knobloch
electronic

fischertechnik Robo Interface

&

"C"

Hinweise zur Verwendung des Renesas C-Compilers für das Robo-Interface

Version 1.66a
Stand: 15.05.2006

Version Interface Firmware: 1.66.0.3
Version Renesas High-performance Embedded Workshop: 4.00.03.001

Knobloch GmbH
Weedgasse 14
55234 Erbes-Büdesheim

entwicklung@knobloch-gmbh.de
www.knobloch-gmbh.de

1 Inhaltsverzeichnis

2 Vorbemerkung

Das Robo-Interface besitzt einen 16bit Mikroprozessor aus der M16C - Reihe des Herstellers Renesas (Typ: M30245). Dieser bietet eine komplette Entwicklungsumgebung unter dem Namen "High-performance Embedded Workshop" an. Da die Entwicklung dieser Profi-Werkzeuge sehr aufwändig, andererseits aber nur von einem "kleinen" Entwicklerkreis eingesetzt wird, sind Kosten beim Kauf für solch eine Version von 2000,00 EUR und mehr durchaus üblich.

Renesas hat nun mit der aktuellen Version 4 ein neues Lizenzmodell eingeführt. Für die Dauer von 60 Tagen ab der ersten Installation kann man das Entwicklerpaket uneingeschränkt nutzen. Danach erzeugt der Compiler nur noch Programme mit einer maximalen Größe von 64kByte. Diese Codegröße dürfte für die allermeisten fischertechnik Projekte mehr als ausreichend sein. Dadurch kann man ein Profi-Werkzeug mit sehr vielen Möglichkeiten kostenlos nutzen.

Aufgrund der Größe der Installationsdatei (ca. 80 MByte) ist er nicht in diesem Paket enthalten. Man kann die Software im Internet direkt von Renesas herunterladen. Falls man keinen geeigneten Internetzugang hat, kann man eine CD mit der Installationsdatei beim fischertechnik Einzelteilservice (vertrieb@knobloch-gmbh.de bzw. www.knobloch-gmbh.de) bestellen. Die CD kann zum Preis von 1,00 EUR zzgl. der üblichen Servicepauschale für die Bearbeitung und den Versand bezogen werden. (Artikelnummer der CD: 79028 Stand: Januar 2006).

Neben der Installationsdatei für die Entwicklungsumgebung wird das Datenpaket FtCComp.ZIP benötigt. In diesem Datenpaket befinden sich neben dieser Anleitung auch Beispielprogramme und ein Programm (FtLoader) um die vom C-Compiler erzeugte Programmdatei in das Interface zu speichern. Diese Datei ist auch auf der CD enthalten. Ferner sind auf der CD auch die FtLib enthalten.

Die Beispielprogramme sollen als Grundlage für eigene Projekte dienen. Mit den vorliegenden Informationen ist es dem Programmierer mit C-Grundkenntnissen möglich, C-Programme zu erstellen und diese in das Interface zu speichern. Hier lauern jedoch auch einige Gefahren, die wir im Kapitel "Maschinennahe Programmierung" ansprechen.

3 Robo-Interface & "C"

In das Robo-Interface können bis zu drei Programme per Download gespeichert werden. Programm 1 und Programm 2 werden dauerhaft in einem FLASH-Speicher abgelegt, ein drittes Programm kann in das RAM gespeichert werden. Der RAM-Speicher wird gelöscht, wenn ein Flash-Programm gestartet wird und bei Stromausfall am Interface.

Die Auswahl des aktiven Programms erfolgt durch den Programmtaster. Wird dieser länger als 0,5 Sek. betätigt, kann das gewünschte Programm ausgewählt werden. Es leuchten nacheinander die beiden Programm-LEDs für Programm 1 / 2 auf. Ist ein Programm im RAM abgelegt, wird dieses durch die beiden leuchtenden LED´s angezeigt. Sind die Speicherplätze leer, kann das betreffende Programm nicht ausgeführt werden. Zum Starten oder Stoppen des angezeigten Programms muss der Programm-Taster kurz betätigt werden (< 500ms). Das Starten und Stoppen von Programmen kann auch über die PC-Schnittstelle mit dem Programm FtLoader erfolgen.

4 PC-Schnittstellen

4.1 Allgemeines

Die Auswahl der Schnittstellen erfolgt am Robo-Interface per Tastendruck. Nach dem Einschalten ist der "AutoScan"-Modus aktiv. Es werden die USB, serielle Schnittstelle und das Funkmodul (falls vorhanden) überprüft, ob Daten vorhanden sind. Erkennbar ist dieser Zustand an dem Aufleuchten der Schnittstellen-Leuchtdioden.

Sobald eine Schnittstelle Daten sendet, werden die anderen Schnittstellen gesperrt. Die aktive Schnittstelle blinkt, um die Datenkommunikation anzuzeigen. Wenn länger als 300ms keine Daten über die aktive Schnittstelle fließen, schaltet sich der AutoScan-Modus wieder ein.

Durch Drücken des "Port"-Tasters wird die nächste Betriebsart anhand nachfolgender Tabelle ausgewählt.

1. AutoScan USB - Seriell - Funk [1)]
2. AutoScan USB - Seriell
3. USB
4. Seriell
5. IR-Direkt

1)
Diese Betriebsart wird nur aktiviert, wenn das Funkmodul installiert ist.

Wird der Port-Taster länger als 3 Sekunden betätigt, schaltet sich das Interface in den "Intelligent-Interface Online-Modus". Die serielle Schnittstelle arbeitet dann mit den Parametern 9600,n,8,1. Zur Anzeige der Betriebsart blinkt die "SER" - Leuchtdiode sehr schnell. In dieser Betriebsart verhält sich das Interface wie ein Intelligent Interface im Online Modus. Es können jedoch keine Programme herunter geladen werden. Durch einen Druck auf den Port-Taster wird wieder die AutoScan Betriebsart eingestellt.

Im passiven Modus wird das aktive Robo-Interface über eine serielle Schnittstelle (RS232), über USB oder über Funk gesteuert.

4.2 Mehrere USB-Interface an einem Rechner

Um mehrere Interfaces am USB-Bus betreiben zu können, muss zunächst jedem Interface eine eigene Seriennummer zuwiesen werden. Standardmäßig werden alle Interfaces mit der gleichen Seriennummer ausgeliefert, um Probleme beim Austausch von Interfaces zu vermeiden. Das Windows Betriebssystem erkennt jedoch nur Interfaces mit verschiedenen Seriennummern. Für "jede" Seriennummer wird dann der entsprechende Treiber installiert. Dazu benötigt man unter Windows 2000 / XP Administrator-Rechte.

Daher werden standardmäßig alle ROBO Interfaces und ROBO I/O-Extensions mit der gleichen Seriennummer ausgeliefert. Solange nur ein Interface an einem Computer verwendet wird, gibt es keine Probleme, wenn die Interfaces an verschiedenen Computern betrieben werden. Der Computer unterscheidet die Produkte durch Ihren Namen (ROBO Interface, ROBO I/O-Extension und Robo RF-DataLink) und an deren jeweiliger Seriennummer. Daher können ein Robo-Interface und eine Robo I/O-Extension

gleichzeitig an einem Rechner betrieben werden ohne die Seriennummer zu ändern, da es sich um unterschiedliche Produkte handelt.

Wenn aber mehrere gleiche Produkte (z.B. Robo Interfaces) an einem Computer über USB betrieben werden sollen, muss vorher die Seriennummer der Interfaces geändert werden, damit diese vom Computer unterschieden werden können.

Hinweis: Bei Anschluss über die serielle Schnittstelle an den Rechner muss keine Seriennummer geändert werden.

Das Interface hat daher zwei Seriennummern gespeichert. Über die Software kann eingestellt werden, ob die Standardseriennummer "1" oder die vom Hersteller einprogrammierte Geräteseriennummer "2" beim Einschalten des Gerätes aktiv ist.

Das Ändern der Seriennummer kann über die Software FtDiag.exe, RoboPro oder die Funktion GetFtDeviceSetting() bzw. SetFtDeviceSetting() der FtLib geändert werden.

Zum Ändern der Seriennummer darf nur ein Produkt am USB angeschlossen sein, da sonst der Computer dieses nicht unterscheiden kann. In FtDiag.exe rufen Sie nach "SCAN USB" und dem Klick auf den Button "USB Device" das Menü Eigenschaften / Setup auf. In dieser Maske kann man die gewünschte Seriennummer, die nach dem nächsten Einschalten aktiv ist einstellen.

Achtung: Wird die Seriennummer geändert, muss beim nächsten Einschalten des Interface eventuell der Windows Treiber neu installiert werden. Dafür benötigt man unter Windows 2000 / XP aber Administratorrechte. Hat man diese nicht, kann man den Treiber nicht installieren und könnte daher auf das Interface über USB nicht mehr zugreifen. In diesem Fall kann man beim Einschalten des Interface den PROG-Taster gedrückt halten. Das Interface verwendet dann die Seriennummer "1" und wird wieder vom installierten Treiber erkannt (um z.B. die Seriennummer auf erlaubte Werte zu ändern).

Beachten Sie bitte, dass es sich bei den auf den Produkten aufgedruckten Seriennummern um Hex-Zahlen handelt!

Auch wenn USB bis zu 127 Geräte "theoretisch" zulässt, hat sich in der Praxis gezeigt, dass unter Windows XP (mit SP1) auf einem 3 GHz Pentium 4 nur etwa 4-5 Produkte zuverlässig parallel betrieben werden können (d.h. die Aktualisierungszeit der TransferArea beträgt max. 10ms). Unter Windows 98 sind es bis zu 10 Geräte. Dies liegt daran, dass die internen Windows-Treiber für die Motherboardhardware unter XP nicht für "Echtzeitanwendungen" optimiert sind. Es ist daher sinnvoller, IO-Extensions an die Robo-Interface anzuschließen, statt jedes Produkt einzeln an USB.

5 Installation der Entwicklungsumgebung

Die Installationsumgebung kann unter folgendem Link herunter geladen werden:

http://eu.renesas.com/fmwk.jsp?cnt=quicklink_updates_product_child.htm&fp=/products/tools/coding_tools/c_compilers_assemblers/m3t_nc30wa/child_folder/&title=QuickLink%20Updates

Diesen Link findet man auch in der Datei "Link.txt" der FtCComp.zip, um sich das mühsame Abschreiben zu ersparen.

Herunter geladen werden müssen knapp 80MB des "hew4_nc30" -Paketes.
Es handelt sich dabei um das "Renesas C Compiler Package for M16C family V.5.4.00 Release 00 with High-performance Embedded Workshop (HEW) V.4".

Hinweis:
Es ist möglich, dass Renesas diesen Link bei Erscheinen einer neuen Version ändert. Man muss dann auf der Renesas Webseite die Entwicklungsumgebung für den M16C/24 (M16C/20 Serie) suchen. Bisher konnte man den C-Compiler anonym herunterladen ohne sich registrieren zu müssen. Da er aber von der japanischen Seite nun nur noch mit Registrierung herunter geladen werden kann, ist es denkbar dass "Renesas Europa" dies vermutlich auch einführen wird.

In der Datei "Link.txt" finden sich auch alternative Adressen (sofern verfügbar).

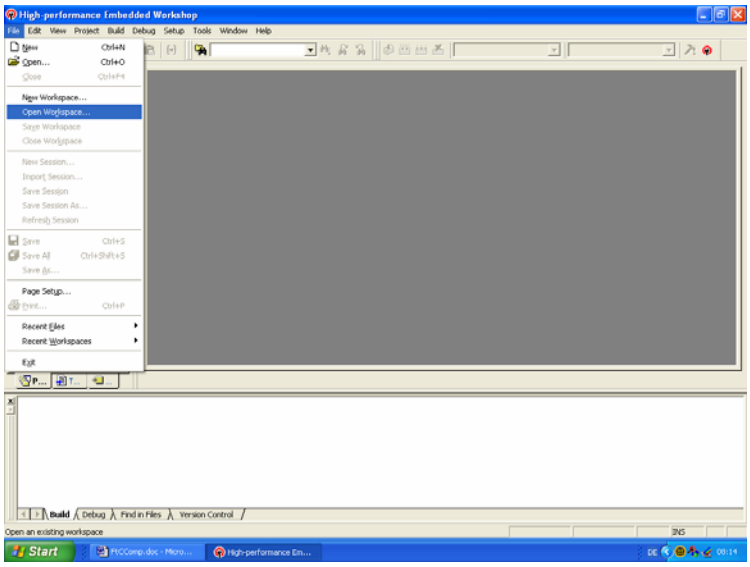
Die genaue Installation des Paketes ist in einer gesonderten Anleitung (Datei "Renesas-Install.pdf") beschrieben.

6 Fünf Schritte bis zum ersten C-Programm

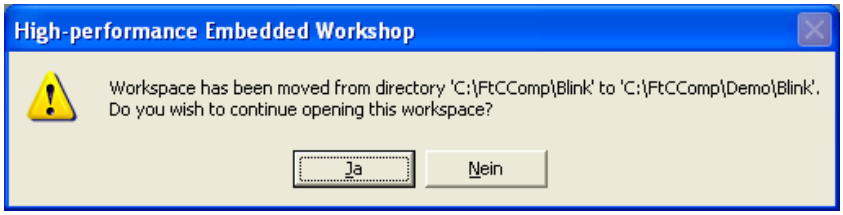
Nachdem die Renesas Entwicklungsumgebung installiert ist, können die FtCComp - Beispielprogramme in ein eigenes Unterverzeichnis auf der Festplatte kopiert werden. In dieser Anleitung verwenden wir für die Beispiele nachfolgend das Verzeichnis c:\FtCComp. Der Inhalt der FtCComp.zip Datei lassen wir dorthin entpacken.

Die nachfolgenden fünf Schritte zeigen den Weg zum ersten C-Programm im Robo-Interface.

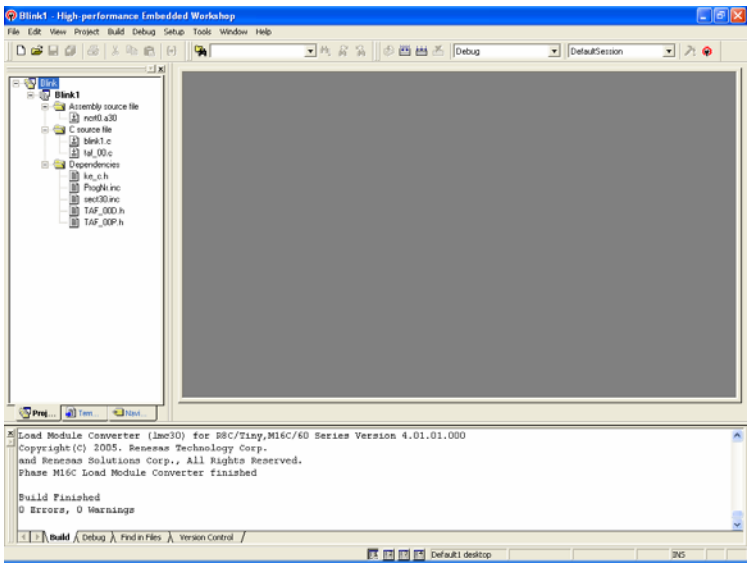
Wichtiger Hinweis:
Das Robo Interface muss mindestens die Firmware 01.58.00.03 oder höher besitzen, da sonst die Unterstützung für eigene Programme in der Firmware fehlt.



1. Starten Sie die Renesas Entwicklungsumgebung " High-performance Embedded Workshop" und laden Sie über "File / Open Workshop" die Datei "Blink.hws" im Verzeichnis "C:\FtCComp\Demo\Blink".

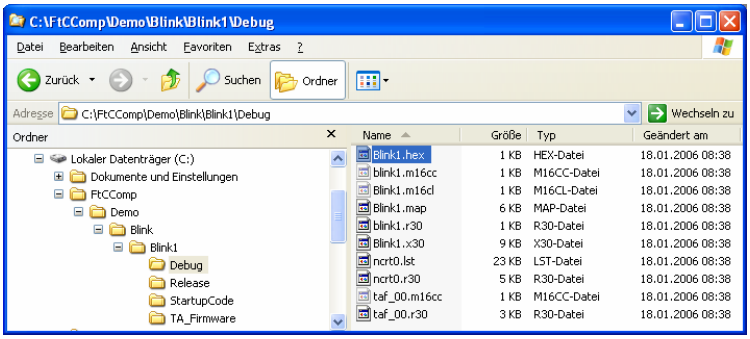


   Falls Sie die Beispieldateien in einem anderen Verzeichnis installiert haben, bekommen Sie eine Meldung, dass die Datei zuletzt in einem anderen Verzeichnis gespeichert wurde und nun verschoben ist. Sie können diese Meldung mit "Ja" übergehen.

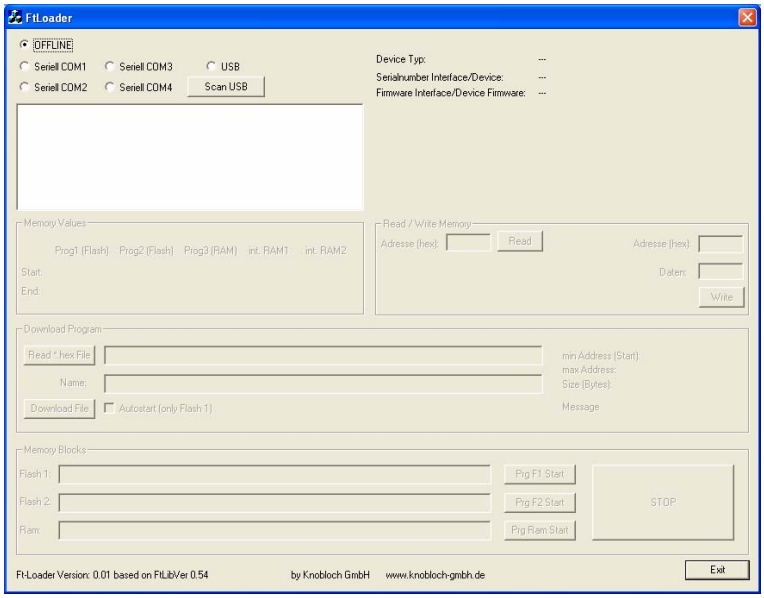


2. Kompilieren Sie nun das Beispielprogramm durch Anklicken des Button "Build All" (links neben "Debug") oder verwenden Sie in der Menüleiste "Build / Build All" (Taste "F7" geht aber auch…).

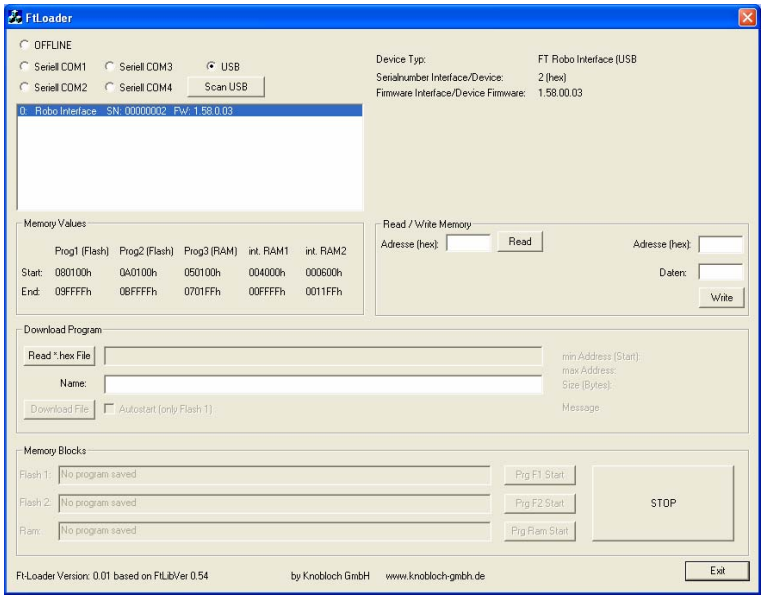
   Im unteren Ausgabefenster laufen nun einige Meldungen sehr schnell durch. Wenn der Compiler und Linker fertig sind, sollte dort "0 Errors" stehen. Sobald die Probezeit von 60 Tagen abgelaufen ist, erscheint immer eine "Warnung" die darauf hinweist, dass nun nur noch 64k große Programme erzeugt werden können.



- Das eigentliche Programm für das Robo Interface wurde in der Datei "Blink1.hex" im Verzeichnis "C:\FtCComp\Demo\Blink\Blink1\Debug" gespeichert.

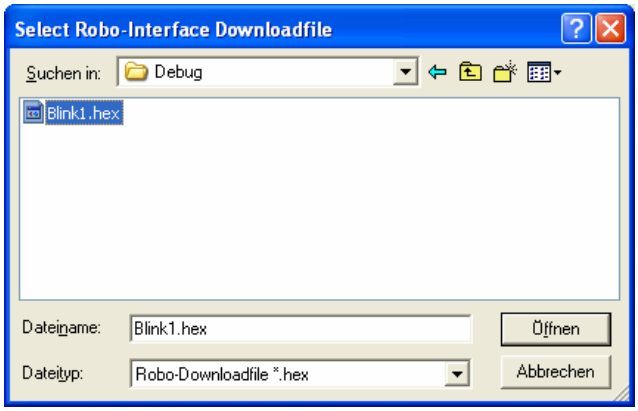


3. Starten Sie das Programm FtLoader.exe (befindet sich im FtCComp – Verzeichnis). Mit diesem Programm wird die Programmdatei in das Interface gespeichert.
   Nachdem das Interface an den Rechner angeschlossen und mit Strom versorgt ist, kann das Programm in das Interface gespeichert werden.

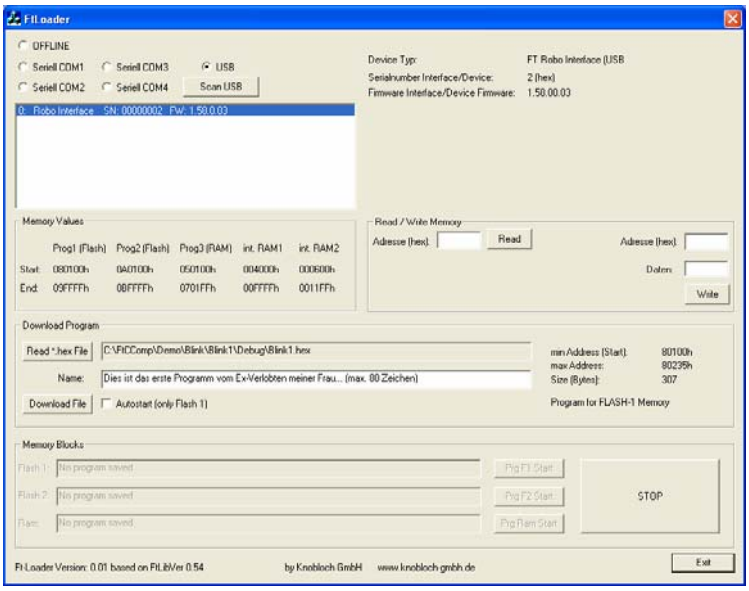


In diesem Beispiel ist das Interface an USB angeschlossen, durch Anklicken des "Scan USB" Buttons werden alle Interface gesucht. Anschließend das gewünschte Interface markieren, so dass es farbig unterlegt ist.

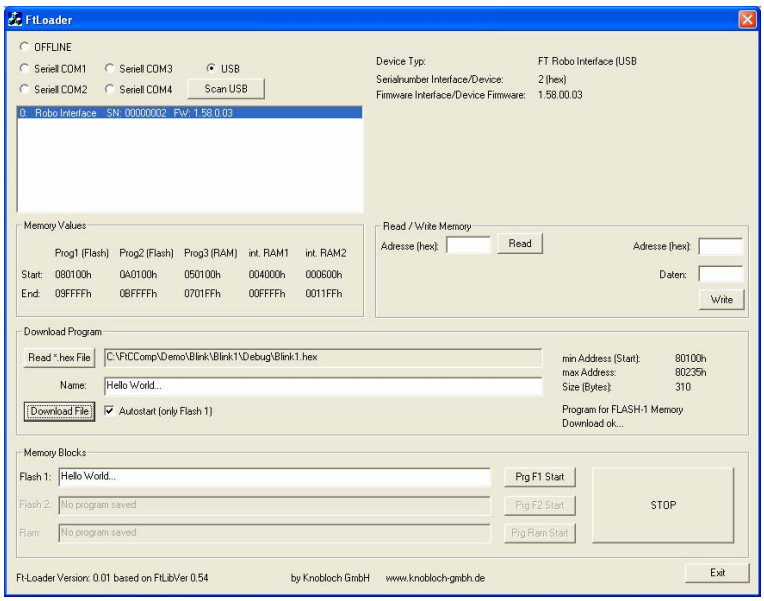
Das Programm fragt nun die aktuellen Speicherbereiche des Interface ab und zeigt die Werte zur Information im Fenster "Memory Values" an. Im unteren Fenster "Memory Blocks" wird der Zustand der drei Speicherbereiche des Interface angezeigt. Zu jedem Programm kann ein Name mit maximal 80 Zeichen gespeichert werden.



4. Öffnen Sie nun die Datei "Blink1.hex" aus dem Verzeichnis "C:\FtCComp\Demo\Blink\Blink1\Debug" durch Betätigen der "Read *.hex File" Schaltfläche.



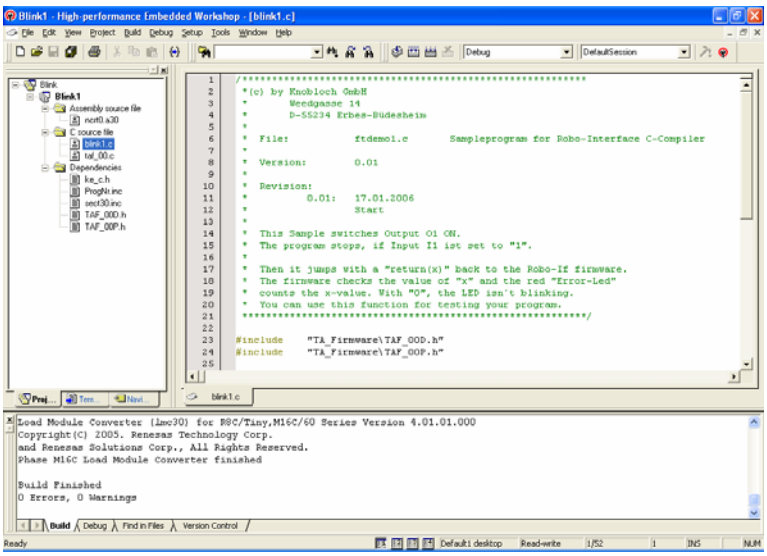
- Die Datei wird eingelesen und automatisch anhand der Adressen ermittelt, für welchen Speicherbereich das Programm ist. In diesem Beispiel für den Bereich "Flash 1". Als Beschreibungstext kann ein Name mit bis zu 80 Zeichen gespeichert werden.



5. Durch den Downloadbutton wird das Programm in das Interface zusammen mit dem Namen gespeichert und die Anzeige aktualisiert. Im Speicherplatz "Flash 1" ist nun das Programm gespeichert. Es kann über die "Prg F1 Start" Schaltfläche gestartet und über "Stop" wieder beendet werden.

   Herzlichen Glückwunsch!
   Sie haben nun Ihr erstes "eigenes" Programm erzeugt und in das Interface gespeichert.

   Auf der nächsten Seite "untersuchen" wir die Funktionsweise des "Blink1" Programmes etwas näher.



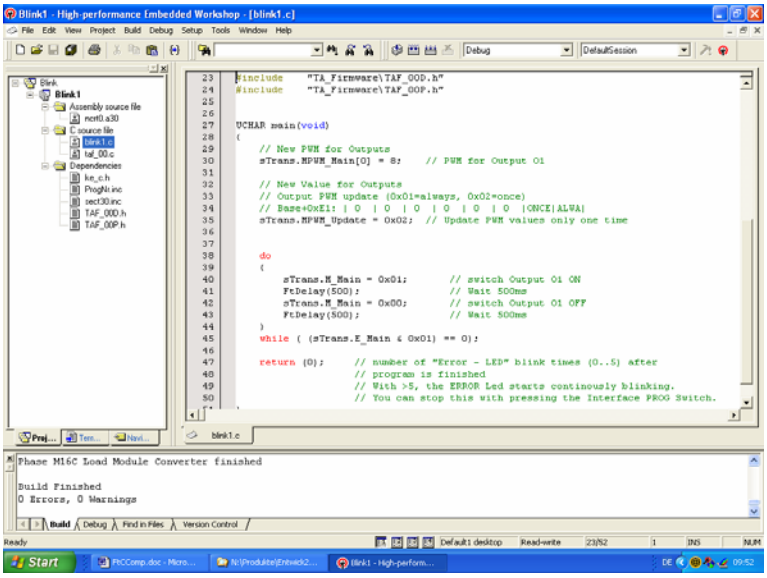
- Durch Anklicken des Dateinamens "Blink1.c" öffnen wir die Programmdatei. Diese wird im rechten Fenster angezeigt.
  Nach der Beschreibung (angezeigt in der Farbe grün) folgen zwei wichtige Zeilen:

```
#include    "TA_Firmware\TAF_00D.h"
#include    "TA_Firmware\TAF_00P.h"
```

  In der Headerdatei "TAF_00D.h" befindet sich die Definition der "Transferarea", einer Datenstruktur über die das Programm mit der Hardware im Interface "kommuniziert".

  In der Headerdatei "TAF_00P.h" befindet sich die Definition von Firmwaremakros um Funktionen innerhalb der Firmware aufzurufen.

  Diese beiden Dateien müssen daher bei jedem Programm enthalten sein. Ferner muss zu jedem Programm noch die Datei "TAF_00.c" hinzugefügt werden. Man findet diese im linken Bereich unter "C source files".



```
UCHAR main(void)
{
    // New PWM for Outputs
    sTrans.MPWM_Main[0] = 8;    // PWM for Output O1

    // New Value for Outputs
    // Output PWM update (0x01=always, 0x02=once)
    // Base+0xE1: | 0  | 0  | 0  | 0  | 0  | 0  |ONCE|ALWA|
    sTrans.MPWM_Update = 0x02;  // Update PWM values only one time


    do
    {
        sTrans.M_Main = 0x01;       // switch Output O1 ON
        FtDelay(500);               // Wait 500ms
        sTrans.M_Main = 0x00;       // switch Output O1 OFF
        FtDelay(500);               // Wait 500ms
    }
    while ( (sTrans.E_Main & 0x01) == 0);

    return (0);     // number of "Error - LED" blink times (0..5) after
                    // program is finished
                    // With >5, the ERROR Led starts continously blinking.
                    // You can stop this with pressing the Interface PROG Switch.
}
```

- Nach den beiden Headerdateien beginnt das "eigentliche" Programm "main".

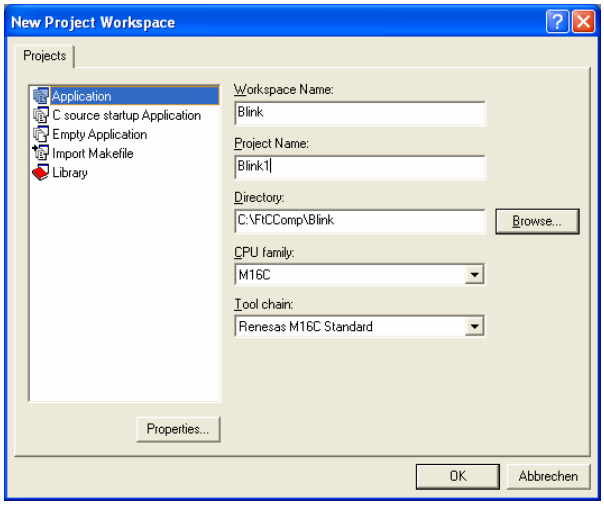
  ```
  sTrans.MPWM_Main[0] = 8;    // PWM for Output O1
  ```
  Zuerst wird dem Ausgang 1 der Geschwindigkeitswert "8" zugewiesen.

  ```
  sTrans.MPWM_Update = 0x02;  // Update PWM values only one time
  ```
  Da die Berechnung der PWM-Werte für die Ansteuerung der Ausgänge etwas Zeit benötigt muss der Firmware mitgeteilt werden, wann eine Neuberechnung stattfinden soll. Dies erfolgt nun in dieser Zeile. Mit dem Wert 0x01 berechnet die Firmware alle 10ms die PWM-Werte neu, bei dem Wert 0x02 nur ein einziges Mal. Das Bit wird nach der Berechnung gelöscht.

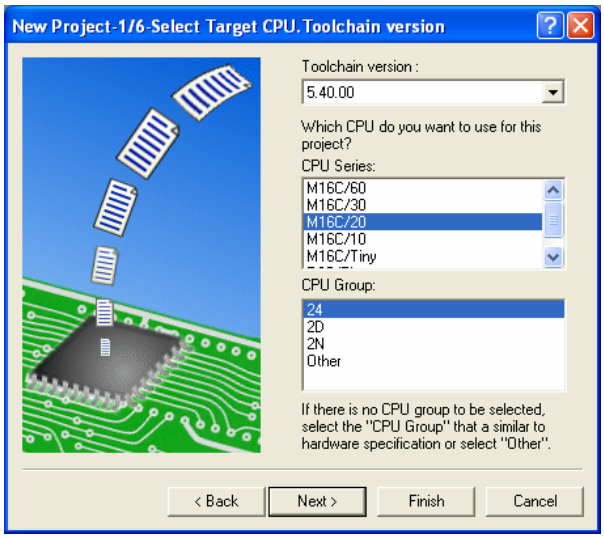
  In der nachfolgenden `do..while` Schleife wird der Ausgang 1 für 500ms an- und wieder ausgeschaltet. Sobald der Eingang I1 gesetzt wird, beendet sich das Programm. Durch die return – Anweisung kann nun ein Zahlenwert an die Firmware gemeldet werden der darüber entscheidet, wie oft die rote Error-Leuchtdiode blinken soll. Bei "0" bleibt sie dunkel, 1..5 entspricht der Anzahl der Blinker und bei einem Wert >5 blinkt die Leuchtdiode dauerhaft (bis der PROG-Taster einmal betätigt wird).

7 Neues Projekt anlegen

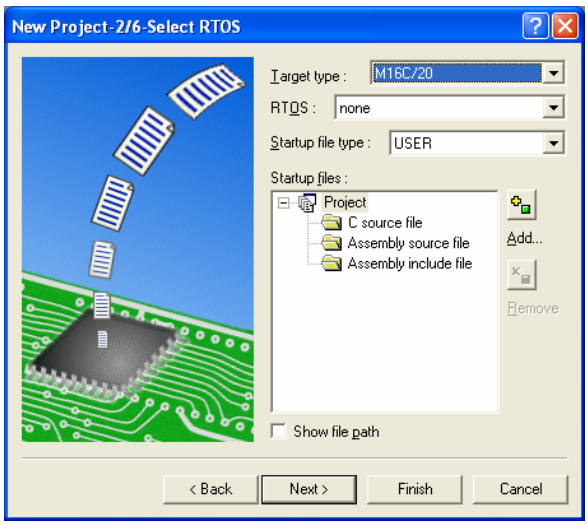
- Falls ein Projekt geöffnet ist, in der HEW (High-performance Embedded Workshop) den aktuellen Workspace schließen (File / Close Workspace).
- Neuen Workspace anlegen (File / New Workspace).



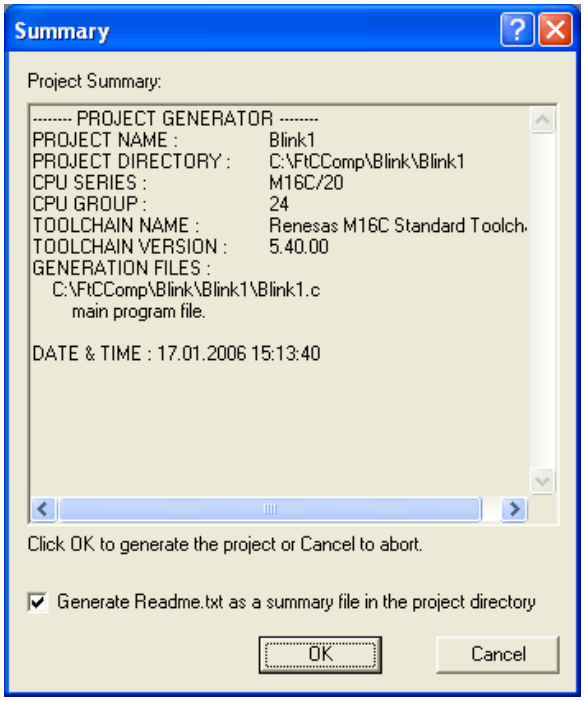
- Zuerst sollten Sie das Verzeichnis auswählen (z.B. c:\FtCComp)
- In diesem Beispiel nennen wir den Workspace Namen "Blink", es wird automatisch das "Blink" - Unterverzeichnis im FtCComp - Verzeichnis vorgeschlagen.
- Das Projekt nennen wir "Blink1".
- Die CPU Familie ist "M16C" und die Toolchain "Renesas M16C Standard".



- An dieser Stelle muss nun die CPU-Serie M16C/20 mit der CPU-Gruppe "24" ausgewählt werden. Anschließend "Next" auswählen.
  Durch "Finish" werden einige Einstellungen nicht abgefragt, die wir aber benötigen.

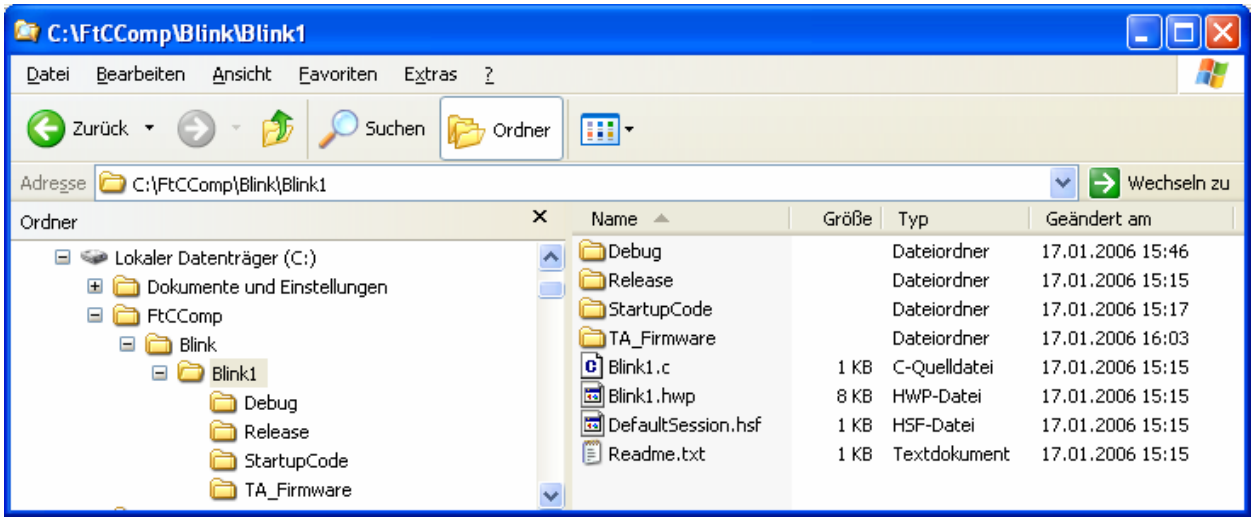


- Der "Target type" muss nun auf "M16C/20" und die "Startup file type" auf "USER" eingestellt werden. Danach "Finish" auswählen.

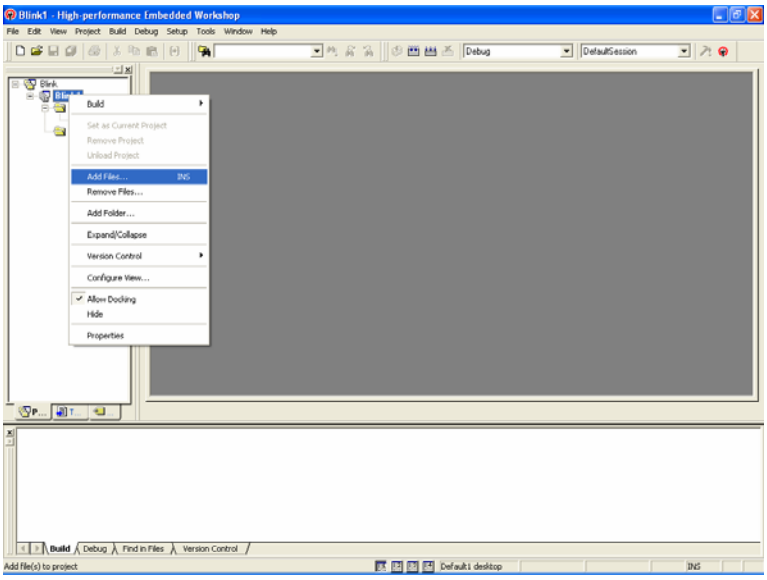


- Es erscheint die Zusammenfassung, mit "OK" wird das Projekt angelegt.

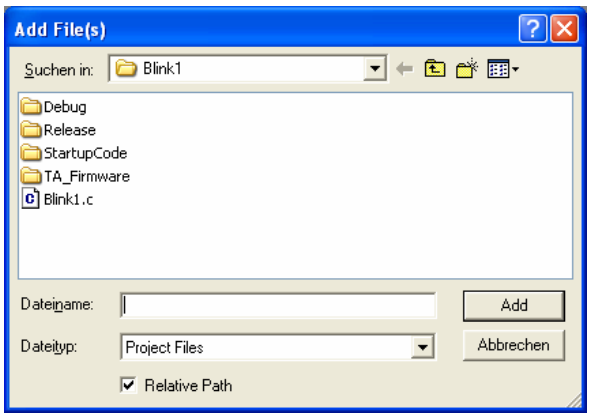
- Der C-Compiler benötigt für das spätere C-Programm einige Robo-Interface spezifische Dateien, die sich im Verzeichnis "StartupCode" und "TA_Firmware" der Beispiele befinden.



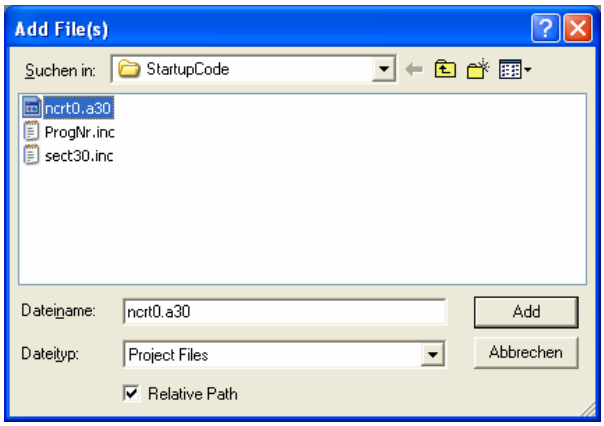
- Kopieren Sie diese Verzeichnisse nun über den Windows Explorer in das Blink-Verzeichnis C:\FtCComp\Blink\Blink1.



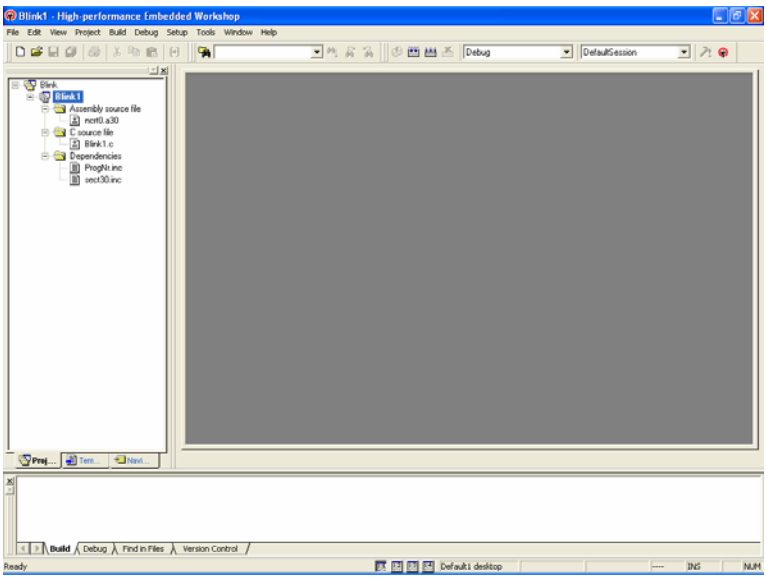
- Dem Projekt müssen nun diese Dateien mitgeteilt werden. Mit einem rechten Mausklick auf den Projektnamen "Blink1" können über den Eintrag "Add Files" die Startdateien hinzugefügt werden.



- Es öffnet sich das "Add Files" Menü. Hier in das Verzeichnis "StartupCode" wechseln und nur die Datei "ncrt0.a30" markieren. Die beiden anderen Dateien (ProgNr.inc und sect30.inc) werden nicht markiert.

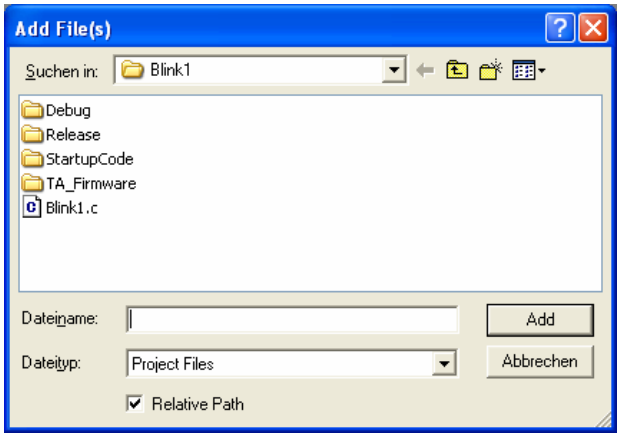


- Mit dem "Add" Button wird diese nun dem Projekt zugefügt.

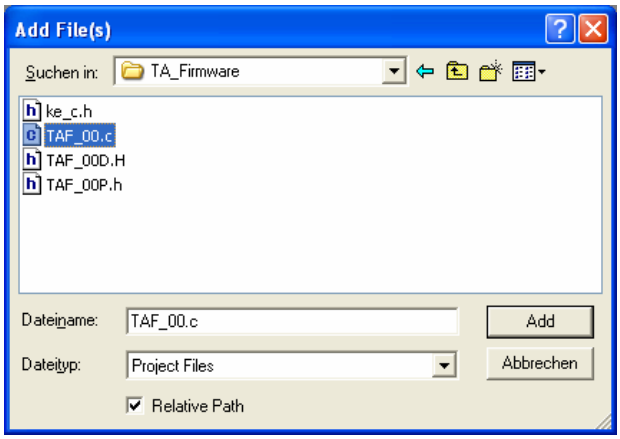


- Abhängig vom Dateityp fügt die Add-Funktion die Datei in ein entsprechendes Verzeichnis im Projektbaum ein. Die Dateien werden durch die Add-Funktion gleichzeitig "analysiert". Man kann dies daran erkennen, dass die beiden Includedateien (ProgNr.inc und sect30.inc) im Verzeichnis "Dependencies" stehen.
  Hinweis: In den FtCComp - Beispielen wurden eigene Verzeichnisse angelegt und die Dateien dementsprechend strukturiert. In diesem Beispiel halten wir die automatische Anordnung bei.

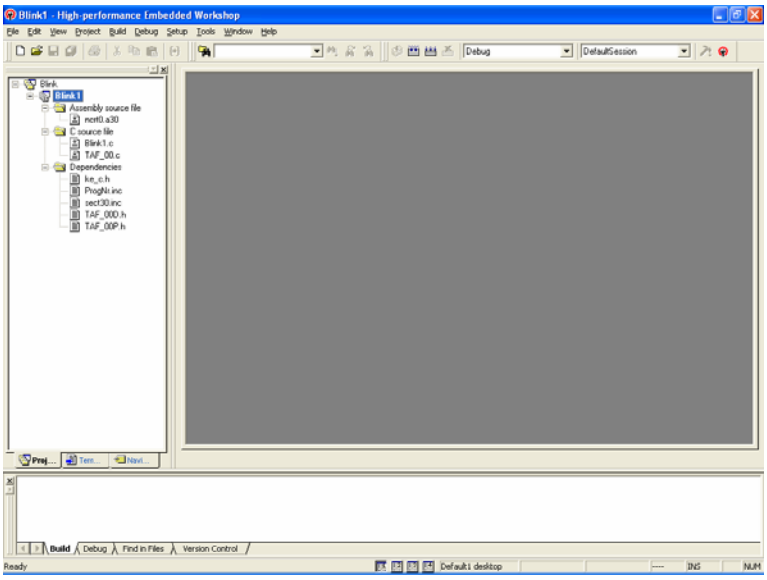
- Im nächsten Schritt werden nun noch die Dateien aus dem TA_Firmware Verzeichnis hinzugefügt. Auch hier wieder ein Klick mit der rechten Maustaste auf den Projektnamen "Blink1" und "Add Files", um die Dateien TA_Firmware hinzuzufügen.

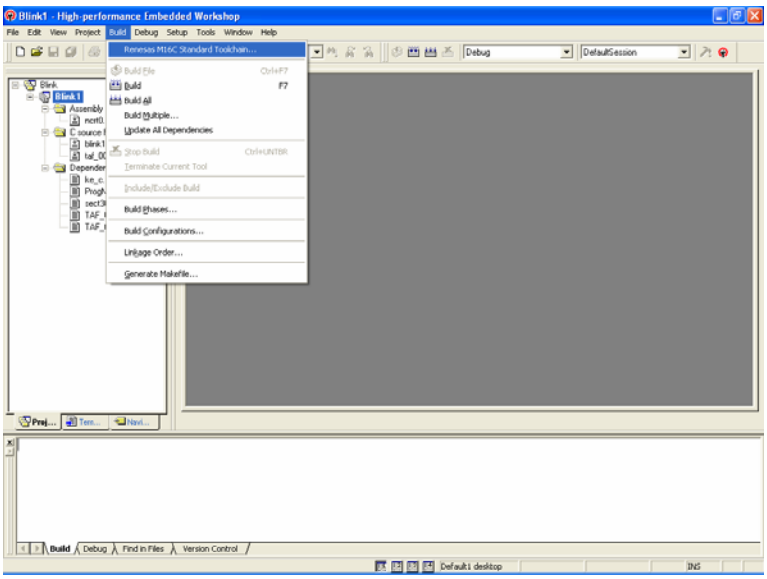


- Es öffnet sich das "Add Files" Menü. Hier nun in das Verzeichnis "TA_Firmware" wechseln und nur die Datei "TAF_00.c" markieren. Die anderen Dateien werden nicht markiert.

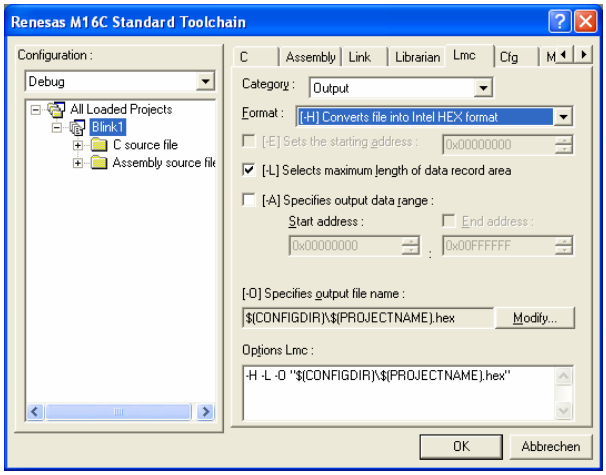


- Mit dem "Add" Button wird diese nun dem Projekt zugefügt.

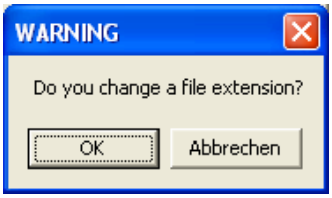




- Nun muss der Entwicklungsumgebung mitgeteilt werden, dass wir die Ausgabedatei im "HEX" Format benötigen. Öffnen Sie dazu "Build / Renesas M16C Standard Toolchain".

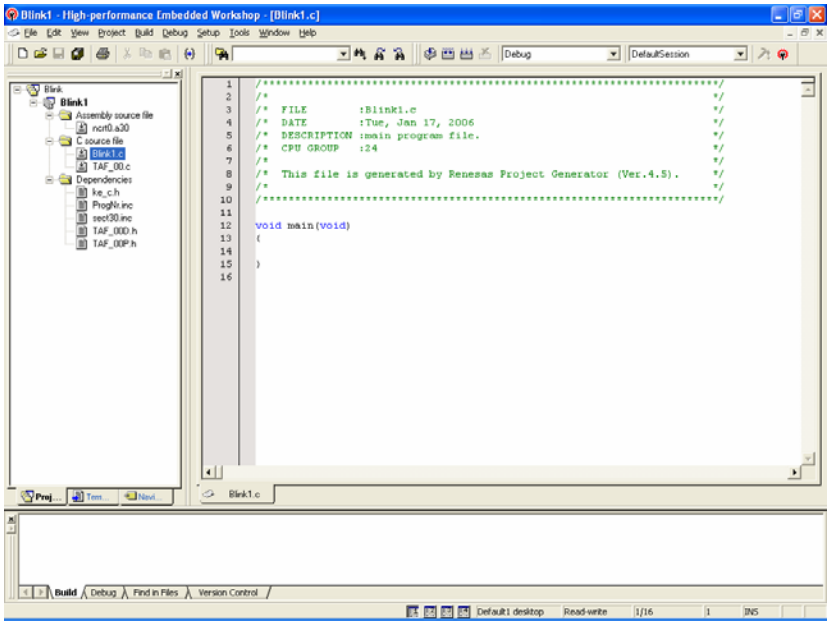


- Auf der Karteikarte "Lmc" in der Kategorie "Output" muss "Converts file into Intel HEX format" ausgewählt werden.



- Den Warnhinweis mit "OK" bestätigen.
- Anschließend auch die Einstellung auf der "Lmc" Karteikarte mit "OK" bestätigen.

- Nachdem nun diese "Vorarbeiten" erfolgt sind, kann es mit dem "eigentlichen Programmieren" losgehen. Hierzu nun einen Doppelklick (linke Maustaste) auf die "C source file" – Datei "Blink1.c" klicken.



- Im rechten Fenster öffnet sich die Datei. Diese wurde automatisch beim Anlegen des Projektes erzeugt. Damit das Programm auch die Hardware des Interface steuern kann, müssen nach der Beschreibung (angezeigt in der Farbe grün) zwei wichtige Zeilen folgen:

```
#include    "TA_Firmware\TAF_00D.h"
#include    "TA_Firmware\TAF_00P.h"
```

  In der Headerdatei "TAF_00D.h" befindet sich die Definition der "Transferarea", einer Datenstruktur über die das Programm mit der Hardware im Interface "kommuniziert".

  In der Headerdatei "TAF_00P.h" befindet sich die Definition von Firmwaremakros, um Funktionen innerhalb der Firmware aufzurufen.

  Diese beiden Dateien müssen daher bei jedem Programm enthalten sein. Ferner muss zu jedem Programm noch die Datei "TAF_00.c" hinzugefügt werden. Man findet diese im linken Bereich unter "C source files". Diese kann über die "Add" – Funktion dem Projekt hinzugefügt werden.

  Die Programmieroberfläche wird normalerweise für die Entwicklung von "Embedded-Systemen" verwendet. Es wird also Steuersoftware für z.B. eine Heizungsregelung in einem Wohnhaus erstellt. Eine solche Software kann man üblicherweise nicht "beenden", daher wird eine "main" - Funktion nach dem Einschalten einmal gestartet und sollte dauerhaft arbeiten. Jedes C-Programm startet mit der "main()" Prozedur. Sie dürfen daher diesen Namen nicht ändern.

  Im Robo Interface darf sich das Programm selbst beenden, wenn dies erforderlich ist. Beim Beenden erwartet die Firmware einen Zahlenwert (0…255). Dieser bestimmt, ob die rote Error-Leuchtdiode nach dem Programmstopp "blinken" soll. Dass dies funktioniert müssen einige Änderungen vorgenommen werden.

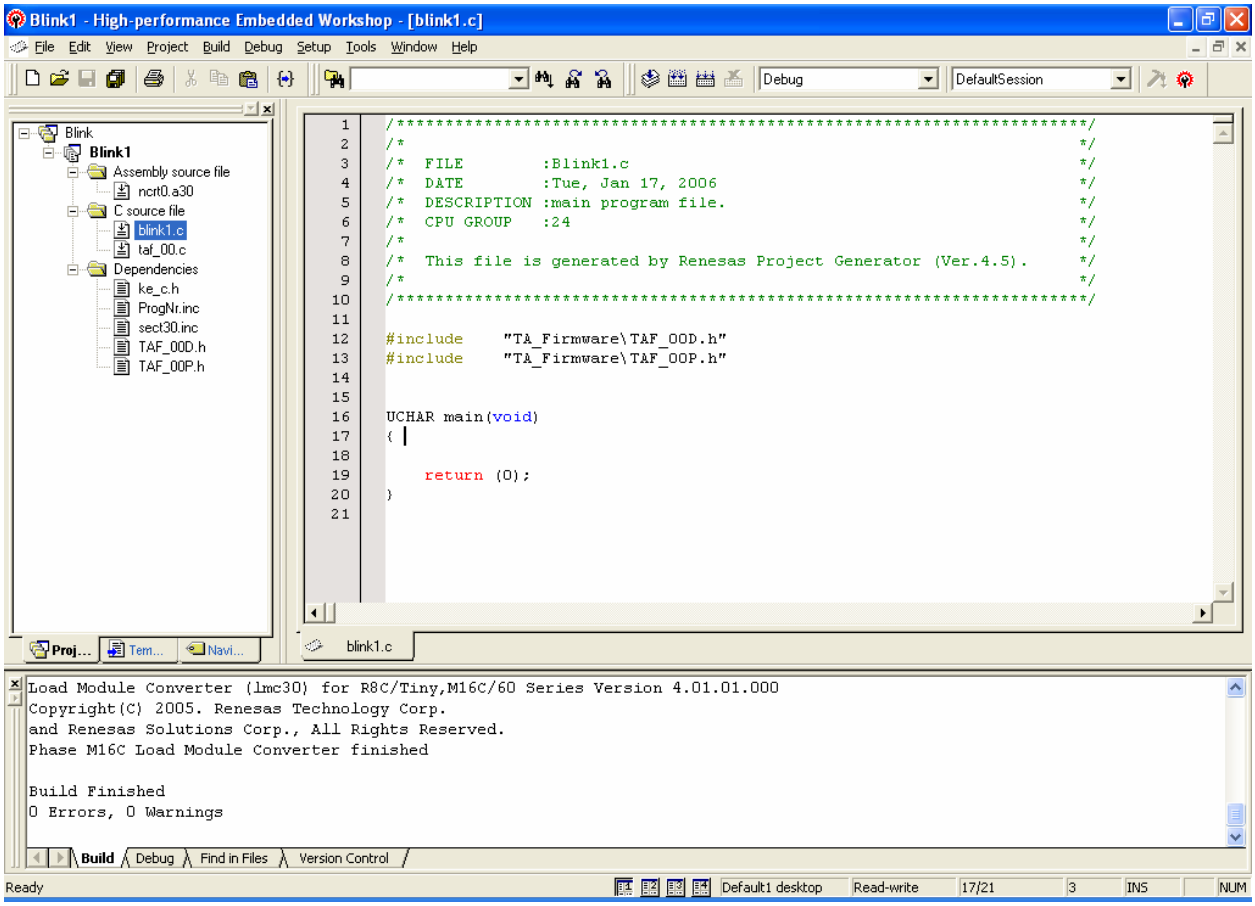
Das "void" vor dem "main" muss in "UCHAR" geändert werden, damit die Funktion einen Wert zurückliefern kann:

```
UCHAR main(void)
```

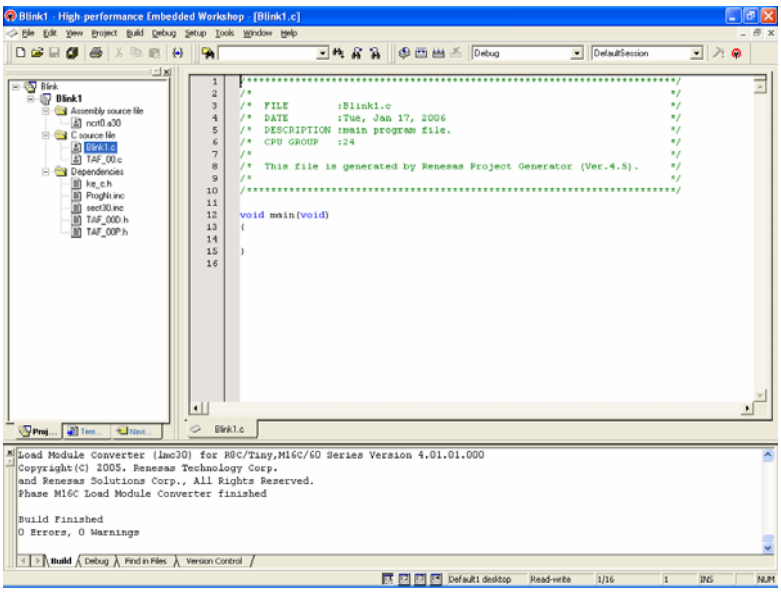
Am Ende der Main-Funktion muss dann ein "return" mit einem Wert angegeben werden:

```
return (0);
```

Das Ergebnis sieht nun so aus:



- Nun der erste Test: klicken Sie den Button "Build All" an (links neben "Debug") oder verwenden Sie in der Menüleiste "Build / Build All" (Taste "F7" geht aber auch…). Der Compiler übersetzt nun das Programm in Maschinenbefehle.



- Im unteren Ausgabefenster laufen nun einige Meldungen sehr schnell durch. Wenn der Compiler und Linker fertig sind, sollte dort "0 Errors" stehen. Sobald die Probezeit von 60 Tagen abgelaufen ist, erscheint immer eine "Warnung" die darauf hinweist, dass nun nur noch 64k große Programme erzeugt werden können.

- Nachdem nun sichergestellt ist, dass die Vorgaben richtig sind, kann mit dem Erstellen des Programms begonnen werden. Das fertige Blink1-Programm finden Sie in den Beispielen. Bitte beachten Sie auch die Hinweise im nachfolgenden Kapitel.

8 Hinweise zur Programmierung

Das Abfragen der Eingänge und Steuern der Ausgänge erfolgt durch eine "Transferarea" (Kommunikationsspeicherbereich). Dieser Bereich wird nach dem Starten des Programms durch die Interfacefirmware alle 10ms mit der Hardware abgeglichen. Programmierer die schon mit der FtLib gearbeitet haben, kennen dieses Prinzip. Den Aufbau der Transferarea findet man im Verzeichnis TA_Firmware in der Headerdatei "TAF_00D.h". Die Transferarea ist identisch zur FtLib-Version. Die Startadresse der Transferarea ist in der Firmware fest auf 0x400 im RAM eingestellt.

Da der Linker das Programm auf absolute Hardwareadressen des Interface "binden" muss, müssen diese Adressen dem Compiler mitgeteilt werden. Dies erfolgt in der Datei " ProgNr.inc" im StartupCode-Verzeichnis. Hier genügt es durch Angabe von FLASH1, FLASH2 oder RAM1 anzugeben, für welchen Speicherbereich das Programm erzeugt wird. In den Beispielen wird immer FLASH1 verwendet.

Hinweis:
Das FtLoader - Programm liest die aktuellen Speicheradressen des angeschlossenen Interface aus und zeigt die Werte an. In zukünftigen Firmwareversionen des Interface wäre es denkbar, dass sich an den Adressen etwas ändert, diese können dann in der "ncrt0.a30" - Datei geändert werden.

Was normalerweise bei Embedded-Programmen nicht möglich ist, ist hier erlaubt: "main()" darf sich selbst beenden und kehrt dann in die Interface Firmware zurück! Wenn main() in die Firmware zurückkehrt wird ein Rückgabewert von der Firmware erwartet. Daher sollte die main() Routine als Funktion deklariert werden.
Durch den Rückgabewert kann die rote Error - LED im Interface geschaltet werden. Ein Wert von 1..5 entspricht der Blinkanzahl. Ab einem Wert größer als 5 blinkt die rote Leuchtdiode dauerhaft bis die Prog Taste betätigt wird. Beim Wert "0" blinkt sie nicht.

9 Maschinennahe Programmierung

Mit dem C-Compiler ist es möglich Programme zu erzeugen, die den Prozessor im Interface beschädigen können! Im Gegensatz zu Programmen der fischertechnik RoboPro-Software kann mittels des eigenen C-Programms direkt auf die Hardwareports im Prozessor zugegriffen werden. Da bei diesem modernen Prozessor ein Großteil der internen Hardware durch Softwarebefehle konfiguriert wird, ist es denkbar, dass bei falschen Einstellungen der Prozessor mit der übrigen Interface Hardware nicht mehr richtig zusammenarbeitet und leider auch Schäden dadurch entstehen können. Insbesondere von Speicherzugriffen über Zeiger geht eine große Gefahr aus, da hierüber auch die Prozessorregister erreicht werden können.

Der Hersteller des Interface muss daher Garantieansprüche bei beschädigten Prozessoren durch fehlerhafte C-Programme ablehnen.

10 Die Transferarea

Das Abfragen der Eingänge und Steuern der Ausgänge erfolgt durch eine "Transferarea" (Kommunikationsspeicherbereich). Dieser Bereich wird nach dem Starten des Programms durch die Interfacefirmware alle 10ms mit der Hardware abgeglichen. Programmierer die schon mit der FtLib gearbeitet haben kennen dieses Prinzip. Den Aufbau der Transferarea findet man im Verzeichnis TA_Firmware in der Headerdatei "TAF_00D.h". Die Transferarea ist identisch zur FtLib-Version. Die Startadresse der Transferarea ist in der Firmware fest auf 0x400 im RAM eingestellt.

Damit diese in einem C-Programm verwendet werden kann, müssen die beiden nachfolgenden Zeilen am Anfang des Programms stehen:

```
#include    "TA_Firmware\TAF_00D.h"
#include    "TA_Firmware\TAF_00P.h"
```

In der Headerdatei "TAF_00D.h" befindet sich die Definition der "Transferarea", einer Datenstruktur über die das Programm mit der Hardware im Interface "kommuniziert".

In der Headerdatei "TAF_00P.h" befindet sich die Definition von Firmwaremakros um Funktionen innerhalb der Firmware aufzurufen.

Diese beiden Dateien müssen daher bei jedem Programm enthalten sein. Ferner muss zu jedem Programm noch die Datei "TAF_00.c" hinzugefügt werden (s. Beispiele).

10.1 Digitaleingänge E1-E32

Die Bits für die Digitaleingänge werden bei einem offenen Eingang auf 0, bei einem mit „+“ verbundenen Eingang auf 1 gesetzt. Nicht vorhandene Eingänge (fehlende Erweiterungsmodule) werden auf 0 gesetzt. Zusätzlich sind alle 32 Eingänge nochmals ab Base+0x100 in einer 16-Bit Variablen pro Eingang abgelegt (1=Eingang betätigt).

10.2 Sondereingänge

Sondereingänge sind die 11 Tasten der IR-Fernbedienung. Die Nummer der am IR-Sender gedrückten Taste, sowie die Information ob Code „1“ oder Code „2“ aktiviert wurde, wird in die Speicherstelle Base+0x0E abgespeichert. Zusätzlich werden alle Tasten nochmals in 16 Bit-Variablen (wie die Digitaleingänge) abgespeichert.

10.3 Analogeingänge

Die Analogeingänge werden als 16 Bit-Werte mit einem Wertebereich von 0…1023 abgelegt. Zusätzlich berechnet die Firmware den Widerstandswert (in Ohm) für die AX und AY Eingänge. Deren Wertebereich geht daher von 0 bis 5700.

10.4 16-Bit Timer

Die 6 16 Bit-Timer mit Inkrementen von 1ms, 10ms, 100ms, 1s, 10s und 60s stehen zur freien Verfügung. Zwischen den einzelnen Timerwerten besteht keine feste Beziehung, d.h. der 10ms Wert ist z.B. nicht 10x der 1ms Wert.

10.5 Ausgänge

Die Ausgänge werden über ein Polaritätsbit, ein Energiesparbit und ein PWM-Wert- Byte gesteuert. Der PWM-Wert und das Polaritätsbit werden pro Einzelausgang angegeben. Das Energiesparbit wird pro Ausgangspaar angegeben. Ist das Polaritätsbit 0, wird der Ausgang auf Masse gesetzt, ist das Polaritätsbit 1 wird der Ausgang auf Versorgung (9V) gesetzt. Ist das Energiesparbit 1 wird ein Ausgangspaar nach einer Verzögerung (1 sek.) hochohmig geschaltet, wenn beide zugehörigen Polaritätsbits auf 0 gesetzt sind. Ist das Energiesparbit auf 0 gesetzt, wird das zugehörige Ausgangspaar nicht hochohmig geschaltet. Über das PWM-Byte, das einen Wertebereich von 0…7 hat, wird die Pulsbreite des Ausgangs in 8 Stufen eingestellt (z.B. in 12.5% Schritten zwischen 12.5% und 100%).

Hinweis für Downloadprogramme:
Die Ausgangseinstellungen werden von der Firmware alle 10ms in einen eigenen Datenbereich kopiert und die PWM-Werte berechnet, wenn das Bit "UpdateAusgänge" (UA1) auf BASE+0xE1 gesetzt ist. Wird anstelle UA1, UA2 gesetzt, dann werden die Ausgänge nur einmalig beim nächsten 10ms Interrupt geschrieben. Nachdem die Ausgänge gesetzt sind, wird das UA2 Bit gelöscht.

Nach der Initialisierung des Interface (Einschalten) sind alle Energiesparbits auf "1" gesetzt. Standardmäßig ist diese Funktionalität aktiviert.

10.6 Betriebsmodus, installierte Erweiterungen

Ab Base+0xE6 findet sich die Anzahl der installierten I/O - Extensionmodule.

11 Firmware Aufrufe

Zusätzlich zur Transferarea bietet die Interface Firmware einige Funktionen (z.B. senden einer Nachricht). Diese werden nachfolgend beschrieben.
Der Aufruf erfolgt über die in der Datei " TAF_00P.h" im Verzeichnis "TA_Firmware" definierten Makros.

11.1 void SetFtDeviceReset (char mode)

```
mode:   0 =   Kaltstart / RESET
              (wie Interface einschalten)

        1 =   PROG-Ende
              (Init, ohne RAM löschen)
```

Diese Prozedur stoppt das Anwendungsprogramm. Beim Aufruf kann bestimmt werden, ob das Interface einen kompletten "Neustart" durchführt (Hardware neu initialisiert, Ram löscht), oder nur die Hardware und die Firmware neu initialisiert ohne den RAM zu löschen.

11.2 void SetFt1msTimerTickAddress(void far *())

Über diese Funktion teilt das Anwendungsprogramm der Firmware die Adresse der Timer-Routine mit. Diese wird danach alle 1ms aufgerufen. Mit dem Wert NULL kann die Funktionalität abgeschaltet werden.

11.3 void SetFtDeviceCommMode (BYTE mode, BYTE value1, UINT *pData)

Diese Routine stellt die Betriebsart der seriellen Schnittstelle im Interface ein. Nach dem Einschalten befindet sich die Schnittstelle im Normalbetrieb. In dieser Betriebsart kann das Interface im Onlinebetrieb gesteuert werden.
Im "Messagebetrieb" können Nachrichten von einem Interface in ein weiteres über die serielle Schnittstelle gesendet werden.

Die eingestellte Betriebsart bleibt solange erhalten, bis mit dieser Funktion eine neue Betriebsart eingestellt wird. Durch Drücken des "Port"-Tasters am Interface wird wieder die IF_COM_ONLINE Betriebsart eingestellt, wenn sich das Interface im AutoScan-Betrieb befindet.

Der aktuell eingestellte Zustand kann über "IF_COM_PARAMETER" abgefragt werden. Das Ergebnis wird ab der übergebenen Adresse auf die "pData" zeigt gespeichert.

Betriebsarten:

```
Mode =    IF_COM_ONLINE:    Standardmodus einstellen
                            Die Schnittstelle wird auf Standardmodus
                            eingestellt (Parameter: 38400,n,8,1).

Mode =    IF_COM_MESSAGE:   Nachrichtenversand über serielle
                            Schnittstelle

Mode =    IF_COM_PARAMETER  Betriebsart auslesen
                            Ergebnis:
                            pData: Highbyte = Value wenn Modus = Data
                                   Lowbyte =  Modus
```

11.4 void GetRfParameter (BYTE *pData)

Diese Funktion liest die Parameter des RF-Moduls aus und speichert diese ab der Adresse, auf die pData zeigt ab. Das Datenfeld muss mindestens 4 Byte groß sein.

```
pData[0]  Fehlercode
          ERROR_SUCCESS: kein Fehler
pData[1]  Modus
          0=Modul abgeschaltet
          1=Modul aktiv
pData[2]  Frequenz
pData[3]  Modulnummer (Funkrufnummer)
```

11.5 void ClearFtMessageBuffer (void)

Diese Funktion löscht den Messagepuffer im Interface.

11.6 void SetFtMessageReceiveAddress (void *())

Über diese Funktion teilt das Anwendungsprogramm der Firmware die Adresse der Receive-Message Funktion mit. Wird p=0 übergeben, wird keine Funktion bei Empfang einer Nachricht von der Firmware aufgerufen. Bei jeder eingehenden Nachricht wird dann diese Funktion aufgerufen. Die Interface Firmware übergibt dann einen "near"-Zeiger auf die empfangene Nachricht.

11.7 BYTE SendFtMessage (BYTE hwid, BYTE subid, ULONG message, UINT uiWaitTime, UINT uiOption)

Diese Funktion sendet eine Nachricht. Die eigentliche Nachricht besteht aus einem 16-Bit Nachrichten-ID im Lowword des Parameters „message“ und einem 16-Bit Nachrichten-Wert im Highword des Parameters „message“. Über den Parameter hwid wird ein physikalischer Kanal ausgewählt:

MSG_HWID_SELF	**(0)**	**Nachricht an sich selbst senden**
MSG_HWID_SER	**(1)**	**Nachricht an die serielle Schnittstelle senden**
MSG_HWID_RF	**(2)**	**Nachricht über Funk verteilen (aber nicht an sich selbst senden)**
MSG_HWID_RF_SELF	**(3)**	**Nachricht über RF verteilen (auch an sich selbst)**

Über den Subkanal ID kann ein physikalischer Kanal in mehrere logische Kanäle aufgeteilt werden. Erlaubt sind die ID´s 0…219. Die Werte von 220 bis 255 sind für interne Aufgaben reserviert

Der Returnwert ist 0 im Erfolgsfall, sonst eine Fehlernummer. Sofern noch Platz im Puffer ist, kehrt die Funktion sofort zurück, anderenfalls wartet die Funktion maximal die in uiWaitTime angegeben Zeit in ms.

Das Messagesystem benötigt für die Verteilung einer Nachricht etwa 5ms. Da insbesondere innerhalb Schleifen diese Funktion wesentlich öfters aufgerufen werden könnte, kann über den Parameter "uiOption" die Anzahl der zu sendenden Nachrichten optimiert werden. Dadurch kann vermieden werden, dass mehrfach die gleiche Nachricht gesendet wird.

Sollen auch Nachrichten an die serielle Schnittstelle des Interface gesendet werden (bHwId = MSG_HWID_SER), so muß die Betriebsart IF_COM_MESSAGE vor dem Starten des Transfer-Threads mit der Funktion " SetFtDeviceCommMode()" aktiviert werden.

Return Werte:

ERROR_SUCCESS	**Kein Fehler**
FTLIB_ERR_MSG_BUFFER_FULL_TIMEOUT	**TimeOut, Nachricht konnte nicht innerhalb der angegeben Zeit gesendet werden**

Aufruf:	BYTE hwId	Hardware-ID
	MSG_HWID_SELF (0x00):	direkt in den eigenen Receive-Puffer kopiert
	MSG_HWID_SER (0x01):	Über Interface-RS232 gesendet
	MSG_HWID_RF (0x02):	Über Funk nur an andere Module
	MSG_HWID_RF_SELF (0x03):	Über Funk auch an sich selbst
	BYTE subId	Logischer Kanal, erlaubt sind die ID´s 0…219. Die Werte von 220 bis 255 sind für interne Aufgaben reserviert
	ULONG message	Lowword: 16-Bit Nachrichten-ID Highword: 16-Bit Nachricht
	UINT uiWaitTime	Falls der interne Puffer voll ist, kann mit diesem Parameter (in ms) bestimmt werden, wie lange gewartet wird, bis die Funktion wieder zurückkehrt.
	UINT uiOption	Sendeoptionen
	MSG_SEND_NORMAL (0):	Die Nachricht wird direkt in den Sendepuffer geschrieben
	MSG_SEND_OTHER_THAN_LAST (1):	Die Nachricht wird nicht gesendet, wenn eine identische Nachricht (bHwId, bSubId, dwMessage) am Ende des Puffers steht. Ist der Puffer leer, oder es steht eine andere Nachricht am Ende des Puffers, wird die Nachricht gesendet.
	MSG_SEND_IF_NOT_PRESENT (2):	Die Nachricht wird nicht gesendet, wenn eine identische Nachricht (bHwId, bSubId, dwMessage) irgendwo im Puffer steht. Ist der Puffer leer, wird die Nachricht gesendet.
Return:	Error-Code (in R0L)	
	ERROR_SUCCESS	kein Fehler
	FTLIB_ERR_MSG_BUFFER_FULL_TIMEOUT	Puffer ist voll, Nachricht konnte in der angegebenen Zeit nicht übertragen werden

11.8 void FtDelay (UINT)

Diese Prozedur wartet die angegebene Zeit (in ms).

11.9 void SetFtDistanceSensorMode (UCHAR ucMode, UCHAR ucTol1, UCHAR ucTol2, UINT uiLevel1, UINT uiLevel2, UCHAR ucRepeat1, UCHAR ucRepeat2)

Diese Routine initialisiert den D1/D2 Eingang am Interface zum Anschluß des fischertechnik Distanzsensors oder zum Messen von Spannungen im Bereich 0-10 Volt.

Wichtiger Hinweis:
Da die Betriebsart der D1 / D2 Eingänge durch Software eingestellt werden kann, empfehlen wir an diese Anschlüsse keine Spannungen "direkt" einzuspeisen, um Beschädigungen des Interface bei Softwarefehlern zu vermeiden. Da die Eingänge hochohmig sind sollte ein Widerstand von ca.
220 Ohm - 470 Ohm direkt an die D1 / D2 Buchse angeschlossen werden (Reihenschaltung). Wir empfehlen erst "dahinter" die zu messende Spannung anzuschließen.

Aufruf:	FT_HANDLE hFt	Handle des Device
	DWORD dwMode	Betriebsart der Anschlüsse: IF_DS_INPUT_VOLTAGE (0x00) = Eingänge messen Spannungen IF_DS_INPUT_DISTANCE (0x01) = Eingänge für ft-Distanzsensor

Die nachfolgenden Parameter sind abhängig von der eingestellten Betriebsart.
Für dwMode = IF_DS_INPUT_DISTANCE gilt:

DWORD dwTol1	Toleranzbereich D1 empfohlen: IF_DS_INPUT_TOL_STD (20)
DWORD dwTol2	Toleranzbereich D2 empfohlen: IF_DS_INPUT_TOL_STD (20)
DWORD dwLevel1	Schwellwert D1
DWORD dwLevel2	Schwellwert D2
DWORD dwRepeat1	Wiederholungswert D1 empfohlen: IF_DS_INPUT_REP_STD (3)
DWORD dwRepeat2	Wiederholungswert D2 empfohlen: IF_DS_INPUT_REP_STD (3)

Der Distanzsensor arbeitet mit Infrarot - Licht und kann daher durch äußere Einflüsse gestört werden (z.B. IR-Handsender). Um diese Störungen auszuschließen, kann durch die Angabe des Wiederholungswertes festgelegt werden, wie oft der "gleiche" Messwert gemessen werden muß, bis dieser als gültig erkannt wird. Da die Messwerte nun von einer zur nächsten Messung etwas schwanken können, gibt es den Toleranzbereich. Sobald eine neue Messung beginnt, dürfen sich die nachfolgenden Messwerte innerhalb dieses "Fensters" verändern, ohne dass es zu einem Neustart der Messungen kommt. Der Schwellwert bestimmt den Level für die Auswertung als "digitale" Abstandssensoren. Unterhalb des Schwellwertes wird eine logische "0", darüber eine "1" gemeldet.
In der TransferArea werden die ermittelten Zustände von der Firmware in den Speicherstellen "Base+0x0C" (digital) und "Base+0x1C / Base+0x1E" (analog) gespeichert.

12 Kommunikation

Mehrere Robo Interfaces können über die RS232-Schnittstelle und die Funkschnittstelle untereinander Nachrichten austauschen. Die Nachrichten bestehen grundsätzlich aus einem 16-Bit Nachrichten-ID und einem 16-Bit Wert.

Die Kommunikation erfolgt auf Funk-Kanälen (Frequenzen). Es können maximal acht Funk-Interfaces auf dem gleichen Funk-Kanal Nachrichten austauschen. Jedes Modul hat dabei eine eigene "ID" Nummer, die im Bereich 1..8 eingestellt werden kann. Das PC-Modul hat immer die ID Nummer "0". Zusätzlich können Nachrichten auch an die serielle Schnittstelle des Interface gesendet und von dieser empfangen werden.

Jeder Funk-Kanal ist in 256 logische Unterkanäle unterteilt, die zur Strukturierung der Kommunikation dienen. Eine Nachricht wird dabei immer an alle Teilnehmer verteilt ("Broadcasting"). Jede Nachricht besteht aus fünf Datenbytes:

SubId	Nummer des Unterkanals (1 Byte)
Nachricht	Nachrichtendaten (4 Bytes)
	B1:B0: Nachrichten ID (Low-Word)
	B3:B2: Nachricht (High-Word)

Es gibt keine Rückmeldung an den "Absender", ob die Nachricht empfangen wurde.

Gesteuert wird die Kommunikation vom PC-Modul, das als Message-Router arbeitet.

12.1 Serielle Nachrichten

C-Programme können Nachrichten auch über die serielle Schnittstelle senden und empfangen. Jede Nachricht wird einzeln über die serielle Schnittstelle gesendet. Dazu muss die Betriebsart im Interface mit der Funktion "SetFtDeviceCommMode()" geändert werden. Die geänderte Betriebsart ist dann durch das dauerhafte Leuchten der COM-Leuchtdiode am Interface erkennbar. Durch einen Tastendruck auf den Port-Taster wird die Betriebsart wieder zurückgesetzt, selbstverständlich kann dies auch durch die Software erfolgen. Es ist also möglich, zwischen zwei Robo-Interfaces Nachrichten über die serielle Schnittstelle auszutauschen. Das dazu passende Verbindungskabel (X-Kabel) kann über den fischertechnik-Einzelteilservice bezogen werden.

12.2 Firmwareunterstützung

Die Interface Firmware bietet die Funktion "SendFtMessage()" für den Datentransport an. Beim Senden kann über eine Hardware-Id ein physikalischer Kanal ausgewählt werden. Falls kein zweites Interface oder keine Funkschnittstelle zur Verfügung stehen, kann man die Nachricht "an sich selbst" senden. Man kann die Nachricht an die serielle Schnittstelle senden oder über Funk (RF) an andere Module durch den Message-Router verteilen lassen.

12.3 Nachrichtenempfang

Sobald das Interface eine Nachricht empfängt, ruft es eine Callback-Routine auf und übergibt dieser einen "near"-Zeiger auf die empfangene Nachricht. Durch die Funktion "SetFtMessageReceiveAddress()" kann der Firmware die Adresse dieser Nachrichtenbearbeitungsfunktion übergeben werden. Im Lieferumfang des C-Compilers gibt es dazu ein Beispiel. Die Nachrichtenbearbeitungsfunktion speichert die Nachricht in einem Puffer des C-Programms ab. Das C-Programm holt sich dann zu einem späteren Zeitpunkt die Nachricht aus dem Zwischenspeicher.

13 Debugging

Das Debuggen von Programmen im Interface ist z.Zt. nicht möglich, da Renesas die Steuercodes der Debuggerschnittstelle für seine Werkzeuge nicht veröffentlicht.

Als Alternative können Speicherbereiche im Interface RAM über den FtLoader ausgelesen werden. In der TransferArea sind dafür die Variablen "ucDbg1F0" … "ucDbg1FF" (Base+0x1F0..0x1FF) vorhanden. Auch das Setzen von Werten im RAM ist möglich, nachdem ein Programm über den FtLoader gestartet wurde.

14 Revision

Version 0.65:
- Komplette Überarbeitung
- Umbenennung ClearFtMessagePuffer() in ClearFtMessageBuffer()

Version 1.66a:
- Umbenennung von 0.65 in 1.66a
- Überarbeitung des Speicherlayoutes
- Unterstützung der Interface-Firmware 01.66.00.03