

2021-12-20

Universal Multiple-Octet Coded Character Set
International Organization for Standardization
Organisation Internationale de Normalisation
Международная организация по стандартизации

Doc Type: Working Group Document
Title: Proposal to add characters from Smalltalk to the UCS
Source: Terminals Working Group
Authors: Rebecca Bettencourt, Doug Ewell, Ricardo Bánffy, Michael Everson, Jarkko Hietaniemi, Eduardo Marín Silva, Elias Mårtenson, Mark Shoulson, Shawn Steele, and Rebecca Turner
Status: Individual Contribution
Action: For consideration by JTC1/SC2/WG2 and UTC
Date: 2021-12-20

1. Introduction. This document proposes the addition to the UCS of 5 new characters to provide compatibility with versions of the Smalltalk programming language originally developed throughout the 1970s.

***NOTE:** Mapping tables between Smalltalk character sets and the allocations in this proposal are attached to the PDF version of this document.*

2. Precedent. Characters from the programming language APL were encoded in the UCS in 1993 (Unicode 1.1) for compatibility with established IBM character sets. One character (U+23E8 DECIMAL EXPONENT SYMBOL) was encoded in 2009 (Unicode 5.2) for compatibility with the ALGOL 60 programming language.

3. Smalltalk. *Smalltalk* is an object-oriented programming language designed and implemented at Xerox PARC by Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Diana Merry, Scott Wallace, and others during the 1970s. It is one of the most historically significant programming languages: it has influenced the creation of Java, Objective-C, Python, Ruby, and many other object-oriented languages, and Smalltalk environments were often the first to develop modern software engineering concepts such as the model-view-controller (MVC) pattern, the graphical user interface (GUI), and the modern integrated development environment (IDE). Many variants of Smalltalk are still in active development and have gathered loyal communities of users.

Early versions of the Smalltalk language in development throughout the 1970s used several novel characters, some of which are still not included in the UCS. Recently, Dan Ingalls has written a paper covering the history of Smalltalk for the fourth History of Programming Languages (HOPL) conference which includes these characters.

Smalltalk uses proportional fonts, and its symbols closely resemble mathematical operators.

4. ZWJ sequences. Smalltalk included three atomic characters which resemble character sequences that can already be represented. For round-trip compatibility, we recommend the use of a zero-width joiner within the corresponding character sequence.

- 's APOSTROPHE S OPERATOR is an identifier for “a generic accessor that you can think of as foo’s value or foo’s caller.” Smalltalk also included U+0027 APOSTROPHE, which began and ended a character string or comment, and U+2019 RIGHT SINGLE QUOTATION MARK, which was used as an apostrophe inside comments. Without the zero-width joiner, the equivalent character sequence would be misinterpreted as a comment beginning with a lowercase s, a premature end of comment followed by a token beginning with s, or a syntax error (the apostrophe used in comments incorrectly appearing in code). For this character we recommend the character sequence:
 - U+2019 RIGHT SINGLE QUOTATION MARK
 - U+200D ZERO WIDTH JOINER
 - U+0073 LATIN SMALL LETTER S
- () LEFT AND RIGHT PARENTHESIS represents a suppressed subarray (a subarray with its contents not displayed) within its containing array. U+0028 LEFT PARENTHESIS and U+0029 RIGHT PARENTHESIS are used as one might expect for array notation. Without the zero-width joiner, the equivalent character sequence would be misinterpreted as an empty array, preventing the interpreter from correctly identifying the input of a suppressed subarray (perhaps copied and pasted from output) as an error (missing information). For this character we recommend the character sequence:
 - U+0028 LEFT PARENTHESIS
 - U+200D ZERO WIDTH JOINER
 - U+0029 RIGHT PARENTHESIS
- →(RIGHTWARDS ARROW TO LEFT PARENTHESIS is used to inject literal objects into code. For example, (3 + →(28 / 7)) would be parsed as (3 + 4) ; the number 4 in this example could just as easily be an array, image, or other object. Smalltalk did not include U+2192 RIGHTWARDS ARROW as an independent character, so there is no round-tripping issue in this case: a lone U+2192 RIGHTWARDS ARROW without U+0028 LEFT PARENTHESIS would be an encoding error. However, for consistency, for this character we recommend the character sequence:
 - U+2192 RIGHTWARDS ARROW
 - U+200D ZERO WIDTH JOINER
 - U+0028 LEFT PARENTHESIS

Alternative 1: Encoding as the corresponding character sequence without a zero-width joiner. This would not be acceptable due to the semantic differences between the atomic characters and the character sequences they resemble as explained above.

Alternative 2: Encoding as atomic characters in Unicode. This has been repeatedly and consistently rejected by the Script Ad Hoc during review of previous drafts of this proposal.

5. Characters not proposed. An earlier draft of this proposal included a character, ZERO WIDTH ENCLOSING CIRCLE, which would overlay the *following* character (in contrast to U+20DD COMBINING ENCLOSING CIRCLE which overlays the *preceding* character). This corresponded to Smalltalk-72 character 0x26. The character could not have been encoded as it violates the Unicode model for combining characters. It is suggested that U+20DD be used instead, with the order of the characters swapped. An alternative is the use of existing mathematical operators such as U+2295 CIRCLED PLUS and U+229B CIRCLED ASTERISK OPERATOR in place of character sequences starting with ZERO WIDTH ENCLOSING CIRCLE.

6. Character names. At least since the 1970s, international SDOs such as ECMA and national bodies such as ANSI and BSI have assigned names to the elements of coded character sets. By contrast, although the Smalltalk developers did assign names to most of the characters in the Smalltalk character set, the names used are not necessarily conformant to the guidelines of the present day, and a handful of other characters were left without names (Figure 3). We have attempted to invent names for these characters that are meaningful, unique, and conformant to WG2 and UTC guidelines.

7. Ordering and code point assignment. All characters (with the exception of an arrow which seemed to fit logically within an existing block) are shown here with a suggested code point in a new block (1CEB0..1CEFF) that is unassigned and adjacent to an existing symbol block, according to the “Roadmap to the SMP,” revision 13.0.3. A placeholder block name, “Miscellaneous Mathematical and Technical Symbols,” is listed in the summary form. However, it is understood that final assignment of blocks, code points, and block and character names is completely at the discretion of UTC and/or WG2.

8. Unicode character properties.

```
1CEB0;HORIZONTAL ZIGZAG LINE;So;0;ON;;;;N;;;;;
1CEB1;KEYHOLE;So;0;ON;;;;N;;;;;
1CEB2;OLD PERSONAL COMPUTER WITH MONITOR IN PORTRAIT ORIENTATION;So;0;ON;;;;N;;;;;
1CEB3;BLACK RIGHT TRIANGLE CARET;So;0;ON;;;;N;;;;;
1F8B2;RIGHTWARDS ARROW WITH LOWER HOOK;So;0;ON;;;;N;;;;;
```

9. References.

- Goldberg, Adele and Kay, Alan. 1976. “Smalltalk-72 Instruction Manual.” Xerox Corporation.
http://www.textfiles.com/bitsavers/pdf/xerox/alto/Smalltalk72_Manual.pdf
- Ingalls, Daniel. Proceedings of the ACM on Programming Languages volume 4 number HOPL.
June 2020. “The evolution of Smalltalk: from Smalltalk-72 through Squeak.”
<https://dl.acm.org/doi/abs/10.1145/3386335>
- Wikipedia. 2020. “Smalltalk.” <https://en.wikipedia.org/wiki/Smalltalk>

10. Disclaimer. All trademarks and registered trademarks mentioned herein are the property of their respective owners. The company and product names used in this document are for identification purposes only.

	1CEB	1CEC	1CED	1CEE	1CEF
0	 1CEB0				
1	 1CEB1				
2	 1CEB2				
3	 1CEB3				
4					
5					
6					
7					
8					
9					
A					
B					
C					
D					
E					
F					

Smalltalk

- 1CEB0  HORIZONTAL ZIGZAG LINE
 → 299A  vertical zigzag line
- 1CEB1  KEYHOLE
 = peek
 → 1F5DD  old key
- 1CEB2  OLD PERSONAL COMPUTER WITH MONITOR IN PORTRAIT ORIENTATION
 → 1F5B3  old personal computer
- 1CEB3  BLACK RIGHT TRIANGLE CARET
 • zero-advance character pointing between two glyphs
 → 2038  caret
 → 2333  slope
 → 1CC86  white lower left pointer

	1F80	1F81	1F82	1F83	1F84	1F85	1F86	1F87	1F88	1F89	1F8A	1F8B	1F8C	1F8D	1F8E	1F8F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

Smalltalk

1F8B2 ⇨ RIGHTWARDS ARROW WITH LOWER HOOK
→ 21AA ⇨ rightwards arrow with hook

Figures.

Figures showing code charts of Smalltalk fonts are presented first, followed by examples of usage and other illustrations.

0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
☒	⌘	⌘	⌘	☒	⌘	@	•	☒	☒	☒	⌘	☒	☒	≠	"
0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
⌘	!	⌘	's	⌘	-	%	v	^	➡	≥	\$	&	§	!	■
0020	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
	↑	☞	#	⌘	⌘	⌘	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	⇒
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
☺	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
☒	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	007F
p	q	r	s	t	u	v	w	x	y	z	{	'	}	?	⚡

Figure 1. Code chart of a system font extracted from a Smalltalk-72 instance, with U+1CEB3 BLACK RIGHT TRIANGLE CARET, U+1CEB1 KEYHOLE, RIGHTWARDS ARROW TO LEFT PARENTHESIS, APOSTROPHE S OPERATOR, U+1CEB2 OLD PERSONAL COMPUTER WITH MONITOR IN PORTRAIT ORIENTATION, and LEFT AND RIGHT PARENTHESIS highlighted in red.

0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
⧫	≤	⧫	∞	⧫	↗	≡	○	⧫	⧫	⧫	⧫	⧫	⧫	≠	↷
0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
⧫	↑	≥	Ⓟ	Ⓛ	—	↙	⊙	⚡	➡	⧫	➡	⧫	□	└	⧫
0020	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
⧫	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	007F
p	q	r	s	t	u	v	w	x	y	z	{		}	~	⧫

Figure 2. Code chart of a Smalltalk-78 system font extracted from a prototype Macintosh system disk, with U+1F8B2 RIGHTWARDS ARROW WITH LOWER HOOK, APOSTROPHE S OPERATOR, U+1CEB0 HORIZONTAL ZIGZAG LINE, and U+1F8B6 NEGATIVE SQUARED RIGHTWARDS ARROW highlighted in red.

Keyboard Equivalents

(Note, there are usually several ways to type a special keyboard character. The following table presents the methods most commonly used.)

To Get	You Type	We Call It
-----	-----	-----
!	LF	do it
☞	<shift> '	hand
⦿	<shift> 5	eyeball (look for)
⦿	<ctrl><shift>;	
⦿	<ctrl> k	keyhole, "peek"
⇒	<shift> /	if ... then
↑	<shift> 1	return
☺	<shift> 2	smiley
☐	<shift> 7	
?	<ctrl> ?	
Ⓢ	<ctrl> s	
done!	<ctrl> d	
-	<shift> -	unary minus
≤	<ctrl> <	less than or equal
≥	<ctrl> >	greater than or equal
≠	<ctrl> =	not equal
%	<ctrl> v	percent sign
@	<ctrl> 2	"at" sign
!	<ctrl> 1	explanation
"	<ctrl> o	double quote sign
\$	<ctrl> 4	dollar sign

Figure 3. Excerpt from the Smalltalk-72 Instruction Manual showing U+1CEB1 KEYHOLE and APOSTROPHE S OPERATOR, highlighted in red.

```

to rectangle a b c / origin extent
  ( <has>
    ( <@c + :. <@origin c origin + extent)
    <@s=>(<@ % eval)
    <@comp>
      ( dcomp origin x extent x origin y extent y)
    <@clear>
      ( dclear origin x extent x origin y extent y :)
    <@intersect>
      ( <@c + :.
        <@a + origin max c's origin.
        <@b + (origin + extent) min c's(origin + extent).
        a <= b => (<@rectangle a b - a) <@false)
    <@include>
      ( <@c + :.
        <@a + origin min c's origin.
        <@b + (origin + extent) max c's(origin + extent).
        <@rectangle a b - a)
    <@moveto => (<@origin + :)
    <@frame =>
      ( <@a + turtle.
        a penup goto origin turn 90 pendn's width + 2.
        a penup goto origin turn 90 pendn's width + 2.
        do 2 (a go extent x turn 90 go extent y turn 90))
    <@is =>(<@ISIT eval)
    <@print =>
      (<@rectangle print sp origin print sp extent print)
    <@paint => (CODE 41)

    isnew => (<@origin + :. <@extent + :))!

to waitnext x
  (<@x + %.
    repeat (x eval => ( ) done)
    repeat (x eval => (done)))!

to bug
  (waitnext button. <@mp)!

```

Is a point inside rectangle?

Expects bit patterns as a message

Creates a rectangle that is the intersection of c and SELF if they have common area else, 'false'.

Creates rectangle around SELF and c.

Move origin to a new point.
Turtles understand how to go to a point as well as two numeric coordinates.

This message was discussed in Chapter II section on Paint Brush.

Stay in this routine until x is first 'false' and then finally 'not-false' again.

Wait to get the mouse point when button 1 is pressed.

Figure 4. Excerpt from the Smalltalk-72 Instruction Manual showing two instances of APOSTROPHE S OPERATOR, highlighted in red.

to dispframe input : winx winwd winy winht frm x frmwd frm y frmht last
 mark lstln char x char y reply justify buf font editor : sub frame dread reread
 defont (

```
...
@'s => (↑ 8 eval)
  'Allows access to instance variables. For example,
    yourframe's (⌘ winx←32)
    will alter the value of window x in the instance of dispframe
    called ⌘ yourframe⌘.'
...)
```

Figure 5. Smalltalk-72 code containing both U+0027 APOSTROPHE (highlighted in green) and APOSTROPHE S OPERATOR (highlighted in red and blue).

```
Welcome to SMALLTALK [May 30]
⌘
Control-B: _ as in abcdef
ampersand: ○ as in ⊕ ⊖ ⊗
```

Figure 6. Example Smalltalk-72 session showing U+1CEB2 OLD PERSONAL COMPUTER WITH MONITOR IN PORTRAIT ORIENTATION and U+1CEB3 BLACK RIGHT TRIANGLE CARET (between lowercase c and d), highlighted in red.

A. Administrative

1. Title

Proposal to add characters from Smalltalk to the UCS

2. Requester's name

Terminals Working Group (Rebecca Bettencourt et al.)

3. Requester type (Member body/Liaison/Individual contribution)

Individual contribution.

4. Submission date

2021-12-20

5. Requester's reference (if applicable)

6. Choose one of the following:

6a. This is a complete proposal

Yes.

6b. More information will be provided later

No.

B. Technical - General

1. Choose one of the following:

1a. This proposal is for a new script (set of characters)

Yes.

1b. Proposed name of script

Miscellaneous Mathematical and Technical Symbols.

1c. The proposal is for addition of character(s) to an existing block

No.

1d. Name of the existing block

2. Number of characters in proposal

5.

3. Proposed category (A-Contemporary; B.1-Specialized (small collection); B.2-Specialized (large collection); C-Major extinct; D-Attested extinct; E-Minor extinct; F-Archaic Hieroglyphic or Ideographic; G-Obscure or questionable usage symbols)

Category B.1.

4a. Is a repertoire including character names provided?

Yes.

4b. If YES, are the names in accordance with the "character naming guidelines" in Annex L of P&P document?

Yes.

4c. Are the character shapes attached in a legible form suitable for review?

Yes.

5a. Who will provide the appropriate computerized font (ordered preference: TrueType, or PostScript format) for publishing the standard?

Rebecca Bettencourt.

5b. If available now, identify source(s) for the font (include address, e-mail, ftp-site, etc.) and indicate the tools used:

Rebecca Bettencourt, FontForge.

6a. Are references (to other character sets, dictionaries, descriptive texts, etc.) provided?

Yes.

6b. Are published examples of use (such as samples from newspapers, magazines, or other sources) of proposed characters attached?

Yes.

7. Does the proposal address other aspects of character data processing (if applicable) such as input, presentation, sorting, searching, indexing, transliteration, etc. (if yes please enclose information)?

Yes.

8. Submitters are invited to provide any additional information about Properties of the proposed Character(s) or Script that will assist in correct understanding of and correct linguistic processing of the proposed character(s) or script.

See above.

C. Technical - Justification

1. Has this proposal for addition of character(s) been submitted before? If YES, explain.

The characters were previously proposed in the first draft of "Proposal to add further characters from legacy computers and teletext to the UCS." They were moved to a separate proposal at the request of the Script Ad Hoc.

2a. Has contact been made to members of the user community (for example: National Body, user groups of the script or characters, other experts, etc.)?

Yes.

2b. If YES, with whom?

Smalltalk user community (Vanessa Freudenberg, Dan Ingalls)

2c. If YES, available relevant documents

3. Information on the user community for the proposed characters (for example: size, demographics, information technology use, or publishing use) is included?

Contemporary use by specialists and hobbyists.

4a. The context of use for the proposed characters (type of use; common or rare)

Rare.

4b. Reference

5a. Are the proposed characters in current use by the user community?

Yes.

5b. If YES, where?

Worldwide, but particularly in North America.

6a. After giving due considerations to the principles in the P&P document, must the proposed characters be entirely in the BMP?

No.

6b. If YES, is a rationale provided?

6c. If YES, reference

7. Should the proposed characters be kept together in a contiguous range (rather than being scattered)?

Mostly yes, but this is not required.

8a. Can any of the proposed characters be considered a presentation form of an existing character or character sequence?

No.

8b. If YES, is a rationale for its inclusion provided?

8c. If YES, reference

9a. Can any of the proposed characters be encoded using a composed character sequence of either existing characters or other proposed characters?

No.

9b. If YES, is a rationale for its inclusion provided?

9c. If YES, reference

10a. Can any of the proposed character(s) be considered to be similar (in appearance or function) to an existing character?

No.

10b. If YES, is a rationale for its inclusion provided?

10c. If YES, reference

11a. Does the proposal include use of combining characters and/or the use of composite sequences (see clauses 4.12 and 4.14 in ISO/IEC 10646-1:2000)?

No.

11b. If YES, is a rationale for such use provided?

11c. If YES, reference

11d. Is a list of composite sequences and their corresponding glyph images (graphic symbols) provided?

11e. If YES, reference

12a. Does the proposal contain characters with any special properties such as control function or similar semantics?

No.

12b. If YES, describe in detail (include attachment if necessary)

13a. Does the proposal contain any Ideographic compatibility character(s)?

No.

13b. If YES, is the equivalent corresponding unified ideographic character(s) identified?