

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**

**© Infineon Technologies AG 2010.
All Rights Reserved.**

LEGAL DISCLAIMER

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

Information

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

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



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).

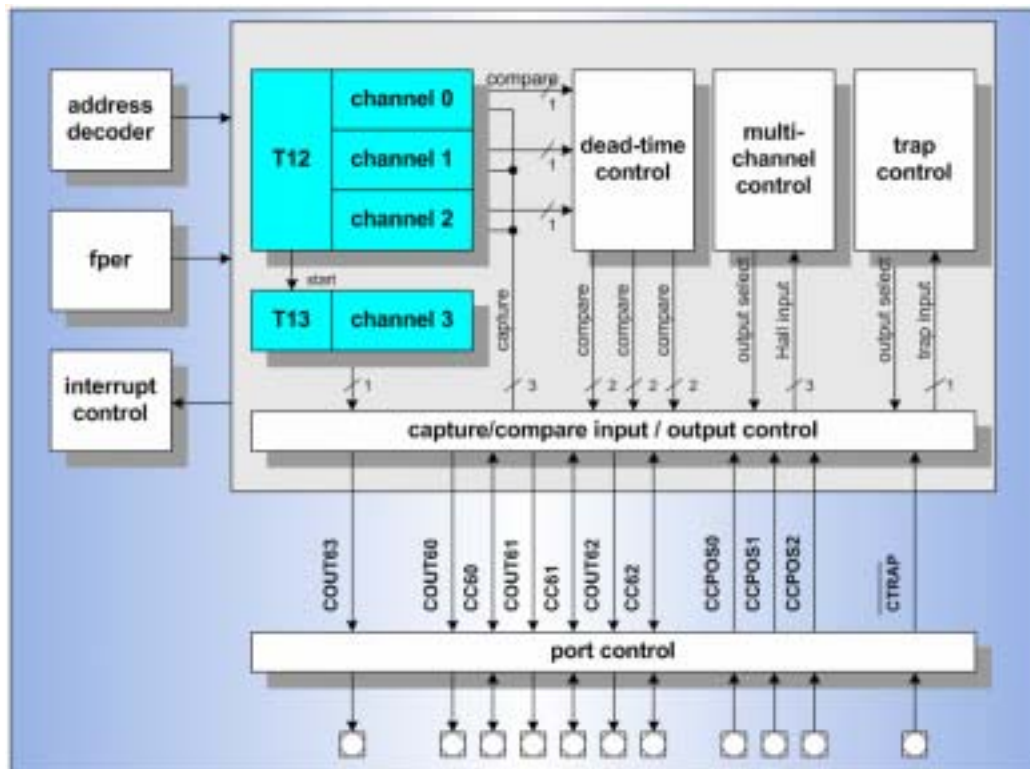


Programming Example

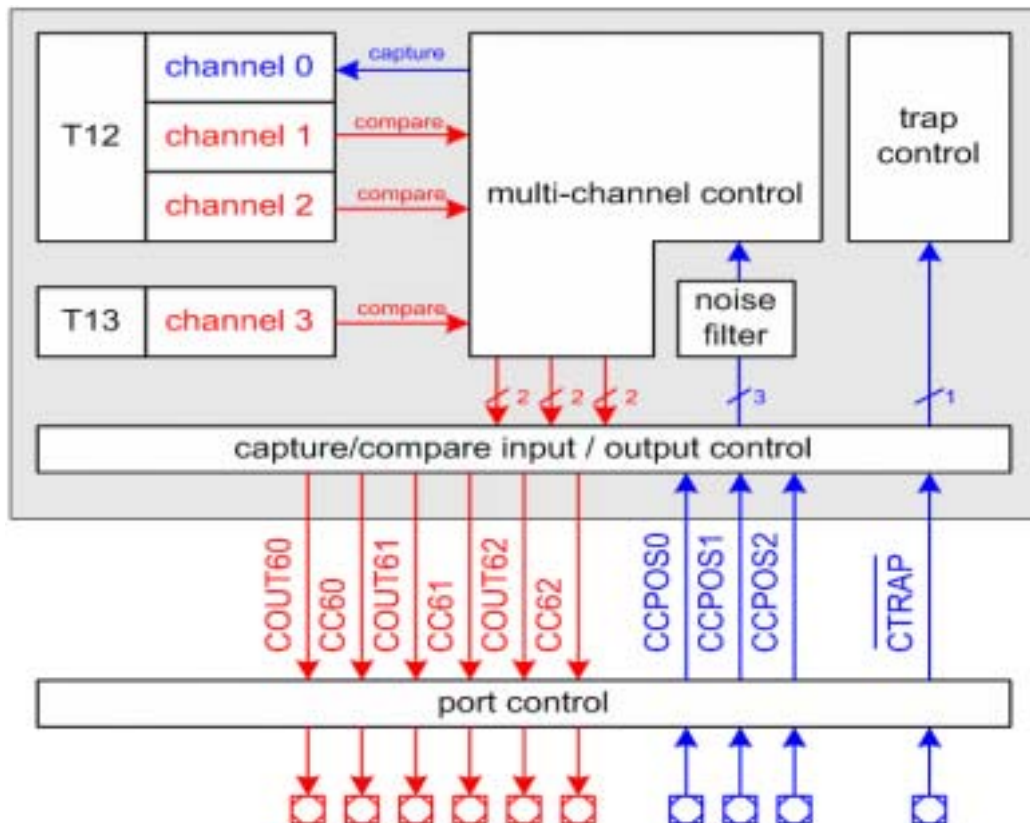
XC878 Easy Kit



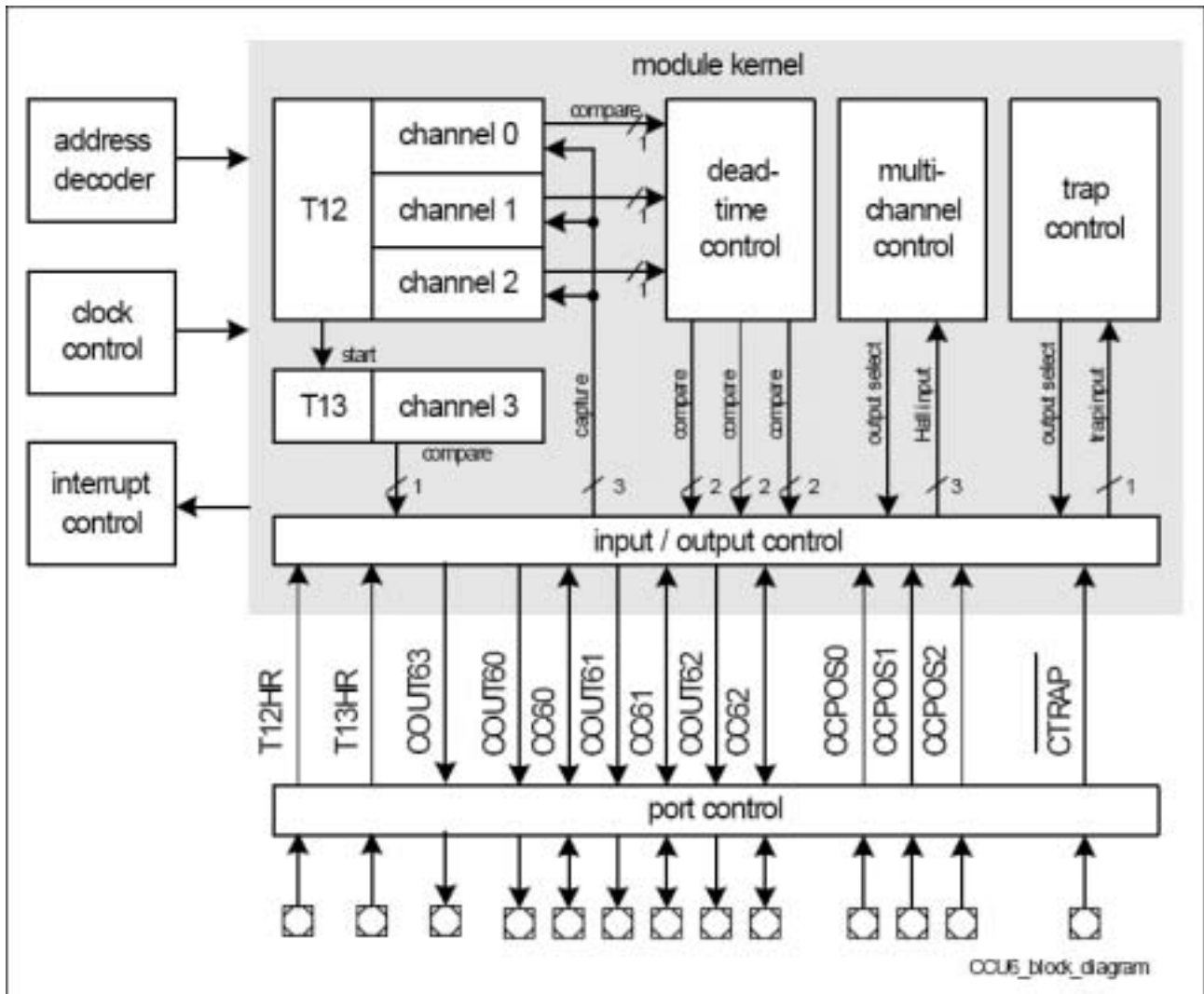
CAPCOM 6 Block Diagram – general use (Source: Product Marketing)



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



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

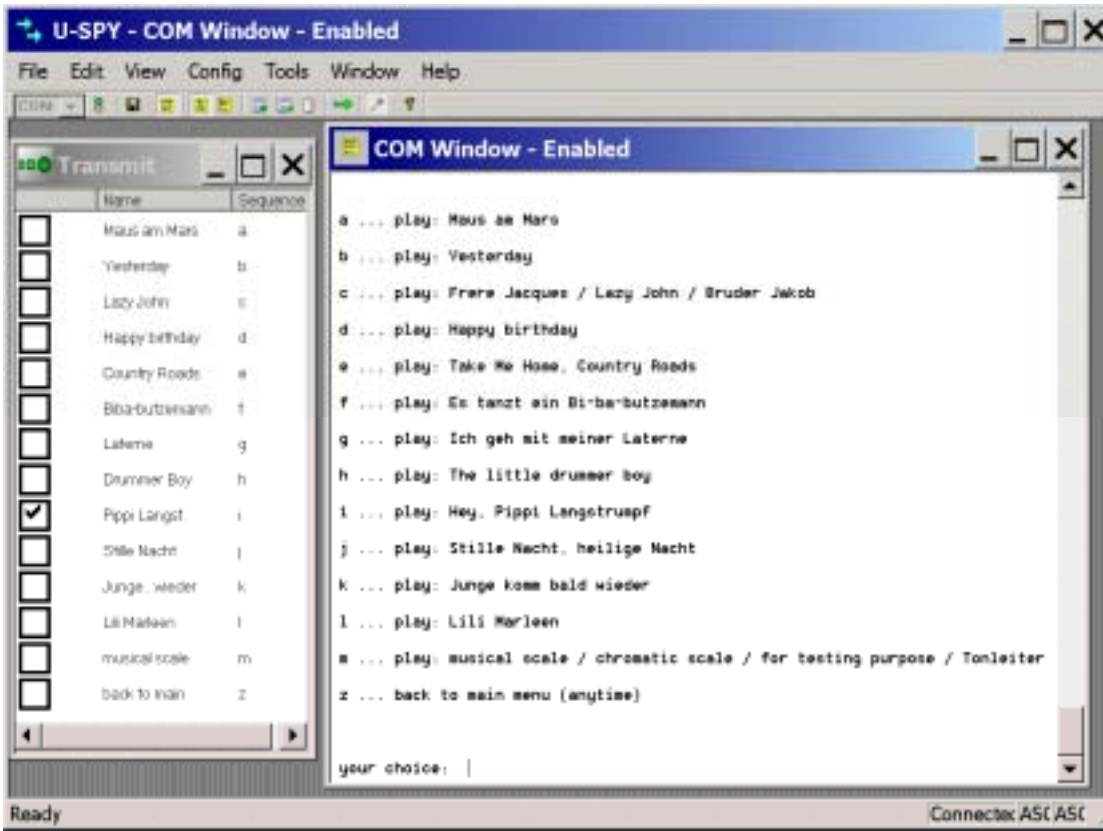


Note:

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	<u>Music Output:</u> 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)

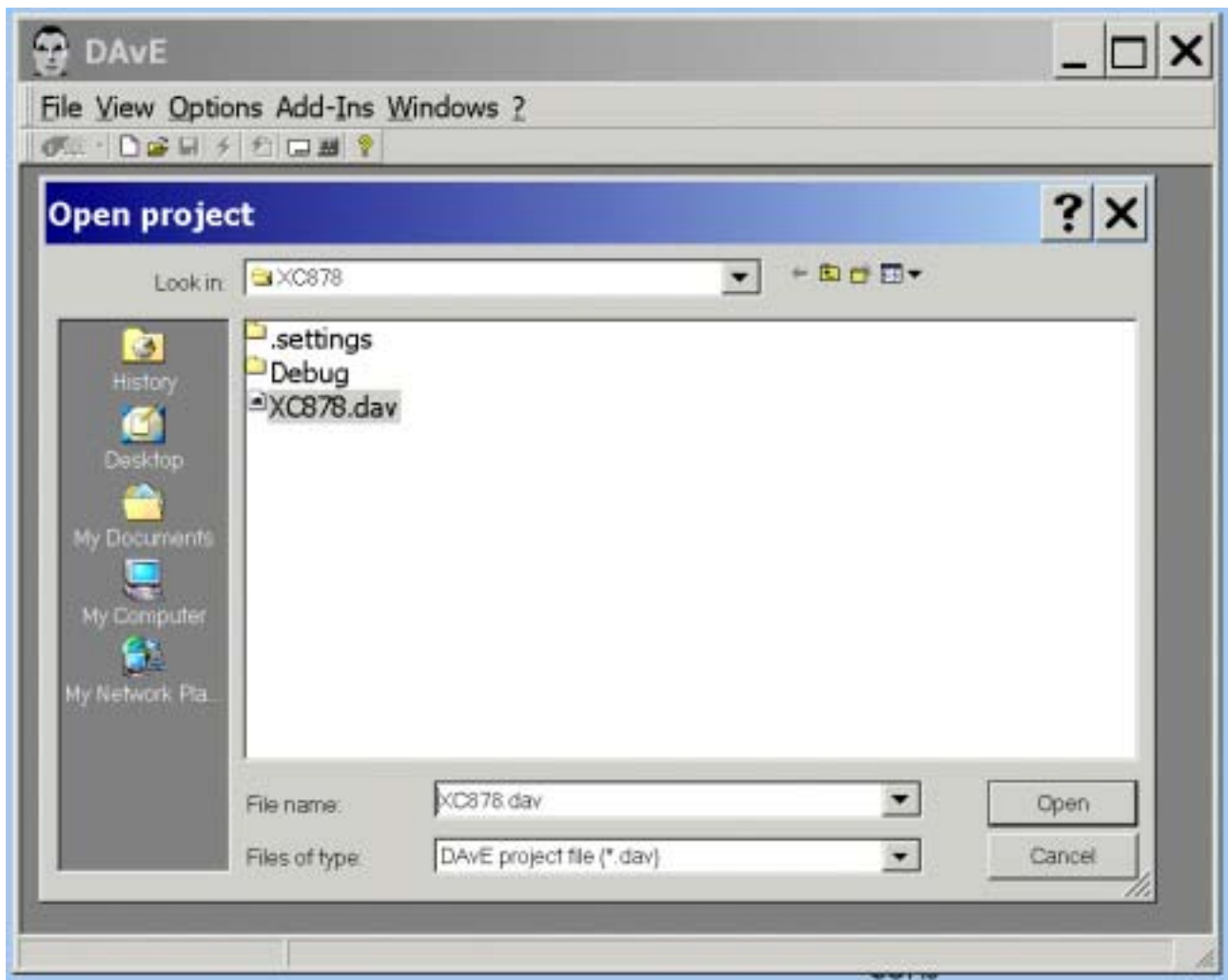
View - Command Window (Closes the Command Window)

File

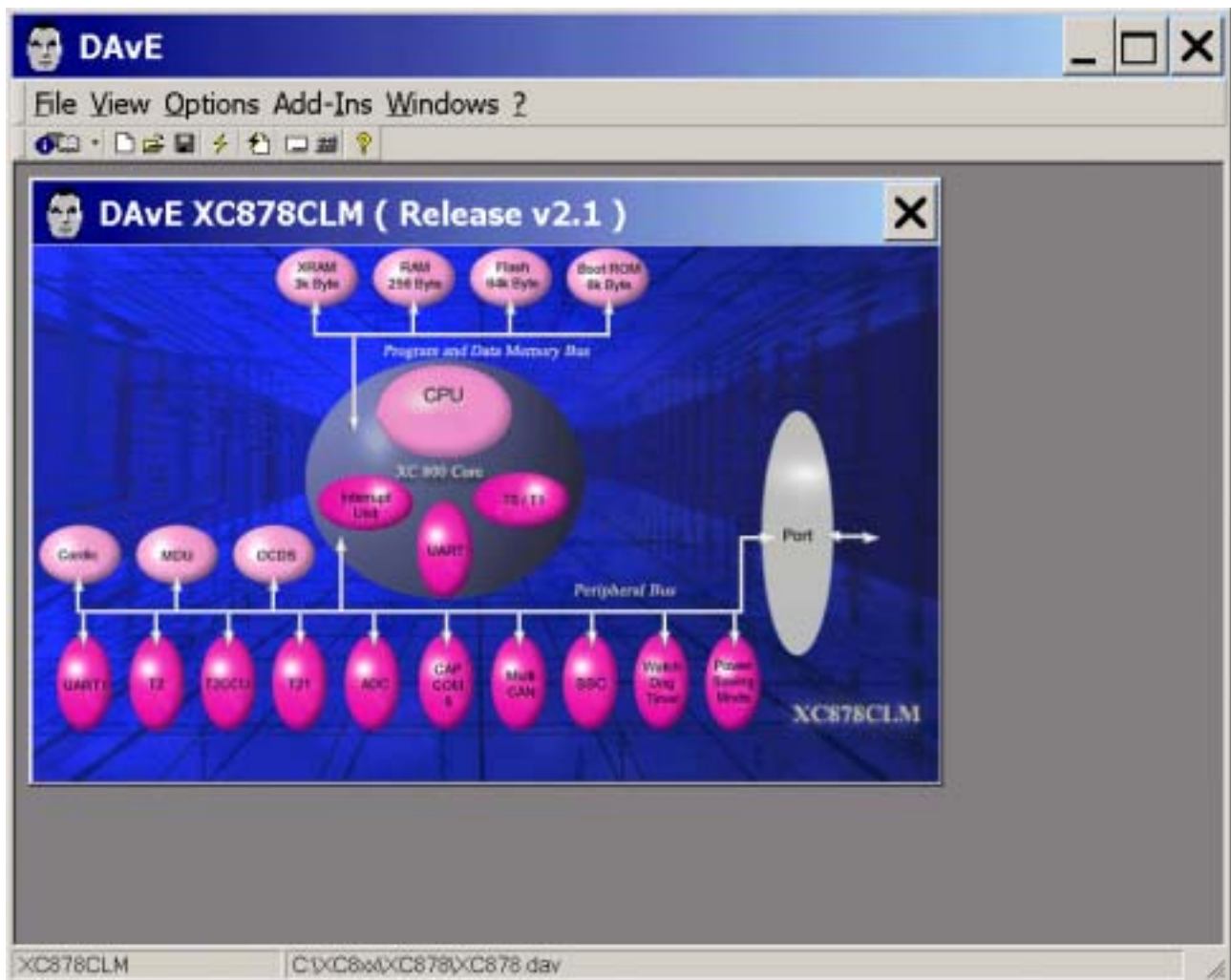
Open

Location: [C:\XC8xx\XC878](#)

Filename: [XC878.dav](#)



Click Open



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):

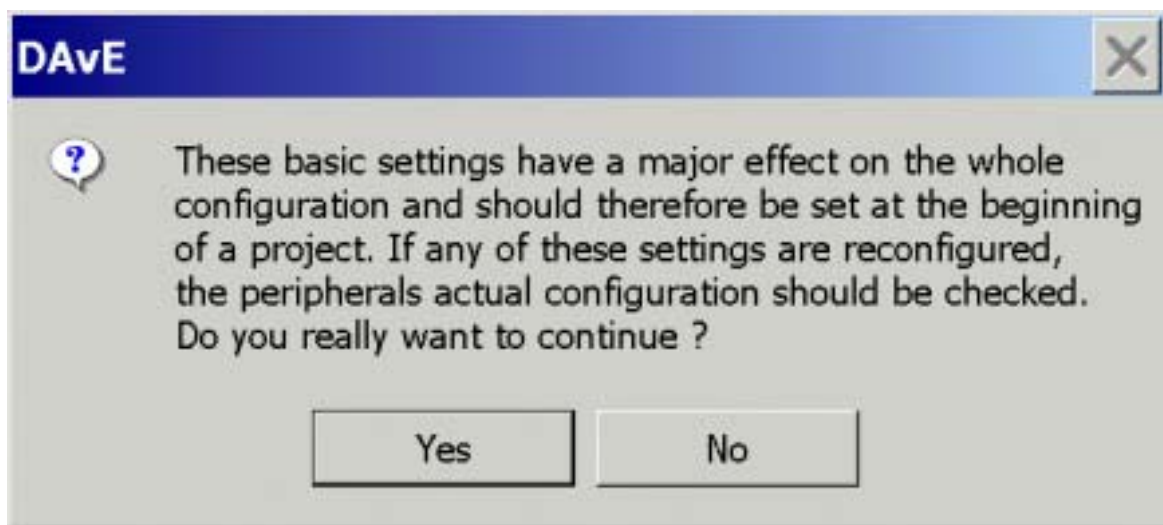
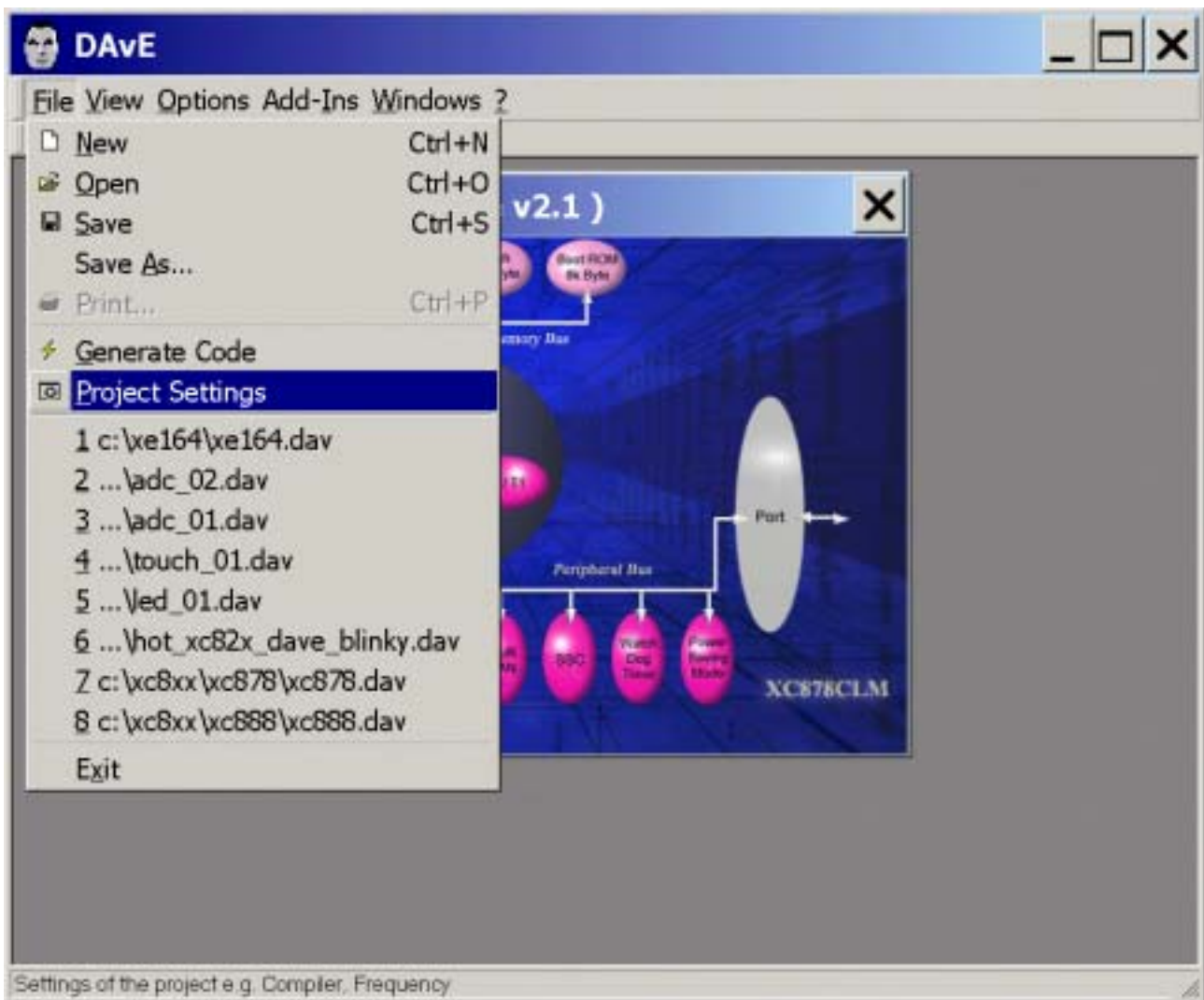
Floats are enabled in large memory model only due to code size limitation. So, print routines like `printf()`, `sprintf()` etc., will output `<NO_FLOAT>` when floats are used with these routines in small and medium memory models.

➔ 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

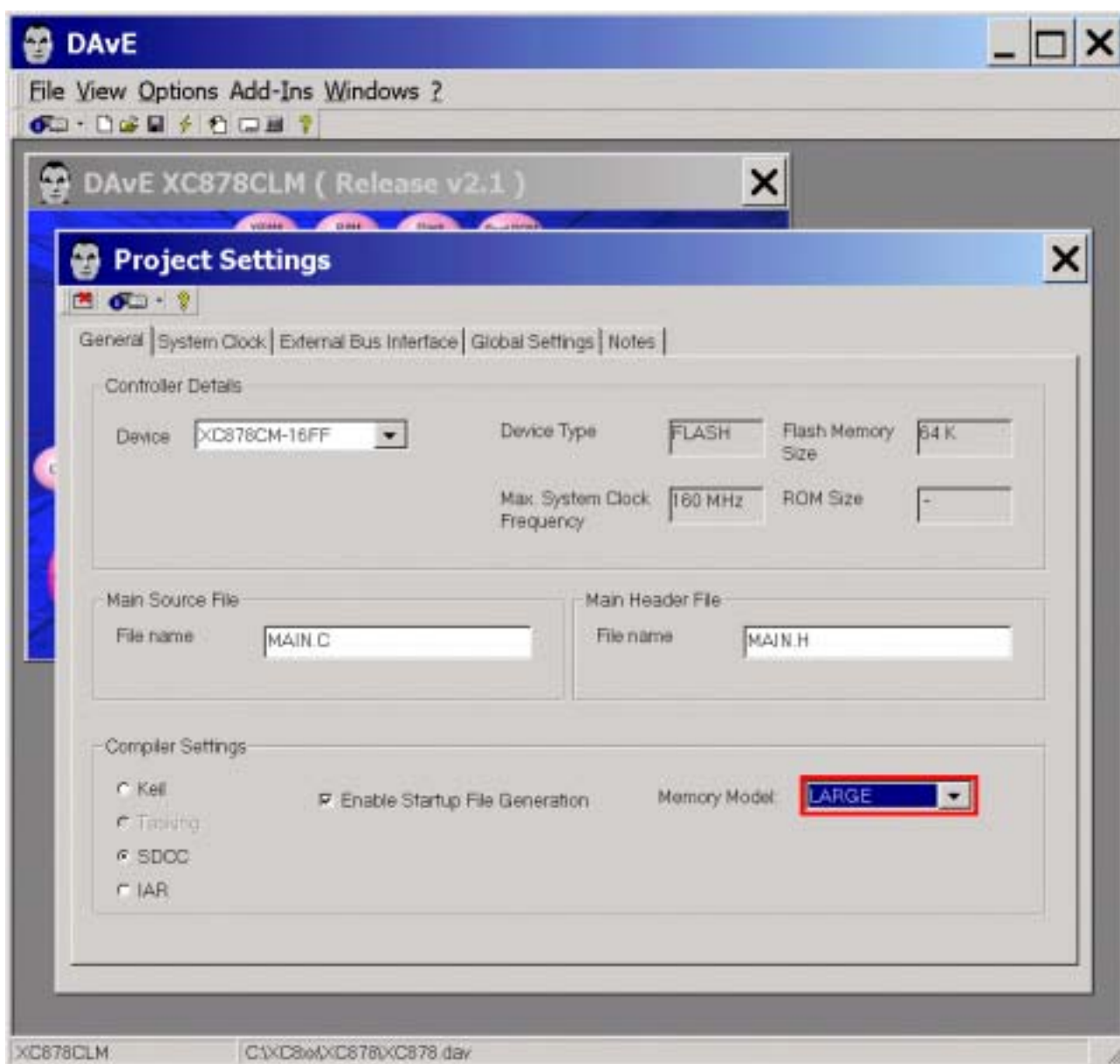
➔ Therefore we are going to use other print formats like `printf_small()` and `printf_fast_f()`.

File – Project Settings



Click Yes

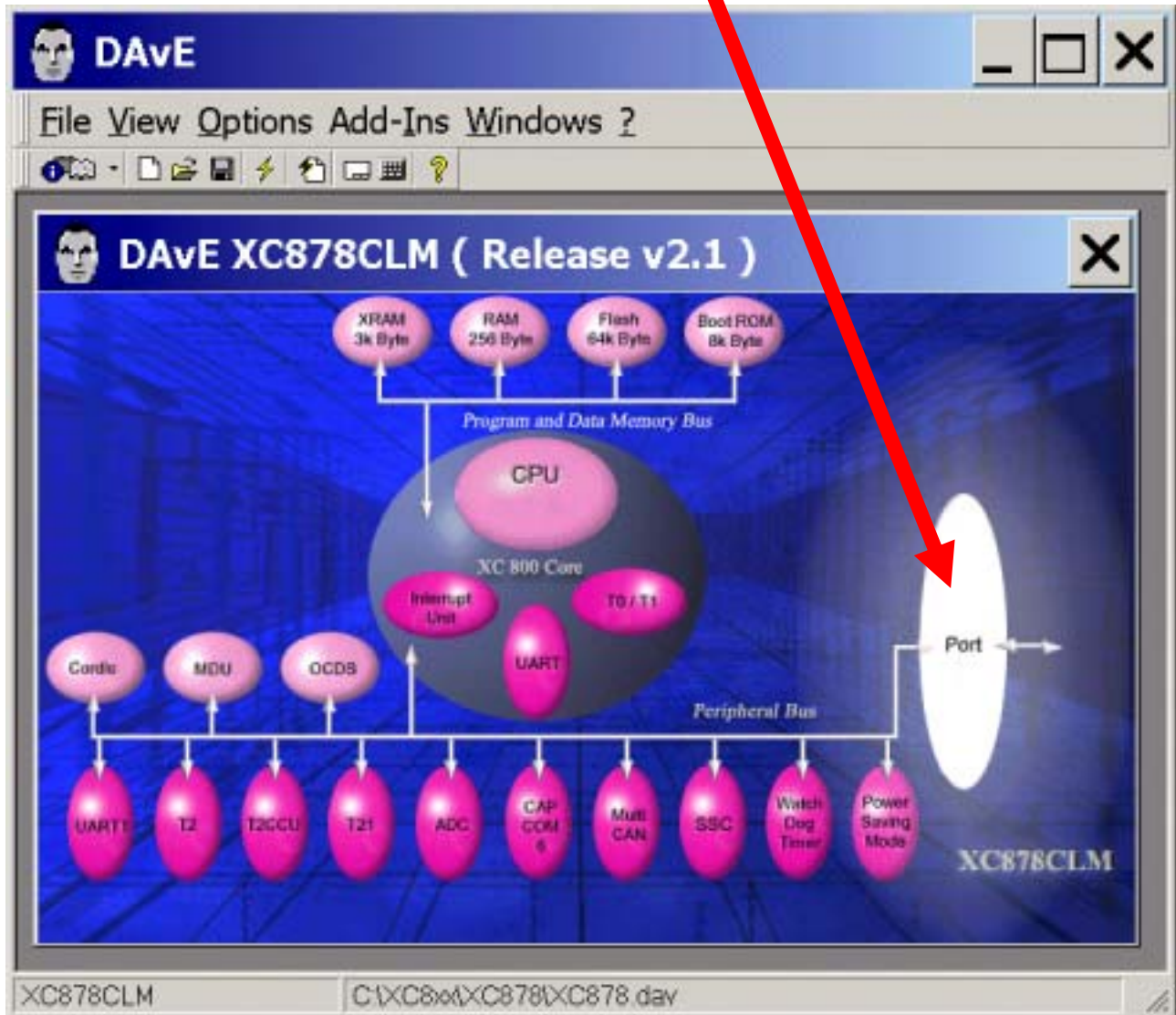
General: Compiler Settings: Memory Model: select Large



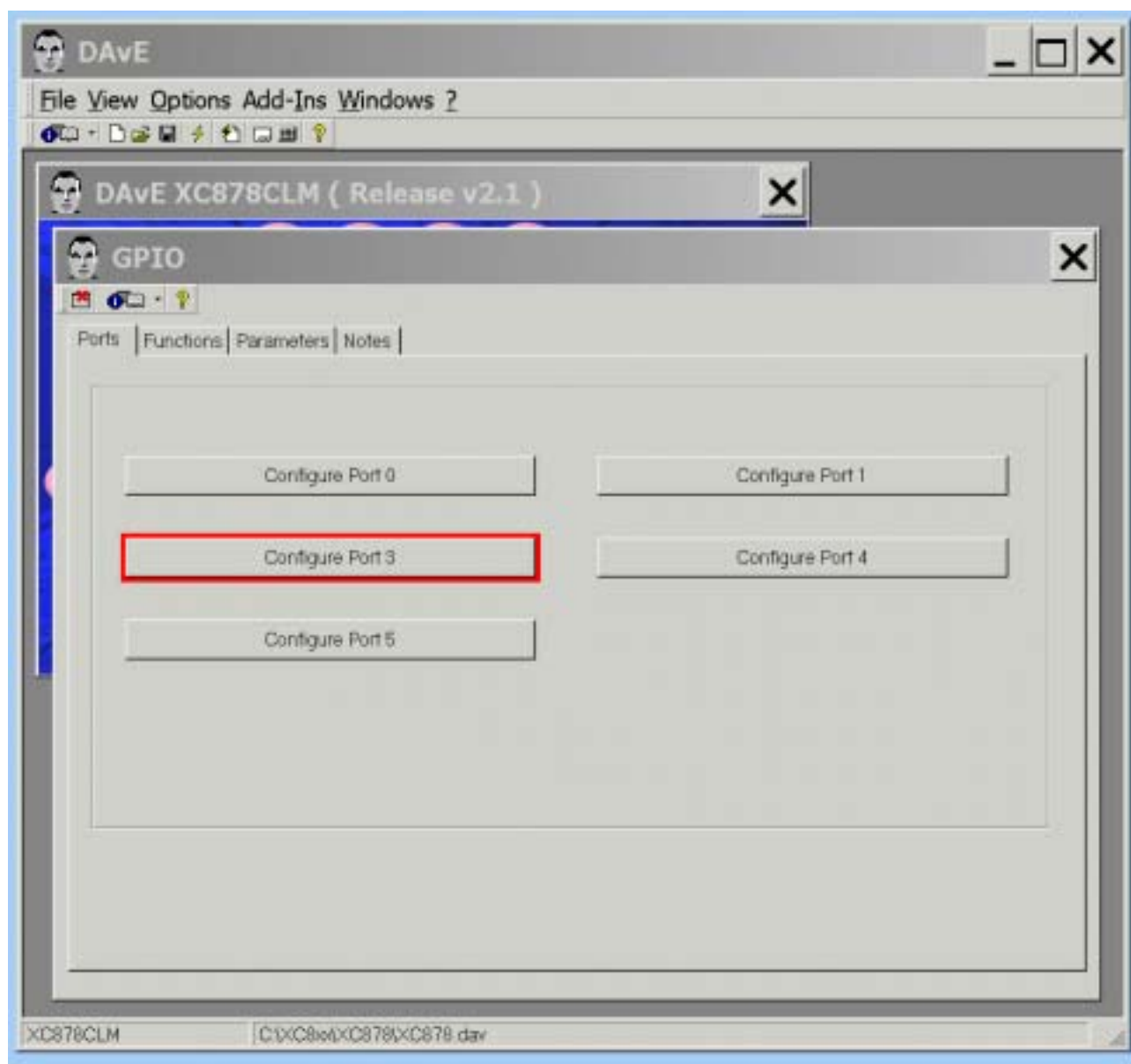
Exit this dialog now by clicking  the close button.

Reconfiguration of Port 3:

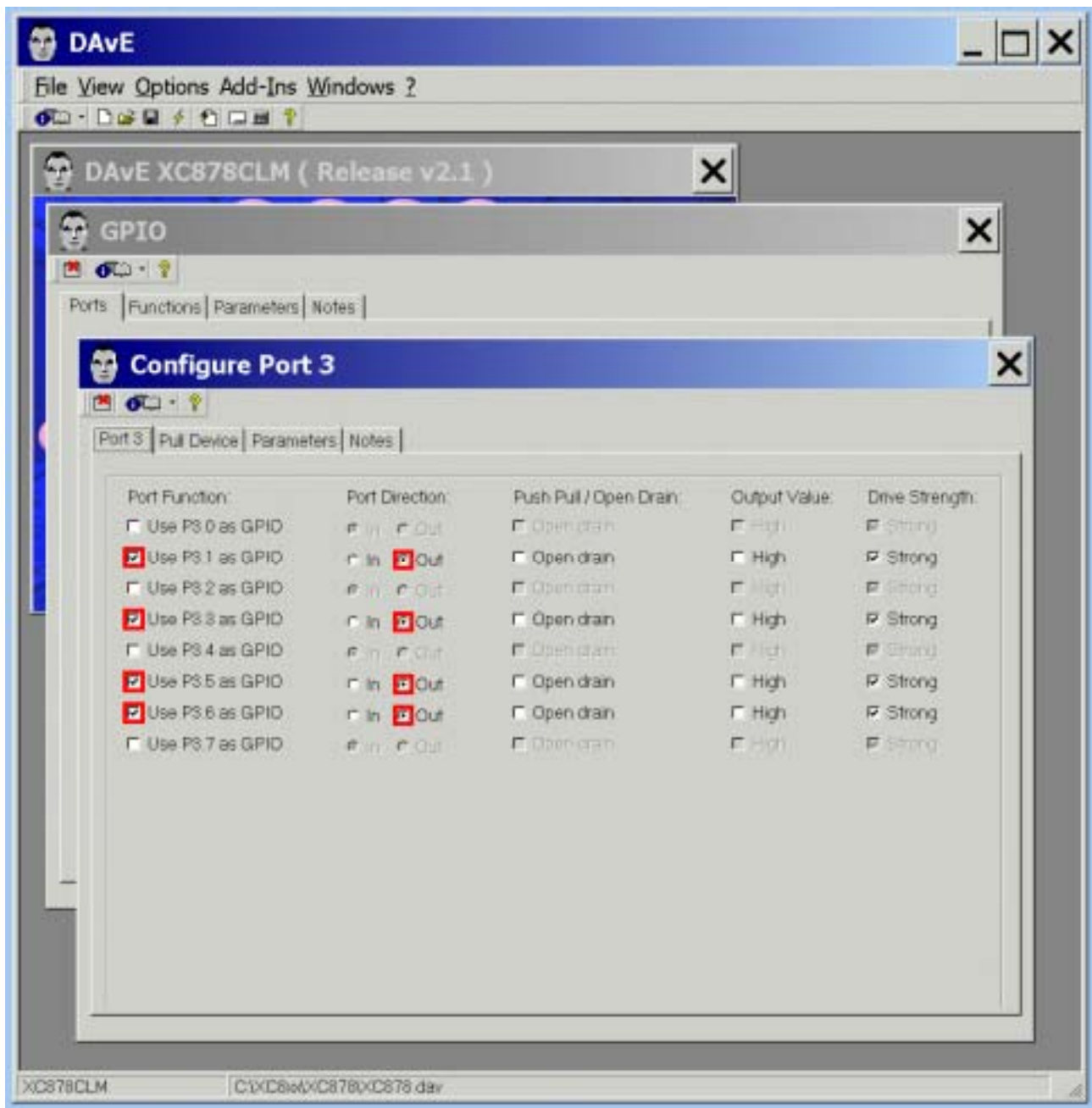
The (re)configuration window/dialog can be opened by clicking the specific block/module (Port).



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)

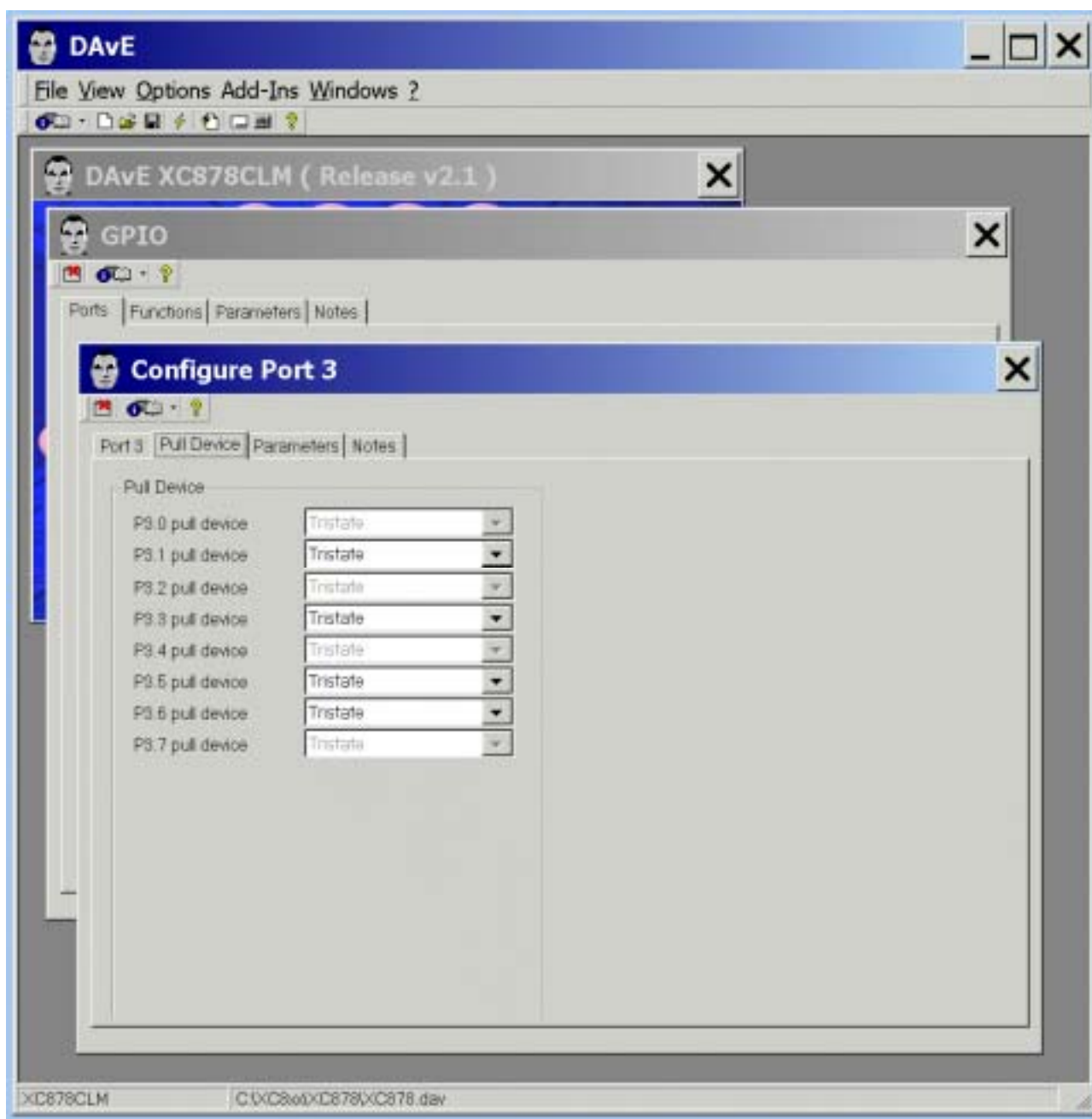
Port_3 pins used as GPIO:

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

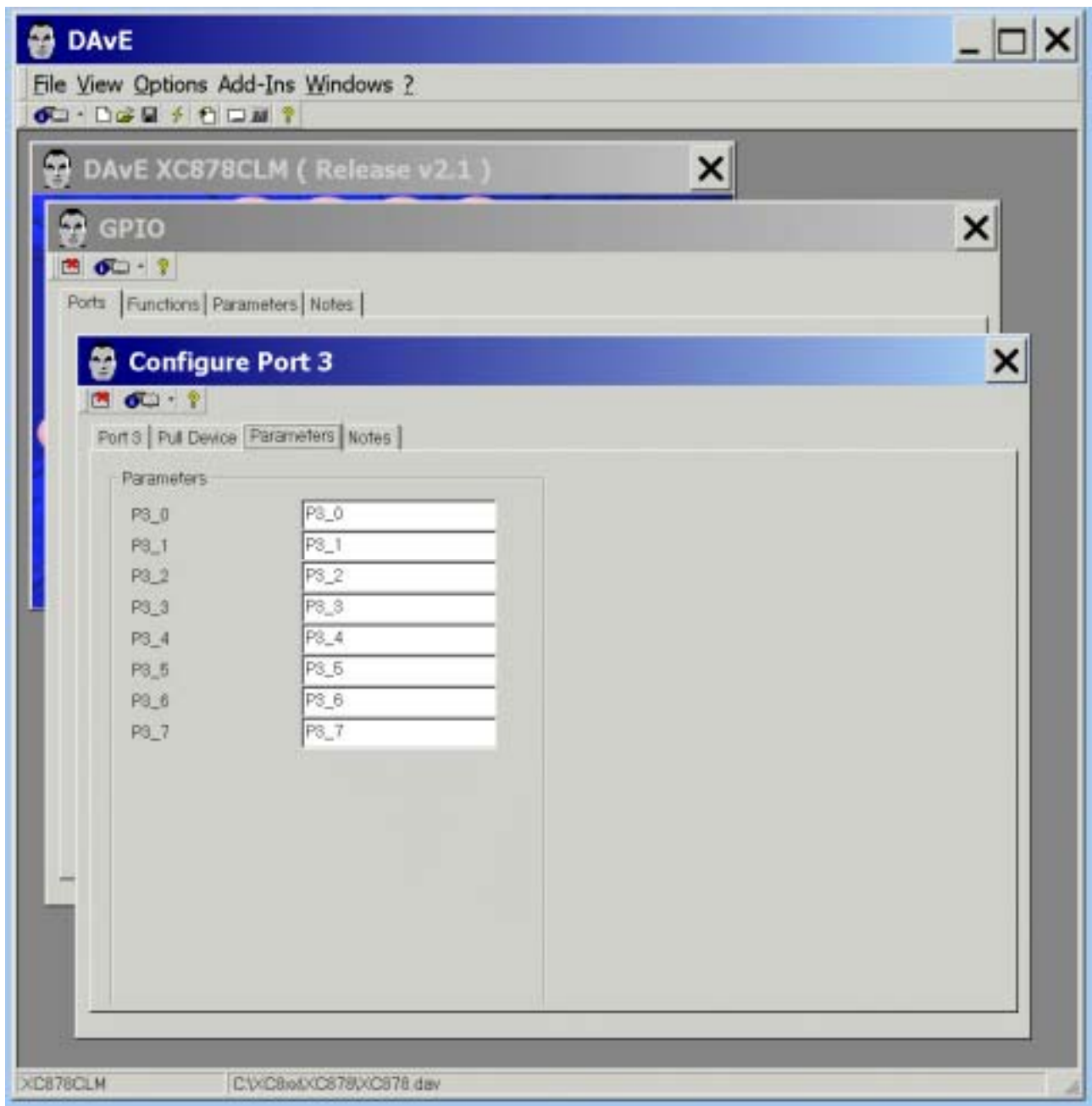
Port 3:

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	<u>Music Output:</u> 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	

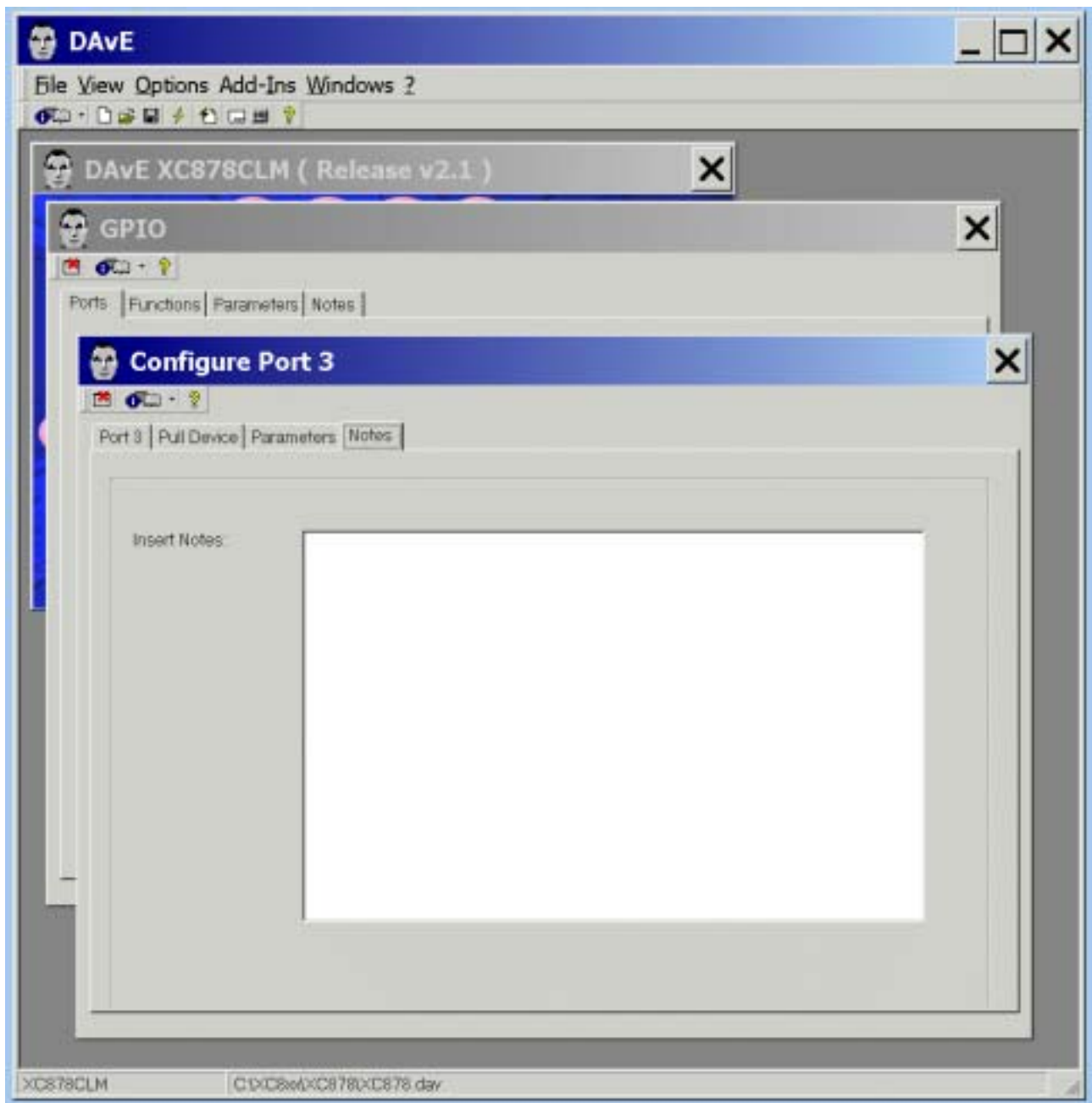
Pull Device: (do nothing)



Parameters: (do nothing)



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

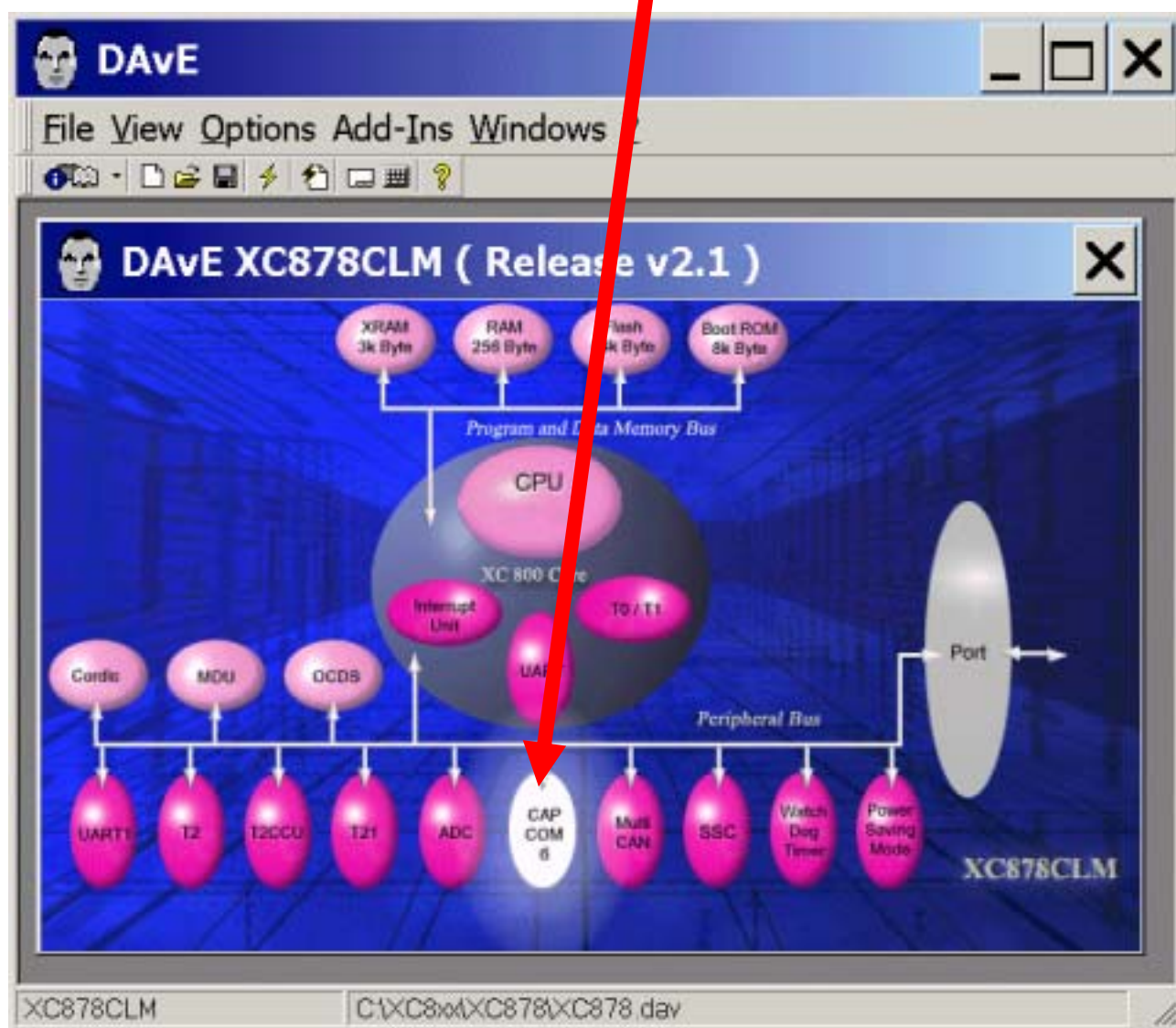


Exit this dialog now by clicking  the close button.

Exit this dialog now by clicking  the close button.

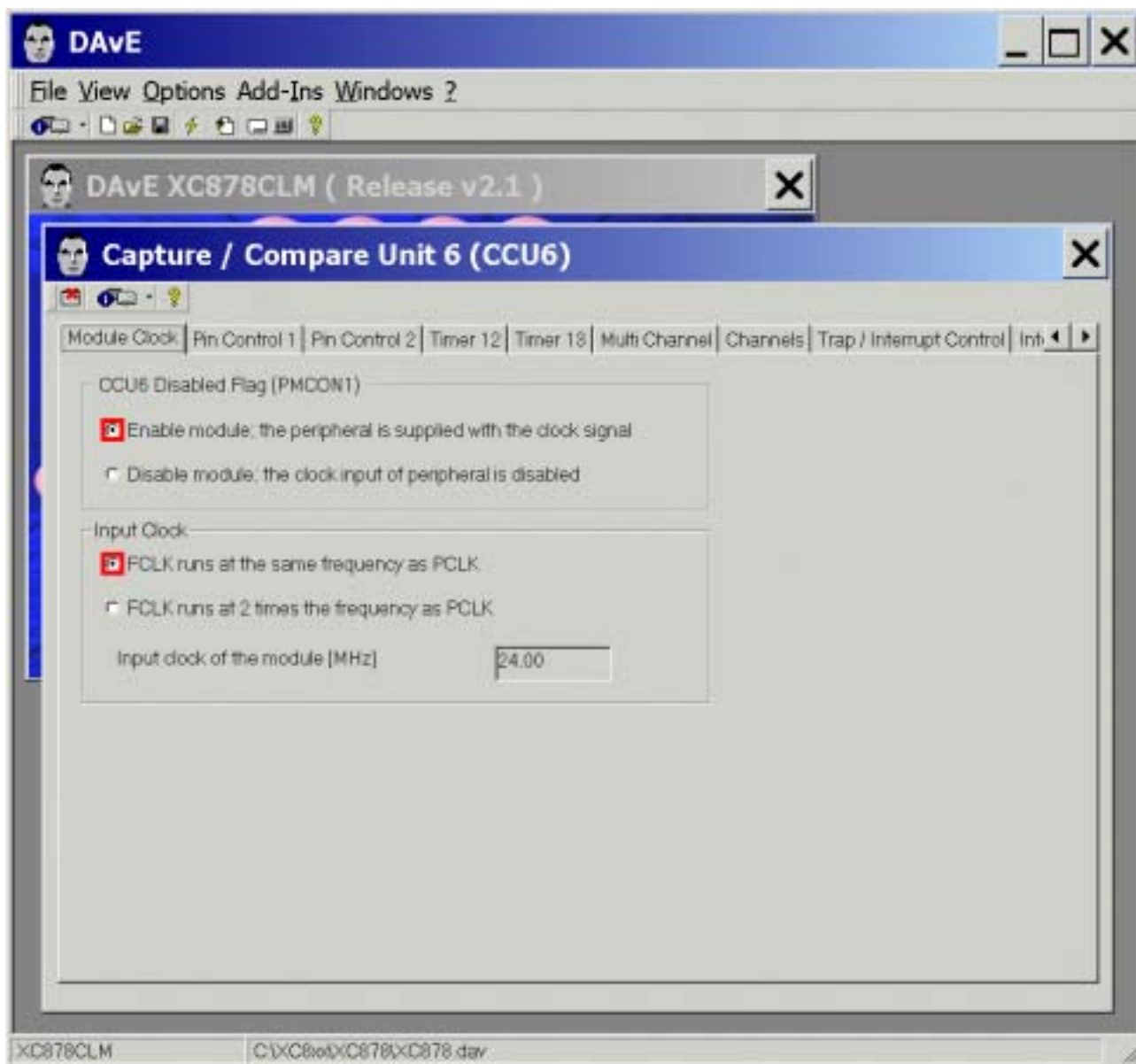
Configuration of the CAPCOM 6 module:

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

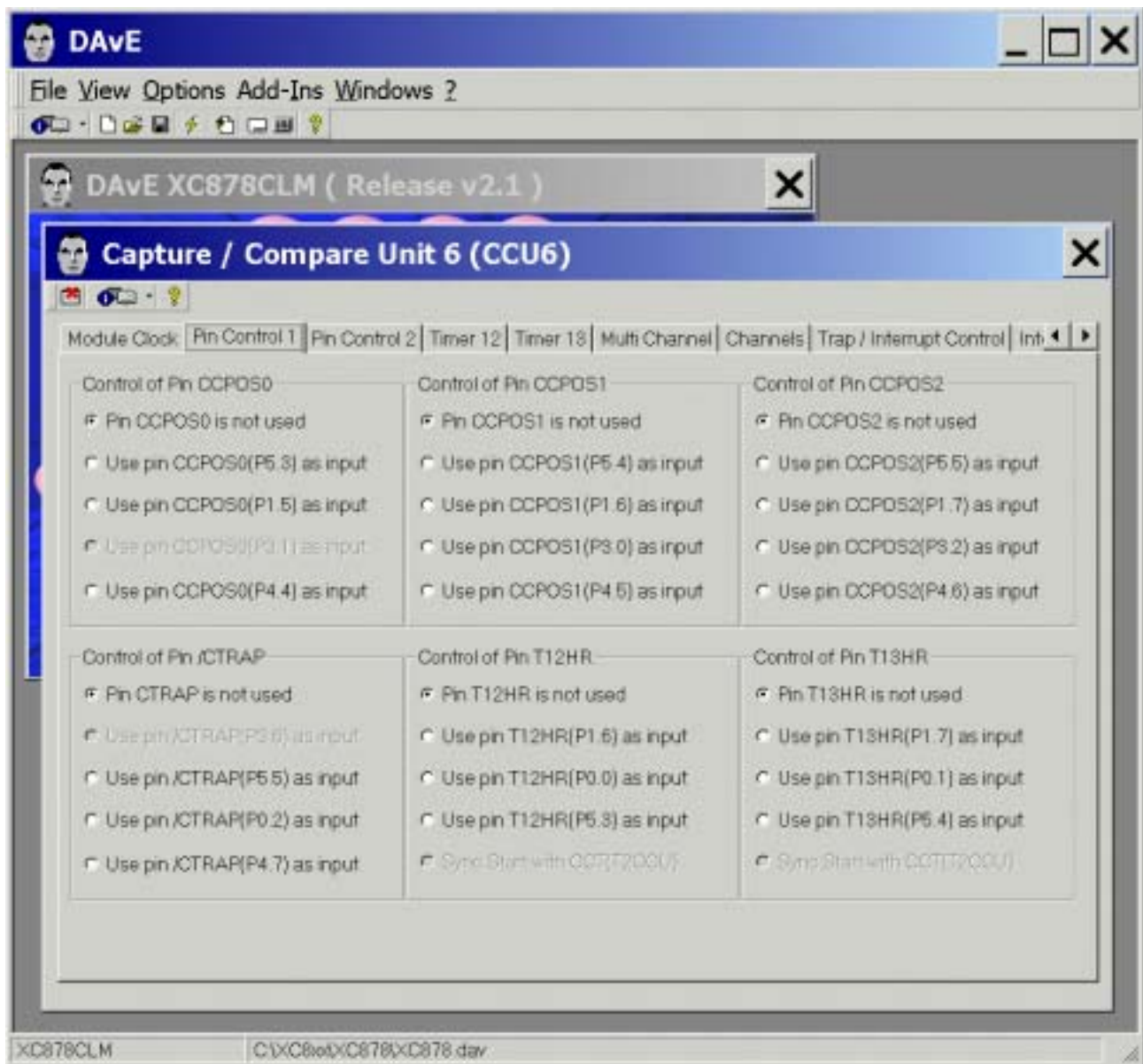


CCU6: Module Clock: CCU6 Disable Flag: **click/check** ☒ Enable module

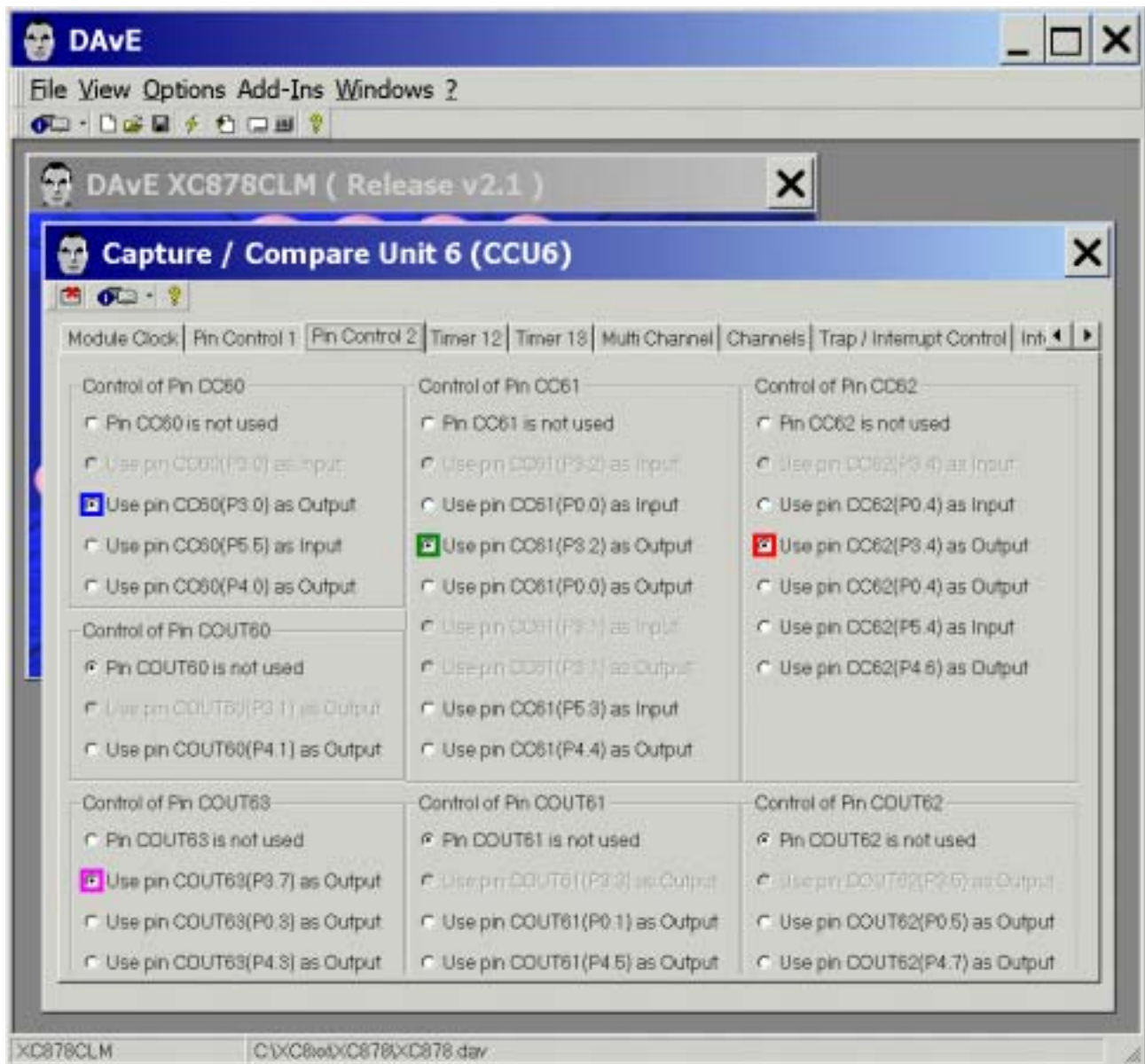
CCU6: Module Clock: Input Clock: **click** ☒ FCLK runs at the same frequency as PCLK



CCU6: Pin Control 1: (do nothing)

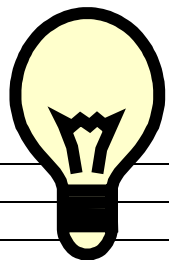


CCU6: Pin Control 2: Control of Pin CC60: **click** ☉ Use pin **CC60** (P3.0) as output
 CCU6: Pin Control 2: Control of Pin CC61: **click** ☉ Use pin **CC61** (P3.2) as output
 CCU6: Pin Control 2: Control of Pin CC62: **click** ☉ Use pin **CC62** (P3.4) as output
 CCU6: Pin Control 2: Control of Pin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



Remember:

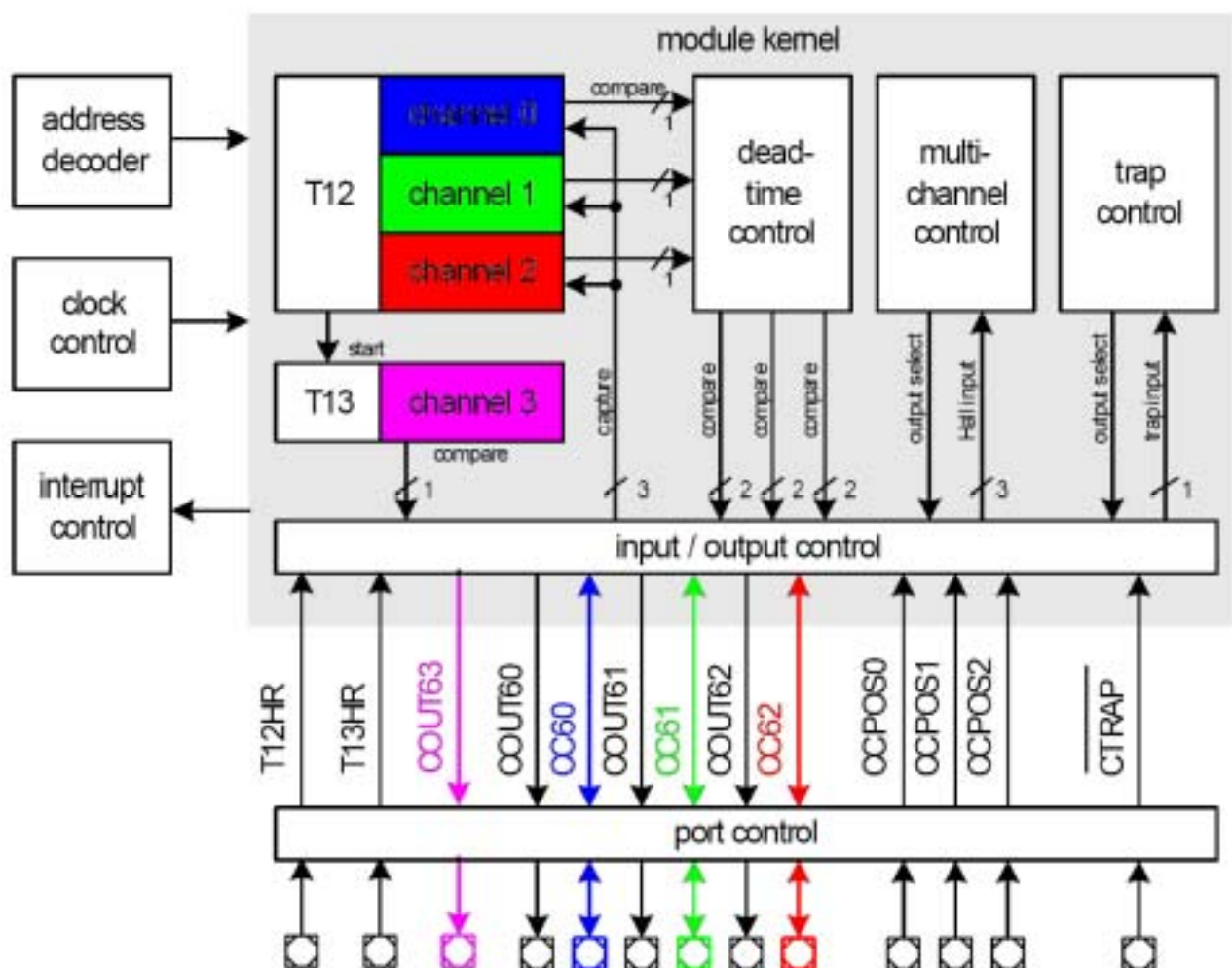
Port_3 pins used by our PWM module:



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:



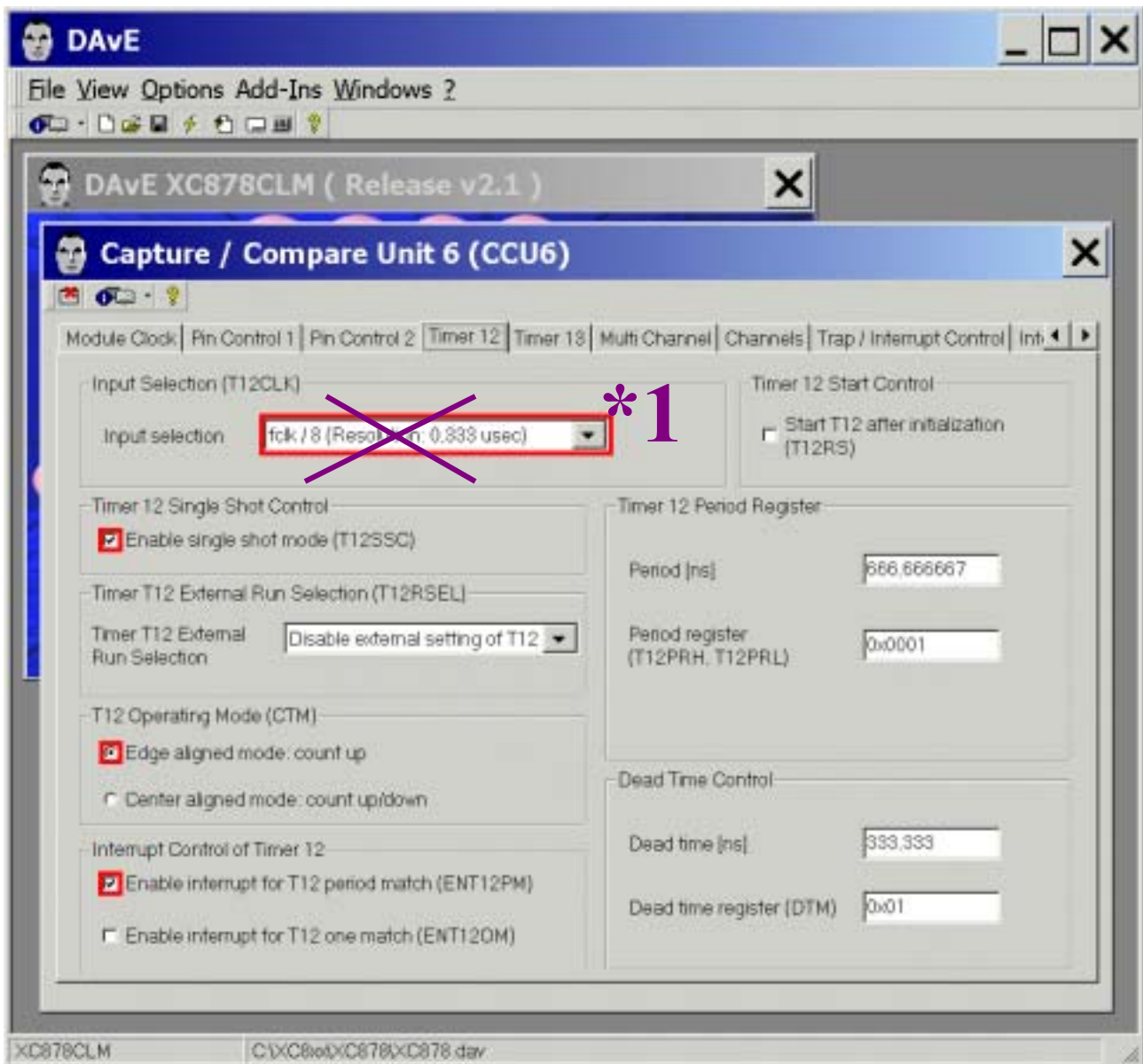
Timer 12: “note length”:

CCU6: Timer 12: Input Selection (T12CLK): choose $f_{clk} / 8 \rightarrow$ Resolution = $85,333 \mu s$ *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 = $11,719 \text{ kHz} / 85,333 \mu s$



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

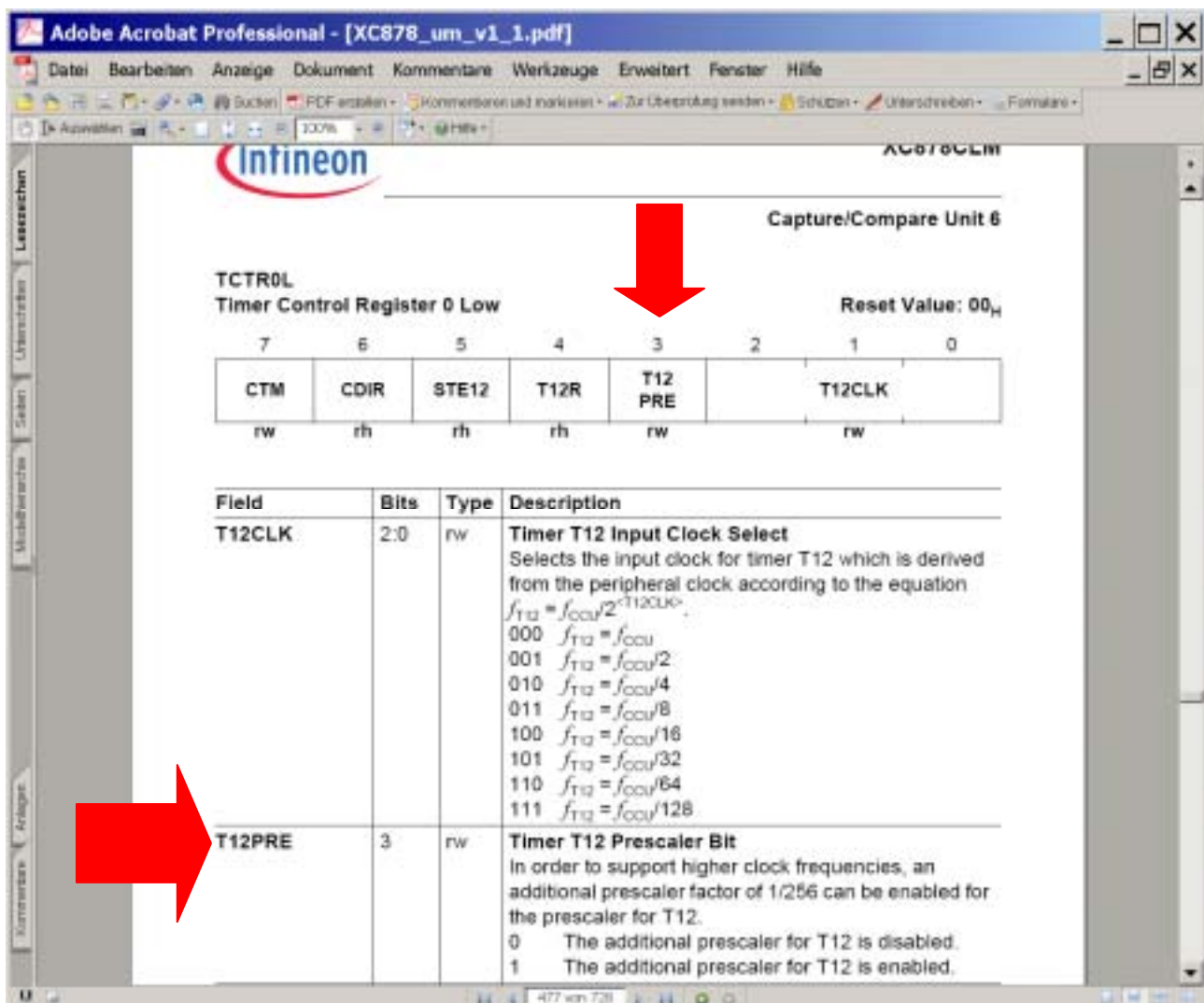


Note:

Unfortunately bit T12PRE is not available in the DAVe dialog.

Source: User's Manual:

The input clock for timer T12 can be from f_{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.



XC878 User Manual - Capture/Compare Unit 6

TCTR0L
Timer Control Register 0 Low

Reset Value: 00_H

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 $f_{T12} = f_{CCU6} / 2^{T12CLK}$. 000 $f_{T12} = f_{CCU6}$ 001 $f_{T12} = f_{CCU6}/2$ 010 $f_{T12} = f_{CCU6}/4$ 011 $f_{T12} = f_{CCU6}/8$ 100 $f_{T12} = f_{CCU6}/16$ 101 $f_{T12} = f_{CCU6}/32$ 110 $f_{T12} = f_{CCU6}/64$ 111 $f_{T12} = f_{CCU6}/128$
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.

*1:

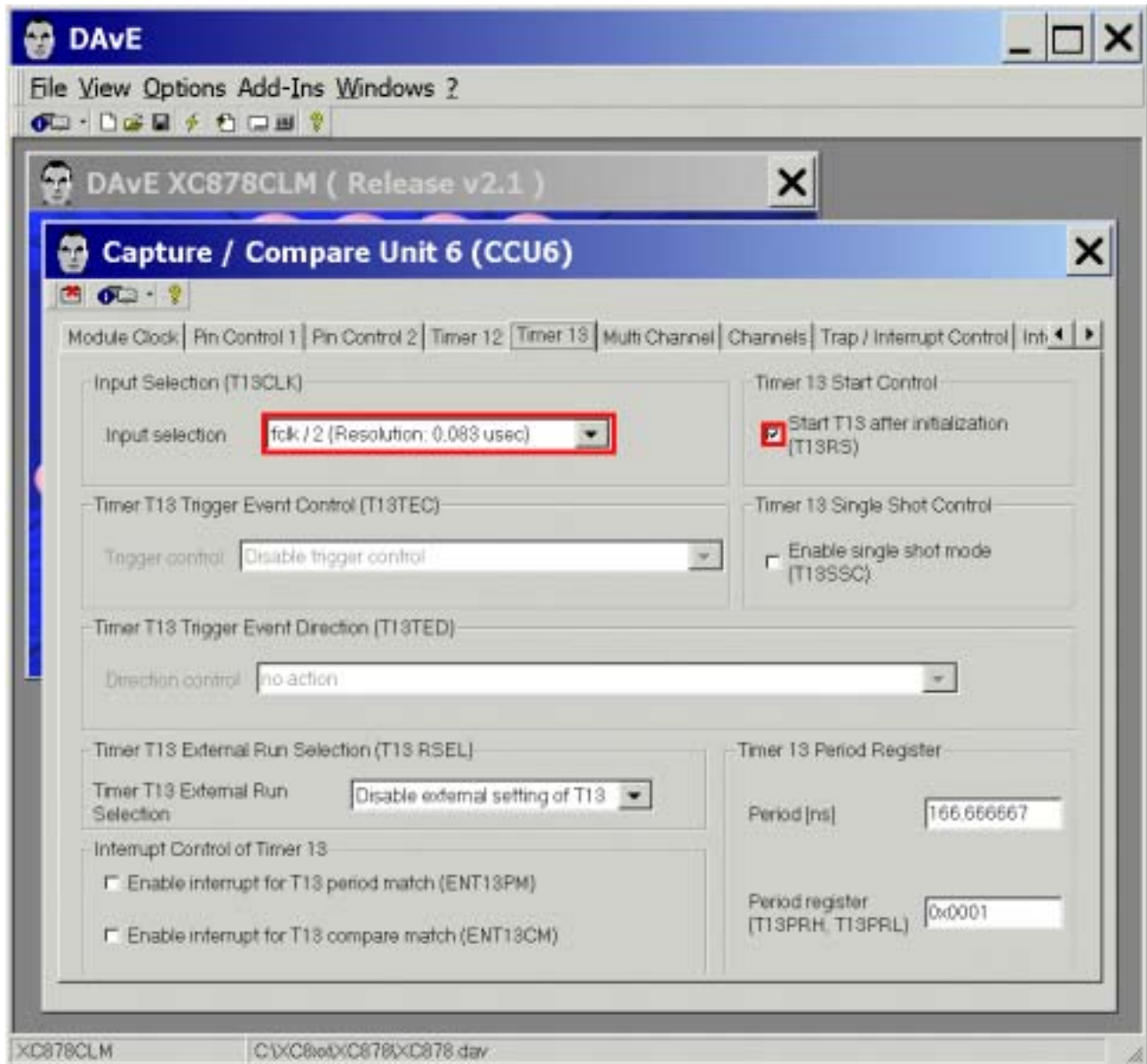
Timer 12 Resolution:

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

Timer 13: "note frequency":

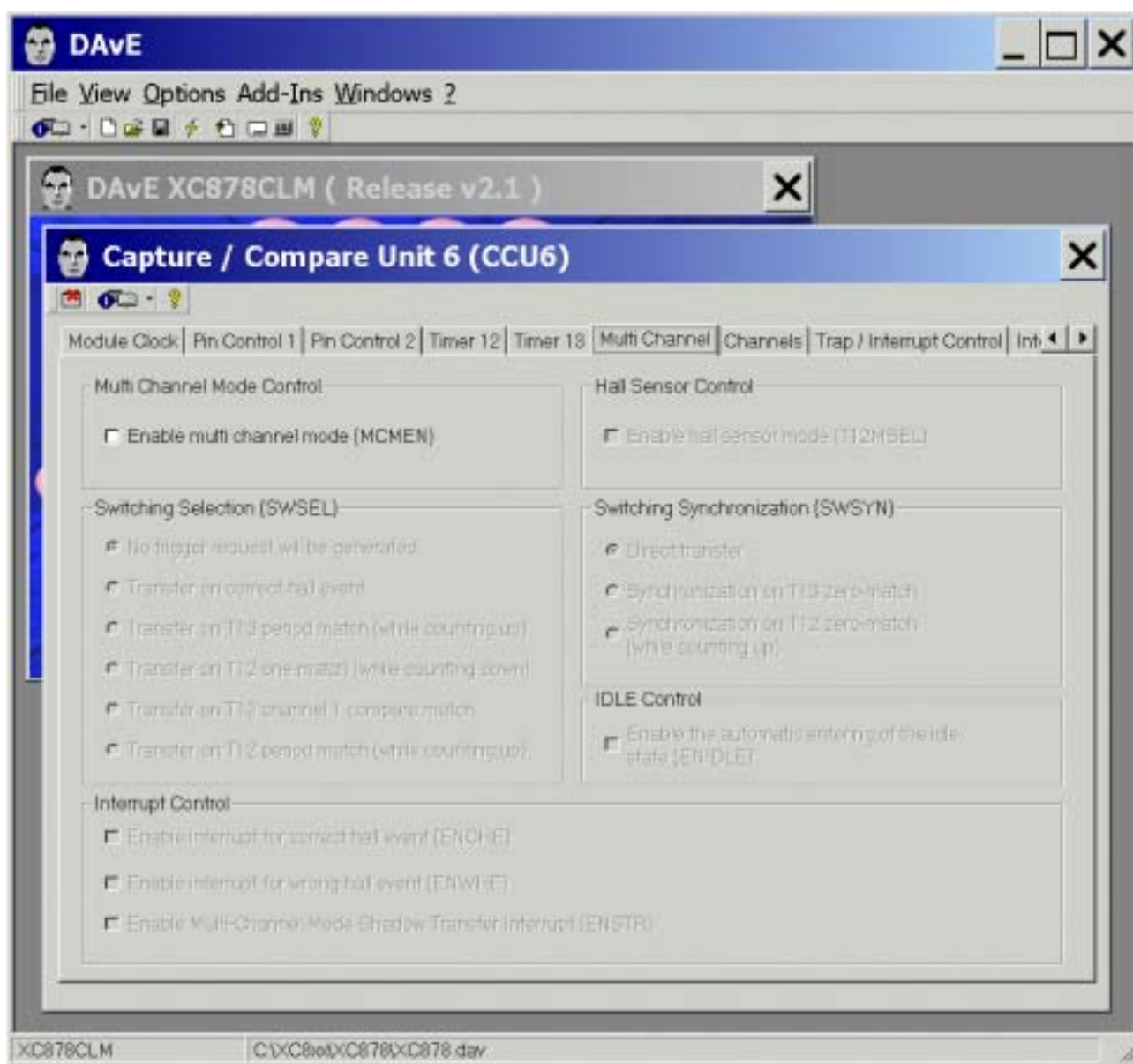
CCU6: Timer T13: Input Selection: Input selection **select** fclk/2 (Resolution: 83,333 ns)

CCU6: Timer T13: Timer 13 Start Control: **click** ✓ Start T13 after initialization (T13RS)

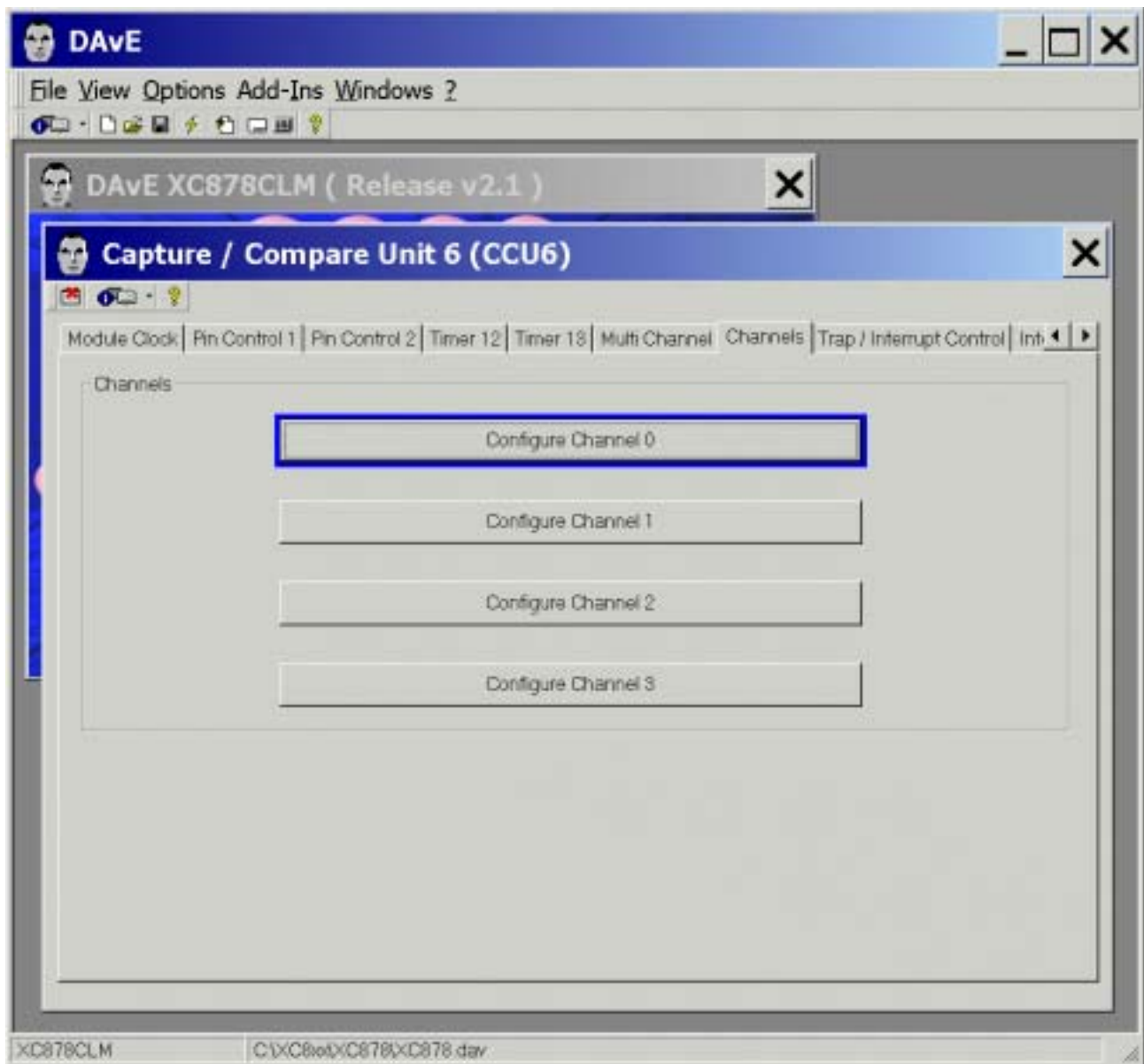


<<< !!! [click here to see more information about music](#) !!! >>>

CCU6: Multi Channel: (do nothing)



CCU6: Channels: **click** Configure Channel 0



Note:

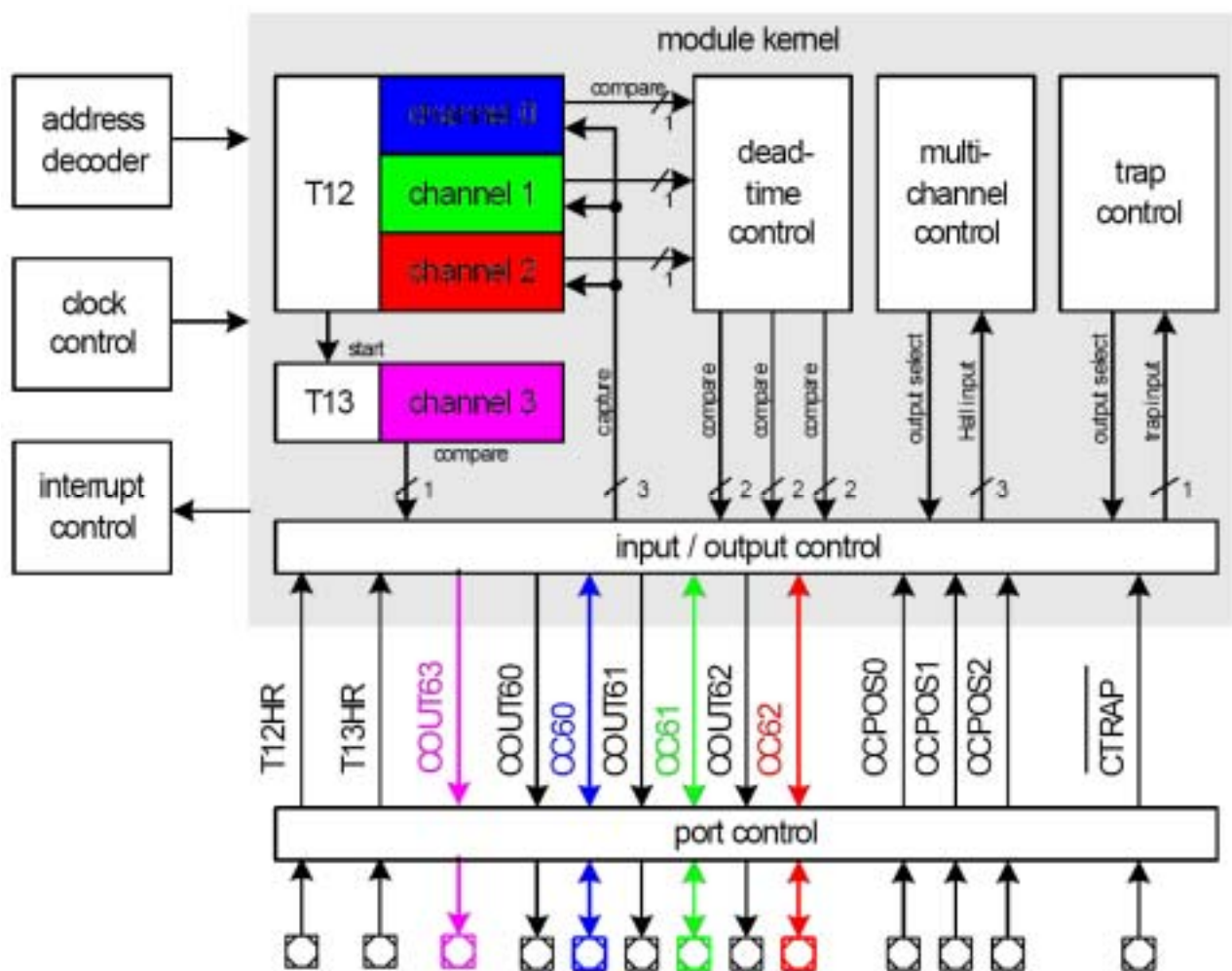
Port pins used by our PWM module (not available as GPIO pins):

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	COU63_0	Channel 3	50 (T13)



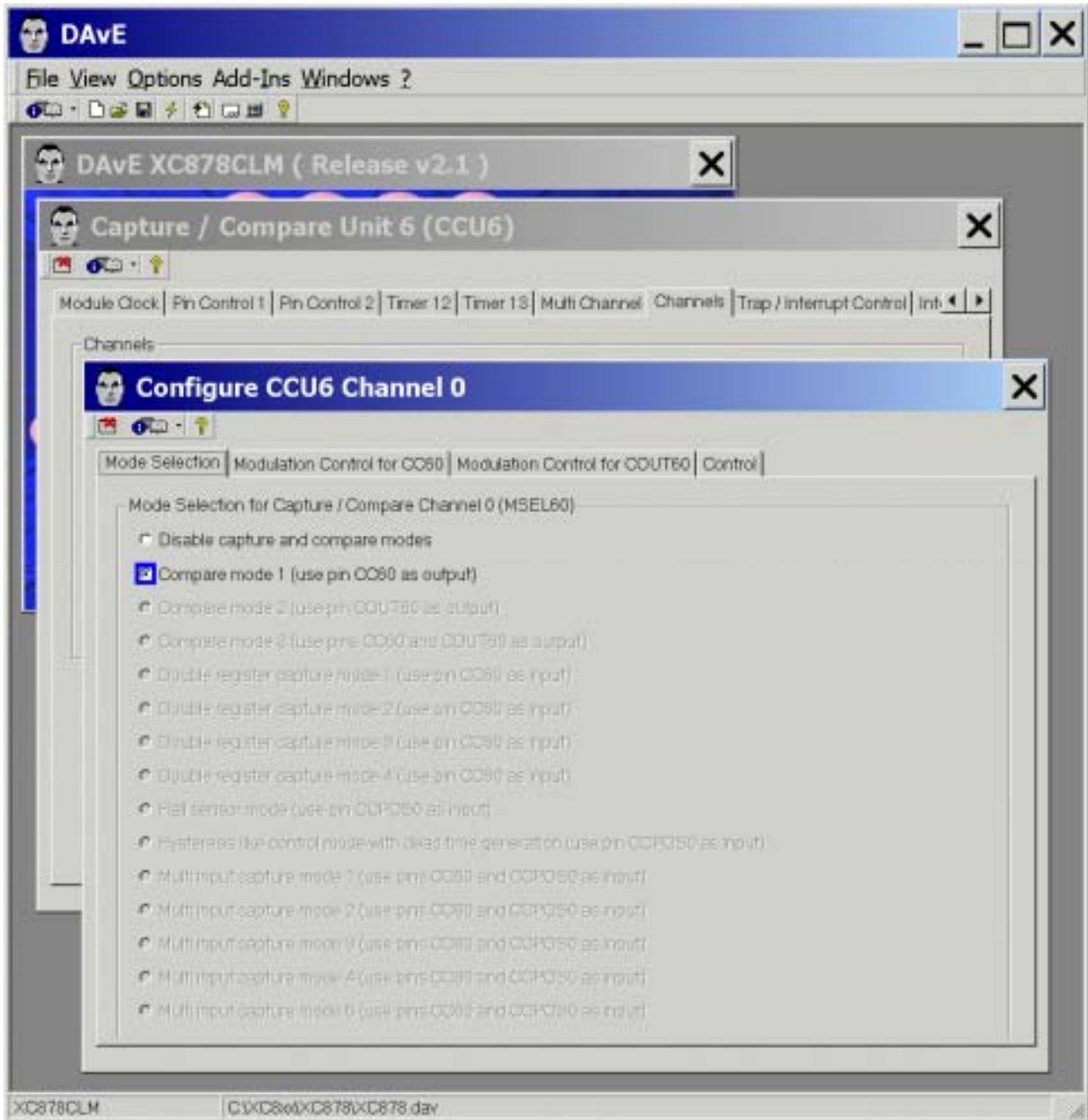


Note:

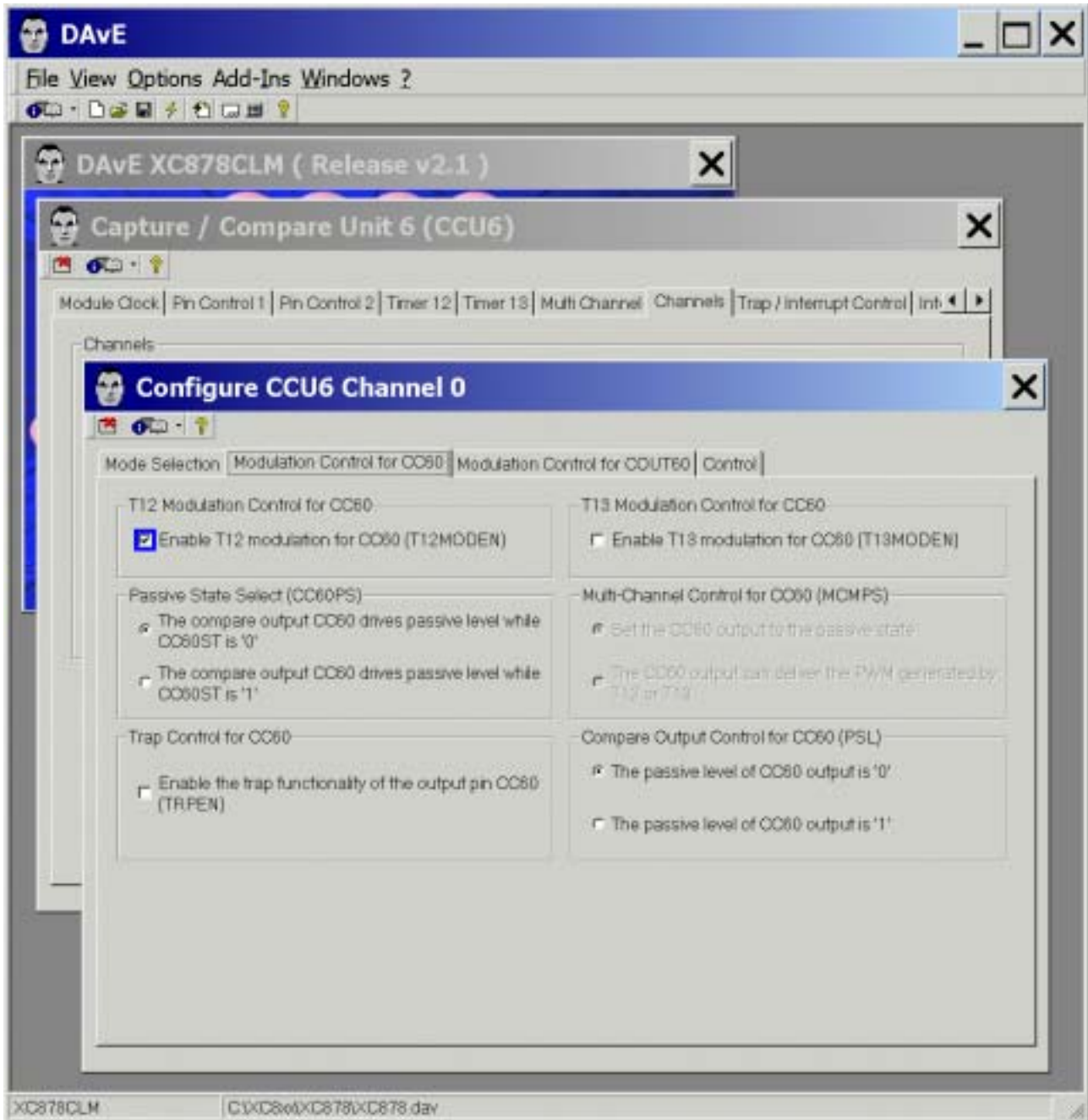


CCU6: Channels: **Configure Channel 0:**

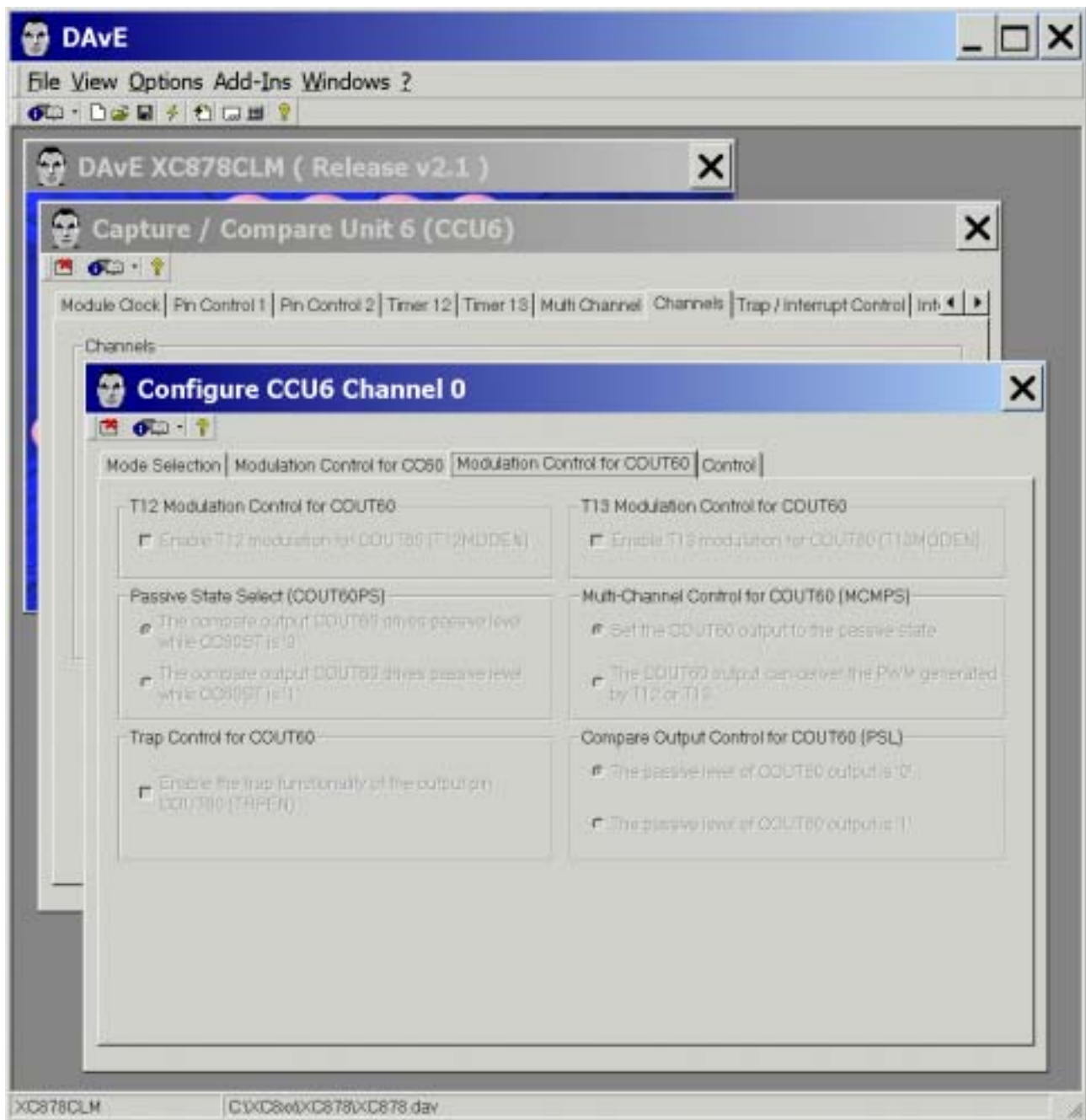
Mode Selection: **Mode Selection for Capture / Compare Channel 0:** click ☒ Compare mode 1



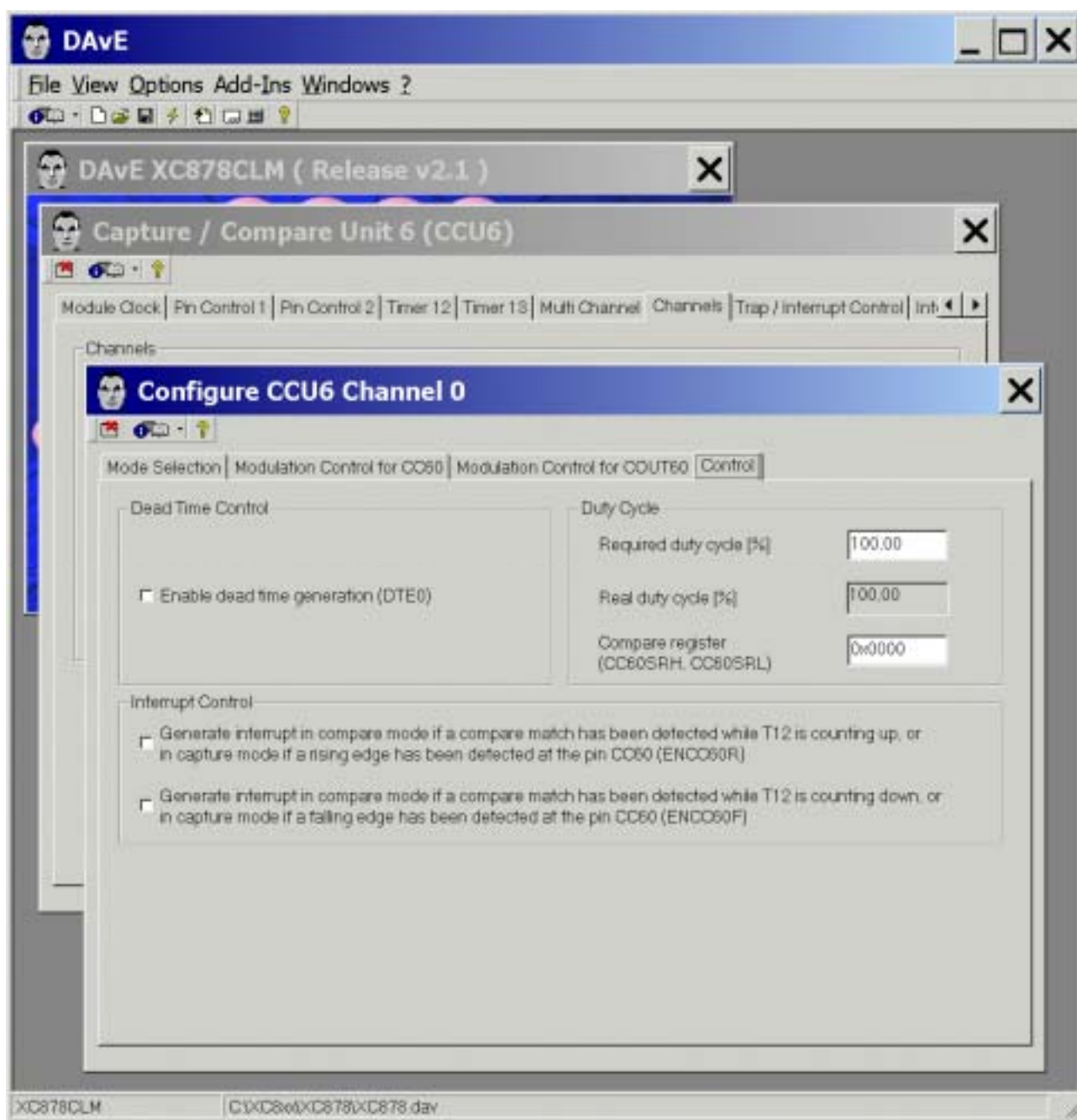
CCU6: Channels: **Configure Channel 0**: Modulation Control for CC60:
T12 Modulation Control for CC60: tick ☒ Enable T12 modulation for CC60



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

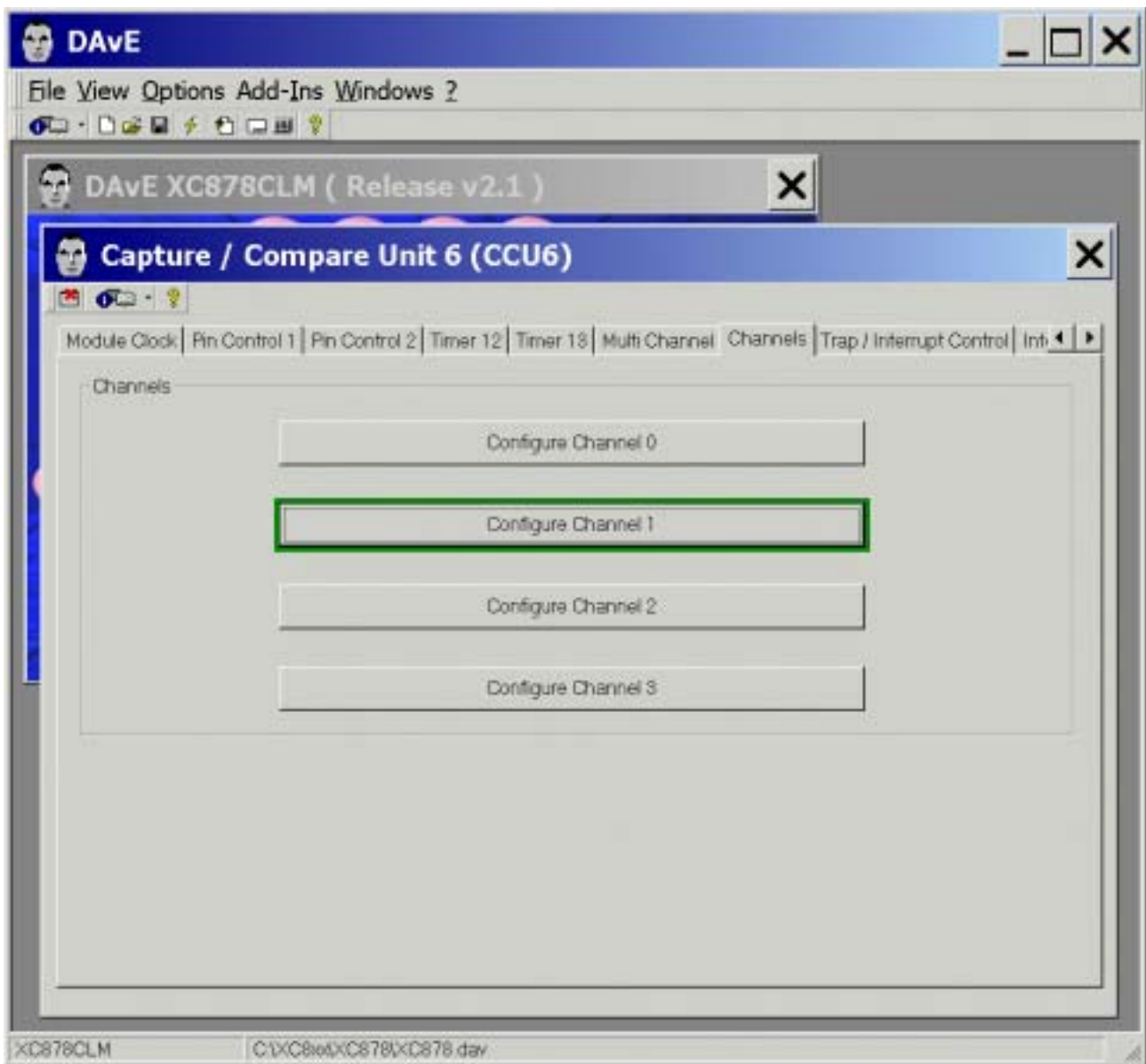


CCU6: Channels: Configure Channel 0: Control: (do nothing)



Exit this dialog now by clicking  the close button.

CCU6: Channels: **click** Configure Channel 1



Note:

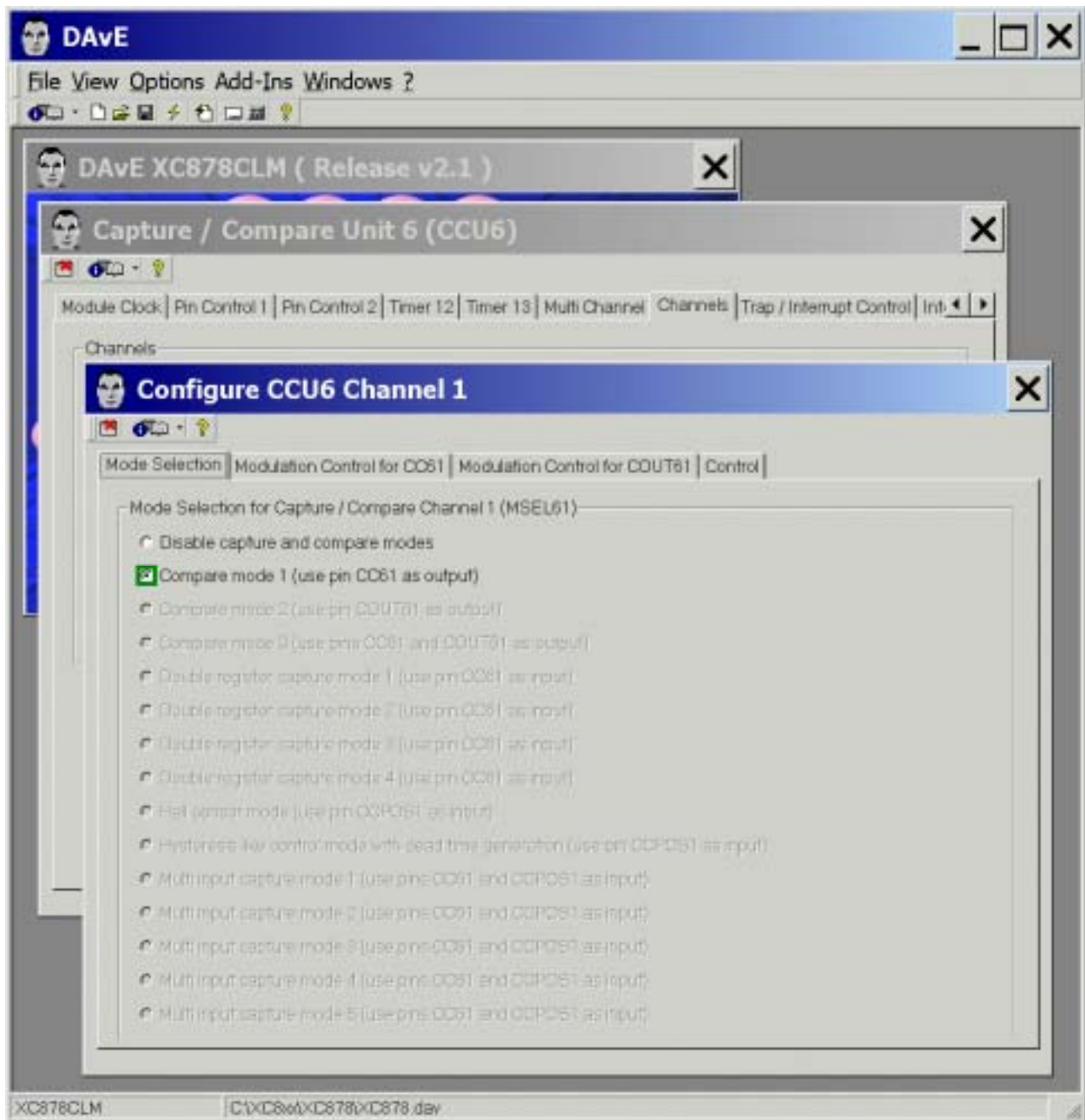
Port pins used by our PWM module (not available as GPIO pins):

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	COU63_0	Channel 3	50 (T13)

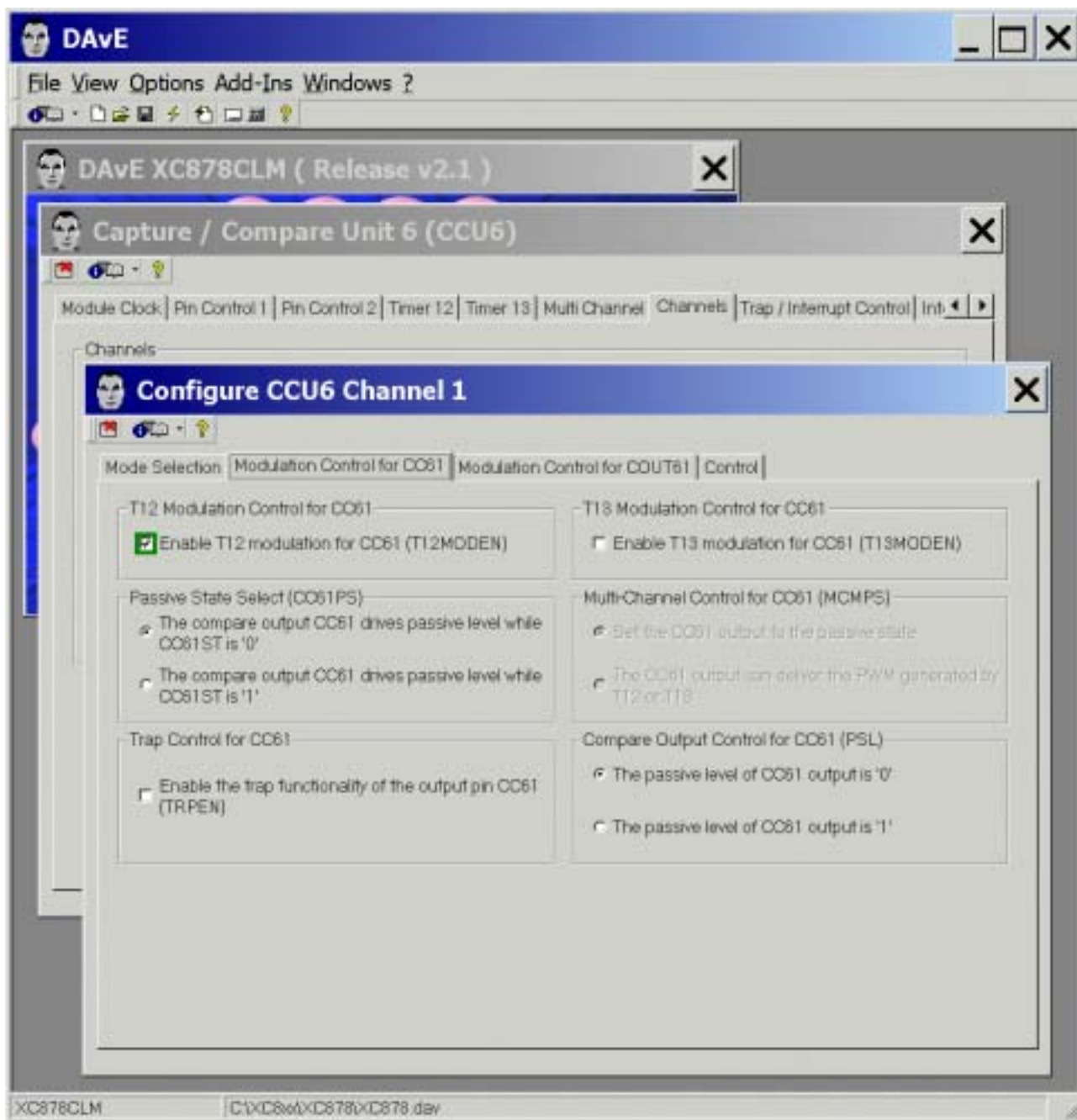


CCU6: Channels: **Configure Channel 1:**

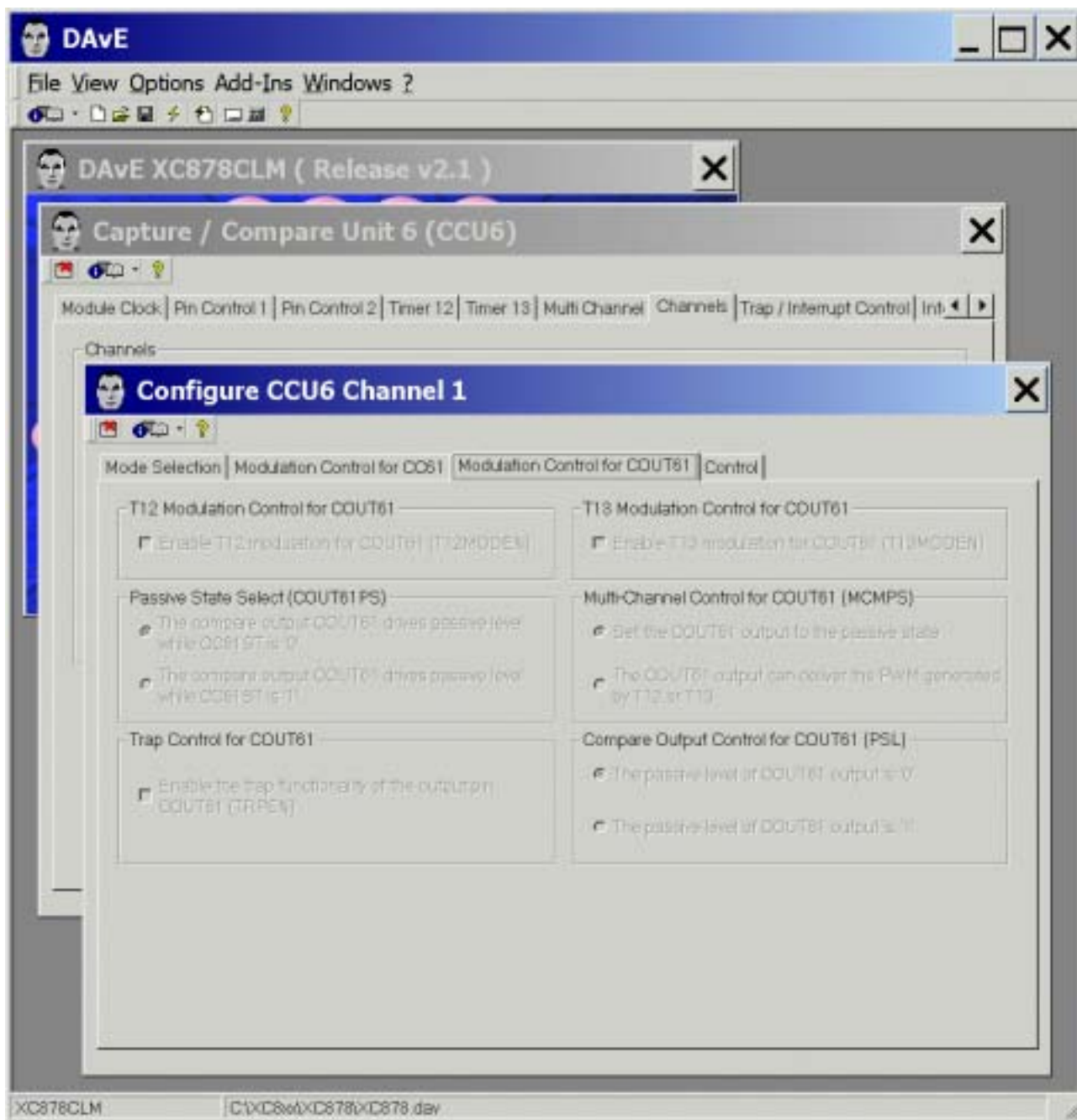
Mode Selection: **Mode Selection for Capture / Compare Channel 1:** click ☒ Compare mode 1



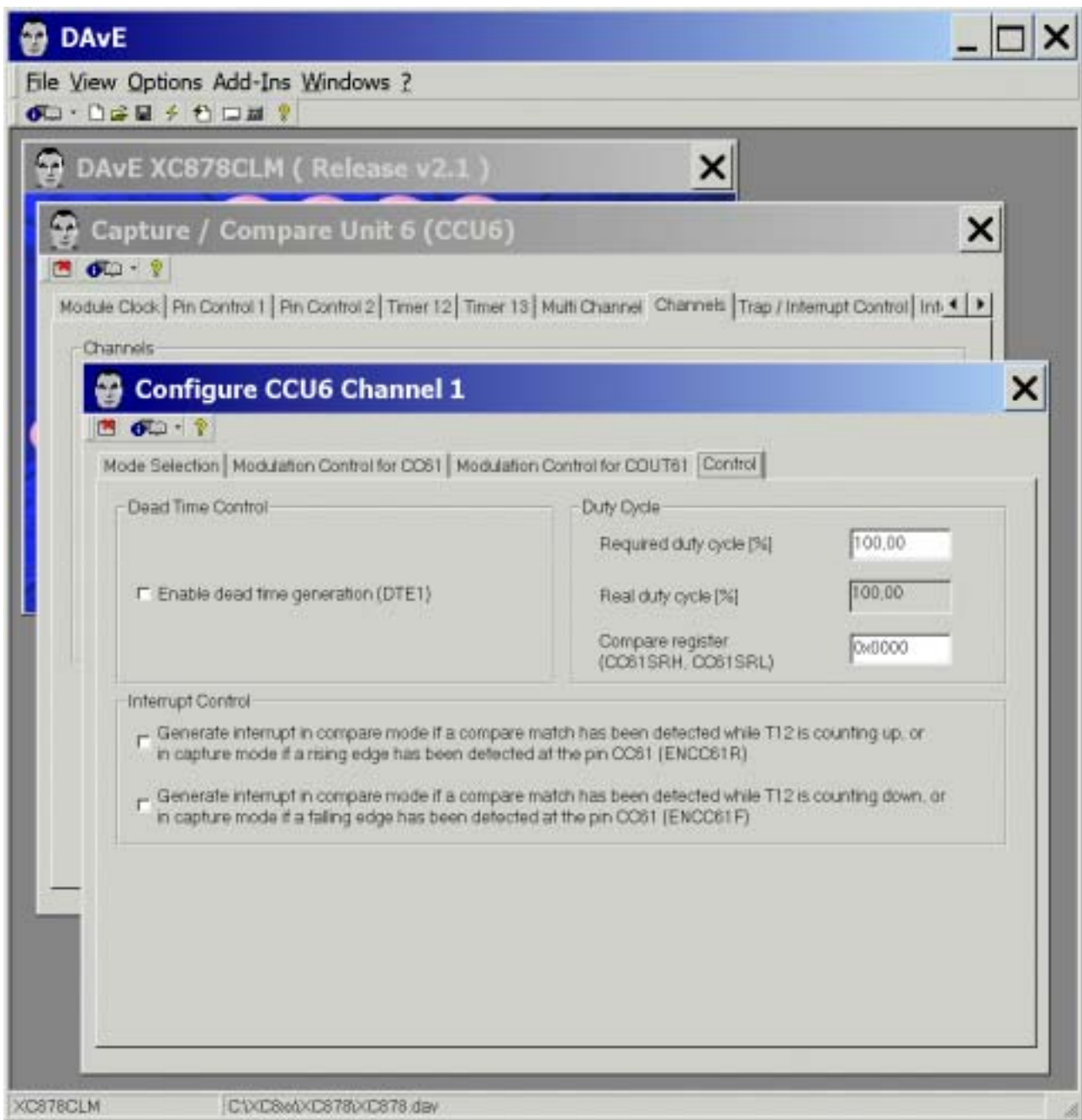
CCU6: Channels: **Configure Channel 1**: Modulation Control for CC61:
T12 Modulation Control for CC61: tick ☒ Enable T12 modulation for CC61



CCU6: Channels: **Configure Channel 1**:
Modulation Control for CCUT61: (do nothing)

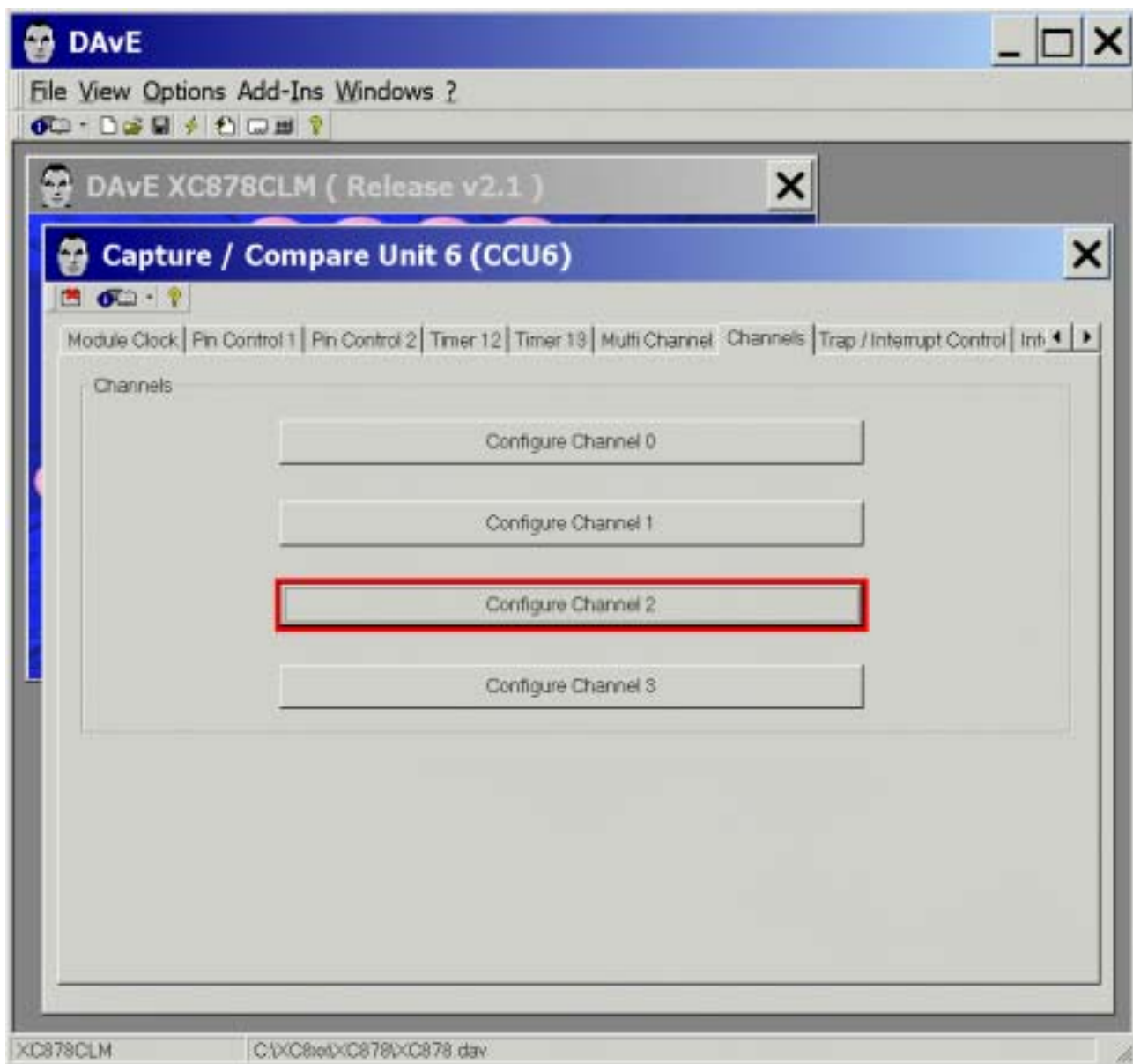


CCU6: Channels: **Configure Channel 1**: **Control**: (do nothing)



Exit this dialog now by clicking  the close button.

CCU6: Channels: **click** Configure Channel 2



Note:

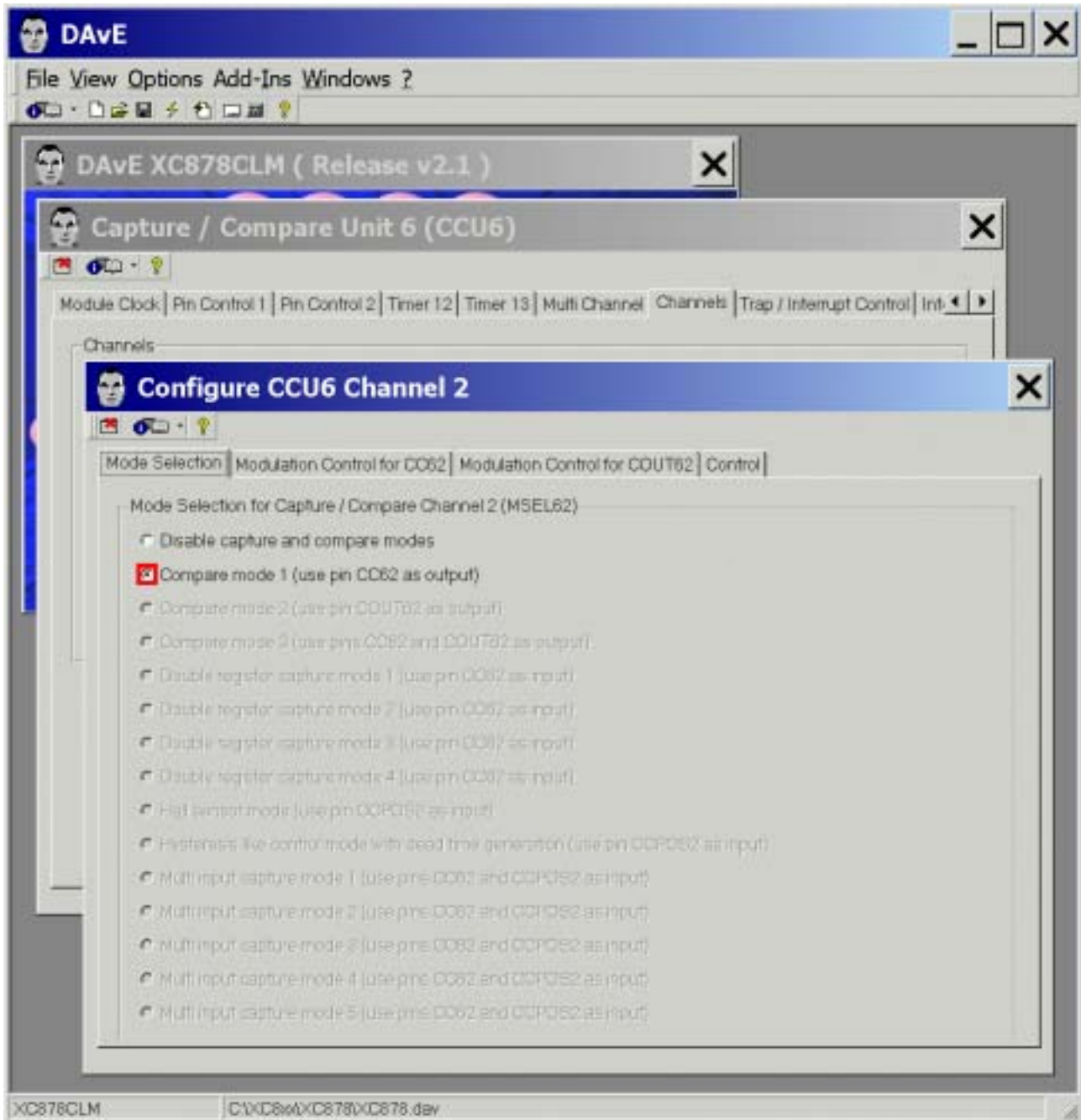
Port pins used by our PWM module (not available as GPIO pins):

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	COU63_0	Channel 3	50 (T13)



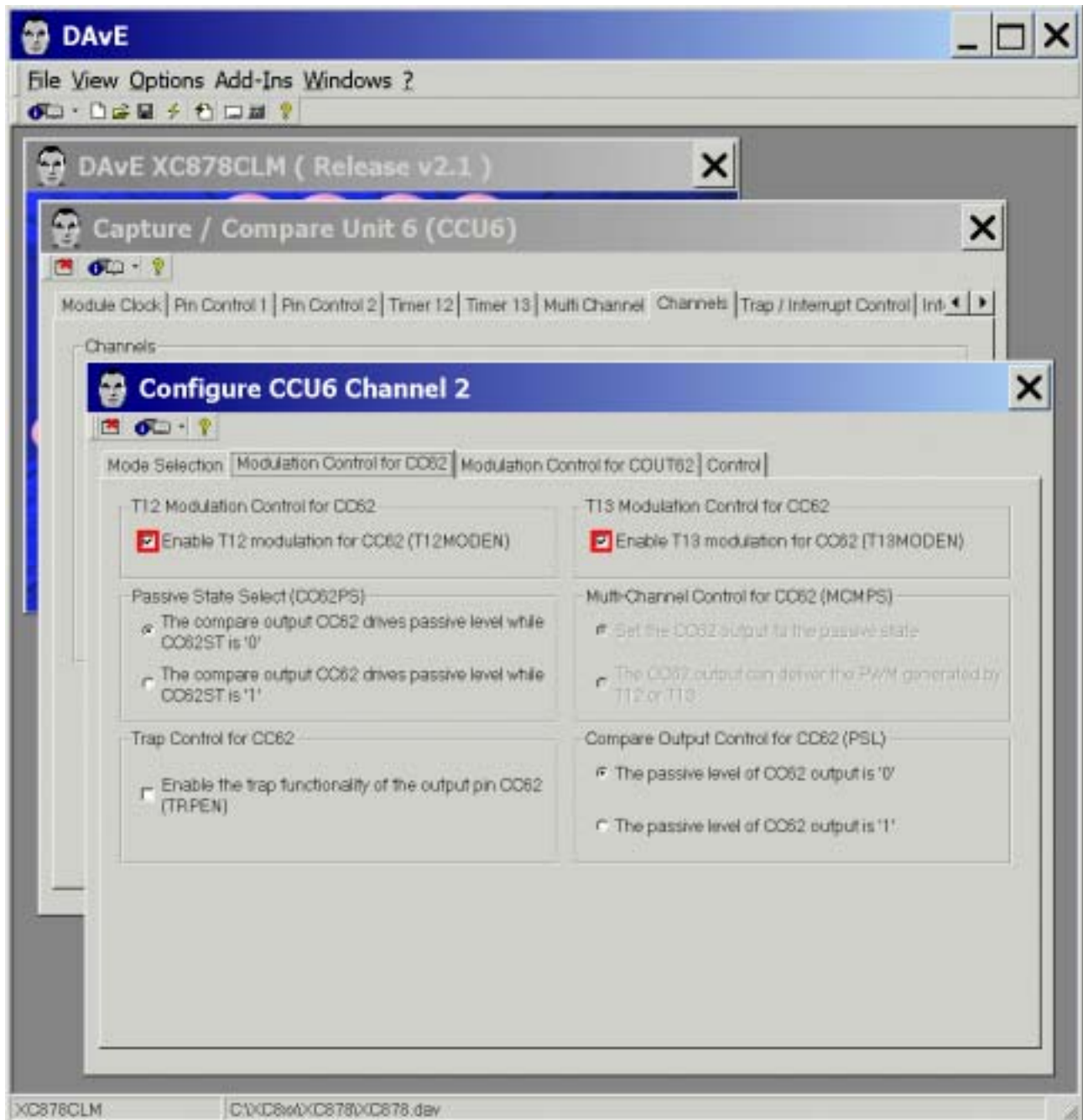
CCU6: Channels: **Configure Channel 2:**

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

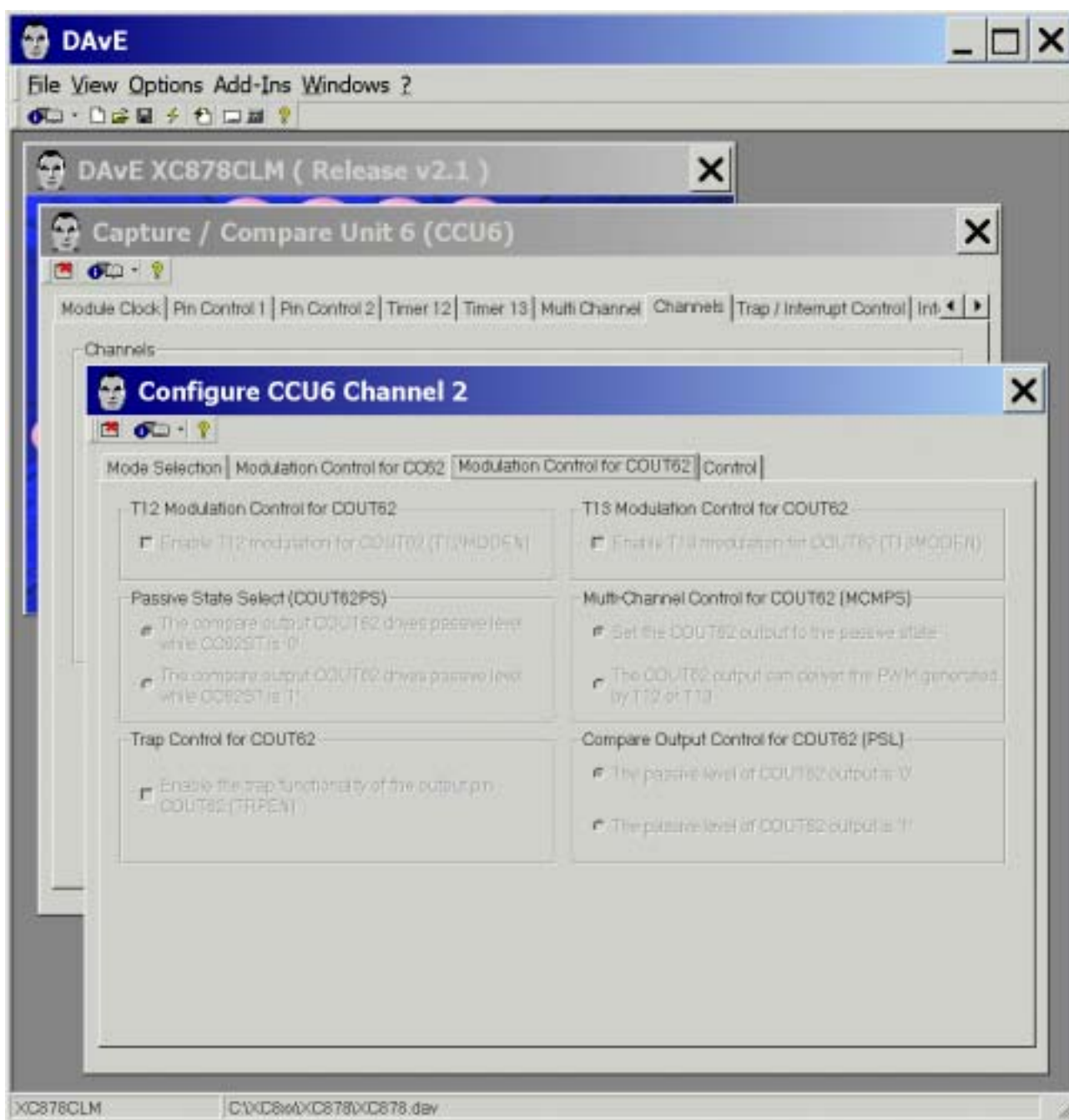


CCU6: Channels: Configure Channel 2: Modulation Control for CC62:
T12 Modulation Control for CC62: tick ☒ Enable T12 modulation for CC62

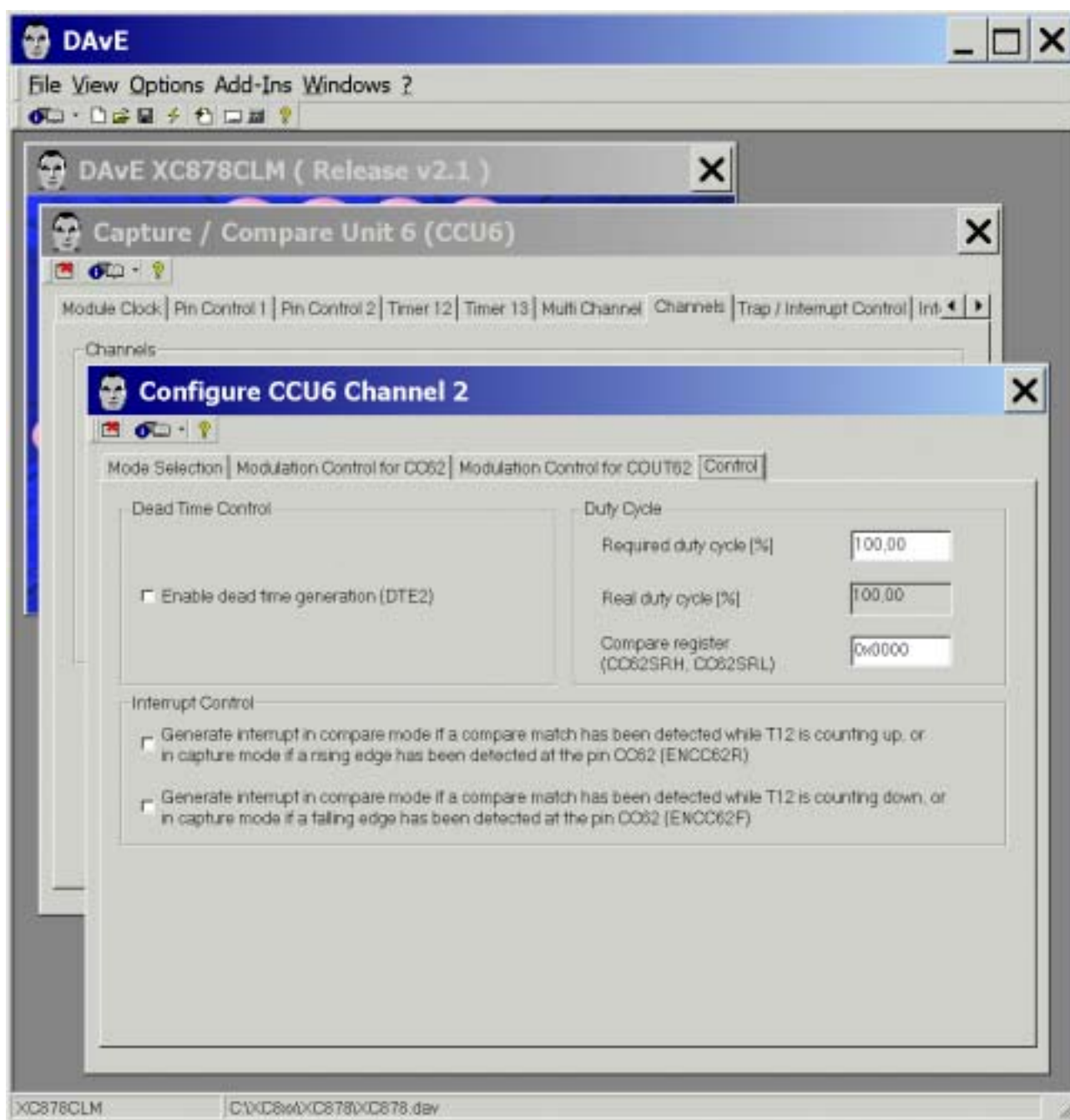
CCU6: Channels: Configure Channel 2: Modulation Control for CC62:
T13 Modulation Control for CC62: tick ☒ Enable T13 modulation for CC62



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

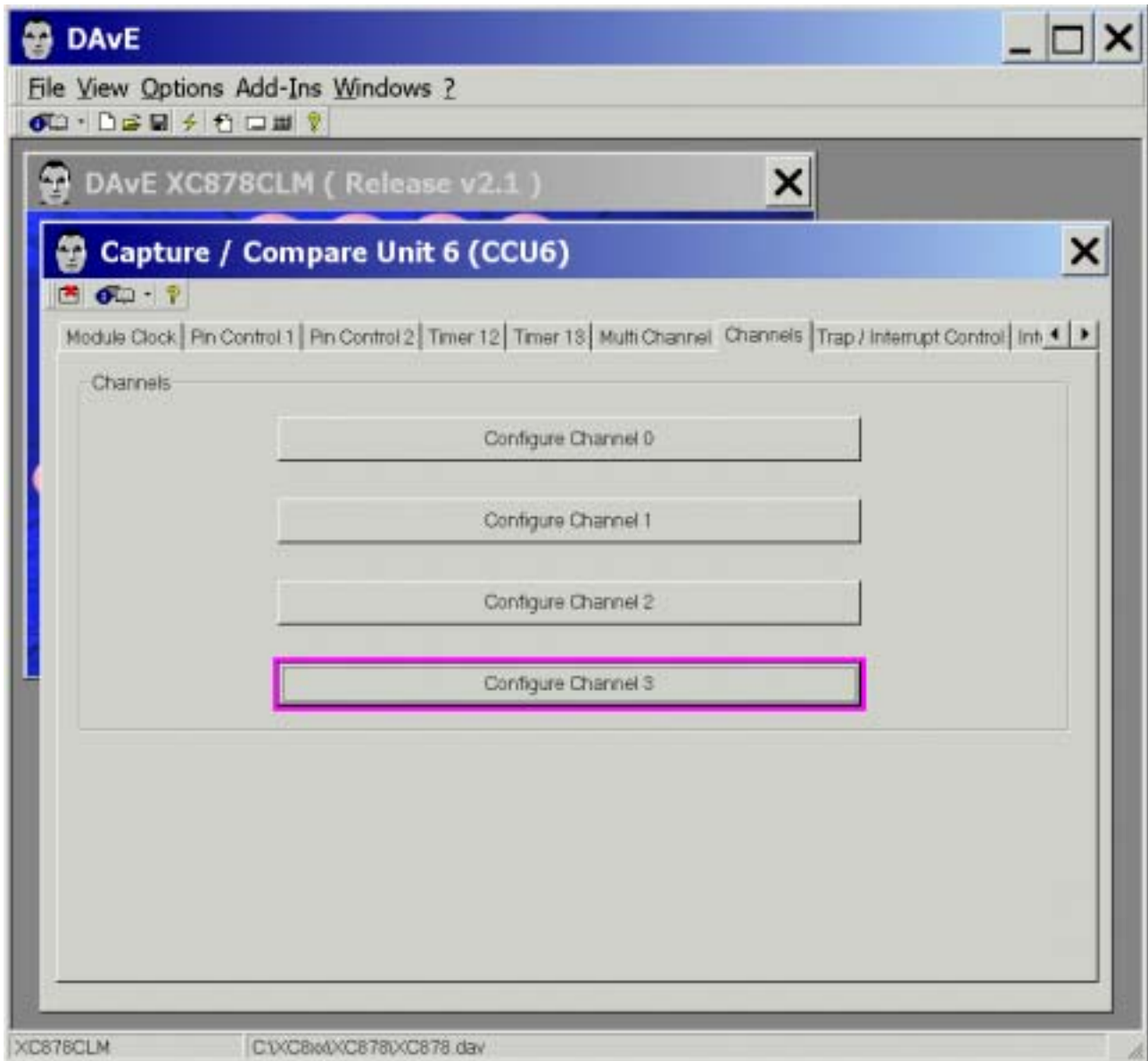


CCU6: Channels: **Configure Channel 2**: **Control**: (do nothing)



Exit this dialog now by clicking  the close button.

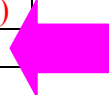
CCU6: Channels: **click** Configure Channel 3



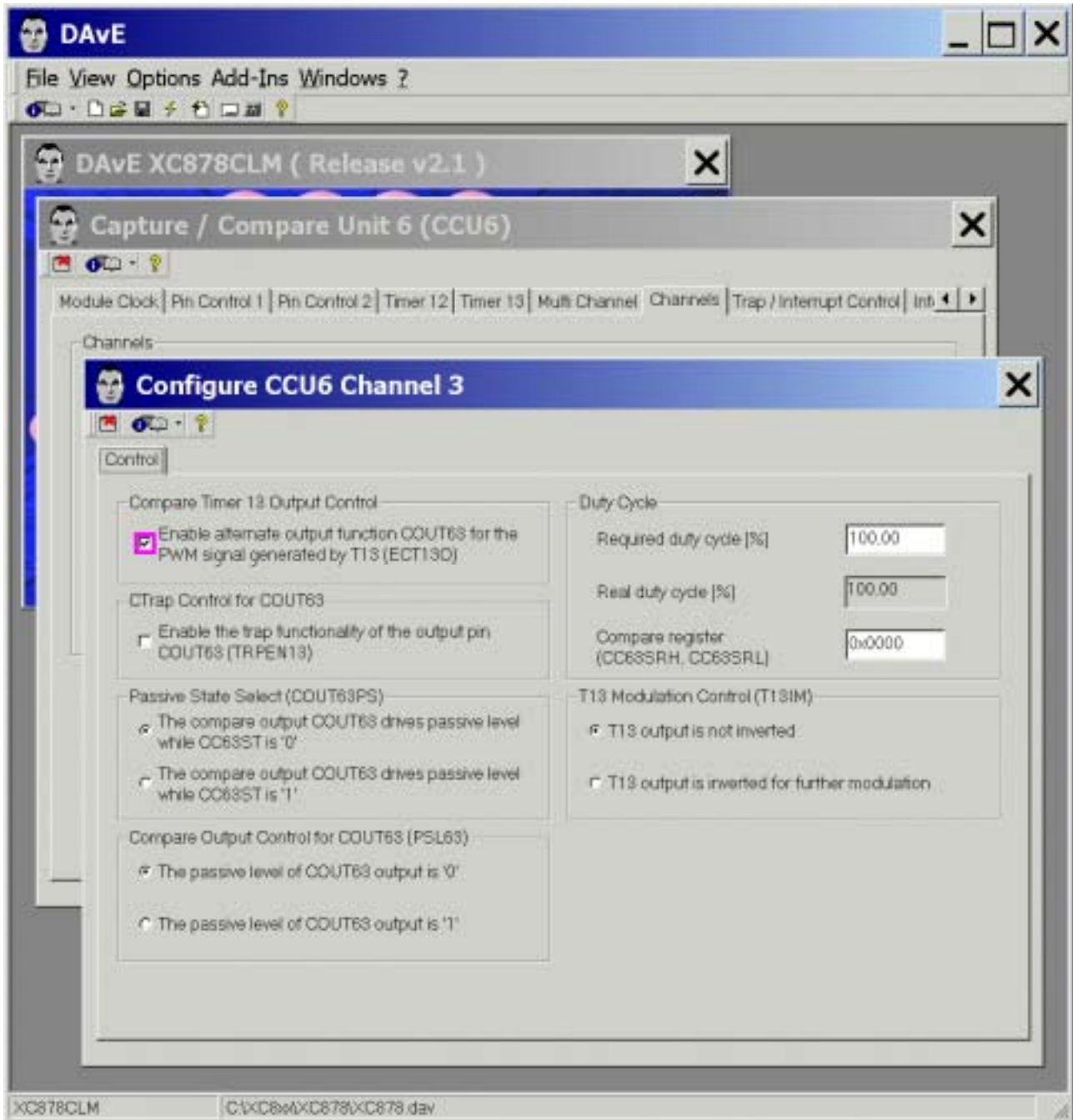
Remember:

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	COU63_0	Channel 3	50 (T13)

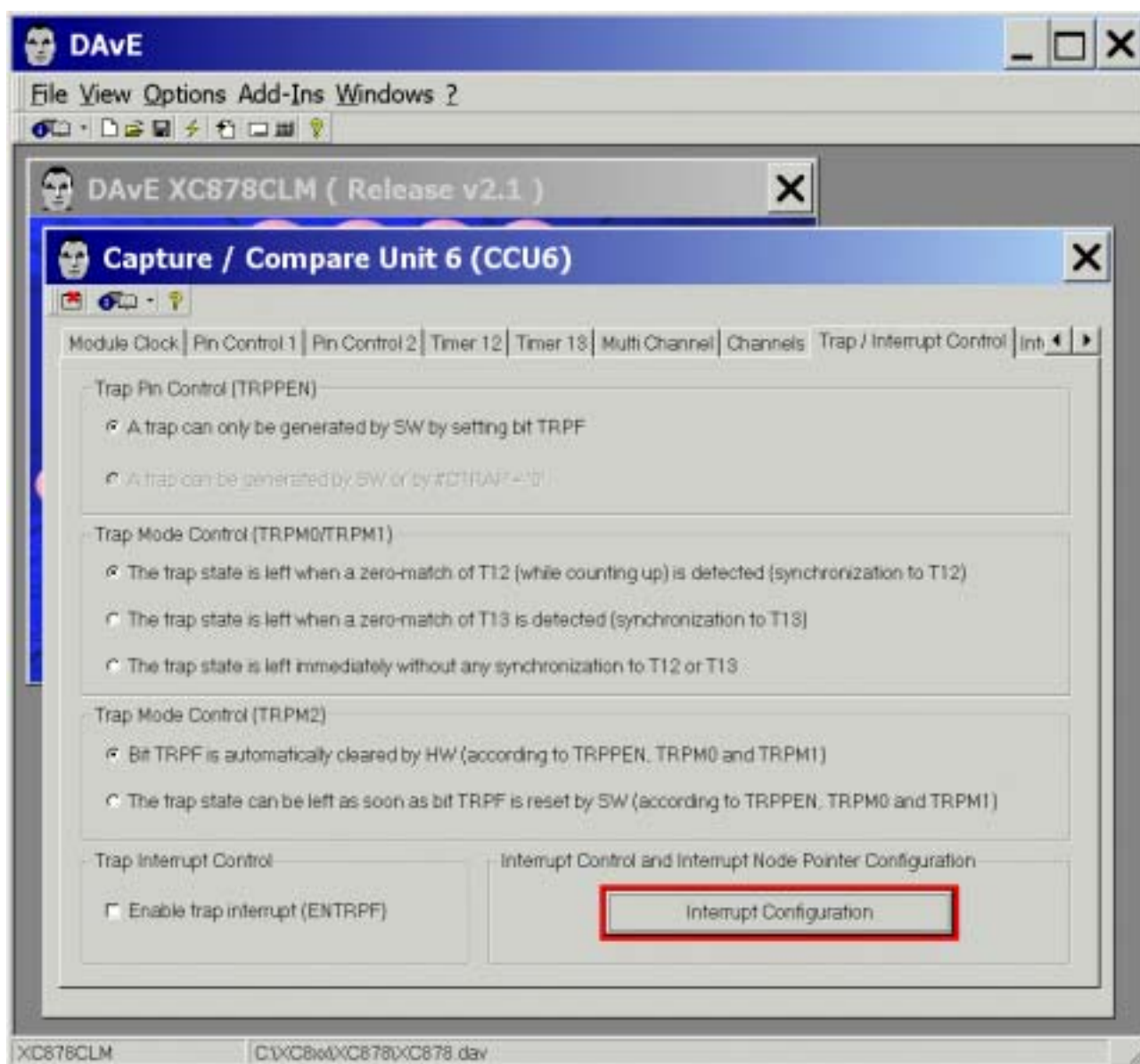


CCU6: Channels: Configure Channel 3: Control: Compare Timer 13 Output Control:
tick ☒ Enable alternate output function COUT63 for the PWM signal generated by T13



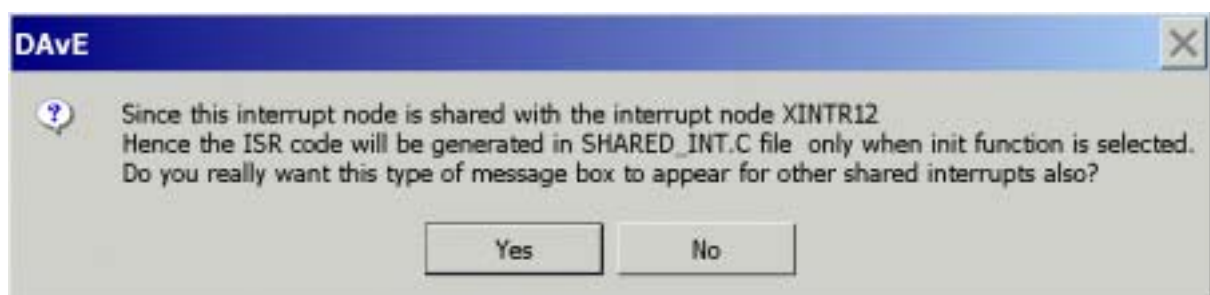
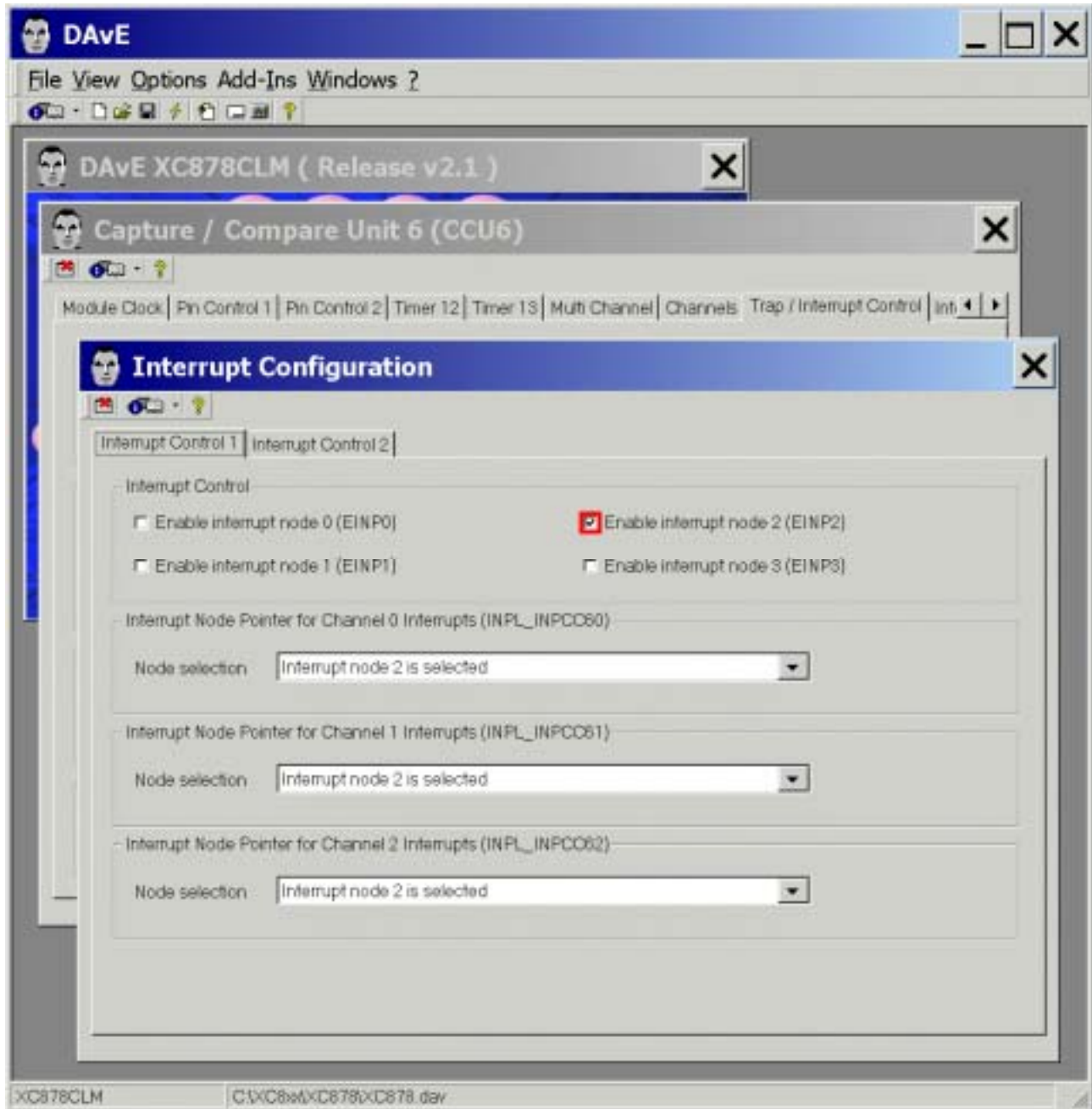
Exit this dialog now by clicking  the close button.

CCU6: Trap / Interrupt Control: [click](#) Interrupt Configuration



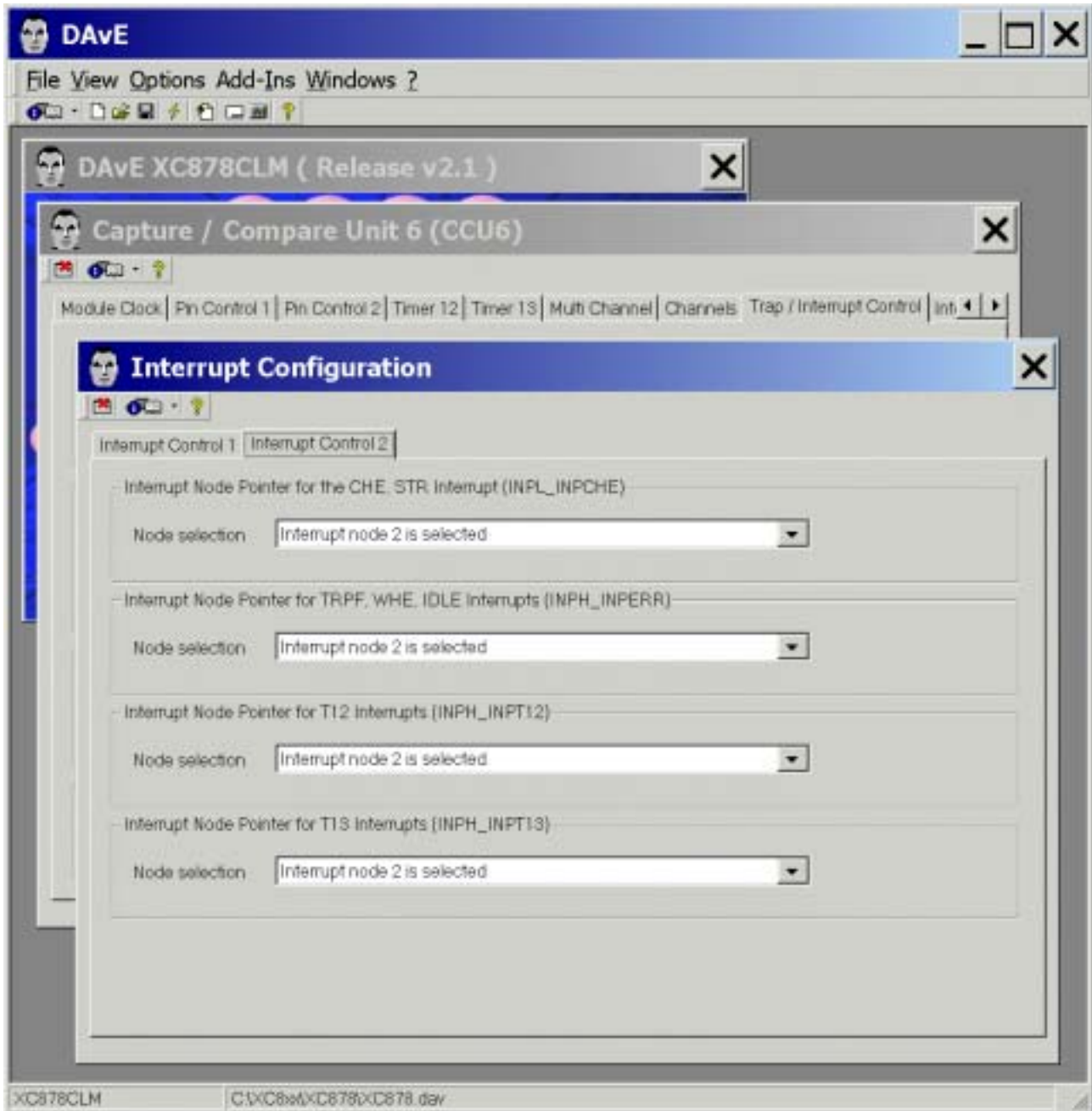
CCU6: Trap / Interrupt Control: Interrupt Configuration:

Interrupt Control 1: Interrupt Control: tick ☒ Enable interrupt node 2



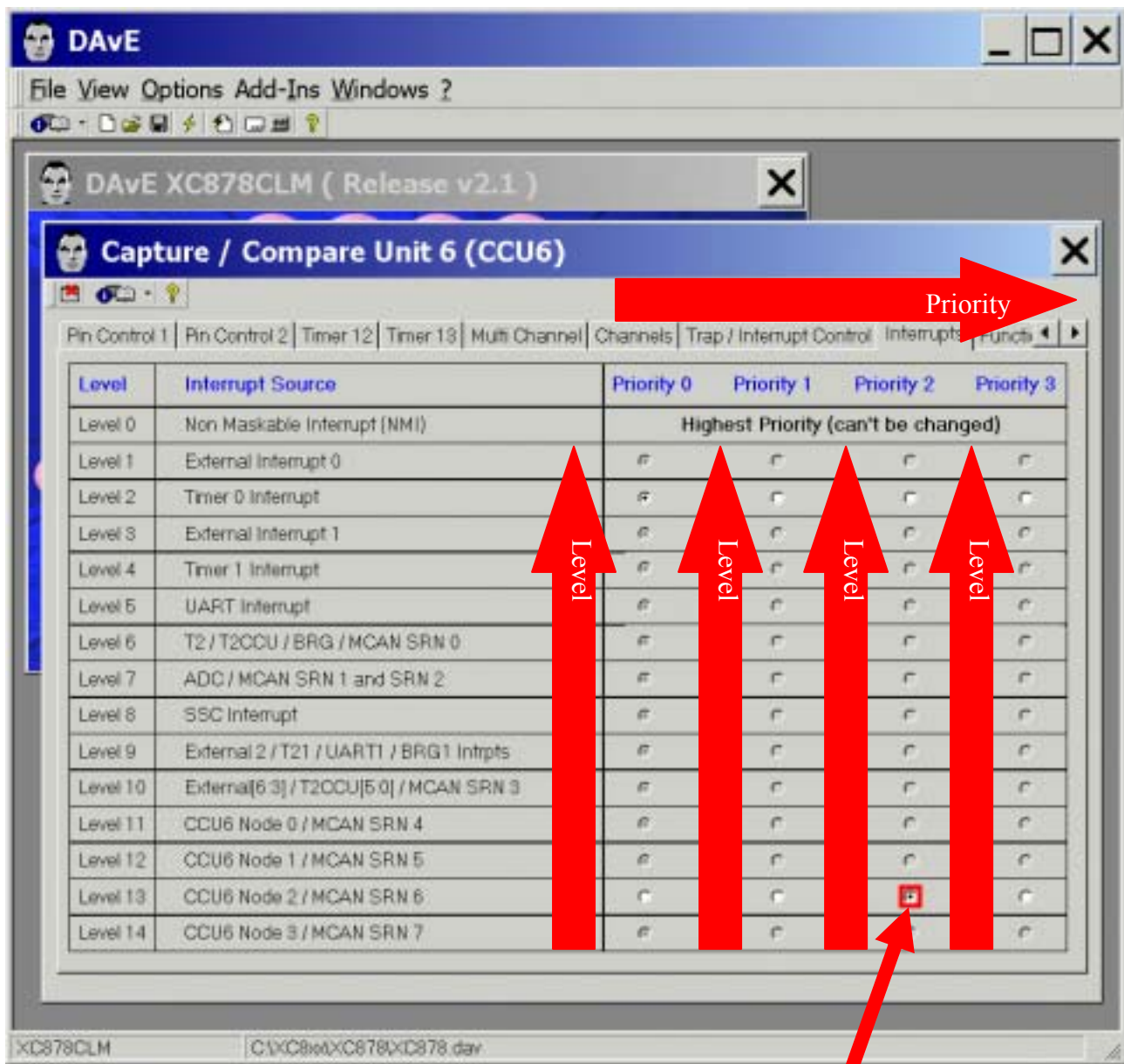
Click Yes

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



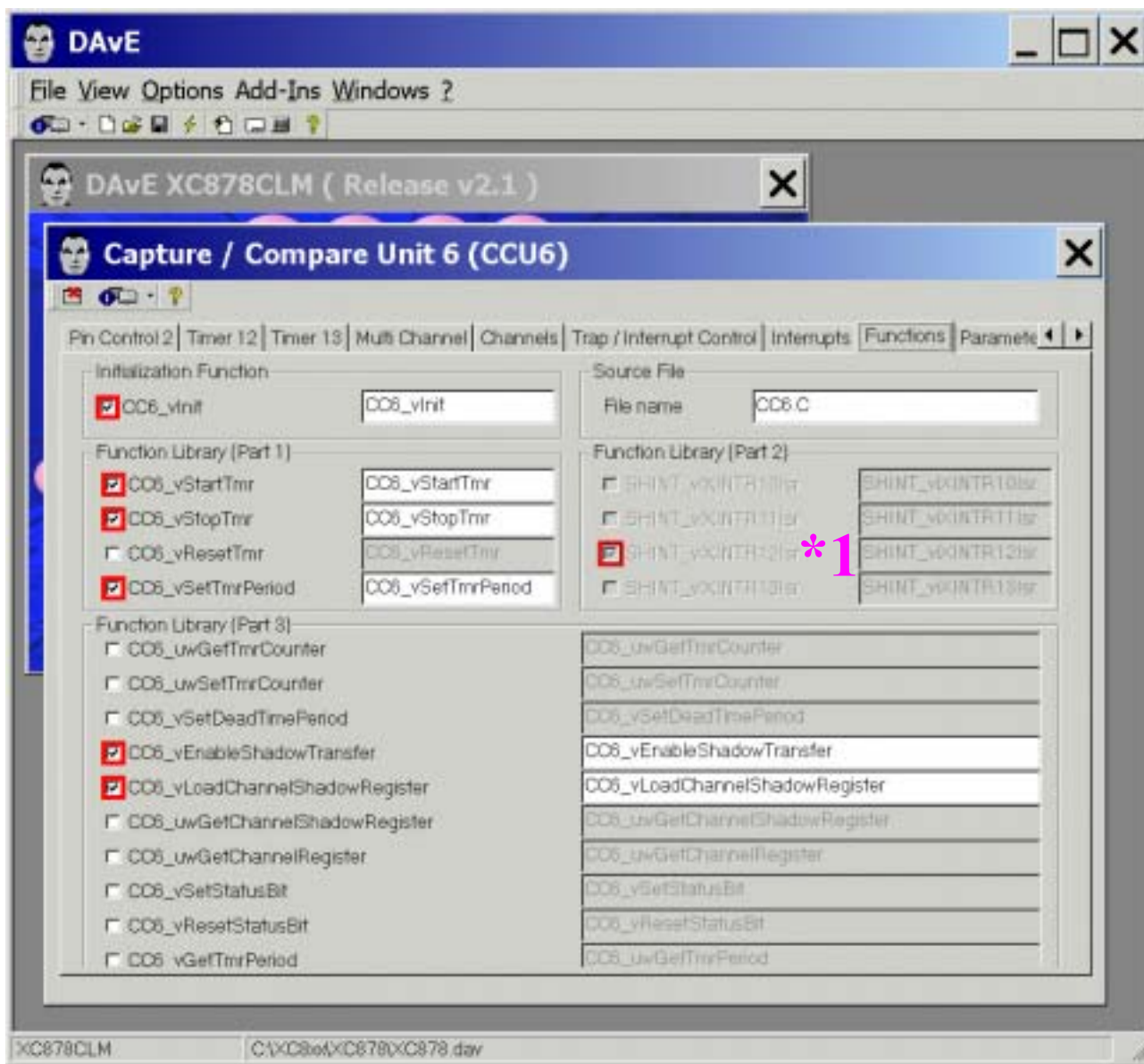
Exit this dialog now by clicking  the close button.

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



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



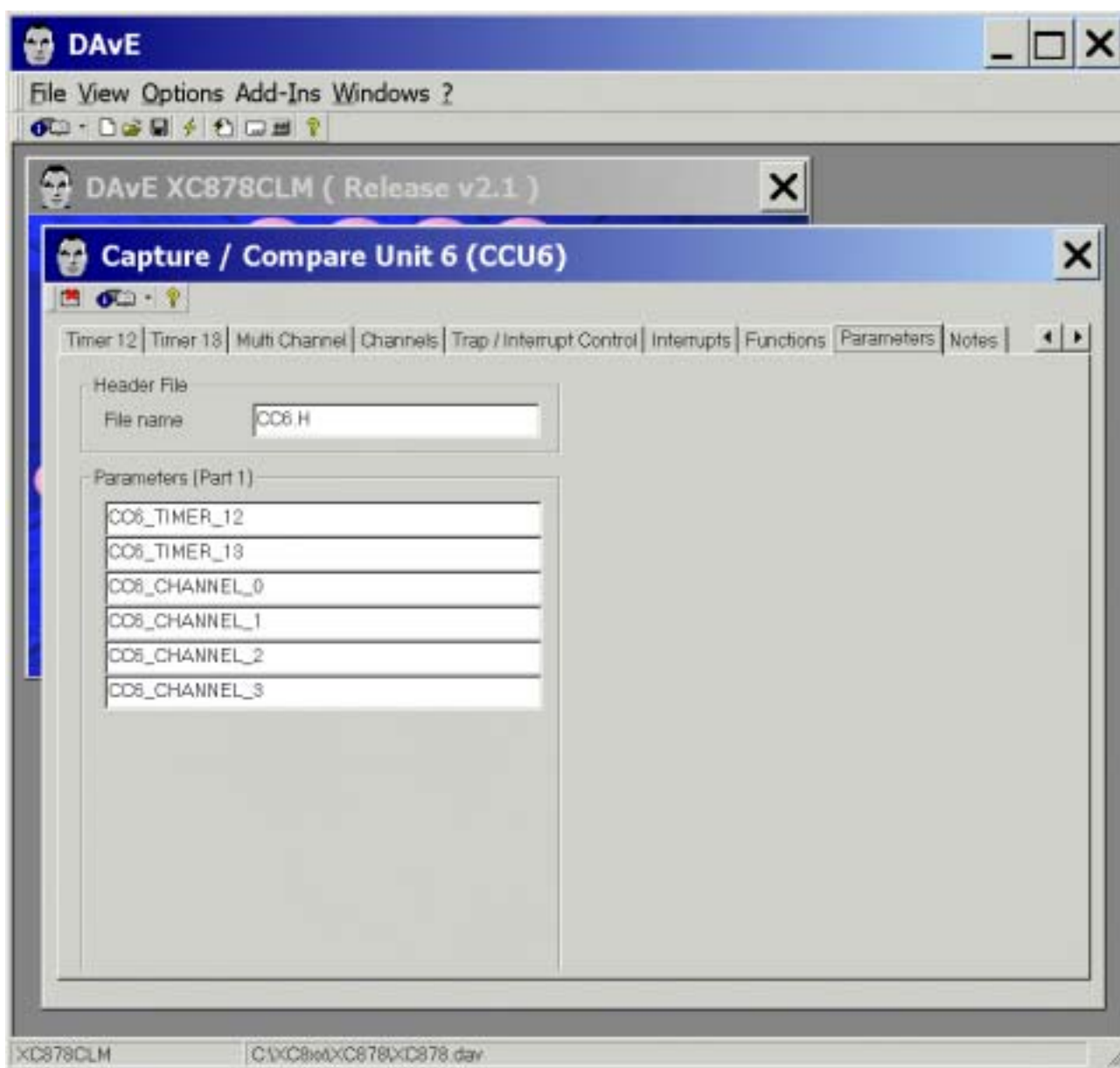
Note (*1):

The CCU6 ISR

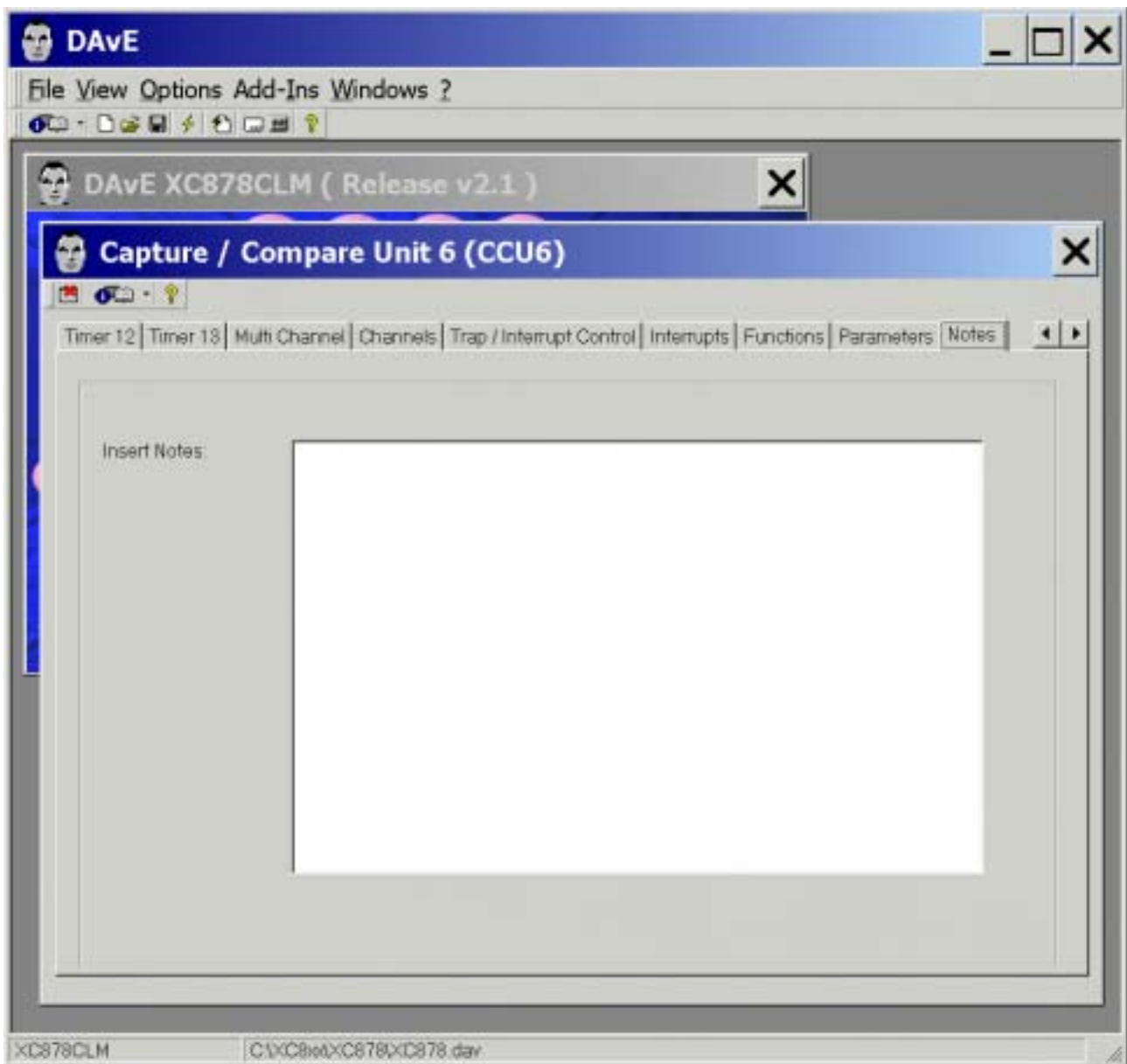
void SHINT_vXINTR12Isr(void) interrupt XINTR12INT {}
 will be generated in the SHARED_INT.C file.



CCU6: Parameters: (do nothing)




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



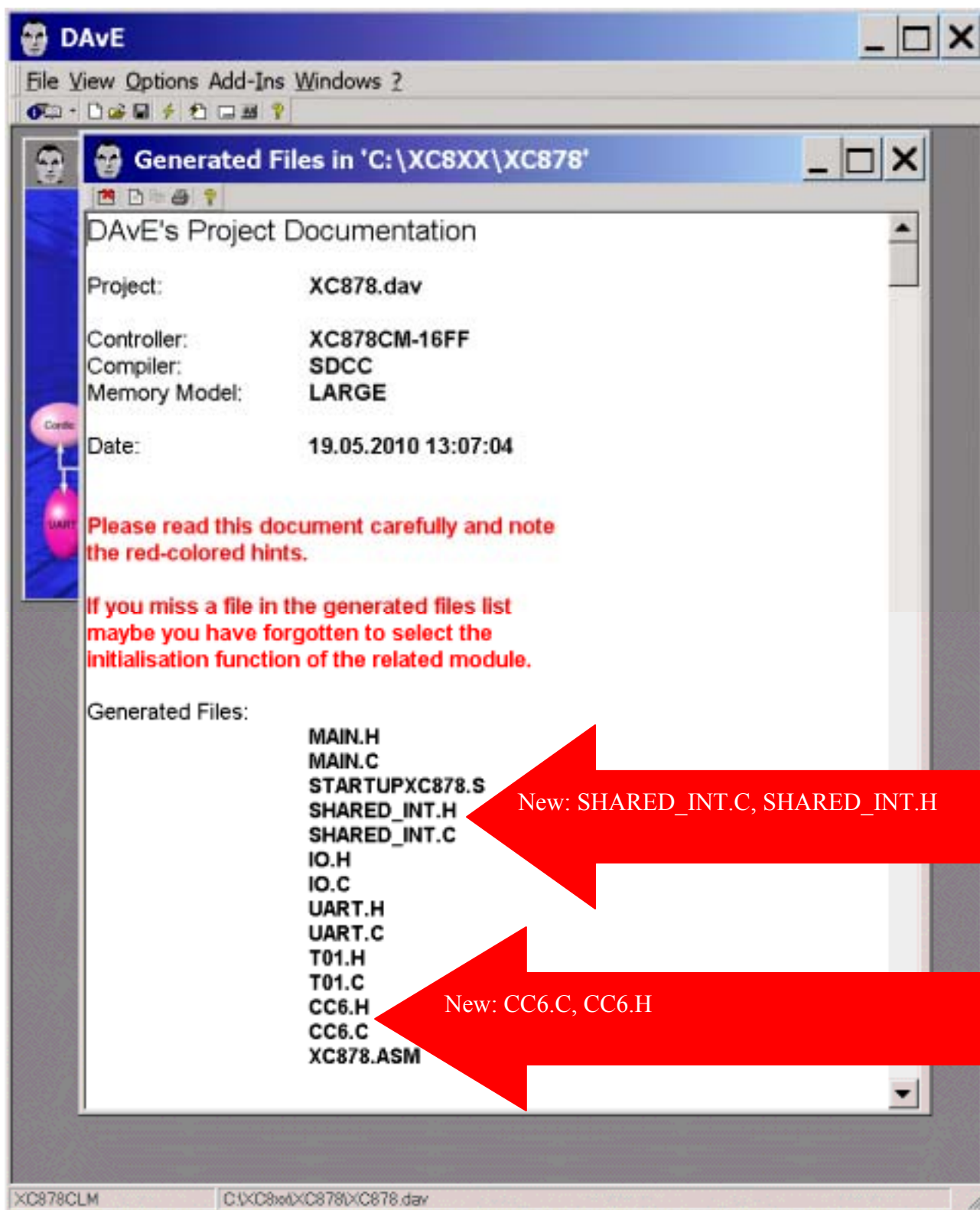
Exit this dialog now by clicking  the close button.

Generate Code:

File - Generate Code	or	click 
----------------------	----	--



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

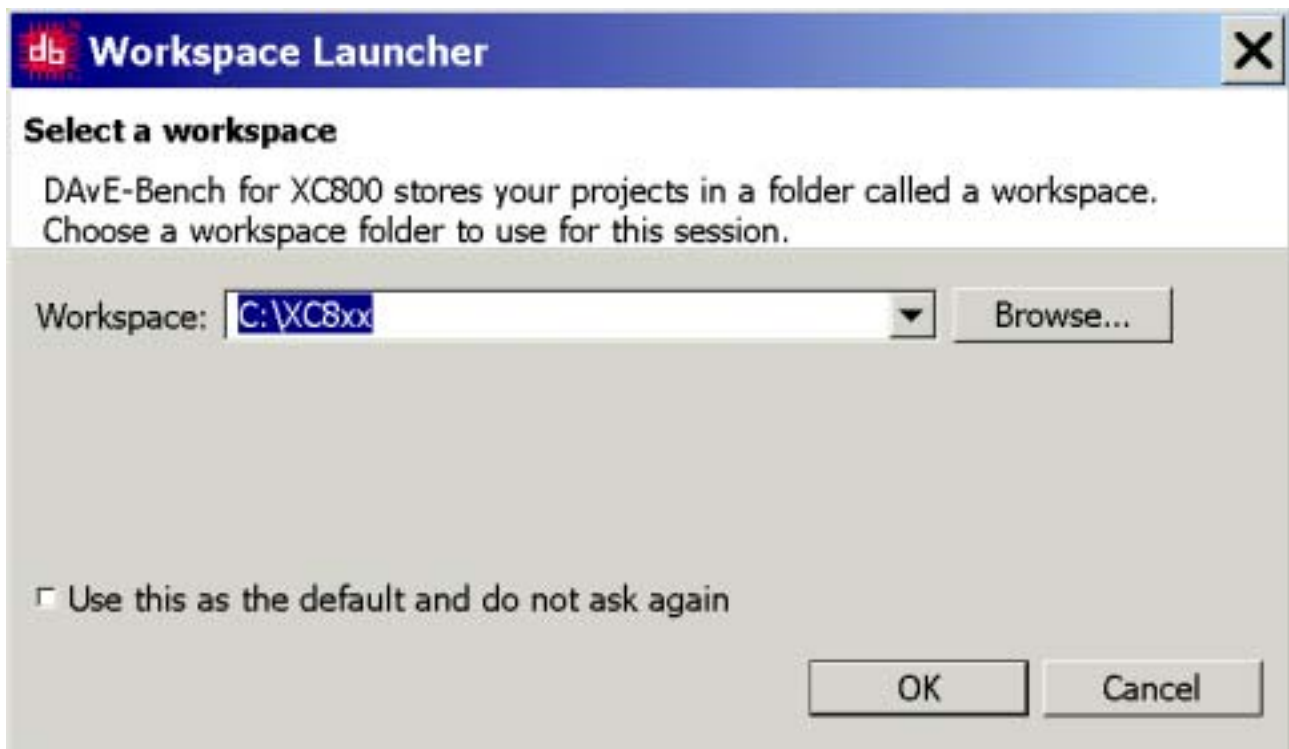


Close DAVe: **File – Exit** Save changes? **click Yes**

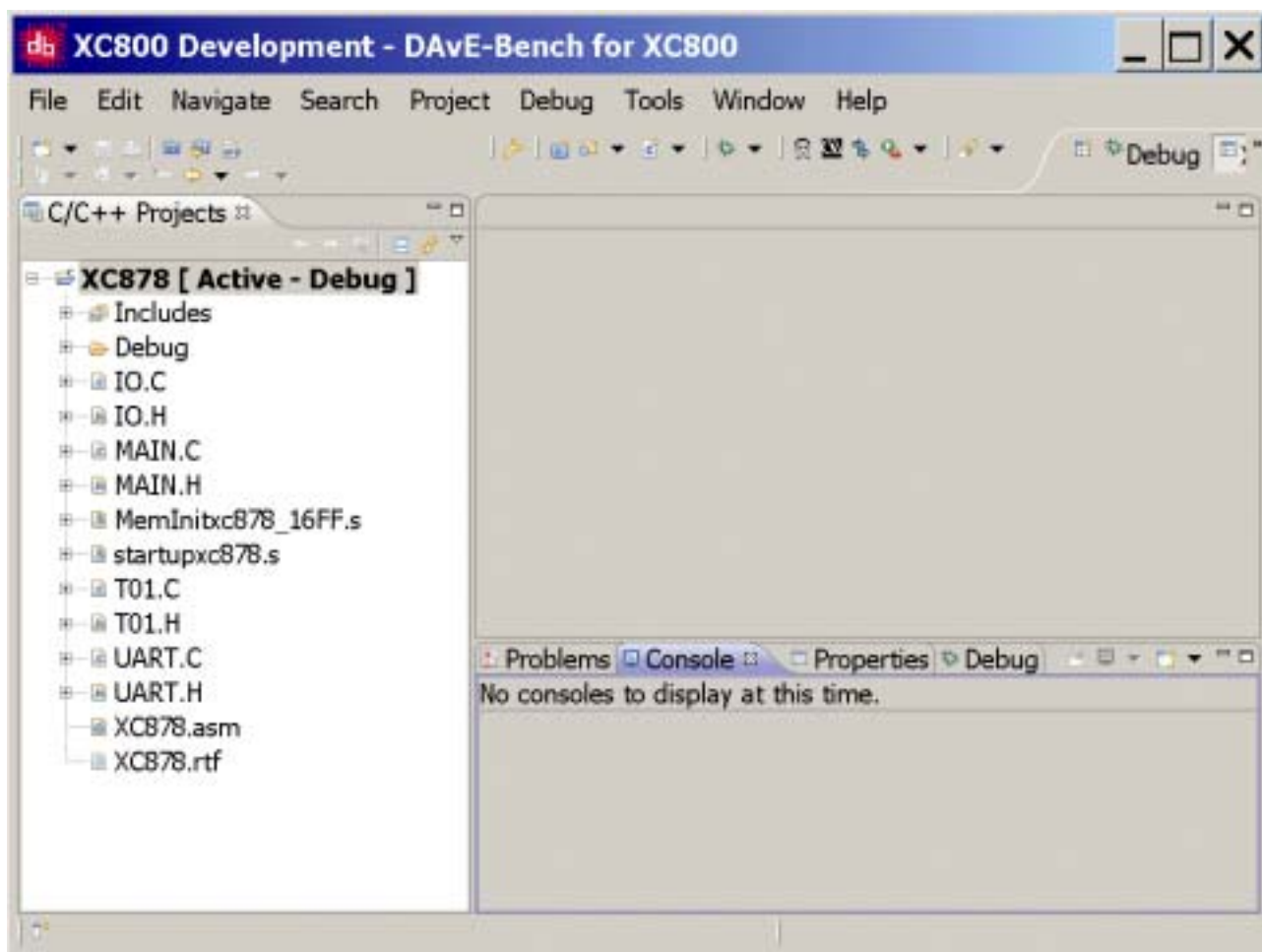
2.2) Open the DAvE Bench project and insert code:



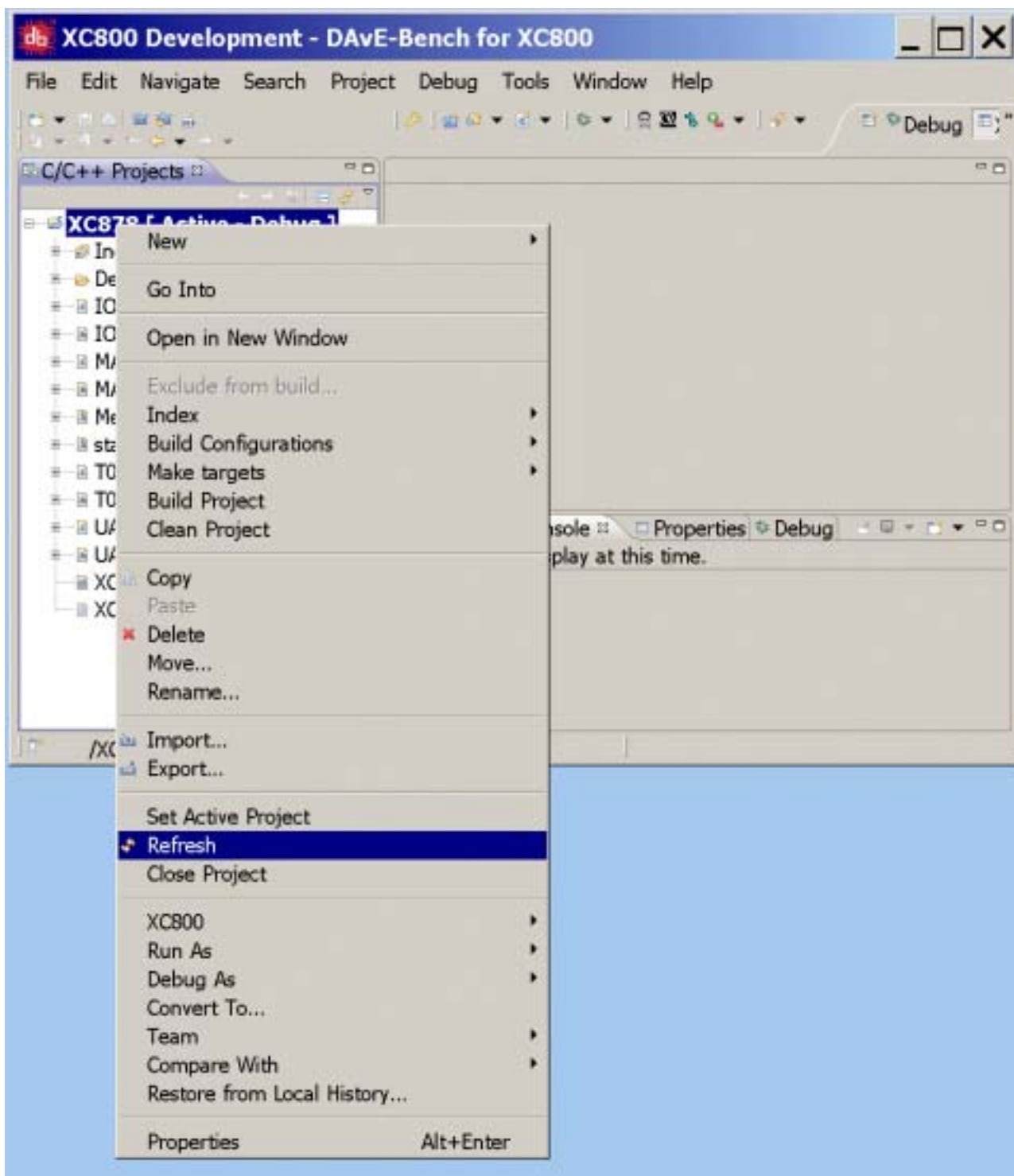
Start DAvE Bench and open your XC878 Project:

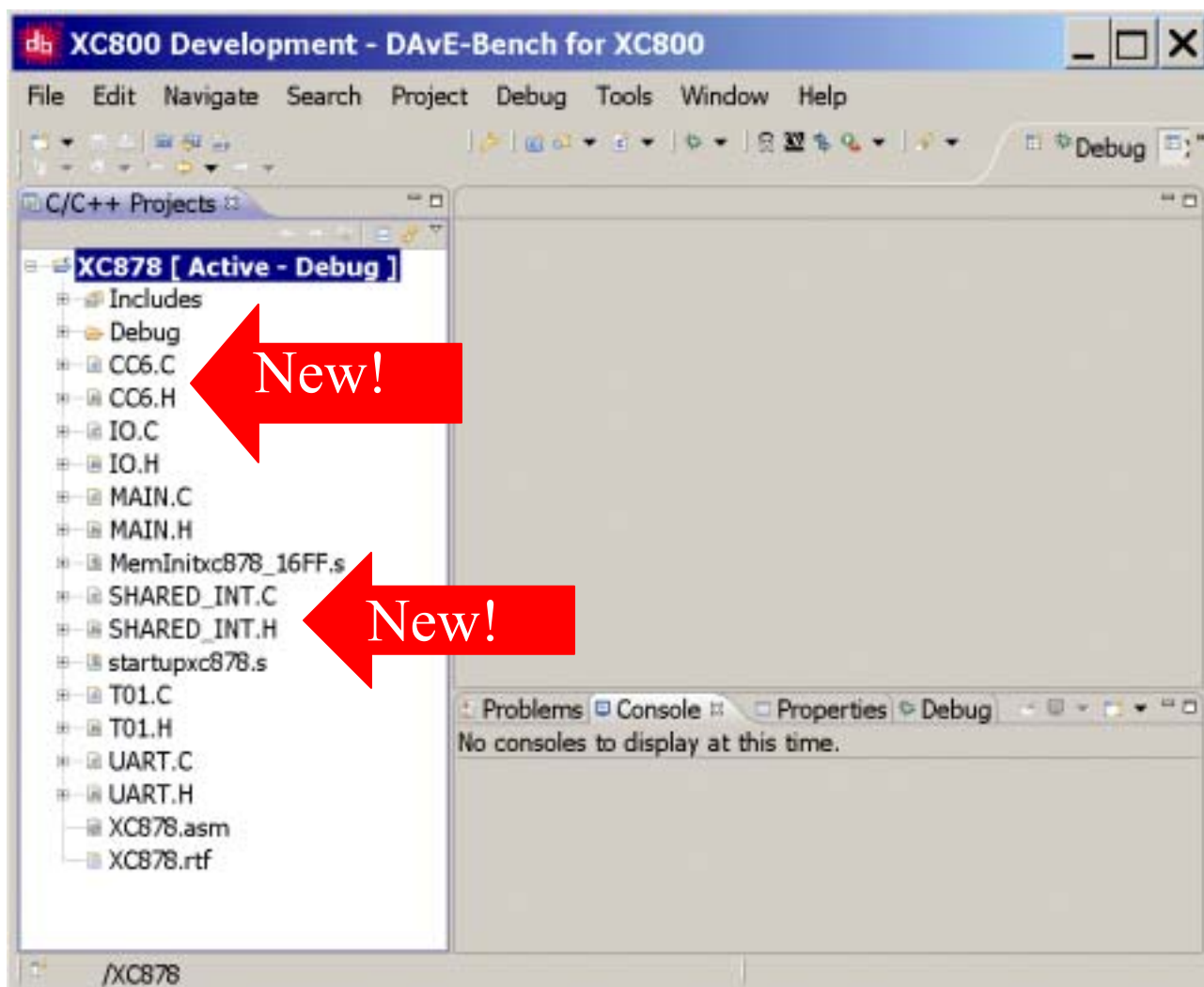


Click OK

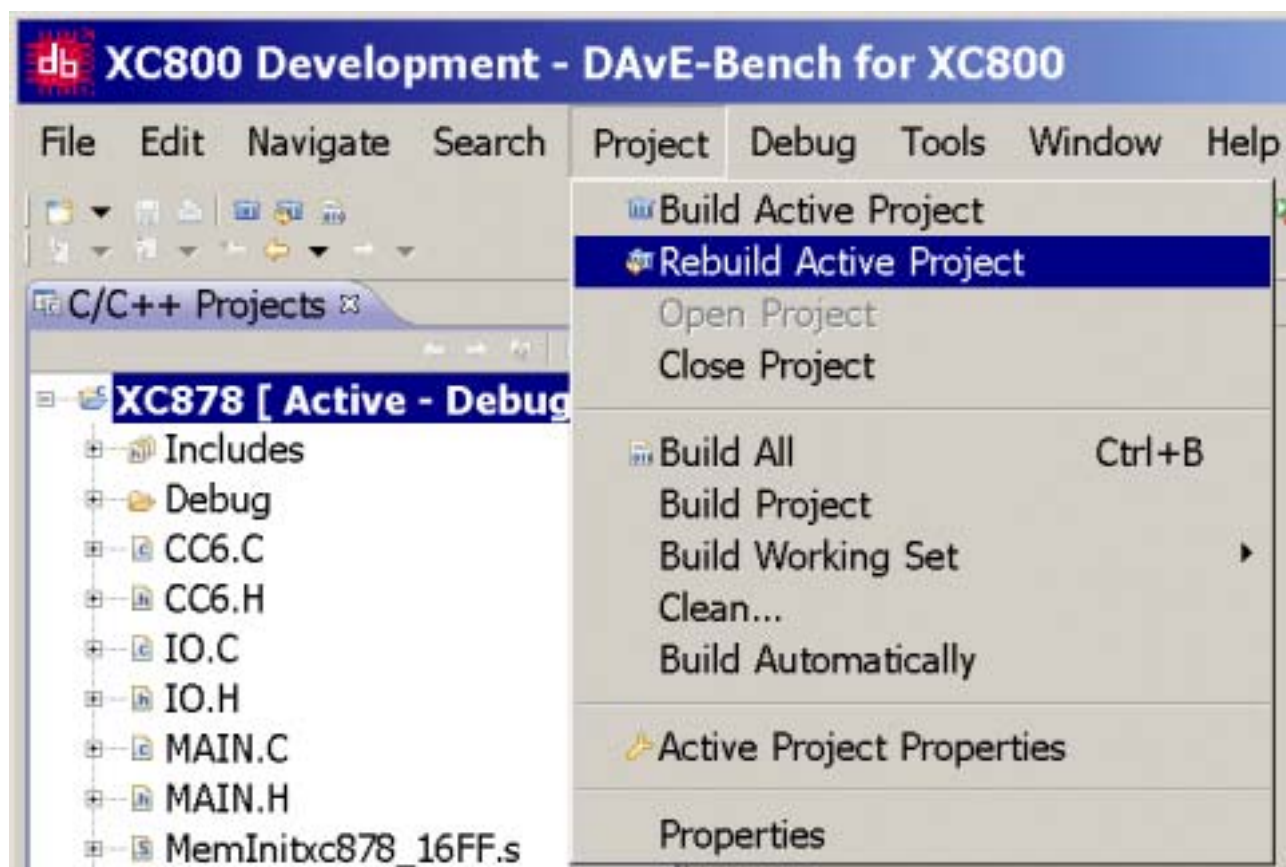


Mouse position: C/C++ Projects, XC878 [Active - Debug]: click right mouse button
click Refresh

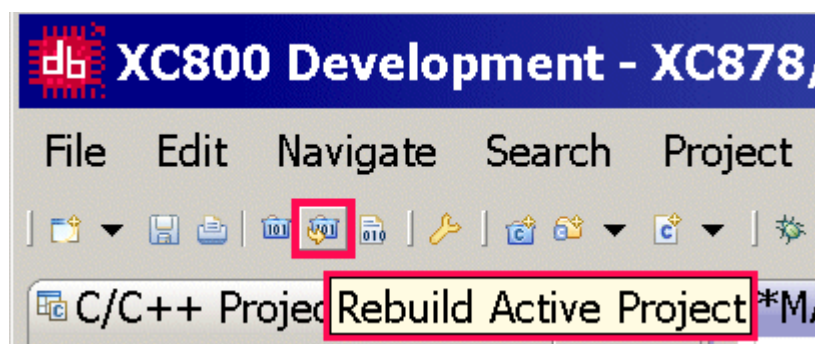


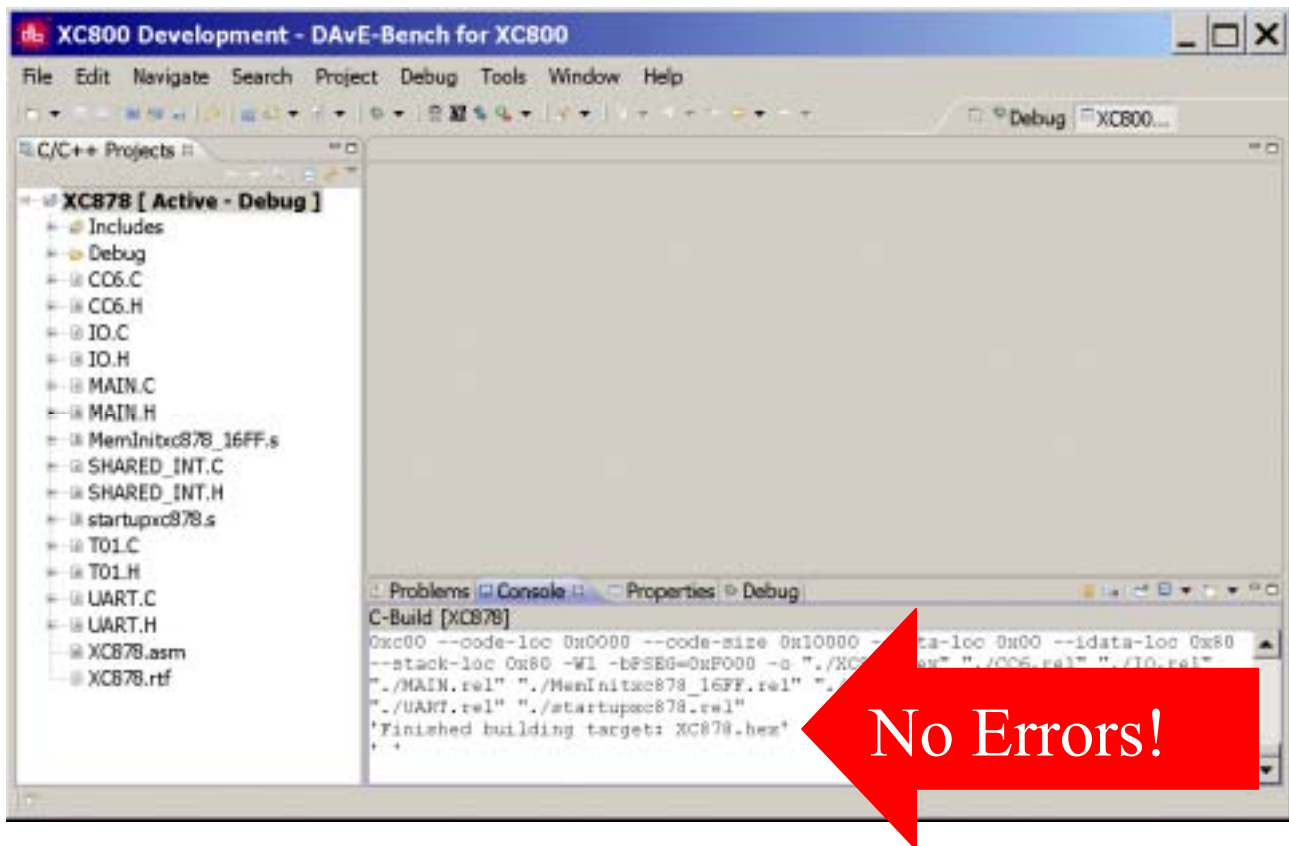


Project – Rebuild Active Project



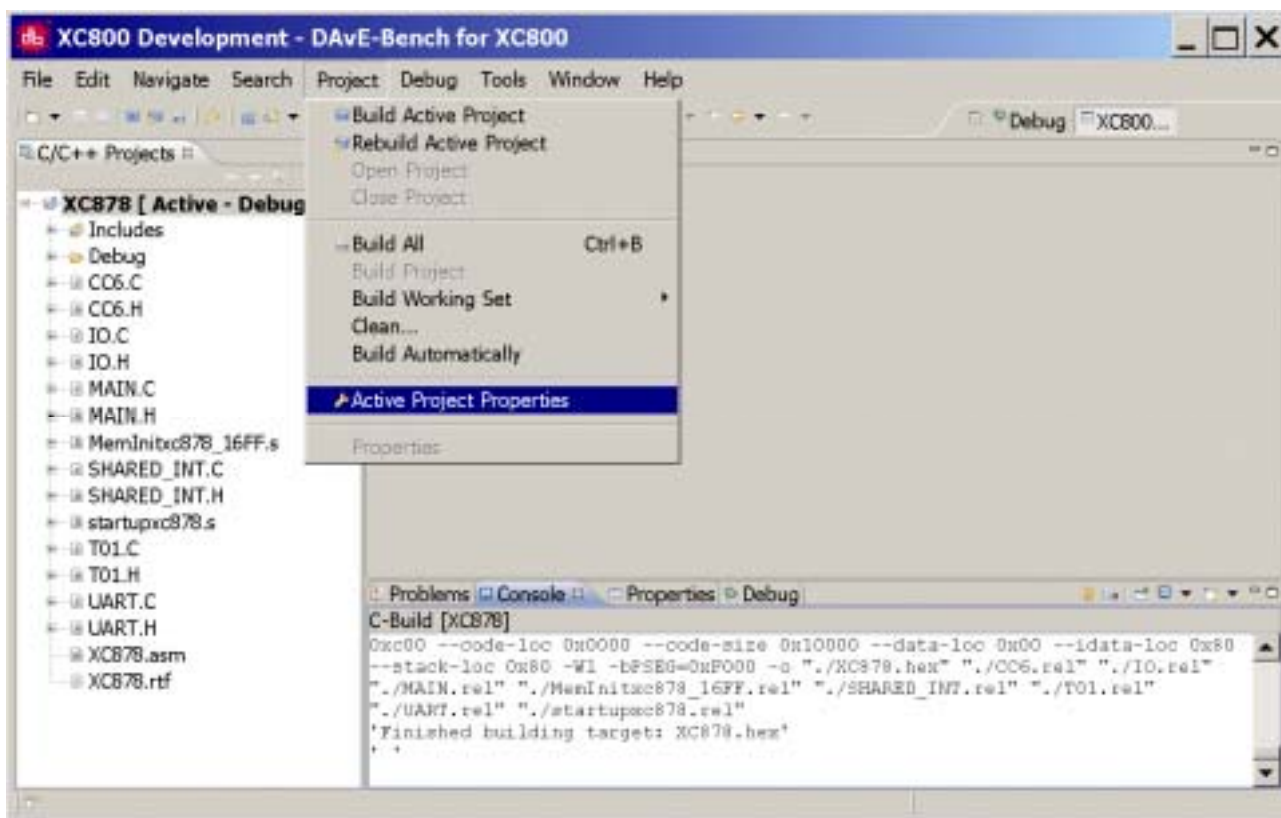
or: [click](#) 



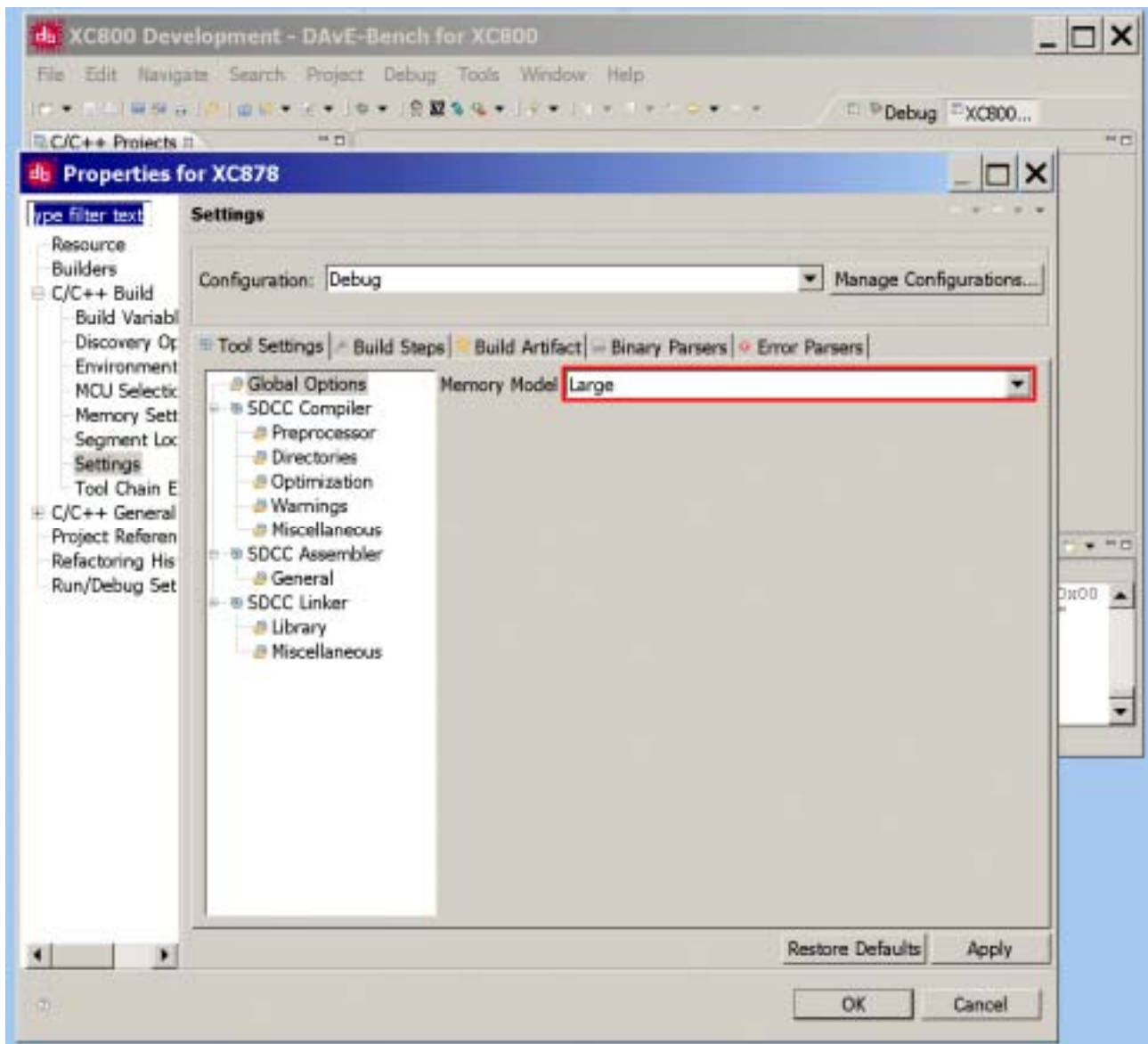


Check/Change the Memory Model:

Project – Active Project Properties



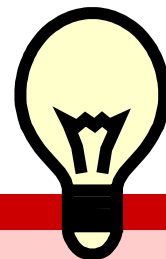
Tool Settings: Global Options: Memory Model: check Large



Click **OK**

Note:

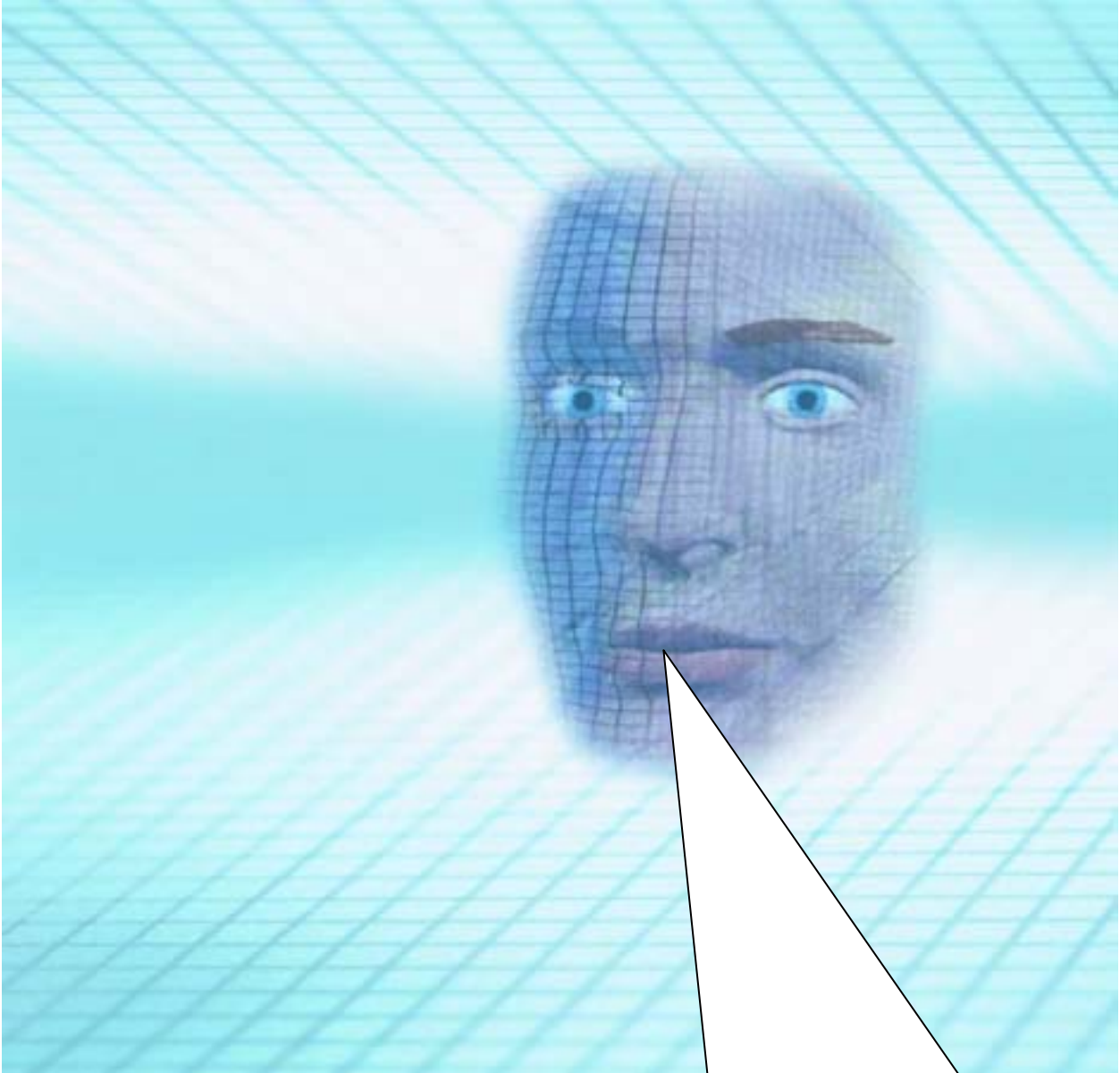
(Release Notes; DAVe Bench for XC800; SDCC Tool chain for XC800):



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`'.

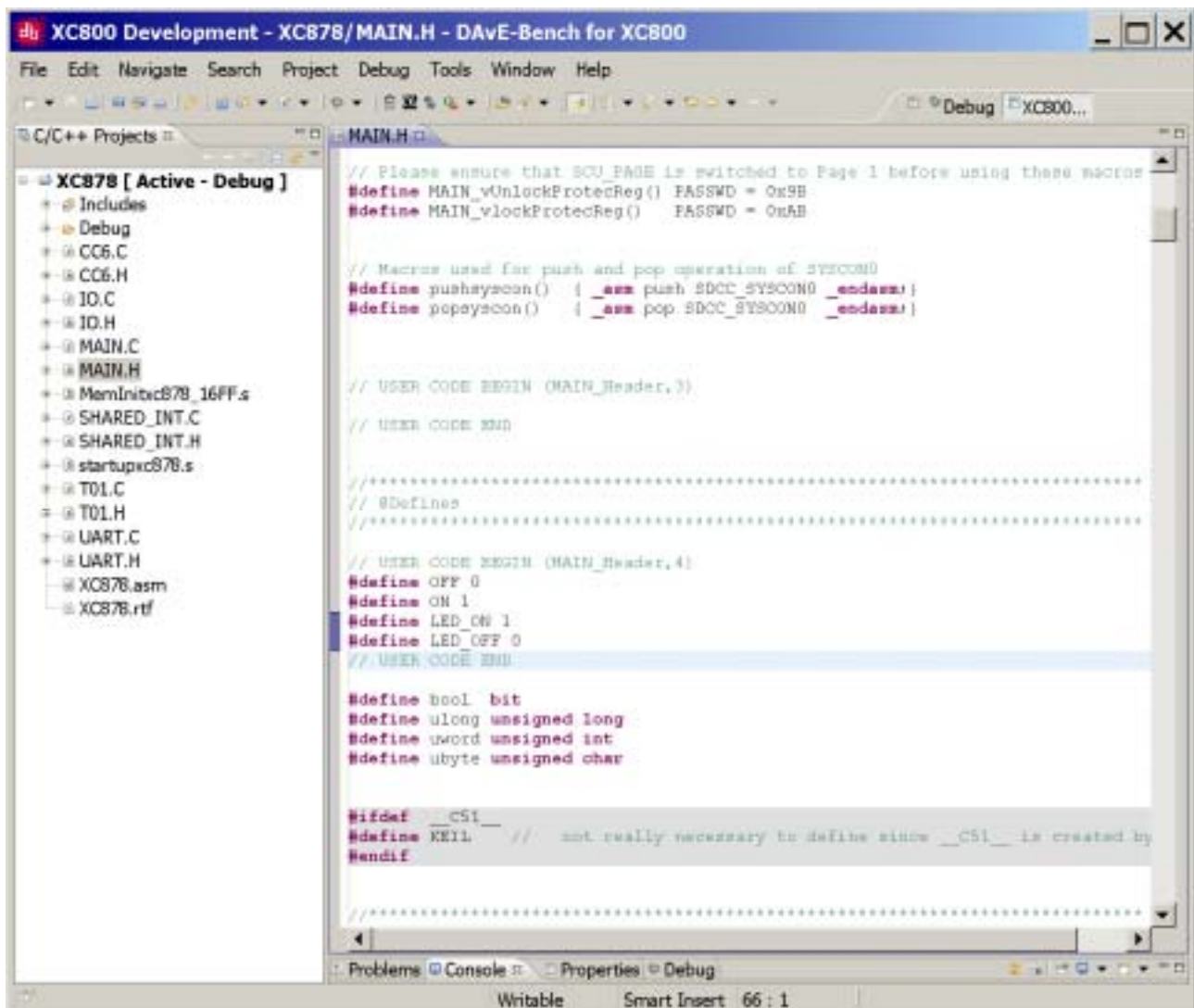
If you wish to change DAvE's generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAvE regenerate code!

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

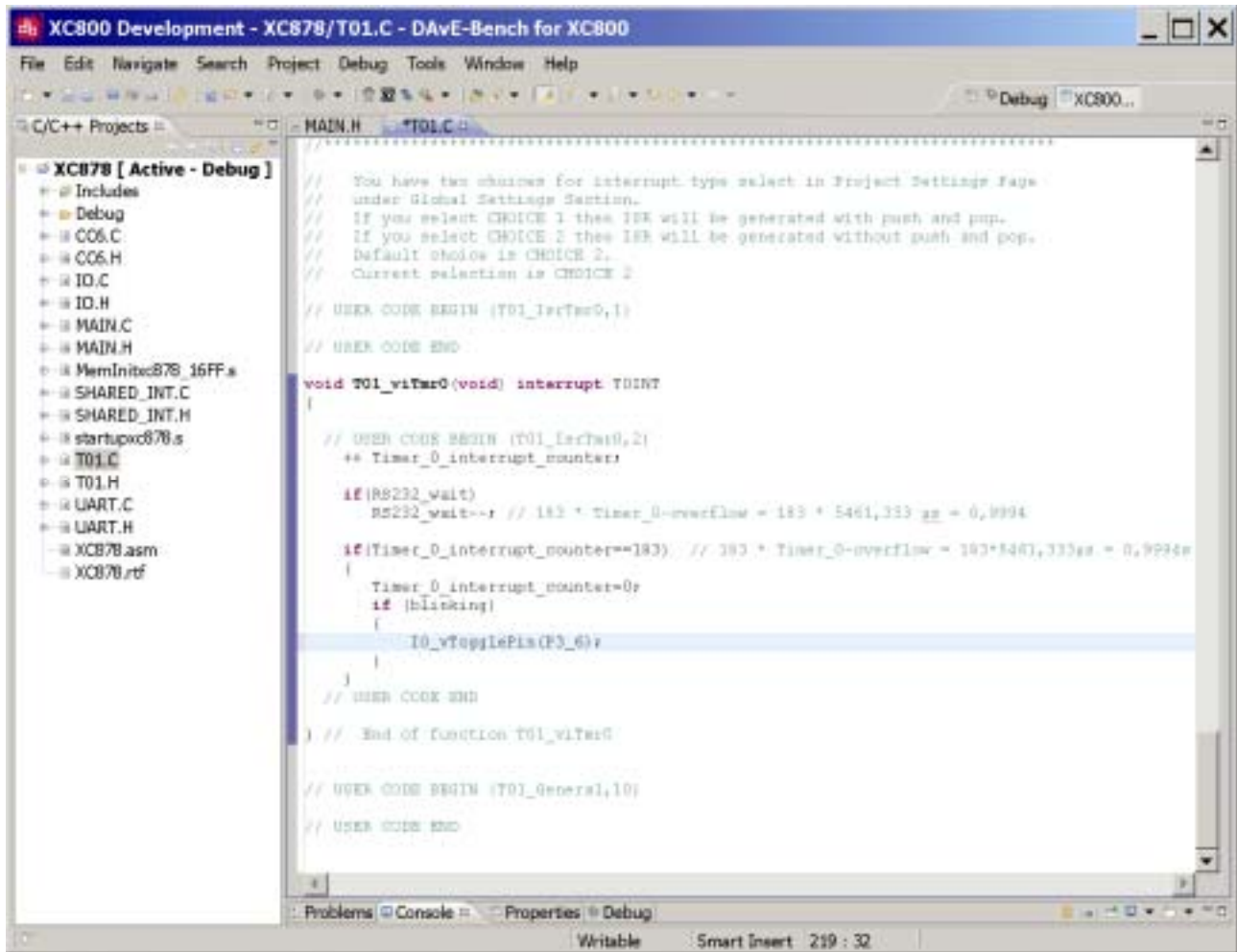
if(Timer_0_interrupt_counter==183) // 183 * Timer_0-overflow = 183*5461,333μs = 0,9994s
{
    Timer_0_interrupt_counter=0;
    if (blinking)
    {
        P3_DATA = P3_DATA^0xFF;
    }
}
```

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.



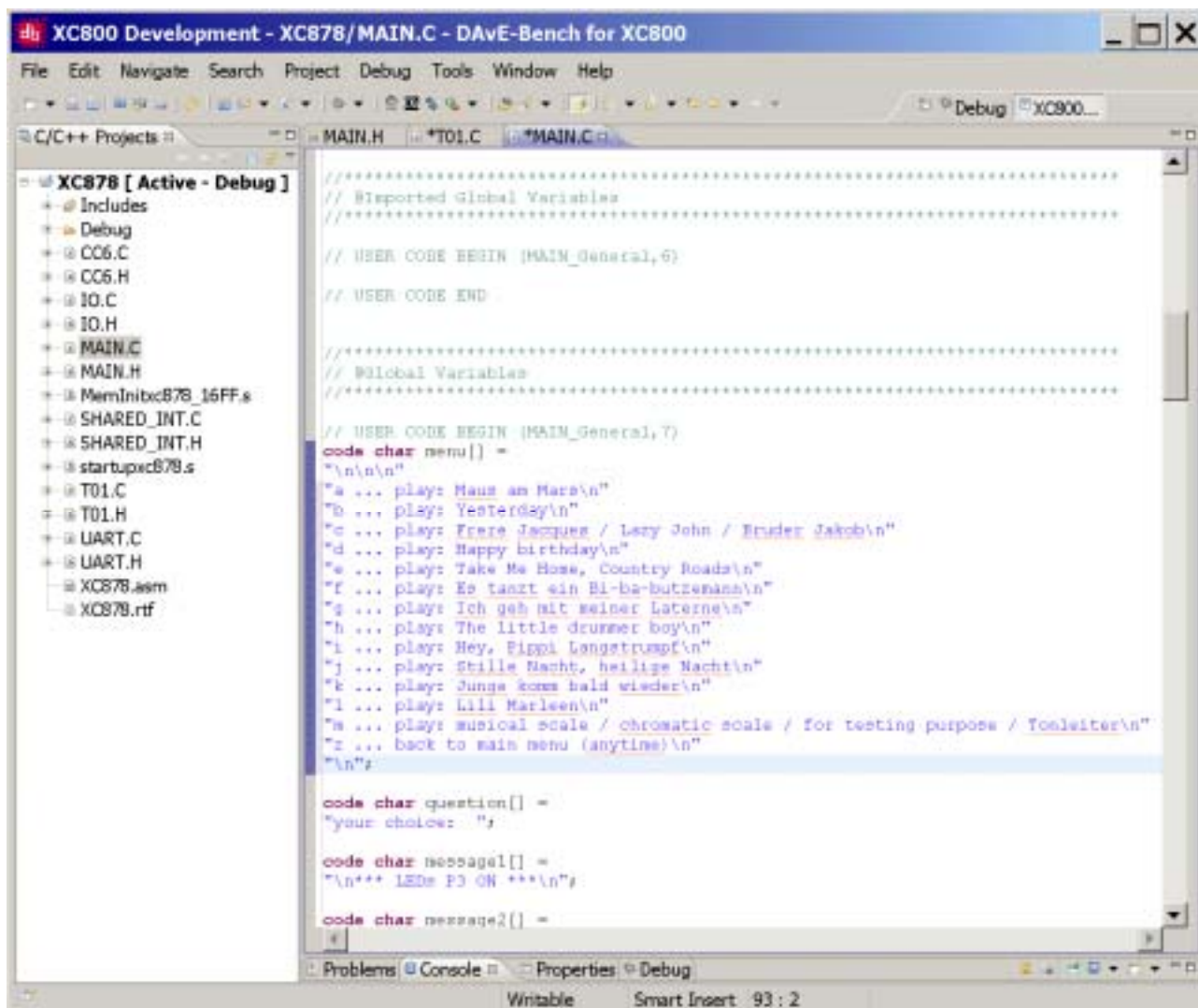
Double click **MAIN.C** and change Global Variable **menu**

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";
```



```

// *****
// Imported 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 / Lory John / Bruder Jakob\n"
"d ... play: Happy birthday\n"
"e ... play: Take Me Home, Country Roads\n"
"f ... play: Es tanzt ein Bi-ba-butzenann\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"
"n ... back to main menu (anytime)\n"
"\n";

code char question[] =
"your choice: ";

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

code char message2[] =

```

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

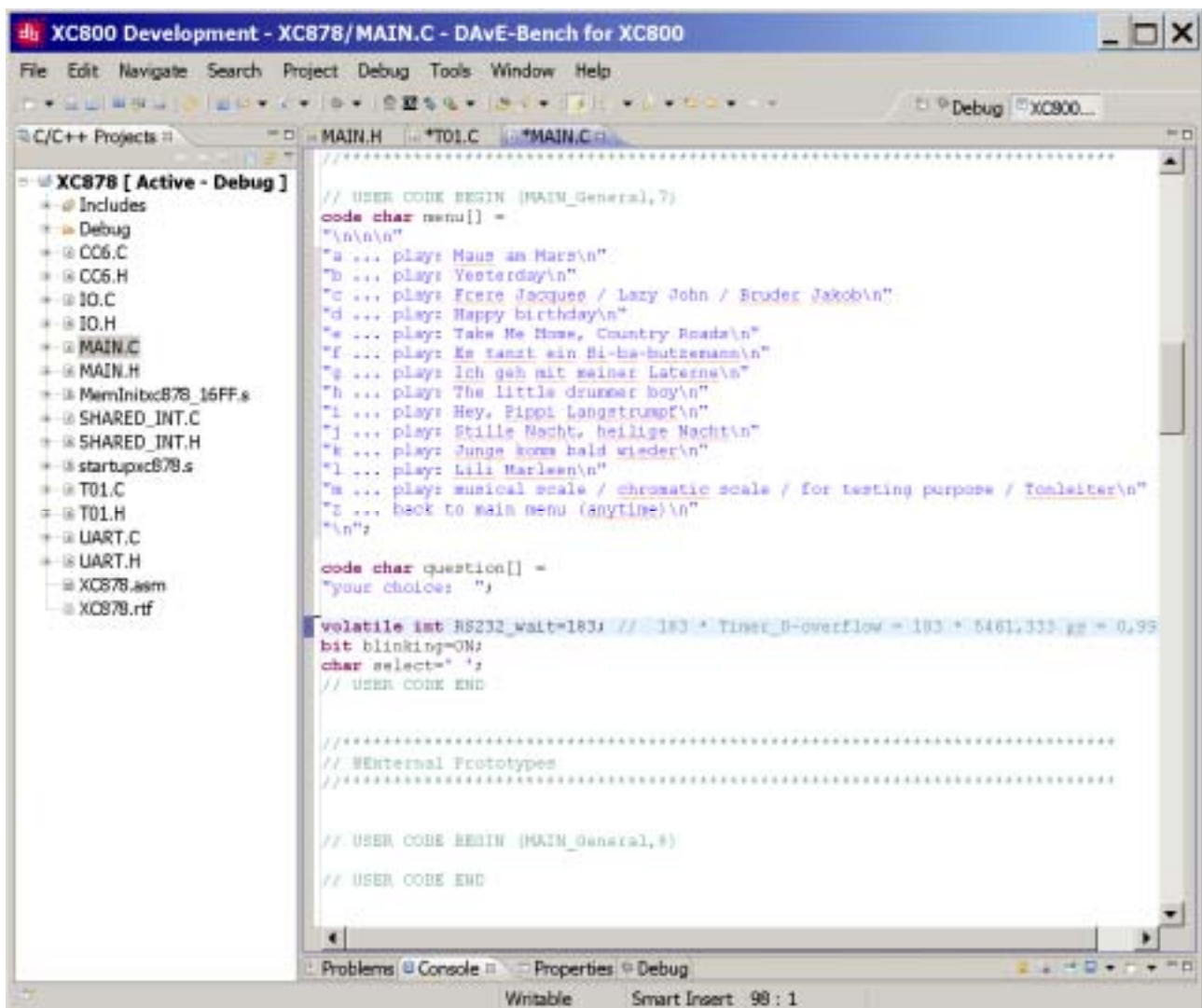
from:

```
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:



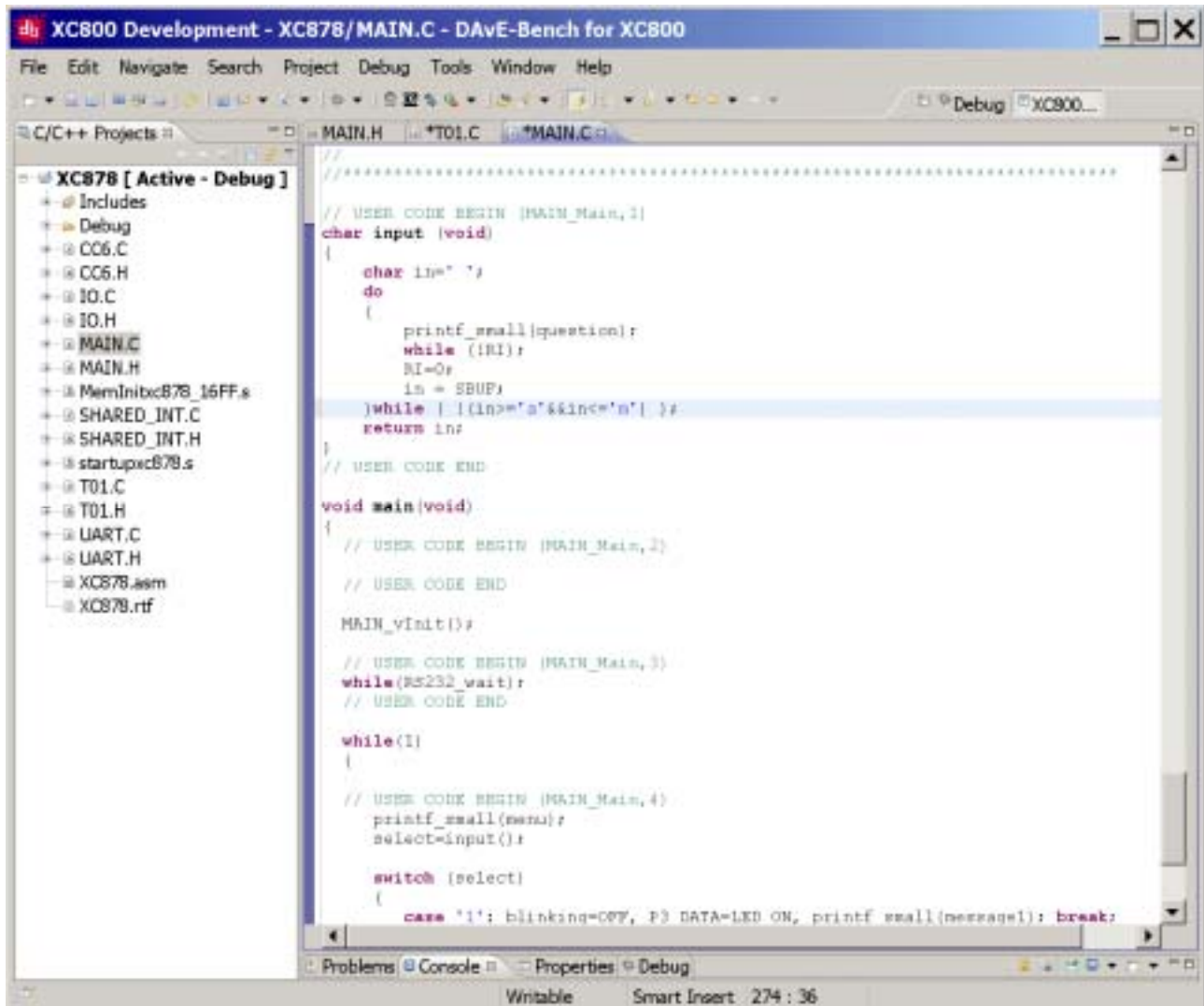
Double click **MAIN.C** and change function "char input (void)":

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

```

// USER CODE BEGIN (MAIN_Main,3)
char input (void)
{
    char in=" ";
    do
    {
        printf_small(question);
        while (!RI);
        RI=0;
        in = SBUP;
    }while (!(in>='a'&&in<='n') );
    return in;
}
// USER CODE END

void main(void)
{
    // USER CODE BEGIN (MAIN_Main,2)
    // USER CODE END

    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 '1': blinking=COFF, P3 DATA=LED ON, printf_small(message1); break;

```

Problems Console Properties Debug

Writable Smart Insert 274 : 36

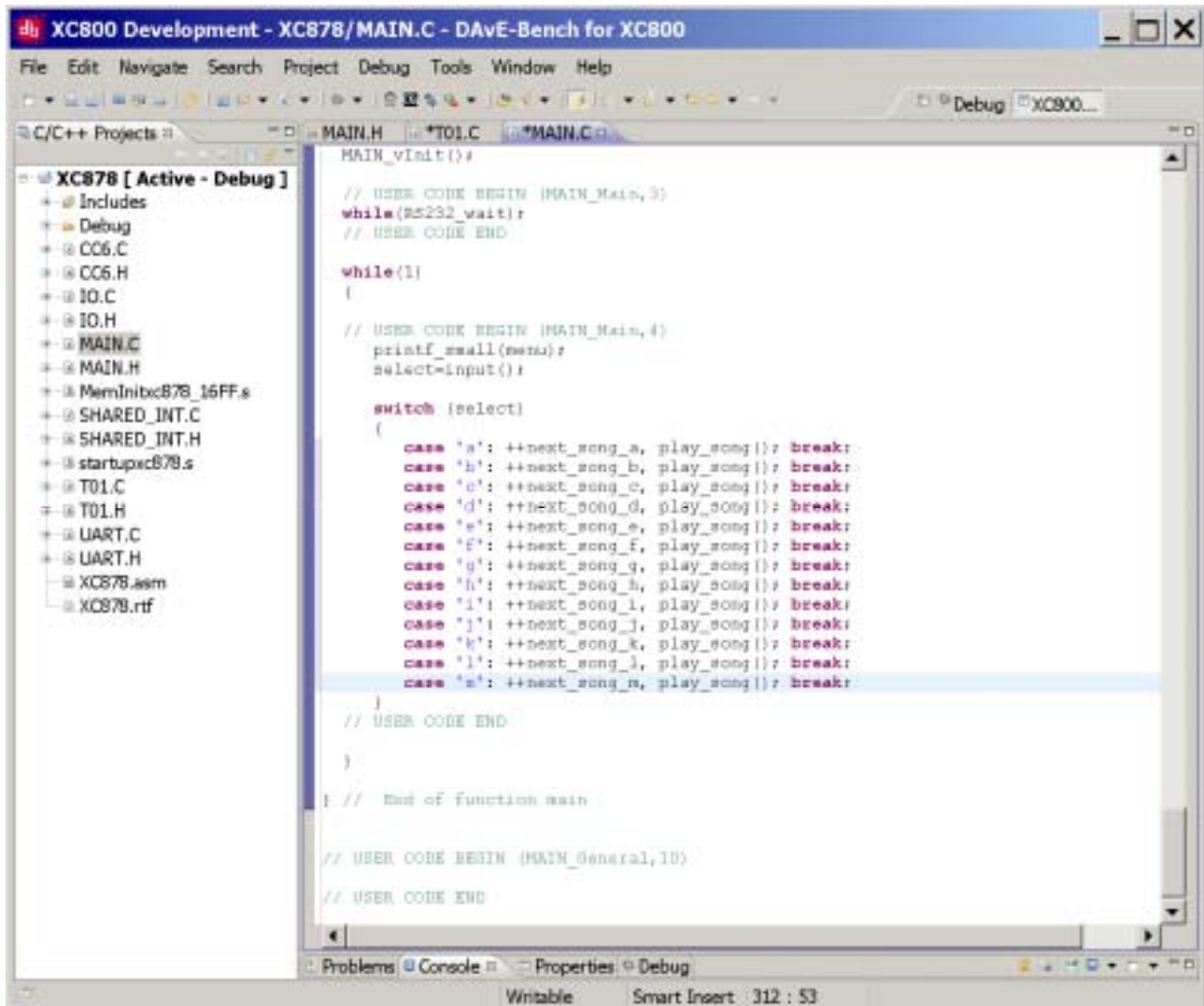
Double click **MAIN.C** and extend/change [“switch/case” in void main (void)] :

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_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(); 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 'x': ++next_song_n, play_song(); break;
}
// 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

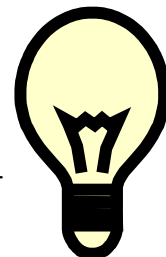
Note:
Be aware that not every song sounds good on a descant recorder.
*/

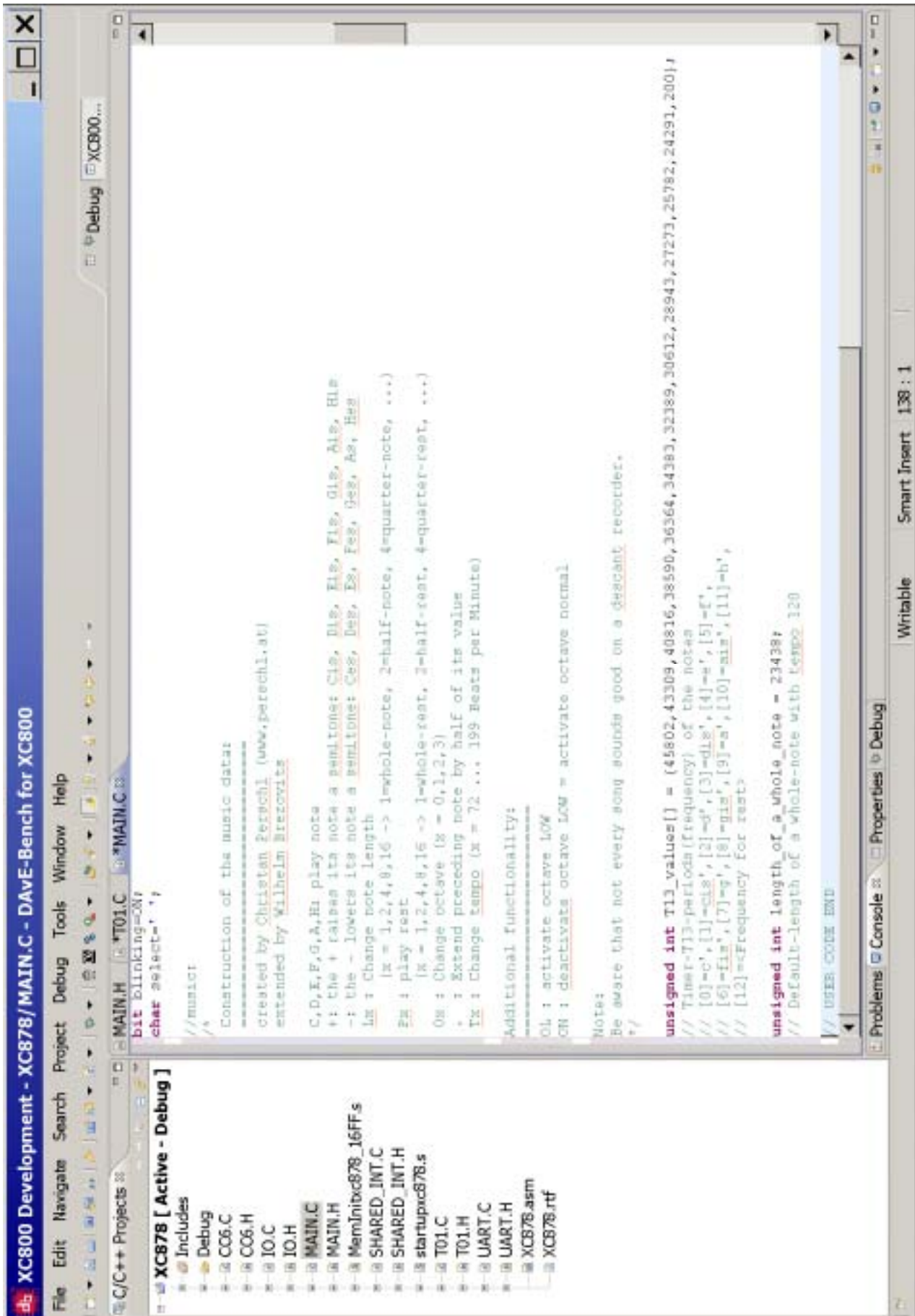
unsigned int T13_values[] =
{45802,43309,40816,38590,36364,34383,32389,30612,28943,27273,25782,24291,200};
// Timer-T13-periods(frequency) of the notes
// [0]=c',[1]=cis',[2]=d',[3]=dis',[4]=e',[5]=f',
// [6]=fis',[7]=g',[8]=gis',[9]=a',[10]=ais',[11]=h',
// [12]=<Frequency for rest>

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.





```

//music1
//
Construction of the music data:
created by Christian Persechl (www.persechl.at)
extended by Wilhelm Brezovits

C,D,E,F,G,A,H play note
+ : the + raises its note a semitone: Cis, Dis, Fis, Gis, Ais, Hls
- : the - lowers its note a semitone: Ces, Des, Es, Fes, Ges, As, Hs
lx : Change note length
lx = 1,2,4,8,16 -> 1=whole-note, 2=half-note, 4=quarter-note, ...
Px : play rest
lx = 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:
Ox : activate octave LOW
ON : deactivate octave LOW = activate octave normal

Note:
Be aware that not every song sounds good on a decent 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]='as',[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
  
```


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

//Songs:

// Maus am Mars (song a) :

code unsigned char

songa[]="T12000L4FL8AL4O1C.O0L8FEGL2O1CO0P4P8L4EL8GO1L4C.O0L8EFAL2O1CP4
P8O0L4FL8AO1L4C.O0L8FH-O1L4DFL8FEDDCO0HO1CDCO0H-GL2F.";

// Yesterday (song b) :

code unsigned char

songb[]="T12000L8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2AAAL4O1DEFL8EDL4E.L8DL4CEFCO0H-
AL8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L8DDCO0H-AGL4H-
L8AL4A.L4GFL8AL2GL8DL4FL8AL2A";

// Bruder Jakob (song c) :

code unsigned char songc[]="T12000L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0L8H-
L4AFO1L8CDCO0L8H-L4AFFCL2FL4FCL2F";

// Happy birthday (song d) :

code unsigned char

songd[]="T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C
CO0L4HGAL2G";

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

code unsigned char

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

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

code unsigned char

songf[]="T19900L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";

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

code unsigned char

songg[]="T12000L8CL4FL8FAFAO1L4C.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+L4GAAHL8AGL4F+L2EP2L4EF+L4GAAHO1L8CO0L8HL4AL2GL8
HAL4GL2F+L8AGL4F+L2EP1L2D.L4EL4F+F+F+F+L8GF+L4GL2F+P1L8EDL4EL2D";

// Hey, Pippi Langstrumpf (song i):

code unsigned char

songi[]="T1800LL4AONO0L4DF+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+L8DDDDDDL8D
+L8DL4DP8L8DL8EEEEEL2GP8L8EL1DP8L8DL8EEEL4E.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];

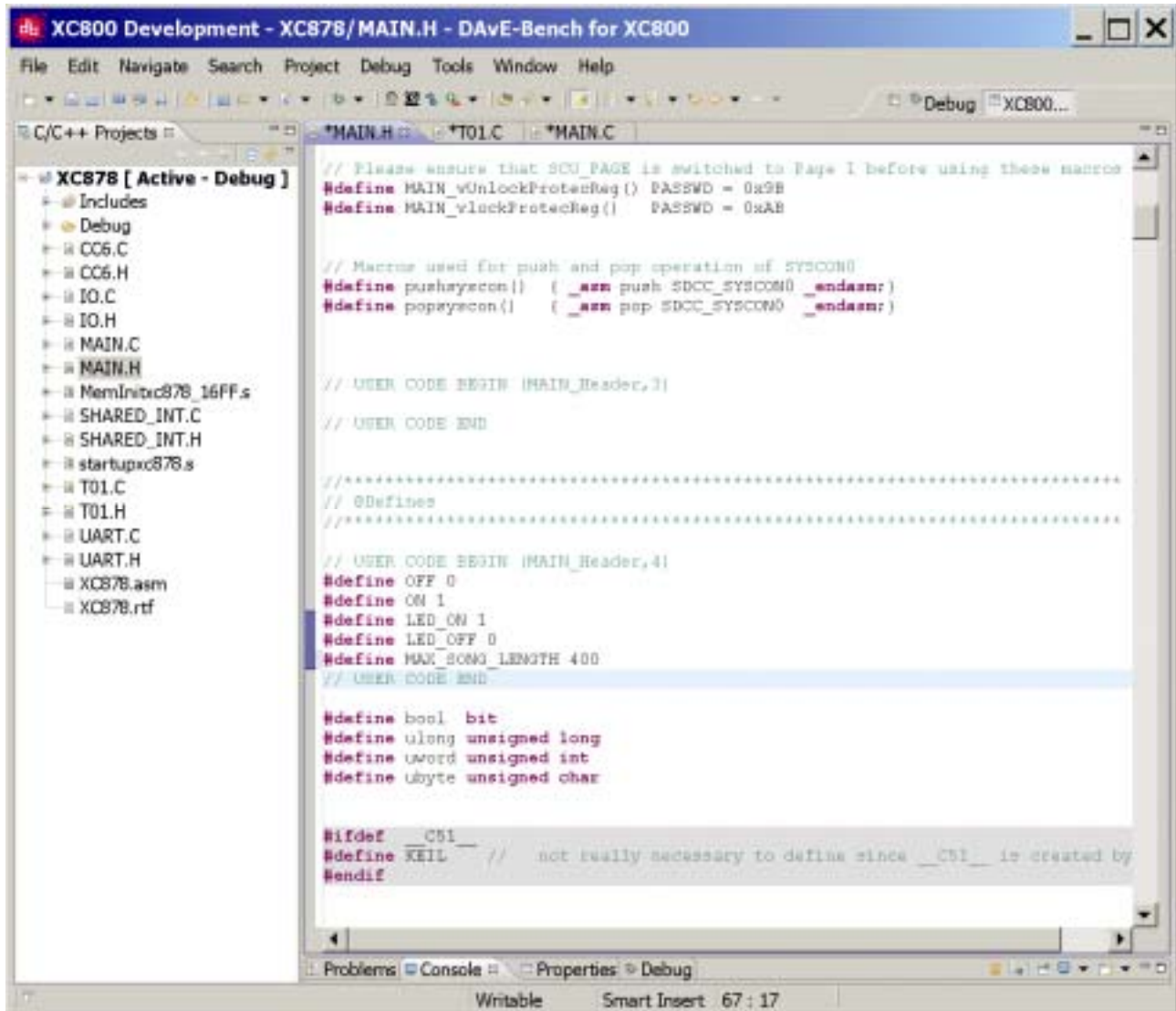
```

```

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
  C06.C
  C06.H
  IO.C
  IO.H
  MAIN.C
  MAIN.H
  MemInit878_16FF.s
  SHARED_INT.H
  startupxc878.s
  T01.C
  T01.H
  UART.C
  UART.H
  XC878.asm
  XC878.rtf
  MAIN.H
  T01.C
  MAIN.C
  // First 1/3-periods (frequency) of the notes
  // [0]='c', [1]='d', [2]='e', [3]='f', [4]='g', [5]='a', [6]='b',
  // [7]='c', [8]='d', [9]='e', [10]='f', [11]='g', [12]='a', [13]='b',
  // [14]='c', [15]='d', [16]='e', [17]='f', [18]='g', [19]='a', [20]='b',
  // [21]='c', [22]='d', [23]='e', [24]='f', [25]='g', [26]='a', [27]='b',
  // [28]='c', [29]='d', [30]='e', [31]='f', [32]='g', [33]='a', [34]='b',
  // [35]='c', [36]='d', [37]='e', [38]='f', [39]='g', [40]='a', [41]='b',
  // [42]='c', [43]='d', [44]='e', [45]='f', [46]='g', [47]='a', [48]='b',
  // [49]='c', [50]='d', [51]='e', [52]='f', [53]='g', [54]='a', [55]='b',
  // [56]='c', [57]='d', [58]='e', [59]='f', [60]='g', [61]='a', [62]='b',
  // [63]='c', [64]='d', [65]='e', [66]='f', [67]='g', [68]='a', [69]='b',
  // [70]='c', [71]='d', [72]='e', [73]='f', [74]='g', [75]='a', [76]='b',
  // [77]='c', [78]='d', [79]='e', [80]='f', [81]='g', [82]='a', [83]='b',
  // [84]='c', [85]='d', [86]='e', [87]='f', [88]='g', [89]='a', [90]='b',
  // [91]='c', [92]='d', [93]='e', [94]='f', [95]='g', [96]='a', [97]='b',
  // [98]='c', [99]='d', [100]='e', [101]='f', [102]='g', [103]='a', [104]='b',
  // [105]='c', [106]='d', [107]='e', [108]='f', [109]='g', [110]='a', [111]='b',
  // [112]='c', [113]='d', [114]='e', [115]='f', [116]='g', [117]='a', [118]='b',
  // [119]='c', [120]='d', [121]='e', [122]='f', [123]='g', [124]='a', [125]='b',
  // [126]='c', [127]='d', [128]='e', [129]='f', [130]='g', [131]='a', [132]='b',
  // [133]='c', [134]='d', [135]='e', [136]='f', [137]='g', [138]='a', [139]='b',
  // [140]='c', [141]='d', [142]='e', [143]='f', [144]='g', [145]='a', [146]='b',
  // [147]='c', [148]='d', [149]='e', [150]='f', [151]='g', [152]='a', [153]='b',
  // [154]='c', [155]='d', [156]='e', [157]='f', [158]='g', [159]='a', [160]='b',
  // [161]='c', [162]='d', [163]='e', [164]='f', [165]='g', [166]='a', [167]='b',
  // [168]='c', [169]='d', [170]='e', [171]='f', [172]='g', [173]='a', [174]='b',
  // [175]='c', [176]='d', [177]='e', [178]='f', [179]='g', [180]='a', [181]='b',
  // [182]='c', [183]='d', [184]='e', [185]='f', [186]='g', [187]='a', [188]='b',
  // [189]='c', [190]='d', [191]='e', [192]='f', [193]='g', [194]='a', [195]='b',
  // [196]='c', [197]='d', [198]='e', [199]='f', [200]='g', [201]='a', [202]='b',
  // [203]='c', [204]='d', [205]='e', [206]='f', [207]='g', [208]='a', [209]='b',
  // [210]='c', [211]='d', [212]='e', [213]='f', [214]='g', [215]='a', [216]='b',
  // [217]='c', [218]='d', [219]='e', [220]='f', [221]='g', [222]='a', [223]='b',
  // [224]='c', [225]='d', [226]='e', [227]='f', [228]='g', [229]='a', [230]='b',
  // [231]='c', [232]='d', [233]='e', [234]='f', [235]='g', [236]='a', [237]='b',
  // [238]='c', [239]='d', [240]='e', [241]='f', [242]='g', [243]='a', [244]='b',
  // [245]='c', [246]='d', [247]='e', [248]='f', [249]='g', [250]='a', [251]='b',
  // [252]='c', [253]='d', [254]='e', [255]='f', [256]='g', [257]='a', [258]='b',
  // [259]='c', [260]='d', [261]='e', [262]='f', [263]='g', [264]='a', [265]='b',
  // [266]='c', [267]='d', [268]='e', [269]='f', [270]='g', [271]='a', [272]='b',
  // [273]='c', [274]='d', [275]='e', [276]='f', [277]='g', [278]='a', [279]='b',
  // [280]='c', [281]='d', [282]='e', [283]='f', [284]='g', [285]='a', [286]='b',
  // [287]='c', [288]='d', [289]='e', [290]='f', [291]='g', [292]='a', [293]='b',
  // [294]='c', [295]='d', [296]='e', [297]='f', [298]='g', [299]='a', [300]='b',
  // [301]='c', [302]='d', [303]='e', [304]='f', [305]='g', [306]='a', [307]='b',
  // [308]='c', [309]='d', [310]='e', [311]='f', [312]='g', [313]='a', [314]='b',
  // [315]='c', [316]='d', [317]='e', [318]='f', [319]='g', [320]='a', [321]='b',
  // [322]='c', [323]='d', [324]='e', [325]='f', [326]='g', [327]='a', [328]='b',
  // [329]='c', [330]='d', [331]='e', [332]='f', [333]='g', [334]='a', [335]='b',
  // [336]='c', [337]='d', [338]='e', [339]='f', [340]='g', [341]='a', [342]='b',
  // [343]='c', [344]='d', [345]='e', [346]='f', [347]='g', [348]='a', [349]='b',
  // [350]='c', [351]='d', [352]='e', [353]='f', [354]='g', [355]='a', [356]='b',
  // [357]='c', [358]='d', [359]='e', [360]='f', [361]='g', [362]='a', [363]='b',
  // [364]='c', [365]='d', [366]='e', [367]='f', [368]='g', [369]='a', [370]='b',
  // [371]='c', [372]='d', [373]='e', [374]='f', [375]='g', [376]='a', [377]='b',
  // [378]='c', [379]='d', [380]='e', [381]='f', [382]='g', [383]='a', [384]='b',
  // [385]='c', [386]='d', [387]='e', [388]='f', [389]='g', [390]='a', [391]='b',
  // [392]='c', [393]='d', [394]='e', [395]='f', [396]='g', [397]='a', [398]='b',
  // [399]='c', [400]='d', [401]='e', [402]='f', [403]='g', [404]='a', [405]='b',
  // [406]='c', [407]='d', [408]='e', [409]='f', [410]='g', [411]='a', [412]='b',
  // [413]='c', [414]='d', [415]='e', [416]='f', [417]='g', [418]='a', [419]='b',
  // [420]='c', [421]='d', [422]='e', [423]='f', [424]='g', [425]='a', [426]='b',
  // [427]='c', [428]='d', [429]='e', [430]='f', [431]='g', [432]='a', [433]='b',
  // [434]='c', [435]='d', [436]='e', [437]='f', [438]='g', [439]='a', [440]='b',
  // [441]='c', [442]='d', [443]='e', [444]='f', [445]='g', [446]='a', [447]='b',
  // [448]='c', [449]='d', [450]='e', [451]='f', [452]='g', [453]='a', [454]='b',
  // [455]='c', [456]='d', [457]='e', [458]='f', [459]='g', [460]='a', [461]='b',
  // [462]='c', [463]='d', [464]='e', [465]='f', [466]='g', [467]='a', [468]='b',
  // [469]='c', [470]='d', [471]='e', [472]='f', [473]='g', [474]='a', [475]='b',
  // [476]='c', [477]='d', [478]='e', [479]='f', [480]='g', [481]='a', [482]='b',
  // [483]='c', [484]='d', [485]='e', [486]='f', [487]='g', [488]='a', [489]='b',
  // [490]='c', [491]='d', [492]='e', [493]='f', [494]='g', [495]='a', [496]='b',
  // [497]='c', [498]='d', [499]='e', [500]='f', [501]='g', [502]='a', [503]='b',
  // [504]='c', [505]='d', [506]='e', [507]='f', [508]='g', [509]='a', [510]='b',
  // [511]='c', [512]='d', [513]='e', [514]='f', [515]='g', [516]='a', [517]='b',
  // [518]='c', [519]='d', [520]='e', [521]='f', [522]='g', [523]='a', [524]='b',
  // [525]='c', [526]='d', [527]='e', [528]='f', [529]='g', [530]='a', [531]='b',
  // [532]='c', [533]='d', [534]='e', [535]='f', [536]='g', [537]='a', [538]='b',
  // [539]='c', [540]='d', [541]='e', [542]='f', [543]='g', [544]='a', [545]='b',
  // [546]='c', [547]='d', [548]='e', [549]='f', [550]='g', [551]='a', [552]='b',
  // [553]='c', [554]='d', [555]='e', [556]='f', [557]='g', [558]='a', [559]='b',
  // [560]='c', [561]='d', [562]='e', [563]='f', [564]='g', [565]='a', [566]='b',
  // [567]='c', [568]='d', [569]='e', [570]='f', [571]='g', [572]='a', [573]='b',
  // [574]='c', [575]='d', [576]='e', [577]='f', [578]='g', [579]='a', [580]='b',
  // [581]='c', [582]='d', [583]='e', [584]='f', [585]='g', [586]='a', [587]='b',
  // [588]='c', [589]='d', [590]='e', [591]='f', [592]='g', [593]='a', [594]='b',
  // [595]='c', [596]='d', [597]='e', [598]='f', [599]='g', [600]='a', [601]='b',
  // [602]='c', [603]='d', [604]='e', [605]='f', [606]='g', [607]='a', [608]='b',
  // [609]='c', [610]='d', [611]='e', [612]='f', [613]='g', [614]='a', [615]='b',
  // [616]='c', [617]='d', [618]='e', [619]='f', [620]='g', [621]='a', [622]='b',
  // [623]='c', [624]='d', [625]='e', [626]='f', [627]='g', [628]='a', [629]='b',
  // [630]='c', [631]='d', [632]='e', [633]='f', [634]='g', [635]='a', [636]='b',
  // [637]='c', [638]='d', [639]='e', [640]='f', [641]='g', [642]='a', [643]='b',
  // [644]='c', [645]='d', [646]='e', [647]='f', [648]='g', [649]='a', [650]='b',
  // [651]='c', [652]='d', [653]='e', [654]='f', [655]='g', [656]='a', [657]='b',
  // [658]='c', [659]='d', [660]='e', [661]='f', [662]='g', [663]='a', [664]='b',
  // [665]='c', [666]='d', [667]='e', [668]='f', [669]='g', [670]='a', [671]='b',
  // [672]='c', [673]='d', [674]='e', [675]='f', [676]='g', [677]='a', [678]='b',
  // [679]='c', [680]='d', [681]='e', [682]='f', [683]='g', [684]='a', [685]='b',
  // [686]='c', [687]='d', [688]='e', [689]='f', [690]='g', [691]='a', [692]='b',
  // [693]='c', [694]='d', [695]='e', [696]='f', [697]='g', [698]='a', [699]='b',
  // [700]='c', [701]='d', [702]='e', [703]='f', [704]='g', [705]='a', [706]='b',
  // [707]='c', [708]='d', [709]='e', [710]='f', [711]='g', [712]='a', [713]='b',
  // [714]='c', [715]='d', [716]='e', [717]='f', [718]='g', [719]='a', [720]='b',
  // [721]='c', [722]='d', [723]='e', [724]='f', [725]='g', [726]='a', [727]='b',
  // [728]='c', [729]='d', [730]='e', [731]='f', [732]='g', [733]='a', [734]='b',
  // [735]='c', [736]='d', [737]='e', [738]='f', [739]='g', [740]='a', [741]='b',
  // [742]='c', [743]='d', [744]='e', [745]='f', [746]='g', [747]='a', [748]='b',
  // [749]='c', [750]='d', [751]='e', [752]='f', [753]='g', [754]='a', [755]='b',
  // [756]='c', [757]='d', [758]='e', [759]='f', [760]='g', [761]='a', [762]='b',
  // [763]='c', [764]='d', [765]='e', [766]='f', [767]='g', [768]='a', [769]='b',
  // [770]='c', [771]='d', [772]='e', [773]='f', [774]='g', [775]='a', [776]='b',
  // [777]='c', [778]='d', [779]='e', [780]='f', [781]='g', [782]='a', [783]='b',
  // [784]='c', [785]='d', [786]='e', [787]='f', [788]='g', [789]='a', [790]='b',
  // [791]='c', [792]='d', [793]='e', [794]='f', [795]='g', [796]='a', [797]='b',
  // [798]='c', [799]='d', [800]='e', [801]='f', [802]='g', [803]='a', [804]='b',
  // [805]='c', [806]='d', [807]='e', [808]='f', [809]='g', [810]='a', [811]='b',
  // [812]='c', [813]='d', [814]='e', [815]='f', [816]='g', [817]='a', [818]='b',
  // [819]='c', [820]='d', [821]='e', [822]='f', [823]='g', [824]='a', [825]='b',
  // [826]='c', [827]='d', [828]='e', [829]='f', [830]='g', [831]='a', [832]='b',
  // [833]='c', [834]='d', [835]='e', [836]='f', [837]='g', [838]='a', [839]='b',
  // [840]='c', [841]='d', [842]='e', [843]='f', [844]='g', [845]='a', [846]='b',
  // [847]='c', [848]='d', [849]='e', [850]='f', [851]='g', [852]='a', [853]='b',
  // [854]='c', [855]='d', [856]='e', [857]='f', [858]='g', [859]='a', [860]='b',
  // [861]='c', [862]='d', [863]='e', [864]='f', [865]='g', [866]='a', [867]='b',
  // [868]='c', [869]='d', [870]='e', [871]='f', [872]='g', [873]='a', [874]='b',
  // [875]='c', [876]='d', [877]='e', [878]='f', [879]='g', [880]='a', [881]='b',
  // [882]='c', [883]='d', [884]='e', [885]='f', [886]='g', [887]='a', [888]='b',
  // [889]='c', [890]='d', [891]='e', [892]='f', [893]='g', [894]='a', [895]='b',
  // [896]='c', [897]='d', [898]='e', [899]='f', [900]='g', [901]='a', [902]='b',
  // [903]='c', [904]='d', [905]='e', [906]='f', [907]='g', [908]='a', [909]='b',
  // [910]='c', [911]='d', [912]='e', [913]='f', [914]='g', [915]='a', [916]='b',
  // [917]='c', [918]='d', [919]='e', [920]='f', [921]='g', [922]='a', [923]='b',
  // [924]='c', [925]='d', [926]='e', [927]='f', [928]='g', [929]='a', [930]='b',
  // [931]='c', [932]='d', [933]='e', [934]='f', [935]='g', [936]='a', [937]='b',
  // [938]='c', [939]='d', [940]='e', [941]='f', [942]='g', [943]='a', [944]='b',
  // [945]='c', [946]='d', [947]='e', [948]='f', [949]='g', [950]='a', [951]='b',
  // [952]='c', [953]='d', [954]='e', [955]='f', [956]='g', [957]='a', [958]='b',
  // [959]='c', [960]='d', [961]='e', [962]='f', [963]='g', [964]='a', [965]='b',
  // [966]='c', [967]='d', [968]='e', [969]='f', [970]='g', [971]='a', [972]='b',
  // [973]='c', [974]='d', [975]='e', [976]='f', [977]='g', [978]='a', [979]='b',
  // [980]='c', [981]='d', [982]='e', [983]='f', [984]='g', [985]='a', [986]='b',
  // [987]='c', [988]='d', [989]='e', [990]='f', [991]='g', [992]='a', [993]='b',
  // [994]='c', [995]='d', [996]='e', [997]='f', [998]='g', [999]='a', [1000]='b',
  // [1001]='c', [1002]='d', [1003]='e', [1004]='f', [1005]='g', [1006]='a', [1007]='b',
  // [1008]='c', [1009]='d', [1010]='e', [1011]='f', [1012]='g', [1013]='a', [1014]='b',
  // [1015]='c', [1016]='d', [1017]='e', [1018]='f', [1019]='g', [1020]='a', [1021]='b',
  // [1022]='c', [1023]='d', [1024]='e', [1025]='f', [1026]='g', [1027]='a', [1028]='b',
  // [1029]='c', [1030]='d', [1031]='e', [1032]='f', [1033]='g', [1034]='a', [1035]='b',
  // [1036]='c', [1037]='d', [1038]='e', [1039]='f', [1040]='g', [1041]='a', [1042]='b',
  // [1043]='c', [1044]='d', [1045]='e', [1046]='f', [1047]='g', [1048]='a', [1049]='b',
  // [1050]='c', [1051]='d', [1052]='e', [1053]='f', [1054]='g', [1055]='a', [1056]='b',
  // [1057]='c', [1058]='d', [1059]='e', [1060]='f', [1061]='g', [1062]='a', [1063]='b',
  // [1064]='c', [1065]='d', [1066]='e', [1067]='f', [1068]='g', [1069]='a', [1070]='b',
  // [1071]='c', [1072]='d', [1073]='e', [1074]='f', [1075]='g', [1076]='a', [1077]='b',
  // [1078]='c', [1079]='d', [1080]='e', [1081]='f', [1082]='g', [1083]='a', [1084]='b',
  // [1085]='c', [1086]='d', [1087]='e', [1088]='f', [1089]='g', [1090]='a', [1091]='b',
  // [1092]='c', [1093]='d', [1094]='e', [1095]='f', [1096]='g', [1097]='a', [1098]='b',
  // [1099]='c', [1100]='d', [1101]='e', [1102]='f', [1103]='g', [1104]='a', [1105]='b',
  // [1106]='c', [1107]='d', [1108]='e', [1109]='f', [1110]='g', [1111]='a', [1112]='b',
  // [1113]='c', [1114]='d', [1115]='e', [1116]='f', [1117]='g', [1118]='a', [1119]='b',
  // [1120]='c', [1121]='d', [1122]='e', [1123]='f', [1124]='g', [1125]='a', [1126]='b',
  // [1127]='c', [1128]='d', [1129]='e', [1130]='f', [1131]='g', [1132]='a', [1133]='b',
  // [1134]='c', [1135]='d', [1136]='e', [1137]='f', [1138]='g', [1139]='a', [1140]='b',
  // [1141]='c', [1142]='d', [1143]='e', [1144]='f', [1145]='g', [1146]='a', [1147]='b',
  // [1148]='c', [1149]='d', [1150]='e', [1151]='f', [1152]='g', [1153]='a', [1154]='b',
  // [1155]='c', [1156]='d', [1157]='e', [1158]='f', [1159]='g', [1160]='a', [1161]='b',
  // [1162]='c', [1163]='d', [1164]='e', [1165]='f', [1166]='g', [1167]='a', [1168]='b',
  // [1169]='c', [1170]='d', [1171]='e', [1172]='f', [1173]='g', [1174]='a', [1175]='b',
  // [1176]='c', [1177]='d', [1178]='e', [1179]='f', [1180]='g', [1181]='a', [1182]='b',
  // [1183]='c', [1184]='d', [1185]='e', [1186]='f', [1187]='g', [1188]='a', [1189]='b',
  // [1190]='c', [1191]='d', [1192]='e', [1193]='f', [1194]='g', [1195]='a', [1196]='b',
  // [1197]='c', [1198]='d', [1199]='e', [1200]='f', [1201]='g', [1202]='a', [1203]='b',
  // [1204]='c', [1205]='d', [1206]='e', [1207]='f', [1208]='g', [1209]='a', [1210]='b',
  // [1211]='c', [1212]='d', [1213]='e', [1214]='f', [1215]='g', [1216]='a', [1217]='b',
  // [1218]='c', [1219]='d', [1220]='e', [1221]='f', [1222]='g', [1223]='a', [1224]='b',
  // [1225]='c', [1226]='d', [1227]='e', [1228]='f', [1229]='g', [1230]='a', [1231]='b',
  // [1232]='c', [1233]='d', [1234]='e', [1235]='f', [1236]='g', [1237]='a', [1238]='b',
  // [1239]='c', [1240]='d', [1241]='e', [1242]='f', [1243]='g', [1244]='a', [1245]='b',
  // [1246]='c', [1247]='d', [1248]='e', [1249]='f', [1250]='g', [1251]='a', [1252]='b',
  // [1253]='c', [1254]='d', [1255]='e', [1256]='f', [1257]='g', [1258]='a', [1259]='b',
  // [1260]='c', [1261]='d', [1262]='e', [1263]='f', [1264]='g', [1265]='a', [1266]='b',
  // [1267]='c', [1268]='d', [1269]='e', [1270]='f', [1271]='g', [1272]='a', [1273]='b',
  // [1274]='c', [1275]='d', [1276]='e', [1277]='f', [1278]='g', [1279]='a', [1280]='b',
  // [1281]='c', [1282]='d', [1283]='e', [1284]='f', [1285]='g', [1286]='a', [1287]='b',
  // [1288]='c', [1289]='d', [1290]='e', [1291]='f', [1292]='g', [1293]='a', [1294]='b',
  // [1295]='c', [1296]='d', [1297]='e', [1298]='f', [1299]='g', [1300]='a', [1301]='b',
  // [1302]='c', [1303]='d', [1304]='e', [1305]='f', [1306]='g', [1307]='a', [1308]='b',
  // [1309]='c', [1310]='d', [1311]='e', [1312]='f', [1313]='g', [1314]='a', [1315]='b',
  // [1316]='c', [1317]='d', [1318]='e', [1319]='f', [1320]='g', [1321]='a', [1322]='b',
  // [1323]='c', [1324]='d', [1325]='e', [1326]='f', [1327]='g', [1328]='a', [1329]='b',
  // [1330]='c', [1331]='d', [1332]='e', [1333]='f', [1334]='g', [1335]='a', [1336]='b',
  // [1337]='c', [1338]='d', [1339]='e', [1340]='f', [1341]='g', [1342]='a', [1343]='b',
  // [1344]='c', [1345]='d', [1346]='e', [1347]='f', [1348]='g', [1349]='a', [1350]='b',
  // [1351]='c', [1352]='d', [1353]='e', [1354]='f', [1355]='g', [1356]='a', [1357]='b',
  // [1358]='c', [1359]='d', [1360]='e', [1361]='f', [1362]='g', [1363]='a', [1364]='b',
  // [1365]='c', [1366]='d', [1367]='e', [1368]='f', [1369]='g', [1370]='a', [1371]='b',
  // [1372]='c', [1373]='d', [1374]='e', [1375]='f', [1376]='g', [1377]='a', [1378]='b',
  // [1379]='c', [1380]='d', [1381]='e', [1382]='f', [1383]='g', [1384]='a', [1385]='b',
  // [1386]='c', [1387]='d', [1388]='e', [1389]='f', [1390]='g', [1391]='a', [1392]='b',
  // [1393]='c', [1394]='d', [1395]='e', [1396]='f', [1397]='g', [1398]='a', [1399]='b',
  // [1400]='c', [1401]='d', [1402]='e', [1403]='f', [1404]='g', [1405]='a', [1406]='b',
  // [1407]='c', [1408]='d', [1409]='e', [1410]='f', [1411]='g', [1412]='a', [1413]='b',
  // [1414]='c', [1415]='d', [1416]='e', [1417]='f', [1418]='g', [1419]='a', [1420]='b',
  // [1421]='c', [1422]='d', [1423]='e', [1424]='f', [1425]='g', [1426]='a', [1427]='b',
  // [1428]='c', [1429]='d', [1430]='e', [1431]='f', [1432]='g', [1433]='a', [1434]='b',
  // [1435]='c', [1436]='d', [1437]='e', [1438]='f', [1439]='g', [1440]='a', [1441]='b',
  // [1442]='c', [1443]='d', [1444]='e', [1445]='f', [1446]='g', [1447]='a', [1448]='b',
  // [1449]='c', [1450]='d', [1451]='e', [1452]='f', [1453]='g', [1454]='a', [1455]='b',
  // [1456]='c', [1457]='d', [1458]='e', [1459]='f', [1460]='g', [1461]='a', [1462]='b',
  // [1463]='c', [1464]='d', [1465]='e', [1466]='f', [1467]='g', [1468]='a', [1469]='b',
  // [1470]='c', [1471]='d', [1472]='e', [1473]='f', [1474]='g', [1475]='a', [1476]='b',
  // [1477]='c', [1478]='d', [1479]='e', [1480]='f', [1481]='g', [1482]='a', [1483]='b',
  // [1484]='c', [1485]='d', [1486]='e', [1487]='f', [1488]='g', [1489]='a', [1490]='b',
  // [1491]='c', [1492]='d', [1493]='e', [1494]='f', [1495]='g', [1496]='a', [1497]='b',
  // [1498]='c', [1499]='d', [1500]='e', [1501]='f', [1502]='g', [1503]='a', [1504]='b',
  // [1505]='c', [1506]='d', [1507]='e', [1508]='f', [1509]='g', [1510]='a', [1511]='b',
  // [1512]='c', [1513]='d', [1514]='e', [1515]='f', [1516]='g', [1517]='a', [1518]='b',
  // [1519]='c', [1520]='d', [1521]='e', [1522]='f', [1523]='g', [1524]='a', [1525]='b',
  // [1526]='c', [1527]='d', [1528]='e', [1529]='f', [1530]='g', [1531]='a', [1532]='b',
  // [1533]='c', [1534]='d', [1535]='e', [1536]='f', [1537]='g', [1538]='a', [1539]='b',
  // [1540]='c', [1541]='d', [1542]='e', [1543]='f', [1544]='g', [1545]='a', [1546]='b',
  // [1547]='c', [1548]='d', [1549]='e', [1550]='f', [1551]='g', [1552]='a', [1553]='b',
  // [1554]='c', [1555]='d', [1556]='e', [1557]='f', [1558]='g', [1559]='a', [1560]='b',
  // [1561]='c', [1562]='d', [156
```

Double click **MAIN.H** and insert Define:

```
#define MAX_SONG_LENGTH 400
```

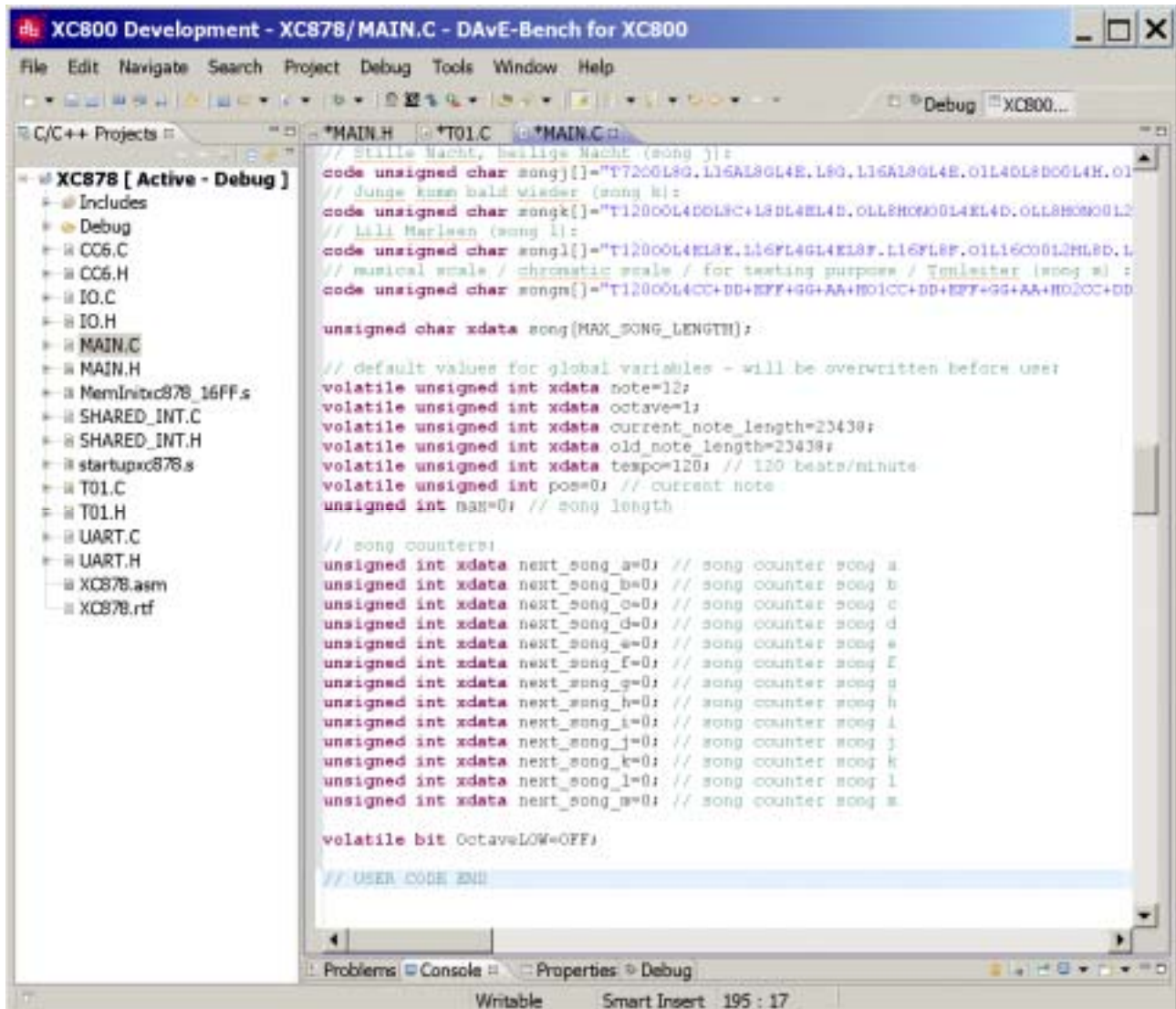


Double click **MAIN.C** and **insert** Global Variables:

```
// 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;
```

```

// Stille Nacht, heilige Nacht (song j):
code unsigned char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L40L8D00L4H.O1
// Junge kuss bald wieder (song k):
code unsigned char songk[]="T12000L4D0L8C+L8DL4EL4B.O1L8H0M00L4EL4D.O1L8H0M00L2
// Lili Marleen (song l):
code unsigned char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16C00L2HL8D.L
// musical scale / chromatic scale / for testing purpose / Tonleiter (song m):
code unsigned char songm[]="T12000L4CC+DD+EFF+GG+AA+H01CC+DD+EFF+GG+AA+H02CC+DD

unsigned char xdata song(MAX_SONG_LENGTH);

// 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 nag=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;

// USER CODE END
  
```



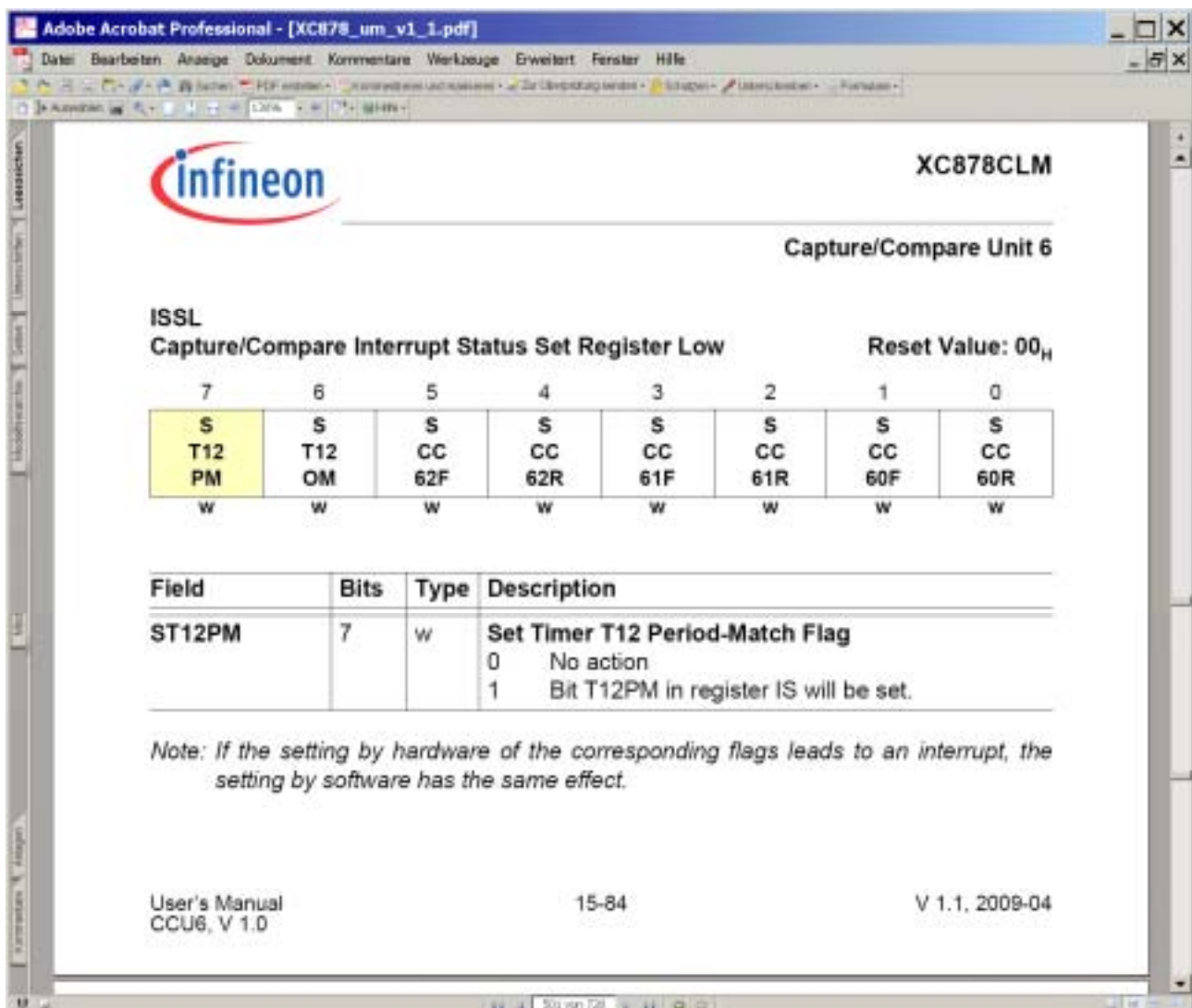

Note:

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).



Infineon **XC878CLM**

Capture/Compare Unit 6

ISSL
Capture/Compare Interrupt Status Set Register Low Reset Value: 00_H

7	6	5	4	3	2	1	0
S	S	S	S	S	S	S	S
T12	T12	CC	CC	CC	CC	CC	CC
PM	OM	62F	62R	61F	61R	60F	60R
W	W	W	W	W	W	W	W

Field	Bits	Type	Description
ST12PM	7	w	Set Timer T12 Period-Match Flag 0 No action 1 Bit T12PM in register IS will be set.

Note: If the setting by hardware of the corresponding flags leads to an interrupt, the setting by software has the same effect.

User's Manual 15-84 V 1.1, 2009-04
CCU6, V 1.0



Note:

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

Adobe Acrobat Professional - [XC878_um_v1_1.pdf]

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

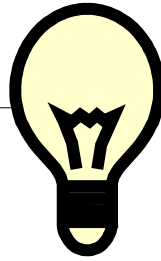
Address	Page 0	Page 1	Page 2	Page 3
9A _H	CC63SRL	CC63RL	T12MSELL	MCMOUTL
9B _H	CC63SRH	CC63RH	T12MSELH	MCMOUTH
9C _H	TCTR4L	T12PRL	IENL	ISL
9D _H	TCTR4H	T12PRH	IENH	ISH
9E _H	MCMOUTSL	T13PRL	INPL	PISEL0L
9F _H	MCMOUTSH	T13PRH	INPH	PISEL0H
A4 _H	ISRL	T12DTCL	ISSL	PISEL2
A5 _H	ISRH	T12DTCH	ISSH	
A6 _H	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

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.

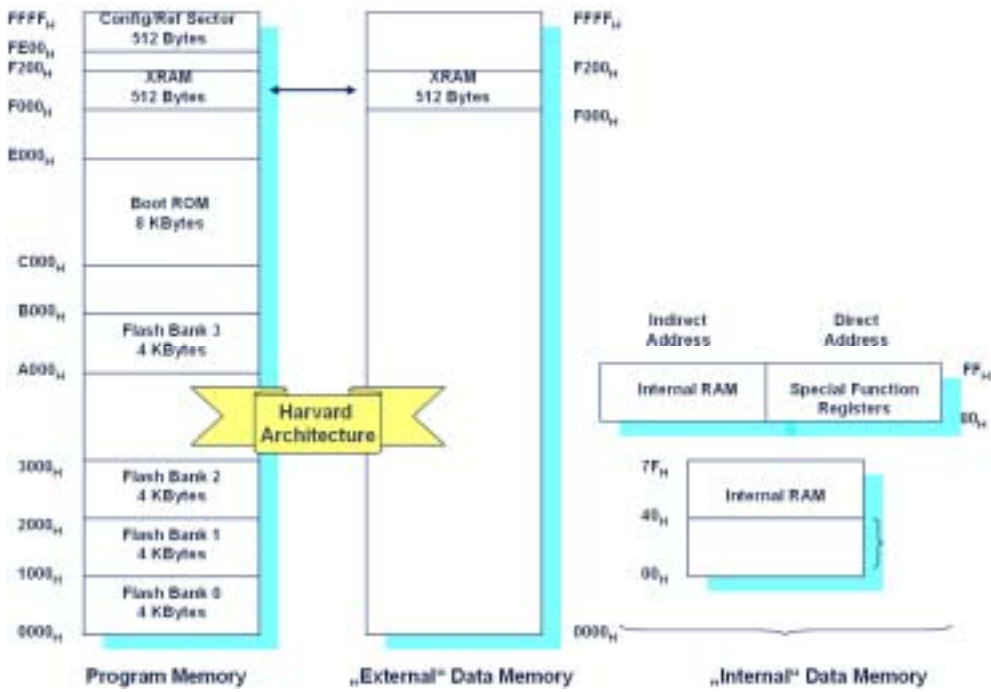
Therefore, we can use the following code sequence:

```
// 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
```

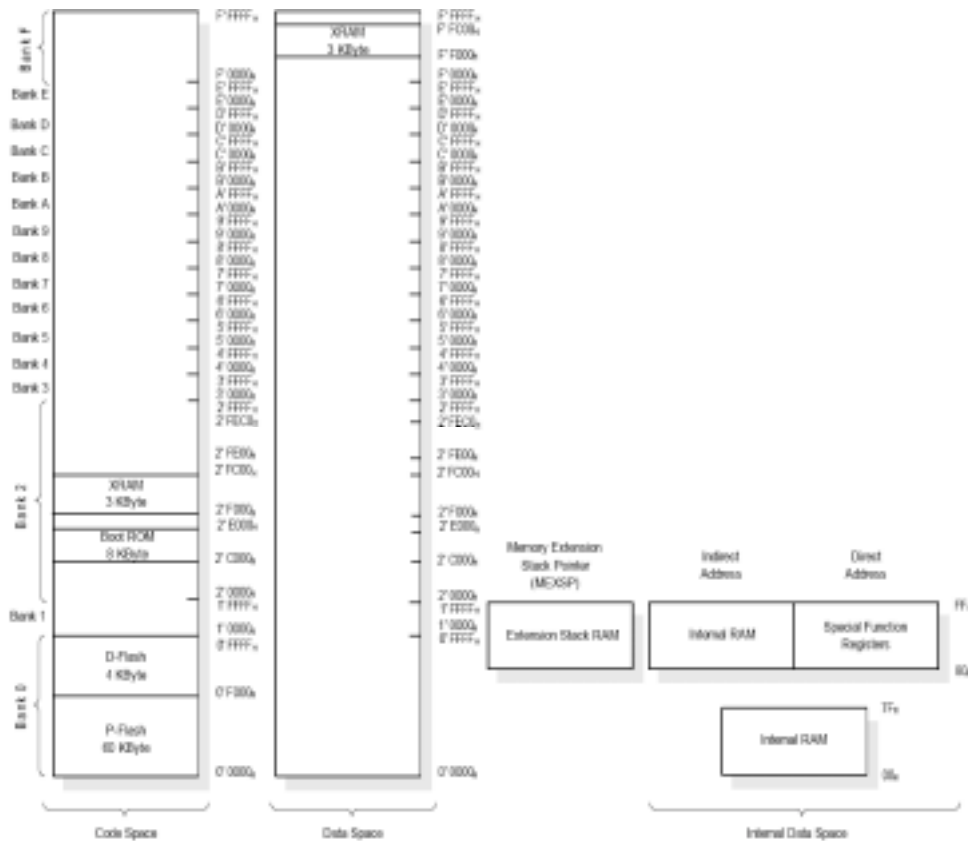


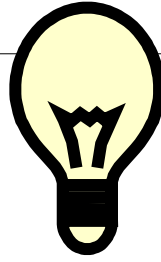
Note: Memory Organization:

From **8052/XC866**:



to **XC878**:



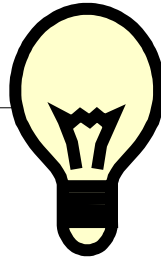


Address Extension:

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):

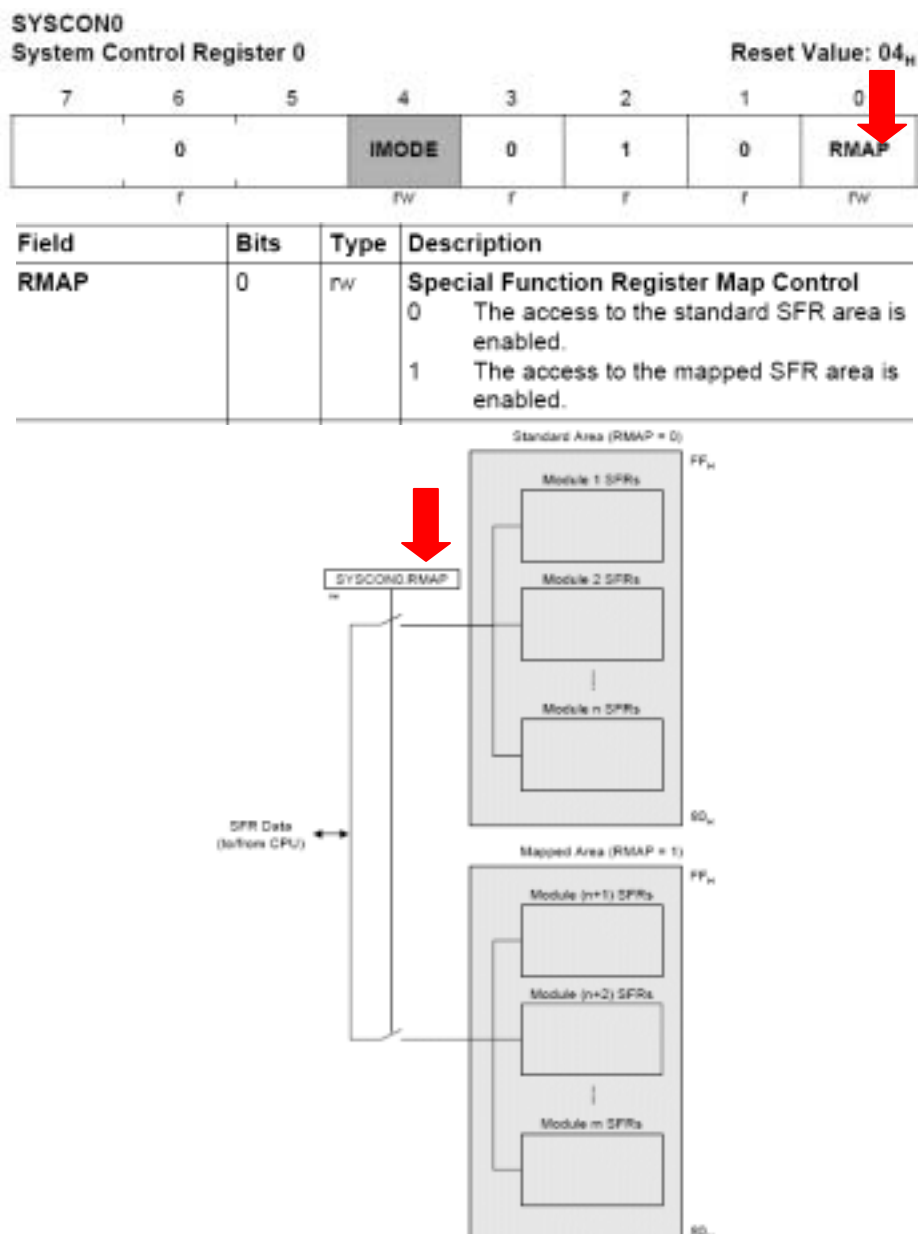
The SFR area is extended into two portions: the standard (non-mapped) SFR area and the mapped SFR area. Each portion supports the same address range 80_H to FF_H , extending the number of addressable SFRs to 256 Bytes.

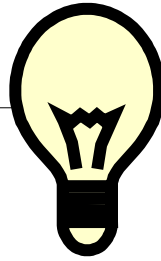
The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit RMAP in the system control register SYSCON0.

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.





Address Extension by Paging:

Note (Source: User's Manual):

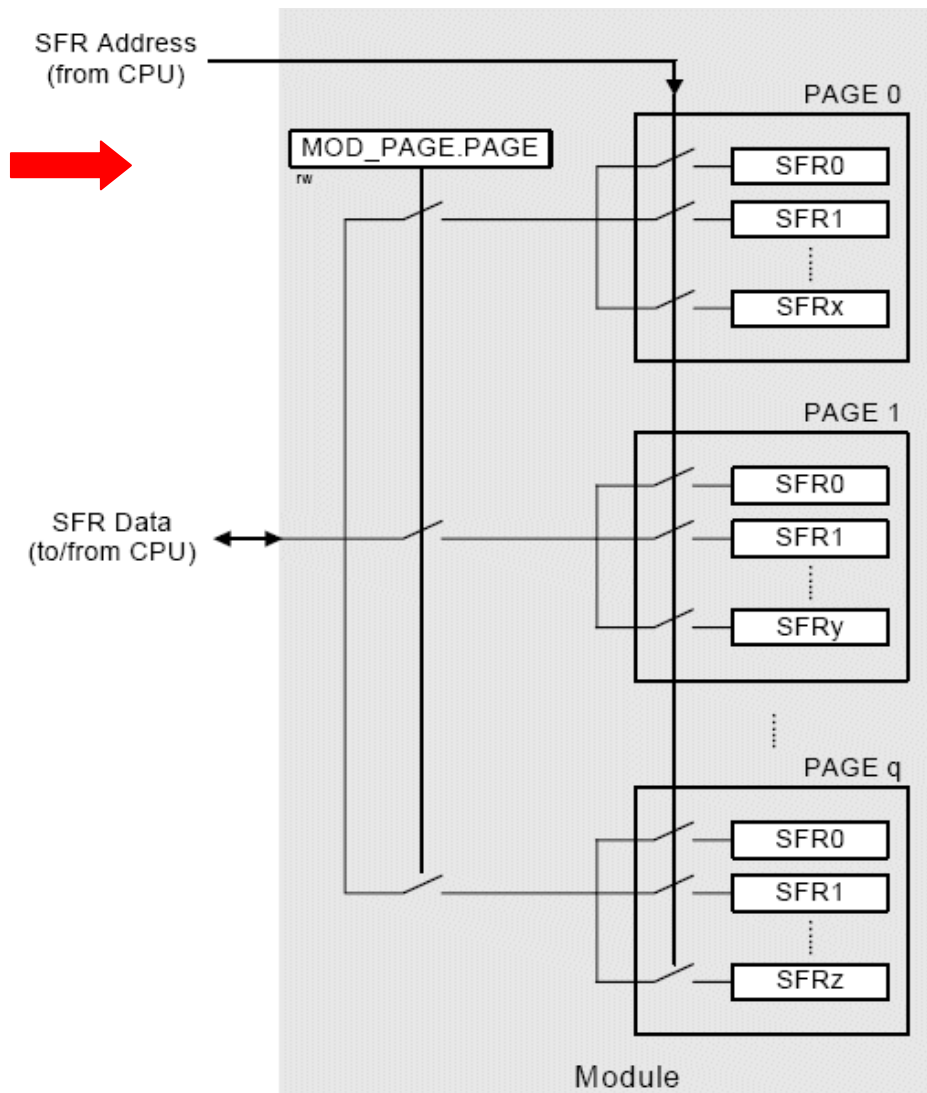
Address extension is further performed at the module level by paging. With the address extension by mapping, the XC8xx has a 256-SFR address range. However, this is still less than the total number of SFRs needed by the on-chip peripherals.

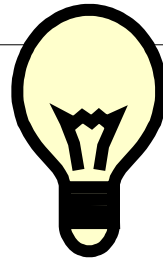
To meet this requirement, some peripherals (Parallel Ports, Analog-to-Digital Converter, Capture/Compare Unit 6, System Control Registers) have a built-in local address extension mechanism for increasing the number of addressable SFRs. The extended address range is not directly controlled by the CPU 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.

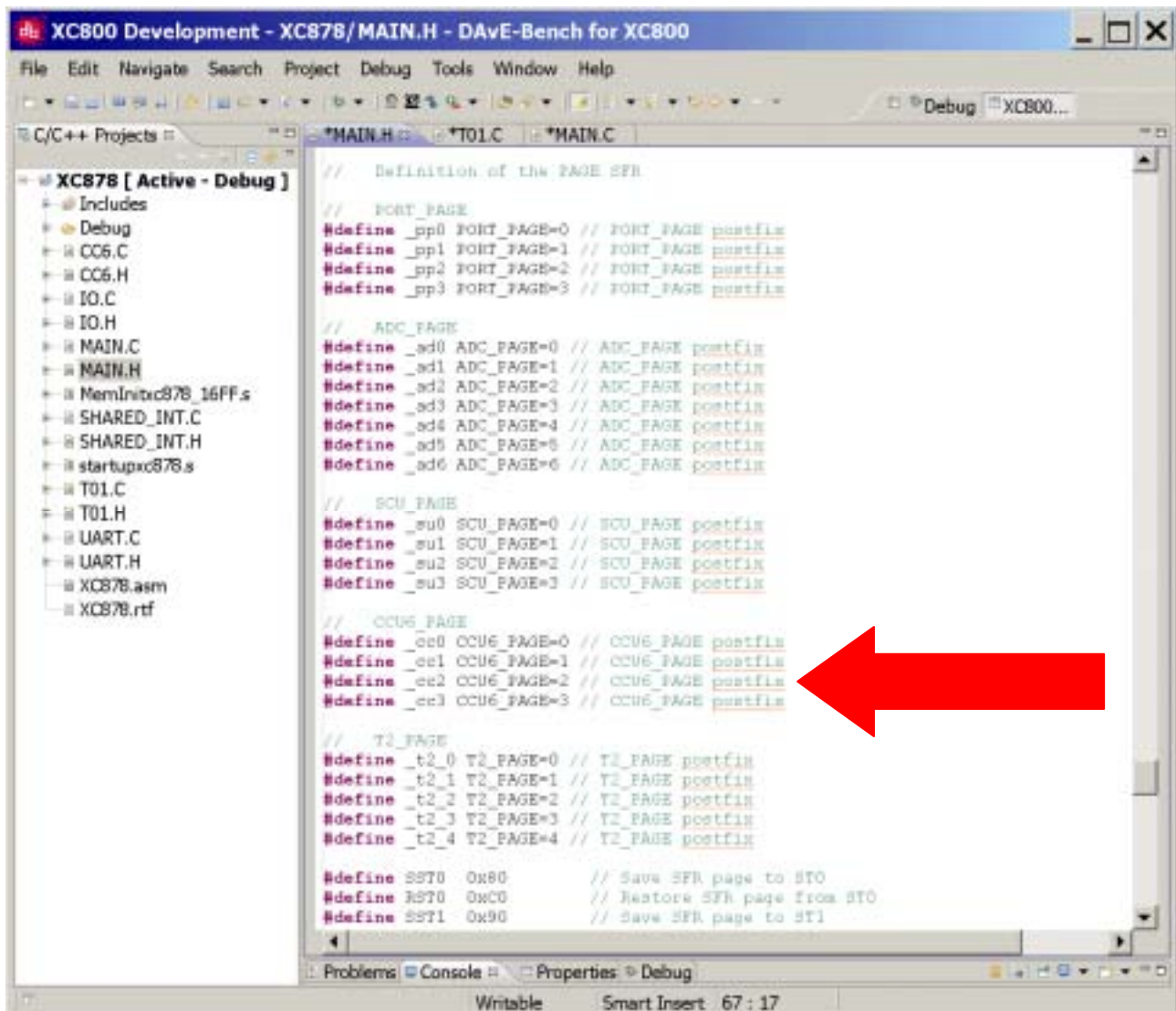
Note (Source: Application Note AP08053):

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:



```

// Definition of the PAGE SFR

// PORT_PAGE
#define _pp0 PORT_PAGE=0 // PORT_PAGE postfin
#define _pp1 PORT_PAGE=1 // PORT_PAGE postfin
#define _pp2 PORT_PAGE=2 // PORT_PAGE postfin
#define _pp3 PORT_PAGE=3 // PORT_PAGE postfin

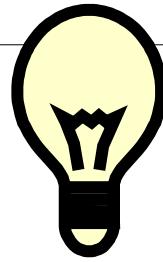
// ADC_PAGE
#define _ad0 ADC_PAGE=0 // ADC_PAGE postfin
#define _ad1 ADC_PAGE=1 // ADC_PAGE postfin
#define _ad2 ADC_PAGE=2 // ADC_PAGE postfin
#define _ad3 ADC_PAGE=3 // ADC_PAGE postfin
#define _ad4 ADC_PAGE=4 // ADC_PAGE postfin
#define _ad5 ADC_PAGE=5 // ADC_PAGE postfin
#define _ad6 ADC_PAGE=6 // ADC_PAGE postfin

// SCU_PAGE
#define _su0 SCU_PAGE=0 // SCU_PAGE postfin
#define _su1 SCU_PAGE=1 // SCU_PAGE postfin
#define _su2 SCU_PAGE=2 // SCU_PAGE postfin
#define _su3 SCU_PAGE=3 // SCU_PAGE postfin

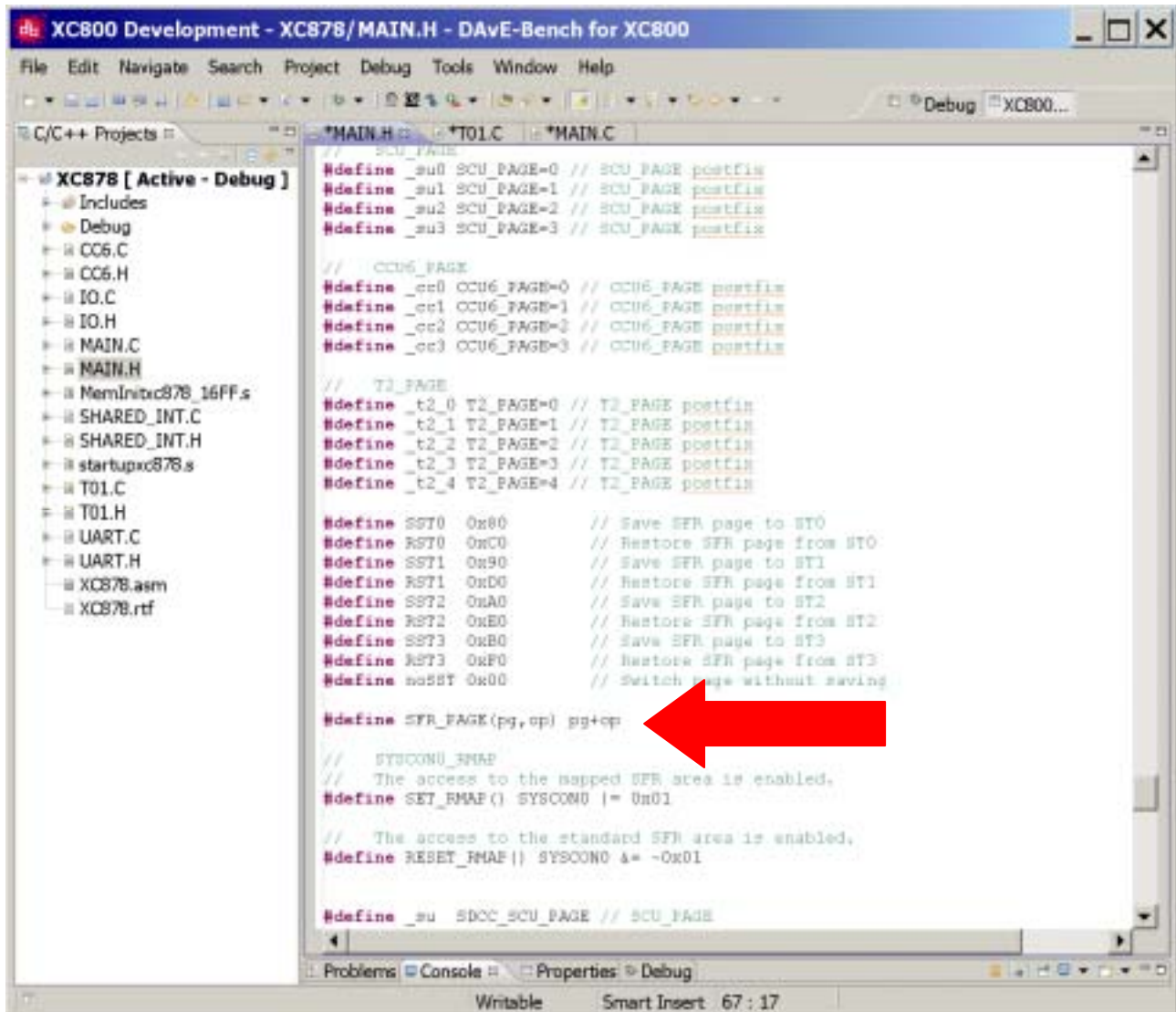
// CCUG_PAGE
#define _cc0 CCUG_PAGE=0 // CCUG_PAGE postfin
#define _cc1 CCUG_PAGE=1 // CCUG_PAGE postfin
#define _cc2 CCUG_PAGE=2 // CCUG_PAGE postfin
#define _cc3 CCUG_PAGE=3 // CCUG_PAGE postfin

// T2_PAGE
#define _t2_0 T2_PAGE=0 // T2_PAGE postfin
#define _t2_1 T2_PAGE=1 // T2_PAGE postfin
#define _t2_2 T2_PAGE=2 // T2_PAGE postfin
#define _t2_3 T2_PAGE=3 // T2_PAGE postfin
#define _t2_4 T2_PAGE=4 // T2_PAGE postfin

#define SST0 0x80 // Save SFR page to ST0
#define RST0 0xC0 // Restore SFR page from ST0
#define SST1 0x90 // Save SFR page to ST1
    
```



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



```

XC800 Development - XC878/MAIN.H - 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

// SCU_PAGE
#define _su0 SCU_PAGE=0 // SCU_PAGE postfix
#define _su1 SCU_PAGE=1 // SCU_PAGE postfix
#define _su2 SCU_PAGE=2 // SCU_PAGE postfix
#define _su3 SCU_PAGE=3 // SCU_PAGE postfix

// CCU6_PAGE
#define _cc0 CCU6_PAGE=0 // CCU6_PAGE postfix
#define _cc1 CCU6_PAGE=1 // CCU6_PAGE postfix
#define _cc2 CCU6_PAGE=2 // CCU6_PAGE postfix
#define _cc3 CCU6_PAGE=3 // CCU6_PAGE postfix

// T2_PAGE
#define _t2_0 T2_PAGE=0 // T2_PAGE postfix
#define _t2_1 T2_PAGE=1 // T2_PAGE postfix
#define _t2_2 T2_PAGE=2 // T2_PAGE postfix
#define _t2_3 T2_PAGE=3 // T2_PAGE postfix
#define _t2_4 T2_PAGE=4 // T2_PAGE postfix

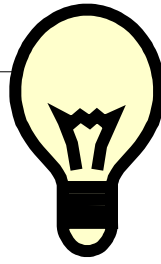
#define SST0 0x80 // Save SFR page to ST0
#define RST0 0xC0 // Restore SFR page from ST0
#define SST1 0x90 // Save SFR page to ST1
#define RST1 0xD0 // Restore SFR page from ST1
#define SST2 0xA0 // Save SFR page to ST2
#define RST2 0xE0 // Restore SFR page from ST2
#define SST3 0xB0 // Save SFR page to ST3
#define RST3 0xF0 // Restore SFR page from ST3
#define noSST 0x00 // Switch page without saving

#define SFR_PAGE(pg,op) pg+op

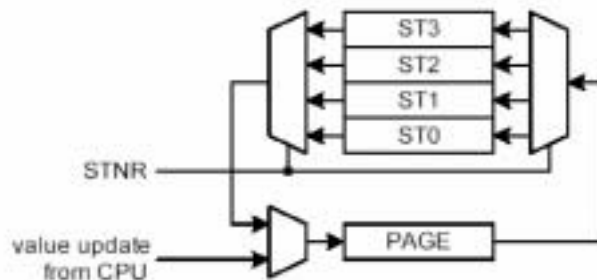
// SYSCON0_RMAP
// The access to the mapped SFR area is enabled.
#define SET_RMAP() SYSCON0 |= 0x01

// The access to the standard SFR area is enabled.
#define RESET_RMAP() SYSCON0 &= ~0x01

#define _su SDOC_SCU_PAGE // SCU_PAGE
  
```



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

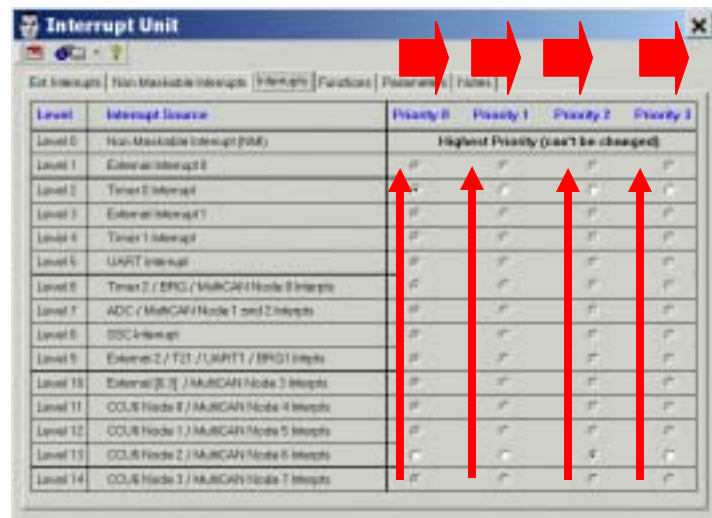


Note (Source: Application Note AP08053):

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.

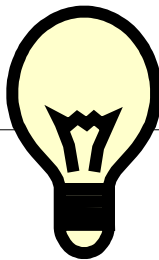
This leaves us with five interrupt priorities and four storage containers.

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.

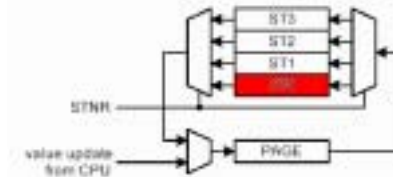
The ISR may also call functions that could modify page registers. If these functions are shared by ISRs of different priority levels, then these functions can be interrupted. It is therefore necessary to save and restore the page registers that are modified in these functions. In such cases, the stack should be used as a storage container.

In summary:

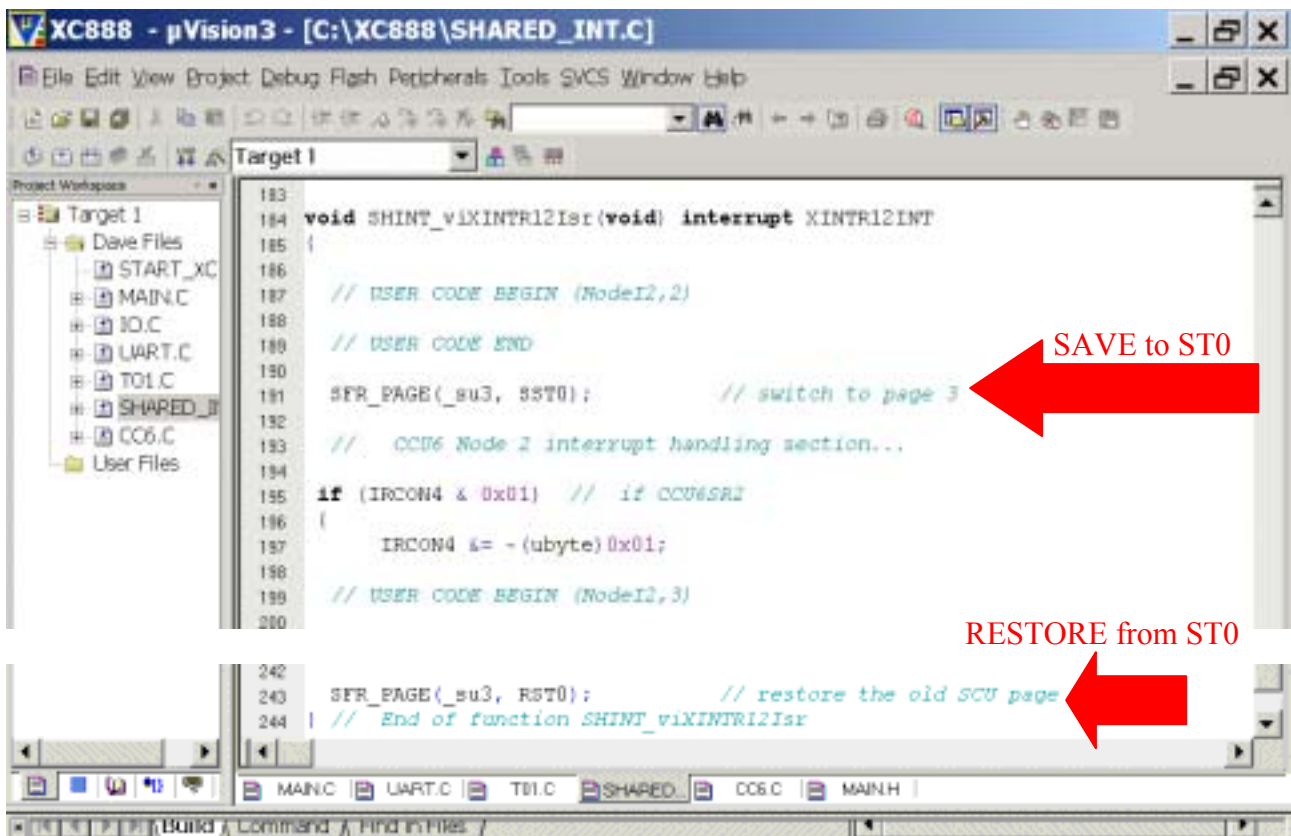
- All the page registers modified in an ISR must be saved.
- The storage container should be unique for each interrupt priority.
- The storage numbers within the ISRs must be changed accordingly when the interrupt priorities are changed.
- No storage container is necessary for the main level.
- Stack 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:



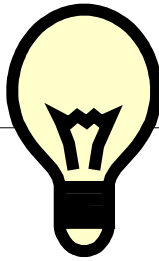
Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input checked=""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



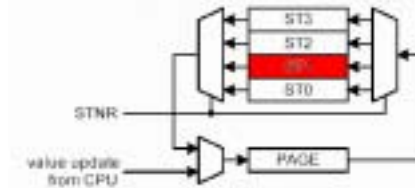
```

183
184 void SHINT_vIXINTR12Isr(void) interrupt XINTR12INT
185 {
186     // USER CODE BEGIN (NodeI2,2)
187
188     // USER CODE END
189
190     SFR_PAGE(_su3, SST0);           // switch to page 3
191
192     // CCU6 Node 2 interrupt handling section...
193
194     if (IRCON4 & 0x01) // if CCU6SA2
195     {
196         IRCON4 &= ~(ubyte)0x01;
197
198         // USER CODE BEGIN (NodeI2,3)
199
200
201
202
203
204     SFR_PAGE(_su3, RST0);           // restore the old SCU page
205     // End of function SHINT_vIXINTR12Isr

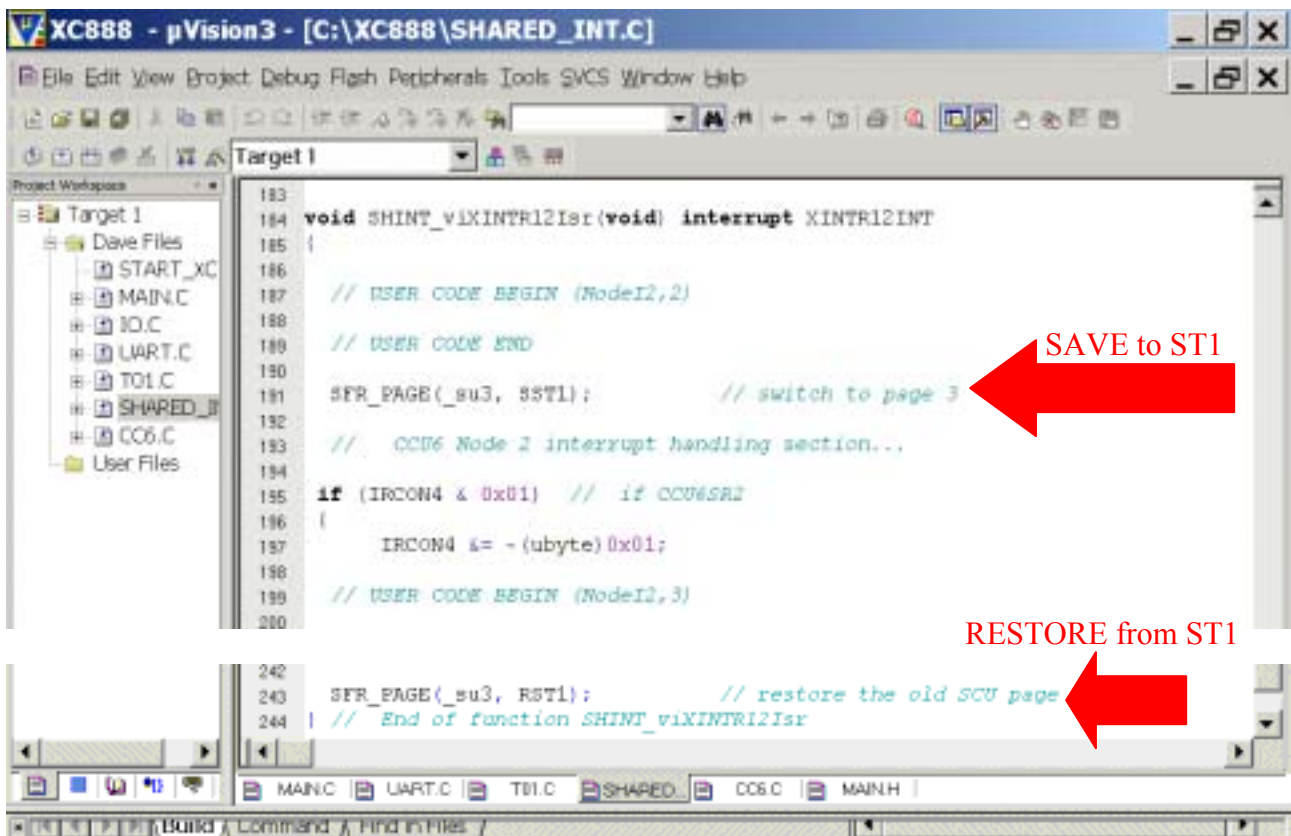
```

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

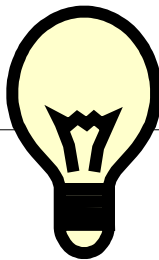


Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input checked=""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

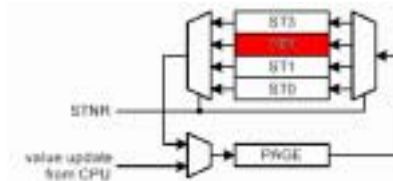


```

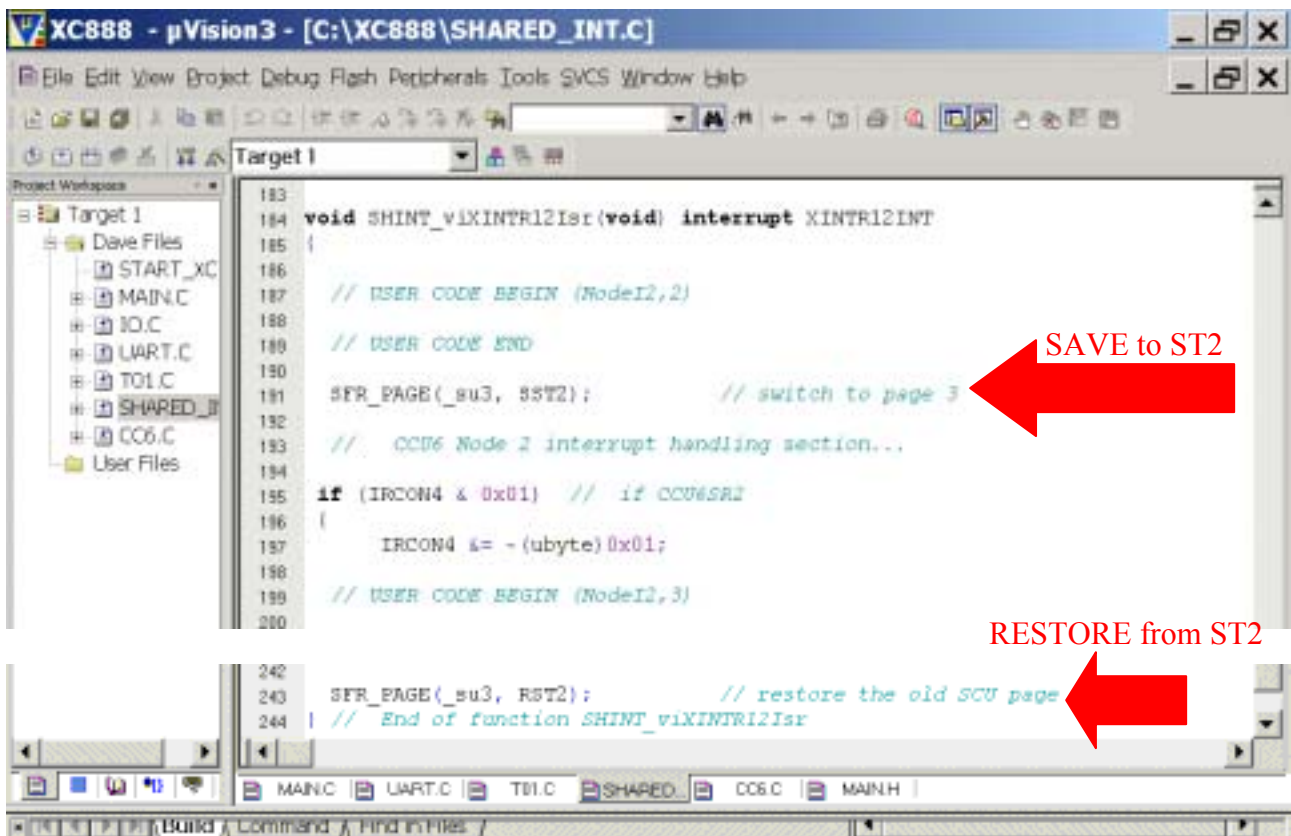
183
184 void SHINT_vIXINTR12Isr(void) interrupt XINTR12INT
185 {
186     // USER CODE BEGIN (NodeI2,2)
187
188     // USER CODE END
189
190     SFR_PAGE(_su3, SST1);           // switch to page 3
191
192     // CCU6 Node 2 interrupt handling section...
193
194     if (IRCON4 & 0x01) // if CCU6SA2
195     {
196         IRCON4 &= ~(ubyte)0x01;
197
198         // USER CODE BEGIN (NodeI2,3)
199
200
201
202
203     SFR_PAGE(_su3, RST1);           // restore the old SCU page
204     // End of function SHINT_vIXINTR12Isr
  
```

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



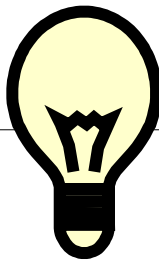
Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input checked=""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



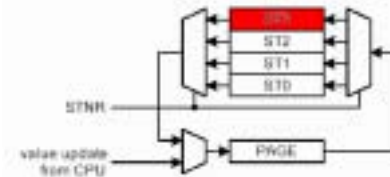
```

183
184 void SHINT_vIXINTR12Isr(void) interrupt XINTR12INT
185 {
186     // USER CODE BEGIN (NodeI2,2)
187
188     // USER CODE END
189
190     SFR_PAGE(_su3, SST2);           // switch to page 3
191
192     // CCU6 Node 2 interrupt handling section...
193
194     if (IRCON4 & 0x01) // if CCU6SA2
195     {
196         IRCON4 &= ~(ubyte)0x01;
197
198         // USER CODE BEGIN (NodeI2,3)
199
200
201
202
203     SFR_PAGE(_su3, RST2);           // restore the old SCU page
204     // End of function SHINT_vIXINTR12Isr

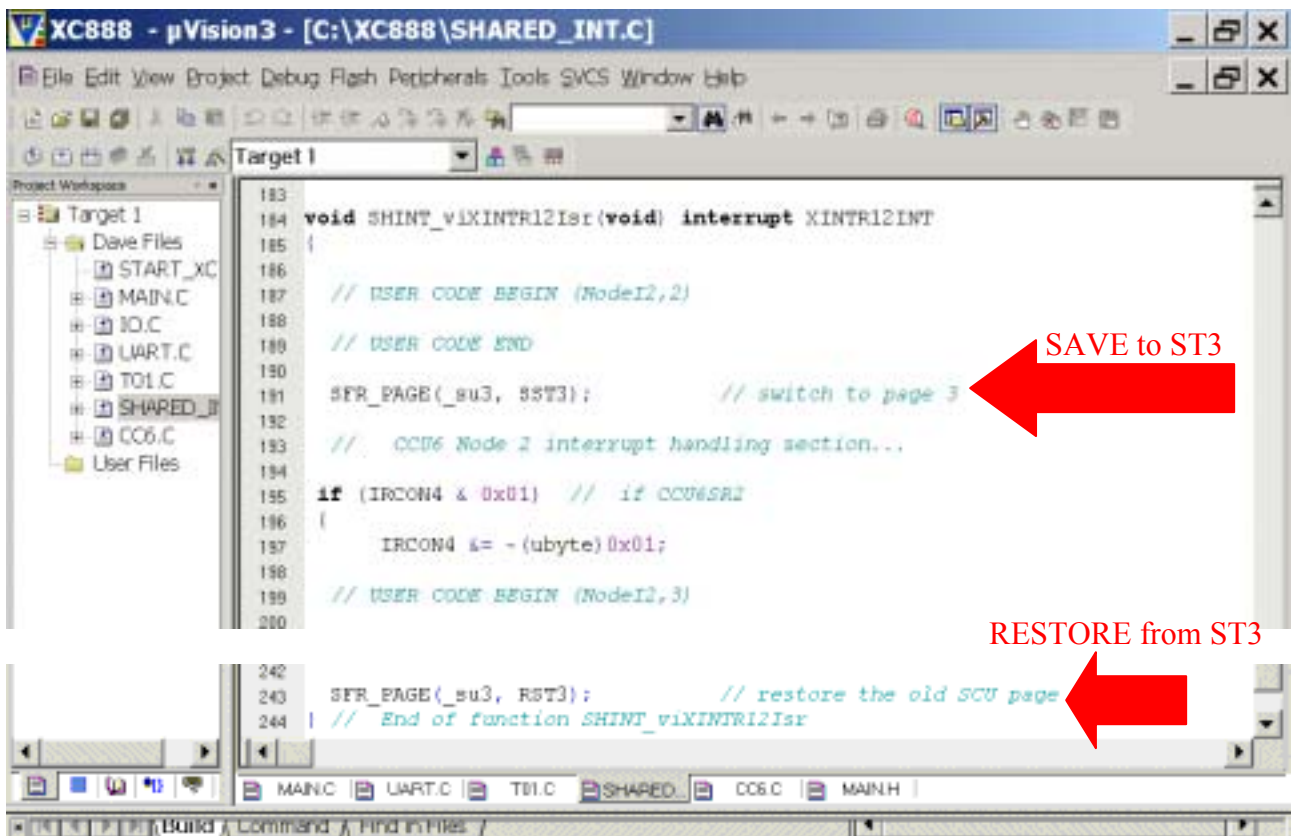
```



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

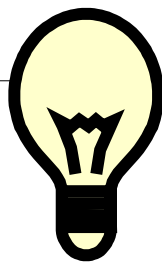


Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input checked=""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

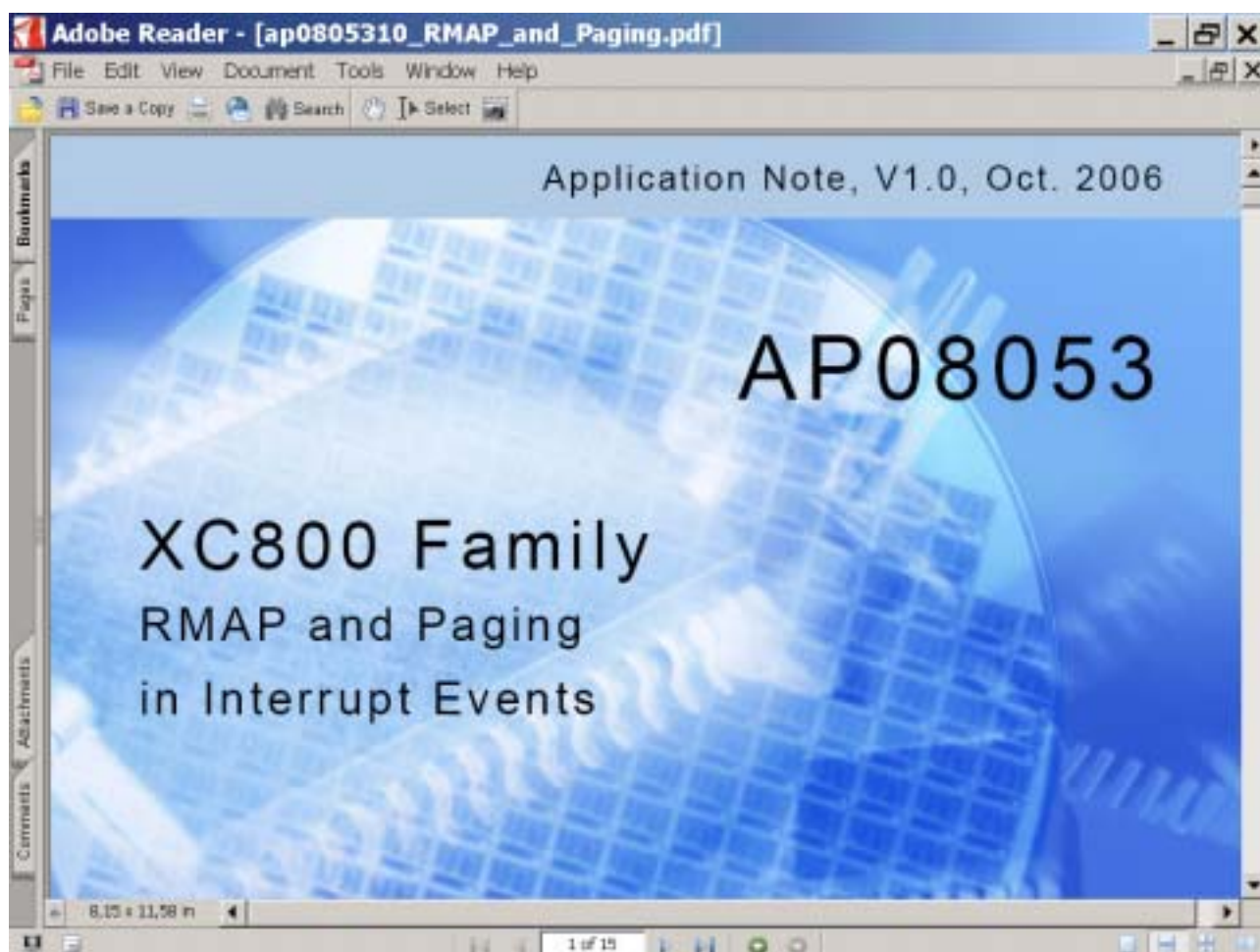


```

183
184 void SHINT_vIXINTR12Isr(void) interrupt XINTR12INT
185 {
186     // USER CODE BEGIN (NodeI2,2)
187     // USER CODE END
188     SFR_PAGE(_su3, SST3); // switch to page 3
189     // CCU6 Node 2 interrupt handling section...
190
191     if (IRCON4 & 0x01) // if CCU6SA2
192     {
193         IRCON4 &= ~(ubyte)0x01;
194     }
195     // USER CODE BEGIN (NodeI2,3)
196
242
243     SFR_PAGE(_su3, RST3); // restore the old SCU page
244     // End of function SHINT_vIXINTR12Isr
  
```



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);  
}
```




```

// USER CODE BEGIN (MAIN_Main,1)
char input [void]
{
    char in=' ';
    do
    {
        printf_small(question);
        while (!RI);
        RI=0;
        in = SEUS;
    }while ( ! (in=='a'&&in!="") );
    return in;
}

void play_song(void)
{
    max=0;
    if (next_song_a && ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
        printf_small("\nplaying: Maus am Meer\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 - Lary 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-ha-butzenmann\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 /");
    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_PASE(_cc2,no887); // switch to OC06_PASE=2 without saving !!!
        OC06_ISSL = OC06_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 (SEUS=='z')
                printf_small("Song aborted.\n");
            else
                printf_fast_f("End of the song reached (pos=%5u of max%5u).\n",pos,max);
        }
    }
}
// USER CODE END

void main(void)

```

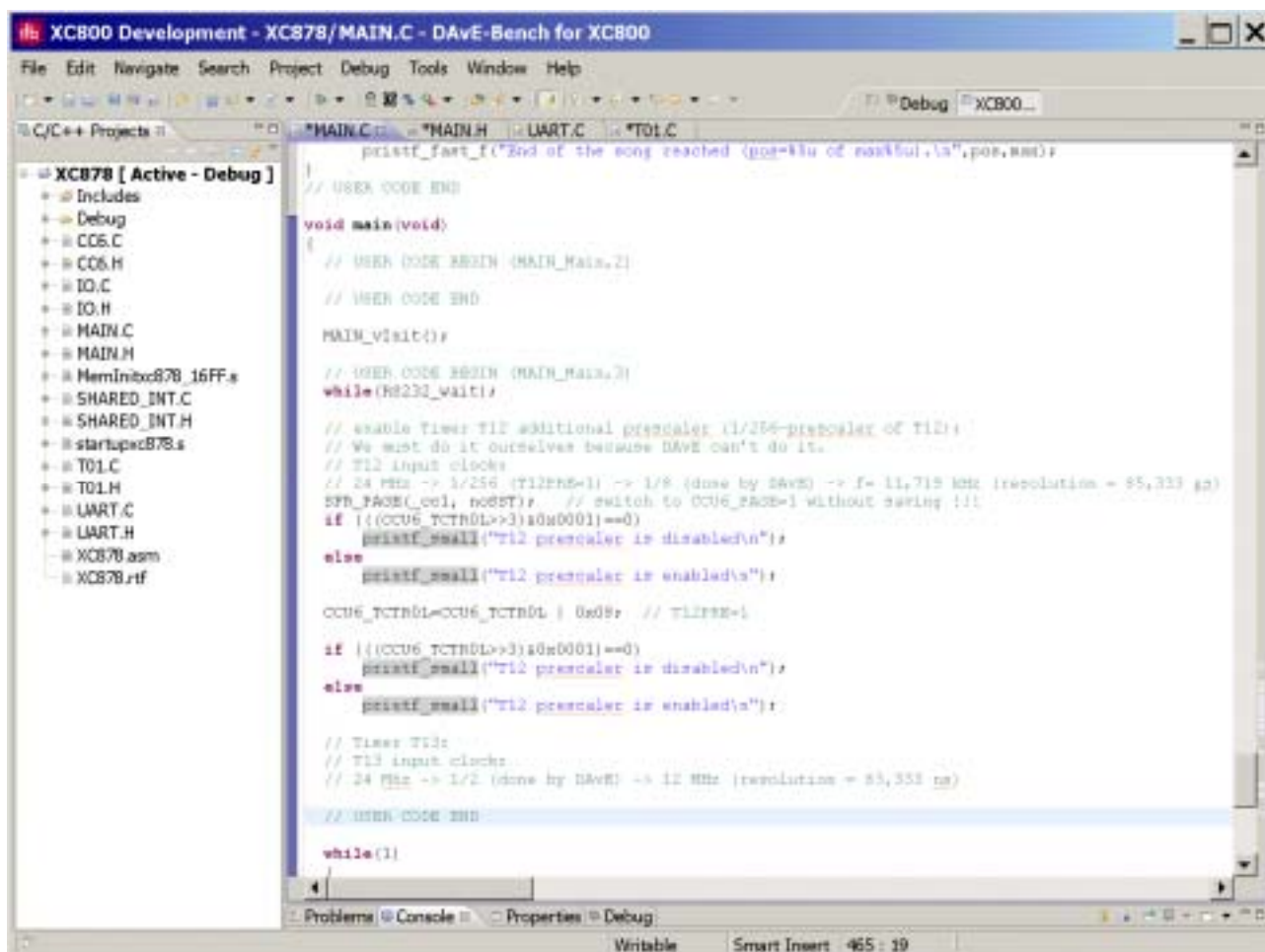

Double click **MAIN.C** and **insert** code (set the T12PRE bit):

```
// 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
  startupxc878.s
  T01.C
  T01.H
  UART.C
  UART.H
  XC878.asm
  XC878.rtf
  *MAIN.C
  *MAIN.H
  *UART.C
  *T01.C

printf_fast_f("End of the song reached (pos=43u of main5ul.is",pos,43u);
// USER CODE END

void main(void)
{
    // USER CODE BEGIN (MAIN_Main,2)
    // USER CODE END

    MAIN_visit();

    // USER CODE BEGIN (MAIN_Main,3)
    while(RS232_wait);

    // enable timer T12 additional prescaler (1/256-prescaler of T12);
    // We must do it ourselves because DAve can't do it.
    // T12 input clocks
    // 24 MHz -> 1/256 (T12PRE=1) -> 1/8 (done by DAve) -> f= 11,718 MHz (resolution = 85,333 ns)
    SFR_PAGE(0u, noBST); // switch to OC6_PAGE=1 without saving !!!
    if ((OC6_TCTR0L>>3)&0x0001)==0)
        printf_small("T12 prescaler is disabled\n");
    else
        printf_small("T12 prescaler is enabled\n");

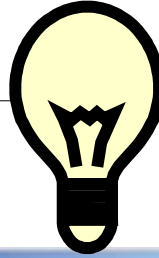
    OC6_TCTR0L=OC6_TCTR0L | 0x08; // T12PRE=1

    if ((OC6_TCTR0L>>3)&0x0001)==0)
        printf_small("T12 prescaler is disabled\n");
    else
        printf_small("T12 prescaler is enabled\n");

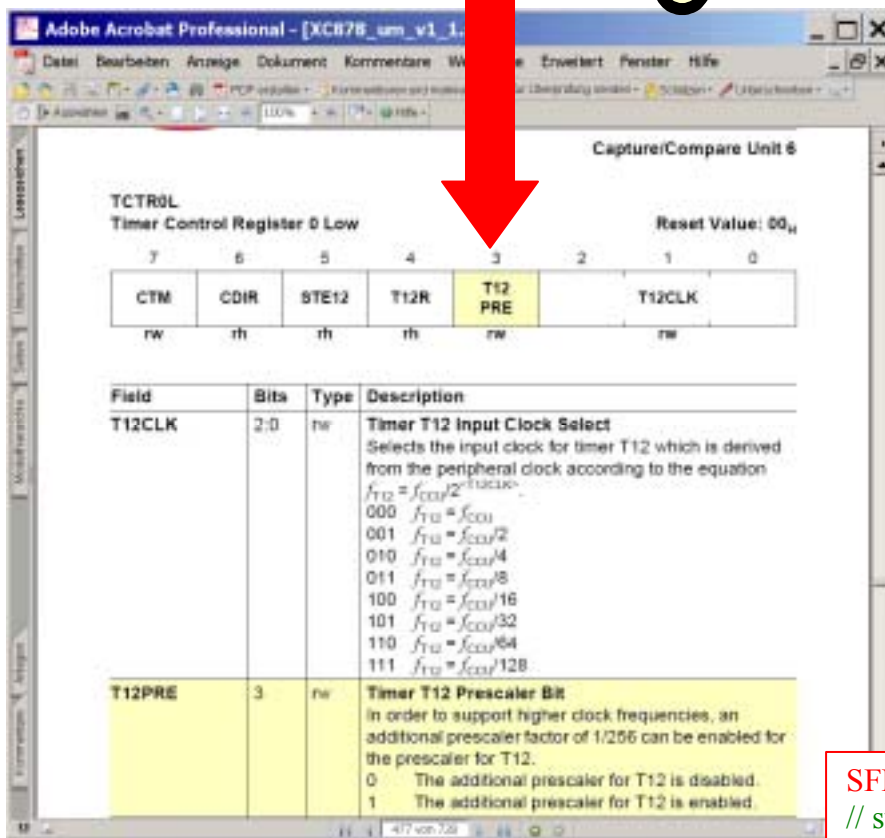
    // Timer T12:
    // T12 input clocks
    // 24 MHz -> 1/2 (done by DAve) -> 12 MHz (resolution = 83,333 ns)

    // USER CODE END

    while(1)
    
```



T12PRE=1:

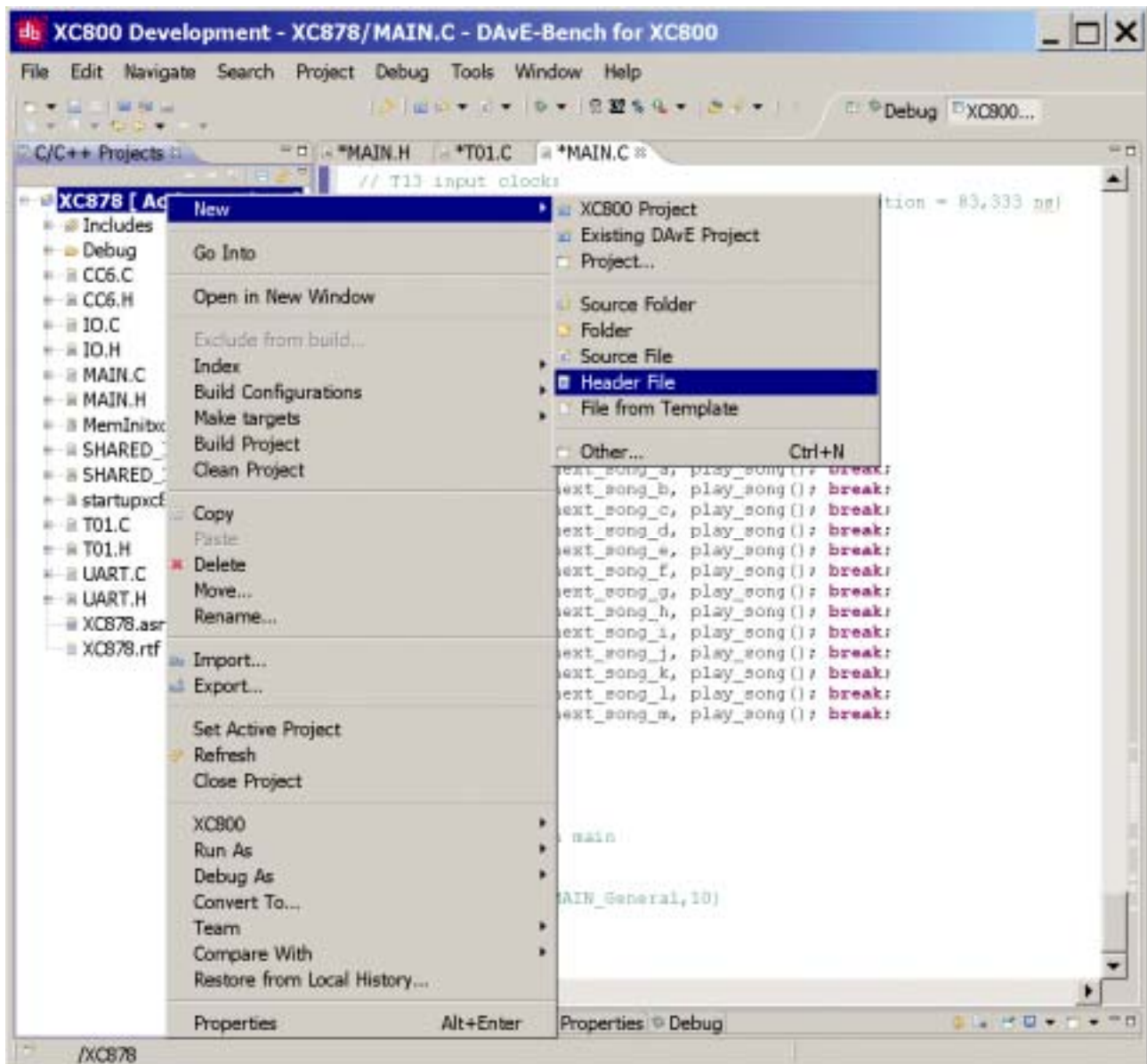


SFR_PAGE(_cc1, noSST);
// switch to CCU6_PAGE=1
// without saving !!!

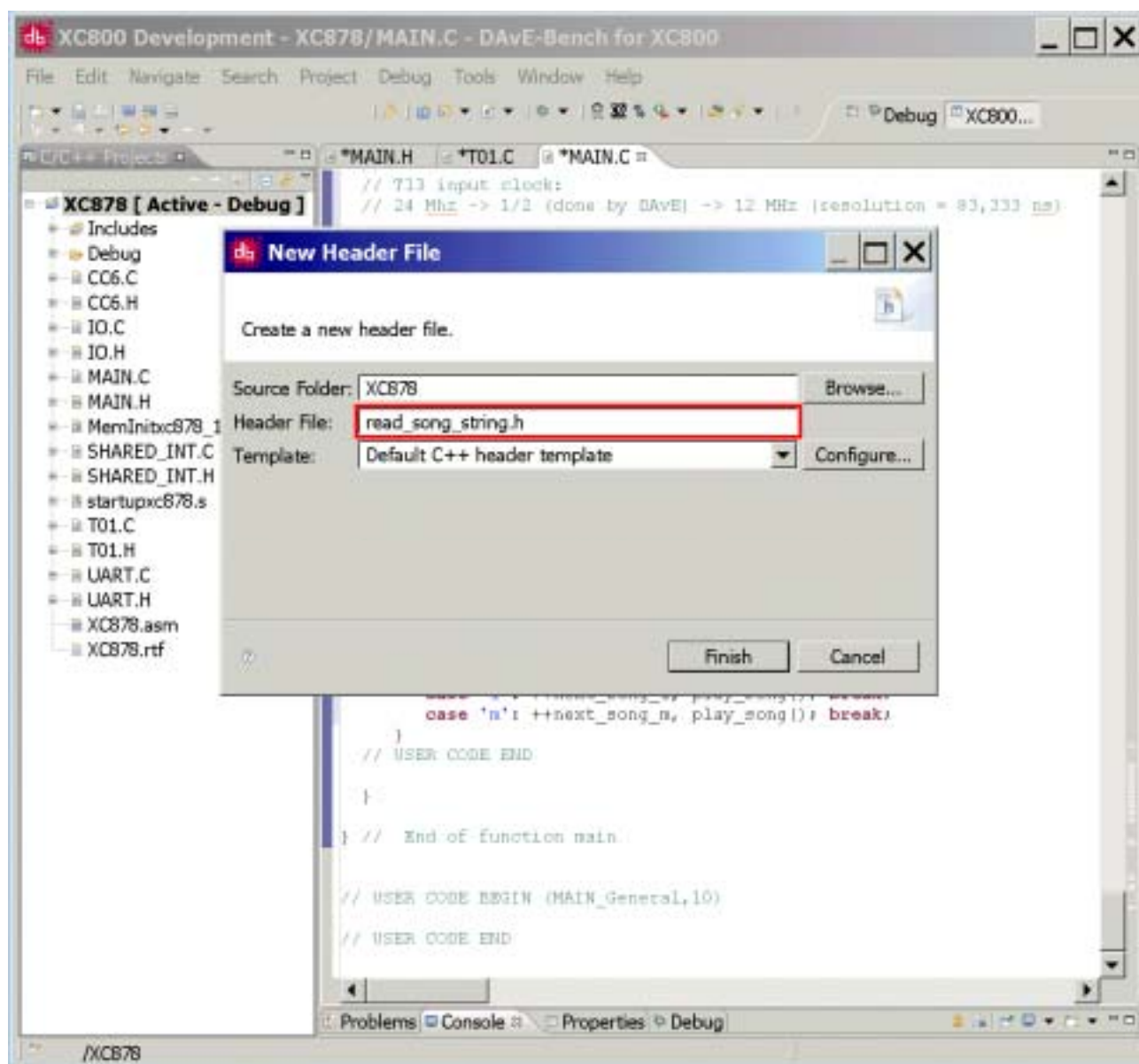
Table 15-4 SFR Address List for Pages 0-3

Address	Page 0	Page 1	Page 2	Page 3
9A _H	CC63SRL	CC63RL	T12MSEL	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



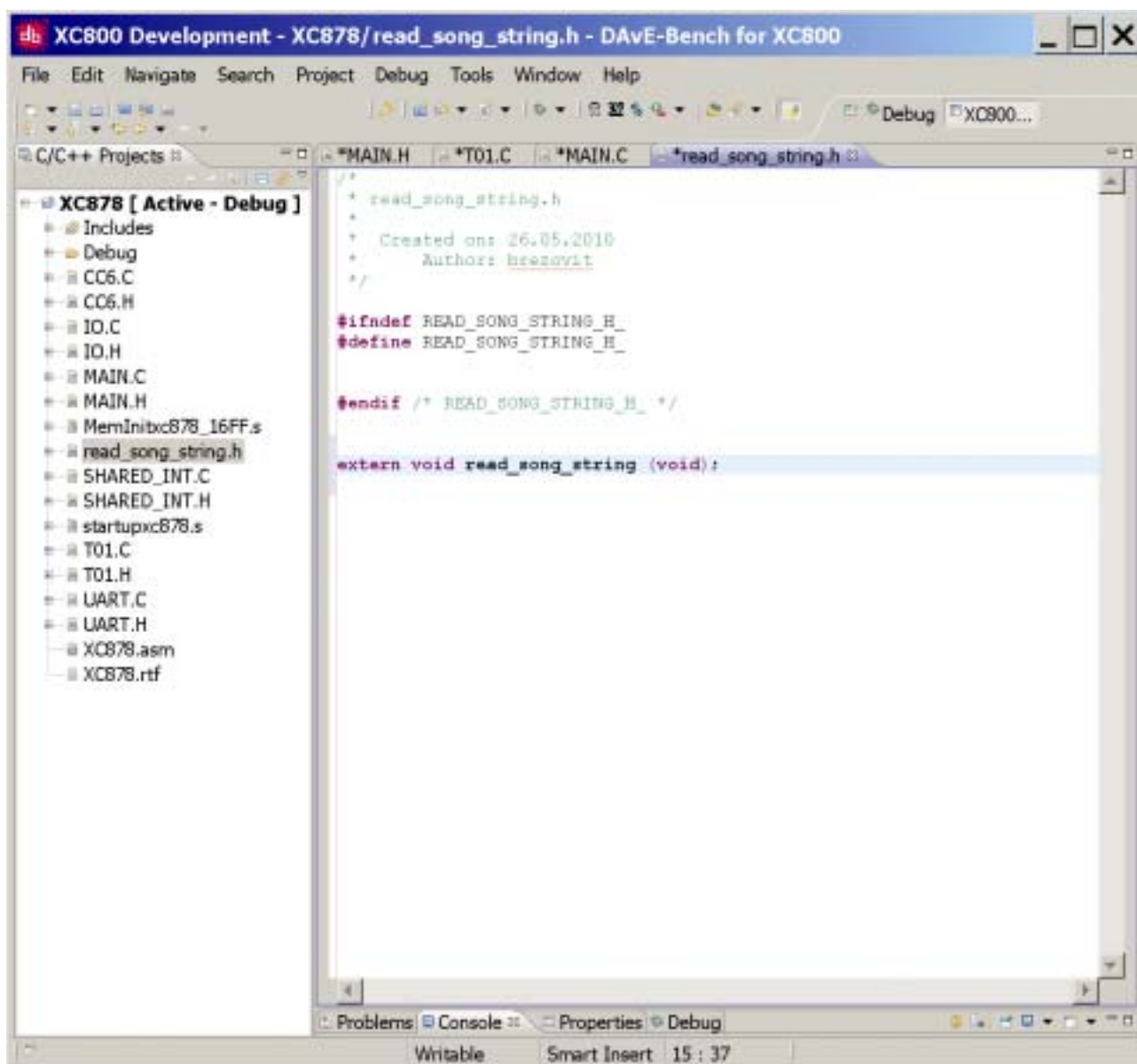
New Header File: Header File: insert: read_song_string.h



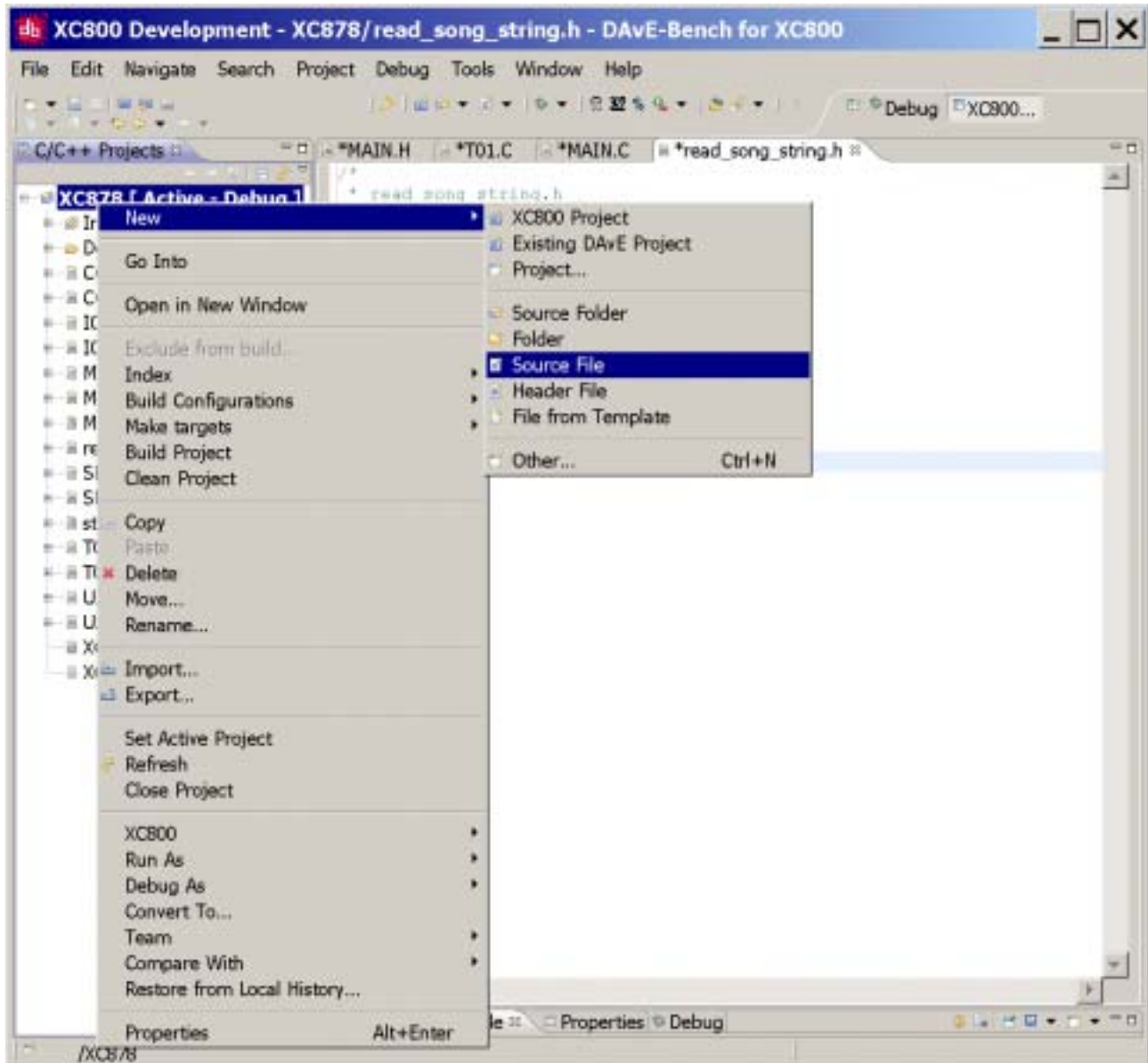
Click Finish

Insert:

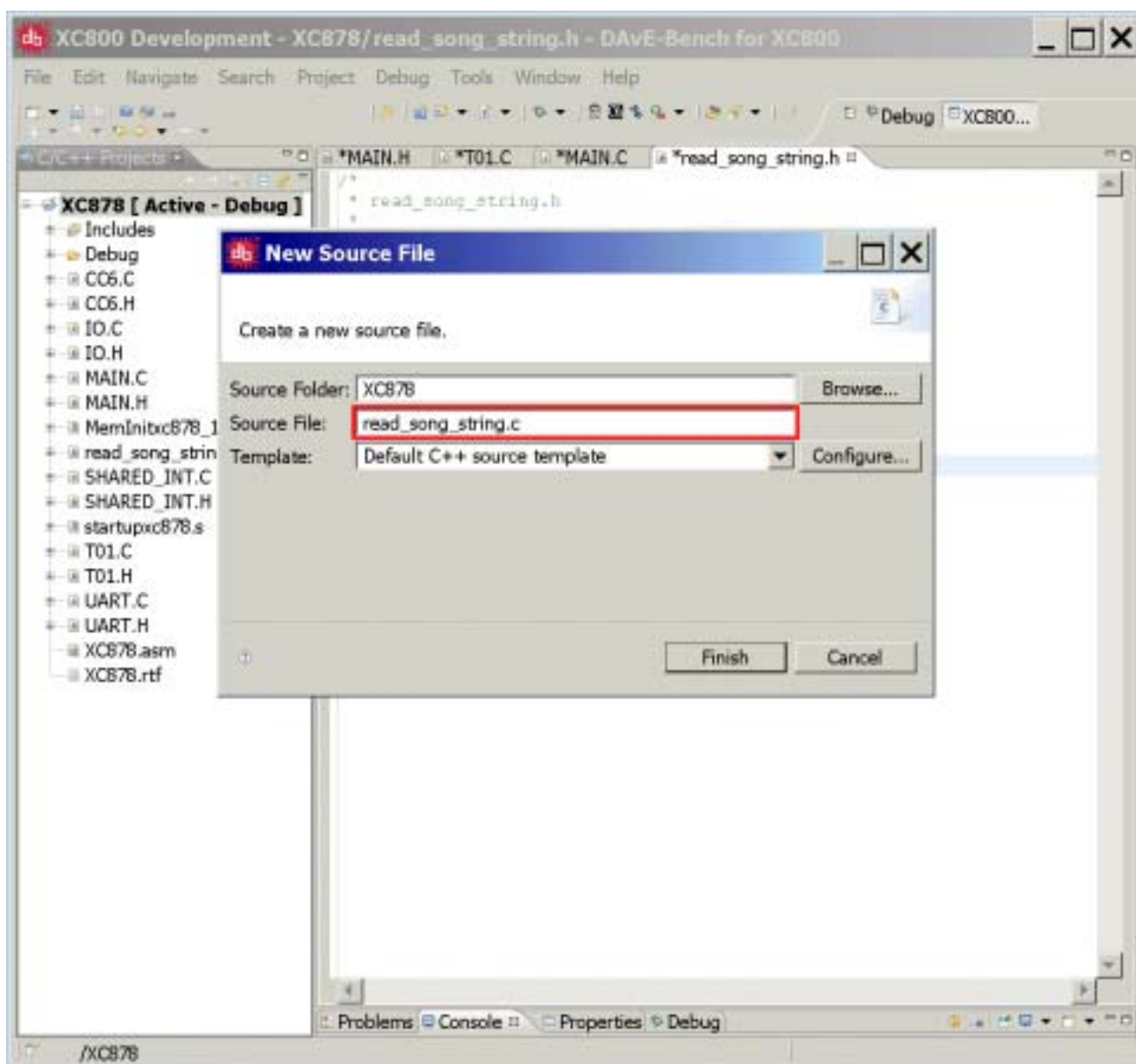
```
extern void read_song_string (void);
```



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++;
}

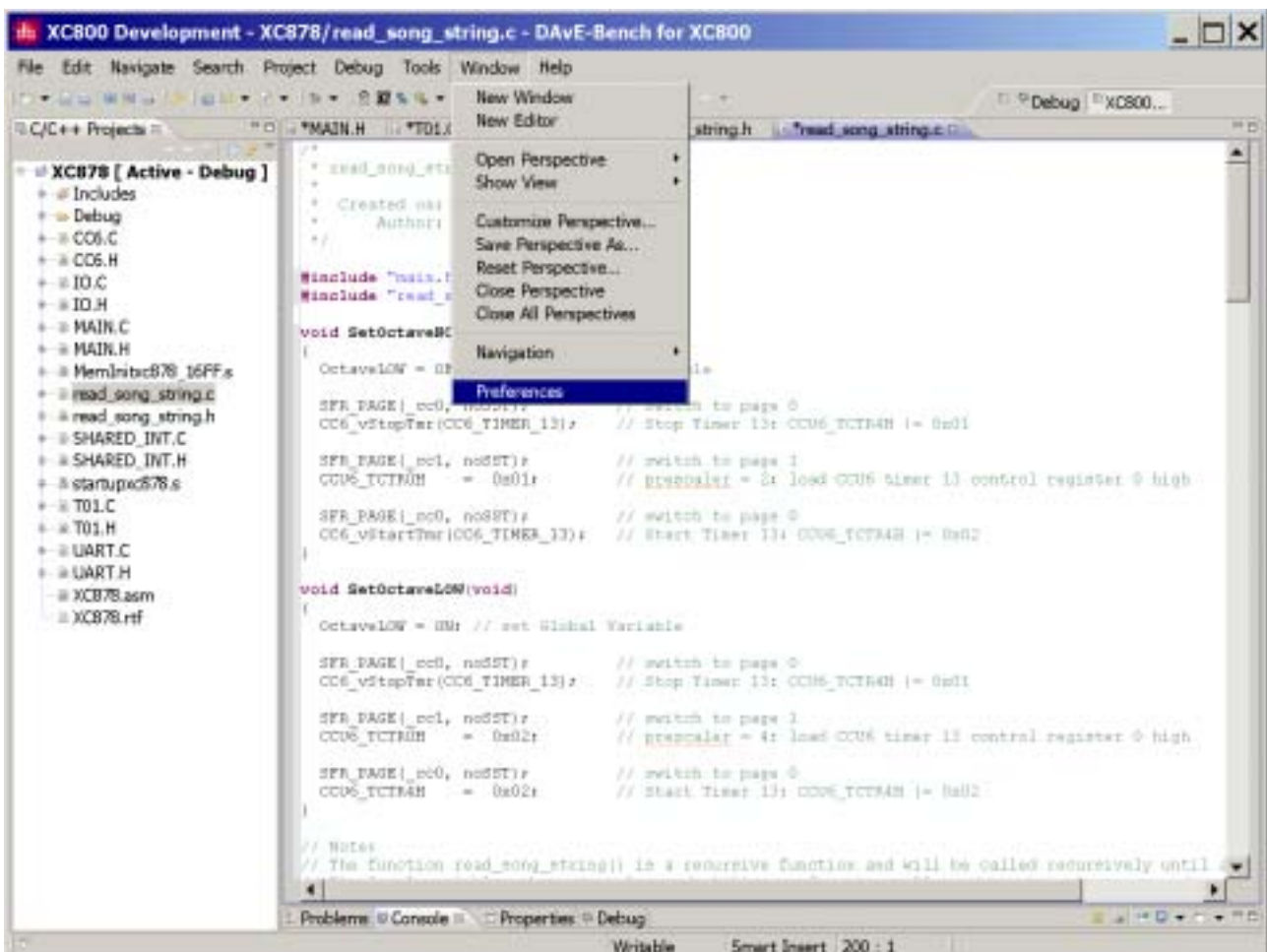
if (pos==max) pos++;

} /* end read_song_string */
```




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

Window - Preferences

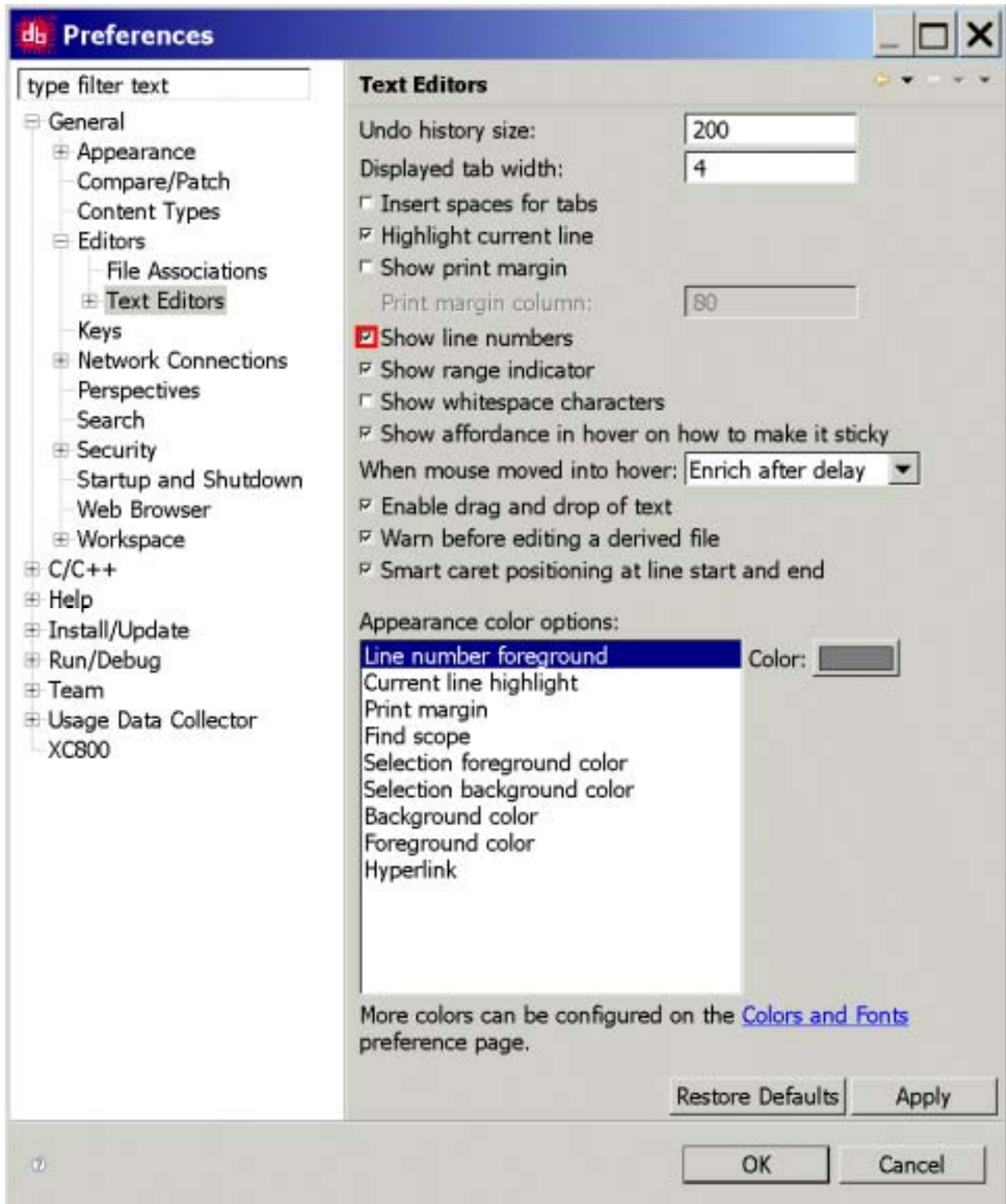




Note: Now we want to see line numbers (page 1/2):

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
- CO6.C
- CO6.H
- IO.C
- IO.H
- MAIN.C
- MAIN.H
- MemInit878_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

```

1 //
2 * read_song_string.c
3 *
4 * Created on: 26.05.2010
5 * Author: Wrenovlt
6 *
7
8 #include "main.h"
9 #include "read_song_string.h"
10
11 void SetOctaveNormal(void)
12 {
13     OctaveLow = OFF; // clear Global Variable
14
15     SFR_PAGE(_cc0, nosST); // switch to page 0
16     CC0_VstopTmr(CC0_TIMER_13); // Stop Timer 13: CC06_TCTR4H |= 0x01
17
18     SFR_PAGE(_cc1, nosST); // switch to page 1
19     CC06_TCTR0H = 0x01; // Prescaler = 2: load CC06 timer 13 control register 0 high
20
21     SFR_PAGE(_cc0, nosST); // switch to page 0
22     CC06_VstartTmr(CC06_TIMER_13); // Start Timer 13: CC06_TCTR4H |= 0x02
23 }
24
25 void SetOctaveLow(void)
26 {
27     OctaveLow = ON; // set Global Variable
28
29     SFR_PAGE(_cc0, nosST); // switch to page 0
30     CC0_VstopTmr(CC0_TIMER_13); // Stop Timer 13: CC06_TCTR4H |= 0x01
31
32     SFR_PAGE(_cc1, nosST); // switch to page 1
33     CC06_TCTR0H = 0x02; // Prescaler = 4: load CC06 timer 13 control register 0 high
34
35     SFR_PAGE(_cc0, nosST); // switch to page 0
36     CC06_TCTR4H = 0x02; // Start Timer 13: CC06_TCTR4H |= 0x02
37 }
38
  
```

```

39// Note:
40// The function read_song_string() is a recursive function and will be called recursively until a note is found.
41// The local variable subptr is only used within one function-call to determine the tempo and
42// can be destroyed from one function-call to another.
43// subptr could also be defined as either a static or a global variable.
44// Therefore, the keyword reentrant is not needed.
45void read_song_string(void)
46{
47    unsigned char subptr[4] = {0};
48    current_note_length = old_note_length;
49
50    switch (song[pos])
51    {
52        // select note:
53        case 'C': note = 0;
54        switch (song[++pos])
55        {
56            case '+': note++; pos++; break;
57            case '-': octave--;
58                note = 11;
59                pos++;
60            default: ;
61        }
62        case 'D': note = 2;
63        switch (song[++pos])
64        {
65            case '+': note++; pos++; break;
66            case '-': note--; pos++; break;
67            default: ;
68        }
69    }
70

```

```

71 case 'E': note=4;
72 switch (song[++pos])
73 {
74     case '+': note++; pos++; break;
75     case '-': note--; pos++; break;
76     default: ; break;
77 }
78
79 case 'F': note=5;
80 switch (song[++pos])
81 {
82     case '+': note++; pos++; break;
83     case '-': note--; pos++; break;
84     default: ; break;
85 }
86
87 case 'G': note=7;
88 switch (song[++pos])
89 {
90     case '+': note++; pos++; break;
91     case '-': note--; pos++; break;
92     default: ; break;
93 }
94
95 case 'A': note=9;
96 switch (song[++pos])
97 {
98     case '+': note++; pos++; break;
99     case '-': note--; pos++; break;
100    default: ; break;
101 }
102
103 case 'H': note=11;
104 switch (song[++pos])
105 {
106     case '+': octave++;
107         note=0;
108         pos++;
109     case '-': note--; pos++;
110     default: ; break;
111 }

```

```

112 // adjust note length:
113 case 'L': switch (song[++pos])
114 {
115     case '1': if (song[++pos]!='G')
116         current_note_length=length_of_a_whole_note/16;
117     else
118     {
119         pos--;
120         current_note_length=length_of_a_whole_note;
121     }
122     break;
123 case '2': current_note_length=length_of_a_whole_note/2;
124 case '4': current_note_length=length_of_a_whole_note/4;
125 case '8': current_note_length=length_of_a_whole_note/8;
126 default:
127     break;
128 }
129 old_note_length=current_note_length;
130 pos++;
131 read_song_string(); break;
132
133 // set rest:
134 case 'P': switch (song[++pos])
135 {
136     case '1': if (song[++pos]!='G')
137         current_note_length=length_of_a_whole_note/16;
138     else
139     {
140         pos--;
141         current_note_length=length_of_a_whole_note;
142     }
143     break;
144 case '2': current_note_length=length_of_a_whole_note/2;
145 case '4': current_note_length=length_of_a_whole_note/4;
146 case '8': current_note_length=length_of_a_whole_note/8;
147 default:
148     break;
149 }
150 note=12;
151 pos++;
152 break;
153

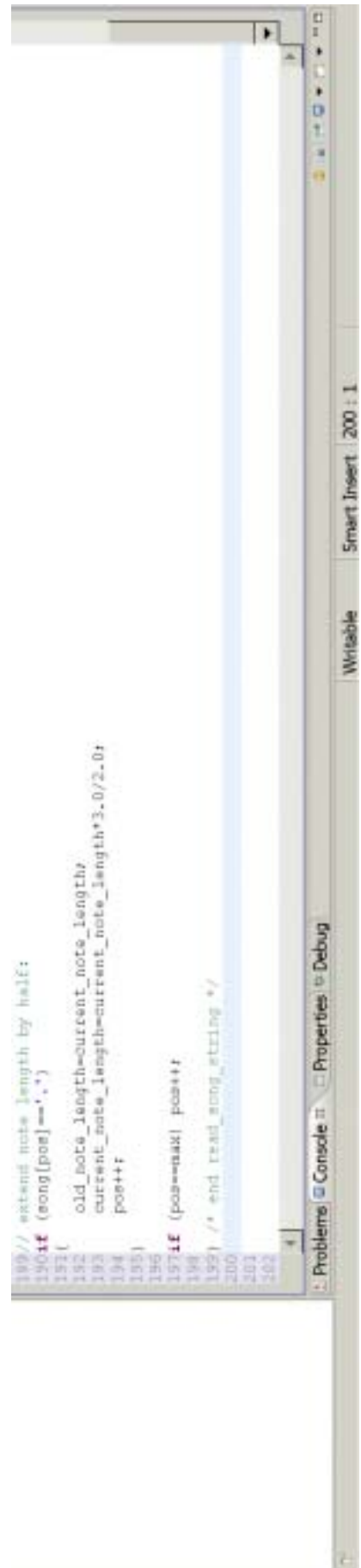
```



```

153// adjust octave:
154 case '0': switch (song[++pos])
155 {
156     case '0': octave=1; break;
157     case '1': octave=2; break;
158     case '2': octave=4; break;
159     case '3': octave=8; break;
160     default: if (song[pos]=='1') octave=1, SetOctaveLOW();
161               if (song[pos]=='N') octave=1, SetOctaveNORMAL();
162               break;
163 }
164 pos++;
165 read_song_string(); break;
166
167// tempo:
168 case 't': pos++;
169 subtr[3]=0; //string termination
170 if (song[pos]=='1')
171 {
172     subtr[0]=song[pos];
173     subtr[1]=song[++pos];
174     subtr[2]=song[++pos];
175 }
176 else
177 {
178     subtr[0]=song[pos];
179     subtr[1]=song[++pos];
180     subtr[2]=' ';
181 }
182 tempo=atoi(subtr);
183 pos++;
184 read_song_string(); break;
185
186 default: break;
187 /* end case */
188

```



```

189 // extend note length by half:
190 if (song[pos] == ".")
191 {
192     old_note_length = current_note_length;
193     current_note_length = current_note_length * 3.0 / 2.0;
194     pos++;
195 }
196
197 if (pos == MAX) pos++;
198
199 /* end read_song_string */
200
201
202

```



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

Auswählen 75% Hilfe

TCTR0H
Timer Control Register 0 High
Reset Value: 00_H

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 $f_{T13} = f_{CCU} / 2^{T13CLK}$ <table> <tr><td>000</td><td>$f_{T13} = f_{CCU}$</td></tr> <tr><td>001</td><td>$f_{T13} = f_{CCU}/2$</td></tr> <tr><td>010</td><td>$f_{T13} = f_{CCU}/4$</td></tr> <tr><td>011</td><td>$f_{T13} = f_{CCU}/8$</td></tr> <tr><td>100</td><td>$f_{T13} = f_{CCU}/16$</td></tr> <tr><td>101</td><td>$f_{T13} = f_{CCU}/32$</td></tr> <tr><td>110</td><td>$f_{T13} = f_{CCU}/64$</td></tr> <tr><td>111</td><td>$f_{T13} = f_{CCU}/128$</td></tr> </table>	000	$f_{T13} = f_{CCU}$	001	$f_{T13} = f_{CCU}/2$	010	$f_{T13} = f_{CCU}/4$	011	$f_{T13} = f_{CCU}/8$	100	$f_{T13} = f_{CCU}/16$	101	$f_{T13} = f_{CCU}/32$	110	$f_{T13} = f_{CCU}/64$	111	$f_{T13} = f_{CCU}/128$
000	$f_{T13} = f_{CCU}$																		
001	$f_{T13} = f_{CCU}/2$																		
010	$f_{T13} = f_{CCU}/4$																		
011	$f_{T13} = f_{CCU}/8$																		
100	$f_{T13} = f_{CCU}/16$																		
101	$f_{T13} = f_{CCU}/32$																		
110	$f_{T13} = f_{CCU}/64$																		
111	$f_{T13} = f_{CCU}/128$																		

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:

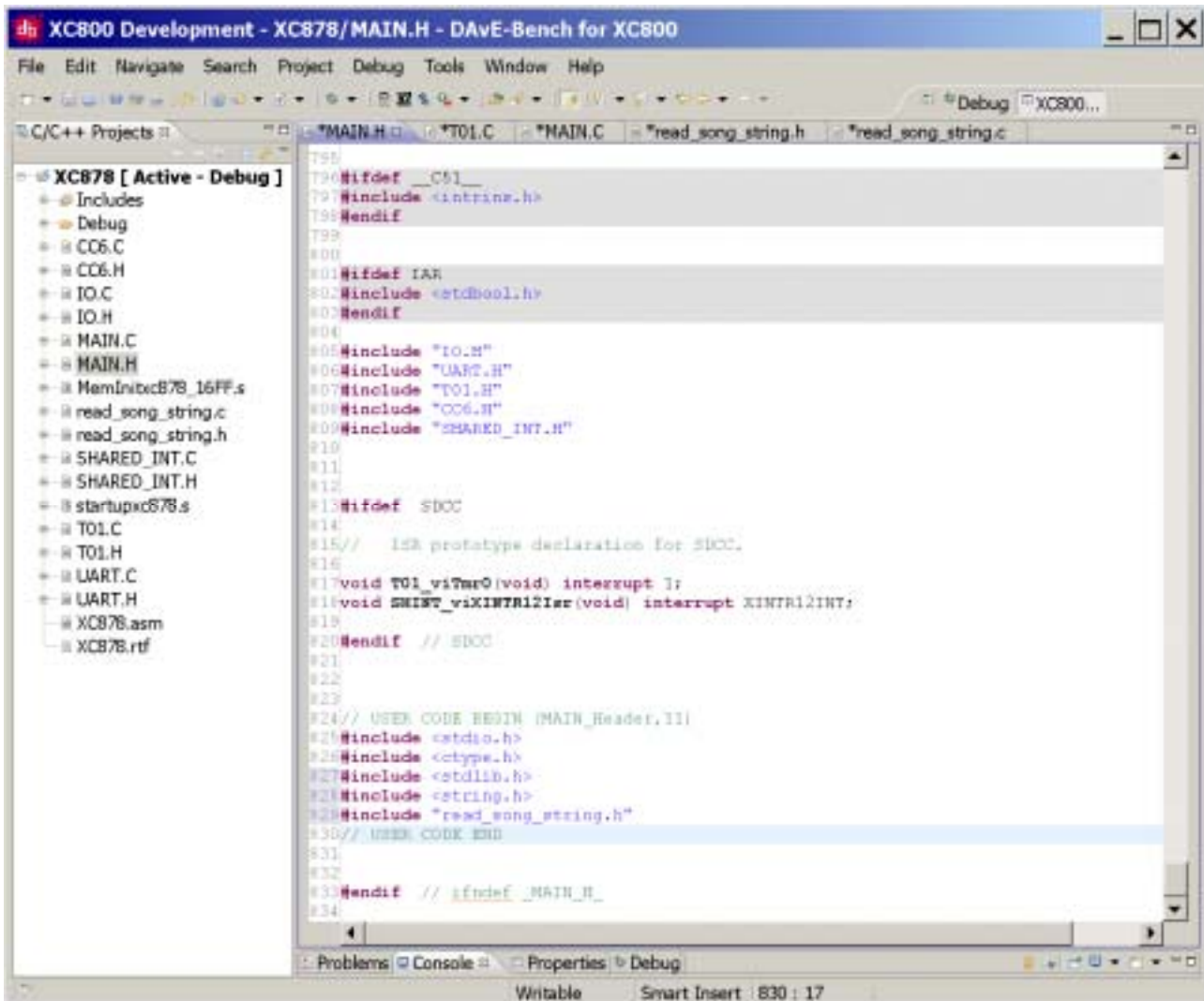
Normally, functions in DAvE-Bench cannot be called recursively or in a fashion which causes reentrancy. The reason for this limitation is that **function arguments** and **local variables** are stored in fixed memory locations. Recursive calls to the function use the same memory locations. And, in this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

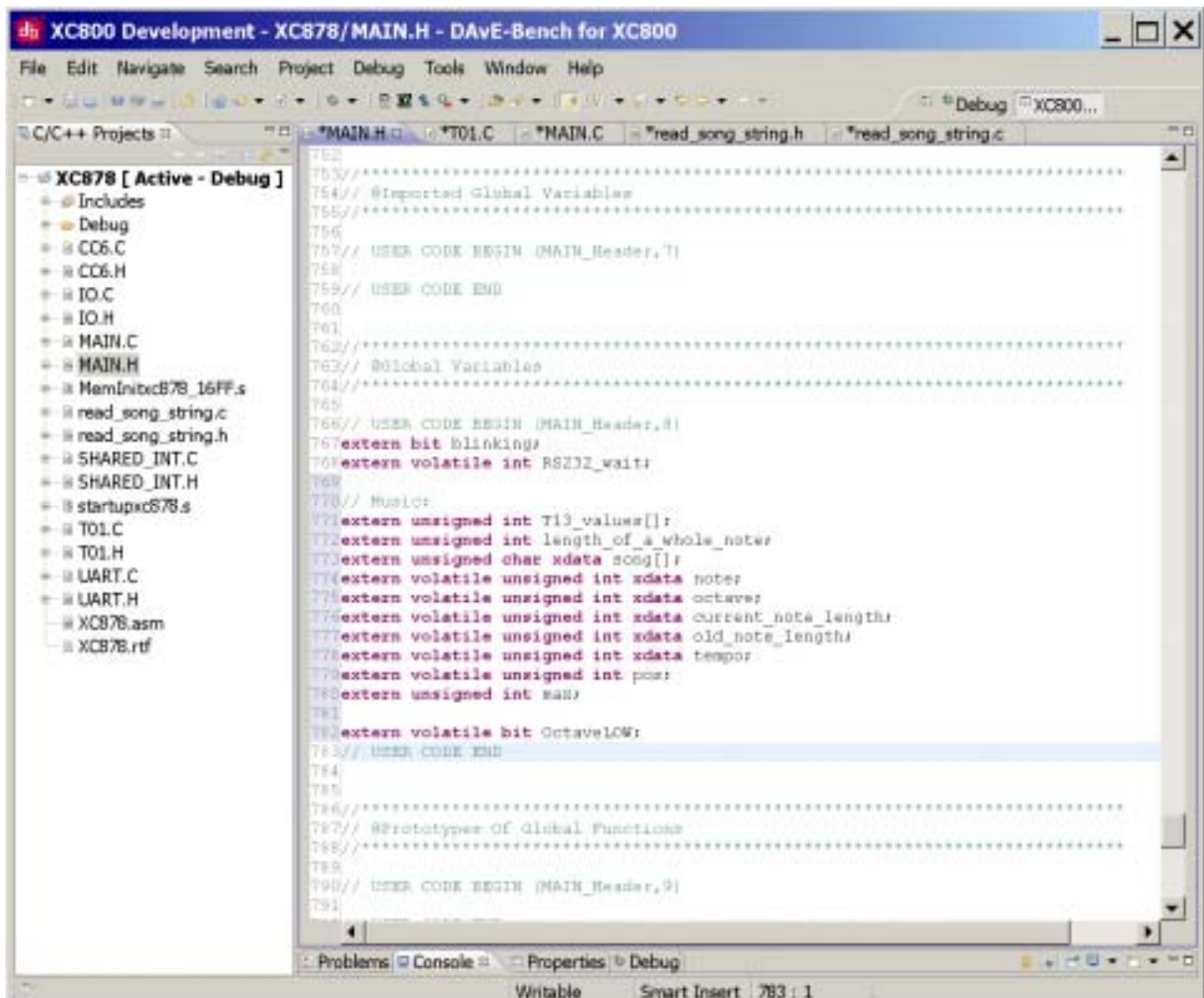
```
#include <stdlib.h>
#include <string.h>
#include "read_song_string.h"
```



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

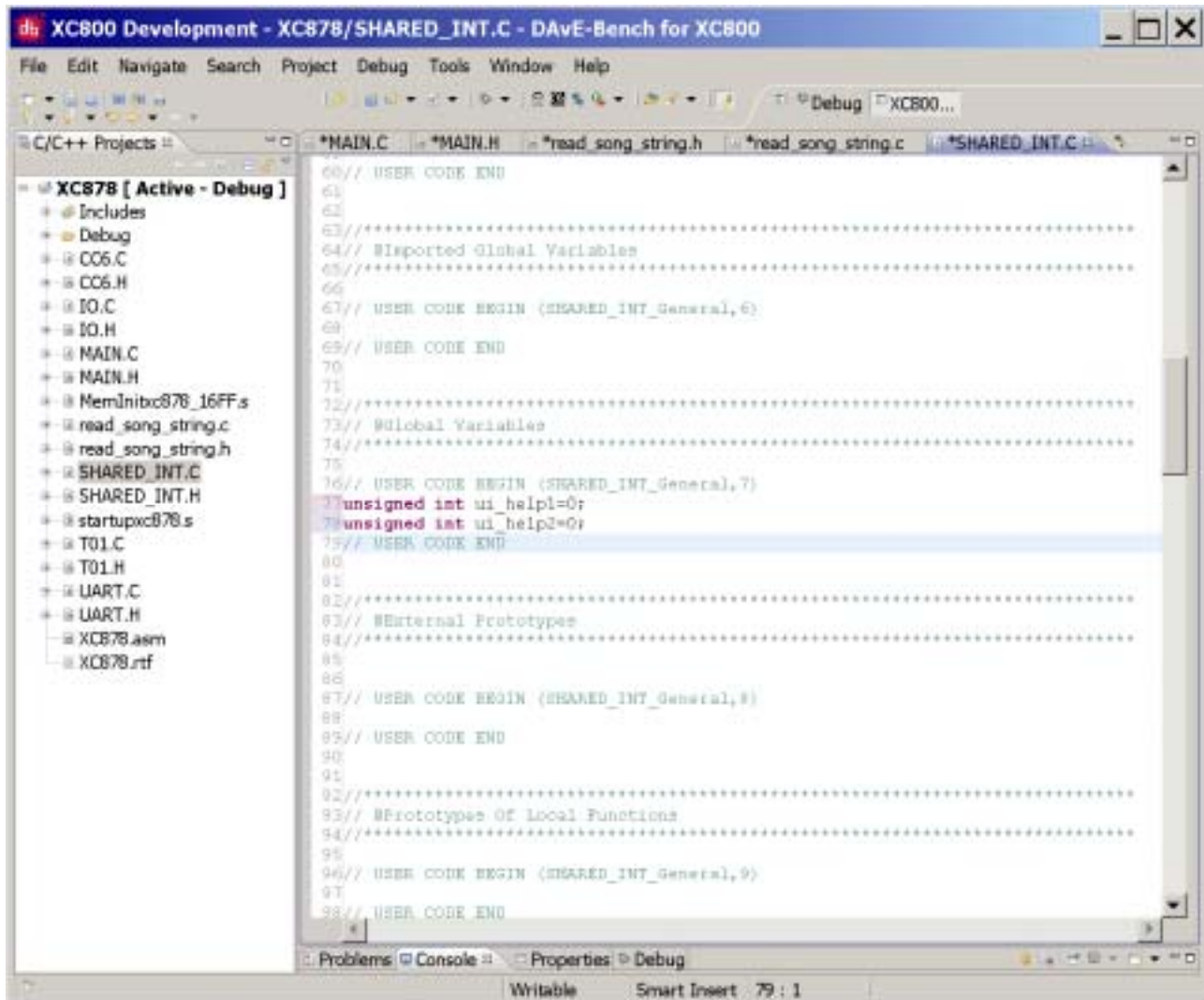
```
// Music:
extern unsigned int T13_values[];
extern unsigned int length_of_a_whole_note;
extern unsigned char xdata song[];
extern volatile unsigned int xdata note;
extern volatile unsigned int xdata octave;
extern volatile unsigned int xdata current_note_length;
extern volatile unsigned int xdata 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));
else if (OctaveLOW==ON)
    printf_fast_f("T13-
f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*166.6667/1000.0/1000.0));

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
- CC06.C
- CC06.H
- IO.C
- IO.H
- MAIN.C
- MAIN.H
- MemInitXC878_35FF.a
- 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

```

180 void SHINT_vXINTNTHier(void) interrupt XINTNTHINT
181 {
182     // USER CODE BEGIN (Node12,2)
183     // USER CODE END
184     SFR_PAGE[eu3, SST2]; // switch to page 2
185     // CC06 Mode 2 interrupt handling section...
186     if (IRCON4 & 0x01) // If CC06M2
187     {
188         INCON4 &= ~0x01;
189         // USER CODE BEGIN (Node12,3)
190         // USER CODE END
191         SFR_PAGE[cc3, SST2]; // switch to page 3
192         if (CC06_ISL & 0x80) // If IFL_T12PW
193         {
194             // timer T12 period match detection
195             SFR_PAGE[cc0, SST2]; // switch to page 0
196             CC06_ISRL = 0x80; // clear flag IFL_T12PW
197             // USER CODE BEGIN (Node12,17)
198             if ( (char)SRUF == 'a' ) // song aborted by user
199                 pos=next+1;
200             if (pos==next)
201             {
202                 read_song_string(); // read next note
203                 // this function is called recursively until a note is found
204             }
205             // T12PW adjust note length / set Timer 12 period value for note length:
206             // Page 1: T12PRL, T12PRH
207             SFR_PAGE[cc1, SST2]; // switch to CC06_PAGE = 1 without saving
208             ui_help1=current_note_length;
209             ui_help1=ui_help1/temper;
210             ui_help1=ui_help1*120;
211             //CC6_vSetTwrPeriod(CC6_TIMER_12,(current_note_length/temper*120));
212             CC6_vSetTwrPeriod(CC6_TIMER_12,ui_help1);
213             SFR_PAGE[cc0, SST2]; // switch to CC06_PAGE = 0 without saving
214             // Page 0: CC06ISRL, CC06ISRH:
215             // Channel 0: not used, only for measurement (100% duty cycle):
216             CC6_vLoadChannelShadowRegister(CC6_CHANNEL_0,0);
217             // Page 0: CC06ISRL, CC06ISRH:
218             // Channel 1: not used, only for measurement (100% duty cycle):
219             CC6_vLoadChannelShadowRegister(CC6_CHANNEL_1,0);
220             // Page 0: CC06ISRL, CC06ISRH:
221             // Channel 2: if compare value CC06_CC06ISR == 0 -> 100 % duty cycle for note length:
222             CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2,0);
223             // Page 0: bit: T12STR in TCTR4L
224             CC6_vEnableShadowTransfer(CC6_TIMER_12);
225             // T13, adjust note frequency / set Timer 13 period value for note frequency:
226             // Page 1: T13PRL, T13PRH
227             SFR_PAGE[cc1, SST2]; // switch to CC06_PAGE = 1 without saving
228             ui_help2=T13_values[note];
229             ui_help2=ui_help2/octaves;
230             //CC6_vSetTwrPeriod(CC6_TIMER_13,(T13_values[note]/octave));
231             CC6_vSetTwrPeriod(CC6_TIMER_13,ui_help2);
232             SFR_PAGE[cc0, SST2]; // switch to CC06_PAGE = 0 without saving
233             // Channel 3: duty cycle note-frequency = 50 %
234             // Page 0 : CC06ISRL, CC06ISRH:
235             ui_help2=ui_help2/2;
236             //CC6_vLoadChannelShadowRegister(CC6_CHANNEL_3,(T13_values[note]/octave/2));
237             CC6_vLoadChannelShadowRegister(CC6_CHANNEL_3,ui_help2);
238             // Page 0: bit: T13STR in TCTR4H
239             CC6_vEnableShadowTransfer(CC6_TIMER_13);
240         }
241     }
242 }

```

```

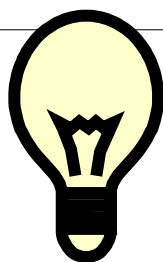
264     if (note == 0) printf_small("note=c ");
265     else if (note == 1) printf_small("note=cis");
266     else if (note == 2) printf_small("note=d ");
267     else if (note == 3) printf_small("note=dia");
268     else if (note == 4) printf_small("note=e ");
269     else if (note == 5) printf_small("note=f ");
270     else if (note == 6) printf_small("note=fis");
271     else if (note == 7) printf_small("note=g ");
272     else if (note == 8) printf_small("note=gis");
273     else if (note == 9) printf_small("note=a ");
274     else if (note == 10) printf_small("note=aia");
275     else if (note == 11) printf_small("note=b ");
276     else if (note == 12) printf_small("note=---");
277     else
278         printf_small("note=???");
279
280     if (octave == 1 && OctaveLOW==OFF) printf_small("**O0**");
281     else if (octave == 1 && OctaveLOW==ON) printf_small("**O1**");
282     else if (octave == 2) printf_small("**O2**");
283     else if (octave == 3) printf_small("**O3**");
284     else
285         printf_small("???");
286
287     printf_fast_f("T12-gv=45u",current_note_length);
288     printf_fast_f("T12-p=41.2f[s]",current_note_length*85.3333/1000.0/1000.0);
289     printf_fast_f("T13-gv=45u",T13_value[note]/octave);
290     if (OctaveLOW==OFF)
291         printf_fast_f("T13-f=87.0f[Hz]\n",1/(T13_value[note]/octave)*85.3333/1000.0/1000.0/1000.0);
292     else if (OctaveLOW==ON)
293         printf_fast_f("T13-f=87.0f[Hz]\n",1/(T13_value[note]/octave)*166.6667/1000.0/1000.0/1000.0);
294
295     IO_vTogglePin(P3_1); // Show start of next note on Port 3 Pin 1
296     // Page 0: bit: T12ES is TCTRL
297     OC6_vStartTwr(OC6_TIMER_12); // Set Timer 12 Run Set bit T12RS
298 }
299
300 // USER CODE END
301
302 }
303
304 // End of OC08082 condition check
305
306 // USER CODE BEGIN (Node1,5)
307 // USER CODE END
308
309 // USER CODE END
310
311 SFR_PAGE( mu5, R5T2); // restore the u1d ECU page
312 // End of function SHINT_VALENTIN12
313
314 // USER CODE BEGIN (SHARED_INT_General,10)
315
316 // USER CODE END
317
318
319

```

Note:

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





Registers used:

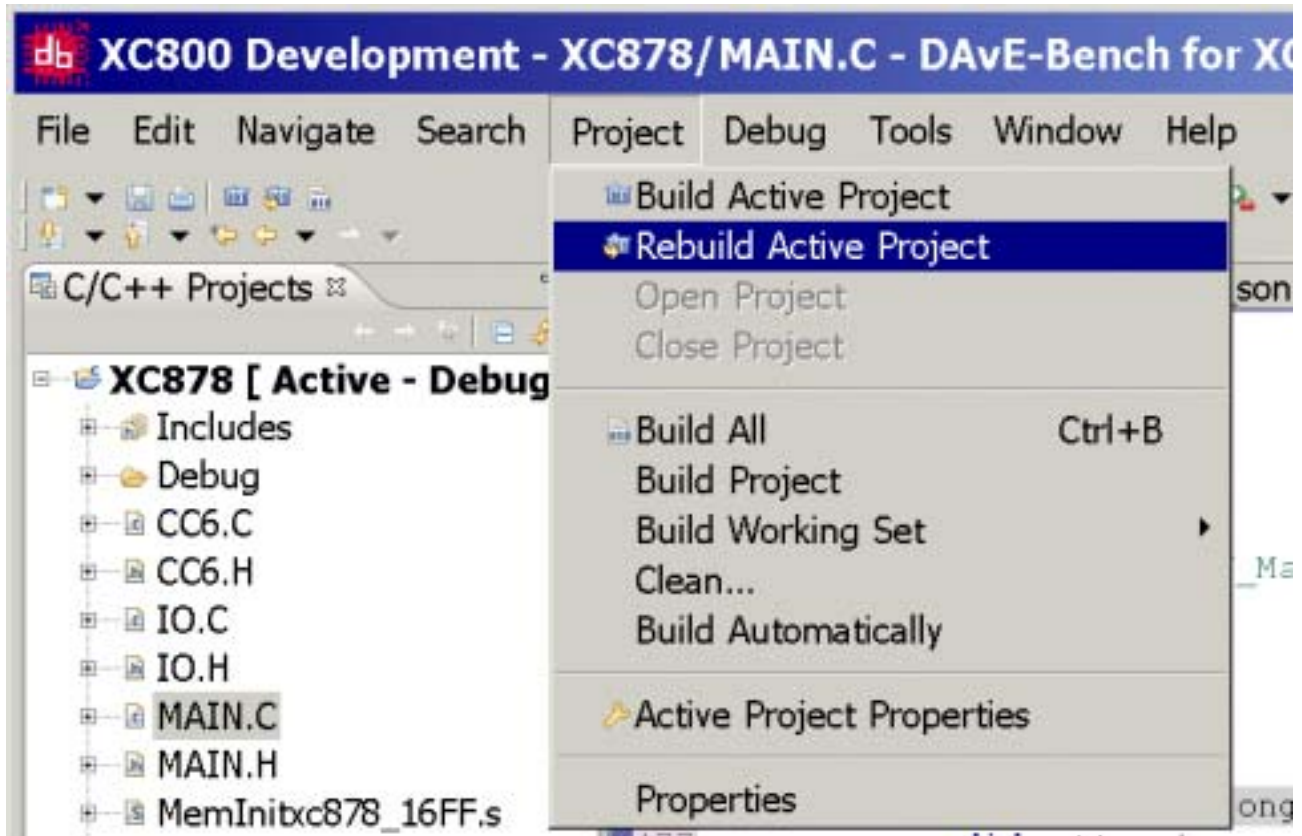
Adobe Acrobat Professional - [XC878_um_v1_1.pdf]

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

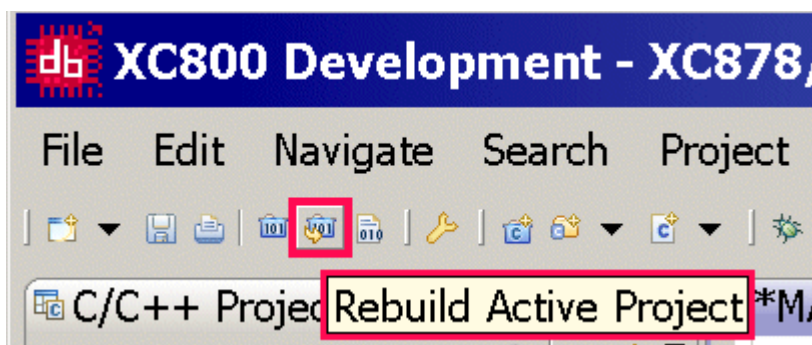
Address	Page 0	Page 1	Page 2	Page 3
9A _H	CC63SRL	CC63RL	T12MSELL	MCMOUTL
9B _H	CC63SRH	CC63RH	T12MSELH	MCMOUTH
9C _H	TCTR4L	T12PRL	IENL	ISL
9D _H	TCTR4H	T12PRH	IENH	ISH
9E _H	MCMOUTSL	T13PRL	INPL	PISEL0L
9F _H	MCMOUTSH	T13PRH	INPH	PISEL0H
A4 _H	ISRL	T12DTCL	ISSL	PISEL2
A5 _H	ISRH	T12DTCH	ISSH	
A6 _H	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

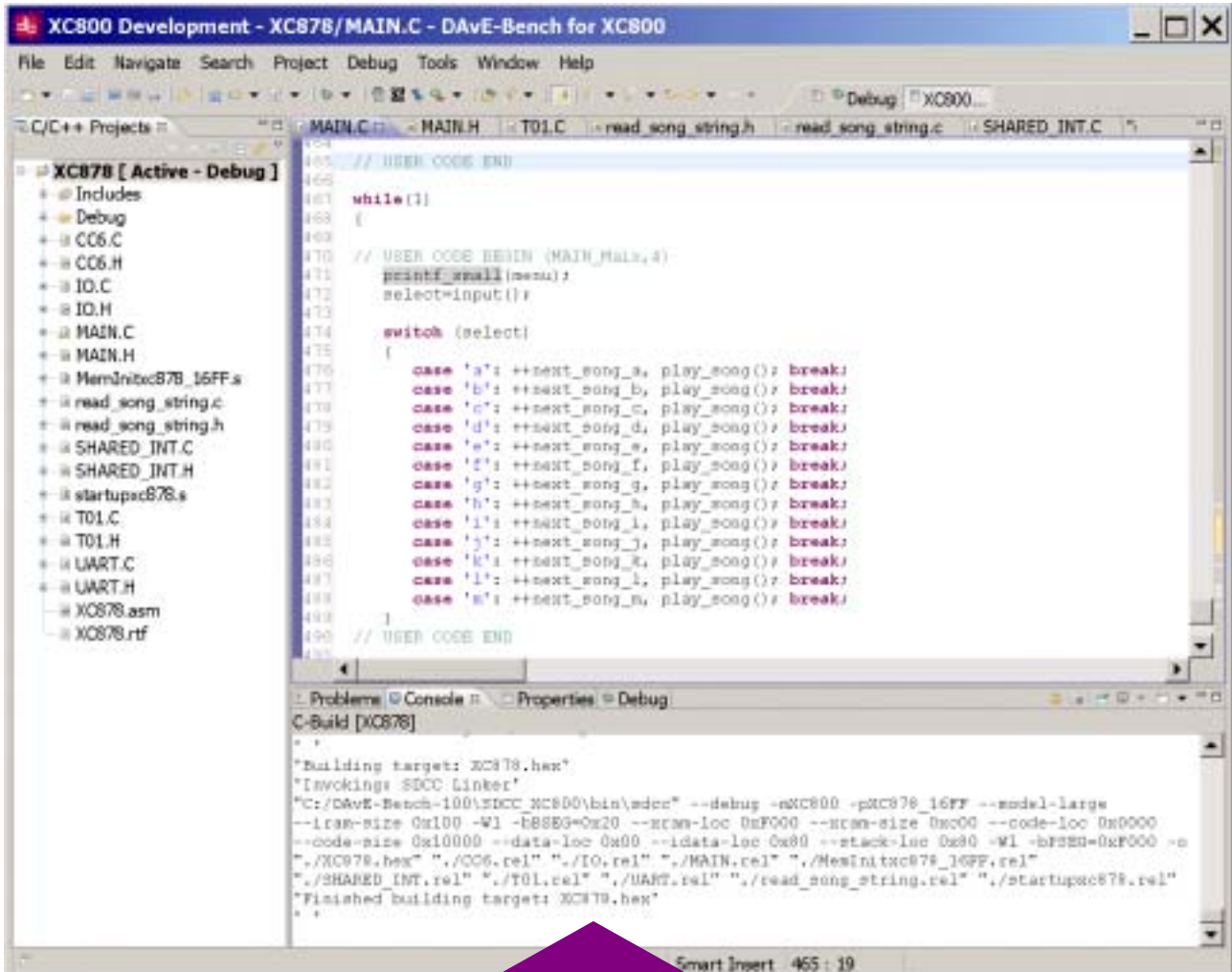
Generate your application program:

Project – Rebuild Active Project



or: [click](#) 





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" -plogffcx "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" -plogffcx "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" -plogffcx "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" -plogffcx
"../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" -plogffcx "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" -plogffcx "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" -plogffcx "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" -plogffcx "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" -plogffcx "../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'
'
```

0 Error(s)



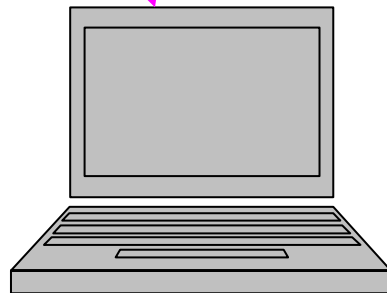


2.3) Using the debugger (DAvE Bench):
See and hear the result:

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.

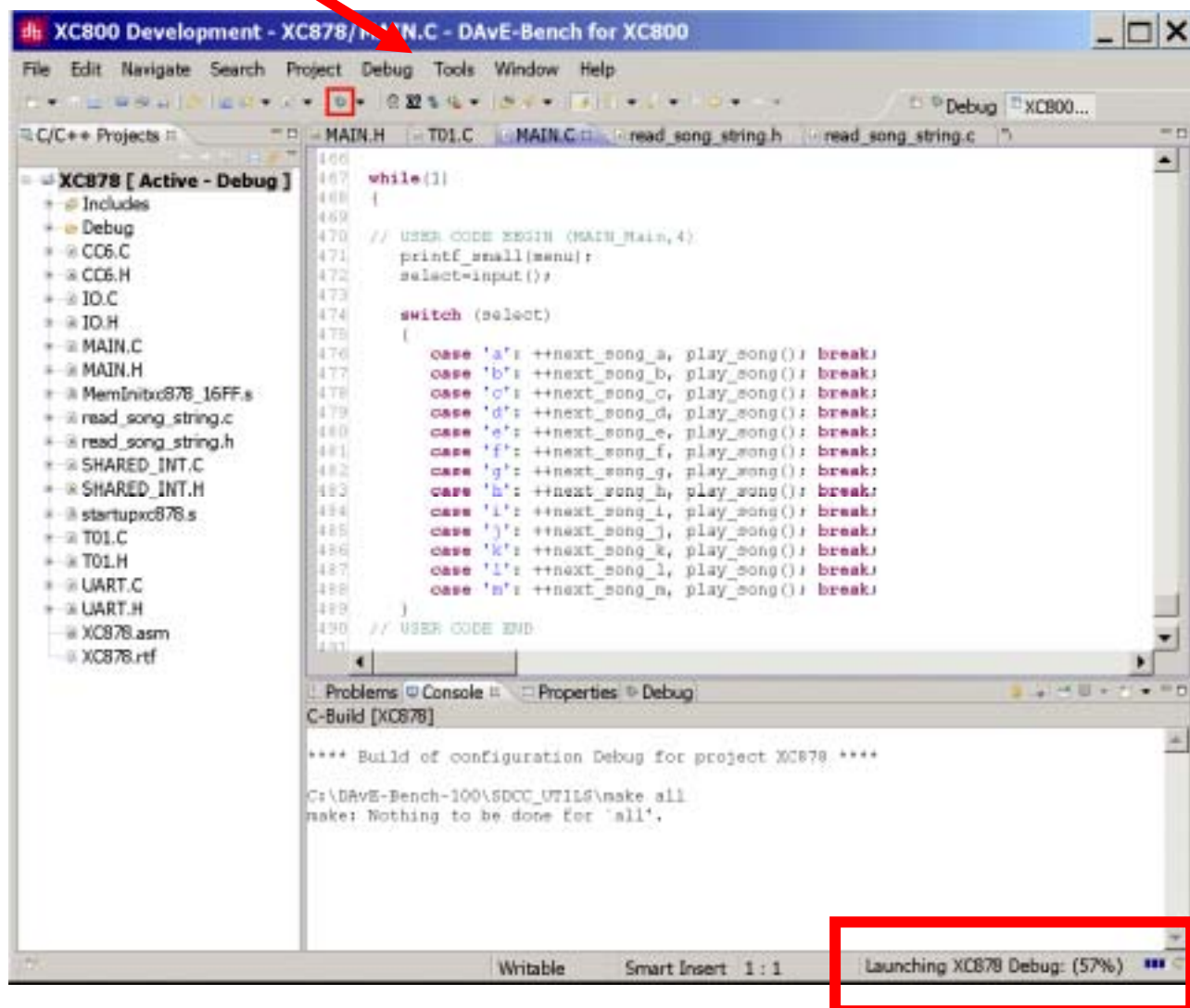


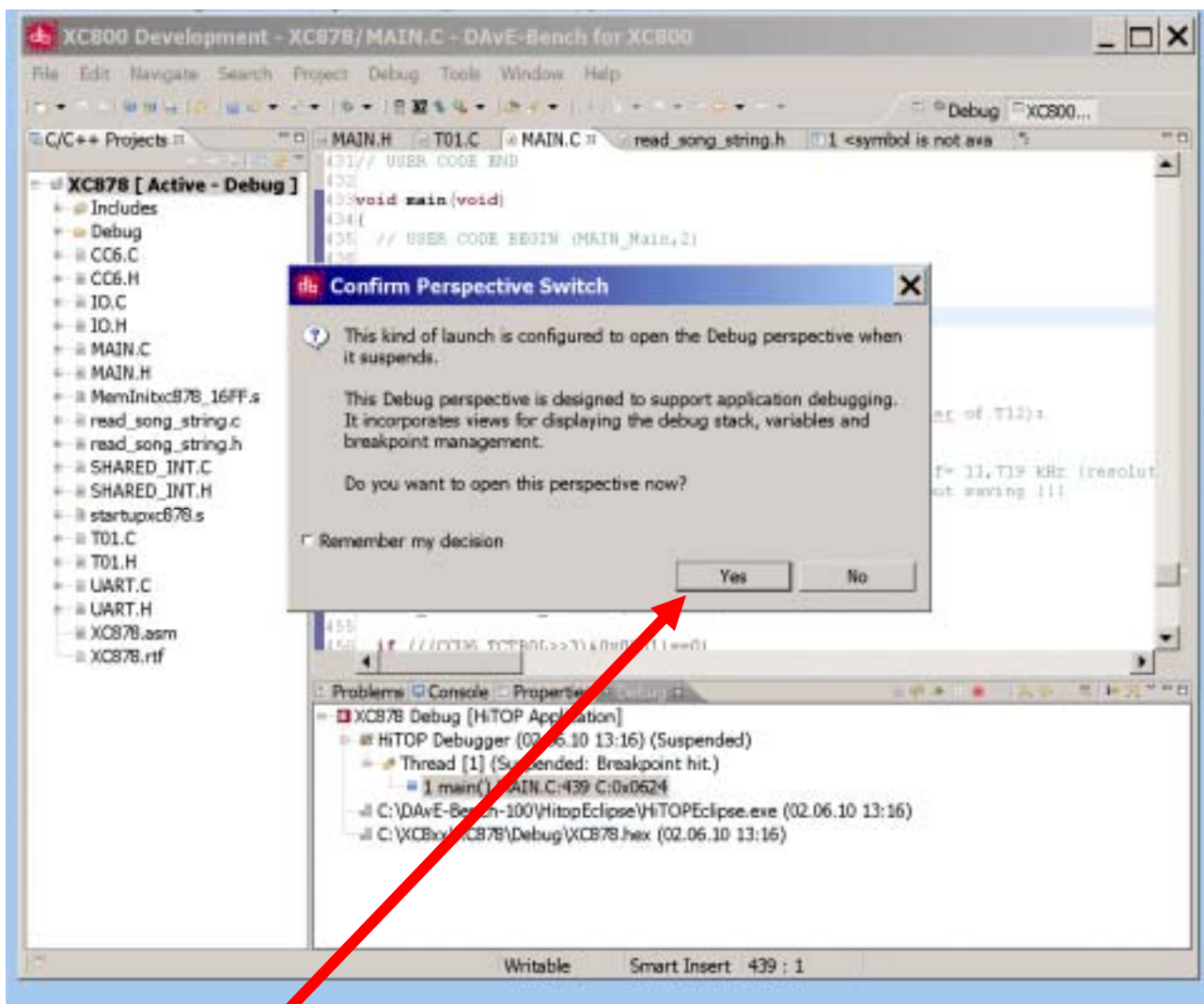
Go back to DAVe Bench and



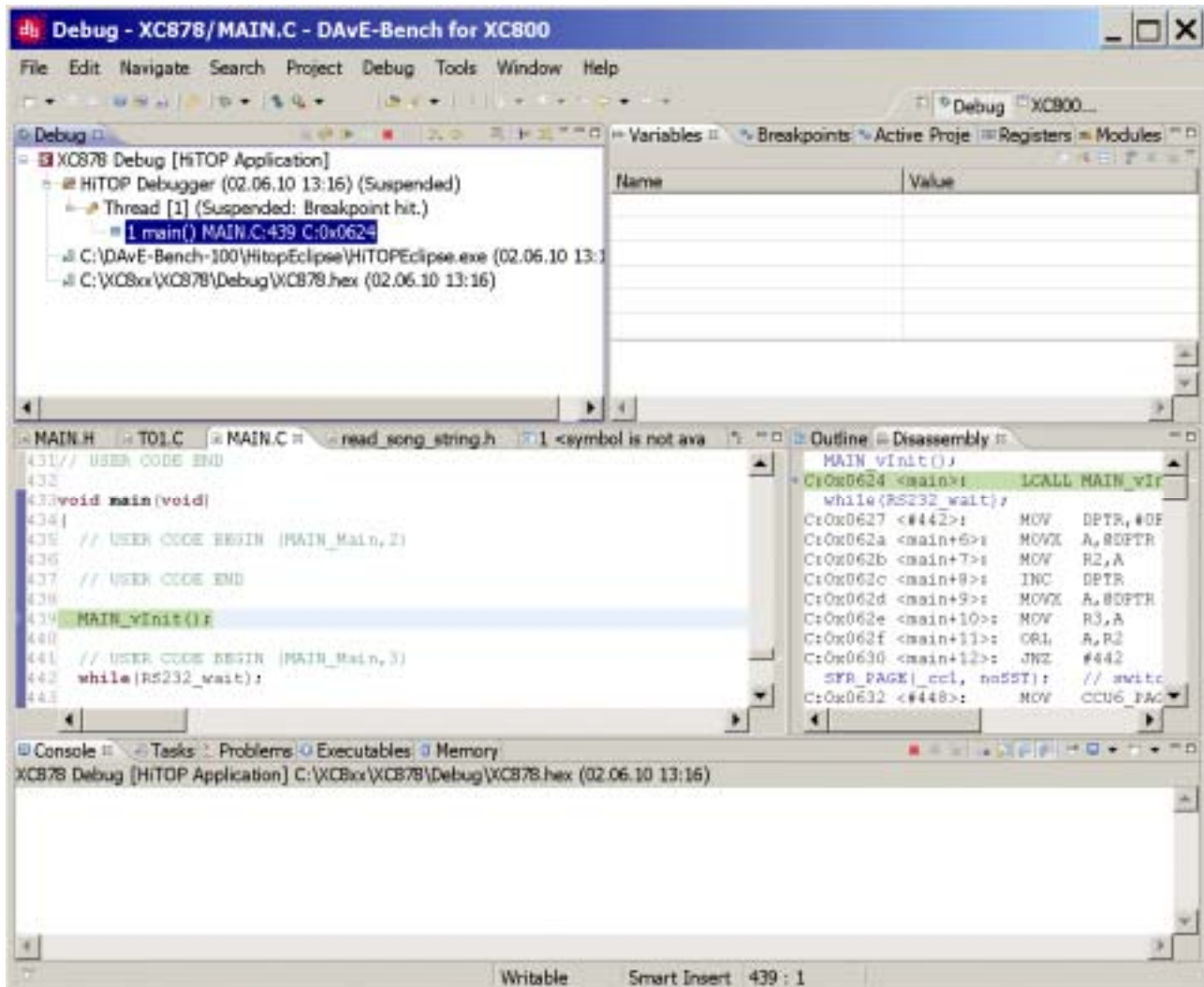
Start/Launch the debugger:

Click



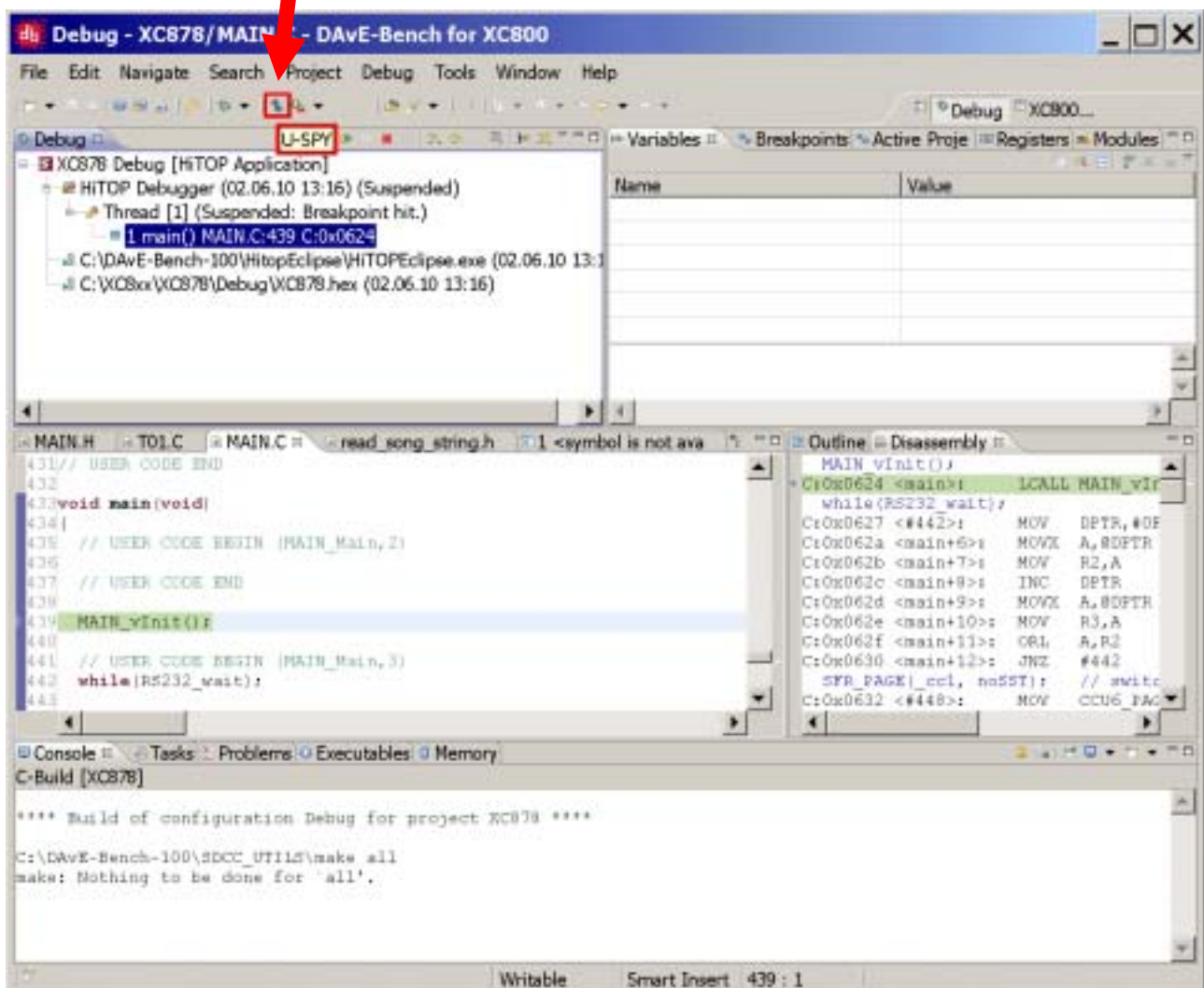


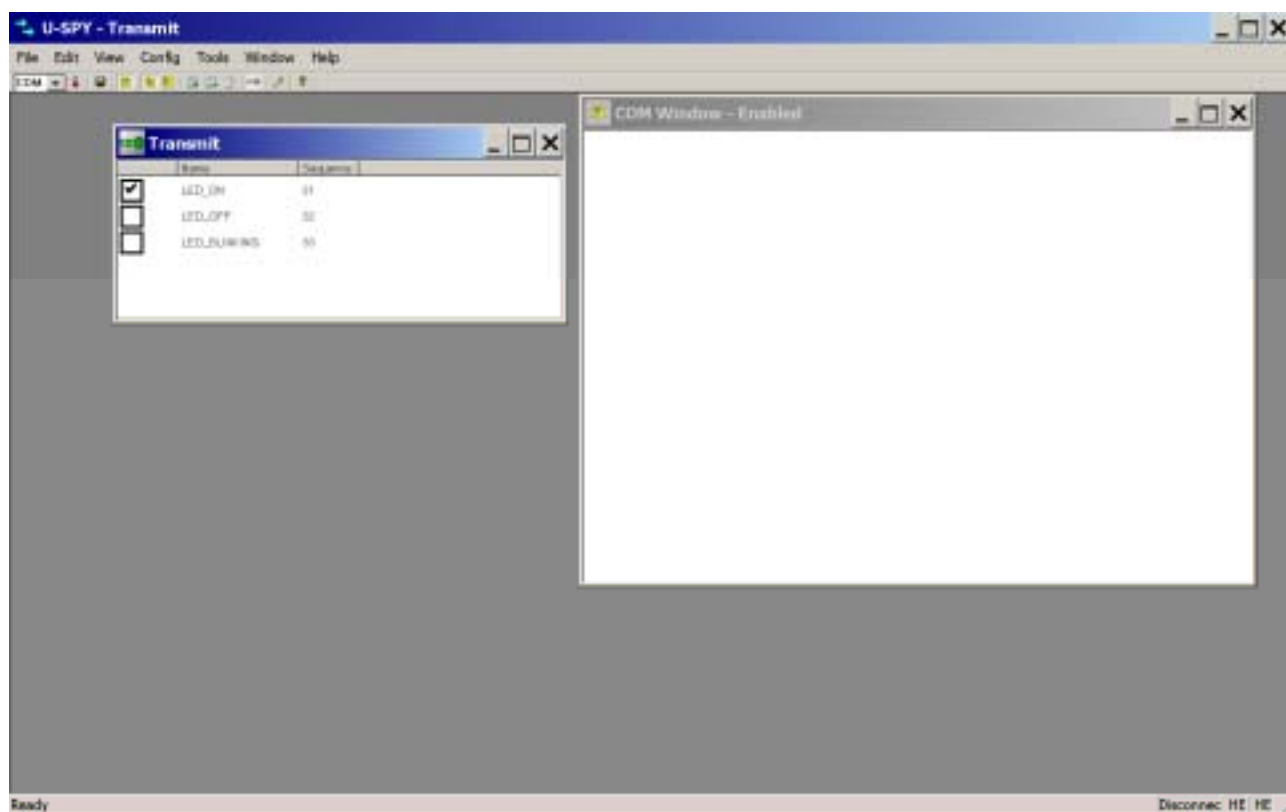
Click Yes



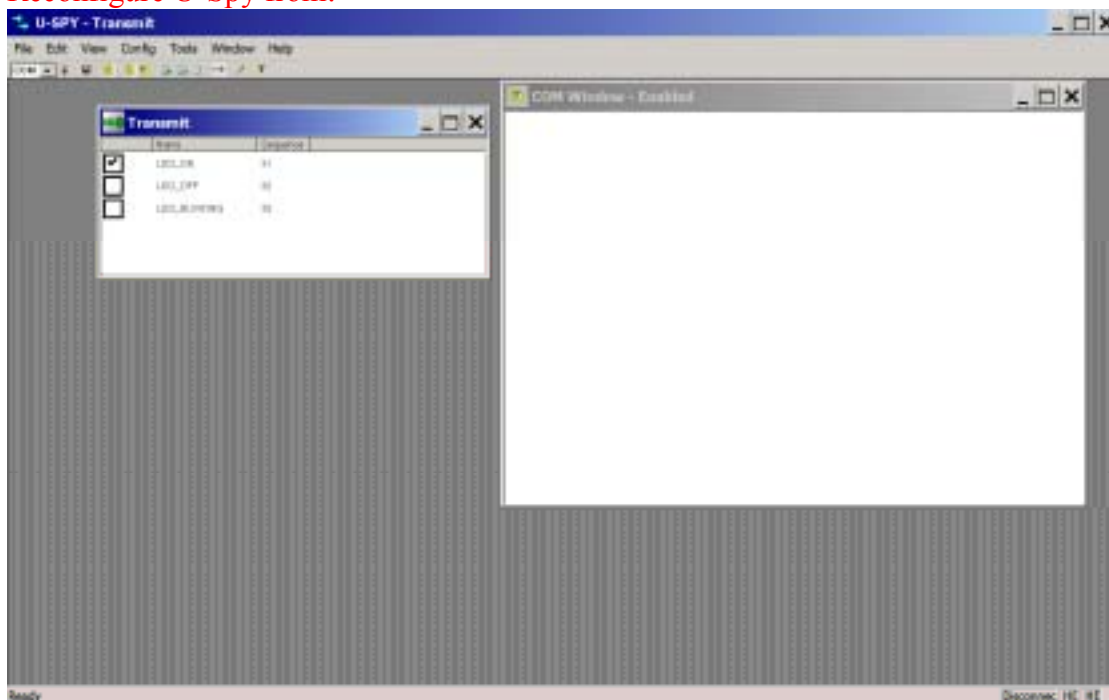


Now, start U-SPY: click 

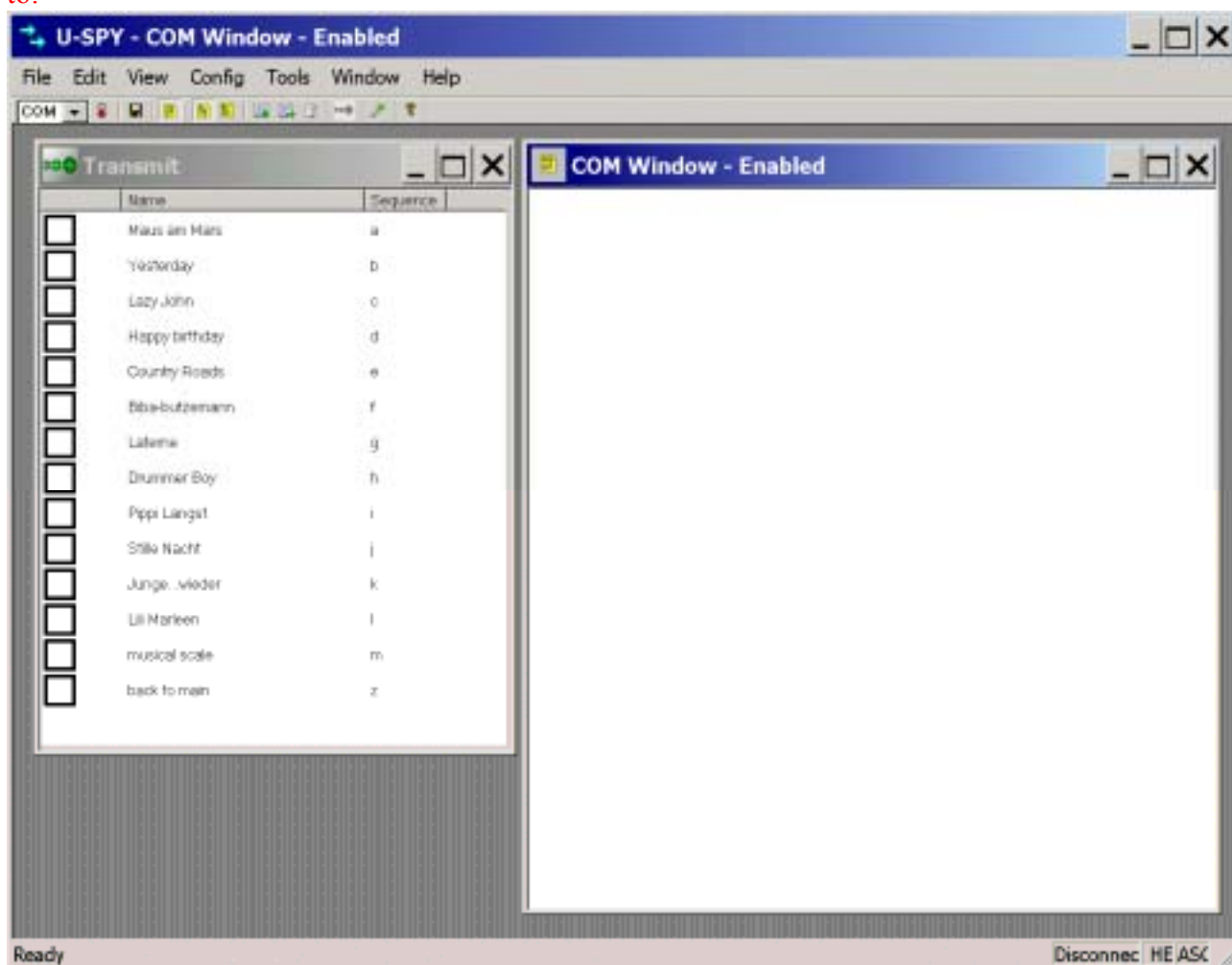


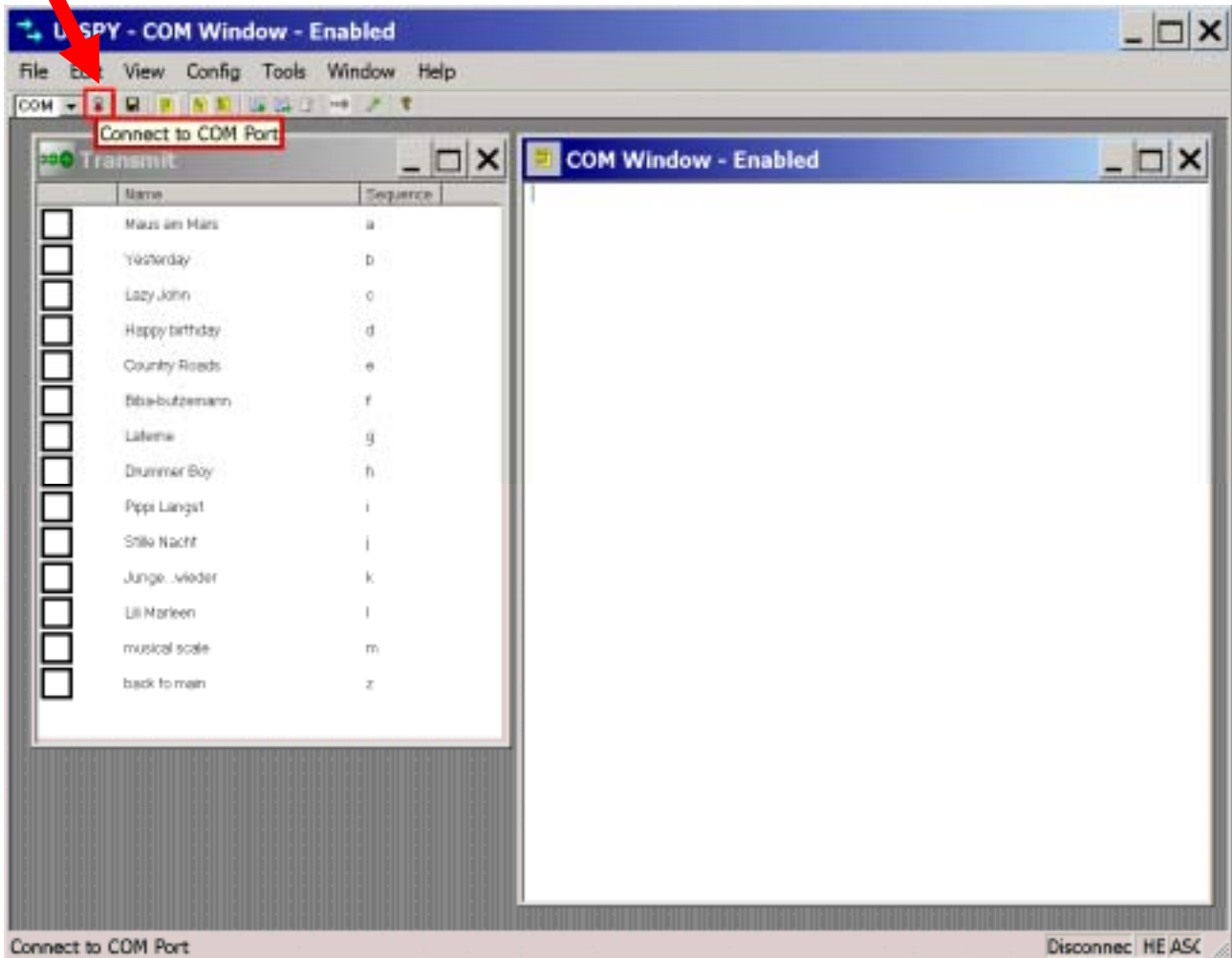
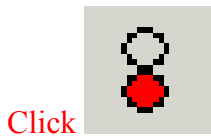


Reconfigure U-Spy from:



to:





Note:

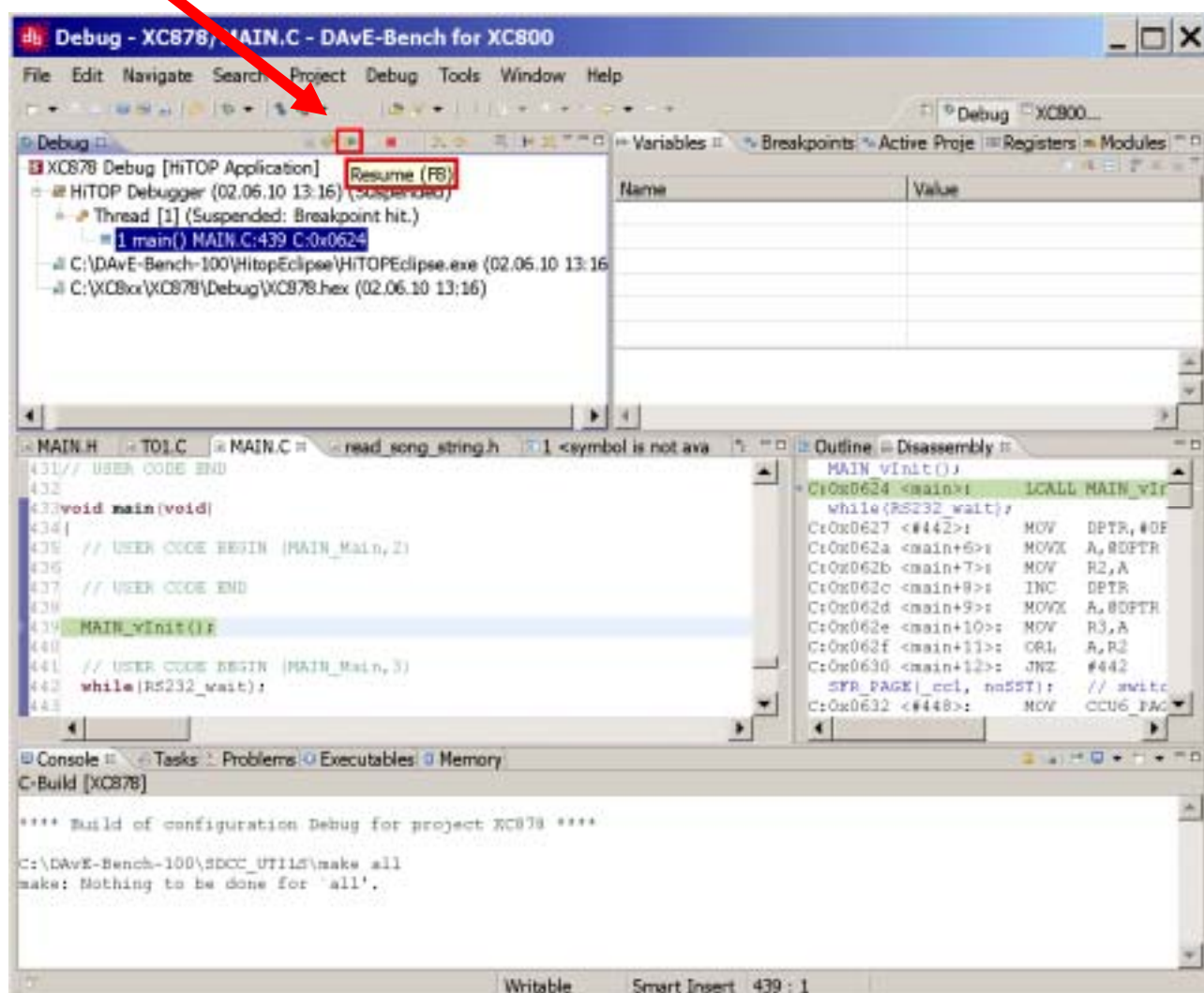


: U-SPY is now ready for serial communication!



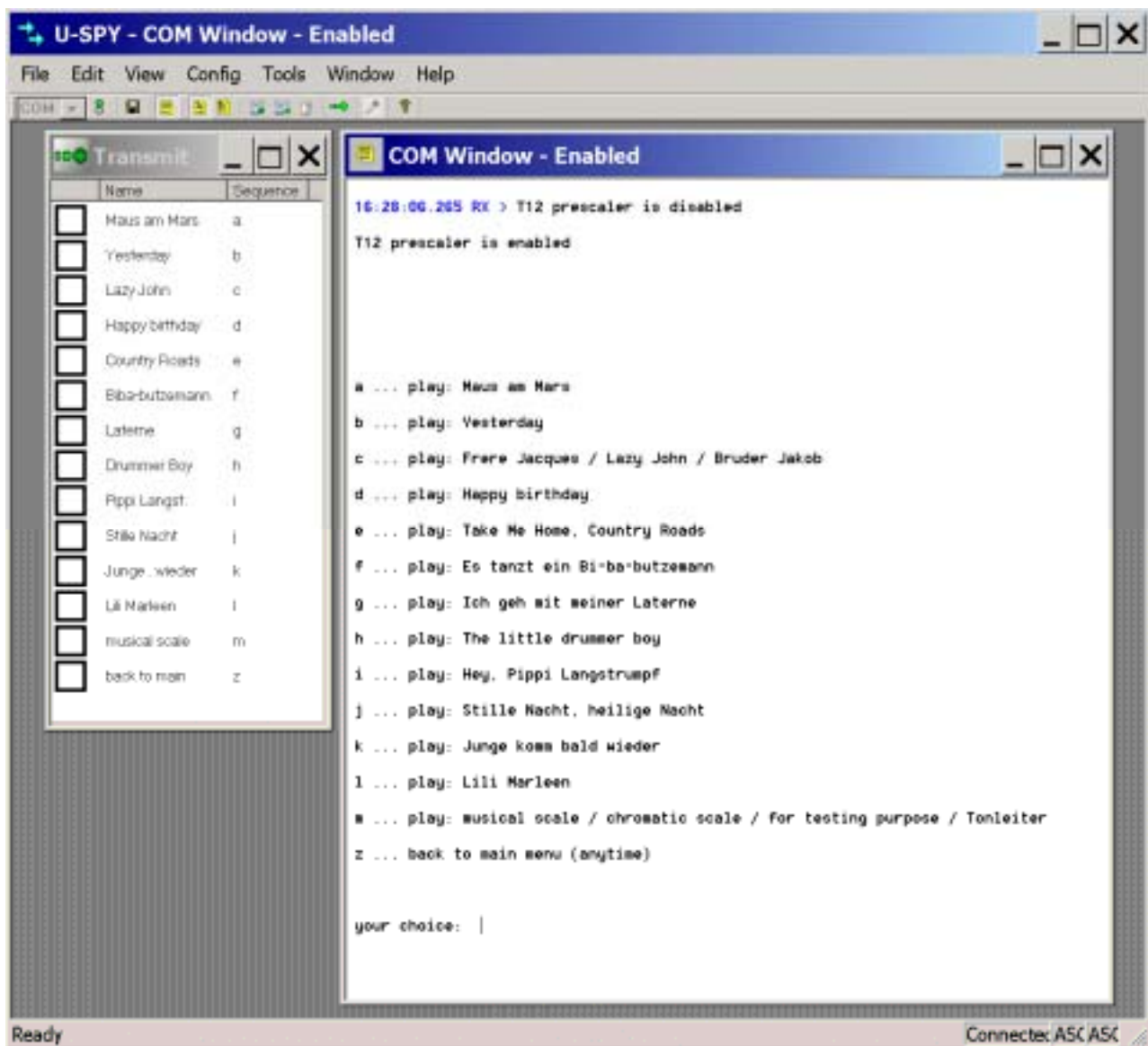
Go back to DAVe Bench and start the debugger

Click:  Resume

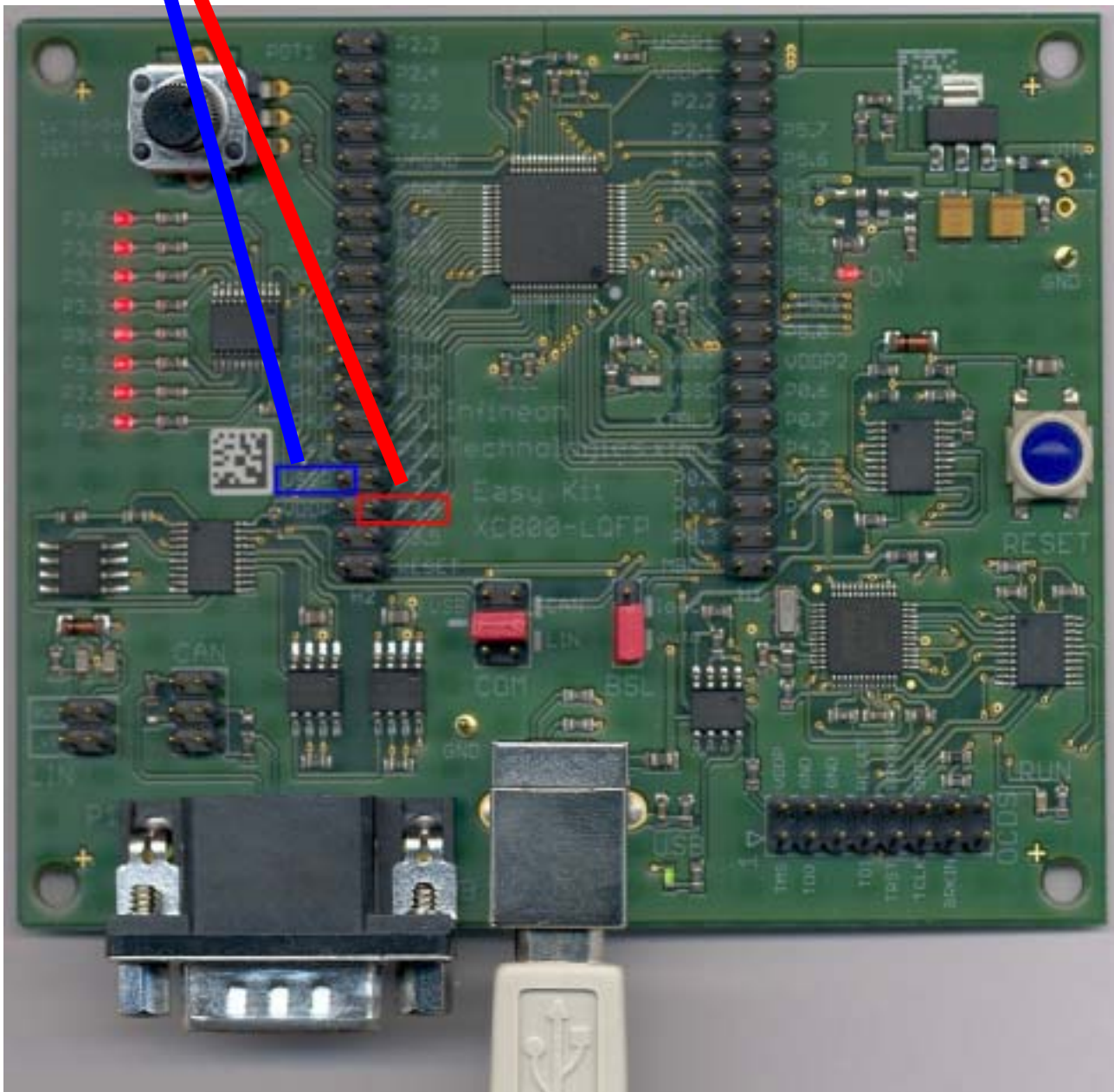





Go back to U-SPY and see the result:

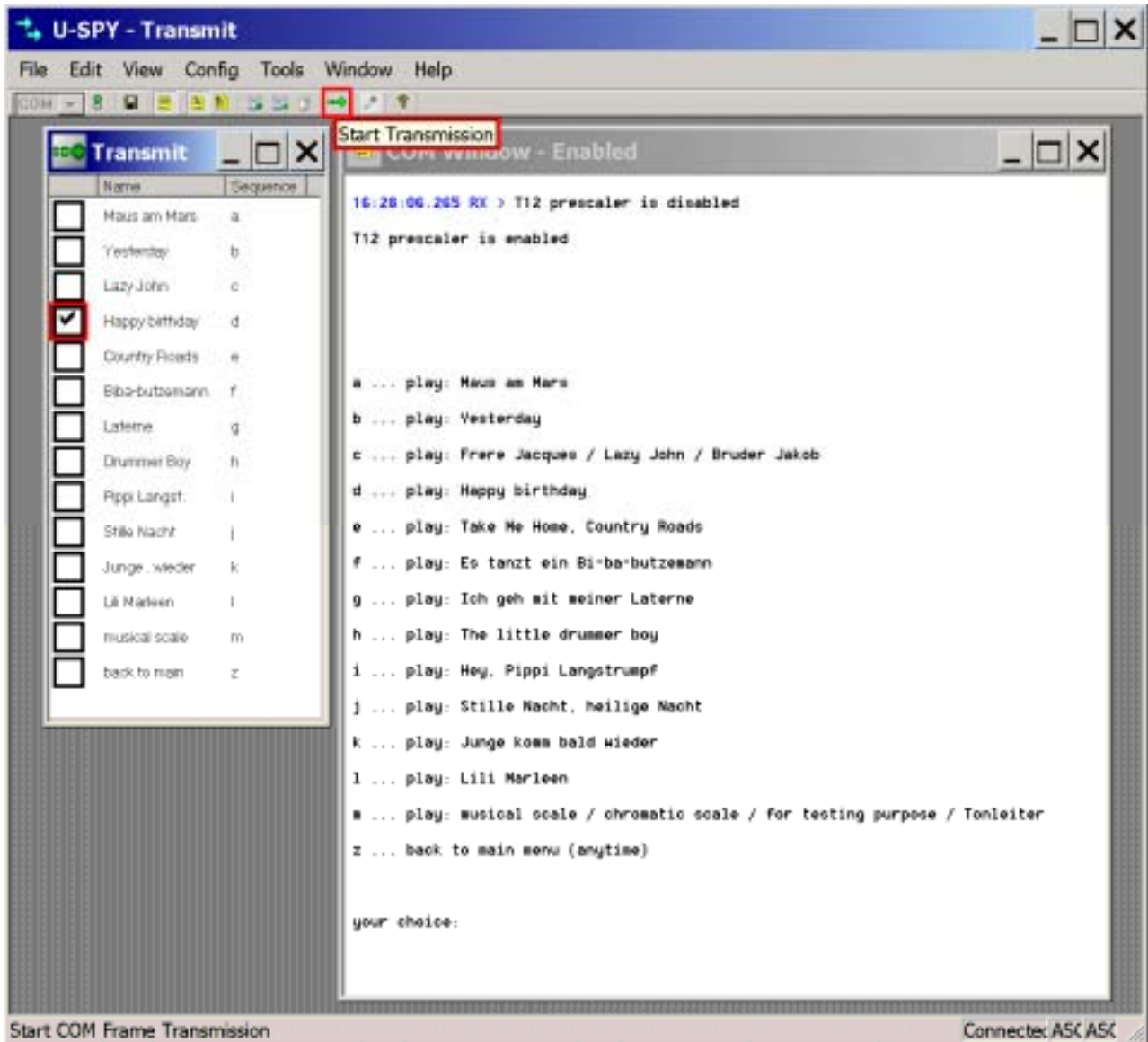


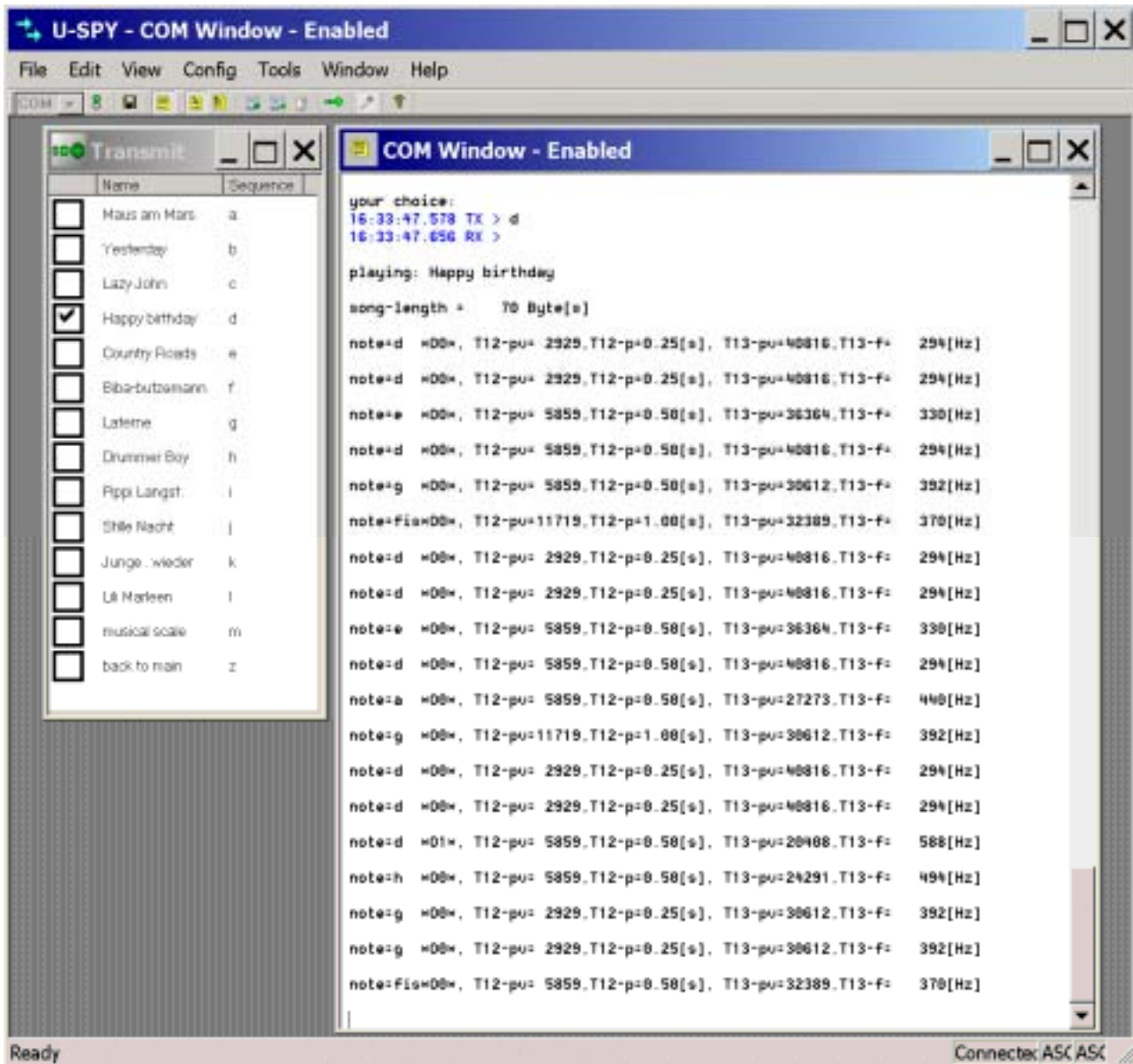
Connect CC62 (P3.4) and GND (VSSP) to your active loudspeaker(s) - and start/select a song:



See / hear / enjoy the results:

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







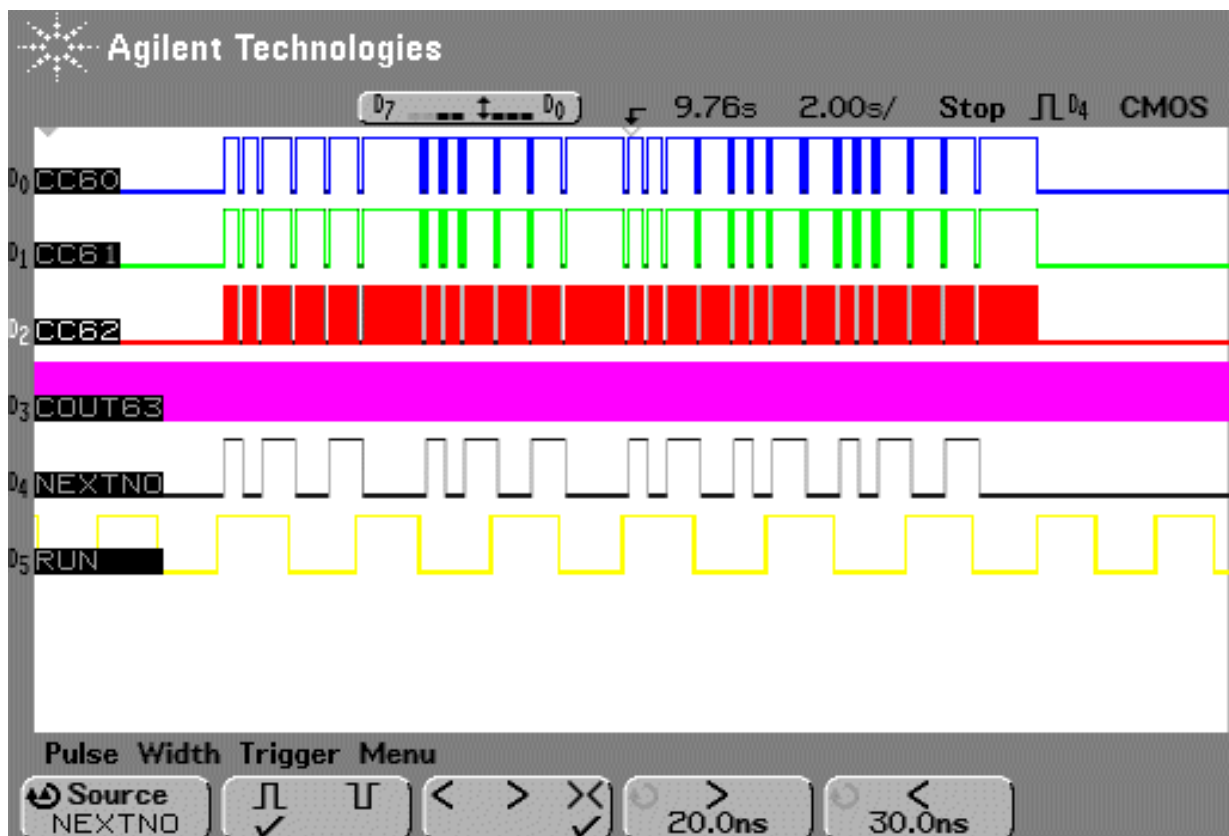
Use any logic analyser and see the result:

Note:

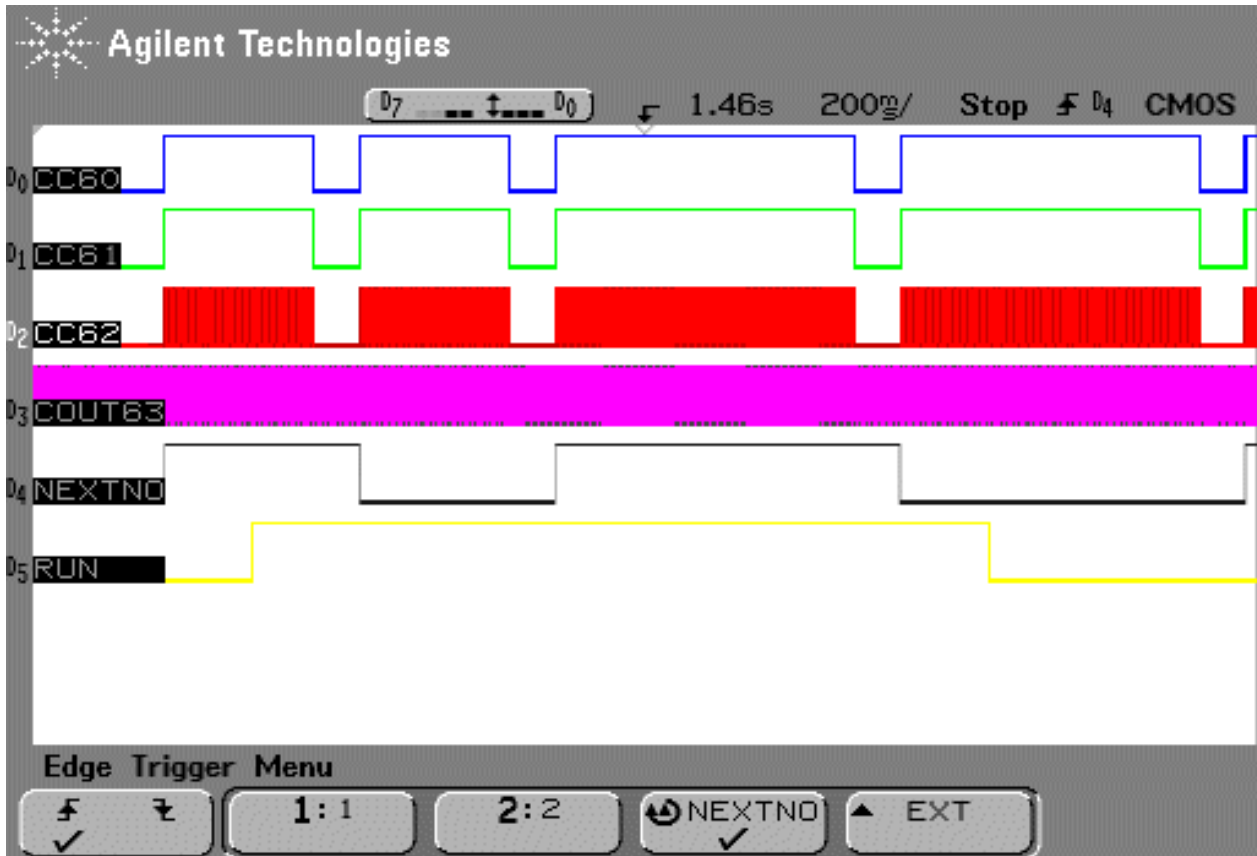
Song d: Happy birthday:

code unsigned char

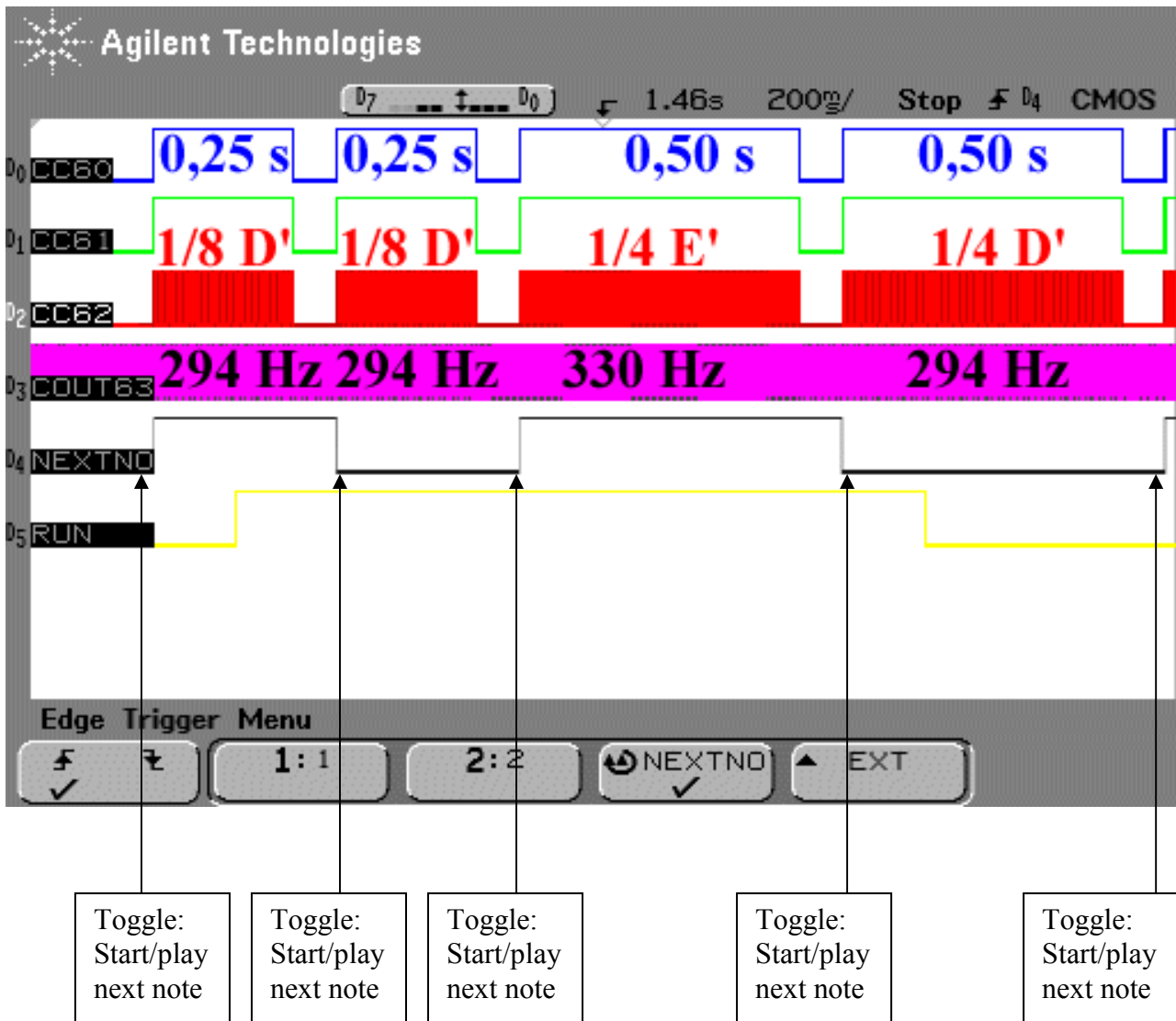
```
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C  
CO0L4HGAL2G";
```



 HAPPY BIRTHDAY =
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F...



 HAPPY BIRTHDAY =
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F...



Note:

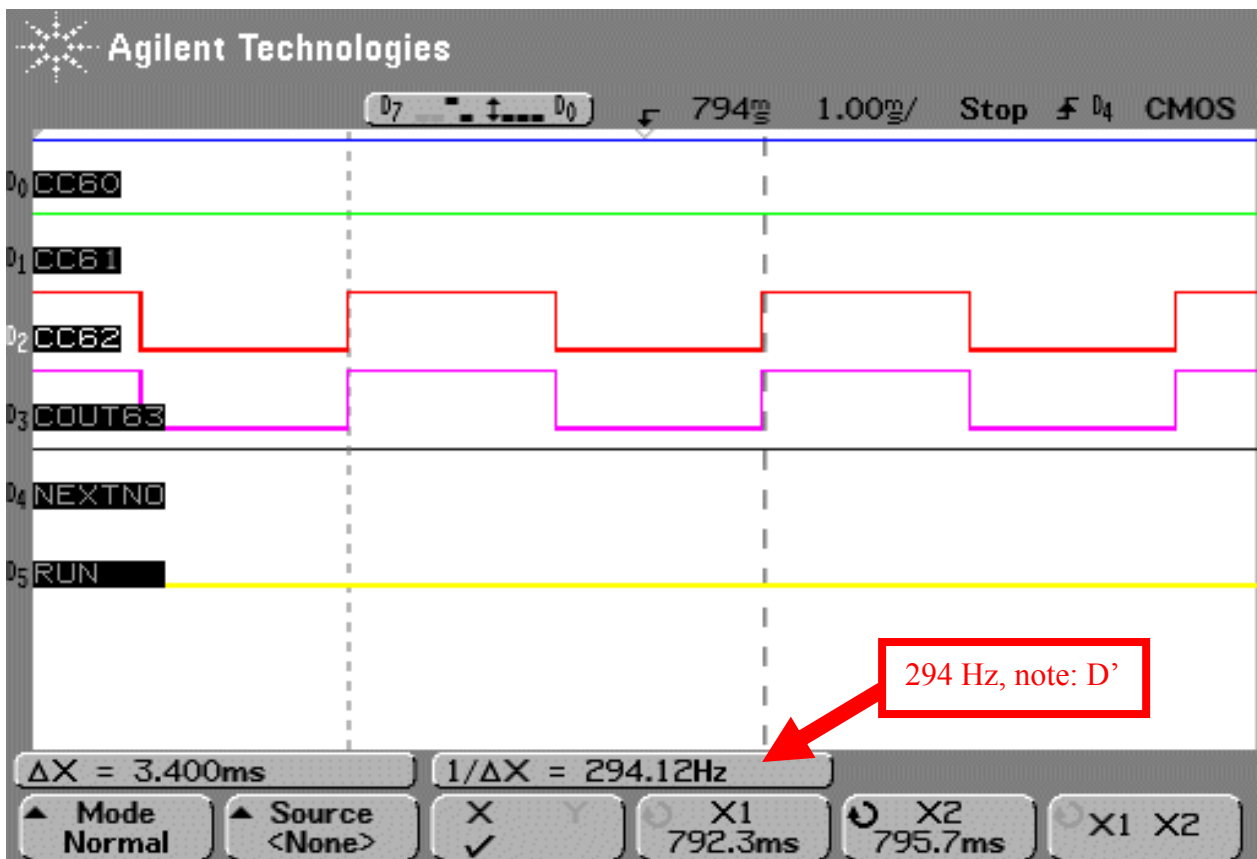
Toggle: Start/play next note:

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

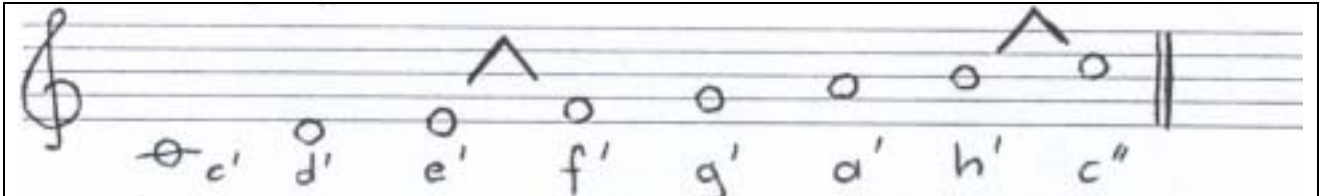
Note:

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

Syntax used in our programming example:

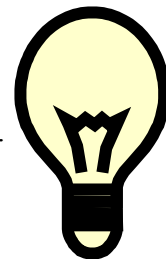
C,D,E,F,G,A,H: play note

Real Music:



Note:

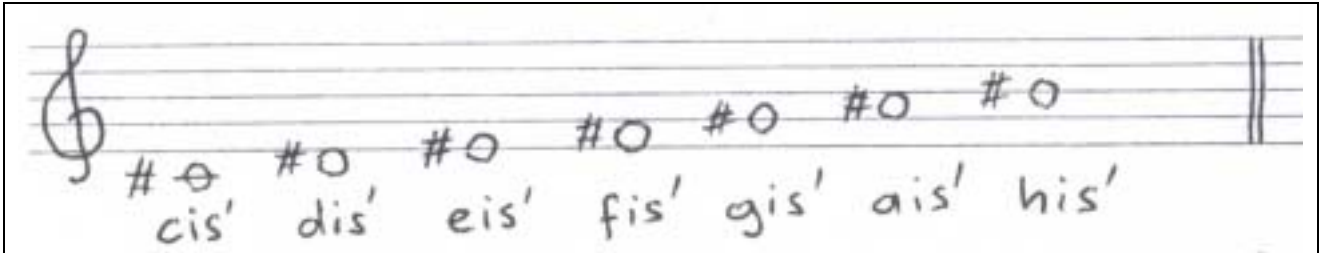
The notes C, D, E, F, G, A, H are named C, D, E, F, G, A, B in other countries.
In this document we stick to the German names.



Syntax used in our programming example:

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

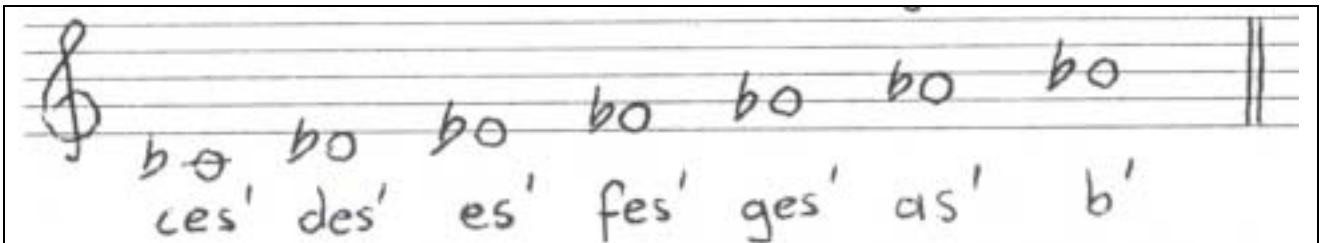
Real Music:



Syntax used in our programming example:

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











Real Music:



Syntax used in our programming example:

Px : play rest/pause/interval of silence
(x = 1,2,4,8,16 -> 1=whole-rest, 2=half-rest, 4=quarter-rest, 8=Eighth-rest, 16=16th-rest)

Real Music:

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

Note:

The realisation of our programming example is easier when we deal with rests as notes.
Therefore, playing a rest means playing a note.
The frequency of the note which is a rest was chosen above our hearing threshold level
(e.g. 60.000 Hz).

Octave:

Definition:

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

Note:

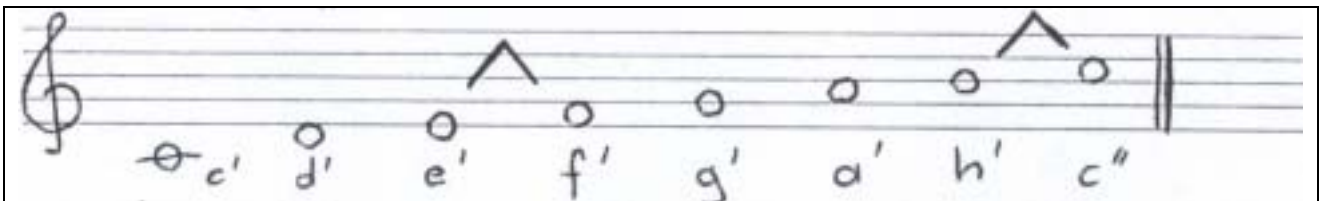
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

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

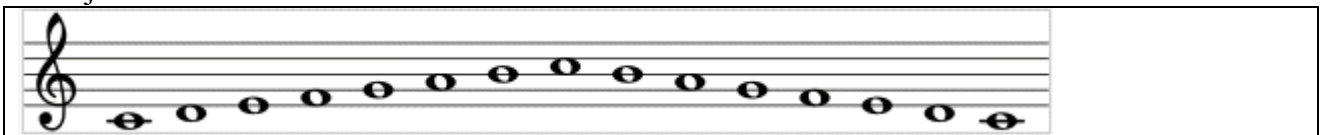
Syntax used in our programming example:

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

Real Music:

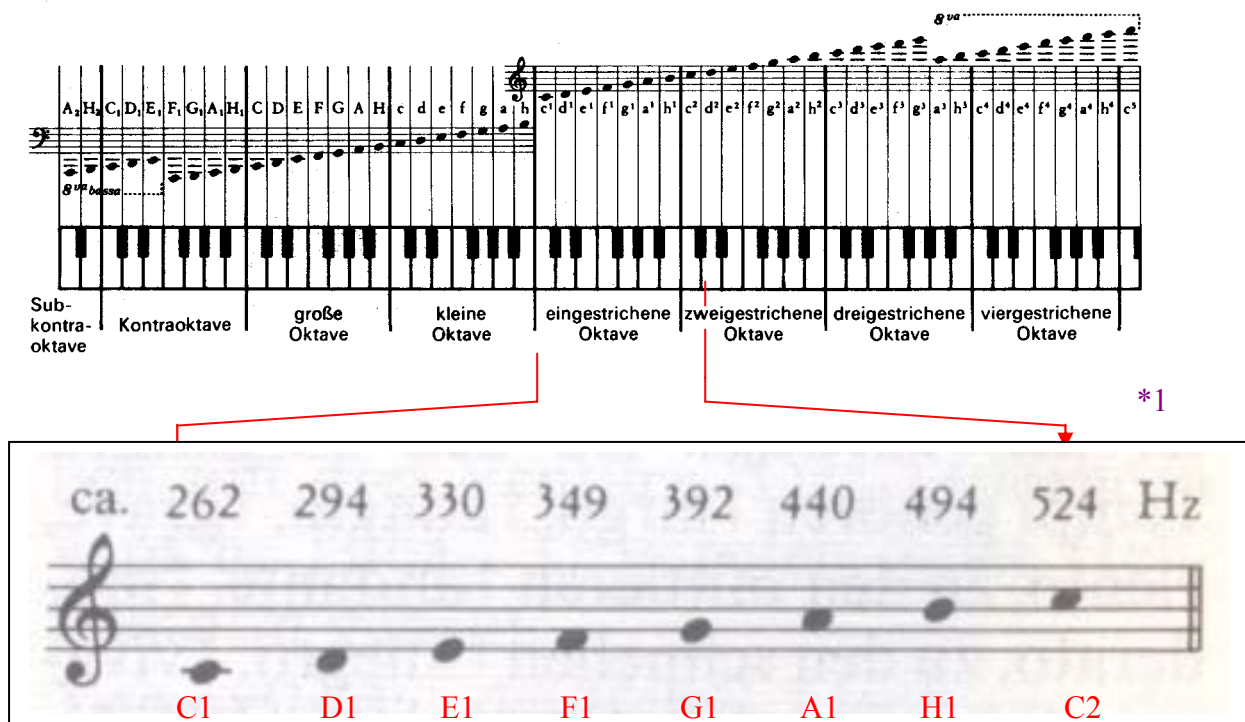


C major scale:



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:



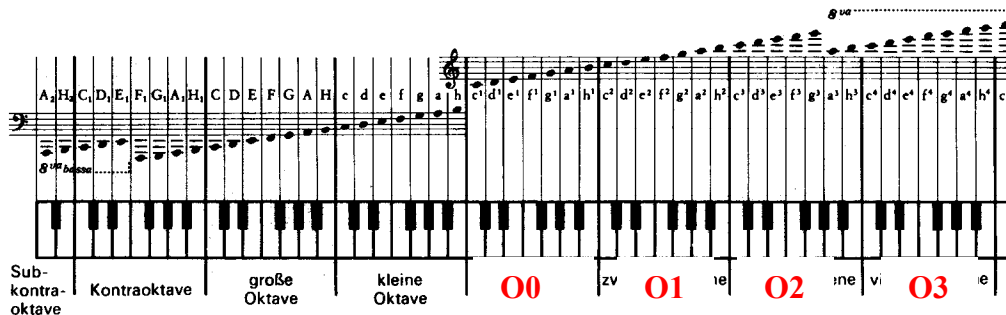
*2

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 = $1/(f_{clk}/2) = 1/(24MHz/2) = 83,333 \text{ ns}$]:

		Octave=0 (=') scaler for T13- Period- value =1	Octave=1 (='') scaler for T13- Period- value =2	Octave=2 (=''') scaler for T13- Period- value =4	Octave=3 (=''''') scaler for T13- Period- value =8
T13 period values	note	f [Hz]	f [Hz]	f [Hz]	f [Hz]
T13_values[0] = 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:

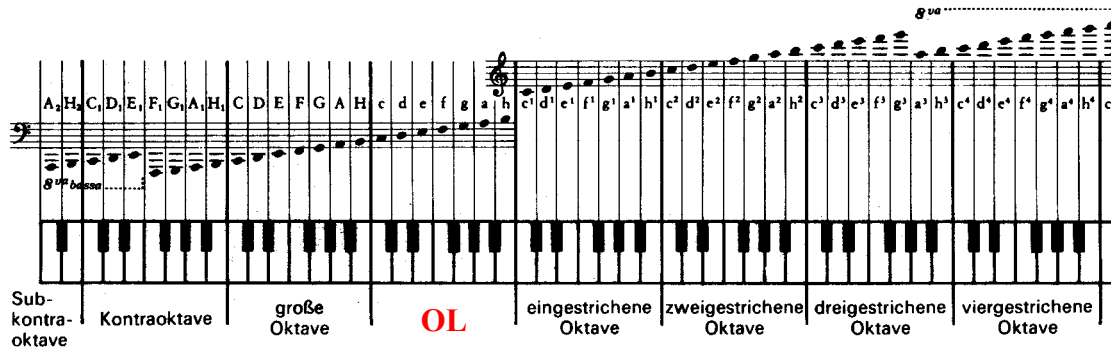
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

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':

$f = 1 / (\text{T13-period-value} \times \text{T13-resolution}) = 1 / (27273 \times 83,333 \text{ ns}) = 440 \text{ Hz}$

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



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

```
unsigned int T13_values[] =
{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 = $1/(f_{clk}/4) = 1/(24MHz/4) = 166,6667 \text{ ns}$]:

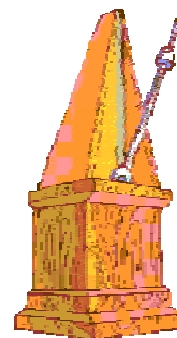
		Octave=OL T13 Prescaler=4	Octave=ON00 T13 Prescaler=2
T13 period values	note	f [Hz]	f [Hz]
T13_values[0] = 45802	c	131	262
T13_values[1] = 43309	cis	139	
T13_values[2] = 40816	d	147	294
T13_values[3] = 38590	dis	156	
T13_values[4] = 36364	e	165	330
T13_values[5] = 34383	f	175	349
T13_values[6] = 32389	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)

○	=	1/1	Note	(= 4 Schläge) (= 4 beats)
♩	=	1/2	Note	(= 2 Schläge) (= 2 beats)
♪	=	1/4	Note	(= 1 Schlag) (= 1 beat)
♫	=	1/8	Note	(= 1/2 Schlag) (= 1/2 beat)
♫	=	1/16	Note	(= 1/4 Schlag) (= 1/4 beat)



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):

T12 period values	note	note	note length [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 = $85,333 \mu\text{s}$)
 Duty cycle = 100 %

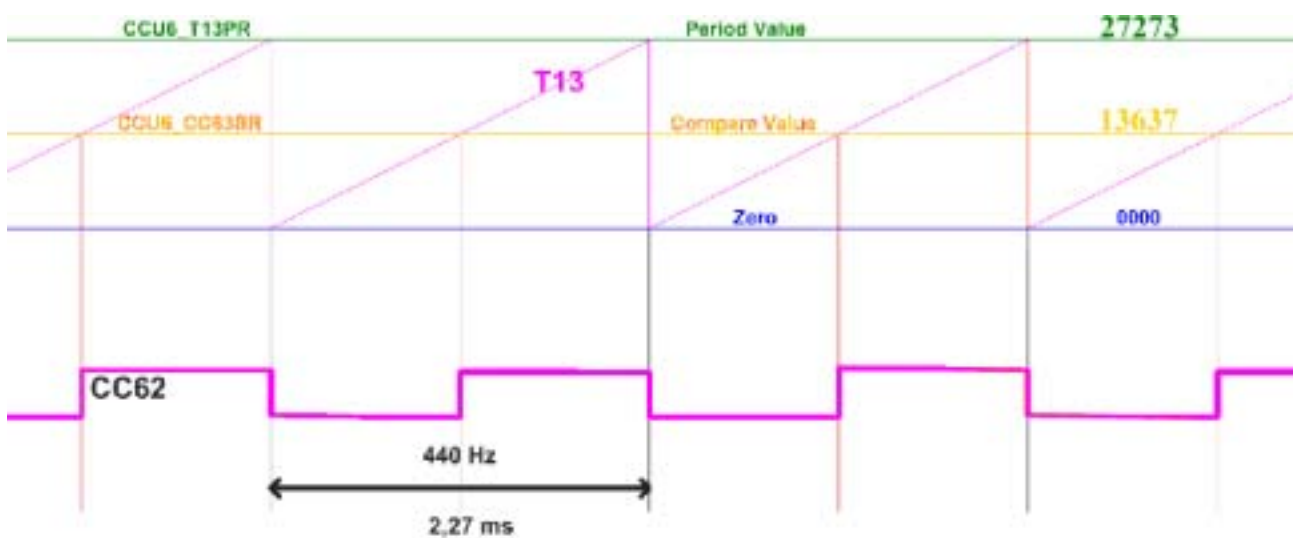
T13: note frequency (Octave = O0, O1, O2 and O3):

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

Französisches Kinderlied



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

schläfst du noch, schläfst du noch?

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

ding, dong, ding, ding dong, ding!

```
// Bruder Jakob (song c):
code unsigned char songc[]="T120O0L4FGAFFGAFDAH-O1L2CO0L4AH-O1L2CL8CDDCO0L8H-
L4AF01L8CDDCO0L8H-L4AFFCL2FL4FCL2F";
```

5.4.) Song d: Happy birthday:

Happy birthday

Englisches Kinderlied



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

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

hap- py birth- day to you!

// Happy birthday (song d):

code unsigned char

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

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

**TAKE ME HOME,
COUNTRY ROADS**



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

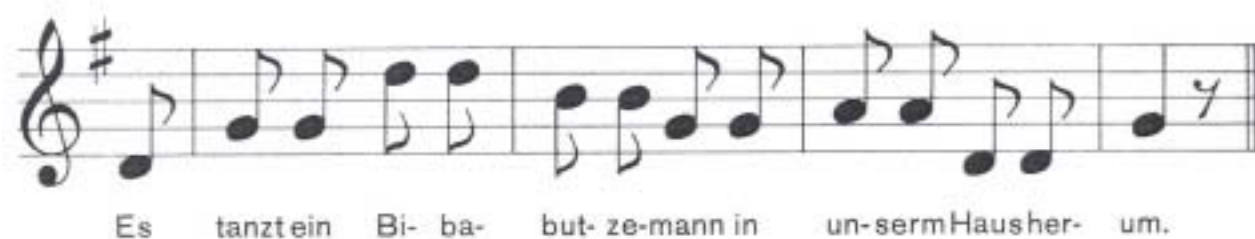
// Take Me Home, Country Roads (song e):

code unsigned char

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

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

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



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



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

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

code unsigned char

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


5.8.) Song h: The little drummer boy:

The little drummer boy



The musical score is written for a guitar and a vocal line. The key signature is one sharp (F#) and the time signature is 4/4. The guitar part consists of a continuous eighth-note accompaniment. The vocal line includes lyrics and drum notation (x) for the drumming. The lyrics are: "Come they told me a - ra-ta-ta - tam, a new-born king to see, a - ra-ta-ta - tam, our fi - nest gifts we bring, a - ra-ta-ta - tam, to lay be - fore the king, a - ra-ta-ta - tam, ra-ta-ta - tam, ra-ta-ta - tam. So to ho-nour Him, a ra-ta-ta - tam, When we come." The score is divided into six systems, each with a vocal line and a guitar line. The drum notation is represented by 'x' marks above the vocal line.

Gesang
Gitarre

Come they told me a - ra-ta-ta - tam,

(alle klatschen)

a new-born king to see, a - ra-ta-ta - tam,

our fi - nest gifts we bring, a - ra-ta-ta - tam,

to lay be - fore the king, a - ra-ta-ta - tam,

ra-ta-ta - tam, ra-ta-ta - tam. So to

ho-nour Him, a ra-ta-ta - tam, When we come.

// The little drummer boy (song h):

code unsigned char

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

5.9.) Song i: Hey, Pippi Langstrumpf:

Hey, Pippi Langstrumpf

A D G A D



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

D G A D



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

B D G A A



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

D G A D Fine



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

C D G A D



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

Hm Em A D A7



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



 raus. Ich hab' ein Haus, ein Äff - chen und ein Pferd und



 je - der, der uns mag, kriegt un - ser Ein - mal - eins ge - lehr't.

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



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

```
code unsigned char
```

```
songi[]="T180OLL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+OLL4AON0  
0L4DF+DL2EL8GF+EDL4C+EOLL4AON00L4C+DP4P2OLL4AON00L4DF+DL2EL8GF+ED  
L4C+EOLAON00L4C+L2DF+OLL4AON00L4DF+DL2EL8GF+EDL4C+EOLL4AON00L4C+  
DP4P2O0L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+  
EEDC+DP4L2F+GAH.O1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+GL  
4AF+GAL2H.O1L4DC+O0L4HAGL2A.O1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+EDP2";
```

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:

Junge komm' bald wieder



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

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

// Junge komm bald wieder (song k):

code unsigned char

```
songk[]="T120O0L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C.L4EEL8D+  
L8EL4F+L4E.L8EL4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+.L8EL4EL2DL4E  
L4D.L8COLL2H.ONO0L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C.L4E  
EL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8DDDDDDDDDL4DP8L8DL8D+L8DDDDDL8D  
+L8DL4DP8L8DL8EEEEEL2GP8L8EL1DP8L8DL8EEEL4E.P8L8GGGF+L8GL1A.";
```

5.12.) Song 1: Lili Marleen:

Lili Marleen



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

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

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

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

// Lili Marleen (song 1):

code unsigned char

```
song1[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL  
8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL  
4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HON00L2C.";
```

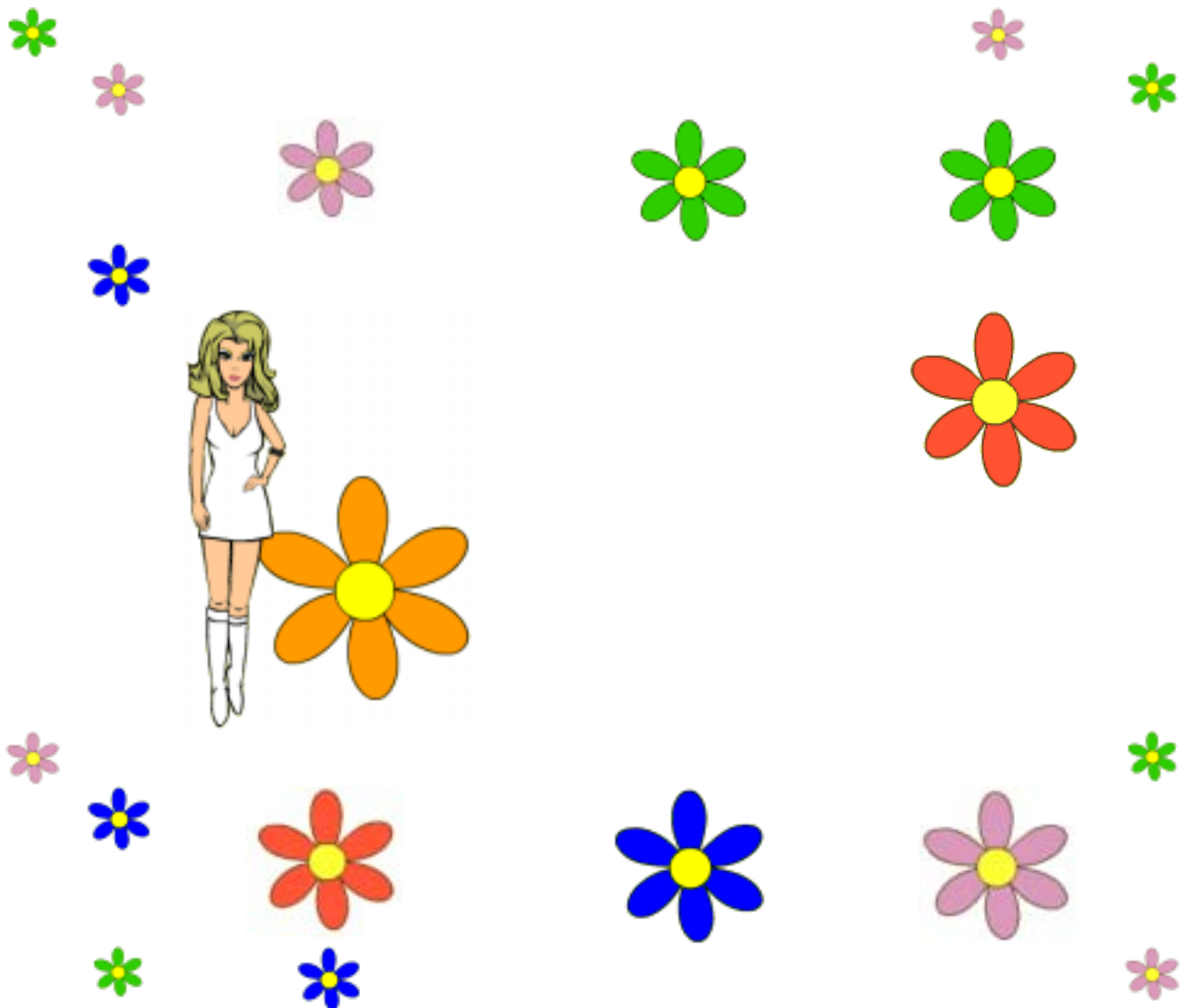
5.13.) Song m: musical scale / chromatic scale / for testing purpose / Tonleiter:

// 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+.P16OLL8AP16
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.

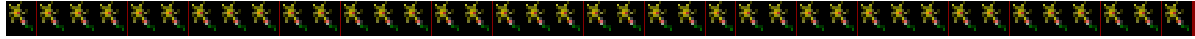
6.) Thanks To



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>