Purpose

To start up the Glk interface for the Glulx virtual machine, and provide Glulx-specific printing functions.

§**1. Summary.** This segment closely parallels "ZMachine.i6t", which provides exactly equivalent functionality (indeed, usually the same-named functions and in the same order) for the Z-machine VM. This is intended to make the rest of the template code independent of the choice of VM, although that is more of an ideal than a reality, because there are so many fiddly differences in some of the grammar and dictionary tables that it is not really practical for the parser (for instance) to call VM-neutral routines to get the data it wants out of these arrays.

§2. Variables and Arrays.

```
Array gg_event --> 4;
Array gg_arguments buffer 28;
Global gg_mainwin = 0;
Global gg_statuswin = 0;
Global gg_quotewin = 0;
Global gg_scriptfref = 0;
Global gg_scriptstr = 0;
Global gg_savestr = 0;
Global gg_commandstr = 0;
Global gg_command_reading = 0;      ! true if gg_commandstr is being replayed
Global gg_foregroundchan = 0;
Global gg_backgroundchan = 0;

Constant INPUT_BUFFER_LEN = 260;    ! No extra byte necessary
Constant MAX_BUFFER_WORDS = 20;
Constant PARSE_BUFFER_LEN = 61;

Array  buffer    buffer INPUT_BUFFER_LEN;
Array  buffer2   buffer INPUT_BUFFER_LEN;
Array  buffer3   buffer INPUT_BUFFER_LEN;
Array  parse     --> PARSE_BUFFER_LEN;
Array  parse2    --> PARSE_BUFFER_LEN;
```

§**3. Infglk.** This section is a verbatim copy of John Cater's invaluable I6 header file `infglk.h`, kindly donated by the author. The routines are convenient to have on hand, and also provide a canonical set of I6 names for the many gestalt and other codes.

```
#IFNDEF infglk_h; ! Standard Glulx definitions contributed by John Cater
Constant infglk_h;
!-----------------------------------------------------------------------------
!  infglk.h - an Inform library to allow easy access to glk functions
!    under glulx
!  Dynamically created by glk2inf.pl on 08/31/2006 at 19:20:21.
!  Send comments or suggestions to: katre@ruf.rice.edu
!-----------------------------------------------------------------------------
#Ifdef infglk_h;  ! remove "Constant declared but not used" warnings
#Endif;

Constant GLK_NULL 0;

! Constant definitions from glk.h
Constant gestalt_Version 0;
Constant gestalt_CharInput 1;
Constant gestalt_LineInput 2;
Constant gestalt_CharOutput 3;
Constant gestalt_CharOutput_CannotPrint 0;
Constant gestalt_CharOutput_ApproxPrint 1;
Constant gestalt_CharOutput_ExactPrint 2;
Constant gestalt_MouseInput 4;
Constant gestalt_Timer 5;
Constant gestalt_Graphics 6;
Constant gestalt_DrawImage 7;
Constant gestalt_Sound 8;
Constant gestalt_SoundVolume 9;
Constant gestalt_SoundNotify 10;
Constant gestalt_Hyperlinks 11;
Constant gestalt_HyperlinkInput 12;
Constant gestalt_SoundMusic 13;
Constant gestalt_GraphicsTransparency 14;
Constant gestalt_Unicode 15;
Constant evtype_None 0;
Constant evtype_Timer 1;
Constant evtype_CharInput 2;
Constant evtype_LineInput 3;
Constant evtype_MouseInput 4;
Constant evtype_Arrange 5;
Constant evtype_Redraw 6;
Constant evtype_SoundNotify 7;
Constant evtype_Hyperlink 8;
Constant keycode_Unknown $ffffffff;
Constant keycode_Left $fffffffe;
Constant keycode_Right $fffffffd;
Constant keycode_Up $fffffffc;
Constant keycode_Down $fffffffb;
Constant keycode_Return $fffffffa;
Constant keycode_Delete $fffffff9;
Constant keycode_Escape $fffffff8;
Constant keycode_Tab $fffffff7;
```

```
Constant keycode_PageUp $fffffff6;
Constant keycode_PageDown $fffffff5;
Constant keycode_Home $fffffff4;
Constant keycode_End $fffffff3;
Constant keycode_Func1 $ffffffef;
Constant keycode_Func2 $ffffffee;
Constant keycode_Func3 $ffffffed;
Constant keycode_Func4 $ffffffec;
Constant keycode_Func5 $ffffffeb;
Constant keycode_Func6 $ffffffea;
Constant keycode_Func7 $ffffffe9;
Constant keycode_Func8 $ffffffe8;
Constant keycode_Func9 $ffffffe7;
Constant keycode_Func10 $ffffffe6;
Constant keycode_Func11 $ffffffe5;
Constant keycode_Func12 $ffffffe4;
Constant keycode_MAXVAL 28;
Constant style_Normal 0;
Constant style_Emphasized 1;
Constant style_Preformatted 2;
Constant style_Header 3;
Constant style_Subheader 4;
Constant style_Alert 5;
Constant style_Note 6;
Constant style_BlockQuote 7;
Constant style_Input 8;
Constant style_User1 9;
Constant style_User2 10;
Constant style_NUMSTYLES 11;
Constant wintype_AllTypes 0;
Constant wintype_Pair 1;
Constant wintype_Blank 2;
Constant wintype_TextBuffer 3;
Constant wintype_TextGrid 4;
Constant wintype_Graphics 5;
Constant winmethod_Left $00;
Constant winmethod_Right $01;
Constant winmethod_Above $02;
Constant winmethod_Below $03;
Constant winmethod_DirMask $0f;
Constant winmethod_Fixed $10;
Constant winmethod_Proportional $20;
Constant winmethod_DivisionMask $f0;
Constant fileusage_Data $00;
Constant fileusage_SavedGame $01;
Constant fileusage_Transcript $02;
Constant fileusage_InputRecord $03;
Constant fileusage_TypeMask $0f;
Constant fileusage_TextMode $100;
Constant fileusage_BinaryMode $000;
Constant filemode_Write $01;
Constant filemode_Read $02;
Constant filemode_ReadWrite $03;
```

```
Constant filemode_WriteAppend $05;
Constant seekmode_Start 0;
Constant seekmode_Current 1;
Constant seekmode_End 2;
Constant stylehint_Indentation 0;
Constant stylehint_ParaIndentation 1;
Constant stylehint_Justification 2;
Constant stylehint_Size 3;
Constant stylehint_Weight 4;
Constant stylehint_Oblique 5;
Constant stylehint_Proportional 6;
Constant stylehint_TextColor 7;
Constant stylehint_BackColor 8;
Constant stylehint_ReverseColor 9;
Constant stylehint_NUMHINTS 10;
Constant stylehint_just_LeftFlush 0;
Constant stylehint_just_LeftRight 1;
Constant stylehint_just_Centered 2;
Constant stylehint_just_RightFlush 3;
Constant imagealign_InlineUp $01;
Constant imagealign_InlineDown $02;
Constant imagealign_InlineCenter $03;
Constant imagealign_MarginLeft $04;
Constant imagealign_MarginRight $05;
! The actual glk functions.
[ glk_exit _vararg_count ret;
! glk_exit ()
! And now the @glk call
@glk 1 _vararg_count ret;
return ret;
];
[ glk_set_interrupt_handler _vararg_count ret;
! glk_set_interrupt_handler (func)
! And now the @glk call
@glk 2 _vararg_count ret;
return ret;
];
[ glk_tick _vararg_count ret;
! glk_tick ()
! And now the @glk call
@glk 3 _vararg_count ret;
return ret;
];
[ glk_gestalt _vararg_count ret;
! glk_gestalt (sel val)
! And now the @glk call
@glk 4 _vararg_count ret;
return ret;
];
[ glk_gestalt_ext _vararg_count ret;
! glk_gestalt_ext (sel val arr arrlen)
! And now the @glk call
```

```
@glk 5 _vararg_count ret;
return ret;
];
[ glk_char_to_lower _vararg_count ret;
! glk_char_to_lower (ch)
! And now the @glk call
@glk 160 _vararg_count ret;
return ret;
];
[ glk_char_to_upper _vararg_count ret;
! glk_char_to_upper (ch)
! And now the @glk call
@glk 161 _vararg_count ret;
return ret;
];
[ glk_window_get_root _vararg_count ret;
! glk_window_get_root ()
! And now the @glk call
@glk 34 _vararg_count ret;
return ret;
];
[ glk_window_open _vararg_count ret;
! glk_window_open (split method size wintype rock)
! And now the @glk call
@glk 35 _vararg_count ret;
return ret;
];
[ glk_window_close _vararg_count ret;
! glk_window_close (win result)
! And now the @glk call
@glk 36 _vararg_count ret;
return ret;
];
[ glk_window_get_size _vararg_count ret;
! glk_window_get_size (win widthptr heightptr)
! And now the @glk call
@glk 37 _vararg_count ret;
return ret;
];
[ glk_window_set_arrangement _vararg_count ret;
! glk_window_set_arrangement (win method size keywin)
! And now the @glk call
@glk 38 _vararg_count ret;
return ret;
];
[ glk_window_get_arrangement _vararg_count ret;
! glk_window_get_arrangement (win methodptr sizeptr keywinptr)
! And now the @glk call
@glk 39 _vararg_count ret;
return ret;
];
```

```
[ glk_window_iterate _vararg_count ret;
! glk_window_iterate (win rockptr)
! And now the @glk call
@glk 32 _vararg_count ret;
return ret;
];
[ glk_window_get_rock _vararg_count ret;
! glk_window_get_rock (win)
! And now the @glk call
@glk 33 _vararg_count ret;
return ret;
];
[ glk_window_get_type _vararg_count ret;
! glk_window_get_type (win)
! And now the @glk call
@glk 40 _vararg_count ret;
return ret;
];
[ glk_window_get_parent _vararg_count ret;
! glk_window_get_parent (win)
! And now the @glk call
@glk 41 _vararg_count ret;
return ret;
];
[ glk_window_get_sibling _vararg_count ret;
! glk_window_get_sibling (win)
! And now the @glk call
@glk 48 _vararg_count ret;
return ret;
];
[ glk_window_clear _vararg_count ret;
! glk_window_clear (win)
! And now the @glk call
@glk 42 _vararg_count ret;
return ret;
];
[ glk_window_move_cursor _vararg_count ret;
! glk_window_move_cursor (win xpos ypos)
! And now the @glk call
@glk 43 _vararg_count ret;
return ret;
];
[ glk_window_get_stream _vararg_count ret;
! glk_window_get_stream (win)
! And now the @glk call
@glk 44 _vararg_count ret;
return ret;
];
[ glk_window_set_echo_stream _vararg_count ret;
! glk_window_set_echo_stream (win str)
! And now the @glk call
```

```
@glk 45 _vararg_count ret;
return ret;
];
[ glk_window_get_echo_stream _vararg_count ret;
! glk_window_get_echo_stream (win)
! And now the @glk call
@glk 46 _vararg_count ret;
return ret;
];
[ glk_set_window _vararg_count ret;
! glk_set_window (win)
! And now the @glk call
@glk 47 _vararg_count ret;
return ret;
];
[ glk_stream_open_file _vararg_count ret;
! glk_stream_open_file (fileref fmode rock)
! And now the @glk call
@glk 66 _vararg_count ret;
return ret;
];
[ glk_stream_open_memory _vararg_count ret;
! glk_stream_open_memory (buf buflen fmode rock)
! And now the @glk call
@glk 67 _vararg_count ret;
return ret;
];
[ glk_stream_close _vararg_count ret;
! glk_stream_close (str result)
! And now the @glk call
@glk 68 _vararg_count ret;
return ret;
];
[ glk_stream_iterate _vararg_count ret;
! glk_stream_iterate (str rockptr)
! And now the @glk call
@glk 64 _vararg_count ret;
return ret;
];
[ glk_stream_get_rock _vararg_count ret;
! glk_stream_get_rock (str)
! And now the @glk call
@glk 65 _vararg_count ret;
return ret;
];
[ glk_stream_set_position _vararg_count ret;
! glk_stream_set_position (str pos seekmode)
! And now the @glk call
@glk 69 _vararg_count ret;
return ret;
];
```

```
[ glk_stream_get_position _vararg_count ret;
! glk_stream_get_position (str)
! And now the @glk call
@glk 70 _vararg_count ret;
return ret;
];
[ glk_stream_set_current _vararg_count ret;
! glk_stream_set_current (str)
! And now the @glk call
@glk 71 _vararg_count ret;
return ret;
];
[ glk_stream_get_current _vararg_count ret;
! glk_stream_get_current ()
! And now the @glk call
@glk 72 _vararg_count ret;
return ret;
];
[ glk_put_char _vararg_count ret;
! glk_put_char (ch)
! And now the @glk call
@glk 128 _vararg_count ret;
return ret;
];
[ glk_put_char_stream _vararg_count ret;
! glk_put_char_stream (str ch)
! And now the @glk call
@glk 129 _vararg_count ret;
return ret;
];
[ glk_put_string _vararg_count ret;
! glk_put_string (s)
! And now the @glk call
@glk 130 _vararg_count ret;
return ret;
];
[ glk_put_string_stream _vararg_count ret;
! glk_put_string_stream (str s)
! And now the @glk call
@glk 131 _vararg_count ret;
return ret;
];
[ glk_put_buffer _vararg_count ret;
! glk_put_buffer (buf len)
! And now the @glk call
@glk 132 _vararg_count ret;
return ret;
];
[ glk_put_buffer_stream _vararg_count ret;
! glk_put_buffer_stream (str buf len)
! And now the @glk call
```

```
@glk 133 _vararg_count ret;
return ret;
];
[ glk_set_style _vararg_count ret;
! glk_set_style (styl)
! And now the @glk call
@glk 134 _vararg_count ret;
return ret;
];
[ glk_set_style_stream _vararg_count ret;
! glk_set_style_stream (str styl)
! And now the @glk call
@glk 135 _vararg_count ret;
return ret;
];
[ glk_get_char_stream _vararg_count ret;
! glk_get_char_stream (str)
! And now the @glk call
@glk 144 _vararg_count ret;
return ret;
];
[ glk_get_line_stream _vararg_count ret;
! glk_get_line_stream (str buf len)
! And now the @glk call
@glk 145 _vararg_count ret;
return ret;
];
[ glk_get_buffer_stream _vararg_count ret;
! glk_get_buffer_stream (str buf len)
! And now the @glk call
@glk 146 _vararg_count ret;
return ret;
];
[ glk_stylehint_set _vararg_count ret;
! glk_stylehint_set (wintype styl hint val)
! And now the @glk call
@glk 176 _vararg_count ret;
return ret;
];
[ glk_stylehint_clear _vararg_count ret;
! glk_stylehint_clear (wintype styl hint)
! And now the @glk call
@glk 177 _vararg_count ret;
return ret;
];
[ glk_style_distinguish _vararg_count ret;
! glk_style_distinguish (win styl1 styl2)
! And now the @glk call
@glk 178 _vararg_count ret;
return ret;
];
```

```
[ glk_style_measure _vararg_count ret;
! glk_style_measure (win styl hint result)
! And now the @glk call
@glk 179 _vararg_count ret;
return ret;
];
[ glk_fileref_create_temp _vararg_count ret;
! glk_fileref_create_temp (usage rock)
! And now the @glk call
@glk 96 _vararg_count ret;
return ret;
];
[ glk_fileref_create_by_name _vararg_count ret;
! glk_fileref_create_by_name (usage name rock)
! And now the @glk call
@glk 97 _vararg_count ret;
return ret;
];
[ glk_fileref_create_by_prompt _vararg_count ret;
! glk_fileref_create_by_prompt (usage fmode rock)
! And now the @glk call
@glk 98 _vararg_count ret;
return ret;
];
[ glk_fileref_create_from_fileref _vararg_count ret;
! glk_fileref_create_from_fileref (usage fref rock)
! And now the @glk call
@glk 104 _vararg_count ret;
return ret;
];
[ glk_fileref_destroy _vararg_count ret;
! glk_fileref_destroy (fref)
! And now the @glk call
@glk 99 _vararg_count ret;
return ret;
];
[ glk_fileref_iterate _vararg_count ret;
! glk_fileref_iterate (fref rockptr)
! And now the @glk call
@glk 100 _vararg_count ret;
return ret;
];
[ glk_fileref_get_rock _vararg_count ret;
! glk_fileref_get_rock (fref)
! And now the @glk call
@glk 101 _vararg_count ret;
return ret;
];
[ glk_fileref_delete_file _vararg_count ret;
! glk_fileref_delete_file (fref)
! And now the @glk call
```

```
@glk 102 _vararg_count ret;
return ret;
];
[ glk_fileref_does_file_exist _vararg_count ret;
! glk_fileref_does_file_exist (fref)
! And now the @glk call
@glk 103 _vararg_count ret;
return ret;
];
[ glk_select _vararg_count ret;
! glk_select (event)
! And now the @glk call
@glk 192 _vararg_count ret;
return ret;
];
[ glk_select_poll _vararg_count ret;
! glk_select_poll (event)
! And now the @glk call
@glk 193 _vararg_count ret;
return ret;
];
[ glk_request_timer_events _vararg_count ret;
! glk_request_timer_events (millisecs)
! And now the @glk call
@glk 214 _vararg_count ret;
return ret;
];
[ glk_request_line_event _vararg_count ret;
! glk_request_line_event (win buf maxlen initlen)
! And now the @glk call
@glk 208 _vararg_count ret;
return ret;
];
[ glk_request_char_event _vararg_count ret;
! glk_request_char_event (win)
! And now the @glk call
@glk 210 _vararg_count ret;
return ret;
];
[ glk_request_mouse_event _vararg_count ret;
! glk_request_mouse_event (win)
! And now the @glk call
@glk 212 _vararg_count ret;
return ret;
];
[ glk_cancel_line_event _vararg_count ret;
! glk_cancel_line_event (win event)
! And now the @glk call
@glk 209 _vararg_count ret;
return ret;
];
```

```
[ glk_cancel_char_event _vararg_count ret;
! glk_cancel_char_event (win)
! And now the @glk call
@glk 211 _vararg_count ret;
return ret;
];
[ glk_cancel_mouse_event _vararg_count ret;
! glk_cancel_mouse_event (win)
! And now the @glk call
@glk 213 _vararg_count ret;
return ret;
];
[ glk_buffer_to_lower_case_uni _vararg_count ret;
! glk_buffer_to_lower_case_uni (buf len numchars)
! And now the @glk call
@glk 288 _vararg_count ret;
return ret;
];
[ glk_buffer_to_upper_case_uni _vararg_count ret;
! glk_buffer_to_upper_case_uni (buf len numchars)
! And now the @glk call
@glk 289 _vararg_count ret;
return ret;
];
[ glk_buffer_to_title_case_uni _vararg_count ret;
! glk_buffer_to_title_case_uni (buf len numchars lowerrest)
! And now the @glk call
@glk 290 _vararg_count ret;
return ret;
];
[ glk_put_char_uni _vararg_count ret;
! glk_put_char_uni (ch)
! And now the @glk call
@glk 296 _vararg_count ret;
return ret;
];
[ glk_put_string_uni _vararg_count ret;
! glk_put_string_uni (s)
! And now the @glk call
@glk 297 _vararg_count ret;
return ret;
];
[ glk_put_buffer_uni _vararg_count ret;
! glk_put_buffer_uni (buf len)
! And now the @glk call
@glk 298 _vararg_count ret;
return ret;
];
[ glk_put_char_stream_uni _vararg_count ret;
! glk_put_char_stream_uni (str ch)
! And now the @glk call
```

```
@glk 299 _vararg_count ret;
return ret;
];
[ glk_put_string_stream_uni _vararg_count ret;
! glk_put_string_stream_uni (str s)
! And now the @glk call
@glk 300 _vararg_count ret;
return ret;
];
[ glk_put_buffer_stream_uni _vararg_count ret;
! glk_put_buffer_stream_uni (str buf len)
! And now the @glk call
@glk 301 _vararg_count ret;
return ret;
];
[ glk_get_char_stream_uni _vararg_count ret;
! glk_get_char_stream_uni (str)
! And now the @glk call
@glk 304 _vararg_count ret;
return ret;
];
[ glk_get_buffer_stream_uni _vararg_count ret;
! glk_get_buffer_stream_uni (str buf len)
! And now the @glk call
@glk 305 _vararg_count ret;
return ret;
];
[ glk_get_line_stream_uni _vararg_count ret;
! glk_get_line_stream_uni (str buf len)
! And now the @glk call
@glk 306 _vararg_count ret;
return ret;
];
[ glk_stream_open_file_uni _vararg_count ret;
! glk_stream_open_file_uni (fileref fmode rock)
! And now the @glk call
@glk 312 _vararg_count ret;
return ret;
];
[ glk_stream_open_memory_uni _vararg_count ret;
! glk_stream_open_memory_uni (buf buflen fmode rock)
! And now the @glk call
@glk 313 _vararg_count ret;
return ret;
];
[ glk_request_char_event_uni _vararg_count ret;
! glk_request_char_event_uni (win)
! And now the @glk call
@glk 320 _vararg_count ret;
return ret;
];
```

```
[ glk_request_line_event_uni _vararg_count ret;
! glk_request_line_event_uni (win buf maxlen initlen)
! And now the @glk call
@glk 321 _vararg_count ret;
return ret;
];
[ glk_image_draw _vararg_count ret;
! glk_image_draw (win image val1 val2)
! And now the @glk call
@glk 225 _vararg_count ret;
return ret;
];
[ glk_image_draw_scaled _vararg_count ret;
! glk_image_draw_scaled (win image val1 val2 width height)
! And now the @glk call
@glk 226 _vararg_count ret;
return ret;
];
[ glk_image_get_info _vararg_count ret;
! glk_image_get_info (image width height)
! And now the @glk call
@glk 224 _vararg_count ret;
return ret;
];
[ glk_window_flow_break _vararg_count ret;
! glk_window_flow_break (win)
! And now the @glk call
@glk 232 _vararg_count ret;
return ret;
];
[ glk_window_erase_rect _vararg_count ret;
! glk_window_erase_rect (win left top width height)
! And now the @glk call
@glk 233 _vararg_count ret;
return ret;
];
[ glk_window_fill_rect _vararg_count ret;
! glk_window_fill_rect (win color left top width height)
! And now the @glk call
@glk 234 _vararg_count ret;
return ret;
];
[ glk_window_set_background_color _vararg_count ret;
! glk_window_set_background_color (win color)
! And now the @glk call
@glk 235 _vararg_count ret;
return ret;
];
[ glk_schannel_create _vararg_count ret;
! glk_schannel_create (rock)
! And now the @glk call
```

```
@glk 242 _vararg_count ret;
return ret;
];
[ glk_schannel_destroy _vararg_count ret;
! glk_schannel_destroy (chan)
! And now the @glk call
@glk 243 _vararg_count ret;
return ret;
];
[ glk_schannel_iterate _vararg_count ret;
! glk_schannel_iterate (chan rockptr)
! And now the @glk call
@glk 240 _vararg_count ret;
return ret;
];
[ glk_schannel_get_rock _vararg_count ret;
! glk_schannel_get_rock (chan)
! And now the @glk call
@glk 241 _vararg_count ret;
return ret;
];
[ glk_schannel_play _vararg_count ret;
! glk_schannel_play (chan snd)
! And now the @glk call
@glk 248 _vararg_count ret;
return ret;
];
[ glk_schannel_play_ext _vararg_count ret;
! glk_schannel_play_ext (chan snd repeats notify)
! And now the @glk call
@glk 249 _vararg_count ret;
return ret;
];
[ glk_schannel_stop _vararg_count ret;
! glk_schannel_stop (chan)
! And now the @glk call
@glk 250 _vararg_count ret;
return ret;
];
[ glk_schannel_set_volume _vararg_count ret;
! glk_schannel_set_volume (chan vol)
! And now the @glk call
@glk 251 _vararg_count ret;
return ret;
];
[ glk_sound_load_hint _vararg_count ret;
! glk_sound_load_hint (snd flag)
! And now the @glk call
@glk 252 _vararg_count ret;
return ret;
];
```

```
[ glk_set_hyperlink _vararg_count ret;
! glk_set_hyperlink (linkval)
! And now the @glk call
@glk 256 _vararg_count ret;
return ret;
];
[ glk_set_hyperlink_stream _vararg_count ret;
! glk_set_hyperlink_stream (str linkval)
! And now the @glk call
@glk 257 _vararg_count ret;
return ret;
];
[ glk_request_hyperlink_event _vararg_count ret;
! glk_request_hyperlink_event (win)
! And now the @glk call
@glk 258 _vararg_count ret;
return ret;
];
[ glk_cancel_hyperlink_event _vararg_count ret;
! glk_cancel_hyperlink_event (win)
! And now the @glk call
@glk 259 _vararg_count ret;
return ret;
];
#ENDIF;
```

§**4. Rocks.** These are unique ID codes used to mark resources; think of them as inedible cookies.

```
Constant GG_MAINWIN_ROCK        201;
Constant GG_STATUSWIN_ROCK      202;
Constant GG_QUOTEWIN_ROCK       203;
Constant GG_SAVESTR_ROCK        301;
Constant GG_SCRIPTSTR_ROCK      302;
Constant GG_COMMANDWSTR_ROCK    303;
Constant GG_COMMANDRSTR_ROCK    304;
Constant GG_SCRIPTFREF_ROCK     401;
Constant GG_FOREGROUNDCHAN_ROCK 410;
Constant GG_BACKGROUNDCHAN_ROCK 411;
```

§**5. Stubs.** These are I6 library-style entry point routines, not used by I7, but retained in case I7 extensions want to do interesting things with Glulx.

```
#Stub HandleGlkEvent    2;
#Stub IdentifyGlkObject 4;
#Stub InitGlkWindow     1;
```

§**6. Starting Up.** `VM_Initialise()` is almost the first routine called, except that the "starting the virtual machine" activity is allowed to go first; and, come to think of it, memory allocation has to be set up before even that, and that in turn calls `VM_PreInitialise()` to do the absolute minimum.

Arrangements are a little different here from on the Z-machine, because some data is retained in the case of a restart.

(Many thanks are due to Eliuk Blau, who found several tricky timing errors here and elsewhere in the Glulx-specific code. Frankly, I feel like hanging a sign on the following routines which reads "Congratulations on bringing light to the Dark Room.")

```
[ VM_PreInitialise res;
    @gestalt 4 2 res; ! Test if this interpreter has Glk...
    if (res == 0) quit; ! ...without which there would be nothing we could do
    unicode_gestalt_ok = false;
    if (glk_gestalt(gestalt_Unicode, 0))
        unicode_gestalt_ok = true;
    ! Set the VM's I/O system to be Glk.
    @setiosys 2 0;
];
[ VM_Initialise res sty i;
    @gestalt 4 2 res; ! Test if this interpreter has Glk...
    if (res == 0) quit; ! ...without which there would be nothing we could do
    ! First, we must go through all the Glk objects that exist, and see
    ! if we created any of them. One might think this strange, since the
    ! program has just started running, but remember that the player might
    ! have just typed "restart".
    GGRecoverObjects();
    ! Sound channel initialisation, and RNG fixing, must be done now rather
    ! than later in case InitGlkWindow() returns a non-zero value.
    if (glk_gestalt(gestalt_Sound, 0)) {
        if (gg_foregroundchan == 0)
            gg_foregroundchan = glk_schannel_create(GG_FOREGROUNDCHAN_ROCK);
        if (gg_backgroundchan == 0)
            gg_backgroundchan = glk_schannel_create(GG_BACKGROUNDCHAN_ROCK);
    }
    #ifdef FIX_RNG;
    @random 10000 i;
    i = -i-2000;
    print "[Random number generator seed is ", i, "]^";
    @setrandom i;
    #endif; ! FIX_RNG
    res = InitGlkWindow(0);
    if (res ~= 0) return;
    ! Now, gg_mainwin and gg_storywin might already be set. If not, set them.
    if (gg_mainwin == 0) {
        ! Open the story window.
        res = InitGlkWindow(GG_MAINWIN_ROCK);
        if (res == 0) {
            ! Left-justify the header style
            glk_stylehint_set(wintype_TextBuffer, style_Header, stylehint_Justification, 0);
            ! Try to make emphasized type in italics and not boldface
```

```
            glk_stylehint_set(wintype_TextBuffer, style_Emphasized, stylehint_Weight, 0);
            glk_stylehint_set(wintype_TextBuffer, style_Emphasized, stylehint_Oblique, 1);
            gg_mainwin = glk_window_open(0, 0, 0, wintype_TextBuffer, GG_MAINWIN_ROCK);
        }
        if (gg_mainwin == 0) quit; ! If we can't even open one window, give in
    } else {
        ! There was already a story window. We should erase it.
        glk_window_clear(gg_mainwin);
    }
    if (gg_statuswin == 0) {
        res = InitGlkWindow(GG_STATUSWIN_ROCK);
        if (res == 0) {
            statuswin_cursize = statuswin_size;
            for (sty=0: sty<style_NUMSTYLES: sty++)
                glk_stylehint_set(wintype_TextGrid, sty, stylehint_ReverseColor, 1);
            gg_statuswin =
                glk_window_open(gg_mainwin, winmethod_Fixed + winmethod_Above,
                    statuswin_cursize, wintype_TextGrid, GG_STATUSWIN_ROCK);
        }
    }
    ! It's possible that the status window couldn't be opened, in which case
    ! gg_statuswin is now zero. We must allow for that later on.
    glk_set_window(gg_mainwin);
    InitGlkWindow(1);
];
[ GGRecoverObjects id;
    ! If GGRecoverObjects() has been called, all these stored IDs are
    ! invalid, so we start by clearing them all out.
    ! (In fact, after a restoreundo, some of them may still be good.
    ! For simplicity, though, we assume the general case.)
    gg_mainwin = 0;
    gg_statuswin = 0;
    gg_quotewin = 0;
    gg_scriptfref = 0;
    gg_scriptstr = 0;
    gg_savestr = 0;
    statuswin_cursize = 0;
    gg_foregroundchan = 0;
    gg_backgroundchan = 0;
    #Ifdef DEBUG;
    gg_commandstr = 0;
    gg_command_reading = false;
    #Endif; ! DEBUG
    ! Also tell the game to clear its object references.
    IdentifyGlkObject(0);
    id = glk_stream_iterate(0, gg_arguments);
    while (id) {
        switch (gg_arguments-->0) {
            GG_SAVESTR_ROCK: gg_savestr = id;
            GG_SCRIPTSTR_ROCK: gg_scriptstr = id;
            #Ifdef DEBUG;
            GG_COMMANDWSTR_ROCK: gg_commandstr = id;
```

```
                            gg_command_reading = false;
            GG_COMMANDRSTR_ROCK: gg_commandstr = id;
                            gg_command_reading = true;
            #Endif; ! DEBUG
            default: IdentifyGlkObject(1, 1, id, gg_arguments-->0);
        }
        id = glk_stream_iterate(id, gg_arguments);
    }
    id = glk_window_iterate(0, gg_arguments);
    while (id) {
        switch (gg_arguments-->0) {
            GG_MAINWIN_ROCK: gg_mainwin = id;
            GG_STATUSWIN_ROCK: gg_statuswin = id;
            GG_QUOTEWIN_ROCK: gg_quotewin = id;
            default: IdentifyGlkObject(1, 0, id, gg_arguments-->0);
        }
        id = glk_window_iterate(id, gg_arguments);
    }
    id = glk_fileref_iterate(0, gg_arguments);
    while (id) {
        switch (gg_arguments-->0) {
            GG_SCRIPTFREF_ROCK: gg_scriptfref = id;
            default: IdentifyGlkObject(1, 2, id, gg_arguments-->0);
        }
        id = glk_fileref_iterate(id, gg_arguments);
    }
    if (glk_gestalt(gestalt_Sound, 0)) {
        id = glk_schannel_iterate(0, gg_arguments);
        while (id) {
            switch (gg_arguments-->0) {
                GG_FOREGROUNDCHAN_ROCK: gg_foregroundchan = id;
                GG_BACKGROUNDCHAN_ROCK: gg_backgroundchan = id;
            }
            id = glk_schannel_iterate(id, gg_arguments);
        }
        if (gg_foregroundchan ~= 0) { glk_schannel_stop(gg_foregroundchan); }
        if (gg_backgroundchan ~= 0) { glk_schannel_stop(gg_backgroundchan); }
    }
    ! Tell the game to tie up any loose ends.
    IdentifyGlkObject(2);
];
```

§**7. Enable Acceleration.** This enables use of March 2009 extension to Glulx which optimises the speed of Inform-compiled story files by moving the work of I6 veneer routines into the interpreter itself. It should have no effect on earlier versions of the Glulx VM, which will lack the gestalt for this feature, but nor should it do any harm.

```
[ ENABLE_GLULX_ACCEL_R addr res;
    @gestalt 9 0 res;
    if (res == 0) return;
    addr = #classes_table;
    @accelparam 0 addr;
    @accelparam 1 INDIV_PROP_START;
    @accelparam 2 Class;
    @accelparam 3 Object;
    @accelparam 4 Routine;
    @accelparam 5 String;
    addr = #globals_array + WORDSIZE * #g$self;
    @accelparam 6 addr;
    @accelparam 7 NUM_ATTR_BYTES;
    addr = #cpv__start;
    @accelparam 8 addr;
    @accelfunc 1 Z__Region;
    @accelfunc 2 CP__Tab;
    @accelfunc 3 RA__Pr;
    @accelfunc 4 RL__Pr;
    @accelfunc 5 OC__Cl;
    @accelfunc 6 RV__Pr;
    @accelfunc 7 OP__Pr;
    rfalse;
];
```

§**8. Release Number.** Like all software, IF story files have release numbers to mark revised versions being circulated: unlike most software, and partly for traditional reasons, the version number is recorded not in some print statement or variable but is branded on, so to speak, in a specific memory location of the story file header.

`VM_Describe_Release()` describes the release and is used as part of the "banner", IF's equivalent to a title page.

```
[ VM_Describe_Release i;
    print "Release ";
    @aloads ROM_GAMERELEASE 0 i;
    print i;
    print " / Serial number ";
    for (i=0 : i<6 : i++) print (char) ROM_GAMESERIAL->i;
];
```

§**9. Keyboard Input.** The VM must provide three routines for keyboard input:

(a) `VM_KeyChar()` waits for a key to be pressed and then returns the character chosen as a ZSCII character.
(b) `VM_KeyDelay(N)` waits up to $N/10$ seconds for a key to be pressed, returning the ZSCII character if so, or 0 if not.
(c) `VM_ReadKeyboard(b, t)` reads a whole newline-terminated command into the buffer `b`, then parses it into a word stream in the table `t`.

There are elaborations to due with mouse clicks, but this isn't the place to document all of that.

```
[ VM_KeyChar win nostat done res ix jx ch;
    jx = ch; ! squash compiler warnings
    if (win == 0) win = gg_mainwin;
    if (gg_commandstr ~= 0 && gg_command_reading ~= false) {
        done = glk_get_line_stream(gg_commandstr, gg_arguments, 31);
        if (done == 0) {
            glk_stream_close(gg_commandstr, 0);
            gg_commandstr = 0;
            gg_command_reading = false;
            ! fall through to normal user input.
        } else {
            ! Trim the trailing newline
            if (gg_arguments->(done-1) == 10) done = done-1;
            res = gg_arguments->0;
            if (res == '\') {
                res = 0;
                for (ix=1 : ix<done : ix++) {
                    ch = gg_arguments->ix;
                    if (ch >= '0' && ch <= '9') {
                        @shiftl res 4 res;
                        res = res + (ch-'0');
                    } else if (ch >= 'a' && ch <= 'f') {
                        @shiftl res 4 res;
                        res = res + (ch+10-'a');
                    } else if (ch >= 'A' && ch <= 'F') {
                        @shiftl res 4 res;
                        res = res + (ch+10-'A');
                    }
                }
            }
            jump KCPContinue;
        }
    }
    done = false;
    glk_request_char_event(win);
    while (~~done) {
        glk_select(gg_event);
        switch (gg_event-->0) {
        5: ! evtype_Arrange
            if (nostat) {
                glk_cancel_char_event(win);
                res = $80000000;
                done = true;
                break;
            }
```

```
            DrawStatusLine();
        2: ! evtype_CharInput
            if (gg_event-->1 == win) {
                res = gg_event-->2;
                done = true;
                }
        }
        ix = HandleGlkEvent(gg_event, 1, gg_arguments);
        if (ix == 2) {
            res = gg_arguments-->0;
            done = true;
        } else if (ix == -1)  done = false;
    }
    if (gg_commandstr ~= 0 && gg_command_reading == false) {
        if (res < 32 || res >= 256 || (res == '\' or ' ')) {
            glk_put_char_stream(gg_commandstr, '\');
            done = 0;
            jx = res;
            for (ix=0 : ix<8 : ix++) {
                @ushiftr jx 28 ch;
                @shiftl jx 4 jx;
                ch = ch & $0F;
                if (ch ~= 0 || ix == 7) done = 1;
                if (done) {
                    if (ch >= 0 && ch <= 9) ch = ch + '0';
                    else                    ch = (ch - 10) + 'A';
                    glk_put_char_stream(gg_commandstr, ch);
                }
            }
        } else {
            glk_put_char_stream(gg_commandstr, res);
        }
        glk_put_char_stream(gg_commandstr, 10); ! newline
    }
.KCPContinue;
    return res;
];
[ VM_KeyDelay tenths  key done ix;
    glk_request_char_event(gg_mainwin);
    glk_request_timer_events(tenths*100);
    while (~~done) {
        glk_select(gg_event);
        ix = HandleGlkEvent(gg_event, 1, gg_arguments);
        if (ix == 2) {
            key = gg_arguments-->0;
            done = true;
        }
        else if (ix >= 0 && gg_event-->0 == 1 or 2) {
            key = gg_event-->2;
            done = true;
        }
    }
    glk_cancel_char_event(gg_mainwin);
```

```
    glk_request_timer_events(0);
    return key;
];
[ VM_ReadKeyboard  a_buffer a_table done ix;
    if (gg_commandstr ~= 0 && gg_command_reading ~= false) {
        done = glk_get_line_stream(gg_commandstr, a_buffer+WORDSIZE,
            (INPUT_BUFFER_LEN-WORDSIZE)-1);
        if (done == 0) {
            glk_stream_close(gg_commandstr, 0);
            gg_commandstr = 0;
            gg_command_reading = false;
            ! L__M(##CommandsRead, 5); would come after prompt
            ! fall through to normal user input.
        }
        else {
            ! Trim the trailing newline
            if ((a_buffer+WORDSIZE)->(done-1) == 10) done = done-1;
            a_buffer-->0 = done;
            VM_Style(INPUT_VMSTY);
            glk_put_buffer(a_buffer+WORDSIZE, done);
            VM_Style(NORMAL_VMSTY);
            print "^";
            jump KPContinue;
        }
    }
    done = false;
    glk_request_line_event(gg_mainwin, a_buffer+WORDSIZE, INPUT_BUFFER_LEN-WORDSIZE, 0);
    while (~~done) {
        glk_select(gg_event);
        switch (gg_event-->0) {
        5: ! evtype_Arrange
            DrawStatusLine();
        3: ! evtype_LineInput
            if (gg_event-->1 == gg_mainwin) {
                a_buffer-->0 = gg_event-->2;
                done = true;
            }
        }
        ix = HandleGlkEvent(gg_event, 0, a_buffer);
        if (ix == 2) done = true;
        else if (ix == -1) done = false;
    }
    if (gg_commandstr ~= 0 && gg_command_reading == false) {
        glk_put_buffer_stream(gg_commandstr, a_buffer+WORDSIZE, a_buffer-->0);
        glk_put_char_stream(gg_commandstr, 10); ! newline
    }
.KPContinue;
    VM_Tokenise(a_buffer,a_table);
    ! It's time to close any quote window we've got going.
    if (gg_quotewin) {
        glk_window_close(gg_quotewin, 0);
        gg_quotewin = 0;
    }
```

```
    #ifdef ECHO_COMMANDS;
    print "** ";
    for (ix=WORDSIZE: ix<(a_buffer-->0)+WORDSIZE: ix++) print (char) a_buffer->ix;
    print "^";
    #endif; ! ECHO_COMMANDS
];
```

§**10. Buffer Functions.** A "buffer", in this sense, is an array containing a stream of characters typed from the keyboard; a "parse buffer" is an array which resolves this into individual words, pointing to the relevant entries in the dictionary structure. Because each VM has its own format for each of these arrays (not to mention the dictionary), we have to provide some standard operations needed by the rest of the template as routines for each VM.

`VM_CopyBuffer(to, from)` copies one buffer into another.

`VM_Tokenise(buff, parse_buff)` takes the text in the buffer `buff` and produces the corresponding data in the parse buffer `parse_buff` – this is called tokenisation since the characters are divided into words: in traditional computing jargon, such clumps of characters treated syntactically as units are called tokens.

`LTI_Insert` is documented in the DM4 and the `LTI` prefix stands for "Language To Informese": it's used only by translations into non-English languages of play, and is not called in the template.

```
[ VM_CopyBuffer bto bfrom i;
    for (i=0: i<INPUT_BUFFER_LEN: i++) bto->i = bfrom->i;
];
[ VM_PrintToBuffer buf len a b c;
    if (b) {
        if (metaclass(a) == Object && a.#b == WORDSIZE
            && metaclass(a.b) == String)
            buf-->0 = Glulx_PrintAnyToArray(buf+WORDSIZE, len, a.b);
        else if (metaclass(a) == Routine)
            buf-->0 = Glulx_PrintAnyToArray(buf+WORDSIZE, len, a, b, c);
        else
            buf-->0 = Glulx_PrintAnyToArray(buf+WORDSIZE, len, a, b);
    }
    else if (metaclass(a) == Routine)
        buf-->0 = Glulx_PrintAnyToArray(buf+WORDSIZE, len, a, b, c);
    else
        buf-->0 = Glulx_PrintAnyToArray(buf+WORDSIZE, len, a);
    if (buf-->0 > len) buf-->0 = len;
    return buf-->0;
];
[ VM_Tokenise buf tab
    cx numwords len bx ix wx wpos wlen val res dictlen entrylen;
    len = buf-->0;
    buf = buf+WORDSIZE;
    ! First, split the buffer up into words. We use the standard Infocom
    ! list of word separators (comma, period, double-quote).
    cx = 0;
    numwords = 0;
    while (cx < len) {
        while (cx < len && buf->cx == ' ') cx++;
        if (cx >= len) break;
        bx = cx;
```

```
        if (buf->cx == '.' or ',' or '"') cx++;
        else {
            while (cx < len && buf->cx ~= ' ' or '.' or ',' or '"') cx++;
        }
        tab-->(numwords*3+2) = (cx-bx);
        tab-->(numwords*3+3) = WORDSIZE+bx;
        numwords++;
        if (numwords >= MAX_BUFFER_WORDS) break;
    }
    tab-->0 = numwords;

    ! Now we look each word up in the dictionary.

    dictlen = #dictionary_table-->0;
    entrylen = DICT_WORD_SIZE + 7;

    for (wx=0 : wx<numwords : wx++) {
        wlen = tab-->(wx*3+2);
        wpos = tab-->(wx*3+3);

        ! Copy the word into the gg_tokenbuf array, clipping to DICT_WORD_SIZE
        ! characters and lower case.
        if (wlen > DICT_WORD_SIZE) wlen = DICT_WORD_SIZE;
        cx = wpos - WORDSIZE;
        for (ix=0 : ix<wlen : ix++) gg_tokenbuf->ix = VM_UpperToLowerCase(buf->(cx+ix));
        for (: ix<DICT_WORD_SIZE : ix++) gg_tokenbuf->ix = 0;

        val = #dictionary_table + WORDSIZE;
        @binarysearch gg_tokenbuf DICT_WORD_SIZE val entrylen dictlen 1 1 res;
        tab-->(wx*3+1) = res;
    }
];

[ LTI_Insert i ch  b y;

    ! Protect us from strict mode, as this isn't an array in quite the
    ! sense it expects
    b = buffer;

    ! Insert character ch into buffer at point i.
    ! Being careful not to let the buffer possibly overflow:
    y = b-->0;
    if (y > INPUT_BUFFER_LEN) y = INPUT_BUFFER_LEN;

    ! Move the subsequent text along one character:
    for (y=y+WORDSIZE : y>i : y--) b->y = b->(y-1);
    b->i = ch;

    ! And the text is now one character longer:
    if (b-->0 < INPUT_BUFFER_LEN) (b-->0)++;
];
```

§**11. Dictionary Functions.** Again, the dictionary structure is differently arranged on the different VMs. This is a data structure containing, in compressed form, the text of all the words to be recognised by tokenisation (above). In I6 for Glulx, a dictionary word is represented at run-time by its record's address in the dictionary.

`VM_InvalidDictionaryAddress(A)` tests whether `A` is a valid record address in the dictionary data structure. In Glulx, dictionary records might in theory be anywhere in the 2 GB or so of possible memory, but we can rule out negative addresses. (This allows −1, say, to be used as a value meaning "not a valid dictionary word".)

`VM_DictionaryAddressToNumber(A)` and `VM_NumberToDictionaryAddress(N)` convert between word addresses and their run-time representations: since, on Glulx, they are the same, these are each the identity function.

```
[ VM_InvalidDictionaryAddress addr;
    if (addr < 0) rtrue;
    rfalse;
];
[ VM_DictionaryAddressToNumber w; return w; ];
[ VM_NumberToDictionaryAddress n; return n; ];
Array gg_tokenbuf -> DICT_WORD_SIZE;
[ GGWordCompare str1 str2 ix jx;
    for (ix=0 : ix<DICT_WORD_SIZE : ix++) {
        jx = (str1->ix) - (str2->ix);
        if (jx ~= 0) return jx;
    }
    return 0;
];
```

§**12. SHOWVERB support.** Further VM-specific tables cover actions and attributes, and these are used by the SHOWVERB testing command.

```
#Ifdef DEBUG;
[ DebugAction a str;
    if (a >= 4096) { print "<fake action ", a-4096, ">"; return; }
    if (a < 0 || a >= #identifiers_table-->7) print "<invalid action ", a, ">";
    else {
        str = #identifiers_table-->6;
        str = str-->a;
        if (str) print (string) str; else print "<unnamed action ", a, ">";
    }
];
[ DebugAttribute a str;
    if (a < 0 || a >= NUM_ATTR_BYTES*8) print "<invalid attribute ", a, ">";
    else {
        str = #identifiers_table-->4;
        str = str-->a;
        if (str) print (string) str; else print "<unnamed attribute ", a, ">";
    }
];
#Endif;
```

§**13. Command Tables.** The VM is also generated containing a data structure for the grammar produced by I6's `Verb` and `Extend` directives: this is essentially a list of command verbs such as DROP or PUSH, together with a list of synonyms, and then the grammar for the subsequent commands to be recognised by the parser.

```
[ VM_CommandTableAddress i;
    return (#grammar_table)-->(i+1);
];
[ VM_PrintCommandWords i wd j dictlen entrylen;
    dictlen = #dictionary_table-->0;
    entrylen = DICT_WORD_SIZE + 7;
    for (j=0 : j<dictlen : j++) {
        wd = #dictionary_table + WORDSIZE + entrylen*j;
        if (DictionaryWordToVerbNum(wd) == i)
            print "'", (address) wd, "' ";
    }
];
```

§**14. Random Number Generator.** No routine is needed for extracting a random number, since I6's built-in `random` function does that, but it's useful to abstract the process of seeding the RNG so that it produces a repeatable sequence of "random" numbers from here on: the necessary opcodes are different for the two VMs.

```
[ VM_Seed_RNG n;
    @setrandom n;
];
```

§**15. Memory Allocation.** This is dynamic memory allocation: something which is never practicable in the Z-machine, because the whole address range is already claimed, but which is viable on recent revisions of Glulx.

```
[ VM_AllocateMemory amount i;
    @gestalt 7 0 i;
    if (i == 0) return i;
    @malloc amount i;
    return i;
];
[ VM_FreeMemory address i;
    @gestalt 7 0 i;
    if (i == 0) return;
    @mfree address;
];
```

§**16. Audiovisual Resources.** The Z-machine only barely supports figures and sound effects, so Glulx is the preferred VM to choose if they are wanted. Properly speaking, it's not Glulx which supports these, but its I/O layer Glk, and implementations of Glk are free to support them or not as they please: "cheapglk", a dumb terminal version, does not, for instance. We therefore have to investigate the "gestalt" to find out.

```
[ VM_Picture resource_ID;
    if (glk_gestalt(gestalt_Graphics, 0)) {
        glk_image_draw(gg_mainwin, resource_ID, imagealign_InlineCenter, 0);
    } else {
        print "[Picture number ", resource_ID, " here.]^";
    }
];
[ VM_SoundEffect resource_ID;
    if (glk_gestalt(gestalt_Sound, 0)) {
        glk_schannel_play(gg_foregroundchan, resource_ID);
    } else {
        print "[Sound effect number ", resource_ID, " here.]^";
    }
];
```

§**17. Typography.** Glk makes an attempt to present typographic styles as being a matter of semantic markup rather than controlling the actual appearance of text: the idea is that the story file should want to print something in a heading kind of way, and then the interpreter – guided by the player's reading preferences – might set that in bold, or larger type, or red ink, or any combination of the three, or with other effects entirely. This is not the place to discuss whether that was a wise decision for Glk to take (it really, really, *really* wasn't): we can only play along.

```
[ VM_Style sty;
    switch (sty) {
        NORMAL_VMSTY:     glk_set_style(style_Normal);
        HEADER_VMSTY:     glk_set_style(style_Header);
        SUBHEADER_VMSTY:  glk_set_style(style_Subheader);
        NOTE_VMSTY:       glk_set_style(style_Note);
        ALERT_VMSTY:      glk_set_style(style_Alert);
        BLOCKQUOTE_VMSTY: glk_set_style(style_BlockQuote);
        INPUT_VMSTY:      glk_set_style(style_Input);
    }
];
```

§**18. Character Casing.** The following are the equivalent of `tolower` and `toupper`, the traditional C library functions for forcing letters into lower and upper case form, for the ZSCII character set. Note that Glulx can also use Unicode characters for some purposes (Unicode was a relatively late addition to the Glulx standard), and we make good use of this when storing indexed text.

```
[ VM_UpperToLowerCase c; return glk_char_to_lower(c); ];
[ VM_LowerToUpperCase c; return glk_char_to_upper(c); ];
```

§**19. Glulx-Only Printing Routines.** Partly because of the smallness of the range of representable values in the Z-machine, there is little run-time type-checking that can be done: for instance a dictionary address cannot be distinguished from a function address because they are encoded differently, so that a function address (which is packed) could well coincide with that of a dictionary word (which is not). On Glulx these restrictions are somewhat lifted, so that it's possible to write a routine which can look at a value, work out what it must mean, and print it suitably. This is only possible up to a point – for instance, it can't distinguish an integer from a function address – and in I7 the use of this sort of trick is much less important because type-checking in the NI compiler handles the problem much better. Still, we retain some Glulx-only features because they are convenient for writing external files to disc, for instance, something which the Z-machine can't do in any case.

`Glulx_PrintAnything` handles strings, functions (with optional arguments), objects, object properties (with optional arguments), and dictionary words.

`Glulx_PrintAnyToArray` does the same, but the output is sent to a byte array in memory. The first two arguments must be the array address and length; subsequent arguments are as for `Glulx_PrintAnything`. The return value is the number of characters output. If the output is longer than the array length given, the extra characters are discarded, so the array does not overflow. (However, the return value is the total length of the output, including discarded characters.) The character set stored here is ZSCII, not Unicode.

`Glulx_ChangeAnyToCString` calls `Glulx_PrintAnyToArray` on a particular array, then amends the result to make it a C-style string – that is, a sequence of byte-sized characters which are null terminated. The character set stored here is once again ZSCII, not Unicode.

```
! Glulx_PrintAnything()                    <nothing printed>
! Glulx_PrintAnything(0)                   <nothing printed>
! Glulx_PrintAnything("string");           print (string) "string";
! Glulx_PrintAnything('word')              print (address) 'word';
! Glulx_PrintAnything(obj)                 print (name) obj;
! Glulx_PrintAnything(obj, prop)           obj.prop();
! Glulx_PrintAnything(obj, prop, args...)  obj.prop(args...);
! Glulx_PrintAnything(func)                func();
! Glulx_PrintAnything(func, args...)       func(args...);
[ Glulx_PrintAnything _vararg_count obj mclass;
    if (_vararg_count == 0) return;
    @copy sp obj;
    _vararg_count--;
    if (obj == 0) return;
    if (obj->0 == $60) {
        ! Dictionary word. Metaclass() can't catch this case, so we do it manually
        print (address) obj;
        return;
    }
    mclass = metaclass(obj);
    switch (mclass) {
    nothing:
        return;
    String:
        print (string) obj;
        return;
    Routine:
        ! Call the function with all the arguments which are already
        ! on the stack.
        @call obj _vararg_count 0;
        return;
```

```
    Object:
        if (_vararg_count == 0) {
            print (name) obj;
        }
        else {
            ! Push the object back onto the stack, and call the
            ! veneer routine that handles obj.prop() calls.
            @copy obj sp;
            _vararg_count++;
            @call CA__Pr _vararg_count 0;
        }
        return;
    }
];
[ Glulx_PrintAnyToArray _vararg_count arr arrlen str oldstr len;
    @copy sp arr;
    @copy sp arrlen;
    _vararg_count = _vararg_count - 2;
    oldstr = glk_stream_get_current();
    str = glk_stream_open_memory(arr, arrlen, 1, 0);
    if (str == 0) return 0;
    glk_stream_set_current(str);
    @call Glulx_PrintAnything _vararg_count 0;
    glk_stream_set_current(oldstr);
    @copy $ffffffff sp;
    @copy str sp;
    @glk $0044 2 0; ! stream_close
    @copy sp len;
    @copy sp 0;
    return len;
];
Constant GG_ANYTOSTRING_LEN 66;
Array AnyToStrArr -> GG_ANYTOSTRING_LEN+1;
[ Glulx_ChangeAnyToCString _vararg_count ix len;
    ix = GG_ANYTOSTRING_LEN-2;
    @copy ix sp;
    ix = AnyToStrArr+1;
    @copy ix sp;
    ix = _vararg_count+2;
    @call Glulx_PrintAnyToArray ix len;
    AnyToStrArr->0 = $E0;
    if (len >= GG_ANYTOSTRING_LEN)
        len = GG_ANYTOSTRING_LEN-1;
    AnyToStrArr->(len+1) = 0;
    return AnyToStrArr;
];
```

§20. The Screen. Our generic screen model is that the screen is made up of windows: we tend to refer only to two of these, the main window and the status line, but others may also exist from time to time. Windows have unique ID numbers: the special window ID −1 means "all windows" or "the entire screen", which usually amounts to the same thing.

Screen height and width are measured in characters, with respect to the fixed-pitch font used for the status line. The main window normally contains variable-pitch text which may even have been kerned, and character dimensions make little sense there.

```
[ VM_ClearScreen window;
    if (window == WIN_ALL or WIN_MAIN) {
        glk_window_clear(gg_mainwin);
        if (gg_quotewin) {
            glk_window_close(gg_quotewin, 0);
            gg_quotewin = 0;
        }
    }
    if (gg_statuswin && window == WIN_ALL or WIN_STATUS) glk_window_clear(gg_statuswin);
];
[ VM_ScreenWidth  id;
    id=gg_mainwin;
    if (gg_statuswin && statuswin_current) id = gg_statuswin;
    glk_window_get_size(id, gg_arguments, 0);
    return gg_arguments-->0;
];
[ VM_ScreenHeight;
    glk_window_get_size(gg_mainwin, 0, gg_arguments);
    return gg_arguments-->0;
];
```

§21. Window Colours. Our generic screen model is that the screen is made up of windows, each of which can have its own foreground and background colours.

The colour of individual letters or words of type is not controllable in Glulx, to the frustration of many, and so the template layer of I7 has no framework for handling this (even though it is controllable on the Z-machine, which is greatly superior in this respect).

```
[ VM_SetWindowColours f b window doclear  i fwd bwd swin;
    if (clr_on && f && b) {
        if (window) swin = 5-window; ! 4 for TextGrid, 3 for TextBuffer
        fwd = MakeColourWord(f);
        bwd = MakeColourWord(b);
        for (i=0 : i<style_NUMSTYLES: i++) {
            if (f == CLR_DEFAULT || b == CLR_DEFAULT) {  ! remove style hints
                glk_stylehint_clear(swin, i, stylehint_TextColor);
                glk_stylehint_clear(swin, i, stylehint_BackColor);
            } else {
                glk_stylehint_set(swin, i, stylehint_TextColor, fwd);
                glk_stylehint_set(swin, i, stylehint_BackColor, bwd);
            }
        }
        ! Now re-open the windows to apply the hints
        if (gg_statuswin) glk_window_close(gg_statuswin, 0);
```

```
        gg_statuswin = 0;
        if (doclear || ( window ~= 1 && (clr_fg ~= f || clr_bg ~= b) ) ) {
            glk_window_close(gg_mainwin, 0);
            gg_mainwin = glk_window_open(0, 0, 0, wintype_TextBuffer, GG_MAINWIN_ROCK);
            if (gg_scriptstr ~= 0)
                glk_window_set_echo_stream(gg_mainwin, gg_scriptstr);
        }
        gg_statuswin =
            glk_window_open(gg_mainwin, winmethod_Fixed + winmethod_Above,
                statuswin_cursize, wintype_TextGrid, GG_STATUSWIN_ROCK);
        if (statuswin_current && gg_statuswin) VM_MoveCursorInStatusLine(); else VM_MainWindow();
        if (window ~= 2) {
            clr_fgstatus = f;
            clr_bgstatus = b;
        }
        if (window ~= 1) {
            clr_fg = f;
            clr_bg = b;
        }
    }
];
[ VM_RestoreWindowColours; ! used after UNDO: compare I6 patch L61007
    if (clr_on) { ! check colour has been used
        VM_SetWindowColours(clr_fg, clr_bg, 2); ! make sure both sets of variables are restored
        VM_SetWindowColours(clr_fgstatus, clr_bgstatus, 1, true);
        VM_ClearScreen();
    }
];
[ MakeColourWord c;
    if (c > 9) return c;
    c = c-2;
    return $ff0000*(c&1) + $ff00*(c&2 ~= 0) + $ff*(c&4 ~= 0);
];
```

§**22. Main Window.** The part of the screen on which commands and responses are printed, which ordinarily occupies almost all of the screen area.

`VM_MainWindow()` switches printing back from another window, usually the status line, to the main window.

```
[ VM_MainWindow;
    glk_set_window(gg_mainwin); ! set_window
    statuswin_current=0;
];
```

§**23. Status Line.** Despite the name, the status line need not be a single line at the top of the screen: that's only the conventional default arrangement. It can expand to become the equivalent of an old-fashioned VT220 terminal, with menus and grids and mazes displayed lovingly in character graphics, or it can close up to invisibility.

`VM_StatusLineHeight(n)` sets the status line to have a height of `n` lines of type. (The width of the status line is always the width of the whole screen, and the position is always at the top, so the height is the only controllable aspect.) The $n = 0$ case makes the status line disappear.

`VM_MoveCursorInStatusLine(x, y)` switches printing to the status line, positioning the "cursor" – the position at which printing will begin – at the given character grid position (x, y). Line 1 represents the top line; line 2 is underneath, and so on; columns are similarly numbered from 1 at the left.

```
[ VM_StatusLineHeight hgt;
    if (gg_statuswin == 0) return;
    if (hgt == statuswin_cursize) return;
    glk_window_set_arrangement(glk_window_get_parent(gg_statuswin), $12, hgt, 0);
    statuswin_cursize = hgt;
];
[ VM_MoveCursorInStatusLine line column;
    if (gg_statuswin == 0) return;
    glk_set_window(gg_statuswin);
    if (line == 0) { line = 1; column = 1; }
    glk_window_move_cursor(gg_statuswin, column-1, line-1);
    statuswin_current=1;
];
```

§**24. Quotation Boxes.** On the Z-machine, quotation boxes are produced by stretching the status line, but on Glulx they usually occupy windows of their own. If it isn't possible to create such a window, so that `gg_quotewin` is zero below, the quotation text just appears in the main window.

```
[ Box__Routine maxwid arr ix lines lastnl parwin;
    maxwid = 0; ! squash compiler warning
    lines = arr-->0;
    if (gg_quotewin == 0) {
        gg_arguments-->0 = lines;
        ix = InitGlkWindow(GG_QUOTEWIN_ROCK);
        if (ix == 0)
            gg_quotewin =
                glk_window_open(gg_mainwin, winmethod_Fixed + winmethod_Above,
                    lines, wintype_TextBuffer, GG_QUOTEWIN_ROCK);
    } else {
        parwin = glk_window_get_parent(gg_quotewin);
        glk_window_set_arrangement(parwin, $12, lines, 0);
    }
    lastnl = true;
    if (gg_quotewin) {
        glk_window_clear(gg_quotewin);
        glk_set_window(gg_quotewin);
        lastnl = false;
    }
    VM_Style(BLOCKQUOTE_VMSTY);
    for (ix=0 : ix<lines : ix++) {
```

```
        print (string) arr-->(ix+1);
        if (ix < lines-1 || lastnl) new_line;
    }
    VM_Style(NORMAL_VMSTY);
    if (gg_quotewin) glk_set_window(gg_mainwin);
];
```

§**25. GlkList Command.** GLKLIST is a testing command best used by those who understand Glulx and its ways: it isn't documented in the I7 manual, because it is pretty inscrutable for "real" users, but it's probably worth keeping just the same.

```
#Ifdef DEBUG;
[ GlkListSub id val;
    id = glk_window_iterate(0, gg_arguments);
    while (id) {
        print "Window ", id, " (", gg_arguments-->0, "): ";
        val = glk_window_get_type(id);
        switch (val) {
        1: print "pair";
        2: print "blank";
        3: print "textbuffer";
        4: print "textgrid";
        5: print "graphics";
        default: print "unknown";
        }
        val = glk_window_get_parent(id);
        if (val) print ", parent is window ", val;
        else     print ", no parent (root)";
        val = glk_window_get_stream(id);
        print ", stream ", val;
        val = glk_window_get_echo_stream(id);
        if (val) print ", echo stream ", val;
        print "^";
        id = glk_window_iterate(id, gg_arguments);
    }
    id = glk_stream_iterate(0, gg_arguments);
    while (id) {
        print "Stream ", id, " (", gg_arguments-->0, ")^";
        id = glk_stream_iterate(id, gg_arguments);
    }
    id = glk_fileref_iterate(0, gg_arguments);
    while (id) {
        print "Fileref ", id, " (", gg_arguments-->0, ")^";
        id = glk_fileref_iterate(id, gg_arguments);
    }
    if (glk_gestalt(gestalt_Sound, 0)) {
        id = glk_schannel_iterate(0, gg_arguments);
        while (id) {
            print "Soundchannel ", id, " (", gg_arguments-->0, ")^";
            id = glk_schannel_iterate(id, gg_arguments);
        }
    }
```

```
];
{-testing-command:glklist}
    *                                           -> Glklist;
#Endif;
```

§**26. Undo.** These are really emulations of the Z-machine's conventions on UNDO: Glulx's undo opcodes used different result codes while providing essentially the same functionality, for reasons which are opaque, but no trouble is caused thereby.

```
[ VM_Undo result_code;
    @restoreundo result_code;
    return (~~result_code);
];
[ VM_Save_Undo result_code;
    @saveundo result_code;
    if (result_code == -1) { GGRecoverObjects(); return 2; }
    return (~~result_code);
];
```

§27. Quit The Game Rule.

```
[ QUIT_THE_GAME_R;
    if (actor ~= player) rfalse;
    GL__M(##Quit, 2); if (YesOrNo()~=0) quit;
];
```

§28. Restart The Game Rule.

```
[ RESTART_THE_GAME_R;
    if (actor ~= player) rfalse;
    GL__M(##Restart, 1);
    if (YesOrNo() ~= 0) {
        @restart;
        GL__M(##Restart, 2);
    }
];
```

§29. Restore The Game Rule.

```
[ RESTORE_THE_GAME_R res fref;
    if (actor ~= player) rfalse;
    fref = glk_fileref_create_by_prompt($01, $02, 0);
    if (fref == 0) jump RFailed;
    gg_savestr = glk_stream_open_file(fref, $02, GG_SAVESTR_ROCK);
    glk_fileref_destroy(fref);
    if (gg_savestr == 0) jump RFailed;
    @restore gg_savestr res;
    glk_stream_close(gg_savestr, 0);
    gg_savestr = 0;
    .RFailed;
    GL__M(##Restore, 1);
];
```

§30. Save The Game Rule.

```
[ SAVE_THE_GAME_R res fref;
    if (actor ~= player) rfalse;
    fref = glk_fileref_create_by_prompt($01, $01, 0);
    if (fref == 0) jump SFailed;
    gg_savestr = glk_stream_open_file(fref, $01, GG_SAVESTR_ROCK);
    glk_fileref_destroy(fref);
    if (gg_savestr == 0) jump SFailed;
    @save gg_savestr res;
    if (res == -1) {
        ! The player actually just typed "restore". We're going to print
        !  GL__M(##Restore,2); the Z-Code Inform library does this correctly
        ! now. But first, we have to recover all the Glk objects; the values
        ! in our global variables are all wrong.
        GGRecoverObjects();
        glk_stream_close(gg_savestr, 0); ! stream_close
        gg_savestr = 0;
        return GL__M(##Restore, 2);
    }
    glk_stream_close(gg_savestr, 0); ! stream_close
    gg_savestr = 0;
    if (res == 0) return GL__M(##Save, 2);
    .SFailed;
    GL__M(##Save, 1);
];
```

§31. Verify The Story File Rule. This is a fossil now, really, but in the days of Infocom, the 110K story file occupying an entire disc was a huge data set: floppy discs were by no means a reliable medium, and cheap hardware often used hit-and-miss components, as on the notorious Commodore 64 disc controller. If somebody experienced an apparent bug in play, it could easily be that he had a corrupt disc or was unable to read data of that density. So the VERIFY command, which took up to ten minutes on some early computers, would chug through the entire story file and compute a checksum, compare it against a known result in the header, and determine that the story file could or could not properly be read. The Z-machine provided this service as an opcode, and so Glulx followed suit.

```
[ VERIFY_THE_STORY_FILE_R res;
    if (actor ~= player) rfalse;
    @verify res;
    if (res == 0) return GL__M(##Verify, 1);
    GL__M(##Verify, 2);
];
```

§32. Switch Transcript On Rule.

```
[ SWITCH_TRANSCRIPT_ON_R;
    if (actor ~= player) rfalse;
    if (gg_scriptstr ~= 0) return GL__M(##ScriptOn, 1);
    if (gg_scriptfref == 0) {
        gg_scriptfref = glk_fileref_create_by_prompt($102, $05, GG_SCRIPTFREF_ROCK);
        if (gg_scriptfref == 0) jump S1Failed;
    }
    ! stream_open_file
    gg_scriptstr = glk_stream_open_file(gg_scriptfref, $05, GG_SCRIPTSTR_ROCK);
    if (gg_scriptstr == 0) jump S1Failed;
    glk_window_set_echo_stream(gg_mainwin, gg_scriptstr);
    GL__M(##ScriptOn, 2);
    VersionSub();
    return;
    .S1Failed;
    GL__M(##ScriptOn, 3);
];
```

§33. Switch Transcript Off Rule.

```
[ SWITCH_TRANSCRIPT_OFF_R;
    if (actor ~= player) rfalse;
    if (gg_scriptstr == 0) return GL__M(##ScriptOff,1);
    GL__M(##ScriptOff, 2);
    glk_stream_close(gg_scriptstr, 0); ! stream_close
    gg_scriptstr = 0;
];
```

§34. Announce Story File Version Rule.

```
[ ANNOUNCE_STORY_FILE_VERSION_R ix;
    if (actor ~= player) rfalse;
    Banner();
    print "Identification number: ";
    for (ix=6: ix <= UUID_ARRAY->0: ix++) print (char) UUID_ARRAY->ix;
    print "^";
    @gestalt 1 0 ix;
    print "Interpreter version ", ix / $10000, ".", (ix & $FF00) / $100,
    ".", ix & $FF, " / ";
    @gestalt 0 0 ix;
    print "VM ", ix / $10000, ".", (ix & $FF00) / $100, ".", ix & $FF, " / ";
    print "Library serial number ", (string) LibSerial, "^";
    #Ifdef LanguageVersion;
    print (string) LanguageVersion, "^";
    #Endif; ! LanguageVersion
    ShowExtensionVersions();
    say__p = 1;
];
```

§**35. Descend To Specific Action Rule.** There are 100 or so actions, typically, and this rule is for efficiency's sake: rather than perform 100 or so comparisons to see which routine to call, we indirect through a jump table. The routines called are the `-Sub` routines: thus, for instance, if `action` is `##Wait` then `WaitSub` is called. It is essential that this routine not be called for fake actions: in I7 use this is guaranteed, since fake actions are not allowed into the action machinery at all.

Strangely, Glulx's action routines table is numbered in an off-by-one way compared to the Z-machine's: hence the `+1`.

```
[ DESCEND_TO_SPECIFIC_ACTION_R;
    indirect(#actions_table-->(action+1));
    rtrue;
];
```