear Y linear |
| | | 02 | X linear Y log |
| | | 03 | X log Y linear |
| | | 04 | X log Y log |
| | | 05 | Polar coordinate |
| | Log scales may not be equal to 0 or - value. | | |
| Choose type of graph | f(x) | Reg 06 | Pie chart |
| | | 07 | Line chart |
| | | 08 | Bar chart |
| | | 09 | Point chart |
| | | 10 | Linear regression |
| | | 11 | Math function |
| Choose type of axis | f(x) | Reg 12 | X & Y positive |
| | | 13 | X & Y (positive or negative) |
| | | 14 | Horizontal lines |
| | | 15 | Hatched |
| Alpha labeling optional | f(x) | Shift 01 | |

CAUTION: Labeling program is set to print only horizontally. If you wish vertical labeling, use the standard alphanumeric procedure.

2. Character list {Table of Alpha Characters}

| CHARACTER | SWITCH TO BE DEPRESSED | KEY TO PRESS |
|---|---|---|
| A | T | 12 |
| B | + | 00 |
| C | + | 12 |
| D | + | 13 |
| E | + | 05 |
| F | | CLEAR {red switch} |
| G | | CLEAR DISPLAY |
| H | + | 01 |
| I | T | 04 |
| J | | 7 |
| K | + | 04 |
| L | – | 15 |
| M | T | 15 |
| N | + | 06 |
| O | T | 09 |
| P | | 5 |
| Q | | 4 |
| R | T | 13 |
| S | T | 01 |
| T | + | 07 |
| U | + | 14 |
| V | T | 14 |
| W | T | 00 |
| X | + | 15 |
| Y | | 1 |
| Z | – | 07 |
| . | T | 06 |
| SPACE | | 2 |
| 0 | – | 01 |
| 1 | + | 09 |
| 2 | — | 06 |
| 3 | — | 14 |
| 4 | — | 09 |
| 5 | — | 05 |
| 6 | — | 04 |
| 7 | — | 13 |
| 8 | — | 12 |
| 9 | — | 00 |
| { | | SET EXP |
| } | | 8 |
| : | ALL UP | 13 |
| ¬ | | CHANGE SIGN |
| ? | | 9 |
| = | | . |
| + | | 6 |
| – | | 0 |
| ' | T | 05 |
| / | | 3 |

## General Procedures and Considerations

In general, most of the plots can be done by following these steps in order:

1. Key $\boxed{00}$ and enter data.
2. Key $\boxed{01}$ and scale the calculator.
3. Key $\boxed{02}$ and plot the graph.
4. Key $\boxed{03}$ and draw the axes.
5. Key $\boxed{04}$ and number the axes.
6. Key $\boxed{05}$ and alpha label.

The pie chart is a program by itself. However, if alpha labeling is desired, then it will be necessary to choose the scaling with X and Y both linear as well as the alpha labeling option. After the pie chart itself is drawn, scale the calculator by touching key $\boxed{01}$ and then locate the point to print according to how it was scaled.

Do not use polar coordinates with any type of axes, with the bar chart or with linear regression. Locate the point to start alpha labeling a polar plot in terms of r and θ.

The axes cannot be numbered with decimal fractions.

With the hatched axes, X-min should be the minimum value of X and X-max should be the maximum value of X.

Do not use any logarithmic type of scaling with either linear regression or the hatched axes.

The following is a one page summary of procedure and options.

## Summary Sheets: Procedure and Notes on Plotting

| | | |
|---|---|---|
| Loading the Program | | |
| Turn on Machine | | Display |
| Clear - Prime - Run Mode | | |
| Insert Cassette | | |
| Rewind - Tape ready | | |
| Load Program | | |
| Verify | | 1444.0000000 |
| Key ⟶ | f(x) Reg 00 | X linear Y linear |
| Choose scale type | f(x) Reg 01 | X linear Y linear |
| | 02 | X linear Y log |
| | 03 | X log Y linear |
| | 04 | X log Y log |
| | 05 | Polar coordinate |
| | Log scales may not be equal to 0 or - value. | |
| Choose type of graph | f(x) Reg 06 | Pie chart |
| | 07 | Line chart |
| | 08 | Bar chart |
| | 09 | Point chart |
| | 10 | Linear regression |
| | 11 | Math function |
| Choose type of axis | f(x) Reg 12 | X & Y positive |
| | 13 | X & Y (positive or negative) |
| | 14 | Horizontal lines |
| | 15 | Hatched |
| Alpha labeling optional | f(x) Shift 01 | |

CAUTION: Labeling program is set to print only horizontally. If you wish vertical labeling, use the standard alphanumeric procedure.

<u>Math Function Usage</u>

The following is a condensed set of instructions for using the math function graph option.

Load options [01] [11] [13] shift [01]
--GO--

When GO is keyed, the program options you have chosen will be automatically loaded.

{Now add your subroutine}

*Verify program ——————→ Key Subroutine
⋮
Run Mode

Subroutine limitations:

Subroutine for math function must carry a F1 mark.
X value must always be brought to Reg 01 {1101}
Y value must always be brought to Reg 00 if polar coordinates are being plotted
Radius is brought to Reg 01
Angle is brought to Reg 00

The following registers are not available for use in calculations:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 07 | 08 | 09 | 10 | 11 | 12 | 13 |
| {T00} | {T01} | {T02} | {T03} | {t04} | {T05} | | | |
| 16 | 17 | 18 | 19 | 20 | 21 | | | |

Subroutines and marks used in the function subroutine may use only the codes listed below or as subroutine using "1507" marks. {SP-Re [07] } {Register Key}

---

*NOTE: It has been found to be very convenient to step over the end program command after verifying and before keying the subroutine. This allows you to very quickly locate the beginning of the subroutine later if changes are desired.

| | | | |
|---|---|---|---|
| ALL up | Reg 00 | → | 15 |
| T | Reg 00 | → | 15 |
| + | Reg 00 | → | 15 |
| - | Reg 00 | → | 15 |
| x | Reg 00 | → | 15 |

| | Display Reads |
|---|---|
| Key f{x} 01 Scaling Calculator | +1.000000000 |
| X min GO | +2.000000000 |
| X max GO | +3.000000000 |
| Y max GO | +4.000000000 |

{Pen goes to plot start point.}

| | Display Reads |
|---|---|
| Key f{x} 02 Plotting Function | +1.000000000 |
| T min GO {initial value} not necessarily X max | +2.000000000 |
| T Max GO {final value} not necessarily X max | +3.000000000 |
| ΔT GO {incriment} | |

{Function will be plotted.}

Key f{x} 03

Drawing axis with hash marks {Dark screen}

Key X Hash Mark Separation--GO{Dark screen}
X axis will be drawn with hash marks.

Key Y Hash Mark Separation--GO{Dark screen}
Y axis will be drawn with hash marks

| | |
|---|---|
| Key f{x} 04 Numbering the Axis | +1.000000000 |
| First X--GO | +2.000000000 |
| Last X--GO | =3.000000000 |
| Frequency of X--GO | |

{X axis will be numbered.}

NOTE: X axis numbers are set up to allow for four {4} digits centered on hash marks. Therefore, single digit numbers will be offset to right of hash marks. You can correct this by adjusting the "0" reference horizontally {X} a small amount prior to the numbering process. Use caution in doing this because you are altering the entire scaling area of the plotter.

| | |
|---|---|
| | +1.000000000 |
| First Y--GO | +2.000000000 |
| Last Y--GO | +3.000000000 |

{Y axis will be numbered}

PRECAUTION: These are restrictions on the labeling program which may cause problems. Labeling can usually be accomplished with more flexibility by using the procedures given in the Flat Bed Plotter instructions section.

Remember, when scaling Log graphs, you may not use "0's" or negative numbers on the logarithmic axis. Also, when scaling for polar coordinates, you need only specify the maximum radius and key GO. Realize also that when choosing the $\boxed{05}$ {Polar} option, an axis need not be specified. You would only choose an axis if you desired one on the drawing.

For other examples of options and combinations, you may refer to the Utility Package book provided by Wang Laboratories.

## XIV. DEBUGGING A PROGRAM

A. Definitions

B. Error Indications

1. Check with known values
2. Program error indicator
3. Machine error indicator
4. Verify codes

C. Finding errors

1. Step
2. Backstep
3. List program function
4. Mark search and search mark {S.M.}
5. Trace

D. Correcting a Program

1. Changing
2. Deleting
3. Inserting

# DEBUGGING A PROGRAM

## Definitions

The most experienced programmer may spend weeks on a program only to find that it does not work. In his program, there are certain errors that keep it from working. These errors, or "bugs," can be in the form of an incorrectly keyed program or in the form of incorrect logic on the part of the programmer. To help the programmer "debug" his programs, the Wang has several features to make debugging easier.

## Error Indication

1. When first running the program, it is always a good idea to enter values to which the answer is already known. This will provide the quickest and easiest check of the program. If the answer is what was expected, then no debugging is necessary.

2. Sometimes, when running the program, the program error indicator will light and the display will flash. This is an automatic feature of the Wang which checks for certain types of errors in a program. These are the conditions which will cause the program error indicator to activate:

    1. Calculated results exceeding $10^{99}$ {overflow condition}
    2. Dividing by zero
    3. Finding the square root of a negative number.
    4. Finding the natural log or common log of a non-positive number.
    5. Searching for a non-existent mark.

6. Exceeding capacity of memory with tape.*
7. No "End Program" command when recording or verifying.
8. Sin, cos or tan of an angle greater than 10 radians or 572°.
9. $Sin^{-1}$, $Cos^{-1}$ of a number whose absolute value is greater than 1.

3. The machine error indicator (located next to the program error indicator) will light if data is not transferred properly from or onto the tape. To correct this situation, prime as you would if the program error indicators were to light and then repeat the loading or recording operation.

4. Another way to check if an error were to be introduced in the program is by using the verify program key. The verify program key adds the first two digits of the program codes beginning at step 0000 until it encounters END code. The sum is shown on the display.

EXAMPLE:

| STEP | KEY | CODE | |
|---|---|---|---|
| 0000 | MARK | 0900 | ← 09 00 |
| 0001 | 1 | 0001 | ← 00 01 |
| 0002 | $x^2$ | 0812 | ← 08 12 |
| 0003 | STOP | 0913 | ← 09 03 |

*NOTE: If an End Program command is located on the last step in the program memory (step 0311 for Model 600-2, step 0825 for Model 600-6, and step 1847 for Model 600-14) the program will load properly, but the error indicator will go on. Even though the End Program command is not missing, the Program Error indicator will go on when this program is transferred to tape. If this is the case, prime and ignore the Program Error indicator.

0004 END PROG. 9014 $\left\{\begin{array}{l}09\\14\end{array}\right.$ 
$\overline{\quad 65}$

THE VERIFY CODE FOR THIS PROGRAM IS 65.

This is valuable when re-introducing a program to the machine. If the correct verify code is known, then it is easy to see if the program put in memory is correct by seeing if it has the correct code. It is very unlikely that two verify codes would be the same for different programs.

{After the execution of the Verify Program operation, the Program Counter is set at the End Program step.}

Now that we have discussed ways of determining if an error exists, the next section will deal with finding where the errors are in the program.

Finding Errors

Suppose the program entered does not work. The cause of error will be either a mistake created by incorrectly entering the program or by incorrect logic when the program was written. A person experienced in programming and with the general operating characteristics of the Wang will make fewer logic errors such as introducing undefined values or improperly arranging steps or subroutines. However, whatever the error is, it should be fairly obvious to a programmer at this stage after a little study of the program.

1. Debugging requires an understanding of the four digit operation code in the machine. This code can be translated by use of the pull-out cards under the Wang or by the same

table reprinted in the back of this manual. By recognizing the codes, theoperator can put the Wang in learn mode and step thraugh the operational codes to see the program in memory with the Step key, starting at a step set by the program counter.

2. It is possible to Back Step in a program by using the upper case of the Verify Program key. This is accessed by putting the Wang in Run-Learn mode. {Simultaneously depress both Run and Learn switches.} Then use the key as you would use the step key.

3. It is also easier to see mistakes if the entire program is printed out on paper. To do this, put the Wang in List Program mode, prime {or set the Program Counter to the point to be listed} and key GO. The program will be printed out until the End Program command is reached. There will also be on the tape an alphanumeric translation of the operation code to make debugging easier.

Suppose we want to watch values run through a program to determine where errors are. If this is the case, put the Wang in run mode, enter values in the appropriate registers and then use the step key and follow the values through the program. The programmer will have some idea of what to expect and will therefore find at what point the error is made.

4. It is possible to only run one subroutine by using the Mark-Search routine. This is done by keying search and then

the flag which corresponds to the subroutine to be run. The program will run from that point.

It may be desirable to go to the point where the subroutine starts and then stop before executing. This is done with the Search-Mark function found on the Prime key (Wang 600 only). Place in the Run-Learn mode, key the Search-Mark key, and then the flag which corresponds with the section of the program to be studied. The display will show the first command after the Mark number. The step key can be used in this mode to check the codes, or the Wang can be returned to run mode and values can be stepped through the subroutine.

5. Suppose we wanted the Wang to step through a program automatically in run mode, print out all the values as they moved through the program after every step and indicate the function being processed. This is possible on the Wang 600 with the Trace function. Before running the program, key Alpha ($\alpha$), $Log_e x$ (or Alpha, $Log_{10} x$) to turn the Trace on. Then run the program. To turn off the Trace function, key Alpha, $e^x$ (or Alpha, $10^x$). It is possible to place the trace on and off functions into the program to trace certain sections as the program runs. This is done by inserting steps in a program, a procedure which will be covered in the next section on correcting a program.

Correcting a Program

1. Many times when debugging a program, a mistake will be

found that only needs to be replaced or "written over." To do this, switch to Learn mode. Set the program counter to the step to be changed and then key in the correct command. The program is now changed. {This method will write over existing steps. It cannot be used to add extra steps.}

2. To correct a program it is sometimes necessary to delete a step or steps. On the Wang 500, this was done by replacing the unnecessary steps with GO commands. On the Model 600, it is possible to delete a step entirely and therefore save program steps. This is done by going to the step or steps to be deleted, placing the Wang in Run-Learn mode and then keying Delete. The step deleted will disappear and all of the remaining steps will move up to take its place. EP {End Program} steps are automatically inserted at the end of the program as the steps move up into the new positions.

3. The most powerful debugging feature on the Wang 600 {only} is its ability to insert steps. On the 500, it is necessary to rewrite the entire program from the point where steps are added; but on the 600, the new steps are inserted directly into the program between existing ones to make room for the new commands.

To insert a step, set the program counter to the command which is to immediately follow the inserted step or steps. Then put the Wang 600 in Run-Learn mode and key Insert. This will insert a GO command into the program. The command that was on the display will have moved up one step along with all the commands after it. To insert two steps, key Insert twice and all the commands after it will move up two steps, and so on. The steps that were inserted were 0803 GO commands. These are now written over by setting the program counter to the desired step (or using the backstep key) and keying the new step or steps over the GO commands.

Remember: Before you insert, the program step that is to follow must be on the display.

If you must insert steps in more than one place in a program it will be easier to locate the places if the last set of inserts is done first. Since doing an insertion changes the step numbers of the steps following the insertion, locating other places in the program would be more difficult if the first set of inserts were done first.

Every programmer develops his own preferences and techniques from the methods described in this chapter. Even if a programmer never makes mistakes, some programs can be changed for the sake of improvement or adapted for other programs and is easily done with the methods previously discussed here.

## XV. SPECIAL ADDRESSING

A. Introduction

B. Memory Layout

1. Diagram

2. Arithmetic

3. Sequential Storage

C. Variable Length Jump

# SPECIAL ADDRESSING

## Introduction

Both the Wang 500 and 600 can interpret the internal registers as either programmed steps or as data storage positions. Programming has already been exolained. There is a technique known as indirect addressing which allows the operator to use the internal storage as data handling register. That is, you can add, subtract, multiply, divide, store and recall values in any of the substorage registers using the upper sixteen registers as "pointer" registers. The term "indirect" comes from the fact that you "get to" the register by an indirect path using the upper registers to point the way.

1. To understand the procedrue, you need to understand the layout of the registers in the machine. Look at the diagram on the following page.

2. To indirectly address any one of registers 16 through 118, you store the number in one of the upper registers and then key, "indirectly," the functional operation switch and the pointer register in which you stored the address.

EXAMPLE: Suppose you wished to add 45 and 79 in register 89. Use 00 as pointer.

```
Key    89
       ST 00 {pointer register 00 now contains the
              address "89"}
       45
       indirect
       ST 00 {45 is now in register 89}
       79
       indirect
       +00   {79 + 45} now in 89
```

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Used as Pointers | | | | | | | | | | | | | | | | Upper Registers |
| | | | | | | | | | | | | | | | | | Step |
| T | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 823 ← 696 |
| + | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 695 ← 568 |
| - | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 567 ← 440 |
| X | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 439 ← 312 |
| ÷ | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 91 | 92 | 93 | 94 | 95 | 95 | 311 ← 184 |
| ST | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 183 ← 56 |
| Re | 112 | 113 | 114 | 115 | 116 | 117 | 118 | | | | | | | | | | 55 ← 0 |

Once you have established a pointer register, arithmetic commands performed in the substorage become simple two step commands.

3. The indirect procedure may also be used to sequentially store numbers in registers.

EXAMPLE: Store the numbers 20 through 30 in Registers 45 through 55. Use 04 as the pointer.

```
Key   20                  Mark 1
      STL                 Re L
      45                  Indirect
      ST 04               ST 04
      Search 1            1
                          +L
                          1
                          +04
                          ST Rt
                          56
                          -Rt
                     --   j if +
                     |    Search
                     |    1
                     --   EP
```

The numbers 20→30 are now stored in Registers 45 → 55.

There are many ways in which the ability to address the lower registers can be extremely useful. It is well worth your time to explore this feature of the 600.

Variable Length Jump

When the program encounters the two step command "Indirect 00XX" {where XX is the register in which the number of steps to be jumped is stored}, it will add the contents of the XX register to the program counter causing the program to skip that number of steps.

EXAMPLE: Suppose you wanted the program to skip from step 450 to 565.

```
 565
-450
-----     You need to skip 115 steps.
 115
```

STORE 115 in one of the upper registers (e.g. 06).
At step 449, the program should read:

| STEP | CODE | COMMAND |
|---|---|---|
| 0449 | 1511 | Indirect |
| 0450 | 0006 | 06 |
| | | |
| 0565 | XXXX | XXXX |

The calculator automatically takes the contents of 06 and adds it to the program counter causing a 115 step jump.

You may jump up to 999 steps if your calculator has that capacity.

NOTE: 1) Only the absolute integer portion of the contents of the keyboard register designated are used. Negative numbers are interpreted as positive. Thus "jump" can only be forward in the program flow.

2) If a jump is attempted to a non-existent step (e.g. --step 2000), execution halts and the program error light turns on.

## XVI. INPUT/OUTPUT WRITER

A. Introduction

B. Modes of Operation

1. Local mode

2. Output mode

3. Input mode

4. Type mode

C. Programming from I/O Writer

D. Correcting Input Errors

E. Printing Display Answers

F. I/O Spacing Commands

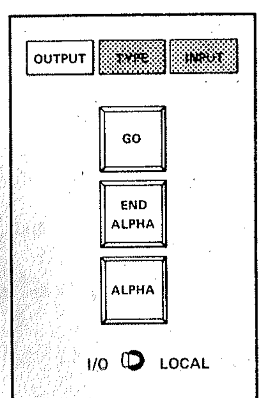
# MODEL 611 INPUT/OUTPUT WRITER

## Introduction

The 611 Input/Output Writer {I/O Writer} is an automatic typewriter. It will print out alphanumerics upon program command from the 600 calculator. The I/O Writer may also input alphanumeric programs into the Wang. And, of course, it may be used as a regular typewriter.

## Modes of Operation

The I/O Writer itself has four modes of operation: output, input, type and local modes. All are indicated on the I/O control panel. For the four modes see the figure below.

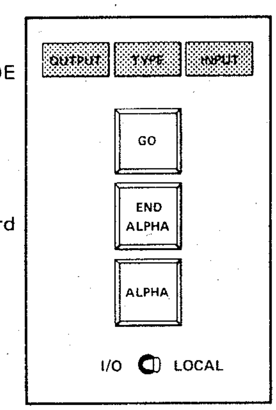



611 IN I/O (OUTPUT) MODE

I/O Writer Keyboard Locked

611 IN LOCAL MODE

I/O Writer Keyboard Unlocked




1. When the toggle switch on the I/O Writer control panel is placed into the local position, the Writer is in the local mode. This enables you to use it as a regular electric typewriter independent of the caluclator. If the calculator is executing a program and the switch is placed to

Local, the calculation will be interrupted and the writer keyboard freed for local maniuplation. When switched back, the execution of the program will continue from where it left off.

2. When the toggle switch on the I/O Writer is placed into the "I/O" position, the output indicator light will illuminate. This shows that the writer is ready to print upon program command from the 600. The writer keyboard is now locked up.

3. The input mode is used to input the alphanumeric program codes into the calculator by use of the I/O keyboard. To change to the input mode, the codes {1514} and {0412} must be generated in the calculator. For example, place calculator into the run mode.

key group 2 {1514}
depress {x} and key Reg 12 {0412}

The I/O control panel should now have the input light illuminated. The calculator keyboard is now locked up, with the exception of the debugging and PRIME keys.

Thechanging of the writer to input mode can also take place in a program.

4. The type mode is a way of letting you type something while the calculator is still running a program. The codes {1514} and {0413} are encountered, the calculation will stop thus allowing you to use the writer without generating program codes into the 600. When you have finished typing, key GO and the calculation will continue under program

control.

Here is an example of how to place the writer into type mode:

Depress run button on calculator.
Key Group 2 {1514}
Depress {x} and key Reg 13 {0413}

Programming from I/O Writer

After the writer has been switched to input mode, any key that is touched on the I/O keyboard will produce a code in the 600.

Any program which is to be printed out on the writer must be preceded by a write alpha command and terminated with an end alpha command. The calculator will then print out anything within the alpha and end alpha commands. For example, to print out "Wang Labs," follow this procedure:

key PRIME
place I/O writer to INPUT MODE
key WRITE ALPHA
key CR/LF {carriage left}
key INDEX
key SPACE
key SPACE
key UP SHIFT
key " {quotation marks}
key W
key DOWN SHIFT
key A
key N
key G
SPACE
UP SHIFT
L
DOWN SHIFT
a
b
s
UP SHIFT
xx
DOWN SHIFT

key END ALPHA

THIS PROGRAM IN NOW IN THE 600.

key GO on I/O {Returns command back to calculator}
depress LEARN mode button on the 600
key END PROGRAM {EP} {Every program must have an EP command.}
depress RUN mode button

THE 600 IS NOW READY TO RUN YOUR PROGRAM.

key GO on calculator to run program.

## Correcting Input Errors

As mentioned before, one of the advantages of programming the calculator via the I/O Writer is that the programmer can watch for input errors as he types and easily correct them. If a major error is found, the easiest way to correct it is to reset the program counter on the calculator {reset P.C.}, and use the I/O Writer to re-enter the entire string of alphanumerics.

Suppose, however, a programmer was using the I/O Writer to enter some alphanumeric characters into the calculator and found himself in the following situation:

| 0041 - - - - |
|---|

Number    Regression Coefdicients ← I/O Writer Carriage in this position
8 positions

That is, while programming some column headings, he inadvertently typed the letter "d" instead of an "f" in the word "coefficients." He could retype the entire line or he could retype just a single letter. If he chooses to retype just the letter, there are two possible ways of proceeding:

PROCEDURE I FOR CORRECTING ERRORS

Step a. Key END ALPHA on the I/O Writer {if this was not already done}

Step b. Put the calculator in the RUN/LEARN mode {that is, depress both the RUN and LEARN mode switches simultaneously}

Step c. Key B.S. {BACKSTEP} <u>8 times</u> {remember--only programmable keys on the calculator are locked at this time--therefore, the BACKSTEP key will work}.

The situation now looks like this:

Programming Position According to Calculator

Program Code for letter "d"

Number Regression Coefdicients

8 positions

| 0033 02 13 |

Actual position of the I/O Writer Carriage is still here

That is, although the I/O Writer is positioned at the space following the letter "s," the program counter {and therefore the program step} of the calculator is positioned at the letter "d."

Step d. Now depress the "f" key on the I/O Writer's keyboard, thereby replacing the "d" with an "f."

The situation would now look like this:

On Paper:

Calculator Display

Number Regression Coefdicientsf

| 0034 01 04 |

I/O Writer positioned here

Program Code for letter "i"

In the Calculator Memory:

Number Regression Coefficients

Programming position according to the calculator

Step e. Put the calculator back in the RUN mode {i.e. depress the RUN mode switch}.

Step f. Since the error has now been corrected, depress the GO key on the I/O Writer. This returns programming control to the calculator.

To be sure that all errors have been corrected and that the alphanumeric programming has the proper format, make a test run of the printout.

NOTE: When attempting to correct alphanumeric programming errors, remember to count all shift up, shift down, and tabbing operations as program steps.

PROCEDURE II FOR ERROR CORRECTION

Step a. Key END ALPHA {if this has not already been done.}

Step b. Key GO on the I/O Writer. This will transfer programming control back to the calculator.

Step c. In RUN mode, SET P.C. to the beginning of the alphanumeric programming, and STEP through the alphanumeric printout. The diagram below shows the situation after the STEP key has been keyed 6 times {i.e. after Program Step 0005 has been executed.}

Num

Position of the I/O Writer

Key STEP until the mistake is printed. The situation will now look like this:

Number Regression Coefd

Position of the I/O Writer

Step d. If the calculator is now put in the RUN/LEARN mode, the display will show the program code of the next letter to be printed:

| 0032 0104 |

The code for letter "i"

If you key B.S. { BACKSTEP }, the display will then show the code for the letter "d", the error:

| 0031 | 0213 |
|---|---|

The code for letter "d"

Therefore, you know that the mistake is located in step no. 0031.

Step e. Put the calculator back into RUN mode.

Step f. Now transfer programming control to the I/O Writer: On the calculator, key GROUP 2, and with the {X} Selector Switch down, key 12 .

Since programming control is now centered in the I/O Writer, the calculator display will show:

| 0031 | 0213 |
|---|---|

although the calculator's RUN mode switch is still depressed.

Step g. Depress the "f" key on the I/O Writer keyboard, thereby replacing the "d" with an "f." The correction has now been made. Therefore, depress the GO key on the I/O Writer to return control to the calculator.

The error correction procedure is now complete.

Depress Learn mode button on the 600.

The 600 is now ready to run your program.

Key GO on calculator to RUN program.

In addition to programming alphanumeric characters into the calculator using the I/O Writer, these characters can be entered into program memory via the calculator keyboard. To do this, the programmer uses the Special Selector switches and the data register keys at the top of the calculator keyboard.

The Selector switches are used to generate the "high order" part of a program code, while the data register keys are used to generate the low order code:

High Order $\underbrace{01}$ $\underbrace{08}$ Low Order

Program Code

Below is a listing of the 611 I/O Writer Format commands.

### 611 I/O WRITER FORMAT COMMANDS

| FUNCTION | CODE | SELECTOR SWITCH SETTING (Generates High-Order Code) | FUNCTION KEY SETTING (Generates Low-Order Code) |
|---|---|---|---|
| Space | 0002 | All Switches Up | 02 |
| Backspace | 0003 | All Switches Up | 03 |
| Index (LF) | 0110 | "T" Switch Down | 10 |
| Shift Up | 0103 | "T" Switch Down | 03 |
| Shift Down | 0102 | "T" Switch Down | 02 |
| CR/LF | 0108 | "T" Switch Down | 08 |
| Tab | 0008 | All Switches Up | 08 |
| Clear Tab | 0011 | All Switches Up | 11 |
| Set Tab | 0010 | All Switches Up | 10 |

The following page contains the manual entry codes for printing.

## MANUAL ENTRY CODES FOR PRINTING

| CHARACTER | CODE | SELECTOR SWITCH SETTING | FUNCTION KEY SETTING |
|---|---|---|---|
| A/a | 0112 | "T" Switch Down | 12 |
| B/b | 0200 | "+" Switch Down | 00 |
| C/c | 0212 | "+" Switch Down | 12 |
| D/d | 0213 | "+" Switch Down | 13 |
| E/e | 0205 | "+" Switch Down | 05 |
| F/f | 0014 | All Switches Up | 14 |
| G/g | 0015 | All Switches Up | 15 |
| H/h | 0201 | "+" Switch Down | 01 |
| I/i | 0104 | "T" Switch Down | 04 |
| J/j | 0007 | All Switches Up | 07 |
| K/k | 0204 | "+" Switch Down | 04 |
| L/l | 0209 | "+" Switch Down | 09 |
| M/m | 0115 | "T" Switch Down | 15 |
| N/n | 0206 | "+" Switch Down | 06 |
| O/o | 0109 | "T" Switch Down | 09 |
| P/p | 0005 | All Switches Up | 05 |
| Q/q | 0004 | All Switches Up | 04 |
| R/r | 0113 | "T" Switch Down | 13 |
| S/s | 0101 | "T" Switch Down | 01 |
| T/t | 0207 | "+" Switch Down | 07 |
| U/u | 0214 | "+" Switch Down | 14 |
| V/v | 0114 | "T" Switch Down | 14 |
| W/w | 0100 | "T" Switch Down | 00 |
| X/x | 0215 | "+" Switch Down | 15 |
| Y/y | 0001 | All Switches Up | 01 |
| Z/z | 0307 | "-" Switch Down | 07 |
| [/] or 1/± | 0315 | "-" Switch Down | 15 |
| @/2 | 0306 | "-" Switch Down | 06 |
| #/3 | 0314 | "-" Switch Down | 14 |
| $/4 | 0309 | "-" Switch Down | 09 |
| %/5 | 0305 | "-" Switch Down | 05 |
| ¢/6 | 0304 | "-" Switch Down | 04 |
| &/7 | 0313 | "-" Switch Down | 13 |
| */8 | 0312 | "-" Switch Down | 12 |
| (/9 | 0300 | "-" Switch Down | 00 |
| )/0 | 0301 | "-" Switch Down | 01 |
| _/- | 0000 | All Switches Up | 00 |
| +/= | 0006 | All Switches Up | 06 |
| 1/4 1/2 or °/l | 0107 | "T" Switch Down | 07 |
| :/; | 0013 | All Switches Up | 13 |
| "/' | 0105 | "T" Switch Down | 05 |
| ,/, | 0012 | All Switches Up | 12 |
| ./. | 0106 | "T" Switch Down | 06 |
| ?// | 0009 | All Switches Up | 09 |

The above commands also must be preceded by an ALPHA command and end with an END ALPHA command (02 02).

NOTE: Each of the commands on the previous two pages must be preceded by an ALPHA command {0902}, and end with an END ALPHA command {0202}.

## Printing Display Answers

Numerical answers can also be printed out on the I/O Writer. The Writer will only print out the number that is in the display of the calculator. To print this number out on the I/O, a two step command is needed. The first step is the I/O command. Key the I/O button thus producing the {1502} code. The second command is the formating command which serves as a means of telling the I/O how it is to print the number out.

You have a choice of how many digits before and after the decimal point are to be printed out.

This command must immediately follow the I/O command.

Here is a table of your formating choices:

The selector switches generate the # of positions to the left of the decimal point.

| Selector Switch | Print Positions |
|---|---|
| All up | 0 |
| "T" Switch down | 1 |
| "+" Switch down | 2 |
| "-" Switch down | 3 |
| "X" Switch down | 4 |
| "÷" Switch down | 5 |
| "St" Switch down | 6 |
| "Re" Switch down | 7 |
| 'Sp" Switch down | 8 |
| "Sp" and "T" down | 9 |

decimal point •

The function keys generate the # of positions to the right of the decimal point.

| Function Key | Print Positions |
|---|---|
| 00 | Suppresses decimal pt. |
| 01 | 1 |
| 02 | 2 |
| 03 | 3 |
| 04 | 4 |
| 05 | 5 |
| 06 | 6 |
| 07 | 7 |
| 08 | 8 |
| 09 | 9 |

As you can see, there can be a maximum of nine digits to the left and to the right of the decimal point.

Example 1

Key the number 987654321, followed by

| | Calculator Keystrokes | Code Generated |
|---|---|---|
| 1. | I/O | 1502 |
| 2. | With the (Sp) and (T) Selector Switches down, key $\boxed{09}$ | 0909 |

The I/O Writer will print out

initial I/O Writer position | |987654321.000000000 final position

1 space left for sign (here "+" is assumed)

Example 2

Now key .123456789. followed by

| | Calculator Keystrokes | Code Generated |
|---|---|---|
| 1. | I/O | 1502 |
| 2. | With the (Sp) and (T) Selector Switches down, key $\boxed{09}$ | 0909 |

The I/O Writer will print out

initial I/O Writer position | | .123456789 final position

10 spaces = 1 for sign + 9 leading spaces

As you can see, the I/O Writer will not print out leading zeros. Instead, it will space over the indicated number of spaces.

To have a printout in scientific notation, all that has to be done is to under format the left of the decimal.

EXAMPLE $\boxed{-123.4560000}$

If the calculator display looked like this and you attempted to print out the number with less than three digits to the left of the decimal point, let's say as follows:

| Calculator Keystrokes | Code Generated |
|---|---|
| 1. Key I/O | {1502} |
| 2. With the {T} switch down, key <u>04</u> | {0104} |

The I/O Writer will print out the following:

-1.234560000e 02

In this case, it did not matter how many digits you instructed the I/O Writer to type after the decimal point. The number would always be printed in scientific notation, since you instructed the Model 611 to type out only one digit in front of the decimal point, when the number actually had three digits in front of the decimal point. This is a case of under-formatting.

## I/O Spacing Commands

These codes will allow you to have the carriage skip up to fifteen spaces without using fifteen steps.

| | | | |
|---|---|---|---|
| I/O<br>one space | 1502<br>1201 | — | Generated by keying $\boxed{01}$ with the (Sp) and (X) switches down. |
| I/O<br>two spaces | 1502<br>1202 | | Generated by keying $\boxed{02}$ with the (Sp) and (X) switches down |
| I/O<br>three spaces | 1502<br>1203 | — | Generated by keying $\boxed{03}$ with the (Sp) and (X) switches down |
| — | — | — | — — |
| — | — | — | — — |
| — | — | — | — — |
| I/O<br>fifteen spaces | 1502<br>1215 | — | Generated by keying $\boxed{15}$ with the (Sp) and (X) switches down |

XVII. RECORDKEEPING

A. Purpose of Recordkeeping

B. What to Include on a Specific Program

1. Program description {storage method}

2. Flow diagram

3. Program steps

4. Operational instructions

5. Example of program results

C. Program Files

1. Card files

a. Master card index

b. Reverse side notes

c. Notebook file

2. Tape

a. Program index

b. Loading instructions

D. Summary

# RECORDKEEPING

## Purpose of Recordkeeping

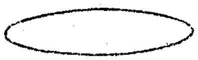
Keeping a complete and thorough record of your programs as they are developed is easily as important as developing the program itself. Even the best program in the world is useless if you cannot remember how to use it or what the results show. It is therefore extremely important for you to develop proper recordkeeping habits right along with your programming skills. What follows is a short discussion concerning what should be included in your documentation of a program as it develops.

## What to Include in a Specific Program

1. When setting out to write a program, write briefly what you want the program to accomplish and the order in which you wish that calculation to be done. Also determine the method of storage: tape, cards, or simply written record. The decisions made at this point determine the layout and progression of the program and form the basis for later formal description of the program.

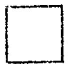
2. Next, lay out a rough flow diagram of the program. This diagram need not include a great amount of detail but should block out the main routine, subroutines and decision points. Later, after the actual program has been finished, the flow chart should be refined to include all the details of the program as loops, marks, data entry points, decision conditions, etc.

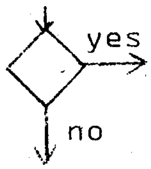
The following flow diagram symbols my be useful.

 start or stop

———————— main flow

-------- subroutine flow

 program block {function block}



decision point

index
data entry point
search and reform point

3. Once you have made the decisions above, you are ready to begin the actual process of writing the program. First, assign any registers that will be required and give marks to the program blocks and subroutines as necessary. Proceed to formulate the proper step to accomplish the program. After the writing, run the program and proceed to remove any errors and make any changes necessary until the program is satisfactory.

4. After you have the program developed to your satisfaction, run it at least once completely through, making a formal record of the results and a printout of the answers if any. Write out in detail the operational procedures needed to run the program, a description of the program and its function, a detailed flow diagram, a code listing of the program and any other information necessary to understanding the program. This material should be well-organized and contain all detail necessary for efficient usage.

5. Organize all of this into a package to be filed for future reference. The following is an example of what the results mught look like for a simple program.

PROGRAM EXAMPLE

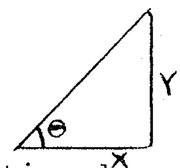
Program Description

This program will make the conversion from polar phaser form to rectangular form. The operator need enter only the magnitude and the angle and the program will print back the horizontal {X} and the vertical {Y} portions of the phaser.

Enter Mag $\angle\theta$

Printout Horiz {X}

Vert {Y}

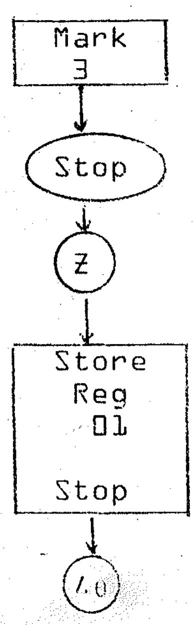



$Z \angle\theta \xrightarrow{\text{conv}} Z \cos\theta + j \sin\theta$ {functional equation}

Operational Procedure

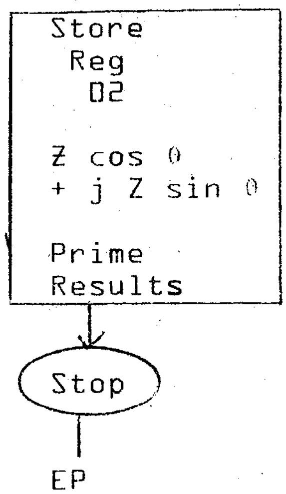
Select card #25 from card file. Put the machine in learn mode and run card through the card reader. Verify that the program is loaded. Verification #427. Key search 3. When the display reads +1, enter magnitude, key GO; +2 enter angle, key GO. Program will print out results.

Flow diagram




Enter Z

Enter $\angle\theta$




Storage Information

Program stored on card #25 {card file}
Number or steps: 28
Verification #427
Print out X→Horiz vaule 6 digits
Y→Vert value 6 digits
Finally, fill out a program description form and file it with the program.

Program Files

1. Card files are one of the two methods that can be used to store programs for use by the Wang calculator. The procedure for entering data on the cards has already been shown and will not be discussed here.

a. When filing cards, you should have a master list which tells basically what each card contains and how to identify it. You will notice that each card has on its edge a place to write a title, a card number, and how many cards are included in the group.

b. It is suggested that all the identification information on the front side be duplicated on the back as well. The reverse side of the card also lends itself well to including an abbreviated set of operational instructions, verification number and register assignments.

c. It has also been found to be extremely useful to have a looseleaf notebook containing a set of program description forms and the operation instructions. A sample of these forms may be found at the end of this section. These have proved to be quite helpful. You may wish to make up your own form or to use the ones provided. We would suggest, however, that whatever form you use, you should have some kind of master list that can be searched out quickly.

2. Cassette tape is the other method of program storage provided for the Wang calculator. It has the advantage of being able to store many programs in a small physical space. The program retrieval system, however, is somewhat more complex and requires the use of a bootstrapping system as a self-designed retrieval method.

a. You should again keep some form of master file of reference material including such items as have been previously mentioned:

> Program description, verification of numbers, operational procedures, etc.

In addition to these, you will find it useful to have a block diagram of the tape showing the position of the program on the tape.

b. Because tape retrieval systems are somewhat more complex than card systems, you should include loading instructions with each program that is loaded on tape.

Summary

As you have probably already noticed, the keeping of program records includes a fair amount of duplication of instructions. While this may seem to be wasted paper, you will find that, especially with tape retieval, the availability of loading and operating instructions without the necessity of actually pulling the file on a particular program is very handy.

It is worthwhile repeating that complete records are extremely important and without them your programs tend to become useless pieces of paper laying on your desk which seldon, if ever, get used.

| REG # | Tape Pos. | Mark Code | Verify | # of Steps | Tape Record General Program Description |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Program
Operation TITLE______________________________________________

| |
|---|
| Program # |
| Number of Steps |
| Tape Number and Position |
| Verification Number |

General Program Description and Function:

Formula or Format:

Use and Application:

Data Register Assignments:

| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |

Operation and Loading:

# XVIII. PROBLEM SUPPLEMENT