(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 600 07 252 T2** 2004.09.16

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 090 344 B1**
(21) Deutsches Aktenzeichen: **600 07 252.5**
(86) PCT-Aktenzeichen: **PCT/US00/05571**
(96) Europäisches Aktenzeichen: **00 916 025.0**
(87) PCT-Veröffentlichungs-Nr.: **WO 00/52564**
(86) PCT-Anmeldetag: **03.03.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **08.09.2000**
(97) Erstveröffentlichung durch das EPA: **11.04.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **17.12.2003**
(47) Veröffentlichungstag im Patentblatt: **16.09.2004**

(51) Int Cl.[7]: **G06F 3/00**

(30) Unionspriorität:
263148 **05.03.1999** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB, IT, NL

(73) Patentinhaber:
Amulet Technologies, LLC, Campbell, Calif., US

(72) Erfinder:
KLASK, J., Kenneth, San Jose, US

(74) Vertreter:
derzeit kein Vertreter bestellt

(54) Bezeichnung: **GRAPHISCHER BENUTZERSCHNITTSTELLENTREIBER FÜR EINGEBETTETE SYSTEME**

Beschreibung

GEBIET DER ERFINDUNG

[0001] Diese Erfindung betrifft Computersysteme und insbesondere eingebettete Systeme, d.h. andere als programmierbare Allzweck-Computer.

HINTERGRUND

[0002] Eingebettete Systeme sind altbekannt; dies bezieht sich auf Mikroprozessoren und Mikrocontroller (die nachstehend generisch als Mikroprozessoren bezeichnet werden), die in anderen Einrichtungen als Allzweck-Computern verwendet werden. Z.B. weisen viele Haushaltsgeräte (beispielsweise Mikrowellenöfen) eingebettete Mikroprozessoren auf, die einen Betrieb des Geräts steuern. Der Mikroprozessor nimmt typischerweise eine Benutzereingabe, z.B. von der Tastatur des Mikrowellenofens, an und steuert einen Betrieb des Mikrowellenofens, z.B. den Erwärmungsgrad und die Dauer des Kochvorgangs. Der eingebettete Mikroprozessor steuert auch die Anzeige der Einrichtung, die in einem Mikrowellenofen eine kleine LCD (Flüssigkristallanzeige) ist. D.h., die Intelligenz von derartigen Geräten befindet sich in dem eingebetteten Mikroprozessor, der die Schnittstelle zu dem menschlichen Benutzer bereitstellt. Typischerweise wird dies durch Firmware durchgeführt, d.h. Computersoftware, die von dem eingebetteten Mikroprozessor ausgeführt und in einem Speicher gespeichert wird, der zu dem Mikroprozessor gehört oder ein Teil von diesem ist. Zusätzlich zu der Ausführung der Software, um mit der gesteuerten Einrichtung in Wechselwirkung zu treten, nimmt der eingebettete Mikroprozessor auch einen Eingang von dem menschlichen Benutzer, z.B. über die Tastatur, an und decodiert diesen und außerdem stellt er eine visuelle Rückkopplung auf der Anzeige bereit, indem Text und/oder grafische Information an einem Anzeige-Controller bereitgestellt wird, der wiederum das LCD-Feld ansteuert.

[0003] Wie in dem Blockdiagramm der **Fig.** 1 gezeigt, ist der eingebettete Mikroprozessor **10** (der in der Zeichnung mit der alternativen Terminologie "Mikrocontroller" bezeichnet ist) eine kommerziell erhältliche Einrichtung, z.B. ein 8- oder 16-Bit-Mikrocontroller des Typs, der von einer Anzahl von Geschäften erhältlich ist. Dieser eingebettete Mikroprozessor schließt herkömmlicherweise, zusätzlich zu seiner logischen Schaltungsanordnung, einen Speicher wie ein ROM (einen Nur-Lese-Speicher), der hält, was als Firmware **12** bezeichnet wird, die ein Typ einer Computersoftware ist, und auch ein herkömmliches RAM (einen Speicher mit wahlfreiem Zugriff), der nicht gezeigt ist, ein. Die Firmware **12** führt die angezeigten Funktionen des Anwendungsflusses, der Reaktion der Einrichtungssteuerung (der gesteuerten Einrichtung, von der der eingebettete Mikroprozessor ein Teil ist) auf einen Benutzereingang, und der Möglichkeit zum Zeichnen von Pixeln an dem Bildpuffer **30** des Anzeige-Controllers **24** aus.

[0004] Wie gezeigt ist der Mikroprozessor **10** mit einer Benutzereingabeeinrichtung **14**, z.B. einer Tastatur, einer Infrarot-Fernsteuerung, wie bei Fernsehgeräten verwendet, oder einer Berührungsbildschirm-Eingabeeinrichtung, gekoppelt. Die zugehörige gesteuerte Einrichtung (die nicht gezeigt ist) ist z.B. ein Gerät wie ein Mikrowellenofen, eine Waschmaschine, oder ein Automobilsystem, oder ein wissenschaftliches Instrument, oder ein Maschinenwerkzeug, und ist in herkömmlicher Weise mit dem Mikroprozessor **10** verbunden. Es sei darauf hingewiesen, dass die Linien, die die Blöcke in **Fig.** 1 verbinden, Busse darstellen, d.h. parallele Mehrfachleitungs-Verbindungen. Der eingebettete Mikroprozessor **10** liefert einen Eingang (Befehle) von dem menschlichen Benutzer über die Benutzereingabeeinrichtung **14**, um die gesteuerte Einrichtung zu steuern, und gibt dem Benutzer Anzeigen auf der Anzeige **20**. Die Anzeige **20** wird über eine herkömmliche Pixeltreiber/Video-Schaltungsanordnung **22** angesteuert. Die Benutzereingabeeinrichtung **14** beeinflusst natürlich nicht direkt die gesteuerte Einrichtung und außerdem steuert sie auch nicht direkt den Anzeigeprozessor **20**. Anstelle davon nimmt der eingebettete Mikroprozessor **10** den Benutzereingang von der Benutzereingabeeinrichtung **14** an und decodiert diesen, steuert dann die gesteuerte Einrichtung und stellt Information für den Benutzer auf der Anzeige **20** bereit. In ähnlicher Weise zeigt die Anzeigeeinrichtung **20** die Anzeigeinformation von der Benutzereingabeeinrichtung **14** und auch nicht von der gesteuerten Einrichtung direkt an; anstelle davon zeigt sie nur Information an, mit der sie von dem eingebetteten Mikroprozessor **10** versehen wird. Die Anzeige findet über den Anzeige-Controller **24** statt, der oft eine getrennte, kommerziell erhältliche, integrierte Schaltung ist. Der Anzeige-Controller **24** umfasst mehrere altbekannte Elemente, die eine Bus-Schnittstelle **28** des Mikrocontrollers (des Mikroprozessors), die den Bildpuffer **30** ansteuert, und die zugehörige LCD/Video-Schnittstelle **34** sind. Wie gezeigt, ist die Anzeigeeinrichtung z.B. eine LCD (eine Flüssigkristallanzeige), eine VFD (Vakuum-Fluoreszenzanzeige), eine CRT (Kathodenstrahlröhre), etc.

[0005] Das System der **Fig.** 1 ist altbekannt und ist seit vielen Jahren in Verwendung gewesen. Es ist allgemein geeignet für Herstellungsprodukte eines großen Volumens, wie beispielsweise Haushaltsgeräte, bei denen die Kosten einer Herstellung (der Teile) wichtig ist und bei denen nicht-wiederkehrende Entwicklungskosten zum Entwickeln von Software relativ weniger bedeutend sind. Der Grund dafür besteht darin, dass die von dem Mikroprozessor **10** ausgeführte Firmware für jede Klasse von gesteuerter Einrichtung, sowie für die Benutzereingabeeinrichtung **14** und die Anzeige **20**, zugeschnitten werden muss. Dies erfordert beträchtliche Softwareentwicklungsanstrengungen. Jedoch ist dieser Ansatz für Produkte,

die nicht in einer Massenherstellung erzeugt werden, beispielsweise für industrielle Steuersysteme, oder für Produkte mit einer begrenzten Herstellung, bei der die Softwareentwicklungskosten relativ wichtiger als die Kosten der integrierten Schaltungen sind, weniger gut geeignet. Sogar für die Produkte in Massenherstellung, die häufigen Änderungen in der Firmware ausgesetzt sind, die von dem eingebetteten Mikroprozessor **10** ausgeführt wird, sind auch die Kosten einer Änderung der Firmware hoch und der Ansatz der **Fig.** 1 ist relativ kostenaufwändig und ineffizient. Somit weist dieser Ansatz Nachteile im Hinblick auf die Entwicklungszeit und die Entwicklungskosten auf.

[0006] Agranat, LD. "Engineering Web Technologies For Embedded Applications" IEEE Internet Computing, May-June 1998, IEEE USA, vol. 2, No. 3, Seiten 40–45 offenbart die Verwendung von Internet (World Wide Web) Technologien in eingebetteten Systemen. Es werden Web-freigeschaltete Einrichtungen offenbart, die das HTTP-Standardprotokoll verwenden, um Web-Seiten von dem eingebetteten System an einen Web-Browser zu übertragen und HTML-Daten von dem Browser zurück an die Einrichtung zu übertragen, wodurch eingebettete Systeme in das Internet integriert werden.

[0007] Smith, I. "Graphics in embedded systems: add HTML to Java – and C-based user-Interface options", EDN (US edition) USA, 18 February 1999, Cahners Publishing, vol. 44, No. 4, Seiten 95–96, 100-101 offenbart die Verwendung von Grafik in eingebetteten Systemen einschließlich einer Verwendung von HTML für Schnittstellen.

[0008] In Übereinstimmung mit dem ersten Aspekt der vorliegenden Erfindung ist ein eingebettetes Steuersystem vorgesehen, um eine zugehörige Einrichtung zu steuern oder zu überwachen, wobei die zugehörige Einrichtung einen Eingangs/Ausgangs-Abschnitt einschließt, wobei das eingebettete Steuersystem umfasst:

einen ersten Prozessor, der über eine Eingangs/Ausgangs-Schaltungsanordnung mit dem Eingangs/Ausgangs-Abschnitt zum Steuern des Betriebs der zugehörigen Einrichtung gekoppelt ist; und

einen Speicher, der zu dem ersten Prozessor gehört, wobei der Speicher ein oder mehrere Dokumente speichert, die miteinander verbunden sind, aber nicht einen ausführbaren Code enthalten, um Verfahren zu beschreiben, um die zugehörige Einrichtung durch Verbinden von Funktionen, die durch den ersten Prozessor ausführbar und lokal zu diesem sind, mit Beschreibungen der Wechselwirkungen, die zum Zugreifen auf die Eingangs/Ausgangs-Schaltungsanordnung benötigt werden, zu steuern oder zu überwachen;

wobei der erste Prozessor das gespeicherte Dokument von dem Speicher empfängt und die Steuerungs- oder Überwachungsverfahren der gespeicherten Dokumente durch Ausführen der Funktionen, auf die von den gespeicherten Dokumenten verwiesen wird, ausführt, um auf der Eingangs/Ausgangs-Schaltungsanordnung verbunden mit den Funktionen durch die gespeicherten Dokumente zu arbeiten, um dadurch einen Betrieb des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen;

einen zweiten Prozessor, der zwischen den ersten Prozessor und die Eingangs/Ausgangs-Schaltungsanordnung gekoppelt ist, wobei der zweite Prozessor interne Ressourcen einschließt, um ein Steuern oder Zugreifen auf der Eingangs/Ausgangs-Schaltungsanordnung zu erleichtern;

wobei die internen Ressourcen mit Funktionen, die von dem ersten Prozessor ausführbar sind, über Verbindungen von den gespeicherten Dokumenten verbunden sind, um dadurch einen Betrieb des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen.

[0009] In Übereinstimmung mit dem zweiten Aspekt der vorliegenden Erfindung ist ein Verfahren zum Verwenden eines eingebetteten Steuersystems, um eine zugehörige Einrichtung zu steuern oder zu überwachen, vorgesehen, wobei die zugehörige Einrichtung einen Eingangs/Ausgangs-Abschnitt einschließt und das eingebettete Steuersystem einen ersten Prozessor, der über eine Eingangs/Ausgangs-Schaltungsanordnung mit dem Eingangs/Ausgangs-Abschnitt verbunden ist, zum Steuern eines Betriebs der zugehörigen Einrichtung, und einen Speicher, der zu dem ersten Prozessor gehört, einschließt, umfassend die folgenden Schritte: Speichern von einem oder mehreren Dokumenten in dem Speicher verbunden miteinander, aber nicht einen ausführbaren Code enthaltend, um Verfahren zu beschreiben, um die zugehörige Einrichtung durch Verbinden von Funktionen, die von dem ersten Prozessor ausführbar und zu diesem lokal sind, mit Beschreibungen der Wechselwirkungen, die zum Zugreifen auf die Eingangs/Ausgangs-Schaltungsanordnung benötigt werden, zu steuern oder zu überwachen; Empfangen des gespeicherten Dokuments von dem Speicher in dem ersten Prozessor; Ausführen der Steuerungs- oder Überwachungs-Verfahren der gespeicherten Dokumente durch Ausführen der Funktionen, auf die von den gespeicherten Dokumenten verwiesen wird, um auf der Eingangs/Ausgangs-Schaltungsanordnung verbunden mit den Funktionen durch die gespeicherten Dokumente zu arbeiten, um dadurch Operationen des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen; Bereitstellen eines zweiten Prozessors (**42d**); Koppeln des zweiten Prozessors zwischen den ersten Prozessor und die Eingangs/Ausgangs-Schaltungsanordnung, wobei der zweite Prozessor interne Ressourcen einschließt, um eine Steuerung der zugehörigen Einrichtung oder einen Zugriff auf der Eingangs/Ausgangs-Schaltungsanordnung zu erleichtern; und Verbinden der internen Ressourcen mit Funktionen, die von dem ersten Prozessor ausführbar sind, über Verbindungen von den

gespeicherten Dokumenten, um dadurch einen Betrieb des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen.

[0010] In Übereinstimmung mit dieser Erfindung arbeitet ein eingebettetes Steuersystem zum Steuern einer Einrichtung derart, dass die Last zum Aufnehmen einer Eingabe eines menschlichen Benutzers (oder einer Maschine) und Bereitstellen von Information (einer Ausgabe) an einen menschlichen Benutzer oder eine Maschine über z.B. eine Anzeige von dem eingebetteten Mikroprozessor an einen zweiten Prozessor verschoben wird. Der zweite Prozessor, der hier als "Hypertext"-Prozessor bezeichnet wird, ist z.B. ein Mikroprozessor, Mikrocontroller, oder ein ähnlicher Aufbau, der in der Lage ist, ein Hypertext-Markup-Language Dokument zu verarbeiten, wie nachstehend erläutert. Das eingebettete Steuersystem steuert und/oder überwacht die gesteuerte Einrichtung und ist anwendungsspezifisch, im Gegensatz z.B. zu einem Personalcomputer, auf dem irgendein Anwendungsprogramm ablaufen kann. Der Anzeigecontroller der **Fig.** 1 wird effektiv beseitigt und seine Funktionen anstelle davon dem Hypertext-Prozessor zugeordnet. Sowohl die Benutzer-(oder Maschinen-)Eingabeeinrichtung als auch die Anzeige (oder eine andere Ausgabeeinrichtung) sind mit dem Hypertext-Prozessor und nicht mit dem eingebetteten Mikroprozessor gekoppelt. Der Hypertext-Prozessor ist ein zweiter z.B. Mikroprozessor, der auf einem Chip getrennt von dem eingebetteten Mikroprozessor sein kann.

[0011] Der Hypertext-Prozessor bestimmt, welche Operationen auf den Empfang z.B. einer Benutzereingabe, z.B. von einer verbundenen Tastatur, vorgenommen werden sollen. Der Hypertext-Prozessor führt Aktionen, die in dem Hypertext-Markup-Language Dokument beschrieben werden, und Befehle des eingebetteten Mikroprozessors aus, um mit der gesteuerten Einrichtung zu arbeiten und deren interne gemeinsam verwendete Variablen zu aktualisieren. Der Hypertext-Prozessor aktualisiert auch die Anzeige als eine Funktion der gemeinsam verwendeten Variablen, die zu dem eingebetteten Mikroprozessor intern sind. Die Benutzerschnittstellen-Software (der Code) liegt nicht in dem Hypertext-Prozessor und außerdem wird sie nicht von dem eingebetteten Mikroprozessor ausgeführt/interpretiert. Anstelle davon ist ein (Hypertext) Dokument, das die Benutzerschnittstelle beschreibt, zu dem Hypertext-Prozessor extern und liegt in dem Speicherraum des eingebetteten Mikrocontrollers oder in einer seriellen Speichereinrichtung (z.B. einem seriellen EEPROM, einem FLASH ROM, einer Smartkarte, etc.). Dieses Hypertext-Dokument, das die Benutzerschnittstelle beschreibt, wird an dem Hypertext-Prozessor bei Aufforderung von dem Hypertext-Prozessor bereitgestellt ("serviert"). Somit wird die Benutzerschnittstelle tatsächlich von dem Hypertext-Prozessor ausgeführt, obwohl sie dort nicht permanent liegt.

[0012] In einer Ausführungsform wird das Benutzerschnittstellendokument in einer bestimmten Hypertext Markup Language (HTML), die hier als µHTML bezeichnet wird, codiert. Der generische Name "Hypertext Markup Language" bezieht sich auf:

[0013] Hypertext – ein Verfahren zum Bereitstellen von Verbindungen innerhalb und zwischen Dokumenten; popularisiert durch Multimedia-Genehmigungssysteme, die das Hypertext-Konzept verwendet haben, um den Inhalt eines Textdokuments mit einem anderen Dokument, welches in bestimmten Multimedia-Formaten codiert ist, zu verbinden.

[0014] Markup Language – ein Verfahren zum Einbetten von speziellen Steuercodes (TAGS), die den Aufbau sowie das Verhalten eines Dokuments beschreiben.

[0015] Wie die herkömmliche HTML enthalten µHTML-Dateien ("Dokumente") sowohl eine Steuerinformation (Markup Tags) als auch einen Inhalt (ASCII-Text), die zusammen das Erscheinen und den Inhalt einer Benutzerschnittstelle beschreiben. Zusätzlich stellen beide Markup Sprachen (Markup Languages) die Möglichkeit bereit, auf Ressourcen zu verweisen, die zu dem Dokument extern sind. Verglichen mit der herkömmlichen HTML ist µHTML kleiner, einfacher zu interpretieren, und definiert eine spezielle Bibliothek von GUI (Grafische Benutzerschnitstelle) Objekten, Datenverarbeitungsobjekten, die für eine Pipeline-Verarbeitung geeignet sind, und andere Systemhilfsmittel, die zu der eingebetteten Systemsoftware gemeinsam sind. Ein Schlüsselmerkmal von µHTML ist deren Fähigkeit, die Schnittstellen zu Ressourcen, die unter vernetzten eingebetteten Untersystemen verteilt sind, zu beschreiben und die Daten von diesen Ressourcen mit den Funktionen eines Host-Prozessors zu verbinden.

[0016] Um µHTML leicht zum Durchsehen (Parsing) zu machen, läuft ihr ein Verzeichnis von Zeigern auf jedes Tag voraus. Um sie kompakt zu machen, wird jedes Tag durch ein einzelnes Byte (nachstehend als ein op-Code bezeichnet) dargestellt. Nach jedem op-Code folgt ein einzigartiger Satz von Daten mit binären Eigenschaften, wie eine X–Y Koordinateninformation, Zeiger auf andere Ressourcen, und andere Eigenschaften. Für jedes Objekt in der GUI Objektbibliothek gibt es einen einzigartigen op-Code (Operationscode). Diese Objekte sind z.B. Drucktasten, hochfahrende Listen, und andere Sorten von visuellen (oder hörbaren) Anzeigen. Es gibt auch op-Codes für Objekte, die Verfahren enthalten, um Daten zu verarbeiten und Daten an und von anderen Objekten oder externen Ressourcen umzuleiten, z.B. ein Objekt, welches auf eine Variable von einer "externen Ressource 0" verweist, kann die Variablendaten zu jeden 100 ms abtasten und die Ergebnisse an ein anderes Objekt lenken, das auf eine Variable von "externe Ressource 1" verweist. Jedem Bibliotheks-Objekt op-Code folgt unmittelbar eine Datenstruktur, die für das Objekt einzigartig ist. Die Daten, die in der Datenstruktur enthalten sind, sind für das Exemplar des Bibliotheksobjekts spezifisch. In dieser Weise wird

der Speicher, der für jedes Exemplar von sämtlichen verwendeten Objekten zugeordnet ist, statisch in dem Speicher, der das µHTML Dokument puffert, zugeordnet. Wenn auf externe Ressourcen verwiesen wird, wird die Datenstruktur bereitgestellt, um das Format der Nachrichten zu beschreiben, die benötigt werden, um einen Zugriff auf die externe Ressource bereitzustellen. Um z.B. eine Variable zu lesen, die zu einer externen Einrichtung gehört, beschreibt die Datenstruktur einen "Get" ("Holen") Befehl und eine "Return" ("Zurückgeben") Antwort. Typischerweise enthält der Get Befehl eine Identifikation zu irgendeiner externen Einrichtung und einen Index in eine Nachschlagtabelle auf der externen Einrichtung, die Referenzen auf Variablen, Funktionen oder Dateien bereitstellt. Zusätzlich zu der Identifikation der externen Einrichtung und dem Nachschlagtabellen-Index enthält die Return Antwort auch die angeforderten Daten.

[0017] In einer Ausführungsform wird dieses Benutzerschnittstellen-Hypertextdokument unter Verwendung von herkömmlichen Internet-Webseiten-Entwicklungswerkzeugen des Typs, der kommerziell erhältlich ist, entwickelt; dies ist nicht beschränkend. Benutzerschnittstellenobjekte werden in einer Ausführungsform mit JAVA Applets simuliert, die Objekten in der GUI Objektbibliothek entsprechen. Auf die simulierten GUI-Objekte wird von innerhalb des herkömmlichen HTML-Dokuments durch Verwenden der gleichen Standard-Tags, die zum Verweisen auf irgendein herkömmliches JAVA Applet verwendet werden, verwiesen. Standardmäßige HTML-Tags werden ebenfalls verwendet, um den Anzeigeinhalt zu formatieren und auf Ressourcen zu zeigen, die in Einrichtungen liegen, die extern zu dem Hypertext-Prozessor sind.

[0018] Das Benutzerschnittstellendokument kann dann auf einem herkömmlichen Web-Browser, für Systementwicklungszwecke, betrachtet werden. (Natürlich hat dies wenig mit dem tatsächlichen Benutzerbetrieb der gesteuerten Einrichtung zu tun, ist aber ein Teil von dessen Benutzerschnittstellen-Konstruktion und -Entwicklung). Diese HTML/JAVA Web-Seite kann dann in ein kompakteres µHTML-Format durch einen Compiler umgewandelt (vor-compiliert) werden, der speziell dafür ausgelegt ist, um: (1) die herkömmlichen HTML-Tags zu entfernen und sie durch einen entsprechenden µHTML op-Code zu ersetzen; (2) die Attributketten (attributes strings) der HTML-Tags in eine binäre Struktur umzuwandeln, die für den µHTML op-Code geeignet ist; (3) Referenzen auf sämtliche JAVA-Applets und Parameter durch einen entsprechenden op-Code und Objektdaten zu ersetzen; (4) zusätzliche Daten umzuformatieren und hinzuzufügen, um ein Parsing und eine Ausführung von dem Hypertext-Prozessor zu vereinfachen, und (5) Referenzen auf Ressourcen, die zu dem Hypertext-Prozessor extern sind (d.h. ein ausführbarer Code oder variable Daten, die an einem externen eingebetteten Mikroprozessor liegen, eine Speicherung in einer externen seriellen Speichereinrichtung, I/O-Funktionen von einer externen seriellen I/O-Einrichtung, etc.) aufzulösen. Dies ist nur illustrativ für eine Entwicklung eines Systems in Übereinstimmung mit dieser Erfindung.

[0019] Ferner ist die vorliegende Erfindung auf mehr als einen Benutzerschnittstellenprozessor gerichtet. Sie ist zusätzlich auf die Verwendung der Hypertext Markup Language gerichtet, um einen Programmfluss und eine Struktur bereitzustellen, während Ressourcen, die unter eingebetteten Untersystemen verteilt sind, zusammen verbunden werden, sogar dann, wenn die Untersysteme Benutzerschnittstellen nicht aufweisen. D.h., die vorliegende Erfindung betrachtet in einem breiteren Aspekt einen speziell ausgelegten Prozessor, der mit einer Hypertext Markup Language anstelle einem herkömmlichen Anwendungscode programmiert ist.

KURZBESCHREIBUNG DER FIGUREN

[0020] In den Zeichnungen zeigen:

[0021] **Fig.** 1 ein herkömmliches eingebettetes Steuersystem für eine gesteuerte Einrichtung;

[0022] **Fig.** 2 ein eingebettetes Steuersystem in Übereinstimmung mit dieser Erfindung;

[0023] **Fig.** 3 ein ausführlicheres Diagramm des Markup Language Prozessors der **Fig.** 2;

[0024] **Fig.** 4 eine HTML-Datei und einen zugehörigen Anforderungsbehandler in Übereinstimmung mit der Erfindung; und

[0025] **Fig.** 5 den Zusammenhang zwischen der HTML-Quellendatei der **Fig.** 4 und einer Version, die auf µHTML compiliert ist.

AUSFÜHRLICHE BESCHREIBUNG

[0026] **Fig.** 2 zeigt ein Blockdiagramm eines Steuersystems für eine gesteuerte Einrichtung in Übereinstimmung mit dieser Erfindung. Blöcke, die ähnlich zu denjenigen der **Fig.** 1 sind, tragen identische Bezugszeichen. In **Fig.** 2 ist der Anzeige-Controller **24** der **Fig.** 1 durch einen zweiten Hypertext-Prozessor **40** ersetzt, der (nicht notwendigerweise) eine einzelne integrierte Schaltung sein kann und der eine intelligente Einrichtung ist, im Gegensatz zu dem Anzeige-Controller **24**. Somit gibt es in der Struktur der **Fig.** 2 zwei intelligente Einrichtungen (Prozessoren), wobei einer davon der Hypertext-Prozessor **40** ist und der zweite davon z.B. der eingebettete Mikroprozessor (oder eine andere Einrichtung), von denen mehrere mit **42a** etc. bezeichnet gezeigt sind, ist. Der Hypertext-Prozessor **40** ist sowohl mit der Benutzereingabeeinrichtung **14** als auch den Anzeigeelementen **20**, **22** gekoppelt. Jede vernetzte Einrichtung, wie **42c** oder **42d**, die einen Speicher für das Benutzerschnittstellen-(Hypertext)-Dokument enthält, kann das Benutzerschnittstellendokument an dem Markup Language Prozessor **40** servieren (bereitstellen). Irgendeine vernetzte I/O-Einrichtung wie **42a**, **42b**

oder **42d**, die mit einer gesteuerten oder überwachten Einrichtung **29** arbeitet, kann Ressourcen aufweisen, auf die von dem Benutzerschnittstellendokument (den Benutzerschnittstellendokumenten) verwiesen wird. "Vernetzt" bedeutet hier eine Einrichtungsverbindbarkeit (connectivity) unter Verwendung von Standardprotokollen. Sie umfasst sowohl eine "Intra-Produkt"-Vernetzung (wobei mehrere Einrichtungen innerhalb eines Gehäuses verbunden sind) als auch eine "Inter-Produkt"-Vernetzung (bei der Einrichtungen, jede in ihrem eigenen Gehäuse, verbunden werden).

[0027] **Fig.** 2 zeigt mehrere Typen von Einrichtungen, die optional von einem herkömmlichen Netz **4**, **6** mit dem Markup-Prozessor verbunden sind. Diese verbundenen Einrichtungen umfassen einen eingebetteten Mikrocontroller **42a**, eine serielle I/O-(Eingabe/Ausgabe)-Einrichtung **42b**, eine µHTML-Speichereinrichtung **42c**, und einen eingebetteten Mikrocontroller-GUI-Server **42d** mit seinem eigenen µHTML-Speicher. Natürlich sind andere Verbindungsanordnungen mit irgendeiner Anzahl oder einer Kombination von Einrichtungen oder Netzen, die mit dem Markup Language Prozessor **40** verbunden sind, möglich, solange wie wenigstens eine Einrichtung, z.B. **42c**, vorhanden ist, die das (die) µHTML-Dokument (Dokumente) speichern kann. Weil ein einzelnes µHTML-Dokument Verbindungen zu den Ressourcen von verschiedenen Einrichtungen auf dem Netz enthalten kann, ist es auch nicht erforderlich, dass jede Einrichtung auf dem Netz **46** einen Speicher für µHTML-Dokumente enthält.

[0028] Obwohl **Fig.** 2 nur eine gesteuerte Einrichtung **29** zeigt, die mit einer Vielzahl von Einrichtungen verbunden ist, kann/können ein oder mehrere derartige gesteuerte Einrichtungen, die von einer oder mehreren der vernetzten I/O-Einrichtungen **42a** etc. gesteuert (oder überwacht) werden können, vorhanden sein. Zusätzlich können die vernetzten I/O **42a** etc. Einrichtungen in dem gleichen physikalischen Gehäuse angeordnet sein oder nicht. Z.B. können die Komponenten eines Mikrowellenofens in dem gleichen physikalischen Gehäuse vernetzt sein. Jedoch können die Komponenten eines Heimunterhaltungssystems (z.B. Umgebungsschall-Empfänger/Verstärker, VCR, CD/DVD-Spieler) alle mit einem Hypertext-Prozessor, z.B. in einem Fernsehgerät, vernetzt sein, aber in ihren eigenen physikalischen Gehäusen untergebracht sein.

[0029] Während die verschiedenen Blöcke **30**, **40**, **20**, **22** und **42a**, **42b** etc. der **Fig.** 2 in einer Ausführungsform getrennte integrierte Schaltungen sind, kann die Aufteilung unter den verschiedenen integrierten Schaltungen auch anders sein, z.B. kann das System der gesamten **Fig.** 2 auf einer einzelnen integrierten Schaltung mit der möglichen Ausnahme der Benutzereingabeeinrichtung **14**, der gesteuerten Einrichtung **29** und der Anzeige **20** sein. Die Aufteilung der angezeigten Blöcke unter verschiedenen integrierten Schaltungen ist für diese Erfindung nicht kritisch.

[0030] Nachstehend wird jeder funktionale Block des Hypertext-Prozessors **40** der **Fig.** 2 beschrieben.

[0031] Der Netz-Controller **58** formatiert und überträgt sämtliche Bytes von Daten, die von dem µHTML-Prozessor **60** in eine Warteschlange eingereiht werden, über das Netz **46**. Er decodiert auch irgendwelche Daten, die von dem Netz **46** empfangen werden, und legt sie in einer Warteschlange ab, um von dem µHTML-Prozessor **60** verarbeitet zu werden.

[0032] Der Benutzereingabe-Decoder **62** erfasst und decodiert eine Eingabe von der Benutzereingabe-Einrichtung **14**, die z.B. eine Tastatur, ein Berührungsbildschirm, ein Sprachbefehls-Decoder oder eine IR (Infrarot) Fernsteuerung ist. Der Decoder **62** legt Daten, die ein Benutzereingabe-Ereignis beschreiben; in einer Warteschlange ab, um von dem µHTML-Prozessor **60** verarbeitet zu werden.

[0033] Der µHTML-Prozessor **60** arbeitet mit Daten, die in dem µHTML-Puffer **64** gespeichert sind, um Ereignisse zu reflektieren, die von dem Benutzereingabe-Decoder **62** und dem Netz-Controller **58** in eine Warteschlange eingereiht werden. Der Prozessor **60** ist auch dafür verantwortlich, um Ereignisse für den Netz-Controller **58** im Ansprechen auf System- oder Benutzer-Ereignisse zu erzeugen und diese in einer Warteschlange einzureihen, wobei die System- oder Benutzer-Ereignisse mit derartigen Ereignissen durch die Daten in dem µHTML-Puffer **64** verbunden sind.

[0034] Der µHTML-Puffer **64** ist ein RAM (Speicher mit wahlfreiem Zugriff) Speicher für ein gesamtes µHTML-Dokument, das sämtliche Objekte beschreibt, die an die Anzeigeeinrichtung **20** gegeben werden sollen. Jedes Objekt, das in dem µHTML-Dokument enthalten ist, kann auch Referenzen auf andere Netzressourcen enthalten. Der Puffer **64** wird nur beschrieben und modifiziert von dem µHTML-Prozessor **60** im Ansprechen auf Benutzereingabe-Ereignisse, Systemereignisse oder Ereignisse, die im Ansprechen auf Netznachrichten erzeugt werden. Er wird sowohl von der Hervorbringungsmaschine **52** als auch dem µHTML-Prozessor **60** gelesen. Der µHTML-Puffer **64** ist ein Abschnitt eines RAMs **72**, auf das nur der Mikroprozessor **68** zugreifen kann (siehe **Fig.** 3).

[0035] Die Hervorbringungsmaschine **52** liest nur die grafische Information für jedes UI-Objekt, wie erforderlich, um die Benutzerschnittstelle an den Bildpuffer **30** (Rahmenpuffer) in geeigneter Weise zu ziehen bzw. zu zeichnen. Der µHTML-Prozessor **60** liest die Information, die benötigt wird, um System- oder Netz-Ereignisse zu erzeugen, im Ansprechen auf andere Ereignisse, die mit jedem UI-Objekt in Beziehung stehen.

[0036] Die Hervorbringungsmaschine **52** zieht bzw. zeichnet sämtliche anzeigefähigen Benutzerschnittstellenobjekte an den Bildpuffer **30**, wie von den Daten beschrieben, die in dem µHTML-Puffer **64** ge-

speichert sind. Sie führt eine Wiederauffrischung jedes UI-Objekts durch, wenn es in dem µHTML-Puffer **64** durch den µHTML-Prozessor **60** als "schmutzig" ("unvollständig") markiert ist. Die Hervorbringungsmaschine **52** ist Firmware, die von dem Mikroprozessor **68** ausgeführt und in dem ROM **70** gespeichert ist (siehe **Fig.** 3). Jedes µHTML-Objekt enthält einen Code, um sämtliche Ansichten des Objekts hervorzubringen.

[0037] Der Bildpuffer (Rahmenpuffer) **30** ist ein RAM-Speicher, der die Daten für jedes Pixel der gesamten angezeigten Seite enthält. Er wird von der Hervorbringungsmaschine **52** beschrieben, wenn sie die Benutzerschnittstelle auf die Anzeige **20** zieht bzw. zeichnet. Er wird von dem Pixel-Serialisierer **36** gelesen, wenn er die Pixelinformation in Signale umwandelt, die geeignet sind, um die physikalische Anzeige **20** anzusteuern. Der Bildpuffer **30** der **Fig.** 2 ist ein Abschnitt des RAMs **72** (siehe **Fig.** 3), auf das der Mikroprozessor **68** (siehe **Fig.** 3) und der Pixel-Serialisierer **36** zugreifen können.

[0038] Der Pixel-Serialisierer **36** erzeugt einen kontinuierlichen Pixelstrom in einem Format, das mit einer spezifischen kommerziell erhältlichen physikalischen Anzeige **20** kompatibel ist. Wenn mit einem LCD-Feld (einer Anzeige **20**) gekoppelt, sammelt und formatiert der Pixel-Serialisierer z.B. jede Zeile von Pixeldaten von dem Bildpuffer **30** und synchronisiert diese zu dem herkömmlichen Anzeigetreiber-Pixeltakt, dem Rahmenimpuls und den Zeilenimpuls-Signalen. Das Pixeltaktsignal taktet die Pixeldaten in das interne Schieberegister der Anzeigetreiber. Das Zeilenimpulssignal zeigt das Ende einer Anzeigezeile an, während das Rahmenimpulssignal die erste Zeile der angezeigten Seite markiert.

[0039] Die Struktur der **Fig.** 2 erlaubt in einer vorteilhaften Weise die Verwendung von kommerziell erhältlichen Internet-Webseiten-Autorisierungswerkzeugen (wie HTML), um eine "Ziehen und Fallenlassen" ("drag and drop") grafische Benutzerschnittstellen-Autorisierung zur Entwicklung von Mikroprozessor-gestützten eingebetteten Systemen zu verwenden. Ferner ermöglicht sie eine einfache und konsistente serielle Schnittstelle über den Netz-Controller **58** zu Einrichtungen **42a**, **42b** etc., unabhängig von der Konfiguration der Anzeige **20**. Mit anderen Worten, die Intelligenz für eine Steuerung der Anzeige **20** ist in dem Prozessor **40** vorgesehen und muss nicht in der Software des eingebetteten Mikroprozessors **42a** codiert werden.

[0040] Dies beseitigt die herkömmliche Programmierung, z.B. in Assembler oder C, die benötigt wird, um grafische Benutzerschnittstellen-Objekte zu implementieren, die mit den Variablen und Funktionen des eingebetteten Mikroprozessors **10** verbunden sind, so wie dies in dem herkömmlichen System der **Fig.** 1 benötigt wird. Dies erlaubt auch die Entwicklung des Programmflusses durch keine mit Software vertrauten Ingenieure, die typischerweise die Anwendung für die gesteuerte Einrichtung **29** der **Fig.** 2 spezifizieren und dadurch die Anwendung und die Benutzerwechselwirkung verstehen, aber vielleicht nicht Firmware programmieren. Dies erlaubt eine schnellere und genauere Programmentwicklung, während erfahrene Firmware-Entwickler davon befreit werden, sich auf das technische Programm zu konzentrieren, und auch eine bessere Aufteilung eines Entwicklungsprojekts in kleinere, besser behandelbare Stücke zu ergeben, die parallel entwickelt werden können.

[0041] **Fig.** 3 zeigt ein "Hardware"-orientiertes Blockdiagramm des Hypertext-Prozessors **40** der **Fig.** 2. Der Prozessor **40** ist mit einer der eingebetteten Einrichtungen **42a** etc. verbunden. In diesem Fall ist die Protokollmaschine **58** der **Fig.** 2 als eine in eine Warteschlange eingereihte serielle Schnittstelle **58'** gezeigt, die z.B. eine UART/SPI/I^2C-Schnittstelle ist. Diese sind Beispiele von Industriestandard-Schnittstellen, die für die voranstehend beschriebene "Intra-Produkt"-Vernetzung geeignet sind. Eine SPI (serielle Peripherie-Schnittstelle; Serial Peripheral Interface) ist ein beliebtes synchrones serielles Kommunikationsschema zum Vernetzen von integrierten Schaltungen, die in eingebetteten Systemen enthalten sind. Sie wurde von Motorola entwickelt und von MAXIM, Harris, SanDisk, und anderen populär gemacht. Sie wird von vielen Mikrocontrollern und seriellen I/O-Einrichtungen, wie A/D und D/A-Wandlern, Solenoidtreibern, digitalen Potentiometern, Echtzeituhren, EEPROM, FLASH ROM, unter anderem, unterstützt. Der I^2C-Bus (Inter-IC Bus) ist eine andere beliebte synchrone serielle Netzarchitektur, die von Philips populär gemacht wurde, und ist einfacher, aber langsamer als die SPI. Wie die SPI sind viele serielle I/O und Speicher-Funktionen verfügbar. Jedoch sind weitaus mehr Verbraucherprodukt-Funktionen verfügbar, d.h. Fernseh- und Stereo-Baublöcke. Beispiele von geeigneten Schnittstellen für die Protokollmaschine **58'** für "Inter-Produkt"-Netze sind IEEE-1394, USB oder Ethernet. In Verbindung mit der geeigneten Firmware, die von dem Mikroprozessor **68** ausgeführt und im ROM **70** gespeichert ist, behandelt die Protokollmaschine **58** Interrupts (Unterbrechungen), die von den verbundenen Einrichtungen erzeugt werden, und behandelt Warteschlangen.

[0042] Der Benutzereingabe-Decoder **62** ist in **Fig.** 3 als ein Tastatur-Scan-Decoder **62'** gezeigt, der mit einer Tastatur **14** verbunden ist. In Verbindung mit der geeigneten Firmware, die von dem Mikroprozessor **68** ausgeführt und in dem ROM (nur-Lese-Speicher) **70** gespeichert ist, bedient der Decoder **62** Interrupts, die von den verbundenen Einrichtungen erzeugt werden, und behandelt Warteschlangen. Die übrigen Blöcke in **Fig.** 3 unterstützen die anderen Funktionen des Markup Language Prozessors **40** der **Fig.** 2. Dies wird im Hinblick auf die Schaltungsanordnung durch den Mikroprozessor "Kern" **68** erreicht (d.h. den Mikroprozessor ohne den Unterstützungsspeicher etc.), der wiederum mit einem Standardbus

76 verbunden ist, der eine Kopplung, wie gezeigt, mit den anderen Blöcken innerhalb des Prozessors **40** vornimmt. Typischerweise würde der gesamte Prozessor **40** der **Fig.** 3 eine einzelne integrierte Schaltung sein.

[0043] Der µHTML-Prozessor **60** der **Fig.** 2 in **Fig.** 3 ist Firmware, die von dem Mikroprozessor **68** ausgeführt und in dem ROM **70** gespeichert ist. Zusätzlich zu Routinen zum Bedienen von Interrupts, zum Behandeln von Ereignissen und zum Verwalten von RAM **72** gestützten Warteschlangen **78** und Puffern enthält dieser auch eine Bibliothek von Routinen, die in Übereinstimmung mit den spezifischen Datenstrukturen jedes µHTML-Objekts arbeiten. Diese Objekte können Benutzerschnittstellen-Objekte, Datenverarbeitungsobjekte und Betriebssystem-Objekte enthalten, sind aber nicht darauf beschränkt. Die Daten für jedes Exemplar eines Objekts sind in dem µHTML-Dokument enthalten, das in dem RAM **72** Gebiet gepuffert ist, das als der µHTML-Puffer **64** bezeichnet wird. Jedes µHTML-Objekt in der Bibliothek **84** in dem ROM **70** enthält einen Code, um (1) auf die Daten, die das Exemplar des Objekts definieren (von dem µHTML-Puffer **64**) zuzugreifen und diese zu modifizieren, (2) sämtliche Ansichten des Objekts an dem RAM-Bildpuffer **30** hervorzubringen, (3) auf Ereignisse zu reagieren, die sich auf das Objekt beziehen, und (4) Nachrichten in eine Warteschlange einzureihen, die an andere Netzressourcen gesendet werden sollen.

[0044] Die Strukturen in **Fig.** 3 umfassen (in dem ROM **70**) einen Hauptprogrammspeicher **88** und Ereignis-Behandler **90** und (in dem RAM **72**) einen Stapel **96** und einen Haufen (Heap) **98**. Der Pixel-Serialisierer **36** der **Fig.** 2 ist als Hardware (eine Schaltungsanordnung) in **Fig.** 3 dargestellt.

[0045] Die Blockdiagramme der **Fig.** 2 und 3 beschreiben einen Bereich von Strukturen, die in verschiedenen Kombinationen von speziell vorgesehener Hardware (einer Schaltungsanordnung) und Software (Computercode), die von verschiedenen Typen von Prozessoren ausgeführt wird, implementiert werden können. Die besondere Aufteilung zwischen Hardware und Software, die hier offenbart ist, ist nicht dafür vorgesehen, um beschränkend zu sein.

[0046] **Fig.** 4 zeigt ein Beispiel einer Anwendung, die in Übereinstimmung mit dieser Erfindung verwendet wird. Insbesondere ist der zentrale Abschnitt der **Fig.** 4, der der Text **86** ist, eine HTML-Datei, die ein Hypertext Markup Language Dokument ist, welches Anzeigeeinzelheiten einer LCD-Anzeige **88** mit Ressourcen eines eingebetteten Mikroprozessors verbindet. Die verschiedenen Textzeilen in **86** enthalten entweder: (1) Text, der angezeigt werden soll, wie "Zwei Variablen" oder "LED 0", oder (2) Markup Tags (enthaltend zwischen < und >), um auf GUI-Objekt-Bibliothekskomponenten zu verweisen und diese mit Ressourcen, die extern zu dem HTML-Dokument sind, und dem Markup Language Prozessor zu verbinden. In diesem Beispiel wird auf Ressourcen des eingebetteten Mikroprozessors über das Softwareprogramm **92** des eingebetteten Mikroprozessors zugegriffen.

[0047] Die vorhandenen Ressourcen des eingebetteten Mikroprozessors, auf die von dem Programm **92** zugegriffen wird, sind: zwei Variablen, die in diesem Fall die Werte **123** und **321** enthalten, und zwei Funktionen, die in diesem Fall eine LED einschalten und eine LED, die an dem eingebetteten Mikroprozessor angebracht ist, ausschalten. Die Variablen werden über IntField Objekte angezeigt und ein Zugriff auf diese erfolgt durch Senden der Befehle in dem <PARAM Name = "Send"...> Tag. Auf das Empfangen des Befehls zum HOLEN (GET) einer Variablen führt der eingebettete Mikroprozessor den Code in **92** aus, um die Variablen nachzuschlagen und den Wert zurück über die ackClient Routine zu senden. Das IntField der GUI-Objektbibliothek der Markup Language Prozessoren sieht die Antwort nach dem <PARAM Name = "Return".. > Tag durch, um den Wert zu isolieren und zu formatieren und diesen an dem Bildpuffer der LCD bereitzustellen.

[0048] Genauso werden die Funktionen, auf die von dem <PARAM Name = "Send".. > verwiesen wird, in Betrieb genommen, wenn der Benutzer die Tasten aktiviert, die von den FunctBtn Objekten hervorgebracht werden.

[0049] Zu diesem Dokument **86** gehört ein eingebetteter Anforderungs-Behandler **92**, der in dem rechten Abschnitt der **Fig.** 4 mit Leitungen gezeigt ist, die ihn mit dem Markup in dem Dokument **86** in Beziehung setzen. Dieser Behandler **92** ist in einem eingebetteten Mikroprozessor, wie beispielsweise unter Bezugnahme auf **Fig.** 2 **42a** oder **42d**, resident, um einen Zugriff auf die Ressourcen bereitzustellen, die über das Netz angefordert werden. Dieser Code in **92** kann in Hardware, z.B. in seriellen Speichereinrichtungen, wie unter Bezugnahme auf **Fig.** 2 **42c**, oder in seriellen I/O-Einrichtungen, wie **42b**, implementiert werden. Der "Client" in dem Code **92** ist eine Referenz (ein Verweis) auf den Markup Language Prozessor **40**. Während die Daten, die von dem Dokument **86** beschrieben werden, tatsächlich von dem Markup Language Prozessor **40** interpretiert werden, wird somit der Code **92** tatsächlich von dem eingebetteten Mikroprozessor **42a** in Verbindung damit ausgeführt.

[0050] **Fig.** 5 zeigt eine Wiederholung der HTML-Quellendatei (linke Seite) **86** der **Fig.** 4 mit einer compilierten µHTML-Version davon (rechte Seite). Die compilierte µHTML-Version ist viel kompakter; die Linien setzen den Quellendateicode mit seiner compilierten Version in Beziehung. Zusätzlich lässt sich die µHTML zur Ablaufzeit (Runtime) einfacher interpretieren, weil Dinge wie Kettenlängen, Tag-Versätze, X-Y-Koordinaten von dem Compiler berechnet und in die Struktur des Dokuments eingebaut werden. Natürlich besteht keine Notwendigkeit, HTML oder µHTML zu verwenden oder diese zu compilieren. Jedoch stellt dies eine Effektivität beim Ausführen einer Ausführungsform der vorliegenden

Erfindung dar.

[0051] Alternativen zur Verwendung der µHTML, die hier offenbart sind, sind andere Formen von Textdokumenten mit Steuercodes, die verwendet werden, um auf Ressourcen zuzugreifen, die an anderer Stelle angeordnet sind. Beispiele von anderen Markup-Sprachen sind Kompakt HTML und HDML. Sogar das alte UNIX "troff' ist eine Markup Language, die ursprünglich für ein Seitenlayout ausgelegt worden ist.

[0052] Speichereinrichtungen (wie **42c**) (**Fig.** 2), die zu dem Prozessor **40** extern sind, sind dadurch verantwortlich, um die µHTML und andere Dateien als "Host aufzunehmen" (ein "Hosting" vorzunehmen). Unabhängig davon, ob die externe Einrichtung ein anderer Mikroprozessor **42d** oder einfach eine serielle Speichereinrichtung **42c** ist, reagiert sie auf Aufforderungen von dem Prozessor **40**, um Dateien zu lesen oder diese zu beschreiben. Zusätzlich können Einrichtungen **42a** etc., die mit dem Prozessor **40** verbunden sind, auch Anforderungen unterstützen, um Variablen zu lesen/zu beschreiben, Funktionen in Betrieb zu nehmen und Zustandsinformation bereitzustellen, während die normale I/O-Einrichtungsfunktionalität ausgeführt wird.

[0053] Die eingebettete Speichereinrichtung **42c** ist dadurch verantwortlich, um die µHTML und andere Dateien als "Rost aufzunehmen". Sie reagiert auf Aufforderungen von dem Hypertext-Prozessor und verfolgt Änderungen an Variablen bei der Verwendung durch den Hypertext-Prozessor und führt die Funktionalität der gesteuerten Einrichtung aus. Der Hypertext-Prozessor ist verantwortlich, um die grafische Benutzerschnittstelle an der Anzeige hervorzubringen. Der Hypertext-Prozessor ist auch dafür verantwortlich, um auf eine Benutzereingabe von der Benutzereingabeeinrichtung zu reagieren, indem Anzeigegrafiken aktualisiert werden und mit externen Einrichtungen kommuniziert werden, um Änderungen an den Werten von externen Variablen anzufordern und externe Funktionen, wie von dem µHTML-Dokument beschrieben, in Betrieb zu nehmen. Der Hypertext-Prozessor ist auch dafür verantwortlich, um auf Änderungen in den Variablen des eingebetteten Mikroprozessors zu reagieren, indem die Anzeigeeinrichtungs-Grafik aktualisiert wird. Typische Anforderungen an den eingebetteten Mikroprozessor durch den Hypertext-Prozessor sind: Verbindung öffnen; Datei holen (z.B. eine µHTML-Datei, eine Bildgrafikdatei oder ein Script); Aufrufen von Funktionen, Holen eines Werts der Variablen; Senden eines Werts der Variablen und Ermitteln eines Status des eingebetteten Mikroprozessors. "Script" bezieht sich hier auf Dateien, die einen Code enthalten, der von dem Mikroprozessorabschnitt des Hypertext-Prozessors ausgeführt werden soll.

[0054] Diese Offenbarung ist illustrativ und nicht einschränkend; weitere Modifikationen werden im Hinblick auf diese Offenbarung Durchschnittsfachleuten nahe liegen und es ist beabsichtigt, dass sie in den Umfang der beigefügten Ansprüche fallen.

Patentansprüche

1. Eingebettetes Steuersystem, um eine zugehörige Einrichtung (**29**) zu steuern oder zu überwachen, wobei die zugehörige Einrichtung einen Eingangs/Ausgangs-Abschnitt (**40**, **20**) einschließt, wobei das eingebettete Steuersystem umfasst:

einen ersten Prozessor (**60**), der über eine Eingangs/Ausgangs-Schaltungsanordnung (**62**, **36**) mit dem Eingangs/Ausgangsabschnitt zum Steuern eines Betriebs der zugehörigen Einrichtung gekoppelt ist; und

einen Speicher (**42c**, **42d**), der zu dem ersten Prozessor gehört, wobei der Speicher ein oder mehrere Dokumente speichert, die miteinander verbunden sind, aber nicht einen ausführbaren Code enthalten, um Verfahren zu beschreiben, um die zugehörige Einrichtung durch Verbinden von Funktionen, die durch den ersten Prozessor ausführbar und lokal zu diesem sind, mit Beschreibungen der Wechselwirkungen, die zum Zugreifen auf die Eingangs/Ausgangs-Schaltungsanordnug benötigt werden, zu steuern oder zu überwachen;

wobei der erste Prozessor (**60**) das gespeicherte Dokument von dem Speicher (**42c**, **42d**) empfängt und die Steuerungs- oder Überwachungsverfahren der gespeicherten Dokumente durch Ausführen der Funktionen, auf die von den gespeicherten Dokumenten verwiesen wird, ausführt, um auf der Eingangs/Ausgangs-Schaltungsanordnung (**62**, **36**) verbunden mit den Funktionen durch die gespeicherten Dokumente zu arbeiten, um dadurch einen Betrieb des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen;

einen zweiten Prozessor (**42a**, **42b**, **42d**), der zwischen den ersten Prozessor (**60**) und die zugehörige Einrichtung (**29**) gekoppelt ist, wobei der zweite Prozessor interne Ressourcen einschließt, um ein Steuern und Zugreifen auf die zugehörige Einrichtung zu erleichtern; und

wobei die internen Ressourcen mit Funktionen, die von dem ersten Prozessor (**60**) ausführbar sind, über Verbindungen von den gespeicherten Dokumenten verbunden sind, um dadurch einen Betrieb des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen.

2. System nach Anspruch 1, wobei der erste Prozessor (**60**) einen Speicher (**64**) einschließt, um eine Kopie des gespeicherten Dokuments zu speichern, während es die verbundenen Funktionen ausführt; und das gespeicherte Dokument eine initialisierte Datenstruktur aufweist, die die variablen Daten für jede Funktion, die damit verbunden ist, darstellt, um dadurch Platz in dem lokalen Speicher des ersten Prozessors für jedes Exemplar der Funktionen, die von dem gespeicherten Dokument verbunden werden, zuzuordnen.

3. System nach Anspruch 1 oder 2, wobei der Eingangs/Ausgangs-Abschnitt eine Treiber-Schaltungsanordnung (**22**) und eine Anzeige (**20**), die mit der Treiber-Schaltungsanordnung gekoppelt ist, einschließt.

4. System nach irgendeinem vorangehenden Anspruch, wobei ein zugehöriger Speicher mit dem zweiten Prozessor (**42d**) gekoppelt ist.

5. System nach Anspruch 4, wobei die Kopplung über ein Netz (**46**, **58**) stattfindet.

6. System nach Anspruch 5, wobei das Netz ein Inter-Produkt- oder ein Intra-Produkt-Netz ist.

7. System nach irgendeinem vorangehenden Anspruch, wobei die Funktionen, die durch den ersten Prozessor ausführbar und lokal zu diesem sind, Objekte zum Steuern oder Überwachen der Eingangs/Ausgangs-Schaltungsanordnung, die mit dem Eingangs/Ausgangs-Abschnitt gekoppelt ist, sind, um dadurch Funktionen der zugehörigen Einrichtung zu steuern.

8. System nach Anspruch 7, wobei die Objekte Objekte einer graphischen Benutzerschnittstelle einschließen.

9. System nach irgendeinem vorangehenden Anspruch, ferner einschließend gespeicherte Dokumente, die nicht einen ausführbaren Code enthalten, sondern Verbindungen zu anderen gespeicherten Dokumenten enthalten, die einen ausführbaren Code enthalten.

10. System nach irgendeinem vorangehenden Anspruch, wobei der Eingangs/Ausgangs-Abschnitt eine Benutzereingabedecodierungs-Schaltungsanordnung (**62**), die mit einer Benutzereingangseinrichtung (**14**) gekoppelt ist, einschließt.

11. System nach irgendeinem vorangehenden Anspruch, wobei das gespeicherte Dokument ein Hypertext-Markup-Language-Dokument ist.

12. System nach irgendeinem vorangehenden Anspruch, wobei das gespeicherte Dokument aus einem Hypertext-Markup-Language-Dokument durch Aufteilen der Interpretation des Dokuments in eine Kompilierungs-Zeitphase und eine Ablauf-Zeitphase kompiliert wird, wobei die Kompilierungs-Phase ein Zwischendokument erzeugt, das in dem Speicher gespeichert werden soll und von dem Prozessor während der Ablauf-Zeitphase ausgeführt werden soll, wodurch die Last an dem Prozessor zum Ausführen des gespeicherten Dokuments verringert wird.

13. System nach irgendeinem vorangehenden Anspruch, wobei der zweite Prozessor in einem Gehäuse untergebracht ist, das von einem Gehäuse, in dem der erste Prozessor untergebracht ist, getrennt ist.

14. Verfahren zum Verwenden eines eingebetteten Steuersystems, um eine zugehörige Einrichtung (**29**) zu steuern oder zu überwachen, wobei die zugehörige Einrichtung einen Eingangs/Ausgangs-Abschnitt (**14**, **20**) einschließt und das eingebettete Steuersystem einen ersten Prozessor (**60**), der über eine Eingangs/Ausgangs-Schaltungsanordnung (**62**, **36**) mit dem Eingangs/Ausgangs-Abschnitt zum Steuern eines Betriebs der zugehörigen Einrichtung gekoppelt ist, und einen Speicher (**42c**, **42d**), der zu dem ersten Prozessor gehört, einschließt, umfassend die folgenden Schritte:

Speichern von einem oder mehreren Dokumenten in dem Speicher verbunden miteinander, aber nicht einen ausführbaren Code enthaltend, um Verfahren zu beschreiben, um die zugehörige Einrichtung durch Verbinden von Funktionen, die von dem ersten Prozessor ausführbar und zu diesem lokal sind, mit Beschreibungen der Wechselwirkungen, die zum Zugreifen auf die Eingangs/Ausgangs-Schaltungsanordnung benötigt werden, zu steuern oder zu überwachen;

Empfangen des gespeicherten Dokuments von dem Speicher in dem ersten Prozessor;

Ausführen der Steuerungs- oder Überwachungs-Verfahren der gespeicherten Dokumente durch Ausführen der Funktionen, auf die von den gespeicherten Dokumenten verwiesen wird, um auf der Eingangs/Ausgangs-Schaltungsanordnung verbunden mit den Funktionen durch die gespeicherten Dokumente zu arbeiten, um dadurch Operationen des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen;

Bereitstellen eines zweiten Prozessors (**42a**, **42b**, **42d**);

Koppeln des zweiten Prozessors zwischen den ersten Prozessor und die zugehörige Einrichtung, wobei der zweite Prozessor interne Ressourcen einschließt, um eine Steuerung der zugehörigen Einrichtung oder einen Zugriff auf die zugehörige Einrichtung zu erleichtern; und

Verbinden der internen Ressourcen mit Funktionen, die von dem ersten Prozessor ausführbar sind, über Verbindungen von gespeicherten Dokumenten, um dadurch einen Betrieb des Eingangs/Ausgangs-Abschnitts zu steuern oder zu überwachen.

Es folgen 4 Blatt Zeichnungen

Anhängende Zeichnungen

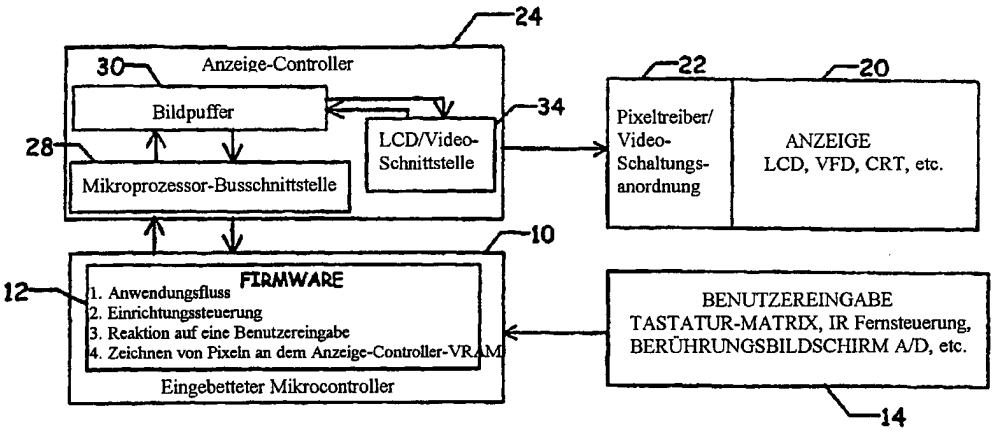

FIG. 1. (Stand der Technik)

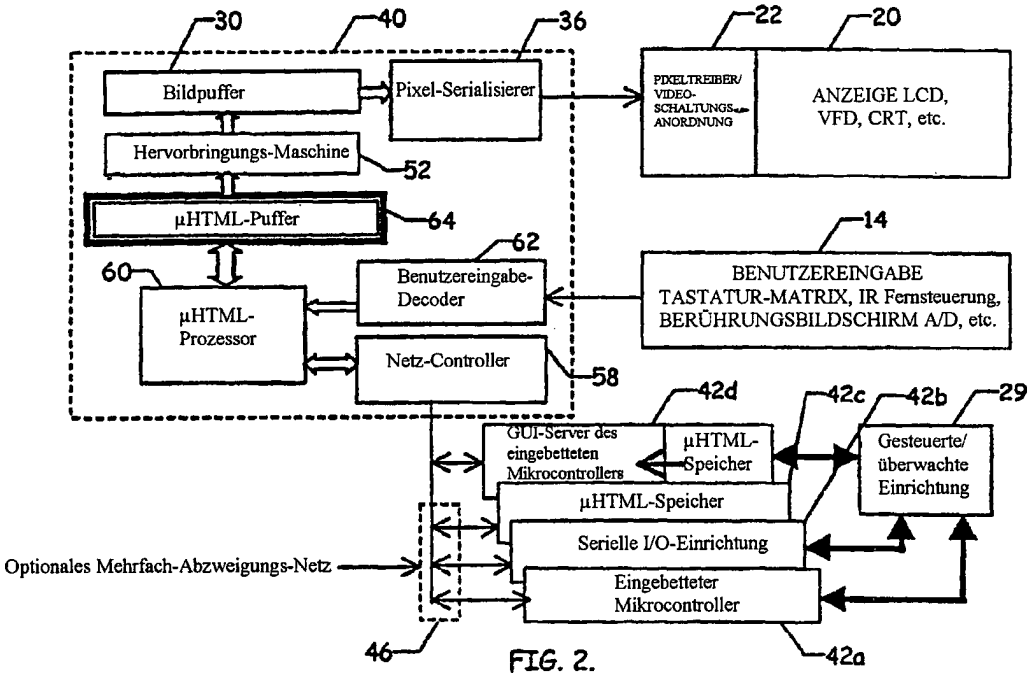

FIG. 2.

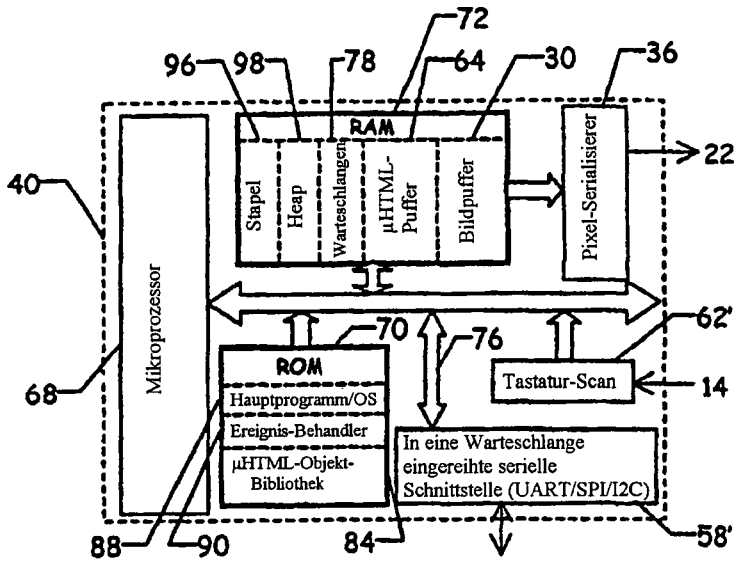
72
96
98
78
64
30
36
RAM
Stapel
Heap
Warteschlangen
µHTML-Puffer
Bildpuffer
Pixel-Serialisierer
22
40
Mikroprozessor
62'
70
76
ROM
Hauptprogramm/OS
Ereignis-Behandler
µHTML-Objekt-Bibliothek
Tastatur-Scan
14
68
In eine Warteschlange eingereihte serielle Schnittstelle (UART/SPI/I2C)
58'
88
90
84
42a

FIG. 3.

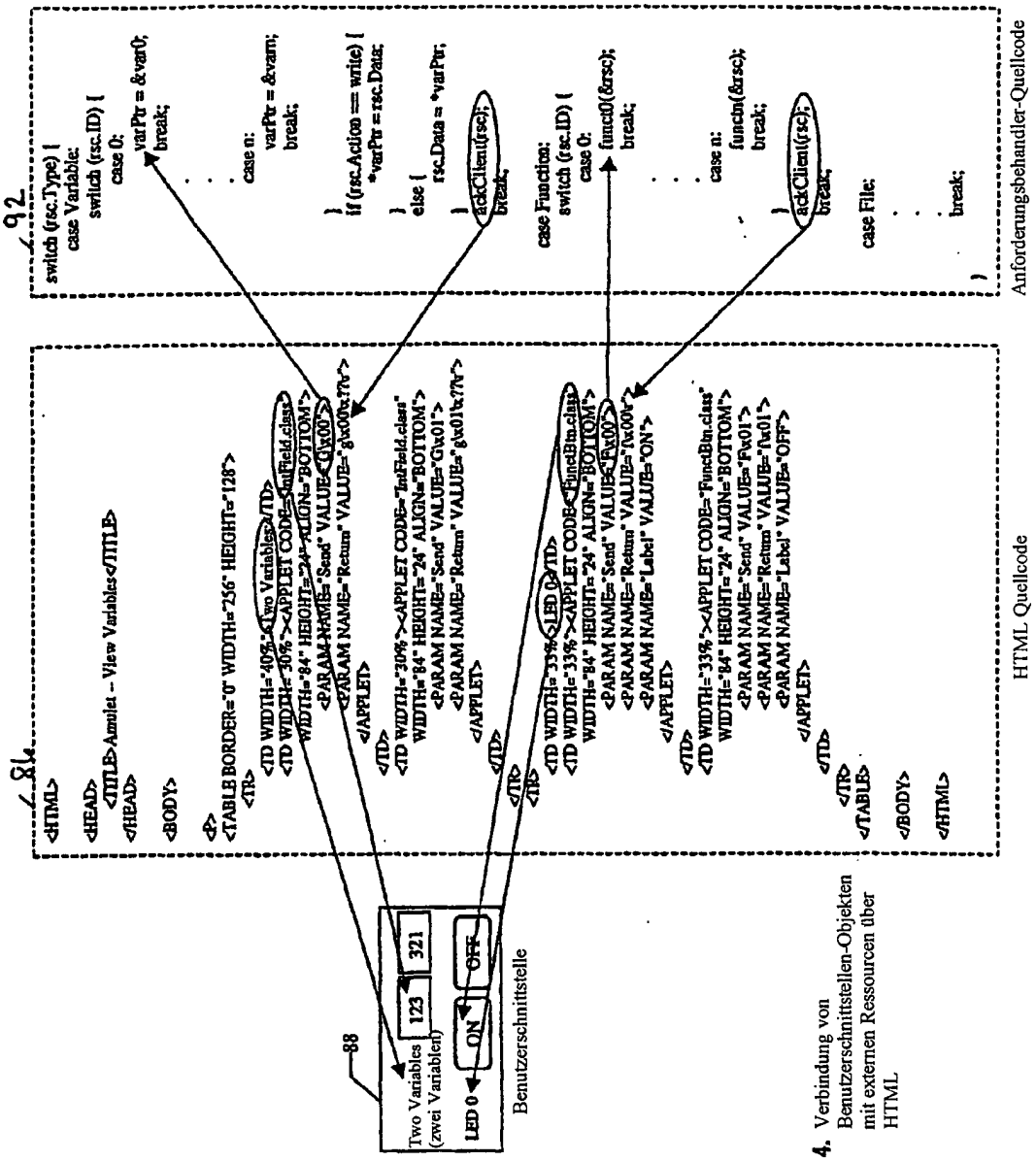

Fig. 4. Verbindung von Benutzerschnittstellen-Objekten mit externen Ressourcen über HTML

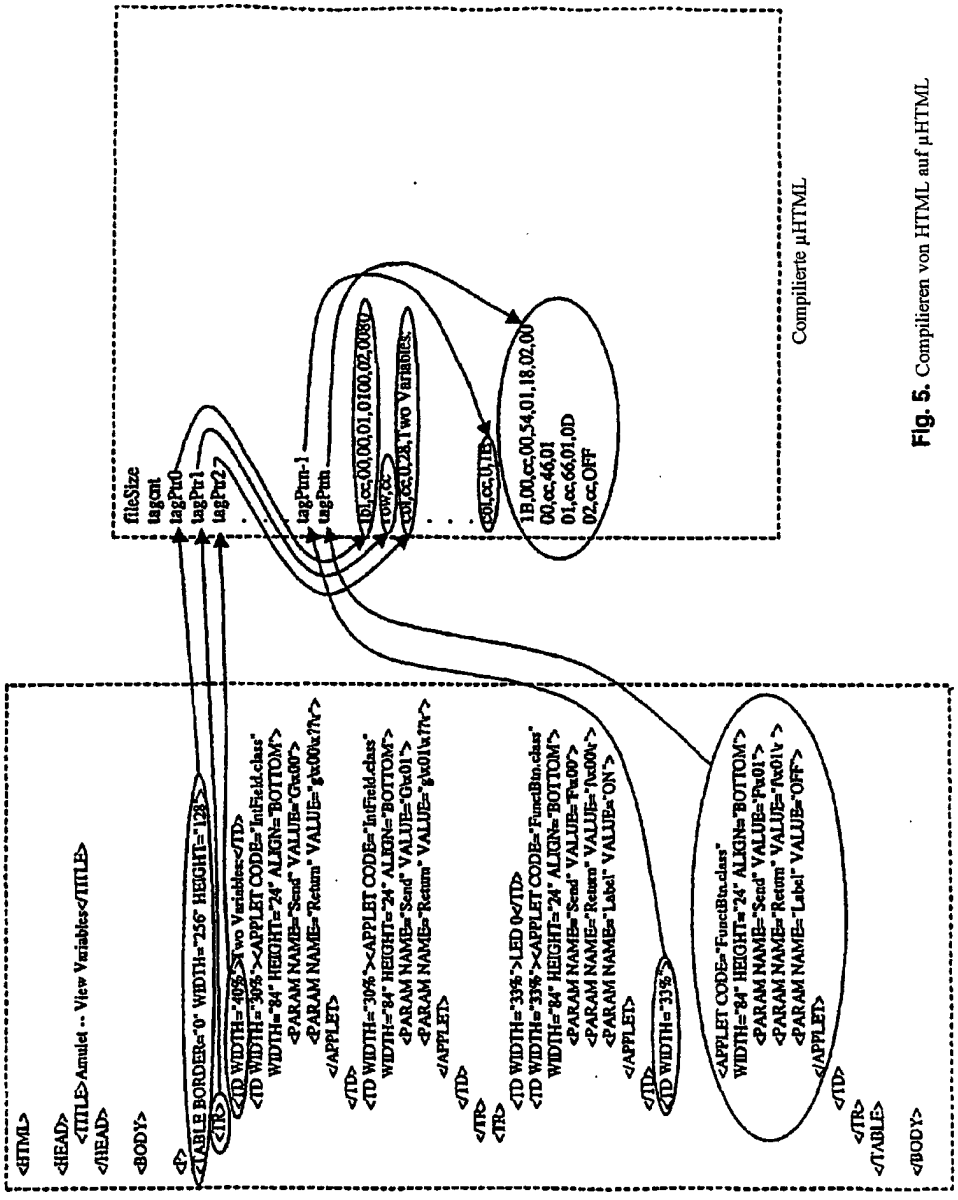

Fig. 5. Compilieren von HTML auf µHTML