Configuration information specification

Many programs have the facility to configure themselves to set default working parameters. More usually the configuration is done by a separate program which modifies the working program file. Each program will have a different configuration program, and often different versions of the same program will have different configuration programs too. All this makes things very difficult for users.

It is proposed that a standard configuration system is used on all new programs and all new releases of existing programs. If this is done, a single configuration program can be used on any application software file even even when several application files are concatenated.

The advantages of this approach are obvious. There are two disadvantages. The first is that each program has to carry with it all the configuration information which will make it larger. The second is that there is no simple means for doing this with compiled basic programs. The first will not usually be a problem as it seems unlikely that a 32k program would have more than about 20 configurable items and their associated descriptions, this would add at most 3% to the program size. The second can be overcome with a little will.

There are two parts to this system: the first is a standard for the format of a configurable file, the second is a program to process files. There can be any number of programs to process files, from any number of suppliers. If the standards for the configurable file are adhered to, then any supplier's configuration program can be used on any (other) supplier's software.

The configuration consists of the following information:

- Configuration ID
- Configuration level
- Software name
- Software version
- List of
  - Type of item (string, integer etc.) (byte)
  - Item Selection keystroke (byte)
  - Pointer to item
  - Pointer to item pre-processing routine
  - Pointer to item post-processing routine
  - Pointer to description of item
  - Pointer to attributes of item (item type dependent)
- End word (value -1)

As time goes by, additional types of item are likely to be added. This will mean that new versions of the configuration program will be required. These new versions will, of course, be able to configure all lower level configurable files. But, if a old configuration program is used, and the level specified in the configuration block is greater than the level supported by the configuration program, it will have to give up gracefully.

The configuration ID is word aligned and is the eight characters "≪QCFX≫", this is followed by two ASCII characters giving the configuration level (minimum "01"). The software name is a standard string and is followed by a word aligned version identification in a standard string (e.g. "1.13a"). The word aligned list of items follows.

Types of item

Level 01 supports 7 types of item. These are: string, character, code selection, code, byte, word and long word. Application specific types of item can be processed by treating them as strings which are handled entirely by an application supplied routine.

String (type=0)

The form of a configurable string is a word giving the maximum string length, followed by a standard string. There should be enough room within the application program for the maximum length string plus one character for a terminator. There is a single word of attributes with bits set to determine special characteristics.

bit 0 do not strip spaces

Character (type=2)

A character is a single byte, if it is a control character, it will be written out as a two character string (e.g. ^A = $01). There is a single word of attributes with bits set to determine the possible characters allowed.

bit 0 non printable characters
bit 1 digits
bit 2 lower case letters
bit 3 upper case letters
bit 4 other printable characters

bit 6 cursor characters

bit 8 control chars + $40, translated to control chars

Bit 8 is, of course, mutually exclusive with bits 0 to 7, although this is not checked. The configuration block in an application program must be correct.

Code (type=4)

A code is a single byte which may take a small number of values. The attributes is a list of codes giving a byte with the value, a byte with the selection keystroke and a standard string. The list is terminated with an end word (value -1). There are two forms. In the first, the selection keystrokes are set to zero. In this case, when a code is selected, the value will step through all possible values. This is best suited to items which can only have two or three possible codes. Otherwise the user may select any one of the possible codes, either from a list (interactive configuration programs) or from a pull down menu (menu driven configuration programs).

Selection (type=6)

A selection is in the same form as a code, but instead of a byte being set to the selected value, the value is treated as an index to a list of status bytes. When one is selected, it is set to wsi.slct ($80), the previous selection (if different) is set to wsi..avbl (zero). If any status bytes are unavailable (set to wsi.unav=$10), then they will be ignored. The first status byte in the list must not be unavailable.

Values (types 8,10,12)

Largely self explanatory. The attributes are the minimum and maximum values. All values are treated as unsigned.

Item Selection Keystroke

The item selection keystroke is an uppercased keystroke which will select the item in the main menu. The action of selecting the item will depend on the item type. For a code or select item a pull-down window may be opened to enable the user to select the appropriate code. For character item, a single keystroke will be expected. for all other types of item, the item will be made available for editing. For interactive configuration programs, the selection keystroke has no meaning.

Pointer to Item

The pointer to item, and all the other pointers in the definition, are relative addresses stored in a word (e.g. `dc.w item-*`).

Pointer to Item Pre-Processing Routine

It is possible to provide a pre-processing routine within the main program which will be called before an item is presented for changing. This will be when the item is selected in a menu configuration program, or before the prompt is written in an interactive configuration program. If there is no pre-processing routine, the pointer should be zero. The amount of pre-processing that application program can do is not limited. It could just set ranges, or it could do the complete configuration operation itself, including pulling down windows.

Pre-processing Routine

Call parameters		Return parameters	
		D0	item set / error
		D1+	scratch
D7	0 / Window Manager vector	D7	scratch
A0	pointer to item	A0	scratch
A1	pointer to description	A1	(new) ptr to description
A2	pointer to attributes	A2	(new) ptr to attributes
A3	pointer to 4 kbyte space	A3	scratch
		A4+	scratch

Completion codes set as D0

>0	item set, do not prompt or change
=0	ok
<0	error

The space pointed to by A3 is not used by the configuration program and can be used by the application code. Initially it is clear. The application code may use up to 512 bytes of stack.

If D0 (and the status) is returned <0, then the Configuration program will write out an error message and stop.

Pointer to Item Post-Processing Routine

It is possible to provide a post-processing routine within the main program which will be called for each item before configuration starts, and for each item after any item is changed. It can be used to set limits or other dependencies.

Post-processing Routine			
Call parameters		Return parameters	
		D0	info reset / error
D1.b	set this item just changed	D1.b	item status (avbl/unav)
		D2+	scratch
D7	0 / Window Manager Vector	D7	scratch
A0	pointer to item	A0	scratch
A1	pointer to description	A1	(new) ptr to description
A2	pointer to attributes	A2	(new) ptr to attributes
A3	pointer to 4 kbyte space	A3	scratch
		A4+	scratch
Completion codes set as D0			
>0	bit 0 item reset		
	bit 1 description reset		
	bit 2 attributes reset		
=0	ok		
<0	error		

The space pointed to by A3 is not used by the configuration program and can be used by the application code. Initially it is clear. The application code may use up to 512 bytes of stack. If an item description is changed, it should occupy the same number of lines as the original description.

The status values for D1 are WSI.AVBL ($00) if the item can be changed or WSI.UNAV ($10) if the item is not available for changing.

If D0 and the status are <0, A1 and A2 and the item status will not be updated, the error messsage will be written out, no further postprocessing routines will be called, and (for an interactive Configuration program) the item will be re-presented.

A post-processing routine can also be used to set up initial descriptions and attributes.

Description of Item

The description of an item is in the form of a string.

Each description can have several lines, separated by newline characters. Each line should be no longer than 64 characters, except the last line must allow space for the longest item. Interactive programs may append a list of states or selections to the description.

Pointer to attributes

The attributes are item dependent. See item types for descriptions.