# #87 Radio Button Indirection using Subfiles

Written by Ron Dell'Aquila Published Date April 7, 1988

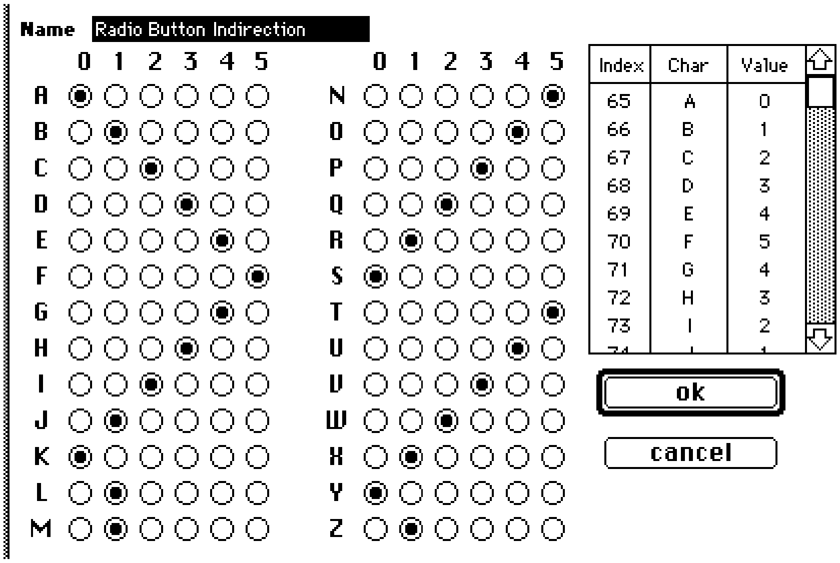
## Using subfiles for "families" of Radio Buttons

Some applications need to access many families of radio buttons. You might use one field per radio button family and use indirection to restore the value of the selected button, such as: ButtonA{[File1]ButtonA}:=1

## General discussion

One field per button is not the most efficient method of storing many families of radio buttons. By using ideas presented in ACIUS Tech Note #19, Checkbox Indirection with Subfiles, we are able to use subrecords to store values of radio button families. Each sub record will contain information for each radio button family. The sub file will contain 2 integer fields of information: 1) the radio button family ASCII identifier and 2) the selected button's value.

In this example, 26 families of buttons were created, named "A", "B", "C"..., and numbered 0 through 5. 26 subrecords will be needed for each parent file record. Stored in each sub record is the numeric ASCII value of each button's family (65 through 90). The selected button value for each family is stored in the other field. If button 2 was selected for the "A" family of buttons, a 2 will be stored with the sub record containing 65 in the ASCII index field, corresponding to character "A". See figure 1.



Name Radio Button Indirection

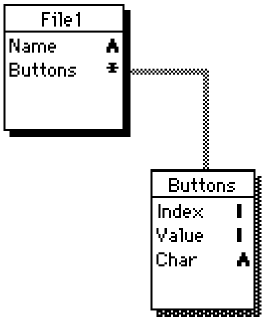
| | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | ● | ○ | ○ | ○ | ○ | ○ | N | ○ | ○ | ○ | ○ | ○ | ● |
| B | ○ | ● | ○ | ○ | ○ | ○ | O | ○ | ○ | ○ | ○ | ● | ○ |
| C | ○ | ○ | ● | ○ | ○ | ○ | P | ○ | ○ | ○ | ● | ○ | ○ |
| D | ○ | ○ | ○ | ● | ○ | ○ | Q | ○ | ○ | ● | ○ | ○ | ○ |
| E | ○ | ○ | ○ | ○ | ● | ○ | R | ○ | ● | ○ | ○ | ○ | ○ |
| F | ○ | ○ | ○ | ○ | ○ | ● | S | ● | ○ | ○ | ○ | ○ | ○ |
| G | ○ | ○ | ○ | ○ | ● | ○ | T | ○ | ○ | ○ | ○ | ○ | ● |
| H | ○ | ○ | ○ | ● | ○ | ○ | U | ○ | ○ | ○ | ○ | ● | ○ |
| I | ○ | ○ | ● | ○ | ○ | ○ | V | ○ | ○ | ○ | ● | ○ | ○ |
| J | ○ | ● | ○ | ○ | ○ | ○ | W | ○ | ○ | ● | ○ | ○ | ○ |
| K | ● | ○ | ○ | ○ | ○ | ○ | X | ○ | ● | ○ | ○ | ○ | ○ |
| L | ○ | ● | ○ | ○ | ○ | ○ | Y | ● | ○ | ○ | ○ | ○ | ○ |
| M | ○ | ● | ○ | ○ | ○ | ○ | Z | ○ | ● | ○ | ○ | ○ | ○ |

| Index | Char | Value |
|---|---|---|
| 65 | A | 0 |
| 66 | B | 1 |
| 67 | C | 2 |
| 68 | D | 3 |
| 69 | E | 4 |
| 70 | F | 5 |
| 71 | G | 4 |
| 72 | H | 3 |
| 73 | I | 2 |

## Initializing the Subfile

It is important to note the need for subrecords containing both the ASCII Index value and its corresponding button value, so when a new record is created, a procedure to create and initialize 26 subrecords must be performed. The global procedure: InitButtons does just that.

The global procedure *InitButtons* adds 26 sub records when creating a new parent record. The sub records are numbered 65 through 90, corresponding to each character of the alphabet.



```
`Global proc: InitButons, used to create and initialize subfile records
$i:=0
While ($i<26)                                    `  26 button families and their subrecords
   $i:=$i+1
   CREATE SUBRECORD([File1]Buttons)
   [File1]Buttons'Index:=$i+64                   `  ASCII Index value, starting with "A"
   [File1]Buttons'Char:=Char($i+64)              `  Actual character, starting with "A"
   [File1]Buttons'Value:=1                       `  Initial button value
End while
```

In order to restore previously assigned radio buttons, a **While** loop placed in the layout's **BEFORE** phase will step through each sub record. Identification of each button family is performed by getting the **Char** of the [File1]Buttons'Index subfield. The value of the selected button is stored in the [File1]Buttons'Value subfield.

```
ALL SUBRECORDS([File1]Buttons)                   `  Insure all subrecords
While (Not(End subselection([File1]Buttons)))    `  Repeat until no subrecords remain
   §(Char([File1]Buttons'Index)+String([File1]Buttons'Value)):=1 `  string indirection
   NEXT SUBRECORD([File1]Buttons)
End while
```

String indirection (§) is used to assemble the combination of the **Char** of the [File1]Buttons'Index) field, with the value of the [File1]Buttons'Value field to activate the proper radio button.

After the record is accepted, the values of the radio buttons are read and saved to disk by using an **APPLY TO SUBSELECTION** to walk down the subfile records, in the **AFTER** phase. Each record has it's own ASCII Index value which is used via string indirection to read it's accompanying button family value.

```
APPLY TO SUBSELECTION([File1]Buttons;[File1]Buttons'Value:=
                    (§(Char([File1]Buttons'Index)+"0")*0)+ (§(Char([File1]Buttons'Index)+"1")+
                    (§(Char([File1]Buttons'Index)+"2")*2)+ (§(Char([File1]Buttons'Index)+"3")*3)+
                    (§(Char([File1]Buttons'Index)+"4")*4)+ (§(Char([File1]Buttons'Index)+"5")*5)))
```

Here is the complete code for a) detecting the need to initialize the subfile with default button values, b) restoring the buttons from the file on disk and c) saving the values of the radio buttons.

```
`   Input layout procedure
Case of
  : (Before)
    [File1]Name:=[File1]Name                          `   set the records' modification flag
    If (Records in subselection([File1]Buttons)=0)   `  if no subs exist, create 'em
      InitButtons                                     `   create 26 sub records, see above code
    End if

    ALL SUBRECORDS([File1]Buttons)  `   ensure all subrecords are present
    While (Not(End subselection([File1]Buttons)))   `   repeat till end of sub file
                                                      `   assign each button
      §(Char([File1]Buttons'Index)+String([File1]Buttons'Value)):=1
      NEXT SUBRECORD([File1]Buttons)
    End while

  : (After)                `   following code is one Apply To Subselection
                                                      `   whose main purpose is to assign every button
                                                      `   to it's accompanying subrecord
            APPLY TO SUBSELECTION([File1]Buttons;[File1]Buttons'Value:=
                (§(Char([File1]Buttons'Index)+"0")*0)+ (§(Char([File1]Buttons'Index)+"1")+
                (§(Char([File1]Buttons'Index)+"2")*2)+ (§(Char([File1]Buttons'Index)+"3")*3)+
                (§(Char([File1]Buttons'Index)+"4")*4)+ (§(Char([File1]Buttons'Index)+"5")*5)))
End case
```