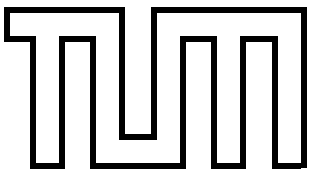


Technische Universität München

Fakultät für Informatik
Forschungs- und Lehreinheit Informatik VI

Übung zur Vorlesung Echtzeitsysteme

Einführung in VxWorks

Simon Barner
barner@in.tum.de

Dominik Sojer
sojer@in.tum.de

Stephan Sommer
sommerst@in.tum.de

Wintersemester 2010/11

1 Einführung zu VxWorks

In der Übung zur Vorlesung "Echtzeitsystem" wird das Echtzeitbetriebssystem *VxWorks* eingesetzt. VxWorks wurde von der *Firma Wind River Systems* (www.windriver.com) entwickelt. Das Betriebssystem ist für den Entwurf verteilter, zeitkritischer Anwendungen ausgelegt und bietet aus diesem Grund nur eine Eingabemöglichkeiten über die Kommandozeile und keine graphische Oberfläche. Um dennoch eine komfortable Entwicklung zu gewährleisten, erfolgt die Applikationsentwicklung auf einem Entwicklungsrechner (Host) mit einem Standardbetriebssystem (in unserem Fall Windows) und der Entwicklungsumgebung *Workbench*, die auf *Eclipse* basiert. Entwickelte Anwendungen können mit Hilfe dieses Werkzeugs auf den eigentlichen Zielrechner (*Target*) herunter geladen und dort ausgeführt werden. Zur Steuerung der Programmausführung ist eine so genannte *RemoteShel*l in der Workbench integriert. Selbstverständlich ist auch eine reine Ausführung auf dem Zielrechner möglich, jedoch für unsere Übung nicht sinnvoll.

Auf dem Hostrechner kann die Entwicklung der Echtzeitapplikationen (Codierung, Kompilieren, Debugging, Simulation, ...) komfortabel durchgeführt werden. Dazu wird mit der Workbench eine umfangreiche Entwicklungsumgebung bereitgestellt. Über sie lässt sich Analyse-Software wie z.B. der Debugger bedienen. Zum Testen der entwickelten Applikationen bieten sich dem Entwickler zwei Möglichkeiten: die Simulation der Anwendung im VxWorks-Simulator (ausgeführt auf dem Entwicklungsrechner) oder die Ausführung der Anwendung auf dem Zielrechner. Ein Nachteil beim Testen mit dem Simulator besteht darin, dass externe Ereignisse, z.B. Interruptanforderungen durch Peripheriekomponenten, simuliert werden müssen. Es ist also nicht ohne weiteres möglich, mit dem Simulator die volle Funktionsfähigkeit jedes beliebigen Realzeitprogramms zu testen.
Überträgt man hingegen das zu testende Programm auf einen Zielrechner, so steht die gesamte Hardware des Zielrechners mit all seinen Peripheriekomponenten zur Verfügung. Somit kann das Programm ohne größeren Aufwand getestet werden. Abbildung 1 gibt einen groben Überblick über verschiedene, wichtige Komponenten des Entwicklungssystems, auf die in den folgenden Kapiteln des öfteren Bezug genommen wird.

Ein Entwicklungsrechner kann mit mehreren Zielrechnern verbunden sein. Mit Hilfe der Workbench und von *Shells*, die auf dem Entwicklungsrechner oder den Zielrechnern gestartet werden, kann ein Benutzer interaktiv Einfluss auf die echtzeit-kritische VxWorks-Anwendung nehmen. Beispielsweise können durch Kommandos, die über Shells abgesetzt werden, Anwendungen vom Entwicklungs- auf einen Zielrechner übertragen, dort ausgeführt und auch wieder vom Zielrechner gelöscht werden. Des Weiteren ist es möglich, Komponenten des zu testenden Programms zu überwachen, um die gewonnenen Daten für spätere Auswertungen, z.B. für die Fehlersuche, heranzuziehen. Für die Kommunikation zwischen Entwicklungs- und Zielrechner sind der *Target-Server* und der *Target-Agent* zuständig. Der Target-Agent arbeitet unabhängig von der Applikation und von VxWorks.

Das Herz des Laufzeitsystems von VxWorks ist der *wind*-Microkernel. Er ist skalierbar; man kann also die Komponenten des Kernels je nach Bedarf zusammenstellen. Der Ker-

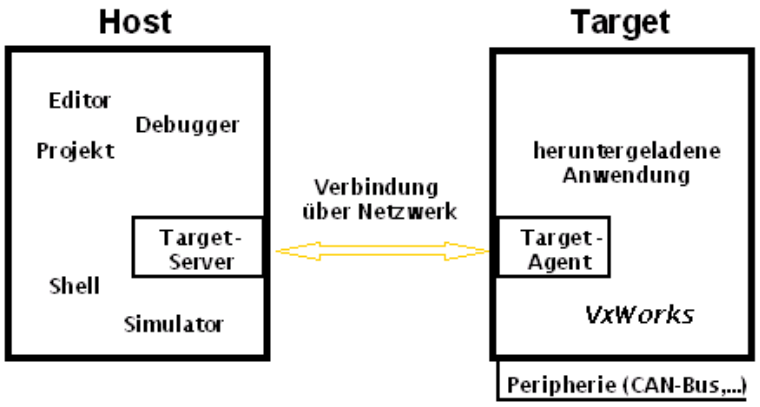


Abbildung 1: Überblick über einige Komponenten der Workbench und die Beziehung zwischen Host und Target

nel stellt Funktionen wie Speichermanagement, eine Multitasking-Umgebung, Zeitdienste, Interprozesskommunikations- und Synchronisations-Mechanismen zur Verfügung. Der Kernel ist immer Bestandteil eines VxWorks-Images, das nach dem Booten eines Zielrechners von Diskette automatisch über das Netzwerk auf den Rechner übertragen wird. Das VxWorks-Image liegt auf dem Rechner `atknoll133`. Für alle Zielrechner wird dasselbe Image verwendet. Es ist entsprechend der Aufgabenstellung so konfiguriert, dass die Funktionalität aller Zielrechner angesprochen werden kann. Spezielle Treiber, die in späteren Aufgaben einzubinden sind, werden mit den Aufgabenstellungen verteilt. Eine Änderung des Images ist im Rahmen der Übung nicht nötig.

Anleitungen und Hilfestellungen erhalten Sie durch Auswahl des Menüpunkts *Help* → *Help Contents* in der Entwicklungsumgebung Workbench

2 Workbench

Wie bereits erwähnt, basiert die Workbench auf dem Eclipse Framework und umfasst neben VxWorks weitere Komponenten. Auf einige von ihnen soll in diesem Kapitel näher eingegangen werden.

2.1 Projekte

Alle Arbeiten zur Erstellung von Anwendungen in VxWorks finden im Kontext von Projekten statt. Ihre Verwendung und mit Projekten verbundene Begriffe werden in diesem Abschnitt erläutert.

2.2 Arbeitsbereiche und Projekte

Ein Arbeitsbereich (*Workspace*) ist ein logischer Container für ein oder mehrere Projekte. Alle Projekte, die inhaltlich miteinander zu tun haben, sollen in einem Workspace abgelegt werden.

Achtung! Für jede Übungsgruppe existiert ein eigenes Verzeichnis auf unserem Server, wo sämtliche Daten abgelegt werden sollen. Der Zugriff auf das Netzlaufwerk geschieht über den Laufwerksbuchstaben `Z:`, der immer mit dem jeweiligen Gruppenordner verbunden ist.

Bitte überprüfen Sie unter *File* → *Switch Workspace* ob der aktuelle Workspace auf den Pfad `Z:` zeigt.

Abbildung 2 zeigt den Workspace mit verschiedenen Projekten. Die einzelnen Projekte heißen `Aufgabe0`, `Aufgabe1`, usw.

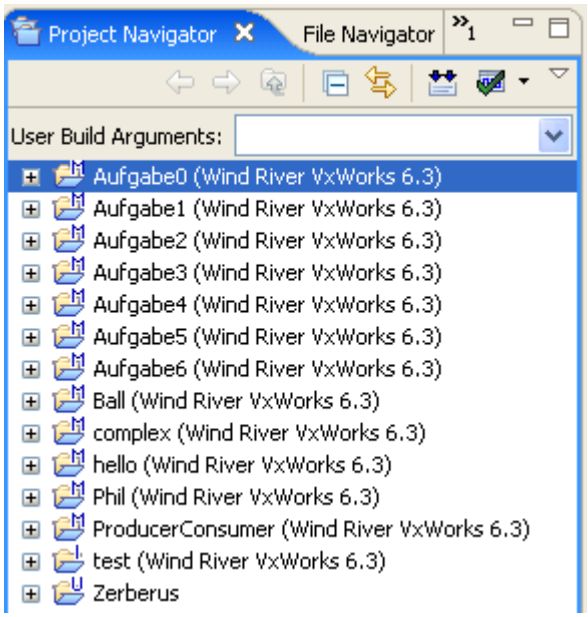


Abbildung 2: Workspace mit verschiedenen Projekten

Jede Anwendung für VxWorks wird einem Projekt zugeordnet. Jedes Projekt benötigt sein eigenes Verzeichnis. Dort liegen der Quellcode, der auf mehrere Dateien verteilt sein kann, sowie Informationen zur Verwaltung des Projekts. Um ein neues Projekt ins Leben zu rufen, klickt man auf *File* → *New* und entscheidet sich anschließend für ein *VxWorks Downloadable Kernel Module Project*.

Im Zusammenhang mit den Übungsaufgaben dürfen nur herunterladbare Anwendungen erstellt werden und niemals bootbare Images. Ein bootbares Image für alle Rechner liegt bereits auf dem Rechner `atknoll133` vor. Wie bereits erwähnt, wird es beim Booten auf den Zielrechner übertragen und Ihre Anwendung kann, nachdem sie ebenfalls auf den Rechner

übertragen wurde, auf das Image aufsetzen. Für weitere Erläuterungen im Manual wird nicht mehr zwischen Anwendung und Projekt unterschieden, weil davon ausgegangen wird, dass jedem Projekt nur eine Anwendung zugeordnet ist.

Im nächsten Schritt gibt man einen Namen an und wählt die Option *Create project in workspace* aus (Abbildung 3).

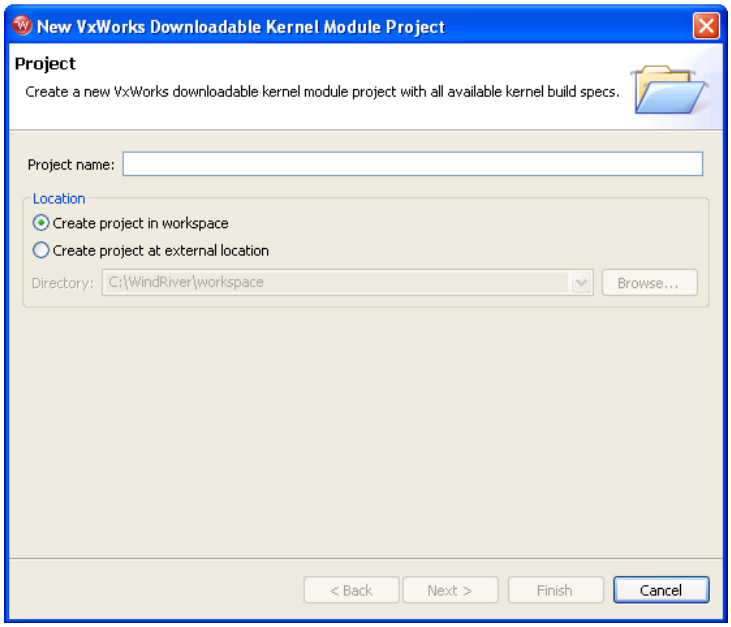


Abbildung 3: Neues Projekt

Mit der Schaltfläche *Next* werden die Dialoge weitergeschaltet, bis nachfolgender Dialog erscheint (Abbildung 4).

Nun muss eine Werkzeugkette (*Toolchain*) für das Projekt (Erklärung zur Werkzeugkette, siehe späterer Abschnitt) angeben werden. Es muss unter *Active build spec* die `PENTIUM-gnu`-Werkzeug-kette für das neue Projekt ausgewählt werden. Diese muss in der Übung immer dann ausgewählt werden, wenn ein Zielrechner zur Ausführung des Programms eingesetzt wird. Bei Verwendung des Simulators ist die Simulator-Werkzeugkette (`SIMNTgnu`) zu benutzen.

2.3 Werkzeugketten für Zielrechner und den Simulator

Eine Werkzeugkette ist eine Sammlung von Werkzeugen, die auf bestimmte Hardwarekonfigurationen zugeschnitten sind und erlaubt die Erstellung von Anwendungen für die Plattformen, sowie das Nutzen spezieller Hardwareeigenschaften. Beispielsweise werden durch eine `PENTIUM`-Werkzeugkette einer Anwendung die speziellen Vorteile, die ein bestimmter Pentium-Prozessors zu bieten hat, zugänglich. Sie brauchen sich nicht darum zu kümmern,

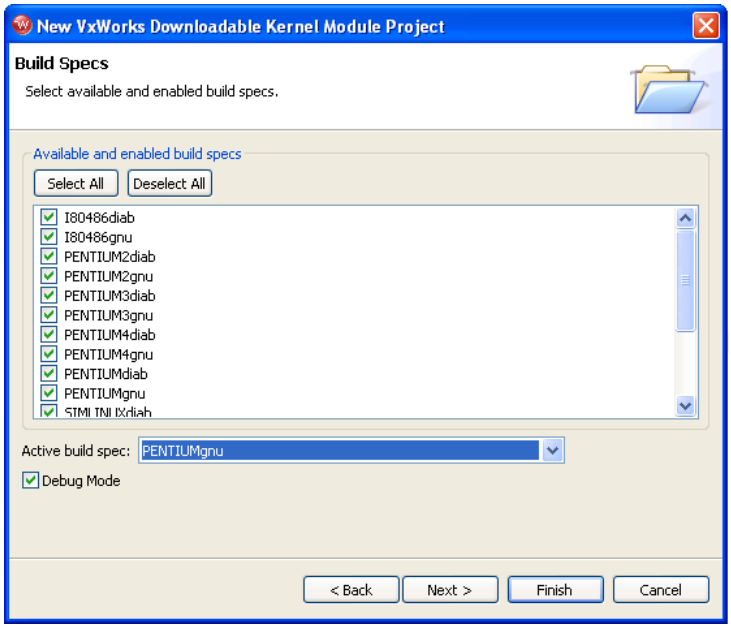


Abbildung 4: Auswahl der Toolchain

wie Teilen der Anwendung der Zugang zu den Eigenschaften der Hardware ermöglicht wird. Diese Aufgabe übernimmt die Werkzeugkette. In späteren Aufgaben muss die `PENTIUMgnu`-Werkzeugkette verwendet werden, weil das VxWorks-Image der Targetrechnern mit ihr übersetzt wurde.

Ein Simulator wird ebenfalls wie eine Zielplattform behandelt, obwohl er, wie der Name schon vermuten lässt, lediglich Hardware simuliert. Der Simulator läuft auf dem jeweiligen Entwicklungsrechner. Durch die `SIMNTgnu`-Werkzeugkette werden einer Anwendung die speziellen Eigenschaften des Simulators zugänglich gemacht. Weil der Simulator auf dem Entwicklungsrechner läuft, muss er sich die Ressourcen mit anderen Rechenprozessen teilen. Ein Simulator kann sich deshalb nicht in allen Punkten wie ein echter Zielrechner verhalten. Beispielsweise kann es zu Abweichungen bei Zeitmessungen kommen; ein Einfluss, der bei der Anwendungsausführung bzw. beim Testen berücksichtigt werden muss.

2.4 Herunterladbare Anwendungen

Nach einem Kalt- oder Warmstart fordert jeder Zielrechner selbständig ein VxWorks-Image vom Rechner `atknoll133` an (gleicht der Neuinstallation eines Betriebssystems). Nachdem das Image auf den Zielrechner übertragen wurde, kann eine (herunterladbare) Anwendung in das Image integriert werden (wie die Installation eines Programms). Die Anwendung lässt sich nun durch Eingabe der Startroutine, meist main, in eine Shell oder über den Debugger starten. Der Startroutine können Parameter getrennt durch Kommata übergeben werden. Ein Aufruf mit Parametern ist z.B.: `main par1, par2, ...` .

Hinweise zum Verwalten eines Projekts

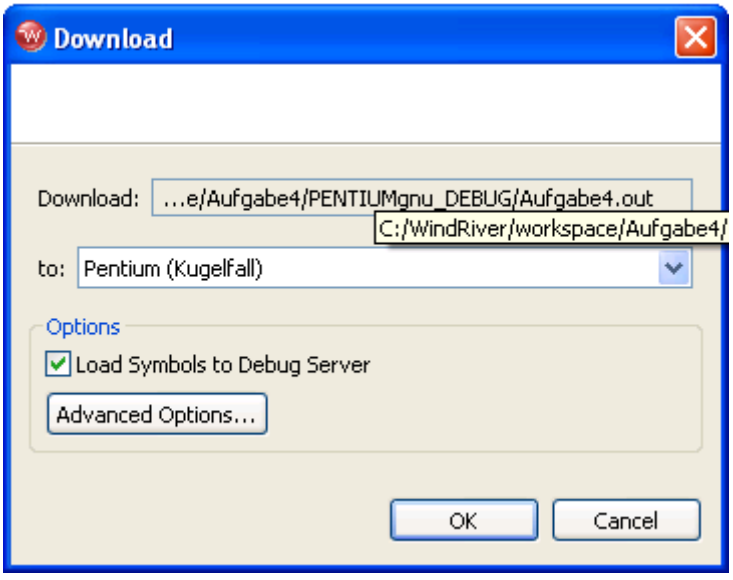
- Dateien einem Projekt hinzufügen: Um eine neue Datei zu erstellen, klickt man auf *File* → *new*, wählt den Typ der Datei, den Namen des Projektes, zu dem sie gehören soll, und gibt einen Namen und den Speicherort für die Datei an. Eine bereits existierende Datei kann zu einem Projekt per Kontextmenü hinzugefügt werden. Dazu öffnet man per Rechtsklick auf das Arbeitsbereichfenster das Menu und wählt im sich öffnenden Menü den Punkt *Import...*. Dann kann die hinzuzufügende Datei ausgewählt werden.

- Das Build-Kommando ausführen Um eine Datei eines Projekts bzw. das ganze Projekt zu kompilieren, kann im Menü *Project* der Menüpunkt *Build Project* bzw. *Rebuild Project* ausgewählt werden. Alternativ kann im Workspace-Fenster nach Rechtsklick auf das Projekt der Eintrag *Build Project* ausgewählt werden.

  Über das Menü *Project* → *Set Active Build Spec* kann ein und dieselbe Anwendung für verschiedene Zielrechner kompiliert werden. Beispielsweise kann ein Projekt, das letztendlich auf einem Pentium-Target laufen soll, zu Testzwecken für den Simulator kompiliert werden. Dazu selektiert man in o.g. Auswahldialog den Simulator (`SIMNTgnu`). Anschließend muss das Projekt noch neu kompiliert werden.

- Objektmodule auf einen Zielrechner herunterladen Um eine kompilierte Anwendung auf einen Zielrechner zu laden, muss zuerst eine Verbindung zum Zielrechner im *Target-Manager* aufgebaut werden. Dies geschieht durch das Anwählen des entsprechenden Rechners und dem Betätigen des Buttons

  Danach wird im Project Navigator das entsprechende Projekt ausgewählt und ein Rechtsklick auf `Projektname.out` ausgeführt. Nun wählt man *Download* und erhält folgenden Dialog, bei dem man den Zielrechner angeben muss.



Nach Klicken auf *OK* startet der Download-Vorgang. In beiden Fällen wird nach erfolgreicher Übertragung keine Rückmeldung ausgegeben.

Beachten Sie, dass ein Projekt, das mit einer Toolchain für den Simulator kompiliert wurde, nicht auf einem Pentium-Target ausgeführt werden kann und umgedreht. Sollten Sie mehrere Projekte gleichzeitig auf ein Target laden, können Namenskonflikte bei gleichlautenden Routinen und Variablen auftreten.
Beachten Sie: Erst beim Übertragen auf den Zielrechner werden die einzelnen Objektdateien endgültig verlinkt und entsprechende Fehlermeldungen bei fehlenden Funktionen erzeugt.

2.5 Shells

Wenn im Target-Manager eine Verbindung zu einem Zielrechner aufgebaut wurde, kann zu diesem Rechner eine *Host-Shell* (*WindShell*) durch Klicken auf das Symbol geöffnet werden. Es existieren neben Host-Shells auch *Target-Shells*. Eine Target-Shell wird normalerweise automatisch bei der Verbindung zu einem Zielrechner geöffnet. Man kann mehrere Host-Shells in Verwendung ha-ben, aber nur eine Target-Shell kann mit einem Zielrechner verbunden sein. Prinzipiell wird empfohlen, jegliche Shell-Kommunikation über Host-Shell abzuwickeln, weil deren Funktionsumfang größer ist und der Zielrechner durch die Verwendung von Host-Shells weniger beansprucht wird. Allerdings besitzen Host-Shells gegenüber der Target-Shell einen großen Nachteil. Die Ausgabe von `printf`-Anweisungen bei Programmen mit nebenläufigen Prozessen erfolgt beim Start auf der Host-Shell nur teilweise und häufig nicht in der richtigen Reihenfolge. Ist die richtige Reihenfolge nötig, so ist ein Start aus der Target-Shell nötig.

Shells bieten Zugang zu vielen Funktionalitäten des Betriebssystems. So können über sie global deklarierte C-Routinen aufgerufen werden sowie Variablen deklariert und initialisiert werden. Shells können zum Debugging einsetzen werden und natürlich ist der gesamte Funktionsumfang von VxWorks über Shells aufrufbar. Über Shells lassen sich neue Prozesse starten (`spawn`) und es kann ein Neustart des Rechners (`reboot`) veranlasst werden. Durch einen Neustart wird ein reines VxWorks-Image ohne Anwendungsdaten auf den Zielrechner übertragen, analog zu einer Neuinstallation des Betriebssystems.

VxWorks unterstützt auch eine Interpretierung von C-Code. So ist eine Eingabe von den meisten C-Ausdrücken direkt in der Eingabezeile möglich. Eine Shell liest dazu den Eingabestrom zeilenweise, parst die Zeilen, wertet sie aus und schickt ein Ergebnis an einen Ausgabestrom. Bis auf wenige Abweichungen wird dieselbe Syntax, wie sie von einem C-Compiler her bekannt ist, akzeptiert. Das System verarbeitet die Anweisungen exakt zu dem Ergebnis, dass während der Ausführung einer Anwendung erhalten werden würde. Die Möglichkeit, Anweisungen interpretativ auszuwerten, erspart das Kompilieren und Herunterladen. Ein Beispiel für den Routinen-Aufruf über Shells, ist der Aufruf der global deklarierten Hauptroutine Ihrer Anwendung.

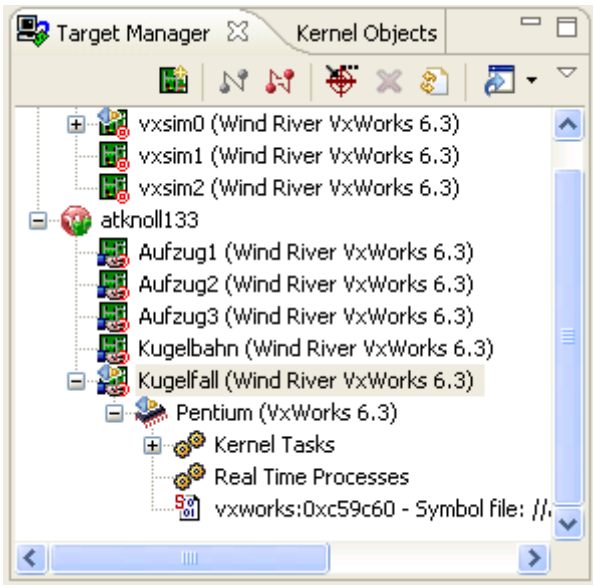
3 Die zentrale Windriver Registry

In späteren Aufgaben der Übung muss eine Verbindung zu einem der Zielrechner hergestellt werden.

Um auszuschließen, dass auf mehreren Entwicklungsrechnern gleichzeitig eine Verbindung zu dem selben Zielrechner aufgebaut wird, wird zur Verwaltung der Verbindungen zwischen Entwicklungs- und Zielrechnern eine zentrale Windriver Registry verwendet. Diese Registry befindet sich auf dem zentralen Übungsrechner `atknoll133`.

Achtung! Falls die Registry vom Rechner `atknoll133` noch nicht im Target-Manager aufgelistet wird, muss diese über die Menüauswahl *Target* → *New Registry* eingerichtet werden, indem dort der Name `atknoll133` eingetragen wird.

Eine gültige Host-Target-Verbindung wird nur aufgebaut, wenn eine Verbindung zu demselben Zielrechner auf dem Rechner `atknoll133` noch nicht registriert ist. Erst nachdem ein Entwicklungsrechner seine Verbindung mit dem Zielrechner abgebaut hat, kann ein anderer Entwicklungsrechner eine Verbindung mit dem Zielrechner aufbauen.



Eine aufgebaute Verbindung zu einem Target-Rechner erkennen Sie im Target-Manager (z.B., in Abbildung 3 der Versuch Kugelfall).

Es ist leider möglich, von jedem Entwicklungsrechner aus eine bestehende Verbindung aus dem Target-Manager auszuwählen und hierfür eine Host-Shell zu öffnen. Hierüber können dann Kommandos zum Zielrechner einer anderen Gruppe abgesetzt werden, was eher unangenehm ist und deshalb unterlassen werden sollte. Benutzen sie nur Verbindungen zu von ihnen genutzten Zielrechnern und geben sie diese sobald sie die Rechner nicht mehr benutzen auch wieder frei.