1.10 Control Objects

So far, while working with the Windows environment, you must be familiar with various controls that make accepting user choices and displaying data easy and convenient. Control objects help with input by directing the program flow and simplifying data entry. They supply or request information and can designate, confirm or cancel actions.

Some control objects commonly seen are **Push Buttons** for OK , CANCEL in a particular dialog box, **Lists** for selecting a file type while opening a new file etc. The various controls objects are :

- Check Boxes
- Lists
- Popups
- Push buttons
- Radio buttons
- Spinners
- Text editing regions

The above control objects can be created using a @..GET command in a FoxPro program.

The basic syntax to create any of the above control objects is :

```
@ <row, col>
GET <memvar> | <field>
FUNCTION <expC1> | PICTURE <expC2>
  [FONT <expC3> [,<expN1>]]
  [STYLE <expC4>]
  [DEFAULT <expR>]
  [SIZE <expN2>, <expN3>]
  [ENABLE|DISABLE]
  [MESSAGE <expC5>]
  [VALID <expL1> | <expN4>]
  [WHEN <expL2>]
  [COLOR SCHEME <expN5>
  [COLOR <color pair list>]
```

Note : Most of the options mentioned above have the same behaviour for each of the control objects. Hence, they are explained with reference to one control object only (the first one). Some options like FONT, STYLE and COLOR SCHEME/ COLOR are already been taken care of along with the @....SAY command and hence are not discussed here.

The above syntax contains only the options which are common to the control objects. If there are options which apply to a particular control objects they are mentioned along with the particular control object.

I. Check box

Check boxes appear singly or in small groups. They act like toggle switches and are used to indicate a state that is one of the two values such as on or off. Check boxes in a screen act independently of each other, therefore any number of check boxes can be on or off at a given time. They can be used to accept choices.

As shown in figure 1.2, a check box appears as a square with text to the right of it describing the option. When you choose a check box, an 'X' appears in the square.

 Executive

Figure 1.1

Clauses

<row, col>

Row and Column options are numeric expressions with values 0 or greater that determine where the check box appears on the screen.

<memvar|field>

When a check box is checked or unchecked, the choice has to be stored in a memory variable, field or an array element which is specified with the option. Storing the choice is useful, since you can specify what action can be taken if a particular check box is checked/unchecked.

The <memvar> or <field> must be of a numeric or logical type. The box is checked initially if the memory variable stores a value of 1 or .T. and unchecked if it stores a value of 0 or .F. The READ command activates the check box. The state of the check box is stored only when the READ is terminated.

FUNCTION <expC1> | PICTURE >expC2>

The FUNCTION/PICTURE clause determines the type of control object to be created. To create a check box, the FUNCTION clause or the PICTURE clause contains the check box specification code '*C'.

The following examples illustrate the use of the FUNCTION, PICTURE or both the clauses to create two check boxes for choosing between ECONOMY or EXECUTIVE class of travel.

```
STORE 0 TO MCHECK
@ 10,10 GET MCHECK FUNCTION '*C ECONOMY'
@ 10,30 GET MCHECK FUNCTION '*C EXECUTIVE'
READ
```

The PICTURE clause character expression <expC2> uses the same syntax as the FUNCTION clause character expression, except the PICTURE clause expression must begin with @ followed by *C. Thus the code to create the check boxes would be :

```
STORE .F. TO MCHECK
@ 10,10 GET MCHECK PICTURE '@*C ECONOMY'
@ 10,30 GET MCHECK PICTURE '@*C EXECUTIVE'
READ
```

Both the FUNCTION and PICTURE clauses can be used to create a check box. If both are included, the FUNCTION character expression <expC1> must contain *C to create the check box and the PICTURE character expression <expC2> must include the prompt.

For example :

```
STORE 0 TO MCHECK
@ 10,10 GET MCHECK FUNCTION '*C' PICTURE 'ECONOMY'
@ 10,30 GET MCHECK FUNCTION '*C' PICTURE 'EXECUTIVE'
READ
```

Using Hot keys

A hot key allows to immediately choose or change the state of the check box. To assign a hot key, a backslash '\' and a less than sign '<' must be placed before the desired character of the check box prompt. The character which serves as a hot key is underlined.
The following example creates a check box with the prompt 'ECONOMY' and assigns a hot key 'E' to it.

```
STORE 0 TO MCHECK
@ 10,10 GET MCHECK FUNCTION '*C' PICTURE '\<ECONOMY'
READ
```

Disabled Check Boxes

A disabled check box cannot be selected or chosen and is displayed in disabled colors. For example if the 'Economy' class of travel is chosen, then the 'Executive' class check box must be disabled and vice versa. To disable a check box, two backslashes '\\' must be placed before the check box prompt.

The following example disables the check box created earlier :

```
@ 10,10 GET MCHECK FUNCTION '*C \\ECONOMY'
READ
```

Options

DEFAULT <expr>

When a check box is checked/unchecked, the state of the box is saved in a memory variable, an array element or a field. If you specify a memory variable that does not exist, it is automatically created and initialised if the DEFAULT clause is included. However, an array element is not created if it is included in the DEFAULT clause.

The DEFAULT clause is ignored if the memory variable already exists or if a field is specified.

The DEFAULT expression <expr> determines the type of memory variable created and its initial value.

Example

```
@ 10,10 GET MCHECK FUNCTION '*C' PICTURE 'ECONOMY' ;
DEFAULT .T.
READ
```

In the above case, a memory variable of type logical is automatically created by FoxPro and is initialised to .T.

Another way of doing the same thing is :

```
STORE .T. to BUTTON
@ 10,10 GET MCHECK FUNCTION '*C REGULAR' DEFAULT ;
BUTTON
READ
```

SIZE <expN2>, <expN3>

The numeric expression <expN2> specifies the height of a check box and <expN3> specifies the width of the prompt. If <expN2> is 1, the focus rectangle around the check box prompt is clipped. In this case, a larger value for <expN2> needs to be included.

The numeric expression <expN3> specifies the width (in columns) of a check box. By default, the width of a check box is determined by the length of the prompt text. <expN3> can be used to make the check box prompt wider than the default.

ENABLE | DISABLE

By default, a check box is enabled when READ is issued. However, you can prevent a check box from being activated when READ is issued by including DISABLE. A disabled check box cannot be selected and is displayed in disabled colors. The ENABLE clause can then be used to enable a disabled check box.

MESSAGE <expC5>

The MESSAGE clause character expression <expC5> is displayed when a check box is selected. The message is placed in the graphics status bar. If the graphical status bar has been turned off with SET STATUS BAR OFF, the message is centered on the last line of the main FoxPro window.

```
STORE 0 to MCHECK
@ 2, 10 SAY "CLASS"
@ 4,10 GET MCHECK FUNCTION "*C ECONOMY" FONT 'ROMAN',16;
STYLE "BI" MESSAGE "Economy Class is selected"
@ 4,30 GET MCHECK FUNCTION "*C EXECUTIVE" ;
FONT 'ROMAN', 16 STYLE "BI" ;
MESSAGE "Executive Class is selected"
READ
```

VALID <expL1> | <expN4>

VALID is evaluated when the check box is actually selected.

<expL1>

Note : The explanation of the above option is out of the scope of our discussion.

<expN4>

A VALID clause that includes a numeric expression <expN4> is used to specify which object is to be activated after a check box is chosen.

The numeric expression has one of three effects:

When <expN4> is 0, the check box remains the active control.
When <expN4> is positive, <expN4> specifies the number of objects to advance.
When <expN4> is negative, <expN4> specifies the number of objects to move back.

For example, consider a situation where a particular data entry screen has the option for choosing between travelling by Economy Class or Executive Class. A check box has been created for each of the class. Now, obviously the choice can be either class at any point of time.
If you want a facility to be incorporated whereby the Executive class check box cannot be chosen if the Economy check box is chosen and vice versa, then the VALID clause will be used in the following manner :

```
STORE 1 TO MCHECK
@ 2, 10 SAY "CLASS"
@ 4, 10 GET MCHECK FUNCTION '*Ç ECONOMY' VALID 1
@ 4, 13 GET MCHECK FUNCTION '*C EXECUTIVE' ;
FONT 'ROMAN', 16
READ
```

In this case VALID 1 causes FoxPro to skip the next object which is nothing but the Executive Class check box.

WHEN <expL2>

The optional WHEN clause allows or prohibits selection of a check box based on the logical value of <expL2>.

If <expL2> evaluates to a :

logical false (.F.), the check box cannot be selected and is skipped over if placed between other objects.

logical true (.T.), the check box can be selected.