Chapter 5

Command Line Handling

Contents

5 Command Line Handling **5–i**

5.1 Input Modes 5–1
5.1.1 Command Line Input 5–1
5.1.2 Menus 5–1
5.2 Command Line Input Syntax 5–1
5.2.1 Verb Syntax 5–2
5.2.2 Qualifier Syntax 5–2
5.2.3 Parameter Syntax 5–2
5.3 Verb Aliases Verb Priority and Group Locking 5–3
5.3.1 Verb Aliases 5–3
5.3.2 Verb Priority 5–3
5.3.3 Group Locking 5–4
5.4 Specification of Verbs Qualifiers and Parameters 5–4
5.4.1 Subprogram Interface 5–4
5.4.2 Definition File 5–4
5.5 Asynchronous Input 5–5
5.6 Verb Parameter and Qualifier Definition Subprograms 5–6
5.6.1 Assign Verb Group 5–7
5.6.2 Deassign Drop a Verb Group 5–8
5.6.3 Define a Verb Group 5–9
5.6.4 Use a Verb Group 5–10
5.6.5 Define Viewport for Group 5–11
5.6.6 Lock Verb Group 5–12
5.6.7 Unlock Verb Group 5–14
5.6.8 Define Verb 5–15
5.6.9 Drop a Verb from Verb Group 5–16
5.6.10 Use Verb 5–17
5.6.11 Define Parameter for Verb 5–18
5.6.12 Drop a Parameter from Verb 5–21
5.6.13 Use Parameter 5–22
5.6.14 Define Qualifier on Verb 5–23
5.6.15 Drop a Qualifier from Verb 5–26

5.6.16 Define Action Routine for Verb 5–27
5.6.17 Drop Action Routine for Verb 5–28
5.6.18 Define Alias Name for Verb 5–29
5.6.19 Drop Delete Alias Name s for Verb 5–30
5.6.20 Define Verb Priority 5–31
5.6.21 Define New Length for Verbs Parameters and Qualifiers . 5–32
5.7 Command Line Decoding Subprograms 5–33
5.7.1 Prompt for Verb 5–33
5.7.2 Pickup next Qualifier 5–37
5.7.3 Get Type of specified Parameter 5–39
5.7.4 Get Type of specified Qualifier 5–41
5.7.5 Test whether Parameters remain on Input Line 5–43

5.1 Input Modes

5.1.1 Command Line Input

Command Line Input takes the form of responses having a regular form, commencing with a Verb which may have several associated Parameters and optional Qualifiers. The VAX/VMS User Interface (DCL) is an example of Command Line Input. UIPACK allows an Application Programmer to create a Command Line User Interface by defining the valid Verbs and their Qualifiers. Verbs may be logically grouped together such that not all Verbs are simultaneously valid and a heirarchical system may be achieved. The Subprograms used within the Prompted Input Interface may also be used to pickup required Parameters associated with each Verb. UIPACK aids the User of such a Program by providing a "Help" facility whereby the set of valid Verbs and a description of each of them may be displayed. Experienced Users may type-ahead in a similar fashion to Prompted Input Mode, thereby bypassing any prompting.

5.1.2 Menus

Menus provide a mechanism whereby less experienced Users of an Application Program may be guided through the set of available choices and allow a complete line of text to be generated via a single keystroke. They furthmore allow sets of program variables, rather than single variables, to have their values adjusted. For example, a set of "switches" build into an Application Program may be named and have their states (On/Off etc.) setup prior to further execution of the Program by presenting a "Toggle" Menu to the User and allowing him to specify the new states. Menus are discussed in detail in the next Chapter.

5.2 Command Line Input Syntax

The Command Line has the following general format:-

```
Verb/Qualifier=Value Parameter/Qualifier Parameter
```

Thus a Command Line consists of a Command Verb, optional Qualifiers that may modify the detailed action of the Verb, and (depending on the Verb) required Parameters. Each Parameter may itself have associated Qualifiers. An example of a Command Line might be:-

```
PRINT/COPIES=2 FILE1/FONT=TIMES,FILE2/FONT=LONDON
```

which would cause the specified files each to be printed twice, the first file in "TIMES" Font, the second in "LONDON" Font. A Command Line is case insensitive, all characters being converted to upper case before being parsed.

Furthermore a Command Line may span several lines on the terminal screen, continuation being indicated by a "-" character as the last significant character of a line (ignoring trailing spaces and comments). Comments are indicated by a "!" character, the remainder of the line being ignored. Multiple Commands Lines may be specified on a single line, separated by spaces or commas.

5.2.1 Verb Syntax

A Verb is the primary response, being a word containing only the alphanumeric characters a-z, A-Z or 0-9, together with the underscore "_" and dollar "$" characters. Verbs may be defined as having up to 32 characters, where only the first 8 are treated as significant by default. This default may be modified if so desired. Verbs are associated with Groups, the default Group being assumed unless explicitly overridden. Verbs within the same Group must be unique, but each Group may contain Verbs that conflict with Verbs from other Groups. Only a single Group of verbs may be "active" at any one time. Groups are identified by a number in the range 0 to 4095, where Group 0 is assumed by default.

5.2.2 Qualifier Syntax

Qualifiers are denoted by a "/" character, followed by the Qualifier name and are optional. Depending on their type they may have a value associated with them (where this value may be Integer, textual etc.). Qualifiers are associated with a particular Verb or Verb Parameter, and must be unique within the set associated with the Verb. Each Verb may have qualifiers that conflict with those for other Verbs. Each qualifier has an attribute that determines whether it is negatable (e.g. /NoList), or has an associated value (e.g. /Copies=2). Qualifiers may be built up from the alphanumeric characters a-z, A-Z and 0-9, together with the underscore "_" and dollar "$" characters. They may contain up to 32 characters of which only the first 8 are significant by default.

5.2.3 Parameter Syntax

Parameters are separated from the Verb (and any qualifiers directly associated with the Verb) by at least one space (or Tab) character, and from each other by spaces or commas. Parameters are not optional, but if they are associated with a Verb, they will be prompted for if they are not already present. This is implemented by the Application Programmer calling the appropriate prompted input subprogram explicitly. Parameters may themselves have associated optional Qualifiers, where some Qualifiers may be position independent (i.e. can be associated with the Command Verb or any associated Parameter) or dependent (i.e. can only be associated with a single Parameter). UIPACK may be used to assign a name for each Parameter associated with a Verb. This name is only used by UIPACK in order to define whether Qualifiers are directly associated with the Verb, or with a particular Parameter of the Verb. The syntax for a Parameter Name is identical to that for Verbs and Qualifiers and thus names may be built up from the alphanumeric characters a-z, A-Z and 0-9, together with the underscore "_" and dollar "$" characters. They may contain up to 32 characters of which only the first 8 are significant by default. In general a particular Verb will be associated with a fixed set of Parameters, but there are instances where a variable number are valid (e.g. the PRINT Command mentioned previously). This situation is catered for by UIPACK, with the restriction that no concatenation of several Commands onto a single line is allowed, since the Verbs from subsequent Commands will be treated as Parameters of the Verb having a variable number of Parameters.

5.3 Verb Aliases Verb Priority and Group Locking

5.3.1 Verb Aliases

Any Verb may have Alias Names defined for it. Thus the Operator may type in either a valid Verb Name or an Alias Name, but the Application Program will receive only valid primary Verb Names. A Verb Alias Name must conform to the same syntax as a Verb, detailed in another Section. There is no limit to the number of Alias Names that may be associated with each Verb, but the Alias Names within a particular Verb Group must be unique.

5.3.2 Verb Priority

The response from a User must normally be uniquely matched with a valid Verb within the currently active Verb Group. However, Verbs may have a Priority assigned to them, such that an ambiguous response may be matched by the Verb having the highest Priority within the Group. The Verb Priority may also be used to disable temporarily a Verb within the Group, whilst retaining the Verb within the Dictionary. Thus all Qualifiers associated with the Verb will not be lost when disabling a particular Verb, whereas they would be lost if the Verb were dropped. Consider the following example. The currently active Group is assumed to have verbs READ, REPLY and RESET. Using the Default Priorities, the User will have to specify either REA, REP or RES in order to ensure non-ambiguity. However, by raising the priority of READ above the default, responses of R or RE will match READ by default. Priorities may be defined to be any valid Integer, where Zero is the Default Priority, and negative priorities will disable the corresponding Verb.

5.3.3 Group Locking

As mentioned earlier, Verbs are associated with Groups, where only a single Group may be active at any one time. In a Multi-Programmer environment, where several people are working on modules to be merged into an overall program, it is possible that one person will attempt to add Verbs etc. to the same Group as another by mistake. This is especially true if they do not assign a free Group ID but assume fixed Groups. In order to prevent this, a Group may be "Locked" such that all subsequent attempts to define more Verbs and Menus, or delete Verbs and Menus within the Group will be treated as illegal. A Password must be specified when Locking a Group, and the same Password must be specified for an attempt to unlock the Group to be satisfied. Note that already existing Menus within a Locked Group may still be modified.

5.4 Specification of Verbs Qualifiers and Parameters

A Command Line style of operation requires that an Application Program have a "Dictionary" of allowed responses, together with a set of subprograms to deal with each response. Thus the Verbs (and their corresponding Qualifiers) must be defined. There are two different mechanisms by which this may be done - either through the use of a Subprogram interface or via a Definition File.

5.4.1 Subprogram Interface

The Subprogram Interface allows each Verb, Qualifier or Menu to be defined via either a single Subprogram call, or a set of calls. These Subprograms allow a flexible, dynamic environment to be setup whereby, for example, individual Buttons on a Menu may be modified during the execution of an Application Program. This Subprogram Interface is discussed in detail in later Sections of the document.

5.4.2 Definition File

The use of a Definition File allows a single Subprogram call to define the complete configuration of Verbs, Qualifiers and Retained Menus. Furthermore, it allows this configuration to be modified without recompiling and rebuilding the Application Code. Thus the descriptive text on a particular Menu Button may be modified within the File and the Application Program will implement this modification the next time it is executed. Note that the Verbs, Qualifiers and Menus defined via a Definition File may be modified during the execution of an Application Program using the Subprogram Interface described earlier. A Definition File may be used to define absolute IDs for Verb Groups or Menus, or it may be used in a modular programming environment to define self-relative Group and Menu IDs. Thus an application program may consist of several modules, each of which wishes to use a Definition File. If only absolute IDs were available, each module designer would have to arrange with the designers of the other modules for a "Block" of Group and Menu IDs to be preassigned. In order to avoid this, in self-relative mode the Groups and Menus are defined relative to a base ID, being the first unassigned ID at the time the definition file is read. The application program may then determine the range of absolute Group and Menu IDs actually assigned to the module. A Definition File is just a normal text file that can be build with a text editor. The default file extension is .UID, and Definition Files are discussed in detail in Chapter 10. Definition Files may themselves reference other Definition Files, where such references may be nested up to a maximum of 4 deep. The syntax for such nesting is the same as that for Indirect Command Files.

5.5 Asynchronous Input

The philosophy used in providing asynchronous input is to allow the Application Program to issue an asynchronous input request (normally coupled to the display of an menu), and to then poll for the detection of input. Once this is detected, the input line may be scanned as normal. By default only the Menu Display Subprograms and the Verb Input Subprogram may be used asynchronously, but UIPACK may be put into a mode whereby all prompted input Subprograms may be used asynchronously. Asynchronous Input is performed by the Application Program requesting asynchronous input mode (via a call to UI_INPUT_MODE, described in a later Section) and then calling the required Menu Display or Prompted Input Subprogram repeatedly until successful input is signalled. Alternatively the Input request may be initiated by calling the appropriate subprogram and then the input status may be polled by using Subprogram UI_TEST_INPUT. The original subprogram should then be recalled in order to act on the input and proceed. For VAX/VMS Systems an Application Programmer may define a Subroutine to be called automatically from the AST Handler associated with Asynchronous Input. This would allow, for example, the appropriate DAQ Package (CDF-166) Subprograms to be called to cancel outstanding "Get Event" or "Get Space" requests.

This Feature only implemented
for VAX/VMS Systems in Text Mode

5.6 Verb Parameter and Qualifier Definition Subprograms

The following subprograms are available for defining Verbs, Parameters and Qualifiers:-

UI_ASSIGN_GROUP UIASGP	) Assign Group ID)
UI_DROP_GROUP UIDRGP	) Deassign (Drop) Group ID)
UI_DEFINE_GROUP UIDFGP	) Define Verb Group)
UI_USE_GROUP UIUSGP	) Use Verb Group)
UI_VIEWPORT_GROUP UIVPGP	) Define Default Viewport) for Menus within Group
UI_LOCK_GROUP UILKGP	) Lock Verb Group)
UI_UNLOCK_GROUP UIULGP	) Unlock Verb Group)
UI_DEFINE_VERB UIDFVB	) Define Verb within Group)
UI_DROP_VERB UIDRVB	) Drop Verb from Group)
UI_USE_VERB UIUSVB	) Use Verb)
UI_DEFINE_PARAMETER UIDFPA	) Define Parameter for Verb)
UI_DROP_PARAMETER UIDRPA	) Drop Parameter from Verb)
UI_USE_PARAMETER UIUSPA	) Use Verb Parameter for) Qualifiers
UI_DEFINE_QUALIFIER UIDFQU	) Define Qualifier for Verb)
UI_DROP_QUALIFIER UIDRQU	) Drop Qualifier from Verb)
UI_DEFINE_ACTION UIDFAC	) Define Action Routine) for Verb
UI_DROP_ACTION UIDRAC	) Drop Action Routine) from Verb

UI_DEFINE_ALIAS) Define Alias Name for Verb
UIDFAL)

UI_DROP_ALIAS) Drop Alias Name for Verb
UIDRAL)

UI_DEFINE_PRIORITY) Define Verb Priority
UIDFPR)

UI_DEFINE_VERB_LENGTH) Define Significant
UIDFVL) Length for Verbs etc.

5.6.1 Assign Verb Group

This Subprogram may be used to assign or allocate a free Verb Group ID. This assignment will be unique within a particular program such that programmers each coding a program module to be later blended together may ensure that they avoid clashes with conflicting verb names etc.

Short Name Calling Sequence

STATUS = UIASGP(GROUP)

Long Name Calling Sequence

STATUS = UI_ASSIGN_GROUP(...)

Input Arguments

None

Output Arguments

GROUP (Integer) Group ID

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__NO_FREE_GROUPS) No Free Groups available
UINOGP)

Note:

1. Group IDs lie in the range 0-4095, where Group 0 (the default) is already assigned.
2. Verbs will be associated by default with Group 0 (zero) unless explicitly defined to reside within another Group.

5.6.2 Deassign Drop a Verb Group

This Subprogram may be used to deassign (drop) a previously assigned Verb Group ID. All Verbs and Menus associated with the Group will automatically also be dropped. This request will be treated as illegal if the specified Group is Locked.

<u>Short Name Calling Sequence</u>

STATUS = UIDRGP(GROUP)

<u>Long Name Calling Sequence</u>

STATUS = UI_DROP_GROUP(...)

<u>Input Arguments</u>

GROUP (Integer) Group ID

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group ID
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

Note:

1. Group 0 is preserved always. Thus, although all Verbs and Menus associated with Group 0 may be dropped by a call to this Subprogram, Group 0 itself will be retained and cannot therefore be assigned or defined by subsequent calls to UI_ASSIGN_GROUP or UI_DEFINE_GROUP.
2. Error UI__LOCKED_GROUP will be returned if the specified Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.3 Define a Verb Group

This Subprogram may be used to define the Group to which all subsequently defined Verbs will belong (unless explicitly overridden by another call to UI_DEFINE_GROUP or UI_USE_GROUP).

Short Name Calling Sequence

STATUS = UIDFGP(GROUP)

Long Name Calling Sequence

STATUS = UI_DEFINE_GROUP(...)

Input Arguments

GROUP (Integer) Group ID

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group ID
UIILGP)

UI__DUPLICATE_GROUP) Duplicate Group ID
UIDUGP)

UI__OVERFLOW) Work Space Overflow
UIOVRF)

Note:

1. Group Numbers must lie in the range 1-4095. Group 0 is always defined and therefore cannot be defined by a call to this subprogram.

5.6.4 Use a Verb Group

This Subroutine may be used to indicate which already existing Verb Group is to be made active. Thus all subsequently defined Verbs will belong to this group, and Command Line responses will be limited to only those Verbs belonging to this Group.

<u>Short Name Calling Sequence</u>

STATUS = UIUSGP(GROUP[,OLDGRP])

<u>Long Name Calling Sequence</u>

STATUS = UI_USE_GROUP(...)

<u>Input Arguments</u>

GROUP (Integer) Group ID

<u>Output Arguments</u>

GROUP (Integer) Old Group ID (Optional)

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group ID
UIILGP)

Note:

1. The specified Group must already have been defined by a prior call to UI_DEFINE_GROUP (or be Group 0, which is always defined).

2. The OLDGRP Argument is optional and, if present, will be set to the Group that was in use prior to this call.

5.6.5 Define Viewport for Group

This Subroutine defines a new Viewport within which Menus within the current Verb Group should be displayed. The default is for Menus to be displayed in the MENU Viewport (UI_MENU_VIEWPORT), but this may be overridden by this call.

<u>Short Name Calling Sequence</u>

STATUS = UIVPGP([VPRTID])

<u>Long Name Calling Sequence</u>

STATUS = UI_VIEWPORT_GROUP(....)

<u>Input Arguments</u>

VPRTID (Integer) Viewport ID (Optional)

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Verb Group
UIILGP)

UI__ILLEGAL_VIEWPORT) Illegal Viewport
UIILVP)

Note:

1. The VPRTID Argument should correspond to a valid Viewport that has already been defined via UI_DEFINE_VIEWPORT.
2. The VPRTID Argument is optional, and, if omitted, will result in the default MENU Viewport being used for subsequent displays of Menus within the current Group.
3. The currently setup Verb Group must have been setup via a prior call to UI_DEFINE_GROUP or UI_USE_GROUP.
4. This Subprogram will only affect the Menu Display for Viewport Modes of operation. It has no effect if UIPACK is in Scroll or Prompted Modes.

5.6.6 Lock Verb Group

This Subprogram locks the currently setup Group such that no more Verbs, Menus etc. may be specified until the Group is unlocked by a call to UI_UNLOCK_GROUP.

Short Name Calling Sequence

STATUS = UILKGP(PASSWD)

Long Name Calling Sequence

STATUS = UI_LOCK_GROUP(...)

Input Arguments

PASSWD (Integer*4) Password

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group
UIILGP)

UI__LOCKED_GROUP) Locked Group
UIGLOK)
Note:

1. The currently setup Group will be locked until a later call to UI_UNLOCK_GROUP

with the same Password.

2. Error UI__LOCKED_GROUP will be signalled if the currently setup Group is already locked.

3. The Password must not be set to zero, or the request will be ignored.

4. Groups are unlocked by default.

5.6.7 Unlock Verb Group

This Subprogram unlocks the currently setup Group such that Verbs, Menus etc. may be specified.

Short Name Calling Sequence

STATUS = UIULGP(PASSWD)

Long Name Calling Sequence

STATUS = UI_UNLOCK_GROUP(...)

Input Arguments

PASSWD (Integer*4) Password

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group
UIILGP)
Note:

1. The Password must correspond to that used to lock the Group in a prior call to UI_LOCK_GROUP or the request will be ignored and error UI__ILLEGAL_GROUP will be returned.
2. Groups are unlocked by default.

5.6.8 Define Verb

This Subprogram defines a new Verb within the currently setup Verb Group. This request will be treated as illegal if the currently setup Group is Locked.

Short Name Calling Sequence

STATUS = UIDFVB(VERB)

Long Name Calling Sequence

STATUS = UI_DEFINE_VERB(VERB)

Input Arguments

VERB (Character) Verb Text

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

UI__ILLEGAL_SYNTAX) Illegal Verb Syntax
UIILSY)

UI__DUPLICATE_VERB) Duplicate Verb
UIDUVB)

UI__OVERFLOW) Work Space Overflow
UIOVRF)

Note:

1. The Verb Text should be made up from only the alpha-numeric characters a-z, A-Z and 0-9 together with the underscore "_" and dollar "$" characters. A maximum of 32 characters may be specified, of which only 8 are significant by default. This default may be modified by a call to UI_DEFINE_VERB_LENGTH which allows the significant length to be adjusted up the maximum of 32 characters.

2. All subsequent calls to define Parameters, Qualifiers and Action Routines etc. will refer to this verb, unless prefixed by another call to UI_DEFINE_VERB or UI_USE_VERB to refer to another valid verb.

3. Error UI__LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.9 Drop a Verb from Verb Group

This Subprogram may be used to drop a previously defined Verb from the currently setup Verb Group. Any Qualifiers associated with the Verb will also be dropped. This request will be treated as illegal if the currently setup Group is Locked.

<u>Short Name Calling Sequence</u>

STATUS = UIDRVB(VERB)

<u>Long Name Calling Sequence</u>

STATUS = UI_DROP_VERB(...)

<u>Input Arguments</u>

VERB (Character) Verb Text

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group ID
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

Note:

1. The specified Verb must refer to an already defined Verb (or truncation thereof) within the currently setup Verb Group.
2. All Verbs within the currently setup Verb Group that satisfy the Name match will be dropped. Thus, if 'A' is specified as the Verb Name, all Verbs beginning with the letter A will be dropped.
3. If the VERB Argument is '*' it is treated as a wildcard and all Verbs associated with the currently setup Verb Group will be dropped.
4. All Parameters and Qualifiers associated with the dropped Verbs will also be dropped.
5. Error UI__LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.10 Use Verb

This Subprogram specifies the Verb to which subsequently defined Parameters and Qualifiers are to be associated.

<u>Short Name Calling Sequence</u>

STATUS = UIUSVB(VERB)

<u>Long Name Calling Sequence</u>

STATUS = UI_USE_VERB(VERB)

<u>Input Arguments</u>

VERB (Character) Verb Text

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

Note:

1. The specified Verb must refer to an already defined Verb within the currently setup Verb Group. The Verb name may be truncated as long as it remains unambiguous.
2. All subsequent calls to define Parameters, Qualifiers and Action Routines will refer to this verb, unless prefixed by another call to UI_DEFINE_VERB or UI_USE_VERB to refer to another valid verb.
3. Subsequently defined Qualifiers will be directly associated with this Verb unless preceeded by a call to UI_DEFINE_PARAMETER to define a Verb Parameter.

5.6.11 Define Parameter for Verb

This Subroutine defines a Parameter for the currently setup Verb (via UI_DEFINE_VERB or UI_USE_VERB) within the currently setup Verb Group. It also specifies whether the Parameter is numeric, text etc. UIPACK does not use this type information other than to make it available to the Application Programmer via UI_GET_PARAMETER_TYPE. Parameters may be used to identifier whether a Qualifier is positional or may be freely associated with the Verb or Parameters. This request will be treated as illegal if the currently setup Group is Locked.

Short Name Calling Sequence

STATUS = UIDFPA(PARAM,TYPE)

Long Name Calling Sequence

STATUS = UI_DEFINE_PARAMETER(...)

Input Arguments

PARAM (Character) Parameter Name
TYPE (Character) Parameter Type (see notes)

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__DUPLICATE_PARAMETER) Duplicate Parameter
UIDUPP)

UI__ILLEGAL_SYNTAX) Illegal Parameter Syntax
UIILSY)

UI__ILLEGAL_VERB) Illegal Verb

UIILVB)

UI__ILLEGAL_GROUP) Illegal Group
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

UI_ILLEGAL_TYPE) Illegal Parameter Type
UIILTP)

Note:

1. The specified Parameter is associated with only the currently setup Verb within the currently setup Group. If this association is desired for several Verbs and Groups, it must be made explicitly by multiple calls to this routine with UI_DEFINE_VERB (or UI_USE_VERB) and UI_DEFINE_GROUP (or UI_USE_GROUP) called as appropriate.

2. The Parameter Name should be made up of only the alphanumeric characters a-z, A-Z and 0-9 and the underscore "_" and dollar "$" characters. Up to 32 characters may be specified, of which 8 are significant by default. This default may be modified by a call to UI_DEFINE_VERB_LENGTH which allows the significant length to be adjusted up the maximum of 32 characters.

3. Following a call to this subprogram any subsequently defined Qualifiers (via UI_DEFINE_QUALIFIER) will be associated with this Verb Parameter rather than directly with the currently setup Verb.

4. Parameters may be of several types such as unspecified, integer, text etc. The TYPE argument specifies the type, and may take on the following values:-

```
' '            Unspecified
'Double'       Double Precision
'Filename'     Filename
'Hexadecimal'  Hexadecimal
'HexBounds'    Hexadecimal Bounds
'HexList'      Hexadecimal List
'Integer'      Integer
'IntBounds'    Integer Bounds
'IntList'      Integer List
'Logical'      Logical (True/False)
'Octal'        Octal
'OctBounds'    Octal Bounds
'OctList'      Octal List
'OnOff'        On/Off
'Real'         Real
'RealBounds'   Real Bounds
'RealList'     Real List
'Text'         Text
'YesNo'        Yes/No
```

where these may be truncated as long as they remain unambiguous (i.e. 'T' is valid) and are case insensitive (i.e. upper or lower case may be used).

Specified Types of 'I', 'In' and 'Int' all default to 'Integer', and similar defaults hold for 'Hex', 'Oct' and 'Real'.

5. Error UI_LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.12 Drop a Parameter from Verb

This Subprogram may be used to drop a previously defined Parameter from the currently setup Verb in the currently setup Verb Group. This will cause any positional Qualifiers associated with the Parameter to also be dropped. This request will be treated as illegal if the currently setup Group is Locked.

<u>Short Name Calling Sequence</u>

STATUS = UIDRPA(PARAM)

<u>Long Name Calling Sequence</u>

STATUS = UI_DROP_PARAMETER(...)

<u>Input Arguments</u>

PARAM (Character) Parameter Name

<u>Output Arguments</u>

None

<u>Function Value</u>

UI_SUCCESS) Success
UISUCC)

UI_NOT_INITIALIZED) Package not Initialized
UININI)

UI_ILLEGAL_GROUP) Illegal Group ID
UIILGP)

UI_LOCKED_GROUP) Group Locked
UIGLOK)

UI_ILLEGAL_VERB) Illegal Verb
UIILVB)

UI_ILLEGAL_PARAMETER) Illegal Parameter
UIILPA)

Note:

1. The specified Parameter must refer to an already defined Parameter (or truncation thereof) for the currently setup Verb.
2. All Parameters satisfying the specified name will be dropped. Thus a call to UI_DROP_PARAMETER with argument 'a' will cause all Parameters beginning with the letter A to be dropped.
3. If the PARAM Argument is '*' it is treated as a wildcard and all Parameters associated with the currently setup Verb will be dropped.
4. All Qualifiers associated with this Parameter will be dropped.
5. Error UI__LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.13 Use Parameter

This Subprogram specifies the Parameter to which subsequently defined Qualifiers are to be associated. Thus such Qualifiers, although associated with the currently setup Verb, may only form part of a valid response when attached directly to this Parameter. This Subprogram is also used when decoding a response from the User in order to identify the currently valid subset of Qualifiers.

<u>Short Name Calling Sequence</u>

STATUS = UIUSPA(PARAM)

<u>Long Name Calling Sequence</u>

STATUS = UI_USE_PARAMETER(PARAM)

<u>Input Arguments</u>

PARAM (Character) Parameter Name

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_PARAMETER) Illegal Parameter
UIILPA)

Note:

1. The specified Parameter must refer to an already defined Parameter associated with the currently setup Verb within the currently setup Verb Group. The Parameter name may be truncated as long as it remains unambiguous.

2. All subsequently defined Qualifiers will be positional dependent, being associated with this Parameter as well as the currently setup Verb.

3. Following a call to this Subprogram, only Qualifiers associated with this Parameter will be treated as legal when decoding a User response. To return to allowing Qualifiers directly associated with the current Verb to be treated as being valid, call UI_USE_VERB again (this being done implicitly by the Action Menu Subprograms).

5.6.14 Define Qualifier on Verb

This Subroutine defines a Qualifier for the currently setup Verb (via UI_DEFINE_VERB or UI_USE_VERB) within the currently setup Verb Group. This Qualifier is positional, being directly associated with the Verb unless it has been preceeded by a call to UI_DEFINE_PARAMETER or UI_USE_PARAMETER in which case it is associated with that Parameter. This Subprogram also specifies whether the Qualifier is negatable, has an associated value, and the type of such a value. This request will be treated as illegal if the currently setup Group is Locked. etc.

Short Name Calling Sequence

STATUS = UIDFQU(QUAL,TYPE)

Long Name Calling Sequence

STATUS = UI_DEFINE_QUALIFIER(...)

Input Arguments

QUAL (Character) Qualifier Name
TYPE (Character) Qualifier Type (see notes)

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__DUPLICATE_QUALIFIER) Duplicate Qualifier
UIDUPQ)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

UI__ILLEGAL_GROUP) Illegal Group
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

UI_ILLEGAL_TYPE) Illegal Qualifier Type
UIILTP)

Note:

1. The specified Qualifier is associated with only the currently setup Verb within the currently setup Group. If this association is desired for several Verbs and Groups, it must be made explicitly by multiple calls to this routine with UI_DEFINE_VERB (or UI_USE_VERB) and UI_DEFINE_GROUP (or UI_USE_GROUP) called as appropriate.

2. This Qualifier is positional, being directly associated with the currently setup Verb or the currently setup Verb Parameter. If this Qualifier should be position-independent, then either no Parameter Names should be associated with this Verb (this is only the names, not the fact that no parameters exist which is determined by the program code) or the Qualifier should be defined explicitly for each Verb and Parameter.

3. The Qualifier text should be made up of only the alphanumeric characters a-z, A-Z and 0-9 and the underscore "_" and dollar "$" characters. Up to 32 characters may be specified, of which 8 are significant by default. This default may be modified by a call to UI_DEFINE_VERB_LENGTH which allows the significant length to be adjusted up the maximum of 32 characters.

4. Qualifiers may be of several types, some of which may have associated values. The TYPE argument specifies the type, and may take on the following values:-

```
' '            No associated Value
'Double'       Double Precision Value
'Filename'     Filename Value
'Hexadecimal'  Hexadecimal Value
'HexBounds'    Hexadecimal Bounds Value
'HexList'      Hexadecimal List Value
'Integer'      Integer Value
'IntBounds'    Integer Bounds Value
'IntList'      Integer List Value
'Logical'      Logical (True/False) Value
'Negatable'    Negatable
'Octal'        Octal Value
'OctBounds'    Octal Bounds Value
'OctList'      Octal List Value
'OnOff'        On/Off Value
'Real'         Real Value
'RealBounds'   Real Bounds Value
'RealList'     Real List Value
'Text'         Text Value
'YesNo'        Yes/No Value
```

where these may be truncated as long as they remain unambiguous (i.e. 'T' is valid) and are case insensitive (i.e. upper or lower case may be used). Specified Types of 'I', 'In' and 'Int' all default to 'Integer', and similar defaults hold for 'Hex', 'Oct' and 'Real'.

5. Error UI__LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.15 Drop a Qualifier from Verb

This Subprogram may be used to drop a previously defined Qualifier from the currently setup Verb in the currently setup Verb Group. This request will be treated as illegal if the currently setup Group is Locked.

<u>Short Name Calling Sequence</u>

STATUS = UIDRQU(QUAL)

<u>Long Name Calling Sequence</u>

STATUS = UI_DROP_QUALIFIER(...)

<u>Input Arguments</u>

QUAL (Character) Qualifier Text

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group ID
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

UI__ILLEGAL_QUALIFIER) Illegal Qualifier
UIILQU)

Note:

1. The specified Qualifier must refer to an already defined Qualifier (or truncation thereof) for the currently setup Verb. Furthermore, it must be associated with the currently setup Verb Parameter if it is associated with one (determined by whether UI_DEFINE_PARAMETER or UI_USEPARAMETER was called prior to calling UI_DEFINE_QUALIFIER).
2. All Qualifiers satisfying the specified name will be dropped. Thus a call to UI_DROP_QUALIFIER with argument 'a' will cause all Qualifiers (associated with the currently setup Parameter if appropriate) beginning with the letter A to be dropped.
3. If the QUAL Argument is '*' it is treated as a wildcard and all Qualifiers associated with the currently setup Verb (and Parameter if appropriate) will be dropped.
4. Error UI__LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.16 Define Action Routine for Verb

This Subroutine defines an Action Routine for the currently setup Verb within the currently setup Verb Group. Whenever this verb is selected (either explicitly by the Operator typing it, or implicitly by the Operator selecting a Menu Button having this verb associated with it), the specified Subroutine will be called automatically.

<u>Short Name Calling Sequence</u>

STATUS = UIDFAC(ACTION)

<u>Long Name Calling Sequence</u>

STATUS = UI_DEFINE_ACTION(....)

<u>Input Arguments</u>

ACTION (see notes)   Action Subroutine Name

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

Note:

1. The ACTION Argument should be the name of the required Action Routine and must be declared EXTERNAL within the calling program.

2. The specified Action Routine will be associated with the currently setup Verb. If association with other Verbs is also desired, this must be performed explicitly by calling this routine multiply, with prior calls to UI_DEFINE_VERB (or UI_USE_VERB) and UI_DEFINE_GROUP (or UI_USE_GROUP) as appropriate.

5.6.17 Drop Action Routine for Verb

This Subroutine drops any Action Routine for the currently setup Verb within the currently setup Verb Group.

<u>Short Name Calling Sequence</u>

STATUS = UIDRAC(DUMMY)

<u>Long Name Calling Sequence</u>

STATUS = UI_DROP_ACTION(....)

<u>Input Arguments</u>

DUMMY (Integer) Dummy Function Argument

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

Note:

1. No error will be reported if the currently setup Verb does not already have an Action Routine.

5.6.18 Define Alias Name for Verb

This Subroutine defines an alternative Alias Name for the currently setup Verb within the currently setup Verb Group. Whenever this Alias Name is entered by the Operator as a response, the primary Verb Name will be returned to the Application Program.

<u>Short Name Calling Sequence</u>

STATUS = UIDFAL(ALIAS)

<u>Long Name Calling Sequence</u>

STATUS = UI_DEFINE_ALIAS(....)

<u>Input Arguments</u>

ALIAS (Character) Verb Alias Name

<u>Output Arguments</u>

None

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Verb Group
UIILGP)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

Note:

1. The specified Alias is only associated with the currently setup Verb in the currently setup Verb Group (via prior calls to UI_DEFINE_VERB or UI_USE_VERB and UI_DEFINE_GROUP or UI_USE_GROUP).
2. An Alias Name must conform to the same naming conventions as for normal Verbs. Thus up to 32 characters may be specified, but only 8 are treated as significant by default. This default may be modified by a call to UI_DEFINE_VERB_LENGTH which allows the significant length to be adjusted up the maximum of 32 characters.
3. There is no limit to the number of Alias Names that may be associated with a Verb, but the Alias Names within each Verb Group must be unique.
4. Error UI__LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.19 Drop Delete Alias Name s for Verb

This Subroutine drops (deletes) Alias Names associated with the currently setup Verb within the currently setup Verb Group. All Aliases beginning with the specified Name will be dropped.

Short Name Calling Sequence

STATUS = UIDRAL(ALIAS)

Long Name Calling Sequence

STATUS = UI_DROP_ALIAS(....)

Input Arguments

ALIAS (Character) Alias Name Text

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_VERB) Illegal Verb
UIILVB)

UI__LOCKED_GROUP) Group Locked
UIGLOK)

Note:

1. The specified Alias Name must refer to an already defined Alias (or truncation thereof) for the currently setup Verb within the currently setup Verb Group.
2. All Aliases that satisfy the Name match will be dropped. Thus, if 'A' is specified as the Alias Name, all Aliases beginning with the letter A will be dropped.
3. If the ALIAS Argument is '*' it is treated as a wildcard and all Aliases associated with the currently setup Verb will be dropped.
4. Error UI_LOCKED_GROUP will be returned if the currently setup Verb Group has been locked by a prior call to UI_LOCK_GROUP.

5.6.20 Define Verb Priority

This Subroutine defines a non-default Priority for the currently setup Verb within the currently setup Verb Group. When an ambiguous response is received from the User, the Verb with the highest priority will be assumed by default (otherwise an ambiguous response error will be flagged). The Priority may also be used to temporarily disable the currently setup Verb without dropping the corresponding Qualifiers etc.

<u>Short Name Calling Sequence</u>

STATUS = UIDFPR(PRIOR)

<u>Long Name Calling Sequence</u>

STATUS = UI_DEFINE_PRIORITY(....)

<u>Input Arguments</u>

PRIOR (Integer) Priority

<u>Output Arguments</u>

None

<u>Function Value</u>

UI_SUCCESS) Success
UISUCC)

UI_NOT_INITIALIZED) Package not Initialized
UININI)

UI_ILLEGAL_VERB) Illegal Verb
UIILVB)

Note:

1. The Default Priority is Zero.
2. A negative Priority will disable the corresponding Verb.

5.6.21 Define New Length for Verbs Parameters and Qualifiers

By default only the first 8 characters of a Verb, Parameter or Qualifier Name are significant. This Subprogram allows this default to be modified up to the maximum of 32 characters.

Short Name Calling Sequence

STATUS = UIDFVL(NEWLEN[,OLDLEN])

Long Name Calling Sequence

STATUS = UI_DEFINE_VERB_LENGTH(...)

Input Arguments

NEWLEN (Integer) New Length

Output Arguments

OLDLEN (Integer) Old Length (Optional)

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

Note:

1. This Subprogram adjusts the significant length for Verbs, Parameters and Qualifiers simultaneously.
2. The NEWLEN Argument specifies the new significant name length. If NEWLEN is zero or negative, the original default of 8 is restored. IF NEWLEN is greater than 32 then 32 is assumed.
3. The OLDLEN Argument is optional and, if present, will be set to the old value of the significant length prior to this call.
4. It is recommended that an Application Program only calls this Subprogram once during its initialisation since UIPACK does not check whether any already existing Verb, Parameter or Qualifier name becomes ambiguous as a result of this call.

b

5.7 Command Line Decoding Subprograms

The following subprograms are available for decoding a Command Line. Note that the decoding of prompted Parameters within the command line is covered in a previous Section.

UI_GET_VERB) Prompt for and Get
UIGTVB) Verb

UI_GET_QUALIFIER) Get next Qualifier from
UIGTQU) Command Line

UI_GET_PARAMETER_TYPE) Get Type of specified
UIGTPT) Parameter

UI_GET_QUALIFIER_TYPE) Get Type of specified
UIGTQT) Qualifier

UI_PARAMETERS_REMAIN) Test whether more
UIPARE) Parameters remain in Line

5.7.1 Prompt for Verb

This Subprogram will prompt for the Operator to supply a text string where the first word in the string will be treated as a Verb and validated against those within the currently setup Group. The full text of the Verb (not just what is typed in as a response) will be returned to the Caller.

Short Name Calling Sequence

STATUS = UIGTVB(PROMPT,VERB)

Long Name Calling Sequence

STATUS = UI_GET_VERB(...)

Input Arguments

PROMPT (Character) Prompt Text

Output Arguments

VERB (Character) Response Verb Text

Function Value

UI__SUCCESS) Success
UISUCC)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

UI__ILLEGAL_GROUP) Illegal Group
UIILGP)

UI__QUAL_PRESENT) Qualifier(s) Present
UIQUPR)

UI__NO_INPUT) No Input Present
UINOIN) (Asynchronous Mode)

UI_INPUT_ABORTED) Input Aborted (Ctrl-Z)
UIABRT)

UI_END_OF_FILE) End-of-File detected
UIEOF) in Batch Mode
Note:

1. Only Verbs (or Aliases thereof) within the currently setup GROUP (via

UI_USE_GROUP etc.) will be legal responses to this call. If an Alias Name is given in response, the Primary Verb Name will be returned to the Application Program.

2. The list of valid response verbs may be determined by typing "?" as a response. This will display a complete list of valid responses. Typing "?NAME", where NAME is a valid verb or truncation thereof will display any detailed help information (supplied via a call to IO_DEFINE_HELP) about all valid verbs commencing with the specified NAME. Thus "?a" will return the Help Information about all Verbs in the currently setup Group that start with the letter "A".

3. If the Verb identified by the response has an Action Routine attached to it (via a prior call to UI_DEFINE_ACTION), then this Subroutine will be called automatically prior to the return from this Subprogram. Great care should be taken in using this feature since it can easily give rise to re-entrant code, this not being supported universally.

4. UI__SUCCESS indicates that the Verb has no directly associated Qualifiers whereas UI__QUAL_PRESENT indicates that the Verb has some attached Qualifiers present. These <u>must</u> be scanned by calls to UI_GET_QUALIFIER until it signifies that no more are present.

5. This Subprogram sets the valid Qualifiers to be those directly associated with the Verb (if any), not those associated with any Verb Parameters. Thus subsequent calls to UI_GET_QUALIFIER will only accept these Verb Qualifiers as valid input.

6. UI__NO_INPUT will only be returned in Asynchronous Mode and indicates that the User has not yet specified any Input. The Application Program should poll using UI_TEST_INPUT until this indicates that some Input has been received, and recall this Subprogram.

7. The UI__INPUT_ABORTED Function Value indicates that the User typed Ctrl-Z to abort input. This is only implemented on a VAX/VMS system. It is an informational function value only, no error message will be reported.

8. The UI__END_OF_FILE Function Value indicates that an End-of-File was detected on the main Input Stream in Batch or non-interactive mode where Input is not taken from a terminal but rather from a file. It is a severe error and therefore causes an error to be reported (unless this has been inhibited by a prior call to UI_ERROR_LEVEL).

9. The Prompt string will only be issued if the response is not already present in the input line.

10. No Prompt will be issued if the specified Prompt String is null (all spaces).

5.7.2 Pickup next Qualifier

This Subprogram will return the text of the next qualifier present on the Command Line. Thus the normal mode of decoding of a Command Line would be to branch on the Verb and then loop calling this subprogram until no more qualifiers are detected. This would then detect all Qualifiers directly associated with the Verb within the Command Line, but not those associated with any Parameters. These may be picked up by applying this Subprogram after each Parameter is itself picked up via the appropriate Prompted Input Subprogram. Positional Qualifiers may be specified by associating a Parameter Name with each Parameter and defining Qualifiers associated with this Parameter Name.

<u>Short Name Calling Sequence</u>

STATUS = UIGTQU(QUAL)

<u>Long Name Calling Sequence</u>

STATUS = UI_GET_QUALIFIER(...)

<u>Input Arguments</u>

None

<u>Output Arguments</u>

QUAL (Character) Text of Qualifier

<u>Function Value</u>

UI__QUAL_PRESENT) Qualifier present having
UIQUPR) no associated Value

UI__QUAL_NEGATED) Qualifier present but
UIQUNE) negated

UI__QUAL_VALUE) Qualifier present with
UIQUVA) associated value

UI__QUAL_DEFAULTED) Qualifier present with
UIQUDF) defaulted value

UI__QUAL_ABSENT) No more Qualifiers
UIQUAB) present

UI__ILLEGAL_QUALIFIER) Illegal Qualifier
UIILQU) present

UI__DUPLICATE_QUALIFIER) Duplicate (ambiquous)
UIDUQU) Qualifier present

UI__ILLEGAL_SYNTAX) Illegal Qualifier Syntax
UIILSY)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

Note:

1. The full text coresponding to the detected qualifier will be returned (in upper case), truncated or padded out if necessary.

2. UI__QUAL_ABSENT is informational only, no error signalling is performed.

3. If no qualifier is detected (corresponding to UI__QUAL_ABSENT function value), then QUAL is set to all spaces.

4. If the qualifier has an associated value (as indicated by the UI__QUAL_VALUE function value), then it may be received by calling the appropriate UI_GET_xxxx subprogram (e.g. UI_GET_TEXT or UI_GET_INTEGER).

5. UI__QUAL_DEFAULTED indicates that a Qualifier defined as having an associated Value (via UI_DEFINE_QUALIFIER) is present but with no associated value specified.

6. Function value UI__ILLEGAL_QUALIFIER will be returned, but no error will be signalled, if an illegal Qualifier is specified on the Input Line. UI__DUPLICATE_QUALIFIER will be returned if an ambiguous Qualifier is specified. QUAL will be set to the text of the illegal or ambiguous Qualifier. It is the Application Programmers responsibility to inform the User that an error was made and ask for further input. The Input Buffer will be flushed in this case unless inhibited by a prior call to UI_FLUSH_MODE.

7. UI__ILLEGAL_SYNTAX signifies that a syntax error was detected. This could imply that a valueless Qualifier has been assigned a value, or that a non-negatable Qualifier has been negated. No error will be signalled, it being the Application Programmer's responsibility to inform the User that an error was made and ask for further input. The Input Buffer will be flushed in this case unless inhibited by a prior call to UI_FLUSH_MODE.

8. The scan for qualifiers proceeds only for the current object on the Command Line. Thus if a verb was the last detected Object, then only Qualifiers attached to the Verb will be detected. Furthermore, only Qualifiers defined as being directly associated with the Verb will be treated as legal. Such Qualifiers were defined (via UI_DEFINE_QUALIFIER) without a prior call to UI_DEFINE_PARAMETER. If a Parameter was the last detected Object in the Input Line, then only Qualifiers attached to this Parameter will be detected. Furthermore, only Qualifiers defined as being associated with the Parameter will be treated as legal. Such Qualifiers were either defined without a prior call to UI_DEFINE_PARAMETER and UI_USE_PARAMETER has not been called prior to this call, or were defined with a prior call to UI_DEFINE_PARAMETER and UI_USE_PARAMETER has been called prior to this call.

5.7.3 Get Type of specified Parameter

This Subprogram will return the Parameter Type of the specified Parameter as a text string. This attribute is not used internally by UIPACK, but could be used by the Application Programmer to determine the UI_GET_xxx Subprogram to be used to prompt for the Parameter.

<u>Short Name Calling Sequence</u>

STATUS = UIGTPT(PARAM,TYPE)

<u>Long Name Calling Sequence</u>

STATUS = UI_GET_PARAMETER_TYPE(...)

<u>Input Arguments</u>

PARAM (Character) Parameter Name

<u>Output Arguments</u>

TYPE (Character) Parameter Type

<u>Function Value</u>

UI__SUCCESS) Success
UISUCC)

UI__ILLEGAL_PARAMETER) Illegal Parameter
UIILPA)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

Note:

1. While this Subprogram is primarily intended for use during the decoding of a Command Line, it also may be used elsewhere, in which case it operates on the currently setup Verb (via UI_USE_VERB).

2. PARAM should refer to a valid Parameter, or an unambiguous truncation thereof, for the currently setup Verb.

3. This Subprogram does not modify the definition of the currently setup Parameter. UI_USE_PARAMETER should be used if this is desired.

4. If the specified Parameter exists for the currently setup Verb, TYPE will be returned as one of the following Text Strings, with the indicated meaning:-

```
' '            Unspecified
'DOUBLE'       Double Precision
'FILENAME'     Filename
'HEXADECIMAL'  Hexadecimal
'HEXBOUNDS'    Hexadecimal Bounds
'HEXLIST'      Hexadecimal List
'INTEGER'      Integer
'INTBOUNDS'    Integer Bounds
'INTLIST'      Integer List
'LOGICAL'      Logical (True/False)
'OCTAL' Octal
'OCTBOUNDS'    Octal Bounds
'OCTLIST'      Octal List
'ONOFF'        On/Off
'REAL'         Real
'REALBOUNDS'   Real Bounds
'REALLIST'     Real List
'TEXT'         Text
'YESNO'        Yes/No
```

   where these will be truncated or padded out with trailing spaces as appropriate.

5. If the specified Parameter is not valid (either because it is not associated with the currently setup Verb, or is ambiguous), TYPE will be set to all spaces.

5.7.4 Get Type of specified Qualifier

This Subprogram will return the Qualifier Type of the specified Qualifier as a text string. It is intended for use in the situation where a Qualifier has been detected by UI_GET_QUALIFIER and the Application Programmer wishes to verify that the returned Function Value detailing whether the Qualifier is negated etc. is consistent with the definition of the Qualifier.

Short Name Calling Sequence

STATUS = UIGTQT(QUAL,TYPE)

Long Name Calling Sequence

STATUS = UI_GET_QUALIFIER_TYPE(...)

Input Arguments

QUAL (Character) Text of Qualifier

Output Arguments

TYPE (Character) Qualifier Type

Function Value

UI__SUCCESS) Success
UISUCC)

UI__ILLEGAL_QUALIFIER) Illegal Qualifier
UIILQU)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

Note:

1. While this Subprogram is primarily intended for use during the decoding of a Command Line, it also may be used elsewhere, in which case it operates on the currently setup Verb (via UI_USE_VERB) and Verb Parameter (via UI_USE_PARAMETER).

2. QUAL should refer to a valid Qualifier, or an unambiguous truncation thereof, for the currently setup Verb. Only Qualifiers associated directly with the currently setup Verb are valid, unless UI_USE_PARAMETER has been called prior to this call, in which case only Qualifiers associated with that Verb Parameter are legal.

3. If the specified Qualifier exists for the currently setup Verb, TYPE will be returned as one of the following Text Strings, with the indicated meaning:-

```
' '             No associated Value
'DOUBLE'        Double Precision Value
'FILENAME'      Filename Value
'HEXADECIMAL'   Hexadecimal Value
'HEXBOUNDS'     Hexadecimal Bounds Value
'HEXLIST'       Hexadecimal List Value
'INTEGER'       Integer Value
'INTBOUNDS'     Integer Bounds Value
'INTLIST'       Integer List Value
'LOGICAL'       Logical (True/False) Value
'NEGATABLE'     Negatable
'OCTAL'         Octal Value
'OCTBOUNDS'     Octal Bounds Value
'OCTLIST'       Octal List Value
'ONOFF'         On/Off Value
'REAL'          Real Value
'REALBOUNDS'    Real Bounds Value
'REALLIST'      Real List Value
'TEXT'          Text Value
'YESNO'         Yes/No Value
```

   where these will be truncated or padded out with trailing spaces as appropriate.

4. If the specified Qualifier is not valid (either because it is not associated with the currently setup Verb, or is ambiguous), TYPE will be set to all spaces.

5.7.5 Test whether Parameters remain on Input Line

This Subprogram tests whether more Parameters remain on the Input Line. It is intended for use in the situation where a Verb may have a variable number of Parameters associated with it. An example of this might be a PRINT command where at least one file must be specified, but more files may be printed if so desired.

Short Name Calling Sequence

STATUS = UIPARE(DUMMY)

Long Name Calling Sequence

STATUS = UI_PARAMETERS_REMAIN(...)

Input Arguments

DUMMY (Integer*4) Dummy Function Argument

Output Arguments

None

Function Value

UI__SUCCESS) Success
UISUCC)

UI__EMPTY) Line empty
UIMPTY)

UI__NOT_INITIALIZED) Package not Initialized
UININI)

Note:

1. The UI__EMPTY Function value indicates that there are no more Parameters on the current Input Line. It is informational only, no signalling is performed.

2. The use of a variable number of Parameters with a particular Verb prohibits the typing of multiple responses on a single Input Line. If this is attempted in this situation, then the additional responses (presumably Verbs and their Parameters) will be treated as additional Parameters to the present Verb.