FH-OÖ Hagenberg/ESD
Embedded Operating Systems, WS 2014
Florian Eibensteiner, Rainer Findenig, Josef Langer © 2014

6. Übung: Windows CE: Remote Controlling

Name(n): Punkte:

1. Aufgabe *Remote Controlling and Code Reuse*

Entwickeln Sie eine Applikation auf dem Colibri-Development-Board, die über TCP-Sockets mit einem PC kommunizieren kann. Auch hierzu haben Sie bereits in früheren Semestern Funktionen und Konzepte kennengelernt, die Sie hier wiederverwenden können.

Am embedded System soll eine Zusatzplatine angeschlossen werden, auf der sich vier LEDs sowie vier Schalter befinden. Vom PC aus sollen über eine einfache Menüführung die LEDs ein- und ausgeschaltet werden können sowie auf Befehl der aktuelle Zustand der Schalter und Taster ausgelesen werden.

Anmerkungen:

- Implementieren Sie ein einfaches, aber *erweiterbares* Protokoll (Stichwort *TLV*: *Type, Length, Value*).
- Müssen Sie bei der Kommunikation zwischen einem embedded System und einem PC ggf. die *byte order* beachten? □
- Es genügt, nur eine Verbindung zwischen dem embedded System und dem PC zuzulassen. Sie können den Server und den Client also *single-threaded* implementieren.
- Wenn der Client die Verbindung beendet, soll der Server wieder in den Listen-Modus wechseln und auf eine neue Verbindung warten.
- Das Colibri-Board bezieht die IP-Adresse, genau wie der PC, von einem DHCP-Server, sie ist also nicht fix. Implementieren Sie Ihr Programm am PC also so, dass sie die IP des Boards einstellen können.
- Im Gegensatz zu älteren Versionen, kann in Windows CE 6.0 von einer User-Applikation aus nicht direkt auf physikalische Adressen zugeriffen werden. Um auf die Hardware zugreifen zu können, muss zur Laufzeit ein entsprechender Treiber geladen werden. Dies kann durch Aufruf der Funktion *ActivateDeviceEx* gemacht werden, welche den Treiber lädt und die *Active Keys* in der Registry entsprechend verändert. Um den Treiber laden zu können, muss die zugehörige DLL (eos_driver.dll) vorhanden sein. Kopieren Sie entweder die Datei in den Windows Ordner am Target, oder konfigurieren Sie *Visual Studio* so, dass die Datei beim Debuggen automatisch auf das Target kopiert wird. Öffnen Sie hierfür die *Project Properties* und stellen Sie unter *Configuration Properties* → *Deployment* → *Additional Files* die benötigte Datei ein (Syntax: <filename>|<src_directory>|<dest_directory>|<register, 1 or 0>).
- Damit ein Treiber geladen werden kann, muss in der Registry im Pfad

HKEY_LOCAL_MACHINE\Drivers\Builtin ein entsprechender Eintrag vorhanden sein. Der Eintrag kann entweder von Hand getätigt, oder zur Laufzeit durch die Applikation geschrieben werden. Orientieren Sie sich dabei an dem Beispielprogramm aus der Übung und beachten Sie, dass der Pfad zur DLL des Treibers richtig angegeben wird.

- Wurde der Treiber erfolgreich geladen, kann mittels den Funktionen *CreateFileW*, *WriteFile* und *ReadFile* auf den Treiber bzw. die Hardware zugeriffen werden. Wenn Daten geschrieben werden, wird das untere Nibble des ersten Bytes an den LEDs ausgegeben, wobei folgendes Mapping vorliegt:

LED	**Bit**
LED1	Bit3
LED2	Bit2
LED3	Bit1
LED4	Bit0

  Wird von der Hardware gelesen, wird ein Byte zurückgegeben, welches die aktuellen Daten der Schalter und Taster beinhaltet. In diesem Fall liegt folgendes Mapping vor:

Schalter/Taster	**Bit**
Switch0	Bit0
Switch1	Bit1
Switch2	Bit2
Switch3	Bit3
Taster0	Bit4
Taster1	Bit5
Taster2	Bit6
Taster3	Bit7

- Wenn die Applikation beendet bzw. der Treiber nicht mehr benötigt wird, muss der Treiber mit der Funktion *DeactivateDevice* entladen werden.

- Ein Beispielprogramm für das Einbinden und benutzen des Treibers finden Sie in *eos-wince-2.zip*.

2. Aufgabe *Theorie*

In den weiteren Übungen werden Sie einen Treiber für Windows CE 6.0 entwickeln, der auf Register zugreift, indem zuerst Speicher *reserviert* (`VirtualAlloc`) und dann ein *Mapping* auf den physikalischen Speicher erstellt wird (`VirtualCopy`).

Beantworten Sie dazu die folgenden Fragen:

1. Erklären Sie den Unterschied zwischen physikalischen und virtuellen Adressen!

2. Was erreichen Sie durch den Aufruf von `VirtualAlloc`? Wird wirklich Speicher allokiert? Was ist der Unterschied zwischen den Flags `MEM_RESERVE` und `MEM_COMMIT`? Erstellen Sie eine Skizze, um Ihre Erklärungen zu verdeutlichen.

   Mehr Informationen zu diesem Thema können Sie [mem] entnehmen.

3. Erläutern Sie kurz den Unterschied zwischen *User Mode* und *Kernel Mode* in einer Win32-Umgebung. Könnten Sie in einer solchen Umgebung "'einfach so"' auf physikalischen Speicher zugreifen? Was ist in Windows CE 5.0 anders? Was wurde bei Windows CE 6.0 geändert? Welche Möglichkeiten bestehen dennoch Code im Kernel-Mode auszuführen?

   Mehr Informationen zu diesem Thema können Sie [ker] entnehmen.

References

[ker] Windows CE Base Team Blog – What is Kernel Mode? Website. Jan 2009, <http://blogs.msdn.com/ce_base/archive/2007/01/29/What-is-kernel-mode.aspx>.

[mem] Windows CE .NET Advanced Memory Management – MSDN Library – Windows Embedded Developer Center. Website. Jan 2009, <http://msdn.microsoft.com/en-us/library/ms836325.aspx>.