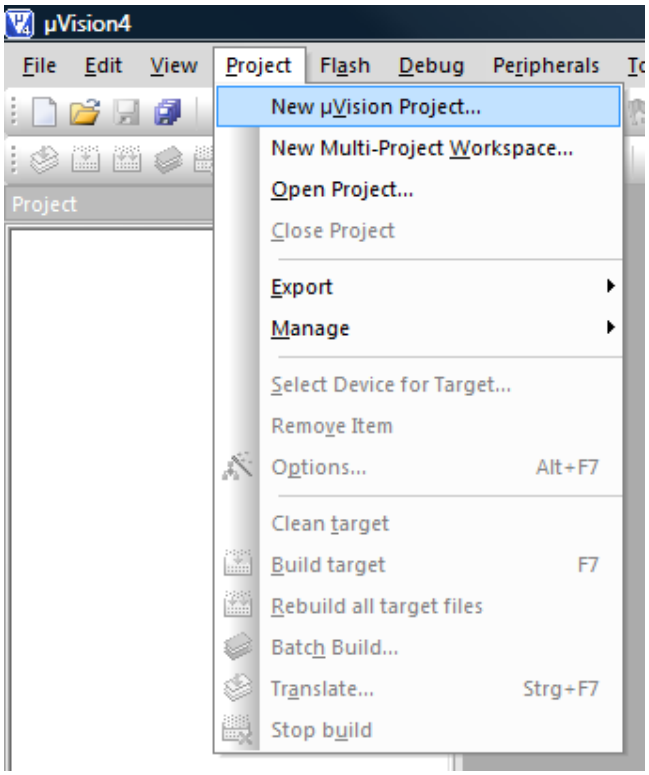
KURZBESCHREIBUNG ZUM ERSTELLEN EINES PROJEKTES MIT µVISION4

NEUES PROJEKT ANLEGEN

Unter **Project → New µVision Project…** kann ein neues Projekt angelegt werden.

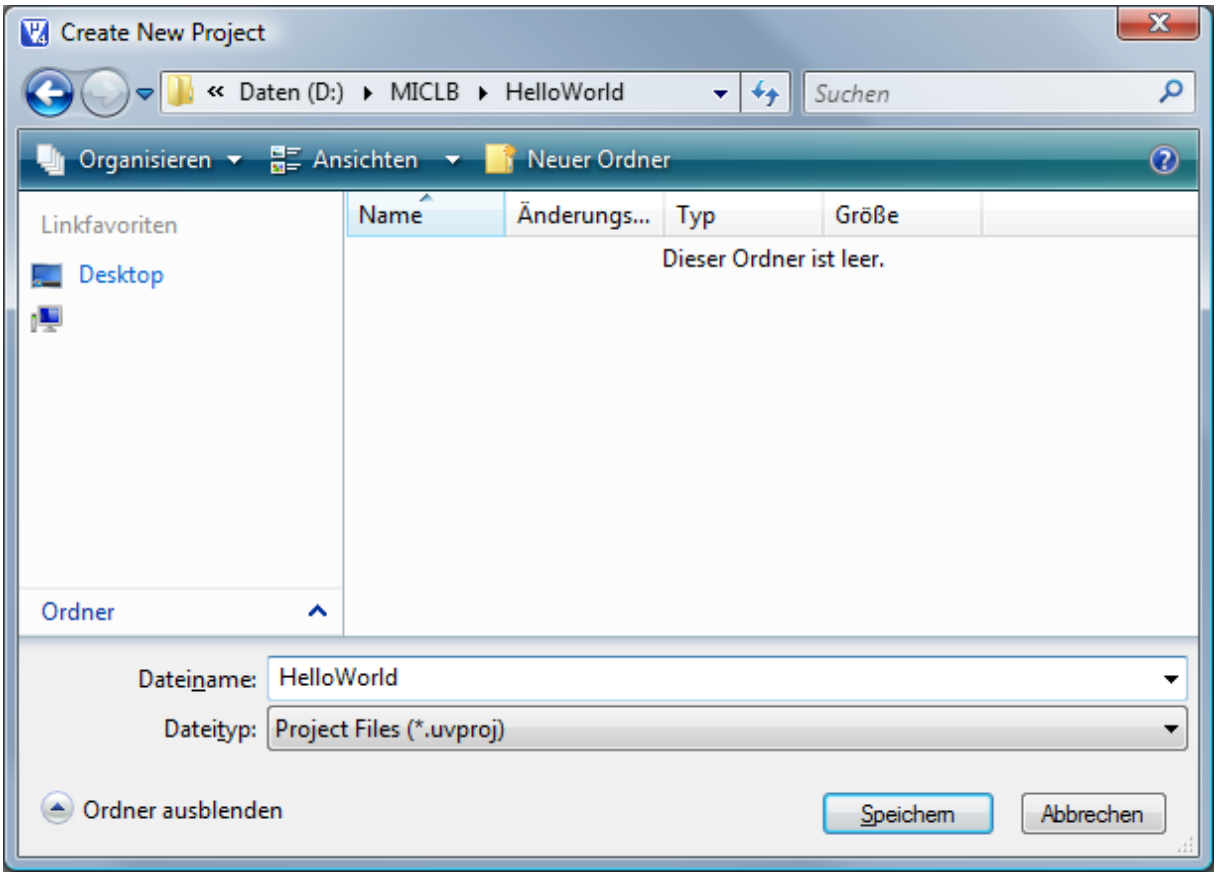


Es öffnet sich ein Dialogfenster, in dem der Speicherort des Projektes angegeben werden kann. Für das Projekt wird ein neues Verzeichnis mit dem Namen **MICLB** und ein Unterordner **HelloWorld** erstellt. Dies stellt das Projektverzeichnis dar, worunter auch die Projektdatei **HelloWorld** gespeichert wird.

Wichtig:

Der Dateipfad inklusive Dateinamen sollte für einen reibungslosen Betrieb möglichst kurz gehalten werden und **keine Leer- bzw. Sonderzeichen** (insbesondere Umlaute) enthalten.

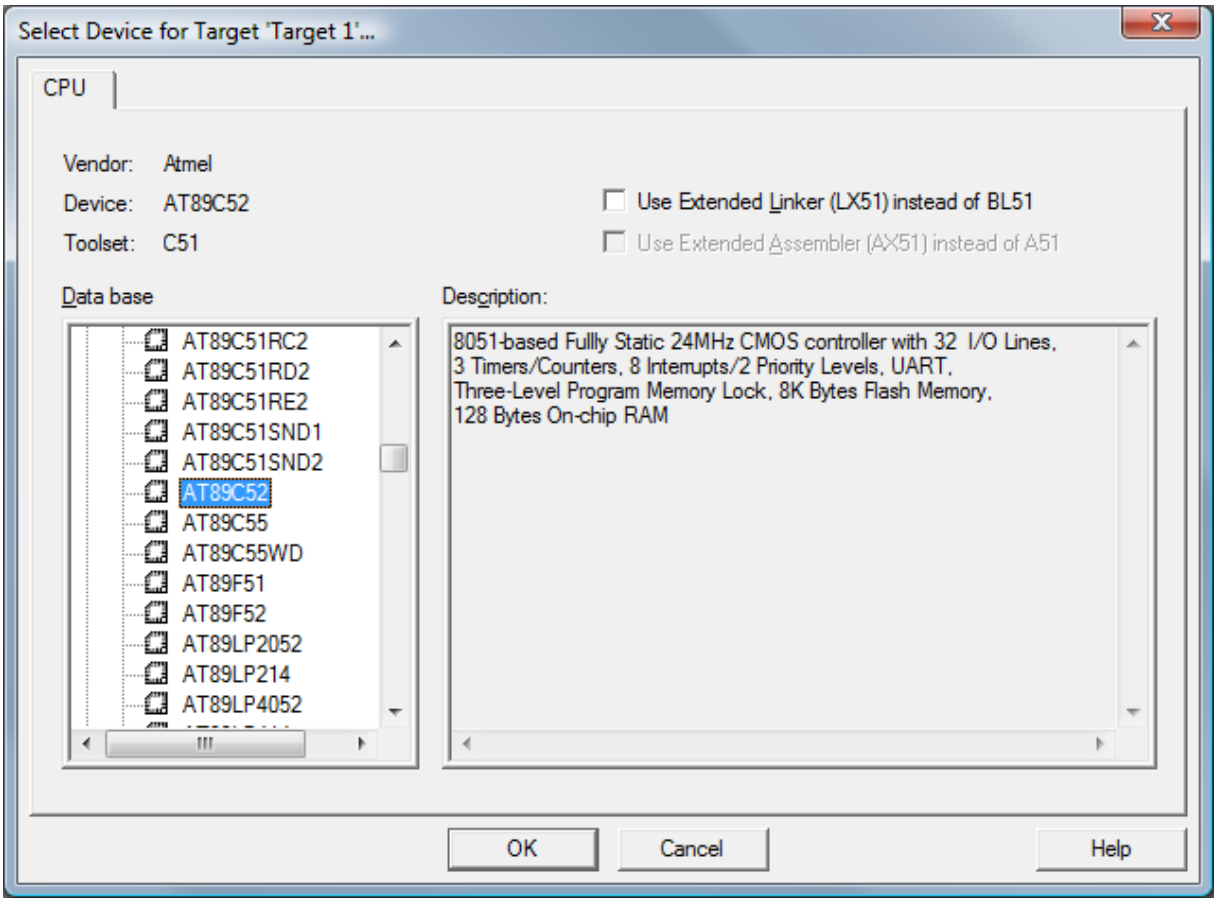
Auf den Laborrechnern sollte generell die Partition **D:** oder ein externes Speichermedium als Speicherziel gewählt werden, um Probleme mit Zugriffsrechten zu vermeiden.



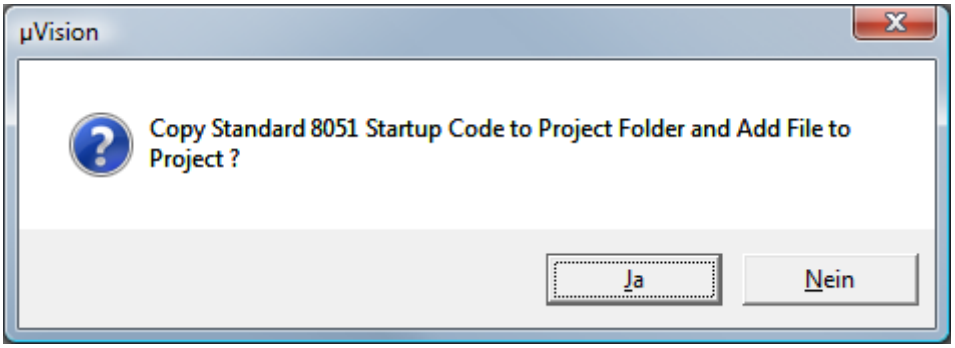
Nach dem Speichern öffnet sich ein Dialogfenster zur Auswahl des Prozessortyps. Die im Labor verwendete Keil µVision4-IDE verwendet den C51-Compiler. Alle angeführten Typen in der Datenbank basieren somit auf der 8051-Architektur. Andere Architekturen (beispielsweise ARM6/7 erfordern eine gesonderte Version des Keil µVision4-Compilers.

Im Labor werden zwei abgewandelte Versionen des **80C51**-Mikrocontrollers (vgl. Vorlesung) verwendet. Hierbei handelt es sich zum einen um den **AT89C52** und zum anderen um den **AT89C51SND1**, beide von der Hersteller-Firma **Atmel** (Kürzel AT am Anfang der Mikrocontroller-Bezeichnungen). Diese unterscheiden sich vom „Ur"-80C51 durch herstellerspezifische Merkmale, die vor allem bei der Programmierung beachtet werden müssen. Darunter fallen insbesondere abgewandelte Registernamen, wodurch die in der Vorlesung behandelten Codebeispiele nicht direkt auf die im Labor verwendeten Mikrocontroller anwendbar sind.

Da in den ersten Laborübungen zunächst im Keil µVision-Simulator gearbeitet wird und anschließend der AT89C52 selbst am Steckbrett in Betrieb genommen wird, wird dieser im folgenden Beispiel als Typ herangezogen.

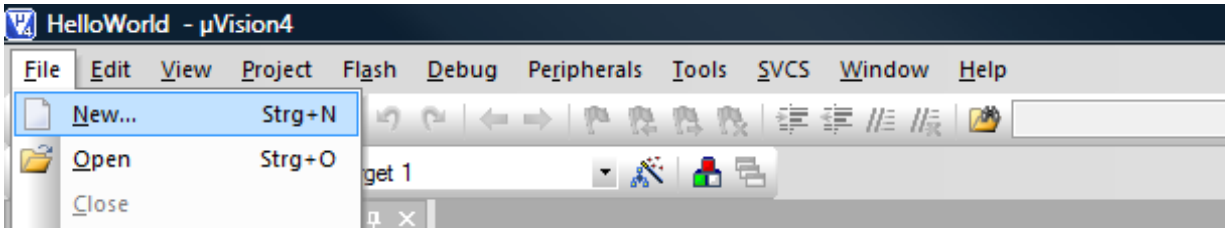


Nach Bestätigung des Prozessortyps durch Drücken von **OK** wird die Auswahl übernommen. Die anschließende Frage, ob der Startup-Code in das Projektverzeichnis kopiert und dem Projekt hinzugefügt werden soll, ist mit **Ja** zu bestätigten. Der Startup-Code initialisiert den Mikrocontroller bei der Inbetriebnahme.

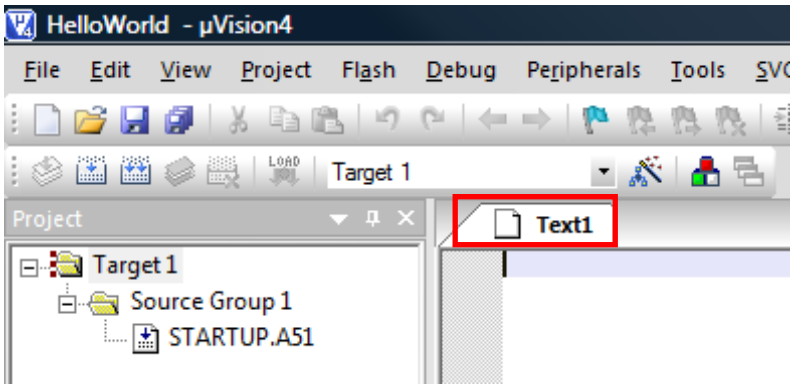


NEUES SOURCE-FILE ERSTELLEN

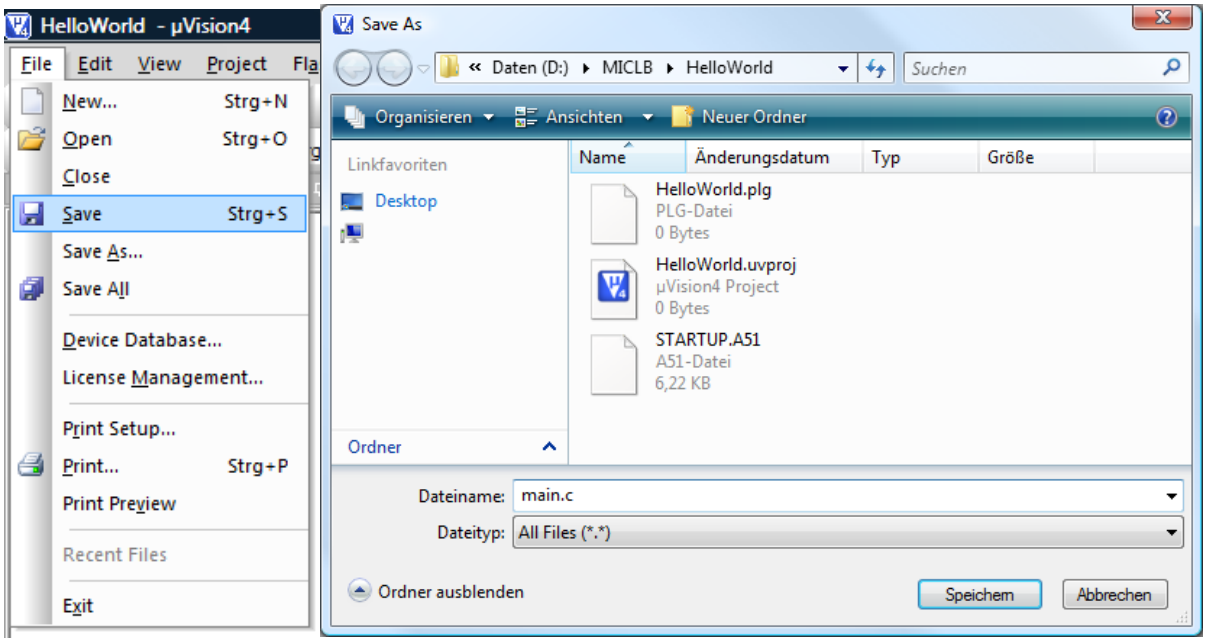
Unter dem Punkt **File → New** wird eine neue, leere Textdatei erstellt, welche in dem Editorfenster bearbeitet werden kann.



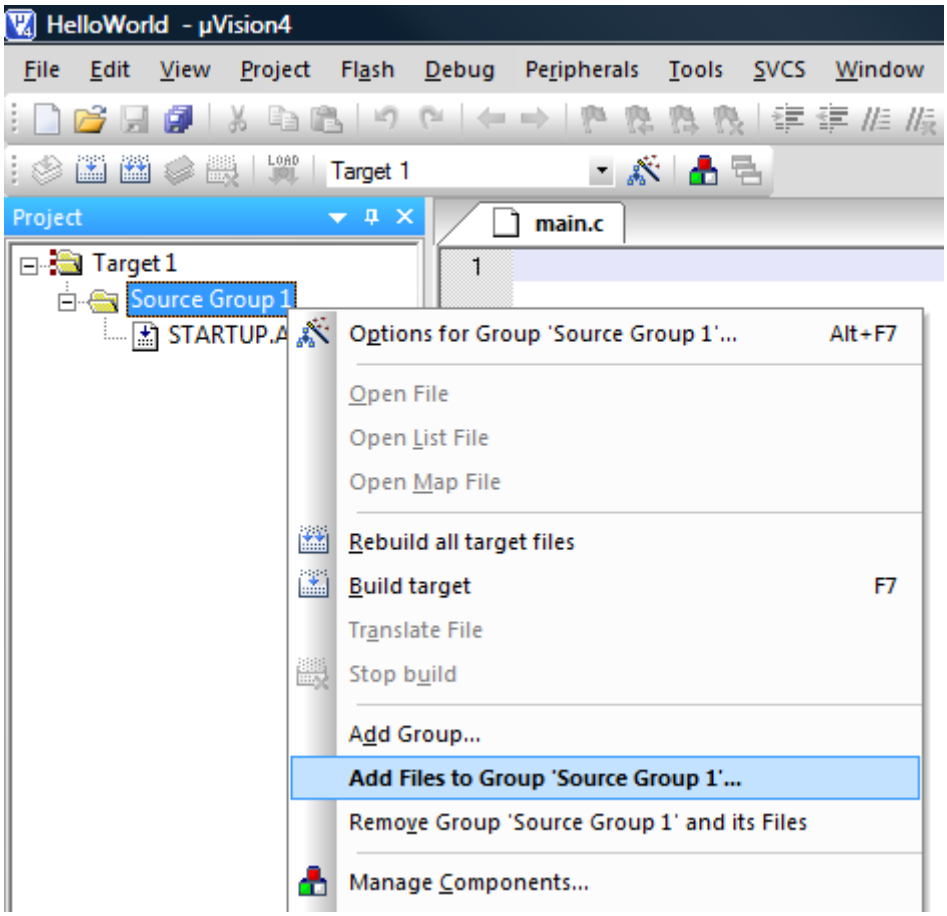
Ein neues leeres Dokument wird als Reiterkarte innerhalb der IDE geöffnet.



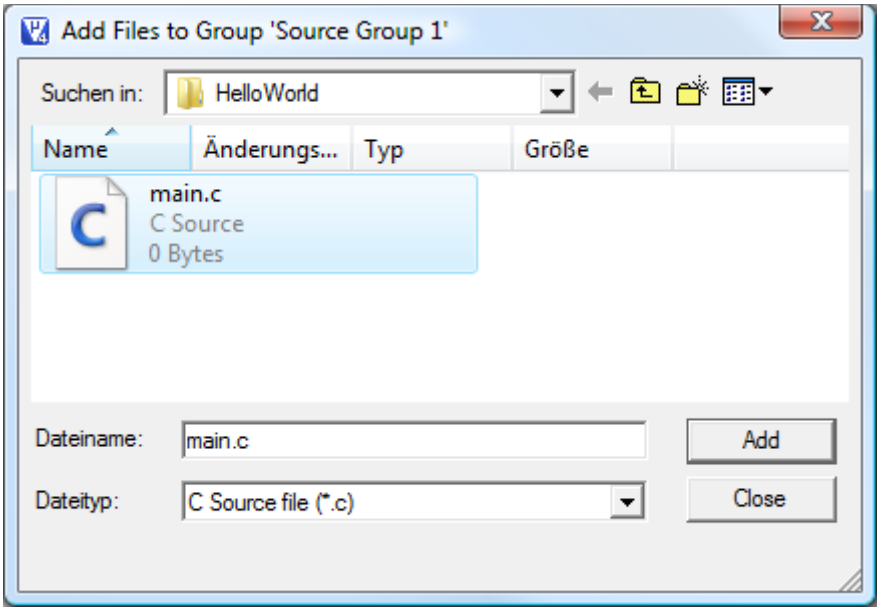
Dieses leere Dokument muss explizit als Datei mit der Endung **.c** abgespeichert werden, damit der Compiler diese als Source-Datei erkennt. Der Dateiname kann beliebig lauten, sollte zur Vermeidung von etwaigen Problemen möglichst kurz gehalten werden und keine Sonderzeichen (insbesondere Umlaute) beinhalten. Sinngemäß wird das Hauptprogramm unter dem Dateinamen **main.c** gespeichert.



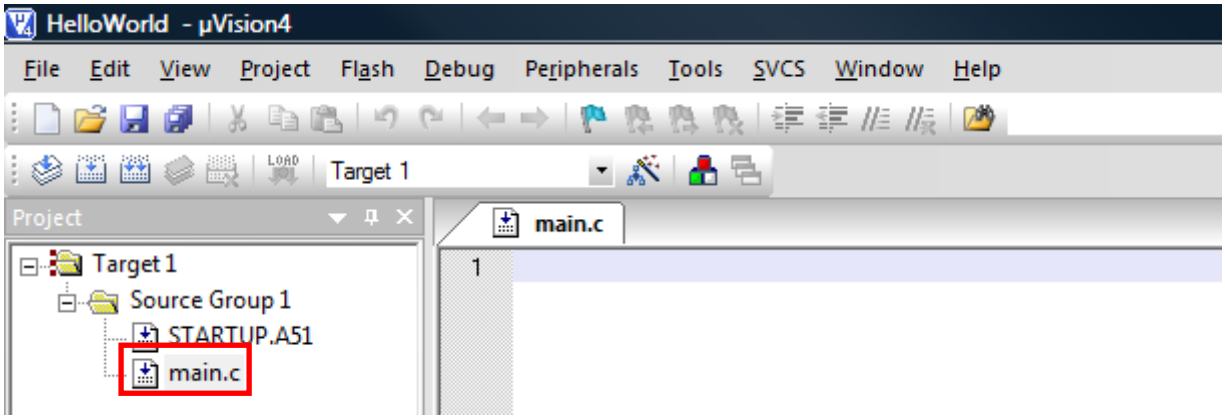
Zudem muss die C-Datei explizit in das Projekt eingebunden werden. Dies erfolgt durch einen Rechtsklick auf den Baumknoten „Source Group 1“ innerhalb der Projekt-Baumstruktur. Mit dem Punkt **Add Files to Group 'Source Group 1'…** ist es möglich, Dateien mit dem Projekt zu verknüpfen. Durch Hinzufügen zum Projekt werden Texte innerhalb von Quellcode-Dateien als ANSI-C-Quellcode erkannt und das Syntax-Highlighting aktiviert.



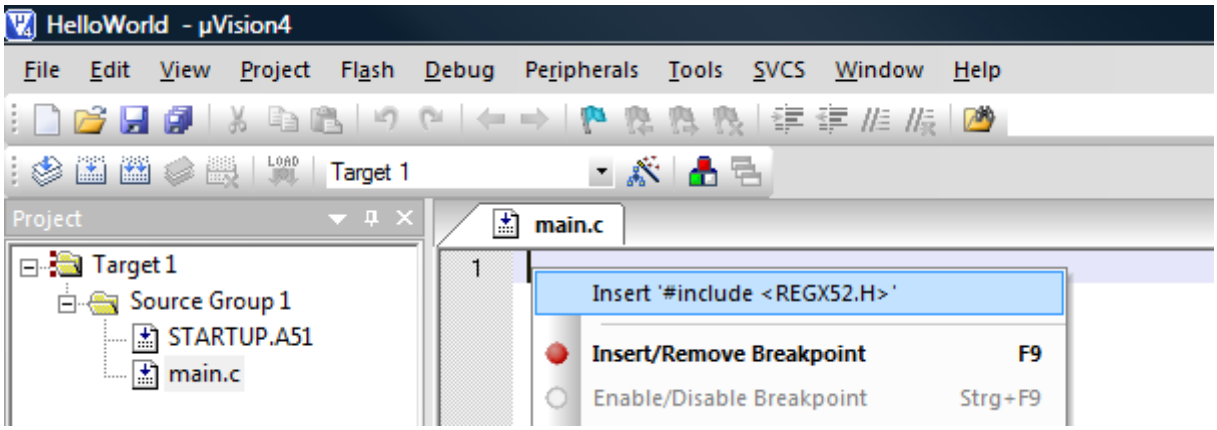
Die zuvor erstellte Quellcode-Datei kann somit in das Projekt eingebunden werden.



Nach erfolgreicher Einbindung scheint diese innerhalb des Projektbaumes auf.

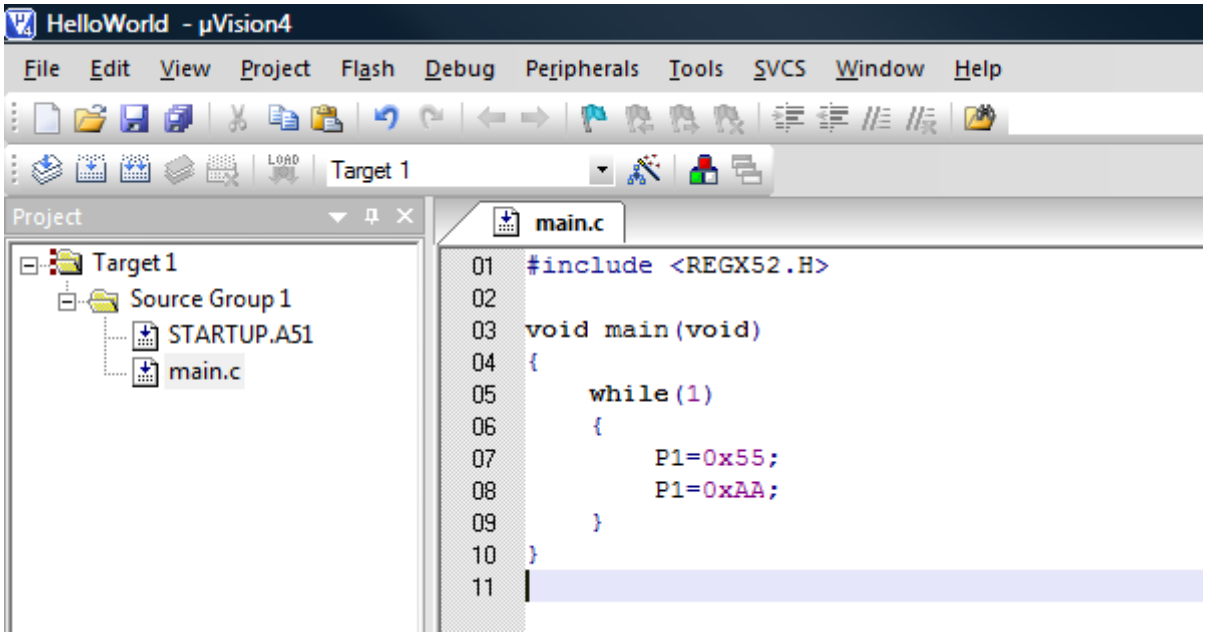


Für die Programmierung des Mikrocontrollers sind hardware- und herstellerspezifische Register und Speicheradressen notwendig. Diese werden von Keil in Form von Include-Dateien zur Verfügung gestellt. Ein Rechtsklick in das Editorfenster der Quellcodedatei öffnet einen Dialog, der als ersten Punkt die Option zum Einfügen der modellspezifischen Include-Datei bereitstellt. Diese Option ist abhängig vom Mikrokontroller-Modell, das bei der Projekterstellung ausgewählt wurde. In diesem Fall lautet die entsprechende Include-Datei **REGX52.H**.

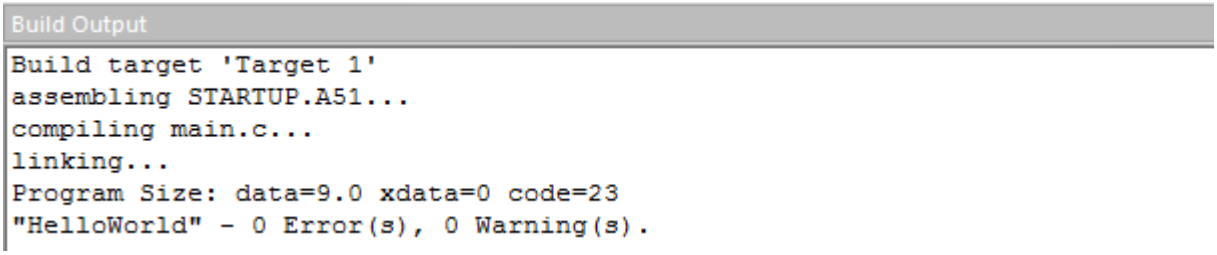


Zu Testzwecken und zur näheren Erläuterung der IDE wird folgender Beispielcode herangezogen:

```
void main(void)
{
	while(1)
	{
		P1=0x55;
		P1=0xAA;
	}
}
```

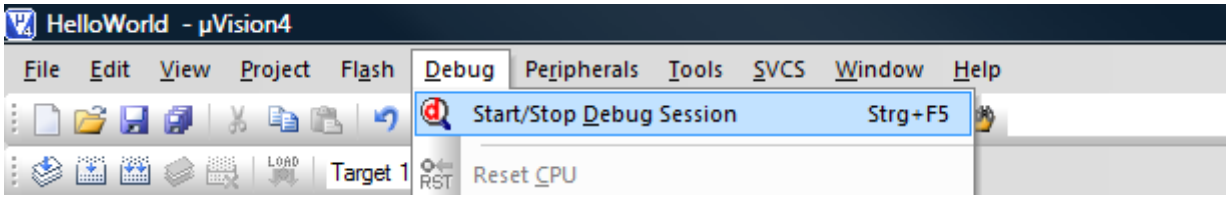


Nun kann der Quellcode kompiliert werden. Unter **Project → Build Target** wird der Kompilierungsvorgang gestartet (alternativ: Funktionstaste **F7** drücken). Der Kompilierungs-Status ist im **Output-Fenster** (links unten in der µVision-IDE) ersichtlich.

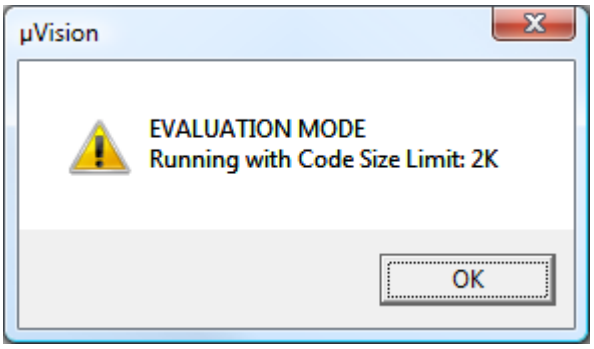


DEBUGGING

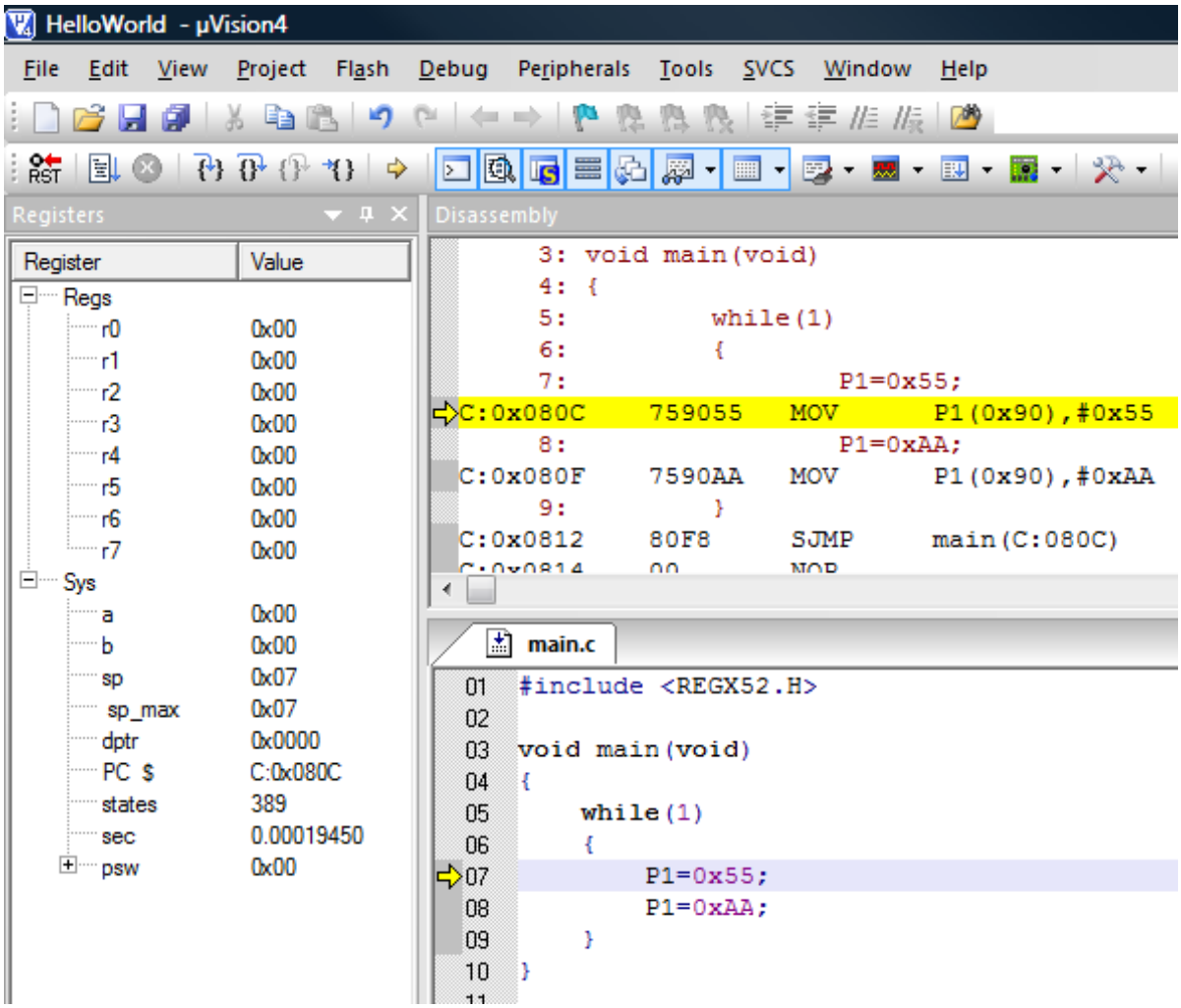
Mit **Debug → Start/Stop Debug Session** wird der Debugger gestartet.



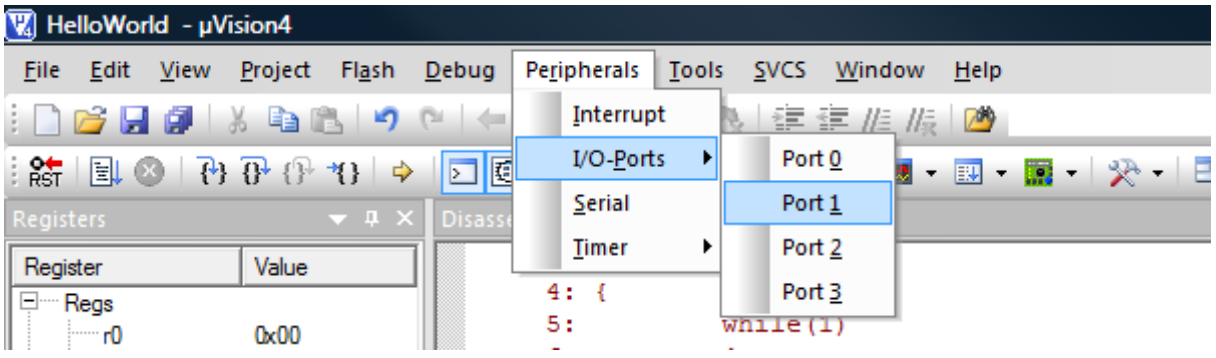
Die Evaluations-Lizenz von Keil µVision2 beschränkt die Quellcodegröße auf 2 Kilobyte. Für einzelne Laborübungen stellt dies eine nicht relevante Einschränkung dar. Für die Kompilierung und das Debugging des Gesamtprojekts ist mitunter eine lizenzierte Version der Keil µVision-IDE erforderlich, die im Labor zur Verfügung gestellt wird. Abhilfe schafft die Aufspaltung des Gesamtquellcodes in einzelne Quellcodedateien und Ein- und Ausbindung der benötigen Dateien im Projekt. Einzelne Module können somit kompiliert und auf Fehler untersucht werden, ohne eine lizenzierte Version zu benötigen.

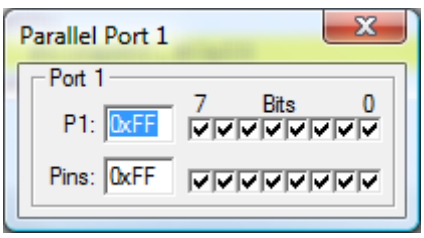


Nach Quittierung des Einschränkungshinweises zeigt ein gelber Pfeil im Editorfenster die aktuell abzuarbeitende Codeanweisung an. Zudem wird oberhalb die Disassembly-Ansicht des Quellcodes eingeblendet, die ebenfalls den aktuellen Abarbeitungsschritt anzeigt.

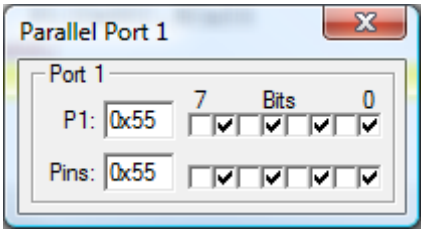


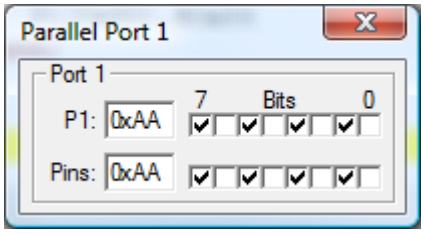
Unter **Peripherals → I/O Ports → Port1** kann der im Programm verwendete, simulierte Port 1 als I/O-Port betrachtet werden. Die Ports des Mikrocontrollers werden standardmäßig mit 0xFF initialisiert. Da die Codeanweisung noch nicht abgearbeitet wurde, befindet sich Port 1 noch im Default-Zustand.





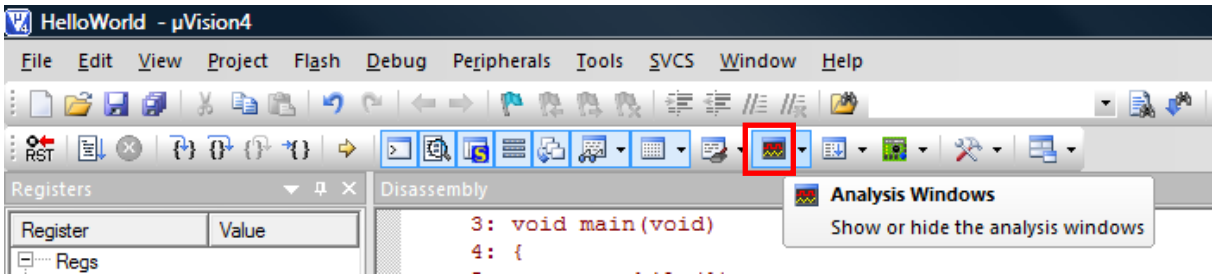
Durch Drücken der Funktionstaste **F10** erfolgt eine Einzelschrittabarbeitung, wodurch ein Togglen der Pins auf Port 1 erfolgt.



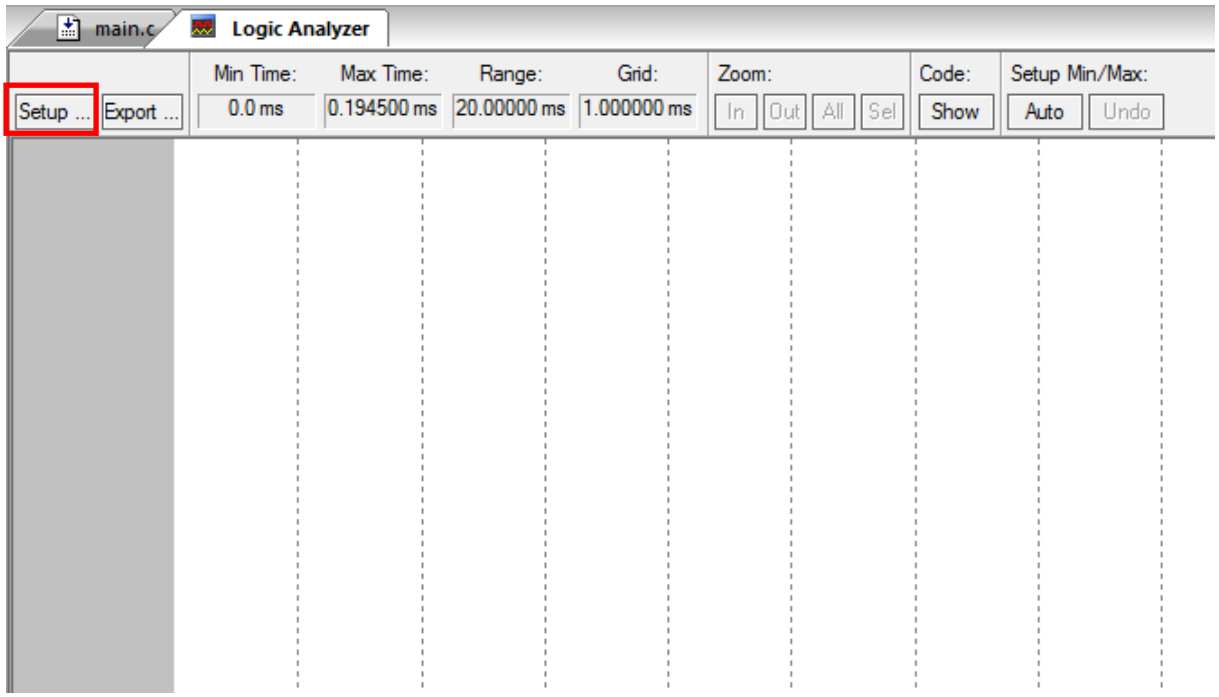


Die Funktionstaste **F5** lässt den Debugger den Quellcode vollständig abarbeiten und stoppt nur bei gesetzten Breakpoints. Die Abarbeitung erfolgt dabei in Echtzeit, wodurch schnelle Wechsel auf simulierte Peripherie nicht schnell genug von der IDE aktualisiert werden bzw. nicht für das menschliche Auge sichtbar sind. Hierbei empfiehlt es sich, entsprechende Breakpoints zu setzen bevor bzw. nachdem ein Wechsel der Peripherieeigenschaften erfolgt.

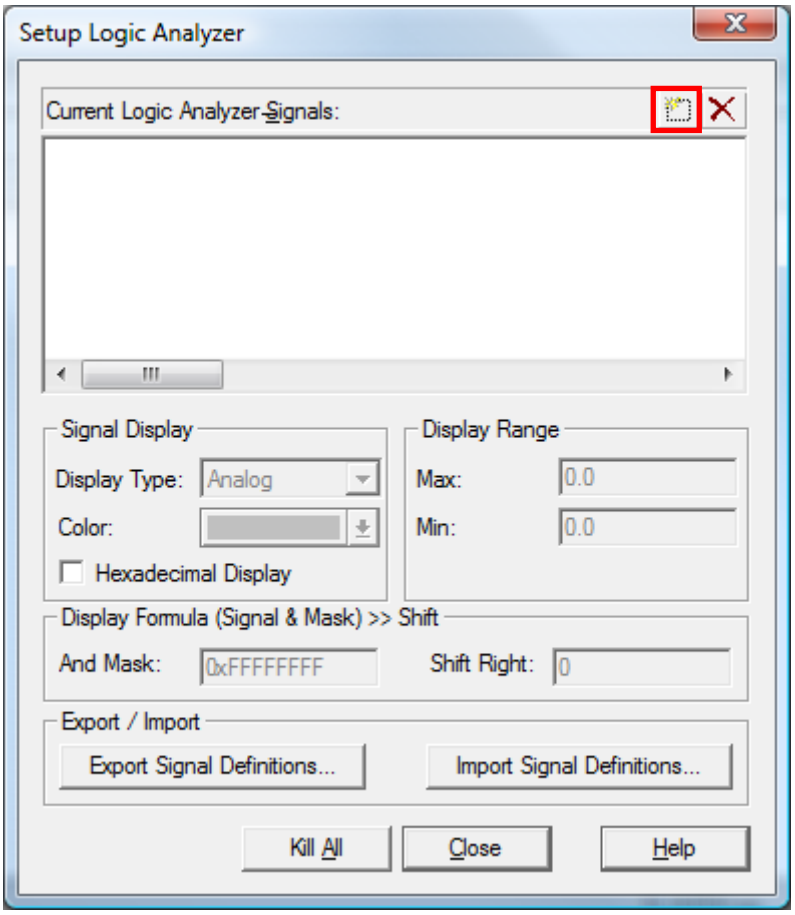
Eine weitere Visulisierungsmethode wird in Form des Logic Analyzers zur Verfügung gestellt. Dieser wird durch Klick auf das Icon bzw. **View → Analysis Windows → Logic Analyzer** aktiviert und als Registerkarte hinzugefügt.



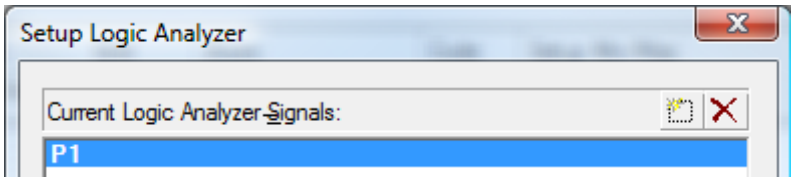
Über den Punkt Setup ist es möglich, dem Logic Analyzer entsprechende Signale hinzuzufügen.



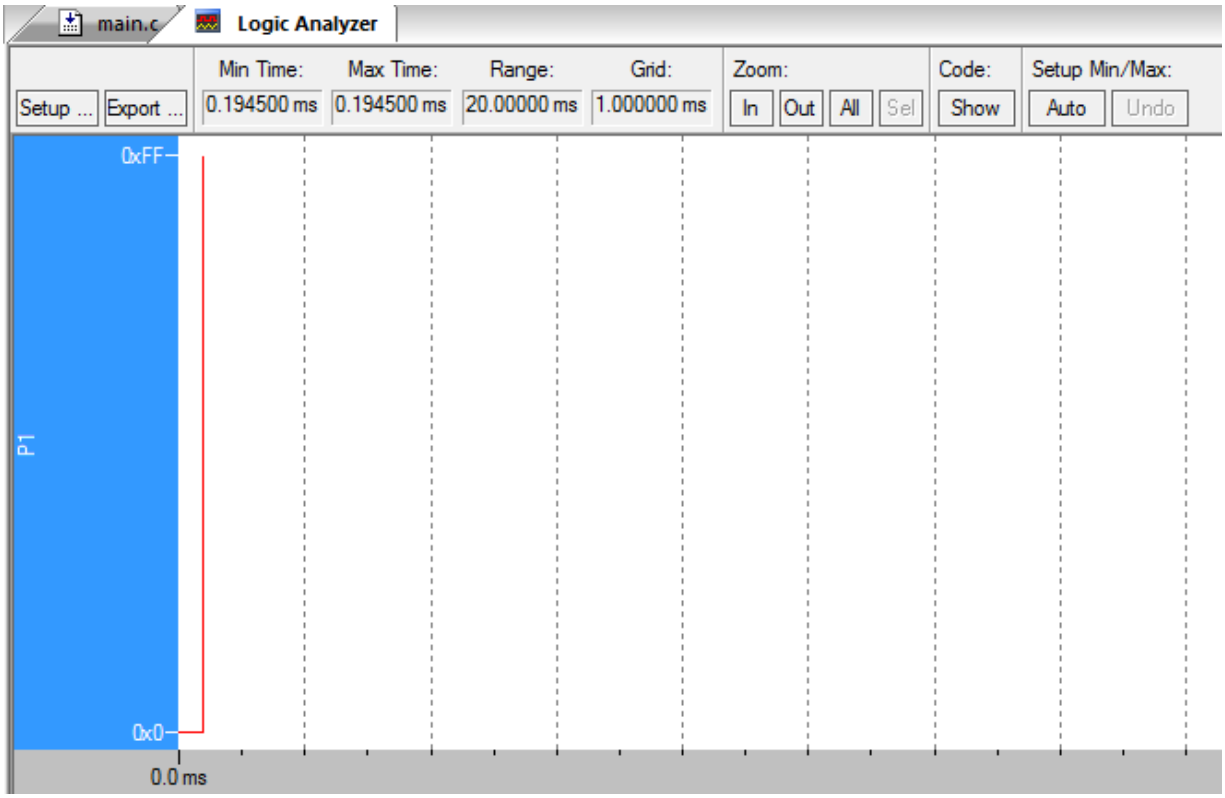
Ein Dialogfenster dient dabei zur Konfiguration des Logic Analyzers. Neue Signale können durch Klick auf das rot markierte Symbol hinzugefügt werden. Dabei ist zu beachten, dass nur im Programm verwendete Variablen und Ports bzw. Pins als Signalverlauf angezeigt werden können.



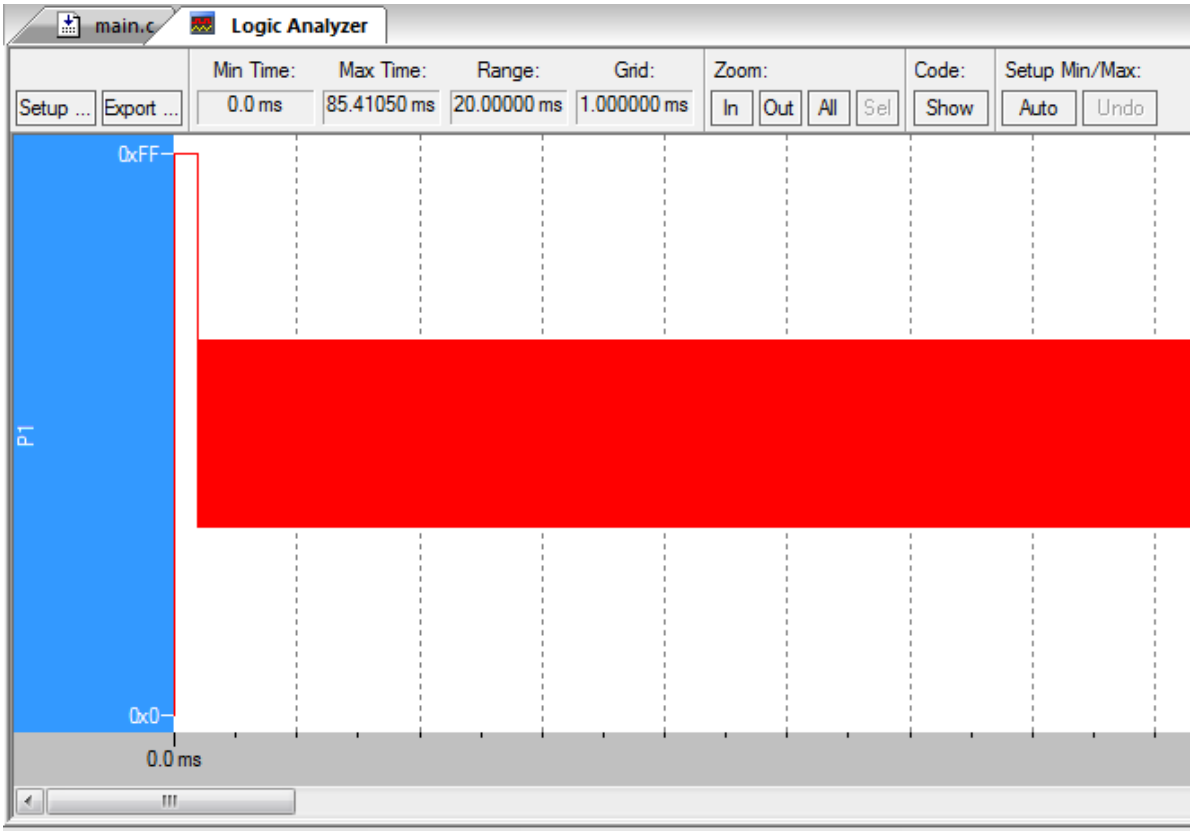
Im gegebenen Beispielcode soll der Port 1 (P1) als Signal untersucht werden. Dabei ist zu beachten, dass Port 1 ein 8 Bit breiter Port ist und somit der Gesamtwert als Signalwert dargestellt wird und nicht der eigentliche Status der einzelnen Pins. Soll der Status einzelner Pins untersucht werden, ist die Notation P1_0 (für P1.0), P1_1 (für P1.1) etc. erforderlich, die entsprechende Verwendung im Programm vorausgesetzt.



Nach Bestätigung der Eingaben durch Klicken auf Close kann der bisherige Signalverlauf im Logic Analyzer betrachtet werden. Nachfolgend ist ersichtlich, dass der Port zum Zeitpunkt 0 auf 0x00 gesetzt war und der Status zum Zeitpunkt 194,5µs auf 0xFF wechselte. Diese Initialisierung des Ports erfolgt durch den Startup-Code, der bei der Projekterstellung mit eingebunden wurde. Dadurch wird der Port als Input-Port initialisiert.

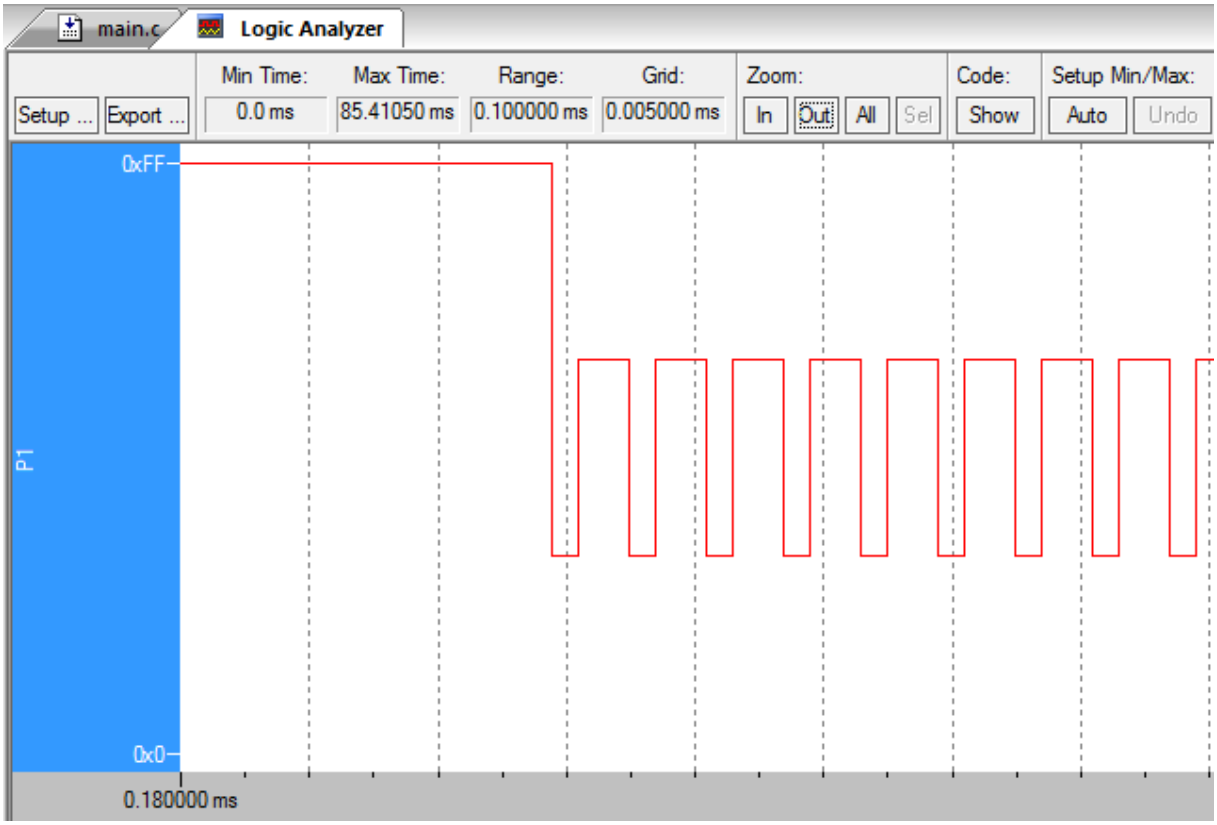


Die Abarbeitung des Quellcodes (Funktionstaste F5) im Programm bewirkt die Veranschaulichung des Signalverlaufs im Logic Analyzer.

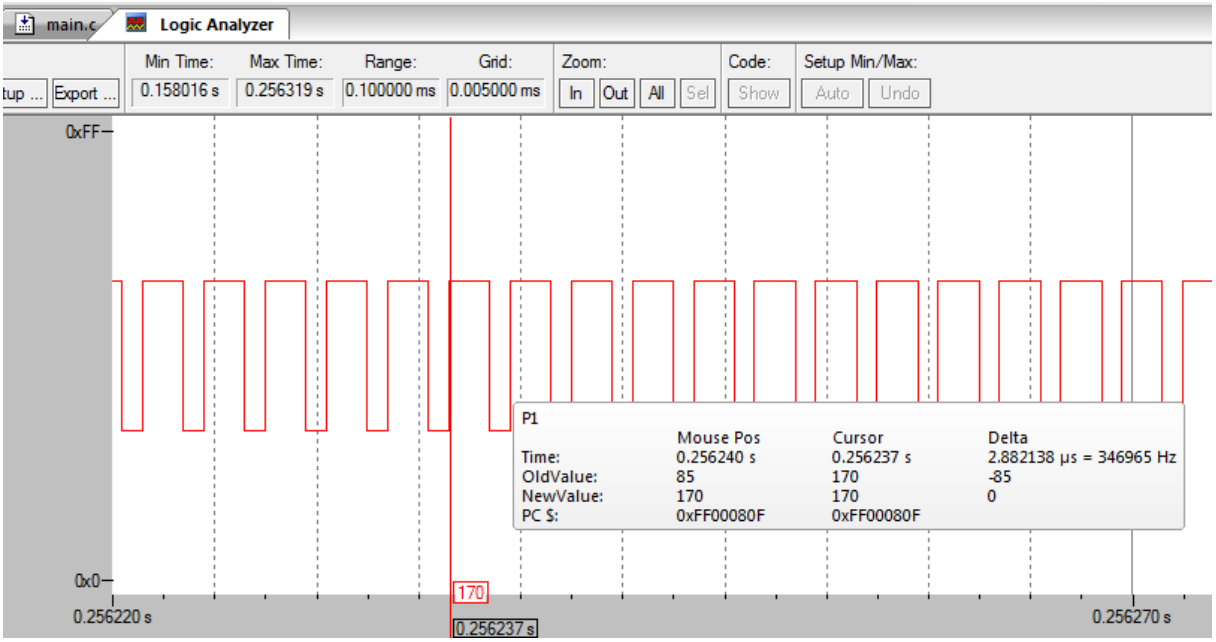


Wie bereits erwähnt erfolgt die Abarbeitung der Instruktionen in Echtzeit, wodurch ein schneller Wechsel des Status von Port 1 erfolgt, da keine Verzögerung im Quellcode implementiert ist. Der

dargestellte Signalverlauf spiegelt dies wider. Erst ein Klicken auf **Zoom In** zeigt eine feinere Darstellung des zeitlichen Signalverlaufs.



Mittels des Cursors, der durch Linksklick gesetzt werden kann, sowie durch den Mauszeiger können Abstände und Frequenzen im Logic Analyzer gemessen werden.



FERTIG!