Application Note, V2.0, June 2008

AP16138

XE164

UConnect-CAN XE164: Playing music using the CAPCOM6 module using the KEIL tool chain

Microcontrollers

Never stop thinking

Edition 2008-07-16

Published by
Infineon Technologies AG
81726 München, Germany

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**).

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:	2008-06	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

Table of Contents **Page**

Note: Table of Contents see page 12 and page 13.

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 AP16137.

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 XE164 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_0 pins used by our PWM module CCU61:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P0.0	CCU61_CC60	100 (note length, Timer_12)
P0.1	CCU61_CC61	100 (note length, Timer_12)
P0.2	CCU61_CC62	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P0.6	CCU61_COUT63	50 (note frequency, Timer_13)

Port_2 pins used as GPIO:

Port Lines	Function	Comment
P2.8	Show start of next note	Toggled via Software
P2.7	„use: program running signal“	Toggled via CAPCOM2_Timer_7 ISR

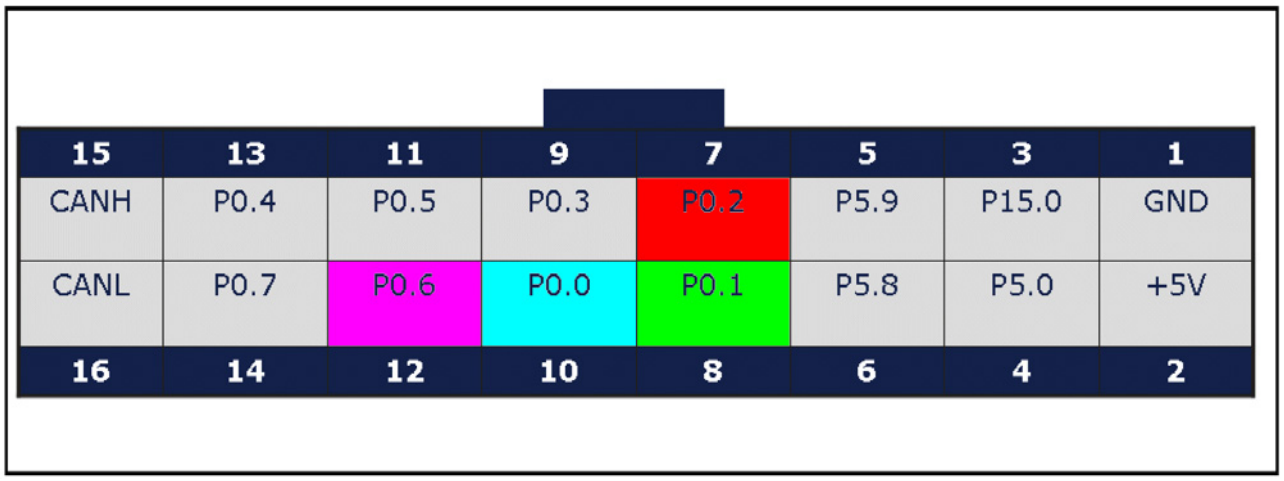
Port pins used:

Pin		CCU61-Channel	Modulated by	Purpose	
P0.0	CC60	CCU61 Channel 0	Modulated by T12	show note length duty cycle = 100 % only for measurement	
P0.1	CC61	CCU61 Channel 1	Modulated by T12	show note length duty-cycle = 100 % only for measurement	
P0.2	CC62	CCU61 Channel 2	Modulated by T12 + T13	Music Output: note length modulated by note frequency	
P2.8		---	Software	start of next note	
P2.7		---	Software	running signal	
P0.6	CC63	CCU61 Channel 3	Modulated by T13	note frequency only for measurement	

Using the UConnect-CAN XE164

Used Pins on the On-board header X400 of the UConnect-CAN XE164:
(Source: XE164 UConnect Manual)

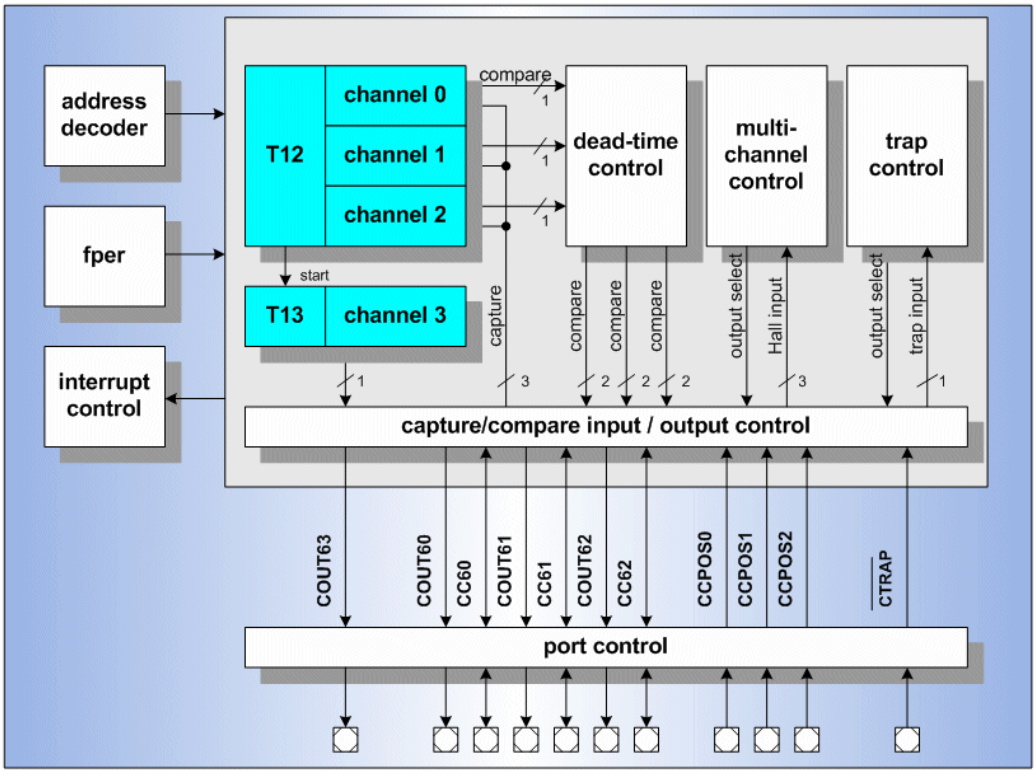
On-board header X400



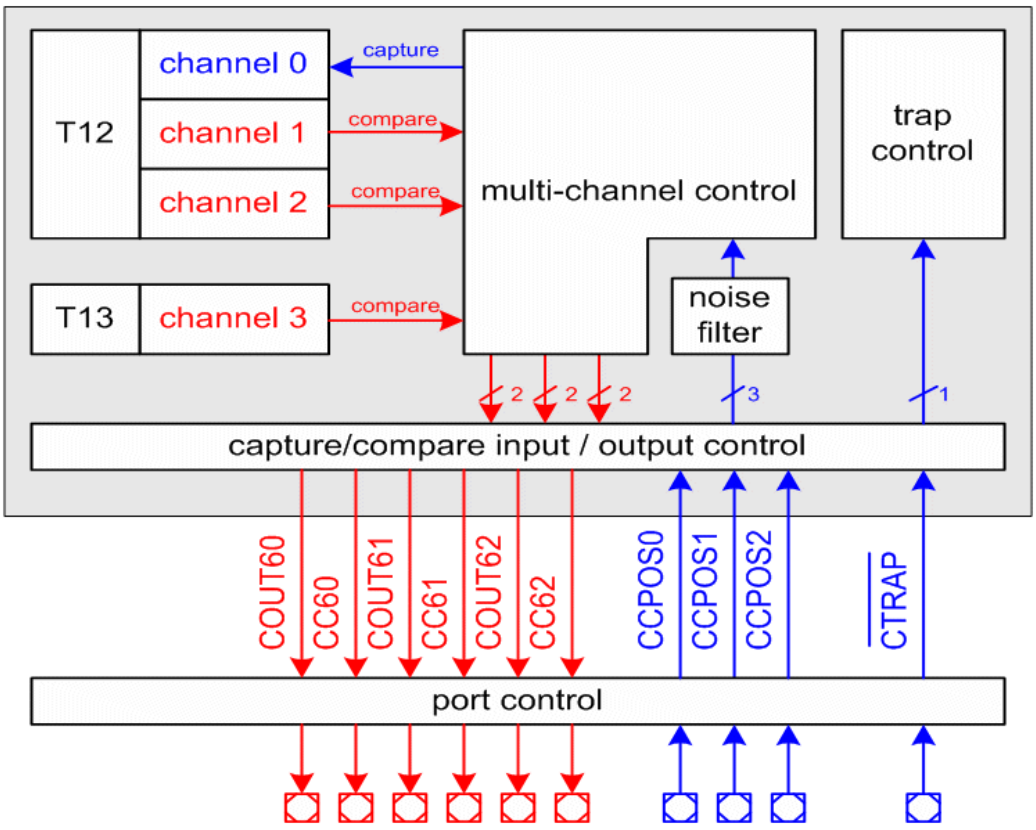
Pin number				
1	Ground			
2	+5V			
3	P15.0	ADC1_CH0		
4	P5.0	ADC0_CH0		
5	P5.9	ADC0_CH9	ADC1_CH9	CC2_T7IN CAPCOM2
6	P5.8	ADC0_CH8	ADC8_CH8	T12HRC / T13HRC CCU6x
7	P0.2	U1C0_SCK	CC62 CCU61	TXDC0 CAN0
8	P0.1	U1C0_DOUT	CC61 CCU61	TXDC0 CAN0
9	P0.3	U1C0_SELO	COUT60 CCU61	RXDC0B CAN0
10	P0.0	U1C0_DX0	CC60 CCU61	
11	P0.5	U1C1_SCK	COUT62 CCU61	
12	P0.6	U1C1_DOUT	COUT63 CCU61	TXDC1 CAN1
13	P0.4	U1C1_SELO	COUT61CCU61	RXDC1B CAN1
14	P0.7	U1C1_DX0	U1C1_DX0	CTRAPB CCU61
15	CANH Signal from CAN transceiver			
16	CANL Signal from CAN transceiver			

Note:
For further information, please refer to the XE164 UConnect Manual, V.1.1 .

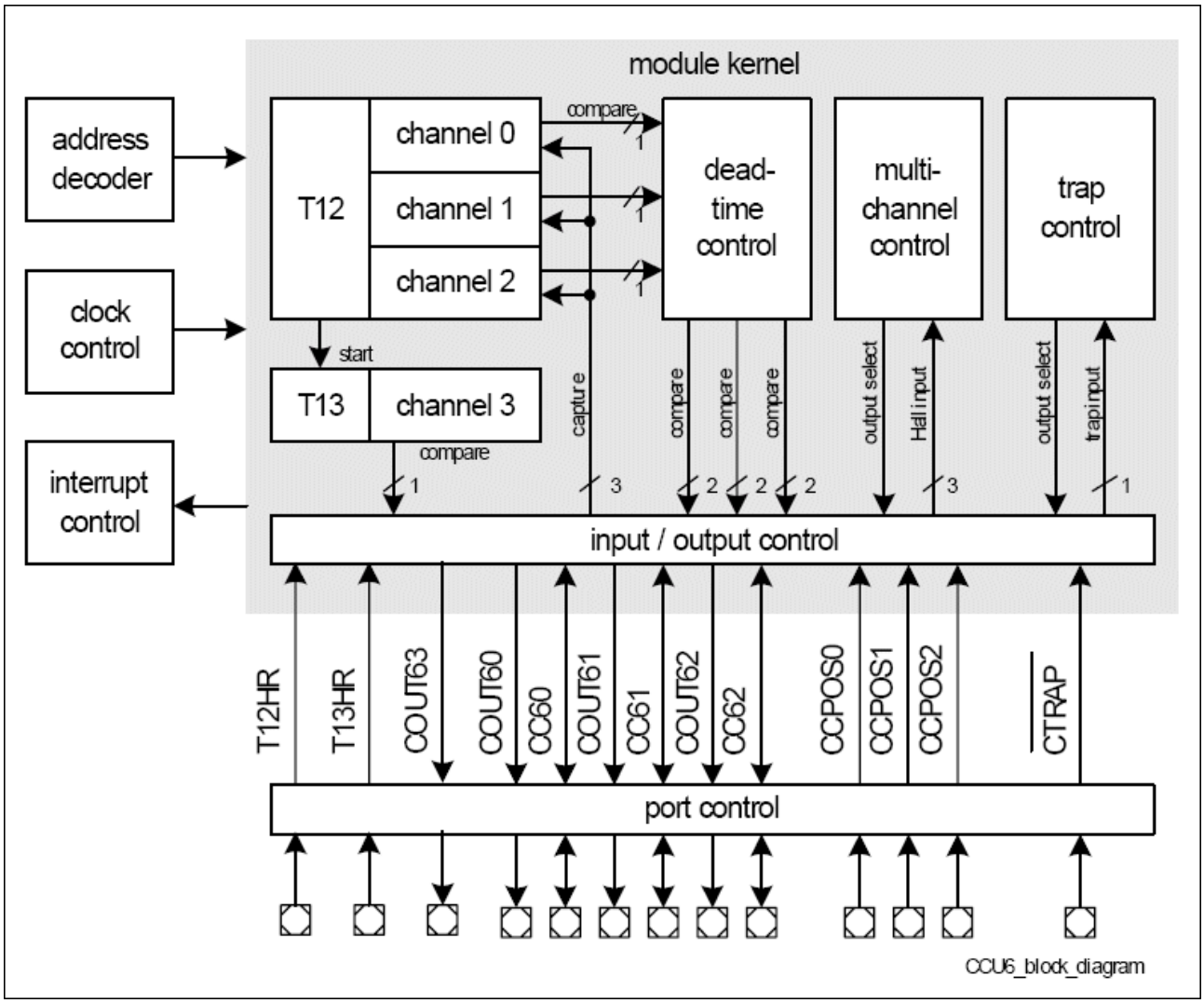
CCU6 Block Diagram – general use (Source: Product Marketing)



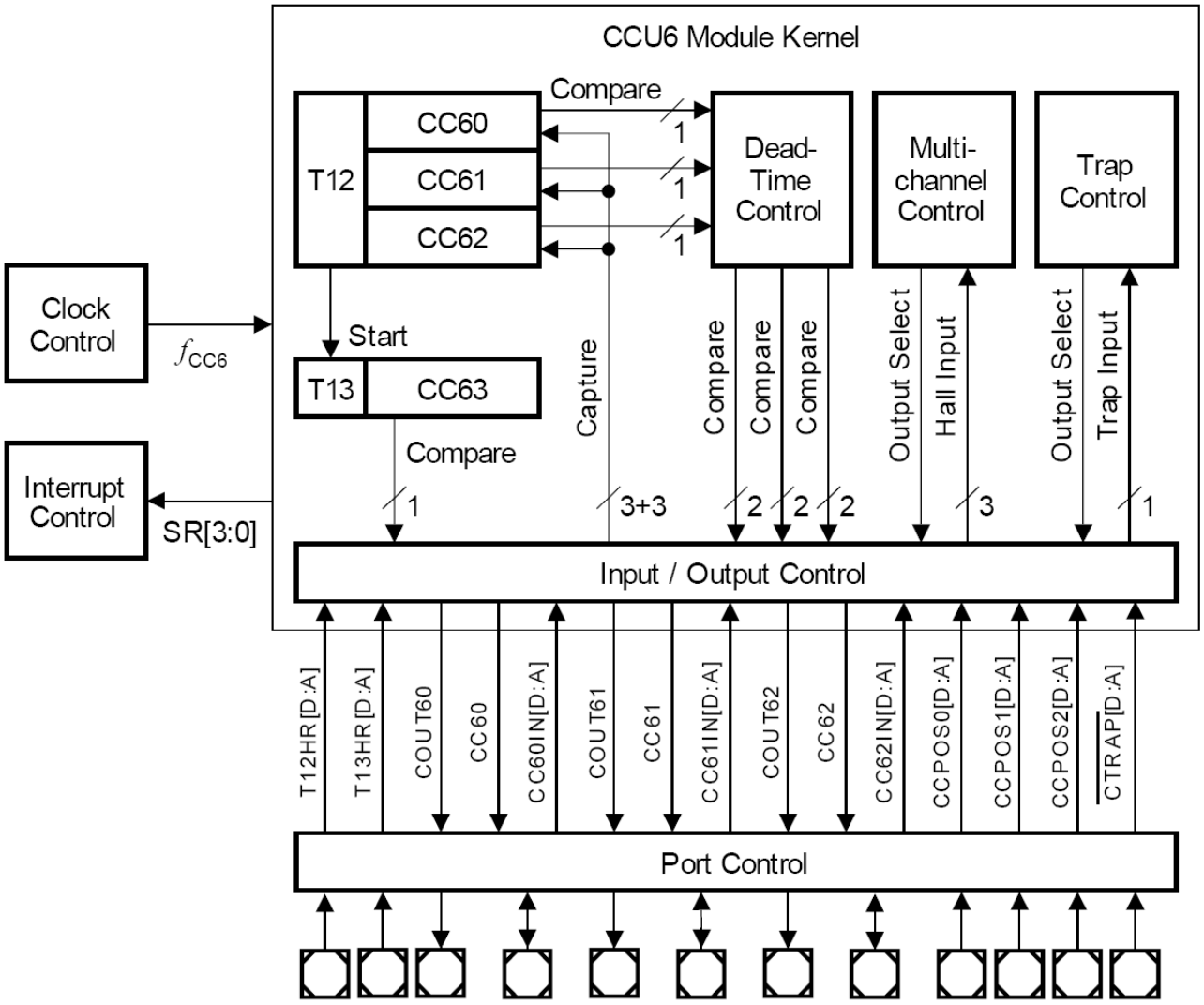
CCU6 Block Diagram – BLDC use (Source: Product Marketing)



CCU6 Block Diagram (Source: User's Manual)



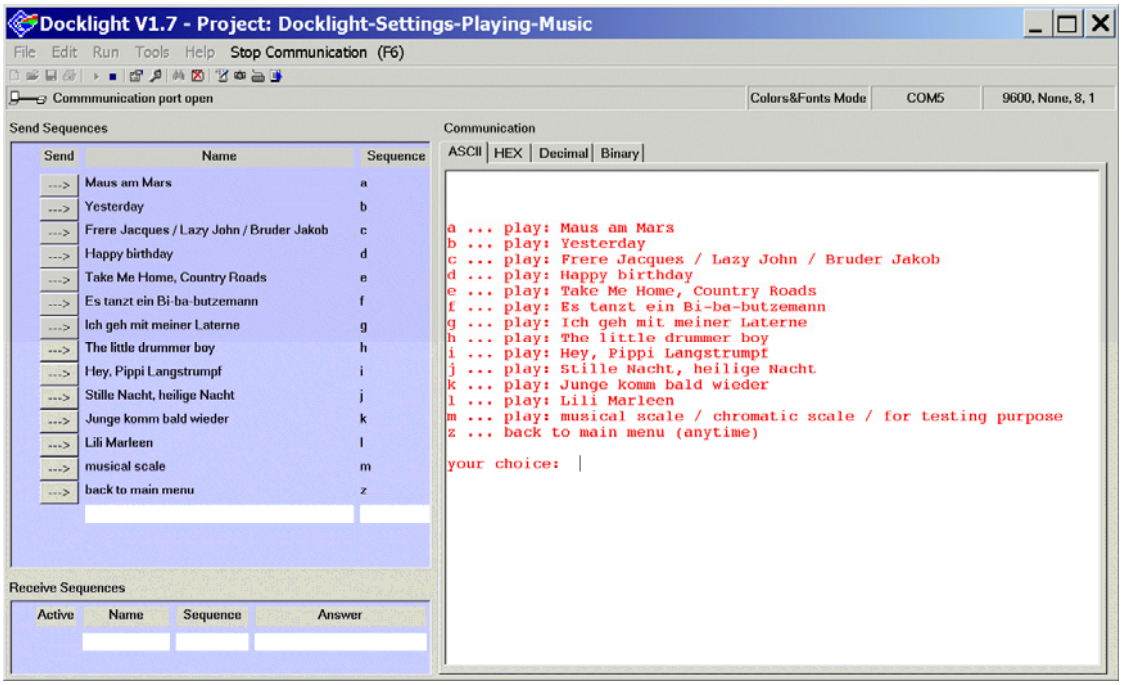
CCU6 Block Diagram (Source: XE166 User's Manual)



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:

Your program:	
Chapter/ Step	***** Recipes *****
1.)	**Asymmetrical / Edge-Aligned PWM generation** **Single Shot Mode (Timer12), Modulation (Timer13)** **Playing music**

Appendix:

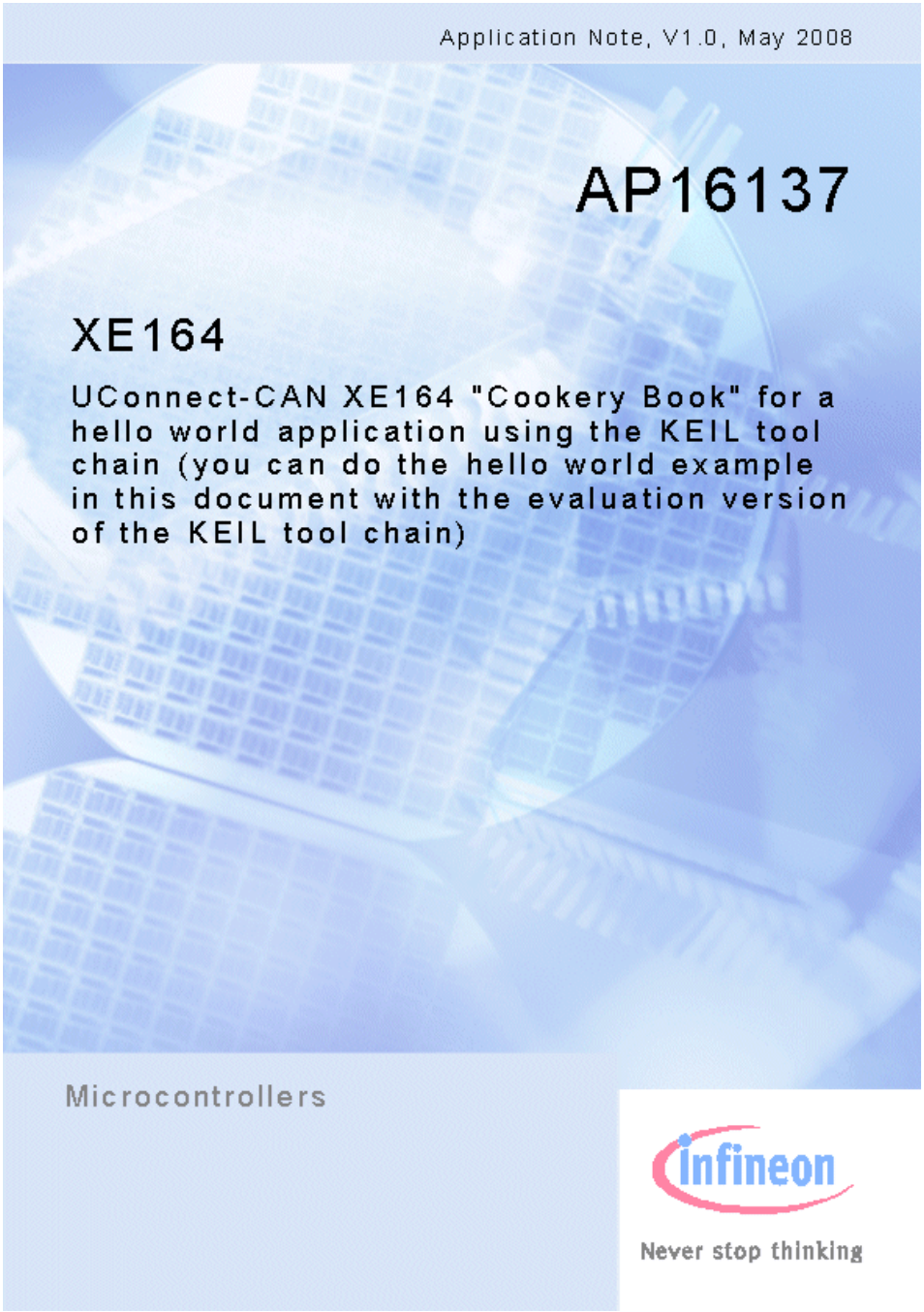
Chapter/ Step	*** Recipes ***
2.)	**Appendix: about music (note length, note frequency)**
3.)	**Appendix: CCU6 use to create note length and note frequency**
4.)	**Appendix: songs used**

Feedback:

5.)	**Thanks To**
6.)	**Feedback**

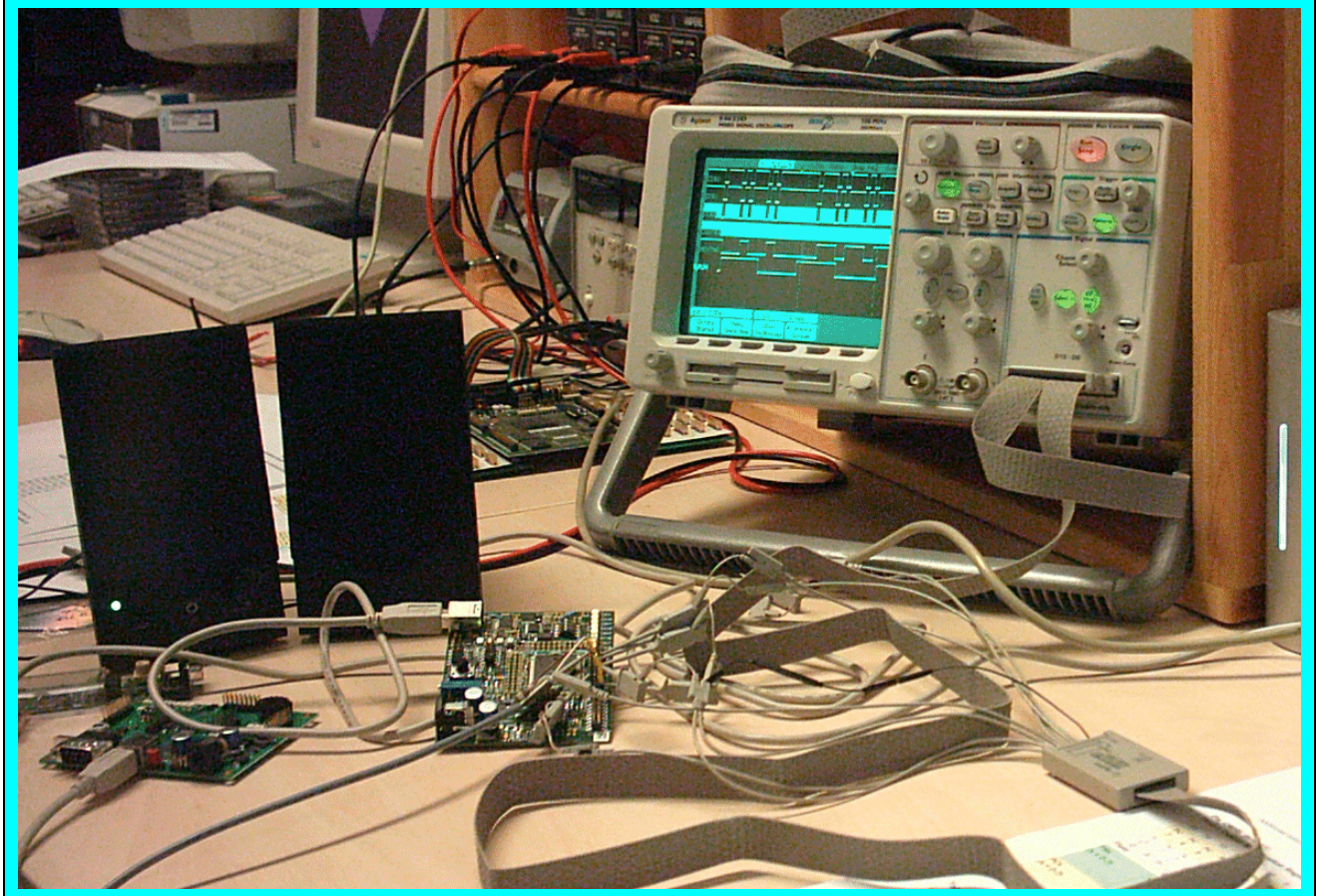
Do the UConnect-CAN XE164 Cookery Book:

Note:
It is necessary to follow all instructions in the UConnect-CAN XE164 "Cookery Book" (AP16137) step by step, as this is the basis for all instructions which will follow later.

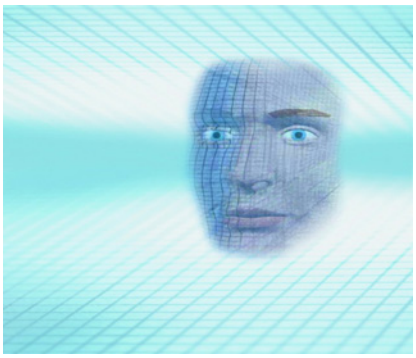


Note:
In the following steps of this document we will expand the “Hello World Application” (Application Note AP16137) with the requirements for PWM generation (playing music).

1.) Let's Get Started:

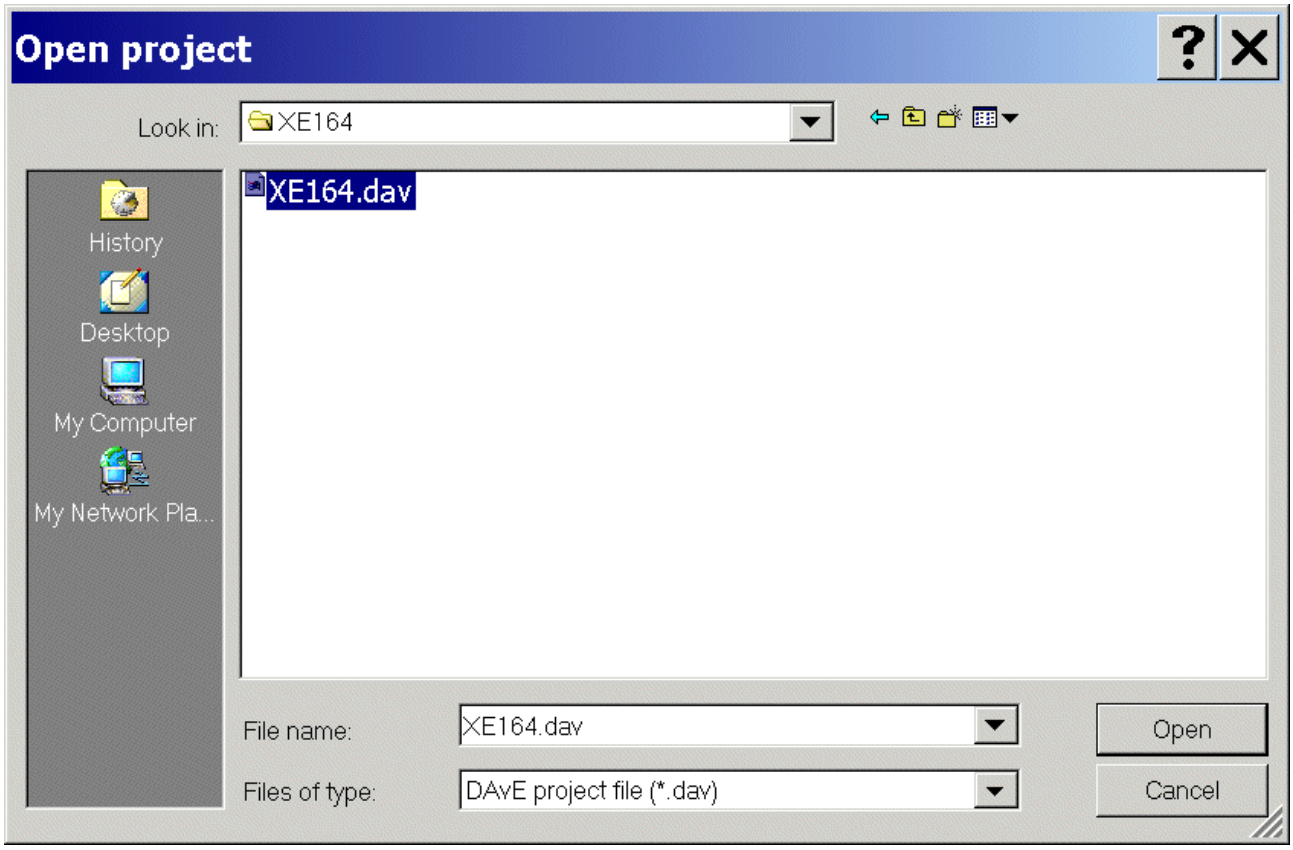


Configuring and Reconfiguring the DAvE Project Settings:

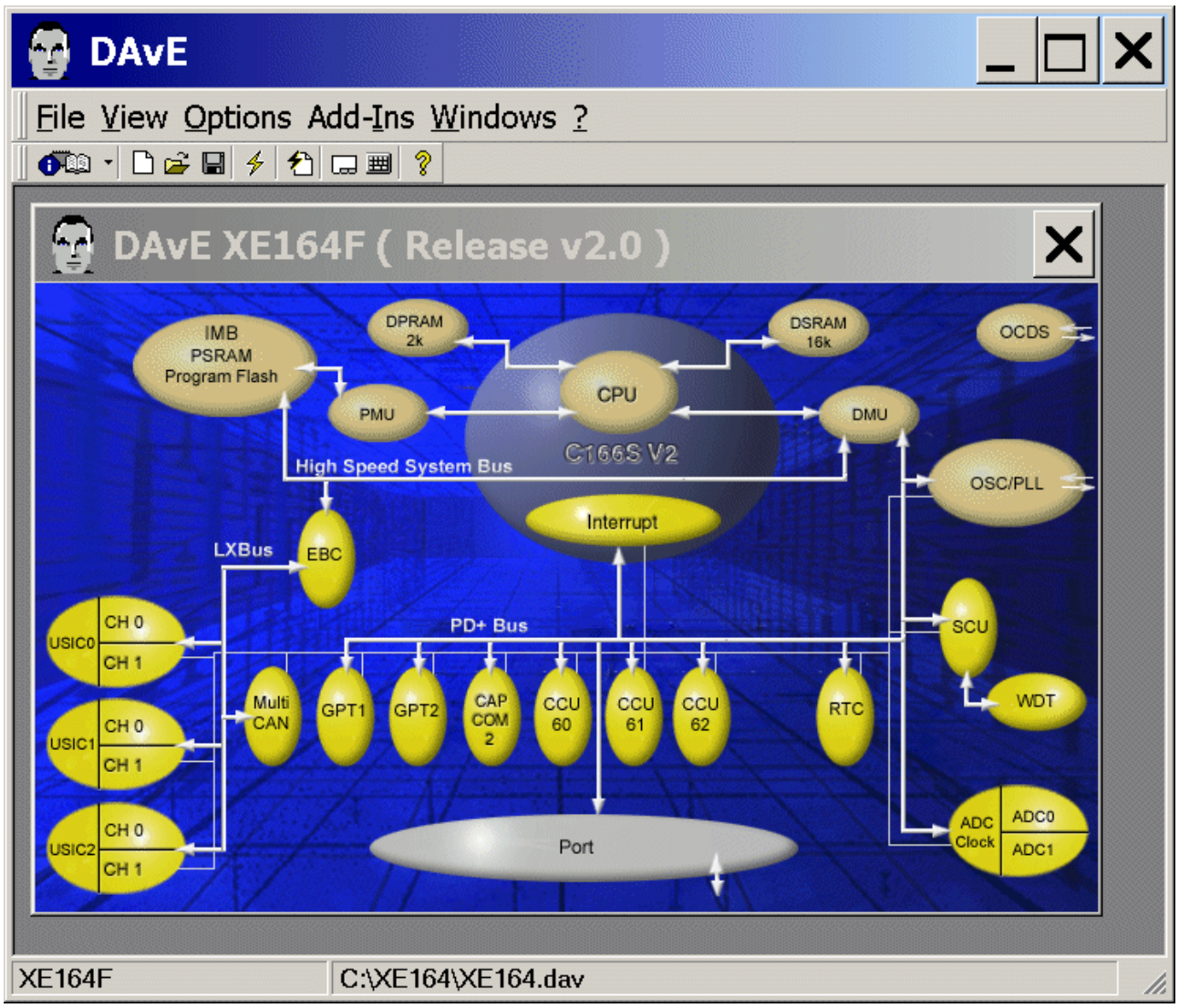


Start the program generator DAvE and open your XE164.dav DAvE project:

File
Open
Location: C:\XE164
Filename: XE164.dav

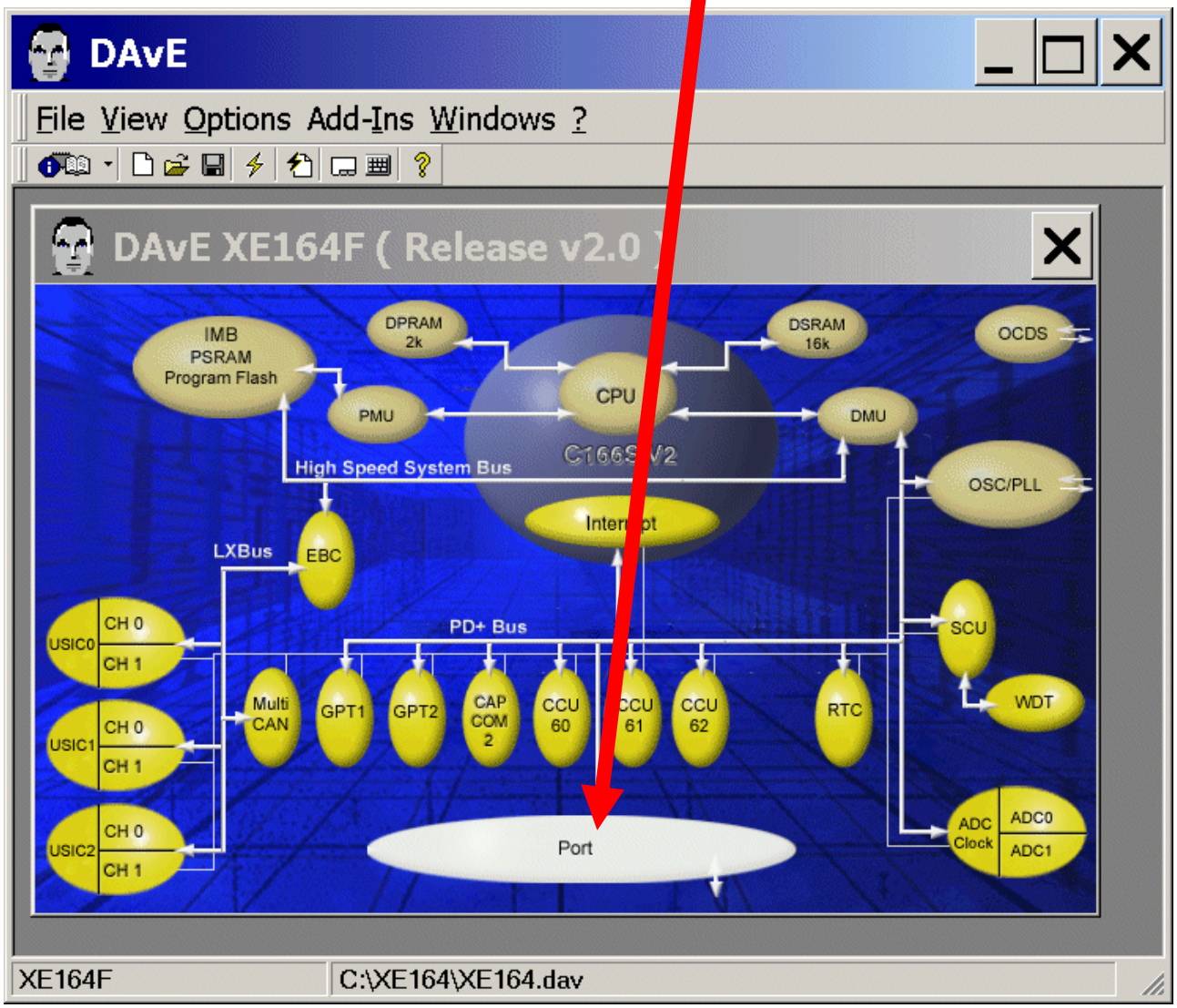


Click Open

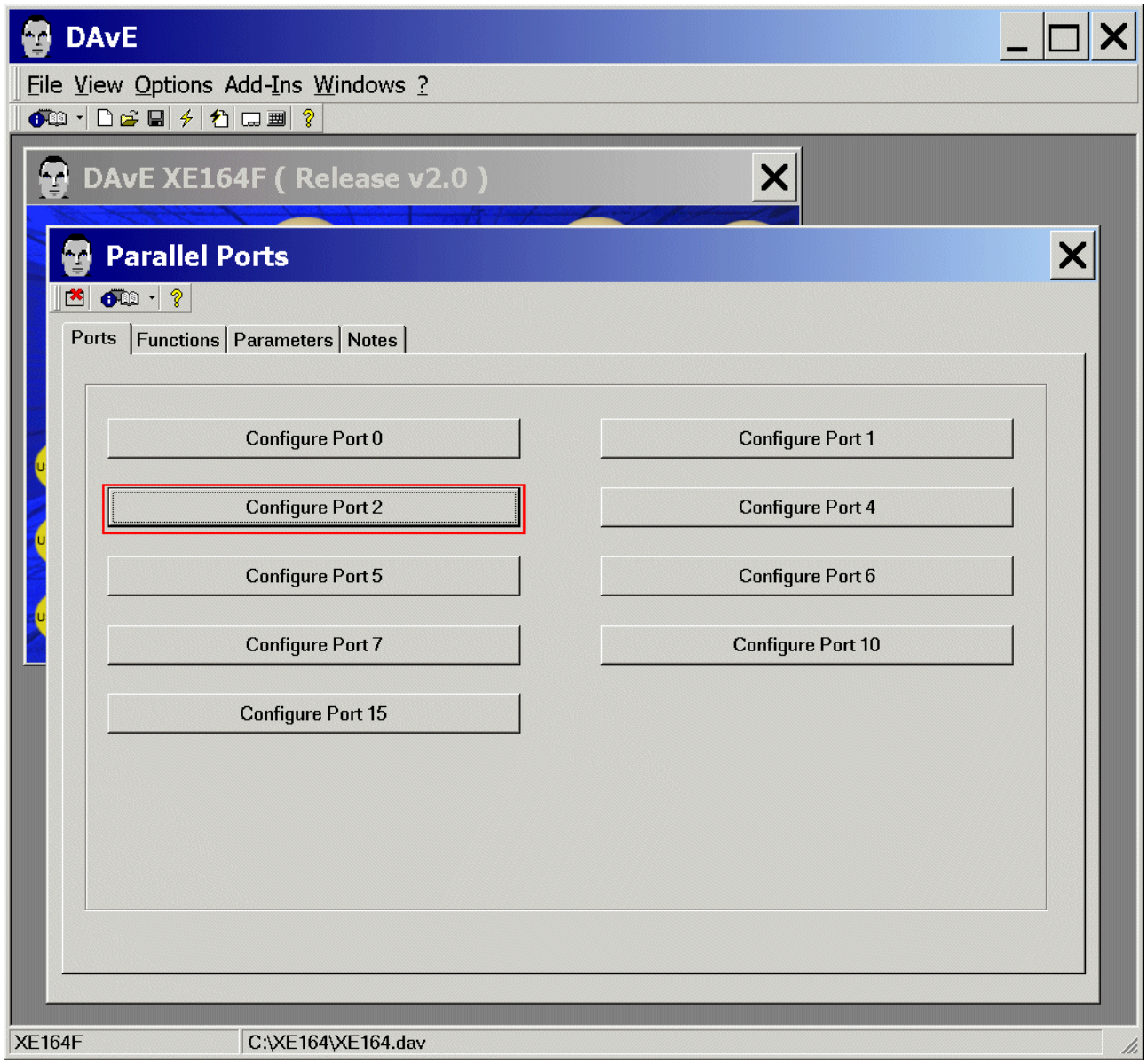

DAvE
File View Options Add-Ins Windows ?
DAvE XE164F (Release v2.0)
IMB
PSRAM
Program Flash
DPRAM
2k
DSRAM
16k
OCDS
PMU
CPU
DMU
C166S V2
High Speed System Bus
OSC/PLL
Interrupt
LXBus
EBC
PD+ Bus
SCU
USIC0
CH 0
CH 1
USIC1
CH 0
CH 1
USIC2
CH 0
CH 1
Multi
CAN
GPT1
GPT2
CAP
COM
2
CCU
60
CCU
61
CCU
62
RTC
WDT
Port
ADC
Clock
ADC0
ADC1
XE164F
C:\XE164\XE164.dav

Reconfiguration of Port 2:

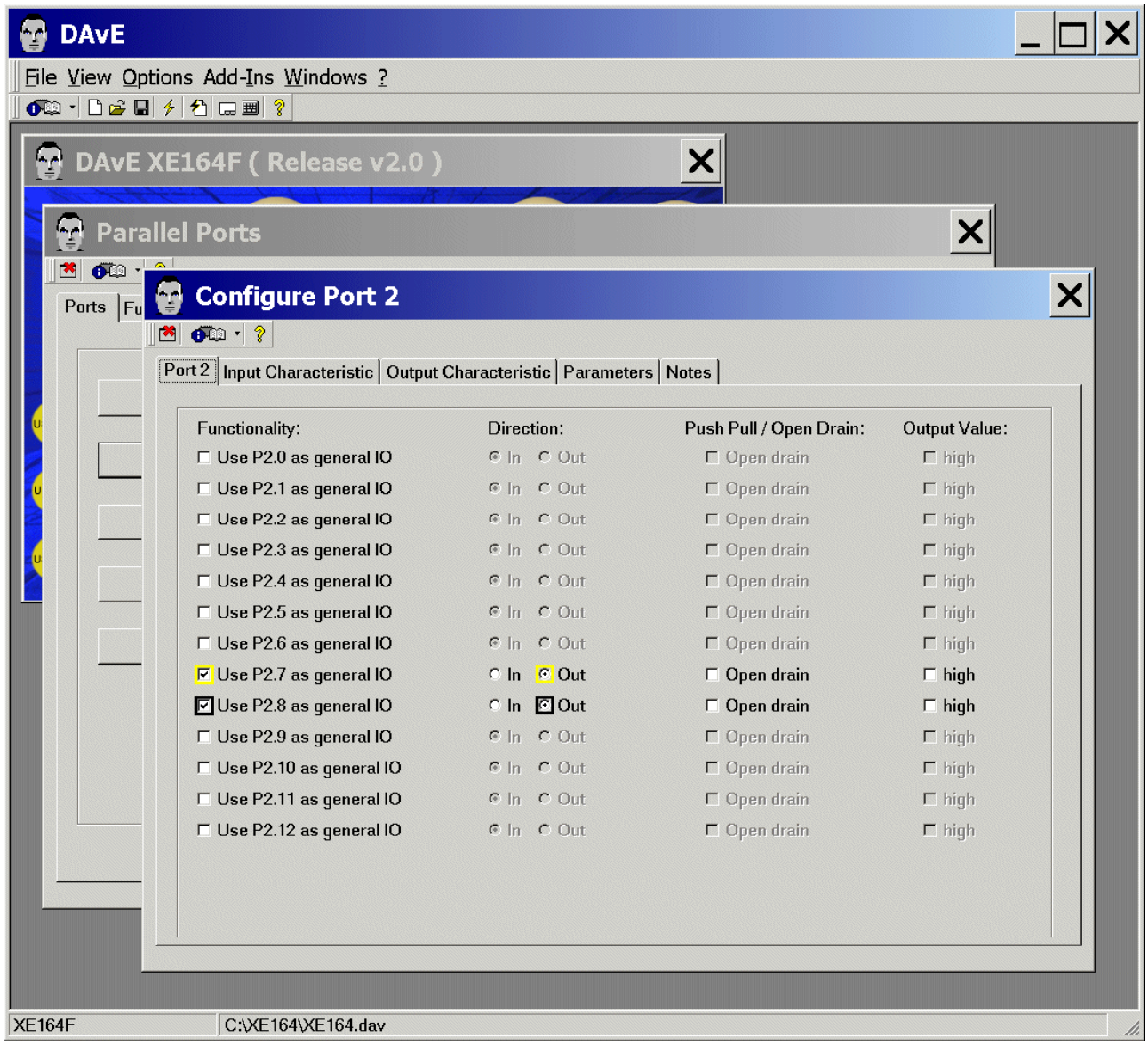
The (re)configuration window/dialog can be opened by clicking the specific block/module (Port).



Ports: click “Configure Port 2”



Port 2: Functionality: click ☑ Use P2.8 as general IO - Direction: click ⊙ Out



Remember:

Port pins used:

Pin		CCU61-Channel	Modulated by	Purpose	
P2.8		---	Software	start of next note	
P2.7		---	Software	running signal	

Input Characteristic: (do nothing)
Output Characteristic: (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.

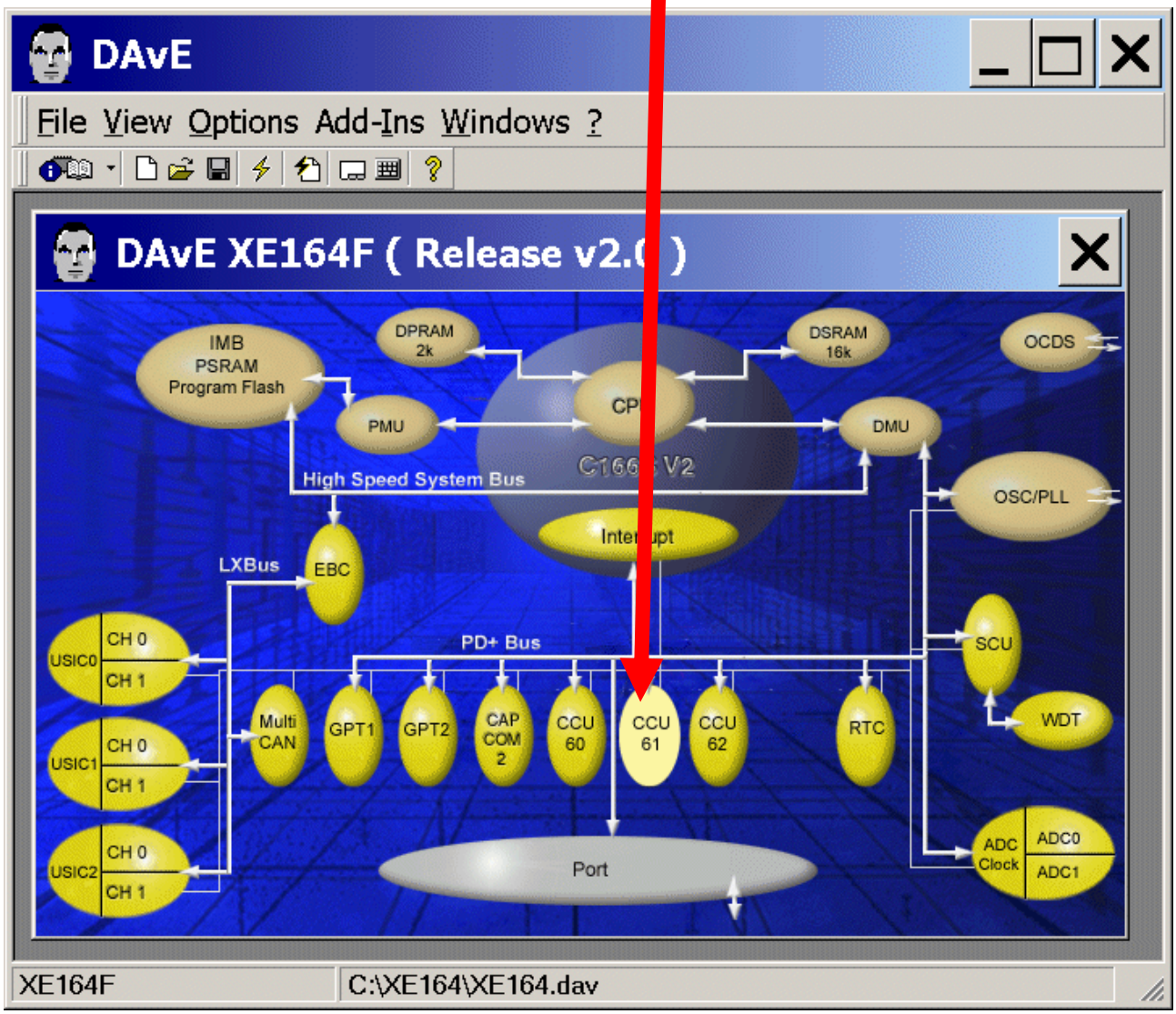
Functions: (do nothing)
Parameters: (do nothing)

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

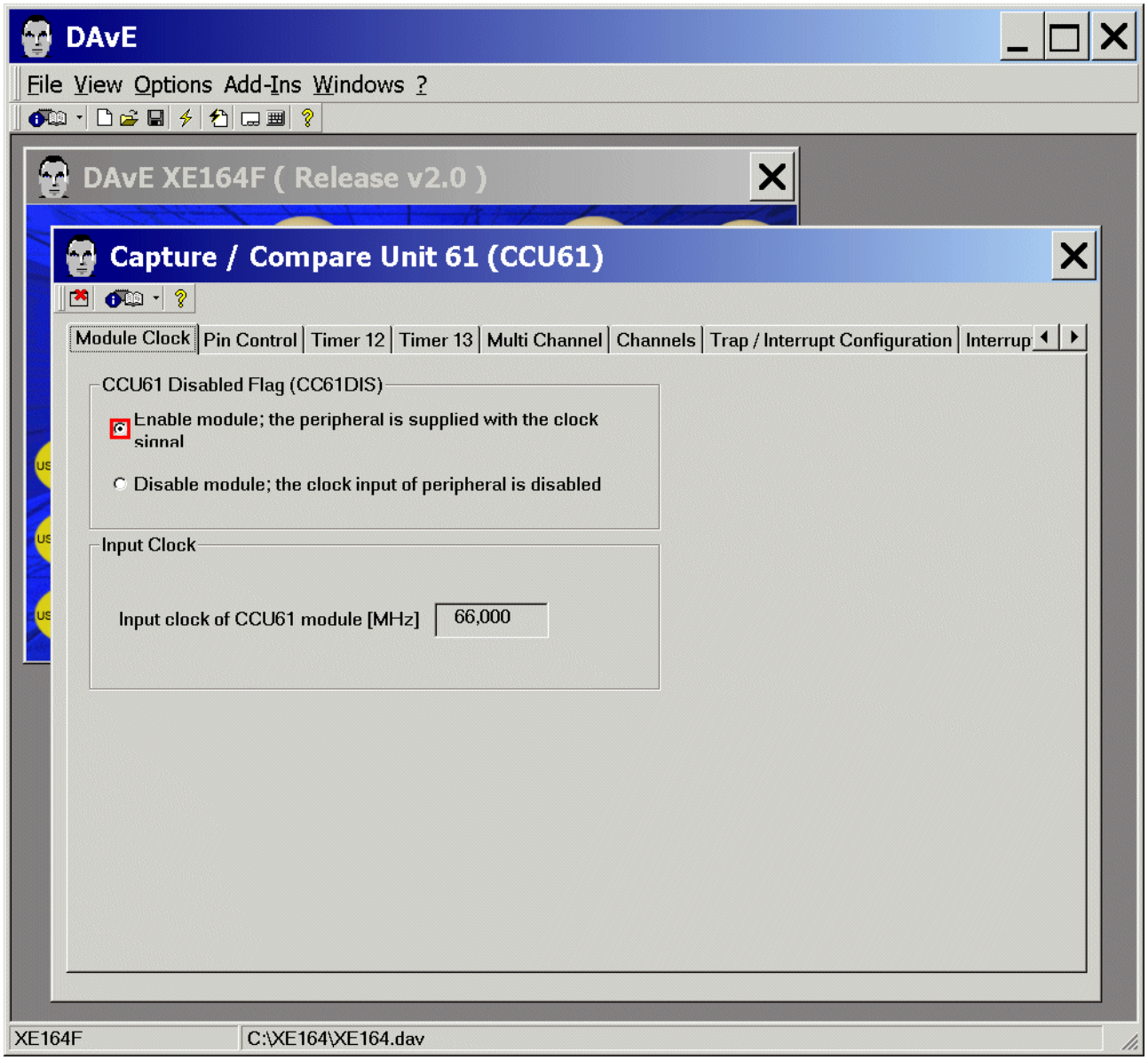
Exit and Save this dialog now by clicking  the close button.

Configuration of the CCU61 module:

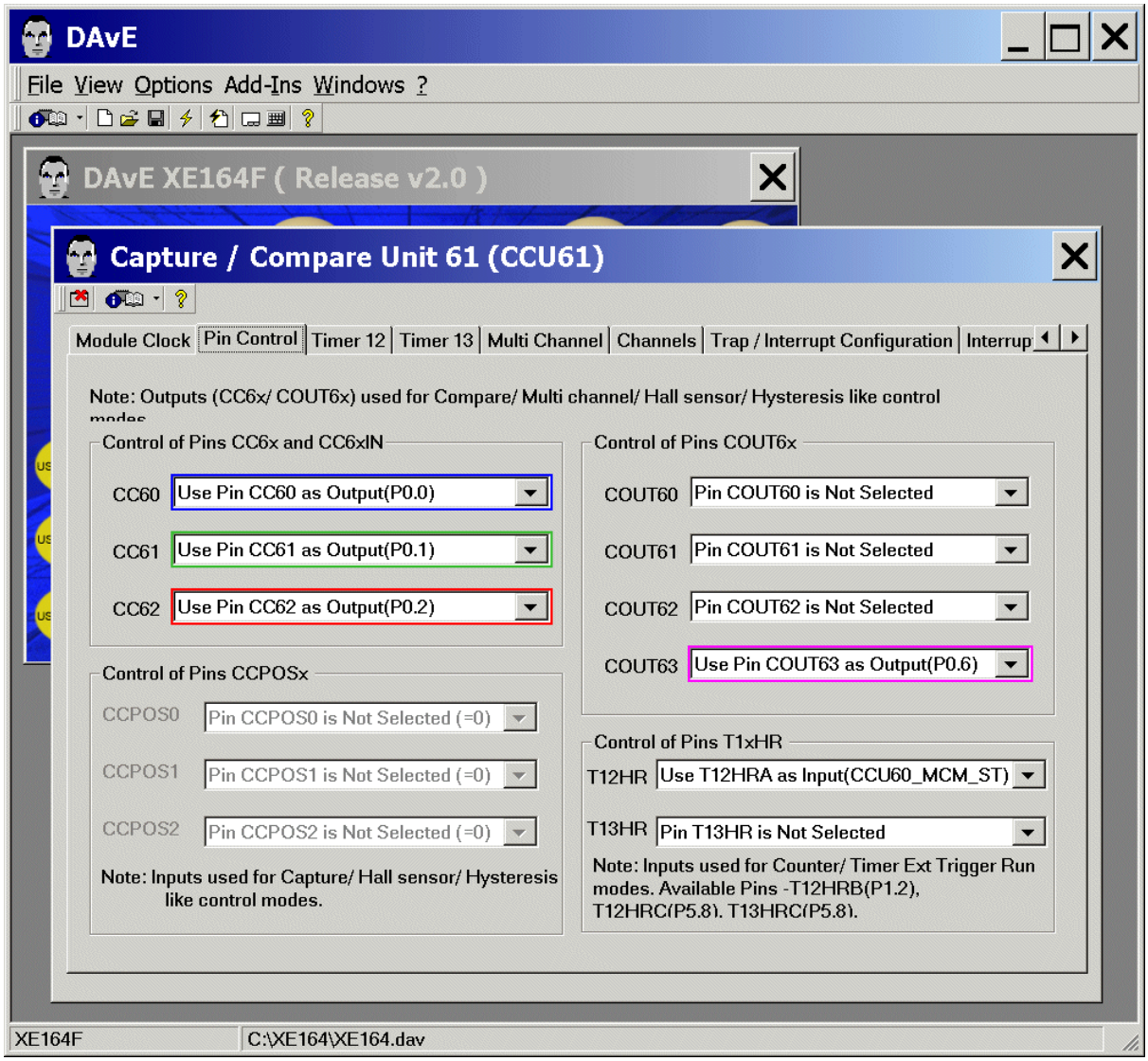
The configuration window/dialog can be opened by clicking the specific block/module (CCU61).



CCU61: Module Clock: click ⊙ Enable module



CCU61:
Pin Control: Control of Pins CC6x and CC6xIN: CC60: select Use pin CC60 as Output (P0.0)
CCU61:
Pin Control: Control of Pins CC6x and CC6xIN: CC61: select Use pin CC61 as Output (P0.1)
CCU61:
Pin Control: Control of Pins CC6x and CC6xIN: CC62: select Use pin CC62 as Output (P0.2)
CCU61:
Pin Control: Control of Pins COUT6x: COUT63: select Use pin COUT63 as Output (P0.6)

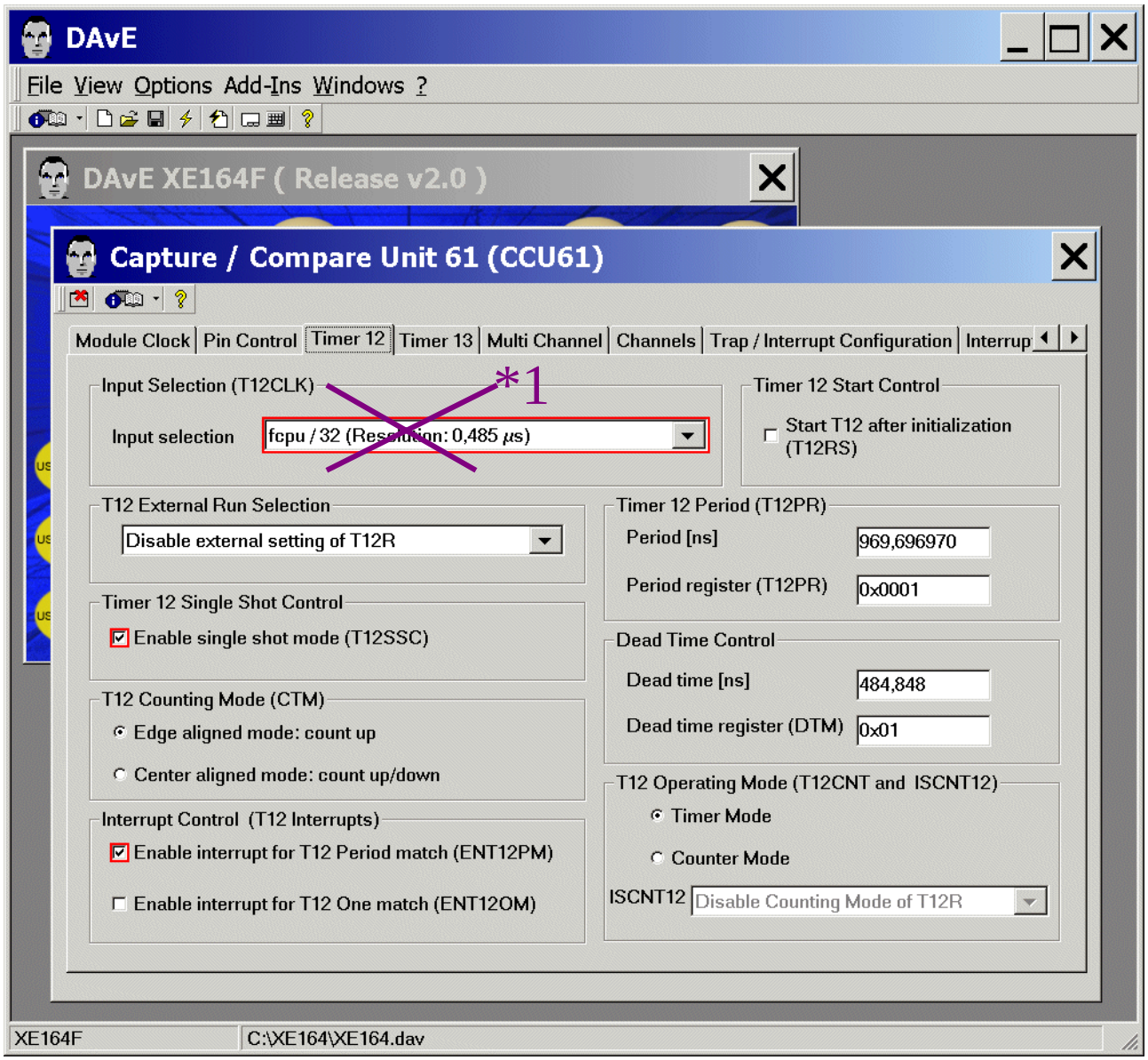


Remember:
Port_0 pins used by our PWM module CCU61:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P0.0	CCU61_CC60	100 (note length, Timer_12)
P0.1	CCU61_CC61	100 (note length, Timer_12)
P0.2	CCU61_CC62	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P0.6	CCU61_COUT63	50 (note frequency, Timer_13)

Timer 12: “note length”:

CCU61: Timer 12: Input Selection: Input selection: choose $f_{CPU}/32$ ➔ Resolution = 124,12 µs *1
CCU61: Timer 12: T12 Single Shot Control: click ☑ Enable single shot mode (T12SSC)
CCU61: Timer 12: Interrupt Control: click ☑ Enable interrupt for T12 Period match

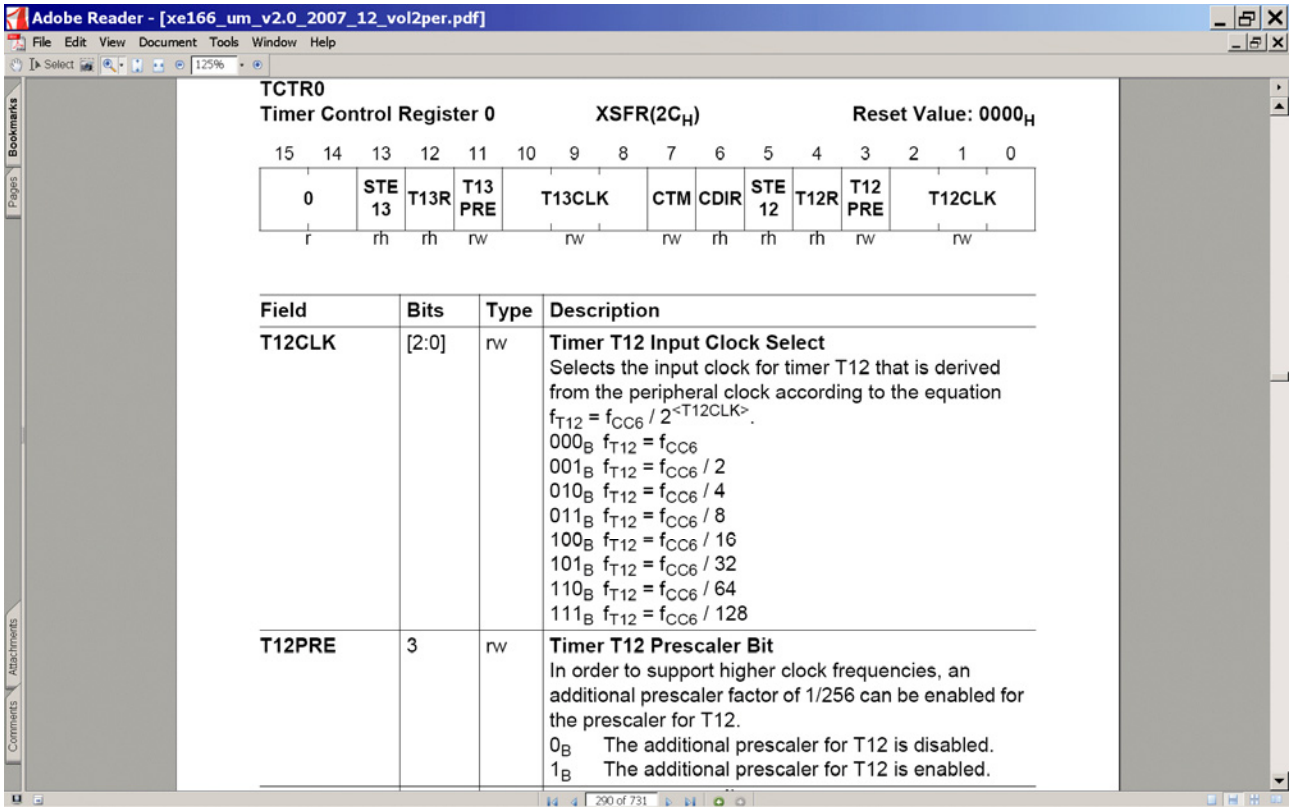


*1:
Timer 12 Resolution:
66 MHz / 256 (T12PRE=1, done by software) / 32 = 8.056,64 Hz ➔ Resolution = 124,12 µs


<<< !!! 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_{CCU61} to a maximum of $f_{CCU61}/128$ 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.



TCTR0
Timer Control Register 0 **XSFR($2C_H$)** **Reset Value: 0000_H**

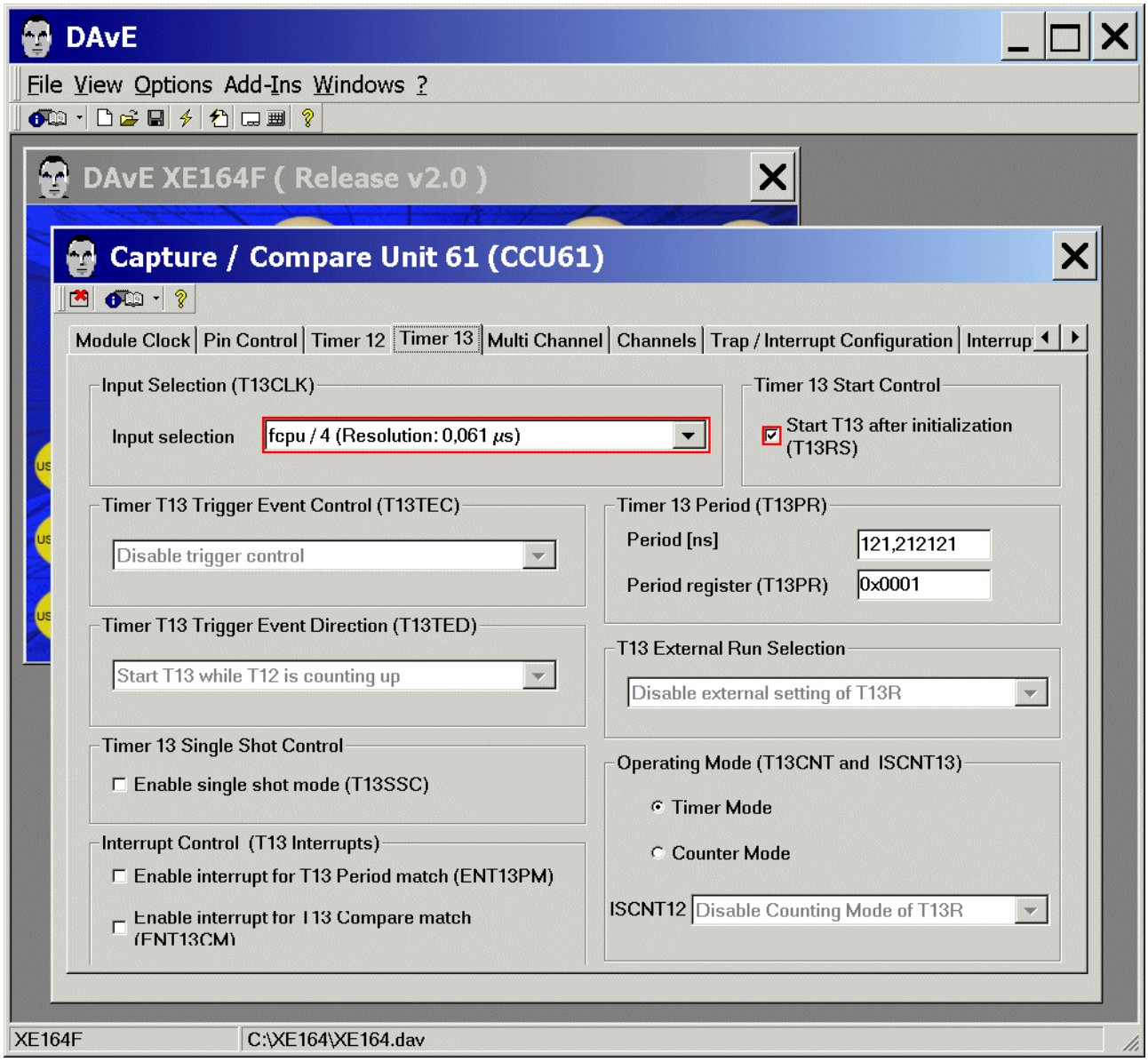
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		STE 13	T13R	T13 PRE	T13CLK			CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK		
r		rh	rh	rw	rw			rw	rh	rh	rh	rw	rw		

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_B $f_{T12} = f_{CC6}$ 001_B $f_{T12} = f_{CC6} / 2$ 010_B $f_{T12} = f_{CC6} / 4$ 011_B $f_{T12} = f_{CC6} / 8$ 100_B $f_{T12} = f_{CC6} / 16$ 101_B $f_{T12} = f_{CC6} / 32$ 110_B $f_{T12} = f_{CC6} / 64$ 111_B $f_{T12} = f_{CC6} / 128$
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_B The additional prescaler for T12 is disabled. 1_B 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

Timer 13: ”note frequency”:

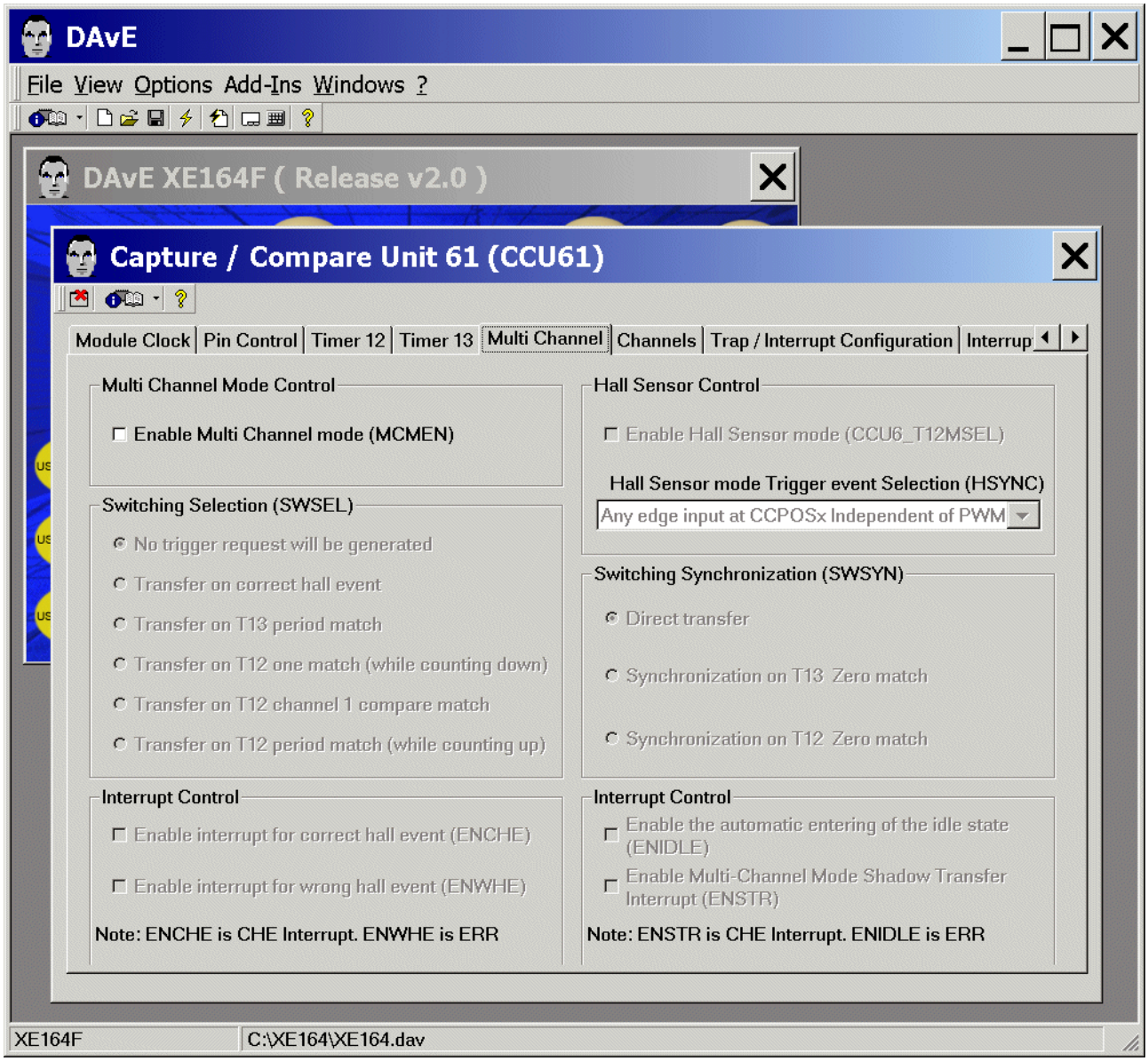
CCU61: Timer T13: Input Selection: Input selection select $f_{CPU}/4$ (Resolution: 60,606 ns)
CCU61: Timer T13: Timer 13 Start Control: click ✓ Start T13 after initialization (T13RS)



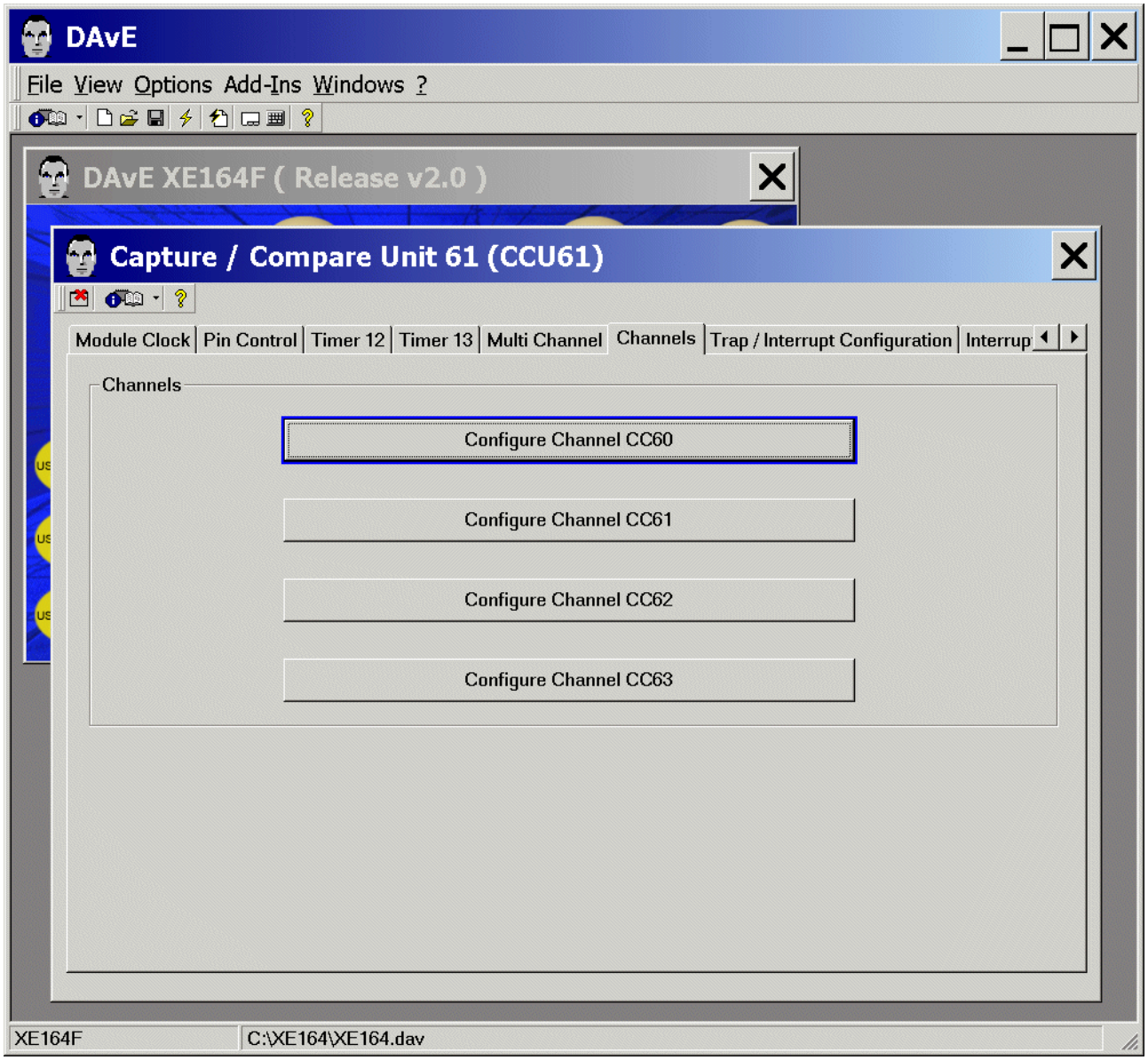
Note:
Timer 13 resolution = $1/(f_{CPU}/4) = 1/(66MHz/4) = 60,606$ ns.

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

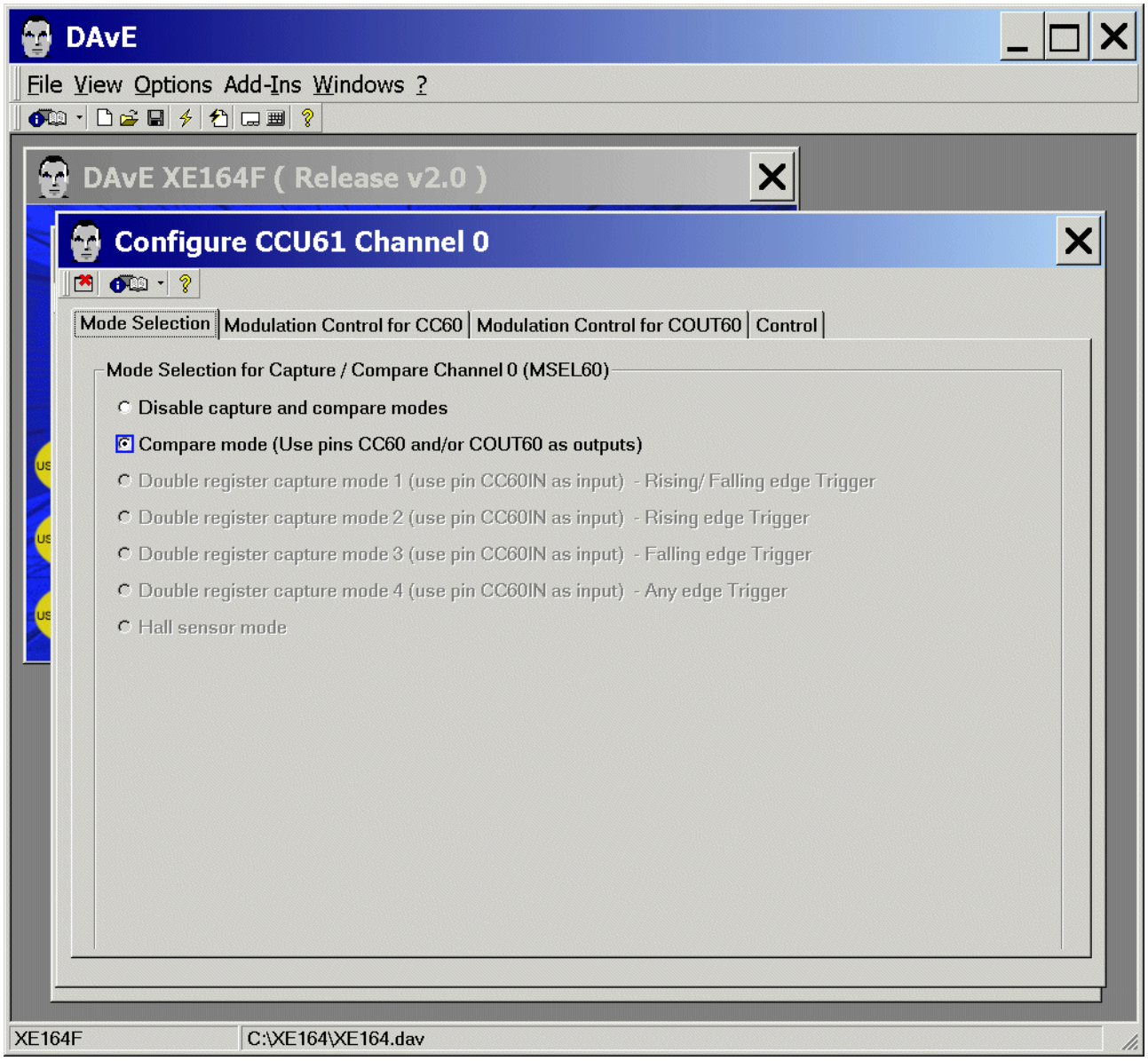
CCU61: Multi Channel: (do nothing)



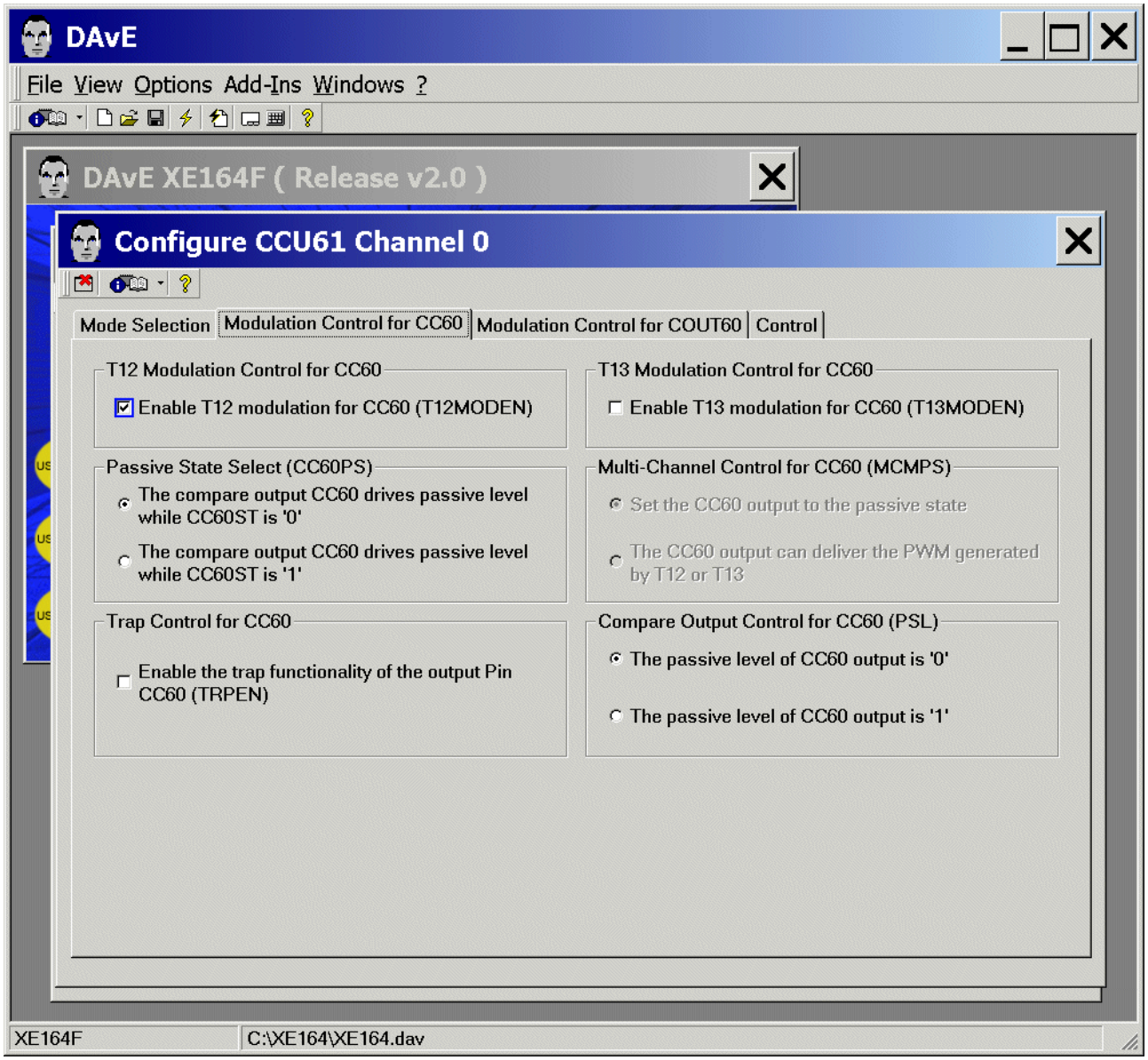
CCU61: Channels: click Configure Channel 0



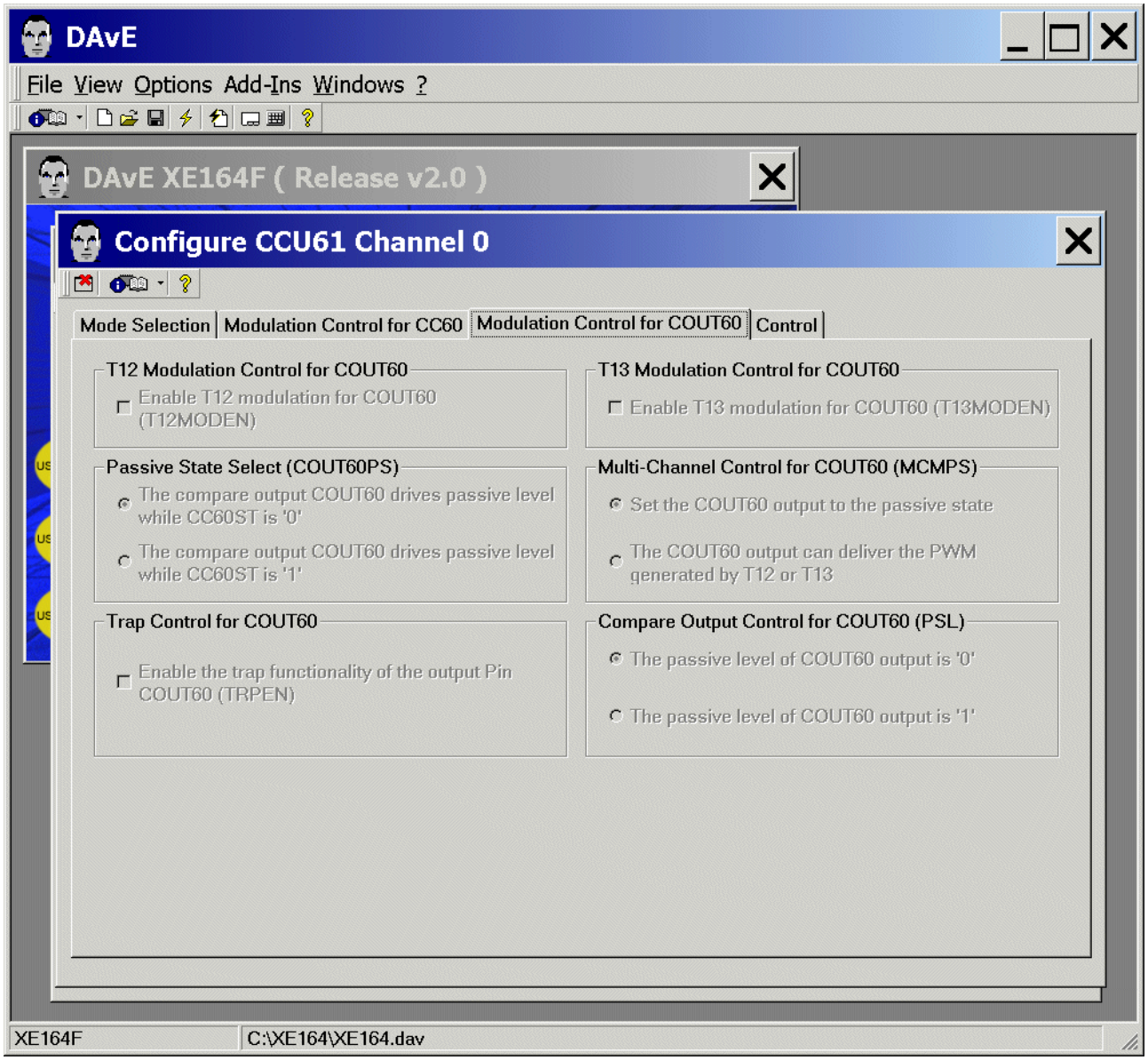
CCU61: Channels: Configure Channel 0:
Mode Selection: Mode Selection for Capture / Compare Channel 0: click ⊙ Compare mode



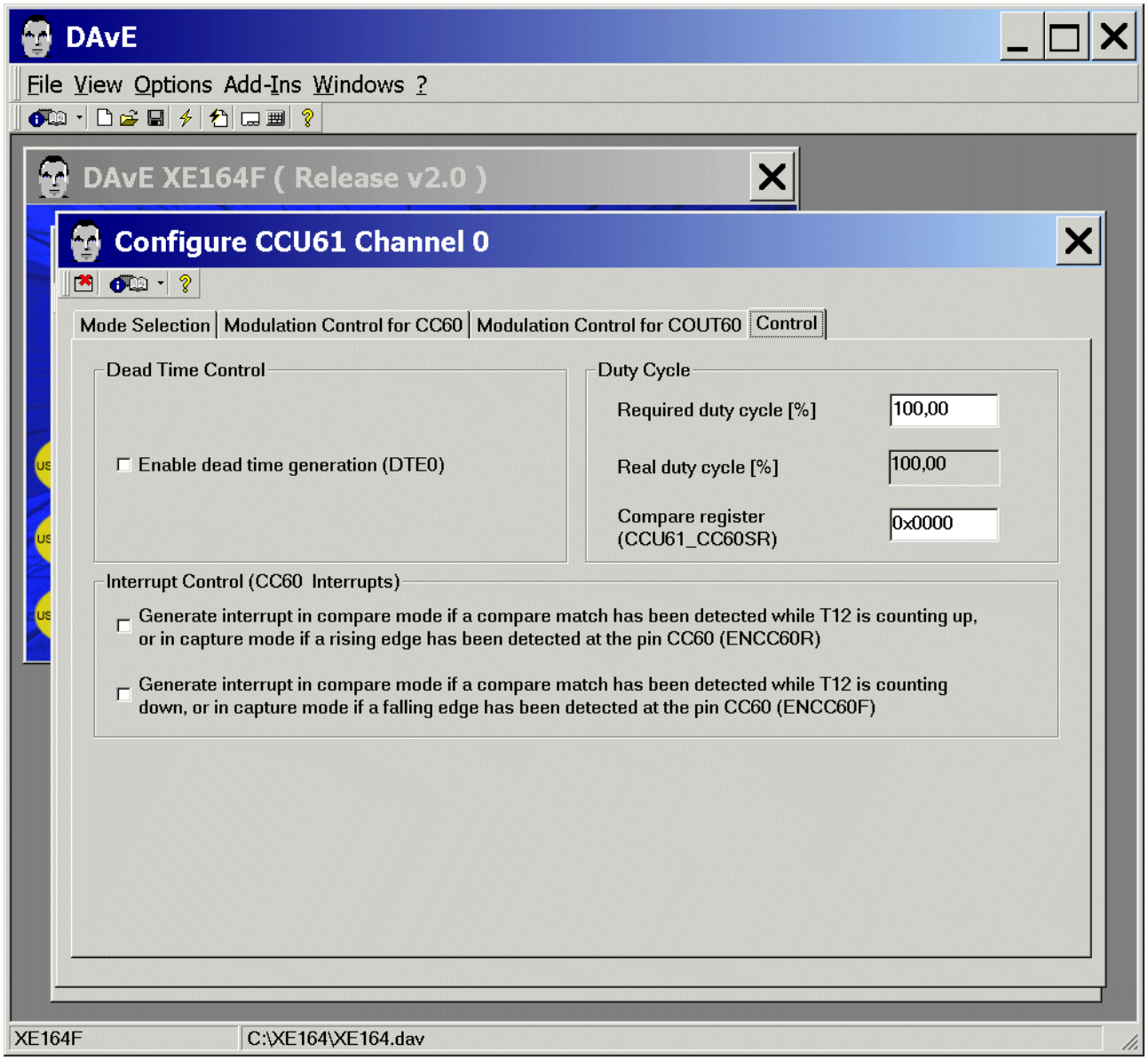
CCU61: Channels: Configure Channel 0:
Modulation Control for CC60:
T12 Modulation Control for CC60: click ☑ Enable T12 modulation for CC60



CCU61: Channels: Configure Channel 0: Modulation Control for COUT60: (do nothing)

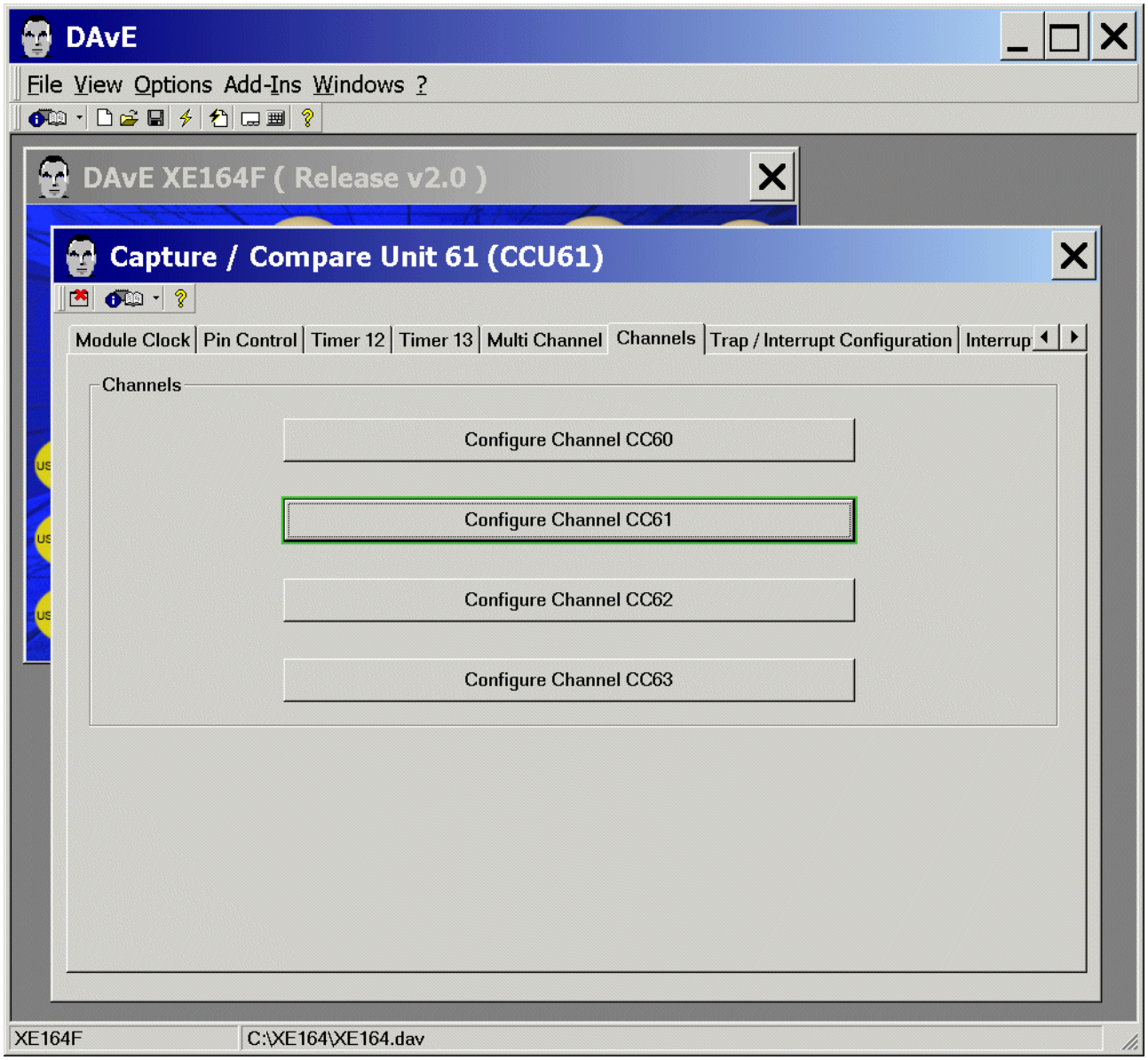


CCU61: Channels: Configure Channel 0: Control: (do nothing)

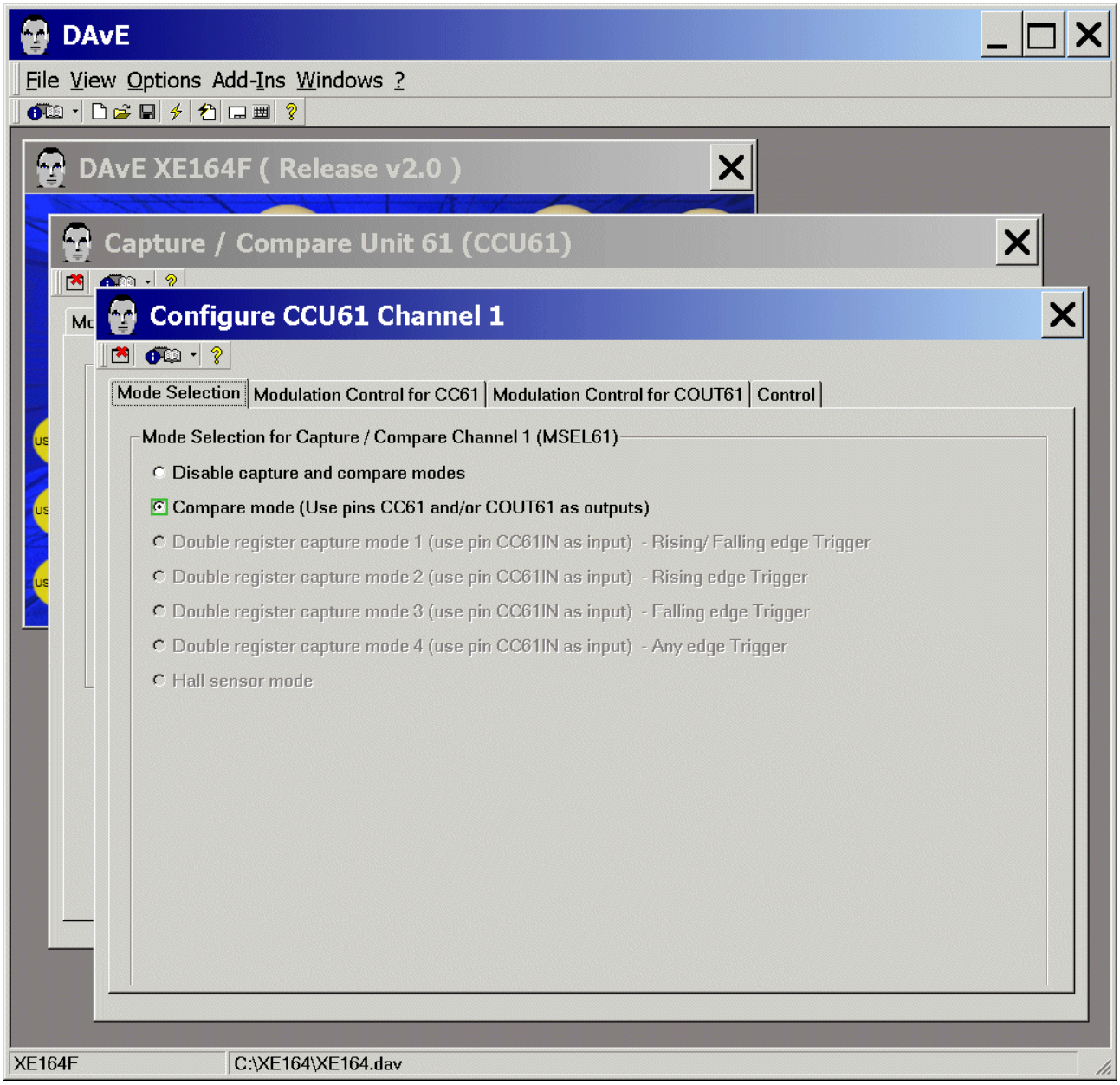


Exit and Save this dialog now by clicking  the close button.

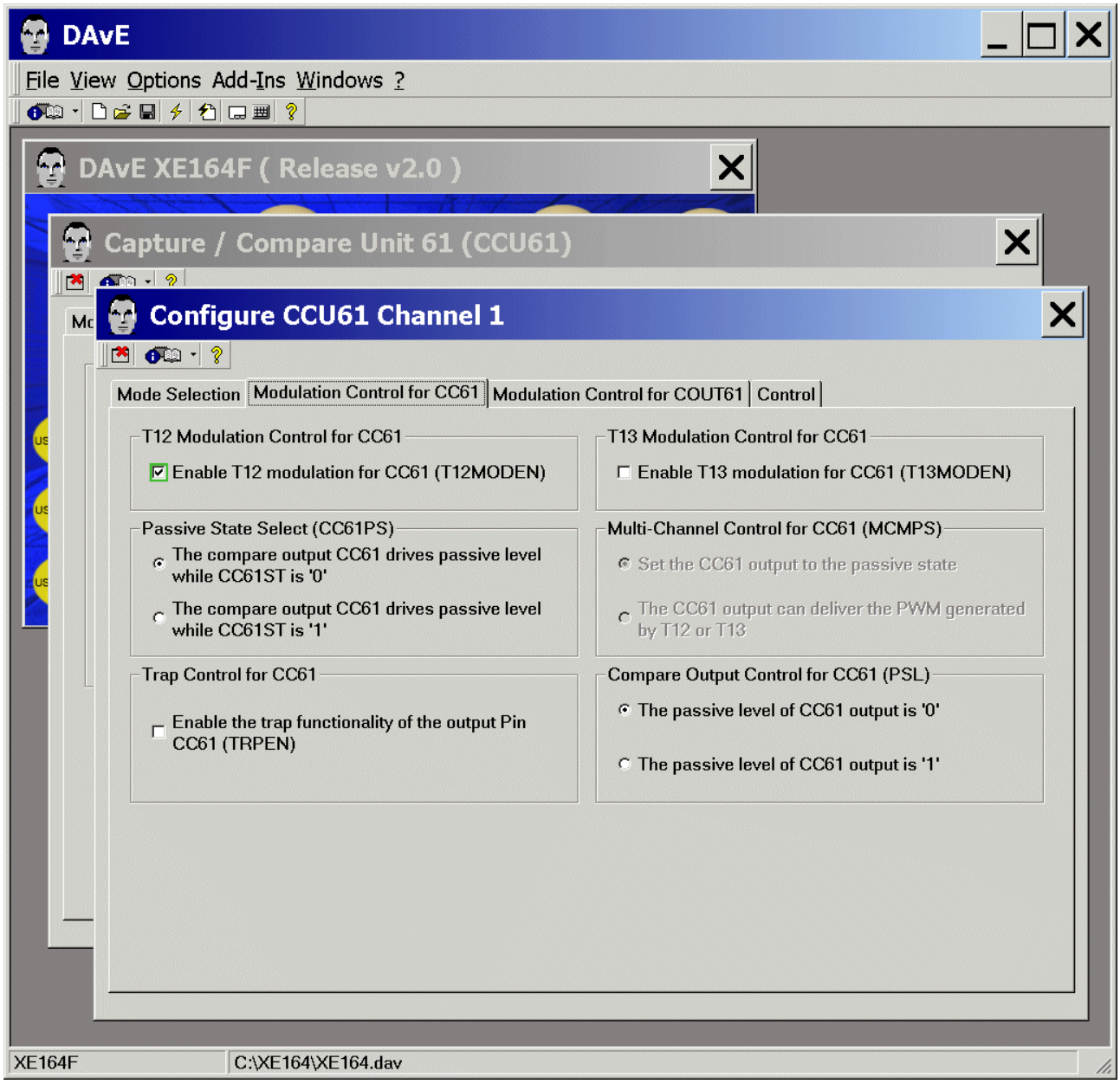
CCU61: Channels: click Configure Channel 1



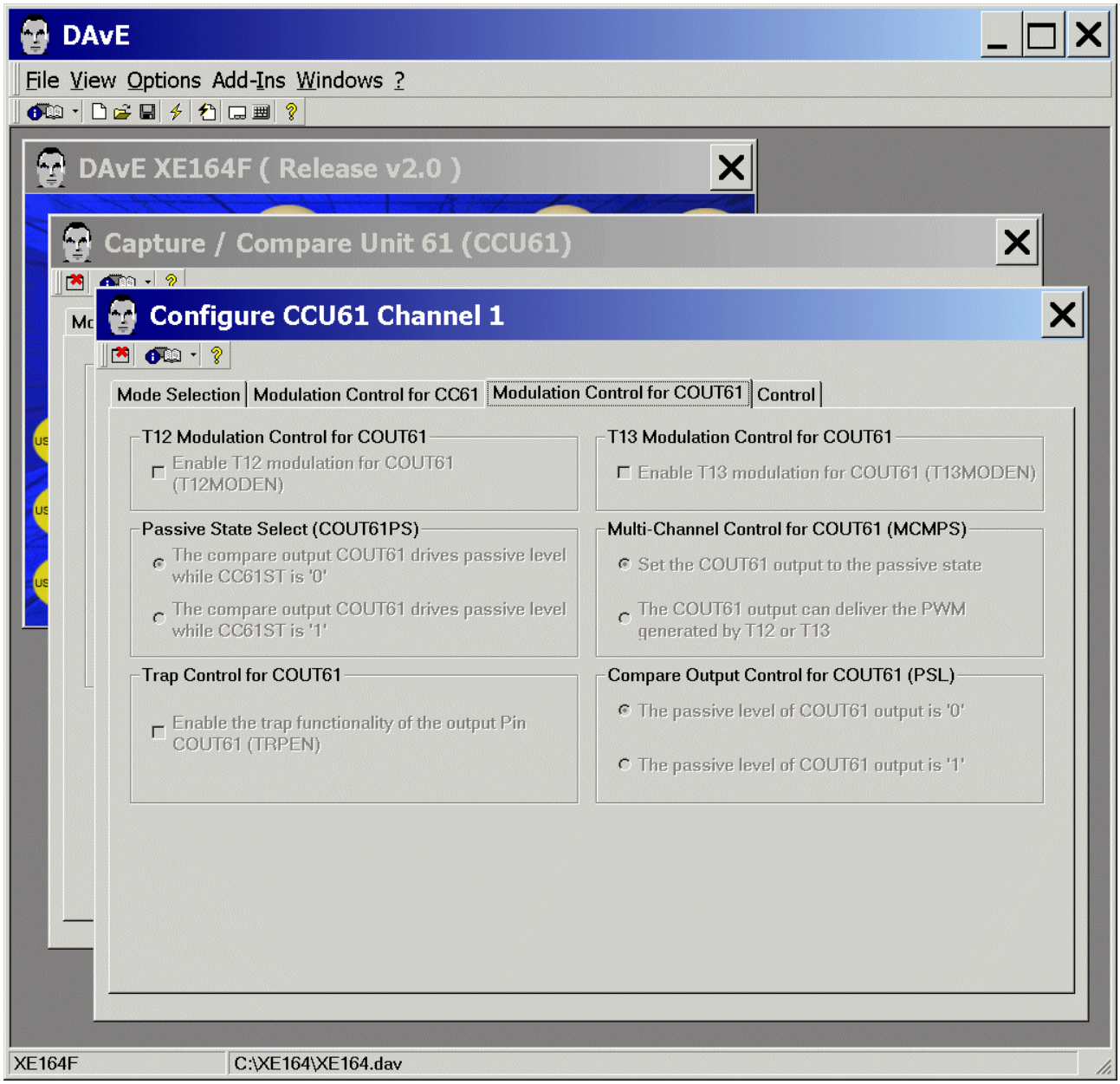
CCU61: Channels: Configure Channel 1:
Mode Selection: Mode Selection for Capture / Compare Channel 1: click ⊙ Compare mode



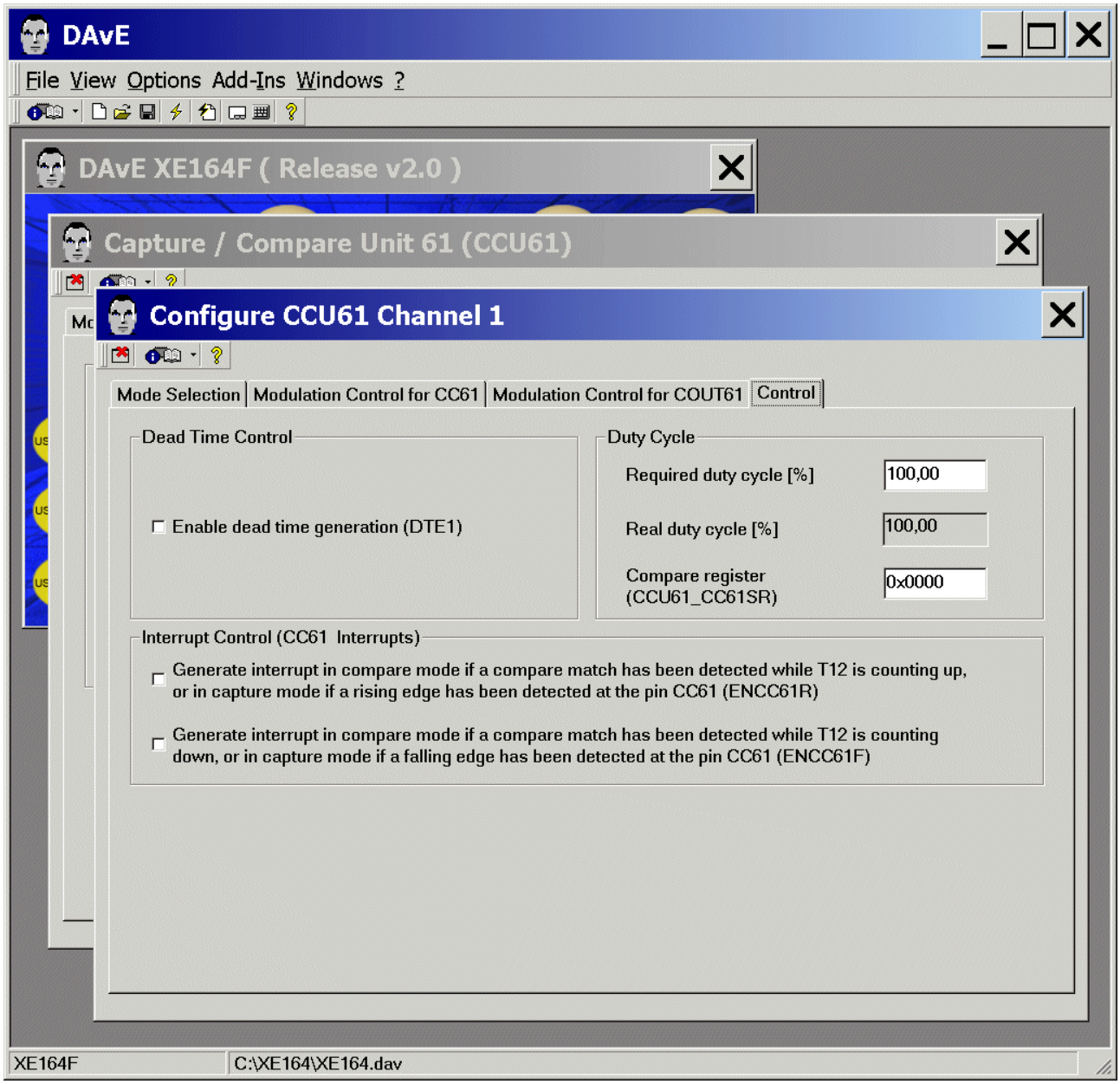
CCU61: Channels: Configure Channel 1:
Modulation Control for CC61:
T12 Modulation Control for CC61: click ☑ Enable T12 modulation for CC61



CCU61: Channels: Configure Channel 1: Modulation Control for COUT61: (do nothing)

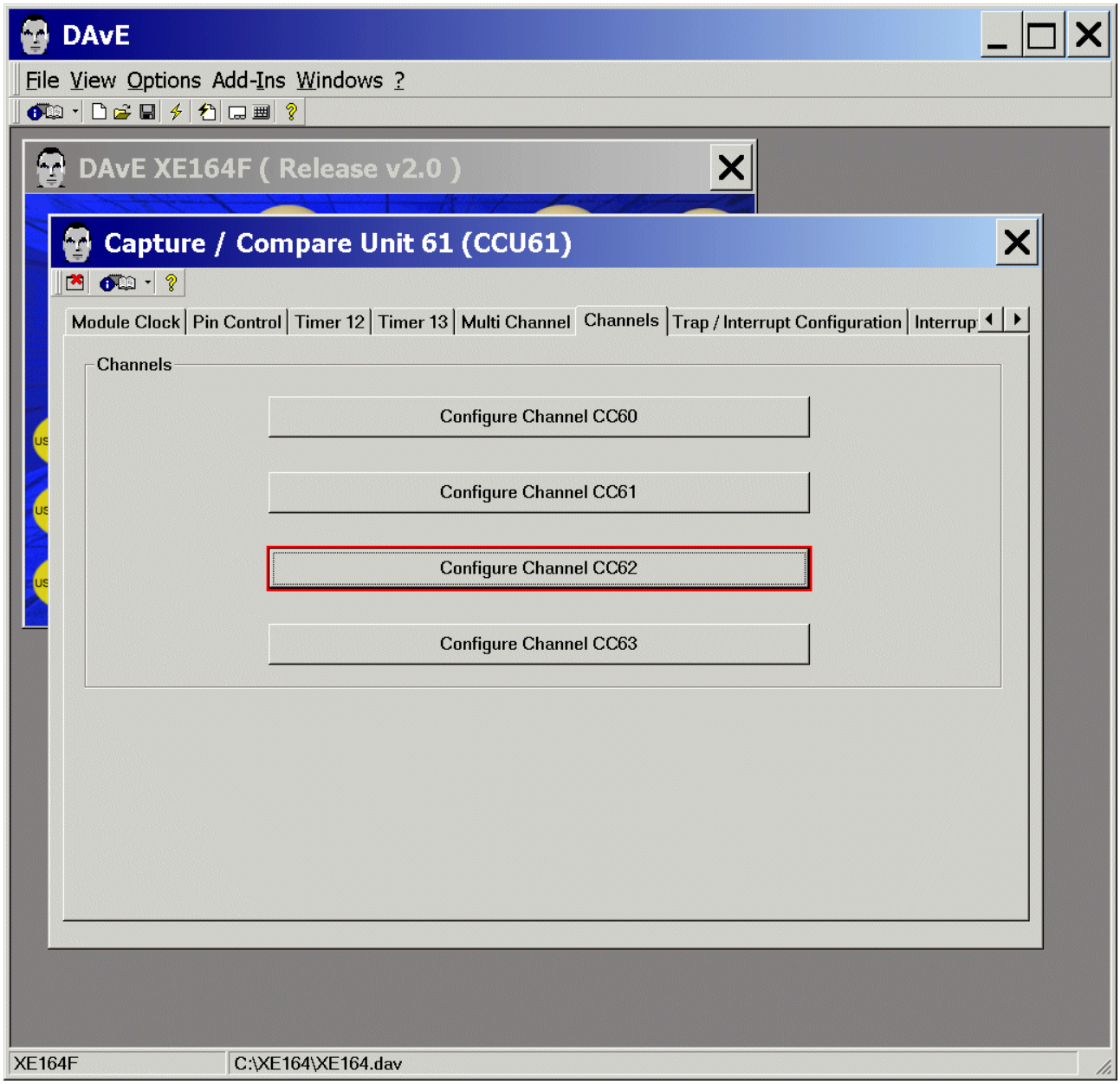


CCU61: Channels: Configure Channel 1: Control: (do nothing)

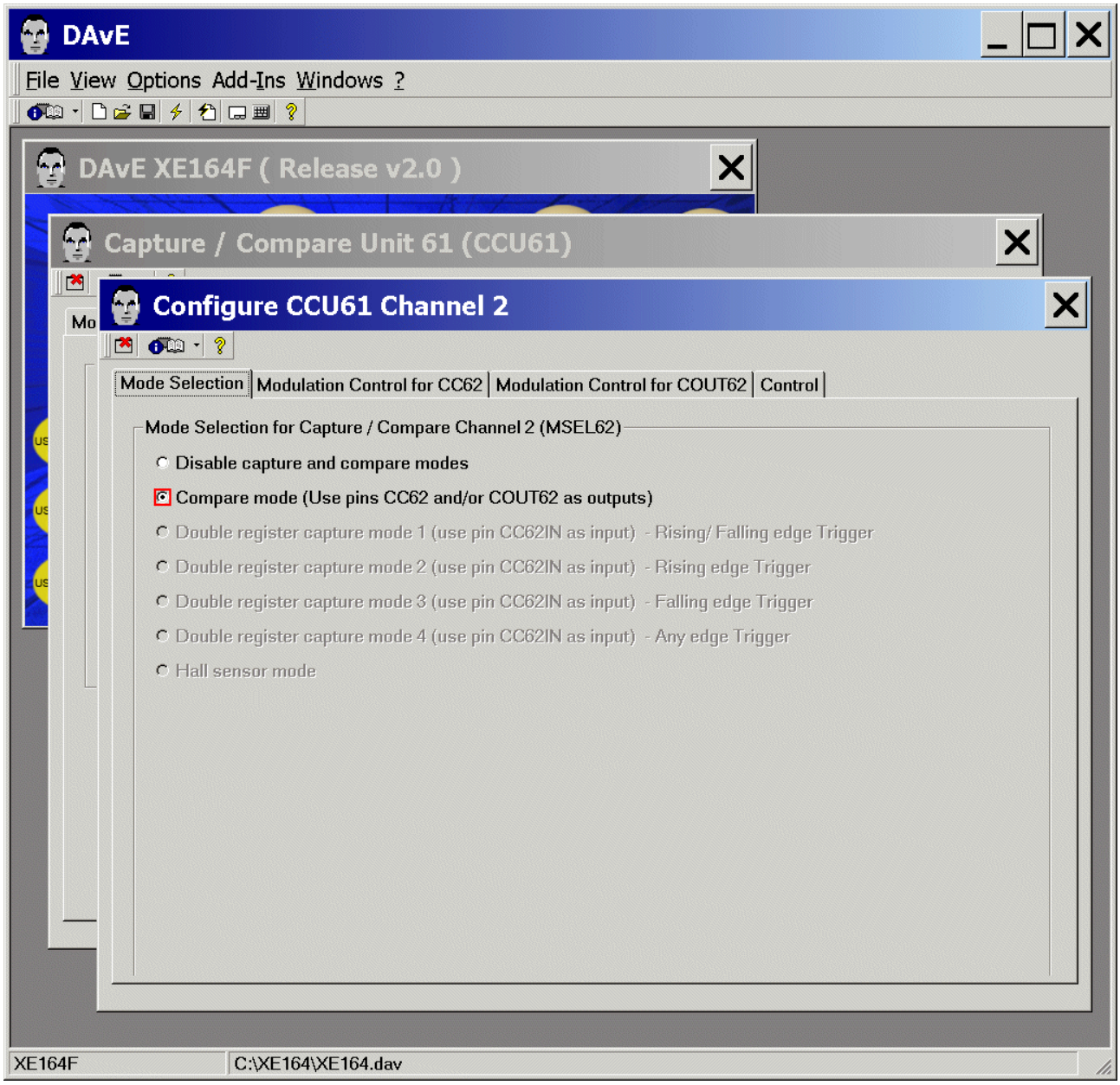


Exit and Save this dialog now by clicking  the close button.

CCU61: Channels: click Configure Channel 2

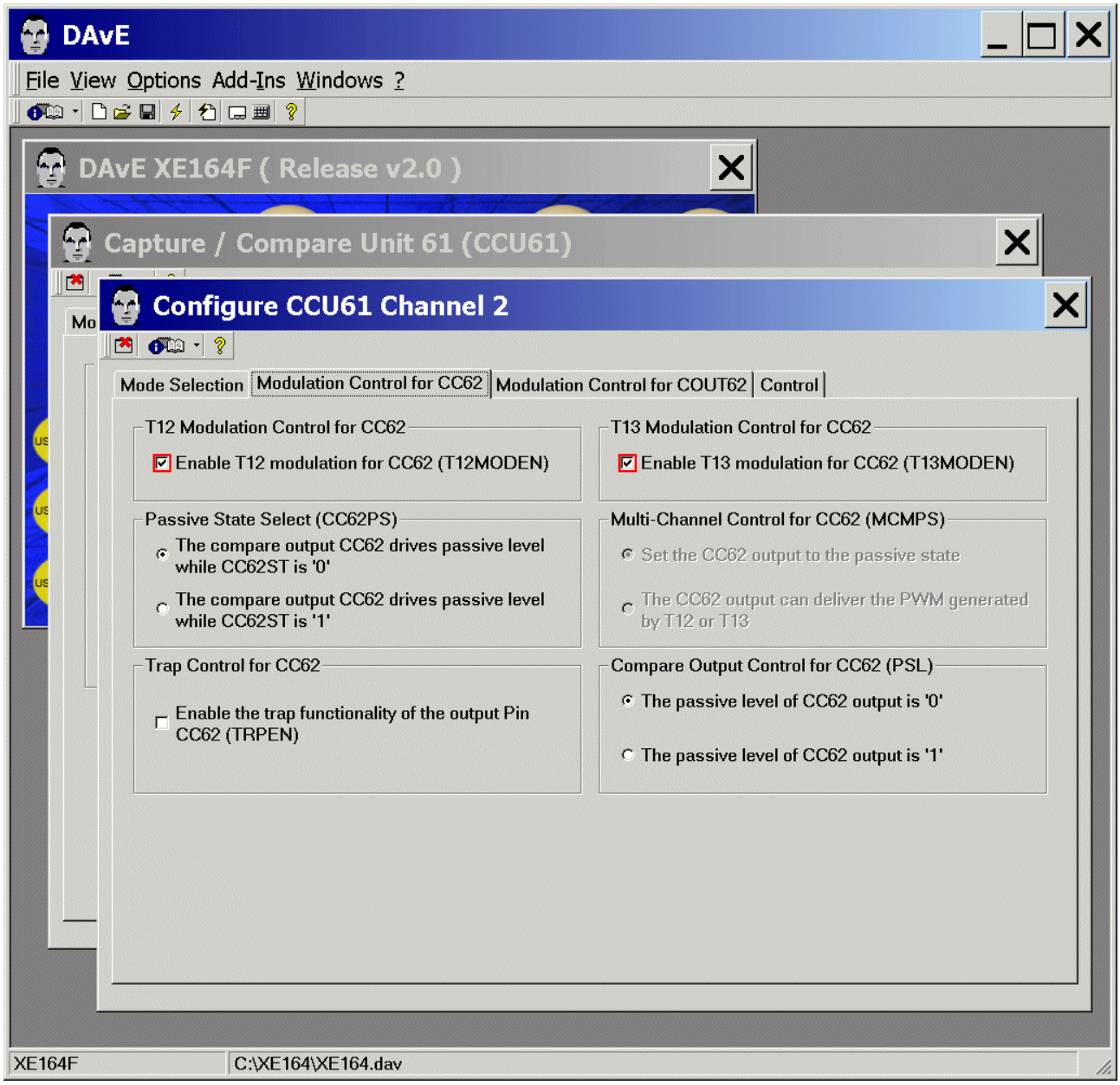


CCU61: Channels: Configure Channel 2:
Mode Selection: Mode Selection for Capture / Compare Channel 2: click ⊙ Compare mode

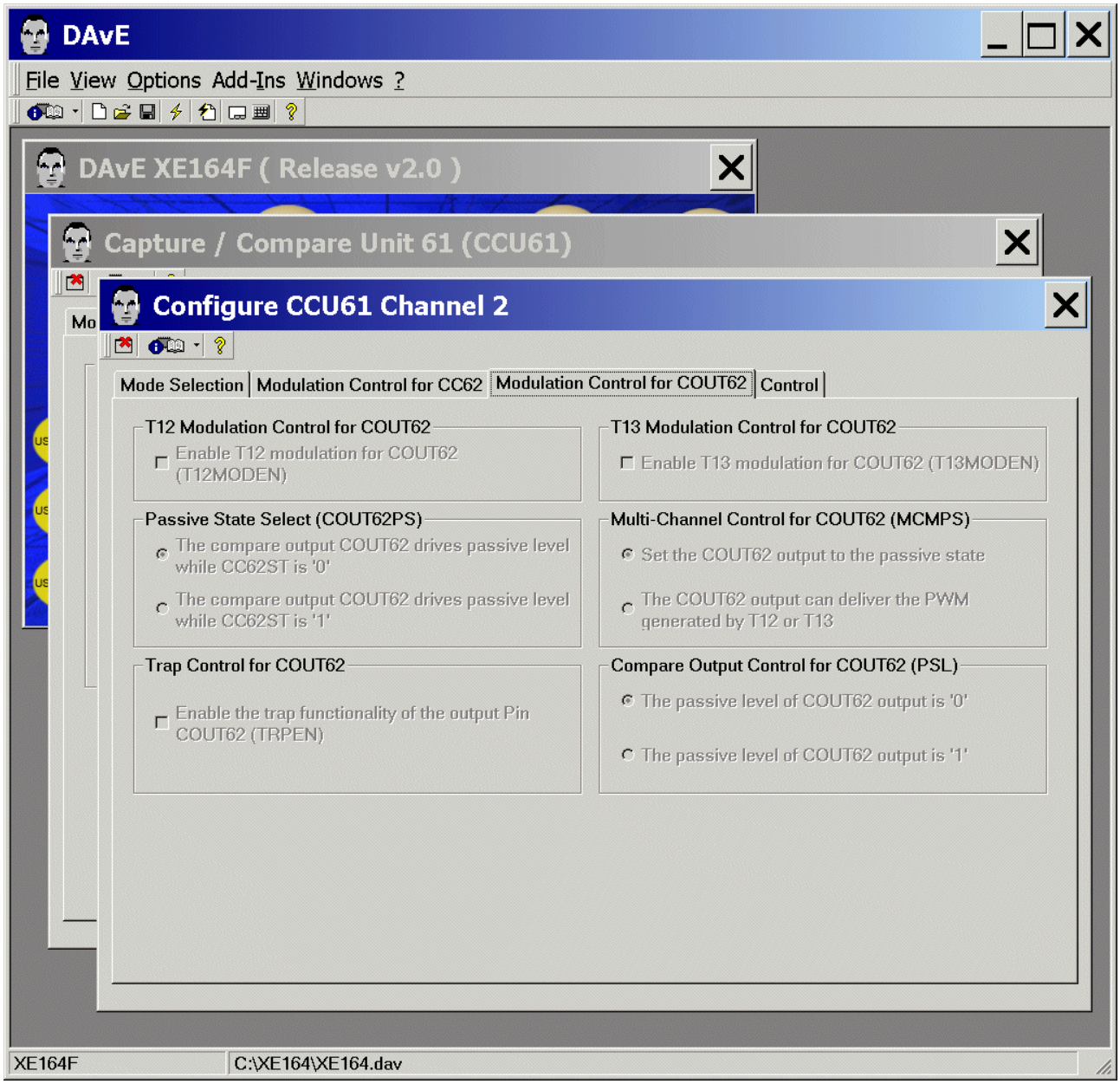


CCU61: Channels: Configure Channel 2:
Modulation Control for CC62:
T12 Modulation Control for CC62: click ☑ Enable T12 modulation for CC62

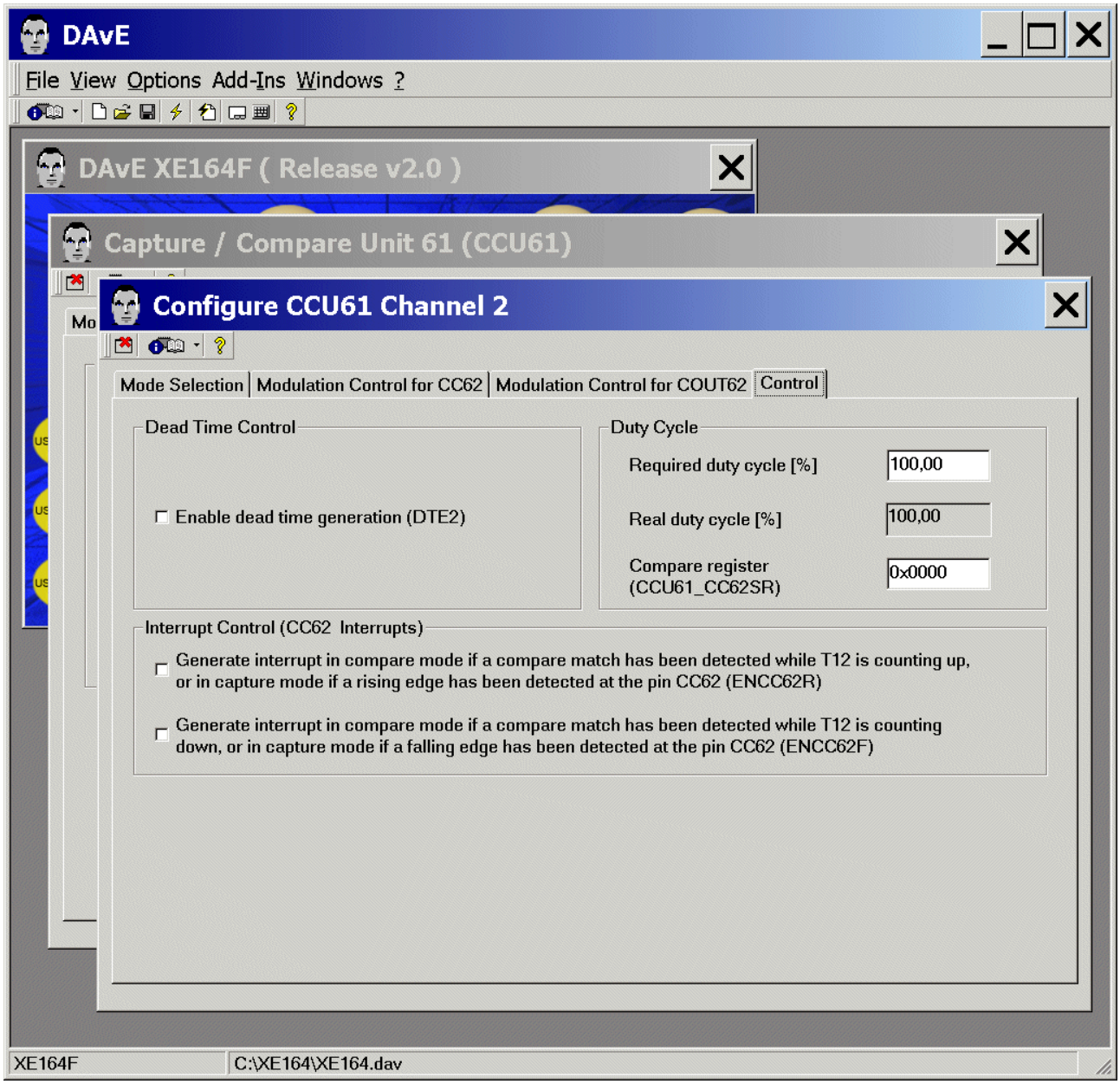
CCU61: Channels: Configure Channel 2:
Modulation Control for CC62:
T13 Modulation Control for CC62: click ☑ Enable T13 modulation for CC62



CCU61: Channels: Configure Channel 2: Modulation Control for COUT62: (do nothing)

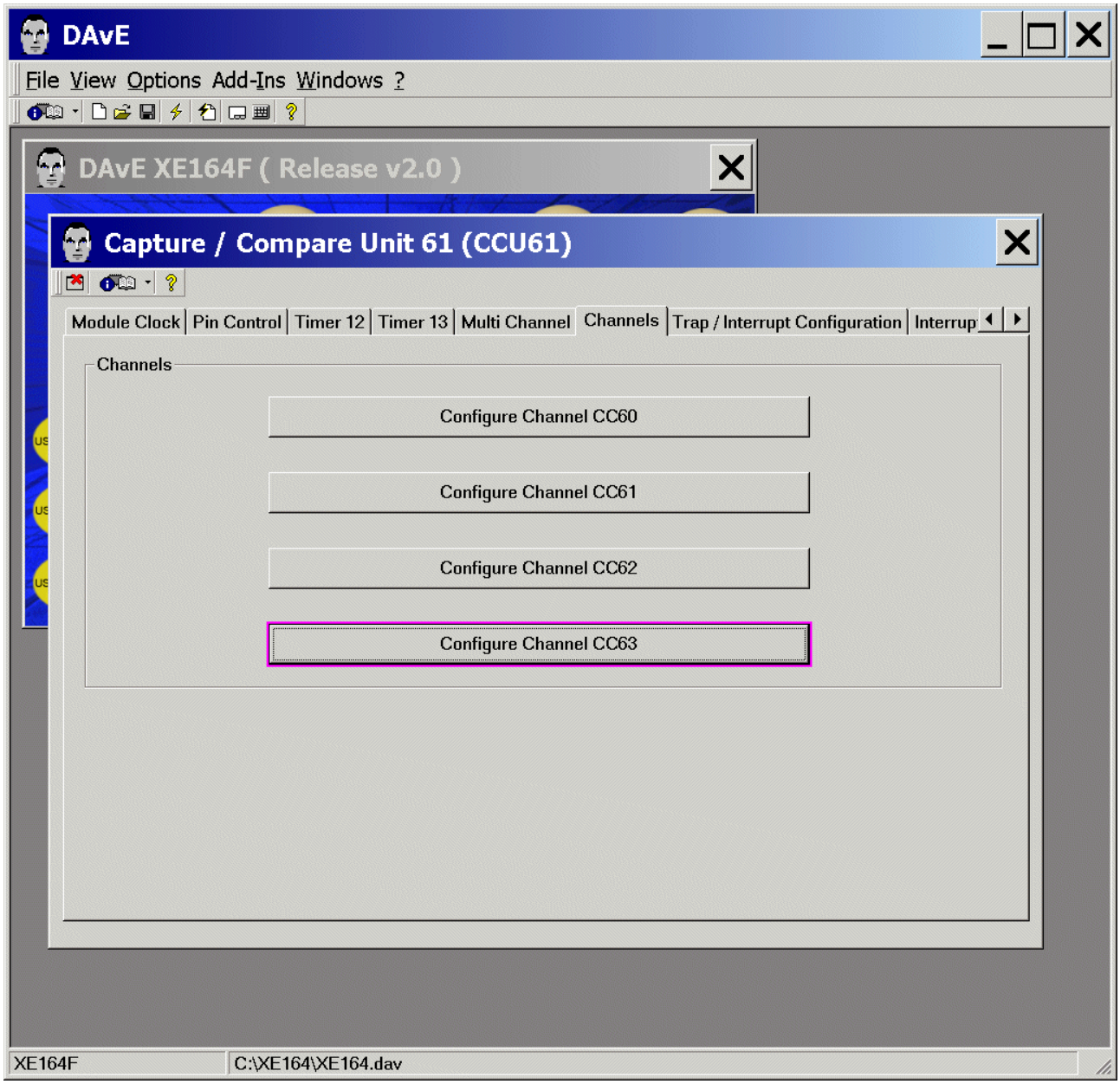


CCU61: Channels: Configure Channel 2: Control: (do nothing)

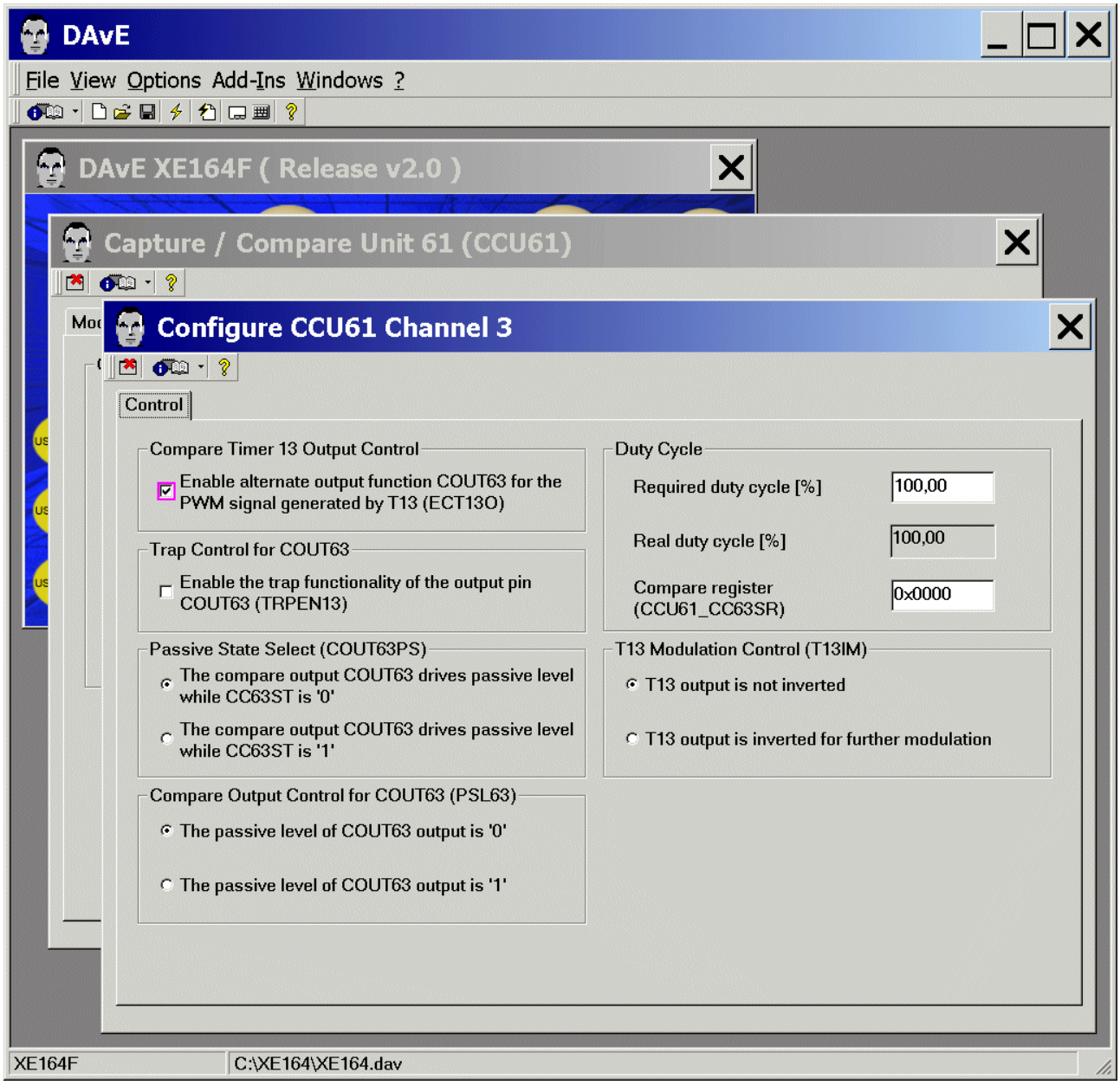


Exit and Save this dialog now by clicking  the close button.

CCU61: Channels: click Configure Channel 3

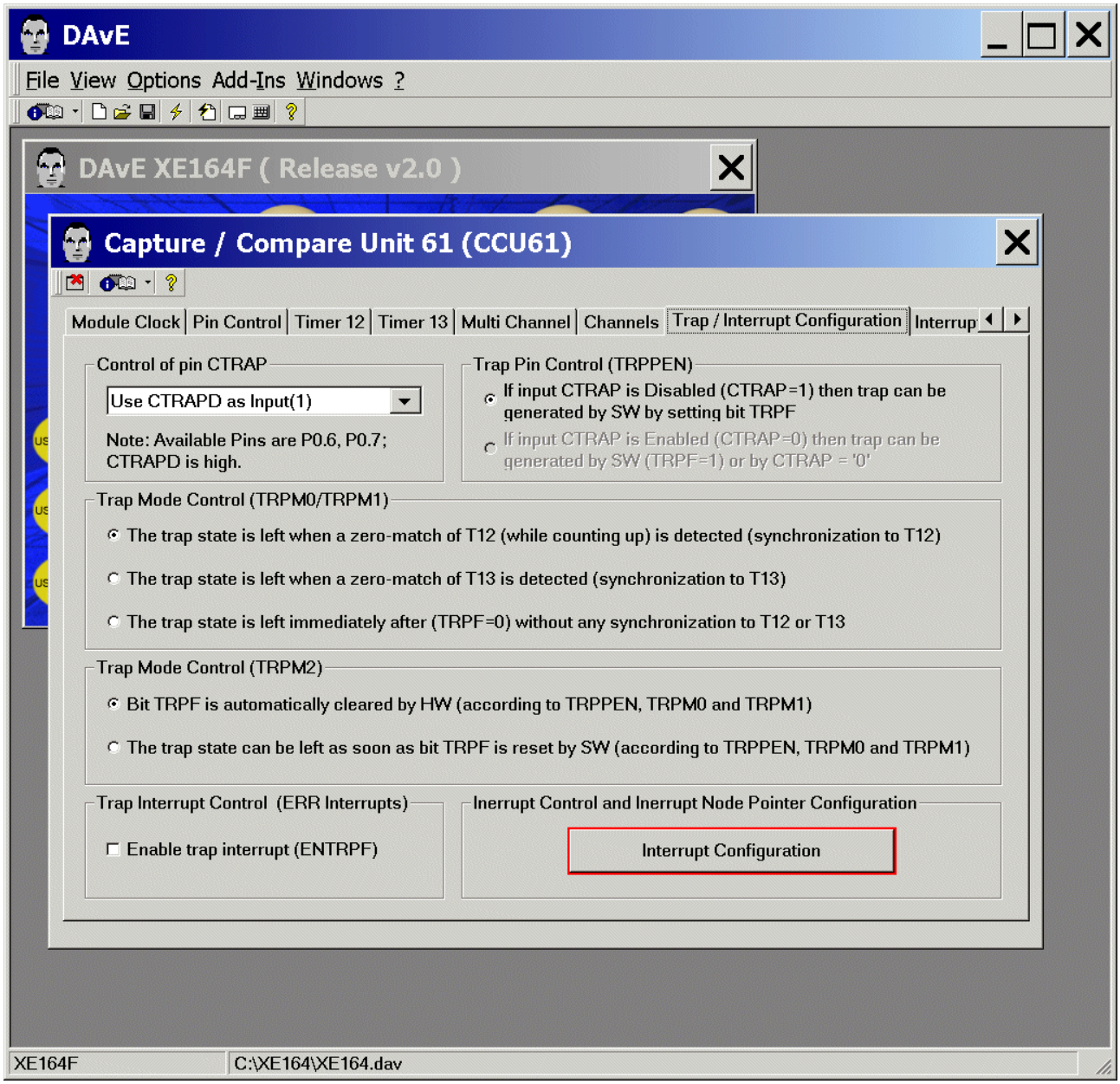


CCU61: Channels: Configure Channel 3:
Control: Compare Timer 13 Output Control: click ☑ Enable alternate output function COUT63

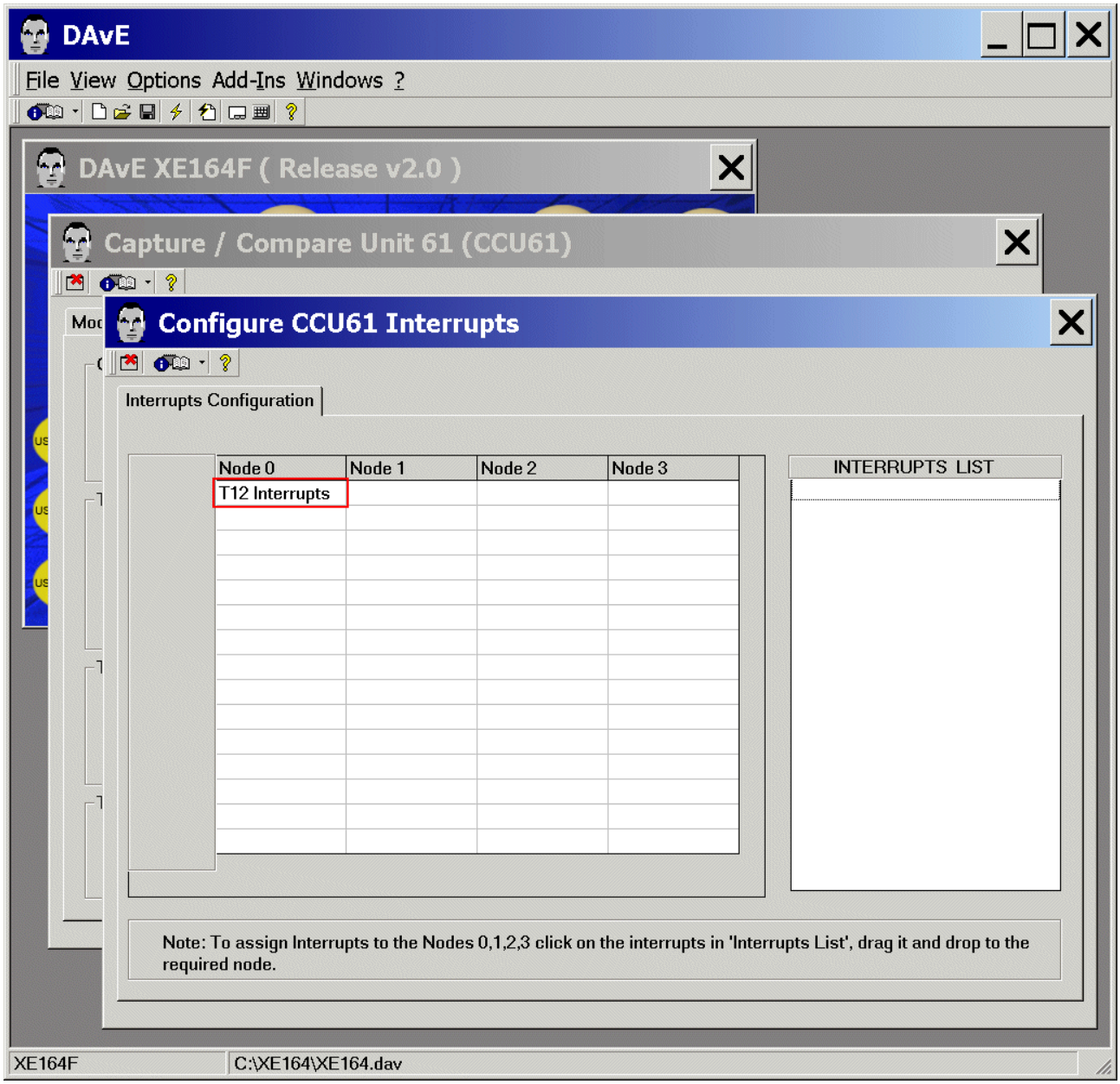


Exit and Save this dialog now by clicking  the close button.

CCU61: Trap / Interrupt Control: click Interrupt Configuration

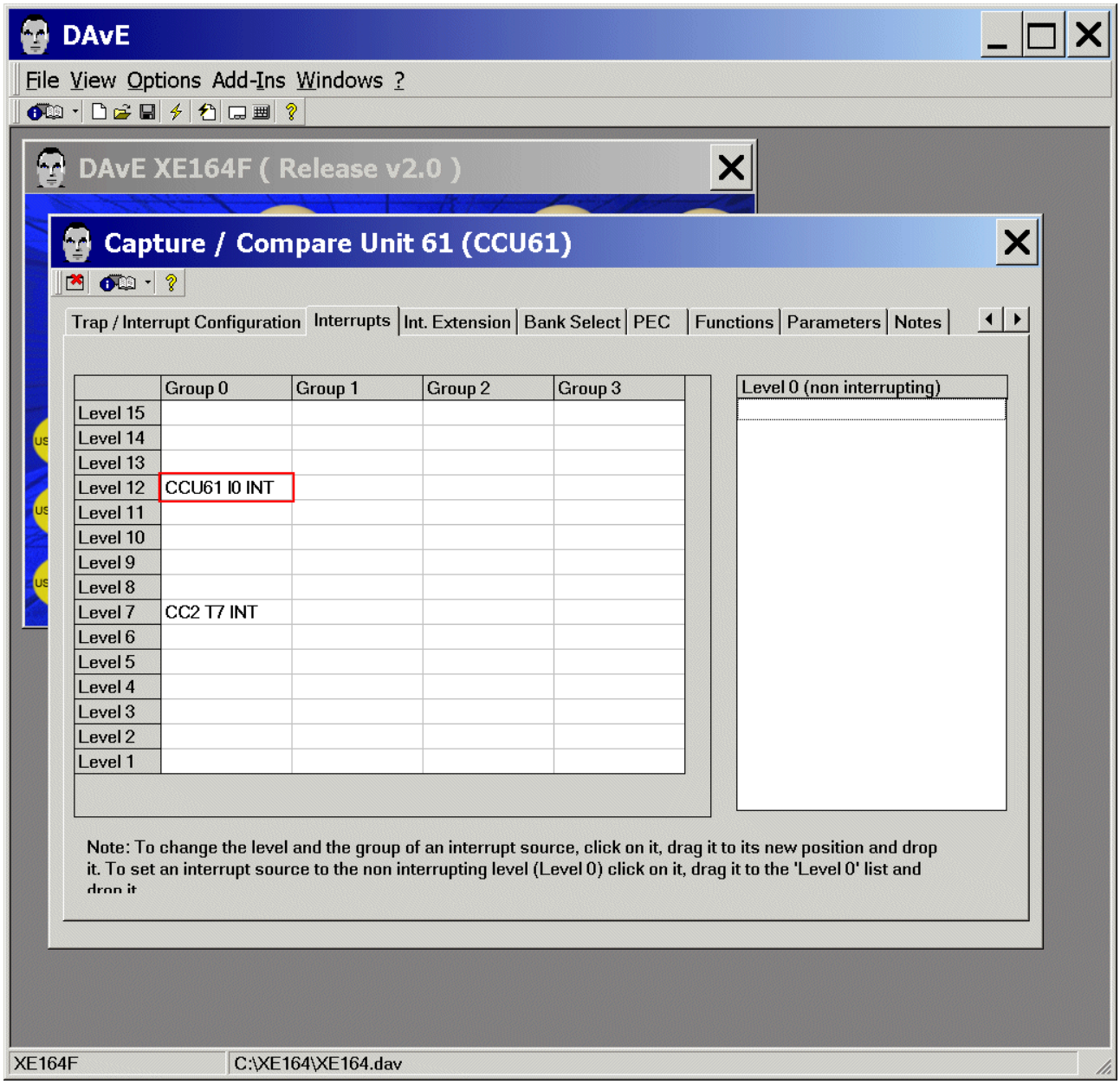


CCU61: Trap / Interrupt Control: Interrupt Configuration:
Interrupts Configuration: drag and drop T12 Interrupts to Node 0

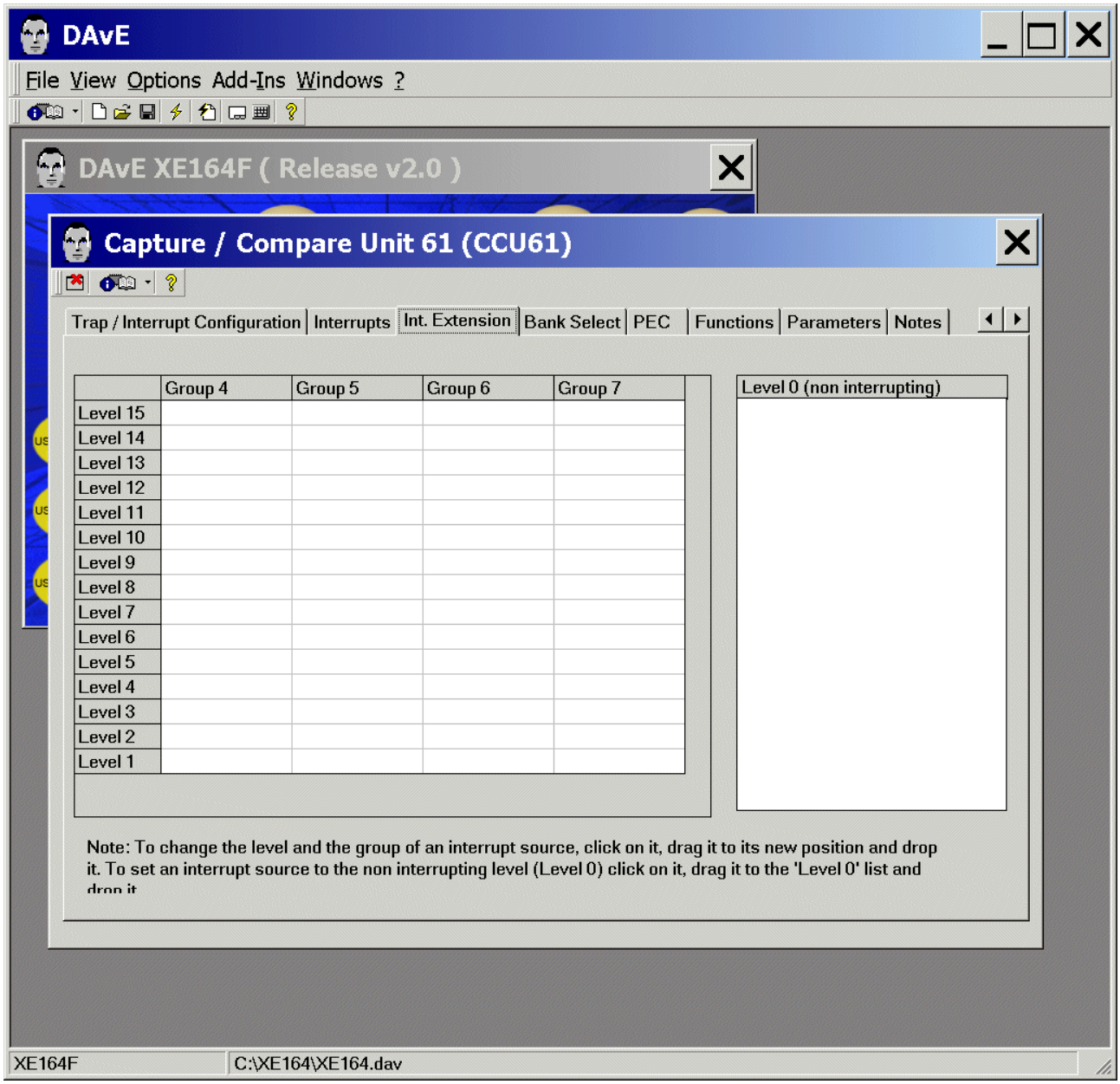


Exit and Save this dialog now by clicking the close button.

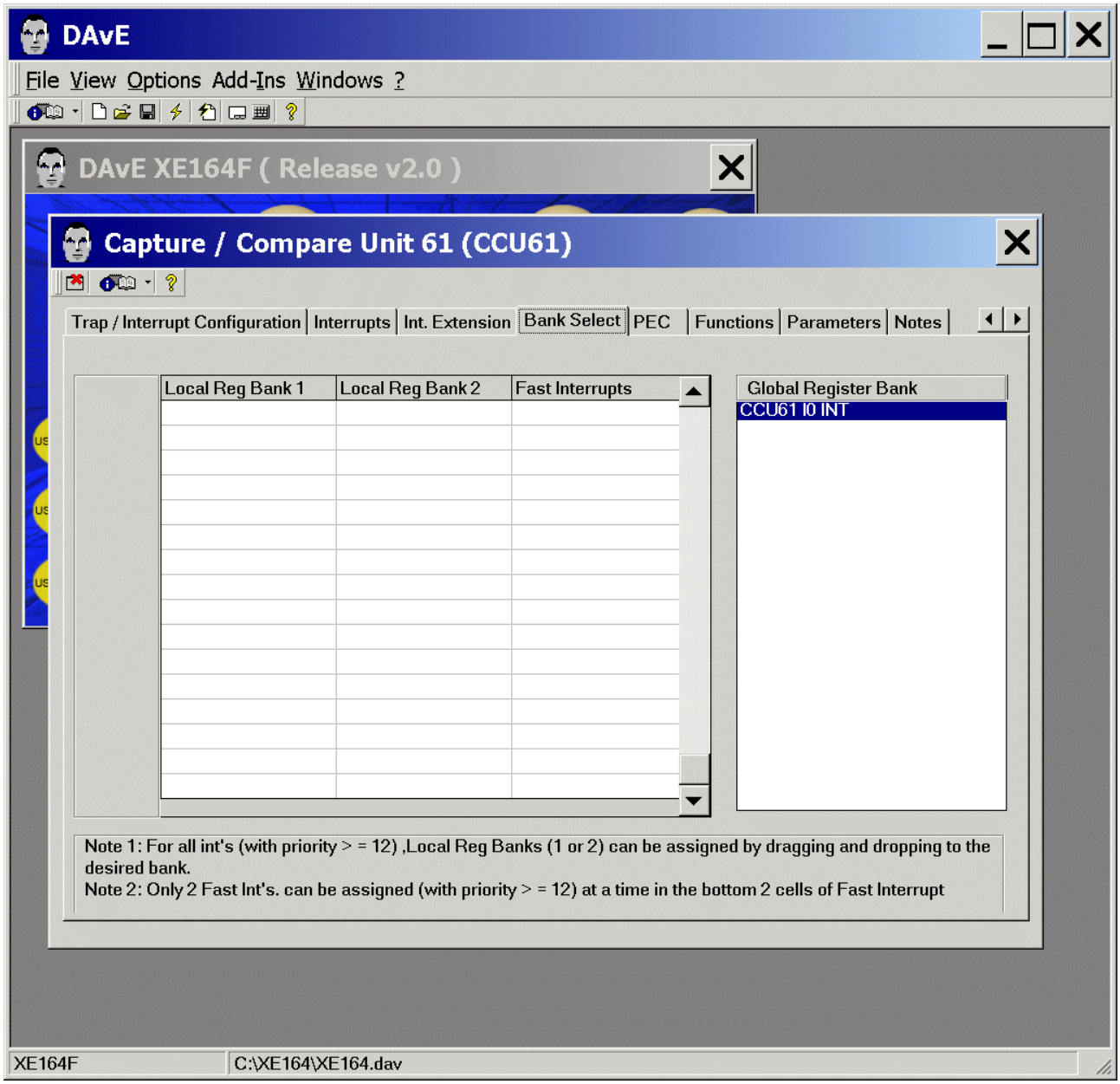
CCU61: Interrupts: drag and drop the CCU61 I0 INT to Interrupt Level 12, Group 0



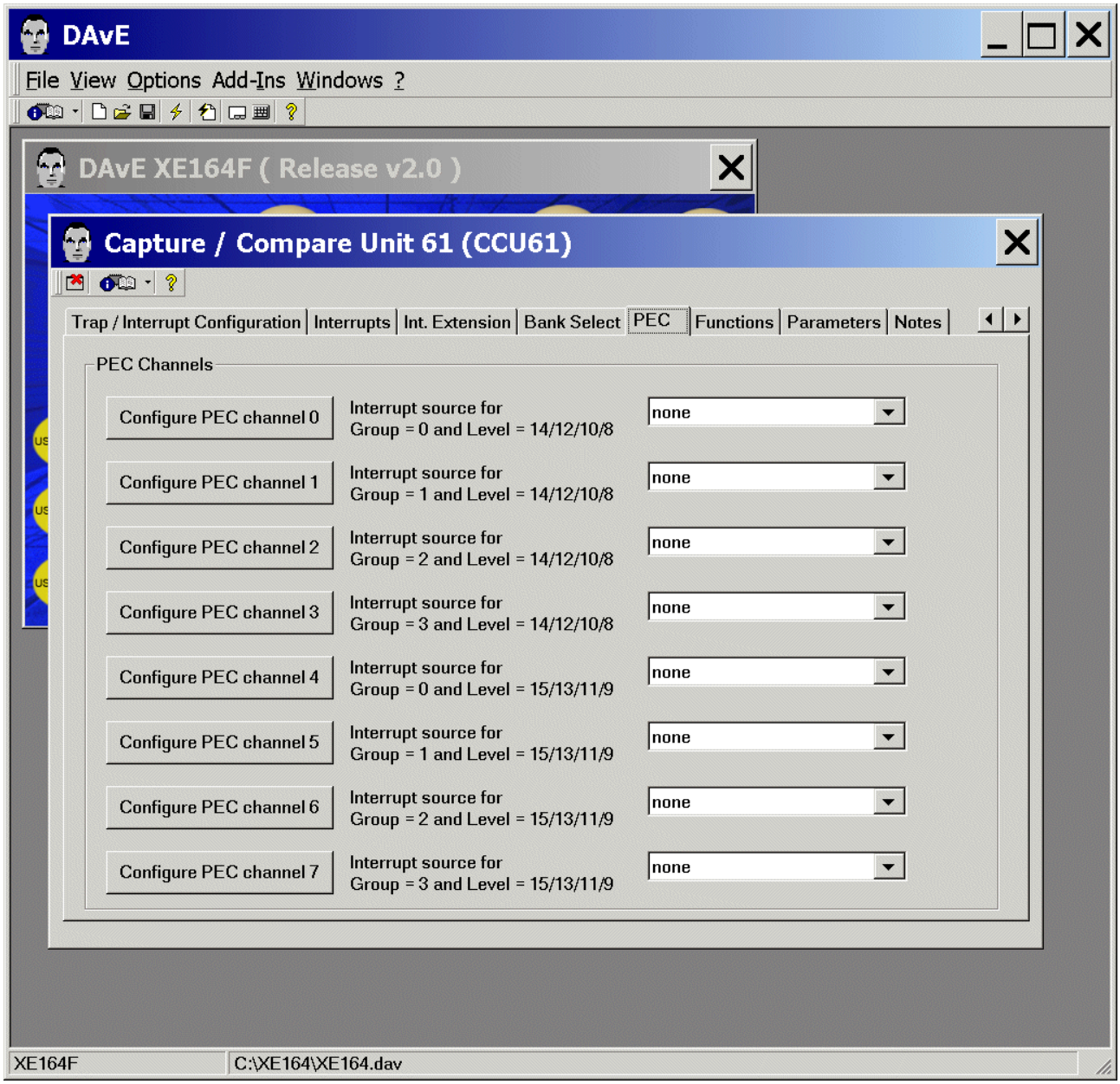
CCU61: Int. Extension: (do nothing)



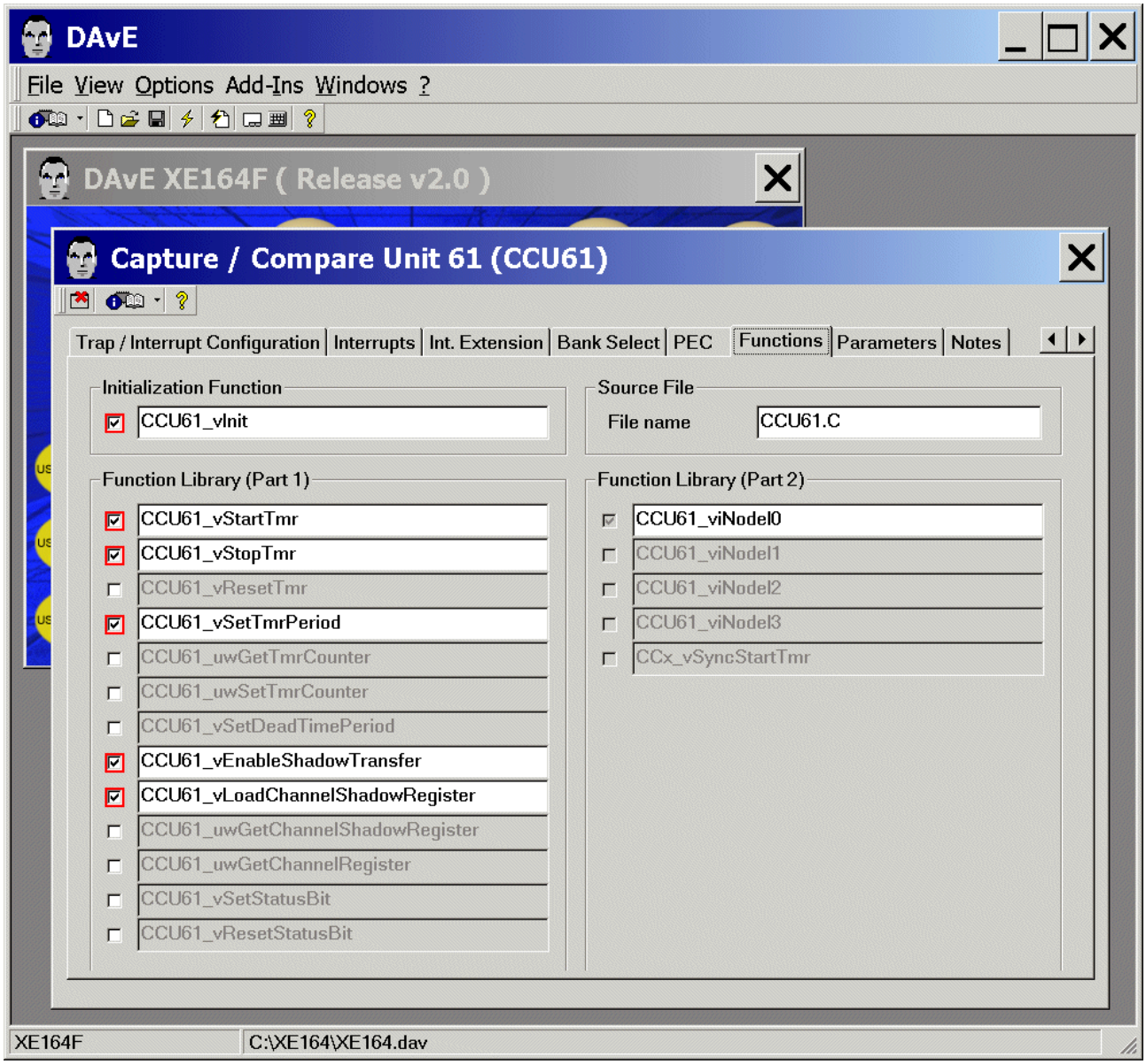
CCU61: Bank Select: (do nothing)



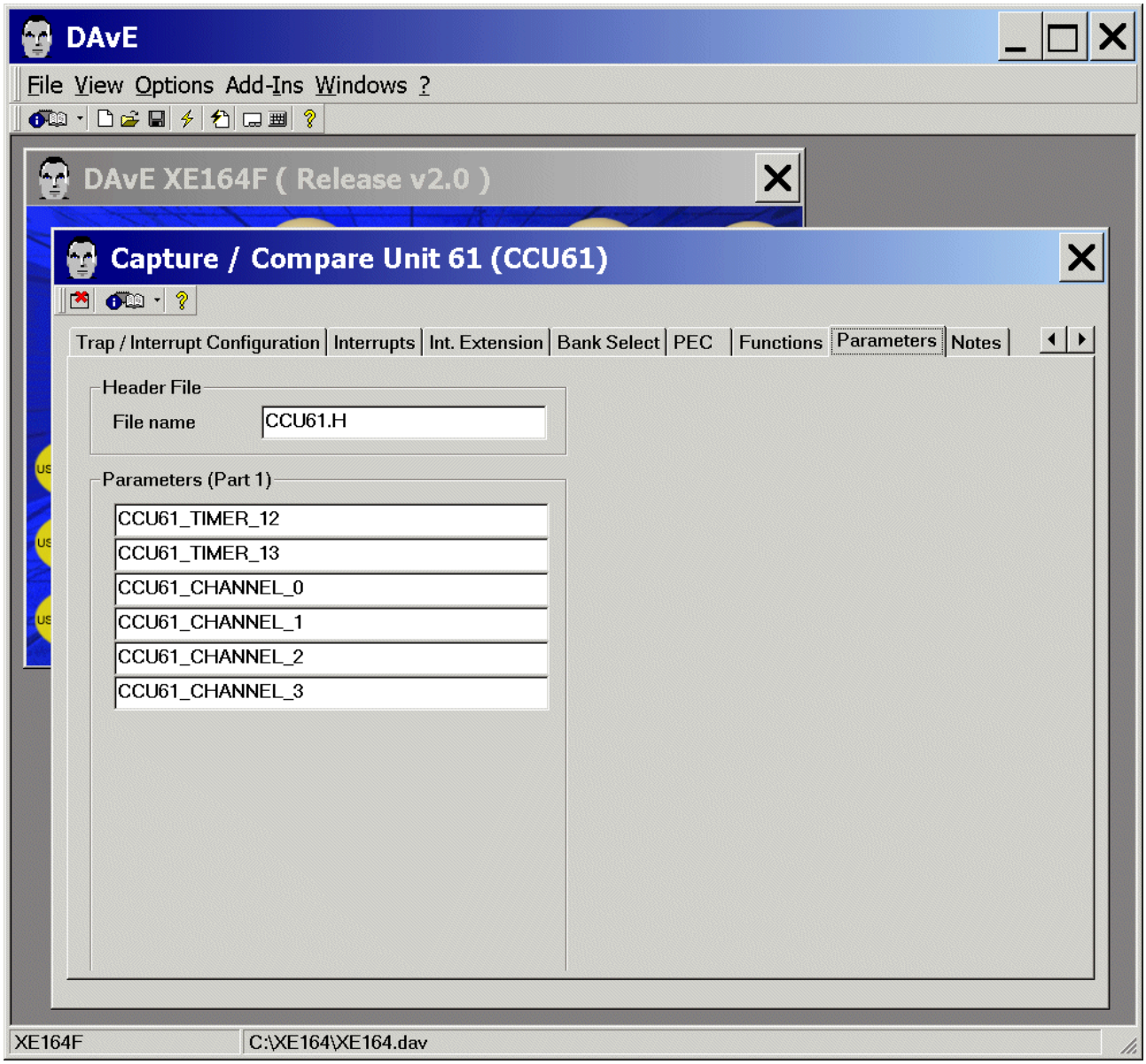
CCU61: PEC: (do nothing)



CCU61: Functions: Initialization Function: click/check ☑ CCU6_vInit
CCU61: Functions: Function Library (Part 1): click ☑ CCU61_vStartTmr
CCU61: Functions: Function Library (Part 1): click ☑ CCU61_vStopTmr
CCU61: Functions: Function Library (Part 1): click ☑ CCU61_vSetTmrPeriod
CCU61: Functions: Function Library (Part 1): click ☑ CCU61_vEnableShadowTransfer
CCU61: Functions: Function Library (Part 1): click ☑ CCU61_vLoadChannelShadowRegister



CCU61: Parameters: (do nothing)



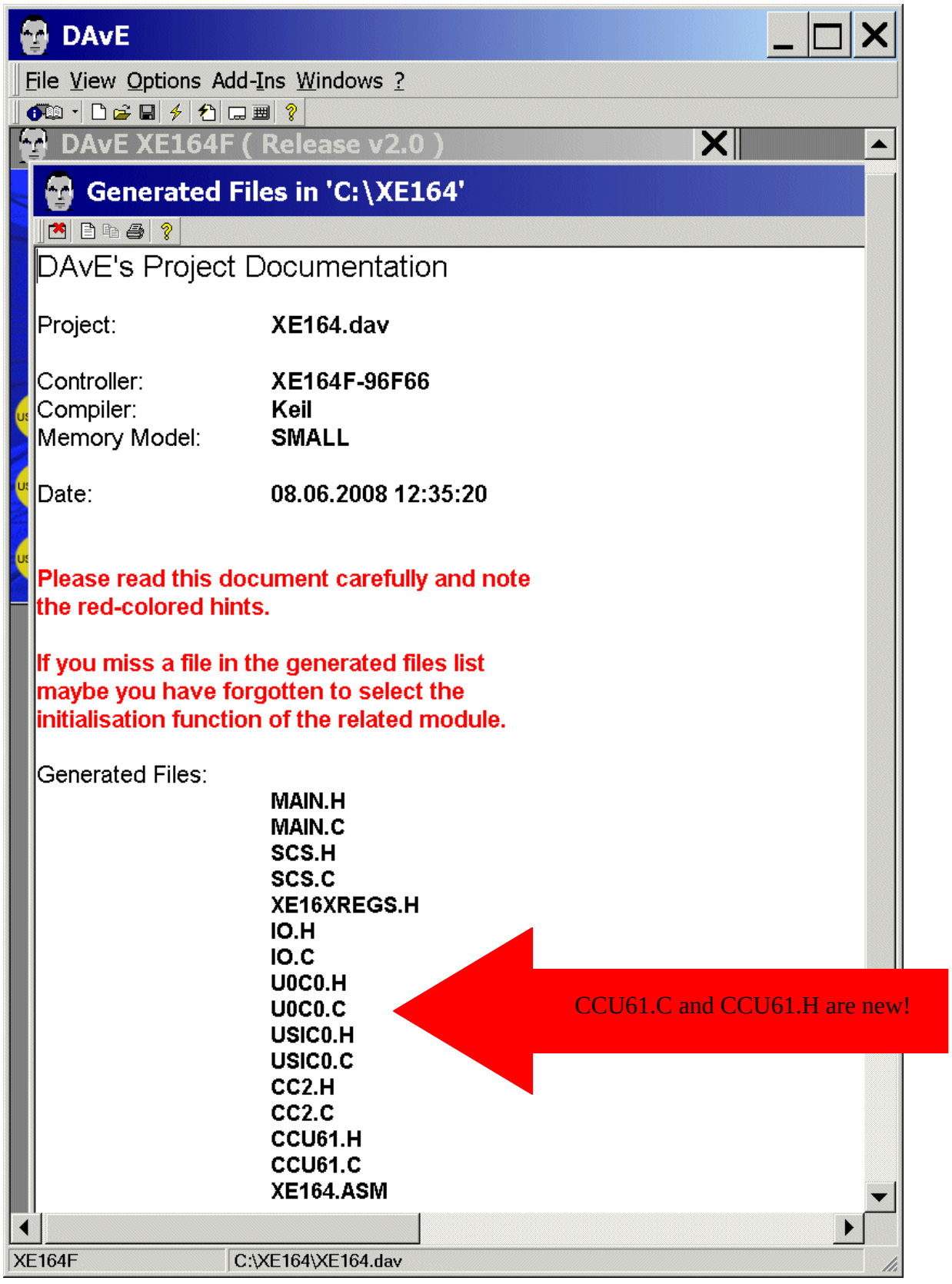
CCU61: Notes: If you wish, you can insert your comments here.

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

Generate Code:

File Generate Code	or click 

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



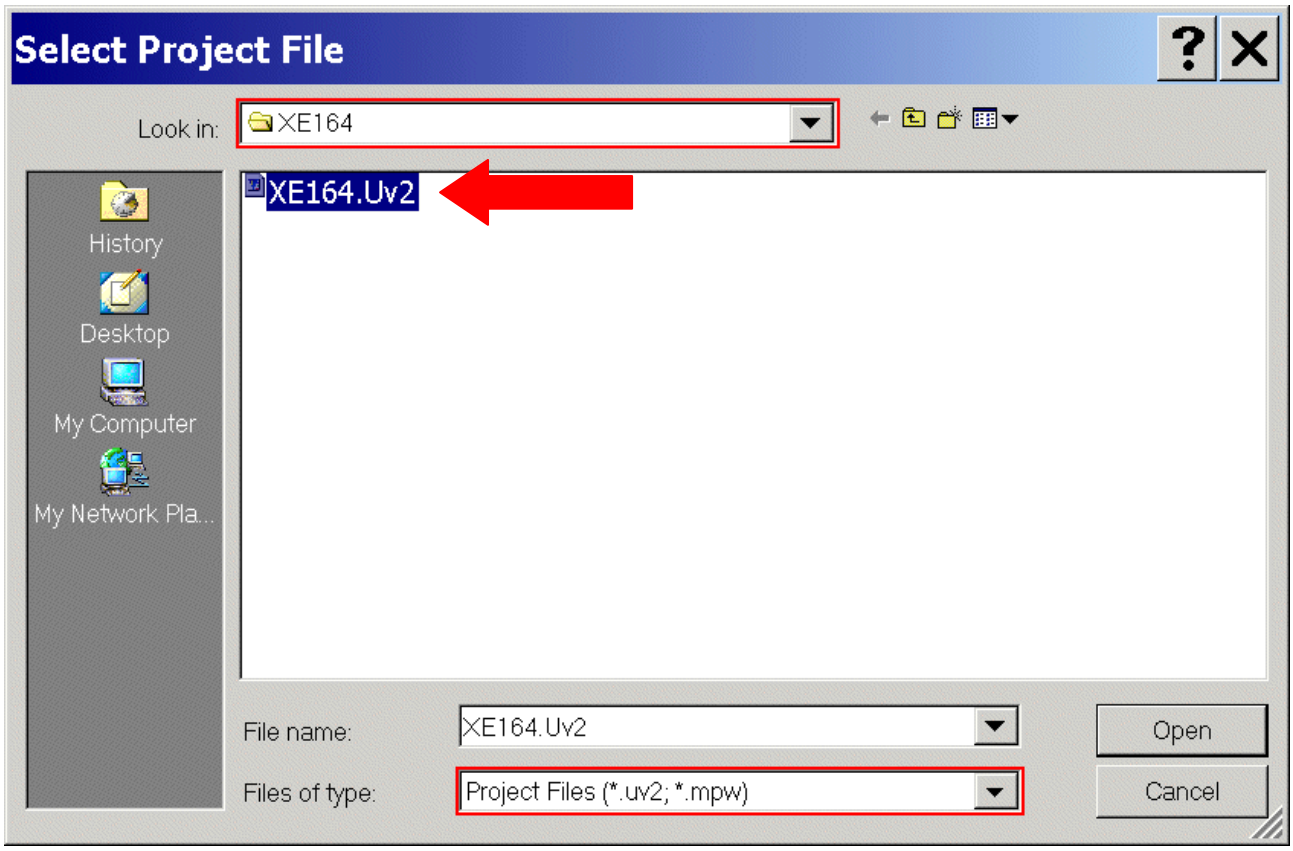
Close DAvE: File – Exit Save changes? click Yes

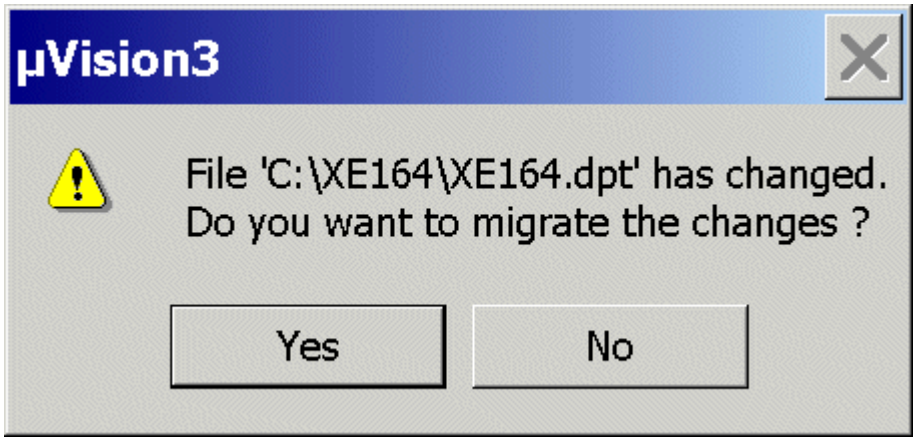
Start Keil µVision and open your Keil Project

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

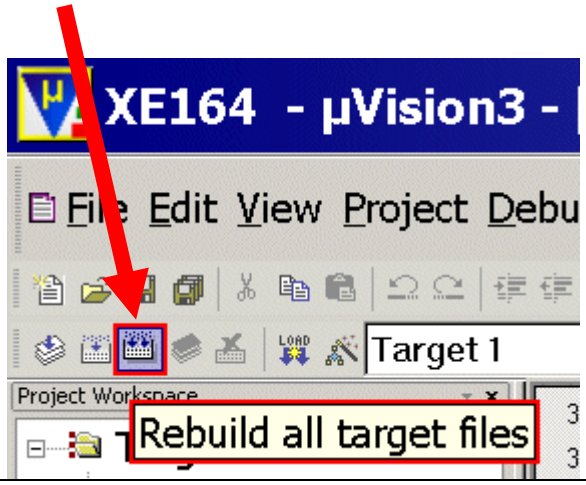
Project - Open Project
Select Project File: Look in: choose C:\XE164
Select Project File: Files of type: choose Project Files (*.uv2)
Click XE164.Uv2

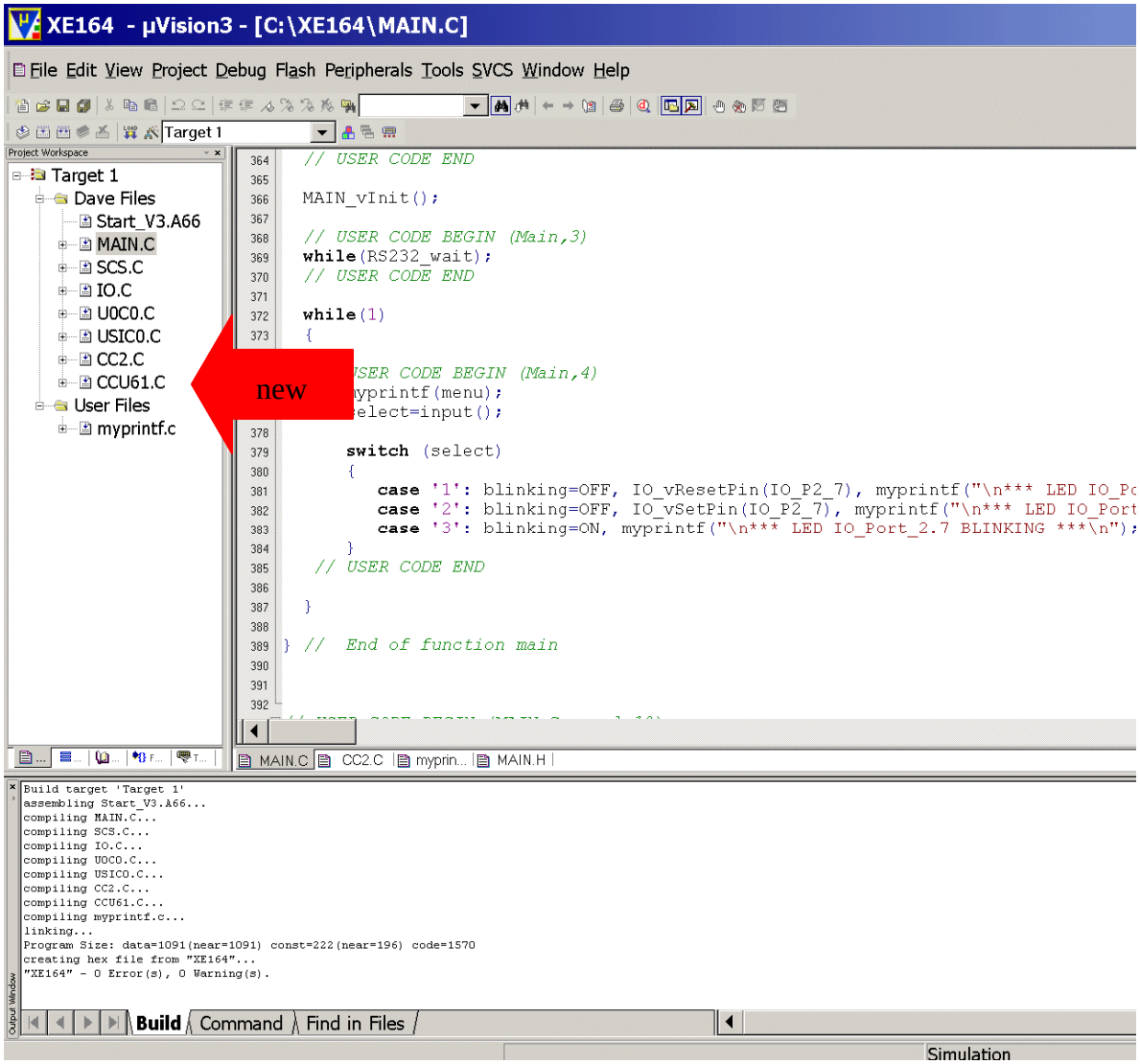
Open



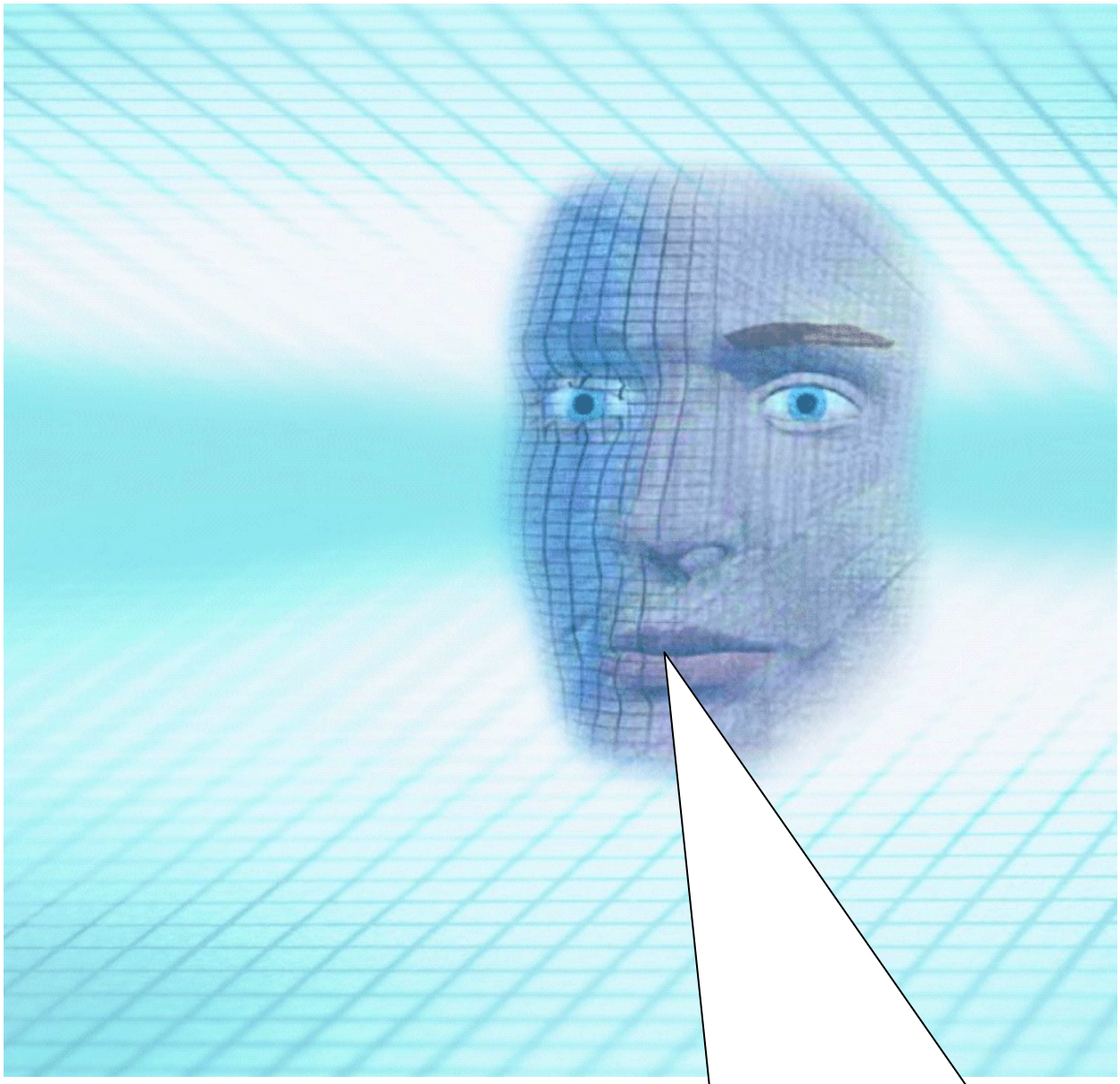

µVision3
File 'C:\XE164\XE164.dpt' has changed.
Do you want to migrate the changes ?
Yes
No

Click Yes

Project – Rebuild all target files	or click 



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!

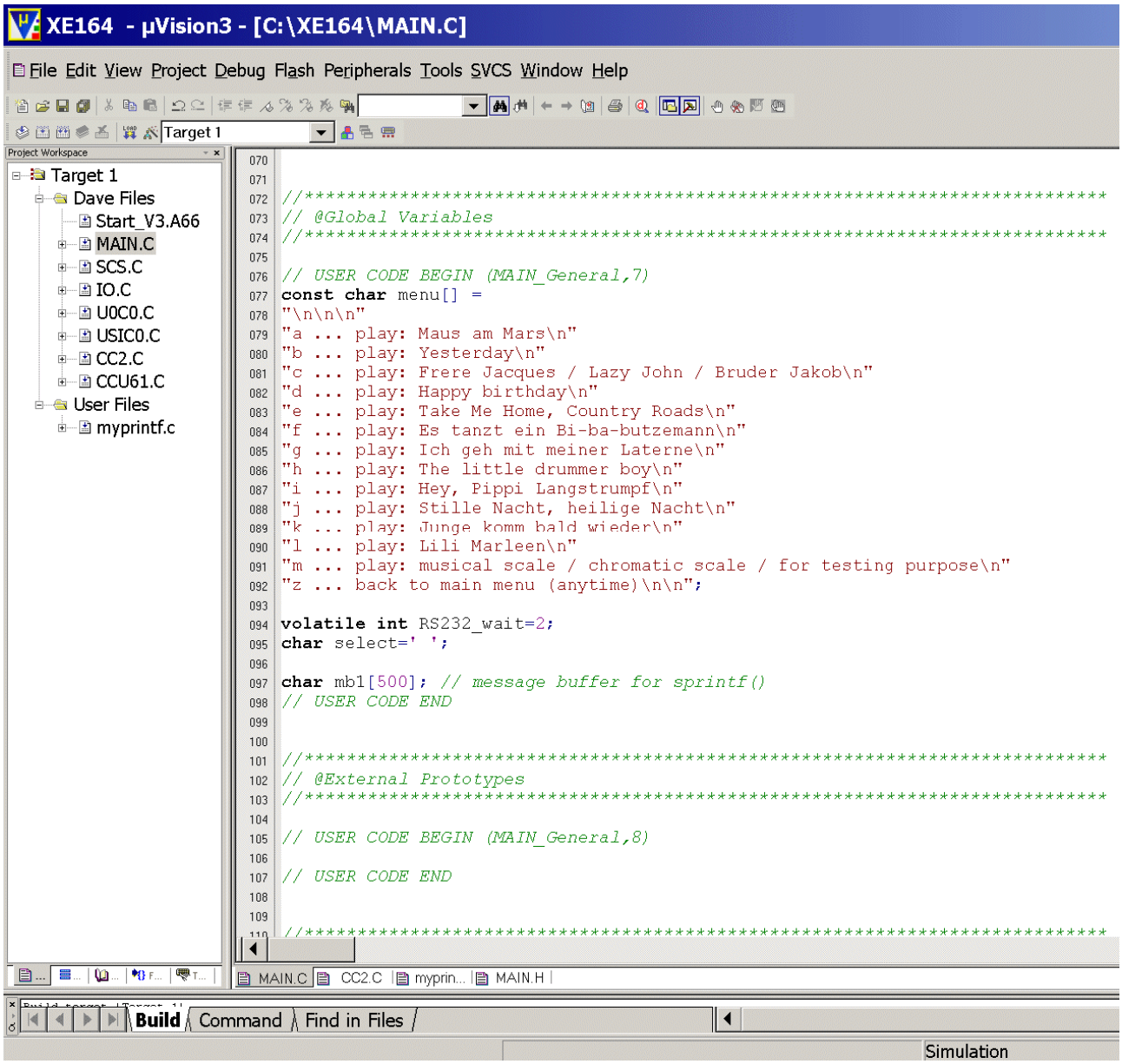
Double click MAIN.C and change Global Variables

from:

```
const char menu[] =
"\n\n\n"
"1 ... LED IO_Port_2.7 ON\n"
"2 ... LED IO_Port_2.7 OFF\n"
"3 ... LED IO_Port_2.7 blinking\n"
"\n";

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

to:

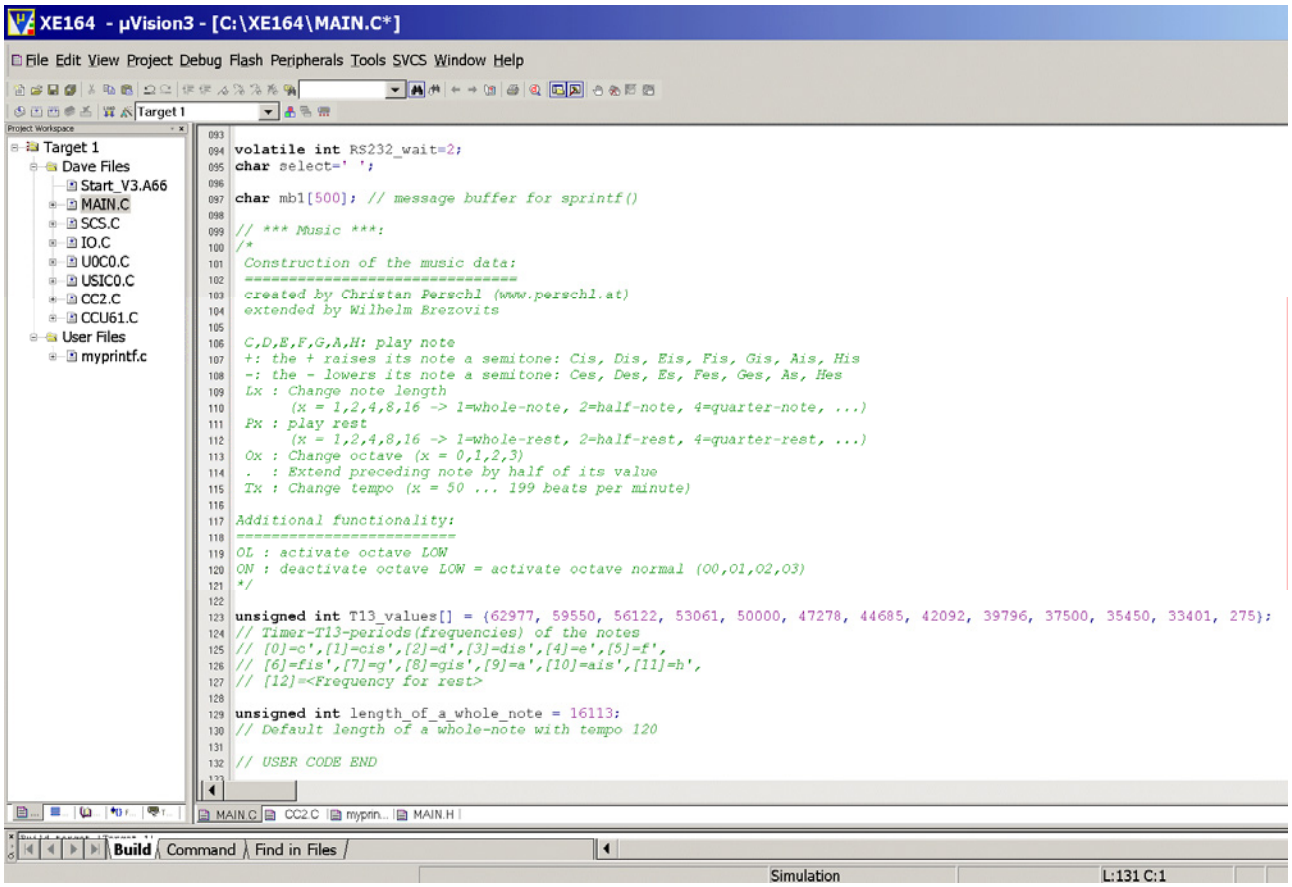
```
const char menu[] =
"\n\n\n"
"a ... play: Maus am Mars\n"
"b ... play: Yesterday\n"
"c ... play: Frere Jacques / Lazy John / Bruder Jakob\n"
"d ... play: Happy birthday\n"
"e ... play: Take Me Home, Country Roads\n"
"f ... play: Es tanzt ein Bi-ba-butzemann\n"
"g ... play: Ich geh mit meiner Laterne\n"
"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()
```




```
//*****************************************************************************
// @Global Variables
//*****************************************************************************

// USER CODE BEGIN (MAIN_General,7)
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()
// USER CODE END


//*****************************************************************************
// @External Prototypes
//*****************************************************************************

// USER CODE BEGIN (MAIN_General,8)

// USER CODE END


//*****************************************************************************
```

Double click MAIN.C and insert Global Variables:

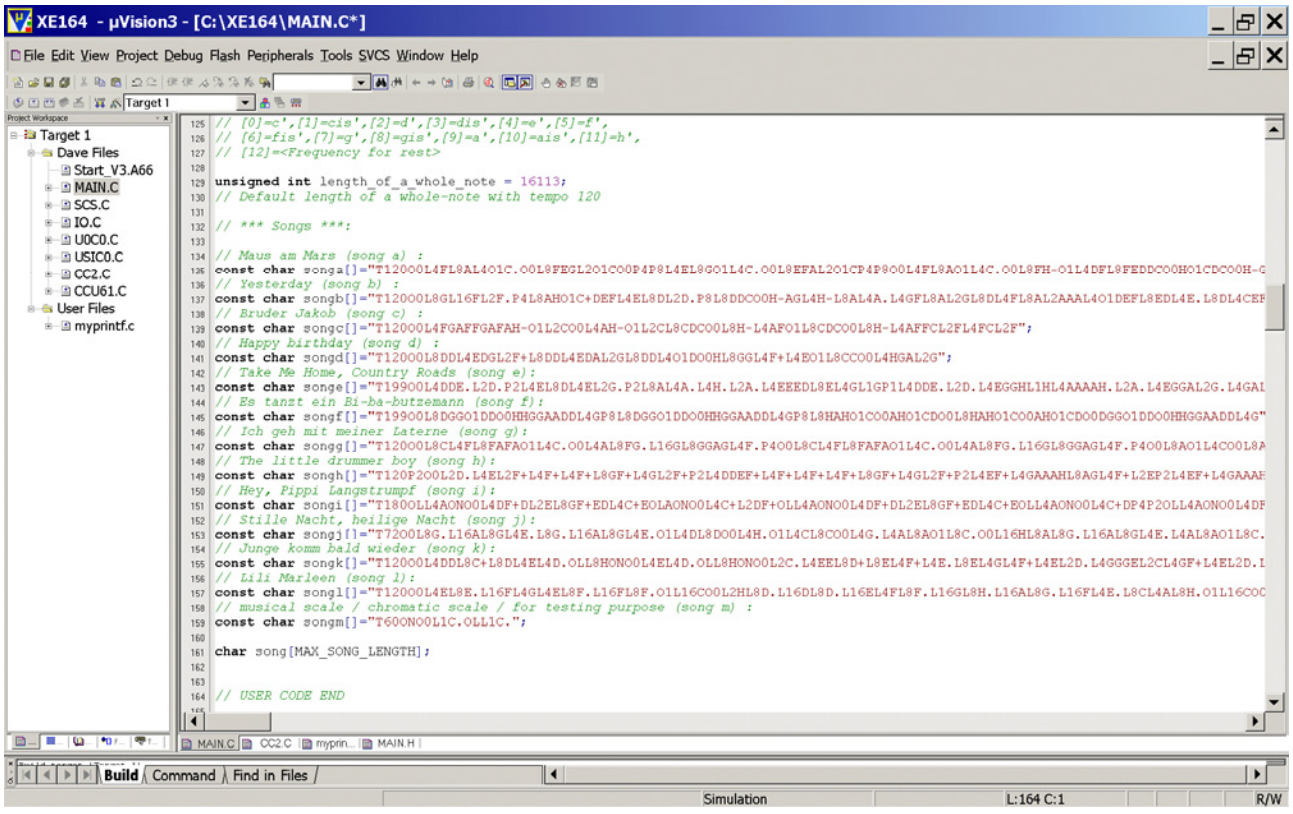
```
// *** Music ***:
/*
 Construction of the music data:
 ===============================
 created by Christan Perschl (www.perschl.at)
 extended by Wilhelm Brezovits

 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
```


XE164 - µVision3 - [C:\XE164\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
CCU61.C
User Files
myprintf.c
volatile int RS232_wait=2;
char select=' ';
char mb1[500]; // message buffer for sprintf()
// *** 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
// USER CODE END
MAIN.C
CC2.C
myprin...
MAIN.H
Build
Command
Find in Files
Simulation
L:131 C:1

Double click MAIN.C and insert Global Variables:

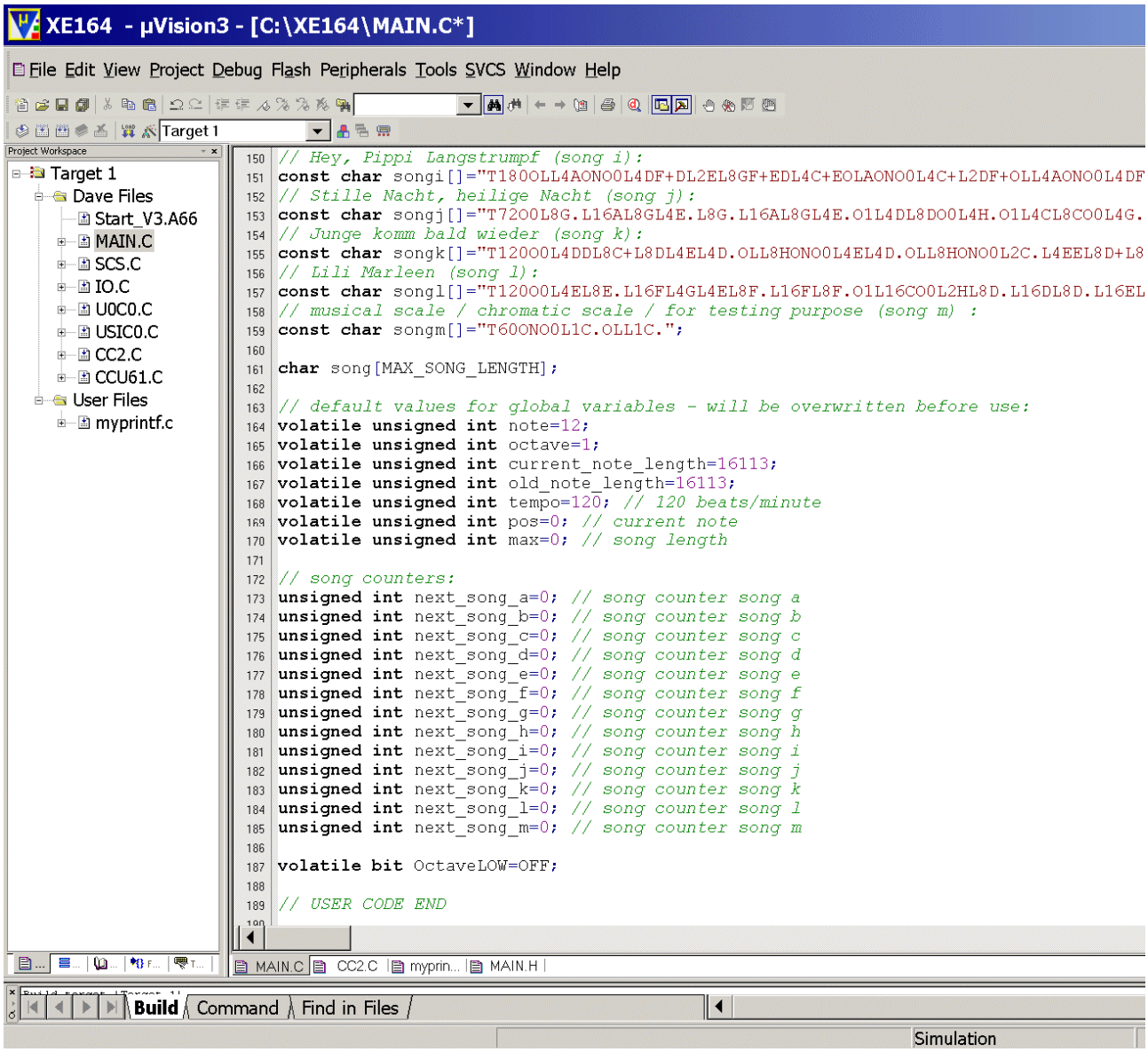
```
// *** Songs ***:

// 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+L4GAAAHL8AGL4F+L2EP2L4EF+L4GAAAHO1L8CO0L8HL4AL2GL8
HAL4GL2F+L8AGL4F+L2EP1L2D.L4EL4F+F+F+F+L8GF+L4GL2F+P1L8EDL4EL2D";
// Hey, Pippi Langstrumpf (song i):
const char
songi[]="T180OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLAONO0L4C+L2DF+OLL4AONO
0L4DF+DL2EL8GF+EDL4C+EOLL4AONO0L4C+DP4P2OLL4AONO0L4DF+DL2EL8GF+ED
L4C+EOLAONO0L4C+L2DF+OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLL4AONO0L4C+
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+L8DDDDDL8D
+L8DL4DP8L8DL8EEEEEEL2GP8L8EL1DP8L8DL8EEEL4E.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];
```



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;
```


XE164 - µVision3 - [C:\XE164\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
CCU61.C
User Files
myprintf.c
150 // Hey, Pippi Langstrumpf (song i):
151 const char songi[]="T180OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLAONO0L4C+L2DF+OLL4AONO0L4DF
152 // Stille Nacht, heilige Nacht (song j):
153 const char songj[]="T72O0L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8DO0L4H.O1L4CL8CO0L4G.
154 // Junge komm bald wieder (song k):
155 const char songk[]="T120O0L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4EEL8D+L8
156 // Lili Marleen (song l):
157 const char songl[]="T12O0O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL
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 // 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
MAIN.C
CC2.C
myprin...
MAIN.H
Build
Command
Find in Files
Simulation

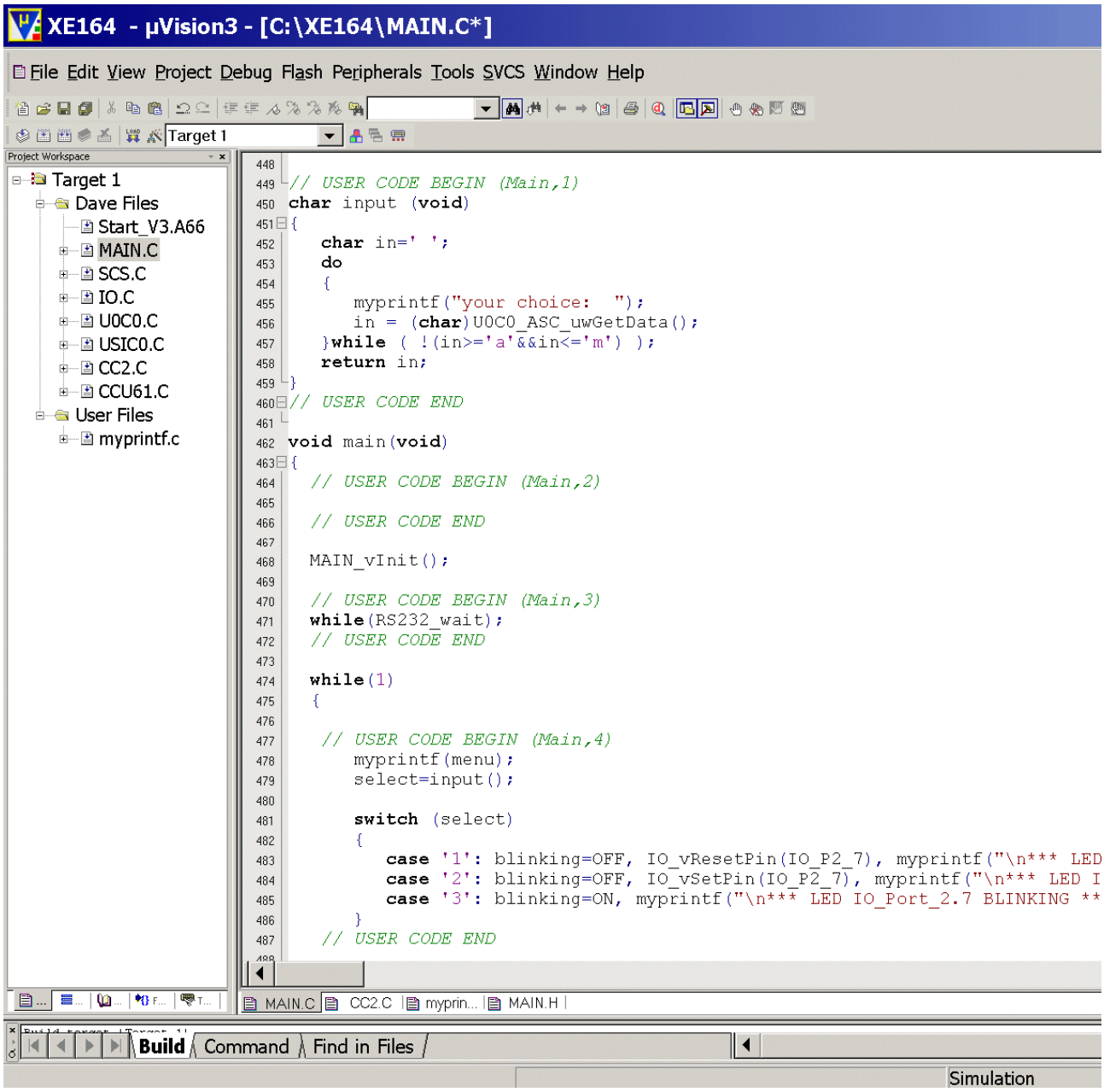
Double click MAIN.C and change function ”char input (void)”:

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:

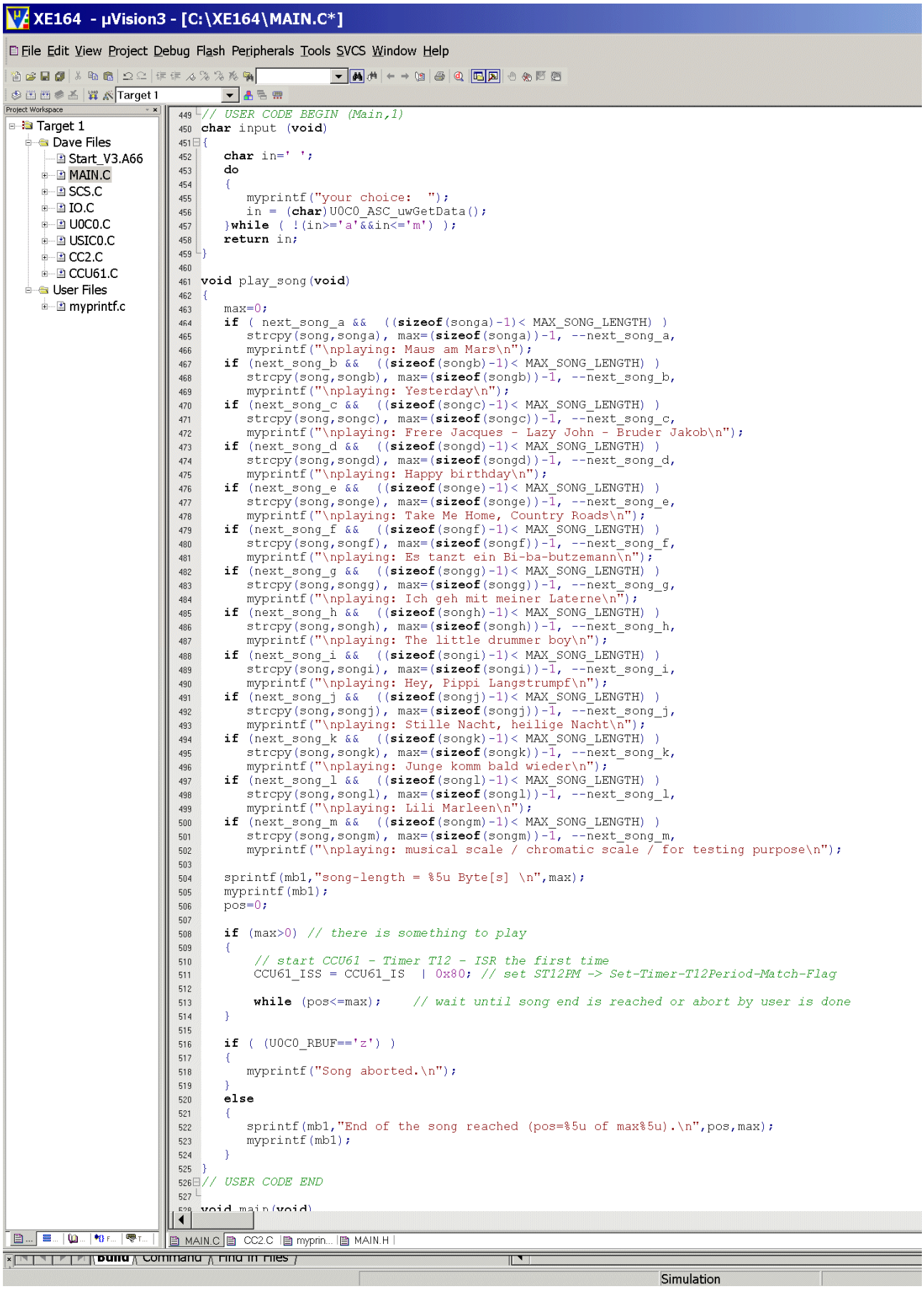
```
char input (void)
{
  char in=' ';
  do
  {
    myprintf("your choice:  ");
    in = (char)U0C0_ASC_uwGetData();
  }while ( !(in>='a'&&in<='m') );
  return in;
}
```


XE164 - µVision3 - [C:\XE164\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
CCU61.C
User Files
myprintf.c
// USER CODE BEGIN (Main,1)
char input (void)
{
char in=' ';
do
{
myprintf("your choice: ");
in = (char)U0C0_ASC_uwGetData();
}while (!(in>='a'&&in<='m'));
return in;
}
// USER CODE END
void main(void)
{
// USER CODE BEGIN (Main,2)
// USER CODE END
MAIN_vInit();
// USER CODE BEGIN (Main,3)
while(RS232_wait);
// USER CODE END
while(1)
{
// USER CODE BEGIN (Main,4)
myprintf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF, IO_vResetPin(IO_P2_7), myprintf("\n*** LED
case '2': blinking=OFF, IO_vSetPin(IO_P2_7), myprintf("\n*** LED I
case '3': blinking=ON, myprintf("\n*** LED IO_Port_2.7 BLINKING **
}
// USER CODE END
MAIN.C
CC2.C
myprin...
MAIN.H
Build
Command
Find in Files
Simulation

Double click MAIN.C and insert the function play_song():

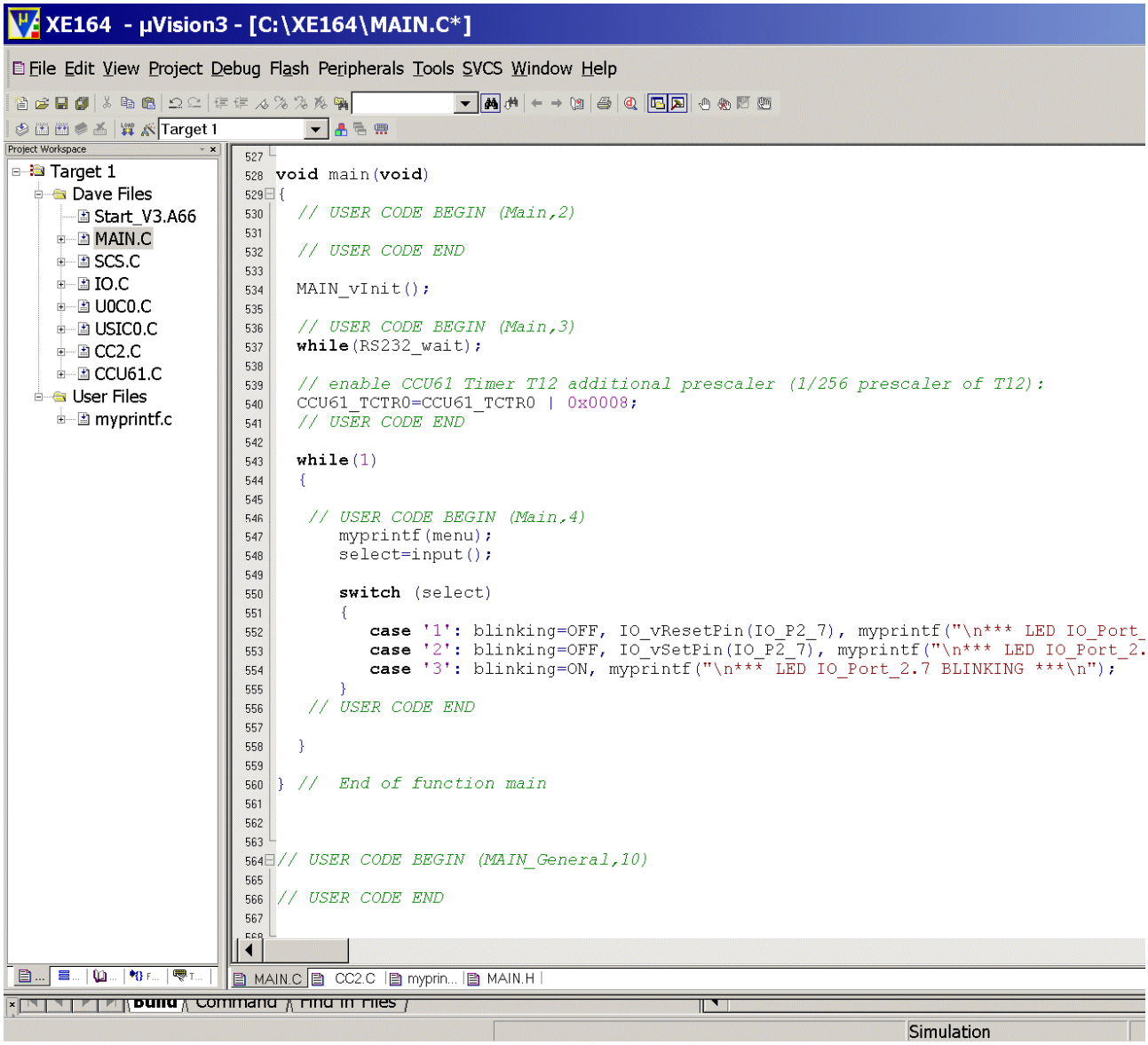
```
void play_song(void)
{
  max=0;
  if ( next_song_a &&  ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
    myprintf("\nplaying: Maus am Mars\n");
  if (next_song_b &&  ((sizeof(songb)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songb), max=(sizeof(songb))-1, --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 CCU61 - Timer T12 - ISR the first time
     CCU61_ISS = CCU61_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);
  }
}
```


XE164 - µVision3 - [C:\XE164\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
CCU61.C
User Files
myprintf.c
// USER CODE BEGIN (Main,1)
char input (void)
{
char in=' ';
do
{
myprintf("your choice: ");
in = (char)U0C0_ASC_uwGetData();
}while (!(in>='a'&&in<='m'));
return in;
}
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 CCU61 - Timer T12 - ISR the first time
CCU61_ISS = CCU61_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);
}
}
// USER CODE END
void main(void)
MAIN.C
CC2.C
myprin...
MAIN.H
Build
Command
Find in Files
Simulation

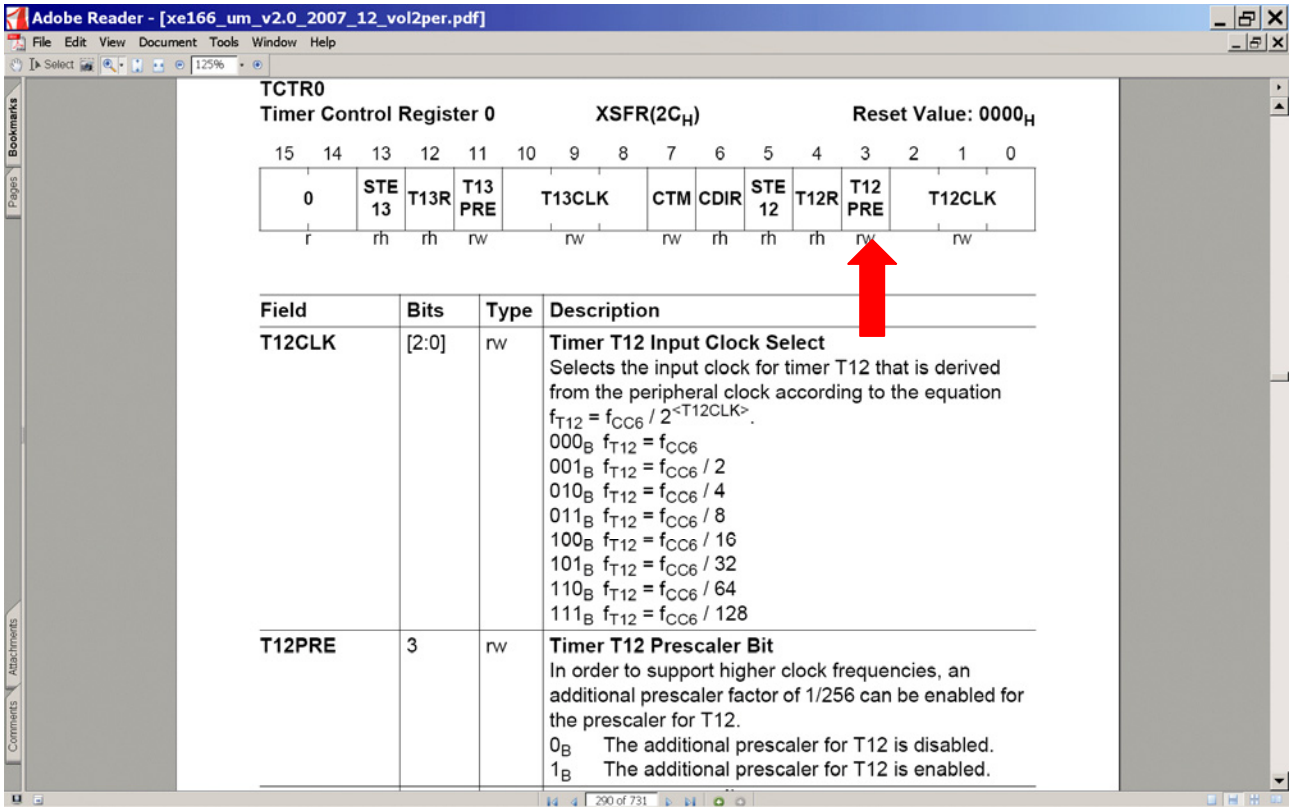
Double click MAIN.C and insert the following code into the main function:

```
// enable CCU61 Timer T12 additional prescaler (1/256 prescaler of T12):
CCU61_TCTR0=CCU61_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_{CCU61} to a maximum of $f_{CCU61}/128$ 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

TCTR0
Timer Control Register 0 **XSFR($2C_H$)** **Reset Value: 0000_H**

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_B $f_{T12} = f_{CC6}$ 001_B $f_{T12} = f_{CC6} / 2$ 010_B $f_{T12} = f_{CC6} / 4$ 011_B $f_{T12} = f_{CC6} / 8$ 100_B $f_{T12} = f_{CC6} / 16$ 101_B $f_{T12} = f_{CC6} / 32$ 110_B $f_{T12} = f_{CC6} / 64$ 111_B $f_{T12} = f_{CC6} / 128$
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_B The additional prescaler for T12 is disabled. 1_B 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

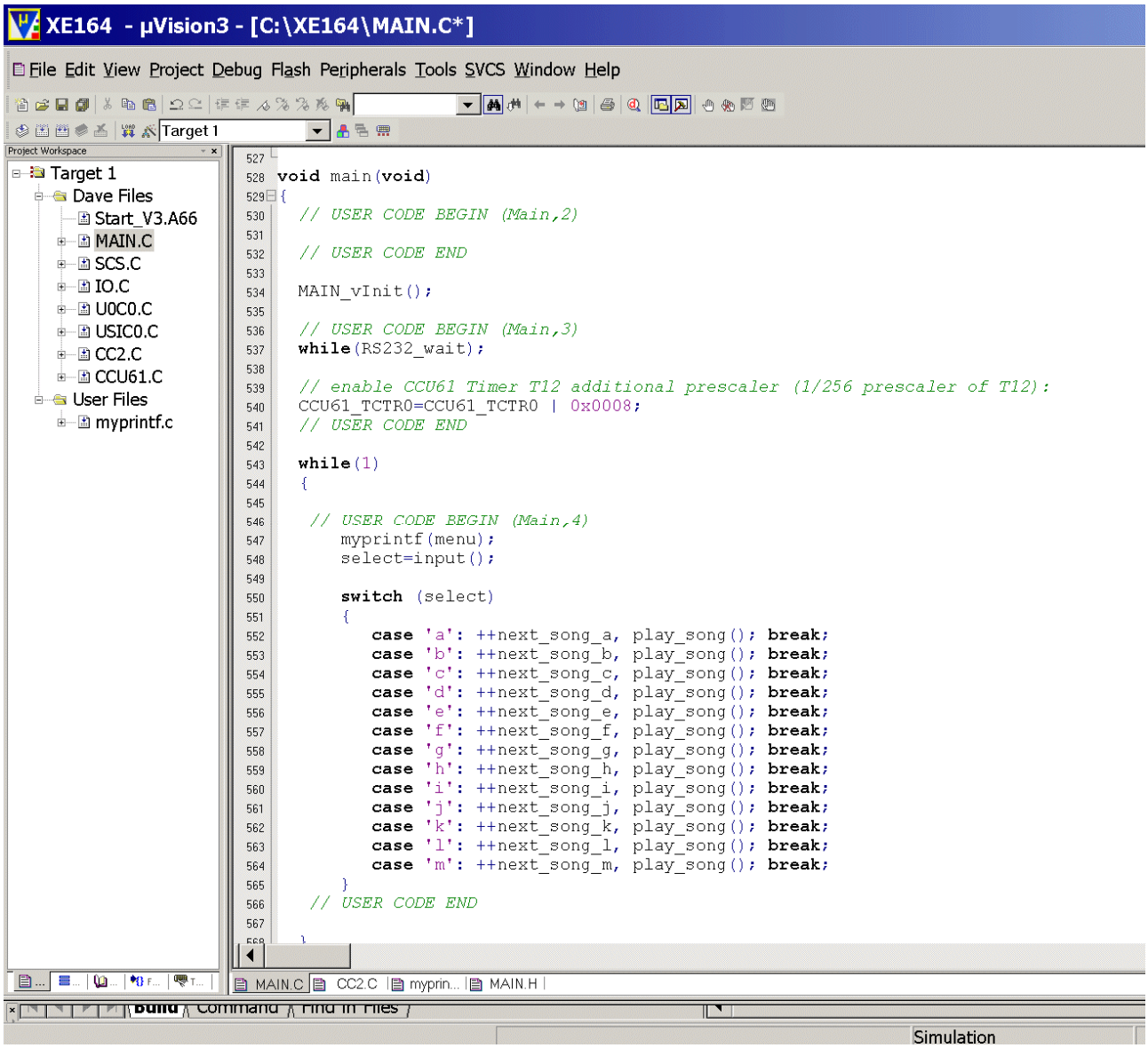
Double click MAIN.C and change the following code (main function, while(1) loop)

from:

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

    switch (select)
    {
      case '1': blinking=OFF, IO_vResetPin(IO_P2_7), myprintf("\n*** LED IO_Port_2.7 ON
***\n"); break;
      case '2': blinking=OFF, IO_vSetPin(IO_P2_7), myprintf("\n*** LED IO_Port_2.7 OFF
***\n");  break;
      case '3': blinking=ON, myprintf("\n*** LED IO_Port_2.7 BLINKING ***\n");
break;
    }
```

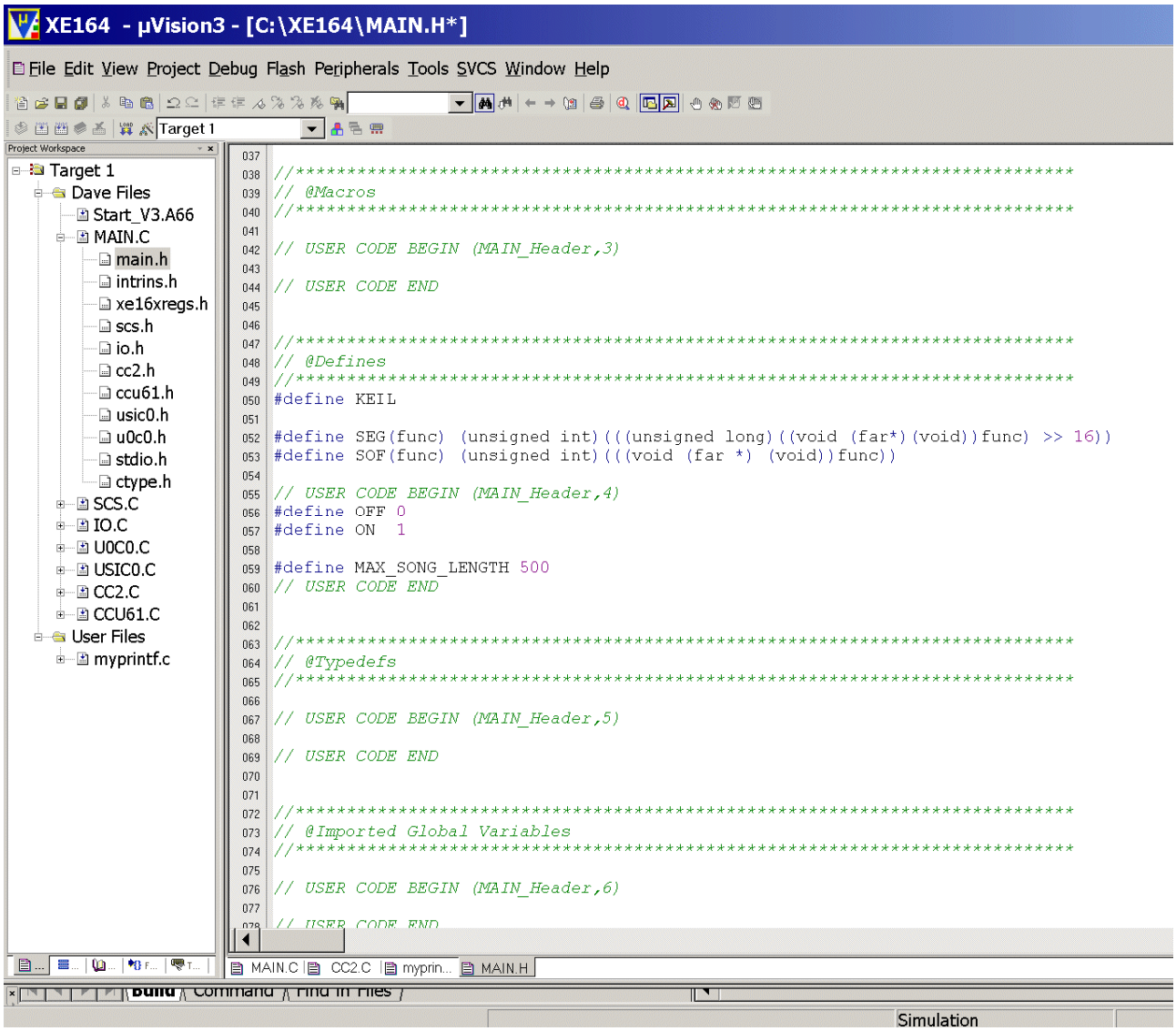
to:

```
    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;
    }
```


XE164 - µVision3 - [C:\XE164\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
CCU61.C
User Files
myprintf.c
void main(void)
{
// USER CODE BEGIN (Main,2)
// USER CODE END
MAIN_vInit();
// USER CODE BEGIN (Main,3)
while(RS232_wait);
// enable CCU61 Timer T12 additional prescaler (1/256 prescaler of T12):
CCU61_TCTR0=CCU61_TCTR0 | 0x0008;
// USER CODE END
while(1)
{
// USER CODE BEGIN (Main,4)
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;
}
// USER CODE END
MAIN.C
CC2.C
myprin...
MAIN.H
Build
Command
Find in Files
Simulation

Double click Main.h and insert the following Defines:

```
#define MAX_SONG_LENGTH 500
```



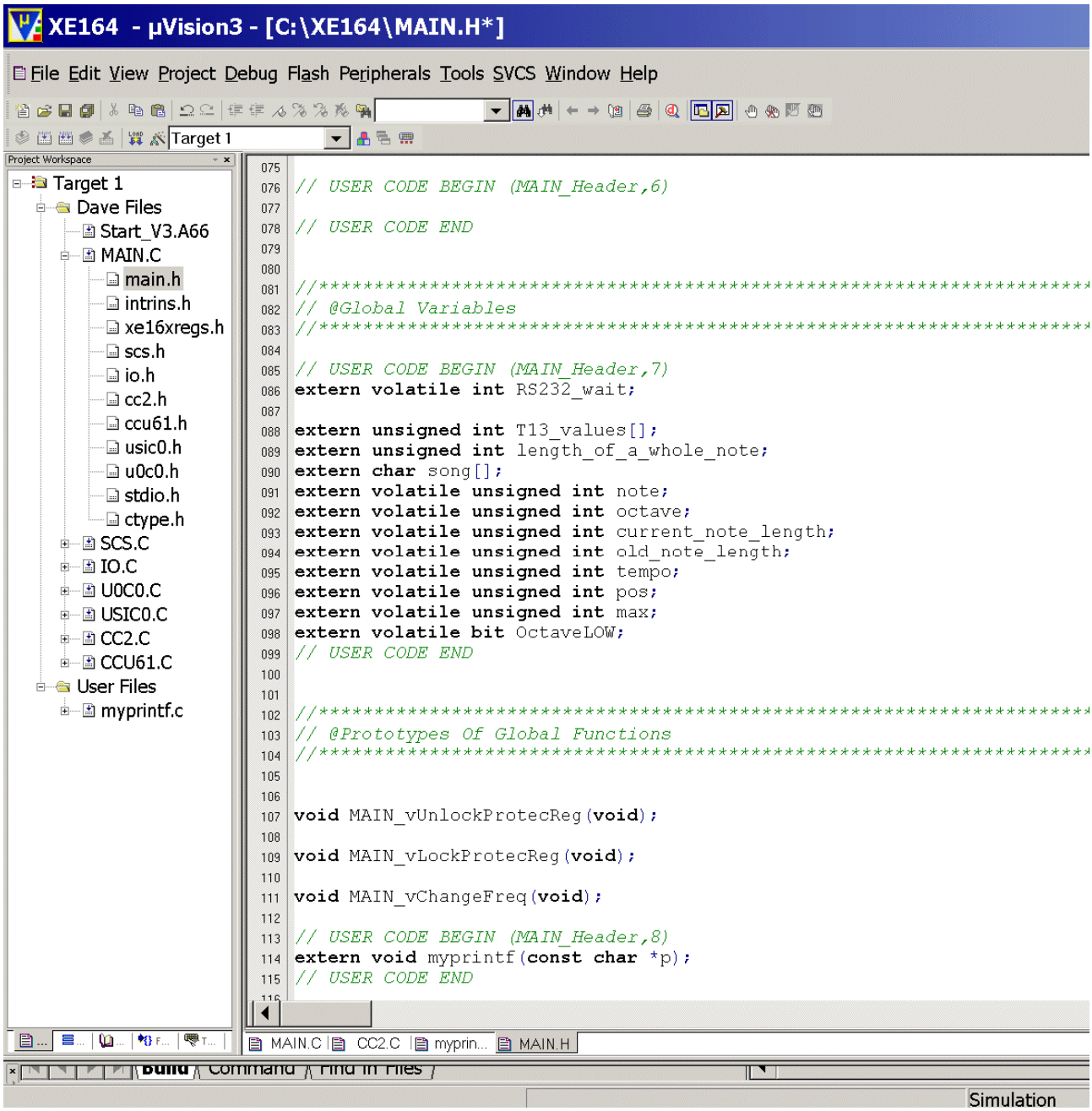
Double click Main.h and change extern declarations “Global Variables”

from:

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

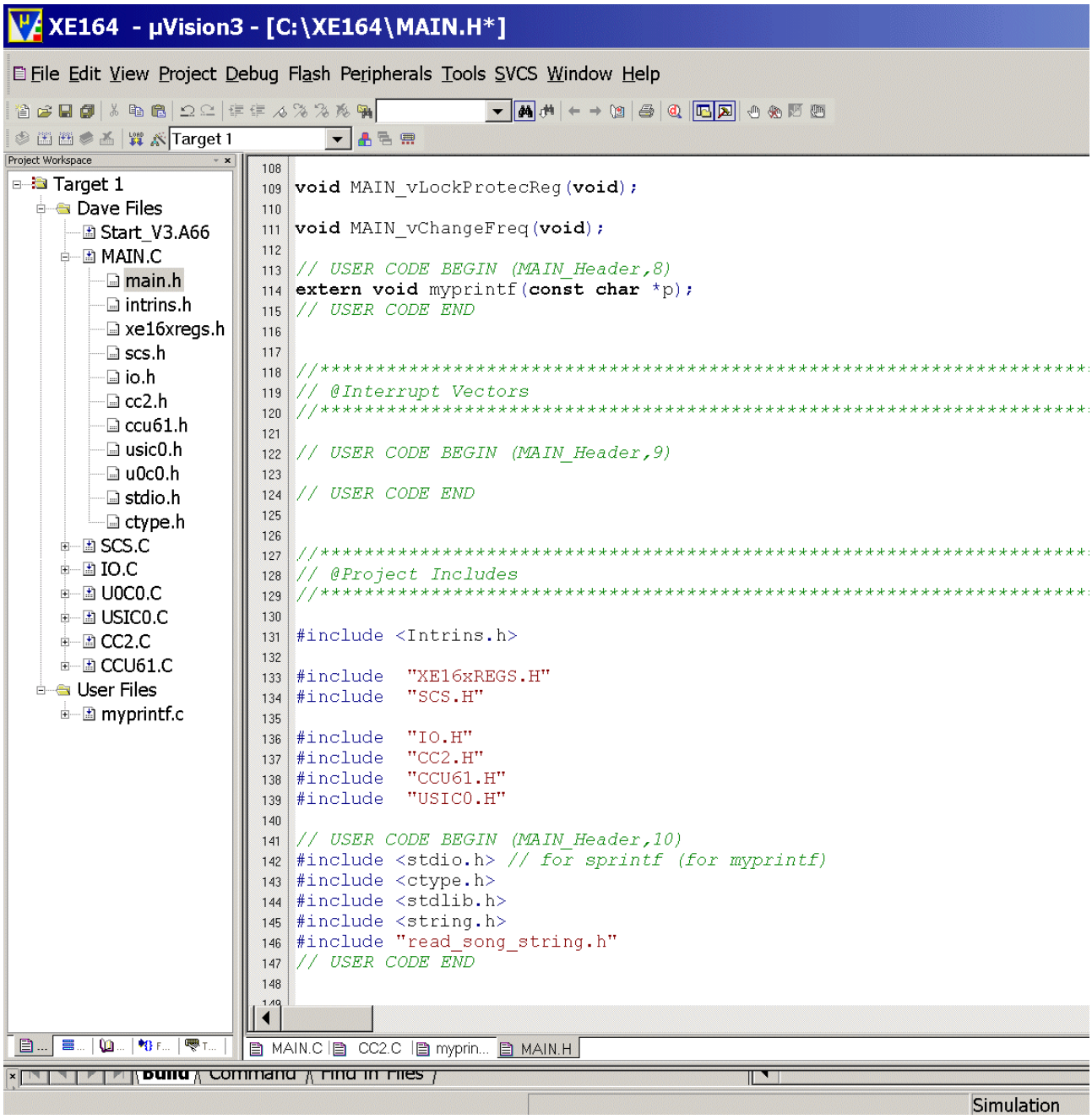
to:

```
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;
```


XE164 - µVision3 - [C:\XE164\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
ccu61.h
usic0.h
u0c0.h
stdio.h
ctype.h
SCS.C
IO.C
U0C0.C
USIC0.C
CC2.C
CCU61.C
User Files
myprintf.c
// USER CODE BEGIN (MAIN_Header,6)
// USER CODE END
//*****
// @Global Variables
//*****
// USER CODE BEGIN (MAIN_Header,7)
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;
// USER CODE END
//*****
// @Prototypes Of Global Functions
//*****
void MAIN_vUnlockProtecReg(void);
void MAIN_vLockProtecReg(void);
void MAIN_vChangeFreq(void);
// USER CODE BEGIN (MAIN_Header,8)
extern void myprintf(const char *p);
// USER CODE END
MAIN.C
CC2.C
myprin...
MAIN.H
Build
Command
Find in Files
Simulation

Double click Main.h and insert include files:

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



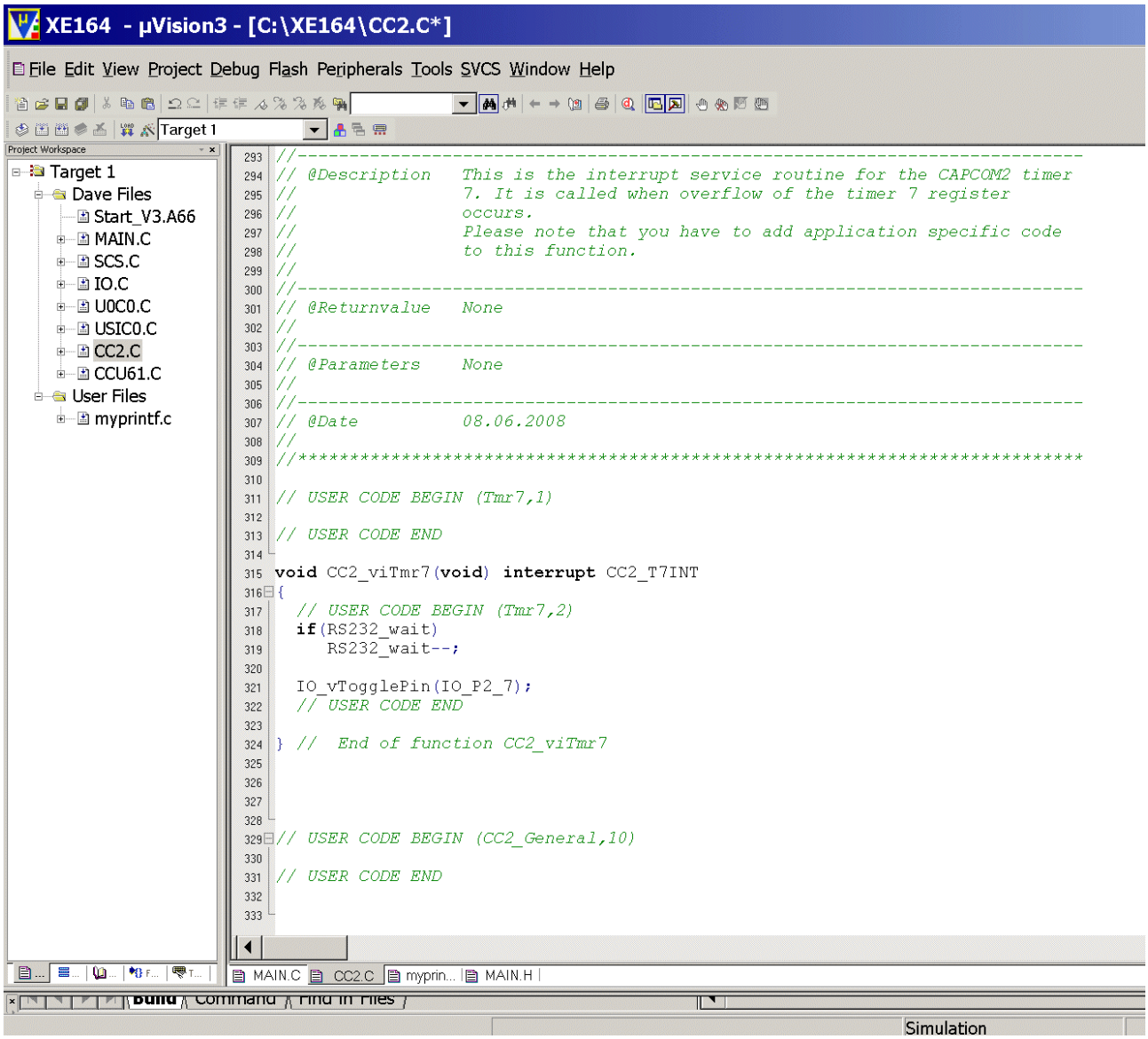
Double click CC2.C change code (CAPCOM 2 Timer 7 Interrupt Service Routine)

from:

```
if(RS232_wait)
  RS232_wait--;

if (blinking)
{
  IO_vTogglePin(IO_P2_7);
}
```

to:

```
if(RS232_wait)
  RS232_wait--;

IO_vTogglePin(IO_P2_7);
```



Remember:

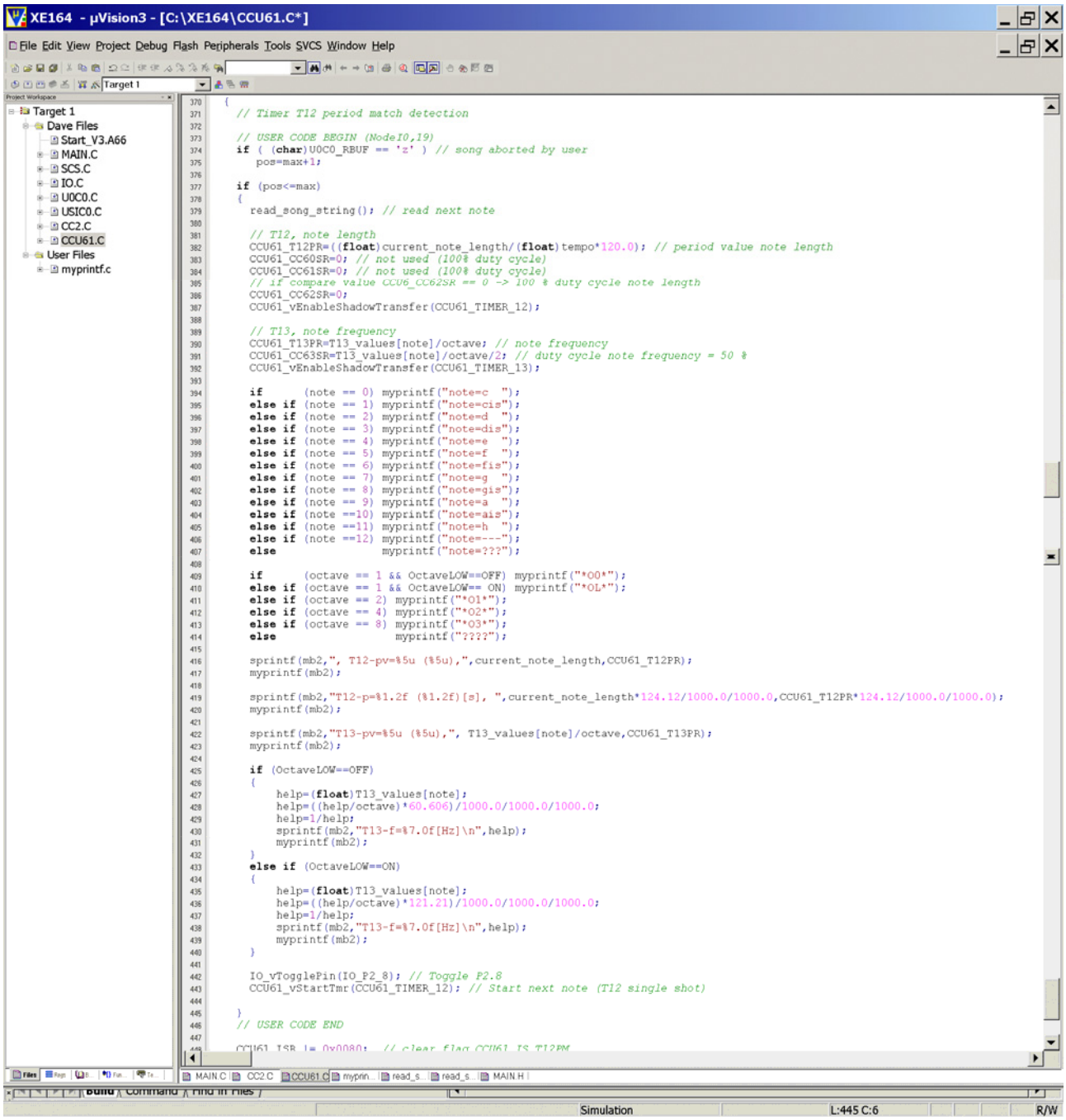
Port_2 pins used as GPIO:

Port Lines	Function	Comment
P2.8	Show start of next note	Toggled via Software
P2.7	„use: program running signal“	Toggled via CAPCOM2_Timer_7 ISR

Double click CCU61.C insert code (CCU61 Interrupt Service Routine, Timer T12 period match) :

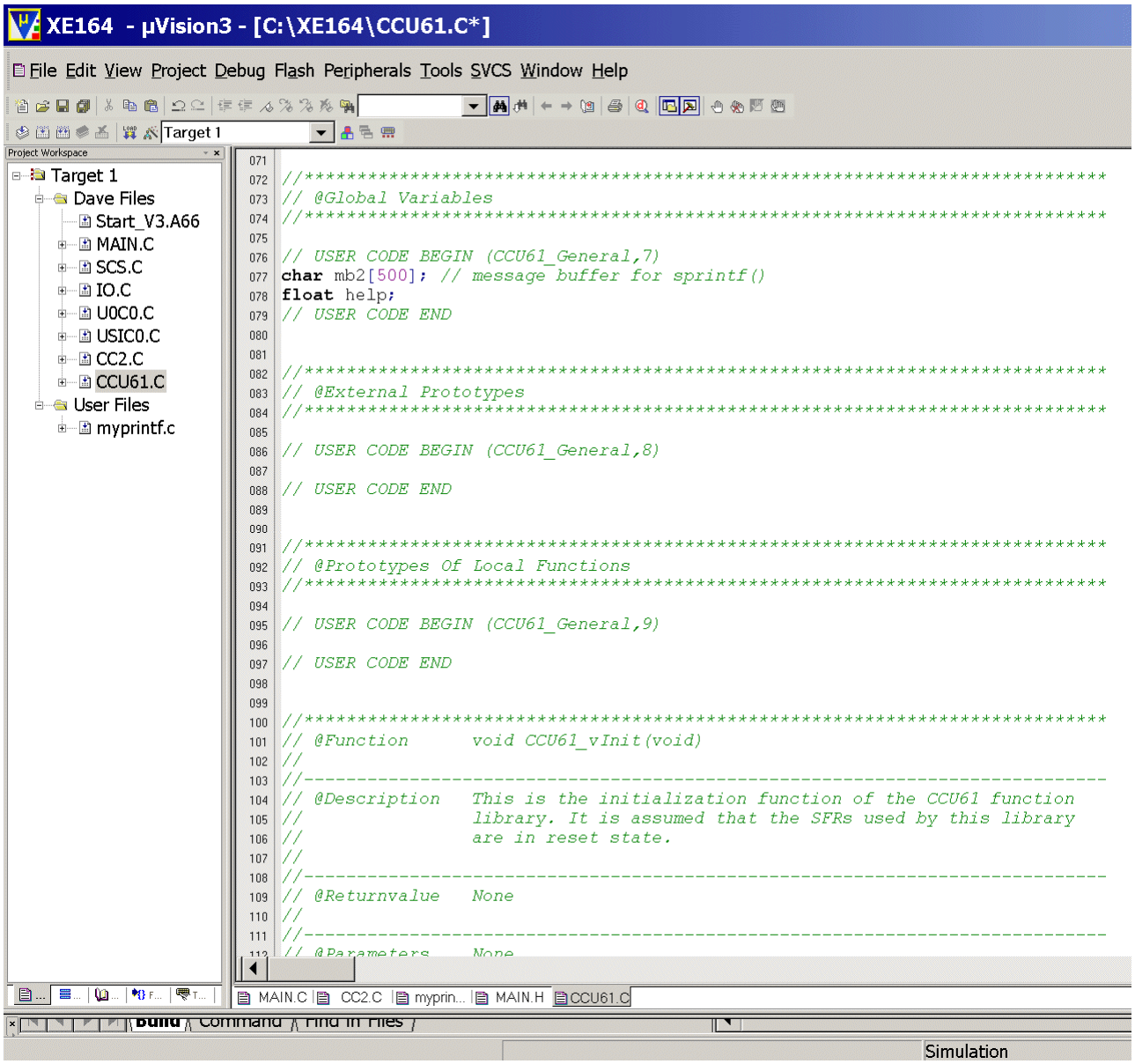
```
if ( (char)U0C0_RBUF == 'z' ) // song aborted by user
  pos=max+1;

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

 // T12, note length
 CCU61_T12PR=((float)current_note_length/(float)tempo*120.0); // period value note length
 CCU61_CC60SR=0; // not used (100% duty cycle)
 CCU61_CC61SR=0; // not used (100% duty cycle)
 // if compare value CCU6_CC62SR == 0 -> 100 % duty cycle note length
 CCU61_CC62SR=0;
 CCU61_vEnableShadowTransfer(CCU61_TIMER_12);

 // T13, note frequency
 CCU61_T13PR=T13_values[note]/octave; // note frequency
 CCU61_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %
 CCU61_vEnableShadowTransfer(CCU61_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,CCU61_T12PR);
       myprintf(mb2);
```

```
        sprintf(mb2,"T12-p=%1.2f (%1.2f)[s],
",current_note_length*124.12/1000.0/1000.0,CCU61_T12PR*124.12/1000.0/1000.0);
        myprintf(mb2);

        sprintf(mb2,"T13-pv=%5u (%5u),", T13_values[note]/octave,CCU61_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);
        }

  IO_vTogglePin(IO_P2_8); // Toggle P2.8
  CCU61_vStartTmr(CCU61_TIMER_12); // Start next note (T12 single shot)

     }
```

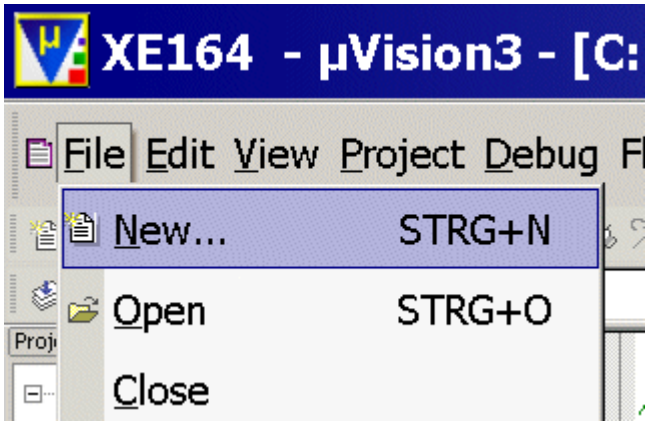

XE164 - µVision3 - [C:\XE164\CCU61.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Dave Files
Start_V3.A66
MAIN.C
SCS.C
IO.C
U0C0.C
USIC0.C
CC2.C
CCU61.C
User Files
myprintf.c
// Timer T12 period match detection
// USER CODE BEGIN (Node10,19)
if ((char)U0C0_RBUF == 'z') // song aborted by user
pos=max+1;
if (pos<=max)
read_song_string(); // read next note
// T12, note length
CCU61_T12PR=((float)current_note_length/(float)tempo*120.0); // period value note length
CCU61_CC60SR=0; // not used (100% duty cycle)
CCU61_CC61SR=0; // not used (100% duty cycle)
// if compare value CCU6_CC62SR == 0 -> 100 % duty cycle note length
CCU61_CC62SR=0;
CCU61_vEnableShadowTransfer(CCU61_TIMER_12);
// T13, note frequency
CCU61_T13PR=T13_values[note]/octave; // note frequency
CCU61_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %
CCU61_vEnableShadowTransfer(CCU61_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,CCU61_T12PR);
myprintf(mb2);
sprintf(mb2,"T12-p=%1.2f (%1.2f)[s], ",current_note_length*124.12/1000.0/1000.0,CCU61_T12PR*124.12/1000.0/1000.0);
myprintf(mb2);
sprintf(mb2,"T13-pv=%5u (%5u),", T13_values[note]/octave,CCU61_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);
IO_vTogglePin(IO_P2_8); // Toggle P2.8
CCU61_vStartTmr(CCU61_TIMER_12); // Start next note (T12 single shot)
// USER CODE END
MAIN.C
CC2.C
CCU61.C
Simulation
L:445 C:6
R/W

Double click CCU61.C insert Global Variables:

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



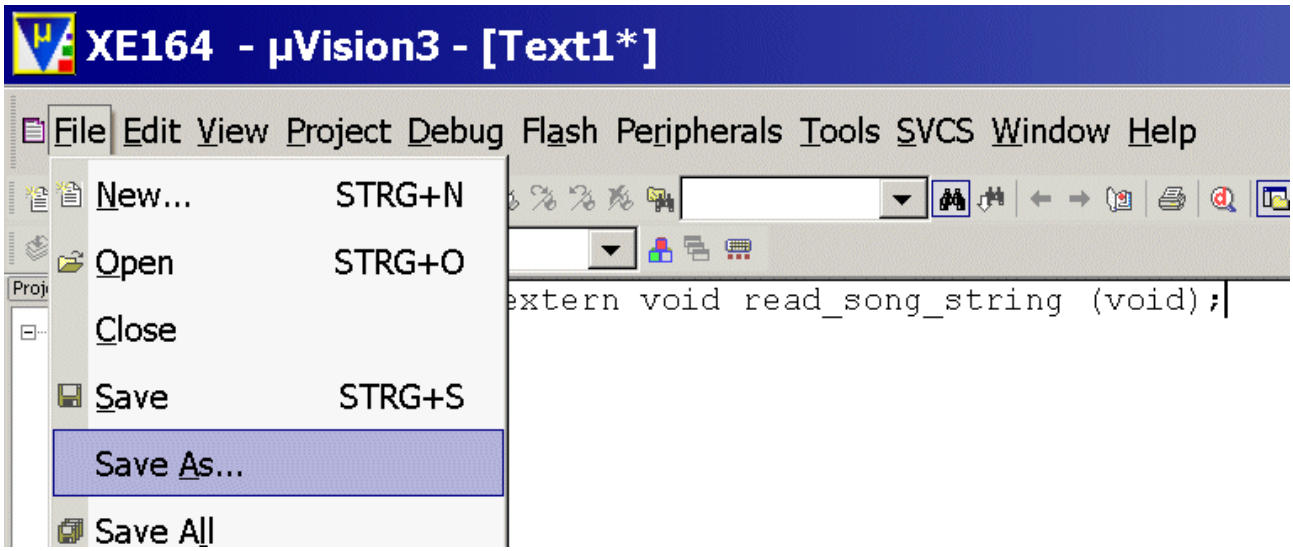
File – New



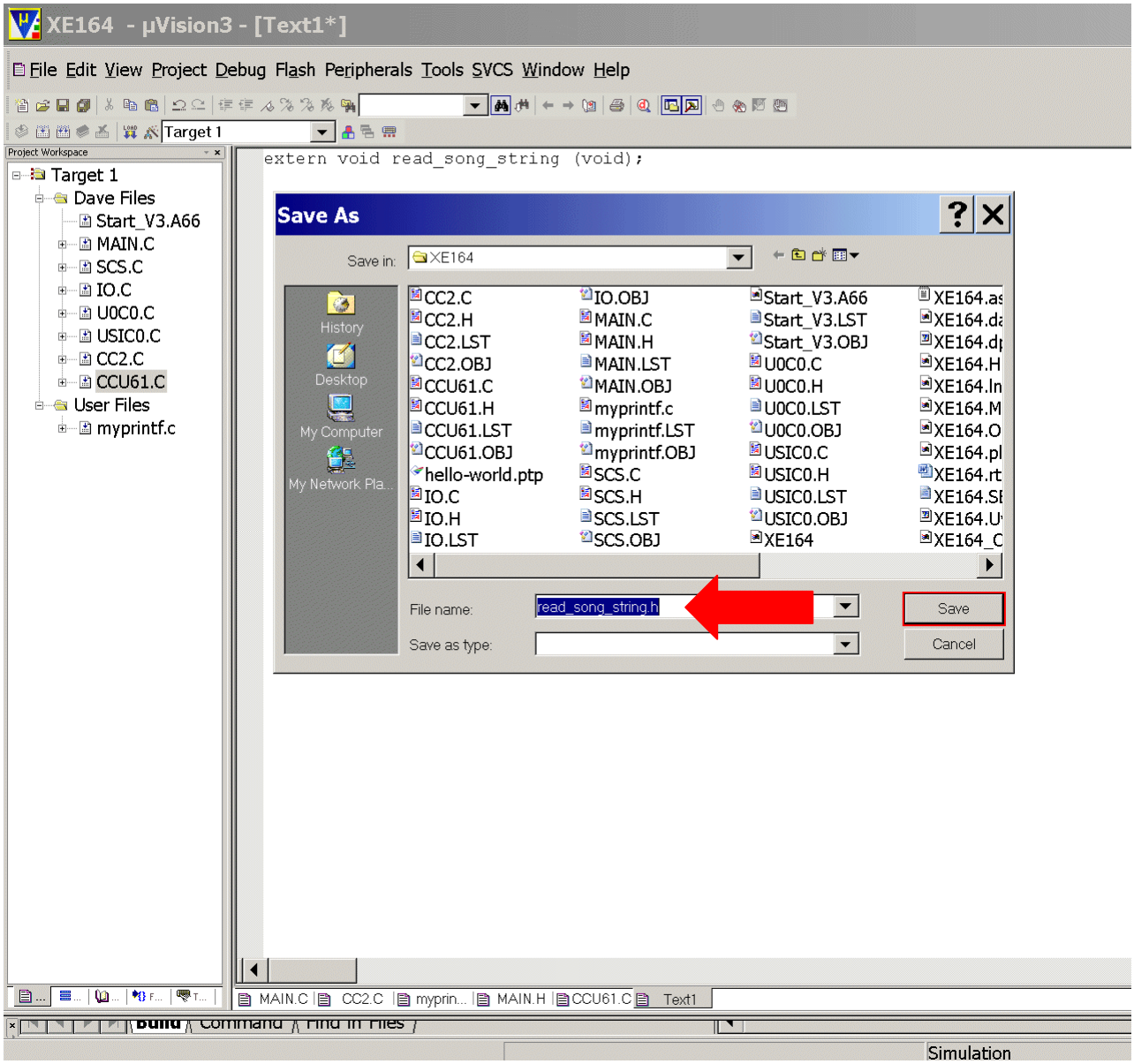
Insert:

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

File – Save As…

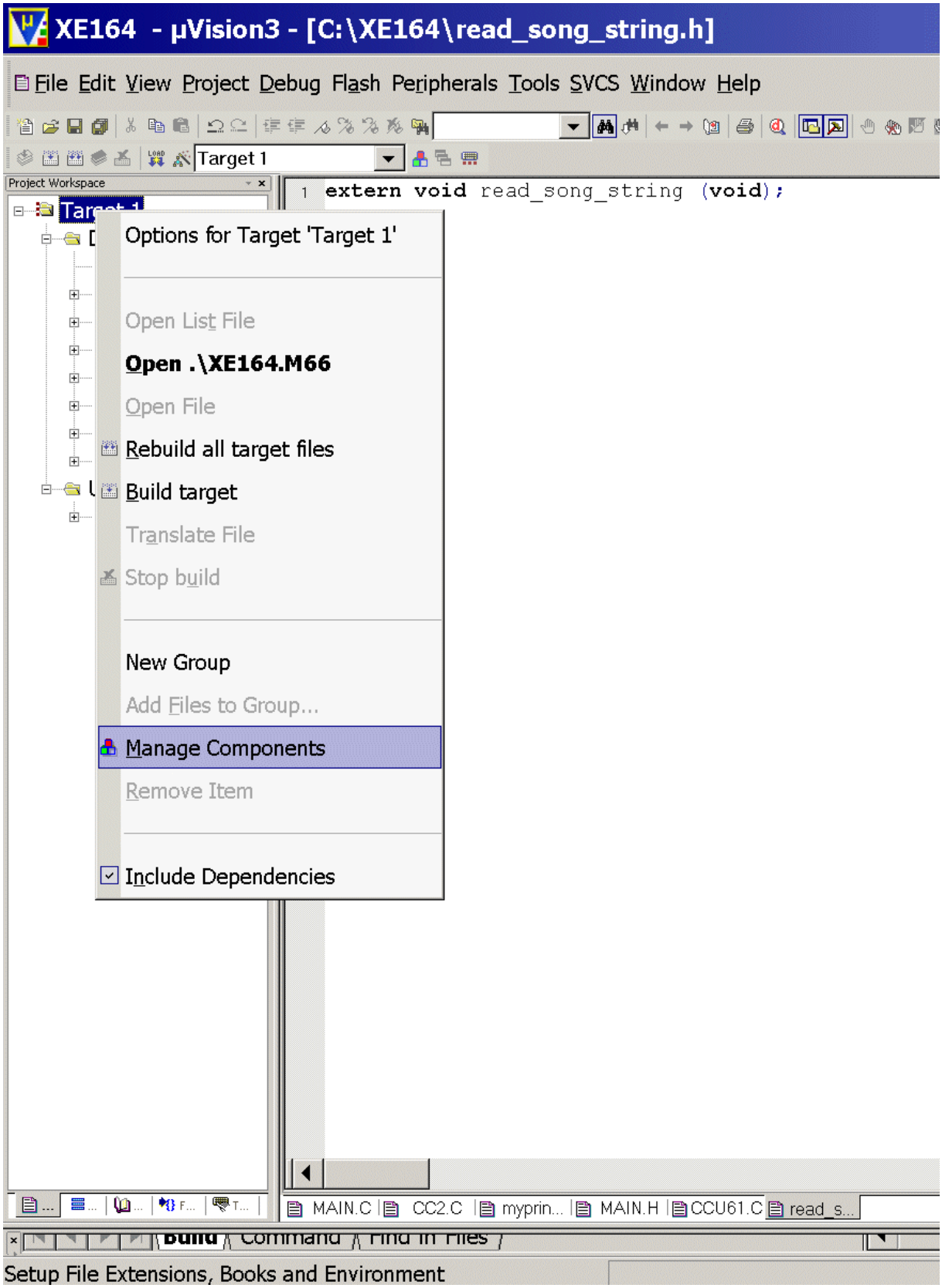


Insert: read_song_string.h

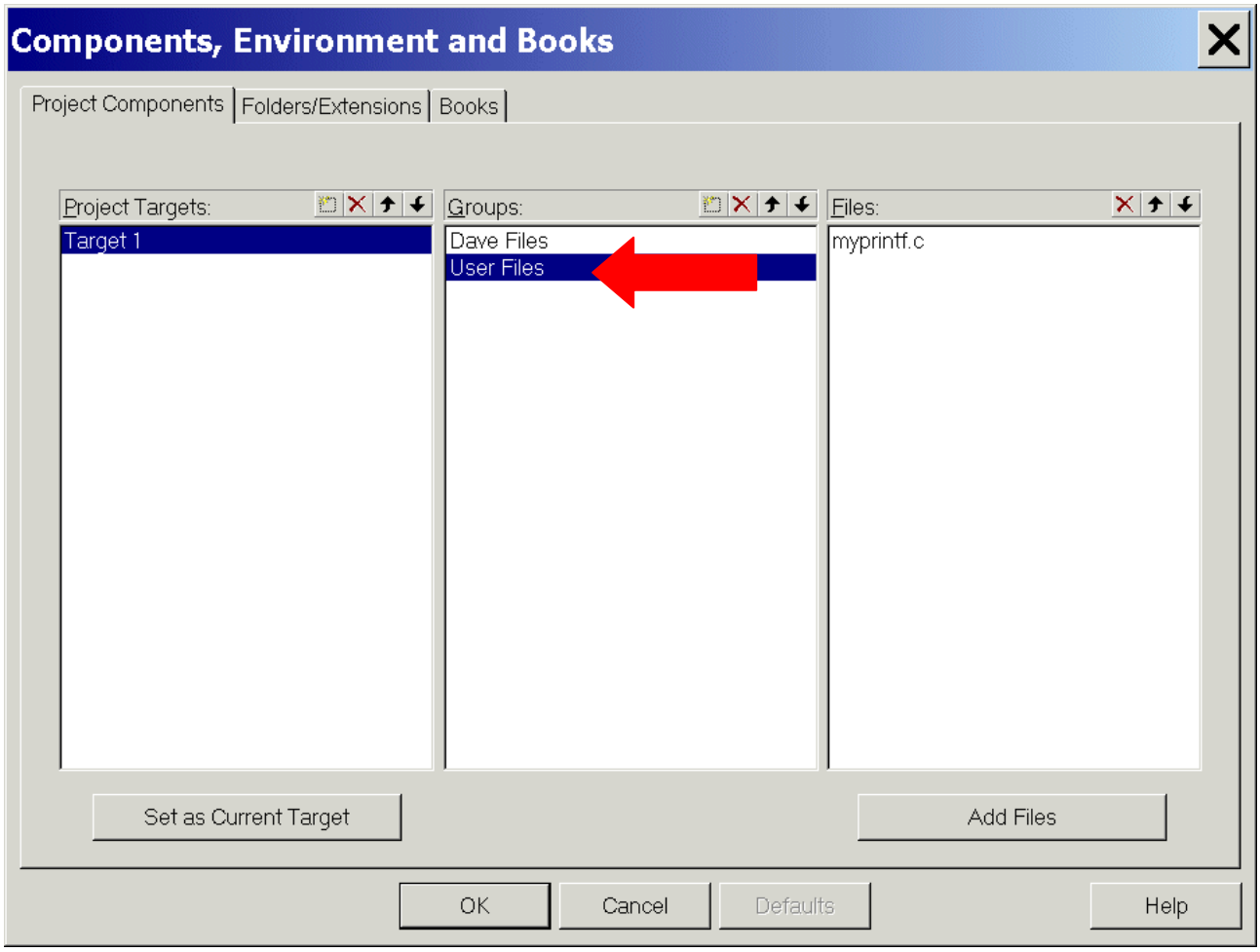


Click Save

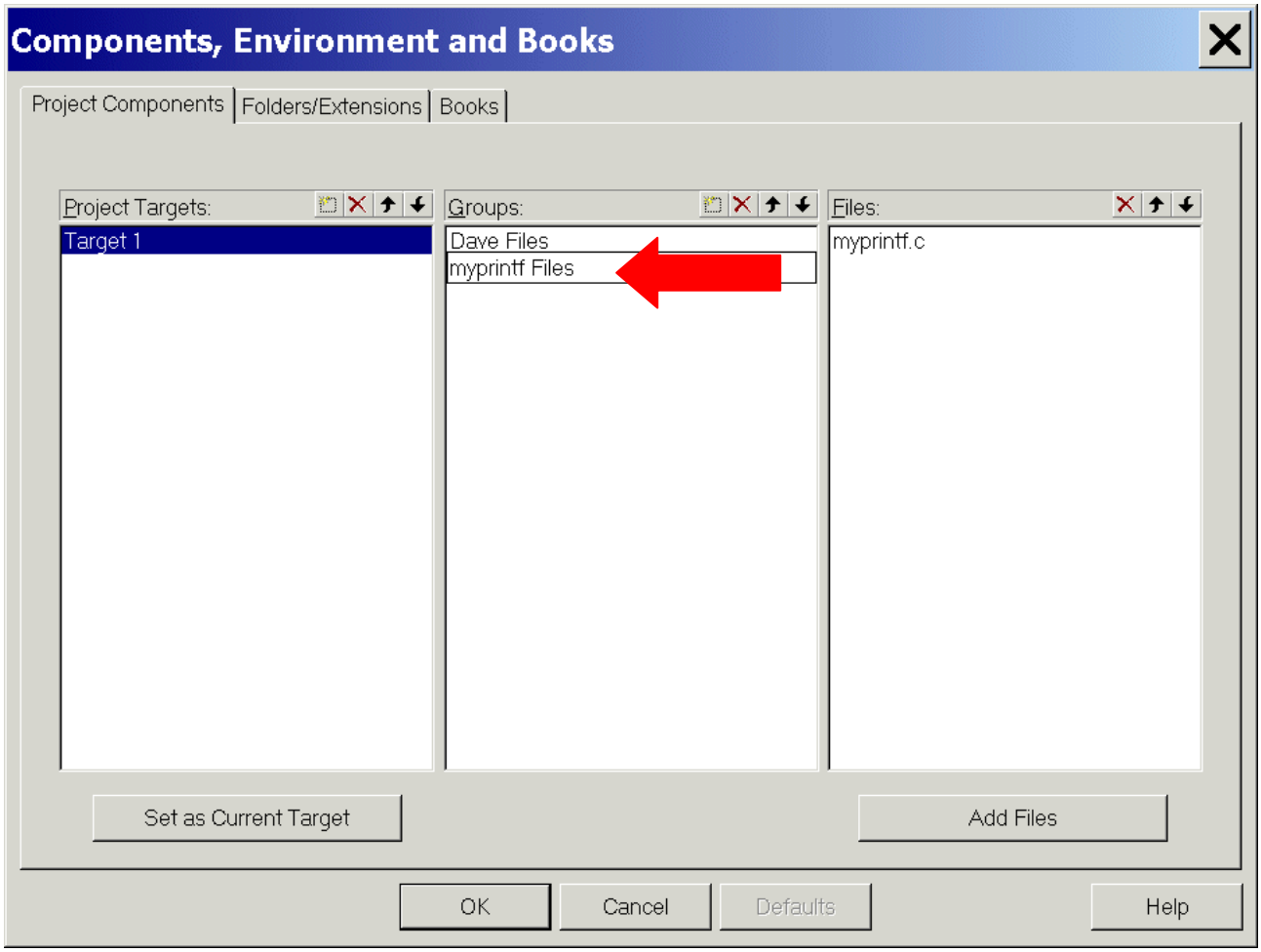
Mouse position: Project Window, Target 1: click right mouse button
click Manage Components



Double click User Files:

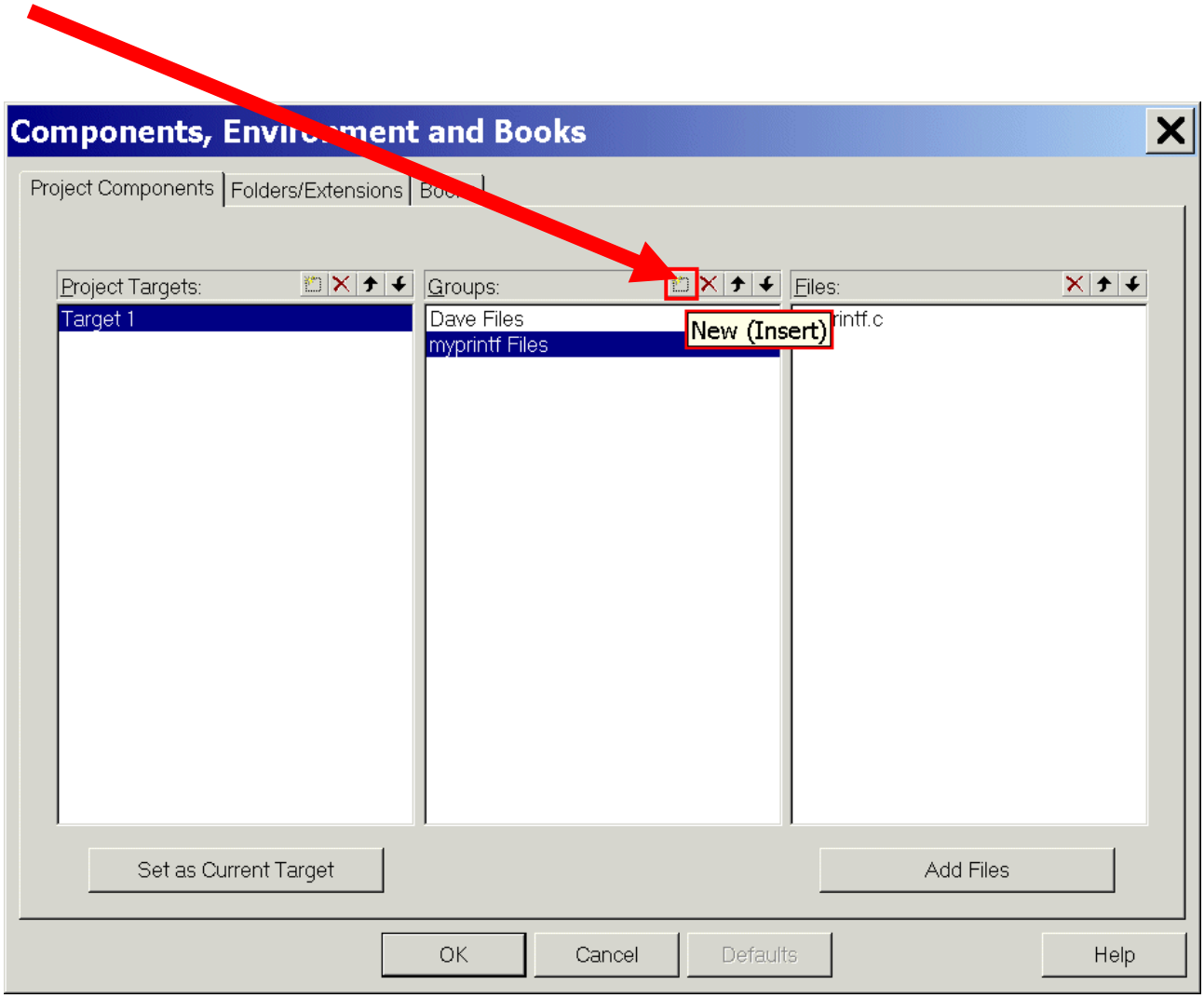


Insert: myprintf Files:

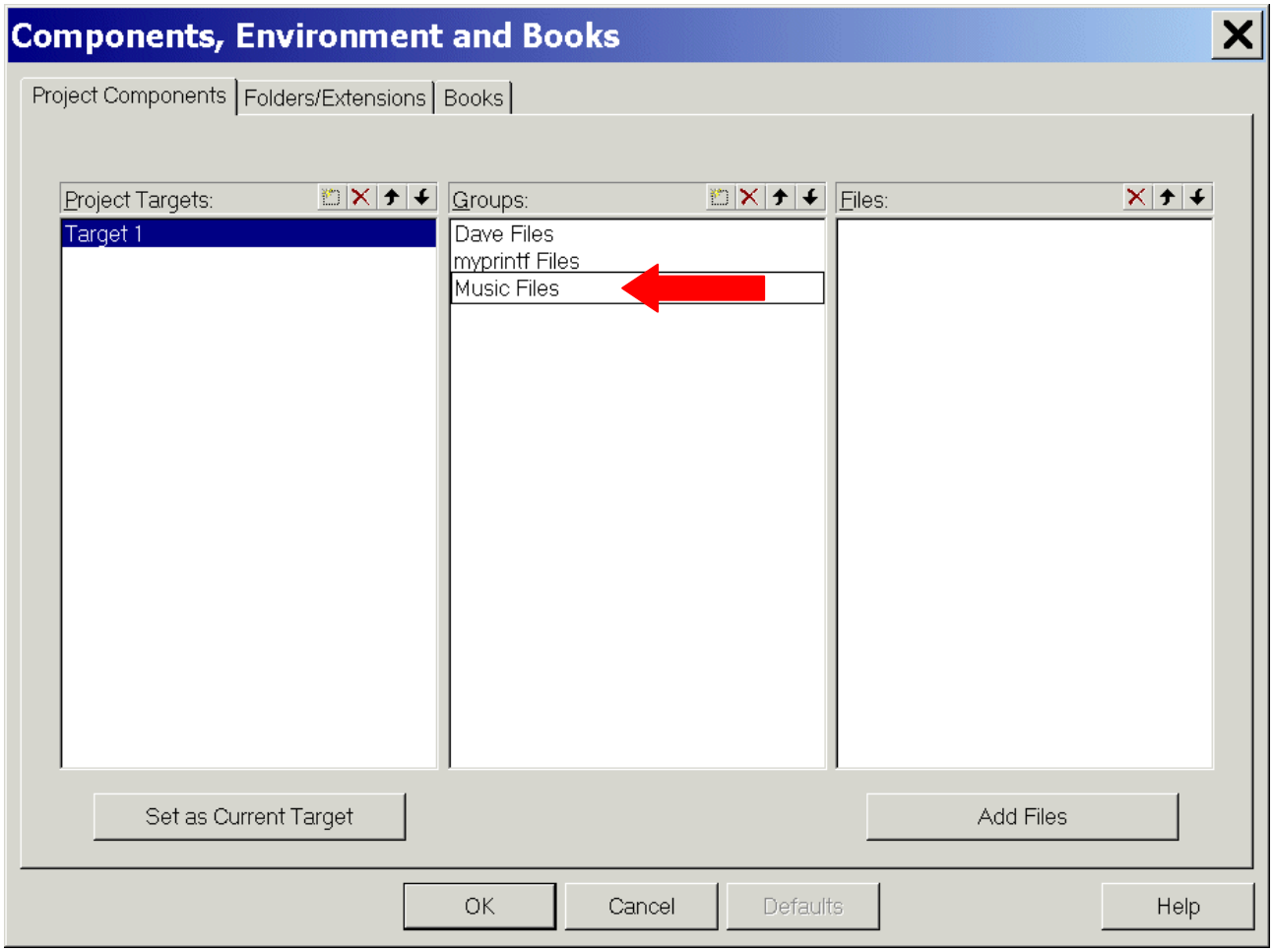


<ENTER>

Click New (Insert):

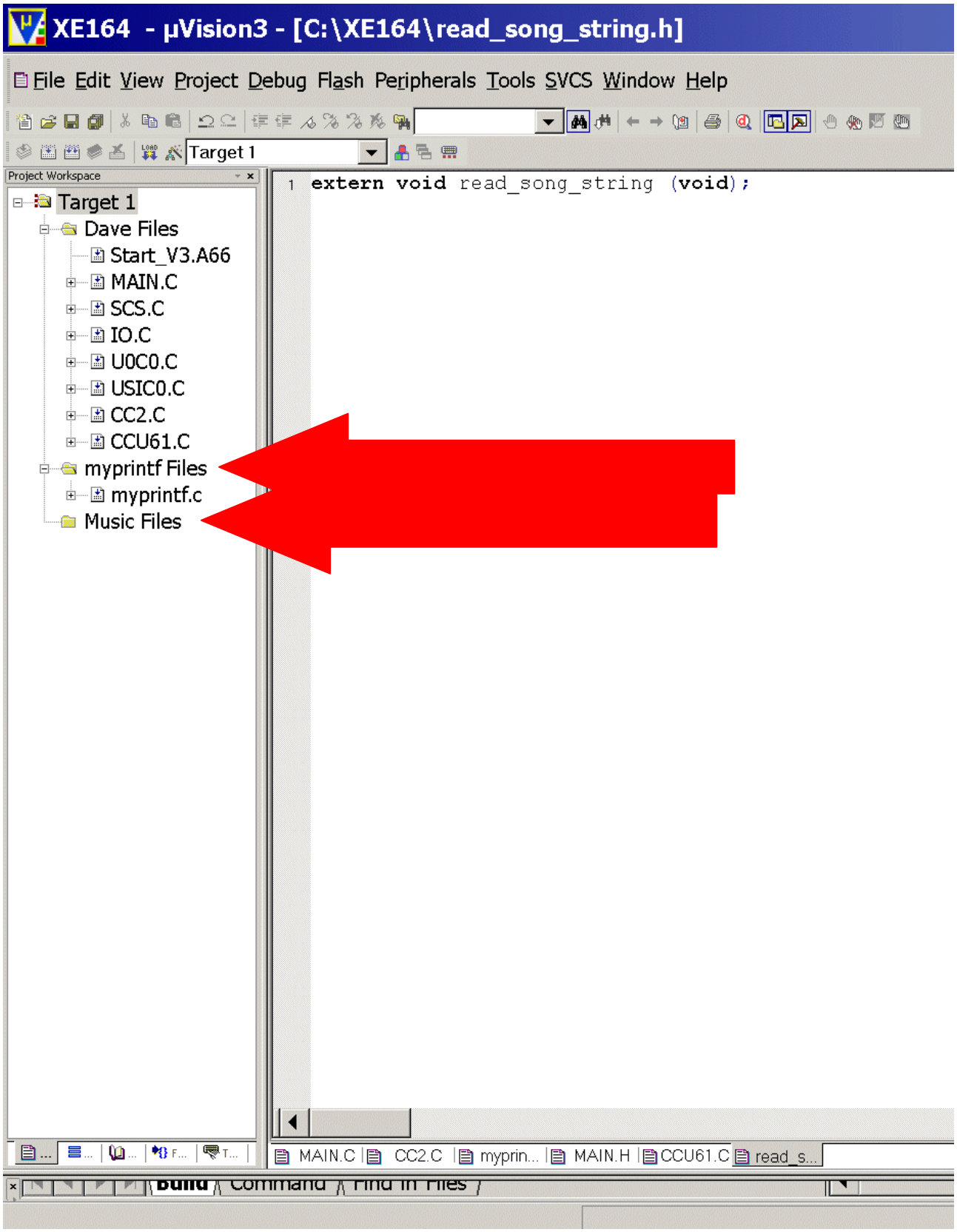


Insert Music Files:

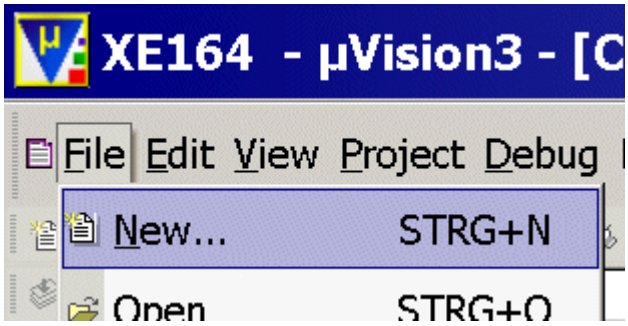


<ENTER>

OK


XE164 - µVision3 - [C:\XE164\read_song_string.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
SCS.C
IO.C
U0C0.C
USIC0.C
CC2.C
CCU61.C
myprintf Files
myprintf.c
Music Files
1 extern void read_song_string (void);
MAIN.C
CC2.C
myprin...
MAIN.H
CCU61.C
read_s...
Build
Command
Find in Files

File – New



Insert:

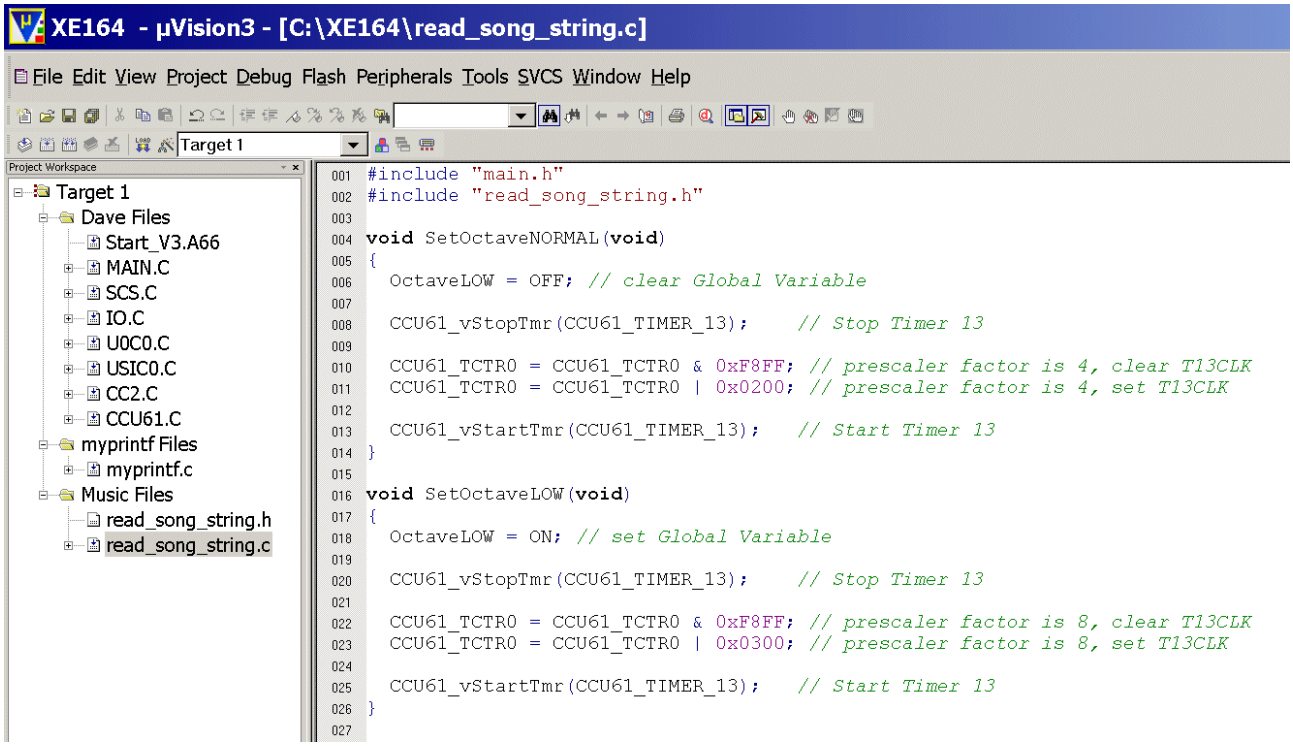
```
#include "main.h"
#include "read_song_string.h"

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

  CCU61_vStopTmr(CCU61_TIMER_13);    // Stop Timer 13

  CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 4, clear T13CLK
  CCU61_TCTR0 = CCU61_TCTR0 | 0x0200; // prescaler factor is 4, set T13CLK

  CCU61_vStartTmr(CCU61_TIMER_13);   // Start Timer 13
}

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

  CCU61_vStopTmr(CCU61_TIMER_13);    // Stop Timer 13

  CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 8, clear T13CLK
  CCU61_TCTR0 = CCU61_TCTR0 | 0x0300; // prescaler factor is 8, set T13CLK

  CCU61_vStartTmr(CCU61_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 */
```

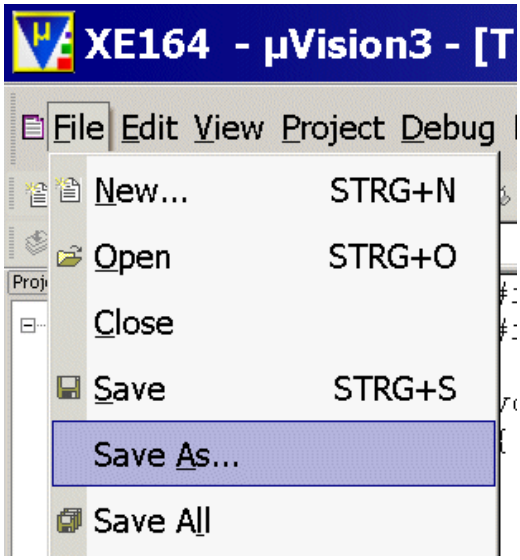

XE164 - µVision3 - [C:\XE164\read_song_string.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Dave Files
Start_V3.A66
MAIN.C
SCS.C
IO.C
U0C0.C
USIC0.C
CC2.C
CCU61.C
myprintf Files
myprintf.c
Music Files
read_song_string.h
read_song_string.c
001 #include "main.h"
002 #include "read_song_string.h"
003
004 void SetOctaveNORMAL(void)
005 {
006 OctaveLOW = OFF; // clear Global Variable
007
008 CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
009
010 CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 4, clear T13CLK
011 CCU61_TCTR0 = CCU61_TCTR0 | 0x0200; // prescaler factor is 4, set T13CLK
012
013 CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
014 }
015
016 void SetOctaveLOW(void)
017 {
018 OctaveLOW = ON; // set Global Variable
019
020 CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
021
022 CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 8, clear T13CLK
023 CCU61_TCTR0 = CCU61_TCTR0 | 0x0300; // prescaler factor is 8, set T13CLK
024
025 CCU61_vStartTmr(CCU61_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
```

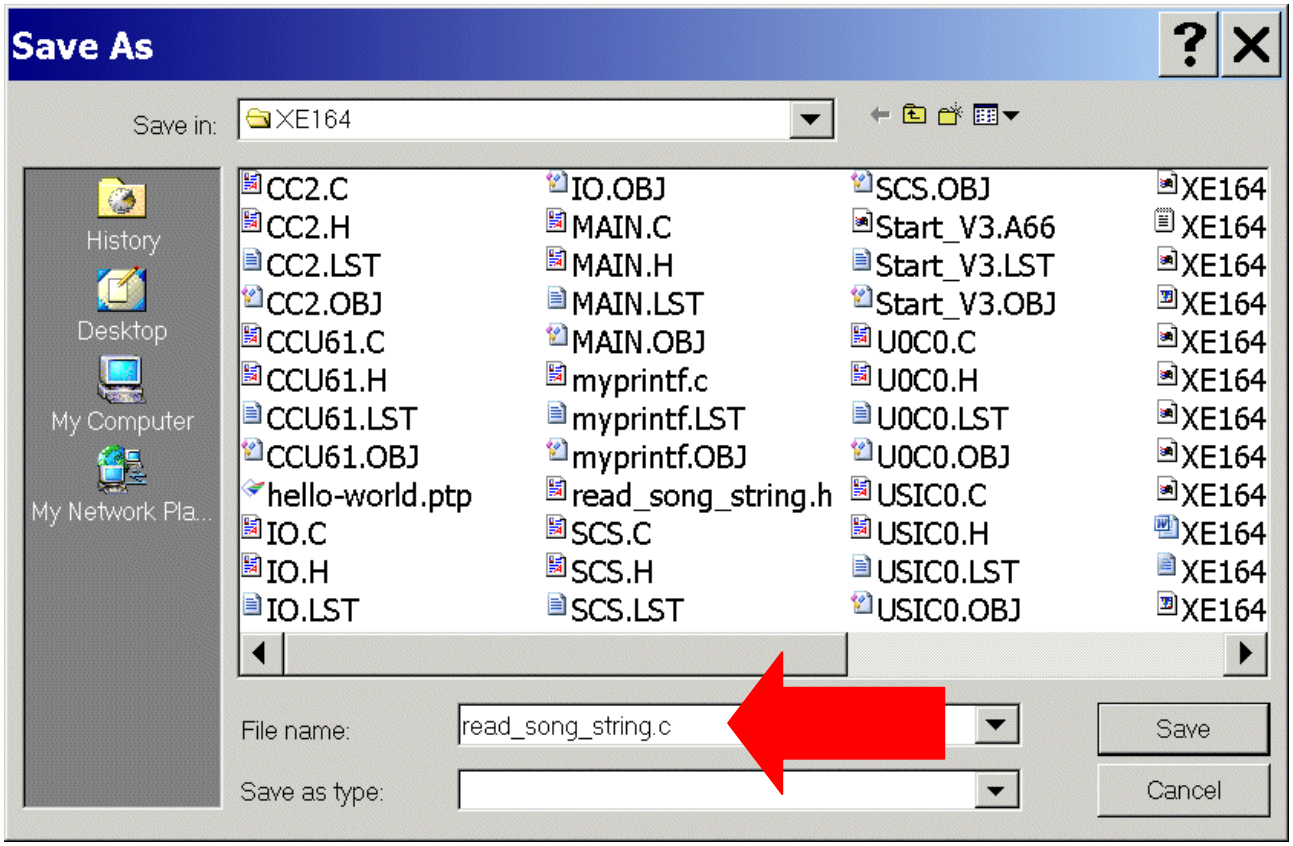
```
// adjust note length:
   case 'L': switch (song[++pos])
             {
                 case '1': if (song[++pos]=='6')
                              current_note_length=length_of_a_whole_note/16;
                           else
                           {
                              pos--;
                              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 */
```

File – Save As…

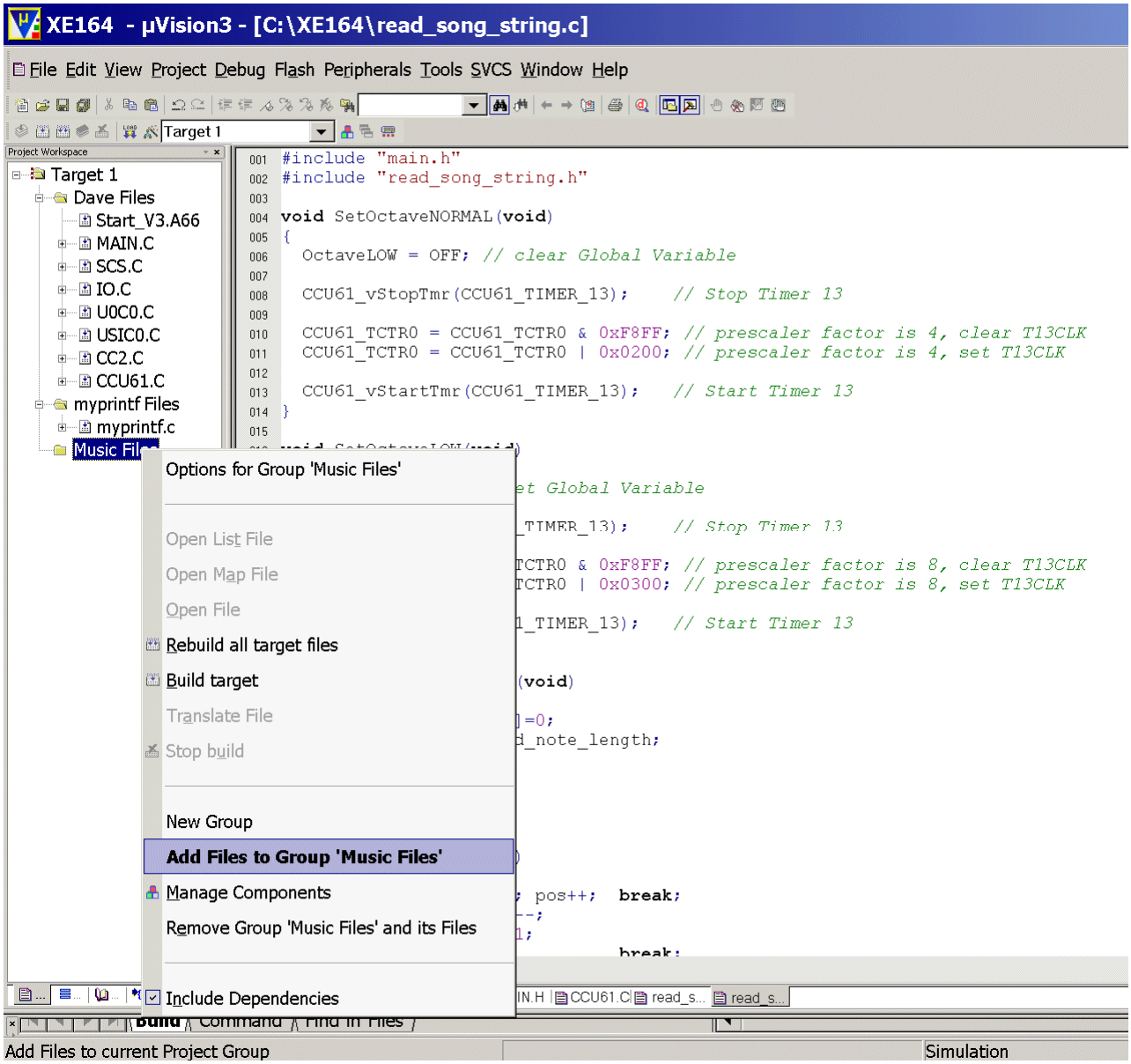


Insert: read_song_string.c

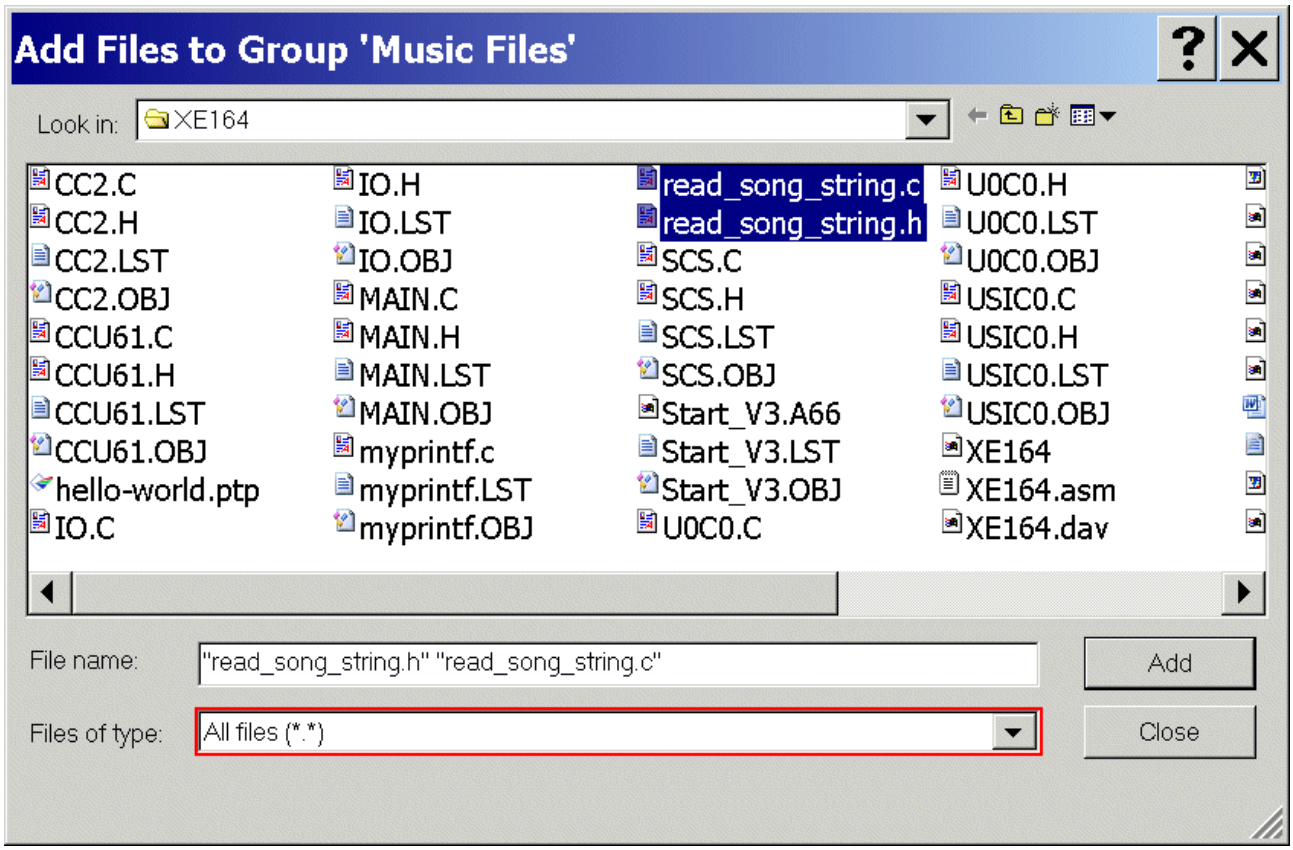


Click Save

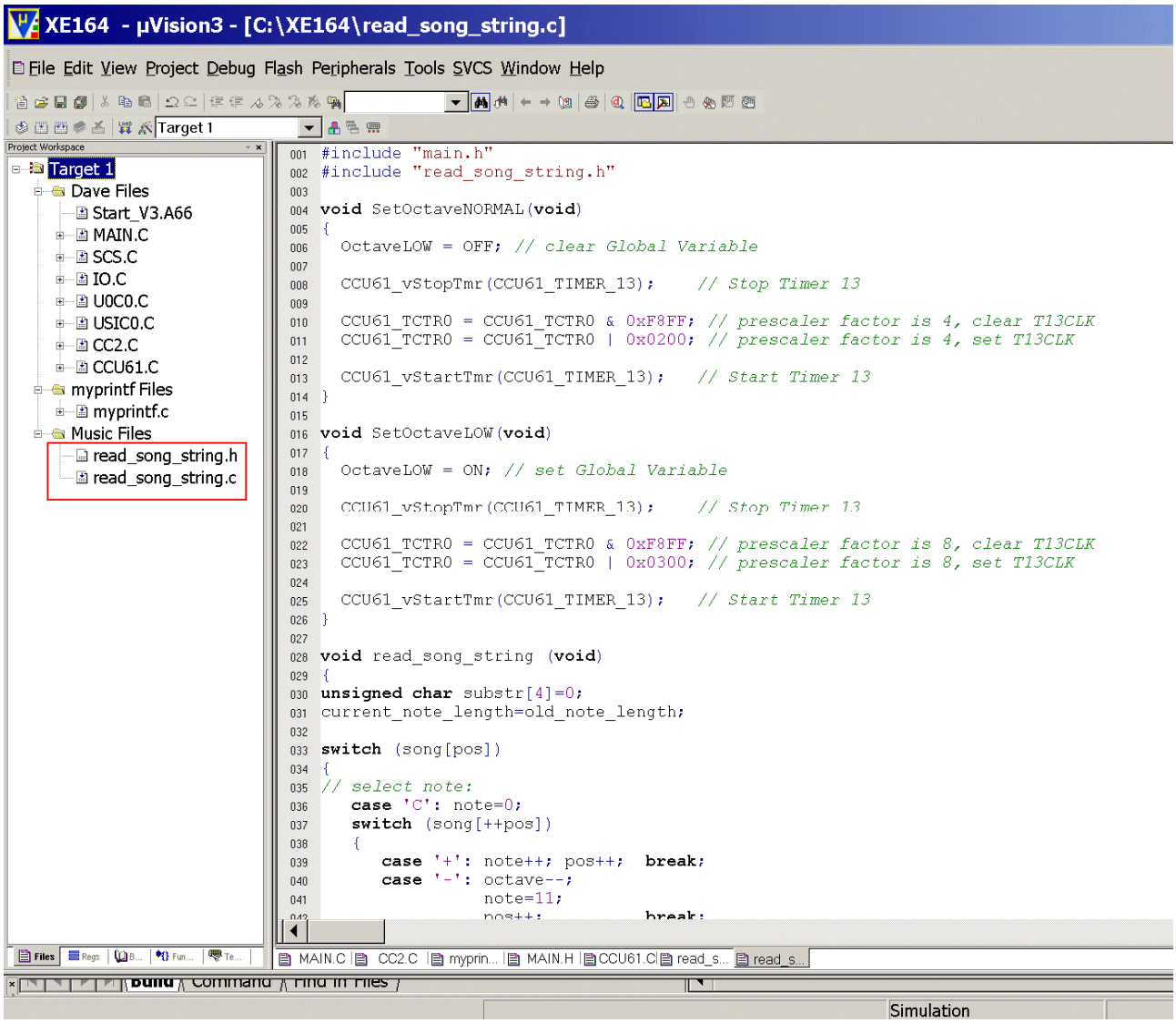
Mouse position: Project Window, Music Files: click right mouse button
Click Add Files to Group ‘Music Files’



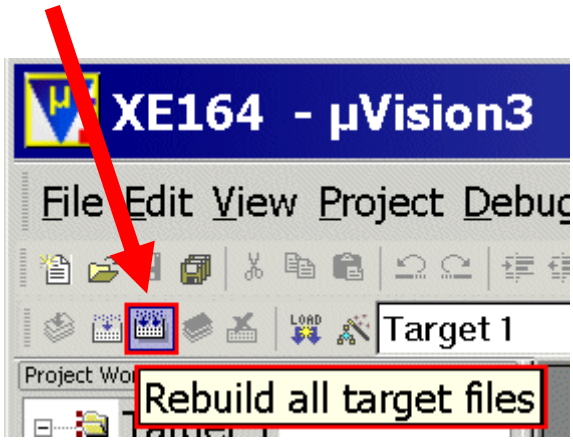
Files of type: select All files
Mark read_song_string.h and read_song_string.c

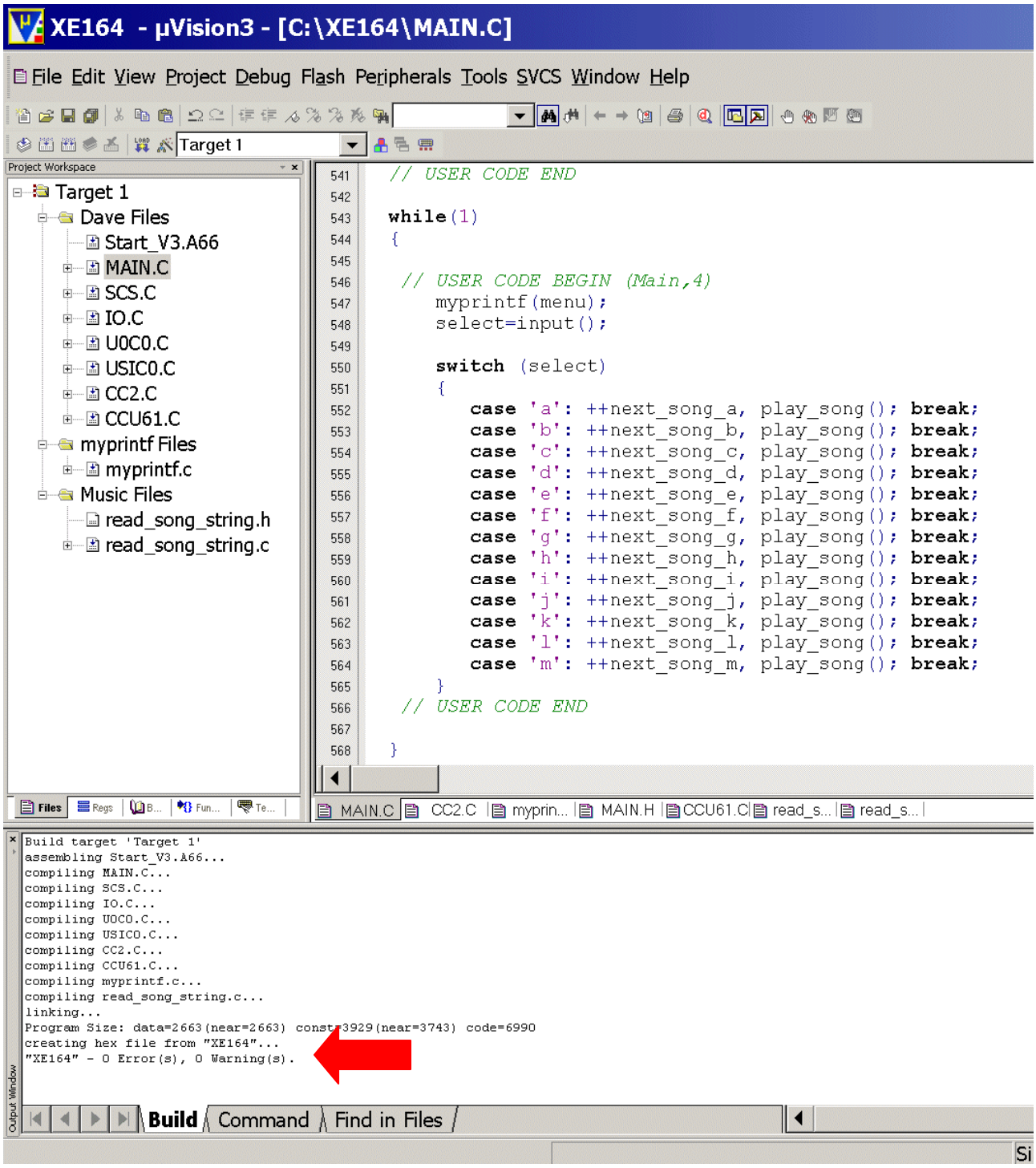


Click Add
Click Close

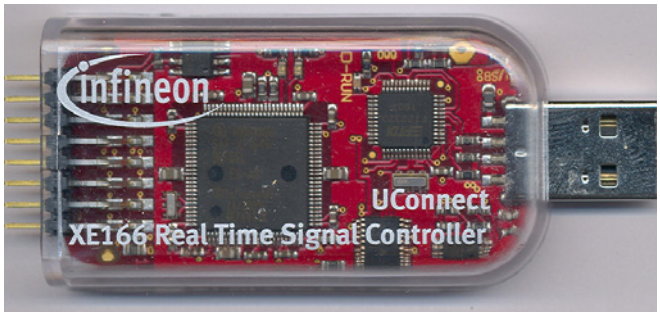

XE164 - µVision3 - [C:\XE164\read_song_string.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Dave Files
Start_V3.A66
MAIN.C
SCS.C
IO.C
U0C0.C
USIC0.C
CC2.C
CCU61.C
myprintf Files
myprintf.c
Music Files
read_song_string.h
read_song_string.c
#include "main.h"
#include "read_song_string.h"
void SetOctaveNORMAL(void)
{
OctaveLOW = OFF; // clear Global Variable
CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 4, clear T13CLK
CCU61_TCTR0 = CCU61_TCTR0 | 0x0200; // prescaler factor is 4, set T13CLK
CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
}
void SetOctaveLOW(void)
{
OctaveLOW = ON; // set Global Variable
CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 8, clear T13CLK
CCU61_TCTR0 = CCU61_TCTR0 | 0x0300; // prescaler factor is 8, set T13CLK
CCU61_vStartTmr(CCU61_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;
MAIN.C CC2.C myprin... MAIN.H CCU61.C read_s... read_s...
Build Command Find in Files
Simulation

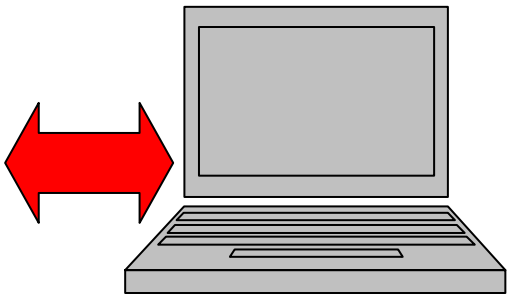
Generate your application program:

Project – Rebuild all target files	or	click 


XE164 - µVision3 - [C:\XE164\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
CCU61.C
myprintf Files
myprintf.c
Music Files
read_song_string.h
read_song_string.c
Files
Regs
B...
Fun...
Te...
541 // USER CODE END
542
543 while(1)
544 {
545
546 // USER CODE BEGIN (Main,4)
547 myprintf(menu);
548 select=input();
549
550 switch (select)
551 {
552 case 'a': ++next_song_a, play_song(); break;
553 case 'b': ++next_song_b, play_song(); break;
554 case 'c': ++next_song_c, play_song(); break;
555 case 'd': ++next_song_d, play_song(); break;
556 case 'e': ++next_song_e, play_song(); break;
557 case 'f': ++next_song_f, play_song(); break;
558 case 'g': ++next_song_g, play_song(); break;
559 case 'h': ++next_song_h, play_song(); break;
560 case 'i': ++next_song_i, play_song(); break;
561 case 'j': ++next_song_j, play_song(); break;
562 case 'k': ++next_song_k, play_song(); break;
563 case 'l': ++next_song_l, play_song(); break;
564 case 'm': ++next_song_m, play_song(); break;
565 }
566 // USER CODE END
567
568 }
MAIN.C
CC2.C
myprin...
MAIN.H
CCU61.C
read_s...
read_s...
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 CCU61.C...
compiling myprintf.c...
compiling read_song_string.c...
linking...
Program Size: data=2663(near=2663) const=3929(near=3743) code=6990
creating hex file from "XE164"...
"XE164" - 0 Error(s), 0 Warning(s).
Output Window
Build
Command
Find in Files

Make sure that the UConnect-CAN XE164 is still connected to the host computer:





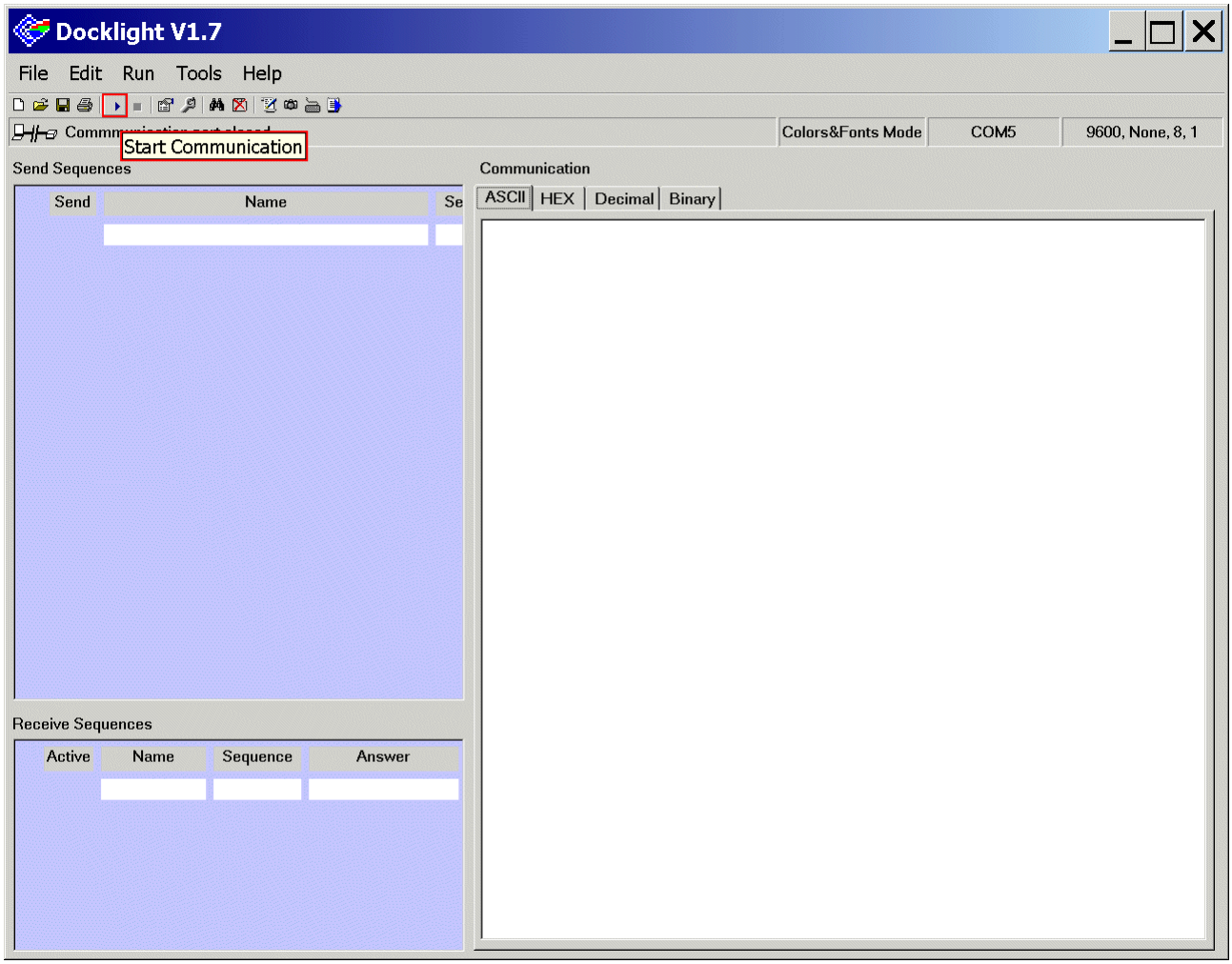
USB Connection:
.) used for: UART communication (the USIC0_CH0/UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
.) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
.) the USB connection works also as the power supply.

Connect your active loudspeaker(s) to the UConnect-CAN XE164:

Connect CCU61_CC62 (P0.2) and GND (VSS) to your active loudspeaker(s):

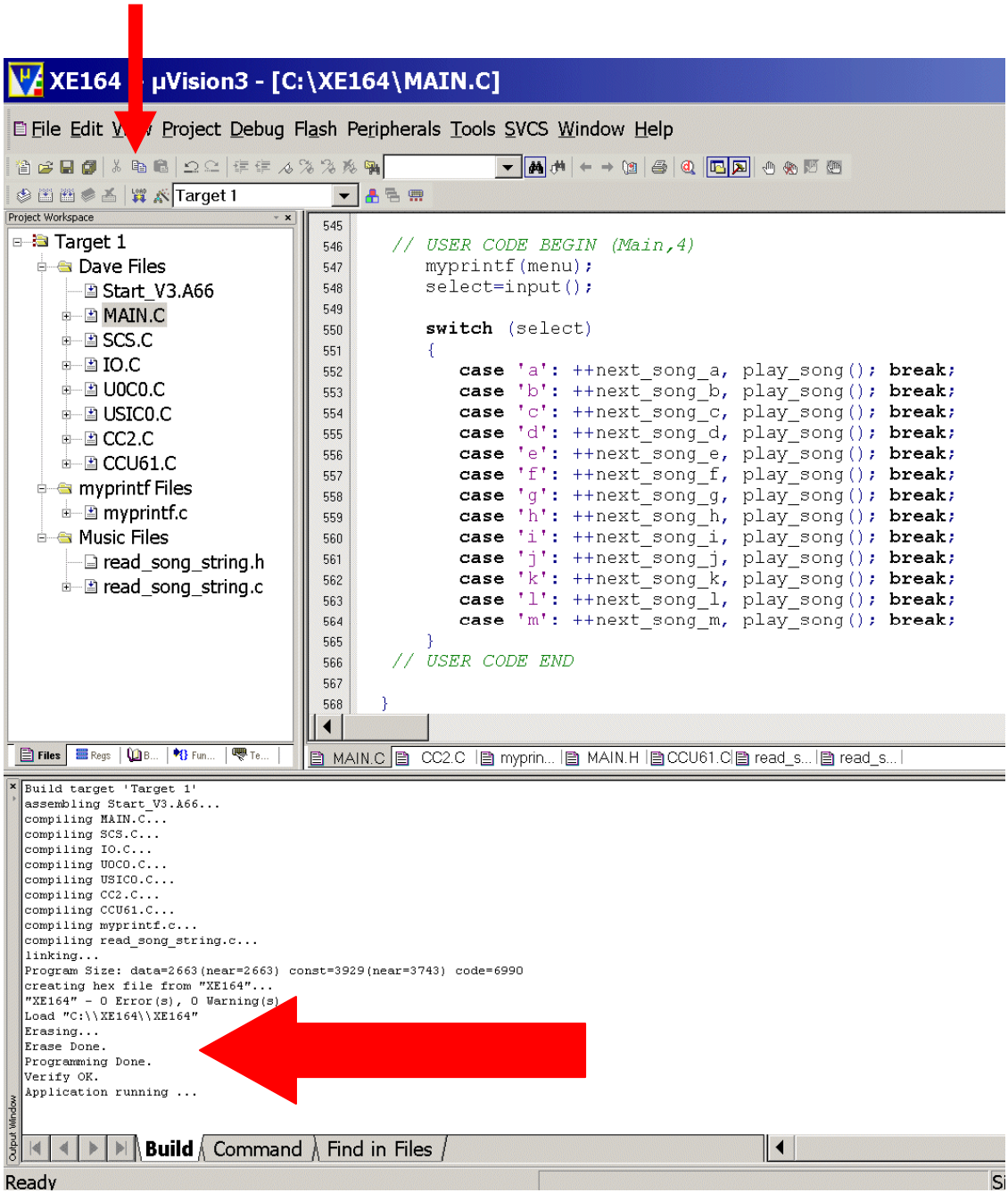
Pin number				
1	Ground			
2	+5V			
3	P15.0	ADC1_CH0		
4	P5.0	ADC0_CH0		
5	P5.9	ADC0_CH9	ADC1_CH9	CC2_T7IN CAPCOM2
6	P5.8	ADC0_CH8	ADC8_CH8	T12HRC / T13HRC CCU6x
7	P0.2	U1C0_SCK	CC62 CCU61	TXDC0 CAN0
8	P0.1	U1C0_DOUT	CC61 CCU61	TXDC0 CAN0
9	P0.3	U1C0_SELO	COUT60 CCU61	RXDC0B CAN0
10	P0.0	U1C0_DX0	CC60 CCU61	
11	P0.5	U1C1_SCK	COUT62 CCU61	
12	P0.6	U1C1_DOUT	COUT63 CCU61	TXDC1 CAN1
13	P0.4	U1C1_SELO	COUT61CCU61	RXDC1B CAN1
14	P0.7	U1C1_DX0	U1C1_DX0	CTRAPB CCU61
15	CANH Signal from CAN transceiver			
16	CANL Signal from CAN transceiver			

Now, start Docklight and click  (Start Communication):

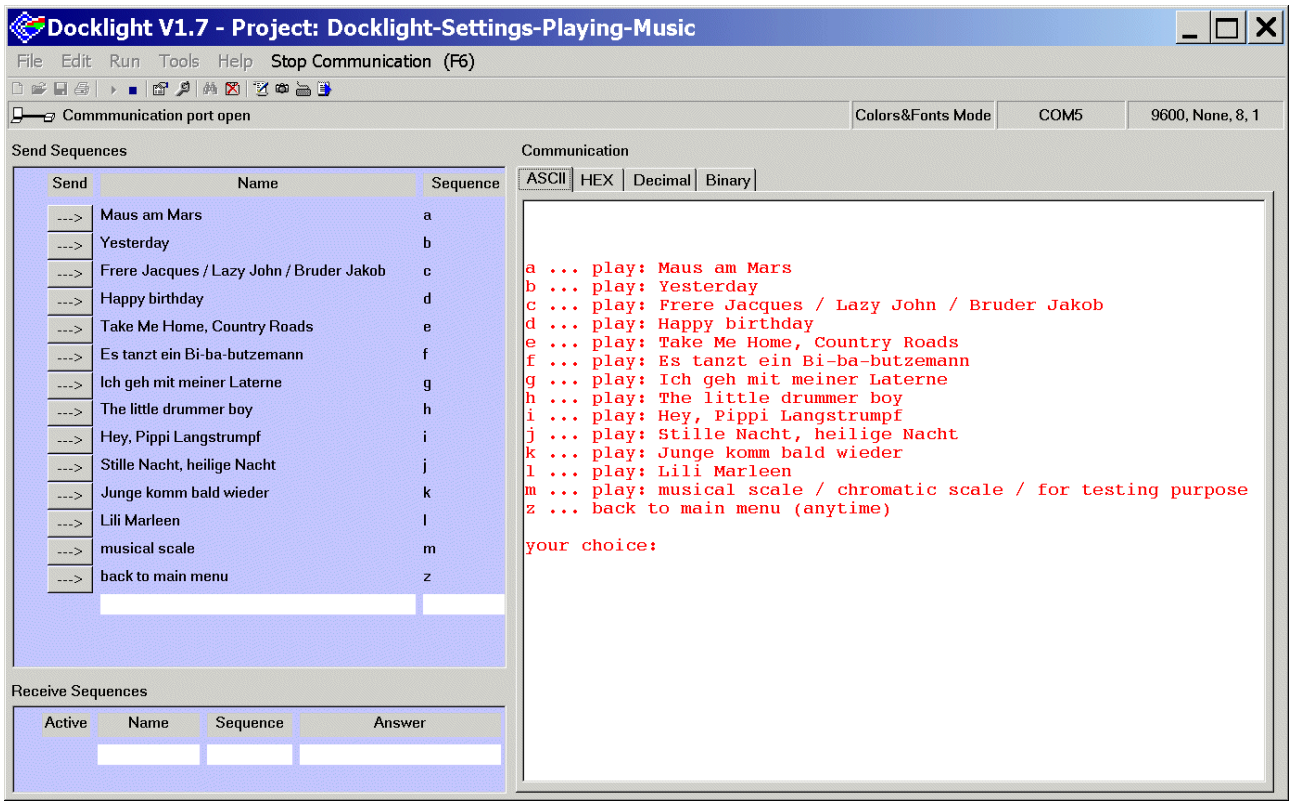


Go to µVision, Download/program your application program into the On Chip Flash:

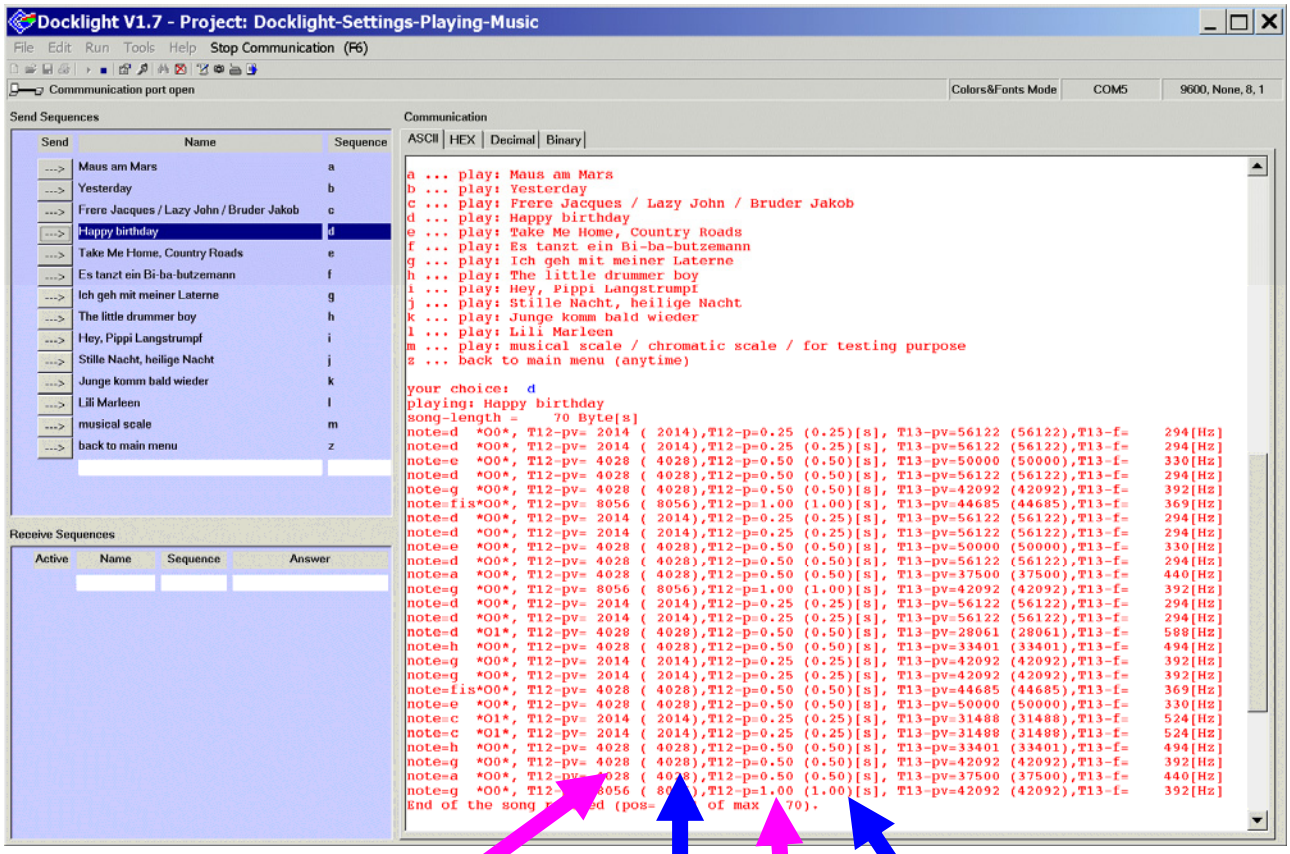
Click: 



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



Insert/select d

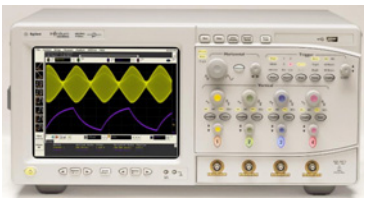


T12 period value @ tempo = 120 beats/minute

T12 period value @ current tempo

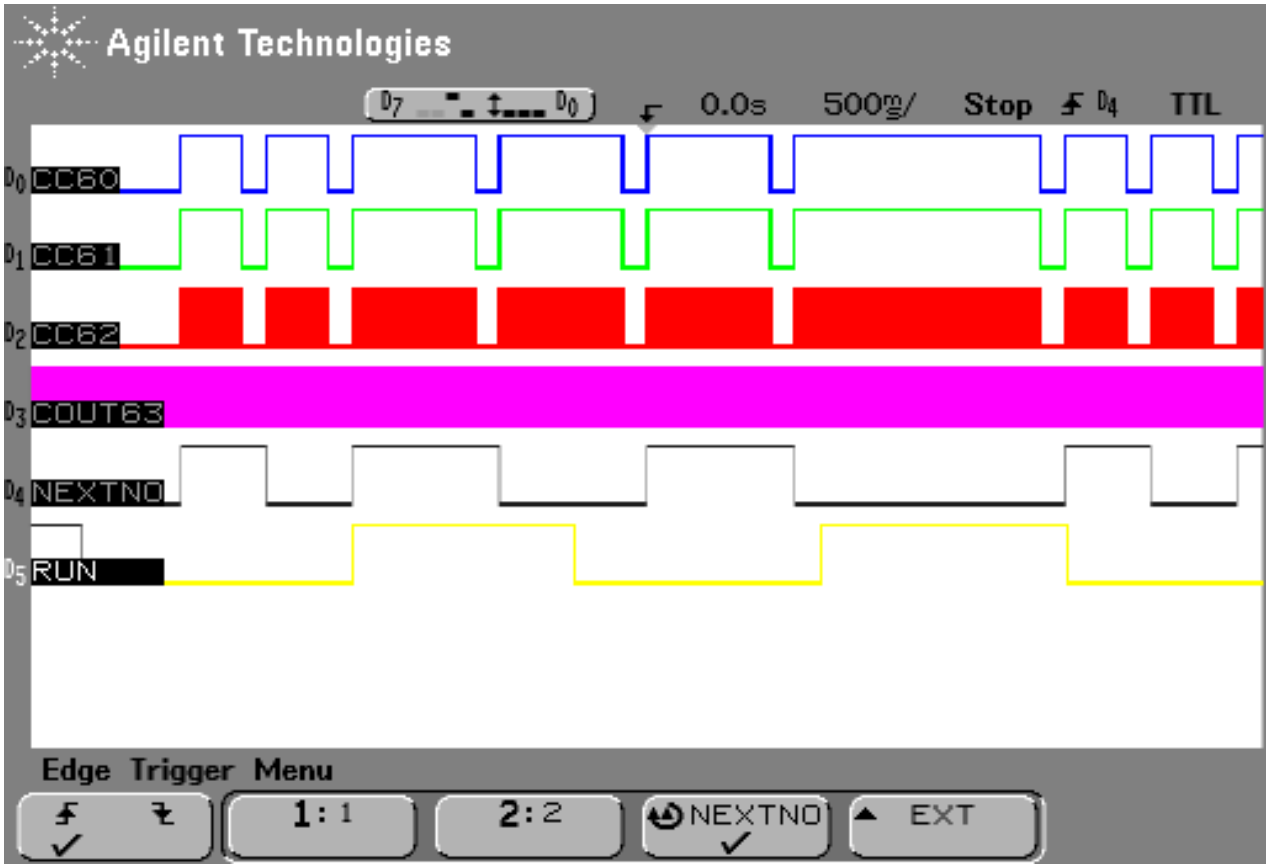
note length @ tempo = 120 beats/minute

Note length @ current tempo

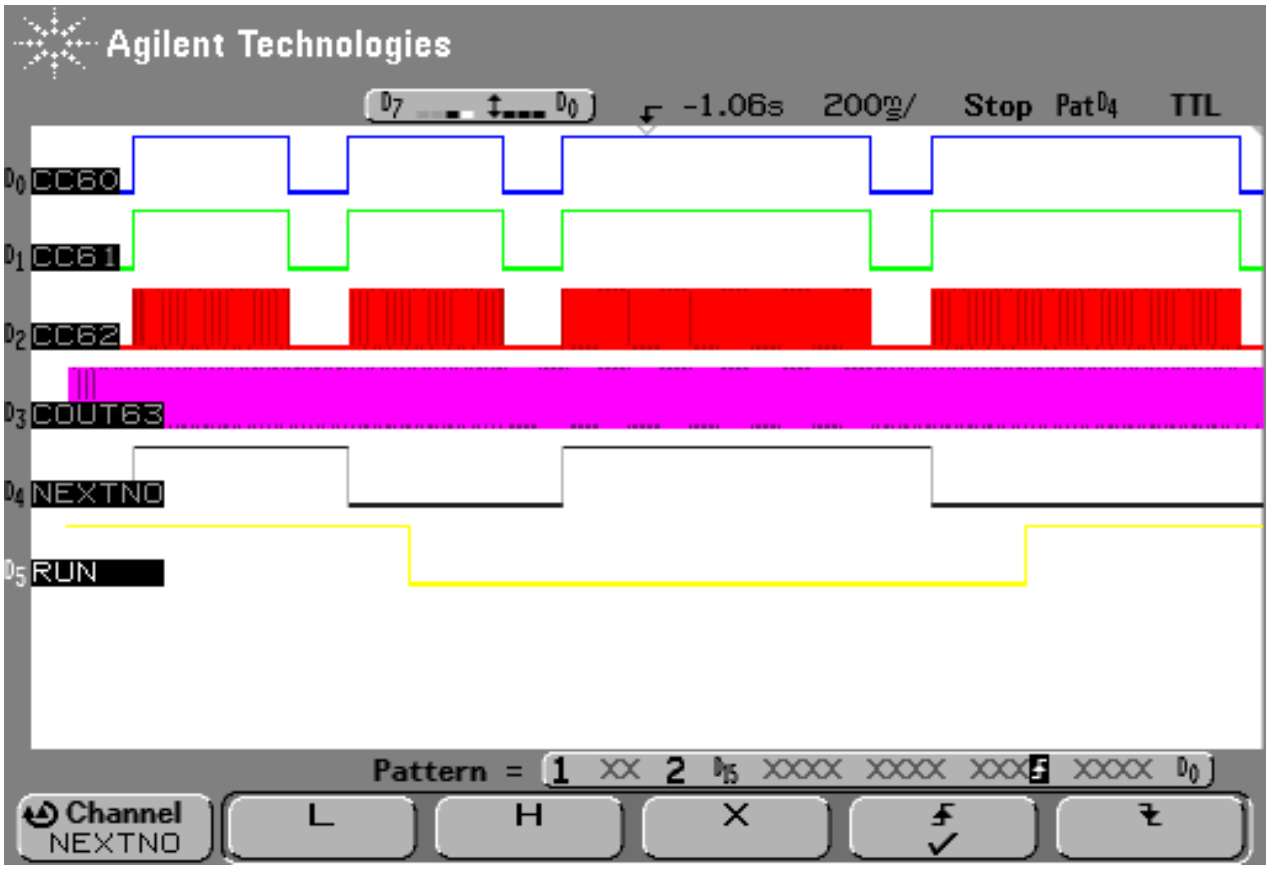


Use any Logic Analyser (LGA) / scope and see the result:

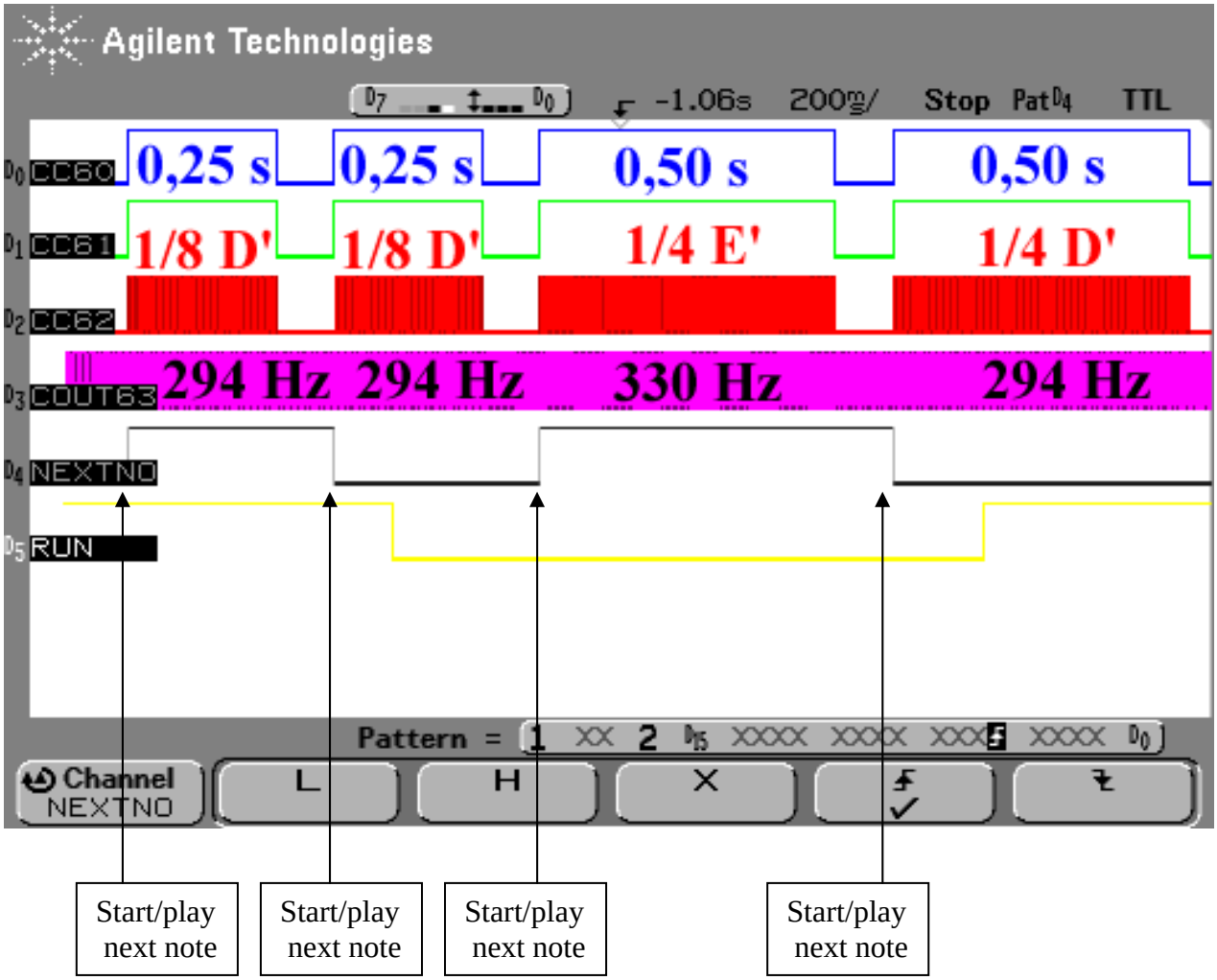
Note:
Song d: Happy birthday:
code unsigned char
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";



HAPPY BIRTHDAY =
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F… ...



HAPPY BIRTHDAY =
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F… ...



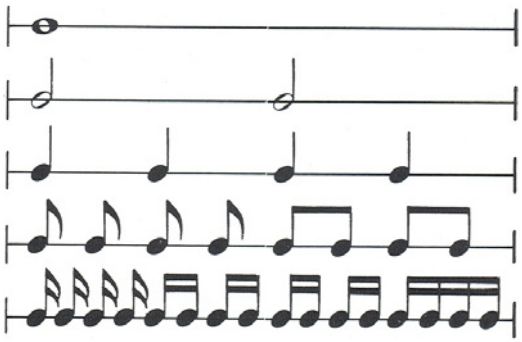
Note:
Start/play next note:
IO_vTogglePin(IO_P2_8); // Show start of next note on Port 2.8
CCU61_vStartTmr(CCU61_TIMER_12); // Start next note (T12 single shot)

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:

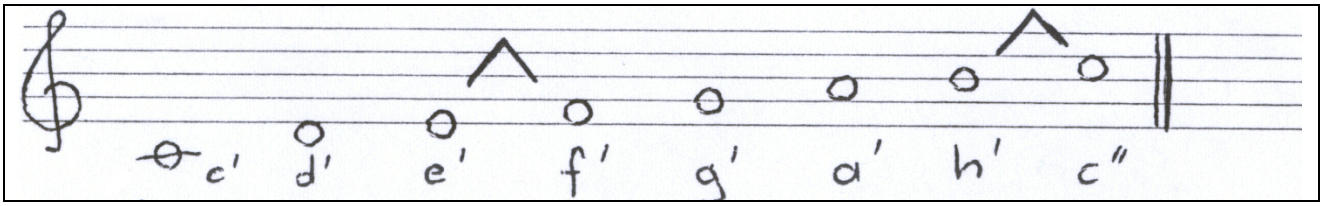
note	LENGTH
	½ (2 beats) + (½)/2 (1 beat) = ¾ (3 beats)

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:

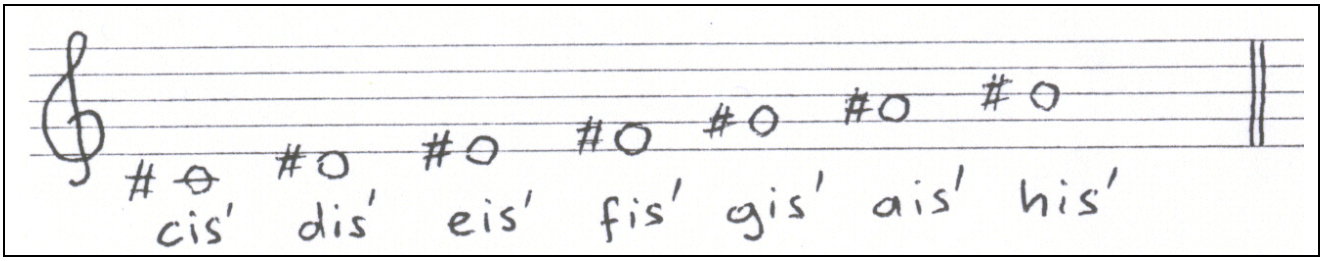


Note:
The notes `C,D,E,F,G,A,H` are named `C,D,E,F,G,A,B` in other countries.
In this document we stick to the German names.

Syntax used in our programming example:

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

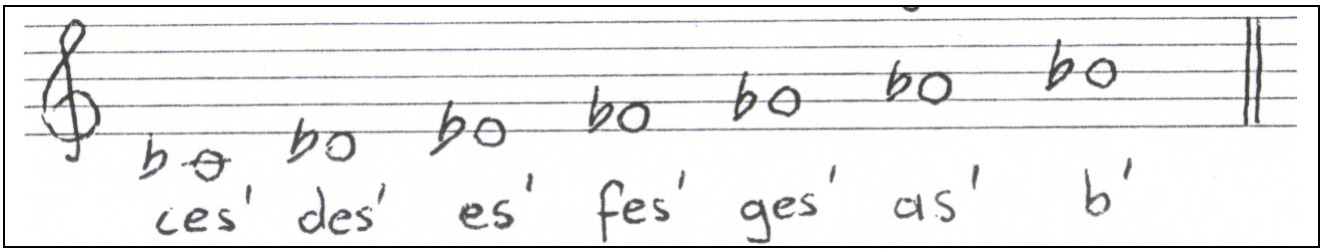
Real Music:



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:

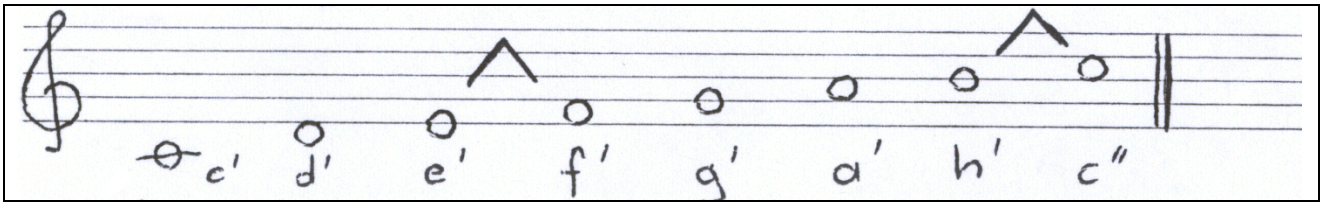
Definition:
In music, an octave is the interval between one musical note and another with half or double its frequency.

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 …).

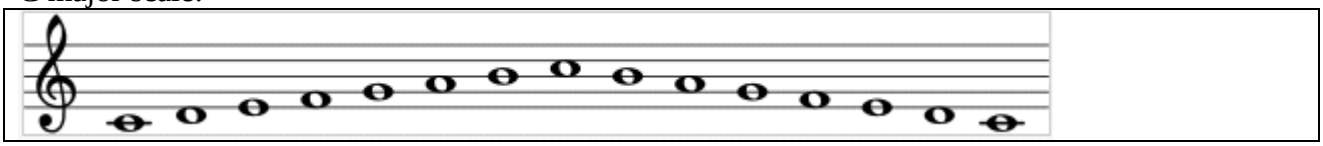
Syntax used in our programming example:

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

Real Music:

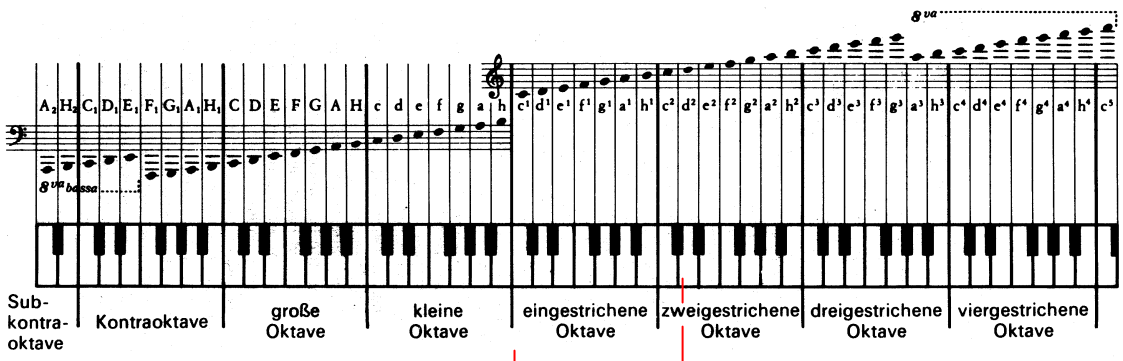


C major scale:

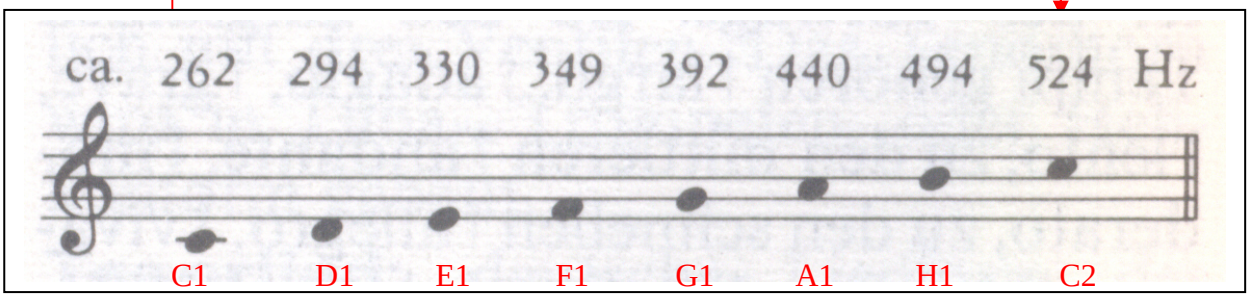


3.) Appendix: CCU6 use to create note length and note frequency

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



*1

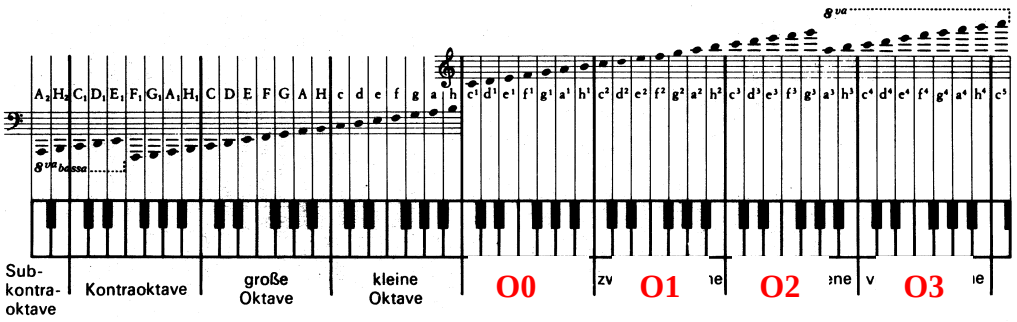


*2

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>
*/
```

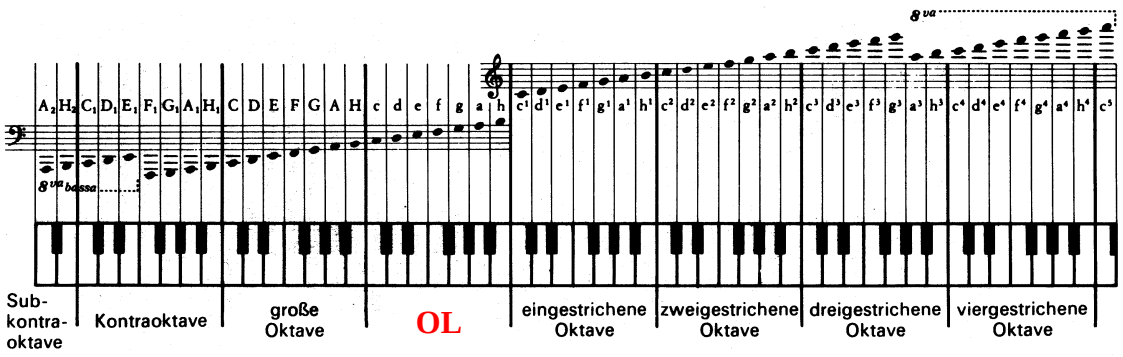
So we get the following values shown in the table below
[Note: Timer 13 resolution = $1/(f_{clk}/4) = 1/(66MHz/4) = 60{,}606$ ns]:

		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 / (T13-period-value x T13-resolution) = 1 / (37500 * 60,606 ns) = 440 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>
*/
```

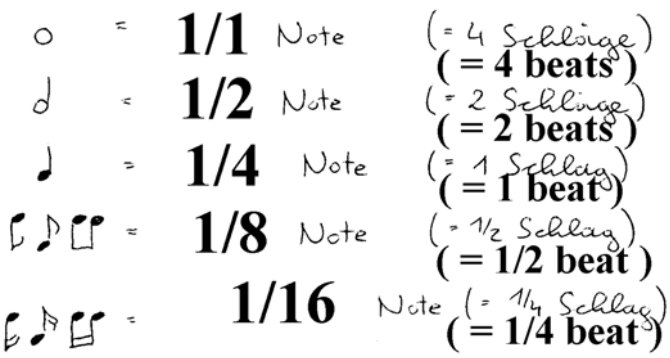
So we get the following values shown in the table below
[Note: Timer 13 resolution = $1/(f_{clk}/8) = 1/(66MHz/8) = 121{,}21$ ns]:

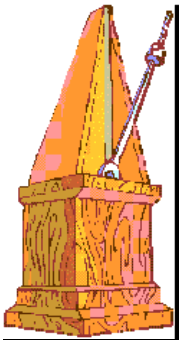
		Octave=OL T13 Prescaler=8	Octave=ONO0 T13 Prescaler=4
T13 period values	note	f [Hz]	f [Hz]
T13_values[0] = 62977	c	131	262
T13_values[1] = 59550	cis	139	
T13_values[2] = 56122	d	147	294
T13_values[3] = 53061	dis	156	
T13_values[4] = 50000	e	165	330
T13_values[5] = 47278	f	175	349
T13_values[6] = 44685	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:

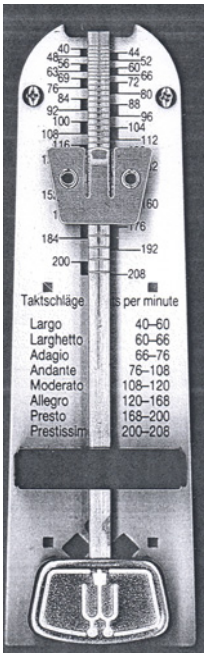
```
// T13, note frequency
CCU61_T13PR=T13_values[note]/octave; // note frequency
CCU61_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %
CCU61_vEnableShadowTransfer(CCU61_TIMER_13);
```

note – length (Timer 12)





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:

```
CCU61_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 = **2** [s]
	1/2 note = 2 beats = 2 * 0,5 = **1** [s]
	1/4 note = 1 beat = 1 * 0,5 = **0,5** [s]
	1/8 note = 1/2 beat = 1/2 * 0,5 = **0,25** [s]
	1/16 note = 1/4 beat = 1/4 * 0,5 = **0,125** [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):

			note length
T12 period values	note	note	[s]
16113 / 1 = 16113		1/1	2
16113 / 2 = 8057		1/2	1
16113 / 4 = 4028		1/4	0,5
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;
```

```
CCU61_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.";

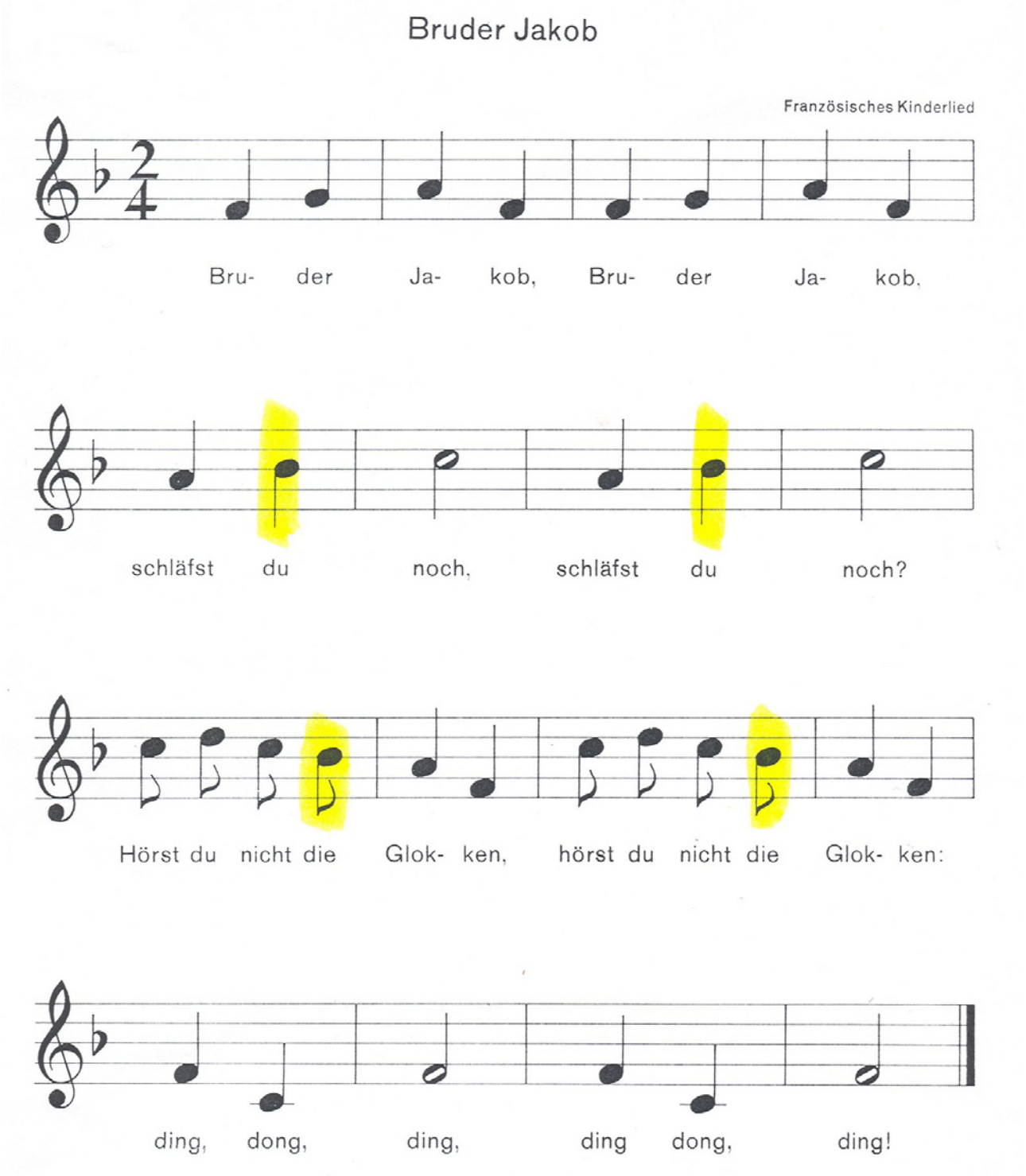
Note:
Thanks to Christian Perschl (www.perschl.at).
The songstring above was written down by Christian while humming the melody.

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

4.3.) Song c: Bruder Jakob:

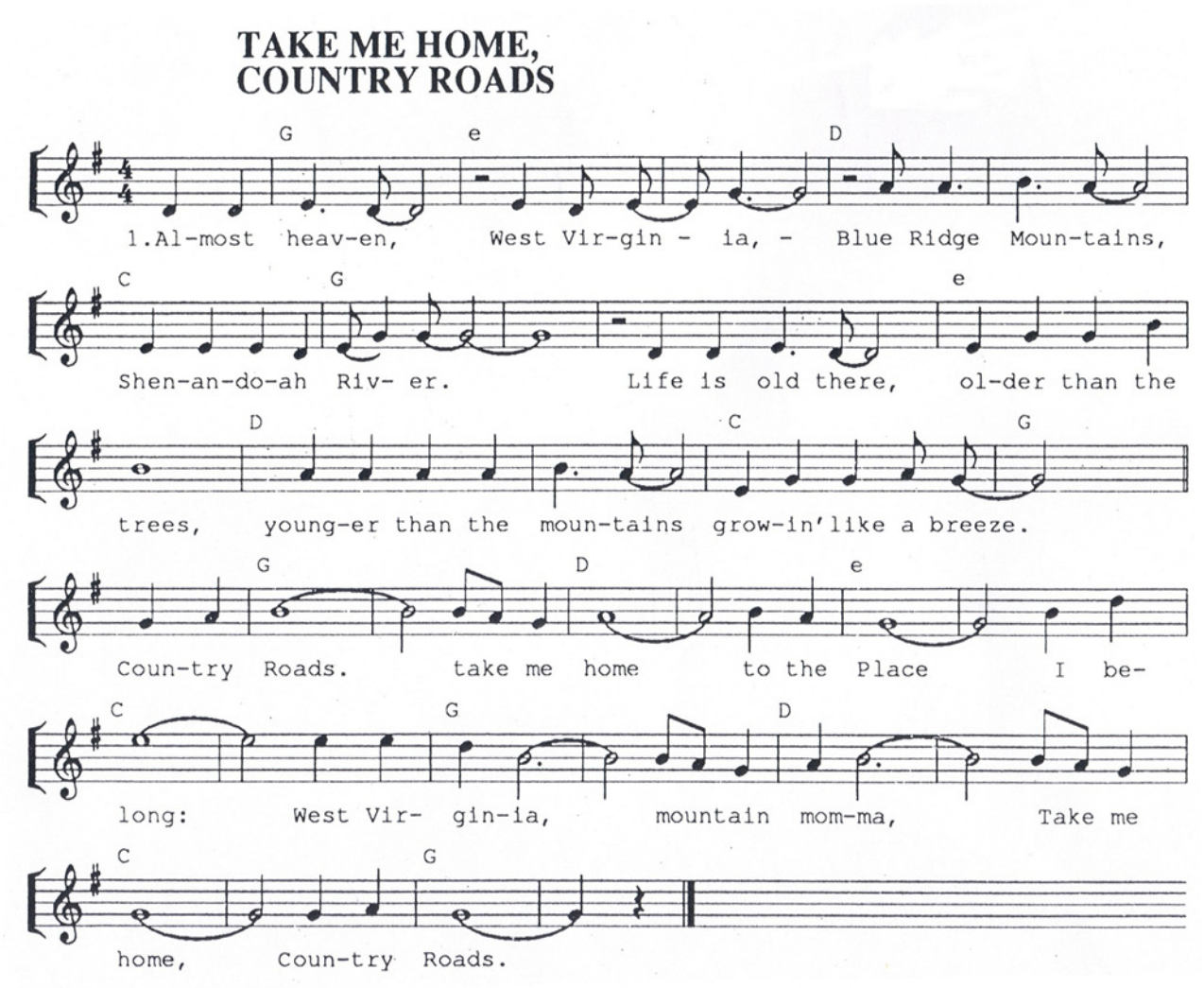


// Bruder Jakob (song c):
code unsigned char songc[]="T120O0L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0L8H-L4AFO1L8CDCO0L8H-L4AFFCL2FL4FCL2F";

4.4.) Song d: Happy birthday:

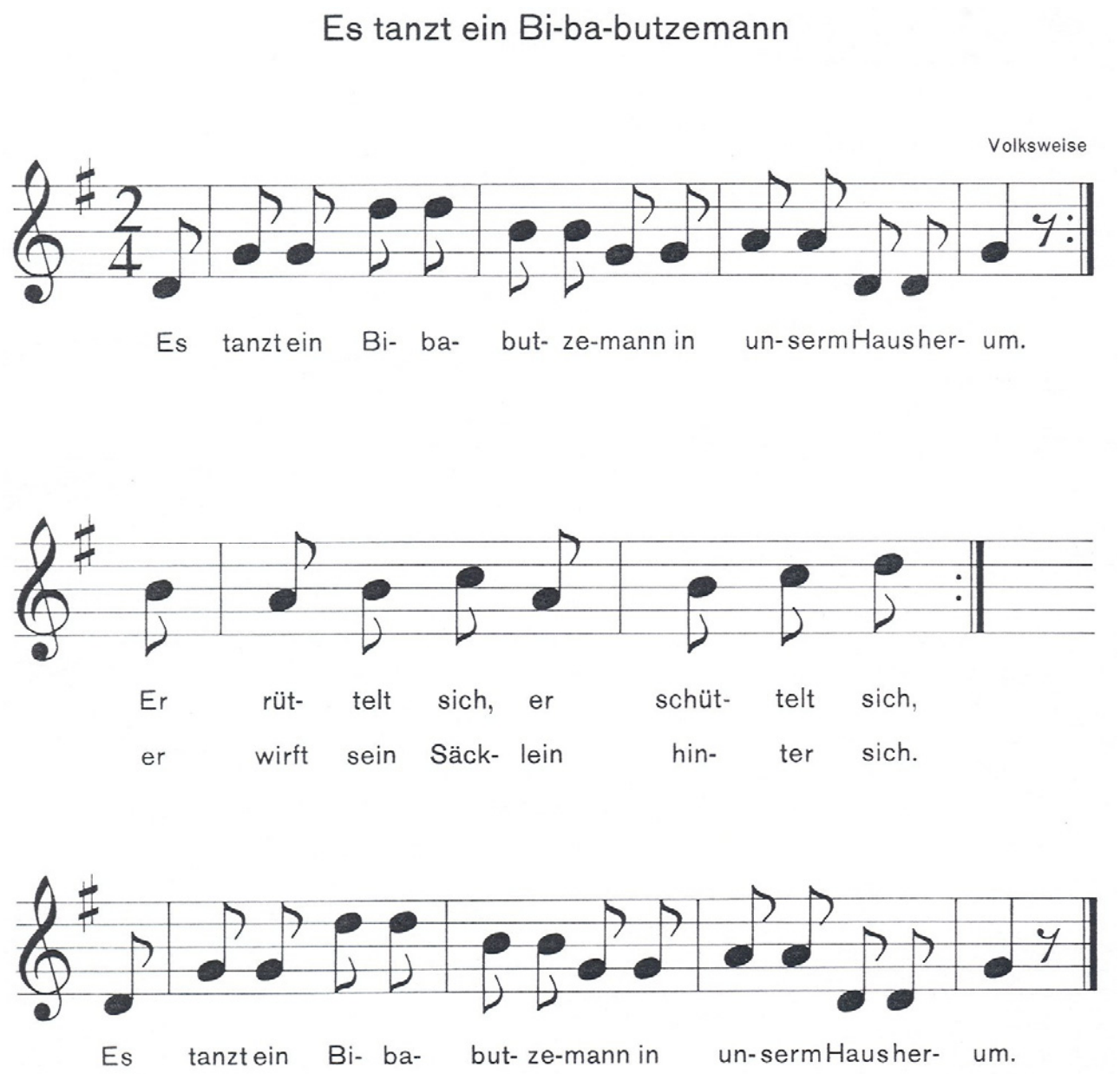

```
// Happy birthday (song d):
code unsigned char
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";
```

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



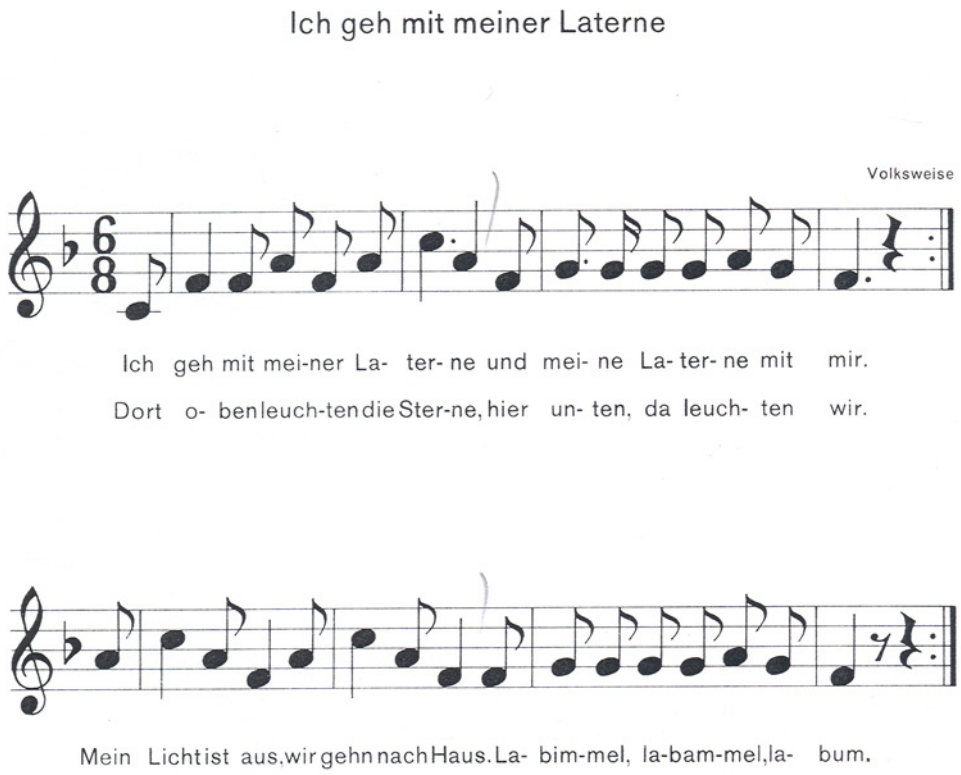

```
// Take Me Home, Country Roads (song e):
code unsigned char
songe[]="T199O0L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EEEDL8EL4GL1GP
1L4DDE.L2D.L4EGGHL1HL4AAAAH.L2A.L4EGGAL2G.L4GAL1HL8HAL4GL1AL4HAL1G
L4HO1L4DL1EL4EEDO0L1HL8HAGAL1HL8HAL4GL1GL4GAL1G";
```

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



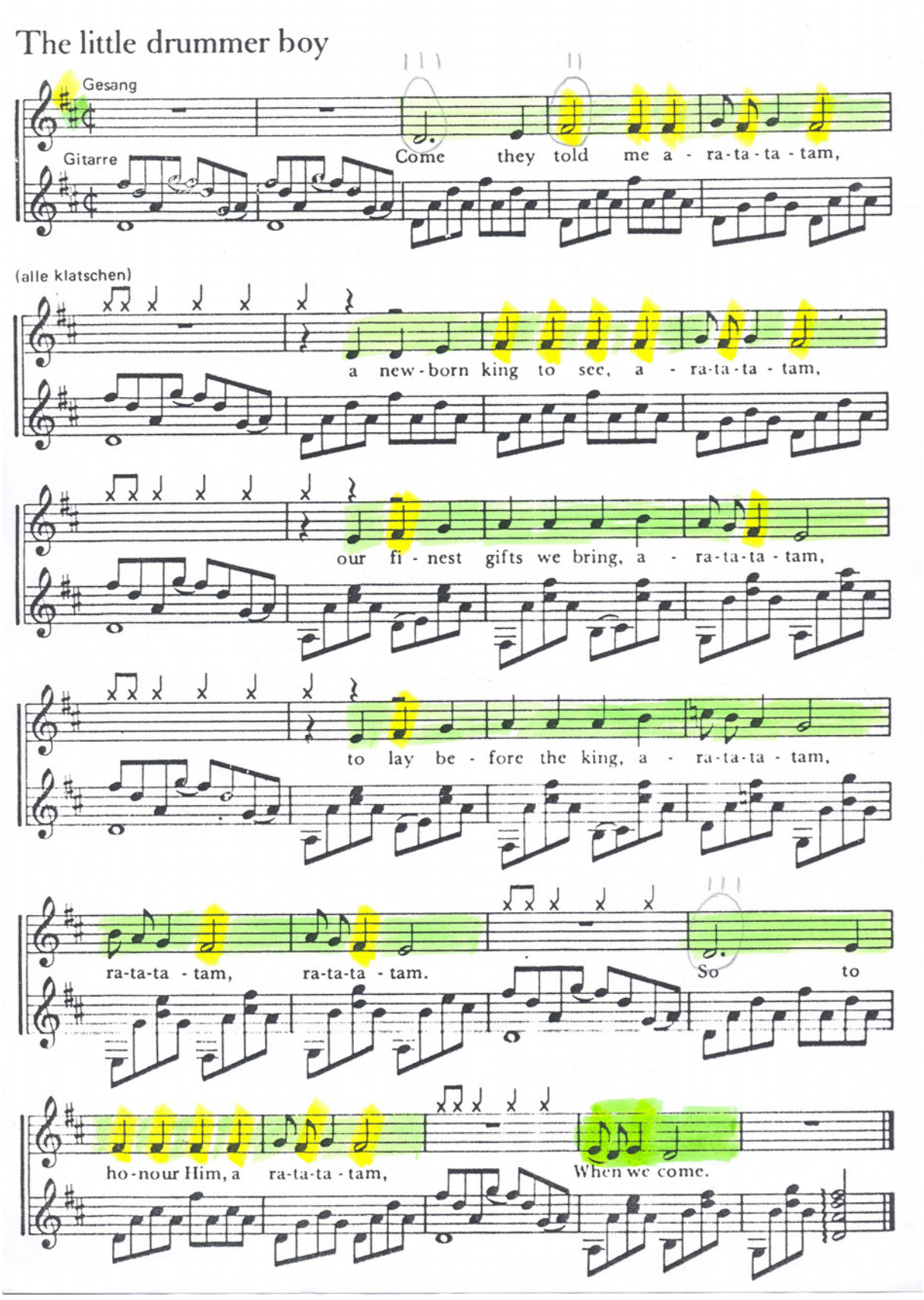

```
// Es tanzt ein Bi-ba-butzemann (song f):
code unsigned char
songf[]="T199O0L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";
```

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



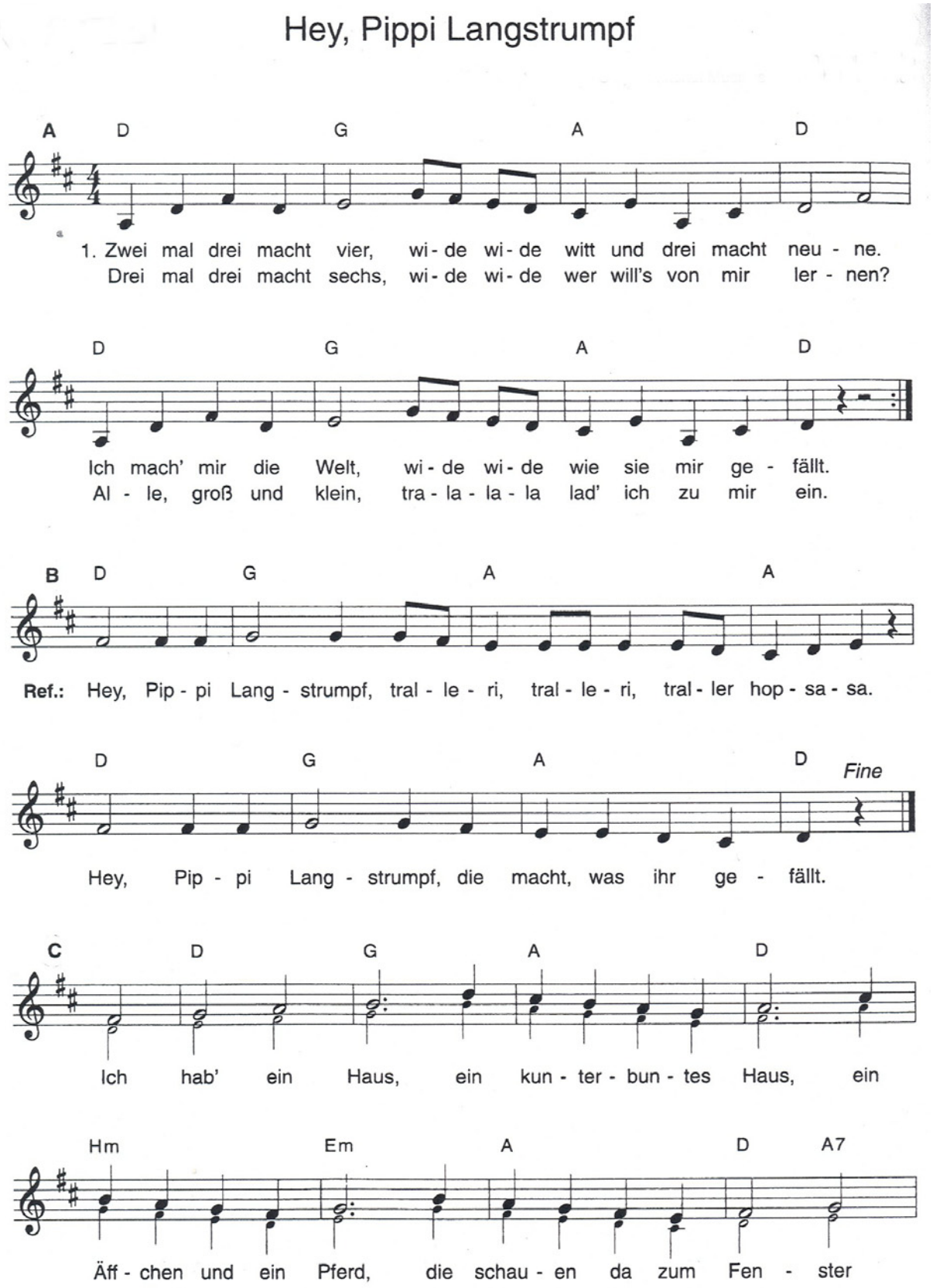

```
// Ich geh mit meiner Laterne (song g):
code unsigned char
songg[]="T120O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8CL4FL8FA
FAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4F
L8FGGGGAGL4FP4.O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4FL8FGGGGAGL4FP4.";
```

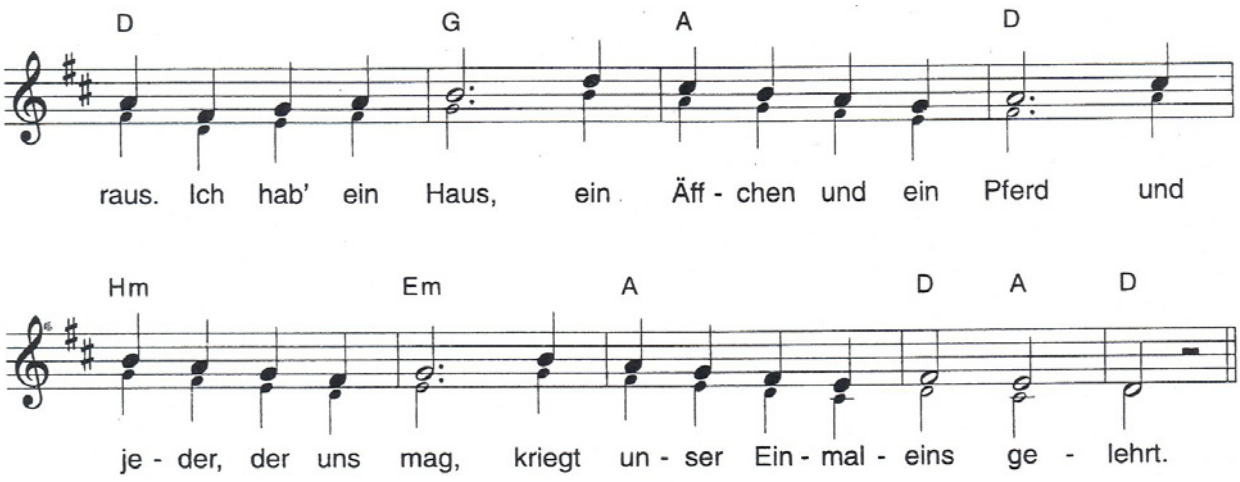
4.8.) Song h: The little drummer boy:



```
// The little drummer boy (song h):
code unsigned char
songh[]="T120P2O0L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+L4F+L4F+L8G
F+L4GL2F+P2L4EF+L4GAAAHL8AGL4F+L2EP2L4EF+L4GAAAHO1L8CO0L8HL4AL2GL8
HAL4GL2F+L8AGL4F+L2EP1L2D.L4EL4F+F+F+F+L8GF+L4GL2F+P1L8EDL4EL2D";
```

4.9.) Song i: Hey, Pippi Langstrumpf:

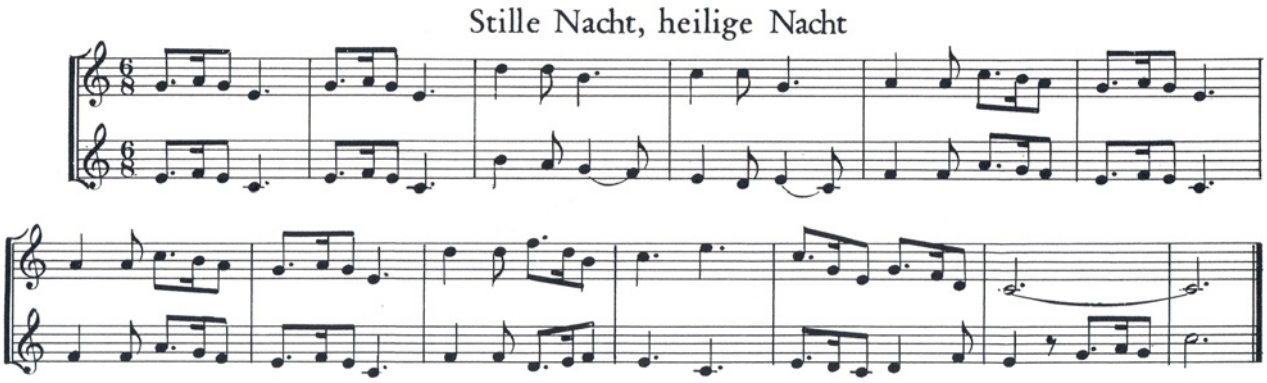




2. Zwei mal drei macht vier, wide wide witt und drei macht neune,
wir machen uns die Welt, wide wide wie sie uns gefällt.
Drei mal drei macht sechs, wide wide wer will's von uns lernen?
Alle, groß und klein, tralala, lad' ich zu uns ein.
Ref.: Hey, Pippi Langstrumpf,...

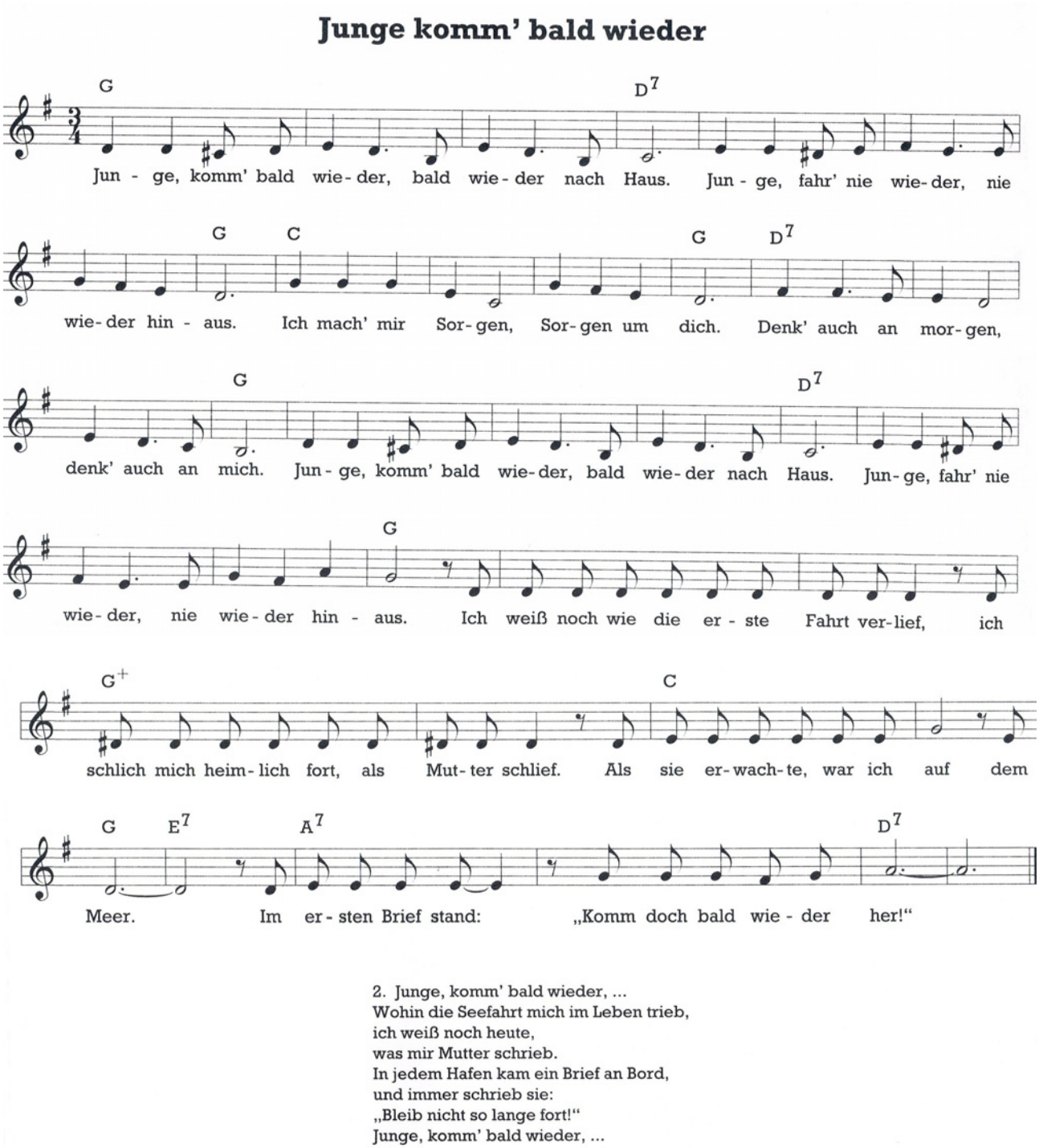
// Hey, Pippi Langstrumpf (song i):
code unsigned char
songi[]="T180OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLAONO0L4C+L2DF+OLL4AONO
0L4DF+DL2EL8GF+EDL4C+EOLL4AONO0L4C+DP4P2OLL4AONO0L4DF+DL2EL8GF+ED
L4C+EOLAONO0L4C+L2DF+OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLL4AONO0L4C+
DP4P2O0L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+
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 (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.";
```

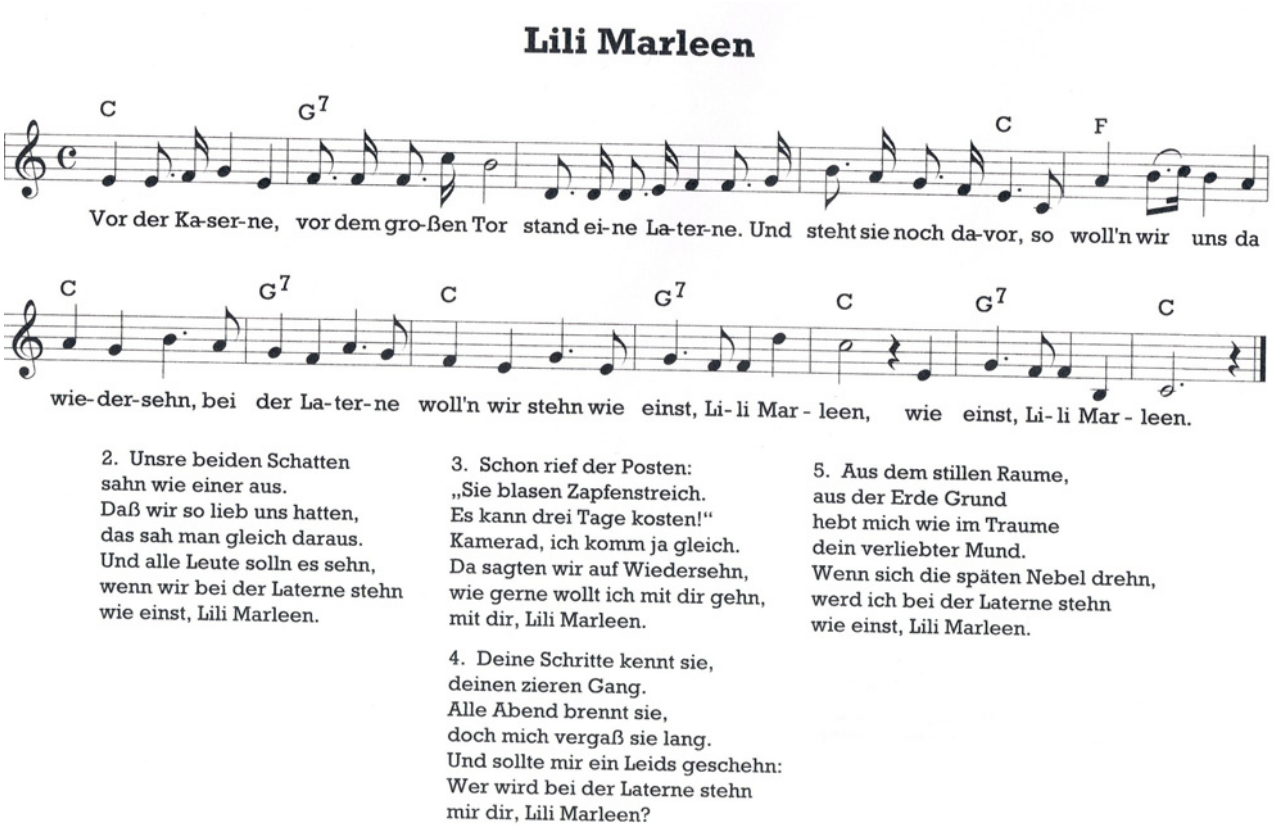
4.11.) Song k: Junge komm bald wieder:



// Junge komm bald wieder (song k):

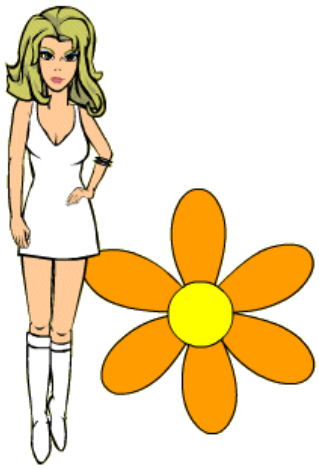
code unsigned char songk[]="T120O0L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4EEL8D+L8EL4F+L4E.L8EL4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+.L8EL4EL2DL4EL4D.L8COLL2H.ONO0L4DDL8C+L8DL4EL4D.OLL8HONO0L4EL4D.OLL8HONO0L2C.L4EEL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8DDDDDDDDDDL4DP8L8DL8D+L8DDDDDL8D+L8DL4DP8L8DL8EEEEEEL2GP8L8EL1DP8L8DL8EEEL4E.P8L8GGGF+L8GL1A.";

4.12.) Song l: Lili Marleen:



// Lili Marleen (song l):
code unsigned char
songl[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HONO0L2C.";

4.13.) Another song: Lady Bird:



// Lady Bird:

```
"T150ONO0L4F+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, XC16x, XE16x and XC2xxx) 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.

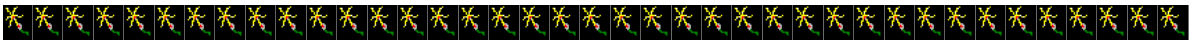
This should give you a good opportunity to get in contact with another Infineon microcontroller family or tool-chain!

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.

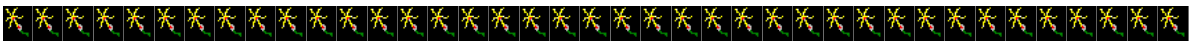
6.) Feedback (XE164 Playing Music, Keil tools): Your opinion, suggestions and/or criticisms



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
FAX: +43 (0) 4242 3020 5783



Your suggestions:

http://www.infineon.com

Published by Infineon Technologies AG