

# AP16134

## XE167

Playing music using the CAPCOM60 module.



Microcontrollers



Never stop thinking

**Edition 2008-07-16**

**Published by  
Infineon Technologies AG  
81726 München, Germany**

**© Infineon Technologies AG 2008.  
All Rights Reserved.**

#### **LEGAL DISCLAIMER**

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

#### **Information**

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office ([www.infineon.com](http://www.infineon.com)).

#### **Warnings**

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

---

**AP08048**

**Revision History:** 2008-01 V2.0

Previous Version: none

Page	Subjects (major changes since last revision)

**We Listen to Your Comments**

Any information within this document that you feel is wrong, unclear or missing at all?  
Your feedback will help us to continuously improve the quality of this document.  
Please send your proposal (including a reference to this document) to:

[mcdocu.comments@infineon.com](mailto:mcdocu.comments@infineon.com)



**Note:** Table of Contents see [page 10](#) and page 11.

### Introduction:

This “Appnote” is a Hands On Training / Cookery Book / step-by-step book.  
It will help inexperienced users to get familiar with the CAPCOM 6 / CCU6 module.  
This step-by-step book is a follow-up to AP16133.

The purpose of this document is to gain know-how of the possibilities offered by the CAPCOM 6 module for PWM generation.

### **Note:**

The style used in this document focuses on working through this material as fast and easily as possible. Which means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

Have fun and enjoy the CAPCOM 6 module!

### **Note:**

Additionally, there is a step-by-step book (AP16109) focusing on BLDC-Motors available, which can be used for all 8/16 and 32 bit microcontrollers equipped with the CAPCOM 6 module.  
To get the most out of the CAPCOM 6 module this additional Cookery Book is the icing on the cake of all available functionalities (modes) offered by this module (e.g. Multi-Channel Mode, Hall Sensor Mode).

### **Note:**

In case you want to start with the CCU6 from scratch (generating Asymmetrical/Edge-Aligned PWM signals or Symmetrical/Center-Aligned PWM signals) we suggest taking a look at AP08068.

### **Note:**

At the time this document was written there was no Keil simulation support for the XE167 microcontroller. If you want to learn how to setup the Keil software simulated logic analyzer to view the PWM signals on the Keil simulator we also suggest taking a look at AP08068.





**Asymmetrical / Edge-Aligned PWM generation:**  
**Single Shot Mode: Timer12 (note length),**  
**Modulation: Timer13 (note frequency),**

**Playing music**





**Note:**


Port\_10 Low pins used by our PWM module CCU60:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P10.0	CCU60_CC60	100 (note length, Timer_12)
P10.1	CCU60_CC61	100 (note length, Timer_12)
P10.2	CCU60_CC62	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P10.7	CCU60_COUT63	50 (note frequency, Timer_13)

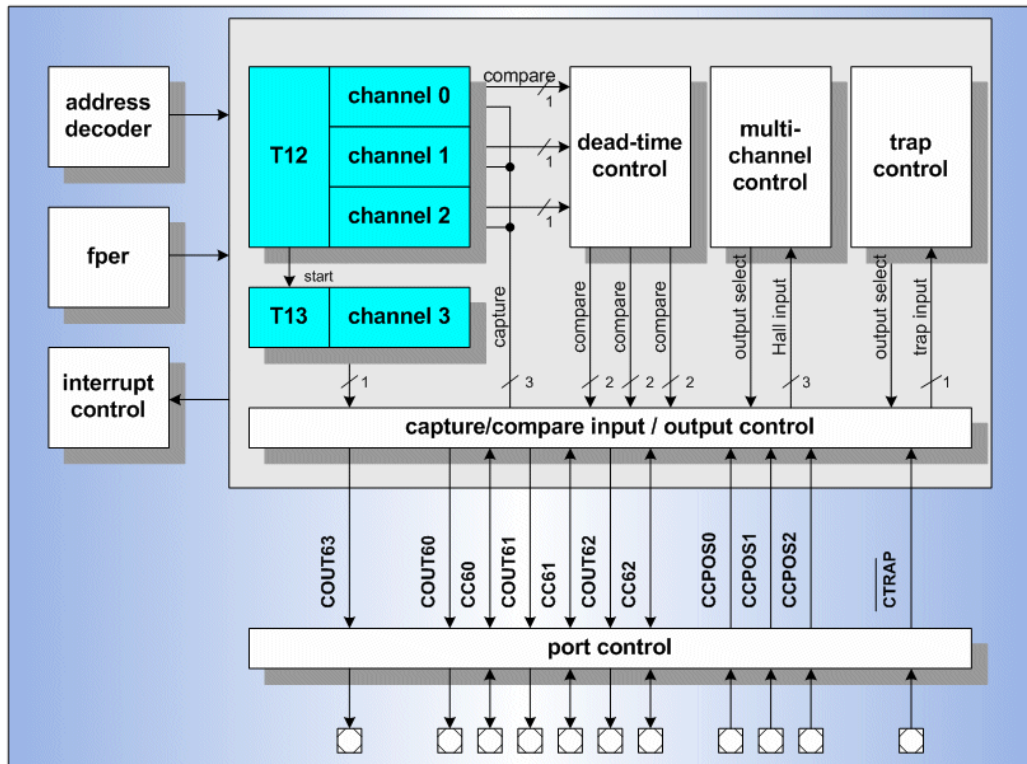
Port\_10 Low pins used as GPIO:

Port Lines	Function	Comment
P10.3	Show start of next note	Toggled via Software
P10.4	Show start of next note	Toggled via Software
P10.5	„use: program running signal“	Toggled via CAPCOM2_Timer_7 ISR
P10.6	„use: program running signal“	Toggled via CAPCOM2_Timer_7 ISR

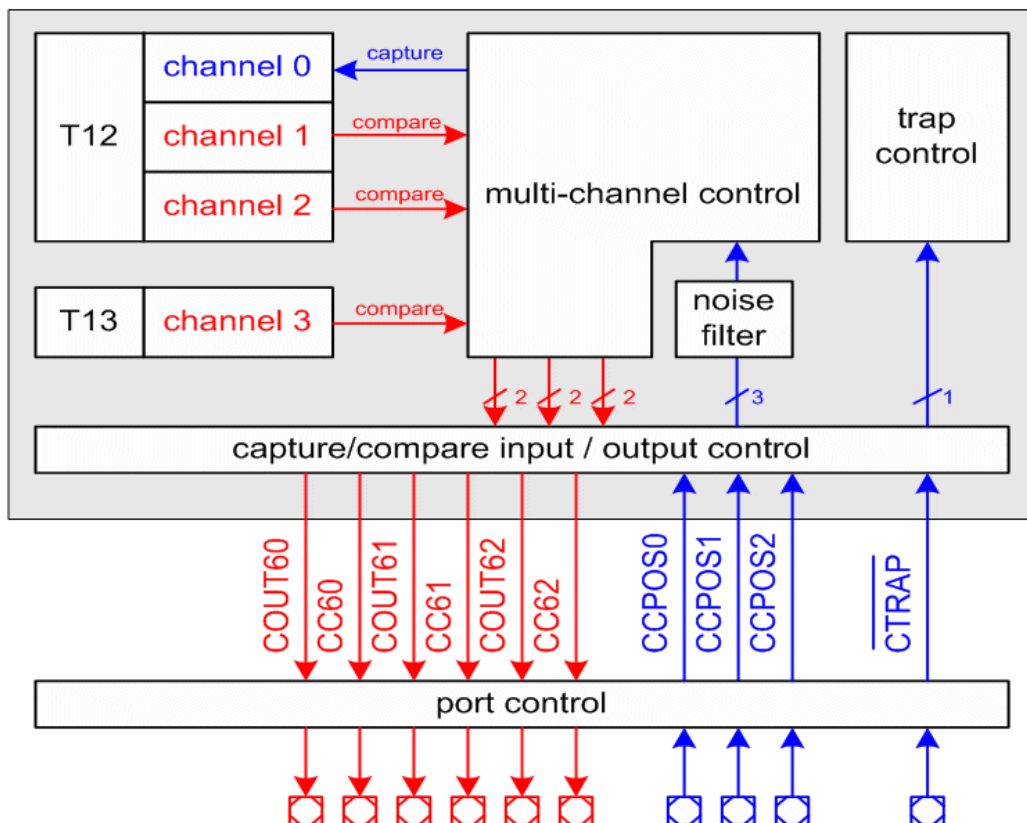
Port 10 Low pins:

Pin		CCU60-Channel	Modulated by	Purpose	
P10.0	CC60	CCU60 Channel 0	Modulated by T12	show note length duty cycle = 100 % only for measurement	
P10.1	CC61	CCU60 Channel 1	Modulated by T12	show note length duty-cycle = 100 % only for measurement	
P10.2	CC62	CCU60 Channel 2	Modulated by T12 + T13	<u>Music Output:</u> note length modulated by note frequency	
P10.3		---	Software	start of next note	
P10.4		---	Software	start of next note	
P10.5		---	Software	running signal	
P10.6		---	Software	running signal	
P10.7	CC63	CCU60 Channel 3	Modulated by T13	note frequency only for measurement	

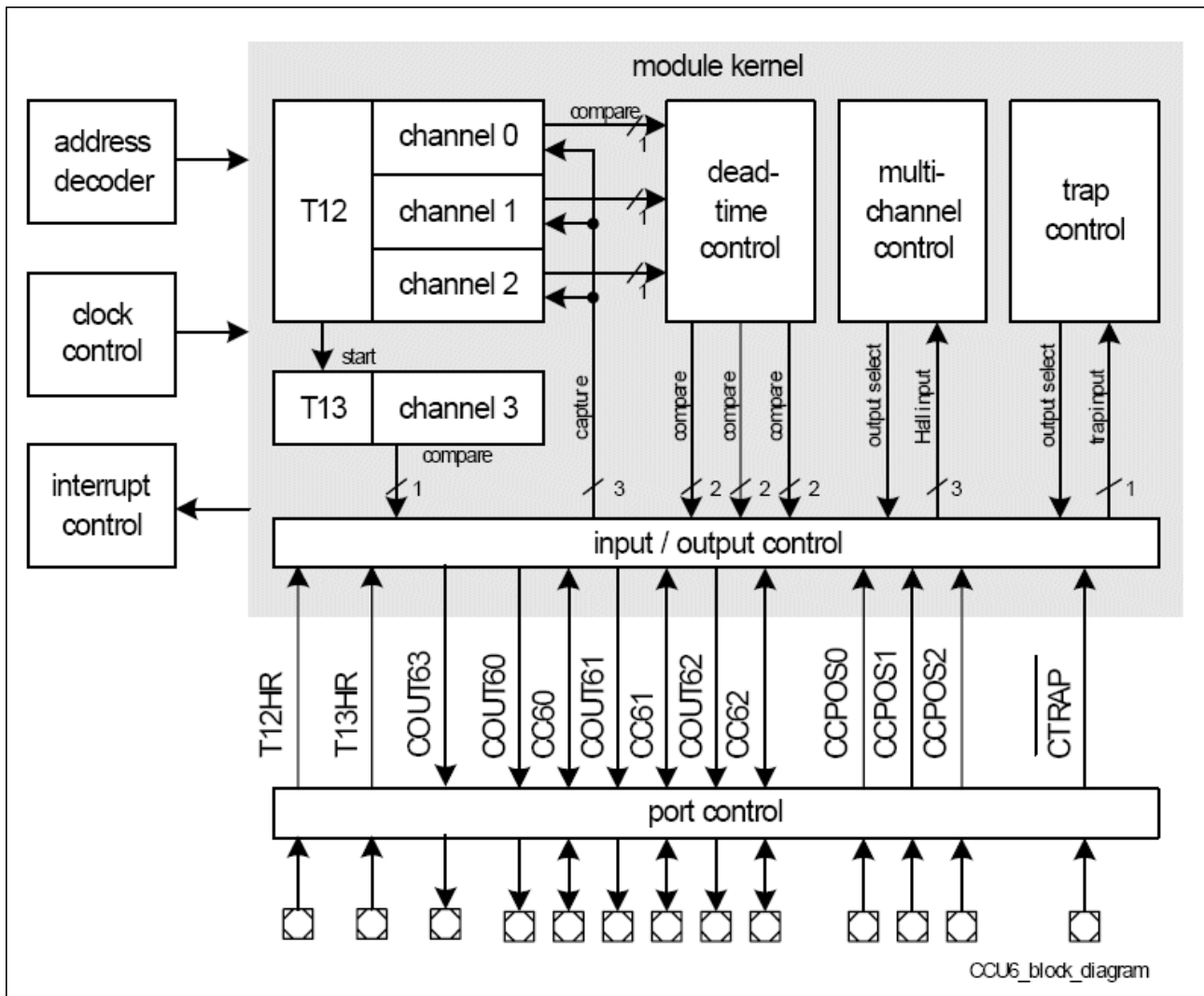
CCU6 Block Diagram – general use (Source: Product Marketing)



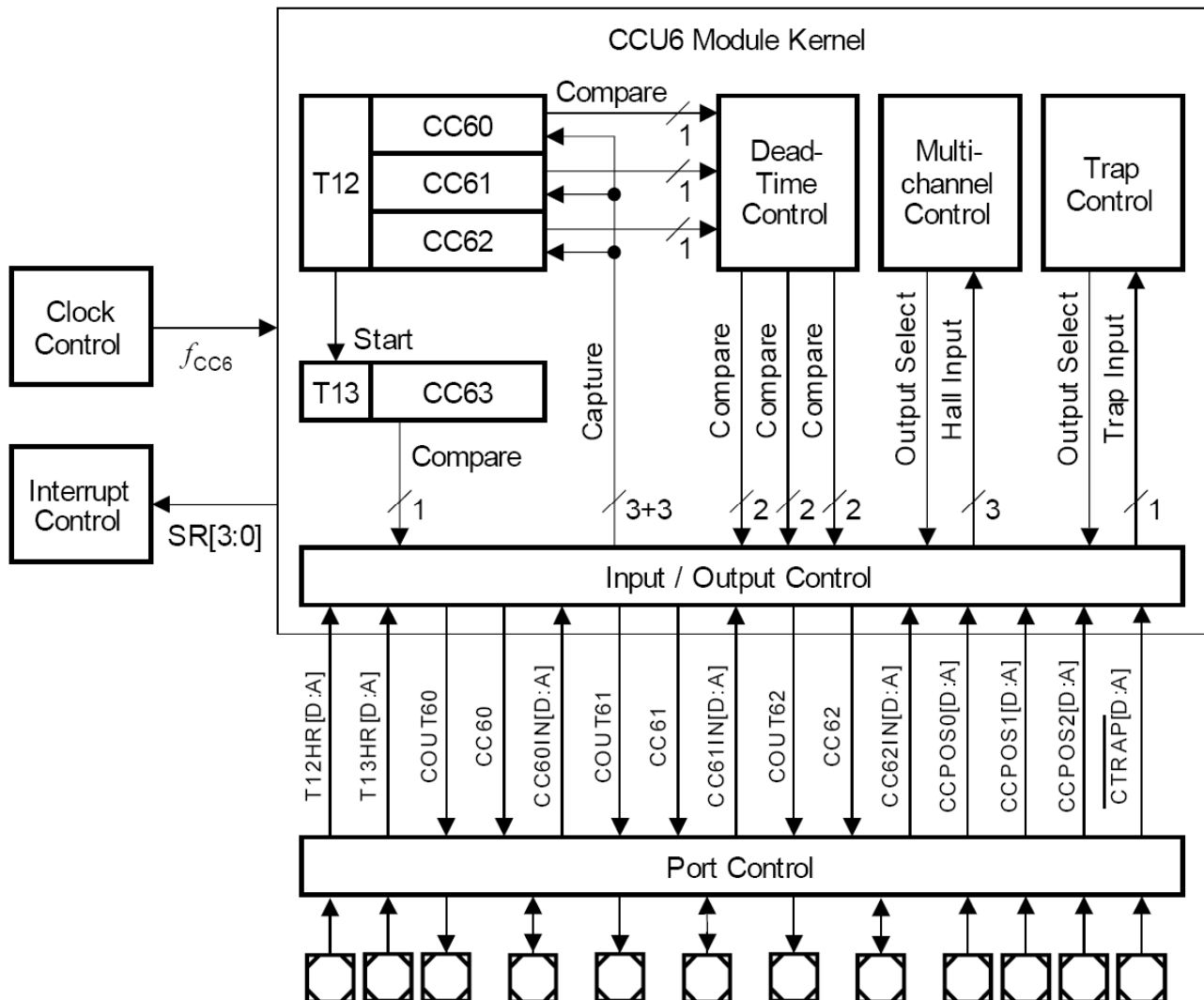
CCU6 Block Diagram – BLDC use (Source: Product Marketing)



CCU6 Block Diagram (Source: User's Manual)



CCU6 Block Diagram (Source: XE166 User's Manual)

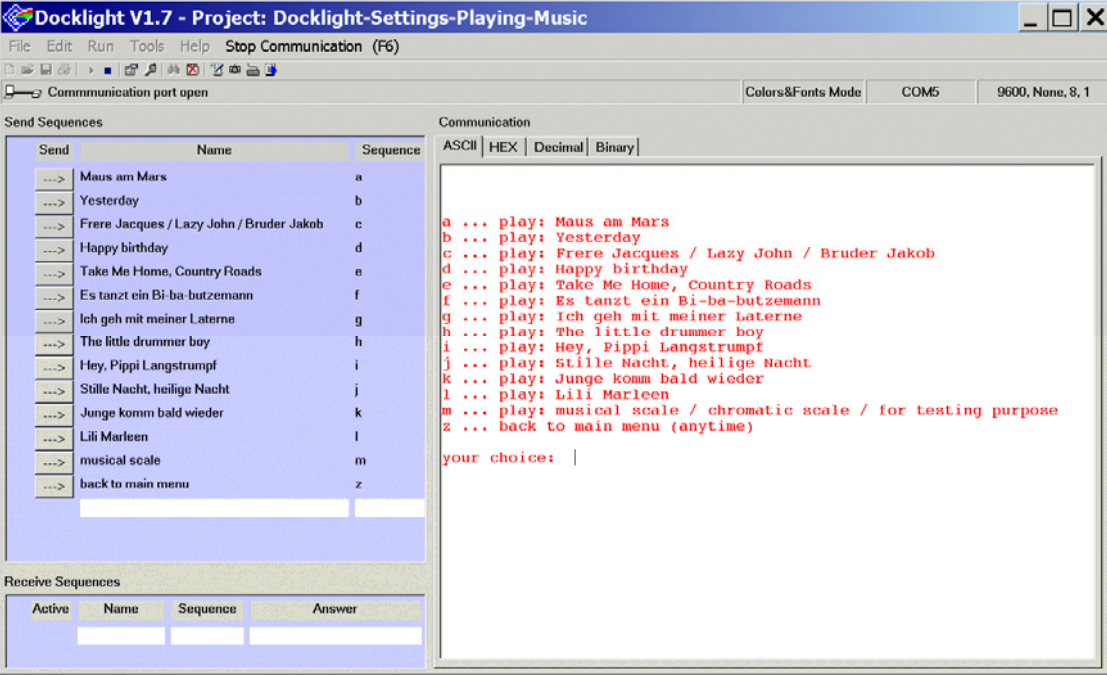


**Note:**

Just by comparing the different sources of the CAPCOM 6 Module Block Diagrams [Capture/Compare Unit 6 (CCU6)], you should be able to get a picture of the module and to answer some of your initial questions.

## “Cookery-book“

For your first programming example for the CCU6:

<p><b>Your program:</b></p>	
<p>Chapter/ Step</p>	<p>*** Recipes ***</p>
<p>1.)</p>	<p><u><a href="#">Asymmetrical / Edge-Aligned PWM generation</a></u>  <u><a href="#">Single Shot Mode (Timer12), Modulation (Timer13)</a></u>  <u><a href="#">Playing music</a></u></p>



## Appendix:

Chapter/ Step	*** Recipes ***
2.)	<a href="#">Appendix: about music (note length, note frequency)</a>
3.)	<a href="#">Appendix: CCU6 use to create note length and note frequency</a>
4.)	<a href="#">Appendix: songs used</a>

## Feedback:

5.)	<a href="#">Thanks To</a>
6.)	<a href="#">Feedback</a>

Do the XE167 Cookery Book:

**Note:**

It is necessary to follow all instructions in the XE167 Cookery Book (AP16133) step by step, as this is the basis for all instructions which will follow later.

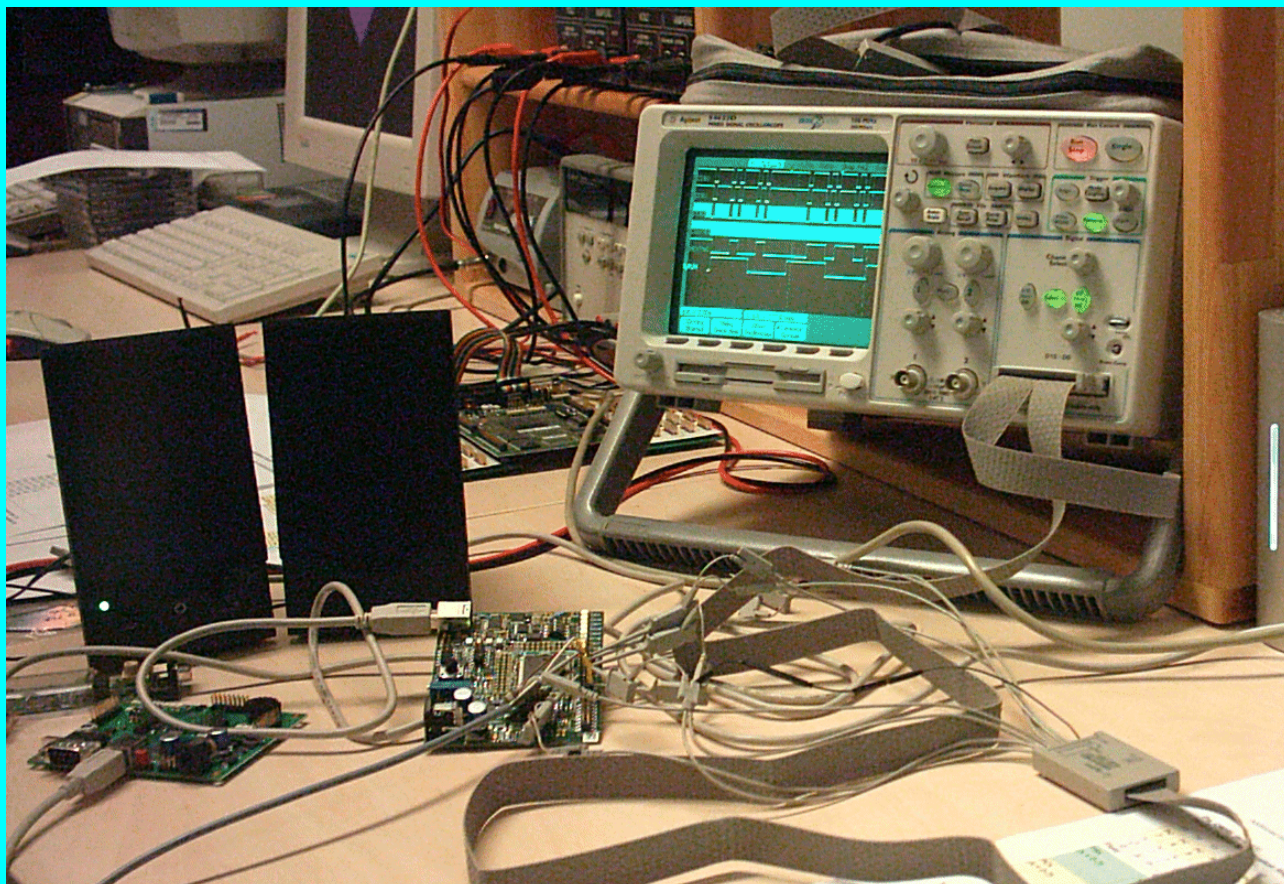


**Note:**

In the following steps of this document we will expand the “Hello World Application” (Application Note AP16133) with the requirements for PWM generation (playing music).



## 1.) Let's Get Started:



## Configuring and Reconfiguring the DAvE Project Settings:



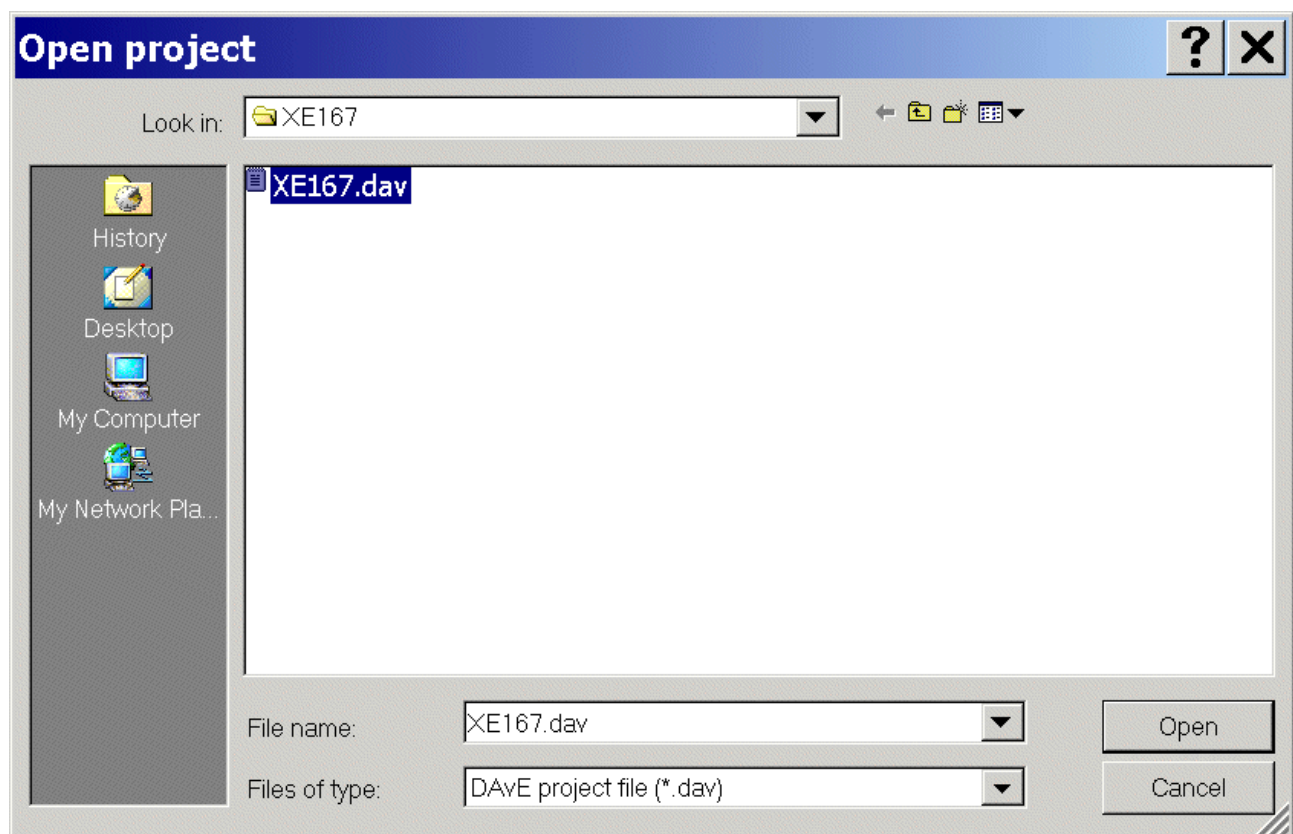
Start the program generator DAvE and open your [XE167.dav](#) DAvE project:

File

Open

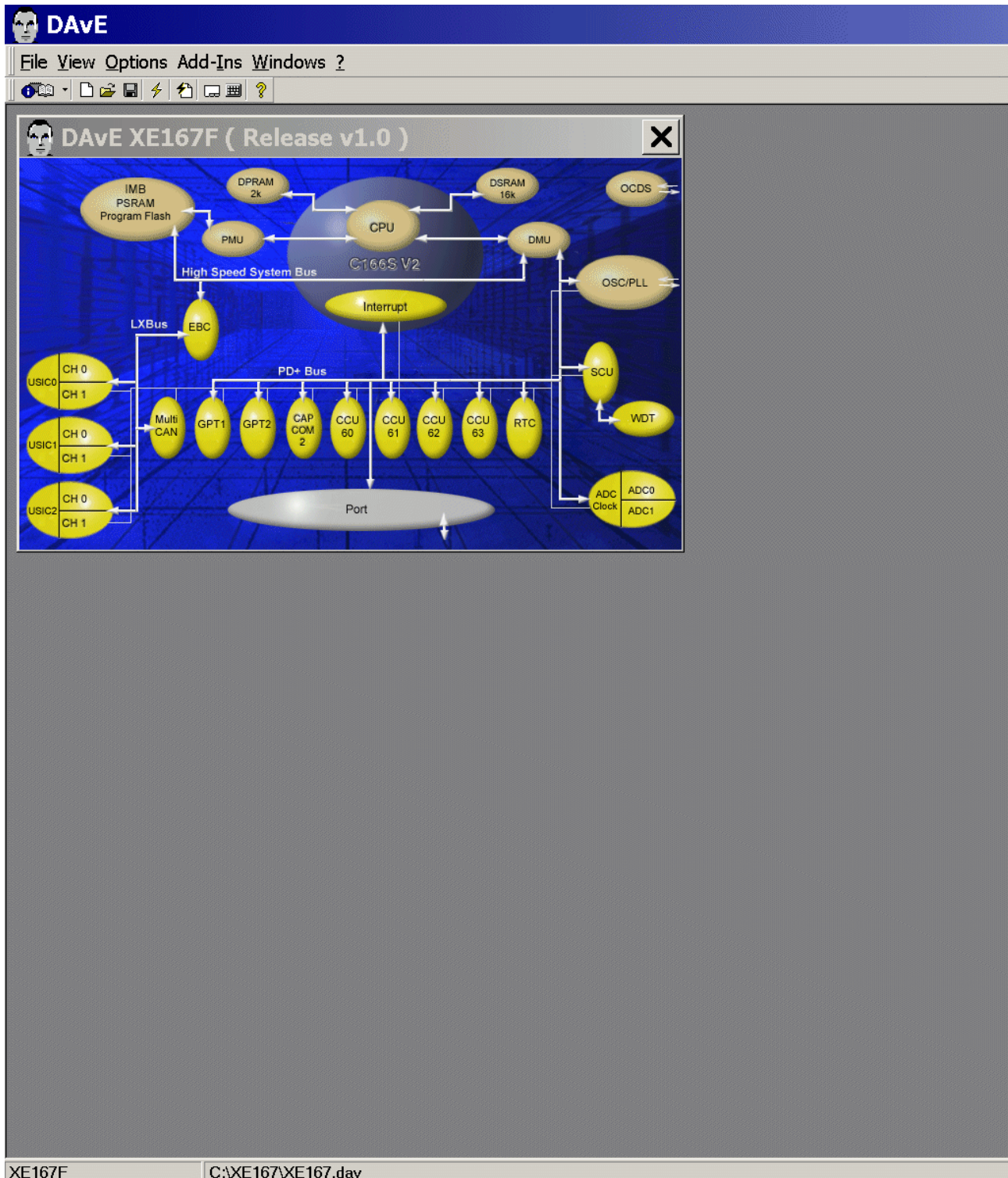
Location: C:\XE167

Filename: [XE167.dav](#)



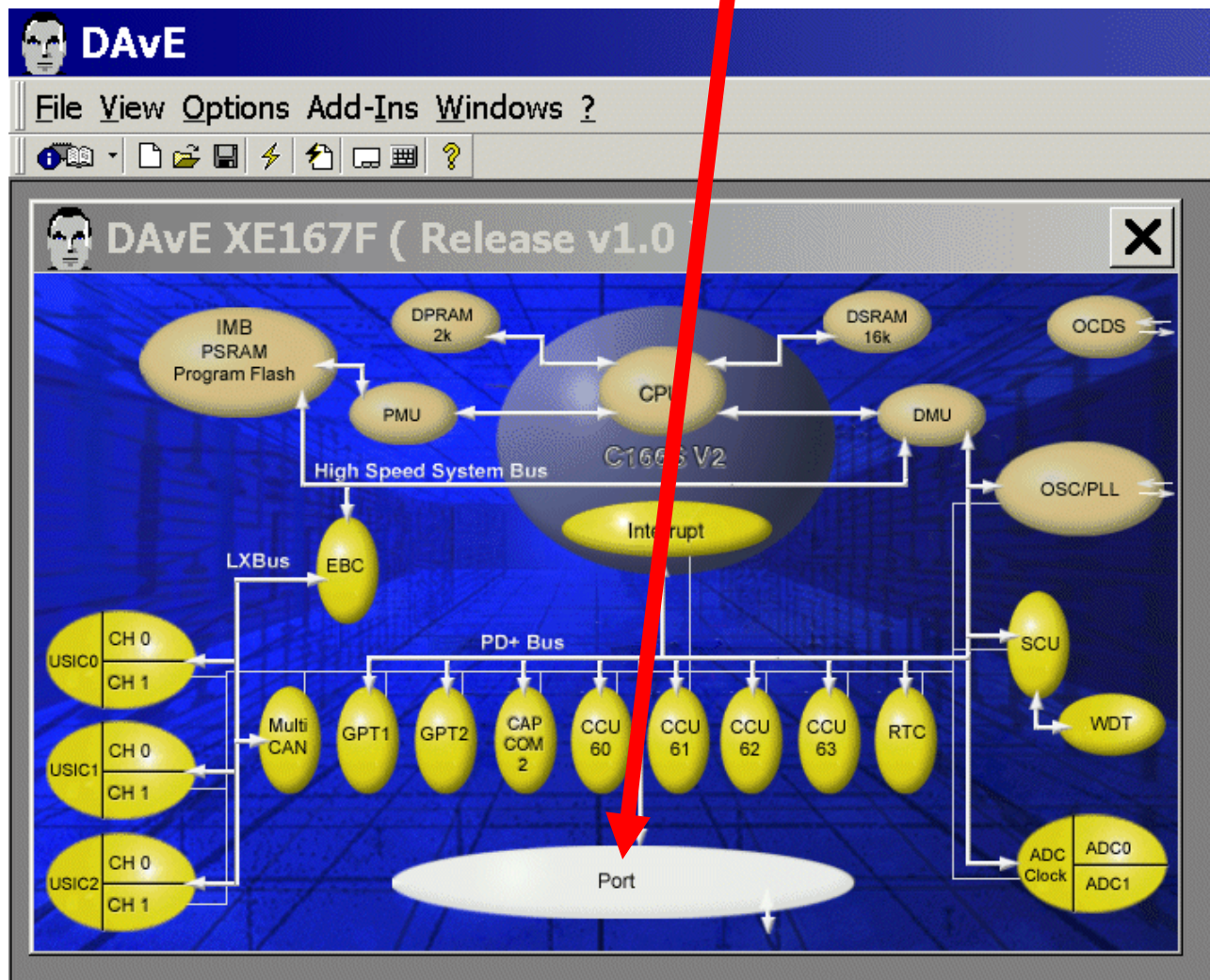
Click Open





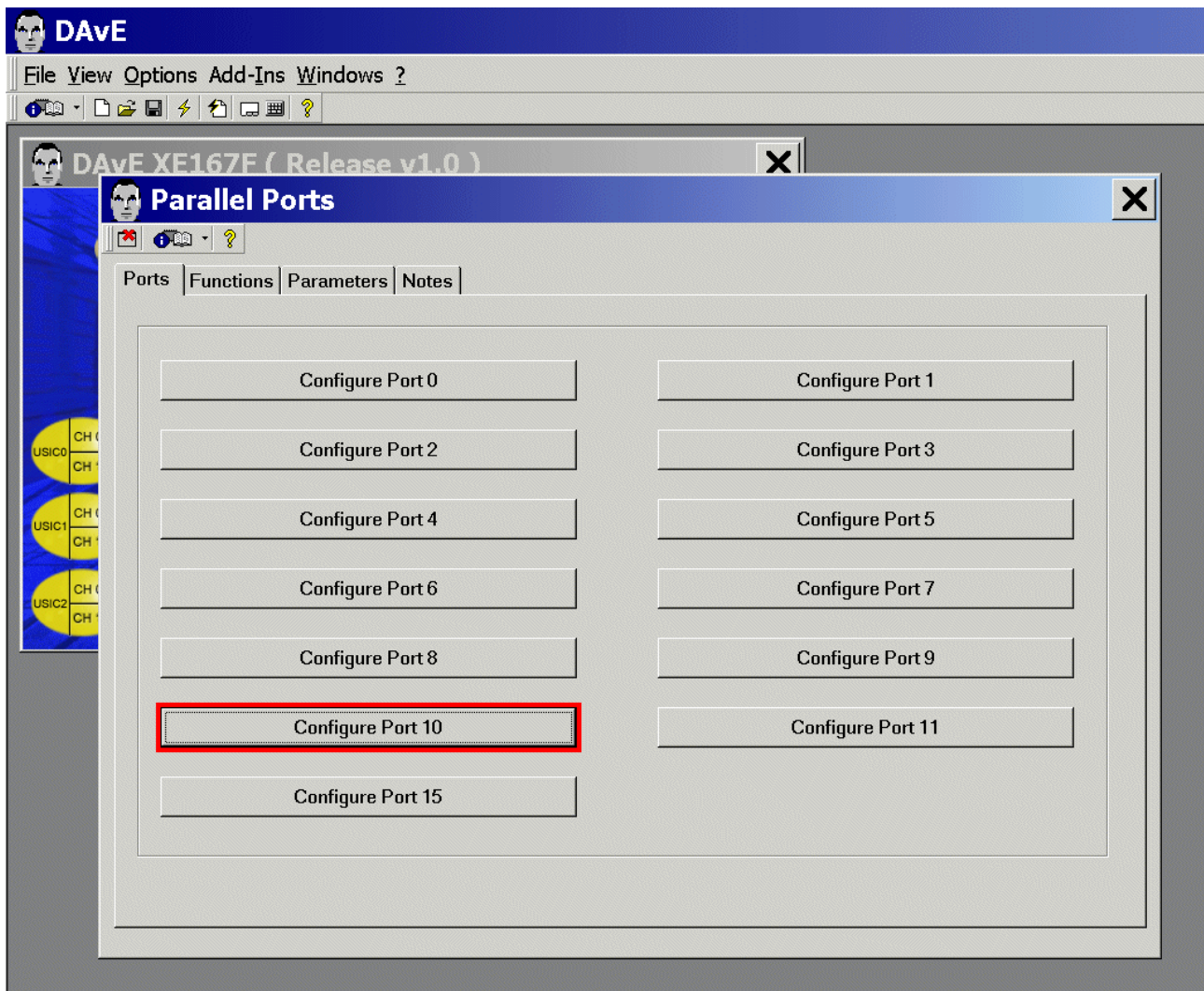
### Reconfiguration of Port 10 Low:

The (re)configuration window/dialog can be opened by clicking the specific block/module (Port).



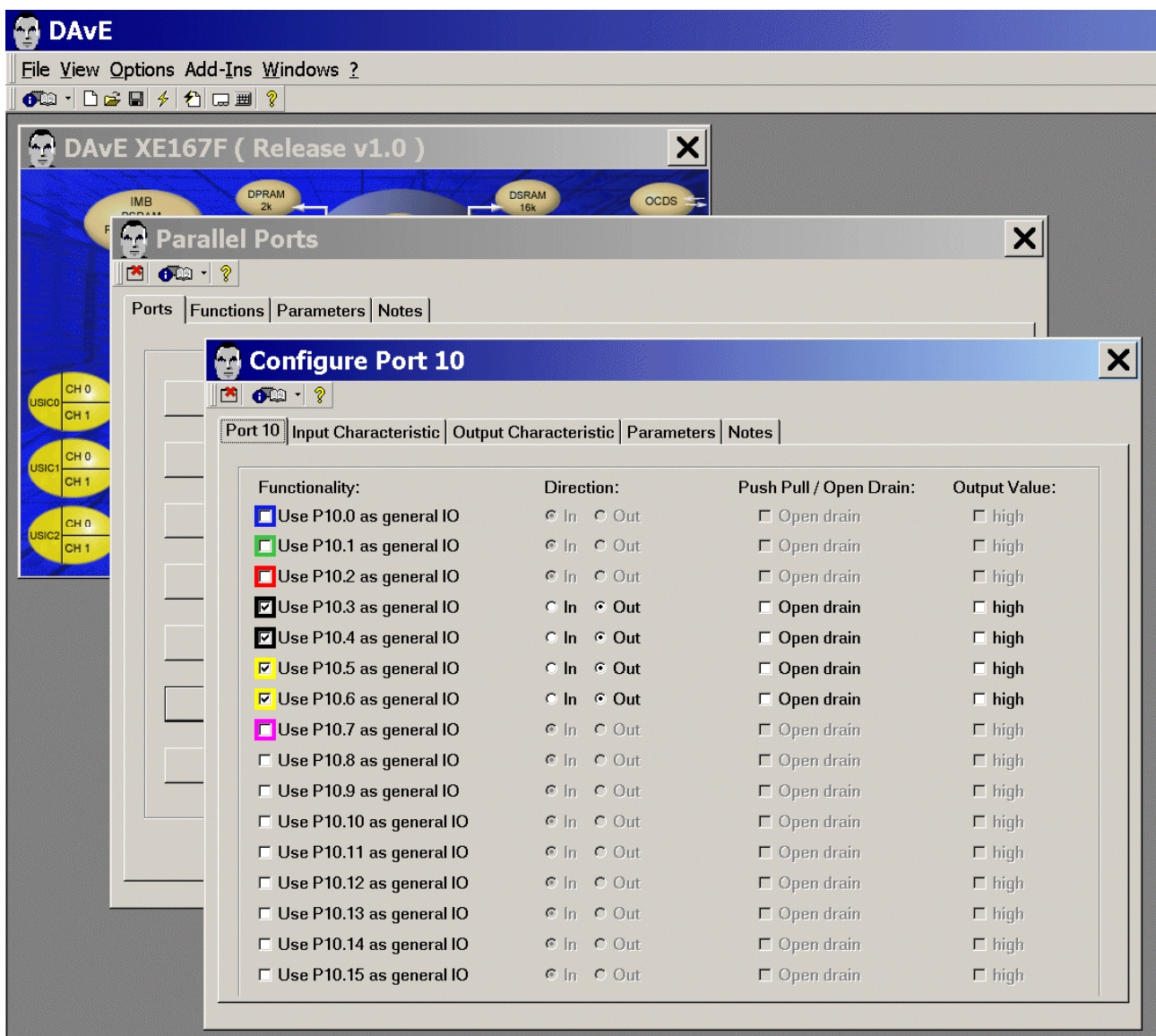


Ports: click “Configure Port 10”



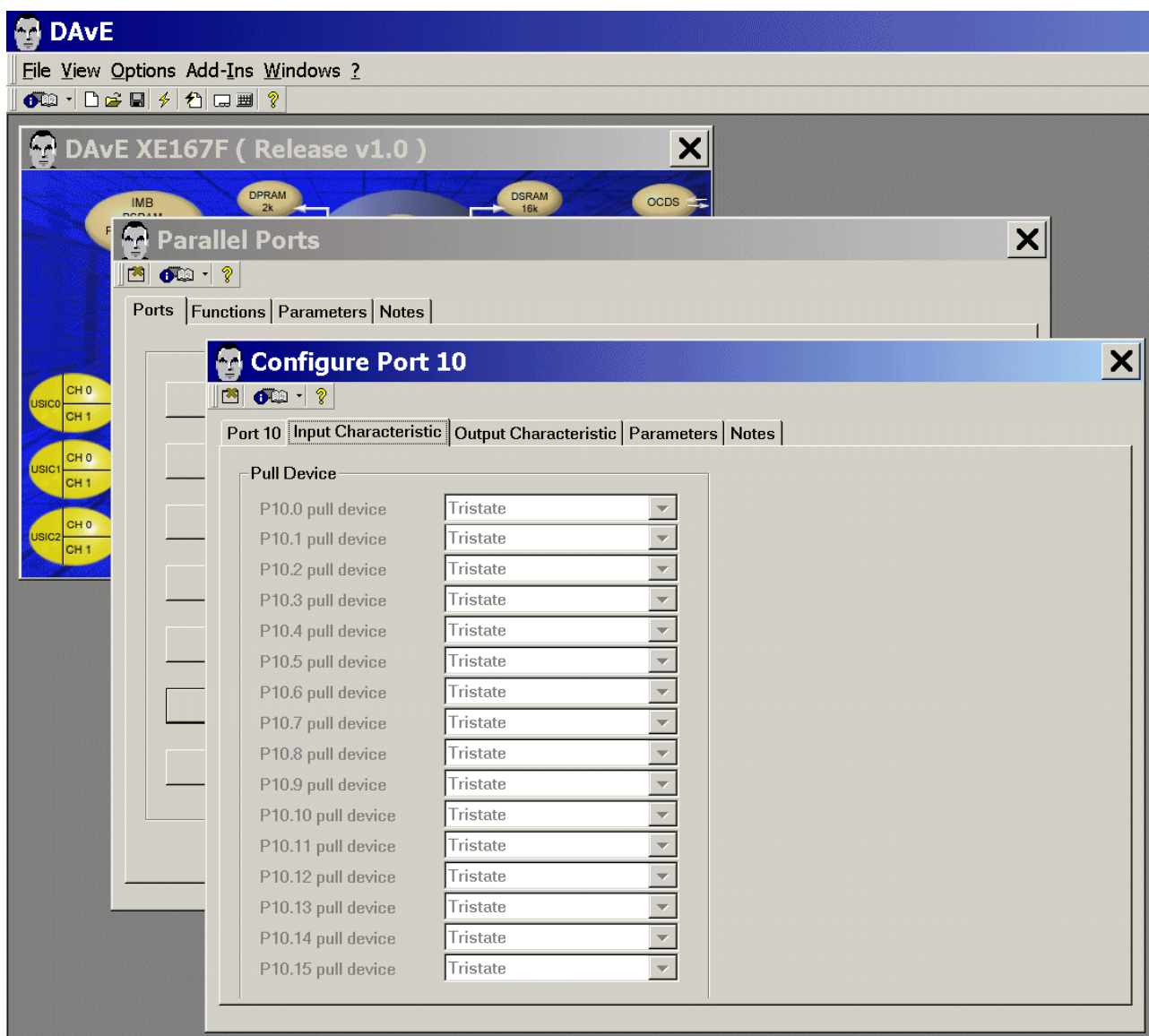
Port 10: Functionality: click to unselect ☐ Use P10.0 to free GPIO pin for CCU60\_CC60 use  
 Port 10: Functionality: click to unselect ☐ Use P10.1 to free GPIO pin for CCU60\_CC61 use  
 Port 10: Functionality: click to unselect ☐ Use P10.2 to free GPIO pin for CCU60\_CC62 use  
 Port 10: Functionality: click to unselect ☐ Use P10.7 to free GPIO pin for CCU60\_COUT63 use

Port 10: Functionality: click/check ☒ Use P10.3 as general IO - Direction: click/check ☐ In ☒ Out  
 Port 10: Functionality: click/check ☒ Use P10.4 as general IO - Direction: click/check ☐ In ☒ Out  
 Port 10: Functionality: click/check ☒ Use P10.5 as general IO - Direction: click/check ☐ In ☒ Out  
 Port 10: Functionality: click/check ☒ Use P10.6 as general IO - Direction: click/check ☐ In ☒ Out

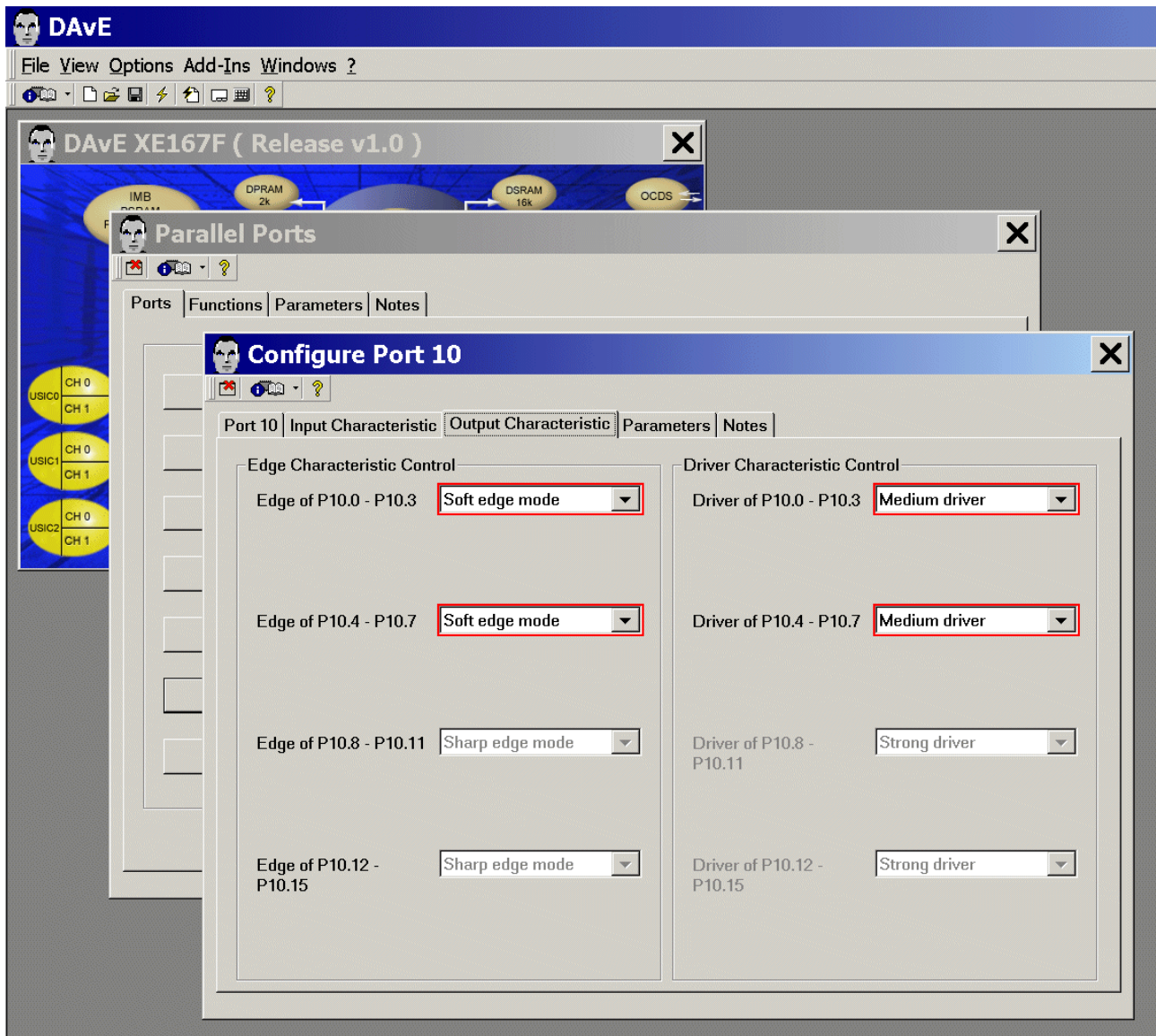




Input Characteristic: (do nothing)

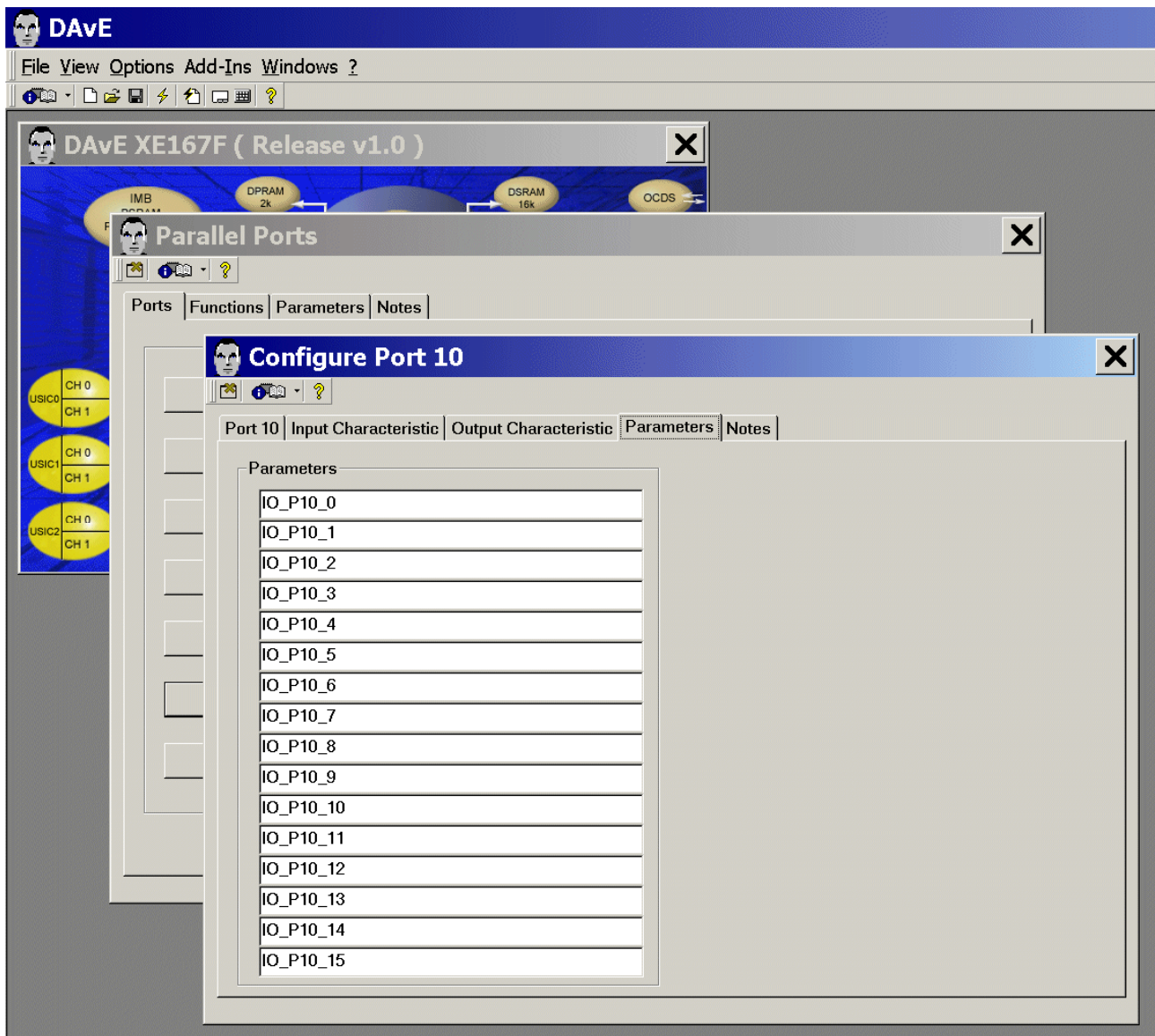


Output Characteristic: Driver Characteristic Control: Driver of P10.0-P10.3: **select** Medium driver  
 Output Characteristic: Driver Characteristic Control: Driver of P10.4-P10.7: **select** Medium driver  
 Output Characteristic: Edge Characteristic Control: Edge of P10.0-P10.3: **select** Soft edge mode  
 Output Characteristic: Edge Characteristic Control: Edge of P10.4-P10.7: **select** Soft edge mode




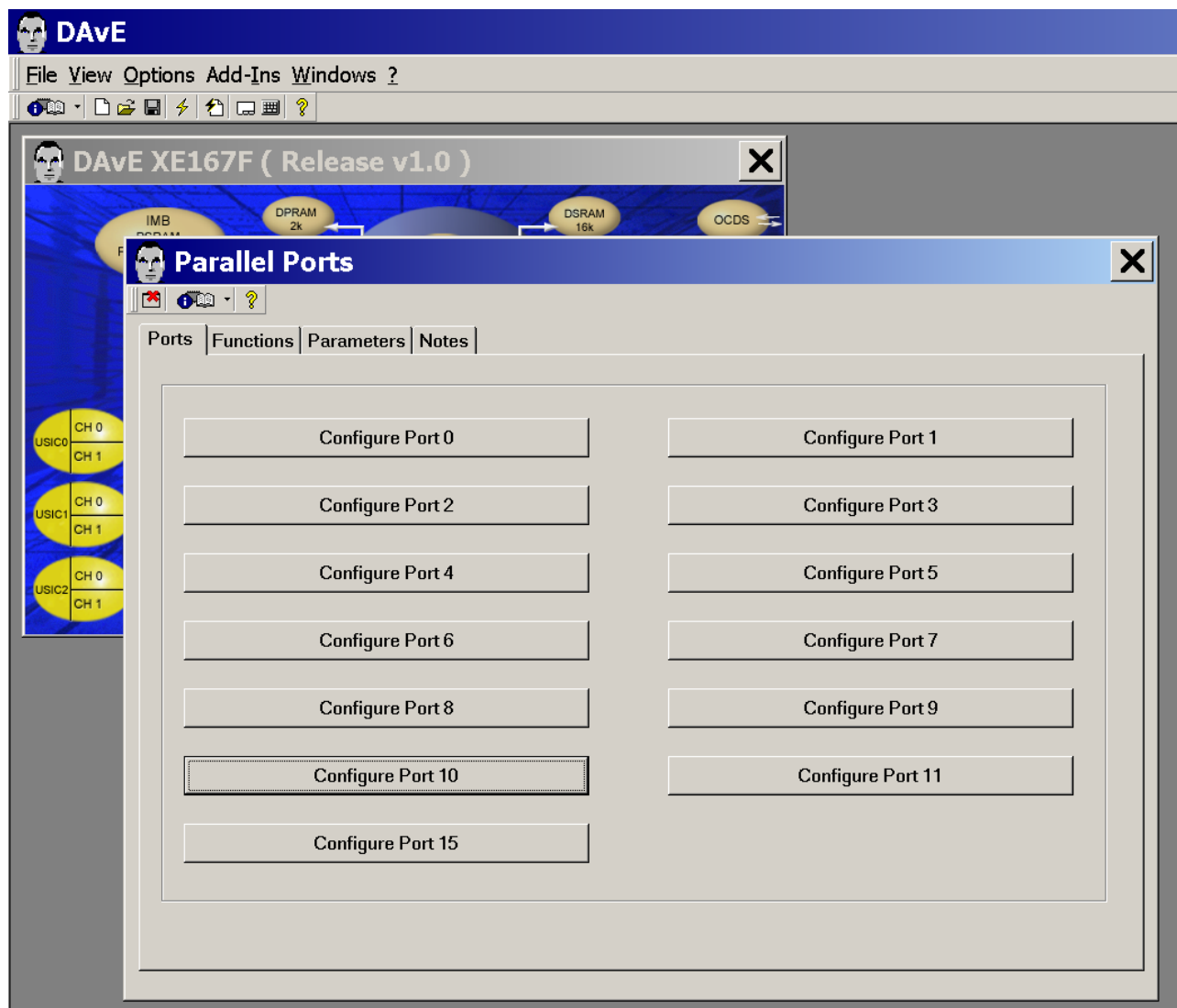


Parameters: (do nothing)

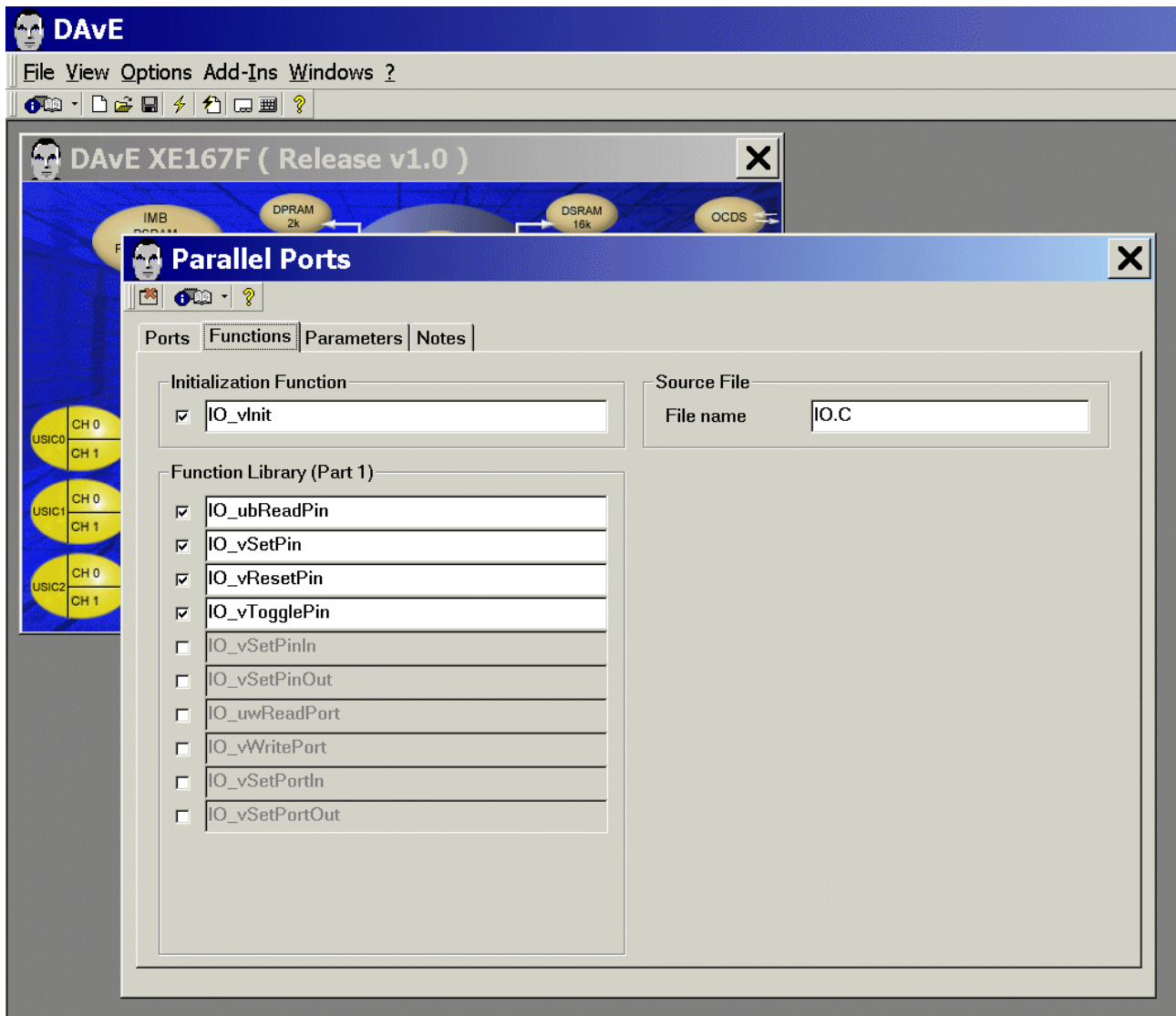


Notes: If you wish, you can insert your comments here.

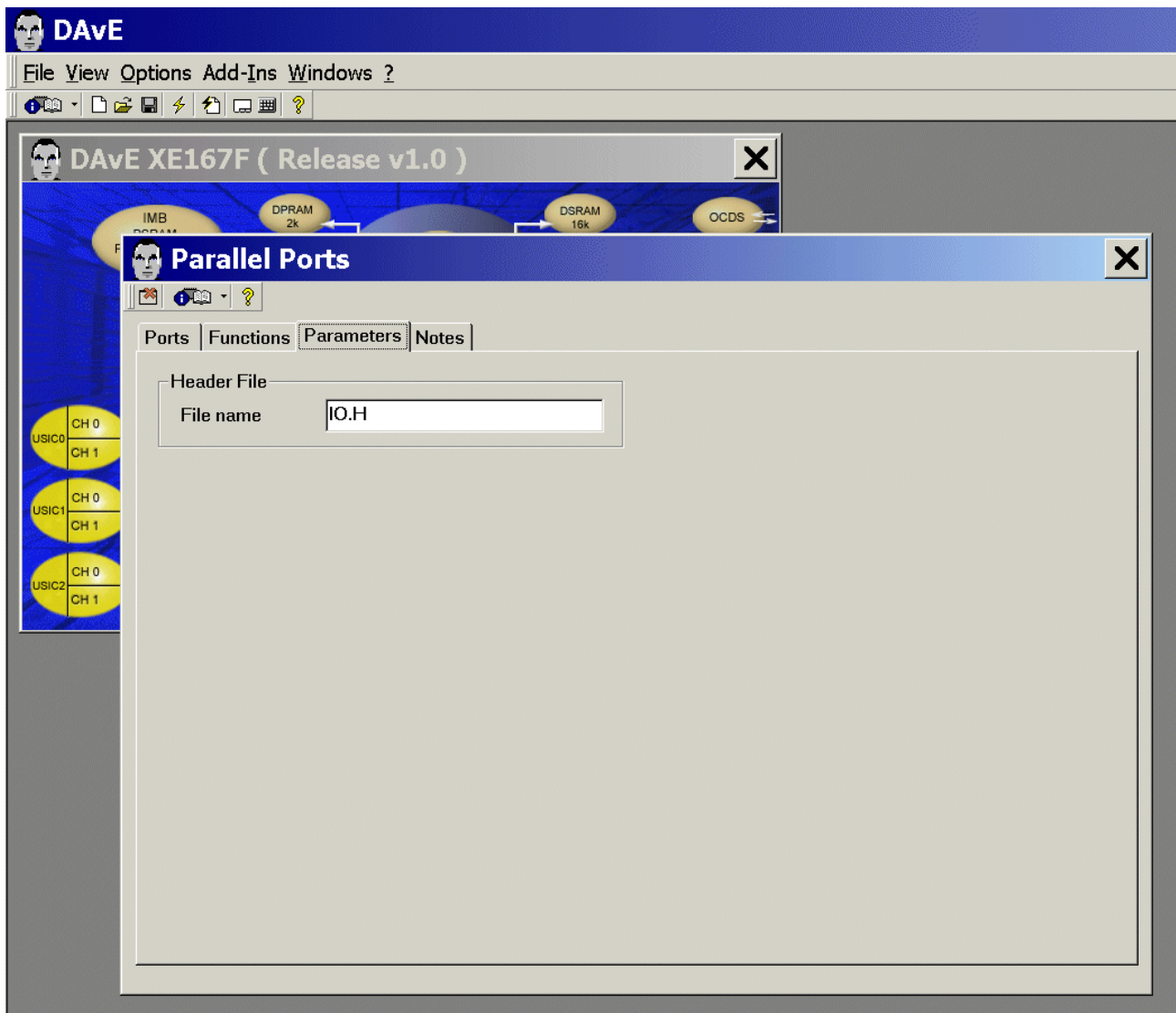
Exit and Save this dialog now by clicking  the close button.




Functions: (do nothing)



Parameters: (do nothing)



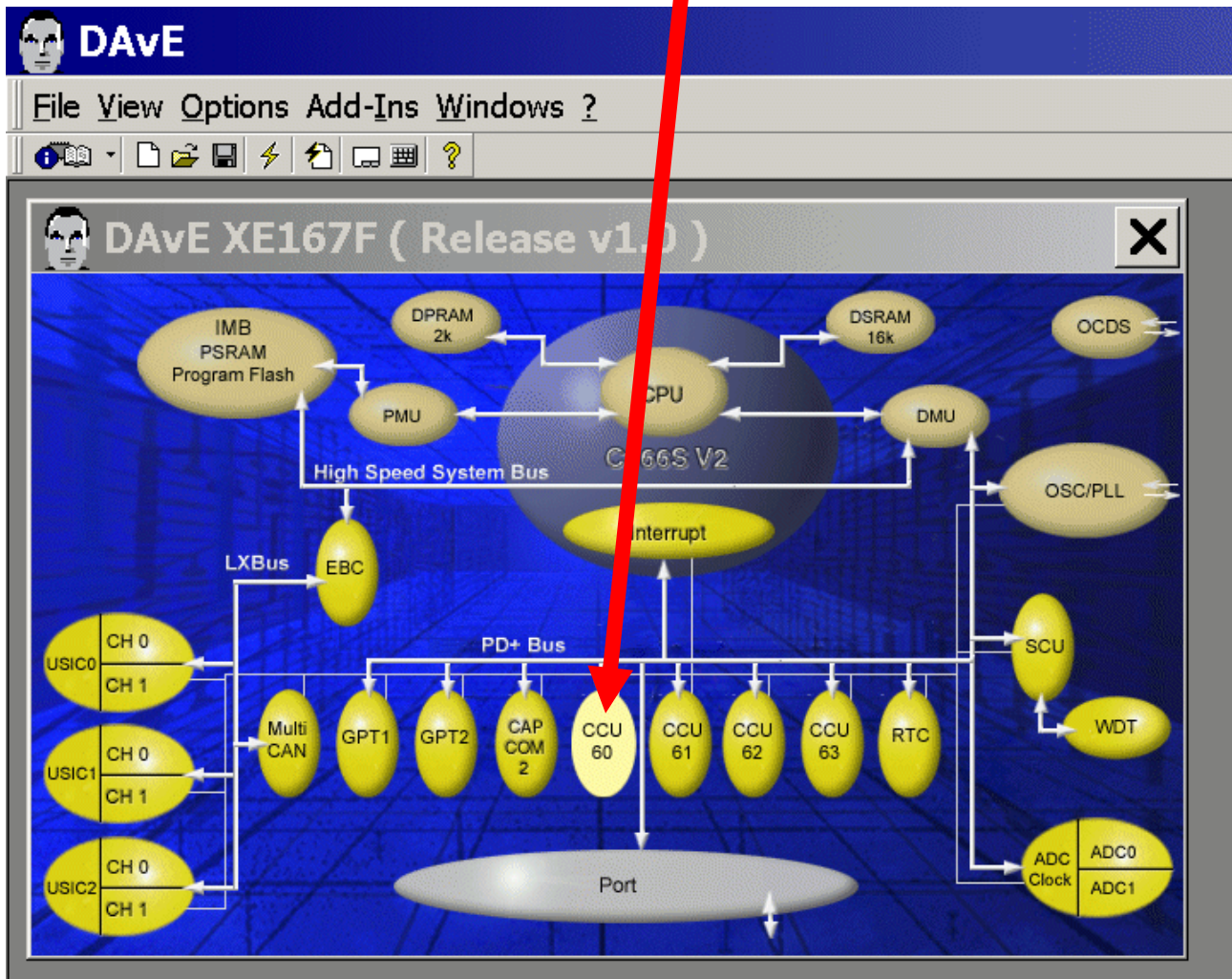
Notes: If you wish, you can insert your comments here.

Exit and Save this dialog now by clicking  the close button.

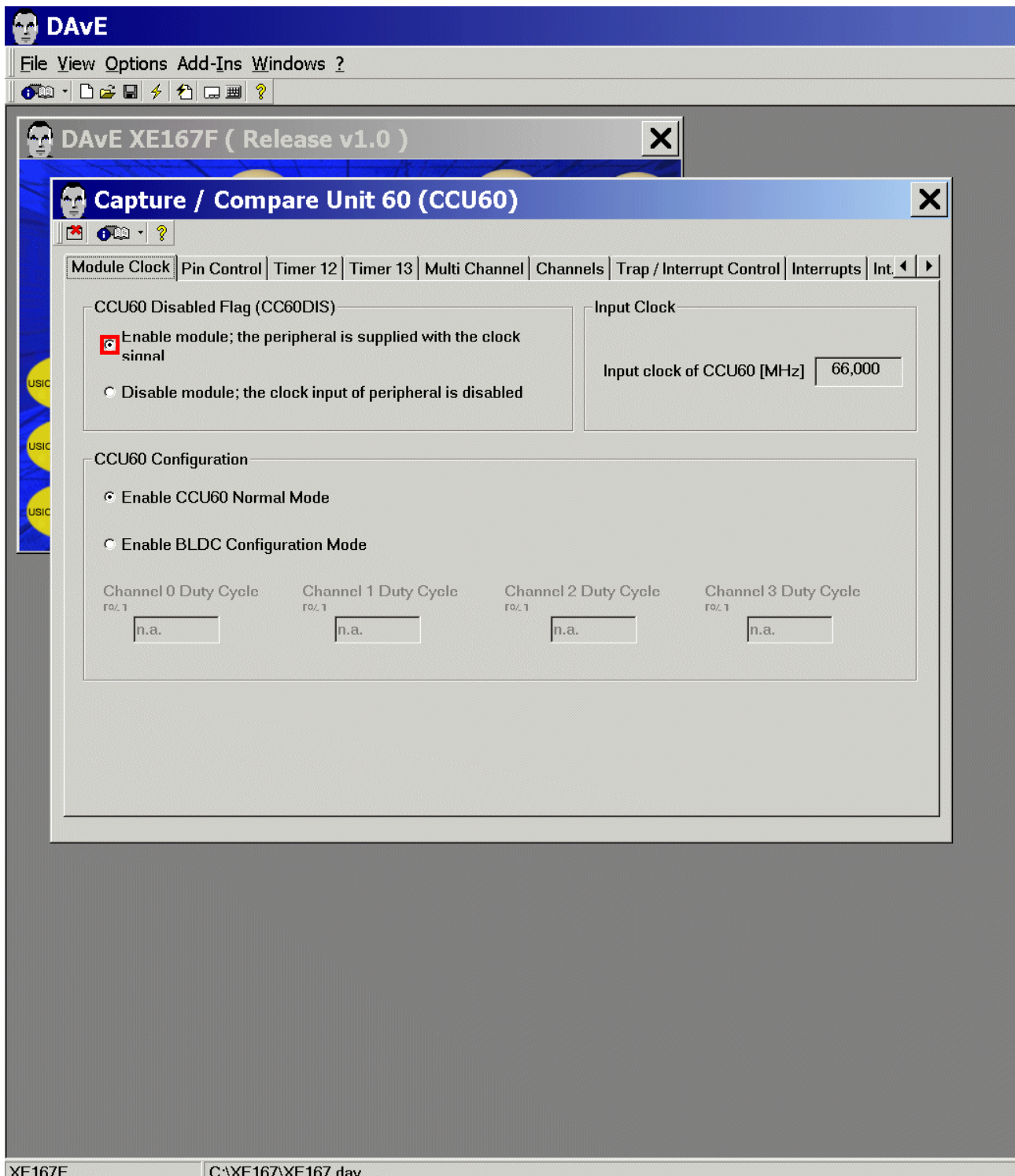


Configuration of the CCU60 module:

The configuration window/dialog can be opened by clicking the specific block/module (CCU60).



CCU60: Module Clock: **click** ☒ Enable module





CCU60:

Pin Control: Control of Pins CC6x and CC6xIN: CC60: select Use pin CC60 as Output (P10.0)

CCU60:

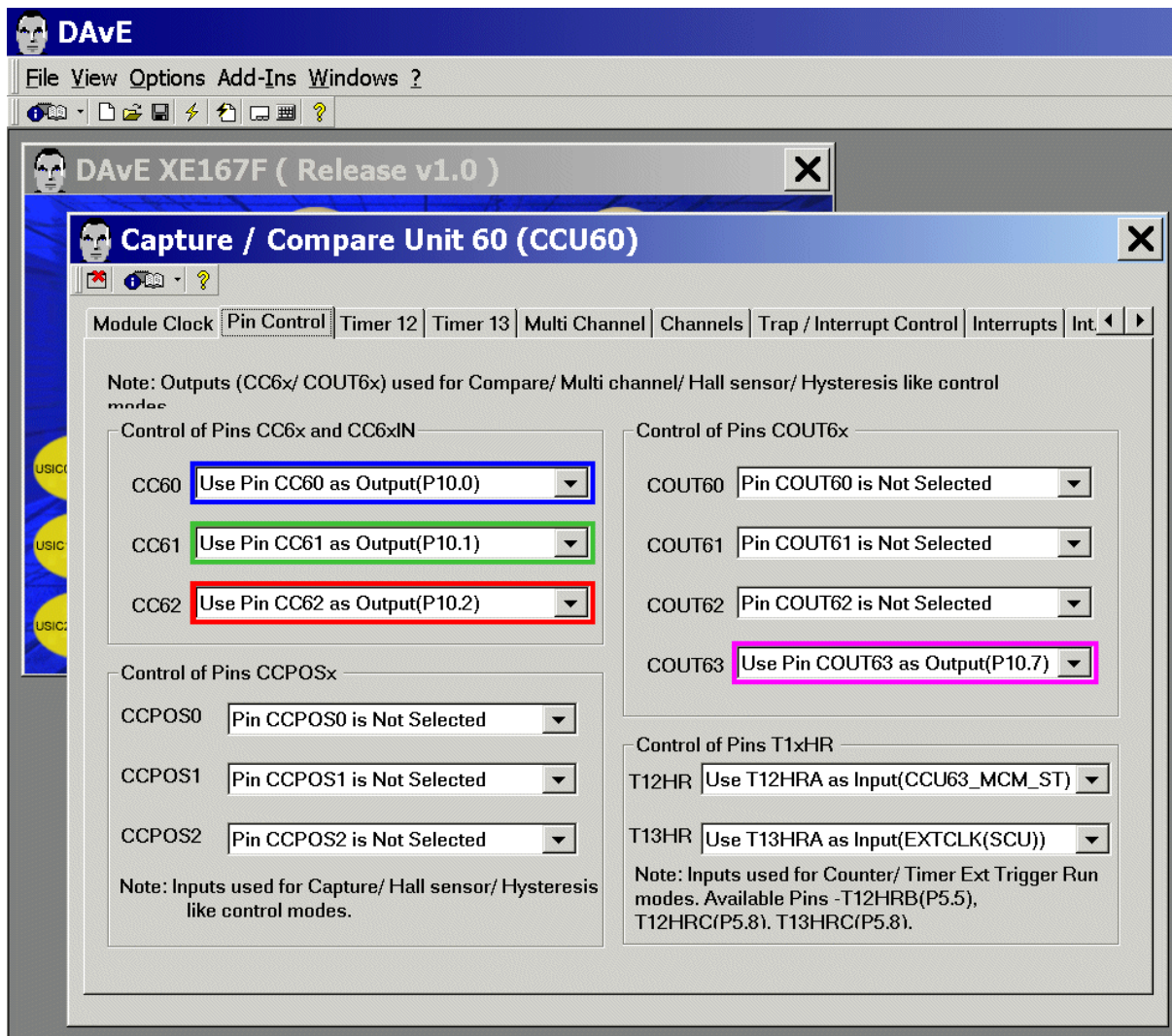
Pin Control: Control of Pins CC6x and CC6xIN: CC61: select Use pin CC61 as Output (P10.1)

CCU60:

Pin Control: Control of Pins CC6x and CC6xIN: CC62: select Use pin CC62 as Output (P10.2)

CCU60:

Pin Control: Control of Pins COUT6x: COUT63: select Use pin COUT63 as Output (P10.7)



**Remember:**

Port\_10 Low pins used by our PWM module CCU60:

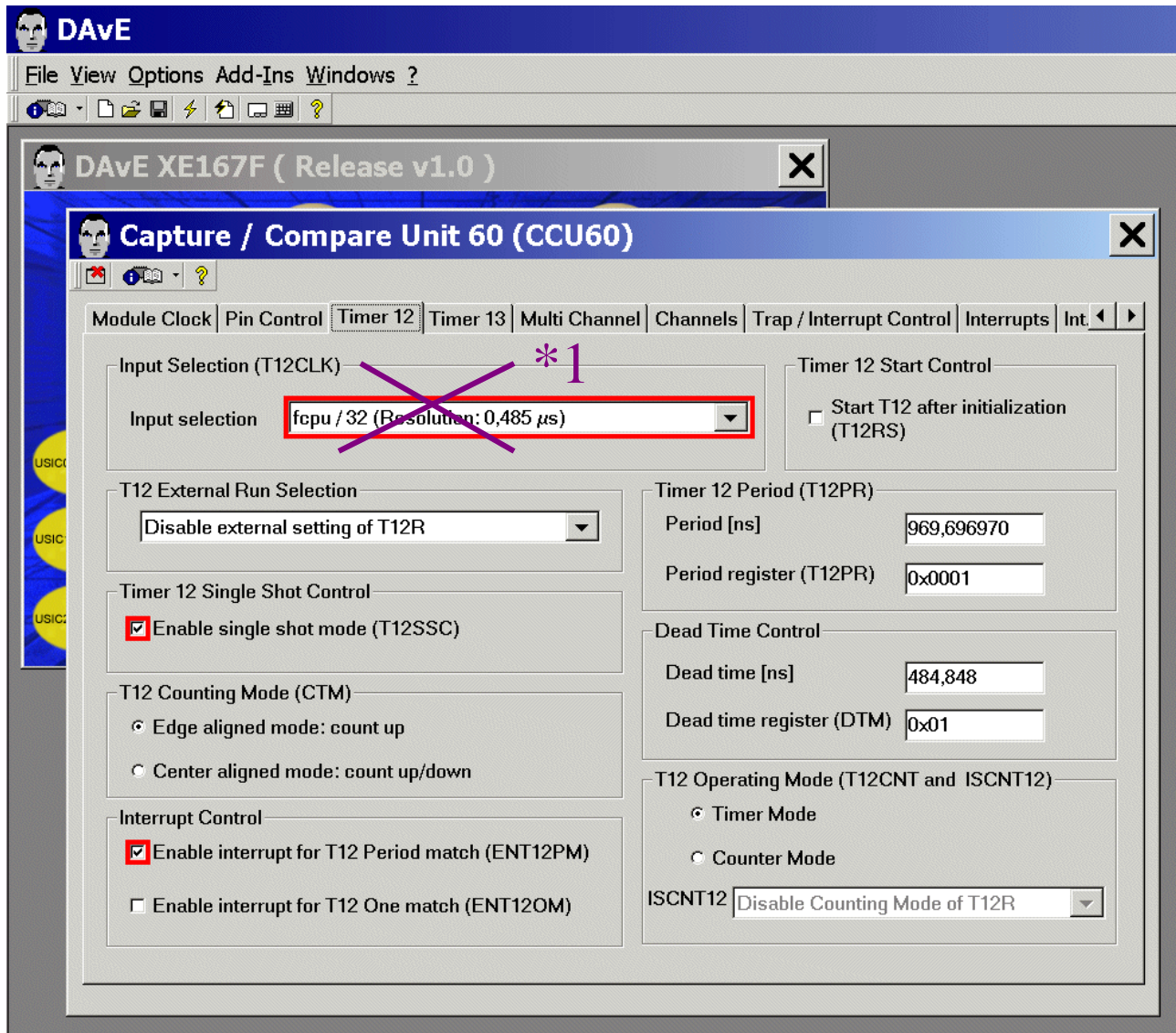
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P10.0	CCU60_CC60	100 (note length, Timer_12)
P10.1	CCU60_CC61	100 (note length, Timer_12)
P10.2	CCU60_CC62	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P10.7	CCU60_COUT63	50 (note frequency, Timer_13)

Timer 12: “note length”:

CCU60: Timer 12: Input Selection: Input selection: choose  $f_{CCU60}/32 \rightarrow$  Resolution = 124,12  $\mu$ s \*1

CCU60: Timer 12: T12 Single Shot Control: click ☒ Enable single shot mode (T12SSC)

CCU60: Timer 12: Interrupt Control: click ☒ Enable interrupt for T12 Period match



\*1:

Timer 12 Resolution:

66 MHz / 256 (T12PRE=1, done by software) / 32 = 8.056,64 Hz  $\rightarrow$  Resolution = 124,12  $\mu$ s



<<< !!! click here to see more information about music !!! >>>





**Note:**

Unfortunately bit T12PRE is not available in the DAVe dialog.

Source: User's Manual:

The input clock for timer T12 can be from  $f_{CCU60}$  to a maximum of  $f_{CCU60}/128$  and is configured by bit field T12CLK. In order to support higher clock frequencies, an additional prescaler factor of  $1/256$  can be enabled for the prescaler of T12 if bit T12PRE = 1.

Adobe Reader - [xe166\_um\_v2.0\_2007\_12\_vol2per.pdf]

File Edit View Document Tools Window Help

125%

**TCTR0**  
Timer Control Register 0  
XSFR(2C<sub>H</sub>)  
Reset Value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STE 13	T13R	T13 PRE	T13CLK	CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK					
r	rh	rh	rw	rw	rw	rh	rh	rh	rw	rw					

Field	Bits	Type	Description
T12CLK	[2:0]	rw	<b>Timer T12 Input Clock Select</b> Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation $f_{T12} = f_{CC6} / 2^{<T12CLK>}$ . 000 <sub>B</sub> $f_{T12} = f_{CC6}$ 001 <sub>B</sub> $f_{T12} = f_{CC6} / 2$ 010 <sub>B</sub> $f_{T12} = f_{CC6} / 4$ 011 <sub>B</sub> $f_{T12} = f_{CC6} / 8$ 100 <sub>B</sub> $f_{T12} = f_{CC6} / 16$ 101 <sub>B</sub> $f_{T12} = f_{CC6} / 32$ 110 <sub>B</sub> $f_{T12} = f_{CC6} / 64$ 111 <sub>B</sub> $f_{T12} = f_{CC6} / 128$
T12PRE	3	rw	<b>Timer T12 Prescaler Bit</b> In order to support higher clock frequencies, an additional prescaler factor of $1/256$ can be enabled for the prescaler for T12. 0 <sub>B</sub> The additional prescaler for T12 is disabled. 1 <sub>B</sub> The additional prescaler for T12 is enabled.

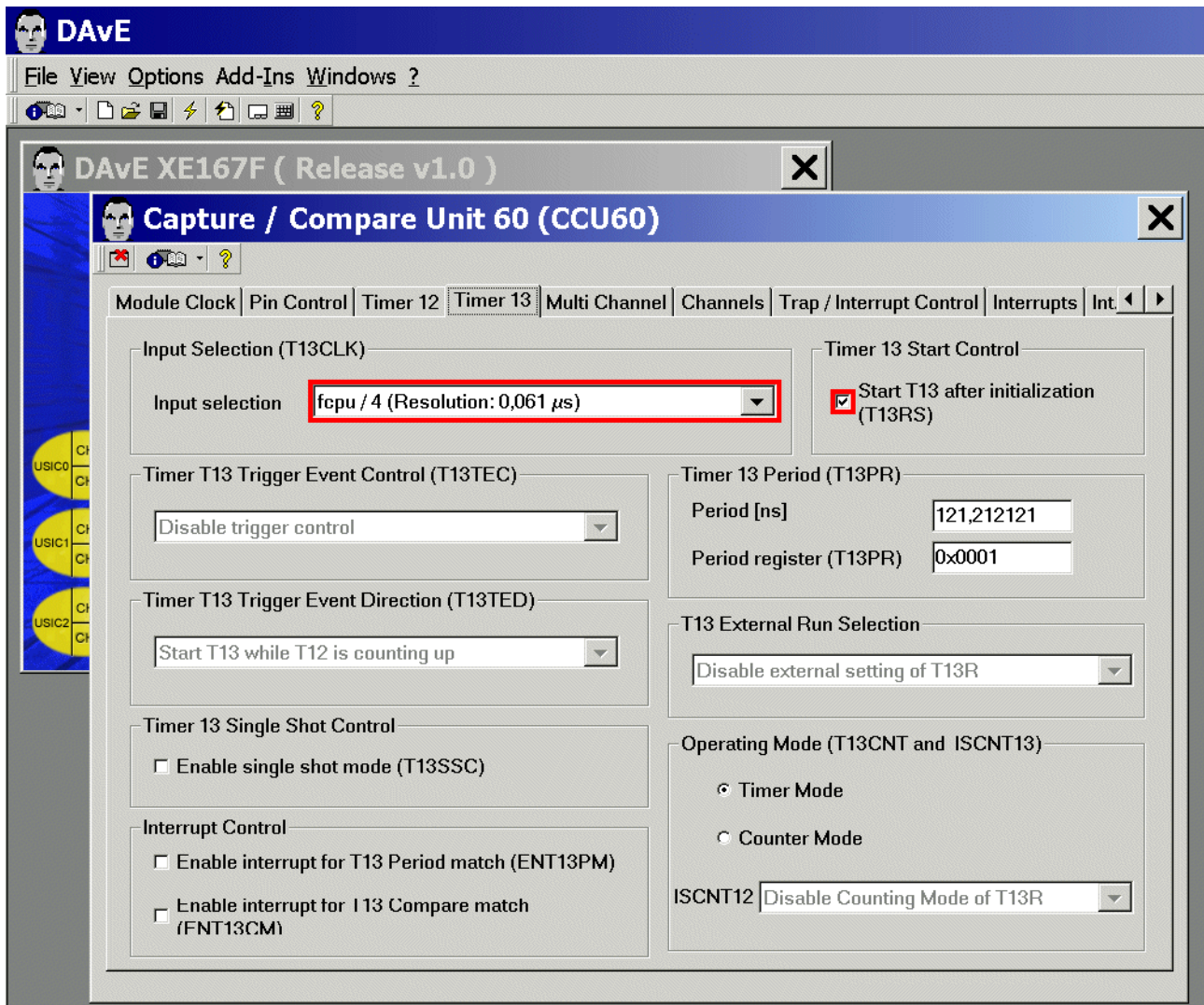
290 of 731

**Timer 12 Resolution:**

$66 \text{ MHz} / 256 \text{ (T12PRE=1, done by software)} / 32 = 8.056,64 \text{ Hz} \rightarrow \text{Resolution} = 124,12 \mu\text{s}$

Timer 13: "note frequency":

CCU60: Timer T13: Input Selection: Input selection select  $f_{CCU60}/4$  (Resolution: 60,606 ns)  
CCU60: Timer T13: Timer 13 Start Control: click ✓ Start T13 after initialization (T13RS)



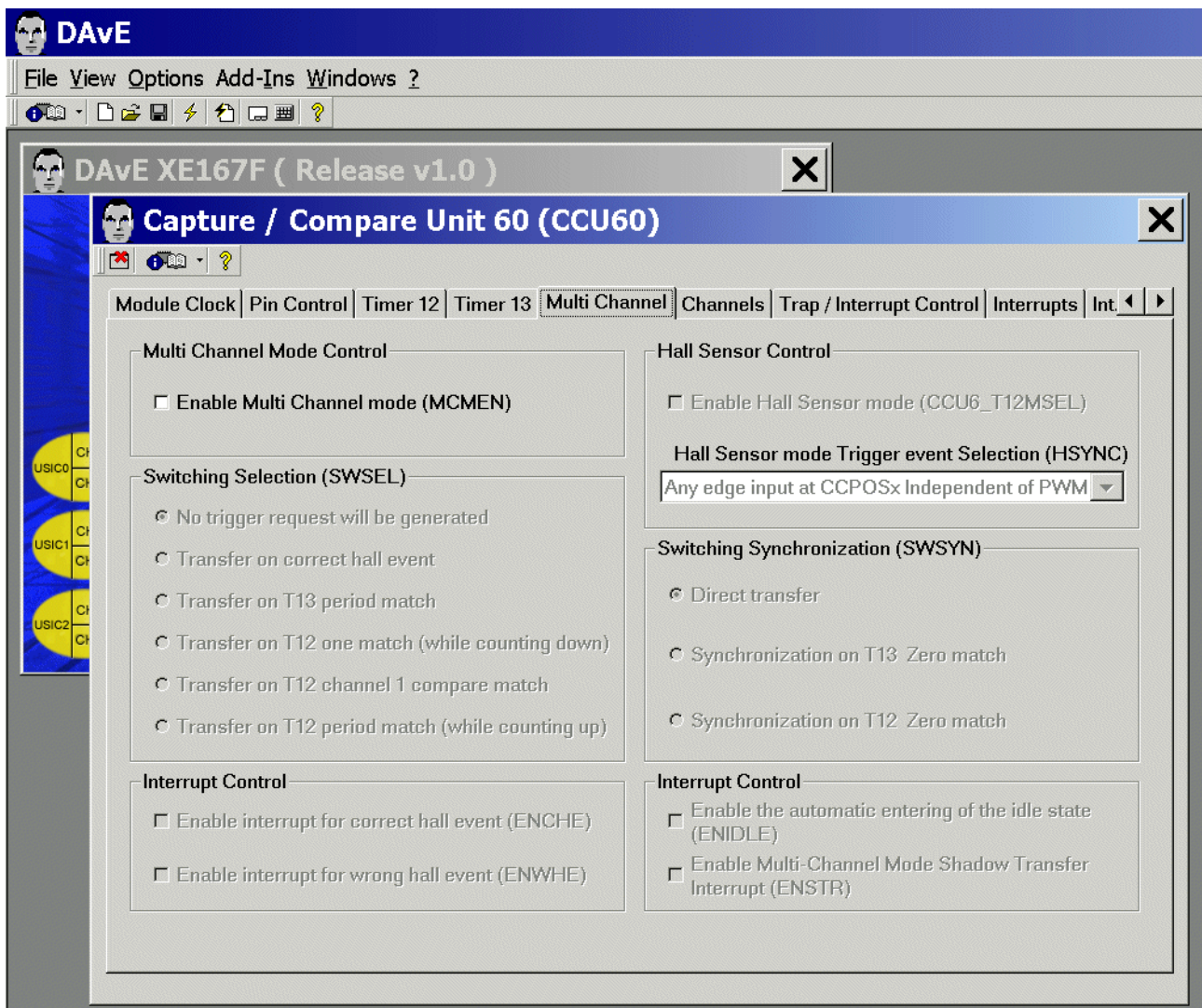
**Note:**

Timer 13 resolution =  $1/(f_{CCU60}/4) = 1/(66\text{MHz}/4) = 60,606 \text{ ns}$ .



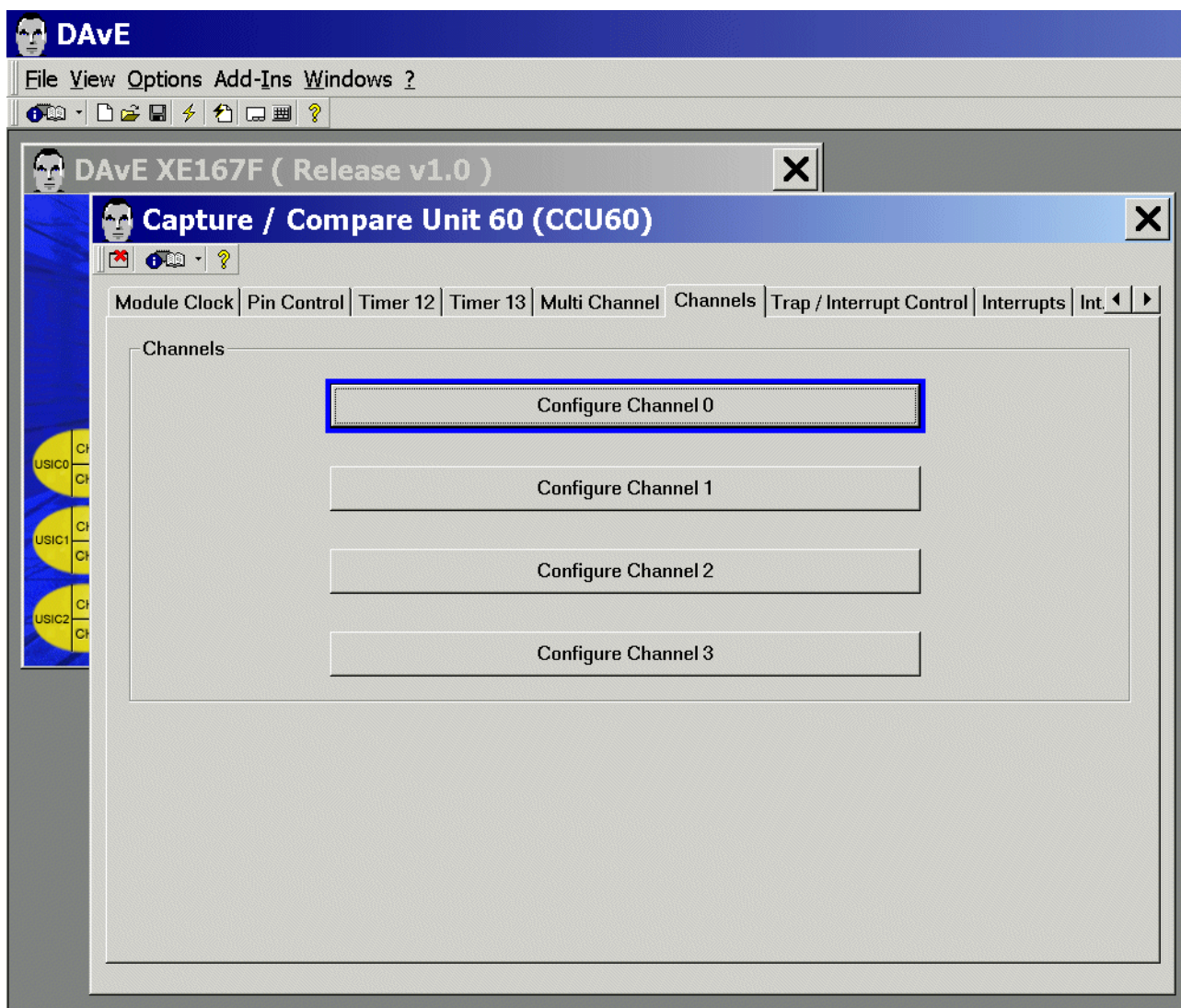
<<< !!! [click here to see more information about music](#) !!! >>>

CCU60: Multi Channel: (do nothing)



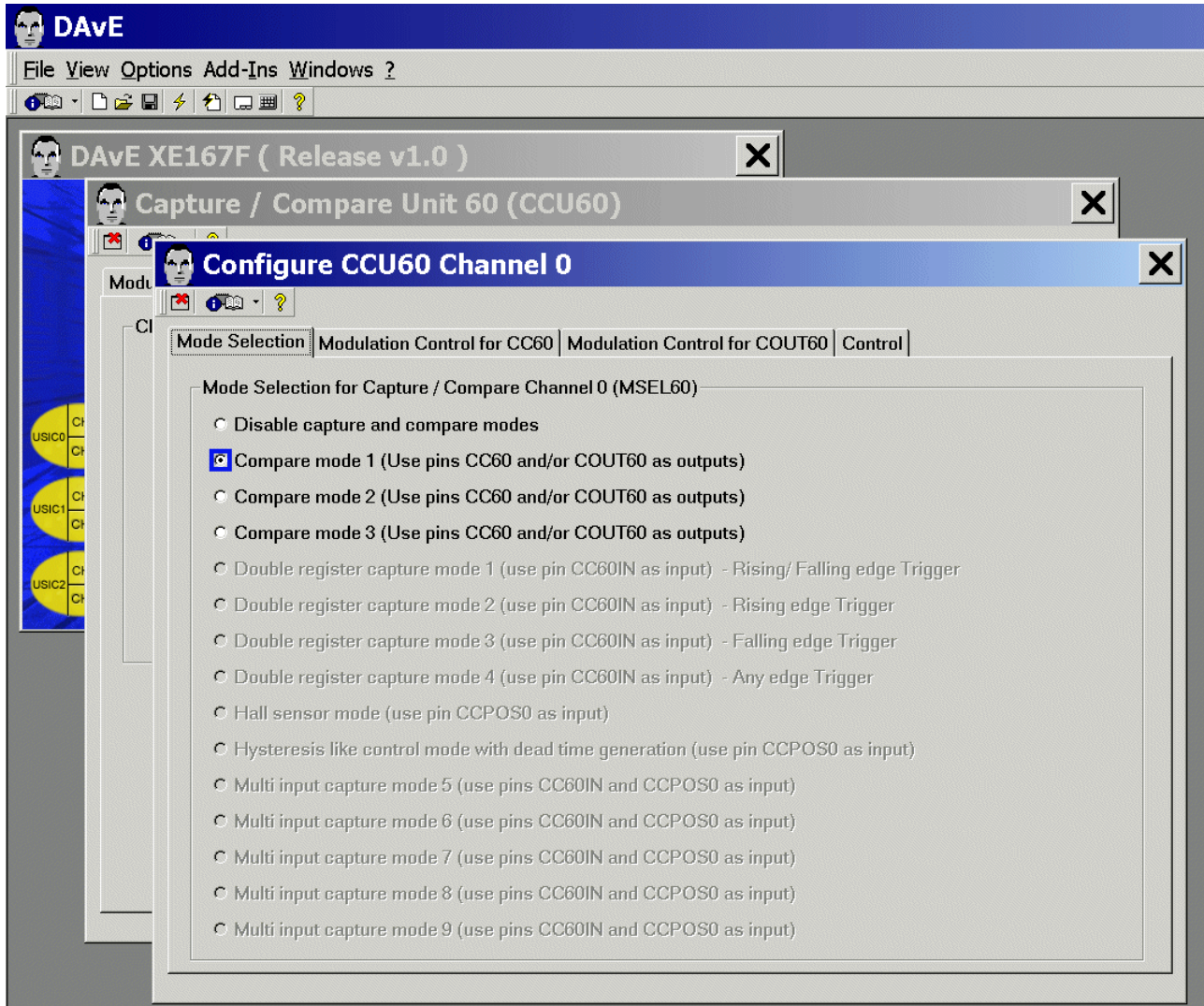


CCU60: Channels: **click** Configure Channel 0



CCU60: Channels: Configure Channel 0:

Mode Selection: Mode Selection for Capture / Compare Channel 0: click ☒ Compare mode 1

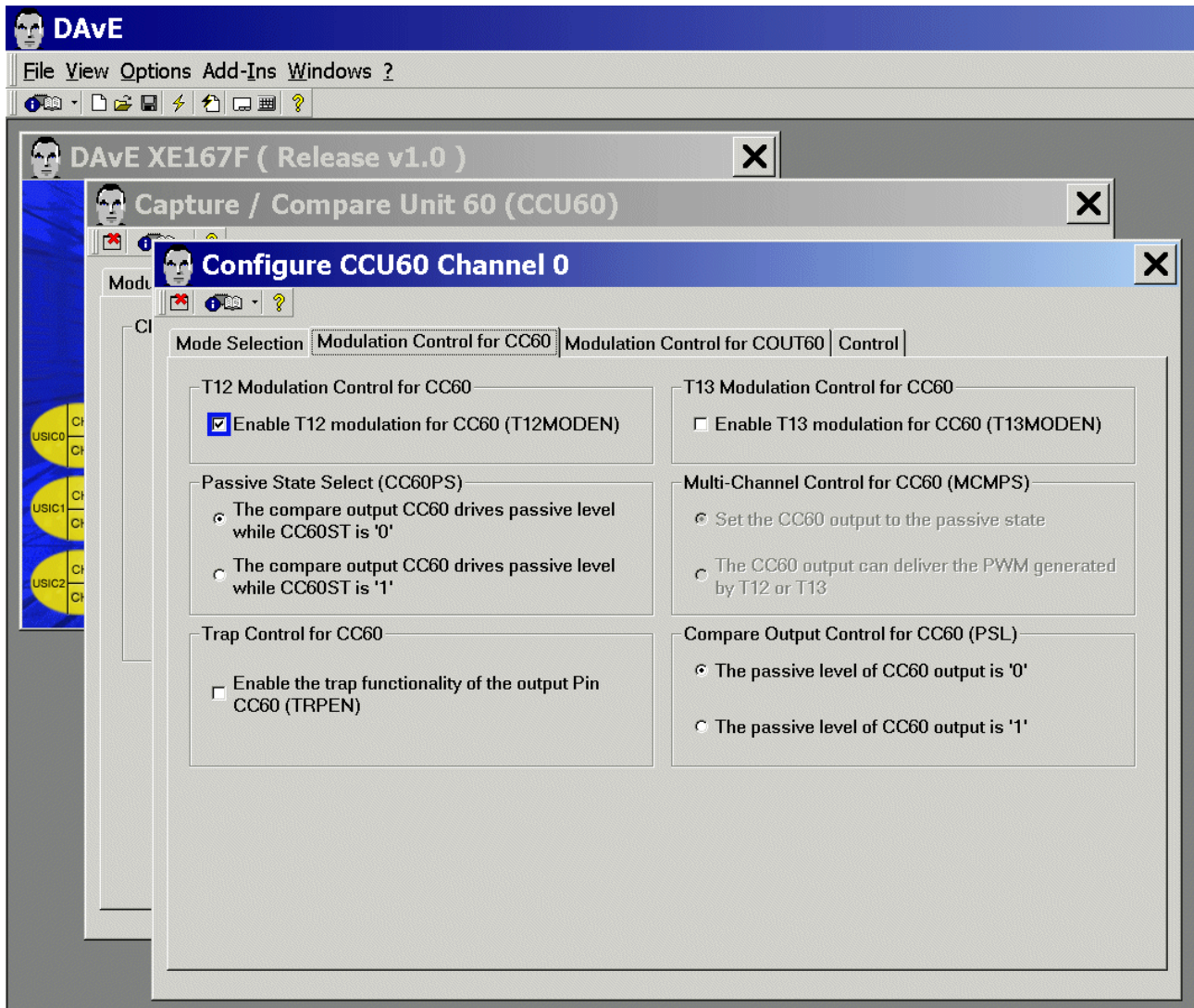




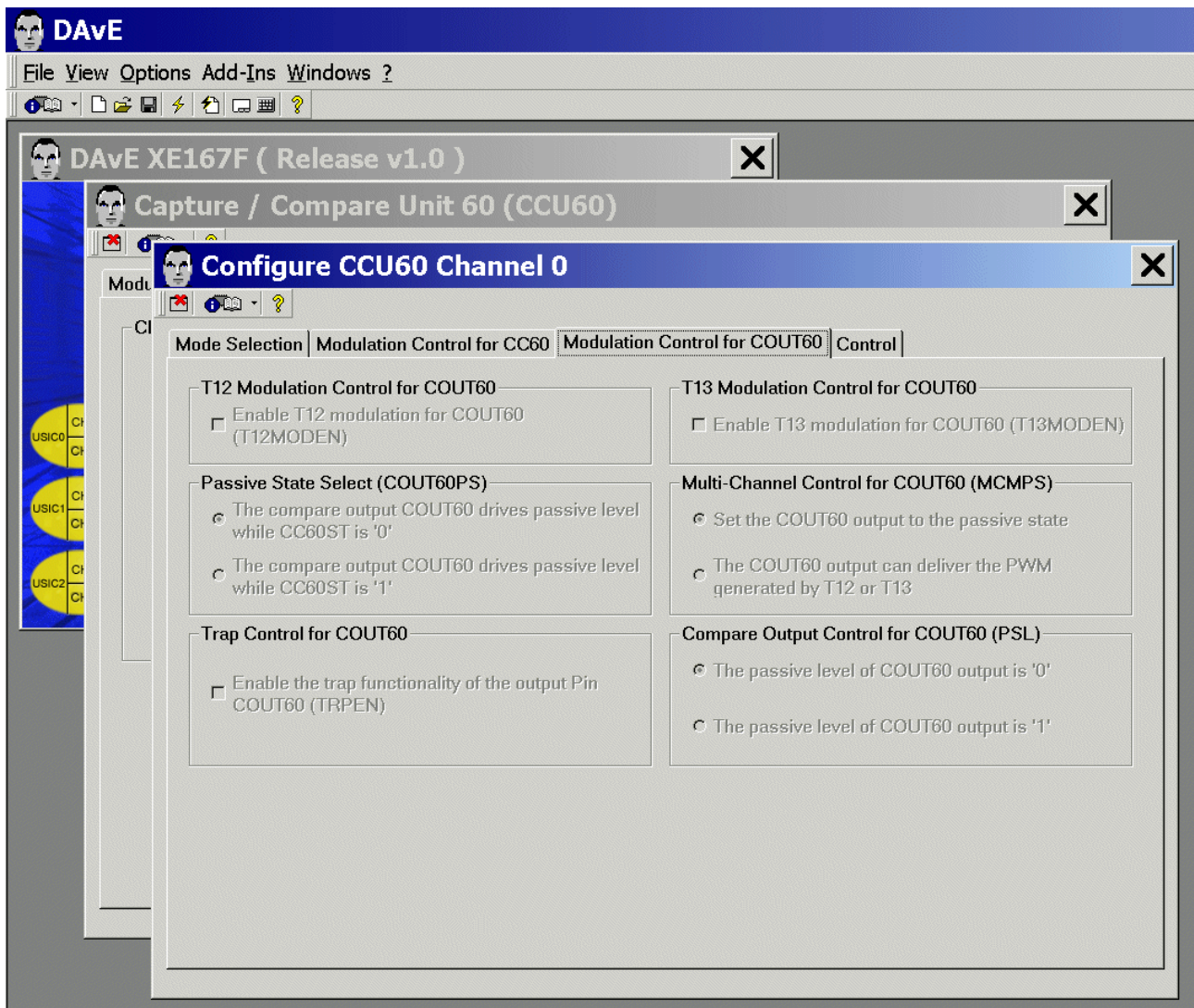
CCU60: Channels: Configure Channel 0:

Modulation Control for CC60:

T12 Modulation Control for CC60: click ☒ Enable T12 modulation for CC60

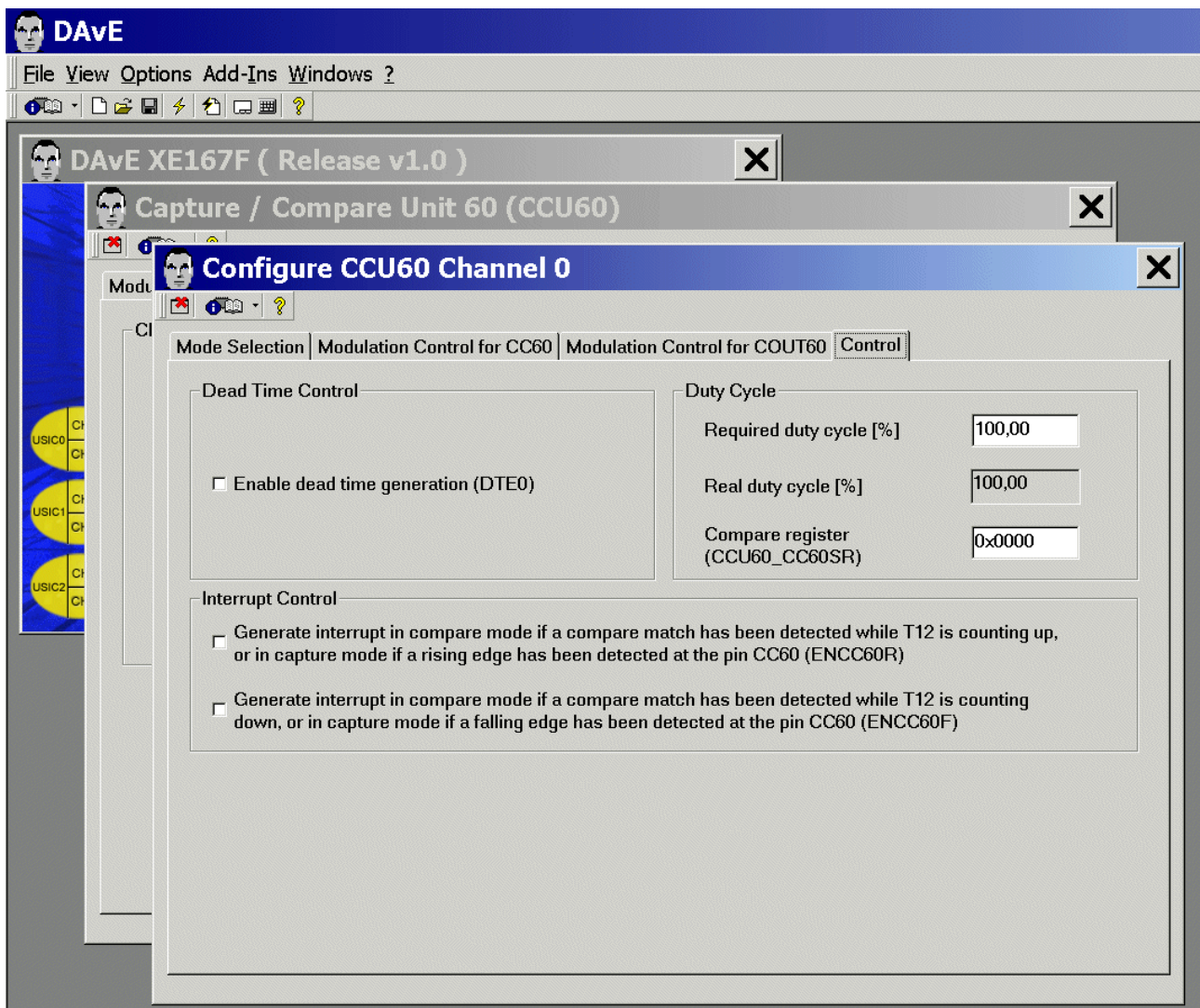



CCU60: Channels: Configure Channel 0: Modulation Control for COUT60: (do nothing)





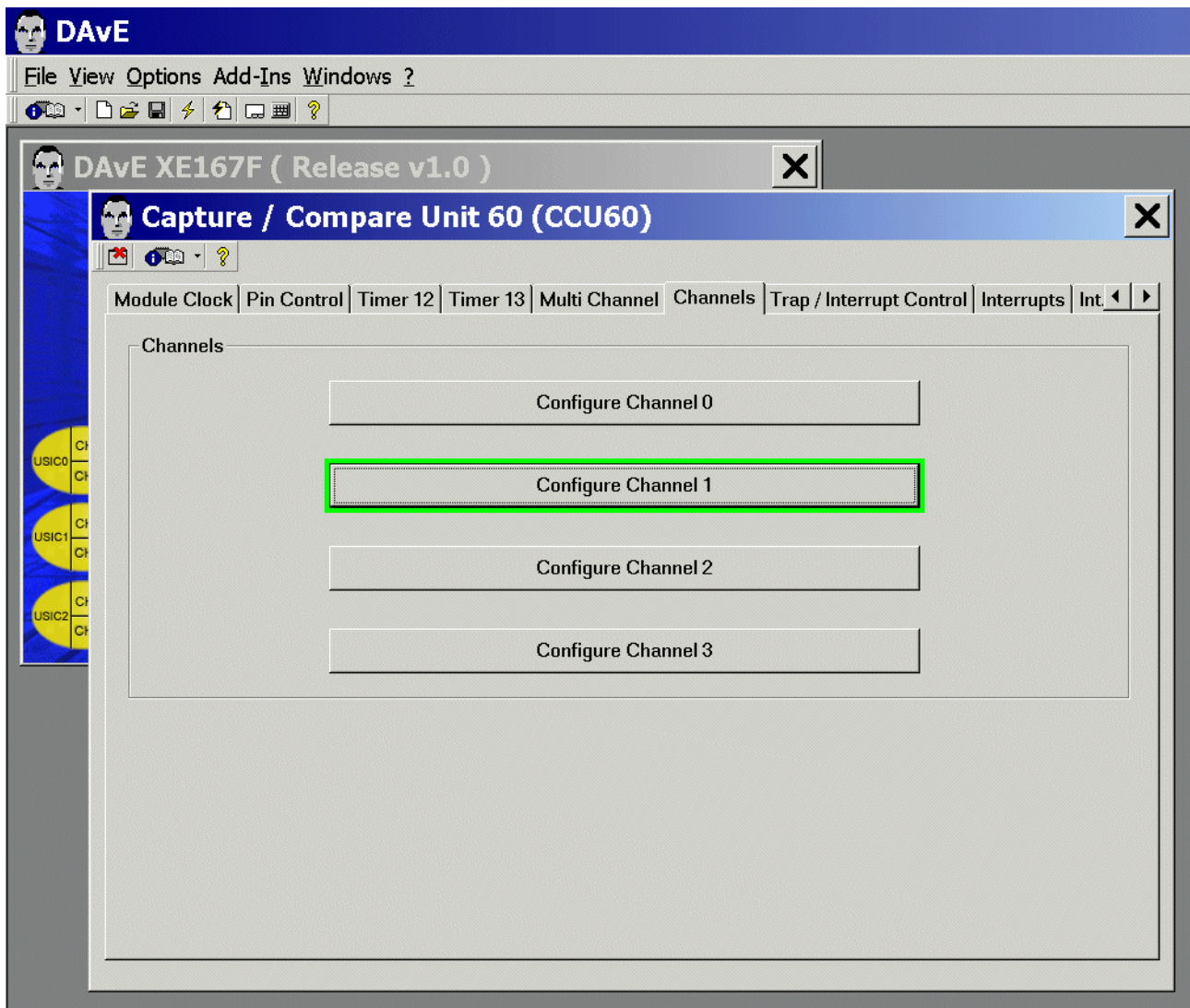
CCU60: Channels: Configure Channel 0: Control: (do nothing)



Exit and Save this dialog now by clicking  the close button.

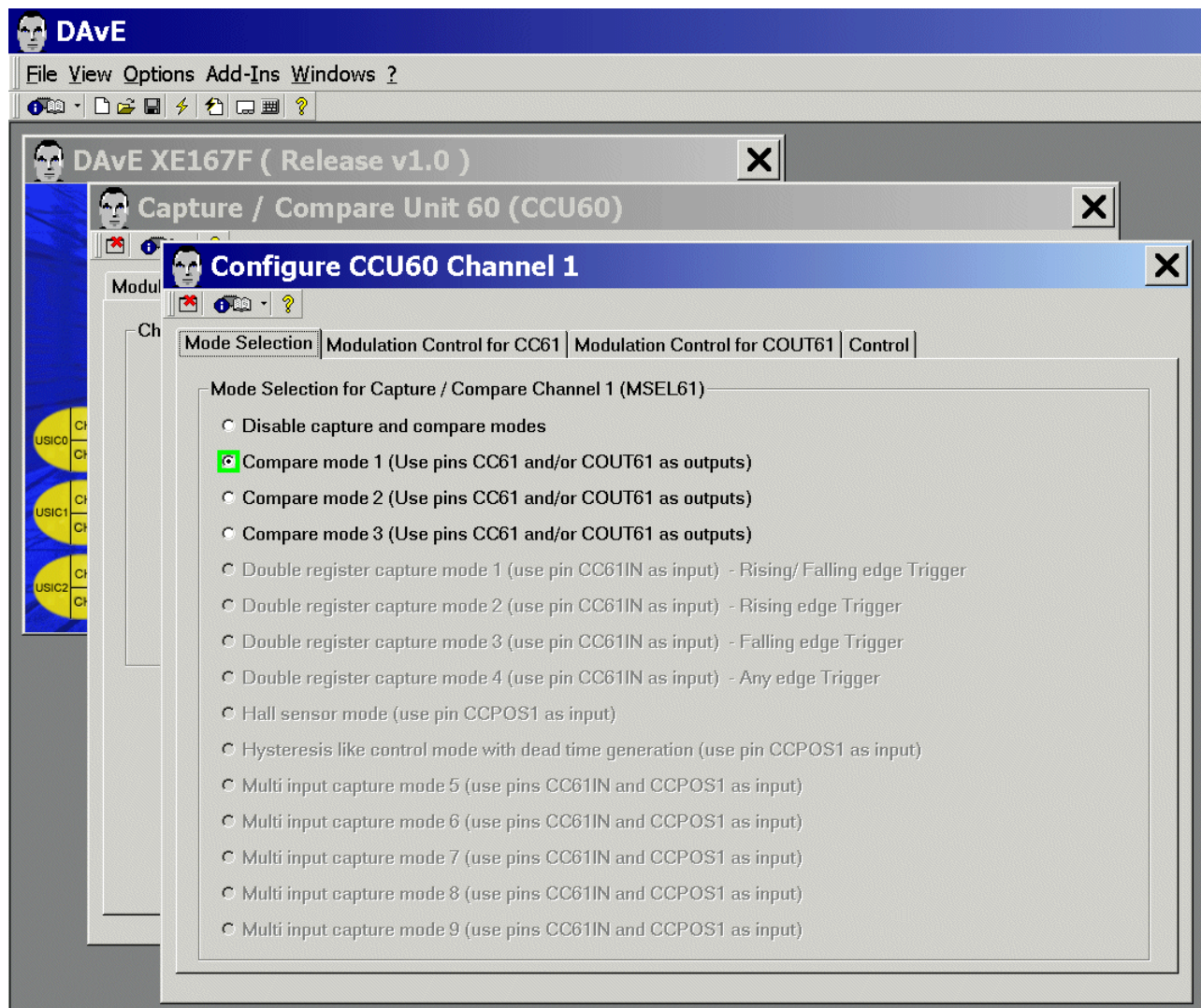


CCU60: Channels: **click** Configure Channel 1



CCU60: Channels: Configure Channel 1:

Mode Selection: Mode Selection for Capture / Compare Channel 1: click ☒ Compare mode 1

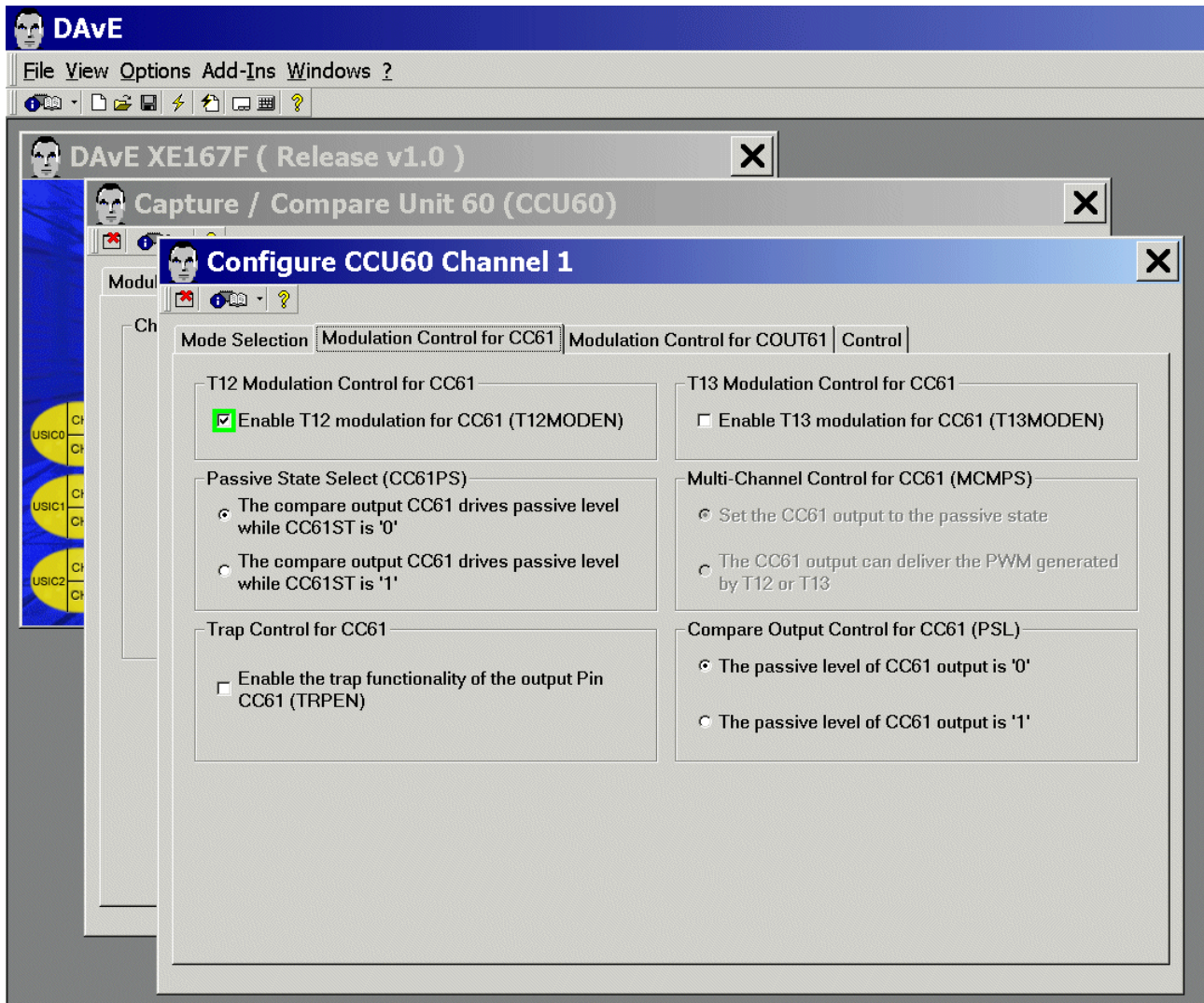




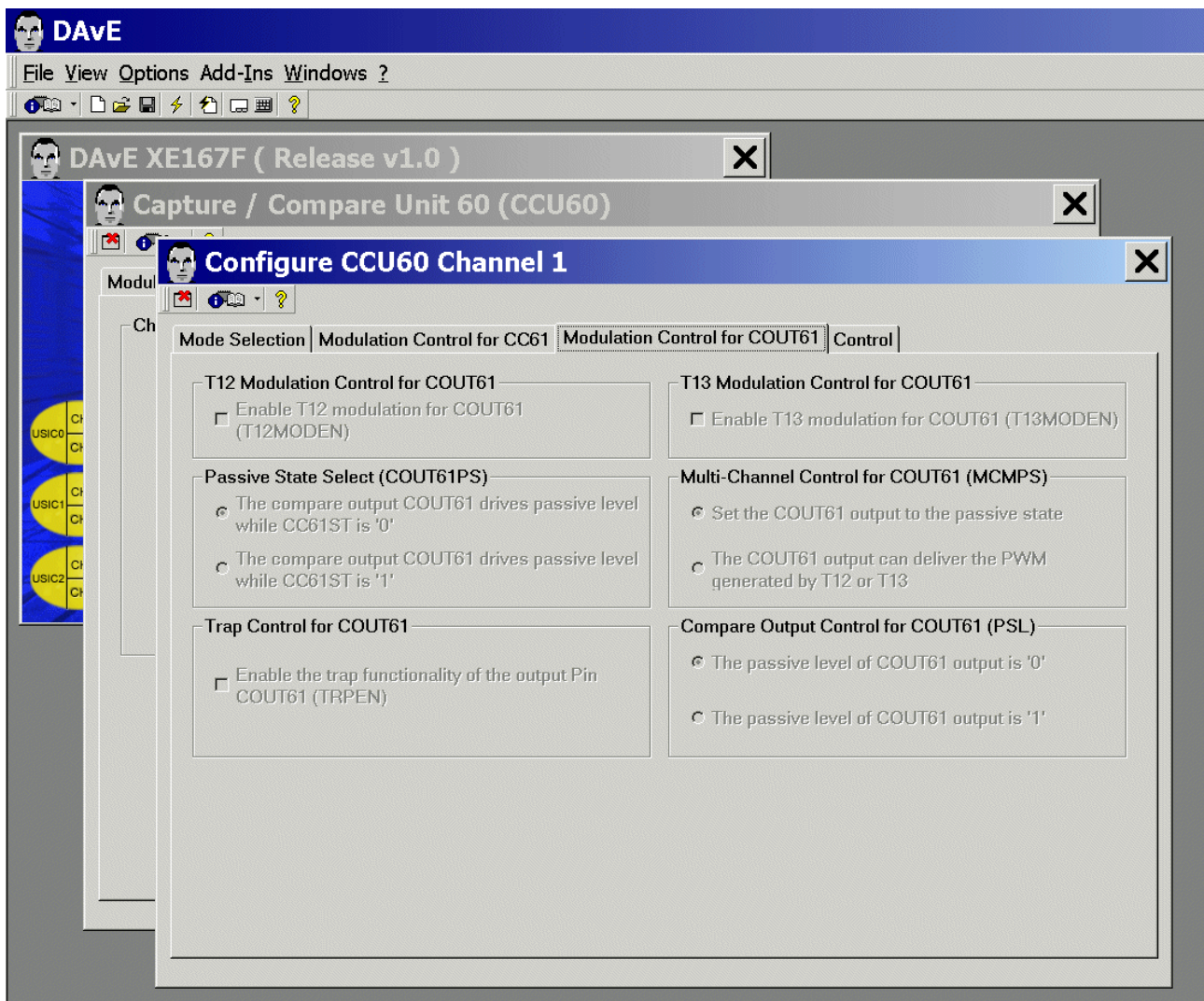
CCU60: Channels: Configure Channel 1:

Modulation Control for CC61:

T12 Modulation Control for CC61: click ☒ Enable T12 modulation for CC61

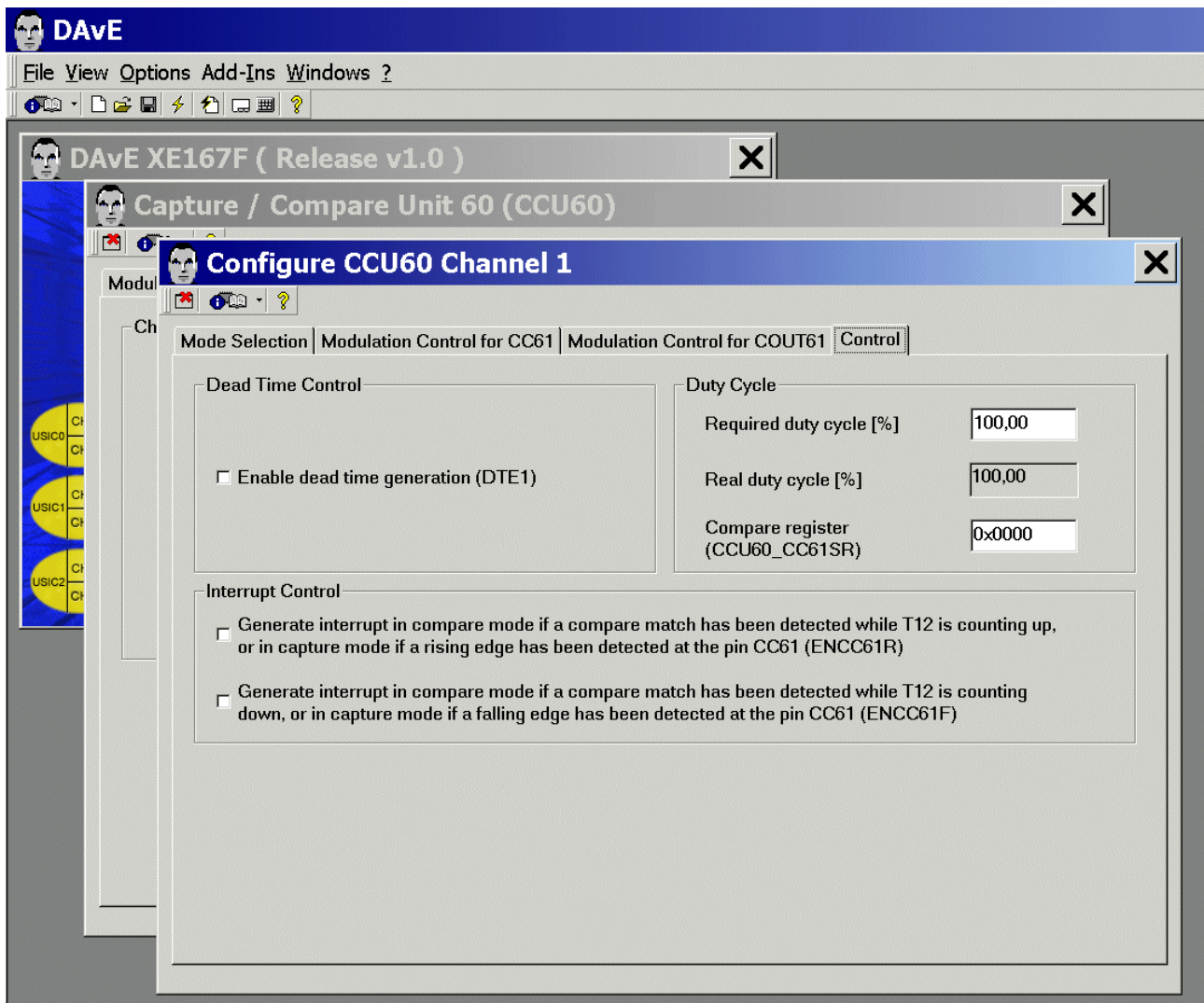



CCU60: Channels: Configure Channel 1: Modulation Control for COUT61: (do nothing)



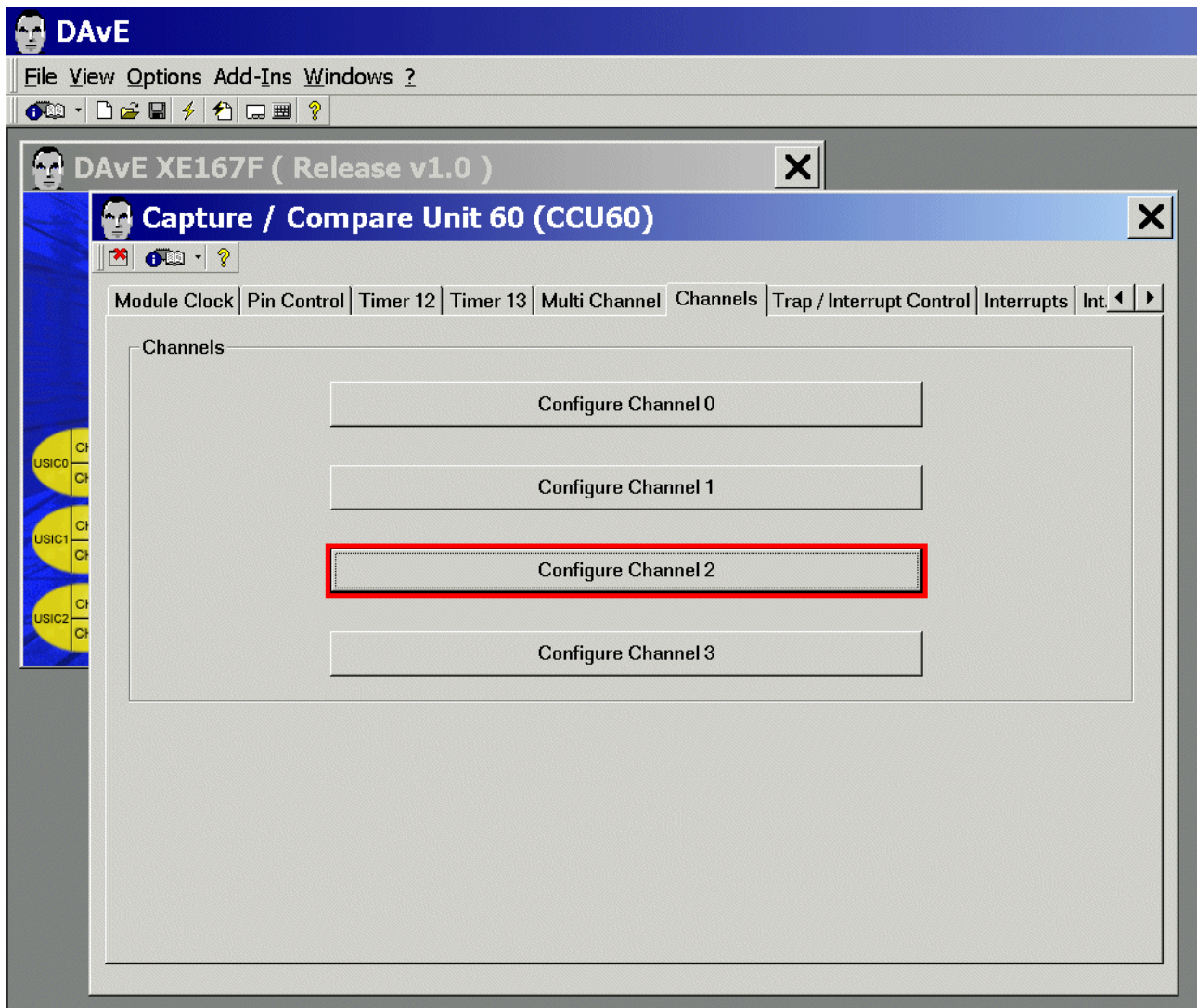


CCU60: Channels: Configure Channel 1: Control: (do nothing)



Exit and Save this dialog now by clicking  the close button.

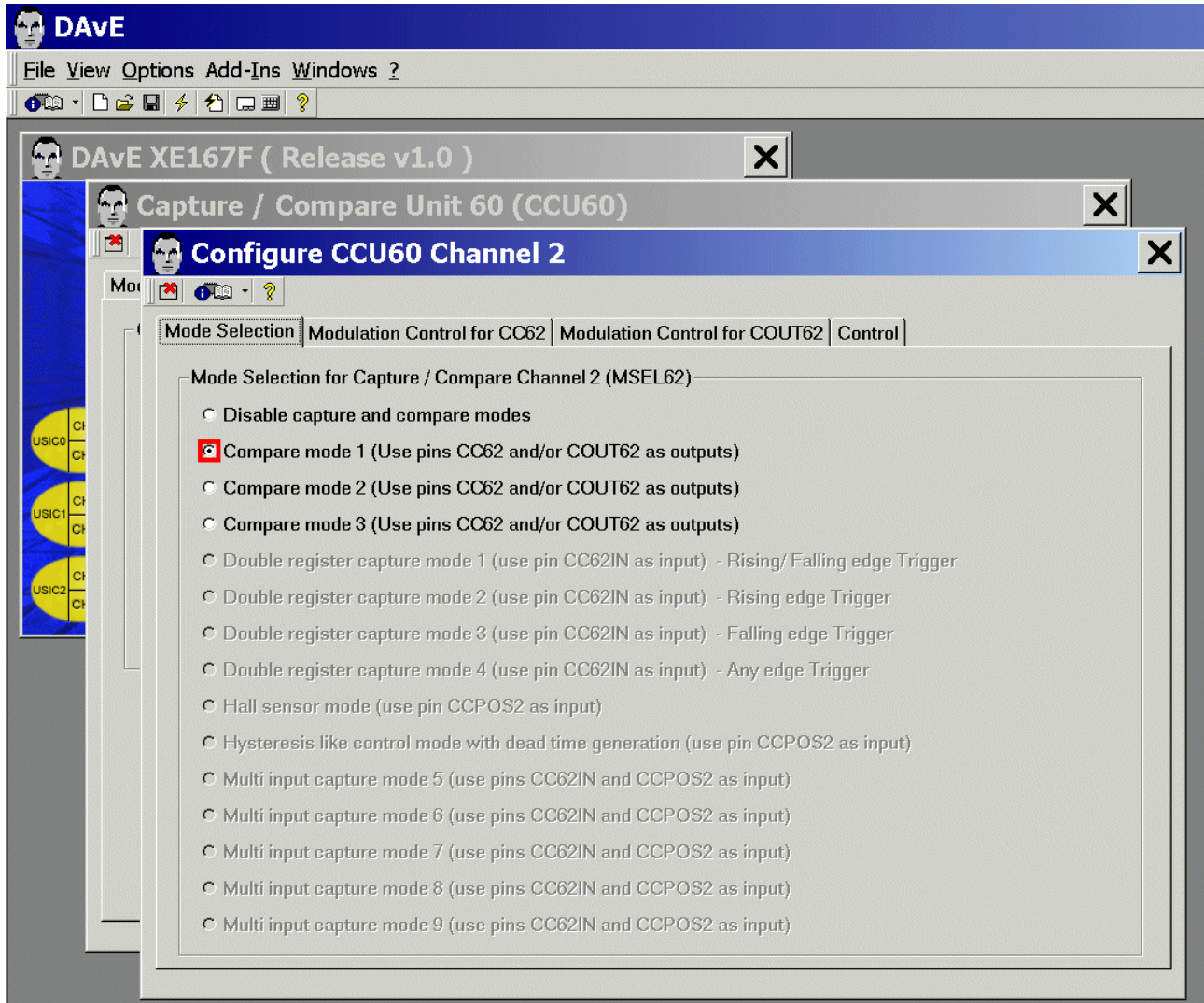
CCU60: Channels: **click** Configure Channel 2





CCU60: Channels: Configure Channel 2:

Mode Selection: Mode Selection for Capture / Compare Channel 2: click ☒ Compare mode 1





CCU60: Channels: Configure Channel 2:

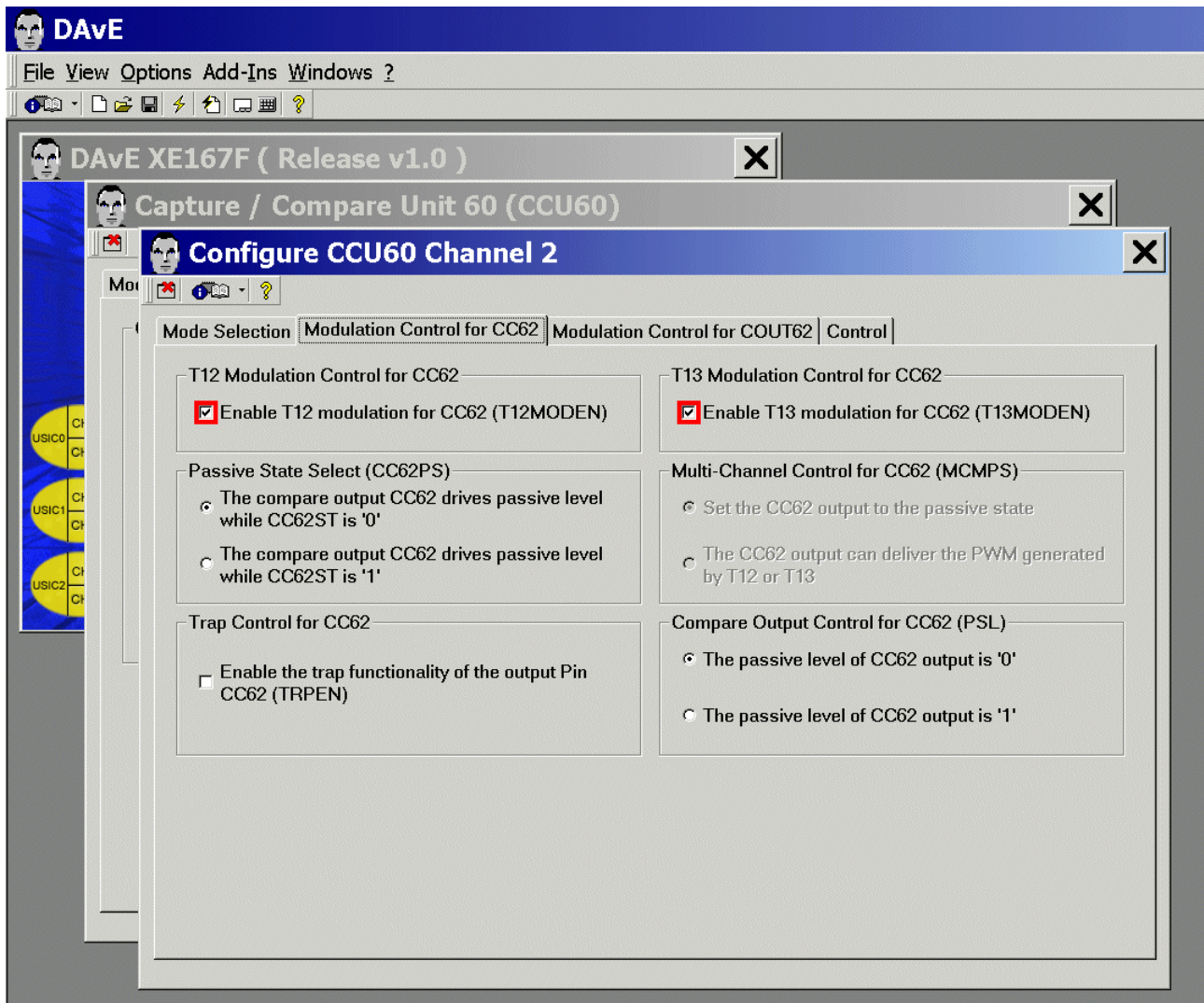
Modulation Control for CC62:

T12 Modulation Control for CC62: click ☒ Enable T12 modulation for CC62

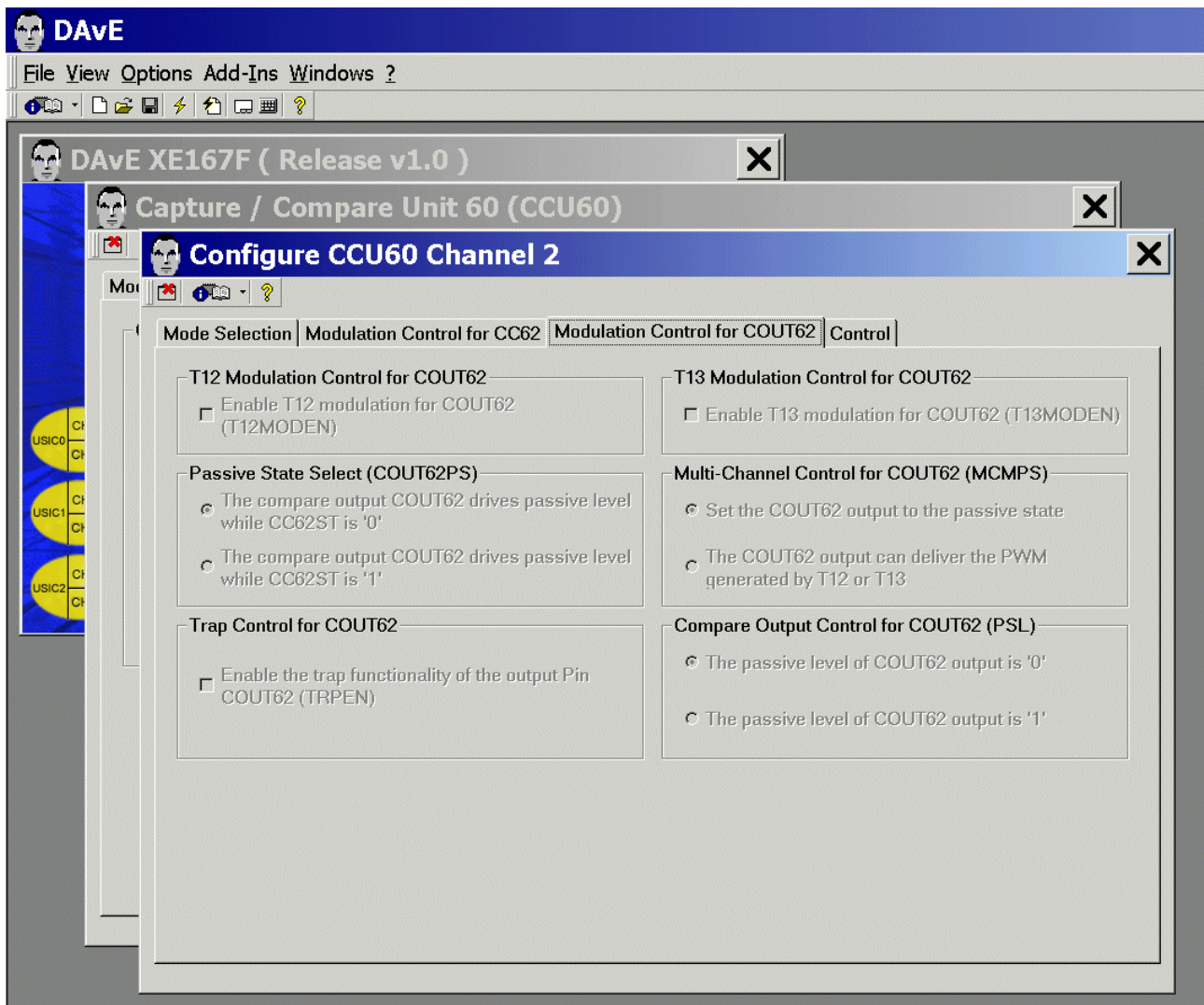
CCU60: Channels: Configure Channel 2:

Modulation Control for CC62:

T13 Modulation Control for CC62: click ☒ Enable T13 modulation for CC62

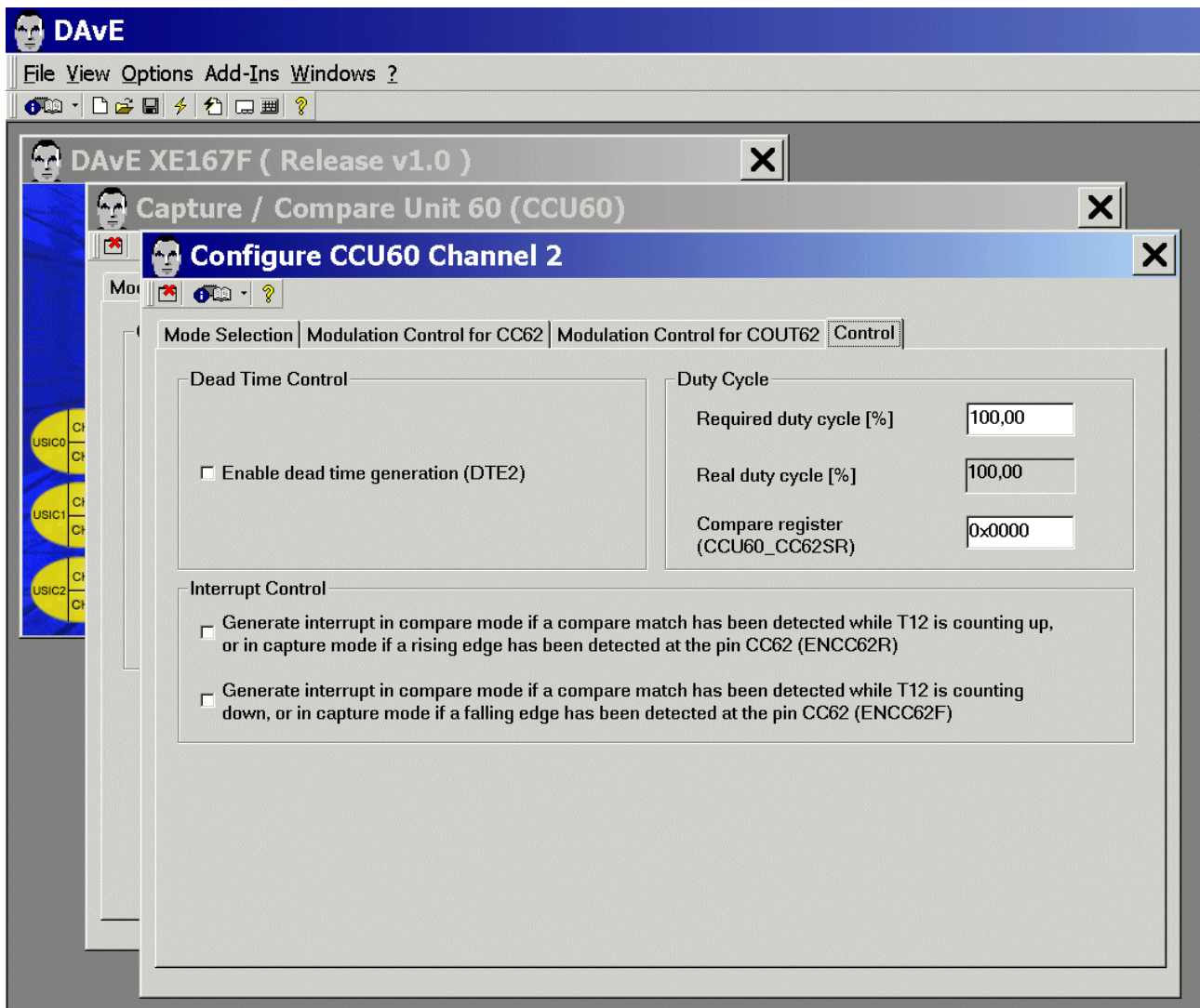



CCU60: Channels: Configure Channel 2: Modulation Control for COUT62: (do nothing)





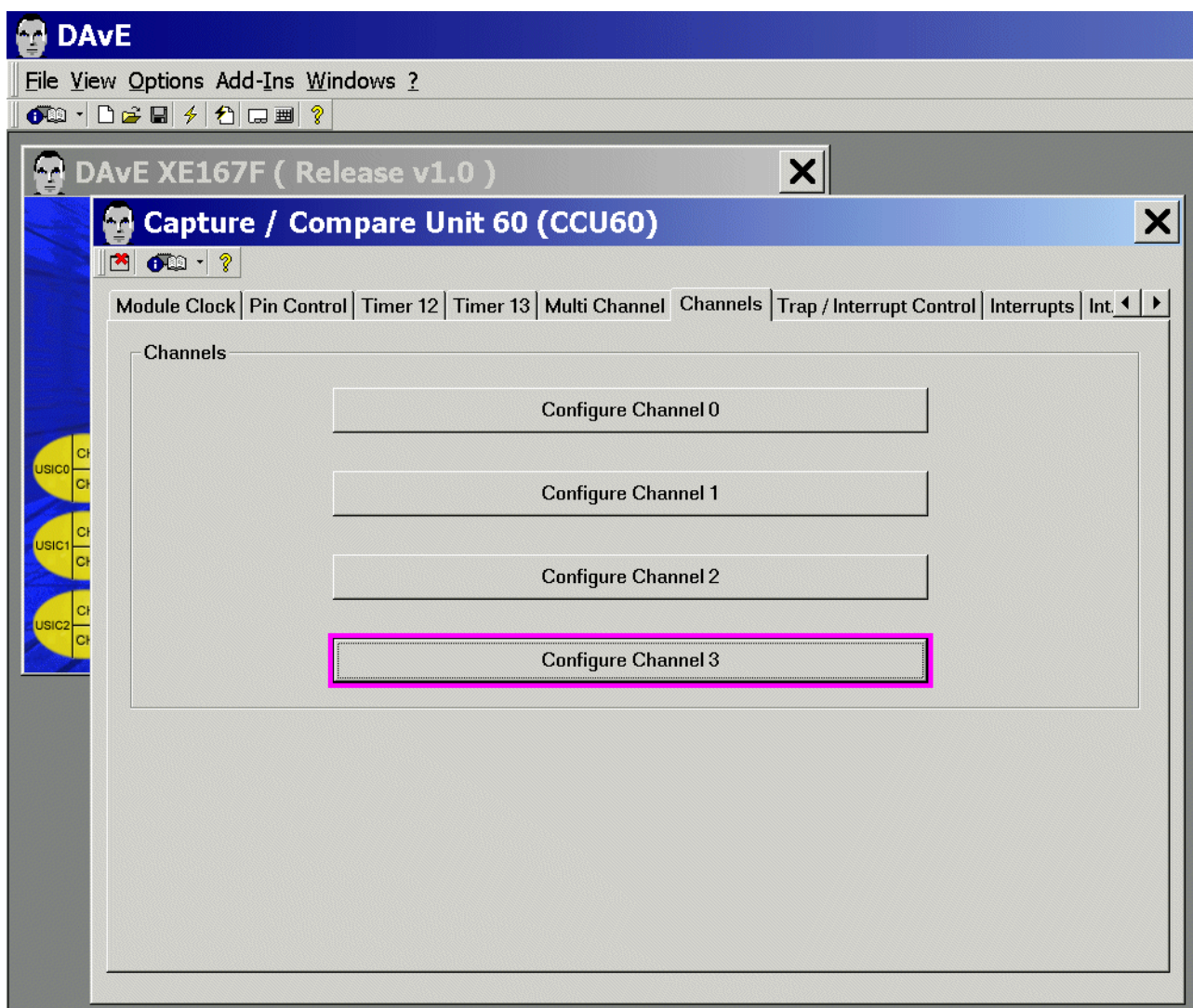
CCU60: Channels: Configure Channel 2: Control: (do nothing)



Exit and Save this dialog now by clicking  the close button.

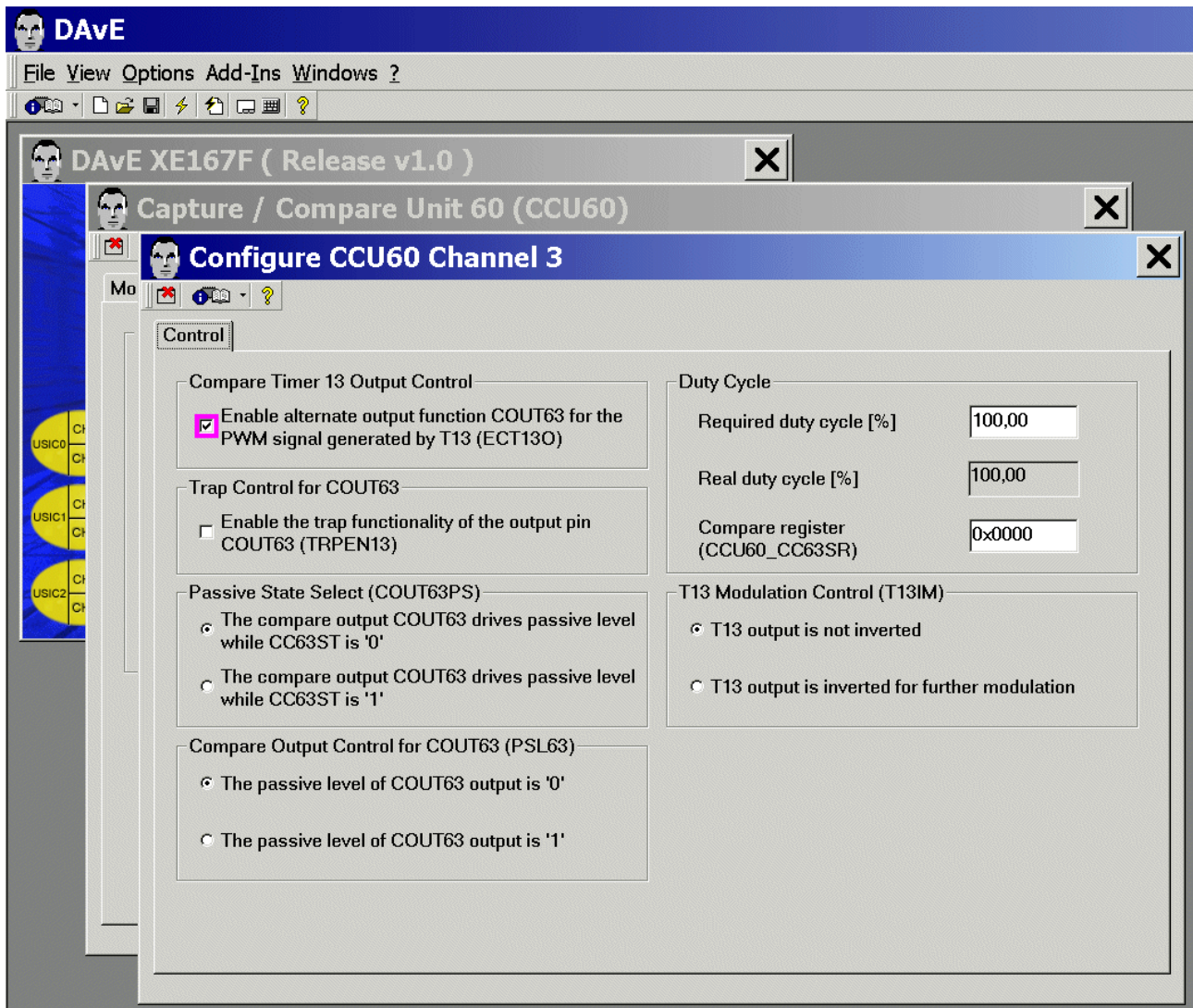



CCU60: Channels: **click** Configure Channel 3



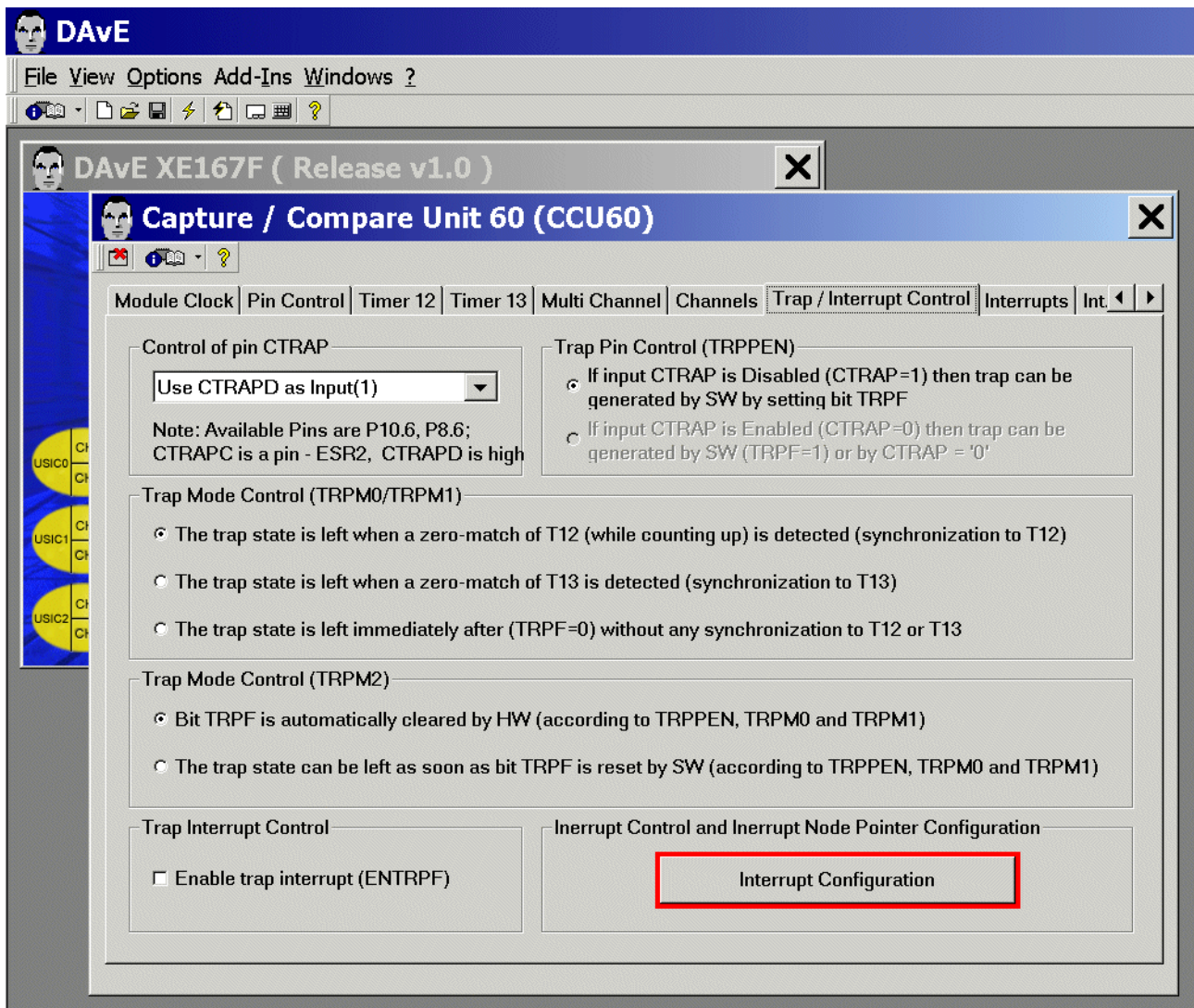
CCU60: Channels: Configure Channel 3:

Control: Compare Timer 13 Output Control: click ☒ Enable alternate output function COUT63



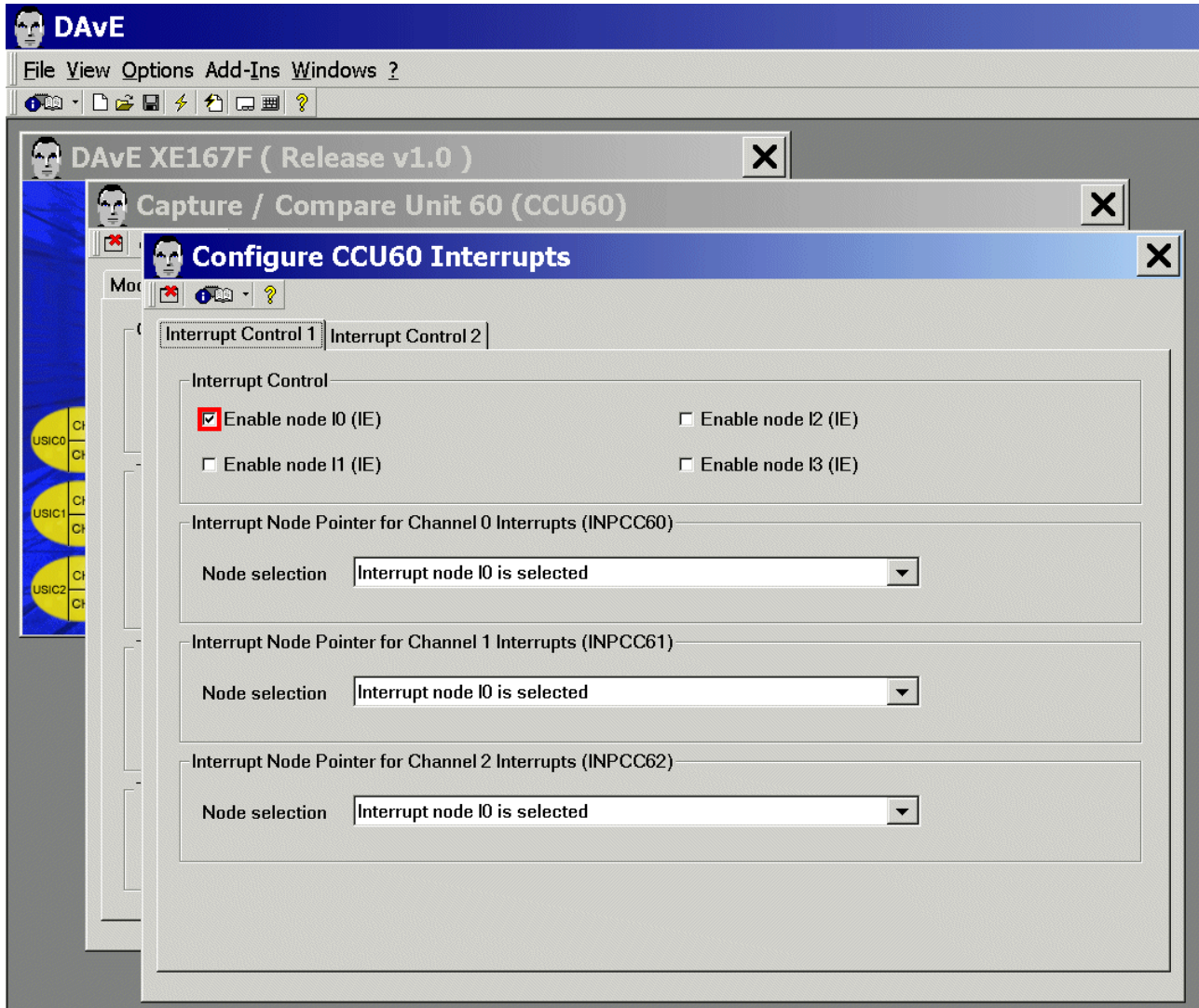
Exit and Save this dialog now by clicking  the close button.

CCU60: Trap / Interrupt Control: [click](#) Interrupt Configuration

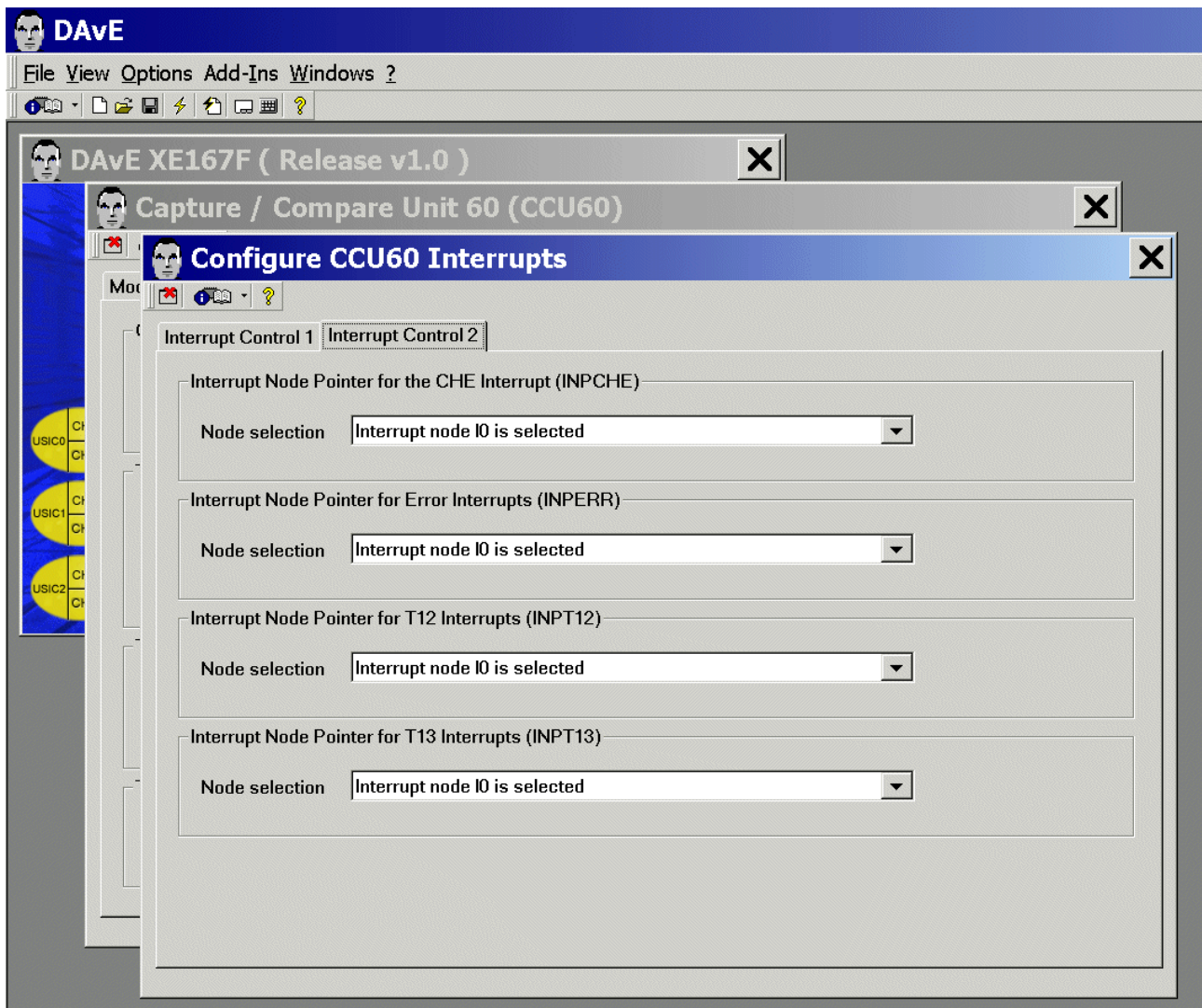





CCU60: Trap / Interrupt Control: Interrupt Configuration:  
Interrupt Control 1: Interrupt Control: click ☒ Enable node I0

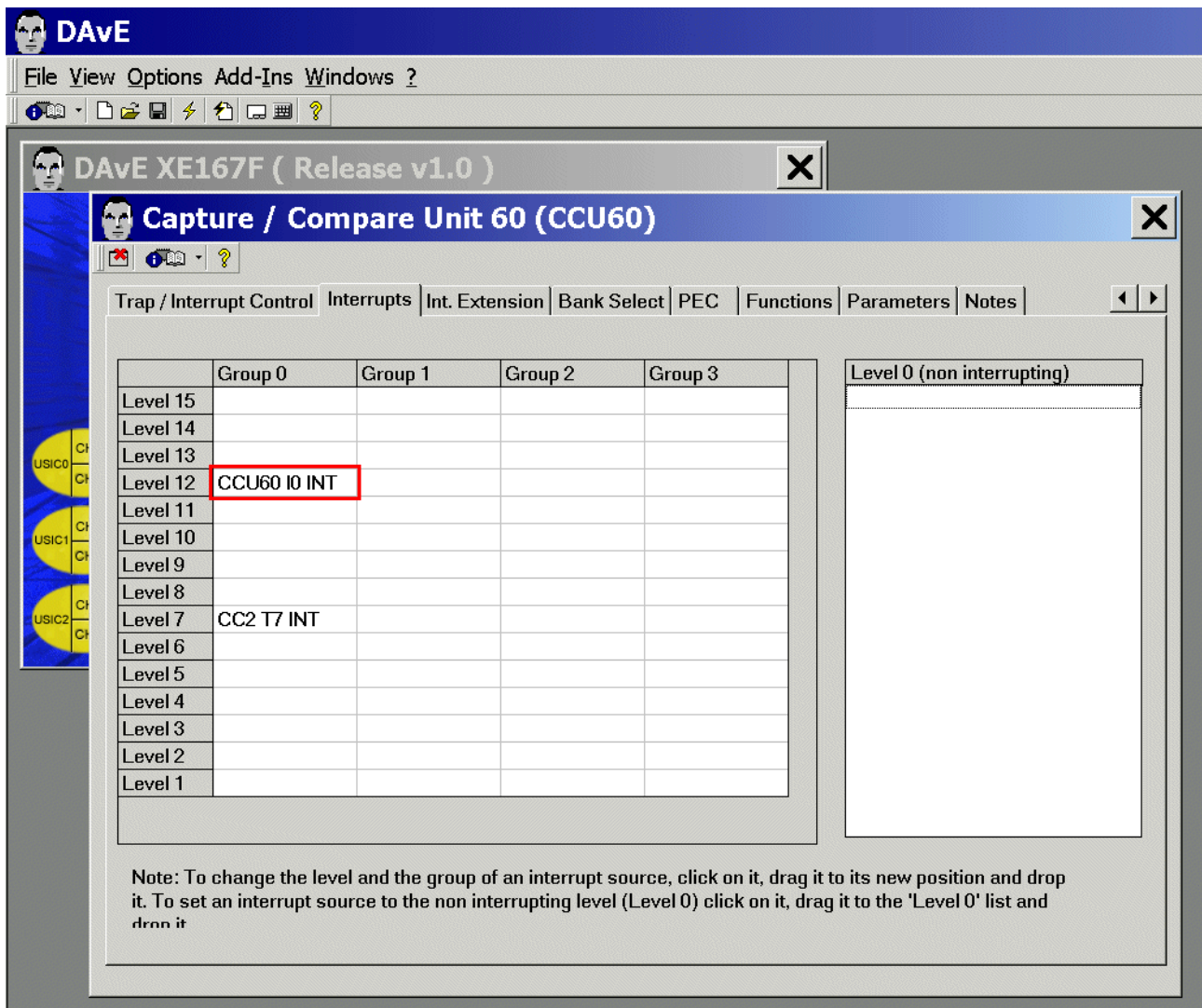


CCU60: Trap / Interrupt Control: Interrupt Configuration: Interrupt Control 2: (do nothing)



Exit and Save this dialog now by clicking  the close button.

CCU60: Interrupts: drag and drop the CCU60 I0 INT to Interrupt Level 12, Group 0



**DAVE**

File View Options Add-Ins Windows ?

DAVE XE167F ( Release v1.0 )

**Capture / Compare Unit 60 (CCU60)**

Trap / Interrupt Control Interrupts Int. Extension Bank Select PEC Functions Parameters Notes

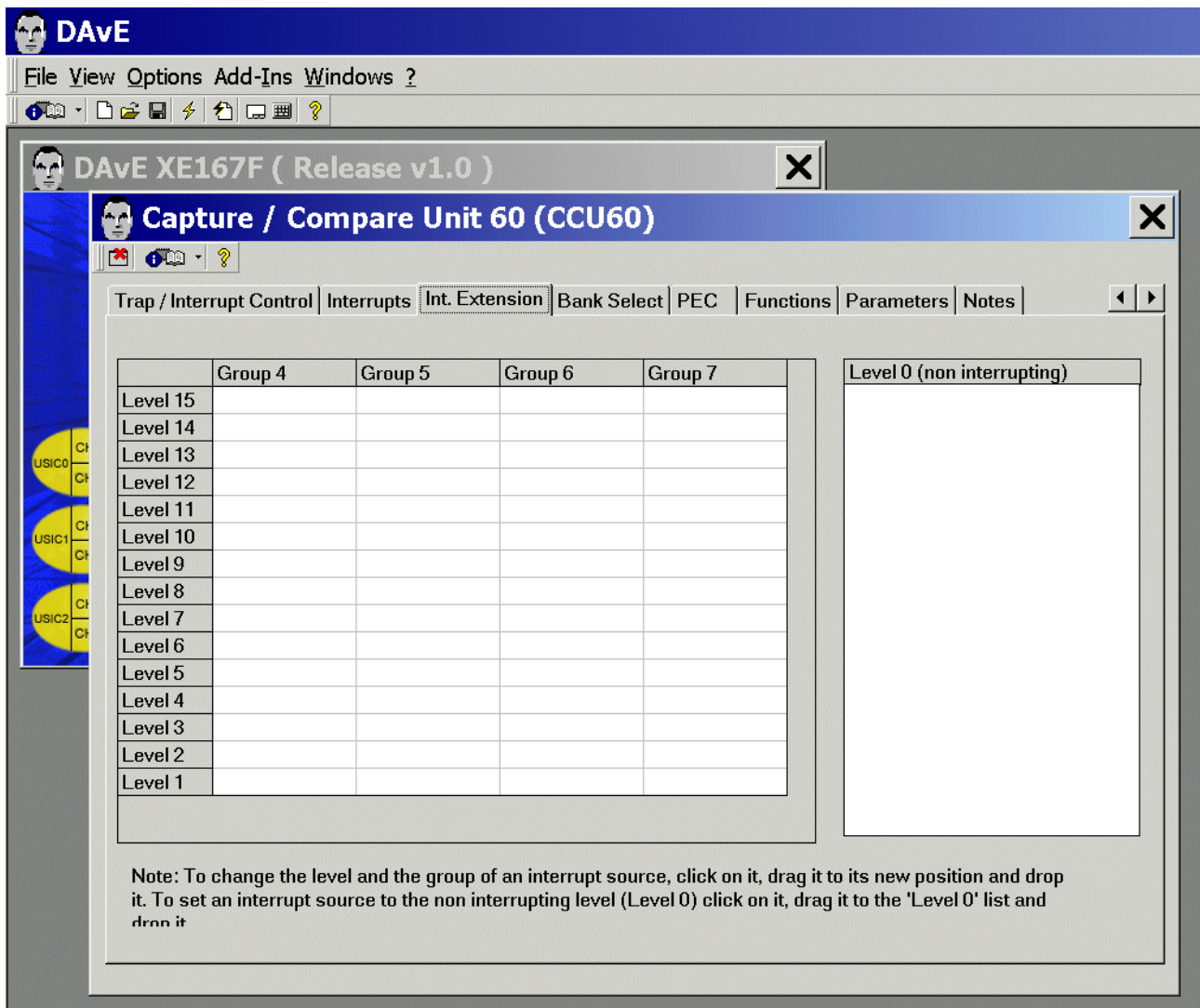
	Group 0	Group 1	Group 2	Group 3
Level 15				
Level 14				
Level 13				
Level 12	CCU60 I0 INT			
Level 11				
Level 10				
Level 9				
Level 8				
Level 7	CC2 T7 INT			
Level 6				
Level 5				
Level 4				
Level 3				
Level 2				
Level 1				

Level 0 (non interrupting)

Note: To change the level and the group of an interrupt source, click on it, drag it to its new position and drop it. To set an interrupt source to the non interrupting level (Level 0) click on it, drag it to the 'Level 0' list and drop it



CCU60: Int. Extension: (do nothing)



**DAVE**

File View Options Add-Ins Windows ?

DAVE XE167F ( Release v1.0 )

**Capture / Compare Unit 60 (CCU60)**

Trap / Interrupt Control | Interrupts | **Int. Extension** | Bank Select | PEC | Functions | Parameters | Notes

	Group 4	Group 5	Group 6	Group 7	Level 0 (non interrupting)
Level 15					
Level 14					
Level 13					
Level 12					
Level 11					
Level 10					
Level 9					
Level 8					
Level 7					
Level 6					
Level 5					
Level 4					
Level 3					
Level 2					
Level 1					

Note: To change the level and the group of an interrupt source, click on it, drag it to its new position and drop it. To set an interrupt source to the non interrupting level (Level 0) click on it, drag it to the 'Level 0' list and drop it

The screenshot shows the DAVE XE167F (Release v1.0) application window. The main menu bar includes File, View, Options, Add-Ins, Windows, and ?. Below the menu is a toolbar with icons for help, file operations, and execution. The central workspace displays the 'Capture / Compare Unit 60 (CCU60)' configuration panel.

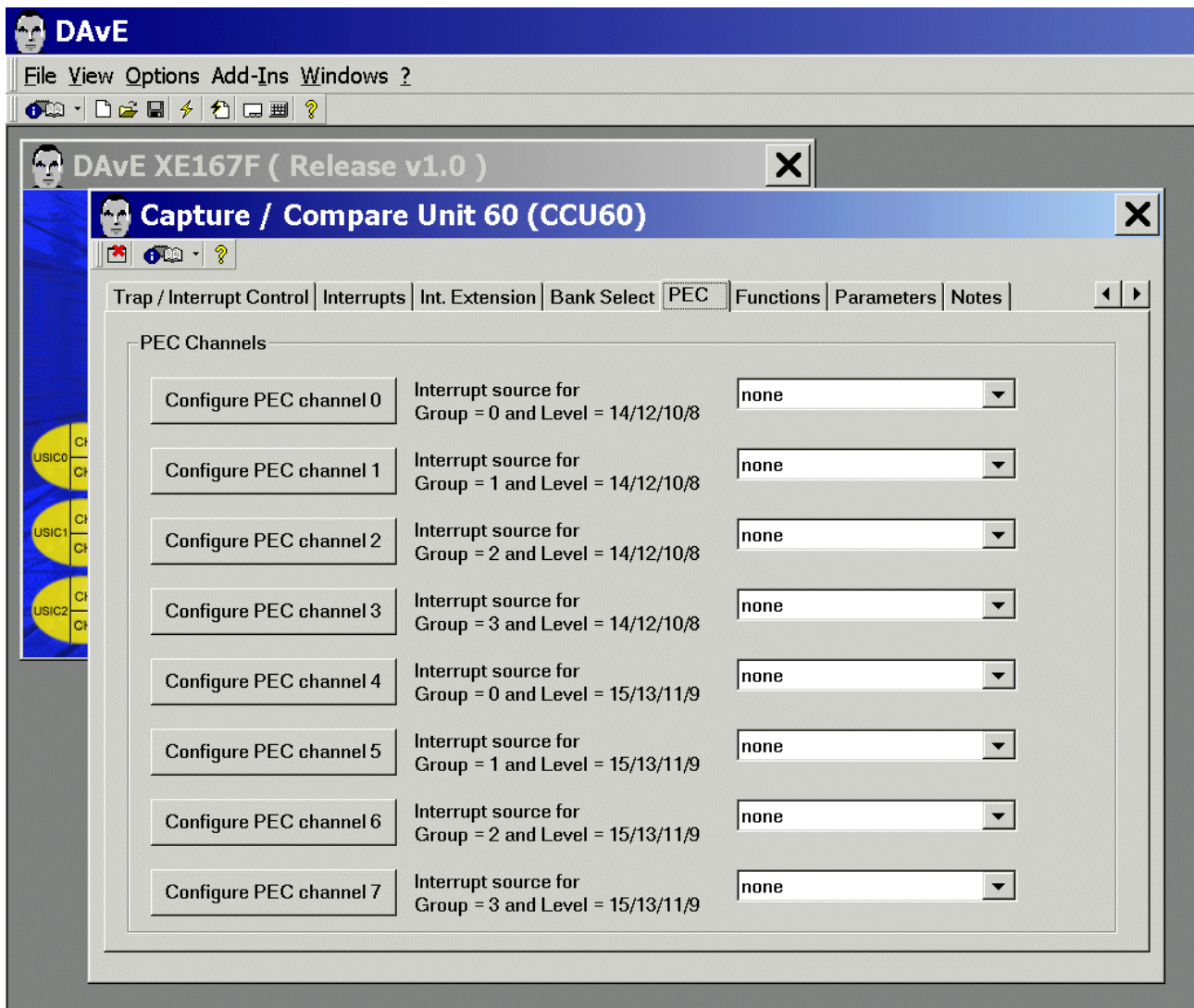
This panel has a title bar with a close button and a toolbar with icons for help, file operations, and execution. It features several tabs: Trap / Interrupt Control, Interrupts, Int. Extension, Bank Select, PEC, Functions, Parameters, and Notes. The 'Bank Select' tab is currently active.

Under the 'Bank Select' tab, there are three columns: Local Reg Bank 1, Local Reg Bank 2, and Fast Interrupts. To the right of these columns is a 'Global Register Bank' section containing 'CCU60 IO INT'. A vertical scrollbar is present next to the Fast Interrupts column.

At the bottom of the window, there are two notes:

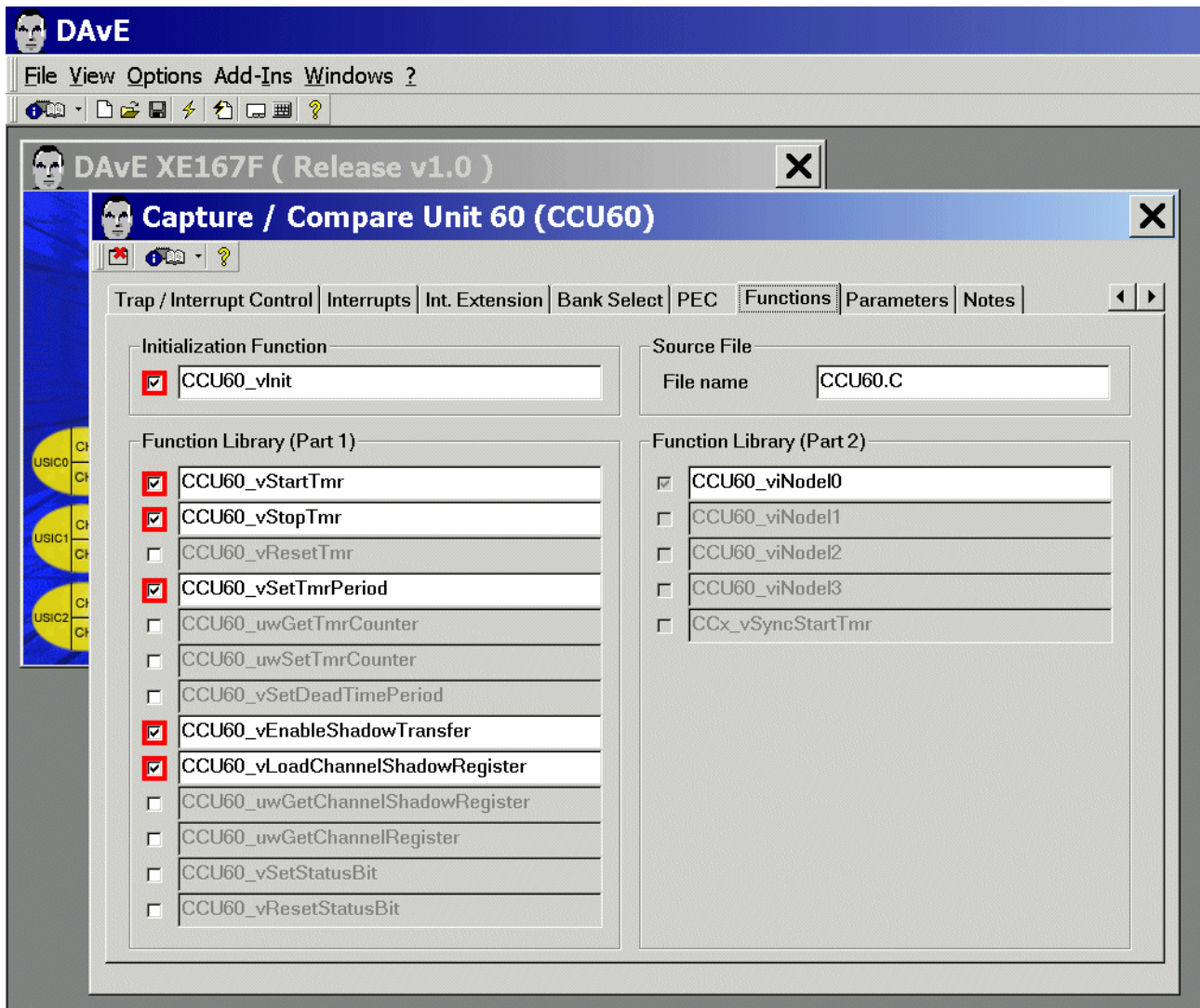
- Note 1: For all int's (with priority > = 12), Local Reg Banks (1 or 2) can be assigned by dragging and dropping to the desired bank.
- Note 2: Only 2 Fast Int's. can be assigned (with priority > = 12) at a time in the bottom 2 cells of Fast Interrupt

CCU60: PEC: (do nothing)

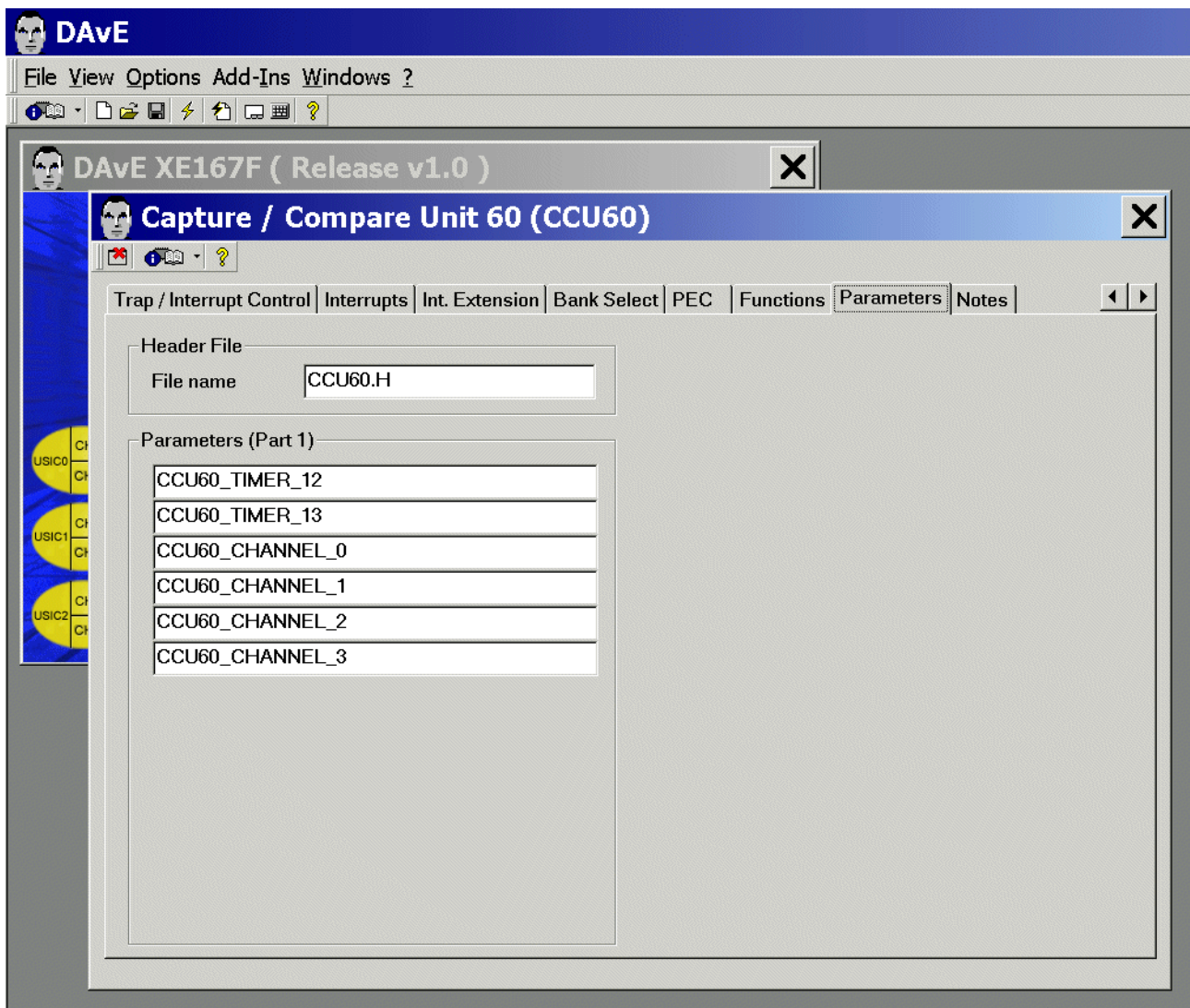





CCU60: Functions: Initialization Function: **click** ☒ CCU6\_vInit  
 CCU60: Functions: Function Library (Part 1): **click** ☒ CCU60\_vStartTmr  
 CCU60: Functions: Function Library (Part 1): **click** ☒ CCU60\_vStopTmr  
 CCU60: Functions: Function Library (Part 1): **click** ☒ CCU60\_vSetTmrPeriod  
 CCU60: Functions: Function Library (Part 1): **click** ☒ CCU60\_vEnableShadowTransfer  
 CCU60: Functions: Function Library (Part 1): **click** ☒ CCU60\_vLoadChannelShadowRegister




CCU60: Parameters: (do nothing)



CCU60: Notes: If you wish, you can insert your comments here.

Exit and Save this dialog now by clicking  the close button.

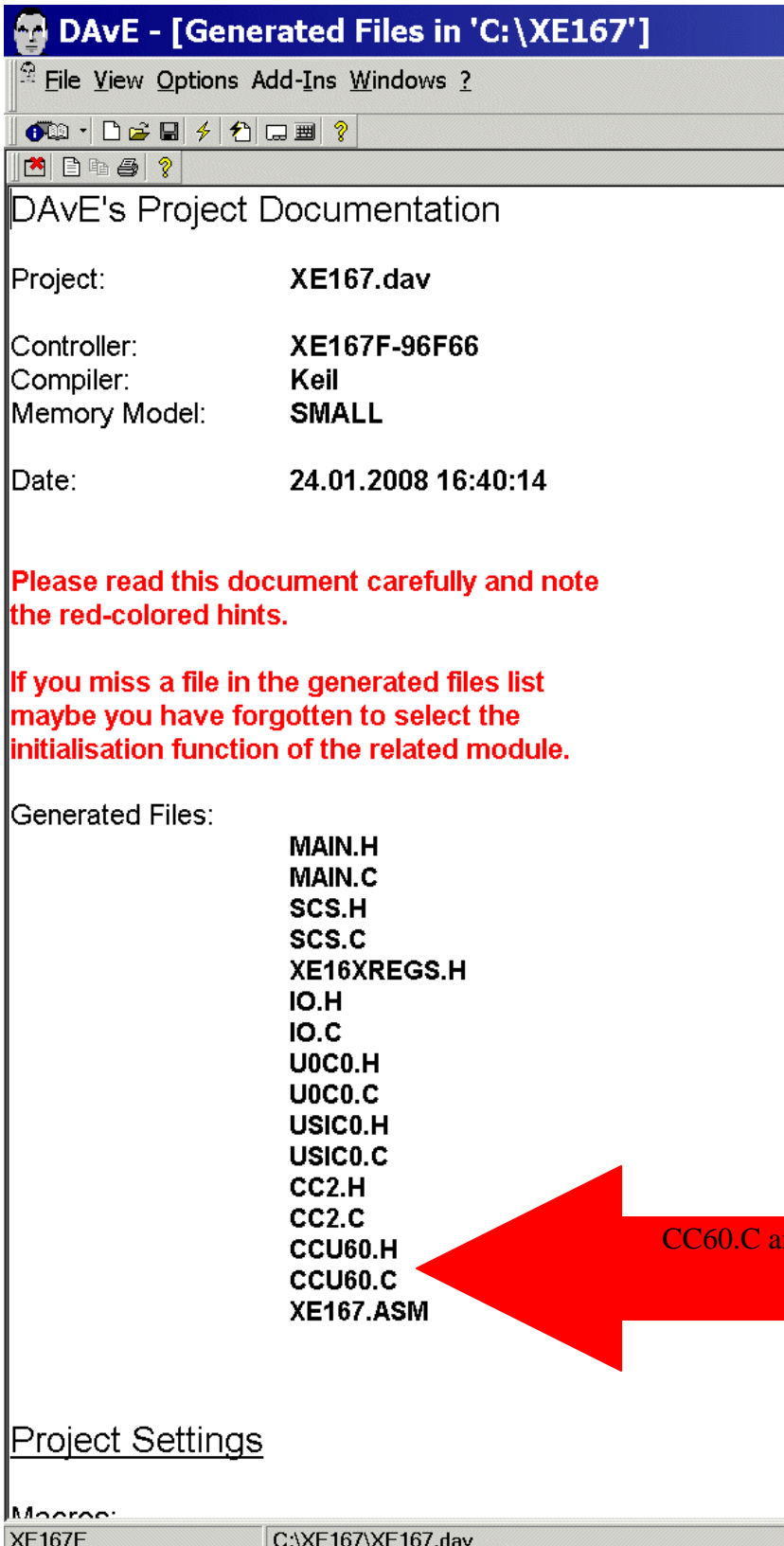
Generate Code:

<p>File Generate Code</p>	<p>or      click </p>
-------------------------------	---



DAvE will show you all the files he has generated  
(File Viewer opens automatically):





**DAvE - [Generated Files in 'C:\XE167']**

File View Options Add-Ins Windows ?

DAvE's Project Documentation

Project: XE167.dav

Controller: XE167F-96F66

Compiler: Keil

Memory Model: SMALL

Date: 24.01.2008 16:40:14

Please read this document carefully and note the red-colored hints.

If you miss a file in the generated files list maybe you have forgotten to select the initialisation function of the related module.

Generated Files:

- MAIN.H
- MAIN.C
- SCS.H
- SCS.C
- XE16XREGS.H
- IO.H
- IO.C
- U0C0.H
- U0C0.C
- USIC0.H
- USIC0.C
- CC2.H
- CC2.C
- CCU60.H
- CCU60.C
- XE167.ASM

Project Settings

Macro:

XE167F	C:\XE167\XE167.dav
--------	--------------------

CC60.C and CC60.H are new!

Close DAVe: **File – Exit** Save changes? **click Yes**



Start Keil  $\mu$ Vision and open your Keil Project

If you see an open project – close it: **Project - Close Project**

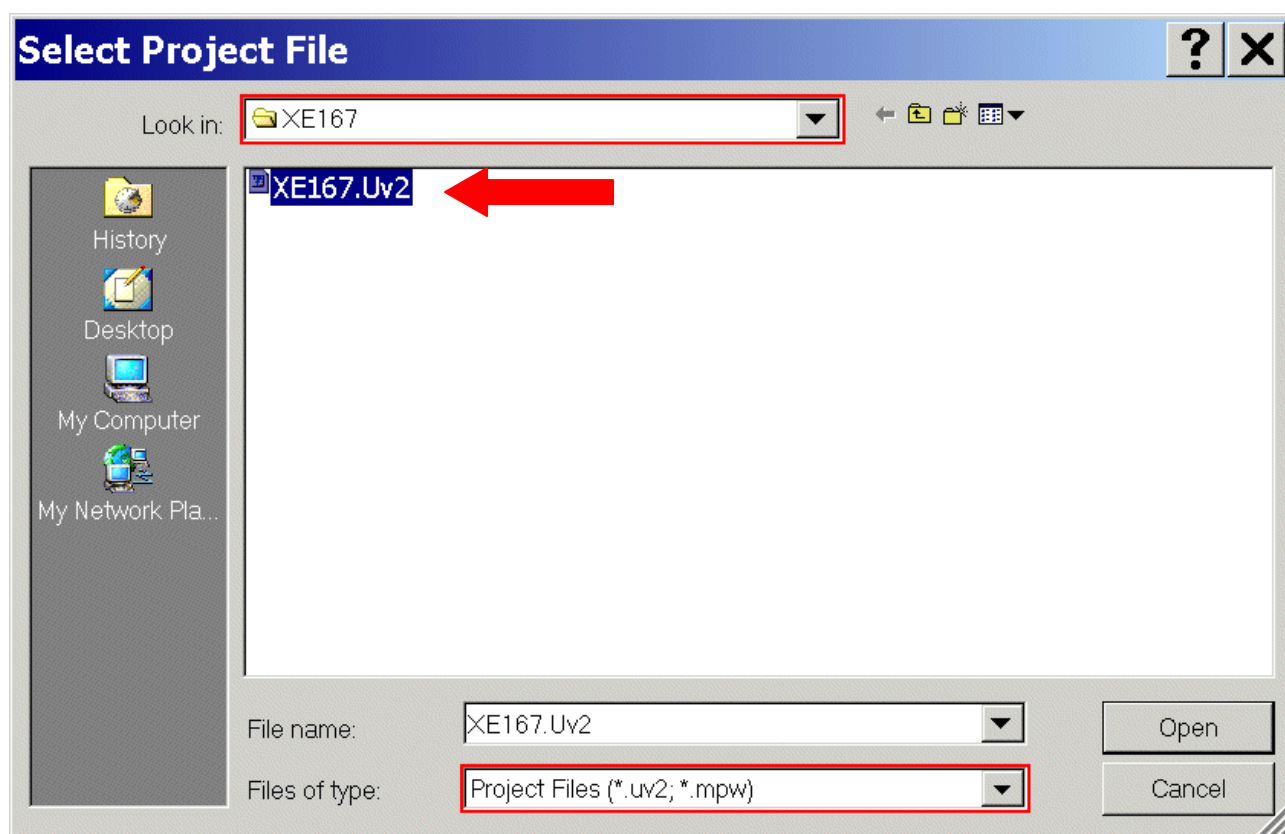
**Project - Open Project**

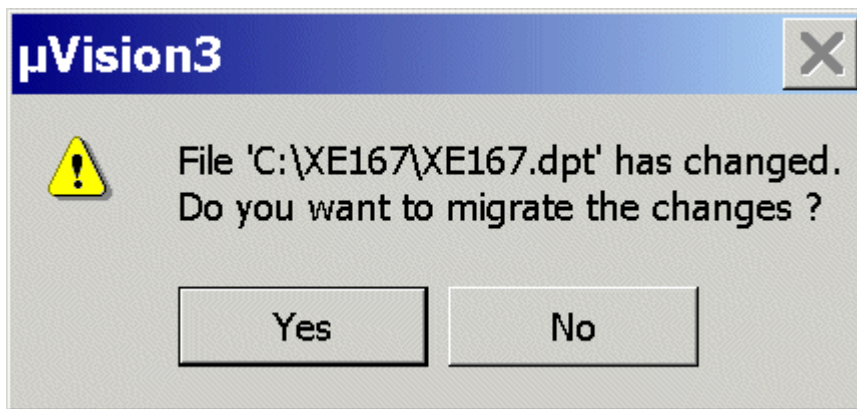
Select Project File: **Look in:** choose C:\XE167

Select Project File: **Files of type:** choose Project Files (\*.uv2)

Click XE167.Uv2

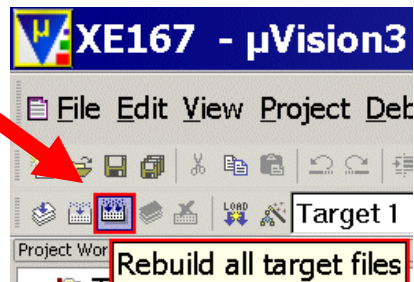
**Open**

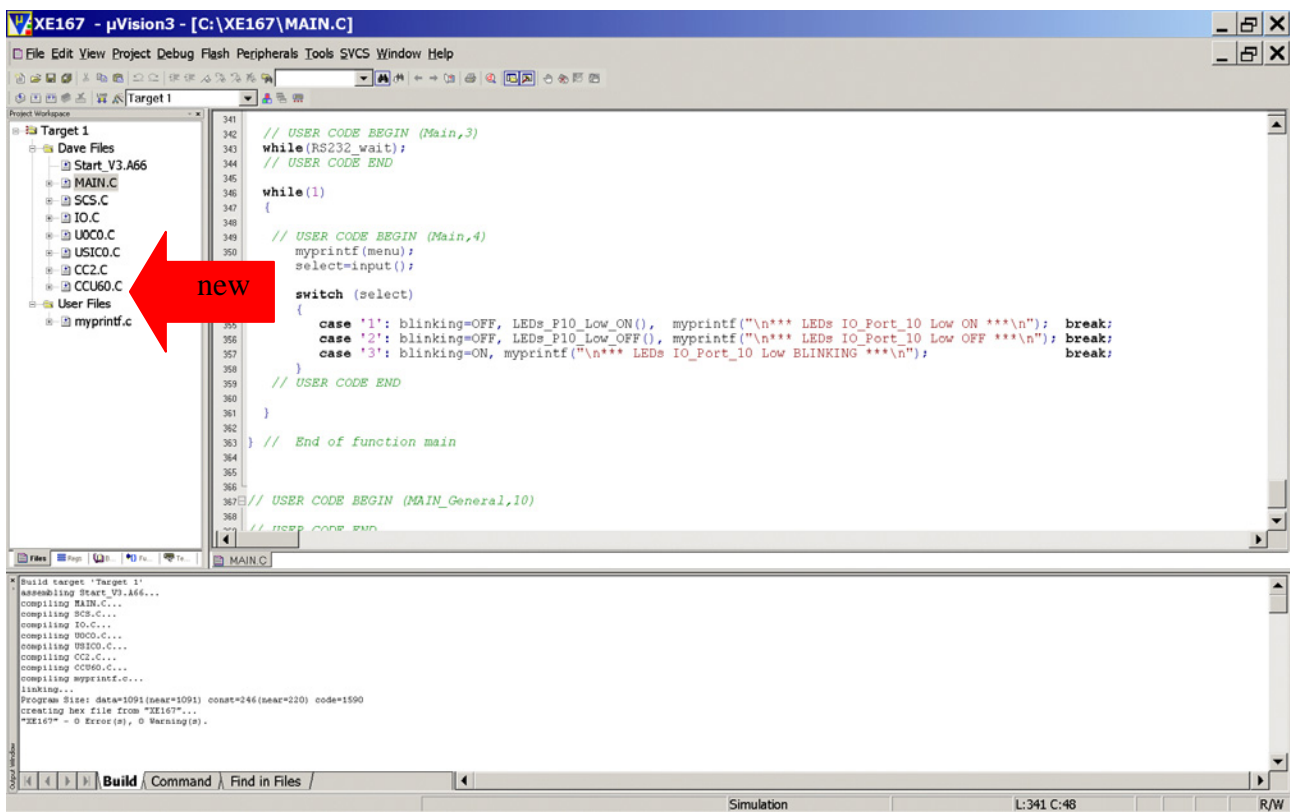




Click Yes

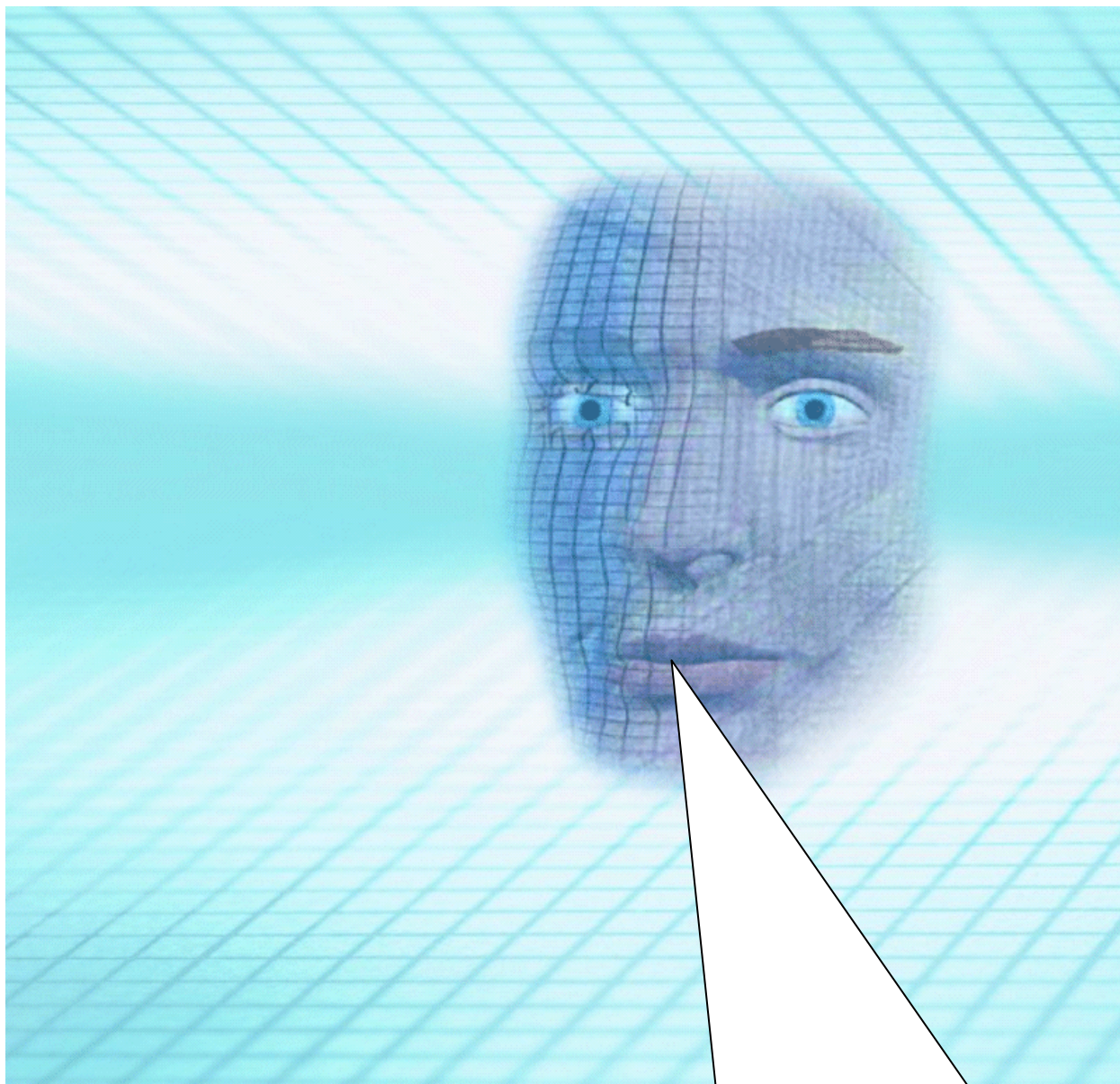


<p>Project – Rebuild all target files</p>	<p>or click</p>  <p>Target 1 Rebuild all target files</p>
---	---



The screenshot shows the XE167 - µVision3 IDE. The top menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar below the menu bar contains various icons for file operations and development. The 'Project Workspace' pane on the left shows a tree view of the project files, including 'Start\_V3.A66', 'MAIN.C', 'SCS.C', 'IO.C', 'U0C0.C', 'USIC0.C', 'CC2.C', 'CCU60.C', and 'User Files'. The 'User Files' folder is expanded, showing 'myprintf.c'. A red arrow points to 'myprintf.c' with the word 'new' written next to it. The main code editor displays the C code for 'myprintf.c', which includes a while loop for RS232 wait, a switch statement for menu selection, and a switch statement for LED control. The bottom status bar shows 'Build target "Target 1"', 'Simulation', 'L:341 C:48', and 'R/W'.

Insert/Change your application specific program:



**Note:**

DAvE doesn't change code which is inserted between '`// USER CODE BEGIN`' and '`// USER CODE END`'. Therefore, whenever adding code to DAvE's generated code, write it between '`// USER CODE BEGIN`' and '`// USER CODE END`'.

If you wish to change DAvE's generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAvE regenerate code!

Double click **MAIN.C** and **change** Global Variables

from:

```
const char menu[] =
"\n\n\n"
"1 ... LEDs IO_Port_10 Low ON\n"
"2 ... LEDs IO_Port_10 Low OFF\n"
"3 ... LEDs IO_Port_10 Low blinking\n"
"\n";

volatile int RS232_wait=2;
volatile bit blinking=ON;
char select=' ';
```

to:

```
const char menu[] =
"\n\n\n"
"a ... play: Maus am Mars\n"
"b ... play: Yesterday\n"
"c ... play: Frere Jacques / Lazy John / Bruder Jakob\n"
"d ... play: Happy birthday\n"
"e ... play: Take Me Home, Country Roads\n"
"f ... play: Es tanzt ein Bi-ba-butzemann\n"
"g ... play: Ich geh mit meiner Laterne\n"
"h ... play: The little drummer boy\n"
"i ... play: Hey, Pippi Langstrumpf\n"
"j ... play: Stille Nacht, heilige Nacht\n"
"k ... play: Junge komm bald wieder\n"
"l ... play: Lili Marleen\n"
"m ... play: musical scale / chromatic scale / for testing purpose\n"
"z ... back to main menu (anytime)\n\n";

volatile int RS232_wait=2;
char select=' ';

char mb1[500]; // message buffer for sprintf()
```



**XE167 - µVision3 - [C:\XE167\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

070
071
072 //*****
073 // @Global Variables
074 //*****
075
076 // USER CODE BEGIN (MAIN_General,7)
077 const char menu[] =
078 "\n\n\n"
079 "a ... play: Maus am Mars\n"
080 "b ... play: Yesterday\n"
081 "c ... play: Frere Jacques / Lazy John / Bruder Jakob\n"
082 "d ... play: Happy birthday\n"
083 "e ... play: Take Me Home, Country Roads\n"
084 "f ... play: Es tanzt ein Bi-ba-butzemann\n"
085 "g ... play: Ich geh mit meiner Laterne\n"
086 "h ... play: The little drummer boy\n"
087 "i ... play: Hey, Pippi Langstrumpf\n"
088 "j ... play: Stille Nacht, heilige Nacht\n"
089 "k ... play: Junge komm bald wieder\n"
090 "l ... play: Lili Marleen\n"
091 "m ... play: musical scale / chromatic scale / for testing purpose\n"
092 "z ... back to main menu (anytime)\n\n";
093
094 volatile int RS232_wait=2;
095 char select=' ';
096
097 char mb1[500]; // message buffer for sprintf()
098
099 // USER CODE END
100
101
102 //*****
103 // @External Prototypes
104 //*****
105
106 // USER CODE BEGIN (MAIN_General,8)
107
108 // USER CODE END
109
110
111 //*****

```

Files Regs B... Fu... Te... MAIN.C

Simulation

Double click **MAIN.C** and insert Global Variables:

```
// *** Music ***:
/*
Construction of the music data:
=====
created by Christan Perschl (www.perschl.at)
extended by Wilhelm Brezovits

C,D,E,F,G,A,H: play note
+: the + raises its note a semitone: Cis, Dis, Eis, Fis, Gis, Ais, His
-: the - lowers its note a semitone: Ces, Des, Es, Fes, Ges, As, Hes
Lx : Change note length
    (x = 1,2,4,8,16 -> 1=whole-note, 2=half-note, 4=quarter-note, ...)
Px : play rest
    (x = 1,2,4,8,16 -> 1=whole-rest, 2=half-rest, 4=quarter-rest, ...)
Ox : Change octave (x = 0,1,2,3)
. : Extend preceding note by half of its value
Tx : Change tempo (x = 50 ... 199 beats per minute)

Additional functionality:
=====
OL : activate octave LOW
ON : deactivate octave LOW = activate octave normal (O0,O1,O2,O3)
*/

unsigned int T13_values[] = {62977, 59550, 56122, 53061, 50000, 47278, 44685, 42092, 39796,
37500, 35450, 33401, 275};
// Timer-T13-periods(frequencies) of the notes
// [0]=c',[1]=cis',[2]=d',[3]=dis',[4]=e',[5]=f',
// [6]=fis',[7]=g',[8]=gis',[9]=a',[10]=ais',[11]=h',
// [12]=<Frequency for rest>

unsigned int length_of_a_whole_note = 16113;
// Default length of a whole-note with tempo 120
```

XE167 - µVision3 - [C:\XE167\MAIN.C\*]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

Target 1

Dave Files

Start\_V3.A66

MAIN.C

SCS.C

IO.C

UOC0.C

USIC0.C

CC2.C

CCU60.C

User Files

myprintf.c

```

093 volatile int RS232_wait=2;
094 char select=' ';
095
096 char mbl[500]; // message buffer for sprintf()
097
098 // *** Music ***:
099 /*
100  Construction of the music data:
101  =====
102  created by Christian Perschl (www.perschl.at)
103  extended by Wilhelm Brezovits
104
105  C,D,E,F,G,A,H: play note
106  +: the + raises its note a semitone: Cis, Dis, Eis, Fis, Gis, Ais, His
107  -: the - lowers its note a semitone: Ces, Des, Es, Fes, Ges, As, Hes
108  Lx : Change note length
109      (x = 1,2,4,8,16 -> 1=whole-note, 2=half-note, 4=quarter-note, ...)
110  Px : play rest
111      (x = 1,2,4,8,16 -> 1=whole-rest, 2=half-rest, 4=quarter-rest, ...)
112  Ox : Change octave (x = 0,1,2,3)
113  . : Extend preceding note by half of its value
114  Tx : Change tempo (x = 50 ... 199 beats per minute)
115
116  Additional functionality:
117  =====
118  OL : activate octave LOW
119  ON : deactivate octave LOW = activate octave normal (00,01,02,03)
120  */
121
122 unsigned int Tl3_values[] = {62977, 59550, 56122, 53061, 50000, 47278, 44685, 42092, 39796, 37500, 35450, 33401, 275};
123 // Timer-Tl3-periods(frequencies) of the notes
124 // [0]=c,[1]=cis,[2]=d,[3]=dis,[4]=e,[5]=f',
125 // [6]=fis,[7]=g,[8]=gis,[9]=a,[10]=ais,[11]=h',
126 // [12]=<Frequency for rest>
127
128 unsigned int length_of_a_whole_note = 16113;
129 // Default length of a whole-note with tempo 120
130
131 // USER CODE END
132
133
134

```

MAIN.C

Simulation L:122 C:127



Double click MAIN.C and insert Global Variables:

```
// *** Songs ***:
```

```
// Maus am Mars (song a) :
```

```
const char
songa[]="T120O0L4FL8AL4O1C.O0L8FEGL2O1CO0P4P8L4EL8GO1L4C.O0L8EFAL2O1CP4
P8O0L4FL8AO1L4C.O0L8FH-O1L4DFL8FEDDCO0HO1CDCO0H-GL2F.";
```

```
// Yesterday (song b) :
```

```
const char songb[]="T120O0L8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-
AGL4H-L8AL4A.L4GFL8AL2GL8DL4FL8AL2AAAL4O1DEFL8EDL4E.L8DL4CEFCO0H-
AL8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2A";
```

```
// Bruder Jakob (song c) :
```

```
const char songc[]="T120O0L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0L8H-
L4AFO1L8CDCO0L8H-L4AFFCL2FL4FCL2F";
```

```
// Happy birthday (song d) :
```

```
const char
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";
```

```
// Take Me Home, Country Roads (song e):
```

```
const char
songe[]="T199O0L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EEEDL8EL4GL1GP
1L4DDE.L2D.L4EGGHL1HL4AAAAH.L2A.L4EGGAL2G.L4GAL1HL8HAL4GL1AL4HAL1G
L4HO1L4DL1EL4EEDO0L1HL8HAGAL1HL8HAL4GL1GL4GAL1G";
```

```
// Es tanzt ein Bi-ba-butzemann (song f):
```

```
const char
songf[]="T199O0L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";
```

```
// Ich geh mit meiner Laterne (song g):
```

```
const char
songg[]="T120O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8CL4FL8FA
FAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4F
L8FGGGGAGL4FP4.O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4FL8FGGGGAGL4FP4.";
```

```
// The little drummer boy (song h):
```

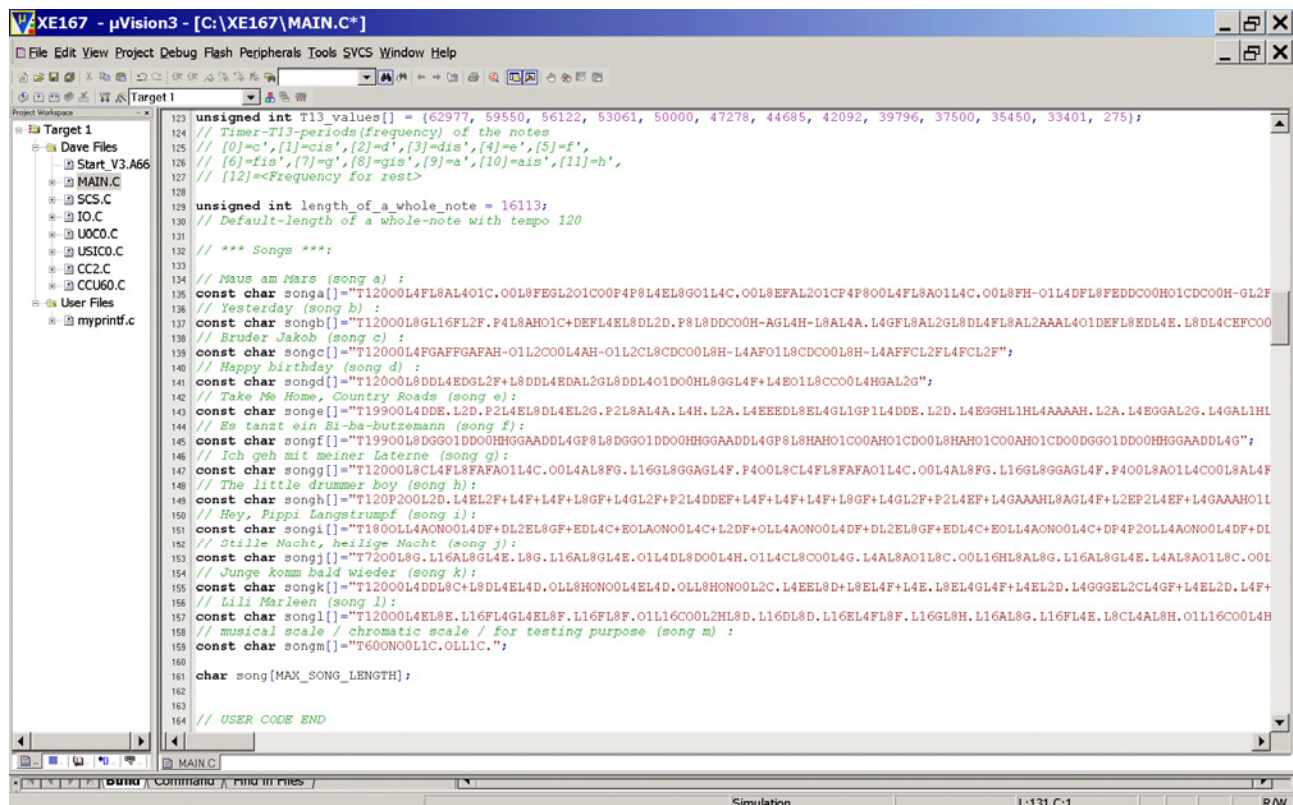
```
const char
songh[]="T120P2O0L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+L4F+L4F+L8G
F+L4GL2F+P2L4EF+L4GAAHL8AGL4F+L2EP2L4EF+L4GAAHO1L8CO0L8HL4AL2GL8
HAL4GL2F+L8AGL4F+L2EP1L2D.L4EL4F+F+F+F+L8GF+L4GL2F+P1L8EDL4EL2D";
```

```
// Hey, Pippi Langstrumpf (song i):
```

```
const char
songi[]="T180OLL4AONOO0L4DF+DL2EL8GF+EDL4C+EOLAONOO0L4C+L2DF+OLL4AONO
O0L4DF+DL2EL8GF+EDL4C+EOLL4AONOO0L4C+DP4P2OLL4AONOO0L4DF+DL2EL8GF+ED
L4C+EOLAONOO0L4C+L2DF+OLL4AONOO0L4DF+DL2EL8GF+EDL4C+EOLL4AONOO0L4C+
DP4P2O0L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+
EEDC+DP4L2F+GAH.O1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+GL
4AF+GAL2H.O1L4DC+O0L4HAGL2A.O1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+EDP2";
```

```
// Stille Nacht, heilige Nacht (song j):
const char
songj[]="T72O0L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8DO0L4H.O1L4CL8CO0L4G.L
4AL8AO1L8C.O0L16HL8AL8G.L16AL8GL4E.L4AL8AO1L8C.O0L16HL8AL8G.L16AL8GL4
E.O1L4DL8DL8F.L16DO0L8HO1L4C.L4E.L8C.O0L16GL8EL8G.L16FL8DL1C.";
// Junge komm bald wieder (song k):
const char
songk[]="T120O0L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4EEL8D+
L8EL4F+L4E.L8EL4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+.L8EL4EL2DL4E
L4D.L8COLL2H.ONO0L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4E
EL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8DDDDDDDDDL4DP8L8DL8D+L8DDDDDDL8D
+L8DL4DP8L8DL8EEEEEEEL2GP8L8EL1DP8L8DL8EEEL4E.P8L8GGGF+L8GL1A.";
// Lili Marleen (song l):
const char
songl[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL
8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL
4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HONO0L2C.";
// musical scale / chromatic scale / for testing purpose (song m) :
const char songm[]="T60ONO0L1C.OLL1C.";

char song[MAX_SONG_LENGTH];
```



```
123 unsigned int T13_values[] = {62977, 59550, 56122, 53061, 50000, 47278, 44685, 42092, 39796, 37500, 35450, 33401, 275};
124 // T13-periods (frequency) of the notes
125 // [0]=c', [1]=cis', [2]=d', [3]=dis', [4]=e', [5]=f',
126 // [6]=fis', [7]=g', [8]=gis', [9]=a', [10]=ais', [11]=h',
127 // [12]=<Frequency for rest>
128
129 unsigned int length_of_a_whole_note = 16113;
130 // Default-length of a whole-note with tempo 120
131
132 // *** Songs ***
133
134 // Maus am Mars (song a) :
135 const char songa[]="T120O0L4FL8AL4O1C.O0L8FEG2L2O1CO0P4P8L4EL8G0L14C.O0L8EFAL2O1CF4P8O0L4FL8A0L14C.O0L8FH-O1L4DFL8FEDDCO0H01CDCO0H-GL2F
136 // Yesterday (song b) :
137 const char songb[]="T120O0L8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-L8AL4A.L4GFL8AL2GL8DL4FL8AL2AAAL4O1DEFL8EDL4E.L8DL4CEFCO0
138 // Bruder Jakob (song c) :
139 const char songc[]="T120O0L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0L8H-L4AFO1L8CDCO0L8H-L4AFFCL2FL4FCL2F";
140 // Happy birthday (song d) :
141 const char songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGGL4F+L4EO1L8CCO0L4HGL2G";
142 // Take Me Home, Country Roads (song e) :
143 const char songe[]="T199O0L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EEEDL8EL4GL1GP1L4DDE.L2D.L4EGGHL1HL4AAAAH.L2A.L4EGGAL2G.L4GAL1HL
144 // Es tanzt ein Bi-ba-butemann (song f) :
145 const char songf[]="T199O0L8DGGO1DDO0HHGGAADDL4GF8L8DGGO1DDO0HHGGAADDL4GF8L8HAHO1CO0AHO1CD00L8HAHO1CO0AHO1CD00DGGO1DDO0HHGGAADDL4G";
146 // Ich geh mit meiner Laterne (song g) :
147 const char songg[]="T120O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8A0L14CO0L8AL4F
148 // The little drummer boy (song h) :
149 const char songh[]="T120P2O0L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+L4F+L4F+L8GF+L4GL2F+P2L4EF+L4GAAHL8AGL4F+L2EP2L4EF+L4GAAHO1L
150 // Hey, Pippi Langstrumpf (song i) :
151 const char songi[]="T18O0LL4AON00L4DF+DL2EL8GF+EDL4C+EO1AON00L4C+L2DF+OLL4AON00L4DF+DL2EL8GF+EDL4C+EO1AON00L4C+DF4P2OLL4AON00L4DF+DL
152 // Stille Nacht, heilige Nacht (song j) :
153 const char songj[]="T72O0L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8DO0L4H.O1L4CL8CO0L4G.L4AL8AO1L8C.O0L16HL8AL8G.L16AL8GL4E.L4AL8AO1L8C.O0L
154 // Junge komm bald wieder (song k) :
155 const char songk[]="T120O0L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4EEL8D+L8EL4F+L4E.L8EL4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+
156 // Lili Marleen (song l) :
157 const char songl[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4H
158 // musical scale / chromatic scale / for testing purpose (song m) :
159 const char songm[]="T60ONO0L1C.OLL1C.";
160
161 char song[MAX_SONG_LENGTH];
162
163
164 // USER CODE END
```

Double click **MAIN.C** and insert Global Variables:

```
// default values for global variables - will be overwritten before use:
```

```
volatile unsigned int note=12;  
volatile unsigned int octave=1;  
volatile unsigned int current_note_length=16113;  
volatile unsigned int old_note_length=16113;  
volatile unsigned int tempo=120; // 120 beats/minute  
volatile unsigned int pos=0; // current note  
volatile unsigned int max=0; // song length
```

```
// song counters:
```

```
unsigned int next_song_a=0; // song counter song a  
unsigned int next_song_b=0; // song counter song b  
unsigned int next_song_c=0; // song counter song c  
unsigned int next_song_d=0; // song counter song d  
unsigned int next_song_e=0; // song counter song e  
unsigned int next_song_f=0; // song counter song f  
unsigned int next_song_g=0; // song counter song g  
unsigned int next_song_h=0; // song counter song h  
unsigned int next_song_i=0; // song counter song i  
unsigned int next_song_j=0; // song counter song j  
unsigned int next_song_k=0; // song counter song k  
unsigned int next_song_l=0; // song counter song l  
unsigned int next_song_m=0; // song counter song m
```

```
volatile bit OctaveLOW=OFF;
```



XE167 - µVision3 - [C:\XE167\MAIN.C\*]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

152 // Stille Nacht, heilige Nacht (song j):
153 const char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8DO0L4H.O1L4CL8CO0L
154 // Junge komm bald wieder (song k):
155 const char songk[]="T12000L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4EEL8D
156 // Lili Marleen (song l):
157 const char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L1
158 // musical scale / chromatic scale / for testing purpose (song m) :
159 const char songm[]="T600NO0L1C.OLL1C.";
160
161 char song[MAX_SONG_LENGTH];
162
163 // default values for global variables - will be overwritten before use:
164 volatile unsigned int note=12;
165 volatile unsigned int octave=1;
166 volatile unsigned int current_note_length=16113;
167 volatile unsigned int old_note_length=16113;
168 volatile unsigned int tempo=120; // 120 beats/minute
169 volatile unsigned int pos=0; // current note
170 volatile unsigned int max=0; // song length
171
172 // song counters:
173 unsigned int next_song_a=0; // song counter song a
174 unsigned int next_song_b=0; // song counter song b
175 unsigned int next_song_c=0; // song counter song c
176 unsigned int next_song_d=0; // song counter song d
177 unsigned int next_song_e=0; // song counter song e
178 unsigned int next_song_f=0; // song counter song f
179 unsigned int next_song_g=0; // song counter song g
180 unsigned int next_song_h=0; // song counter song h
181 unsigned int next_song_i=0; // song counter song i
182 unsigned int next_song_j=0; // song counter song j
183 unsigned int next_song_k=0; // song counter song k
184 unsigned int next_song_l=0; // song counter song l
185 unsigned int next_song_m=0; // song counter song m
186
187 volatile bit OctaveLOW=OFF;
188
189 // USER CODE END
190
191
192 //*****
193 // @External Prototypes

```

MAIN.C

Simulation

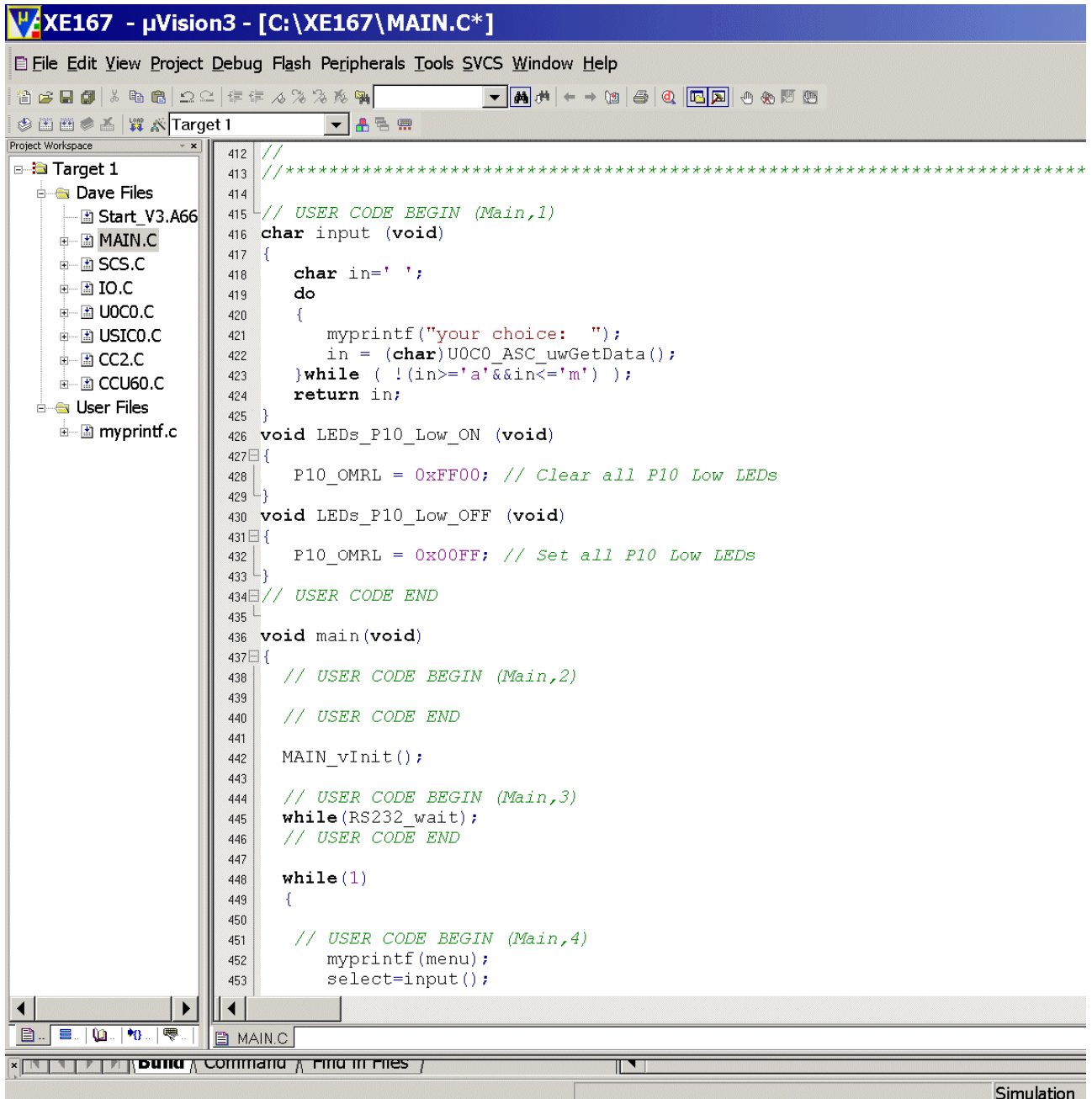
Double click **MAIN.C** and **change** function "char input (void)":

from:

```
char input (void)
{
    char in=' ';
    do
    {
        myprintf("your choice: ");
        in = (char)U0C0_ASC_uwGetData();
    }while (in!='1' && in!= '2' && in != '3');
    return in;
}
```

to:

```
char input (void)
{
    char in=' ';
    do
    {
        myprintf("your choice: ");
        in = (char)U0C0_ASC_uwGetData();
    }while ( !(in>='a'&&in<='m') );
    return in;
}
```



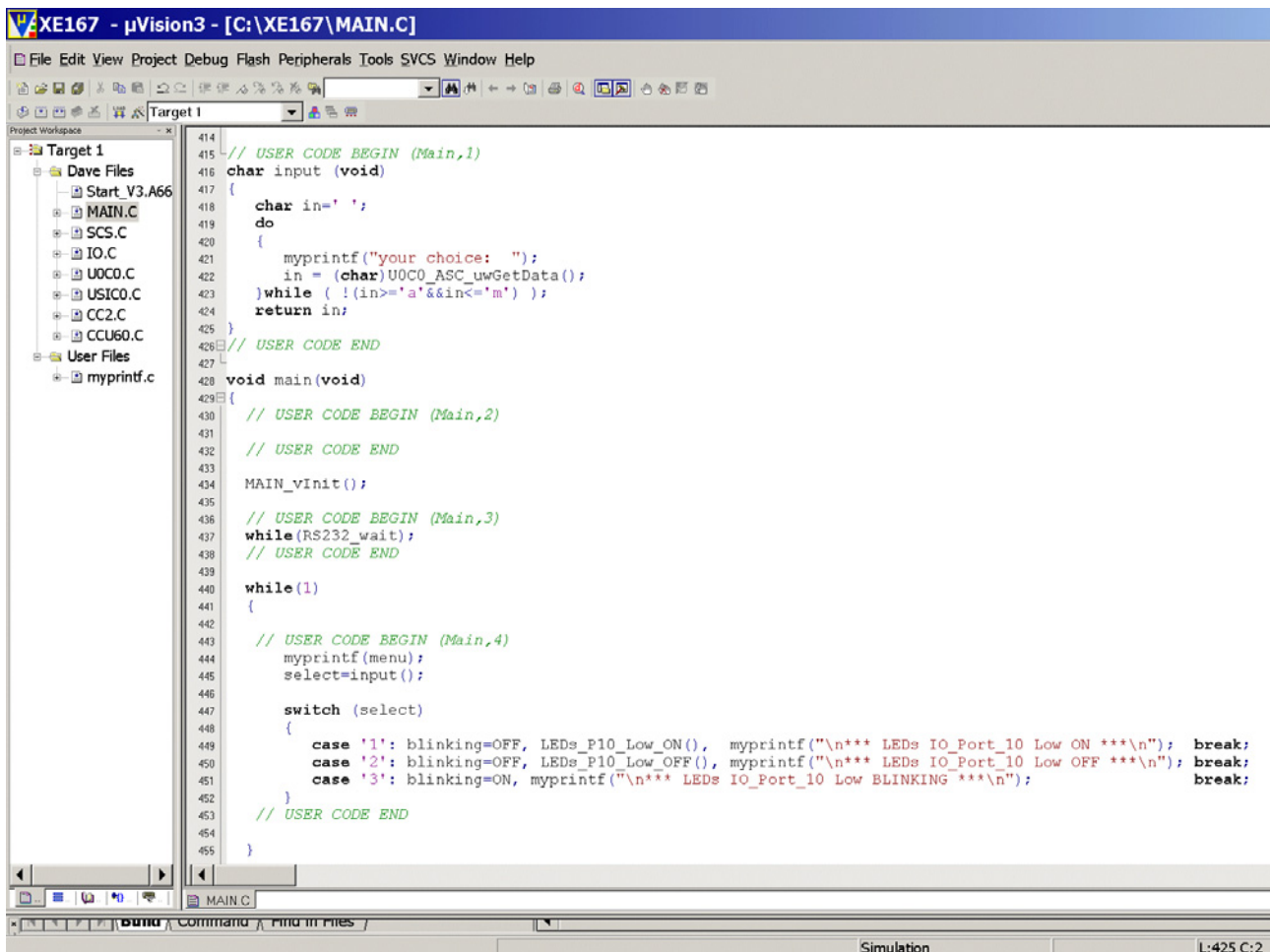
```

412 //
413 //*****
414 // USER CODE BEGIN (Main,1)
415 char input (void)
416 {
417     char in=' ';
418     do
419     {
420         myprintf("your choice: ");
421         in = (char)UOC0_ASC_uwGetData();
422     }while ( !(in>='a'&&in<='m') );
423     return in;
424 }
425 void LEDs_P10_Low_ON (void)
426 {
427     P10_OMRL = 0xFF00; // Clear all P10 Low LEDs
428 }
429 void LEDs_P10_Low_OFF (void)
430 {
431     P10_OMRL = 0x00FF; // Set all P10 Low LEDs
432 }
433 // USER CODE END
434 void main(void)
435 {
436     // USER CODE BEGIN (Main,2)
437     // USER CODE END
438     MAIN_vInit();
439     // USER CODE BEGIN (Main,3)
440     while(RS232_wait);
441     // USER CODE END
442     while(1)
443     {
444         // USER CODE BEGIN (Main,4)
445         myprintf(menu);
446         select=input();
447     }
448 }
449
450
451
452
453

```



Double click **MAIN.C** and **delete** functions:  
LEDs\_P10\_Low\_ON() and void LEDs\_P10\_Low\_OFF():

```

414 // USER CODE BEGIN (Main,1)
415 char input (void)
416 {
417     char in=' ';
418     do
419     {
420         myprintf("your choice: ");
421         in = (char)U0C0_ASC_ufGetData();
422     }while ( !(in>='a' && in<='m') );
423     return in;
424 }
425 // USER CODE END
426
427 void main(void)
428 {
429     // USER CODE BEGIN (Main,2)
430     // USER CODE END
431
432     MAIN_vinit();
433
434     // USER CODE BEGIN (Main,3)
435     while(RS232_wait);
436     // USER CODE END
437
438     while(1)
439     {
440         // USER CODE BEGIN (Main,4)
441         myprintf(menu);
442         select=input();
443
444         switch (select)
445         {
446             case '1': blinking=OFF, LEDs_P10_Low_ON(), myprintf("\n*** LEDs IO_Port_10 Low ON ***\n"); break;
447             case '2': blinking=OFF, LEDs_P10_Low_OFF(), myprintf("\n*** LEDs IO_Port_10 Low OFF ***\n"); break;
448             case '3': blinking=ON, myprintf("\n*** LEDs IO_Port_10 Low BLINKING ***\n"); break;
449         }
450         // USER CODE END
451     }
452 }
453
454
455

```

Double click **MAIN.C** and insert the function **play\_song()**:

```
void play_song(void)
{
    max=0;
    if ( next_song_a && ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
        myprintf("\nplaying: Maus am Mars\n");
    if (next_song_b && ((sizeof(songb)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songb), max=(sizeof(songb))-1, --next_song_b,
        myprintf("\nplaying: Yesterday\n");
    if (next_song_c && ((sizeof(songc)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songc), max=(sizeof(songc))-1, --next_song_c,
        myprintf("\nplaying: Frere Jacques - Lazy John - Bruder Jakob\n");
    if (next_song_d && ((sizeof(songd)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songd), max=(sizeof(songd))-1, --next_song_d,
        myprintf("\nplaying: Happy birthday\n");
    if (next_song_e && ((sizeof(songe)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songe), max=(sizeof(songe))-1, --next_song_e,
        myprintf("\nplaying: Take Me Home, Country Roads\n");
    if (next_song_f && ((sizeof(songf)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songf), max=(sizeof(songf))-1, --next_song_f,
        myprintf("\nplaying: Es tanzt ein Bi-ba-butzemann\n");
    if (next_song_g && ((sizeof(songg)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songg), max=(sizeof(songg))-1, --next_song_g,
        myprintf("\nplaying: Ich geh mit meiner Laterne\n");
    if (next_song_h && ((sizeof(songh)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songh), max=(sizeof(songh))-1, --next_song_h,
        myprintf("\nplaying: The little drummer boy\n");
    if (next_song_i && ((sizeof(songi)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songi), max=(sizeof(songi))-1, --next_song_i,
        myprintf("\nplaying: Hey, Pippi Langstrumpf\n");
    if (next_song_j && ((sizeof(songj)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songj), max=(sizeof(songj))-1, --next_song_j,
        myprintf("\nplaying: Stille Nacht, heilige Nacht\n");
    if (next_song_k && ((sizeof(songk)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songk), max=(sizeof(songk))-1, --next_song_k,
        myprintf("\nplaying: Junge komm bald wieder\n");
    if (next_song_l && ((sizeof(songl)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songl), max=(sizeof(songl))-1, --next_song_l,
        myprintf("\nplaying: Lili Marleen\n");
    if (next_song_m && ((sizeof(songm)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songm), max=(sizeof(songm))-1, --next_song_m,
        myprintf("\nplaying: musical scale / chromatic scale / for testing purpose\n");

    sprintf(mb1,"song-length = %5u Byte[s] \n",max);
    myprintf(mb1);
}
```

```
pos=0;

if (max>0) // there is something to play
{
    // start CCU60 - Timer T12 - ISR the first time
    CCU60_ISS = CCU60_IS | 0x80; // set ST12PM -> Set-Timer-T12Period-Match-Flag

    while (pos<=max); // wait until song end is reached or abort by user is done
}

if ( (U0C0_RBUF=='z') )
{
    myprintf("Song aborted.\n");
}
else
{
    sprintf(mb1,"End of the song reached (pos=%5u of max%5u).\n",pos,max);
    myprintf(mb1);
}
}
```





**XE167 - µVision3 - [C:\XE167\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

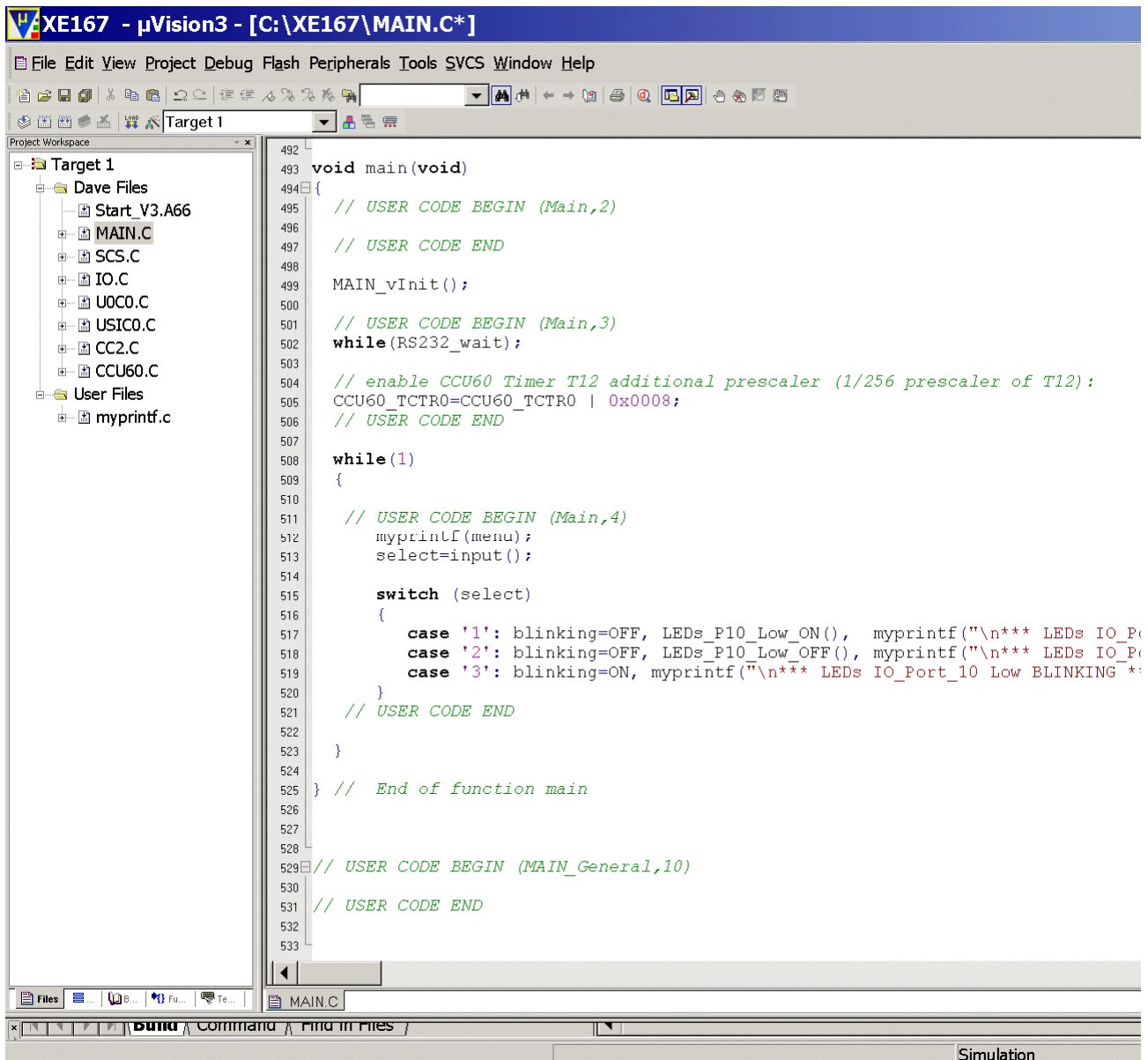
415 // USER CODE BEGIN (Main,1)
416 char input (void)
417 {
418     char in=' ';
419     do
420     {
421         myprintf("your choice: ");
422         in = (char)U0C0_ASC_uwGetData();
423     }while ( !(in>='a'&&in<='m') );
424     return in;
425 }
426 void play_song(void)
427 {
428     max=0;
429     if ( next_song_a && ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
430         strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
431         myprintf("\nplaying: Maus am Mars\n");
432     if (next_song_b && ((sizeof(songb)-1)< MAX_SONG_LENGTH) )
433         strcpy(song,songb), max=(sizeof(songb))-1, --next_song_b,
434         myprintf("\nplaying: Yesterday\n");
435     if (next_song_c && ((sizeof(songc)-1)< MAX_SONG_LENGTH) )
436         strcpy(song,songc), max=(sizeof(songc))-1, --next_song_c,
437         myprintf("\nplaying: Frere Jacques - Lazy John - Bruder Jakob\n");
438     if (next_song_d && ((sizeof(songd)-1)< MAX_SONG_LENGTH) )
439         strcpy(song,songd), max=(sizeof(songd))-1, --next_song_d,
440         myprintf("\nplaying: Happy birthday\n");
441     if (next_song_e && ((sizeof(songe)-1)< MAX_SONG_LENGTH) )
442         strcpy(song,songe), max=(sizeof(songe))-1, --next_song_e,
443         myprintf("\nplaying: Take Me Home, Country Roads\n");
444     if (next_song_f && ((sizeof(songf)-1)< MAX_SONG_LENGTH) )
445         strcpy(song,songf), max=(sizeof(songf))-1, --next_song_f,
446         myprintf("\nplaying: Es tanzt ein Bi-ba-butzemann\n");
447     if (next_song_g && ((sizeof(songg)-1)< MAX_SONG_LENGTH) )
448         strcpy(song,songg), max=(sizeof(songg))-1, --next_song_g,
449         myprintf("\nplaying: Ich geh mit meiner Laterne\n");
450     if (next_song_h && ((sizeof(songh)-1)< MAX_SONG_LENGTH) )
451         strcpy(song,songh), max=(sizeof(songh))-1, --next_song_h,
452         myprintf("\nplaying: The little drummer boy\n");
453     if (next_song_i && ((sizeof(songi)-1)< MAX_SONG_LENGTH) )
454         strcpy(song,songi), max=(sizeof(songi))-1, --next_song_i,
455         myprintf("\nplaying: Hey, Pippi Langstrumpf\n");
456     if (next_song_j && ((sizeof(songj)-1)< MAX_SONG_LENGTH) )
457         strcpy(song,songj), max=(sizeof(songj))-1, --next_song_j,
458         myprintf("\nplaying: Stille Nacht, heilige Nacht\n");
459     if (next_song_k && ((sizeof(songk)-1)< MAX_SONG_LENGTH) )
460         strcpy(song,songk), max=(sizeof(songk))-1, --next_song_k,
461         myprintf("\nplaying: Junge komm bald wieder\n");
462     if (next_song_l && ((sizeof(songl)-1)< MAX_SONG_LENGTH) )
463         strcpy(song,songl), max=(sizeof(songl))-1, --next_song_l,
464         myprintf("\nplaying: Lili Marleen\n");
465     if (next_song_m && ((sizeof(songm)-1)< MAX_SONG_LENGTH) )
466         strcpy(song,songm), max=(sizeof(songm))-1, --next_song_m,
467         myprintf("\nplaying: musical scale / chromatic scale / for testing purpose\n");
468
469     sprintf(mbl,"song-length = %5u Byte[s] \n",max);
470     myprintf(mbl);
471     pos=0;
472
473     if (max>0) // there is something to play
474     {
475         // start CCU60 - Timer T12 - ISR the first time
476         CCU60_ISS = CCU60_IS | 0x80; // set ST12PM -> Set-Timer-T12Period-Match-Flag
477
478         while (pos<=max); // wait until song end is reached or abort by user is done
479     }
480
481     if ( (U0C0_RBUF=='z') )
482     {
483         myprintf("Song aborted.\n");
484     }
485     else
486     {
487         sprintf(mbl,"End of the song reached (pos=%5u of max%5u).\n",pos,max);
488         myprintf(mbl);
489     }
490 }
491 // USER CODE END
492
493 void main(void)
494 {
495     // USER CODE BEGIN (Main,2)

```

Simulation

Double click **MAIN.C** and insert the following code into the **main** function:

```
// enable CCU60 Timer T12 additional prescaler (1/256 prescaler of T12):
CCU60_TCTR0=CCU60_TCTR0 | 0x0008;
```





**Remember:**

Unfortunately bit T12PRE is not available in the DAVe dialog.

Source: User's Manual:

The input clock for timer T12 can be from  $f_{CCU60}$  to a maximum of  $f_{CCU60}/128$  and is configured by bit field T12CLK. In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler of T12 if bit T12PRE = 1.

Adobe Reader - [xe166\_um\_v2.0\_2007\_12\_vol2per.pdf]

File Edit View Document Tools Window Help

125%

**TCTR0**  
Timer Control Register 0

Reset Value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STE 13	T13R	T13 PRE	T13CLK			CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK			
r	rh	rh	rw	rw			rw	rh	rh	rh	rw	rw			

**Field**      **Bits**      **Type**      **Description**

**T12CLK**      [2:0]      rw      **Timer T12 Input Clock Select**  
Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation  $f_{T12} = f_{CC6} / 2^{T12CLK}$ .  
000<sub>B</sub>  $f_{T12} = f_{CC6}$   
001<sub>B</sub>  $f_{T12} = f_{CC6} / 2$   
010<sub>B</sub>  $f_{T12} = f_{CC6} / 4$   
011<sub>B</sub>  $f_{T12} = f_{CC6} / 8$   
100<sub>B</sub>  $f_{T12} = f_{CC6} / 16$   
101<sub>B</sub>  $f_{T12} = f_{CC6} / 32$   
110<sub>B</sub>  $f_{T12} = f_{CC6} / 64$   
111<sub>B</sub>  $f_{T12} = f_{CC6} / 128$

**T12PRE**      3      rw      **Timer T12 Prescaler Bit**  
In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12.  
0<sub>B</sub>      The additional prescaler for T12 is disabled.  
1<sub>B</sub>      The additional prescaler for T12 is enabled.

**Timer 12 Resolution:**

66 MHz / 256 (T12PRE=1, done by software) / 32 = 8.056,64 Hz → Resolution = 124,12 μs



Double click **MAIN.C** and **change** the following code (**main** function, **while(1)** loop)

from:

```
myprintf(menu);
select=input();

switch (select)
{
    case '1': blinking=OFF, LEDs_P10_Low_ON(), myprintf("\n*** LEDs IO_Port_10 Low
ON ***\n"); break;
    case '2': blinking=OFF, LEDs_P10_Low_OFF(), myprintf("\n*** LEDs IO_Port_10 Low
OFF ***\n"); break;
    case '3': blinking=ON, myprintf("\n*** LEDs IO_Port_10 Low BLINKING ***\n");
break;
}
```

to:

```
myprintf(menu);
select=input();

switch (select)
{
    case 'a': ++next_song_a, play_song(); break;
    case 'b': ++next_song_b, play_song(); break;
    case 'c': ++next_song_c, play_song(); break;
    case 'd': ++next_song_d, play_song(); break;
    case 'e': ++next_song_e, play_song(); break;
    case 'f': ++next_song_f, play_song(); break;
    case 'g': ++next_song_g, play_song(); break;
    case 'h': ++next_song_h, play_song(); break;
    case 'i': ++next_song_i, play_song(); break;
    case 'j': ++next_song_j, play_song(); break;
    case 'k': ++next_song_k, play_song(); break;
    case 'l': ++next_song_l, play_song(); break;
    case 'm': ++next_song_m, play_song(); break;
}
```

**XE167 - µVision3 - [C:\XE167\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - UOC0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

492 void main(void)
493 {
494     // USER CODE BEGIN (Main,2)
495     // USER CODE END
496
497     MAIN_vInit();
498
499     // USER CODE BEGIN (Main,3)
500     while(RS232_wait);
501
502     // enable CCU60 Timer T12 additional prescaler (1/256 prescaler of T12):
503     CCU60_TCTR0=CCU60_TCTR0 | 0x0008
504     // USER CODE END
505
506     while(1)
507     {
508         // USER CODE BEGIN (Main,4)
509         myprintf(menu);
510         select=input();
511
512         switch (select)
513         {
514             case 'a': ++next_song_a, play_song(); break;
515             case 'b': ++next_song_b, play_song(); break;
516             case 'c': ++next_song_c, play_song(); break;
517             case 'd': ++next_song_d, play_song(); break;
518             case 'e': ++next_song_e, play_song(); break;
519             case 'f': ++next_song_f, play_song(); break;
520             case 'g': ++next_song_g, play_song(); break;
521             case 'h': ++next_song_h, play_song(); break;
522             case 'i': ++next_song_i, play_song(); break;
523             case 'j': ++next_song_j, play_song(); break;
524             case 'k': ++next_song_k, play_song(); break;
525             case 'l': ++next_song_l, play_song(); break;
526             case 'm': ++next_song_m, play_song(); break;
527
528         }
529         // USER CODE END
530     }
531 }
532
533

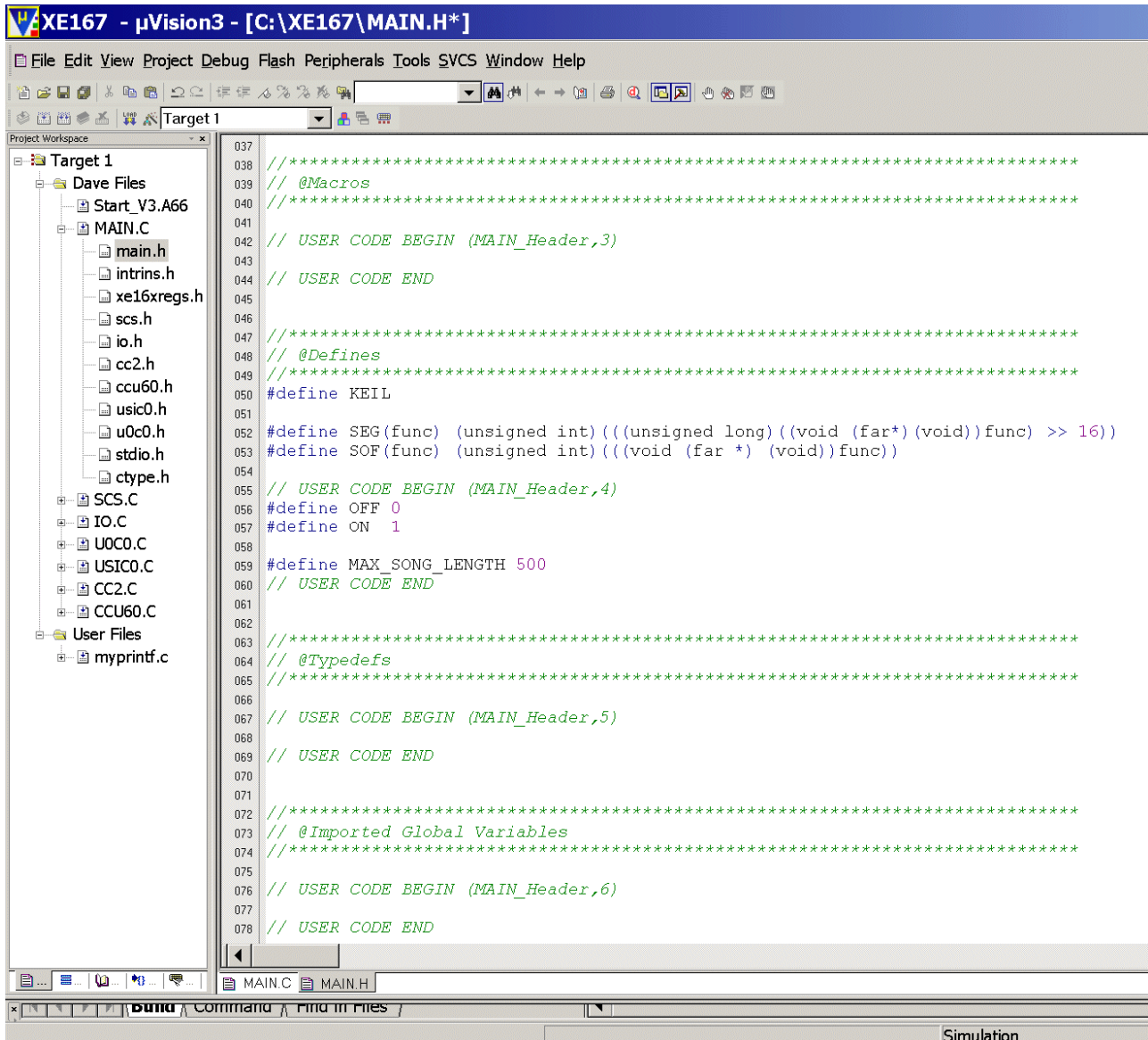
```

MAIN.C

Simulation

Double click **Main.h** and **insert** the following Defines:

```
#define MAX_SONG_LENGTH 500
```





Double click **Main.h** and **change** extern declarations “Global Variables”

from:

```
extern volatile int RS232_wait;  
extern volatile bit blinking;
```

to:

```
extern volatile int RS232_wait;  
  
extern unsigned int T13_values[];  
extern unsigned int length_of_a_whole_note;  
extern char song[];  
extern volatile unsigned int note;  
extern volatile unsigned int octave;  
extern volatile unsigned int current_note_length;  
extern volatile unsigned int old_note_length;  
extern volatile unsigned int tempo;  
extern volatile unsigned int pos;  
extern volatile unsigned int max;  
extern volatile bit OctaveLOW;
```

**XE167 - µVision3 - [C:\XE167\MAIN.H\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
      - main.h
      - intrins.h
      - xe16xregs.h
      - scs.h
      - io.h
      - cc2.h
      - ccu60.h
      - usic0.h
      - u0c0.h
      - stdio.h
      - ctype.h
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

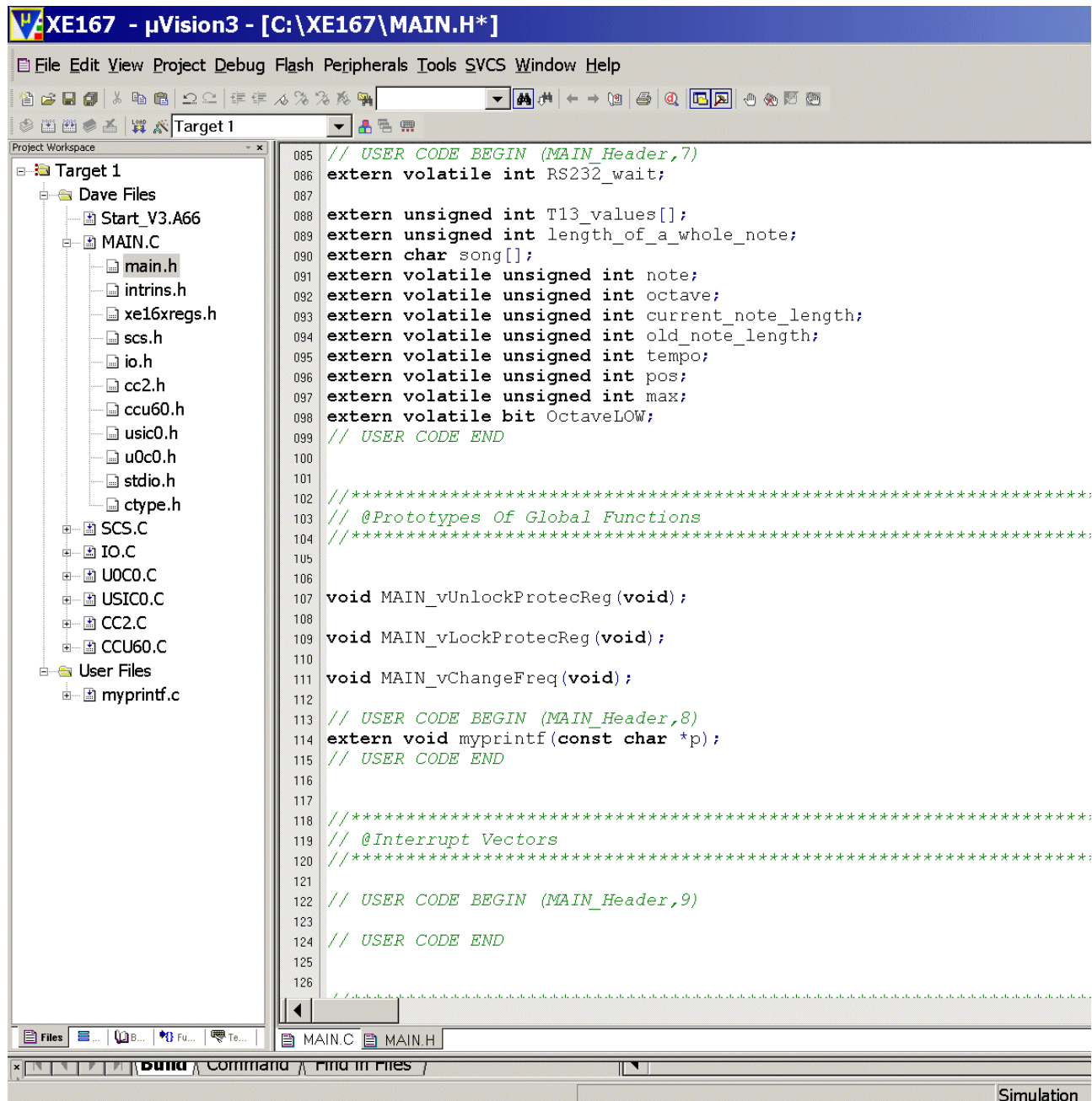
075 // USER CODE BEGIN (MAIN_Header,6)
076
077 // USER CODE END
078
079
080 //*****
081 // @Global Variables
082 //*****
083
084 // USER CODE BEGIN (MAIN_Header,7)
085 extern volatile int RS232_wait;
086
087
088 extern unsigned int T13_values[];
089 extern unsigned int length_of_a_whole_note;
090 extern char song[];
091 extern volatile unsigned int note;
092 extern volatile unsigned int octave;
093 extern volatile unsigned int current_note_length;
094 extern volatile unsigned int old_note_length;
095 extern volatile unsigned int tempo;
096 extern volatile unsigned int pos;
097 extern volatile unsigned int max;
098 extern volatile bit OctaveLOW;
099 // USER CODE END
100
101 //*****
102 // @Prototypes Of Global Functions
103 //*****
104
105
106 void MAIN_vUnlockProtecReg(void);
107
108 void MAIN_vLockProtecReg(void);
109
110 void MAIN_vChangeFreq(void);
111
112 // USER CODE BEGIN (MAIN_Header,8)
113 extern void LEDs_P10_Low_ON(void);
114 extern void LEDs_P10_Low_OFF(void);
115 extern void myprintf(const char *p);
116 // USER CODE END
  
```

MAIN.C MAIN.H

Simulation

Double click **Main.h** and **delete** extern declarations “Prototypes Of Global Functions”

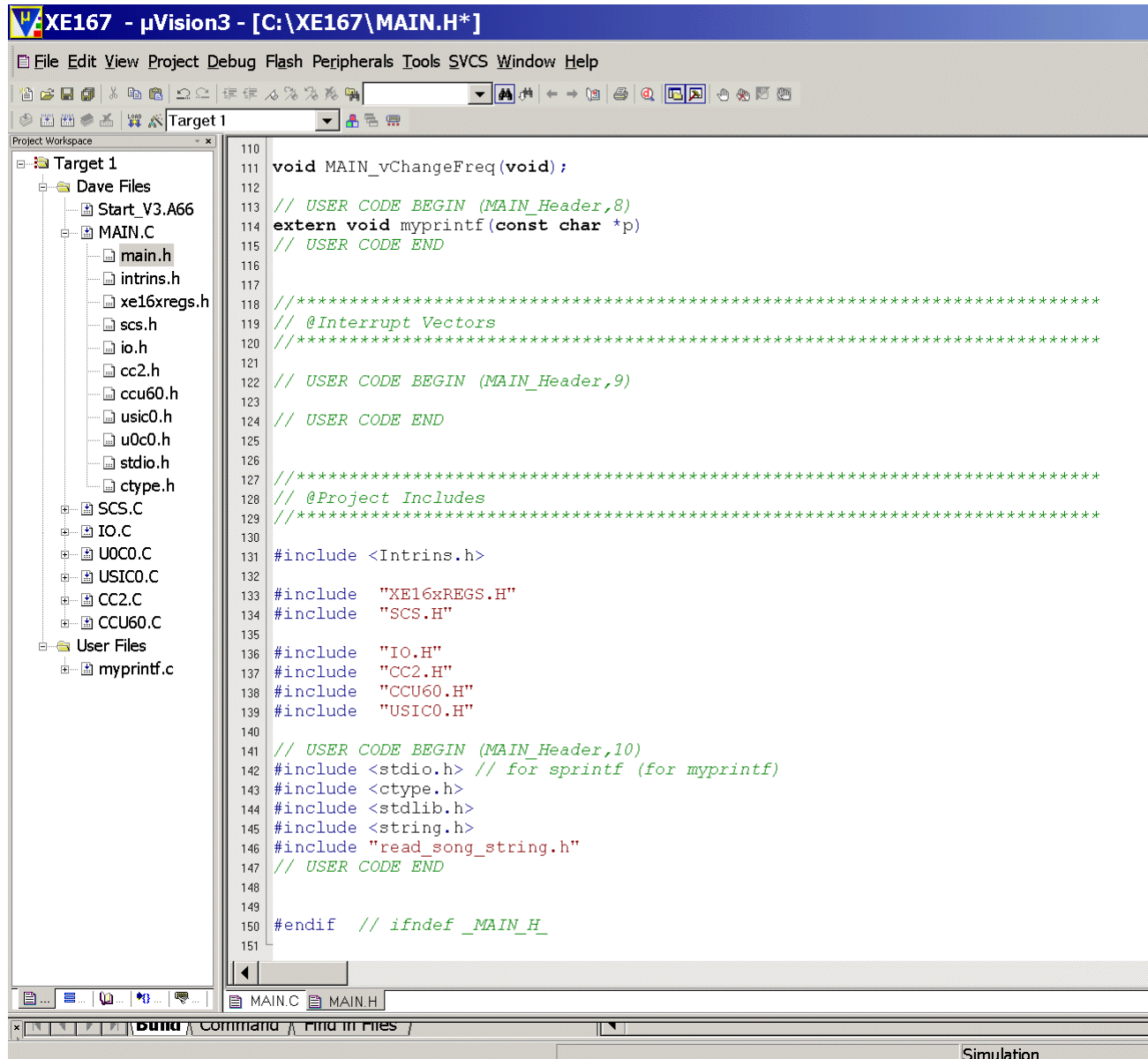
extern void LEDs\_P10\_Low\_ON (void);  
extern void LEDs\_P10\_Low\_OFF (void);:





Double click **Main.h** and **insert** include files:

```
#include <stdlib.h>
#include <string.h>
#include "read_song_string.h"
```



Double click **CC2.C** change code (CAPCOM 2 Timer 7 Interrupt Service Routine)

from:

```
if(RS232_wait)
    RS232_wait--;

if (blinking)
{
    P10_OMRL = 0xFFFF; // Toggle P10 Low LEDs
}
```

to:

```
if(RS232_wait)
    RS232_wait--;

P10_OMRL = 0x6060; // Toggle P10 Low Pin 5 and Pin 6 LED
```

**XE167 - µVision3 - [C:\XE167\CC2.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - UOC0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

295 // 7. It is called when overflow of the timer 7 register
296 // occurs.
297 // Please note that you have to add application specific code
298 // to this function.
299 //
300 //-----
301 // @Returnvalue  None
302 //
303 //-----
304 // @Parameters  None
305 //
306 //-----
307 // @Date        24.01.2008
308 //
309 //*****
310 // USER CODE BEGIN (Tmr7,1)
311 // USER CODE END
312
313 void CC2_viTmr7(void) interrupt CC2_T7INT
314 {
315 // USER CODE BEGIN (Tmr7,2)
316 if (RS232_wait)
317 RS232_wait--;
318
319 P10_OMRL = 0x6060; // Toggle P10 Low Pin 5 and Pin 6 LED
320 // USER CODE END
321 } // End of function CC2_viTmr7
322
323
324
325
326
327
328
329 // USER CODE BEGIN (CC2_General,10)
330 // USER CODE END
331
332
333

```

MAIN.C MAIN.H CC2.C

Simulation



**Remember:**

Port\_10 Low pins used as GPIO:

Port Lines	Function	Comment
P10.3	Show start of next note	Toggled via Software
P10.4	Show start of next note	Toggled via Software
P10.5	„use: program running signal“	Toggled via CAPCOM2_Timer_7 ISR
P10.6	„use: program running signal“	Toggled via CAPCOM2_Timer_7 ISR



Double click CCU60.C insert code (CCU60 Interrupt Service Routine, Timer T12 period match) :

```

if ( (char)U0C0_RBUF == 'z' ) // song aborted by user
    pos=max+1;

if (pos<=max)
{
    read_song_string(); // read next note

    // T12, note length
    CCU60_T12PR=((float)current_note_length/(float)tempo*120.0); // period value note length
    CCU60_CC60SR=0; // not used (100% duty cycle)
    CCU60_CC61SR=0; // not used (100% duty cycle)
    // if compare value CCU6_CC62SR == 0 -> 100 % duty cycle note length
    CCU60_CC62SR=0;
    CCU60_vEnableShadowTransfer(CCU60_TIMER_12);

    // T13, note frequency
    CCU60_T13PR=T13_values[note]/octave; // note frequency
    CCU60_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %
    CCU60_vEnableShadowTransfer(CCU60_TIMER_13);

    if (note == 0) myprintf("note=c ");
    else if (note == 1) myprintf("note=cis");
    else if (note == 2) myprintf("note=d ");
    else if (note == 3) myprintf("note=dis");
    else if (note == 4) myprintf("note=e ");
    else if (note == 5) myprintf("note=f ");
    else if (note == 6) myprintf("note=fis");
    else if (note == 7) myprintf("note=g ");
    else if (note == 8) myprintf("note=gis");
    else if (note == 9) myprintf("note=a ");
    else if (note ==10) myprintf("note=ais");
    else if (note ==11) myprintf("note=h ");
    else if (note ==12) myprintf("note=---");
    else
        myprintf("note=???");

    if (octave == 1 && OctaveLOW==OFF) myprintf("*O0*");
    else if (octave == 1 && OctaveLOW== ON) myprintf("*OL*");
    else if (octave == 2) myprintf("*O1*");
    else if (octave == 4) myprintf("*O2*");
    else if (octave == 8) myprintf("*O3*");
    else
        myprintf("????");

    sprintf(mb2," T12-pv=%5u (%5u)",current_note_length,CCU60_T12PR);
    myprintf(mb2);
}

```



```

        sprintf(mb2, "T12-p=%1.2f (%1.2f)[s],
,current_note_length*124.12/1000.0/1000.0,CCU60_T12PR*124.12/1000.0/1000.0);
        myprintf(mb2);

        sprintf(mb2, "T13-pv=%5u (%5u)", T13_values[note]/octave,CCU60_T13PR);
        myprintf(mb2);

        if (OctaveLOW==OFF)
        {
            help=(float)T13_values[note];
            help=((help/octave)*60.606)/1000.0/1000.0/1000.0;
            help=1/help;
            sprintf(mb2, "T13-f=%7.0f[Hz]\n",help);
            myprintf(mb2);
        }
        else if (OctaveLOW==ON)
        {
            help=(float)T13_values[note];
            help=((help/octave)*121.21)/1000.0/1000.0/1000.0;
            help=1/help;
            sprintf(mb2, "T13-f=%7.0f[Hz]\n",help);
            myprintf(mb2);
        }

        P10_OMRL = 0x1818; // Toggle P10 Low Pin 3 and Pin 4 LED
        CCU60_vStartTmr(CCU60_TIMER_12); // Start next note (T12 single shot)

    }

```

XE167 - µVision3 - [C:\XE167\CCU60.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - UOC0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - User Files
    - myprintf.c

```

367 {
368     // Timer T12 period match detection
369
370     // USER CODE BEGIN (NodeI0,19)
371     if ( (char)UOC0_RBUF == 'z' ) // song aborted by user
372         pos=max+1;
373
374     if (pos<=max)
375     {
376         read_song_string(); // read next note
377
378         // T12, note length
379         CCU60_T12PR=((float)current_note_length/(float)tempo*120.0); // period value no
380         CCU60_CC60SR=0; // not used (100% duty cycle)
381         CCU60_CC61SR=0; // not used (100% duty cycle)
382         // if compare value CCU60_CC62SR == 0 -> 100 % duty cycle note length
383         CCU60_CC62SR=0;
384         CCU60_vEnableShadowTransfer(CCU60_TIMER_12);
385
386         // T13, note frequency
387         CCU60_T13PR=T13_values[note]/octave; // note frequency
388         CCU60_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %
389         CCU60_vEnableShadowTransfer(CCU60_TIMER_13);
390
391         if (note == 0) myprintf("note=c ");
392         else if (note == 1) myprintf("note=cis");
393         else if (note == 2) myprintf("note=d ");
394         else if (note == 3) myprintf("note=dis");
395         else if (note == 4) myprintf("note=e ");
396         else if (note == 5) myprintf("note=f ");
397         else if (note == 6) myprintf("note=fis");
398         else if (note == 7) myprintf("note=g ");
399         else if (note == 8) myprintf("note=gis");
400         else if (note == 9) myprintf("note=a ");
401         else if (note ==10) myprintf("note=ais");
402         else if (note ==11) myprintf("note=h ");
403         else if (note ==12) myprintf("note=---");
404         else myprintf("note=???");
405
406         if (octave == 1 && OctaveLOW==OFF) myprintf("*O0*");
407         else if (octave == 1 && OctaveLOW== ON) myprintf("*OL*");
408         else if (octave == 2) myprintf("*O1*");
409         else if (octave == 4) myprintf("*O2*");
410         else if (octave == 8) myprintf("*O3*");
411         else myprintf("???");
412
413         sprintf(mb2," T12-pv=%5u (%5u)",current_note_length,CCU60_T12PR);
414         myprintf(mb2);
415
416         sprintf(mb2,"T12-p=%1.2f (%1.2f)[s] ",current_note_length*124.12/1000.0/1000.0
417         myprintf(mb2);
418
419         sprintf(mb2,"T13-pv=%5u (%5u)", T13_values[note]/octave,CCU60_T13PR);
420         myprintf(mb2);
421
422         if (OctaveLOW==OFF)
423         {
424             help=(float)T13_values[note];
425             help=((help/octave)*60.606)/1000.0/1000.0/1000.0;
426             help=1/help;
427             sprintf(mb2,"T13-f=%7.0f[Hz]\n",help);
428             myprintf(mb2);
429         }
430         else if (OctaveLOW==ON)
431         {
432             help=(float)T13_values[note];
433             help=((help/octave)*121.21)/1000.0/1000.0/1000.0;
434             help=1/help;
435             sprintf(mb2,"T13-f=%7.0f[Hz]\n",help);
436             myprintf(mb2);
437         }
438
439         P10_OMRL = 0x1818; // Toggle P10 Low Pin 3 and Pin 4 LED
440         CCU60_vStartTmr(CCU60_TIMER_12); // Start next note (T12 single shot)
441
442     }
443     // USER CODE END
444
445     CCU60_ISR |= 0x0080; // clear flag CCU60_IS_T12PM
446 }

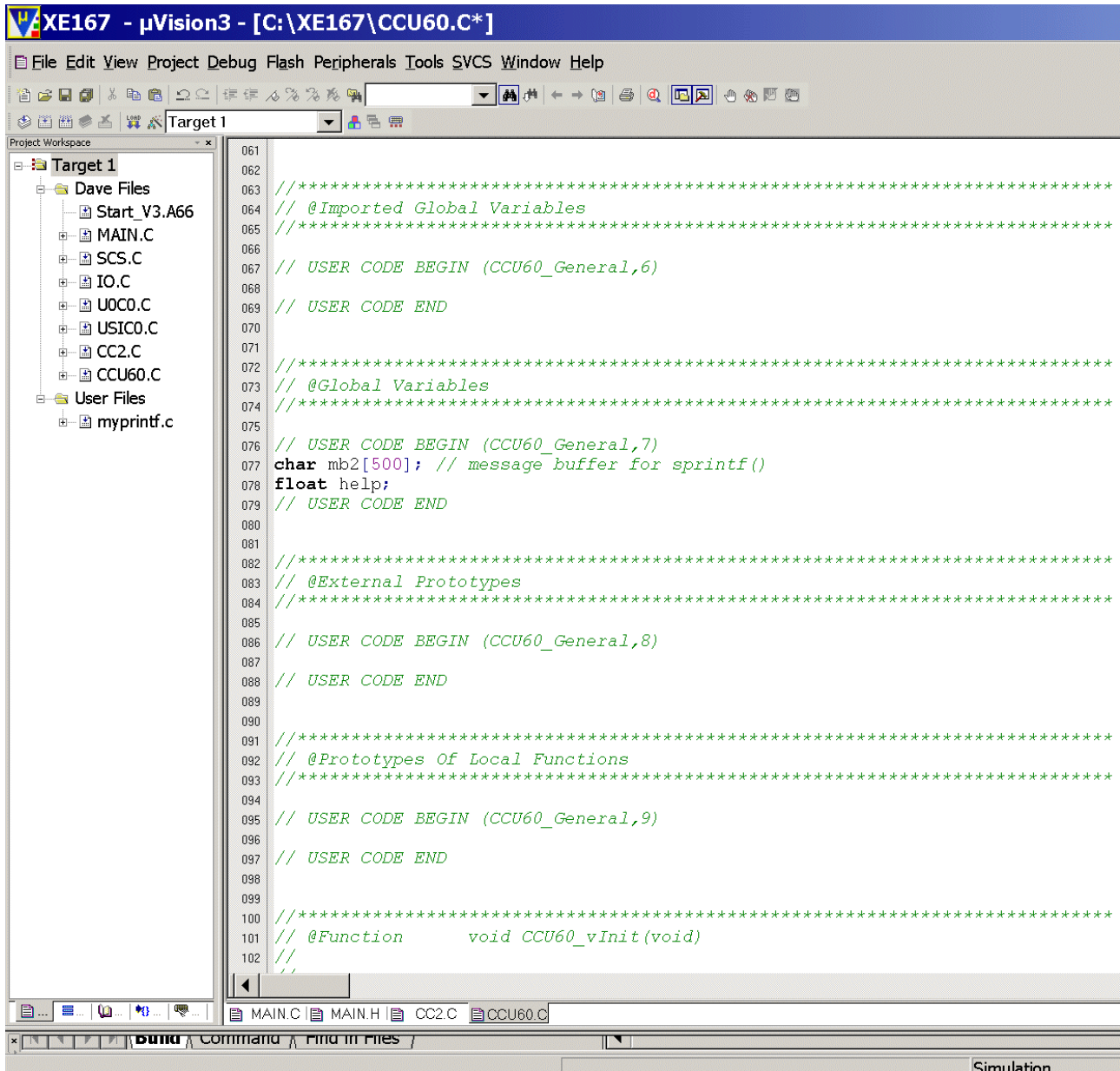
```

Files | B... | Fu... | Te... | MAIN.C | MAIN.H | CC2.C | CCU60.C

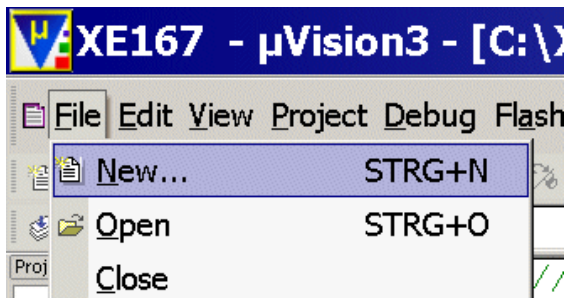
Simulation

Double click CCU60.C insert Global Variables:

```
char mb2[500]; // message buffer for sprintf()
float help;
```



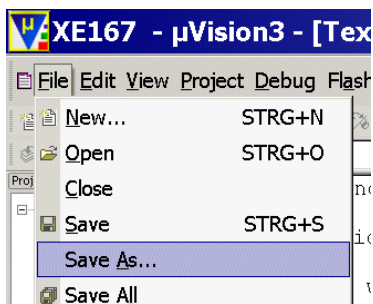
### File – New



### Insert:

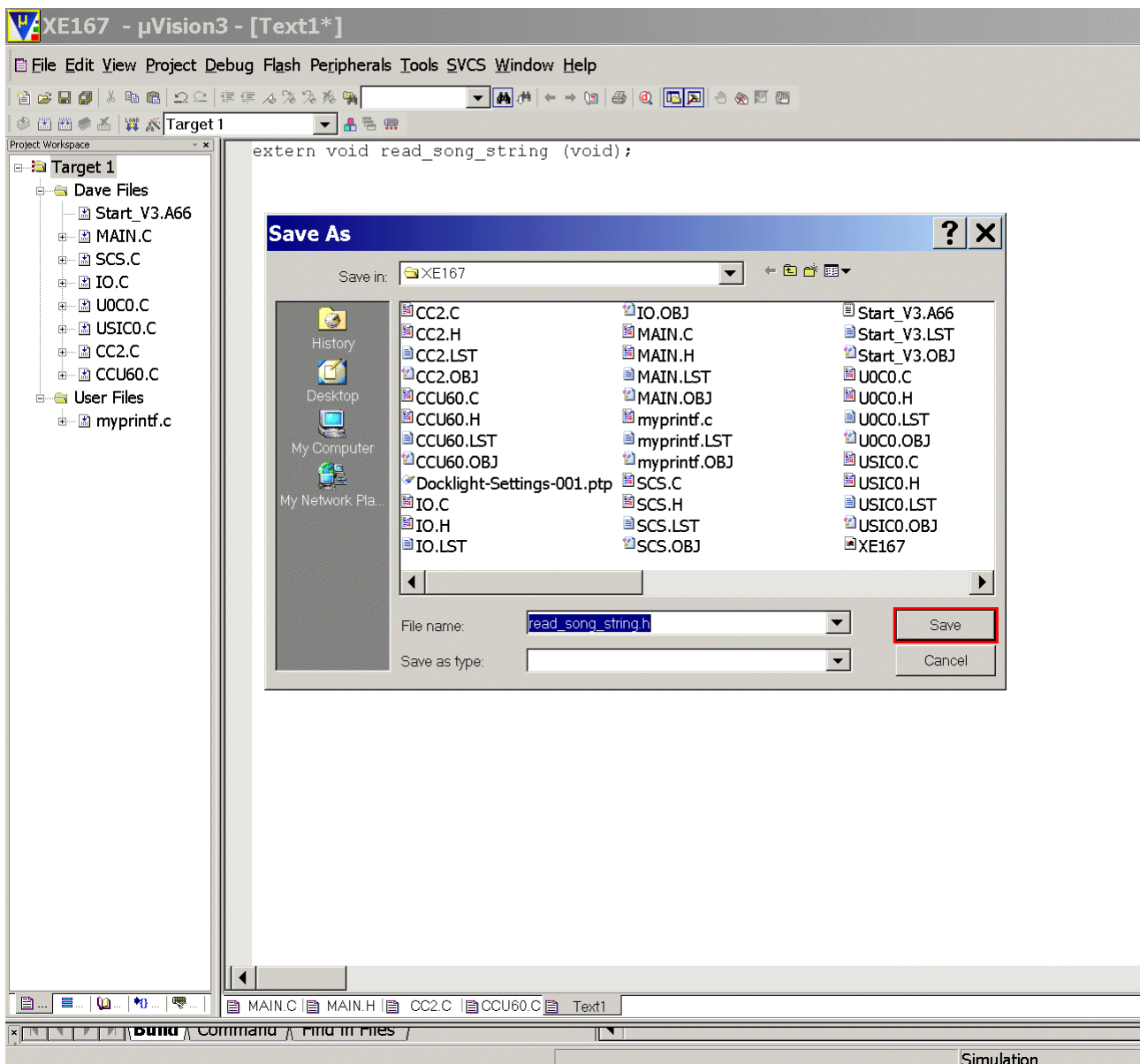
```
extern void read_song_string (void);
```

### File – Save As...



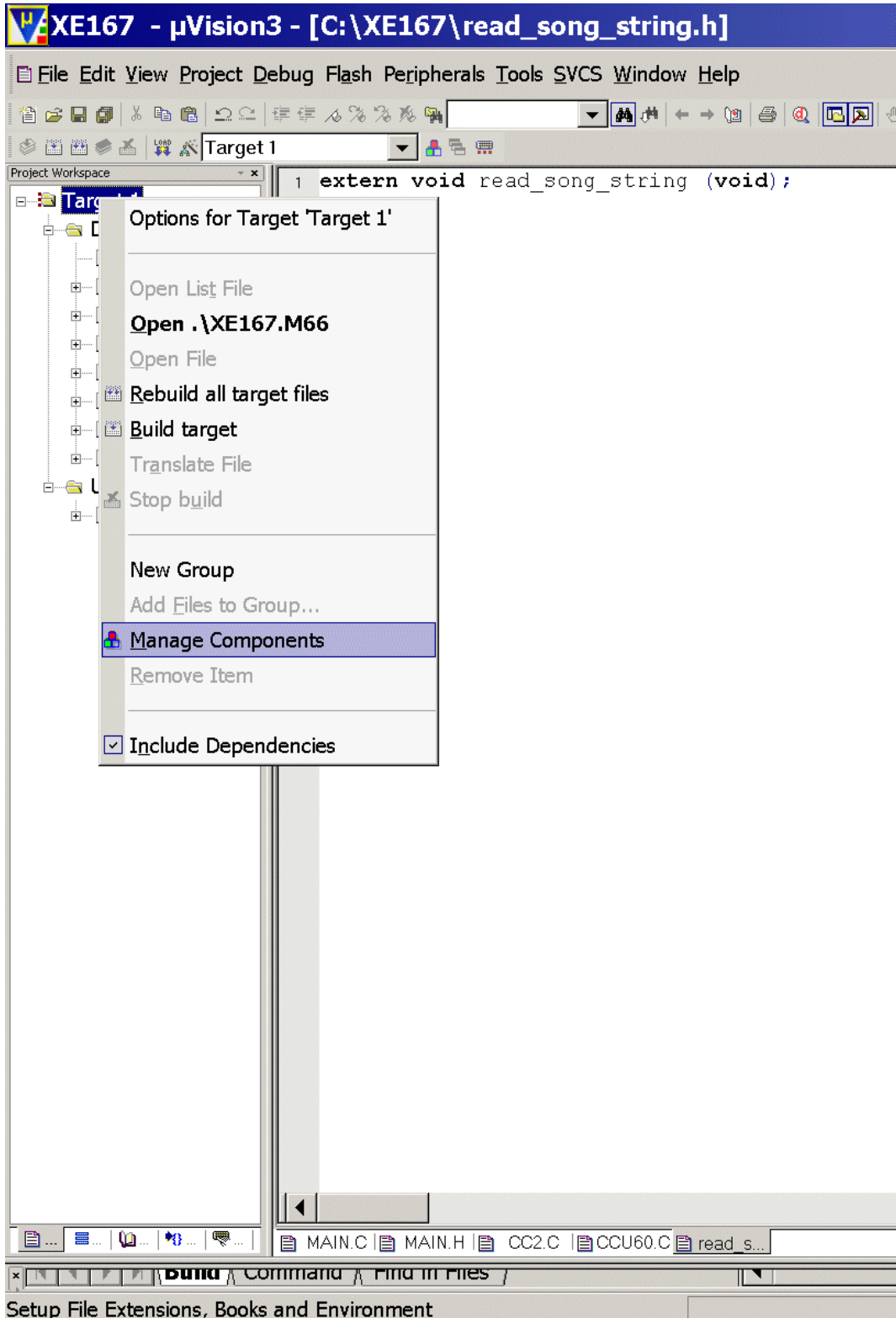


Insert: read\_song\_string.h

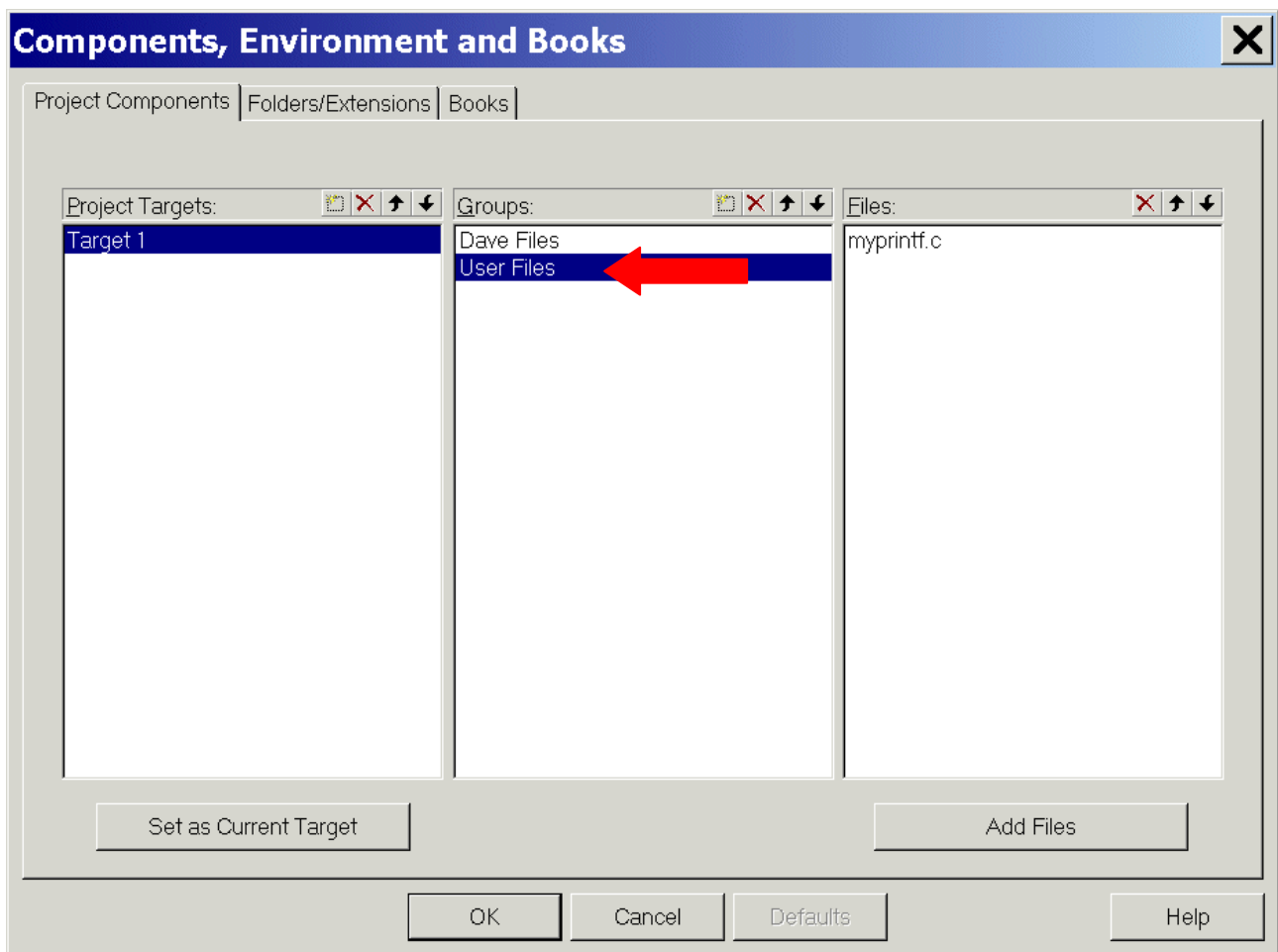


Click Save

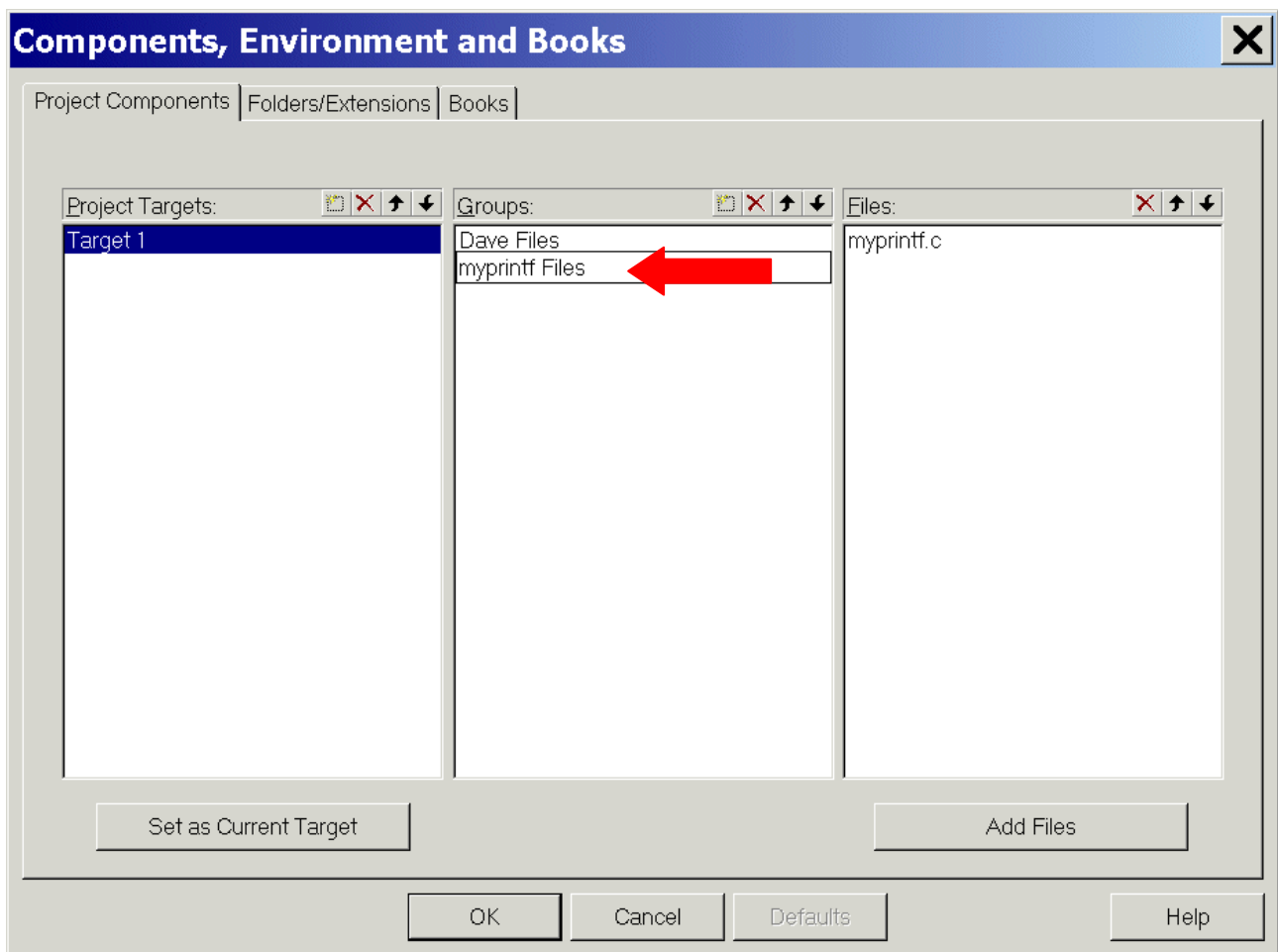
Mouse position: **Project Window**, Target 1: **click right mouse button**  
**click** Manage Components



Double click User Files:



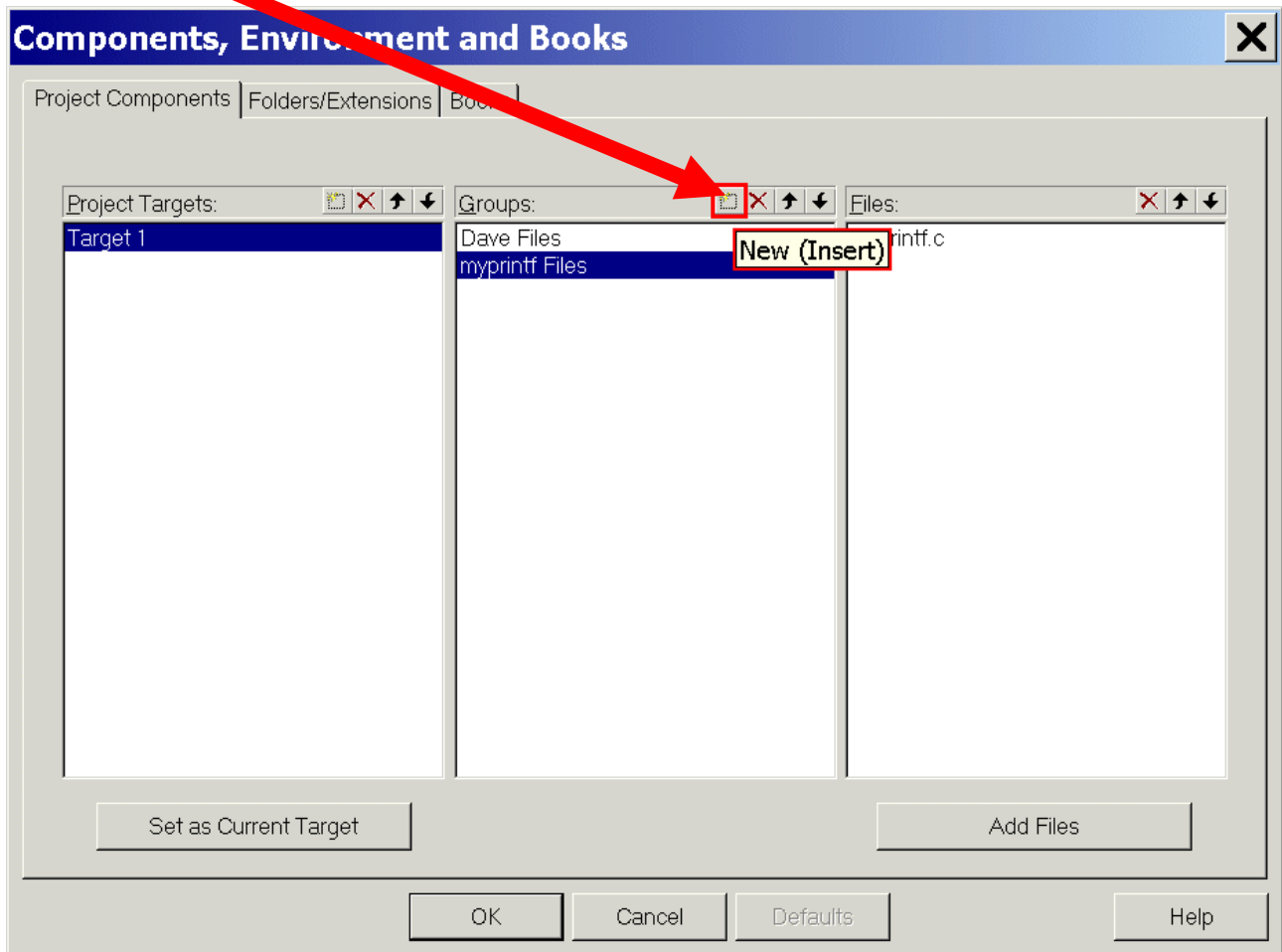
**Insert:** myprintf Files:



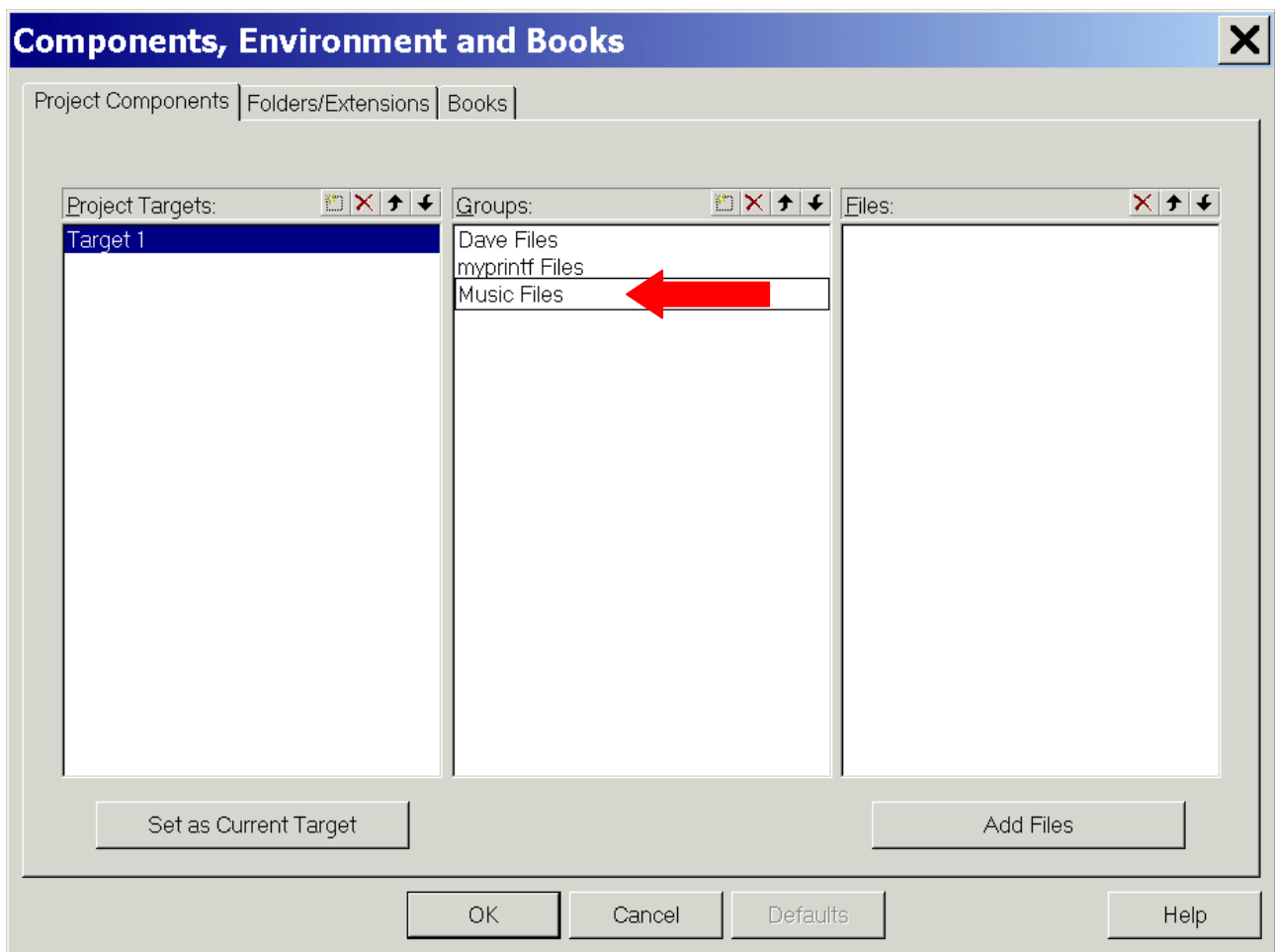
**<ENTER>**



Click New (Insert):

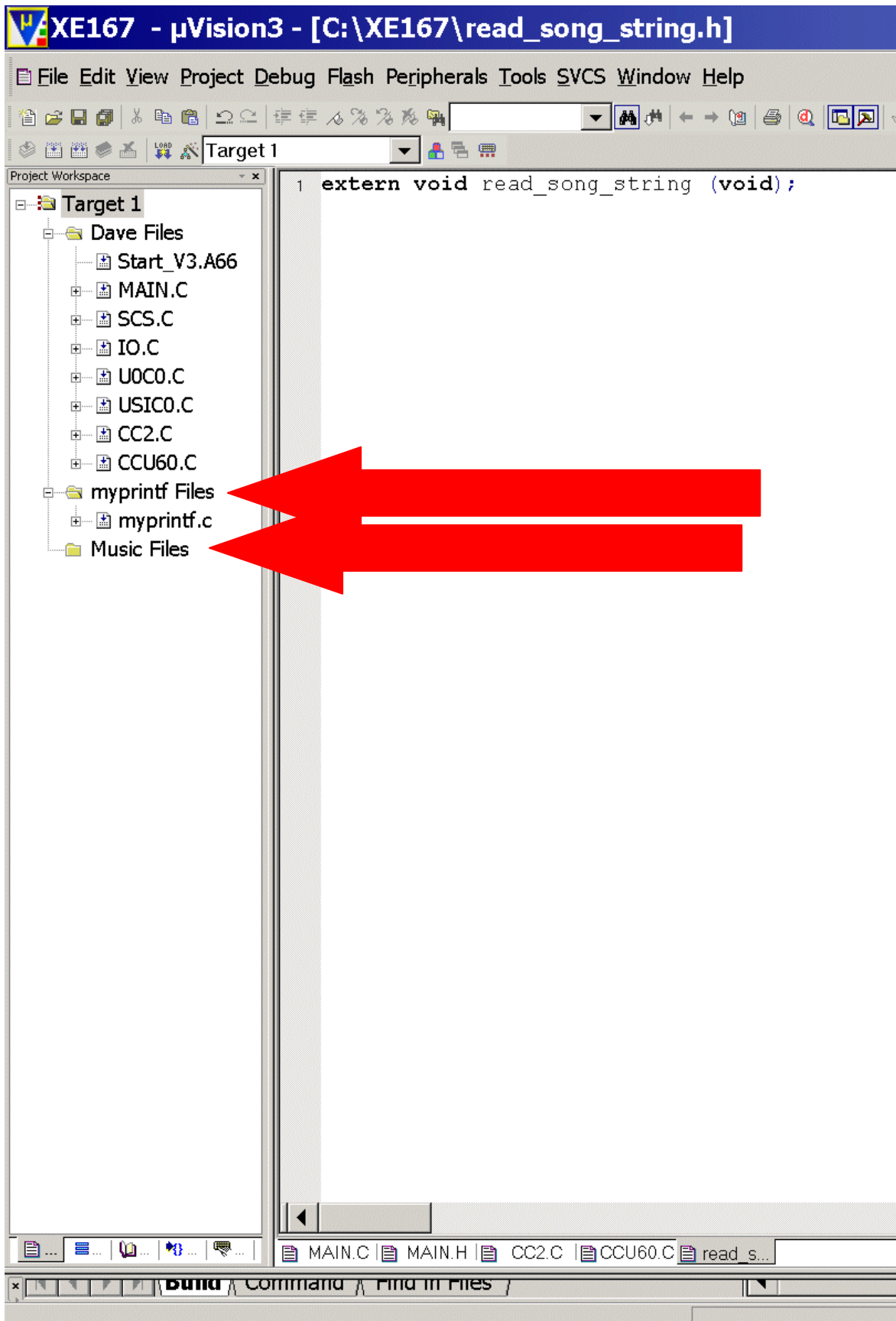


**Insert** Music Files:

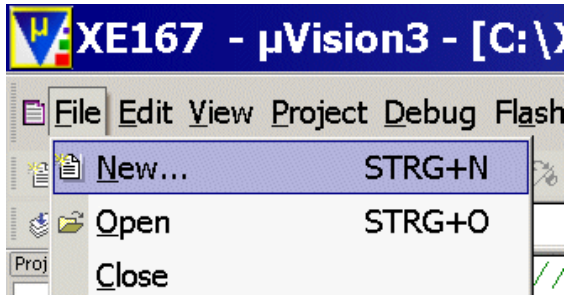


<ENTER>

OK



File – New



Insert:

```
#include "main.h"
#include "read_song_string.h"

void SetOctaveNORMAL(void)
{
    OctaveLOW = OFF; // clear Global Variable

    CCU60_vStopTmr(CCU60_TIMER_13); // Stop Timer 13

    CCU60_TCTR0 = CCU60_TCTR0 & 0xF8FF; // prescaler factor is 4, clear T13CLK
    CCU60_TCTR0 = CCU60_TCTR0 | 0x0200; // prescaler factor is 4, set T13CLK

    CCU60_vStartTmr(CCU60_TIMER_13); // Start Timer 13
}

void SetOctaveLOW(void)
{
    OctaveLOW = ON; // set Global Variable

    CCU60_vStopTmr(CCU60_TIMER_13); // Stop Timer 13

    CCU60_TCTR0 = CCU60_TCTR0 & 0xF8FF; // prescaler factor is 8, clear T13CLK
    CCU60_TCTR0 = CCU60_TCTR0 | 0x0300; // prescaler factor is 8, set T13CLK

    CCU60_vStartTmr(CCU60_TIMER_13); // Start Timer 13
}

void read_song_string (void)
{
    unsigned char substr[4]=0;
    current_note_length=old_note_length;

    switch (song[pos])
    {
```



```
// select note:
case 'C': note=0;
switch (song[++pos])
{
    case '+': note++; pos++; break;
    case '-': octave--;
                note=11;
                pos++; break;
    default : ; break;
} break;

case 'D': note=2;
switch (song[++pos])
{
    case '+': note++; pos++; break;
    case '-': note--; pos++; break;
    default : ; break;
} break;

case 'E': note=4;
switch (song[++pos])
{
    case '+': note++; pos++; break;
    case '-': note--; pos++; break;
    default : ; break;
} break;

case 'F': note=5;
switch (song[++pos])
{
    case '+': note++; pos++; break;
    case '-': note--; pos++; break;
    default : ; break;
} break;

case 'G': note=7;
switch (song[++pos])
{
    case '+': note++; pos++; break;
    case '-': note--; pos++; break;
    default : ; break;
} break;

case 'A': note=9;
switch (song[++pos])
{
    case '+': note++; pos++; break;
    case '-': note--; pos++; break;
    default : ; break;
}
```

```

}                break;

case 'H': note=11;
switch (song[++pos])
{
    case '+': octave++;
                note=0;
                pos++;                break;
    case '-': note--; pos++;        break;
    default : ;                break;
}                break;

```

// adjust note length:

```

case 'L': switch (song[++pos])
{
    case '1': if (song[++pos]=='6')
                current_note_length=length_of_a_whole_note/16;
                else
                {
                    pos--;
                    current_note_length=length_of_a_whole_note;
                }
                break;
    case '2': current_note_length=length_of_a_whole_note/2; break;
    case '4': current_note_length=length_of_a_whole_note/4; break;
    case '8': current_note_length=length_of_a_whole_note/8; break;
    default : ;                break;
}
old_note_length=current_note_length;
pos++;
read_song_string(); break;

```

// set rest:

```

case 'P': switch (song[++pos])
{
    case '1': if (song[++pos]=='6')
                current_note_length=length_of_a_whole_note/16;
                else
                {
                    pos--;
                    current_note_length=length_of_a_whole_note;
                }
                break;
    case '2':current_note_length=length_of_a_whole_note/2; break;
    case '4':current_note_length=length_of_a_whole_note/4; break;
    case '8':current_note_length=length_of_a_whole_note/8; break;
    default : ;                break;
}

```

```
        note=12;
        pos++;
        break;

// adjust octave:
case 'O': switch (song[++pos])
{
    case '0': octave=1; break;
    case '1': octave=2; break;
    case '2': octave=4; break;
    case '3': octave=8; break;
    default : if (song[pos]=='L') octave=1, SetOctaveLOW();
               if (song[pos]=='N') octave=1, SetOctaveNORMAL();
               break;
}
pos++;
read_song_string(); break;

// tempo:
case 'T': pos++;
        substr[3]=0; //string termination
        if (song[pos]=='1')
        {
            substr[0]=song[pos];
            substr[1]=song[++pos];
            substr[2]=song[++pos];
        }
        else
        {
            substr[0]=song[pos];
            substr[1]=song[++pos];
            substr[2]=' ';
        }
        tempo=atoi(substr);
        pos++;
        read_song_string(); break;

        default: ; break;
} /* end case */

// extend note length by half:
if (song[pos]=='.')
{
    old_note_length=current_note_length;
    current_note_length=current_note_length*3.0/2.0;
    pos++;
}

if (pos==max) pos++;
```

```
} /* end read_song_string */
```





**XE167 - µVision3 - [C:\XE167\read\_song\_string.c]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - UOC0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
    - myprintf Files
      - myprintf.c
    - Music Files

```

001 #include "main.h"
002 #include "read_song_string.h"
003
004 void SetOctaveNORMAL(void)
005 {
006     OctaveLOW = OFF; // clear Global Variable
007
008     CCU60_vStopTmr(CCU60_TIMER_13); // Stop Timer 13
009
010     CCU60_TCTR0 = CCU60_TCTR0 & 0xF8FF; // prescaler factor is 4, clear T13CLK
011     CCU60_TCTR0 = CCU60_TCTR0 | 0x0200; // prescaler factor is 4, set T13CLK
012
013     CCU60_vStartTmr(CCU60_TIMER_13); // Start Timer 13
014 }
015
016 void SetOctaveLOW(void)
017 {
018     OctaveLOW = ON; // set Global Variable
019
020     CCU60_vStopTmr(CCU60_TIMER_13); // Stop Timer 13
021
022     CCU60_TCTR0 = CCU60_TCTR0 & 0xF8FF; // prescaler factor is 8, clear T13CLK
023     CCU60_TCTR0 = CCU60_TCTR0 | 0x0300; // prescaler factor is 8, set T13CLK
024
025     CCU60_vStartTmr(CCU60_TIMER_13); // Start Timer 13
026 }
027

```

```

028 void read_song_string (void)
029 {
030     unsigned char substr[4]=0;
031     current_note_length=old_note_length;
032
033     switch (song[pos])
034     {
035         // select note:
036         case 'C': note=0;
037             switch (song[++pos])
038             {
039                 case '+': note++; pos++; break;
040                 case '-': octave--;
041                     note=11;
042                     pos++; break;
043                 default : ; break;
044             } break;
045
046         case 'D': note=2;
047             switch (song[++pos])
048             {
049                 case '+': note++; pos++; break;
050                 case '-': note--; pos++; break;
051                 default: ; break;
052             } break;
053
054         case 'E': note=4;
055             switch (song[++pos])
056             {
057                 case '+': note++; pos++; break;
058                 case '-': note--; pos++; break;
059                 default : ; break;
060             } break;
061
062         case 'F': note=5;
063             switch (song[++pos])
064             {
065                 case '+': note++; pos++; break;
066                 case '-': note--; pos++; break;
067                 default : ; break;
068             } break;
069
070         case 'G': note=7;
071             switch (song[++pos])
072             {
073                 case '+': note++; pos++; break;
074                 case '-': note--; pos++; break;
075                 default : ; break;
076             } break;
077
078         case 'A': note=9;
079             switch (song[++pos])
080             {
081                 case '+': note++; pos++; break;
082                 case '-': note--; pos++; break;
083                 default : ; break;
084             } break;
085
086         case 'H': note=11;
087             switch (song[++pos])
088             {
089                 case '+': octave++;
090                     note=0;
091                     pos++; break;
092                 case '-': note--; pos++; break;
093                 default : ; break;
094             } break;
095

```

```

096 // adjust note length:
097 case 'L': switch (song[++pos])
098 {
099     case '1': if (song[++pos]=='6')
100         current_note_length=length_of_a_whole_note/16;
101     else
102     {
103         pos--;
104         current_note_length=length_of_a_whole_note;
105     }
106
107     case '2': current_note_length=length_of_a_whole_note/2; break;
108     case '4': current_note_length=length_of_a_whole_note/4; break;
109     case '8': current_note_length=length_of_a_whole_note/8; break;
110     default : ; break;
111 }
112 old_note_length=current_note_length;
113 pos++;
114 read_song_string(); break;
115
116
117 // set rest:
118 case 'P': switch (song[++pos])
119 {
120     case '1': if (song[++pos]=='6')
121         current_note_length=length_of_a_whole_note/16;
122     else
123     {
124         pos--;
125         current_note_length=length_of_a_whole_note;
126     }
127
128     case '2':current_note_length=length_of_a_whole_note/2; break;
129     case '4':current_note_length=length_of_a_whole_note/4; break;
130     case '8':current_note_length=length_of_a_whole_note/8; break;
131     default : ; break;
132 }
133 note=12;
134 pos++; break;
135

```

```

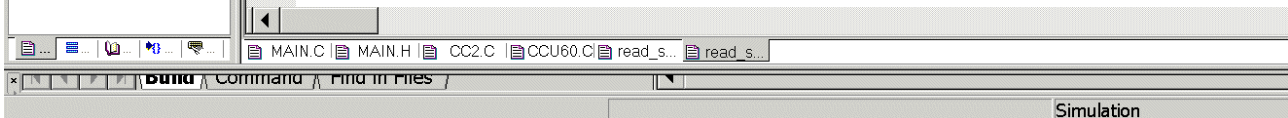
136 // adjust octave:
137 case 'O': switch (song[++pos])
138 {
139     case '0': octave=1; break;
140     case '1': octave=2; break;
141     case '2': octave=4; break;
142     case '3': octave=8; break;
143     default : if (song[pos]=='L') octave=1, SetOctaveLOW();
144               if (song[pos]=='N') octave=1, SetOctaveNORMAL();
145               break;
146 }
147 pos++;
148 read_song_string();    break;
149
150 // tempo:
151 case 'T': pos++;
152     substr[3]=0; //string termination
153     if (song[pos]=='1')
154     {
155         substr[0]=song[pos];
156         substr[1]=song[++pos];
157         substr[2]=song[++pos];
158     }
159     else
160     {
161         substr[0]=song[pos];
162         substr[1]=song[++pos];
163         substr[2]=' ';
164     }
165     tempo=atoi(substr);
166     pos++;
167     read_song_string();    break;
168
169     default: ;    break;
170 } /* end case */
171

```

```

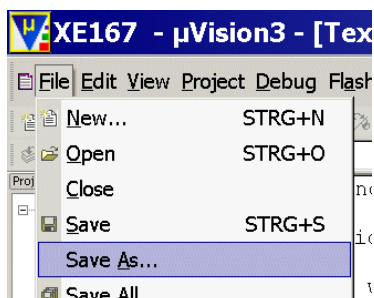
172 // extend note length by half:
173 if (song[pos]=='.')
174 {
175     old_note_length=current_note_length;
176     current_note_length=current_note_length*3.0/2.0;
177     pos++;
178 }
179
180 if (pos==max) pos++;
181
182 } /* end read_song_string */
183

```

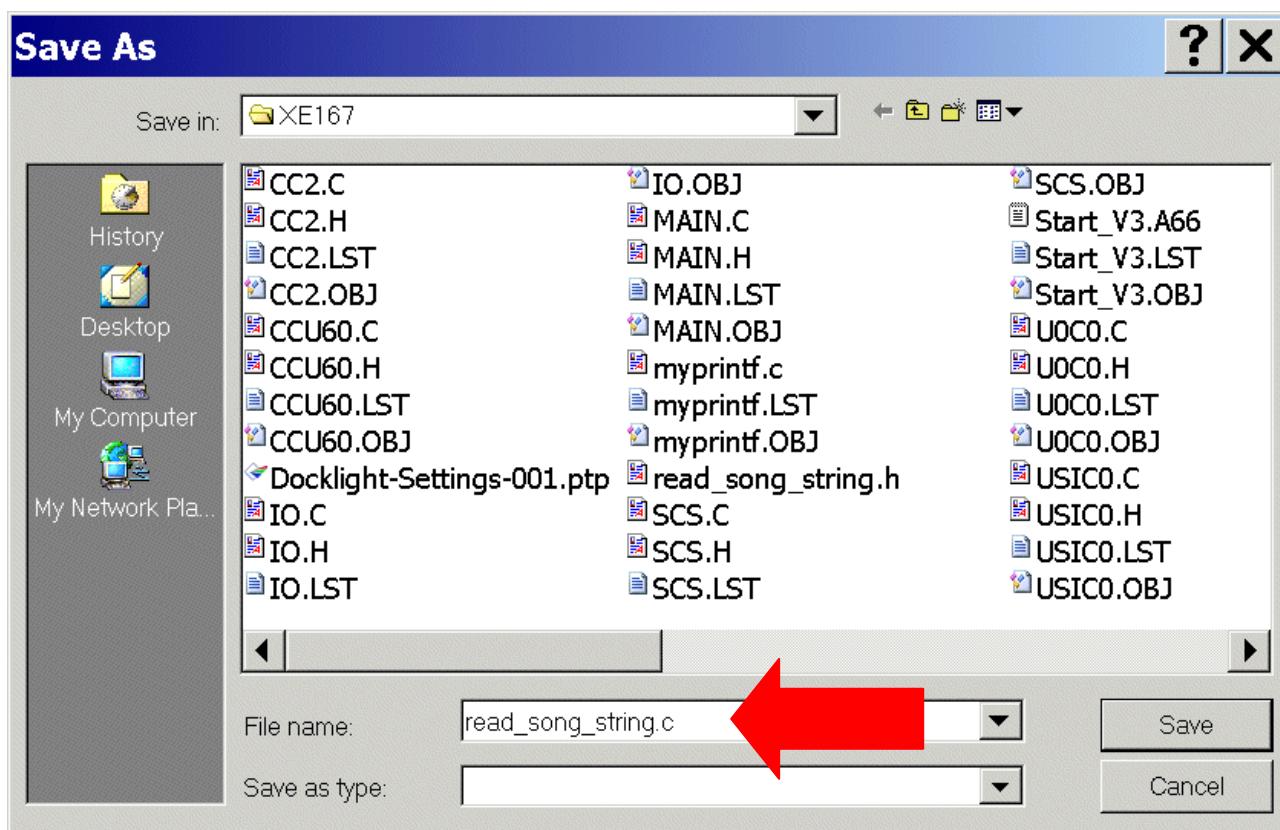




File – Save As...

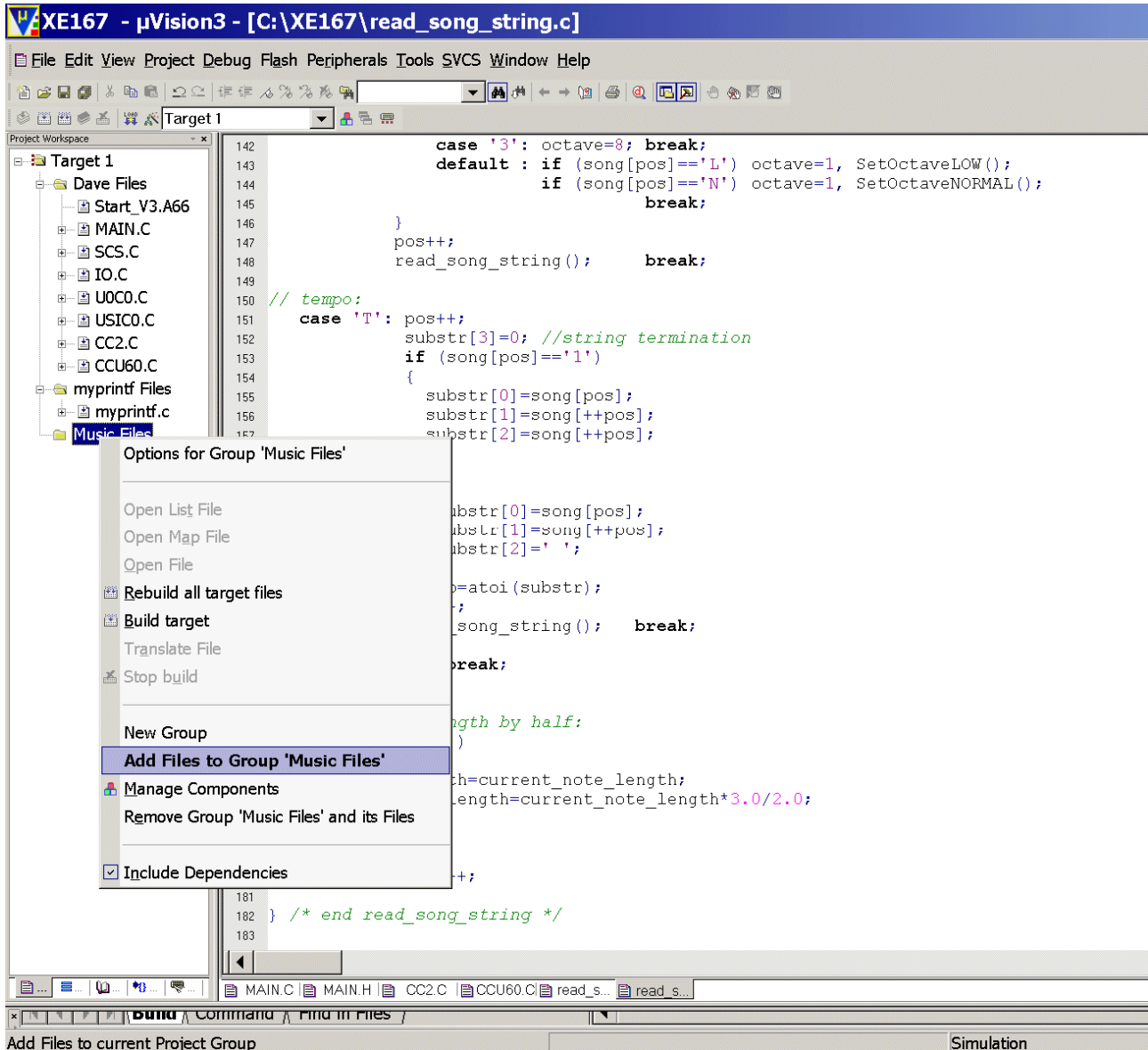


Insert: read\_song\_string.c



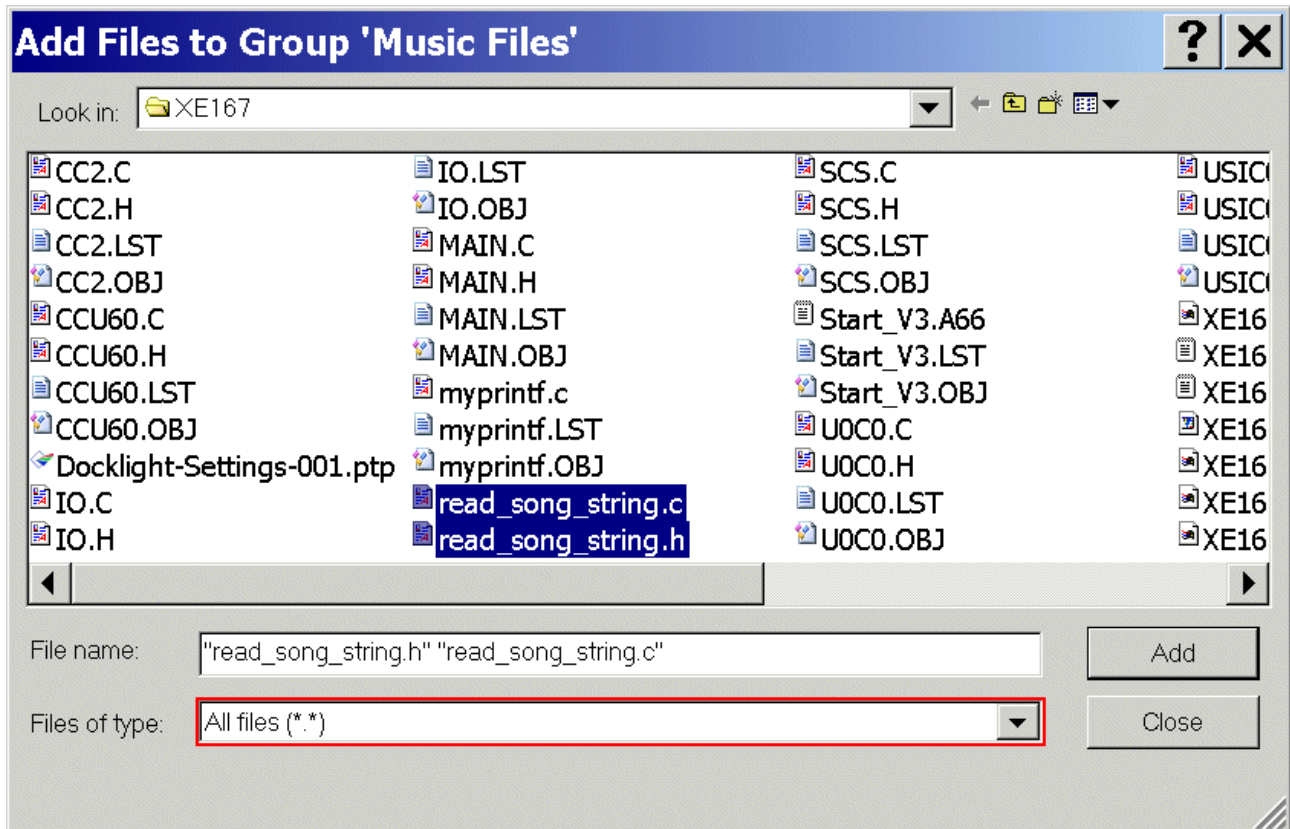
Click Save

Mouse position: **Project Window**, Music Files: **click right mouse button**  
**Click** Add Files to Group 'Music Files'



Files of type: **select** All files

**Mark** read\_song\_string.h **and** read\_song\_string.c



**Click** Add

**Click** Close

XE167 - µVision3 - [C:\XE167\read\_song\_string.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU60.C
  - myprintf Files
    - myprintf.c
  - Music Files
    - read\_song\_string.h
    - read\_song\_string.c

```

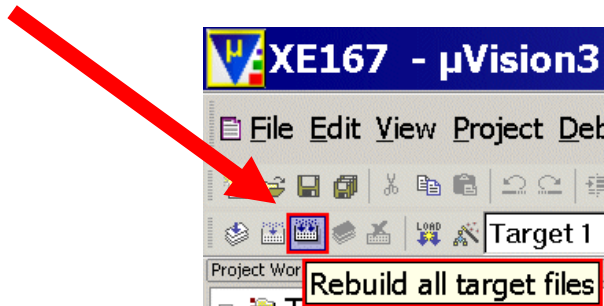
142         case '3': octave=8; break;
143         default : if (song[pos]=='L') octave=1, SetOctaveLOW();
144                   if (song[pos]=='N') octave=1, SetOctaveNORMAL();
145                   break;
146     }
147     pos++;
148     read_song_string();    break;
149
150 // tempo:
151 case 'T': pos++;
152           substr[3]=0; //string termination
153           if (song[pos]=='1')
154           {
155               substr[0]=song[pos];
156               substr[1]=song[++pos];
157               substr[2]=song[++pos];
158           }
159           else
160           {
161               substr[0]=song[pos];
162               substr[1]=song[++pos];
163               substr[2]=' ';
164           }
165           tempo=atoi(substr);
166           pos++;
167           read_song_string();    break;
168
169     default: ;    break;
170 } /* end case */
171
172 // extend note length by half:
173 if (song[pos]=='.')
174 {
175     old_note_length=current_note_length;
176     current_note_length=current_note_length*3.0/2.0;
177     pos++;
178 }
179
180 if (pos==max) pos++;
181
182 } /* end read_song_string */
183

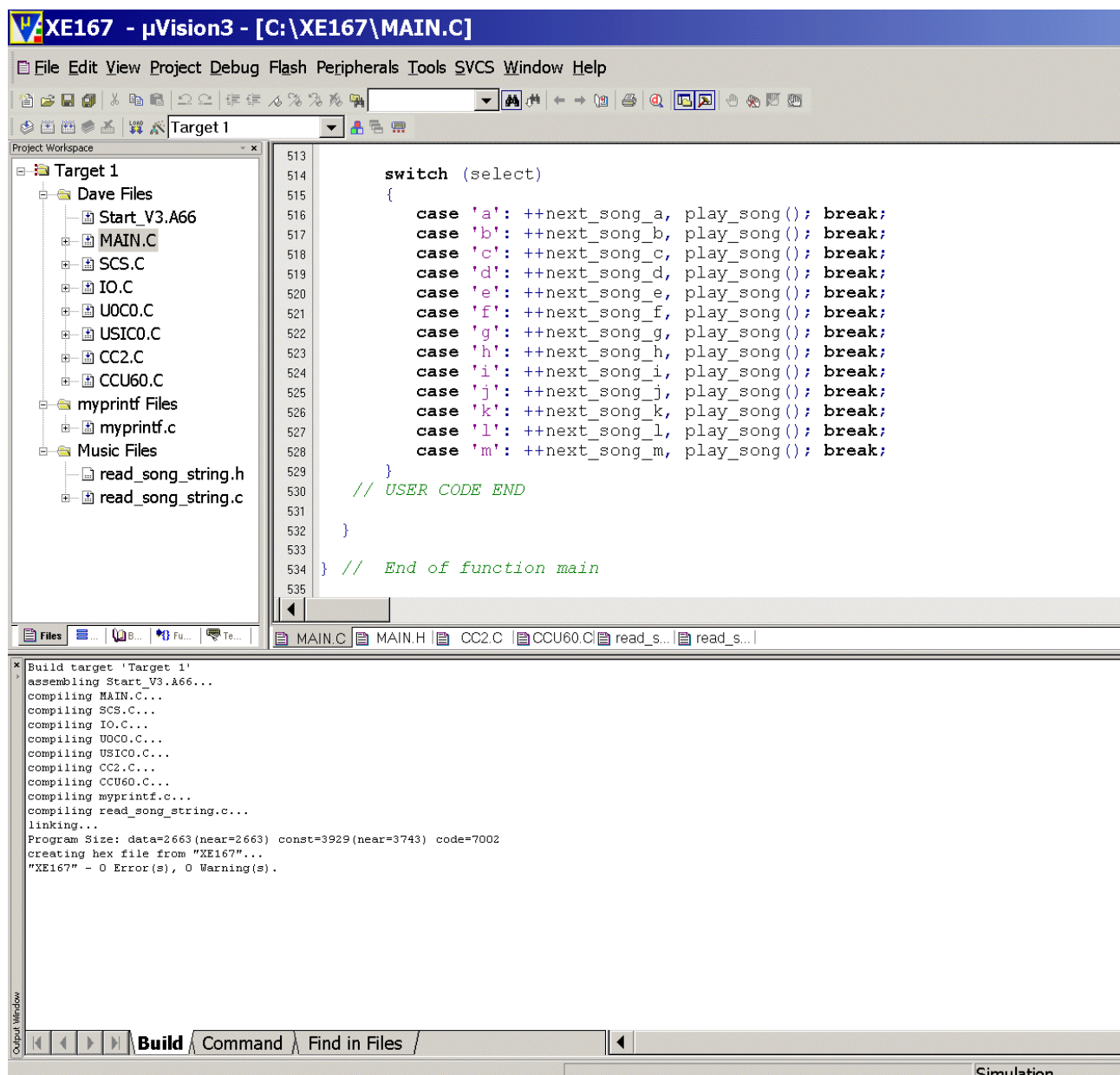
```

Files | Build | Command | Find in Files | Simulation



Generate your application program:

<p>Project – Rebuild all target files</p>	<p>or click</p> 
---	--



The screenshot shows the XE167 - µVision3 IDE interface. The title bar indicates the project is located at [C:\XE167\MAIN.C]. The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and project management. The Project Workspace pane on the left shows a tree view of the project files, including Dave Files, Start\_V3.A66, MAIN.C, SCS.C, IO.C, U0C0.C, USIC0.C, CC2.C, CCU60.C, myprintf Files, myprintf.c, and Music Files (read\_song\_string.h, read\_song\_string.c). The main editor window displays the source code for MAIN.C, which includes a switch statement for playing songs. The Output window at the bottom shows the build process for Target 1, including compilation of various files and the final hex file creation.

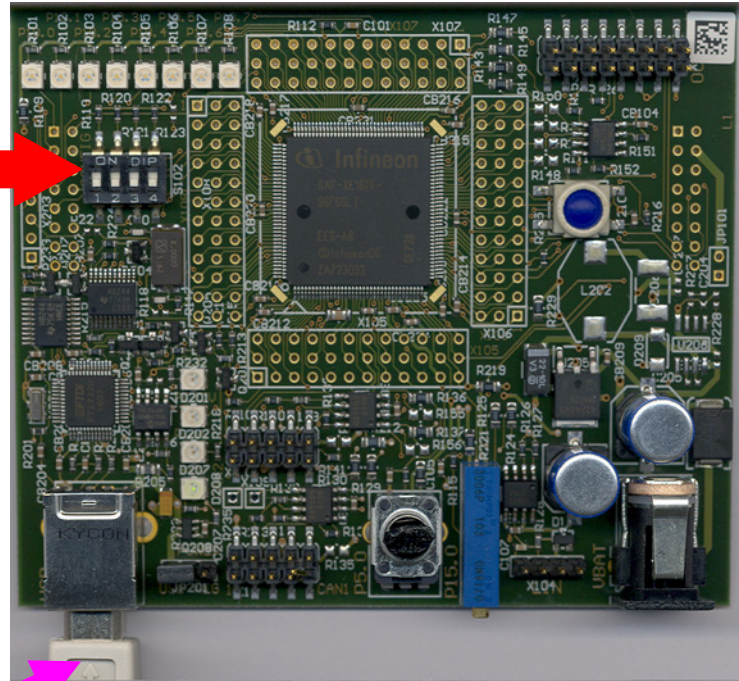
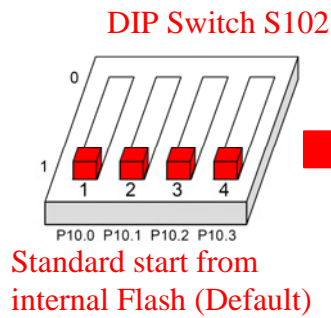
```

513
514
515      switch (select)
516      {
517          case 'a': ++next_song_a, play_song(); break;
518          case 'b': ++next_song_b, play_song(); break;
519          case 'c': ++next_song_c, play_song(); break;
520          case 'd': ++next_song_d, play_song(); break;
521          case 'e': ++next_song_e, play_song(); break;
522          case 'f': ++next_song_f, play_song(); break;
523          case 'g': ++next_song_g, play_song(); break;
524          case 'h': ++next_song_h, play_song(); break;
525          case 'i': ++next_song_i, play_song(); break;
526          case 'j': ++next_song_j, play_song(); break;
527          case 'k': ++next_song_k, play_song(); break;
528          case 'l': ++next_song_l, play_song(); break;
529          case 'm': ++next_song_m, play_song(); break;
530      }
531      // USER CODE END
532
533
534 } // End of function main
535

```

Build target 'Target 1'  
 assembling Start\_V3.A66...  
 compiling MAIN.C...  
 compiling SCS.C...  
 compiling IO.C...  
 compiling U0C0.C...  
 compiling USIC0.C...  
 compiling CC2.C...  
 compiling CCU60.C...  
 compiling myprintf.c...  
 compiling read\_song\_string.c...  
 linking...  
 Program Size: data=2663 (near=2663) const=3929 (near=3743) code=7002  
 creating hex file from "XE167"...  
 "XE167" - 0 Error(s), 0 Warning(s).

Connect the XE167 Easy Kit Board to the host computer:



**USB Cable:**

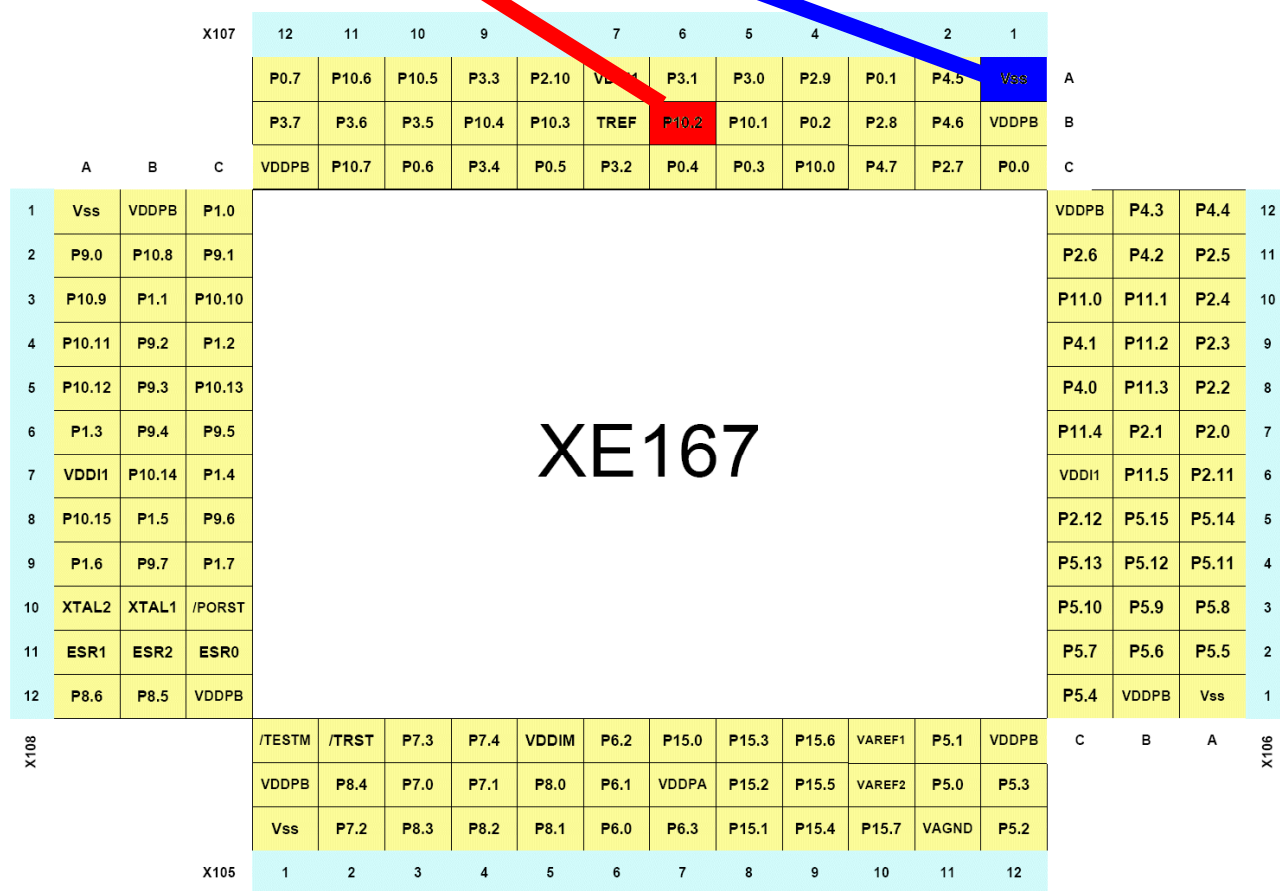
.) used for: UART communication (the UART/RS232/ASC0/USIC0\_CH0 serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).

.) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).

.) the USB connection works also as the power supply.

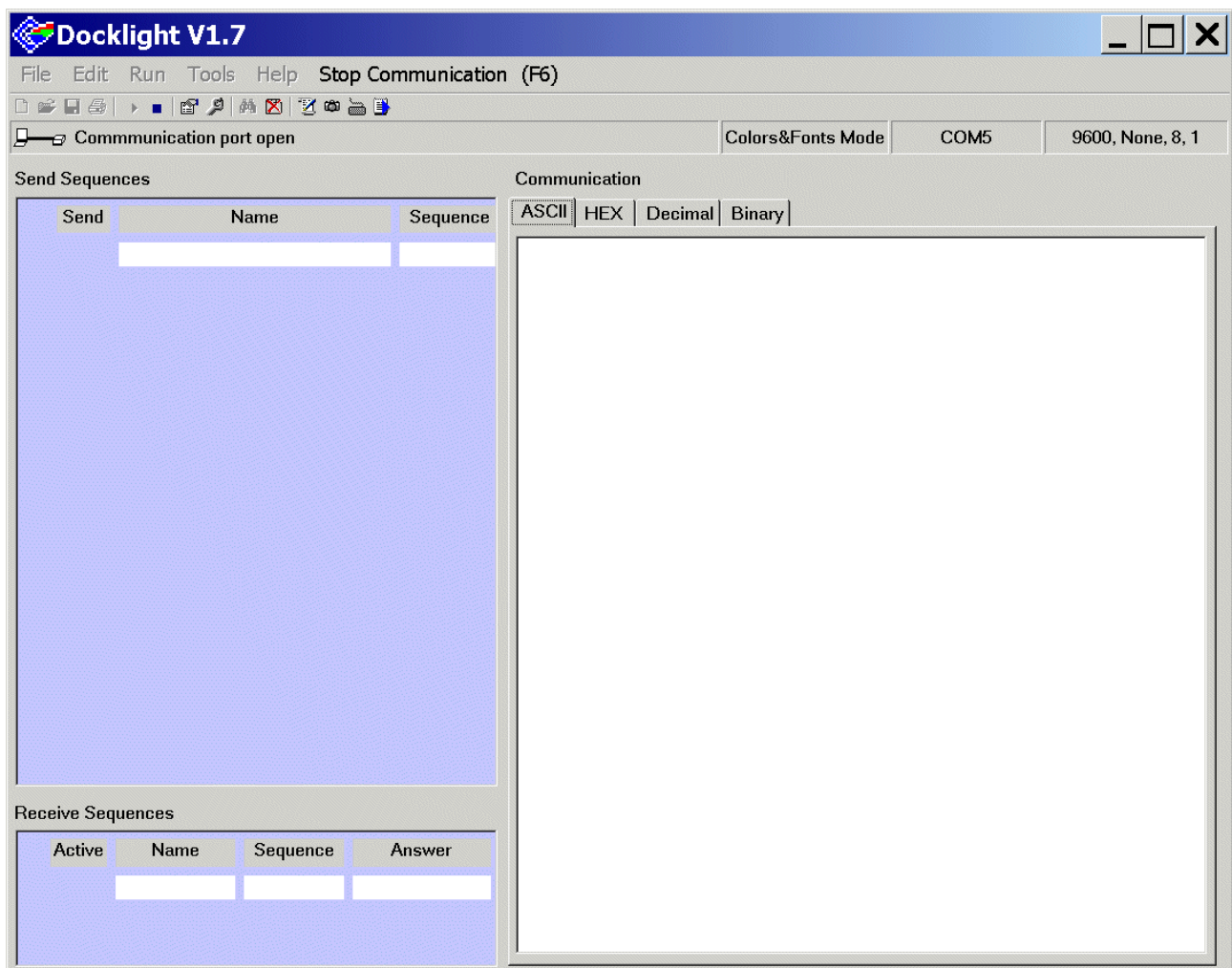
Connect your active loudspeaker(s) to the XE167 Easy Kit Board:

Connect CCU60\_CC62 (P10.2) and GND (VSS) to your active loudspeaker(s):






Now, **start** Docklight and **click**  (Start Communication):

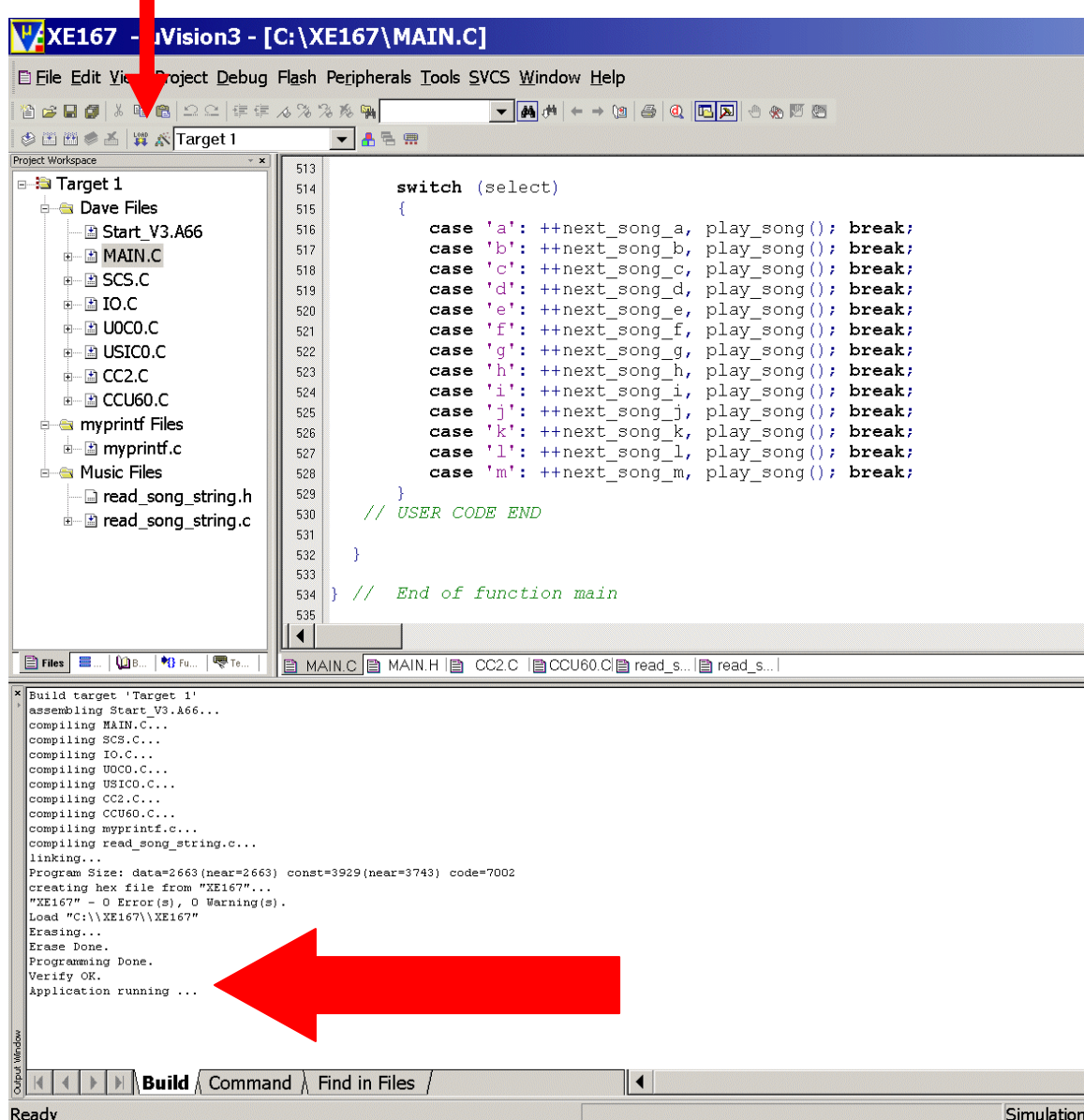






Go to  $\mu$ Vision, [Download/Program](#) your application program into the On Chip Flash:

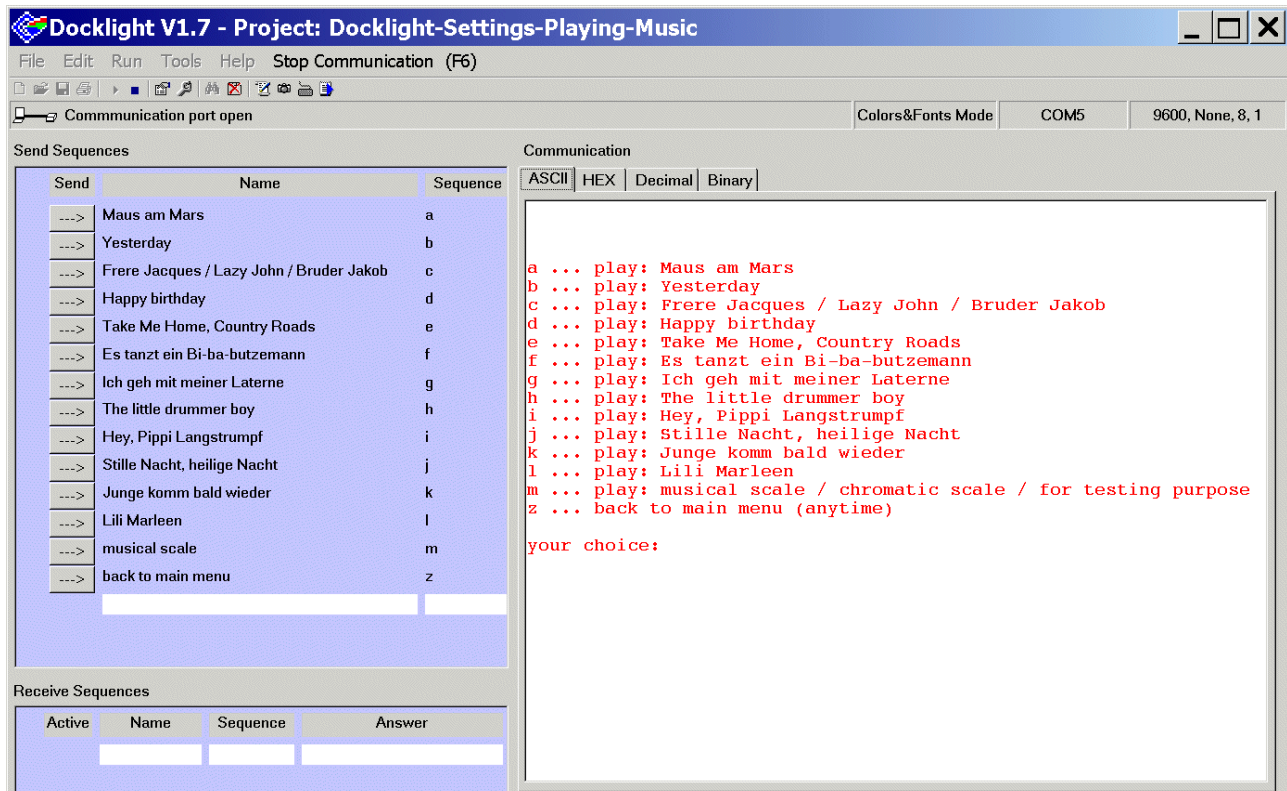
Click: 



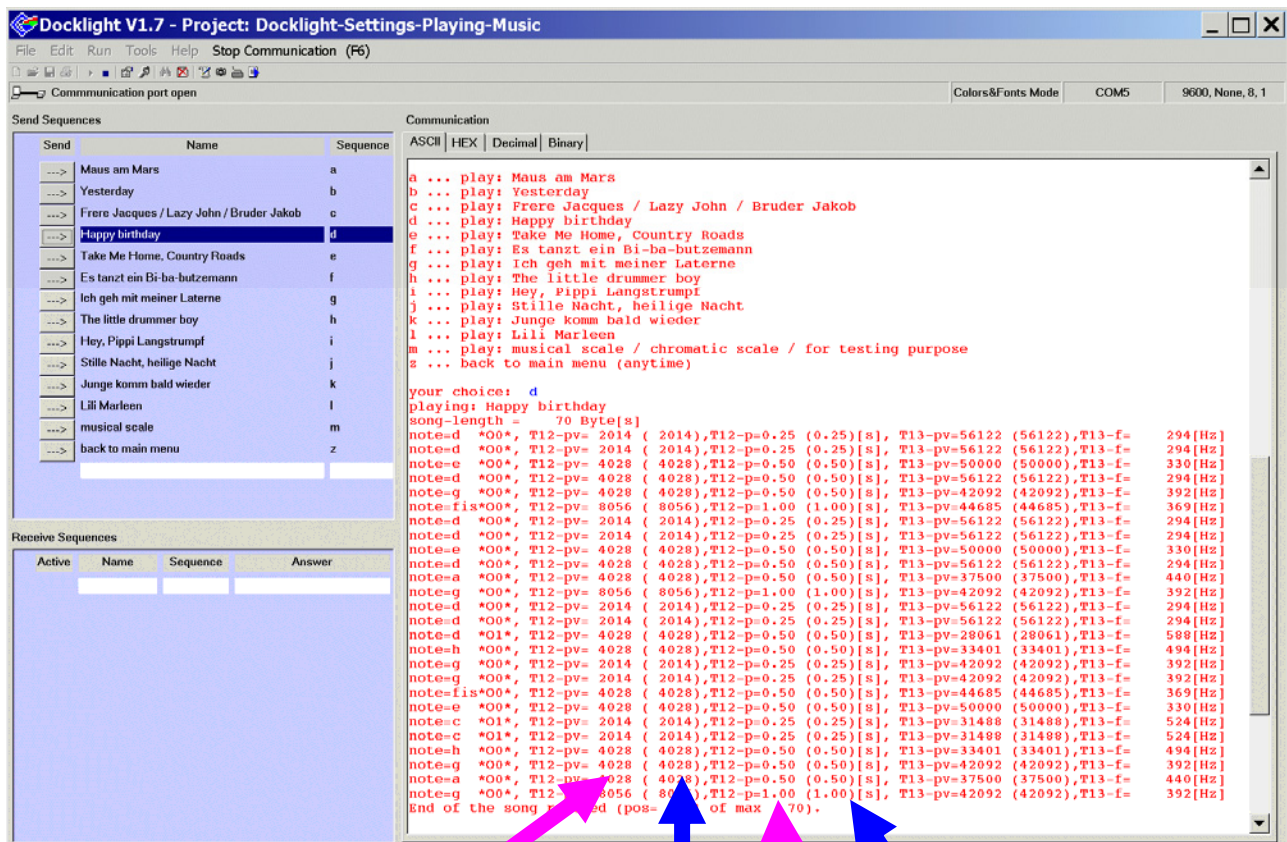
The screenshot shows the  $\mu$ Vision3 IDE interface. The top menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, project management, and debugging. The Project Workspace on the left shows a tree view for 'Target 1' with folders for 'Dave Files', 'myprintf Files', and 'Music Files'. The main editor displays the code for 'MAIN.C', which includes a switch statement for playing songs. The bottom status bar shows 'Ready' and 'Simulation'. The Output Window at the bottom displays the build process, including compilation and linking steps, and a large red arrow points to the 'Application running ...' status.



Go to Docklight and see / hear / enjoy the result:



Insert/select d

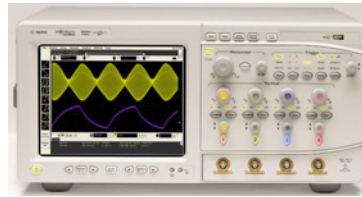


T12 period value  
@ tempo = 120  
beats/minute

T12 period value  
@ current tempo

note length  
@ tempo = 120  
beats/minute

Note length  
@ current tempo



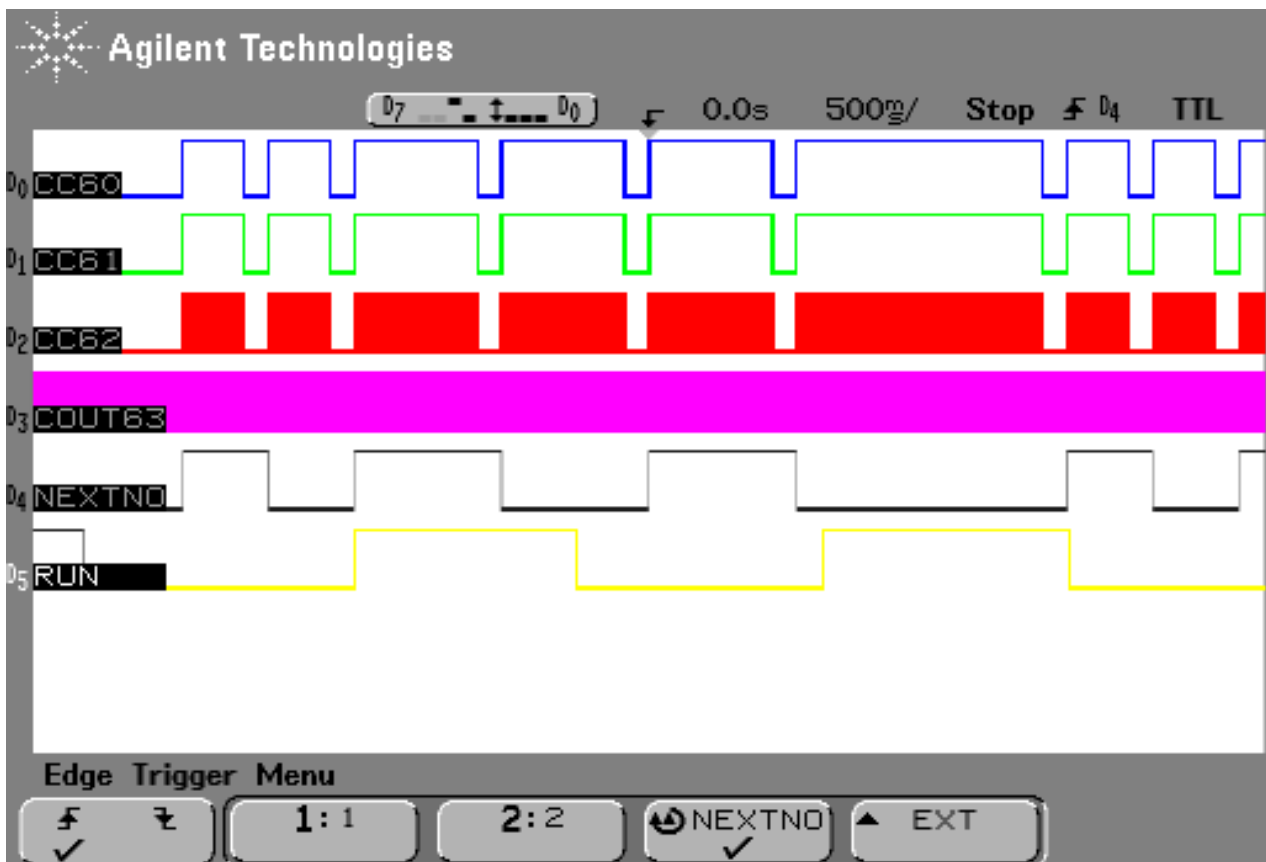
Use any **Logic Analyser (LGA)** / scope and see the result:

Note:

Song d: Happy birthday:

code unsigned char

```
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C  
CO0L4HGAL2G";
```

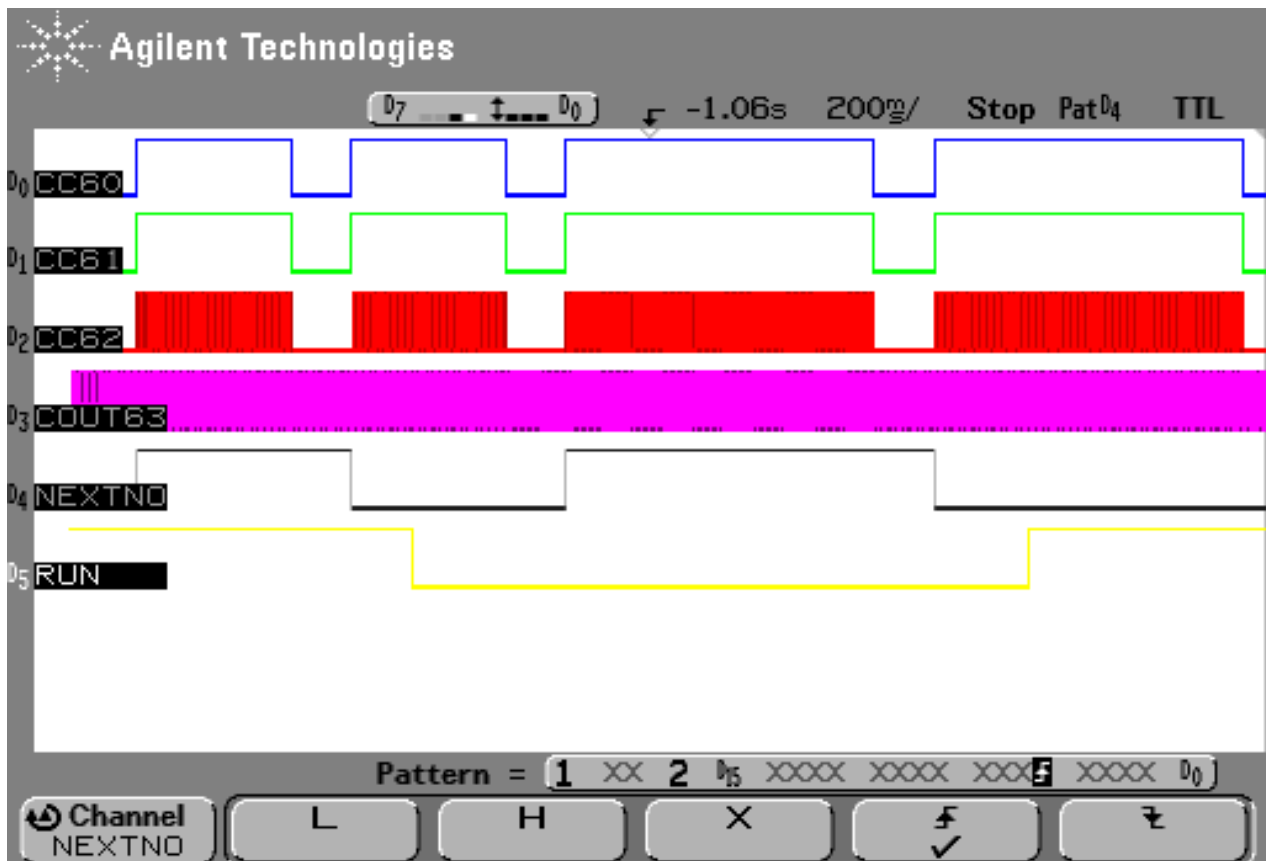






HAPPY BIRTHDAY =

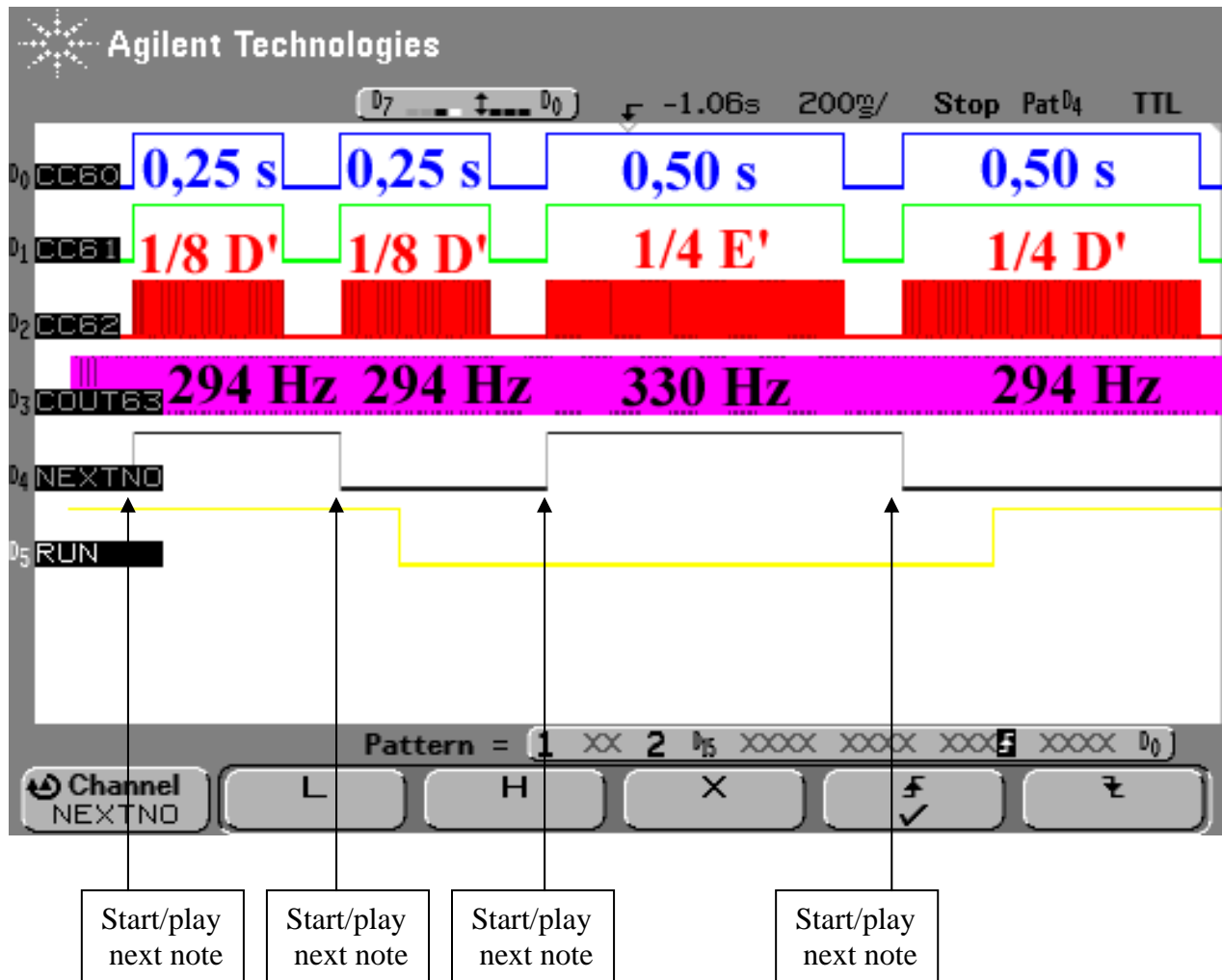
T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F... ..





HAPPY BIRTHDAY =

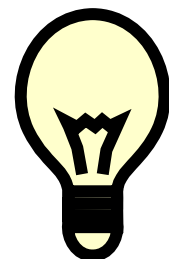
T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F... ..



Note:

Start/play next note:

P10\_OMRL = 0x1818; // Show start of next note on Port 10 Pin 3 and/or Pin10.4  
CCU60\_vStartTmr(CCU60\_TIMER\_12); // Start next note (T12 single shot)



## 2.) Appendix: about music (note length and note frequency)






Syntax used in our programming example:

Lx : Change note length

(x = 1,2,4,8,16 -> 1=whole-note, 2=half-note, 4=quarter-note, 8=Eighth-note, 16=16th-note)

Real Music:




note	LENGTH
	1/1 Whole Note (Semi-breve) (4 beats)
	1/2 Half-note (Minim) (2 beats)
	1/4 Quarter-note (Crotchet) (1 beat)
	1/8 Eighth-note (Quaver) (1/2 beat)
	1/16 Sixteenth-note/16th-note (Semiquaver) (1/4 beat)

Syntax used in our programming example:

. : Extend preceding note by half of its value

Real Music:

note	LENGTH
	$\frac{1}{2} \text{ (2 beats)} + (\frac{1}{2})/2 \text{ (1 beat)} = \frac{3}{4} \text{ (3 beats)}$

**Note:**

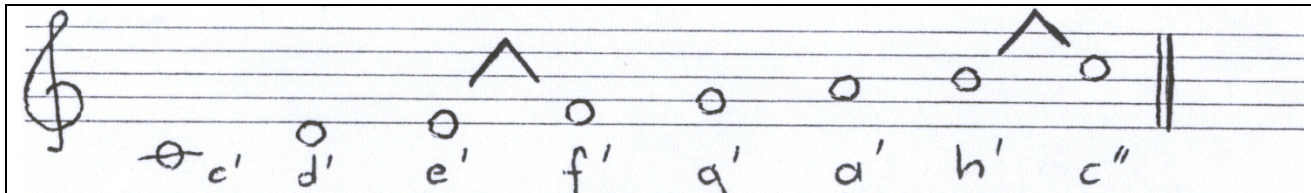
The . extends the length of the note by half of its length.



Syntax used in our programming example:

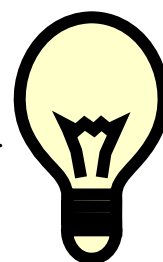
C,D,E,F,G,A,H: play note

Real Music:



**Note:**

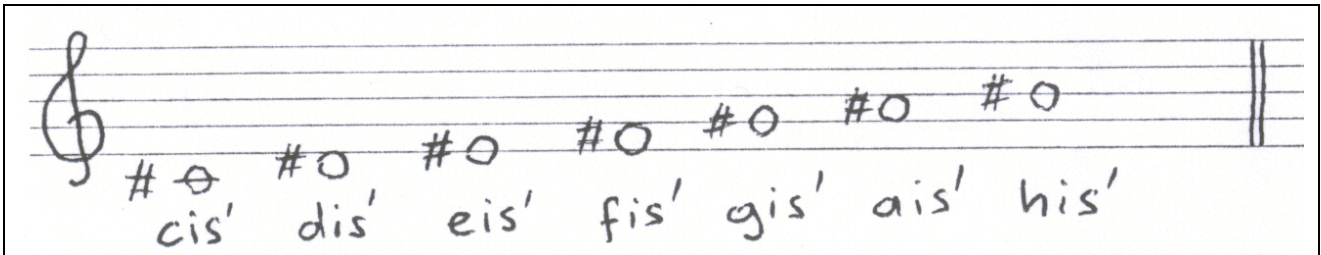
The notes C, D, E, F, G, A, H are named C, D, E, F, G, A, B in other countries.  
In this document we stick to the German names.



Syntax used in our programming example:

+: The + (Sharp) raises its note (frequency) a semitone: Cis, Dis, Eis, Fis, Gis, Ais, His

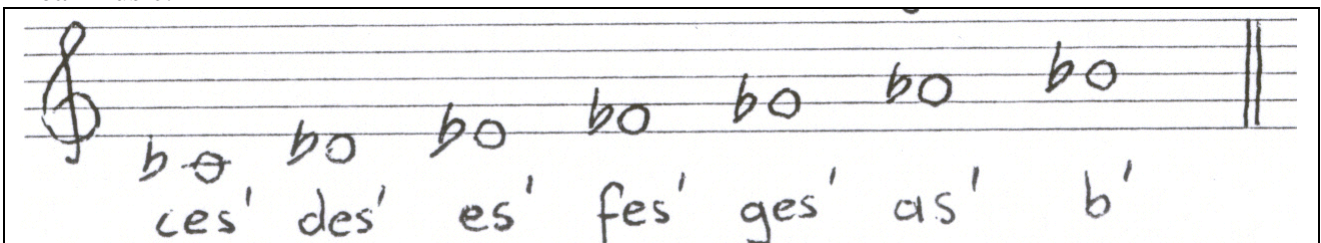
Real Music:



Syntax used in our programming example:

-: The - (Flat) lowers its note (frequency) a semitone: Ces, Des, Es, Fes, Ges, As, Hes

Real Music:













Syntax used in our programming example:

Px : play rest/pause/interval of silence

(x = 1,2,4,8,16 -> 1=whole-rest, 2=half-rest, 4=quarter-rest, 8=Eighth-rest, 16=16th-rest)

Real Music:

rest	rest	LENGTH
		1/1 Whole Rest (4 beats)
		1/2 Half-rest (2 beats)
		1/4 Quarter-rest (1 beat)
		1/8 Eighth-rest (1/2 beat)
		1/16 Sixteenth-rest/16th-rest (1/4 beat)

**Note:**

The realisation of our programming example is easier when we deal with rests as notes.

Therefore, playing a rest means playing a note.

The frequency of the note which is a rest was chosen above our hearing threshold level (e.g. 60.000 Hz).

Octave:

**Definition:**

In music, an octave is the interval between one musical note and another with half or double its frequency.

**Note:**

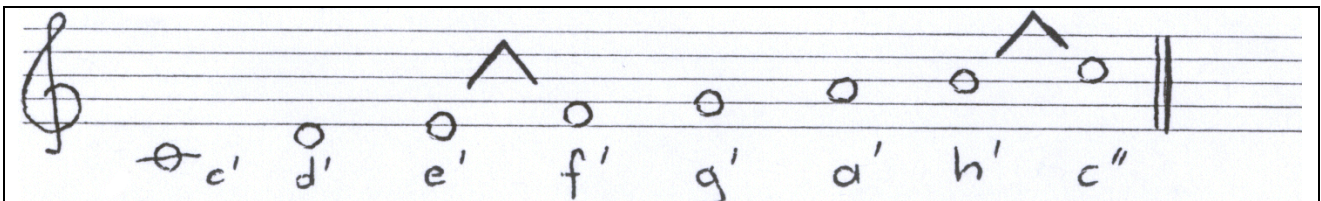
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

Further octaves of a note occur at  $2^n$  times the frequency of that note (where  $n$  is an integer, such as 2, 4, 8, 16 ...).

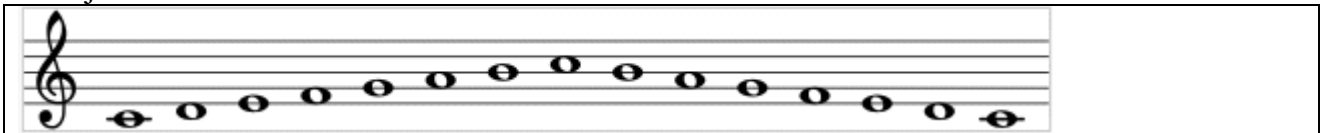
Syntax used in our programming example:

Ox : change octave (x = 0,1,2,3)

Real Music:



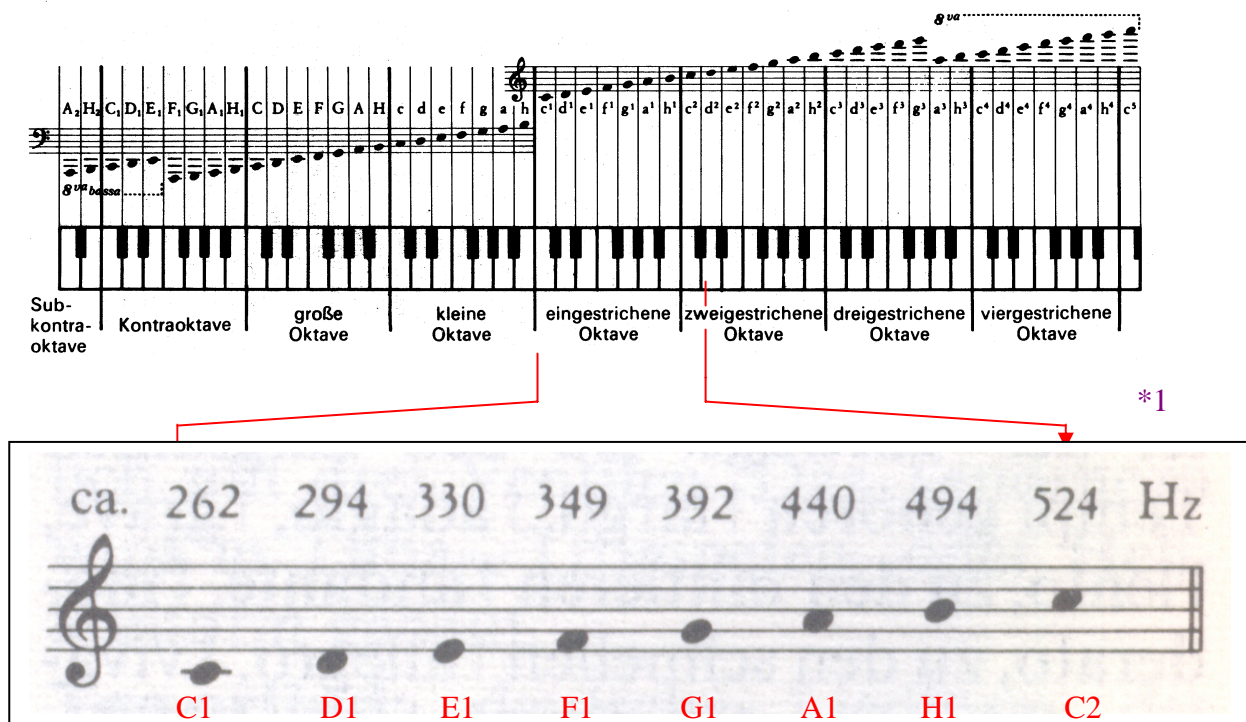
C major scale:





### 3.) Appendix: CCU6 use to create note length and note frequency

If note a' is equal to 440 Hz then we get the following frequencies for the musical scale:



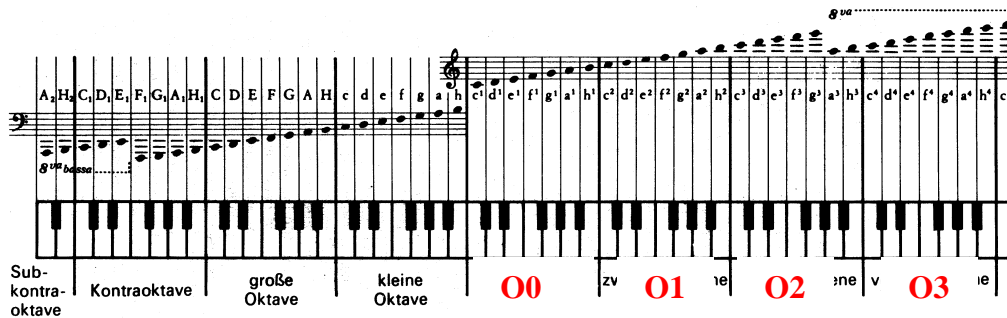
\*2

frequency	note
264 Hz	C1
297 Hz	D1
330 Hz	E1
352 Hz	F1
396 Hz	G1
440 Hz	A1
495 Hz	B1/H1
528 Hz	C2

\*1: frequency/note: source: Schüler Duden, Die Musik

\*2: frequency/note: source: <http://de.wikipedia.org/wiki/Tonleiter>

Note – frequency (Timer 13), **octave = O0, O1, O2 and O3:**



In our programming example we are going to use the following period-values for Timer 13:

```
unsigned int T13_values[] =
{62977, 59550, 56122, 53061, 50000, 47278, 44685, 42092, 39796, 37500, 35450, 33401, 275};
/*
[0]=c',[1]=cis',[2]=d',[3]=dis',[4]=e',[5]=f',[6]=fis',[7]=g',[8]=gis',[9]=a',[10]=ais',[11]=h',
[12]=<Frequency for rest>
*/
```

So we get the following values shown in the table below

**Note:** Timer 13 resolution =  $1/(f_{clk}/4) = 1/(66MHz/4) = 60,606 \text{ ns}$ :

		Octave=0 (=') scaler for T13- Period- value =1	Octave=1 (='') scaler for T13- Period- value =2	Octave=2 (=''') scaler for T13- Period- value =4	Octave=3 (=''''') scaler for T13- Period- value =8
T13 period values	note	f [Hz]	f [Hz]	f [Hz]	f [Hz]
T13_values[ 0] = 62977	c'	262	523		
T13_values[ 1] = 59550	cis'				
T13_values[ 2] = 56122	d'	294			
T13_values[ 3] = 53061	dis'				
T13_values[ 4] = 50000	e'	330			
T13_values[ 5] = 47278	f'	349			
T13_values[ 6] = 44685	fis'				
T13_values[ 7] = 42092	g'	392			
T13_values[ 8] = 39796	gis'				
T13_values[ 9] = 37500	a'	440	880	1760	3520
T13_values[10] = 35450	ais'				
T13_values[11] = 33401	h'	494	988		
T13_values[12] = 275	----	60000			

**Note:**

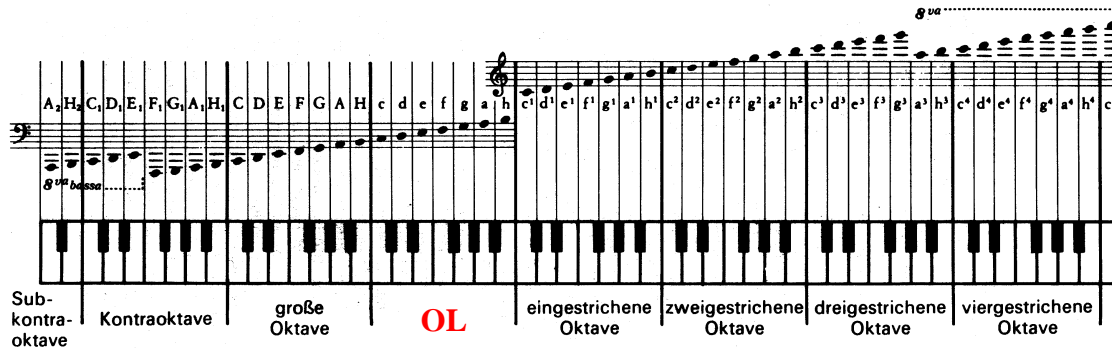
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

Further octaves of a note occur at  $2^n$  times the frequency of that note (where  $n$  is an integer, such as 2, 4, 8, 16 ...).

e.g. for a':

$$f = 1 / ( \text{T13-period-value} \times \text{T13-resolution} ) = 1 / ( 37500 \times 60,606 \text{ ns} ) = 440 \text{ Hz}$$

Note – frequency (Timer 13), **octave = OL**:



In our programming example we are going to use also the following period-values for Timer 13 for octave = **OL**:

```
unsigned int T13_values[] =
{62977, 59550, 56122, 53061, 50000, 47278, 44685, 42092, 39796, 37500, 35450, 33401, 275};
/*
[0]=c',[1]=cis',[2]=d',[3]=dis',[4]=e',[5]=f',[6]=fis',[7]=g',[8]=gis',[9]=a',[10]=ais',[11]=h',
[12]=<Frequency for rest>
*/
```

So we get the following values shown in the table below

[**Note**: Timer 13 resolution =  $1/(f_{clk}/8) = 1/(66\text{MHz}/8) = 121,21 \text{ ns}$ ]:

		Octave=OL T13 Prescaler=8	Octave=ON00 T13 Prescaler=4
T13 period values	note	f [Hz]	f [Hz]
T13_values[ 0] = 62977	c	131	262
T13_values[ 1] = 59550	cis	139	
T13_values[ 2] = 56122	d	147	294
T13_values[ 3] = 53061	dis	156	
T13_values[ 4] = 50000	e	165	330
T13_values[ 5] = 47278	f	175	349
T13_values[ 6] = 44685	fis	186	
T13_values[ 7] = 42092	g	196	392
T13_values[ 8] = 39796	gis	208	
T13_values[ 9] = 37500	a	220	440
T13_values[10] = 35450	ais	234	
T13_values[11] = 33401	h	247	494
T13_values[12] = 275	----	30000	60000

Therefore we use the following program sequence in our application:

```
// T13, note frequency  
CCU60_T13PR=T13_values[note]/octave; // note frequency  
CCU60_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %  
CCU60_vEnableShadowTransfer(CCU60_TIMER_13);
```

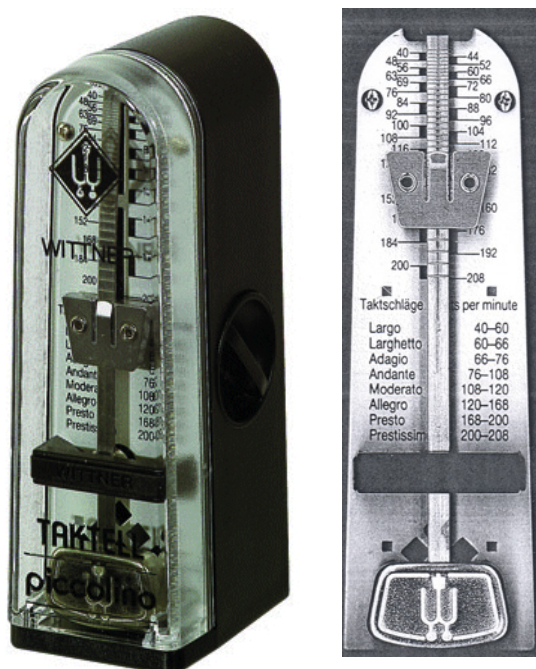


note – length (Timer 12)

$\circ$	=	<b>1/1</b>	Note	(= 4 Schläge) (= <b>4 beats</b> )
$\text{d}$	=	<b>1/2</b>	Note	(= 2 Schläge) (= <b>2 beats</b> )
$\text{J}$	=	<b>1/4</b>	Note	(= 1 Schlag) (= <b>1 beat</b> )
$\text{♩} \text{ ♪} \text{ ♫}$	=	<b>1/8</b>	Note	(= 1/2 Schlag) (= <b>1/2 beat</b> )
$\text{♩} \text{ ♪} \text{ ♫} \text{ ♬}$	=	<b>1/16</b>	Note	(= 1/4 Schlag) (= <b>1/4 beat</b> )



The metronome (a piece of equipment that repeats a regular beat, used by musicians to help them play music at the right speed) allows the exact definition of the tempo.



So we get the following table for speed:

Tempo	Beats per minute
Grave	
Largo/Lento	40-60
Larghetto moderato	
Larghetto	60-66
Adagio moderato	
Adagio	66-76
Adagio cantabile	
Andantino moderato	
Andantino	
Andante moderato	
Andante	76-108
Allegretto moderato	
Allegretto	
Moderato 1	
Moderato 2	108-120
Allegro moderato	
Allegro	120-168
Vivace 1	
Vivace 2	
Presto moderato	
Presto/Allegro assai	168-200
Prestissimo moderato	
Prestissimo	200-208




**Note:**






Our software supports 50 to 199 Beats per minute:

`Tx : Change tempo (x = 50 ... 199 Beats per Minute)`

And tempo is used in the following way:






```
CCU60_T12PR= ((float)current_note_length/ (float)tempo*120.0); // period value
note length
```


e.g.  @ 120 means:  
120 “beats” / minute =  
2 “beats” / second →  
1 “beat” = 0,5 second

	1/1 note = 4 beats = 4 * 0,5 = <b>2</b> [s]
	1/2 note = 2 beats = 2 * 0,5 = <b>1</b> [s]
	1/4 note = 1 beat = 1 * 0,5 = <b>0,5</b> [s]
	1/8 note = 1/2 beat = 1/2 * 0,5 = <b>0,25</b> [s]
	1/16 note = 1/4 beat = 1/4 * 0,5 = <b>0,125</b> [s]

So we get the following values shown in the table below

(**Note:** Timer 12 resolution = 66 MHz / 256 (T12PRE=1, done by software) / 32 = 8,0566 kHz →  
Resolution = 124,12 µs):

T12 period values	note	note	note length [s]
16113 / 1 = 16113		1/1	2
16113 / 2 = 8057		1/2	1
16113 / 4 = 4028		1/4	<b>0,5</b>
16113 / 8 = 2014		1/8	0,25
16113 / 16 = 1007		1/16	0,125

e.g. for  :  
note length = T12-period-value / 4 \* T12-resolution  
note length = **16133** / 4 \* 124,12 µs = **0,5** [s]

In our programming example we use the following code sequences:

```
unsigned int length_of_a_whole_note = 16113;  
// Default length of a whole-note with tempo 120
```

```
// note length:  
case 'L': switch (song[++pos])  
    {  
        case '1': if (song[++pos]=='6')  
            current_note_length=length_of_a_whole_note/16;  
            else  
            {  
                pos--;  
                current_note_length=length_of_a_whole_note;  
            }  
            break;  
        case '2': current_note_length=length_of_a_whole_note/2;  
            break;  
        case '4': current_note_length=length_of_a_whole_note/4;  
            break;  
        case '8': current_note_length=length_of_a_whole_note/8;  
            break;  
        default : ;  
            break;  
    }  
old_note_length=current_note_length;  
pos++;  
read_song_string();  
break;
```

```
CCU60_T12PR=((float)current_note_length/(float)tempo*120.0); // period  
value note length
```



#### 4.) Appendix: songs used

##### 4.1.) Song a: Maus am Mars:



// Maus am Mars (song a):

code unsigned char

```
songa[]="T120O0L4FL8AL4O1C.O0L8FEGL2O1CO0P4P8L4EL8GO1L4C.O0L8EFAL2O1CP4  
P8O0L4FL8AO1L4C.O0L8FH-O1L4DFL8FEDDCO0HO1CDCO0H-GL2F.";
```

**Note:**

Thanks to Christian Perschl ([www.perschl.at](http://www.perschl.at)).

The songstring above was written down by Christian while humming the melody.

#### 4.2.) Song b: Yesterday:



// Yesterday (song b):

code unsigned char

```
songb[]="T120O0L8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2AAAL4O1DEFL8EDL4E.L8DL4CEFCO0H-
AL8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2A";
```

#### Note:

Thanks to Christian Perschl ([www.perschl.at](http://www.perschl.at)).

The songstring above was written down by Christian while humming the melody.

4.3.) Song c: Bruder Jakob:

Bruder Jakob

Französisches Kinderlied



Bru- der Ja- kob, Bru- der Ja- kob,



schläfst du noch, schläfst du noch?



Hörst du nicht die Glock- ken, hörst du nicht die Glock- ken:



ding, dong, ding, ding dong, ding!

// Bruder Jakob (song c):

code unsigned char songc[]="T120O0L4FGAFFGAF4H-O1L2CO0L4AH-O1L2CL8CDCO0L8H-L4AF01L8CDCO0L8H-L4AFFCL2FL4FCL2F";

4.4.) Song d: Happy birthday:

Happy birthday

Englisches Kinderlied



Hap- py birth- day to you, hap- py birth- day to you,

hap- py birth- day, hap- py birth- day,

hap- py birth- day to you!

// Happy birthday (song d):

code unsigned char

songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C  
CO0L4HGAL2G";

4.5.) Song e: Take Me Home, Country Roads:

**TAKE ME HOME,  
COUNTRY ROADS**



1. Al-most heav-en, West Vir-gin - ia, - Blue Ridge Moun-tains,  
Shen-an-do-ah Riv- er. Life is old there, ol-der than the  
trees, young-er than the moun-tains grow-in' like a breeze.  
Coun-try Roads. take me home to the Place I be-  
long: West Vir- gin-ia, mountain mom-ma, Take me  
home, Coun-try Roads.

// Take Me Home, Country Roads (song e):

code unsigned char

```
songe[]="T19900L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EEEDL8EL4GL1GP  
1L4DDE.L2D.L4EGGHL1HL4AAAAH.L2A.L4EGGAL2G.L4GAL1HL8HAL4GL1AL4HAL1G  
L4HO1L4DL1EL4EEDO0L1HL8HAGAL1HL8HAL4GL1GL4GAL1G";
```



4.6.) Song f: Es tanzt ein Bi-ba-butzemann:

Es tanzt ein Bi-ba-butzemann

Volkswiese



Es tanzt ein Bi- ba- but- ze-mann in un-serm Hausher- um.



Er rüt- telt sich, er schüt- telt sich,  
er wirft sein Säck- lein hin- ter sich.



Es tanzt ein Bi- ba- but- ze-mann in un-serm Hausher- um.

// Es tanzt ein Bi-ba-butzemann (song f):

code unsigned char

songf[]="T199O0L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8  
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";

#### 4.7.) Song g: Ich geh mit meiner Laterne:

Ich geh mit meiner Laterne



Ich geh mit mei-ner La- ter- ne und mei- ne La- ter- ne mit mir.  
Dort o- benleuch- tendieSter-ne, hier un- ten, da leuch- ten wir.



Mein Lichtist aus,wir gehnnachHaus.La- bim-mel, la-bam-mel,la- bum.

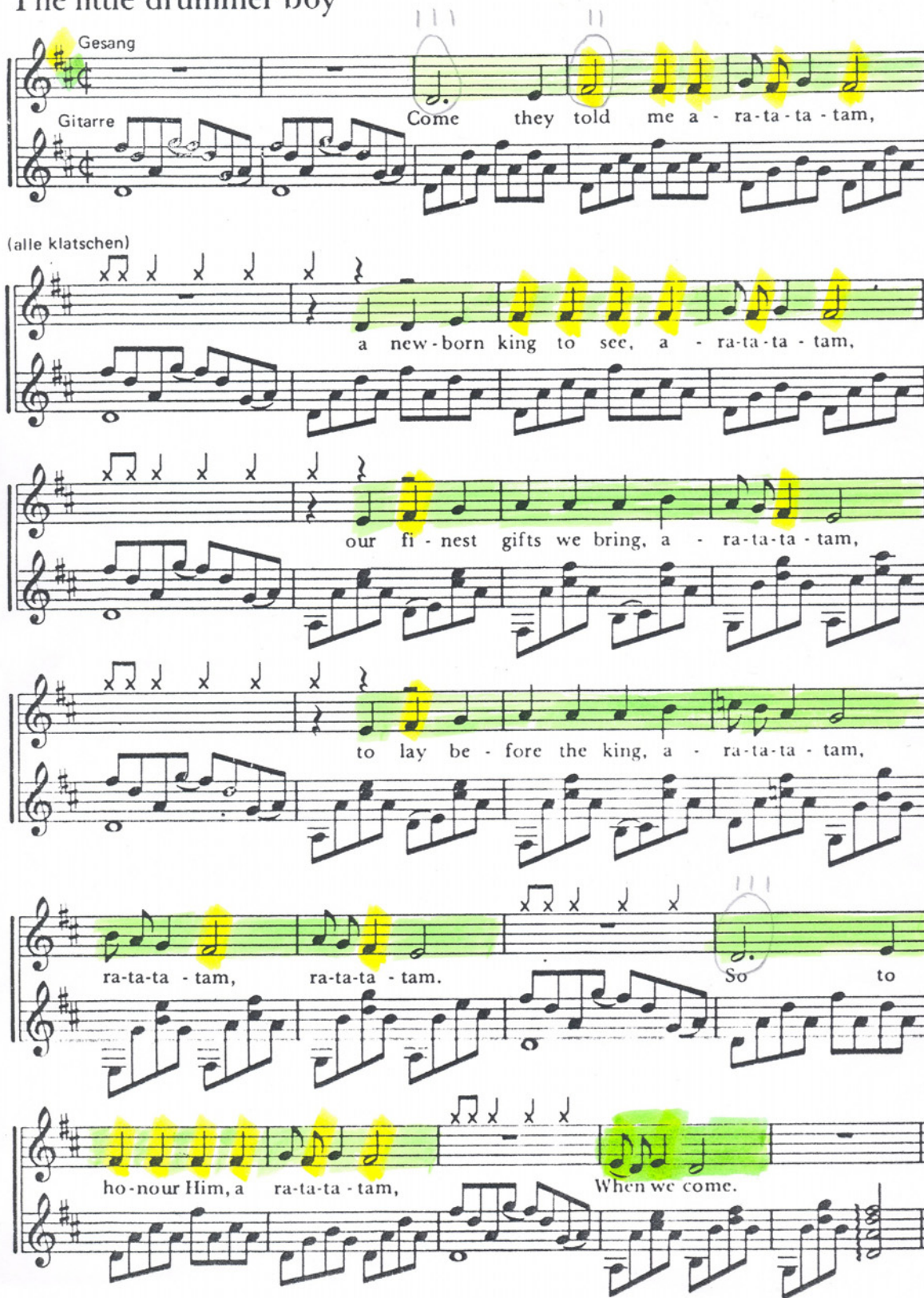
// Ich geh mit meiner Laterne (song g):

code unsigned char

```
songg[]="T120O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8CL4FL8FA  
FAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4F  
L8FGGGGAGL4FP4.O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4FL8FGGGGAGL4FP4.";
```

4.8.) Song h: The little drummer boy:

The little drummer boy



The musical score for "The little drummer boy" is presented in a two-staff format. The top staff is labeled "Gesang" (Vocal) and the bottom staff is labeled "Gitarre" (Guitar). The key signature is one sharp (F#) and the time signature is 4/4. The lyrics are written below the vocal staff. The score includes a drum part indicated by "x" marks above the vocal staff, with the instruction "(alle klatschen)" (all clap) written above the first drum part. The lyrics are: "Come they told me a - ra-ta-ta - tam, a new-born king to see, a - ra-ta-ta - tam, our fi - nest gifts we bring, a - ra-ta-ta - tam, to lay be - fore the king, a - ra-ta-ta - tam, ra-ta-ta - tam, ra-ta-ta - tam. So to ho-nour Him, a ra-ta-ta - tam, When we come." The score is divided into measures by vertical bar lines. Some notes and measures are highlighted in yellow. There are also some handwritten annotations, such as "111" and "11" above certain notes.

// The little drummer boy (song h):

code unsigned char

```
songh[]="T120P2O0L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+L4F+L4F+L8G  
F+L4GL2F+P2L4EF+L4GAAHL8AGL4F+L2EP2L4EF+L4GAAHO1L8CO0L8HL4AL2GL8  
HAL4GL2F+L8AGL4F+L2EP1L2D.L4EL4F+F+F+F+L8GF+L4GL2F+P1L8EDL4EL2D";
```



4.9.) Song i: Hey, Pippi Langstrumpf:

## Hey, Pippi Langstrumpf



1. Zwei mal drei macht vier, wi - de wi - de witt und drei macht neu - ne.  
Drei mal drei macht sechs, wi - de wi - de wer will's von mir ler - nen?



Ich mach' mir die Welt, wi - de wi - de wie sie mir ge - fällt.  
Al - le, groß und klein, tra - la - la - la lad' ich zu mir ein.



Ref.: Hey, Pip - pi Lang - strumpf, tral - le - ri, tral - le - ri, tral - ler hop - sa - sa.



Hey, Pip - pi Lang - strumpf, die macht, was ihr ge - fällt.



Ich hab' ein Haus, ein kun - ter - bun - tes Haus, ein



Äff - chen und ein Pferd, die schau - en da zum Fen - ster



D G A D  
 raus. Ich hab' ein Haus, ein Äff - chen und ein Pferd und  
 Hm Em A D A D  
 je - der, der uns mag, kriegst un - ser Ein - mal - eins ge - lehrt.

2. Zwei mal drei macht vier, wide wide witt und drei macht neune,  
 wir machen uns die Welt, wide wide wie sie uns gefällt.  
 Drei mal drei macht sechs, wide wide wer will's von uns lernen?  
 Alle, groß und klein, tralala, lad' ich zu uns ein.  
 Ref.: Hey, Pippi Langstrumpf,...



// Hey, Pippi Langstrumpf (song i):

code unsigned char

```
songi[]="T180OLL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+OLL4AONO
0L4DF+DL2EL8GF+EDL4C+EOLL4AON00L4C+DP4P2OLL4AON00L4DF+DL2EL8GF+ED
L4C+EOLAON00L4C+L2DF+OLL4AON00L4DF+DL2EL8GF+EDL4C+EOLL4AON00L4C+
DP4P2O0L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+
EEDC+DP4L2F+GAH.O1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+GL
4AF+GAL2H.O1L4DC+O0L4HAGL2A.O1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+EDP2";
```

4.10.) Song j: Stille Nacht, heilige Nacht:



Stille Nacht, heilige Nacht



// Stille Nacht, heilige Nacht (song j):

code unsigned char

```
songj[]="T72O0L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8DO0L4H.O1L4CL8CO0L4G.L
4AL8AO1L8C.O0L16HL8AL8G.L16AL8GL4E.L4AL8AO1L8C.O0L16HL8AL8G.L16AL8GL4
E.O1L4DL8DL8F.L16DO0L8HO1L4C.L4E.L8C.O0L16GL8EL8G.L16FL8DL1C.";
```

4.11.) Song k: Junge komm bald wieder:

**Junge komm' bald wieder**



Jun - ge, komm' bald wie - der, bald wie - der nach Haus. Jun - ge, fahr' nie wie - der, nie  
wie - der hin - aus. Ich mach' mir Sor - gen, Sor - gen um dich. Denk' auch an mor - gen,  
denk' auch an mich. Jun - ge, komm' bald wie - der, bald wie - der nach Haus. Jun - ge, fahr' nie  
wie - der, nie wie - der hin - aus. Ich weiß noch wie die er - ste Fahrt ver - lief, ich  
schlich mich heim - lich fort, als Mut - ter schlief. Als sie er - wach - te, war ich auf dem  
Meer. Im er - sten Brief stand: „Komm doch bald wie - der her!“

2. Junge, komm' bald wieder, ...  
Wohin die Seefahrt mich im Leben trieb,  
ich weiß noch heute,  
was mir Mutter schrieb.  
In jedem Hafen kam ein Brief an Bord,  
und immer schrieb sie:  
„Bleib nicht so lange fort!“  
Junge, komm' bald wieder, ...

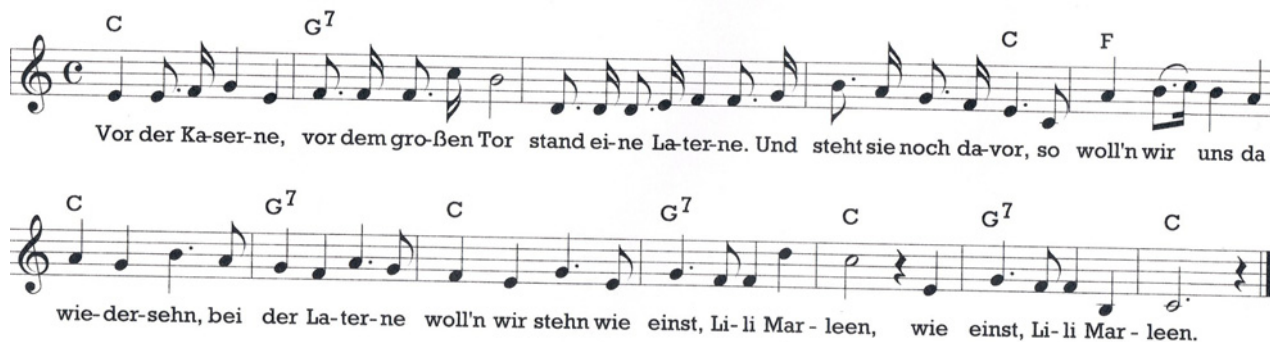
// Junge komm bald wieder (song k):

code unsigned char

```
songk[]="T12000L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C.L4EEL8D+
L8EL4F+L4E.L8EL4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+.L8EL4EL2DL4E
L4D.L8COLL2H.ONO00L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C.L4E
EL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8DDDDDDDDDDL4DP8L8DL8D+L8DDDDDL8D
+L8DL4DP8L8DL8EEEEEEEL2GP8L8EL1DP8L8DL8EEEL4E.P8L8GGGF+L8GL1A.";
```

#### 4.12.) Song 1: Lili Marleen:

### Lili Marleen



Vor der Ka-ser-ne, vor dem gro-ßen Tor stand ei-ne La-ter-ne. Und steht sie noch da-vor, so woll'n wir uns da

wie-der-sehn, bei der La-ter-ne woll'n wir stehn wie einst, Li-li Mar-leen, wie einst, Li-li Mar-leen.

2. Unsre beiden Schatten  
sahn wie einer aus.  
Daß wir so lieb uns hatten,  
das sah man gleich daraus.  
Und alle Leute solln es sehn,  
wenn wir bei der Laterne stehn  
wie einst, Lili Marleen.

3. Schon rief der Posten:  
„Sie blasen Zapfenstreich.  
Es kann drei Tage kosten!“  
Kamerad, ich komm ja gleich.  
Da sagten wir auf Wiedersehn,  
wie gerne wollt ich mit dir gehn,  
mit dir, Lili Marleen.

4. Deine Schritte kennt sie,  
deinen zieren Gang.  
Alle Abend brennt sie,  
doch mich vergaß sie lang.  
Und sollte mir ein Leids geschehn:  
Wer wird bei der Laterne stehn  
mir dir, Lili Marleen?

5. Aus dem stillen Raume,  
aus der Erde Grund  
hebt mich wie im Traume  
dein verliebter Mund.  
Wenn sich die späten Nebel drehn,  
werd ich bei der Laterne stehn  
wie einst, Lili Marleen.

// Lili Marleen (song 1):

code unsigned char

```
song1[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL
8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL
4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HONOO0L2C.";
```

#### 4.13.) Another song: Lady Bird:



// Lady Bird:

```
"T1500NO0L4F+P16F+P16F+.P8L16C+P16L8EP16E.P16L2EL8EP16L4C+P16C+P16C+.P16OLL8AP16
L4HP16G+P16L2EP4ONO0L4F+P16L8F+.P16L2F+P4L4EP16L8E.P16L2EP4L4C+P16L8C+.P16L4C+.O
LAP16L4HP16G+P16EP16L4EP16L8F+.P16L16F+P16L1F+P8L4G+P16G+P16G+P16L8G+.P16F+P16L1
F+";
```

**Note:**

Thanks to Maureen Sturgeon.  
She wrote down the songstring above.



**Summary:**

In this step-by-step book you have learned how to use the PWM Unit.

Have fun and enjoy working with microcontrollers with CCU6 modules!

**Note:**

There are step-by-step books for 8 bit microcontrollers (e.g. XC866 and XC88x), 16 bit microcontrollers (e.g. C16x and XC16x) and 32 bit microcontrollers (e.g. TC1796 and TC1130).

All these step-by-step books use the same microcontroller resources and the same example code.

This means: configuration steps, function names and variable names are identical.

This should give you a good opportunity to get in contact with another Infineon microcontroller family or tool-chain!

There are even more programming examples available using the same style [e.g. ADC-examples, CAPCOM6-examples (e.g. BLDC-Motor), Simulator examples, C++ examples] based on these step-by-step books.

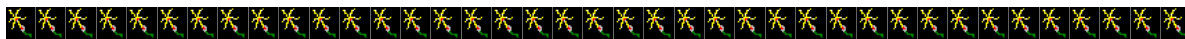
## 5.) Thanks To



Maria, Christian, Hermann and Maureen for their support.



**6.) Feedback (XE167 Playing Music):**  
**Your opinion, suggestions and/or criticisms**



**Contact Details (this section may remain blank should you wish to offer feedback anonymously):**

---

---

---

If you have any suggestions please send this sheet back to:

**email:** [mcdocu.comments@infineon.com](mailto:mcdocu.comments@infineon.com)

**FAX:** +43 (0) 4242 3020 5783



**Your suggestions:**

---

---

---

---

---

---

---

---

---

<http://www.infineon.com>

Published by Infineon Technologies AG