The Problem of Embedded Software in UCITA and Drafts of Revised Article 2

Philip Koopman, Ph.D.
Associate Professor
Electrical & Computer Engineering Department
Carnegie Mellon University
and

Cem Kaner, J.D., Ph.D.
Professor
Department of Computer Sciences
Florida Institute of Technology

This is the first part of a two-part article. This one focuses primarily on UCITA; the next will focus more on Article 2. All references to UCITA are to the amended, commented draft dated July 28-August 4 2000, at www.law.upenn.edu/bll/ulc/ucita/ucita1200.htm.

1. Introduction

If one of the key purposes of commercial law is to facilitate commerce, we should listen carefully to the repeated pleas from business advocates (such as many Chambers of Commerce) to simplify the law and avoid needless regulation. Regulations should serve a societal purpose whose value outweighs the cost of compliance. The treatment of software embedded in goods in UCITA and proposed revisions to Article 2 does not serve this purpose.

UCITA and some recent drafts of Article 2 attempt to make distinctions between embedded and non-embedded software. This issue was discussed repeatedly in the Article 2B / UCITA drafting committee meetings (Kaner attended 15 of the 16 meetings). The drafters and most of the observers repeatedly agreed that embedded software, such as the software the controls the fuel injectors in your car, should be governed under Article 2 rather than UCITA. Recent meetings of the Article 2 drafting committee and the 1999 annual meeting of the American Law Institute considered proposals to take non-embedded software (such as financial application software on mainframes or office productivity software on desktop PC's) outside of the scope of Article 2.

We believe that these distinctions are fundamentally flawed. They are based on an outdated view of embedded software, from a time when computer processors and memory were expensive, energy consumption of computer components was high, and the computational power of inexpensive processors was very limited. To complicate the problem further, the proposals treat software that is embedded in a computer (or in a computer peripheral) as non-embedded. This exception (software embedded in a computer or peripheral) swallows the rule (embedded software should be treated differently).

It is increasingly difficult to distinguish between embedded and non-embedded software as embedded systems and desktop computing merge into a more integrated computing environment.

The simplest approach is to make the distinction a matter of contract. If the software and the associated goods are sold together, under the same contract, treat the software as embedded in the goods. Alternatively, if the software is sold or licensed under a separate contract, treat the software as being

separate and non-embedded. (We've been told repeatedly that this is essentially the distinction made by the United States military when it acquires hardware and software.)

We're not fond of this distinction because it says that a vendor can bring any software within or out of UCITA simply by choosing to use one contract or two. However, this rule has two important advantages:

(a) It is absolutely clear. Both parties to the contract and the court will be able to quickly decide whether software sold (or licensed) with goods is to be treated as part of the goods.

(b) The rule doesn't drive companies into making artificial engineering design tradeoffs in order to bring a product within (or out of) UCITA.

Based on those advantages, Kaner has repeatedly proposed, at meetings of the Article 2B/UCITA drafting committee, the Article 2 drafting committee, and at the 1999 ALI annual meeting, that this be the distinction made within UCITA and Article 2. The proposal was repeatedly rejected.

We are not advocating this simple distinction in this article. Instead, we are raising it here as a baseline for comparison.

Suppose that Criterion X is proposed as a distinction between embedded and non-embedded software, such that software is considered non-embedded (and is thus governed by UCITA) if it does meets Criterion X and is considered embedded (Article 2) if it fails Criterion X. One way to evaluate the merit of Criterion X is to ask how hard it would be for a designer of a product to make the product meet or fail Criterion X intentionally. For example, how hard would it be for a car manufacturer to make its fuel injector software satisfy Criterion X? If the manufacturer can pick its governing law by making relatively simple engineering choices, then the law is equivalent to the rule that the software is embedded if it is sold with the hardware, and non-embedded if it is the subject of a separate contract. For a manufacturer in this situation, Criterion X is merely a regulation, and the cost of bringing the software within UCITA is merely the cost of complying with the regulation. Such a regulation has social value only if the actions required to achieve compliance create some value, such as by making the product safer or more reliable, or by bringing income to the state.

1.1 Three Questions to Evaluate Criteria for Distinguishing Embedded from Non-Embedded Software

The drafting committees keep entertaining new proposals for criteria that purport to distinguish between embedded and non-embedded software. We'll consider some of them in this paper, but we can't anticipate all the new variations that will be proposed. We suggest that when you see a proposed criterion, you ask three questions:

(a) Does this distinction go to the heart of the difference between embedded and non-embedded software, or does it merely reflect a difference in how these types of software are (as far as you or the drafters know) commonly implemented today?

(b) What would it take for a manufacturer to redesign its product in a way that brings the embedded software under UCITA? Are there examples already on the market that most people would consider to be embedded products that would fall under UCITA without such modifications?

(c) Suppose that a manufacturer made the least-cost design changes that bring its embedded software under UCITA. What are the expected impacts of the changes? For example, do we expect the resulting product to be safer? Easier to set up and use? Less likely to need repairs?

We respectfully suggest that a distinction that doesn't go to the heart of the difference between embedded and non-embedded software, that can be worked around easily, and whose likely workarounds won't improve the product, is a bad distinction, in effect a regulation of the kind that is reviled by businesses because it imposes a cost on the business for no good purpose.

2. Why Does This Difference Matter?

An argument that surfaced repeatedly in the Article 2 meeting in St. Louis (November 18-19, 2000; Kaner attended the meeting) was that the distinction is not very important. It was suggested that the Article 2 rules are not so different from common law or UCITA rules that it would be worthwhile to intentionally bias the design of a product in order to bring it within one law instead of another.

However, at the same meeting it was repeatedly suggested that Article 2 might fail due to industry opposition if we did not exclude software from its scope. Evidently, the distinction between Article 2, UCITA, and common law rules is material to some influential people.

Let us suppose that a manufacturer is designing a product that includes software, and can bias the design in a way that brings the software under Article 2 or under UCITA. Here are just a few of the advantages that the manufacturer would gain by selecting UCITA. (Note: for detailed references, please see Cem Kaner, *Software Engineering & UCITA*, 18 J. Comp. & Info. Law 435 (Winter, 1999; <www.badsoftware.com/engr2000.htm>).

- For non-mass-market products, there is no perfect tender rule.
- The clickwrap/shrinkwrap contract formation model is explicitly adopted in UCITA but not in current Article 2 and not so completely embraced in recent drafts of revised Article 2.
- The implied warranty of merchantability in UCITA is more vendor-friendly than Article 2's.
- Under UCITA, a disclaimer of implied warranty can meet a requirement of conspicuousness even if it is unavailable for inspection by the customer prior to the sale.
- Under UCITA, it is easier to argue that some feature or aspect of performance that appeared in a product demonstration does not give rise to an express warranty than under Article 2.
- UCITA defines material breach differently from the common law (Restatement of Contracts, Second, Section 241), and the definition is more favorable to the vendor.
- UCITA allows the vendor to place significant use restrictions on the product, restricting how, when, where, how often, and for what purpose the customer can use the product. Such restrictions are unusual in a sale of goods and not explicitly authorized in Article 2.
- UCITA allows the vendor to place transfer restrictions on the product, blocking the customer from reselling the used product. Such restrictions on alienation are foreign to a sale of goods and have historically been unenforceable in mass-market sales of intellectual property-based products such as books and records. UCITA has been sharply criticized because it "circumvents copyright's first sale doctrine." Committee on Copyright and Literary Property of the Association of the Bar of the City of New York (co-sponsored by the Communications and Media Law Committee and the Entertainment Law Committee), Report on a proposal of the National Conference Of Commissioners on Uniform State Laws to adopt a proposed Uniform Computer Information Transactions Act (June 21, 1999 at 18) <www.2bguide.com/docs/Copy.Comm1.pdf>.

We think that some manufacturers will consider these differences significant enough to be willing to spend some design and implementation money in order to bring their products under UCITA.

Here's an example. The modern automobile often has over a million lines of embedded code. Software controls the feel and effectiveness of the brakes, the smoothness of the ride, the feel of the suspension, the feel of the steering, the fuel efficiency and responsiveness of the car, security features, many aspects of the maintainability of the car, and so on. These all play significant roles in buyers' decisions between car models, and so they are arguably material to the transaction.

Imagine the consequences of bringing a car's embedded software within UCITA. Apart from the liability issues, just think of the transfer issue. The car manufacturer would be able to say that when you buy your new car, you cannot transfer the car's software to another person without paying a substantial transfer fee to the manufacturer. You or your customer would either have pay the fee or buy (at substantial expense) replacement software for dozens of computers scattered throughout the car. This will drive up the price of used cars, making them less competitive with new cars.

Additionally, note that under UCITA section 104, if the embedded software is governed by UCITA and if access to that software is a material part of the transaction, then the vendor can also bring the rest of the transaction within UCITA. If the software in your car is material to the transaction, the manufacturer can bring the whole car sale within UCITA.

We believe that this capability alone (the ability to require customers to get the manufacturer's permission and pay a fee in order to resell a used product) would be a significant incentive to many manufacturers.

3. UCITA's Treatment of Embedded Software

UCITA purports to apply only to what might be called "general purpose" computers and their peripherals. To a casual computer user, this brings to mind visions of a Windows PC or a Macintosh, keyboards, mice, and printers. However, the actual wording in the UCITA documents is vague and inconsistent, and actually includes a wide variety of everyday items used by ordinary people, professionals, and businesses. The official comments do not remedy this situation, and in fact contain substantial technical inaccuracies and apparent contradictions on the point of embedded systems.

We cannot walk through a detailed analysis of the comments in the space available in this paper. The comments are not law, they were not reviewed in the drafting committee meetings, and so we believe that judges will rely much more heavily on the black letter of UCITA than on the comments. However, we invite the interested reader to review Philip Koopman's analysis of the comments in his paper, "Why UCITA Falls Short for Embedded Systems" available at <www.ices.cmu.edu/koopman/ucita/embedded_ucita.pdf>

3.1 The Exclusion of Embedded Software

UCITA defines its scope (and thus includes or excludes embedded software and associated goods) in Sections 103 and 104. Here are the most relevant parts:

> *"103(b)(1) If a transaction includes computer information and goods, this [Act] applies to the part of the transaction involving computer information, informational rights in it, and creation or modification of it. However, if a copy of a computer program is contained in and sold or leased as part of goods, this [Act] applies to the copy and the computer program only if:"*
>
> > *"103(b)(1)(A) the goods are a computer or computer peripheral; or"*
> >
> > *"103(b)(1)(B) giving the buyer or lessee of the goods access to or use of the program is ordinarily a material purpose of transactions in goods of the type sold or leased."*
>
> *"104 The parties may agree that this [Act], including contract-formation rules, governs the transaction, in whole or part, . . . if a material part of the subject matter to which the agreement applies is computer information or informational rights in it that are within the scope of this*

[Act], or is subject matter within this [Act] under Section 103(b) . . . However, any agreement to do so is subject to the following rules:"

> *"104(a)(4) A copy of a computer program contained in and sold or leased as part of goods and which is excluded from this [Act] by Section 103(b)(1) cannot provide the basis for an agreement under this section that this [Act] governs the transaction."*

Section 103(b)(1) appears to exclude embedded software, but the exception to the exclusion (software that is embedded in a computer or computer peripheral) swallows the rule.

3.2 Defining a "Computer"

UCITA Section 102(a)(9) defines "computer" as

> *"an electronic device that accepts information in digital or similar form and manipulates it for a result based on a sequence of instructions."*

This definition encompasses virtually all digital computers, embedded or otherwise.

The Institute for Electrical & Electronics Engineers Standard Glossary of Computer Hardware Terminology, IEEE 610.10-1994 defines computer as

> *"a device that consists of one or more processing units and peripheral units, that is controlled by internally stored programs, and that can perform substantial computations, including numerous arithmetic operations, or logic operations, without human intervention during a run. Note: may stand alone, or may consist of several interconnected units."*

This definition is generally comparable to the UCITA definition in scope. It is worth noting that IEEE 610.10-1994 sees fit to identify an "embedded computer" as a subset of the class of all computers:

> *"a computer system that is part of a larger system and which performs some of the requirements of that system; for example a computer system used in an aircraft or rapid transit system."*

Thus, using either the UCITA definition of "computer" or the IEEE definition of "computer," the concept of a computer means all computers, not just desktop computers.

Here are some examples of devices that are computers within UCITA's definition:

Fuel Injection Control for an Auto

Essentially all future automobile engine controllers will have reprogrammable memory (*e.g.*, flash memory) to reduce the potential cost of recalls in the event of a software defect being discovered, and it is common to have it even today. Currently flash memory is being used by cell phones, hard drives, network hubs, and most automotive engine controllers. (source: http://www.bizjournals.com/sanjose/stories/1996/09/23/story6.html, September 20, 1996 article on flash memory usage, and our discussions with automotive engineers.)

It is possible on newer vehicles to reprogram engine operation by updating an existing flash memory chip with a new software version, much as one updates the "BIOS" software in a desktop or laptop computer. On older cars the equivalent operation can be accomplished by replacing either the memory chip itself or the entire engine control computer (both hardware and software). The Crossfire upgrade kit is an example of a combined software/hardware upgrade to an old vehicle's fuel injection control:

> *"Our Turbo City Corvette '82 - '84 Crossfire Upgrade Kit gives you: a 90's computer to make the Crossfire Fuel Injection react more quickly and more accurately. The package includes the 90's Crossfire upgrade computer with Eprom, crimping pliers, connectors and instructions to change-out the '82-'84 ECM. This will improve low and midrange performance, and fuel and spark delivery. The ECM will have our custom stock Crossfire chip. Once your upgrade computer is installed, we*

can provide special programming for replacement chips that will compensate for almost any type of performance modification or added power accessory. Any number of custom performance or special requirement chips will be made for your request, for only $129.00 +S&H per chip." (Turbo City Performance Headquarters, *Hey There Corvette Crossfire Owners!* <www.turbocity.com/CorvetteCrossfileECMUpgrade.htm>)

This is being sold by a third party, not by the makers of the Corvette. Other discussions on the site make it clear that the software comes with a license. Note that this chip is an EPROM (erasable programmable read only memory). Further updates can be made by bringing the car to a service center to have the chip reprogrammed, or by simply purchasing a new EPROM and throwing the old one away as one would do with a disk or CD for an out-of-date piece of desktop software.

The Crossfire fuel injection software comes on a computer chip. This chip is not sold as part of any other goods--you use it by installing it in your car, but that's not how you buy it. You buy software that is contained in a good (the Crossfire replacement computer) in a manner no different than buying a video game cartridge or software on CD-ROM for a desktop computer. Note also that the primary purpose of the transaction is to give the buyer use of a computer program (the one that will change the performance characteristics of the car). Thus, the software also falls within UCITA under Section 103(b)(1)(B).

This software transaction is clearly within UCITA when you buy it from Turbo City.

1. What if you bought a used Corvette and this chip was already installed in the car when you bought the car? The software is still contained in the Crossfire computer. Shouldn't UCITA still cover the transaction?
2. What if you signed two contracts when you bought the used car? One contract covers the car in general, but not the Crossfire computer. The other is a transferred license to the Crossfire computer and its software. Now the software is contained in a Crossfire computer that has been sold separately to you. How could UCITA *not* cover this transaction?

What if General Motors wanted to bring its original equipment fuel injectors within UCITA? Let us consider this possibility in terms of the three questions of part 1.1 above:

(a) *(Does the distinction go to the heart of embedded software?)* The distinction between a fuel injector control chip supplied by General Motors as original equipment and an equivalent chip supplied by a third party cannot go to the heart of the difference between embedded and non-embedded software. Both chips are plugged into the same machine, and they do essentially the same things, in the same ways.

(b) *(What would it take to bring the product within UCITA?)* If we accept that the Turbo City chip is covered within UCITA, then GM could achieve the same result for its own chips by selling its fuel injector chips and software as options. For example, the Corvette dealer could offer customers three different choices: one would optimize the car for fuel efficiency, the other could optimize the car for responsive handling, and the third could be the Turbo City chip. On making the choice, the customer would sign a separate contract, write a separate check, and receive a separate license and warranty page that is specific to the chip.

(c) *(What is the benefit of changes made to bring the product within UCITA?)* The consumer benefits gained from GM's offering a selection of fuel injector chips might outweigh the additional costs in design, testing, manufacturing, inventory management, and support — or they might not.

Laser Printers

A laser printer is an electronic device. It accepts information in digital form. It stores the information in memory (often 2 megabytes or more). The printer processes the stored information in a sequence, interpreting it as a series of commands with associated data. The printer might interpret the stored information using Hewlett-Packard's PCL (printer control language) or Adobe's PostScript language or some other language. Whatever language is used, the information is interpreted as a sequence of instructions and the printer's execution of those instructions leads to printouts, displays or messages on the printer's control panel, or messages back to the application that requested printing services.

Therefore, a laser printer is a computer within UCITA's definition, section 102(a)(9).

The software that comes with the printer, for example in the printer firmware, is a classic example of embedded software. But UCITA will not treat it as embedded because it is embedded in a computer (or in a computer peripheral).

Home Medical Device: Blood Glucose Testing

Consider Johnson & Johnson's SureStep Blood Glucose Monitoring System. (For current information on this product line, go to www.lifescan.com.) Initially this device had very limited functionality. Prick your finger (to draw blood). Wipe the blood on a test strip. Insert the test strip into the LifeScan reader and the device it presents you with a reading of blood sugar level. More recent versions of the device come with diagnostics. More recent versions of the device store the last several readings. Now, you can download software, get a cable, and transfer data from the monitor to your personal computer. One version of the device, which sells for under $100, is available at many pharmacies and does not require a prescription, stores up to 150 tests and automatically creates 14-day and 30-day test averages. A newer model still, the Profile, can hold 250 tests, it records additional information about the user (such as insulin type and dosage), labels tests by events (such as exercise), thus creating fairly detailed database records that carry an activity code, a reading, a time stamp, a date stamp, and probably other information. This can all be downloaded to a computer.

This is an electronic device. Inputs to the device are from the buttons (which are probably encoded, like a keyboard, to send a digital signal back to the central processor) and from a scanner that analyzes the blood on a test strip. The scanner output is probably digitally encoded and passed to a central processing unit that further interprets, stores, and transfers the results. The handling of the button inputs, the display of information on the device's LCD screen, analysis of the blood and the transfer of results to another computer are all done in accordance with a stored series of instructions. Therefore, we believe, this device is a computer within the definition of UCITA 102(a)(9).

By the way, even if you don't think of this device as a computer, you should think of it as a peripheral because it collects data that it can transfer to a computer. Therefore, even though this seems to be an obvious example of an embedded computer running embedded software, the software will be governed by UCITA.

Another consideration for this device, if you don't think of it as a computer, is to ask what would be needed to make it a general enough purpose computer to fit within your interpretation of UCITA 102(a)(9) (or some other appropriate definition of a computer). For example, we could add other inputs. Maybe it would take your blood pressure (computerized blood pressure measurement tools are popular in the home medical market). Or it could test you for pregnancy. Check the potassium levels in your blood. Check your children's blood for evidence of drug abuse. Check your blood alcohol levels before you drive. Or it could connect to your family doctor through a web connection (embedded web controllers are available inexpensively, see below), uploading data and perhaps browsing your medical records for you. Perhaps you could use it to check the status of your prescriptions at your pharmacist's website. These "enhancements" are possible and at some point this device must fit any reasonable definition of computer. However, the functions all look like traditional embedded hardware/software functions to us, we certainly

wouldn't want such a device covered by UCITA, and in terms of societal value, the more functionality that you add to a device, the more risk of error. A manufacturer might inflate the feature count in order to bring the product under UCITA but thereby achieve a less safe or less reliable device as a result.

Other Devices that Accept Analog Inputs

The Motorola 68020 is the central processing chip that controlled the Macintosh II series computer, as well as the Sun Microsystems series 3 Workstation, both classic general purpose desktop machines. As faster central processors came to the market, the 68020 moved into laser printers, becoming a workhorse for running Adobe Postscript. Today versions of this chip are still used in many industrial applications, and in those cases the chip and its software look like an embedded computer and software.

The 68020 doesn't interface directly with the physical world, but an embedded version of that same processor called the 68332 does. The 68332 is a 68020 processor with additional features, including 16 digital Input/Output pins to interact with the external world. (The fact that the pins are digital does not prevent them from being used to accept analog inputs and produce analog outputs with appropriate support circuitry added external to the processor chip.) Both chips can in general execute the same software and use the same engineering development tool set. In the last decade millions of 68332 chips have been used as engine controllers in cars from General Motors and Ford, among other uses. The newest version of this processor family (now marketed as the Coldfire processor by Motorola) also includes an Ethernet network controller on the same chip as the processor, and emphasizes the fact that it can run the same general software as the 68020 as a major selling point.

The 68332 is clearly "an electronic device that accepts information in digital or similar form and manipulates it for a result based on a sequence of instructions." The recently announced MCF5272 Coldfire chip can do not only that, but also run the Linux operating system and communicate directly with the Internet. Surely, these chips are no less a computer simply because they can control a car engine or a cash register. On this argument, then, despite the definition of computer in UCITA section 102(a)(9), an electronic device that accepted information in analog form but manipulated it in digital form for a result based on a sequence of instructions should also be a computer, and it certainly would be a computer within the definition provided by the IEEE.

Electronic Games and Sewing Machines

A product like the Nintendo Game Boy lets you play computer games within the UCITA definition of computer. The Game Boy is an electronic device. It accepts inputs (such as movements of a joypad) that are converted to digital values, and responds to your inputs according to a stored program. Additionally, the Game Boy can run many different programs (computer game cartridges contain programs) and new games can be developed at any time that can run on the Game Boy, without requiring any modification of the Game Boy itself. Game Boys can even be expanded with computer peripherals, including a printer and a digital camera.

The Game Boy's versatility as a computer shows up in its use within Singer's Izek sewing system (www.singershop.com/whats_new.html, accessed 12/26/00). This system will automatically sew stitch patterns, buttonholes and lettering. The system "includes a sewing machine, Game Boy, connection wire and special cartridge that contains stitch patterns and designs." These programs are not computer games. The Game Boy is therefore capable of running applications of various types (games, sewing, and therefore probably many others).

One would normally consider the Game Boy to be an embedded computer in a situation like this. The programs to sew stitch patterns, buttonholes, and lettering, are focused on driving the output device (the sewing machine) in a limited number of ways. However, UCITA makes no provision for a device that is sometimes a general purpose computer and sometimes a narrow embedded device. Therefore, the sewing

machine software should be, under UCITA, treated as if it were non-embedded because it runs on a computer.

Products that Use Embedded Web Servers

A web server is normally a big machine (for descriptions of inexpensive web servers, check www.dell.com/us/en/bsd/products/line_servers.htm). Recently, companies have been offering embedded web servers--a full web server on a chip.

The world's (allegedly) smallest web server is featured at www-ccs.cs.umass.edu/~shri/iPic.html. This device is smaller than a matchbox and costs less than $1. The author suggests a wide range of applications. For example,

> *"The iPic chip can be embedded in every appliance … in the house. These devices and appliances can all then be controlled from your web-browser. . … Once this is all set up, you do not need the computer to control the appliances -- they can communicate with each other through the power wiring and co-ordinate each other's activities. For instance, your alarm clock might tip off the rest of the house that its alarm time-setting has changed."*

As another example,

> *"Many industrial computers and devices are equipped with their own remote management facilities. With technology like the iPic they can be connected to common network facilities, instead of using dedicated wires and a dedicated control terminal, for each device or equipment. All these devices, which may include HVAC equipment, climate control in offices and large buildings, lighting and power management, security surveillance and monitoring, process control equipment, and many others can now all be controlled and managed using a unified terminal and with simplified procedures. This can lead to lower costs, better management, and sometimes, even increased safety."*

Another small, inexpensive web server that claims to be the smallest is the SitePlayer SP1 at www.siteplayer.com/products.htm, available in quantity for $19.95 each. The SP1 is based on the venerable 8051embedded processor chip design, which has been shipped in more than a billion embedded systems to date. According to the SitePlayer manual:

> *The SitePlayer is "the first product in a family of embedded web servers designed to enable any microprocessor-based device to become web enabled easily and inexpensively. In approximately one square inch, SitePlayer includes a web server, 10baseT Ethernet controller, flash web page memory, graphical object processor, and a serial device interface. . . . Example applications include audio equipment, appliance thermostats, home automation, industrial control, process control, test equipment, medical equipment, automobiles, machine control, remote monitoring, and cellular phones. . . . SitePlayer is a web server coprocessor that handles web protocols and Ethernet packets . . . SitePlayer can also be used in some applications in a 'stand alone' mode where simple I/O can be performed."*

These are electronic devices. They receive digital input. They manipulate the input, following some sophisticated programs. They are, under UCITA's definition, computers. Software that runs on them is therefore not to be treated as if it were embedded software. But these are classic examples of embedded computers and embedded software applications.

Other Examples

- The Palm is an example of a personal digital assistant (PDA). This type of handheld device offers several simple applications, such as managing address books, calendars, and to-do lists. This is a computer, under UCITA's definition.

- Windows CE (consumer electronics) is an operating system that is used in hand-held PCs for general purpose software such as PowerPoint, but can also be used in embedded systems. The main website for MS embedded systems is www.microsoft.com/windows/embedded/default.asp. Their end user license agreement for embedded use of the software is at www.microsoft.com/windows/embedded/ce/licensing/sampleeula.asp. It contains all the usual disclaimers that we see for desktop software. (It should be noted that all the embedded operating system vendor we have contacted have similar licensing terms; this is not simply an issue with Microsoft's approach.) Microsoft lists companies that use Windows CE as embedded software at http://www.microsoft.com/windows/embedded/ce/guide/casestudies/default.asp. For example, at www.microsoft.com/windows/embedded/ce/guide/casestudies/idexx.asp, Microsoft describes a milk-quality testing device to ensure that only wholesome milk is sold. *Anything powerful enough to run Windows CE will easily meet the UCITA definition of a computer.*
- A digital telephone or a digital answering machine is probably a computer under UCITA.
- A digital camera is probably a computer. We recently saw an advertisement for a digital camcorder that can record digital video, take single snapshots, display the images you are about to tape or shoot on a color LCD, and print (on a printer that is built into the camera) color snapshots. It can also download data to a connected computer or store data on memory sticks.
- Kaner recently bought a Timex alarm clock as a gift. It plays (digitized) nature sounds and CDs. It has about 25 buttons. You can set several different alarms, for different days. This is electronic. It accepts what is probably digital input from its buttons. It saves settings in its memory, displays them on its LCD screen, and then acts on them by sounding an alarm or a piece of music. It appears to be a computer (as defined in UCITA). If you think it doesn't quite meet the criterion, ask yourself what would be the cheapest enhancement to the clock to make it meet the criterion. It would not take much. Then ask, what social benefit was created by that enhancement?

In conclusion, by saying that software that is bundled with a computer is not to be treated as embedded software is to say, because the definition of computer is so broad, that an extremely wide selection of embedded software will not be treated, in UCITA, as embedded. *We will review the Article 2 proposals in more detail in our next paper but we will note here that this same problem applies to them.*

Additionally, if the software that is embedded in the computer is legitimate UCITA subject matter (under Section 103(b)(1)), then under UCITA Section 104, the vendor can specify that UCITA will also cover the computer in which the software is embedded. Looking back at the list of examples, an enormous range of goods are being pulled out of Article 2 and put into UCITA.

3.3. Defining a "Computer Peripheral"

If the definition of "computer" was not broad enough, UCITA 103(b)(1)(A) also specifies that software is to be treated as non-embedded if it is sold with goods that are a computer peripheral. UCITA doesn't define "peripheral."

There are two possible ways to clarify the meaning, both of which are too imperfect to be practical: definition by listing items, and definition by connectivity.

An attempt to list computer peripherals has the advantage of superficial clarity. A nice tidy list such as: "printer, scanner, keyboard, hard disk drive, floppy disk drive, CD-ROM drive, DVD-ROM drive, mouse, or trackball" is reasonably specific. However, it suffers in that new peripherals are continually being introduced (e.g., joystick, parallel-port video camera, virtual reality glove). Furthermore, even a seemingly obvious list and complete list would be complicated by the issue of dual-purpose items such as hard disk drives that are used both in computers and embedded systems.

The only plausible default definition of “computer peripheral” other than a list of specific items is "anything that is attached to a computer". IEEE standard 610.10-1994 defines a peripheral as:

> *“...a device that operates in combination or conjunction with the computer but is not physically part of the computer and is not essential to the basic operation of the system; for example, printers, keyboards, graphic digital converters, disks, and tape drives.”*

This definition conveys the same general sense and has the virtue of being internationally standard technical terminology. But this approach is fraught with problems.

Let us take the example of an ordinary laser printer, which most people would agree is a computer peripheral when placed in a normal office setting. This printer could be connected via a cable directly to a computer, making it a classical peripheral device. Or it could be attached to a local area network. Does being attached via a network make a printer not a peripheral? Most would probably say not, since it does not change the inherent property of being a printer (especially models that are factory configured to work both via network or via dedicated peripheral cable right out of the box). But, does that make anything connected to a computer via a network or other indirect connection a peripheral? If so, then very soon a dizzying array of items could become computer peripherals merely by adding a network connection to an existing class of product that most clearly is not a computer peripheral, including:

- Internet-enabled household appliances (“smart” refrigerators, ovens, and so on, which are now coming on the market)
- Gas and electric meters incorporating modems to dial in meter readings (these are already widely installed)
- Digital television recording devices that use a modem or cable modem to download new programming information (these are already established in the marketplace)
- Sensors that use embedded web servers to display status information (or perhaps in that case a household thermostat would really a “computer” instead of a “peripheral” – it will be difficult to really know until each and every such type of device is dealt with by litigation).

If connectivity or the ability to send or receive information to a general purpose computer is used as the defining quality of being a computer peripheral, then a number of devices that are usually not considered to be peripherals would then be transformed, by fiat, into computer peripherals. These would include such software-bearing items as:

- Telephones (cordless, corded with electronics, cellular – essentially all phones sold today), which are used as computer input devices for phone menus, whether input is touch-toned or spoken. Similarly, voice mail systems are implemented with general purpose computers and use telephones as input/output devices.
- Hotel doorknobs, which use magnetic cards or other devices to send request for entry into a general purpose computer in the hotel office.
- Smoke detectors, fire alarms, and other such safety critical devices which in large buildings, which are often monitored by a general-purpose computer.
- Laptop computer batteries that have integrated monitoring chips to provide charge level information to the laptop computer. While part of a computer, they are hardly what is normally thought of as a “peripheral device.”

4. Conclusions

UCITA and Article 2 are attempting to distinguish between embedded and non-embedded software. Software that is embedded in goods will be treated as part of the goods (under Article 2) whereas software that is not embedded will be (it is proposed) taken out of the scope of Article 2 and (in states that have adopted UCITA) left within UCITA.

There are five fundamental problems with this distinction:

- In general, it is almost impossible to distinguish between embedded and non-embedded software. We can think of prototypic examples of embedded software that are not being treated that way under UCITA or draft Article 2.
- UCITA and Article 2 both treat software that is embedded in a computer (used by distinct embedded processors in its keyboard, disk drives, monitor, and so on) as if it were not embedded software. Worse, the definition of "computer" is so broad that a large portion of classically embedded software will be arguably non-embedded within UCITA.
- UCITA and some of the Article 2 proposals treat software that is embedded in a computer *peripheral* as if it were not embedded software. This pulls even more embedded software out of the scope of Article 2.
- Non-embedded and embedded software are converging. Complex collections of functionality, such as web servers, are now available as embedded software. Similarly, complex operating systems (Windows CE for example) and programming languages (Java, Postscript) are being embedded in devices. Distinguishing between embedded and non-embedded software as they become more similar will be increasingly difficult.
- Finally, because the distinctions are so fuzzy, it is often cheap and easy to adjust some nonessential aspect of the product in order to make a stronger argument that the product belongs under UCITA or outside of Article 2. Such changes will rarely, if ever, achieve any benefit other than the benefit to the vendor of having pulled the product out of Article 2.

In the next article in this pair, we will look at distinctions between that the Article 2 drafting committee has been trying to make. The same problems apply.