BUNDESPATENTGERICHT

IM NAMEN DES VOLKES

URTEIL

Verkündet am
31. Juli 2014
…

6 Ni 3/14 (EP)

(Aktenzeichen)

In der Patentnichtigkeitssache

…

betreffend das europäische Patent 0 932 865
(DE 697 14 752)

hat der 6. Senat (Nichtigkeitssenat) des Bundespatentgerichts auf Grund der mündlichen Verhandlung vom 31. Juli 2014 durch den Vorsitzenden Richter Voit, die Richterin Kortge, den Richter Dipl.-Phys. Dr. Schwengelbeck, die Richterin Dipl.-Phys. Univ. Dr. Otten-Dünnweber und den Richter Dipl.-Ing. Altvater

für Recht erkannt:

I. Das europäische Patent 0 932 865 wird im Umfang des Patentanspruchs 1 mit Wirkung für das Hoheitsgebiet der Bundesrepublik Deutschland für nichtig erklärt.

II. Die Kosten des Rechtsstreits trägt die Beklagte.

III. Das Urteil ist im Kostenpunkt gegen Sicherheitsleistung in Höhe von 120 % des zu vollstreckenden Betrages vorläufig vollstreckbar.

T a t b e s t a n d

Die Beklagte ist Inhaberin des auch mit Wirkung für das Hoheitsgebiet der Bundesrepublik Deutschland erteilten europäischen Patents 0 932 865 (Streitpatent), das aus der PCT-Anmeldung PCT/US97/18999 - veröffentlicht am 7. Mai 1998 als WO 98/19237 A1 - hervorgeht und am 22. Oktober 1997 unter Inanspruchnahme der Priorität der US-amerikanischen Patentanmeldung 29,057 vom 25. Oktober 1996 angemeldet worden ist. Das Streitpatent ist in der Verfahrenssprache Englisch veröffentlicht und wird beim Deutschen Patent- und Markenamt unter der Nummer DE 697 14 752 geführt. Es trägt die Bezeichnung: „Using a high level programming language with a microcontroller” (übersetzt: „Verwendung einer hohen Programmiersprache in einem Mikrokontroller“) und umfasst in der erteilten

Fassung 33 Ansprüche, von denen nur der Anspruch 1 angegriffen wird. Anspruch 1 lautet in der Verfahrenssprache Englisch wie folgt:

> „**1.** A microcontroller (10) having a set of resource constraints and comprising: a memory, an interpreter (16) loaded in memory and operating within the set of resource constraints, the microcontroller (10)
> **characterized by** 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 programming environment comprising:
> a) a compiler (22) for compiling application source programs (20) in high level language source code form into a compiled form (24),
> b) a converter (26) for post processing the compiled form (24) into a minimized form (27) suitable for interpretation by the interpreter (16).”

In der deutschen Übersetzung hat Anspruch 1 folgenden Wortlaut:

> „**1.** Ein Mikrocontroller (10) mit einem Set von Ressourcen-Vorgaben und bestehend aus:
> einem Speicher,
> einem im Speicher geladenen Interpreter (16), welcher innerhalb des Sets von Ressourcen-Vorgaben funktioniert, dem Mikrocontroller (10),
> **dadurch gekennzeichnet, dass** er:
> mindestens eine durch den Interpreter zu übersetzende im Speicher geladene Applikation hat, wobei mindestens eine Applikation durch eine Programmierungsumgebung generiert wird, die folgenden Elementen umfasst:

a) einem Compiler (22), welcher Anwendungs-Quellprogramme (20) in Quellcodeform höherer Programmiersprache in eine kompilierte Form (24) kompiliert,
b) einem Konverter (26) zur Umformatierung der kompilierten Form (24) in eine sich zur Übersetzung durch den Interpreter (16) eignende minimierte Form (27).“

Im Übrigen wird auf die Streitpatentschrift EP 0 932 865 B1 (NK1) und deren deutsche Übersetzung DE 697 14 752 T2 (NK2) Bezug genommen.

Die Klägerin macht den Nichtigkeitsgrund der mangelnden Patentfähigkeit des Gegenstands des erteilten Anspruchs 1 geltend (Art. II § 6 Abs. 1 Nr. 1 IntPatÜG i. V. m. Art. 138 Abs. 1 Buchstabe a) EPÜ). Als weiteren Nichtigkeitsgrund führt sie die unzulässige Erweiterung an (Art. II § 6 Abs. 1 Nr. 3 IntPatÜG i. V. m. Art. 138 Abs. 1 Buchstabe c) EPÜ). Sie bezieht sich dabei auf folgenden Stand der Technik:

NK5 Dissertation von Olivier Caron, „Méthodologies de conception et d’évaluation d’architectures R.I.S.C. adaptées aux futures cartes à microprocesseur”, vom 20. Januar 1994, Universität Lille mit deutscher Übersetzung NK5a;

NK6a Olivier Caron et al., „New architectures for smart cards: the OCEAN approach”, in Proceedings of the Third International Workshop on Hardware/Software Codesign, Grenoble, 1994, Seiten 148 bis 155, mit deutscher Übersetzung NK6c;

NK6b Olivier Caron u. Georges Grimonprez, „OCEAN: A C compiler for new smart card applications“, in Proceedings of the International Conference in Compiler Construction, April 1994, Edinburgh, Seiten 31 bis 39, mit deutscher Übersetzung NK6d;

NK7 FR 2 667 171 A1 mit deutscher Übersetzung NK7a

NK8 Java™ Card API Frequently Asked Questions, Version 1.1, October 25, 1996, mit deutscher Übersetzung NK8a;

NK9 Abbildung 64 auf Seite 115 der NK5

NK10 „Sun Microsystems, Inc. Announces Java Card API“, Pressemitteilung vom 29. Oktober 1996 auf der Internetseite der Sun Microsystems, Inc. (www.sun.com), Auszug aus dem Internet-Archiv (web.archive.org) vom 20. Dezember 1996;

NK11 Java Commerce Home Page mit Hinweis auf Java Card API, Auszug aus dem Internet-Archiv (web.archive.org) vom 8. Juni 1997;

NK12 Scott B. Guthery, „Java Card: Internet Computing on a smart card“, in IEEE Internet Computing, Januar/Februar 1997, Vol. 1, No. 1, Seiten 57 bis 59;

NK15 Bildschirmfoto der NK11 mit Download-Link;

NK16 „PersonalJava and EmbeddedJava Development Tools“, Internetseite www.javasoft.com vom 30. September 1997, Auszug der aus dem Internet-Archiv (web.archive.org) vom 8. Oktober 1997.

Die Klägerin ist der Auffassung, der Gegenstand des erteilten Anspruchs 1 sei gegenüber den Druckschriften NK5, NK6a mit NK6b, NK7, NK8 und NK16 weder neu, noch beruhe er - auch unter Berücksichtigung der NK12 - auf erfinderischer Tätigkeit. Insbesondere könnten dessen Software-Merkmale den beanspruchten Mikrocontroller nicht erfinderisch weiterbilden. Sie verweist dazu auf BPatG, Urt. v. 14. November 2013 – 2 Ni 4/12 (EP), BeckRS 2014, 08225. Die Druckschriften NK10 und NK11 belegten, dass die Entgegenhaltung NK8 vor dem Anmeldetag des Streitpatents öffentlich zugänglich gewesen sei. Dem erteilten Anspruch 1 stehe auch nicht der Anmeldetag der Prioritätsanmeldung zu, weil sich diese ausschließlich auf Applikationen in der Programmiersprache Java beziehe. Ferner sei er im Hinblick auf die Verallgemeinerung von Konsolidieren und Komprimieren in Minimieren und in Bezug auf die generalisierte Form des Interpreters und des Konverters unzulässig erweitert.

Die Klägerin beantragt,

> das europäische Patent 0 932 865 im Umfang des Patentanspruchs 1 für das Hoheitsgebiet der Bundesrepublik Deutschland für nichtig zu erklären.

Die Beklagte beantragt,

> die Klage kostenpflichtig abzuweisen,
> hilfsweise verteidigt sie das Streitpatent gemäß dem mit Schriftsatz vom 14. Februar 2013 eingereichten Hilfsantrag 1, weiter hilfsweise nach Maßgabe der Hilfsanträge 2 bis 5 im Schriftsatz vom 23. Mai 2014.

In der Fassung des Hilfsantrags 1, der eine Kombination der erteilten Ansprüche 1 und 3 darstellt, lautet Patentanspruch 1 wie folgt (Änderungen gegenüber der Fassung gemäß Streitpatentschrift sind durch Unterstreichung kenntlich gemacht):

„1. A microcontroller (10) having a set of resource constraints and comprising: a memory, an interpreter (16) loaded in memory and operating within the set of resource constraints, the microcontroller (10) characterized by 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 programming environment comprising:

a) a compiler (22) for compiling application source programs (20) in high level language source code form into a compiled form (24),
b) a converter (26) for post processing the compiled form (24) into a minimized form (27) suitable for interpretation by the interpreter (16).

wherein the compiled form (24) is in a standard Java class file format and the converter (26) accepts as input compiled form (24) in the standard Java class file format and produces output (27) in a form suitable for interpretation by the interpreter (16)."

Hilfsantrag 2 ist gegenüber der Fassung gemäß Hilfsantrag 1 in der letzten Merkmalsgruppe wie folgt geändert (Änderungen gegenüber Hilfsantrag 1 sind durch Unterstreichung kenntlich gemacht):

„1. A microcontroller […]
wherein the compiled form (24) is in a standard Java class file format and the converter (26) accepts as input compiled form (24) in the standard Java class file format and produces output (27) in a class file format suitable for interpretation by the interpreter (16)."

Hilfsantrag 3 ist gegenüber der Fassung gemäß Hilfsantrag 2 in der letzten Merkmalsgruppe wie folgt geändert (Änderungen gegenüber Hilfsantrag 2 sind durch Streichung bzw. Unterstreichung kenntlich gemacht):

„1. A microcontroller […]
wherein the compiled form (24) is constituted by a plurality of Java class files in a standard Java class file format and the converter (26) accepts as input the compiled form (24) in the standard Java class file format and produces as output (27) ~~in~~ a single class file in a form ~~format~~ suitable for interpretation by the interpreter (16)."

Hilfsantrag 4 hat gegenüber der Fassung gemäß Hilfsantrag 3 in der letzten Merkmalsgruppe folgenden geänderten Wortlaut (Änderungen gegenüber Hilfsantrag 3 sind durch Unterstreichung kenntlich gemacht):

„1. A microcontroller […]
wherein the compiled form (24) is constituted by a plurality of Java class files in a standard Java class file format and the converter (26) accepts as input the compiled form (24) in the standard Java class file format and produces as output (27) a single class file in a class file format suitable for interpretation by the interpreter (16)."

Hilfsantrag 5 ist gegenüber der Fassung gemäß Hilfsantrag 4 wie folgt geändert (Änderungen gegenüber Hilfsantrag 4 sind durch Unterstreichung kenntlich gemacht):

„1. A microcontroller (10) having a set of resource constraints and comprising: a memory, an interpreter (16) loaded in memory and operating within the set of resource constraints, the microcontroller (10) characterized by 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 programming environment comprising:

a) a compiler (22) for compiling application source programs (20) in high level language Java source code form into a compiled form (24),
b) a converter (26) for post processing the compiled form (24) into a minimized form (27) suitable for interpretation by the interpreter (16).

wherein the compiled form (24) is constituted by a plurality of Java class files in a standard Java class file format and the converter (26) accepts as input the compiled form (24) in the standard Java class file format and produces as output (27) a single class file in a class file format suitable for interpretation by the interpreter (16).”

Die Beklagte hält den Gegenstand des Anspruchs 1 in der erteilten Fassung, zumindest aber die Anspruchsfassung gemäß eines ihrer Hilfsanträge, gegenüber sämtlichen Angriffen für patentfähig, im Gegensatz zu den Ausführungen der Klägerin.

Der Senat hat den Parteien mit Schriftsatz vom 1. April 2014 einen frühen gerichtlichen Hinweis gemäß § 83 Abs. 1 PatG zukommen lassen.

Wegen weiterer Einzelheiten wird auf das Protokoll der mündlichen Verhandlung sowie auf den gesamten Akteninhalt, insbesondere auf die Schriftsätze der Parteien mit sämtlichen Anlagen Bezug genommen.

E n t s c h e i d u n g s g r ü n d e

Die zulässige Klage ist begründet. Sie führt zur Nichtigerklärung des Streitpatents im Umfang des Patentanspruchs 1 mit Wirkung für das Hoheitsgebiet der Bundesrepublik Deutschland, denn der Anspruch 1 des Streitpatents hat weder in der erteilten Fassung noch in einer der von der Beklagten hilfsweise beantragten Fassungen Bestand (Art. II § 6 Abs. 1 Nr. 1 IntPatÜG, Art. 138 Abs. 1 Buchst. a, Art. 56 EPÜ).

I.

1. Soweit die Beklagte den Angriff der Klägerin auf die Neuheit und die erfinderische Tätigkeit des Streitpatents insbesondere in der Fassung der Hilfsanträge 2 bis 5 mittels der NK16 als verspätet gerügt hat, ist dies nicht begründet.

   Der am 2. Juli 2014 per Telefax eingegangene Schriftsatz der Klägerin mit der Druckschrift NK16 ist innerhalb der einmonatigen Frist bei Gericht eingegangen, die im frühen gerichtlichen Hinweis für eine Stellungnahme auf gegnerisches Vorbringen gesetzt worden war. Da der zum gerichtlichen Hinweis Stellung nehmende Schriftsatz der Beklagten vom 23. Mai 2014 mit ihren Hilfsanträgen 2 bis 5 am 2. Juni 2014 an die Klägerin zugestellt worden ist, hat die einmonatige Erwiderungsfrist für die Klägerin erst am 2. Juli 2014 geendet.

   Die Ausführungen der Klägerin, dass dem erteilten Anspruch 1 auch nicht der Anmeldetag der Prioritätsanmeldung zustehe, gehen ins Leere, da die im Hinblick auf die erfinderische Tätigkeit relevante Druckschrift NK7 bereits am 27. März 2002 und damit lange vor dem Anmeldetag des Streitpatents und dem strittigen Prioritätsdatum veröffentlicht worden ist.

2. Das Streitpatent betrifft einen Mikrocontroller bzw. die Verwendung einer hohen Programmiersprache in einem Mikrocontroller (vgl. Bezeichnung des Streitpatents in der deutschen Übersetzung, DE 697 14 752 T2).

   Das Streitpatent geht dabei beispielgebend vom Einsatz von Java als höherer Programmiersprache in einem in seinen Ressourcen begrenzten Mikrocontroller, insbesondere in einer Chipkarte, aus. Für typischerweise in einer Chipkarte angewandte Mikrocontroller gebe es keine herkömmlichen Java-Implementierungen, sondern diese würden üblicherweise nur für Mikroprozessoren bereitgestellt (ebd., S. 2, Zn. 14-19 und Z. 29 bis S. 3, Z. 3). Zudem würden Applikationen für Chipkarten wegen der vorgegebenen begrenzten Ressourcen typischerweise in einer niedrigen Programmiersprache (z. B. Assembler-Sprache) geschrieben, um Speicherplatz zu sparen (ebd., S. 4, zweiter Abs.). Das Streitpatent beschreibt sinngemäß weiter, dass die Einführung öffentlicher Digitalnetzwerke (z. B. Internet) den Chipkarten neue Anwendungsgebiete eröffnet habe, was zu einem Bedarf an neuen, auf die Karte ladbaren Anwendungen geführt habe. Da diese Anwendungen jedoch die Kartensicherheit nicht kompromittieren sollen, könnten hierzu typischerweise keine herkömmlichen Karten verwendet werden, die in niedrigen Sprachen programmiert seien (ebd., S. 6, zweiter Abs.).

   Software-Anwendungen seien in JAVA so konzipiert, dass sie ohne Änderungen auf verschiedenen Plattformen lauffähig sind, wobei zur Ausführung der Anwendungen eine so genannte JAVA Virtual Machine (JVM) benötigt werde, die zusammen mit den Bibliotheken als JAVA Runtime Environment (JRE) diene. Damit eine Java-Applikation auf einer spezifischen Plattform ablauffähig sei, müsse eine Java Virtual Machine-Implementierung bereitgestellt werden, welche innerhalb der Plattform-Vorgaben funktioniere. Um eine Anwendung ausführen zu können, würde diese in Byte-Code enthaltende Klassendateien kompiliert, wobei es sich beim Byte-Code um Anweisungen für die Java Virtual Machine (JVM) handele. Solle eine Java-Anwendung dann auf einer bestimmten Plattform zum Ablauf gebracht werden, werde der vorher kompi-

lierte, als Klassendateien vorliegende Byte-Code durch die Java Virtual Machine für die gewählte Plattform interpretiert (ebd., S. 1, Z. 10 bis S. 2, Z. 13).

Zu bekannten Möglichkeiten der Programmierung von Chipkarten mit höheren Programmiersprachen im Stand der Technik wird im Streitpatent auf die französische Patentanmeldung FR 2 667 171 A1 (Druckschrift NK7) verwiesen. Als Nachteil dieses Standes der Technik benennt das Streitpatent, dass dort keine Themenbereiche im Zusammenhang mit der Datensicherheit angesprochen würden oder wie eine Programmierungsumgebung bereitgestellt werden könne, die einen Programmierer befähige, ein Programm für eine Chipkarte in einer Programmiersprache wie JAVA zu erstellen und das Programm mittels einem Interpreter auf der Chipkarte, welcher innerhalb der Ausführungsvorgaben der Chipkarte arbeitet, ausführen zu können (vgl. DE 697 14 752 T2, S. 4, dritter Abs.).

Die Beschreibung des Streitpatents nennt keine explizite Aufgabe; lediglich bei der Würdigung des Standes der Technik wird als dessen Nachteil auf das Fehlen von Sicherheitsmechanismen verwiesen. Die objektive technische Problemstellung, d.h. das durch die Erfindung – für den Fachmann erkennbar – tatsächlich zu lösende technische Problem (vgl. bspw. BGH, Urt. v. 23. Januar 1990 – X ZR 75/87, GRUR 1991, 522 – Feuerschutzabschluss; BGH, Urt. v. 12. Februar 2003 – X ZR 200/99, GRUR, 2003, 693 – Hochdruckreiniger) besteht darin, einen Mikrocontroller in einer hohen Programmiersprache zu programmieren, wobei die Programmierung den besonderen technischen Randbedingungen („set of resource constraints“) des Mikrocontrollers Rechnung tragen soll.

Diese Aufgabe richtet sich an einen Fachmann, der Ingenieur der Elektrotechnik oder Technischen Informatik mit Hochschulabschluss ist und der eine mehrjährige Erfahrung in der Programmierung von Mikrocontroller-basierten Systemen und dem damit verbundenen Gebiet der Software-Programmierungsumgebungen vorweist.

Zur Lösung der angesprochenen Problemstellung sieht das Streitpatent in Anspruch 1 vor, dass der Mikrocontroller einen in den Speicher geladenen Interpreter aufweist, welcher innerhalb eines Sets von Ressourcen-Vorgaben des Mikrocontrollers funktioniert und der eine in den Speicher geladene Applikation übersetzen kann. Die Applikation soll hierzu durch eine Programmierungsumgebung generiert werden, die einen Compiler zum Kompilieren des Anwendungs-Quellprogramms und einen Konverter zum Umwandeln der kompilierten Form in eine sich zur Übersetzung (d. h. zum Interpretieren) durch den Interpreter eignende minimierte Form aufweist.

3. Patentanspruch 1, mit dem die Beklagte das Patent gemäß Hauptantrag verteidigt, lautet unter Hinzufügen einer Gliederung:

M1 „A microcontroller (10) having a set of resource constraints and comprising:
M1.1 a memory,
M1.2 an interpreter (16) loaded in memory and
M1.2.1 operating within the set of resource constraints,

the microcontroller (10) characterized by having:

M2 at least one application loaded in the memory to be interpreted by the interpreter,
M2.1 wherein the at least one application is generated by a programming environment comprising:

M2.1.1 a) a compiler (22) for compiling application source programs (20) in high level language source code form into a compiled form (24),

M2.1.2 b) a converter (26) for post processing the compiled form (24) into a minimized form (27) suitable for interpretation by the interpreter (16).“

In deutscher Übersetzung lautet der mit einer Merkmalsgliederung versehene erteilte Patentanspruch 1 (Hauptantrag) wie folgt:

M1 „Ein Mikrocontroller (10) mit einem Set von Ressourcen-Vorgaben und bestehend aus:
M1.1 einem Speicher,
M1.2 einem im Speicher geladenen Interpreter (16),
M1.2.1 welcher innerhalb des Sets von Ressourcen-Vorgaben funktioniert,

dem Mikrocontroller (10), dadurch gekennzeichnet, dass er:

M2 mindestens eine durch den Interpreter zu übersetzende im Speicher geladene Applikation hat,
M2.1 wobei mindestens eine Applikation durch eine Programmierungsumgebung generiert wird, die folgenden Elementen umfasst:

M2.1.1 a) einem Compiler (22), welcher Anwendungs-Quellprogramme (20) in Quellcodeform höherer Programmiersprache in eine kompilierte Form (24) kompiliert,
M2.1.2 b) einem Konverter (26) zur Umformatierung der kompilierten Form (24) in eine sich zur Übersetzung durch den Interpreter (16) eignende minimierte Form (27).“

Patentanspruch 1 gemäß Hilfsantrag 1 fügt dem Anspruch 1 gemäß Hauptantrag im Anschluss an Merkmal M2.1.2 die folgenden Merkmale hinzu:

M2.1.3 „wherein the compiled form (24) is in a standard Java class file format and

M2.1.4 the converter (26) accepts as input compiled form (24) in the standard Java class file format and produces output (27) in a form suitable for interpretation by the interpreter (16).”

Patentanspruch 1 gemäß Hilfsantrag 2 basiert auf Hilfsantrag 1 und unterscheidet sich von diesem im geänderten Merkmal M2.1.4* (Änderungen gegenüber Hilfsantrag 1 sind durch Unterstreichung hervorgehoben):

M2.1.4* „the converter (26) accepts as input compiled form (24) in the standard Java class file format and produces output (27) in a class file format suitable for interpretation by the interpreter (16).”

Patentanspruch 1 gemäß Hilfsantrag 3 basiert auf Hilfsantrag 1 und unterscheidet sich von diesem in den geänderten Merkmalen M2.1.3* und M2.1.4** (Änderungen gegenüber Hilfsantrag 1 sind durch Unterstreichung hervorgehoben):

M2.1.3* „wherein the compiled form (24) is constituted by a plurality of Java class files in a standard Java class file format and

M2.1.4** the converter (26) accepts as input the compiled form (24) in the standard Java class file format and produces as output (27) a single class file in a form suitable for interpretation by the interpreter (16).”

Patentanspruch 1 gemäß Hilfsantrag 4 verbindet die Änderungen des Hilfsantrags 2 mit den Änderungen des Hilfsantrags 3 in den Merkmalen M2.1.3* und M2.1.4*** (Änderungen gegenüber Hilfsantrag 1 sind durch Unterstreichung hervorgehoben):

M2.1.3* „wherein the compiled form (24) is constituted by a plurality of Java class files in a standard Java class file format and

M2.1.4*** the converter (26) accepts as input the compiled form (24) in the standard Java class file format and produces as output (27) a single class file in a class file format suitable for interpretation by the interpreter (16)."

Patentanspruch 1 gemäß Hilfsantrag 5 basiert auf Anspruch 1 gemäß Hilfsantrag 4 unter Änderung des Merkmals M2.1.1* (Änderungen gegenüber Hilfsantrag 4 sind durch Unterstreichung hervorgehoben):

M2.1.1* „a) a compiler (22) for compiling application source programs (20) in high level language Java source code form into a compiled form (24),"

4. Einzelne Merkmale und Merkmalsgruppen der verteidigten Fassungen des Anspruchs 1 bedürfen der Auslegung.

In Abgrenzung zu einem Mikroprozessor beinhaltet ein Mikrocontroller im Sinne des Streitpatents eine CPU, eine Speichereinheit und weitere Funktionselemente auf einem einzigen Haltleitersubstrat oder einem einzigen integrierten Schaltkreis. Der Mikrocontroller weist gemäß der deutschen Fassung des Streitpatents ein Set von „Ressourcen-Vorgaben" auf (englische Fassung des Streitpatents: „set of resource constraints"; vgl. jeweils Anspruch 1, Merkmale M1, M1.2.1); dabei wird auf die Größe des Speichers verwiesen, auf die ein Mikrocontroller üblicherweise Zugriff hat (vgl. deutsche Fassung des Streitpatents, S. 2, Z. 26 – S. 3, Z. 3). Hinweise auf eine „Vorgabe" der Ressourcen finden sich in der Beschreibung nicht. Es handelt sich somit entsprechend der englischen Fassung des Streitpatents um ein Set von Ressourcen-Beschränkungen des Mikrocontrollers.

Unter einem Compiler wird in der Regel ein Programm verstanden, das ein Quellprogramm, das in einer bestimmten Programmiersprache geschrieben ist, in ein Format übersetzt, das von einem Computer ausgeführt werden kann. Bei diesem Format kann es sich um eine vom Computer direkt ausführbare Maschinensprache oder auch um einen Byte-Code handeln. Ein Byte-Code ist in der Regel ein hardwareunabhängiger Zwischencode, der von einem Interpreter meist zur Laufzeit in Maschinensprache übersetzt wird. Im Anspruch 1 des Streitpatents ist die vom Compiler erzeugte kompilierte Form des Anwendungs-Quellprogramms ein Zwischencode, der durch einen Konverter nachbearbeitet und in eine für die Übersetzung durch den Interpreter geeignete Form gebracht wird. In der deutschsprachigen Fassung des Streitpatents wird für die Funktion des Compilers der Begriff „Kompilieren“ (engl. Fassung: „compiling“; vgl. Anspruch 1, Merkmal M2.1.1), für die Funktion des Interpreters der Begriff „Übersetzung“ verwendet (engl. Fassung: „interpretation“; vgl. Merkmale M2, M2.1.2).

Bei klassenbasierten Programmiersprachen liegt der kompilierte Zwischencode als Klassendateien in einem Klassendateiformat (engl. Fassung: „class file format“) vor. Ein Beispiel einer solchen klassenbasierten Programmiersprache, die einen in Klassendateien vorliegenden Byte-Code als Zwischencode und eine virtuelle Maschine als Interpreter verwendet, ist Java. In den Ausführungsbeispielen des Streitpatents dient eine „Java Virtual Machine“ („JVM“) als Interpreter einer Applikation, die in eine Chipkarte geladen ist, wobei die kompilierte Anwendung in einem Java-Klassendateiformat vorliegt. Der Begriff Klassendateiformat („class file format“) wird dabei in seiner üblichen Bedeutung für das Dateiformat eines als Klassendatei vorliegenden kompilierten und interpretierbaren Codes einer klassenbasierten Programmiersprache verwendet (vgl. bspw. deutsche Fassung des Streitpatents, S. 37, Zn. 17-24). Das Streitpatent unterscheidet hierbei zwischen einem standardmäßigen Java-Klassendateiformat (engl. Fassung: „standard Java class file format“) und einem kompakteren Karten-Klassendateiformat (engl. Fassung: „card class file format“), das für eine Verwendung in Chipkarten angepasst ist und

auf dem Standard-Java Klassendateiformat basiert (vgl. Streitpatent, Beschreibung zu Fig. 3 und 4 sowie Anhang A).

Der Patentanspruch 1 des Streitpatents weist funktionelle Merkmale zum Erzeugen eines Anwendungscodes (d. h. zum Erzeugen der in den Speicher des Mikrocontrollers zu ladenden Applikation) einer Programmierungsumgebung auf, die außerhalb des beanspruchten Mikrocontrollers in einer Programmierungsumgebung ablaufen (vgl. Merkmale M2.1.1, M2.1.2). Bei Sachansprüchen, bei denen der beanspruchte Gegenstand zumindest teilweise nicht unmittelbar durch (räumlich-körperlich oder funktionell umschriebene) Sachmerkmale, sondern durch ein Verfahren definiert ist, ist nach höchstrichterlicher Rechtsprechung durch Auslegung des Patentanspruchs zu ermitteln, ob und inwieweit sich aus dem angegebenen Verfahren Merkmale ergeben, die den Anspruchsgegenstand qualifizieren (vgl. BGH, Urt. v. 19. Juni 2001 – X ZR 159/98,GRUR 2001, 1129, Leitsatz - Zipfelfreies Stahlband; BGH, Urt. v. 19. Mai 2005 – X ZR 188/01, GRUR 2005, 749, dritter Leitsatz - Aufzeichnungsträger).
Anspruch 1 des Streitpatents sieht in den Merkmalen M2.1 bis M2.1.2 Verfahrensschritte zur Erzeugung des Anwendungscodes vor, der nach Merkmal M2 in den Speicher des Mikrocontrollers geladen und durch einen Interpreter interpretiert werden soll. Da der erteilte Anspruch 1 als Vorrichtungsanspruch auf einen Mikrocontroller gerichtet ist, kann dieser durch die Verfahrensmerkmale zur Erzeugung des geladenen Anwendungscodes in einer Programmierungsumgebung nur insofern charakterisiert werden, wie diese Verfahrensmerkmale eindeutig ihren Niederschlag in dem erzeugten Anwendungscode selbst finden und dieser Code – und damit der Mikrocontroller – dadurch von einem auf anderem Wege generierten Anwendungscode unterscheidbar ist.
Die einzige Anforderung, welche der in den Mikrocontroller geladene Anwendungscode nach dem erteilten Anspruch 1 erfüllen muss, ist dessen Eignung, vom Interpreter interpretiert werden zu können („[…] suitable for interpretation by the interpreter“; vgl. Merkmale M2.1 bis M2.1.2). Diese Anforderung lässt jedoch für den Fachmann anhand des Anwendungscodes keine gesicherten

Rückschlüsse auf die Zahl oder Art der Erzeugungsschritte oder die Eigenschaften des Quellprogramms bzw. eines kompilierten, nachbearbeiteten Zwischencodes zu. Denn weitergehende Angaben, aus denen eindeutig Inhalte des in den Speicher des Mikrocontrollers geladenen Anwendungscodes selbst ableitbar wären, sind den Verfahrensmerkmalen zur Codeerzeugung des erteilten Anspruchs 1 nicht zu entnehmen. Dies gilt in gleicher Weise für die weiteren verteidigten Fassungen des Anspruchs 1 in den Hilfsanträgen 1 bis 5.

Das Streitpatent sieht vor, dass ein Konverter in Bezug auf eine kompilierte Form des Anwendungsprogramms ein Ergebnis in „minimierter Form“ erzeugt (engl. Fassung: „a minimized form suitable for interpretation by the interpreter“, vgl. Merkmal M2.1.2). Der Begriff „minimiert“ findet in den ursprünglich eingereichten Anmeldungsunterlagen jedoch keine Verwendung. Mit der Verarbeitung des Anwendungscodes durch den Konverter befasst sich die Anmeldung in der veröffentlichten ursprünglichen Fassung nur in den Ausführungsbeispielen (vgl. insbesondere PCT-Veröffentlichung WO 98/19237 A1, Beschreibung zu Fig. 2, S. 16, Z. 5-22; sowie S. 19, Zn. 14-16 zu Fig. 4). Die Anmeldung spricht in diesem Zusammenhang von einem Konsolidieren und Komprimieren der Kartenklassen-Dateien (ebd., „[…] which consolidates and compresses the files […]“). Daneben zeigen weitere Ausführungsbeispiele im Zusammenhang mit der Verwendung von Java verschiedene Möglichkeiten, mit denen ein Karten-Klassendatei-Konverter („card class file converter“) den Umfang eines kompilierten Zwischencodes („Java class file“) verringern oder konsolidieren kann (vgl. Figuren 4 bis 11 mit zugehöriger Beschreibung). Entgegen den Ausführungen der Klägerin bezeichnet der Begriff „Minimieren“ nicht zwingend das Erzeugen eines absoluten Minimums, sondern kann auch allgemein ein „Verringern“ oder „Vermindern“ bezeichnen. In der Zielsetzung des Karten-Klassendatei-Konverters zur Nachverarbeitung („post processing“) eines kompilierten Java-Zwischencodes („Java class file“) für einen in seinen Ressourcen beschränkten Mikrocontroller erkennt der Fachmann in der „minimierten Form“ eine zusammenfassende Umschreibung für die in den Aus-

führungsbeispielen gezeigten Möglichkeiten zum Konsolidieren und Komprimieren von Java-Klassendateien.

Der erteilte Anspruch 1 des Streitpatents macht keine weiteren Angaben dazu, wie sich die Minimierung auf die kompilierte Form der Anwendungs-Quellprogramme auswirkt, sondern beschreibt die minimierte Form nur in Bezug auf ein Zwischenergebnis der Verarbeitung – den kompilierten Quellcode. Für diese minimierte Form des Anwendungscodes folgt aus Anspruch 1 des Streitpatents nur, dass dieser Code zur Interpretierung durch den Interpreter geeignet sein muss (vgl. Merkmal M2.1.2). Anhand dieses Anwendungscodes selbst ist jedoch nicht zu erkennen, ob der Code ausgehend von einem anderen Format minimiert oder direkt in einem minimierten Format erzeugt wurde, oder welche Maßnahmen und einzelnen Verarbeitungsschritte zu dem erzeugten Code geführt haben. Dass der Umfang des in den Mikrocontroller geladenen Anwendungscodes den vorhandenen Ressourcenanforderungen des Mikrocontrollers („resource constraints“) genügen muss, folgt für den Fachmann zwangsläufig aus der Anforderung an den geladenen Code, durch den Interpreter im Mikrocontroller interpretiert werden zu können. Dies gilt in gleicher Weise für die Nachverarbeitung von kompilierten Java-Klassendateien gemäß den weiteren verteidigten Fassungen des Anspruchs 1 in den Hilfsanträgen 1 bis 5.

II.

Der Anspruch 1 des Streitpatents (Hauptantrag) und die jeweiligen Ansprüche 1 der Hilfsanträge 1 bis 5 sind nicht patentfähig (Art. II § 6 Abs. 1 Nr. 1 IntPatÜG, Art. 138 Abs. 1 Buchst. a, Art. 56 EPÜ). Darüber hinaus liegen in Anspruch 1 des Hauptantrags und den jeweiligen Ansprüchen 1 der Hilfsanträge auch unzulässige Erweiterungen vor (Art. II § 6 Abs. 1 Nr. 3 IntPatÜG i.V.m. Art. 138 Abs. 1 Buchst. c EPÜ).

1. Hauptantrag

Der Gegenstand des erteilten Anspruchs 1 beruht gegenüber dem Stand der Technik gemäß Druckschrift NK7, FR 2 667 171 A1, nicht auf einer erfinderischen Tätigkeit (Art. II § 6 Abs. 1 Nr. 1 IntPatÜG i.V.m. Art. 138 Abs. 1 Buchst. a, Art. 56 EPÜ).

Denn aus Druckschrift NK7 ist ein Mikrocontroller mit einem Set von Ressourcen-Beschränkungen („resource constraints“) bekannt („*micro-circuit électronique*“, S. 8, Z. 20 – S. 9, Z. 1; sowie Fig. 1 mit Beschreibung, S. 10, Zn. 23-25; i.V.m. S. 5, Zn. 7-10 „*[…] cette contrainte de place limitée*“ / **Merkmal M1**). Dieser besteht aus einem Speicher („*mémoire*“ / **Merkmal M1.1**) und einem im Speicher geladenen Interpreter („*[…] programme I représentant le programme interpréteur directement exécutable tel quel par le microprocesseur 7*.“, S. 11, Zn. 21-26 und Fig. 1 / **Merkmal M1.2**). Dass der Interpreter zwangsläufig innerhalb des Sets von Ressourcen-Beschränkungen des Mikrocontollers funktioniert, liest der Fachmann bei der Ausführung des in den Mikrocontroller geladenen Interpreters und dem Interpretieren des Anwendungsprogramms im Mikrocontroller mit, da sonst eine Ausführung der Anwendung nicht möglich wäre (vgl. S. 11, Zn. 15-26 / **Merkmal M1.2.1**). Der Mikrocontroller weist dabei mindestens eine durch den Interpreter zu übersetzende und im Speicher geladene Applikation auf („*Les instructions du programme d'application, dans l'invention, doivent être interprétées, au moyen d'un programme d'interprétation contenu […] programme P représentant l'application*“, S. 11, Zn. 15-24 und Fig. 1 / **Merkmal M2**). Aus Druckschrift NK7 ist zudem bekannt, dass die Applikation durch eine Programmierungsumgebung generiert wird (Fig. 2, Schritt 14 i.V.m. Schritt 15 mit Beschreibung, S. 12, Zn. 4-6 und S. 13, Zn. 1-5 / **Merkmal M2.1**), welche einen Compiler umfasst, der Anwendungs-Quellprogramme in Quellcodeform einer höheren Programmiersprache in eine kompilierte Form kompiliert („*Le programme compilateur qui permet la production du programme d'application sous sa forme intermédiaire […]*“, S. 13, Zn. 2-5 / **Merkmal M2.1.1**).

Die in Druckschrift NK7 offenbarte Programmierungsumgebung unterscheidet sich von der des erteilten Anspruchs darin, dass keine minimierte Form des kompilierten interpretierbaren Zwischencodes mittels eines Konverters erzeugt wird (Merkmal M2.1.2 fehlt).

Wie einleitend zu den Verfahrensmerkmalen zum Erzeugen des in den Mikrocontroller geladenen Anwendungscodes dargelegt (vgl. Merkmale M2.1 bis M2.1.2), können diese Merkmale den beanspruchten Mikrocontroller höchstens dann charakterisieren, wenn die Verfahrensschritte – hier die zweistufige Aufbereitung des Anwendungsprogramms durch Compiler und Konverter in der Programmierungsumgebung – ihren eindeutig erkennbaren Niederschlag im erzeugten Code des Anwendungsprogramms finden und dieser Code dadurch von einem auf andere Weise erzeugten, für die Übersetzung im Mikrocontroller geeigneten Code der Anwendung unterscheidbar ist, bzw. sich der beanspruchte Mikrocontroller selbst durch den geladenen Anwendungscode von einem aus dem Stand der Technik bekannten Mikrocontroller mit auf andere Weise erzeugtem Anwendungscode unterscheidet.

Ein solcher Anwendungscode, der gegenüber Druckschrift NK7 zu besonderen Merkmalen oder Eigenschaften des Mikrocontrollers, also zu einem von Druckschrift NK7 unterscheidbaren, mit Interpreter und Applikation geladenen Mikrocontroller führt, ergibt sich für den beanspruchten Mikrocontroller jedoch nicht aus den Verfahrensschritten der Merkmale M2 bis M2.1.2. Vielmehr ist die einzige Anforderung, die der gemäß Anspruch 1 erzeugte Code des Anwendungsprogramms erfüllen muss, dessen Eignung, vom Interpreter ausgeführt werden zu können (vgl. Merkmal M2.1.2: „[…] suitable for interpretation by the interpreter“). Ein entsprechend geeigneter Anwendungscode wird auch gemäß Druckschrift NK7 mittels eines Compilers direkt erzeugt, so dass aus dieser Eignung des jeweils erzeugten Codes kein Unterschied zum Code gemäß Druckschrift NK7 folgt. Darüber hinausgehende Merkmale, wie die von der Beklagten angesprochenen Sicherheitsmerkmale

des Codes sind nicht Gegenstand des Anspruchs 1 gemäß Hauptantrag. Denn der beanspruchten minimierten Form des kompilierten Anwendungscodes gemäß Anspruch 1 sind keine inhaltlichen Merkmale oder besondere Eigenschaften zu entnehmen, die ihn von einem auf andere Weise erzeugten, für den Interpreter geeigneten Code erkennbar unterscheiden würden (**Merkmal M2.1.2**).

Das außerhalb des beanspruchten Mikrocontrollers in der Programmierungsumgebung ablaufende zweistufige Verfahren unter Verwendung eines Compilers in Verbindung mit einem Konverter zum Erzeugen einer minimierten Form der Anwendung ist daher nicht geeignet, den Mikrocontroller gemäß Anspruch 1 nach Hauptantrag zu charakterisieren und gegenüber der Lehre der Druckschrift NK7 abzugrenzen. Die bestehenden Unterschiede in den (Verfahrens-)Merkmalen zur Erzeugung des Anwendungscodes sind dabei nicht geeignet, eine erfinderische Tätigkeit zu begründen.

Die Beklagte hat darauf verwiesen, dass im Stand der Technik gemäß Druckschrift NK7 mit einem reduzierten Sprachumfang bzw. Befehlssatz gearbeitet würde (vgl. Druckschrift NK7, S. 14, Zn. 21 ff und S. 8, Zn. 7-9) und hieraus unter anderem eine Unterscheidbarkeit des generierten Codes folge, der in den Mikrocontroller geladen wird. Ungeachtet der Tatsache, dass Druckschrift NK7 von einer Erstellung des Quellcodes ohne Einschränkungen ausgeht („*Premièrement, dans une phase 14 on écrit le programme de l'application comme si les problèmes cités relatifs aux cartes à micro-circuit étaient inexisitants.*“, vgl. S. 12, Zn. 4-6), ist anhand des in den Mikrocontroller geladenen Anwendungscodes nicht zu unterscheiden, ob bestimmte Befehle per Definition nicht verwendet, nicht benötigt, durch einen zusätzlichen Verarbeitungsschritt entfernt oder durch andere Befehle ersetzt wurden. Aus diesem Grund sind die beanspruchten Verfahrensschritte zur Codeerzeugung nicht geeignet, das geladene Anwendungsprogramm und damit den Mikrocontroller eindeutig zu charakterisieren.

Der Anspruch 1 nach Hauptantrag ist damit nicht patentfähig.

Es kann daher auch dahinstehen, ob der vorgesehene Zwischenschritt zum Konvertieren des kompilierten Quellcodes (Merkmal M2.1.1 i. V. m. Merkmal M2.1.2), der die Verarbeitung des Anwendungsprogramms vom ausschließlichen Kompilieren eines Zwischencodes nach Druckschrift NK7 unterscheidet (vgl. Fig. 2, Schritt 15, mit Beschreibung, S. 13, Zn. 1 - 18), einen technischen Beitrag zur Lösung der zu Grunde liegenden Problemstellung liefert (vgl. BGH, Urt. v. 18. Dezember 2012 – X ZR 2/12, GRUR 2013, 275, Leitsätze - Routenplanung). Die Aufteilung der Programmierungsumgebung in Compiler und Konverter führt entgegen den Ausführungen der Beklagten neben der fehlenden Unterscheidbarkeit der geladenen Applikation weder zu erkennbaren technischen Besonderheiten des Codes (der gemäß Anspruch 1 nur zur Ausführung durch den Interpreter geeignet sein muss), noch sind Randbedingungen oder Zusammenhänge im erteilten Anspruch 1 des Streitpatents erkennbar, die eine solche Aufteilung der Codeerzeugung in Compiler und Konverter technisch erforderlich machen.

2. Hilfsantrag 1

Der Gegenstand des Anspruchs 1 nach Hilfsantrag 1 beruht gegenüber dem Stand der Technik gemäß Druckschrift NK7 ebenfalls nicht auf einer erfinderischen Tätigkeit.

Anspruch 1 gemäß Hilfsantrag 1 unterscheidet sich von Anspruch 1 des Hauptantrags darin, dass das Ergebnis des Kompilierungsschrittes in einem standardmäßigen Java-Klassendateiformat vorliegt („standard Java class file format“, Merkmal M2.1.3 i. V. m. M2.1.1) und der Konverter dieses Datenformat verarbeitet, um zu einer für den Interpreter interpretierbaren Form zu gelangen („[…] in a form suitable for interpretation by the interpreter“, Merkmal M2.1.4 i. V. m. M2.1.2). Zu den gegenüber dem Hauptantrag unveränderten Merkmalen sei auf die Ausführungen zum Hauptantrag verwiesen, die für die

Merkmale M1 bis M2.1.2 für den Anspruch 1 des Hilfsantrags 1 in gleicher Weise gelten (die Merkmale M1 bis M2.1.1 sind aus Druckschrift NK7 bekannt; das zusammen mit Merkmal M2.1.1 außerhalb des beanspruchten Mikrocontrollers liegende Merkmal M2.1.2 ist nicht geeignet, den beanspruchten Mikrocontroller von einem Mikrocontroller abzugrenzen, wie er aus Druckschrift NK7 bekannt ist).

Damit unterscheidet sich Anspruch 1 gemäß Hilfsantrag 1 vom Gegenstand der Druckschrift NK7 im Format des Kompilierungsergebnisses als standardmäßiges Java-Klassendateiformat („standard Java class file format“), da Druckschrift NK7 für keine der dort genannten Programmiersprachen Angaben zu dem vom Compiler erzeugten Dateiformat macht (**Merkmal M2.1.3 teilweise**). Zudem ist Druckschrift NK7 kein Konverter zu entnehmen, der geeignet ist, aus dem vom Compiler erzeugten Dateiformat eine vom Interpreter interpretierbare Ausgabe zu erzeugen (Merkmal M2.1.4 fehlt).

Der Fachmann wird durch Druckschrift NK7 nicht zwingend auf die Wahl einer bestimmten Programmiersprache (im Ausführungsbeispiel: C) festgelegt. Vielmehr verweist Druckschrift NK7 auf eine Reihe weiterer, prinzipiell geeigneter Programmiersprachen (vgl. S. 2, Z. 32 bis S. 3, Z. 1 und Patentanspruch 2). Insbesondere sieht Druckschrift NK7 ausdrücklich die Verwendung eines Interpreters und die Erzeugung eines geeigneten interpretierbaren, kompilierten Anwendungscodes vor – mithin ist die Lehre der Druckschrift NK7 weder auf eine beliebige Programmiersprache, noch auf eine spezielle hardwareabhängige Codeerzeugung gerichtet (vgl. bspw. S. 11, Zn. 15-25), sondern sie gibt eine Vorgehensweise vor, wie sie beispielsweise für Java typisch ist.
Aus dem in Druckschrift NK7 beschriebenen Interpreter-Konzept und den Verweisen auf verschiedene geeignete Hochsprachen folgt eine Veranlassung für den Fachmann, ausgehend von der zugrunde liegenden Problemstellung, auch andere geeignete Sprachen aufgrund der diesen innewohnenden Vorteile in Erwägung zu ziehen. Hierbei stellte Java gerade aufgrund der großen

Resonanz in Fachkreisen zum Anmeldezeitpunkt des Streitpatents einen hierfür naheliegenden Kandidaten dar.
Die Ressourcenangaben hinsichtlich Mikrocontrollern bzw. Chipkarten in der sieben Jahre vor dem Streitpatent angemeldeten Druckschrift NK7 stellten zum Anmeldezeitpunkt des Streitpatents keinen Anlass dar, Java nicht als höhere Programmiersprache in Erwägung zu ziehen. Denn es wurde in diesem Zeitraum mit Java nicht nur eine neue Programmiersprache entwickelt, sondern insbesondere auch die in Mikrocontrollern verwendeten Technologien weiterentwickelt. Daher hätte der auf dem Gebiet der Mikrocontroller oder Chipkarten arbeitende Fachmann zum Anmeldezeitpunkt des Streitpatents im Jahr 1997 nicht die Ressourcenanforderungen der Druckschrift NK7 ungeprüft übernommen.
Die Wahl von Java als Interpreter-Sprache bzw. die Verwendung eines entsprechenden Java Dateiformats für den kompilierten Anwendungscode lag daher zum Anmeldezeitpunkt des Streitpatents im Rahmen der vom Fachmann in naheliegender Weise in Erwägung zu ziehenden Optionen, um eine höhere Programmiersprache auf Mikrocontrollern zu verwenden und damit von Vorteilen dieser bspw. plattformunabhängigen Programmiersprache zu profitieren (**Merkmal M2.1.3 Rest**).

Im Hinblick auf das gemäß Hilfsantrag 1 gewählte, vom Compiler erzeugte Zwischenformat („standard Java class file format“) ergibt sich für die Anforderungen an den Code des Anwendungsprogramms nichts anderes als in Anspruch 1 gemäß Hauptantrag. Auch Anspruch 1 gemäß Hilfsantrag 1 fordert für den in den Mikrocontroller geladenen Code nur eine Eignung zur Übersetzung/Interpretierung durch den Interpreter. Die Verwendung von standardmäßigen Java-Klassendateien als kompiliertes Zwischenformat und damit implizit eines Java Compilers im Laufe der Erzeugung der zu ladenden Applikation (Merkmal M2.1.3) kennzeichnet zwar diesen Verfahrensablauf, nicht aber das damit erzeugte Ergebnis. Denn dem Ergebnis (d. h. dem in den Mikrocontroller geladenen Anwendungscode) gemäß Anspruch 1 ist weder entnehmbar, welche besonderen Merkmale dieses Codes aus der Wahl des Zwischencode-

Formats („standard Java class file format“) folgen, noch welche charakteristischen Eigenschaften des Codes sich aus der zweistufigen Verarbeitung mit Compiler und Konverter ergeben.

Weder aus der zweistufigen Verarbeitung des Codes bei der zweistufigen Erzeugung einer minimierten Form des Anwendungsprogramms mit Compiler und Konverter (Merkmal M2.1.4 i. V. m M2.1.3), noch aus der Vorgabe des kompilierten Zwischenformats („standard Java class file format“) folgen daher Merkmale oder Eigenschaften der in den Mikrocontroller geladenen Applikation, die geeignet wären, den Mikrocontroller selbst zu charakterisieren und diesen von einem Mikrocontroller mit bspw. direkt mittels angepasstem Compiler entsprechend Druckschrift NK7 erzeugtem Anwendungscode abzugrenzen (**Merkmal M2.1.4**).

Der von der Beklagten vertretenen Auffassung, der Fachmann würde ausgehend von Druckschrift NK7 nicht Java als höhere Programmiersprache verwenden, kann nicht gefolgt werden. Selbst wenn Bedenken wegen des Speicherbedarfs einer Java Virtual Machine zur Interpretierung des Zwischencodes bestünden, zeigt Druckschrift NK7 allgemein, dass eine Virtual Machine für eine Hochsprache zur Verwendung auf einem ressourcenbegrenzten Mikrocontroller realisiert werden kann (vgl. Druckschrift NK7, S. 6, Zn. 9-16 i.V.m. S. 8, Z. 20 bis S. 9, Z. 1). Zudem verweist Druckschrift NK7 auf den Zusammenhang zwischen beschränkten Speicherressourcen und dem Umfang des Befehlssatzes (vgl. S. 14-15, seitenübergreifender Absatz). Der Speicherbedarf bestimmter Virtual Machines bzw. Interpreter für Hochsprachen wäre angesichts des genannten Standes der Technik vielmehr Veranlassung für den Fachmann, Maßnahmen zur Anpassung für bestimmte, gewünschte Merkmale dieser Hochsprachen an die durch das Zielsystem vorgegebenen technischen Randbedingungen in Erwägung zu ziehen.

Die von der Beklagten angesprochenen Sicherheitsmerkmale von Java finden auch im Anspruch 1 gemäß Hilfsantrag 1 keinen Niederschlag und ergeben

sich auch nicht zwangsläufig aus dem Java-Zwischenformat („standard Java class file format“), da Anspruch 1 weder auf Merkmale dieses Zwischenformats, noch auf die Auswirkungen der Minimierung auf den kompilierten Code durch den Konverter eingeht. Wie beispielsweise aus der Beschreibung des Streitpatents ersichtlich ist, kann vielmehr eine Minimierung des kompilierten Codes gerade auch zu Lasten der angesprochenen Sicherheitsmerkmale gehen (vgl. deutsche Fassung des Streitpatents, DE 697 14 752 T2, S. 22, Z. 22 bis S. 23, Z. 2). Mögliche besondere Eigenschaften der Programmiersprache Java können daher eine erfinderische Tätigkeit gegenüber Druckschrift NK7 nicht begründen, da solche Besonderheiten der Programmiersprache gemäß Anspruch 1 keinen erkennbaren Niederschlag in der in den Mikrocontroller geladenen Applikation finden und damit dem Mikrocontroller selbst nicht anzusehen sind.. Auch aus Merkmal M2.1.4 folgt als einzige Anforderung an diesen Code die Interpretierbarkeit durch den Interpreter.

3. Hilfsantrag 2

Der Gegenstand des Anspruchs 1 nach Hilfsantrag 2 beruht gegenüber dem Stand der Technik gemäß Druckschrift NK7 ebenfalls nicht auf einer erfinderischen Tätigkeit.

Anspruch 1 des Hilfsantrags 2 unterscheidet sich von Anspruch 1 des Hilfsantrags 1 im Merkmal M2.1.4*, das nunmehr auf das Erzeugen einer Ausgabe in einem zum Interpretieren durch den Interpreter geeigneten Klassendatei-Format („class file format“) an Stelle einer allgemeinen, zum Interpretieren geeigneten minimierten Form der kompilierten Anwendung gerichtet ist. Zu den gegenüber dem Hilfsantrag 1 unveränderten Merkmalen M1 bis M2.1.3 des Anspruchs 1 sei auf die vorstehenden Ausführungen zum Hilfsantrag 1 verwiesen, die für die Merkmale in Anspruch 1 des Hilfsantrags 2 in gleicher Weise gelten (die Merkmale M1 bis M2.1.1 sind aus Druckschrift NK7 bekannt; die Merkmale M2.1.2 und M2.1.3 sind wiederum nicht geeignet, den beanspruch-

ten Mikrocontroller von einem Mikrocontroller abzugrenzen, wie er aus Druckschrift NK7 bekannt ist).

Merkmal M2.1.4* sieht nunmehr vor, dass der Konverter als Ergebnis das Anwendungsprogramm in einem bestimmten Format – einem Klassendateiformat („class file format“) – liefert. Die weiteren Anforderungen an diesen Anwendungscode gemäß Merkmal M2.1.4* i. V. m. M2.1.2 entsprechen dabei dem Anspruch 1 des Hilfsantrags 1. Die neben dem hinzugefügten Dateiformat einzige weitere Bedingung ist daher die Eignung, vom Interpreter interpretiert werden zu können. Somit ergeben sich auch aus Anspruch 1 gemäß Hilfsantrag 2 keine Merkmale des Anwendungscodes, welche die zweistufige Erzeugung oder inhaltliche Besonderheiten des Quell- oder Zwischencodes in dem in den Mikrocontroller geladenen Anwendungscode widerspiegeln würden. Ganz allgemein gilt dabei, dass auch der aus Druckschrift NK7 bekannte Compiler als Ausgabe an den Interpreter den Code des Anwendungsprogramms in einem definierten Format liefert, wobei der Code zum Übersetzen/Interpretieren durch den Interpreter geeignet ist (**Merkmal M2.1.4* teilweise**).

Damit liegt die einzige Besonderheit des gemäß Anspruchs 1 generierten Anwendungscodes gegenüber Druckschrift NK7 in der Festlegung des erzeugten Dateiformats als Klassendateiformat („class file format“).

Bei Anwendungen in klassenbasierten Programmiersprachen wie Java liegt in der Regel der interpretierbare Byte-Code als Klassendatei vor, d. h. in einem Klassendateiformat. Somit stellt aber das Ergebnisformat („class file format“) nach Merkmal M2.1.4* keine Besonderheit der zweistufigen Code-Erzeugung mit Compiler und Konverter des Streitpatents dar, sondern beschreibt unabhängig davon für den Fall klassenbasierter Programmiersprachen das übliche Dateiformat des als Klassendateien vorliegenden interpretierbaren Codes. Da gemäß Druckschrift NK7 ein Anwendungscode als kompilierter, interpretierbarer Code für einen Interpreter erzeugt wird, folgt daraus für den Fachmann im-

plizit, dass der in den Mikrocontroller geladene interpretierbare Anwendungscode bei einer klassenbasierten Programmiersprache – beispielsweise bei der naheliegenden Verwendung von Java – auch in einem Klassendatei-Format vorliegt (**Merkmal M2.1.4* Rest**).

Somit ist weder das Vorsehen eines Konverters zum Minimieren eines zuvor mittels Compiler erzeugten Klassendateiformats geeignet, den Mikrocontroller in Abgrenzung zu einem direkt erzeugten Anwendungscode entsprechend Druckschrift NK7 zu charakterisieren, noch ist das Dateiformat („class file format“), in dem das Anwendungsprogramm als Ausgabe an den Interpreter erzeugt wird, geeignet, gegenüber einem Mikrocontroller gemäß Druckschrift NK7 eine erfinderische Tätigkeit zu begründen. Auch der Anspruch 1 nach Hilfsantrag 2 ist somit nicht patentfähig.

4. Hilfsantrag 3

Der Gegenstand des Anspruchs 1 nach Hilfsantrag 3 beruht gegenüber dem Stand der Technik gemäß Druckschrift NK7 ebenfalls nicht auf einer erfinderischen Tätigkeit.

Anspruch 1 des Hilfsantrags 3 unterscheidet sich von Anspruch 1 des Hilfsantrags 1 in den Merkmalen Merkmal M2.1.3* und M2.1.4**, die auf das Erzeugen einer Ausgabe in einer einzigen zum Interpretieren durch den Interpreter geeigneten Klassendatei („single class file“) aus einer Mehrzahl kompilierter Java Klassendateien („plurality of Java class files“) gerichtet ist. Zu den gegenüber dem Hilfsantrag 1 unveränderten Merkmalen sei auf die vorstehenden Ausführungen zum Hilfsantrag 1 verwiesen, die für die Merkmale M1 bis M2.1.2 des Anspruchs 1 des Hilfsantrags 3 in gleicher Weise gelten (die Merkmale M1 bis M2.1.1 sind aus Druckschrift NK7 bekannt; das Merkmal M2.1.2 ist nicht geeignet, den beanspruchten Mikrocontroller von einem Mikrocontroller abzugrenzen, wie er aus Druckschrift NK7 bekannt ist).

Nach Anspruch 1 des Hilfsantrags 3 ist der in den Mikrocontroller geladene Anwendungscode dadurch charakterisiert, dass der Code durch den Interpreter interpretierbar ist (Merkmale M2.1.2 i. V. m. M2.1.4**) und als einzelne Ergebnisdatei („single class file“, Merkmal M2.1.4**) vorliegt. Auch bei der einzelnen Ergebnisdatei ist nicht erkennbar, auf welchem Weg sie erzeugt wurde, d. h. ob sie durch einen Compiler, Konverter, einer Kombination daraus oder auf anderem Weg – z. B. durch Eingriffe eines Nutzers – erzeugt wurde. Insbesondere ist anhand der Ergebnisdatei nicht erkennbar, ob der Quellcode bzw. Zwischencode in mehreren Dateien vorlag.
Zwischenschritte der Codeerzeugung, wie die Anzahl von kompilierten und mittels eines Konverters aufbereitenten Dateien finden keinen eindeutig identifizierbaren Niederschlag im Anwendungscode, der daher in Bezug auf das Verfahren zur Codeerzeugung den Mikrocontroller nicht charakterisieren kann (**Merkmal M2.1.3***; **Merkmal M2.1.4** teilweise** bzgl. des Dateiformats und der Eignung zur Ausführung durch den Interpreter).

Das Vorliegen einer einzelnen Klassendatei ist nicht zwangsläufig das Ergebnis eines Minimieren im Sinne eines verringerten Code-Umfangs des Anwendungsprogramms, da Anspruch 1 mit dem Verweis auf die Umsetzung von mehreren Klassendateien in eine Klassendatei durch den Konverter keine Angabe zum Umfang des Codes, sondern nur zur Form des ausgegebenen Anwendungscodes macht. Allein die Wahl des Dateiformats stellt aber allenfalls eine Maßnahme der Datenverarbeitung im Belieben des Programmierers dar, die ohne weitergehende technische Zusammenhänge, insbesondere im Hinblick auf die Charakterisierung und Funktion des Mikrocontrollers keinen Beitrag zur Lösung des zu Grunde liegenden technischen Problems im Sinne der ständigen BGH Rechtsprechung (vgl. bspw. BGH Entscheidung Routenplanung, a. a. O.) liefert (**Merkmal M2.1.4** Rest**).

Auch die Angaben zur jeweiligen Anzahl der Klassendateien in den einzelnen Verfahrensschritten zur Erzeugung des in den Mikrocontroller geladenen An-

wendungscodes nach Anspruchs 1 des Hilfsantrags 3 ist daher nicht geeignet, gegenüber Druckschrift NK7 eine erfinderische Tätigkeit zu begründen.

5. Hilfsantrag 4

Der Gegenstand des Anspruchs 1 nach Hilfsantrag 4 beruht gegenüber dem Stand der Technik gemäß Druckschrift NK7 ebenfalls nicht auf einer erfinderischen Tätigkeit.

Anspruch 1 des Hilfsantrags 4 stellt eine Kombination der jeweiligen Ansprüche 1 der Hilfsanträge 2 und 3 dar. Zu den gegenüber Hilfsantrag 1 bzw. Hilfsantrag 3 unveränderten Merkmalen sei auf die vorstehenden Ausführungen verwiesen, die für die Merkmale M1 bis M2.1.3* des Anspruchs 1 im Hilfsantrag 4 in gleicher Weise gelten (die Merkmale M1 bis M2.1.1 sind aus Druckschrift NK7 bekannt; die Merkmale M2.1.2 und M2.1.3* - vgl. jeweils Hilfsantrag 3 - sind wiederum nicht geeignet, den beanspruchten Mikrocontroller von einem Mikrocontroller technisch abzugrenzen, wie er aus Druckschrift NK7 bekannt ist).

Merkmal M2.1.4*** umfasst die Beschreibung des vom Konverter erzeugten Anwendungscodes in einem Klassendateiformat („class file format“) entsprechend Hilfsantrag 2 in Verbindung mit dem Vorliegen des Codes als einzelne Datei („single class file“ entsprechend Hilfsantrag 3).
Das Vorliegen des interpretierbaren Anwendungscodes in einem definierten Klassendateiformat ist typisch für ein interpretierbares, kompiliertes Format des Anwendungscodes einer klassenbasierten Programmiersprache wie Java. Allein diese Formatvorgabe ist daher nicht geeignet, eine erfinderische Tätigkeit in Abgrenzung zum Mikrocontroller mit geladenem interpretierbaren Anwendungscode gemäß Druckschrift NK7 zu begründen (vgl. auch vorstehende Ausführungen zu Merkmal M2.1.4* des Hilfsantrags 2).
Aus der verringerten Zahl der Dateien des in den Mikrocontroller zu ladenden Anwendungscodes folgt nicht zwangsläufig eine Minimierung im Sinne einer

verringerten Datenmenge. Das Vorliegen einer einzigen Datei („a single class file“) gemäß Anspruch 1 nach Hilfsantrag 4 beschreibt damit eine Maßnahme der Datenverarbeitung (vgl. vorstehende Ausführungen zu Merkmal M2.1.4** des Hilfsantrags 3), die ebenfalls nicht geeignet ist, eine erfinderische Tätigkeit in Abgrenzung zum Mikrocontroller mit geladenem interpretierbaren Anwendungscode zu begründen, wie er aus Druckschrift NK7 bekannt ist (**Merkmal M2.1.4*****).

6. Hilfsantrag 5

Der Gegenstand des Anspruchs 1 nach Hilfsantrag 5 beruht gegenüber dem Stand der Technik gemäß Druckschrift NK7 ebenfalls nicht auf einer erfinderischen Tätigkeit.

Anspruch 1 des Hilfsantrags 5 unterscheidet sich vom Gegenstand des Anspruchs 1 des Hilfsantrags 4 allein darin, dass der Quellcode des Anwendungsprogramms in der Programmiersprache Java vorliegen soll (Merkmal M2.1.1*). Für die **Merkmale M1 bis M2.1** gelten die vorstehenden Ausführungen zu Anspruch 1 des Hilfsantrags 4 in gleicher Weise (die genannten Merkmale sind aus Druckschrift NK7 bekannt).

Das Vorsehen von Java als Alternative zu der in Druckschrift NK7 verwendeten Programmiersprache C war dem Fachmann zum Anmeldezeitpunkt durch Hinweise auf weitere, auf der Verwendung einer höheren Programmiersprache mit einem auf dem Mikrocontroller laufenden Interpreter nahegelegt, wie dies zu Anspruch 1 des Hilfsantrags 1 dargelegt wurde (**Merkmal M2.1.1***).

Das Format des Quellcodes ist aber für den in den Mikrocontroller geladenen Anwendungscode nicht weiter von Bedeutung, da der erzeugte Code gemäß Anspruch 1 zur Interpretierung durch den Interpreter geeignet sein muss (Merkmal M2.1.2 i. V. m. M2.1.4***), aus einer Datei bestehen (Merkmal M2.1.4***) und in einem Klassendatei-Format vorliegen soll (Merkmal

M2.1.4***). Wie vorstehend zu Anspruch 1 des Hilfsantrags 4 dargelegt, geben die beanspruchten Verfahrensschritte zur Codeerzeugung aber keinen Anhaltspunkt dafür, dass über die genannten Anforderungen an den ausgegebenen Code hinaus die Verfahrensschritte selbst (bspw. das vom Compiler erzeugte Zwischenergebnis) erkennbar und unterscheidbar einen Niederschlag in dem in den Mikrocontroller geladenen Anwendungscode findet.
Die oben genannten Merkmale M2.1.2 bis M2.1.4*** sind daher ebenfalls nicht geeignet, eine erfinderische Tätigkeit gegenüber dem Mikrocontroller mit geladenem interpretierbaren Anwendungscode zu begründen, wie er aus Druckschrift NK7 bekannt ist (**Merkmale M2.1.2 bis M2.1.4*****).

7. Darüber hinaus geht der jeweilige Gegenstand des Anspruchs 1 des Streitpatents und der Hilfsanträge 1 bis 5 über den Inhalt der Anmeldung in der ursprünglich eingereichten Fassung hinaus (Art. II § 6 Abs. 1 Nr. 3 IntPatÜG i. V. m. Art. 138 Abs. 1 Buchst. c EPÜ). Die verteidigten Fassungen des Anspruchs 1 sind jeweils nicht zulässig.

   Der Gegenstand des erteilten Anspruchs 1 geht über den Inhalt der ursprünglich als PCT-Anmeldung eingereichten und als WO 98/19237 A1 veröffentlichten Unterlagen hinaus, da ein zweistufiges Erzeugen der Applikation aus einem in einer beliebigen Hochsprache vorliegenden Anwendungs-Quellprogramm unter Verwendung eines Compilers zum Kompilieren des Quellprogramms und eines Konverters zur Nachverarbeitung des kompilierten Codes (Merkmale M2.1, M2.1.1, M2.1.2), nicht zu entnehmen ist.
   Zwar sieht der allgemeine Teil der ursprünglichen Beschreibung allgemein die Verwendung von Hochsprachen für die in den Mikrocontroller geladenen Applikationen vor (vgl. ursprüngliche Beschreibung, veröffentlicht als WO 98/19237 A1, S. 7, Zn. 15-19 und S. 11, Z. 33 – S. 12, Z. 1). Im allgemeinen Teil der Beschreibung ist daneben jedoch nur eine Umwandlung einer Applikation in ein zweites Format unabhängig von einer Programmierungsumgebung beschrieben (ebd., S. 7, Zn. 20-24 i. V. m. S. 10, Zn. 20-24). Eine zweistufige Verarbeitung mit Compiler und Konverter findet sich auch nicht in den

ursprünglich eingereichten Ansprüchen wieder. Die Verfahrensansprüche sprechen zwar allgemein von einem Konvertieren von Applikationsprogrammen. Das jeweilige Verfahren ist aber auf Chipkarten gerichtet und nimmt nur Bezug auf zwei Anwendungen, die unabhängig von einer Programmierungsumgebung mit Compiler und Konverter beschrieben sind (vgl. ursprüngliche Ansprüche 67, 79, 87-89). Diese ursprünglichen Ansprüche treffen keine weitere Aussage über die Maßnahmen zum Erzeugen dieser Dateien.

Eine Programmierungsumgebung mit Compiler und Konverter gemäß Anspruch 1 ist in den ursprünglichen Unterlagen dagegen nur den Anwendungsbeispielen entnehmbar. Die dort beschriebene Vorgehensweise beruht auf einem spezifischen, auf einem Java-Klassendateiformat basierenden Karten-Klassendateiformat und sieht neben der Verwendung von Java als Programmiersprache die Verwendung eines Karten-Klassendatei-Konverters (ebd., Fig. 3, S. 16, Zn. 23-33) und einer Card Java Virtual Machine („Card JVM“) als Interpreter vor (vgl. ebd., S. 16, Zn. 1-4).

Die in Anspruch 1 beanspruchte Merkmalskombination aus allgemeiner Hochsprache und Programmierungsumgebung mit Compiler und allgemeinem Konverters folgt damit nicht direkt und eindeutig aus der ursprünglichen Beschreibung des Anmeldungsgegenstands. Die zweistufige Vorgehensweise zur Codeerzeugung gemäß den Ausführungsbeispielen lässt sich dabei auch nicht im Sinne des Anspruchs 1 auf beliebige Programmiersprachen und eine beliebige kompilierte Form verallgemeinern (vgl. Merkmal M2.1.1), da sich die Ausführungsbeispiele auf die Verwendung von Java als einer klassenbasierten Programmiersprache stützen. Denn die ursprüngliche Beschreibung sieht eine zweistufige Verwendung von Compiler und Konverter sowie eine Minimierung der kompilierten Anwendung allein im Zusammenhang mit einem spezifischen Karten-Klassendateiformat vor, dessen Erzeugung mit einem dafür vorgesehenen Karten-Klassendatei-Konverter verbunden ist und für dessen Verwendung wiederum nur eine daran angepasste Card Java Virtual Machine („Card JVM“) als Interpreter offenbart ist.

Die Beklagte hat zur ursprünglichen Offenbarung der Merkmalskombination des Anspruchs 1 darauf hingewiesen, dass Ausführungen in der Anmeldung zu anderen Ausführungsformen wie bspw. einer Chipkarte („integrated circuit card“) auch für den beanspruchten Mikrocontroller gelten würden (vgl. PCT-Veröffentlichung, S. 37, Zn. 5-13 und S. 15, Zn. 7-13 und 34-35). Eine Konvertierung des Anwendungscodes sei daher auch in Bezug auf einen Mikrocontroller offenbart. Den in diesem Zusammenhang zitierten ursprünglichen Ansprüche 1 und 67 und 68 beziehen sich aber nicht auf eine Programmierungsumgebung, sondern beschreiben nur allgemein ein konvertieren von Anweisungen („instructions“) einer ersten in Anweisungen einer zweite Applikation. Eine Minimierung einer kompilierten Form des in einer beliebigen Hochsprache vorliegenden Quellcodes lässt sich hieraus nicht herleiten. Weitere zur Verwendung von Programmiersprachen außer Java genannte Textstellen (ebd., S. 37, Z. 19 bis S. 38, Z. 7; S. 11, Z. 33 bis S. 12, Z. 4) sehen wiederum keine allgemeine Hochsprache im Sinne des Anspruchs 1, sondern nur klassenbasierte Hochsprachen, d.h. Sprachen, aus deren Quellcode Klassendateien erzeugt werden können.

Die im erteilten Anspruch 1 (Hauptantrag) beanspruchte Merkmalskombination eines allgemeinen Konverters (Merkmal M2.1.2) und eines allgemeinen Interpreters (Merkmal M1.2 i.V.m. M2.1.2) in Verbindung mit der Minimierung einer kompilierten Form des in einer beliebigen Hochsprache vorliegenden Quellcodes stellt daher eine Verallgemeinerung dar, die über den ursprünglich offenbarten Anmeldungsgegenstand hinausgeht.

Der Gegenstand des Anspruchs 1 gemäß Hilfsantrag 1 geht ebenfalls über den Inhalt der Anmeldung hinaus, wie als PCT-Anmeldung ursprünglich eingereicht worden ist. Im Unterschied zum Hauptantrag sieht der Anspruch 1 gemäß Hilfsantrag 1 vor, dass es sich bei der kompilierten Form des Anwendungsquellcodes um ein standardmäßiges Java-Klassendateiformat („standard Java class file format“) handelt. Ein Konverter, der eine kompilierte Form des Anwendungs-Quellprogramms in diesem Klassendateiformat als Ein-

gangsgröße hat und eine allgemeine minimierte Form dieser kompilierten Anwendung zur Ausführung durch einen allgemeinen Interpreter erzeugt, ist der ursprünglichen Anmeldung jedoch nicht zu entnehmen. Diese sieht eine zweistufige Verwendung von Compiler und Konverter sowie eine Minimierung der kompilierten Anwendung vielmehr allein im Zusammenhang mit einem spezifischen Karten-Klassendateiformat vor, dessen Erzeugung mit einem dafür vorgesehenen Karten-Klassendatei-Konverter verbunden ist und für dessen Verwendung wiederum nur eine dafür geeignete Java Card Class Virtual Machine als Interpreter offenbart ist. Auch für Anspruch 1 gemäß Hilfsantrag 1 gilt daher, dass sich die beanspruchte Merkmalskombination aus zweistufiger Programmierungsumgebung und allgemeinem Konverter und Interpreter nicht aus der ursprünglichen Beschreibung des Anmeldungsgegenstands ergibt.

Der Gegenstand der jeweiligen Ansprüche 1 der Hilfsanträge 2 bis 5 geht ebenfalls über den Inhalt der Anmeldung hinaus, wie sie als PCT-Anmeldung ursprünglich eingereicht worden ist.
In den Ansprüchen 1 der Hilfsanträge 2 bis 5 ist jeweils ein Konverter allgemein beansprucht, der den in einem standardmäßigen Java-Klassendateiformat („standard Java class file format“, Merkmal M2.1.3 bzw. M2.1.3*) vorliegenden kompilierten Anwendungscode nachbearbeitet und ein Ergebnis in einem Klassendateiformat erzeugt („class file format“; Hilfsantrag 2, Merkmal M2.1.4*) oder als Ergebnis eine einzelne Klassendatei erzeugt („single class file“, Hilfsantrag 3, Anspruch 1, Merkmal M2.1.4**), oder beides (Hilfsanträge 4 und 5, Merkmal M2.1.4***). Weiterhin ist ein Interpreter allgemein beansprucht, der das so erzeugte Ergebnis interpretiert bzw. übersetzt. In Anspruch 1 der Hilfsanträge 2 bis 5 wird damit das Format der in den Mikrocontroller geladenen Applikation, d. h. das Ergebnis des Konvertierungsschrittes, näher als Klassendateiformat bzw. einzelne Klassendatei definiert. Solche Angaben zum Dateiformat bzw. der Anzahl der Klassendateien selbst sind dem allgemeinen Teil der ursprünglichen Beschreibung oder den ursprünglichen Ansprüchen in Verbindung mit einem Konverter nicht zu entnehmen, da die zweistufige Verarbeitung des Anwendungs-Quellprogramms mit Compiler und

Konverter nur Gegenstand der Ausführungsbeispiele der Anmeldung ist. Der jeweilige Anspruch 1 gibt jedoch auch keines der Ausführungsbeispiele vollständig wieder, da sich diese nur auf das spezifische Karten-Klassendateiformat gemäß Anhang A der Anmeldung bzw. auf einen dafür geeigneten Karten-Klassendatei-Konverter beziehen (vgl. PCT-Veröffentlichung, Fig. 3, S. 16, Zn. 23-33). Die Ausführungsbeispiele sehen zur Übersetzung der mittels Compiler und Konverter erzeugten Klassendateien in einem speziellen Karten-Klassendateiformats zudem eine entsprechend angepasste Ausgestaltung des Interpreters als eine Card Java Virtual Machine („Card JVM“) vor (ebd., S. 16, Zn. 1-4).

Die Erzeugung von nur irgendwie minimierten Klassendateien in einem beliebigen Klassendateiformat gemäß Anspruch 1 des Hilfsantrags 2 (Merkmal M2.1.4*) aus einem standardmäßigen Java-Klassendateiformat (Merkmal M2.1.3) durch einen allgemeinen Konverter, der nur durch dieses Eingangs- und Ausgangsdateiformat bestimmt ist, lässt sich daher weder aus den genannten Ausführungsbeispielen, noch unter Berücksichtigung der weiteren Beschreibung des Streitpatents herleiten. Dies gilt auch für die damit verbundene Verwendung dieser in einem allgemeinen minimierten Klassendateiformat vorliegenden Klassendateien durch einen allgemeinen Interpreter beim Ausführen der in den Mikrocontroller geladenen Applikation. Ebenso kann auch das Erzeugen einer einzelnen allgemeinen Klassendatei (Merkmal M2.1.4**) aus einer Mehrzahl an standardmäßigen Java-Klassendateien (Merkmal M2.1.3*) gemäß Anspruch 1 des Hilfsantrags 3 nicht aus den Ausführungsbeispielen des Streitpatents entnommen werden. Entsprechendes gilt schließlich ein allgemeines minimiertes Klassendateiformat in Verbindung mit einer einzigen Klassendatei gemäß Merkmal M2.1.4*** im jeweiligen Anspruch 1 der Hilfsanträge 4 und 5.

Der Senat ist daher zu der Auffassung gelangt, dass für den Fachmann eine Kombination eines allgemeinen Interpreters für höhere Programmiersprachen und eines allgemeinen Konverters zusammen mit einzelnen, aus dem Kontext der Ausführungsbeispiele genommenen Merkmalen zum Erzeugen des Anwendungscodes in einer Programmierungsumgebung nicht eindeutig und

zweifelsfrei aus den ursprünglich eingereichten Unterlagen folgen (vgl. Merkmale M2.1.4*, M2.1.4** bzw. M2.1.4*** des jeweiligen Anspruchs 1 der Hilfsanträge 2 bis 5). Der Fachmann müsste dazu die den Ausführungsbeispielen entnehmbare Lehre nicht nur verallgemeinern, sondern vielmehr unter Hinzuziehen weiteren Wissens über Klassendateiformate und Programmiersprachen eine gezielte Auswahl aus den offenbarten Merkmalen treffen.

III.

Die Kostenentscheidung beruht auf § 84 Abs. 2 PatG i. V. m. § 91 Abs. 1 Satz 1 ZPO, die Entscheidung über die vorläufige Vollstreckbarkeit auf § 99 Abs. 1 PatG i. V. m. § 709 ZPO.

IV.

Rechtsmittelbelehrung

Gegen dieses Urteil steht den Beteiligten das Rechtsmittel der Berufung zu.

Die Berufungsschrift muss von einer in der Bundesrepublik Deutschland zugelassenen Rechtsanwältin oder Patentanwältin oder von einem in der Bundesrepublik Deutschland zugelassenen Rechtsanwalt oder Patentanwalt unterzeichnet und innerhalb eines Monats beim Bundesgerichtshof, Herrenstraße 45a, 76133 Karlsruhe schriftlich eingereicht werden. Die Berufungsfrist beginnt mit der Zustellung des in vollständiger Form abgefassten Urteils, spätestens aber mit dem Ablauf von fünf Monaten nach der Verkündung. Die Berufungsfrist kann nicht verlängert werden.

Die Berufungsschrift muss die Bezeichnung des Urteils, gegen das die Berufung gerichtet wird, sowie die Erklärung enthalten, dass gegen dieses Urteil Berufung

eingelegt werde. Mit der Berufungsschrift soll eine Ausfertigung oder beglaubigte Abschrift des angefochtenen Urteils vorgelegt werden.

Voit Zugleich für die wegen Krankheit verhinderte Richterin Kortge Voit

Dr. Schwengelbeck Dr. Otten-Dünnweber Altvater

prö