(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

PCT

(43) Internationales Veröffentlichungsdatum
18. Mai 2006 (18.05.2006)

(10) Internationale Veröffentlichungsnummer
WO 2006/050967 A1

(51) Internationale Patentklassifikation:
G06F 9/46 (2006.01)

(21) Internationales Aktenzeichen: PCT/EP2005/012104

(22) Internationales Anmeldedatum:
11. November 2005 (11.11.2005)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
10 2004 054 571.5
11. November 2004 (11.11.2004) DE

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **SYSGO AG** [DE/DE]; Am Pfaffenstein 14, 55270 Klein-Winternheim (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **KAISER, Robert** [DE/DE]; Riederbergstrasse 23, 65195 Wiesbaden (DE). **FUCHSEN, Rudolf** [DE/DE]; Finkenschlag 7, 55271 Stadecken-Elsheim (DE).

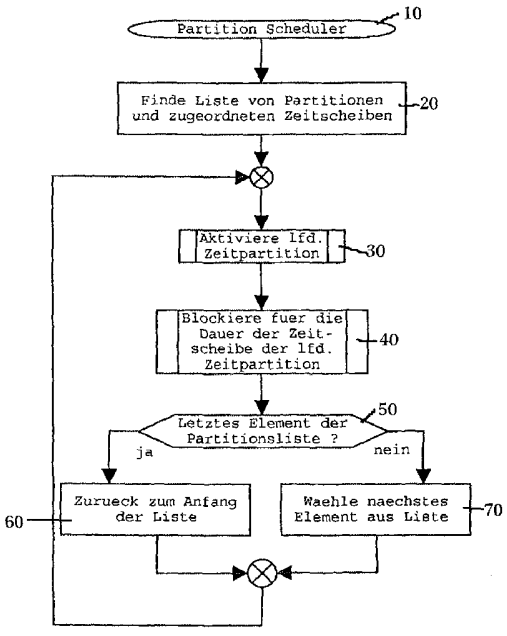
(74) Anwälte: **FELDKAMP, Rainer** usw.; Garmischer Strasse 4, 80339 München (DE).

(81) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare nationale Schutzrechtsart)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG,

[Fortsetzung auf der nächsten Seite]

(54) **Title:** METHOD FOR DISTRIBUTING COMPUTING TIME IN A COMPUTER SYSTEM

(54) **Bezeichnung:** VERFAHREN ZUR VERTEILUNG VON RECHENZEIT IN EINEM RECHNERSYSTEM



10 ... PRIORITY LEVEL SCHEDULER
20 ... FIND LIST OF PRIORITY LEVELS AND ASSOCIATED TIME SLICES
30 ... ACTIVATE RUNNING TIME PRIORITY LEVEL
40. BLOCK FOR THE DURATION OF THE TIME SLICE OF THE RUNNING TIME PRIORITY LEVEL
50 ... LAST ELEMENT OF THE PRIORITY LEVEL LIST?
JA ... YES
NEIN ... NO
60 ... BACK TO BEGINNING OF LIST
70 ... SELECT NEXT ELEMENT FROM LIST

(57) **Abstract:** The invention relates to a method for distributing computing time in a computer system on which run a number of partial processes or threads (NO, SO, P0, W1, W2) to which an assignment process or scheduler assigns computing time as required, priorities being associated with individual threads (NO, SO, P0, W1, W2) and the assignment of computing time being carried out according to the respective priorities. According to said method, the individual threads (NO, SO, P0, W1, W2) are respectively associated with a number of time priority levels (TauO, Taut, Tau2). A first (TauO) time priority level contains threads (NO, SO, P0) to which computing time is assigned as required at any time. A first scheduler respectively allocates a time slice to the individual time priority levels (TauO, Tau 1, Tau2), and respectively activates one of the time priority levels for the duration of the time slice thereof. A second scheduler monitors the threads of the first time priority level (TauO) and the threads of the respectively activated time priority level, and assigns computing time to said threads according to the priorities thereof.

(57) **Zusammenfassung:** Ein Verfahren zur Verteilung von Rechenzeit in einem Rechnersystem, auf dem eine Anzahl von Teilprozessen oder „Threads" (NO, SO, P0, W1, W2) abläuft, denen ein Zuteilungsprozess oder „Scheduler" bei Bedarf Rechenzeit zuteilt, wobei einzelnen Threads (NO, SO, P0, W1, W2) Prioritäten zugeordnet werden und die Zuteilung von Rechenzeit in Abhängigkeit von den jeweiligen Prioritäten erfolgt, werden die einzelnen Threads (NO, SO, P0, W1, W2) jeweils einer einer Anzahl von Zeitpartitionen (TauO, Taut, Tau2) zugeordnet. Eine erste (TauO) der Zeitpartitionen enthält Threads (NO, SO, P0), denen bei Bedarf zu jeder Zeit Rechenzeit zugeteilt wird. Ein erster Scheduler ordnet den einzelnen Zeitpartitionen (TauO, Tau 1, Tau2) jeweils eine Zeitscheibe zu und aktiviert jeweils eine der Zeitpartitionen für die Dauer ihrer Zeitscheibe. Ein zweiter Scheduler überwacht die Threads

[Fortsetzung auf der nächsten Seite]

WO 2006/050967 A1

SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Veröffentlicht:

— *mit internationalem Recherchenbericht*

— *vor Ablauf der für Änderungen der Ansprüche geltenden Frist; Veröffentlichung wird wiederholt, falls Änderungen eintreffen*

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

der ersten Zeitpartition (TauO) und die Threads der jeweils aktivierten Zeitpartition und weist diesen Threads in Abhängigkeit von ihren Proritäten Rechenzeit zu.

Verfahren zur Verteilung von Rechenzeit in einem Rechnersystem

Die Erfindung bezieht sich auf ein Verfahren zur Verteilung von Rechenzeit in einem Rechnersystem der im Oberbegriff des Anspruchs 1 genannt

Das erfindungsgemäße Verfahren ist insbesondere auf Rechnersysteme anwendbar, die mehrere Aufgaben in Form konkurrierender Teilprozesse oder „Threads" bearbeiten. Als „Thread" wird in diesem Zusammenhang eine Aktivität, bzw. ein in Ausführung befindliches Programm bezeichnet. Hierfür sind auch die Begriffe 'Task' oder 'Prozess' gebräuchlich, die jedoch je nach Kontext implizit Annahmen über weitere Eigenschaften wie zum Beispiel den Adressraum in dem das Programm abläuft, mit einschließen. Diese weiteren Eigenschaften sind für die vorliegende Erfindung jedoch ohne Bedeutung.

Welcher Thread wann zur Ausführung gelangt, d.h. welchem Thread Rechenzeit bzw. ein Prozessor zugeteilt wird, bestimmt ein Zuteilungsprozess, der üblicherweise als „Scheduler" bezeichnet wird. Der Scheduler ist damit ein Algorithmus, der entscheidet, welcher Thread wann zur Ausführung gelangt.

In einem derartigen Rechnersystem, in dem mehrere, voneinander unabhängige Aufgaben in Form von Threads abgearbeitet werden, kann ein Thread verschiedene Zustände annehmen:

rechnend: Der Thread hat einen Rechenzeitbedarf und befindet sich in Ausführung, d.h. ihm wurde ein Prozessor zugeteilt.

rechenwillig Der Thread ist bereit zur Ausführung und hat damit Rechenzeitbedarf und wartet darauf, dass ihm ein Prozessor zugeteilt wird.

blockiert Der Thread ist nicht bereit zur Ausführung, er wartet auf ein externes Ereignis.

Ein Thread geht aktiv vom Zustand rechnend in den Zustand blockiert über, indem er einen entsprechenden Aufruf an das Betriebssystem vornimmt. Der Wechsel von blockiert zu rechenwillig geschieht durch den Eintritt des externen Ereignisses oder auf Veranlassung eines anderen Thread, der dazu einen entsprechenden Systemaufruf vornimmt. Damit ein rechenwilliger Thread rechnend werden kann, muss ihm vom Scheduler einer der verfügbaren Prozessoren zugeteilt werden. Immer dann, wenn ein Thread entweder

aus dem Zustand rechnend in den Zustand blockiert, oder

aus dem Zustand blockiert in den Zustand rechenwillig

wechselt,

wird der Scheduler aufgerufen, um entweder den frei werdenden Prozessor einem neuen rechenbereiten Prozess zuzuteilen, oder um einem der rechnenden Threads den Prozessor zu entziehen, falls der rechenwillig werdende Thread als wichtiger eingestuft wird.

Aus Sicht des Schedulers existiert eine Anzahl rechenwilliger Threads, die um die Zuteilung eines Prozessors konkurrieren. Es wird hier davon ausgegangen, dass die Anzahl verfügbarer Prozessoren im allgemeinen kleiner als die Anzahl rechenbereiter Threads ist, d.h. der Scheduler findet normalerweise mehr 'Bewerber' um einen Prozessor vor, als er Prozessoren zu vergeben hat.

Es ist eine Vielzahl von Vorgehensweisen bekannt, wie mit Hilfe des Schedulers festgelegt wird, welchem Thread zum einem jeweiligen Zeitpunkt ein Prozessor zugeteilt wird.

Bei einem ersten Verfahren, das als Prioritätsverfahren bekannt ist, werden den einzelnen Threads Prioritäten zugeordnet. Ein Thread höherer Priorität erhält den

Vorzug gegenüber einem Thread niedrigerer Priorität. Threads gleicher Priorität sind je nach Anforderung entweder nicht zulässig, oder sie werden in der Reihenfolge, in der sie rechenwillig werden, in eine Warteschlange eingereiht. Die Priorität der einzelnen Threads kann je nach Anforderungen sowohl statisch festgelegt als auch dynamisch ermittelt und fortlaufend nachgeführt werden.

Beim Prioritätsverfahren besteht keine Möglichkeit, einem Thread ein Rechenzeitkontingent zu garantieren: eine hohe Priorität aufweisende Threads können die Prozessoren beliebig lange belegen und es kann im Extremfall vorkommen, dass eine niedrige Priorität aufweisende Threads niemals rechnend werden.

In einem weiteren Verfahren, das auch als Zeitscheibenverfahren (round robin) bekannt ist, wird die Zeitdauer, für die ein Thread einen Prozessor nutzen darf, begrenzt. Sobald ein gerade arbeitender Thread den ihm zugewiesenen Prozessor für mehr als eine festgelegte Dauer genutzt hat (d.h. er seine 'Zeitscheibe verbraucht' hat), wird er zwangsweise unterbrochen. Der Thread bleibt dabei rechenwillig und wird am Ende einer Warteschlange eingereiht. Die Dauer der Zeitscheibe kann je nach Anforderungen eine Konstante oder eine Thread-spezifische Variable sein. Dieses Verfahren erfordert einen externen Zeitgeber, der in bestimmten Zeitabständen über einen Interrupt das Betriebssystem aktiviert, damit dieses die vom laufenden Thread verbrauchte Zeit überwachen und ggf. den Scheduler aktivieren kann.

Das Zeitscheibenverfahren ermöglicht die Kontingentierung von Rechenzeit, doch es erlaubt keine schnelle Reaktion auf externe Ereignisse: Ein Thread, der z.B. aufgrund eines Interrupts rechenwillig wird, muss im ungünstigsten Fall den gesamten Zyklus, d.h. die Zeitscheiben sämtlicher anderer rechenwilliger Threads abwarten, bevor er rechnend werden kann.

Bei vielen Rechnersystemen, insbesondere bei Rechnersystemen mit einer Anzahl von Threads, von denen einige Echtzeitanforderungen unterliegen, müssen jedoch zumindest diese 'Echtzeit-Threads' innerhalb einer garantierten und idealerweise möglichst kurzen Maximalzeit auf externe Ereignisse reagieren, d.h. rechnend werden können. Zugleich existieren Threads, die zur Verrichtung ihrer jeweiligen Tätigkeiten ein Mindestkontingent an Rechenzeit benötigen. Auch ein solches Zeitkontingent für Threads muss garantiert werden können. Entsprechend sind Kombinationen der vorstehend genannten Verfahren bekannt.

Bei einem aus der EP-A2-0658 841 bekannten Verfahren werden Gruppen von Threads Prioritäten zugeordnet, und innerhalb der Gruppen wird den einzelnen Threads Rechenzeit in einem Zeitscheibenverfahren zugeteilt.

Bei einem weiteren aus der EP-A2-880 059 bekannten Verfahren werden die Threads in Klassen gruppiert, die in eine Hierarchie eingeordnet werden. Jede Klasse hat einen Zeitfunktionswert, der bestimmt, wann der Klasse Rechenzeit zugeteilt wird. Innerhalb einer Ebene der Hierarchie werden Rechenzeit-Prioritäten durch eine oder mehrere zeitbasierte Funktionen definiert, die konstant sein können oder sich dynamisch ändern.

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren der eingangs genannten Art zu schaffen, das verlässliche Zusagen über die einzelnen Threads zustehende Rechenzeit erlaubt, und bei dem zugesagte, dann aber nicht in Anspruch genommene Rechenzeit dynamisch umverteilt werden kann.

Diese Aufgabe wird durch die im Patentanspruch 1 angegebenen Merkmale gelöst.

Vorteilhafte Ausgestaltungen und Weiterbildungen der Erfindung ergeben sich aus den Unteransprüchen.

Erfindungsgemäß wird ein Verfahren zur Verteilung von Rechenzeit in einem Rechnersystem geschaffen, auf dem eine Anzahl von Teilprozessen oder „Threads" (N0, S0, P0, W1, W2) abläuft, denen ein Zuteilungsprozess oder „Scheduler" bei Bedarf Rechenzeit zuteilt, wobei einzelnen Threads (N0, S0, P0, W1, W2) Prioritäten zugeordnet werden und die Zuteilung von Rechenzeit in Abhängigkeit von den jeweiligen Prioritäten erfolgt.

Das erfindungsgemäße Verfahren ist dadurch gekennzeichnet,

dass die einzelnen Threads jeweils einer einer Anzahl von Zeitpartitionen zugeordnet werden,

dass eine erste der Zeitpartitionen Threads enthält, denen bei Bedarf zu jeder Zeit Rechenzeit zugeteilt wird,

dass ein erster Scheduler den einzelnen Zeitpartitionen jeweils eine Zeitscheibe zuordnet und jeweils eine der Zeitpartitionen für die Dauer ihrer Zeitscheibe aktiviert,

und dass ein zweiter Scheduler die Threads der ersten Zeitpartition und die Threads der jeweils aktivierten Zeitpartition überwacht und diesen Threads in Abhängigkeit von ihren Proritäten Rechenzeit zuweist.

Gemäß einer Ausgestaltung der Erfindung ist der erste Scheduler ein Thread der ersten Zeitpartition.

Hierbei können vorzugsweise eine niedrige Priorität aufweisende Threads der ersten Zeitpartition zusätzlich Rechenzeit der jeweils aktivierten weiteren Zeitpartitionen erhalten, wenn die weiteren Zeitpartitionen keine einen Rechenzeitbedarf aufweisende Threads enthalten.

Der erste Scheduler kann als Endlosschleife implementiert sein und eine hohe Priorität aufweisen.

Der zweite Scheduler wird immer dann aufgerufen, wenn ein Thread Rechenzeitbedarf anmeldet oder aufgibt.

Bei dem erfindungsgemäßen Verfahren kann damit sichergestellt werden, dass die Reaktionszeit von Threads auf externe Ereignisse stets unterhalb einer zugesicherten Maximalzeit liegt.

Die Erfindung wird nachfolgend anhand von in der Zeichnung dargestellten Ausführungsbeispielen noch näher erläutert.

In der Zeichnung zeigen:

Figur 1 ein Ausführungsbeispiel eines Flussdiagramms des ersten (Zeitpartitions-) Schedulers gemäß der Erfindung;

Figur 2 ein Ausführungsbeispiel eines Flussdiagramms des zweiten (Thread-) Schedulers gemäß der Erfindung;

Figur 3 eine Darstellung eines Beispiels für einen Schedulingablauf.

Das erfindungsgemäße Verfahren kombiniert die Eigenschaften des Zeitscheiben- und des Prioritätsverfahrens. Wie beim Prioritätsverfahren besitzen die Threads eine Priorität als Attribut. Zusätzlich wird jeder rechenwillige Thread einer jeweiligen Zeitpartition zugeordnet. Eine Zeitpartition bezeichnet somit eine Menge von rechenwilligen Threads mit im allgemeinen unterschiedlicher Priorität.

Es existiert eine spezielle erste Zeitpartition, die sogenannte Hintergrundpartition. Die Threads, die der Hintergrundpartition zugeordnet sind, können zu jeder Zeit rechnend werden. Für alle übrigen Zeitpartitionen gilt, dass die ihnen zugeordneten Threads immer nur dann rechnend werden können, wenn die betreffende Zeitpartition 'aktiv' ist, wobei zu jeder Zeit neben der Hintergrundpartition maximal eine weitere Zeitpartition aktiv sein kann.

Das Verfahren verwendet zwei überlagerte Scheduler: Ein erster, als Zeitpartitions-Scheduler bezeichneter, Scheduler ordnet den verschiedenen Zeitpartitionen jeweils eine Zeitscheibe zu und aktiviert je eine der Zeitpartitionen für die Dauer ihrer Zeitscheibe. Ein zweiter, als Thread-Scheduler bezeichneter, Scheduler betrachtet die Vereinigungsmenge der Threads der Hintergrundpartition und der derzeit aktiven Partition und weist den die höchste Priorität aufweisenden Threads aus dieser Menge die verfügbaren Prozessoren zu.

Die Threads der Hintergrundpartition haben, abhängig von ihrer Priorität verschiedene Funktion:

1. Eine hohe Priorität aufweisende Threads der Hintergrundpartition können Dienste implementieren, die von Threads verschiedener anderer Zeitpartitionen gemeinsam genutzt werden sollen. Darüber hinaus können solche Threads Aufgaben des Betriebssystems übernehmen, wie zum Beispiel die Überwachung anderer Threads oder des Zustandes des Gesamtsystems. Insbesondere kann der Zeitpartitions-Scheduler selbst als Thread der Hintergrundpartition realisiert werden.

2. Eine niedrigere Priorität aufweisende Threads der Hintergrundpartition erhalten dann Rechenzeit, wenn die derzeit aktive Zeitpartition keine rechenwilligen Threads besitzt, d.h. wenn die ihr zugeordneten Threads das ihnen zugesagte Zeitkontingent nicht in Anspruch nehmen, zum Beispiel, weil sie auf externe

Ereignisse warten müssen. Es besteht damit die Möglichkeit, zugesagte und dann doch nicht in Anspruch genommene Rechenzeit einer sinnvollen Nutzung zuzuführen.

In Figur 1 ist die Arbeitsweise eines Zeitpartitions-Schedulers 10 an einem typischen Beispiel veranschaulicht. Hierbei wird angenommen, dass die Zeitpartitionen des Systems zyklisch, jeweils für eine definierte Zeitscheibendauer aktiviert werden sollen. Der Zeitpartitions-Scheduler ist in diesem Beispiel als Endlosschleife implementiert, die als Thread hoher Priorität innerhalb der Hintergrundpartition läuft. Der Thread liest im Schritt 20 eine Liste von Zeitpartitionen mit den zugehörigen Zeitscheibendauern. Für jedes Listenelement wird zunächst die bezeichnete Zeitpartition im Schritt 30 aktiviert und anschließend blockiert der Thread im Schritt 40 für die Dauer der Zeitscheibe der soeben aktivierten Zeitpartition. Im Schritt 50 wird geprüft, ob das Ende der Liste erreicht ist. Wenn das Ende der Liste noch nicht erreicht ist, so wird im Schritt 70 das nächste Element aus der Liste ausgewählt. Ist das Ende der Liste erreicht, so beginnt die Abarbeitung über den Schritt 60 wieder am Anfang

In Figur 2 ist ein Ausführungsbeispiel des Thread-Schedulers 110 gezeigt. Der Thread-Scheduler wird immer dann aufgerufen, wenn der Zustand eines Thread von rechnend nach blockiert wechselt, oder wenn ein zuvor blockierter Thread den Zustand rechenwillig annimmt (z.B. aufgrund eines externen Ereignisses), was im Schritt 120 festgestellt wird. Im erstgenannten Fall muss im Schritt 130 der neuerdings blockierte Thread aus der Menge rechenwilliger Threads der ihm zugeordneten Zeitpartition entfernt werden, während im zweitgenannten Fall im Schritt 140 alle derzeit rechnenden Threads rechenwillig bleiben. Anschließend werden im Schritt 150 die höchsten Prioritäten sowohl der gerade aktiven Zeitpartition als auch der Hintergrundpartition ermittelt. Liegt die Maximalpriorität der gerade aktiven Zeitpartition über der der Hintergrundpartition, so wird im Schritt

160 der die höchste Priorität aufweisende Thread dieser Zeitpartition in den Zustand rechnend überführt, anderenfalls wird im Schritt 170 der die höchste Priorität aufweisende Thread der Hintergrundpartition gewählt. Nachfolgend wird im Schritt 180 der ausgewählte Thread aktiviert.

Wenn die Maximalpriorität der gerade aktiven Zeitpartition gleich der Maximalpriorität der Hintergrundpartition ist, so wird in diesem Beispiel die Hintergrundpartition bevorzugt. In Abhängigkeit von den jeweiligen Anforderungen könnte hier ebensogut eine Bevorzugung der gerade aktiven Zeitpartition bewirkt werden.

In Figur 3 ist ein Beispiel eines Scheduling-Ablaufs gezeigt. Diese Figur verdeutlicht anhand eines Beispieles die Funktionsweise des erfindungsgemäßen Verfahrens. In diesem Beispiel wird davon ausgegangen, dass nur ein Prozessor vorhanden ist. Es existieren zwei Zeitpartitionen Tau1 und Tau2 mit den Threads W1 und W2, zudem existiert die Hintergrundpartition Tau0. In Tau0 existiert ein Thread niedriger Priorität namens N0, ein Thread mittlerer Priorität namens S0, sowie ein Thread hoher Priorität namens P0.

P0 sei der Zeitpartitions-Scheduler gemäß Figur 1, d.h. dieser Thread wird jeweils für kurze Zeit rechnend, um die aktive Zeitpartition umzuschalten und blockiert anschließend jeweils für die Dauer der Zeitscheibe der aktiven Zeitpartition (iT1, bzw. T2 in Figur 3).

S0 sei ein von W1 und W2 gemeinsam genutzter 'Server', der auf Anforderung dieser Threads Dienste erbringt und ansonsten blockiert.

N0 sei ein ständig rechenwilliger Hintergrundthread, der niemals blockiert.

Im Beispiel ist zunächst keine der beiden Zeitpartitionen Tau1 bzw. Tau2 aktiv. Es existieren drei rechenwillige Threads: W1, W2 und N0. Da aber die Zeitpartitionen von W1 und W2 nicht aktiv sind, werden diese Threads trotz ihrer höheren Priorität nicht rechnend. In der Hintergrundpartition Tau0 ist N0 der einzige rechenwillige Thread (P0 und S0 sind blockiert). Daher ist N0 zunächst rechnend. Zum Zeitpunkt A wird der Zeitpartitions-Scheduler P0 rechenwillig und zugleich rechnend, da seine Priorität über der von N0 liegt. P0 aktiviert die Zeitpartition Tau1 und blockiert anschließend für die Dauer T1. Da nun Tau1 aktiv ist, wird der dieser Zeitpartition zugeordnete Thread W1 rechenwillig, und, da seine Priorität über der von N0 liegt, wird er rechnend. Zum Zeitpunkt B blockiert W1 um ein externes Ereignis abzuwarten. Dieses Ereignis tritt zum Zeitpunkt C ein. Die Zeit zwischen den Zeitpunkten B und C war der Zeitpartition Tau1 zugesichert worden, jedoch gibt es keinen Tau1 zugeordneten Thread, der diese Zeit in Anspruch nehmen könnte. Deshalb wird diese ungenutzte Zeit N0 zugewiesen. Nach Ablauf der Zeitscheibe T1 der Partition Tau1 zum Zeitpunkt D wird der Zeitpartitions-Scheduler P0 rechnend. Er aktiviert die Zeitpartition Tau2 und blockiert anschließend für die Dauer T2. Da nun Tau2 aktiv ist, wird der dieser Zeitpartition zugeordnete Thread W2 rechnend. Zum Zeitpunkt E fordert W2 einen Dienst vom Server S0 an und blockiert, um auf dessen Antwort zu warten. S0 wird daraufhin rechenwillig, und, da seine Priorität über der von N0 liegt, rechnend. Zum Zeitpunkt F hat S0 die angeforderte Dienstleistung erbracht, sendet eine entsprechende Benachrichtigung an seinen Auftraggeber W2 und blockiert in Erwartung des nächsten Auftrags. W2 wird daraufhin wieder rechnend und arbeitet bis zum Ablauf der Zeitscheibe seiner Zeitpartition Tau2. An dieser Stelle wird der Zeitpartitions-Scheduler P0 wieder rechnend und aktiviert wieder die Hintergrundpartition Tau0. Damit herrschen wieder die gleichen Verhältnisse wie zu Beginn des Zyklus.

Patentansprüche:

1. Verfahren zur Verteilung von Rechenzeit in einem Rechnersystem, auf dem eine Anzahl von Teilprozessen oder „Threads" (N0, S0, P0, W1, W2) abläuft, denen ein Zuteilungsprozess oder „Scheduler" bei Bedarf Rechenzeit zuteilt, wobei einzelnen Threads (N0, S0, P0, W1, W2) Prioritäten zugeordnet werden und die Zuteilung von Rechenzeit in Abhängigkeit von den jeweiligen Prioritäten erfolgt, dadurch gekennzeichnet,

dass die einzelnen Threads (N0, S0, P0, W1, W2) jeweils einer einer Anzahl von Zeitpartitionen (Tau0, Tau1, Tau2) zugeordnet werden,

dass eine erste (Tau0) der Zeitpartitionen Threads (N0, S0, P0) enthält, denen bei Bedarf zu jeder Zeit Rechenzeit zugeteilt wird,

dass ein erster Scheduler den einzelnen Zeitpartitionen (Tau0, Tau1, Tau2) jeweils eine Zeitscheibe zuordnet und jeweils eine der Zeitpartitionen für die Dauer ihrer Zeitscheibe aktiviert,

und dass ein zweiter Scheduler die Threads der ersten Zeitpartition (Tau0) und die Threads der jeweils aktivierten Zeitpartition überwacht und diesen Threads in Abhängigkeit von ihren Proritäten Rechenzeit zuweist.

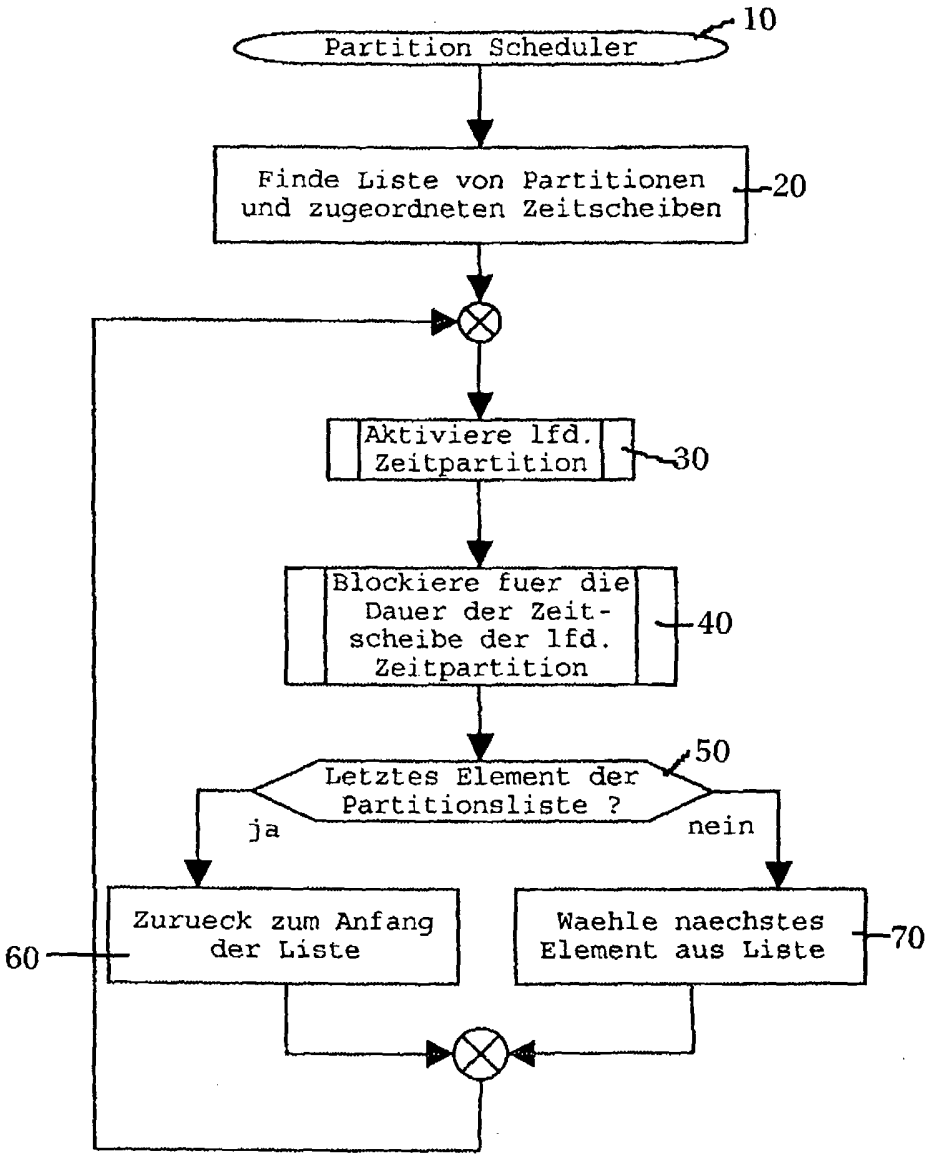
2, Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass der erste Scheduler ein Thread der ersten Zeitpartition (Tau0) ist.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass eine niedrige Priorität aufweisende Threads der ersten Zeitpartition (Tau0) zusätzlich Rechenzeit der jeweils aktivierten weiteren Zeitpartitionen (Tau1, Tau2) erhalten, wenn die weiteren Zeitpartitionen (Tau1, Tau2) keine einen Rechenzeitbedarf aufweisenden Threads enthalten.

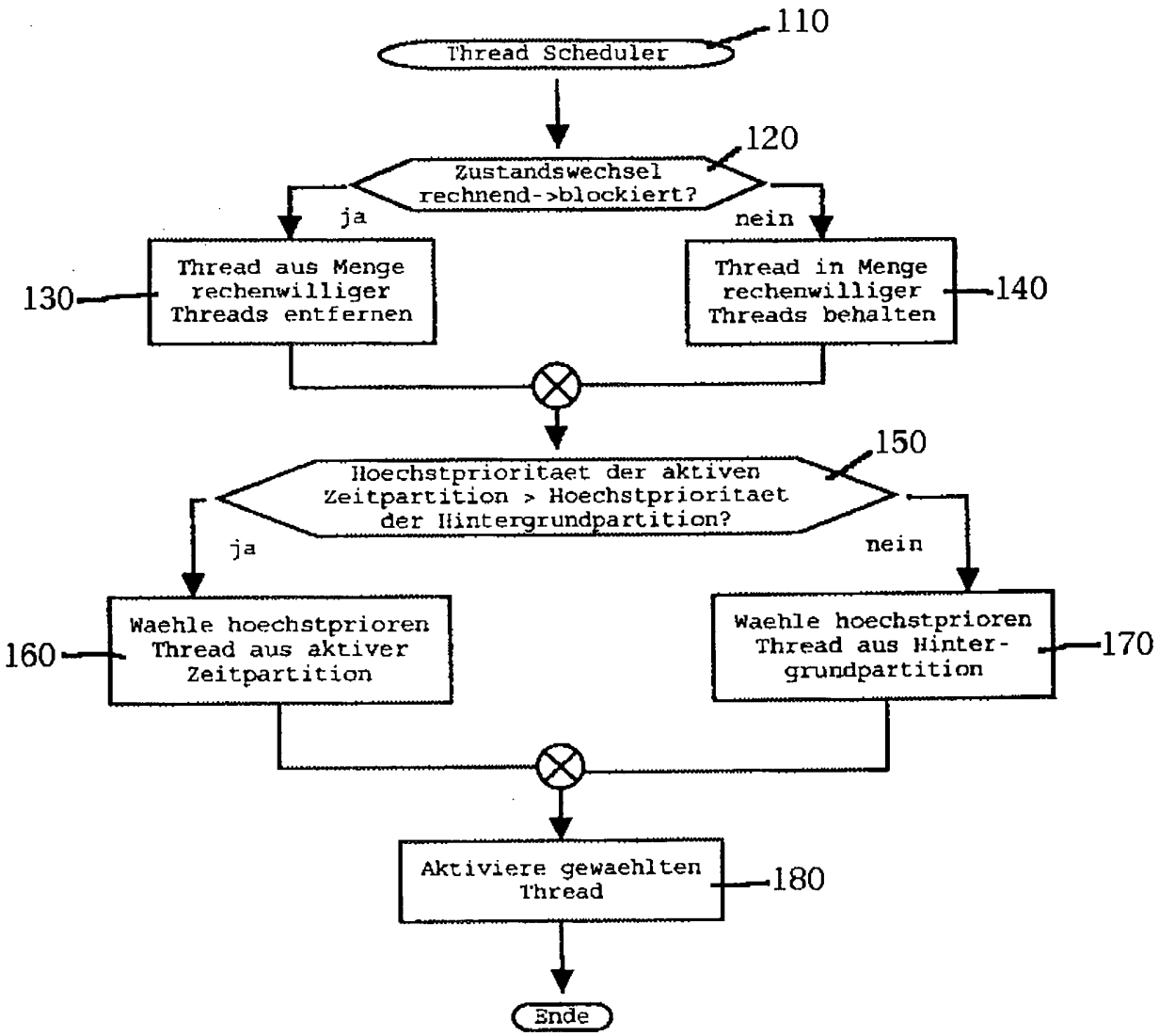
4. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, dass der erste Scheduler als Endlosschleife implementiert ist und eine hohe Priorität aufweist.

5. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, dass der zweite Scheduler immer dann aufgerufen wird, wenn ein Thread Rechenzeitbedarf anmeldet oder aufgibt.

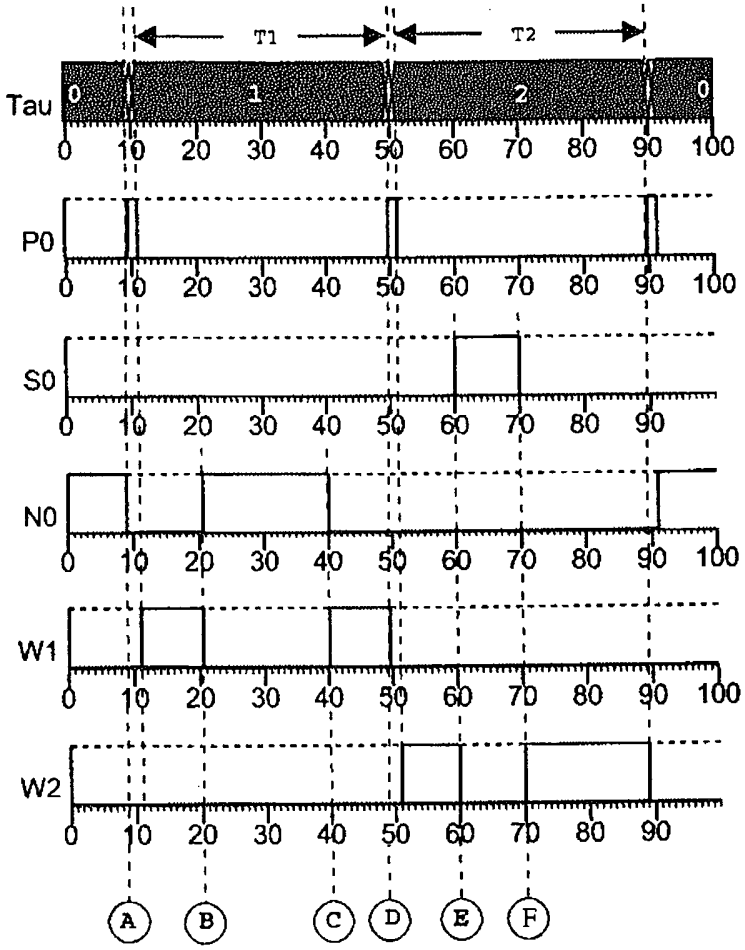
6. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, dass die Reaktionszeit von Threads auf externe Ereignisse stets unterhalb einer zugesicherten Maximalzeit liegt.


Partition Scheduler
10
Finde Liste von Partitionen und zugeordneten Zeitscheiben
20
Aktiviere lfd. Zeitpartition
30
Blockiere fuer die Dauer der Zeit-scheibe der lfd. Zeitpartition
40
Letztes Element der Partitionsliste ?
50
ja
nein
Zurueck zum Anfang der Liste
60
Waehle naechstes Element aus Liste
70

Figur 1


Thread Scheduler
110
Zustandswechsel rechnend->blockiert?
120
ja
nein
Thread aus Menge rechenwilliger Threads entfernen
130
Thread in Menge rechenwilliger Threads behalten
140
Hoechstprioritaet der aktiven Zeitpartition > Hoechstprioritaet der Hintergrundpartition?
150
ja
nein
Waehle hoechstprioren Thread aus aktiver Zeitpartition
160
Waehle hoechstprioren Thread aus Hinter-grundpartition
170
Aktiviere gewaehlten Thread
180
Ende

Figur 2



Figur 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2005/012104

A. CLASSIFICATION OF SUBJECT MATTER
G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	DATABASE INSPEC [Online] THE INSTITUTION OF ELECTRICAL ENGINEERS, STEVENAGE, GB; 1998, TERRASA A ET AL: "Extending RT-Linux to support flexible hard real-time systems with optional components" XP002366057 Database accession no. 6174825 abstract -/--	1-6

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 March 2006	27/03/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Noll, J

4

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	& PROCEEDINGS OF ACM SIGPLAN 1998 WORKSHOP ON LANGUAGES, COMPILERS, AND TOOLS FOR EMBEDDED SYSTEMS (IN CONJUNCTION W/PLDI'98) 19-20 JUNE 1998 MONTREAL, QUE., CANADA, June 1998 (1998-06), pages 41-50, Languages, Compilers, and Tools for Embedded Systems. ACM SIGPLAN Workshop LCTES'98. Proceedings Springer-Verlag Berlin, Germany ISBN: 3-540-65075-X page 42, paragraph 1-5 page 43, paragraph 1-4 page 44, paragraphs 1,2 page 46, paragraph 2 page 47, paragraphs 2,3 page 49, paragraphs 1,2 figure 1	
X	TERRASA A ET AL: "Real-time synchronization between hard and soft tasks in RT-Linux" REAL-TIME COMPUTING SYSTEMS AND APPLICATIONS, 1999. RTCSA '99. SIXTH INTERNATIONAL CONFERENCE ON HONG KONG, CHINA 13-15 DEC. 1999, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 13 December 1999 (1999-12-13), pages 434-441, XP010365387 ISBN: 0-7695-0306-3 abstract page 434, left-hand column, paragraph 1 - page 437, left-hand column, paragraph 2	1-6
A	US 5 745 778 A (ALFIERI ET AL) 28 April 1998 (1998-04-28) abstract column 1, line 56 - column 2, line 17 column 2, line 35 - column 4, line 35 column 5, line 4 - column 6, line 19 column 7, line 11 - column 9, line 30 figures 1-5E	1-6
A	EP 1 286 264 A (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD) 26 February 2003 (2003-02-26) abstract paragraphs [0033] - [0055] figures 1-17	1-6

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2005/012104

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5745778 A	28-04-1998	NONE	
EP 1286264 A	26-02-2003	US 2003037091 A1	20-02-2003

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2005/012104

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
G06F9/46

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
G06F

Recherchierte, aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, WPI Data, PAJ, INSPEC

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	DATABASE INSPEC [Online] THE INSTITUTION OF ELECTRICAL ENGINEERS, STEVENAGE, GB; 1998, TERRASA A ET AL: "Extending RT-Linux to support flexible hard real-time systems with optional components" XP002366057 Database accession no. 6174825 Zusammenfassung -/--	1-6

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen [X] Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
7. März 2006	27/03/2006

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Noll, J

4

Formblatt PCT/ISA/210 (Blatt 2) (April 2005)

C. (Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
	& PROCEEDINGS OF ACM SIGPLAN 1998 WORKSHOP ON LANGUAGES, COMPILERS, AND TOOLS FOR EMBEDDED SYSTEMS (IN CONJUNCTION W/PLDI'98) 19-20 JUNE 1998 MONTREAL, QUE., CANADA, Juni 1998 (1998-06), Seiten 41-50, Languages, Compilers, and Tools for Embedded Systems. ACM SIGPLAN Workshop LCTES'98. Proceedings Springer-Verlag Berlin, Germany ISBN: 3-540-65075-X Seite 42, Absatz 1-5 Seite 43, Absatz 1-4 Seite 44, Absätze 1,2 Seite 46, Absatz 2 Seite 47, Absätze 2,3 Seite 49, Absätze 1,2 Abbildung 1	
X	TERRASA A ET AL: "Real-time synchronization between hard and soft tasks in RT-Linux" REAL-TIME COMPUTING SYSTEMS AND APPLICATIONS, 1999. RTCSA '99. SIXTH INTERNATIONAL CONFERENCE ON HONG KONG, CHINA 13-15 DEC. 1999, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 13. Dezember 1999 (1999-12-13), Seiten 434-441, XP010365387 ISBN: 0-7695-0306-3 Zusammenfassung Seite 434, linke Spalte, Absatz 1 - Seite 437, linke Spalte, Absatz 2	1-6
A	US 5 745 778 A (ALFIERI ET AL) 28. April 1998 (1998-04-28) Zusammenfassung Spalte 1, Zeile 56 - Spalte 2, Zeile 17 Spalte 2, Zeile 35 - Spalte 4, Zeile 35 Spalte 5, Zeile 4 - Spalte 6, Zeile 19 Spalte 7, Zeile 11 - Spalte 9, Zeile 30 Abbildungen 1-5E	1-6
A	EP 1 286 264 A (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD) 26. Februar 2003 (2003-02-26) Zusammenfassung Absätze [0033] - [0055] Abbildungen 1-17	1-6

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/EP2005/012104

Im Recherchenbericht angeführtes Patentdokument		Datum der Veröffentlichung	Mitglied(er) der Patentfamilie		Datum der Veröffentlichung
US 5745778	A	28-04-1998	KEINE		
EP 1286264	A	26-02-2003	US	2003037091 A1	20-02-2003