A Descriptive Language for Video Formats

By Gregory M. Eitzmann and John Hallesy

A language has been developed to reduce the complexity of the description of a video format standard and fully specify the pattern of the video synchronization signal. It also describes the excursions of the video synchronization signal in both horizontal and vertical blanking intervals as well as the relationship between the sync and active portions of the line. The language is processed by a compiler, which is a computer program that derives information and provides it to the hardware that generates electrical output. In this paper, this language is explained both formally and by example.

Although most video engineers deal with one of the NTSC, PAL, or SECAM video standards, the demands of a growing computer and multimedia market create a need for video for cathode ray tubes (CRTs), the formats of which are different than those of their broadcast counterparts. The increasing number of differently sized CRT devices currently available, as well as the introduction of affordable multisync monitors, gives rise to an almost inexhaustible selection of formats. In an attempt to control the abundance of different timings and resolutions, a simple language has been specified to describe a component video signal. The language is presented to a compiler, which is a computer program used to derive detailed information that is customized to control the hardware used to generate a video signal.

This language was designed with several goals in mind. To encompass a wide range of formats, the language had to be flexible enough to describe the intricacies of the vertical blanking interval of RS-170A and yet be simple enough to describe the less complicated and more popular VGA format with a minimum of specification. In addition, it was important to incorporate common video engineering terminology wherever possible. For these reasons, the minimum amount of information required to generate typical video formats needed to be identified.

Presented at the 1994 SMPTE Advanced Television and Electronic Imaging Conference in Chicago (paper no. 28-32) on February 5, 1994. Gregory Eitzmann (who read the paper) and John Hallesy are with Silicon Graphics Computer Systems, Mountain View, CA 94043; e-mail: eitzman@sgi.com. An unedited version of this paper appears in *Unveiling New Technologies and Applications: Proceedings of the SMPTE Advanced Television and Electronic Imaging Conference*, SMPTE, 1994.

The Sync Signal as Basis for Description

Most of the signal has been described once the user fully specifies the transitions of the sync signal. This is done by separating the logical functioning of the sync signal from that of the active region, handling each of them independently within the language. Once separated from the active region, the sync signal has only two states, specified simply as "high" and "low" at sync and blanking levels, respectively (Fig. 1).

The active and sync portions of the video signal are related by a constant factor, described by the durations of Horizontal Front Porch, Horizontal Sync, and Horizontal

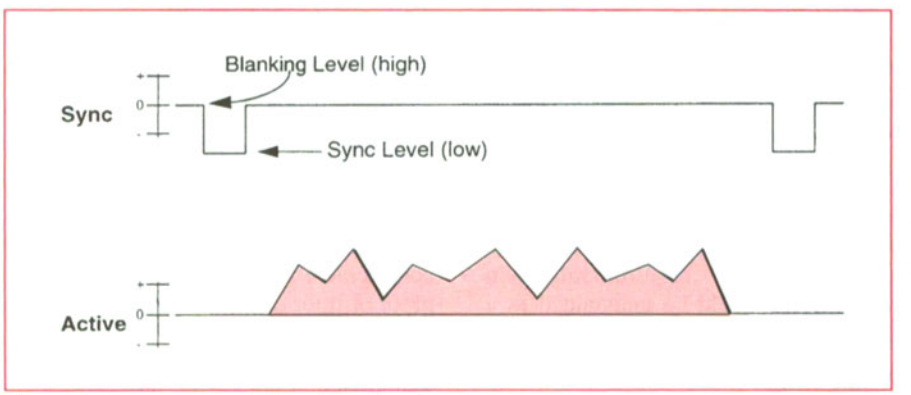

Figure 1. Language terminology for voltage levels.

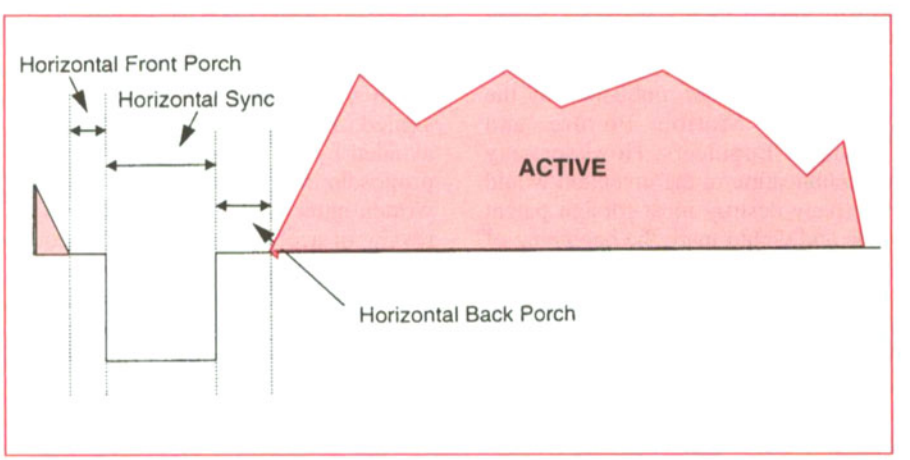

Figure 2. Language terminology for horizontal blanking.

Table 1 — Variables in General Description Section

Variable Name	Description	Units
FieldsPerFrame	The number of fields in one frame of video. For progressive scan displays, this value is 1; for interlaced displays, this value is typically 2.	Integer
FramesPerSecond	The number of frames of video per second. This is usually an integer number, but can contain a fractional component.	Decimal number
TotalLinesPerFrame	The number of lines in the frame, including all vertical blanking lines.	Integer
TotalPixelsPerLine	The number of pixels in one line of video, including all horizontal blanking pixels.	Integer
ActiveLinesPerFrame	The number of lines in the frame that contain picture content (i.e., the number of lines exclusive of vertical blanking lines).	Integer
ActivePixelsPerLine	The number of pixels in the line that contain picture content (i.e., the number of pixels in a line exclusive of horizontal blanking pixels).	Integer
HorizontalBackPorch	The duration of the back porch on each line (the length of time between the rising edge of horizontal sync and the rising edge of active video).	Time
HorizontalFrontPorch	The duration of the front porch on each line (the length of time between the falling edge of active video and the falling edge of horizontal sync).	Time
HorizontalSync	The duration of horizontal sync.	Time

Back Porch (Fig. 2). By using these values, whose meanings are shown in Table 1, it is possible to determine the positions of the start and end of the active portion of the signal. The values used are commonly quoted by SMPTE and CCIR standards, as well as by CRT manufacturers. Note that burst has been removed to simplify the diagram.

The video lines in the vertical interval have no active video associated with them and can thus be described in terms of the sync signal alone.

Language Specification

Each format file describes one frame of video. While this is sufficient to describe component signals, a composite signal would require additional information to implement color frame sequencing. The language for describing a frame of video consists of two major sections.

- *A general description section.* Via a series of algebraic assignment statements, the user provides a general description of the video format by specifying counts and durations of fundamental elements of the frame (such as frames per second or number of active lines per frame). The general description section requires an assignment statement in the form *variable = value.*
- *A sync signal description section.* In this section, the user describes the high/low transitions of the sync signal in every part of the frame.

The variables that require values are shown in Table 1. Table 2 gives an example of these variables as applied to NTSC. The semicolon (;) is required syntax and serves as a statement terminator. The line numbers are artificially supplied.

For any parameters dealing with time, several formats are acceptable:

- *Absolute time.* To enter a time, specify a decimal number with a time unit suffix. Recognized units are sec (seconds), msec (milliseconds), and μsec (microseconds); the suffixes are not case sensitive (e.g., `14.5 usec, 32.0 msec`).
- *Pixels.* To specify time in pixels, an integer number with no suffix or a decimal number with the "Pixels" suffix must be supplied (e.g., `40, 29.2 Pixels`).
- *Lines.* Time durations in fractions of lines can be specified by using the "Lines" or "H" suffix (e.g., `14.3 Lines, 0.5 H`).

• *Predefined variables.* Any of the field names described in Table 1 may be used (e.g., `Horizontal FrontPorch/2`).

Note that the value must be assigned before the variable may be referenced in another equation.

To describe the transitions of the synchronization signal, a construct other than simple assignment statements must be used. The formal grammar is shown in Table 3. The frame is divided into the following functional regions:

• Active — that portion of the frame containing programmatic picture content, i.e., everything except the field blanking interval. For example, PAL contains active video lines 23-310 (field 1) and 336-623 (field 2).

• Vertical Front Porch — the portion of the field blanking interval preceding vertical sync pulses.

• Vertical Sync — those lines in the field blanking interval containing vertical sync pulses (pulses low for a duration greater than the horizontal sync period).

• Vertical Back Porch — the portion of the field blanking interval following vertical sync.

By requiring the user to identify these regions, the language implies information regarding other functions that the target hardware may need to perform during different portions of the frame. These other functions most notably include the generation of video during the active period, but can also include housekeeping operations during the field blanking interval.

Language Example and Explanation

Perhaps the easiest way to understand such a rigid description is by way of example. Table 4 provides a full description of the VGA format; the longer NTSC format is shown in Table 5.

Note the comment delimiters `/*` (begin comment) and `*/` (end comment) on lines 1 and 3, respectively. Comments allow the user to annotate the language with explanatory remarks in a manner similar to the computer programming language C.

Table 2 — General Description Section

```
1       FieldsPerFrame = 2;
2       FramesPerSecond = 29.97;
3       TotalLinesPerFrame = 525;
4       TotalPixelsPerLine = 780;
5       ActiveLinesPerFrame = 486;
6       ActivePixelsPerLine = 646;
7       HorizontalFrontPorch = 1.5 usec;
8       HorizontalSync = 4.7 usec;
9       HorizontalBackPorch = 4.7 usec;
```

Table 3 — Formal Grammar

*sync_description:**
    field_description...
field_description:
    `field` *field_number* = { *field_contents* }
field_contents:
    initial_condition
    region_of_frame = { *sync_transition_set* }
initial_condition:
    `initial` *condition_state*
condition_state:
    `low`
    `high`
region_of_frame:
    `VerticalFrontPorch`
    `VerticalSync`
    `VerticalBackPorch`
    `Active`
sync_transition_set:
    `repeat` *multiplier* { *sync_instruction;* ... }
    { *sync_instruction;* ... }
multiplier:
    integer_expression
sync_instruction:
    initial_condition
    `Length` = *time_value*
    condition_state = *time_value*
time_value:
    integer
    decimal `Pixels`
    decimal `H`
    decimal `Lines`
    decimal `Sec`
    decimal `MSec`
    decimal `Usec`

*The format in this grammar is commonly used in computer science and lists nonterminal tokens followed by a colon. Any one of the indented items following that token are valid solutions to that token. An ellipsis (...) following any item indicates it may be repeated. Items in `special type` indicate text that should be used literally.

Table 4 — The VGA Format

```
1     /*
2      * VGA 640 x 480
3      */
4
5     FieldsPerFrame = 1;
6     FramesPerSecond = 60;
7     TotalLinesPerFrame = 525;
8     TotalPixelsPerLine = 800;
9     ActiveLinesPerFrame = 480;
10    ActivePixelsPerLine = 640;
11
12    HorizontalFrontPorch = 15;
13    HorizontalSync = 3.77 usec;
14    HorizontalBackPorch = 50;
15
16    Field 1 =
17    {
18          Initial High
19          VerticalFrontPorch =
20          {
21                repeat 10 {
22                      Length = 1.0H;
23                      Low = 0 usec;
24                      High = HorizontalSync;
25                }
26          }
27
28          VerticalSync =
29          {
30                repeat 2 {
31                      Length = 1.0H;
32                      Low = 0 usec;
33                }
34          }
35
36          Initial Low
37          VerticalBackPorch =
38          {
39                repeat 33 {
40                      Length = 1.0H;
41                      Low = 0 usec;
42                      High = 3.77 usec;
43                }
44          }
45          Initial High
46          Active = {
47                repeat 480 {
48                      Length = 1.0H;
49                      Low = 0 usec;
50                      High = HorizontalSync;
51                }
52          }
53    }
```

Any text within the comment delimiters is ignored and serves only the user.

White space — the spaces, tabs, and new lines — is ignored and serves only to make the language more readable. The entire frame definition may be placed on a single line if desired.

The focus of the explanation concerns only lines 16 to 53 and describes the transitions of sync. The VGA format contains only one field whose contents are contained within the curly brackets ({ and }) that specify the limits of the definition on lines 17 and 52; the curly bracket plays the containing role in much of this language. If an interlaced format is defined, the definition of the second field would follow the definition of the first.

The single VGA field is identified in its constituent regions: Vertical Front Porch (defined within the brackets on lines 20 to 25), Vertical Sync (lines 29 to 34), Vertical Back Porch (lines 38 to 44), and active (lines 46 to 52). For each region, a series of *sync_instruction* statements are defined. For example, Vertical Front Porch is defined with the statements as shown in Table 4.

- The multiplier of 10 (line 21) indicates the statements enclosed in the brackets on lines 21 and 25 should be repeated ten times. Alternatively, the user could have simply entered the material between the curly brackets ten times. The multiplier serves only as a short cut.
- The length of the sequence (line 22) within the curly brackets is `1.0H`, or one line. The length of the line is defined by TotalPixelsPerLine. Used with the multiplier of 10, the Vertical Front Porch will be ten lines long. Had the length been specified as `0.5H`, Vertical Front Porch would have been only five lines long. Length of frame regions may total a fractional number of lines.
- The low statement (line 23) indicates the vertical sync should be a transition from high (the default state at the beginning of the frame) to low (specified as `0 usec`) at the beginning of the line, as defined within the curly brackets.

Table 5 — The NTSC Format

```
/*
 * NTSC
 */

FieldsPerFrame = 2;
FramesPerSecond = 29.97;
TotalLinesPerFrame = 525;
TotalPixelsPerLine = 780;
ActiveLinesPerFrame = 486;
ActivePixelsPerLine = 646;
HorizontalFrontPorch = 1.5 usec;
HorizontalSync = 4.7 usec;
HorizontalBackPorch = 4.7 usec;

#define SER 67 X 5
#define EQ 2.28 usec

Field 0 =
{
        Initial High
        VerticalFrontPorch =
        {
                repeat 6 {
                        Initial High;
                        Length = 0.5H;
                        Low = 0;
                        High = EQ;
                }
        }

        VerticalSync =
        {
                repeat 6 {
                        Initial High;
                        Length = 0.5H;
                        Low = 0;
                        High = SER;
                }
        }

        VerticalBackPorch =
        {
                repeat 6 {
                        Inital High;
                        Length = 0.5H;
                        Low = 0;
                        High = EQ;
                }

                repeat (20 - 10 + 1) {
                        Initial High;
                        Length = 1.0H;
                        Low = 0;
                        High =
                         HorizontalSync;
                }
        }

        Active = {
                repeat 242 {
                        Inital High;
                        Length = 1.0H;
                        Low = 0;
                        High =
                         HorizontalSync;
                }
                {
                        Inital High;
                        Length = 0.5H;
                        Low = 0;
        High = HorizontalSync;
                }
        }
}       /* End of Field 0 */

Field 1 =
{
        Initial High
        VerticalFrontPorch =
        {
                repeat 6 {
                        Initial High;
                        Length = 0.5H;
                        Low = 0;
                        High = EQ;
                }
        }

        VerticalSync =
        {
                repeat 6 {
                        Initial High;
                        Length = 0.5H;
                        Low = 0;
                        High = SER;
                }
        }

        VerticalBackPorch =
        {
                repeat 6 {
                        Initial High;
                        Length = 0.5H;
                        Low = 0;
                        High = EQ;
                }
                {
                        Initial High;
                        Length = 0.5H;
                }
                repeat (19 - 10 + 1) {
                        Initial High;
                        Length = 1.0H;
                        Low = 0;
                        High =
                         HorizontalSync;
                }
                {
                        Initial High;
                        Length = 0.5H -
                    HorizontalFrontPorch;
                        Low = 0;
                        High =
                         HorizontalSync;
                }

        }

        Active = {
                {
                        /* No sync edge
            transitions needed here. */
                        Inital High;
                        Length = 0.5H +
                    HorizontalFrontPorch;
                }
                repeat 242 {
                        Initial High;
                        Length = 1.0H;
                        Low = 0;
                        High =
                           HorizontalSync;
                }
        }
}
```

Table 6 — Description of Curve Shape and Duration

```
define blanking voltage 0.0
define sync voltage 0.286
define rising edge shape = [ (-pi/2) : 0 ]  (cos(x) *
   cos(x));
define falling edge shape = [ 0 : (pi/2) ]  (cos(x) *
   cos(x));
        .
        .
        .
VerticalFrontPorch =
{
      repeat 6 {
            Initial High,
            Length = 0.5H,
            Rise Time = 0.2 usec
            Low = 0,
            High = 2.28 usec
      }
}
```

• The high statement (line 24) indicates the vertical sync should move from low (where the preceding statement directed it to transition) to high at a time equal to what the user specified in the variable `HorizontalSync`. This could have been alternatively specified as `3.77 usec` (as shown on line 42). Using a predefined variable, as this format does, minimizes the number of times a number must be entered and reduces opportunity for error.

If different kinds of lines are found within each frame region, they can be specified as additional sets of lines within the curly brackets enclosing the region definition.

The several *initial_condition* statements throughout the language aids the user. These statements do not set the initial condition; instead, they force the compiler to ensure the state of the sync signal is as described (e.g., `Initial High` forces a check that the sync signal is high at the time of the check); if the state of sync is other than what is asserted, the compiler produces a warning message. If the user chooses, no *initial_condition* statements need to be specified.

The Compiler

The compiler can be implemented in several different ways. Shown in this paper is a textual interface closely paralleling the actual internal operation. It is also possible to employ the same internal operation, yet have the user interface be graphical and manipulated with the familiar window and mouse mechanisms.

The final stage of the compiler must be closely tied to the hardware that generates the video output. With the information supplied by the user, the compiler can generate a great deal of information needed for a detailed description of internal hardware operation. For example, the compiler can easily generate the pattern for many signals other than strictly sync and active; although most attendant video operations can have their timing derived strictly from the activity of the sync signal, the language provides the user with means to describe without ambiguity the nature of functions of different portions of the frame (e.g., delineation of Vertical Front Porch).

The compiler can also produce outputs other than those strictly required by the hardware, as well as draw detailed timings of the frame. These drawings can be annotated to indicate timings for sync and active signal transitions.

The Hardware

The language itself does not specify any particular hardware directly. Given a flexible enough circuit with which to generate output, the language can be used to specify operation of many different hardware platforms.

It should be noted that although sync excursions are defined to be simply "low" or "high," valid video transitions must be carefully shaped. This can be performed within the compiler by either specifying additional variables in the general description section to outline how the compiler should shape the transitions or by generating it as a hard edge in hardware and filtering to achieve the necessary frequency restrictions.

An example of the former is shown in Table 6, which uses a rise and fall of 0.2 μsec and a curve shape described by $\cos^2 x$. The rising and falling edge curves are optional components of the general description section and show the stage of the curve by supplying an equation for the curve. The range of x values is given in square brackets. The shape and position of the color burst envelope must be similarly described.

The hardware to perform this function has been implemented in a custom circuit called a video output format processor, which was used in the Silicon Graphics RealityEngine graphics hardware.

Other Extensions

One of the most rudimentary forms of the language has been presented here. Additional extensions either implemented or contemplated include an enhanced description of sync state to accommodate trilevel sync required by some HDTV formats, stereo field selection, and color field selection for field-sequential color monitors.