AP08082

XC878

Playing music using the CAPCOM6 module.
Using DAvE (Code Generator) and
DAvE Bench (Open Platform for Free Tools:
IDE, Compiler, Debugger, Utility Tools)

Microcontrollers

Never stop thinking

Edition 2010-06-21

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:	2010-05	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 8 and page 9.

Introduction:

This “Application Note / Appnote” is a Hands On Training / Cookery Book / step-by-step book.
It will help inexperienced users to get familiar with the CAPCOM 6 module.
This step-by-step book is a follow-up to AP08081!

The purpose of this document is to gain know-how of the possibilities offered by the CAPCOM 6 / CCU6 module for PWM generation.

Note:
The style used in this document focuses on working through this material as fast and easily as possible. That 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.

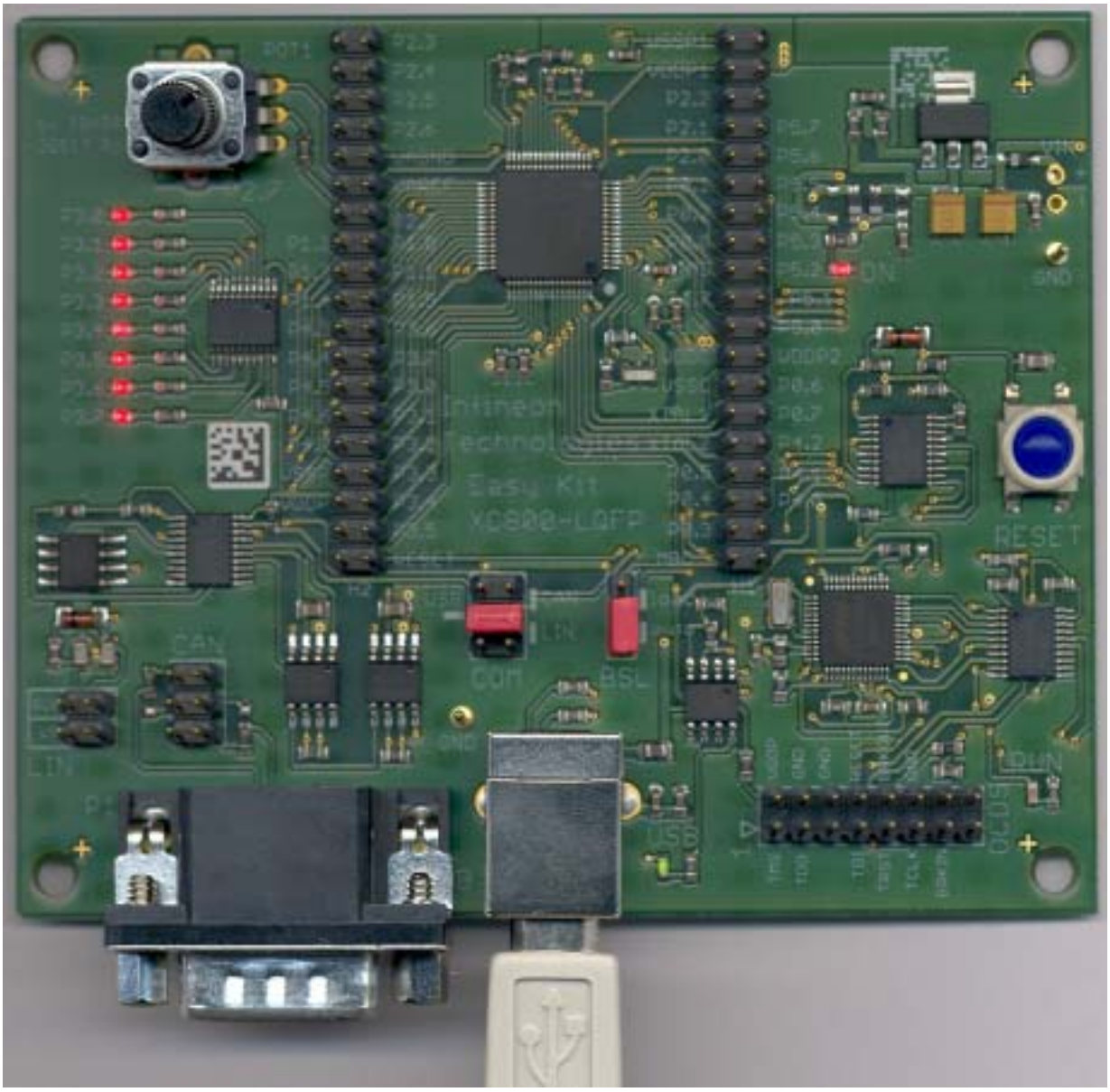
Have fun and enjoy the CAPCOM 6 module!

Note:
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.

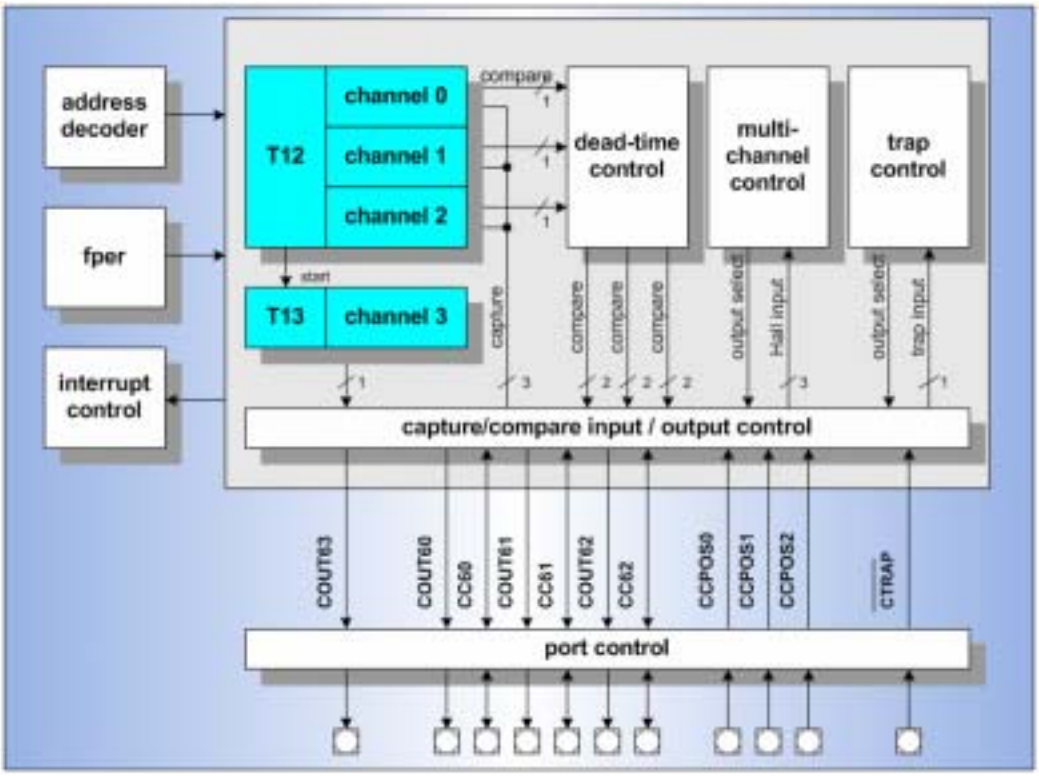
Note:
Additionally, there is a step-by-step book (AP16109) focusing on BLDC-Motors available, which can be used for all 8/16 and 32 bit microcontrollers equipped with the CAPCOM 6 module.
To get the most out of the CAPCOM 6 module this additional Cookery Book is the icing on the cake of all available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

Programming Example

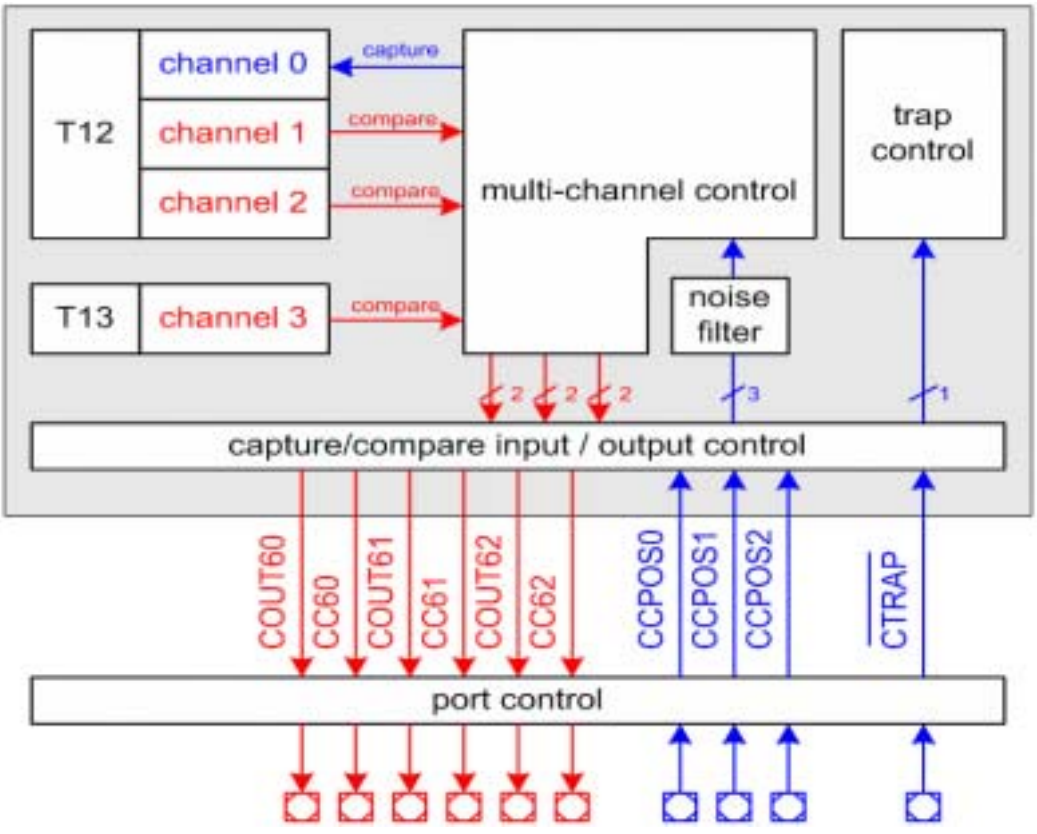
XC878 Easy Kit



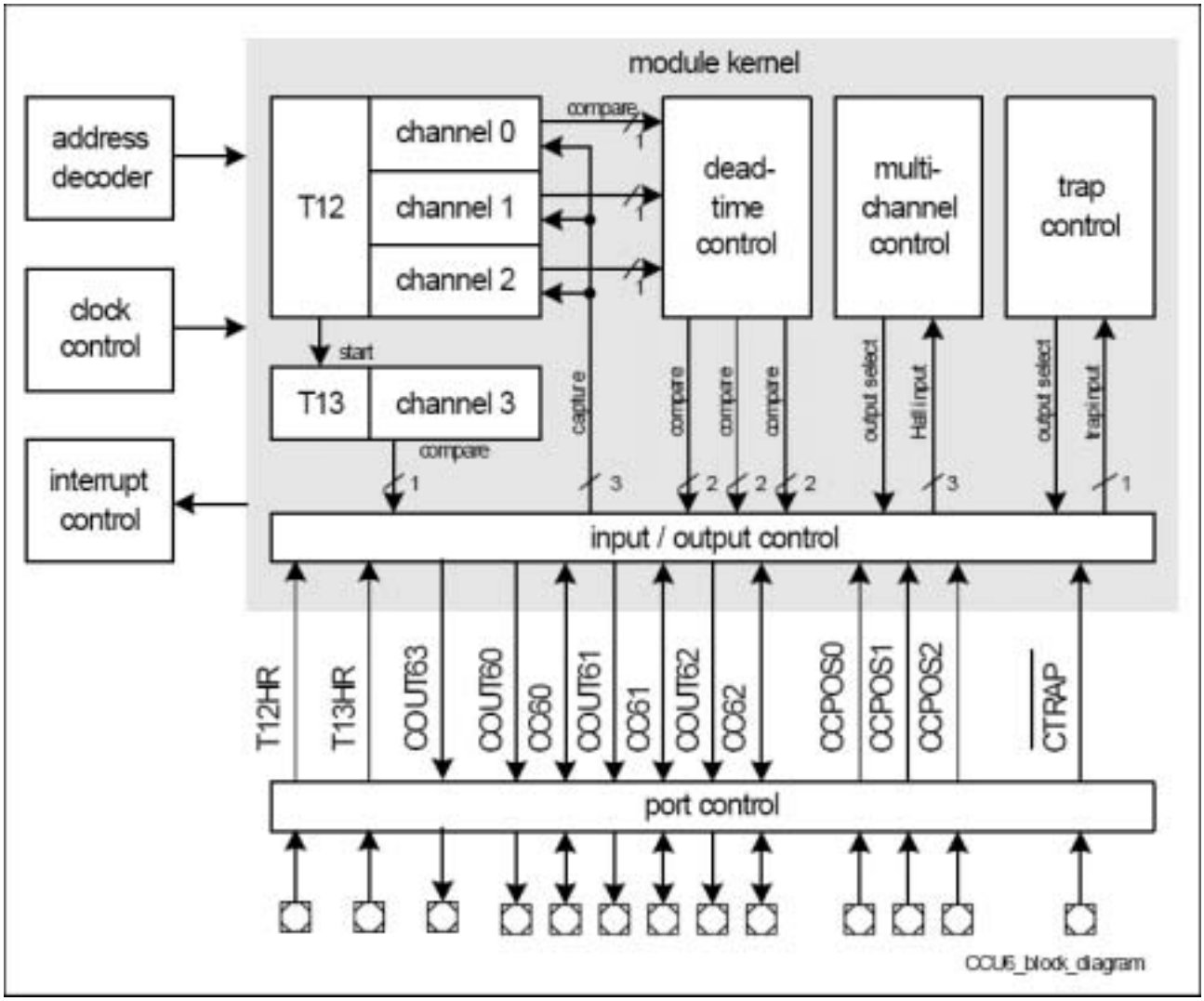
CAPCOM 6 Block Diagram – general use (Source: Product Marketing)



CAPCOM 6 Block Diagram – BLDC use (Source: Product Marketing)



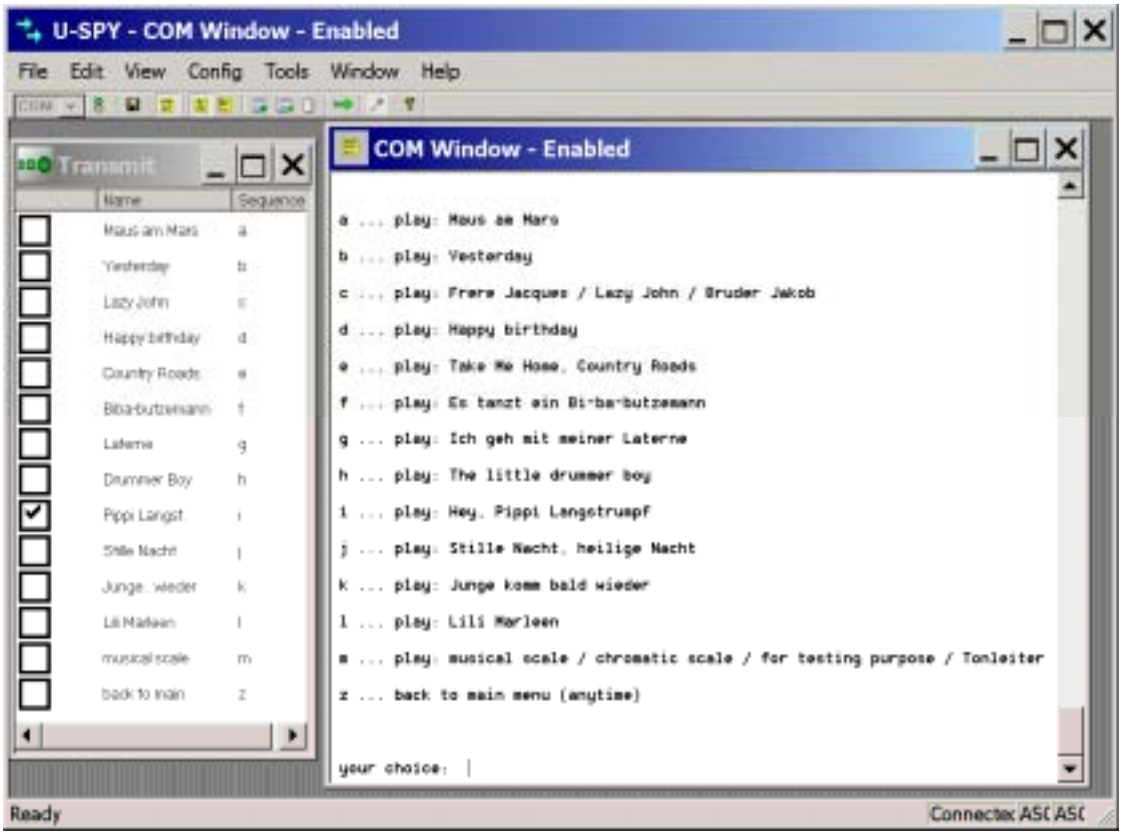
CAPCOM 6 Block Diagram (Source: User's Manual)



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.)	**Do the XC878 Cookery Book**
2.)	**Playing music**
2.1)	**Configuring and Reconfiguring the DAvE Project Settings**
2.2)	**Open the DAvE Bench project and insert code**
2.3)	**See and hear the result; using the Debugger**

Appendix:

Chapter/ Step	*** Recipes ***
3.)	Appendix: about music (note length, note frequency)
4.)	Appendix: CAPCOM6 / CCU6 use to create note length and note frequency
5.)	Appendix: songs used

Feedback:

6.)	Thanks To
7.)	Feedback

1.) Do the XC878 Cookery Book:

Note:
It is necessary to follow all instructions in the XC878 Cookery Book (AP08081) 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 AP08081) with the requirements for PWM generation.

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

Playing music

Note:

Port_3 pins used by our PWM module:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	100 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

Port_3 pins used as GPIO:

Port Lines	Function	Comment
P3.1	Show start of next note	Toggled via Software
P3.6	use: „program running signal“	Toggled via Timer_0 ISR

Port 3:

Pin		CCU6-Channel	Modulated by	Purpose	
P3.0	CC60	CCU6-Channel-0	Modulated by T12	show note length duty cycle = 100 % only for measurement	
P3.1		---	Software	start of next note	
P3.2	CC61	CCU6-Channel-1	Modulated by T12	show note length duty cycle = 100 % only for measurement	
P3.3		---	---	---	
P3.4	CC62	CCU6-Channel-2	Modulated by T12 + T13	Music Output: note length modulated by note frequency	
P3.5		---	---	---	
P3.6		---	Software	running signal	
P3.7	CC63	CCU6-Channel-3	Modulated by T13	note frequency duty cycle = 50 % only for measurement	

2.1) Configuring and Reconfiguring the DAvE Project Settings:

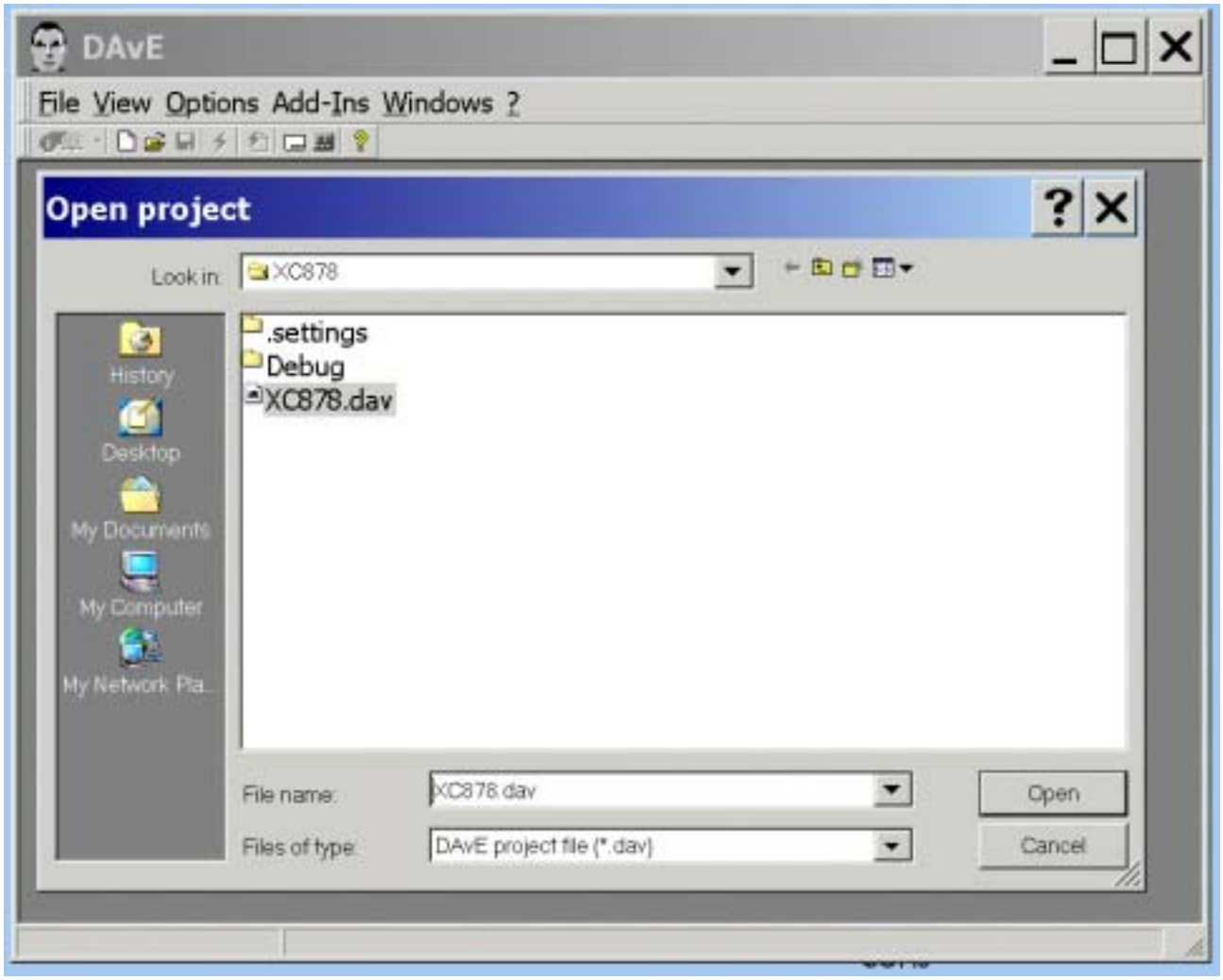
Let's Get Started:

Configuring and Reconfiguring the DAvE Project Settings:

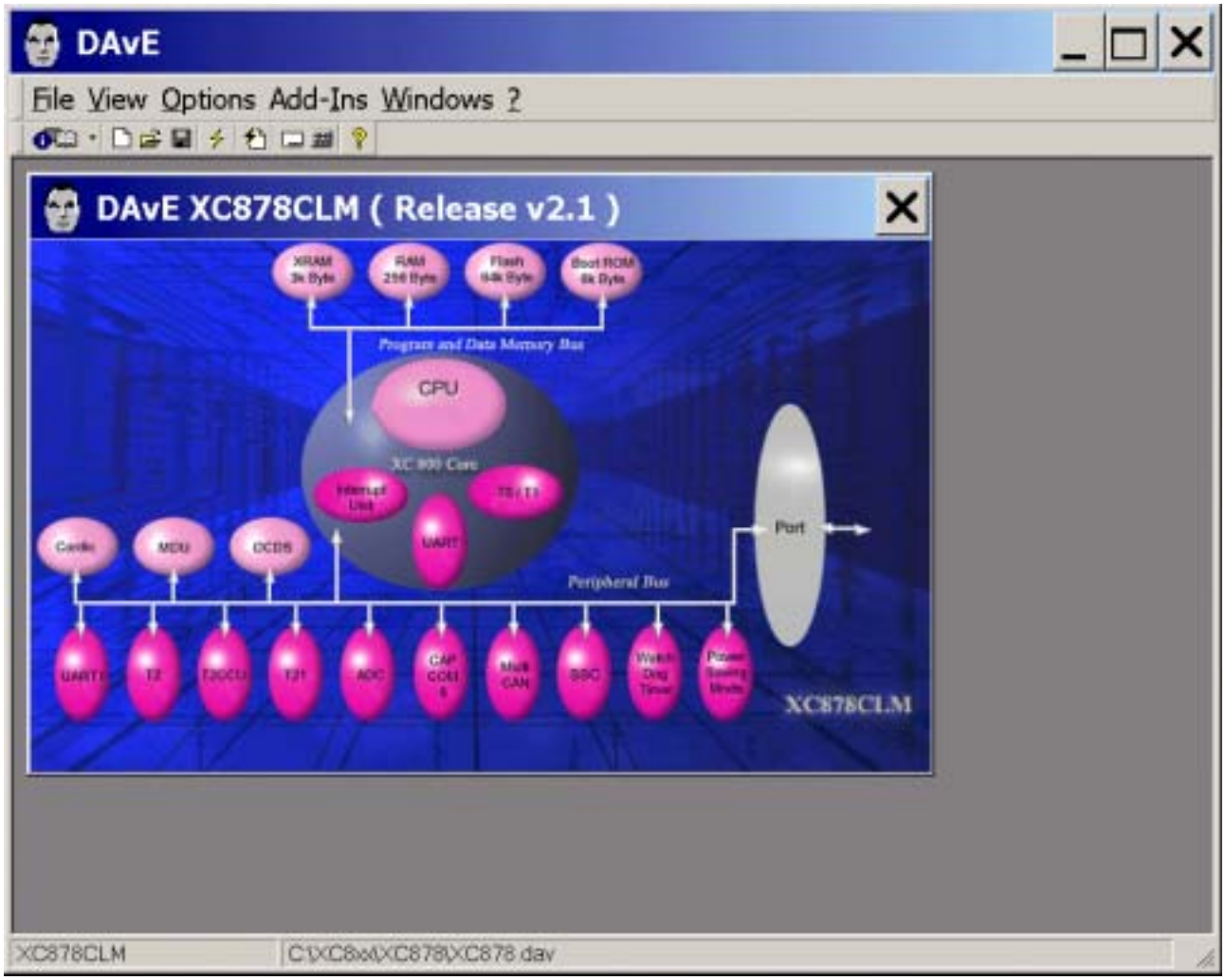
Start the program generator DAvE and open your XC878.dav DAvE project:

View - Project Window (Closes the Project Window)
View - Command Window (Closes the Command Window)

File
Open
Location: C:\XC8xx\XC878
Filename: XC878.dav



Click Open


DAvE
File View Options Add-Ins Windows ?
DAvE XC878CLM (Release v2.1)
CPU
Port
XC878CLM
XC878CLM
C:\XC8xx\XC878\XC878.dav

Changing the Memory Model:

Note
(Release Notes; DAvE Bench for XC800; SDCC Tool chain for XC800; List of Limitations and Deviations; List of known Issues):

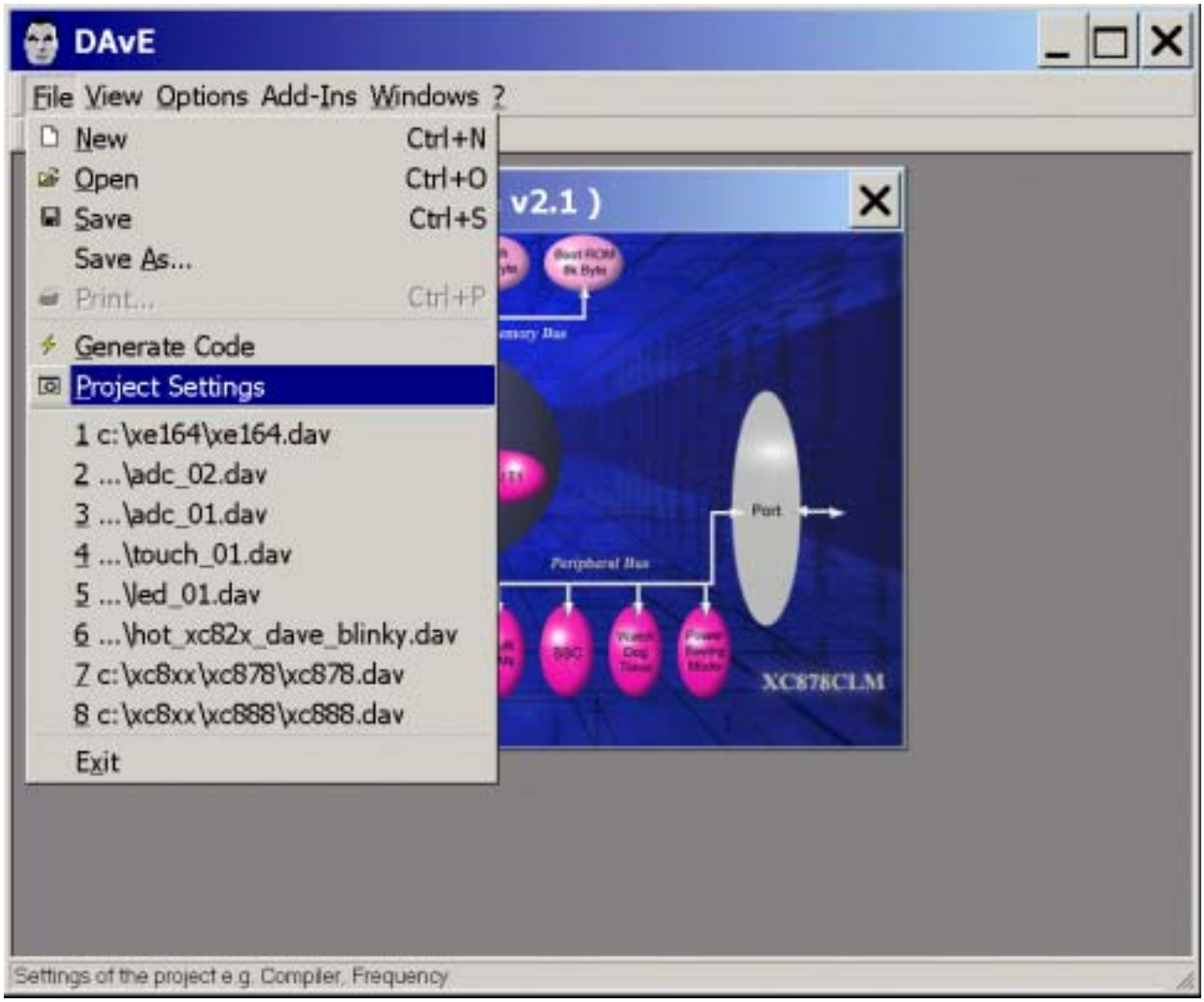
Floats are enabled in large memory model only due to code size limitation. So, print routines like printf(), sprintf() etc., will output <NO_FLOAT> when floats are used with these routines in small and medium memory models.

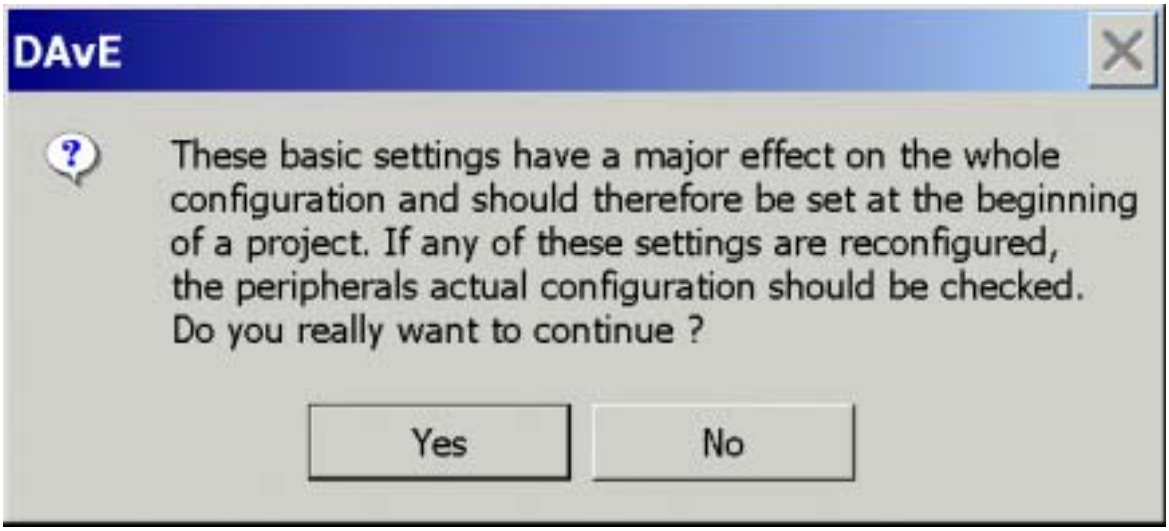
➔ Because we are going to use float variables in this programming example we have to change the Memory Model from small to large.

Printf does not work on XC878 16FF

➔ Therefore we are going to use other print formats like printf_small() and printf_fast_f().

File – Project Settings





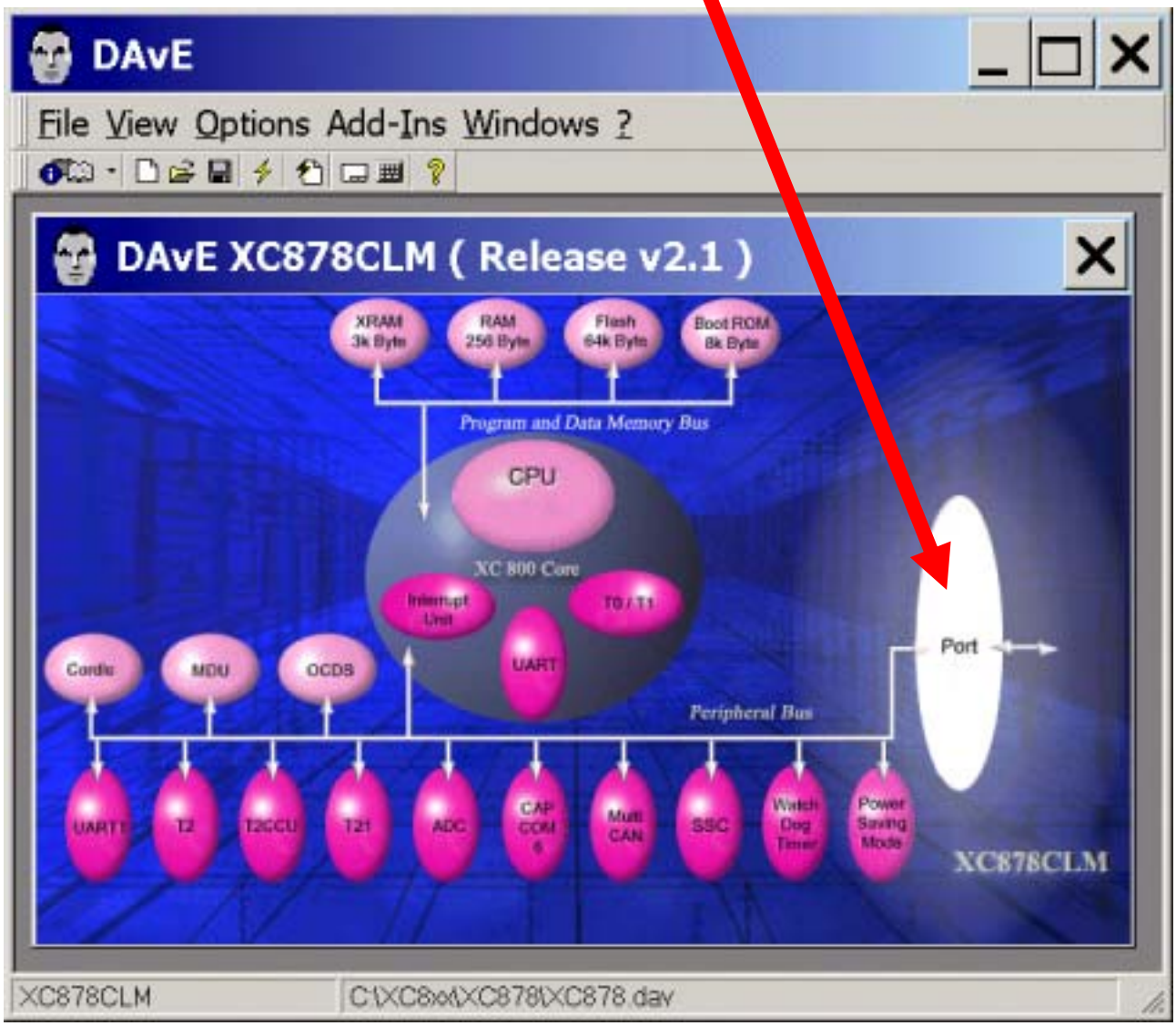
Click Yes

General: Compiler Settings: Memory Model: select Large

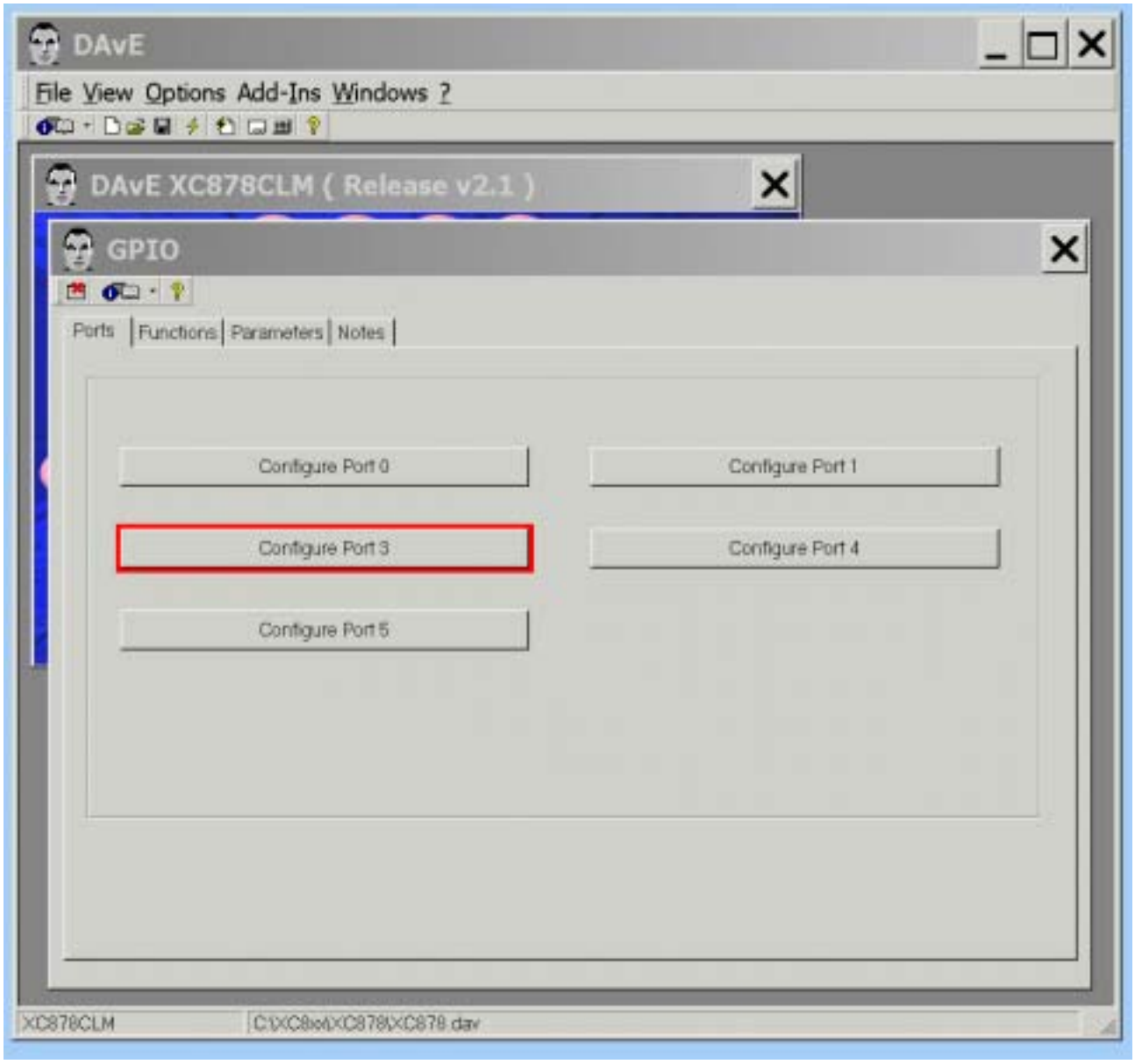
Exit this dialog now by clicking  the close button.

Reconfiguration of Port 3:

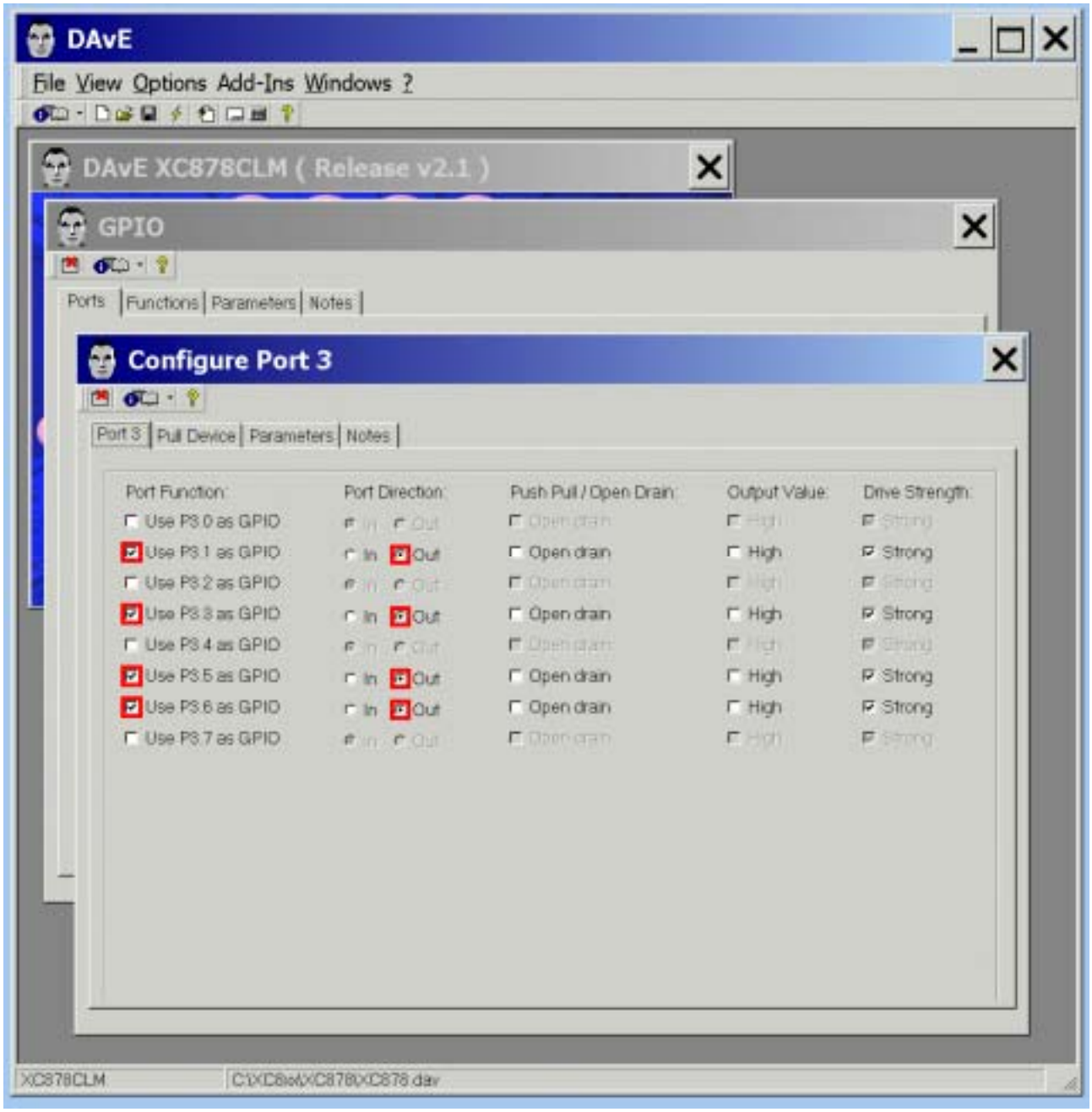
The (re)configuration window/dialog can be opened by clicking the specific block/module (Port).



Ports: click “Configure Port 3”



Port 3: Port Function: untick to unselect ☐ Use P3.0 to free GPIO pin P3.0 for CCU6 use
Port 3: Port Function: untick to unselect ☐ Use P3.2 to free GPIO pin P3.2 for CCU6 use
Port 3: Port Function: untick to unselect ☐ Use P3.4 to free GPIO pin P3.4 for CCU6 use
Port 3: Port Function: untick to unselect ☐ Use P3.7 to free GPIO pin P3.7 for CCU6 use
Port 3: Port Function: tick/check ✓ Use P3.1 as general IO - Port Direction: click/check ⊙ Out
Port 3: Port Function: tick/check ✓ Use P3.3 as general IO - Port Direction: click/check ⊙ Out
Port 3: Port Function: tick/check ✓ Use P3.5 as general IO - Port Direction: click/check ⊙ Out
Port 3: Port Function: tick/check ✓ Use P3.6 as general IO - Port Direction: click/check ⊙ Out



Note:

Port_3 pins used by our PWM module:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	100 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

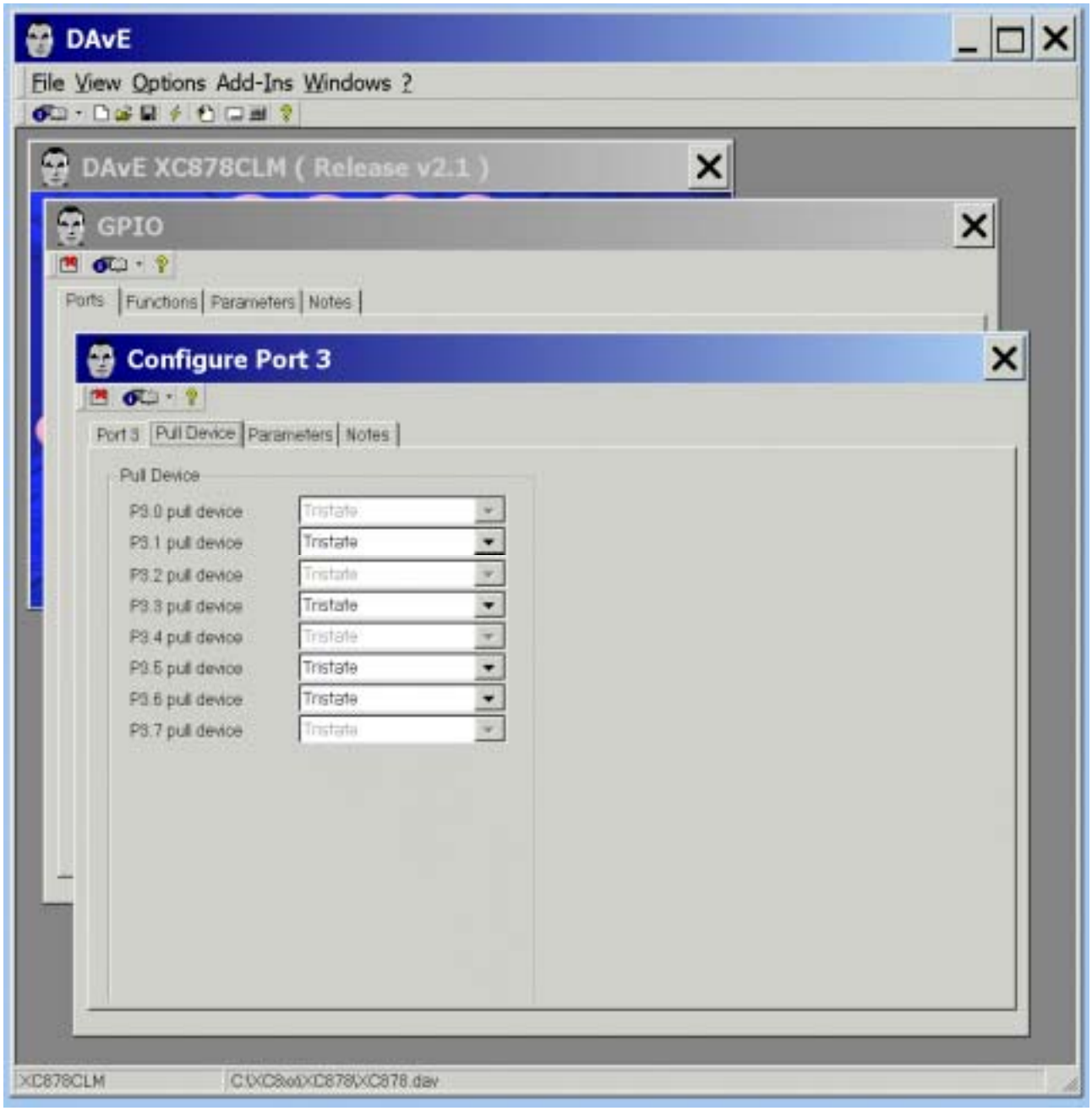
Port_3 pins used as GPIO:

Port Lines	Function	Comment
P3.1	Show start of next note	Toggled via Software
P3.6	use: „program running signal“	Toggled via Timer_0 ISR

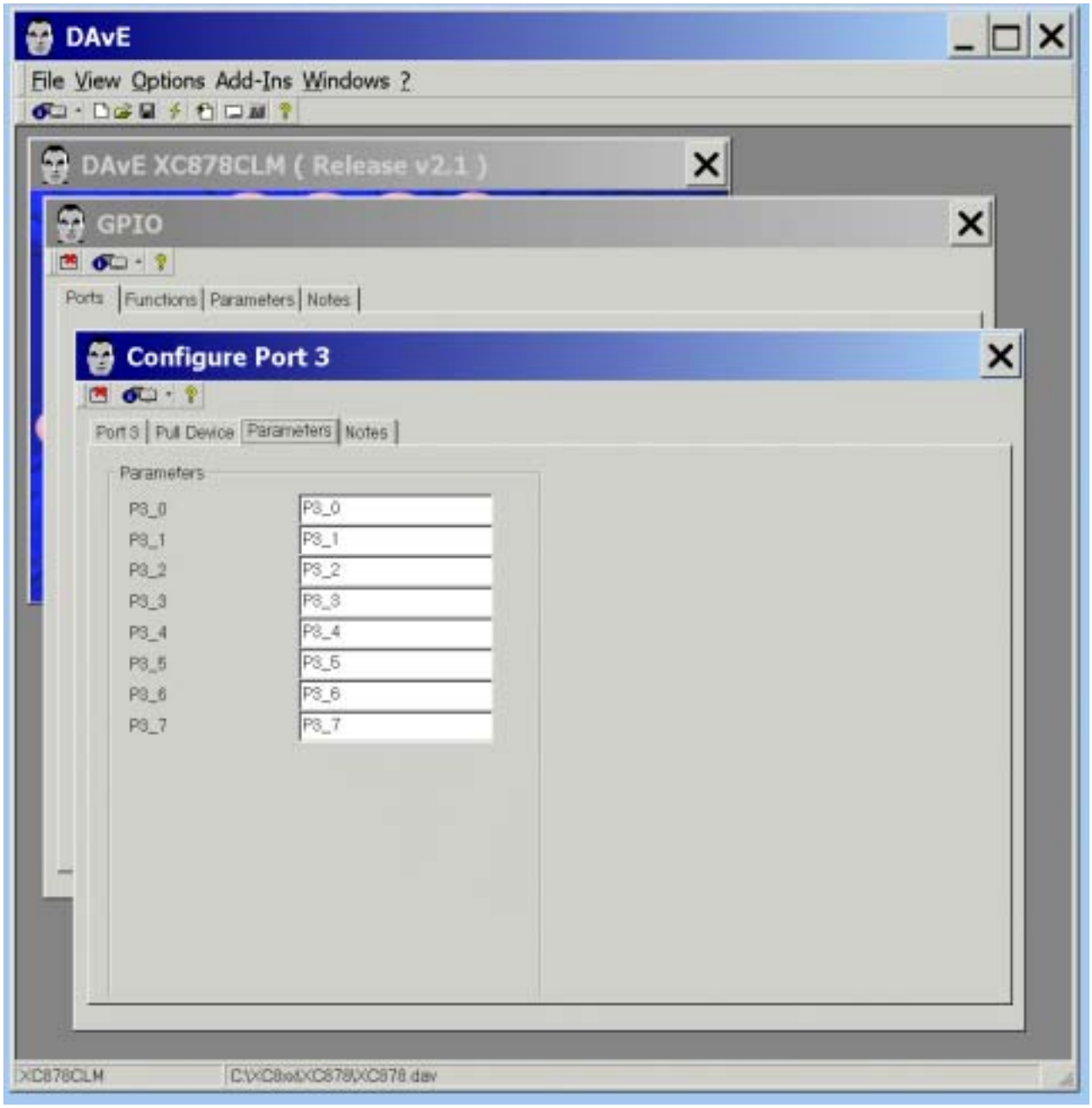
Port 3:

Pin		CCU6-Channel	Modulated by	Purpose	
P3.0	CC60	CCU6-Channel-0	Modulated by T12	show note length duty cycle = 100 % only for measurement	
P3.1		---	Software	start of next note	
P3.2	CC61	CCU6-Channel-1	Modulated by T12	show note length duty cycle = 100 % only for measurement	
P3.3		---	---	---	
P3.4	CC62	CCU6-Channel-2	Modulated by T12 + T13	Music Output: note length modulated by note frequency	
P3.5		---	---	---	
P3.6		---	Software	running signal	
P3.7	CC63	CCU6-Channel-3	Modulated by T13	note frequency duty cycle = 50 % only for measurement	

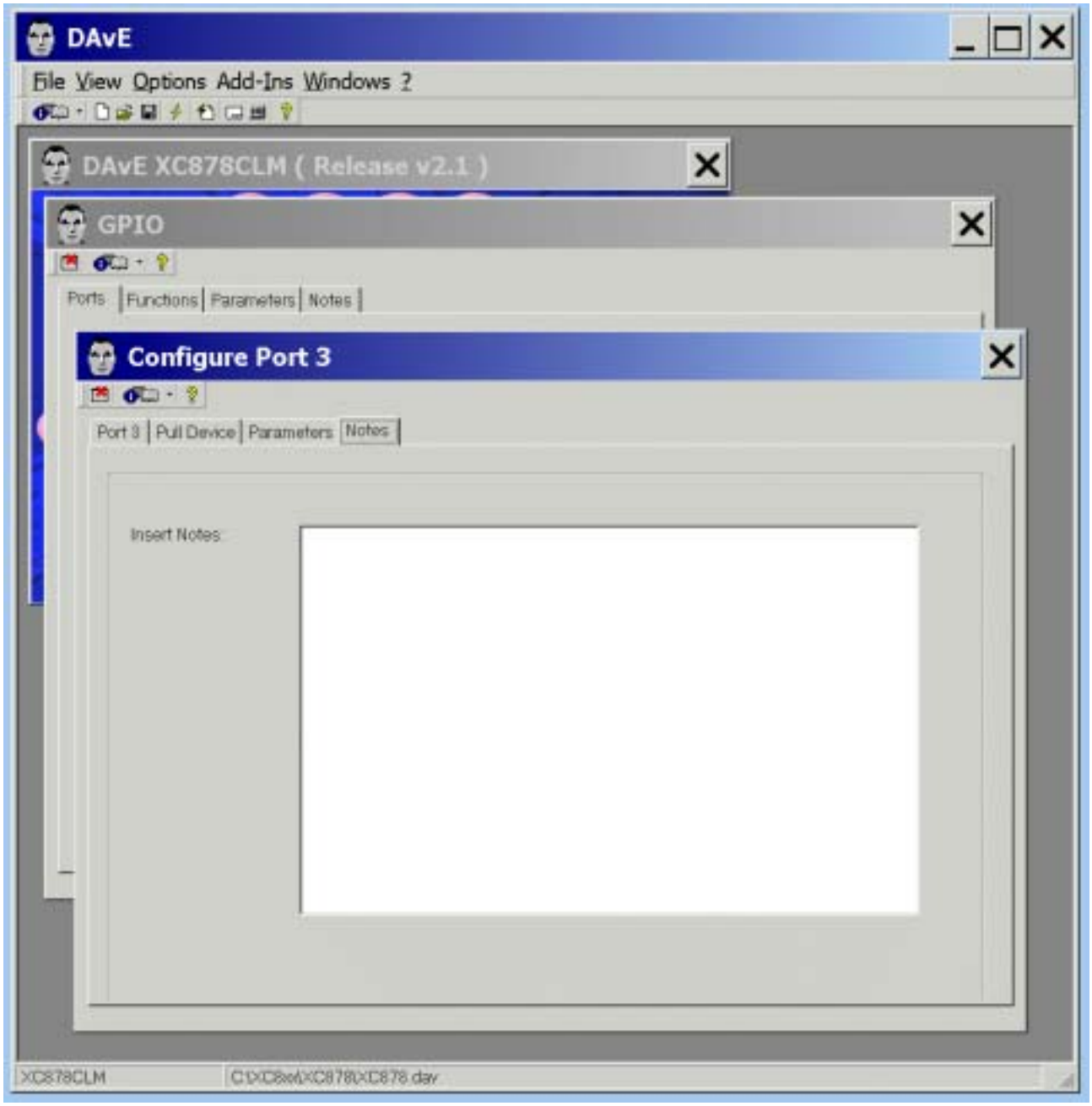
Pull Device: (do nothing)



Parameters: (do nothing)



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

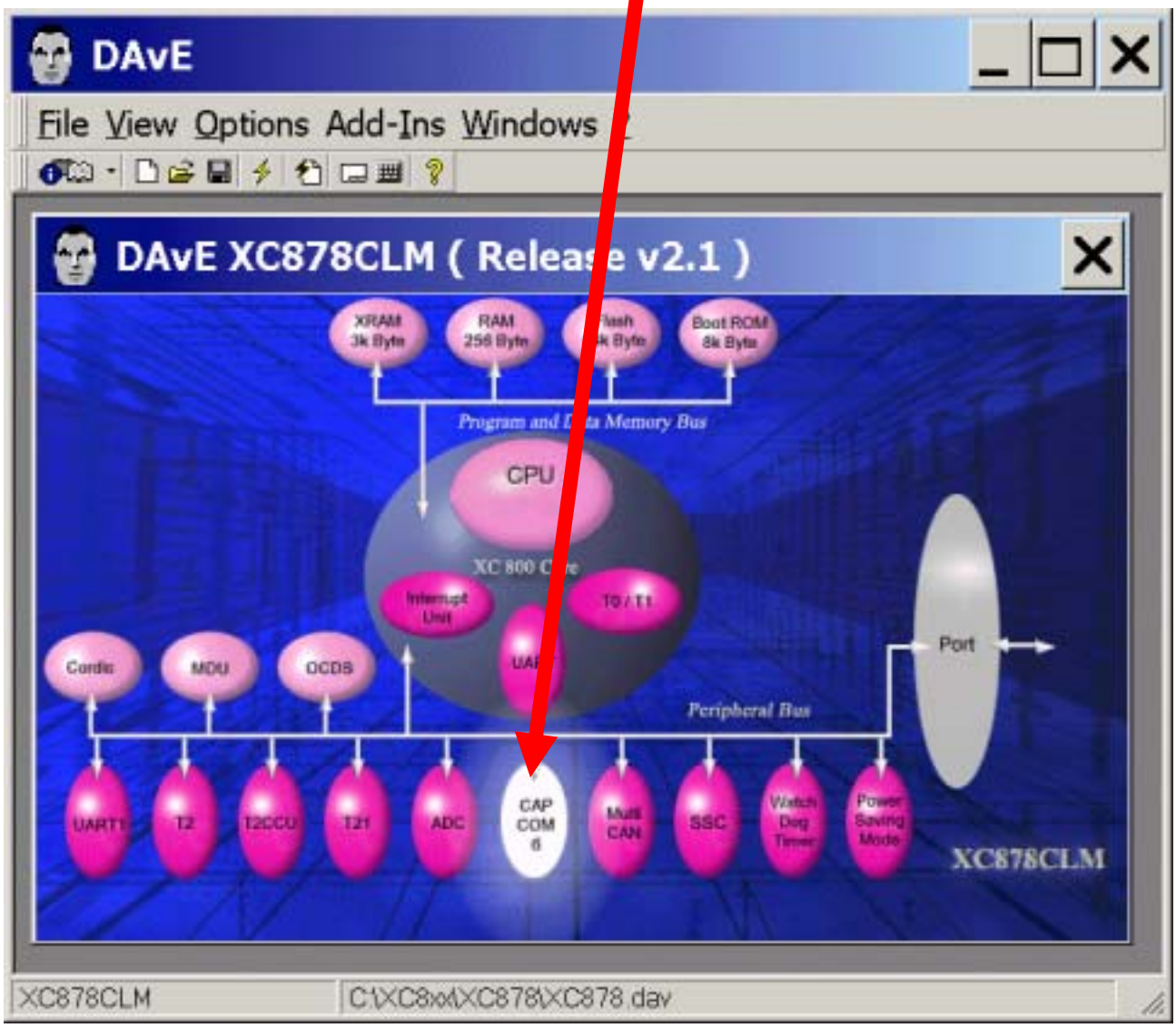


Exit this dialog now by clicking the close button.

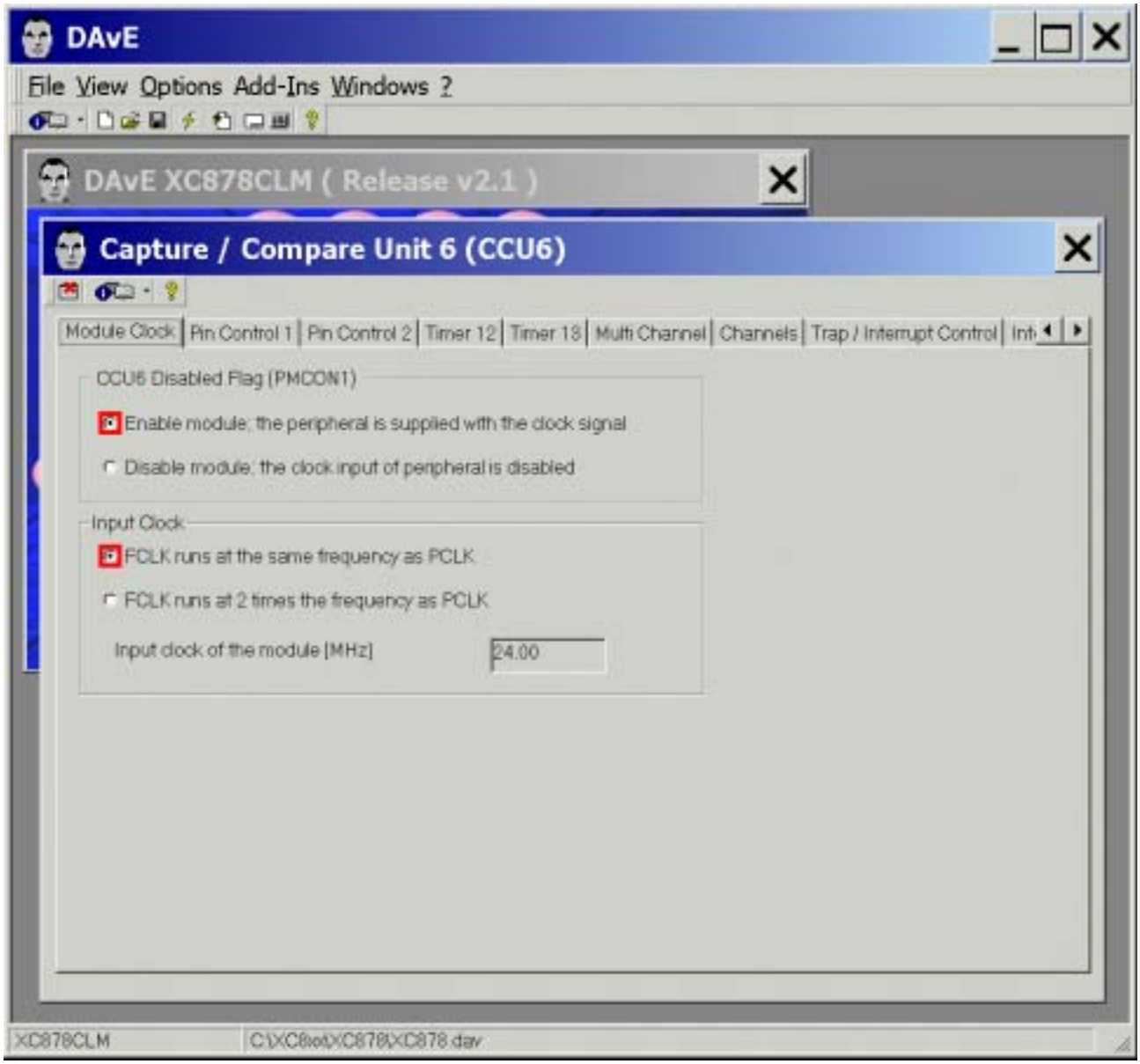
Exit this dialog now by clicking the close button.

Configuration of the CAPCOM 6 module:

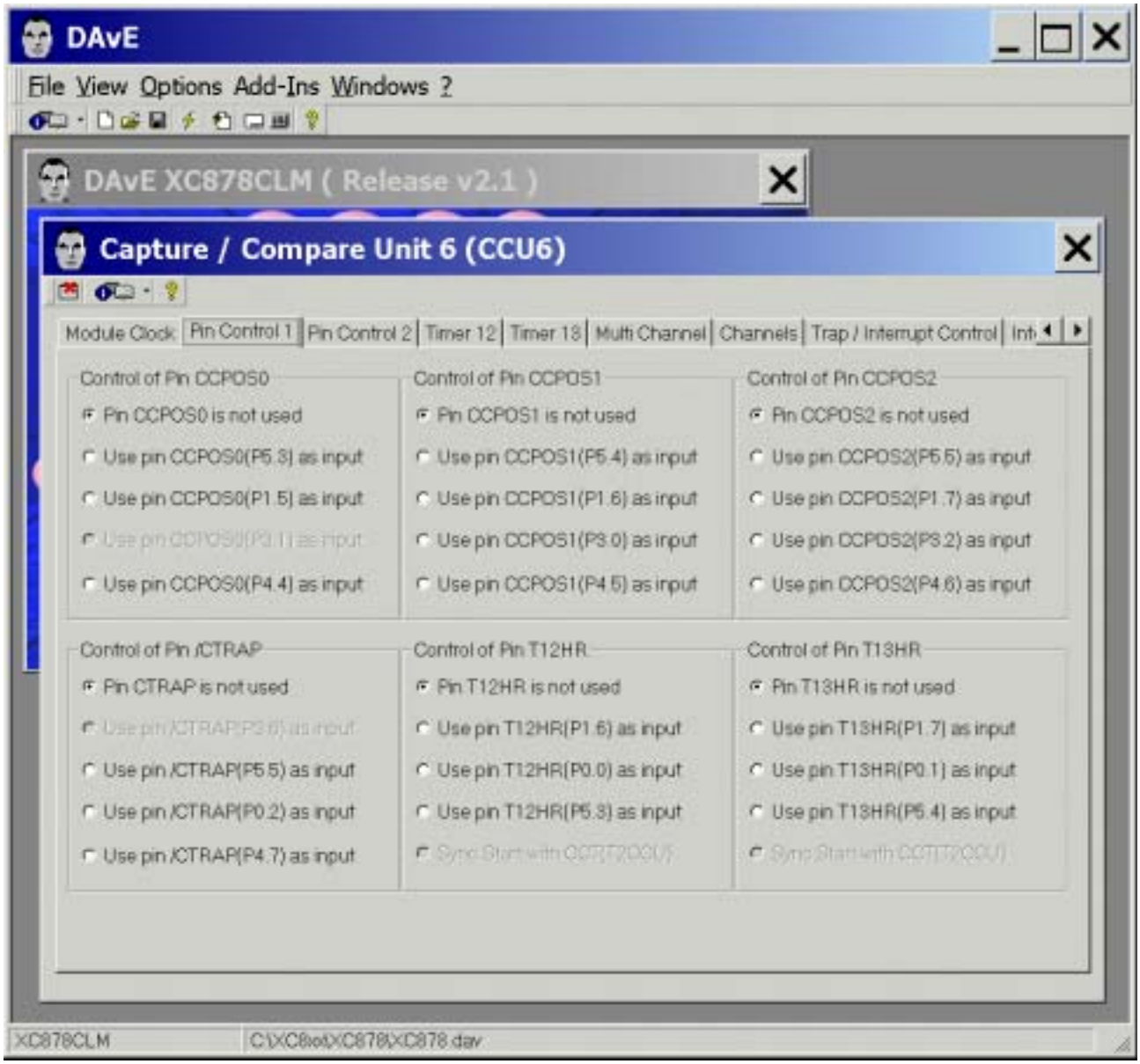
The configuration window/dialog can be opened by clicking the specific block/module (CAPCOM6).



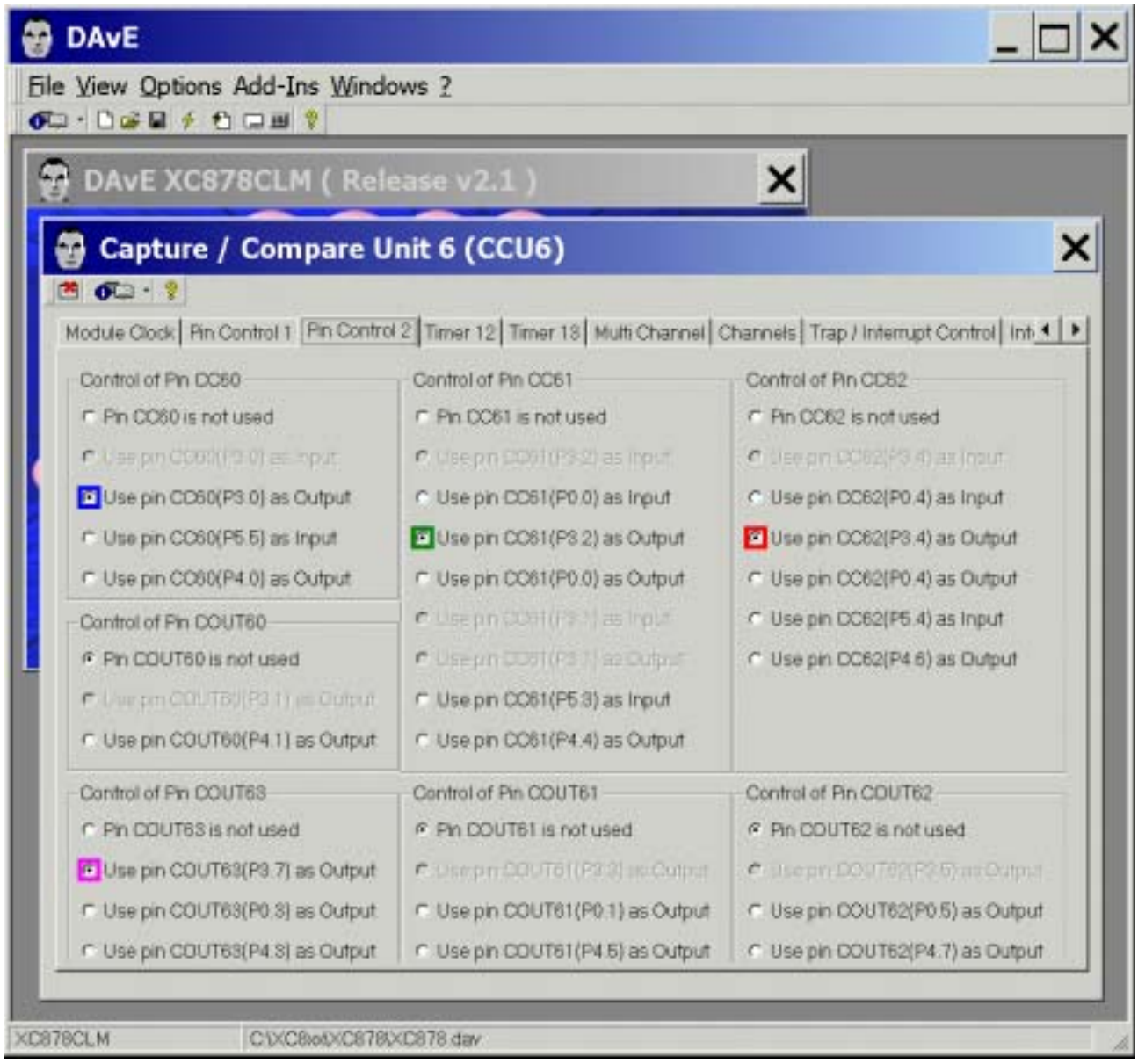
CCU6: Module Clock: CCU6 Disable Flag: click/check ⊙ Enable module
CCU6: Module Clock: Input Clock: click ⊙ FCLK runs at the same frequency as PCLK



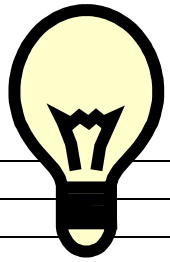
CCU6: Pin Control 1: (do nothing)



CCU6: Pin Control 2: Control of Pin CC60: click ⊙ Use pin CC60 (P3.0) as output
CCU6: Pin Control 2: Control of Pin CC61: click ⊙ Use pin CC61 (P3.2) as output
CCU6: Pin Control 2: Control of Pin CC62: click ⊙ Use pin CC62 (P3.4) as output
CCU6: Pin Control 2: Control of Pin COUT63: click ⊙ Use pin COUT63 (P3.7) as output



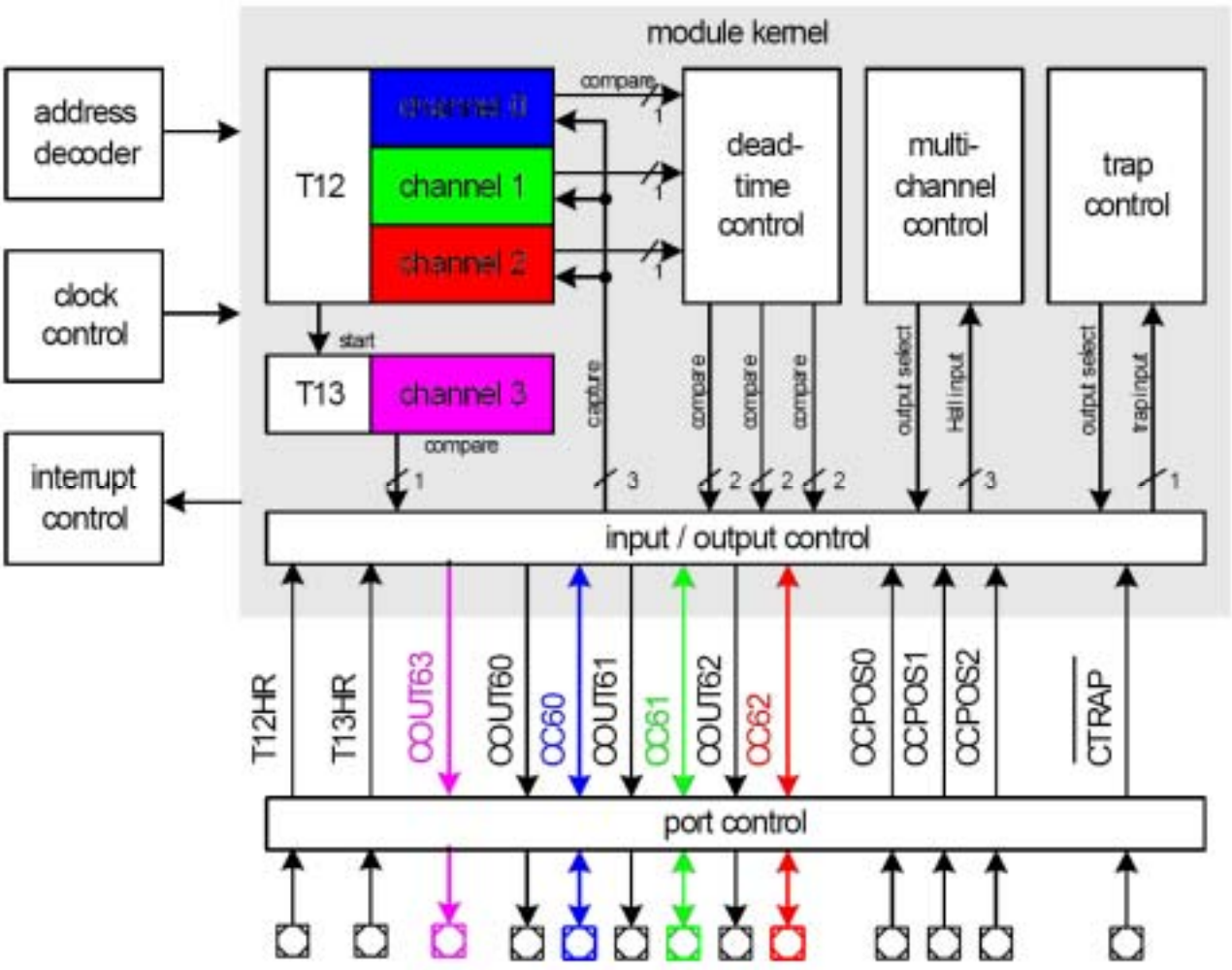
Remember:



Port_3 pins used by our PWM module:

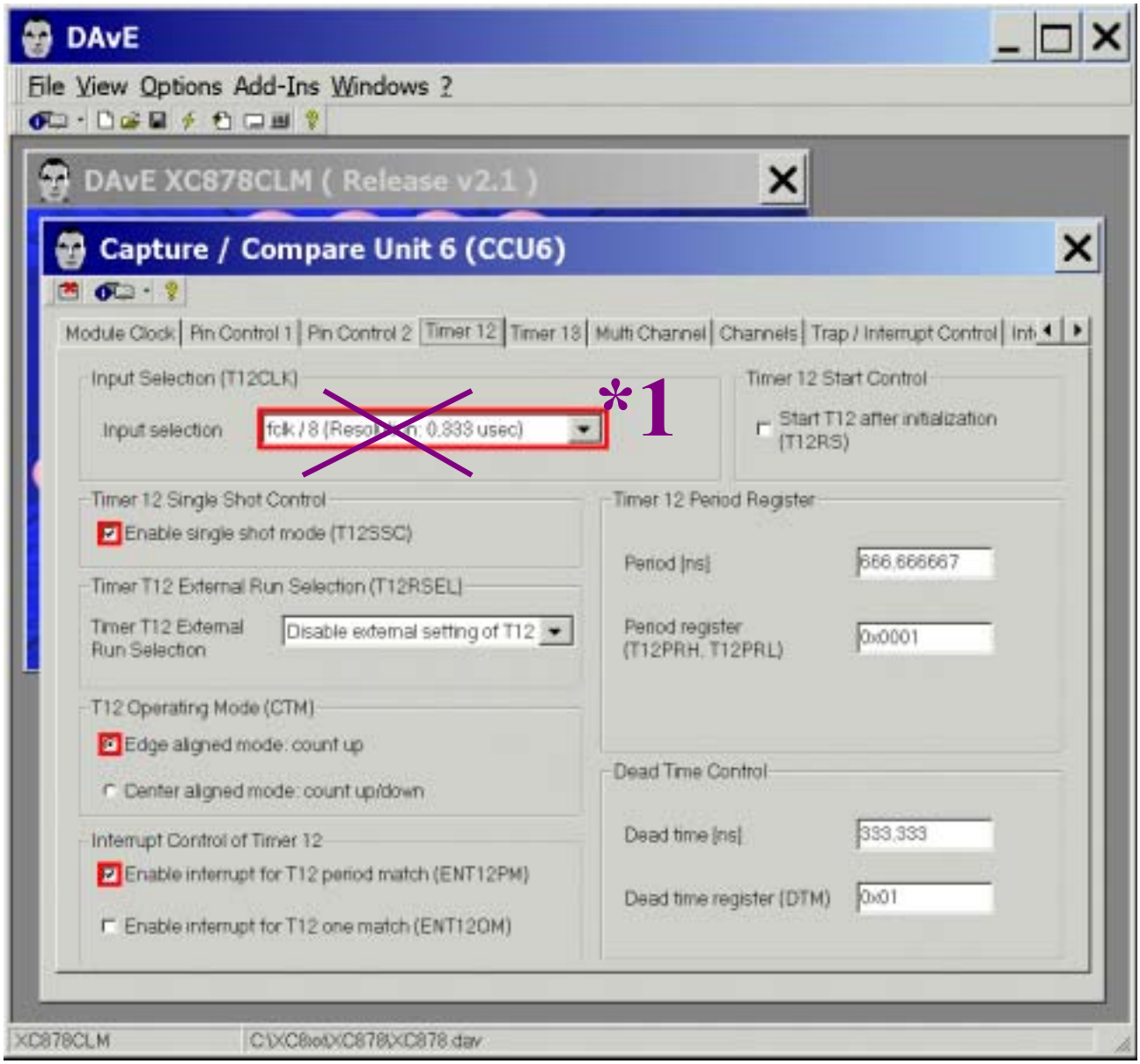
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	100 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

Note:


module kernel
address decoder
clock control
interrupt control
T12
channel 0
channel 1
channel 2
T13
channel 3
compare
start
capture
dead-time control
multi-channel control
trap control
output select
Hall input
trap input
input / output control
T12HR
T13HR
COUT63
COUT60
CC60
COUT61
CC61
COUT62
CC62
CCPOS0
CCPOS1
CCPOS2
CTRAP
port control

Timer 12: “note length”:

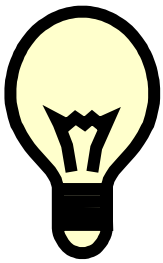
CCU6: Timer 12: Input Selection (T12CLK): choose f_{clk} / 8 ➔ Resolution = 85,333 µs *1
CCU6: Timer 12: T12 Single Shot Control: tick ☑ Enable single shot mode (T12SSC)
CCU6: Timer 12: T12 Operating Mode: click/check ⊙ Edge aligned mode: count up
CCU6: T12: Interrupt Control of Timer 12: tick ☑ Enable interrupt for T12 period match



*1: See next page !!!

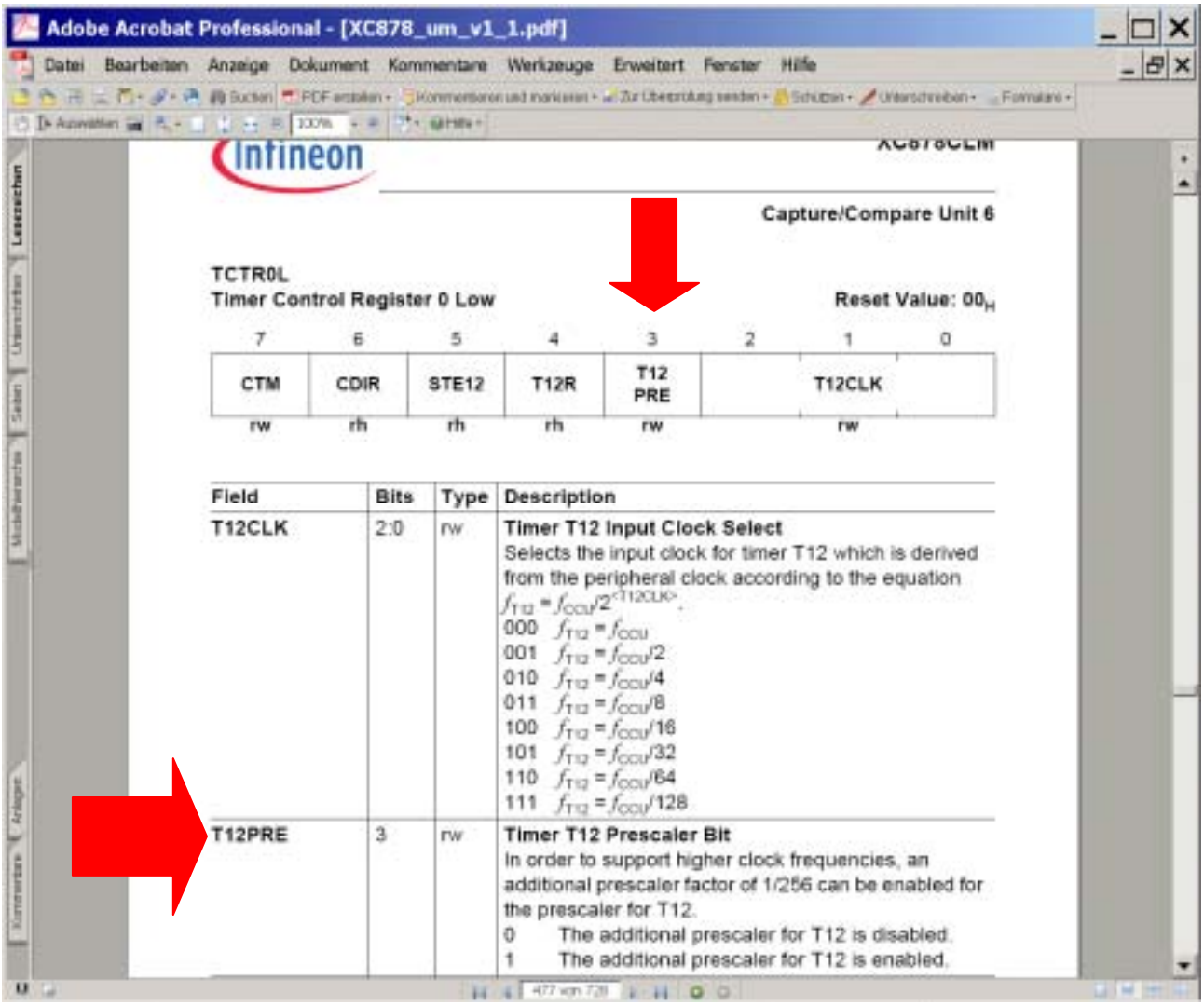
Timer 12 Resolution = 11,719 kHz / 85,333 µs

<<< !!! 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_{CCU6} to a maximum of $f_{CCU6}/128$ 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.



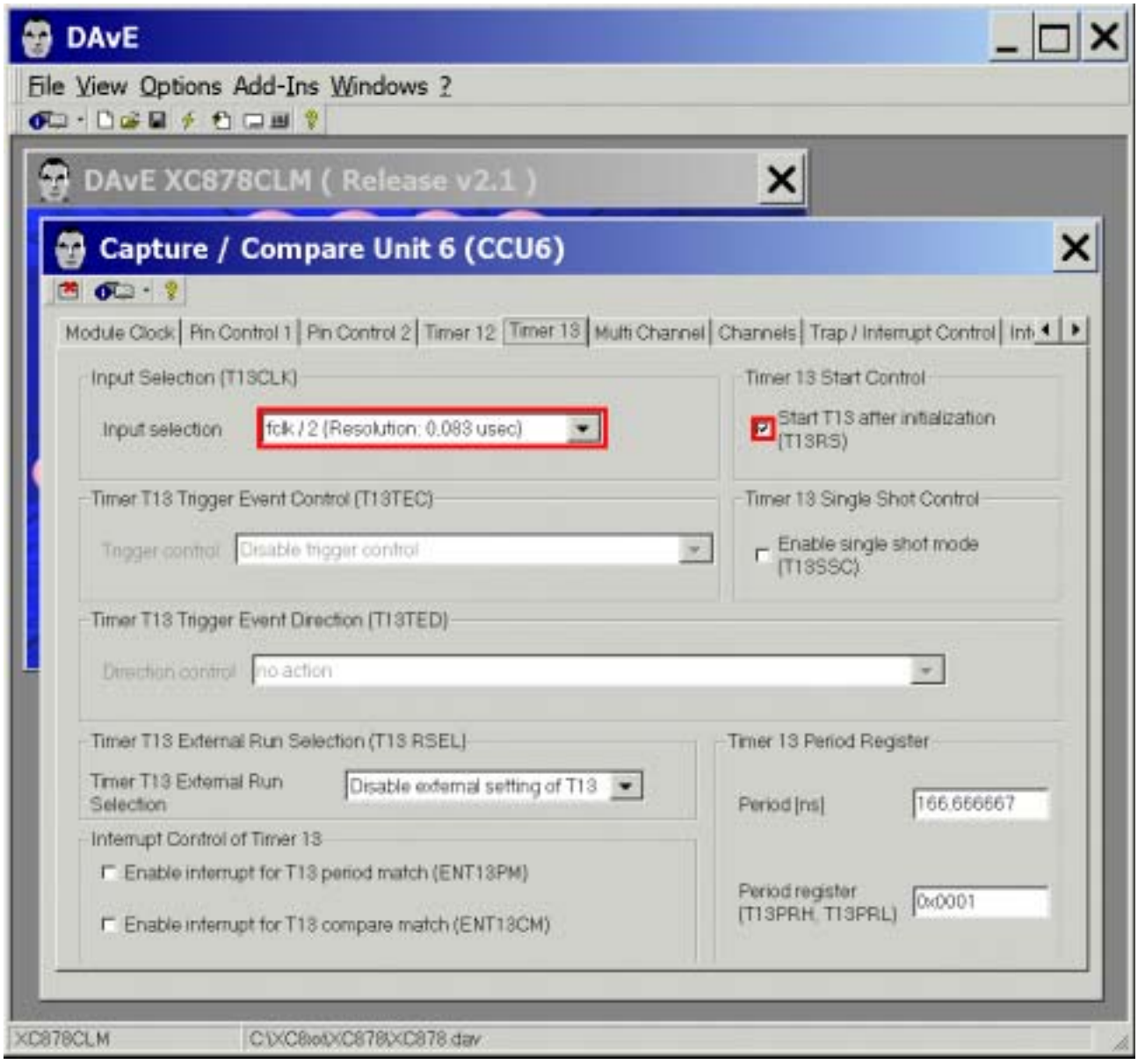
*1:

Timer 12 Resolution:

24 MHz / 256 (T12PRE=1, done by software) / 8 = 11,719 kHz ➔ Resolution = 85,333 µs

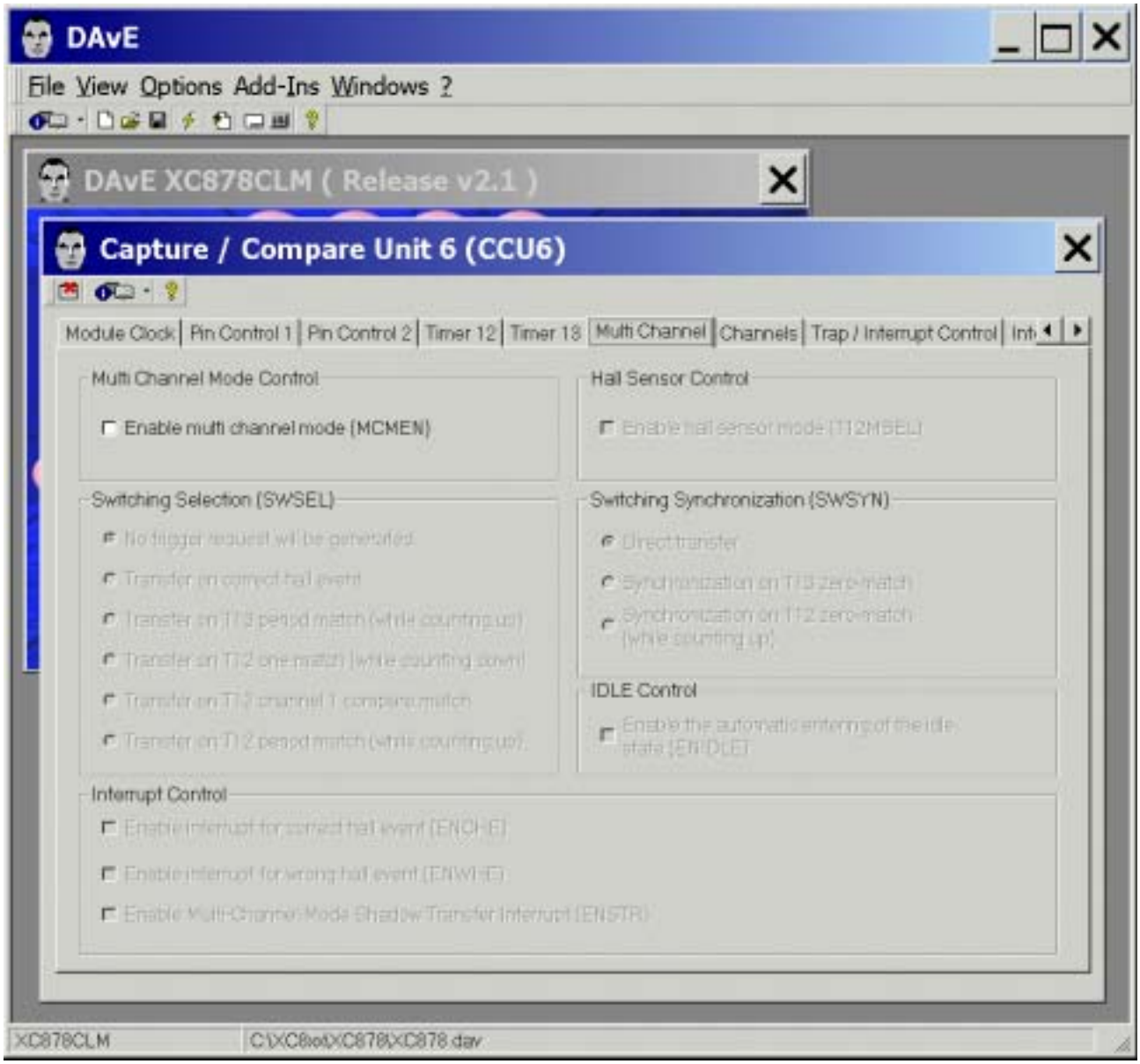
Timer 13: ”note frequency”:

CCU6: Timer T13: Input Selection: Input selection select fclk/2 (Resolution: 83,333 ns)
CCU6: Timer T13: Timer 13 Start Control: click ✓ Start T13 after initialization (T13RS)

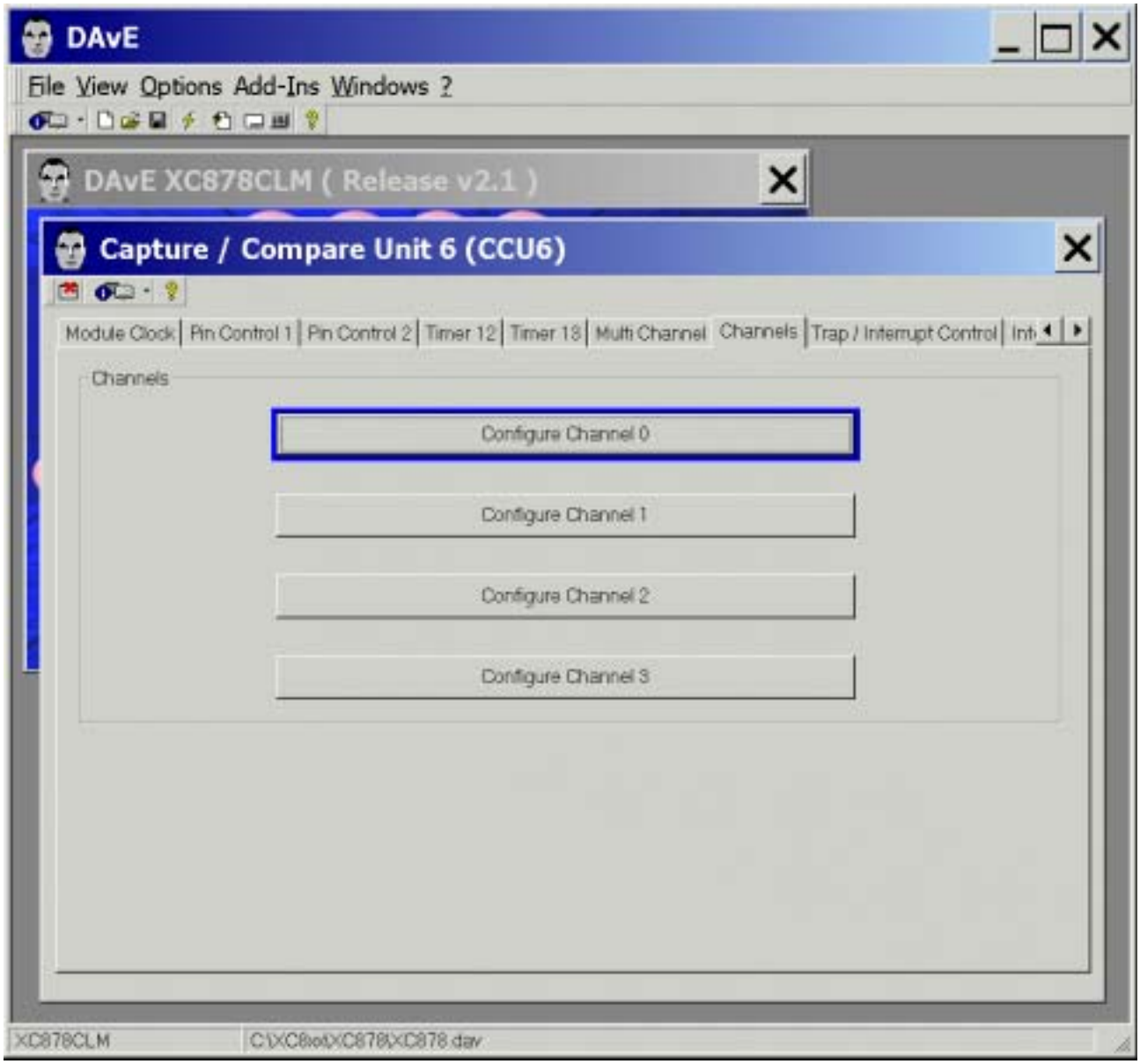


<<< !!! click here to see more information about music !!! >>>

CCU6: Multi Channel: (do nothing)



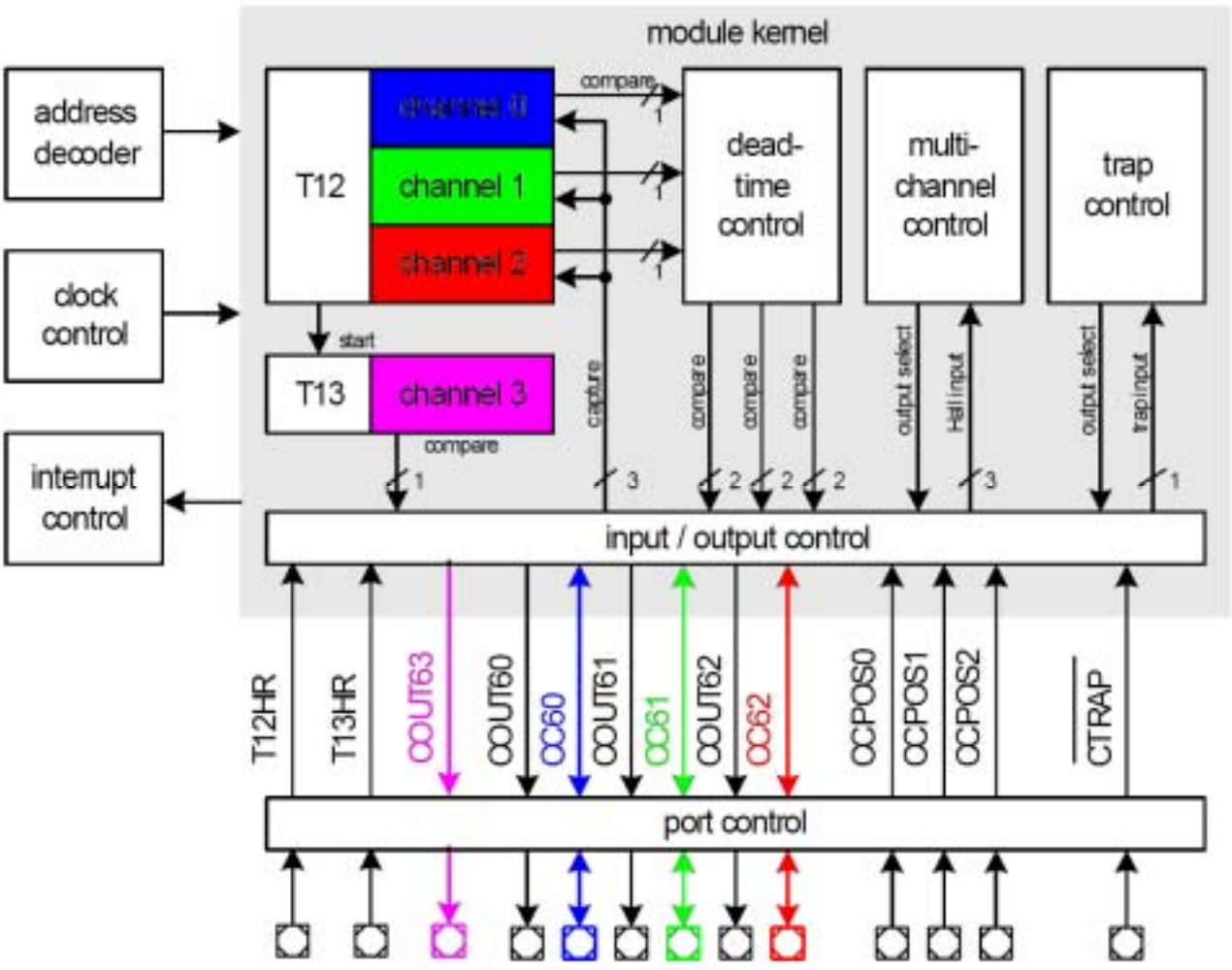
CCU6: Channels: click Configure Channel 0



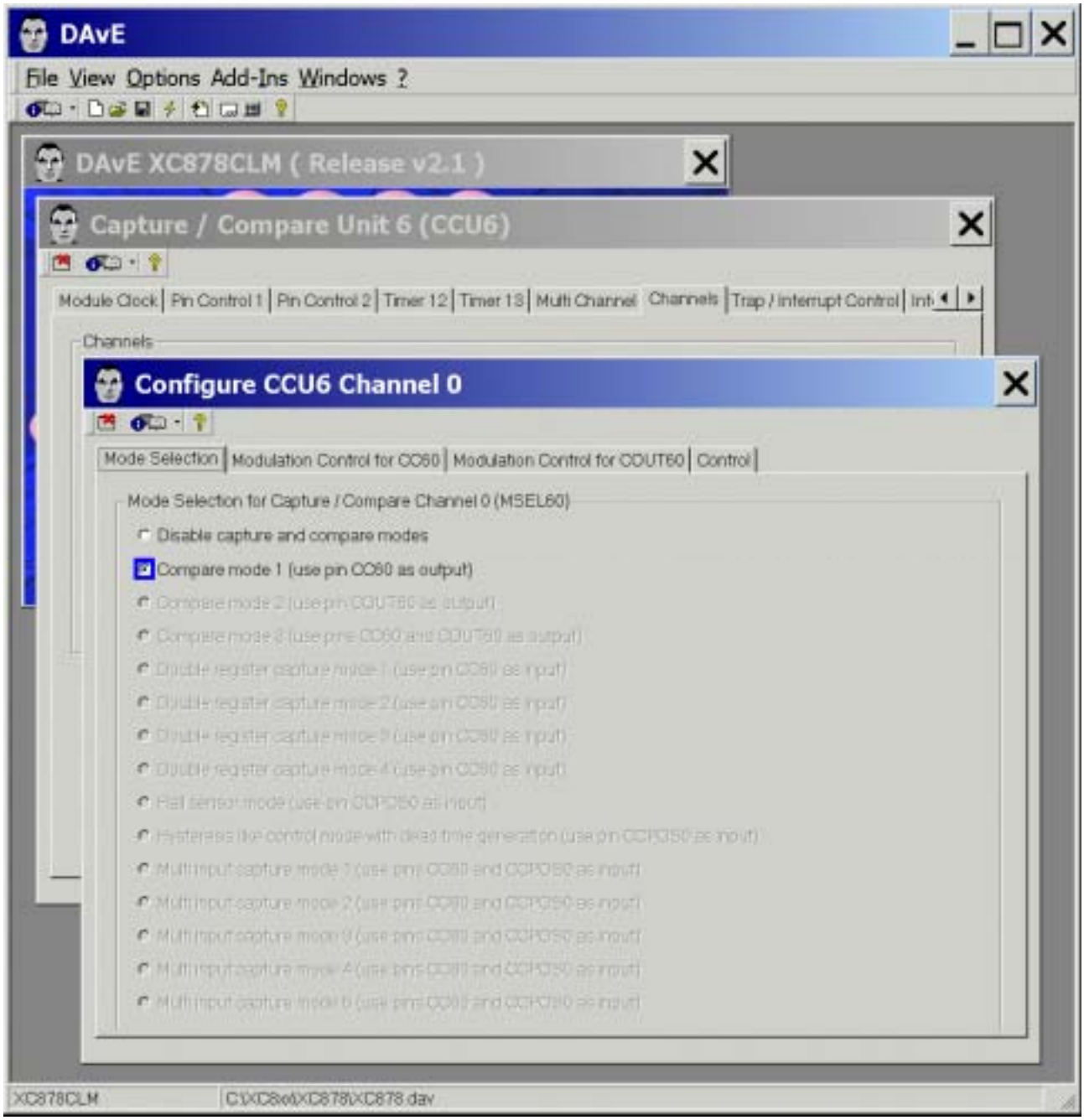
Note:
Port pins used by our PWM module (not available as GPIO pins):

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	100 (T12)
P3.2	CC61_0	Channel 1	100 (T12)
P3.4	CC62_0	Channel 2	100 (T12)+50 (T13)
P3.7	COUT63_0	Channel 3	50 (T13)

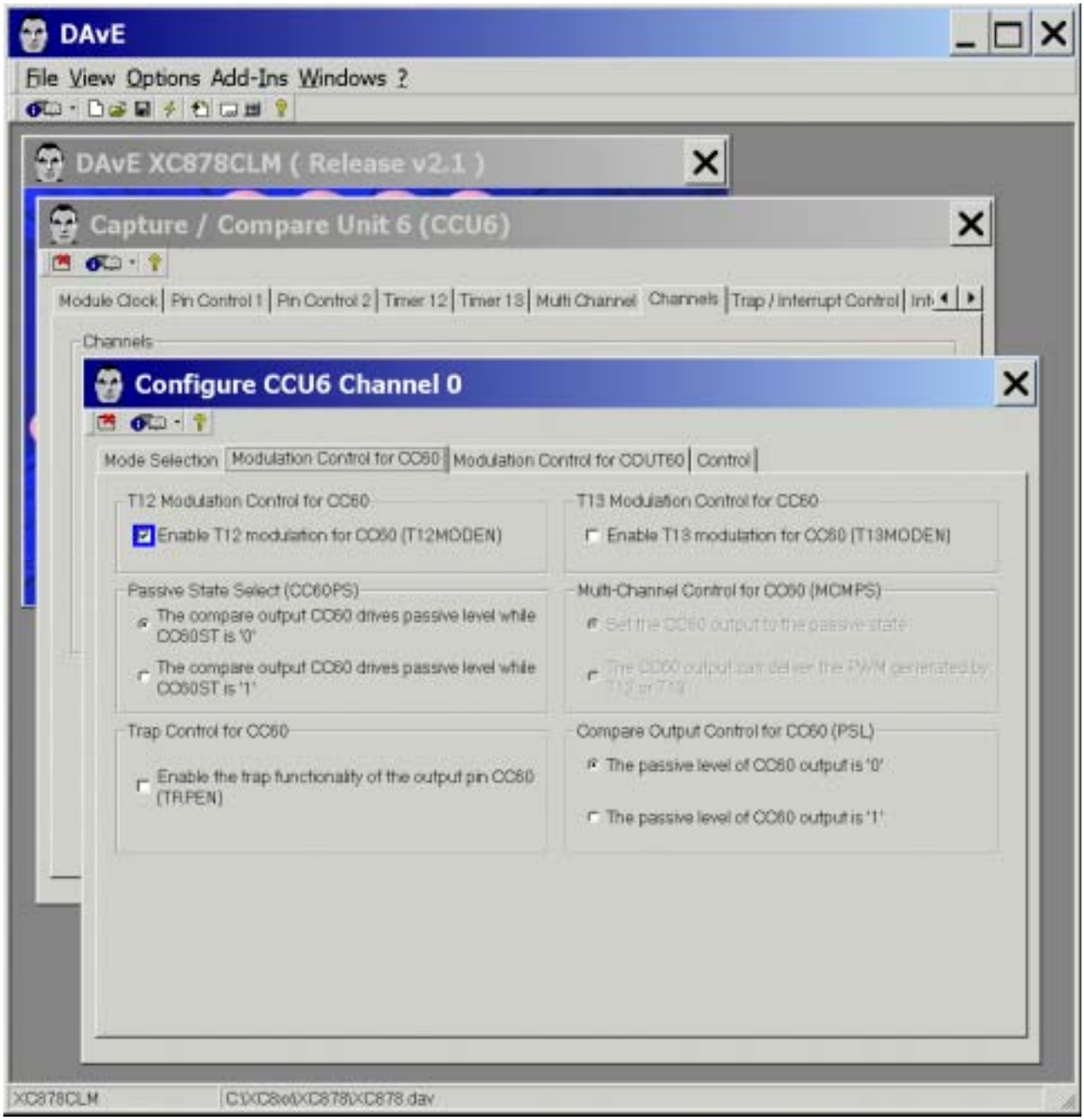
Note:


module kernel
address decoder
clock control
interrupt control
T12
channel 0
channel 1
channel 2
T13
channel 3
start
compare
capture
dead-time control
multi-channel control
trap control
output select
Hall input
trap input
input / output control
T12HR
T13HR
COUT63
COUT60
CC60
COUT61
CC61
COUT62
CC62
CCPOS0
CCPOS1
CCPOS2
CTRAP
port control

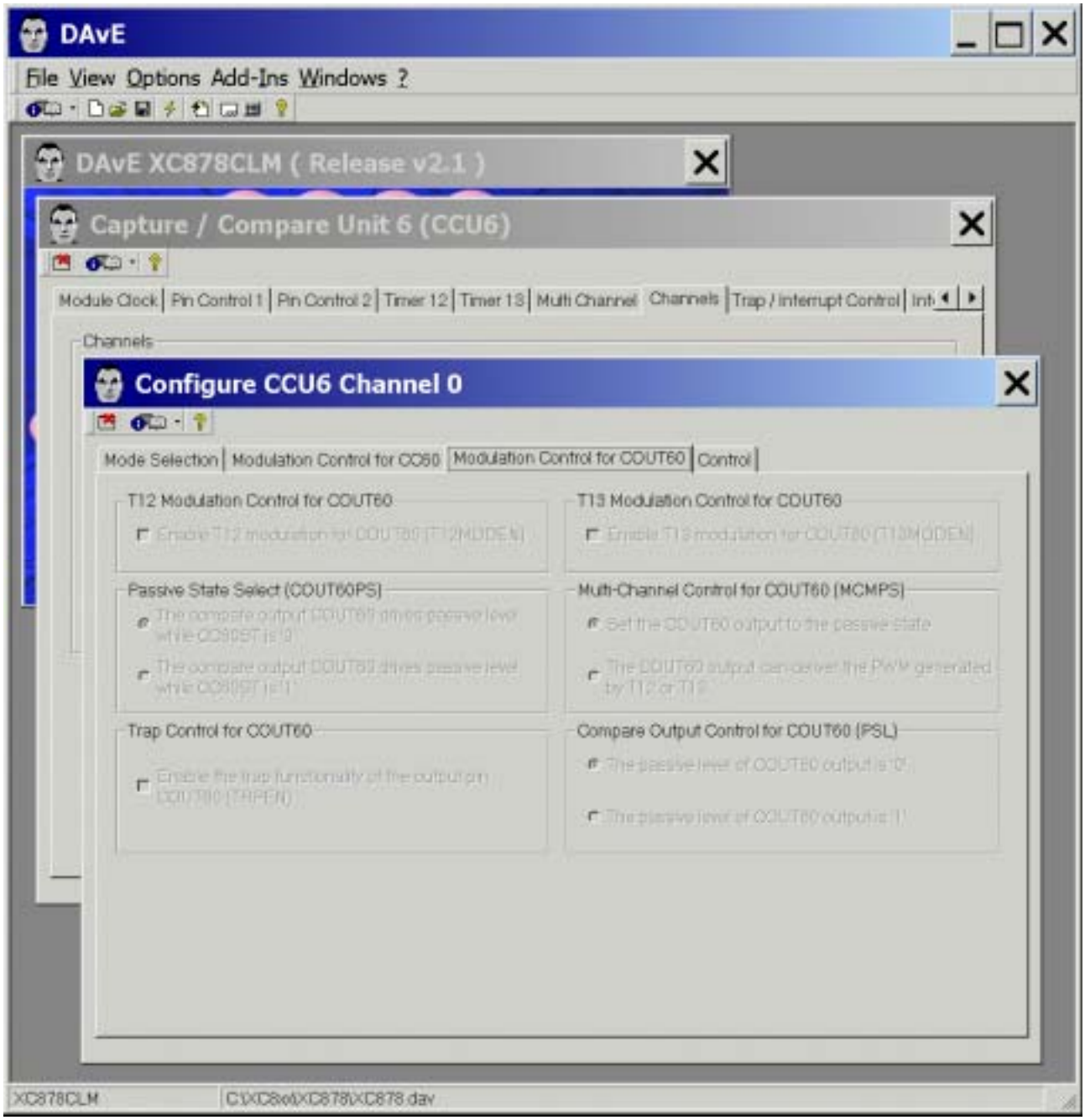
CCU6: Channels: Configure Channel 0:
Mode Selection: Mode Selection for Capture / Compare Channel 0: click ⊙ Compare mode 1



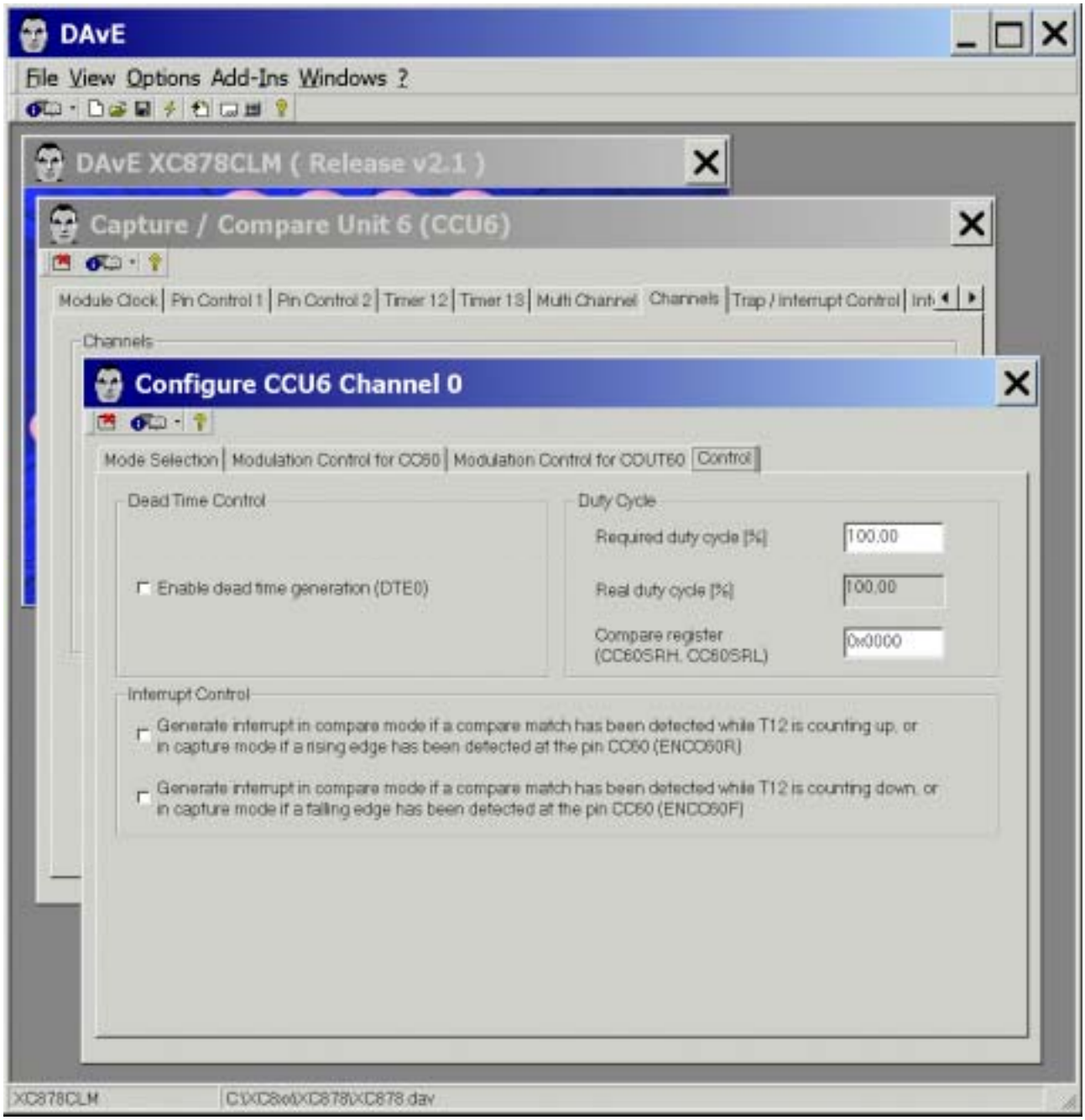
CCU6: Channels: Configure Channel 0: Modulation Control for CC60:
T12 Modulation Control for CC60: tick ☑ Enable T12 modulation for CC60



CCU6: Channels: Configure Channel 0: Modulation Control for COUT60: (do nothing)

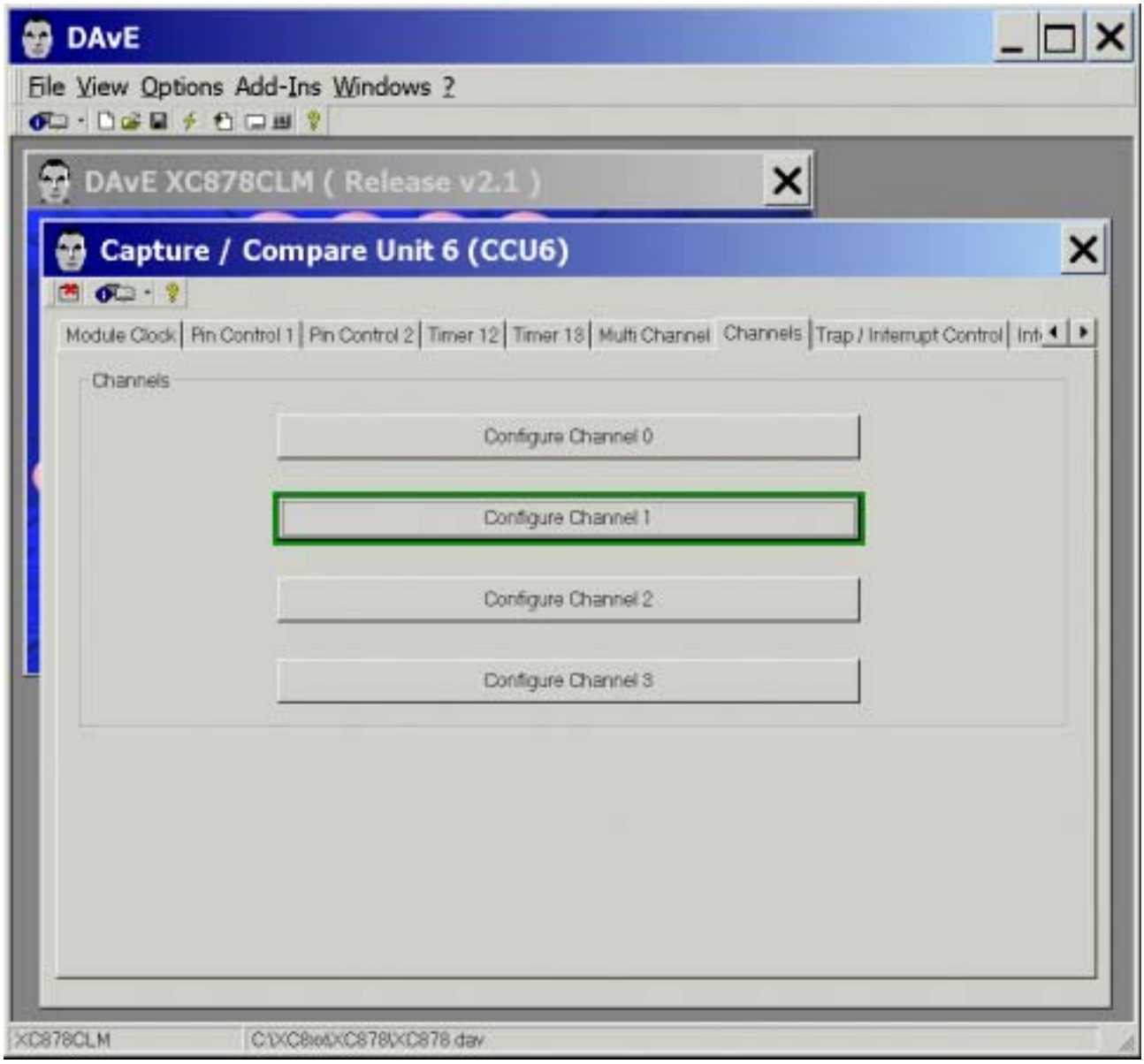


CCU6: Channels: Configure Channel 0: Control: (do nothing)



Exit this dialog now by clicking  the close button.

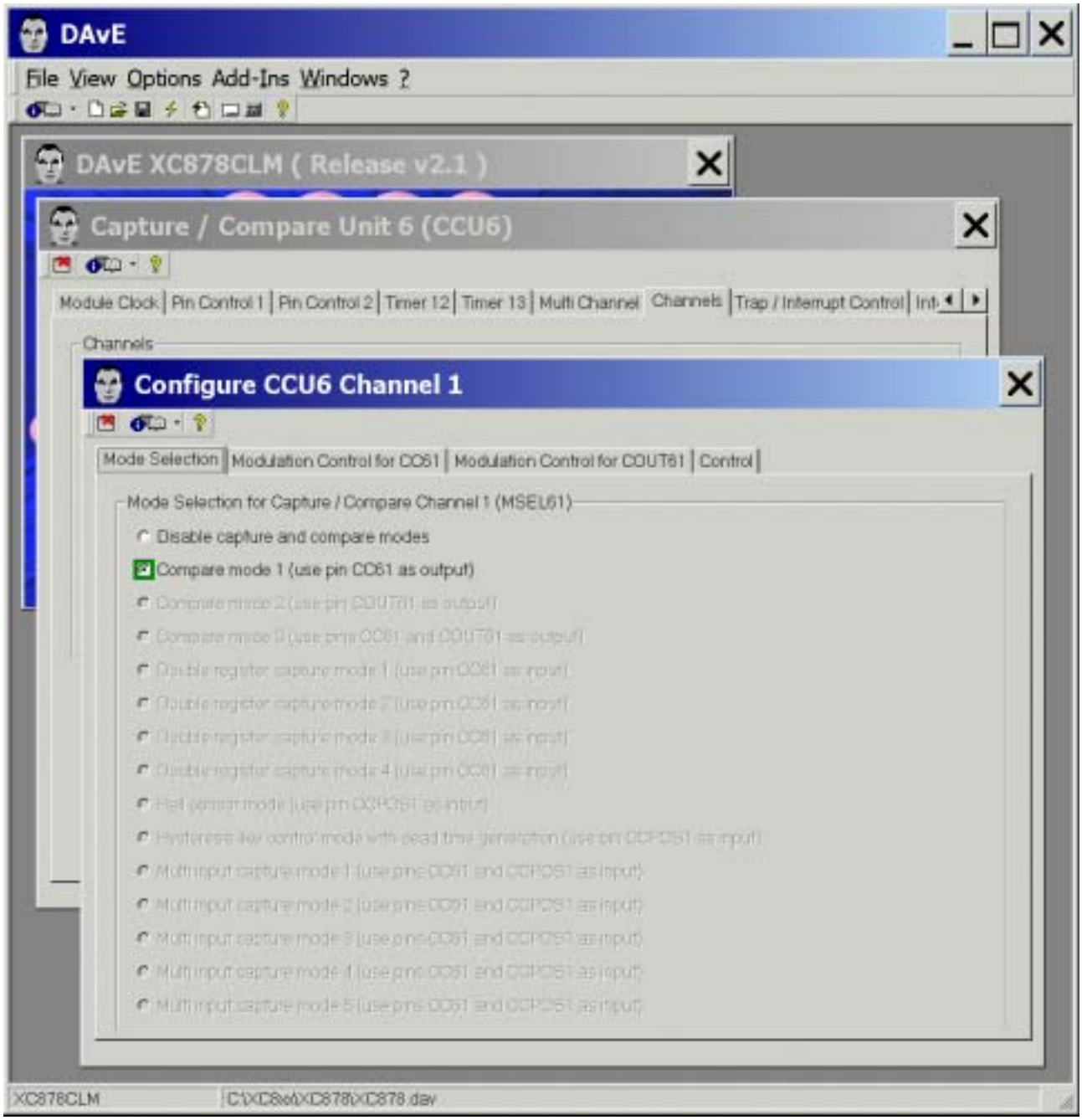
CCU6: Channels: click Configure Channel 1



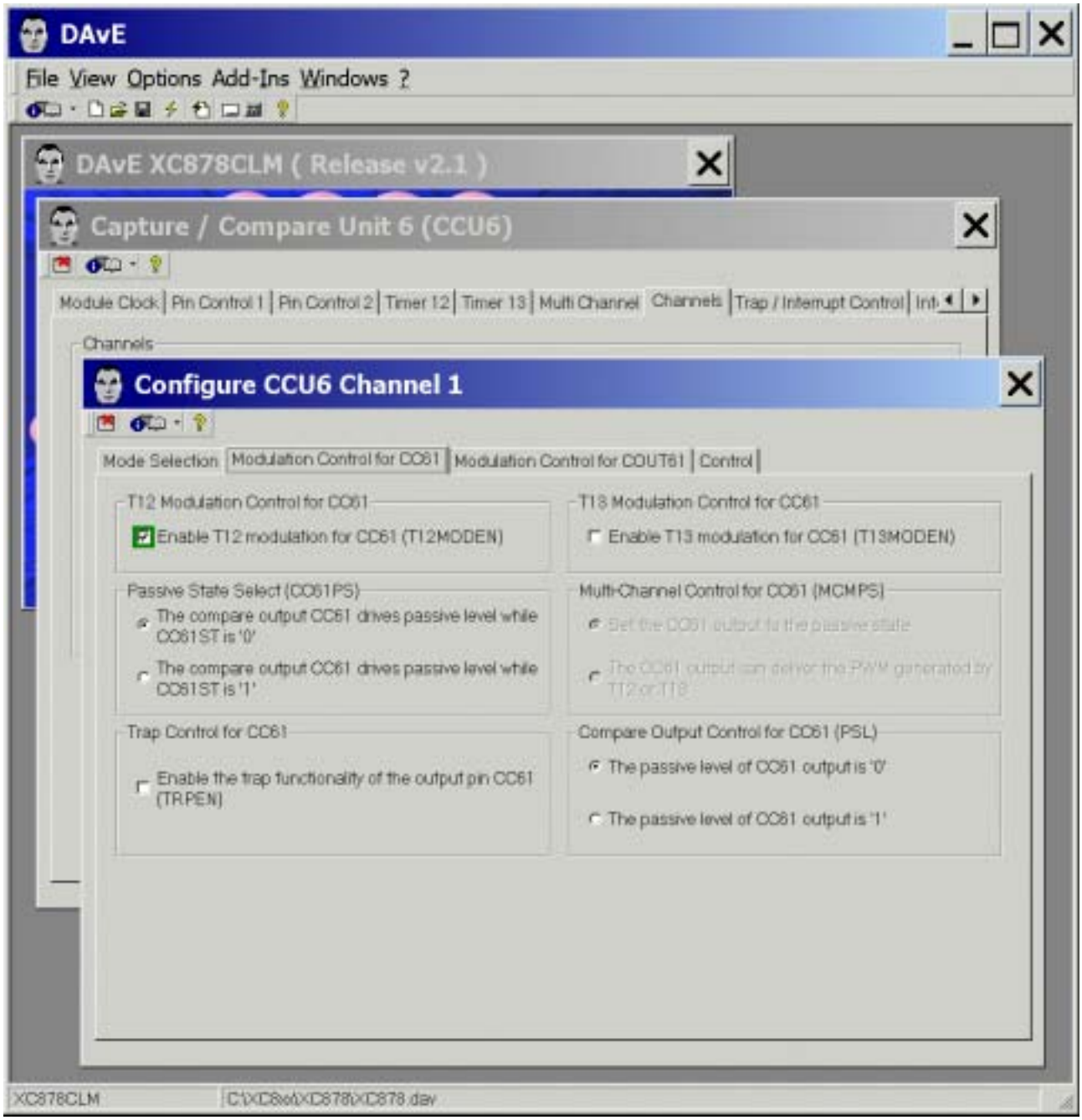
Note:
Port pins used by our PWM module (not available as GPIO pins):

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	100 (T12)
P3.2	CC61_0	Channel 1	100 (T12)
P3.4	CC62_0	Channel 2	100 (T12)+50 (T13)
P3.7	COUT63_0	Channel 3	50 (T13)

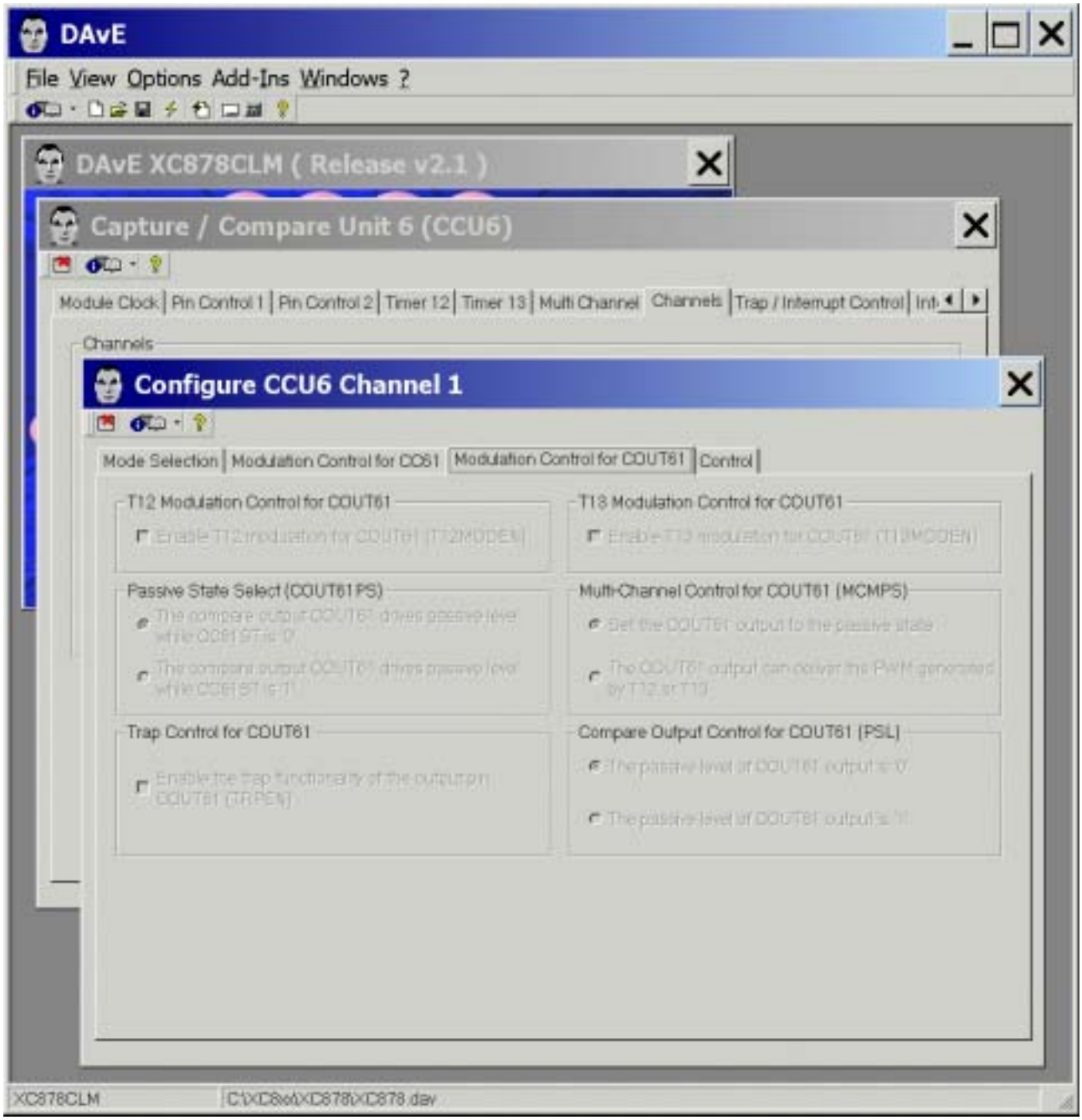
CCU6: Channels: Configure Channel 1:
Mode Selection: Mode Selection for Capture / Compare Channel 1: click ⊙ Compare mode 1



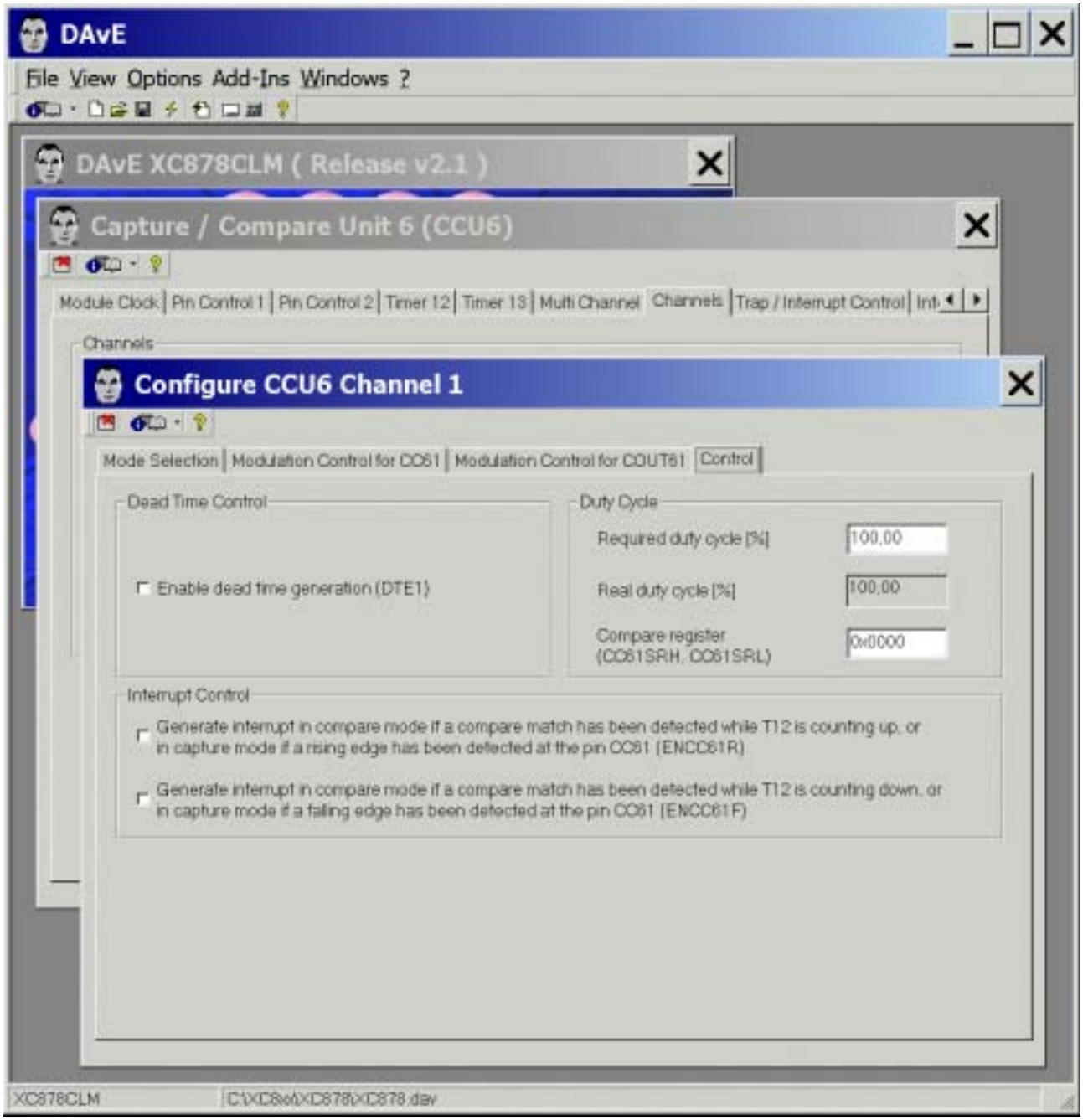
CCU6: Channels: Configure Channel 1: Modulation Control for CC61:
T12 Modulation Control for CC61: tick ☑ Enable T12 modulation for CC61



CCU6: Channels: Configure Channel 1:
Modulation Control for COUT61: (do nothing)

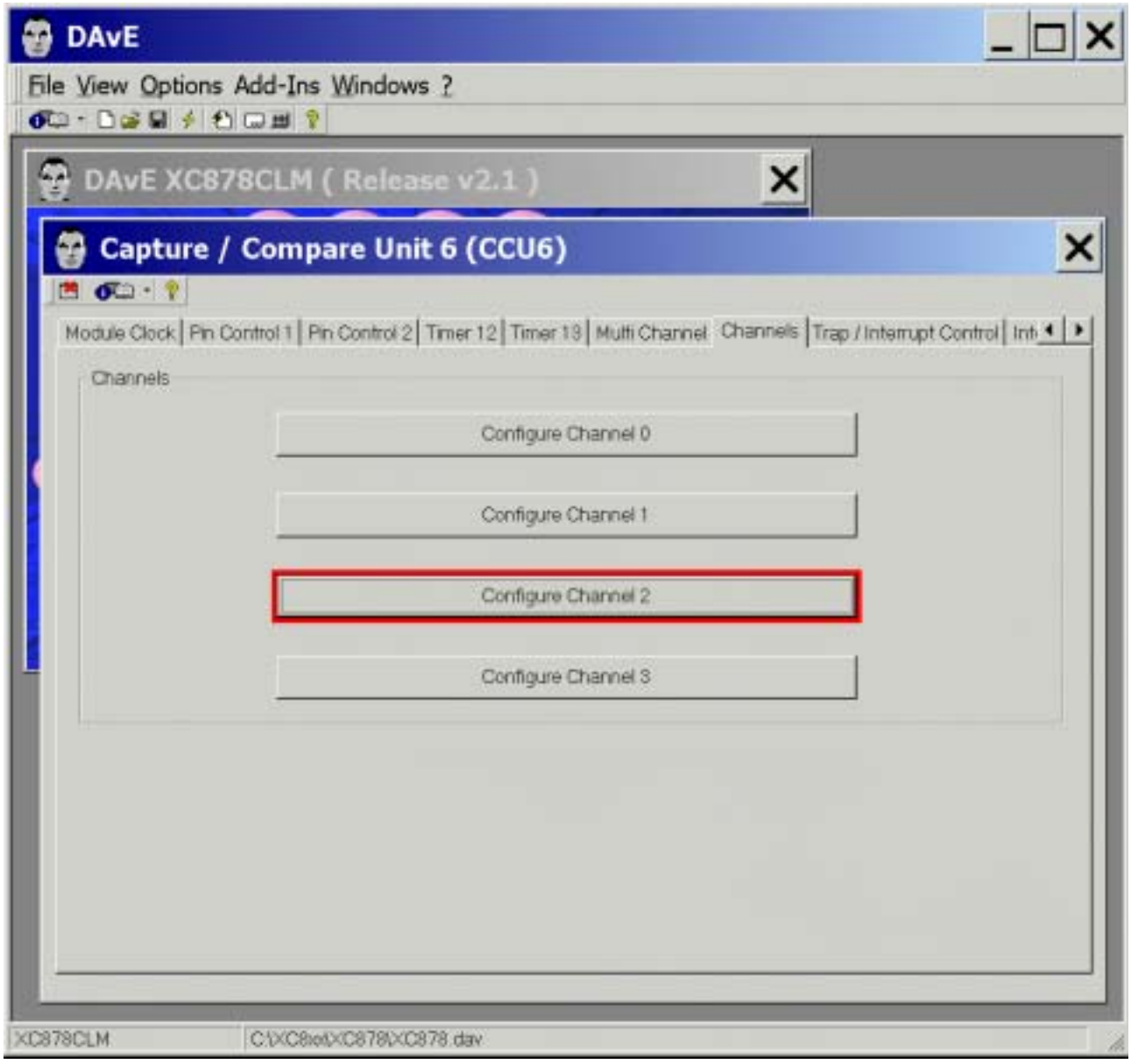


CCU6: Channels: Configure Channel 1: Control: (do nothing)

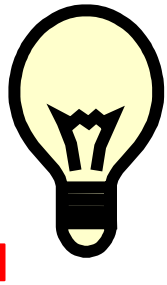


Exit this dialog now by clicking  the close button.

CCU6: Channels: click Configure Channel 2

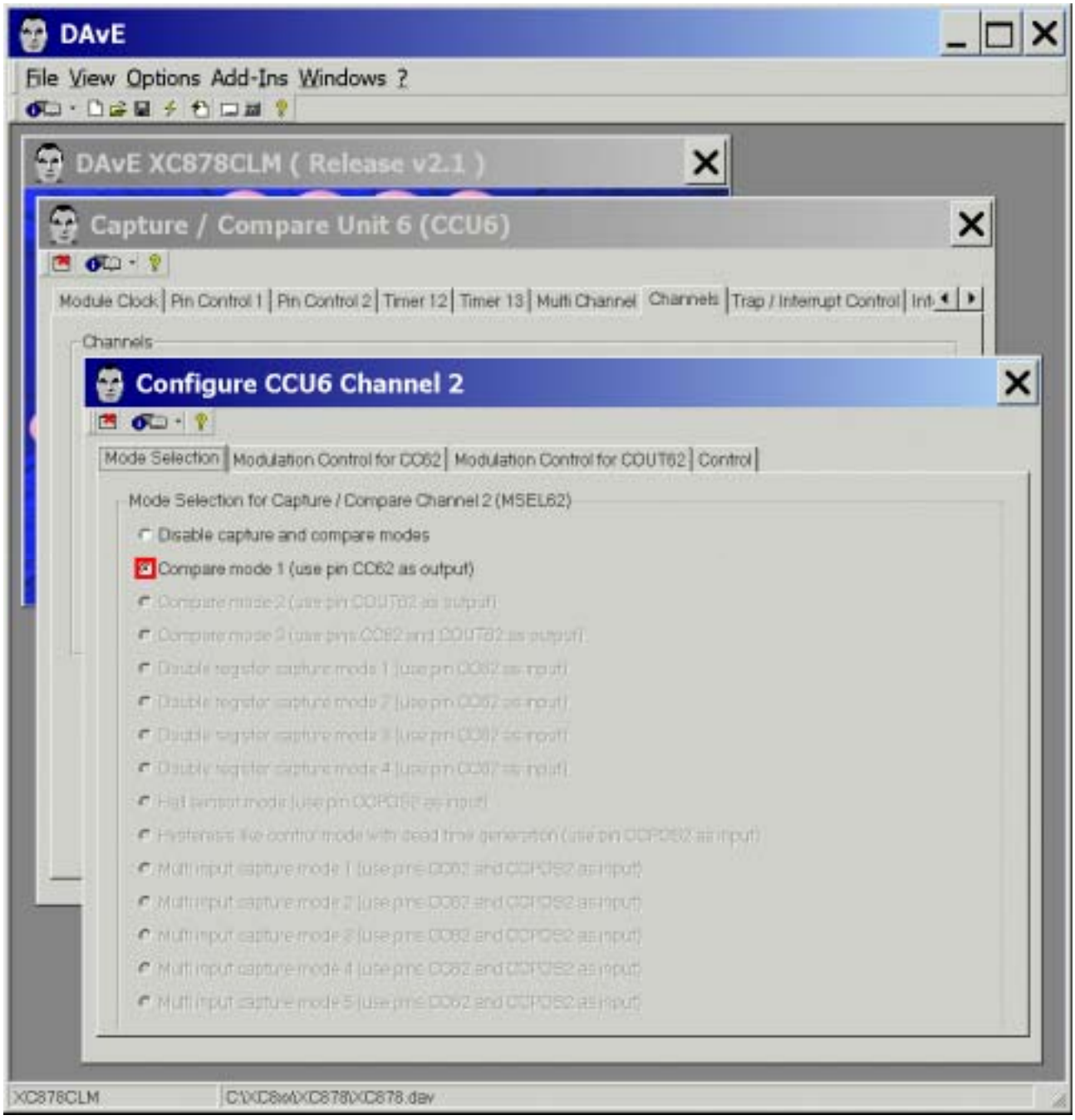


Note:
Port pins used by our PWM module (not available as GPIO pins):



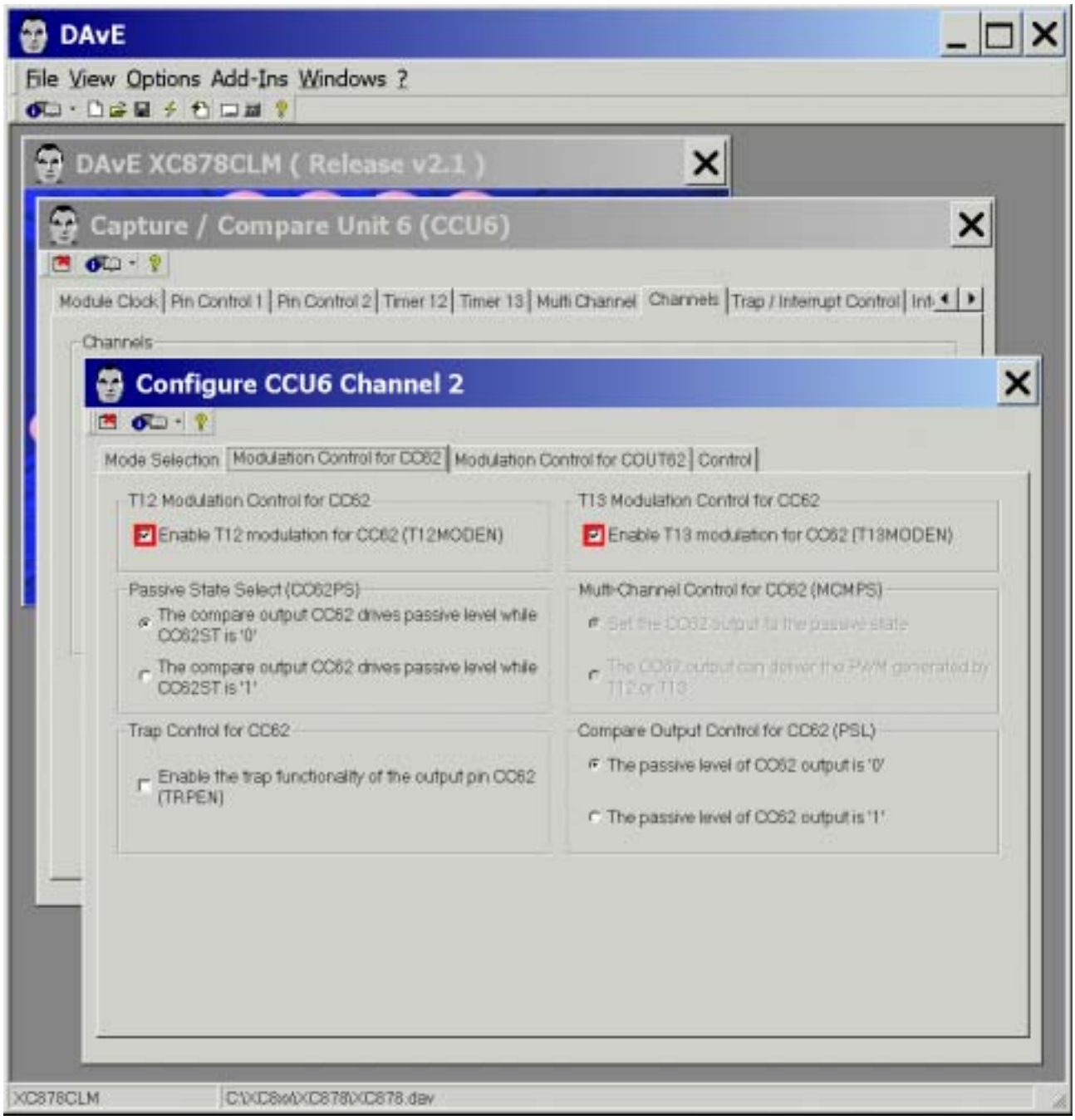
Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	100 (T12)
P3.2	CC61_0	Channel 1	100 (T12)
P3.4	CC62_0	Channel 2	100 (T12)+50 (T13)
P3.7	COUT63_0	Channel 3	50 (T13)

CCU6: Channels: Configure Channel 2:
Mode Selection: Mode Selection for Capture / Compare Channel 2: click ⊙ Compare mode 1

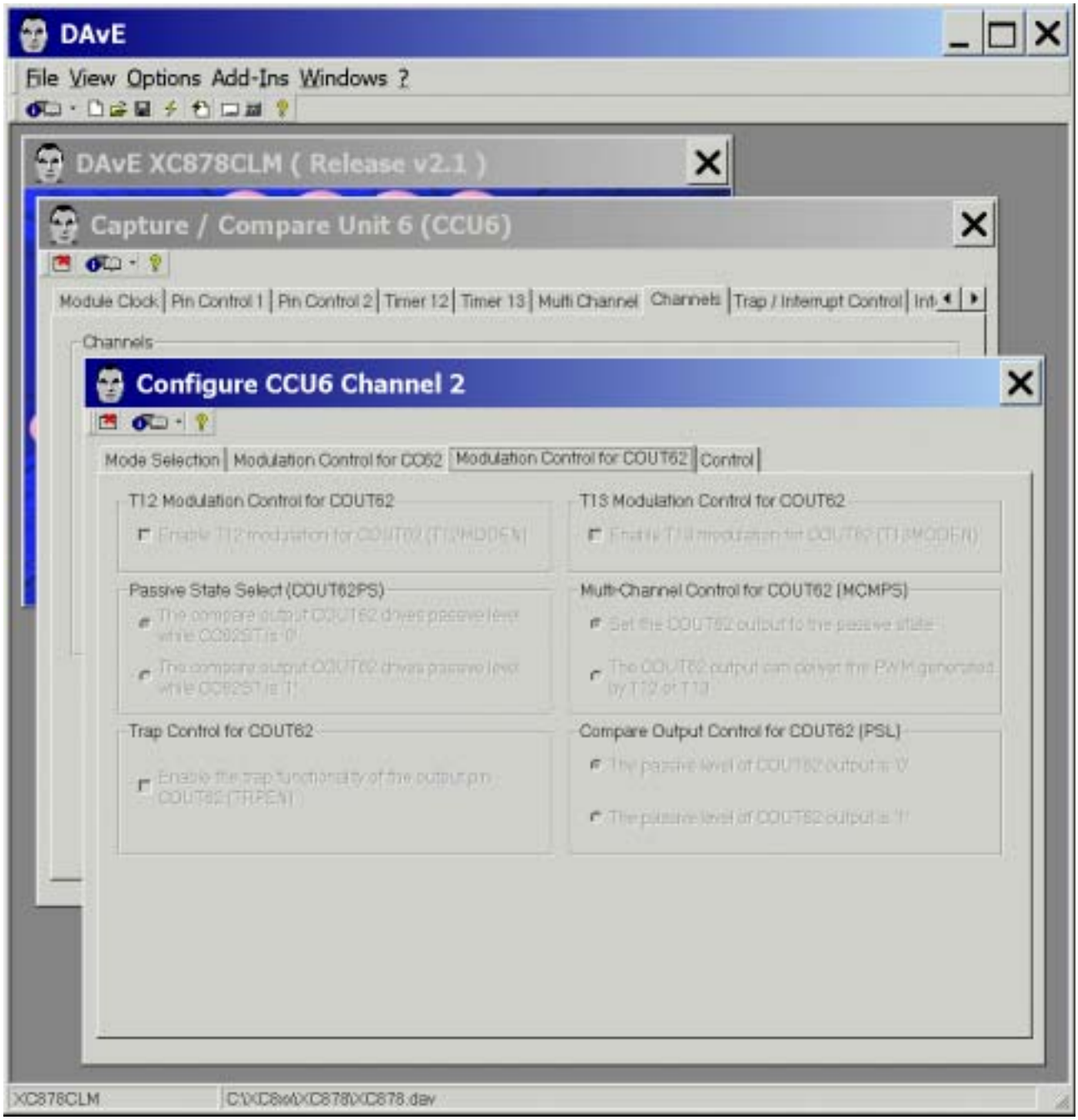


CCU6: Channels: Configure Channel 2: Modulation Control for CC62:
T12 Modulation Control for CC62: tick ☑ Enable T12 modulation for CC62

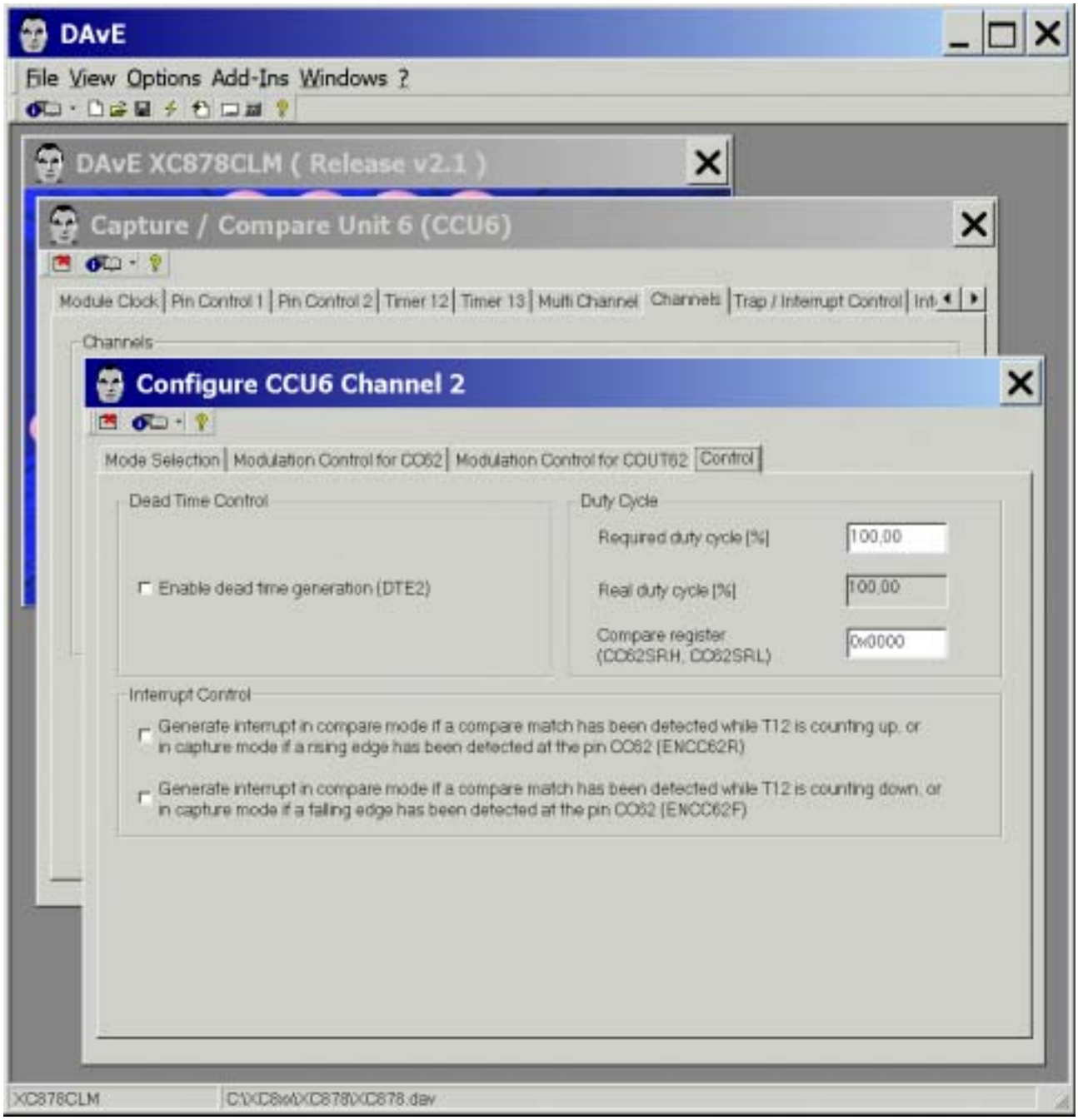
CCU6: Channels: Configure Channel 2: Modulation Control for CC62:
T13 Modulation Control for CC62: tick ☑ Enable T13 modulation for CC62



CCU6: Channels: Configure Channel 2: Modulation Control for COUT62: (do nothing)

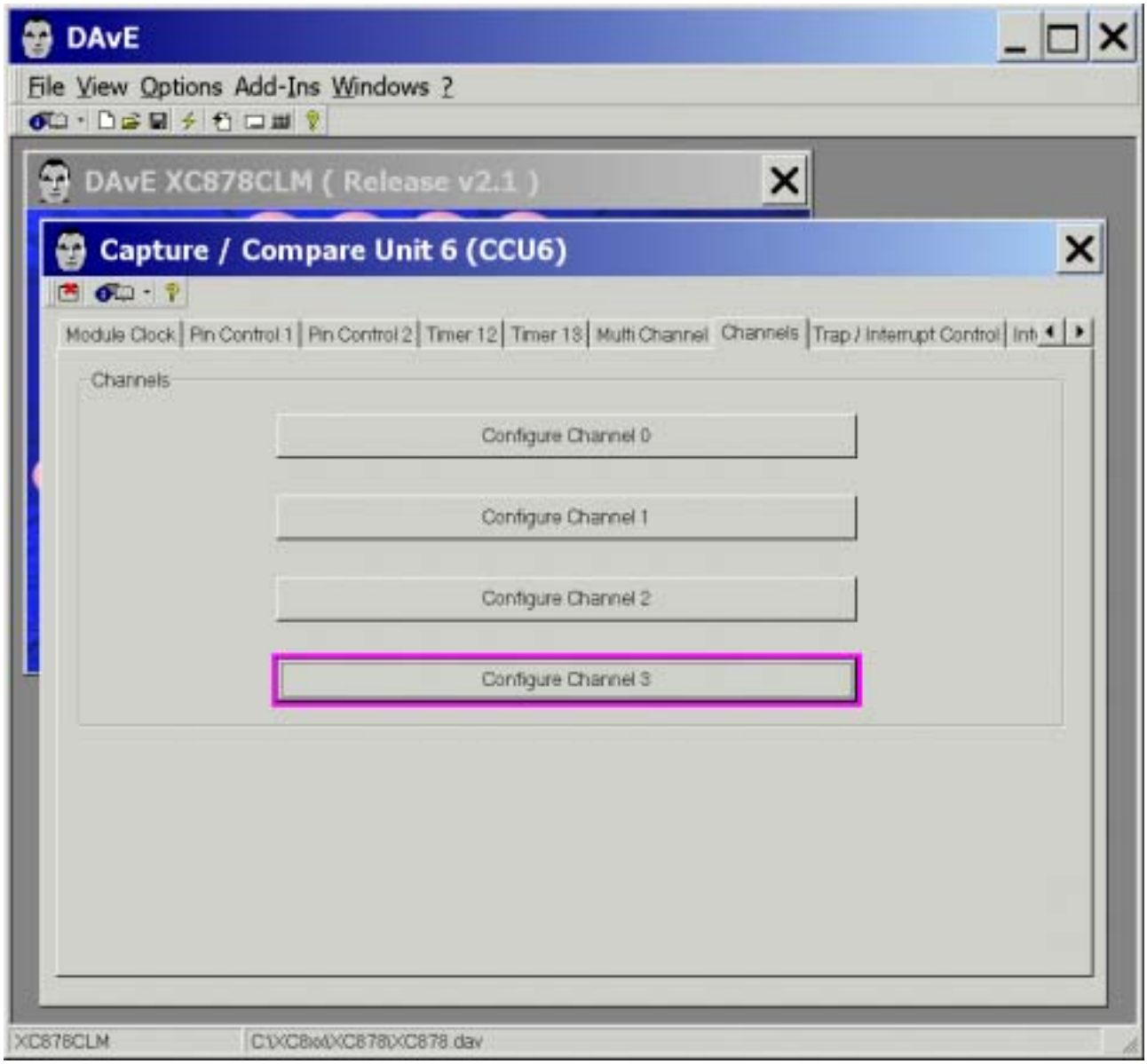


CCU6: Channels: Configure Channel 2: Control: (do nothing)



Exit this dialog now by clicking  the close button.

CCU6: Channels: click Configure Channel 3

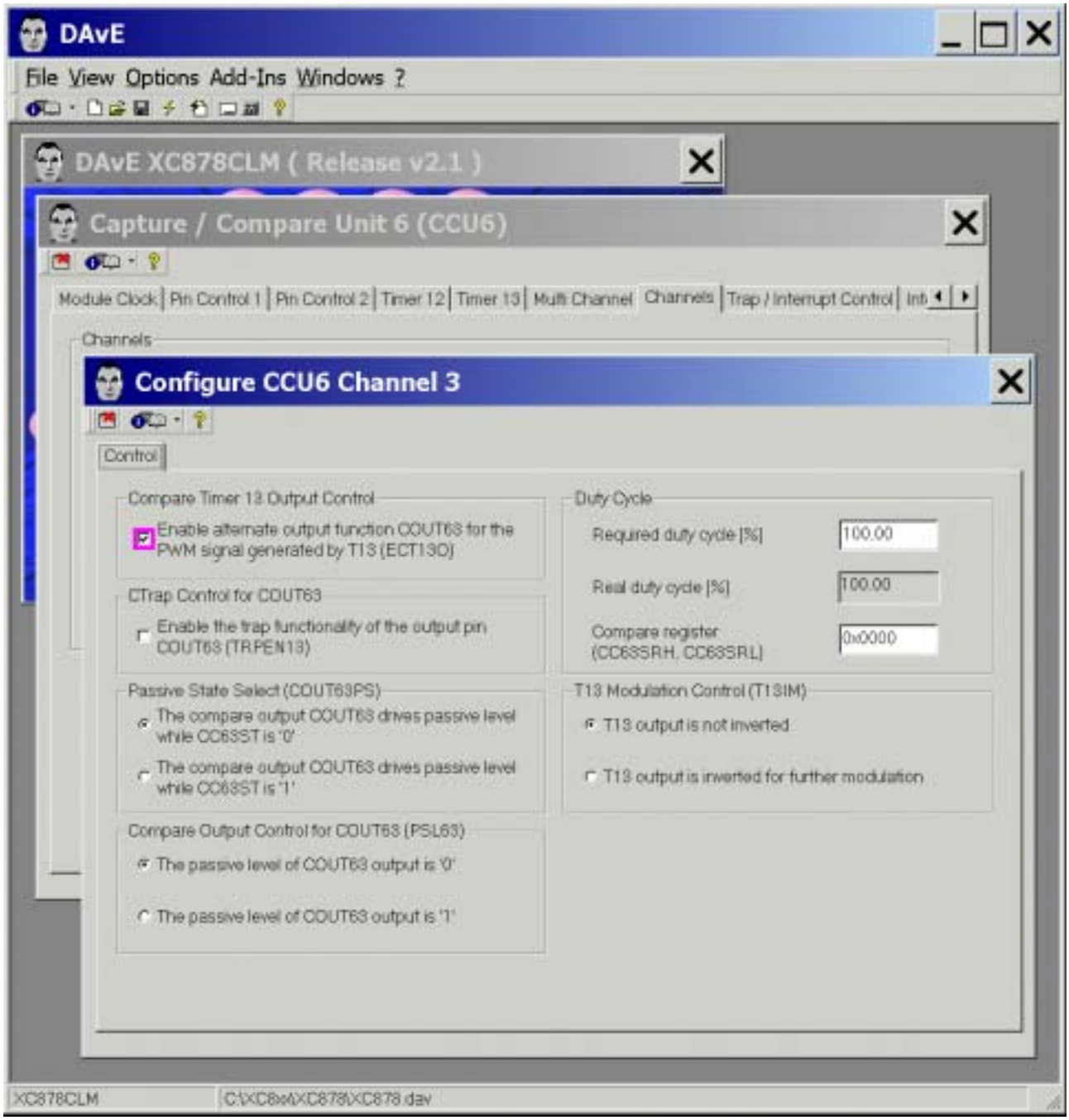


Remember:

Port pins used by our PWM module (not available as GPIO pins):

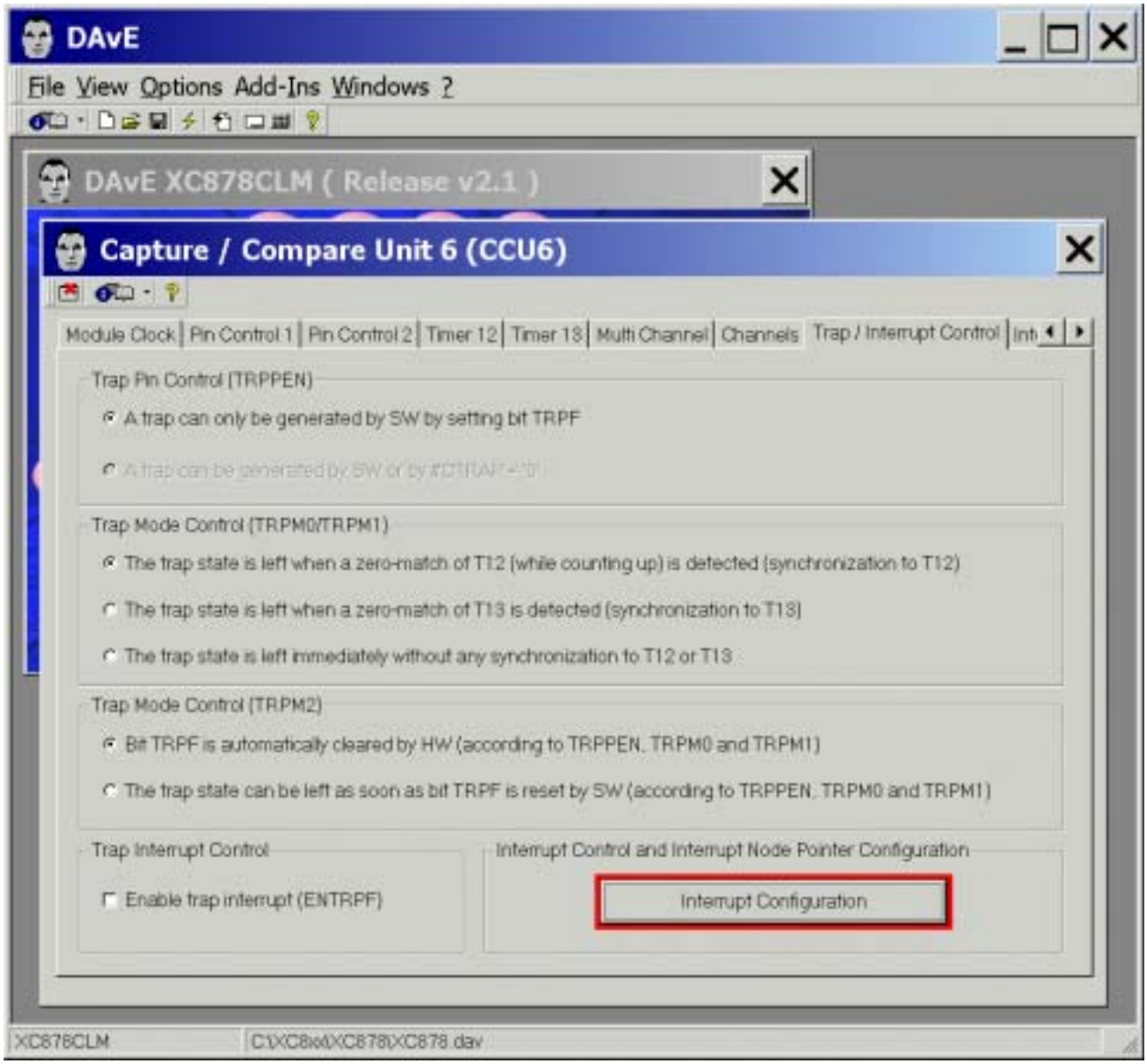
Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	100 (T12)
P3.2	CC61_0	Channel 1	100 (T12)
P3.4	CC62_0	Channel 2	100 (T12)+50 (T13)
P3.7	COUT63_0	Channel 3	50 (T13)

CCU6: Channels: Configure Channel 3: Control: Compare Timer 13 Output Control:
tick ☑ Enable alternate output function COUT63 for the PWM signal generated by T13

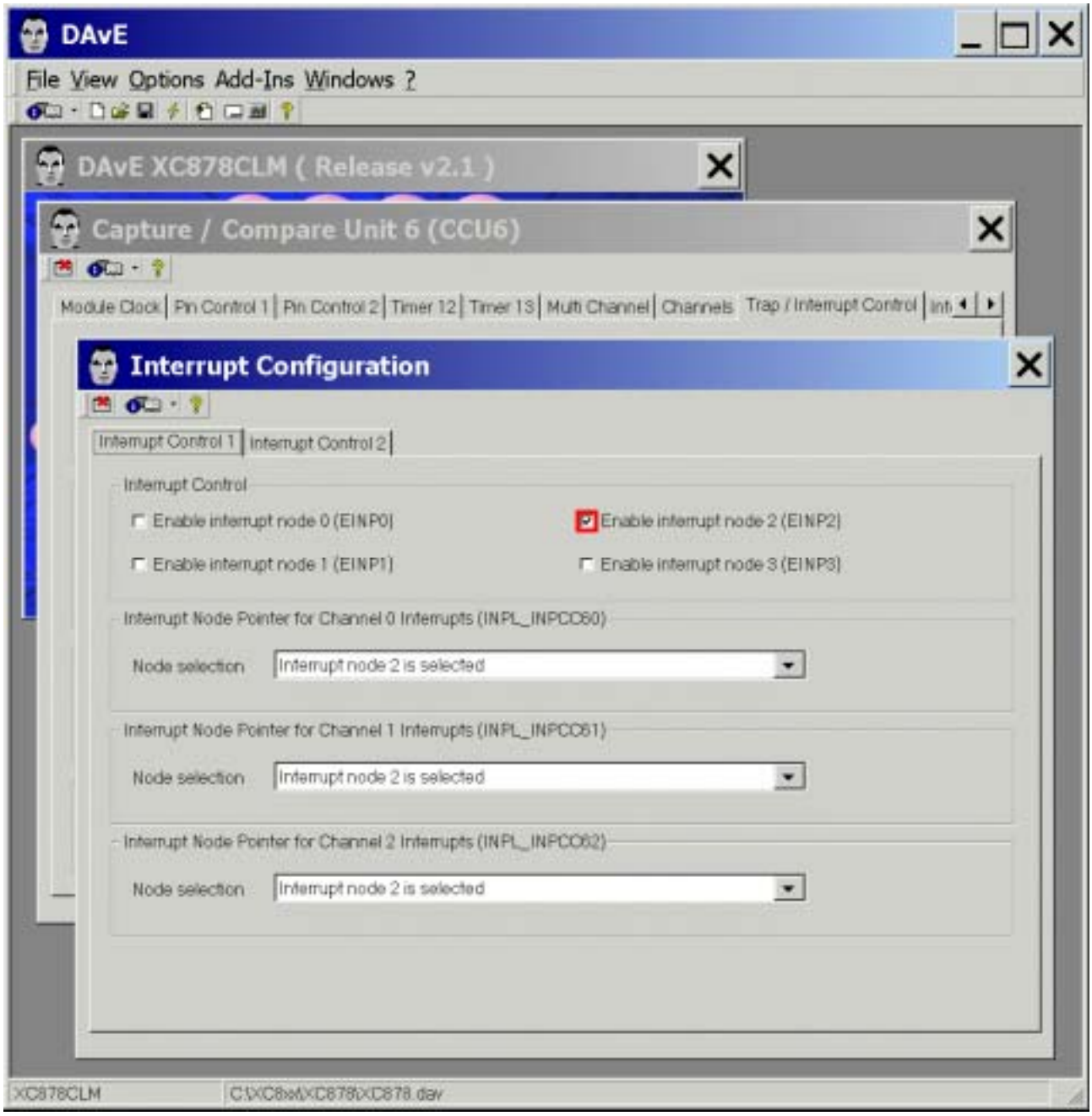


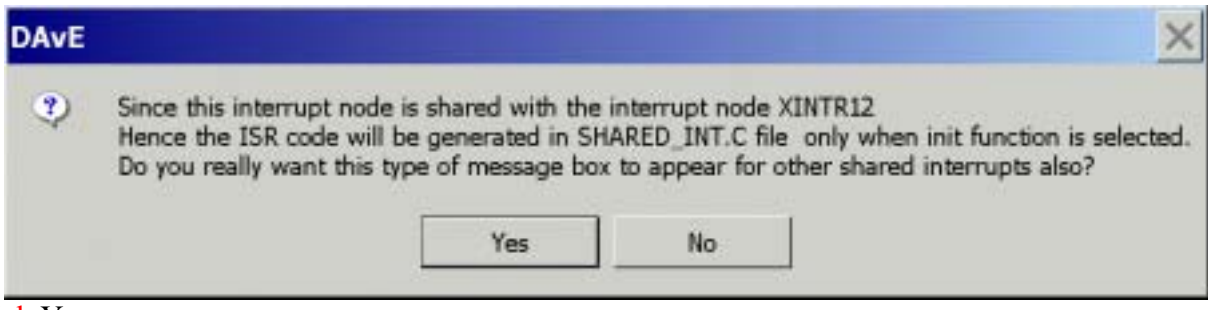
Exit this dialog now by clicking  the close button.

CCU6: Trap / Interrupt Control: click Interrupt Configuration



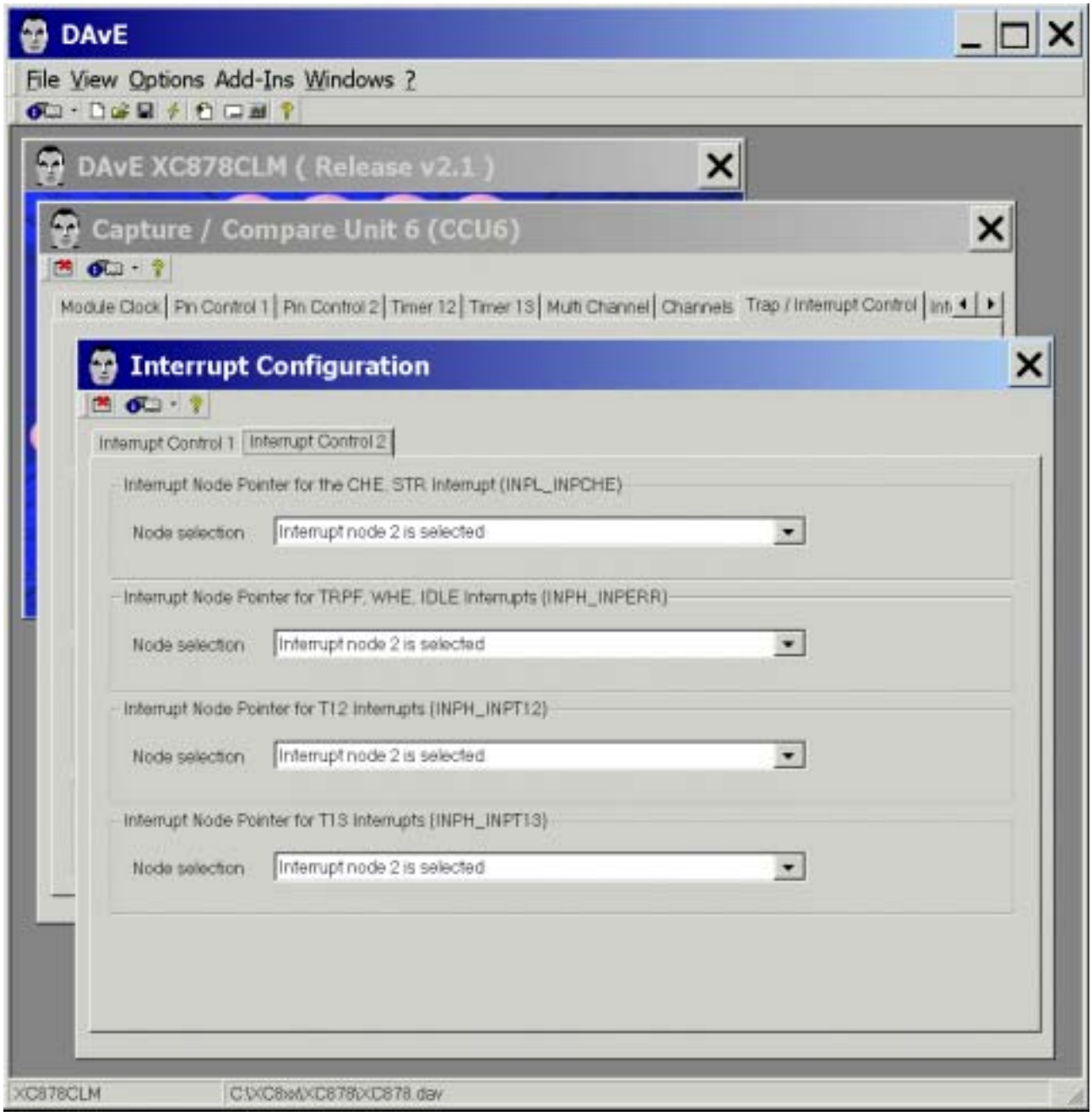
CCU6: Trap / Interrupt Control: Interrupt Configuration:
Interrupt Control 1: Interrupt Control: tick ☑ Enable interrupt node 2





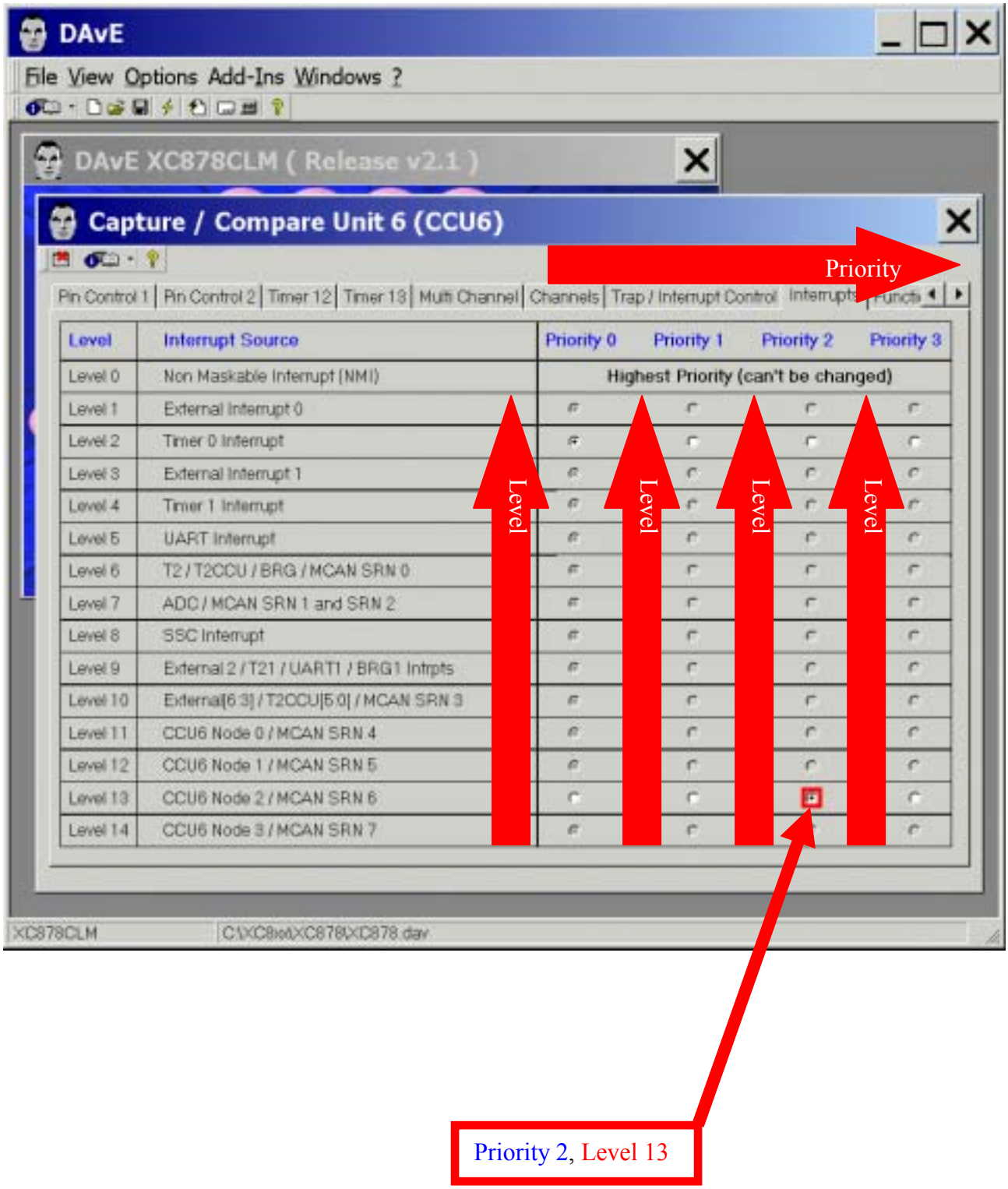
Click Yes

CCU6: Trap / Interrupt Control: Interrupt Configuration:
Interrupt Control 2: (do nothing)

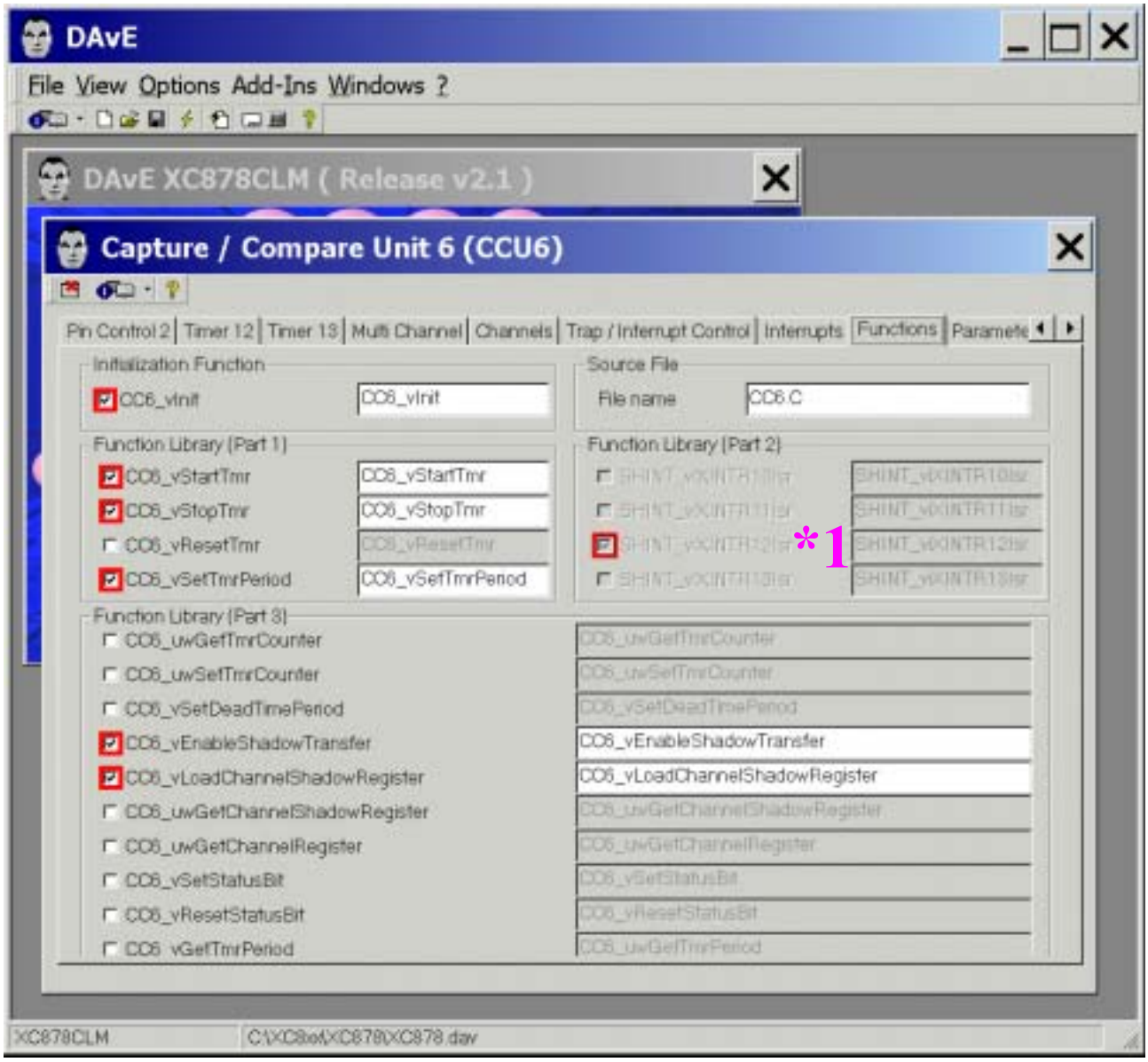


Exit this dialog now by clicking  the close button.

CCU6: Interrupts: change CCU6 Interrupt Node 2 from Priority 0, Level 13 to Priority 2, Level 13

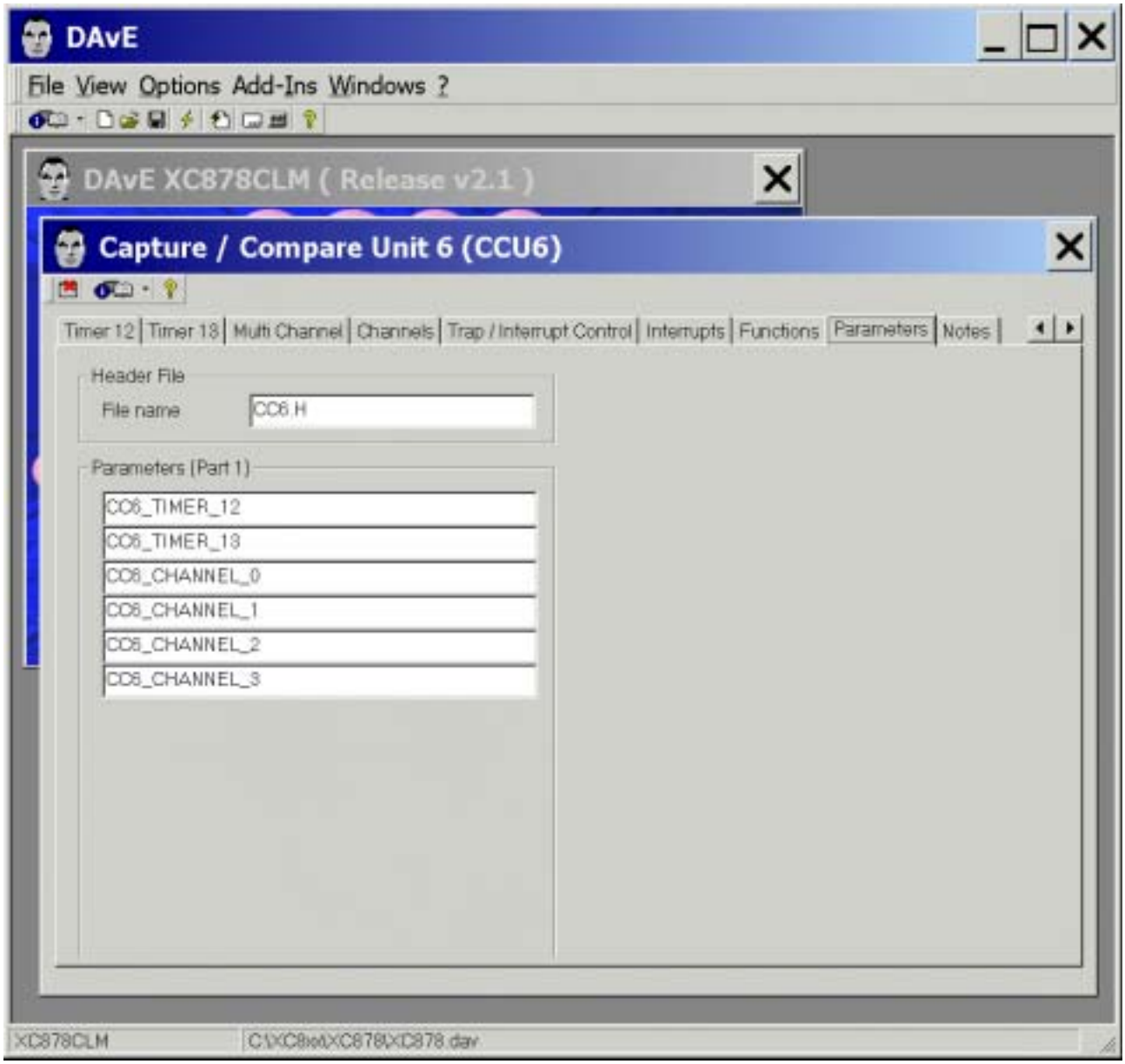


CCU6: Functions: Initialization Function: tick ☑ CCU6_vInit
CCU6: Functions: Function Library (Part 1): tick ☑ CCU6_vStartTmr
CCU6: Functions: Function Library (Part 1): tick ☑ CCU6_vStopTmr
CCU6: Functions: Function Library (Part 1): tick ☑ CCU6_vSetTmrPeriod
CCU6: Functions: Function Library (Part 3): tick ☑ CCU6_vEnableShadowTransfer
CCU6: Functions: Function Library (Part 3): tick ☑ CCU6_vLoadChannelShadowRegister

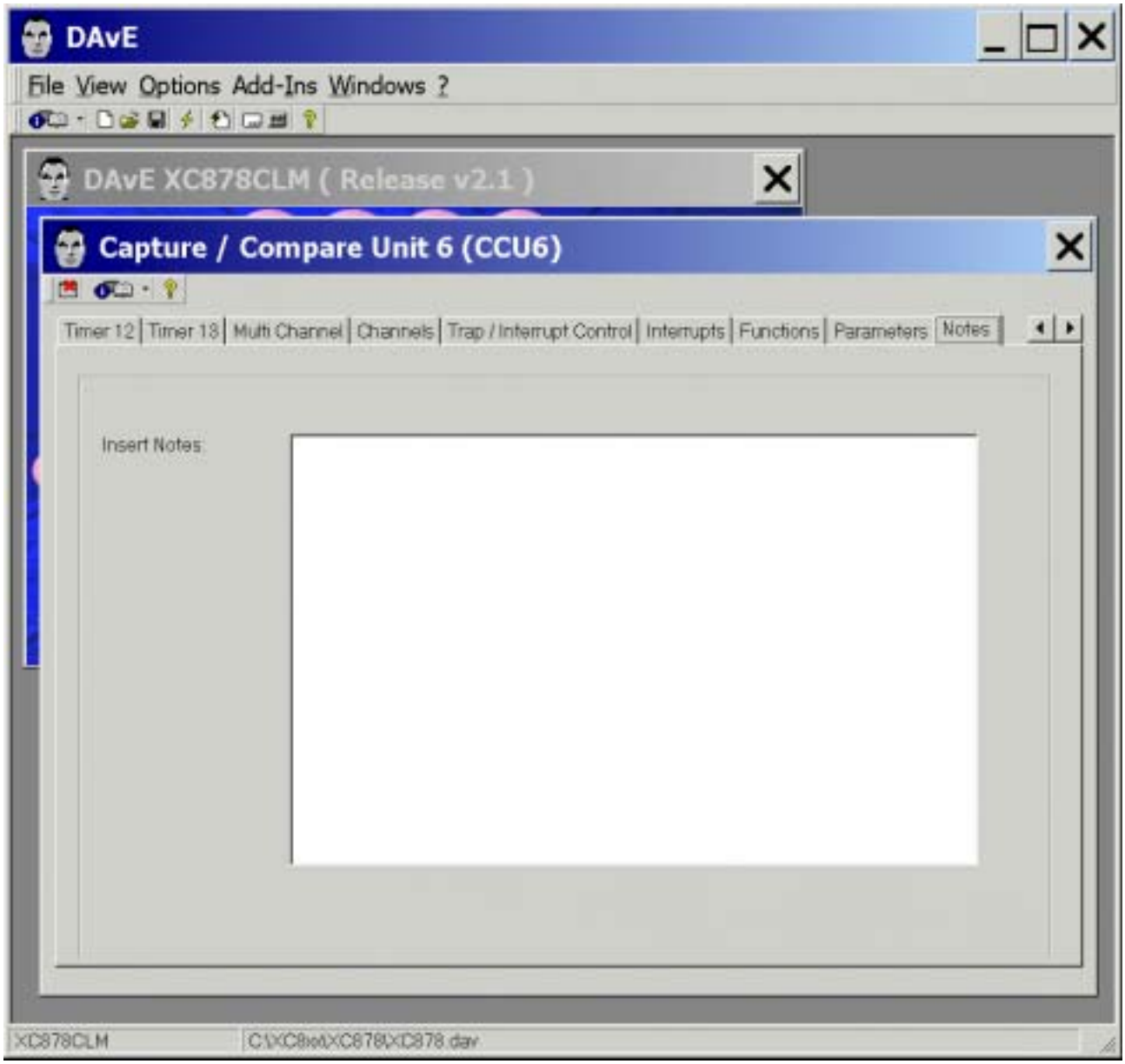


Note (*1):
The CCU6 ISR
void SHINT_viXINTR12Isr(void) interrupt XINTR12INT {}
will be generated in the SHARED_INT.C file.

CCU6: Parameters: (do nothing)



CCU6: Notes: If you wish, you can insert your comments here.

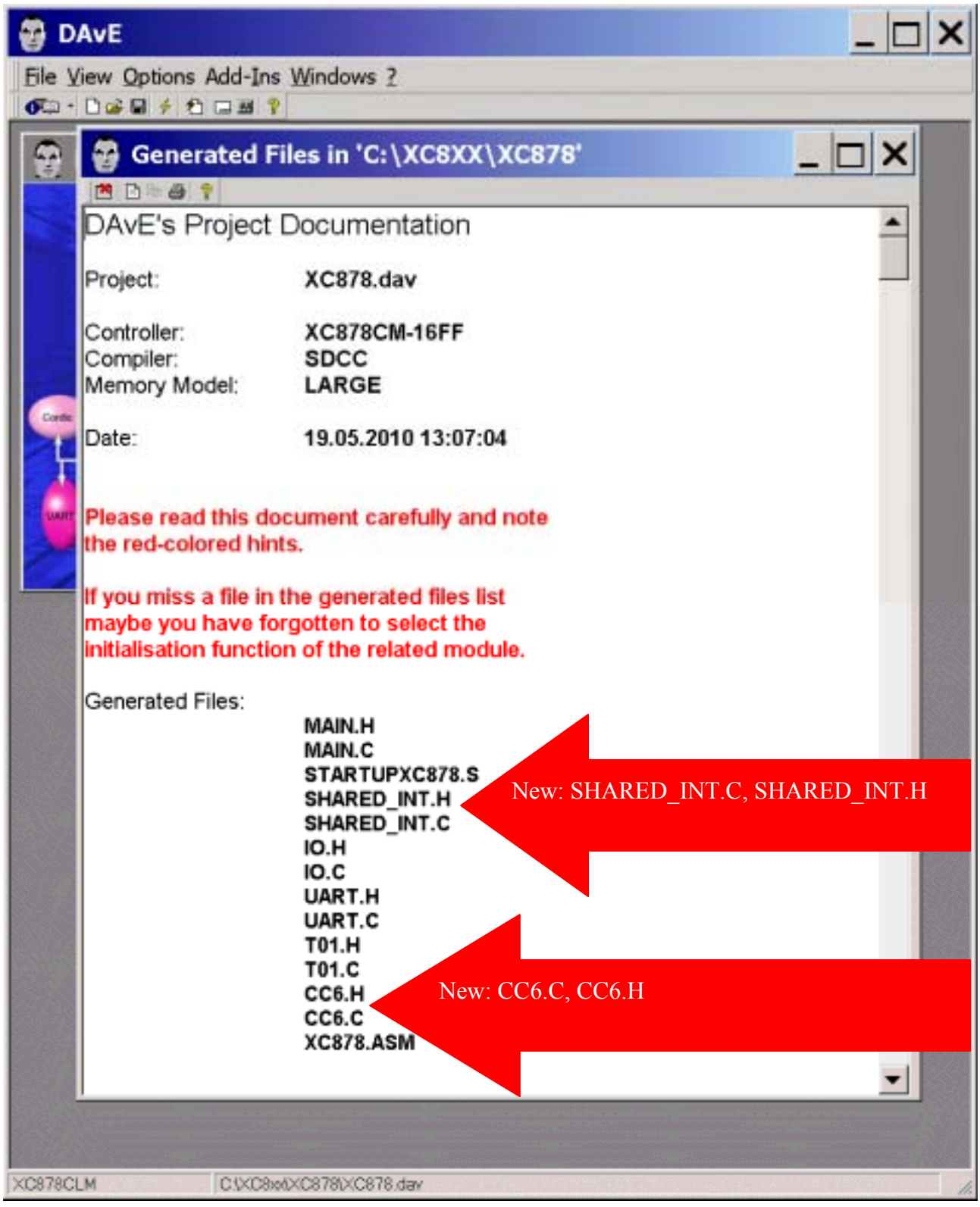


Exit this dialog now by clicking the close button.

Generate Code:

File - Generate Code	or	click 

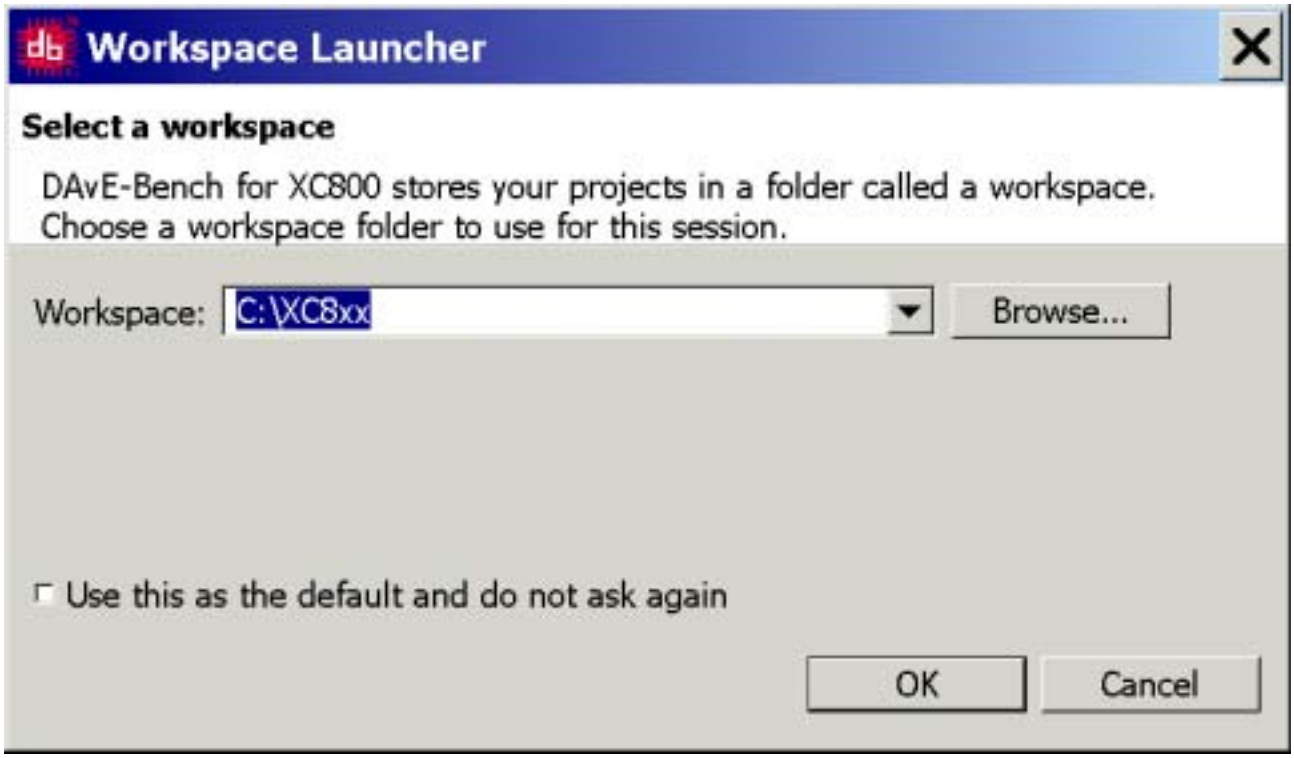
DAvE will show you all the files he has generated
(File Viewer opens automatically):



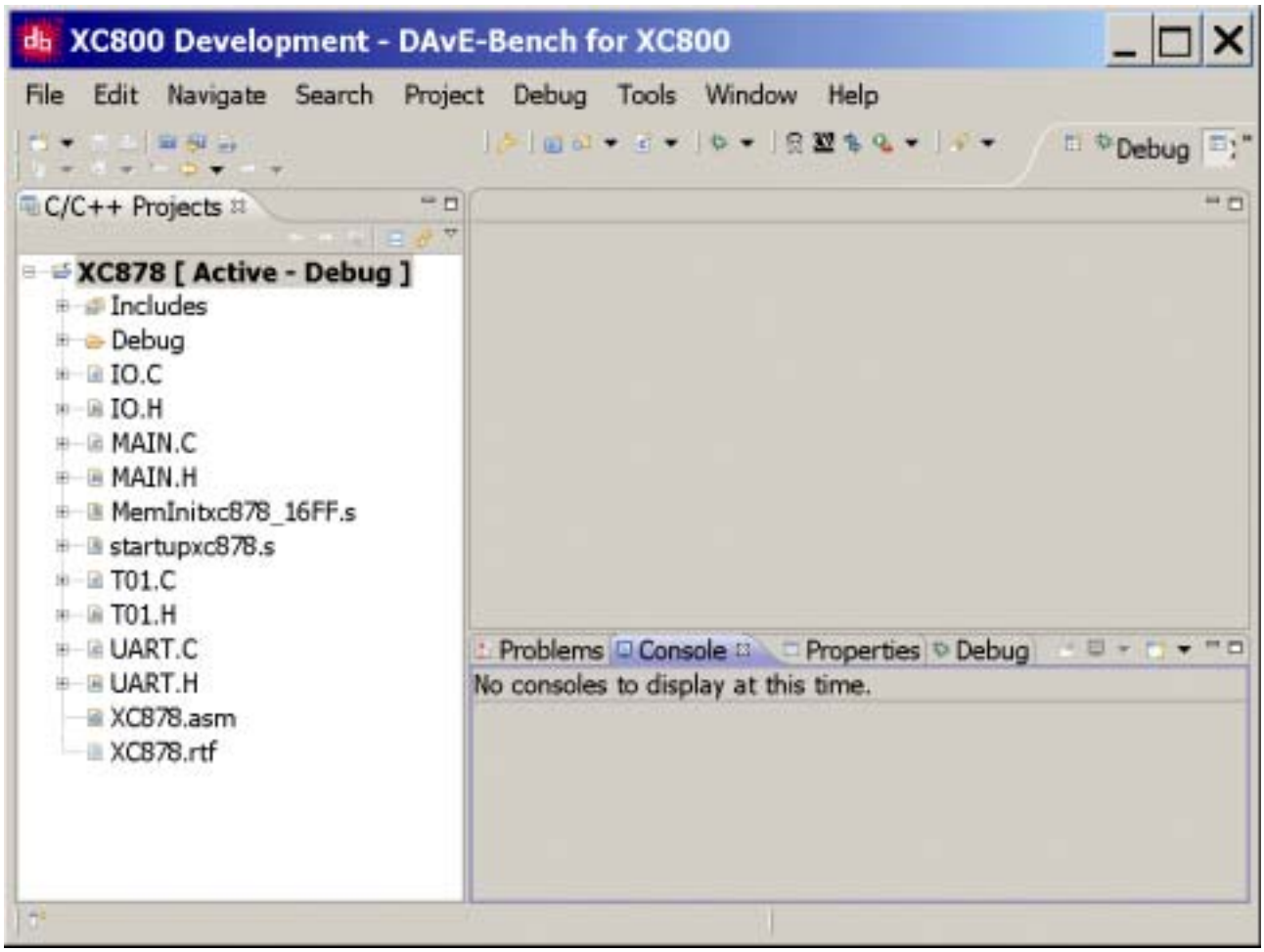
Close DAvE: File – Exit Save changes? click Yes

2.2) Open the DAvE Bench project and insert code:

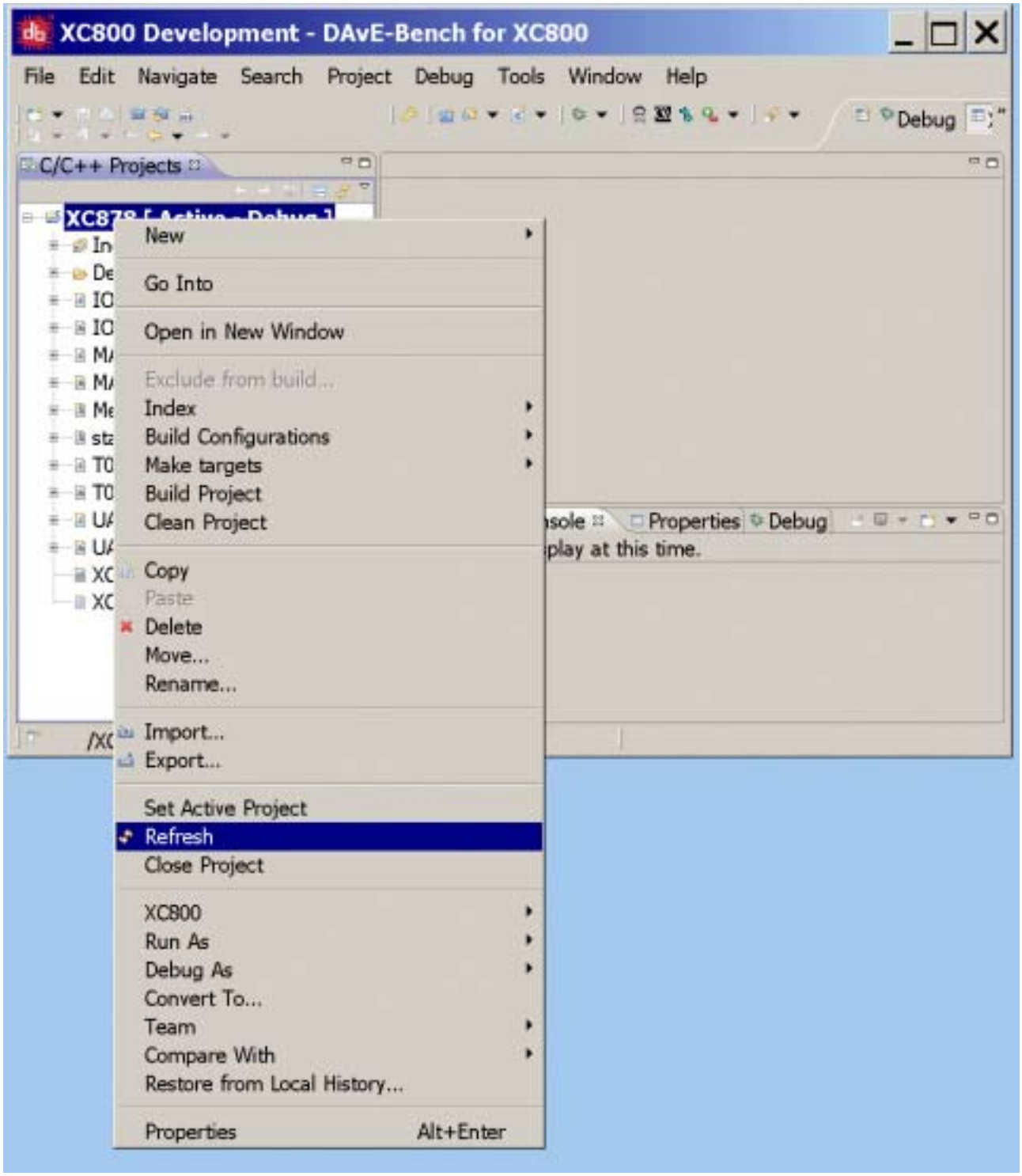
Start DAvE Bench and open your XC878 Project:

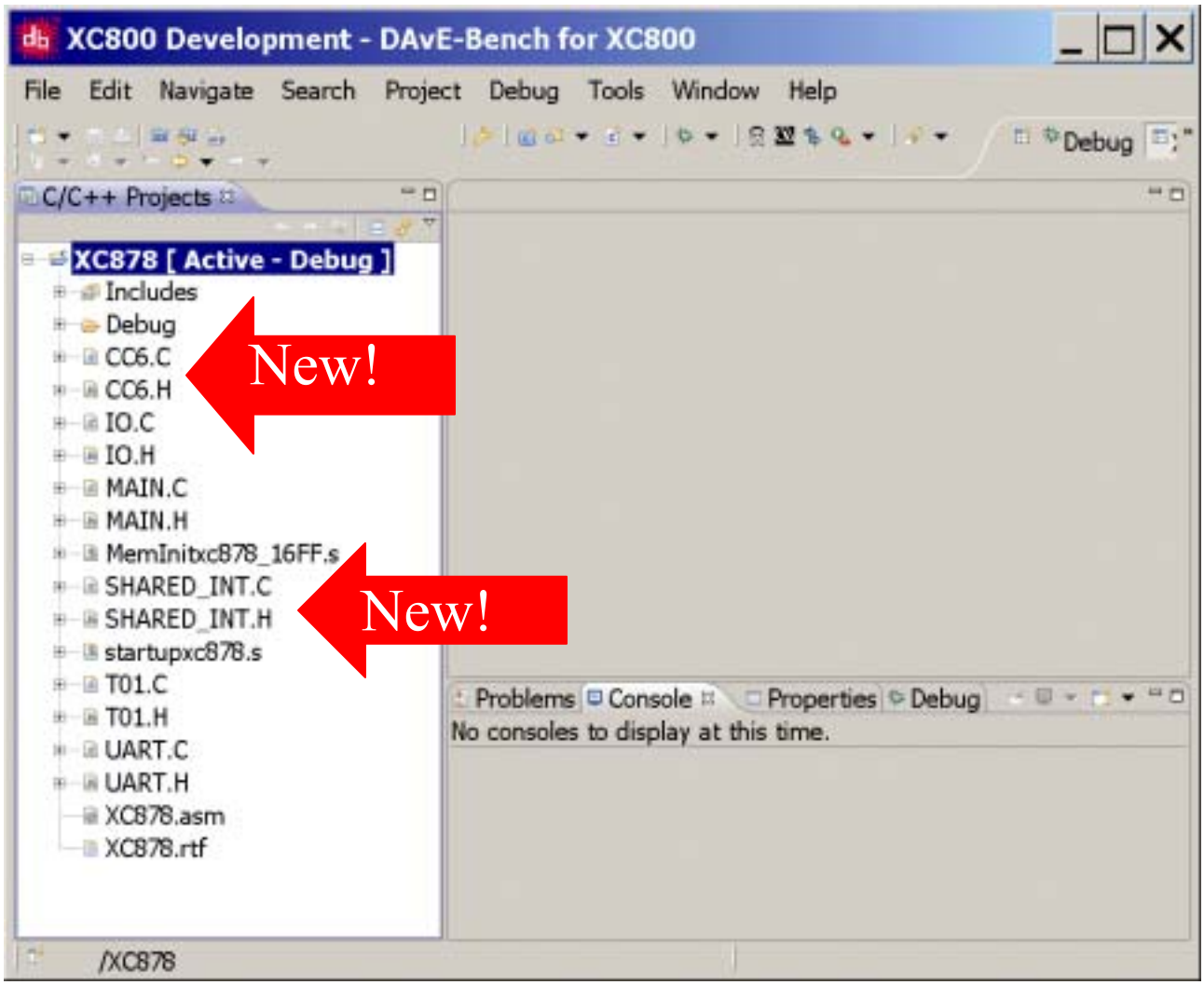


Click OK


XC800 Development - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
Debug
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
Problems Console Properties Debug
No consoles to display at this time.

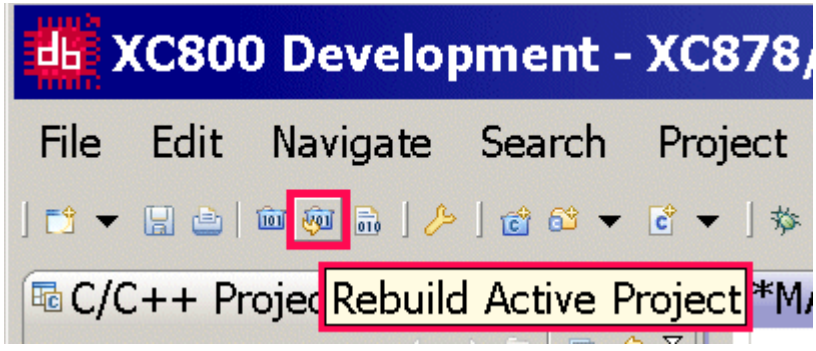
Mouse position: C/C++ Projects, XC878 [Active - Debug]: click right mouse button
click Refresh

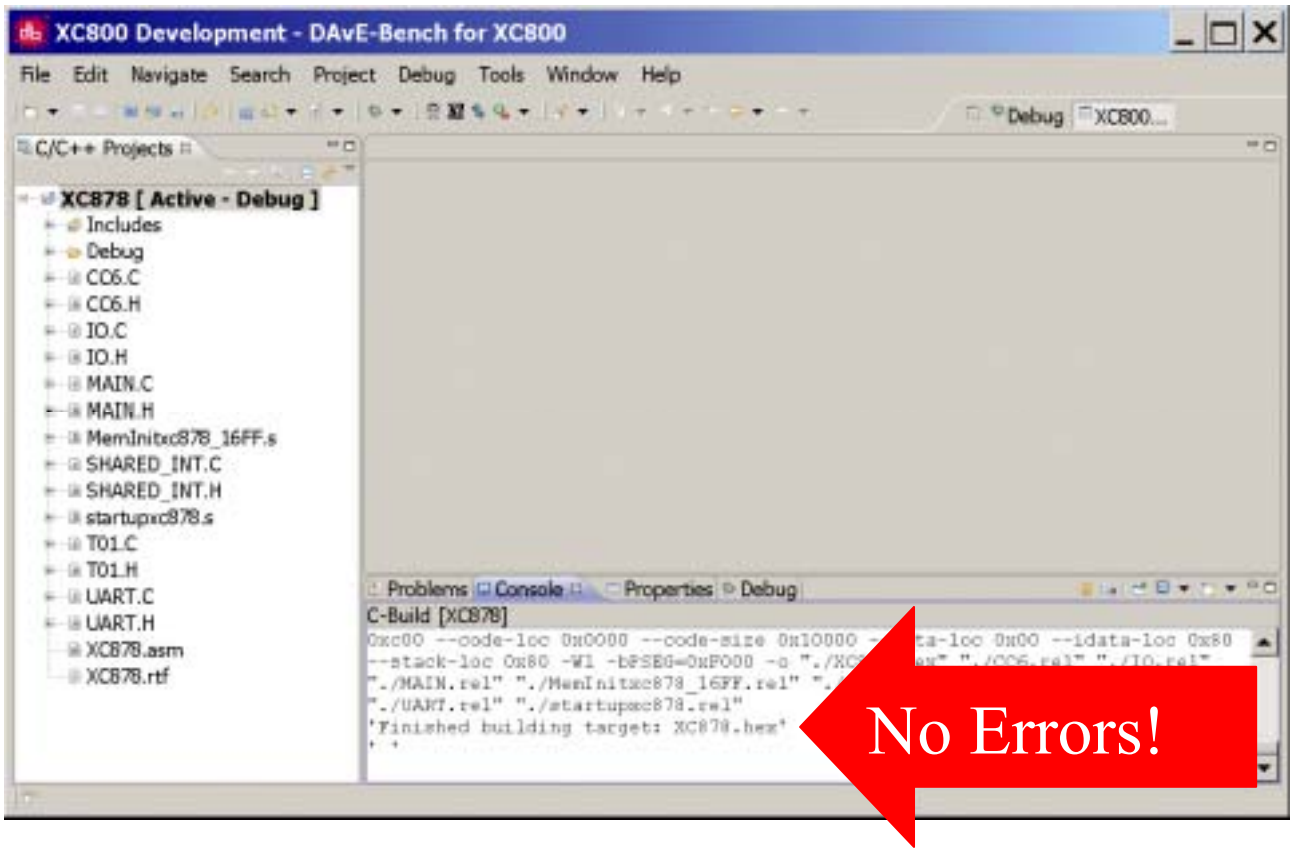



XC800 Development - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
Debug
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
New!
New!
Problems Console Properties Debug
No consoles to display at this time.
/XC878

Project – Rebuild Active Project

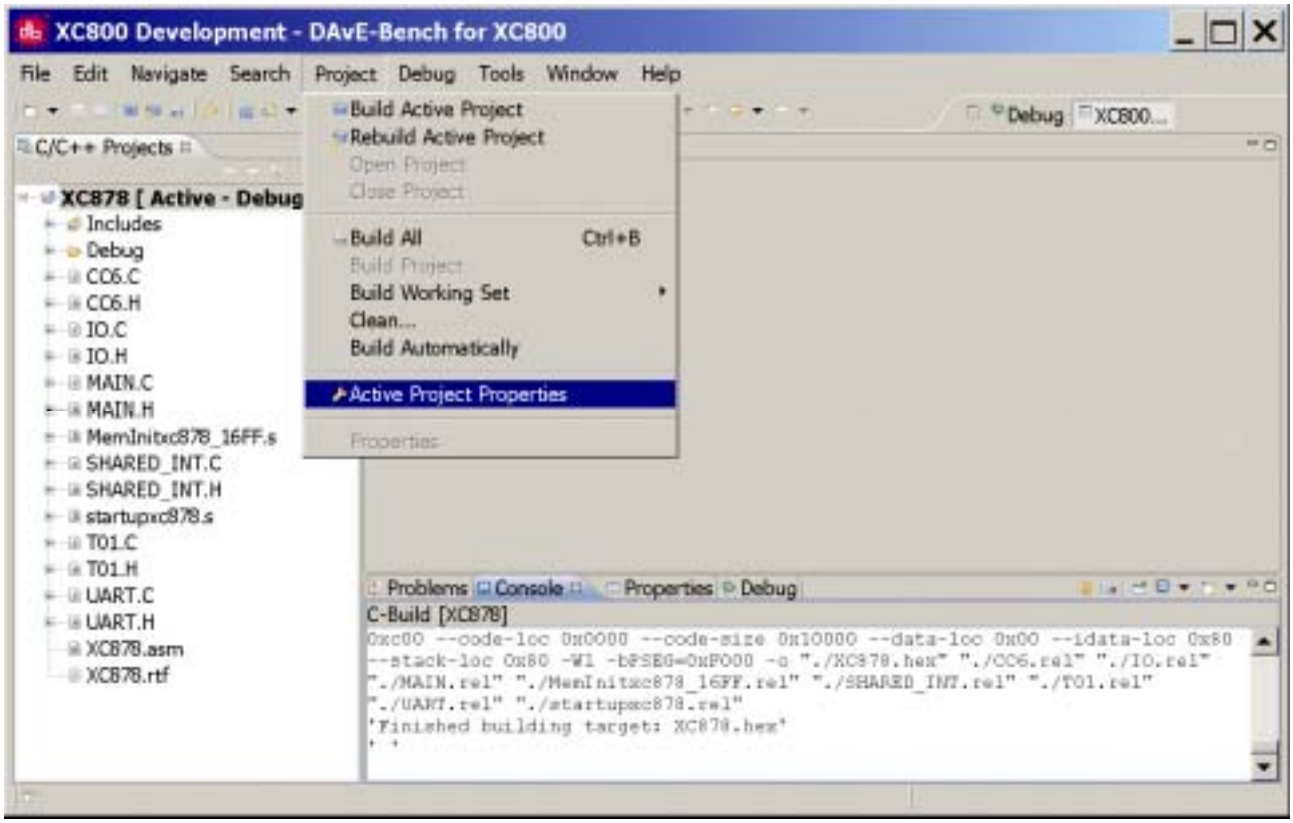
or: click 



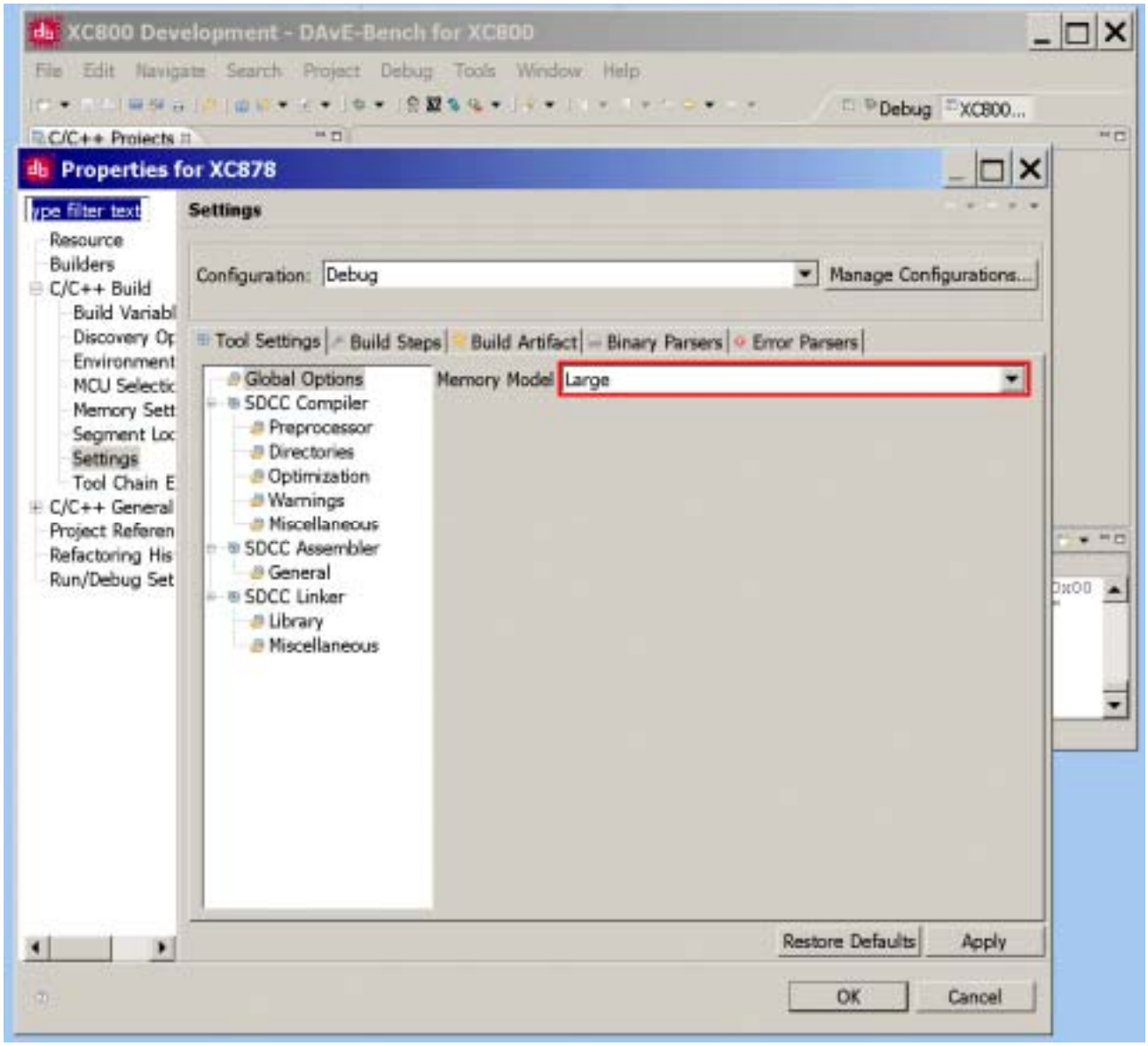

XC800 Development - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CO6.C
CO6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
Problems Console Properties Debug
C-Build [XC878]
"./MAIN.rel" "./MemInitxc878_16FF.rel"
"./UART.rel" "./startupxc878.rel"
'Finished building target: XC878.hex'
No Errors!

Check/Change the Memory Model:

Project – Active Project Properties



Tool Settings: Global Options: Memory Model: check Large



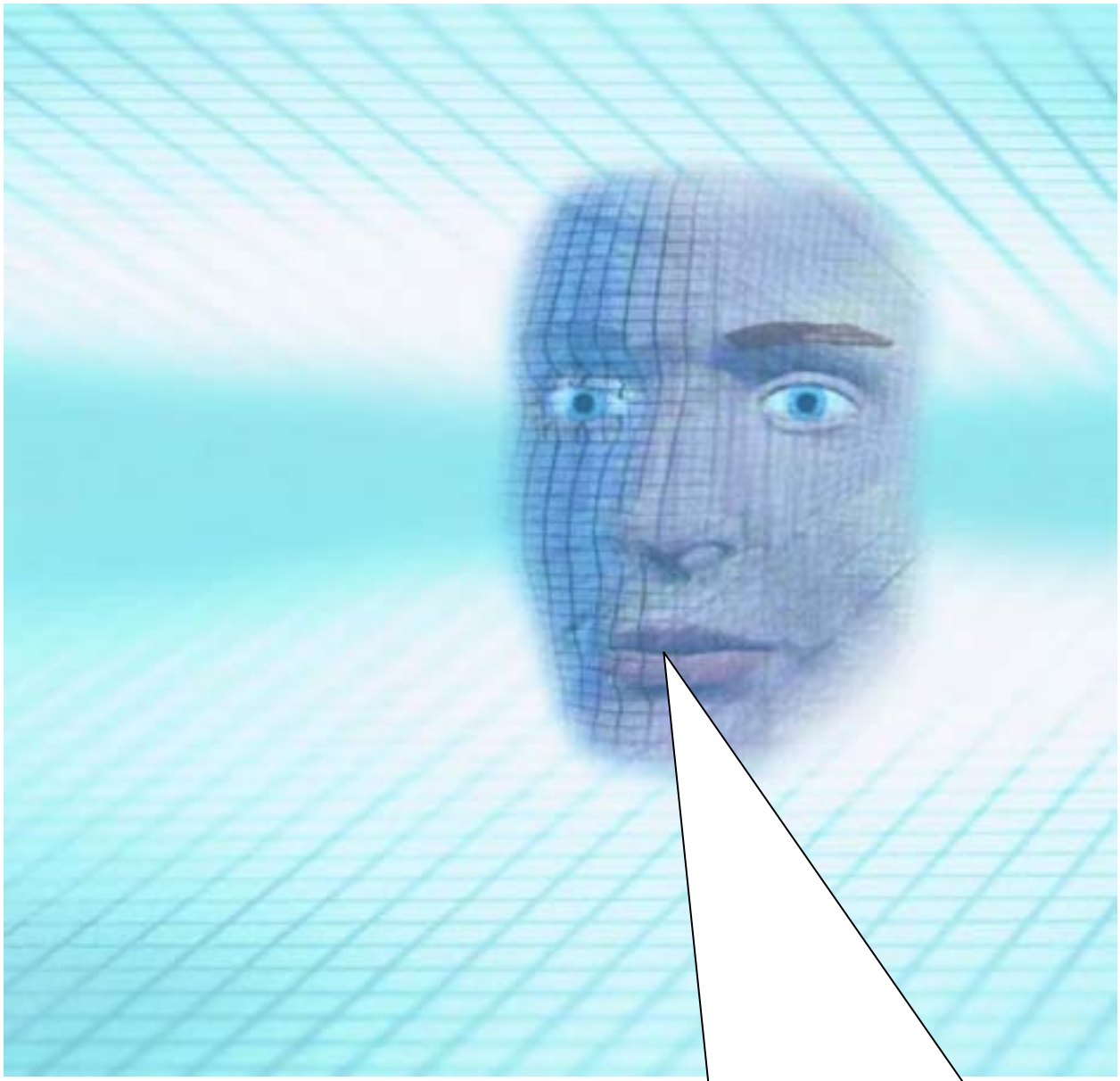
Click OK

Note:
(Release Notes; DAvE Bench for XC800; SDCC Tool chain for XC800):

Floats are enabled in large memory model only due to code size limitation.
So, print routines like printf(), sprintf() etc., will output <NO_FLOAT> when floats are used with these routines in small and medium memory models.

➔ Because we are going to use float variables in this programming example we have to use the Large Memory Model.

Insert your application specific program:



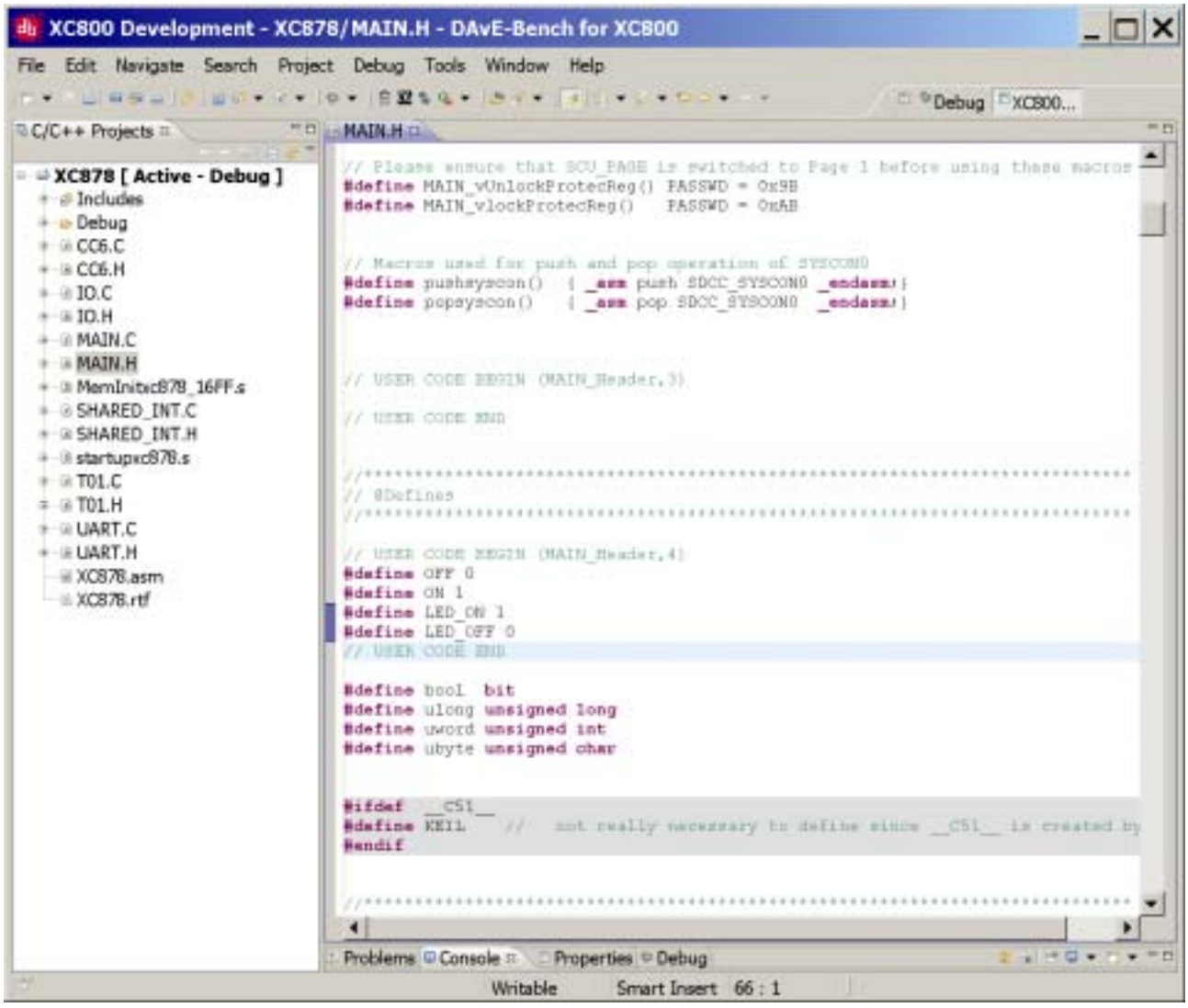
Note:
DAvE doesn't change code which is inserted between '// USER CODE BEGIN' and '// USER CODE END'. Therefore, whenever adding code to DAvE's generated code, write it between '// USER CODE BEGIN' and '// USER CODE END'.
If you wish to change DAvE´s generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAvE regenerate code!

Double click Main.h and change the Defines from:

```
#define OFF 0
#define ON 1
#define LED_ON 0xFF
#define LED_OFF 0x00
```

to:

```
#define OFF 0
#define ON 1
#define LED_ON 1
#define LED_OFF 0
```

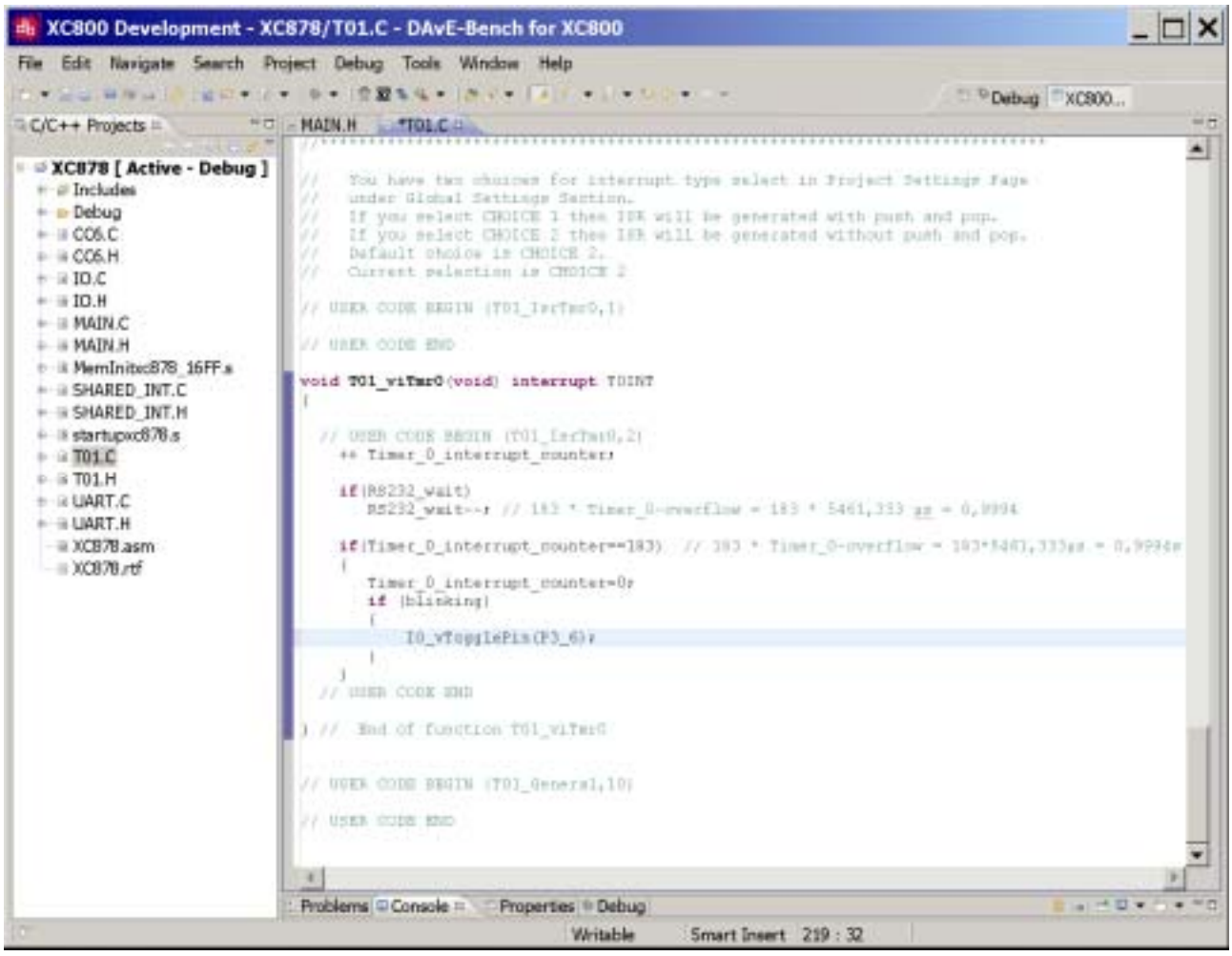


Double click T01.C and change code from:

```
++ Timer_0_interrupt_counter;

if(RS232_wait)
        RS232_wait--; // 183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994

if(Timer_0_interrupt_counter==183)  // 183 * Timer_0-overflow = 183*5461,333µs = 0,9994s
{
      Timer_0_interrupt_counter=0;
      if (blinking)
      {
          P3_DATA = P3_DATA^0xFF;
       }
 }
```

to:

```
++ Timer_0_interrupt_counter;

if(RS232_wait)
        RS232_wait--; // 183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994

if(Timer_0_interrupt_counter==183)  // 183 * Timer_0-overflow = 183*5461,333µs = 0,9994s
{
      Timer_0_interrupt_counter=0;
      if (blinking)
      {
          IO_vTogglePin(P3_6);
       }
 }
```



The LED on IO_Port_3.6 will be blinking with a frequency of about 1 second.

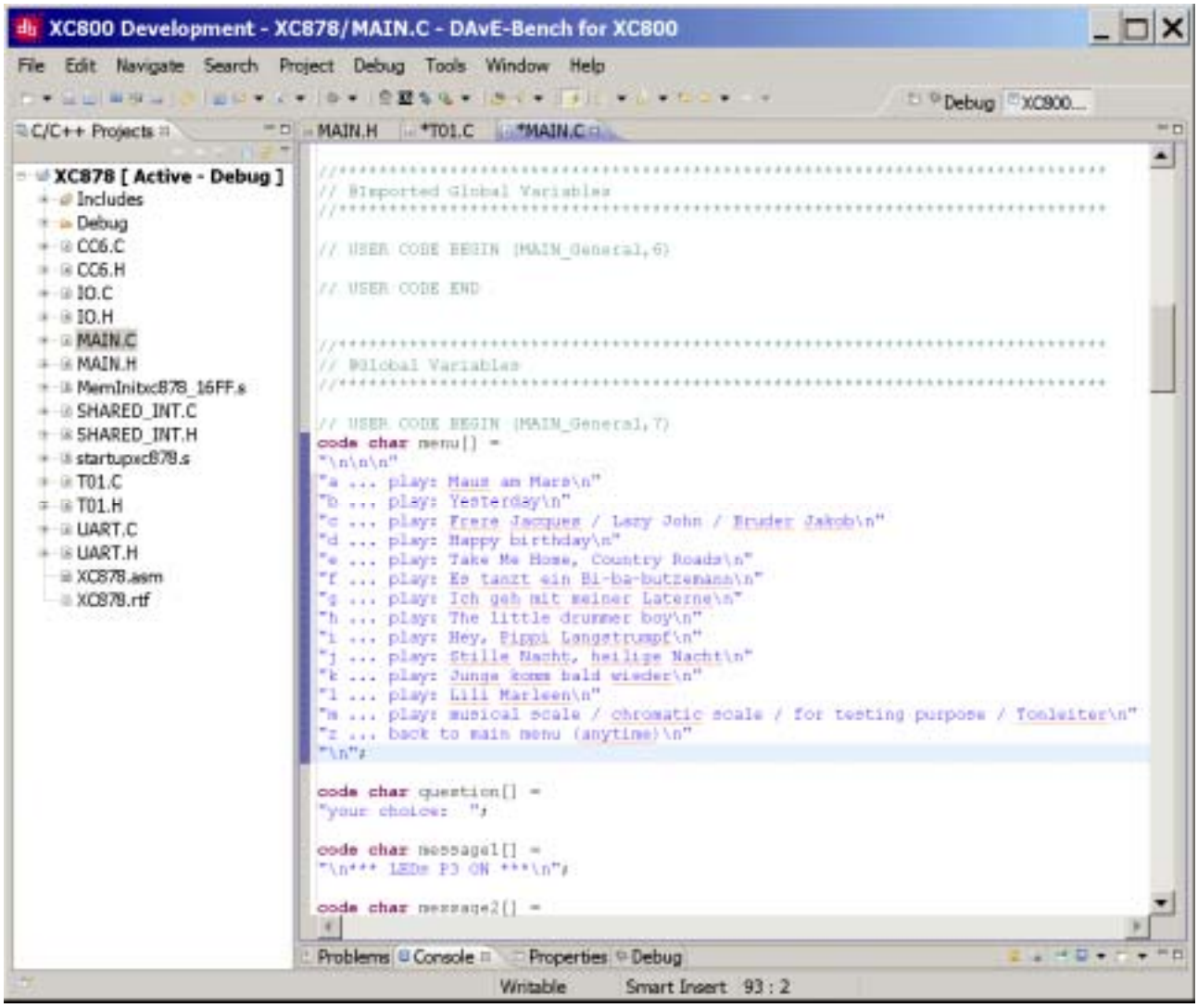
Double click MAIN.C and change Global Variable menu

from:

```
code char menu[] =
"\n\n\n"
"1 ... LEDs P3 ON\n"
"2 ... LEDs P3 OFF\n"
"3 ... LEDs P3 blinking\n"
"\n";
```

to:

```
code 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 / Tonleiter\n"
"z ... back to main menu (anytime)\n"
"\n";
```

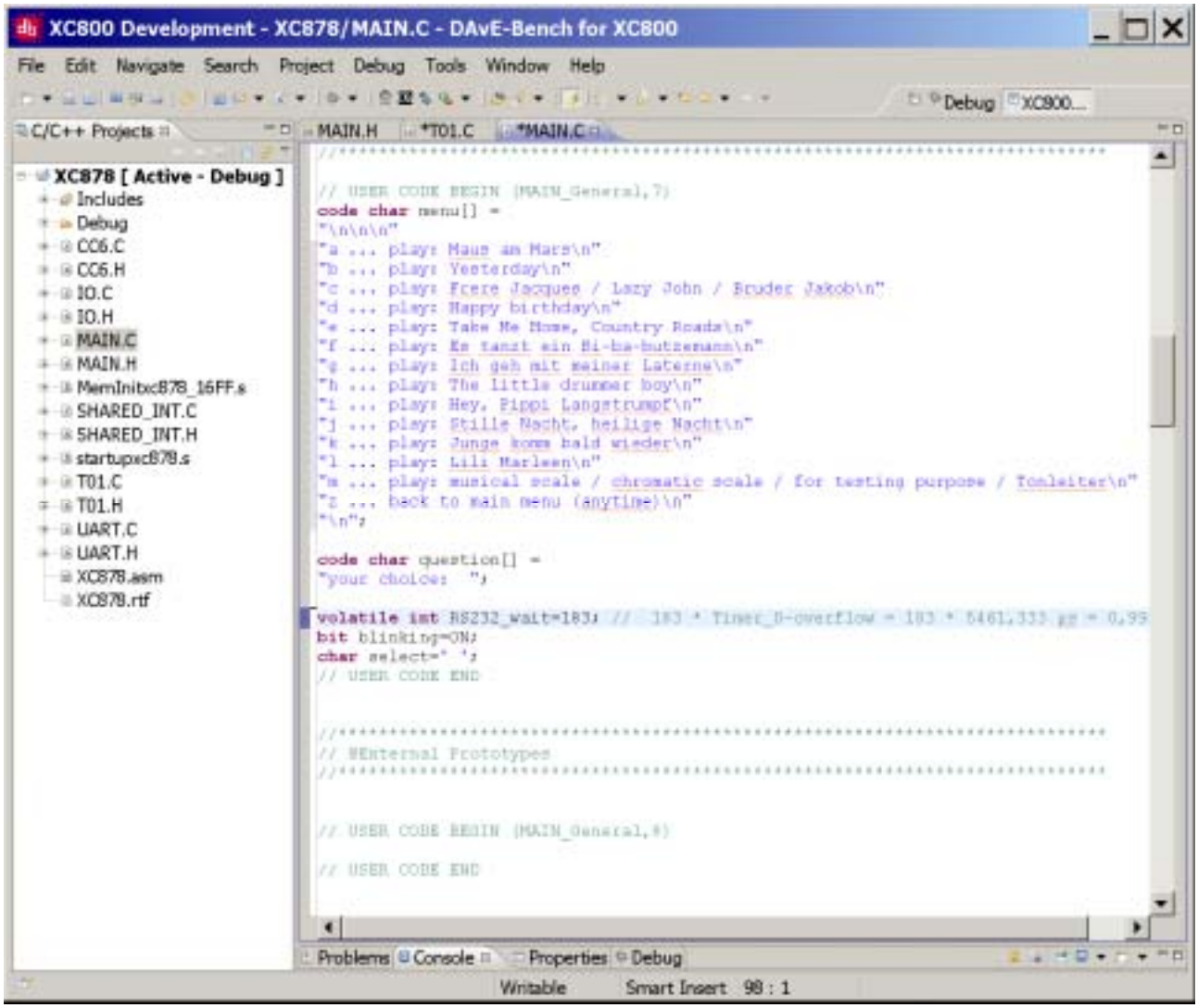

XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.H
*T01.C
*MAIN.C
// Exported Global Variables
// USER CODE BEGIN (MAIN_General,6)
// USER CODE END
// Global Variables
// USER CODE BEGIN (MAIN_General,7)
code 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 / Tonleiter\n"
"z ... back to main menu (anytime)\n"
"\n";
code char question[] =
"your choice: ";
code char message1[] =
"\n*** LEDs P3 ON ***\n";
code char message2[] =
Problems Console Properties Debug
Writable Smart Insert 93 : 2

Double click MAIN.C and delete Global Variables message1, message2 and message3

from:

```
code char message1[] =
"\n*** LEDs P3 ON ***\n";

code char message2[] =
"\n*** LEDs P3 OFF ***\n";

code char message3[] =
"\n*** LEDs P3 BLINKING ***\n";
```

to:



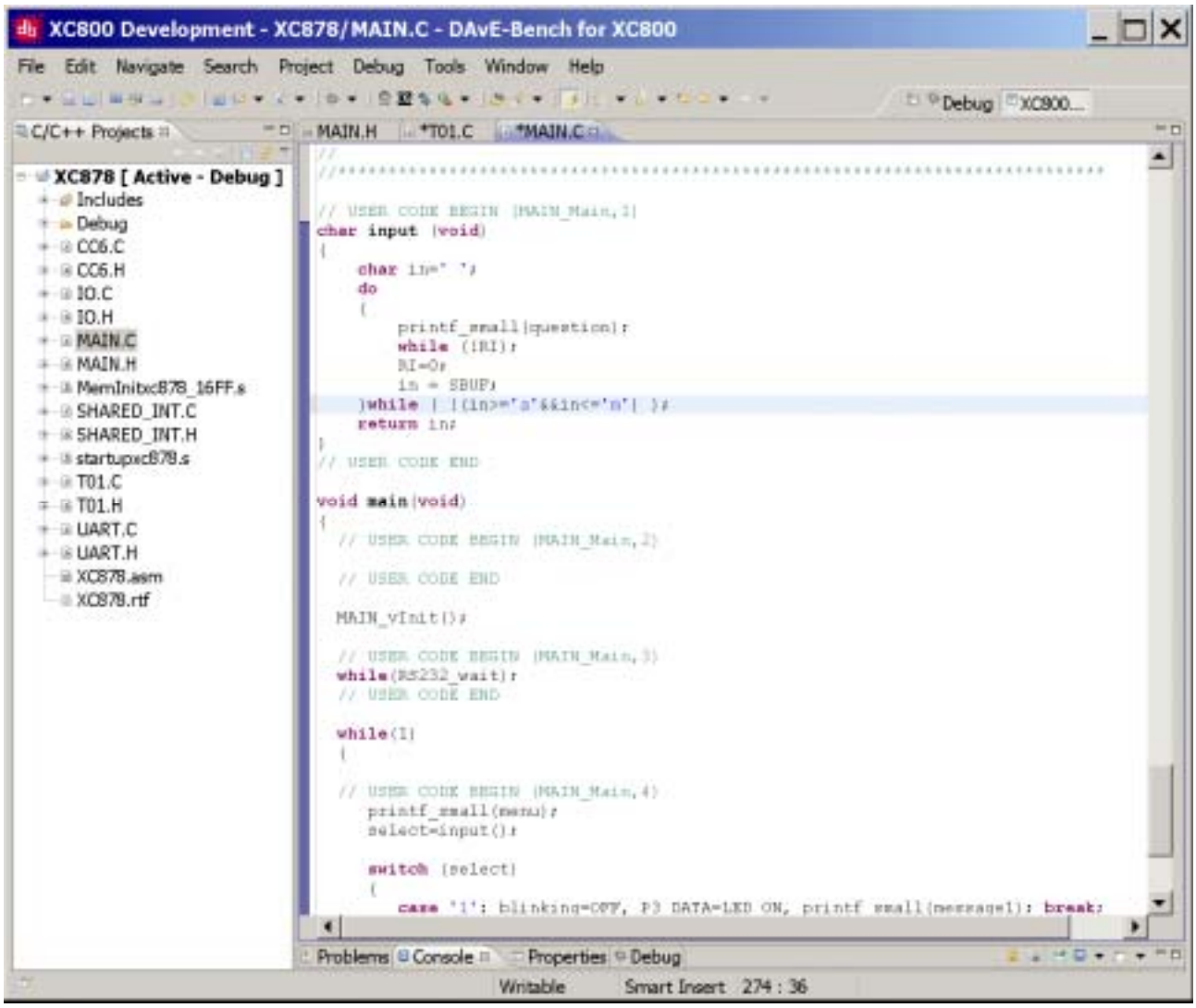
Double click MAIN.C and change function ”char input (void)”:

from

```
char input (void)
{
        char in=' ';
        do
        {
                printf_small(question);
                while (!RI);
                RI=0;
                in = SBUF;
        }while (in!='1' && in!= '2' && in != '3');
        return in;
}
```

to

```
char input (void)
{
        char in=' ';
        do
        {
                printf(question);
                while (!RI);
                RI=0;
                in = SBUF;
        }while ( !(in>='a'&&in<='m') );
        return in;
}
```


XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.H
*T01.C
*MAIN.C
Problems
Console
Properties
Debug
Writable
Smart Insert
274 : 36

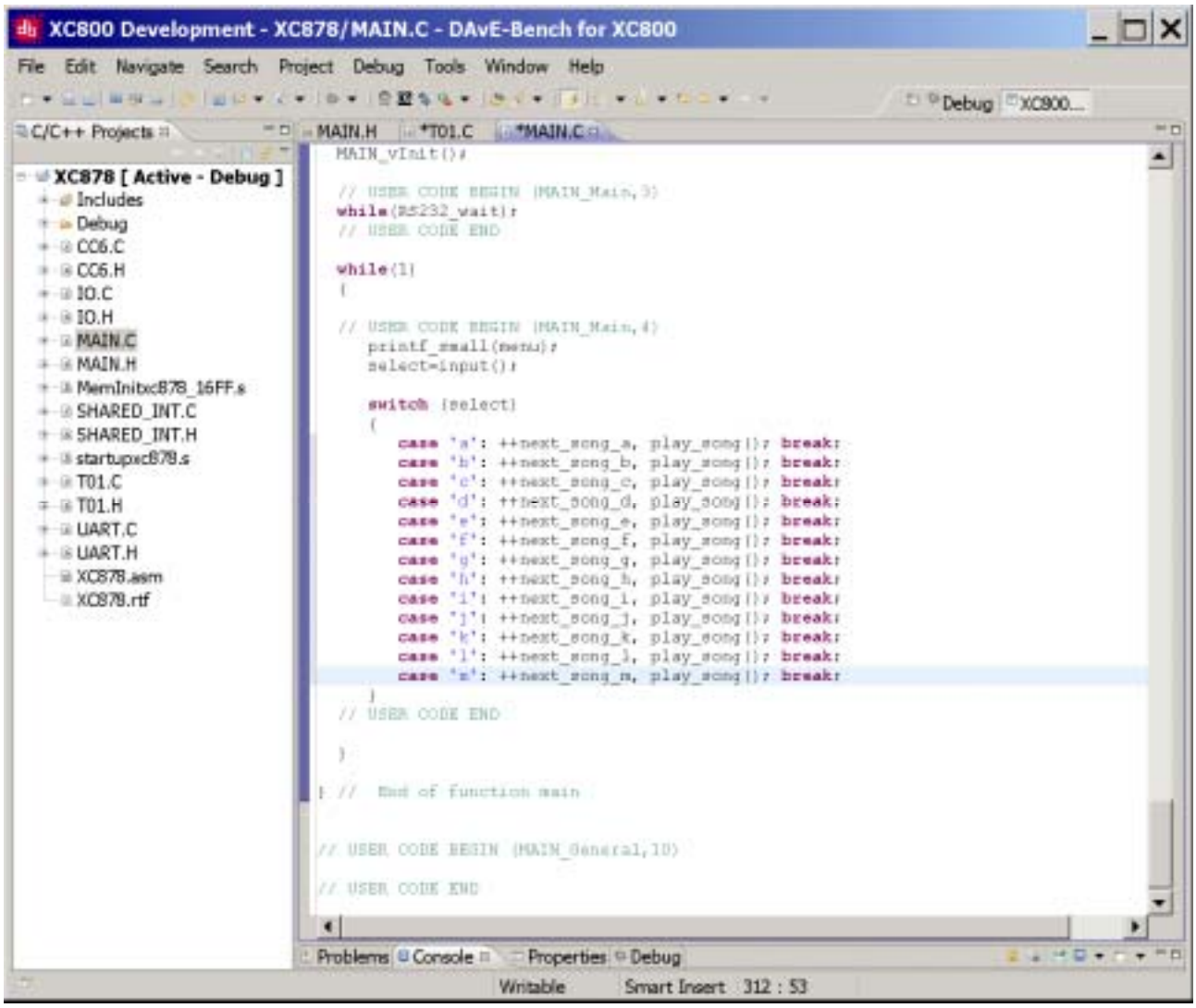
Double click MAIN.C and extend/change [“switch/case” in void main (void)] :

from:

```
switch (select)
{
  case '1': blinking=OFF, P3_DATA=LED_ON, printf_small(message1); break;
  case '2': blinking=OFF, P3_DATA=LED_OFF, printf_small(message2); break;
  case '3': blinking=ON, printf_small(message3); break;
}
```

to:

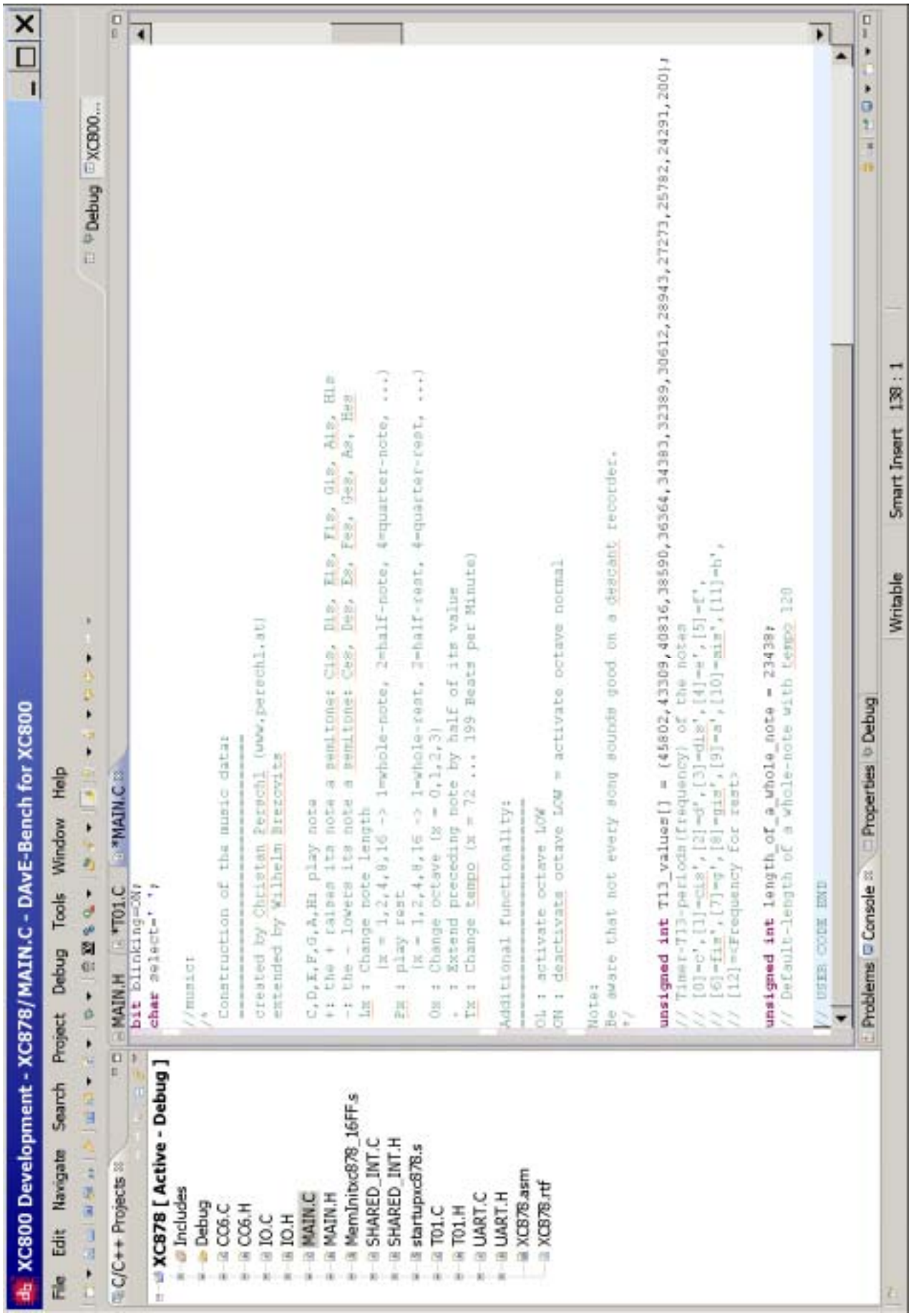
```
switch (select)
{
   case 'a': ++next_song_a, play_song(); break;
   case 'b': ++next_song_b, play_song(); break;
   case 'c': ++next_song_c, play_song(); break;
   case 'd': ++next_song_d, play_song(); break;
   case 'e': ++next_song_e, play_song(); break;
   case 'f': ++next_song_f, play_song(); 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;
}
```


XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.H
*T01.C
*MAIN.C
MAIN_vInit();
// USER CODE BEGIN (MAIN_Main,3)
while(RS232_wait);
// USER CODE END
while(1)
{
// USER CODE BEGIN (MAIN_Main,4)
printf_small(menu);
select=input();
switch (select)
{
case 'a': ++next_song_a, play_song();
case 'b': ++next_song_b, play_song();
case 'c': ++next_song_c, play_song();
case 'd': ++next_song_d, play_song();
case 'e': ++next_song_e, play_song();
case 'f': ++next_song_f, play_song();
case 'g': ++next_song_g, play_song();
case 'h': ++next_song_h, play_song();
case 'i': ++next_song_i, play_song();
case 'j': ++next_song_j, play_song();
case 'k': ++next_song_k, play_song();
case 'l': ++next_song_l, play_song();
case 'm': ++next_song_m, play_song();
}
// USER CODE END
}
} // End of function main
// USER CODE BEGIN (MAIN_General,10)
// USER CODE END
Problems Console Properties Debug
Writable
Smart Insert 312 : 53

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 = 72 ... 199 Beats per Minute)

Additional functionality:
=========================
OL : activate octave LOW
ON : deactivate octave LOW = activate octave normal

Note:
Be aware that not every song sounds good on a descant recorder.
*/

unsigned int T13_values[] =
{45802,43309,40816,38590,36364,34383,32389,30612,28943,27273,25782,24291,200};
// Timer-T13-periods(frequency) 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 = 23438;
// Default-length of a whole-note with tempo 120
```

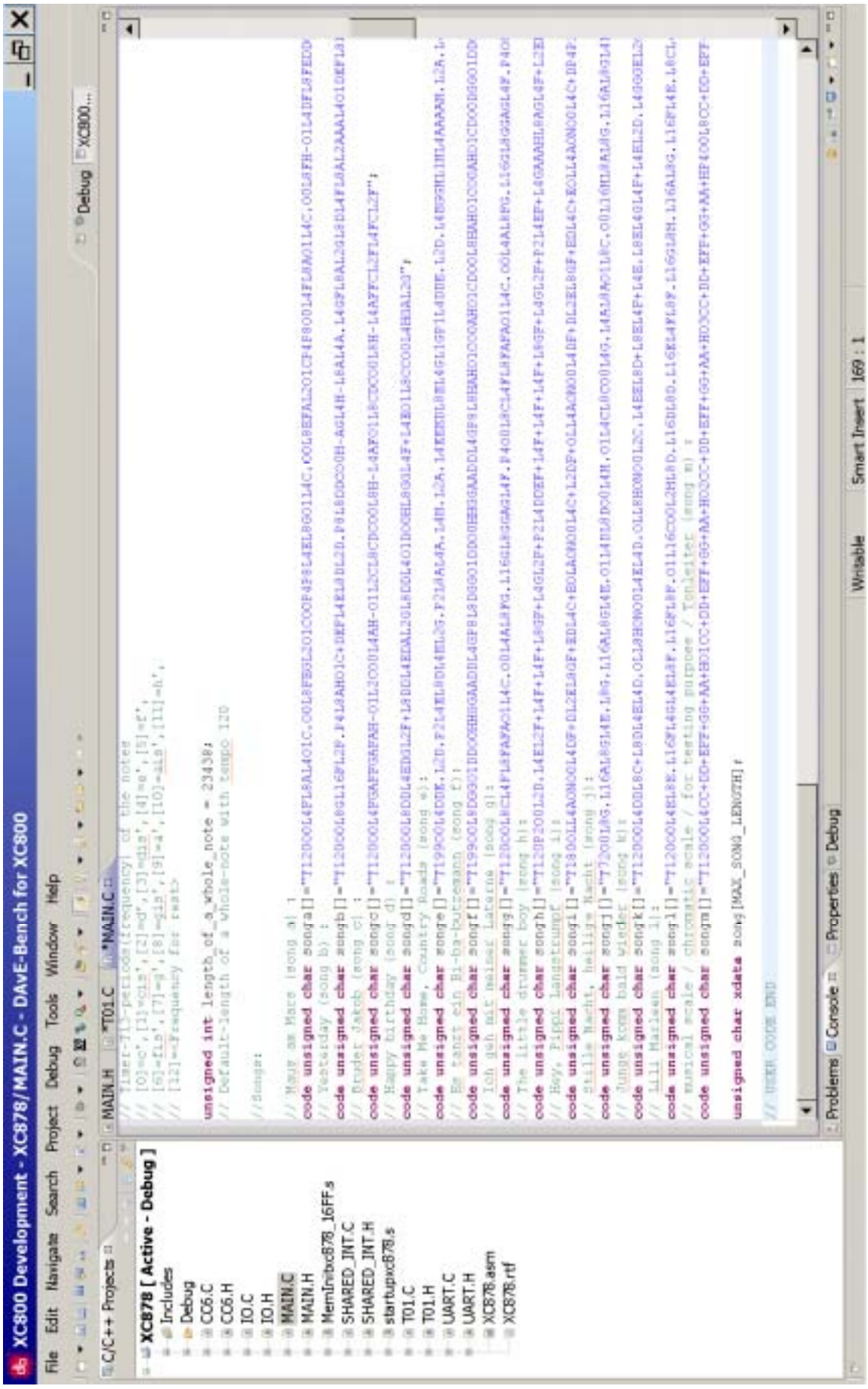
Note:
The notes C,D,E,F,G,A,H are named C,D,E,F,G,A,B in other countries.
In this document we stick to the German names.


XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.H
*T01.C
*MAIN.C
bit blinking=0;
char select='';
//music
/*
Construction of the music data:
created by Christian Zeppel (www.zeppel.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 = 72 ... 199 Beats per Minute)
Additional functionality:
OL : activate octave LOW
OH : deactivate octave LOW = activate octave normal
Note:
Be aware that not every song sounds good on a descant recorder.
*/
unsigned int T13_values[] = {45802,43309,40816,38590,36364,34383,32389,30612,28943,27273,25782,24291,200};
// Timer-T13-Periods(frequency) 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 = 23438;
// Default-length of a whole-note with tempo 120
// USER CODE END
Problems Console Properties Debug
Writable
Smart Insert 138 : 1
Debug
XC800...

Double click MAIN.C and insert Global Variables (”songstrings”):

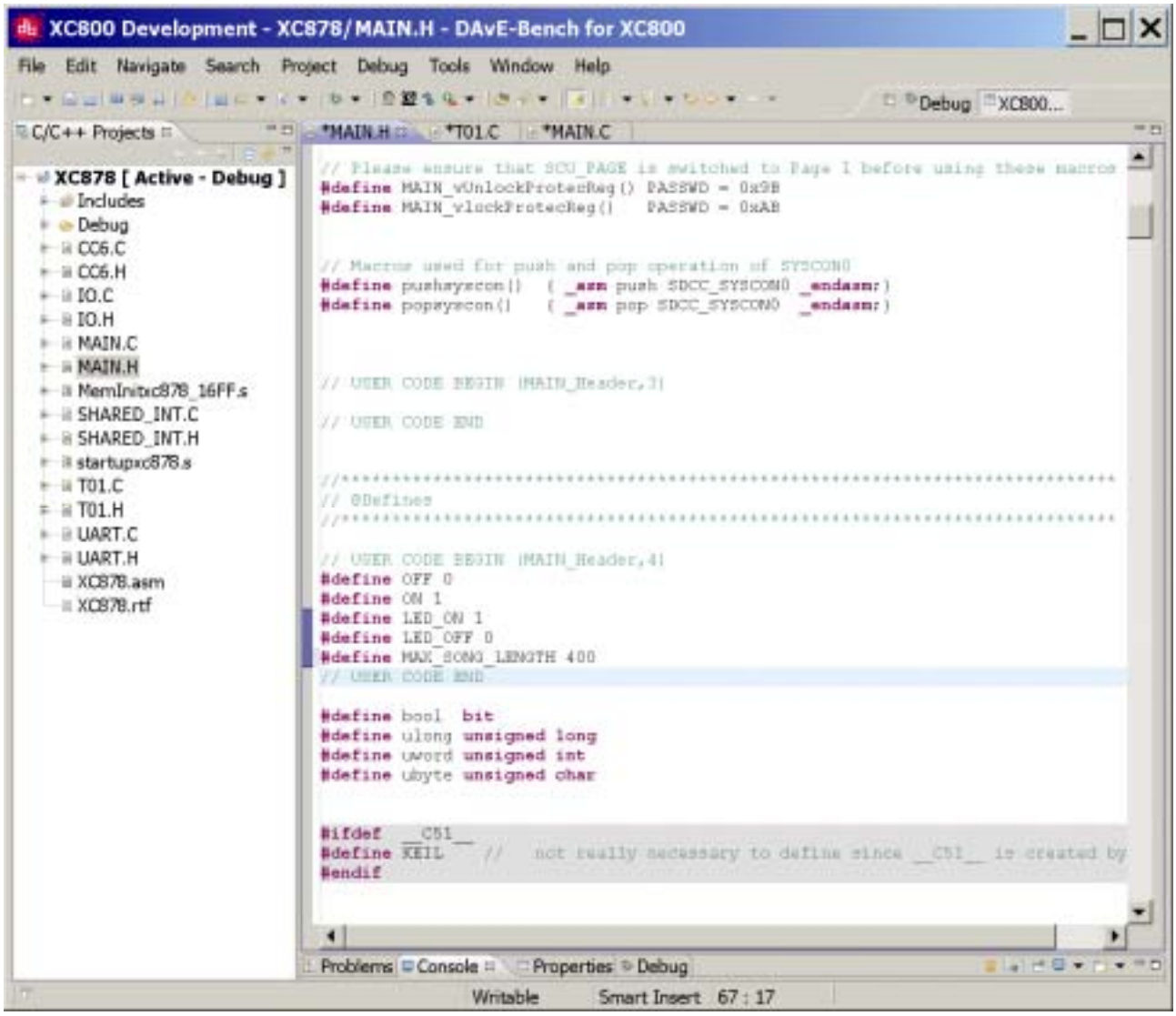
```
//Songs:

// Maus am Mars (song a) :
code unsigned char
songa[]="T120O0L4FL8AL4O1C.O0L8FEGL2O1CO0P4P8L4EL8GO1L4C.O0L8EFAL2O1CP4
P8O0L4FL8AO1L4C.O0L8FH-O1L4DFL8FEDDCO0HO1CDCO0H-GL2F.";
// Yesterday (song b) :
code unsigned char
songb[]="T120O0L8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2AAAL4O1DEFL8EDL4E.L8DL4CEFCO0H-
AL8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2A";
// Bruder Jakob (song c) :
code unsigned char songc[]="T120O0L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0L8H-
L4AFO1L8CDCO0L8H-L4AFFCL2FL4FCL2F";
// Happy birthday (song d) :
code unsigned char
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";
// 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";
// Es tanzt ein Bi-ba-butzemann (song f):
code unsigned char
songf[]="T199O0L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";
// Ich geh mit meiner Laterne (song g):
code unsigned char
songg[]="T120O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8CL4FL8FA
FAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4F
L8FGGGGAGL4FP4.O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4FL8FGGGGAGL4FP4.";
// 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";
// 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";
// 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.";
// Junge komm bald wieder (song k):
code unsigned 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):
code unsigned 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 / Tonleiter (song m) :
code unsigned char
songm[]="T120O0L4CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+AA+HO2CC+DD+EFF+G
G+AA+HO3CC+DD+EFF+GG+AA+HP4O0L8CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+
AA+HO2CC+DD+EFF+GG+AA+HO3CC+DD+EFF+GG+AA+HP8O0L16CC+DD+EFF+GG+A
A+HO1CC+DD+EFF+GG+AA+HO2CC+DD+EFF+GG+AA+HO3CC+DD+EFF+GG+AA+HP16
";

unsigned char xdata song[MAX_SONG_LENGTH];
```


XC800 Development - XC878/MAIN.C - DAVE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.H
*T01.C
*MAIN.C
Debug
XC800...
Problems
Console
Properties
Debug
Writable
Smart Insert
169 : 1

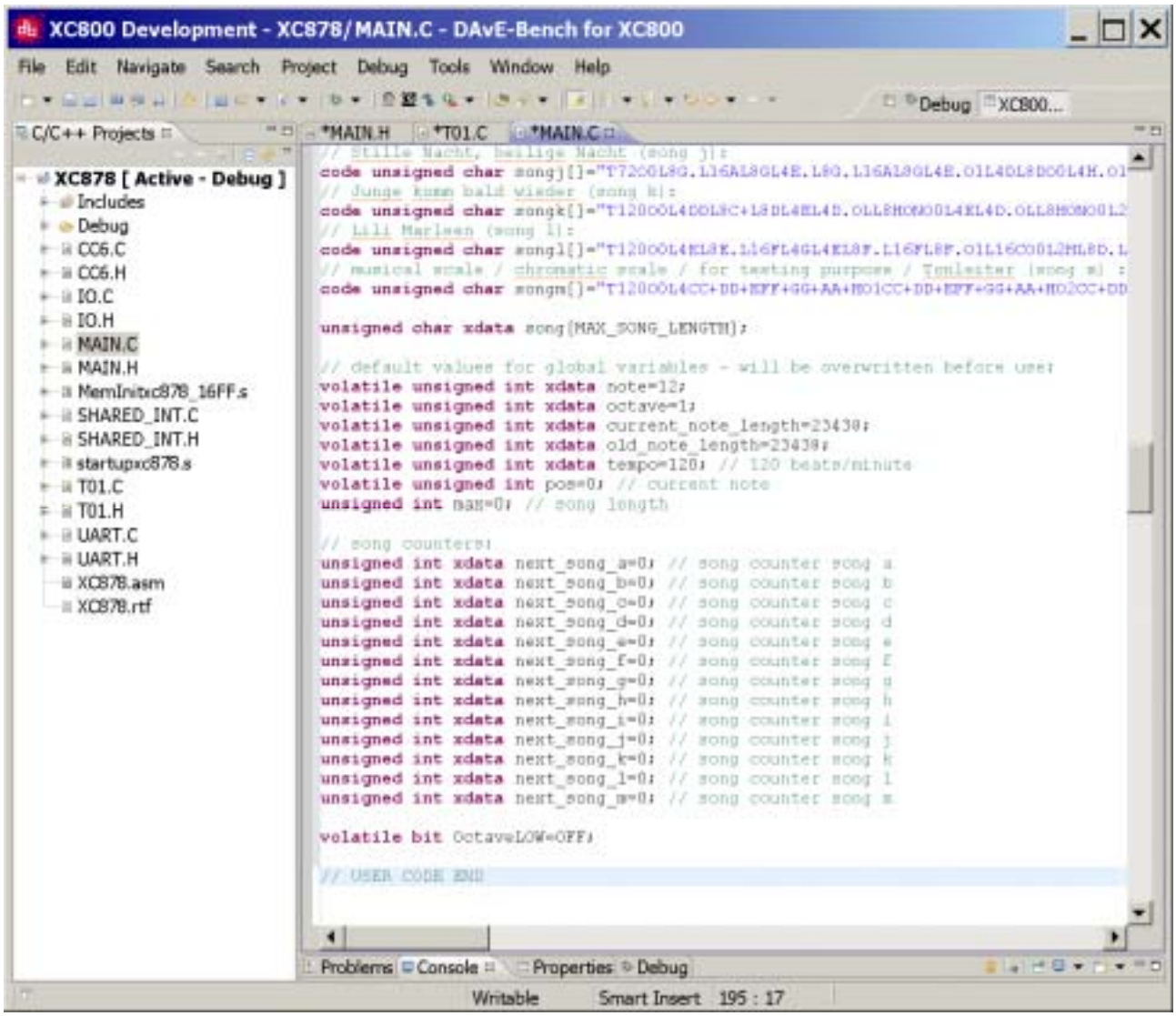
Double click MAIN.H and insert Define:

```
#define MAX_SONG_LENGTH 400
```



Double click MAIN.C and insert Global Variables:

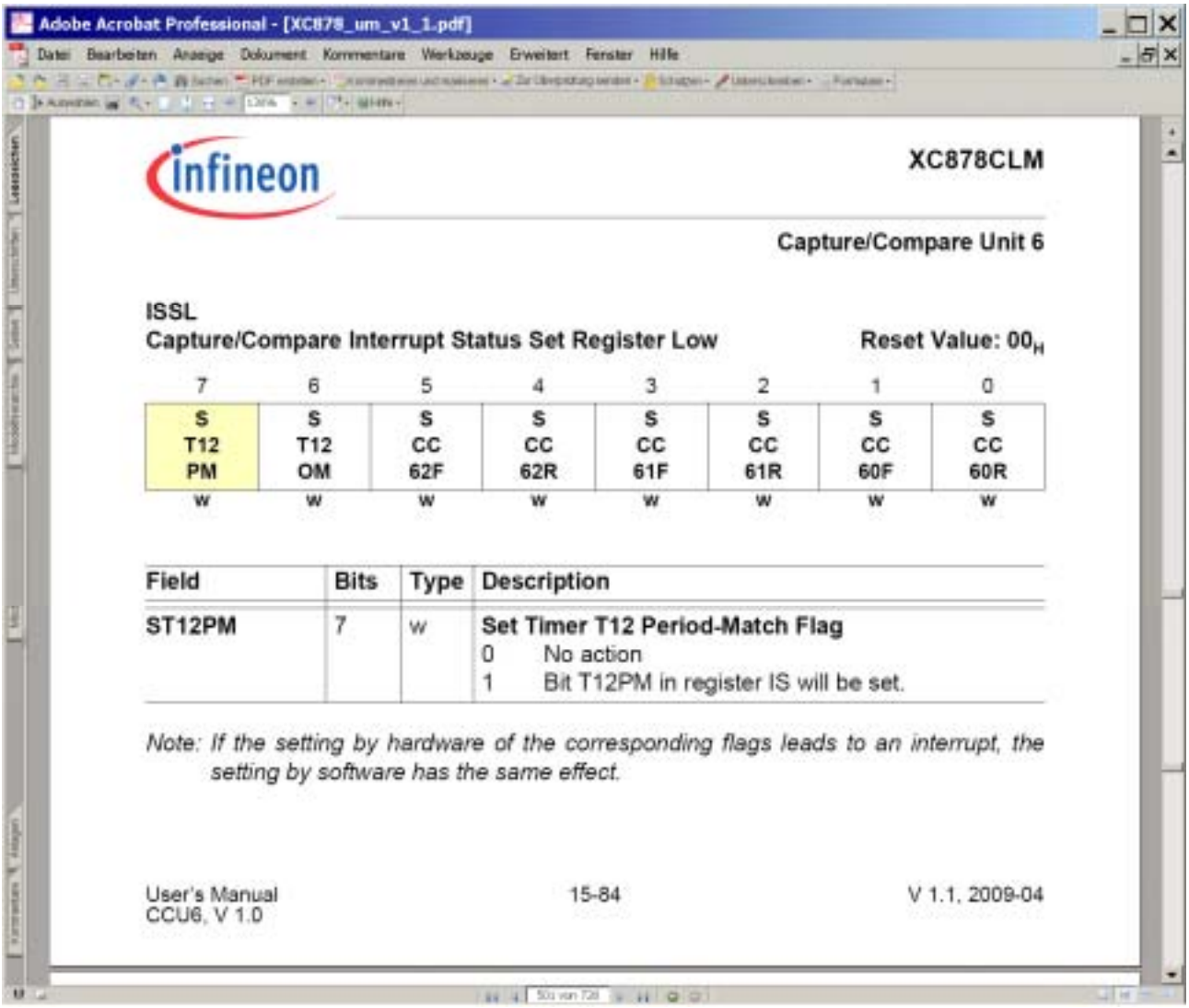
```
// default values for global variables - will be overwritten before use:
volatile unsigned int xdata note=12;
volatile unsigned int xdata octave=1;
volatile unsigned int xdata current_note_length=23438;
volatile unsigned int xdata old_note_length=23438;
volatile unsigned int xdata tempo=120; // 120 beats/minute
volatile unsigned int pos=0; // current note
unsigned int max=0; // song length

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

volatile bit OctaveLOW=OFF;
```


XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
*MAIN.H
*T01.C
*MAIN.C
unsigned char xdata song[MAX_SONG_LENGTH];
volatile unsigned int xdata note=12;
volatile unsigned int xdata octave=1;
volatile unsigned int xdata current_note_length=23430;
volatile unsigned int xdata old_note_length=23430;
volatile unsigned int xdata tempo=120; // 120 beats/minute
volatile unsigned int xdata pos=0; // current note
unsigned int nam=0; // song length
volatile bit OctaveLOW=OFF;
// USER CODE END
Problems Console Properties Debug
Writable Smart Insert 195 : 17

Note:
In the following code sequence

```
CCU6_ISSL = CCU6_ISSL  | 0x80
```

we have to access/set the ST12PM bit (Set Timer 12 Period-Match Flag).
The ST12PM bit is located in the ISSL register (Capture/Compare Interrupt Status Set Register Low).



Note:
The ISSL register (Capture/Compare Interrupt Status Set Register Low) is located in Page 2.

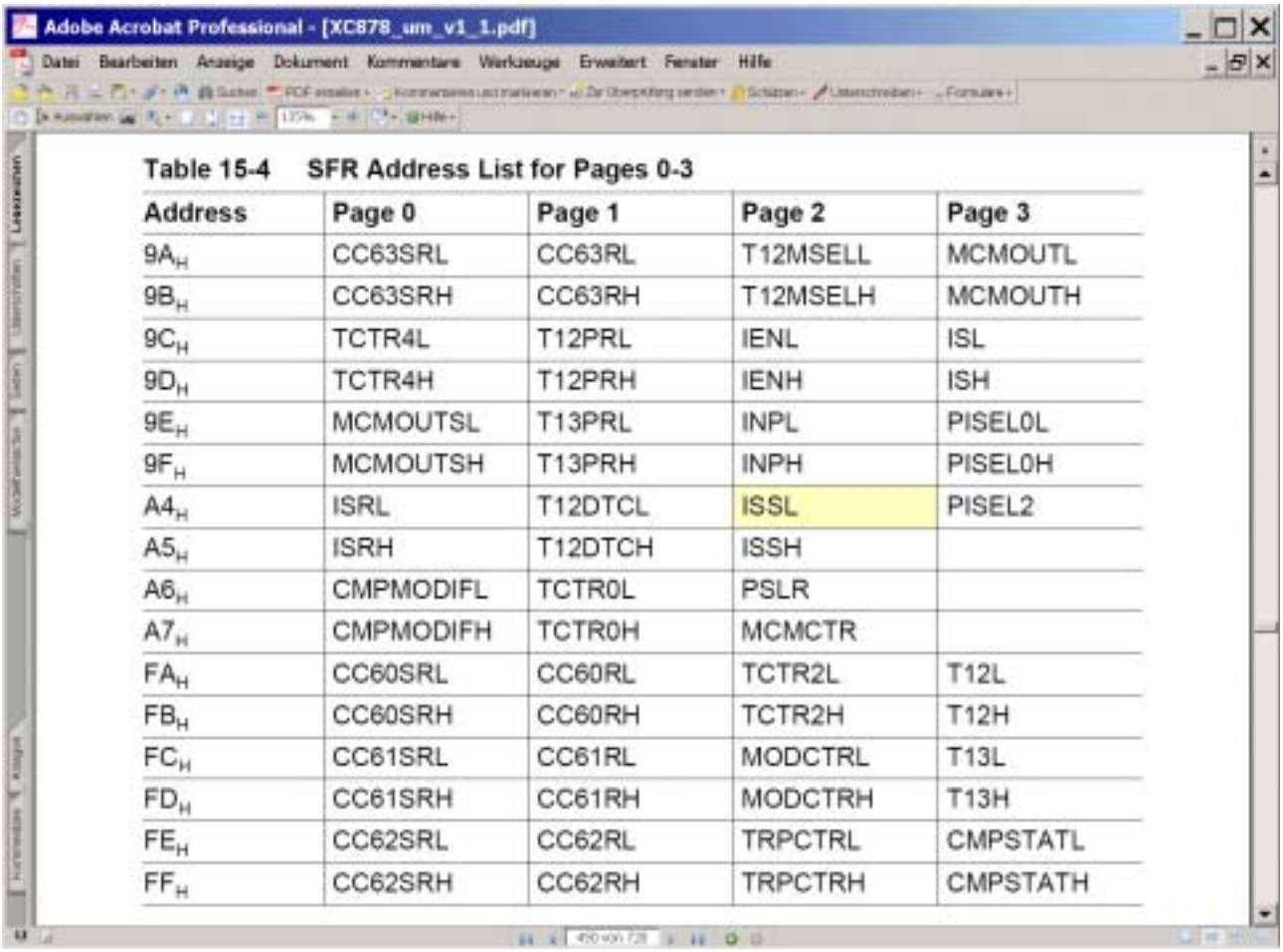


Table 15-4 SFR Address List for Pages 0-3

Address	Page 0	Page 1	Page 2	Page 3
$9A_H$	CC63SRL	CC63RL	T12MSELL	MCMOUTL
$9B_H$	CC63SRH	CC63RH	T12MSELH	MCMOUTH
$9C_H$	TCTR4L	T12PRL	IENL	ISL
$9D_H$	TCTR4H	T12PRH	IENH	ISH
$9E_H$	MCMOUTSL	T13PRL	INPL	PISEL0L
$9F_H$	MCMOUTSH	T13PRH	INPH	PISEL0H
$A4_H$	ISRL	T12DTCL	ISSL	PISEL2
$A5_H$	ISRH	T12DTCH	ISSH	
$A6_H$	CMPMODIFL	TCTR0L	PSLR	
$A7_H$	CMPMODIFH	TCTR0H	MCMCTR	
FA_H	CC60SRL	CC60RL	TCTR2L	T12L
FB_H	CC60SRH	CC60RH	TCTR2H	T12H
FC_H	CC61SRL	CC61RL	MODCTRL	T13L
FD_H	CC61SRH	CC61RH	MODCTRH	T13H
FE_H	CC62SRL	CC62RL	TRPCTRL	CMPSTATL
FF_H	CC62SRH	CC62RH	TRPCTRH	CMPSTATH

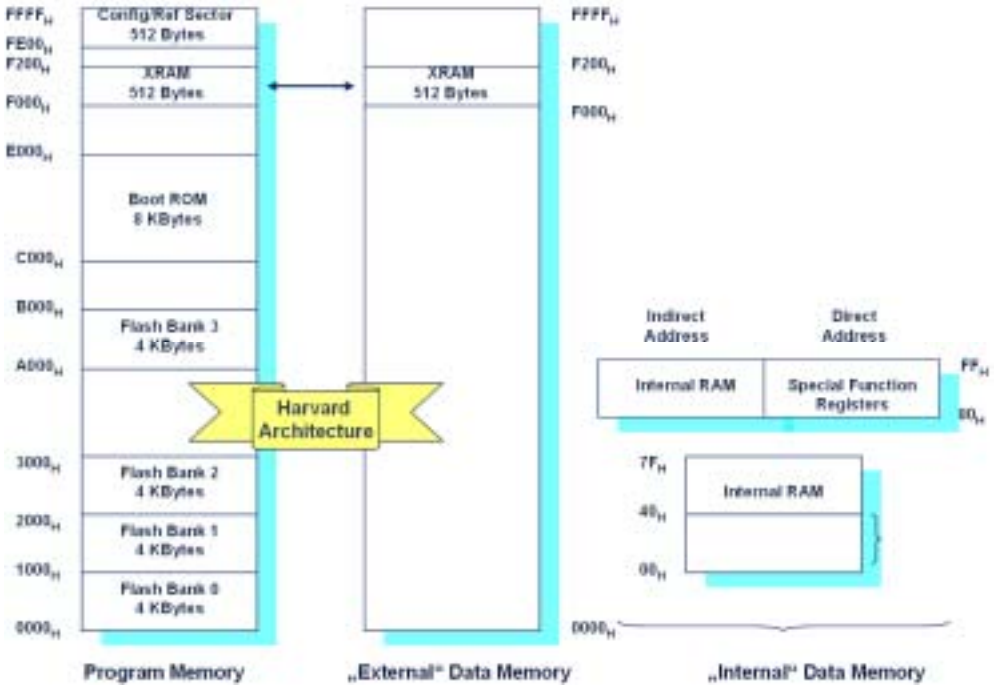
Note (Source: User's Manual):
The CCU6 SFRs are located in the standard memory area (RMAP = 0) and are organized into 4 pages. The CCU6_PAGE register contains the page value and the page control information.

Therefore, we can use the following code sequence:

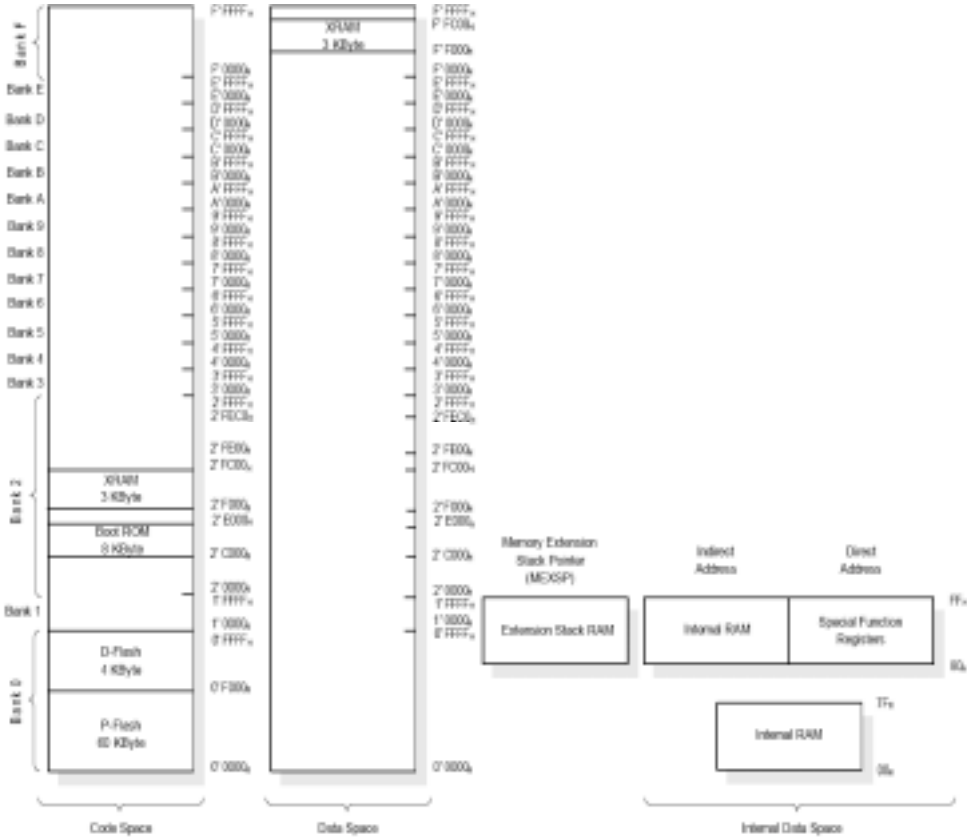
```
// start CAPCOM 6 - Timer T12 – ISR the first time:
SFR_PAGE(_cc2,noSST); // CCU6_PAGE = Page 2 !!!
// Access the module SFR :
CCU6_ISSL = CCU6_ISSL  | 0x80; // set ST12PM -> Set-Timer-T12-Period-Match-Flag
```

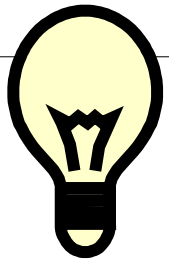
Note: Memory Organization:

From **8052/XC866**:



to **XC878**:





Address Extension:

Note:

In the XC800 architecture, the Special Function Registers (SFRs) occupy the direct internal data memory space in the range 80_H to FF_H. However, the 128-Byte-SFR range is less than the total number of registers required and therefore address extension mechanisms are used to increase the number of addressable SFRs. The address extension mechanisms are:

.) Mapping
.) Paging

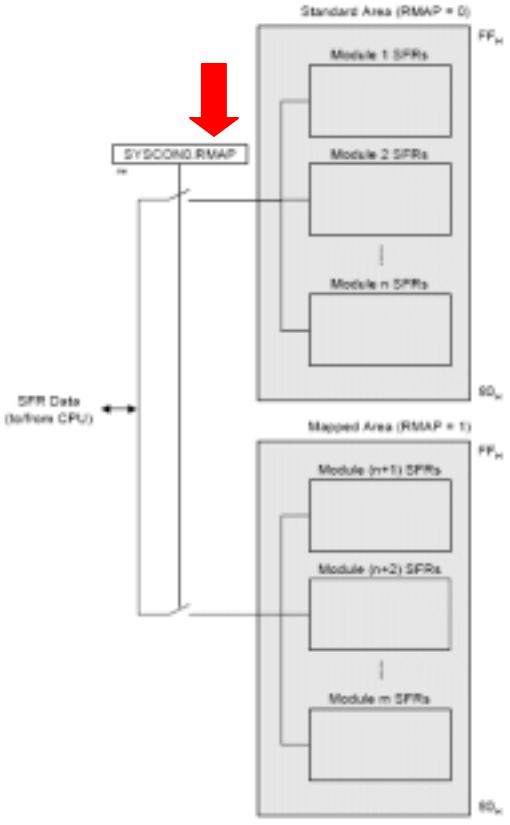
Address Extension by Mapping:

Note (Source: User's Manual):
The SFR area is extended into two portions: the standard (non-mapped) SFR area and the mapped SFR area. Each portion supports the same address range 80_H to FF_H, extending the number of addressable SFRs to 256 Bytes.
The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit RMAP in the system control register SYSCON0.
To access SFRs in the mapped area, bit RMAP in SFR SYSCON0 must be set.
The SFRs in the standard area can be accessed by clearing bit RMAP.
As long as bit RMAP is set, the mapped SFR area can be accessed. This bit is not cleared automatically by hardware. Thus, before standard/mapped registers are accessed, bit RMAP must be cleared/set, respectively, by software.

SYSCON0
System Control Register 0 — **Reset Value: 04_H**

7	6	5	4	3	2	1	0
0			IMODE	0	1	0	RMAP
r			rw	r	r	r	rw

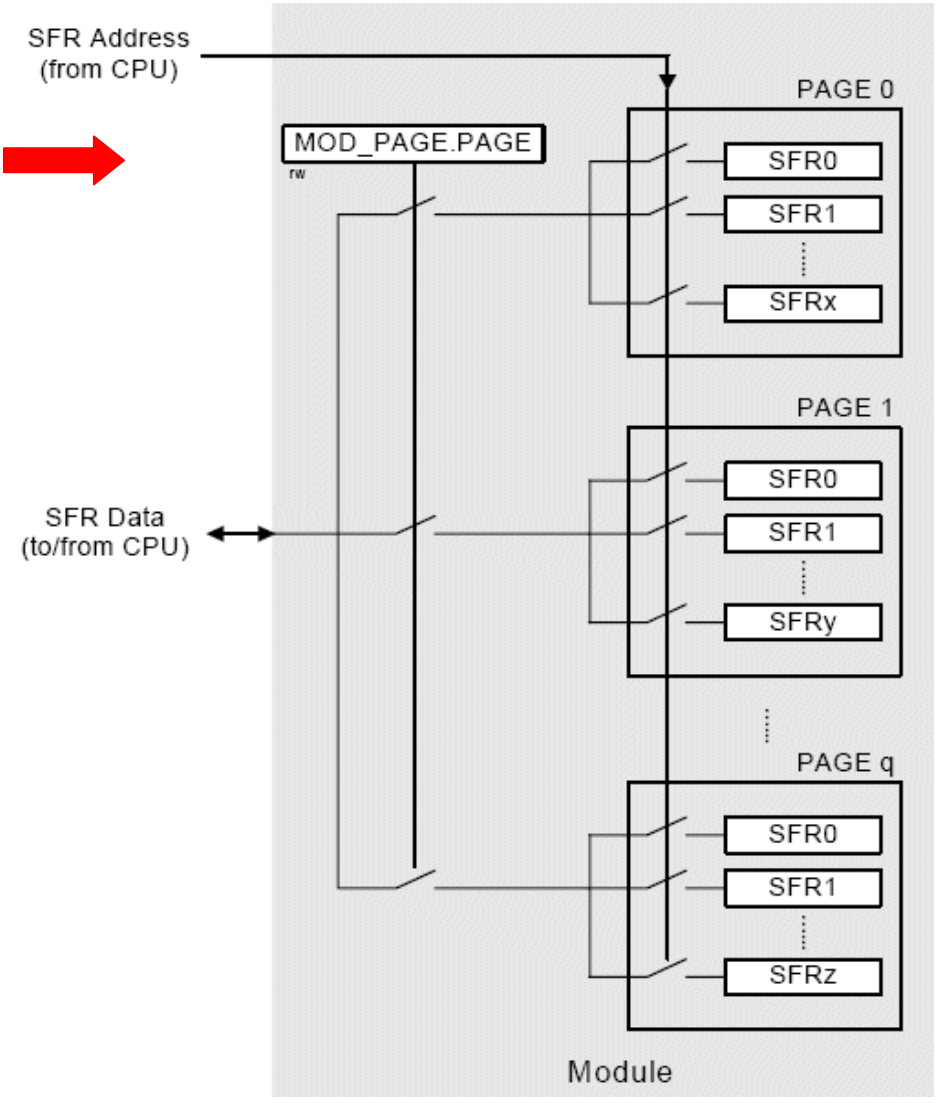
Field	Bits	Type	Description
RMAP	0	rw	**Special Function Register Map Control** 0 The access to the standard SFR area is enabled. 1 The access to the mapped SFR area is enabled.



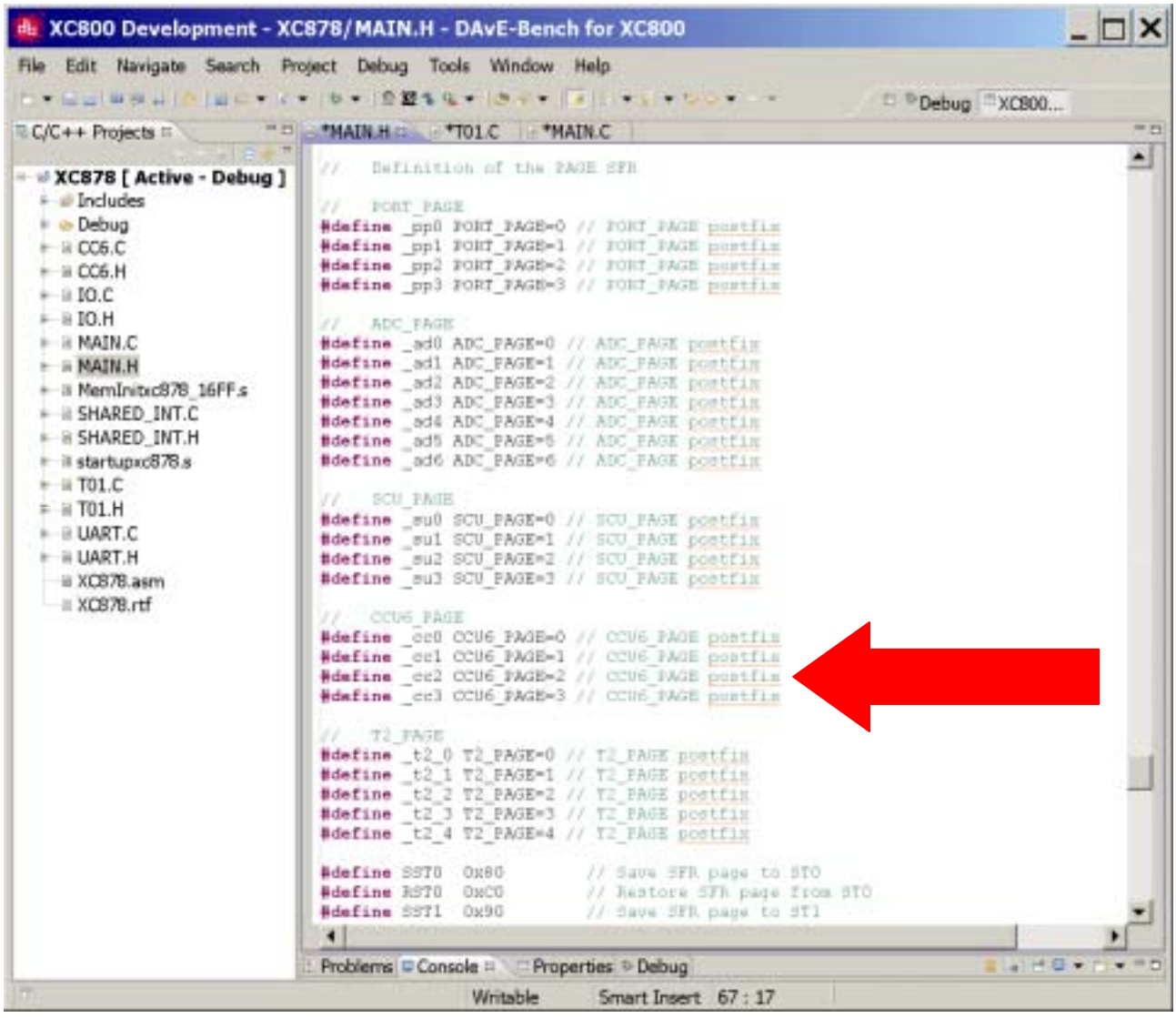
Address Extension by Paging:

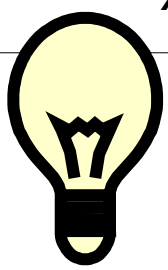
Note (Source: User's Manual):
Address extension is further performed at the module level by paging. With the address extension by mapping, the XC8xx has a 256-SFR address range. However, this is still less than the total number of SFRs needed by the on-chip peripherals.
To meet this requirement, some peripherals (Parallel Ports, Analog-to-Digital Converter, Capture/Compare Unit 6, System Control Registers) have a built-in local address extension mechanism for increasing the number of addressable SFRs. The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit field PAGE in the module page register MOD_PAGE. Hence, the bit field PAGE must be programmed before accessing the SFRs of the target module. Each module may contain a different number of pages and a different number of SFRs per page, depending on the specific requirement.
Besides setting the correct RMAP bit value to select the SFR area, the user must also ensure that a valid PAGE is selected to target the desired SFRs.

Note (Source: Application Note AP08053):
It should be noted that each peripheral that supports paging has its own page register.

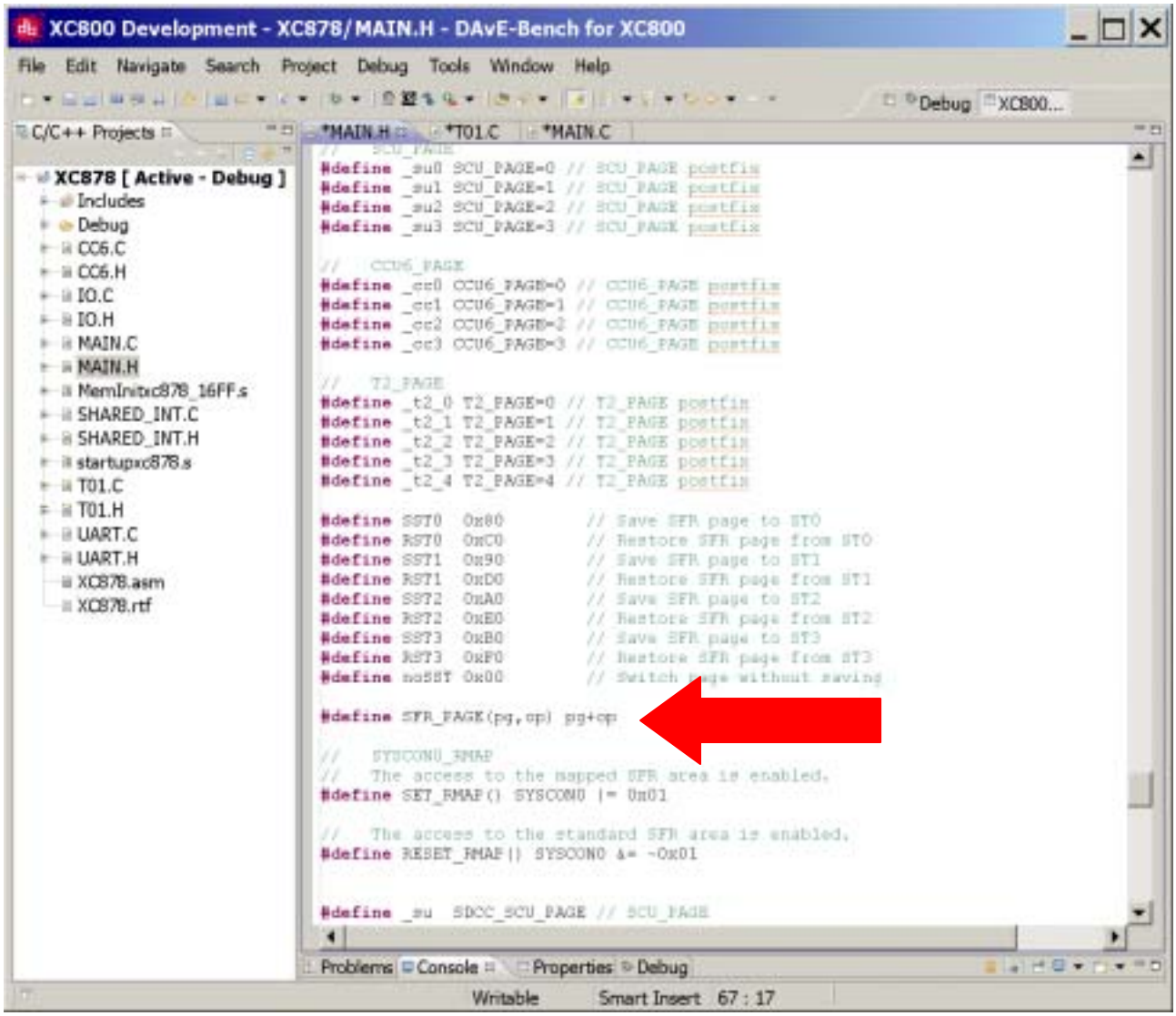


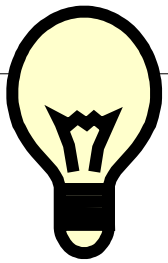
We can see the PAGE SFR definition in the MAIN.H file:



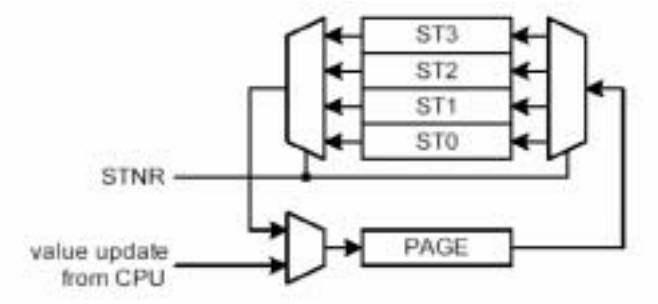


Additionally, there are useful macros available with which we can easily handle the Address Extension by Paging during interrupt using the Storage Containers:



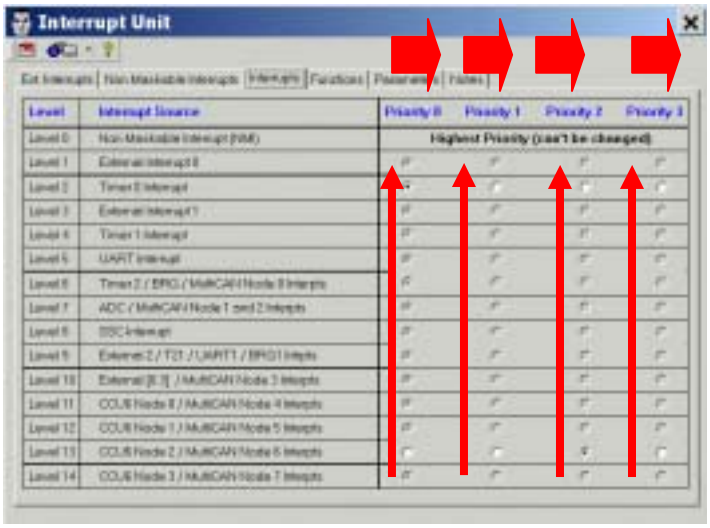


Address Extension (via Mapping and Paging) with respect to the Interrupt System using Storage Containers:



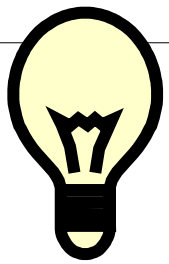
Note (Source: Application Note AP08053):
There could be six interrupt priorities.
These priorities, with 6 being the highest, are as follows:

Interrupt Priority:	
6	NMI
5	Interrupt Priority 3
4	Interrupt Priority 2
3	Interrupt Priority 1
2	Interrupt Priority 0
1	Main



Main refers to routines that run prior to any interrupt and can be interrupted by any interrupt.
Each interrupt source can be programmed to any of the four interrupt priorities (0-3).
An interrupt that is currently being serviced can only be interrupted by a higher priority interrupt, but not by another interrupt of the same or lower priority.
Hence, an interrupt of the highest priority cannot be interrupted by any other interrupt.
In any case, the NMI always has the highest priority (above priority 3) and its priority cannot be programmed.

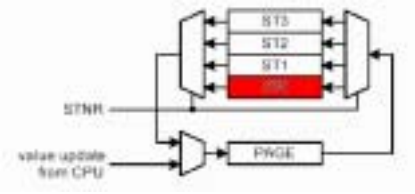
The XC800 architecture provides an efficient mechanism to save and modify the current page setting without using the stack.
This paging mechanism contains 4 storage containers for the save and restore action.



For any of the six interrupt priorities, the storage number should be unique for each priority to avoid being overwritten by a different storage number when it is interrupted by a higher priority interrupt that is accessing the same module.
Users must also ensure that the storage numbers within the ISRs are changed accordingly when the interrupt priorities are changed.

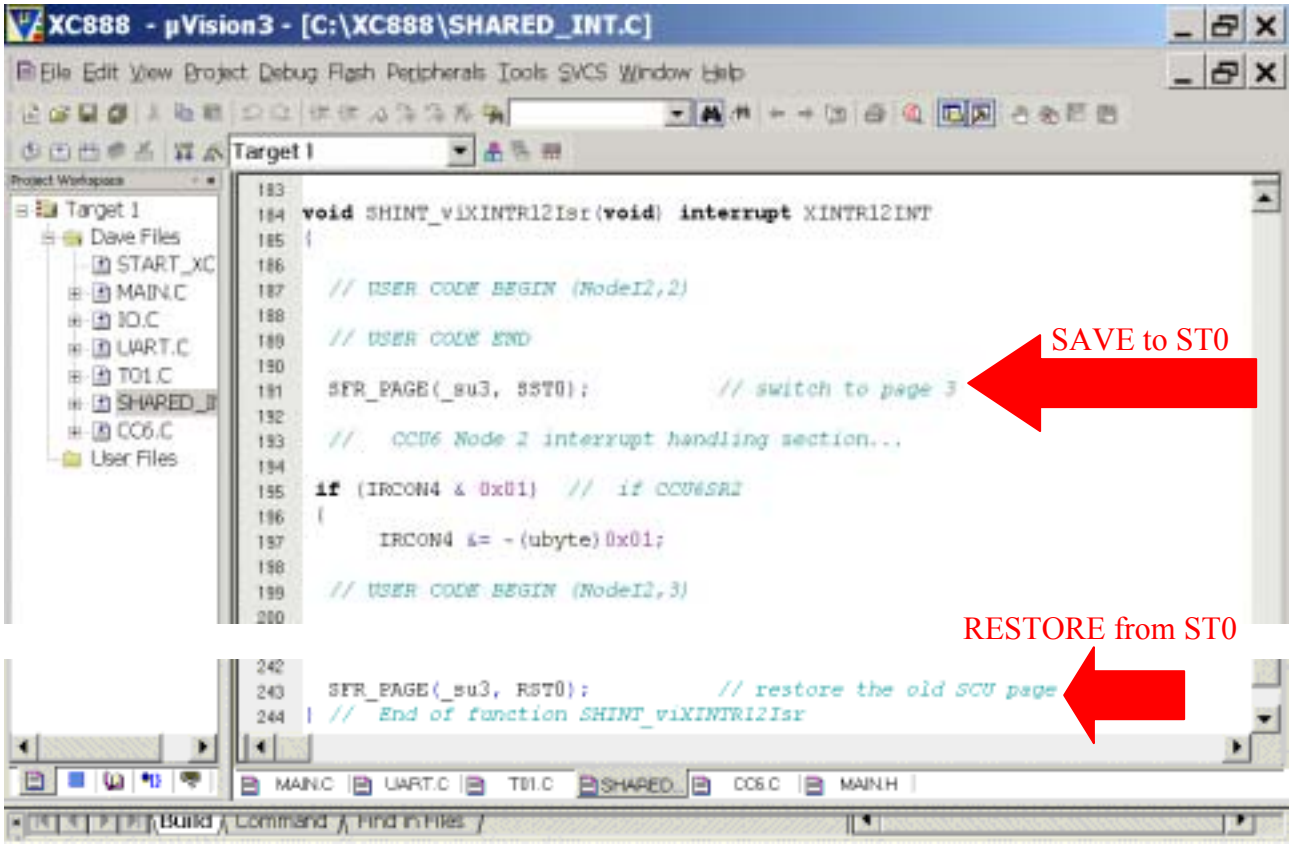
The main routine should not use a storage container.
This leaves us with five interrupt priorities and four storage containers.
If all priorities are used in an application, then not every interrupt priority can have its own storage container. A workaround is to make use of the stack as the extended storage container.
The ISR may also call functions that could modify page registers. If these functions are shared by ISRs of different priority levels, then these functions can be interrupted. It is therefore necessary to save and restore the page registers that are modified in these functions. In such cases, the stack should be used as a storage container.

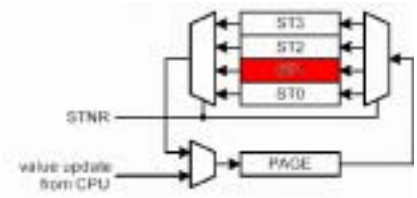
In summary:
• All the page registers modified in an ISR must be saved.
• The storage container should be unique for each interrupt priority.
• The storage numbers within the ISRs must be changed accordingly when the interrupt priorities are changed.
• No storage container is necessary for the main level.
• Stack can be used as an extended storage container.
• Page registers modified in functions called by the ISRs should use the stack as the storage container if the functions are shared by different interrupt priorities.



As you can see in the screenshots below,
DAvE uses Storage Container 0 for Interrupt Priority 0:

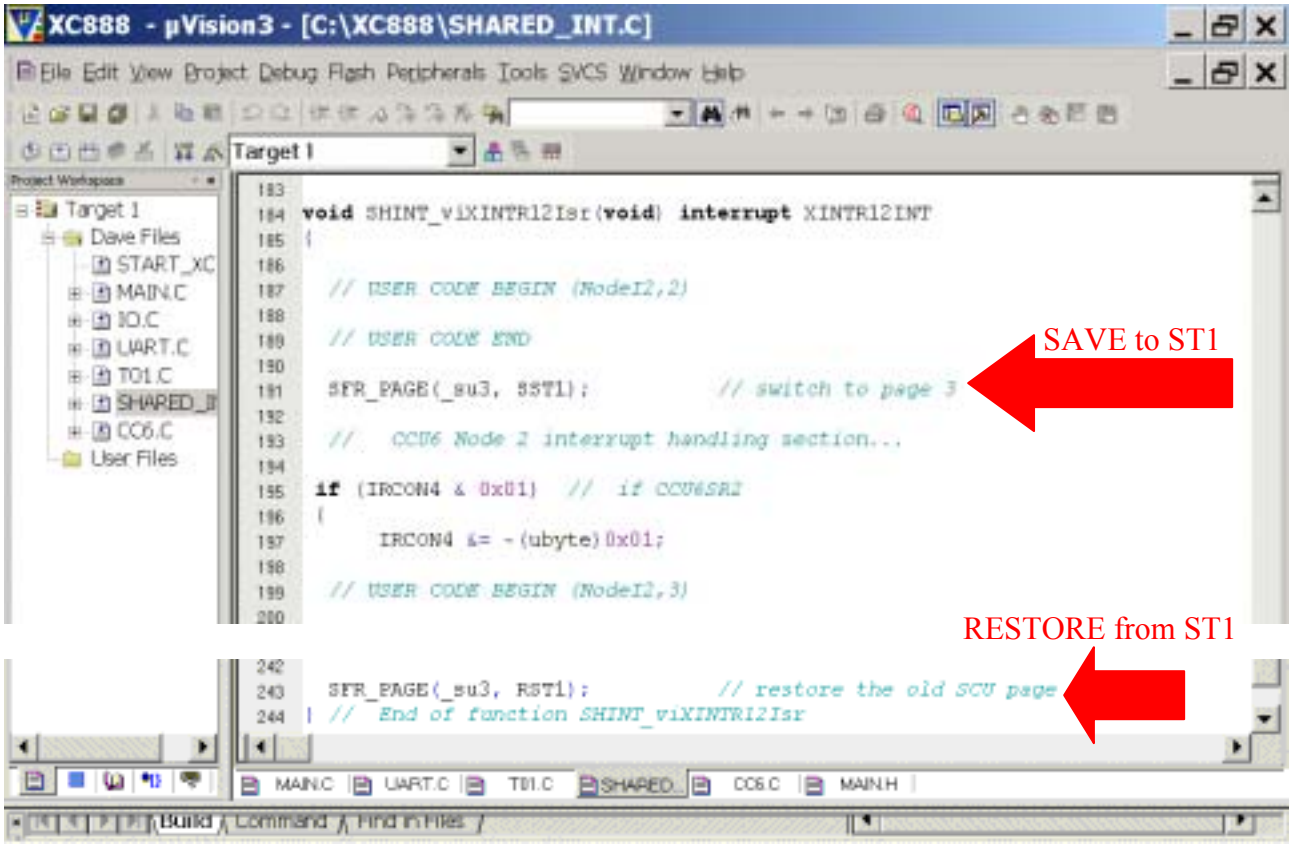
Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	○	○	○	○
Level 2	Timer 0 Interrupt	◉	○	○	○
Level 3	External Interrupt 1	○	○	○	○
Level 4	Timer 1 Interrupt	○	○	○	○
Level 5	UART Interrupt	○	○	○	○
Level 6	Timer 2 / BRG / MultiCAN Node 0 Interpts	○	○	○	○
Level 7	ADC / MultiCAN Node 1 and 2 Interpts	○	○	○	○
Level 8	SSC Interrupt	○	○	○	○
Level 9	External 2 / T21 / UART1 / BRG1 Intrpts	○	○	○	○
Level 10	External [6:3] / MultiCAN Node 3 Interpts	○	○	○	○
Level 11	CCU6 Node 0 / MultiCAN Node 4 Interpts	○	○	○	○
Level 12	CCU6 Node 1 / MultiCAN Node 5 Interpts	○	○	○	○
Level 13	CCU6 Node 2 / MultiCAN Node 6 Interpts	◉	○	○	○
Level 14	CCU6 Node 3 / MultiCAN Node 7 Interpts	○	○	○	○

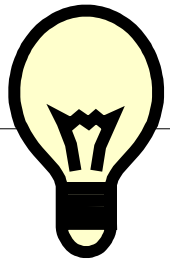


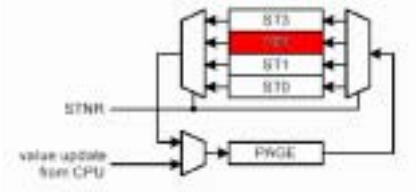


As you can see in the screenshots below,
DAvE uses Storage Container 1 for Interrupt Priority 1:

Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	○	○	○	○
Level 2	Timer 0 Interrupt	◉	○	○	○
Level 3	External Interrupt 1	○	○	○	○
Level 4	Timer 1 Interrupt	○	○	○	○
Level 5	UART Interrupt	○	○	○	○
Level 6	Timer 2 / BRG / MultiCAN Node 0 Interpts	○	○	○	○
Level 7	ADC / MultiCAN Node 1 and 2 Interpts	○	○	○	○
Level 8	SSC Interrupt	○	○	○	○
Level 9	External 2 / T21 / UART1 / BRG1 Intrpts	○	○	○	○
Level 10	External [6:3] / MultiCAN Node 3 Interpts	○	○	○	○
Level 11	CCU6 Node 0 / MultiCAN Node 4 Interpts	○	○	○	○
Level 12	CCU6 Node 1 / MultiCAN Node 5 Interpts	○	○	○	○
Level 13	CCU6 Node 2 / MultiCAN Node 6 Interpts	○	◉	○	○
Level 14	CCU6 Node 3 / MultiCAN Node 7 Interpts	○	○	○	○

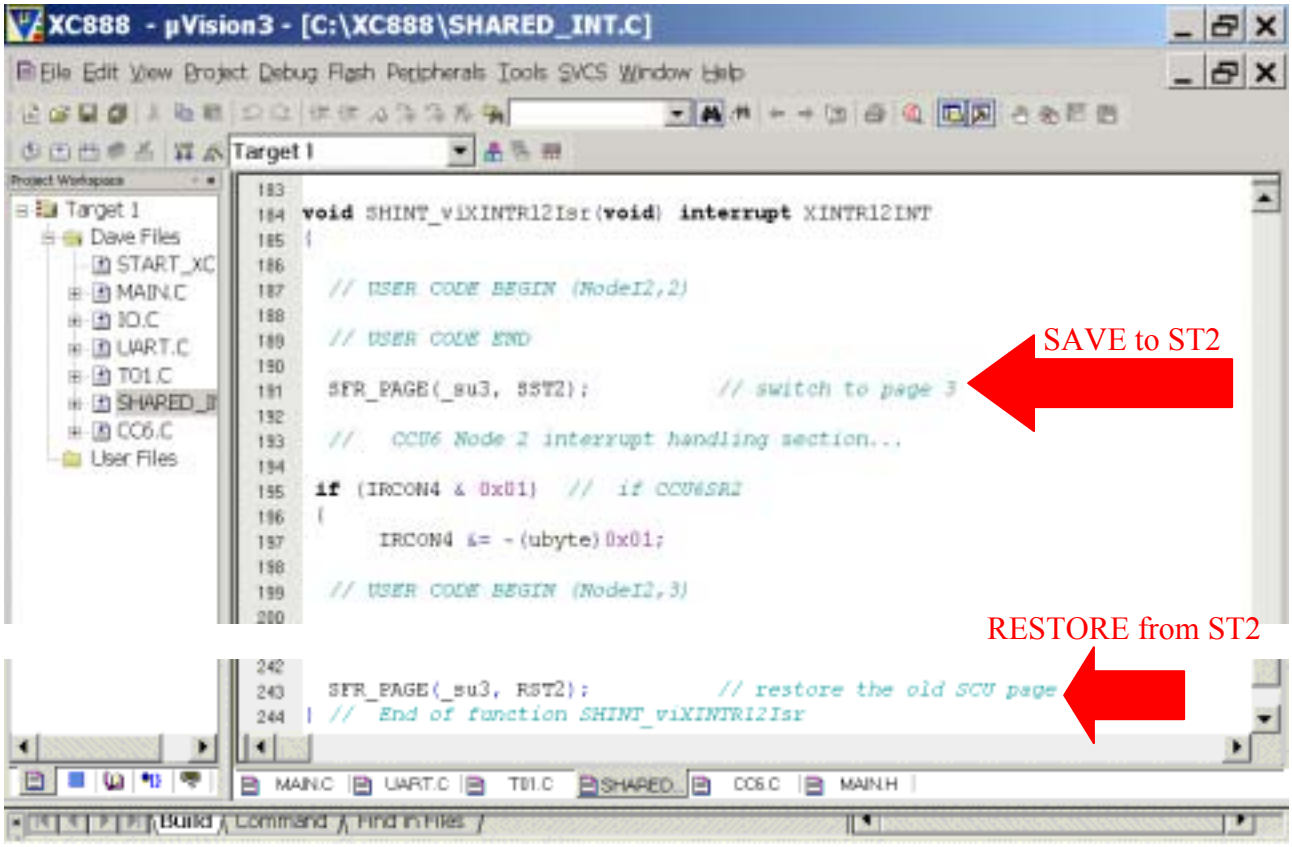


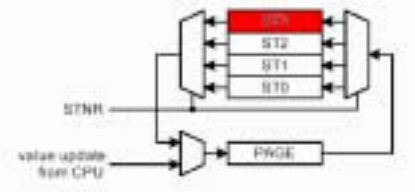




As you can see in the screenshots below,
DAvE uses Storage Container 2 for Interrupt Priority 2:

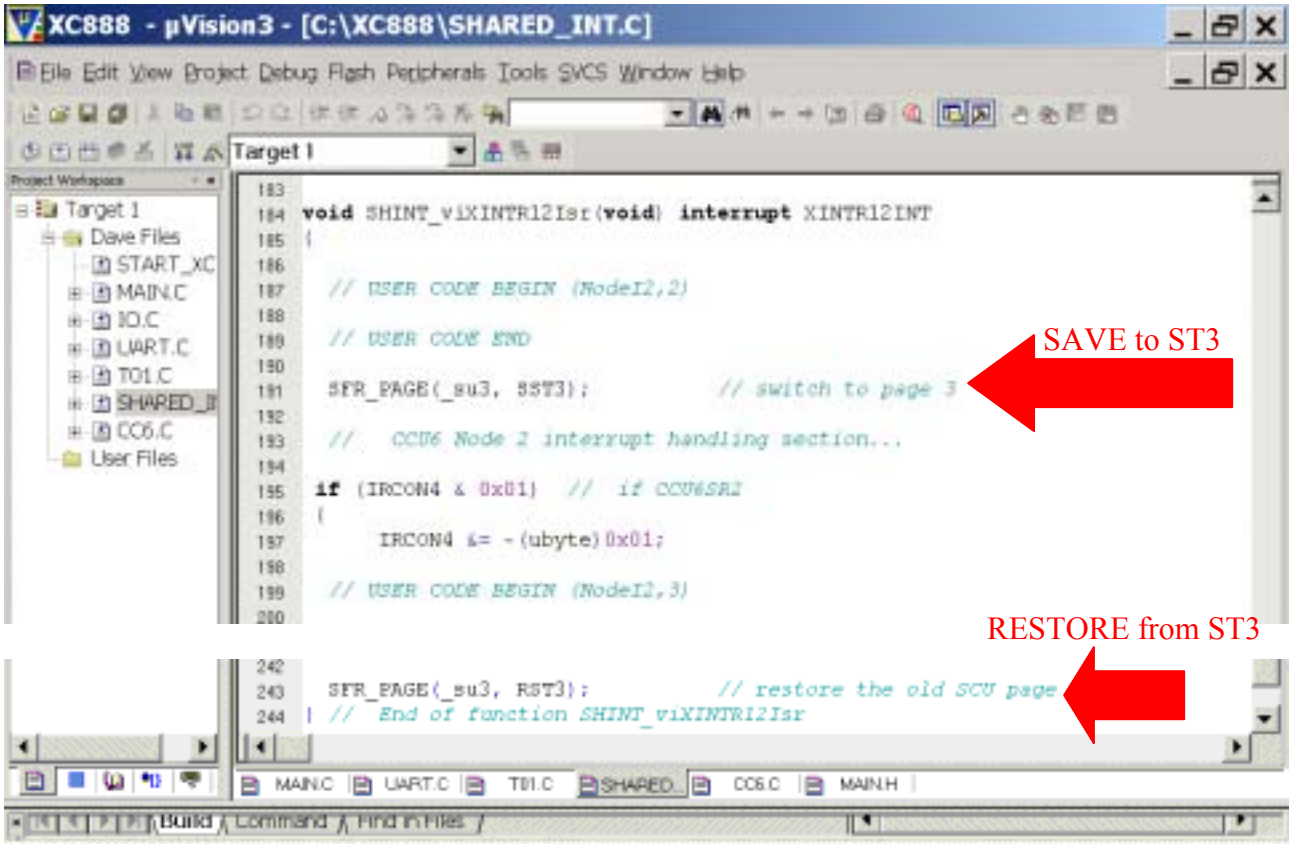
Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	○	○	○	○
Level 2	Timer 0 Interrupt	●	○	○	○
Level 3	External Interrupt 1	○	○	○	○
Level 4	Timer 1 Interrupt	○	○	○	○
Level 5	UART Interrupt	○	○	○	○
Level 6	Timer 2 / BRG / MultiCAN Node 0 Interpts	○	○	○	○
Level 7	ADC / MultiCAN Node 1 and 2 Interpts	○	○	○	○
Level 8	SSC Interrupt	○	○	○	○
Level 9	External 2 / T21 / UART1 / BRG1 Intrpts	○	○	○	○
Level 10	External [6:3] / MultiCAN Node 3 Interpts	○	○	○	○
Level 11	CCU6 Node 0 / MultiCAN Node 4 Interpts	○	○	○	○
Level 12	CCU6 Node 1 / MultiCAN Node 5 Interpts	○	○	○	○
Level 13	CCU6 Node 2 / MultiCAN Node 6 Interpts	○	○	●	○
Level 14	CCU6 Node 3 / MultiCAN Node 7 Interpts	○	○	○	○



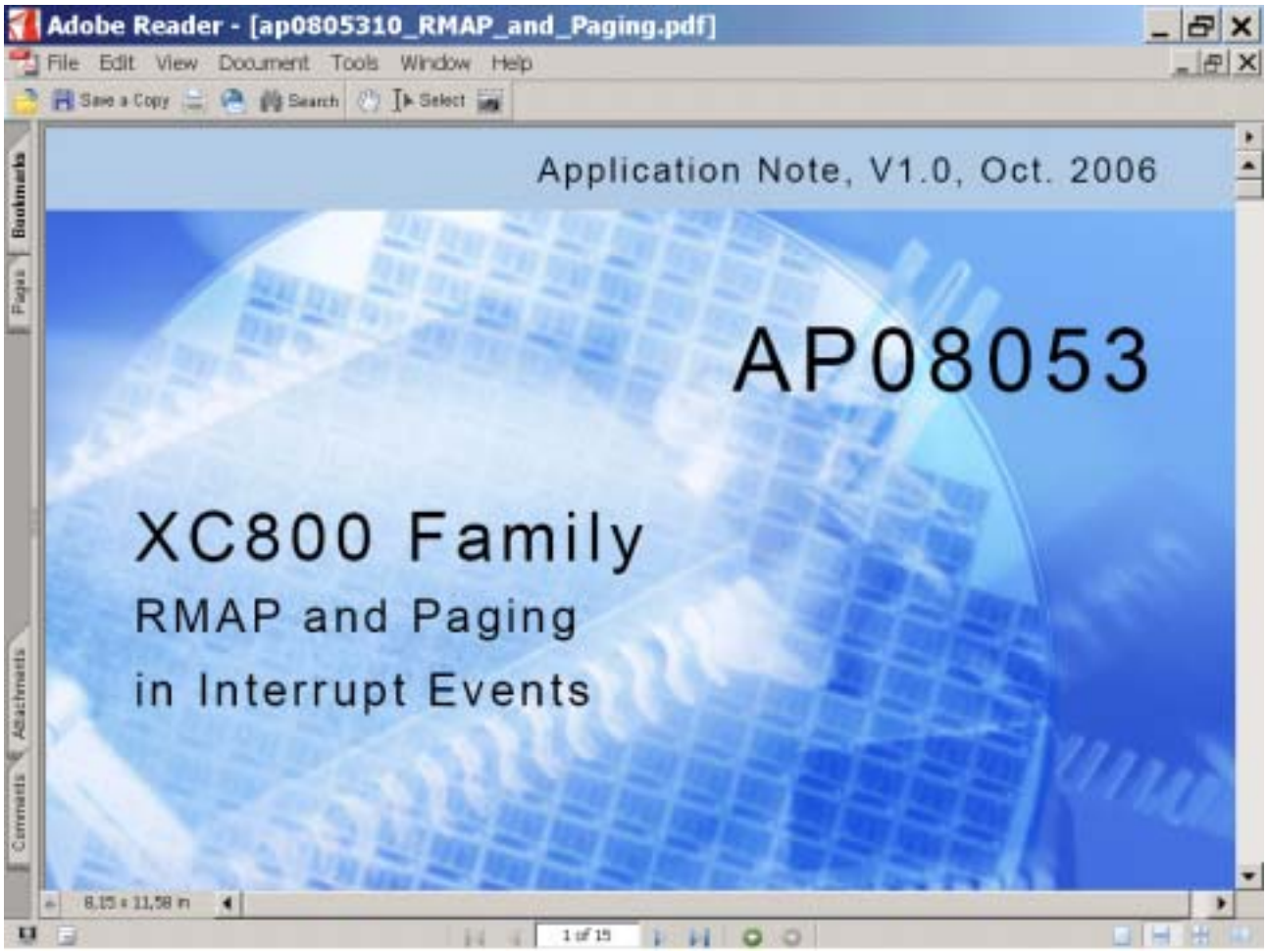


As you can see in the screenshots below,
DAvE uses Storage Container 3 for Interrupt Priority 3:

Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	○	○	○	○
Level 2	Timer 0 Interrupt	●	○	○	○
Level 3	External Interrupt 1	○	○	○	○
Level 4	Timer 1 Interrupt	○	○	○	○
Level 5	UART Interrupt	○	○	○	○
Level 6	Timer 2 / BRG / MultiCAN Node 0 Interpts	○	○	○	○
Level 7	ADC / MultiCAN Node 1 and 2 Interpts	○	○	○	○
Level 8	SSC Interrupt	○	○	○	○
Level 9	External 2 / T21 / UART1 / BRG1 Intrpts	○	○	○	○
Level 10	External [6:3] / MultiCAN Node 3 Interpts	○	○	○	○
Level 11	CCU6 Node 0 / MultiCAN Node 4 Interpts	○	○	○	○
Level 12	CCU6 Node 1 / MultiCAN Node 5 Interpts	○	○	○	○
Level 13	CCU6 Node 2 / MultiCAN Node 6 Interpts	○	○	○	●
Level 14	CCU6 Node 3 / MultiCAN Node 7 Interpts	○	○	○	○



In addition to the User's Manual, we suggest reading Application Note AP08053 for a better understanding of address extension via Mapping or Paging – especially if interrupts occur:



Double click MAIN.C and insert function " play_song() " - [after function "input()"]:

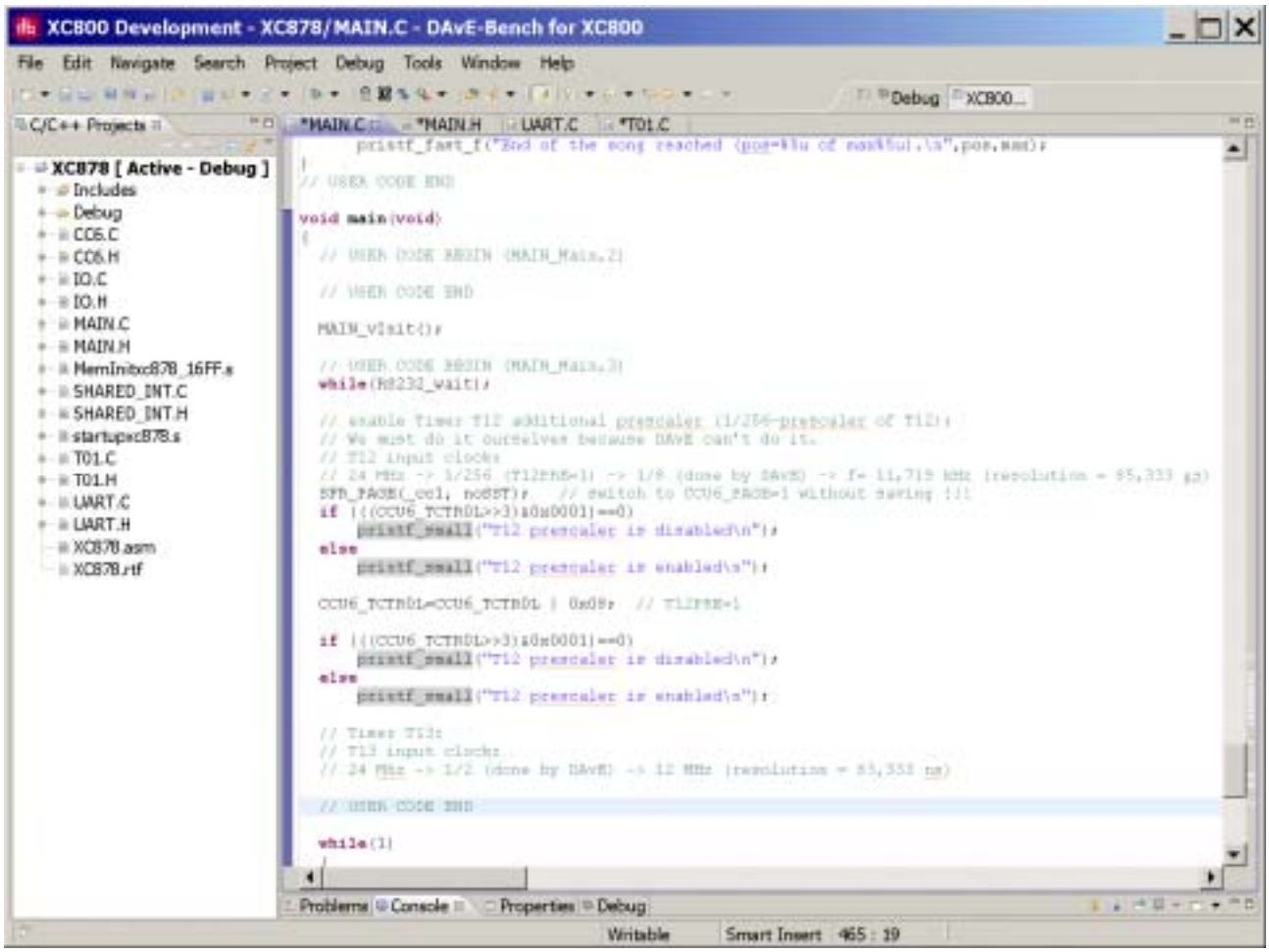
```
void play_song(void)
{
  max=0;
  if (next_song_a &&  ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
    printf_small("\nplaying: Maus am Mars\n");
  if (next_song_b &&  ((sizeof(songb)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songb), max=(sizeof(songb))-1, --next_song_b,
    printf_small("\nplaying: Yesterday\n");
  if (next_song_c &&  ((sizeof(songc)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songc), max=(sizeof(songc))-1, --next_song_c,
    printf_small("\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,
    printf_small("\nplaying: Happy birthday\n");
  if (next_song_e &&  ((sizeof(songe)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songe), max=(sizeof(songe))-1, --next_song_e,
    printf_small("\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,
    printf_small("\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,
    printf_small("\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,
    printf_small("\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,
    printf_small("\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,
    printf_small("\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,
    printf_small("\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,
    printf_small("\nplaying: Lili Marleen\n");
  if (next_song_m &&  ((sizeof(songm)-1)< MAX_SONG_LENGTH) )
    strcpy(song,songm), max=(sizeof(songm))-1, --next_song_m,
    printf_small("\nplaying: musical scale / chromatic scale / for testing purpose / Tonleiter\n");
  printf_fast_f("song-length = %5u Byte[s] \n",max);
  pos=0;

  if (max>0) // there is something to play
  {
```

```
      // start CAPCOM 6 - Timer T12 – ISR the first time:
      SFR_PAGE(_cc2,noSST); // switch to CCU6_PAGE=2 without saving !!!
      CCU6_ISSL = CCU6_ISSL  | 0x80; // set ST12PM -> Set-Timer-T12Period-Match-Flag

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

  if ( (SBUF=='z') )
    printf_small("Song aborted.\n");
  else
    printf_fast_f("End of the song reached (pos=%5u of max%5u).\n",pos,max);
}
```

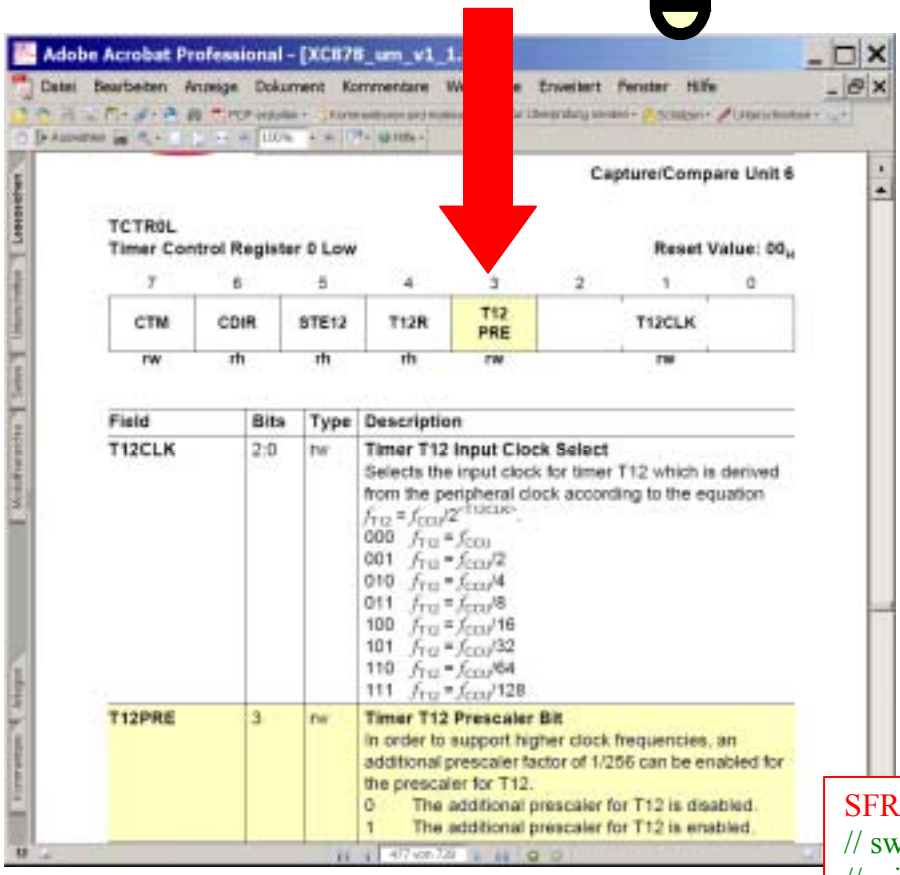

XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
*MAIN.C
*MAIN.H
UART.C
*T01.C
Problems
Console
Properties
Debug
Writable
Smart Insert
431 : 2

Double click MAIN.C and insert code (set the T12PRE bit):

```
// enable Timer T12 additional prescaler (1/256-prescaler of T12):
// We must do it ourselves because DAvE can't do it.
// T12 input clock:
// 24 MHz -> 1/256 (T12PRE=1) -> 1/8 (done by DAvE) -> f= 11,719 kHz (resolution = 85,333 µs)
   SFR_PAGE(_cc1, noSST);   // switch to CCU6_PAGE=1 without saving !!!
   if (((CCU6_TCTR0L>>3)&0x0001)==0)
     printf_small("T12 prescaler is disabled\n");
   else
     printf_small("T12 prescaler is enabled\n");

   CCU6_TCTR0L=CCU6_TCTR0L | 0x08;  // T12PRE=1

   if (((CCU6_TCTR0L>>3)&0x0001)==0)
     printf_small("T12 prescaler is disabled\n");
   else
     printf_small("T12 prescaler is enabled\n");

// Timer T13:
// T13 input clock:
// 24 Mhz -> 1/2 (done by DAvE) -> 12 MHz (resolution = 83,333 ns)
```


XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
*MAIN.C
*MAIN.H
UART.C
*T01.C
void main(void)
{
MAIN_vInit();
while(RS232_wait);
if ((CCU6_TCTR0L>>3)&0x0001)==0)
pristf_small("T12 prescaler is disabled\n");
else
pristf_small("T12 prescaler is enabled\n");
CCU6_TCTR0L=CCU6_TCTR0L | 0x08;
if ((CCU6_TCTR0L>>3)&0x0001)==0)
pristf_small("T12 prescaler is disabled\n");
else
pristf_small("T12 prescaler is enabled\n");
while(1)
Problems Console Properties Debug
Writable
Smart Insert
465 : 19

T12PRE=1:



Capture/Compare Unit 6

TCTR0L
Timer Control Register 0 Low — Reset Value: 00H

7	6	5	4	3	2	1	0
CTM	CDIR	STE12	T12R	T12 PRE		T12CLK	
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 which is derived from the peripheral clock according to the equation $f_{T12} = f_{CCU}/2^{T12CLK}$. 000 $f_{T12} = f_{CCU}$ 001 $f_{T12} = f_{CCU}/2$ 010 $f_{T12} = f_{CCU}/4$ 011 $f_{T12} = f_{CCU}/8$ 100 $f_{T12} = f_{CCU}/16$ 101 $f_{T12} = f_{CCU}/32$ 110 $f_{T12} = f_{CCU}/64$ 111 $f_{T12} = f_{CCU}/128$
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 The additional prescaler for T12 is disabled. 1 The additional prescaler for T12 is enabled.

```
SFR_PAGE(_cc1, noSST);
// switch to CCU6_PAGE=1
// without saving !!!
```

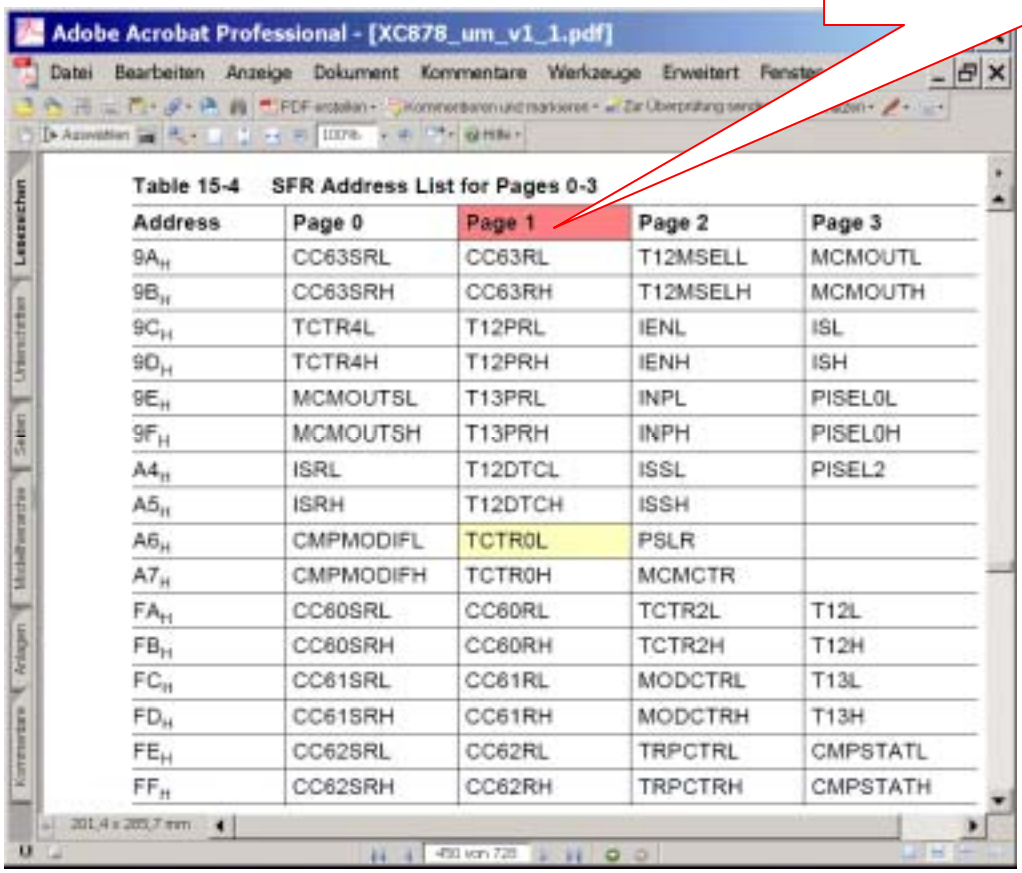
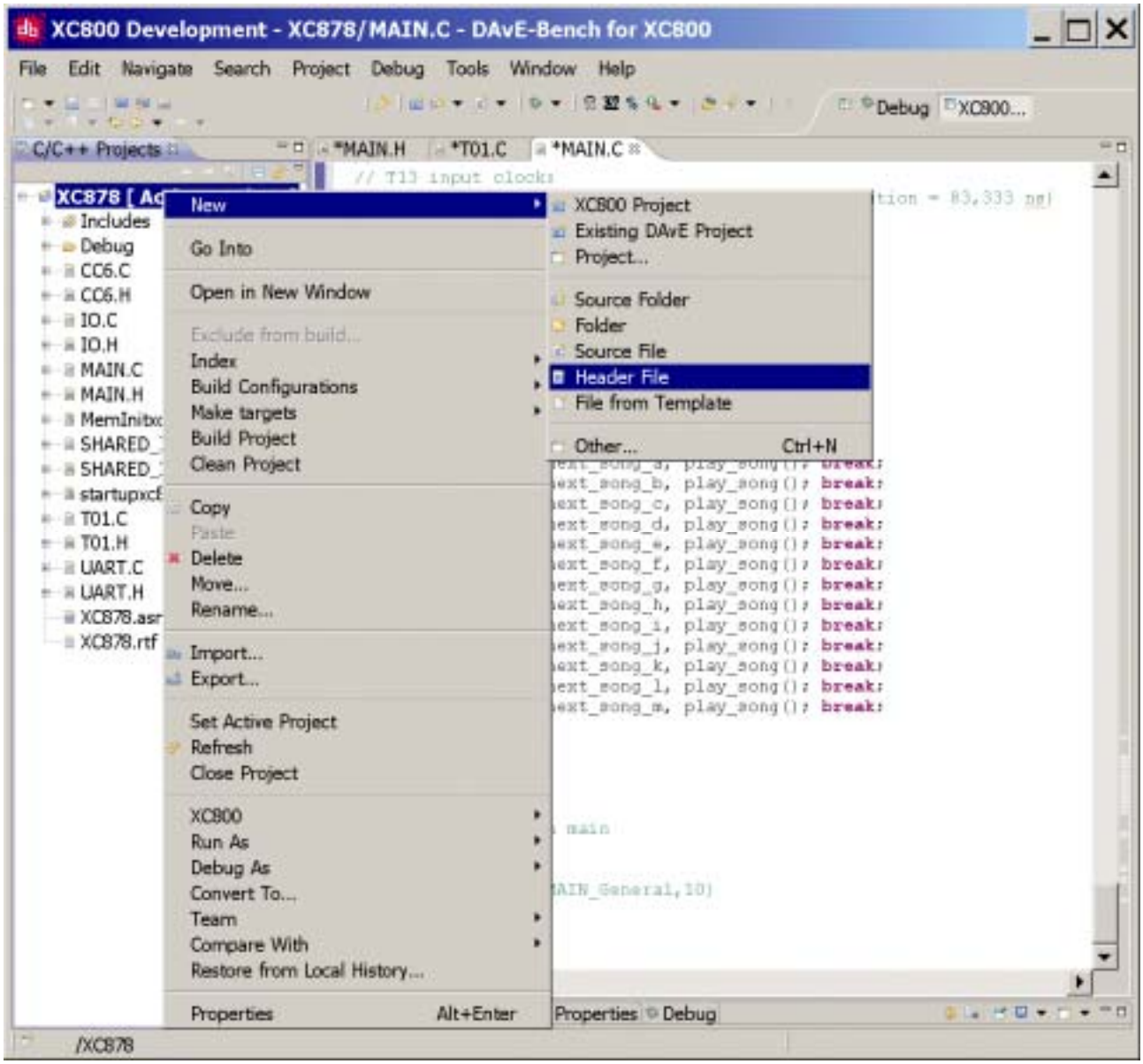


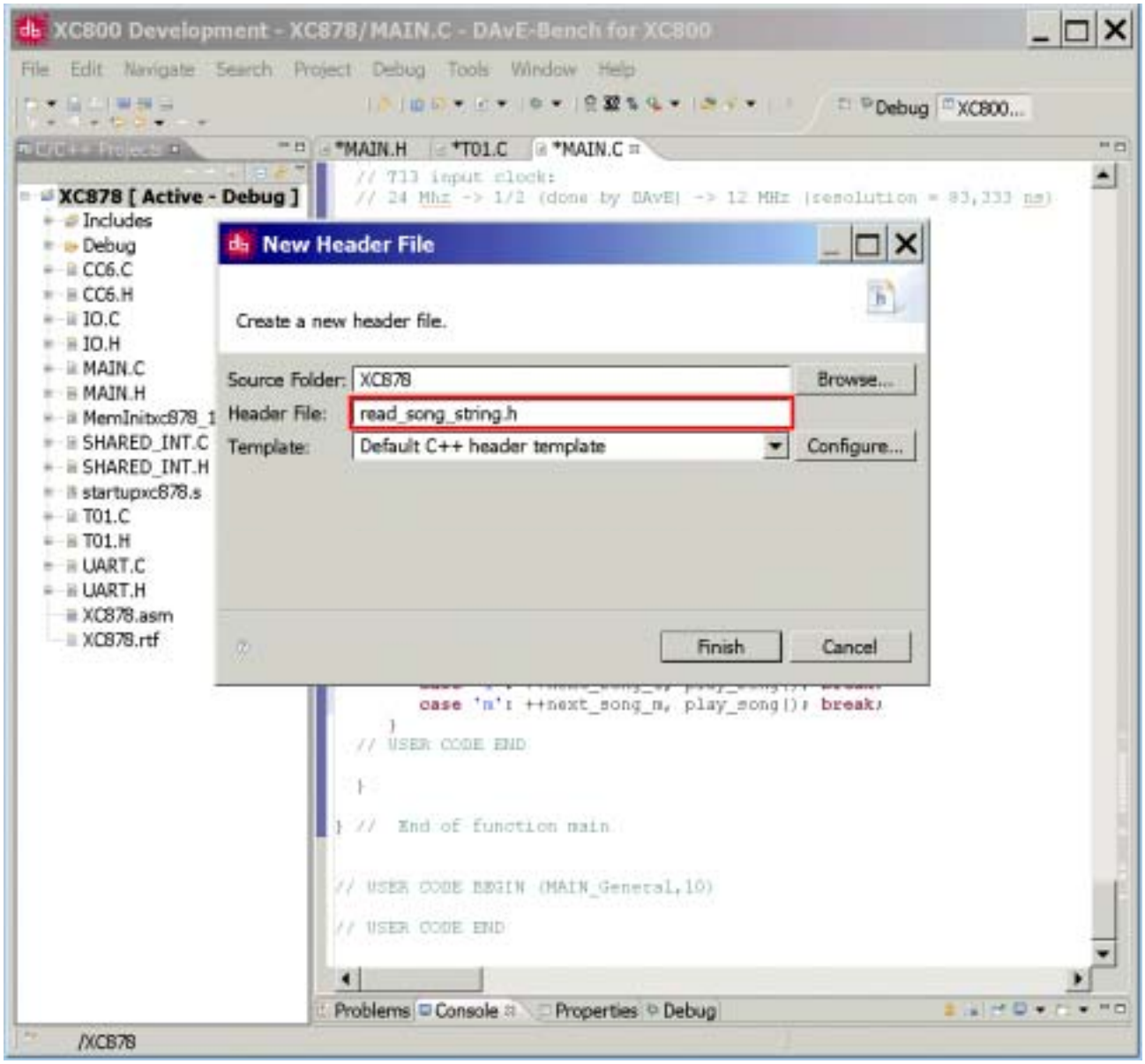
Table 15-4 SFR Address List for Pages 0-3

Address	Page 0	Page 1	Page 2	Page 3
$9A_H$	CC63SRL	CC63RL	T12MSELL	MCMOUTL
$9B_H$	CC63SRH	CC63RH	T12MSELH	MCMOUTH
$9C_H$	TCTR4L	T12PRL	IENL	ISL
$9D_H$	TCTR4H	T12PRH	IENH	ISH
$9E_H$	MCMOUTSL	T13PRL	INPL	PISEL0L
$9F_H$	MCMOUTSH	T13PRH	INPH	PISEL0H
$A4_H$	ISRL	T12DTCL	ISSL	PISEL2
$A5_H$	ISRH	T12DTCH	ISSH	
$A6_H$	CMPMODIFL	TCTR0L	PSLR	
$A7_H$	CMPMODIFH	TCTR0H	MCMCTR	
FA_H	CC60SRL	CC60RL	TCTR2L	T12L
FB_H	CC60SRH	CC60RH	TCTR2H	T12H
FC_H	CC61SRL	CC61RL	MODCTRL	T13L
FD_H	CC61SRH	CC61RH	MODCTRH	T13H
FE_H	CC62SRL	CC62RL	TRPCTRL	CMPSTATL
FF_H	CC62SRH	CC62RH	TRPCTRH	CMPSTATH

Mouse position: C/C++ Projects, XC878 [Activ - Debug]: click right mouse button
select New click Header File



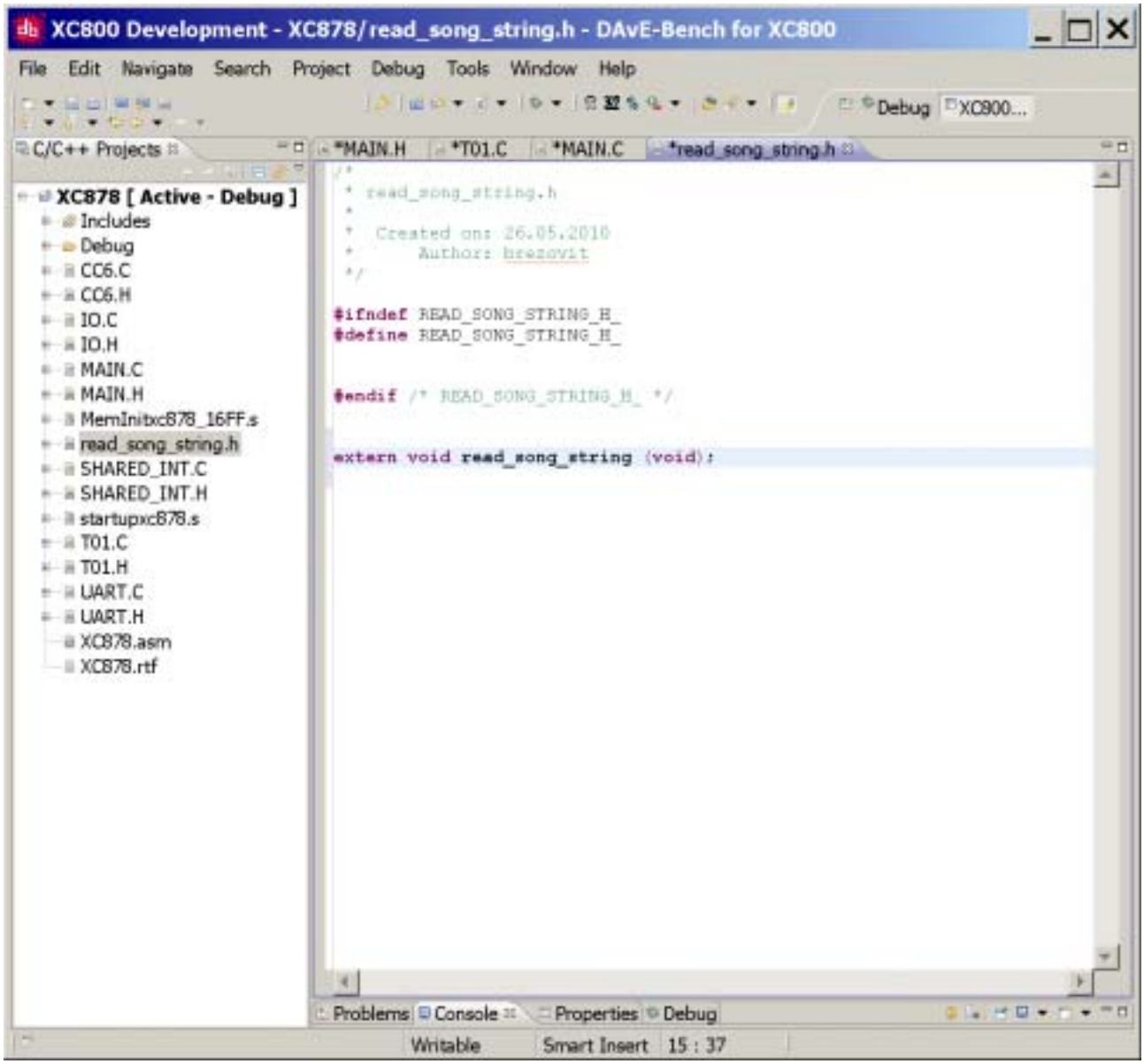
New Header File: Header File: insert: read_song_string.h



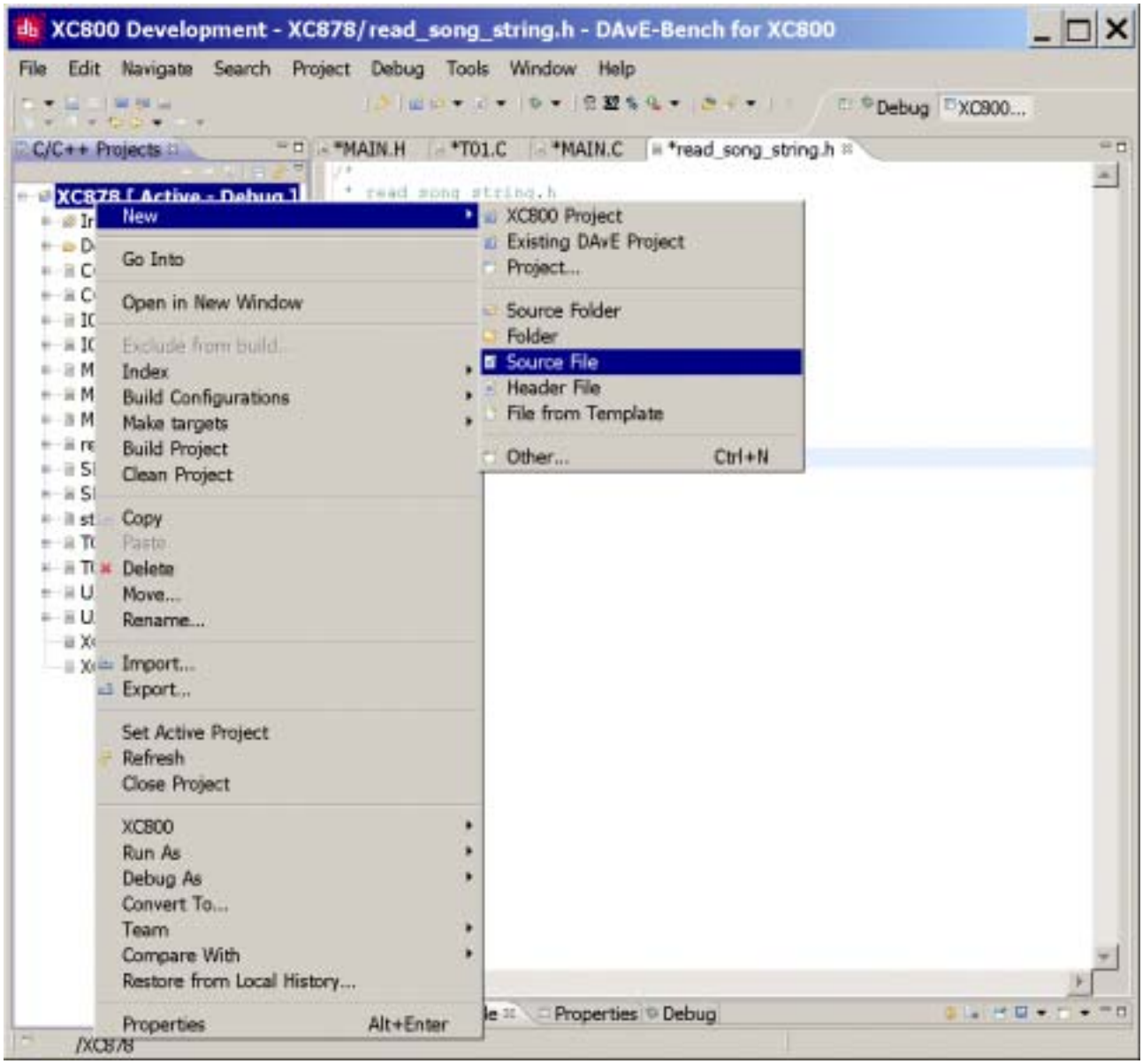
Click Finish

Insert:

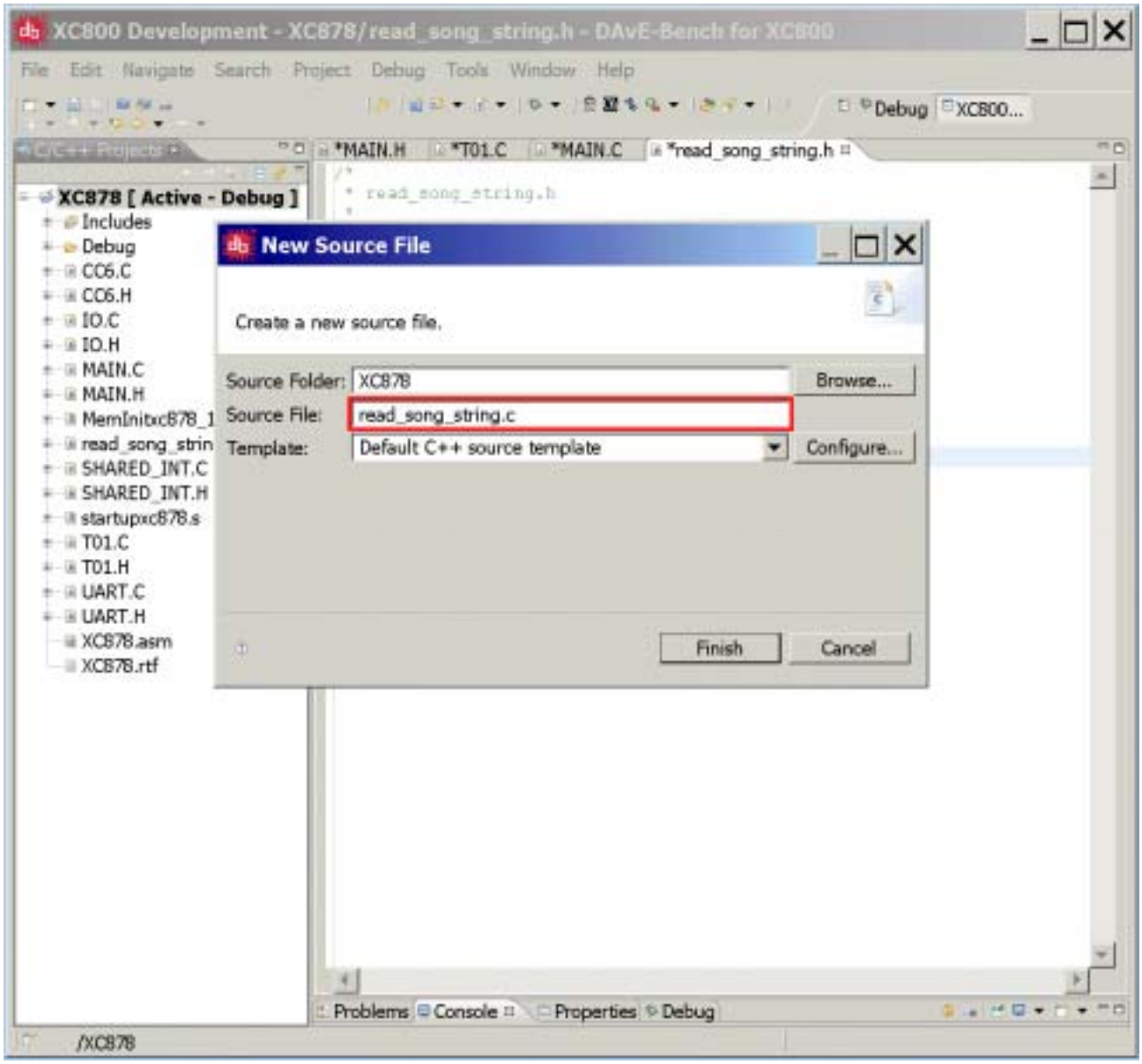
```
extern void read_song_string (void);
```



Mouse position: C/C++ Projects, XC878 [Activ - Debug]: click right mouse button
select New click Source File



New Source File: Source File: insert: read_song_string.c



Click Finish

Insert:

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

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

  SFR_PAGE(_cc0, noSST);        // switch to page 0
  CC6_vStopTmr(CC6_TIMER_13);    // Stop Timer 13: CCU6_TCTR4H |= 0x01

  SFR_PAGE(_cc1, noSST);        // switch to page 1
  CCU6_TCTR0H    =  0x01;       // prescaler = 2: load CCU6 timer 13 control register 0 high

  SFR_PAGE(_cc0, noSST);        // switch to page 0
  CC6_vStartTmr(CC6_TIMER_13);   // Start Timer 13: CCU6_TCTR4H |= 0x02
}

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

  SFR_PAGE(_cc0, noSST);        // switch to page 0
  CC6_vStopTmr(CC6_TIMER_13);    // Stop Timer 13: CCU6_TCTR4H |= 0x01

  SFR_PAGE(_cc1, noSST);        // switch to page 1
  CCU6_TCTR0H    =  0x02;       // prescaler = 4: load CCU6 timer 13 control register 0 high

  SFR_PAGE(_cc0, noSST);        // switch to page 0
  CCU6_TCTR4H    =  0x02;       // Start Timer 13: CCU6_TCTR4H |= 0x02
}

// Note:
// The function read_song_string() is a recursive function and will be called recursively until a note
is found.
// The local variable substr is only used within one function-call to determine the tempo and
// can be destroyed from one function-call to another.
// substr could also be defined as either a static or a global variable.
// Therefore, the keyword reentrant is not needed.
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;
```

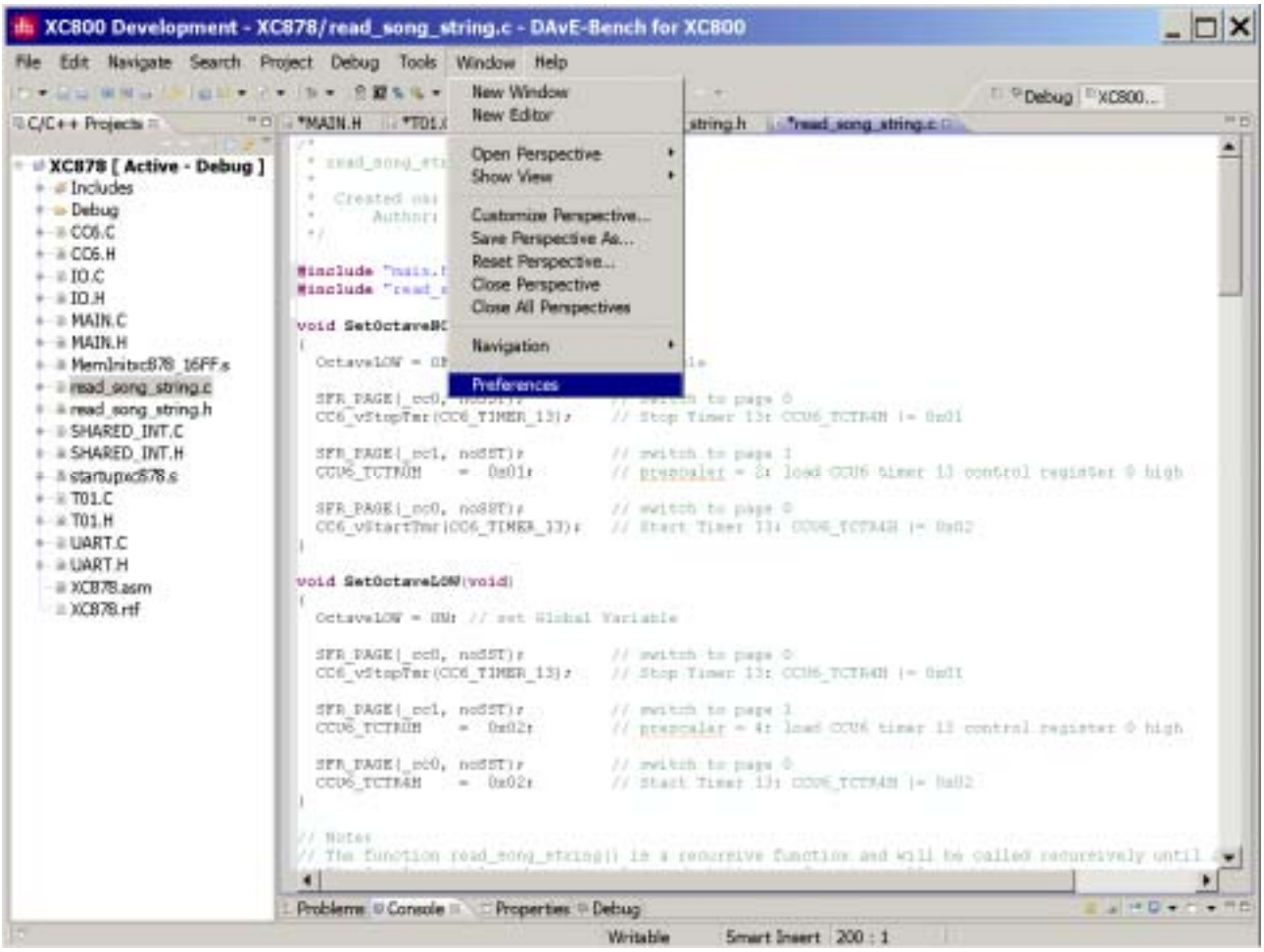
```
switch (song[++pos])
{
  case '+': note++; pos++;   break;
  case '-': octave--;
                        note=11;
                        pos++;                  break;
  default : ;           break;
}                   break;

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 */
```

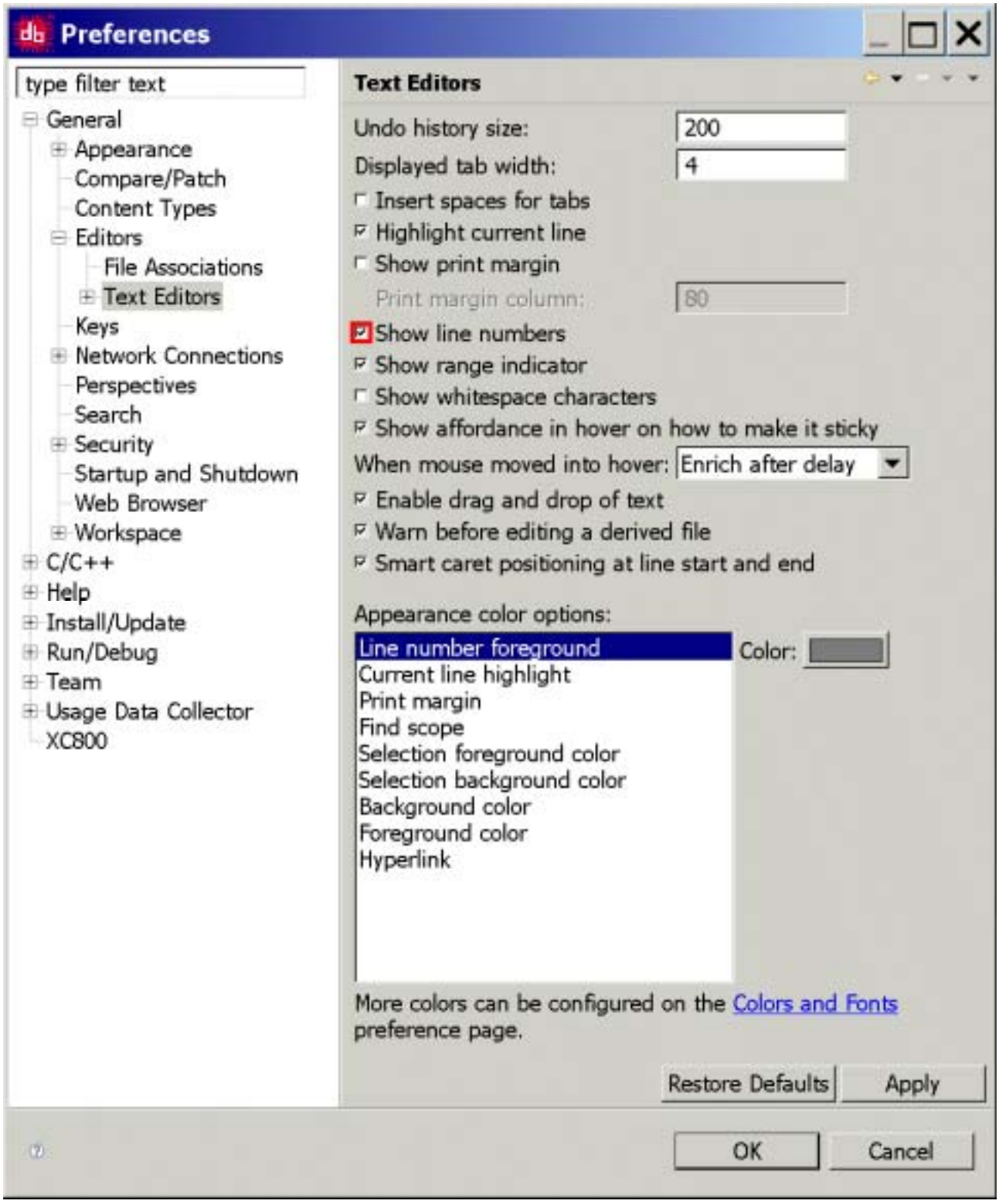
Note: Now we want to see line numbers (page 1/2):

Window - Preferences



Note: Now we want to see line numbers (page 1/2):

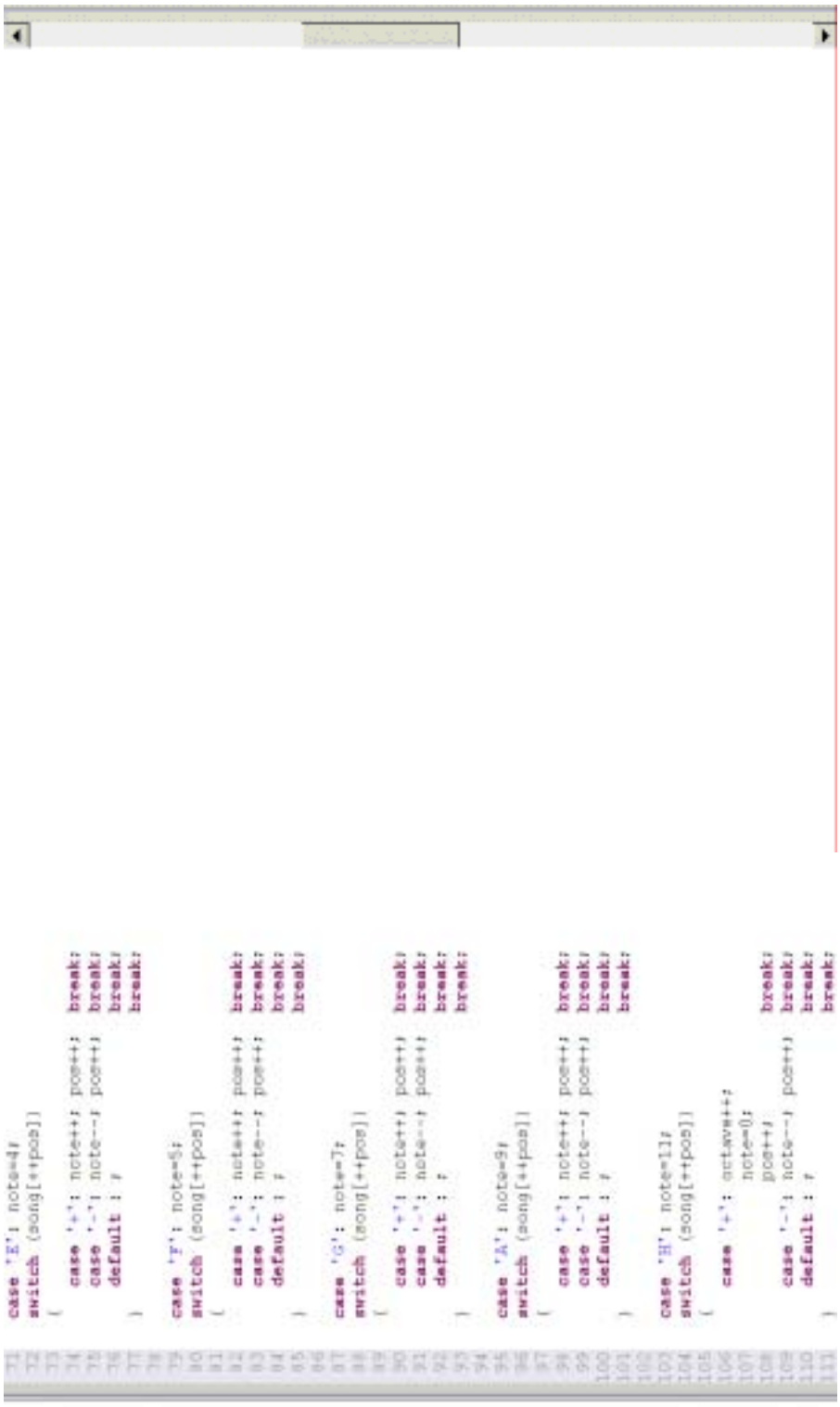
Window - Preferences
Preferences: General: Editors: Text Editors: tick Show line numbers



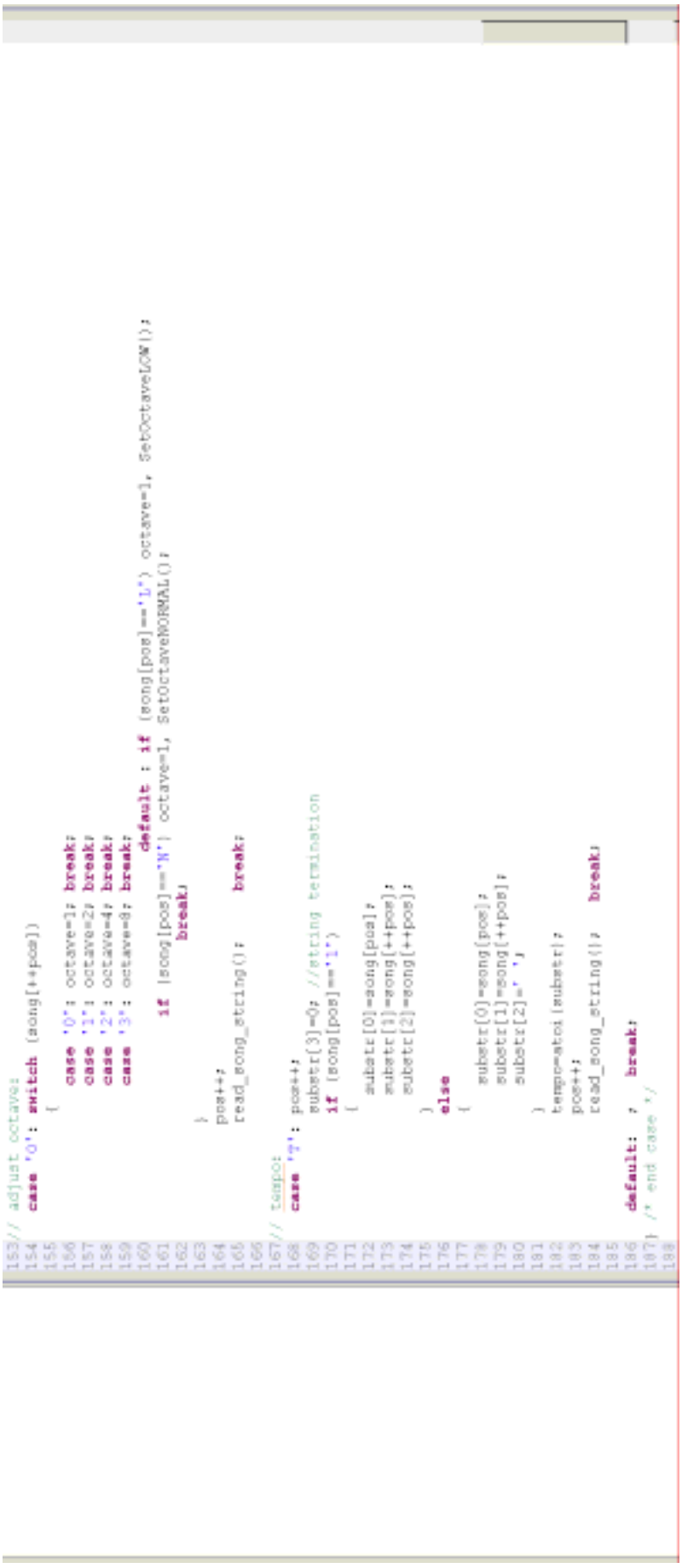
Click OK


XC800 Development - XC878/read_song_string.c - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemUnitxc878_16FF.s
read_song_string.c
read_song_string.h
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
*MAIN.H
*T01.C
*MAIN.C
*read_song_string.h
*read_song_string.c
Debug
XC800...

```
// Note:
// The function read_song_string() is a recursive function and will be called recursively until a note is found.
// The local variable substr is only used within one function-call to determine the tempo and
// can be destroyed from one function-call to another.
// substr could also be defined as either a static or a global variable.
// Therefore, the keyword reentrant is not needed.
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;
}
```

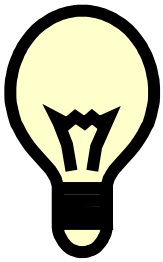


```
// adjust note length:
case 'L': switch (song[++pos])
            {
              case '1': if (song[++pos]=='6')
                          current_note_length=length_of_a_whole_note/16;
                        else
                          {
                            pos--;
                            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 : ;
            }
          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]=='I')
          {
            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 */
Problems
Console
Properties
Debug
Writable
Smart Insert
200 : 1



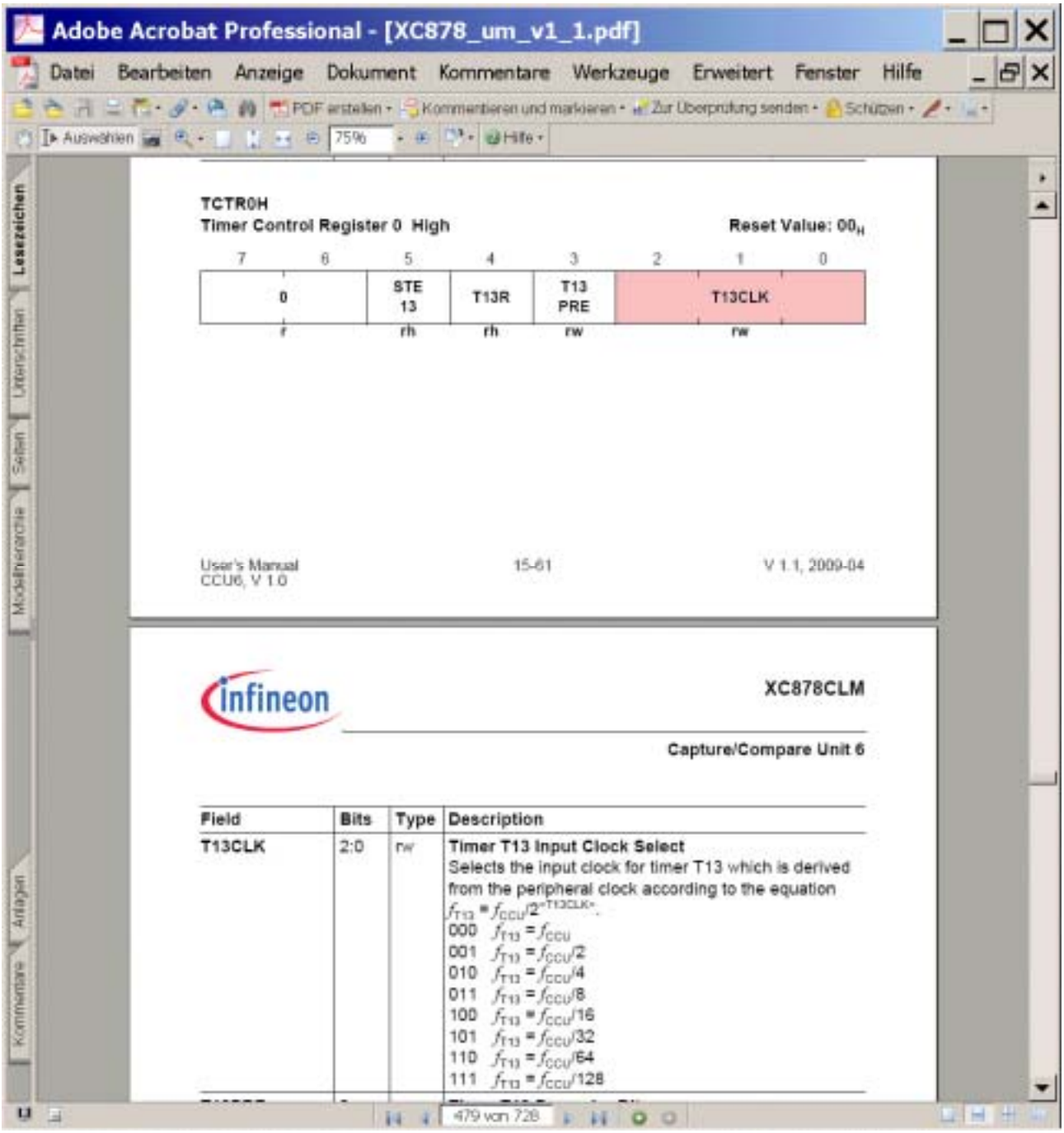
Note:
In the following code sequences

```
SFR_PAGE(_cc1, noSST);      // switch to page 1
CCU6_TCTR0H   =  0x01;      // prescaler = 2: load CCU6 timer 13 control register 0 high
```

and

```
SFR_PAGE(_cc1, noSST);      // switch to page 1
CCU6_TCTR0H   =  0x02;      // prescaler = 4: load CCU6 timer 13 control register 0 high
```

we have to access the T13CLK bit field in the TCTR0H register.


Adobe Acrobat Professional - [XC878_um_v1_1.pdf]
Datei Bearbeiten Anzeige Dokument Kommentare Werkzeuge Erweitert Fenster Hilfe
TCTR0H
Timer Control Register 0 High
Reset Value: 00H
7 6 5 4 3 2 1 0
0
STE 13
T13R
T13 PRE
T13CLK
r rh rh rw rw
User's Manual
CCU6, V 1.0
15-61
V 1.1, 2009-04
XC878CLM
Capture/Compare Unit 6
Field Bits Type Description
T13CLK 2:0 rw Timer T13 Input Clock Select
Selects the input clock for timer T13 which is derived from the peripheral clock according to the equation
000 fT13 = fCCU
001 fT13 = fCCU/2
010 fT13 = fCCU/4
011 fT13 = fCCU/8
100 fT13 = fCCU/16
101 fT13 = fCCU/32
110 fT13 = fCCU/64
111 fT13 = fCCU/128
479 von 728

```
void read_song_string (void)
{
    // code
    read_song_string();  // recursive call
    // code
}
```

Note (DAvE_Bench_XC800_Release_Notes.doc, Limitations):

> Functions are static by default and are non-reentrant.

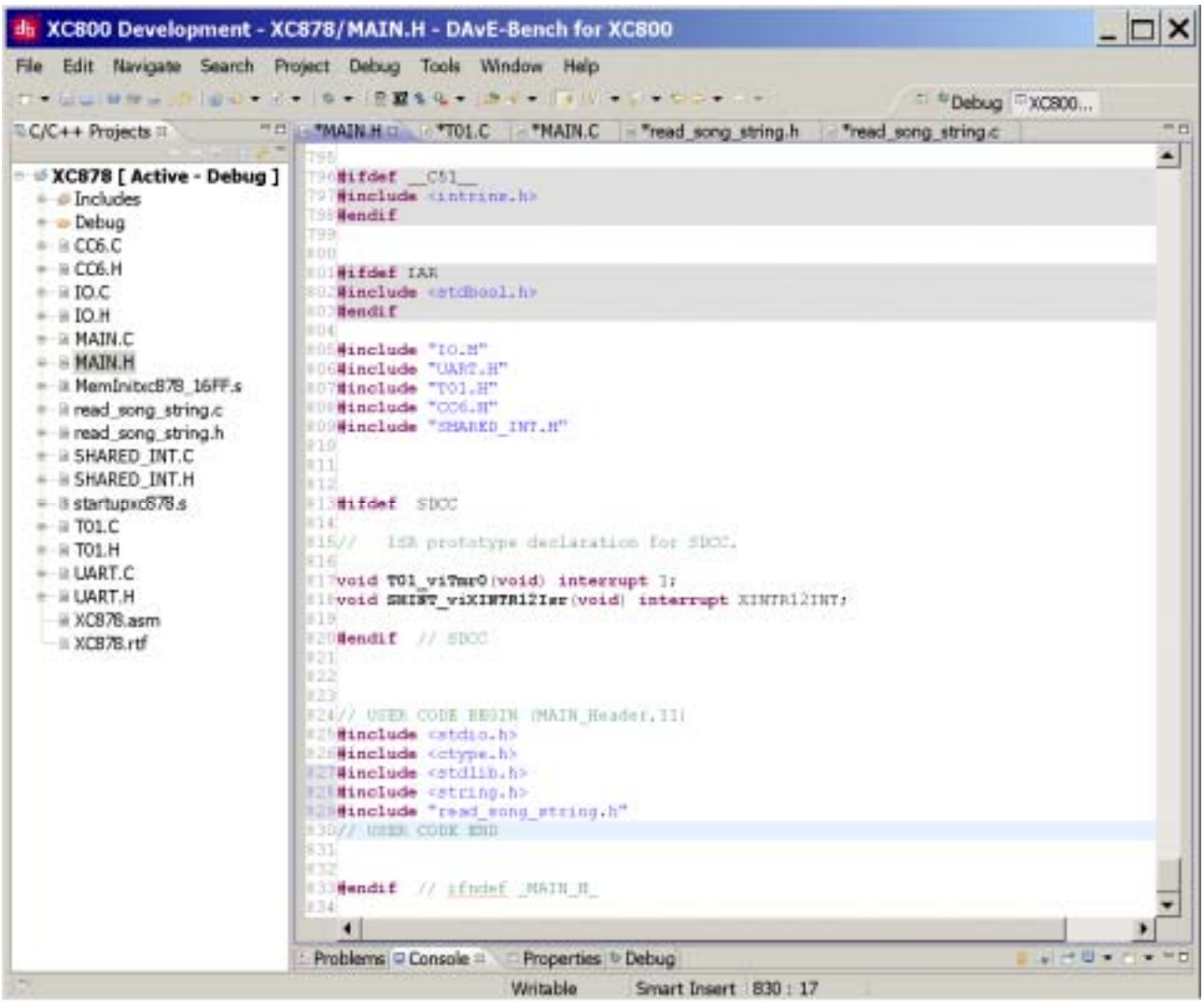
Note:
Function read_song_string calls itself until a note is found (recursive function).
The break condition for the recursion is that a note is found.

Note:
Normally, functions in DAvE-Bench cannot be called recursively or in a fashion which causes reentrancy. The reason for this limitation is that function arguments and local variables are stored in fixed memory locations. Recursive calls to the function use the same memory locations. And, in this case, arguments and locals would get corrupted.

Note:
In our case we do not use any function arguments or any local variables which must be saved.
The function read_song_string only sets global variables for note length, rest, octave, tempo, and extend note length by half.

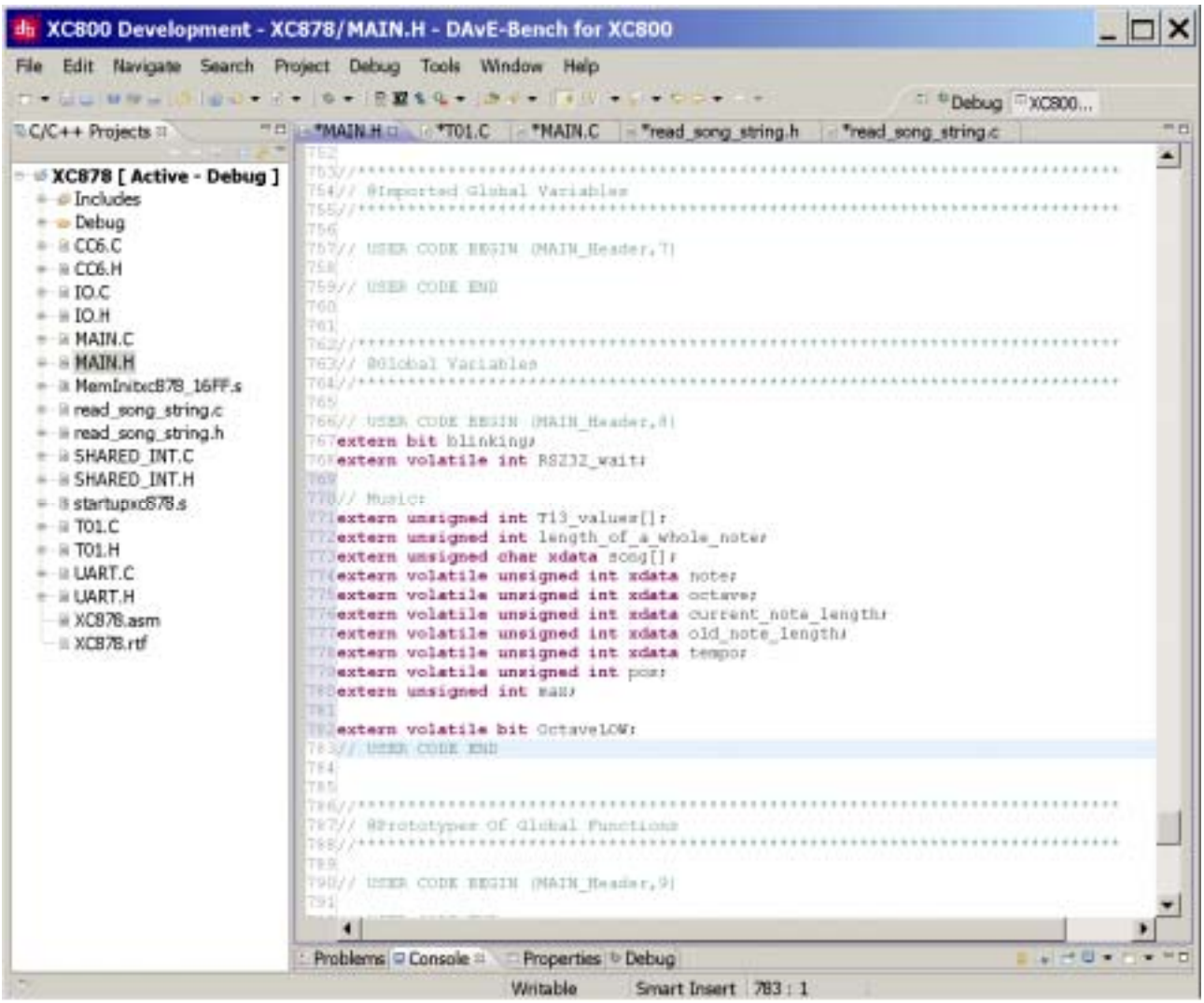
Double click MAIN.H and insert Project Includes:

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



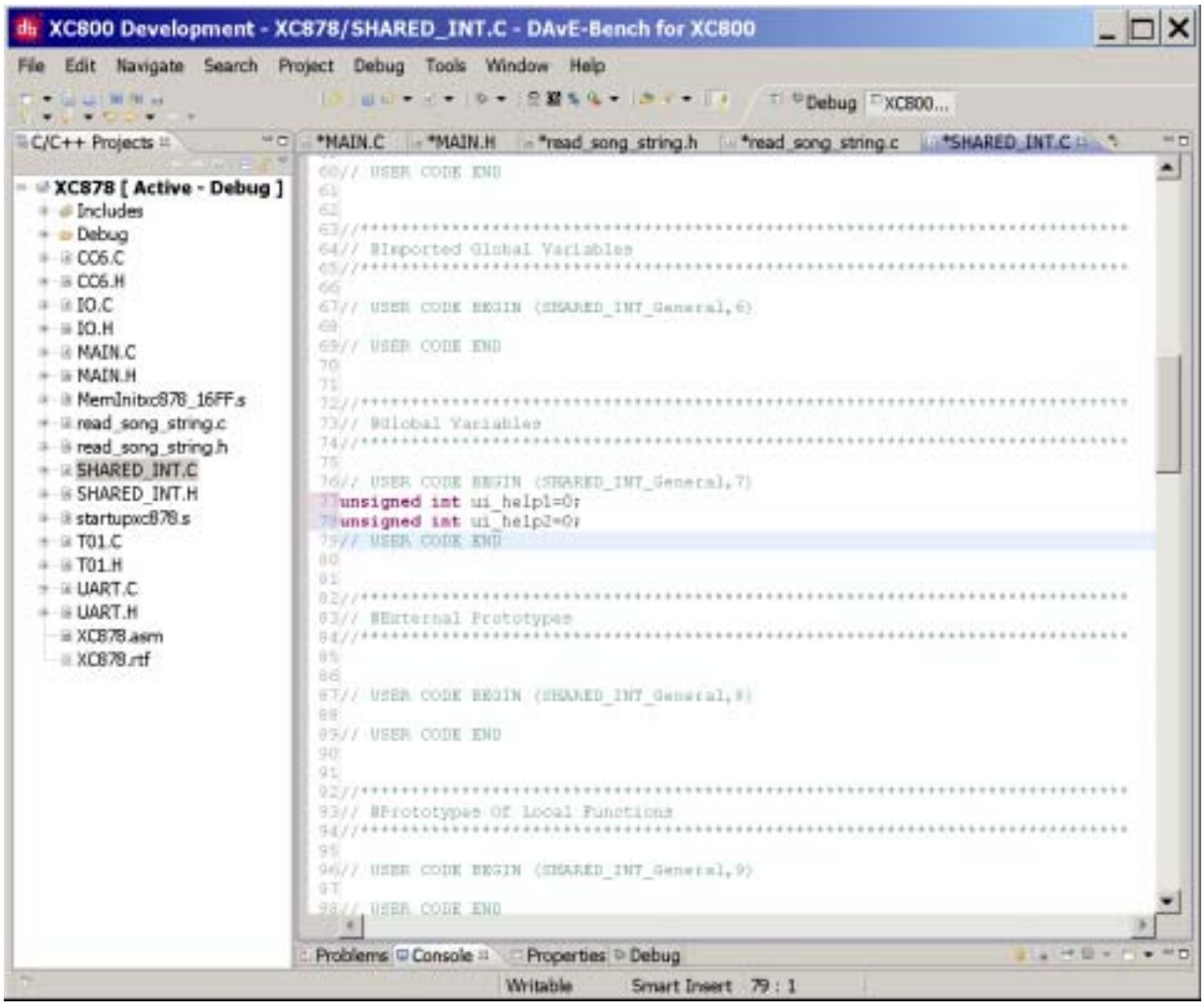
Double click MAIN.H and insert Global Variables (extern declaration):

```
// Music:
extern unsigned int T13_values[];
extern unsigned int length_of_a_whole_note;
extern unsigned char xdata song[];
extern volatile unsigned int xdata note;
extern volatile unsigned int xdata octave;
extern volatile unsigned int xdata current_note_length;
extern volatile unsigned int xdata old_note_length;
extern volatile unsigned int xdata tempo;
extern volatile unsigned int pos;
extern unsigned int max;

extern volatile bit OctaveLOW;
```



Double click SHARED_INT.C and insert Global Variables:

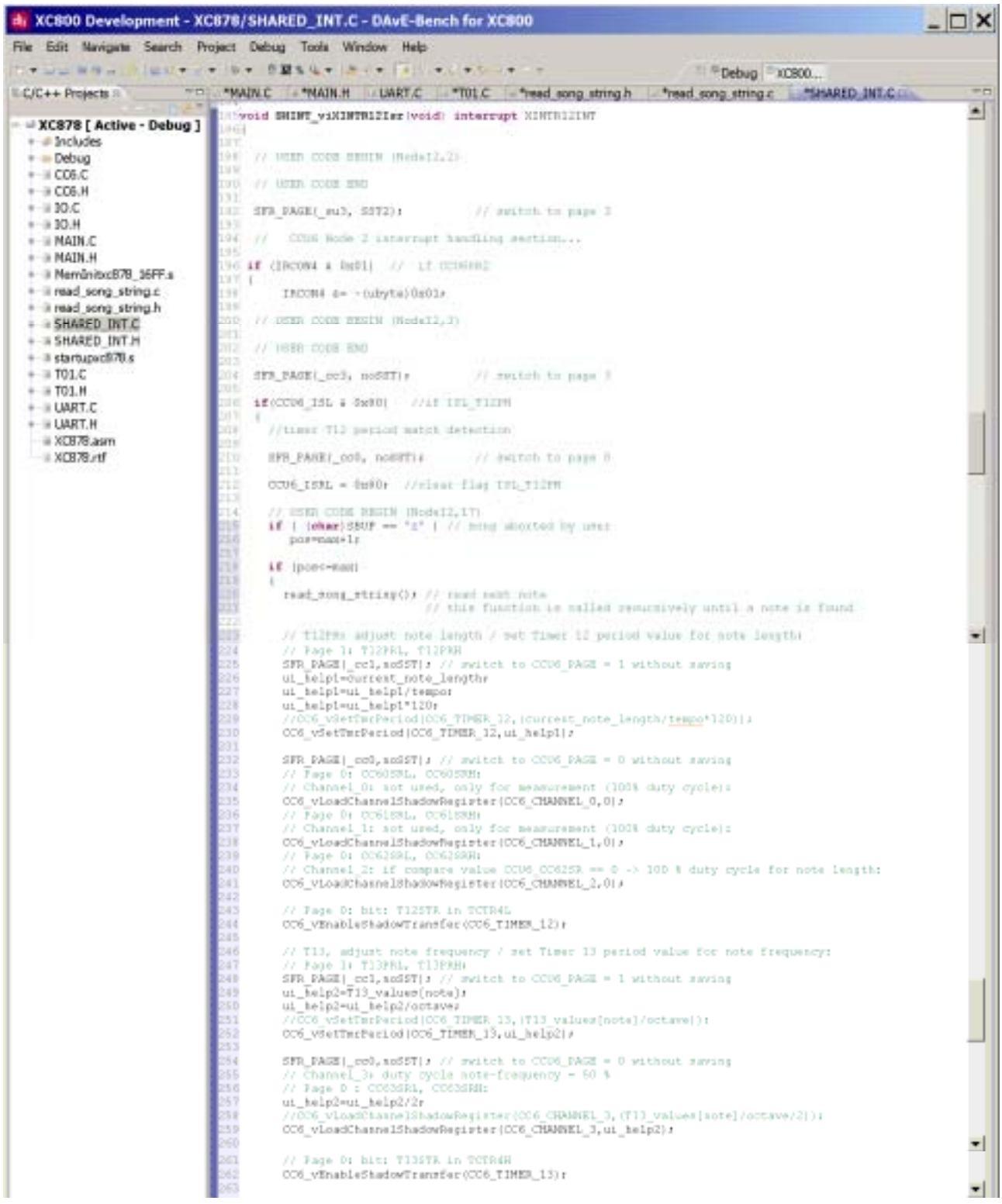
```
unsigned int ui_help1=0;
unsigned int ui_help2=0;
```



Double click SHARED_INT.C and insert ISR-Code (timer T12 period match):

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

if (pos<=max)
{
  read_song_string(); // read next note
                      // this function is called recursively until a note is
found

  // T12PR: adjust note length / set Timer 12 period value for note length:
  // Page 1: T12PRL, T12PRH
  SFR_PAGE(_cc1,noSST); // switch to CCU6_PAGE = 1 without saving
  ui_help1=current_note_length;
  ui_help1=ui_help1/tempo;
  ui_help1=ui_help1*120;
  //CC6_vSetTmrPeriod(CC6_TIMER_12,(current_note_length/tempo*120));
  CC6_vSetTmrPeriod(CC6_TIMER_12,ui_help1);

  SFR_PAGE(_cc0,noSST); // switch to CCU6_PAGE = 0 without saving
  // Page 0: CC60SRL, CC60SRH:
  // Channel_0: not used, only for measurement (100% duty cycle):
  CC6_vLoadChannelShadowRegister(CC6_CHANNEL_0,0);
  // Page 0: CC61SRL, CC61SRH:
  // Channel_1: not used, only for measurement (100% duty cycle):
  CC6_vLoadChannelShadowRegister(CC6_CHANNEL_1,0);
  // Page 0: CC62SRL, CC62SRH:
  // Channel_2: if compare value CCU6_CC62SR == 0 -> 100 % duty cycle for note
length:
  CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2,0);

  // Page 0: bit: T12STR in TCTR4L
  CC6_vEnableShadowTransfer(CC6_TIMER_12);

  // T13, adjust note frequency / set Timer 13 period value for note frequency:
  // Page 1: T13PRL, T13PRH:
  SFR_PAGE(_cc1,noSST); // switch to CCU6_PAGE = 1 without saving
  ui_help2=T13_values[note];
  ui_help2=ui_help2/octave;
  //CC6_vSetTmrPeriod(CC6_TIMER_13,(T13_values[note]/octave));
  CC6_vSetTmrPeriod(CC6_TIMER_13,ui_help2);

  SFR_PAGE(_cc0,noSST); // switch to CCU6_PAGE = 0 without saving
  // Channel_3: duty cycle note-frequency = 50 %
  // Page 0 : CC63SRL, CC63SRH:
  ui_help2=ui_help2/2;
  //CC6_vLoadChannelShadowRegister(CC6_CHANNEL_3,(T13_values[note]/octave/2));
  CC6_vLoadChannelShadowRegister(CC6_CHANNEL_3,ui_help2);

  // Page 0: bit: T13STR in TCTR4H
  CC6_vEnableShadowTransfer(CC6_TIMER_13);

  if      (note == 0) printf_small("note=c  ");
  else if (note == 1) printf_small("note=cis");
  else if (note == 2) printf_small("note=d  ");
  else if (note == 3) printf_small("note=dis");
  else if (note == 4) printf_small("note=e  ");
  else if (note == 5) printf_small("note=f  ");
  else if (note == 6) printf_small("note=fis");
  else if (note == 7) printf_small("note=g  ");
  else if (note == 8) printf_small("note=gis");
  else if (note == 9) printf_small("note=a  ");
```

```
  else if (note ==10) printf_small("note=ais");
  else if (note ==11) printf_small("note=h  ");
  else if (note ==12) printf_small("note=---");
  else                printf_small("note=???");

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

  printf_fast_f(", T12-pv=%5u,",current_note_length);
  printf_fast_f("T12-p=%1.2f[s], ",current_note_length*85.3333/1000.0/1000.0);
  printf_fast_f("T13-pv=%5u,", T13_values[note]/octave);
  if (OctaveLOW==OFF)
  printf_fast_f("T13-
f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*83.3333/1000.0/1000.0/1000.0));
  else if (OctaveLOW==ON)
  printf_fast_f("T13-
f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*166.6667/1000.0/1000.0/1000.0));

  IO_vTogglePin(P3_1); // Show start of next note on Port 3 Pin 1
  // Page 0: bit: T12RS in TCTR4L
  CC6_vStartTmr(CC6_TIMER_12); // Set Timer 12 Run Set bit T12RS
}
```


XC800 Development - XC878/SHARED_INT.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_36FF.s
read_song_string.c
read_song_string.h
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
*MAIN.C
*MAIN.H
UART.C
*T01.C
*read_song_string.h
*read_song_string.c
*SHARED_INT.C

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

    if      (octave == 1 && OctaveLOW==OFF) printf_small("*00*");
    else if (octave == 1 && OctaveLOW== ON) printf_small("*01*");
    else if (octave == 2) printf_small("*01*");
    else if (octave == 4) printf_small("*02*");
    else if (octave == 8) printf_small("*03*");
    else                  printf_small("????");

    printf_fast_f(", T12-pw=%5u,",current_note_length);
    printf_fast_f("T12-p=%1.2f[s], ",current_note_length*85.3333/1000.0/1000.0);
    printf_fast_f("T13-pw=%5u,", T13_values[note]/octave);
    if (OctaveLOW==OFF)
    printf_fast_f("T13-f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*83.3333/1000.0/1000.0/1000.0));
    else if (OctaveLOW==ON)
    printf_fast_f("T13-f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*166.6667/1000.0/1000.0/1000.0));

    IO_vTogglePin(P3_1); // Show start of next note on Port 3 Pin 1
    // Page 0: bit: T12RS is TCTRAL
    CC6_vStartTmr(CC6_TIMER_12); // Set Timer 12 Run Set bit T12RS
  }
  // USER CODE END
}

}          // End of CCU6SR2 condition check

// USER CODE BEGIN (Module12,5)

// USER CODE END

SFR_PAGE(_su5, RST2);          // restore the old SCU page
} // End of function SHINT_vLEINTR12Isr

// USER CODE BEGIN (SHARED_INT_General,10)

// USER CODE END
```

Problems | Console | Properties | Debug

Writable | Smart Insert | 299 : 1

Note:

IO_Port_3.1 will be toggled when a new note is started.

Registers used:

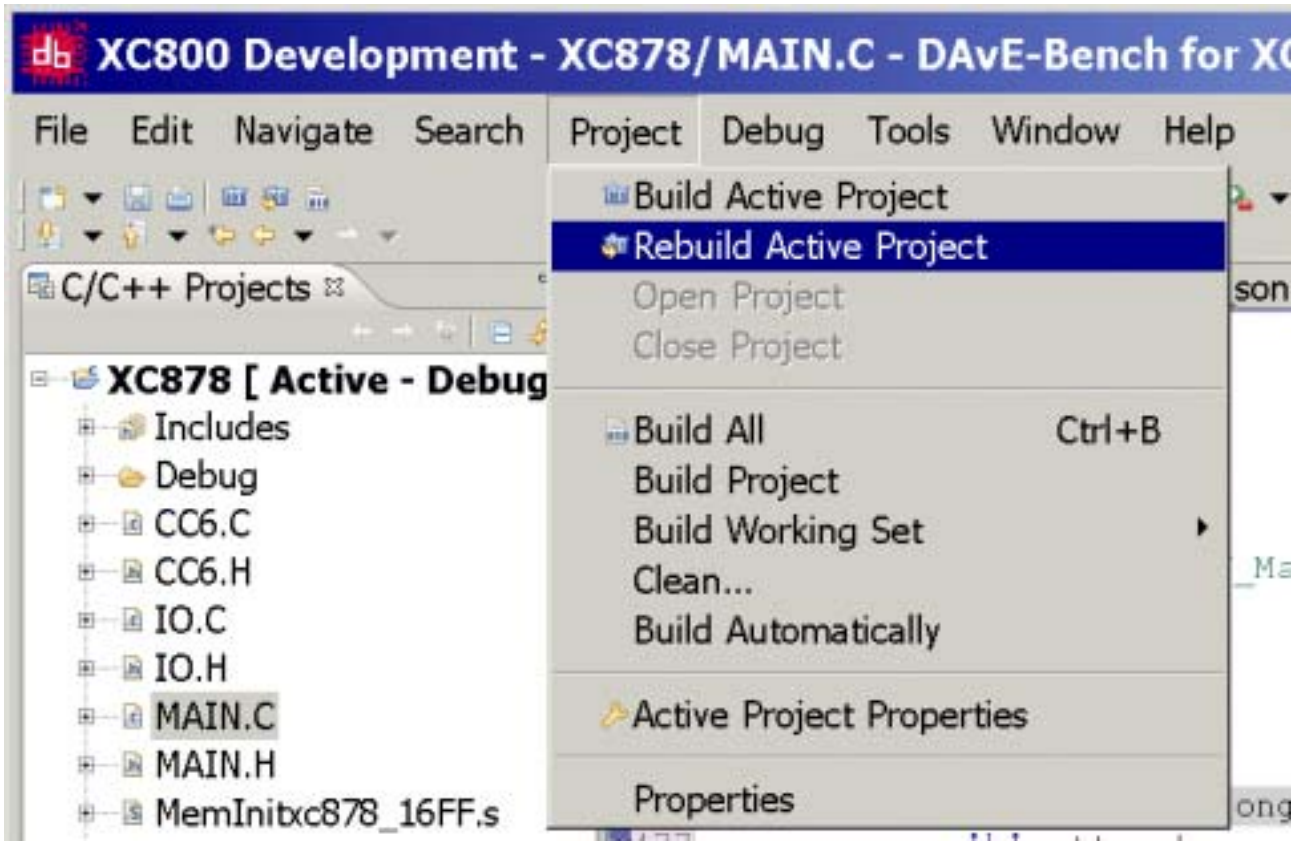
Adobe Acrobat Professional - [XC878_um_v1_1.pdf]

Table 15-4 SFR Address List for Pages 0-3

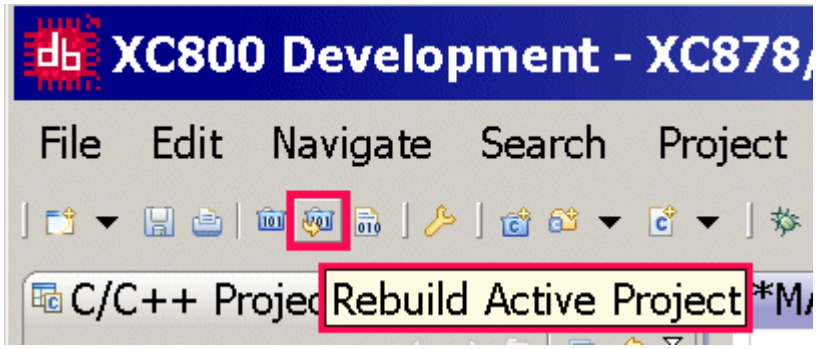
Address	Page 0	Page 1	Page 2	Page 3
$9A_H$	CC63SRL	CC63RL	T12MSELL	MCMOUTL
$9B_H$	CC63SRH	CC63RH	T12MSELH	MCMOUTH
$9C_H$	TCTR4L	T12PRL	IENL	ISL
$9D_H$	TCTR4H	T12PRH	IENH	ISH
$9E_H$	MCMOUTSL	T13PRL	INPL	PISEL0L
$9F_H$	MCMOUTSH	T13PRH	INPH	PISEL0H
$A4_H$	ISRL	T12DTCL	ISSL	PISEL2
$A5_H$	ISRH	T12DTCH	ISSH	
$A6_H$	CMPMODIFL	TCTR0L	PSLR	
$A7_H$	CMPMODIFH	TCTR0H	MCMCTR	
FA_H	CC60SRL	CC60RL	TCTR2L	T12L
FB_H	CC60SRH	CC60RH	TCTR2H	T12H
FC_H	CC61SRL	CC61RL	MODCTRL	T13L
FD_H	CC61SRH	CC61RH	MODCTRH	T13H
FE_H	CC62SRL	CC62RL	TRPCTRL	CMPSTATL
FF_H	CC62SRH	CC62RH	TRPCTRH	CMPSTATH

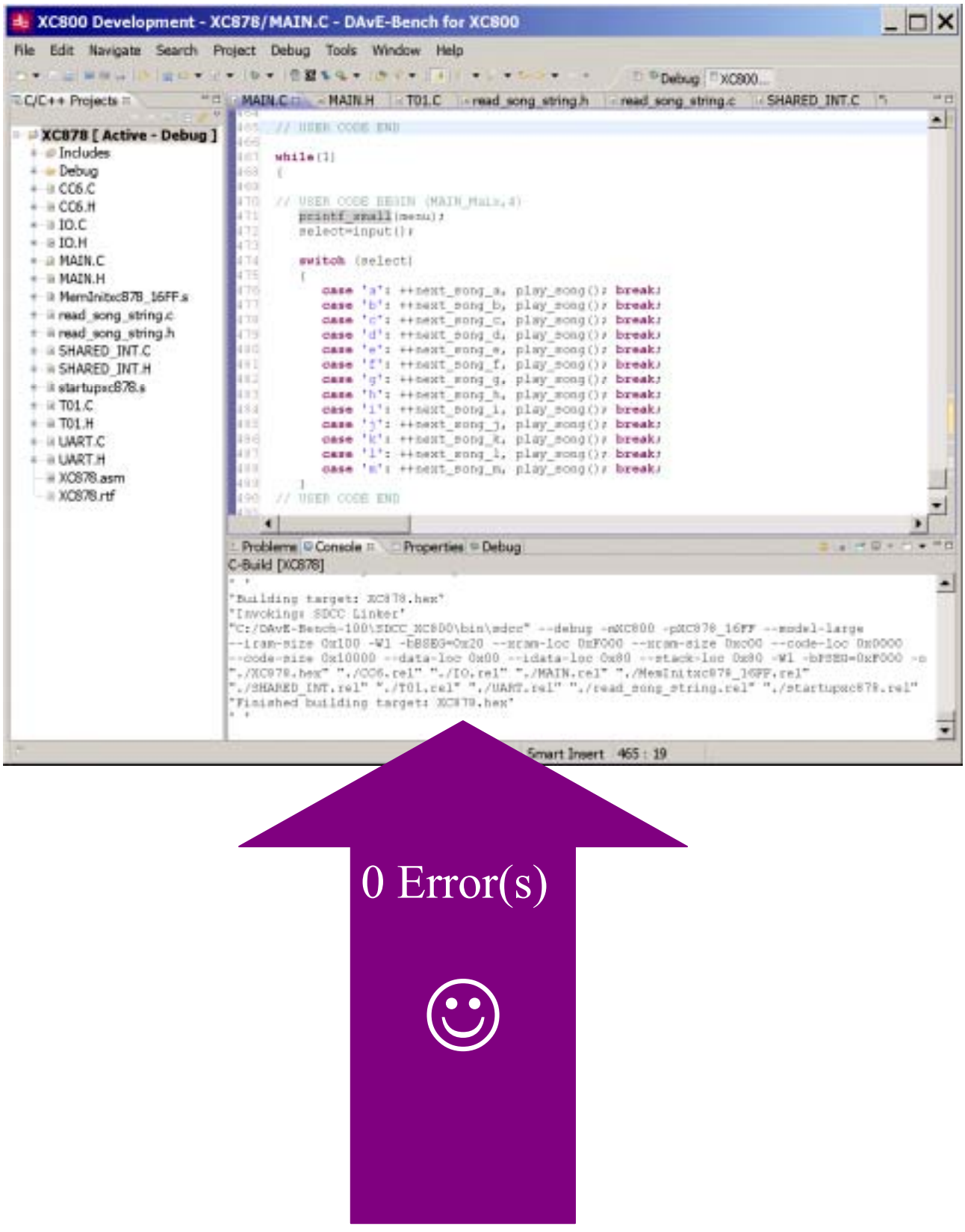
Generate your application program:

Project – Rebuild Active Project



or: click 

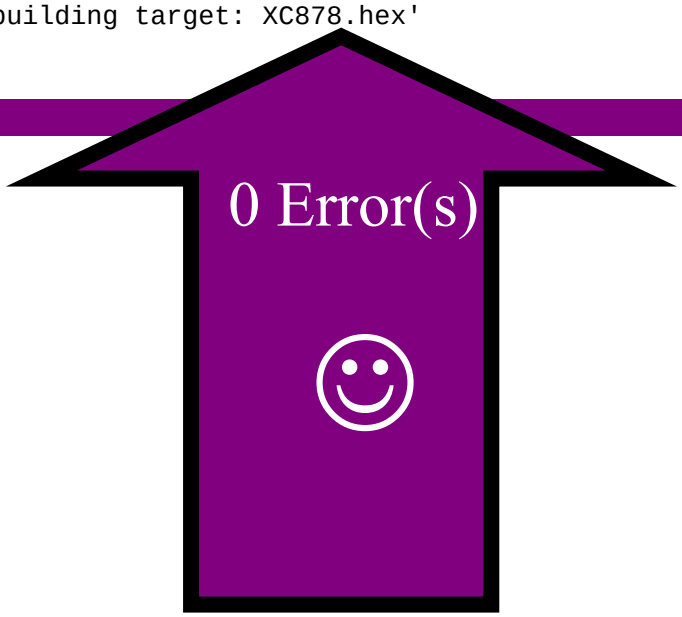



XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
read_song_string.c
read_song_string.h
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.C
MAIN.H
T01.C
read_song_string.h
read_song_string.c
SHARED_INT.C
// USER CODE END
while(1)
{
// USER CODE BEGIN (MAIN_Main,4)
printf_small(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
Problems Console Properties Debug
C-Build [XC878]
"Building target: XC878.hex"
"Invoking: SDCC Linker"
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" --debug -mXC800 -pXC878_16FF --model-large
--iram-size 0x100 -Wl -bBSEG=0x20 --xram-loc 0xF000 --xram-size 0xc00 --code-loc 0x0000
--code-size 0x10000 --data-loc 0x00 --idata-loc 0x80 --stack-loc 0x80 -Wl -bPSEG=0xF000 -o
"./XC878.hex" "./CC6.rel" "./IO.rel" "./MAIN.rel" "./MemInitxc878_16FF.rel"
"./SHARED_INT.rel" "./T01.rel" "./UART.rel" "./read_song_string.rel" "./startupxc878.rel"
"Finished building target: XC878.hex"
Smart Insert 465 : 19
0 Error(s)

```
**** Build of configuration Debug for project XC878 ****

C:\DAvE-Bench-100\SDCC_UTILS\make all
'Building file: ../CC6.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "CC6.s" "../CC6.C"
'Finished building: ../CC6.C'
' '
'Building file: CC6.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "CC6.s" -O "CC6.rel"
'Finished building: CC6.s'
' '
'Building file: ../IO.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "IO.s" "../IO.C"
'Finished building: ../IO.C'
' '
'Building file: IO.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "IO.s" -O "IO.rel"
'Finished building: IO.s'
' '
'Building file: ../MAIN.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "MAIN.s" "../MAIN.C"
../MAIN.C:375: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:378: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:381: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:384: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:387: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:390: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:393: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:396: warning 94: comparison is always true due to limited range of
```

```
data type
../MAIN.C:402: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:408: warning 94: comparison is always true due to limited range of
data type
../MAIN.C:411: warning 94: comparison is always true due to limited range of
data type
'Finished building: ../MAIN.C'
' '
'Building file: MAIN.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "MAIN.s" -O
"MAIN.rel"
'Finished building: MAIN.s'
' '
'Building file: ../MemInitxc878_16FF.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx
"../MemInitxc878_16FF.s" -O "MemInitxc878_16FF.rel"
'Finished building: ../MemInitxc878_16FF.s'
' '
'Building file: ../SHARED_INT.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "SHARED_INT.s" "../SHARED_INT.C"
'Finished building: ../SHARED_INT.C'
' '
'Building file: SHARED_INT.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "SHARED_INT.s" -O
"SHARED_INT.rel"
'Finished building: SHARED_INT.s'
' '
'Building file: ../T01.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "T01.s" "../T01.C"
'Finished building: ../T01.C'
' '
'Building file: T01.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "T01.s" -O "T01.rel"
'Finished building: T01.s'
' '
'Building file: ../UART.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "UART.s" "../UART.C"
'Finished building: ../UART.C'
' '
'Building file: UART.s'
```

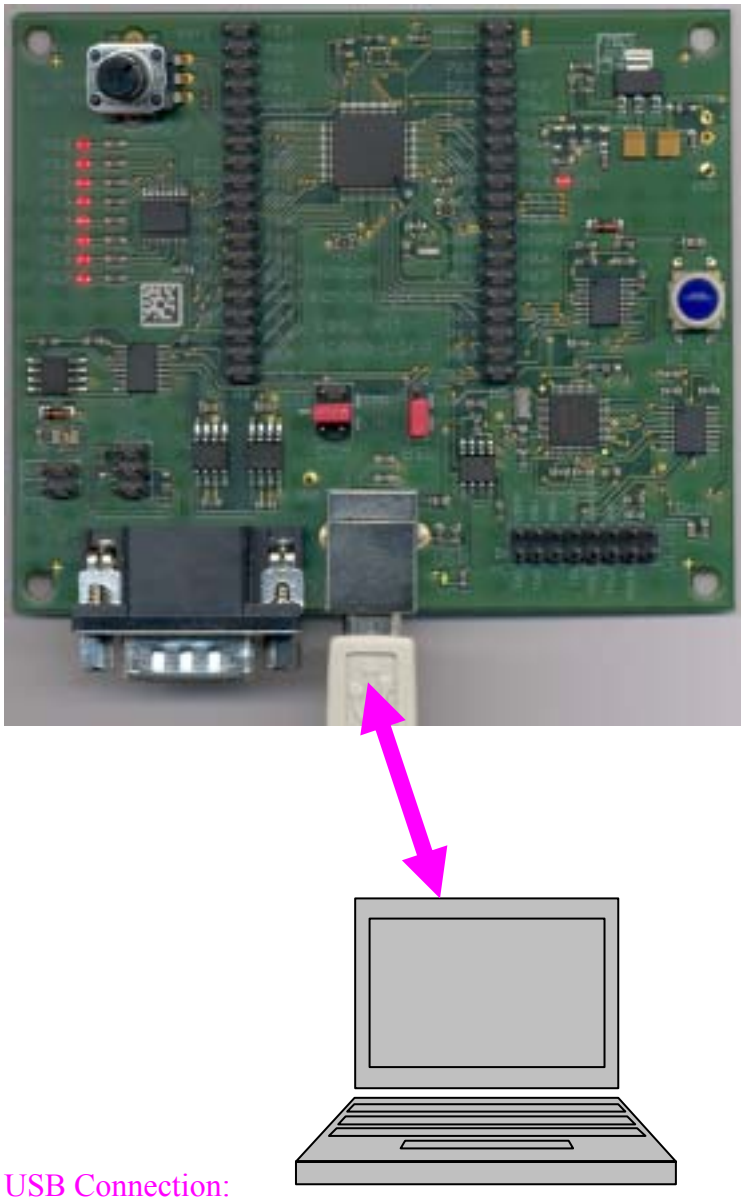
```
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "UART.s" -O
"UART.rel"
MOV dir(0x82),dir(0x99) found at 1641 of UART.s
'Finished building: UART.s'
' '
'Building file: ../read_song_string.C'
'Invoking: SDCC Compiler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" -mXC800 -pXC878_16FF --model-large
-I"C:/DAvE-Bench-100\SDCC_XC800\include" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\xc800" -I"C:/DAvE-Bench-
100\SDCC_XC800\include\asm\xc800" --opt-code-size --nooverlay --noinduction
--debug -S  -o "read_song_string.s" "../read_song_string.C"
'Finished building: ../read_song_string.C'
' '
'Building file: read_song_string.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "read_song_string.s"
-O "read_song_string.rel"
MOV dir(0xf0),dir(0x83) found at 3097 of read_song_string.s
MOV dir(0xf0),dir(0x83) found at 3208 of read_song_string.s
'Finished building: read_song_string.s'
' '
'Building file: ../startupxc878.s'
'Invoking: SDCC Assembler'
"C:/DAvE-Bench-100\SDCC_XC800\bin\as-xc800" -plosgffcx "../startupxc878.s" -
O "startupxc878.rel"
'Finished building: ../startupxc878.s'
' '
'Building target: XC878.hex'
'Invoking: SDCC Linker'
"C:/DAvE-Bench-100\SDCC_XC800\bin\sdcc" --debug -mXC800 -pXC878_16FF --
model-large --iram-size 0x100 -Wl -bBSEG=0x20 --xram-loc 0xF000 --xram-size
0xc00 --code-loc 0x0000 --code-size 0x10000 --data-loc 0x00 --idata-loc 0x80
--stack-loc 0x80 -Wl -bPSEG=0xF000 -o "./XC878.hex" "./CC6.rel" "./IO.rel"
"./MAIN.rel" "./MemInitxc878_16FF.rel" "./SHARED_INT.rel" "./T01.rel"
"./UART.rel" "./read_song_string.rel" "./startupxc878.rel"
'Finished building target: XC878.hex'
' '
```



2.3) Using the debugger (DAvE Bench):
See and hear the result:

We will use
any Terminal Program (e.g. U-SPY) + any Logic Analyser / scope + loudspeakers

Make sure that the XC878 Easy Kit is still connected to the host computer:

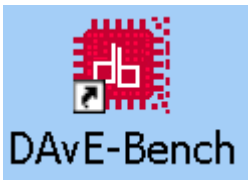


USB Connection:

.) used for: UART communication (the 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).

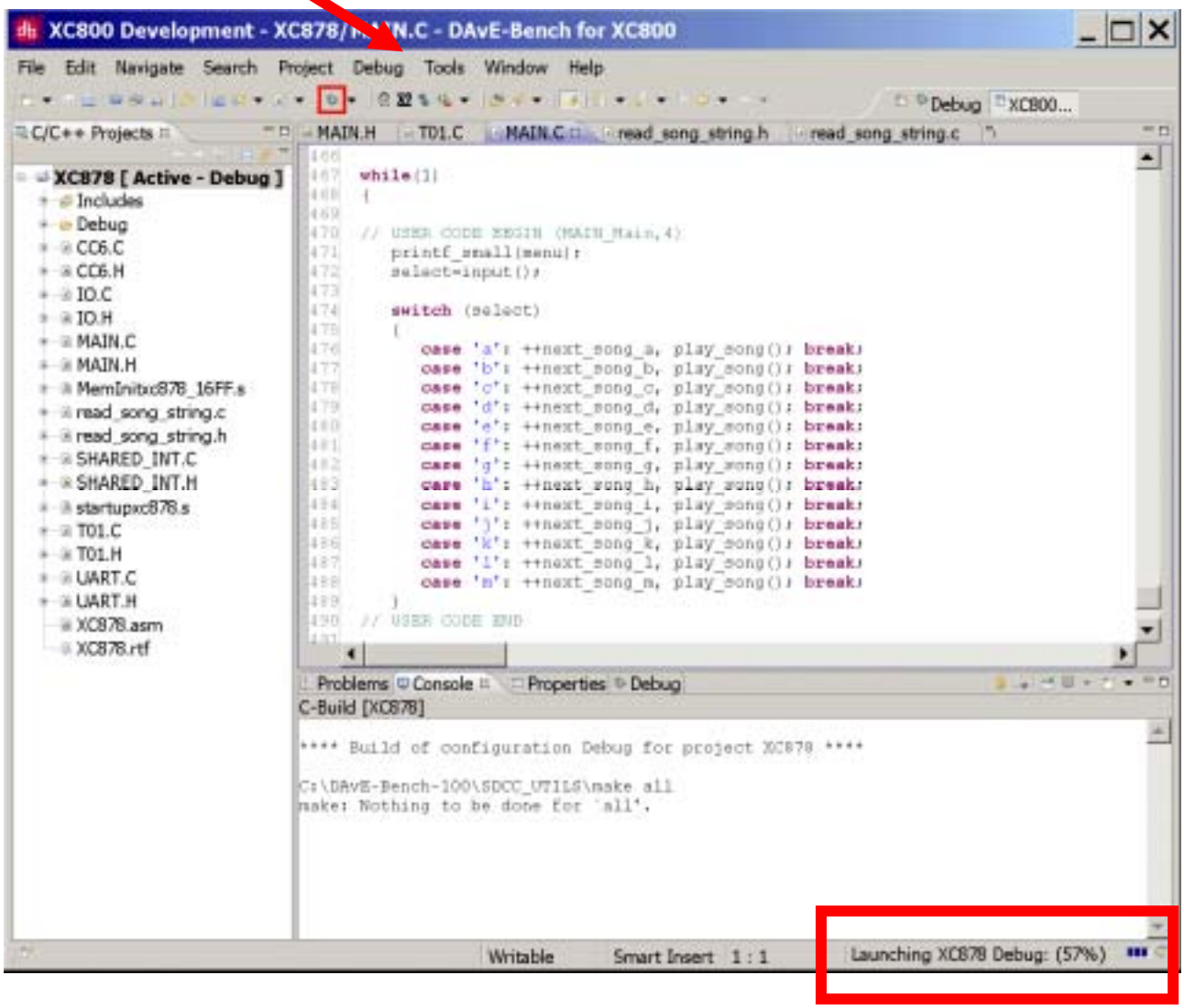
.) the USB connection works also as the power supply.

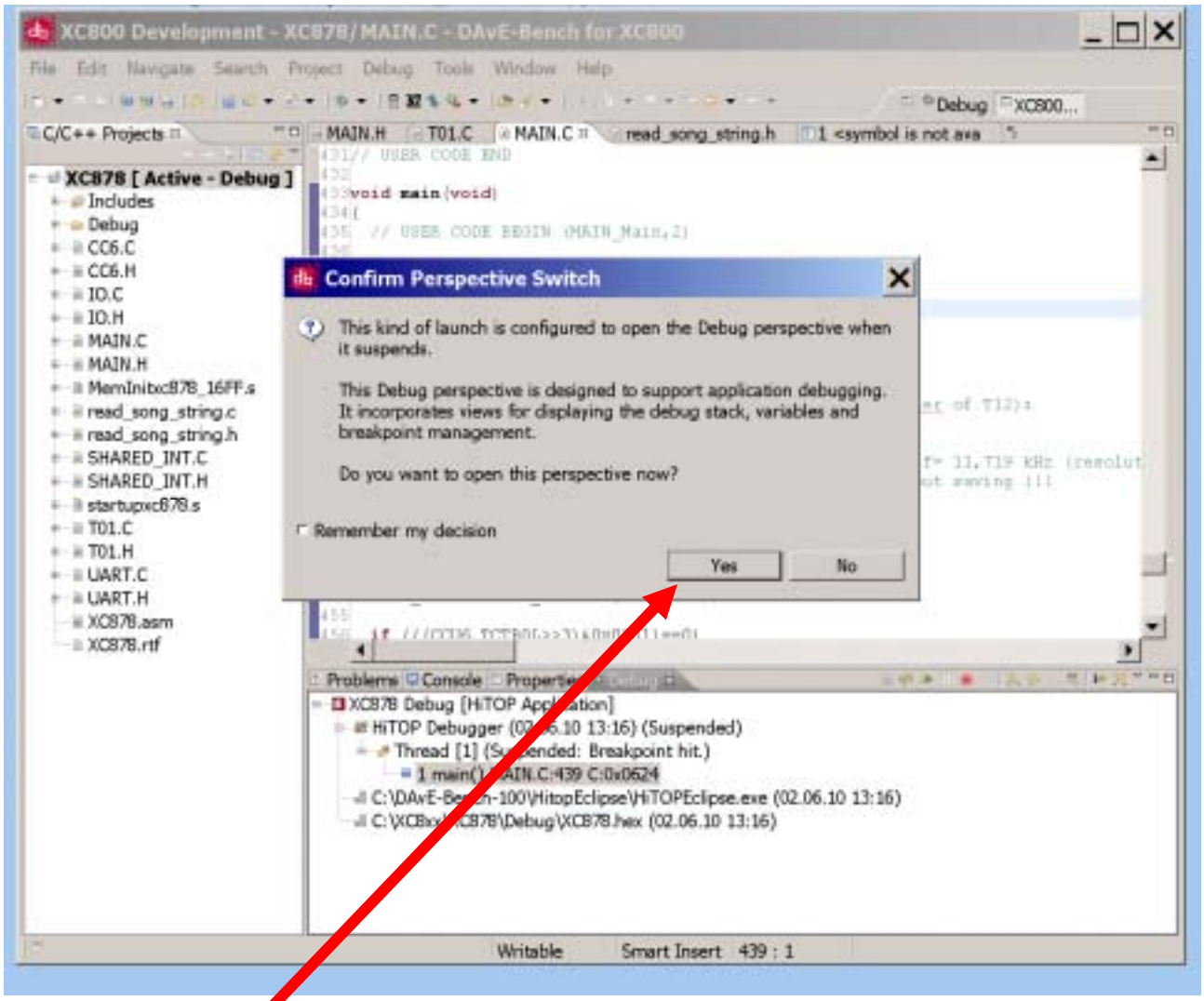


Go back to DAvE Bench and

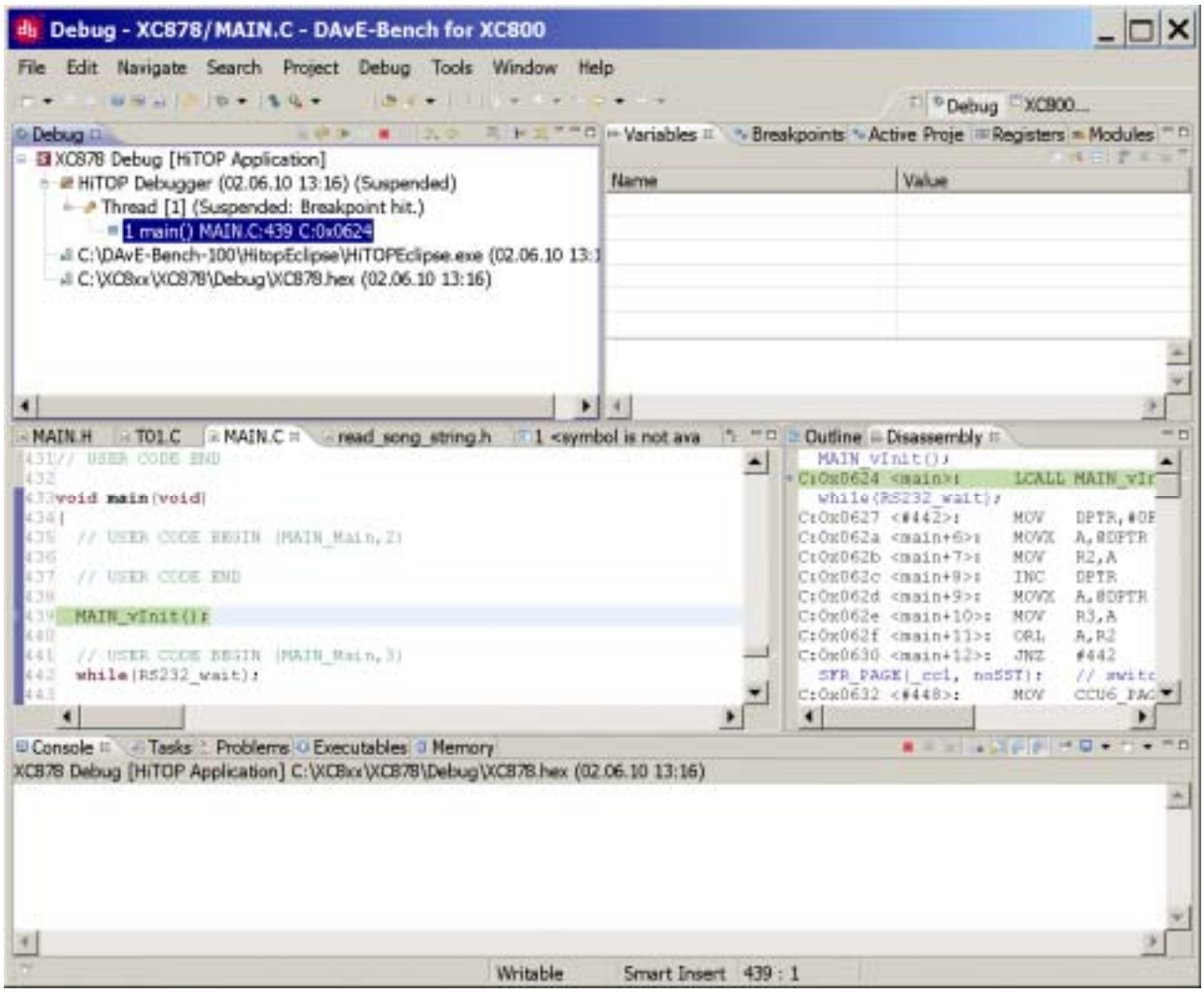
Start/Launch the debugger:

Click  :

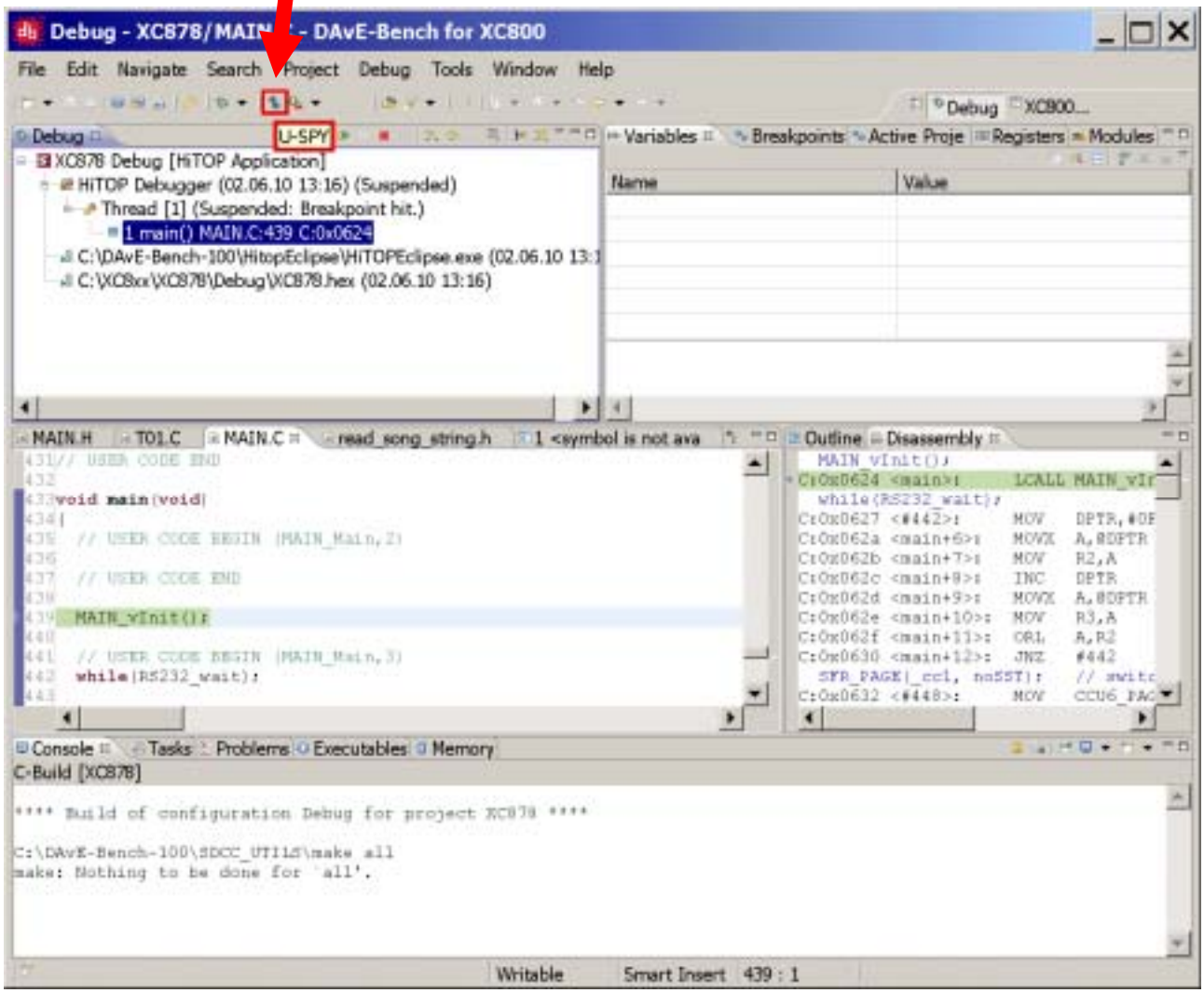


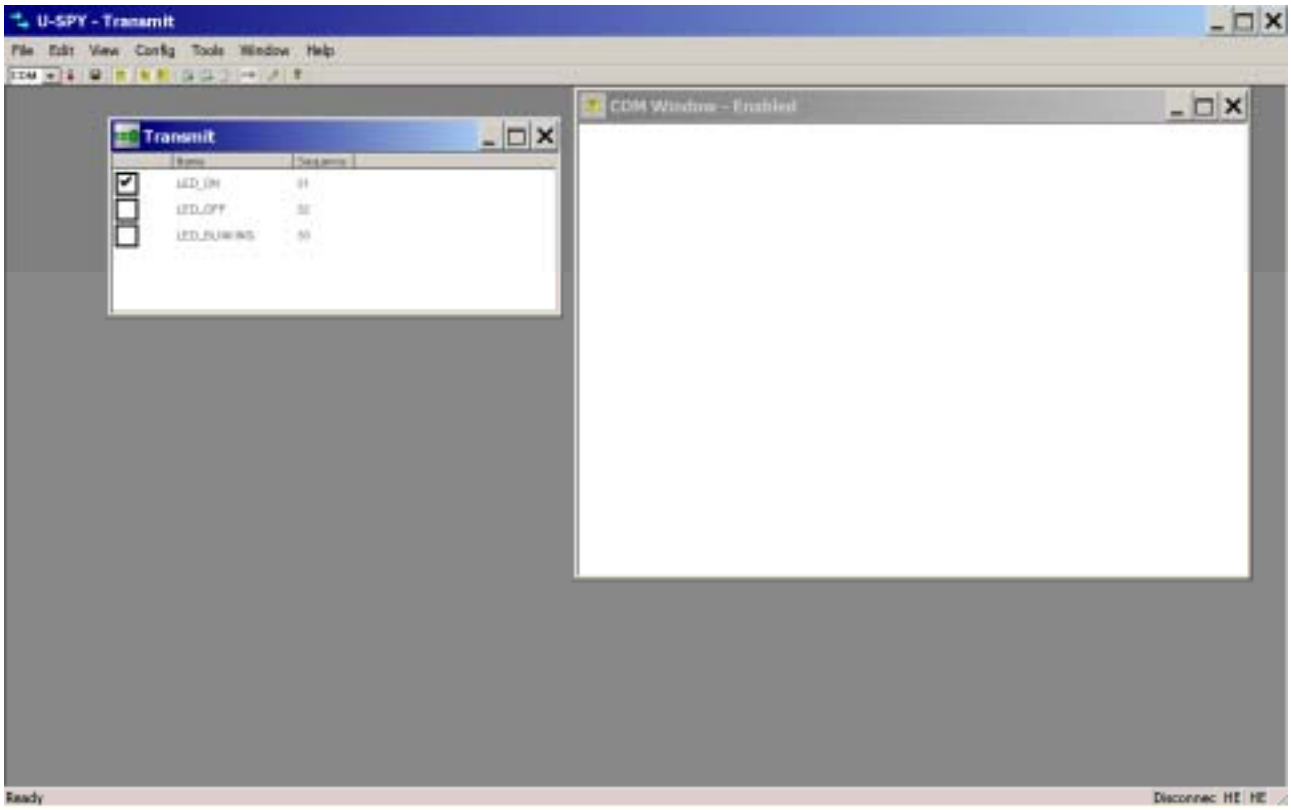

XC800 Development - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC878 [Active - Debug]
Includes
Debug
CC6.C
CC6.H
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc878_16FF.s
read_song_string.c
read_song_string.h
SHARED_INT.C
SHARED_INT.H
startupxc878.s
T01.C
T01.H
UART.C
UART.H
XC878.asm
XC878.rtf
MAIN.H
T01.C
MAIN.C
read_song_string.h
Confirm Perspective Switch
This kind of launch is configured to open the Debug perspective when it suspends.
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.
Do you want to open this perspective now?
Remember my decision
Yes
No
Problems
Console
XC878 Debug [HiTOP Application]
HiTOP Debugger
Thread [1]
Writable
Smart Insert
439 : 1

Click Yes

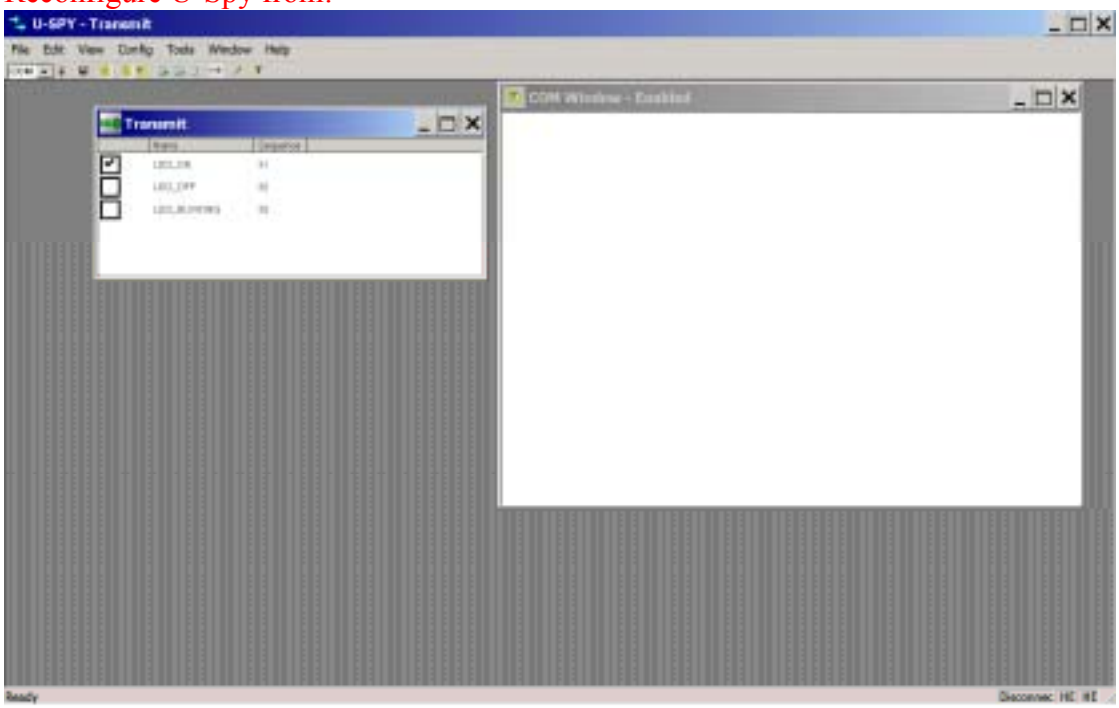

Debug - XC878/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
Debug
XC878 Debug [HiTOP Application]
HiTOP Debugger (02.06.10 13:16) (Suspended)
Thread [1] (Suspended: Breakpoint hit.)
1 main() MAIN.C:439 C:0x0624
C:\DAvE-Bench-100\HitopEclipse\HiTOPEclipse.exe (02.06.10 13:1
C:\XC8xx\XC878\Debug\XC878.hex (02.06.10 13:16)
Variables
Breakpoints
Active Proje
Registers
Modules
Name
Value
MAIN.H
T01.C
MAIN.C
read_song_string.h
1 <symbol is not ava
Outline
Disassembly
431// USER CODE END
432
433void main(void)
434{
435 // USER CODE BEGIN (MAIN_Main,2)
436
437 // USER CODE END
438
439 MAIN_vInit();
440
441 // USER CODE BEGIN (MAIN_Main,3)
442 while(RS232_wait);
443
MAIN_vInit();
C:0x0624 <main>: LCALL MAIN_vIr
while(RS232_wait);
C:0x0627 <#442>: MOV DPTR,#0F
C:0x062a <main+6>: MOVX A,@DPTR
C:0x062b <main+7>: MOV R2,A
C:0x062c <main+8>: INC DPTR
C:0x062d <main+9>: MOVX A,@DPTR
C:0x062e <main+10>: MOV R3,A
C:0x062f <main+11>: ORL A,R2
C:0x0630 <main+12>: JNZ #442
SFR_PAGE(_ccl, noSST); // switc
C:0x0632 <#448>: MOV CCU6_PAG
Console
Tasks
Problems
Executables
Memory
XC878 Debug [HiTOP Application] C:\XC8xx\XC878\Debug\XC878.hex (02.06.10 13:16)
Writable
Smart Insert
439 : 1

Now, start U-SPY: click 

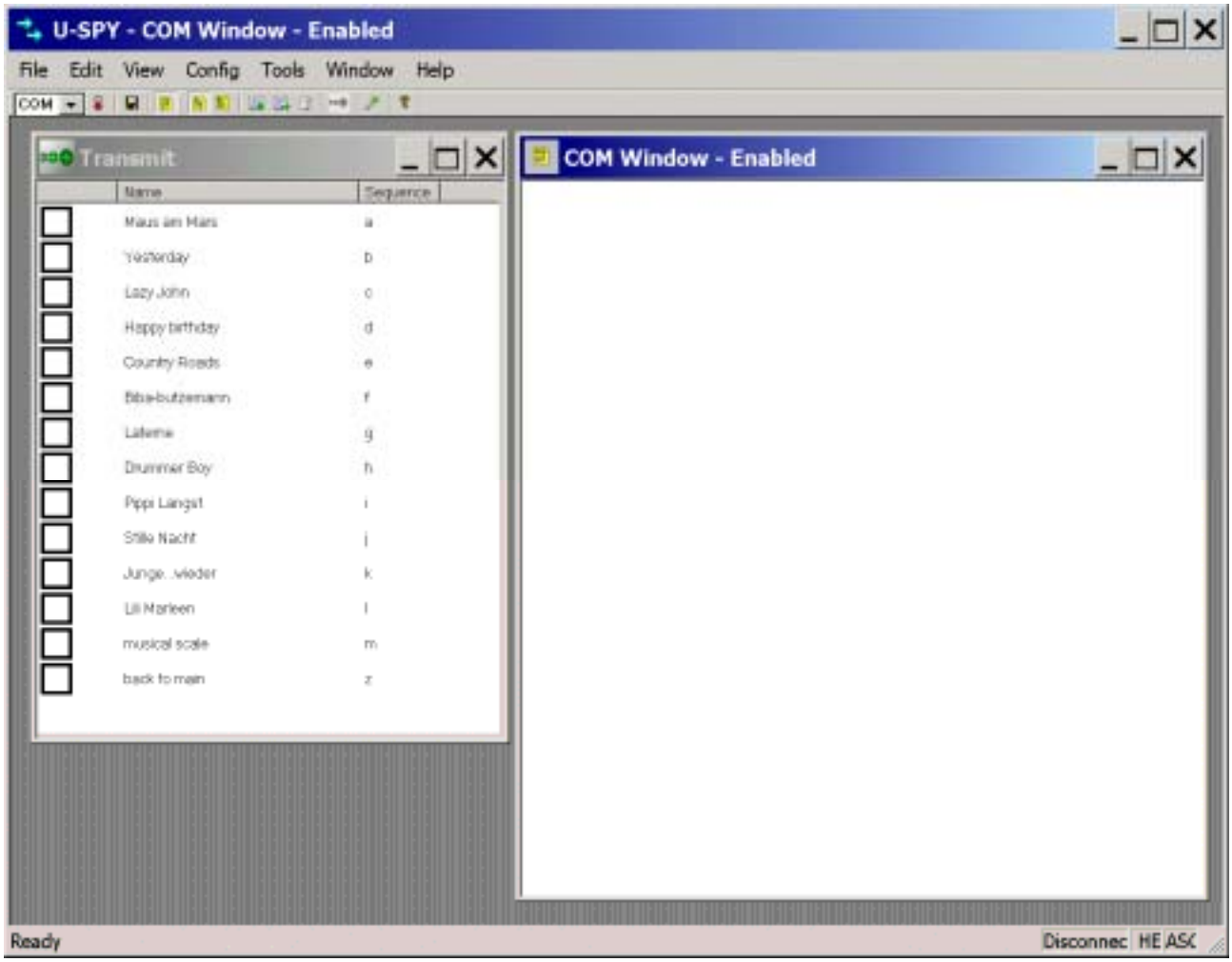


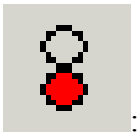

U-SPY - Transmit
File Edit View Config Tools Window Help
Transmit
COM Window - Enabled
Ready

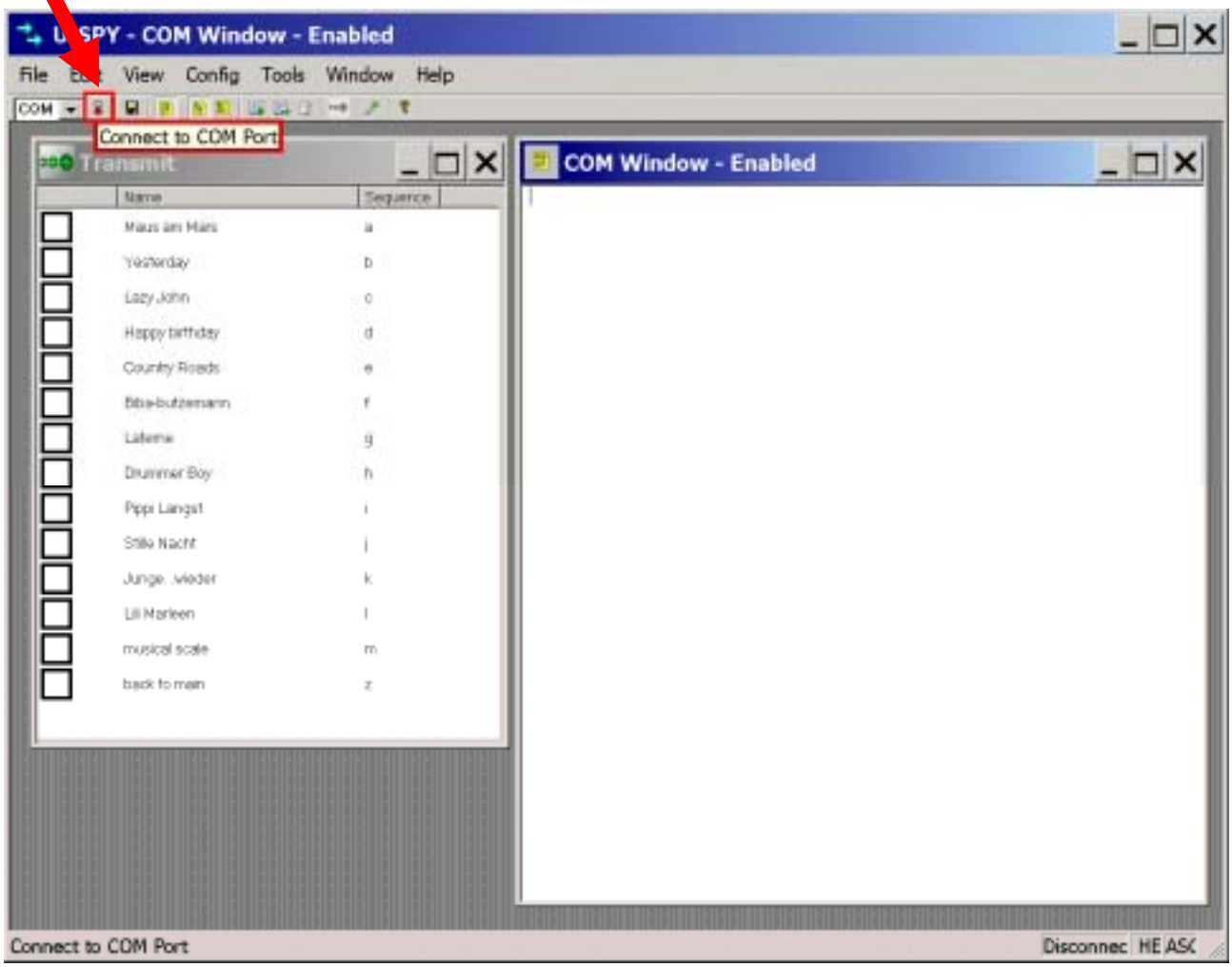
Reconfigure U-Spy from:



to:

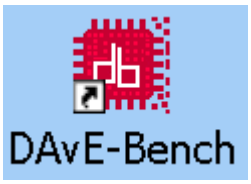


Click  :



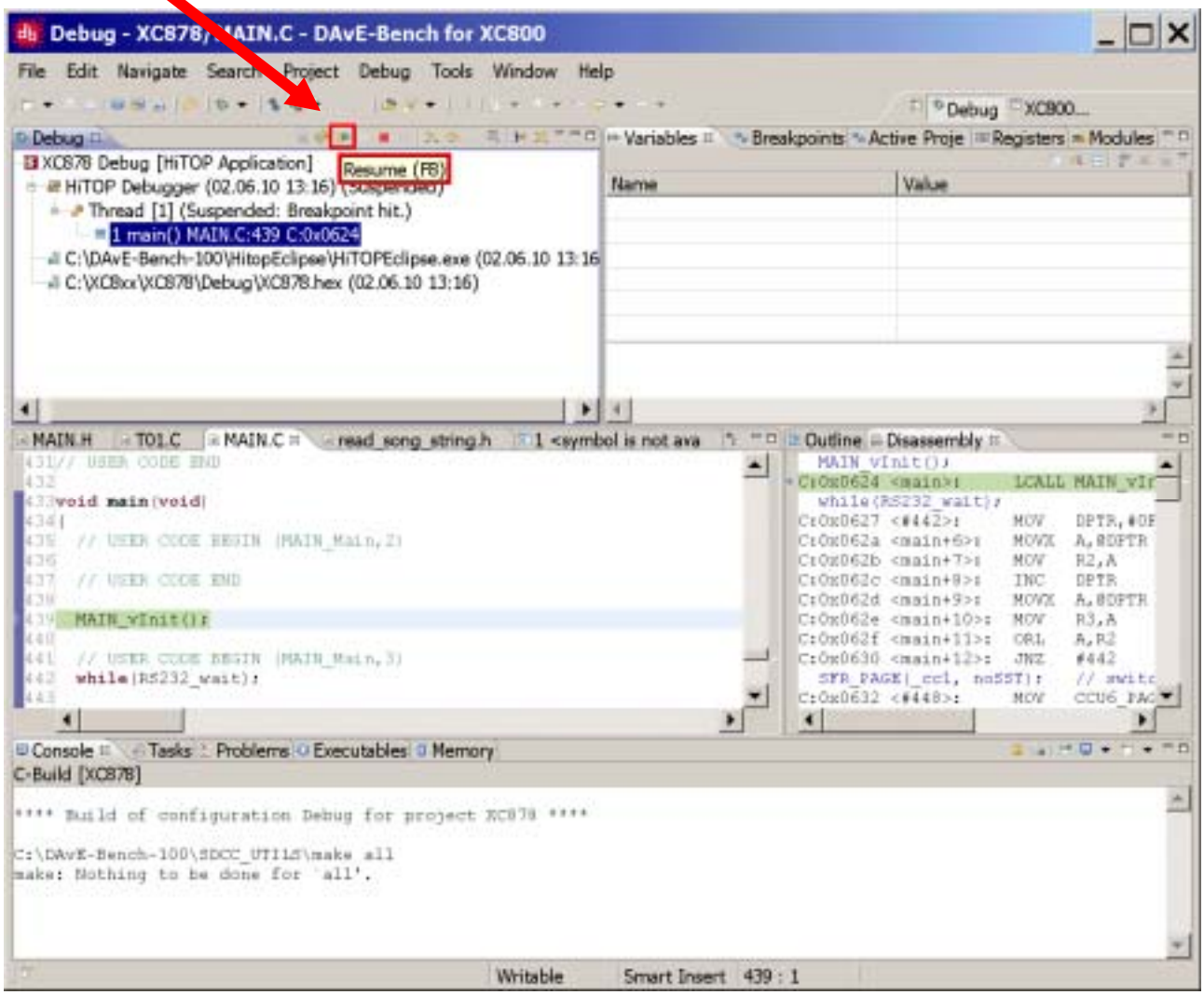
Note:

 : U-SPY is now ready for serial communication!

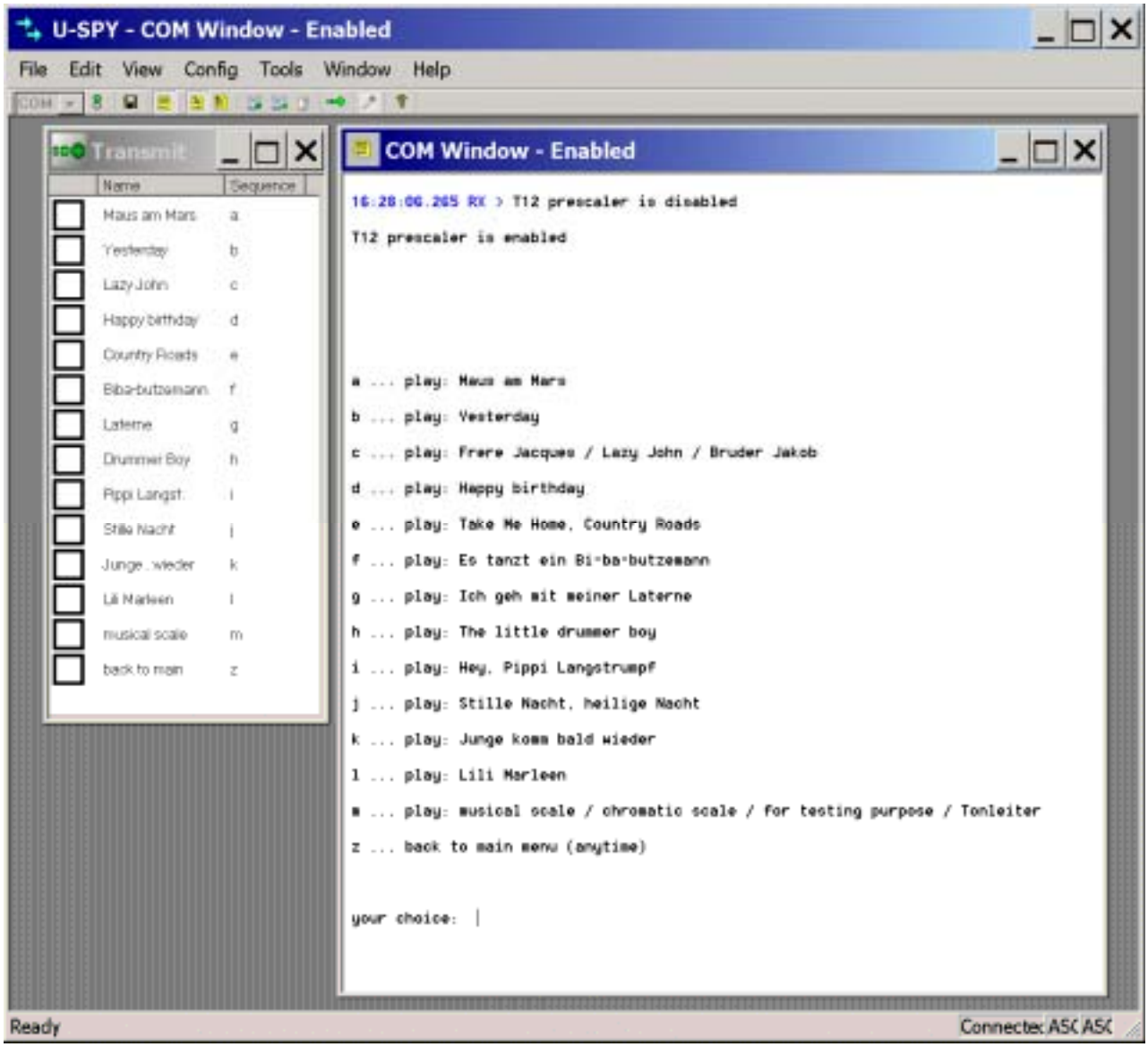


Go back to DAvE Bench and start the debugger

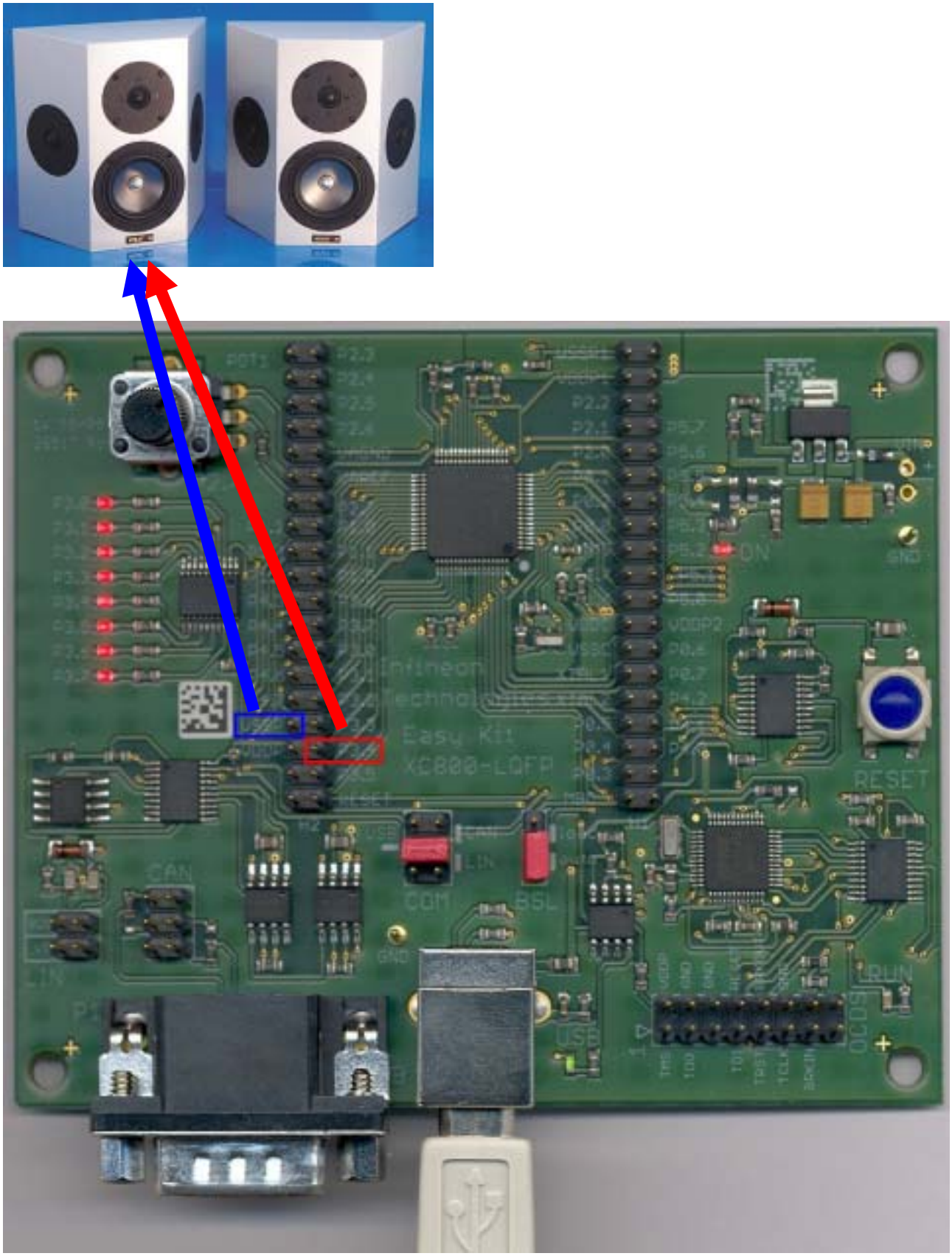
Click:  Resume



Go back to U-SPY and see the result:

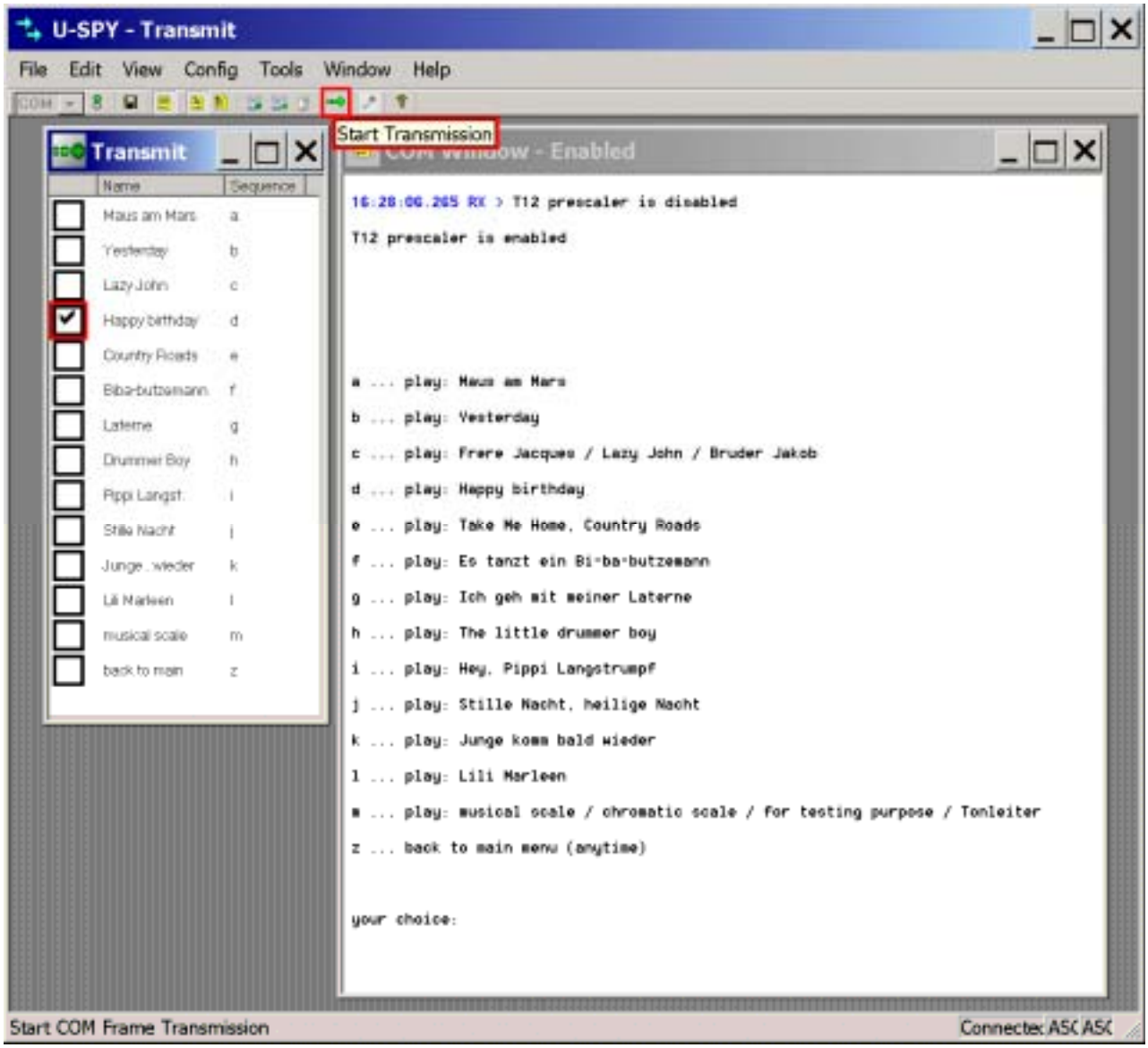


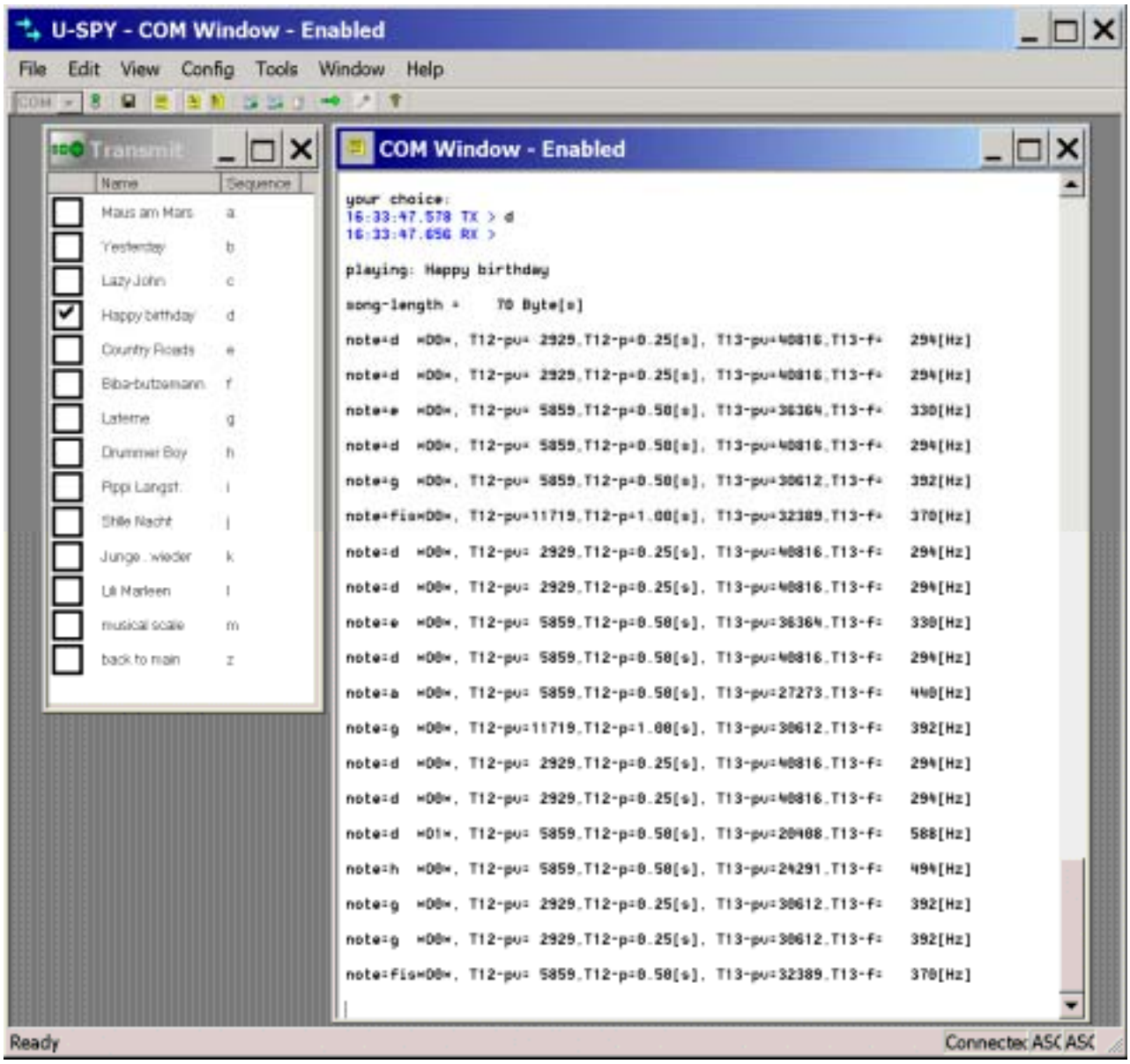
Connect CC62 (P3.4) and GND (VSSP) to your active loudspeaker(s) - and start/select a song:



See / hear / enjoy the results:

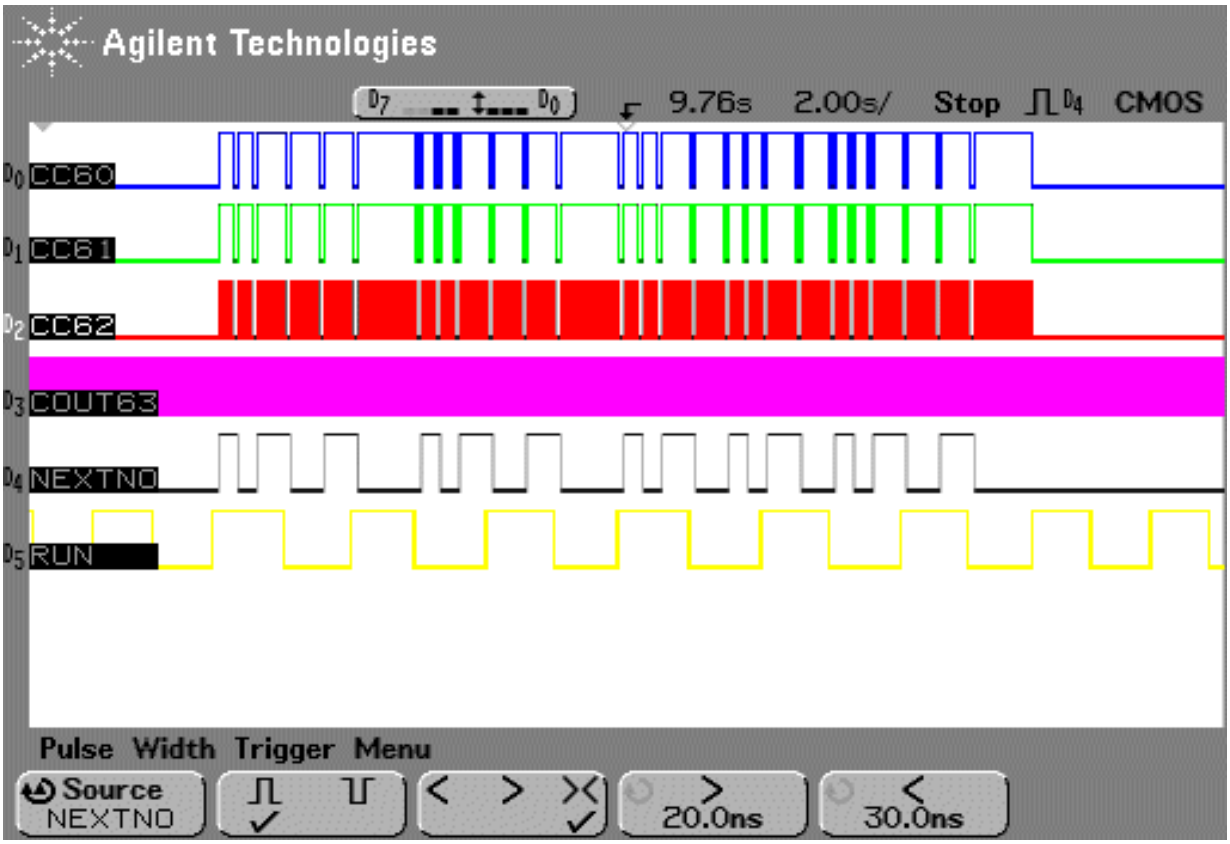
Select/insert/tick ☑ Happy birthday d and click  (Start Transmission)



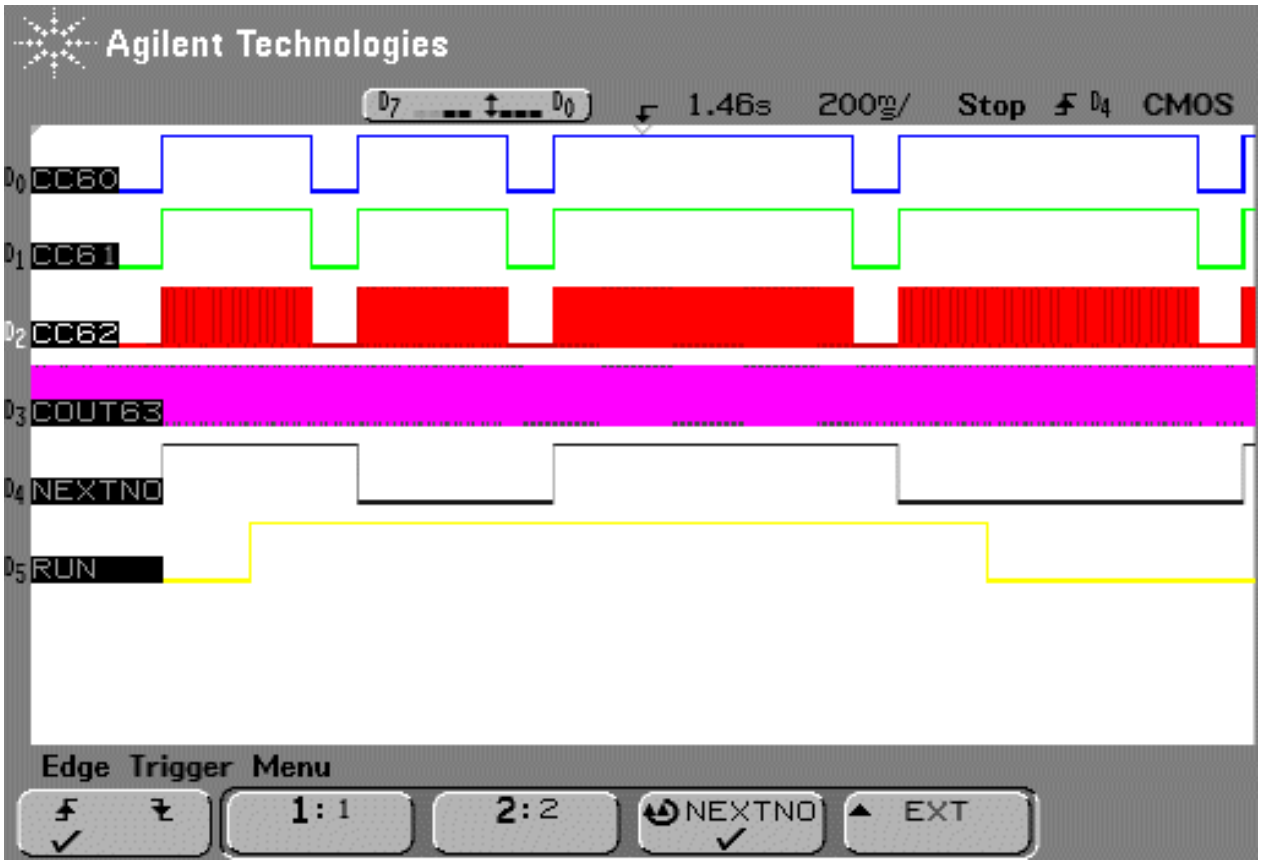

U-SPY - COM Window - Enabled
File Edit View Config Tools Window Help
Transmit
Name Sequence
Maus am Mars a
Yesterday b
Lazy John c
Happy birthday d
Country Roads e
Biba-butzemann f
Laterne g
Drummer Boy h
Pippi Langst i
Stille Nacht j
Junge...wieder k
Lili Marleen l
musical scale m
back to main z
COM Window - Enabled
your choice:
16:33:47.578 TX > d
16:33:47.656 RX >
playing: Happy birthday
song-length = 70 Byte[s]
note=d =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=40816,T13-f= 294[Hz]
note=d =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=40816,T13-f= 294[Hz]
note=e =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=36364,T13-f= 330[Hz]
note=d =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=40816,T13-f= 294[Hz]
note=g =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=30612,T13-f= 392[Hz]
note=fis=00=, T12-pu=11719,T12-p=1.00[s], T13-pu=32389,T13-f= 370[Hz]
note=d =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=40816,T13-f= 294[Hz]
note=d =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=40816,T13-f= 294[Hz]
note=e =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=36364,T13-f= 330[Hz]
note=d =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=40816,T13-f= 294[Hz]
note=a =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=27273,T13-f= 440[Hz]
note=g =00=, T12-pu=11719,T12-p=1.00[s], T13-pu=30612,T13-f= 392[Hz]
note=d =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=40816,T13-f= 294[Hz]
note=d =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=40816,T13-f= 294[Hz]
note=d =01=, T12-pu= 5859,T12-p=0.50[s], T13-pu=20408,T13-f= 588[Hz]
note=h =00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=24291,T13-f= 494[Hz]
note=g =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=30612,T13-f= 392[Hz]
note=g =00=, T12-pu= 2929,T12-p=0.25[s], T13-pu=30612,T13-f= 392[Hz]
note=fis=00=, T12-pu= 5859,T12-p=0.50[s], T13-pu=32389,T13-f= 370[Hz]
Ready
Connected: ASC ASC

Use any logic analyser and see the result:

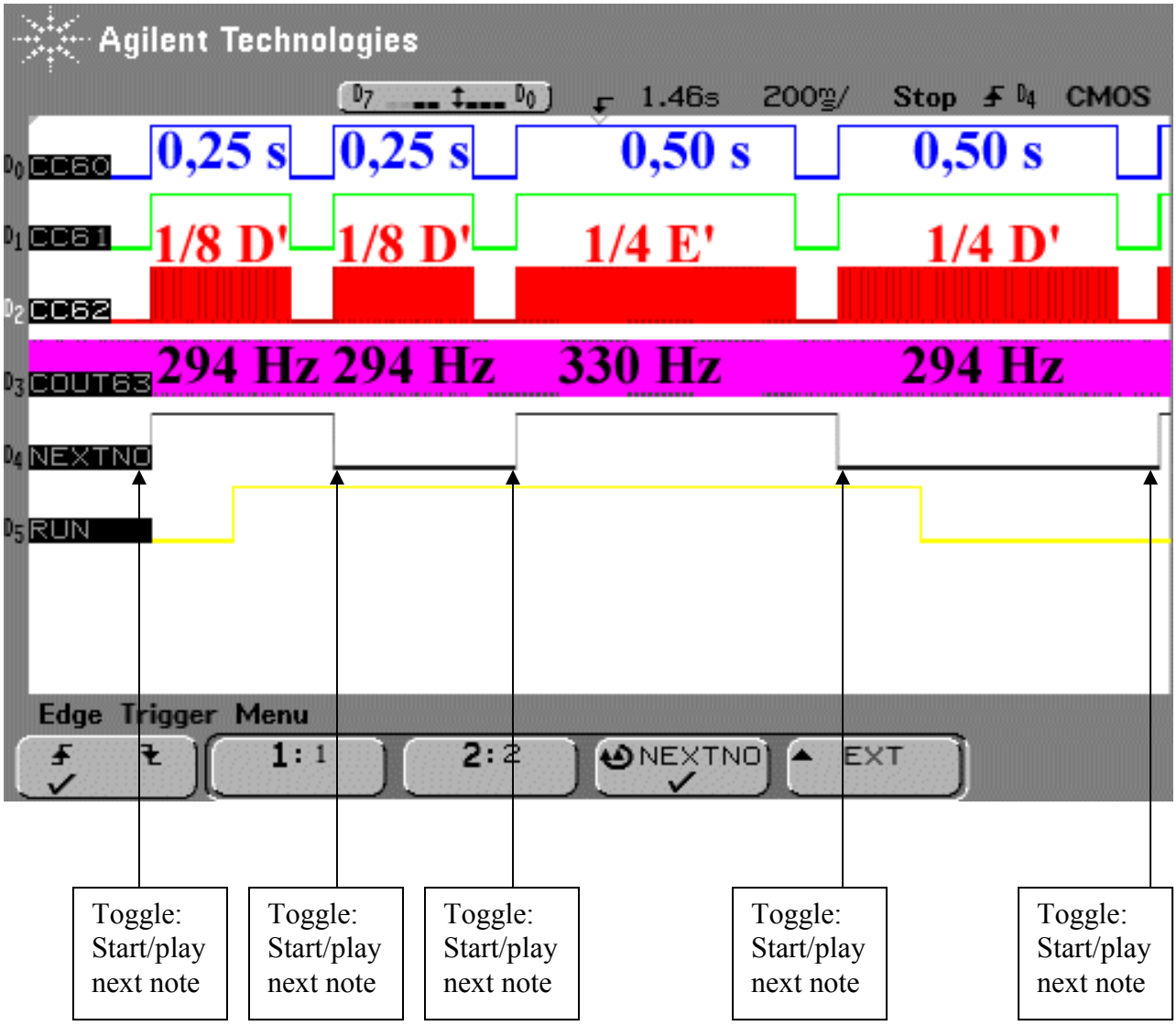
Note:
Song d: Happy birthday:
code unsigned char
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";



HAPPY BIRTHDAY =
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F…

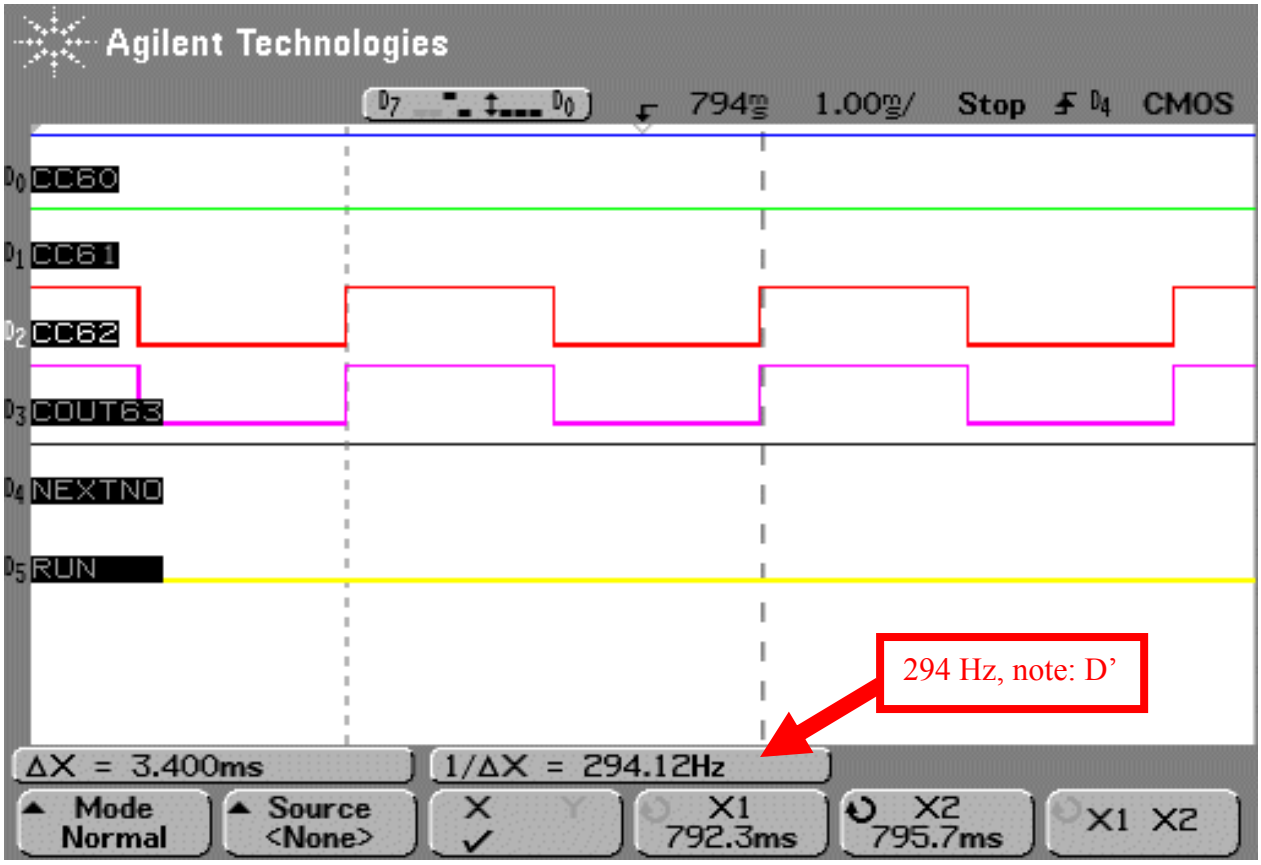


HAPPY BIRTHDAY =
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F…



Note:
Toggle: Start/play next note:
IO_vTogglePin(P3_1); // Show start of next note on Port 3 Pin 1
CC6_vStartTmr(CC6_TIMER_12); // Set Timer 12 Run Set bit T12RS

Now we can measure the frequency of note d (we expect 294 Hz) with the LGA:

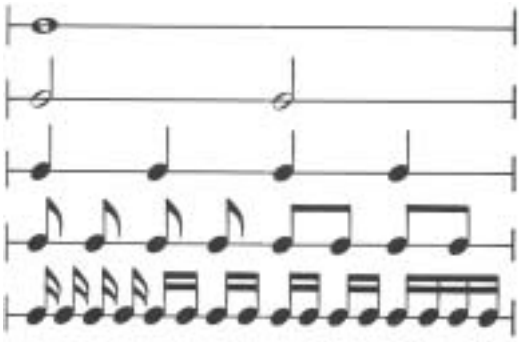


3.) Appendix: about music (note length and note frequency)

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:

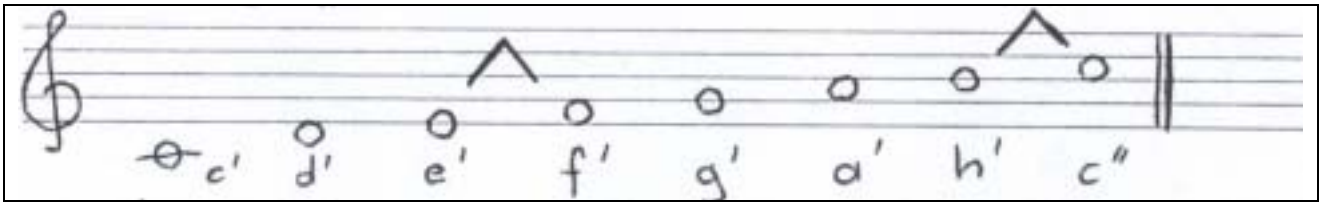
note	LENGTH
	½ (2 beats) + (½)/2 (1 beat) = ¾ (3 beats)

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:

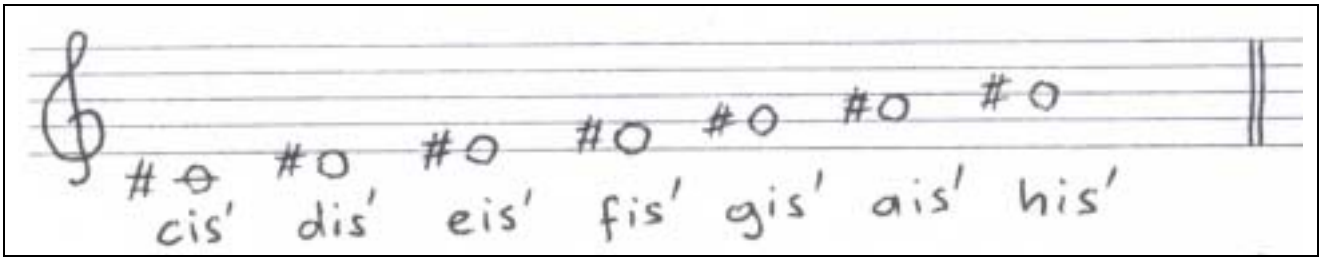


Note:
The notes `C, D, E, F, G, A, H` are named `C, D, E, F, G, A, B` in other countries.
In this document we stick to the German names.

Syntax used in our programming example:

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

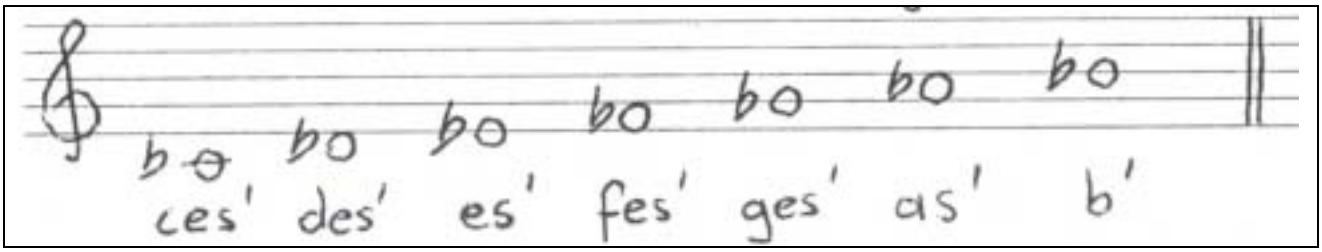
Real Music:



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:

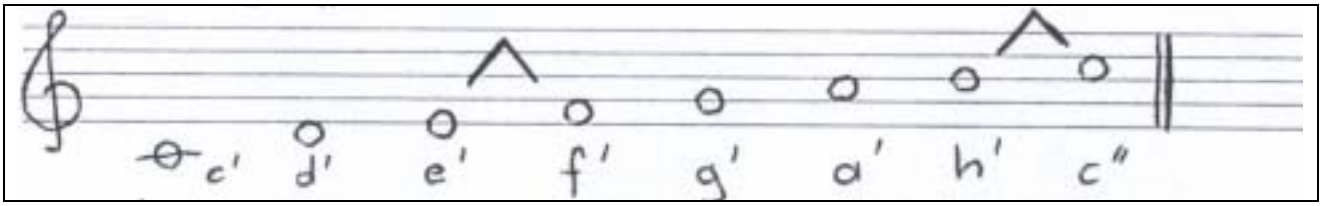
Definition:
In music, an octave is the interval between one musical note and another with half or double its frequency.

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 …).

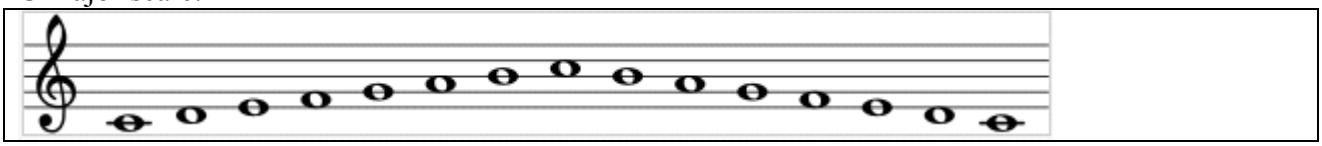
Syntax used in our programming example:

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

Real Music:

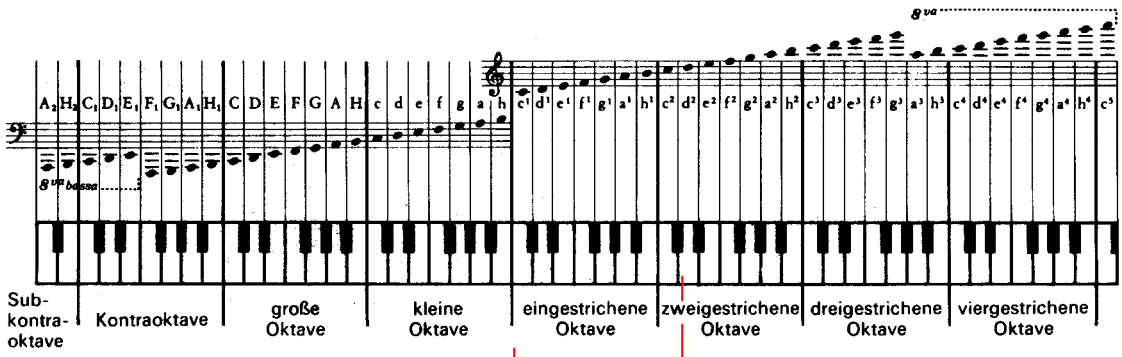


C major scale:

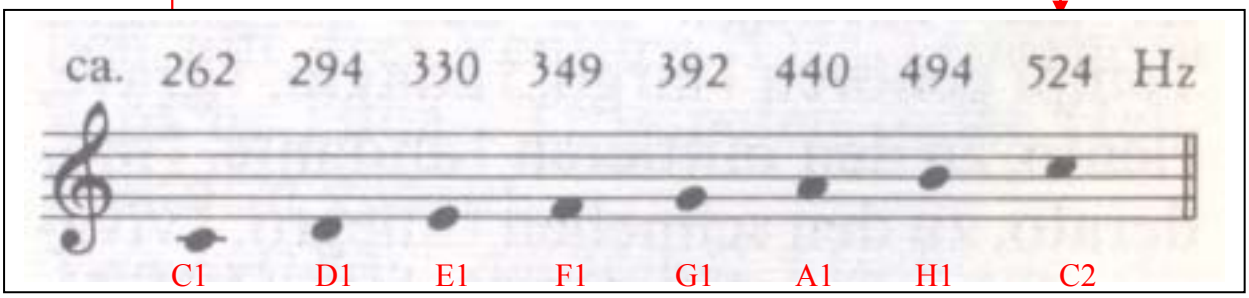


4.) Appendix: CAPCOM6 / CCU6 use to create note length and note frequency

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



*1

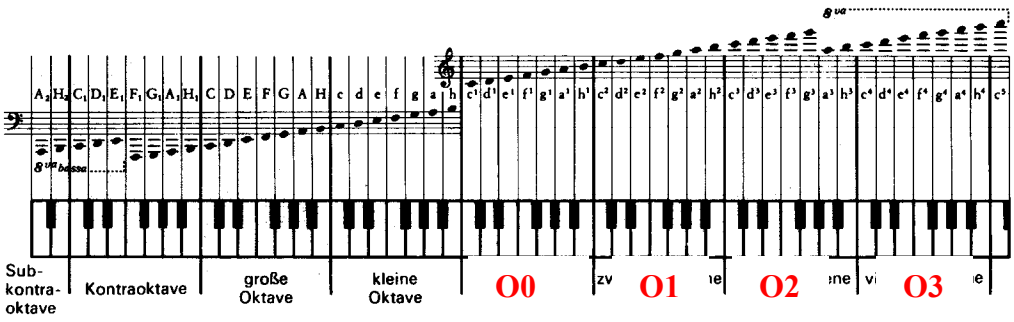


*2

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[] =
{45802,43309,40816,38590,36364,34383,32389,30612,28943,27273,25782,24291,200};
/*
[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>
*/
```

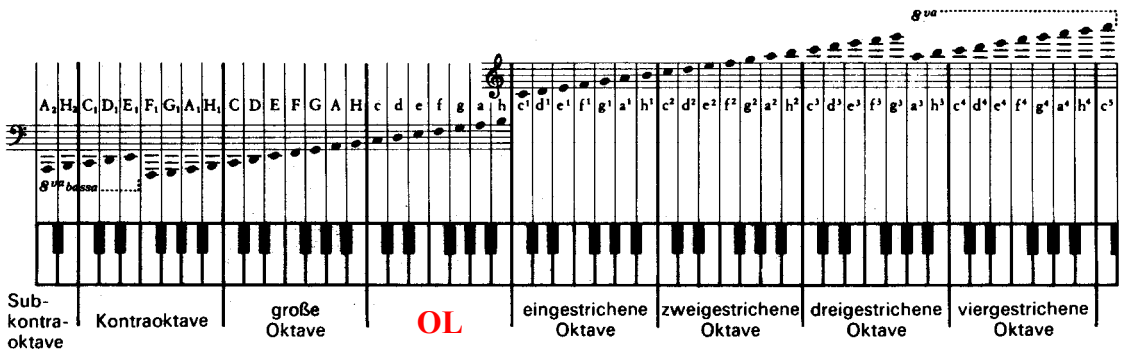
So we get the following values shown in the table below
[Note: Timer 13 resolution = $1/(f_{clk}/2) = 1/(24MHz/2) = 83{,}333$ ns]:

		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] = 45802	c'	262	523		
T13_values[1] = 43309	cis'				
T13_values[2] = 40816	d'	294			
T13_values[3] = 38590	dis'				
T13_values[4] = 36364	e'	330			
T13_values[5] = 34383	f'	349			
T13_values[6] = 32389	fis'				
T13_values[7] = 30612	g'	392			
T13_values[8] = 28943	gis'				
T13_values[9] = 27273	a'	440	880	1760	3520
T13_values[10] = 25782	ais'				
T13_values[11] = 24291	h'	494	988		
T13_values[12] = 200	----	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 / (27273 * 83,333 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[] =
{45802,43309,40816,38590,36364,34383,32389,30612,28943,27273,25782,24291,200};
/*
[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>
*/
```

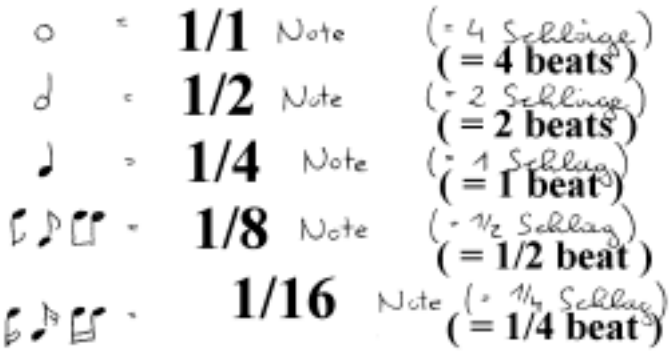
So we get the following values shown in the table below
[Note: Timer 13 resolution = $1/(f_{clk}/4) = 1/(24MHz/4) = 166,6667$ ns]:

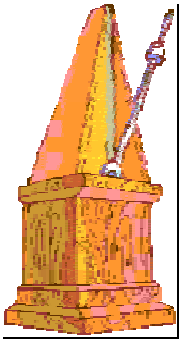
		Octave=OL T13 Prescaler=4	Octave=ONO0 T13 Prescaler=2
T13 period values	note	f [Hz]	f [Hz]
T13_values[0] = 45802	c	131	262
T13_values[1] = 43309	cis	139	
T13_values[2] = 40816	d	147	294
T13_values[3] = 38590	dis	156	
T13_values[4] = 36364	e	165	330
T13_values[5] = 34383	f	175	349
T13_values[6] = 32389	fis	186	
T13_values[7] = 30612	g	196	392
T13_values[8] = 28943	gis	208	
T13_values[9] = 27273	a	220	440
T13_values[10] = 25782	ais	234	
T13_values[11] = 24291	h	247	494
T13_values[12] = 200	----	30000	60000

Therefore we use the following program sequence in our application:

```
// note-frequency:
CC6_vSetTmrPeriod(CC6_TIMER_13,(T13_values[note]/octave));

// duty-cycle = 50 %:
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_3,(T13_values[note]/octave/2));

CC6_vEnableShadowTransfer(CC6_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 72 to 199 Beats per minute:

```
Tx : Change tempo (x = 72 ... 199 Beats per Minute)
```

And tempo is used in the following way:

```
// note-length:
CC6_vSetTmrPeriod(CC6_TIMER_12,(current_note_length/tempo*120));

// 100% duty-cycle:
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_0,0);
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_1,0);
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2,0);
```

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 = 85,333 µs):

			note length
T12 period values	note	note	[s]
23438 / 1 = 23438		1/1	2
23438 / 2 = 11719		1/2	1
23438 / 4 = 5859		1/4	0,5
23438 / 8 = 2930		1/8	0,25
23438 / 16 = 1465		1/16	0,125

e.g. for  :
note length = T12-period-value / 4 * T12-resolution
note length = **23438** / 4 * 85,333 µs = 0,5 [s]

In our programming example we use the following code sequences:

```
// Standard - length of a whole note with tempo 120:
unsigned int length_of_a_whole_note = 23438;
```

```
// 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;
```

```
// T12, note-length:
// period value note-length
CC6_vSetTmrPeriod(CC6_TIMER_12,(current_note_length/tempo*120));

// Channel_2:
// if compare value CCU6_CC62SR == 0 -> 100 % duty cycle
// for note length:
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2,0);

CC6_vEnableShadowTransfer(CC6_TIMER_12);
```

Implementing note length and note frequency on real hardware:

Using XC878 using CAPCOM6 / CCU6: T12 and T13:

T12: note length:

24 MHz -> 1/256 (T12PRE=1) -> 1/8 (done by DAvE) ->
f= 11,719 kHz (resolution = 85,333 µs)
Duty cycle = 100 %

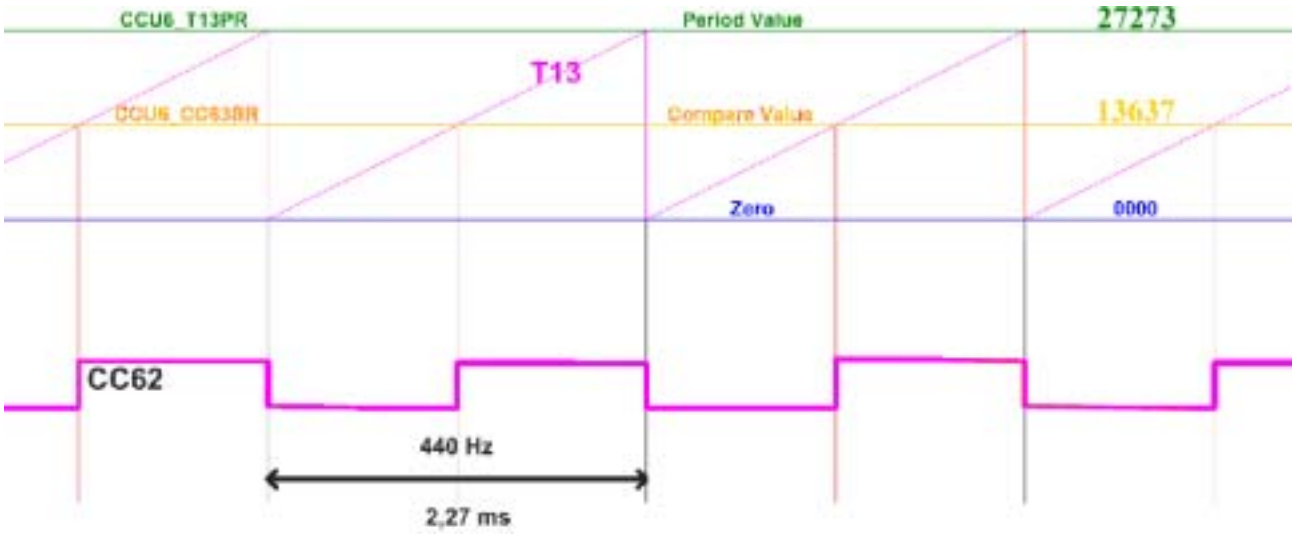
T13: note frequency (Octave = O0, O1, O2 and O3):

24 Mhz -> 1/2 (done by DAvE) ->
12 MHz (resolution = 83,333 ns)
Duty cycle = 50 %

e.g. note = a' (440 Hz):

CCU6_T13PR=T13_values[note]/octave;
CCU6_T13PR=T13_values[9]/1
CCU6_T13PR=27273/1
CCU6_T13PR=27273

CCU6_CC63SR=CCU6_T13PR/2 ;
CCU6_CC63SR=27273/2
CCU6_CC63SR=13637



Note:
Modulation of CAPCOM6 T12 CC62 output done by T13 (done by hardware functionality)

Note:

T13: note frequency (Octave = OL):

24 Mhz -> 1/4 (done by DAvE) ->
12 MHz (resolution = 166,6667 ns)
Duty cycle = 50 %

5.) Appendix: songs used

5.1.) Song a: Maus am Mars:

```
// Maus am Mars (song a):
code unsigned char
songa[]="T120O0L4FL8AL4O1C.O0L8FEGL2O1CO0P4P8L4EL8GO1L4C.O0L8EFAL2O1CP4
P8O0L4FL8AO1L4C.O0L8FH-O1L4DFL8FEDDCO0HO1CDCO0H-GL2F.";
```

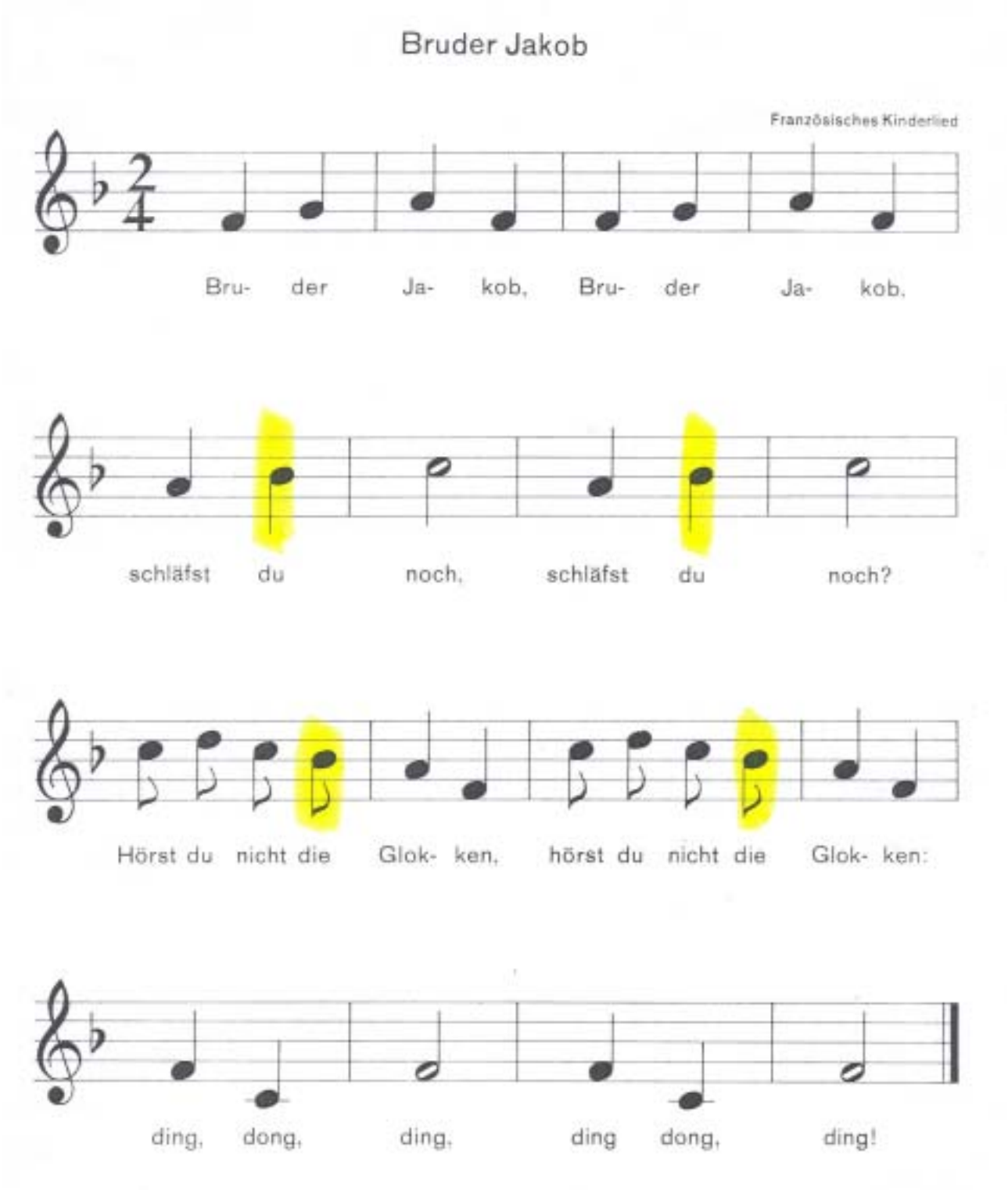
Note:
Thanks to Christian Perschl (www.perschl.at).
The songstring above was written down by Christian while humming the melody.

5.2.) 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.

5.3.) Song c: Bruder Jakob:

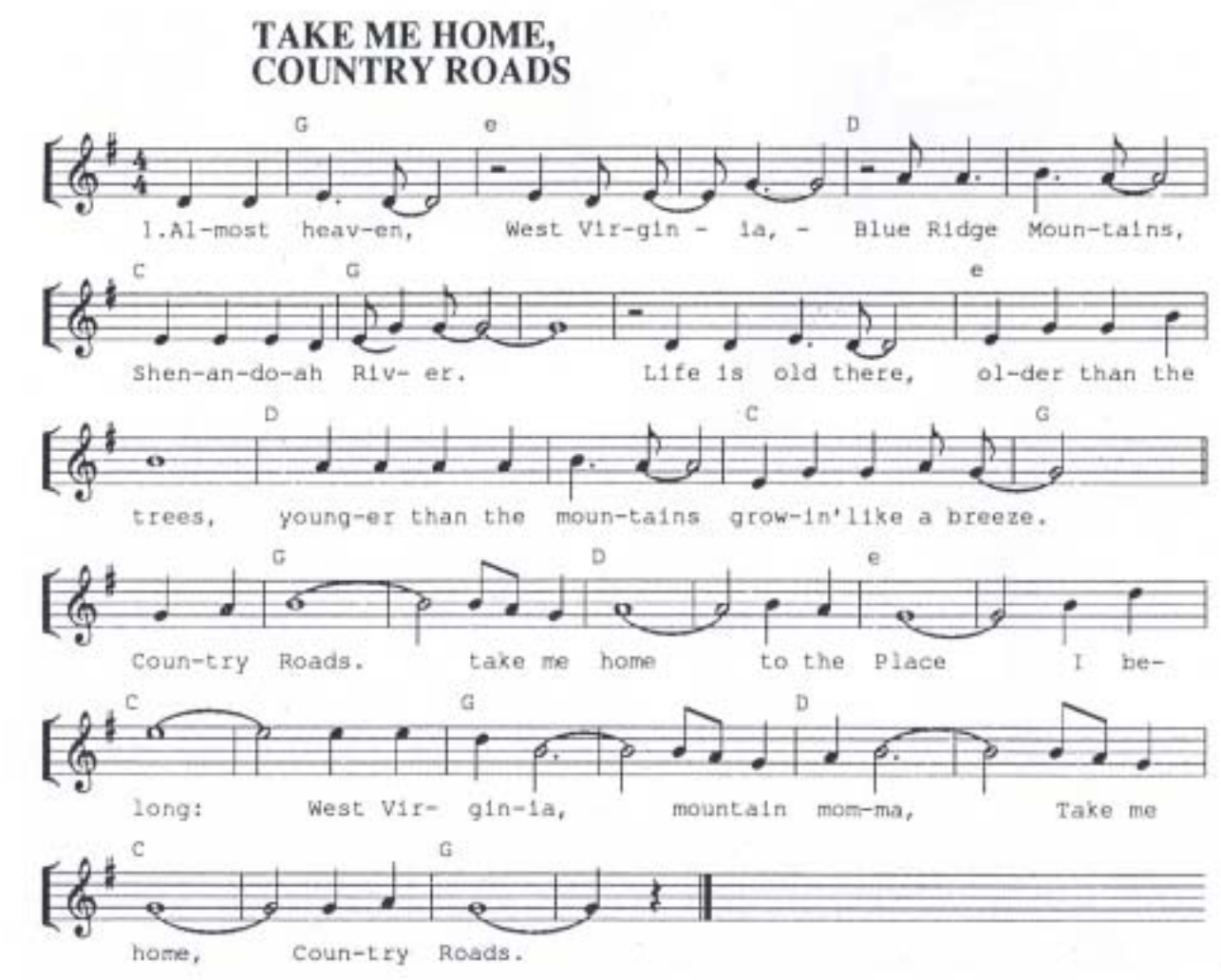


// Bruder Jakob (song c):
code unsigned char songc[]="T120O0L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0L8H-L4AFO1L8CDCO0L8H-L4AFFCL2FL4FCL2F";

5.4.) Song d: Happy birthday:

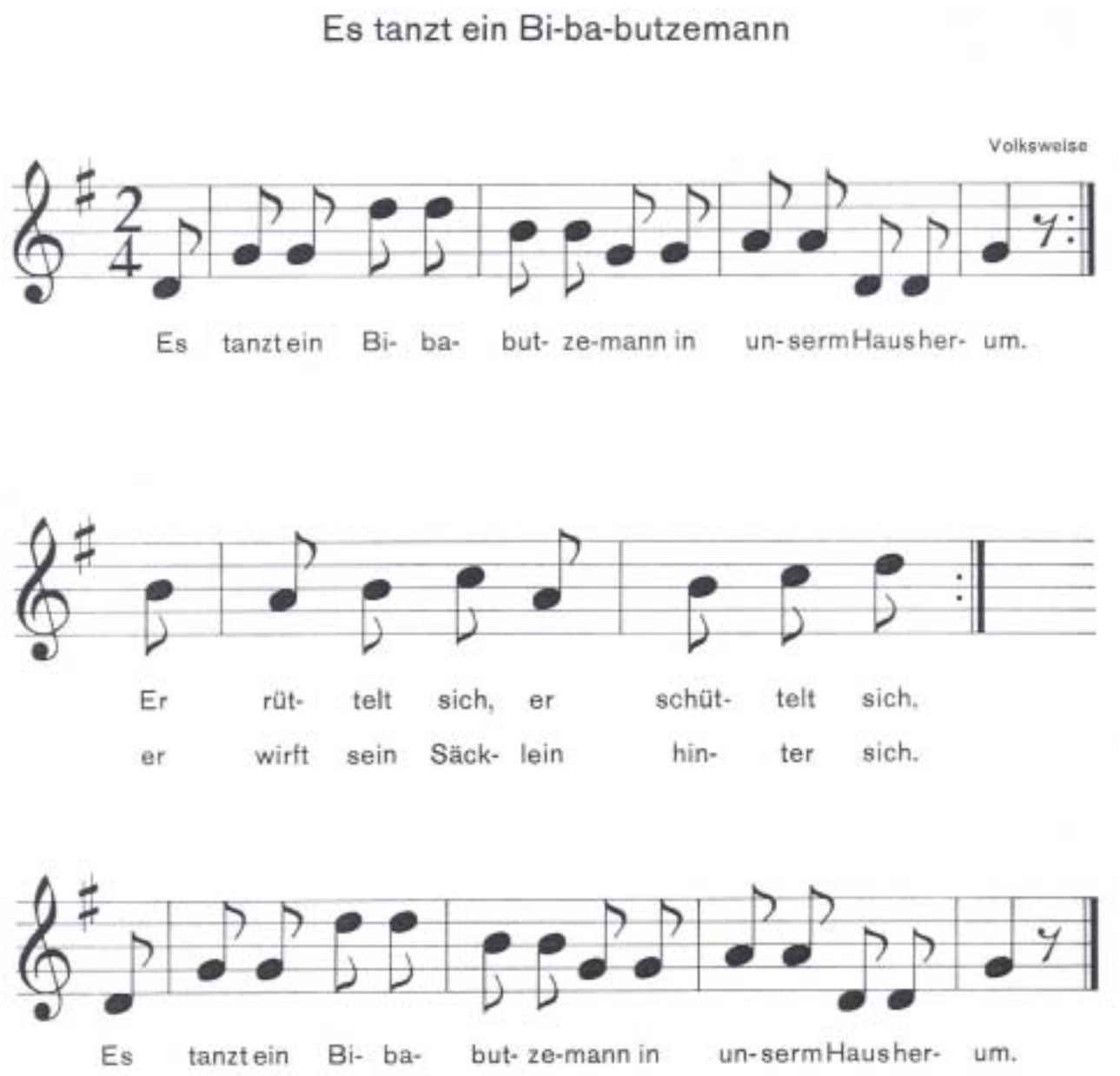

```
// Happy birthday (song d):
code unsigned char
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";
```

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



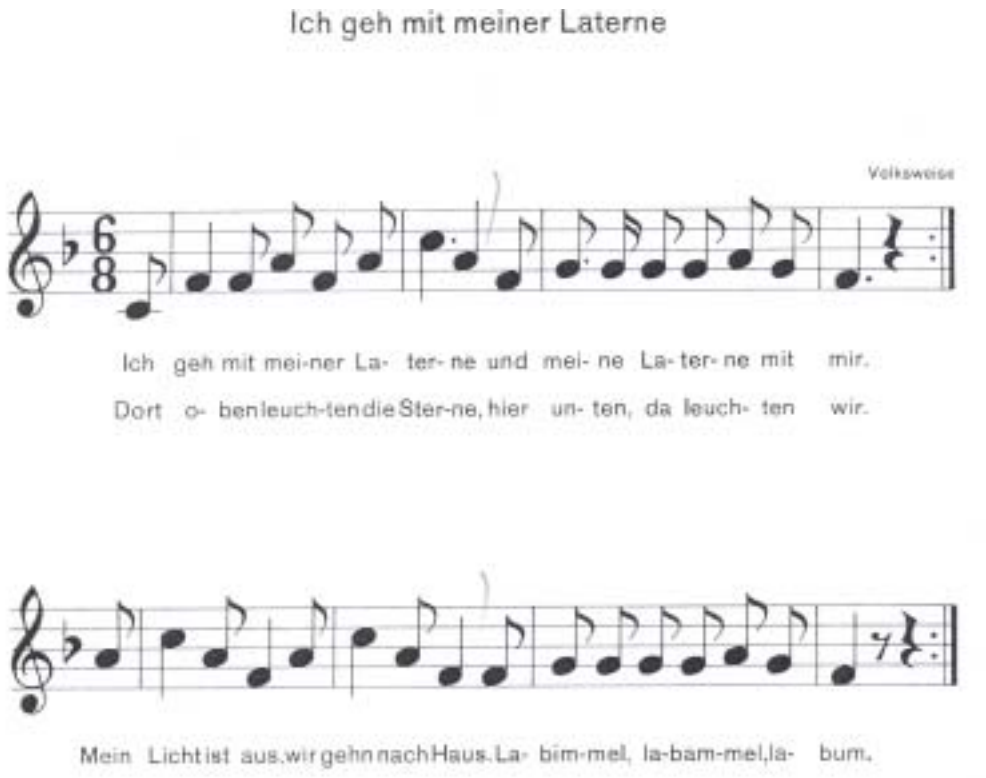

```
// Take Me Home, Country Roads (song e):
code unsigned char
songe[]="T199O0L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EEEDL8EL4GL1GP
1L4DDE.L2D.L4EGGHL1HL4AAAAH.L2A.L4EGGAL2G.L4GAL1HL8HAL4GL1AL4HAL1G
L4HO1L4DL1EL4EEDO0L1HL8HAGAL1HL8HAL4GL1GL4GAL1G";
```

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



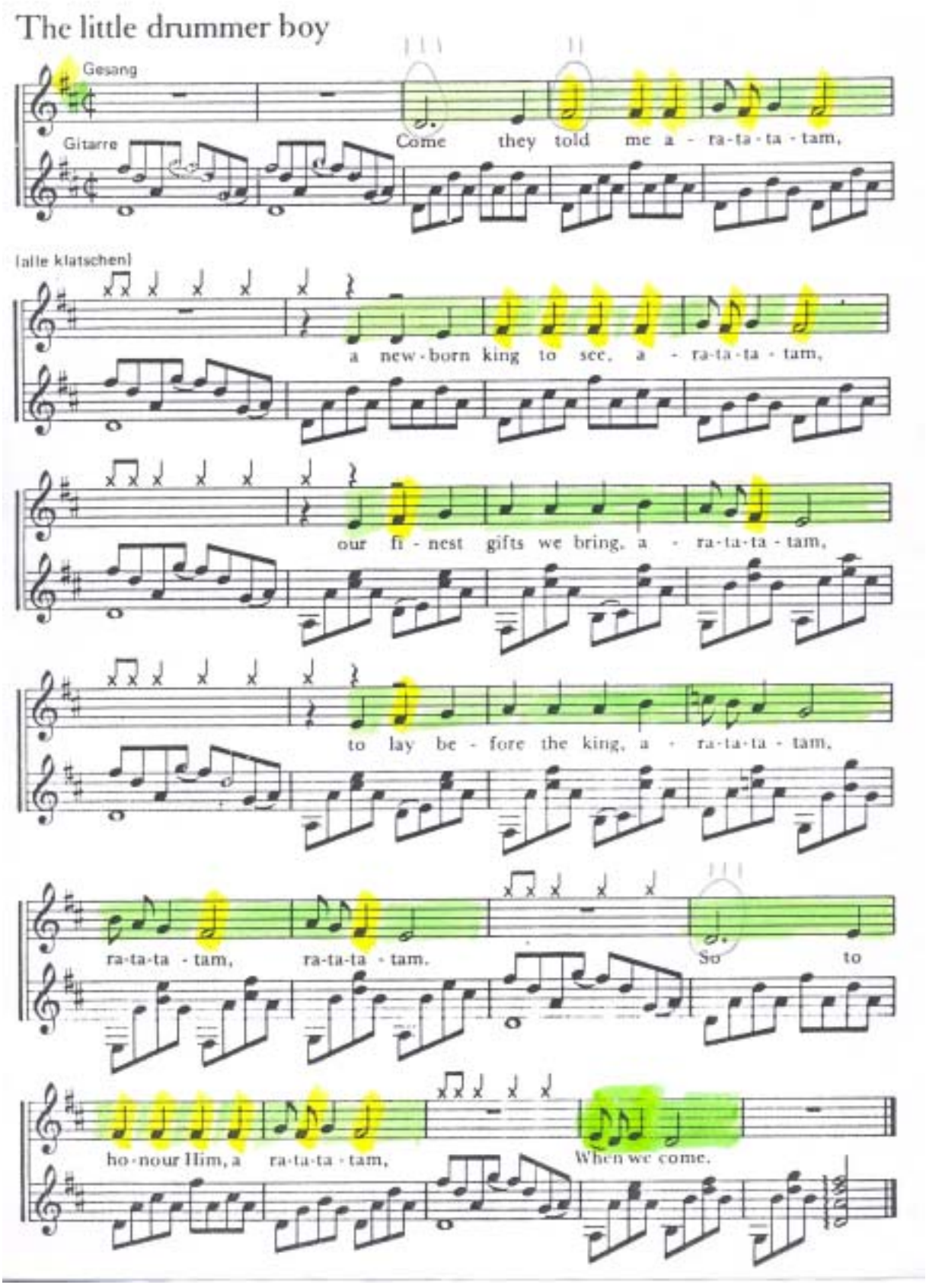

```
// Es tanzt ein Bi-ba-butzemann (song f):
code unsigned char
songf[]="T199O0L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";
```

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



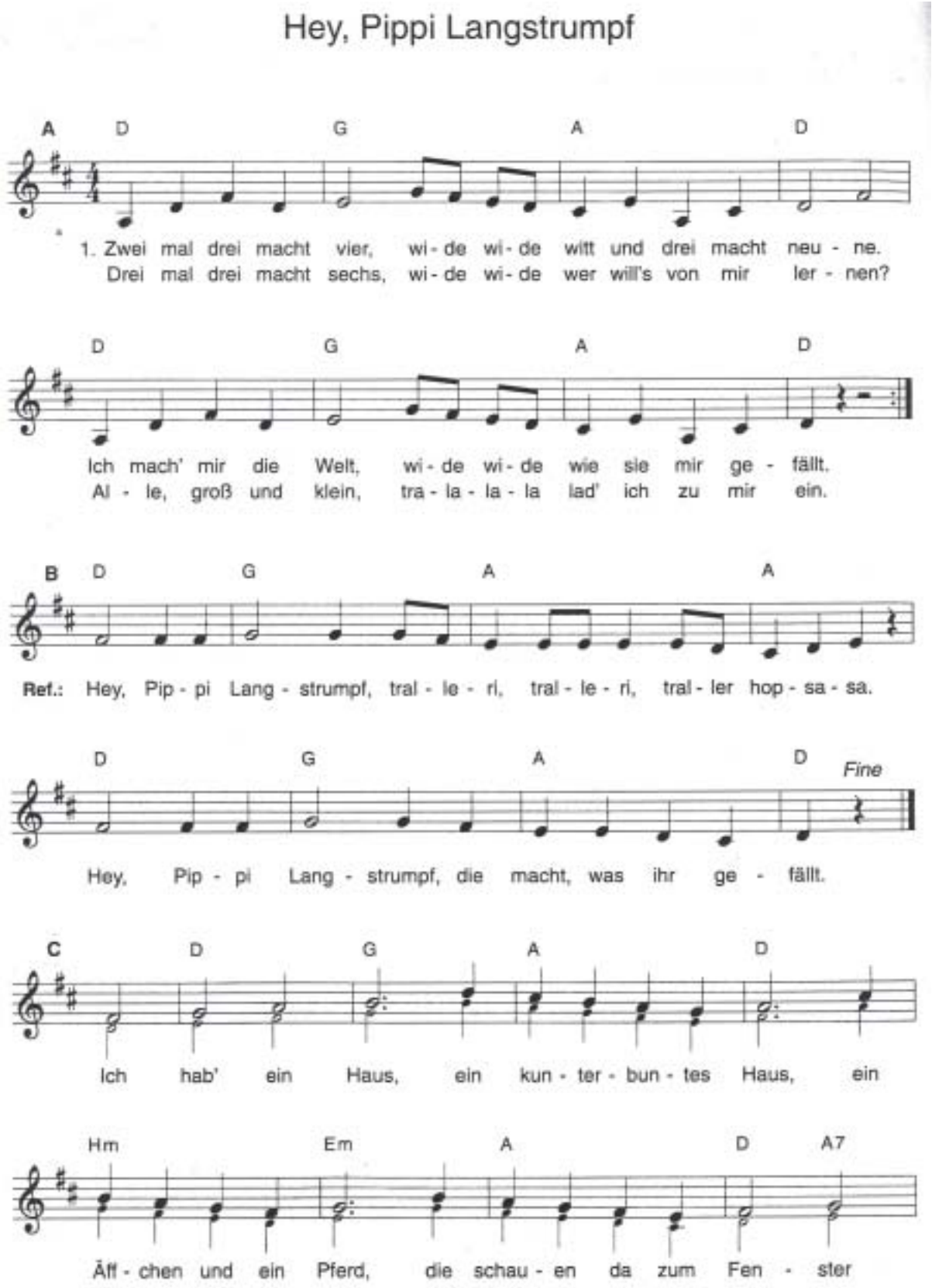

```
// Ich geh mit meiner Laterne (song g):
code unsigned char
songg[]="T120O0L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8CL4FL8FA
FAO1L4C.O0L4AL8FG.L16GL8GGAGL4F.P4O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4F
L8FGGGGAGL4FP4.O0L8AO1L4CO0L8AL4FL8AO1L4CO0L8AL4FL8FGGGGAGL4FP4.";
```

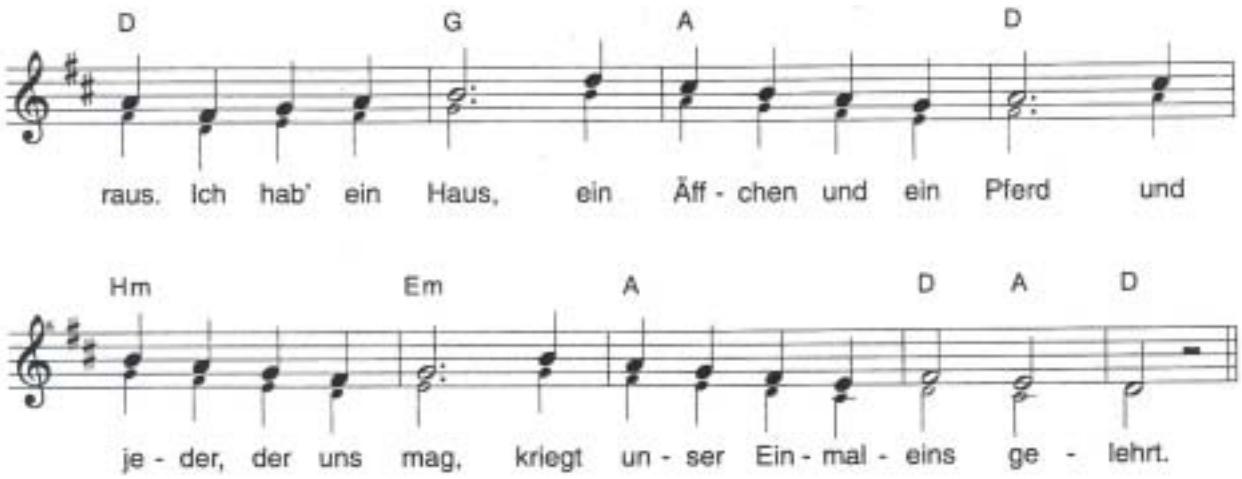
5.8.) Song h: The little drummer boy:



```
// 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";
```

5.9.) Song i: Hey, Pippi Langstrumpf:

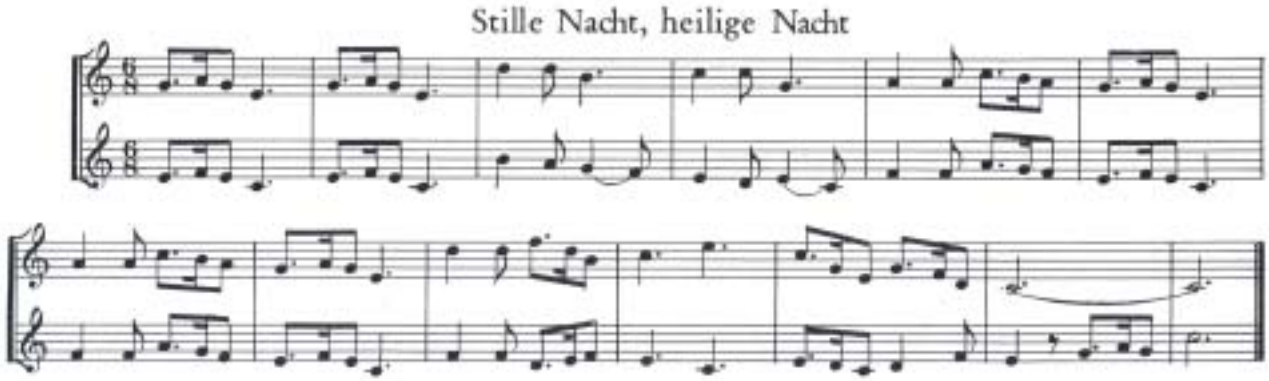




2. Zwei mal drei macht vier, wide wide witt und drei macht neune,
wir machen uns die Welt, wide wide wie sie uns gefällt.
Drei mal drei macht sechs, wide wide wer will's von uns lernen?
Alle, groß und klein, tralala, lad' ich zu uns ein.
Ref.: Hey, Pippi Langstrumpf,...

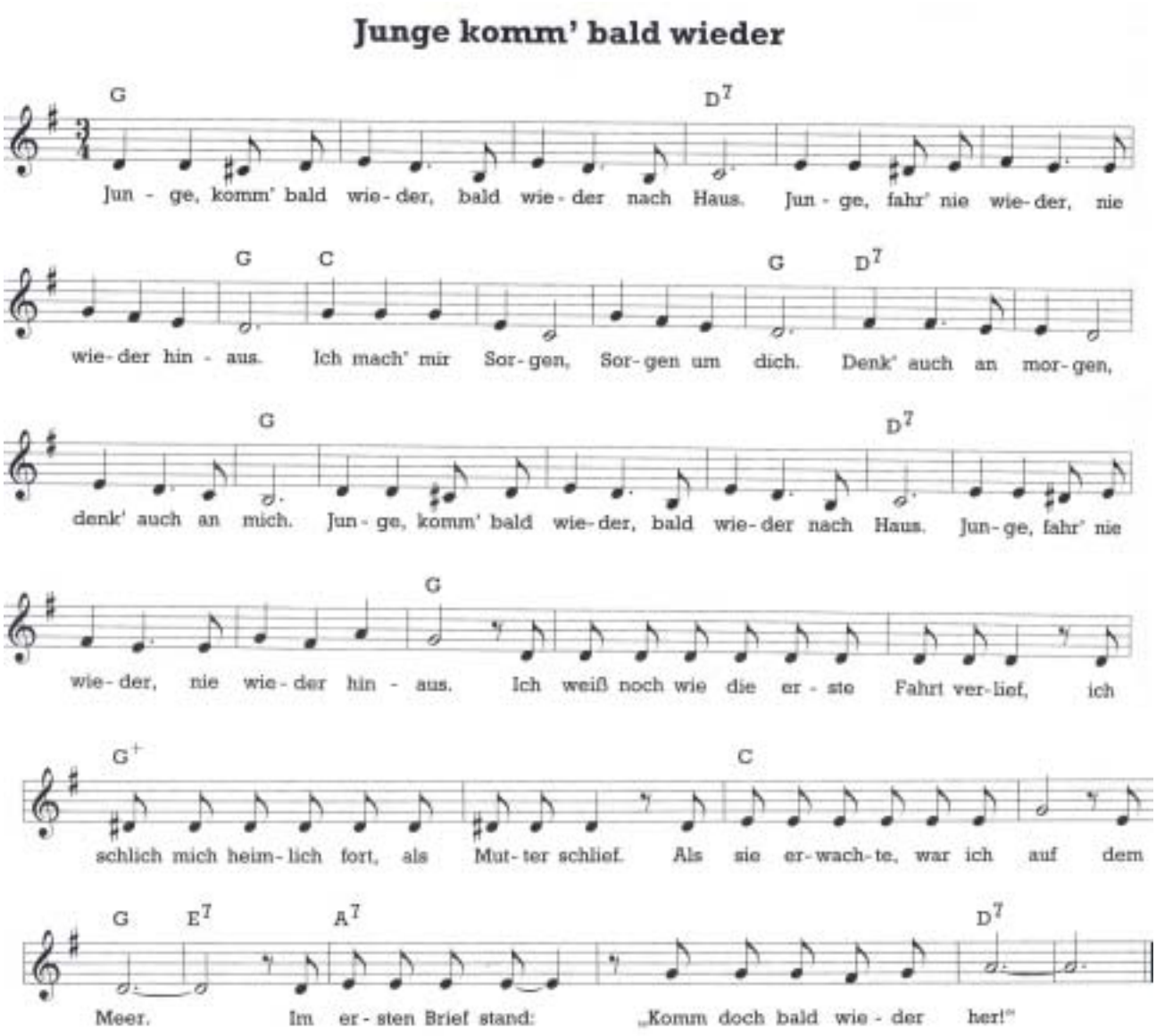
```
// Hey, Pippi Langstrumpf (song i):
code unsigned char
songi[]="T180OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLAONO0L4C+L2DF+OLL4AONO
0L4DF+DL2EL8GF+EDL4C+EOLL4AONO0L4C+DP4P2OLL4AONO0L4DF+DL2EL8GF+ED
L4C+EOLAONO0L4C+L2DF+OLL4AONO0L4DF+DL2EL8GF+EDL4C+EOLL4AONO0L4C+
DP4P2O0L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+
EEDC+DP4L2F+GAH.O1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+GL
4AF+GAL2H.O1L4DC+O0L4HAGL2A.O1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+EDP2";
```

5.10.) 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.";
```

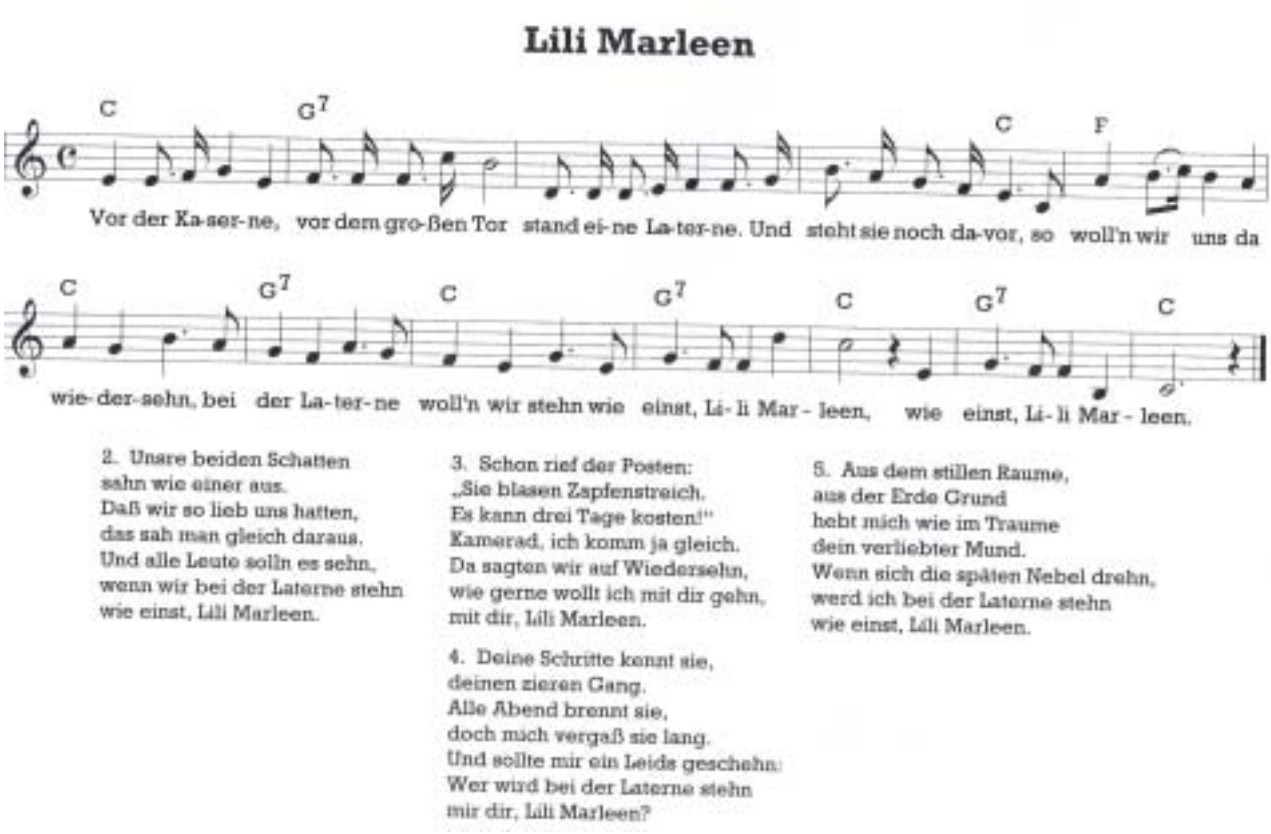
5.11.) Song k: Junge komm bald wieder:



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

```
// Junge komm bald wieder (song k):
code unsigned char
songk[]="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.";
```

5.12.) Song l: Lili Marleen:

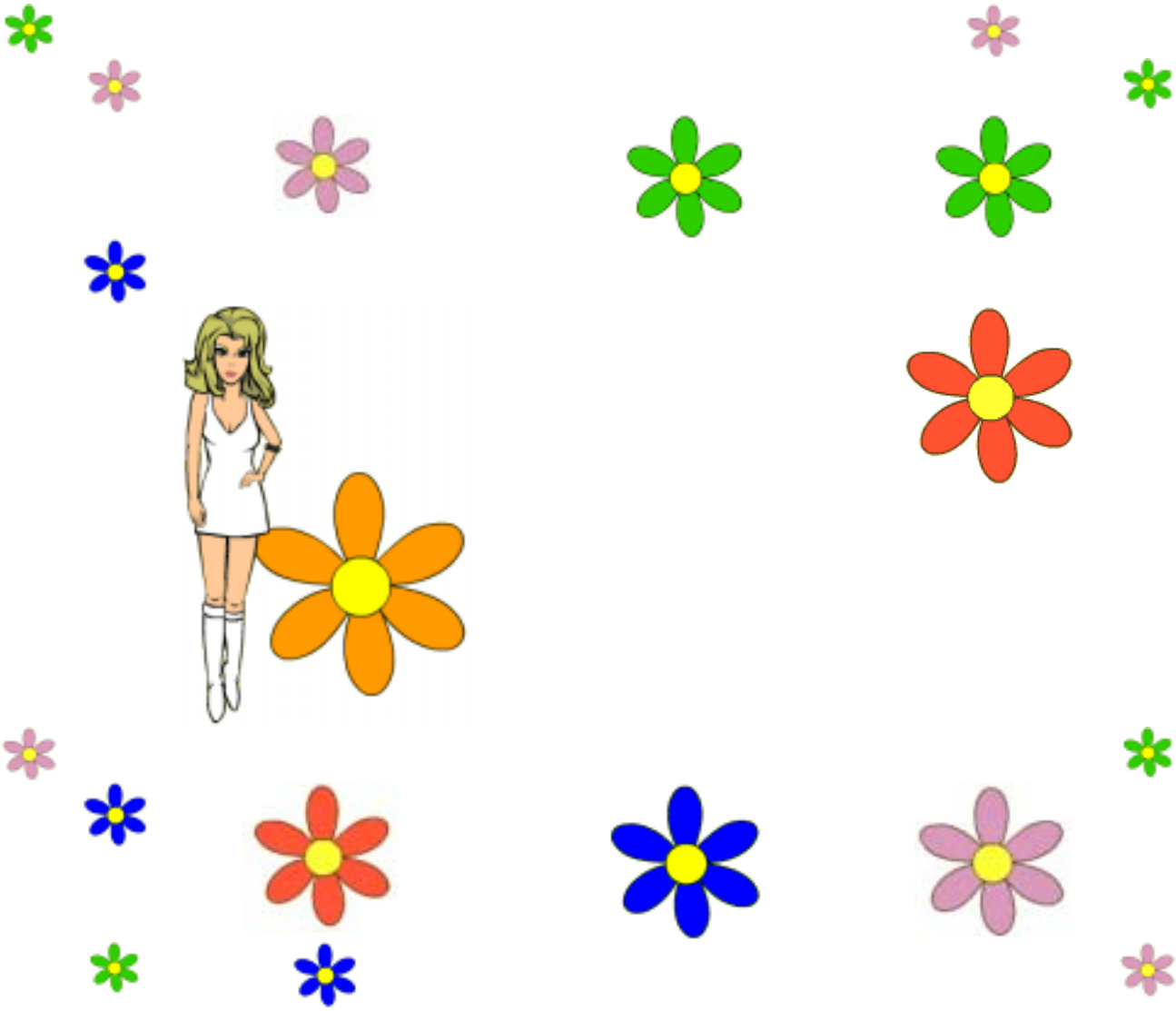


// Lili Marleen (song l):
code unsigned char songl[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HONO0L2C.";

5.13.) Song m: musical scale / chromatic scale / for testing purpose / Tonleiter:

```
// musical scale / chromatic scale / for testing purpose / Tonleiter (song m):
code unsigned char
songi[]="T120O0L4CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+AA+HO2CC+DD+EFF+G
G+AA+HO3CC+DD+EFF+GG+AA+HP4O0L8CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+
AA+HO2CC+DD+EFF+GG+AA+HO3CC+DD+EFF+GG+AA+HP8O0L16CC+DD+EFF+GG+A
A+HO1CC+DD+EFF+GG+AA+HO2CC+DD+EFF+GG+AA+HO3CC+DD+EFF+GG+AA+HP16
";
```

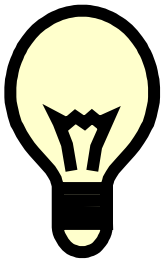
5.14.) Another song: Lady Bird:



// Lady Bird:

```
"T150ON00L4F+P16F+P16F+.P8L16C+P16L8EP16E.P16L2EL8EP16L4C+P16C+P16C+.P160LL8AP16
L4HP16G+P16L2EP4ON00L4F+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 CAPCOM 6 / CCU6 / 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, and XE16x) and 32 bit microcontrollers (e.g. TC1796, TC1766 and TC1130).

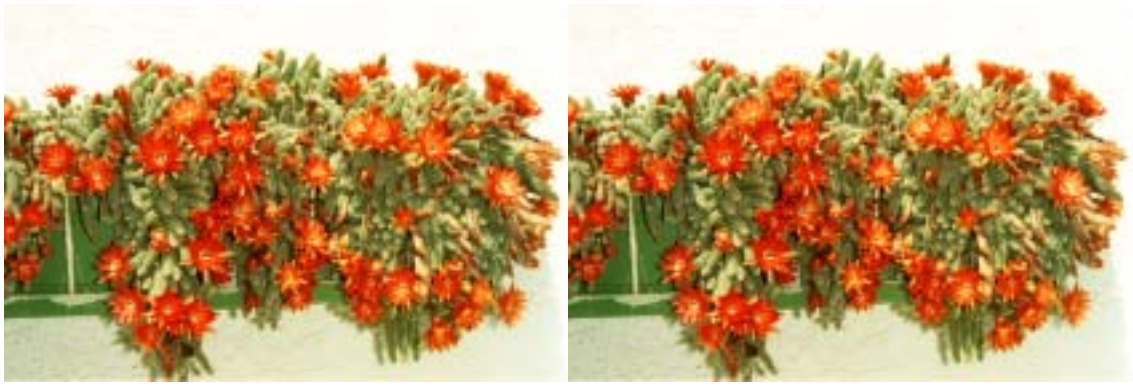
All these step-by-step books use the same microcontroller resources and the same example code.

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.

6.) Thanks To



Maria, Christian, Hermann and Maureen for their support.

7.) Feedback (XC878 Playing Music using DAvE Bench): 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