SPSSim DP-Referenz

Martin Kögler e9925248@stud4.tuwien.ac.at

9. Mai 2006

Inhaltsverzeichnis

1 Einleitung

Um mit der SPSSIM-DP-Schnittstelle zu arbeiten, braucht man:

- SPSSIM für Teil 2 ab der Version 31
- SPSSIM DP-Schnittstelle
- C / C++ Compiler (für Windows)

  Hier wird offiziell nur Microsoft Visual C++ 6.0 unterstützt, bei allen anderen kann mangels Testmöglichkeiten keine Hilfestellung garantiert werden.[1] Theoretisch sollte aber jeder Compiler funktionieren, der 32-Bit Programme für Windows erstellen kann.

2 HowTo

1. Wenn man die SPSSIM DP-Schnittstelle entpackt, sollten folgende Dateien enthalten sein:
   - 5613_ret.h
   - dp_5613.h
   - dpsize.cpp
   - size.lst
   - dplib.def
   - dplib.dll
2. 5613_ret.h und dp_5613.h sind die Headerdateien, die statt den Orginaldateien verwendet werden müssen, wenn man seine Übungsbeispiele mit SPSSIM lösen will. Die Verwendung der originalen Headerdateien funktioniert nicht und kann zu Abstürzen führen.
3. Wenn man einen anderen Compiler als Microsoft Visual C++ 6.0 verwendet, sollte man prüfen, ob er die Strukturen mit der richtigen Größe anlegt. Dazu kompiliert man dpsize.cpp als Consolenanwendung und startet es. Die Ausgabe sollte mit der in size.lst enthaltenen übereinstimmen. Wenn eine Zeile nicht übereinstimmt, muss man den Compiler so umstellen, dass er die Strukturen mit der vorgegebenen Größe anlegt.

[1] Installanleitungen und Benutzungshinweise für andere Compiler nehme ich gerne in dieses Dokument auf.

4. dplib.dll muss man statt der originalen Bibliothek ins Projekt einbinden. Dazu kann man eine Importbibliothek erzeugen (DEF-Datei dafür ist dplib.def).

   Dazu ruft man im bin-Verzeichnis von Microsoft Visual C++ 6.0

   ```
   lib /def:dplib.def
   ```

   aus. Die daraus erzeugte Datei (dplib.lib) muss man zum Projekt hinzufügen. Bei der Fehlermeldung, DLL für lib.exe nicht gefunden, muss man in einem Kommandozeilenfenster *vcvars32.bat* (ebenfalls in diesem Verzeichnis) starten, bevor man in diesem Fenster *lib.exe* ausführt. [2]

5. Beim Ausführen kopiert man am besten die DLL in jenes Verzeichnis, in dem die EXE-Datei erzeugt werden soll. Bei Microsoft Visual C++ ist es beispielsweise das Verzeichis *Debug*, das im Projektverzeichnis liegt.

6. Die Headerdateien *5613_ret.h* und *dp_5613.h* kopiert man am besten zu seinen Quelltexten. Damit müssen die Includepfade nicht geändert werden.

7. Es werden zur Zeit nur die in der Headerdatei definierten Funktionen unterstützt.

   Als Name für die Karte verwendet man wie im Labor *CP_L2_1:* .

8. Die DP-DLL braucht einen TCP-Port, auf den sie einen Server startet. Standardmäßig wird 5841 verwendet, man kann auch einen anderen Wert verwenden. Dazu übergibt man beim ersten Aufruf von *DP_start_cp* im Parameter database einen String, in dem die Portnummer steht.

   Im SPSSIM ruft man *Profibus DP/Verbinden* auf, um sich mit den Masterprogramm zu verbinden. Die Standardwerte sollten für die Übung genügen. Man kann aber auch die IP-Adresse und den Port eines anderen Rechner eintragen, auf dem das Masterprogramm läuft. Die Slaveadressen des Encoders und des CP242-8 können auch geändert werden, bzw. man kann auch mehrere SPSSIM Instanzen mit einem Masterprogramm verbinden, wenn die verwendeten Slaveadressen unterschiedlich sind.

[2] Bei den Borland Compilern wird das dazugehörige Hilfsprogramm unter den Namen *impdef* geführt

Mit *Profibus DP/Trennen* kann man die Verbindung wieder beenden. Ob ein Verbindung besteht, kann man überprüfen, indem man prüft, ob *Trennen* oder *Verbinden* im Menü *Profibus DP* anwählbar ist.

Eine Verbindung ist erst möglich, sobald im Masterprogramm *DP_start_cp* ausgeführt wurde. In späteren Versionen wird auch keine Verbindung mehr möglich sein, sobald *DP_reset_cp* ausgeführt wurde, bis wieder *DP_start_cp* ausgeführt wird.

Wenn man Desktop-Firewalls in Verwendung hat, muss man diese so konfiguieren, dass SPSSIM auf das Masterprogramm zugreifen darf.

3 Andere Compiler für Windows

Die folgenden Compiler, sind alle nicht offiziell unterstützt, sollten aber funktionieren:

- Microsoft Visual C++ 2005 Express Edition (`http://msdn.microsoft.com/vstudio/express/visualc/`)

  Microsoft stellt eine abgespeckte Version ihrer C++ IDE zum freien Download zur Verfügung (ca. 100 MB). Weiters werden noch Teile vom Plattform SDK benötigt. (Online Installation unter `http://www.microsoft.com/msdownload/platformsdk/sdkupdate/`. Es wird nur das *Build Enverionment für 32 Bit* benötigt, 59 MB). Die Integration vom Plattform SDK kann, wie auf der Microsoft Homepage angegeben, durchgeführt werden.

  dplib.lib erstellt man, indem man im *Visual Studio 2005 Command Prompt*

  ```
  PFAD\bin\link.exe /LIB /def:dplib.def
  ```

  ausführt (wobei dplib.def im aktuellen Verzeichnis sein muss).

  Als Projekt-Typ wird von SPSSIM nur *Win32 Console Application* unterstützt. Die Headerfile vom SPSSIM muss man in den Include Pfad aufnehmen. *dplib.lib* muss als zusätzliches beim Linker eingebunden werden.

- Microsoft Visual C++ Toolkit 2003 (`http://msdn.microsoft.com/visualc/vctoolkit2003/`)

  Microsoft stellt den aktuellen C++ Compiler als Komandozeilenversion zum freien Download zur Verfügung. Er läuft ab Windows 2000.

Man muss mit einer Downloadmenge von 100-500 MB rechnen, je nach installierten Komponenten.

Als Basis muss man 32 MB Downloaden, nach der Installation kann man aus den Arbeitsverzeichnis den Compiler aufrufen. Im Folgenden wird davon ausgegangen, das nach *PFAD* installiert wurde. (im Allgemeinen C:\Programme\Microsoft Visual C++ Toolkit 2003\). Wenn der Pfad ein Leerzeichen enthält, muss man ihn in Hochkommas stellen.

Zum Compilieren ruft man

```
PFAD\bin\cl.exe -I PFAD\include /c cpp-Datei
```

auf. Dabei sollte man jeweils im Arbeitsverzeichnis befinden, wo sich die zu kompilierenden Datein befinden. Weiters sollte auch in diesen Verzeichnis die DLL und die Headerdatein stehen.

Zum Linken ruft man

```
PFAD\bin\link.exe /LIBPATH:PFAD\lib /OUT:Ziel.exe obj-Dateien dplib.lib
```

auf. *dplib.lib* kann man analog wie bei Microsoft Visual C++ 6.0 erzeugen, nur ruft man statt *lib.exe*

```
PFAD\bin\link.exe /LIB /def:dplib.def
```

auf.

Das Make-Programm nmake findet sich .NET Framework SDK (verlinkt von `http://msdn.microsoft.com/netframework/`, etwas über 100MB). Man braucht aber im Prinzip keines, es reicht auch eine Batchdatei oder ein anderes Make-Programm, wie GNU-make oder man gibt die Befehle gleich in die Kommandozeile ein.

Weiters braucht man noch die Windows-Headerdatein, diese im Platform SDK enthalen sind. Der entsprechnede Punkt ist *Build environment* von *Core SDK (Windows Server 2003)*. Die Datenmenge ist bis 1 GB, wenn man eine Komplettinstalltion macht. (Online Installation unter
`http://www.microsoft.com/msdownload/platformsdk/sdkupdate/`).
Für die Übung reichen nur die Windows Header + Libraries für 32 Bit. Mit den Headerdatein von einen MS VC++ 6.0 hat es auch funktioniert Beim Compilieren muss man mit einen weiteren *-I Verzeichnis*,

den Pfad zu diesen Headerdateien angeben. (Beim Plattform SDK im Allgemeinen: C:\Programme\Microsoft Plattform SDK\include).

Dieser Compiler hält sich mehr an den C++ Standard, daher kann es Probleme bei der Übernahme von Code von/nach Microsoft Visual C++ 6.0 geben.

Weiters muss man die 32 Bit Build Environments verwenden, da 64 Bit momentan nicht unterstützt wird. Es sollte aber dabei kein Problem geben, wenn diese auf einen 64 Bit Prozessor verwendet werden.

- Cygwin Mingw (`http://www.cygwin.com`)

  Bei Cygwin gibt es für den gcc eine Extension (vgl `http://www.mingw.org`), mit der man native Windowsprogramme erzeugen kann. An Paketen wird man w32api und gcc-mingw-g++, gcc-mingw-core mit ihren Abhängigkeiten brauchen.

  Zum Kompilieren muss man einfach

  ```
  g++ -mno-cygwin -o Programm.exe Source1.cpp ... dplib.dll
  ```

  eingeben. An sich wird die C-Runtime Umgebung von Microsoft benutzt, man sollte aber seine Programme im Labor auch testen, da es trotzdem Unterschiede geben kann.

- Mingw (`http://www.mingw.org`)

  Die Mingw Seite selbst bietet auch eine Downloadmöglichkeit des Compilers ohne Cygwin.

- Microsoft Visual Studio .NET Student

  Eine aktulle Version von MS Visual Studio .NET gibt es als Studentenversion (vgl. `http://sts.tuwien.ac.at/sss.php`) im LMZ zu kaufen. Der Preis ist momentan (Mai 2006) 18 EUR.

  Die Programme sollten mit den vom *Microsoft Visual C++ Toolkit 2003* bzw. *Microsoft Visual C++ 2005 Express Edition* ident sein, es gehören nur Pfade angepasst.

4 Verwendung anderer Betriebsysteme

Obwohl SPSSIM für Windows geschrieben wurde, kann man es auch unter Linux mit wine SPSSIM sehr gut ablaufen lassen (ich empfehle eine möglichst aktuelle Version).

Für C-Programme kann man MinGW (`http://www.mingw.org/`) verwenden. Damit kann man gegen die DLL-Linken und per Wine das Programm ablaufen lassen.
Diese habe ich mit den Debian-Versionen 0.0.20040309-1 von wine und 3.2.1.20021201.3-1 von mingw überprüft.