Gemalto S.A. v. HTC Corporation et al Doc. 242

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

GEMALTO S.A.,	§	
	§	
Plaintiff,	§	
	§	
vs.	§	**CASE NO. 6:10-CV-561 LED-JDL**
	§	
HTC CORPORATION, et al.,	§	
	§	
***Defendants*.**	§	
	§	

MEMORANDUM OPINION AND ORDER

This claim construction opinion construes the disputed claim terms in U.S. Patent Nos. 6,308,317 ("the '317 patent"); 7,117,485 ("the '485 patent"); and 7,818,727 ("the '727 patent")[1] (collectively, the "patents-in-suit"). The patents-in-suit all share the same title, "Using a High Level Programming Language with a Microcontroller." On May 3, 2012, the Court held a claim construction hearing to construe the disputed terms. For the reasons stated herein, the Court adopts the constructions set forth below.

OVERVIEW OF THE PATENTS

The patents-in-suit are generally directed towards methods of implementing a high level programming language such as Java on resource constrained devices such as smartcards. The patents-in-suit disclose a method of compiling Java source code such that the Java Card applet uses less byte code than a traditional Java applet. This process conserves memory, which is a necessity of operating in resource constrained devices like smartcards. The patents describe a Java application with Java source code files that are compiled to produce application class files.

[1] The '727 patent is a continuation of the '485 patent which is a continuation of the '317 patent, which claims priority to U.S. provisional application ser. No. 60/029,057.

Dockets.Justia.com

These application class files are then fed into a card class file converter, which consolidates and compresses the files to produce a single class file, which is then loaded to the smart card microcontroller. In other words, the patents-in-suit teach a method for converting Java byte codes into byte codes that minimize the computing resources consumed by the application and the Java Virtual Machine so that the application can run within the constrained environment of a smartcard. Claim 1 of the '317 patent is representative of the asserted claims:

> An integrated circuit card for use with a terminal, comprising:
> a communicator configured to communicate with the terminal;
> a memory storing:
> an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step comprising:
> recording all jumps and their destinations in the original byte codes;
> performing a conversion operation selected from the group: converting specific byte codes into equivalent generic byte codes; modifying byte code operands from references using identifying strings to references using unique identifiers; and
> renumbering byte codes in the compiled form to equivalent byte codes in an instruction set supported by an interpreter on the integrated circuit card; and
> relinking jumps for which the destination address is affected by the conversion operation; and
> an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and
> a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

CLAIM CONSTRUCTION PRINCIPLES

"It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (quoting *Innova/Pure Water, Inc. v. Safari Water Filtration Sys., Inc.*,

381 F.3d 1111, 1115 (Fed. Cir. 2004)). The Court examines a patent's intrinsic evidence to define the patented invention's scope. *Id.* at 1313-1314; *Bell Atl. Network Servs., Inc. v. Covad Commc'ns Group, Inc.*, 262 F.3d 1258, 1267 (Fed. Cir. 2001). Intrinsic evidence includes the claims, the rest of the specification and the prosecution history. *Phillips*, 415 F.3d at 1312-13; *Bell Atl. Network Servs.*, 262 F.3d at 1267. The Court gives claim terms their ordinary and customary meaning as understood by one of ordinary skill in the art at the time of the invention. *Phillips*, 415 F.3d at 1312-13; *Alloc, Inc. v. Int'l Trade Comm'n*, 342 F.3d 1361, 1368 (Fed. Cir. 2003).

Claim language guides the Court's construction of claim terms. *Phillips*, 415 F.3d at 1314. "[T]he context in which a term is used in the asserted claim can be highly instructive." *Id.* Other claims, asserted and unasserted, can provide additional instruction because "terms are normally used consistently throughout the patent." *Id.* Differences among claims, such as additional limitations in dependent claims, can provide further guidance. *Id.*

"[C]laims 'must be read in view of the specification, of which they are a part.'" *Id.* (quoting *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995)). "[T]he specification 'is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id.* (quoting *Vitronics Corp.v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)); *Teleflex. Inc. v. Ficosa N. Am. Corp.*, 299 F.3d 1313, 1325 (Fed. Cir. 2002). In the specification, a patentee may define his own terms, give a claim term a different meaning that it would otherwise possess, or disclaim or disavow some claim scope. *Phillips*, 415 F.3d at 1316. Although the Court generally presumes terms possess their ordinary meaning, this presumption can be overcome by statements of clear disclaimer. *See SciMed Life Sys., Inc. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d

1337, 1343-44 (Fed. Cir. 2001). This presumption does not arise when the patentee acts as his own lexicographer. *See Irdeto Access, Inc. v. EchoStar Satellite Corp.*, 383 F.3d 1295, 1301 (Fed. Cir. 2004).

The specification may also resolve ambiguous claim terms "where the ordinary and accustomed meaning of the words used in the claims lack sufficient clarity to permit the scope of the claim to be ascertained from the words alone." *Teleflex, Inc.*, 299 F.3d at 1325. For example, "[a] claim interpretation that excludes a preferred embodiment from the scope of the claim 'is rarely, if ever, correct." *Globetrotter Software, Inc. v. Elam Computer Group Inc.*, 362 F.3d 1367, 1381 (Fed. Cir. 2004) (quoting *Vitronics Corp.*, 90 F.3d at 1583). But, "[a]lthough the specification may aid the court in interpreting the meaning of disputed language in the claims, particular embodiments and examples appearing in the specification will not generally be read into the claims." *Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1571 (Fed. Cir. 1988); *see also Phillips*, 415 F.3d at 1323.

The prosecution history is another tool to supply the proper context for claim construction because a patentee may define a term during prosecution of the patent. *Home Diagnostics Inc. v. LifeScan, Inc.*, 381 F.3d 1352, 1356 (Fed. Cir. 2004) ("As in the case of the specification, a patent applicant may define a term in prosecuting a patent"). The well-established doctrine of prosecution disclaimer "preclud[es] patentees from recapturing through claim interpretation specific meanings disclaimed during prosecution." *Omega Eng'g Inc. v. Raytek Corp.*, 334 F.3d 1314, 1323 (Fed. Cir. 2003). The prosecution history must show that the patentee clearly and unambiguously disclaimed or disavowed the proposed interpretation during prosecution to obtain claim allowance. *Middleton Inc. v. 3M Co.*, 311 F.3d 1384, 1388 (Fed. Cir. 2002). "Indeed, by distinguishing the claimed invention over the prior art, an

applicant is indicating what the claims do not cover." *Spectrum Int'l v. Sterilite Corp.*, 164 F.3d 1372, 1378-79 (Fed. Cir. 1988) (quotation omitted). "As a basic principle of claim interpretation, prosecution disclaimer promotes the public notice function of the intrinsic evidence and protects the public's reliance on definitive statements made during prosecution." *Omega Eng'g, Inc.*, 334 F.3d at 1324.

Although, "less significant than the intrinsic record in determining the legally operative meaning of claim language," the Court may rely on extrinsic evidence to "shed useful light on the relevant art." *Phillips*, 415 F.3d at 1317 (quotation omitted). Technical dictionaries and treatises may help the Court understand the underlying technology and the manner in which one skilled in the art might use claim terms, but such sources may also provide overly broad definitions or may not be indicative of how terms are used in the patent. *Id.* at 1318. Similarly, expert testimony may aid the Court in determining the particular meaning of a term in the pertinent field, but "conclusory, unsupported assertions by experts as to the definition of a claim term are not useful." *Id.* Generally, extrinsic evidence is "less reliable than the patent and its prosecution history in determining how to read claim terms." *Id.*

The patent in suit may contain means-plus-function limitations that require construction. Where a claim limitation is expressed in means-plus-function language and does not recite definite structure in support of its function, the limitation is subject to 35 U.S.C. § 112 ¶ 6. *Braun Med., Inc. v. Abbott Labs.*, 124 F.3d 1419, 1424 (Fed. Cir. 1997). In relevant part, § 112 mandates that "such a claim limitation be construed to cover the corresponding structure . . . described in the specification and equivalents thereof." *Id.* (citing 35 U.S.C. § 112 ¶ 6.). Accordingly, when faced with means-plus-function limitations, courts "must turn to the written

description of the patent to find the structure that corresponds to the means recited in the [limitations]." *Id.*

Construing a means-plus-function limitation involves two inquiries. The first step requires "a determination of the function of the means-plus-function limitation." *Medtronic, Inc. v. Advanced Cardiovascular Sys., Inc.*, 248 F.3d 1303, 1311 (Fed. Cir. 2001). Once a court has determined the limitation's function, "the next step is to determine the corresponding structure disclosed in the specification and equivalents thereof." *Medtronic*, 248 F.3d at 1311. A structure is corresponding "only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim." *Id.* Moreover, the focus of the corresponding structure inquiry is not merely whether a structure is capable of performing the recited function, but rather whether the corresponding structure is "clearly linked or associated with the [recited] function." *Id.*

DISCUSSION

I. Hardware Terms

a. microcontroller[2]

Plaintiff's Proposed Construction	Defendants' Proposed Construction
a device designed specifically for embedded applications that includes a central processing unit, memory and other functional elements on a single semiconductor substrate, or integrated circuit	a single semiconductor substrate or integrated circuit designed specifically for embedded applications that includes a central processing unit, memory and other functional elements, and that does not require external memory to function properly; not a microprocessor

There are two key disputes with regards to the "microcontroller" term: (1) to what extent a microcontroller may access external memory and (2) whether the preamble of claim 1 of the '485 patent[3] is limiting.

[2]The term "microcontroller" is found in the '317 patent at claims 58, 59, 62, 63, 65-69, 73, 75, 77-79, 81, 87-89, 91, and 92; and the '485 patent at claims 1, 2, 6-11, 14, 16, 18, 20-22, and 25.

i. Court's Construction of Microcontroller

The parties agree that a microcontroller includes a central processing unit, memory, and other functional elements on a single semiconductor substrate or integrated circuit. *See, e.g.*, PLAINTIFF'S OPENING CLAIM CONSTRUCTION BRIEF (Doc. No. 163) ("PLAINTIFF'S BRIEF ") at 13; DEFENDANTS' RESPONSIVE CLAIM CONSTRUCTION BRIEF (Doc. No. 180) ("RESPONSE") at 6. At the Claim Construction Hearing, the Court proposed that "microcontroller" should be construed as "a single semiconductor substrate integrating electronic circuit components that includes a central processing unit and all program memory making it suitable for use as an embedded system." Although Defendants originally proposed a construction that included language defining "microcontroller" as "not a microprocessor," Defendants agreed at the Claim Construction Hearing that the Court's proposed construction was proper. MARKMAN TRANSCRIPT at 22:11-13.

Plaintiff contends that the specification defines "microcontroller" as "including a central processing unit, memory, and other functional elements, all on a single semiconductor substrate or integrated circuit (e.g., a 'chip')." '317 patent at 2:2-5. Further, Plaintiff argues that the "specification affirmatively states a microcontroller *requires* external components: '[d]ue to the small number of *external components required* and their small size, microcontrollers . . . .'" PLAINTIFF'S BRIEF at 14 (quoting '317 patent at 2:35-36) (alterations as in PLAINTIFF'S BRIEF). Thus, for Plaintiff, a microcontroller need not house "all program memory" because a microcontroller is required to have external components.

[3] In its briefing, Plaintiff argues that the preamble of Claim 7 of the '485 patent should not be limiting. PLAINTIFF'S BRIEF at 13 n.6. However, at the Claim Construction Hearing, the parties agreed the preamble of claim 7 of the '485 patent is not limiting but disputed whether the preamble of claim 1 of the '485 patent is limiting. *See* CLAIM CONSTRUCTION HEARING TRANSCRIPT (Doc. No. 214) ("MARKMAN TRANSCRIPT") at 76.

The Court finds that while microcontrollers may access external components as argued by Plaintiff, all program memory necessary for execution of the compressed application must be located on the microcontroller. This construction finds support both in the specification and the extrinsic evidence cited by both parties. Looking first to the specification, the patents-in-suit teach that unlike microprocessors, microcontrollers have limited access to memory:

> Microcontrollers differ from microprocessors in many ways. For example, a microprocessor typically has a central processing unit that requires certain external components (e.g., memory, input controls and output controls) to function properly. A typical microprocessor can access from a megabyte to a gigabyte of memory, and is capable of processing 16, 32, or 64 bits of information or more with a single instruction. In contrast to the microprocessor, a microcontroller includes a central processing unit, memory and other functional elements, all on a single semiconductor substrate, or integrated circuit (e.g., a "chip"). As compared to the relatively large external memory accessed by the microprocessor, the typical microcontroller accesses a much smaller memory. A typical microcontroller can access one to sixty-four kilobytes of built-in memory, with sixteen kilobytes being common.

'317 patent at 1:62 – 2:10 (emphasis added); *see also id.* at 2:26-34 (noting that because of the physical characteristics of a microcontroller being located on a single "chip," it has considerably less memory than a microprocessor); *id.* at 2:14-16 ("In a microcontroller, the amount of each kind of memory available is constrained by the amount of space on the integrated circuit used for each kind of memory").

Additionally, the specification teaches that prior to the '317 patent, Java platforms generally operated on "microprocessor-based computers, with access to relatively large amounts of memory and hard disc storage space." 317 patent at 1:55-57. The patentee addressed the problem of fitting the operating system, application code, and a Java Virtual Machine to interpret the Java byte codes into machine code on the microcontroller. PLAINTIFF'S BRIEF at 3. Indeed, the CEO of Sun Microsystems, the creator of the Java programing language, characterized the problem as "fitting Java technology inside smartcards was like playing golf in a phone booth."

EX. B TO PLAINTIFF'S BRIEF, JAVACARD TECHNOLOGY GROWS UP SMART, (DOC. NO. 163-3); PLAINTIFF'S BRIEF at 3. Thus, an interpretation of microcontroller that permits access to off chip memory would eviscerate the distinction between operating high level programming language on an integrated circuit card or microcontroller and the "conventional platforms" that existed at the time of the invention.

This understanding is further supported by arguments made in the prosecution history. The applicants repeatedly distinguished between existing microprocessor systems that had the ability to run high level languages at the time of the invention and microcontrollers such as smartcards or integrated circuit cards. For example, in their appeal brief to the Board of Patent Appeals and Interferences, the applicants conceded that the steps to compile and run Java were well known in the prior art. *See* Ex. E to RESPONSE (Doc. No. 180-5) ("'485 PROSECUTION HISTORY") at GEM3763. However, "[a]t the time of the invention, the typical Java Virtual Machine required over 1MB of memory." *Id.* at GEM3766. In contrast, a typical microcontroller at the time of filing only had approximately 0.1 to 2.0 KB of RAM, 2 to 8KB of EEPROM, and 8 to 56K ROM. '317 patent at 2:26-34. Moreover, the patentee highlighted this distinction in arguments made to the Patent Office during the Reexamination of the '317 patent:

> …[U]sing Java as an example, providing Java technology onto smart cards would be very challenging due to the size constrains of smart cards as constrained to the minimum requirements of Java. The Java run time environment, the JRE, requires a system that has a minimum 32 MB of memory, 125 of free disk space. However, in 1996 smart cards only had 512 bytes, not kilobytes or megabytes, just 512 bytes of RAM and 4K bytes of EEPPROM.

EX. G. to RESPONSE (Doc. No. 180-7) ("REEXAM EXERPTS") at GEM4515. Thus, for the patentee, one of the key aspects of the invention disclosed in the patents-in-suit is its ability to operate in an environment with a finite amount of memory.

Lastly, interpreting "microcontroller" to require all program memory to be located on the chip is supported by the extrinsic evidence. Plaintiff's own evidence, The Microcontroller Idea Book, explains that a microcontroller is limited by the amount of memory it can hold:

> To make a complete computer, a microprocessor requires memory for storing data and programs, and input/output (I/P) interfaces for connecting external devices like keyboards and displays.
>
> In contrast, a microcontroller is a single-chip computer because it contains memory and I/O interfaces in addition to the CPU. Because the amount of memory and interfaces that can fit on a single chip is limited, microcontrollers tend to be used in smaller systems that require little more than the microcontroller and a few support components.

JAN AXELSON, THE MICROCONTROLLER IDEA BOOK 2 (1997) (Doc. No. 163-13) ("MICROCONTROLLER IDEA BOOK").

Plaintiff argues that the specification and extrinsic evidence show that "microcontrollers may use external memory." *See* PLAINTIFF'S BRIEF at 14 (quoting '317 patent at 2:35-36) ("In fact the specification affirmatively states a microcontroller *requires* external components: '[d]ue to the small number of *external components required* and their small size, microcontrollers …"); PLAINTIFF'S REPLY TO DEFENDANTS' RESPONSIVE CLAIM CONSTRUCTION BRIEF (Doc. No. 189) ("REPLY") at 3. The Court's construction does not, however, prevent a microcontroller from accessing any external memory. Under the Court's construction, a microcontroller need only possess sufficient memory to run the Java code[4] in accordance with the patentee's invention. For example, a microcontroller may access off chip memory to store and retrieve data stored in a "static RAM." *See* REPLY at 3 (citing Ex. BB, THE MICROCONTROLLER IDEA BOOK, at 24).[5]

[4] The Court uses "Java code" throughout the opinion only as an example of a high level programming language implemented on the microcontroller and does not intend to limit the claims to only Java embodiments.

[5] Plaintiff also relies heavily on various unrelated patents that describe "microcontroller." *See* PLAINTIFF'S BRIEF at 14 n.8. In construing "microcontroller," the Court has considered these references, but nevertheless finds that a person having ordinary skill in the art would interpret "microcontroller" as written in the '317 patent and its progeny as described above.

Furthermore, Plaintiff's arguments that the specification qualifies its description of microcontrollers as only "typical" microcontrollers and therefore should not limit microcontroller is similarly unavailing. As explained above, neither the specification nor the extrinsic record supports the conclusion that a microcontroller can function with program memory located off chip.

In conclusion, the Court construes "microcontroller" as "a single semiconductor substrate integrating electronic circuit components that includes a central processing unit and all program memory making it suitable for use as an embedded system."

ii. The Preamble of Claim 1 of the '485 Patent is Limiting

Defendants argue that the preamble of claim 1 of the '485 patent is limiting. *See* RESPONSE at 10; MARKMAN TRANSCRIPT at 76:16-22. Generally, a preamble is considered limiting when it "recites essential structure or steps, or if it is 'necessary to give life, meaning and vitality to claims or counts.'" *Catalina Marketing Int'l. v. Coolsavings.com, Inc.*, 289 F.3d 801, 808 (Fed. Cir. 2002) (quoting *Pitney Bowes, Inc. v. Hewlett-Packard Co.*, 182 F.3d 1298, 1309 (Fed. Cir. 1999)); *Kropa v. Robie*, 187 F.2d 150, 1952 (C.C.P.A. 1951). Moreover, "clear reliance on the preamble during prosecution to distinguish the claimed invention from the prior art transforms the preamble into a claim limitation because such reliance indicates use of the preamble to define, in part, the claimed invention." *Catalina Marketing Int'l*, 182 F.3d at 808. However, "a preamble is not limiting 'where a patentee defines a structurally complete invention in the claim body and uses the preamble only to state a purpose or intended use for the invention.'" *Id.* (quoting *Rowe v. Dror*, 112 F.3d 473, 478 (Fed. Cir. 1997)).

As explained above in the context of the Court's construction of "microcontroller," the patentee repeatedly emphasizes in both the specification and the prosecution history, that a key

aspect of the invention was the ability to fit a "high level programming language" such as Java on resource constrained devices such as a "microcontroller." *See, e.g.,* '485 PROSECUTION HISTORY at GEM3725 ("Applicants recognized the difficulty in operating Java (or other high level language) programs within the limited resources of an integrated circuit card or other microcontroller"). Thus, the Court finds that the term "microcontroller" as used in the preamble of claim 1 of the '485 patent breathes life and meaning into the claim and is therefore limiting.

b. integrated circuit card[6]

Plaintiff's Proposed Construction	Defendants' Proposed Construction
an integrated circuit containing a central processing unit, memory and other functional elements on a base	a card containing a central processing unit, memory and other functional elements, all on a single semiconductor substrate or integrated circuit, that does not require external memory to function properly; not a microprocessor system; OR a card containing a microcontroller

The essential dispute between the parties is whether the "integrated circuit card" must contain all program memory and whether the integrated circuit card is a "card" or can also be a "base." At the Claim Construction Hearing, the Court proposed "a card containing a single semiconductor substrate having a central processing unit and all program memory." Defendants agreed with the Court's construction; however, Plaintiff objected based on the Court's inclusion of the "all program memory" limitation and the Court's limiting of "integrated circuit card" to a "card."

First, Plaintiff's arguments regarding "all program memory" are essentially the same as they were in the context of "microcontroller." The Court rejects Plaintiff's arguments for the reasons discussed above. Further, the specification discloses an "integrated circuit card" that "includes a communicator and a memory that stores an interpreter and first instructions of a first

[6] The term "integrated circuit card" is found in the '317 patent at claims 1, 4-11, 13-15, 22, 24, 25, 30, 31, 55, 64, 84-86, 93, & 94; and the '485 patent at claims 38, 39, 40, 42, and 43.

application." '317 patent at 5:31-33. The integrated circuit card also "includes a processor that is coupled to the memory and is configured to use the interpreter to execute the first instructions to communicate with the terminal via the communicator." *Id.* at 5:36-40. Thus, the integrated circuit card contains the memory necessary to execute the condensed high level program.

Second, Plaintiff seeks to read the word "card" out of the claims by interpreting "integrated circuit card" as "an integrated circuit . . . on a base." The plain language of the term "integrated circuit card" requires a "card." This is supported by the specification which repeatedly refers to an integrated circuit card as a "card." *See, e.g.*, '317 patent at 18:62 – 19:4 ("Fig. 21 shows an integrated circuit card, or smart card, which includes a microcontroller 210 that is mounted to a plastic card 212. The plastic card 212 has approximately the same form factor as a typical credit card"); *id.* at 2:35-41 (noting that "microcontrollers frequently are used in integrated circuit cards, such as smart cards" and that these include both "contact-based" and "contactless cards"); *id.* at 2:55-60 (integrated circuit cards include "debit cards"); *id.* at 7:57-59 ("In some embodiments, the microcontroller, memory and communicator are embedded in a plastic card that has substantially the same dimensions as a typical credit card").

Plaintiff argues that an integrated circuit card need not be a card at all, but may be a "base." Plaintiff relies on the following portion of the specification to support its conclusion that an integrated circuit card may actually be a "base" rather than a "card:"

> In some embodiments, the microcontroller, memory and communicator are embedded in a plastic card that has substantially the same dimensions as a typical credit card. In other embodiments, the microcontroller, memory and communicator are mounted within bases other than a plastic card, such as jewelry (e.g., watches, rings or bracelets), automotive equipment, telecommunication equipment (e.g., subscriber identity module (SIM) cards), security devices (e.g., cryptographic modules) and appliances.

'317 patent at 7:57-65. Specifically, Plaintiff argues that "[t]he elements of an integrated circuit card do not have to be mounted on a plastic card or in the case of a microcontroller a single

semiconductor substrate: '[i]n other embodiments, … *mounted within bases* other than a plastic card.'" PLAINTIFF'S BRIEF at 16 (quoting '317 patent at 7:59-62) (alterations within quotation in original). However, as Defendants point out, when read in context with the surrounding language, this portion of the specification merely shows that microcontrollers can be used in embodiments not "embedded in a plastic card." Thus, while a microcontroller may be mounted on other surfaces, this portion of the specification does little to persuade the Court that an "integrated circuit card" need not be a "card." Rather, an "integrated circuit card" must have its components on a single semiconductor substrate. *See* '317 patent at 2:35-37 ("Due to the small number of external components required and their small size, microcontrollers frequently are used in integrated circuit cards, such as smart cards").

Lastly, Plaintiff argued at the Claim Construction Hearing that an "integrated circuit card" is intended to be broader than simply a "microcontroller." In support, Plaintiff points to portions of the specification that generally describe an "integrated circuit card" as including a "processor that is coupled to the memory" and including "other functional elements." *See, e.g.*, '317 patent at 5:36-40. In other words, Plaintiff argues that an "integrated circuit card" is merely a base with a processor on it. However, Plaintiff concedes in its opening claim construction brief that the "Integrated Circuit Card Claims . . . narrowly refer to a specific type of device that includes a microcontroller and can use high level programming language – *i.e.*, an integrated circuit card, as shown in the embodiment in Figure 21." PLAINTIFF'S BRIEF at 27-28 (citing '317 patent at 18:65 – 19:4); *see also* '317 patent at Fig. 21 (depicting an integrated circuit card as a "card"); PLAINTIFF'S BRIEF at 28 ("The prosecution history shows that the '317 patent was always prosecuted as including two classes of claims: a broad set directed to microcontrollers on devices generally, and a narrower set directed to microcontrollers on integrated circuit cards

specifically"). Accordingly, the Court rejects Plaintiff's invitation to construe "integrated circuit card" broadly and construes "integrated circuit card" as "a card containing a single semiconductor substrate having a central processing unit and all program memory."

c. programmable device[7]

Plaintiff's Proposed Construction	Defendants' Proposed Construction
This term does not need construction. To the extent that the term is construed, it should be given its plain and ordinary meaning, such as: a device that can execute a computer program	integrated circuit card or other device controlled by a microcontroller; not a microprocessor-based system

At the Claim Construction Hearing, the Court suggested that "programmable device" should be construed the same way as "microcontroller," i.e., a programmable device is "a single semiconductor substrate integrating electronic circuit components that includes a central processing unit and all program memory making it suitable for use as an embedded system." Defendants agreed with the Court's proposed construction; however, Plaintiff argued that a "programmable device" is "a device that can execute a computer program." For the reasons stated below, the Court adopts its proposed construction.[8]

Plaintiff's chief argument is that while the specification may describe microcontrollers rather than broader "programmable devices," the patentee was permitted to claim the genus of "programmable devices" of which "microcontroller" is a species. PLAINTIFF'S BRIEF at 16-17. Thus, for Plaintiff, "[t]he invention disclosed relates to conversion of byte codes and the invention is applicable to other programmable devices, not just the preferred embodiment" such as microcontrollers. PLAINTIFF'S BRIEF at 17. To support this argument, Plaintiff relies on Federal Circuit case law interpreting 28 U.S.C. § 112, the written description requirement.

[7] The term "programmable device" is found in the '727 patent at claims 1-7, 10, 12, 14, and 16-18.

[8] The parties also dispute whether the preamble of claim 3 of the '727 patent is limiting. For the same reasons discussed in the context of "microcontroller," the Court finds the preamble of the '727 patent limiting.

While Plaintiff may be correct that "disclosure of a species may be sufficient written description support for a later claimed genus including that species," that argument presupposes that a person having ordinary skill in the art would understand "programmable device" in light of the specification as broader than "microcontroller." PLAINTIFF'S BRIEF at 16-17; REPLY at 5.

Because "programmable device" is not a term of art and has no plain and ordinary meaning, it is particularly important to construe the term in the context of the intrinsic record. *See Network Commerce, Inc. v. Microsoft Corp.*, 422 F.3d 1353, 1360-61 (Fed. Cir. 2005). While in the abstract, a "programmable device" may simply be a device that can execute a computer program as argued by Plaintiff, the specification does not support such a broad disclosure. Although the specification does not use the term "programmable device," the patents-in-suit contemplate a microcontroller that has the ability to be "programmed" using a high level language:

> In other embodiments, the above-described techniques are used with a microcontroller (such as the processor 12) [sic] may control devices (e.g., part of an automobile engine) other than an integrated circuit card. In these applications, the microcontroller provides a small platform (i.e., a central processing unit, a memory, both of which are located on a semiconductor substrate) for storing and executing high level programming languages. Most existing devices and new designs that utilize a microcontroller could use this invention to provide the ability to program the microcontroller using a high level language, and application of this invention to such devices is specifically included.

'317 patent at 18:31-42 (emphasis added). Furthermore, as explained above in the context of "microcontrollers" and "integrated circuit cards," the patentee repeatedly distinguished the invention from high level programming language operating on traditional microprocessors. Thus, "programmed device" must be narrower than simply a "device that can execute a computer program." Because "programmable device" is not a term of art, the only embodiments disclosed in the patents-in-suit use microcontrollers, and the patentee repeatedly distinguished the

invention from generic microprocessors, the Court finds that a "programmable device" is a "microcontroller." Accordingly, the Court construes "programmable device as "a single semiconductor substrate integrating electronic circuit components that includes a central processing unit and all program memory making it suitable for use as an embedded system."

d. "resource constraints"[9]

Plaintiff's Proposed Constructions	Defendants' Proposed construction
"computing resources that are limited when compared to conventional computing platforms, such as microprocessor-based desktop and personal computers" Alternatively, "limits on computing resources imposed by the physical characteristics of the device" Alternatively, "insufficient computing resources to run the application in the compiled form more efficiently than in a converted form" Alternatively, "a fixed amount of memory and/or processing resources on the device" Alternatively, "computing resources that are limited as compared to typical desktop computers."	Indefinite

Over the course of the claim construction process, Plaintiff has proposed no less than five different constructions for the term "resource constraints." Defendants, on the other hand, have consistently argued that the phrase is indefinite. For the reasons set forth below, the Court rejects Plaintiff's proposals and construes the phrase "a set of resource constraints" as "insufficient memory to run the compiled application source program in an unconverted form."

i. Plaintiff's Proposals

[9] The term "resource constraints" is found in the '317 patent at claims 65, 78, 91 & 92; the '485 patent at claims 7 & 21; and the '727 patent at claims 3 & 17.

In its initial claim construction briefing, Plaintiff argued that "resource constraints" should be construed as "computing resources that are limited when compared to conventional computing platforms, such as microprocessor-based desktop computers." *See* PLAINTIFF'S BRIEF at 17. In response to Plaintiff's initial construction, Defendants countered that "[o]ne of ordinary skill would not know what 'limited computing resources' means, which resources should be compared to those of 'conventional' computing platforms, or how many resources would avoid the 'resource constraints' limitations." RESPONSE at 16-17. Defendants also criticized Plaintiff's proposal on the grounds that the scope changes over time, i.e., it is "based on how one of ordinary skill in the art would understand memory constraints by comparison to 'conventional computing platforms' *today*." *Id.*

In response, Plaintiff stated "[u]pon review of Defendants' briefing, Gemalto better understood why Defendants asserted confusion as to the original construction – making assessments at different times and the use of labels (e.g., 'personal computer') can lead to complexities." PLAINTIFF'S OPPOSITION TO MSJ OF INVALIDITY FOR INDEFINITENESS (Doc. No. 191) ("RESPONSE TO MSJ") at 2. Accordingly, Plaintiff proposed an alternative construction for resource constraints: "limits on computing resources imposed by the physical characteristics of the device." *Id.* at 3. Again, Defendants argued that Plaintiff's proposal was ambiguous. Specifically, Defendants argued that one of ordinary skill in the art would not know "why 'physical characteristics' such as size should be considered to evaluate 'limits on computing resources,' or even which 'device' matters for purposes of 'resource constraints.'" DEFENDANTS' REPLY IN SUPPORT OF MSJ OF INVALIDITY FOR INDEFINITENESS (Doc. No. 201) ("MSJ REPLY") at 2. Defendants also argued that the specification focuses on resource constrained "integrated circuits," not generic "devices" as Plaintiff asserts. *Id.* at 2. Moreover,

because the Plaintiff advanced an alternative claim construction based on previously undisclosed expert testimony, the Court permitted Defendants to depose Plaintiff's expert and submit a supplemental brief based on the expert's deposition. *See* APRIL 30, 2012 ORDER (Doc. No. 199); DEFENDANTS' SUPPLEMENTAL BRIEF IN SUPPORT OF MSJ (Doc. No. 234) ("SUPP. MSJ BRIEF"); PLAINTIFF GEMALTO S.A.'S RESPONSE TO DEFENDANTS' SUPP. BRIEF (Doc. No. 237) ("RESPONSE TO SUPP. MSJ BRIEF").

After the Claim Construction Hearing, and at the behest of the Court, Plaintiff filed a supplemental brief putting forth an additional alternative construction, arguing that "resource constraints" should be construed as "a fixed amount of memory and/or processing resources on the device." *See* PLAINTIFF'S SUPPLEMENTAL CLAIM CONSTRUCTION BRIEF (Doc. No. 248) ("SUPP. CC BRIEF") at 2. Plaintiff included yet another proposal, partly returning to its original construction, arguing that "resource constraints" should be construed as "computing resources that are limited as compared to typical desktop computers." *Id.* Despite its earlier proposals, Plaintiff requests the Court to adopt one of these two constructions. RESPONSE TO SUPP. MSJ BRIEF at 2. However, both proposals are flawed because they fail to adequately define the metes and bounds of the term.

Plaintiff's first final alternative construction, "a fixed amount of memory and/or processing resources on the device" is more ambiguous than the term it seeks to define. Plaintiff argues that unlike microprocessor based platforms such as desktop computers, microcontroller based platforms are not "expandable," meaning that additional memory or a faster processor may not be easily added. SUPP. CC BRIEF at 1. However, Plaintiff's proposal provides no explanation as to what it means for "memory and/or processing resource" to be "fixed." The ambiguity is highlighted by a series of hypothetical question posed by Defendants:

> What does it mean for 'memory and/or processing resources to be fixed? Is it memory or speed when shipped? Does it mean memory or speed cannot be improved? If a user can increase memory on a smartphone with a microSD card, is the memory not 'fixed'? Also, what is the 'device' to look to? Is it the microcontroller itself or what device the microcontroller may be mounted in?

RESPONSE TO SUPP. CC BRIEF at 3 n.3. Moreover, this proposal improperly focuses on the physical characteristics of an unidentified device at the exclusion of the application/execution environment. As will be explained in more detail below in the context of the Court's construction, the patentee described the "constraints" of various platforms in relation to executing high level programming language such as Java.

Plaintiff's second final alternative construction, "computing resources limited as compared to typical desktop computers," is flawed by Plaintiff's own admission. *See* RESPONSE TO MSJ at 2 (noting that "making assessments at different times and the use of labels (e.g., 'personal computer') can lead to complexities" and confusion); *see also* MARKMAN TRANSCRIPT at 60:3-9 ("In our initial construction, we said, limits on computing resources as compared to a desktop computer or a personal computer. And the – and the defendants pointed out, and I think rightfully so, that while that is typically the case you're putting labels on something. What is a desktop computer, what is a personal computer as your comparison point"). Like Plaintiff's initial proposed construction, which Plaintiff concedes utilizes ambiguous comparison points, Plaintiff's second final alternative construction relies on the "label" of "typical desktop computers."

Furthermore, as Defendants' point out, the point of comparison is ambiguous because "at any given time, desktop computers, cell phones, smart phones, laptops, and tablets of many different vintages are in use." DEFENDANTS' RESPONSE TO SUPP. CC BRIEF (Doc. No. 223) ("RESPONSE TO SUPP. CC BRIEF") at 2. Thus, if a competitor wished to determine whether a

smartphone was "resource constrained," it would have no guidance as to what year or model "desktop computer" it should compare to. *See id.* Further, Plaintiff provides no guidance as to how "limited" the resources must be in comparison or to which computing resources it should be applied. Accordingly, the Court rejects Plaintiff's second final alternative construction.

ii. The Court's Construction

Instead of adopting any of Plaintiff's proposed constructions, the Court construes "a set of resource constraints" as "insufficient amount of memory to run the compiled application source program in an unconverted form." Looking first to the specification, the patents-in-suit describe the invention in terms of the execution environment for a Java application called a Java Virtual Machine ("JVM") that interprets byte codes of a compiled Java application loaded onto a platform. *See* '317 patent at 1:15-35. The specification explains that "[c]onventional platforms that support Java are typically microprocessor-based computers, with access to relatively large amounts of memory and hard disk storage space." *Id.* at 1: 55-57. These "microprocessor-based implementations frequently are used in desktop in personal computers." *Id.* at 1:58-59. The patentee explained that, in contrast to traditional microprocessor based platforms that "can access from a megabyte to a gigabyte of memory," microcontroller and integrated circuit platforms can only access one to sixty-four kilobytes. *Id.* at 1:66 – 2:10.

The specification further explains, "[i]n order for a Java application to run on a specific platform, a Java virtual machine implementation must be written that will run within the constraints of the platform, and a mechanism must be provided for loading the desired Java application on the platform, again keeping within the constraints of the platform." *Id.* at 1:49-54. Further, the specification describes the "constraints of the platform" exclusively in terms of reduced memory capacity. *See* '317 patent at 1:55 – 2:10 (describing various types of memory

capacity in microcontroller and microprocessor based platforms). Specifically, the specification describes the memory capacity of microprocessor based platforms and microcontroller based platforms in the context of their capacity for the Java application and the JVM. *Id; see also supra* pp 7-10 (discussing memory capacity of microcontrollers). Thus, the Java application and the JVM for interpreting the compiled byte code form of the Java application must be written such that they can fit into the available memory of the platform. However, due to the constraints of microcontroller and integrated circuit platforms, Java could not fit on the available memory. In order to fit a high level programming language onto platforms like microcontrollers and integrated circuit cards, the patents-in-suit teach compiling a high level program language before converting the compiled programming language into a minimized, i.e., converted, form. Thus, the application must be converted in order to fit within the "resource constraints" of the platform. It therefore follows that "a set of resource constraints" means "insufficient amount of memory to run the compiled application source program in an unconverted form."

Additionally, the Court's construction is supported by the claim language itself. For example, claim 65 of the '317 patent recites:

> A microcontroller having a set of resource constraints and comprising:
>
> a memory, and
>
> an interpreter loaded in memory and operable within the set of resource constraints, the microcontroller having: at least one application loaded in the memory to be interpreted by the interpreter, wherein the at least one application is generated by a programmable environment comprising:
>
> a) a compiler for compiling application source programs written in high level language source code form into a compiled form, and
>
> b) a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter, wherein the converter comprises means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by:
>
> b.l) recording all jumps and their destinations in the original byte codes;

> b.2) performing a conversion operation selected from the group:
>
> > b.2.1) converting specific byte codes into equivalent generic byte codes
> >
> > b.2.2) modifying byte code operands from references using identifying strings to references using unique identifiers; and
> >
> > b.2.3.) renumbering byte codes in the compiled form to equivalent byte codes in an instruction set supported by an interpreter on the integrated circuit card; and
>
> b.3) relinking jumps for which the destination address is affected by the conversion operation.

'317 patent at Claim 65 (showing claim as modified by reexamination) (emphasis added). The claim lays out an application and an interpreter operating within a set of resource constraints on a microcontroller platform. Particularly, step b identifies a "a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter." This language suggests that a compiled form that is not converted into a minimized (or converted) form, is not suitable for interpretation within the set of resource constraints by the interpreter.[10] In other words, the unconverted or unminimized form of the code requires more memory than is available on the microcontrollers and integrated circuits described in the patents-in-suit. Accordingly, the Court finds that a "set of resource constraints" means "insufficient amount of memory to run the compiled application source program in a unconverted form."

Defendants argue that "to the extent there could be a definite construction of 'resource constraints,' it would be in reference to a device that does not have the resources to run programs written in a high level language, such as Java, in 1997." SUPP. RESPONSE at 3. In support,

[10] All of the asserted claims that include the term "resource constraints" also include "converting" steps. The claims identify the result of the "converting" step as either a "minimized form" or a "second intermediate code." *Compare* '317 patent at claim 65; '485 patent at claim 7; and '727 patent at claim 3 *with* '317 patent at claim 78, 88, 91 & 92; '485 patent at claim 21; and '727 patent at claim 17. Nevertheless, the "minimized form" and the "second intermediate code" are both the result of the "converting" step, *i.e.* are "converted forms."

Defendants rely on passages in the specification and the prosecution history purportedly limiting the invention to the Java embodiments. *Id.* (citing, e.g., REEXAM EXCERPTS at GEM4515-16 ("using Java as an example, providing Java technology onto smart cards would be very challenging due to the size constraints of smart cards as contrasted to the minimum requirements of Java.")). While Defendants are correct that the patents-in-suit often describe the problem the inventors were attempting to solve in the context of Java or other high level programming languages, it does not follow that "resource constraints" must be defined in reference to Java or high level programming language. As Defendants themselves point out, the question is what resources are required to run a particular program in an unconverted form. *See id.* at 3 (quoting '317 patent at 1:55:-59). Moreover, as explained above in the context of "high level programming language," the claims themselves dictate the execution environment of the claimed program applications. Thus, the Court declines to define "resource constraints" with reference to Java or a high level programming language.

Moreover, Defendants argue that the Court's construction does not provide definite "metes and bounds" for the term because it could result in a device having a set of resource constraints in reference to one program while not having a set of resource constraints in reference to a second program SUPP. RESPONSE at 2. However, as noted above, the point of comparison is not between two unrelated programs, but between a "converted" application and the "unconverted" form of that same application. Further, the definiteness standard is one of reasonableness under the circumstances and a construction need only be as precise as the subject matter allows. *See, e.g., Orthokinetics, Inc. v. Safety Travel Chairs, Inc.*, 806 F.2d 1565, 1576 (Fed. Cir. 1986); *Shatterproof Glass Corp.*, 758 F.2d at 624. Moreover, defining "resource constraints" as "insufficient memory to run the compiled application source program in an

unconverted form" provides a "workable objective standard" from which a person having ordinary skill in the art can determine the scope of the claim. *Cf. Datamize, LLC v. Plumtree Software, LLC,* 417 F.3d 1342, 1351 (Fed. Cir. 2005) (finding the term "aesthetically pleasing" indefinite because it failed to provide a "workable objective standard").

Lastly, Plaintiff suggests that the Court's construction is flawed because it focuses on a platform's mere capability to run a program at the exclusion of platforms that may be able to run the program, just not "suitab[ly]." *See* SUPP. BRIEF at 2. However, Plaintiff fails to provide any support from the disclosures for a maximizing efficiency of programs on microcontrollers or integrate circuits. Instead, as explained above, the patentee focused on microcontrollers' and integrated circuits' inability to run Java based programming.

II. Software Related Terms

a. "high level language" / "high level programming language"[11]

Plaintiff's Proposed Construction	Defendants' Proposed Construction
programing language that must be compiled and interpreted before it can be executed by a computer	programming language that must be compiled or interpreted before it can be executed by a computer

The dispute between the parties focuses on the term in relation to the program language's execution environment. Defendants argue that the plain and ordinary meaning of "high level programming language" permits the programming language to be compiled **or** interpreted before it can be executed by a computer. RESPONSE at 22. Plaintiff appears to concede that, in the abstract, a "high level programming language" can be compiled **or** interpreted before execution; however, Plaintiff argues that in the context of the patents-in-suit, "a high level programming language" must be compiled **and** interpreted. *See* PLAINTIFF'S BRIEF at 4; RESPONSE at 22.

[11] The term "high level language" or "high level programming language" are found in the '317 patent at claims 1, 5, 31, 35, 64, 65, 84-86, 93, & 94; the '485 patent at claims 7, 38, 40, & 42; and the '727 patent at claim 3.

The parties' dispute over the proper execution environment is resolved by the claim language itself. As Plaintiff points out, every claim of the patents-in-suit is directed towards a programming language that is first compiled into a compiled form and then converted into an interpreted form. *See* REPLY at 6. For example, claim 1 of the '317 patent requires

> An integrated circuit card… comprising…
>
> a memory storing:
>
> > an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form...
> >
> > an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

'317 patent at claim 1 (emphases added). Thus, the claim language itself requires that a high level programming language is both compiled and interpreted before execution. This is further supported by the specification, which describes high level languages that must be both compiled and interpreted. *See* '317 patent at 18:47-64 ("Regardless of the source of the class files, the above description applies to languages other than Java to generate codes to be interpreted") (emphasis added). Thus, the Court finds no construction necessary because the claims fully identify the execution environment of the high level language program.

b. "class file format"[12] and "application having a class file format"[13]

Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction

[12] The term "class file format" is found in the '317 patent at claims 58, 63, 67, & 87; the '485 patent at claims 1, 6, & 9; and the '727 patent at claims 1, 2, & 5.

[13] The term "application having a class file format" is found in the '317 patent at claims 58, 63, &87; the '485 patent at claims 1, 6, & 9; and the '727 patent at claim 1.

class file format	a file format containing byte codes that are instructions for a virtual machine	a file format containing byte codes that are instructions for a hypothetical computer or a virtual machine
application having a class file format	a program whose compiled form is in a class file format	No construction necessary.

The parties agree that "class file format" relates to a "file format containing byte codes;" however, the parties dispute whether these files contain instructions for a "virtual machine" or a "hypothetical computer." Defendants characterize the dispute as whether byte codes "must be executed on a 'virtual machine' (Plaintiff) or whether they can be executed on real computers as well (Defendants)." RESPONSE at 26. Defendants argue that "class file format" should be construed to permit execution on a "hypothetical computer" because, as a technical matter, byte codes may be directly executed by real machines operating Java Optimized processors. The Court, however, finds it unnecessary to include either "virtual machine" or "hypothetical computer" in its construction of "class file format" because each of the claims clearly specifies the execution environment, i.e., that the compiled byte code instructions must be interpreted before execution by the processor. *See, e.g.,* '317 patent at claim 1 ("an interpreter operatable to interpret"). Thus, the Court construes "class file format" as "a file format containing byte codes."

Plaintiff asks the Court to construe "application having a class file format" as "a program whose compiled form is a class file format." Having construed "class file format", the Court finds further construction unnecessary. Plaintiff's proposal rewrites the claims which plainly describe an "application having class file format" as an application's form before compiling, i.e. when it is written in a high level programming language. For example, claim 58 of the '317 patent recites "a derivative application derived from an application having a class file format wherein the application is derived from an application having a class file format by first

compiling the application having a class file format into a compiled form." Similarly, claim 2 of the '317 patent states that "the high level programming language format comprises a class file format." While Plaintiff may be correct that the specification teaches that "an application [having a class file format] is a program written in a high level programming language that – when compiled – is compiled into 'class files containing byte codes,'" that is not what the claims recite. PLAINTIFF'S BRIEF at 9. Thus, the Court declines Plaintiff's invitation to rewrite the claims and therefore finds no construction necessary.

c. "compiling" and "compiled form"[14]

Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction
compiling	transforming a program written in a high level programming language into class files containing byte codes	transforming a program written in a high level programming language from source code to object code or byte code
compiled form	class files containing byte codes	No construction necessary

The dispute between the parties is whether the term "compiling" should be limited to "transforming program source code to byte code" or should also permit transforming program source code to "object code." Plaintiff does not dispute that the plain and ordinary meaning of "compiling" includes transforming program code into either object code or byte code; however, Plaintiff argues that "compiling," as disclosed in the specification and claimed in the patents-in-suit, is limited to byte code.

Defendants argue that limiting the result of "compiling" to "class file" embodiments by excluding "object code" from the construction of "compiling" is flawed. Defendants primarily argue that this (1) imports limitations from the specification into the claims and (2) is directly at odds with the claims because "some independent claims that require 'compiling' explicitly

[14] The terms "compiling" or "compiled form" are found in the '317 patent at claims 1, 31, 64-68, 73, 84-88, & 91-94; the '485 patent at claims 1, 7-10, 14, 16, 21, 24, & 38-43; and the '727 patent at claims 1, 3-6, 10, 12, 17, & 20.

require the inclusion of a 'class file format,' but others do not." RESPONSE at 24. Plaintiff argues that "compilers that compile an application programming language into object code are transforming the program into codes that are directly executable by a specific computing platform and thus are not interpretable by a virtual machine" as required by the claims. PLAINTIFF'S BRIEF at 5-6.

Defendants further argue that the patentee "confirmed" that "'compiling' is a term of art of Computer Science meaning 'To transform a program written in a high level programming language from source code to object code'" while prosecuting the '485 patent. RESPONSE at 24 (quoting Ex. E. to RESPONSE (Doc. No. 180-5)); *see Home Diagnostics, Inc.*, 381 F.3d at 1356 (noting that a patentee may define a term during prosecution). The Court is presented with the unique situation where Defendants are attempting to hold the patentee to a broad definition of a claim term in prosecution, rather than attempting to show "disclaimer." In this case, the applicants defined "compiling" broadly while distinguishing the claimed invention from a piece of prior art that did not disclose any form of compiling. Thus, the distinguishing feature of the claimed invention was not that it specifically "compiled" source code to object code, but that it compiled at all. To the extent that the prosecution history lends credence to Defendants' arguments, it is outweighed by specific language in the claims and the specification describing a system for "compiling" program source code into only byte code, not object code. *See Phillips*, 415 F.3d at 1314 ("To begin with, the context in which a term is used in the asserted in the claim can be highly instructive").

The claim language itself describes the compiling operation as one that produces an intermediate representation derived from the source code that is thereafter interpreted for execution. *See, e.g.*, '317 patent at claim 1 (" . . . a memory storing: an application derived

from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converted the compiled form into a converted form …). Thus, the claim language itself limits compiling to intermediate code. The claim language does not, however, describe the compiling operation as one that produces an object code that can be directly executed. This understanding is reinforced by the specification, which only discusses compiling in the context of byte codes. *See, e.g.,* '317 patent at 1:24-34 ("When a Java application is written, it is compiled into "Class" files containing byte codes that are instructions for a hypothetical computer called a Java Virtual Machine . . . . The Java virtual machine for the selected platform is run, and interprets the byte codes in the class file, thus effectively running the Java application") (emphasis added); *id.* at 18:47-64.

Lastly, Defendants' arguments that including "class file format" in the construction of "compiling" creates redundancies in the claims is well taken but can be remedied by simply not including the phrase "class files." Thus, the Court finds that the term "compiled," when read in the context of the claims, the specification, and the prosecution history means "transforming program source code to byte code." Construing "compiling" as such captures the meaning of "compiling" as disclosed in the patents-in-suit without the risk of adding redundancy by including "class files" in the construction. Having construed "compiling," the Court sees no further need to construe "compiled form."

d. "converting" and "converted form"[15]

Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction
converting	post processing the byte codes of the compiled form into a converted form	No construction necessary

[15] The terms "converting" or "converted form" are found in the '317 patent at claims 1, 31, 58, 64, 78, 84-88, & 91-94; the '485 patent at claims 1, 21, 38, & 42; and the '727 patent at claims 1, 10, 17, & 20.

converted form	Byte codes interpretable by a different virtual machine than the compiled form	No construction necessary

Plaintiff argues that both "converting" and "converted form" need to be construed to prevent Defendants from "improperly expand[ing] the scope of these claims terms . . . or otherwise confus[ing] the jury in support of their validity challenge." PLAINTIFF'S BRIEF at 9-10. Because "the asserted claims make clear that program written in a high level programming language is first compiled into byte codes and then converted to converted form," Plaintiff argues that "converting" should be interpreted as "post processing the byte codes from the compiled form into a converted form." *Id.* at 10. In contrast, Defendants argue that "converting" has no special meaning and is readily understandable by a jury in the context of the claims themselves. The Court agrees with Defendants.

In light of the Court's construction of "compiling" as "transforming program source code to byte code," Plaintiff's construction merely restates what the claim clearly sets forth. For example, inclusion of "postprocessing" is unnecessary because the claims specify that converting takes place after compiling. '317 patent at claim 1 ("… wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form …."); PLAINTIFF'S BRIEF at 9-10. Thus, Plaintiff is correct that "when read in the context of all the claims, it is clear that the 'converting' must occur after the generation of byte codes (compilation) and is therefore a post-processing step of the byte codes form the compiled form into the converted form." REPLY at 9. However, because the meaning of "converted" is clear from the claim language, the Court finds that no construction is necessary.

Plaintiff seeks to introduce three limitations to "converted form:" (1) "byte codes," (2) "virtual machines," and (3) the notion that the compiled form be interpretable by a different

virtual machine. First, limiting "converted form" to "byte codes" or "virtual machines" runs afoul of the doctrine of claim differentiation. While some claims require that the converted form be in "byte code" format, other claims do not. *Compare* '485 patent at claim 10 *with id.* at claim 14 (("The microcontroller of claim 10 wherein the compiled form is in a byte code format . . ."). Furthermore, some claims require that the converted form be interpreted by a "virtual machine" whereas others require the converted form to be interpreted by an "interpreter." *Compare* '727 patent at claim 17 *with id.* at claim 3. Thus, the Court declines to introduce "byte codes" or "virtual machine" into the construction of "converted form."

Plaintiff's construction also introduces a limitation that contrasts byte codes of the compiled form and byte codes of the converted form by requiring them to be interpreted by different virtual machines. As an initial matter, the Court is not persuaded that this contrast provides any relevant or meaningful guidance, particularly in light of the Court's construction of "compiling." Furthermore, as Defendants' point out, "only six[] of the twenty-four asserted independent claims suggest that the compiled form need be 'interpretable' at all, let alone the compiled form be interpreted by a different virtual machine than the converted form." RESPONSE at 29. Thus, the Court declines to introduce the additional limitations suggested by Defendants. Accordingly, the Court finds that "converted form" requires no construction.

e. "byte codes"[16] and "virtual machines"[17]

Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction
byte codes/byte code format	program instructions interpretable by a virtual machine	No construction necessary
virtual machine/	programs used to interpret byte	No construction necessary

[16] The terms "byte codes" or "bye code format" are found in the '317 patent at claims 1, 31, 58, 64, 65, 78, 84-88, & 91-94; the '485 patent at claims 14, 24, & 38-43; and the '727 patent at claims 10, & 20.

[17] The terms "virtual machine" or "intermediate code virtual machine" are found in the '317 patent at claims 1, 31, 58, 64, 65, 78, 84-88, & 91-94; the '485 patent at claim 21; and the '727 patent at claim 17.

intermediate virtual machine	codes	

Plaintiff argues that "byte codes" must be construed to assist the trier of fact and avoid confusion. Plaintiff draws support for its construction from specific embodiments disclosed in the specification that describe "byte codes" as "instructions for a hypothetical computer called a Java Virtual Machine." PLAINTIFF'S BRIEF at 7-8 (quoting '317 patent at 1:26-27). Plaintiff further argues that because the invention is not limited to a *Java* virtual machine, "byte codes" should only be limited to "program instruction interpreted by a virtual machine." *Id.* at 8.

Defendants argue that that Plaintiff's construction draws an improper distinction between codes executed on a virtual machine and codes executed on a real machine. For Defendants, the term "byte codes" is readily understandable to a lay jury, i.e. "byte codes" are "codes" in "byte" format that correspond to an instruction for a computer, either real or virtual. RESPONSE at 25. While the Court agrees with Defendants that Plaintiff's construction is overly limiting, the Court is of the opinion that a jury would benefit from a construction of "byte codes." Plaintiff's attempt to insert the type of platform configuration the byte codes are interpreted for execution in is an extraneous limitation beyond the mere meaning of the terms themselves. Thus, whether the byte code instructions are executed by a real or virtual machine is not a property of the codes themselves. Accordingly, the Court construes "byte codes" as "program instructions that are interpreted before execution." Similarly, the Court construes "virtual machine" as "program used to interpret byte code."[18] *See* '317 patent at 1:33-35 ("The Java virtual machine . . . interprets the byte codes in the class file, thus effectively running the Java Applications").

f. "specific byte code" and "generic byte code"[19]

[18] Defendants do not meaningfully contest Plaintiff's proposed construction for "virtual machine."

[19] The term "generic byte codes" or "specific byte codes" is found in the '317 patent at claims 1, 31, 58, 64, 65, 78, 84-87, 91, & 94; the '484 patent at claims 14, 24, 39, 41, 43; and the '727 patent at claims 10 & 20.

Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction
specific byte code	a byte code with built-in argument or operand	smaller byte codes
generic byte code	a byte code with a separate, accompanying argument or operand	larger byte codes for the same operation

The parties' dispute appears to be limited to whether Plaintiff's use of "argument or operand" is likely to confuse a lay jury. *See* RESPONSE at 29. Defendants argue that their constructions "captures the essence of this element" without introducing a potentially confusing concept. *Id;* REPLY at 10. However, Defendants do not argue that Plaintiff's proposed constructions are incorrect. The Court finds that Plaintiff's proposed constructions present a more accurate reading of the terms to one of ordinary skill in the art and would be understandable to a lay jury. Accordingly the Court construes "specific byte code" as "a byte code with a built-in argument or operand" and "generic byte code" as "a byte code with a separate, accompanying argument or operand." *See* '317 patent at 10:40-44 (noting that specific byte code such as "ILOAD_0" is replaced with an "argument 0").

g. terminal[20]

Plaintiff's Proposed Construction	Defendants' Proposed Construction
a device that communicates with the integrated circuit card or microcontroller	System, external to the microcontroller or ICC system, that communicates with the microcontroller or ICC system

The essential dispute between the parties is whether the "terminal" must be a system separate from the microcontroller or integrated circuit card. *See* PLAINTIFF'S BRIEF at 30; RESPONSE at 20-22. For the reasons set forth below, the Court finds that a "terminal" must be separate from the microcontroller or the circuit card and therefore construes "terminal" as "a

[20] The term "terminal" is found in the '317 patent at claims 1, 10, 25, 30, 31, 40, 55, 59, 62, 64, 84-86, 93, & 94; and the '485 patent at claims 2, 38, 40, & 42.

system, external to the integrated circuit card or microcontroller, that communicates with the ICC or microcontroller."

This construction is supported by the specification which consistently recites a terminal that communicates with the integrated circuit card, which suggests that the terminal is separate from the integrated circuit card. '317 patent at 4:59-64. Furthermore, the specification lists examples of terminals that are all external to the system that contains the integrated circuit card or microcontroller: "Terminals can be automated teller machines (ATMS), point-of-sale terminals, door security systems, toll payment systems, access control systems, or any other system that communicates with an integrated circuit card or microcontroller." *Id.* at 8:15-19. Plaintiff relies on this same portion of the '317 specification for the argument that "terminals can be . . . any other system that communicates with an integrated circuit card or microcontroller." PLAINTIFF'S BRIEF at 30; RESPONSE at 21. However, as Defendants' highlight, this language supports the notion that a terminal is a separate system by emphasizing that terminals can be any other systems, i.e. not the same system. Accordingly, the Court construes terminal as "a system, external to the integrated circuit card or microcontroller, that communicates with the integrated circuit card or microcontroller."

III. Means Plus Function Claims

a. attributes[21]

Plaintiff's Proposed Construction	Defendants' Proposed Construction
This term does not need construction. To the extent that the term is construed, it should be given its plain and ordinary meaning, such as: information in a class file	Data structures having the following general format: attibute_info { u2 attribute_name_index; u4 attribute _length; u1 info[attribute_length];

[21] The term "terminal" is found in the '317 patent at claims 1, 10, 25, 30, 31, 40, 55, 59, 62, 64, 84-86, 93, & 94; and the '485 patent at claims 2, 38, 40, & 42.

	}

The essential dispute between the parties is whether the term "attributes" should be construed as part of a means-plus-function element. *See* RESPONSE at 30. Claim 66 reads: "The microcontroller of claim 65, wherein the compiled form includes attributes, and the converter comprises a means for including attributes required by the interpreter while not including the attributes not required by the interpreter." '317 patent at claim 66. Because the term "attributes" is introduced before the means-plus-function limitation in claim 66, it is not part of the means-plus-function limitations. Accordingly, the Court declines to adopt Defendants' construction limiting "attributes" to the specific structure disclosed in the specification. One of ordinary skill in the art would understand that in the Java programming language, one of the basic sections in a class file is "attributes," which give general information about the particular class defined by the file. Accordingly, the Court construes "attributes" as "information in a class file."

b. **"means for translating …"**[22]

The parties agree that "means for translating" is a means-plus-function element governed by 35 U.S.C. § 112 ¶ 6 but disagree over what constitutes the function. Plaintiff argues that the function is "means for translating from the byte codes in the compiled form to byte code in a format suitable for interpretation by the interpreter," and the remainder of the claim, steps b.1 through b.3 recite the structure. Defendants, on the other hand, argue that the recited function encompasses steps b.1. through b.3. As will be explained in more detail below, the Court finds that the "means for translating" terms are not "means-plus-function" terms governed by 112 ¶ 6 because the claims recite the complete structure for "means for translating."

[22] The entirety of the disputed "means for translating" terms and the parties' proposed constructions are set forth in Appendix A to this Order.

As an initial matter, although the parties agree to a particular claim construction, the Court is not obligated to accept a construction that is contrary to law. *See Rodime PLC v. Seagate Tech., Inc.*, 174 F.3d 1294, 1302 (Fed. Cir. 1999) (noting that a party's concession that a term is governed by §112 ¶ 6 does not relieve the court of its responsibility to interpret the claims as a matter of law). Moreover, it is bedrock law that "[m]eans-plus-function claiming applies only to purely functional limitations that do not provide the structure that performs the recited function." *Phillips*, 415 F.3d at 1311 (citing *Watts v. XL Sys. Inc.*, 232 F.3d 877, 880-81 (Fed.Cir.2000)). Although there is a rebuttable presumption that § 112 ¶ 6 applies "[i]f the word 'means' appears in a claim element in association with a function," that presumption does not apply where the claim language itself provides the structure that performs the recited function. *Micro Chem., Inc. v. Great Plains Chem., Co.*, 194 F.3d 1250, 1257 (Fed. Cir. 1999) (citations omitted); *see also Callicrate v. Wadsworth Mfg., Inc.*, 427 F.3d 1361, 1368 (Fed. Cir. 2005) (same). Thus, "where a claim recites a function, but then goes on to elaborate sufficient structure, material, or acts within the claim itself to perform entirely the recited function, the claim is not in means-plus-function format." *Sage Products, Inc. v. Devon Indus., Inc.*, 126 F.3d 1420, 1427-28 (Fed. Cir. 1997) (citing *Cole v. Kimberly-Clark Corp.*, 102 F.3d 524 (Fed. Cir. 1996)).

With respect to computer-implemented inventions, i.e. software based inventions, the Federal Circuit has been particularly cautious to avoid purely functional claiming. *See Aristocrat Techs. Australia Pty Ltd. v. Int'l Gaming Tech.,* 521 F.3d 1328, 1333 (Fed. Cir. 2008) ("Because general purpose computers can be programmed to perform very difference tasks in very different ways, simply disclosing a computer as the structure designed to perform a particular function does not . . . [comply with] section 112 paragraph 6"); *Altiris, Inc. v. Symantec Corp.* 318 F.3d

1363, 1376 (Fed. Cir. 2003) (noting that the term "commands" is equivalent to "software" which does not connote adequate structure). As a result, the patentee is required to disclose adequate structure in the form of an algorithm. *See Aristocrat Techs. Australia Pty Ltd.*, 521 F.3d at 1333 (finding an algorithm adequate structure for a computer implemented means-plus-function limitation). The corresponding structure to a computer implemented "means for" term may be disclosed in the form of a step-by-step procedure for implementing the recited function as an algorithm, in prose, or in flow-chart form. *See Ergo Licensing, LLC v. CareFusion 303, Inc.*, 673 F.3d 1361, 1365 (Fed. Cir. 2012) (citing *Typhoon Touch Tech., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1385 (Fed. Cir. 2011)).

In this case, the disputed term, "means for translating" is presumed to be governed by § 112 ¶ 6 because it uses the phrase "means for." As shown below, "means for" is immediately followed by a clearly defined function, "translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter." The claim language signals the end of the recited function and beginning of the structure of the term by the word "by:" [23]

> A microcontroller having a set of resource constraints operable within the set of resource constraints and comprising:
>
> A memory, and
>
> An interpreter loaded in memory and operable within the set of resource constraints, the microcontroller having: at least one application loaded in the memory . . . wherein the at least one application is generated by a programmable environment comprising:
>
> (a) A compiler . . .
>
> (b) A converter . . . wherein the converter comprises means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by:
>
> b.1) recording all jumps and their destinations in the original byte codes;

[23] The Court notes that the recited steps are not identical in all three iterations of the "means for translating" claims. However, all three iterations of the "means for translating" claims include the word "by" signifying the end of the recited function and the beginning of the recited structure. *See* '727 patent at claim 10; '485 at claim 14. Moreover, both the other "means for translating" claims patent recite complete algorithms as a series of steps. They differ only in their numbering.

b.2) performing a conversion operation selected from the group:
b.2.1) converting specific byte codes into equivalent generic byte codes;
b.2.2) modifying byte code operands from references using identifying strings to references using unique identifiers; and
b.2.3.) renumbering byte codes in the compiled form to equivalent byte codes in an instruction set supported by an interpreter on the integrated circuit card; and
b.3) relinking jumps for which the destination address is affected by the conversion operation.

'317 patent at claim 65.

Thus, the claims recite a structurally complete "means for translating" because they recite a "programmable environment"[24] and include all the necessary algorithmic steps to perform the "means for translating" function. One of ordinary skill in the art would understand "programmable environment" as including a processor for executing commands or instructions. Specifically, while not explicitly reciting a "processor" for performing the recited steps of the "means for translating" limitation, the claims clearly contemplate a "programmable environment" that includes a processor for executing the "compiler" and the "converter" where the "converter" includes the "means for translating." *See, e.g.*, '317 C1 patent at 3:45 (claim 65) ("a converter for post processing").

The patentee's recitation of a step-by-step procedure for performing the "means for translating" function, i.e., steps b.1 through b.3, provides even greater detail and clarity than the algorithm found sufficient in *Typhoon Touch Technologies.* In *Typhoon Touch Technologies v. Dell, Inc.*, the Federal Circuit found that the specification adequately disclosed the structure for the term "means for cross-referencing." 659 F.3d at 1386. Specifically, the Court found that the patent's recitation of "'[c]ross-referencing entails the matching of entered responses with a library of possible responses, and, if a match is encountered, displaying the fact of the match,

[24] The '485 patent and the '727 patent recite a "programming environment," which also necessarily includes a processor. *See* '485 patent at claim 7; '727 patent at claim 3.

otherwise alerting the user, or displaying information stored in memory fields associated with the library entry"' was sufficient structure under §112 ¶ 6. In this case, the claim clearly lays out steps b.1 through b.3 for performing the recited function that are executed by implied processor in the programmable environment and therefore recite adequate structure. *See Aristocrat Tech. Australia Pty Ltd.*, 521 F.3d at 1338 (quoting *WMS Gaming, Inc. v. Int'l Gaming Tech.,* 184 F.3d 1339, 1349 (Fed. Cir. 1999) (patentee was required to "disclose the algorithm that transforms the general purpose microprocessor to a 'special purpose computer programmed to perform the disclosed algorithm.'"); *cf. Altiris, Inc.*, 318 F.3d at 1376 (rejecting plaintiff's argument that "commands" provided sufficient structure "because 'commands' (i.e., software) is so broad as to give little indication of the particular structure used here and is described only functionally, one must still look to the specification for an adequate understanding of the structure of that software.").

Indeed, Plaintiff requests the Court to hold that "the corresponding structure is a computer programmed to perform the algorithm/steps explicitly set forth in the claims." *Id.* at 24. However, as noted above, if the corresponding structure is set forth in the claims as advocated by Plaintiffs, the claim term cannot fall under § 112 ¶ 6 and is therefore not a "means-plus-function" claim. *See Cole v. Kimberly-Clark Corp.*, 102 F.3d 524, 531 (Fed. Cir. 1996) ("To invoke this statute [§112 ¶ 6], the alleged means-plus-function claim element must not recite a definite structure which performs the described function."). Thus, for the reasons stated above, the Court finds that "means for translating" is not a means-plus-function term governed by §112 ¶ 6. Beyond that finding, however, the Court declines to construe "means for translating."

CONCLUSION

For foregoing reasons, the Court adopts the constructions set forth above.

So ORDERED and SIGNED this 28th day of June, 2012.

JOHN D. LOVE
UNITED STATES MAGISTRATE JUDGE

Appendix A

Disputed Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Court's Construction
means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by: b.1) recording all jumps and their destinations in the original byte codes; b.2) performing a conversion operation selected from the group: b.2.1) converting specific byte codes into equivalent generic byte codes; b.2.2) modifying byte code operands from references using identifying strings to references using unique identifiers; and b.2.3.) renumbering byte codes in the compiled form to equivalent byte codes in an instruction set supported by an interpreter on the integrated circuit card;	This limitation is governed by § 112, ¶ 6. **RECITED FUNCTION:** translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter **CORRESPONDING STRUCTURE**: computer programmed to perform the algorithm of: b.1) recording all jumps and their destinations in the original byte codes; b.2) performing a conversion operation selected from the group: b.2.1) converting specific byte codes into equivalent generic byte codes; b.2.2) modifying byte code operands from references using identifying strings to references using unique identifiers; and b.2.3.) renumbering byte codes in the compiled form to equivalent byte codes in an instruction set supported by an interpreter on the integrated circuit card; and	Indefinite	This limitation is not governed by § 112 ¶ 6.

Disputed Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Court's Construction
and b.3) relinking jumps for which the destination address is affected by the conversion operation. '317 Patent, Claim 65	b.3) relinking jumps for which the destination address is affected by the conversion operation. Alternatively, should the Court conclude that the algorithm steps are part of the recited function, rather than the corresponding structure as Gemalto proposes, Gemalto identifies the following portions of the specification as part of the corresponding structure for those steps: "recording" step: column 10:16-17, 22-26; "converting specific" step: column 10:35-44; "modifying" step: column 10:52-54, 57-61; "renumbering" step: column 11:4-10, 18-23; "relinking" step: column 11:36-41.		
means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by: using at least one step in a process including the	This limitation is governed by § 112, ¶ 6. **RECITED FUNCTION:** translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter **CORRESPONDING**	Indefinite	This limitation is not governed by § 112 ¶ 6.

Disputed Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Court's Construction
steps: a) recording all jumps and their destinations in the original byte codes; b) converting specific byte codes into equivalent generic byte codes or viceversa; c) modifying byte code operands from references using identifying strings to references using unique identifiers; and d) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and relinking jumps for which destination address is effected by conversion step a), b), c), or d) '485 Patent, Claim 14[25]	**STRUCTURE**: computer programmed to perform the algorithm of: using at least one step in a process including the steps: a) recording all jumps and their destinations in the original byte codes; b) converting specific byte codes into equivalent generic byte codes or viceversa; c) modifying byte code operands from references using identifying strings to references using unique identifiers; and d) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and relinking jumps for which destination address is effected by conversion step a), b), c), or d) Alternatively, should the Court conclude that the algorithm steps are part of the recited function, rather than the corresponding structure as Gemalto proposes, Gemalto identifies the following portions of the specification as part		

[25] Claim 14 depends on Claim 10 which depends on Claim 7 which recites an application "generated by a programming environment."

Disputed Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Court's Construction
	of the corresponding structure for those steps: "recording" step: column 10:16-17, 22-26; "converting specific" step: column 10:35-44; "modifying" step: column 10:52-54, 57-61; "renumbering" step: column 11:4-10, 18-23; "relinking" step: column 11:36-41.		
means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by: recording all jumps and their destinations in the original byte codes: converting the compiled form using at least one step in a process including the steps: a) converting specific byte codes into equivalent generic byte codes or viceversa; b) modifying byte code operands from references	This limitation is governed by § 112, ¶ 6. **RECITED FUNCTION:** translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter **CORRESPONDING STRUCTURE**: computer programmed to perform the algorithm of recording all jumps and their destinations in the original byte codes: converting the compiled form using at least one step in a process including the steps: a) converting specific byte codes into equivalent generic byte codes or vice versa;	Indefinite	This limitation is not governed by § 112 ¶ 6.

Disputed Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Court's Construction
using identifying strings to references using unique identifiers; and c) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and relinking jumps for which destination address is effected by conversion step a), b), or c) '727 Patent, Claim 10[26]	b) modifying byte code operands from references using identifying strings to references using unique identifiers; and c) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and relinking jumps for which destination address is effected by conversion step a), b), or c) Alternatively, should the Court conclude that the algorithm steps are part of the recited function, rather than the corresponding structure as Gemalto proposes, Gemalto identifies the following portions of the specification as part of the corresponding structure for those steps: "recording" step: column 10:16-17, 22-26; "converting specific" step: column 10:35-44; "modifying" step: column 10:52-54, 57-61;		

[26] Claim 10 depends on Claim 6 which depends on Claim 3 which recites an application "generated by a programming environment."

Disputed Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Court's Construction
	"renumbering" step: column 11:4-10, 18-23; "relinking" step: column 11:36-41.		