EUROSplus

Webserver

Ausgabe März 1999

Ein Produkt eines Unternehmens der PHYTEC Technologie Holding AG

Informieren Sie sich:

	EUROPA	NORD AMERIKA
Adresse:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 255 Ericksen Avenue NE Bainbridge Island, WA 98110 USA
Angebots Hotline:	+49 (800) 0749832 order@phytec.de	+1 (800) 278-9913 order@phytec.com
Technische Hotline:	+49 (6131) 9221-31 support@phytec.de	+1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	+1 (206) 780-9135
Web Seite:	http://www.phytec.de	http://www.phytec.com

Inhalt

Kapitel 1
Einleitung

1.1 Allgemeine Funktionsweise

Der EUROS Web-Server implementiert einen HTTP-Server für eingebettete Systeme. Es werden die HTTP-Versionen 0.9 und 1.0 unterstützt. Der Server bearbeitet die HTTP-Methoden GET, HEAD und POST.

1.2 Voraussetzungen

Der Web-Server setzt das Betriebssystem EUROS und dessen TCP/IP-Stack voraus.

1.3 Files

Gelieferte Files:

`webserv.h`	Header-Datei für den Web-Server, Definition aller benötigten Strukturen.
`webserv.lib`	Bibliothek mit dem Server-Code

Kapitel 2
Aufruf des Web-Servers

2.1 Aufruf des Web-Servers

Der Web-Server muß aus einer EUROS-Task heraus aufgerufen werden. Der Server läuft dann in einer Endlosschleife im Kontext der aufrufenden Task ab. User-Funktionen, die der Web-Server bei dynamischen Seiten aufruft, laufen ebenfalls im Kontext dieser Task ab.

2.2 Aufruf

Die Server-Funktion ist wie folgt definiert:

```
int WebServMain(uint16 PortNum, int SendBufSize);
```

Der Parameter `PortNum` gibt die Port-Nummer an, unter der der Server Anfragen entgegennimmt. Die für HTTP übliche Port-Nummer ist 80.

Der Parameter `SendBufSize` gibt die Socket-Buffer-Größe für die Senderichtung an. Wenn 0 übergeben wird, wird die Default-Größe verwendet.

Die Funktion kehrt nur bei einem TCP/IP-Fehler zurück; der Rückgabewert ist dann `FAIL`.

Kapitel 3
Definition von Web-Seiten

3.1 Definition von Web-Seiten

Die Seiten, die der Web-Server anbieten kann, werden in einem Array von Strukturen `tWebDoc` beschrieben. Das Ende des Arrays wird durch einen besonderen Eintrag (s.u.) gekennzeichnet.

3.2 Definition einer einzelnen Seite

Eine einzelne Seite wird durch eine Struktur `tWebDoc` beschrieben. Die Komponenten dieser Struktur sind:

`pName`	Zeigt auf den Namen der Seite, z.B. `"index.htm"`. Der Server findet die vom Browser in der URL angegebene Seite anhand dieses Eintrags. Hierarchische Namen wie `"sub/index2.htm"` sind möglich. Ein `"/"` am Anfang darf *nicht* angegeben werden! Ein `NULL` in diesem Feld kennzeichnet das Ende des Arrays der Seitendefinitionen. Das Array *muß* einen solchen Eintrag am Ende des Arrays enthalten!
`pContentType`	Zeigt auf den MIME-Typ des Dokuments. Dieses Feld wird im HTTP-Header “Content-Type” an den Browser übertragen. Übliche Einträge sind `"text/html"` für HTML-Seiten oder `"image/gif"` für GIF-Bilder.
`pChunkTable`	Zeigt auf ein Array, das auf die Teilabschnitte des Seiteninhalts verweist (s.u.).
`Expires`	Wird derzeit nicht verwendet, bitte auf 0 setzen.
`LastModified`	Wird derzeit nicht verwendet, bitte auf 0 setzen.
`pPrepFunc`	Zeigt auf eine User-Funktion, die bei dynamischen Seiten aufgerufen wird. Bei statischen Seiten bitte auf `NULL` setzen.
`pPostFunc`	Zeigt auf eine User-Funktion, die bei Forms aufgerufen wird. Bei Seiten ohne Forms bitte auf `NULL` setzen.
`pAccessTable`	Zeigt auf eine Tabelle von Zugriffscodes. Bei allgemein zugänglichen Seiten bitte auf `NULL` setzen.
`pRealm`	Zeigt bei geschützten Seiten auf der Namen des “Protection Space”. Bei allgemein zugänglichen Seiten bitte auf `NULL` setzen.
`pCleanupFunc`	Zeigt auf eine User-Funktion, die bei dynamischen Seiten aufgerufen wird, und zwar nachdem der Server die Seite an den Client geschickt hat. Bei `NULL` in diesem Feld wird keine weitere Funktion aufgerufen.

3.3 Festlegung der Default-Seite

Wenn die Anfrage an den Server keinen Namen der Seite enthält (`http://servername/`), so sendet der Server seine Default-Seite. Diese wird durch den globalen Zeiger `pDefaultDoc` dem Server mitgeteilt. Dieser Zeiger zeigt auf das Array der Seiten-Beschreibungen. Die erste Seite im Array ist die Default-Seite. Der Server sucht auch alle anderen Seiten über diesen Zeiger. Die Anwendung *muß* diesen Zeiger definieren und belegen!

3.4 Festlegung des Seiteninhalts

Der Inhalt der Seite wird durch ein Array von `tChunk`-Strukturen beschrieben. Das Feld `pChunkTable` in `tWebDoc` zeigt auf ein solches Array. Das Ende des Arrays wird durch einen speziellen Eintrag gekennzeichnet.

Die Komponenten dieser Struktur sind:

`pData`	Zeigt auf den Datenblock des Abschnitts. Ein `NULL` in diesem Feld kennzeichnet das Ende des Arrays. Das Array *muß* einen solches Element am Ende enthalten!
`Length`	Anzahl der Datenbytes dieses Abschnittes. Falls dieses Feld 0 enthält, so überspringt der Server diesen Abschnitt. Auf diese Weise kann ein Abschnitt bei dyna-

mischen Seiten ausgeblendet werden.

Durch die Aufteilung einer Seite in mehrere Abschnitte wird die Erstellung von dynamischen Seiten vereinfacht. Außerdem ist es so möglich, auch größere Seiten im Server abzulegen. Drittens kann man Speicher für die Seiteninhalte sparen, indem man immer gleiche Abschnitte auf verschiedenen Seiten (z.B. Seitenköpfe mit Firmen/Produktlogos) nur einmal im Zielsystem ablegt und dann in verschiedenen `tChunk`-Arrays auf diesen einen Abschnitt verweist.

3.5 Tool TEXT2C

Mit dem Tool TEXT2C lassen sich HTML-Seiten (als ASCII-Files) in C-Quellcode umwandeln und so in die Anwendung einbinden. Der Aufruf ist:

```
TEXT2C <inputfile> <outputfile> <arrayname> <linelimit>
```

Die Aufrufparameter sind:

`inputfile`	Name der Eingabedatei (HTML-Text)
`outputfile`	Name der erzeugten Quellcode-Datei. Diese Datei wird ggf. neu angelegt und überschrieben.
`arrayname`	Basis-Name der erzeugten Arrays. Die einzelnen Abschnitte werden durchnumeriert.
`linelimit`	Maximalzahl der Zeilen aus der Eingabedatei, die in einen einzelnen Abschnitt umgewandelt werden. Nach dieser Anzahl von Zeilen wird ein neuer Abschnitt erzeugt.

TEXT2C erzeugt nicht nur die einzelnen Abschnitte der Seite, sondern auch das Verzeichnis der Abschnitte (`tChunk`-Array). Im `tWebDoc`-Array kann man sofort auf dieses Verzeichnis verweisen.

3.6 Tool BIN2C

Mit dem Tool BIN2C lassen sich Binär-Files (Bilder, Dateien, Java-Applets etc.) in C-Quellcode umwandeln und so in die Anwendung einbinden. Der Aufruf ist:

```
BIN2C <inputfile> <outputfile> <arrayname> <sizelimit>
```

Die Aufrufparameter sind:

`inputfile`	Name der Eingabedatei
`outputfile`	Name der erzeugten Quellcode-Datei. Diese Datei wird ggf. neu angelegt und überschrieben.
`arrayname`	Basis-Name der erzeugten Arrays. Die einzelnen Abschnitte werden durchnumeriert.
`sizelimit`	Maximalgröße (in Bytes) der einzelnen Abschnitte.

BIN2C erzeugt nicht nur die einzelnen Abschnitte der Seite, sondern auch das Verzeichnis der Abschnitte (`tChunk`-Array). Im `tWebDoc`-Array kann man sofort auf dieses Verzeichnis verweisen.

Kapitel 4
Dynamische Web-Seiten

4.1 Dynamische Web-Seiten

Dynamische Seiten sind solche Seiten, deren Inhalt sich bei jedem Abruf ändern kann. So können in einer dynamischen Seite z.B. Statusmeldungen des Server-Systems und dessen Anwendung abgelegt werden.

4.2 Eintragen der Vorbereitungs-Funktion

Um den Inhalt einer Seite veränderbar zu machen, ruft der Server bei jedem Abruf der Seite eine Funktion auf, die vom Anwender bereitgestellt wird. Die Funktion wird im Feld `pPrepFunc` der `tWebDoc`-Struktur eingetragen. Ein `NULL` in diesem Feld kennzeichnet dagegen eine statische Seite.

Eine solche Vorbereitungsfunktion muß die folgende Form haben:

```
int PrepFunc(uint32 Dest, tWebDoc *pDoc, char *pParam);
```

Der Server ruft die Vorbereitungsfunktion nach dem Analysieren der Anfrage auf. Nach der Rückkehr aus der Funktion sendet er (je nach Rückgabewert, s.u.) den nun veränderten Inhalt der Seite an den Browser.

4.3 Parameter der Funktion

Die Parameter der Funktion sind:

`Dest`	IP-Adresse der anfragenden Gegenseite (Browser).
`pDoc`	Zeiger auf die Definition der Seite, die angefragt wurde. Über diesen Zeiger kann man zum Inhalt der Seite gelangen. So könnte man die gleiche Funktion für verschiedene Seiten verwenden.
`pParam`	Zeiger auf die Parameter der Anfrage. Zeigt auf den Parameter-String, den der Browser bei der Anfrage mitgeschickt hat. Bei einer Anfrage `http://server/seite.htm?a=1&b=param` zeigt `pParam` auf den String `"a=1&b=param"`. Die Vorbereitungsfunktion kann diesen String auswerten. Wenn die Anfrage keine Parameter enthält, wird `NULL` übergeben.

4.4 Rückgabewerte

Der Rückgabewert der Funktion an den Server entscheidet, wie sich der Server weiter verhält:

`PREP_OK`	Der Server sendet den Inhalt der Seite an den Browser.
`PREP_BADREQ`	Der Server sendet die Fehlermeldung “400 Bad request” an den Browser. Dieser Rückgabewert sollte geliefert werden, wenn ein Fehler im Parameterstring gefunden wurde.
`PREP_ERROR`	Der Server sendet die Fehlermeldung “500 Server error” an den Browser. Dieser Rückgabewert sollte geliefert werden, wenn ein anderer Fehler bei der Aufbereitung der Seite aufgetreten ist.

4.5 Aufräumfunktion

Wenn die Vorbereitungsfunktion Resourcen (Speicher, Semaphore etc.) belegt und nicht sofort wieder freigegeben hat, so muß in `pCleanupFunc` eine weitere Funktion eingetragen werden. Diese wird vom Web-Server aufgerufen, nachdem das Dokument an den Client gesendet wurde, und zwar mit den gleichen Parametern wie die Vorbereitungsfunktion. In dieser Funktion können die Resourcen wieder freigegeben werden. Der Web-Server ruft die Funktion jedoch nur dann auf, wenn die Vorbereitungsfunktion `PREP_OK` geliefert hatte. Der Rückgabewert der Aufräumfunktion wird ignoriert.

4.6 Tips

Es ist auf vielfältige Weise möglich, den Inhalt der Seite zu ändern:

- Man kann die Inhalte der Speicherbereiche ändern, auf die das `tChunk`-Array verweist. So kann man dort Texte erzeugen oder verändern.

- Man kann das `tChunk`-Array ändern und somit einzelne Abschnitte der Seite komplett austauschen. So könnte man z.B. einen Statustext austauschen, indem man den Zeiger in einer `tChunk`-Struktur auf verschiedene bereits vorbereitete Texte zeigen läßt. Man kann einen Abschnitt auch komplett ausblenden, indem man das Feld `Length` in der `tChunk`-Struktur auf 0 setzt.
- Man kann komplette Seiteninhalte austauschen, indem man den Zeiger `pChunkTable` in der `tWebDoc`-Struktur ändert.

Welche Methode man anwendet, hängt stark davon ab, welche für die jeweilige Seite geeigneter erscheint. Man kann auch mehrere Methoden kombinieren.

Kapitel 5
Zugriffsschutz

5.1 Zugriffsschutz

Man kann einzelne Seiten vor unbefugtem Zugriff schützen. Beim Abruf der Seite fragt der Browser dann zunächst nach der User-ID und dem Passwort für diese Seite. Nur bei korrekter Eingabe wird der Seiteninhalt geschickt, ansonsten die Fehlermeldung “403 Unauthorized”.

5.2 Festlegung der Zugriffsparameter

Bei einer geschützten Seite müssen die Felder `pAccessTable` und `pRealm` in der `tWebDoc`-Struktur gesetzt werden. `pAccessTable` zeigt dabei auf auf ein Array von `char`-Zeigern. Die Elemente dieses Arrays zeigen auf Base-64-codierte User-ID/Passwort-Kombinationen. Ein `NULL`-Zeiger kennzeichnet das Ende des Arrays. Das Tool MKAUTH kann verwendet werden, um solche codierten Strings zu erzeugen (s.u.).

Das Feld `pRealm` zeigt auf dem Namen des Zugriffsraums dieser Seite. Dieser Name wird von manchen Browsern bei der Abfrage des Passworts angezeigt und hat sonst keine weitere Bedeutung.

5.3 Tool MKAUTH

Mit dem Tool MKAUTH erstellt aus einer ASCII-Datei mit User-ID/Passwort-Kombinationen im Klartext eine C-Quelldatei mit einem Array der codierten Passwörter. Auf dieses Array kann im Feld `pAccessTable` in der `tWebDoc`-Struktur verwiesen werden.

Der Aufruf des Tools ist:

```
MKAUTH <inputfile> <outputfile> <arrayname>
```

Die Aufrufparameter sind:

`inputfile`	Name der Eingabedatei. Die Datei enthält in einer Zeile genau eine User-ID/Passwort-Kombination im Format `userid:passwort`. Leerzeichen am Anfang und Ende der Zeile sind erlaubt, zählen dann aber als Bestandteil der User-ID bzw. des Passworts!
`outputfile`	Name der erzeugten C-Quelldatei
`arrayname`	Name des erzeugten Arrays

Kapitel 6
Forms

6.1 Forms

Der Web-Server unterstützt auch HTML-Forms zum Senden von Steuerungsdaten an das Zielsystem. Das HTML-Formular wird dazu in einer HTML-Seite untergebracht. Im HTML-Tag `<form>` wird festgelegt, mit welcher Methode die Eingabedaten an den Server geschickt werden. Beide Methoden werden nachfolgend beschrieben.

6.2 Methode GET

Bei der Methode GET werden die Eingabedaten als Parameter an die URL angehängt und die angegebene Seite mit diesen Parametern angefordert. Bei der Zielseite handelt es sich sinnvollerweise um eine dynamische Seite, die die Parameter dann auswertet.

Beispiel

Ein Formular enthält die Eingabefelder `geraet` und `format`. Die Zielseite ist `status.htm`. Wenn man nun die Felder mit `"kanal1"` und `"kurz"` füllt und das Formular abschickt, so wird die Seite `status.htm` angefordert, und die Vorbereitungsfunktion erhält als dritten Parameter einen Zeiger auf den String `"geraet=kanal1&format=kurz"`. Die Funktion kann nun den Inhalt der Seite entsprechend aufbereiten.

Die Methode GET eignet sich eher für das Anfragen von dynamischen Seiten mit Parametern als zur Übermittlung von Steuerungsdaten. Für letzteres sollte man die Methode POST verwenden.

6.3 Methode POST

Bei der Methode POST werden die Eingabedaten direkt an die angegebene Zielseite geschickt. Diese Zielseite muß lediglich existieren, aber sie muß keinen Inhalt haben. Sie ist dann nur zum Empfang der Eingabedaten vorgesehen. Der Browser bekommt dann keinen neuen Seiteninhalt. Die Methode POST eignet sich also zum reinen Übertragen von Steuerungsdaten. Das Format der Parameter ist das gleiche wie bei der Methode GET.

6.3.1 Eintragen der Post-Handler-Funktion

Wenn der Server eine Anfrage mit Methode POST bekommt, so ruft er eine User-Funktion auf, die die übermittelten Daten verarbeitet. Die Funktion wird vom Anwender bereitgestellt und im Feld `pPostFunc` der `tWebDoc`-Struktur eingetragen.

Eine solche Funktion muß die folgende Form haben:

```
int PostFunc(uint32 Dest, tWebDoc *pDoc, char *pParam);
```

6.3.2 Parameter der Funktion

Die Post-Handler-Funktion hat die folgenden Parameter:

`Dest`	IP-Adresse der anfragenden Gegenseite (Browser).
`pDoc`	Zeiger auf die Definition der Zielseite. Über diesen Zeiger kann man zum Inhalt der Seite gelangen. So könnte man die gleiche Funktion für verschiedene Seiten verwenden.
`pParam`	Zeiger auf die Parameter der Anfrage. Zeigt auf den Parameter-String, den der Browser bei der Anfrage mitgeschickt hat.

6.3.3 Rückgabewerte der Funktion

Mit dem Rückgabewert steuert die Funktion das weitere Verhalten des Servers. Die folgenden Werte sind erlaubt:

`POST_OK`	Die Anfrage wurde korrekt verarbeitet, und der Inhalt der Seite steht bereit. Der

	Server sendet dann den Inhalt der Seite an den Browser. Der Inhalt könnte z.B. eine Bestätigung der Eingabe sein. Falls die Seite lediglich zum Empfang der Eingabedaten existiert und keinen Inhalt haben soll, so muß stattdessen `POST_NOCONT` zurückgeliefert werden.
`POST_BADREQ`	Die Funktion hat einen Fehler in den Parametern entdeckt. Der Server sendet dann die Meldung “400 Bad request” an den Browser.
`POST_ERROR`	Die Funktion hat einen anderen Fehler entdeckt. Der Server sendet dann die Meldung “500 Server error” an den Browser.
`POST_CREATED`	Die Funktion hat eine neue Seite (also ein neues `tWebDoc`-Element) erzeugt. Wird derzeit nicht unterstützt.
`POST_NOCONT`	Die Eingabedaten wurden verarbeitet, aber die Seite soll keinen Inhalt haben. Der Server sendet die Meldung “204 No content” an den Browser.

Published by

SYS TEC

Ordering No. L-428d_1
Printed in Germany