GRiDBASIC UPGRADE NOTES: VERSION 3.0.0

June 20, 1983

GRiDBASIC UPGRADE NOTES: VERSION 3.0.0

These notes document the new GRiDBASIC Options form and five new commands. NOTE: We have purposely printed these notes on one side of each page to make pages easy to include in your manual.

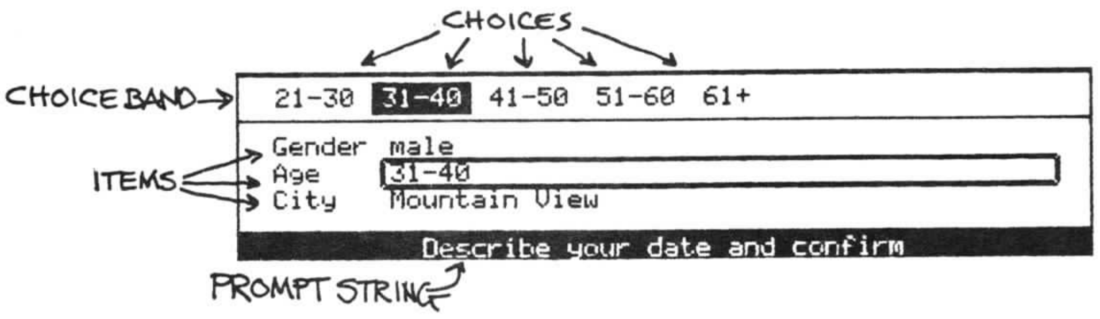
NEW GRiDBASIC COMMANDS

The new GRiDBASIC commands are:

- DOFORM
- FORMCHOICE
- FORMCHOICE$
- HOME
- INPUT USING

The first three concern themselves with creating and using Navigator-like forms in your own BASIC programs. INPUT USING lets you offer users a default input that they can modify. HOME clears the screen.

In discussing forms, keep in mind what we mean by the words "item" and "choice." An item contains one or more choices. An item name always rests on the left side of a form. You move the rectangle over the choices for each item and see the choices displayed in the choice bar at the top of the form. See Figure 1 below.



Gender male
Age 31-40
City Mountain View

Describe your date and confirm

Figure 1. Parts of a Form Identified

THE GRiDBASIC OPTIONS FORM AND LINE NUMBERING

Once you have invoked GRiDBASIC, you get the GRiDBASIC Options form by pressing CODE-O. The Options form has line renumbering and typeface choices.

With the line numbering option, you can select the starting line number for the current program and the increments separating each line. The default setting is "1000" as the first line number with increments by 100 thereafter.

To start the current program with line number 100 and increase line numbers by increments of 10, you would set "First line number" to 100 and "Line number increment" to 10.

You can put old line numbers under a new numbering system by first setting the new start and increments. Then renumber the current program by doing a CODE-? and confirm.

NOTE: GRiDBASIC does not save your renumbering and typeface choices when you quit the program; it only saves actual line numbers and their statements. You must reset these numbering criteria (and typefaces) each time you return to the program.

DOFORM

This function draws a standard GRiD form. These forms are excellent for offering choices within programs and for gathering data in standard formats. Like DOMENU, you can put this function in your programs to give them a look and feel that integrates well with GRiD software.

FORMAT

```
stringVariable = DOFORM (promptString, formString)
```

NOTES

DOFORM takes two arguments, "promptString" and "formString." These arguments can take one of two forms: A string enclosed in quotation marks or a string variable. The formString variable includes the names of the form's items and all the choices that follow these items. The promptString is simply the message that prompts the user to some action ("Enter data and confirm"). It appears on the bottom (inverse video) line of the screen.

The formstring has two delimiter characters (the tilde (~) and the bar (¦)) and two operator characters (the at sign (@) and the caret (^). The tilde separates choices from item names and from other choices. The bar indicates the end of a list of choices. For example, in the formString

```
1000 MyForm$="Sandwich~Ham~Cheese~BLT¦Drink~Cola~Orange~Grape¦"
```

"Sandwich" (The system assumes the first word in the string is an item) and "Drink" are the items. Likewise, the choices under Sandwich are Ham, Cheese, and BLT; those under Drink are Cola, Orange, and Grape.

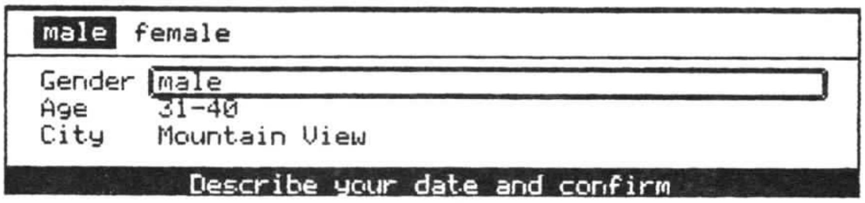
The two operator characters modify the way a form displays choices. Placing the "at" sign (@) before a choice causes that choice to appear as the default choice for its item. ("Defaults" are those settings that the form displays when it first appears.)

The caret (^) makes a blank that the user can fill in and modify (before confirming). This makes it like an INPUT statement. You can also put the blank in along with a choice as in the example below. Thus the user can confirm the default city (Mountain View) or erase the default choice and type in a city name.

In the example, the "at" sign makes "31-40" the default, even though "31-40" is not the first age range on the list following the Age item.

EXAMPLE

```
1000 DateForm$="Gender~male~female¦
 Age~21-30~@31-40~41-50~51-60~61+¦
 City~^Mountain View¦
1100 Prompt$="Describe your date and confirm
1200 YourChoice$=DOFORM(Prompt$,DateForm$)
1300 PRINT YourChoice$
1400 END
```



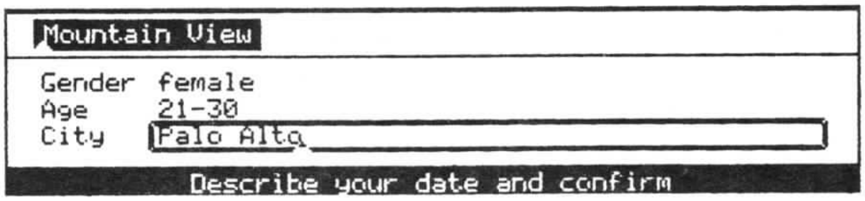
```
male  female

Gender  male
Age     31-40
City    Mountain View

          Describe your date and confirm
```

Figure 2. The Form Created by the Example

This example prints a form (shown in Figure 2) and when confirmed, it displays the user's choices against the structure given in the string variable, DateForm$ (see Figure 3).

Line 1000 gives the formString and line 1100 the promptString.

Line 1200 executes these two strings as arguments to DOFORM. Line 1300 displays your input to the form (albeit in a rather crude manner). For an clean and easy way to extract the input from its form, see the FORMCHOICE$ command below.



```
Mountain View

Gender  female
Age     21-30
City    Palo Alto

          Describe your date and confirm
```

```
Gender~male~@female~|Age~@21-30~31-40~41-50~51-60~61
+~|City~@^Palo Alto~|
```

Figure 3. A Filled out Form and Resulting Data

FORMCHOICE

FORMAT

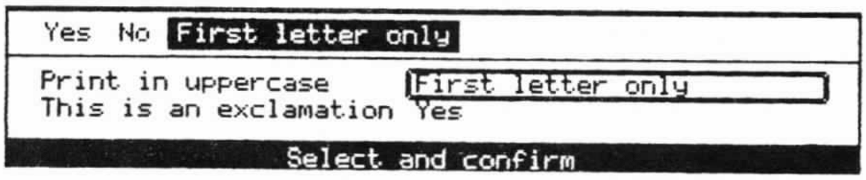
```
variable = FORMCHOICE(formString, itemNumber)
```

NOTES

The FORMCHOICE function takes the name of a form string and an item number and returns the number of the choice made for that item and in that form. As such, its purpose is to let your program act on the data gained from the form. You can use these choice numbers in ON GOTO and ON GOSUB statements to point to appropriate program actions.

EXAMPLE

```
1000 Case$="Print in uppercase~Yes~No~@First letter only|This is an
exclamation~Yes~No|"
1100 Prompt$="Select and confirm"
1200 NewForm$=DOFORM(Prompt$,Case$)
1300 ON FORMCHOICE(NewForm$,1) GOSUB 1600, 1700, 1800
1400 IF FORMCHOICE(NewForm$,2)=1 THEN PRINT "!" ELSE PRINT "."
1500 END
1600 PRINT "HERE IT IS";: RETURN
1700 PRINT "here it is";: RETURN
1800 PRINT "Here it is";: RETURN
```



```
Yes  No  First letter only

Print in uppercase       First letter only
This is an exclamation   Yes

Select and confirm
```

Figure 4. The FORMCHOICE Example Form

This example generates the form shown in Figure 4. Line 1300 generates a 1 for "Yes," a 2 for "no" and a 3 for "First letter only". These numbers then produce the desired result through ON GOSUB.

FORMCHOICE$

FORMAT

```
stringVariable = FORMCHOICE$(formString, itemNumber)
```

NOTES

The FORMCHOICE$ function returns the choices made under a particular item. As the FORMAT syntax above shows, you can assign these data to a string variable. You can use FORMCHOICE$ to get data from the form, usually with the purpose of writing this data to some device (like the screen) or file. Other string functions (like LEN, MID$, etc.) also work on FORMCHOICE$.

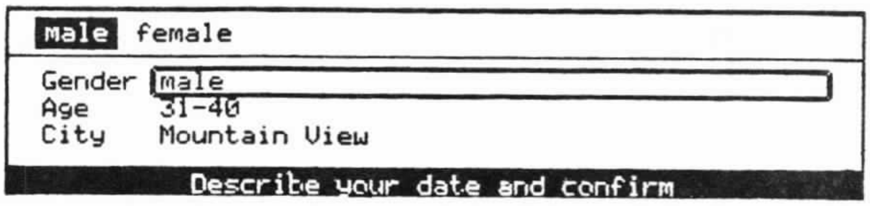
The function takes two arguments -- the form string and the item number. The form string should be the one that includes the item under which the desired choice exists.

In the example below, we use a number variable instead of a number. This way, the FOR NEXT loop serially inserts each number as an argument, so that we find out all three pieces of data. The results of a FORMCHOICE operation go to a string variable.

EXAMPLE

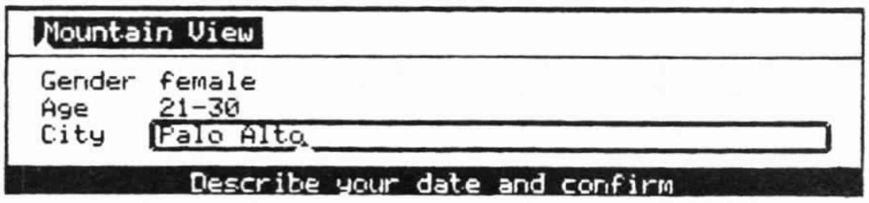
```
1000 DIM Info$(3)
1100 DateForm$="Gender~male~female|
Age~21-30~@31-40~41-50~51-60~61+|
City~^|
1200 Prompt$="Describe your date and confirm
1300 Whee$=DOFORM(Prompt$,DateForm$)
1400 FOR N=1 TO 3
1500 Info$(N) = FORMCHOICE$(Whee$,N)
1600 NEXT N
1700 PRINT "Your date is with a ";Info$(1);" from "; Info$(3): PRINT
"in the age range of ";Info$(2)
1800 END
```

In this example, an array collects the data from FORMCHOICE$. We use the array's individual subscripts to deposit the results in a sentence. Figure 5 gives the program's form; see Figure 6 for the results.



```
male  female

Gender  male
Age     31-40
City    Mountain View

        Describe your date and confirm
```

Figure 5. The Form created by the FORMCHOICE$ Program



```
Mountain View

Gender  female
Age     21-30
City    Palo Alto

        Describe your date and confirm
```

```
Your date is with a female from Palo Alto
in the age range of 21-30
```

Figure 6. A Filled out Form and the resulting FORMCHOICE$ String

HOME

FORMAT

```
HOME
```

NOTES

The HOME command sends the cursor to the upper left corner of the screen. In doing this, it completely clears the screen. This is an easy way to clean up a screen between input and printout routines.

EXAMPLE

```
1000 PRINT "This is the first line"
1100 FOR X=1 TO 100: NEXT X
1200 HOME
1300 PRINT "And this is a second line."
1400 END
```

To see the difference between a screen cleared by HOME and one without this statement, first run the program above as is. You'll see HOME in operation. Then put a single quote in front of HOME (to turn it into a non-executing "remark") and run the program once again.

INPUT USING

FORMAT

```
INPUT [;] ["promptString"] {;|,} [USING stringExpression] {;|,}
variableList
```

NOTES

The INPUT USING statement places an item before the user that the user can either confirm or edit. This item is the default answer to the prompt for input. In the example below, "10" is the default response to the prompt "Which numeric base." Confirming leaves "10" as the answer. A user could change this to any base -- binary, octal, hexadecimal, etc.

A syntactical note: You can place either a semicolon or a comma after the stringExpression. It doesn't matter which punctuation mark, because your choice has no bearing on format. The mark only serves to separate the stringExpression from variableList

EXAMPLE

```
1000 INPUT "Which numeric base" USING "10",A;b
1100 PRINT "Your input was ";A
1200 PRINT
1300 INPUT "Change date or confirm" USING Date$, A$
1400 PRINT "Your date is ";A$
1500 END
```