A STANDARD SOFTWARE PLATFORM FOR DIGITAL INTERACTIVE TELEVISION

Robert Thibadeau, Ph.D.
School of Computer Science
Carnegie Mellon University

James Large, B.S.E.E. and Joe Newcomer, Ph.D.
Television Computer, Inc.
Pittsburgh, Pennsylvania

Abstract

This paper describes a standard for the representation of portable application programs in the television set and set-top environment. The representation solves many practical problems in distributing digital interactive television services.

THE TELEVISION COMPUTER

The word "digital" in "digital interactive television" is a code word for "computer." Digital interactive television, in any conceivable meaning, implies "the television set contains a computer." The television computer makes it possible, in fact, outrageously important, to investigate how to apply the "hardware-software" distinction we exploit in computer science.

As we will see, the result of the investigation reveals straightforward, economically viable and stimulative, solutions to many problems in cable television and telecommunications in general, including problems of copyright ownership and interoperability among systems.

The purpose of this paper is to present the technology that enables a sharing of the "software" of digital interactive television by varieties of "hardware." This is to reduce the "box count" in the living room, while, perhaps, at the same time increasing the "invisible box" count in the house and office. This goal is kind not only to the consumer, but also to the box maker because of manufacturing volume requirements. A hardware manufacturer can make more money by contributing components of widely accepted boxes than from diverse, low volume, boxes. Software companies can make more money by contributing components as well.

We coined the term "television computer" [1][2][3] to refer to the computer in the television. This is distinct from the term "telecomputer" coined by Jim Clark [4] that extends past computer, television, and into telephony as well. There are many television computers. Most televisions sold today contain micro controllers. Every game machine and converter box contains micro controllers. To date these

controllers are "hard programmed" in the sense that they are re-programmable only at the factory. This situation will change.

Open programmability is on the horizon and already available in a number of boxes, such as the Philips CD-I television computer. However, there is no way to program, or write software, that can operate reliably across the different television computer platforms. Each platform must be individually coded. This is true whether we state that the meaning of "platform" is the "raw hardware" or, even, "the software operating system on the raw hardware." In other words, there is no way to write a piece of code that is the same across software operating systems. We believe there is a fairly simple, straightforward, and well understood way to free up this bottleneck without otherwise "giving up the store."

THE G-CODE SYSTEM

Our proposal is analogous to the NTSC standard for analog video data streams into (and out of) television sets. NTSC allows any supplier of television programs to send those video programs to any body's television set. Open programmability for digital interactive television then, quite naturally, means the transmission of computer software, digital programs, into television sets with the same guarantee of interoperability. If there is technology that can provide open programmability by controlling the form of the transmission of computer software into television sets, this technology should be examined. The technological know-how does exist. There are many computer scientists aware of this. To date the proposals have been for proprietary standards.

Our proposal is distinguished from others in that it proposes a free and public technology. Our technology does not replace, but properly augments, the many valuable contributions made by the authors of the proprietary standards, other software makers, telecommunications companies, and hardware manufacturers. Television Computer, Inc., copyrights the work, but this is in order to provide a mechanism for a single authoritative source for the software. The company is also committed to cooperating with the various industry standards groups and we have endeavored to place the contribution in the context of actual or emerging industry standards.

The clean separation of software and hardware in the television computer is achieved by providing a software layer that speaks a standard language. But what should this language be? Our view is that the language should not be the top level of the operating system, but should rather be a level that allows direct programming of the television computer. This programming level should not be a high level programming language, such as "C" or "FORTRAN," but a low level programming language similar to "assembly code." It has been recognized and well known for decades that all computers share basic sets of operations at the "assembly code" level. This shared set, at minimum, can provide a universal framework for minimal programming.

What is required of the operating system software is an "interpreter" that can read the universal assembly code

(commonly called "virtual machine code") and translate this into the precise hardware environment in which that operating system resides. A second, and last, minimal requirement is for a few well defined "system calls" for input and output from the computation that the television computer can be made to perform. Every operating system must support these few system calls that take the form of code call outs to the real operating system.

We have named this code "G-Code" where "G" stands for "Group." The ***group allocator*** mechanism described below provides a means by which different "networks or channels" can operate on one set-top without the possibility of interfering with one another. But, basically, G-Code was inspired by the "P-Code" or "Pseudo Code" of the old UCSD Pascal Compiler from the 1970s. In contrast to "P-Code" and others such as the GNU C intermediate, Microsoft C P-Code or even FORTH, G-Code was developed expressly for the purpose of creating a standard software platform for digital interactive television. This idea is not new to either science or practice. Another example is the IBM-Apple consortium's proprietary "Kaleida" operating system and high level language that uses proprietary (but probably similar) instruction codes to achieve interoperability between the MacIntosh and IBM PC platforms.

For the G-Code to be interesting, it must be simple and universally interpretable. The requirement of simplicity has to do with the manufacturing and materials cost of putting a G-Code interpreter into every television or set-top box. ***The incremental cost to support G-Code must approach insignificance.***

To this end, the present proposal is to specify G-Code as a possible "payload" within a SMPTE Header/Descriptor framework. MIT initiated efforts to define a Header/Decriptor framework for advanced television and digital video systems, and this prompted formation of a Society of Motion Pictures and Television Engineers (SMPTE) task force on Headers/Descriptors. The task force completed its work early in 1992 and a working group, whose work is nearing completion, was formed to take the task force report and produce a standard. The SMPTE Header/Descriptor provides for unique identification (using the ISO/ITU registration system) of payload encoding rules. Descriptors are defined to provide for payload parameterization and the insertion of application oriented information including copyright notification. Payloads can be digital video programming, such as MPEG compressed NTSC, or, as in our case, transportable computer codes. The header standard is itself embedded in an international standard for declarative syntax known as ITU/ISO ASN.1 [6]. Done in this way, if a television does not support any G-Code interpreter, it can simply reject payloads that have the G-Code header. This is done simply because a single internationally standard authorizing number does not match. If it does support G-Code because the header number matches a number in the television's capability list, it can accept the payload and interpret it. Thus, the incremental cost of G-Code, through a standard already worked out

internationally, does indeed approach zero dollar cost.

The next step up in cost is when the G-Code interpreter is present in the operating system of the television computer. We provide a G-Code interpreter for as many micro controllers as possible (without discrimination but dependent on resources). Furthermore, unlike the several propriety systems, G-Code is sufficiently simple and straightforward that a vendor can freely "roll his own version," if, for one reason or another, he does not desire to use ours. This would allow any operating system vendor to include a G-Code interpreter at minimal cost. The working G-Code Draft Document is available from Television Computer, Inc. [7].

At the core of the G-Code system is the G-Code Virtual Machine — an abstract computer that does not favor any one vendor's hardware over another's. Programs written for the G-Code virtual machine are executed in either of three ways:

- An emulator directly interprets the program's instructions and system calls.

- The program's instructions are translated to native code, and the native code is executed in a software environment that emulates the system calls.

  -or-

- The hardware directly implements the G-Code instruction set.

The G-Code virtual machine is designed to facilitate translation of G-Code programs to both CISC and RISC style native instruction sets. G-Code defines a rich set of instructions that allows a translator to make effective use of the rich instruction sets of typical CISC processors. For RISC type machines, it contains hints about flow of control and storage classes that simplify register and branch optimizations.

Some set-top and television manufacturers may wish to make only a minimal commitment to supporting the G-Code standard. For this reason, the G-Code virtual machine is defined as a ***core instruction and system call set*** and a set of ***standard virtual machine extensions.*** The virtual machine core specifies the minimum set of data types and operations needed to allow software to be loaded and run on any set-top or television. It provides for integer arithmetic, simplistic graphics, and the most essential system calls.

The standard virtual machine extensions are optional instructions and data types that, if either present or absent, can be handled by the program, or by the G-Code interpreter's ***group allocator.***

To take one example: A hypothetical "atlas of the planets" program might use floating point arithmetic. When run on set-top box A, that has a floating point co-processor, the G-Code floating point operations are translated to co-processor instructions, and executed directly. On another set-top box, B, there is no floating point hardware support, but the manufacturer has elected to provide

software emulation for the floating point G-Code extension in native code. Finally, on a third set-top box, C, that does not support the floating point G-Code extension, the app can still run if its group allocator provides a software emulator for the extension.

In the example above, if the app's group allocator provides floating point emulation in a discardable library, and the app is run on either set-top box A, or B, then the group allocator can discard the emulator library and reclaim the associated memory, knowing that the app-provided emulator will not be needed on those set-top boxes.

If a certain set-top box contains a unique "hardware accelerator", the manufacturer is encouraged to define a non-standard G-Code extension, thereby enabling application vendors to use the accelerator. If it is meaningful for them to do so, application vendors can define software extension emulators to mimic the custom hardware and thereby enable their applications to run on other vendor's set-top boxes. (Presumably, they will not run as fast without the custom hardware). In order to handle speed of operation variations, there is a standard extension G-Code system call provided for clocking. An application can use this when it requires certain real-time constraints to operate.

A "group" in our proposal refers to a single managed collection of software programs made to run on the television computer through initial "G-Code" booting. In practice, Groups are identified by the SMPTE Header under a branch of the ISO/ITU name space. For example, a header number {iso(1) organization(3) smpte(52) 68} could be the prefix used to identify a SMPTE defined class for downloadable television G-Code. In this hypothetical case, the ID and identifiers with this prefix (for example, {1 3 52 68 1}) uniquely identify G-Code programs in the class. A G-Code allocator, in keeping with the proposed standards, could be declared in the Descriptor portion (the parameterization) of the SMPTE Header which contains the G-Code payload.

The "Time Warner Full Service Network" might have (a version) that request to use {1 3 52 68 1} as its "G-Code publisher's" number. Since this number, as per the SMPTE standard, is unique, the group will have a unique Header ID and identifier in the operating system of the television computer. Because we cannot guarantee symmetric communications in and out of the television computer, it is impractical to have the television computer generate a supposedly unique number for a group.

However, if symmetric communications is possible in a particular plant, and in any manner that symmetric communications is possible, the G-Code Group allocator mechanism allows the definition of cooperative communication processes between any two or any one-to-many configuration of tasks running on different machines.

The most significant potential problem with something like a G-Code standard for digital interactive television is that the code may not be efficient on a particular piece of hardware or for a particular computer programming language. The microcomputer manufacturers extend the instruction sets of their computers

precisely in order to provide higher computing efficiencies. Furthermore, high level programming languages like "C," "Basic," and "Pascal," and scripting languages like "Lingua" and "ScriptX," also may not compile or interpret efficiently in G-Code. Indeed, the box makers and computer language people extend the system calls and sometimes the instruction sets in order to accomodate specialized hardware "accelerators." The most well known instance of this is the "sprite controller" in game boxes such as Atari, Sega, and Nintendo. We believe that our G-Code proposal effectively and correctly addresses both the hardware interoperatability and software interoperability problems. We also believe that it simultaneously addresses the scaleability problem that allows a box maker to control the cost of supporting G-Code.

Once there is a commitment to have the set-top or television set handle SMPTE Header/Descriptors, basic G-Code support is designed to have a real cost that is as little as a fraction of a dollar. At the other end, for the more aggressive among us, it facilitates efficient code deployment for high-end digital interactive television. This in no way decreases the value, or necessity, of native code in high, or low, end hardware.

Security

We understand that the very idea of allowing someone else to run a computer program in one's television computer can send shivers up the spine. What happens if the program grabs the television's display so the poor homeowner has to "power cycle" his television to get it back? What happens if the program has a "bug" and "crashes?" Curiously, the technology that can guarantee against these failures (except when there is an outright electronics failure that interacts with computer state conditions) has been well understood for many years. The basic idea is quite simple:

- Because the G-Codes are symbolic and actively interpreted by the computer,
- because they do not run directly on the hardware or in the operating system, and
- because the Group allocator mechanism allows program groups to be isolated from each other,

any G-Code task can, at worst, get itself messed up. This technology is rarely employed in PCs, but it has been common in workstations, and mainframes for decades.

We believe that it should be overtly, and standardly, employed in television, precisely to offer the interoperatibility that permits the classic interplay of independent hardware, transport, and content. A box maker, for example a Sega, might have a G-Code demonstration that runs, albeit slowly and only illustratively, on a Nintendo box, and, vice versa. Furthermore, with G-Code Groups, Sega can distribute highly optimized G-Code to varying models of Sega boxes.

In summary, the box and the operating system are secure. This security is absolute. It is impossible to create a virus using G-Codes except for a virus internal to one's own publishing group. There is no "password" mechanism outside of a group.

The word "impossible" is very strong. We should caution that, if a hardware or operating system, or software vendor, publishes a G-Code extension that opens the door on his hardware, operating system, or software, the door is opened to anyone using that extension. It is up to these people to protect their own security when they proactively generate and publish new extensions.

"First Copy" Protection

We will close this short paper with what we regard as perhaps one of the most interesting examples of the use of G-Code that we call, "first copy protection." This is the kind of protection offered by pay-per-view. However, it can be controlled on a finer grain, by the publisher. A publisher can change his encryption frequently. He uses a G-Code program to create a permission for the receipt of further programming. In receivers that can decrypt an entire program the permission can provide absolute first copy protection.

Of course, once a receiver has gained permission to play the program once, it would be possible for a hacker to gain an illegal copy in principle (even if hardware architecture may make that difficult). Nevertheless, this is copy protection that is under the control of the publisher. In effect, he downloads G-Code that hides a permission.

Interestingly, this does not disenfranchise the forms of copy protection that have been developed for pay-per-view. Technically, at least in the computer world, the method employed by box makers like General Instrument and Scientific Atlanta for "addressable converters" is known as "multicasting." First copy protection through G-Code downloading is enhanced by a multicasting architecture, because multicasting secures an *authorization,* as opposed to a *permission,* on a box-by-box and event-by-event basis. The Group Allocator mechanism provides the means of tying the permission with the authorization. A G-Code program cannot obtain *authorization* without a G-Code extension provided by the hardware manufacturer that gives the G-Code, or certain G-Code groups (e.g., the Turner or Viacom XYZ channel) such authorization.

The Information Highway

G-Code programs are designed to support the information highway, again through the adoption of international standards. Structured text is supported by the G-Code system. Structured text objects are represented by arbitrary length sequences of characters. The structuring is based on SGML or "Standard Generalized Markup Language" developed at the National Institutes of Standards and Technology and widely employed on the Internet. SGML is made to enable computer documents to be marked up for retrieval, search, and playback by specific "players" (such as MPEG players). For example, in one SGML application, World Wide Web (from CERN in Geneva, Switzerland), one can traverse the global community on Internet through hyperlinks. SGML is on the other end of the spectrum from the SMPTE Header/Descriptor standard. G-Code provides a natural method by

which publishers can control the region in between the two. An SGML interpreter is possible on anybody's box, even with only core G-Code support.

SUMMARY

The G-Code system is a software system, freely distributed, that adequately and correctly describes a standard for the representation of portable application programs in the television set and set-top environment. The G-Code system has the following technical attributes:

- MODULAR SYSTEM with STANDARD EXTENSIONS requires only a minimal commitment from a set-top box maker who wishes to comply with the standard.

- EFFICIENT, HIGH-PERFORMANCE implementations are possible using many different RISC and CISC type processors.

- LOADABLE EXTENSION EMULATORS allow any application software to be run on a minimal system in which the standard extensions are not built-in.

- DESCRIPTOR BASED ADDRESSING allows each application to run in a protected address space, even when there is no specific hardware support.

- MULTITASKING allows one vendor's information services to be "on line", even when another vendor's services are "in use".

- EXTENSIBLE RESOURCE MANAGEMENT allows different applications to share not only the standard resources such as memory, audio-video display, infrared remote control, PCMCIA cards, and printers, but also APPLICATIONS

- DEFINED HARDWARE AND SOFTWARE RESOURCES.

- STANDARD, COMPACT ENCODINGS for STRUCTURED TEXT and GRAPHICS.

REFERENCES

[1] Thibadeau, R. *The Television Computer.* Pittsburgh, Pennsylvania: Television Computer, Inc., 1992.

[2] Thibadeau, R. "The television as a robot servant." Technical Report CMU-RI-TR-93-22, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, August, 1993.

[3] Thibadeau, R. "The question of standards in digital interactive television." Technical Report CMU-RI-TR-93-23, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, November 1993.

[4] Clark, J. "A TeleComputer." Unpublished manuscript, Silicon Graphics, Inc., 1992.

[5] Gerovac, B. & Soloman, R. "Protect revenues, not bits: identify your intellectual property." Unpublished manuscript, Richard Soloman, MIT DOHRS Program at the Center for Technology, Policy and Industrial Development, Cambridge, Mass. Also see June 1992 issue of *SMPTE Journal.*

[6] ISO 8824/8825 also, Rose, M.T. *The Open Book: a practical perspective on OSI.* Prentice-Hall, 1990.

[7] "The G-Code System: First Draft for Comments. " Technical Report TCTR-7, Television Computer, Inc., Pittsburgh, PA 15203. Phone 412 488 3610 or Fax 412 488 3611, $75 First Class, $90 Overnight U.S./CAN., MC/VISA/AMEX.