VTT Control Program

# 1 Main Goal

Simplify the definition of control values and the specification of their behavior under scaling (“inheritance”) by combining VTT’s current private “cvt” and “prep” tables into a single private table named “control program”, (contracted from “control values” and “preparation program”) and avoid the typographers having to program in an assembly language. The new control program will be compiled into the current private “cvt” and “prep” tables to minimize architectural changes to the existing VTT and to permit power users to add extra low level code to the “prep” if need really should be. The control program will be declarative in nature, making it easier to learn for non-programmers since it avoids having to sequence commands. At the same time a declarative control program paves the way for its future graphical representation.

The control program allows for a much easier setup of CVTs and their hierarchy (inheritance) and exceptions (deltas) than the previous CVT and pre program.

It is intended to declare:

* CVT values
* Relations between dependent (child) and independent (parent) CVTs
* The PPEM size below which child and parent CVTs have the same values
* Relative or absolute inheritance
* Deltas for CVTs in much the same way as deltas for control points in VTT Talk
* Additional (user-defined) character groups
* Spacing characters for user-defined character groups
* Additional (user-defined) CVT categories
* Global (font-wide) settings for
* The range of PPEM sizes at which instructions are on
* The PPEM size below which drop-out control is on
* Values for CVT Cut-Ins
* CVT attributes, last but not least, without using hexadecimal numbers...

All these declarations are compiled to the actual prep upon compiling the CVT.

# 2 Syntax

### 2.1 User definable attributes

AttributeDeclaration = GroupDeclaration | CategoryDeclaration.

GroupDeclaration = “GROUP” Identifier [ SpacingString ].

CategoryDeclaration = “CATEGORY” Identifier.

Identifier = Char { Char | Digit }.

Char = ‘A’..’Z’ | ‘a’..’z’.

Digit = ‘0’..’9’.

SpacingString = ‘"’ { AsciiChar | ‘^’DecimalNumber’^’ | ‘~’HexadecimalNumber’~’ } ‘"’.

AsciiChar = Any ASCII char except ‘ ’, ‘^’, and ‘~’.

DecimalNumber = Digit { Digit }.

HexadecimalNumber = ‘0’ (‘X’ | ‘x’) HexDigit { HexDigit }.

HexDigit = Digit | ‘A’..’F’ | ‘a’..’f’.

#### Example:

/\* introduce two new character groups \*/

GROUP lowerLatin “nn nono oo ”

GROUP lowerGreek “~0x123~~0x124~ ~0x123~~0x124~”

GROUP lowerCyrillic “^574^^574^ ^574^^576^^574^^576^ ^576^^576^^576^ ”

/\* introduce a new cvt category \*/

CATEGORY flare7

### 2.2 Declarative inheritance and deltas

CvtDeclaration = [AttributeAssociation] ValueAssociation [InheritanceRelation ] [DeltaDeclaration].

AttributeAssociation = Identifier [ Identifier [ Identifier [ Identifier]]].

ValueAssociation = CvtNumber “:” DesignSize

CvtNumber = Number.

Number = Digit { Digit }.

DesignSize = [‘+’ | ‘-‘] Number.

InheritanceRelation = “=” CvtNumber “@” PpemSize.

PpemSize = Number.

DeltaDeclaration = “[”DeltaParameter { “,” DeltaParameter } “]”.

DeltaParameter = DeltaAmount “@” PpemRange.

DeltaAmount = Number.

PpemRange = Range { “;” Range }.

Range = PpemSize [ “..” PpemSize ].

#### Context rules:

1. CvtNumbers must be in range 0 through some upper limit (currently 2047).
2. CvtNumbers must be declared before they can be used in an Inheritance Relation (this avoids “circular inheritance”, i.e. the misleading intention to have cvt 65 control cvt 66, which in turn controls cvt 68, which then attempts to control cvt 65).
3. Cvt attributes (character groups, feature categories) must be declared before they are used in an AttributeAssociation (this is to avoid to inadvertently introduce a new character group or feature category simply by mistyping an old one, and the only way for the compiler to test against this is to clearly separate declaration from usage, and to enforce declaration prior to usage).
4. Cvt attributes apply to all following cvt declarations, until either another cvt attribute, or a default attribute (anyGroup, anyColor, anyDirection, anyCategory), is specified. Specifying no attribute at all will default to the default attribute.
5. PpemSize generally must be in range 1 through some upper limit (currently 255, which corresponds to the values for deltas in the VTT Talk compiler). PpemSize for DropOutControlOff must be in range 0 (for always off) through 255 (for always on). PpemSize for InstructionsOn must be in range 1 through 32767.
6. PpemSize in an Inheritance Relation must be strictly larger than PpemSize in inherited cvt’s Inheritance Relation (it simply doesn’t make sense to have the “grand child cvt” break at a smaller ppem size than the “child cvt”, and the compiler can test against such a typo).
7. PpemSize in a DeltaParameter’s range must be larger or equal to the PpemSize of the Inheritance Relation (again, it doesn’t make sense to specify a delta at a ppem size below the break size, since at that size the respective cvt will be controlled by its parent cvt, along with the parent cvt’s deltas, anyhow, and the compiler can test against such a typo).
8. PpemSizes can only occur once in all the DeltaParameters of a cvt number (it doesn’t make sense to delta a cvt, say, by –1 at 31 ppem, and in a subsequent parameter delta it by +1 in range 30 through 32 ppem, and again, the compiler can test against such a typo).
9. DeltaAmounts must be in range –8 through +8
10. Much like in the current VTT Talk compiler, (fractional) DeltaAmounts may be expressed with decimal places (e.g. 0.5) or as common fractions (e.g. ½).
11. C-style comments (/\* this is a comment \*/) can be used anywhere in the control program in the “usual” way, i.e. anywhere except within multiple character symbols (such as identifiers, numbers).
12. White space (blanks, tabs) and indentation are optional but encouraged to increase legibility.

#### Example:

/\* define cvt-numbers and their “heritage” \*/  
/\* the following cvt declarations will all be anyGroup anyColor anyDirection anyCategory \*/  
65: 200 /\* meaning: cvt 65 has 200 fUnits and isn’t controlled by any other cvt \*/  
66: 200 = 65 @17 /\* meaning: cvt 66 has 200 fUnits, is controlled by cvt 65, and breaks at 17 ppem \*/  
67: 67 = 65 @25 /\* meaning: cvt 67 has 67 fUnits, is controlled by cvt 65, and breaks at 25 ppem \*/

/\* tag subsequent cvts by character group, link color, link direction, and feature category \*/  
uppercase  
black  
x  
straight

/\* the following cvt declarations will all be uppercase black x straight \*/  
68: 192 = 66 @28 [1@29, -1@30..32] /\* meaning: delta by +1 at 29 ppem, -1 at 30 through 32 ppem \*/  
69: 212 = 66 @28 [1@38;40] /\* meaning: delta by +1 at 38 and 40 ppem \*/

### 2.3 User-definable settings

GROUP, COLOR, DIRECTION, CATEGORY,

FPgmBias,

**InstructionsOn**, **DropOutCtrlOff**, **ScanCtrl**, **ScanType**, **CvtCutIn**,

**ClearTypeCtrl**, **LinearAdvanceWidths**,

Delta, BDelta, GDelta, ASM

|  |  |
| --- | --- |
| Turn on instructions | |
| *Syntax*  InstructionsOn @8..2047  *Meaning*  Turns on instructions in range 8 through 2047 ppem.  *Note:* Default range (i.e. if InstructionsOn is not specified) would be 8..2047 ppem | *Code*  #PUSHOFF MPPEM[] #PUSH, 2047 GT[] MPPEM[] #PUSH, 8 LT[] OR[] IF[] #PUSH, 1,1 INSTCTRL[] EIF[] |

|  |  |
| --- | --- |
| Specify scan control and scan types | |
| *Syntax*  (for most users)  DropOutCtrlOff @144  *Meaning*  Turn off drop out control at 144 ppem.  Note: 0 ppem means never do drop out control while 255 ppem means always do drop out control  (for power users wishing to control all the flags)  ScanCtrl = 511  ScanType = 1  Note: Default scan ctrl and type would be 511 (“always”) and 5 (“smart” drop out control).  Note: scan ctrl bits higher than the 8th bit are used to control scan control under rotation and similar situations, which power users may wish to control specifically. | *Code*  SCANCTRL[], (256+144)  Note: 0 ppem will translate to SCANCTRL[], 0 while 255 ppem will translate to SCANCTRL[], 511  SCANCTRL[], 511  SCANTYPE[], 1  *Note:* Scan types 5 and 6 will have to translate to a pair of SCANTYPE[] instructions, the first one with actual type 1 or 2 respectively (old scan type, for backwards compatibility) and the second one with the specified type 5 or 6 (cf. TT specs, TT Instruction Set, note at end of section on scan type) |

|  |  |
| --- | --- |
| Set control value cut in | |
| *Syntax*  CvtCutIn = 4, 1.5@29, 0@128  *Meaning*  set cvt cut-in to 4 pixels, at and above 29 ppem, set it to 1.5 pixels, and at and above 128 ppem set it to 0 pixels (turn it off)  *Note:* Default would be 4 pixels (debatable, but it has to have a noticeable effect at the lowest ppem sizes for the good user experience of the unexperienced user, and we will provide a template anyhow)  *Note 2:* setting storage area 22 to 1 (if cvt cut-in > 0) or 0 (cvt cut-in = 0) retained for backwards compatibility, needed in functions. 64 through 66, which by themselves are called from DStroke and IStroke commands | *Code*  WS[], 22, 1 /\* diag control on \*/ SVTCA[X] SCVTCI[], 256 /\* 4 pixels measured in 64ths \*/ #PUSHOFF MPPEM[] #PUSH, 29 /\* 21 pt @96PPI \*/ GTEQ[] IF[] #PUSH, 96 /\* 1.5 pixels measured in 64ths \*/  SCVTCI[] EIF[] MPPEM[] #PUSH, 128 /\* 96 pt @96PPI \*/ GTEQ[] IF[] #PUSH, 0 SCVTCI[] #PUSH, 22, 0 WS[] /\* diag control off \*/ EIF[] #PUSHON |

|  |  |
| --- | --- |
| Set ClearType™ Native Mode | |
| *Syntax*  ClearTypeCtrl= 1  *Meaning*  Enable native ClearType mode for all TrueType instructions, thus indicating a font that is ClearType aware (and hence the rasterizer should not e.g. bypass delta instructions). See the TrueType Instruction Set documentation and [*Backwards Compatibility of TrueType Instructions with Microsoft ClearType*](http://www.microsoft.com/typography/cleartype/truetypecleartype.aspx) for additional clarification on the meaning of this feature.  If the parameter is set to zero, TrueType will not be in native mode. If the parameter is greater than zero it will be in ClearType native mode. | *Code*  #PUSHOFF  #PUSH, 2, 2 RS[]  LTEQ[] IF[]  #PUSH, 4, 3  INSTCTRL[] EIF[] #PUSHON |

|  |  |
| --- | --- |
| Set Linear Advance Widths | |
| *Syntax*  LinearAdvanceWidths = 1  *Meaning*  LinearAdvanceWidths permits to override the default usage of USE\_INTEGER\_SCALING flag in the flags field of the ‘head’ table. Acceptable values for n are 0 (default) and 1 (don’t use integer scaling).  Non-integer scaling yields advance widths that are closer to the designed advance widths because it does not e.g. take 11 pt. at 96 dpi and round the resulting 14 2/3 ppem up to 15 ppem. Internal to the rasterizer e.g. MPPEM will still return an integer ppem size, but instructions keying off of this ppem size may no longer produce the same results. For instance, at 11 pt, 96 dpi (MPPEM rounds to 15 ppem from 14 2/3), the rasterization may get one pixel pattern, while at 15 pt, 72 dpi (or true 15 ppem), the rasterization may get a different pixel pattern, but since both ppem sizes are reported as the same number, there is no easy way to distinguish.  NOTE: This value can have a serious impact on rendering of text. Adding this to an existing, instructed and proofed font would necessitate reproofing the typeface. | *Code* |

|  |  |
| --- | --- |
| Set Font Program Bias | |
| *Syntax*  FPgmBias = <n>  *Meaning*  Font Program functions used by VTT Talk are offset by the value *n.* E.g. if FDEF | *Code* |

#### Context rules:

1. For InstructionsOn, low ppem size must be smaller than high ppem size.
2. For DropOutControlOff, ppem size must be in range 1 through 256
3. For ScanCtrl, value must be in range 0 through 16383 (bits 14 and 15 reserved for future use)
4. For ScanType, value must be in range 1 through 6
5. For CvtCutIn, pixel amounts must be monotonously decreasing, while ppem sizes must be monotonously increasing
6. For CvtCutIn, the number of cut-ins can be anything from 1 to some upper limit (suggested 4)
7. Each of the 5 settings (InstructionsOn, DropOutControlOff, ScanCtrl, ScanType, and CvtCutIn) can occur at most once. If a setting does not occur, a reasonable default (as noted in the description of the respective setting) will be assumed such as to prevent the font from complete misbehaviour.
8. Cannot have ScanCtrl and ScanType together with DropOutCtrlOff
9. For ClearTypeCtrl, value must be in the range 0 through 1.
10. For LinearAdvanceWidths, value must be in the range 0 through 1.

### 2.4 Other settings

|  |  |
| --- | --- |
| Square aspect ratio | |
| No user input necessary, storage 18 set if current aspect ratio is square, else cleared | *Code*  #PUSH, 18 MPPEM[] SVTCA[Y] MPPEM[] EQ[] WS[] /\* s[18] = 1 X 1 (Square) Aspect Ratio \*/ |

|  |  |
| --- | --- |
| Rendering Type | |
| No user input necessary, storage 2 is set based on the spacing and rendering options currently being used. The following conditions are available:  bit 0 (value 1) = Grey-Scaling bit 1 (value 2) = ClearType(tm) bit 2 (value 4) = Compatible Width ClearType bit 3 (value 8) = Vertical Direction   (horizontally striped) ClearType bit 4 (value 16) = BGR as opposed to RGB  Devices bit 5 (value 32) = ClearType on Rapier CE  Devices bit 6 (value 64) = ClearType with fractional  advance widths bit 7 (value 128) = ClearType with non-  ClearType direction anti-aliasing bit 8 (value 256) = ClearType with gray full-  pixel | *Code*  See fpgm FN 84 |

### 2.5 Loop-hole for dyed-in-the-wool power users

The control program can have multiple ASM("…") commands, much like the VTT Talk compiler has already. This will permit to add extra prep code within an ASM statement, if need really should be. Likewise, for backwards compatibility, the control program compiler will skip compilation if the control program window is empty and if there exists already a prep and a cvt source table. If a user wants to delete an existing prep and cvt source, absent a control program source, he or she will have to do so explicitly. Simply recompiling the empty glyph program won’t do, much like the VTT Talk compiler doesn’t attempt to compile the (empty) VTT Talk of a composite glyph.

# 3 User Interface

### 3.1 General

The control program is a text window, much like the existing cvt and prep windows. The compiler will translate the control program into the prep and cvt tables, and in turn, invoke the respective assemblers to translate the cvt and the prep into their binary forms.

### 3.2 Special: Double specified heights (round vs. square heights)

From a user’s point of view, the inheritance of double specified heights (round vs. square heights, overshoots vs. flat heights, “absolute” vs. “relative” heights, whichever way you call them) are merely a variant of the general inheritance. The (old and new) difference from the general inheritance are the overshoots that are expressed relative to their ancestors. (Notice that the “relative” heights implement the same concept as the links, which move a child point relative to a parent point, just that there need not be a control point at the “parent” height. Therefore, the overshoots will always be “relative” numbers by concept, rather than by “ignorance”.)

#### Example:

/\* define cvt-numbers for heights and their “heritage” \*/  
uppercase  
grey  
y  
absolute  
2: 1466 /\* the caps line \*/  
8: 0 /\* the base line \*/  
relative  
3: 26 = 2 @42 /\* meaning: cvt 3 has 26 fUnits more than cvt 2, and breaks away from cvt 2 at 42 ppem \*/  
9: -26 = 8 @42 /\* meaning: cvt 9 has 26 fUnits less than cvt 8, and breaks away from cvt 8 at 42 ppem \*/

# 4 Self-contained Example

/\* Visual TrueType “standard” control program template \*/

CATEGORY Stroke  
CATEGORY StrokeInheritance

InstructionsOn @8..2047  
DropOutControlOff @144  
CvtCutIn = 4, 1.5@29, 0@128  
ClearTypeCtrl=1  
LinearAdvanceWidths=0  
  
/\* introduce two new character groups \*/  
GROUP lowerGreek  
GROUP lowerCyrillic  
/\* define more character groups here as desired \*/  
  
/\* introduce a new feature category \*/  
CATEGORY flare7  
/\* define more cvt categories here \*/  
  
/\* define height cvt-numbers and their “heritage” \*/  
uppercase  
grey  
y  
absolute  
2: 1466 /\* the caps line \*/  
8: 0 /\* the base line \*/  
relative  
3: 26 = 2 @42 /\* meaning: cvt 3 has 26 fUnits more than cvt 2, and breaks away from cvt 2 at 42 ppem \*/  
9: -26 = 8 @42 /\* meaning: cvt 9 has 26 fUnits less than cvt 8, and breaks away from cvt 8 at 42 ppem \*/  
/\* define more heights here \*/  
  
/\* the following cvts needed only for an italic font \*/  
uppercase  
grey  
x  
italicRun  
 36: 307  
y  
italicRise  
 37: 1466  
  
/\* reset all the cvt attributes \*/  
anyGroup  
anyColor  
anyDirection  
anyCategory  
/\* define general stroke cvt-numbers and their “heritage” \*/  
65: 200 /\* meaning: cvt 65 has 200 fUnits and isn’t controlled by any other cvt \*/  
66: 200 = 65 @17 /\* meaning: cvt 66 has 200 fUnits, is controlled by cvt 65, and breaks at 17 ppem \*/  
 67: 67 = 65 @25 /\* meaning: cvt 67 has 67 fUnits, is controlled by cvt 65, and breaks at 25 ppem \*/  
  
/\* tag subsequent cvts by character group, link color, link direction, and feature category \*/  
upperCase  
black  
x  
straight  
/\* the following cvt declarations will all be uppercase black x straight \*/  
68: 192 = 66 @28 [1@29, -1@30..32] /\* meaning: delta by +1 at 29 ppem, -1 at 30 through 32 ppem \*/  
 69: 212 = 66 @28 [1@38;40] /\* meaning: delta by +1 at 38 and 40 ppem \*/  
round  
/\* the following cvt declaration will be uppercase black x round \*/  
 70: 200 = 66 @25  
  
/\* more cvt declarations here… \*/  
  
lowerGreek  
black  
x  
round  
/\* the following cvt declarations will all be lowercase Greek black x round \*/  
122: 60 = 66 @32 /\* lowercase Greek controlled by cvt 66 and breaks at 32 ppem in this example \*/  
123: 70 = 66 @32 /\* alternate round stroke for lowercase Greek \*/  
  
/\* even more cvt declarations here…\*/