1. ArgoxBasic Syntax

ArgoxBasic does **NOT** comply with standard ANSI BASIC. It is the subset of ANSI BASIC and some extension for serial IO. Supposedly, the Interpreter based on this Syntax definition will be employed into ARGOKEE as an universal emulator to support all kinds of BAR CODE printers.

1.1 STRUCTURE OF BASIC

The rules on the structure of the language are as follows:

- **Each BASIC command must appear on a separate line. (except "IF THEN ELSE ", "FOR TO STEP") (" : " mark for separate command is not allowed)**
- **A statement cannot exceed 100 characters (80 characters after BASIC Command) .**
- **A statement must start with a statement number. It is a positive integer.**
- **No two statements shall have the same number.**
- **The statement must be in an ascending order.**
- **Each statement number shall be followed by a BASIC command.**
- **You may use blank space to increase the readability of the program.**
- **An individual BASIC size should not exceed 1024 * 20 bytes.**
- **The total size of GRAPHIC data should not be more than 1024* 40 bytes.**
- **The Maximum statements in an individual BASIC program shall not exceed 500 lines.**
- **The Maximum Var. numbers in an individual BASIC shall not exceed 50.**

1.2 COMMAND INTRODUCTION

1.2.1 #BASIC

The NAME statement must be the 1st statement in the BASIC program to be identified as this program name.

Syntax:

#BASIC "TEST BASIC"

"TEST BASIC" is the NAME of this program.

1.2.2 #GRAPHIC

#GRAPHIC "GRAPHIC NAME " is the prefix for Graphic Object .

Syntax:
#GRAPHIC "GRAPHIC NAME"+ GRAPHIC data

The string within quotation marks is the object NAME for this subsequent graphic data. And there should be no additional data between the quotation mark (right side) and the GRAPHIC data. In other word, the control code such as CR or LF must NOT be inserted between the quotation mark (right side) and GRAPHIC data.

1.2.3 REM

The content of the line following REM is not for execution and as such should be ignored. REM is used purely to enhance the documentation aspect of a program.

Syntax
Line Number REM ?????????

1.2.4 END

Whenever END is used, it must be the last line. Any statement after the END is irrelevant to the interpreter.

Syntax
Line Number   END

1.2.5   LET
Syntax:
***Line number* LET (*variable name*)=(Constants or Variables or expression).**

Example:

```
LET      PartsName$ = P-IV
LET      UnitPrice = 200
LET      Qty= 10
LET      Total   =   UnitPrice * Qty     * (   1 + 0.05 )
```

1.2.6   PRECISE

PRECISE   statement is applied to specify the digit number after **Decimal Point** while performing the Arithmetic calculation.

Syntax
Line Number   PRECISE   N

The N in PRECISE statement is to define the digit number after Decimal Point. The default value for N is 2.

1.2.7   INPUT Statement
INPUT statement will wait for the user to key in the Var. value. By the way, while INPUT statement being executing, user can press " ESC key " to abort running BASIC program.

Syntax
Line number   INPUT   "PROMPT 1" ,Variable   1, "POMPT 2 ",Variable 2………

Example

INPUT A,B
When this command is executed, ArgoxBasic will print "=?" on standard output device (In ArgoKee, this is LCD) and wait to read in a number from the standard input device(In ArgoKee , this is ArgoKee itself). The variable A will be set to this value. After the ENTER KEY (on ArgoKee) pressed , another " = ?" is printed and variable B is set to the value of the next number read from the input device.

INPUT "WHAT IS THE WEIGHT", A, "AND SIZE", B
This is the same as the above command , except that the prompt "=?" is replaced by "WHAT IS THE WEIGHT?" and the second "=?" is replaced by "AND SIZE?".

1.2.8   READ-DATA Statement
The LET and INPUT statement have certain limitations. By a LET statement we can assign one value for each variable and if we have to get 10 values for 10 variables we have to use 10 statements with LET. Also if a value is assigned through a LET statement its value can be changed only by replacing that assignment statement with another assignment statement. If there is a large amount of data to be processed, it is inconvenient to key in the entire data during the execution of the program. In such cases, READ statements are found useful. READ statements will always have a DATA statement along with it.

Syntax

Line number   READ Variable, List....
Line number   DATA Constant, List....

Example of a READ....DATA Statement

```
1          REM PROGRAM EXPLAINING READ....DATA STATEMENT
5          READ X2,Y,Z1,K
9          READ A,B3, C4, L
22         DATA 8,9,13,15,16,51,30,92
30         END
```

When the machine encounters the line number 5 with the READ statement followed by the variables X2,Y,Z1,K it will collect from the DATA statements the values for these variables in the same order. A one-on-one correspondence exists in the READ-DATA statement, i.e. X2=8, Y=9, Z1=13, K=15.
In the next READ statement in line number 9 we have four more variables A,B3,C4,L.These values will follow in the same order and one-on-one correspondence after the earlier READ variables values, i.e., after 8,9,13,15, i.e., A=16, B3=51, C4=30, L=92.

Since all the values in the DATA here are exhausted, another i.e. third, READ statement cannot be used until another DATA statement is included or more DATA is included in the above DATA list in line number 22. A DATA statement can be anywhere in the program but must be someplace before the END statement. It is a normal practice to keep all DATA statements together at the end of the program before END statement so that in case you want to alter any data at the end, it will be very easy.

1.2.9 DATA BLOCK

The BASIC incorporates the contents of all the DATA statements into one single data block. When the READ statement are executed a pointer moves along with data block starting from the first element:

EXAMPLE

```
1       READ X1, Y1, Z1
7       READ N, M, L, K
12      READ A
19      READ X,Y
24      DATA 122
26      DATA-35,49, -101
28      DATA -691,81
30      DATA 8,10.-5,5
40      END
```

DATA BLOCK

Value	Pointer	Variable
122	<---	X1
-35	<---	Y1
49	<---	Z1
-101	<---	N
-691	<---	M
81	<---	L
8	<---	K
10	<---	A
-5	<---	X
5	<---	Y

Now look at the following examples wherein all the programs do exactly the same work.

```
(a)     10      DATA "RAMA","DELHI",100,200,300
        20      READ A$,B$,A, B, C
        30  - - - - -
        40      - - - - -
        - - - - - - - - - - - - - - -
        80      END

(b)     10      READ A$,B$
        20      READ A,B,C
        30      - - - - - -
        40  - - - - - -
```

```
      60      DATA "RAMA","DELHI"
      70      DATA 100,200,300
      100     END

(c)   10      DATA "RAMA","DELHI"
      20      DATA 100
      30      DATA 200
      40      DATA 300
      50      READ A$,B$
      60      READ A,B,C
      - - - - - - - - - - - - -
      100     END

(d)   10      READ A$
      20      READ B$,A,B,C
      30      DATA "RAMA"
      40      DATA "DELHI"
      50      DATA 100,200,300
      - - - - - - - - - - - - -
      100     END
```

In all the above cases, A$ is assigned the value RAMA and B$ is assigned the value DELHI. .All the numeric variables A,B,C are assigned the values 100,200,300 respectively.

So we note that for READ....DATA statement we should follow the subsequent rules:

- There could be any number of READ and DATA statements.
- As many variables are read by READ statement, at least the same number of values should be there in the DATA statement. It may be more. If the number of values in DATA is , then an error message (Fail to read the data in READ command) will be shown on ArgoKee when there are no more data to be read. However, you can use RESTORE command to **reset** the reading POINTER to repeatedly read the data defined in DATA statement.

- The type of variables (i.e., numeric, string) in the READ statements containing many variables.
- Although we normally keep all the DATA statements at the end of the program and before END statement, DATA statement can occur anywhere in the program but before END statement.

1.2.10 RESTORE

The RESTORE statement will reset the pointer to the first value of the DATA statement regardless of the current position of the pointer.

Syntax

Line number RESTORE → Reset the reading pointer for both numeric and string Var.
Line number RESTORE $ → Reset the reading pointer for string Var.
Line number RESTORE * → Reset the reading pointer for numeric Var.

For example,

```
  10  RESTORE
```

The RESTORE statements may be used anywhere in the program, of course, before the END statement.The following example will clearly illustrates the use of RESTORE statement

```
10        READ X,Y
20        DATA 10,12,-7,,3.2, "PANKAJ"
30        RESTORE
40        READ A,B,C,D,N$
50        END
```

When the statement number 10 is executed, the variables X and Y are assigned the values of 10 and 12, the position of the pointer at this stage is shown below:

```
          10        12       -7        3.2  "PANKAJ"
                    ↑ (Pointer)
```

(Position of pointer before the execution of RESTORE statement)
When the statement 30 RESTORE is executed, the pointer is brought back to the first data value of data statement as shown below:

```
10               12                -7                 3.2   "PANKAJ"
↑ ( Pointer)
```

Now, the execution of the statement number 40 against the value to A,B,C,D and N$ will be as shown below:

A	10
B	12
C	-7
D	3.2
N$	PANKAJ

The RESTORE statement can be used in the following forms also:

Statement number RESTORE *
and
Statement number RESTORE $

If the key word RESTORE is followed by asterisk (*), then only numeric pointer is reset as shown in the previous example. However, if the key word RESTORE is followed by a dollar sign ($), then only string pointer is reset to the first string data. The simultaneous use of both asterisk (*) and dollar sign ($) will not be allowed.
For example, if a BASIC program contains the following statements,

```
10      READ A,B,N$,M$
20      - - - - - - -
30      - - - - - - -
40      RESTORE *
50      READ P,Q,A$,B$
60      - - - - - - -
70      - - - - - - -
80      DATA 5,8,"DINESH",UPMA",10,12,"RAJESH","SANJU"
90      END
```

When the statement 10 is executed, A, B are assigned values 5 and 8 whereas the variables N$ and M$ are assigned the values DINESH and UPMA respectively. The execution of statement 40, restores the numeric pointer to the first numeric value in the DATA statement i.e. to the value 5. The execution of the statement 50, assigns the value 5,8, RAJESH and SANJU respectively to the variable P,Q,A$ and B$. The values DINESH and UPMA shall be ignored. Similarly, the execution of

RESTORE$ can also be shown by taking similar examples.

Consider another example

```
10      READ A,B,N$,M$
20      RESTORE $
30      READ P,Q,A$,B$
40      DATA 5,8, "DINESH", "UPMA", 10,12,"RAJESH",SANJU"
50      END
```

Here, the statement RESTORE$ is used. The values assigned to the variable A,B,N$ and M$ are as same as the previous example. However, when statement 20 is executed, the string pointer is reset to the first string data namely “DINESH.” The execution of the statement 30 would assign the values 10 and 12 to the variables P and Q and then the pointer moves back-ward to assign the values DINESH and UPMA to the variables A$ and B$

1.2.11 PRINT

PRINT Statement will pass all the information within the quotation (“) to Printer via the serial port on ArgoKee. The Variables (content) in PRINT statement are also transparently sent to Printer. In other word, it means any information in the PRINT statement will be faithfully sent to Printer without any modifications.
This strategy would maintain ArgoxBasic Interpreter to be independent of any specific Printer Language. By the way, the BASIC Interpreter will not append LF+CR automatically, after all the data in the PRINT statement have been sent to PRINTER. If you want to append CR or LF in this statement, the PRINT statement must manually tail with CHR$(10,13) .

Line number PRINT (Variable) separator (variable) separator or item......

The separation may be comma (,) or semicolon (;) (Refer to 1.2.10.1, 1.2.10.2 for the description about ; and ,)

Syntax1
Line number PRINT Variable

Syntax 2
Line number PRINT "PROMPT 1" ,Variable 1, "POMPT 2 ",Variable 2………

Syntax 3
Line number PRINT CHR$(2),"c0200", CHR$(10,13)
Where CHR$(2) will enforce ArgoxBasic to send the control code,0x02, directly.CHR$(10,13) is Line Feed (LF)and Carriage Return (CR).

1.2.11.1 The Semicolon (;) Control

Program

```
10      LET S =1175.50
20      PRINT "TOTAL SALARY =";S; " RUPEES"
30      END
```

The output of the program 1 will be as

T	O	T	A	L		S	A	L	A	R	Y	=	1	1	7	5	.	5	0		R	U	P	E	E	S

Using semicolon in a print statement, the items are close to each other, and as a result more items can be printed in one line,

1.2.11.2 The Comma (,) Control
100 PRINT "NUM","TEMP","SIZE", REMARK"
110 PRINT 65,-15.56,36,34
If the comma,"," ,is applied as the separators of the arguments in PRINT command. These two lines will be printed as TABULATION COMANND defined.
Since the default TABULATION are : 0,14,28,42,56,70,84,98,112,126 , the printed out message will be:

0	14	28	42	56
Zone 1	Zone 2	Zone 3	Zone 4	Zone 5
NUM 65	TEMP -15.56	SIZE 36	REMARK	 34

1.2.11.3 CHR$( )
This command is invoked in PRINT command for those printer codes

can not be presented within quotation marks. If the arguments in CHR$ function are the numeric between 0~255, this numeric string will be sent to printer after it had been converted to one byte Hexadecimal Value. Otherwise the arguments in CHR$ function will be sent to printer faithfully. Max. 4 arguments are allowed in CHR$ (). The Argument s should be separated with Comma mark.

e.g.: If you want to send LF & CR to printer, CHR$(10, 13) can be appended in the
PRINT statement. If you want to send the QUOTATION MARK(") to printer you can make the BASIC statement as: PRINT CAR$("),"1234566",CHR$(").
This Statement will send "1234567" to Printer included ".

1.2.11.4 TAB ( )
TAB (n) moves the printer head to the n th column and print out of any data starts from that column.
e.g: Line number PRINT TAB (n); X ➔ X is printed out from column n.

1.2.12 GRAPH( )
GRAPH (Graphic Object Name) in PRINT command will let the BASIC Interpreter send out the GRAPHIC data which had been saved in the key board named as Graphic Object Name.

1.2.12 PRINTF

PRINTF Statement will show the information within the quotation (") on the LCD of ArgoKee. The Variables (content) in PRINTF statement will be shown on LCD too. The BASIC INTERPRETER will stay at this statement till " ENTER" or "ESC" key on ArgoKee pressed by user. It implies the message shown on LCD will be sustained before ENTER or ESC pressed too.

Syntax:
Line number PRINTF "xxxxxxxx" → xxxxxx will be shown on LCD faithfully
Line number PRINTF VAR NAME → The content of VAR will be shown on LCD.

Only a single Argument is allowed.

1.2.13 TABULATION
This Command is to define the tags of tabulation while the arguments in PRINT statement being separated with Comma (,).

Syntax
Line number TABULATION column 1 , column 2 ,……., column n

The max. column n is 10.
The default Tab Tags are: { 0,14,28,42,56,70,84,98,112,126 };

1.2.14 Unconditional Go To Statement; GOTO
GOTO statement is used to transfer control from a statement, say S1 to another statement, say S2, generally, S2 does not follow S1 immediately in sequence.

Syntax: Line number GO TO n
n is the line number of the statement where control will be transferred.

1.2.15 The Branching Statement ; IF... THEN

Syntax: Line number IF (EVALUATION) THEN (line number)
If (EVALUATION) is TRUE , the Line number after THEN will be executed. Otherwise the next line will be executed.

The IF...THEN is a decision making statement, depending upon the decision, it can change the order of execution. The EVALUATIONs could consist of several **RELATIONAL EXPRESSION** which are linked with **OR** or **AND** . The Max. RELATIONAL EXPRESSIONs in one IF.. THEN .. ELSE are 5.

Program
```
10        LET X = 1
20        LET Y = X*X
30        PRINT Y
40        LET X = X+1
50        IF X> = 31 THEN 70
60        GO TO 20
70        END
```

In Line 50, if the evaluation; (X>= 31), is true , line 70 will be executed. Otherwise Line 60 will be executed.

1.2.16 The Branching Statement ; IF...THEN...ELSE

Syntax:
Line number IF (EVALUATIONs) THEN (line number) ELSE (line number)

The IF...THEN...ELSE statement is a decision-making statement as it decides the path of the program. It helps in making comparisons and testing whether a condition is true or false. IF always followed by a valid BASIC condition or expression. If the condition is found true then the line number after THEN is performed otherwise line number after ELSE is performed. The EVALUATIONs could consist of several **RELATIONAL EXPRESSION** which are linked with **OR** or **AND**. The Max. RELATIONAL EXPRESSIONs in one IF.. THEN .. ELSE are 5.

EXAMPLE 4
Problem 1

Ages of different students appearing in the Board examination are taken. If the age is below 17 the student is not eligible, otherwise he can appear in the Board examination. We are asked to write a program for this problem.

Program 1
```
10     INPUT "AGE"; A
20     IF A>= 17 THEN 30 ELSE 50
```

```
30      PRINT “WELCOME FOR BOARD EXAMINATION”
40      GO TO 60
50      PRINT “YOU ARE NOT ELIGIBLE FOR BOARD EXAM.”
60      INPUT  “WANT TO INPUT AGAIN (Y/N)”; Y$
70      IF Y$ = “Y” THEN 10
80      END
```

The line number 10 will cause the message in screen AGE? We input the age through the ArgoKee, say 18. Line number 20 tests whether A>17 or not. Since A=18>17 line number 30 is executed.
Line number 30 prints WELCOME FOR BOARD
Line number 40 causes the control to pass to line number 60.
Line number 60 causes the message WANT TO INPUT AGAIN (Y/N)? We input either Y or N. In line number 70 if input is Y then control goes to line number 10. Otherwise if input is N then control goes to the line number 80 i.e. END. Now if in line number 70 we input Y then control will pass again line number 10, we give another age, say 13 In line number 20 value of A (i.e. 13) is not greater than 17, therefore ELSE part will be executed and control will go to line number 50. Line number 50 will print YOU ARE NOT ELIGIBLE FOR BOARD EXAM. Then line number 60 as before will be executed. In this way a large number of students age can be tested. When we want to stop we should input N in line 60 for Y$.

1.2.17 The Looping Statement; FOR-TO...NEXT

We have already seen that a loop can be built in BASIC by using the IF-THEN and GOTO statements. When it is known in advance how many times the loop must be repeated the statement FOR-TO...NEXT is the most effective statement.

A loop is built up by FOR-TO and ended by NEXT.
Syntax:

```
         ┌──▶ Line number FOR I = M TO N STEP J
Loop     │     -  -  -  -  -  -  -  -
Loop     │     -  -  -  -  -  -  -  -
         └──▶ Line number NEXT I
```

The numeric variable name following FOR is called the **control variable** or **loop variable**. M and N are numeric variable or constants (immediate value) where M gives the initial or starting value of the loop and N gives the final value, J followed by keyword STEP gives the increment in M till N is reached. The increment can be negative also.

When M,N,J are numeric variable names, their numeric values should be assigned before the starting of the loop, i.e. before coming to FOR-TO statement. The companied command, NEXT, should have the same **control variable** followed by it. Thus one loop can be started with FOR-TO and ended with NEXT. Only **1 single character** for the mnemonic symbol of **control Var**. is allowed. The control Var. value should be greater than **“-32768”** and less than **“32767”**.

Inside one FOR-TO ...NEXT loop there can be more FOR-TO...NEXT loop. But once a FOR-TO...NEXT is inside another FOR-TO...NEXT, it should remain completely inside the former loop. Such FOR-TO...NEXT loops are called Nested loops. In the absence of the STEP clause, the increment is assumed to be 1.

EXAMPLE 5
Problem 1
Suppose we want to print the output in the following format:

```
* * * * *
* * * *
* * *
* *
*
```

Program 1

```
10      FOR S=5 TO 1 STEP-1
20      FOR X=1 TO S
30      PRINT "*" ;
40      NEXT X
50      PRINT
60      NEXT S
70      END
```

This program contains two loops: the outer loop is from the line number 10 to 60 and inner loop from line numbers 20 to 40. In line number 10, initially S is assigned a value 5. Since the value of S is greater than 1, control is transferred to line number 20, which causes the inner loop to execute 5 times resulting into printing of 5 stars (*) in one row. The statement at line number 50 will transfer the printer control to the beginning of the next line. When line 60 is encountered, the control goes back to line number 10. Now the value of S becomes 4 and once again the inner loop is executed 4 times resulting in printing of 4 (*) stars in second row. This process will continue till the value of S becomes 1. After that it will come to an end.

2.How to run a "ArgoxBasic Program" on ArgoKee

2.1 Edit BASIC program

1. Choose a suitable TEXT editors, like NotePad, WordPad (Pure Text*.txt mode only. **Don't choose WinWord Mode**), Visual C++, PEII … etc. Some popular word editors, like WinWord , WordPerfect.. , are **NOT** suitable to be chosen as your BASIC editor.
2. Then follow the ArgoBasic Syntax which have been described in chapter 1. to edit a BASIC program. (**#BASIC "Program Name " must be prefixed on your BASIC program**)
3. Save it in the PC with file name you like.

2.2 Download the BASIC program into ArgoKee

1. Press "S" on ArgoKee then scroll it (Press Left or Right arrow) to EMULATION option. "BASIC" item should be selected as the emulation Mode.
2. Set up the baud rate for ArgoKee to be consistent with what the Hyper Terminal had been (or will be) set.
3. Press "D" on ArgoKee to let it enter "Download Mode ".
4. Employ "Hyper Terminal (select "TEXT file") to download your ArgoxBasic program into ArgoKee.
5. If any error is found in the ArgoxBasic file, an error message will be shown on LCD. You can refer the Error Code listed below to find out more information. Please

press ENTER key to find out more errors in the downloaded BASIC program. If all the BASIC errors had been shown, the ArgoKee will return to Stand-by mode.

Note : In the download mode, the Syntax checking is simplified. Most of the Syntaxs will be checked later when this program is running.

2.3 Run the BASIC program on ArgoKee

1.Press "P" in stand by mode (when " BASIC MODE CHOICE:" is shown on LCD").
2.Press DOWN/UP Arrow to find the BASIC program you want it to be Run. (The program name, shown on LCD is the name field followed #BASIC command.)
3.When you get the desired BASIC program, Press " ENTER" to run it.
4.Then the BASIC program you selected will be run on ArgoKee till the END command in the BASIC program is executed.
5.If there are any syntax errors being checked, the Line No &Error code will be shown on LCD.
6.By the way, the running BASIC program can be unconditionally aborted, if ESC key is entered when INPUT command in the running BASIC program being executed.

2.4 How to Save a "GRAPHIC OBJECT " in ArgoKee?

2.4.1 Edit a Prefix file

1. Employ a Text Editor to edit a simple statement as follows, #GRAPHIC " Graphic Object Name". Then you might save it as the file name,"G_Prefix.txt".

<NOTE >
Any data or control word are NOT allowed to be appended in the right quotation mark of Graphic Object Name following . Therefore the "Text Editor" will automatically tail with " end of file mark" , "Carriage Return" , "Line Feed" .. , are not suitable to be employed to edit this Prefix File. Note Pad in Microsoft Window are a suitable candidate for this requirement. .

2.4.2 Download a "GARPHIC OBJECT" to ArgoKee

1. Press "S" on ArgoKee then scroll it (Pess Left or Right arrow) to EMULATION option. "BASIC" item should be selected as the emulation Mode.
2. Set up the baud rate for PC port in ArgoKee(19200 bps is recommended) .
3. Issue " MODE com port: baud rate , n,8,1" on MS DOS prompt to have the baud rate be consistent with the baud rate you had set on ArgoKee. (e.g. mode com1:19200,n,8,1 ➔ set 19200 bps on PC COM1 port)
4. Press "D" on ArgoKee to let it enter "Download Mode".
5. Issue " COPY G_Prefix.txt + graphic file COM1:/B" on PC MS DOS prompt. The format of graphic file you downloaded to ArgoKee should be able to be accepted by the printer connected with ArgoKee, or some unexpected result and errors will be encountered in printing.
6. After had been saved in ArgoKee , this Graphic Object will be sent to printer while the BASIC statement, "PRINT GRAPH (Graphic Object Name) ", being executed in the running BASIC program.

2.5 Error code

Error code	Description
01	The Prefix Command, #BASIC ,#GRAPHIC are found .
02	Nothing found after #BASIC
03	The Program Name is not specified. The name should be companied by #BASIC command .
04	Illegal"BASIC Program Name",Maybe Left ",Right" missed
05	Unknown BASIC command found .
06	The SPACE between 2 field is too long.
07	Too many characters in 1 statement.
08	"line number" in the field of statement missed
09	Line number are not in ascending order
10	Unknown BASIC operator
11	Miss Line Number
12	Miss Command in on statement
13	Statements in this program are too much
14	Can not find END command till all data processe .
15	Line Number for GOTO command not found in this program or Not specified.
16	The Label defined is too long.
17	The Var. defined is too long.
18	The Operand extracted is too long .
19	No Operand found in the command statement
20	There are too many Vars. in this BASIC
21	The Control VAr name is Illegal(over 1 Char)
22	Void Control Var., Initial Var missed.
23	"TO" command is not found in FOR statement
24	Miss "End value" in For statement
25	There are Too many Control Var. (Too many NEST loop) .
26	The Control Var. redefined .
27	The End name for LOOP end value is void .
28	Void Initial Value in For LOOP Control Var.
29	Not "STEP" command in step field
30	STEP value followed with STEP are not found
32	Out of Control Var range, -32768~32767
33	This Var. can not be found
34	The Evaluation Pair in IF statement is too many
35	The Arguments in IF statement are not comparable
36	The Result of Arithmetic operation is overflow
37	The Arithmetic expression is illegal
38	Too many "Arithmetic arguments" in a statement
39	The parameter followed with PRECISE command is void
40	Too many Tabulation Tag defined in TAB command
41	Illegal argument defined in TAB function
42	Not all numeric when the Var Name is for numeric only
43	Fail to read the data in READ command
44	Data in DATA statement are illegal
45	BAD syntax in READ statement
46	Syntax error in Restore statement
47	The Program size is overflow

3. The Sample Code (Argox Basic)

3.1 Calculator

```
#BASIC         "CACULATOR"
100  PRINTF   "BASIC DEMO PROGRAM   CALCULATOR   "
105  PRECISE  4                                  → The digit dots is 4
110  INPUT    "Enter a numeric",First            → Wait for User to enter the first Argument
120  INPUT    "Enter + - x / ",OP$               → Wait for the user to enter the Arith. Operation,
130  INPUT    "Enter a numeric",Second           → Wait  for User to enter the second argument
160  IF    OP$ = "x" THEN  220                   → If x → Perform “ Multiply”
170  IF    OP$ = "/" THEN  240                   → If / → Perform  Division
180  IF    OP$ = "+" THEN  260                   → If + → Perform Addition
190  IF    OP$ = "-" THEN  280                   → If  - → Perform Subtraction
200  PRINTF "PLS.ENTER AGAIN! +-x/ for Arith.OP" → Not +-x/→ Aske user to enter again
210  GOTO   110
220  LET    RESULT = First *  Second
230  GOTO   290
240  LET    RESULT = First / Second
250  GOTO   290
260  LET    RESULT = First + Second
270  GOTO   290
280  LET    RESULT = First - Second
290  PRINTF  RESULT
300  INPUT   "Press Q to Quit ", quit$            → Ask user if he or she would like to quit from this
                                                    BASIC program or not
310  IF quit $ ="Q"OR quit $= "q" THEN 400 ELSE 110→ If Q or q entered → Quit
400  END
```

3.2 Print out Labels from PPLA Printer

```
 #BASIC   "PPLA"
110   INPUT "Label Count=?",cnt$                     → Input the label amount for one parts
120   LET     times = 1000                           → How many kind of parts will you entered?
130   FOR   I = times  TO  1  STEP  -1
140   INPUT "Parts Name=?";Parts$                    → Input the Parts name
150   INPUT "PLS. Enter PCS",pcs                     → The quantity for this pars ?
160   INPUT "Date=? dd/mm/yy",date$                  →  Date  entry
210   PRINT CHR$(2);"L";CHR$(13,10)                  → enter Label formatting command mode
220   PINT "H12";CHR$(13,10)                         → Heat  ratio for TPH
230   PRINT "PC";CHR$(13,10)                         → Print Speed
240   PRINT "D11";CHR$(13,10)                             → Width &Height Dot size
250   PRINT "131100000600046";pcs;" PCS";CHR$(13,10)      → send pcs to printer in Text form
260   PRINT "1A4202500500130";pcs;CHR$(13,10)             → send pcs to printer in BAR code form
270   PRINT "121100000600240";date$;CHR$(13,10).          → date
271   280   PRINT "131100000200046";Parts$;CHR$(13,10)    → the Parts’ name in Text
290   PRINT "1A4202500100130"Parts$;CHR$(13,10)           → the Parts’ name in Bar Code
300   PRINT "^01";CHR$(13,10)
310   PRINT "Q";cnt$;CHR$(13,10)                           →  Define the label amount to be
printed out
320   PRINT "E";CHR$(13,10)                                → Command PRINT
330   NEXT  I                                              → Go back 140 if I > = 1
340   END
```

3.3 Print out Labels from PPLB Printer

3.3.1 Case I

```
  #BASIC         "POS"
10    REM     DATA BASE for product name & unit PRICE
100   DATA   "GF100",100.23,"GF200",105.12,"GF300",200
110   DATA   "X1000",300.12,"X2000",499.99,"X3000",799.99,"X4000",1200
120   DATA   "OS-214ZIP",199.1,"A-200",399,"G-6000",550,"TP-180",99.9
130   DATA   "PS-II", 299.99,"X-BOX",299.99
140   DATA   "End$Flag",0                          → A Flag to identify there are no more dates
150   LET    TRANS_I = 0                           → Reset Transaction number
160   INPUT  "Enter Label count",Count             → Input  how many labels for each and
every label entry
170   REM    PRINTF  Count
180   LET     Cnt = Count
190   RESTORE                                       → Reset Data/String Index in this small
Data Base
270   INPUT  "Enter Product Name", PRODUCT$         → Input the Product Name
280   INPUT  "Enter Quantity", Qty                  → Input the quantity for this product
290   READ   Match$                                 → Read a Product name in data base

300   READ   Price                                  → And its corresponded PRICE
310   IF     Match$ = "End$Flag" THEN 330  ELSE 350  → If no more data in data base->
                                                       request to reenter
330   PRINTF  "PRODUCT NOT found!   Enter for Next"
340   GOTO   270
350   IF     Match$ = PRODUCT$  THEN 360  ELSE  290  → If Product entered by user is equal
                                                        to any one in data base
                                                     → Print It out in PPLB language
360   PRINT  "N",CHR$(13,10)                          → Clear Image buffer in PPLB printer
370   PRINT  "q592";CHR$(13,10)                       → Label width to be 592 dots (3 inch)
380   PRINT  "Q196,24";CHR$(13,10);"JB";CHR$(13,10)   → Form Length=192
                                                        dots,gapdots=24
390   PRINT  "D9";CHR$(13,10);"S2";CHR$(13,10)        → Print Density = 9 ( Heat)
400   PRINT  "O";CHR$(13,10)                          → Disable all options
410   PRINT  "A05,180,3,4,1,1,R,";CHR$(");"E-Mall";CHR$(",13,10)→Print E-MALL ( reverse)
                                                                 on the Top
420   PRINT  "A35,180,3,4,1,1,R,";CHR$(");"RECEIPT";CHR$(",13,10) →  Print  “RECEIPT “
430   PRINT  "A75,180,3,4,1,1,N,";CHR$(");Match$;CHR$(",13,10)     → Print the Product
                                                                     Name
450   PRINT  "A115,180,3,4,1,1,N,";CHR$(");"  Q'ty= ";Qty;CHR$(",13,10) ..→ Print Q’ty  text,
                                                                        and its Value
460   PRINT  "A155,180,3,4,1,1,N,";CHR$(");" Unit$=";CHR$(",13,10)  →  Print  “Unit$”
465   PRINT  "A185,150,3,4,1,1,N,";CHR$(");"$";Price;CHR$(",13,10)  →  Print the price
470   LET    Total = Price * Qty                                   → To get the total result
480   PRINT  "A215,180,3,4,1,1,N,";CHR$(");"Total=";CHR$(",13,10)   →  Print  “Total=” text
490   PRINT  "A245,180,3,4,1,1,R,";CHR$(");"$";Total;CHR$(",13,10)  →  Print  Total value
500   PRINT  "B350,4,0,3,2,4,51,B,";CHR$(");TRANS_I;CHR$(",13,10) → Print out the Bar code of
                                                                    Transaction number
510   PRINT  "P1,1";CHR$(13,10)          → Print out the image buffer saved by  the above PRINT
                                             commands
520   LET    Cnt = Cnt -1                → Decrease Count
530   LET    TRANS_I = TRANS_I + 1 → Increase Transaction Number
```

```
540   IF     Cnt  = 0  THEN  180 ELSE 360     ➔ If Cnt ==0 → Wait another input, otherwise
                                                   Print it again
550   END
```

3.3.2 Case II

```
#BASIC          "POS"
10    REM     DATA BASE for product name & unit PRICE
100   DATA   "GF100",100.23,"GF200",105.12,"GF300",200
105   DATA   "X1000",300.12,"X2000",499.99,"X3000",799.99,"X4000",1200
110   ATA   "OS-214ZIP",199.1,"A-200",399,"G-6000",550,"TP-180",99.9
120   DATA   "PS-II", 299.99,"X-BOX",299.99
130   DATA   "End$Flag",0
140   LET    TRANS_I = 0                   ➔ Reset Transaction number
160   LET    Base = 0                      ➔ Reset Base position to print
165   LET    Total = 0                     ➔ Reset Total  Money required
170   INPUT  "Enter Label count",Count     ➔ Input how many labels for every label entry
180   REM    PRINTF Count                  ➔ Commented. PRINTF could help you debugging
190   RESTORE                              ➔ Reset Data/String Index in this small Data Base
200   INPUT  "Enter Product Name", PRODUCT$     ➔ Invoke& Wait user to enter Product
                                                   name
210   INPUT  "Enter Quantity", Qty          ➔ Invoke& Wait user to enter the Q'ty for this product
220   READ   Match$                         ➔ Read a Product name in data base
230   READ   Price                          ➔ And its companied Price
240   IF Match$ = "End$Flag" THEN 250 ELSE 270 ➔ A Flag to identify there are no more data
250   PRINTF  "PRODUCT NOT found!   Enter for Next"
260   GOTO   160
270   IF Match$ = PRODUCT$  THEN 280 ELSE 220  ➔ If  Not matched -> Read next Product
                                                   in Data base
280   IF     Base = 0  THEN  290 ELSE  380 ➔  If 1st time → Send  out the data  in Line 290
                                              ~370 to Printer
290   PRINT  "N",CHR$(13,10)                ➔ Clear Image buffer in PPLB  printer
300   PRINT  "q592";CHR$(13,10)
310   PRINT  "Q196,24";CHR$(13,10);"JB";CHR$(13,10)
320   PRINT  "D9";CHR$(13,10);"S2";CHR$(13,10)
330   PRINT  "O";CHR$(13,10)
340   LET    Row = Base                     ➔ To get the Row Position
350   PRINT  "A";Row;",180,3,4,1,1,R,";CHR$(");"E-MART";CHR$(",13,10) ➔ Print E-MAR as LOGO ( Rotate
                                                        270)
360   LET    Row = Base + 25
370   PRINT  "A";Row;",180,3,4,1,1,R,";CHR$(");"RECEIPT";CHR$(",13,10)  ➔ Print RECEIPT
(270 )
380   LET    Row = Base +65
390   PRINT  "A";Row;",180,3,4,1,1,N,";CHR$(");Match$;CHR$(",13,10)  ➔ Current Product
                                                          Name (270 )
400   LET    Row = Base + 90
410   PRINT  "A";Row;",172,3,4,1,1,N,";CHR$(");"Q'ty=";Qty;CHR$(",13,10)  ➔ Print Q'ty Text ,
                                                             and its value
420   LET    Row = Base + 115
430   PRINT  "A";Row;",172,3,4,1,1,N,";CHR$(");"Unit$=";CHR$(",13,10)     ➔ Print Unit$= Text
440   LET    Row = Base + 140
450   PRINT   "A";Row;",172,3,4,1,1,N,";CHR$("); Price;CHR$(",13,10)       .➔ Print   Price
460   INPUT   "Press P to Print",Over$         ➔ Ask user to Print out or Not
```

```
470   LET     Base = Row
480   LET    Total = Price * Qty+ Total
490   IF      Over$ = "P"  OR  Over$ ="p" THEN  500  ELSE 190 ➔ If user Press  "P" in Line 460
                                                                  -> Print It .
500   PRINT  "B350,4,0,3,2,4,51,B,";CHR$(");TRANS_I;CHR$(",13,10) ➔Print out the  BAR Code of
                                                                  Transaction number
510   PRINT  "A520,180,3,4,1,1,N,";CHR$(");"Total=";CHR$(",13,10)        ➔ Print Total text
520   PRINT  "A560,180,3,4,1,1,R,";CHR$(");Total;CHR$(",13,10)           ➔ Print Total Value
530   PRINT  "P";Count;CHR$(13,10) ➔ Actually command PRINTER to Print out data.  Count is the Var set
                                      by user
540   LET    TRANS_I = TRANS_I + 1      ➔ Increase Transaction number
550   GOTO   160
560   END
```

3.4 Print out a Graphic Object to PPLB printer

```
#BASIC          "PrnGraph"
100     PRINT  "GK";CHR$(");"LOGO1";CHR$(",13,10)  ➔ Delete the graphic object named by
                                                      "LOGO1"
110     PRINT  "GM";CHR$(");"LOGO1";CHR$(");"7838";CHR$(10)  ➔ Store LOGO1 object ( file
                                                                size =7838 byte)
120     PRINT  GRAPH(LOGO1)                ➔  Sent LOGO1 object in ArgoKee to Printer
130     PRINT  "N",CHR$(13,10)             ➔  Clear Image Buffer in Printer
140     PRINT  "q592";CHR$(13,10)          ➔ Label Width , 592 pixel
150     PRINT  "Q200,24";CHR$(13,10);"JB";CHR$(13,10)      ➔ Length = 200 pixel
160     PRINT  "D9";CHR$(13,10);"S2";CHR$(13,10)           ➔ Dark level = 9
170     PRINT  "O";CHR$(13,10)             ➔ Disable all options
180     PRINT  "A100,200,0,2,1,1,N,";CHR$(");"TEST basic graphic";CHR$(",13,10)
190     PRINT  "GG100,200,";CHR$(");"LOGO1";CHR$(",13,10)      ➔ Print out LOGO1 object
200     PRINT  "P1,1";CHR$(13,10)
210     END
```