

# Laborversuch 1

## Teil 1

### **Signalgenerator (Sinus)**

*Ausgabe eines Sinustones über den Stereo-Codec TLV320AIC23B auf dem DSP Starter Kit DSK6713 von TI.*

*Die auszugebenden Sinuswerte liegen als Gleitkommazahlen im Bereich -1 ... +1 in einer z.B. 100 Elemente großen Tabelle im Speicher (sog. Look-up-table).*

## **1. Einführende Informationen**

### **1.1 Allgemeines**

Der erste Laborversuch dient dem **Einstieg in die digitale Signalverarbeitung mit DSPs** (Digitalen Signalprozessoren). Anhand eines vorgegebenen, auf dem DSP Starter Kit DSK6713 von Texas Instruments lauffähigen Projekts, wird Ihnen die Möglichkeit gegeben, sich mit der Entwicklungsumgebung Code Composer Studio und dem Demoboard vertraut zu machen.

Um Ihnen den Einstieg zu erleichtern, haben wir für Sie das zu Beginn Wesentliche in einigen Infoblättern zusammengestellt. Die Lektüre der **Infoblätter** vor dem ersten Labortermin wird **dringend** empfohlen.

Zusätzliche Informationen zum DSP TMS320C6713 und zur Entwicklungsumgebung CCS können Sie natürlich in zahlreichen PDF-Dokumenten, die TI über das Internet zur Verfügung stellt, erhalten. Weiter steht Ihnen auf den Laborrechnern das Hilfesystem von Code Composer Studio zur Verfügung.

Die **Programmierung des DSPs** erfolgt im Signale & Systeme - Labor ausschließlich in **Standard-C** (Grundlagenliteratur: Kernighan / Ritchie, Programmieren in C). Der C-Compiler setzt dann Ihren C-Code mehr oder minder effizient in Assembler-Code um.

Natürlich könnten Sie auch direkt Assembler-Code erstellen bzw. eine Kombination von C und Assembler verwenden. Für spezielle Programnteile, bei denen es auf eine besonders schnelle Ausführung ankommt, kann das vorteilhaft sein.

### **1.2 Signalgenerierung mit Hilfe einer Look-Up-Table**

Das erste DSP-Programm gibt einen Sinus-Dauerton über den Analogausgang des Demoboards aus. Dieser Ton kann z.B. über einen Kopfhörer abgehört werden.

Der Ton entsteht durch die fortlaufende Ausgabe von Amplitudenwerten einer Sinusschwingung über den McBSP1 des DSPs an den Codec auf dem Demoboard. Die Sinuswerte liegen dabei als Gleitkommazahlen (Zahlenbereich: -1.0 ... +1.0, Datentyp: `float`) in Form einer Tabelle im Speicher des Demoboards.

**SS 2013**

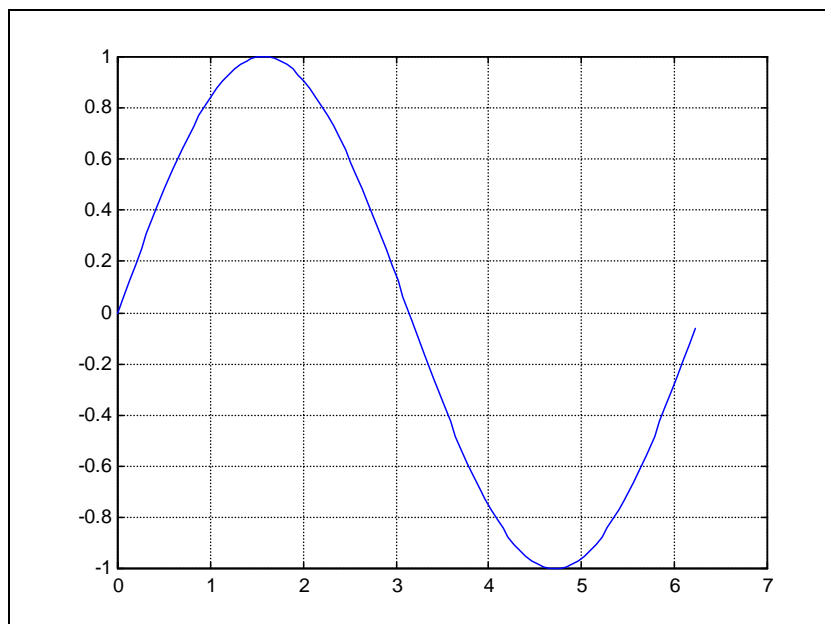
Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker

Eine solche Sinustabelle kann z.B. mit Hilfe von MATLAB sehr einfach erstellt und entsprechend formatiert in einer Datei gespeichert werden. Durch folgendes MATLAB m-File haben wir für Sie eine Sinustabelle mit 100 Einträgen erzeugt:

```
cnt_samples=100
x_vektor=[0:(2*pi/cnt_samples):(2*pi-2*pi/cnt_samples)];
y_vektor=sin(x_vektor);
figure(1)
plot(x_vektor,y_vektor)
% Ausgabe als Spaltenvektor mit angehängtem Komma:
sin_werte=y_vektor.';
fid=fopen('sin_tab100.dat','w');
fprintf(fid,'%1.7e,\n',sin_werte);
fclose(fid);
```

Die so erzeugte Datei wurde um eine entsprechende Variablendeklaration erweitert und dann als **Header-Datei** *sin\_tab\_100.h* in den C-Quellcode eingebunden (siehe Projekt1 – *sinetone1.c*).



**Abbildung 1.1:** Graphische Darstellung der Sinustabelle

Die **Abtastrate** des Codecs auf dem Demoboard wird bei der Initialisierung auf **32 kHz** eingestellt (siehe Infoblatt ‚DSP Starter Kit DSK6713‘, Kapitel ‚Schnittstelle des DSK zur analogen Welt‘). Pro Sekunde werden also 32000 Werte analoggewandelt, d.h. die Tabelle wird bei einer Schrittweite von 1 genau 320 mal pro Sekunde durchlaufen.

### 1.3 Fragen

- F 1.1 Welche Frequenz wird demnach die analoge Sinusschwingung am D/A-Wandlerausgang haben?
- F 1.2 Welche Frequenz hat sie, wenn die Sinustabelle mit einer Schrittweite von z.B. 3 durchlaufen wird?
- F 1.3 Welche Frequenz ist die höchste, die mit dieser Sinustabelle ausgegeben werden kann und mit welcher Schrittweite erreichen Sie diese?
- F 1.4 Wieviele Einträge müsste die Tabelle haben, wenn man die Frequenz auf 1 Hz genau einstellen möchte bzw. die ausgegebene Frequenz um 1 Hz ändern möchte?

(Die Fragen sind handschriftlich in der Versuchsvorbereitung zu beantworten!)

### 1.4 Aussteuerung der D/A-Wandler im Stereo-Codec

Die gewünschte Aussteuerung der beiden Sigma-Delta-D/A-Wandler (16-Bit äquivalent) im Stereo-Codec erfolgt durch Multiplikation der im Gleitkommaformat vorliegenden Sinuswerte mit einem **Gain-Faktor**, der passend zur Ausgabe an die D/A-Wandler im K2 Festkommaformat gewählt werden muss.

Nach der passenden Gleitkomma-Multiplikation muss eine Typkonvertierung ins K2 Festkommaformat erfolgen (siehe Infoblatt ‚Code Composer Studio...‘, Kapitel ‚Datentypen – Typkonvertierungen‘).

In diesem ersten Beispielpogramm wird jedes „angepasste“ Sample über den McBSP1 des DSP gleichzeitig zweimal an den Codec übertragen (also zwei 16-Bit-Samples mit Hilfe eines seriellen 32-Bit-Zugriffs). Beide D/A-Wandler im Codec (linker und rechter Kanal) erhalten hier also jeweils das gleiche Sample, so dass im späteren Betrieb ein Monosignal ausgegeben wird. Genauer betrachtet werden an den D/A-Wandler für den linken Kanal im Stereo-Codec die oberen (höherwertigen) 16 Bit und an den D/A-Wandler für den rechten Kanal die unteren (niederwertigen) 16 Bit des 32-Bit-Datenwortes übergeben (siehe Infoblatt ‚DSP Starter Kit DSK6713‘, Kapitel ‚Schnittstelle des DSK zur analogen Welt‘ bzw. TI PDF-Dokument *tlv320aic23b.pdf*, TLV320AIC23B Stereo Audio CODEC Data Manual‘).

Für Vollaussteuerung (0 dBFS dBFullScale) der D/A-Wandler müssen die Gleitkommawerte der Sinustabelle (Wertebereich -1,0 ... +1,0) also mit  $0x7FFF = 2^{15} - 1 = 32767$  multipliziert werden.

Soll die Aussteuerung z.B. um 10 dB geringer sein, dann muss eben nur mit 10362 multipliziert werden. ( $-10 \text{ dBFS} = 20 \cdot \log(10362/32767)$ )

Einige Beispiele:

Multiplikation mit	entspricht einer Aussteuerung von
32767	0 dBFS
23197	-3 dBFS
16422	-6 dBFS
8231	-12 dBFS
4125	-18 dBFS

**Tabelle 1.1:** Einige Multiplikatoren zur Aussteuerungseinstellung

**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker

Das heißt aber gleichzeitig, dass der durch den jeweiligen D/A-Wandler vorgegebene maximale Dynamikbereich nicht mehr ausgenutzt wird. Das Signal-Rauschverhältnis (S/N) geht entsprechend zurück.

Wenn man also eine hohe Audioqualität erreichen will, dann sollte die Lautstärkeanpassung deshalb möglichst auf der analogen Seite (also nach dem D/A-Wandler) erfolgen. Der Codec bietet dafür beim Kopfhörerausgang eine analoge Gain-Einstellung in 1 dB Schritten mit einer Dämpfung von bis zu 73 dB an (siehe Codec-Datenblatt: *tlv320aic23b.pdf*, Kapitel 3.1.3, Headphone Volume Control Registers).

Im ersten Beispielprogramm verzichten wir allerdings auf die analoge Gain-Einstellung per Control-Register. Stattdessen sollten Sie einige Gain-Faktoren testen (z.B.: -12 dBFS und -36 dBFS).

## 1.5 Ausgabe der Datenwörter

Die Ausgabe der beiden 16 Bit Datenwörter an den Codec erfolgt, wie bereits erwähnt, seriell über den McBSP1 des DSPs. Die Funktion `mcbbsp1_write()` im ersten Beispielprogramm „pollt“ dazu das McBSP1-Serial-Port-Control-Register (SPCR) so lange, bis ein Schreiben auf das DX-Register (Data Transmit Register, DXR) möglich ist, also das zuvor geschriebene Datenwort zum Senden über den seriellen Bus in das XS-Register (Transmit Shift Register, XSR) übernommen wurde. Da das DXR ein 32 Bit Register ist, erwartet es auch ein 32 Bit Datenwort, das in unserem Fall aus dem linken und rechten Sample besteht.

C-Beispielcode für die Gleitkomma-Multiplikation eines Sinuswertes aus der Tabelle mit einem Gain-Faktor, beispielsweise mit 32767.0, anschließender Typkonvertierung, Maskierung und Ausgabe:


```
short_wert = (short)(float_wert * float_gain);  
mcbbsp1_write(((int)short_wert)<<16 | ((int)short_wert & 0x0000FFFF));
```

## 2. Durchführung

### 2.1 Projekt 1 kopieren und entpacken

Kopieren Sie sich den Projektordner **Project1.zip** von der Labor-Internetseite auf die lokale Festplatte in das Verzeichnis **c:\userdata\Vhr\_Arbeitsverzeichnis\** bzw. in **Eigene Dateien** (siehe Einführungsdokument „Organisatorisches“). Die Labor-Internetseite erreichen Sie auf den Labor-PCs im Raum T1.4.01 am schnellsten über das Startmenü:

Start -> Labore -> Signale und Systeme -> Labor Signale und Systeme

Die auf Eclipse basierende Entwicklungsumgebung Code Composer Studio v5  von Texas Instruments greift auf Projekte in einem bestimmten, benutzerbezogenen Arbeitsverzeichnis zu. Dieser sog. **CCSv5-Workspace** für die Projekte befindet sich im Verzeichnis

**c:\userdata\Vhr\_Arbeitsverzeichnis\eclipse\workspace\_v5\_1**

Dieses Verzeichnis legt die Entwicklungsumgebung beim aller ersten Start durch Sie in Ihrem persönlichen Arbeitsverzeichnis an. (Entpacken Sie daher den Projektordner **Project1** aus der Datei **Project1.zip** erst nach dem erstmaligen Start der Entwicklungsumgebung, der in Kapitel 2.3 erfolgt, in das dann bestehende Unterverzeichnis **workspace\_v5\_1**.)

**SS 2013**


Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker

## 2.2 Inbetriebnahme des Demoboards

Versorgen Sie das Starterkit (DSK6713) mit Spannung (Netzteil) und warten Sie den jetzt ablaufenden **PowerOnSelfTest** (POST) ab. Die LED1 leuchtet dabei für einige Sekunden. Am Ende des Tests blinken die USER-LEDs des Boards einige Male und leuchten nach fehlerfrei abgelaufenem Test dauerhaft.

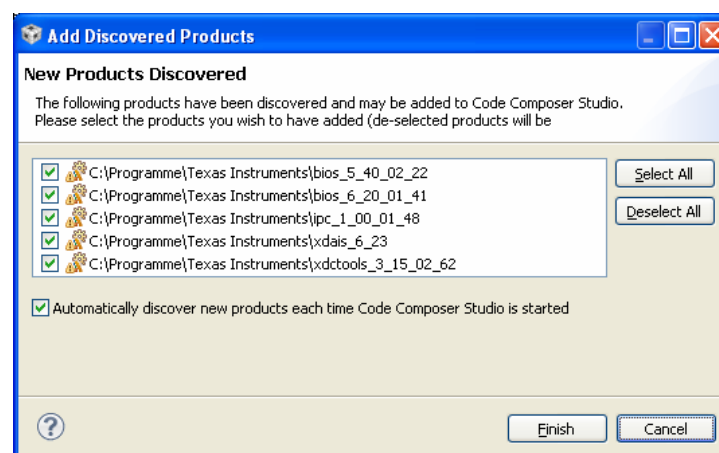
## 2.3 Entwicklungsumgebung starten und Projekt 1 öffnen

Starten Sie erst nach der Inbetriebnahme des Demoboards (vgl. Kapitel 2.2 ) die Entwicklungsumgebung Code Composer Studio v5  über das Startmenü:

Start -> Labore -> Signale und Systeme -> Code Composer Studio



Am Ende des Startvorgangs öffnet sich eventuell ein Fenster, über das der Entwicklungsumgebung weitere Komponenten hinzugefügt werden könnten. Quittieren Sie diese Abfrage mit **Cancel**.



Anmerkung:

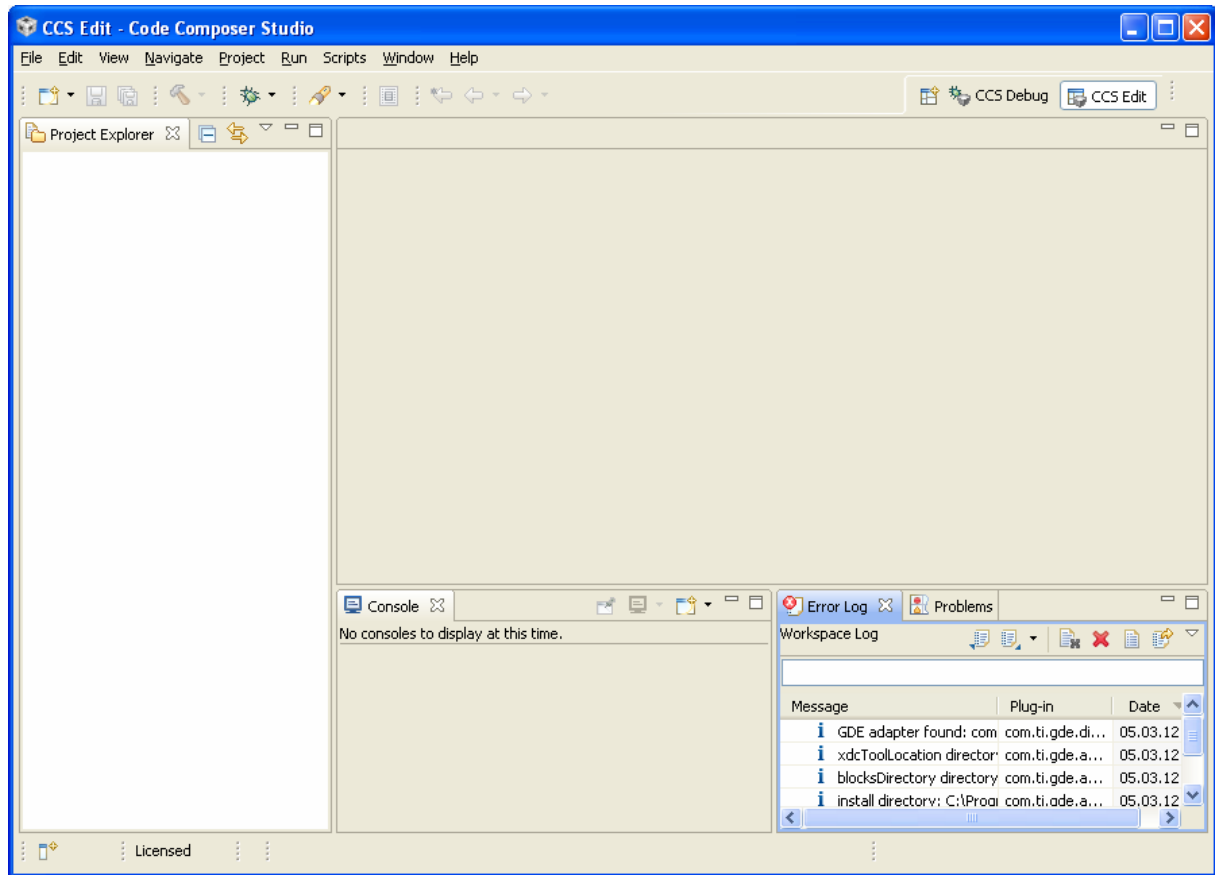
Beim ersten Start von Code Composer Studio v5 erscheint innerhalb der Entwicklungsumgebung zunächst das Fenster TI Resource Explorer, das nicht benötigt wird und deshalb geschlossen werden kann.

**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker


Danach steht die Entwicklungsumgebung für die Arbeit mit dem DSK6713 zur Verfügung.




**Abbildung 2.1:** Entwicklungsumgebung CCS v5 auf Basis von Eclipse

**Entpacken** Sie jetzt den Projektordner *Project1* aus der Datei *Project1.zip* in das Unterverzeichnis **workspace\_v5\_1**.

*c:\userdata\lhr\_Arbeitsverzeichnis\eclipse\workspace\_v5\_1\Project1*

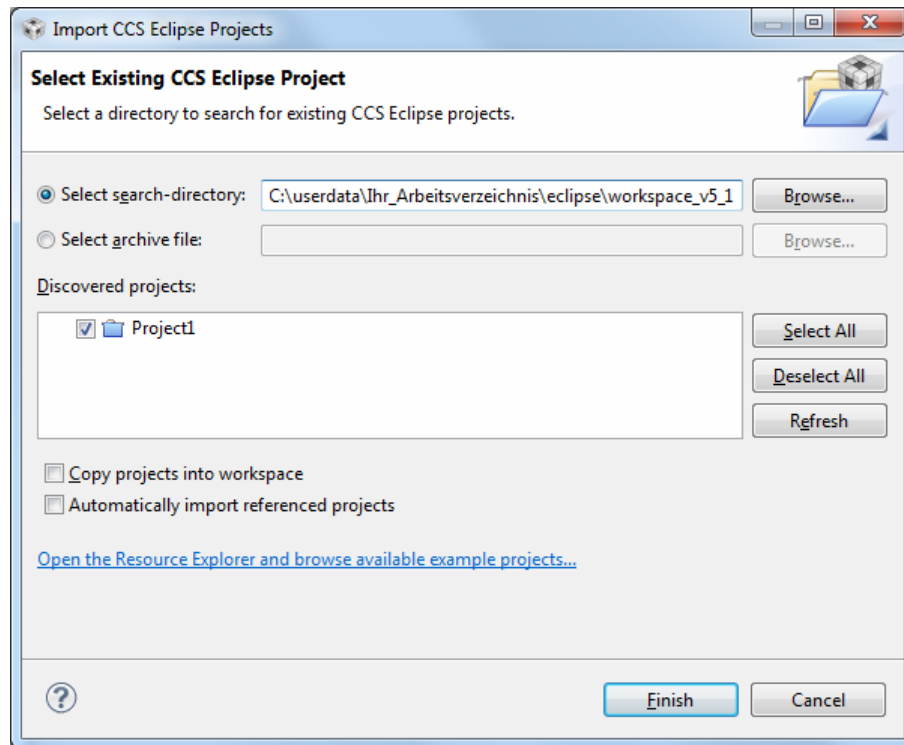
Wählen Sie in der dargestellten Ansicht CCS Edit von CCS v5 im Menü **Project** den Eintrag  **Import Existing CCS/CCE Eclipse Project**.

Im sich öffnenden Fenster  **Import CCS Eclipse Projects** wählen Sie über **Browse...** den Pfad zu Ihrem **CCSv5-Workspace** aus, wodurch das zuvor dort hin kopierte Project1 gefunden und im Fensterbereich **Discovered projects** gelistet wird. Klicken Sie anschließend auf **Finish**.

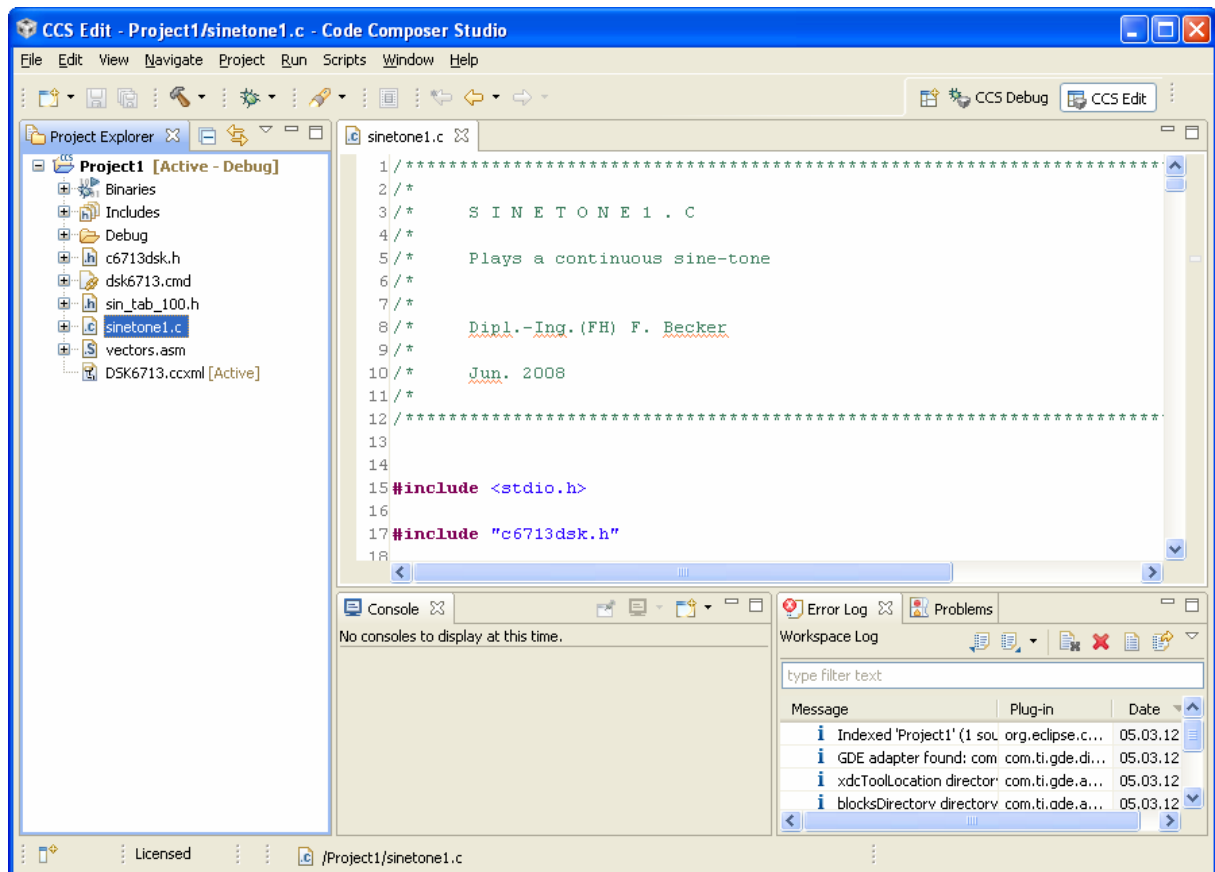
**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker



**Abbildung 2.2:** Import CCS Eclipse Projects



**Abbildung 2.3:** Entwicklungsumgebung CCS v5 mit geöffneter C-Quellcodedatei

**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker

Verschaffen Sie sich einen Überblick über die in dieses Projekt eingebundenen Dateien. (Erläuterungen siehe Infoblatt ,Code Composer Studio...')




Sehen Sie sich die Dateien **c6713dsk.h**, **sin\_tab\_100.h**, **sinetone1.c**, **vectors.asm** und **dsk6713.cmd** genauer an und versuchen Sie die Funktionsweise nachzuvollziehen !!!

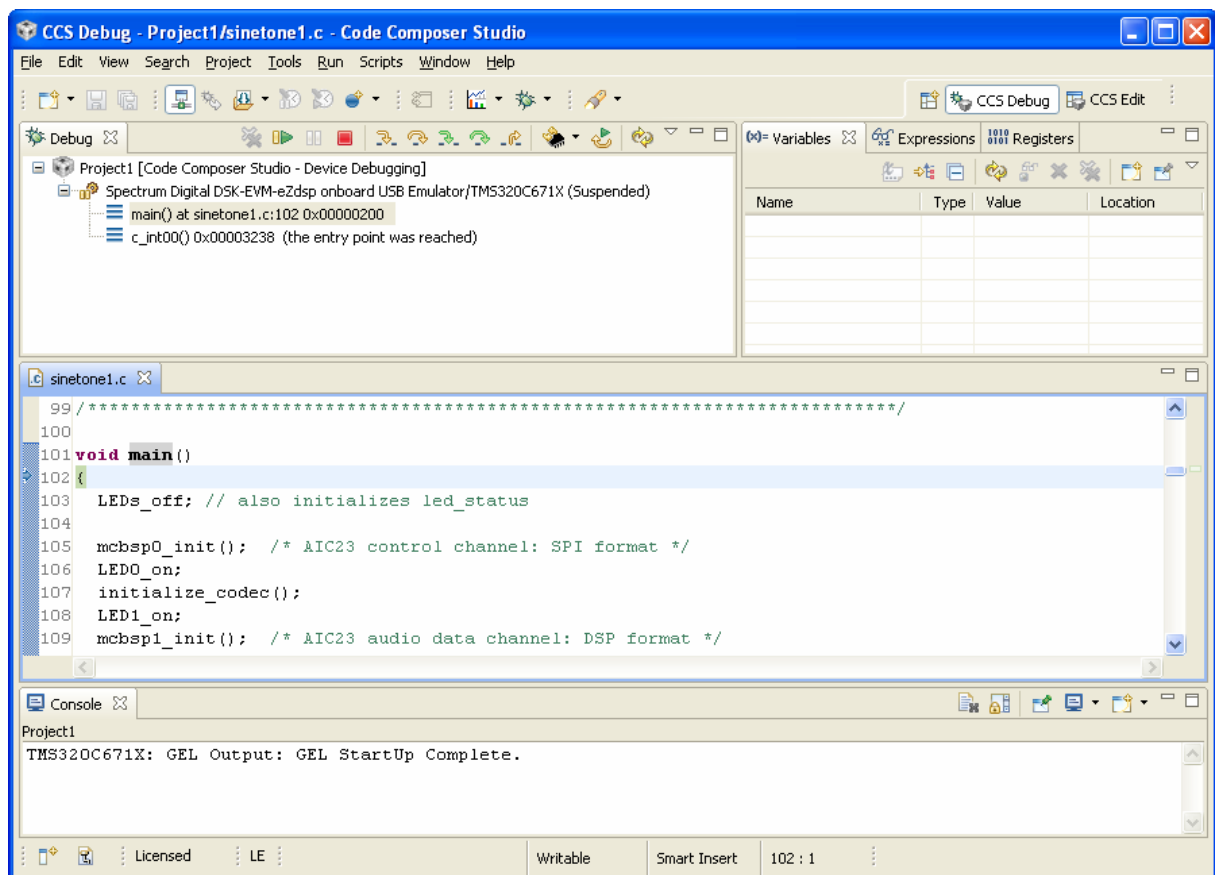
## 2.4 Vorbereiten des Programmstarts auf dem Demoboard / Ansicht CCS Debug

Da das Starterkit (DSK6713) bereits in Kapitel 2.2 mit Spannung versorgt wurde, kann jetzt direkt der Debugger von CCS gestartet werden. Das Starten des Debuggers bewirkt einen Wechsel zur Ansicht CCS Debug von Code Composer Studio v5 mit anderen Fenstern und Menüs und gleichzeitig das Vorbereiten des Demoboards für den Programmstart. Dabei wird u.a. die Programmierdatei *Project1.out* auf das DSK6713 heruntergeladen.

Der Start des Debuggers und damit das Laden der Programmierdatei auf die Zielplattform kann auf vier verschiedene Möglichkeiten geschehen:

Bei ausgewähltem/markiertem  Projekt im  Project Explorer

- entweder im Menü **Run** den Eintrag  Debug wählen
- oder in der Symbolleiste den Käfer  anklicken
- oder den Hot-Key F11 drücken
- oder mit der rechten Maustaste das zum ausgewählten Projekt gehörige Kontextmenü öffnen und darin **Debug As**  Code Composer Debug Session wählen.



**Abbildung 2.4:** Ansicht CCS Debug der Entwicklungsumgebung



**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI


Dipl.-Ing.(FH) Felix Becker

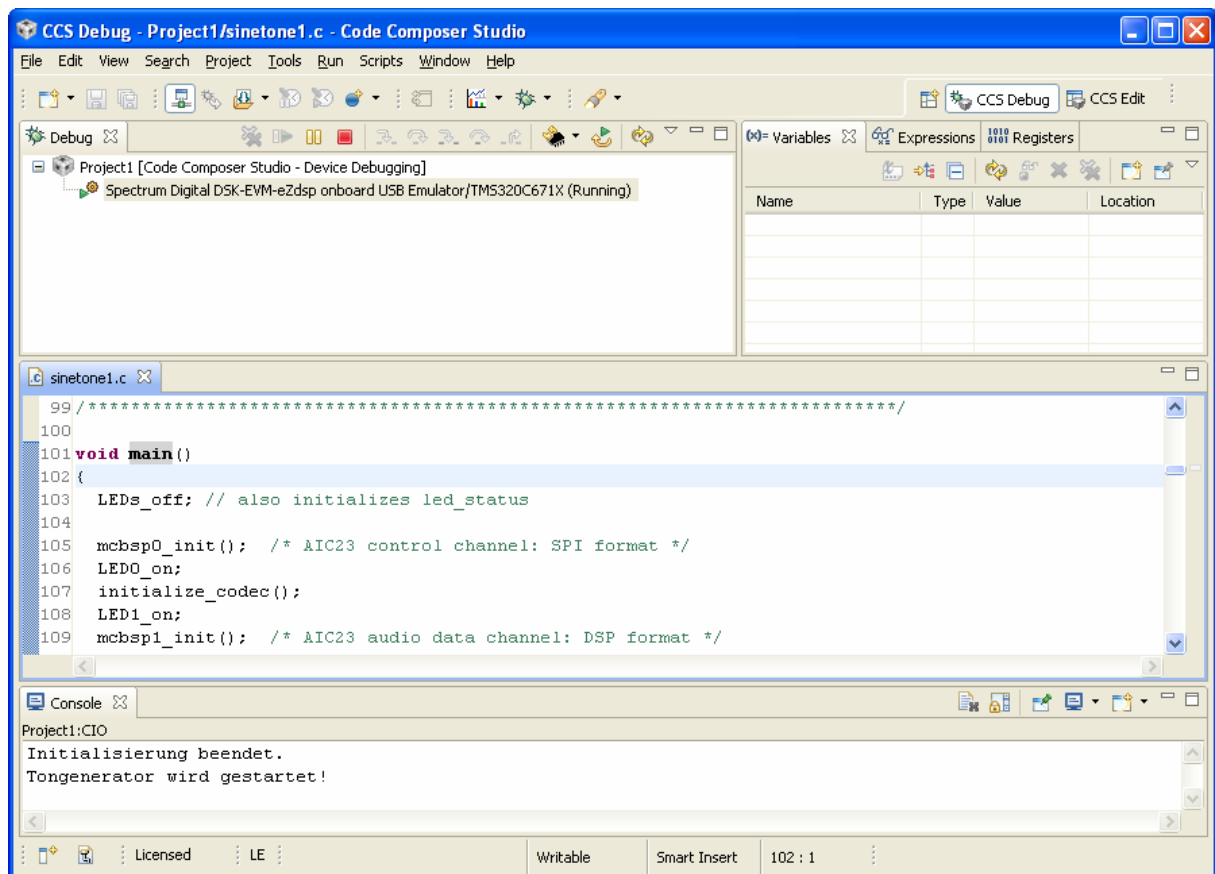
Die Interrupt-Vektortabelle und der Code werden in den internen Speicher des DSPs geladen, einige Bereiche werden aber auch, wie in der Datei *dsk6713.cmd* angegeben, in das externe SDRAM kopiert. Danach leuchtet auf dem DSK6713 kurz die LED0 und anschließend nur noch die LED1. Im Quellcodefenster wird der Code ab der C-Einsprungsadresse angezeigt (Label `c_int00`).

## 2.5 Programm auf der Zielplattform starten

In der Ansicht CCS Debug der Entwicklungsumgebung kann direkt das Programm auf dem DSK6713 gestartet werden.

Der Programmstart kann auf drei verschiedene Möglichkeiten erfolgen:

- entweder das Symbol  anklicken
- oder über das Menü **R**un den Eintrag **R**esume wählen
- oder einfach den Hot-Key F8 drücken.



**Abbildung 2.5:** Ansicht CCS Debug nach Programmstart

Beim Programmstart leuchten die USER-LEDs des Boards gemäß des Programmcodes in `main()` kurz auf und im Konsolenfenster wird der Text des `puts()`-Befehls ausgegeben.

Danach können die LEDs mit den vier USER-Switches auf dem Board aktiviert werden. Dies signalisiert, neben dem vermutlich hörbaren Sinuston, dass das Programm ordnungsgemäß abgearbeitet wird (vgl. Programmcodes in `play_tone()`).

Hören Sie sich den Sinuston über den Kopfhörer an.



**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

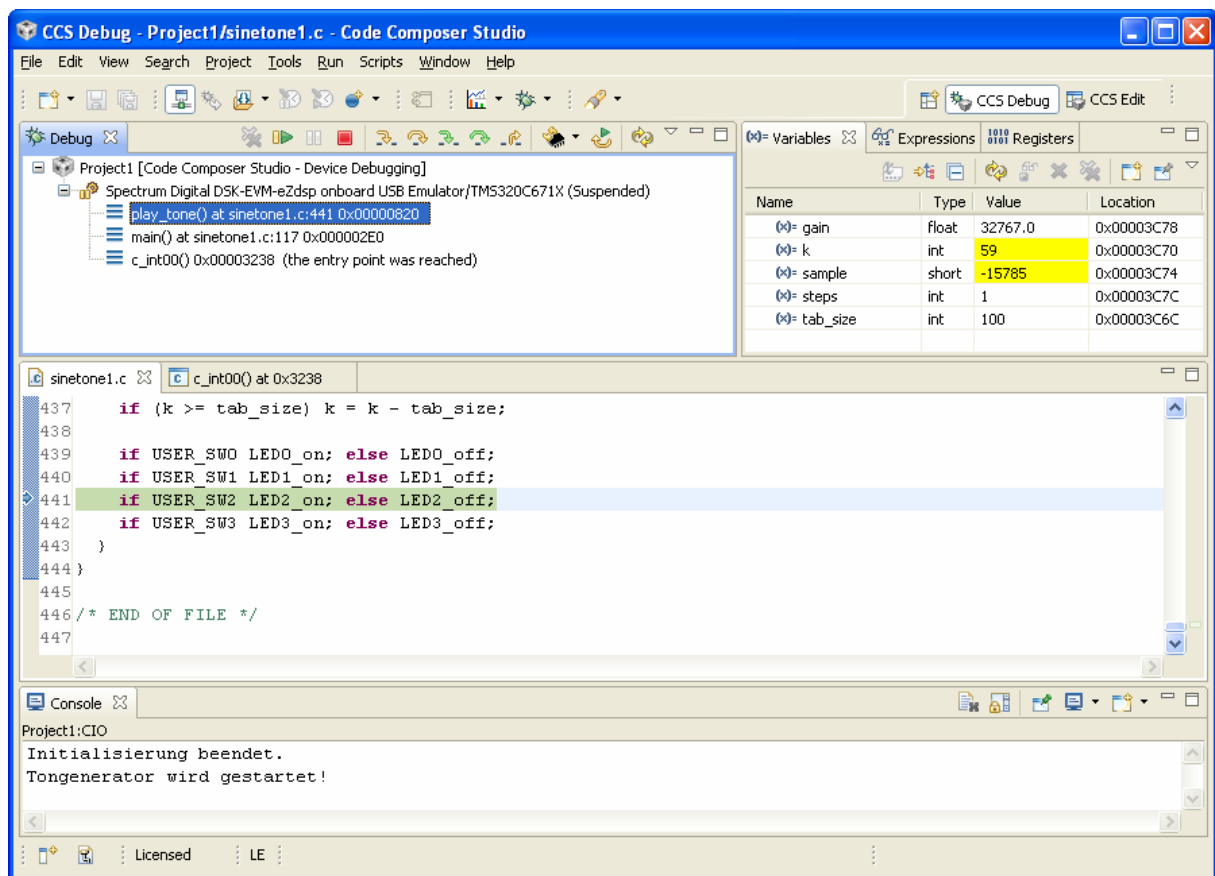
Dipl.-Ing.(FH) Felix Becker

(Balance- und Pegelsteller für LINE IN 2/3, MAIN MIX und PHONES sowie Schalter TAPE TO PHONES am Mischpult beachten! Siehe dazu auch das Infoblatt ‚Erläuterungen zur Audioverkabelung‘)

## 2.6 Programmabarbeitung auf der Zielplattform unterbrechen

Klicken Sie auf das Symbol  oder wählen Sie im Menü **Run** den Eintrag  Suspend um die Programmabarbeitung auf dem DSK6713 zu unterbrechen.

Ja nach Unterbrechungszeitpunkt in Bezug zur Programmabarbeitung erhalten Sie eine Ansicht, die einer der beiden folgenden Abbildungen ähnlich ist. Ein Pfeil am linken Rand des Quellcodefensters zeigt auf die beim Stoppen gerade aktuelle Programmstelle und die Codezeile ist grün hinterlegt. Die zweite der beiden Ansichten ist übrigens wahrscheinlicher! Warum?

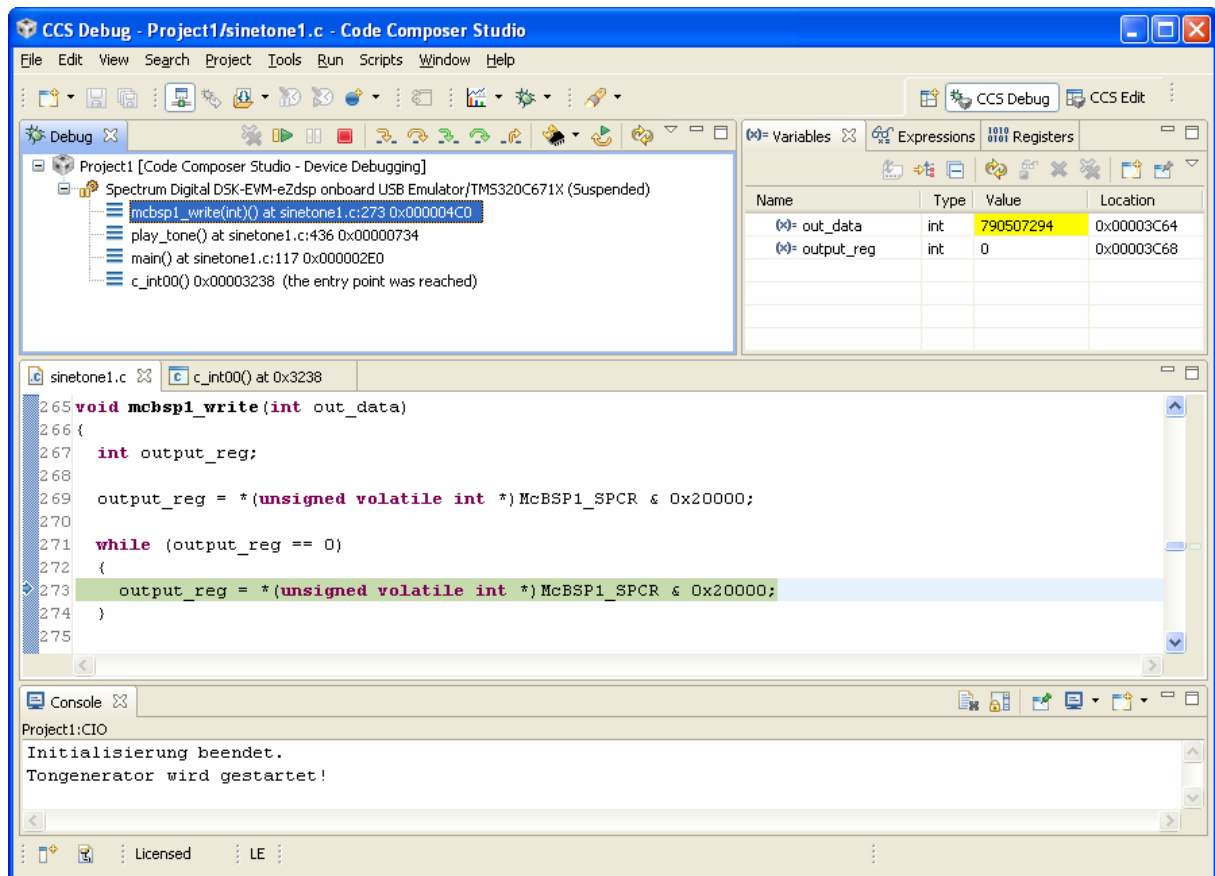


**Abbildung 2.6:** Eine mögliche Ansicht nach Unterbrechung der Programmabarbeitung

**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker



**Abbildung 2.7:** Wahrscheinlichste Ansicht nach Unterbrechung der Programmabarbeitung

## 2.7 Erste Code-Änderung, Build-Prozess und Program-Reload

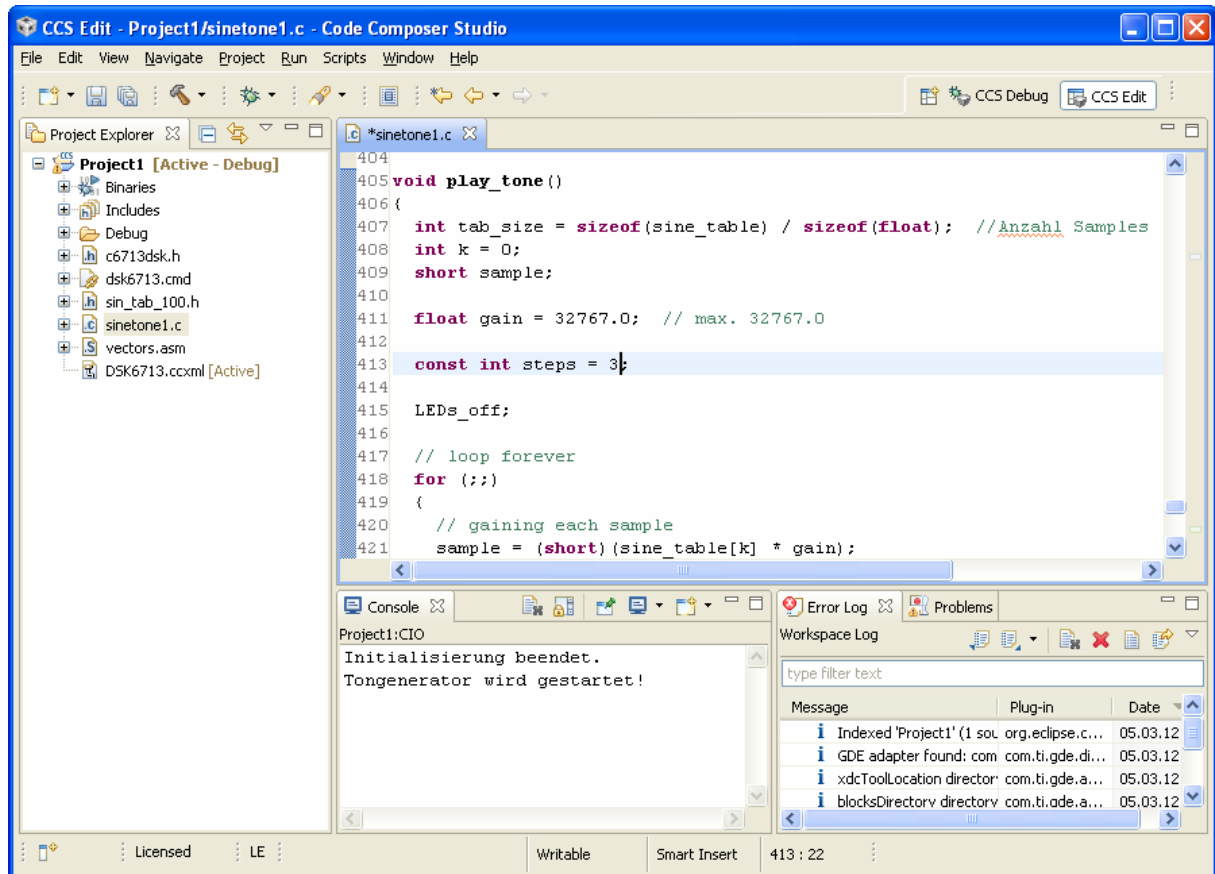
Wechseln Sie mit Hilfe des Buttons im oberen rechten Bereich der Entwicklungsumgebung zurück in die Ansicht CCS Edit. Ändern Sie die Tonhöhe auf 960 Hertz ab. Editieren Sie dazu die Datei *sinetone1.c* entsprechend.

Halten Sie in Ihrem Versuchsprotokoll fest, was Sie wo und wie ändern müssen und beschreiben Sie kurz was die Änderung beim Programmablauf bewirkt.

**SS 2013**




Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker



**Abbildung 2.8:** Ansicht CCS Edit

Projekt compilieren – Programmierdatei *Project1.out* erstellen:

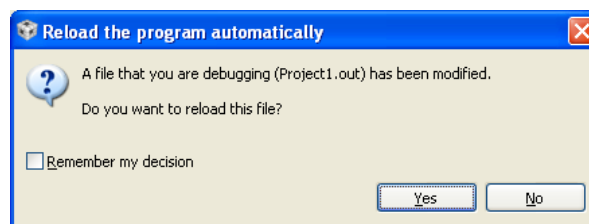
Starten Sie Compiler und Linker indem Sie bei ausgewähltem/markiertem  Projekt im  Project Explorer entweder im Menü **Project** den Eintrag  Build All wählen oder mit der rechten Maustaste das zugehörige Kontextmenü öffnen und darin Build Project wählen.

Nach erfolgreichem Build-Prozess erhalten Sie im Konsolenfenster die abschließende Meldung:

Finished building target: Project1.out

\*\*\*\* Build Finished \*\*\*\*

und folgende Abfrage:

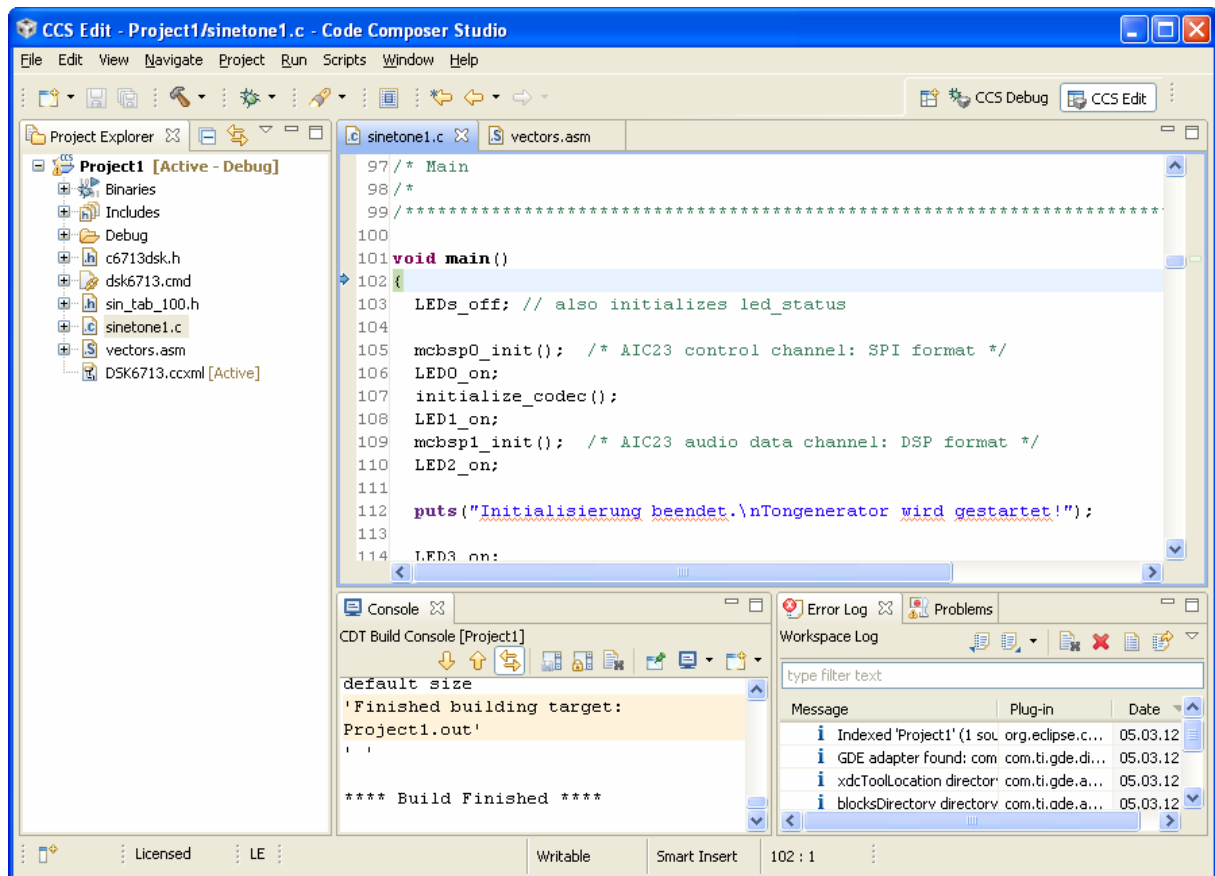


Quittieren Sie die Abfrage mit **Yes**.

**SS 2013**


Fakultät für Technik, Studiengänge EIT/TI

Dipl.-Ing.(FH) Felix Becker



**Abbildung 2.9:** Entwicklungsumgebung CCS v5 nach erfolgreichem Build-Prozess (Textausgabe im Fenster Console)

Nach dem Wechsel in die Ansicht CCS Debug (Button im oberen rechten Bereich der Entwicklungsumgebung) kann dann das neue Programm sofort gestartet werden:

- entweder das Symbol  anklicken
- oder über das Menü **Run** den Eintrag Resume wählen
- oder einfach den Hot-Key F8 drücken.

Überprüfen Sie das Ergebnis wieder mit dem Kopfhörer.

## 2.8 Weitere Code-Änderungen

Verringern Sie anschließend den Pegel des Sinustons über den Gain-Faktor schrittweise um 12 dB, um 24 dB und um 36 dB.

Editieren Sie dazu die Datei *sinetone1.c* entsprechend und überprüfen Sie Ihre Änderung mit dem Kopfhörer.

Übernehmen Sie die Berechnung der benötigten Faktoren in Ihr Versuchsprotokoll.

## 2.9 Fragen

F 2.1 Wozu dient die Datei *dsk6713.cmd* und warum wird die Datei *vectors.asm* benötigt?

F 2.2 Wieviel mal pro Sekunde wird die for-Schleife der Funktion `play_tone()` durchlaufen?

**SS 2013**

Fakultät für Technik, Studiengänge EIT/TI

*Dipl.-Ing.(FH) Felix Becker*

- F 2.3 Hat die eingestellte Tonhöhe darauf einen Einfluss?  
Begründen Sie kurz Ihre Antwort.
- F 2.4 Was beschreibt der Begriff ‚Echtzeit-Verarbeitung‘ bzw. ‚Realtime-Processing-System‘?  
Nennen bzw. erläutern Sie Hauptmerkmale.
- F 2.5 Erläutern Sie den Begriff ‚Polling‘.  
Welche Auswirkungen (besser gesagt Nebenwirkungen) hat Polling-Betrieb?
- F 2.6 An welcher Stelle des Sinusgeneratorprogramms tritt ‚Polling‘ auf (präzise Angabe)?  
(Die Fragen sind handschriftlich in der Versuchsvorbereitung bzw. Ihrem Versuchsprotokoll zu beantworten!)