Xerox Corporation
3333 Coyote Hill
Palo Alto, California 94304
415 494-4000

March 19, 1982

XEROX

Paul McCullough
Tektronix
Mail Stop 92-805
P. O. Box 500
Beaverton, Oregon 97077

Dear Paul:

Here we go again, for the last time. Thank you for your patience and help through this experience.

Enclosed you will find another tape. It is different from the last one we sent, both in terms of the image itself and small changes (you requested) in the virtual machine. Barring major catastrophe, it will be the same as the tape we want Xerox to license. Changes in this image are described below.

User Guide

Included in this package is a draft of part 1 of The Smalltalk-80 System: a user guide and reference manual. You have already received one version of this, the one enclosed should match the image on the tape. We would appreciate comments on this manual, especially on omitted user interface information.

Please do not make copies of this guide; it is only a draft. Our publisher has agreed to make many copies available when it is in its final form. Let us know if you need copies of the published version.

License Status

You may be interested in the status of the Smalltalk-80 licenses. So are we.

With respect to your license, we are at the point where we would like to finish that process. The only thing that is left is to satisfy the "laying of hands" condition; that Dan see and approve your implementation. We would like to set the date of June 1, 1982 as the deadline for this. Please arrange with us for an appropriate time to have Dan come. If you have any questions regarding your continued use of the system, please do not hesitate to ask. The primary principle is that you cannot distribute the

virtual image independent of its implementation on your company's hardware, and independent of the sale of that hardware. Your implementation of the virtual machine can be freely distributed.

First Book Status

XEROX

You may not recognize the "book". The system changed so much that Adele and Dave started over again, keeping well in mind all of your hearty criticisms. Fourteen chapters have been distributed within Xerox for review, and the rest will be distributed next week. We will send a draft to the publishers, Addison-Wesley, in April for design work. You will be given a pre-publication version, when it is available.

Implementors' Book

As we discussed at the implementors convention, many of us are still interested in compiling a collection of papers based on your experiences implementing the Smalltalk-80 system(s) into a book for publication. Because others will soon be implementing the system, the book ought to happen soon as well. How about the following deadlines:

May 1 -- we should have the titles, outlines and page estimates of your articles,

June 1 -- we should have a draft of your completed articles.

We have a technical editor available to assist in editing drafts. Let us know the extent of your interest and whether you can meet the above schedule.

Changes to the System

The current image is different from the last one, otherwise why would we have bothered? There are three minor Virtual Machine changes, and some minor changes in the user interface.

The Virtual Machine changes are by popular demand. The first is optional and the other two were de facto in the last release. Since they are so small, they will be documented here, rather than us sending you another draft of the implementation chapters now. They are the addition of

primitive 105: <String> replaceFrom:to:with:startingAt:, which allows faster string copying and replacement;

primitive 115: <SystemDictionary> oopsLeft, which returns the number of unallocated object pointers in the system; and

primitive 116: <SystemDictionary>

signal:atOopsLeft:wordsLeft:, which sets the threshhold for memory, below which a semaphore is to be signalled.

Note that these primitives actually were in the last image, but had different numbers.

We should also mention that the image contains only one primitive with number greater than 127, which is the Alto file primitive, to enable the sample Alto file system to include everything.

XEROX

Changes to the user interface include further commenting of methods and classes, an 'explain' feature, and various improvements to make the system and user guide match. Enjoy.

Again, thank you for all the help you have provided us in making the Smalltalk-80 system a reality. We look forward to seeing your implementations, reading your papers, and working with you in the future.

Sincerely,

Glenn

All of us at SCG,
Glenn Krasner, scribe

Inter-Mittent Memorandum

To	Smalltalk-80 Implementors	Date	March 19, 1982
From	Glenn Krasner	Location	XEROX PARC
Subject	Trademark of "Smalltalk-80"	Organization	SCG

PS. You may be interested in a little clarification about our trademark for the term "Smalltalk-80". Xerox has applied for federal trademark registration for that term, and intends to protect it as a trademark.

What this means is that you do not have the right to use this term. However, your license (which goes into effect after Dan blesses your system) does allow you to use the term when referring to the system or implementation (e.g. "Smalltalk-80 System") as long as you indicate that "Smalltalk-80" is a trademark of Xerox Corporation. In addition, as a trademark "Smalltalk-80" must be used as a adjective.

Now you understand and will not unintentionally misuse the term "Smalltalk-80", right?

If you have any questions about this, just ask. We almost understand.

Inter-Mittent Memorandum

To	Smalltalk-80 Implementors	Date	March 17, 1982
From	Glenn Krasner	Location	XEROX PARC
Subject	Smalltalk-80 Specifics not covered in the book	Organization	SCG

Scope

The Smalltalk book contains information necessary to implement Smalltalk, including the list of bytecodes and their implementations, the list of primitives and their implementations, and a description of storage management. The book, however, does not contain specifics for any given Smalltalk-80 virtual memory image, and in particular the image accompanying this memo. This memo is intended to supply those specifics.

Contained below are a description of the file formats for the tape, plus a list of those objects and classes known to the interpreters.

What is on the Tape

The Smalltalk-80 Virtual Memory Image, and associated files, are written on a 9-track, 1600 bpi phase-encoded magnetic tape. The tape consists of binary files in 'continuous stream' mode, with 512 byte records, and an eof mark after each file. The files (in order of appearance) are:

1) Virtual Memory Image (copy 1)	1011 records,
2) Virtual Memory Image (copy 2)	1011 records,
3) Sources file (in the image, called Smalltalk80.sources')	2425 records,
4) Changes file (in the image, called Smalltalk80.changes')	1 record,
5) Trace file 1 of simulator	58 records,
6) Trace file 2 of simulator	39 records,
7) Trace file 3 of simulator	43 records,
8) List of object pointers for classes	27 records,
9) List of object pointers for methods	332 records,

Order of Bytes

All bytes are considered 8-bits, all words are 16-bits. Words in the file are stored in the order of more significant byte followed by less significant byte. We realize that some implementations would prefer to have words stored low byte followed by high byte. We think that the transformation of the image that would work for most such machines is to swap the bytes of all fields accessed as words, and to not swap the fields accessed as bytes.

For word-type objects: swap every field.
For CompiledMethods: swap Length, Class, Header and Literal fields only.
For all other byte-type objects: swap Length and Class fields only.

THE FILES

Virtual Memory Image File

There are two copies of the Virtual Memory Image file. The Virtual Memory Image consists of length information, followed by the data representing objects (object space), followed by the data representing the object table. The first word (stored as most significant byte first) contains the high-order 16 bits of the length of the object space, and the second contains the low-order 16 bits of this length. The third word of the file is the high-order 16 bits of the length of the object table, followed by the low-order 16 bits. The next 252 words are set to 0. By convention, an image file is defined to be in interchange format if the fifth word (ninth and tenth byte) is zero.

For this image, the first ten bytes are:
0, 3, 170_8, 307_8, 0, 0, 170_8, 272_8, 0, 0.

Following this (starting at the 257th word) is the object space. The first word encountered is the first word of the object whose object pointer (oop) is 2 and whose object space address is 0 (20 bit address). The next words are the fields of the other objects, stored consecutively, up to the length of the object space. Following the object space are enough 0's to start the object table at a page (256-word) boundary.

The next word is the first word of the object table entry for the object with oop (object pointer) = 0. (Which, of course is an invalid oop, but the object table entry exists anyway.) This is followed by the rest of the words of the object table. The last word of the object table is the last word of the file.

> Note:
> --The length and object space portions of the file are padded with 0's to the end of a page, but the object table is not.
> --The object table may contain unused entries. These have the 'freeEntry' bit set, but all other bits in both words are 0. Implementors will have to link these themselves, if desired.
> --The object table entries either point to objects in the object space or are free entries, and the object space contains only objects; there is no free memory in the object space and no entry in the object table for free memory blocks.
> --The object table assumes that objects are stored contiguously starting at address 0 in a 20-bit address space. There is no distinction for "segment" boundaries. Suitable address translation as necessary is up to the implmentors.

Sources and Changes Files

The third file of the tape is a copy of the system sources. Print it. This file consists of the definition of each class in the system, followed by the code for that class's methods. (There is a form feed character between each class, ASCII 12). The format of this file may be understood by reading the source code for the nextChunk method in class ReadWriteStream and the getSource method in class CompiledMethod; it is the same format as that used in fileIn/fileOut.

Each CompiledMethod in the image contains a pointer to its source code, encoded in the last three bytes of that method. The two msb's of the first of these bytes determine the file on which the source is stored (00=Smalltalk80.sources', 01=Smalltalk80.changes', 10=unused, 11=unused). The six lsb's of this byte with the two following bytes make up a 22-bit pointer specifying where in that file the source code begins. The source code for that method is terminated by $! (imbedded $!s are doubled).

> Note:
> The sources file on this tape is quite long, about 350 pages when printed; and the changes file has only one expression on it.

Trace Files

The fifth, sixth, and seventh files on the tape contain three traces of the Smalltalk-80 interpreter executing the first bytecodes in this Smalltalk-80 virtual image. These were made by running the formal specification of the interpreter written in Smalltalk-80 itself. The three traces show decreasing levels of detail over increasing durations.

- The first trace shows all memory references, allocations, bytecodes, message transmissions, returns and primitive invocations for the first 100 bytecodes executed.

- The second trace shows only the bytecodes, message transmissions, returns and primitives for the first 424 bytecodes.

- The third trace shows message transmissions, primitives and returns for the first 1985 bytecodes. The lines of this trace are indented according to the level of method invocation (i.e., the depth of the context stack).

The format of each type of entry is given below. All numbers are shown in decimal.

Memory Reference (only in first trace)

Pointer Fetch

object-pointer pointer: *fieldIndex* = *field contents*
(e.g., 20656 pointer: 20 = 1617)

Byte Fetch

object-pointer byte: *byteIndex* = *byte contents*
(e.g., 3872 byte: 46 = 208)

Word Fetch

object-pointer word: *fieldIndex* = *field contents*
(e.g., 18168 word: 0 = 5)

Pointer Store

object-pointer pointer: *fieldIndex* ← *field contents*
(e.g., 20654 pointer: 1 ← 15)

Allocation

allocating oop: *object-pointer*
(e.g., allocating oop: 20654)

Bytecodes (in first and second traces)

Bytecode <*bytecode-index*> *bytecode-description*
(e.g., Bytecode <16> Push Temporary Variable 0)

Message Transmission (in all traces)

[cycle=*bytecode cycle*] *receiver-description selector-string argument-descriptions*
The bytecode cycle is the number of bytecodes that have been executed. The receiver and argument descriptions will show the class of the appropriate object except in the case of SmallIntegers, Strings, true, false and nil which print more nicely.
(e.g., [cycle=408] aLargePositiveInteger digitAt: 3
[cycle=75] 40 digitMultiply:neg: 808 false)

Primitive Invocations (in all traces)

Primitive *#primitive-index*
(e.g., Primitive #70)

Returns (in all traces)

↑ (method / block) of *returned value description*
(e.g., ↑ (method) of aLargePositiveInteger
↑ (block) of 64)

Object Pointers Lists

As an aid to debugging, the eighth and ninth files are a list of the oops of all classes and methods in the system.

Objects Known to Interpreters

There is a set of objects that must be known by a Smalltalk-80 interpreter. The oops of these objects are usually used as constants to interpreters, but could be located in special tables. These special oops/objects are (those marked * are not necessarily needed by interpreters, but are included in this table as debugging aids):

2 - the object nil
4 - the object false
6 - the object true
010 08h - an Association whose value field is Processor
012 0Ah - *Symbol classVariable USTable, the table of Symbols
014 0Ch - class SmallInteger
016 0Eh - class String
020 10h - class Array
022 12h - *an Association whose value field is the SystemDictionary, Smalltalk
024 14h - class Float
026 16h - class MethodContext
030 18h - class BlockContext
032 1Ah - class Point
034 1Ch - class LargePositiveInteger
036 1Eh - *class DisplayBitmap
040 20h - class Message
042 22h - class CompiledMethod
044 24h - *#unusedOop18
046 26h - class Semaphore
050 28h - class Character
052 2Ah - symbol #doesNotUnderstand:
054 2Ch - symbol #cannotReturn:
056 2Eh - *symbol #monitor:
060 30h - SystemDictionary classVariable SpecialSelectors, the array of selectors for bytecodes 0260-0317
062 32h - Character classVariable CharacterTable, table of Characters
064 34h - symbol #mustBeBoolean