

AP08068

XC886/XC888

Playing music using the CAPCOM6 module.



Microcontrollers



Never stop thinking

Edition 2008-07-11

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

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

LEGAL DISCLAIMER

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

Information

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

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: 2007-11 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 “Appnote” is an Infineon 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 AP08067.

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

The main chapter is:
[Chapter 3](#): Playing Music.

Note:

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

Have fun and enjoy the CAPCOM 6 module!

Note:

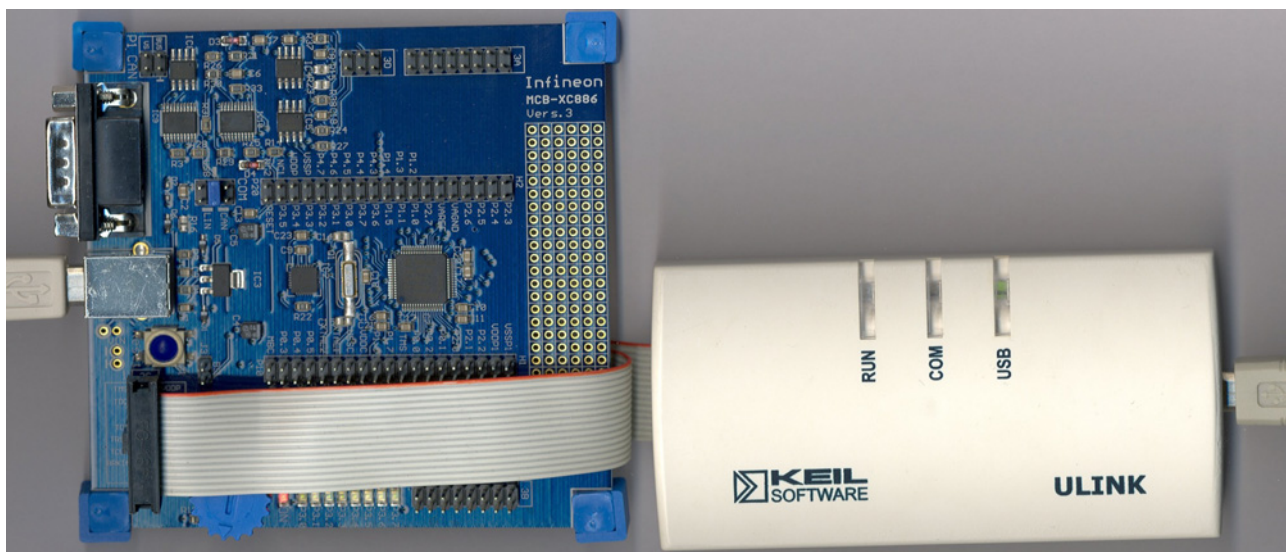
Additionally, there is a step-by-step-book (AP16109) focusing on BLDC-Motors available, which can be used for all 8/16 and 32 bit microcontrollers equipped with the CAPCOM 6 module. To get the most out of the CAPCOM 6 module this additional Cookery-Book is the icing on the cake of all available functionalities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



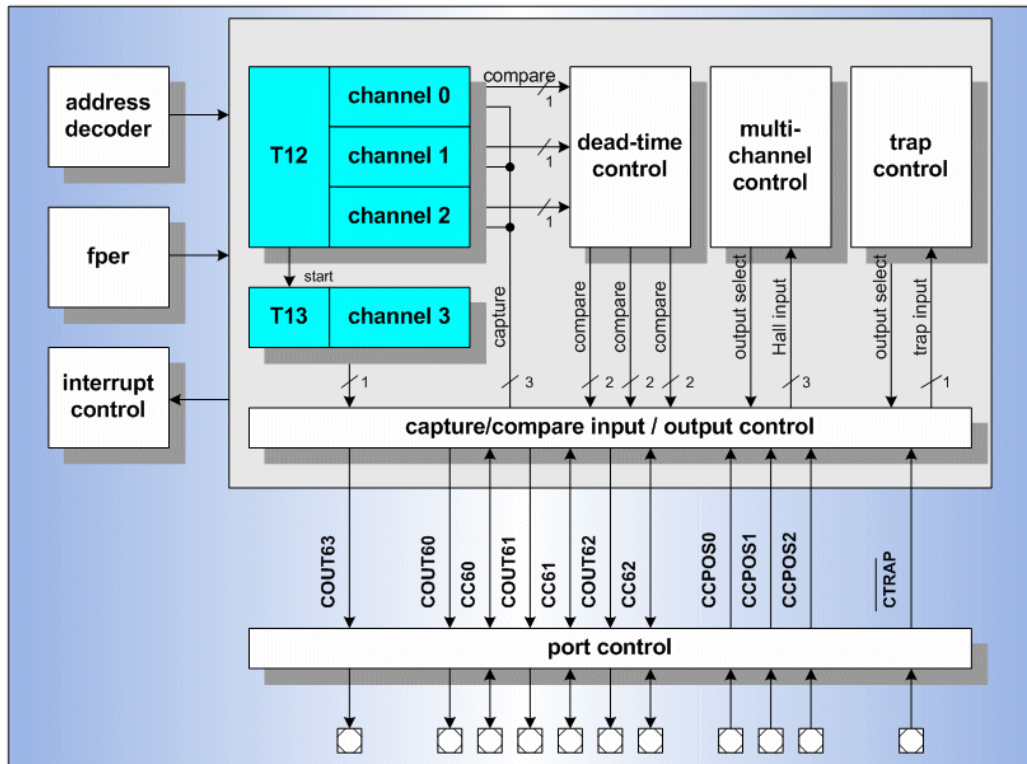
The New
XC800-Family
Designed to Make the Difference

Programming Examples

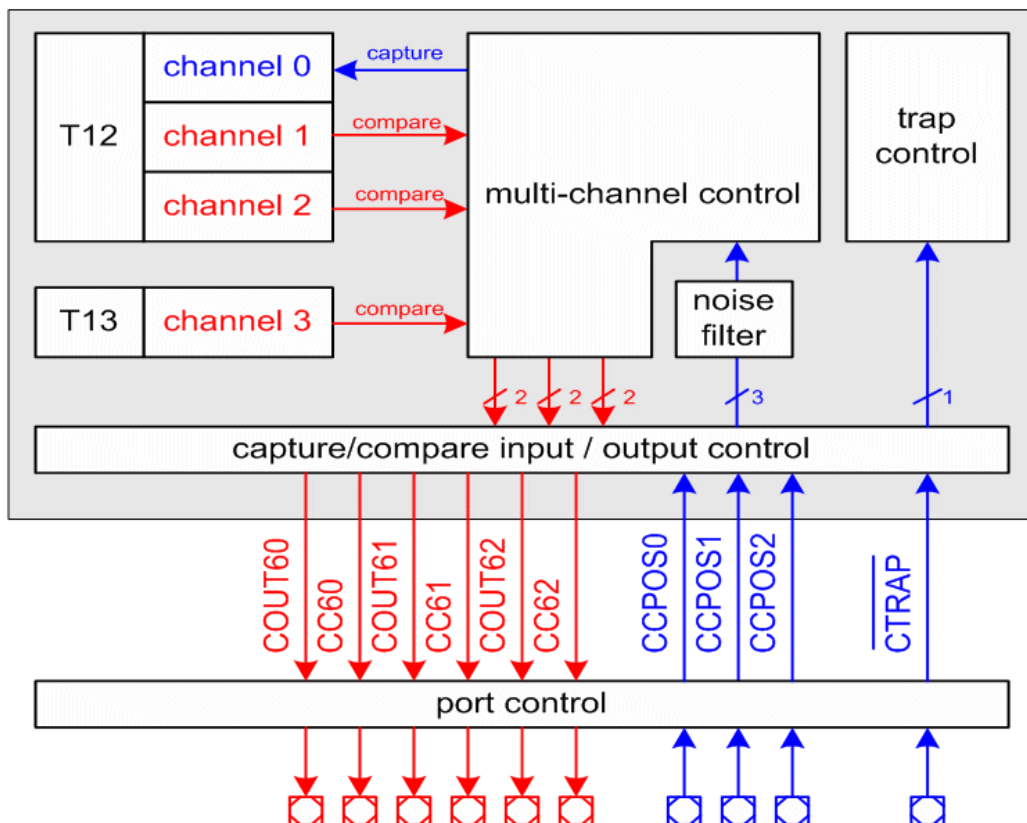
XC888CM-8FFA



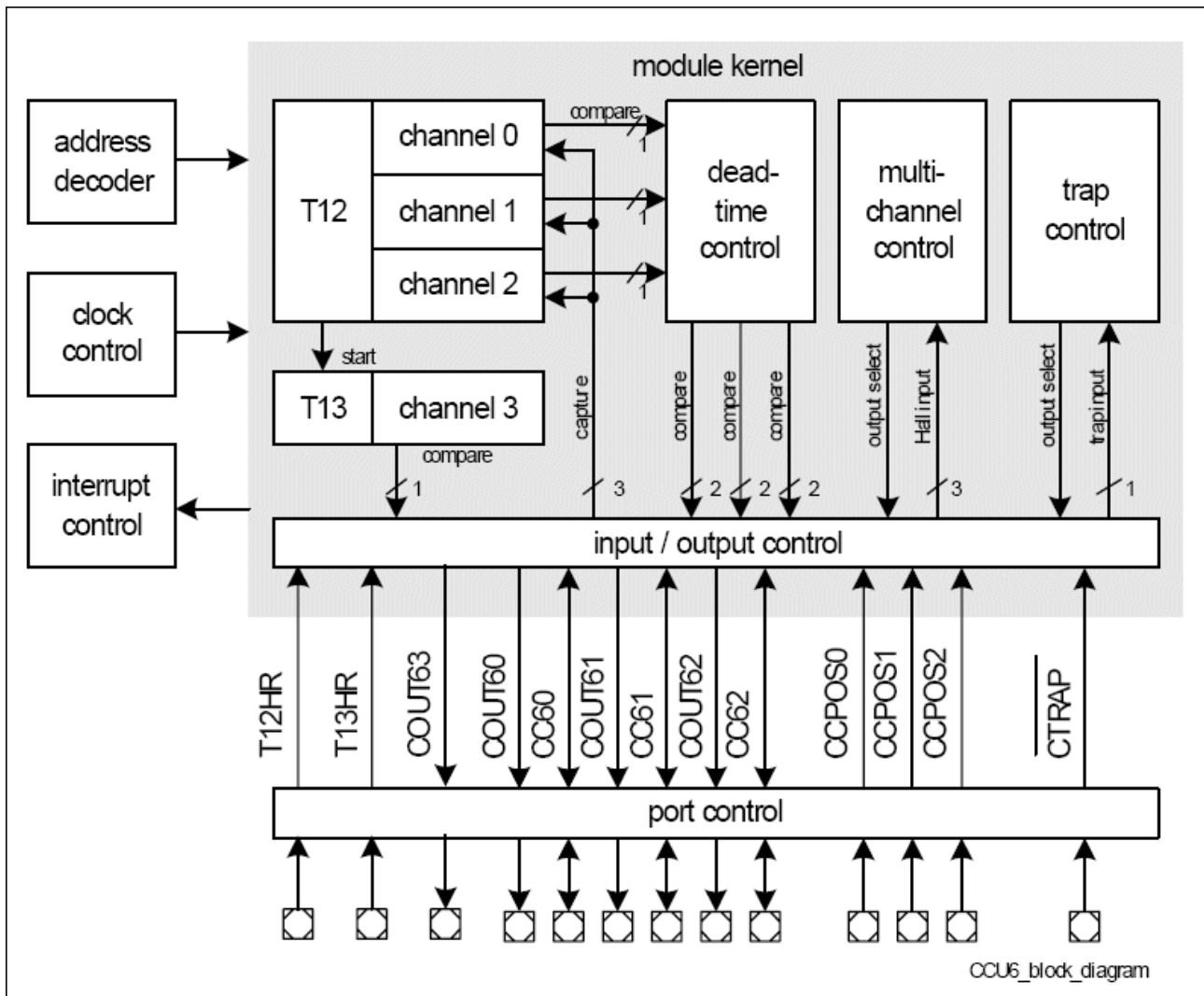
CAPCOM 6 Block Diagram – general use (Source: Product Marketing)



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



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

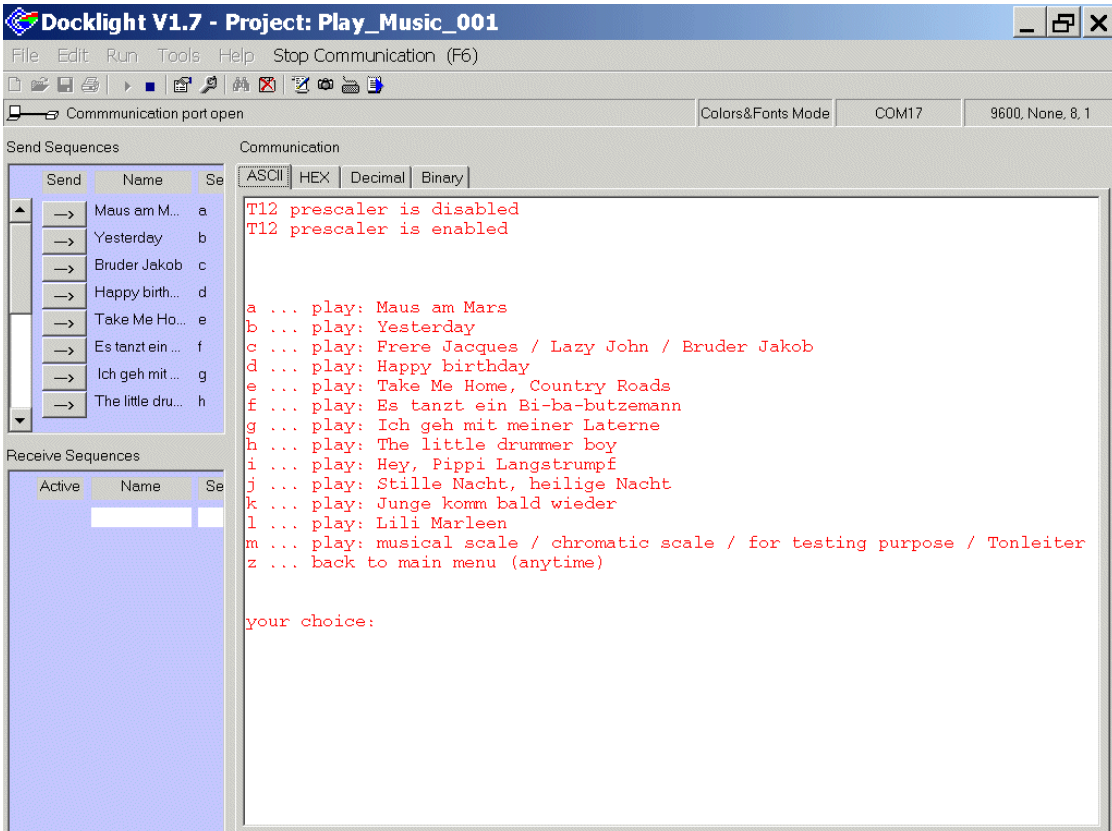


Note:

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 examples for the CCU6:

<p>Your program:</p>	
<p>Chapter/ Step</p>	<p>*** Recipes ***</p>
<p>1.)</p>	<p><u>Asymmetrical / Edge-Aligned PWM generation using CCU6</u></p>
<p>2.)</p>	<p><u>Symmetrical / Center-Aligned PWM generation using CCU6</u></p>
<p>3.)</p>	<p><u>Asymmetrical / Edge-Aligned PWM generation Single Shot Mode (Timer12), Modulation (Timer13) Playing music</u></p>
<p>4.)</p>	<p><u>PERFECTIONISTS ONLY!</u></p>

Appendix:

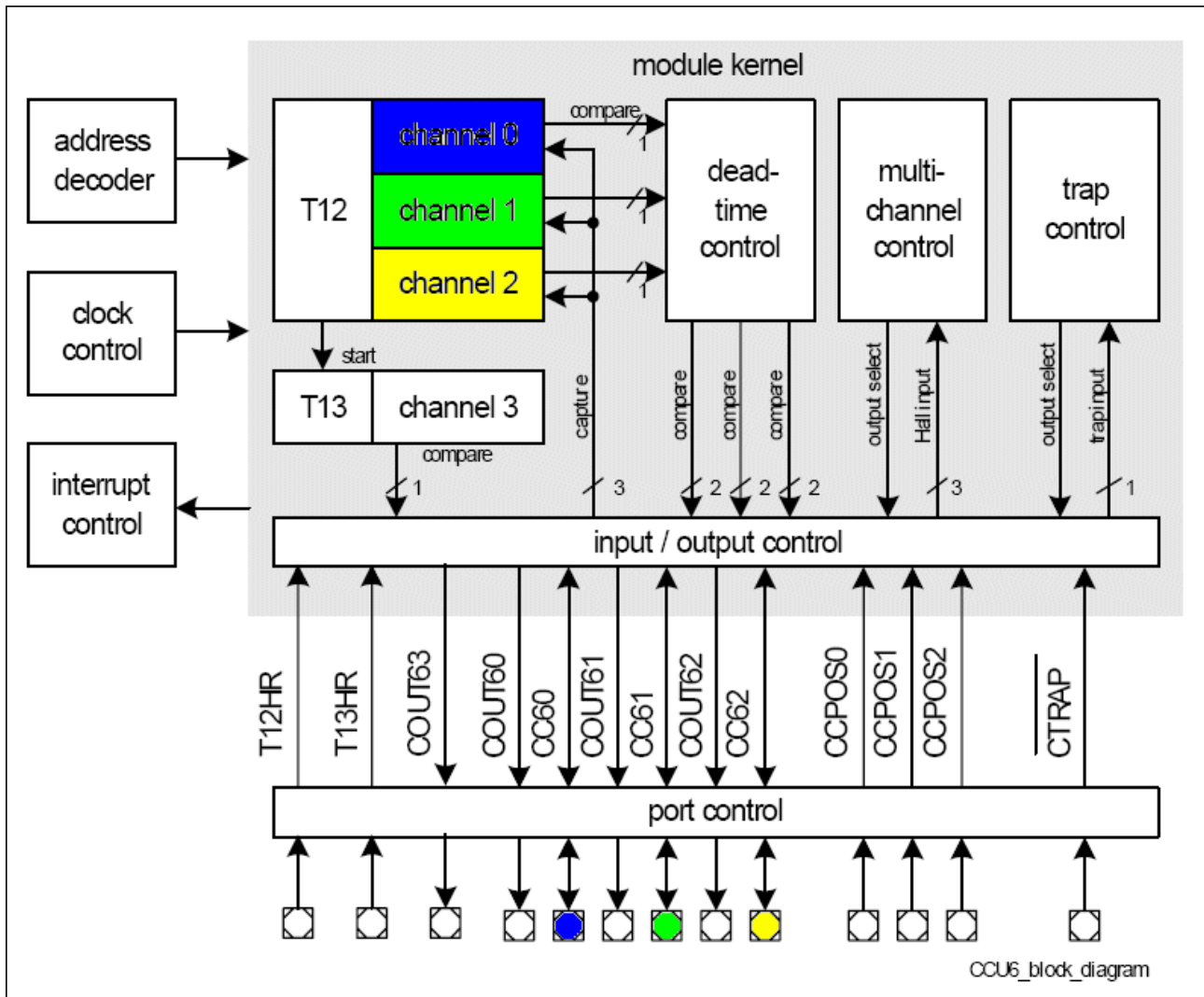
Chapter/ Step	*** Recipes ***
5.)	<u>Appendix: about music (note length, note frequency)</u>
6.)	<u>Appendix: CCU6 use to create note length and note frequency</u>
7.)	<u>Appendix: songs used</u>

Feedback:

8.)	<u>Thanks To</u>
9.)	<u>Feedback</u>

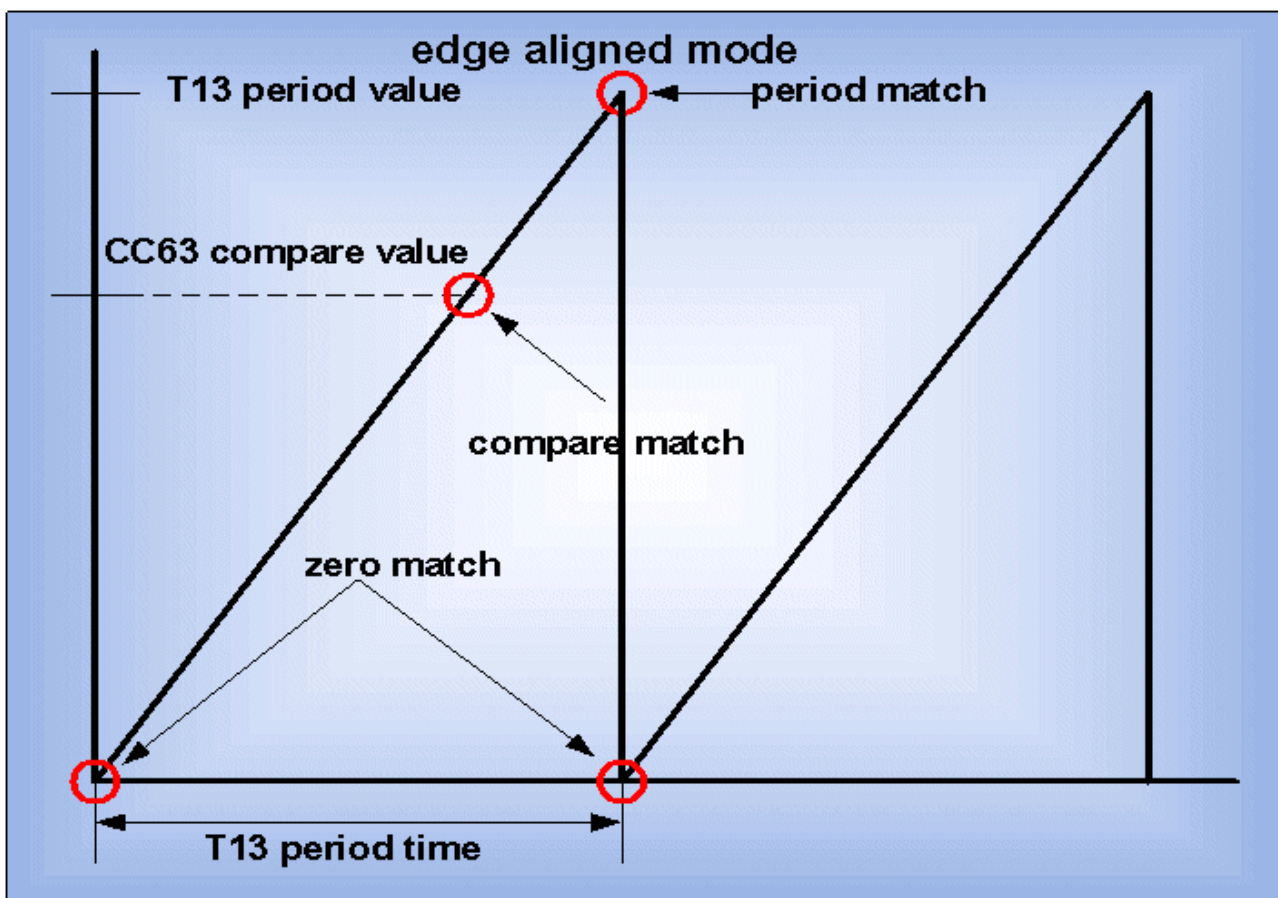
1.) Asymmetrical / Edge-Aligned PWM generation using CCU6

Used channels for PWM generation (blue, green, black/yellow):





Example of an Asymmetrical / Edge-Aligned PWM-signal:

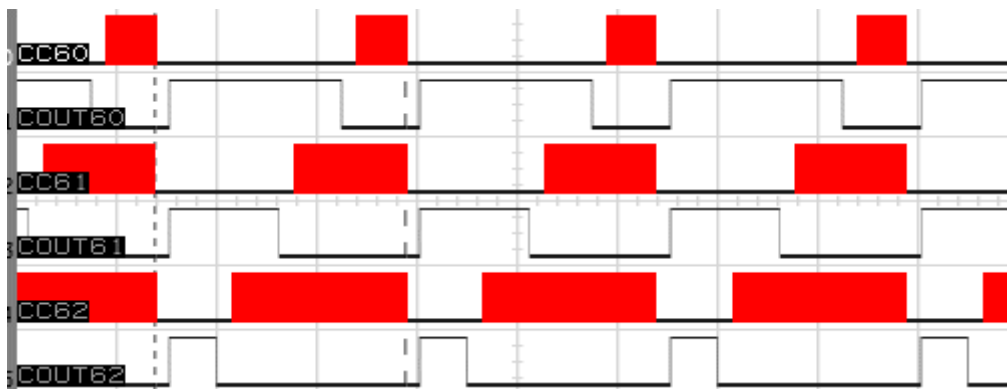


Note:

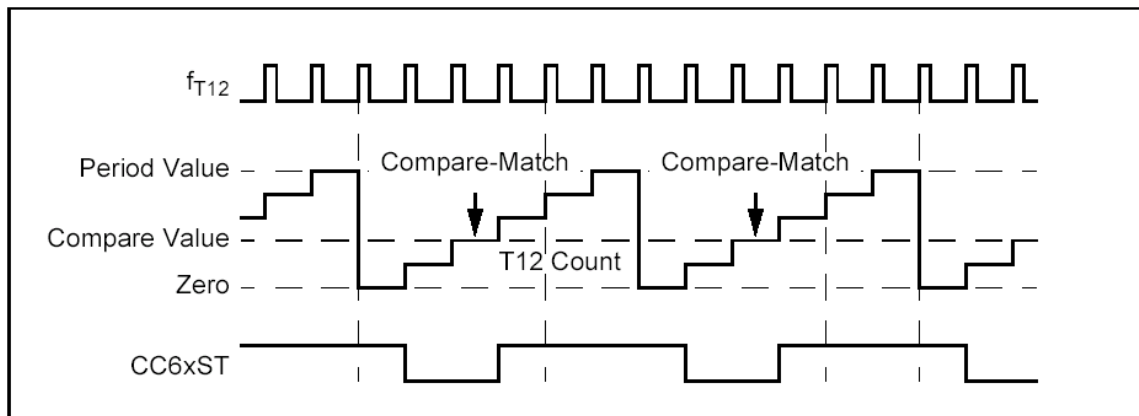
Ignore Timer 13 in the picture above.
The purpose of this picture is to explain asymmetrical PWM signals.
We are going to use only Timer 12 in chapter 1 and chapter 2.



Example of 3 (6) Asymmetrical / Edge-Aligned PWM-signals:



Example of a Compare Event:



Note:

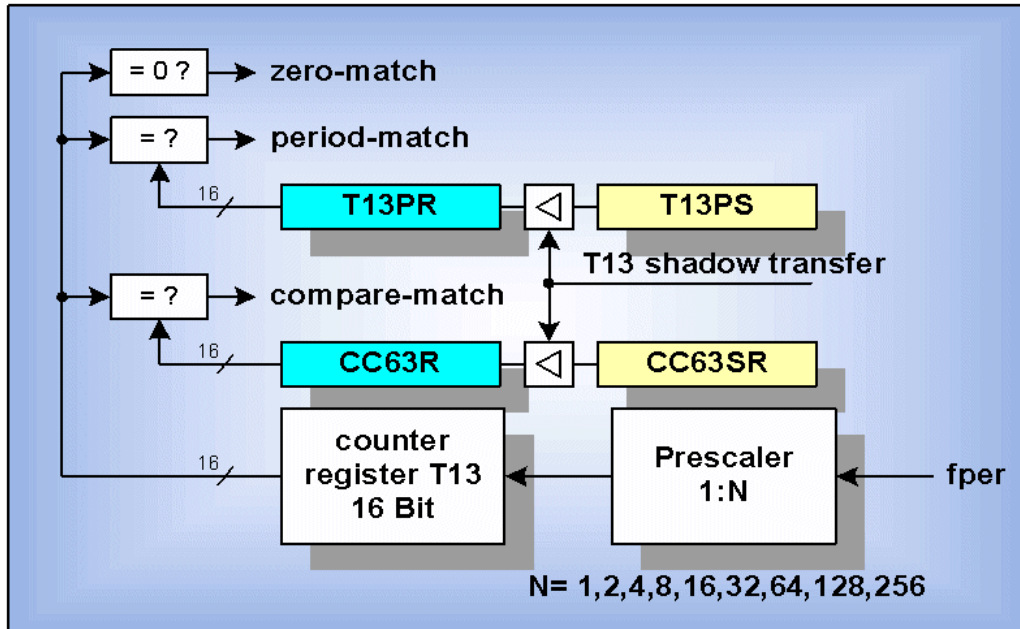
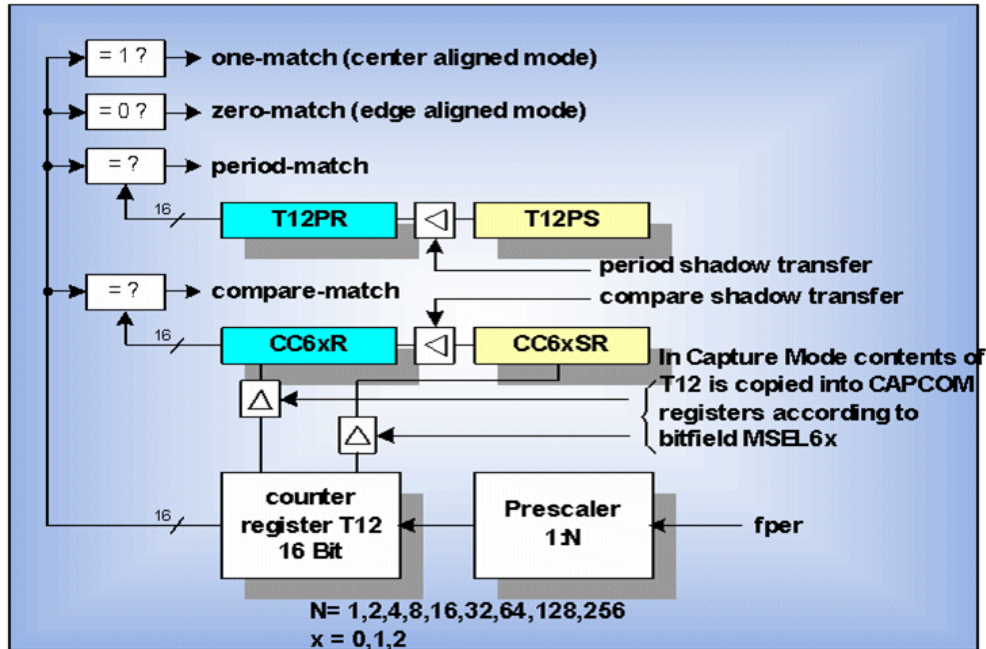
The bit CC6xST indicates a compare event.

CC6xST **Set**: T12 counter value above the compare value.

CC6xST **Reset**: T12 counter value below the compare value



Accessing CCU6 registers (period, compare value):



Note (Source: User's Manual):

The timer period and the compare values are written to shadow registers and not directly to the actual registers, while the read access targets the registers actually used (except for the three compare channels, where both the actual and the shadow registers can be read).

The transfer from the shadow registers to the actual registers is enabled by setting the shadow transfer enable bit STE12/STE13.



Note:

First we are going to generate the following Asymmetrical / Edge-Aligned PWM-signals.

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75

Additionally, we intend to show more information on the **Logic Analyser (LGA)** Screenshots (e.g. **one match**, **period match**).

GPIO use:

Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR

Note:

The input clock of the CCU6 module is not important at this stage.

Do the XC886/XC888 Cookery Book:



Note:

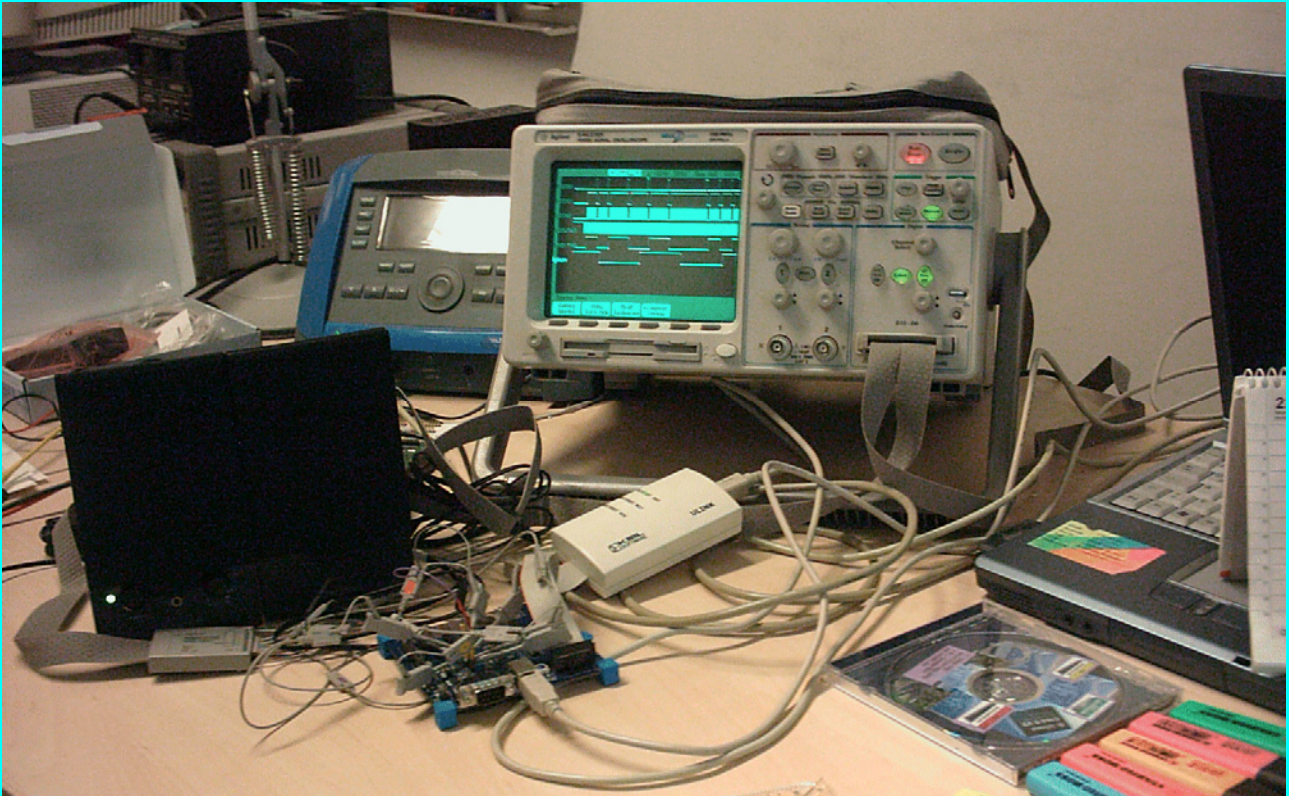
It is necessary to follow all instructions in the XC886/XC888 Cookery Book (AP08067) step by step, as this is the basis for all instructions which will follow later.



Note:

In the following steps of this document we will expand the “Hello World Application” (Application Note AP08067) with the requirements for PWM generation.

Let's Get Started:



Configuring and Reconfiguring the DAVe Project Settings:



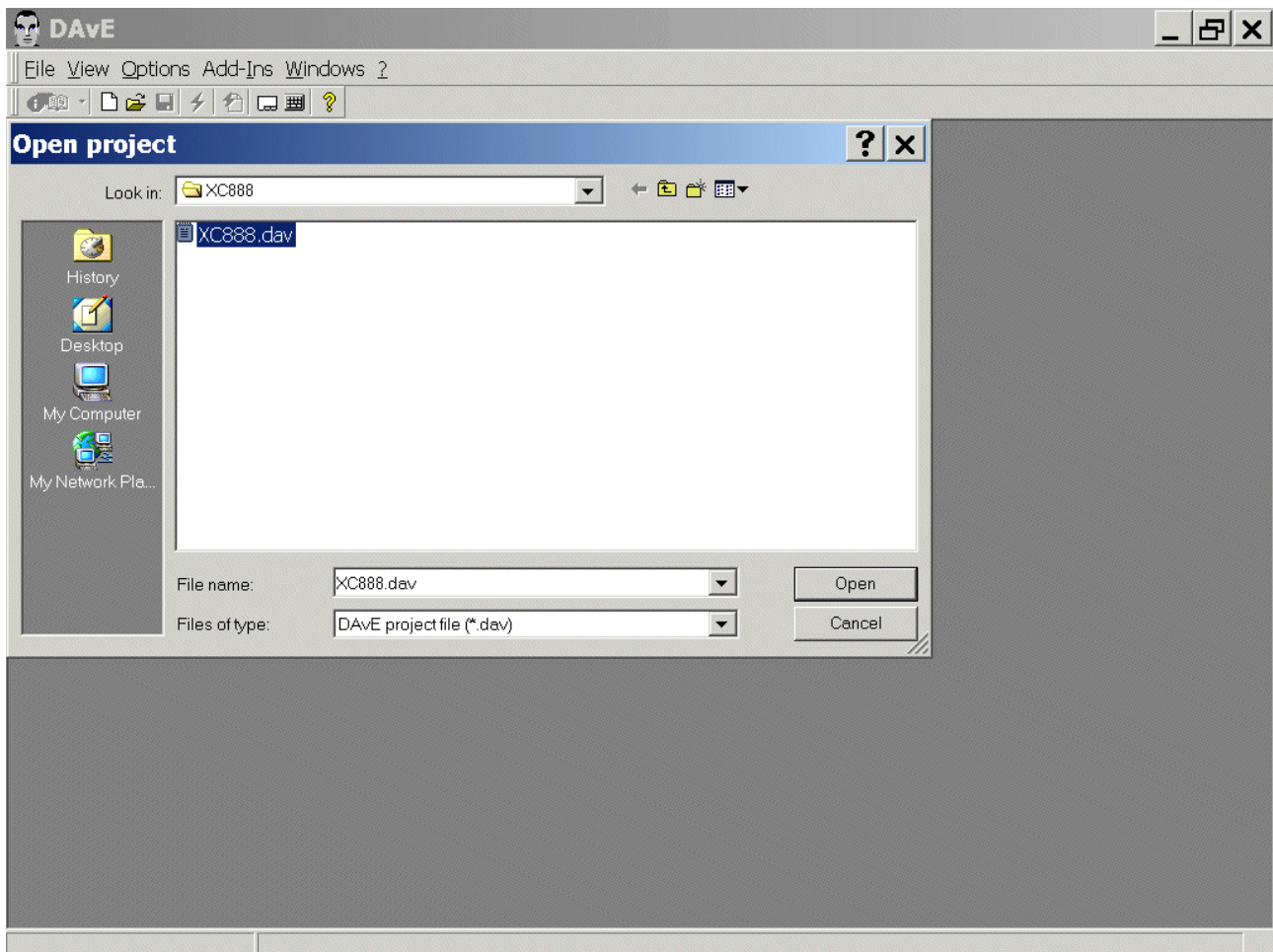
Start the program generator DAvE and open your [XC888.dav](#) DAvE project:

File

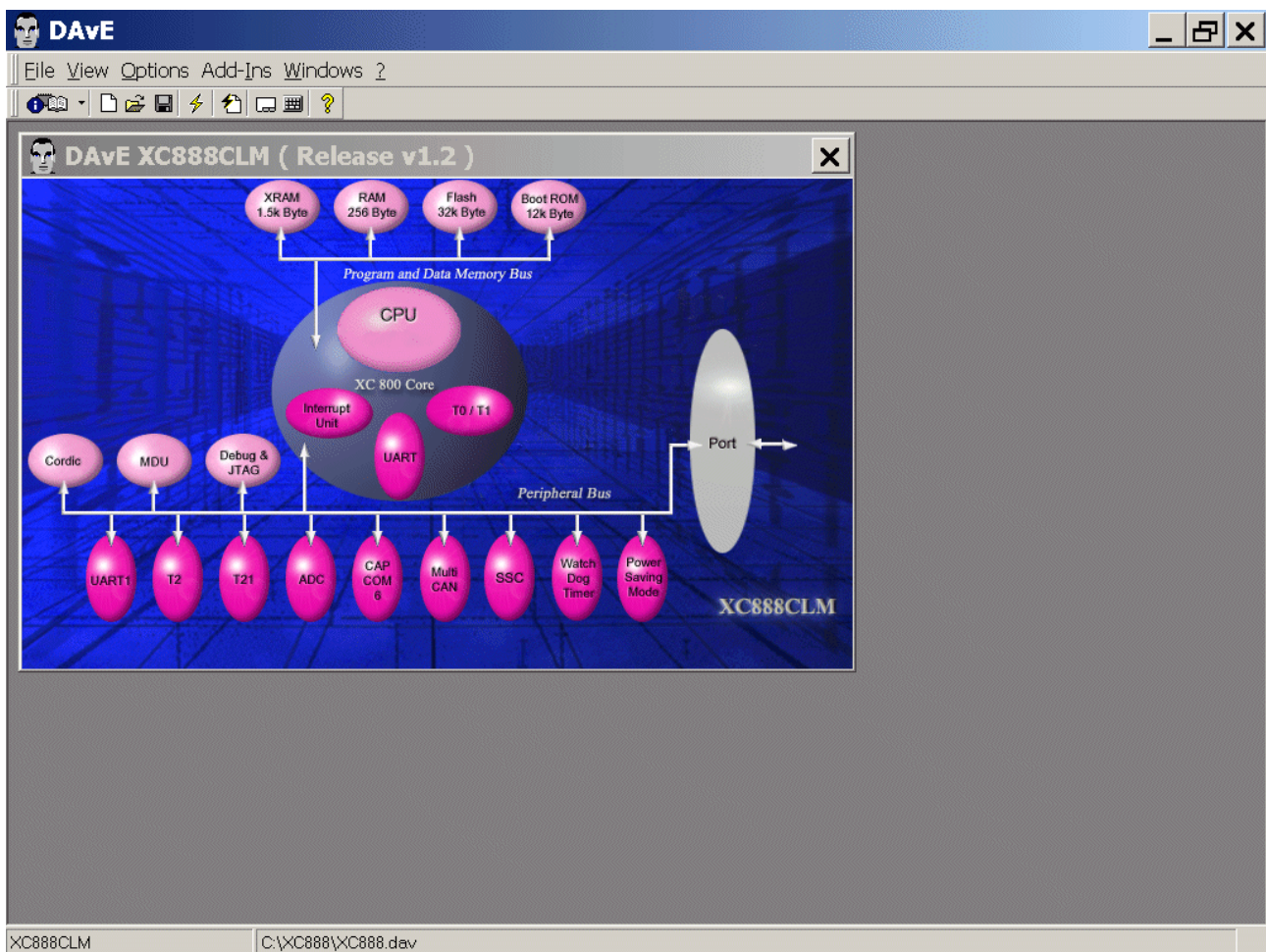
Open

Location: [C:\XC888](#)

Filename: [XC888.dav](#)

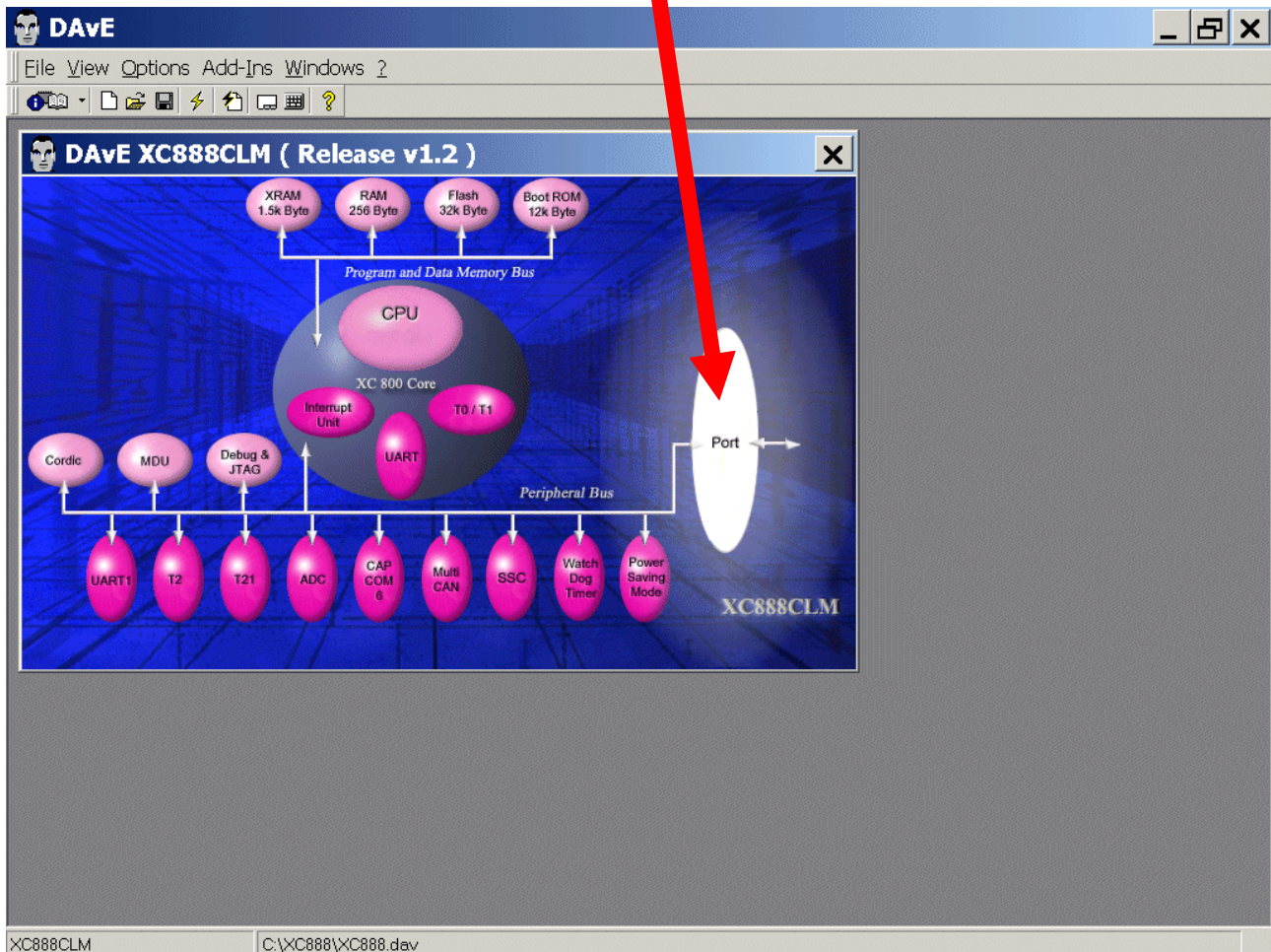


Click Open

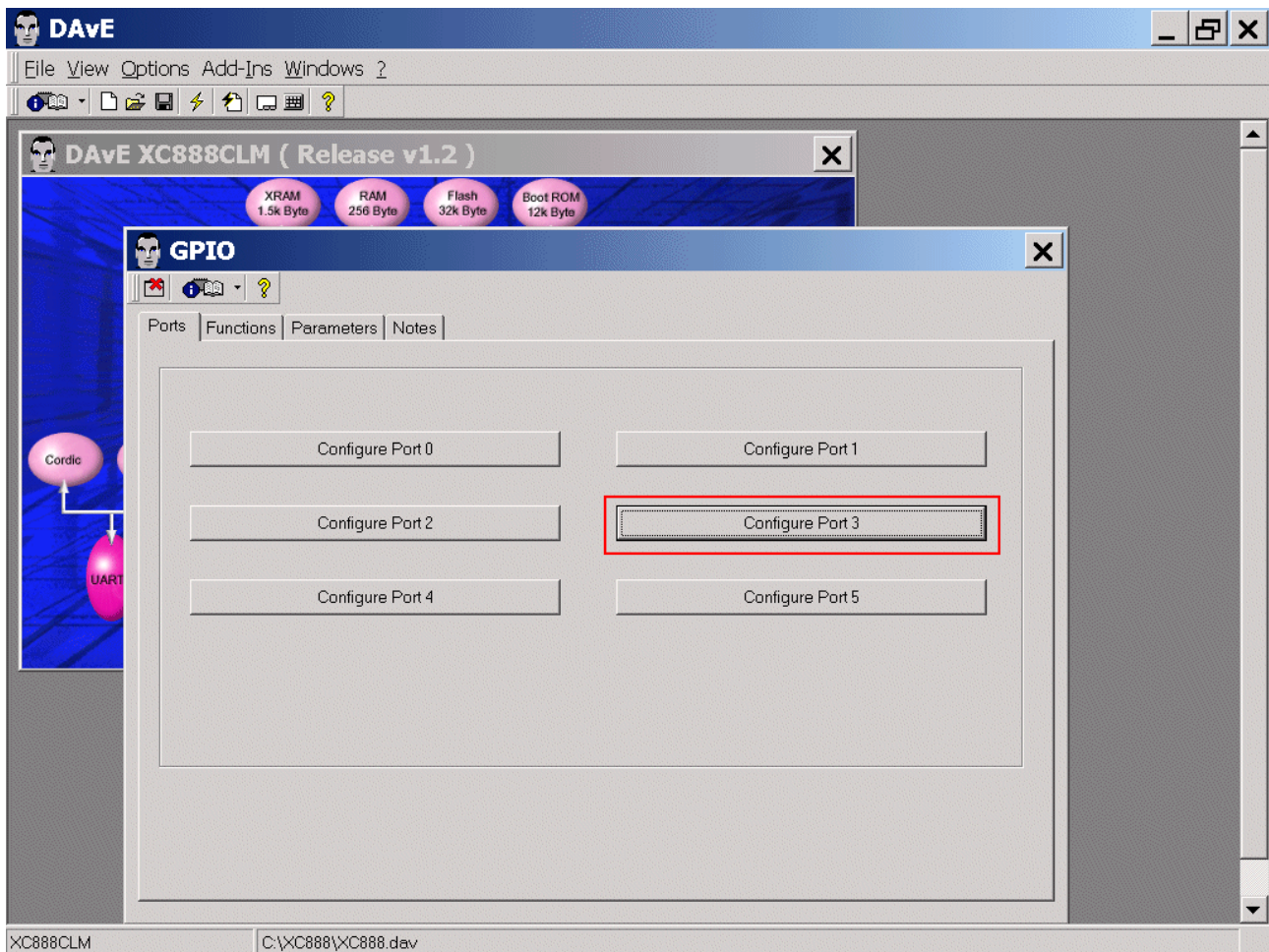


Reconfiguration of Port 3:

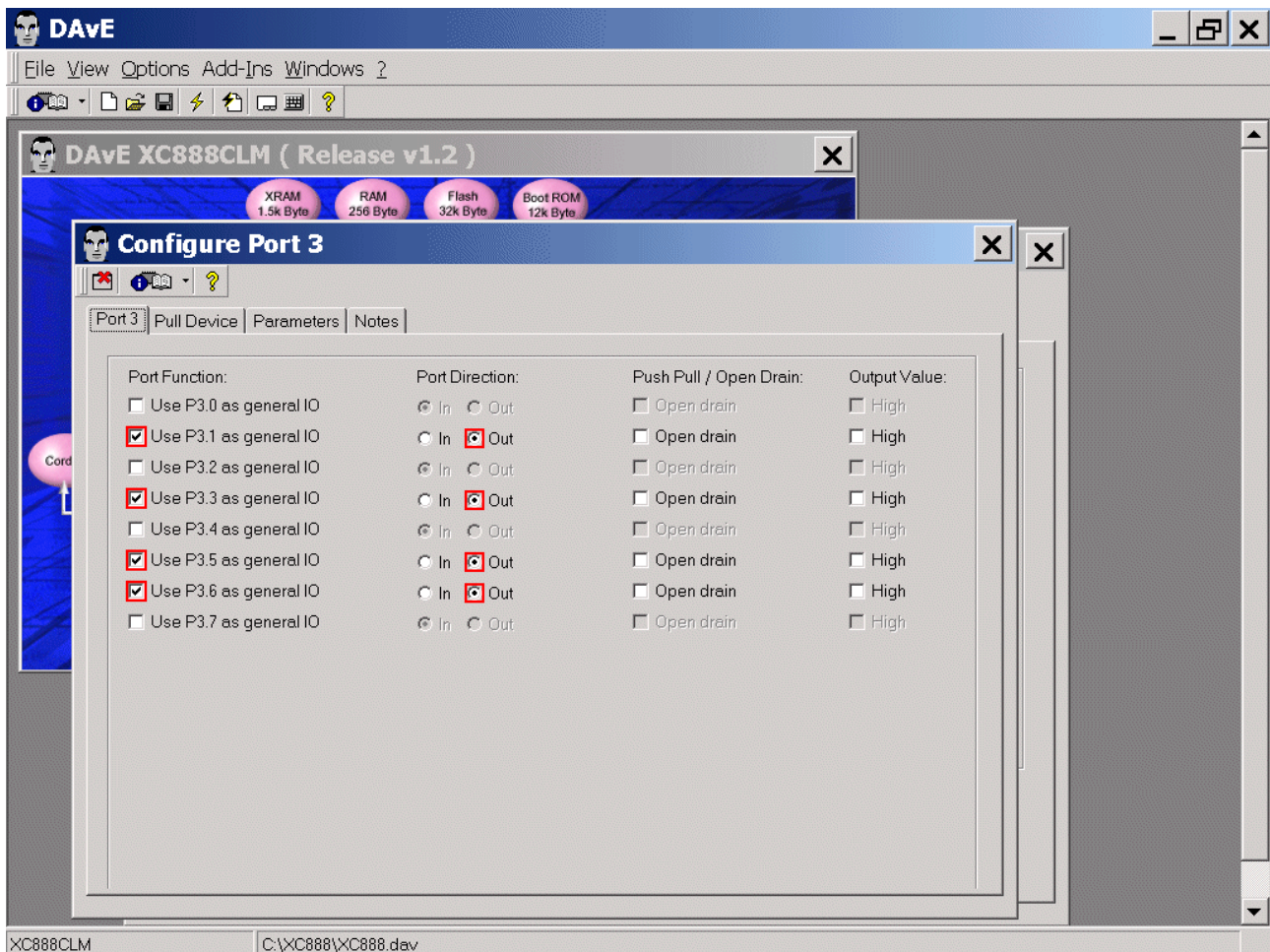
The (re)configuration dialog can be opened by clicking the specific block/module.



Ports: click “Configure Port 3”



Port 3: Port Function: [click to unselect](#) ☐ Use P3.0 to free GPIO pin P3.0 for CCU6 use
 Port 3: Port Function: [click to unselect](#) ☐ Use P3.2 to free GPIO pin P3.2 for CCU6 use
 Port 3: Port Function: [click to unselect](#) ☐ Use P3.4 to free GPIO pin P3.4 for CCU6 use
 Port 3: Port Function: [click to unselect](#) ☐ Use P3.7 to free GPIO pin P3.7 for CCU6 use
 Port 3: Port Function: [click/check](#) ☒ Use P3.1 as general IO - Port Direction: [click](#) ☒ Out
 Port 3: Port Function: [click/check](#) ☒ Use P3.3 as general IO - Port Direction: [click](#) ☒ Out
 Port 3: Port Function: [click/check](#) ☒ Use P3.5 as general IO - Port Direction: [click](#) ☒ Out
 Port 3: Port Function: [click/check](#) ☒ Use P3.6 as general IO - Port Direction: [click](#) ☒ Out





Note:

Port pins used by our PWM module (not available as GPIO pins, directly controlled by CCU6):

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75
P3.7	COU63_0	not used

GPIO use:

Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.5	not used	
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

Pull Device: (do nothing)

Parameters: (do nothing)

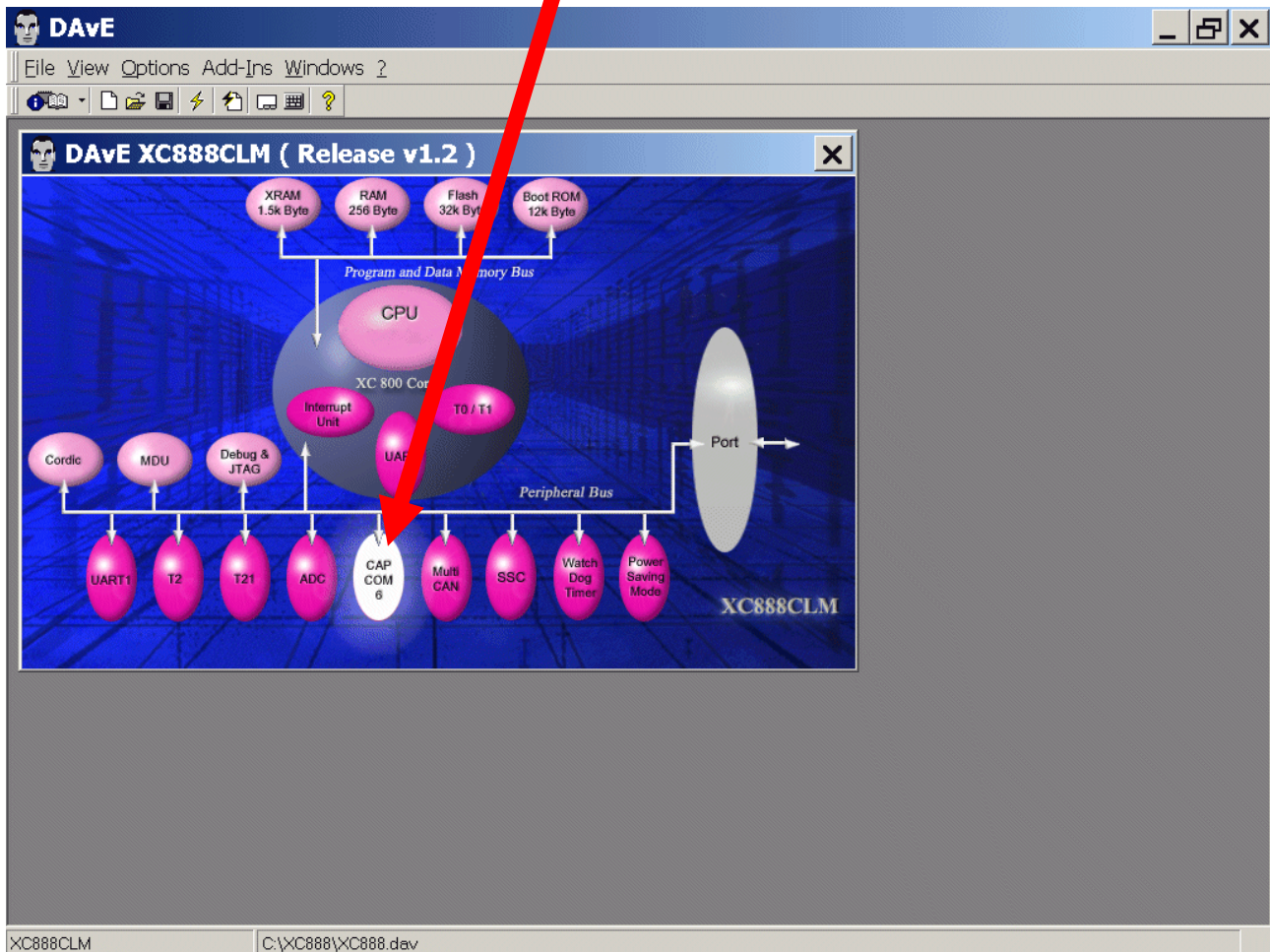
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking  the close button.

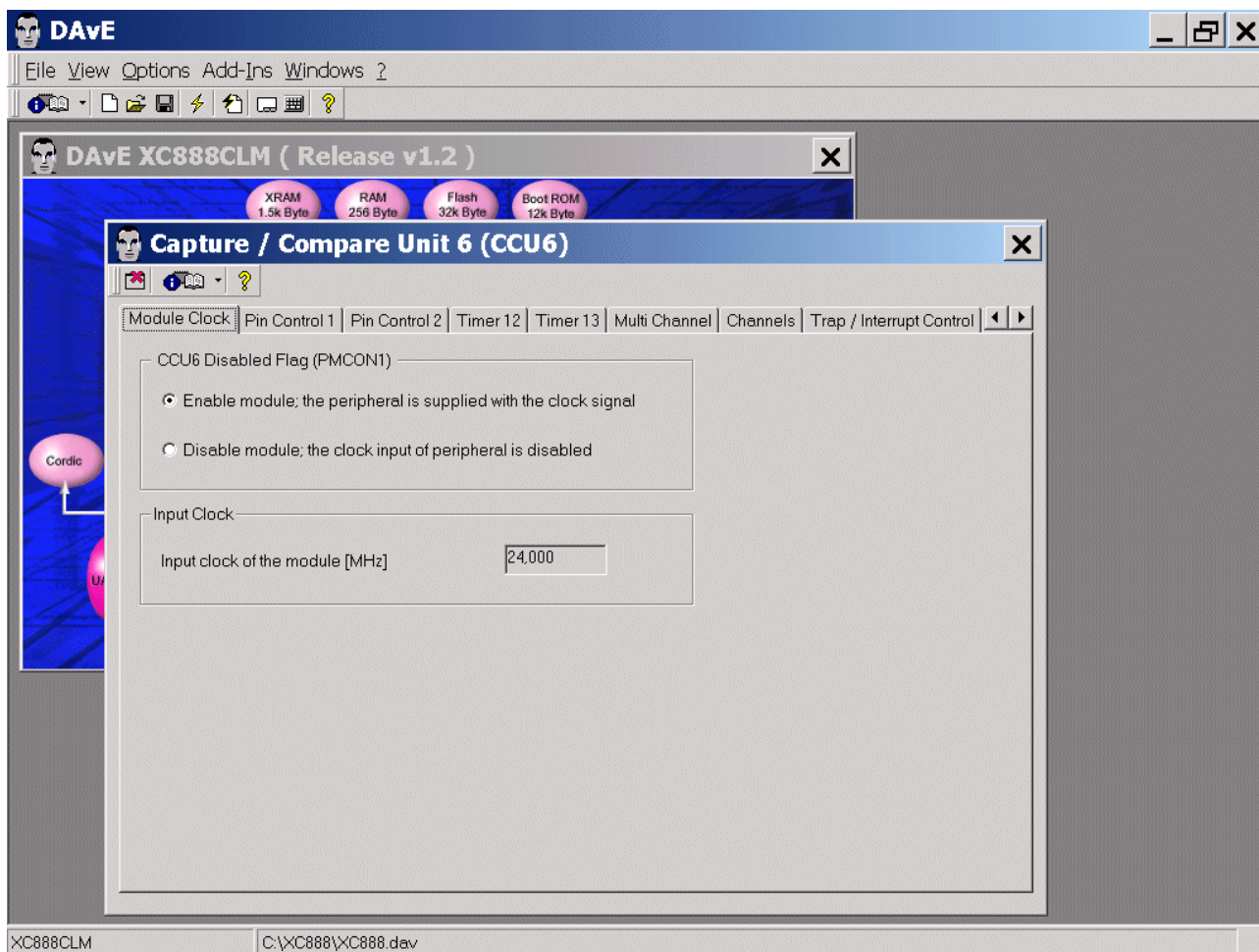
Exit this dialog now by clicking  the close button.

Configuration of the CAPCOM 6 module:

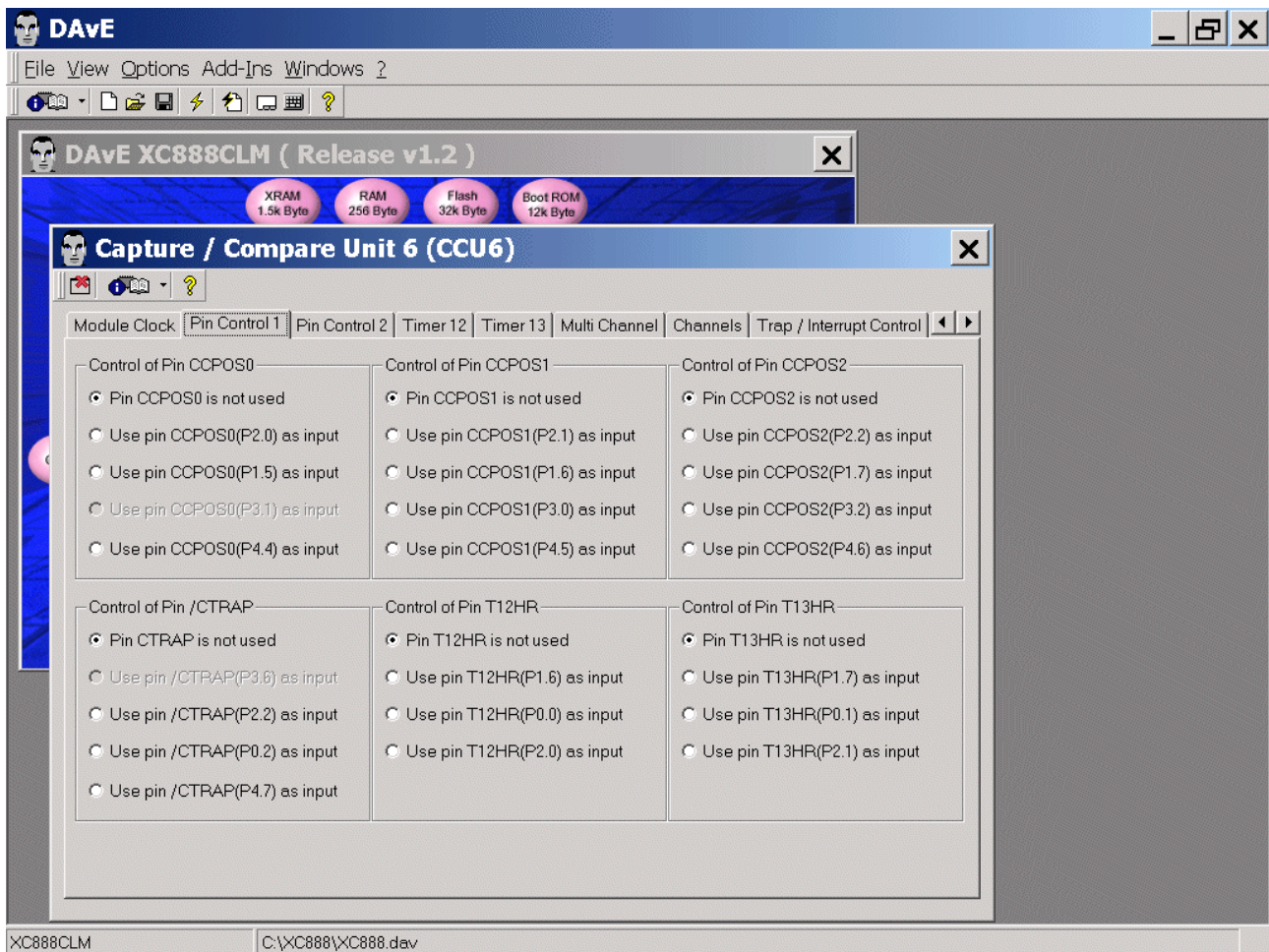
The configuration dialog can be opened by clicking the specific block/module.



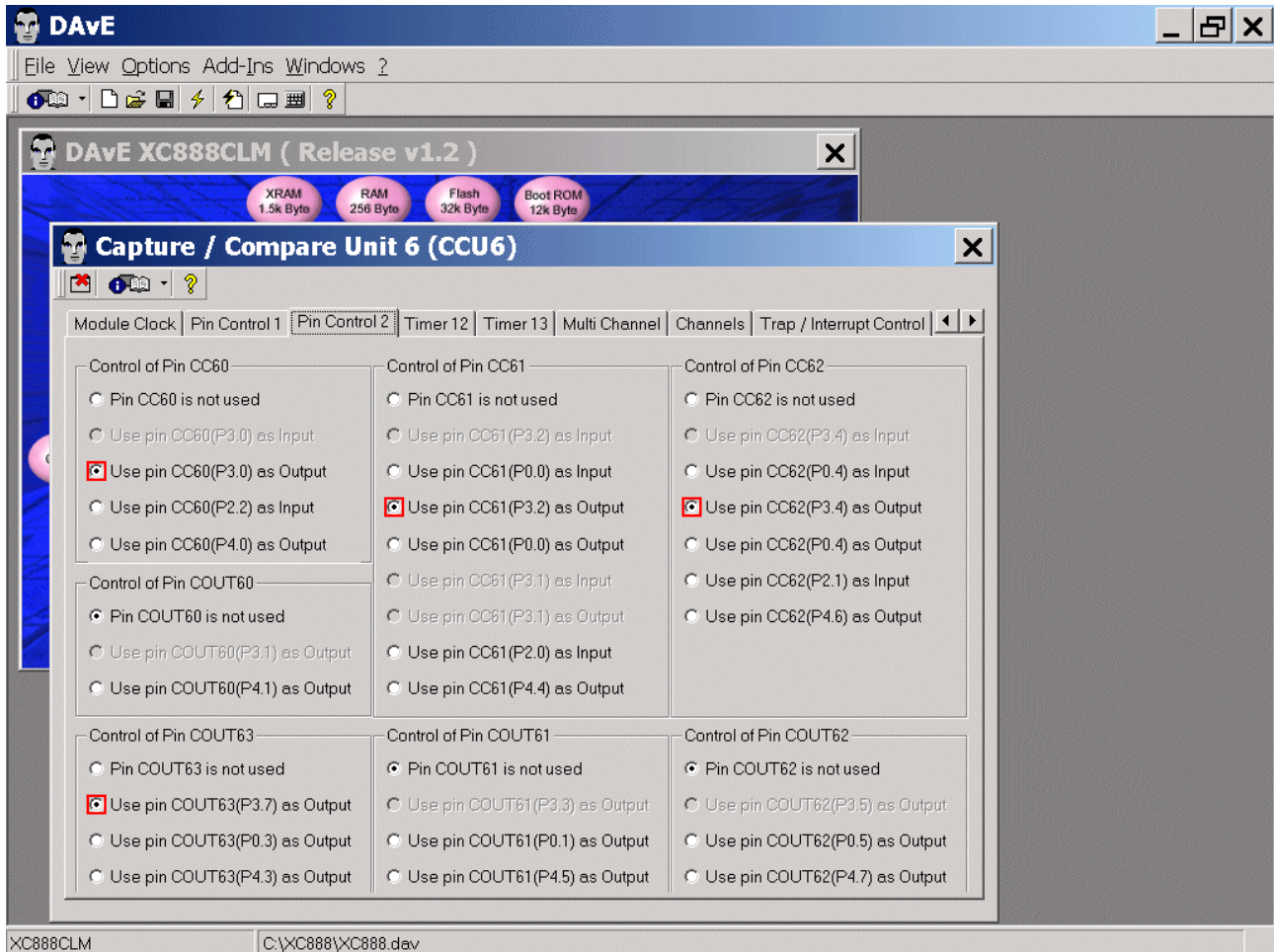
CCU6: Module Clock: CCU6 Disable Flag: **click** ☒ Enable module



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



Note:

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

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75
P3.7	COUT63_0	not used



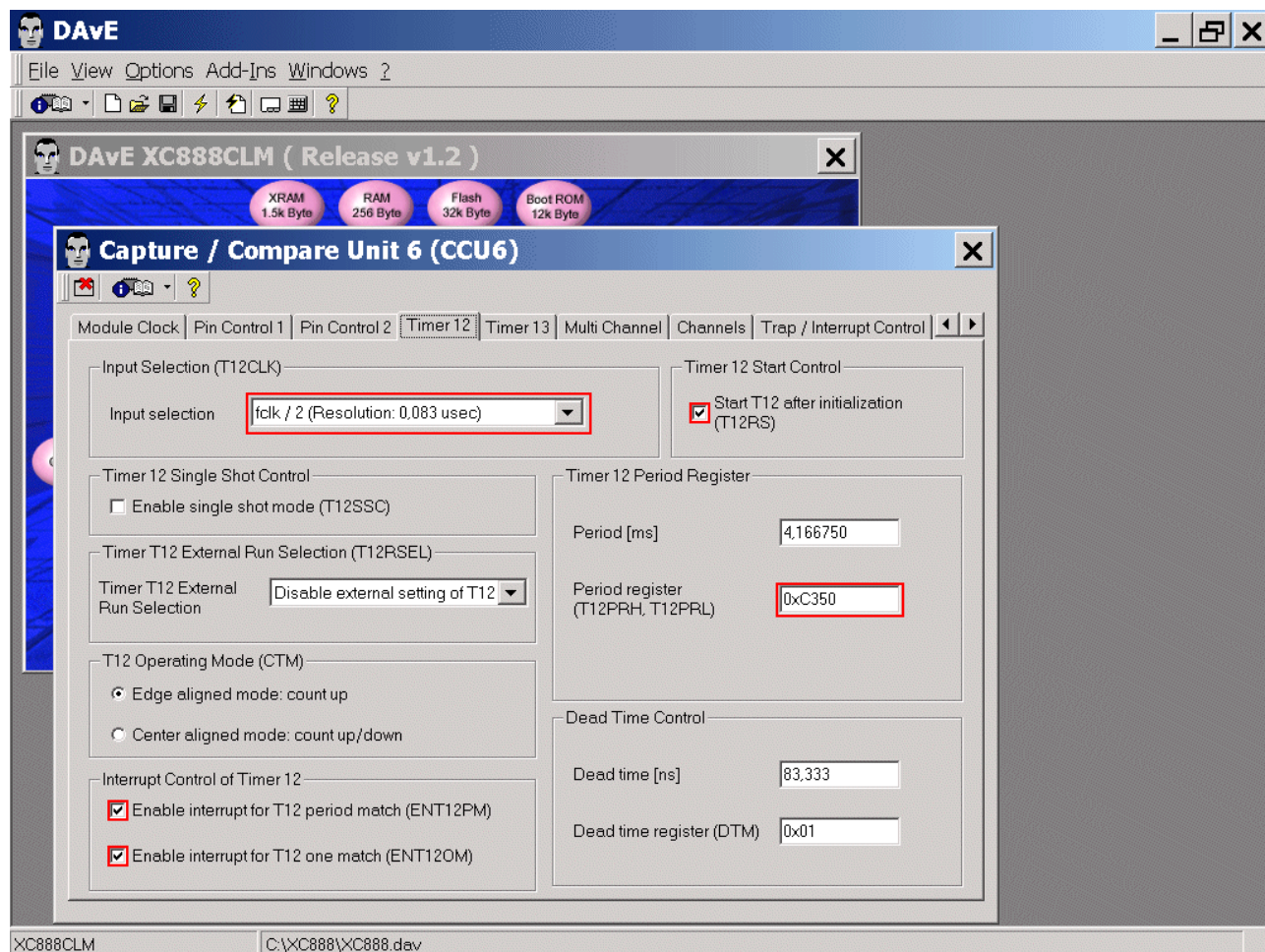
CCU6: T12: Input Selection (T12CLK): Input selection: select fclk/2 (Resolution: 0,083 μ sec)

CCU6: T12: Timer 12 start Control: click ☒ Start T12 after initialization

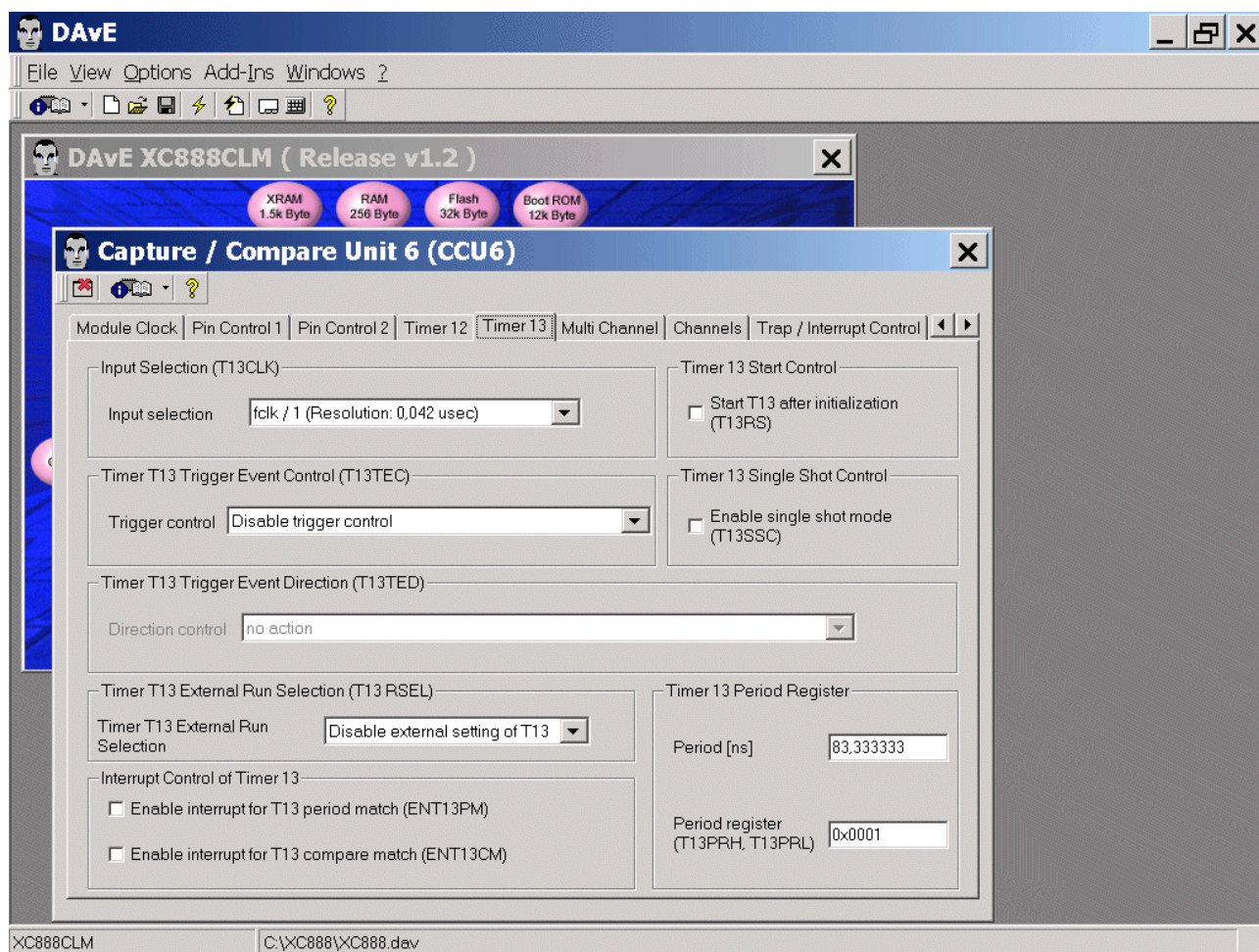
CCU6: T12: Timer 12 Period Register: Period register insert 50000 <ENTER>

CCU6: T12: Interrupt Control of Timer 12: click ☒ Enable interrupt for T12 period match

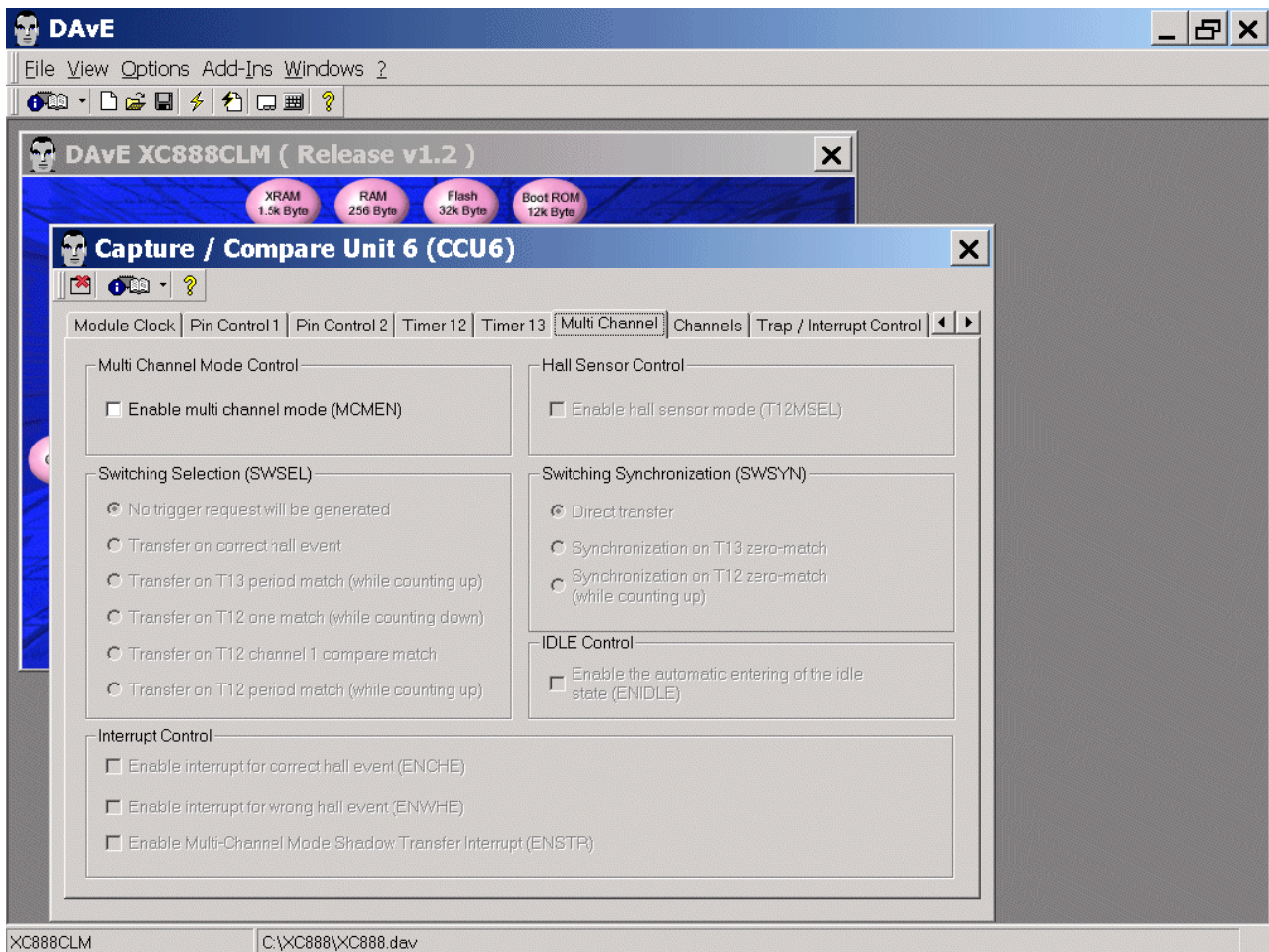
CCU6: T12: Interrupt Control of Timer 12: click ☒ Enable interrupt for T12 one match



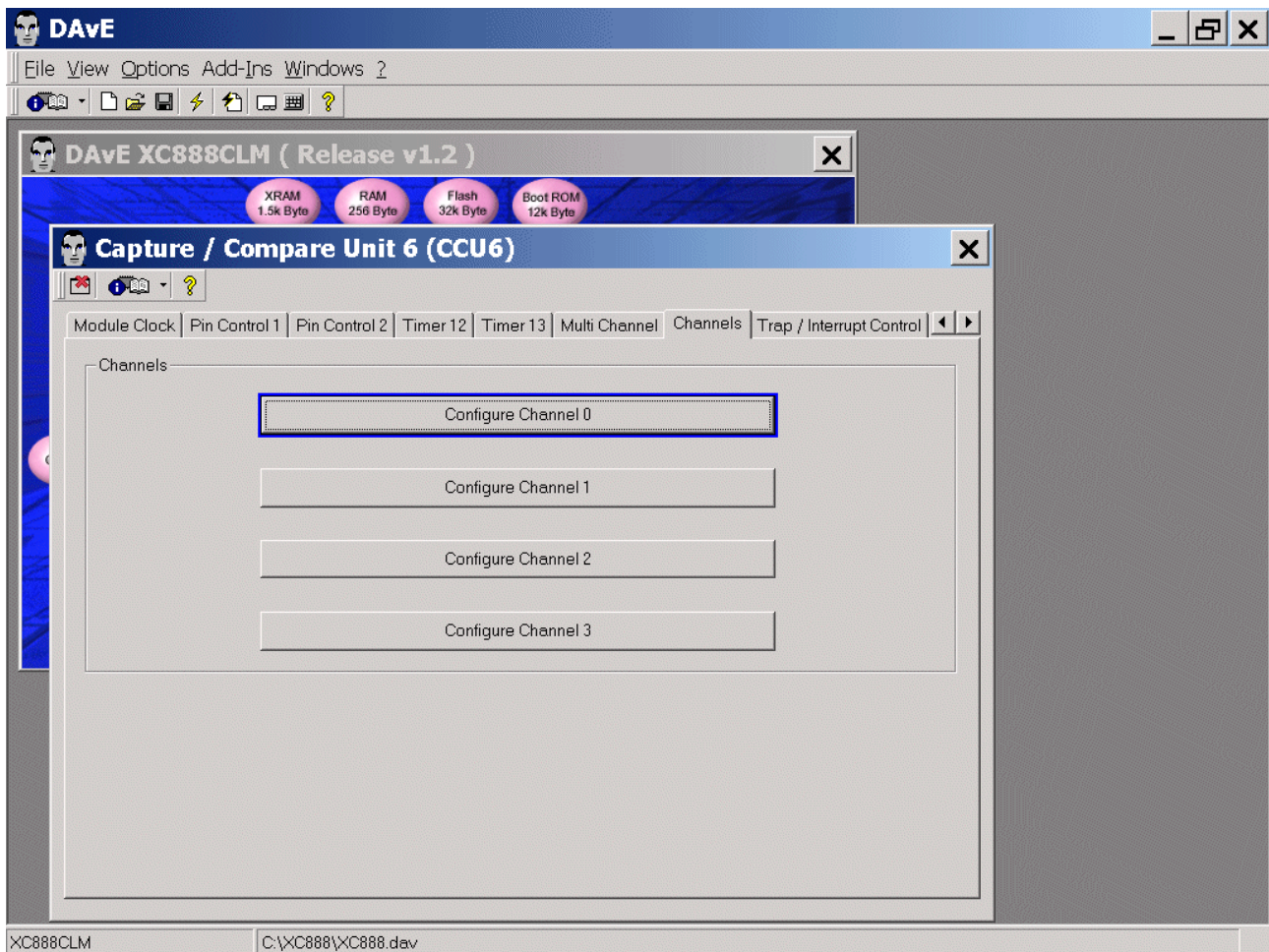
CCU6: Timer 13: (do nothing)



CCU6: Multi Channel: (do nothing)



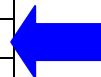
CCU6: Channels: **click** Configure Channel 0



Note:

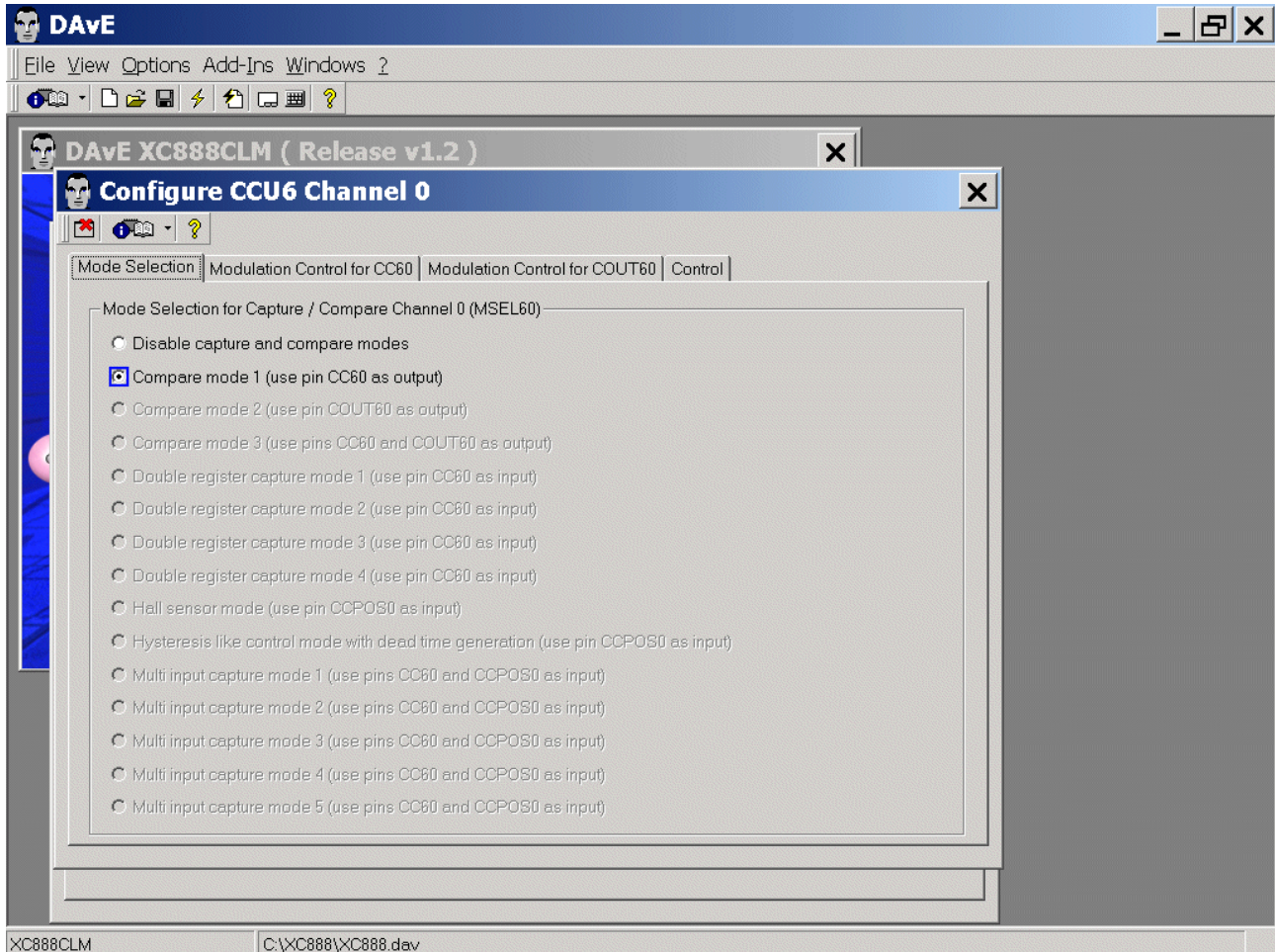
Port pins used by our PWM module (not available as GPIO pins):

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	25
P3.2	CC61_0	Channel 1	50
P3.4	CC62_0	Channel 2	75
P3.7	COU63_0	Channel 3	not used

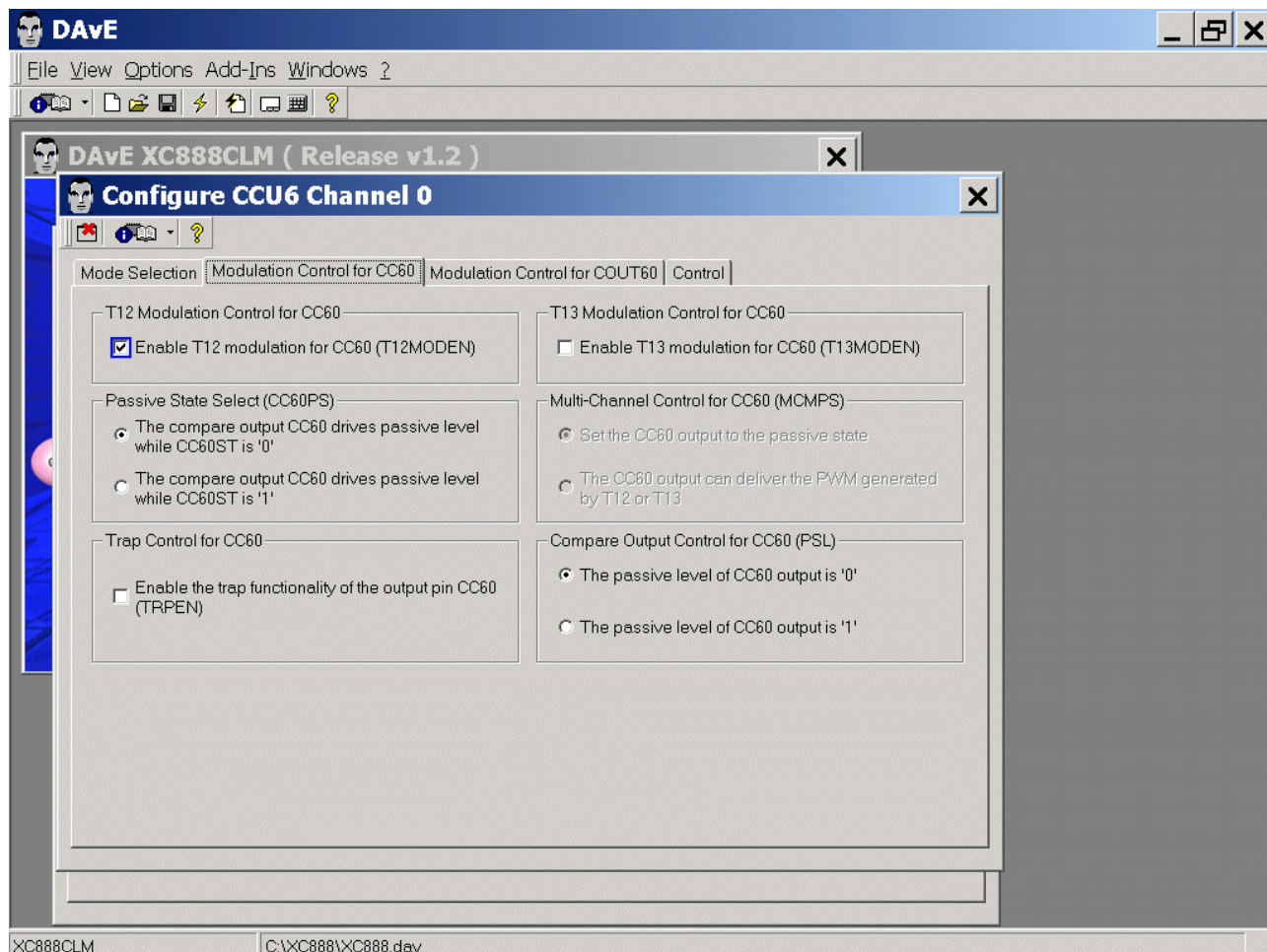


CCU6: Channels: Configure Channel 0:

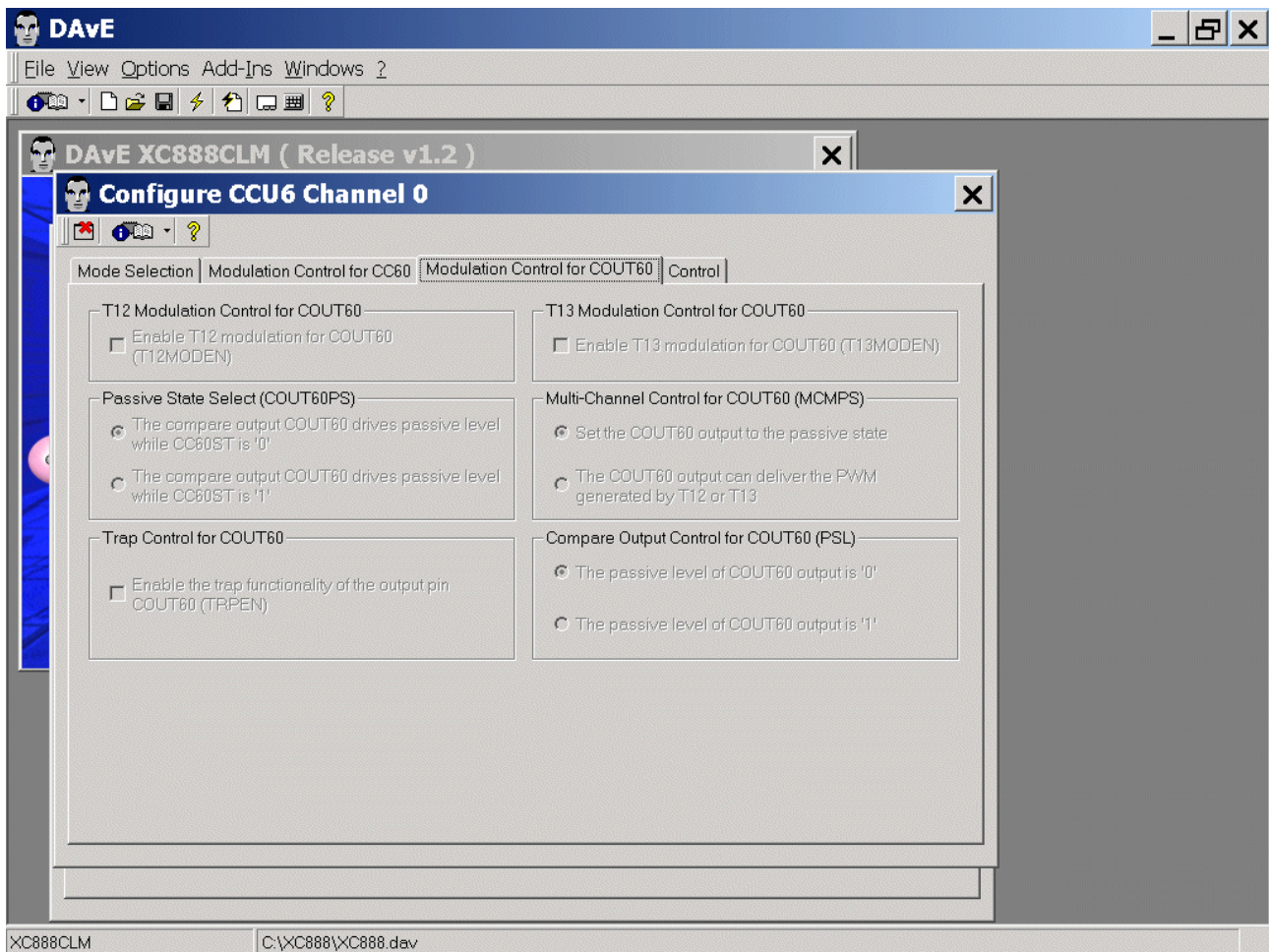
Mode Selection: Mode Selection for Capture / Compare Channel 0: click ☒ Compare mode 1



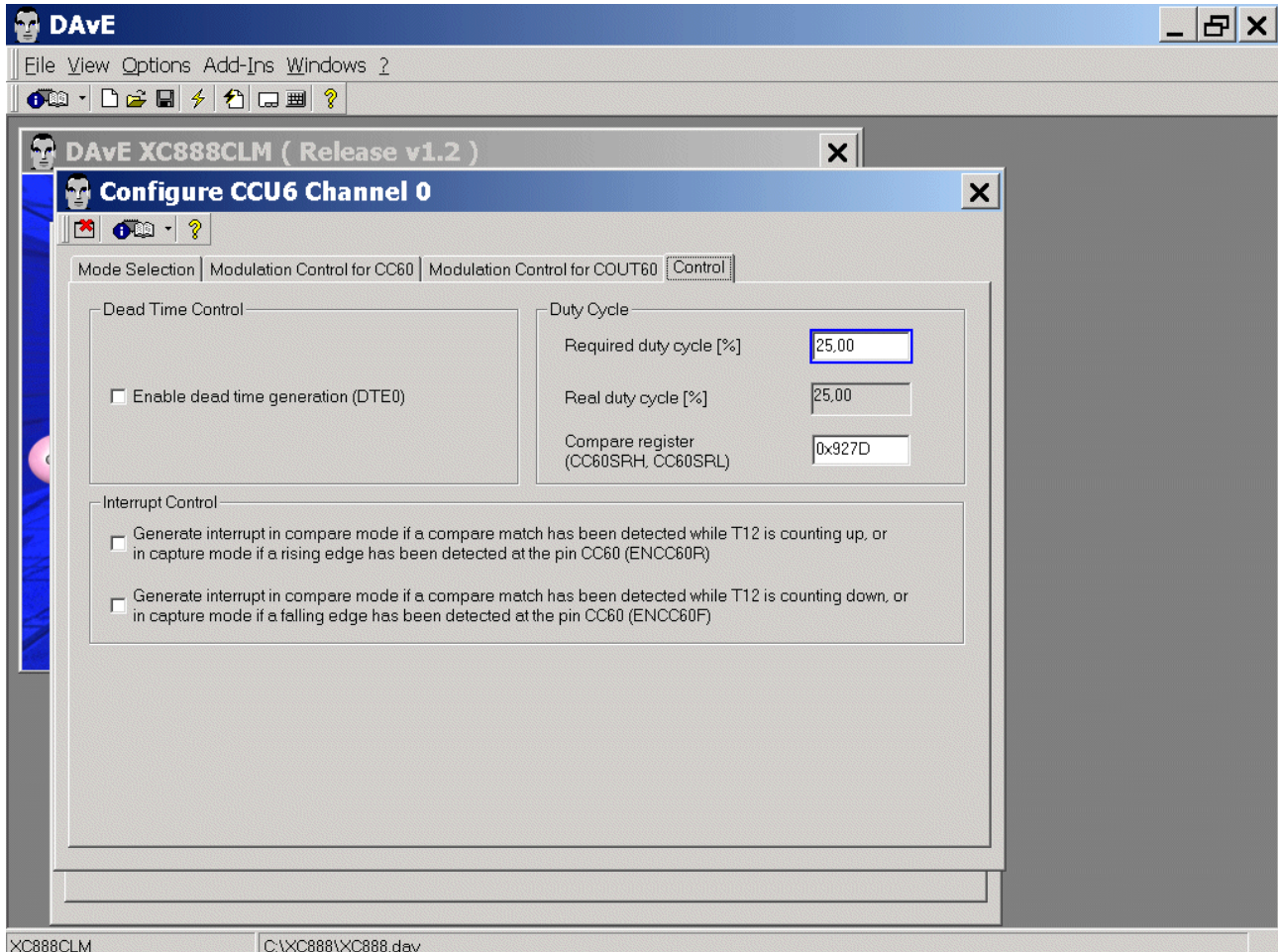
CCU6: Channels: Configure Channel 0: Modulation Control for CC60:
T12 Modulation Control for CC60: click ☒ Enable T12 modulation for CC60



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

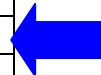


CCU6: Channels: Configure Channel 0: Control:
Duty Cycle: Required duty cycle [%] insert 25 <ENTER>



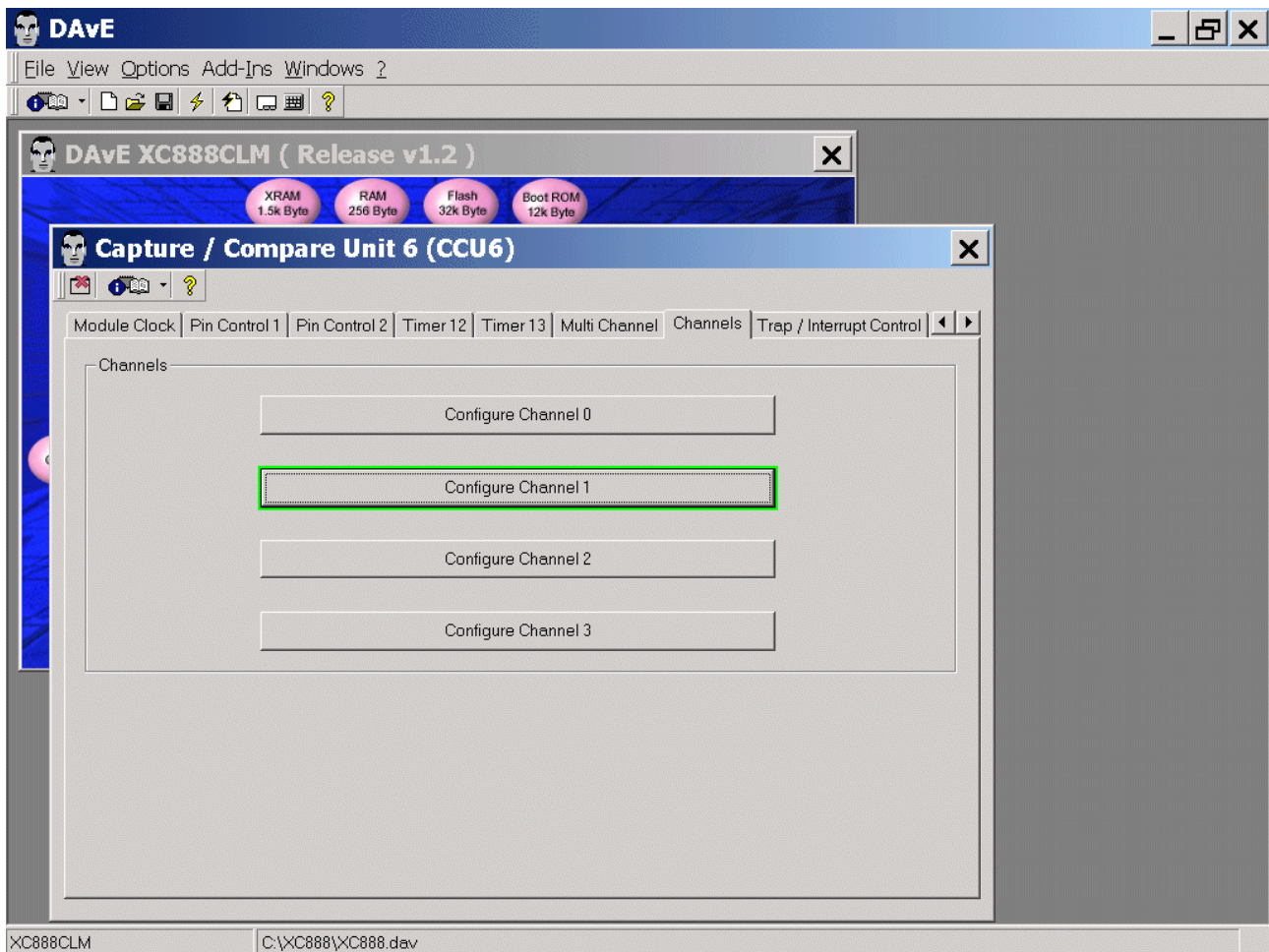
Note:
Channel 0 Duty Cycle = 25%:

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	25
P3.2	CC61_0	Channel 1	50
P3.4	CC62_0	Channel 2	75
P3.7	COU63_0	Channel 3	not used



Exit this dialog now by clicking  the close button.

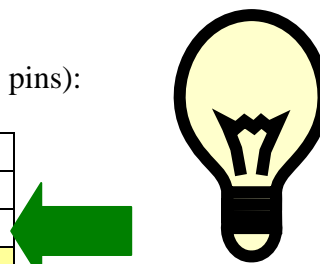
CCU6: Channels: **click** Configure Channel 1



Note:

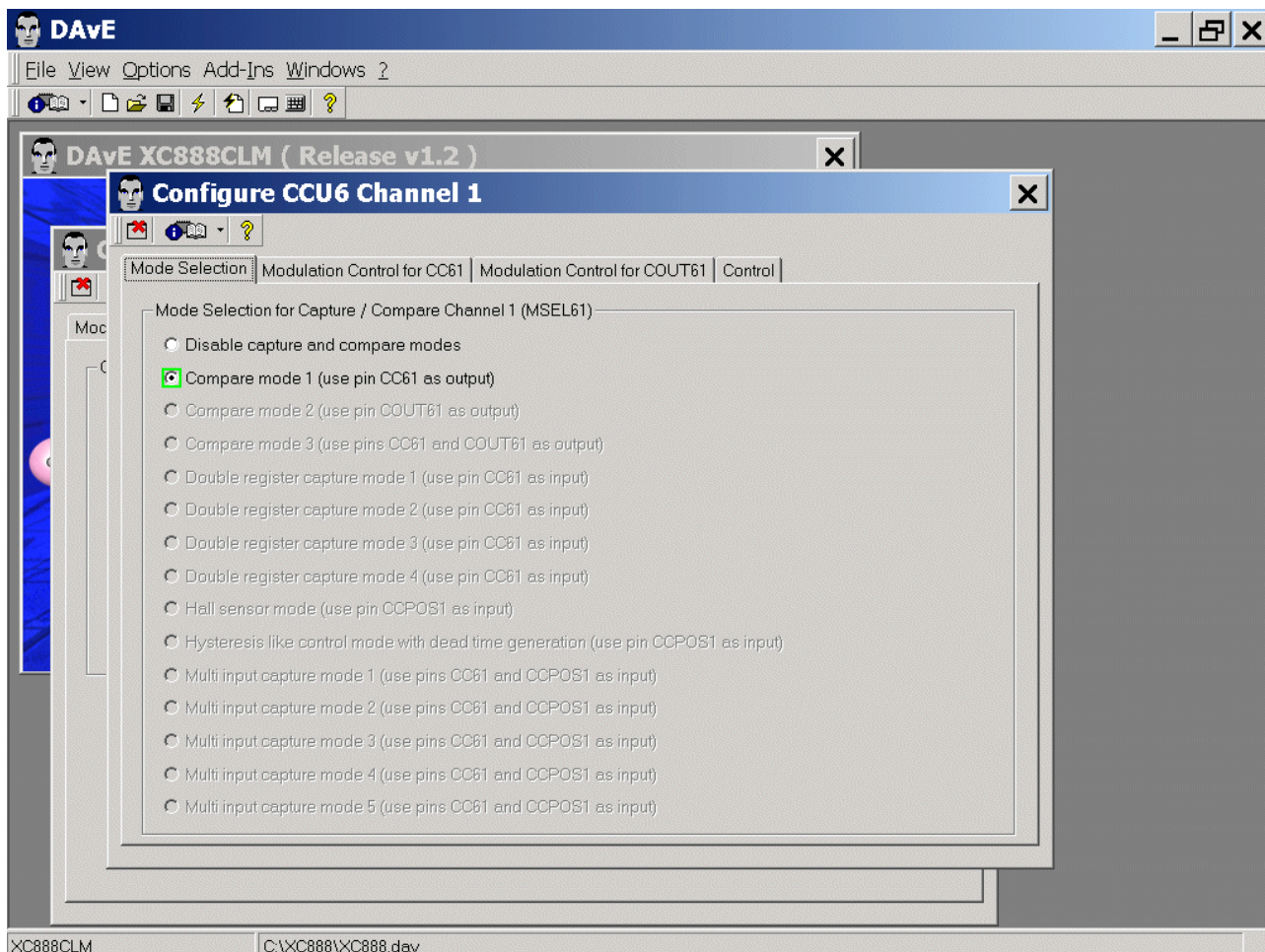
Port pins used by our PWM module (not available as GPIO pins):

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	25
P3.2	CC61_0	Channel 1	50
P3.4	CC62_0	Channel 2	75
P3.7	COU63_0	Channel 3	not used

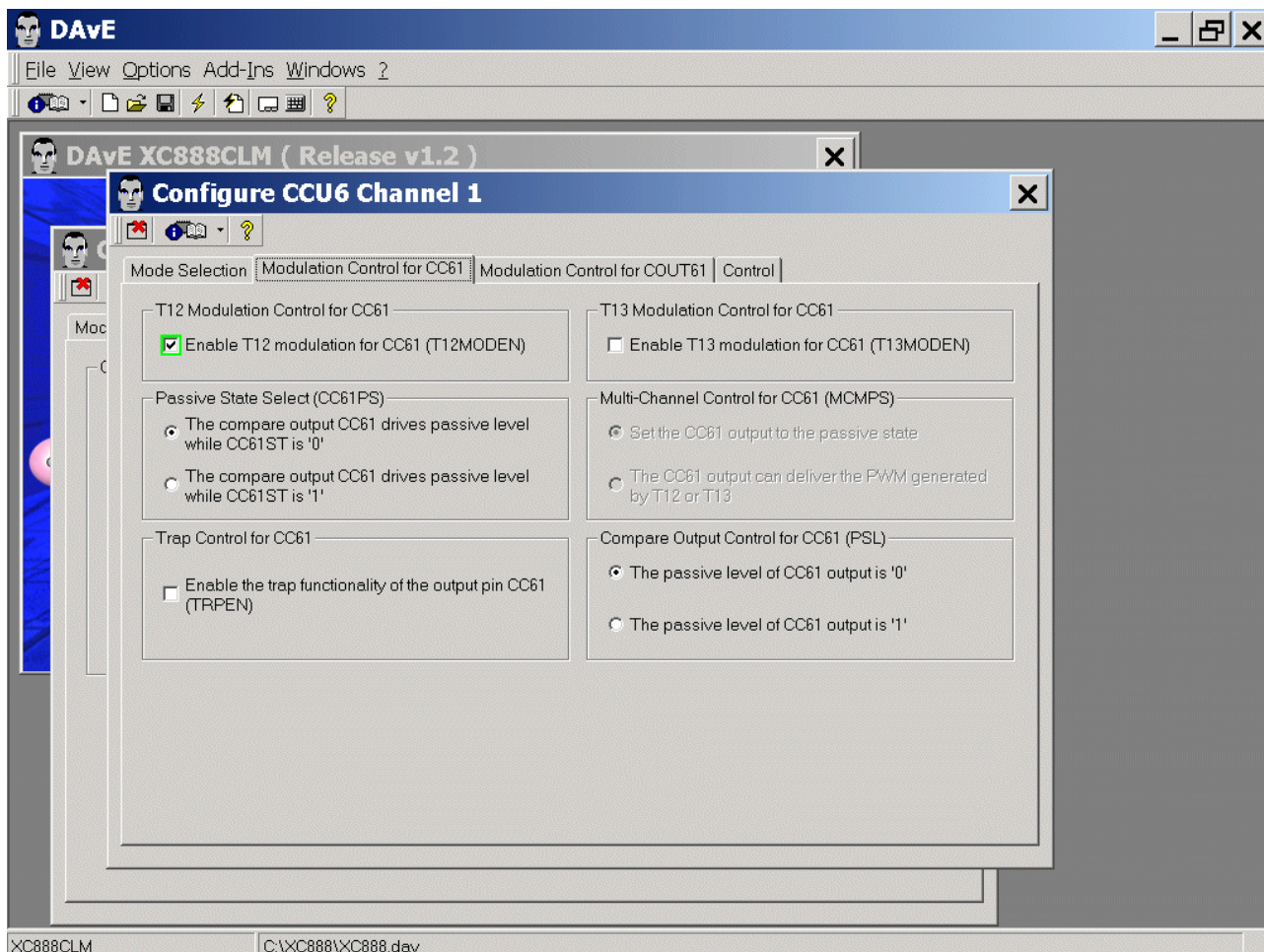


CCU6: Channels: **Configure Channel 1:**

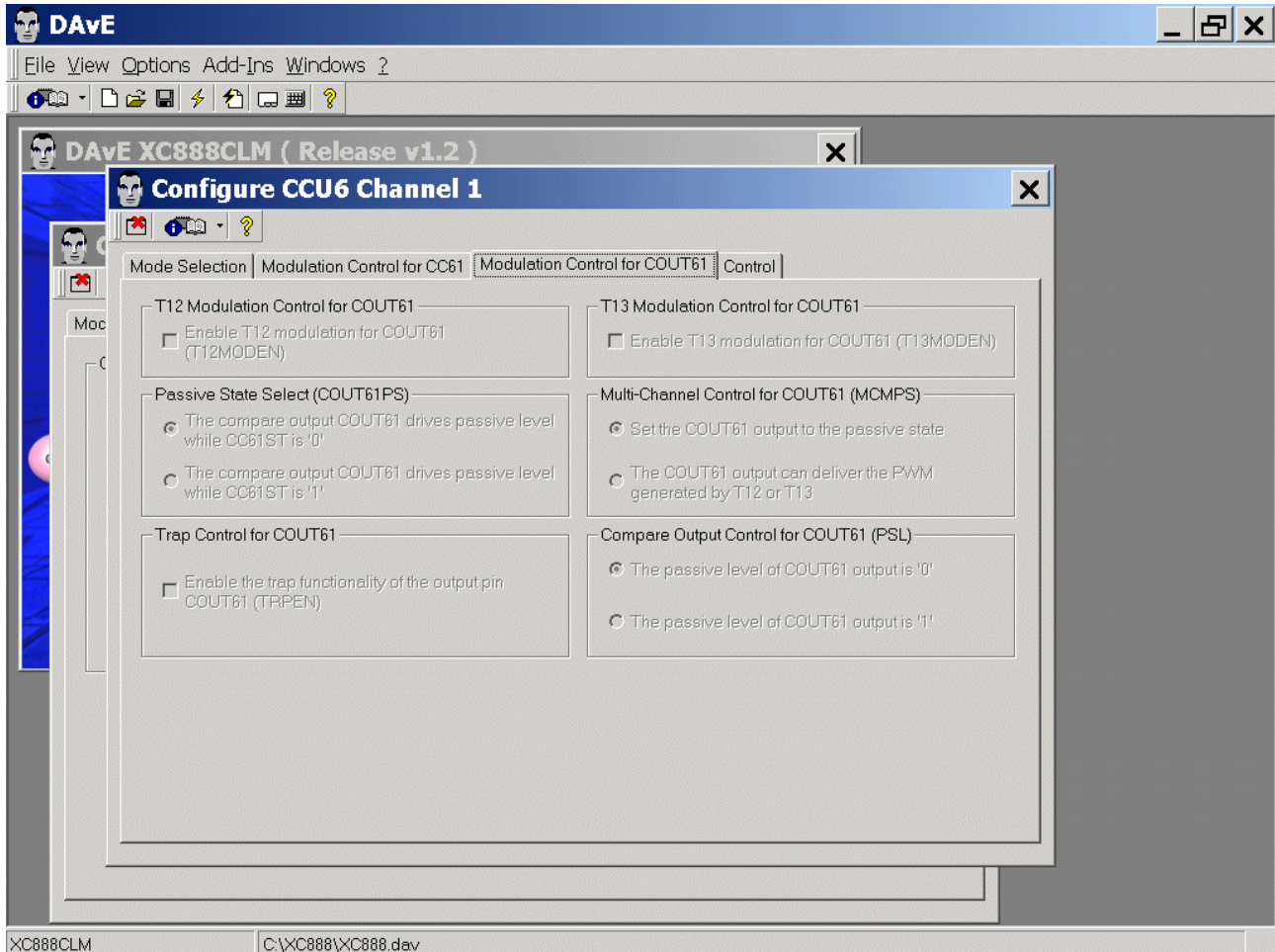
Mode Selection: **Mode Selection for Capture / Compare Channel 1:** click ☒ Compare mode 1



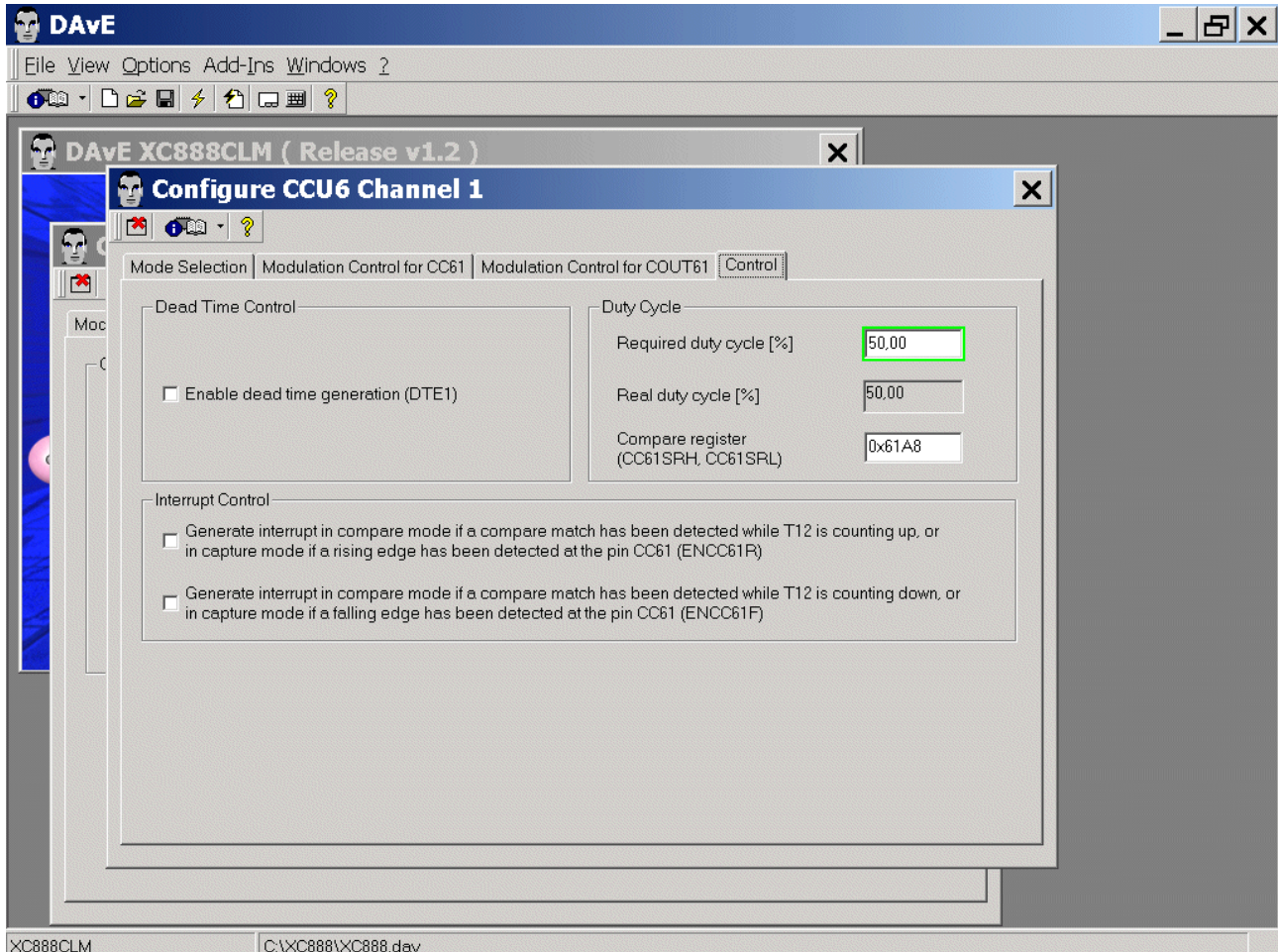
CCU6: Channels: Configure Channel 1: Modulation Control for CC61:
T12 Modulation Control for CC61: click ☒ Enable T12 modulation for CC61



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

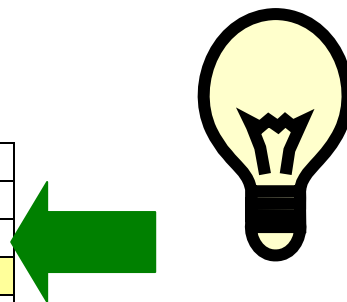


CCU6: Channels: Configure Channel 1: Control:
Duty Cycle: Required duty cycle [%] insert 50 <ENTER>



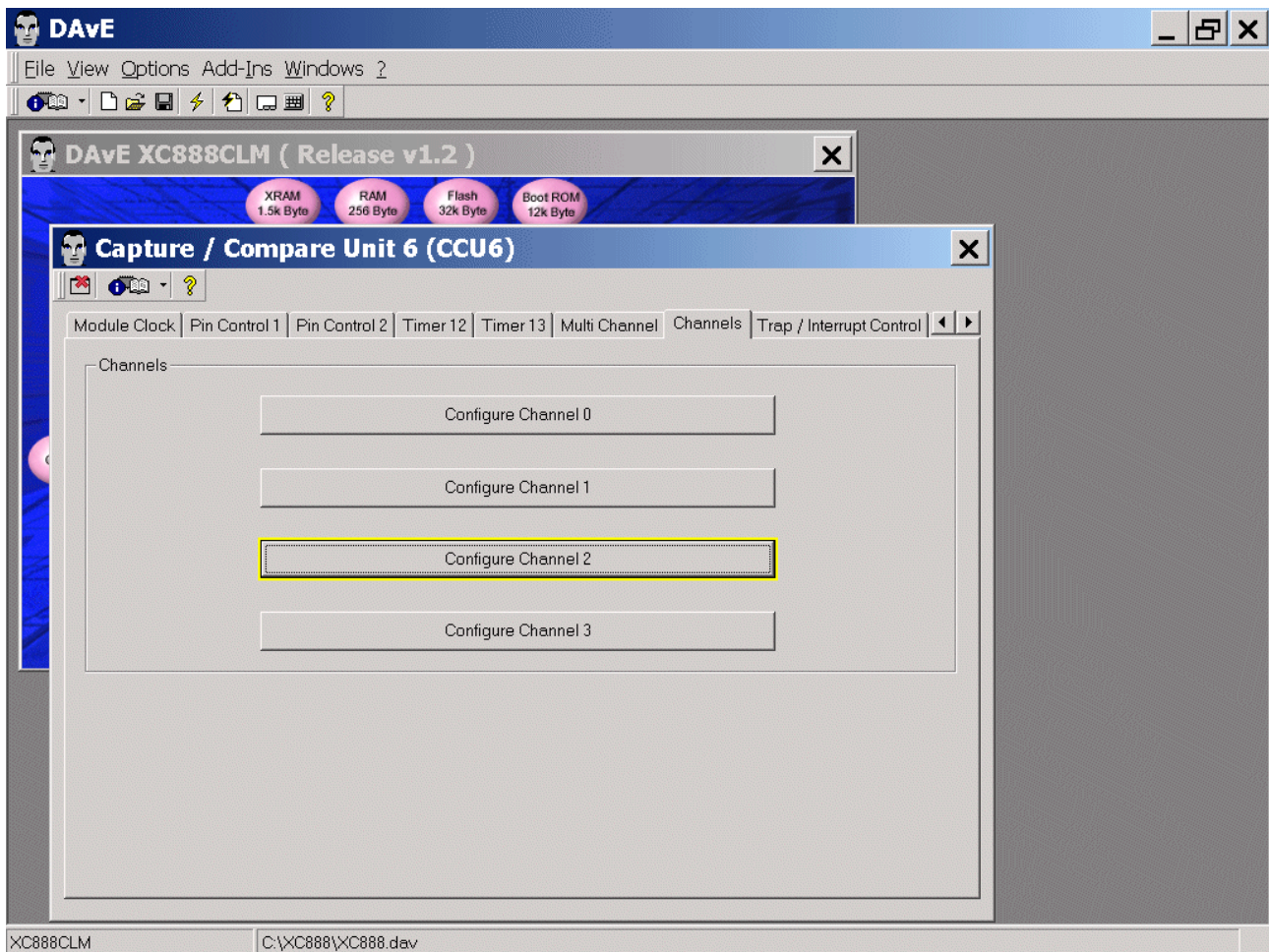
Note:
Channel 1 Duty Cycle = 50%:

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	25
P3.2	CC61_0	Channel 1	50
P3.4	CC62_0	Channel 2	75
P3.7	COU63_0	Channel 3	not used



Exit this dialog now by clicking  the close button.

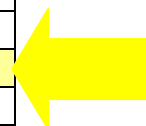
CCU6: Channels: **click** Configure Channel 2



Note:

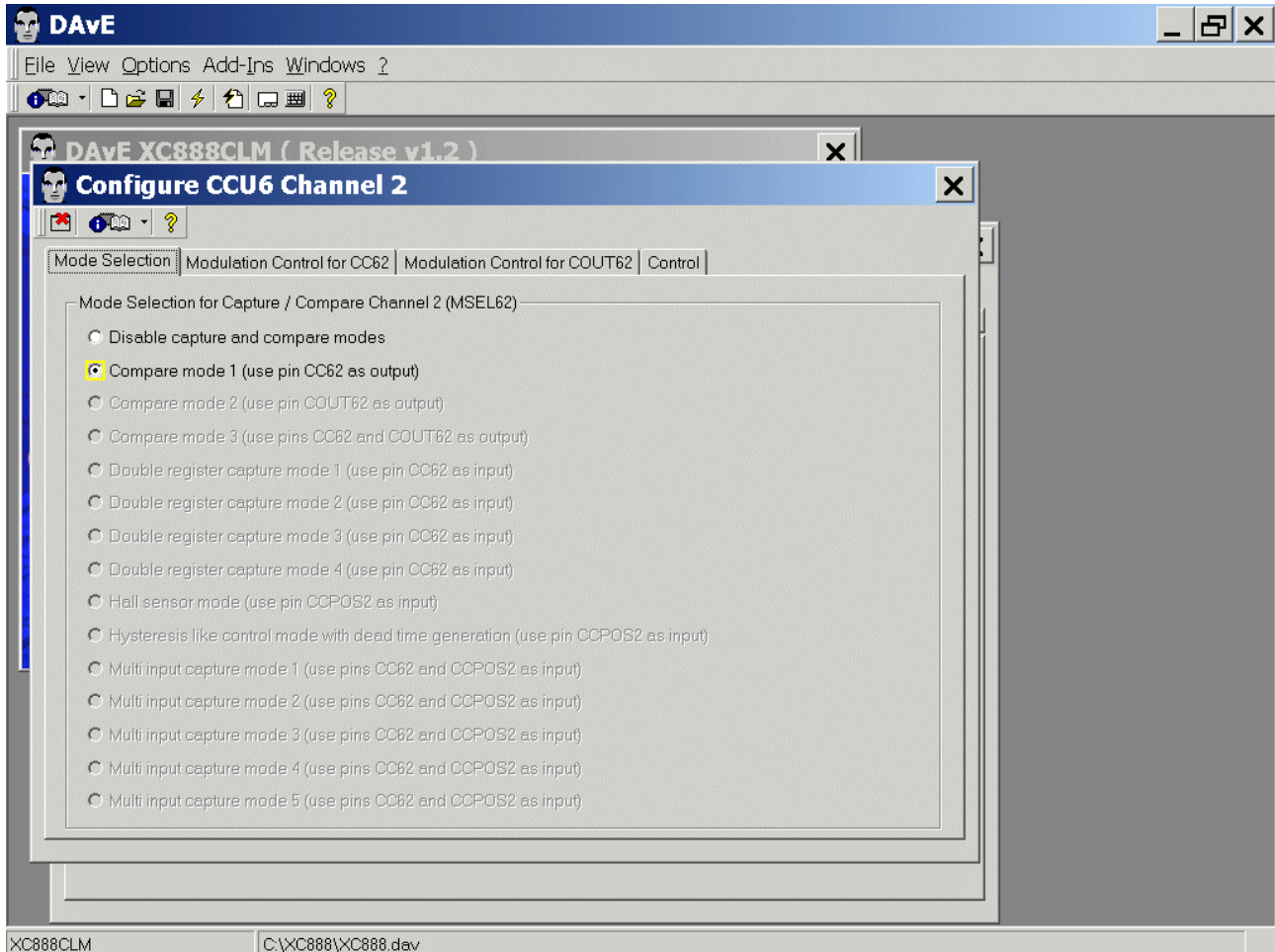
Port pins used by our PWM module (not available as GPIO pins):

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	25
P3.2	CC61_0	Channel 1	50
P3.4	CC62_0	Channel 2	75
P3.7	COU63_0	Channel 3	not used

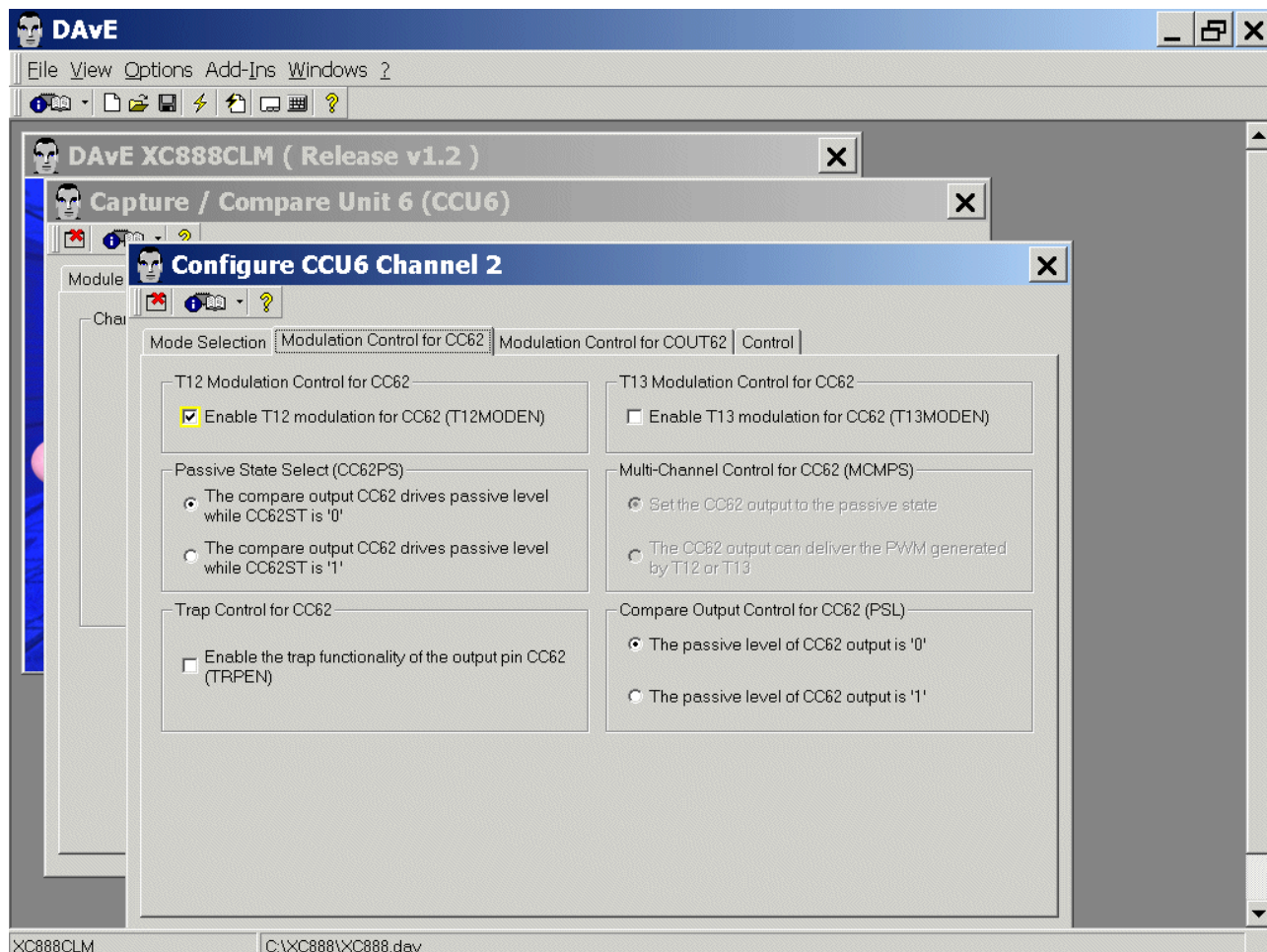


CCU6: Channels: Configure Channel 2:

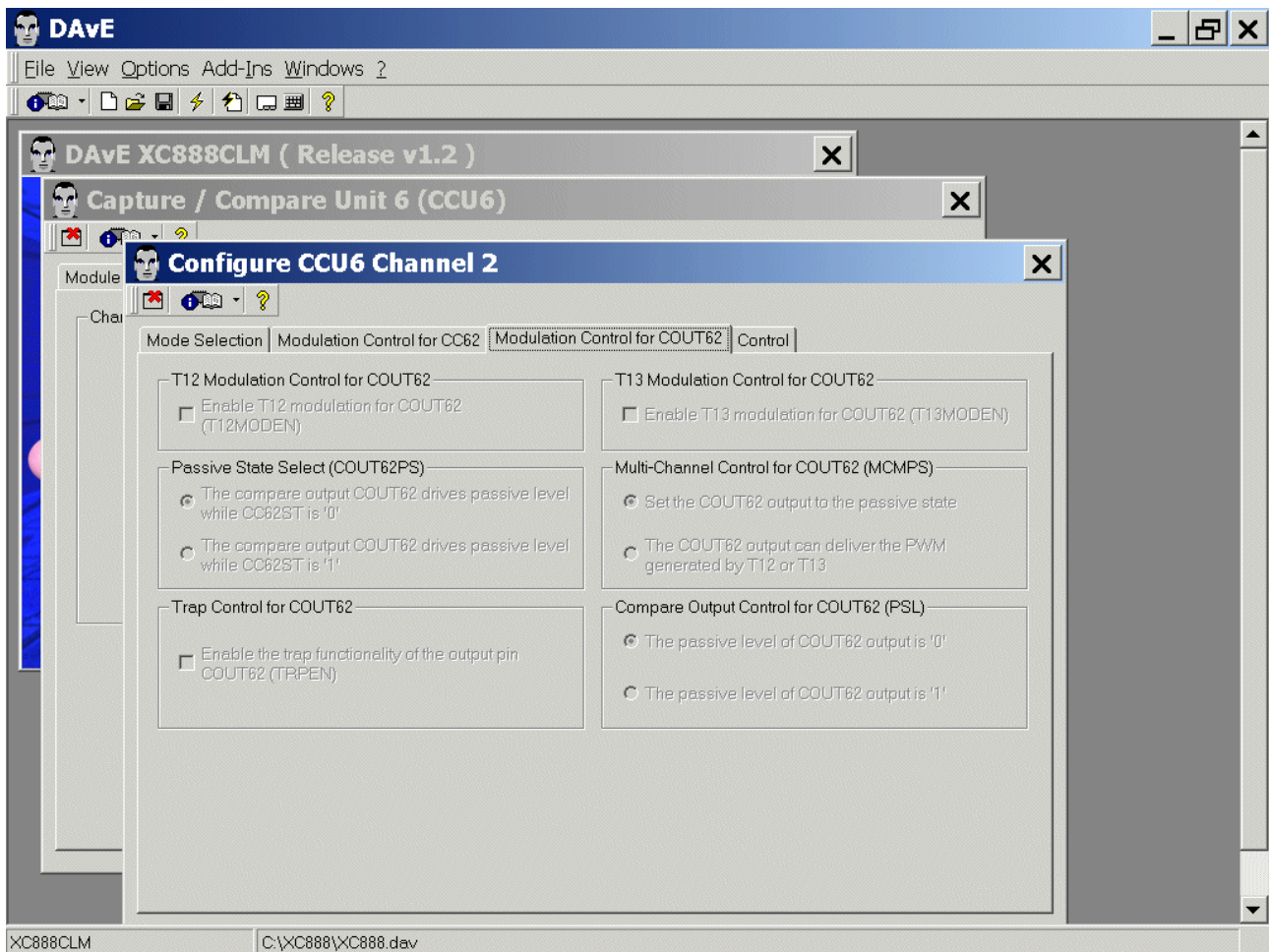
Mode Selection: Mode Selection for Capture / Compare Channel 2: click ☒ Compare mode 1



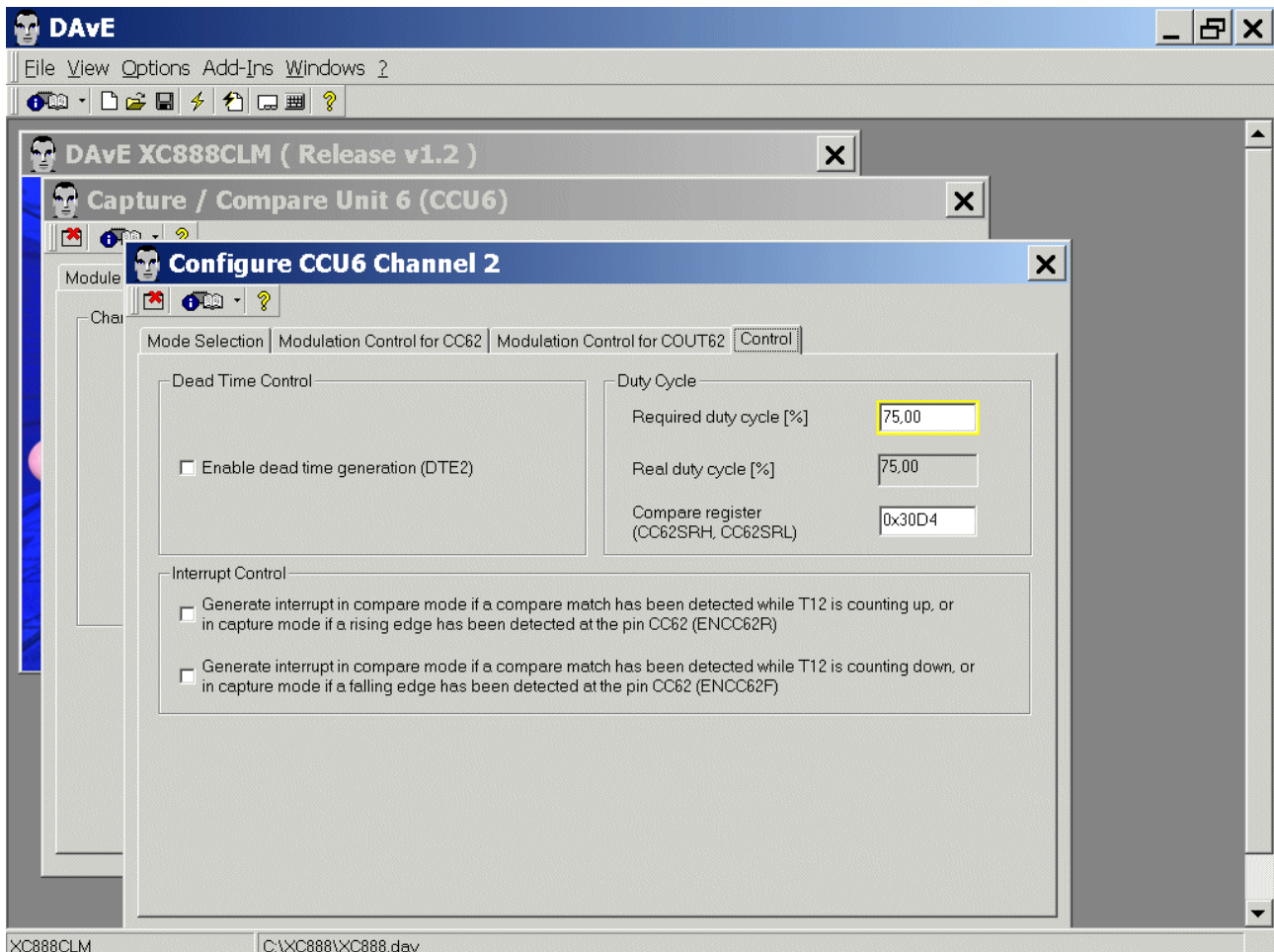
CCU6: Channels: Configure Channel 2: Modulation Control for CC62:
T12 Modulation Control for CC62: click Enable T12 modulation for CC62



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

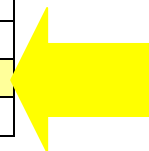


CCU6: Channels: Configure Channel 2: Control:
Duty Cycle: Required duty cycle [%] insert 75 <ENTER>



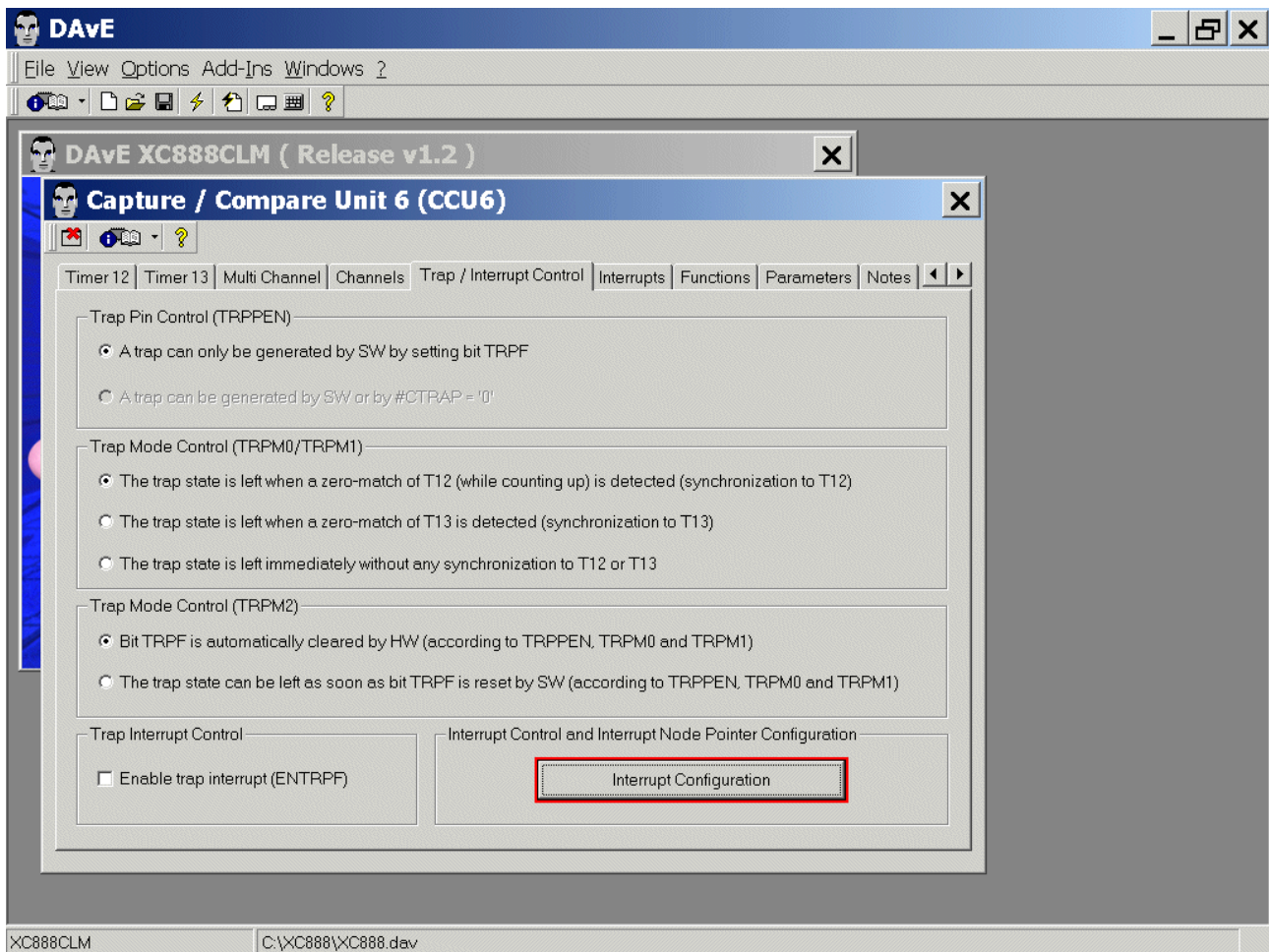
Note:
Channel 2 Duty Cycle = 75%:

Port Lines	Signal	Channel	Duty Cycle [%]
P3.0	CC60_0	Channel 0	25
P3.2	CC61_0	Channel 1	50
P3.4	CC62_0	Channel 2	75
P3.7	COU63_0	Channel 3	not used



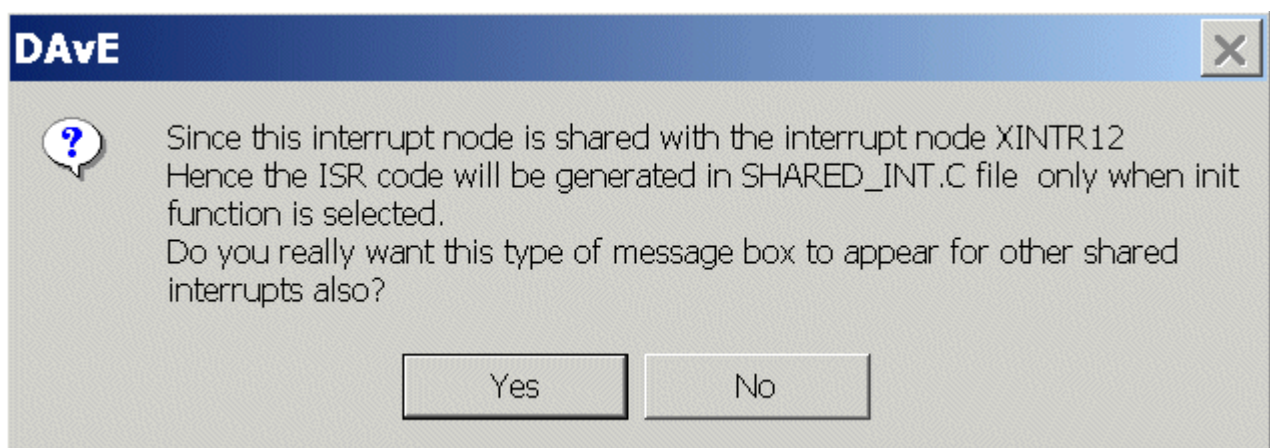
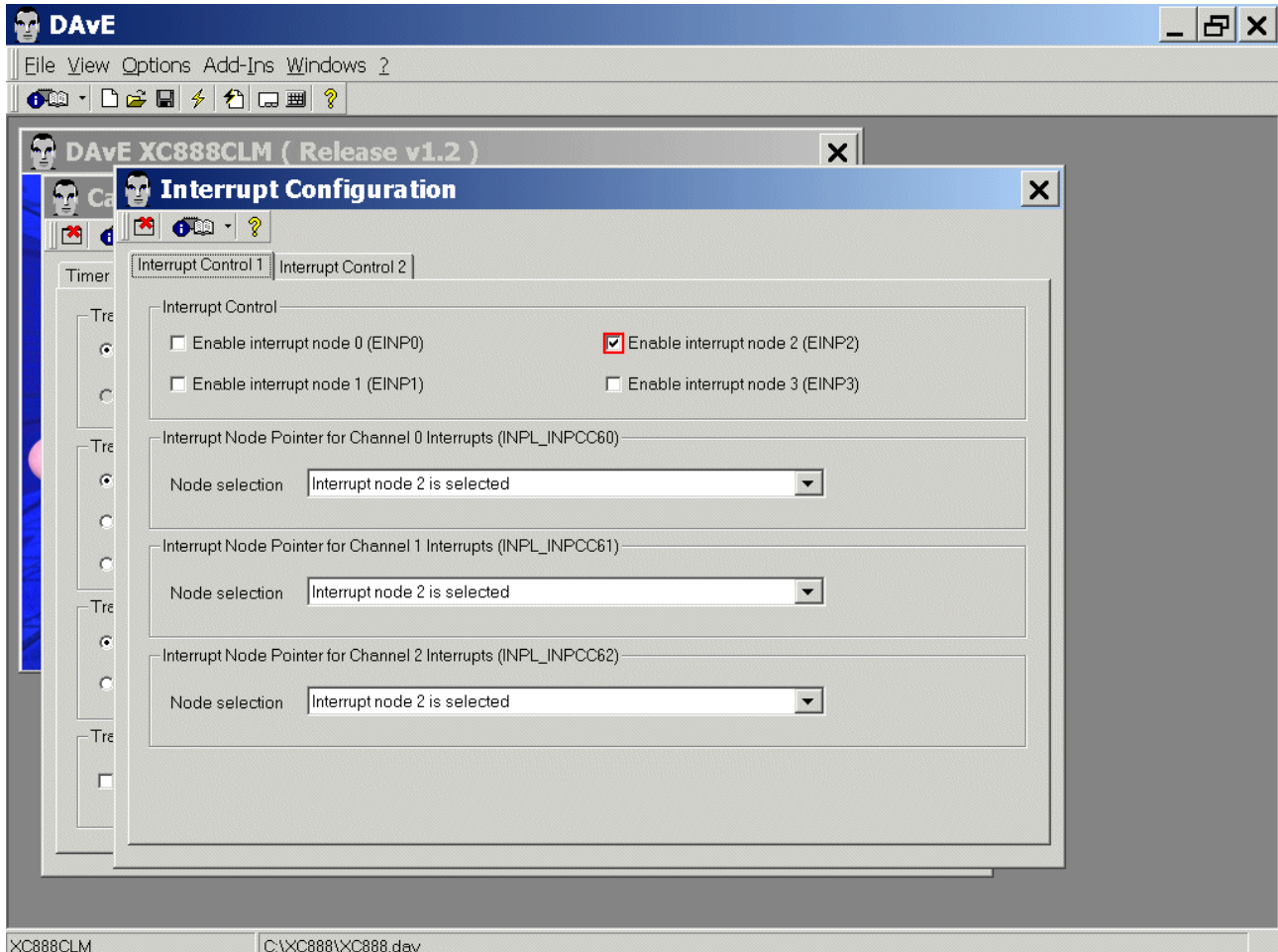
Exit this dialog now by clicking  the close button.

CCU6: **Trap / Interrupt Control**: **click** Interrupt Configuration



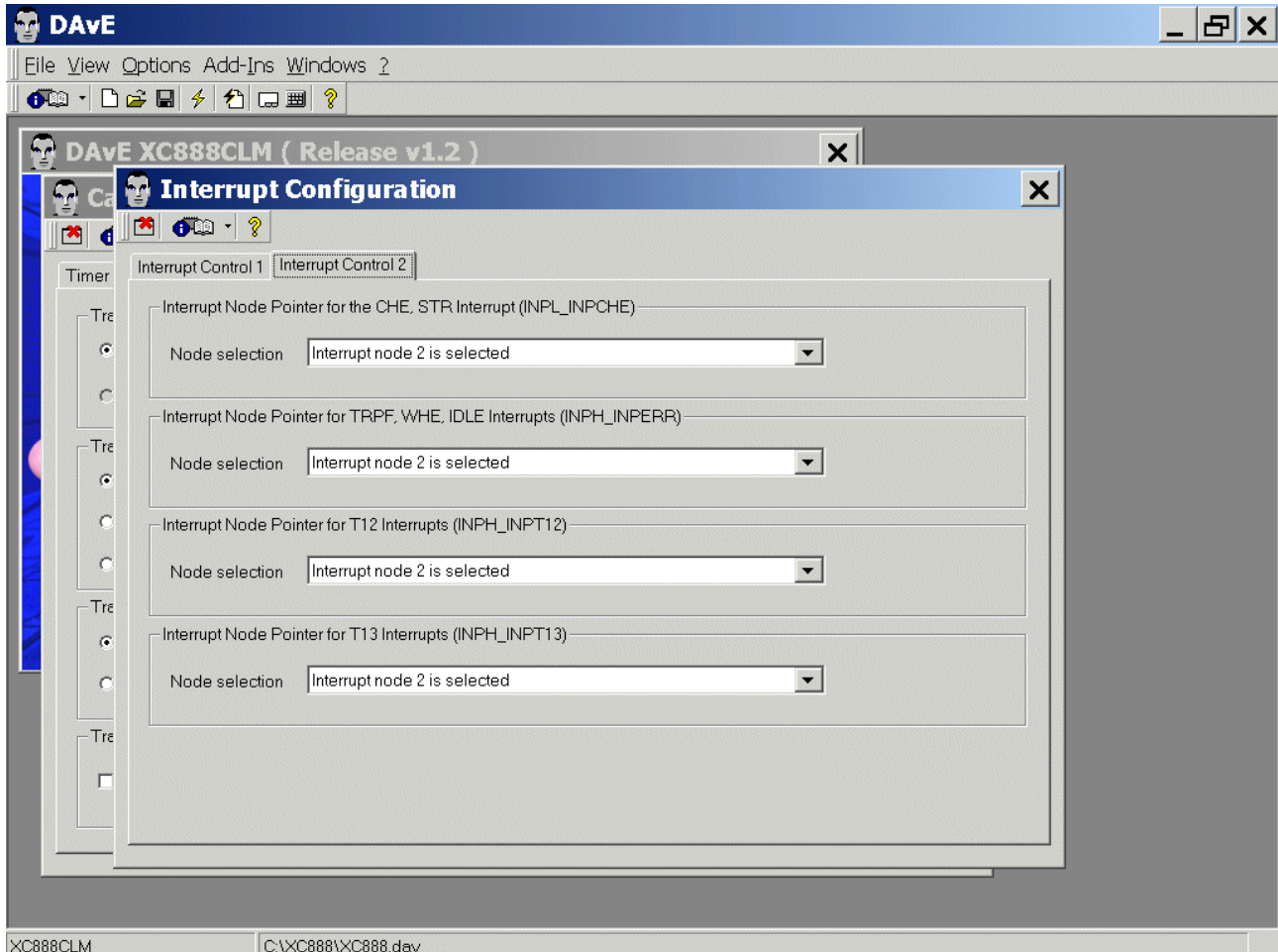
CCU6: Trap / Interrupt Control: Interrupt Configuration:

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



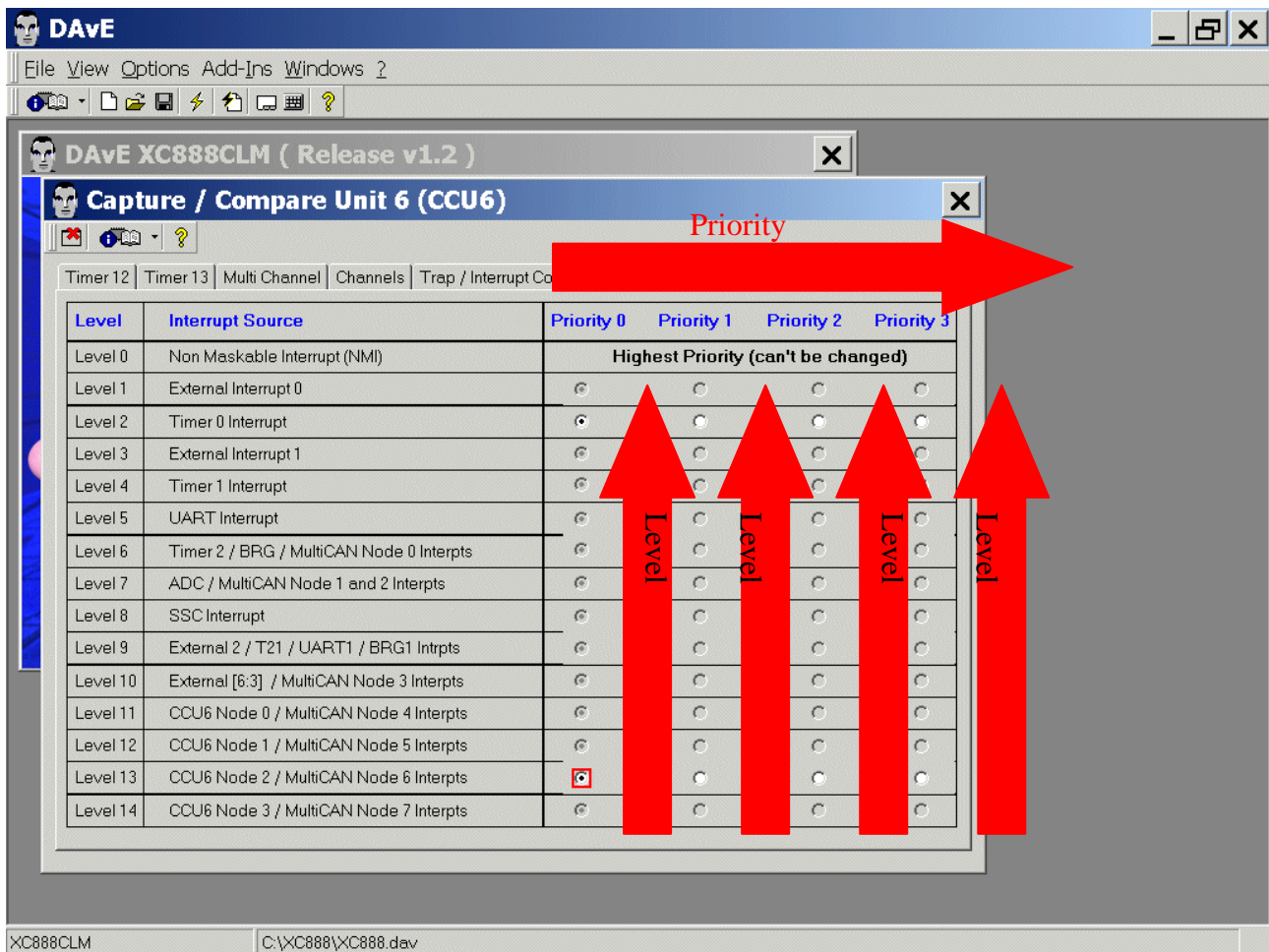
Click Yes

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

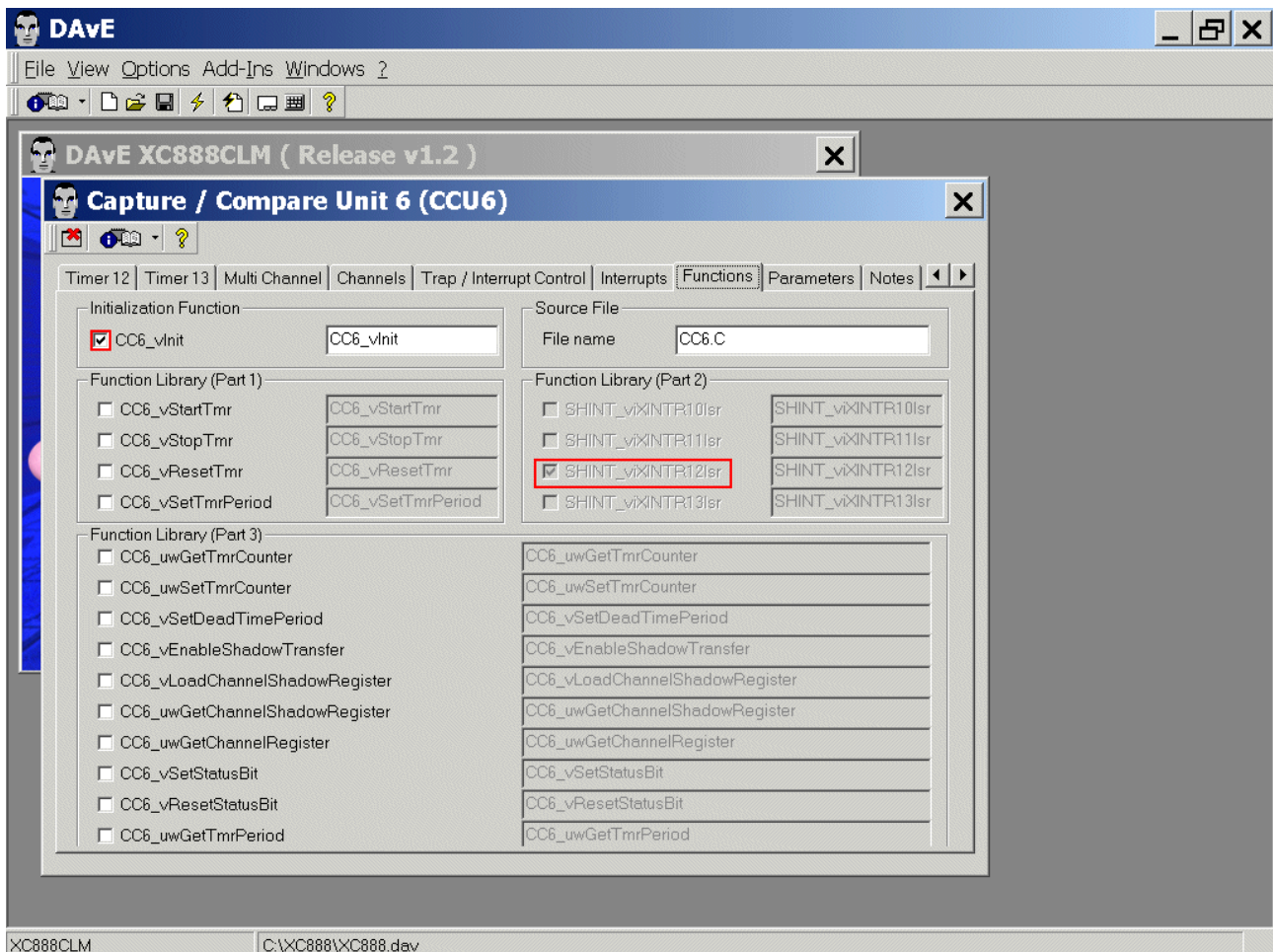


Exit this dialog now by clicking  the close button.

CCU6: Interrupts: (do nothing)



CCU6: Functions: Initialization Function: click ☒ CCU6_vInit



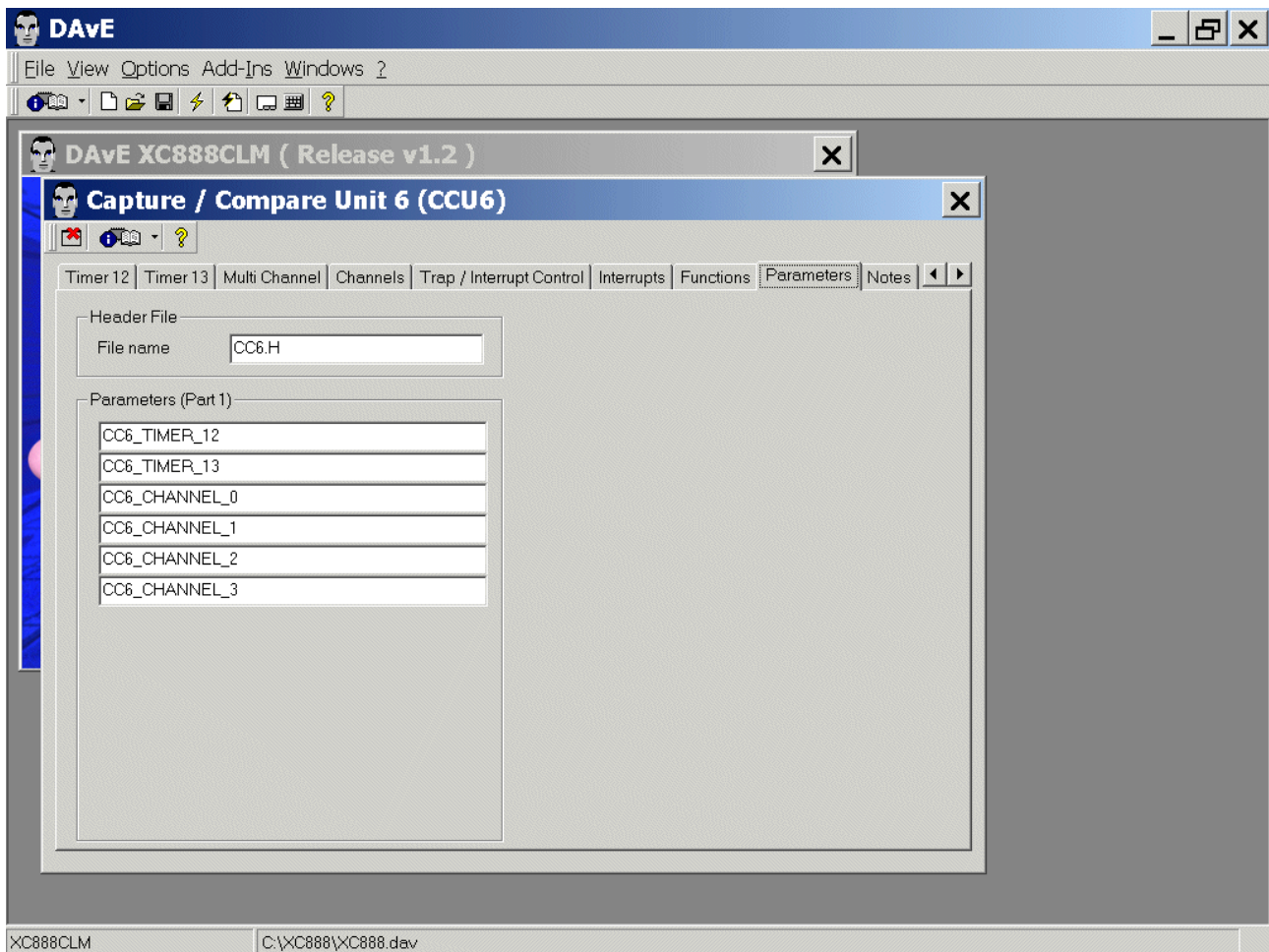
Note:

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.

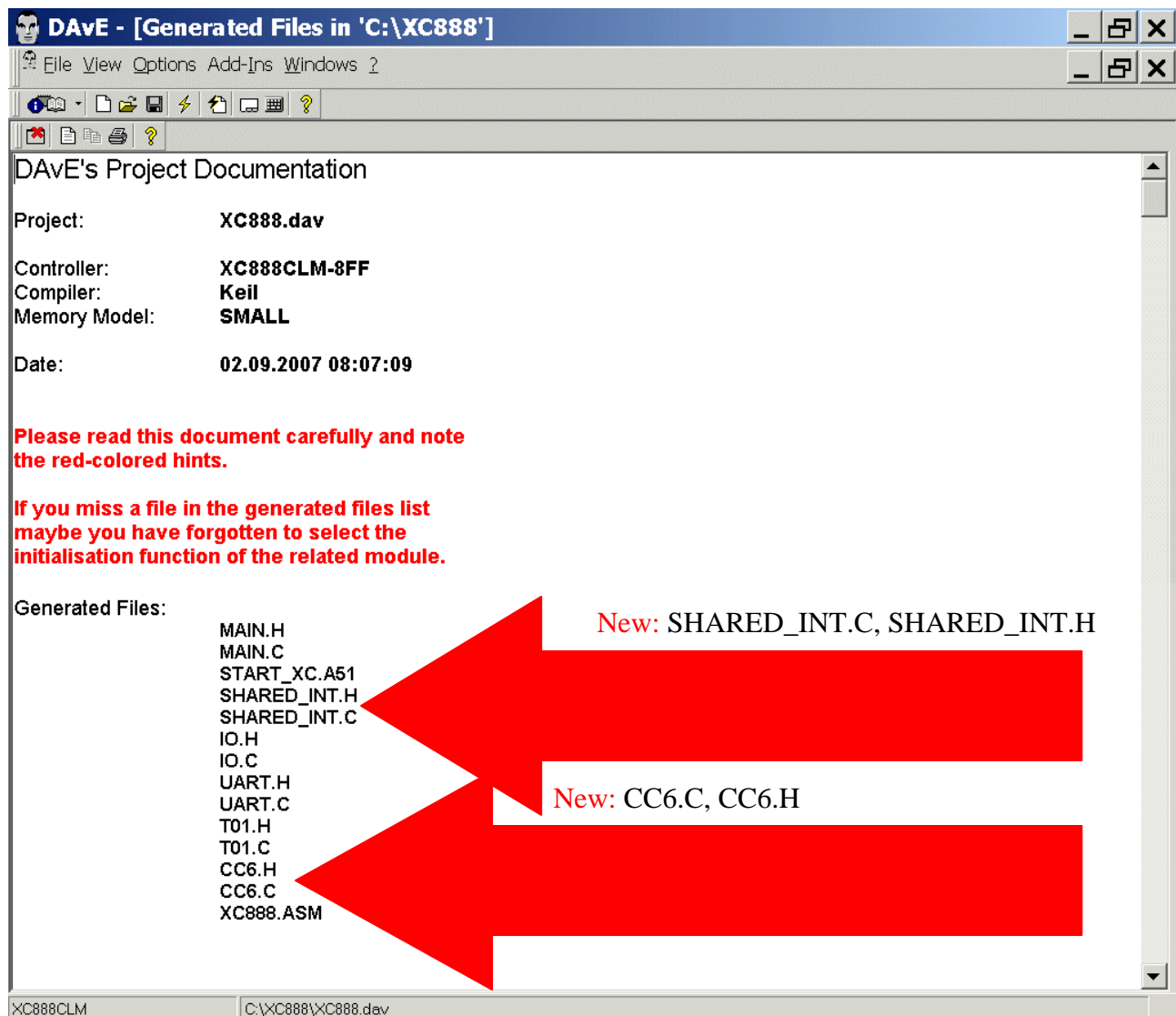
Exit this dialog now by clicking  the close button.

Generate Code:

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



DAvE will show you all the files he has generated
(Project Documentation opens automatically).



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



Start Keil µVision and open your Keil Project:

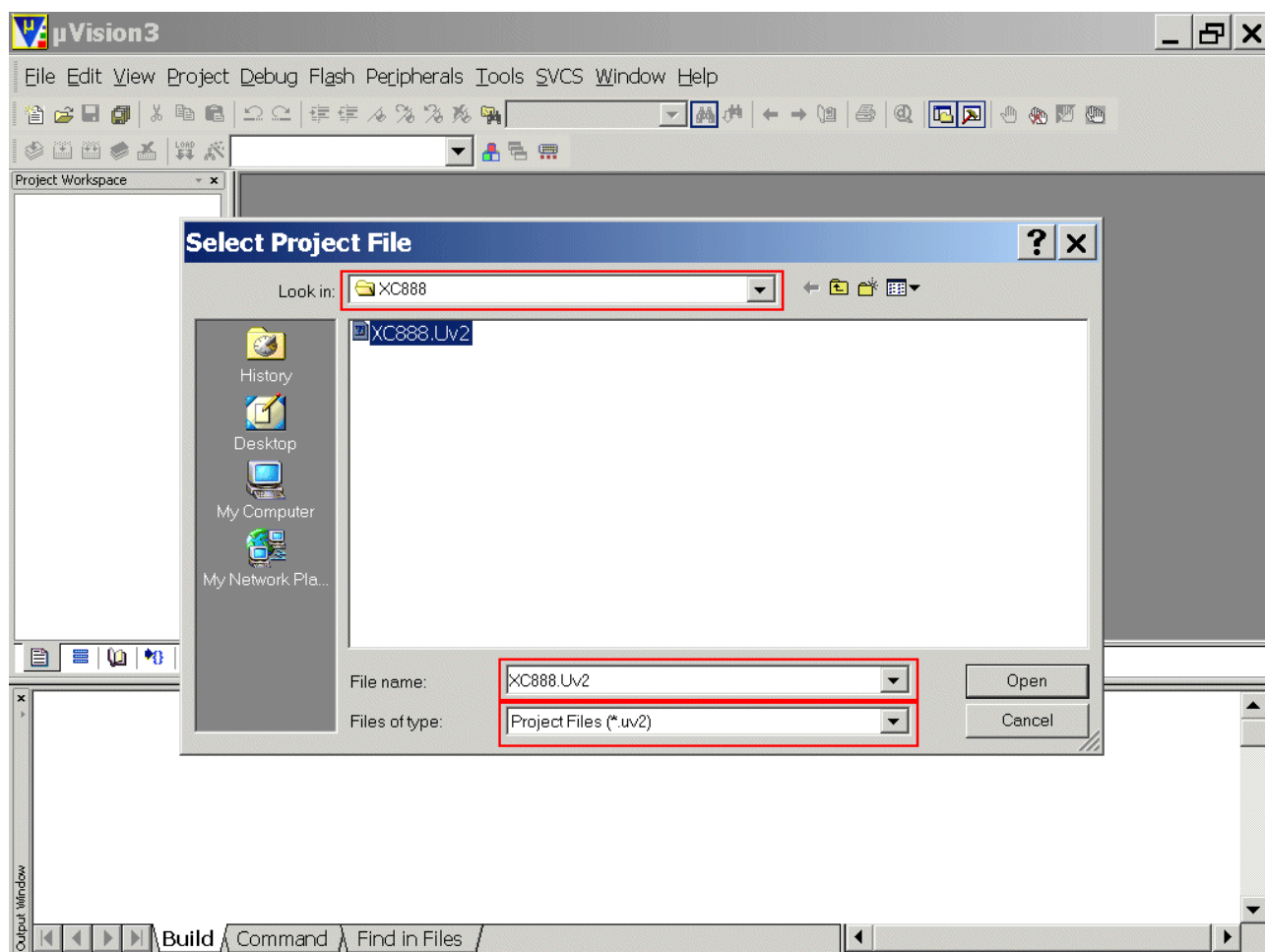
If you see an open project – close it: **Project - Close Project**

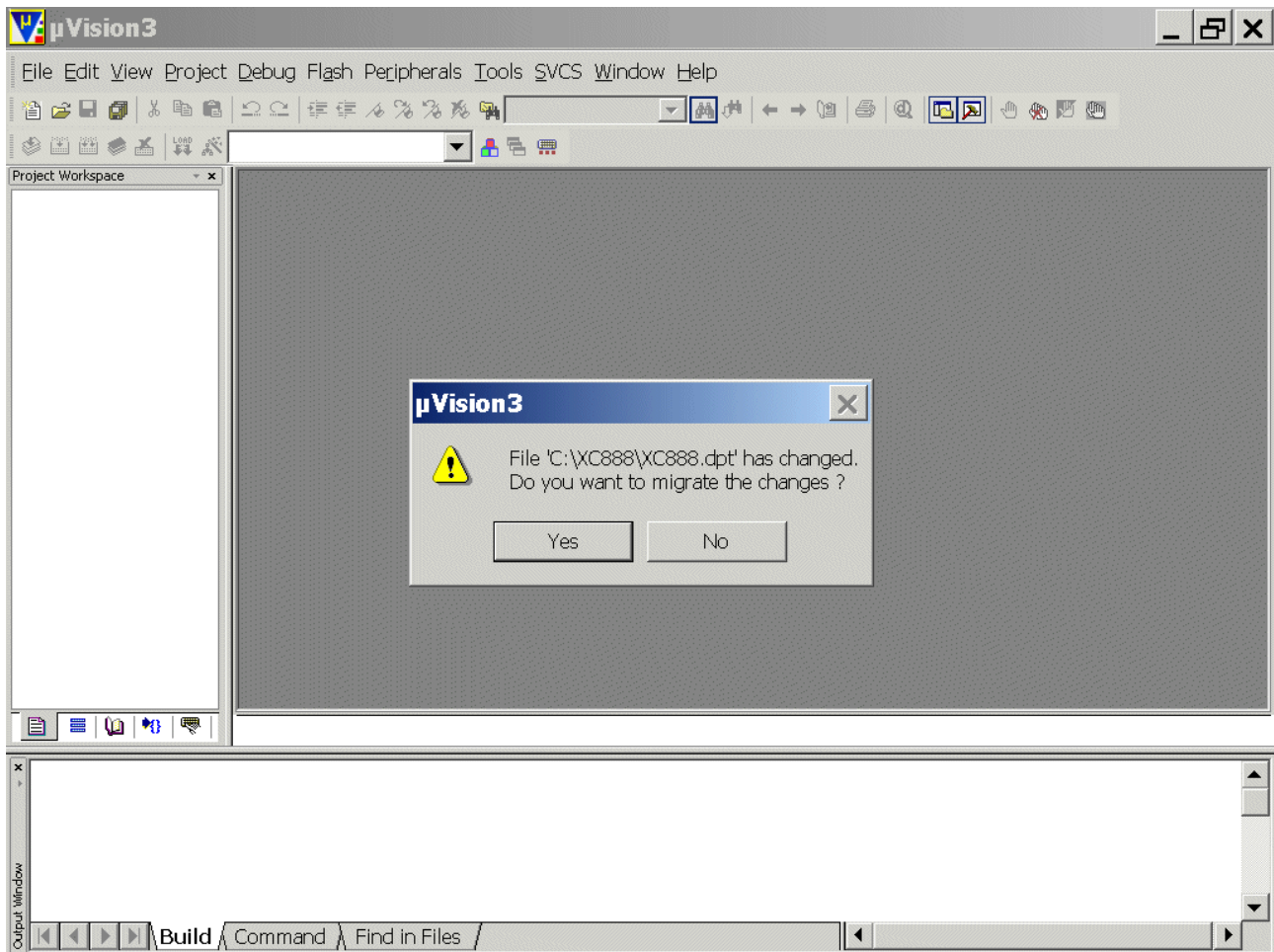
Project - Open Project

Select Project File: **Look in:** choose C:\XC888

Select Project File: **Files of type:** choose Project Files (*.uv2)
choose XC888.Uv2

Open

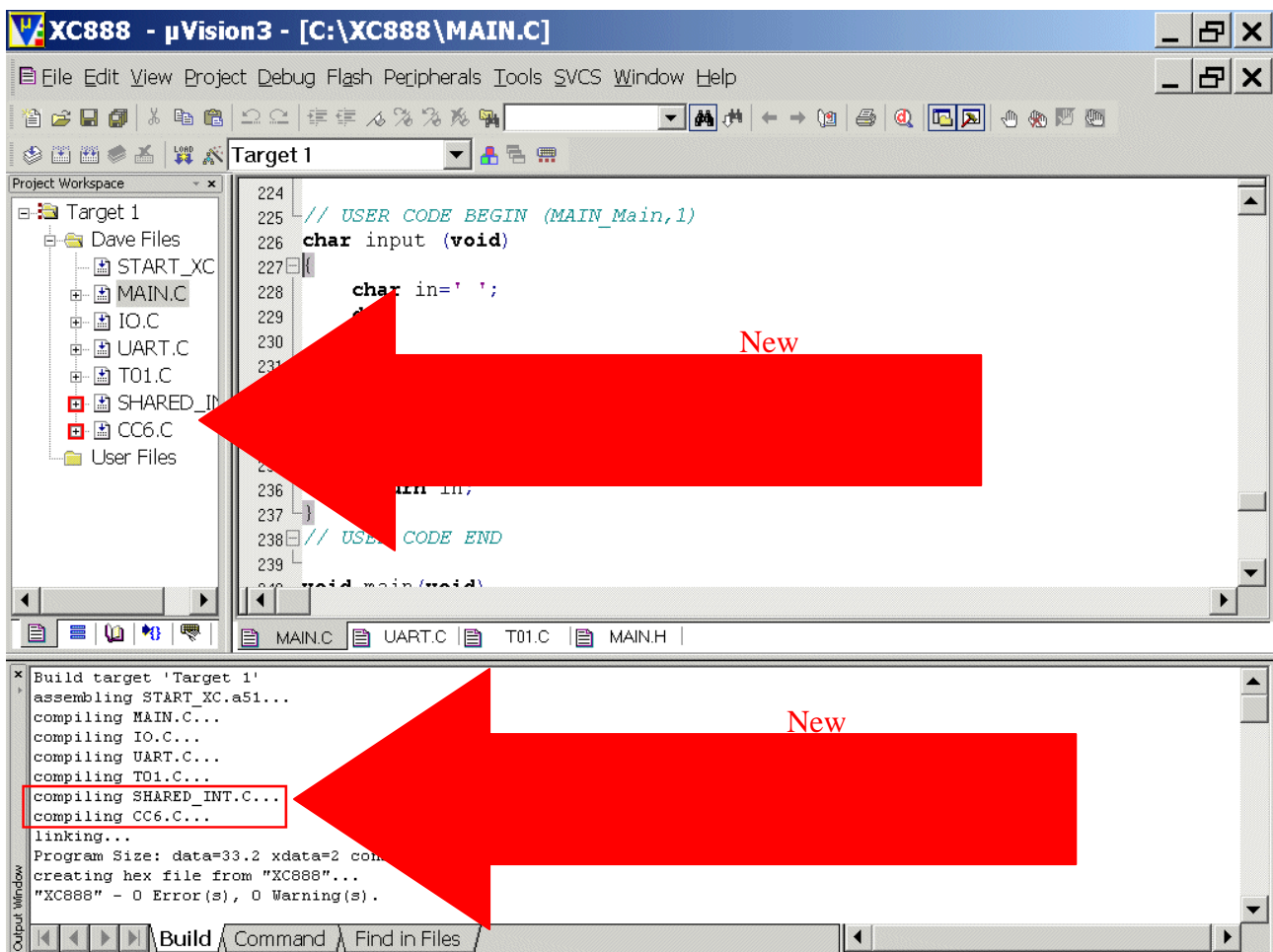
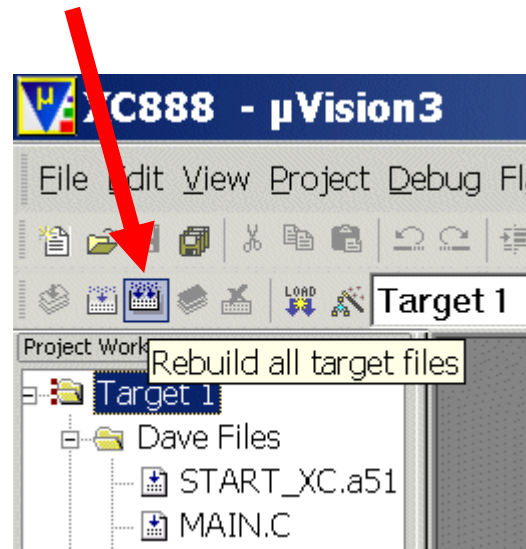




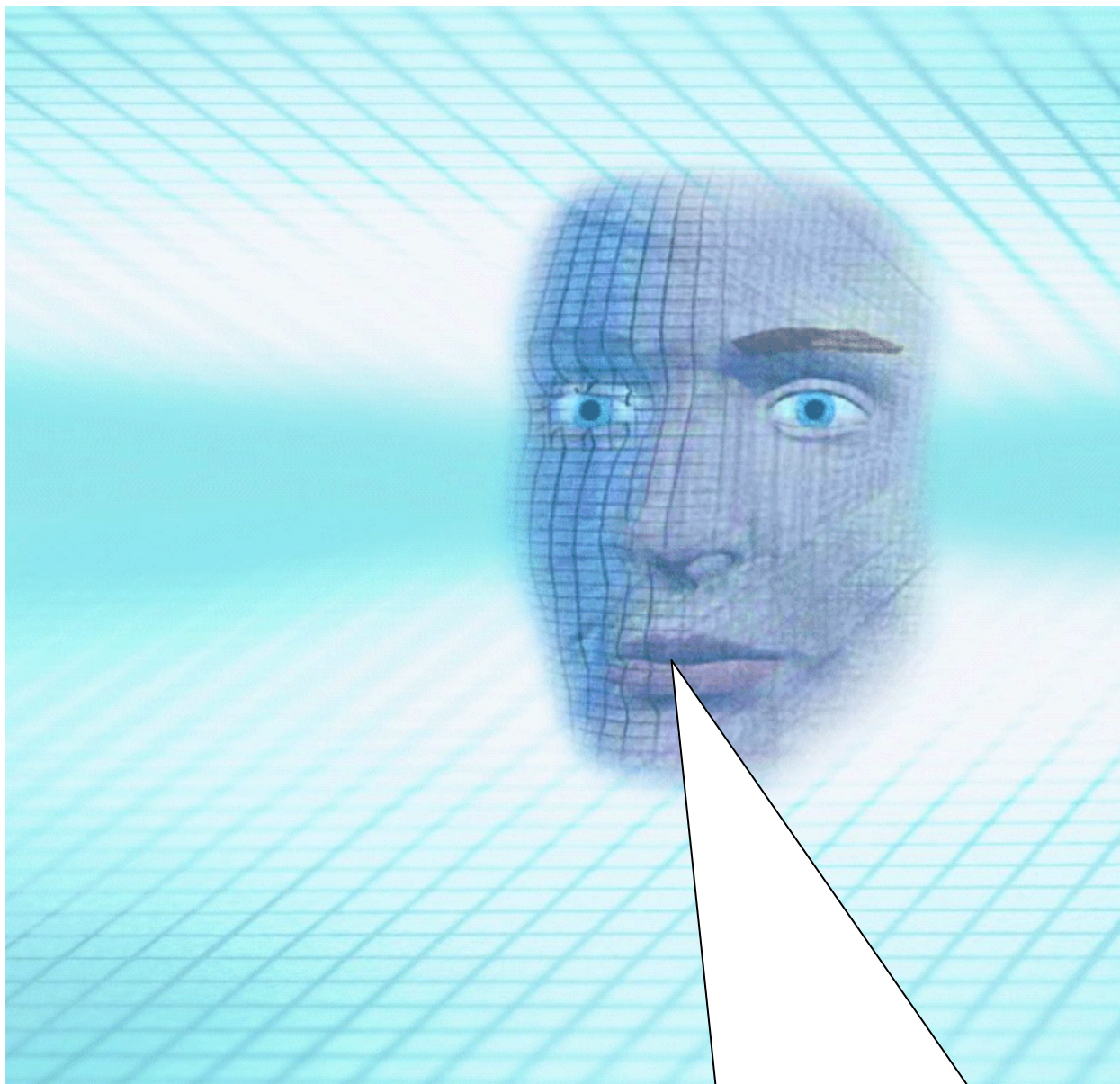
Click Yes

Project – Rebuild all target files

or click



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`'.

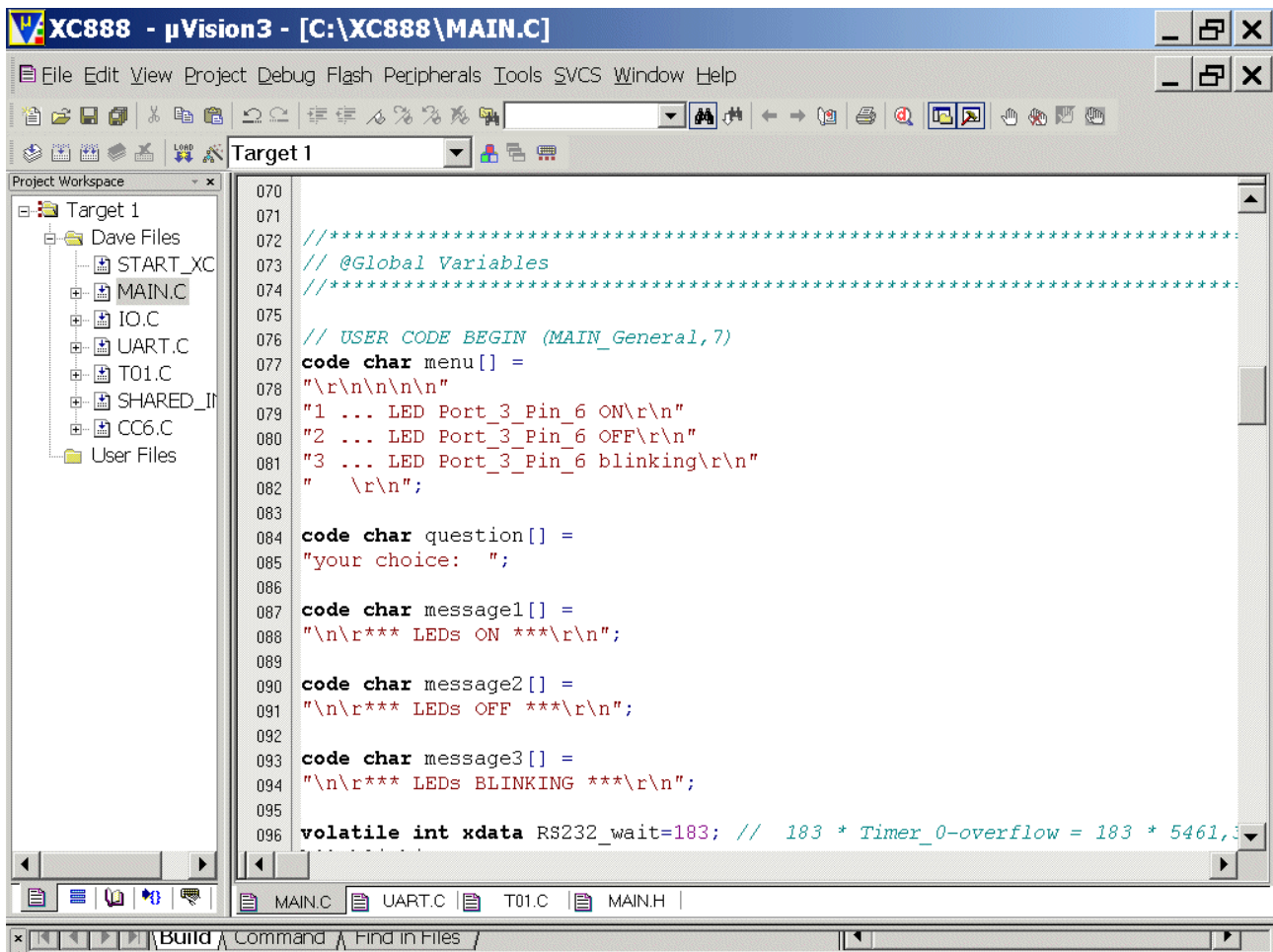
If you wish to change DAVE's generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAVE regenerate code!

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

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

to:

```
code char menu[] =
"\r\n\r\n\r\n"
"1 ... LED Port_3_Pin_6 ON\r\n"
"2 ... LED Port_3_Pin_6 OFF\r\n"
"3 ... LED Port_3_Pin_6 blinking\r\n"
" \r\n";
```



Double click **MAIN.C** and change Global Variables message1, message2 and message3 from:

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

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

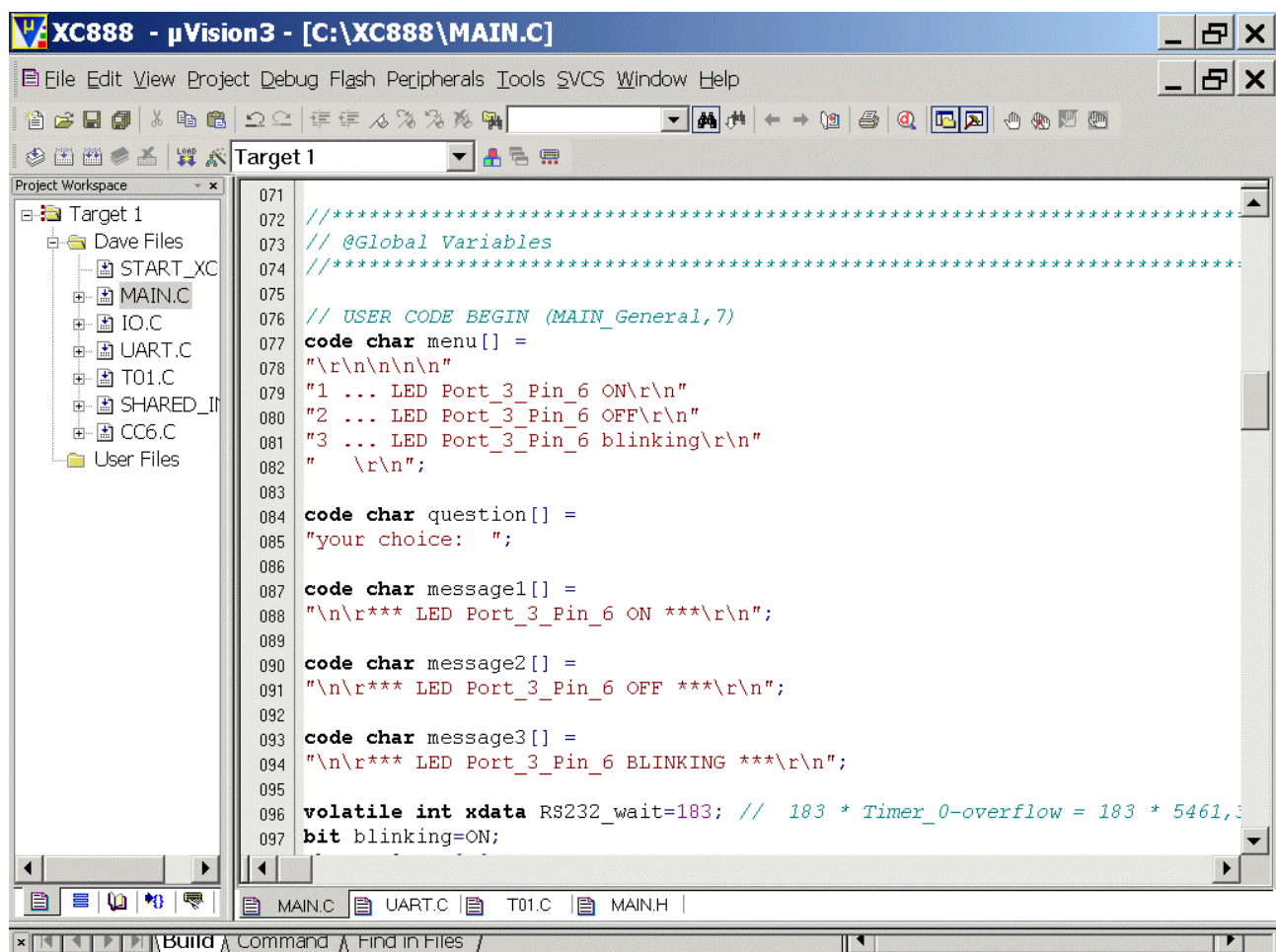
code char message3[] =
"\n*** LEDs BLINKING ***\r\n";
```

to:

```
code char message1[] =
"\n*** LED Port_3_Pin_6 ON ***\r\n";

code char message2[] =
"\n*** LED Port_3_Pin_6 OFF ***\r\n";

code char message3[] =
"\n*** LED Port_3_Pin_6 BLINKING ***\r\n";
```



Double click **MAIN.C** and change code from:

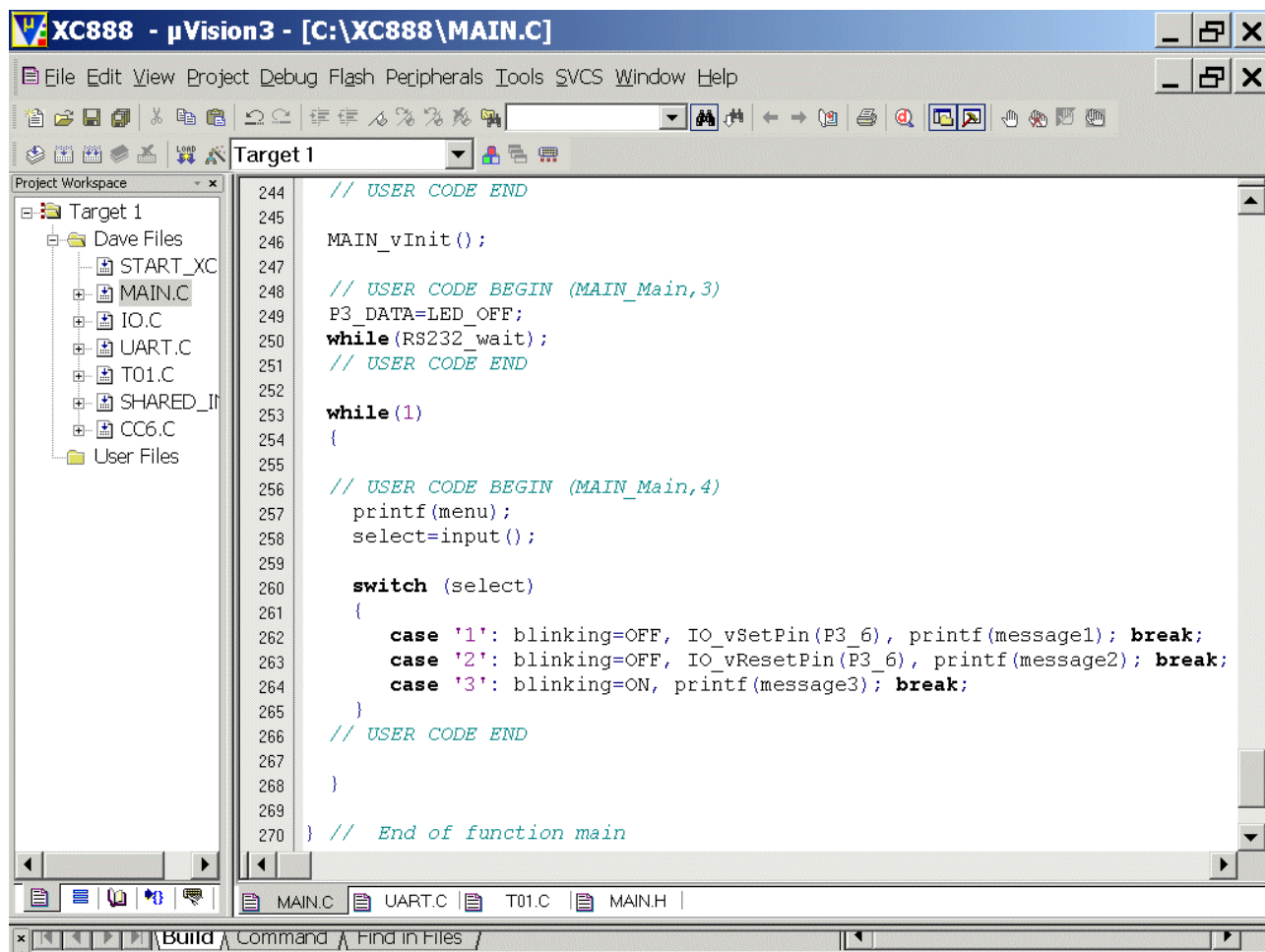
```
printf(menu);
select=input();

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

to:

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

switch (select)
{
    case '1': blinking=OFF, IO_vSetPin(P3_6), printf(message1); break;
    case '2': blinking=OFF, IO_vResetPin(P3_6), printf(message2); break;
    case '3': blinking=ON, printf(message3); break;
}
```



XC888 - µVision3 - [C:\XC888\MAIN.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
 - Dave Files
 - START_XC
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C
 - SHARED_I
 - CC6.C
 - User Files

```

244 // USER CODE END
245
246 MAIN_vInit();
247
248 // USER CODE BEGIN (MAIN_Main,3)
249 P3_DATA=LED_OFF;
250 while(RS232_wait);
251 // USER CODE END
252
253 while(1)
254 {
255
256 // USER CODE BEGIN (MAIN_Main,4)
257 printf(menu);
258 select=input();
259
260 switch (select)
261 {
262     case '1': blinking=OFF, IO_vSetPin(P3_6), printf(message1); break;
263     case '2': blinking=OFF, IO_vResetPin(P3_6), printf(message2); break;
264     case '3': blinking=ON, printf(message3); break;
265 }
266 // USER CODE END
267
268 }
269
270 } // End of function main
  
```

MAIN.C | UART.C | T01.C | MAIN.H

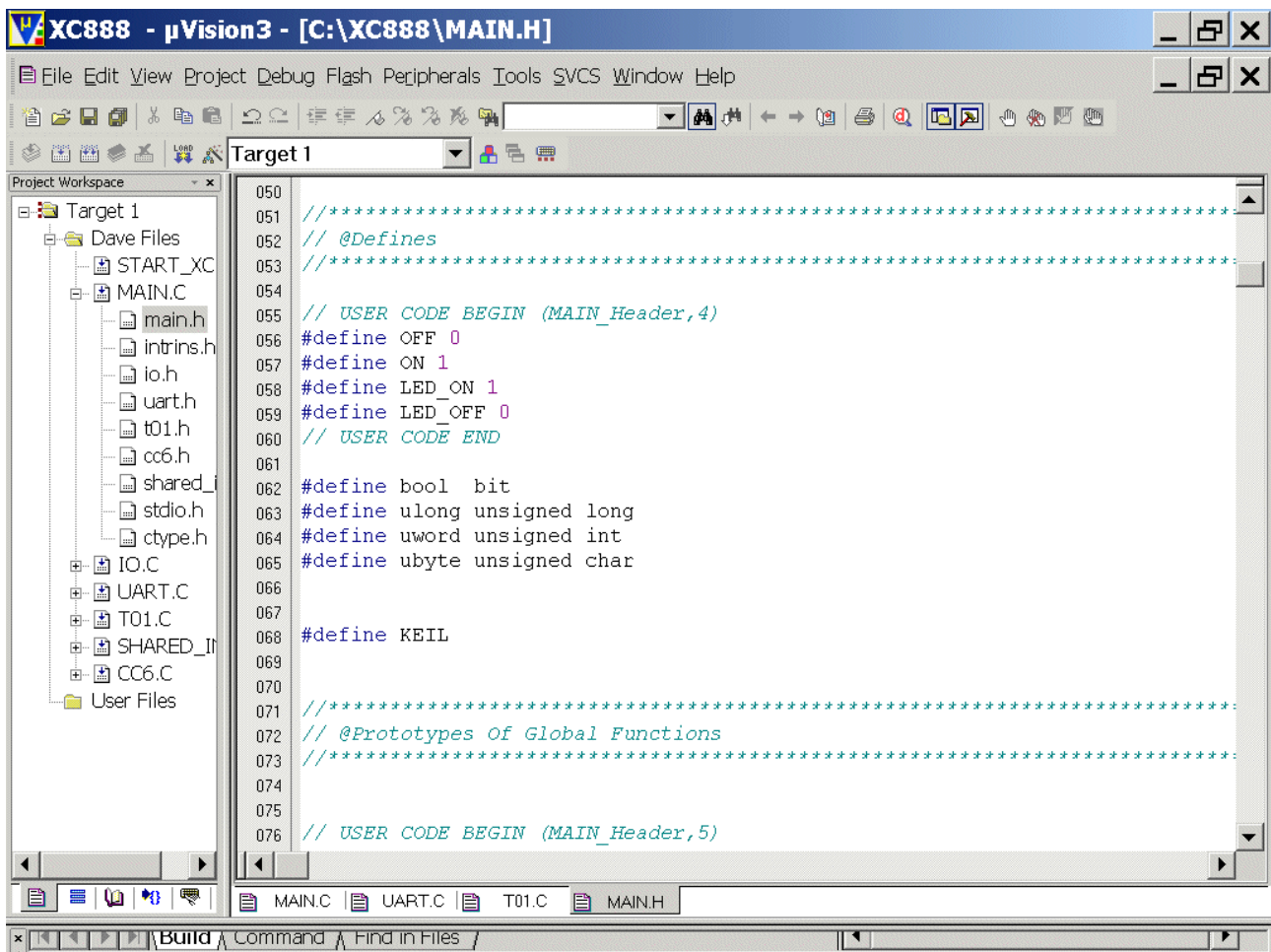
Build Command Find in Files

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

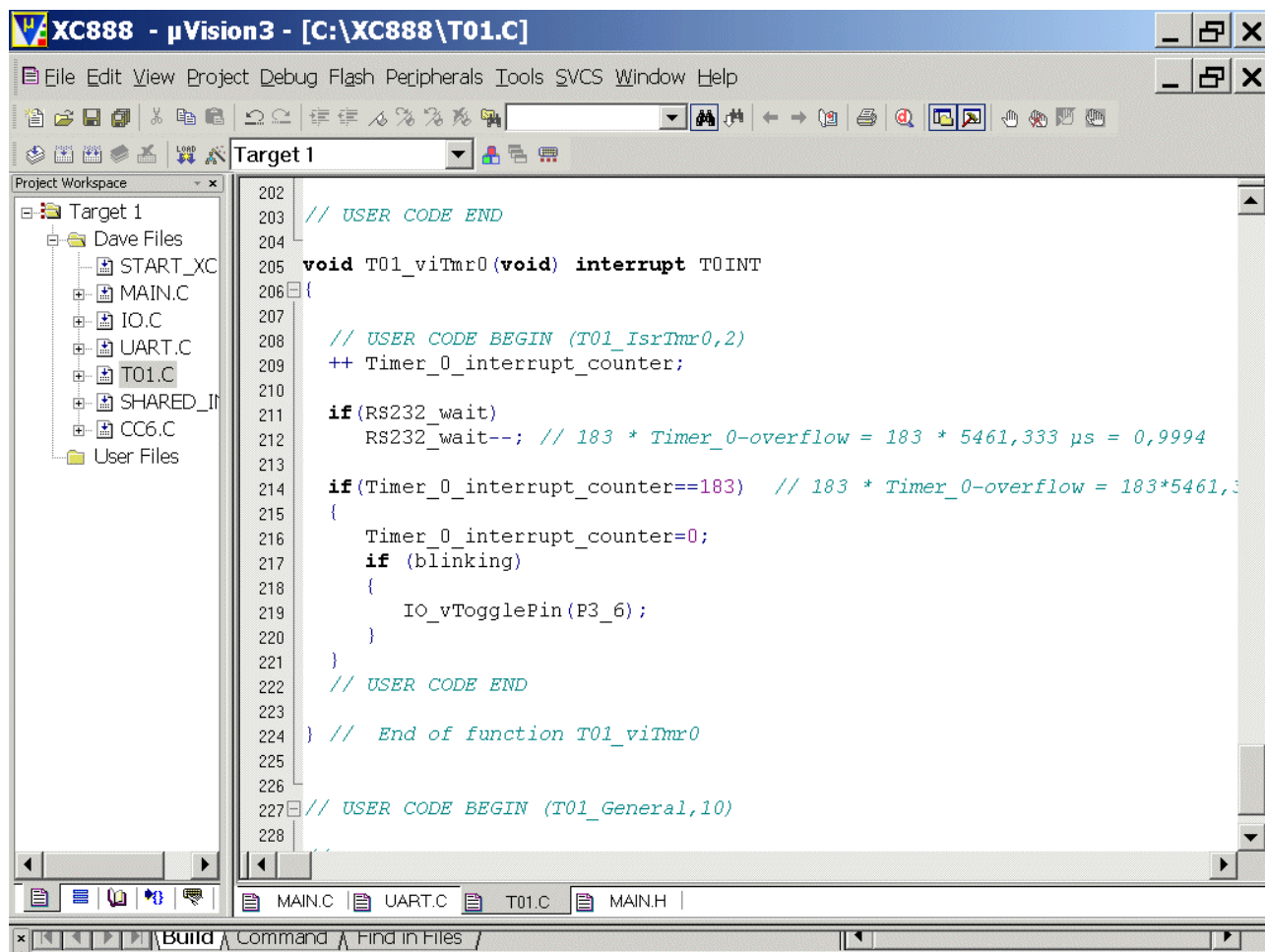
if(Timer_0_interrupt_counter==183) // 183 * Timer_0-overflow = 183*5461,333μs = 0,9994s
{
    Timer_0_interrupt_counter=0;
    if (blinking)
    {
        P3_DATA = P3_DATA^0xFF;
    }
}
```

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);
    }
}
```



XC888 - µVision3 - [C:\XC888\T01.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
 - Dave Files
 - START_XC
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C**
 - SHARED_I
 - CC6.C
 - User Files

```

202
203 // USER CODE END
204
205 void T01_viTmr0(void) interrupt T0INT
206 {
207
208 // USER CODE BEGIN (T01_IsrTmr0,2)
209 ++ Timer_0_interrupt_counter;
210
211 if(RS232_wait)
212     RS232_wait--; // 183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994
213
214 if(Timer_0_interrupt_counter==183) // 183 * Timer_0-overflow = 183*5461,3
215 {
216     Timer_0_interrupt_counter=0;
217     if (blinking)
218     {
219         IO_vTogglePin(P3_6);
220     }
221 }
222 // USER CODE END
223
224 } // End of function T01_viTmr0
225
226
227 // USER CODE BEGIN (T01_General,10)
228

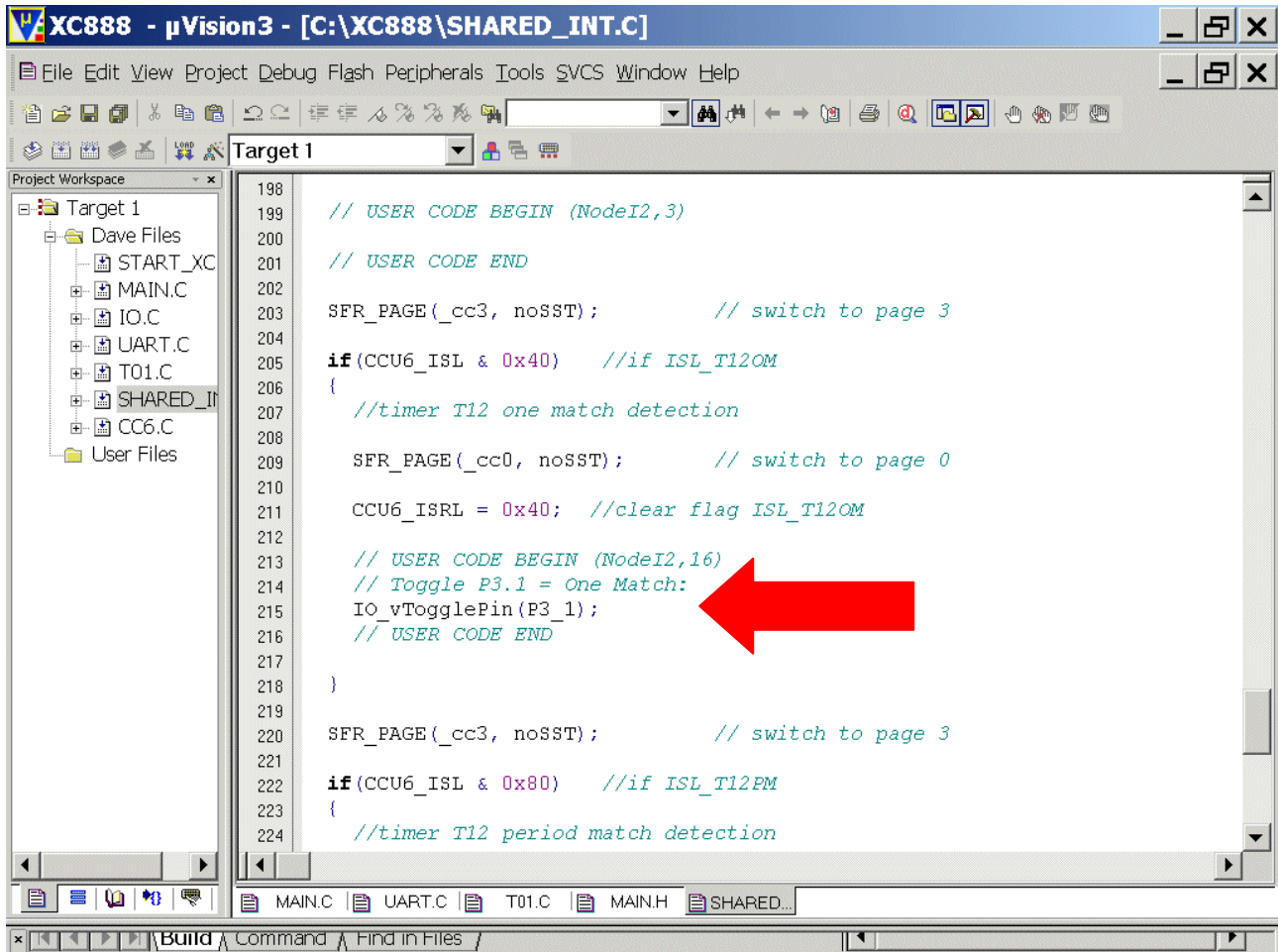
```

MAIN.C | UART.C | T01.C | MAIN.H

Build Command Find in Files

Double click **SHARED_INT.C** and insert Code:

```
// Toggle P3.1 = One Match:
IO_vTogglePin(P3_1);
```



Remember:

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

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75
P3.7	COU63_0	not used

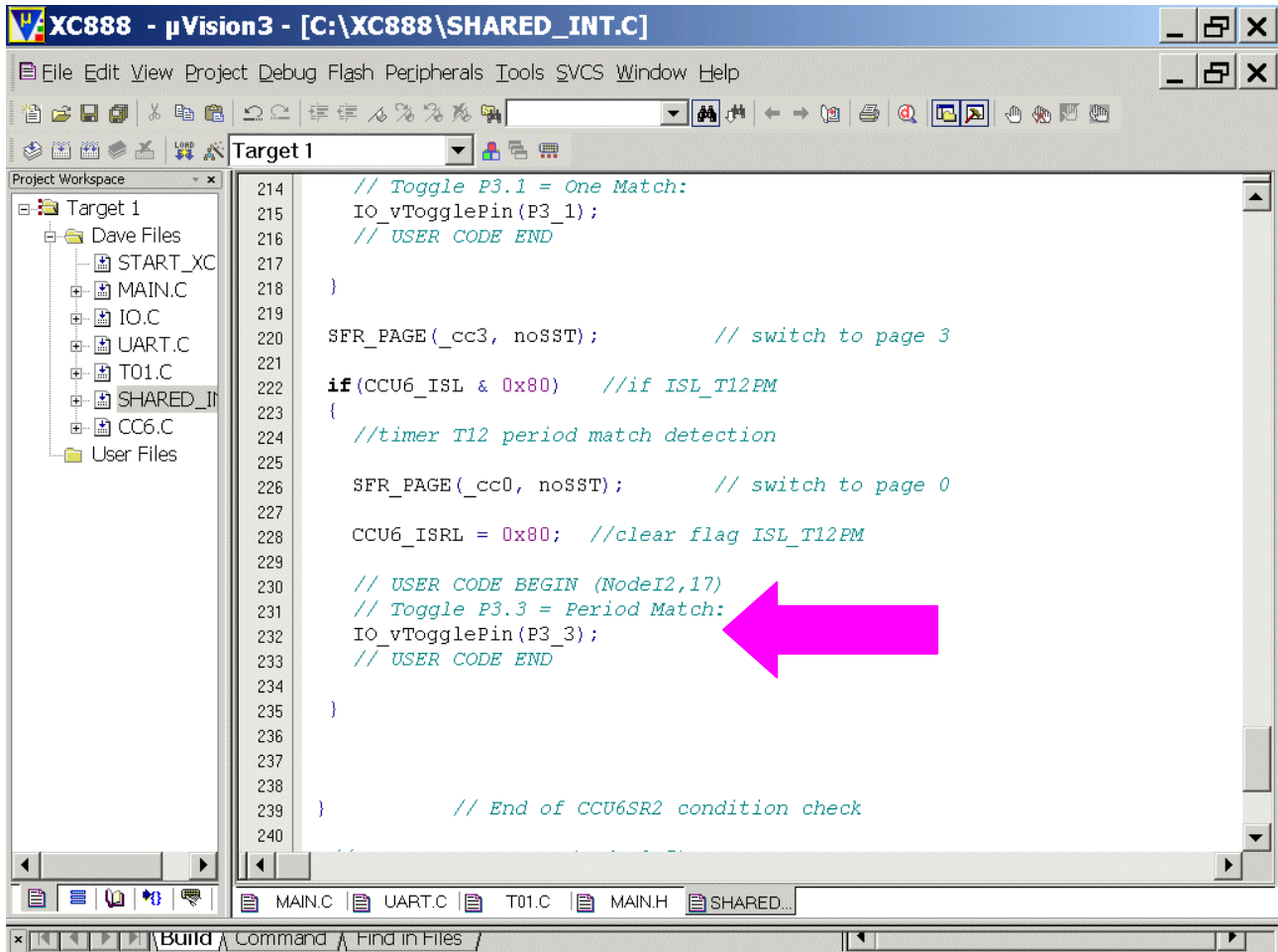


GPIO use:

Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.5	not used	
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

Double click **SHARED_INT.C** and insert Code:

```
// Toggle P3.3 = Period Match:
IO_vTogglePin(P3_3);
```



Remember:

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

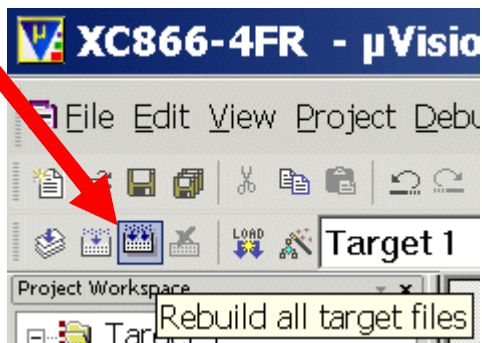
Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75
P3.7	COU63_0	not used

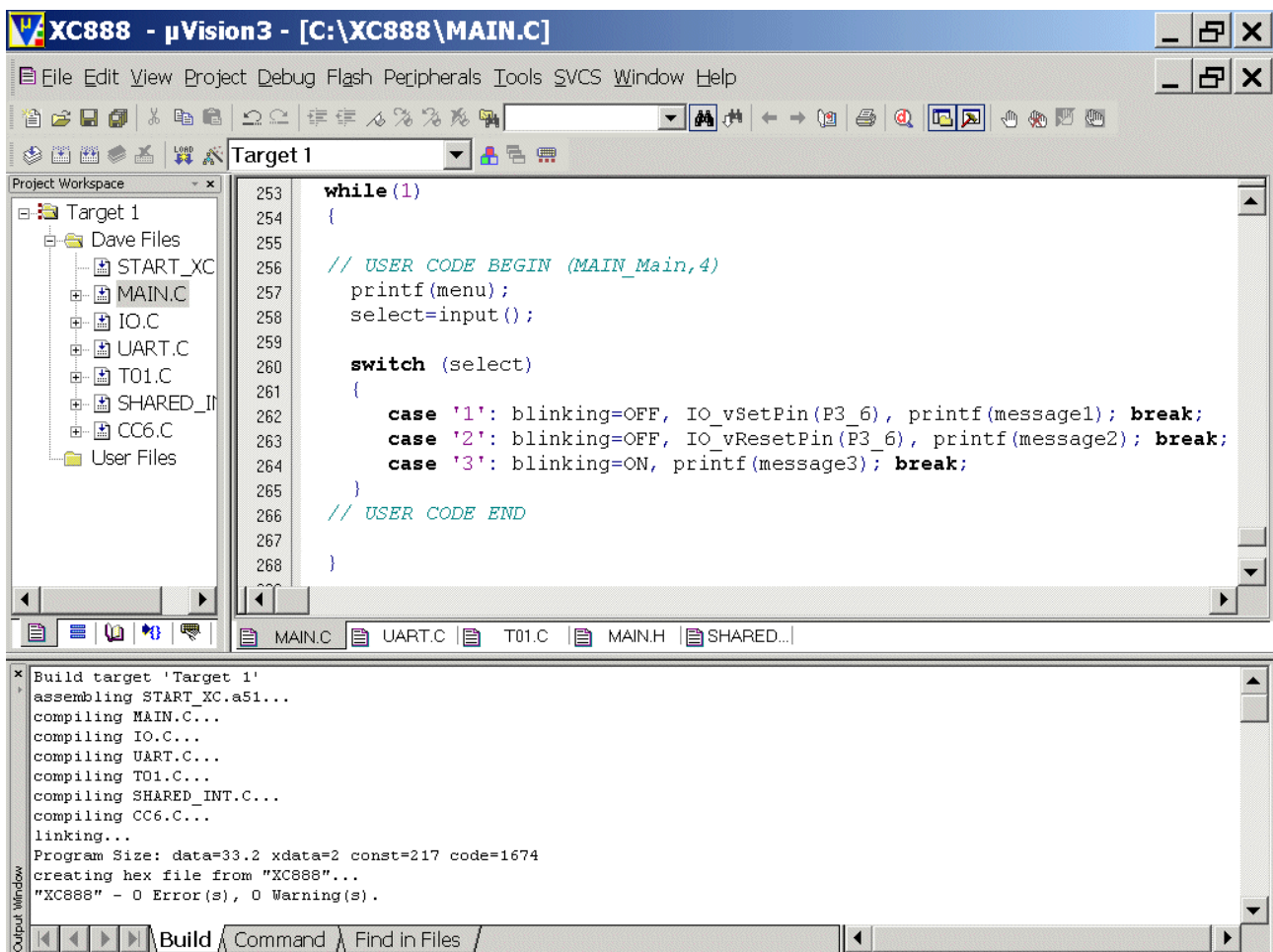


GPIO use:

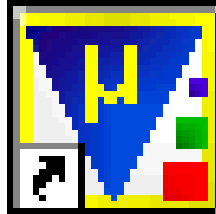
Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.5	not used	
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

Generate your application program:

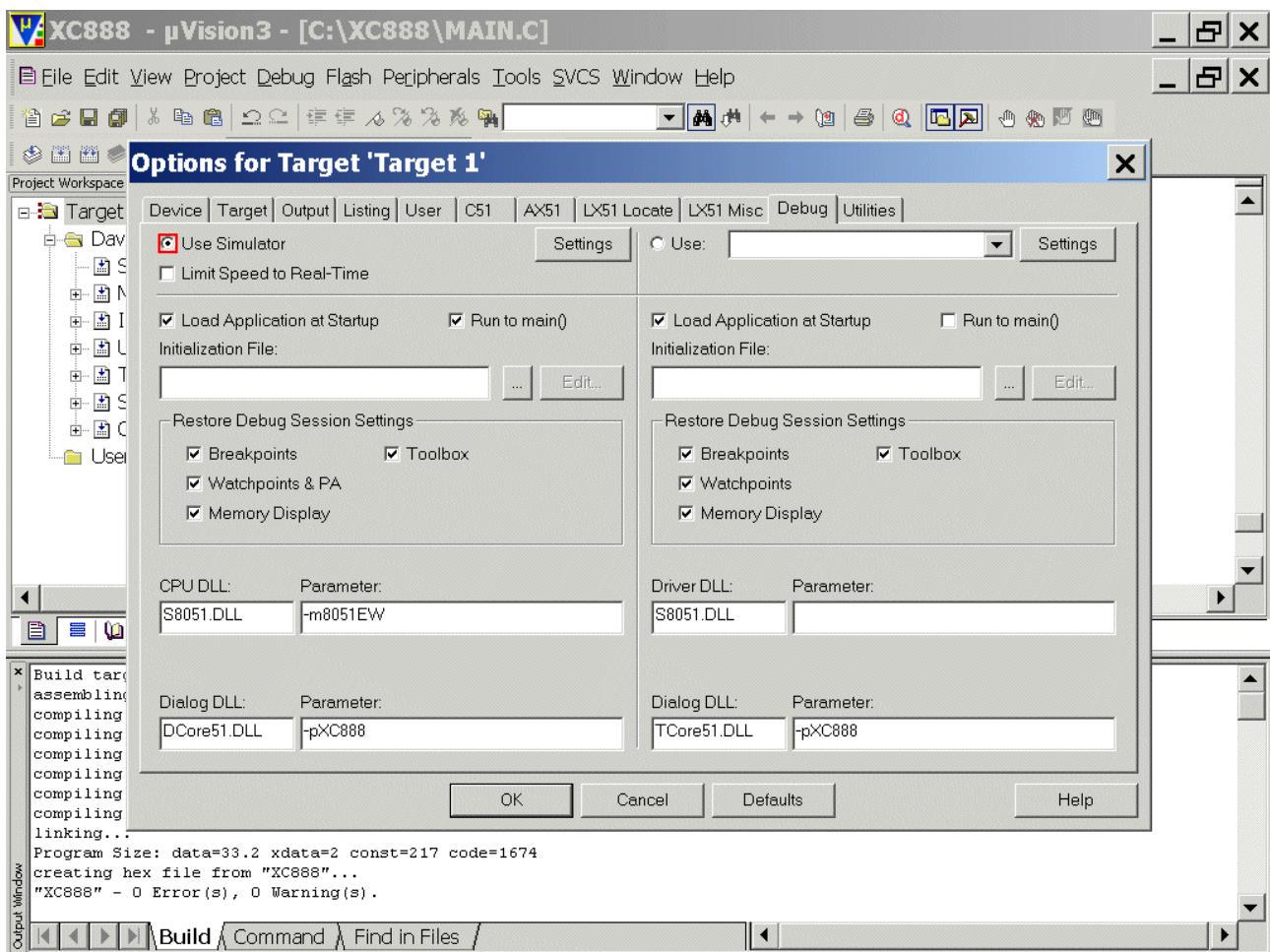
<p>Project – Rebuild all target files</p>	<p>or</p>	<p>click</p> 
---	-----------	---



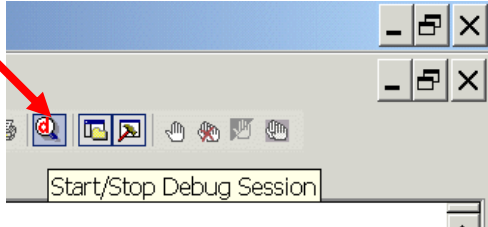
See the result:
Using the Simulator:

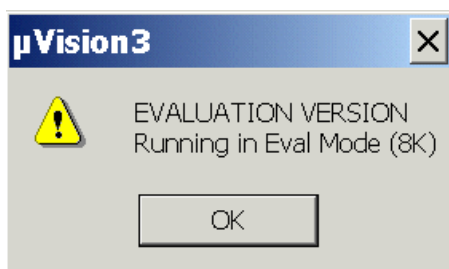


Check/Configure: ☒ Use Simulator

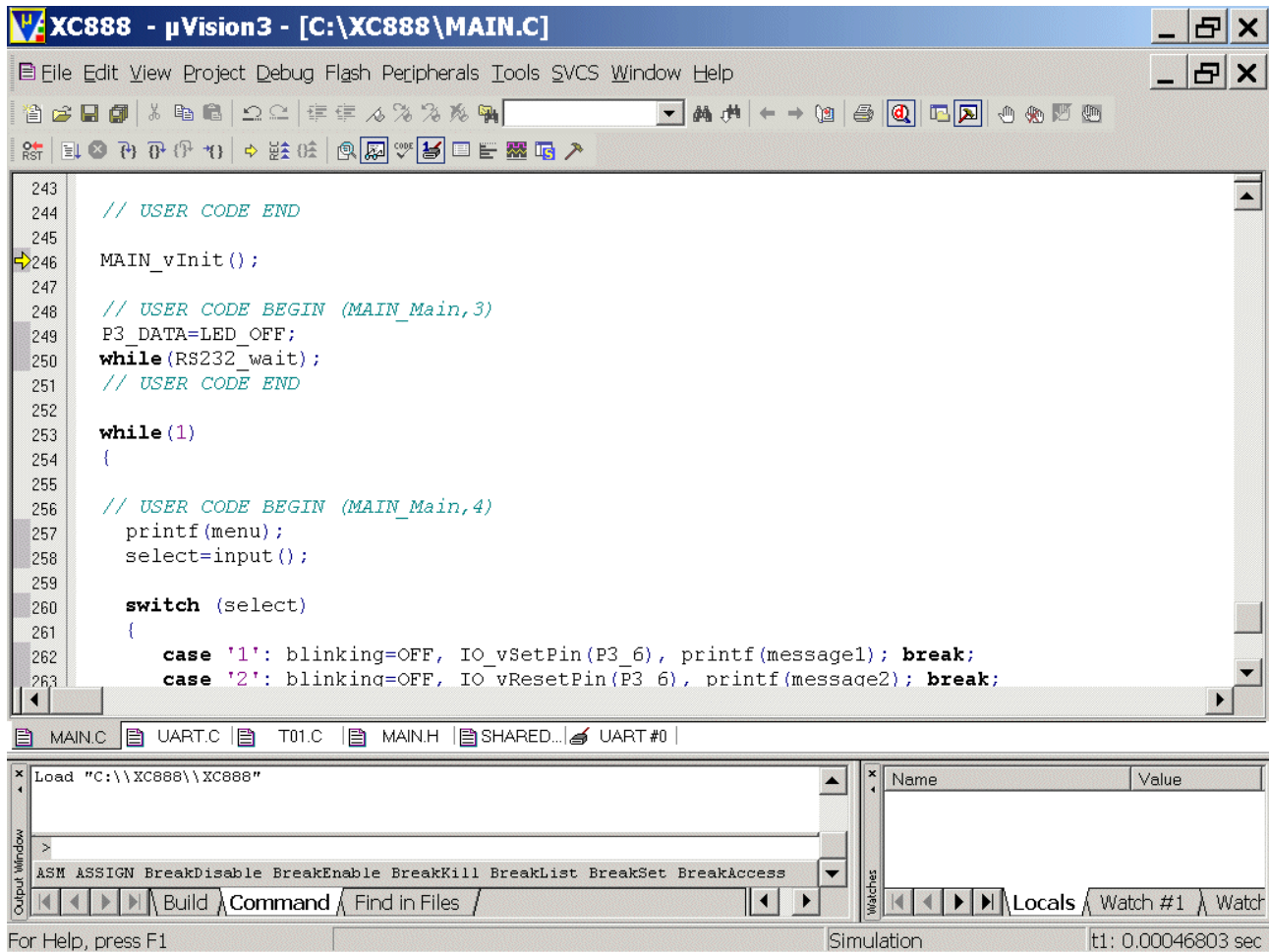


OK

Debug - Start/Stop Debug Session	or	click
		

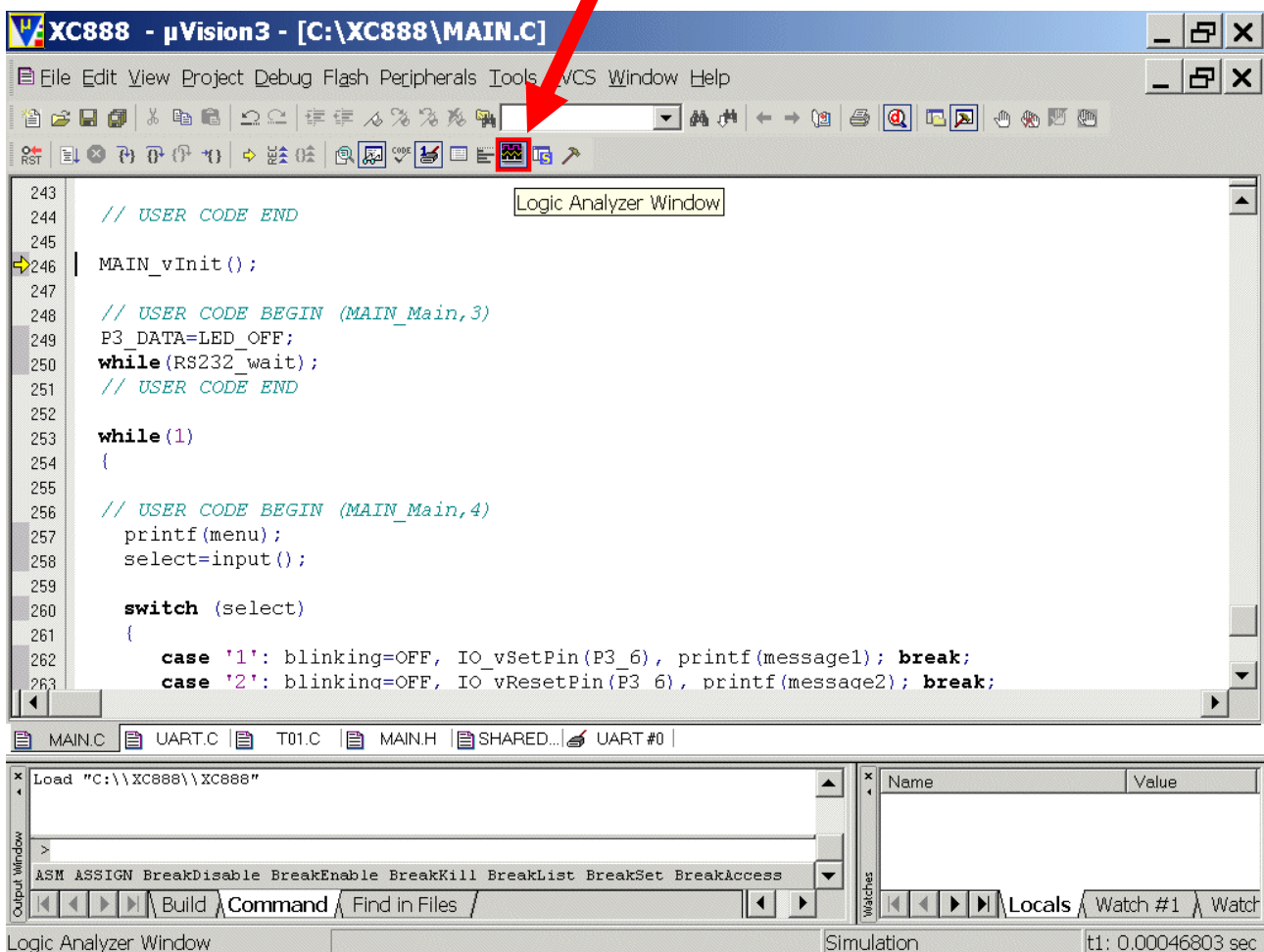


OK

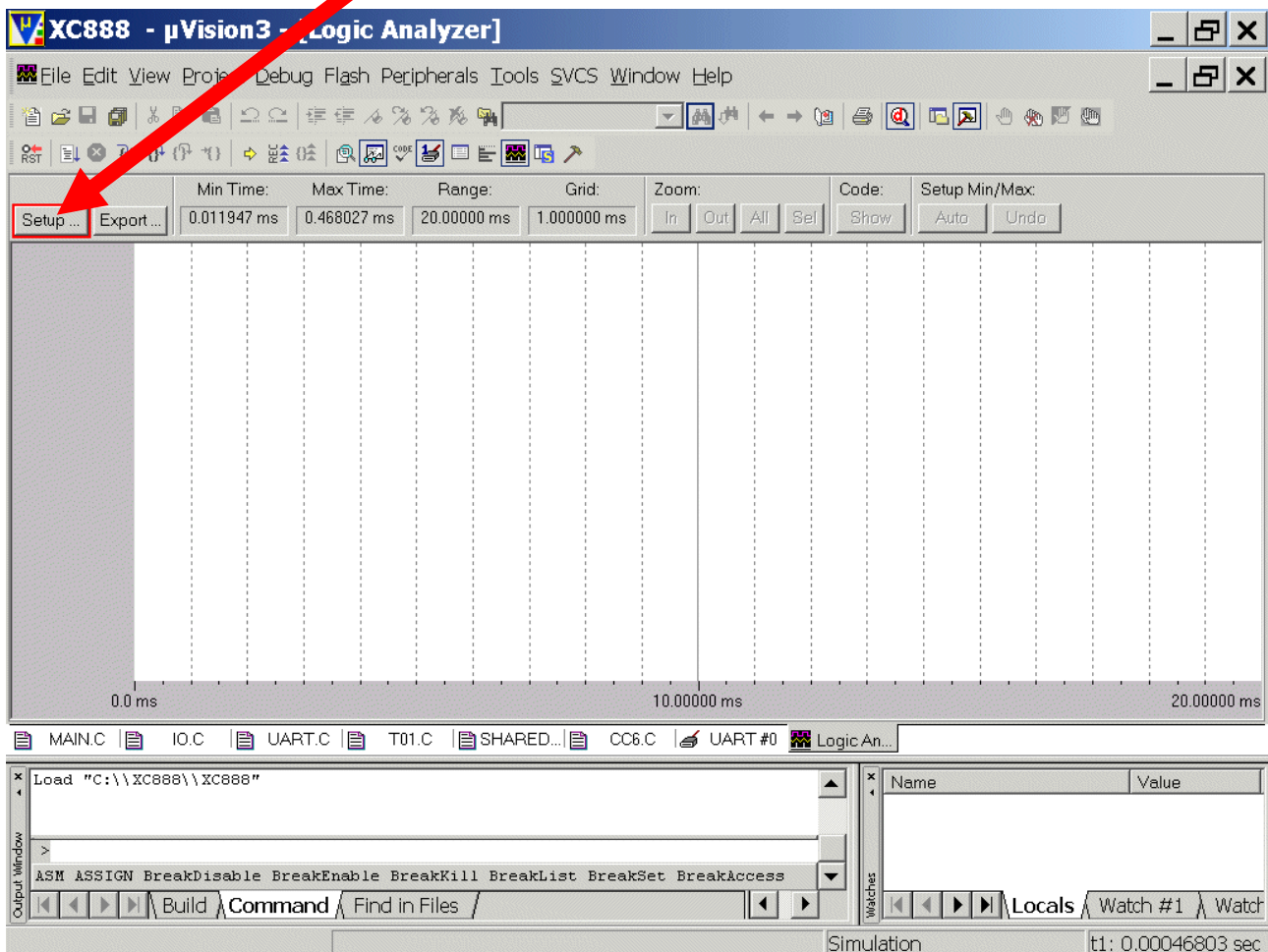


Configure the Software-Simulator-LGA (Logic Analyzer):

Click 



Click



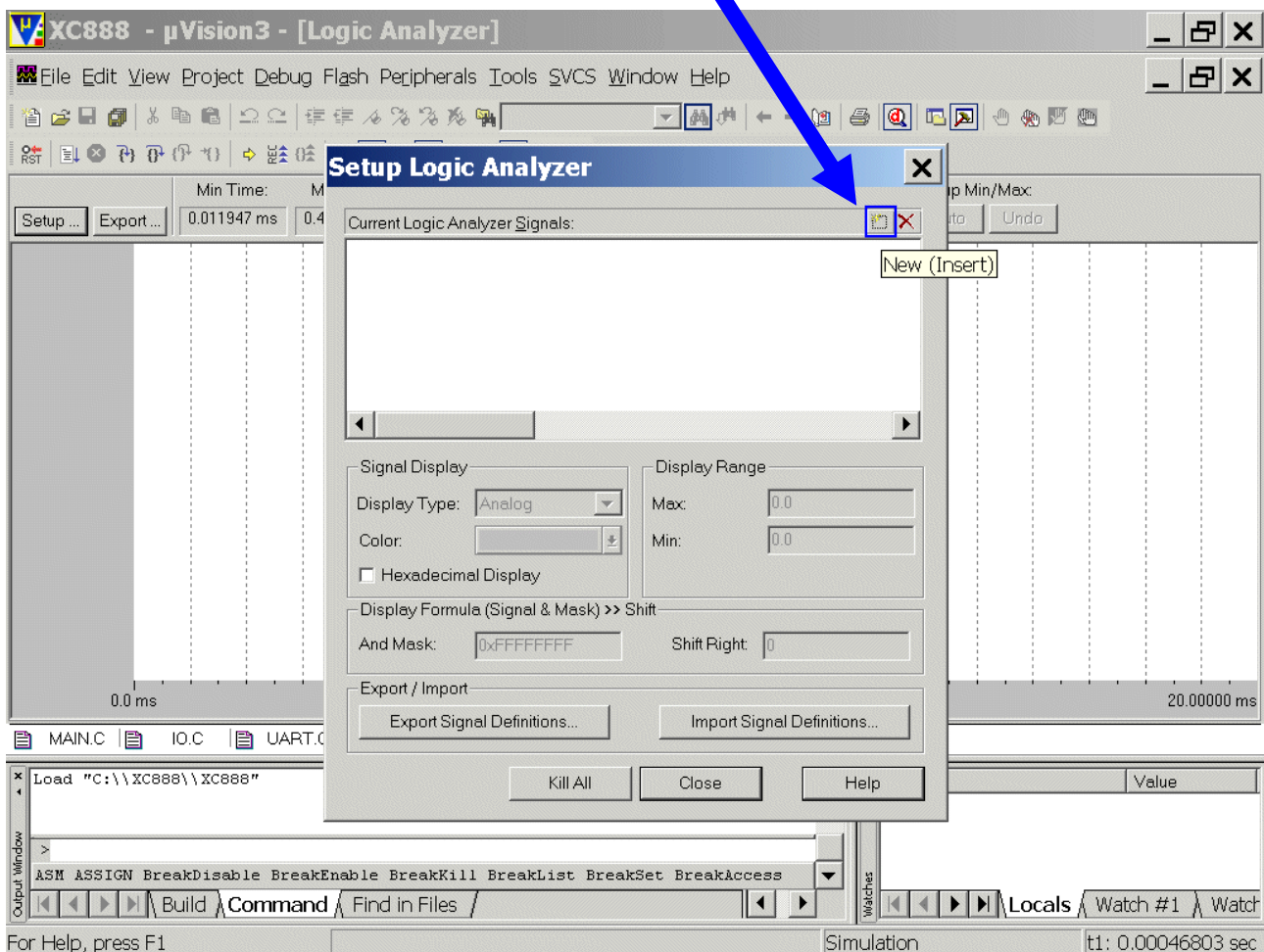
Note:

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75

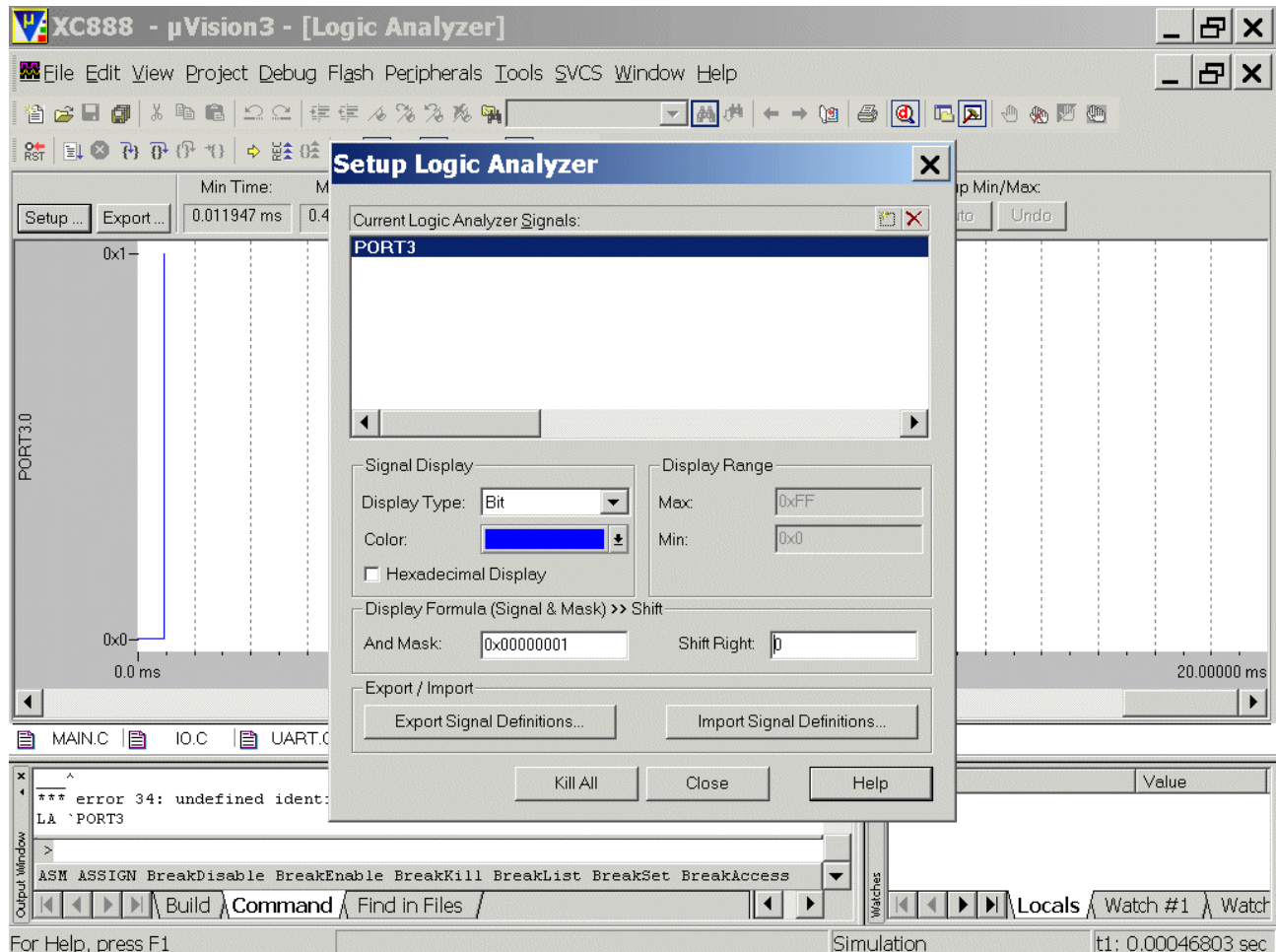


Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

click



Insert PORT 3 <RETURN>
Display Type: select Bit
Color: select "blue"
And Mask: insert 0x1
Shift Right: insert 0



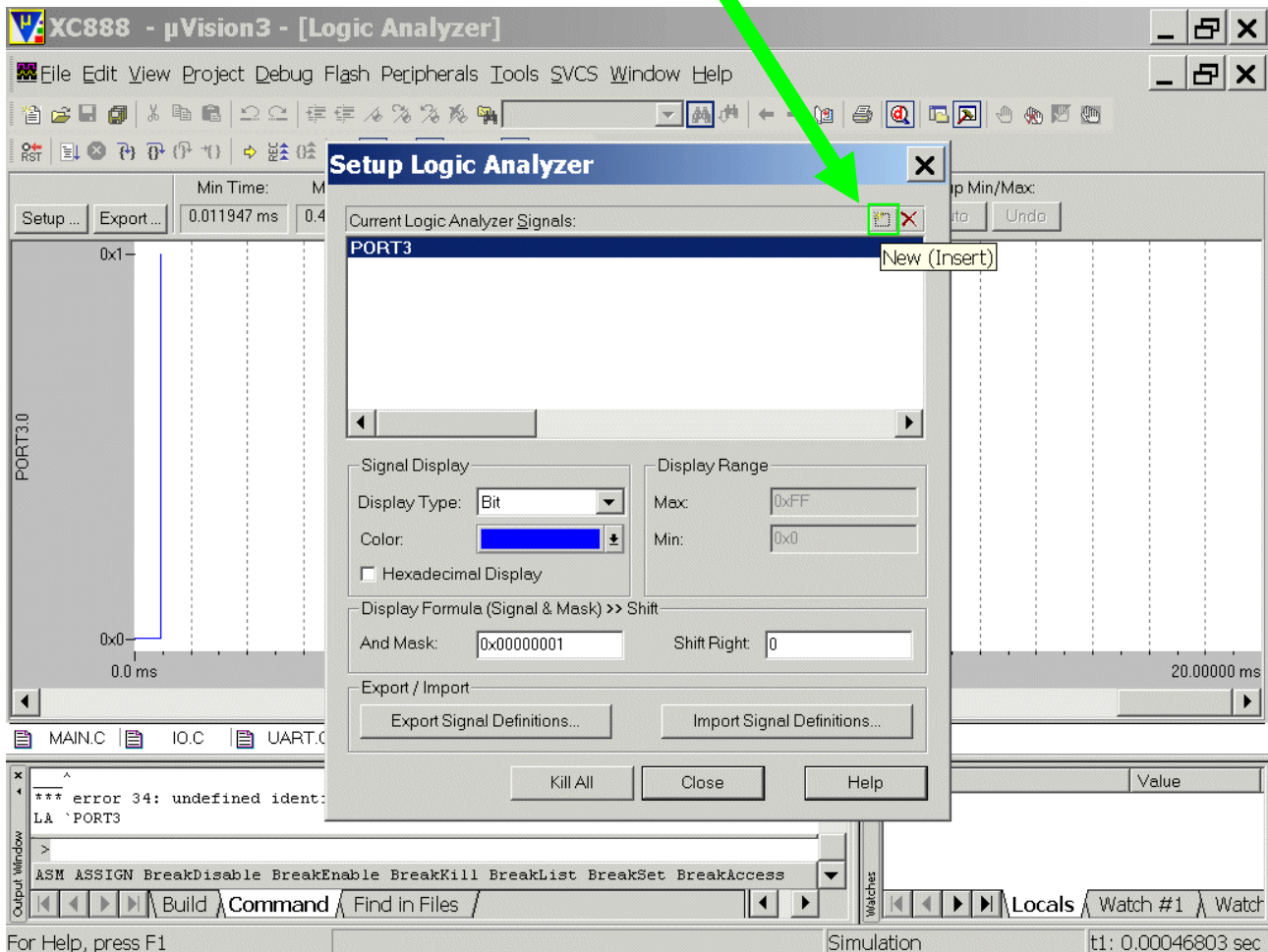
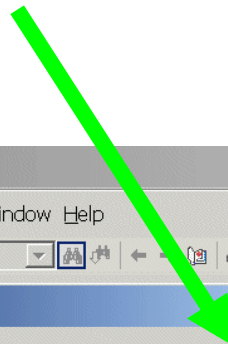
Note:

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75

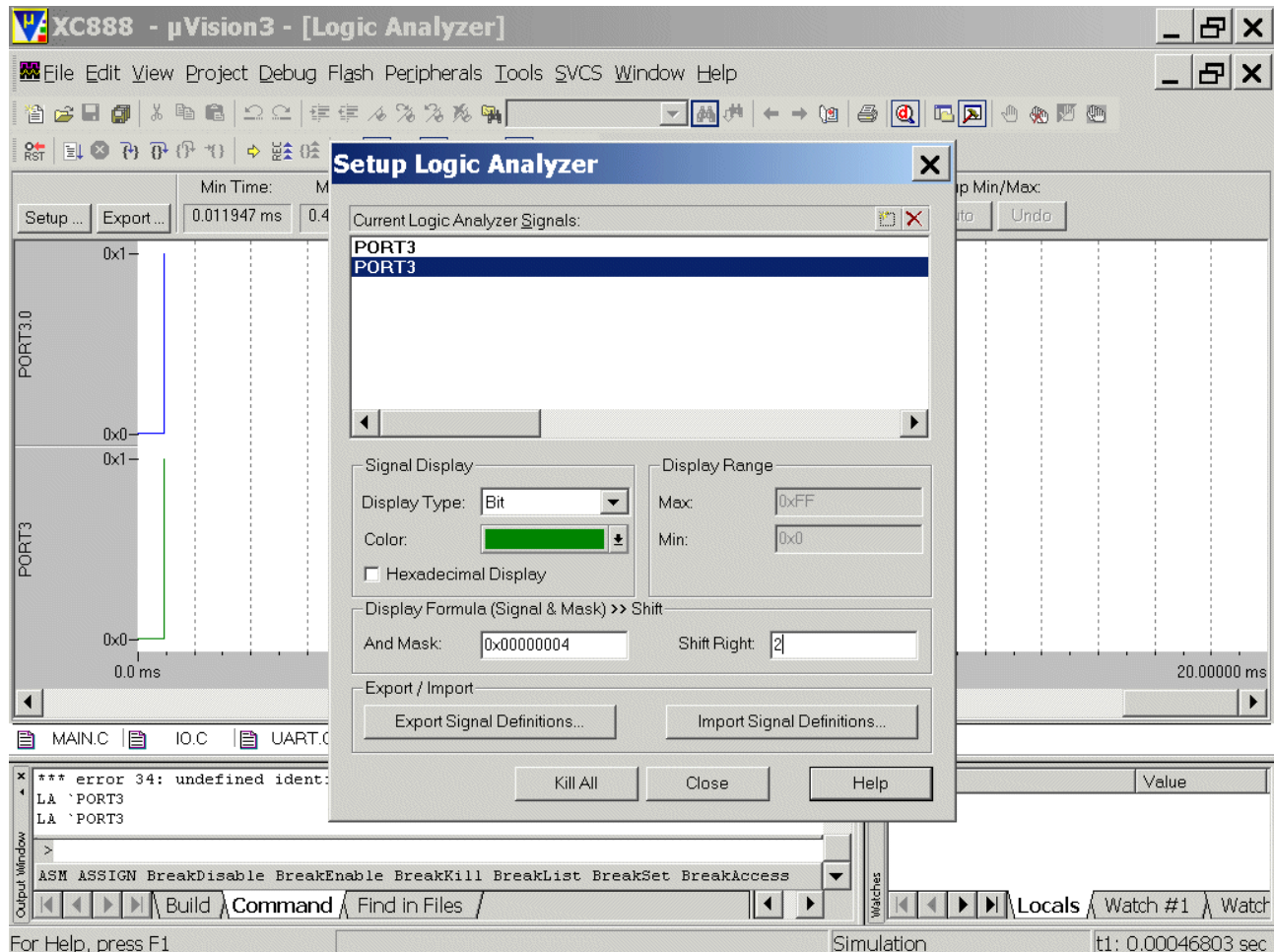


Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

click

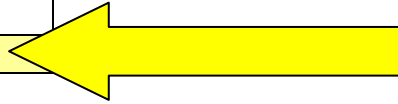


Insert PORT 3 <RETURN>
Display Type: select Bit
Color: select "green"
And Mask: insert 0x4
Shift Right: insert 2



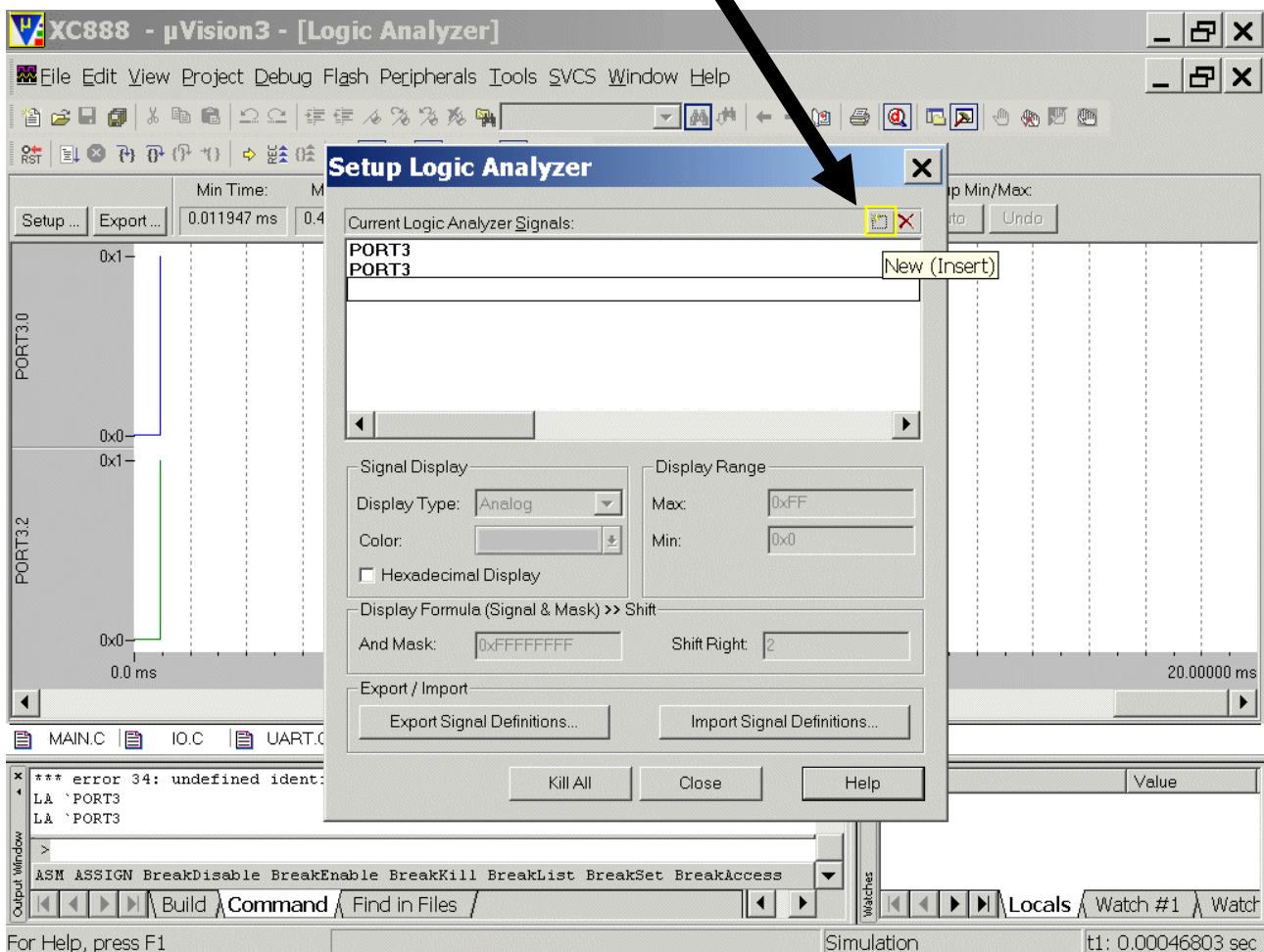
Note:

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75



Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

click



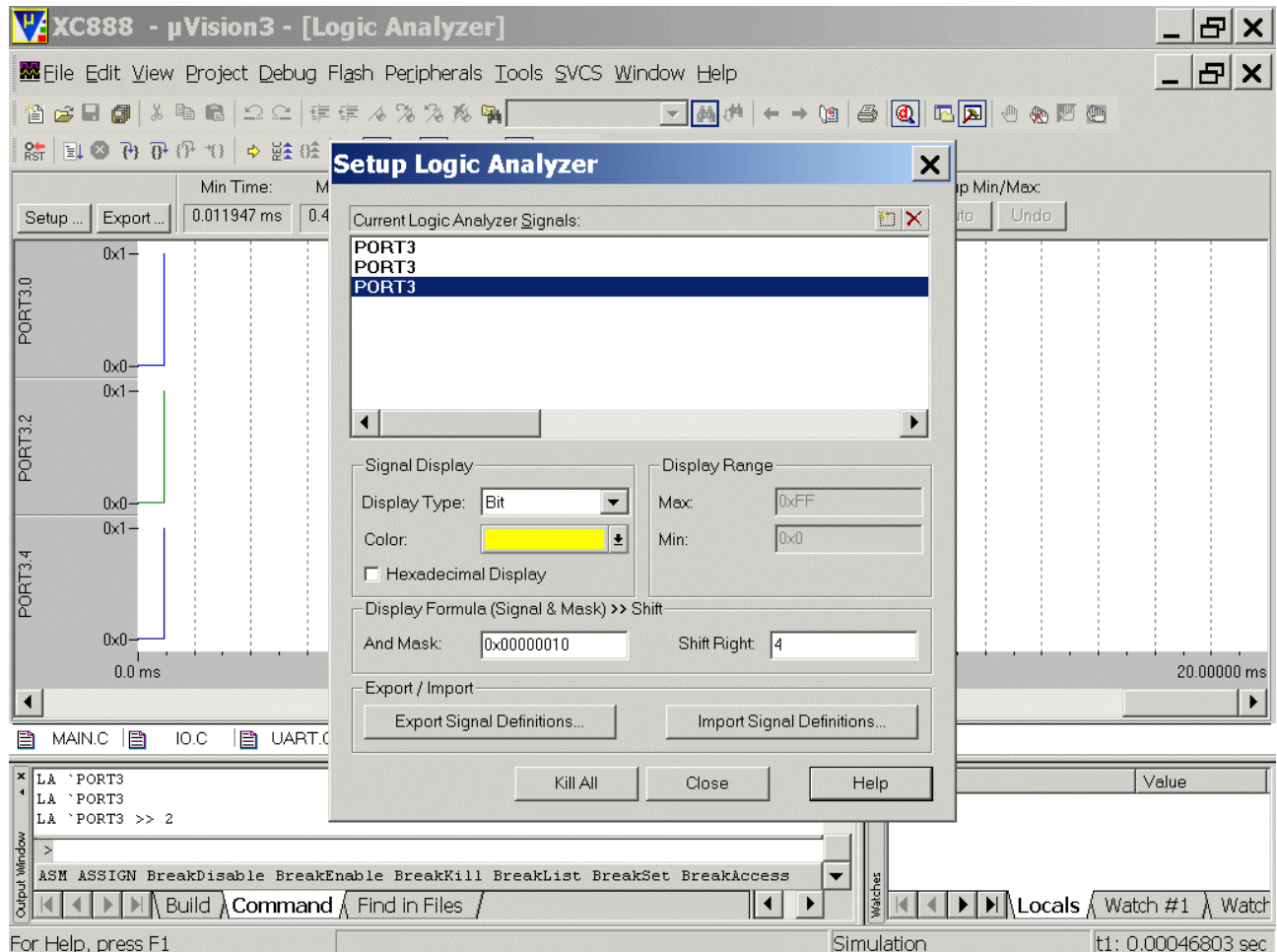
Insert PORT 3 <RETURN>

Display Type: select Bit

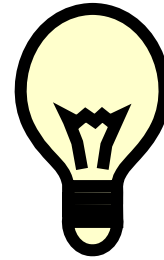
Color: select "yellow"

And Mask: insert 0x10

Shift Right: insert 4



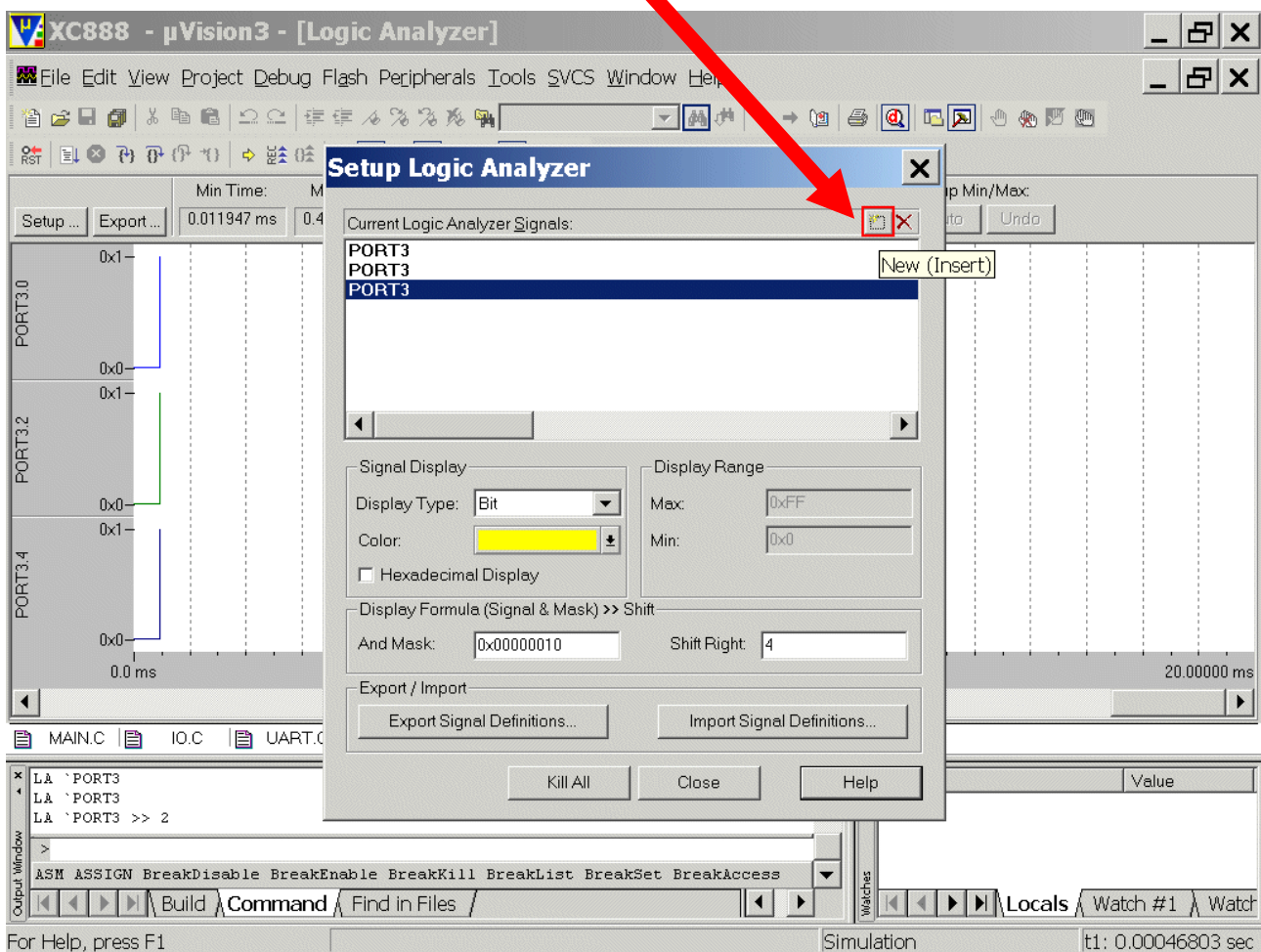
Note:



Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75

Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

click



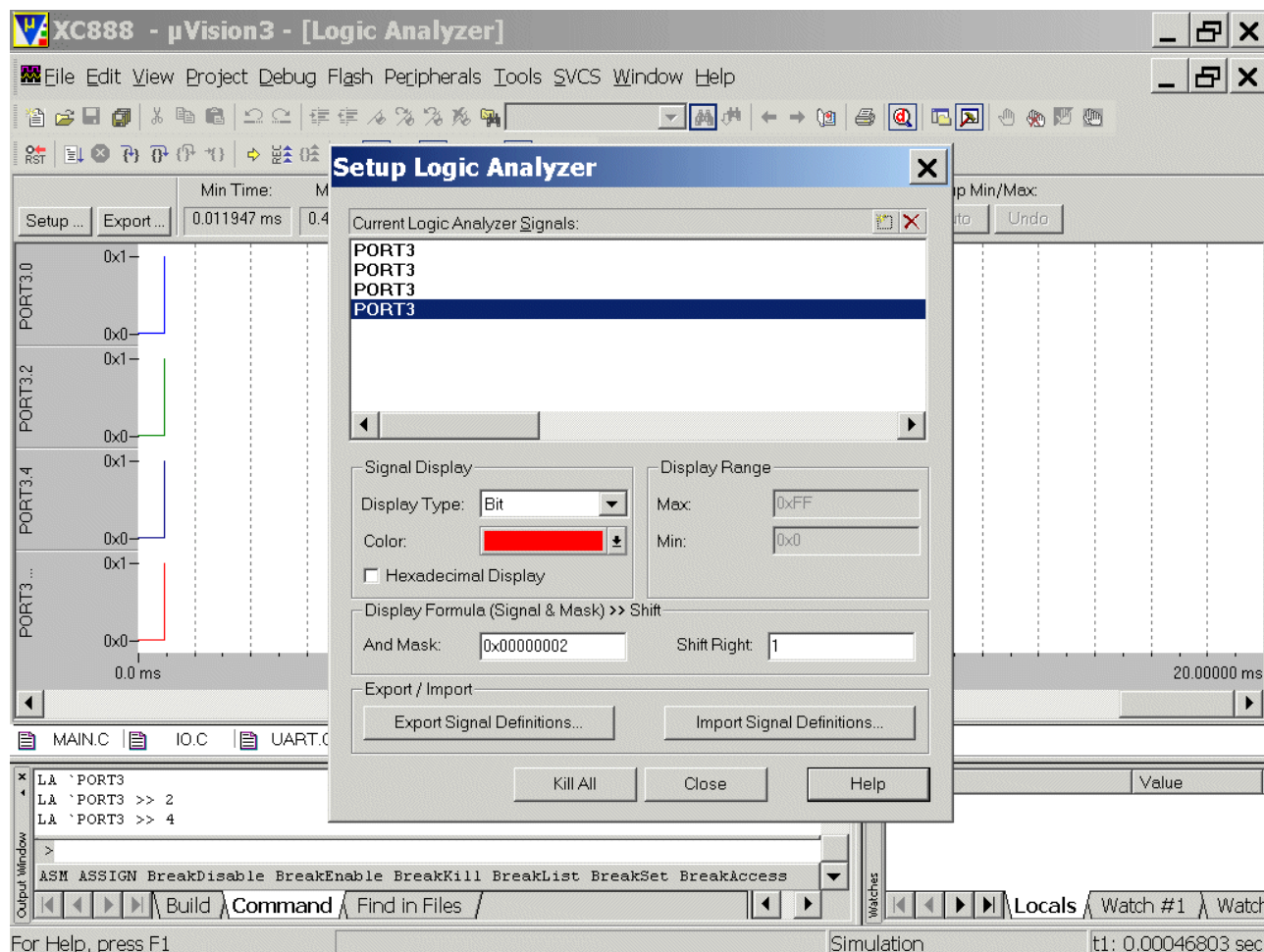
Insert PORT 3 <RETURN>

Display Type: select Bit

Color: select "red"

And Mask: insert 0x2

Shift Right: insert 1



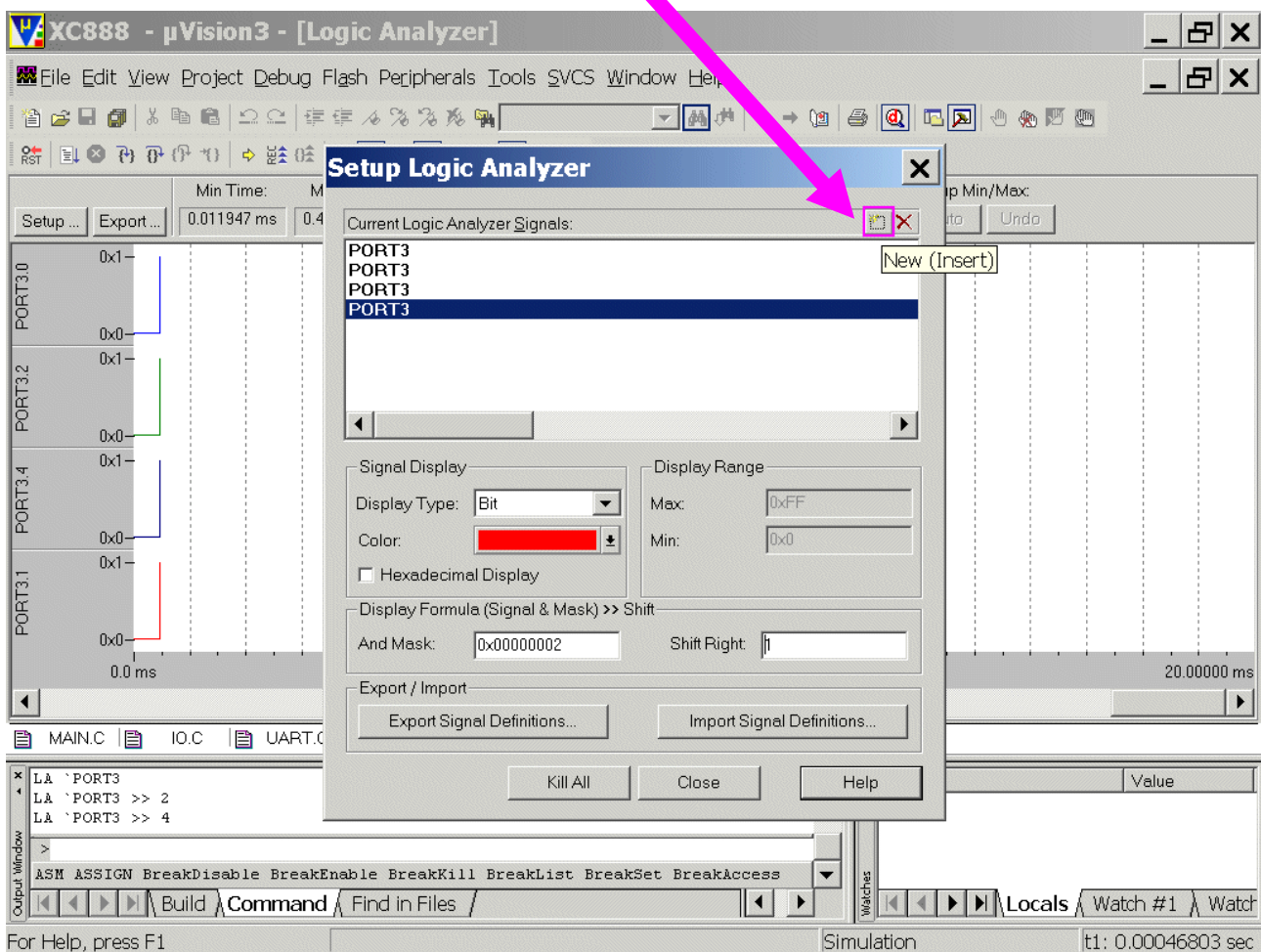
Note:

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75



Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.6	„use: program running signal“	Toggled via Timer_0 ISR

click



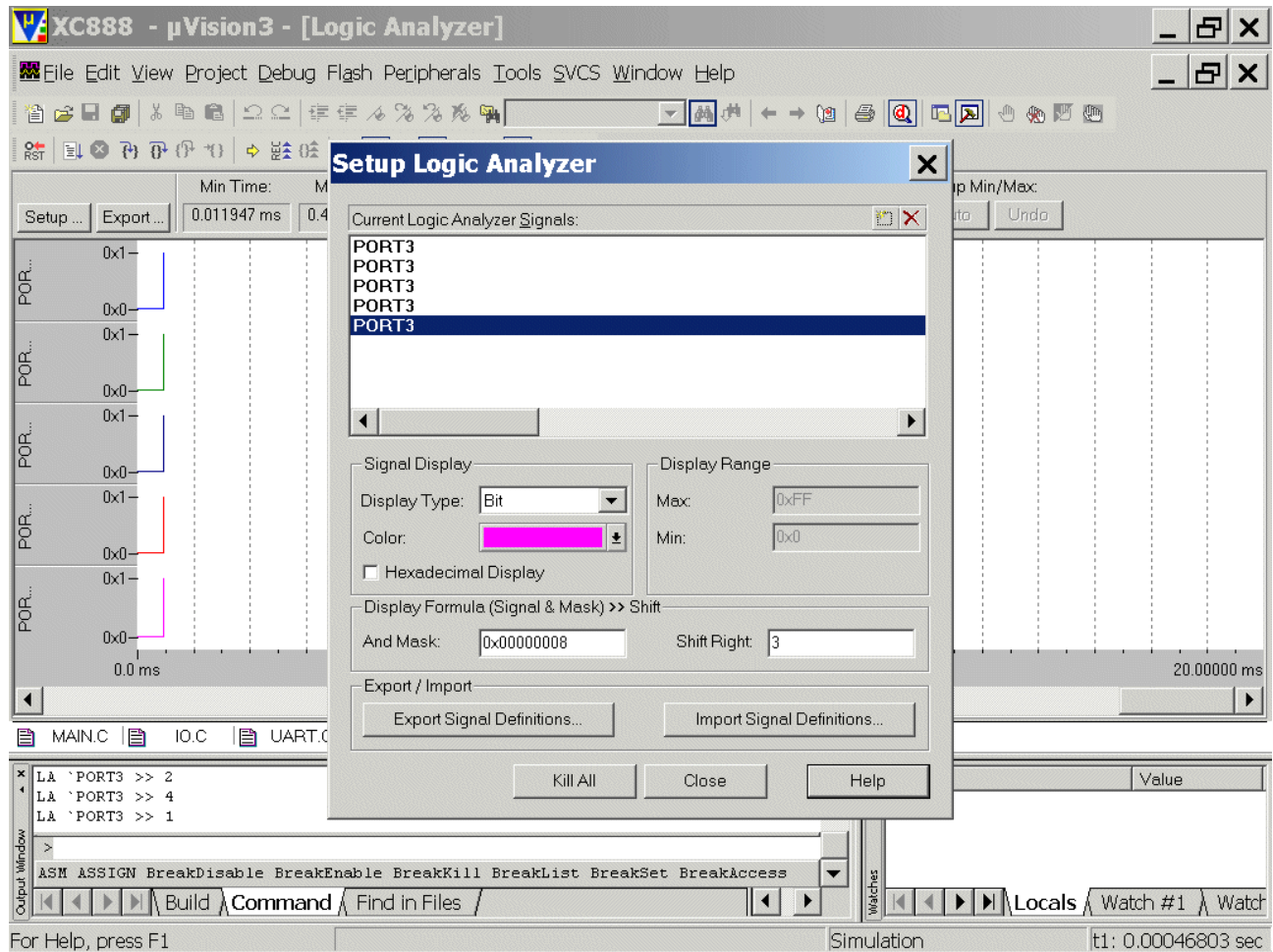
Insert PORT 3 <RETURN>

Display Type: select Bit

Color: select "magenta"

And Mask: insert 0x8

Shift Right: insert 3



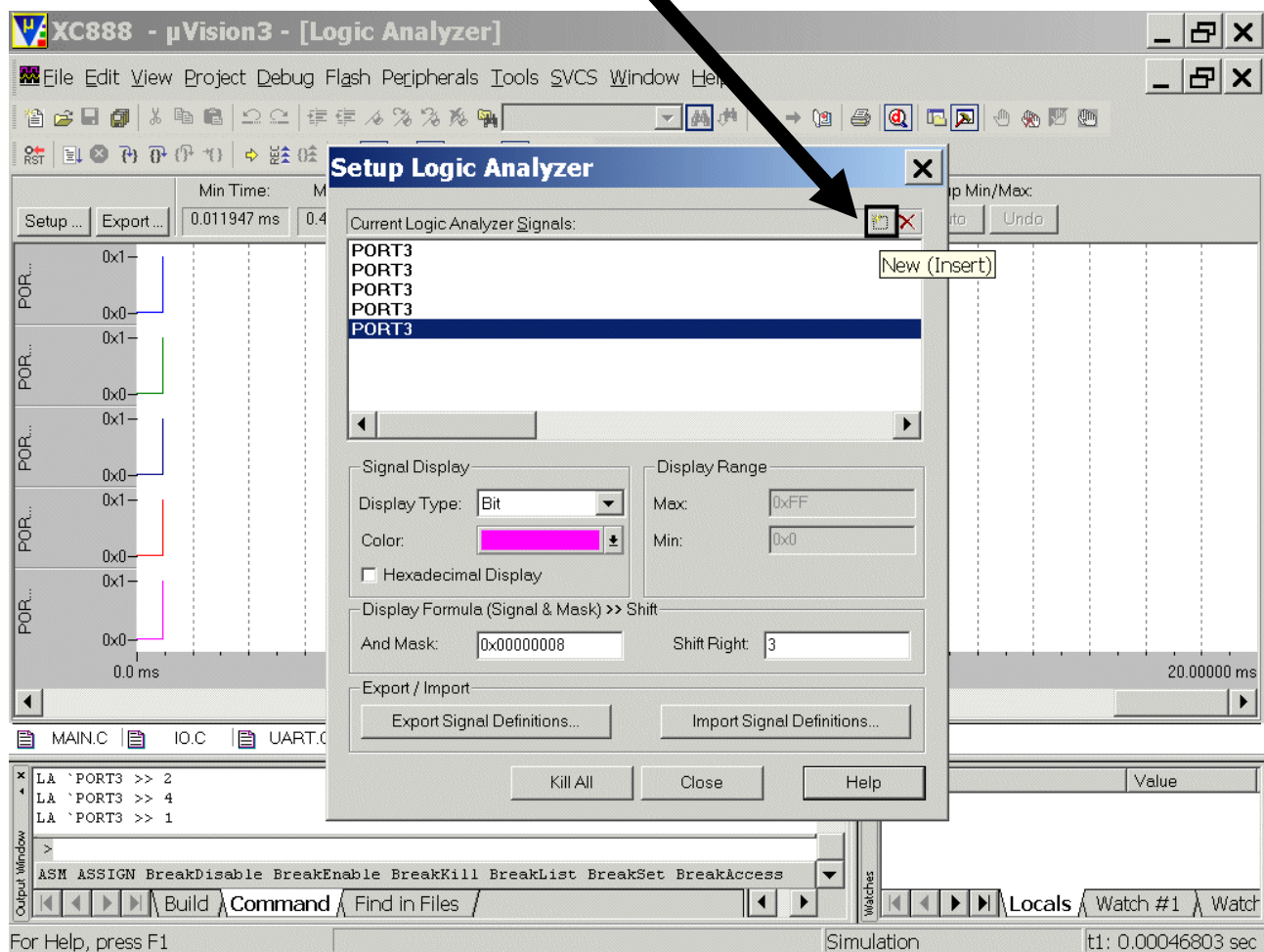
Note:

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75

Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR
P3.6	„use: program running signal“	Toggled via Timer_0 ISR



click



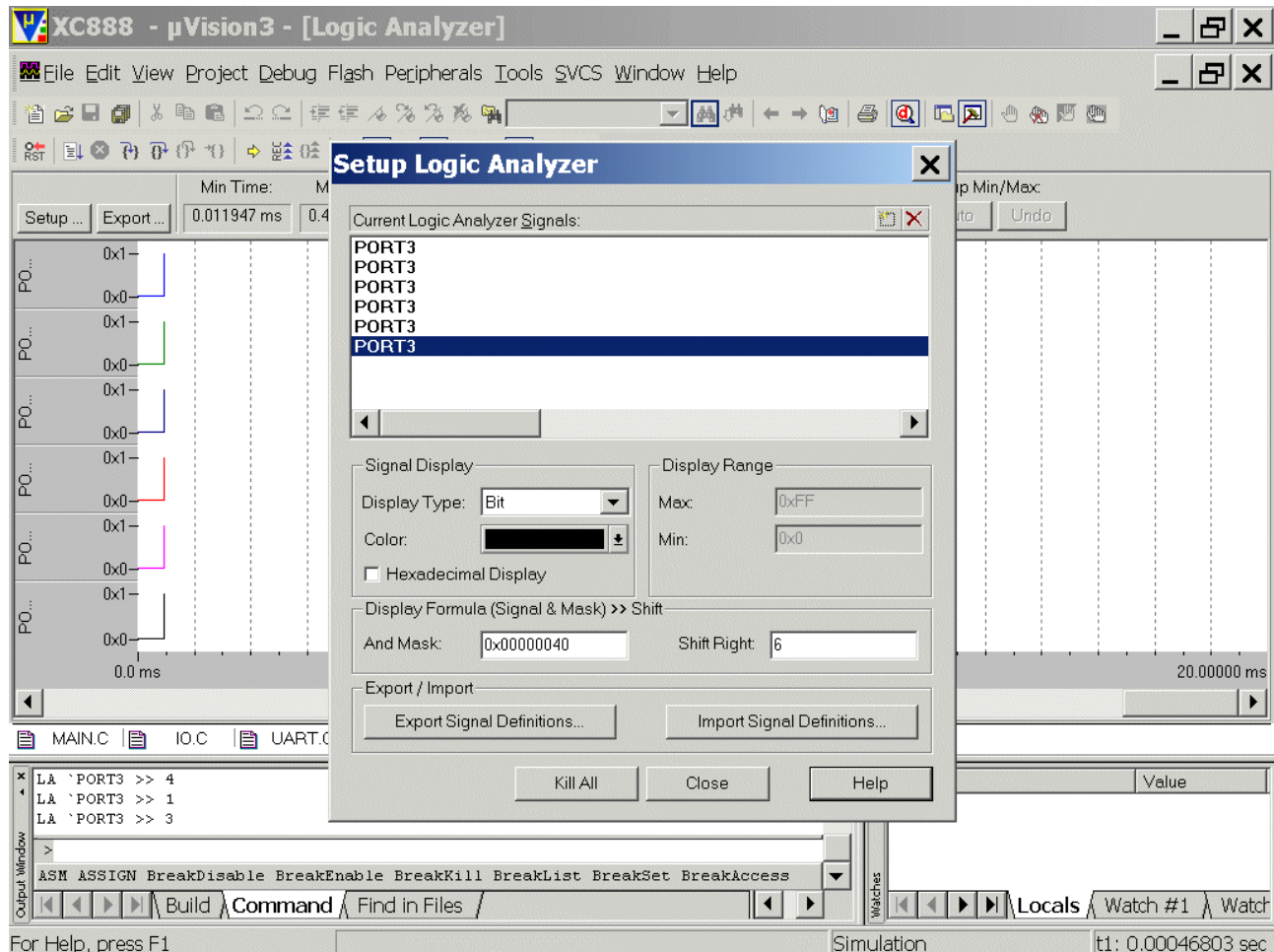
Insert PORT 3 <RETURN>

Display Type: **select** Bit

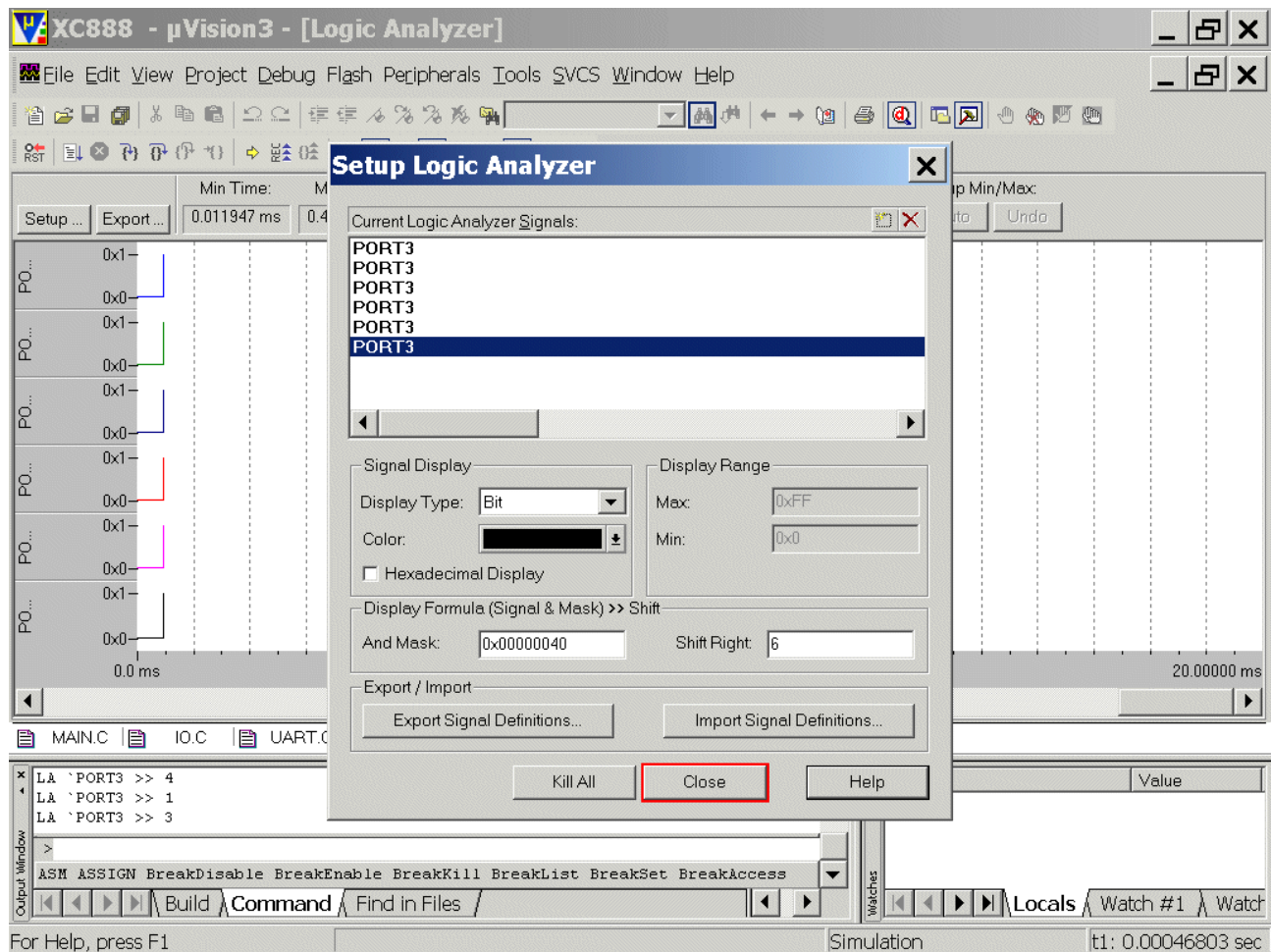
Color: **select** "black"

And Mask: **insert** 0x40

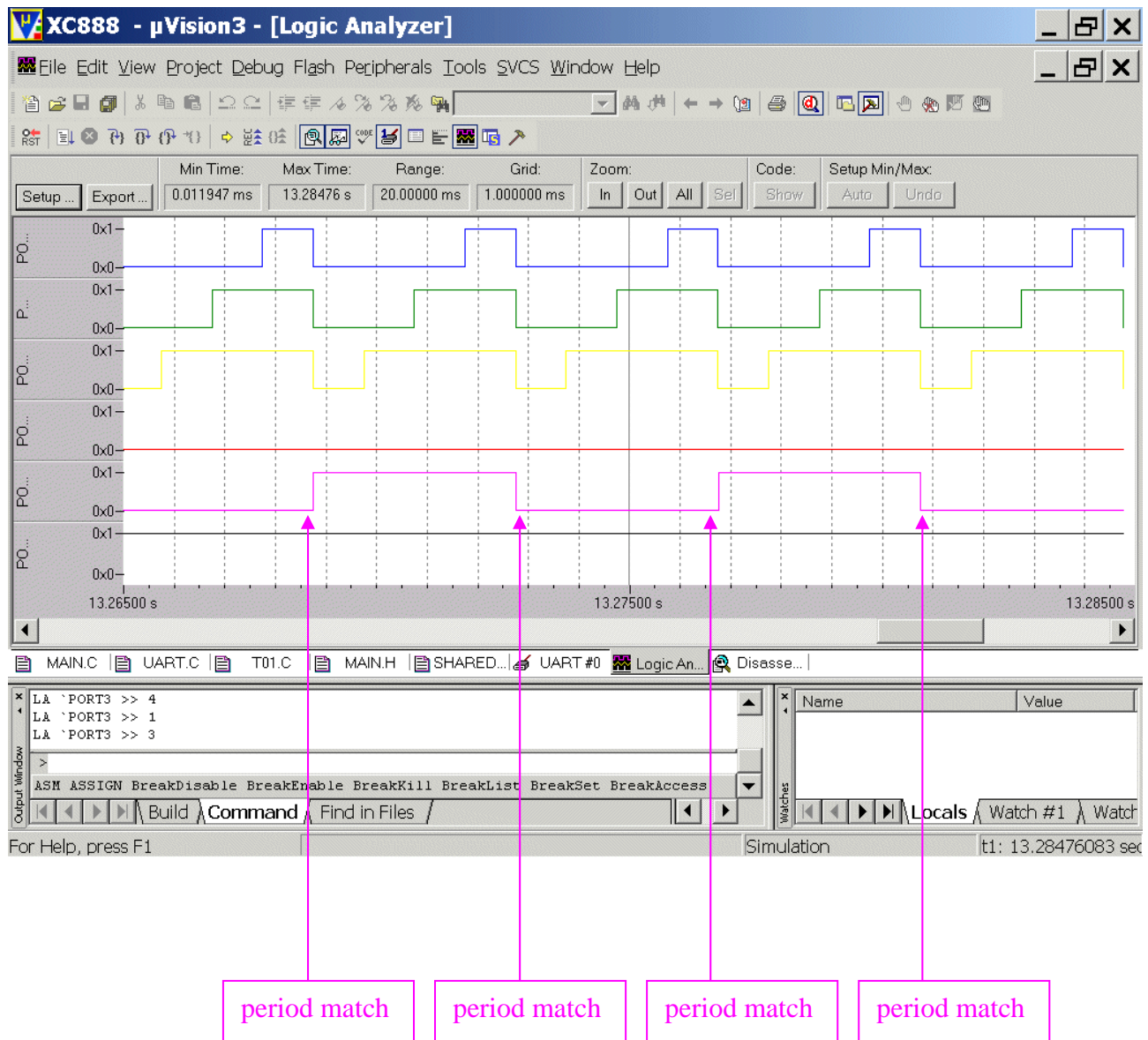
Shift Right: **insert** 6



Click Close:



Debug – Run

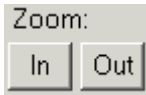


Note:

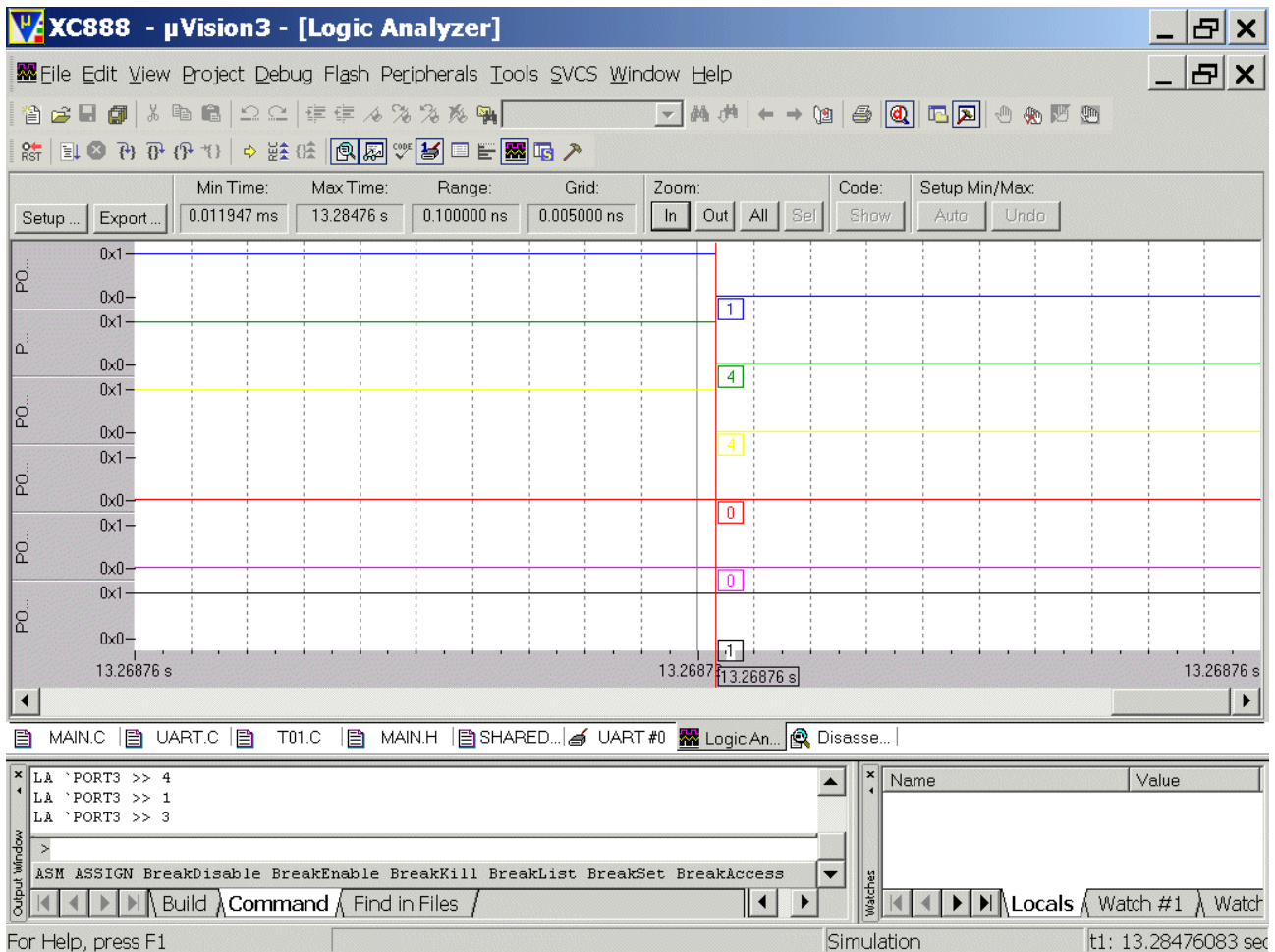
There is only a **period match** in Asymmetrical / Edge-Aligned PWM mode.



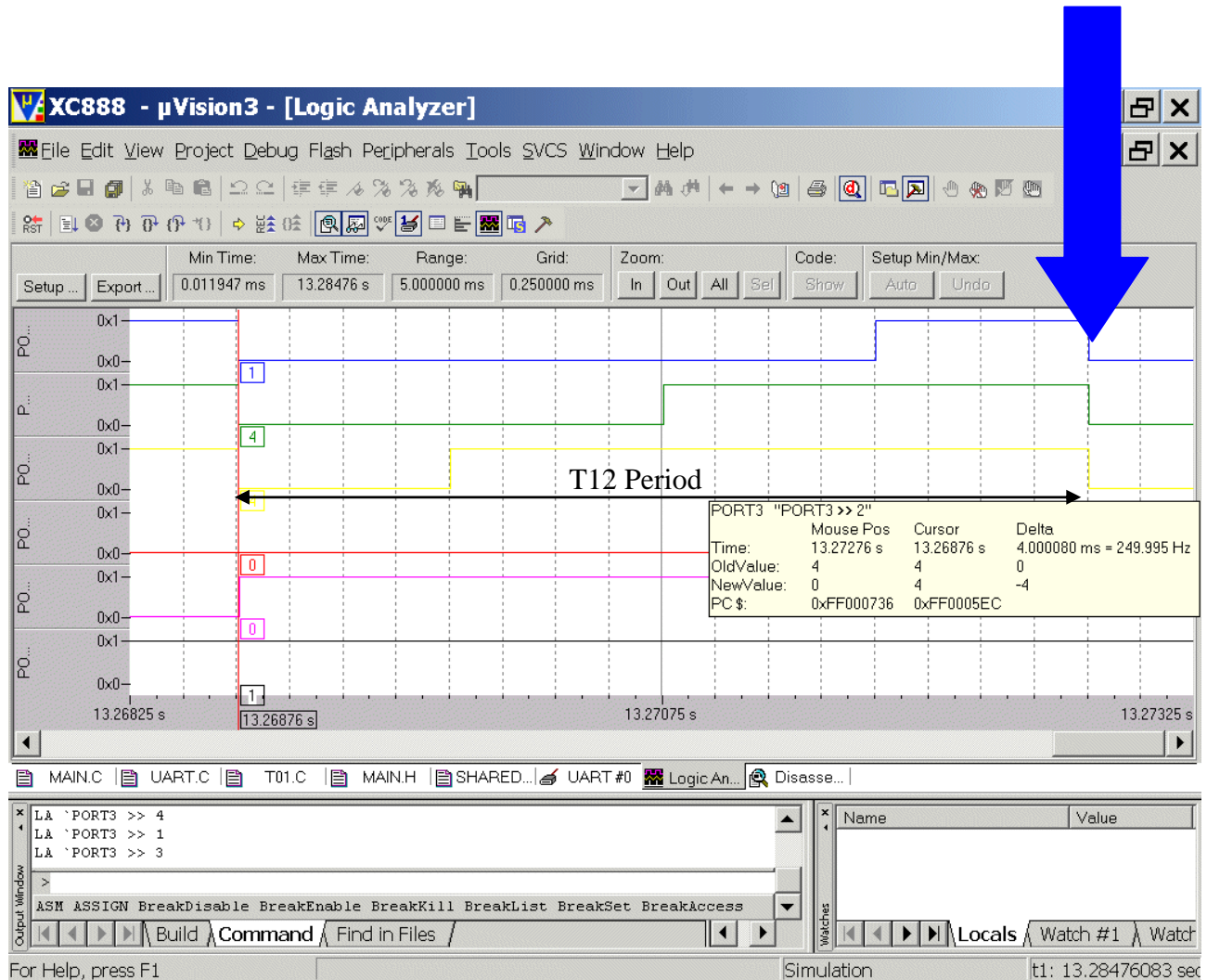
Now we could measure the T12 Period:



Use **In** **Out** to position the cursor in the LGA window exactly.
And **click** the left mouse button to fix the "measurement line":



To measure a time-difference **move** the **mouse cursor slowly** and see the timing information window appear:



Timer 12 Period Register		Timer 12 Period Register	
Period [ms]	4.166750	Period [ms]	4.166750
Period register (T12PRH, T12PRL)	0xC350	Period register (T12PRH, T12PRL)	0xC350

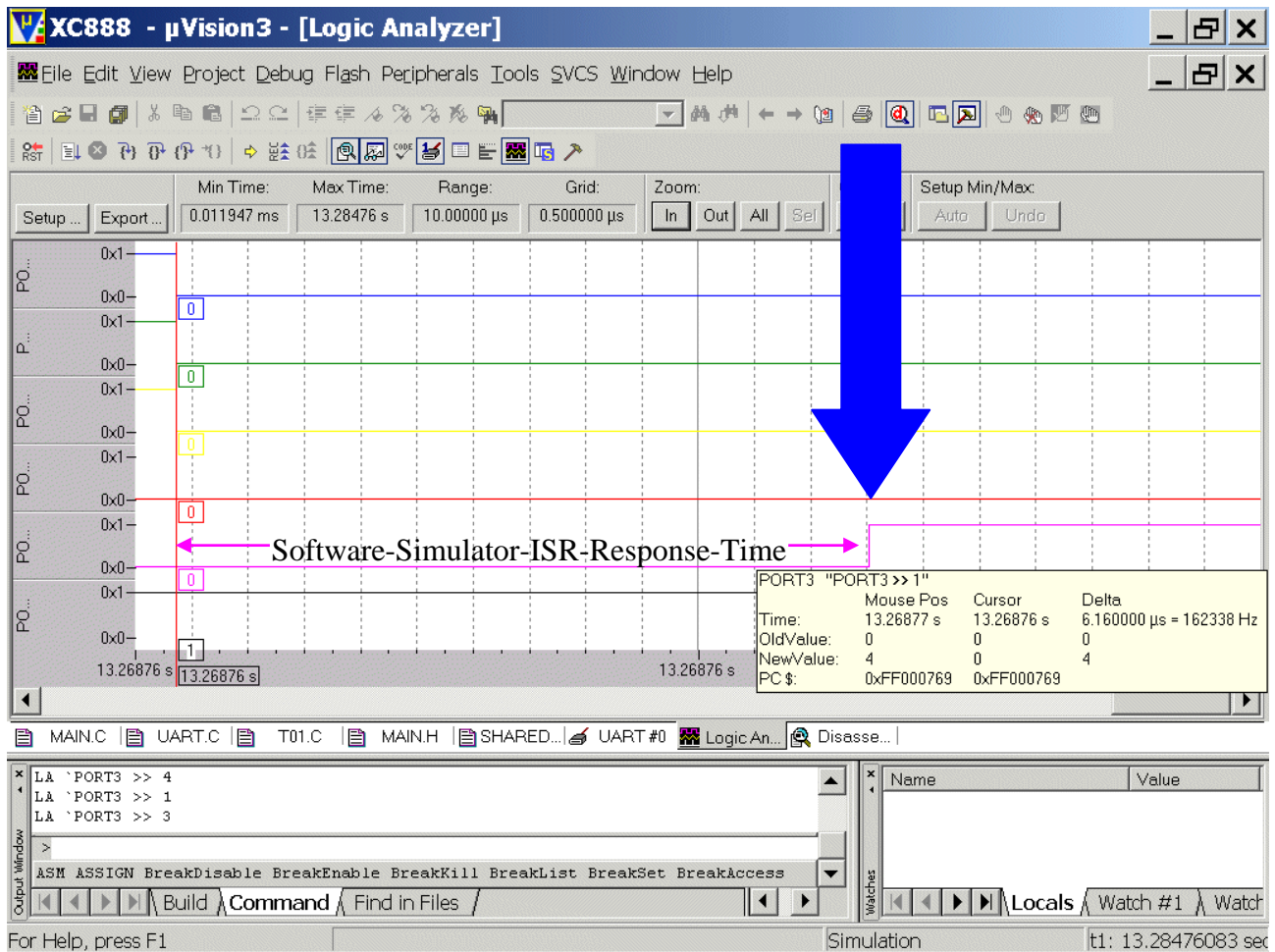
T12 Period measurement results:

	T12 Period [ms]	comment
µ Vision Software-Simulator-LGA	4,000080	
DAvE - Configuration	4,166750	
LGA	4,160000	measurement see page 94

Now we could measure the ISR-Response-Time:

Note:

Because **Port_3_Pin_3** is toggled via Software (ISR: `void SHINT_viXINTR12Isr(void) interrupt XINTR12INT {}`, file: `SHARED_INT.C`) in the corresponding ISR (**period match**) there must be a delay [ISR-Response-Time = **6,16** μ s (Simulator), **4,94** μ s (LGA)]:



ISR Response Times:

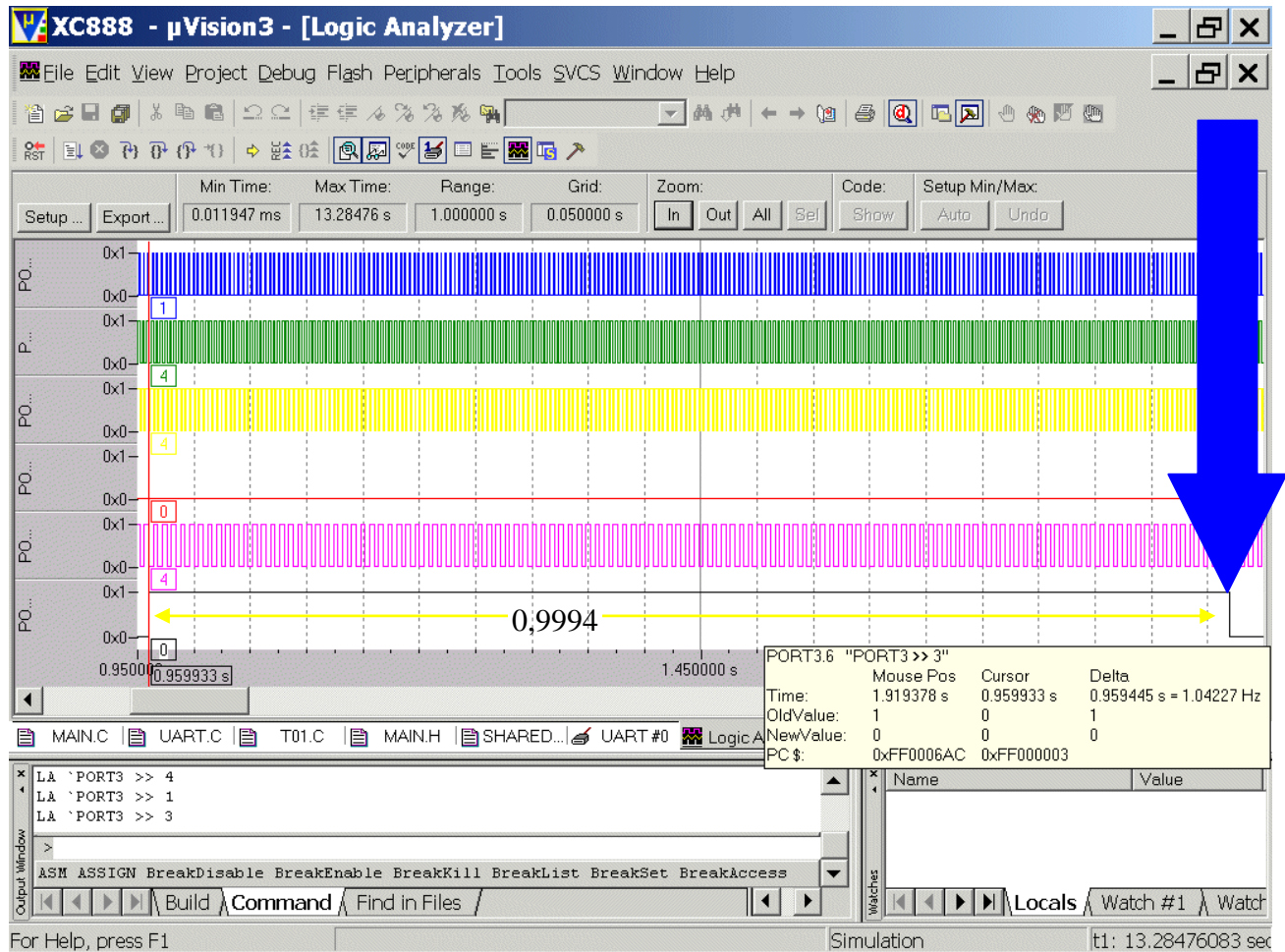
	ISR Response Time [μ s]	comment
µVision Software-Simulator-LGA	6 , 16	
LGA	4 , 94	measurement see page 95

Now we could measure the blinking frequency:

Note:

Port_3_Pin_6 is toggled via Software in the corresponding ISR (Timer_0).

$(183 * \text{Timer_0-overflow} = 183 * 5461,333 \mu\text{s} = 0,9994 \text{ s})$.



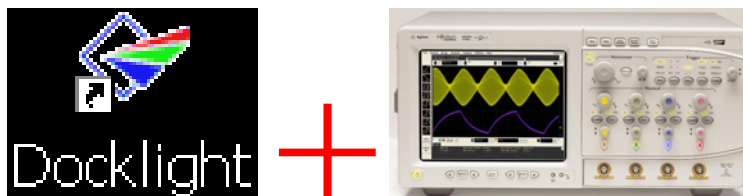
0,9994 s:

	0,9994 [s]	comment
µVision Software-Simulator-LGA	0,959445	
LGA	0,999000	measurement see page 96

See the result:

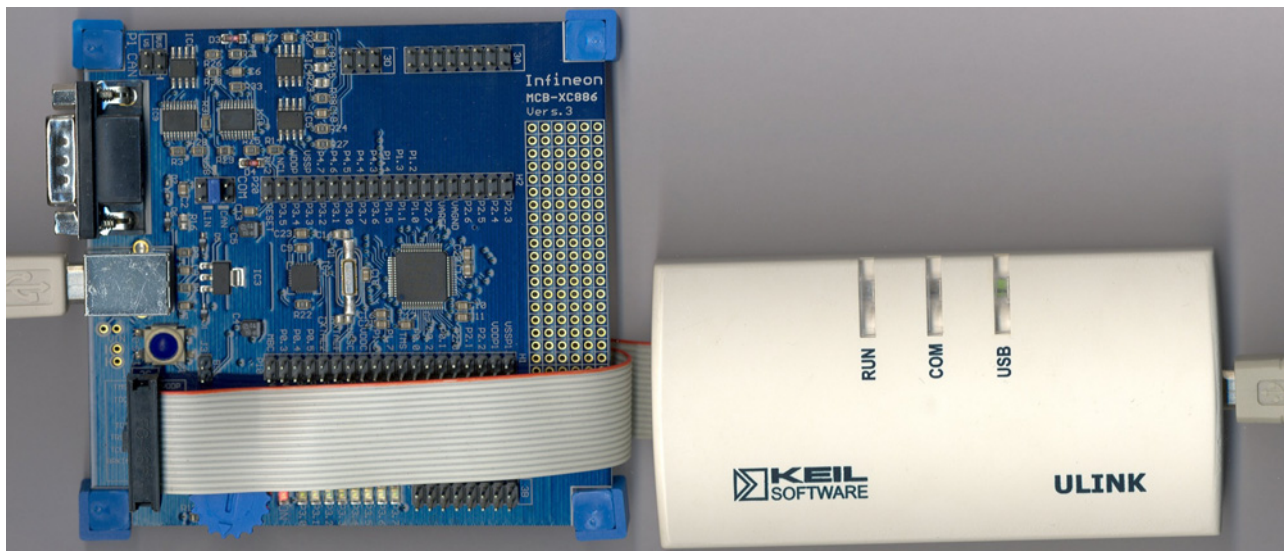
Using real hardware:

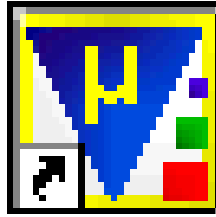
Any Terminal Program (e.g. Docklight) + Any Logic Analyser



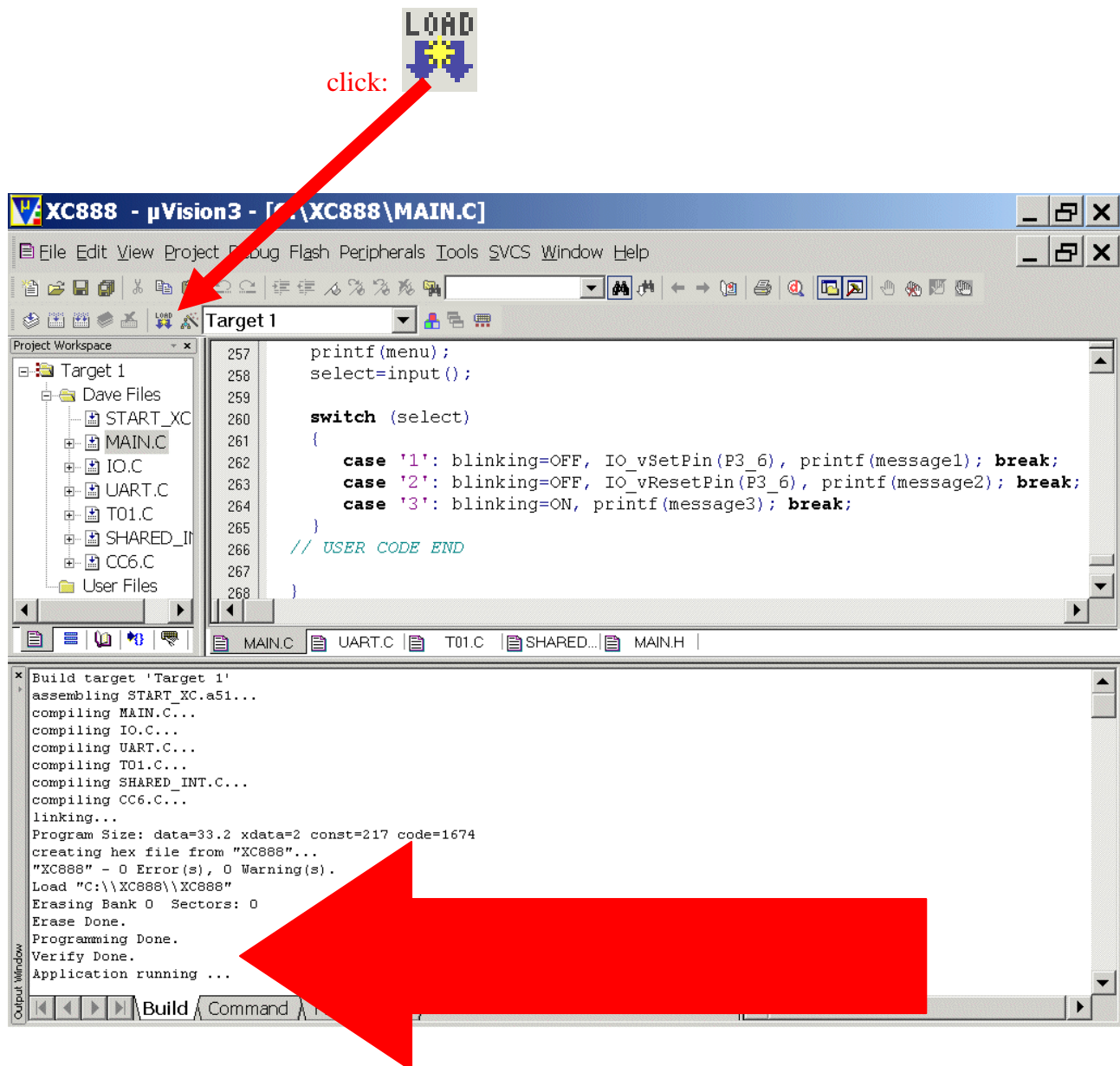
Connect your ULINK (used for: program download),

Connect your serial interface UART/RS232 via USB (used for: serial communication):



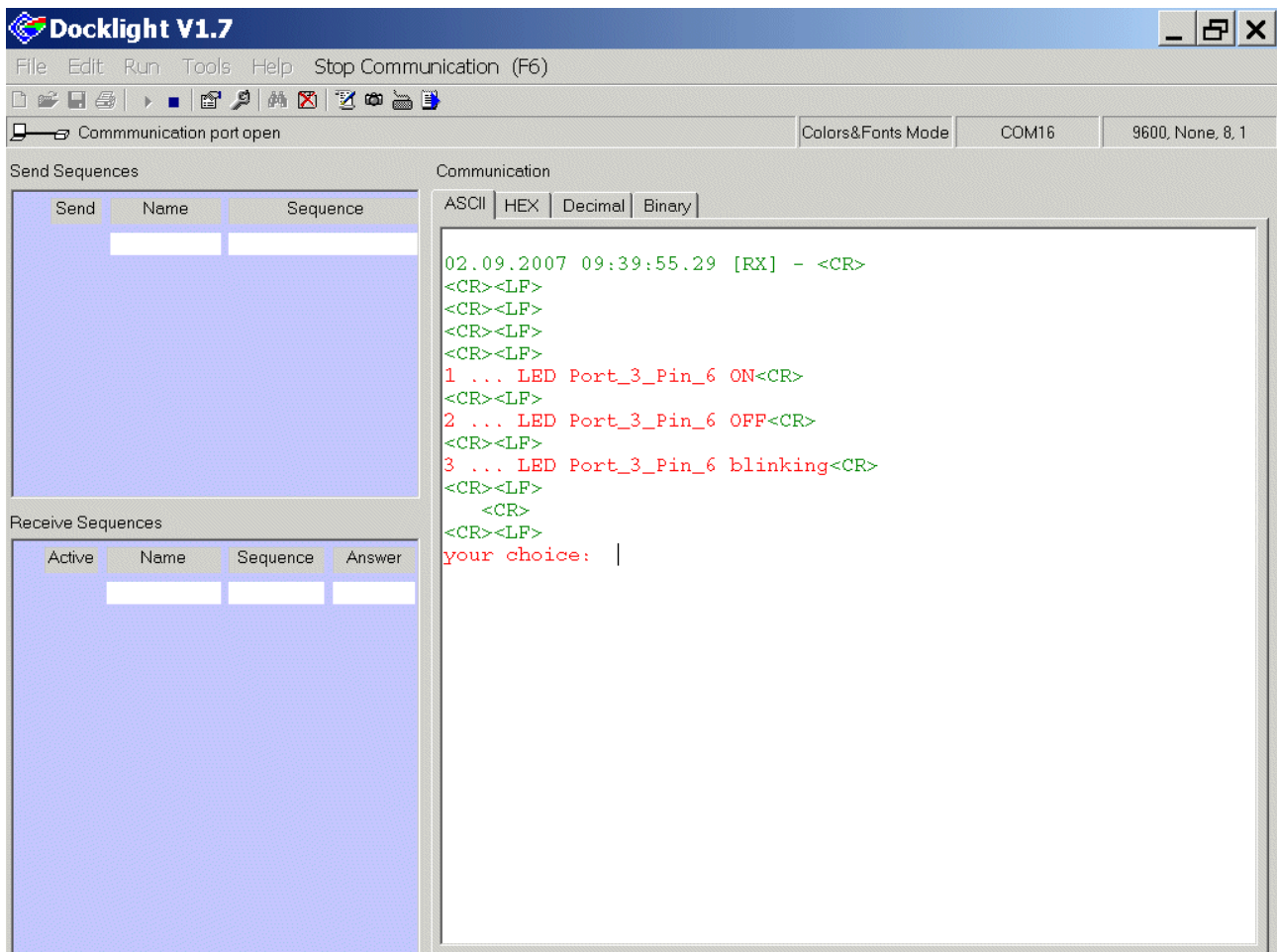


Go to µVision, Download/Program our application program into the On-Chip-Flash:

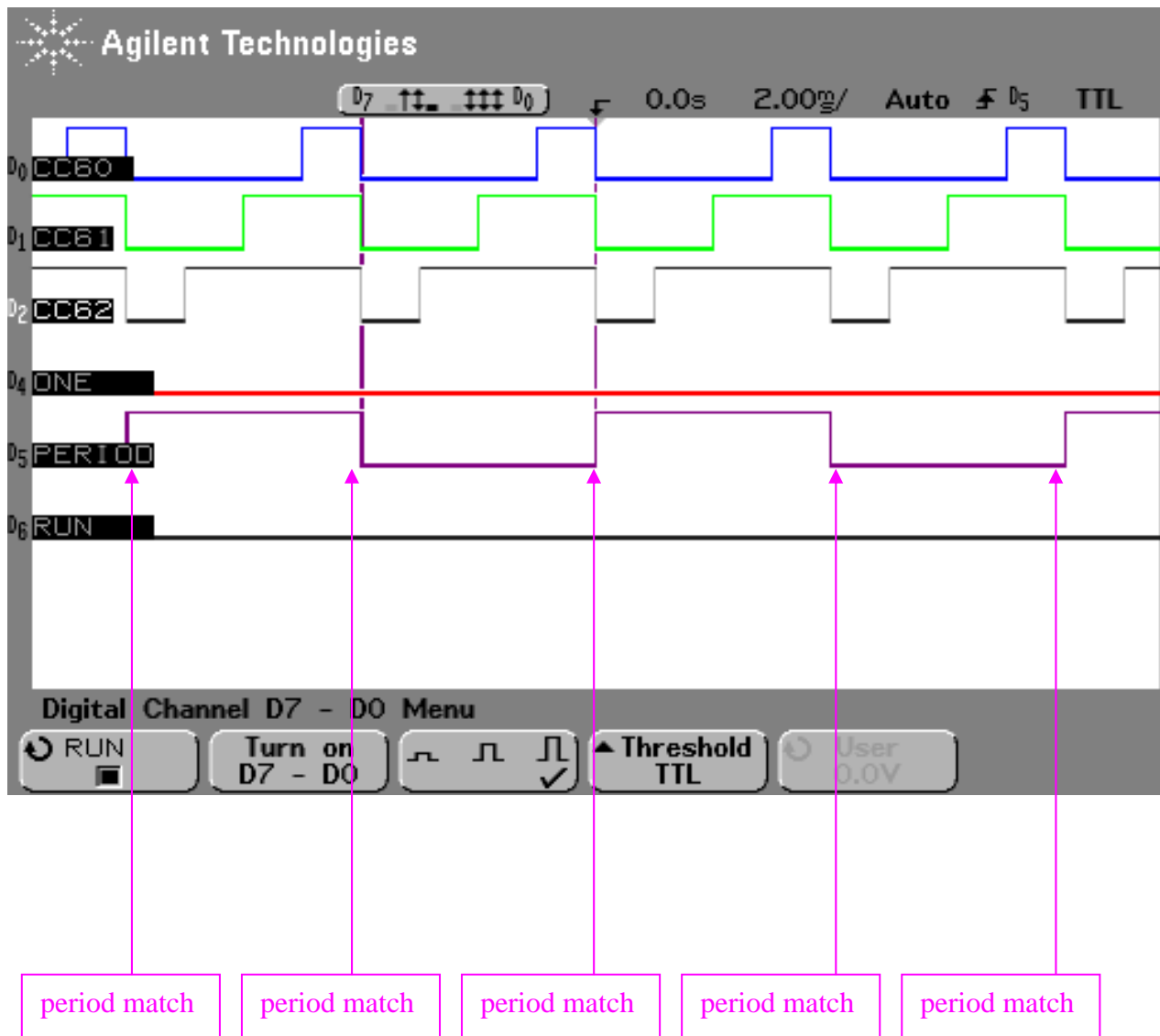




Go to Docklight and see the result:



Use any logic analyser and see the result:

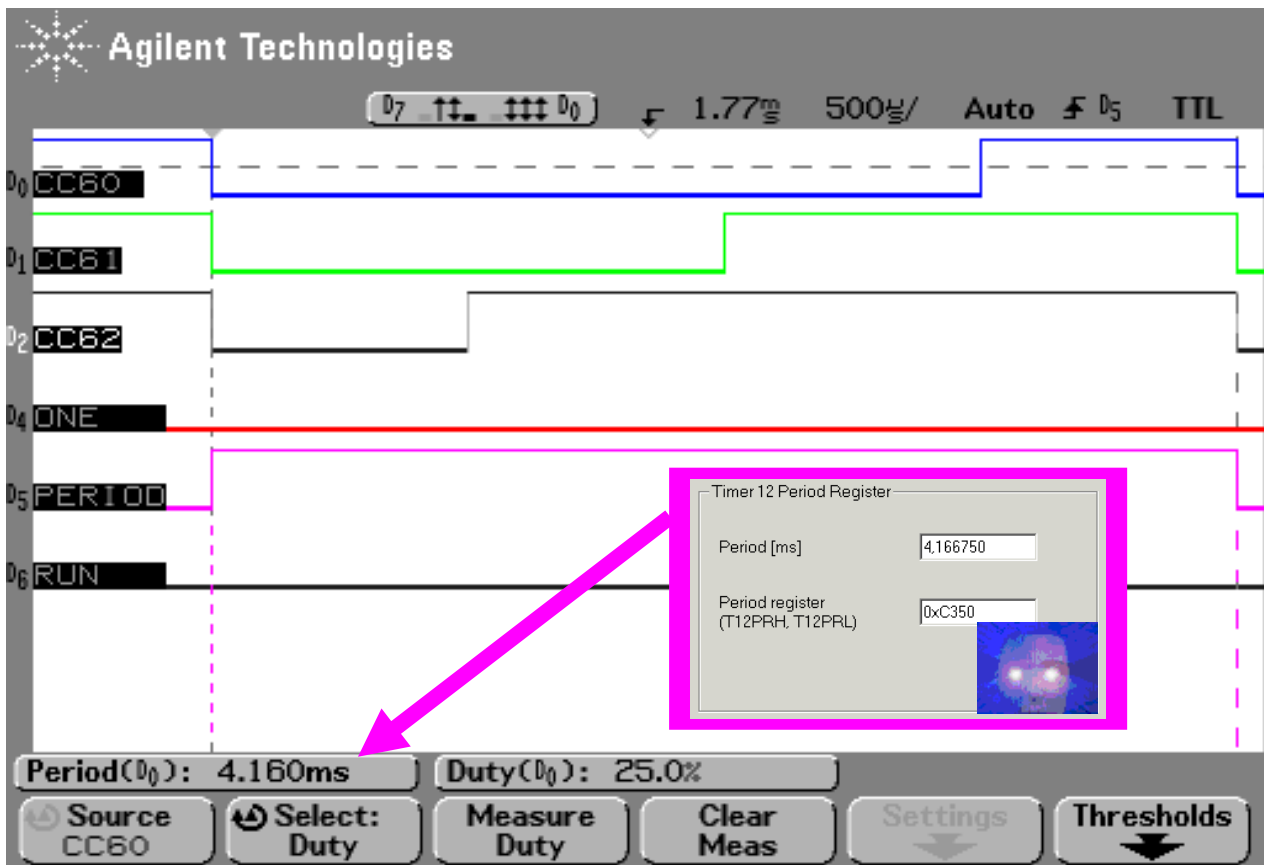


Note:

There are only **period matches** in Asymmetrical / Edge-Aligned PWM mode.



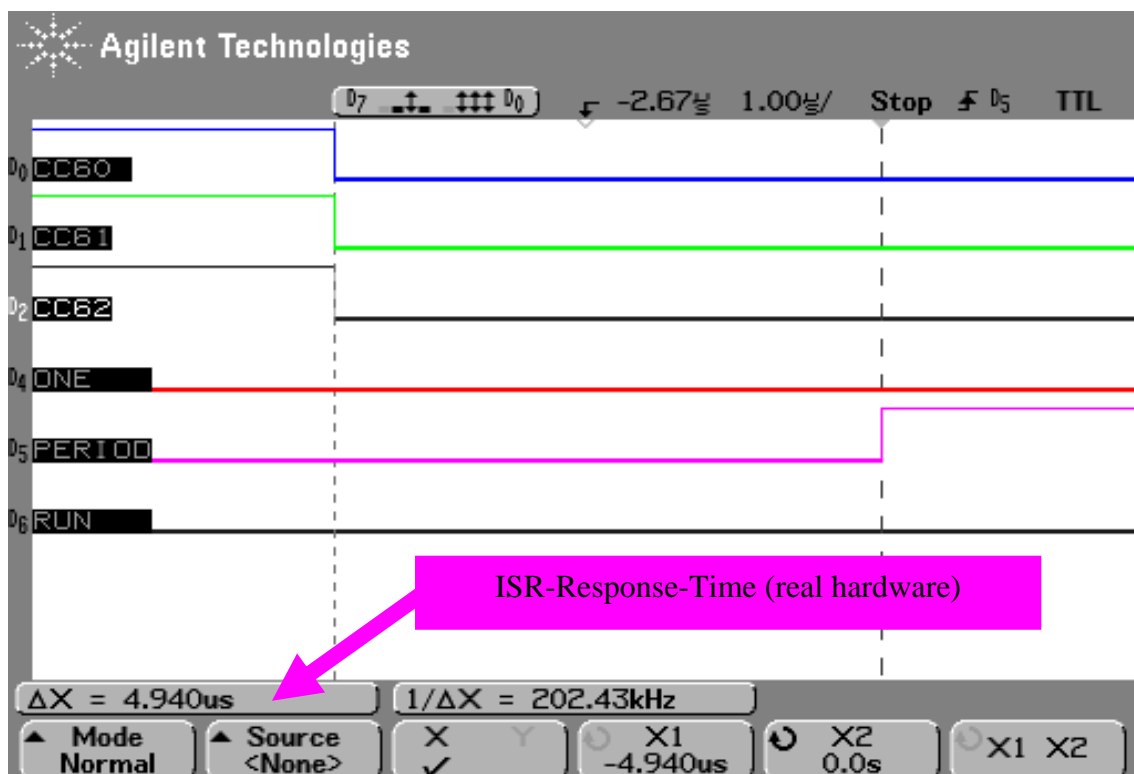
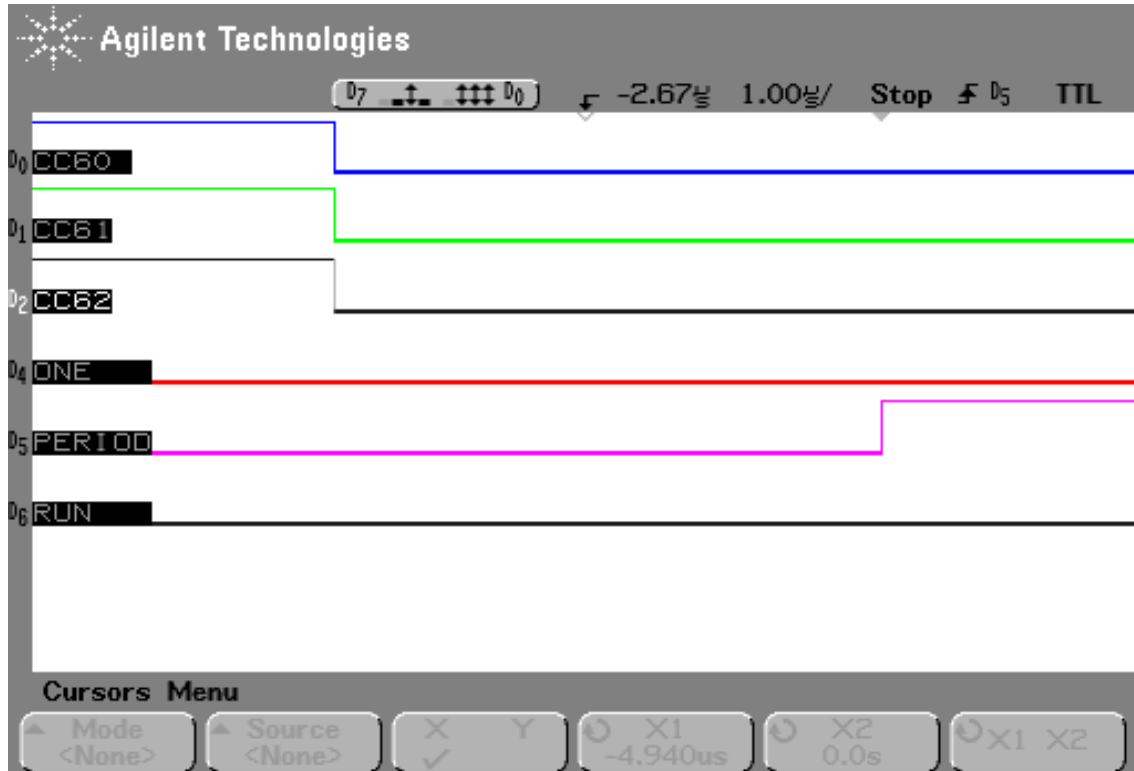
T12 Period measurement:



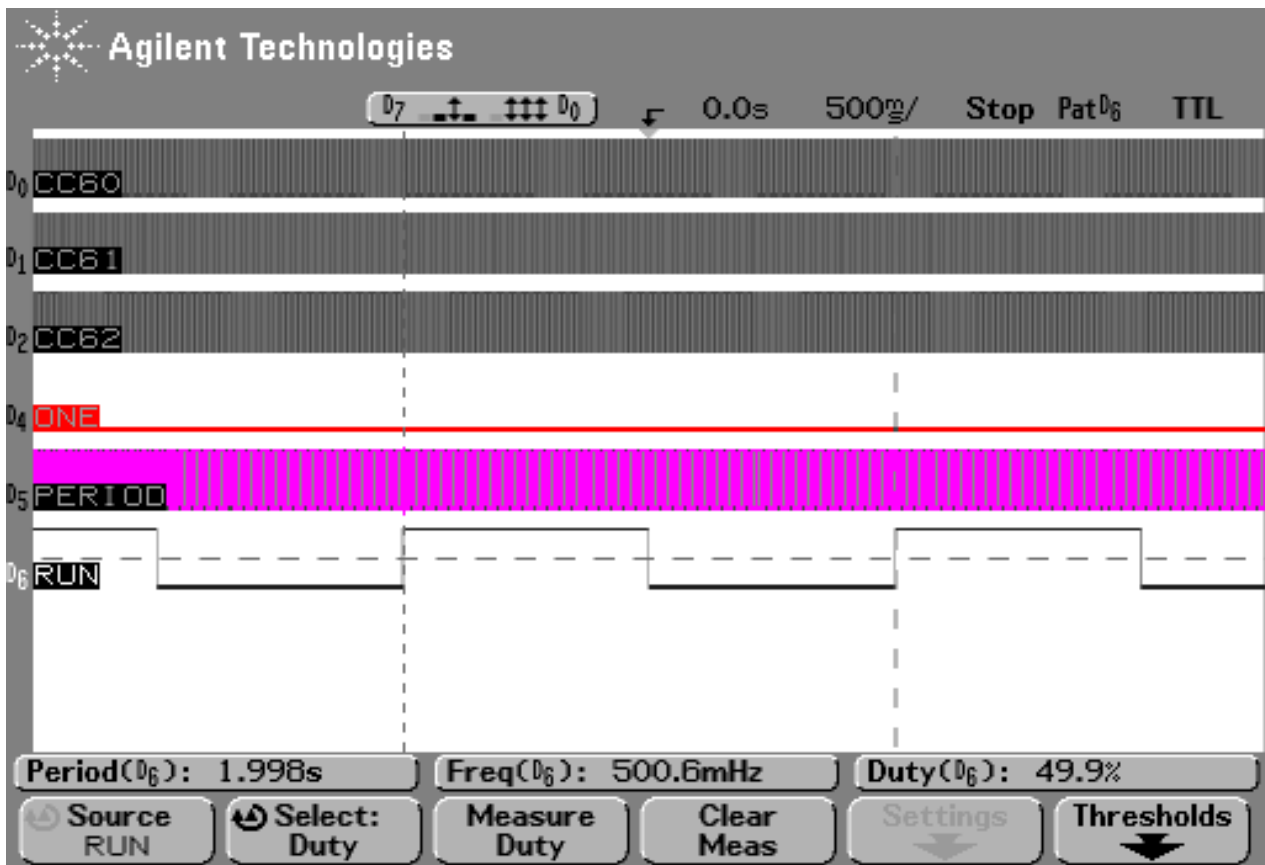
ISR-Response-Time measurment:

Note:

Because Port_3_Pin_3 is toggled via Software in the corresponding ISR (**period match**) there must be a delay [ISR-Response-Time = **6,16** μs (Simulator), **4,940** μs (LGA)]:



Blinking frequency measurement:



Note:

$$1,998 \text{ s} / 2 = 0,999 \text{ s}$$



Comparison:

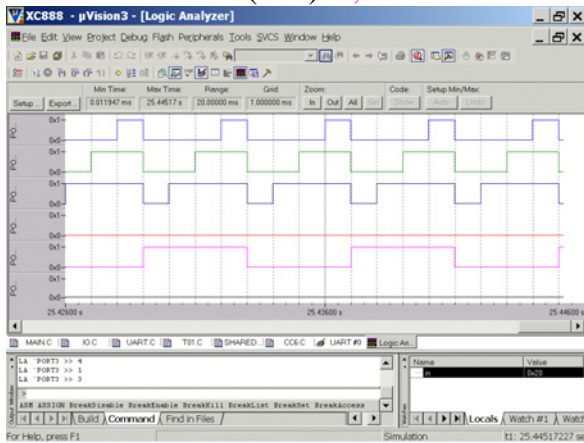
Software:

Hardware:

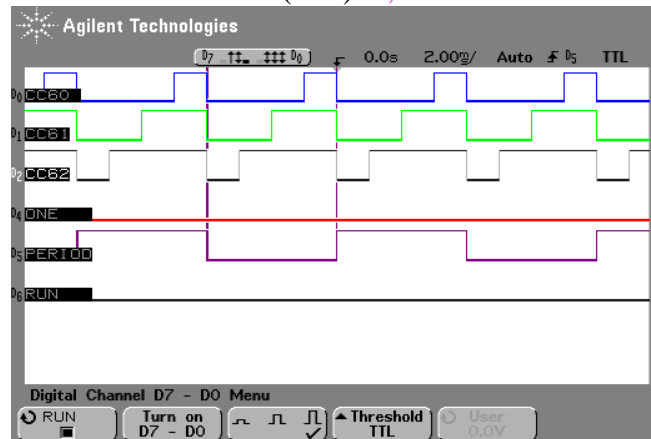
Keil μ Vision Simulator:

Logic Analyser (LGA):

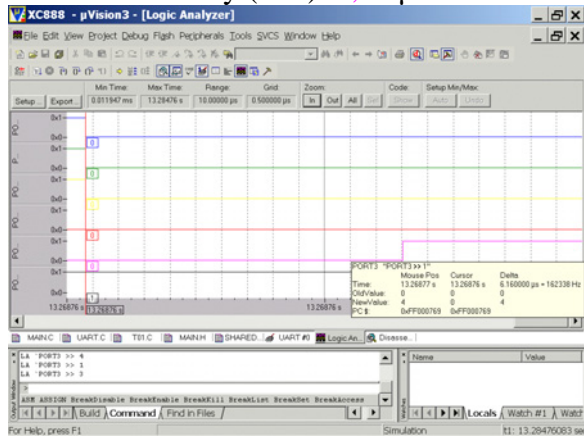
Period (T12): 4,00 ms



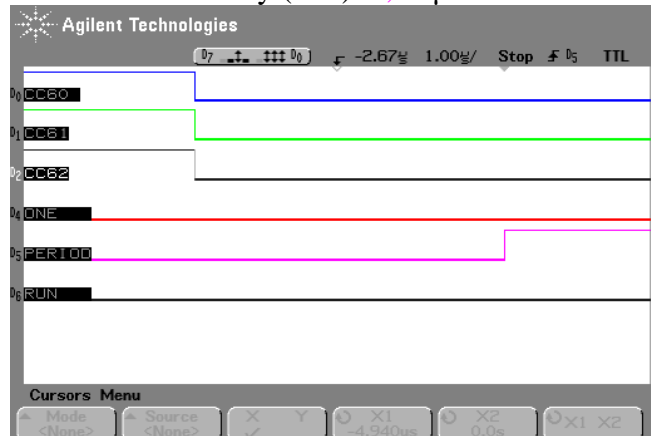
Period (T12): 4,16 ms



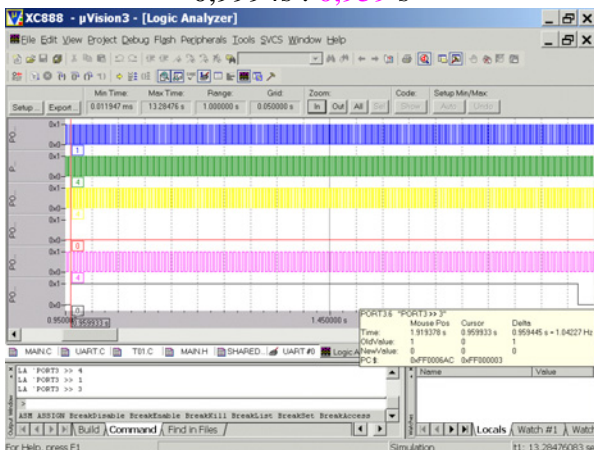
Delay (ISR): 6,16 μ s



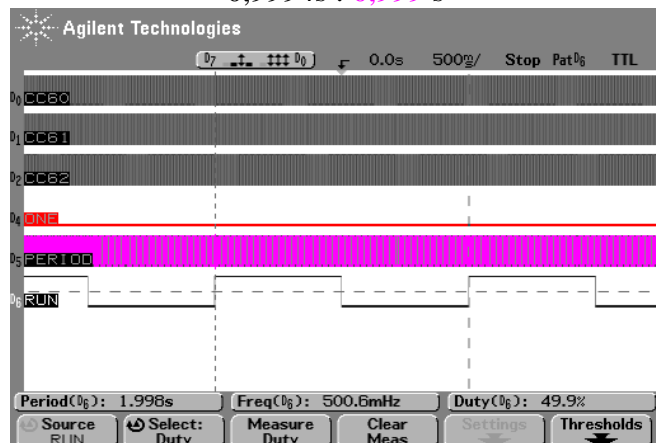
Delay (ISR): 4,94 μ s



0,9994s : 0,959 s



0,9994s : 0,999 s



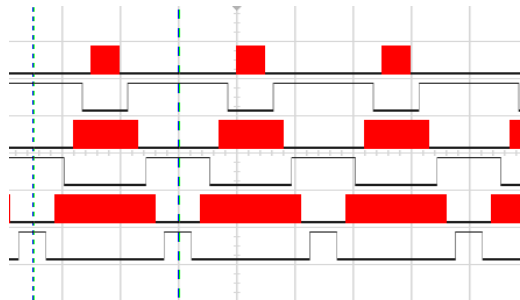


Note:



Asymmetrical / Edge-Aligned Mode (Bit CTM = 0):

The count direction is set to count up (CDIR = 0). The counter is reset to zero when a **period-match** is detected, and the T12 shadow register transfer takes place if STE12 = 1.

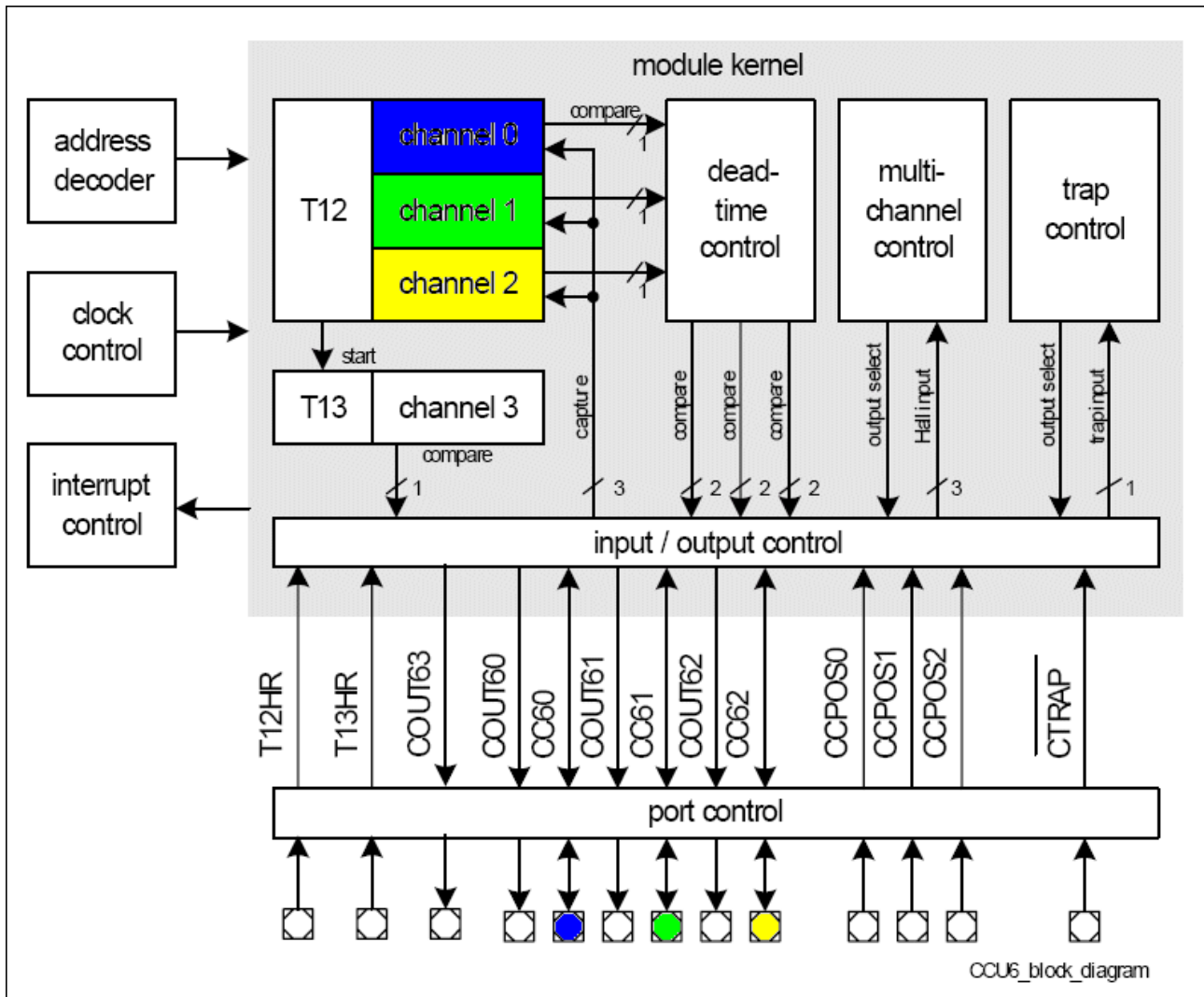


Symmetrical / Center-Aligned Mode (Bit CTM = 1):

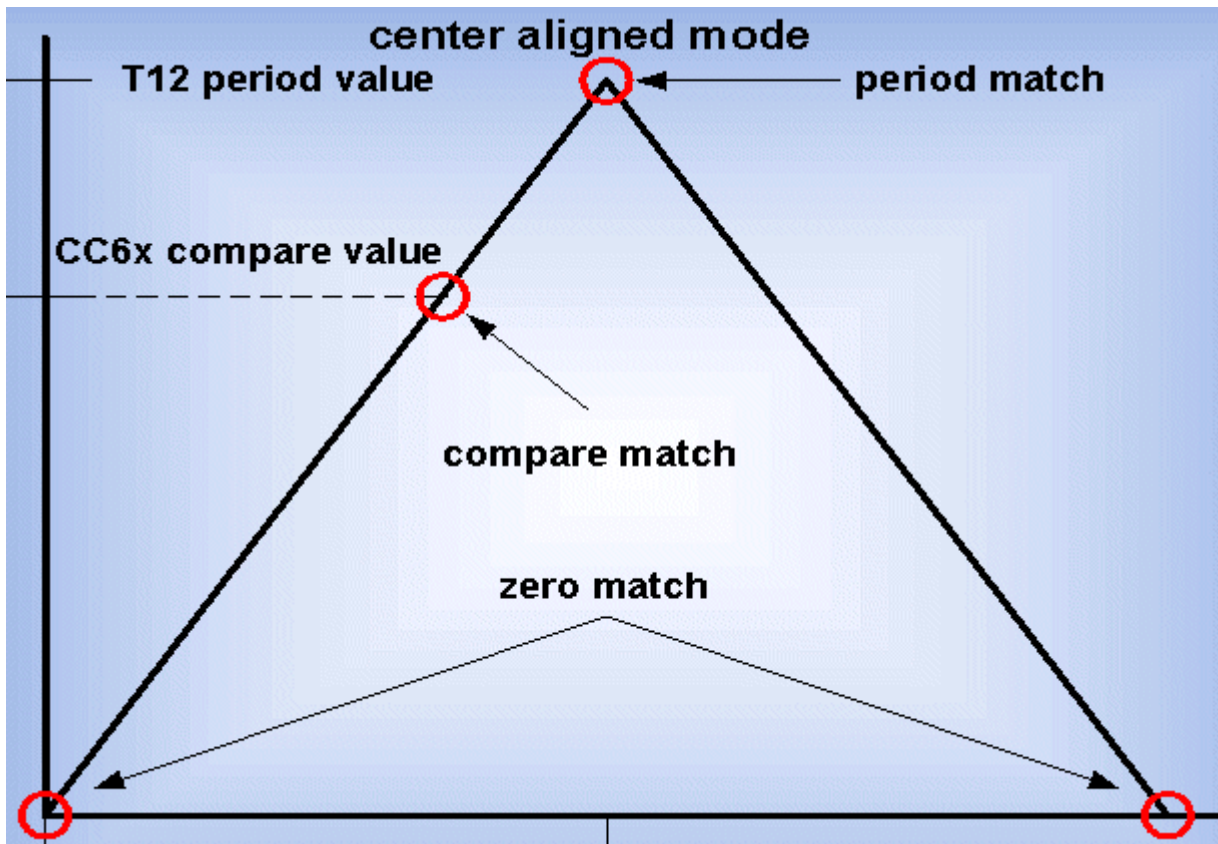
- The count direction is set to count up (CDIR = 0) when a **one-match** is detected while counting down.
- The count direction is set to count down (CDIR = 1) when a **period-match** is detected while counting up.
- If STE12 = 1, a shadow transfer takes place when:
 - a **period-match** is detected while counting up
 - a **one-match** is detected while counting down

2.) Symmetrical / Center-Aligned PWM generation using CCU6

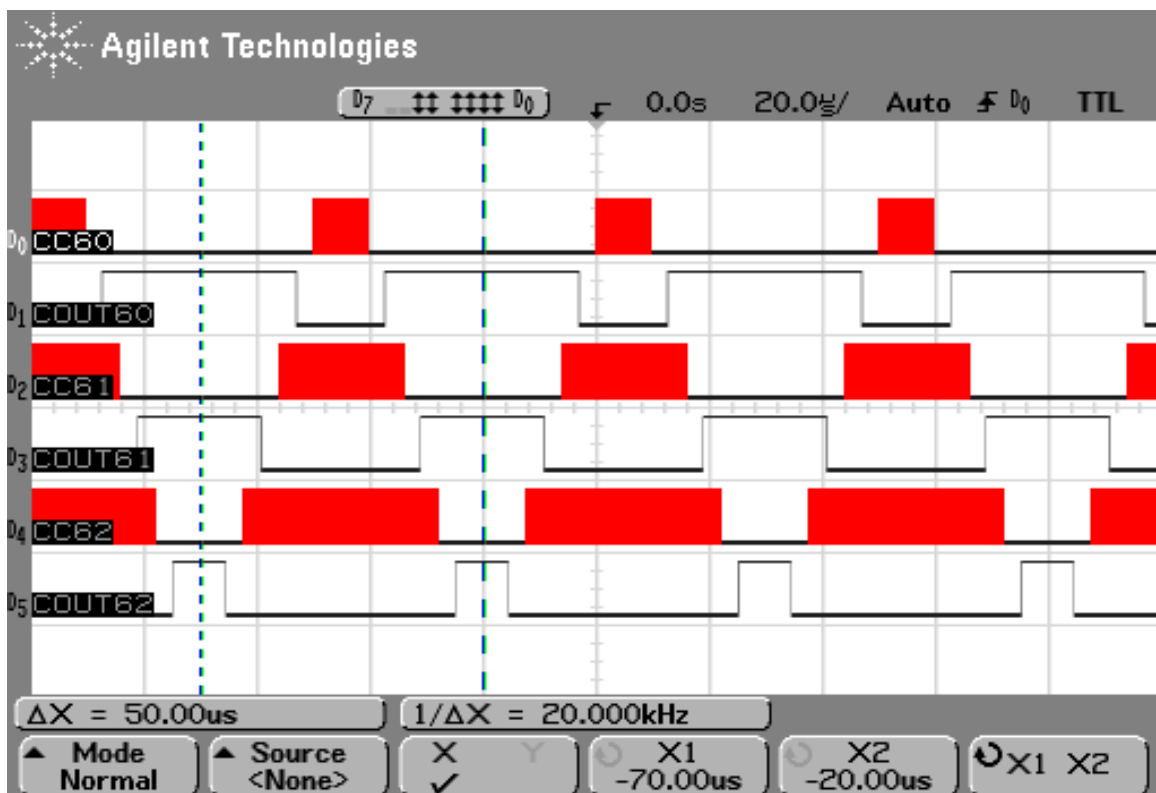
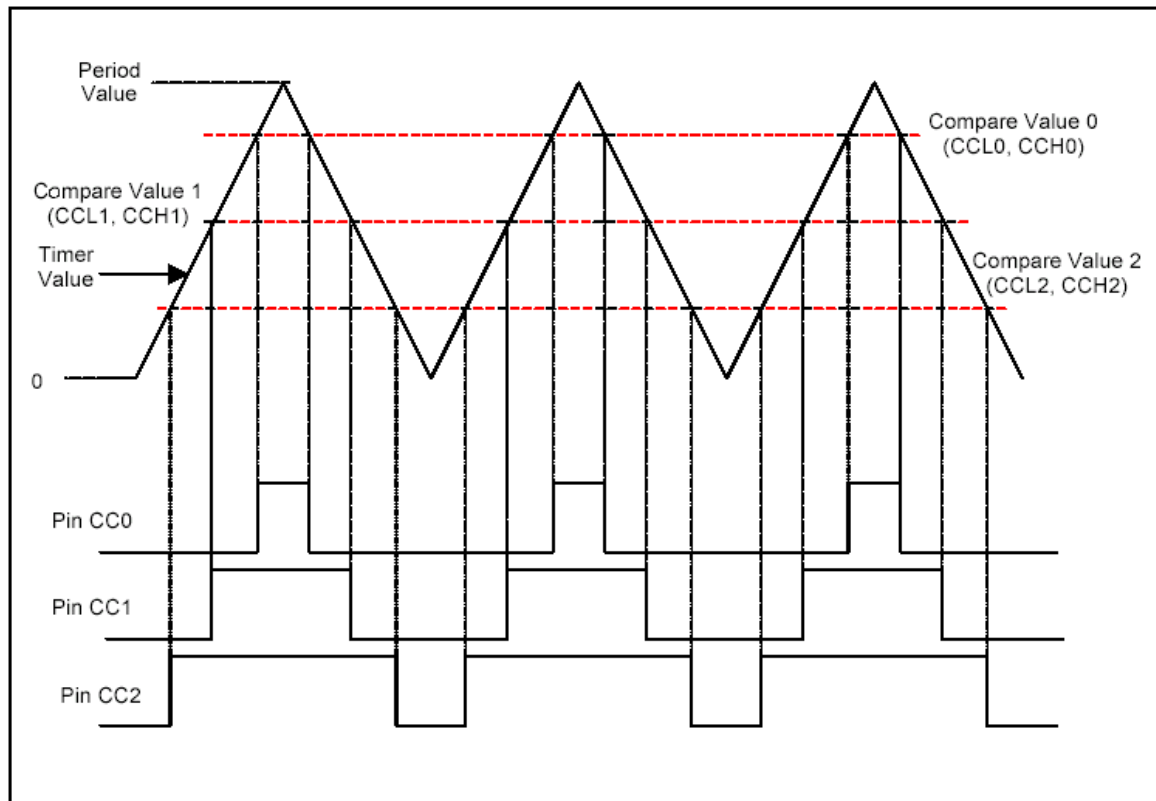
Used channels for PWM generation (blue, green, black/yellow):



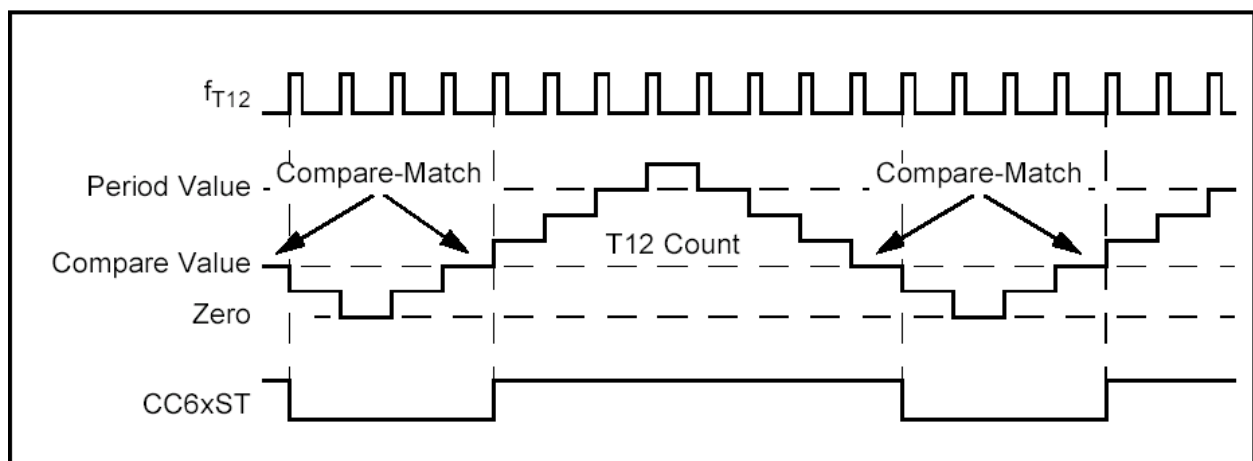
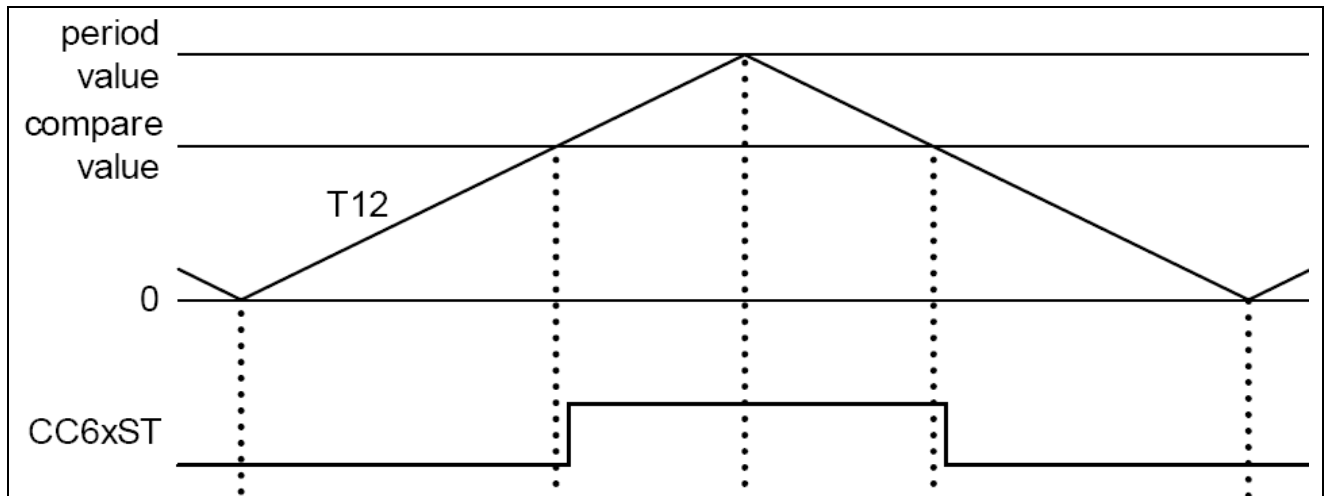
Example of a Symmetrical / Center-Aligned PWM-signal:



Two examples of 3 (6) Symmetrical / Center-Aligned PWM-signals:



Example of a Compare Event:



Note:

The bit CC6xST indicates a compare event.

CC6xST **Set**: T12 counter value above the compare value.

CC6xST **Reset**: T12 counter value below the compare value.





Note:

Now we are going to generate the following Symmetrical / Center-Aligned PWM-signals:

Port Lines	Signal	Duty Cycle [%]
P3.0	CC60_0	25
P3.2	CC61_0	50
P3.4	CC62_0	75

The input clock of the CCU6 module is not important.

Additionally, we intend to show additional information on the Logic Analyser (LGA) Screenshots (e.g. **one match**, **period match**).

GPIO use:

Port Lines	Function	Comment
P3.1	Show one match	Toggled via Software in the corresponding ISR
P3.3	Show period match	Toggled via Software in the corresponding ISR



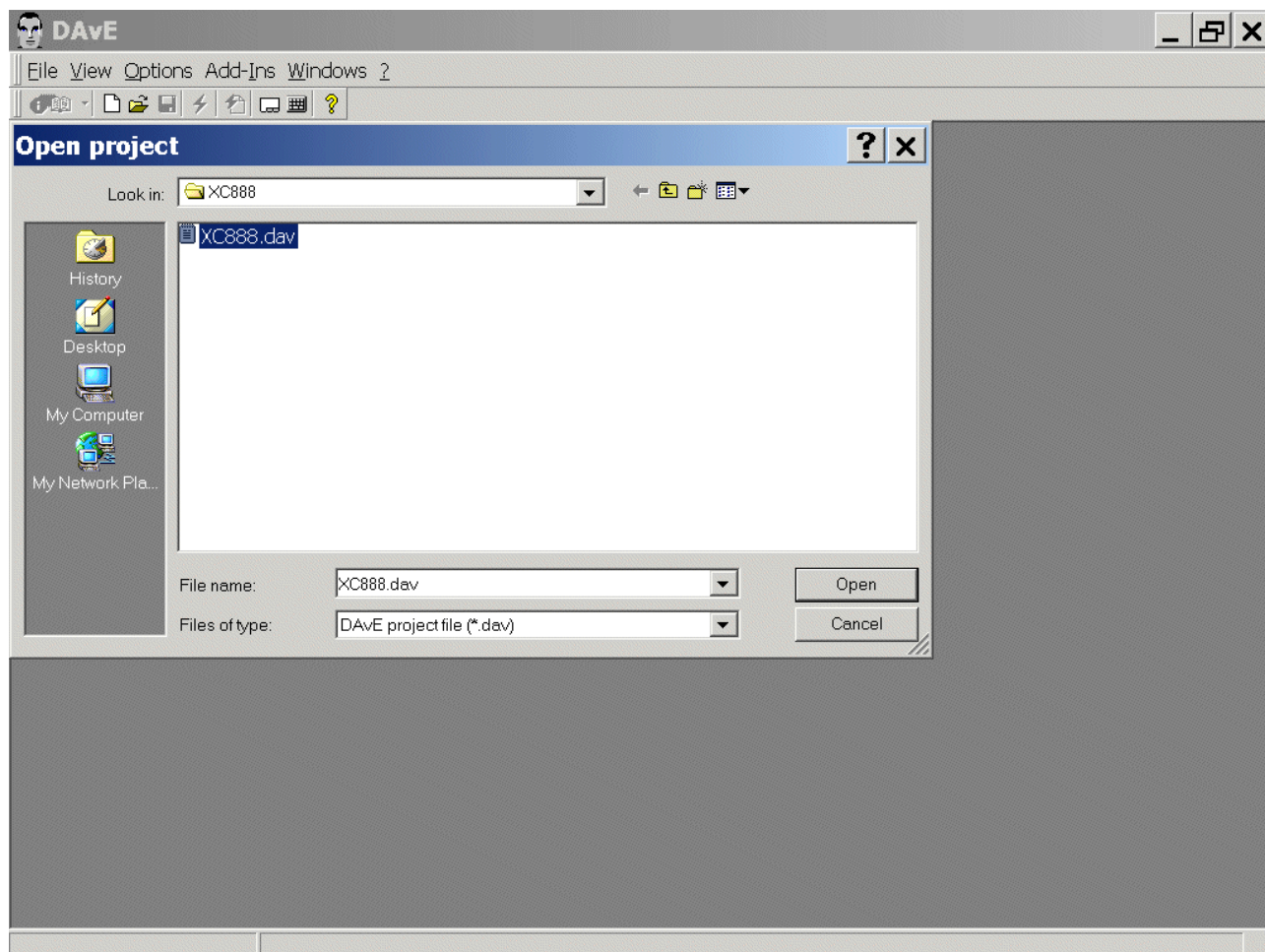
Start the program generator DAVe and open your [XC888.dav](#) DAVe project:

File

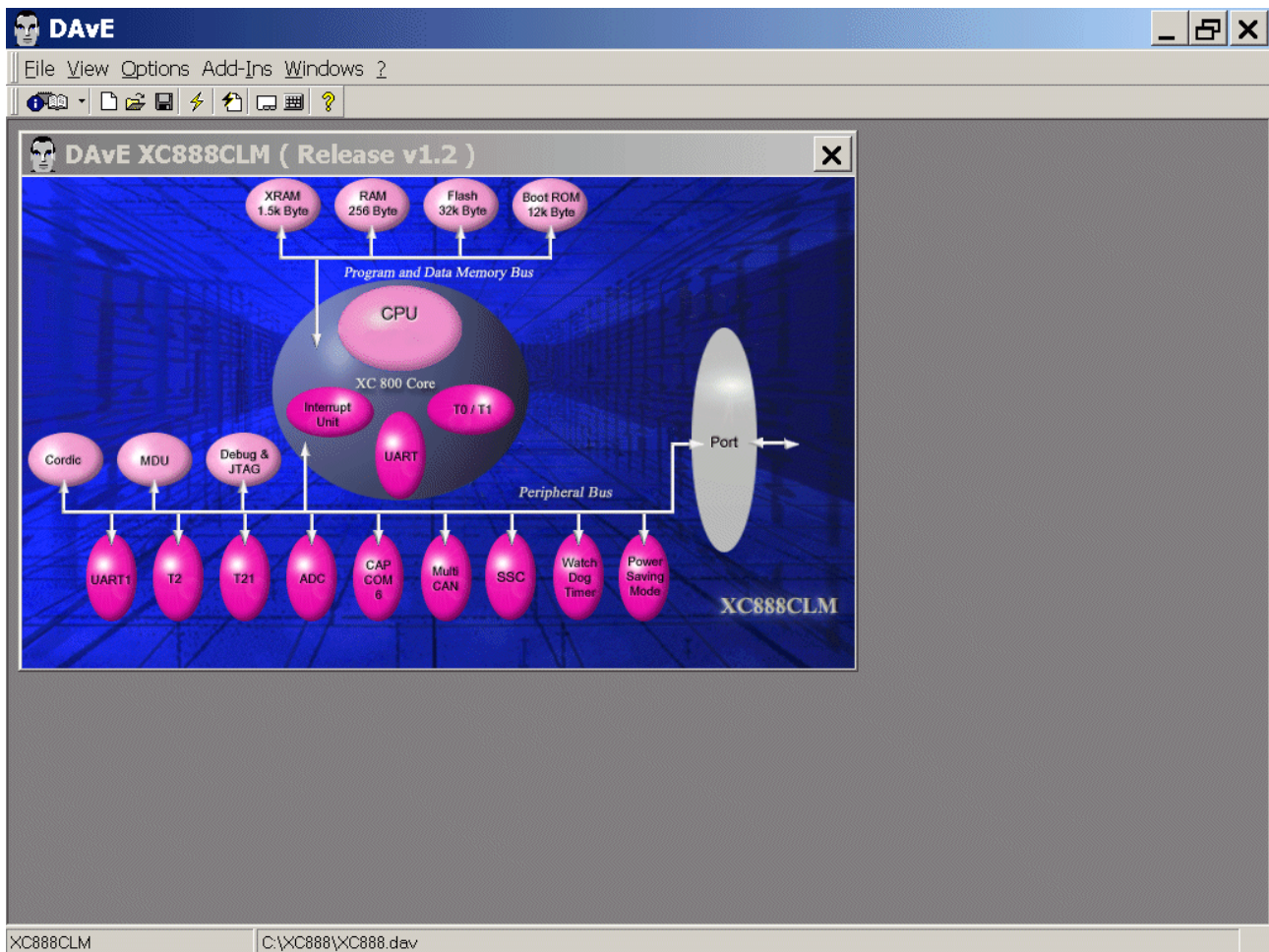
Open

Location: [C:\XC888](#)

Filename: [XC888.dav](#)

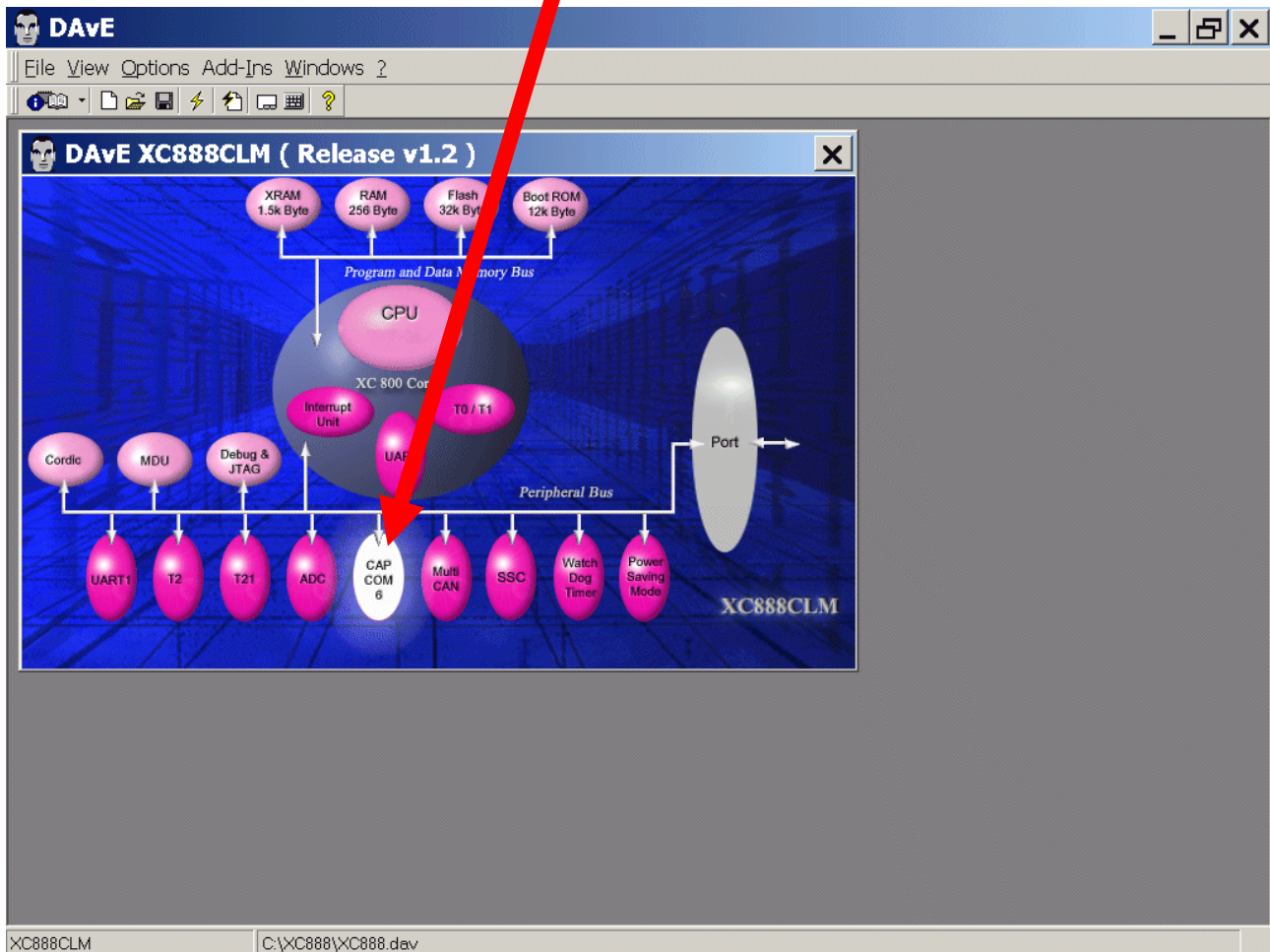


Click Open



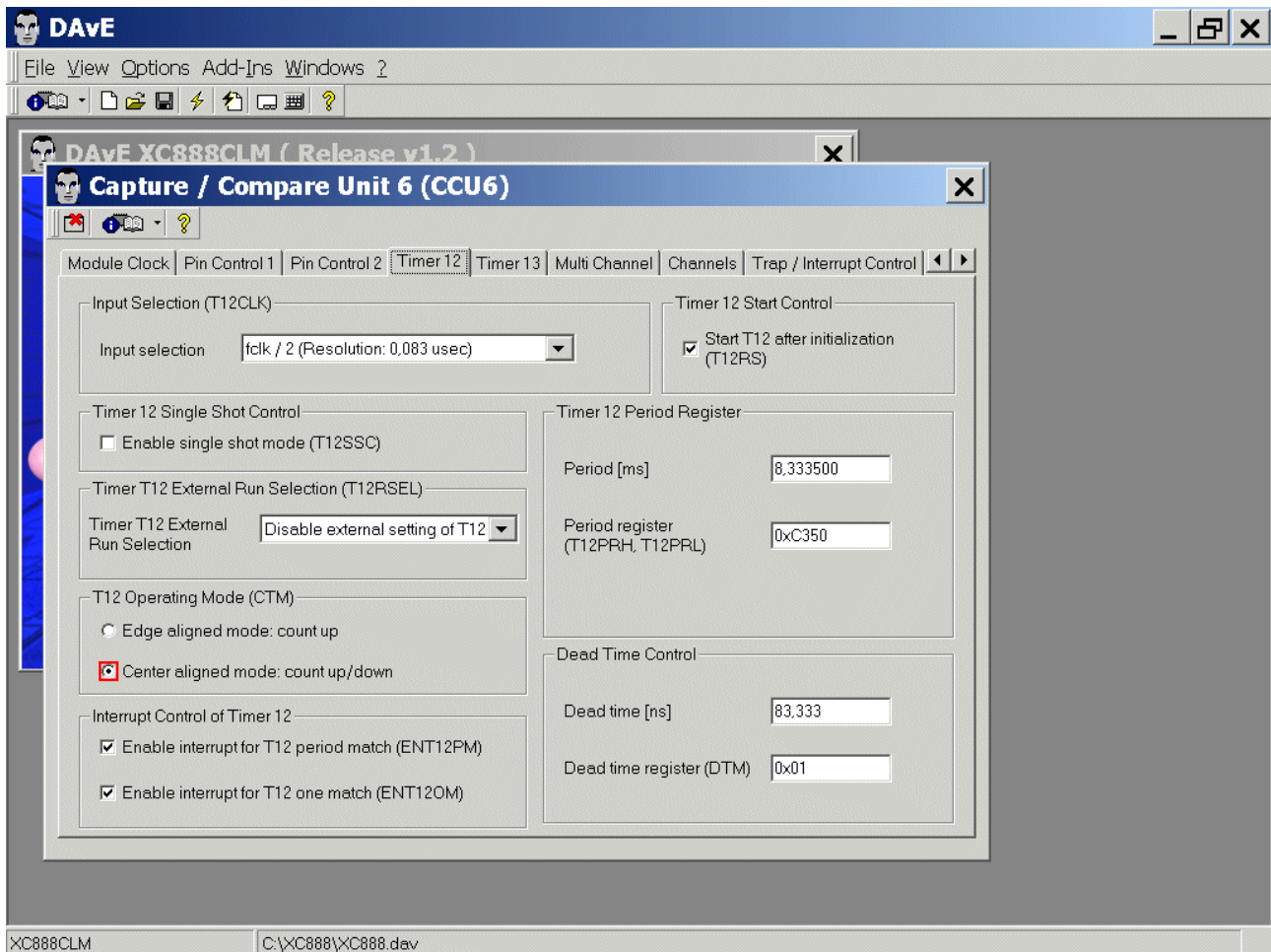
(Re)Configuration of the CAPCOM 6 (CCU6) module:

The configuration dialog can be opened by clicking the specific block/module.




CCU6: Module Clock: (do nothing)
CCU6: Pin Control 1: (do nothing)
CCU6: Pin Control 2: (do nothing)

CCU6: Timer T12: T12 Operating Mode (CTM): **click**  Center aligned mode: count up/down



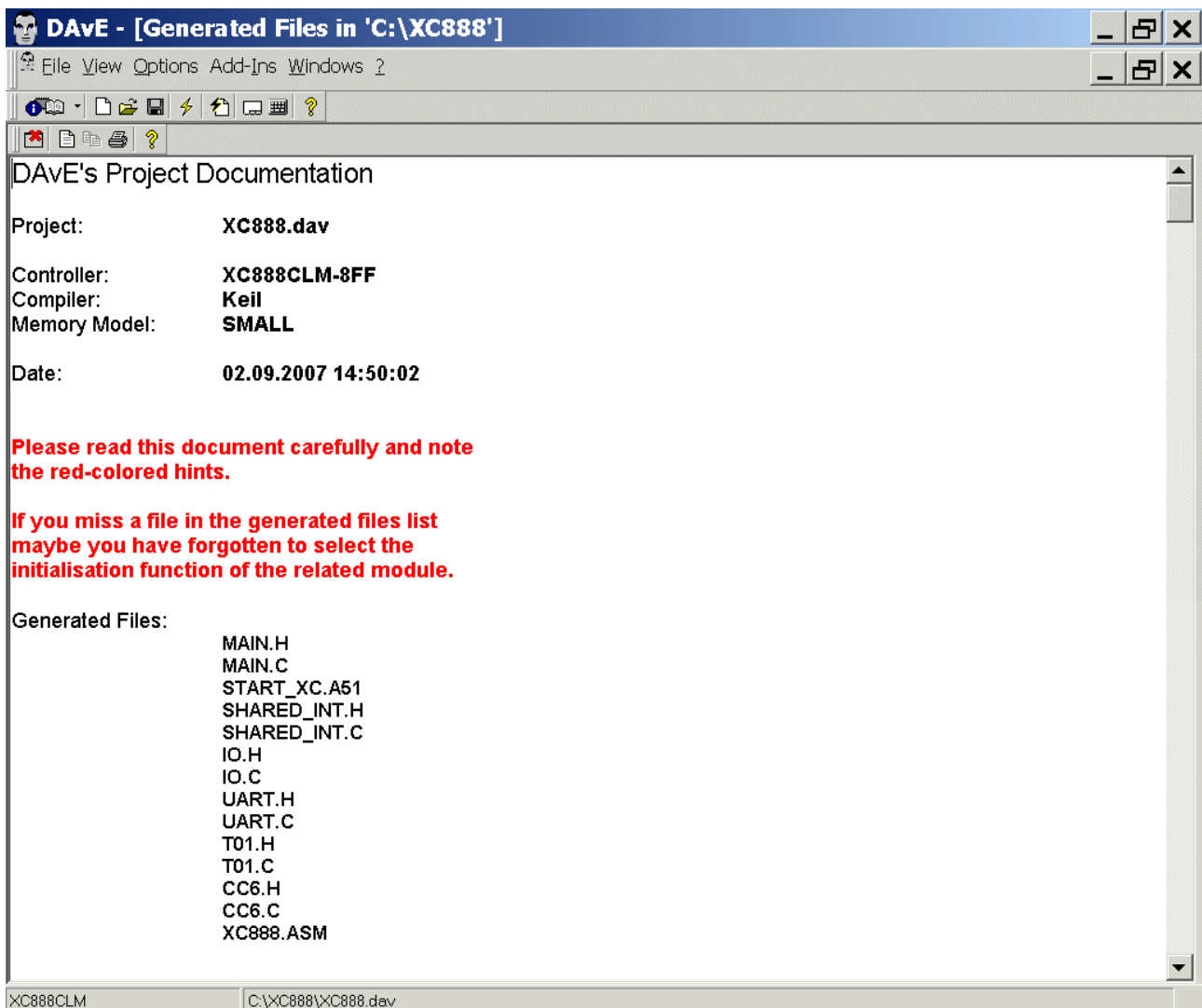
Exit this dialog now by clicking  the close button.

Generate Code:

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



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



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



Start Keil μ Vision and open your Keil Project:

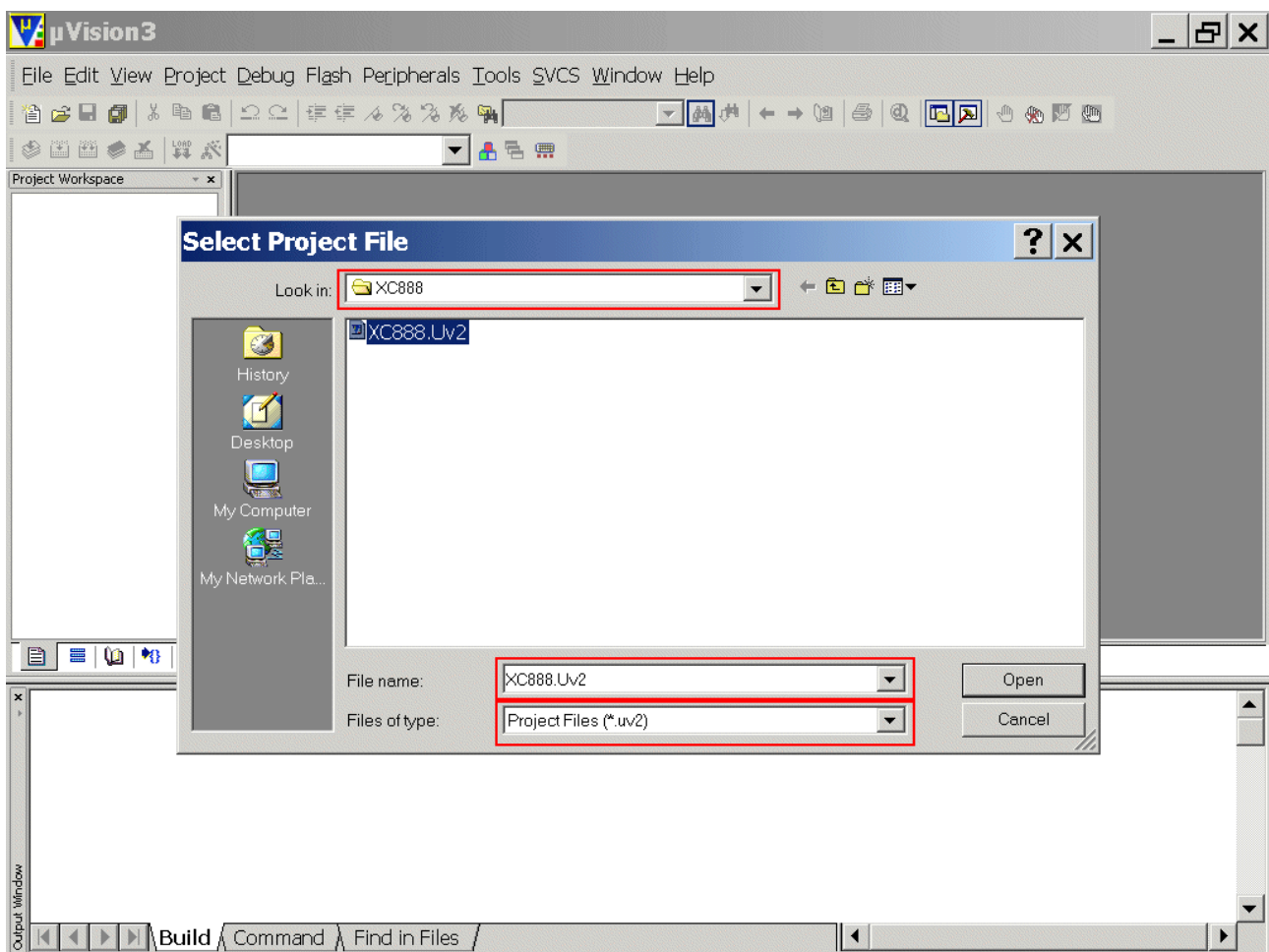
If you see an open project – close it: **Project - Close Project**

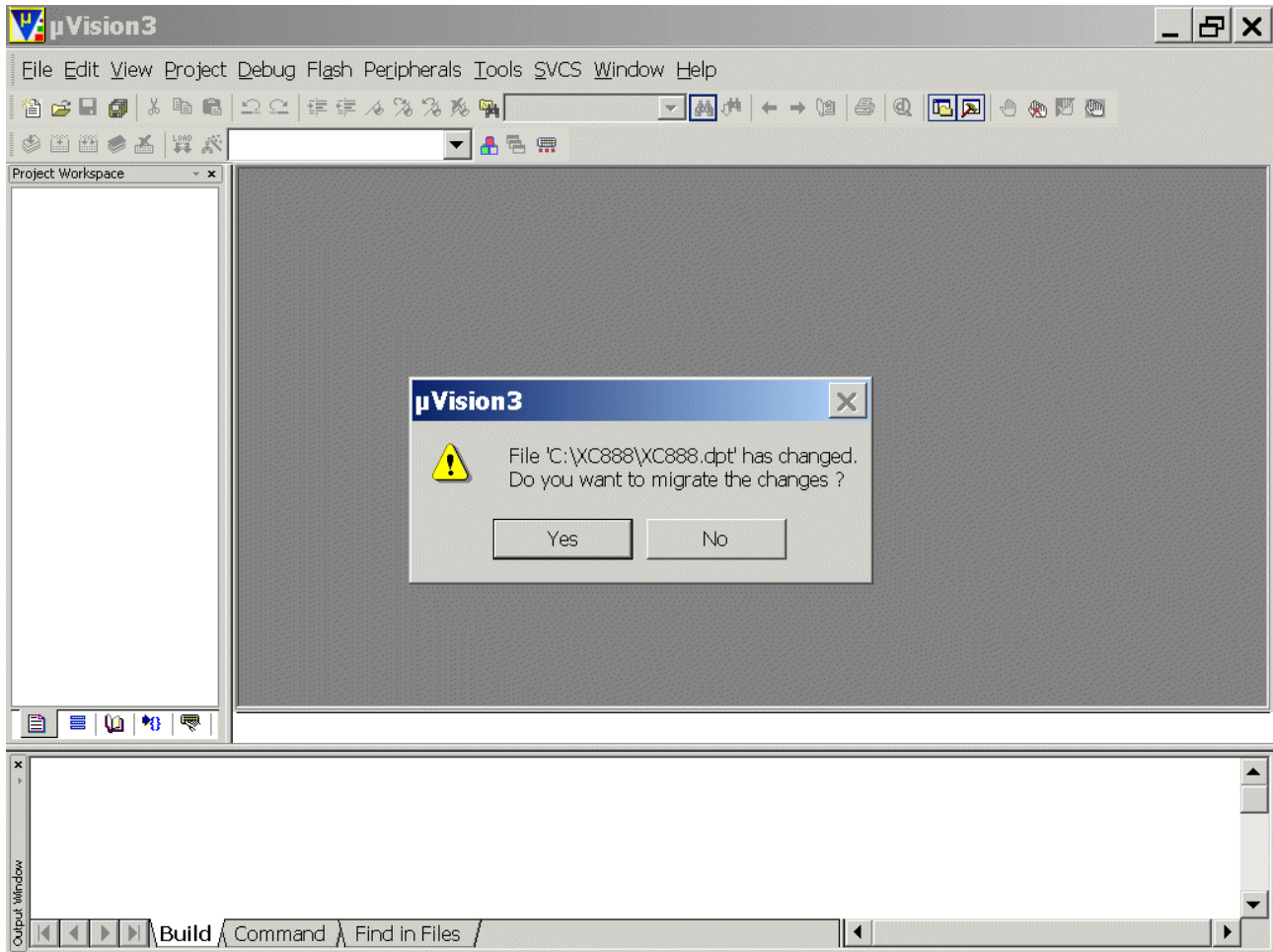
Project - Open Project

Select Project File: **Look in:** choose C:\XC888

Select Project File: **Files of type:** choose Project Files (*.uv2)
choose XC888.Uv2


Open

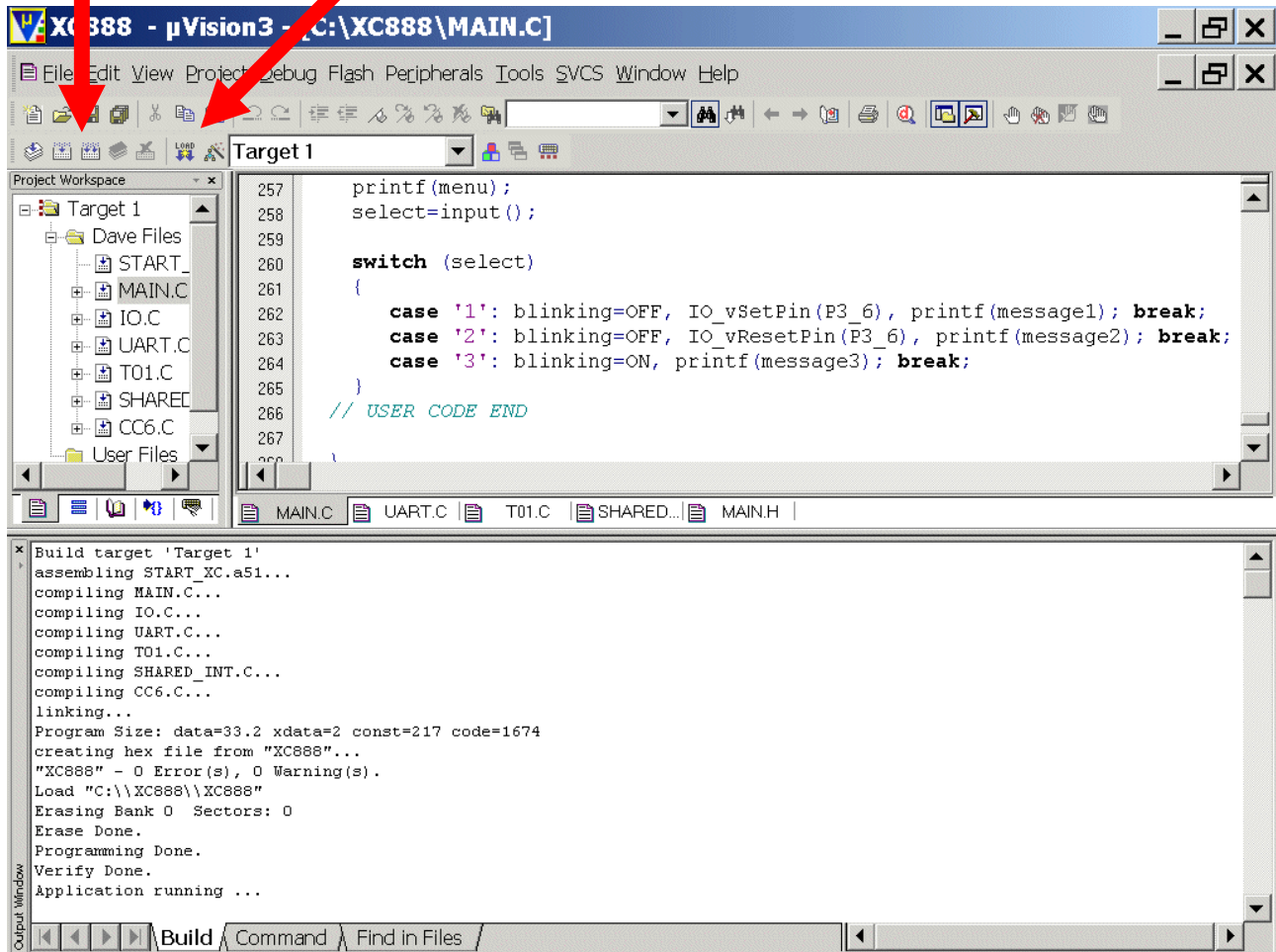





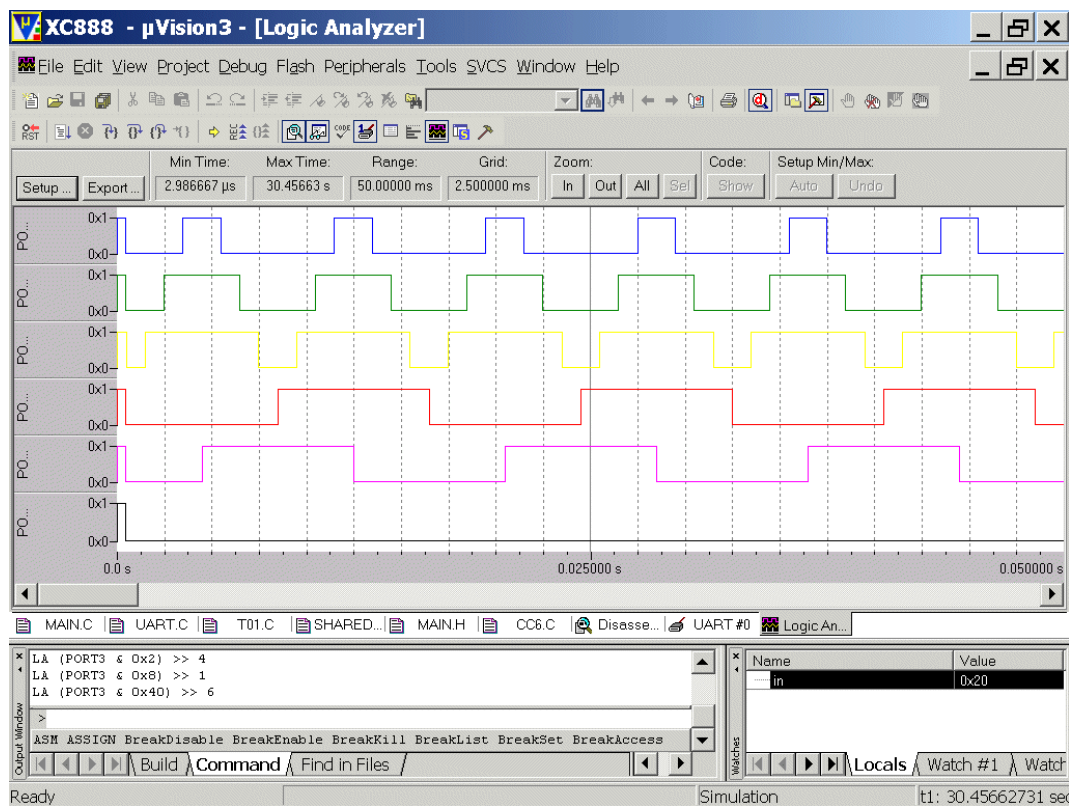
Click Yes


1.) click:  Rebuild all target files

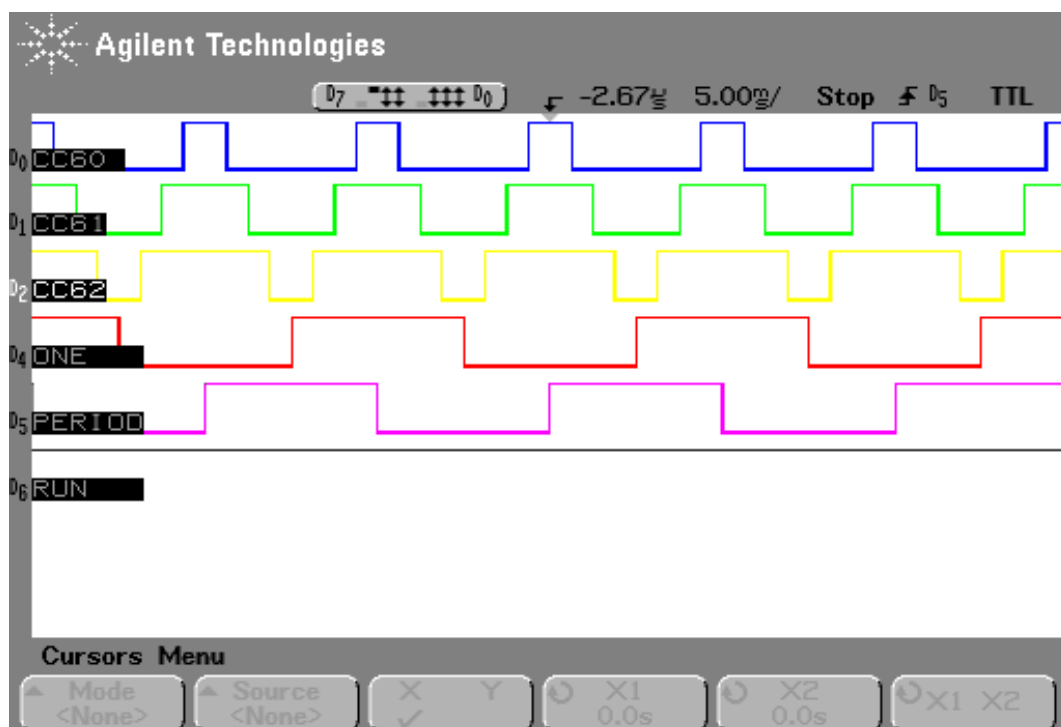
2.) click:  Download to On-Chip-Flash Memory



See the result (Simulator ):



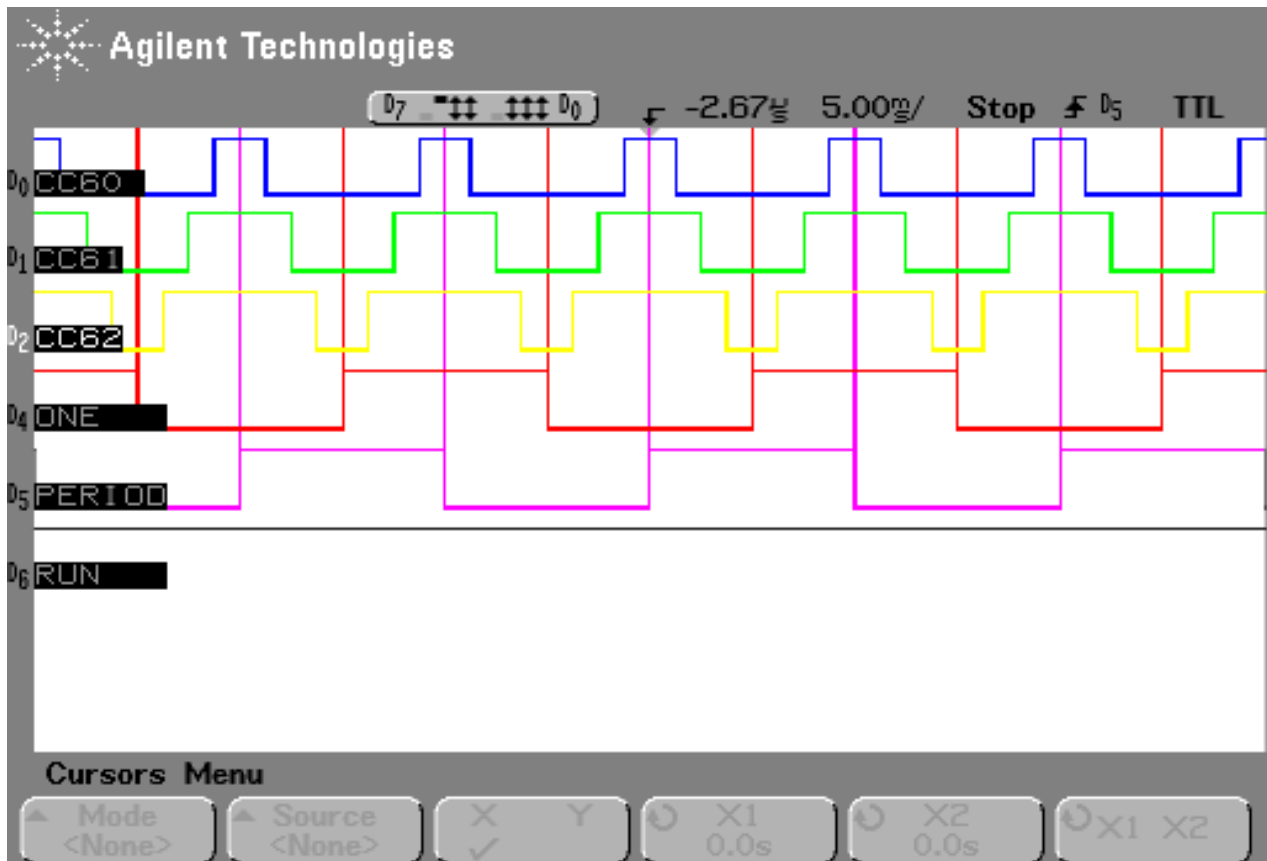
See the result (LGA ):





Note:

You can see a **period-match** while counting up.
You can see a **one-match** while counting down.



**3.) 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	5 (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 = 5 % 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 only for measurement	



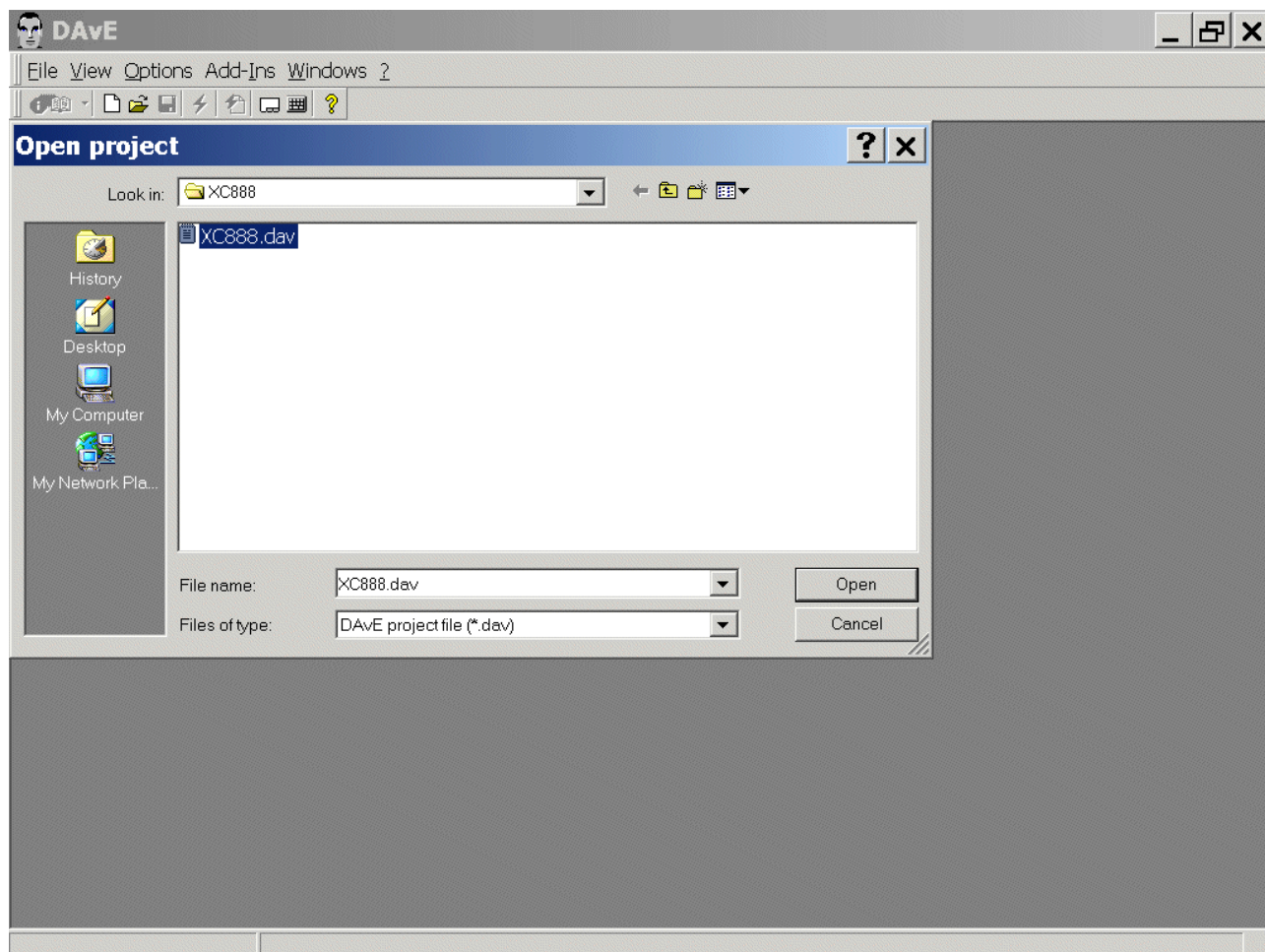
Start the program generator DAVe and open your [XC888.dav](#) DAVe project:

File

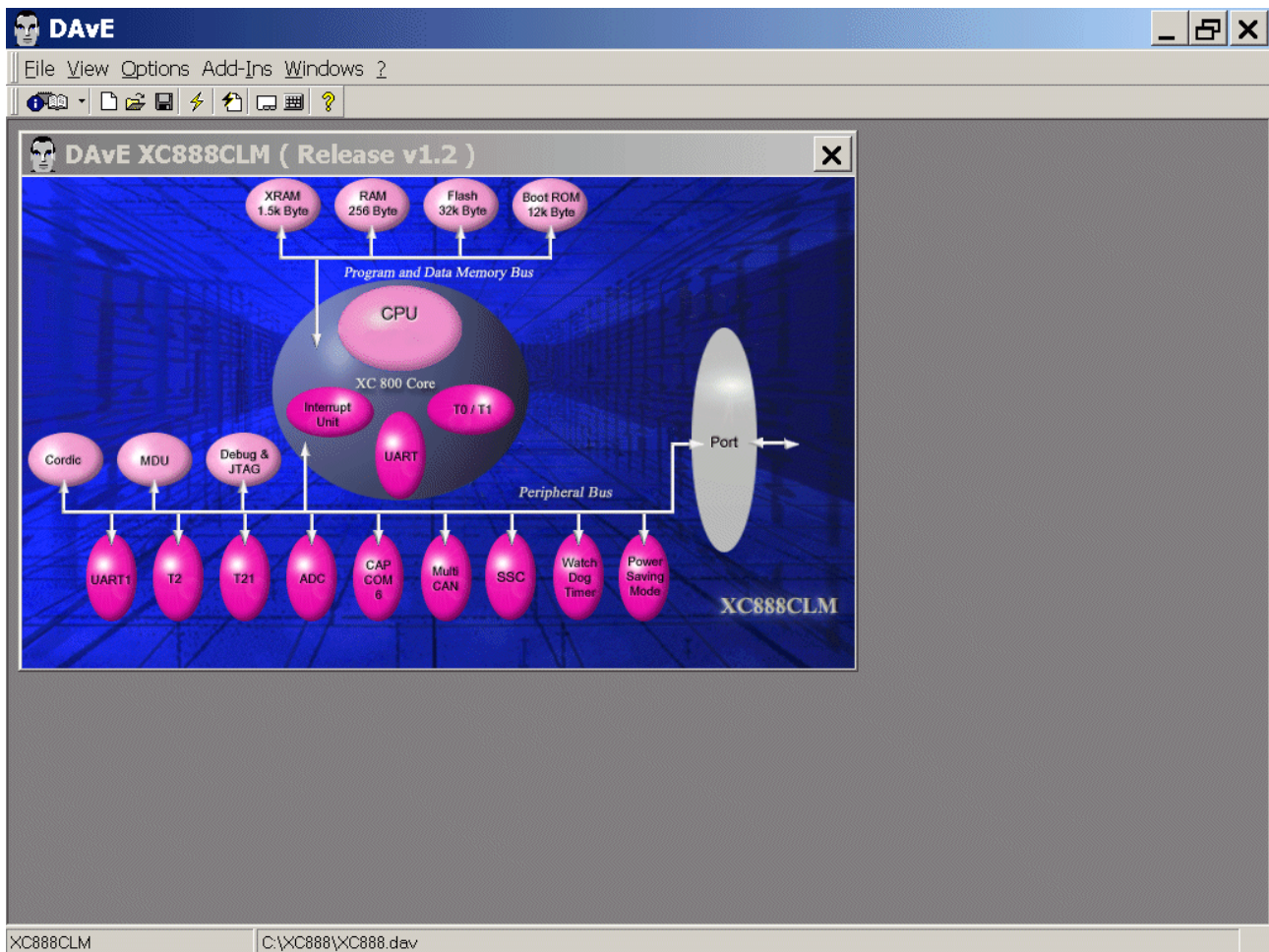
Open

Location: [C:\XC888](#)

Filename: [XC888.dav](#)

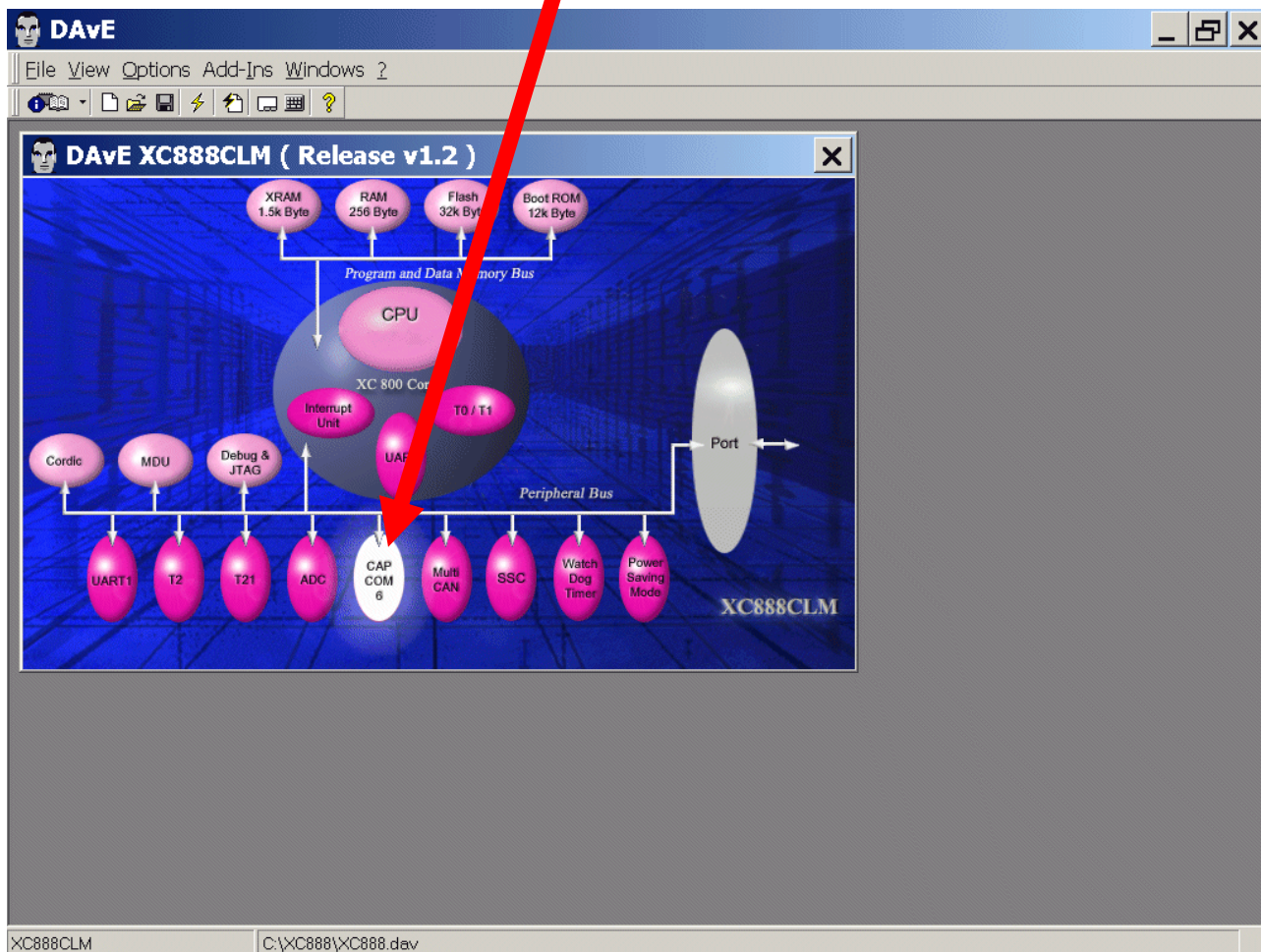


Click Open

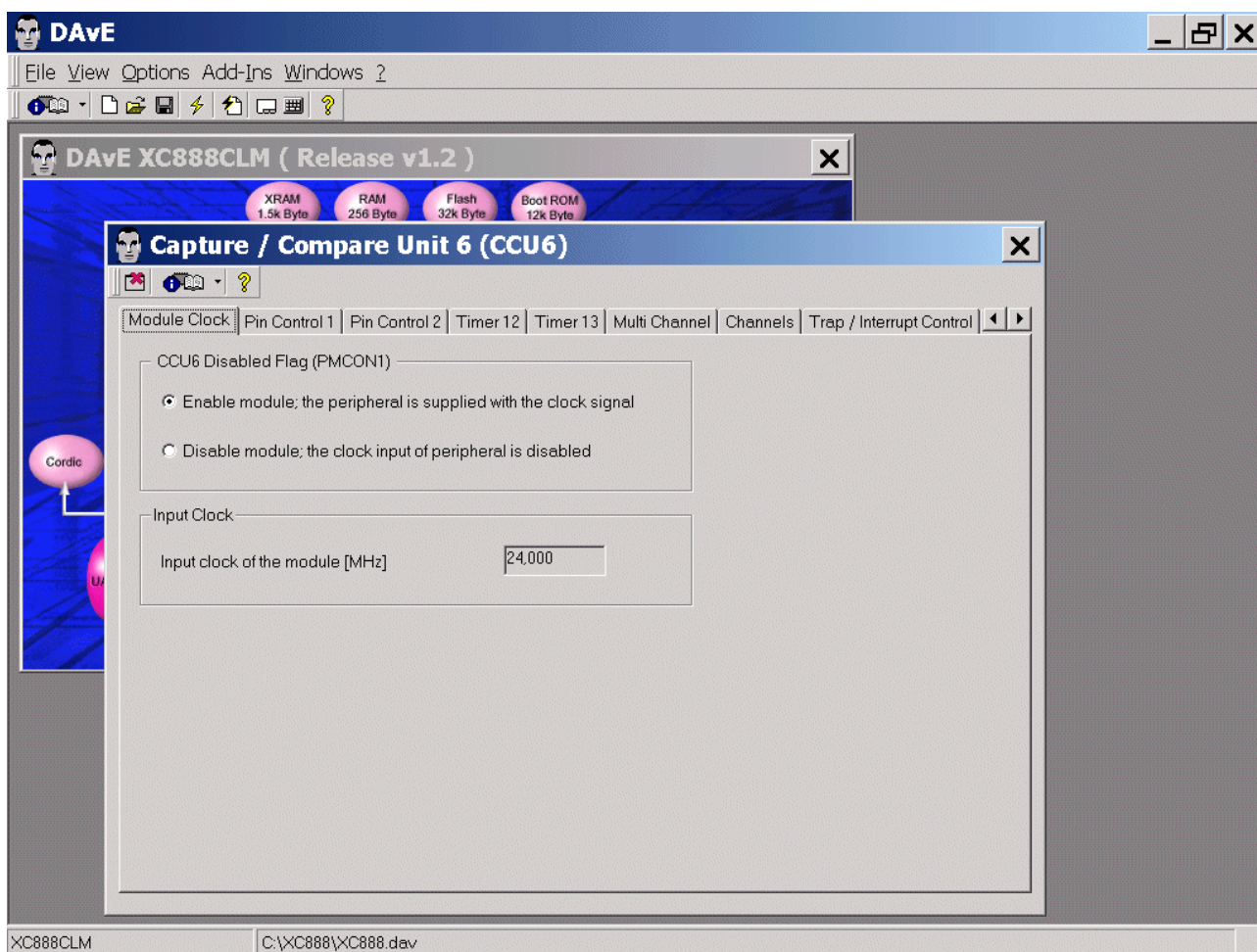


Reconfiguration of the CAPCOM 6 module:

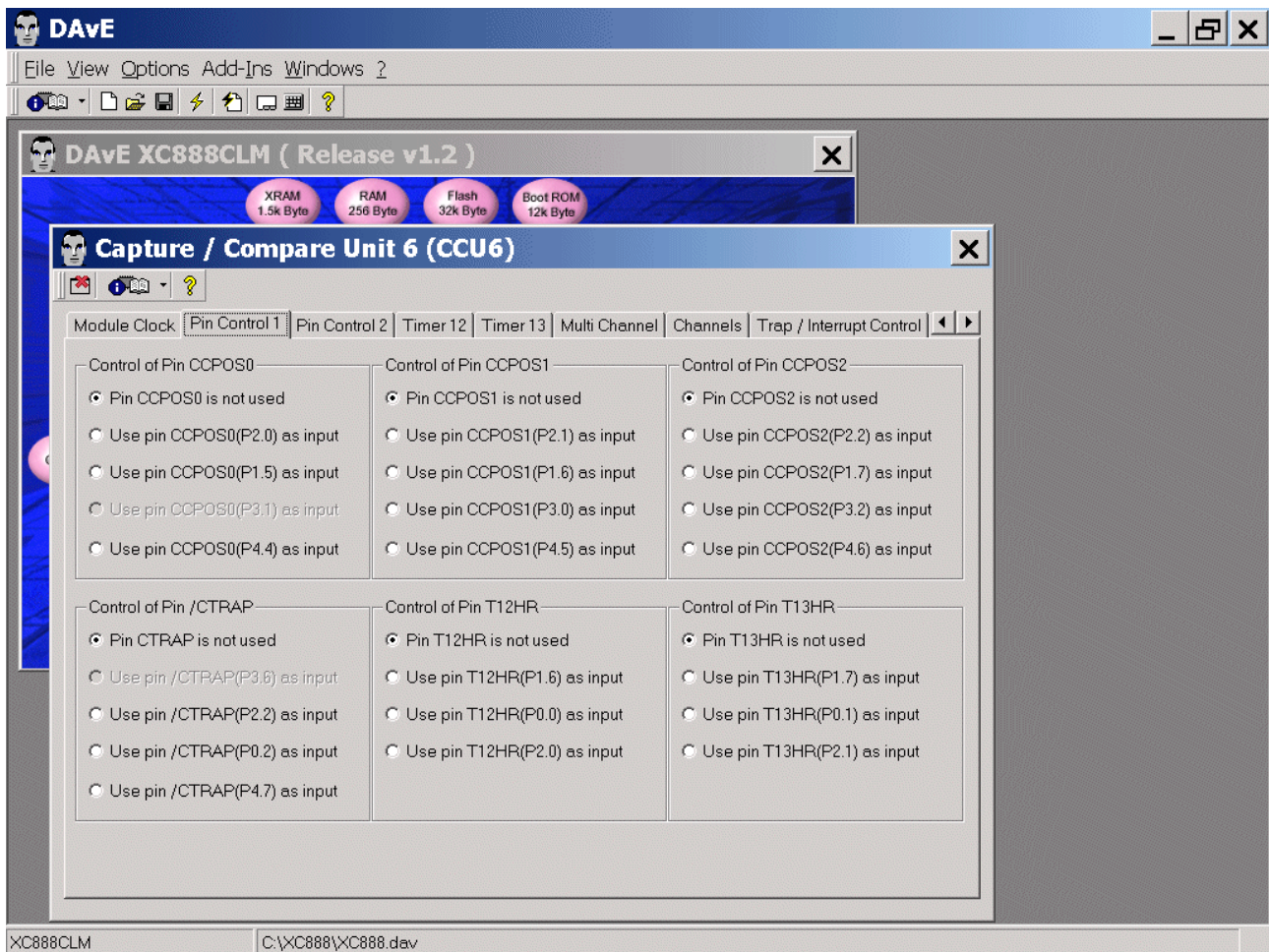
The configuration dialog can be opened by clicking the specific block/module.



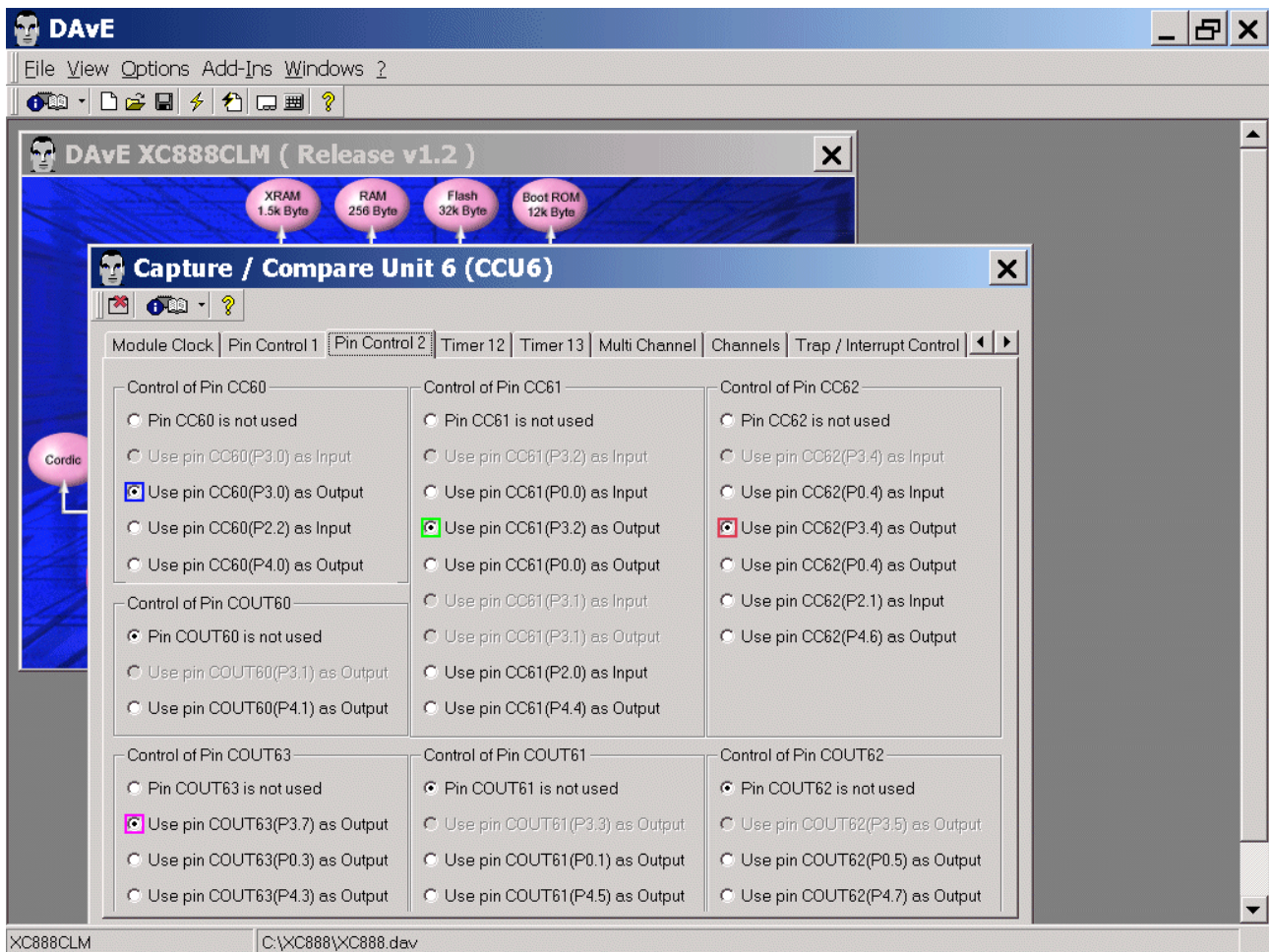
CCU6: Module Clock: (do nothing)



CCU6: Pin Control 1: (do nothing)



CCU6: Pin Control 2: (do nothing)



Remember:

Port_3 pins used by our PWM module:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (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)

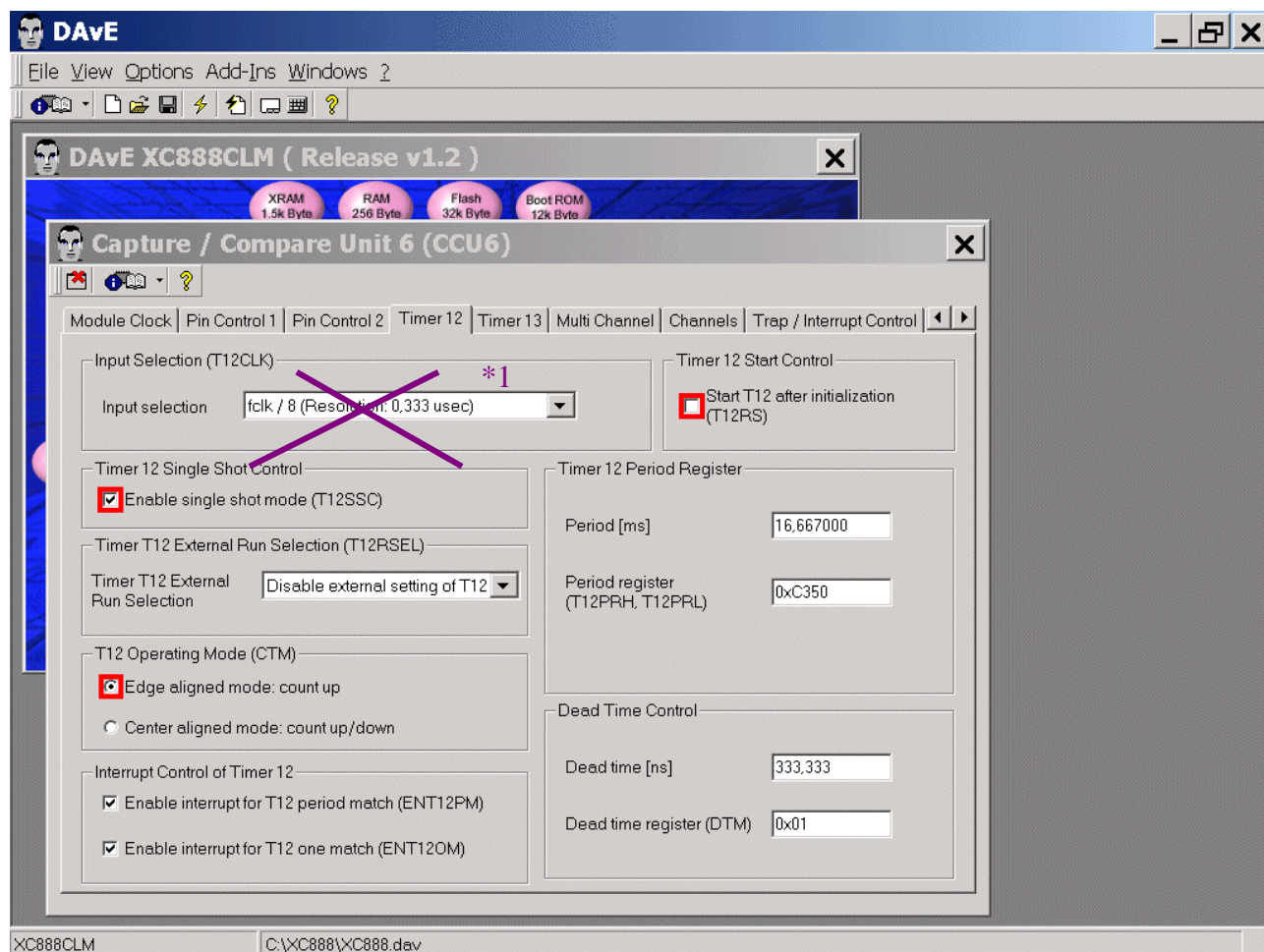
Timer 12: “note length”:

CCU6: Timer 12: Input Selection (T12CLK): choose $f_{clk} / 8 \rightarrow$ Resolution = $85,333 \mu s$ *1

CCU6: Timer 12: T12 Start Control: click to unselect ☐ Start T12 after initialization

CCU6: Timer 12: T12 Single Shot Control: click ☒ Enable single shot mode (T12SSC)

CCU6: Timer 12: T12 Operating Mode: click ☒ Edge aligned mode: count up



*1: See next page !!!

Timer 12 Resolution = $11,719 \text{ kHz} / 85,333 \mu s$



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

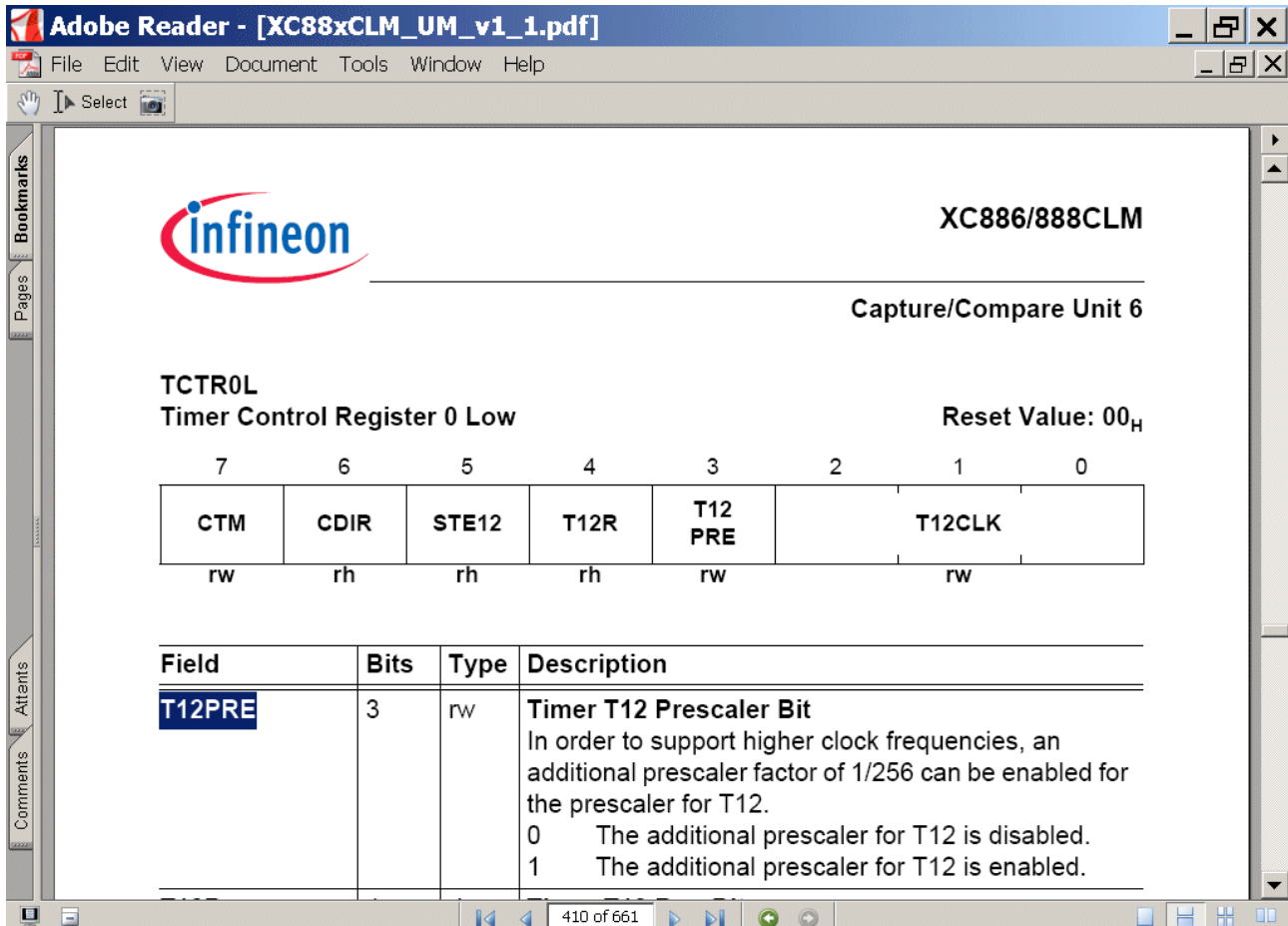


Note:

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.



XC886/888CLM

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
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:

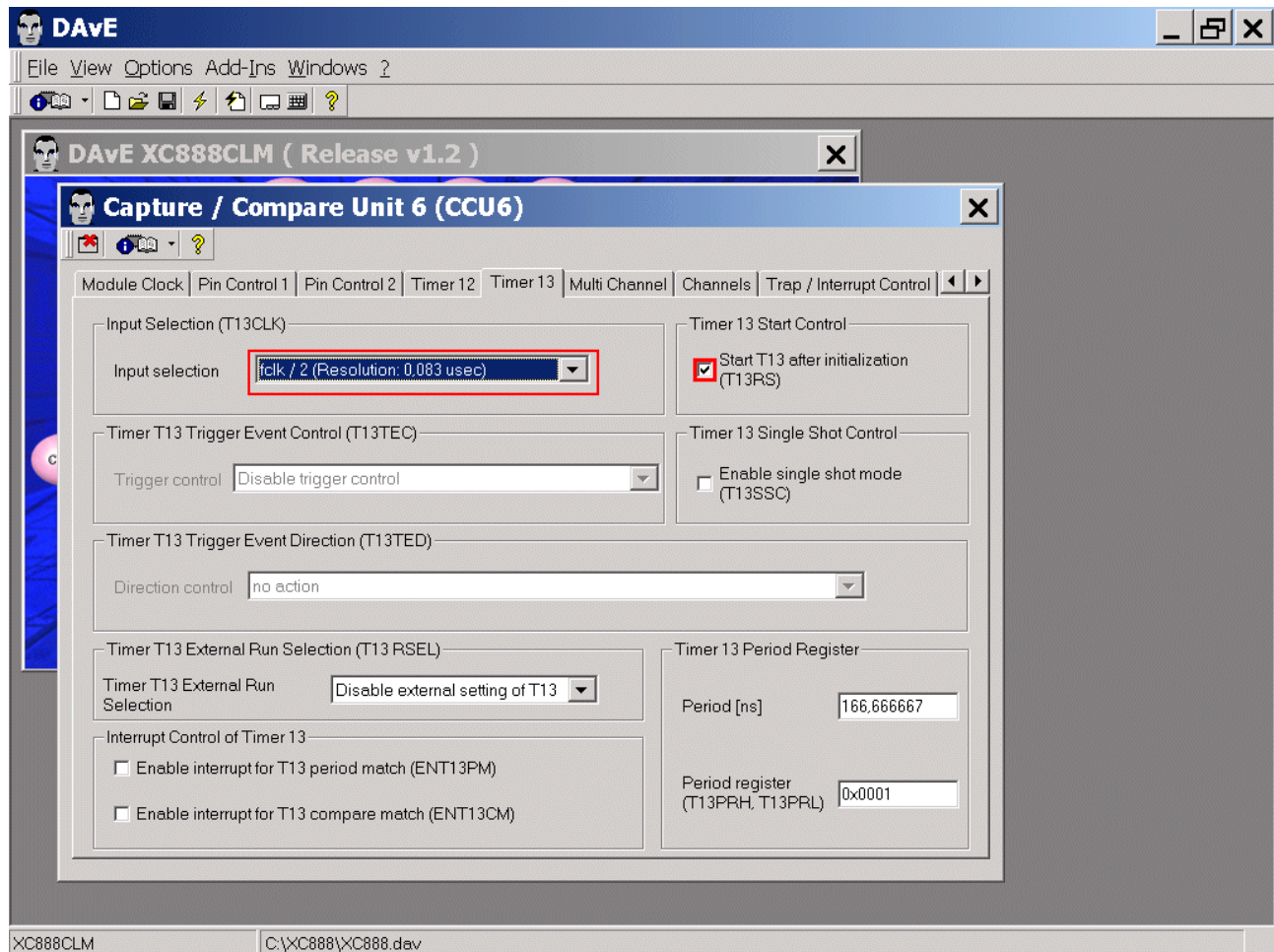
Timer 12 Resolution:

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)

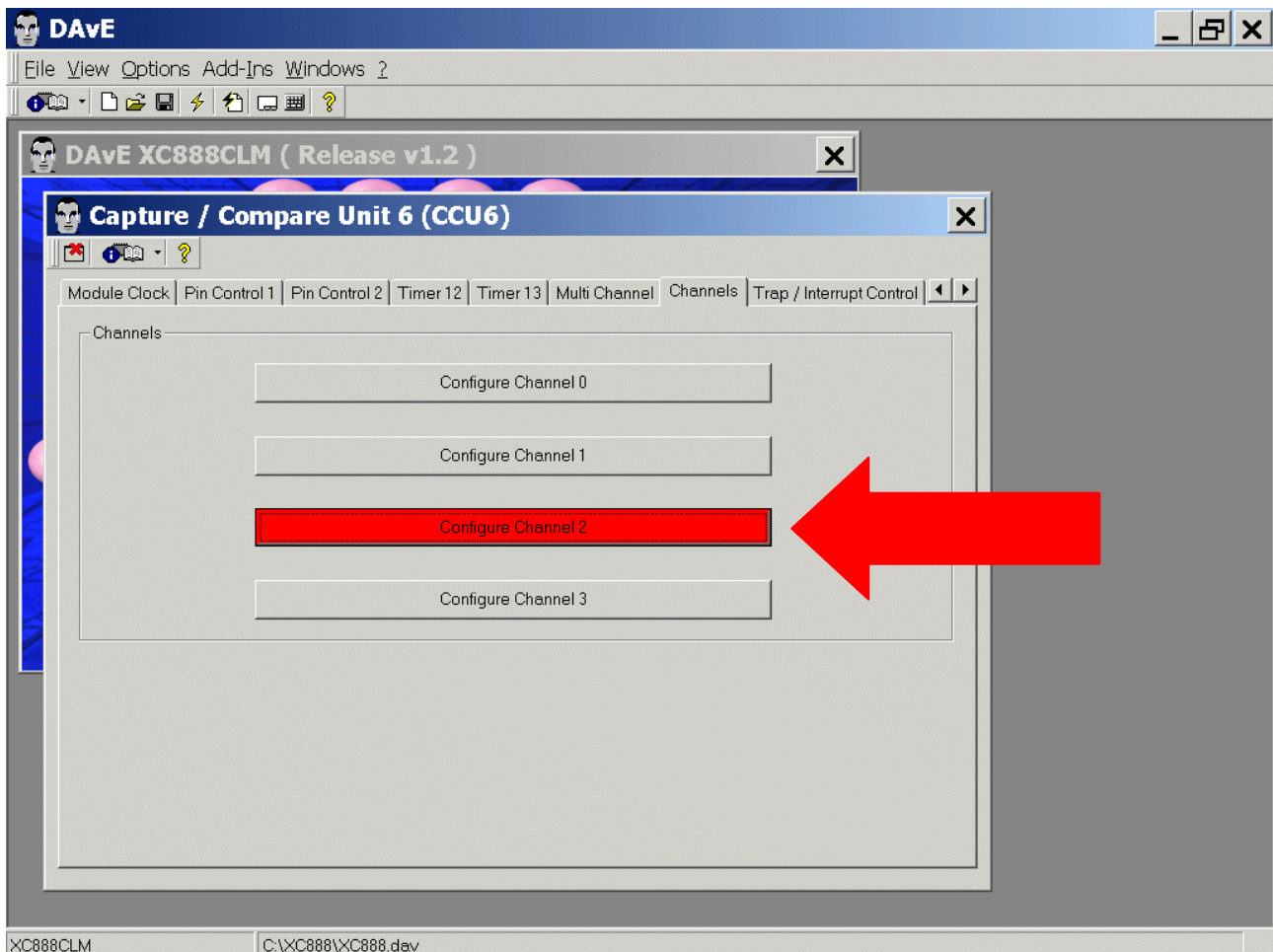
CCU6: Timer T13: Timer 13 Start Control: **click** ✓ Start T13 after initialization (T13RS)



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

CCU6: Multi Channel: (do nothing)

CCU6: Channels: click Configure Channel 2

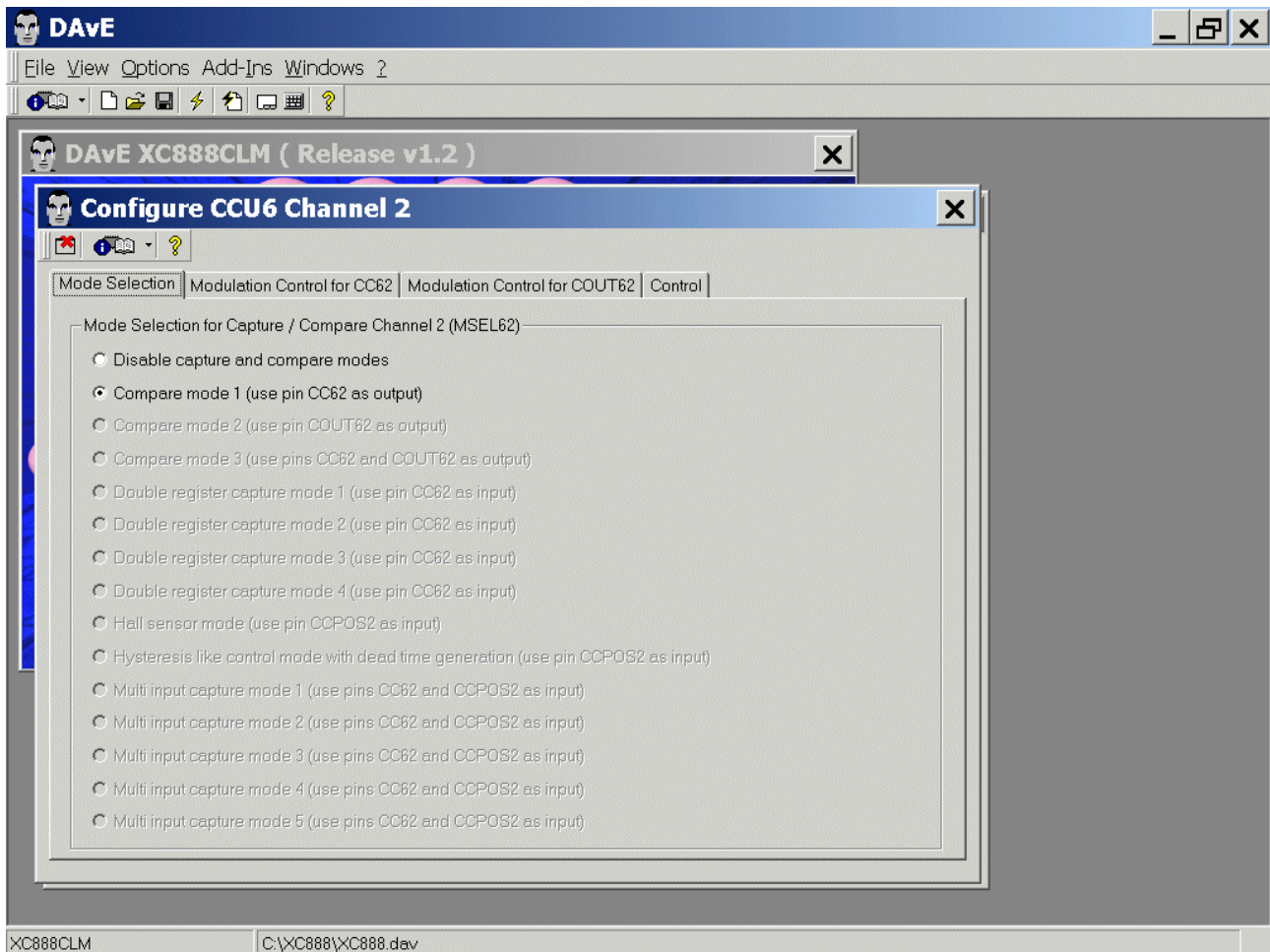


Remember:

Port_3 pins used by our PWM module:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (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	COU63_0	50 (note frequency, Timer_13)

CCU6: Channels: Configure Channel 2: Mode Selection: (do nothing)

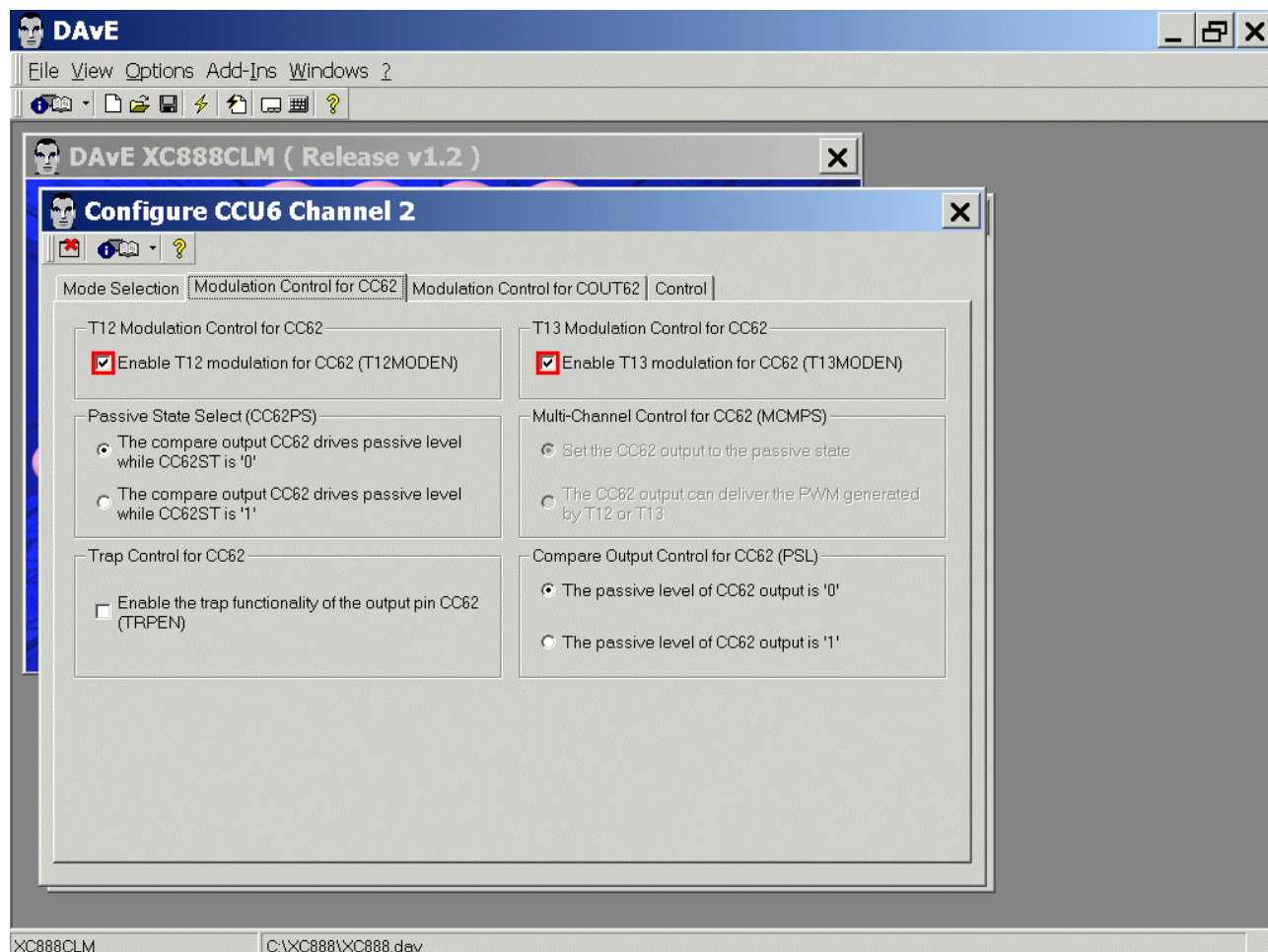


CCU6: Channels: Configure Channel 2: Modulation Control for CC62:

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

CCU6: Channels: Configure Channel 2: Modulation Control for CC62:

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

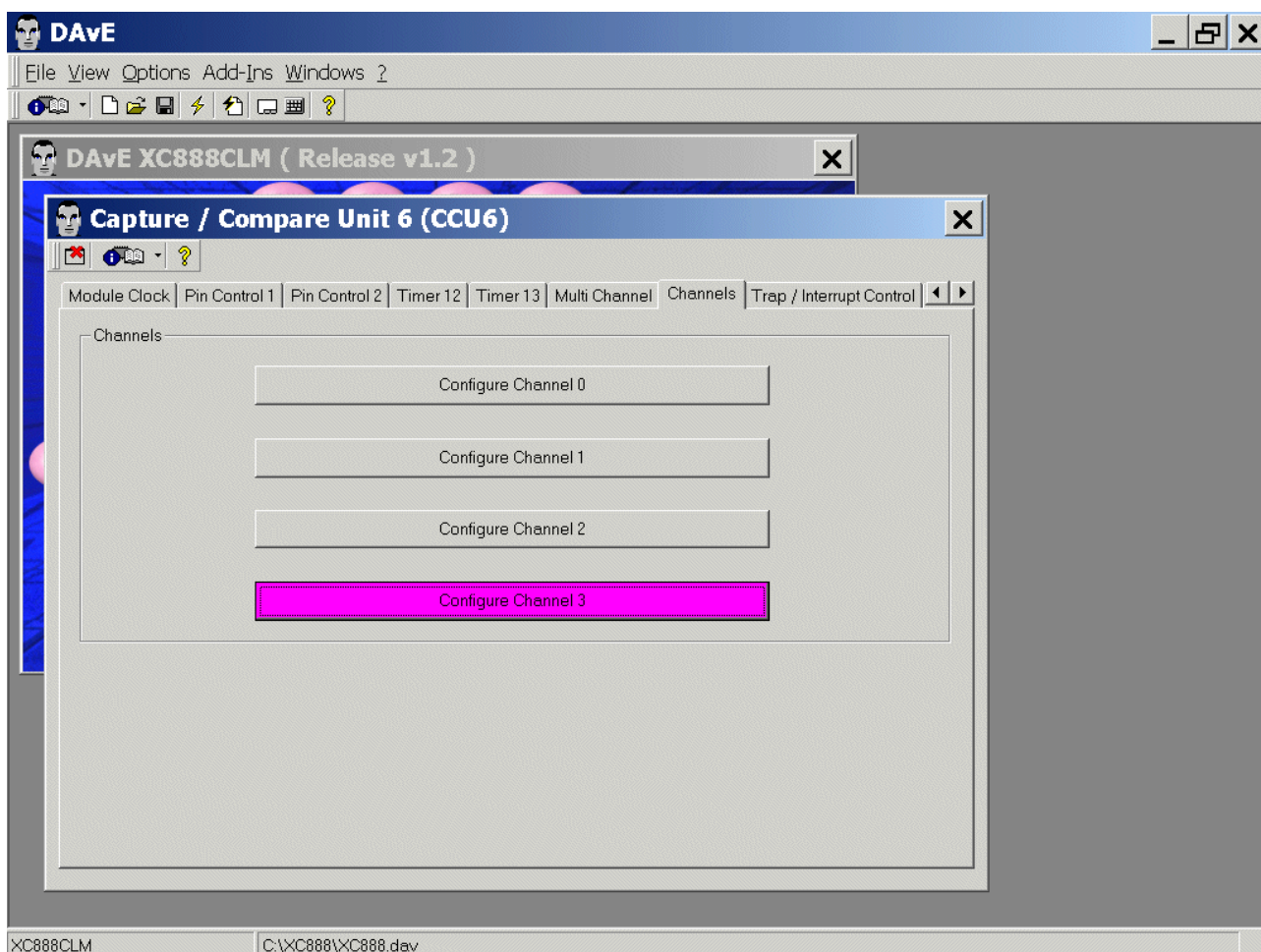


CCU6: Channels: Configure Channel 2: Modulation Control for COUT62: (do nothing)

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

Exit this dialog now by clicking  the close button.

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

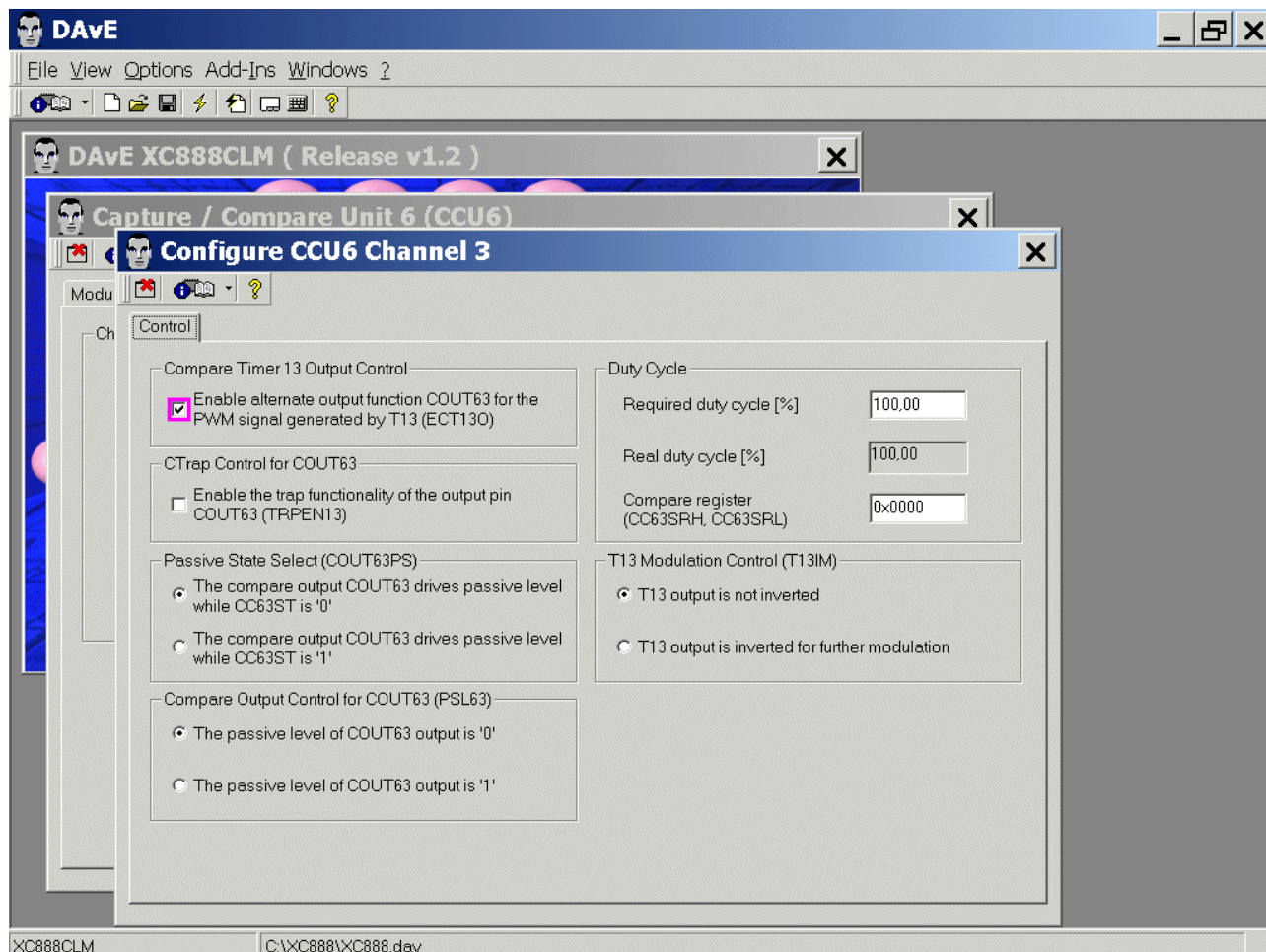


Remember:

Port_3 pins used by our PWM module:

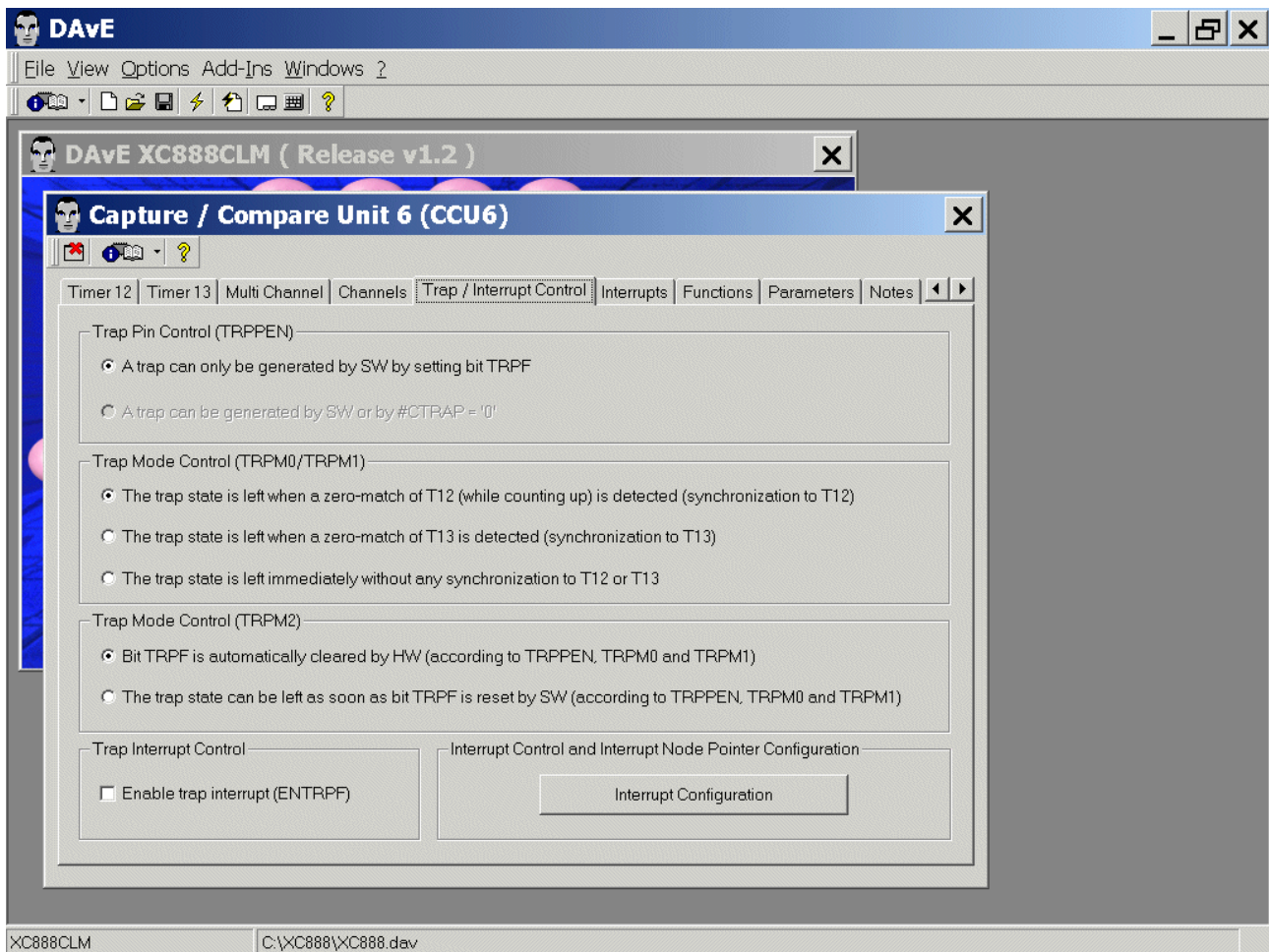
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

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

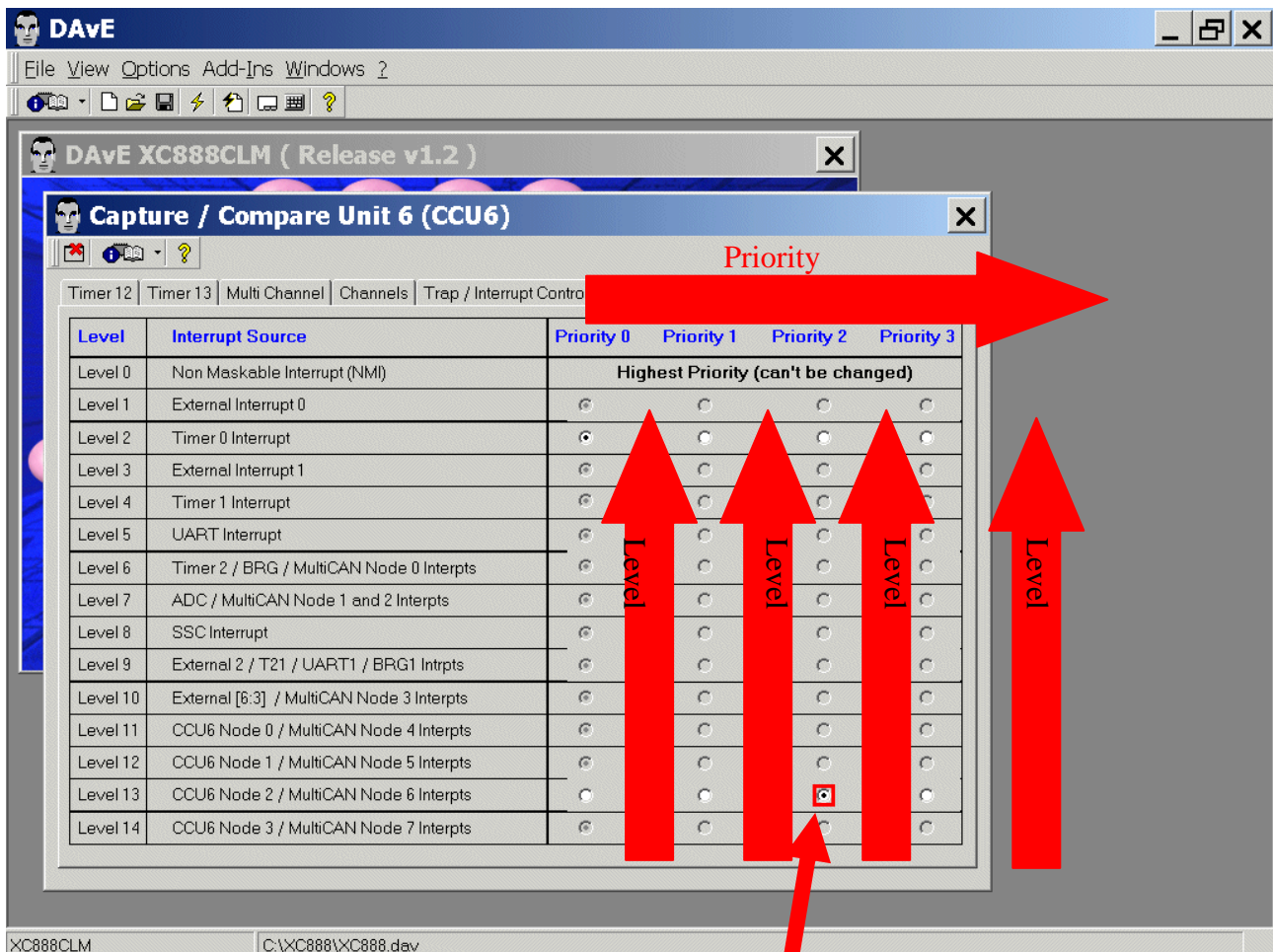


Exit this dialog now by clicking  the close button.

CCU6: Trap / Interrupt Control: (do nothing)



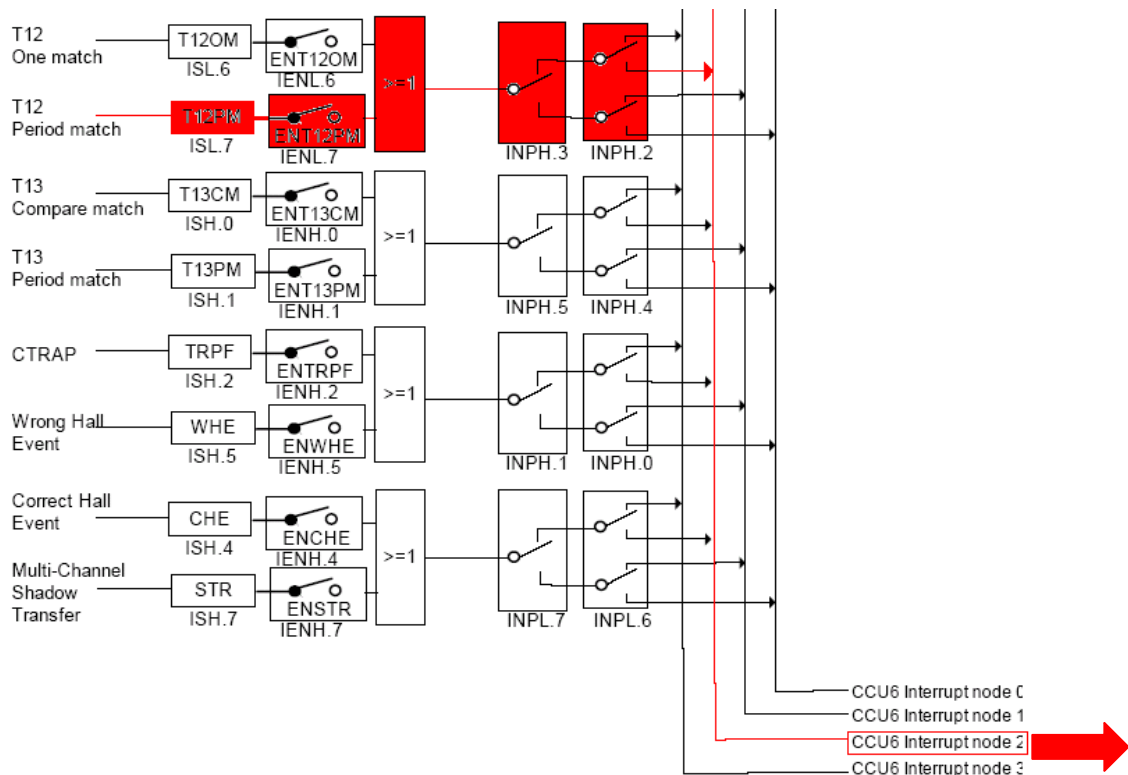
CCU6: Interrupts: change CCU6 Interrupt Node 2 to Priority 2, Level 13



Priority 2, Level 13



Interrupt Node Selection:



Note:

I could not find a similar picture in the XC888 User's Manual, therefore I took the picture from the XC866 User's Manual.



CCU6 interrupt node 0
MultiCAN_4

CCU6SR0
IRCON3.0

CANSRC4
IRCON3.1

≥ 1

ECCIP0
IEN1.4

0053 H

IP1.4/
IPH1.4

CCU6 interrupt node 1
MultiCAN_5

CCU6SR1
IRCON3.4

CANSRC5
IRCON3.5

≥ 1

ECCIP1
IEN1.5

005B H

IP1.5/
IPH1.5

CCU6 interrupt node 2
MultiCAN_6

CCU6SR2
IRCON4.0

CANSRC6
IRCON4.1

≥ 1

ECCIP2
IEN1.6

0063 H

IP1.6/
IPH1.6

CCU6 interrupt node 3
MultiCAN_7

CCU6SRC3
IRCON4.4

CANSRC7
IRCON4.5

≥ 1

ECCIP3
IEN1.7

006B H

IP1.7/
IPH1.7

EA
IEN0.7

Polling Sequence

Highest

Lowest

Priority Level



Interrupt Source and Vector:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

Interrupt System

Table 5-1 Interrupt Vector Addresses (cont'd)

Interrupt Node	Vector Address	Assignment for XC886/888	Enable Bit	SFR
XINTR6	0033 _H	MultiCAN Nodes 1 and 2 ADC[1:0]	EADC	IEN1
XINTR7	003B _H	SSC	ESSC	
XINTR8	0043 _H	External Interrupt 2 T21 CORDIC UART1 UART1 Fractional Divider (Normal Divider Overflow) MDU[1:0]	EX2	
XINTR9	004B _H	External Interrupt 3 External Interrupt 4 External Interrupt 5 External Interrupt 6 MultiCAN Node 3	EXM	
XINTR10	0053 _H	CCU6 INP0 MultiCAN Node 4	ECCIP0	
XINTR11	005B _H	CCU6 INP1 MultiCAN Node 5	ECCIP1	
XINTR12	0063 _H	CCU6 INP2 MultiCAN Node 6	ECCIP2	
XINTR13	006B _H	CCU6 INP3 MultiCAN Node 7	ECCIP3	

123 of 661



Priority Structure within Interrupt Level:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

Table 5-2 Priority Structure within Interrupt Level

Source	Level
Non-Maskable Interrupt (NMI)	(highest)
External Interrupt 0	1
Timer 0 Interrupt	2
External Interrupt 1	3
Timer 1 Interrupt	4
UART Interrupt	5
Timer 2, UART Normal Divider Overflow, LIN, MultiCAN Interrupt	6
ADC, MultiCAN Interrupt	7
SSC Interrupt	8
External Interrupt 2, Timer 21, UART1, UART1 Normal Divider Overflow, CORDIC, MDU Interrupt	9
External Interrupt [6:3], MultiCAN	10
CCU6 Interrupt Node Pointer 0, MultiCAN Interrupt	11
CCU6 Interrupt Node Pointer 1, MultiCAN Interrupt	12
CCU6 Interrupt Node Pointer 2, MultiCAN Interrupt	13
CCU6 Interrupt Node Pointer 3, MultiCAN Interrupt	14

User's Manual
Interrupt System, V 1.0

5-13

V1.1, 2007-05

124 of 661



Interrupt Priority Level Selection:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

Bookmarks Pages Attachments Comments

Table 5-3 Interrupt Priority Level Selection

IPH.x / IPH1.x	IP.x / IP1.x	Priority Level
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

7,93 x 11,25 in 143 of 661



Interrupt Priority Register:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

IP1
Interrupt Priority 1 Register Reset Value: 00_H

7	6	5	4	3	2	1	0
PCCIP3	PCCIP2	PCCIP1	PCCIP0	PXM	PX2	PSSC	PADC
rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
PADC	0	rW	Priority Level Low Bit for Interrupt Node XINTR6
PSSC	1	rW	Priority Level Low Bit for Interrupt Node XINTR7
PX2	2	rW	Priority Level Low Bit for Interrupt Node XINTR8
PXM	3	rW	Priority Level Low Bit for Interrupt Node XINTR9
PCCIP0	4	rW	Priority Level Low Bit for Interrupt Node XINTR10
PCCIP1	5	rW	Priority Level Low Bit for Interrupt Node XINTR11
PCCIP2	6	rW	Priority Level Low Bit for Interrupt Node XINTR12

User's Manual
Interrupt System, V 1.0

5-33

V1.1, 2007-05

144 of 661



Interrupt Priority High Register:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

IPH1
Interrupt Priority 1 High Register Reset Value: 00_H

7	6	5	4	3	2	1	0
PCCIP3H	PCCIP2H	PCCIP1H	PCCIP0H	PXMH	PX2H	PSSCH	PADCH
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PADCH	0	rw	Priority Level High Bit for Interrupt Node XINTR6
PSSCH	1	rw	Priority Level High Bit for Interrupt Node XINTR7
PX2H	2	rw	Priority Level High Bit for Interrupt Node XINTR8
PXMH	3	rw	Priority Level High Bit for Interrupt Node XINTR9
PCCIP0H	4	rw	Priority Level High Bit for Interrupt Node XINTR10
PCCIP1H	5	rw	Priority Level High Bit for Interrupt Node XINTR11
PCCIP2H	6	rw	Priority Level High Bit for Interrupt Node XINTR12
PCCIP3H	7	rw	Priority Level High Bit for Interrupt Node XINTR13

145 of 661



Location of the Interrupt Request Flags:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

Interrupt System

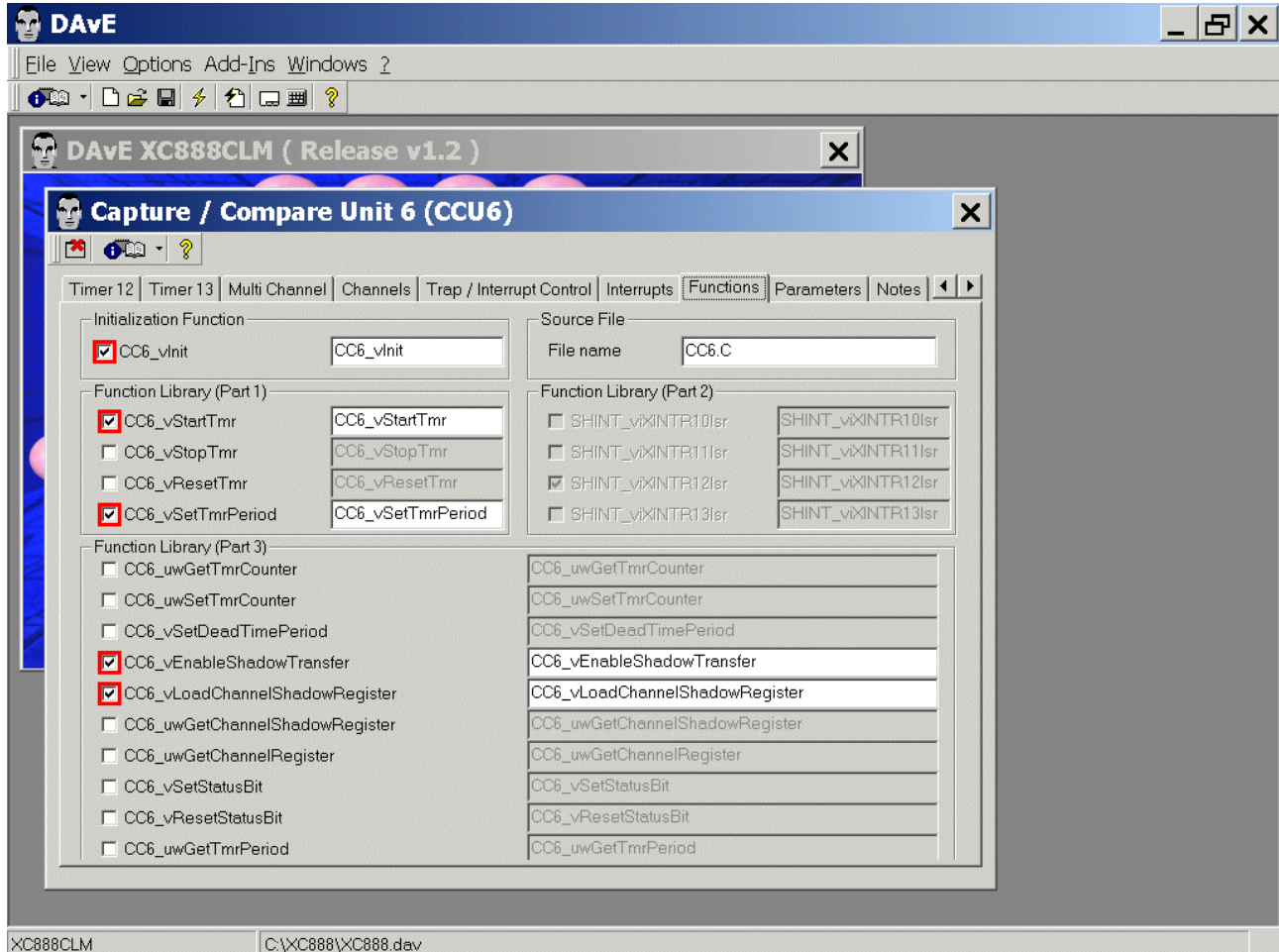
Table 5-4 Locations of the Interrupt Request Flags (cont'd)

Interrupt Source	Interrupt Flag	SFR
SSC Error	EIR	IRCON1
SSC Transmit	TIR	IRCON1
SSC Receive	RIR	IRCON1
MultiCAN Interrupt 0	CANSRC0 ¹⁾	IRCON2
MultiCAN Interrupt 1	CANSRC1 ¹⁾	IRCON1
MultiCAN Interrupt 2	CANSRC2 ¹⁾	IRCON1
MultiCAN Interrupt 3	CANSRC3 ¹⁾	IRCON2
MultiCAN Interrupt 4	CANSRC4 ¹⁾	IRCON3
MultiCAN Interrupt 5	CANSRC5 ¹⁾	IRCON3
MultiCAN Interrupt 6	CANSRC6 ¹⁾	IRCON4
MultiCAN Interrupt 7	CANSRC7 ¹⁾	IRCON4
CCU6 Node 0 Interrupt	CCU6SR0	IRCON3
CCU6 Node 1 Interrupt	CCU6SR1	IRCON3
CCU6 Node 2 Interrupt	CCU6SR2	IRCON4
CCU6 Node 3 Interrupt	CCU6SR3	IRCON4
Watchdog Timer NMI	FNMIWDT	NMISR
PLL NMI	FNMIPLL	NMISR
Flash NMI	FNMIFLASH	NMISR
VDD Prewarning NMI	FNMI VDD	NMISR
VDDP Prewarning NMI	FNMI VDDP	NMISR
Flash ECC NMI	FNMI ECC	NMISR

¹⁾ Different MultiCAN interrupt can be assigned to different MultiCAN interrupt output lines [7:0] via MultiCAN registers NIPR_x/MOIPR_n.

147 of 661

CCU6: Functions: Initialization Function: **click/check** ☒ CCU6_vInit
 CCU6: Functions: Function Library (Part 1): **click** ☒ CCU6_vStartTmr
 CCU6: Functions: Function Library (Part 1): **click** ☒ CCU6_vSetTmrPeriod
 CCU6: Functions: Function Library (Part 3): **click** ☒ CCU6_vEnableShadowTransfer
 CCU6: Functions: Function Library (Part 3): **click** ☒ CCU6_vLoadChannelShadowRegister




CCU6: Parameters: (do nothing)

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

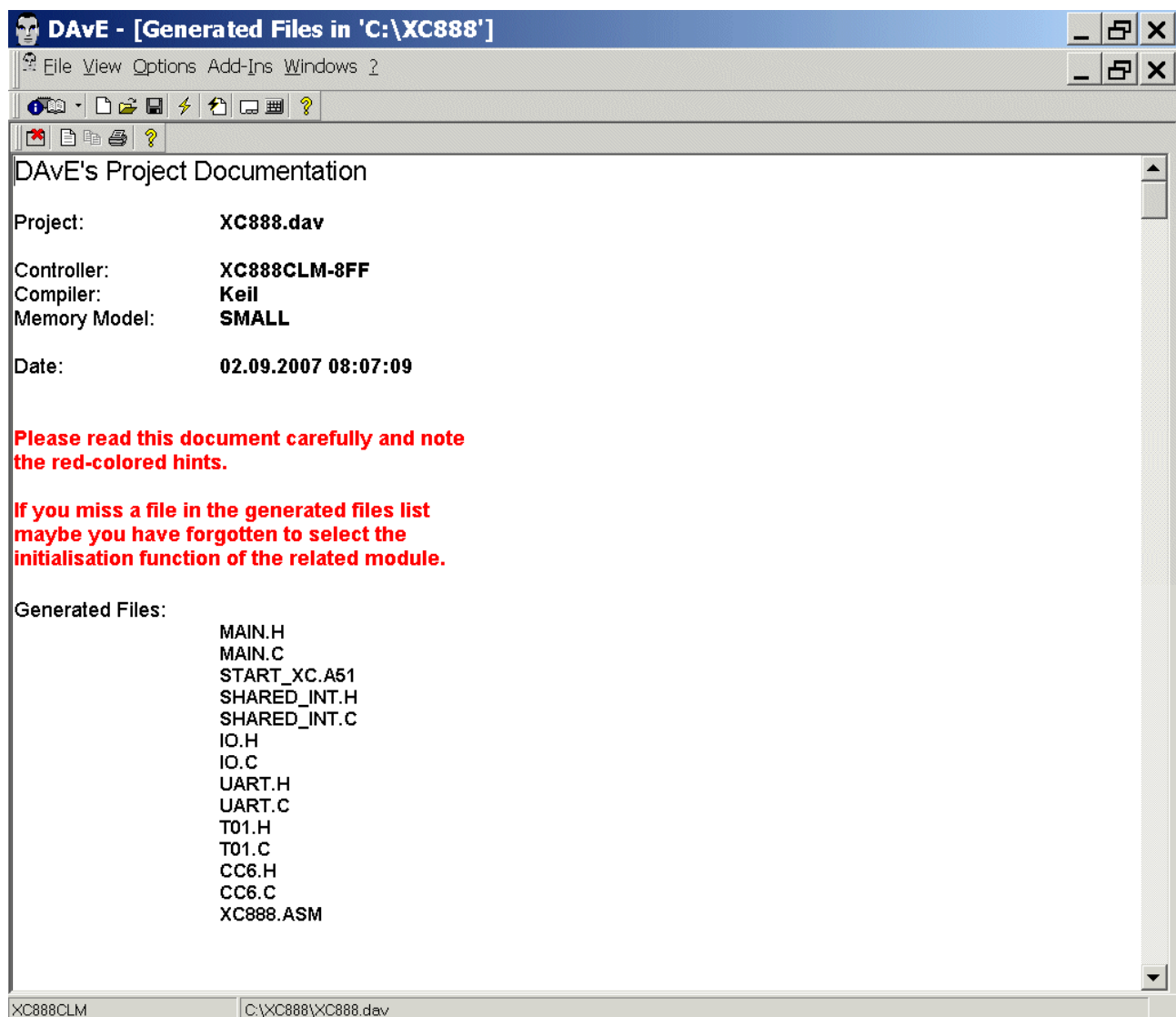
Exit this dialog now by clicking  the close button.

Generate Code:

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



DAvE will show you all the files he has generated
(Project Documentation opens automatically).



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



Start Keil μ Vision and open your Keil Project:

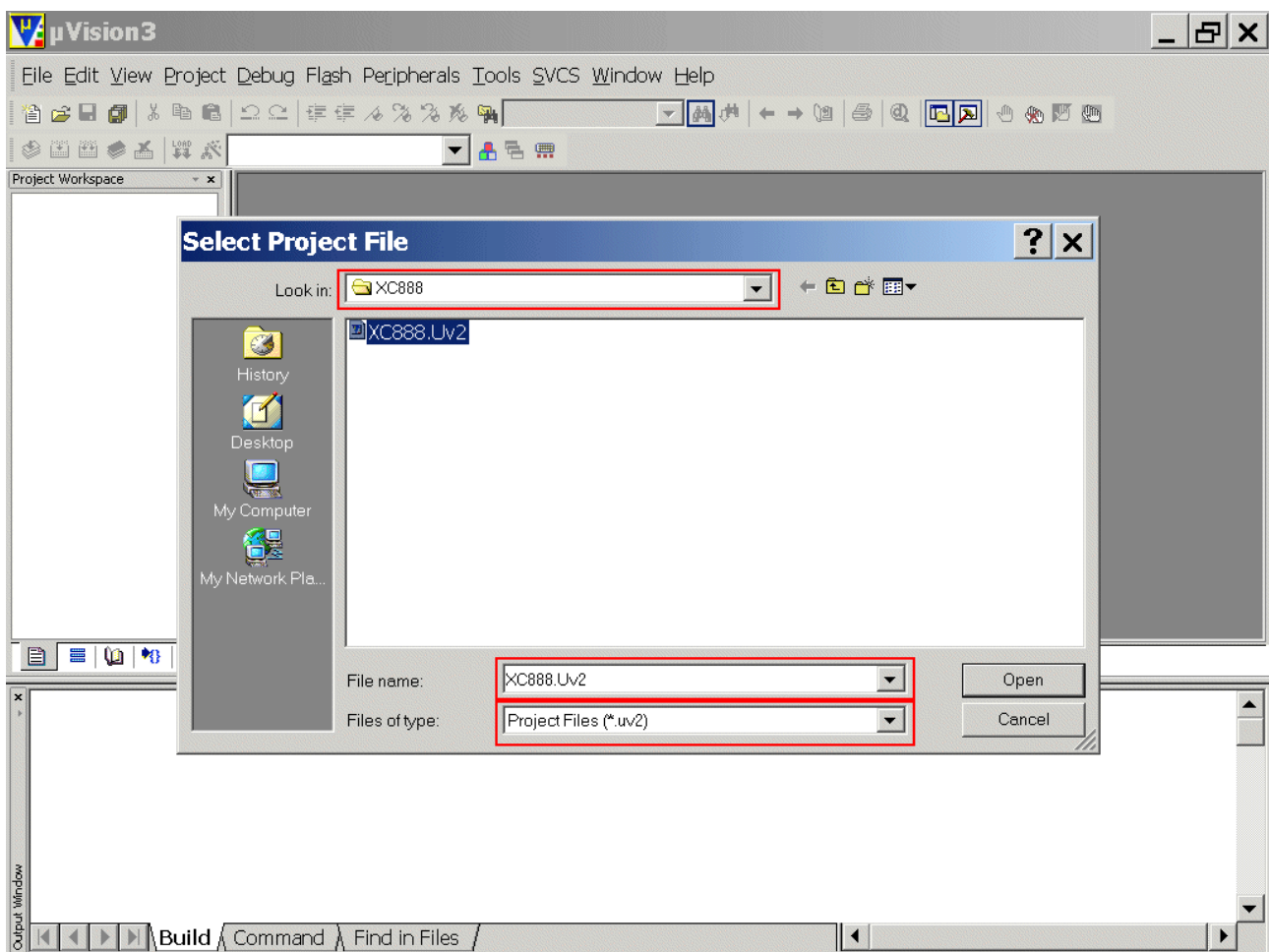
If you see an open project – close it: **Project - Close Project**

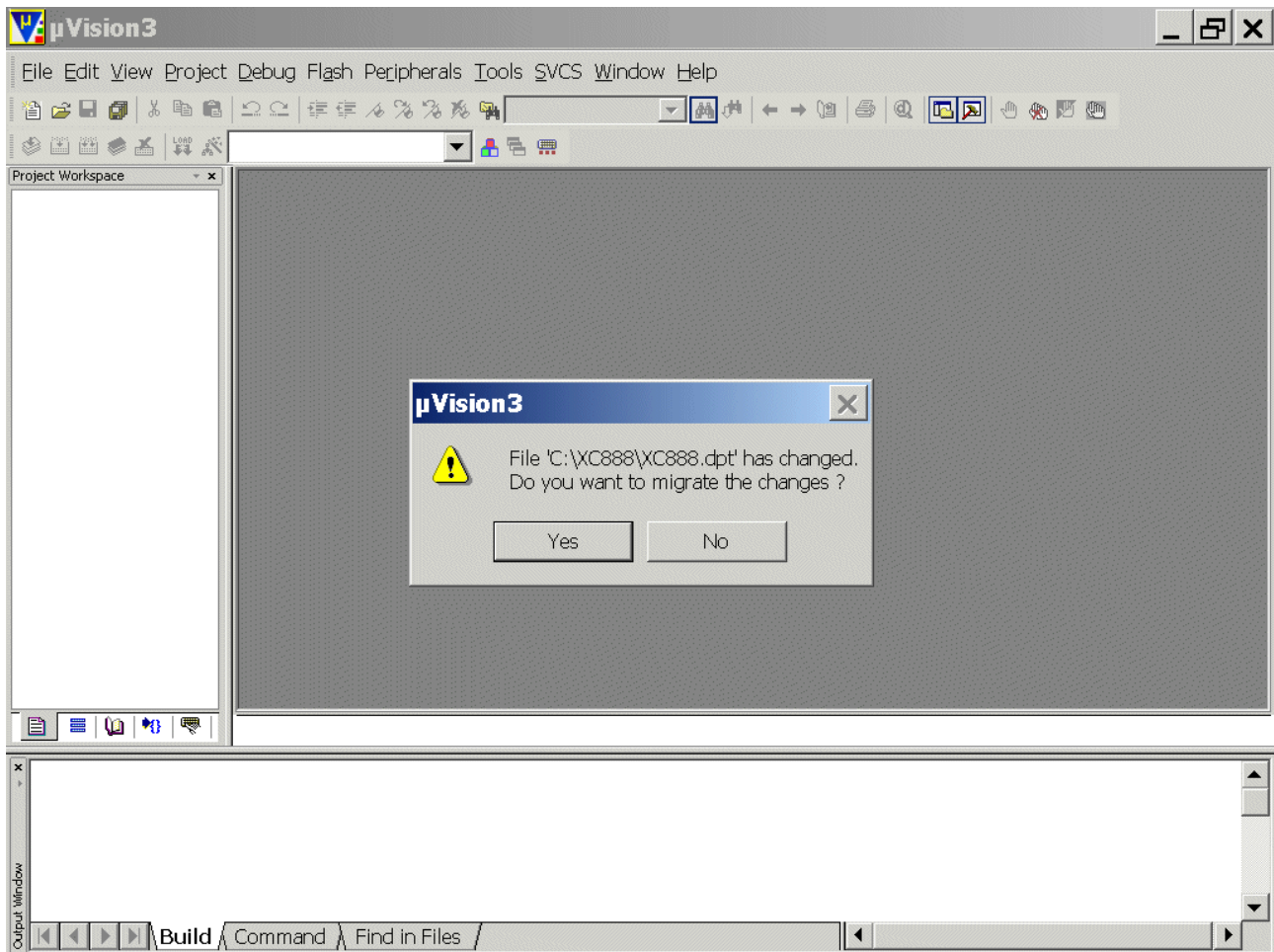
Project - Open Project

Select Project File: **Look in:** choose C:\XC888

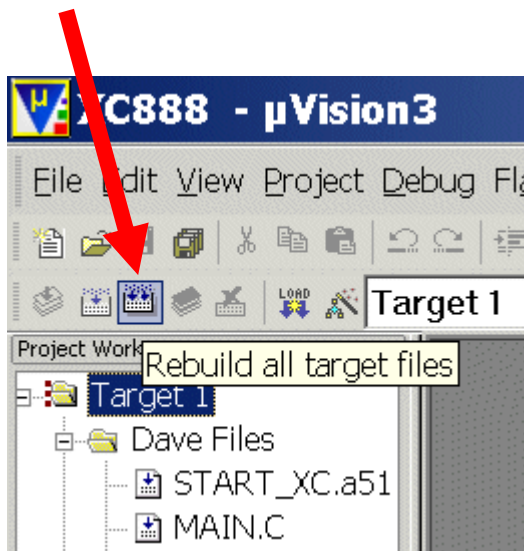
Select Project File: **Files of type:** choose Project Files (*.uv2)
choose XC888.Uv2

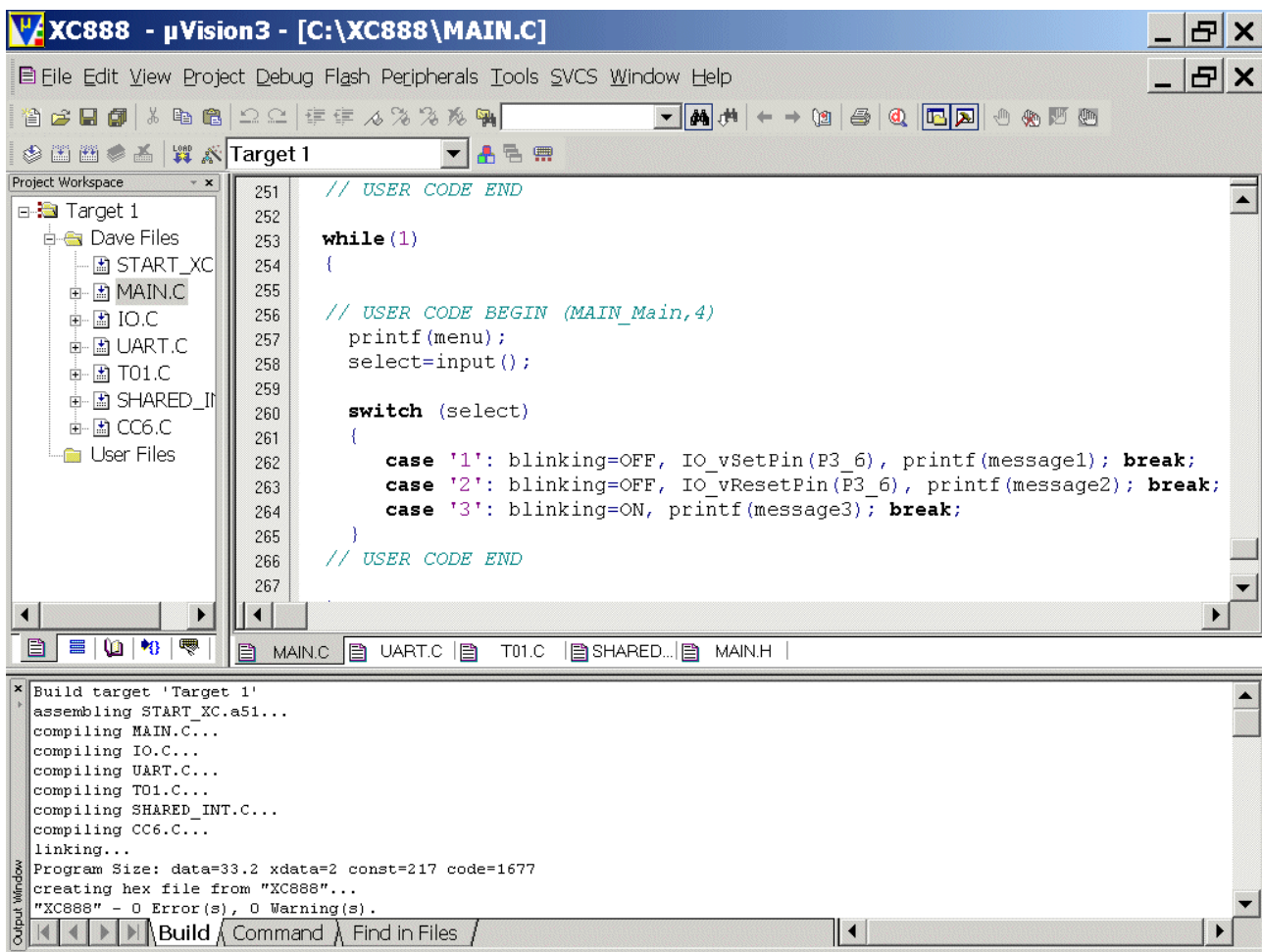
Open



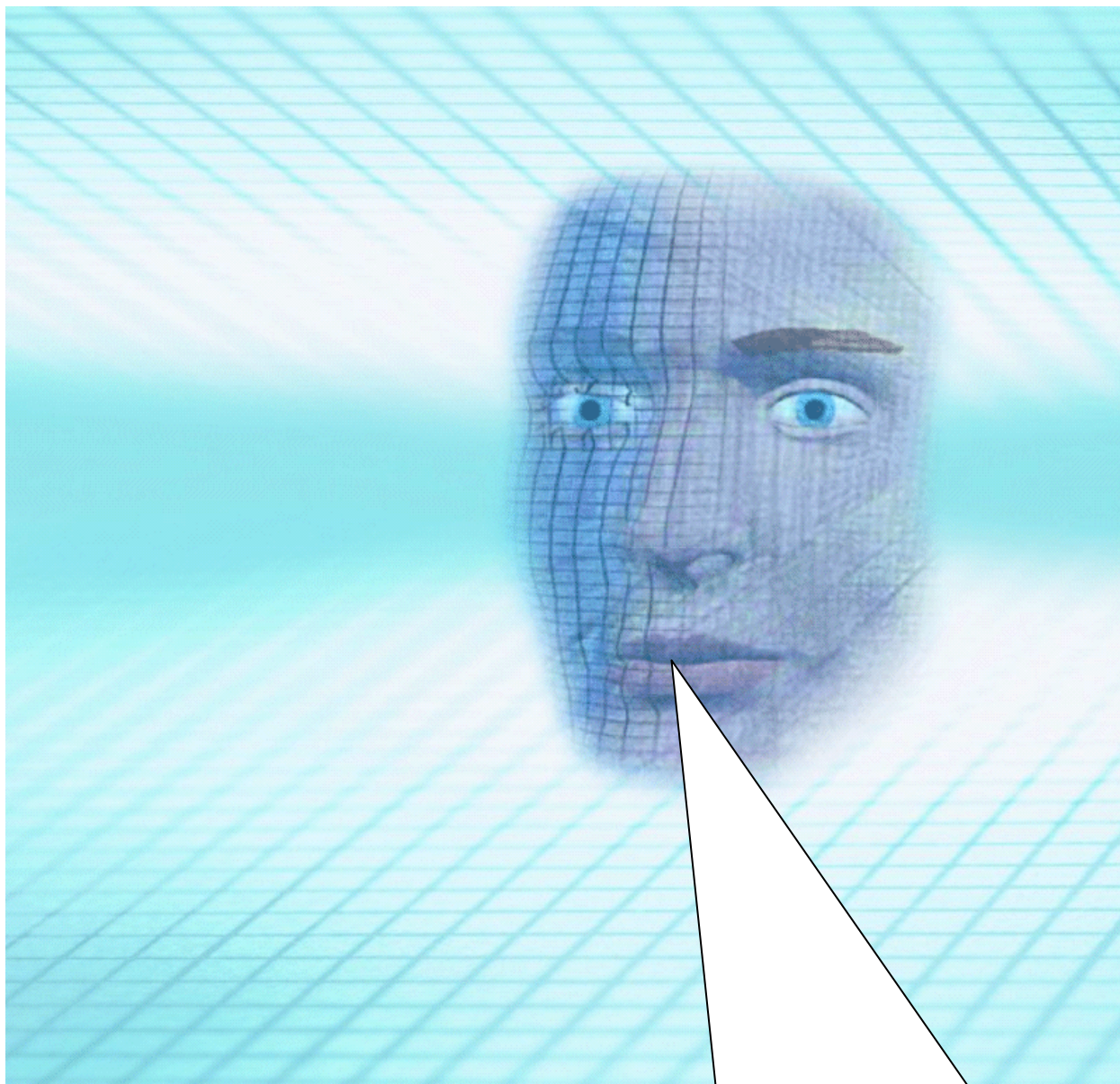


Click Yes

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



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!

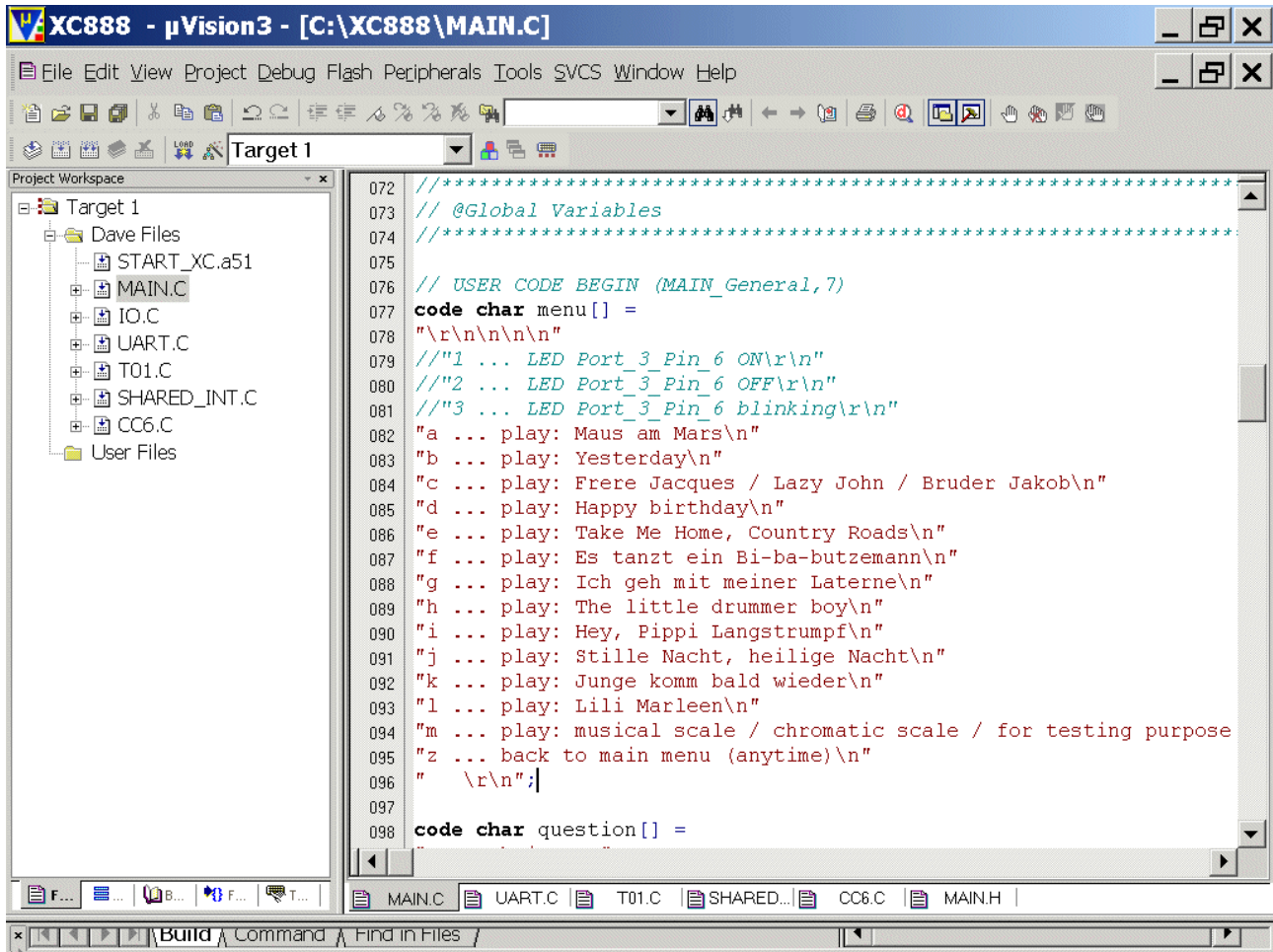
Double click **MAIN.C** and **change** Global Variable menu

from:

```
code char menu[] =
"\r\n\r\n\r\n"
"1 ... LED Port_3_Pin_6 ON\r\n"
"2 ... LED Port_3_Pin_6 OFF\r\n"
"3 ... LED Port_3_Pin_6 blinking\r\n"
" \r\n";
```

to:

```
code char menu[] =
"\r\n\r\n\r\n"
// "1 ... LED Port_3_Pin_6 ON\r\n"
// "2 ... LED Port_3_Pin_6 OFF\r\n"
// "3 ... LED Port_3_Pin_6 blinking\r\n"
"a ... play: Maus am Mars\r\n"
"b ... play: Yesterday\r\n"
"c ... play: Frere Jacques / Lazy John / Bruder Jakob\r\n"
"d ... play: Happy birthday\r\n"
"e ... play: Take Me Home, Country Roads\r\n"
"f ... play: Es tanzt ein Bi-ba-butzemann\r\n"
"g ... play: Ich geh mit meiner Laterne\r\n"
"h ... play: The little drummer boy\r\n"
"i ... play: Hey, Pippi Langstrumpf\r\n"
"j ... play: Stille Nacht, heilige Nacht\r\n"
"k ... play: Junge komm bald wieder\r\n"
"l ... play: Lili Marleen\r\n"
"m ... play: musical scale / chromatic scale / for testing purpose / Tonleiter\r\n"
"z ... back to main menu (anytime)\r\n"
" \r\n";
```



The screenshot shows the µVision3 IDE interface for the XC888 project. The Project Workspace on the left lists the files: Target 1, Dave Files, START_XC.a51, MAIN.C, IO.C, UART.C, T01.C, SHARED_INT.C, CC6.C, and User Files. The main editor displays the code for MAIN.C, which includes global variables, a menu array, and a question array. The code is as follows:

```

072 //*****
073 // @Global Variables
074 //*****
075
076 // USER CODE BEGIN (MAIN_General,7)
077 code char menu[] =
078 "\r\n\r\n\r\n"
079 //"1 ... LED Port_3_Pin_6 ON\r\n"
080 //"2 ... LED Port_3_Pin_6 OFF\r\n"
081 //"3 ... LED Port_3_Pin_6 blinking\r\n"
082 "a ... play: Maus am Mars\r\n"
083 "b ... play: Yesterday\r\n"
084 "c ... play: Frere Jacques / Lazy John / Bruder Jakob\r\n"
085 "d ... play: Happy birthday\r\n"
086 "e ... play: Take Me Home, Country Roads\r\n"
087 "f ... play: Es tanzt ein Bi-ba-butzemann\r\n"
088 "g ... play: Ich geh mit meiner Laterne\r\n"
089 "h ... play: The little drummer boy\r\n"
090 "i ... play: Hey, Pippi Langstrumpf\r\n"
091 "j ... play: Stille Nacht, heilige Nacht\r\n"
092 "k ... play: Junge komm bald wieder\r\n"
093 "l ... play: Lili Marleen\r\n"
094 "m ... play: musical scale / chromatic scale / for testing purpose
095 "z ... back to main menu (anytime)\r\n"
096 " \r\n";
097
098 code char question[] =

```

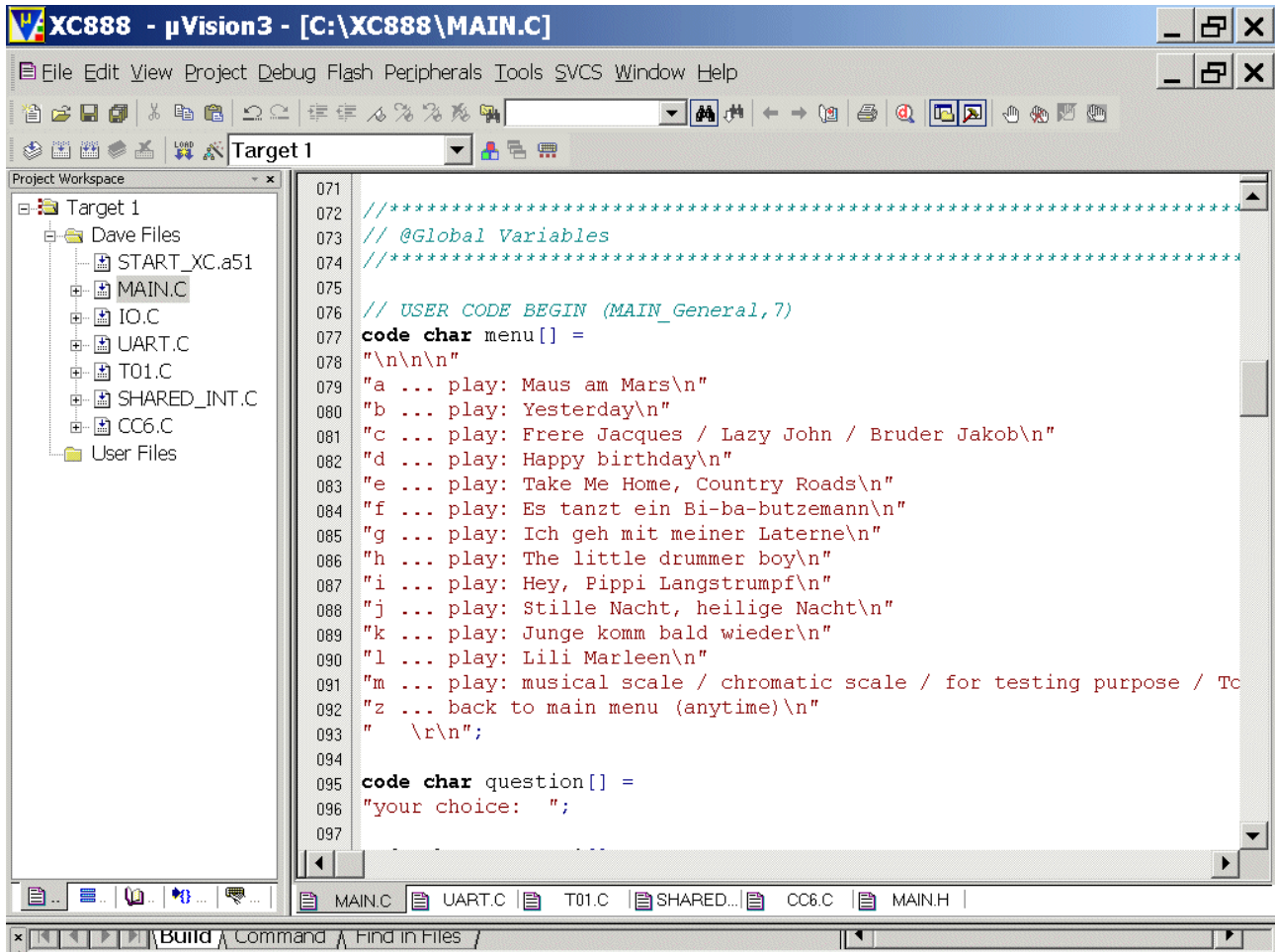

Double click **MAIN.C** and **change** Global Variable menu

from:

```
code char menu[] =
"\r\n\n\n"
// "1 ... LED Port_3_Pin_6 ON\r\n"
// "2 ... LED Port_3_Pin_6 OFF\r\n"
// "3 ... LED Port_3_Pin_6 blinking\r\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"
" \r\n";
```

to:

```
code char menu[] =
"\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"
" \r\n";
```



The screenshot shows the µVision3 IDE interface. The title bar reads "XC888 - µVision3 - [C:\XC888\MAIN.C]". The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The Project Workspace on the left shows a tree structure for "Target 1" with folders "Dave Files" and "User Files". Under "Dave Files", there are files: "START_XC.a51", "MAIN.C" (selected), "IO.C", "UART.C", "T01.C", "SHARED_INT.C", and "CC6.C". The main editor window displays the source code for "MAIN.C" with line numbers 071 to 097. The code includes comments for global variables and a menu of songs to play. The status bar at the bottom shows "Build", "Command", and "Find in Files" tabs.

```

071
072 //*****
073 // @Global Variables
074 //*****
075
076 // USER CODE BEGIN (MAIN_General,7)
077 code char menu[] =
078 "\n\n\n"
079 "a ... play: Maus am Mars\n"
080 "b ... play: Yesterday\n"
081 "c ... play: Frere Jacques / Lazy John / Bruder Jakob\n"
082 "d ... play: Happy birthday\n"
083 "e ... play: Take Me Home, Country Roads\n"
084 "f ... play: Es tanzt ein Bi-ba-butzemann\n"
085 "g ... play: Ich geh mit meiner Laterne\n"
086 "h ... play: The little drummer boy\n"
087 "i ... play: Hey, Pippi Langstrumpf\n"
088 "j ... play: Stille Nacht, heilige Nacht\n"
089 "k ... play: Junge komm bald wieder\n"
090 "l ... play: Lili Marleen\n"
091 "m ... play: musical scale / chromatic scale / for testing purpose / To
092 "z ... back to main menu (anytime)\n"
093 "  \r\n";
094
095 code char question[] =
096 "your choice: ";
097

```

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

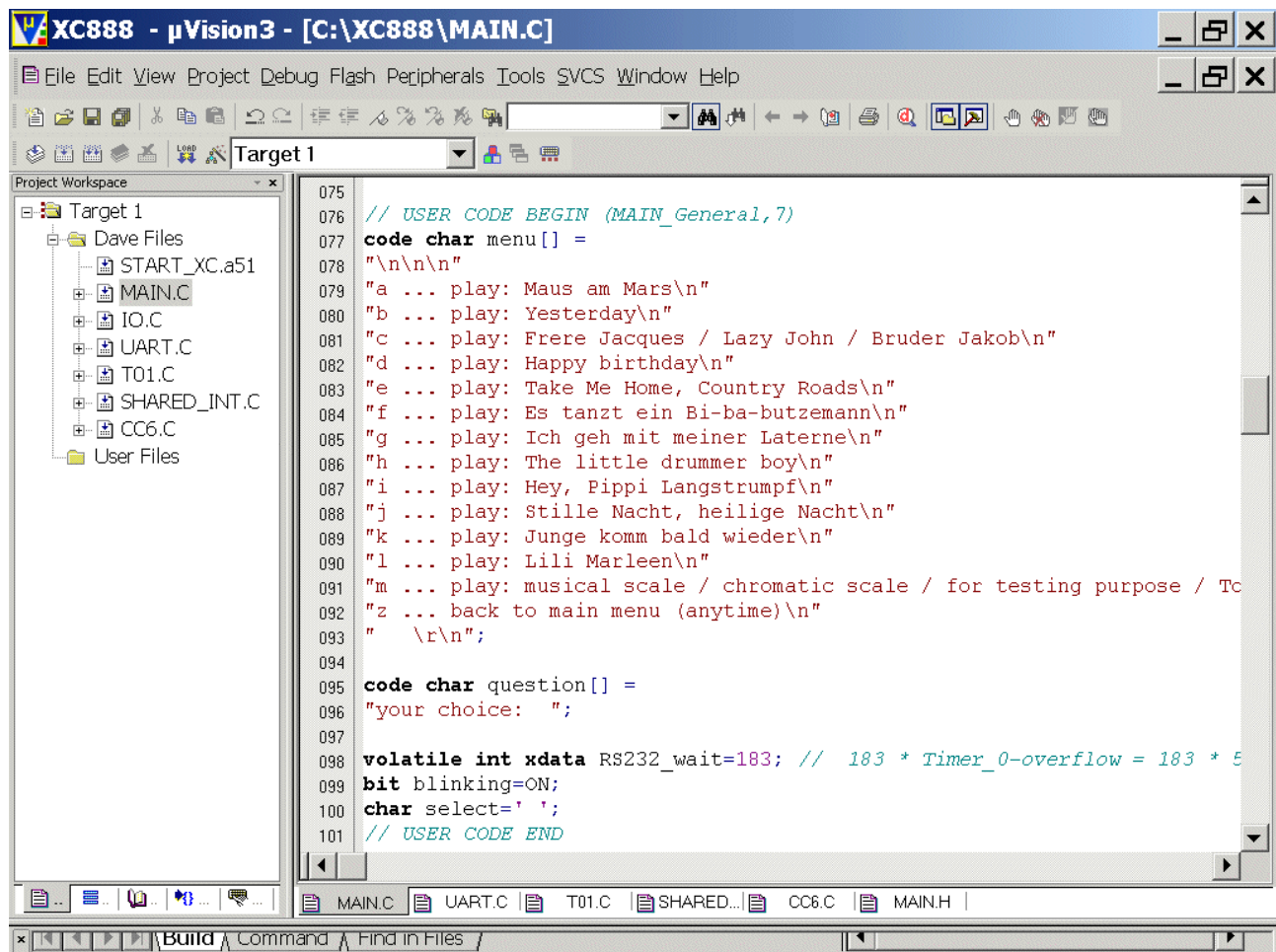
from:

```
code char message1[] =
"\n*** LED Port_3_Pin_6 ON ***\r\n";

code char message2[] =
"\n*** LED Port_3_Pin_6 OFF ***\r\n";

code char message3[] =
"\n*** LED Port_3_Pin_6 BLINKING ***\r\n";
```

to:



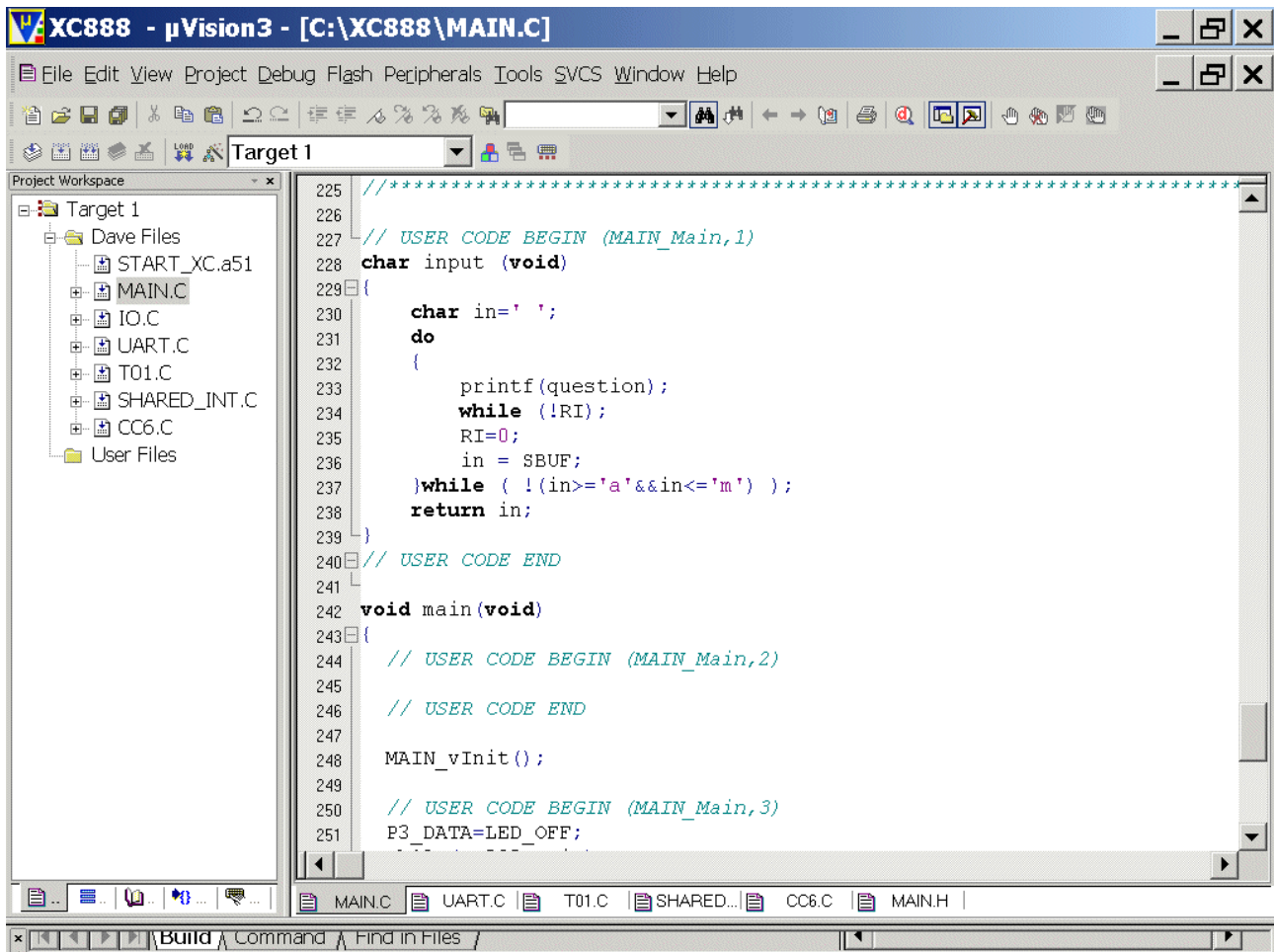
Double click **MAIN.C** and **change** function "char input (void)":

from

```
char input (void)
{
    char in=' ';
    do
    {
        printf(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;
}
```

XC888 - µVision3 - [C:\XC888\MAIN.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
 - Dave Files
 - START_XC.a51
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C
 - SHARED_INT.C
 - CC6.C
 - User Files

```

225 //*****
226
227 // USER CODE BEGIN (MAIN_Main,1)
228 char input (void)
229 {
230     char in=' ';
231     do
232     {
233         printf(question);
234         while (!RI);
235         RI=0;
236         in = SBUF;
237     }while ( !(in>='a'&&in<='m') );
238     return in;
239 }
240 // USER CODE END
241
242 void main(void)
243 {
244     // USER CODE BEGIN (MAIN_Main,2)
245
246     // USER CODE END
247
248     MAIN_vInit();
249
250     // USER CODE BEGIN (MAIN_Main,3)
251     P3_DATA=LED_OFF;
  
```

MAIN.C | UART.C | T01.C | SHARED... | CC6.C | MAIN.H

Build | Command | Find in Files

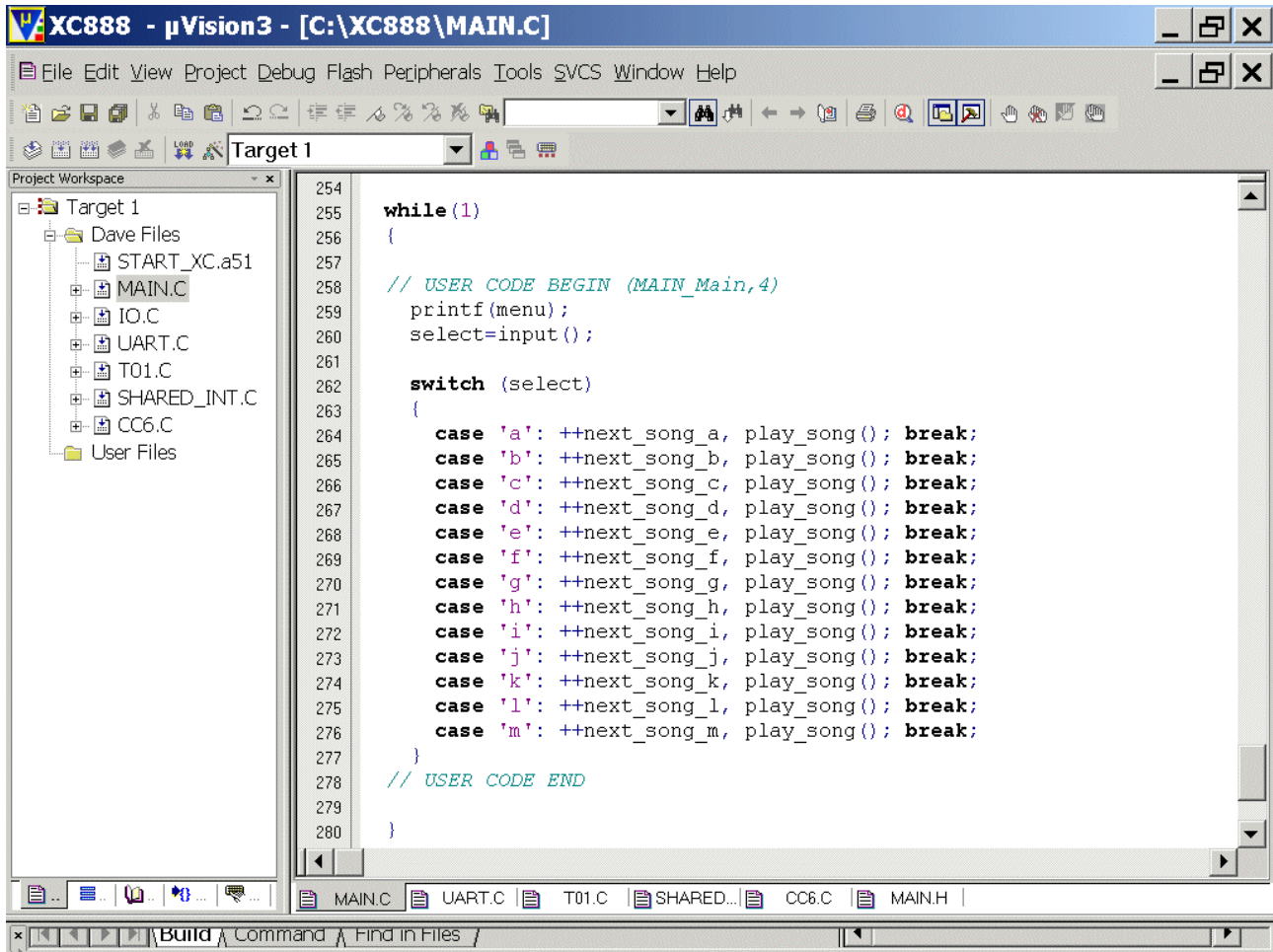
Double click **MAIN.C** and extend/change [“switch/case” in void main (void)] :

from:

```
switch (select)
{
    case '1': blinking=OFF, IO_vSetPin(P3_6), printf(message1); break;
    case '2': blinking=OFF, IO_vResetPin(P3_6), printf(message2); break;
    case '3': blinking=ON, printf(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;
}
```

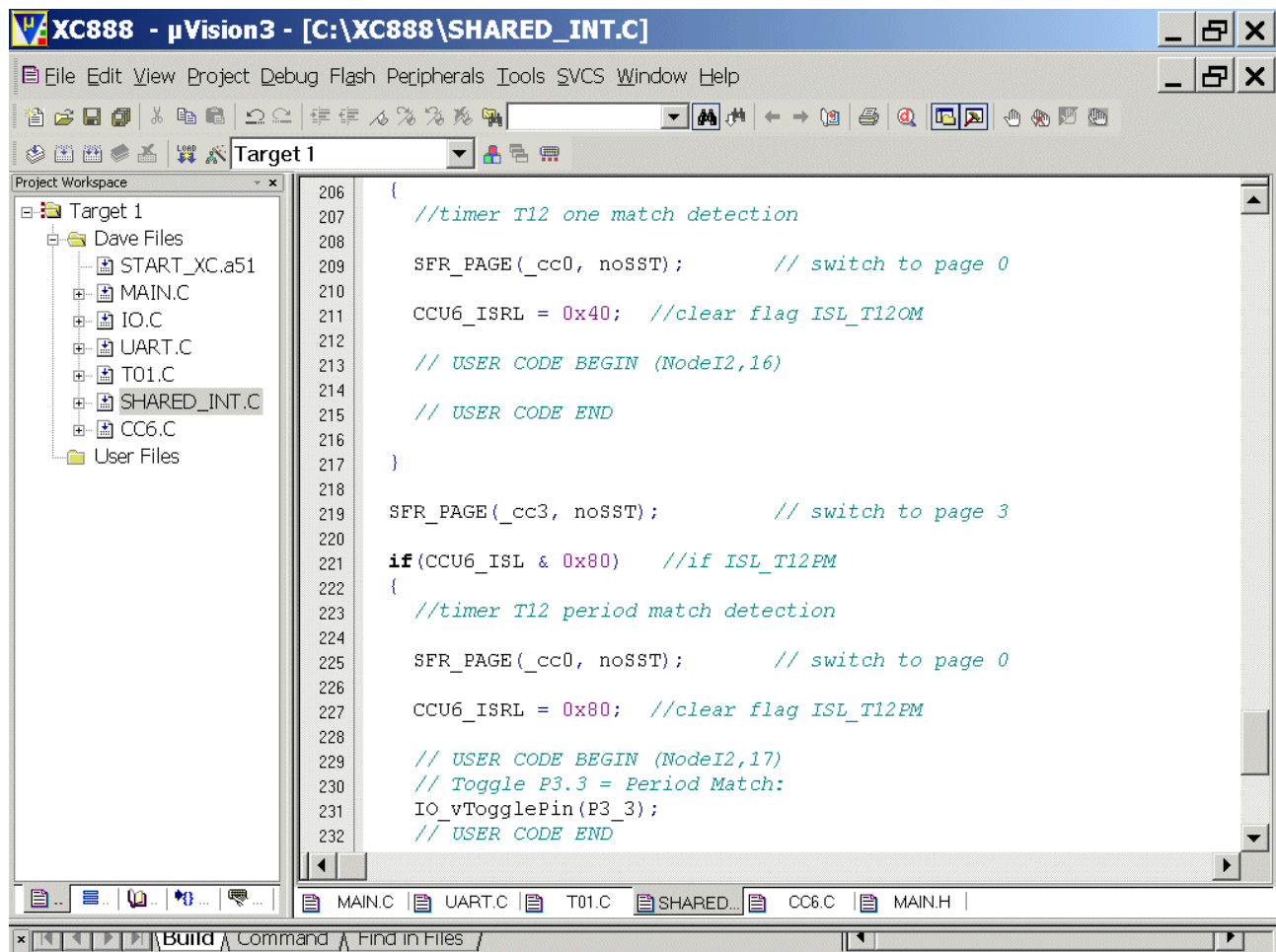


Double click **SHARED_INT.C** and **delete** code

from:

```
// Toggle P3.1 = One Match:
IO_vTogglePin(P3_1);
```

to:



The screenshot shows the µVision3 IDE interface. The title bar reads "XC888 - µVision3 - [C:\XC888\SHARED_INT.C]". The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations and development. The Project Workspace on the left shows a tree view with "Target 1" expanded, containing "Dave Files" (START_XC.a51, MAIN.C, IO.C, UART.C, T01.C, SHARED_INT.C, CC6.C) and "User Files". The main editor window displays the following code:

```

206 {
207     //timer T12 one match detection
208
209     SFR_PAGE(_cc0, nosST);      // switch to page 0
210
211     CCU6_ISRL = 0x40; //clear flag ISL_T12OM
212
213     // USER CODE BEGIN (NodeI2,16)
214
215     // USER CODE END
216
217 }
218
219 SFR_PAGE(_cc3, nosST);      // switch to page 3
220
221 if(CCU6_ISL & 0x80) //if ISL_T12PM
222 {
223     //timer T12 period match detection
224
225     SFR_PAGE(_cc0, nosST);      // switch to page 0
226
227     CCU6_ISRL = 0x80; //clear flag ISL_T12PM
228
229     // USER CODE BEGIN (NodeI2,17)
230     // Toggle P3.3 = Period Match:
231     IO_vTogglePin(P3_3);
232     // USER CODE END

```

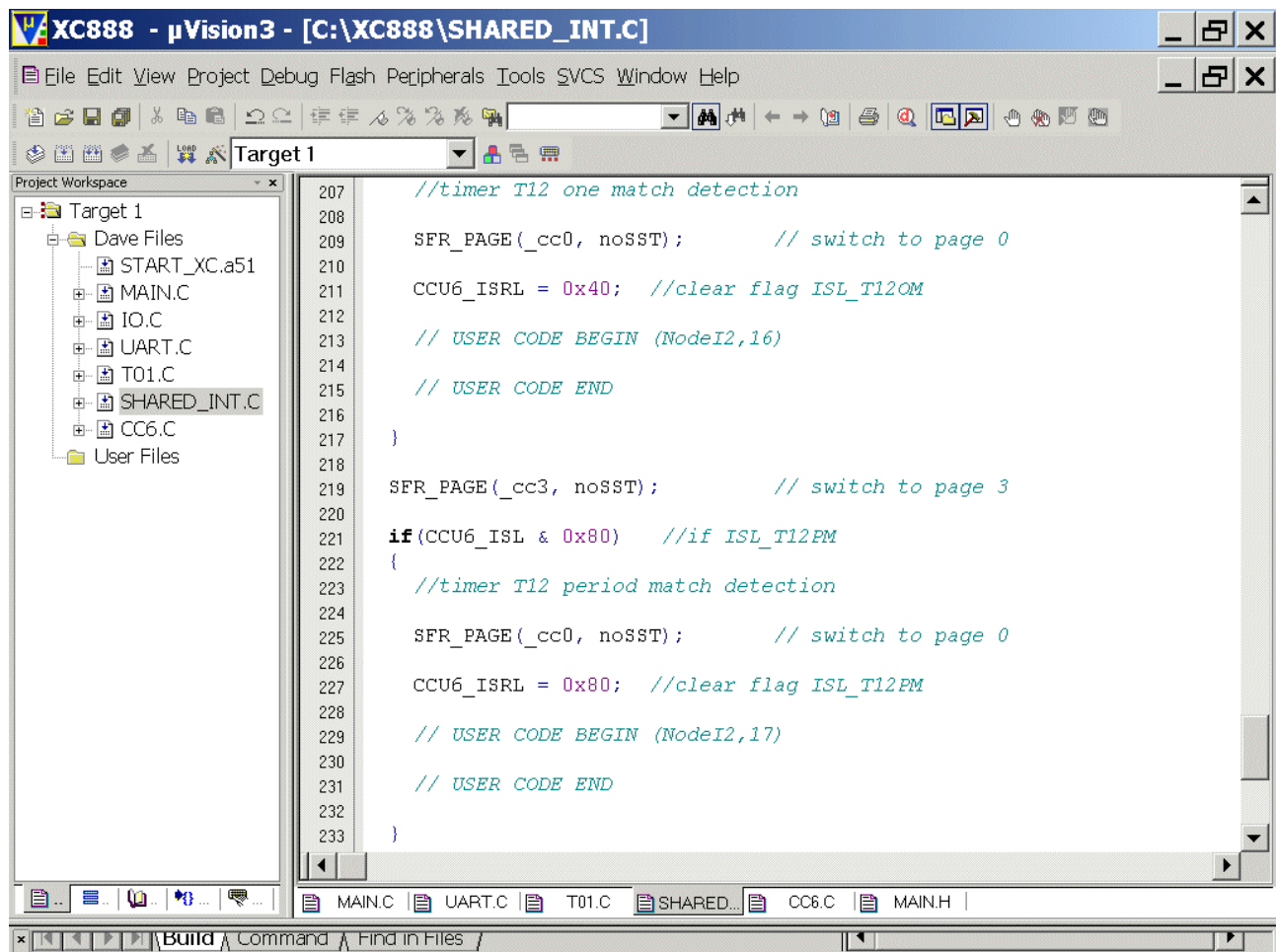
The status bar at the bottom shows "Build" and "Command" tabs, and a "Find in Files" search bar.

Double click **SHARED_INT.C** and **delete** code

from:

```
// Toggle P3.3 = Period Match:
IO_vTogglePin(P3_3);
```

to:



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

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

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)

Note (restrictions/limits):
The chosen octaves supported by the algorithm used
[read_song_string()] fit a descant recorder perfectly.
Therefore, to keep the programming example simple, notes below
octave 0 (00) are not supported!
Also, be aware that not every song sounds good on a descant
recorder.
*/

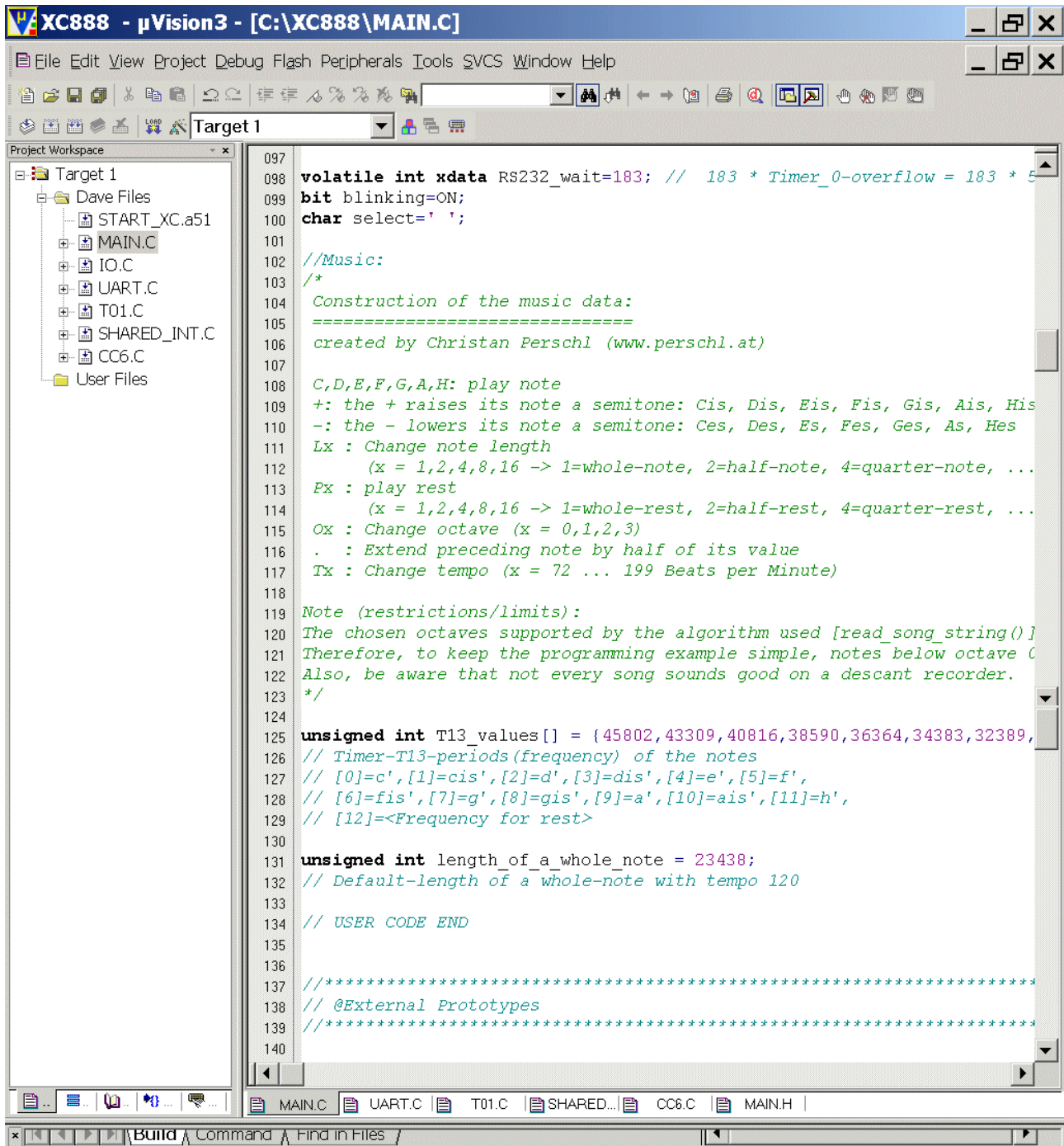
unsigned int T13_values[] =
{45802,43309,40816,38590,36364,34383,32389,30612,28943,27273,25782,24291,200};
// Timer-T13-periods(frequency) of the notes
// [0]=c', [1]=cis', [2]=d', [3]=dis', [4]=e', [5]=f',
// [6]=fis', [7]=g', [8]=gis', [9]=a', [10]=ais', [11]=h',
// [12]=<Frequency for rest>

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.





```

097
098 volatile int xdata RS232_wait=183; // 183 * Timer_0-overflow = 183 * 5
099 bit blinking=ON;
100 char select=' ';
101
102 //Music:
103 /*
104  Construction of the music data:
105  =====
106  created by Christan Perschl (www.perschl.at)
107
108  C,D,E,F,G,A,H: play note
109  +: the + raises its note a semitone: Cis, Dis, Eis, Fis, Gis, Ais, His
110  -: the - lowers its note a semitone: Ces, Des, Es, Fes, Ges, As, Hes
111  Lx : Change note length
112      (x = 1,2,4,8,16 -> 1=whole-note, 2=half-note, 4=quarter-note, ...
113  Px : play rest
114      (x = 1,2,4,8,16 -> 1=whole-rest, 2=half-rest, 4=quarter-rest, ...
115  Ox : Change octave (x = 0,1,2,3)
116  . : Extend preceding note by half of its value
117  Tx : Change tempo (x = 72 ... 199 Beats per Minute)
118
119  Note (restrictions/limits):
120  The chosen octaves supported by the algorithm used [read_song_string()]
121  Therefore, to keep the programming example simple, notes below octave 0
122  Also, be aware that not every song sounds good on a descant recorder.
123  */
124
125 unsigned int T13_values[] = {45802,43309,40816,38590,36364,34383,32389,
126 // Timer-T13-periods(frequency) of the notes
127 // [0]=c', [1]=cis', [2]=d', [3]=dis', [4]=e', [5]=f',
128 // [6]=fis', [7]=g', [8]=gis', [9]=a', [10]=ais', [11]=h',
129 // [12]=<Frequency for rest>
130
131 unsigned int length_of_a_whole_note = 23438;
132 // Default-length of a whole-note with tempo 120
133
134 // USER CODE END
135
136
137 //*****
138 // @External Prototypes
139 //*****
140

```



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

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[]="T18000L4ADF+DL2EL8GF+EDL4C+EAC+L2DF+L4ADF+DL2EL8GF+EDL4C+EAC
+DP4P2L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+EE
DC+DP4P2L2F+GAHO1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2GL4HAGF+EL2F+GL4
AF+GAL2HO1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2GL4HAGF+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.L8CL4EL4D.L8CL2C.L4EEL8D+L8EL4F+L4E.L8E
L4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+L8EL4EL2DL4EL4D.L8CL2C.L4D
DL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4EEL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8D
DDDDDDDDL4DP8L8DL8D+L8DDDDDL8D+L8DL4DP8L8DL8EEEEEL2GP8L8EL1DP8L8
DL8EEL4E.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.L8FL4FO1L4DL2CO0L4EL4G.L8FL4FL4CL2C.";
```

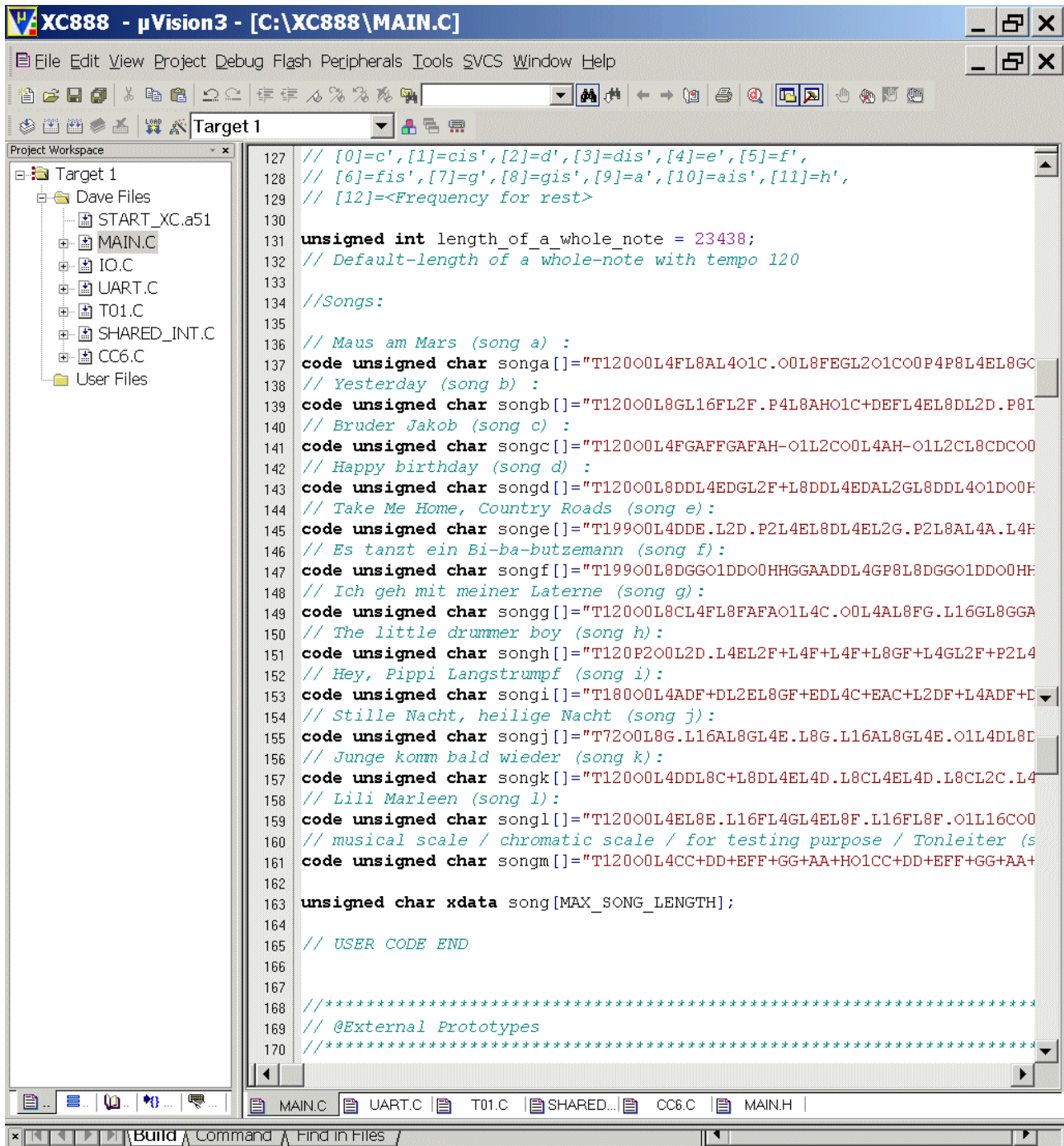
// musical scale / chromatic scale / for testing purpose / Tonleiter (song m) :

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];



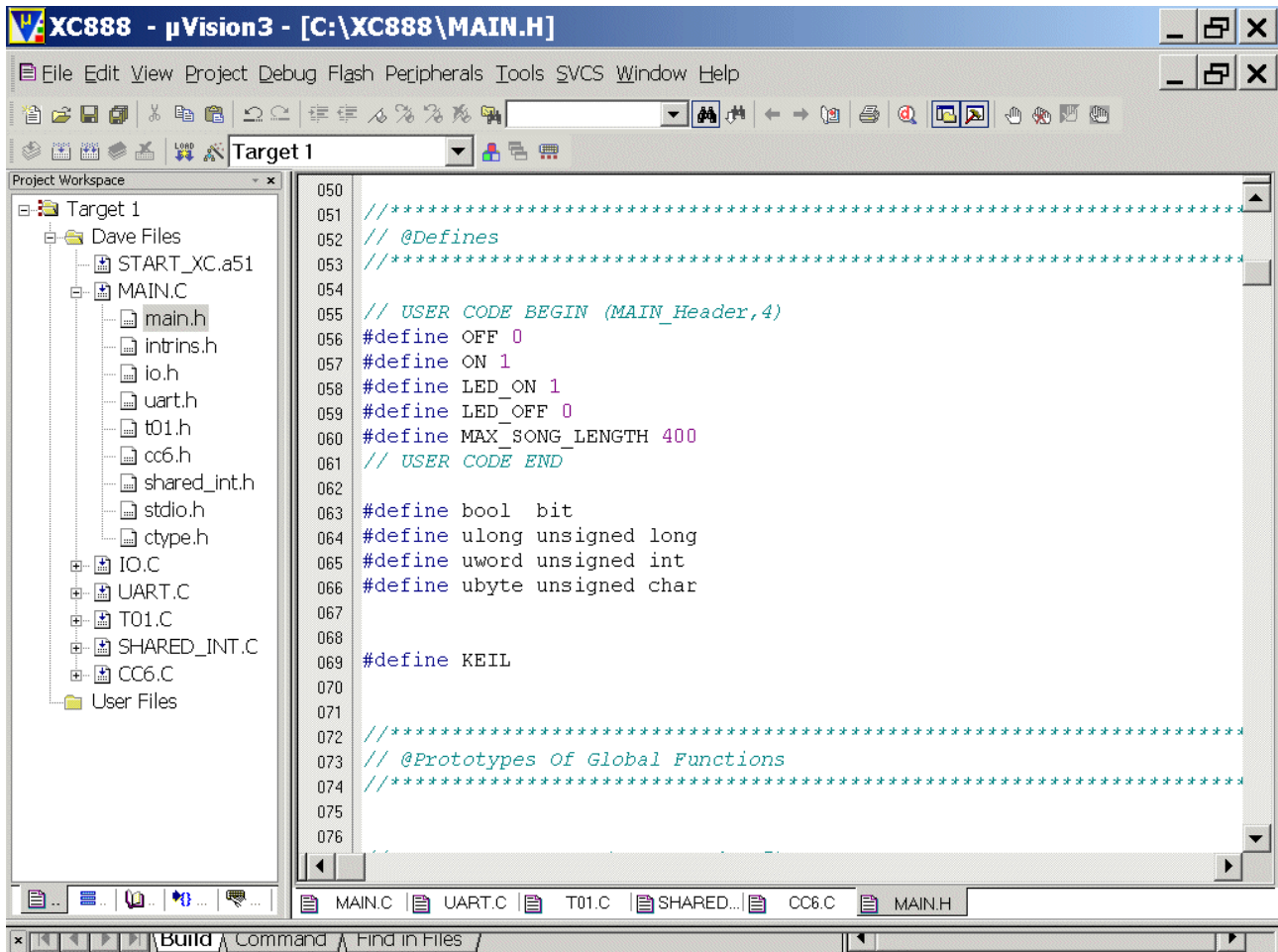


```

127 // [0]='c',[1]='cis',[2]='d',[3]='dis',[4]='e',[5]='f',
128 // [6]='fis',[7]='g',[8]='gis',[9]='a',[10]='ais',[11]='h',
129 // [12]='<Frequency for rest>'
130
131 unsigned int length_of_a_whole_note = 23438;
132 // Default-length of a whole-note with tempo 120
133
134 //Songs:
135
136 // Maus am Mars (song a) :
137 code unsigned char songa[]="T12000L4FL8AL4O1C.O0L8FEGL2O1CO0P4P8L4EL8GC
138 // Yesterday (song b) :
139 code unsigned char songb[]="T12000L8GL16FL2F.P4L8AHO1C+DEFL4EL8DL2D.P8L
140 // Bruder Jakob (song c) :
141 code unsigned char songc[]="T12000L4FGAFFGAFAH-O1L2CO0L4AH-O1L2CL8CDCO0
142 // Happy birthday (song d) :
143 code unsigned char songd[]="T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0H
144 // Take Me Home, Country Roads (song e):
145 code unsigned char songe[]="T19900L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4F
146 // Es tanzt ein Bi-ba-butzemann (song f):
147 code unsigned char songf[]="T19900L8DGG01DDO0HHGGAADDL4GP8L8DGG01DDO0H
148 // Ich geh mit meiner Laterne (song g):
149 code unsigned char songg[]="T12000L8CL4FL8FAFAO1L4C.O0L4AL8FG.L16GL8GGA
150 // The little drummer boy (song h):
151 code unsigned char songh[]="T120F2O0L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4
152 // Hey, Pippi Langstrumpf (song i):
153 code unsigned char songi[]="T18000L4ADF+DL2EL8GF+EDL4C+EAC+L2DF+L4ADF+D
154 // Stille Nacht, heilige Nacht (song j):
155 code unsigned char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8E
156 // Junge komm bald wieder (song k):
157 code unsigned char songk[]="T12000L4DDL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4
158 // Lili Marleen (song l):
159 code unsigned char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0
160 // musical scale / chromatic scale / for testing purpose / Tonleiter (s
161 code unsigned char songm[]="T12000L4CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+AA+
162
163 unsigned char xdata song[MAX_SONG_LENGTH];
164
165 // USER CODE END
166
167
168 //*****
169 // @External Prototypes
170 //*****
  
```

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

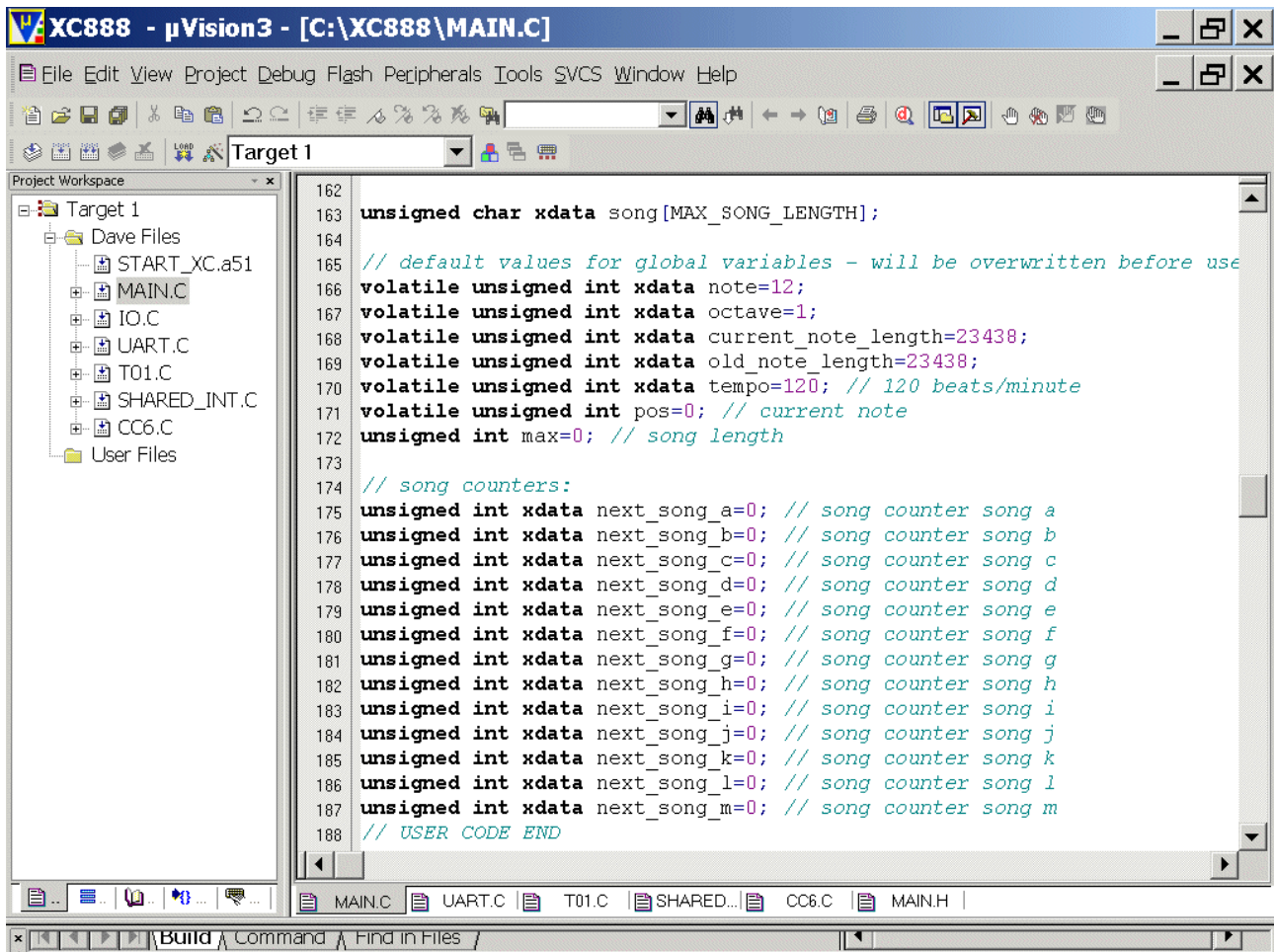
```
#define MAX_SONG_LENGTH 400
```



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
```

XC888 - µVision3 - [C:\XC888\MAIN.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
 - Dave Files
 - START_XC.a51
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C
 - SHARED_INT.C
 - CC6.C
 - User Files

```

162
163 unsigned char xdata song[MAX_SONG_LENGTH];
164
165 // default values for global variables - will be overwritten before use
166 volatile unsigned int xdata note=12;
167 volatile unsigned int xdata octave=1;
168 volatile unsigned int xdata current_note_length=23438;
169 volatile unsigned int xdata old_note_length=23438;
170 volatile unsigned int xdata tempo=120; // 120 beats/minute
171 volatile unsigned int pos=0; // current note
172 unsigned int max=0; // song length
173
174 // song counters:
175 unsigned int xdata next_song_a=0; // song counter song a
176 unsigned int xdata next_song_b=0; // song counter song b
177 unsigned int xdata next_song_c=0; // song counter song c
178 unsigned int xdata next_song_d=0; // song counter song d
179 unsigned int xdata next_song_e=0; // song counter song e
180 unsigned int xdata next_song_f=0; // song counter song f
181 unsigned int xdata next_song_g=0; // song counter song g
182 unsigned int xdata next_song_h=0; // song counter song h
183 unsigned int xdata next_song_i=0; // song counter song i
184 unsigned int xdata next_song_j=0; // song counter song j
185 unsigned int xdata next_song_k=0; // song counter song k
186 unsigned int xdata next_song_l=0; // song counter song l
187 unsigned int xdata next_song_m=0; // song counter song m
188 // USER CODE END
  
```

MAIN.C | UART.C | T01.C | SHARED... | CC6.C | MAIN.H

Build | Command | Find in Files



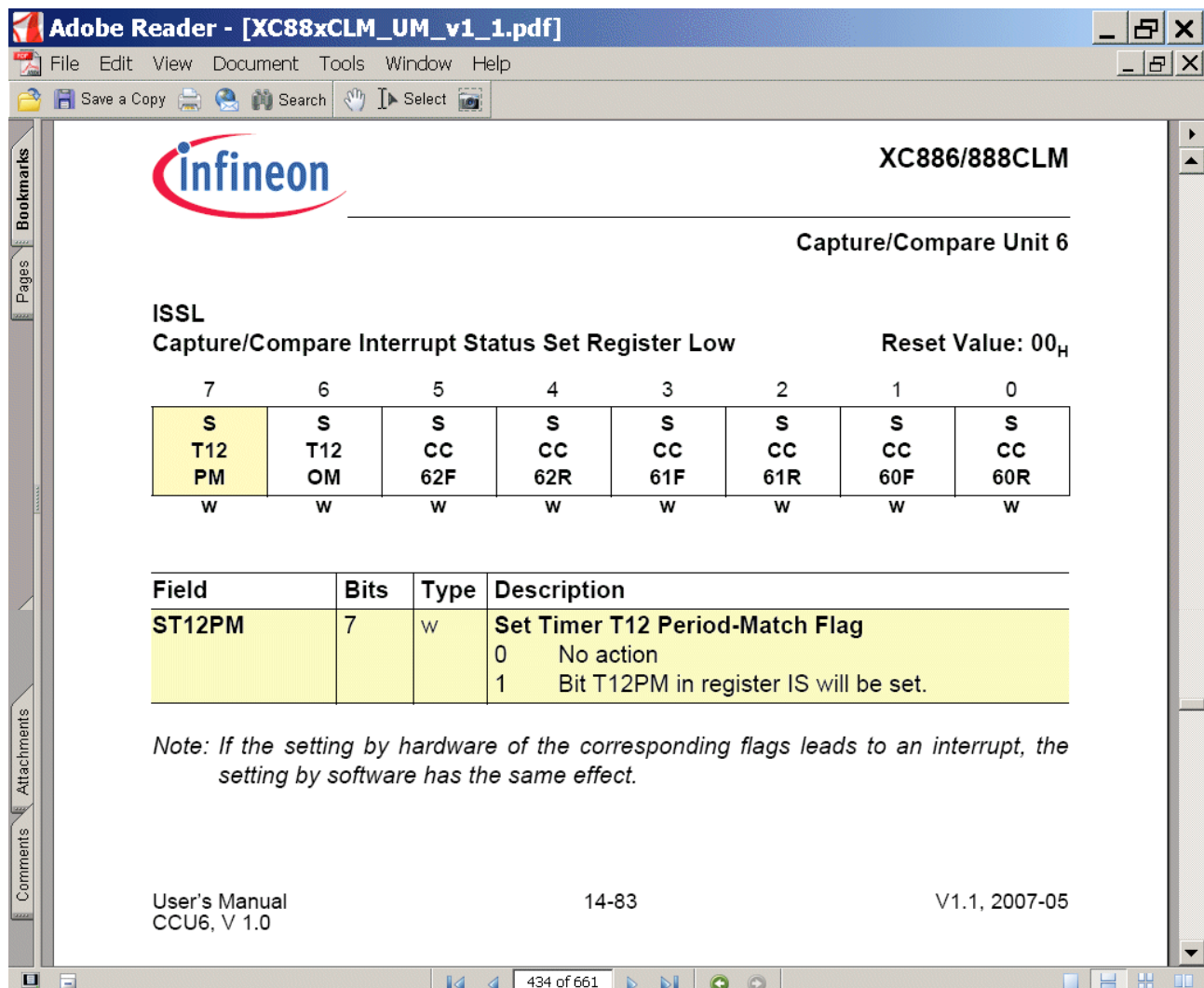
Note:

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).



XC886/888CLM

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
CCU6, V 1.0

14-83

V1.1, 2007-05



Note:

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

Table 14-3 SFR Address List for Pages 0-3

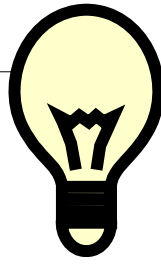
Address	Page 0	Page 1	Page 2	Page 3
9A _H	CC63SRL	CC63RL	T12MSELL	MCMOUTL
9B _H	CC63SRH	CC63RH	T12MSELH	MCMOUTH
9C _H	TCTR4L	T12PRL	IENL	ISL
9D _H	TCTR4H	T12PRH	IENH	ISH
9E _H	MCMOUTSL	T13PRL	INPL	PISEL0L
9F _H	MCMOUTSH	T13PRH	INPH	PISEL0H
A4 _H	ISRL	T12DTCL	ISSL	PISEL2
A5 _H	ISRH	T12DTCH	ISSH	
A6 _H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

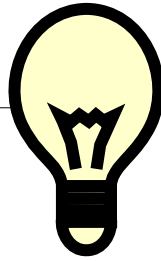


Address Extension:

Note:

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

- .) Mapping
- .) Paging



Address Extension by Mapping:

Note (Source: User's Manual):

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

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

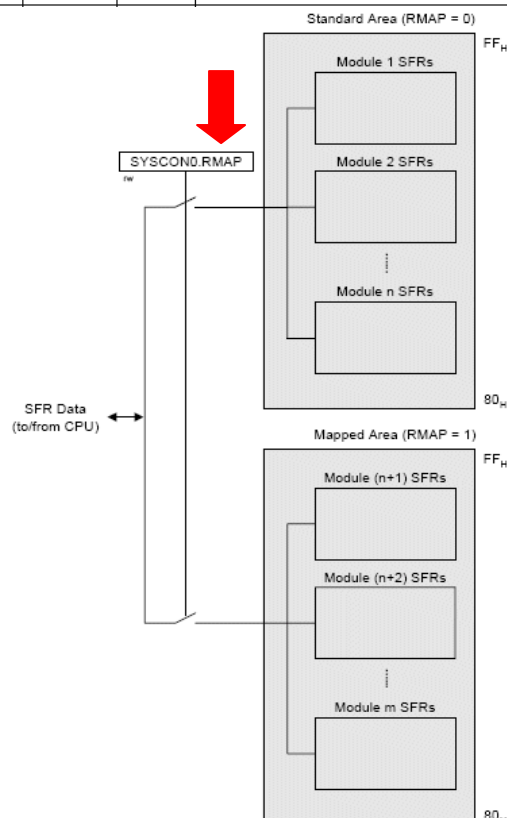
To access SFRs in the mapped area, bit RMAP in SFR SYSCON0 must be set.

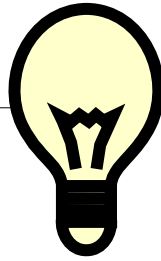
The SFRs in the standard area can be accessed by clearing bit RMAP.

As long as bit RMAP is set, the mapped SFR area can be accessed. This bit is not cleared automatically by hardware. Thus, before standard/mapped registers are accessed, bit RMAP must be cleared/set, respectively, by software.

SYSCON0 System Control Register 0							Reset Value: 04 _H
7	6	5	4	3	2	1	0
	0		IMODE	0	1	0	RMAP
	r		rw	r	r	r	rw

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





Address Extension by Paging:

Note (Source: User's Manual):

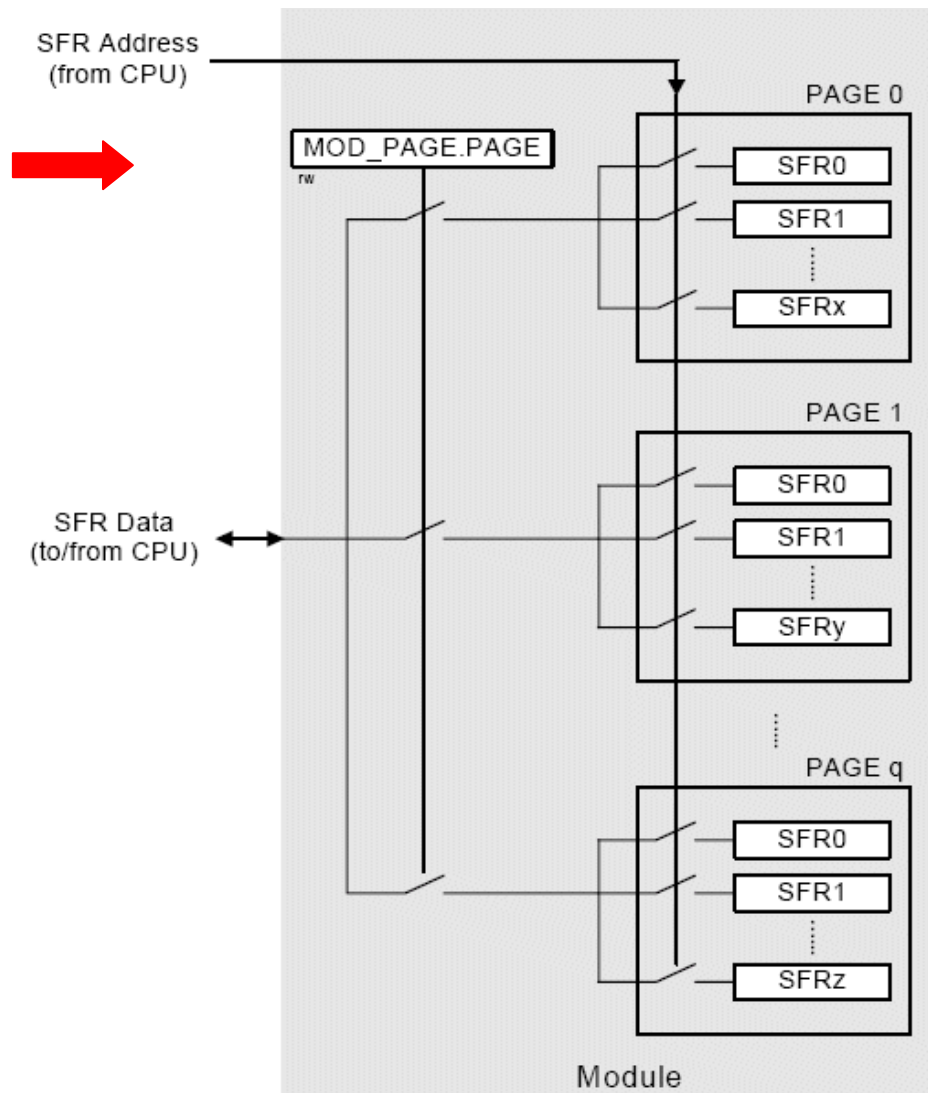
Address extension is further performed at the module level by paging. With the address extension by mapping, the XC886/888 has a 256-SFR address range. However, this is still less than the total number of SFRs needed by the on-chip peripherals.

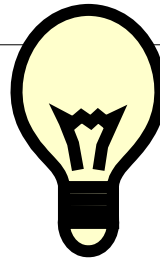
To meet this requirement, some peripherals ([Parallel Ports](#), [Analog-to-Digital Converter](#), [Capture/Compare Unit 6](#), [System Control Registers](#)) have a built-in local address extension mechanism for increasing the number of addressable SFRs. The extended address range is not directly controlled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

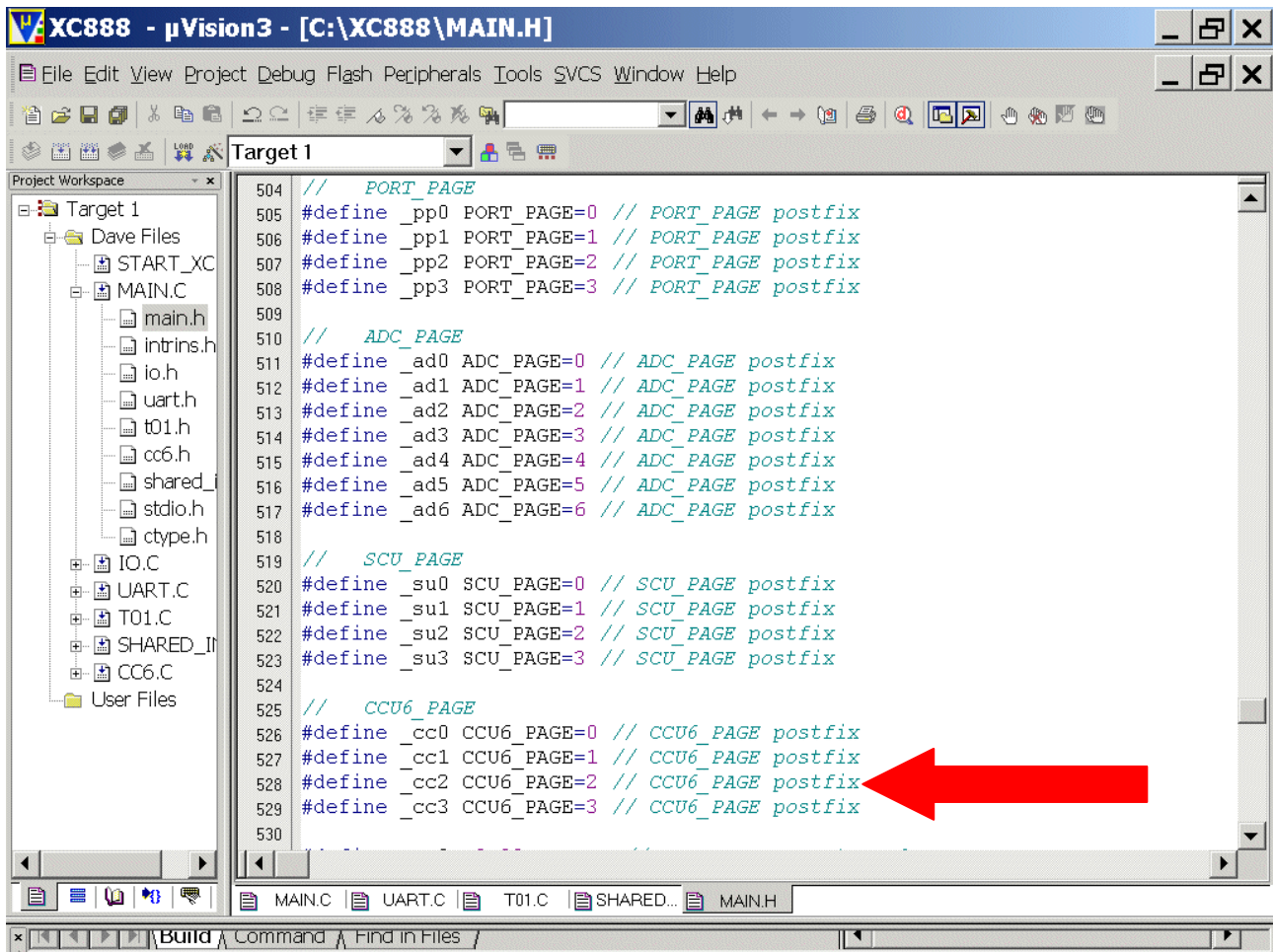
Note (Source: Application Note AP08053):

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:



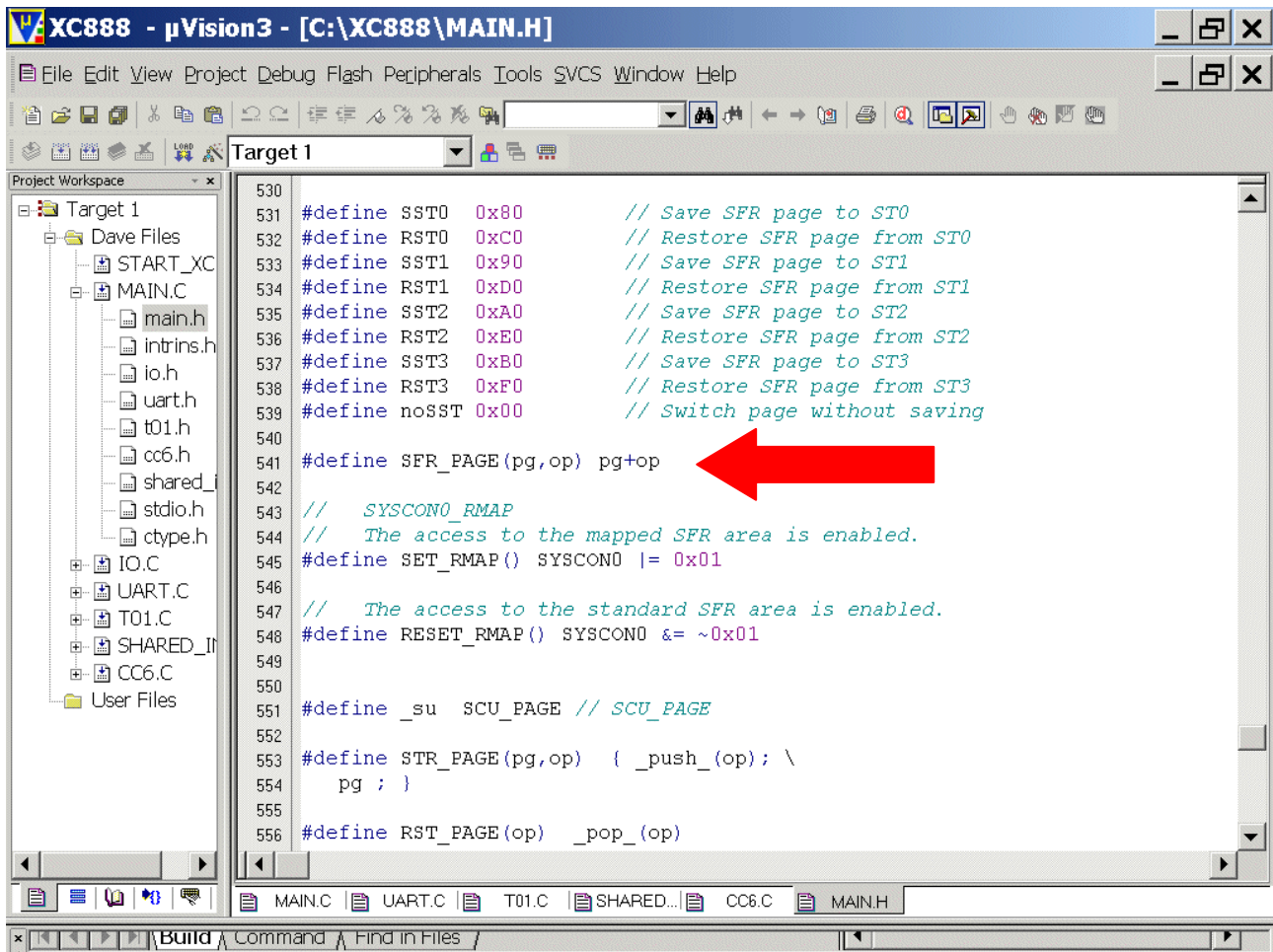
```

504 // PORT_PAGE
505 #define _pp0 PORT_PAGE=0 // PORT_PAGE postfix
506 #define _pp1 PORT_PAGE=1 // PORT_PAGE postfix
507 #define _pp2 PORT_PAGE=2 // PORT_PAGE postfix
508 #define _pp3 PORT_PAGE=3 // PORT_PAGE postfix
509
510 // ADC_PAGE
511 #define _ad0 ADC_PAGE=0 // ADC_PAGE postfix
512 #define _ad1 ADC_PAGE=1 // ADC_PAGE postfix
513 #define _ad2 ADC_PAGE=2 // ADC_PAGE postfix
514 #define _ad3 ADC_PAGE=3 // ADC_PAGE postfix
515 #define _ad4 ADC_PAGE=4 // ADC_PAGE postfix
516 #define _ad5 ADC_PAGE=5 // ADC_PAGE postfix
517 #define _ad6 ADC_PAGE=6 // ADC_PAGE postfix
518
519 // SCU_PAGE
520 #define _su0 SCU_PAGE=0 // SCU_PAGE postfix
521 #define _su1 SCU_PAGE=1 // SCU_PAGE postfix
522 #define _su2 SCU_PAGE=2 // SCU_PAGE postfix
523 #define _su3 SCU_PAGE=3 // SCU_PAGE postfix
524
525 // CCU6_PAGE
526 #define _cc0 CCU6_PAGE=0 // CCU6_PAGE postfix
527 #define _cc1 CCU6_PAGE=1 // CCU6_PAGE postfix
528 #define _cc2 CCU6_PAGE=2 // CCU6_PAGE postfix
529 #define _cc3 CCU6_PAGE=3 // CCU6_PAGE postfix
530

```



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

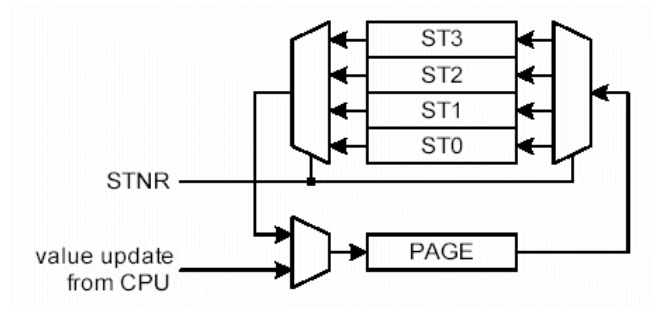


```

530
531 #define SST0  0x80      // Save SFR page to ST0
532 #define RST0  0xC0      // Restore SFR page from ST0
533 #define SST1  0x90      // Save SFR page to ST1
534 #define RST1  0xD0      // Restore SFR page from ST1
535 #define SST2  0xA0      // Save SFR page to ST2
536 #define RST2  0xE0      // Restore SFR page from ST2
537 #define SST3  0xB0      // Save SFR page to ST3
538 #define RST3  0xF0      // Restore SFR page from ST3
539 #define noSST 0x00      // Switch page without saving
540
541 #define SFR_PAGE(pg,op) pg+op
542
543 //  SYSCON0_RMAP
544 //  The access to the mapped SFR area is enabled.
545 #define SET_RMAP() SYSCON0 |= 0x01
546
547 //  The access to the standard SFR area is enabled.
548 #define RESET_RMAP() SYSCON0 &= ~0x01
549
550
551 #define _su  SCU_PAGE // SCU_PAGE
552
553 #define STR_PAGE(pg,op)  { _push_(op); \
554   pg ; }
555
556 #define RST_PAGE(op)  _pop_(op)
  
```




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

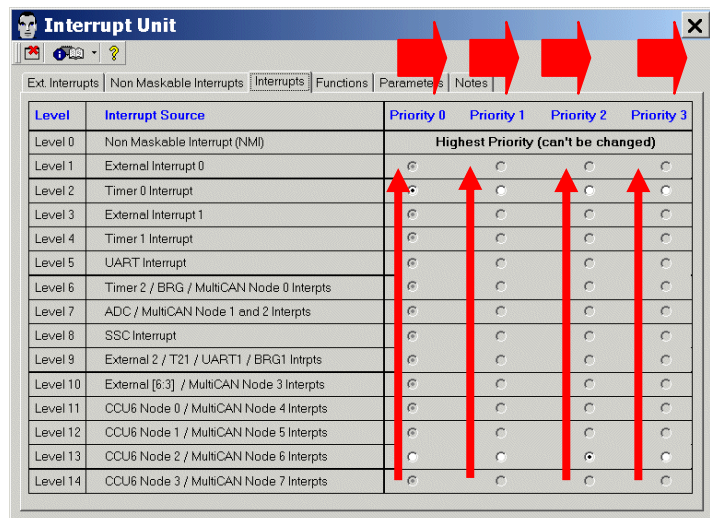


Note (Source: Application Note AP08053):

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.

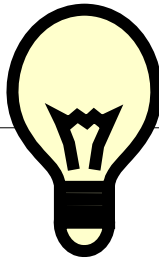
This leaves us with five interrupt priorities and four storage containers.

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.

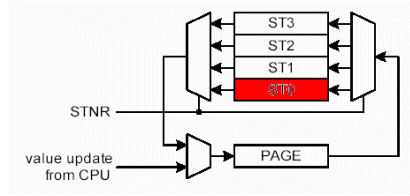
The ISR may also call functions that could modify page registers. If these functions are shared by ISRs of different priority levels, then these functions can be interrupted. It is therefore necessary to save and restore the page registers that are modified in these functions. In such cases, the stack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

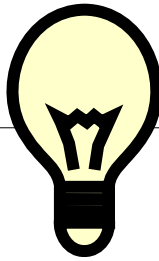


Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input checked="" 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"/>

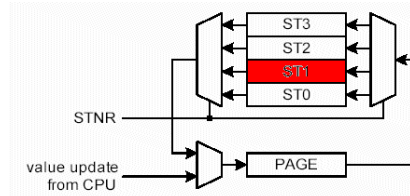
```

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
195     if (IRCON4 & 0x01) // if CCU6SR2
196     {
197         IRCON4 &= ~(ubyte)0x01;
198
199         // USER CODE BEGIN (NodeI2,3)
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243     SFR_PAGE(_su3, RST0);           // restore the old SCU page
244 } // 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"/>

XC888 - µVision3 - [C:\XC888\SHARED_INT.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

```

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
195     if (IRCON4 & 0x01) // if CCU6SR2
196     {
197         IRCON4 &= ~(ubyte)0x01;
198
199         // USER CODE BEGIN (NodeI2,3)
200
242
243     SFR_PAGE(_su3, RST1);           // restore the old SCU page
244 } // End of function SHINT_vixINTR12Isr

```

SAVE to ST1 (points to line 190)

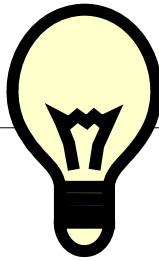
RESTORE from ST1 (points to line 243)

Project Workspace

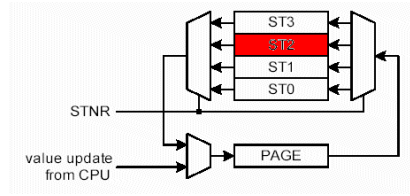
- Target 1
 - Dave Files
 - START_XC
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C
 - SHARED_I
 - CC6.C
 - User Files

MAIN.C | UART.C | T01.C | SHARED... | CC6.C | MAIN.H

Build Command Find in Files



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

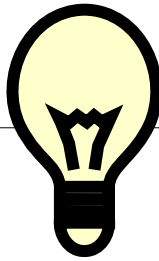


Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input checked="" 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"/>

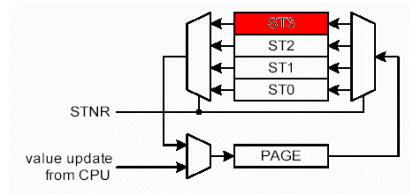
```

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
195     if (IRCON4 & 0x01) // if CCU6SR2
196     {
197         IRCON4 &= ~(ubyte)0x01;
198
199         // USER CODE BEGIN (NodeI2,3)
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243     SFR_PAGE(_su3, RST2);           // restore the old SCU page
244 } // 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"/>

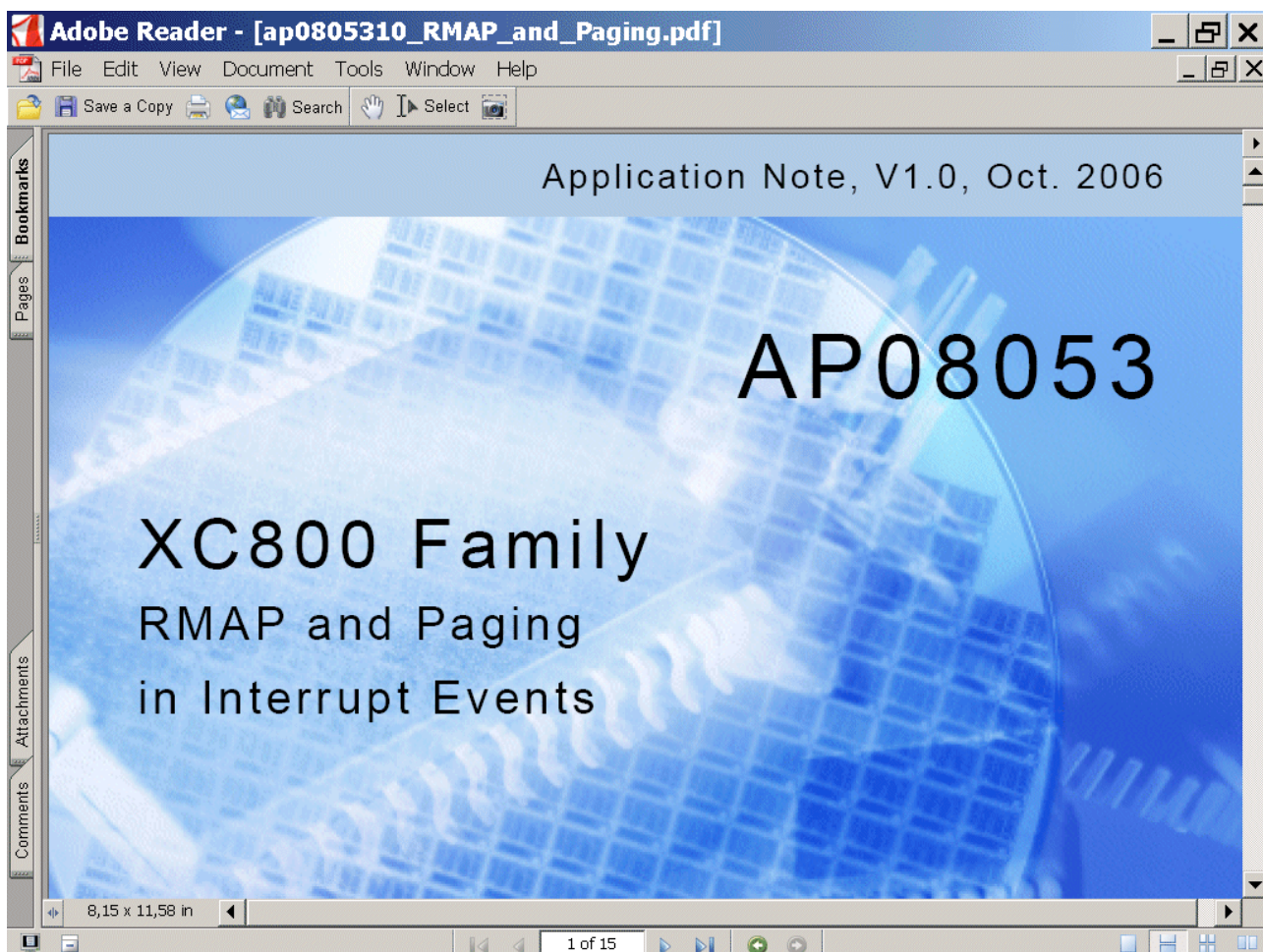
```

183
184 void SHINT_vixINTR12Isr(void) interrupt XINTR12INT
185 {
186     // USER CODE BEGIN (NodeI2,2)
187
188     // USER CODE END
189
190     SFR_PAGE(_su3, SST3);           // switch to page 3
191
192     // CCU6 Node 2 interrupt handling section...
193
194
195     if (IRCON4 & 0x01) // if CCU6SR2
196     {
197         IRCON4 &= ~(ubyte)0x01;
198
199         // USER CODE BEGIN (NodeI2,3)
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
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("\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("\nplaying: Yesterday\n");
    if (next_song_c && ((sizeof(songc)-1)< MAX_SONG_LENGTH) )
        strcpy(song,songc), max=(sizeof(songc))-1, --next_song_c,
        printf("\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("\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("\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("\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("\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("\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("\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("\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("\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("\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("\nplaying: musical scale / chromatic scale / for testing purpose / Tonleiter\n");
    printf("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("Song aborted.\n");
else
    printf("End of the song reached (pos=%5u of max%5u).\n",pos,max);
}
```

XC888 - µVision3 - [C:\XC888\MAIN.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
 - Dave Files
 - START_XC.a5
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C
 - SHARED_INT
 - CC6.C
 - User Files

```

314 // USER CODE BEGIN (MAIN_Main,1)
315 char input (void)
316 {
317     char in=' ';
318     do
319     {
320         printf(question);
321         while (!RI);
322         RI=0;
323         in = SBUF;
324     }while ( !(in>='a'&&in<='m') );
325     return in;
326 }
327
328 void play_song(void)
329 {
330     max=0;
331     if ( next_song_a && ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
332         strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
333         printf("\nplaying: Maus am Mars\n");
334     if ( next_song_b && ((sizeof(songb)-1)< MAX_SONG_LENGTH) )
335         strcpy(song,songb), max=(sizeof(songb))-1, --next_song_b,
336         printf("\nplaying: Yesterday\n");
337     if ( next_song_c && ((sizeof(songc)-1)< MAX_SONG_LENGTH) )
338         strcpy(song,songc), max=(sizeof(songc))-1, --next_song_c,
339         printf("\nplaying: Frere Jacques - Lazy John - Bruder Jakob\n");
340     if ( next_song_d && ((sizeof(songd)-1)< MAX_SONG_LENGTH) )
341         strcpy(song,songd), max=(sizeof(songd))-1, --next_song_d,
342         printf("\nplaying: Happy birthday\n");
343     if ( next_song_e && ((sizeof(songe)-1)< MAX_SONG_LENGTH) )
344         strcpy(song,songe), max=(sizeof(songe))-1, --next_song_e,
345         printf("\nplaying: Take Me Home, Country Roads\n");
346     if ( next_song_f && ((sizeof(songf)-1)< MAX_SONG_LENGTH) )
347         strcpy(song,songf), max=(sizeof(songf))-1, --next_song_f,
348         printf("\nplaying: Es tanzt ein Bi-ba-butzemann\n");
349     if ( next_song_g && ((sizeof(songg)-1)< MAX_SONG_LENGTH) )
350         strcpy(song,songg), max=(sizeof(songg))-1, --next_song_g,
351         printf("\nplaying: Ich geh mit meiner Laterne\n");
352     if ( next_song_h && ((sizeof(songh)-1)< MAX_SONG_LENGTH) )
353         strcpy(song,songh), max=(sizeof(songh))-1, --next_song_h,
354         printf("\nplaying: The little drummer boy\n");
355     if ( next_song_i && ((sizeof(songi)-1)< MAX_SONG_LENGTH) )
356         strcpy(song,songi), max=(sizeof(songi))-1, --next_song_i,
357         printf("\nplaying: Hey, Pippi Langstrumpf\n");
358     if ( next_song_j && ((sizeof(songj)-1)< MAX_SONG_LENGTH) )
359         strcpy(song,songj), max=(sizeof(songj))-1, --next_song_j,
360         printf("\nplaying: Stille Nacht, heilige Nacht\n");
361     if ( next_song_k && ((sizeof(songk)-1)< MAX_SONG_LENGTH) )
362         strcpy(song,songk), max=(sizeof(songk))-1, --next_song_k,
363         printf("\nplaying: Junge komm bald wieder\n");
364     if ( next_song_l && ((sizeof(songl)-1)< MAX_SONG_LENGTH) )

```

```

365     strcpy(song,songl), max=(sizeof(songl))-1, --next_song_l,
366     printf("\nplaying: Lili Marleen\n");
367     if (next_song_m && ((sizeof(songm)-1)< MAX_SONG_LENGTH) )
368     strcpy(song,songm), max=(sizeof(songm))-1, --next_song_m,
369     printf("\nplaying: musical scale / chromatic scale / for testing pur
370     printf("song-length = %5u Byte[s] \n",max);
371     pos=0;
372
373     if (max>0) // there is something to play
374     {
375         // start CAPCOM 6 - Timer T12 - ISR the first time:
376         SFR_PAGE(_cc2,nosST); // switch to CCU6_PAGE=2 without saving !!!
377         CCU6_ISSL = CCU6_ISSL | 0x80; // set ST12PM -> Set-Timer-T12Period
378
379         while (pos<=max); // wait until song end is reached or abort by
380     }
381
382     if ( (SBUF=='z') )
383     printf("Song aborted.\n");
384     else
385     printf("End of the song reached (pos=%5u of max%5u).\n",pos,max);
386 }
387
388 // USER CODE END
389
390 void main(void)
391 {
392     // USER CODE BEGIN (MAIN_Main,2)
393
394     // USER CODE END
395
396     MAIN_vInit();
397
398     // USER CODE BEGIN (MAIN_Main,3)

```

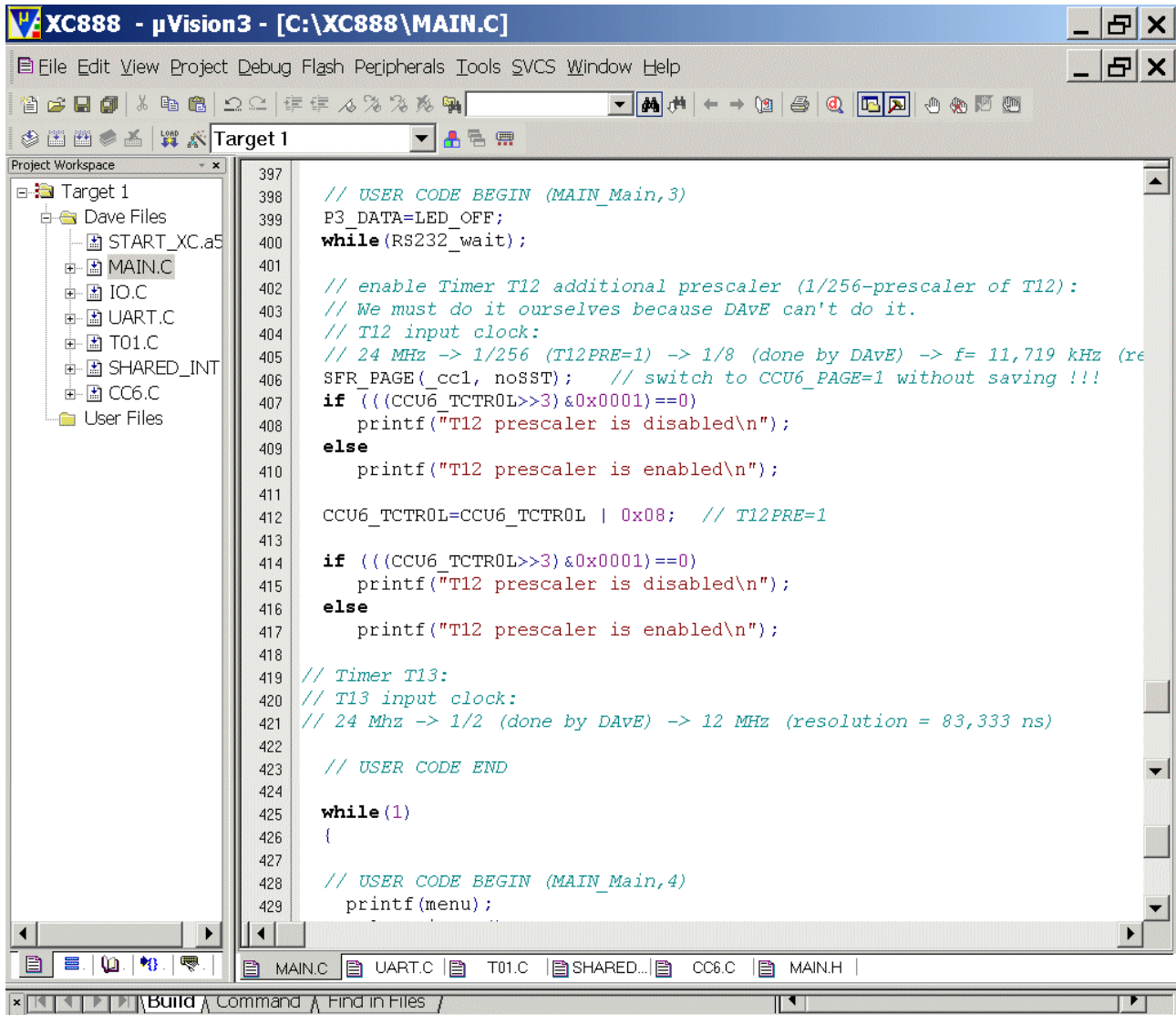
Double click **MAIN.C** and **insert** code (set the T12PRE bit):

```
// 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("T12 prescaler is disabled\n");
else
    printf("T12 prescaler is enabled\n");

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

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

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

```

397
398 // USER CODE BEGIN (MAIN_Main,3)
399 P3_DATA=LED_OFF;
400 while(RS232_wait);
401
402 // enable Timer T12 additional prescaler (1/256-prescaler of T12):
403 // We must do it ourselves because DAVE can't do it.
404 // T12 input clock:
405 // 24 MHz -> 1/256 (T12PRE=1) -> 1/8 (done by DAVE) -> f= 11,719 kHz (re
406 SFR_PAGE( cc1, noSST); // switch to CCU6_PAGE=1 without saving !!!
407 if (((CCU6_TCTRL0L>>3)&0x0001)==0)
408     printf("T12 prescaler is disabled\n");
409 else
410     printf("T12 prescaler is enabled\n");
411
412 CCU6_TCTRL0L=CCU6_TCTRL0L | 0x08; // T12PRE=1
413
414 if (((CCU6_TCTRL0L>>3)&0x0001)==0)
415     printf("T12 prescaler is disabled\n");
416 else
417     printf("T12 prescaler is enabled\n");
418
419 // Timer T13:
420 // T13 input clock:
421 // 24 Mhz -> 1/2 (done by DAVE) -> 12 MHz (resolution = 83,333 ns)
422
423 // USER CODE END
424
425 while(1)
426 {
427
428 // USER CODE BEGIN (MAIN_Main,4)
429     printf(menu);

```

T12PRE=1:



Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

XC886/888CLM

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 $f_{T12} = f_{CCU} / 2^{<T12CLK>}$ 000 $f_{T12} = f_{CCU}$ 001 $f_{T12} = f_{CCU}/2$ 010 $f_{T12} = f_{CCU}/4$ 011 $f_{T12} = f_{CCU}/8$ 100 $f_{T12} = f_{CCU}/16$ 101 $f_{T12} = f_{CCU}/32$ 110 $f_{T12} = f_{CCU}/64$ 111 $f_{T12} = f_{CCU}/128$
T12PRE	3	rw	Timer T12 Prescaler Bit In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0 The additional prescaler for T12 is disabled. 1 The additional prescaler for T12 is enabled.
T12R	4	rh	Timer T12 Run Bit T12R starts and stops timer

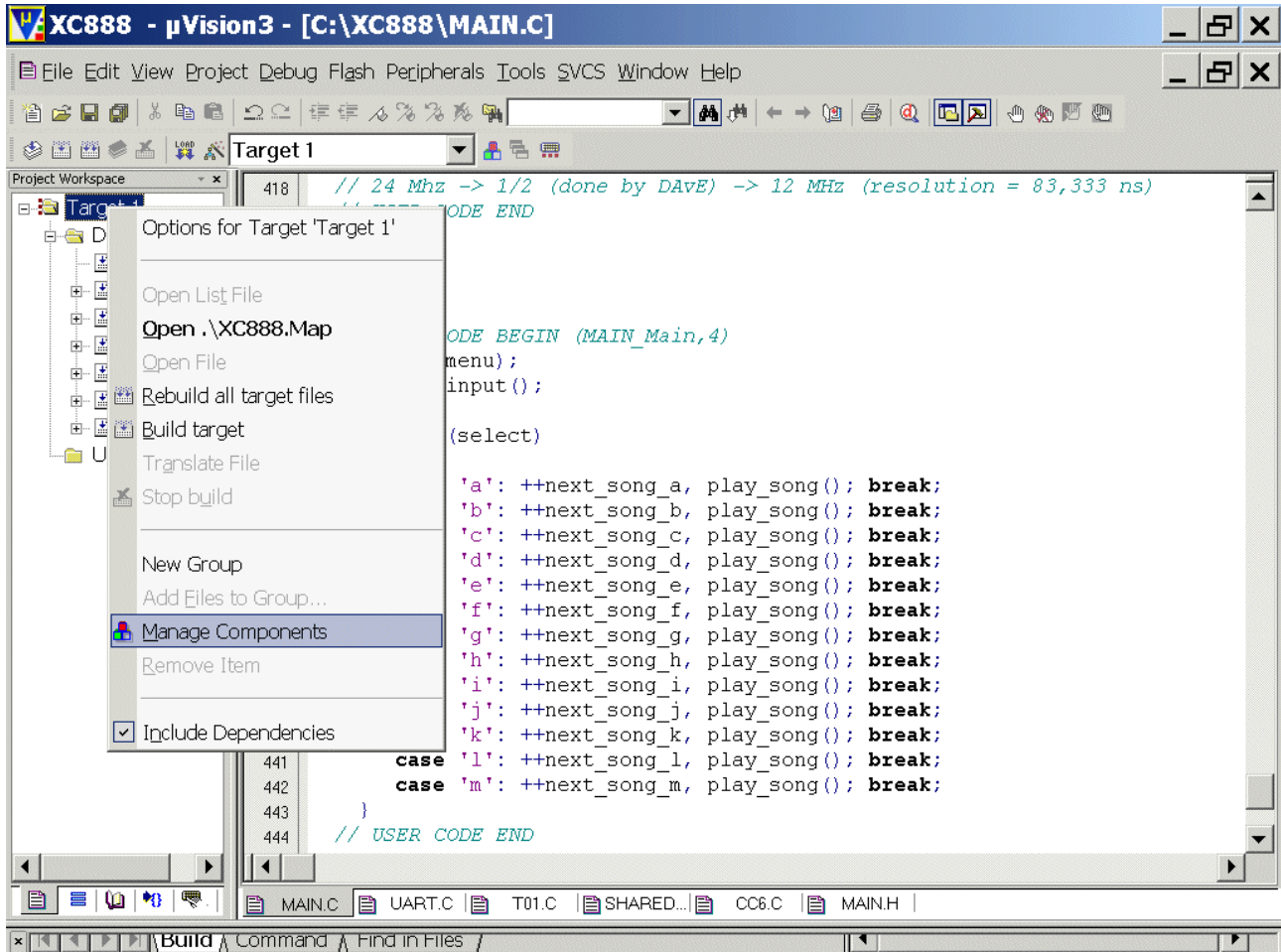
SFR_PAGE(_cc1, noSST);
// switch to CCU6_PAGE=1
// without saving !!!

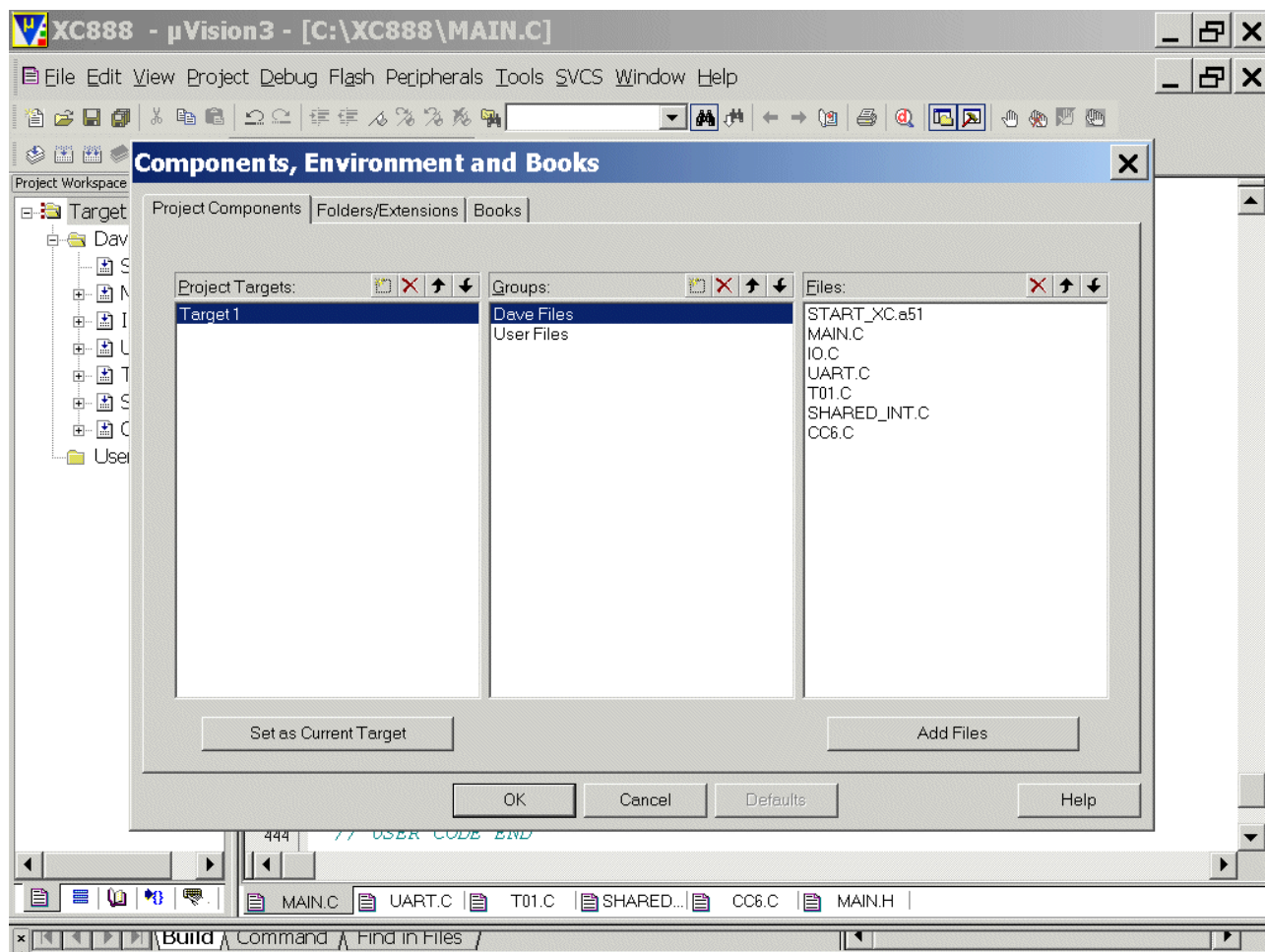
Table 14-3 SFR Address List for Pages 0-3

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

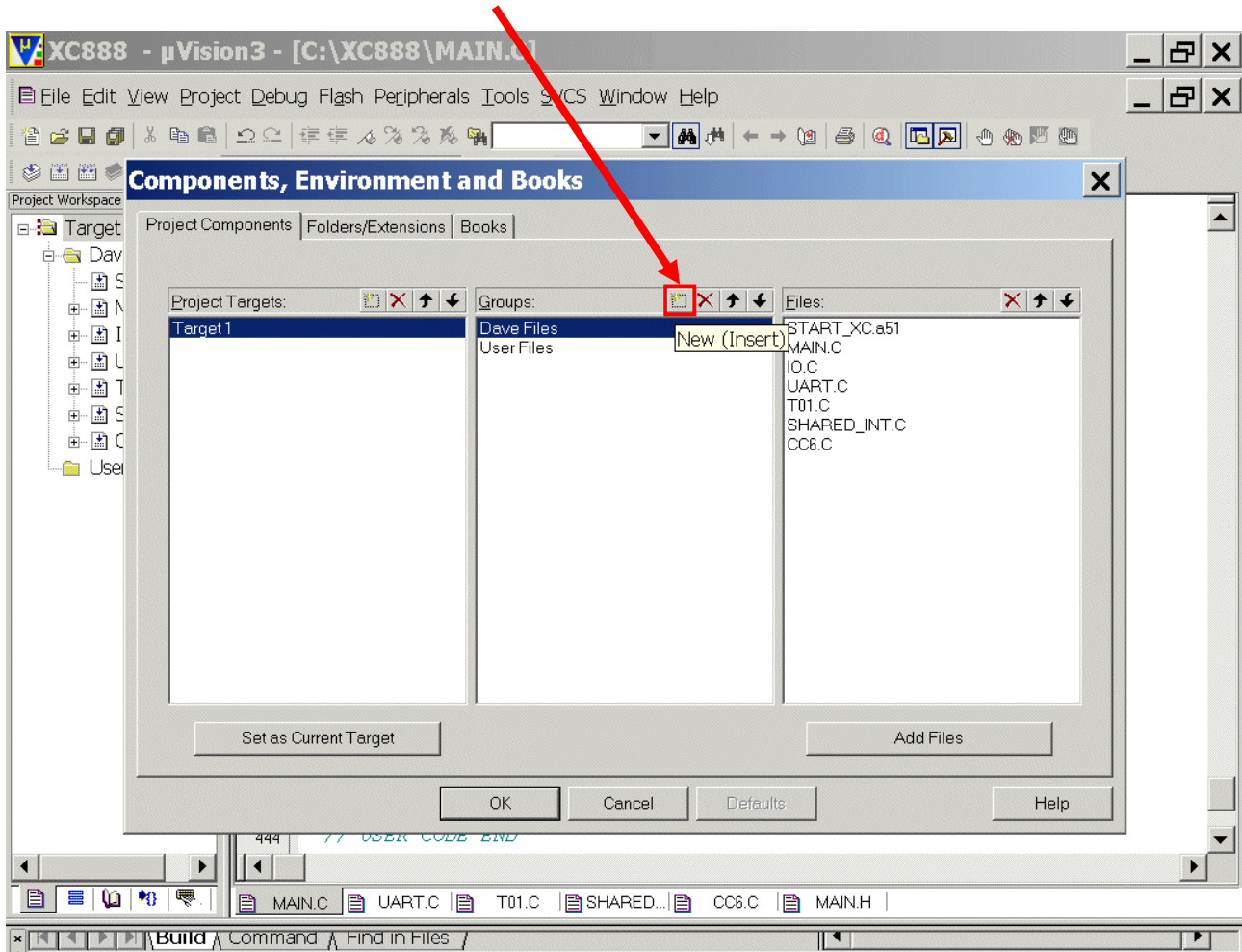
383 of 661

Mouse position: **Project Window**, Target1: **click right mouse button**
click Manage Components

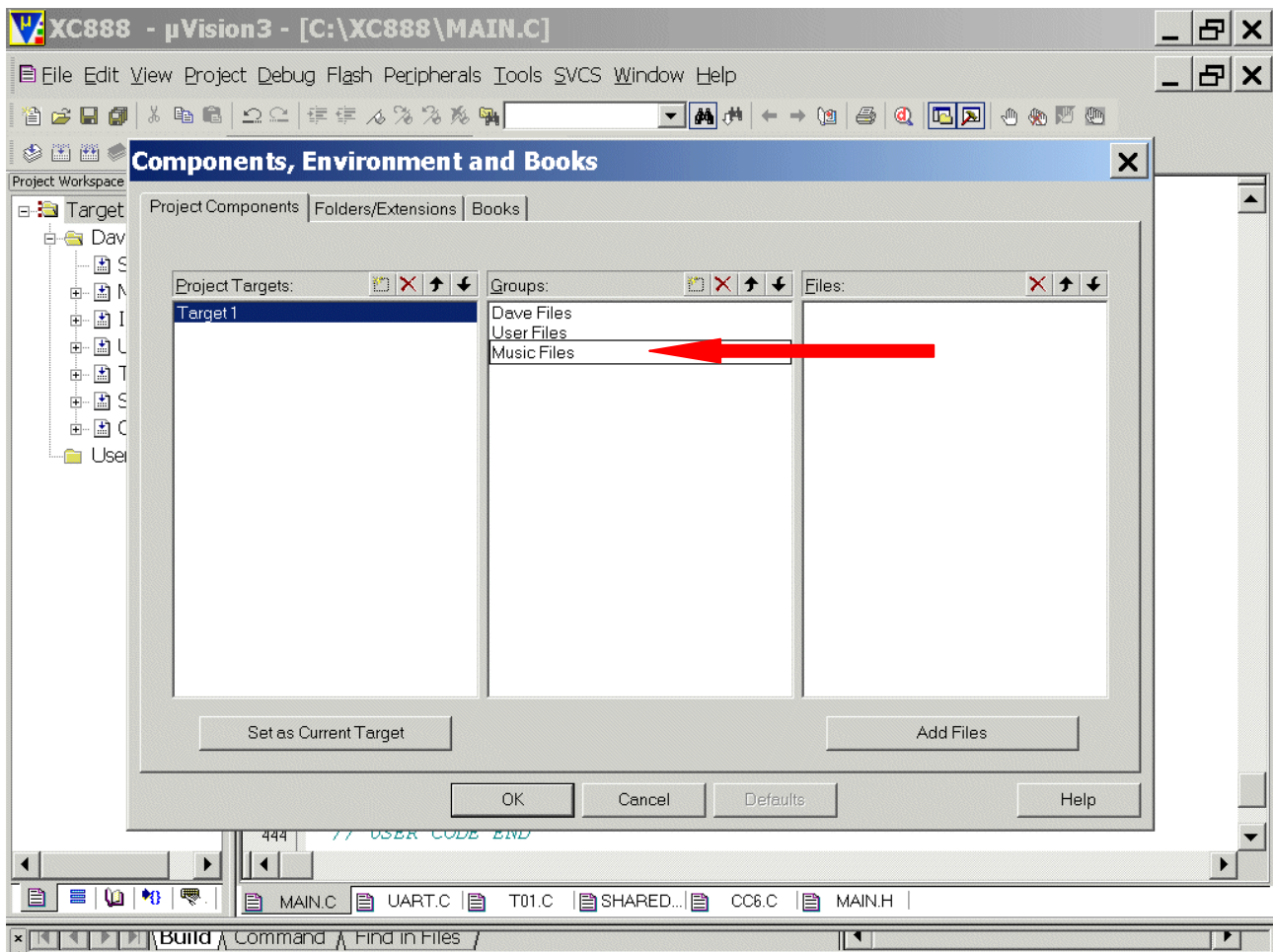




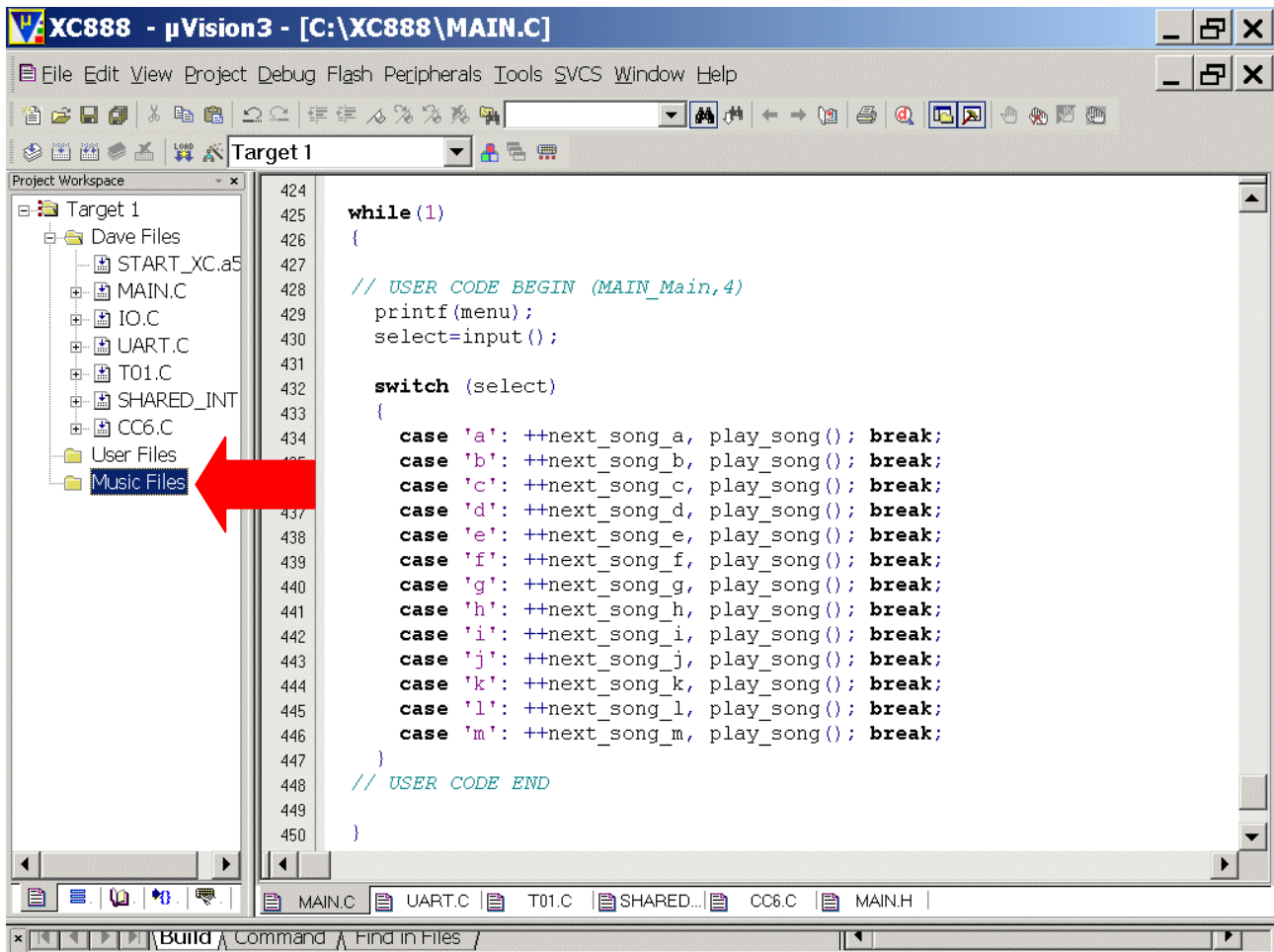
Project Components: Groups: **click** New (Insert)



Insert Music Files



Click OK



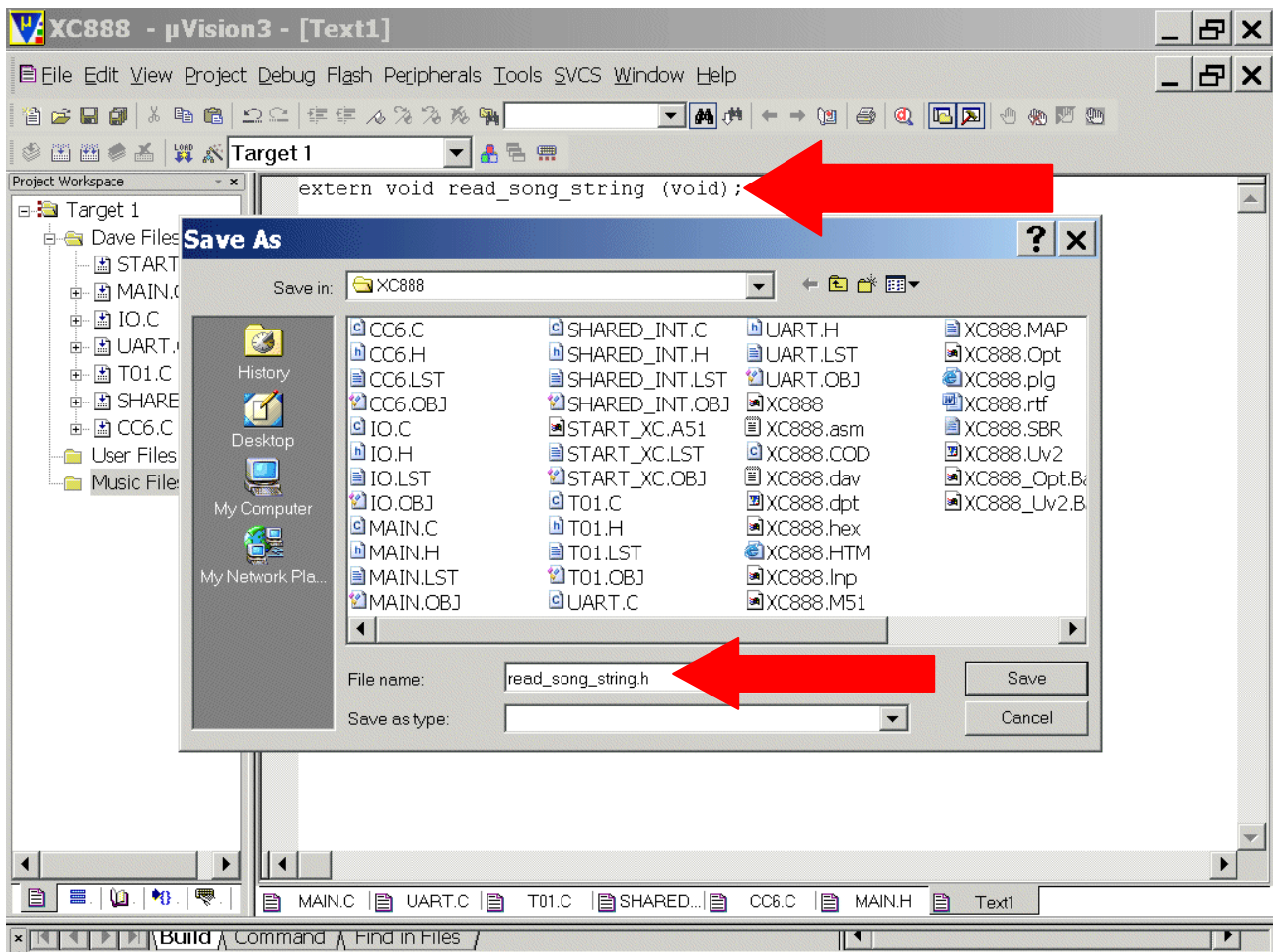
File – New

Insert:

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

File - Save As

Insert: read_song_string.h



Click Save

File – New

Insert:

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

// 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 :
            ; 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];
    }

```

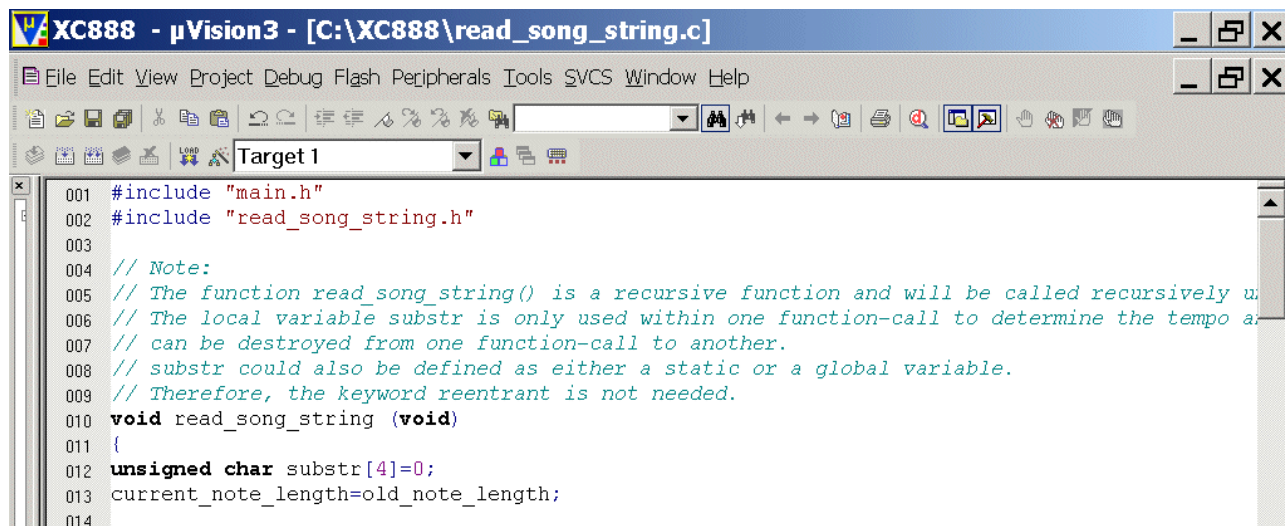
```
        substr[2]=' ';
    }
    tempo=atoi(substr);
    pos++;
    read_song_string(); break;

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 */
```

```
001 #include "main.h"
002 #include "read_song_string.h"
003
004 // Note:
005 // The function read_song_string() is a recursive function and will be called recursively u
006 // The local variable substr is only used within one function-call to determine the tempo a
007 // can be destroyed from one function-call to another.
008 // substr could also be defined as either a static or a global variable.
009 // Therefore, the keyword reentrant is not needed.
010 void read_song_string (void)
011 {
012     unsigned char substr[4]=0;
013     current_note_length=old_note_length;
014 }
```

```

015 switch (song[pos])
016 {
017     // select note:
018     case 'C': note=0;
019     switch (song[++pos])
020     {
021         case '+': note++; pos++; break;
022         case '-': octave--;
023                 note=11;
024                 pos++;          break;
025         default : ;          break;
026     }          break;
027
028     case 'D': note=2;
029     switch (song[++pos])
030     {
031         case '+': note++; pos++; break;
032         case '-': note--; pos++; break;
033         default : ;          break;
034     }          break;
035
036     case 'E': note=4;
037     switch (song[++pos])
038     {
039         case '+': note++; pos++; break;
040         case '-': note--; pos++; break;
041         default : ;          break;
042     }          break;
043
044     case 'F': note=5;
045     switch (song[++pos])
046     {
047         case '+': note++; pos++; break;
048         case '-': note--; pos++; break;
049         default : ;          break;
050     }          break;
051
052     case 'G': note=7;
053     switch (song[++pos])
054     {
055         case '+': note++; pos++; break;
056         case '-': note--; pos++; break;
057         default : ;          break;
058     }          break;
059
060     case 'A': note=9;
061     switch (song[++pos])
062     {
063         case '+': note++; pos++; break;
064         case '-': note--; pos++; break;
065         default : ;          break;
066     }          break;
067
068     case 'H': note=11;
069     switch (song[++pos])
070     {
071         case '+': octave++;
072                 note=0;
073                 pos++;          break;
074         case '-': note--; pos++; break;
075         default : ;          break;
076     }          break;
077

```

```

078 // adjust note length:
079 case 'L': switch (song[++pos])
080 {
081     case '1': if (song[++pos]=='6')
082         current_note_length=length_of_a_whole_note/16;
083     else
084     {
085         pos--;
086         current_note_length=length_of_a_whole_note;
087     }
088     break;
089     case '2': current_note_length=length_of_a_whole_note/2; break;
090     case '4': current_note_length=length_of_a_whole_note/4; break;
091     case '8': current_note_length=length_of_a_whole_note/8; break;
092     default : ; break;
093 }
094 old_note_length=current_note_length;
095 pos++;
096 read_song_string(); break;
097
098

```

```

099 // set rest:
100 case 'P': switch (song[++pos])
101 {
102     case '1': if (song[++pos]=='6')
103         current_note_length=length_of_a_whole_note/16;
104     else
105     {
106         pos--;
107         current_note_length=length_of_a_whole_note;
108     }
109     break;
110     case '2':current_note_length=length_of_a_whole_note/2; break;
111     case '4':current_note_length=length_of_a_whole_note/4; break;
112     case '8':current_note_length=length_of_a_whole_note/8; break;
113     default : ; break;
114 }
115 note=12;
116 pos++; break;
117

```

```

118 // adjust octave:
119 case 'O': switch (song[++pos])
120 {
121     case '0': octave=1; break;
122     case '1': octave=2; break;
123     case '2': octave=4; break;
124     case '3': octave=8; break;
125     default :      ; break;
126 }
127 pos++;
128 read_song_string();    break;
129

```

```

130 // tempo:
131 case 'T': pos++;
132     substr[3]=0; //string termination
133     if (song[pos]=='1')
134     {
135         substr[0]=song[pos];
136         substr[1]=song[++pos];
137         substr[2]=song[++pos];
138     }
139     else
140     {
141         substr[0]=song[pos];
142         substr[1]=song[++pos];
143         substr[2]=' ';
144     }
145     tempo=atoi(substr);
146     pos++;
147     read_song_string();    break;
148
149     default: ;    break;
150 } /* end case */
151

```

```

152 // extend note length by half:
153 if (song[pos]=='.')
154 {
155     old_note_length=current_note_length;
156     current_note_length=current_note_length*3.0/2.0;
157     pos++;
158 }
159
160 if (pos==max) pos++;
161
162 } /* end read_song_string */
163

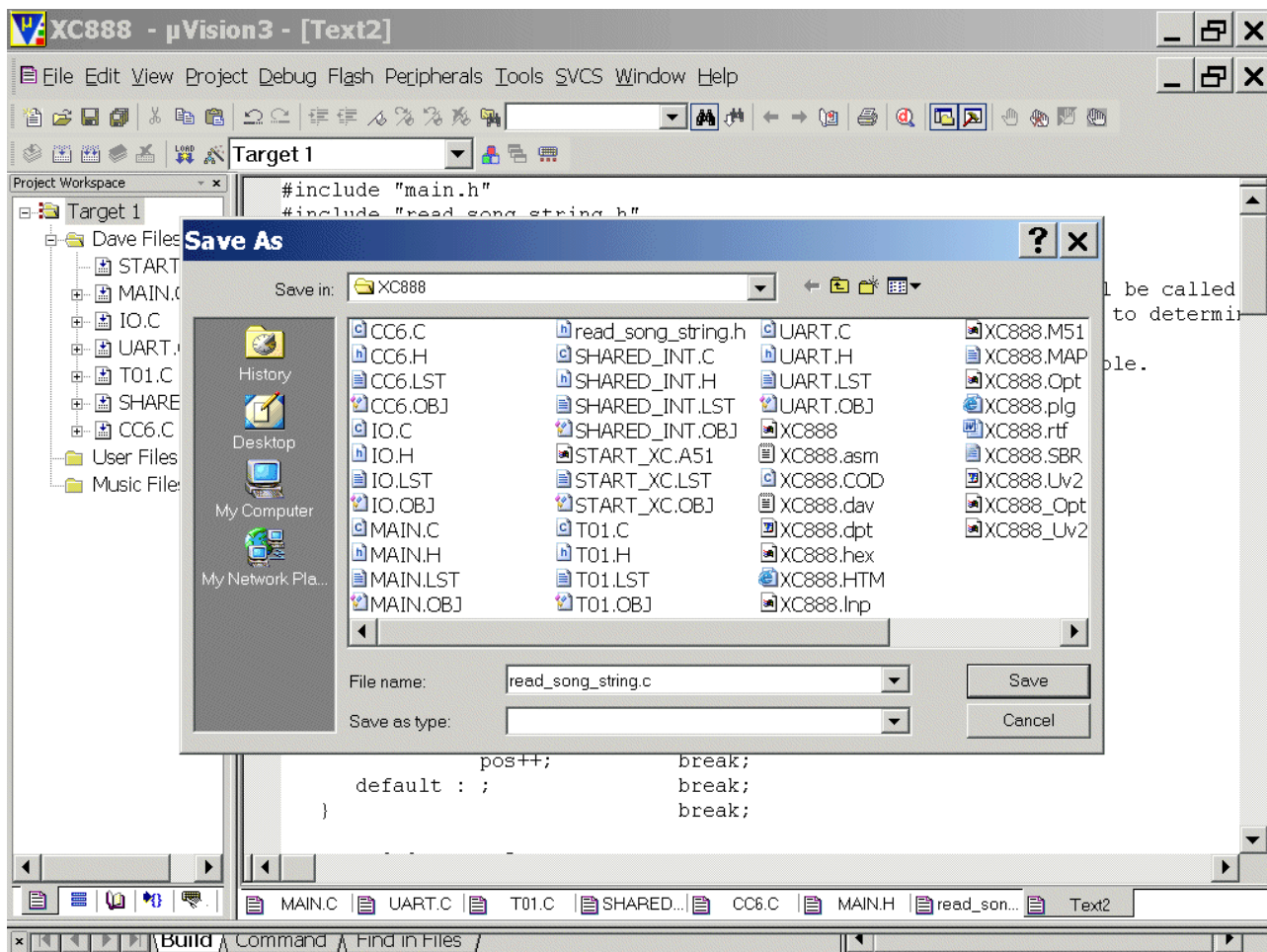
```

MAIN.C | UART.C | T01.C | SHARED... | CC6.C | MAIN.H | read_son... | read_son...

Build | Command | Find in Files

File - Save As

Insert: read_song_string.c



Click Save



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

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 (Source: Keil Books, Complete User's Guide Selection, C51 Development Tools):

Normally, functions in Cx51 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.

The reentrant function attribute allows you to declare functions that may be reentrant and, therefore, may be called recursively.

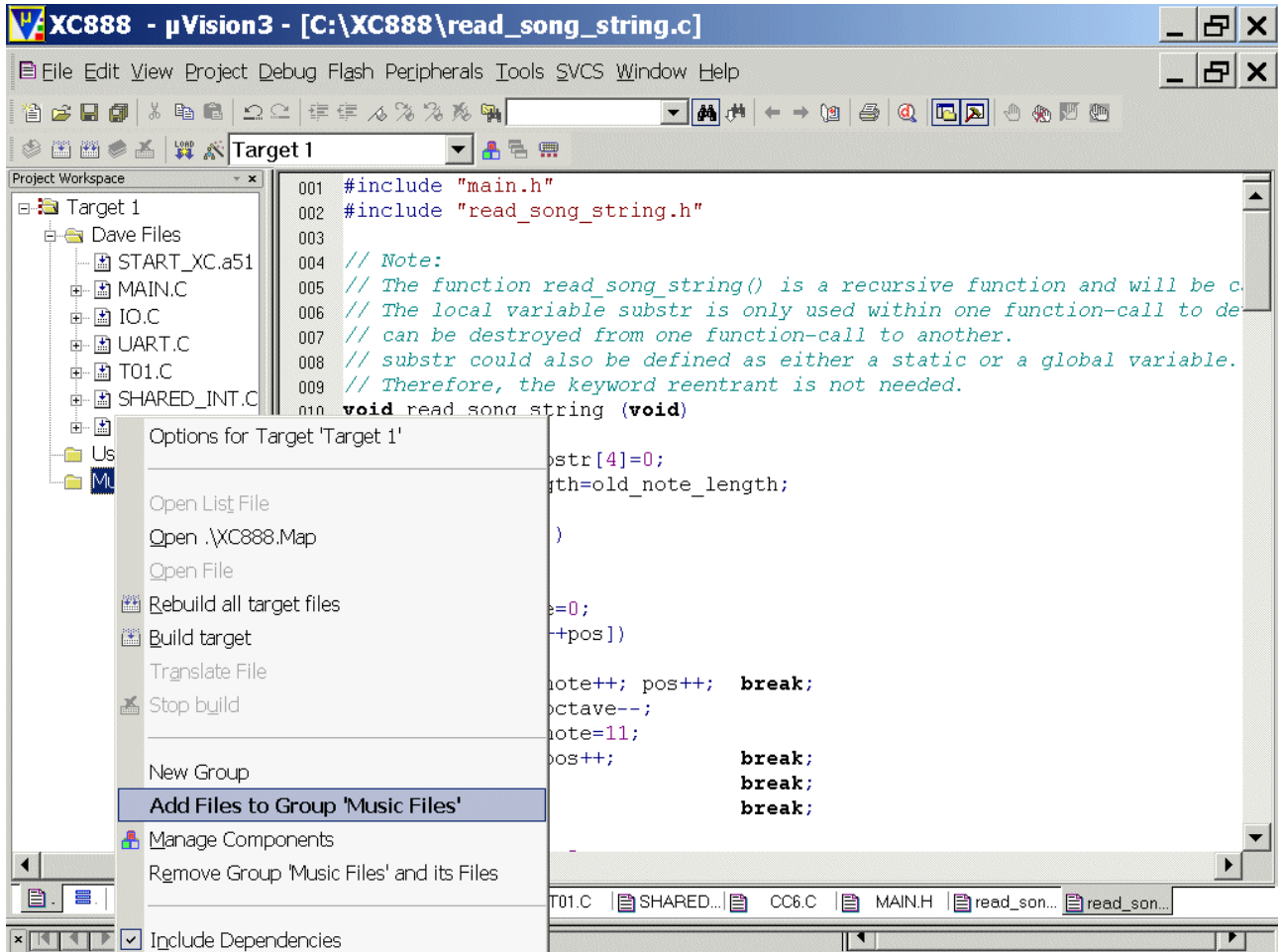
Reentrant functions can be called recursively and can be called simultaneously by two or more processes. Reentrant functions are often required in real-time applications or in situations where interrupt code and non-interrupt code must share a function.

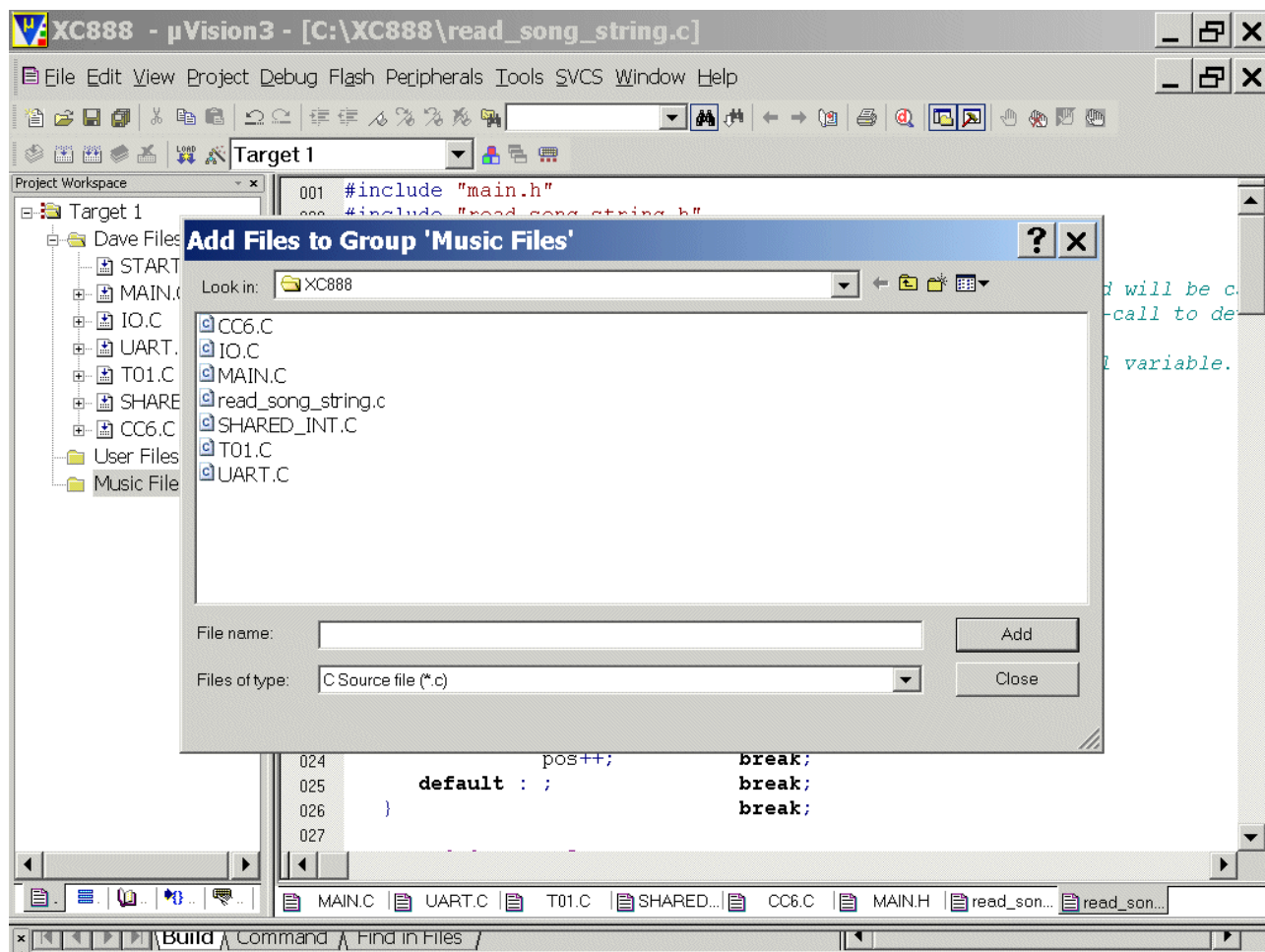
Note:

In our case the reentrant function attribute is not needed because 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**.

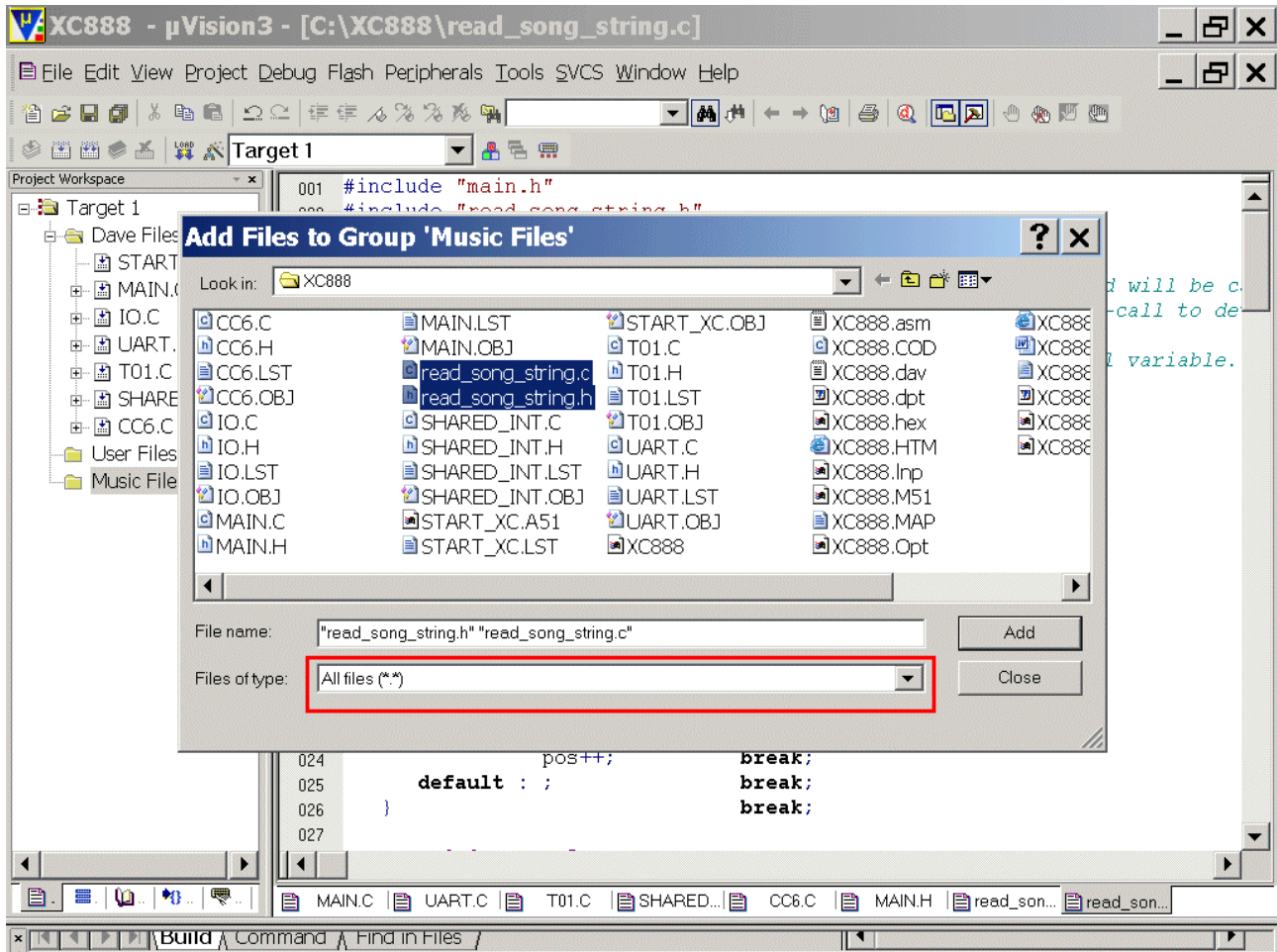
Mouse position: **Project Window**, Music Files: **click right mouse button**
Click Add Files to Group 'Music Files'





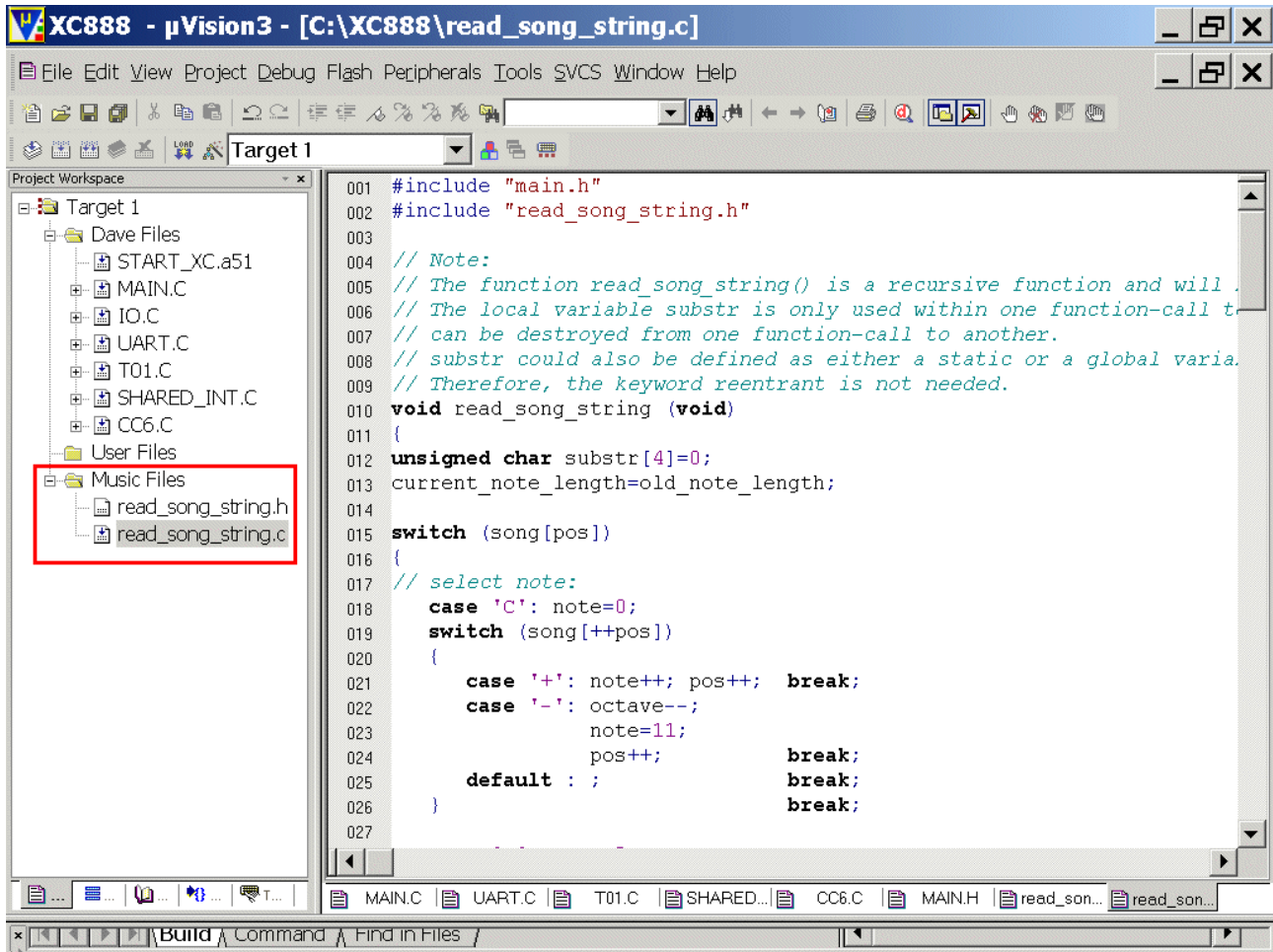
Files of type: select All files

Mark read_song_string.h and read_song_string.c



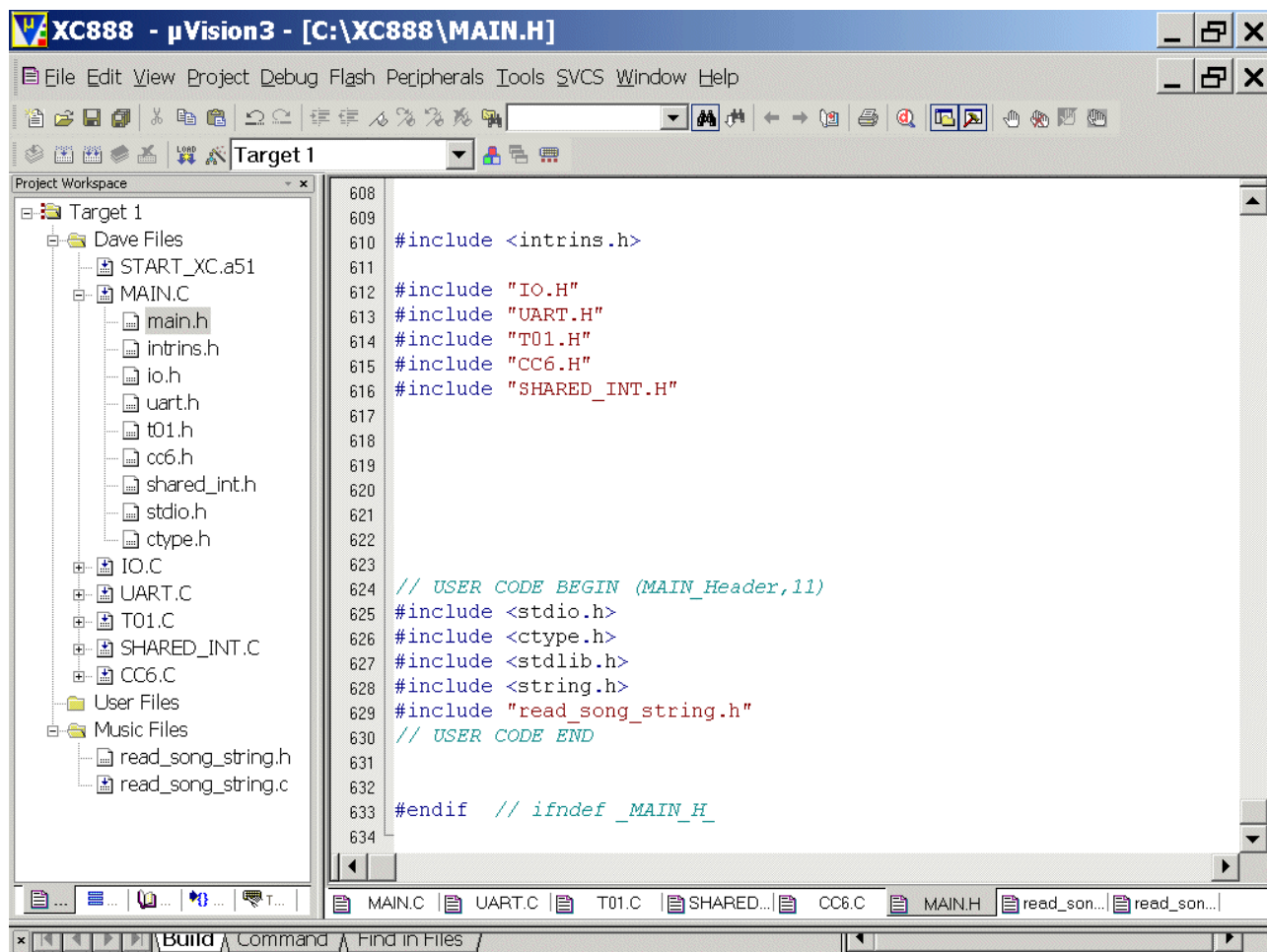
Click Add

Click Close



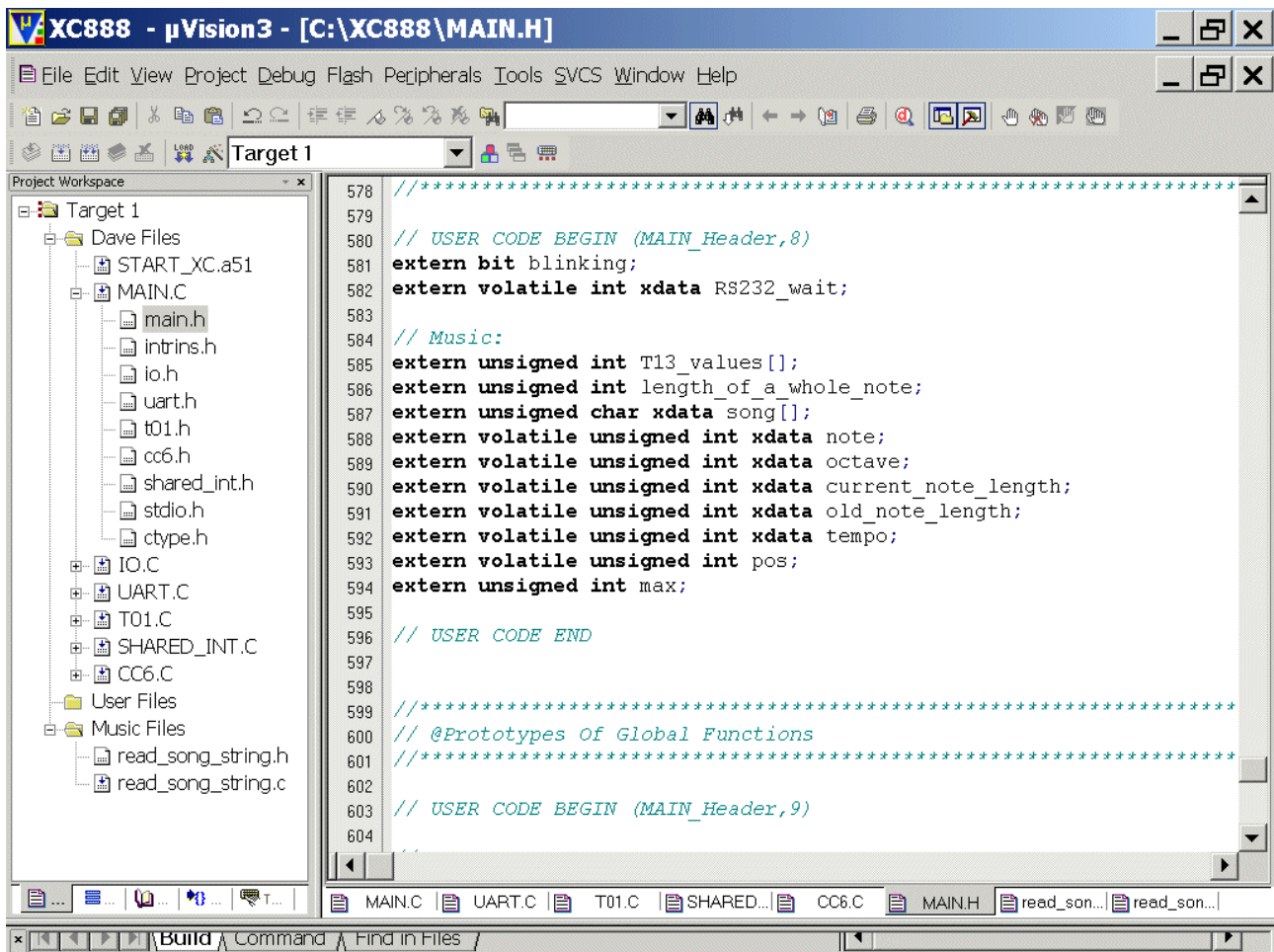
Double click **MAIN.H** and insert Project Includes:

```
#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;
```



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
    CC6_vSetTmrPeriod(CC6_TIMER_12, (current_note_length/tempo*120));

    SFR_PAGE(_cc0,noSST); // switch to CCU6_PAGE = 0 without saving
    // Page 0: CC60SRL, CC60SRH:
    // Channel_0: not used, only for measurement (5% duty cycle):

    CC6_vLoadChannelShadowRegister(CC6_CHANNEL_0, (current_note_length/tempo*120)/100
*95);
    // Page 0: CC61SRL, CC61SRH:
    // Channel_1: not used, only for measurement (100% duty cycle):

    CC6_vLoadChannelShadowRegister(CC6_CHANNEL_1, (current_note_length/tempo*120)/100
*0);
    // Page 0: CC62SRL, CC62SRH:
    // Channel_2: if compare value CCU6_CC62SR == 0 -> 100 % duty cycle for note
length:

    CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2, (current_note_length/tempo*120)/100
*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
    CC6_vSetTmrPeriod(CC6_TIMER_13, (T13_values[note]/octave));

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

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

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

```

```
if      (octave == 1) printf("'  ");
else if (octave == 2) printf("' ' ");
else if (octave == 4) printf("' ' ' ");
else if (octave == 8) printf("' ' ' '");
else      printf("????");

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

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



XC888 - µVision3 - [C:\XC888\SHARED_INT.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
 - Dave Files
 - START_
 - MAIN.C
 - IO.C
 - UART.C
 - T01.C
 - SHARED
 - CC6.C
 - User Files
 - Music Files
 - read_so
 - read_so

```

218 SFR_PAGE(_cc3, noSST);           // switch to page 3
219
220 if(CCU6_ISL & 0x80) //if ISL_T12PM
221 {
222     //timer T12 period match detection
223
224     SFR_PAGE(_cc0, noSST);       // switch to page 0
225
226     CCU6_ISRL = 0x80; //clear flag ISL_T12PM
227
228     // USER CODE BEGIN (NodeI2,17)
229
230 if ( (char)SBUF == 'z' ) // song aborted by user
231     pos=max+1;
232
233 if (pos<=max)
234 {
235     read_song_string(); // read next note
236                         // this function is called recursively until a note is f
237
238     // T12PR: adjust note length / set Timer 12 period value for note length:
239     // Page 1: T12PRL, T12PRH
240     SFR_PAGE(_cc1,noSST); // switch to CCU6_PAGE = 1 without saving
241     CC6_vSetTmrPeriod(CC6_TIMER_12,(current_note_length/tempo*120));
242
243     SFR_PAGE(_cc0,noSST); // switch to CCU6_PAGE = 0 without saving
244     // Page 0: CC60SRL, CC60SRH:
245     // Channel_0: not used, only for measurement (5% duty cycle):
246     CC6_vLoadChannelShadowRegister(CC6_CHANNEL_0,(current_note_length/tempo*120)
247     // Page 0: CC61SRL, CC61SRH:
248     // Channel_1: not used, only for measurement (100% duty cycle):
249     CC6_vLoadChannelShadowRegister(CC6_CHANNEL_1,(current_note_length/tempo*120)
250     // Page 0: CC62SRL, CC62SRH:
251     // Channel_2: if compare value CCU6_CC62SR == 0 -> 100 % duty cycle for note
252     CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2,(current_note_length/tempo*120)
253
254     // Page 0: bit: T12STR in TCTR4L
255     CC6_vEnableShadowTransfer(CC6_TIMER_12);
256
257     // T13, adjust note frequency / set Timer 13 period value for note frequency
258     // Page 1: T13PRL, T13PRH:
259     SFR_PAGE(_cc1,noSST); // switch to CCU6_PAGE = 1 without saving
260     CC6_vSetTmrPeriod(CC6_TIMER_13,(T13_values[note]/octave));
261
262     SFR_PAGE(_cc0,noSST); // switch to CCU6_PAGE = 0 without saving
263     // Channel_3: duty cycle note-frequency = 50 %
264     // Page 0 : CC63SRL, CC63SRH:
265     CC6_vLoadChannelShadowRegister(CC6_CHANNEL_3,( T13_values[note]/octave/2)) ;
266
267     // Page 0: bit: T13STR in TCTR4H
268     CC6_vEnableShadowTransfer(CC6_TIMER_13);
269

```

```

270 if (note == 0) printf("note=c ");
271 else if (note == 1) printf("note=cis");
272 else if (note == 2) printf("note=d ");
273 else if (note == 3) printf("note=diss");
274 else if (note == 4) printf("note=e ");
275 else if (note == 5) printf("note=f ");
276 else if (note == 6) printf("note=fis");
277 else if (note == 7) printf("note=g ");
278 else if (note == 8) printf("note=gis");
279 else if (note == 9) printf("note=a ");
280 else if (note == 10) printf("note=ais");
281 else if (note == 11) printf("note=h ");
282 else if (note == 12) printf("note=---");
283 else
284     printf("note=???");
285
286 if (octave == 1) printf("' ");
287 else if (octave == 2) printf("' ' ");
288 else if (octave == 4) printf("' ' ' ");
289 else if (octave == 8) printf("' ' ' ' ");
290 else
291     printf("????");
292
293 printf(", T12-pv=%5u,", current_note_length);
294 printf("T12-p=%1.2f[s], ", current_note_length*85.3333/1000.0/1000.0);
295 printf("T13-pv=%5u,", T13_values[note]/octave);
296 printf("T13-f=%7.0f[Hz]\n", 1/((T13_values[note]/octave)*83.3333/1000.0/1000.0);
297
298 IO_vTogglePin(P3_1); // Show start of next note on Port 3 Pin 1
299 // Page 0: bit: T12RS in TCTR4L
300 CC6_vStartTmr(CC6_TIMER_12); // Set Timer 12 Run Set bit T12RS
301 }
302
303 // USER CODE END
304
305 }
306
307 // End of CCU6SR2 condition check
308
309 // USER CODE BEGIN (NodeI2,5)
310
311 // USER CODE END
312
313 SFR_PAGE(_su3, RST2); // restore the old SCU page
314 } // End of function SHINT_vIXINTR12Isr
315
316 // USER CODE BEGIN (SHARED_INT_General,10)
317
318 // USER CODE END
319
320
321

```




Registers used:

Adobe Reader - [XC88xCLM_UM_v1_1.pdf]

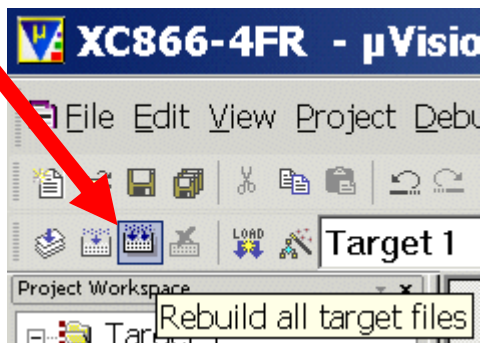
File Edit View Document Tools Window Help

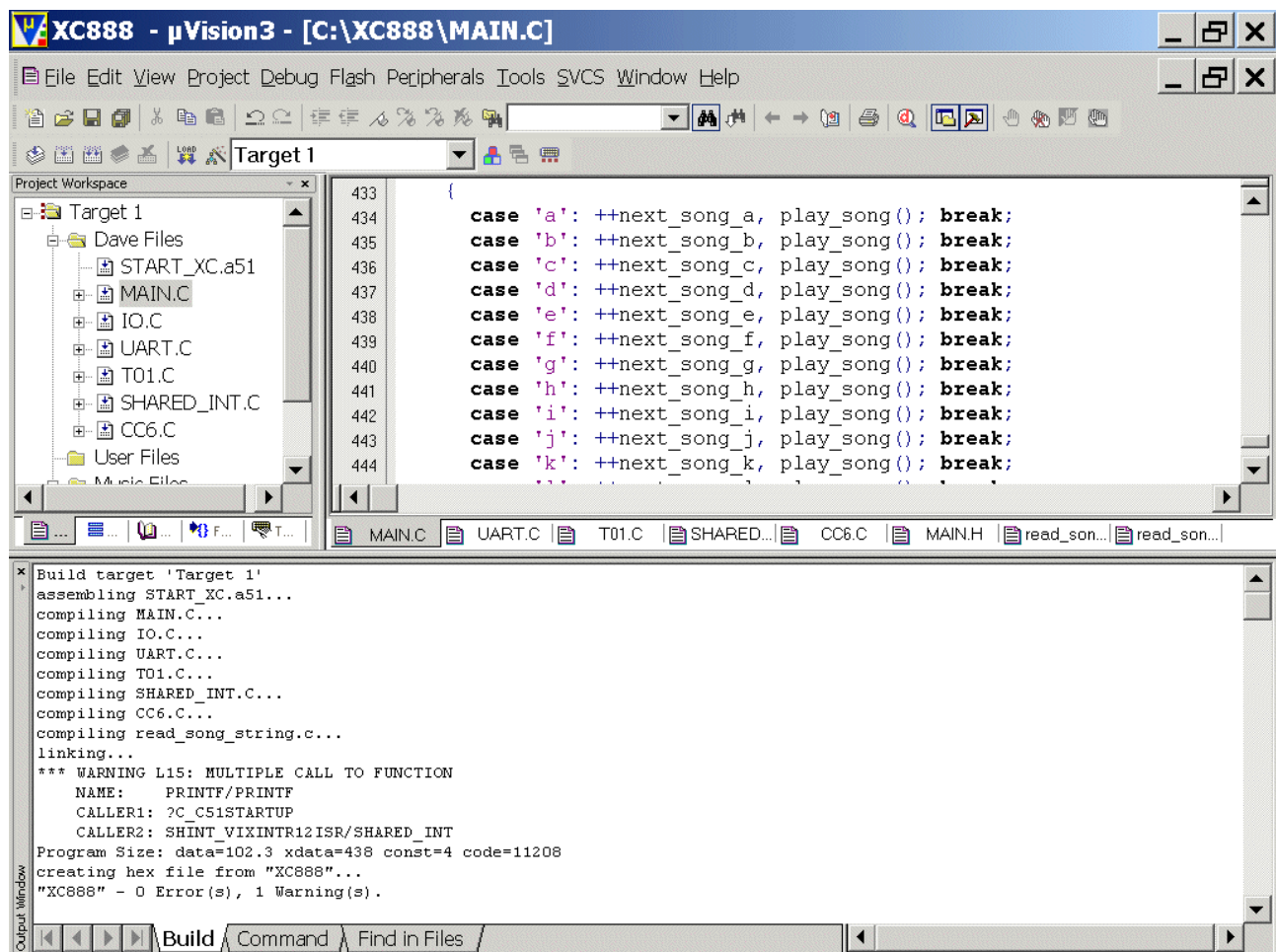
Table 14-3 SFR Address List for Pages 0-3

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

383 of 661

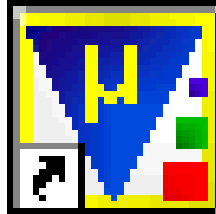
Generate your application program:

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

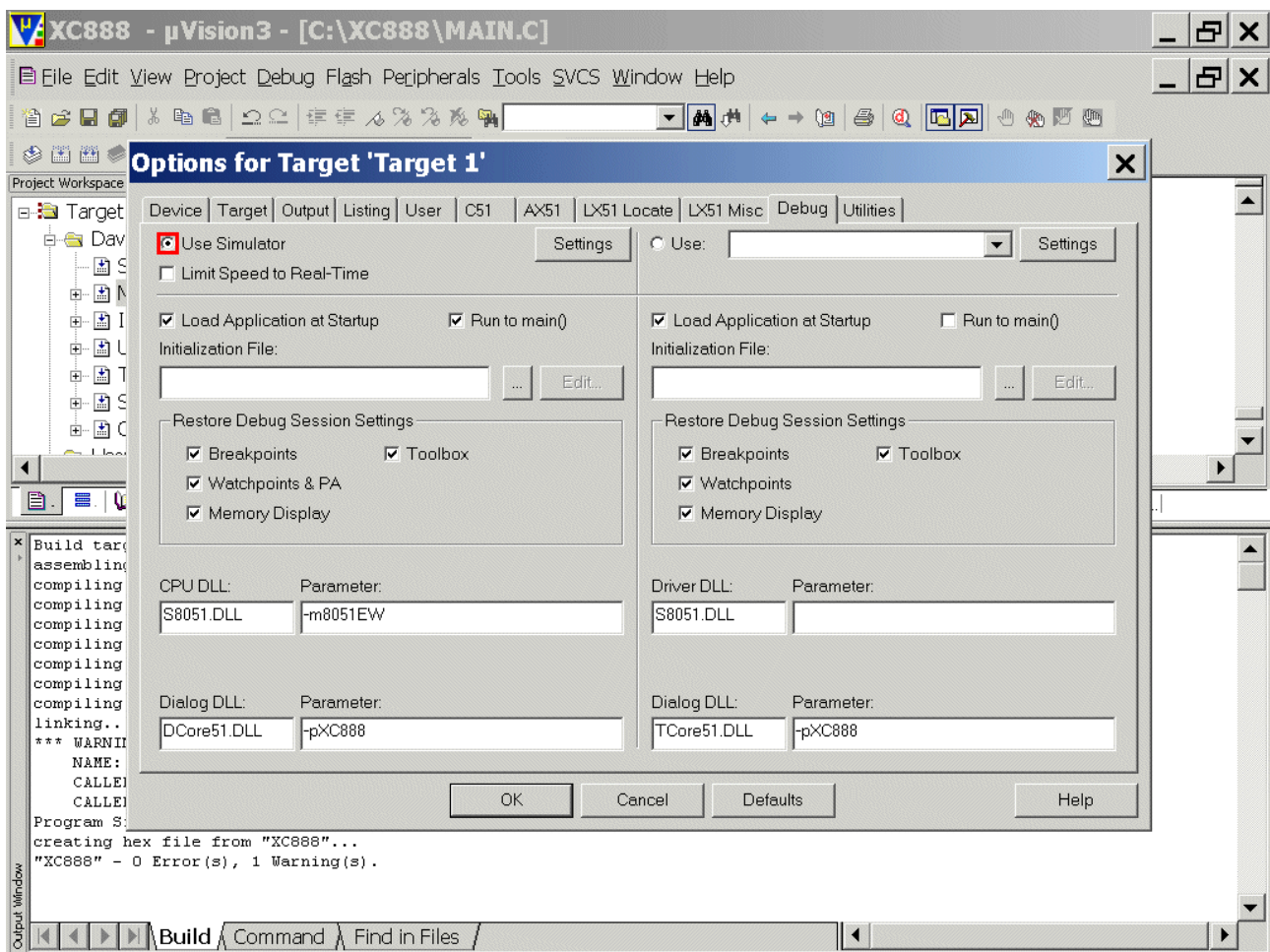


The screenshot shows the µVision3 IDE interface. The top window is titled "XC888 - µVision3 - [C:\XC888\MAIN.C]". The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The "Project Workspace" pane on the left shows a tree view of the project files, including "Target 1", "Dave Files", "START_XC.a51", "MAIN.C", "IO.C", "UART.C", "T01.C", "SHARED_INT.C", "CC6.C", "User Files", and "Music Files". The main editor window displays the source code for "MAIN.C", showing a series of case statements for playing different songs (a through k). The "Output Window" at the bottom shows the build process for "Target 1", including assembly and compilation steps, and a warning about a multiple call to the PRINTF/PRINTF function. The status bar at the bottom indicates "Build", "Command", and "Find in Files".

See the result:
Using the Simulator:

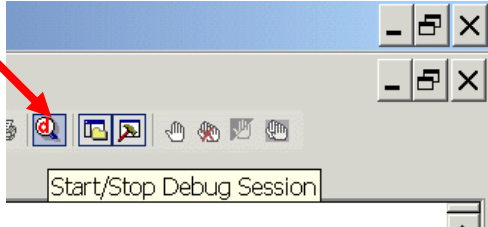
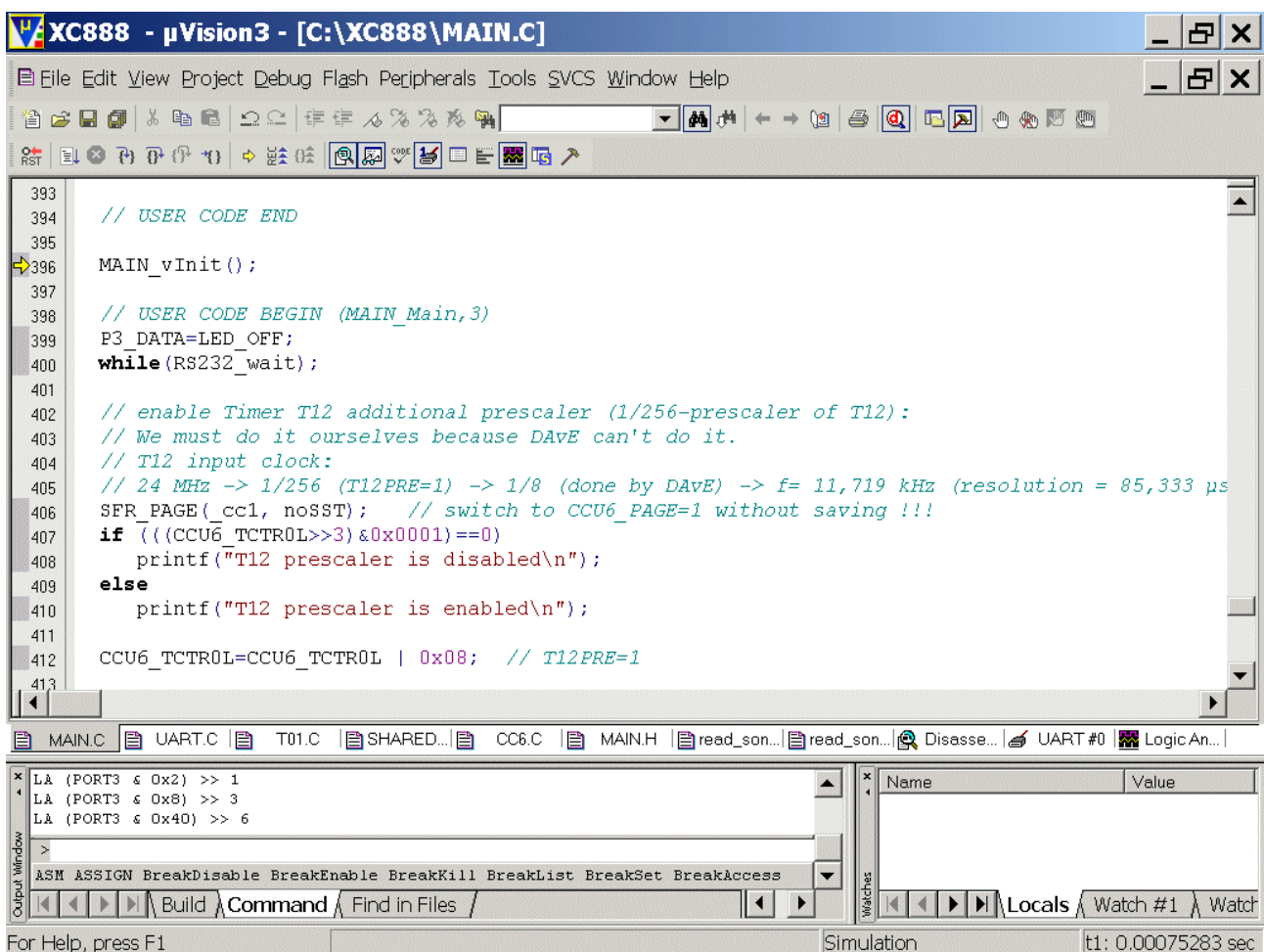


Check/Configure: ☒ Use Simulator



OK

Debug - Start/Stop Debug Session	or	click
----------------------------------	----	-------

XC888 - µVision3 - [C:\XC888\MAIN.C]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

```

393 // USER CODE END
394
395 MAIN_vInit();
396
397 // USER CODE BEGIN (MAIN_Main,3)
398 P3_DATA=LED_OFF;
399 while(RS232_wait);
400
401 // enable Timer T12 additional prescaler (1/256-prescaler of T12):
402 // We must do it ourselves because DAVE can't do it.
403 // T12 input clock:
404 // 24 MHz -> 1/256 (T12PRE=1) -> 1/8 (done by DAVE) -> f= 11,719 kHz (resolution = 85,333 µs
405 SFR_PAGE(cc1, nosST); // switch to CCU6_PAGE=1 without saving !!!
406 if (((CCU6_TCTR0L>>3)&0x0001)==0)
407     printf("T12 prescaler is disabled\n");
408 else
409     printf("T12 prescaler is enabled\n");
410
411 CCU6_TCTR0L=CCU6_TCTR0L | 0x08; // T12PRE=1
412
413

```

MAIN.C | UART.C | T01.C | SHARED... | CC6.C | MAIN.H | read_son... | read_son... | Disasse... | UART #0 | Logic An...

LA (PORT3 & 0x2) >> 1
LA (PORT3 & 0x8) >> 3
LA (PORT3 & 0x40) >> 6

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

Build Command Find in Files

For Help, press F1

Simulation

t1: 0.00075283 sec

Reconfigure the Software-Simulator-LGA (Logic Analyzer):

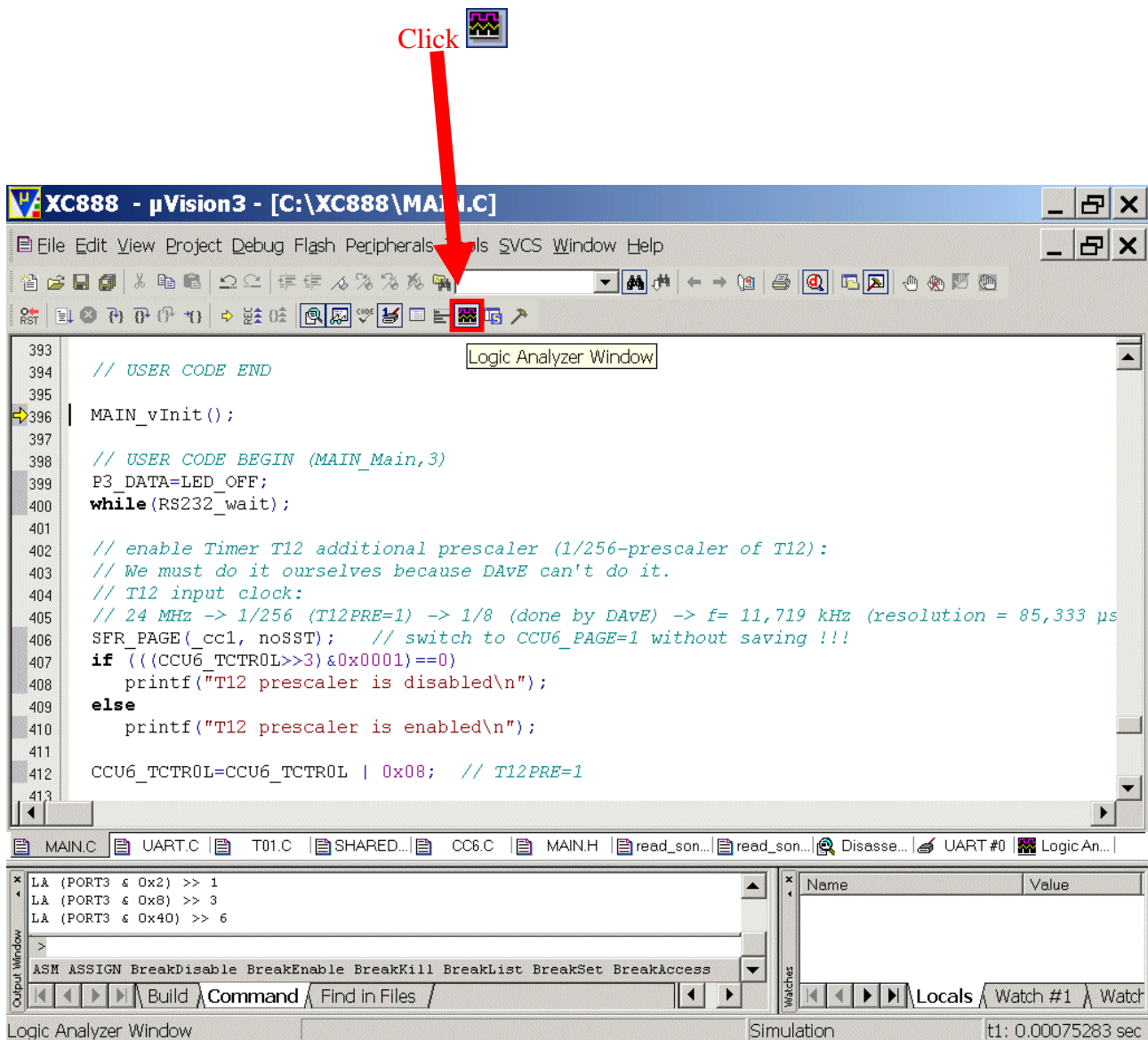
We are going to configure the following Pins for the Keil-Software-Simulator-LGA:

Port_3 pins used by our PWM module:

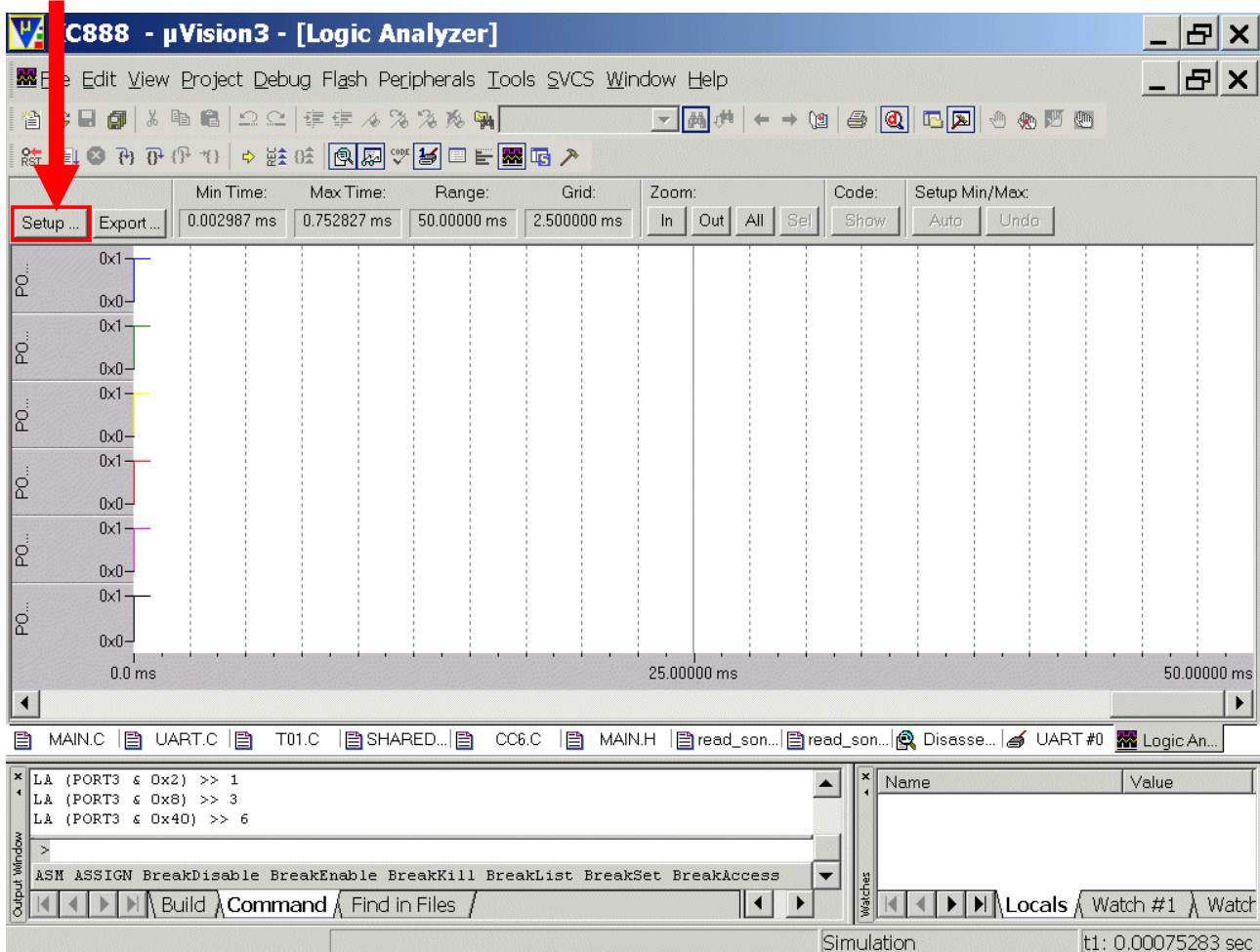
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

Port_3 pins used as GPIO:

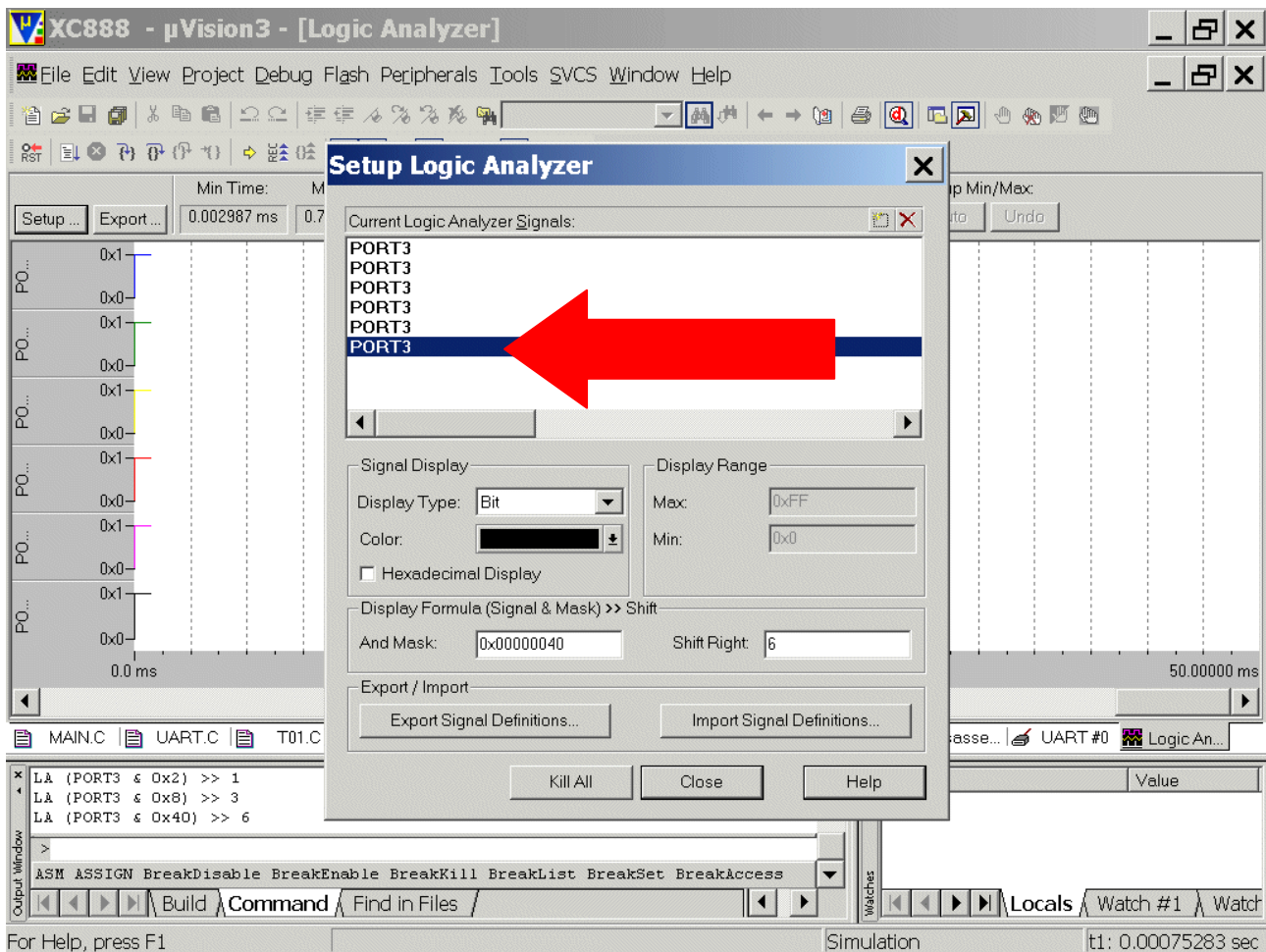
Port Lines	Function	Comment
P3.1	Show start of next note	Toggled via Software
P3.6	„use: program running signal“	Toggled via Timer_0 ISR



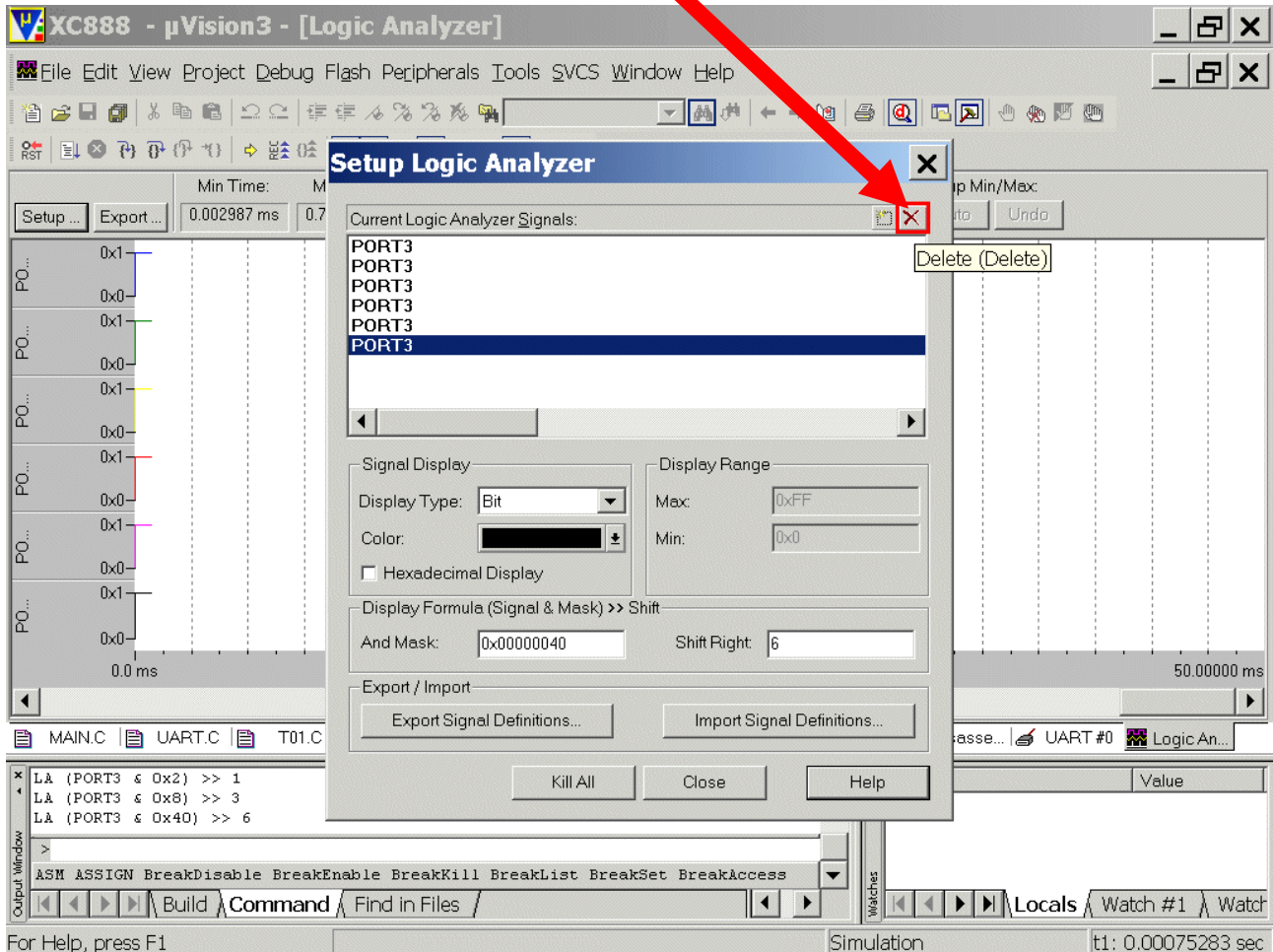
Click Setup ...



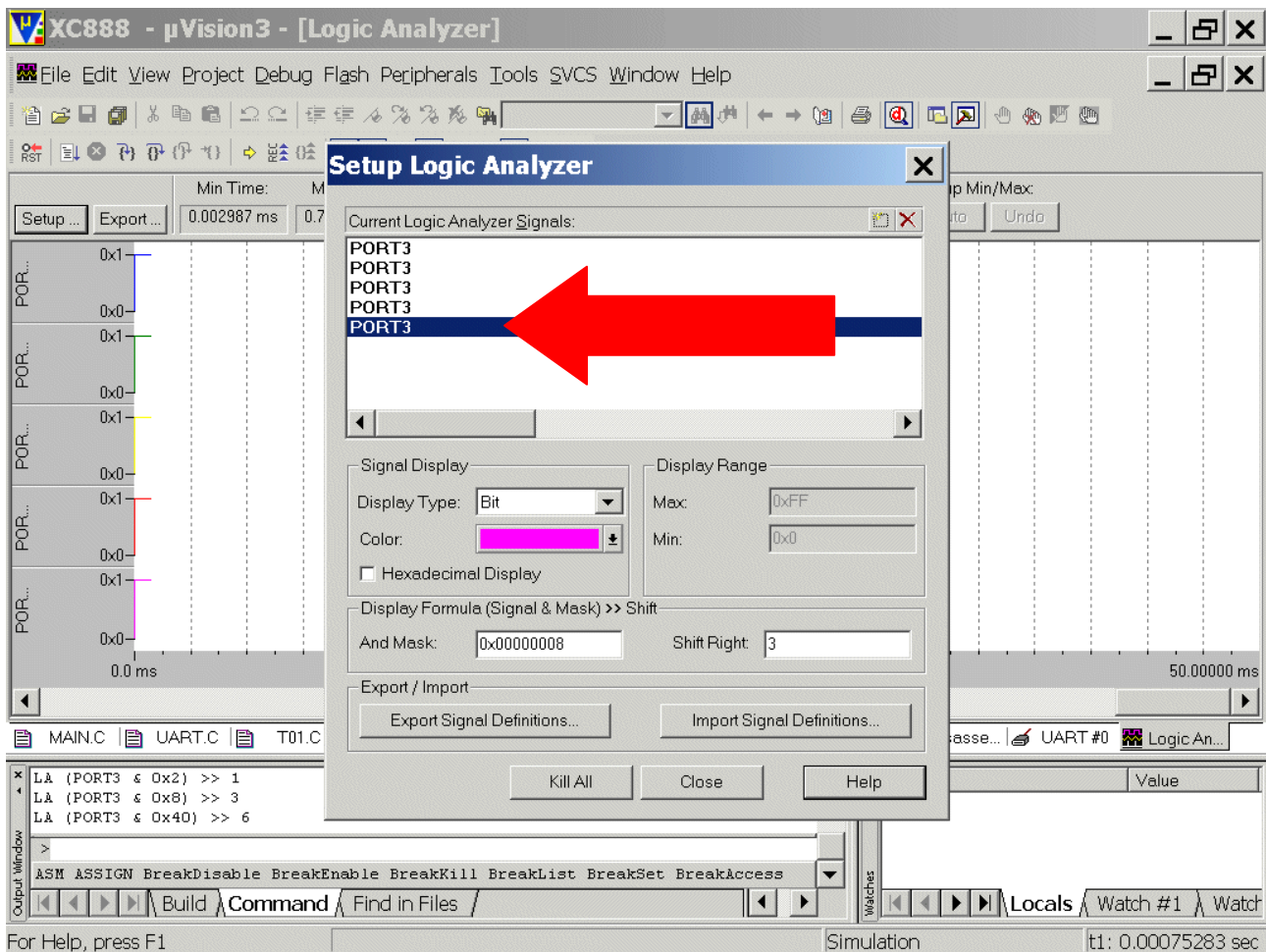
Mark/Click PORT3:



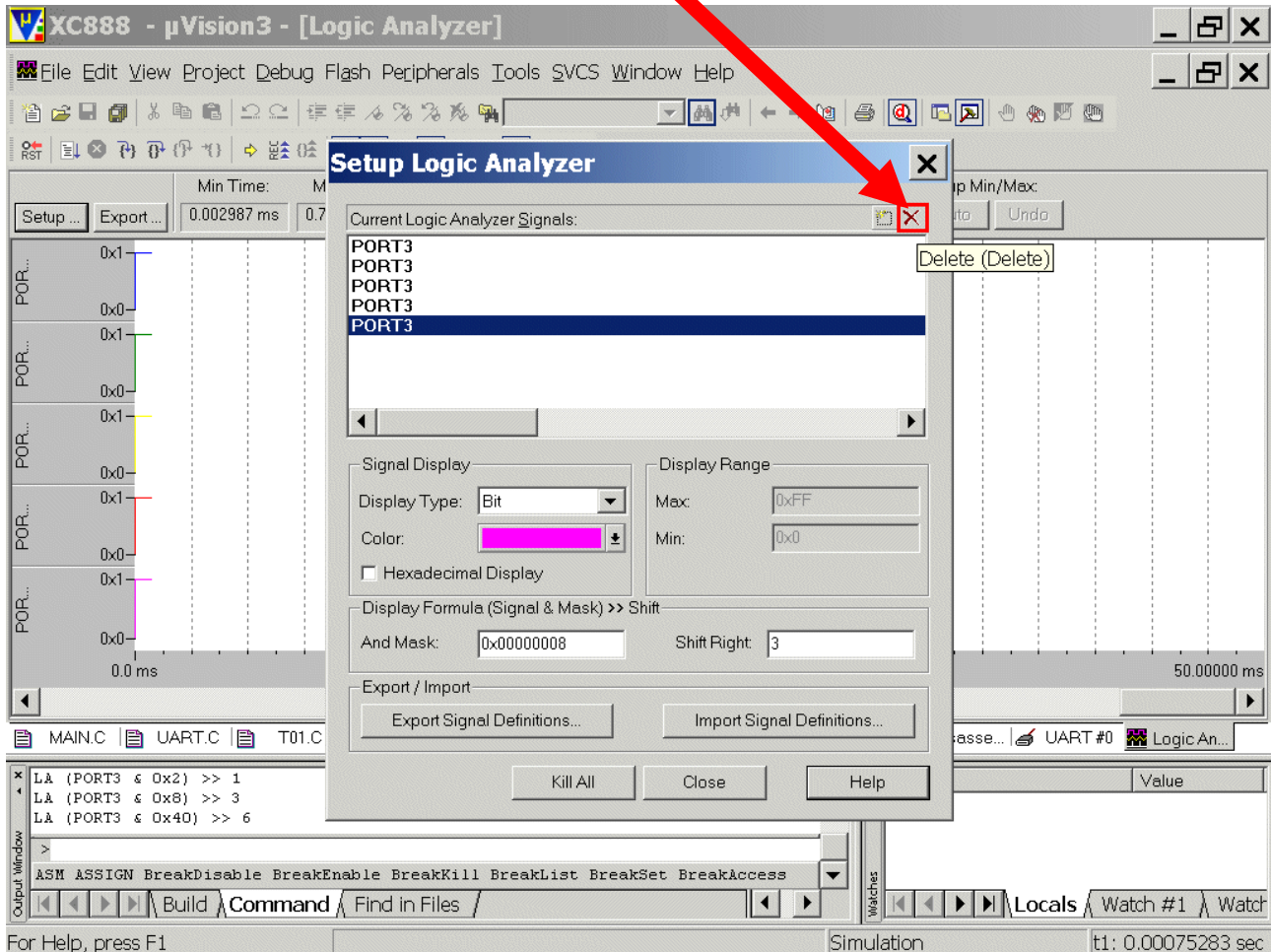
Click  (Delete Button)



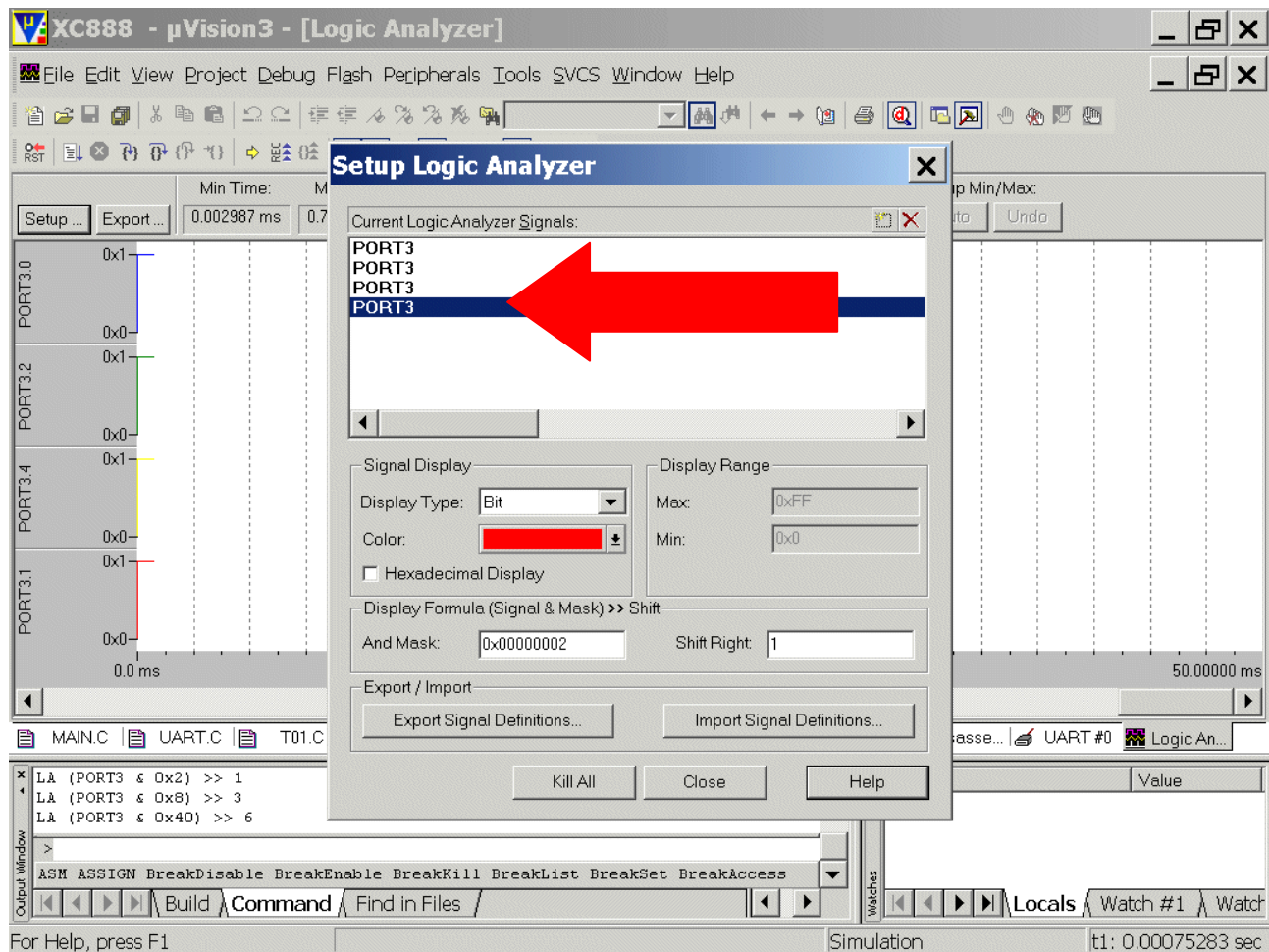
Mark/Click PORT3:



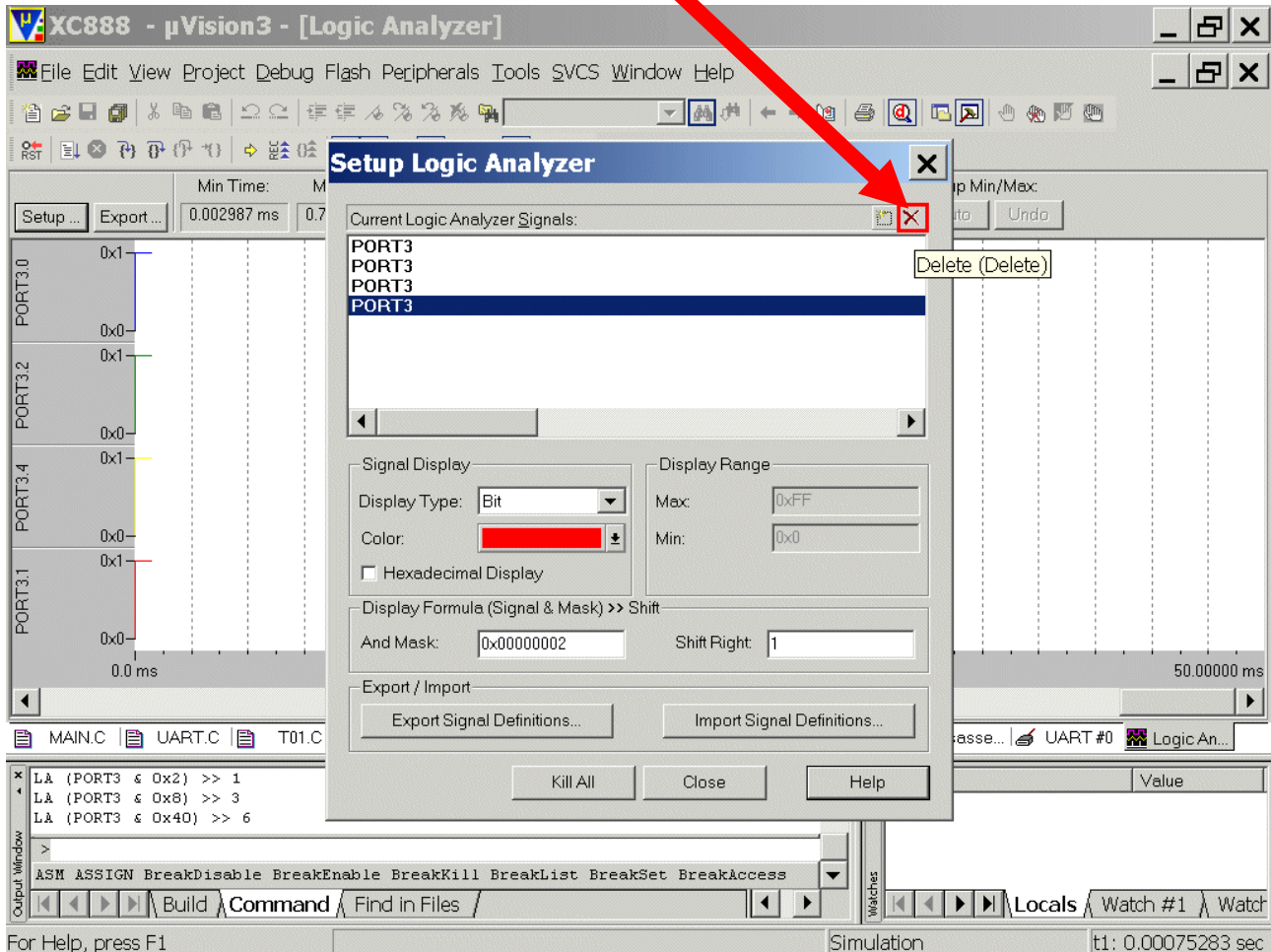
Click  (Delete Button)



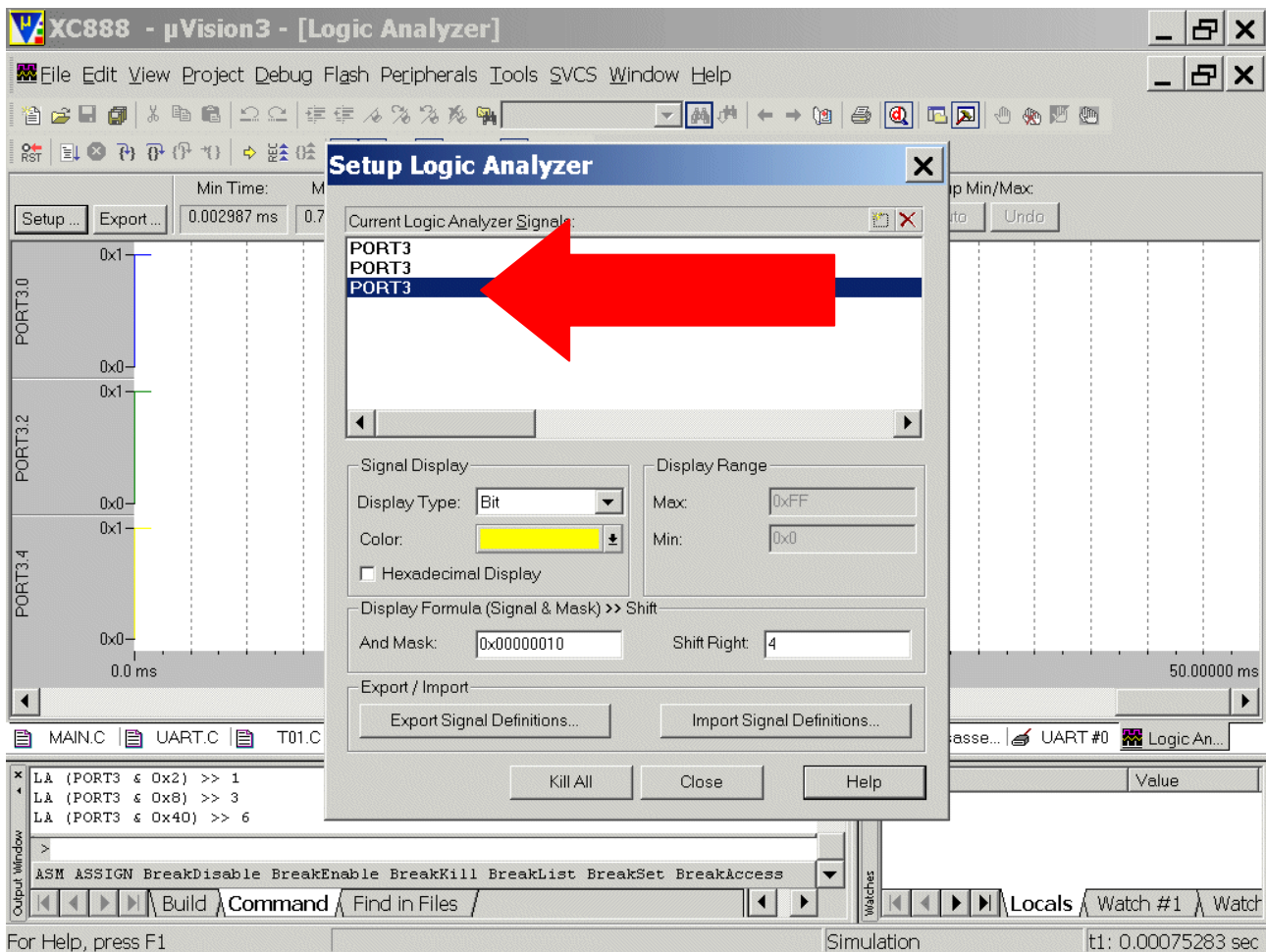
Mark/Click PORT3:



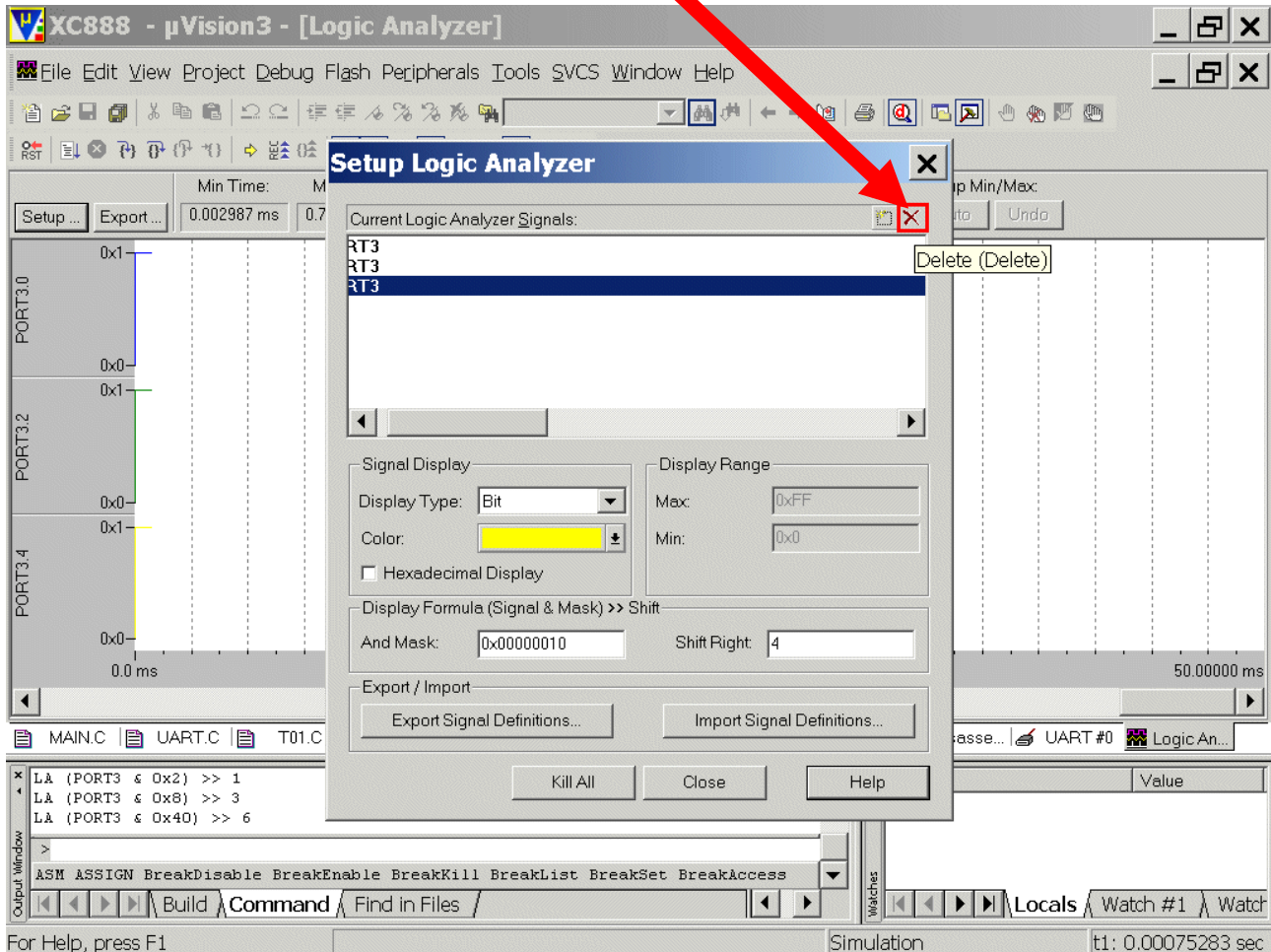
Click  (Delete Button)



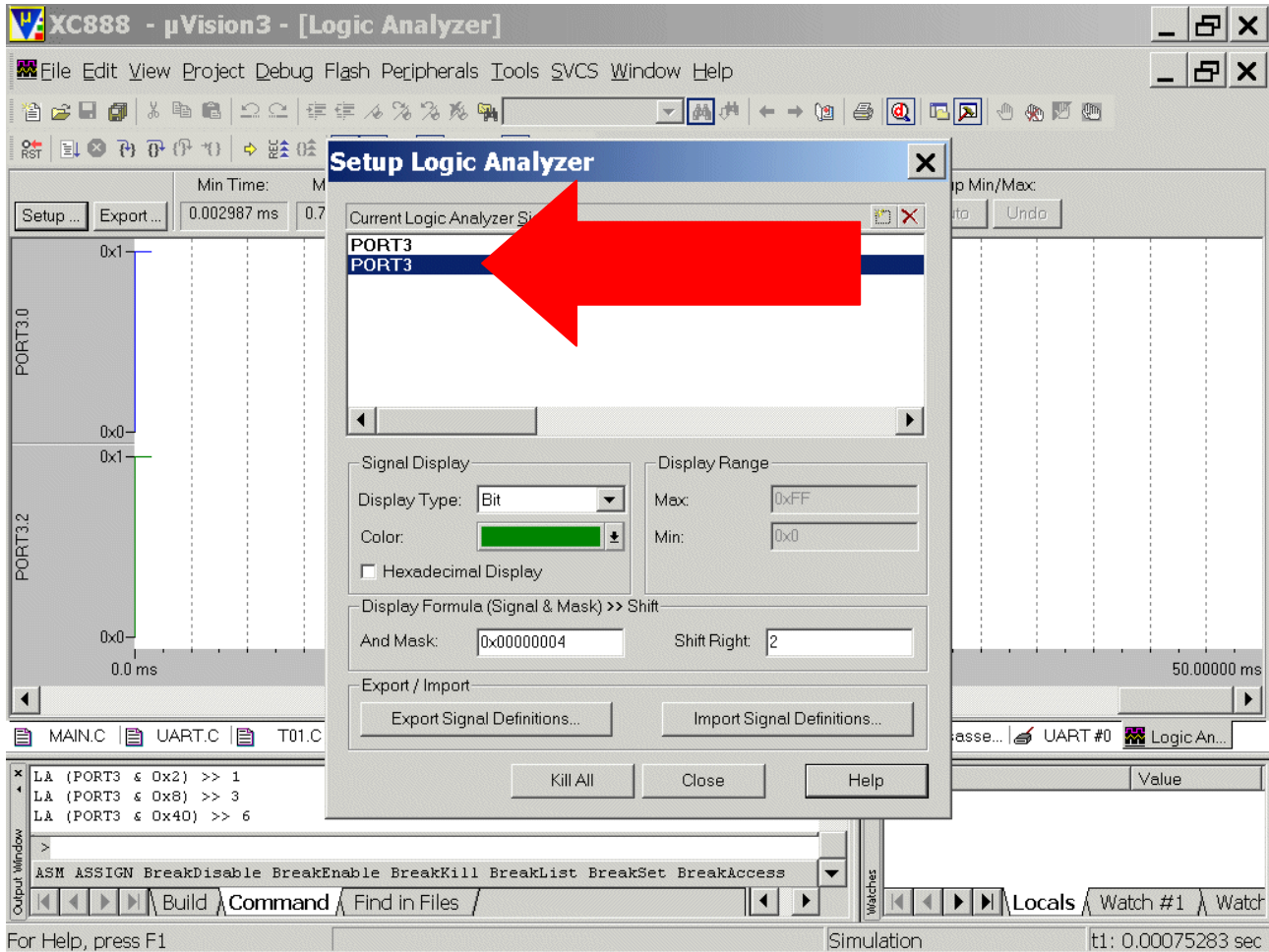
Mark/Click PORT3:

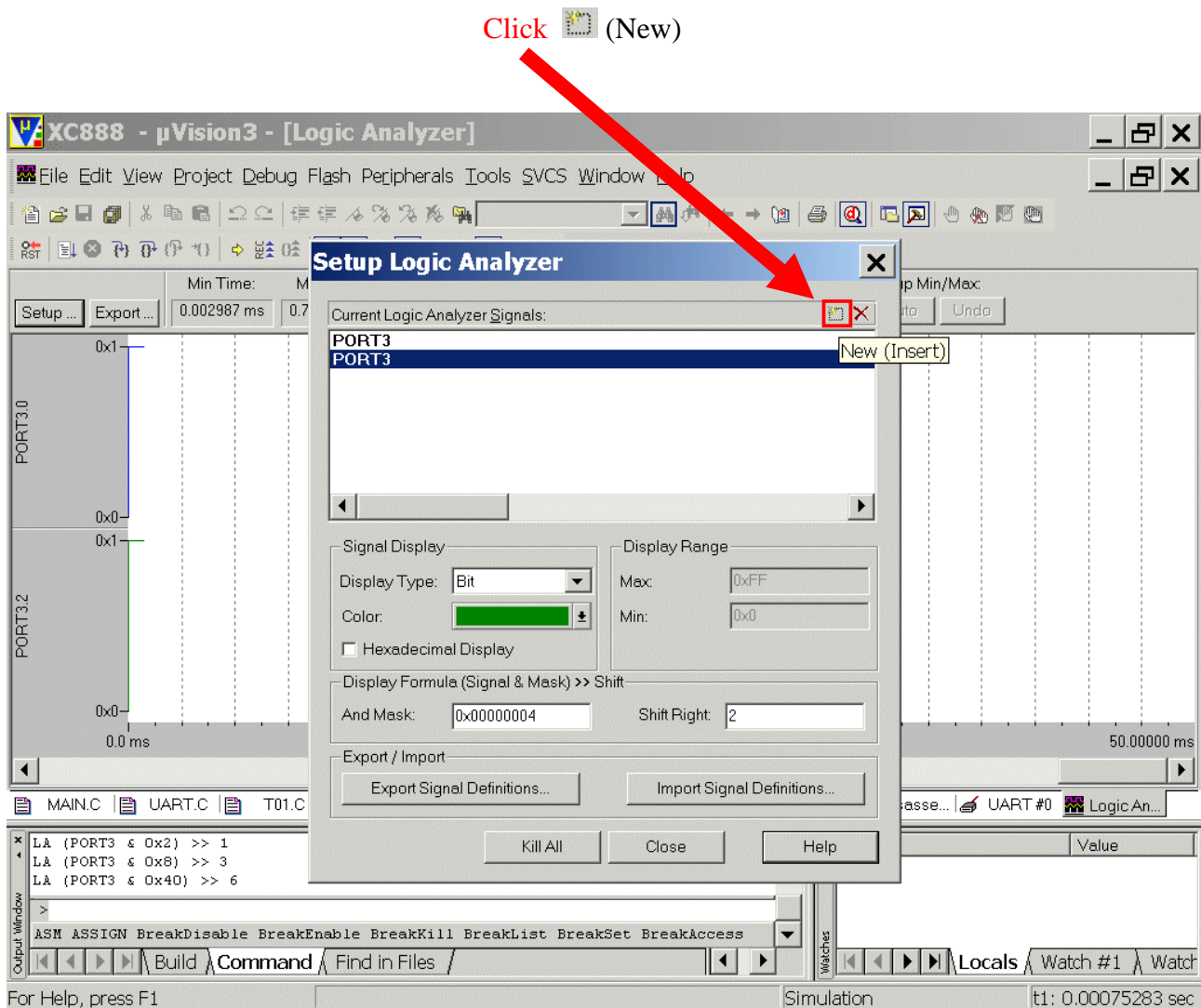


Click  (Delete Button)



Mark/Click PORT3:





Note:

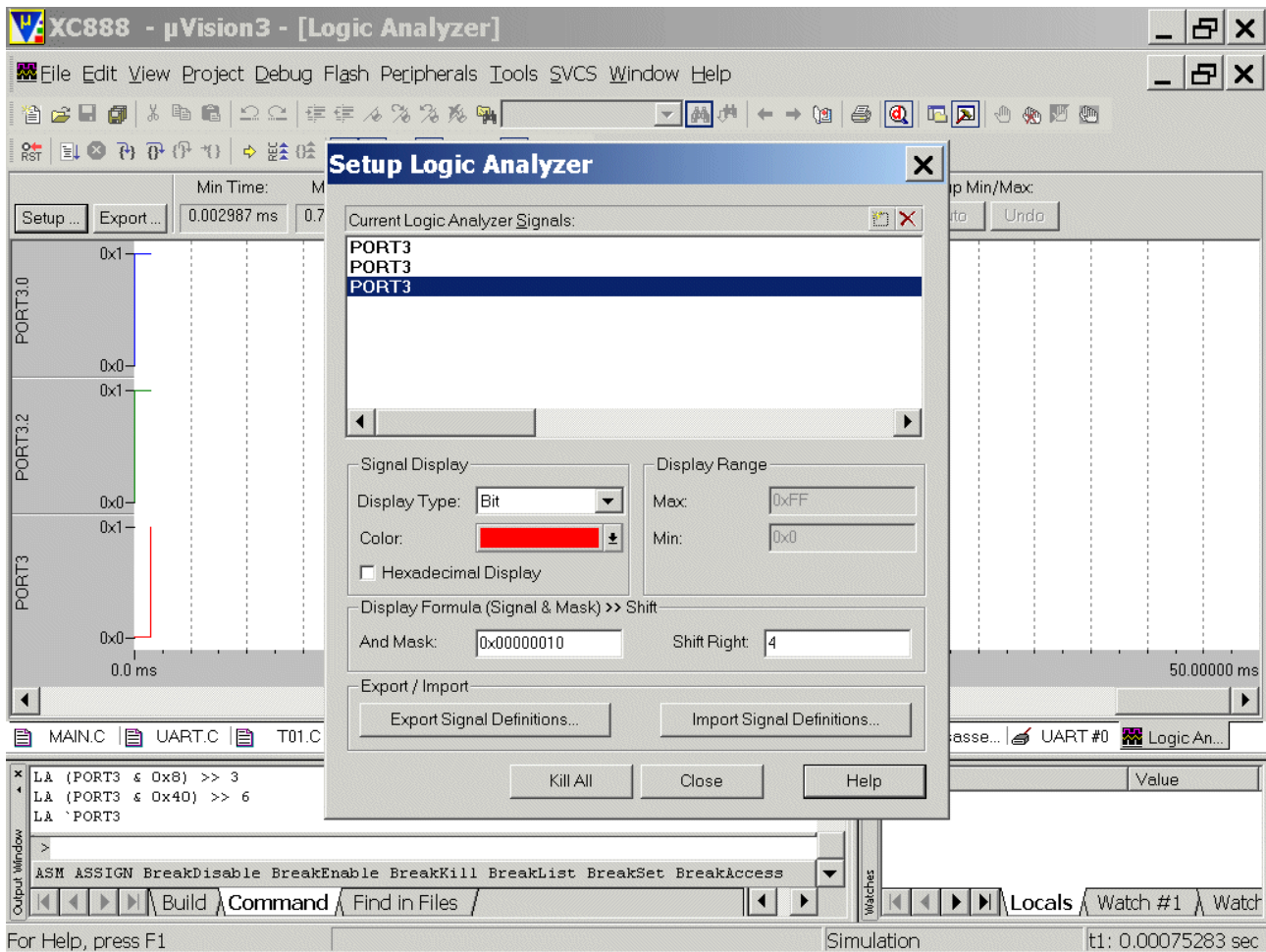
Port_3 pins used by our PWM module:

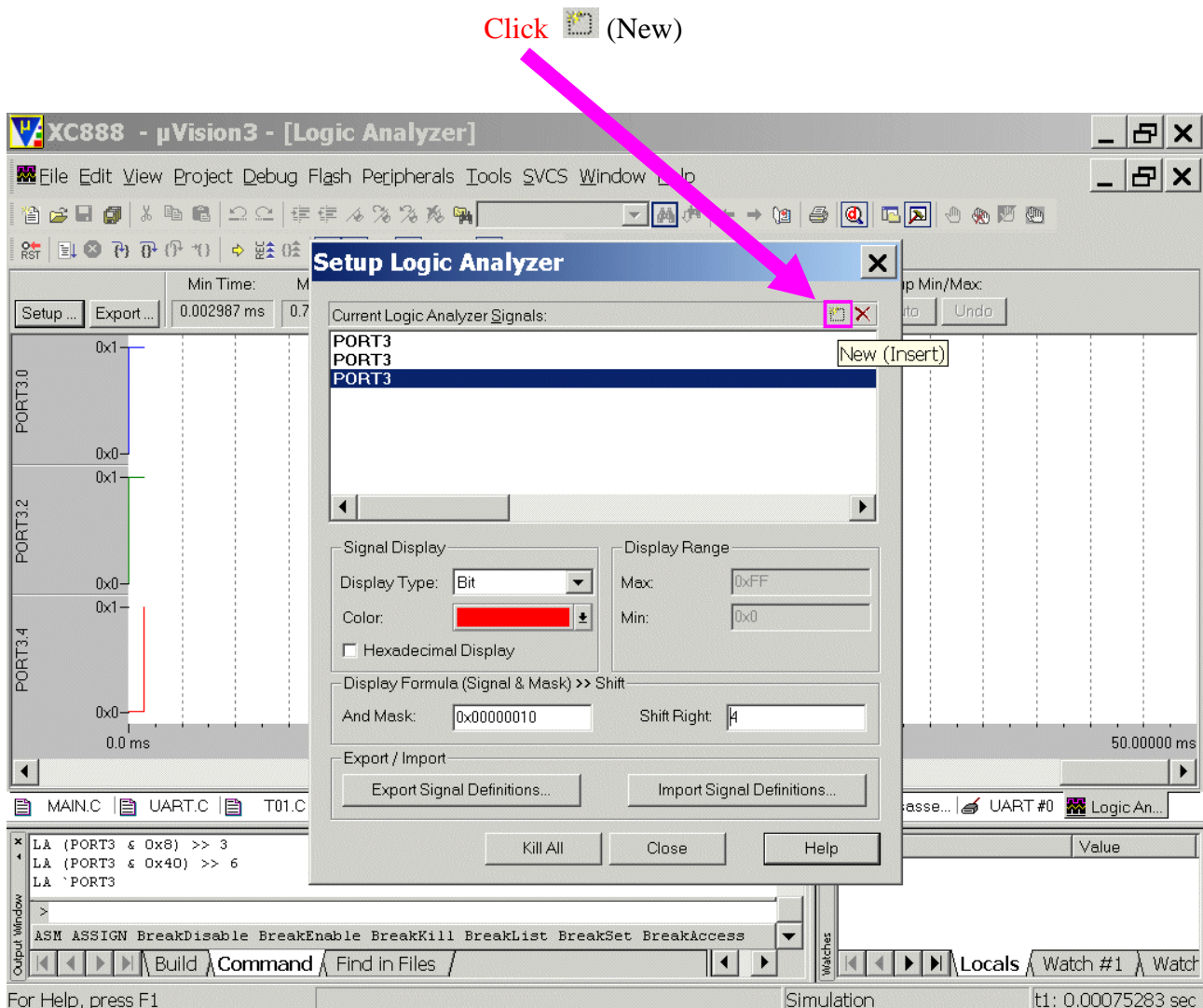
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

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

Insert PORT 3 <RETURN>
Display Type: select Bit
Color: select "red"
And Mask: insert 0x10
Shift Right: insert 4





Note:

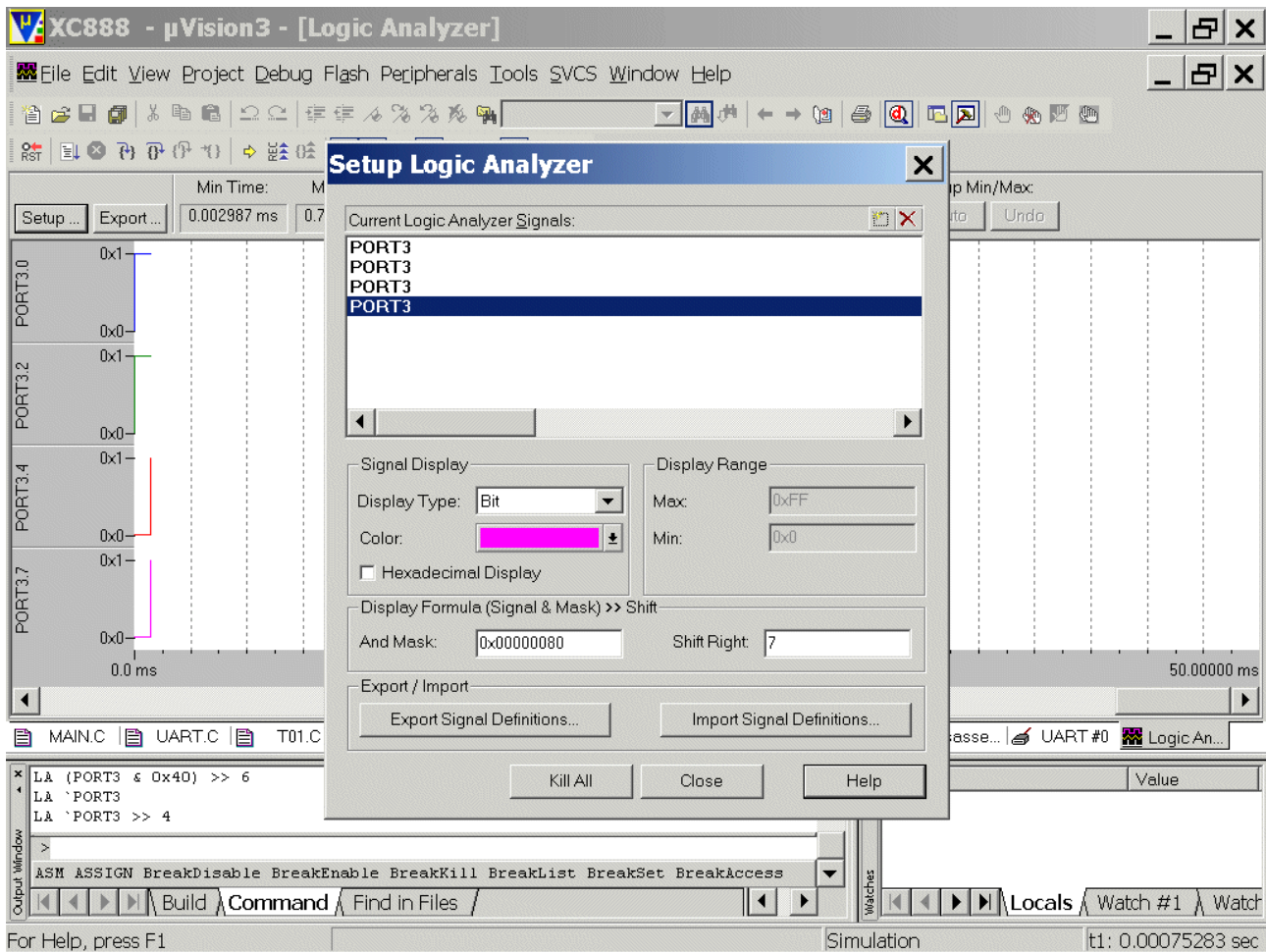
Port_3 pins used by our PWM module:

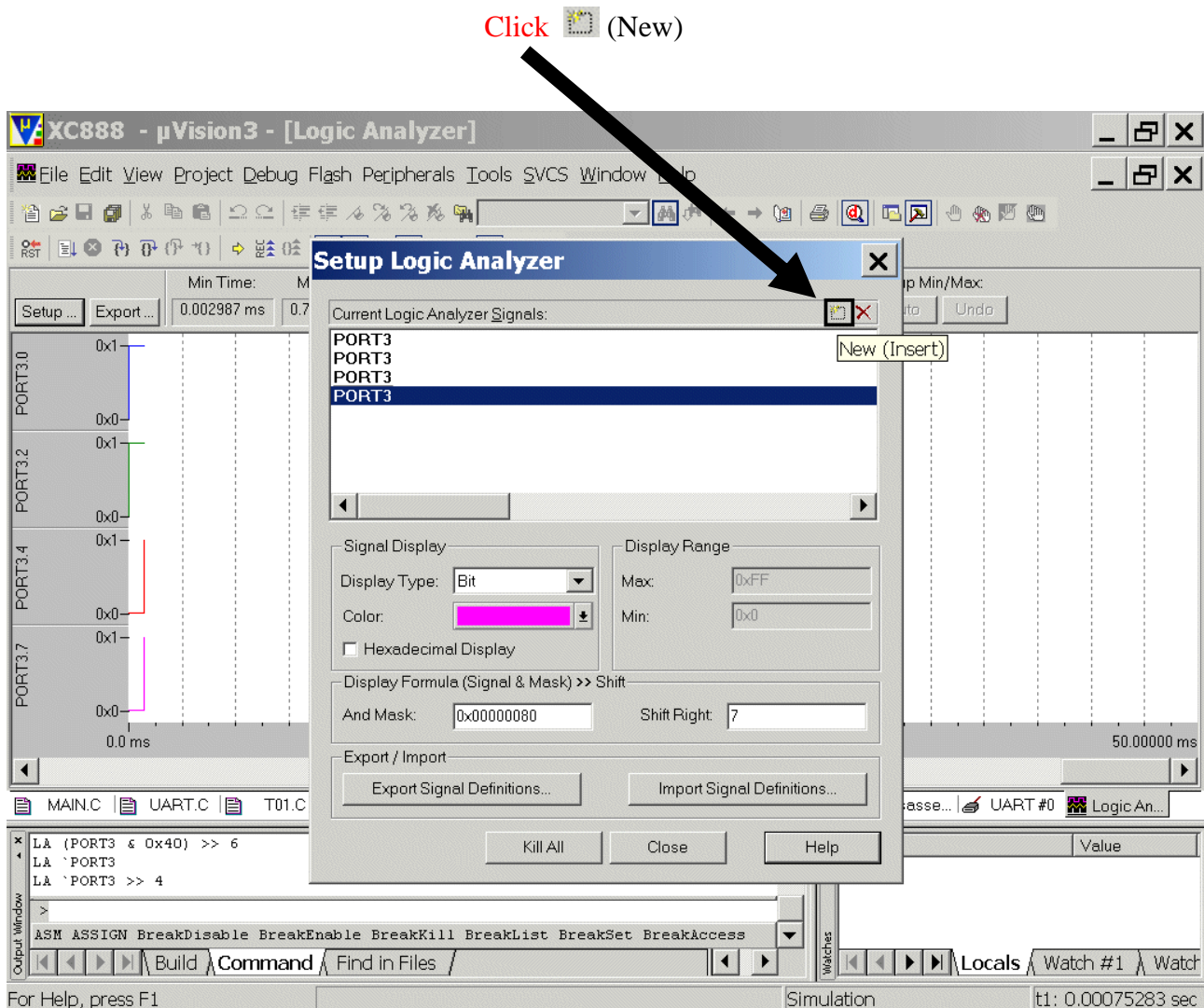
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COUT63_0	50 (note frequency, Timer_13)

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

Insert PORT 3 <RETURN>
Display Type: select Bit
Color: select "magenta"
And Mask: insert 0x80
Shift Right: insert 7





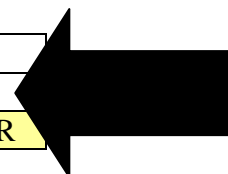
Note:

Port_3 pins used by our PWM module:

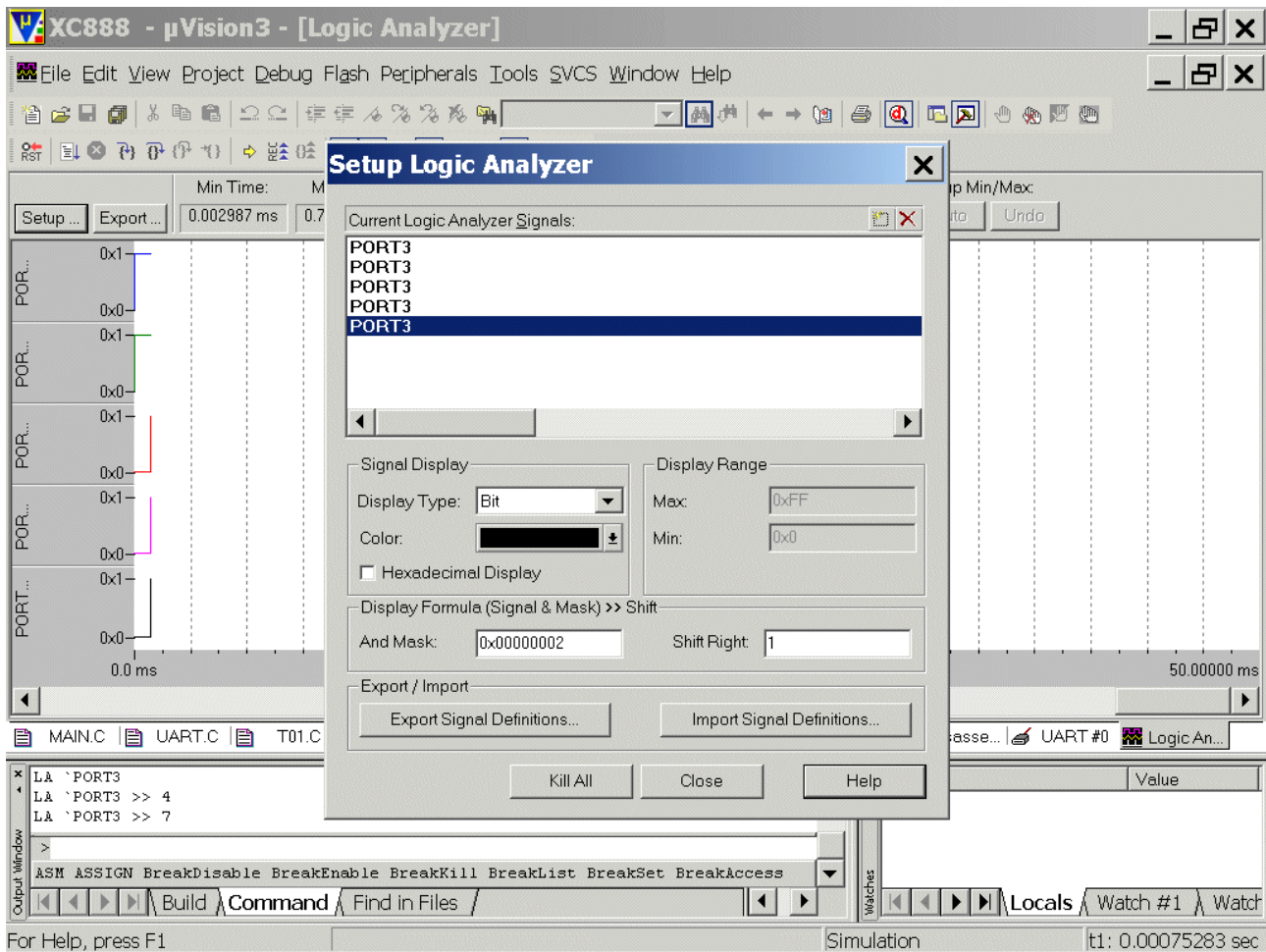
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (note length, Timer_12)
P3.2	CC61_0	100 (note length, Timer_12)
P3.4	CC62_0	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P3.7	COU63_0	50 (note frequency, Timer_13)

Port_3 pins used as GPIO:

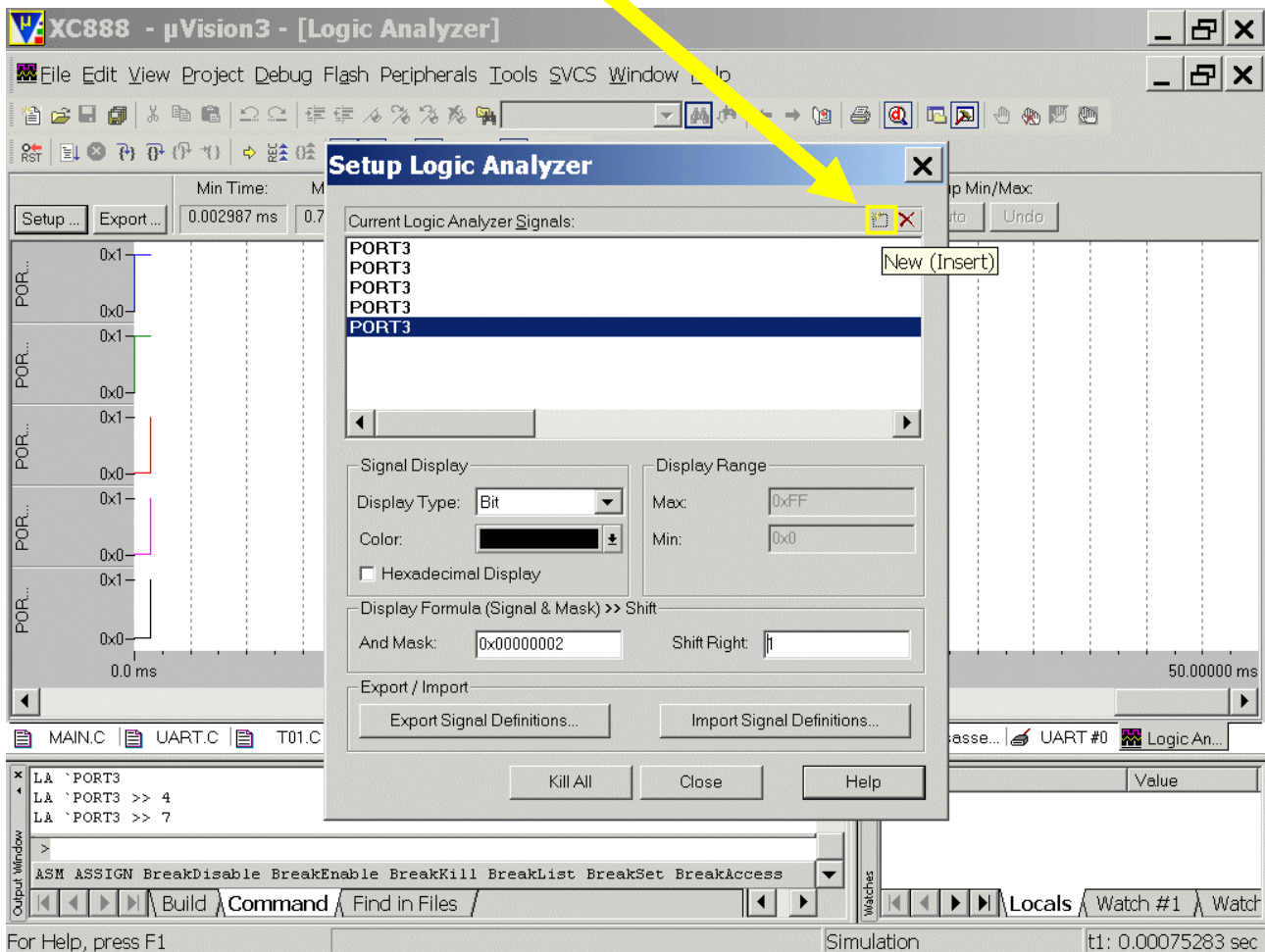
Port Lines	Function	Comment
P3.1	Show start of next note	Toggled via Software
P3.6	„use: program running signal“	Toggled via Timer_0 ISR



Insert PORT 3 <RETURN>
Display Type: select Bit
Color: select "black"
And Mask: insert 0x2
Shift Right: insert 1



Click  (New)



Note:

Port_3 pins used by our PWM module:

Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P3.0	CC60_0	5 (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

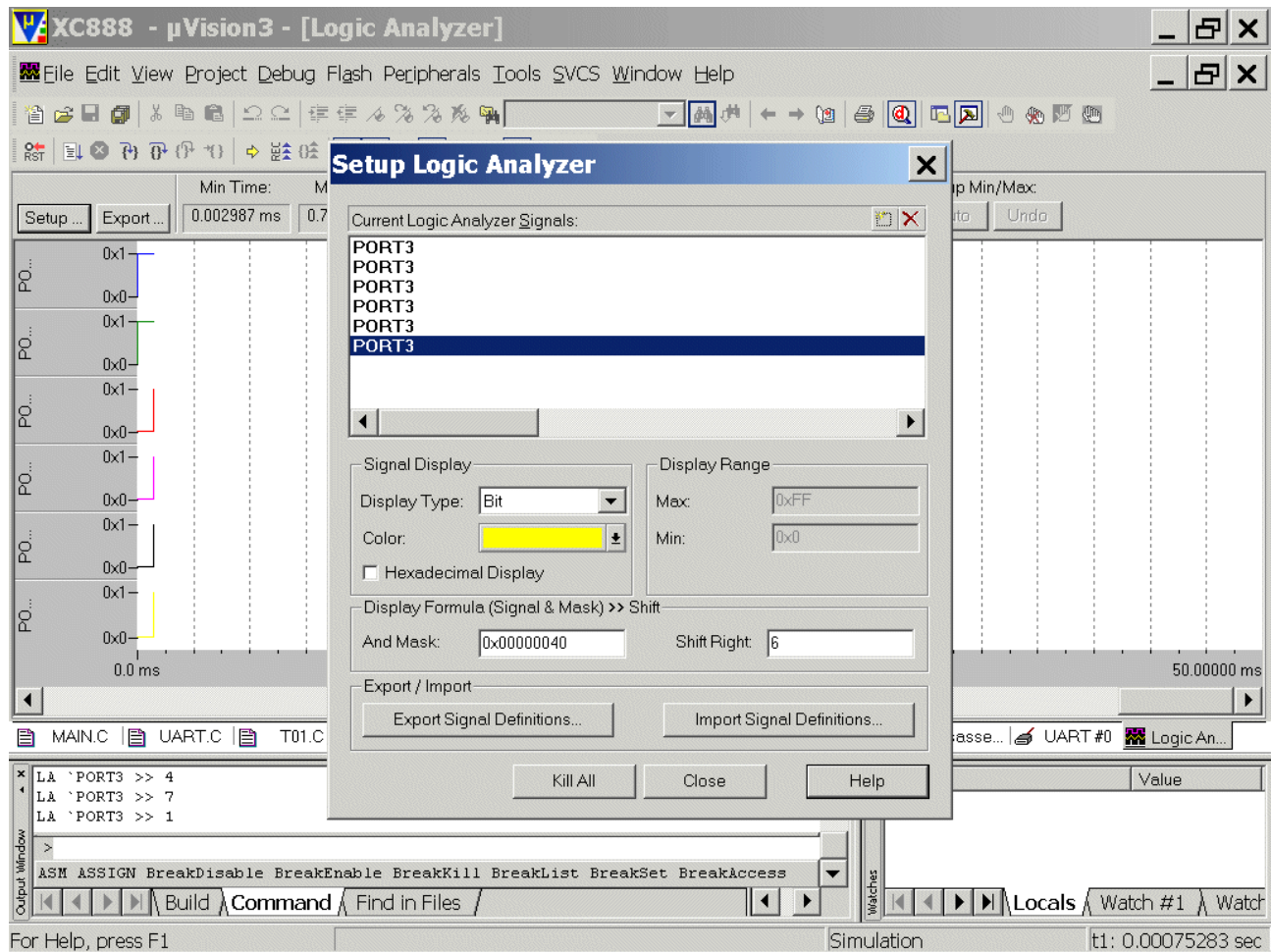
Insert PORT 3 <RETURN>

Display Type: select Bit

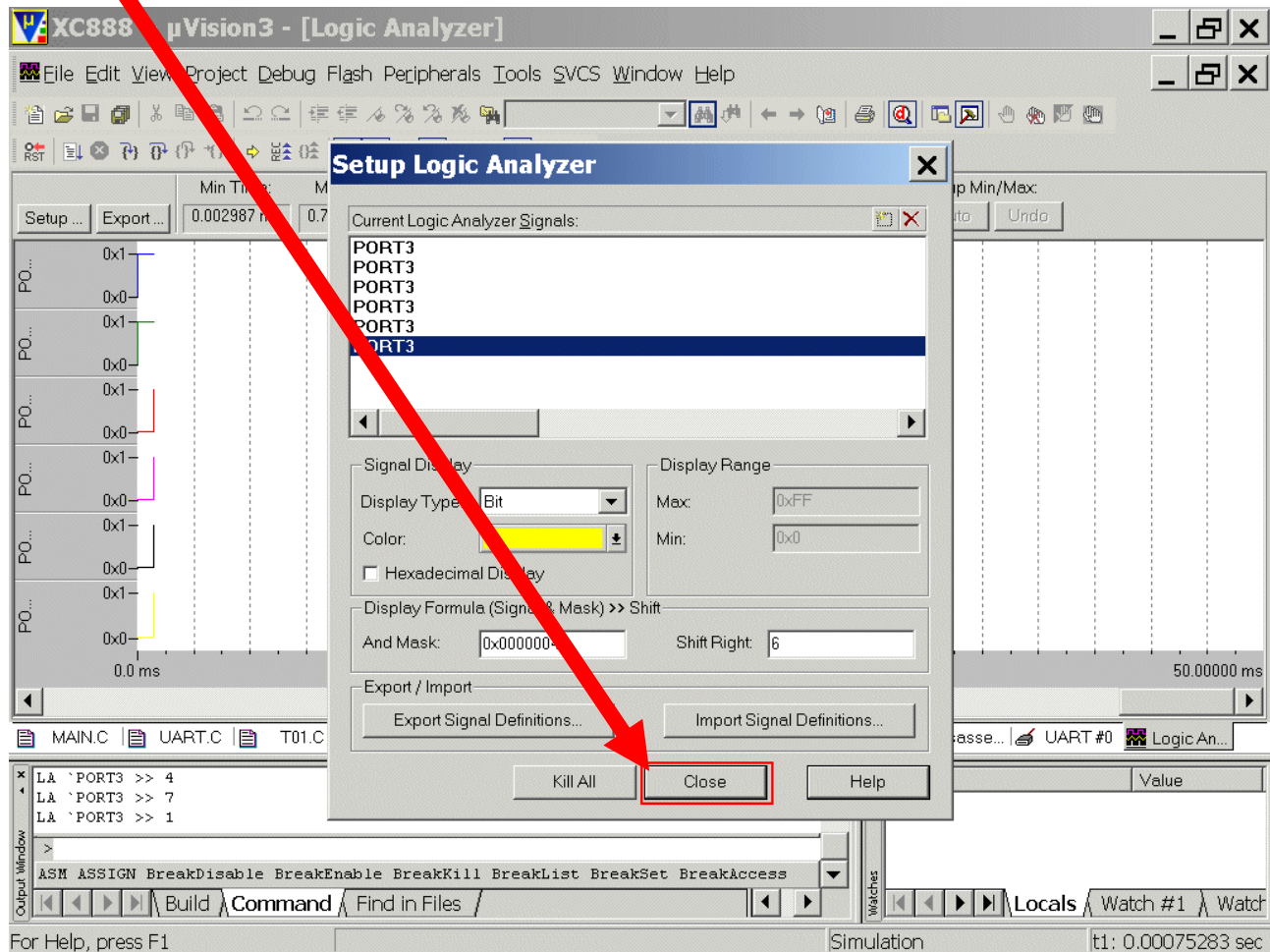
Color: select "yellow"

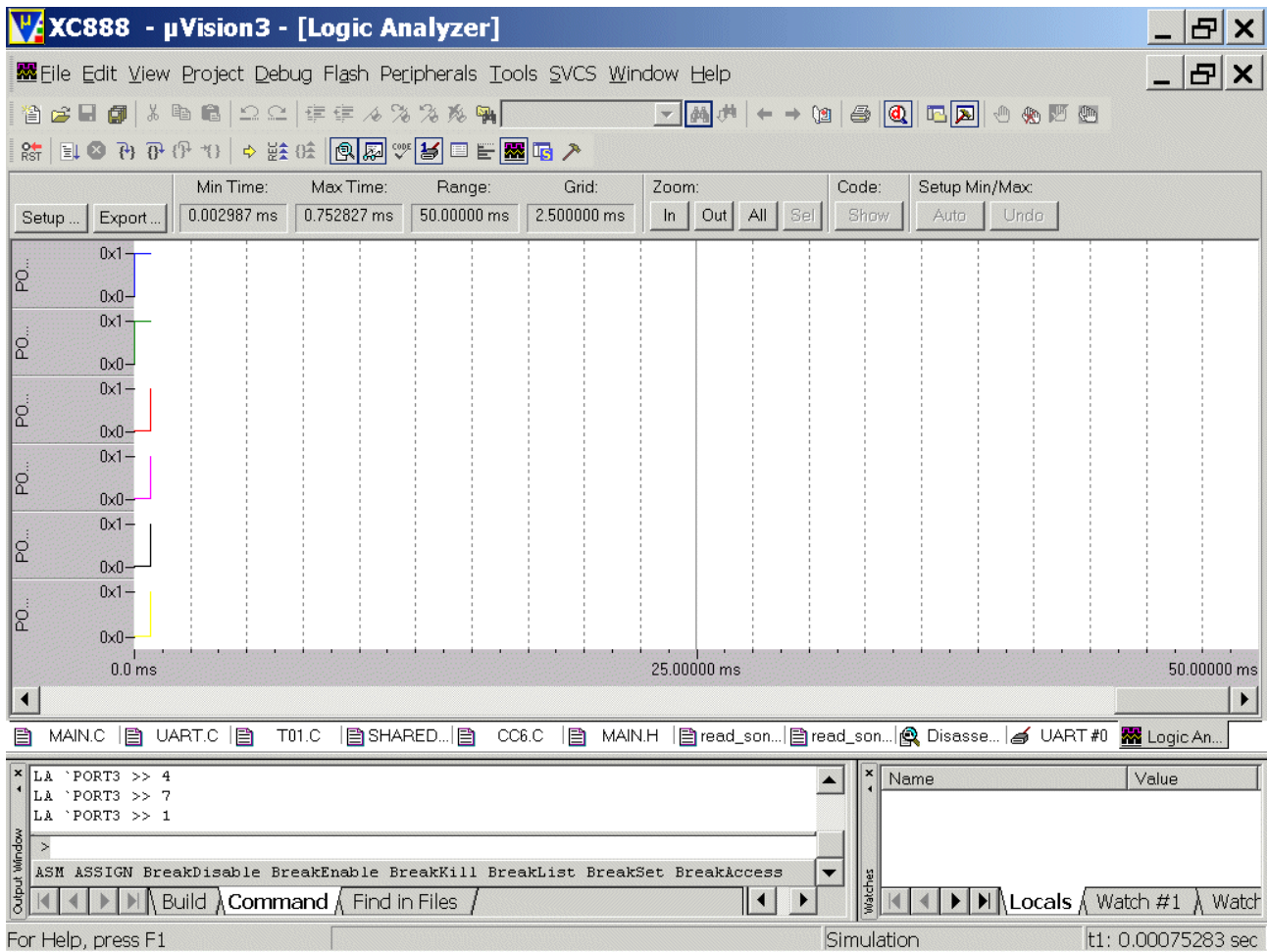
And Mask: insert 0x40

Shift Right: insert 6

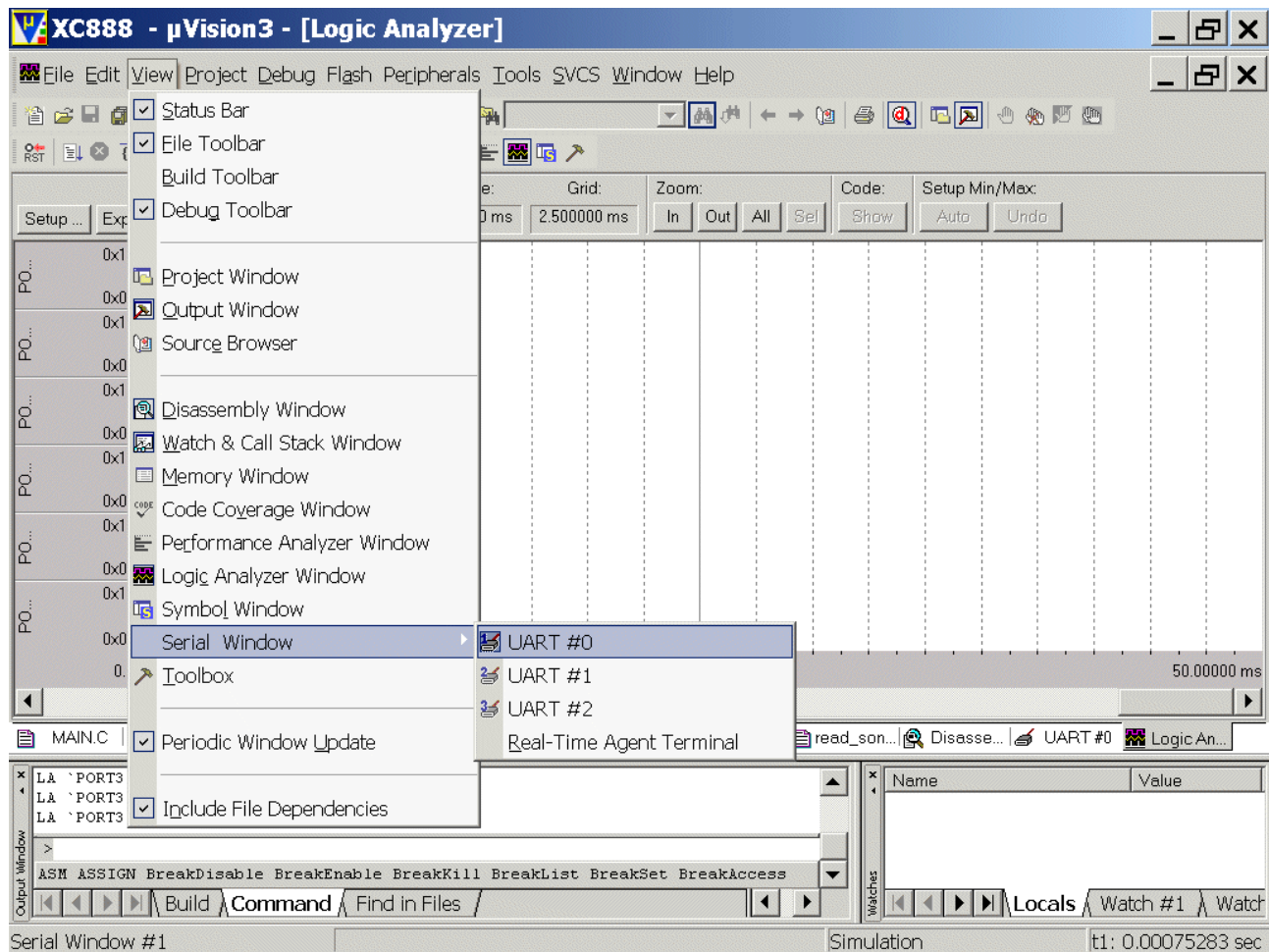


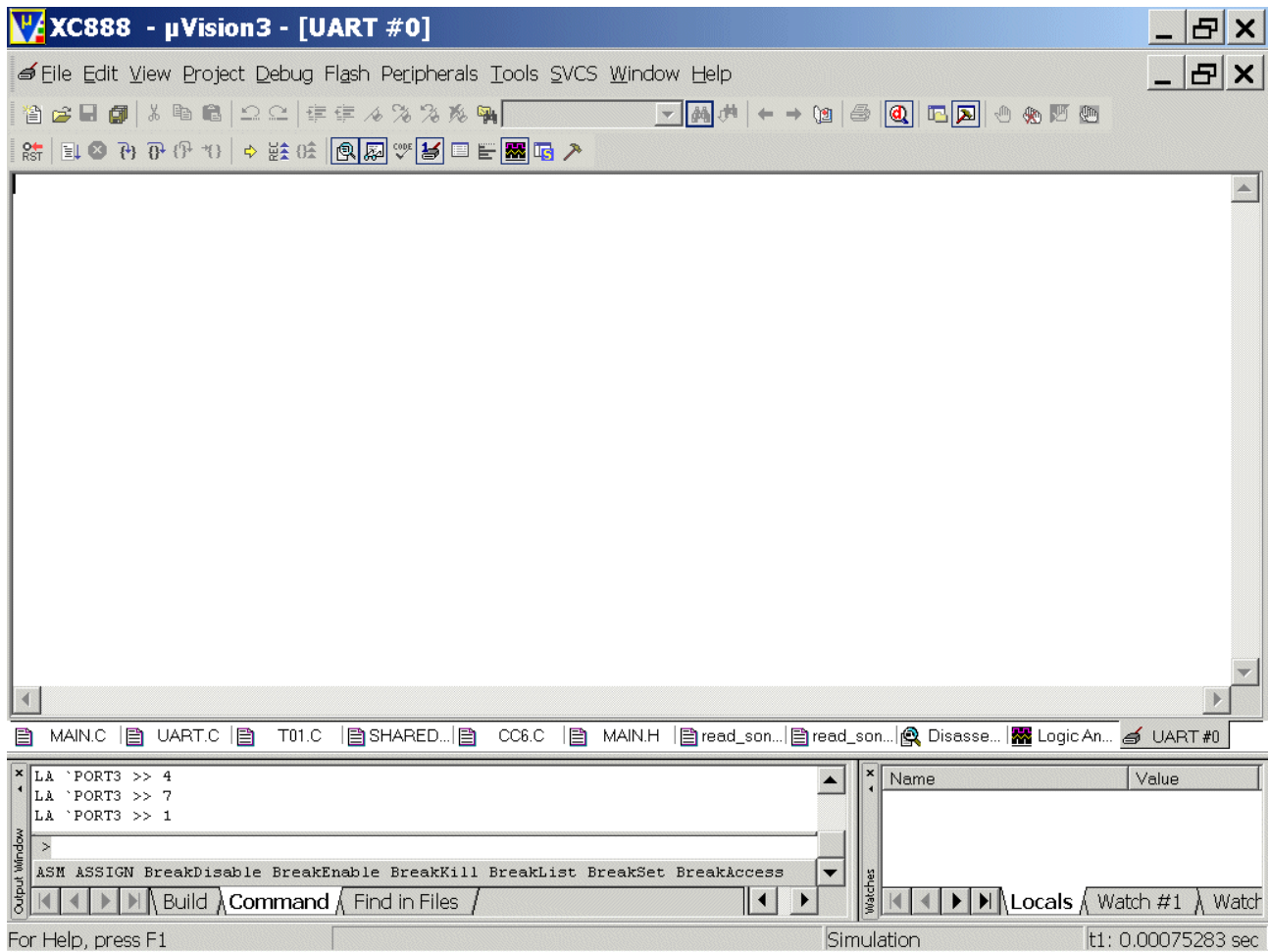
Click Close:



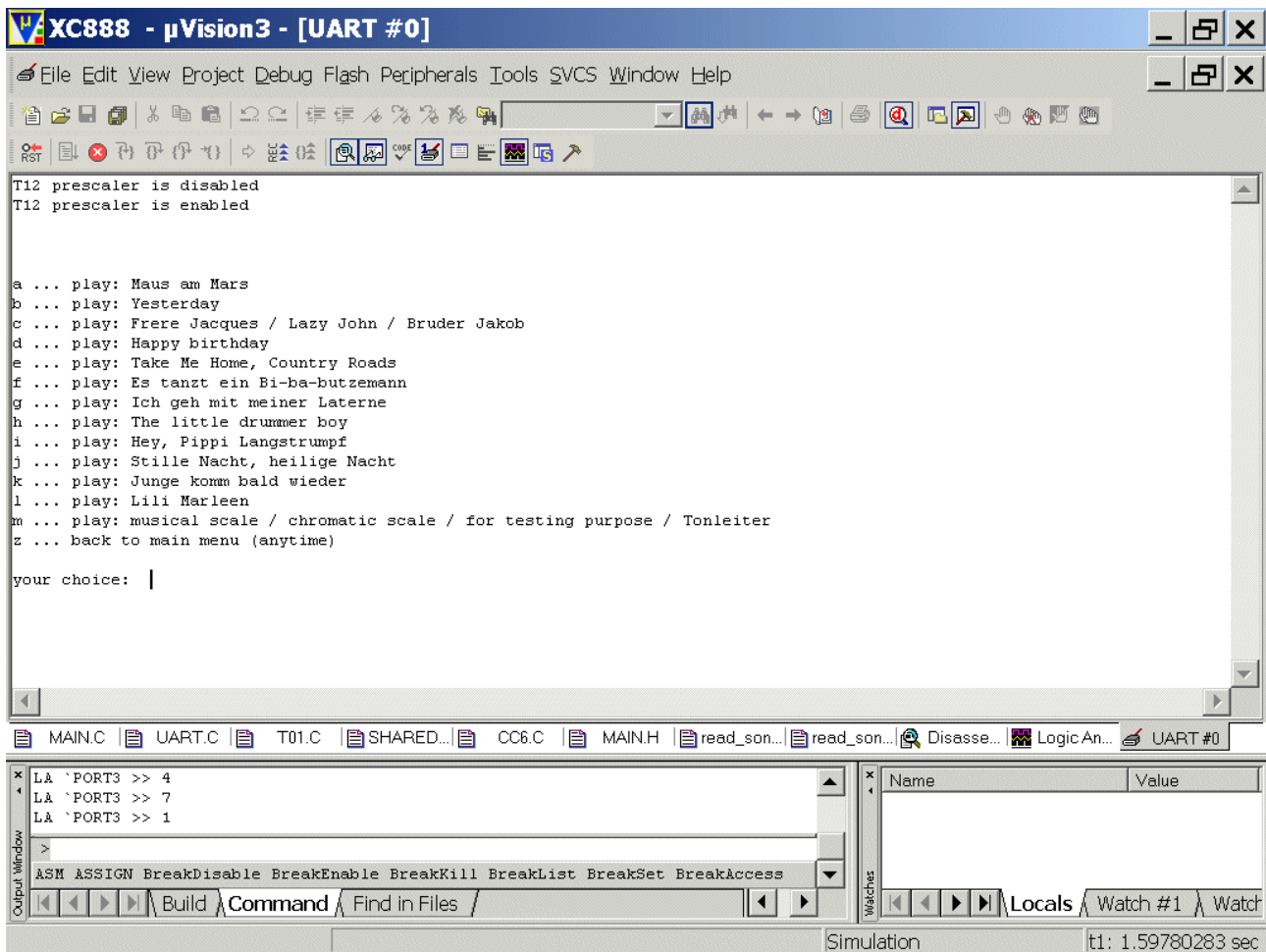


View – Serial Window – UART #0





Debug – Run



The screenshot shows the µVision3 IDE interface for the XC888 microcontroller. The main window displays the UART output for UART #0, which shows the program's execution. The program starts by disabling and then enabling the T12 prescaler. It then enters a menu loop where the user can select from various songs to play. The songs listed are: Maus am Mars, Yesterday, Frere Jacques / Lazy John / Bruder Jakob, Happy birthday, Take Me Home, Country Roads, Es tanzt ein Bi-ba-butzemann, Ich geh mit meiner Laterne, The little drummer boy, Hey, Pippi Langstrumpf, Stille Nacht, heilige Nacht, Junge komm bald wieder, Lili Marleen, and a musical scale / chromatic scale / for testing purpose / Tonleiter. The user's choice is currently empty.

```

T12 prescaler is disabled
T12 prescaler is enabled

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

your choice: |
  
```

The bottom of the IDE shows the Project Manager with files like MAIN.C, UART.C, T01.C, SHARED..., CC6.C, MAIN.H, read_son..., and Disasse... The Output Window shows the command prompt with the following commands:

```

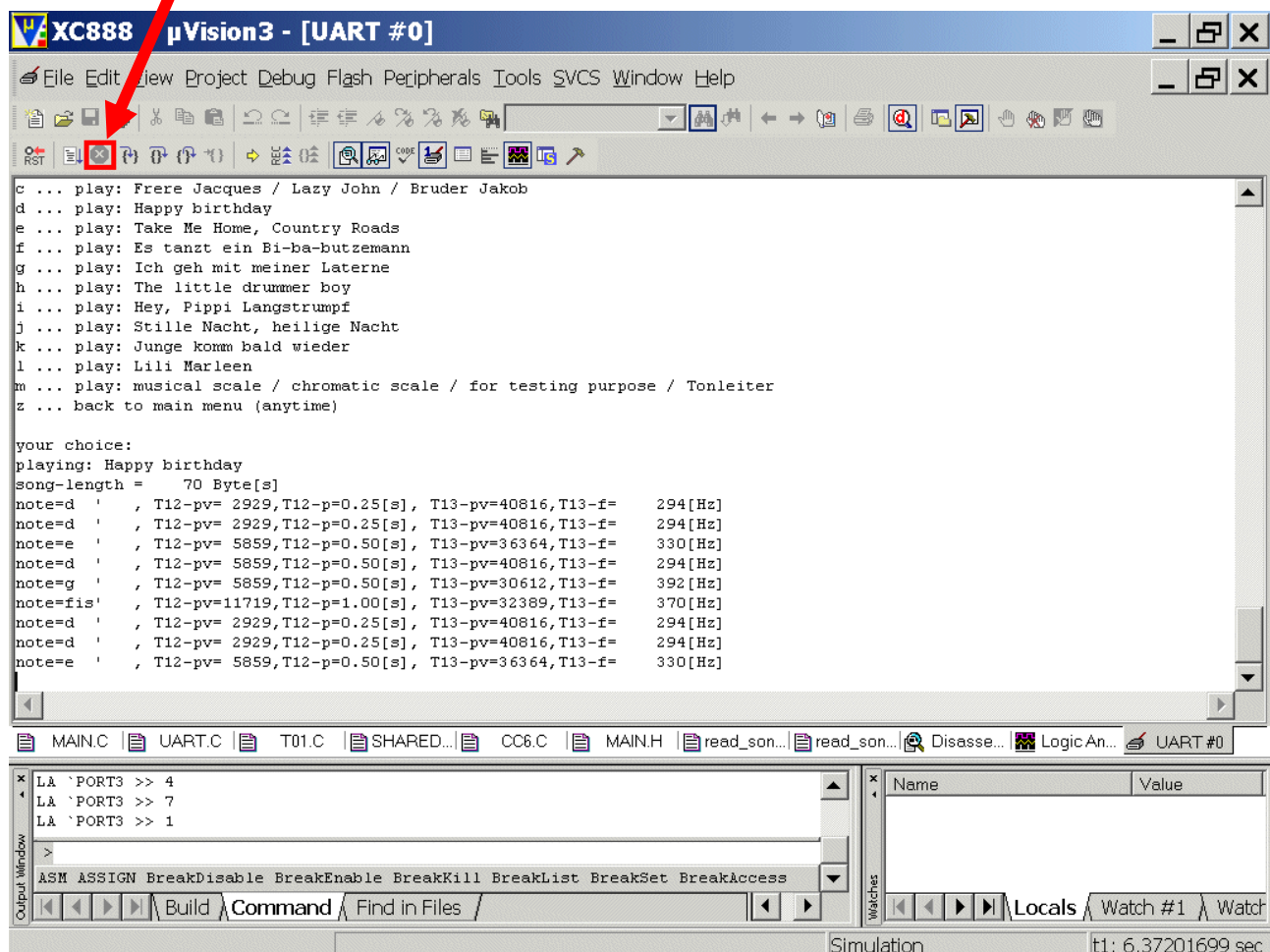
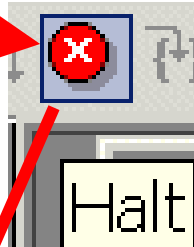
LA `PORT3 >> 4
LA `PORT3 >> 7
LA `PORT3 >> 1
>
  
```

The Watch Window is empty. The Status Bar at the bottom indicates the simulation is running at t1: 1.59780283 sec.

UART #0 insert d

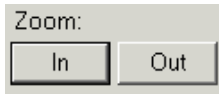
(... after some notes are played)

click



And see the result via the Keil Software Simulator LGA:

use:

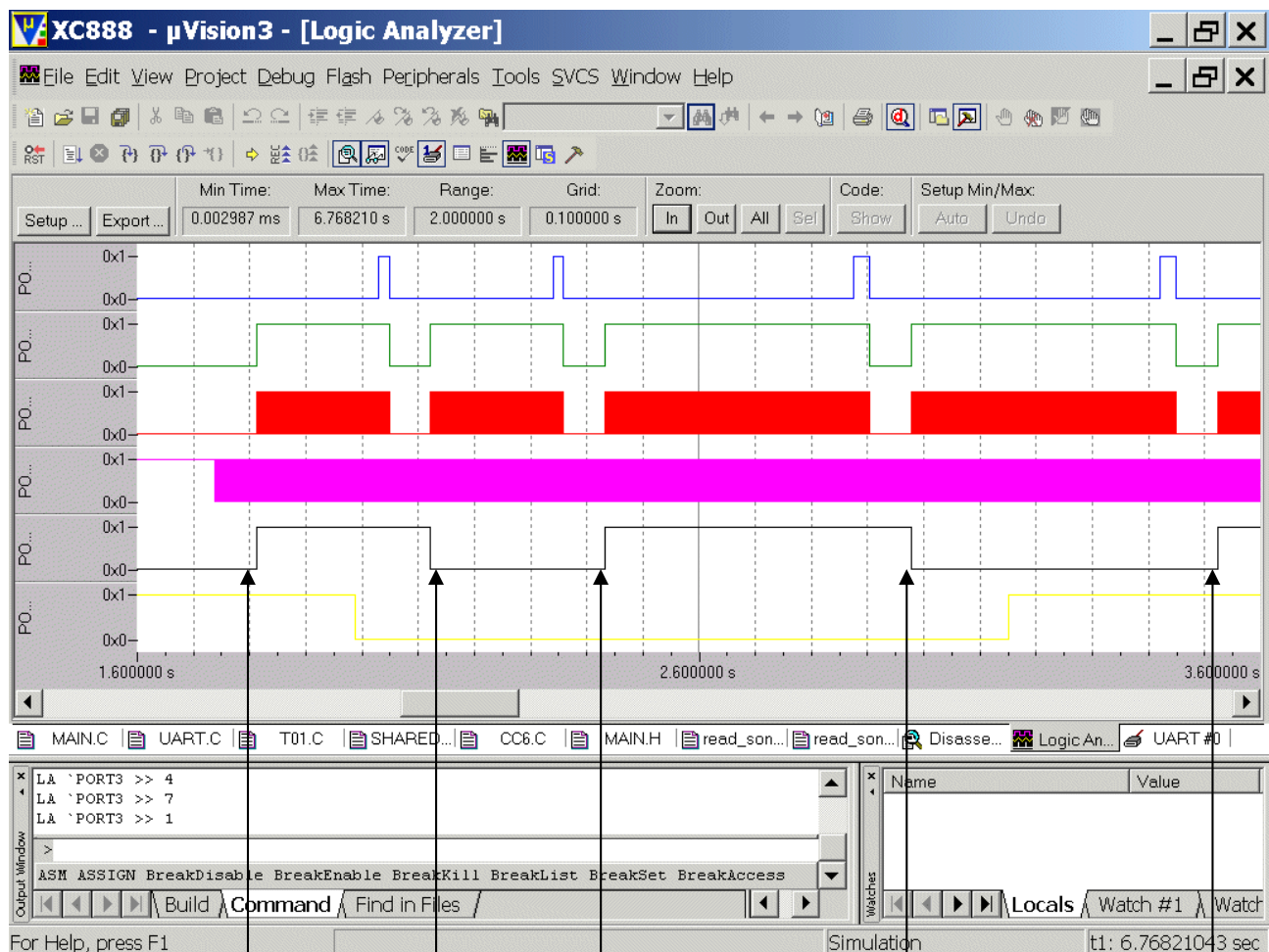


Note:

Song d: Happy birthday:

code unsigned char

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



Start/play
next note

Start/play
next note

Start/play
next note

Start/play
next note

Start/play
next note



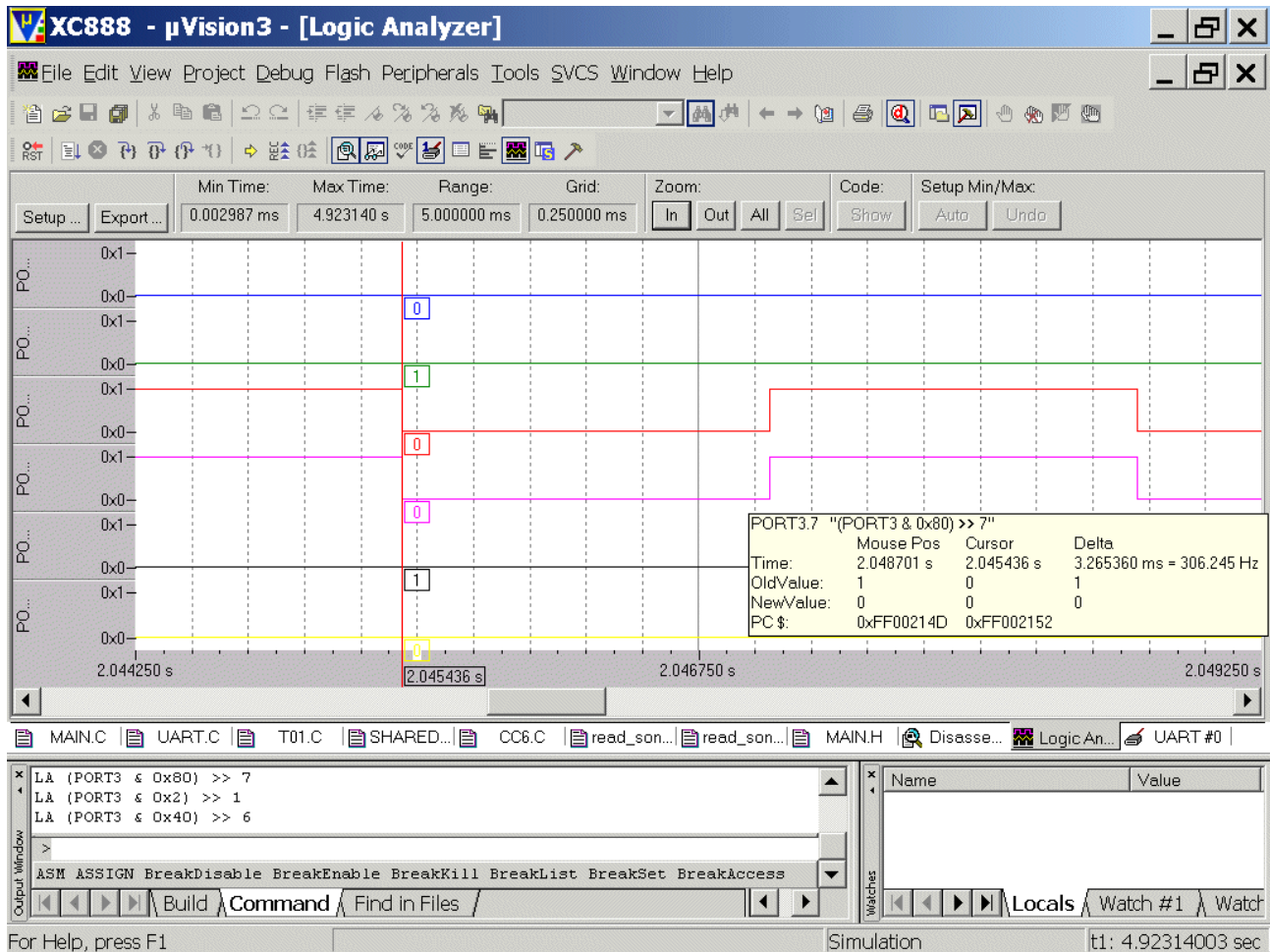
Note:

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

With the Keil Software Simulator LGA we could measure the frequency of note d (we expect 294 Hz):

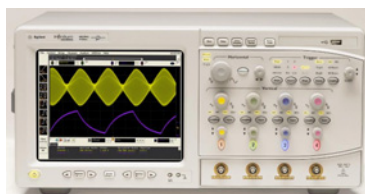


We expect: 294 Hz, we measure 306 Hz with the Keil Software Simulator LGA and 294 Hz with a real LGA (see page 252).

See and hear the result:

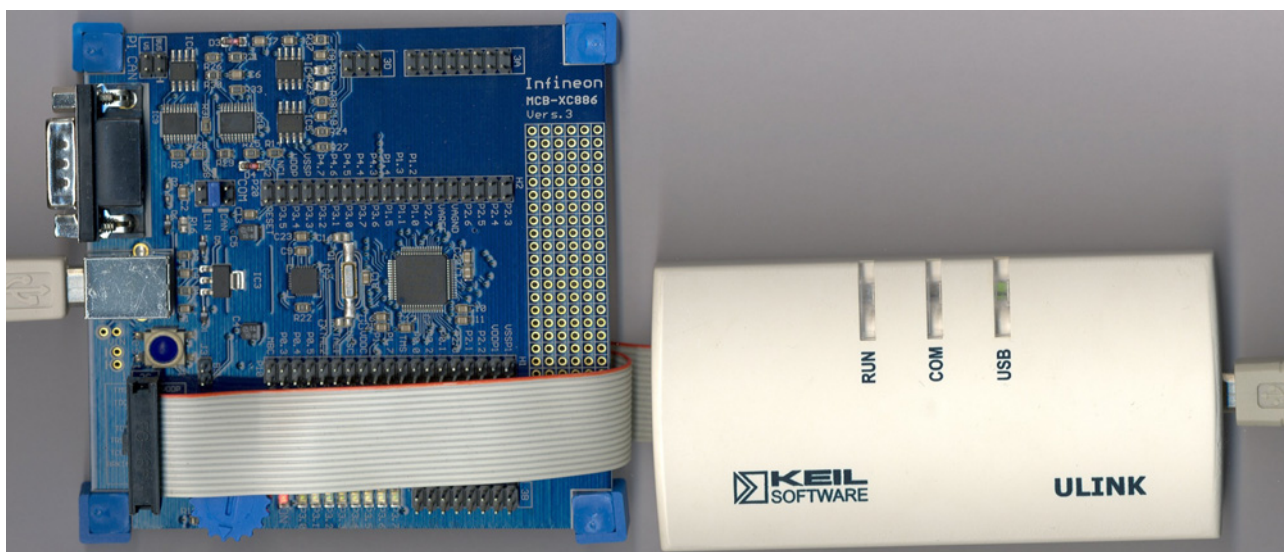
Using real hardware:

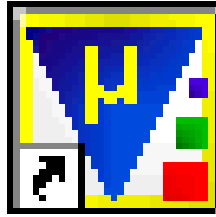
Any Terminal Program (e.g. Docklight) + Any Logic Analyser / scope + loudspeakers



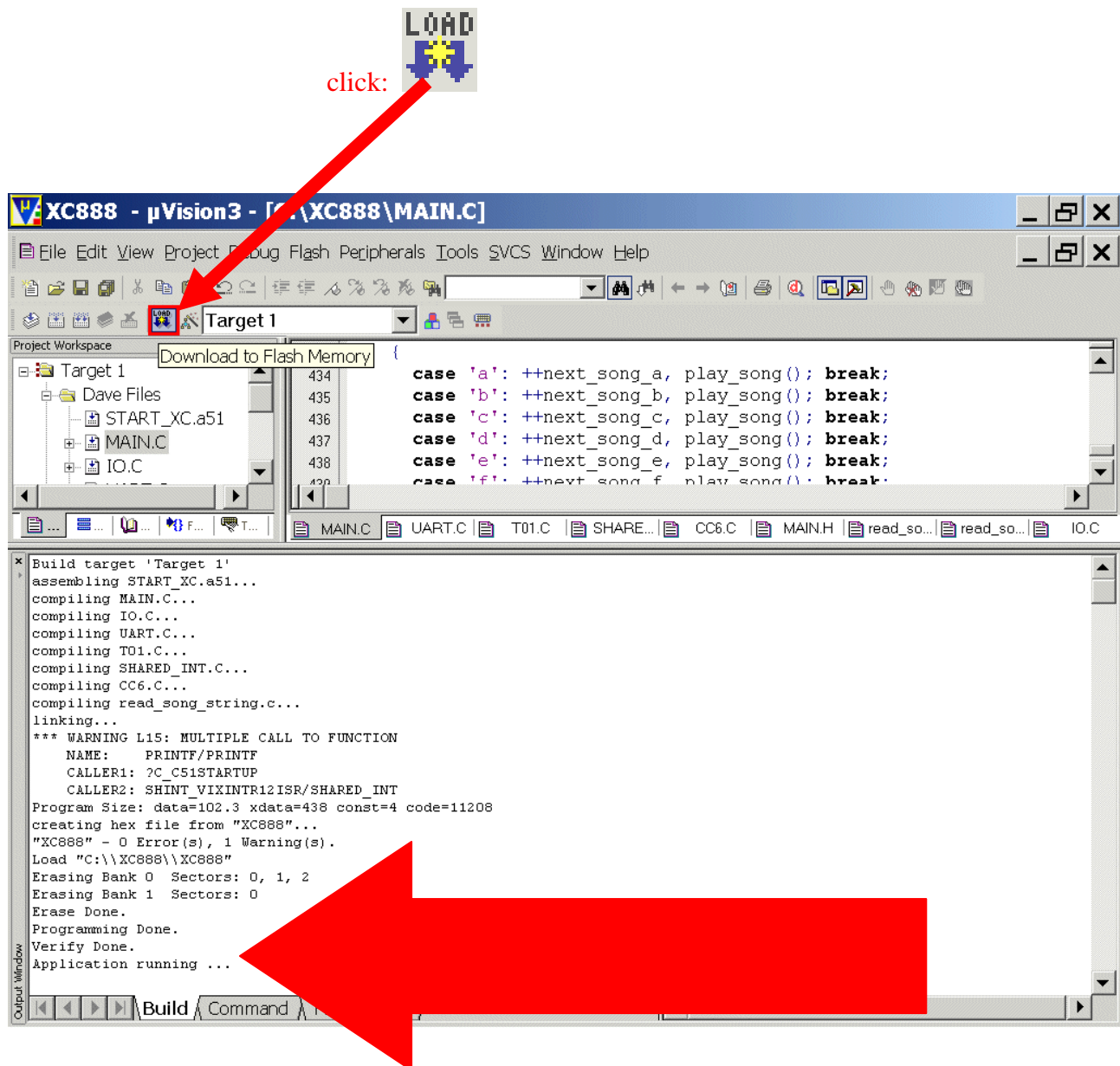
Connect your ULINK (used for: program download),

Connect your serial interface UART/RS232 via USB (used for: serial communication):



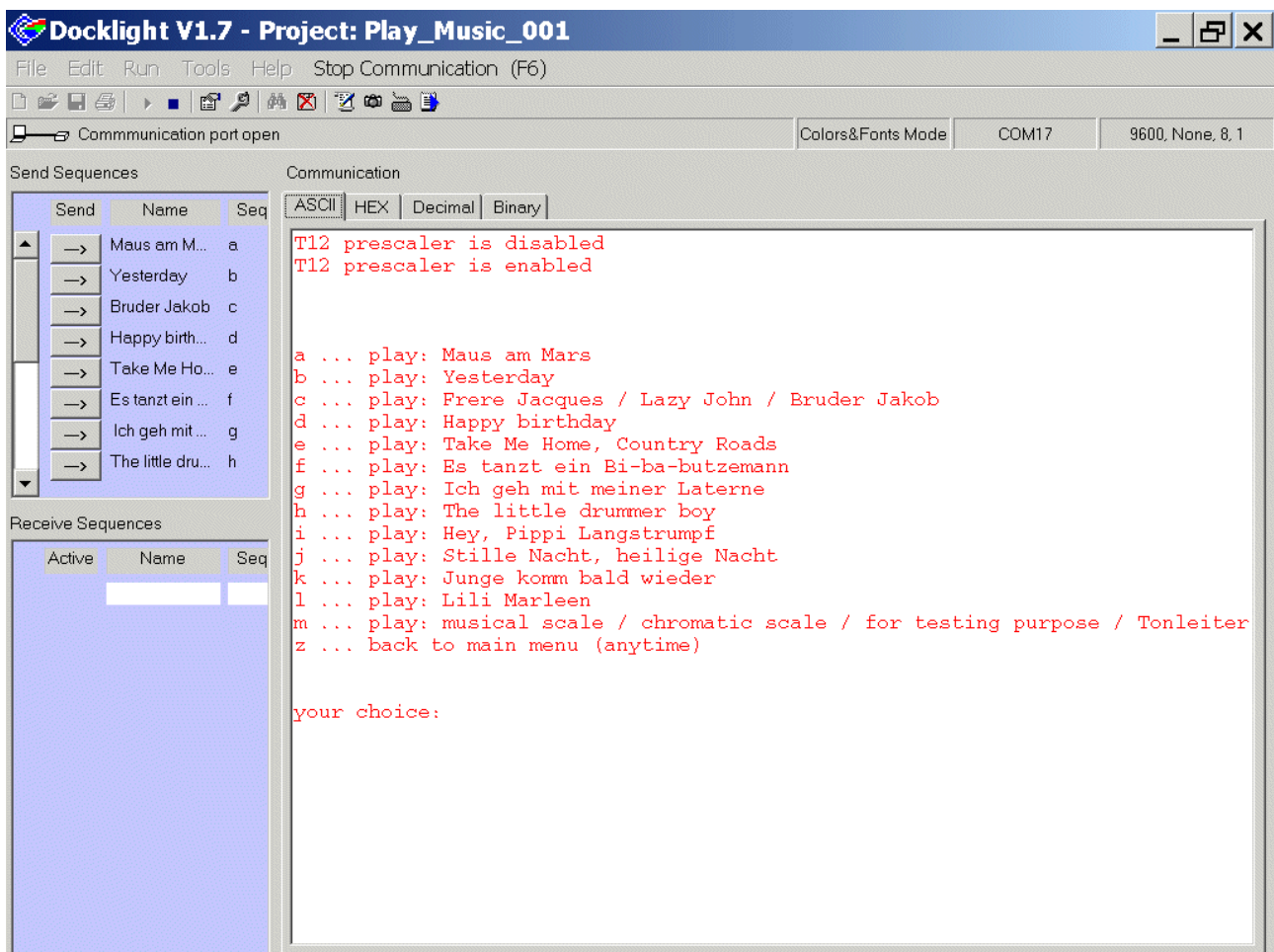


Go to [µVision](#), [Download/Program](#) our application program:

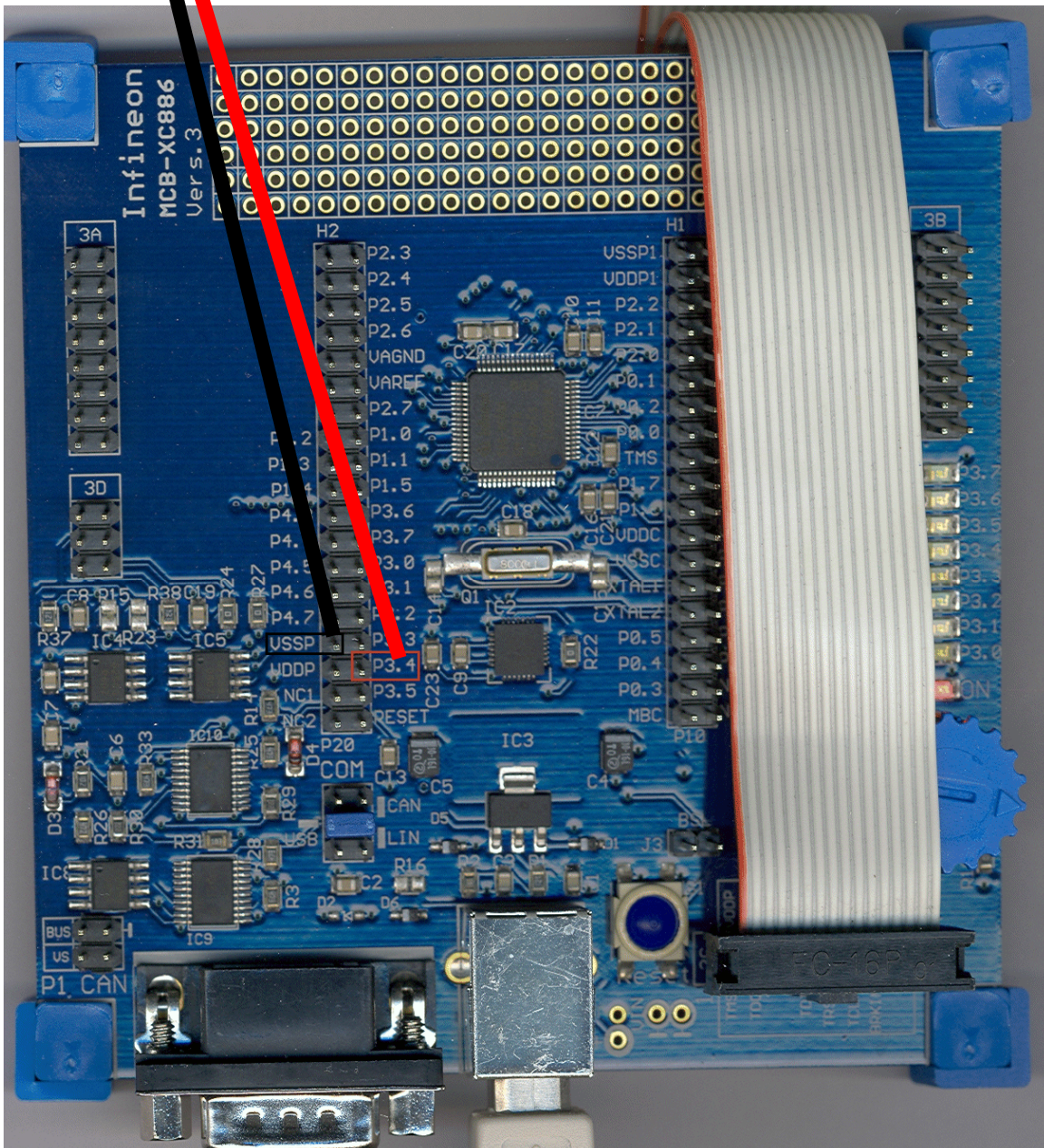




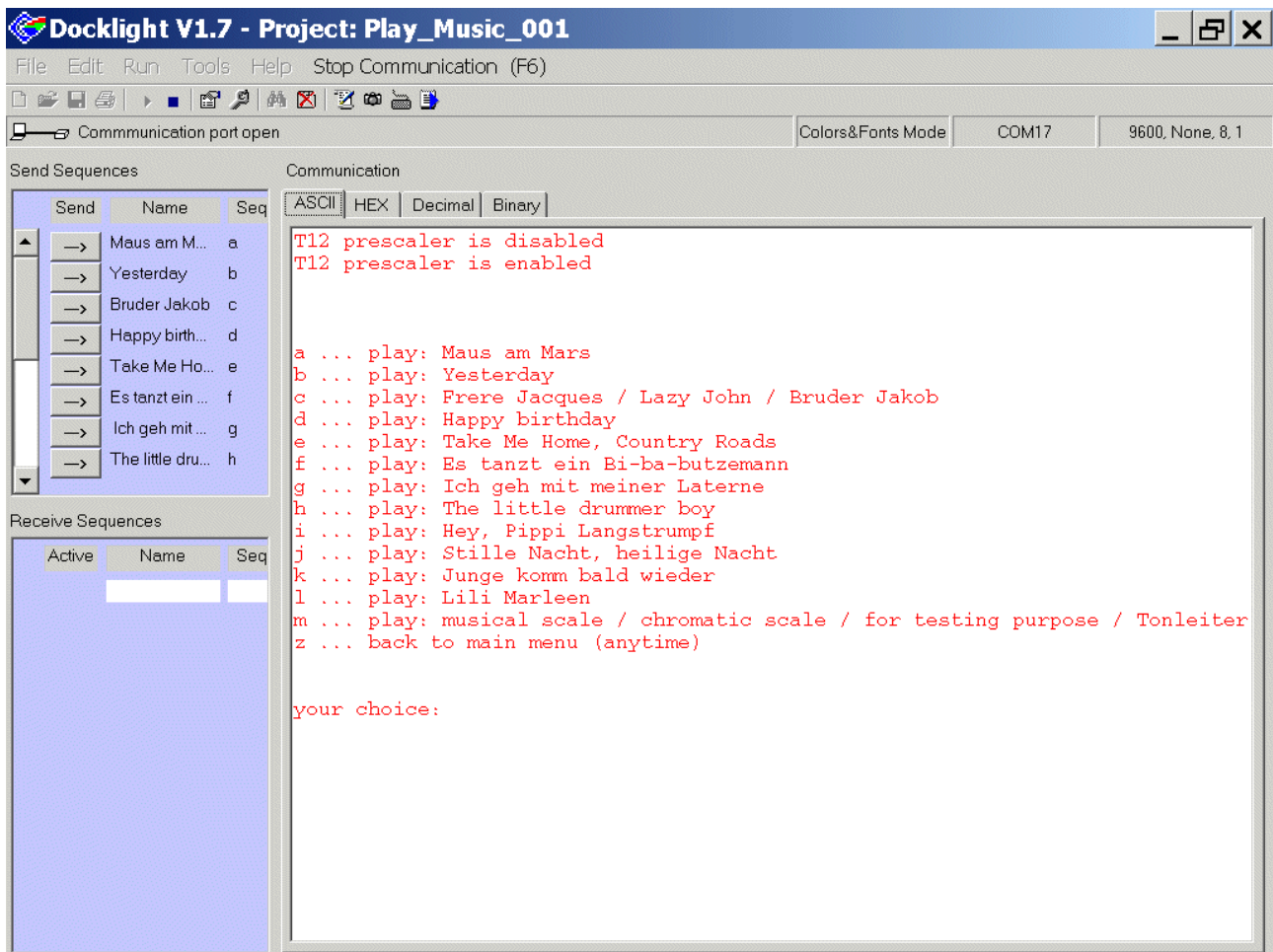
Go to Docklight and see the result:



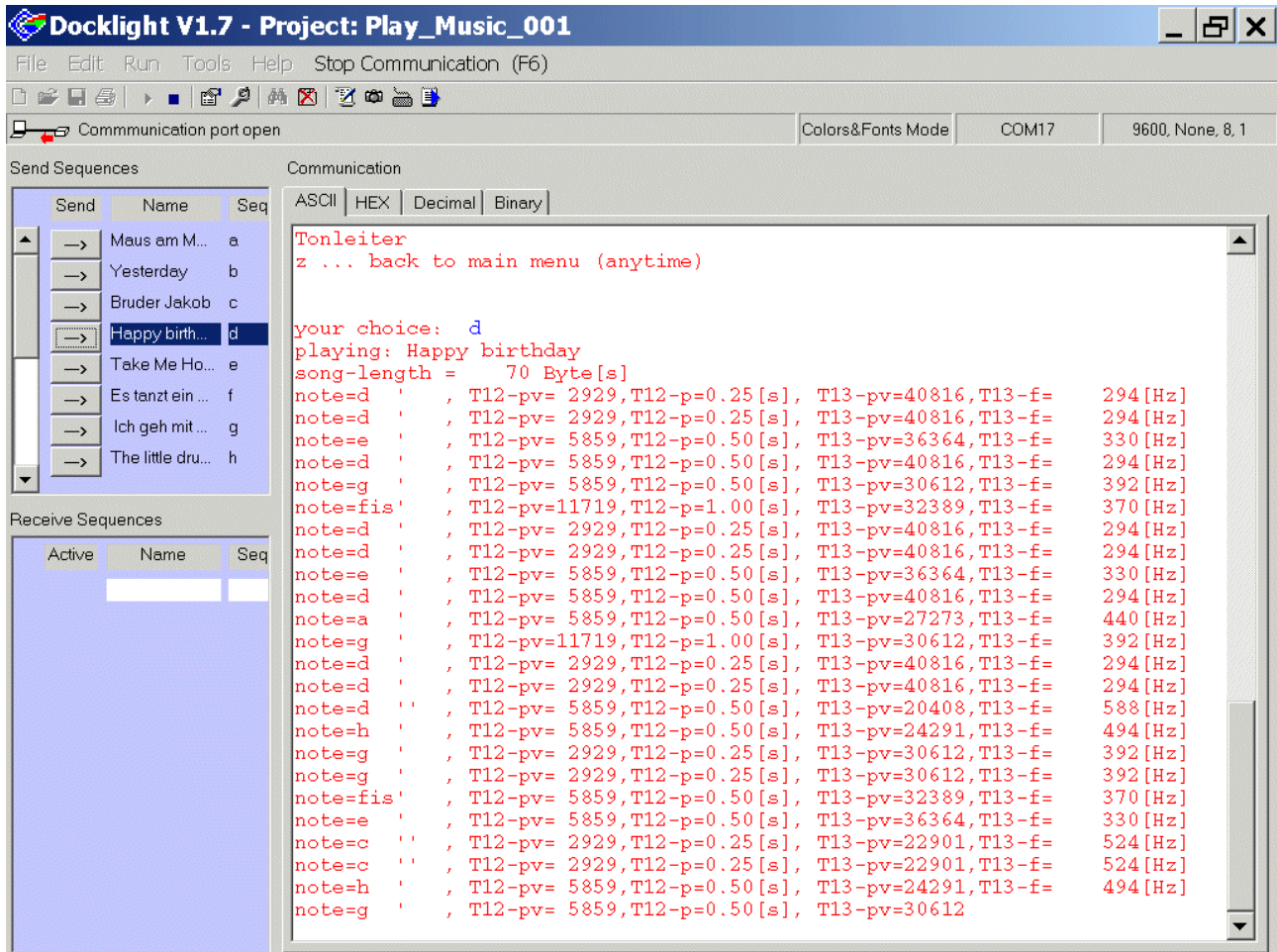
Connect CC62 (P3.4) and GND (VSSP) to your active loudspeaker(s) - and start/select a song:



See / hear / enjoy the results:



Insert/select d



Use any logic analyser and see the result:

Note:

Song d: Happy birthday:

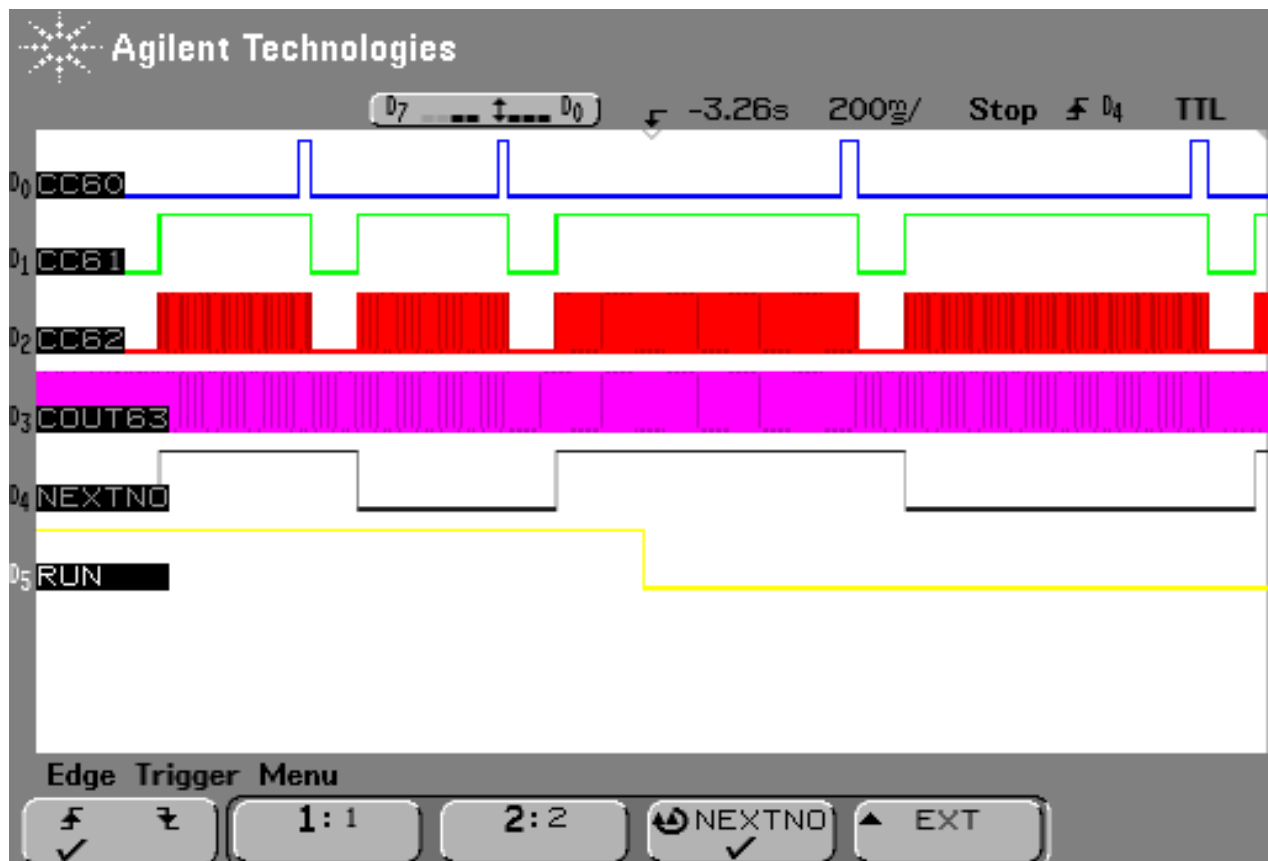
code unsigned char

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



HAPPY BIRTHDAY =

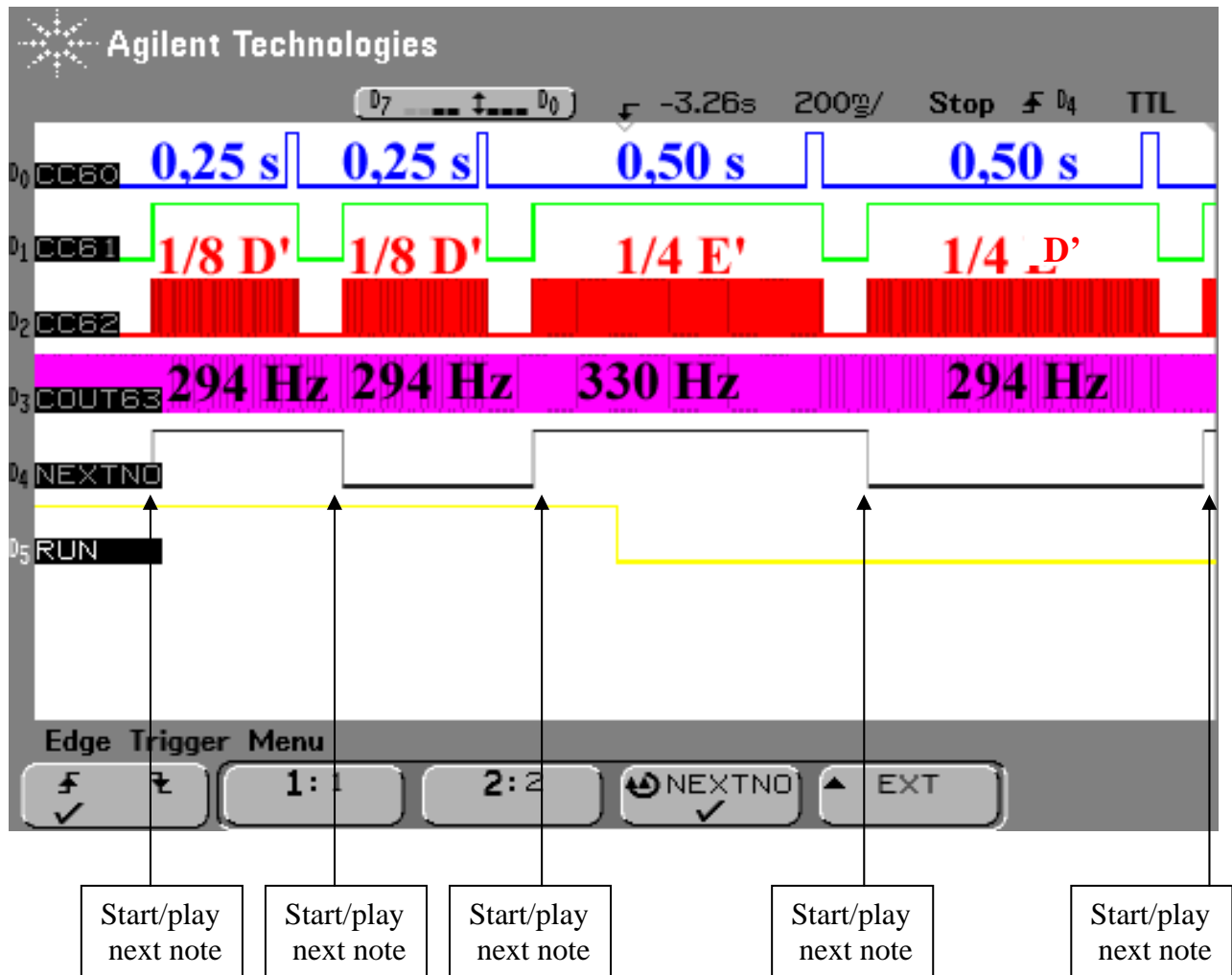
T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F... ..





HAPPY BIRTHDAY =

T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F... ..



Note:

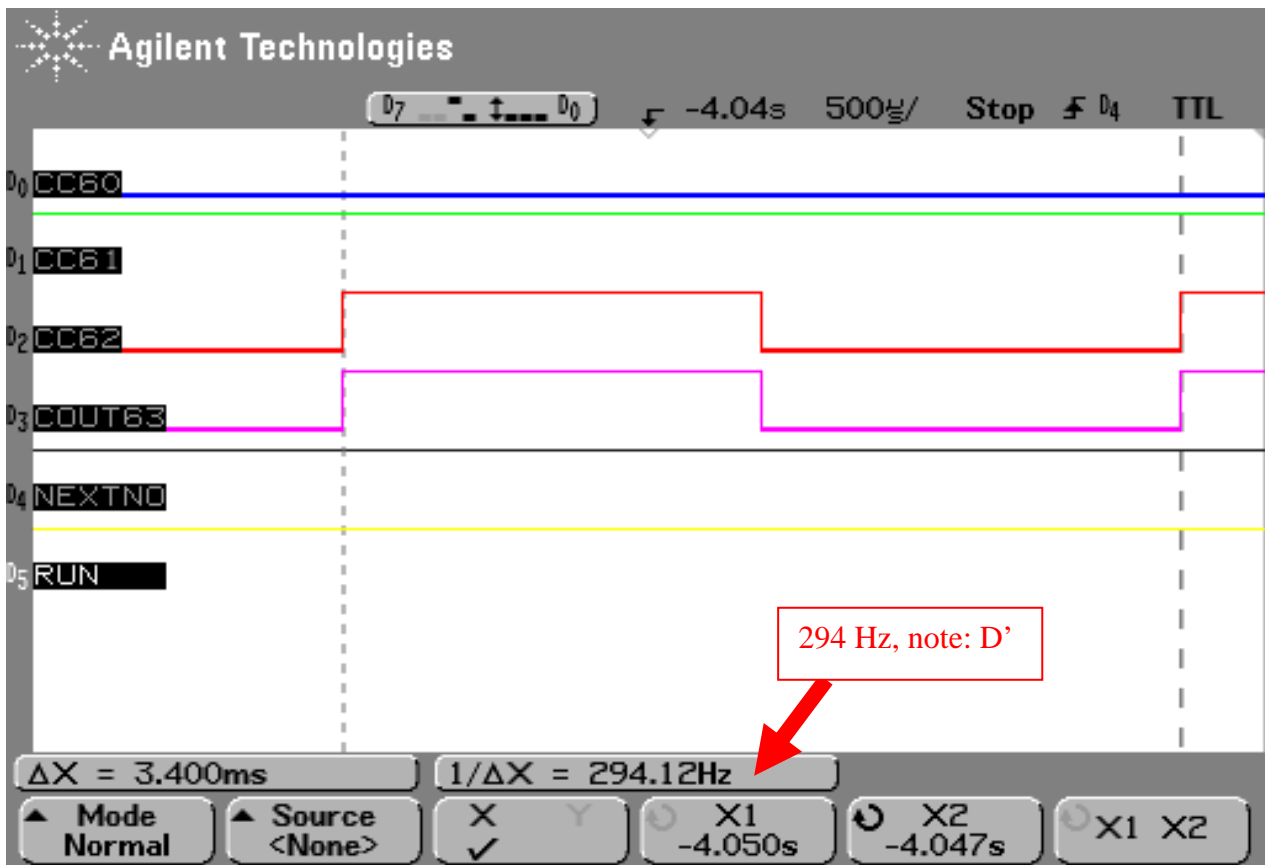
Start/play next note:

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

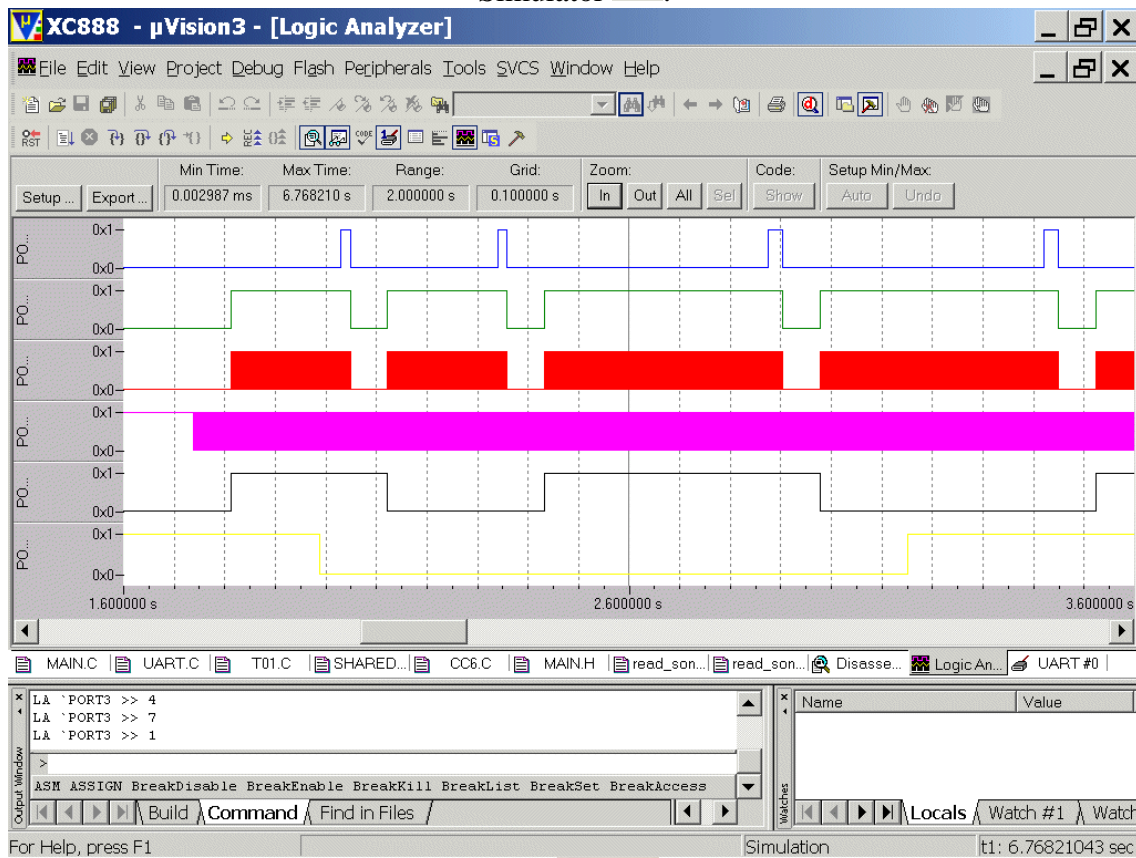


Now we can measure the frequency of note d (we expect 294 Hz) with the LGA:

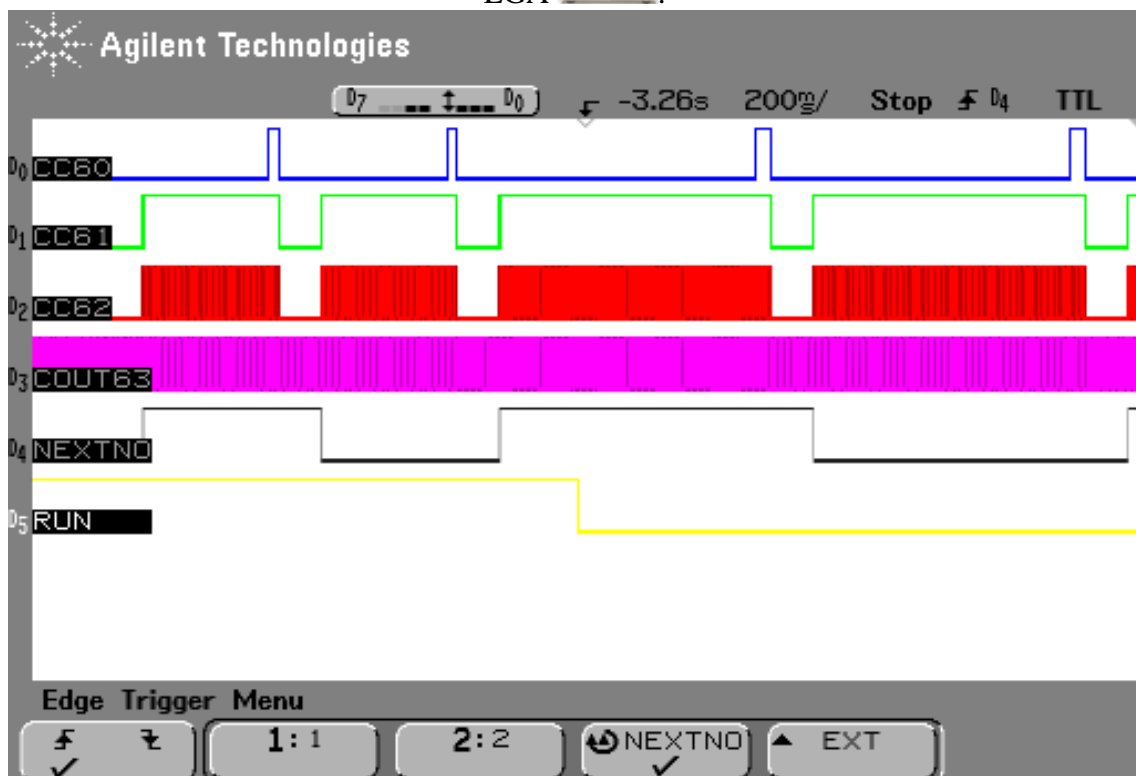


Compare the results:

Simulator :



LGA :



4.) PERFECTIONISTS ONLY!

Perhaps you noticed:

Playing a note like the ones in the **red boxes** below is not possible, because there is no support for this octave.

Hey, Pippi Langstrumpf



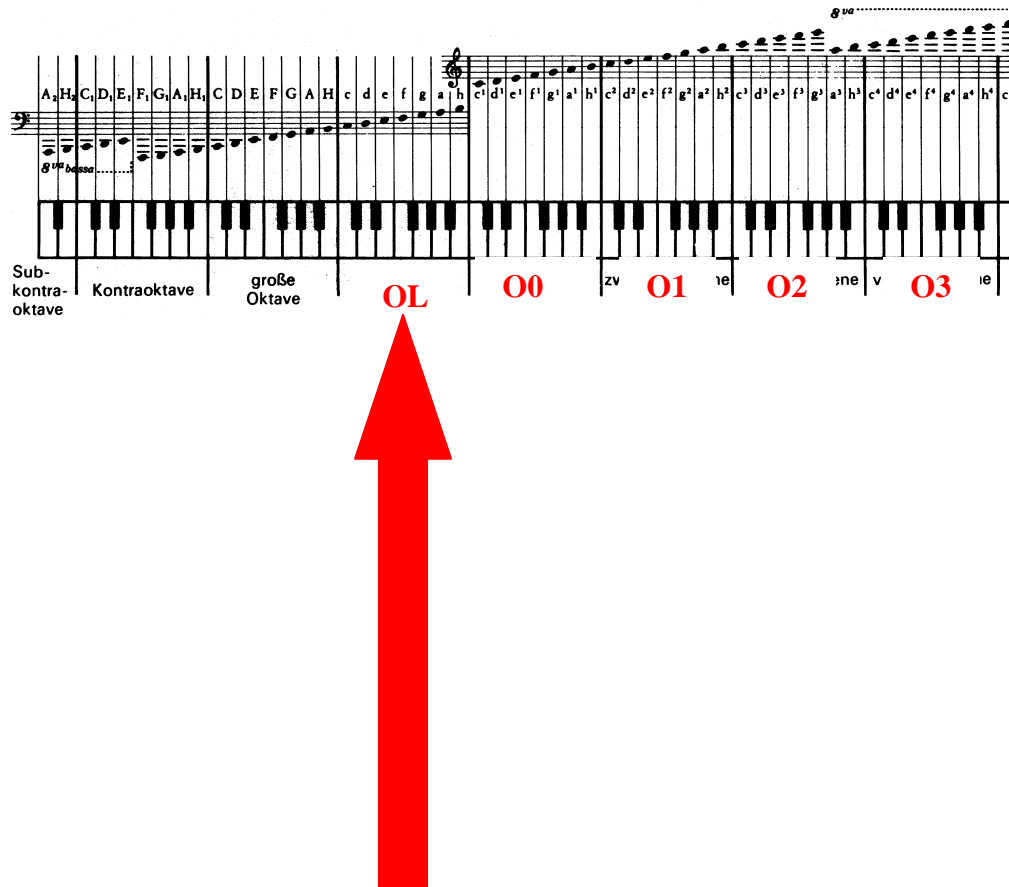
1. Zwei mal drei macht vier, wi-de wi-de witt und
Drei mal drei macht sechs, wi-de wi-de wer will's

Junge komm' bald wieder



d wie-der, bald wie-der nach Haus. Jun -

Therefore, we are now introducing (in this chapter: **PERFECTIONISTS ONLY!**)
a “lower octave” called “octave low” or “**OL**”!





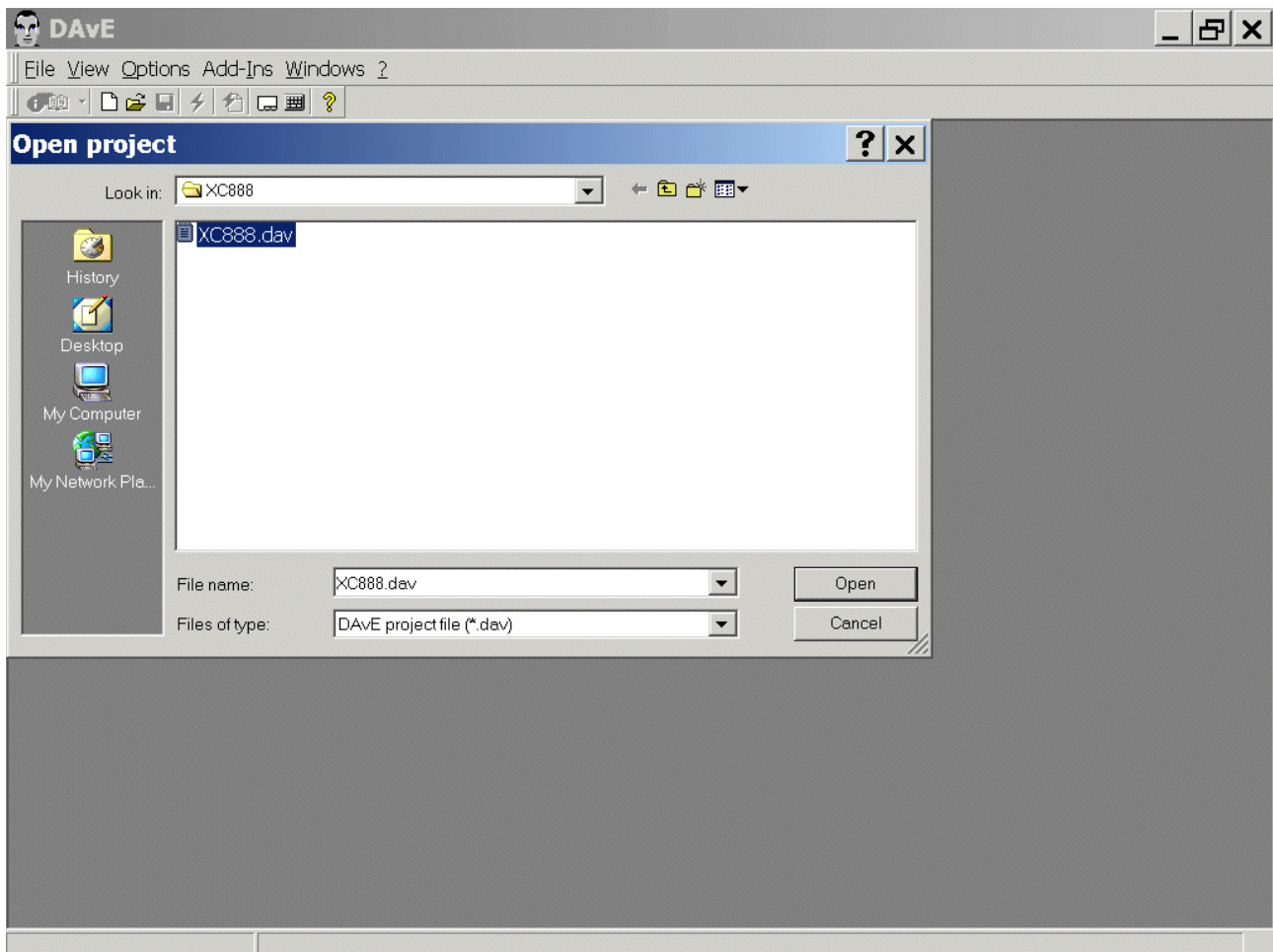
Start the program generator DAVe and open your [XC888.dav](#) DAVe project:

File

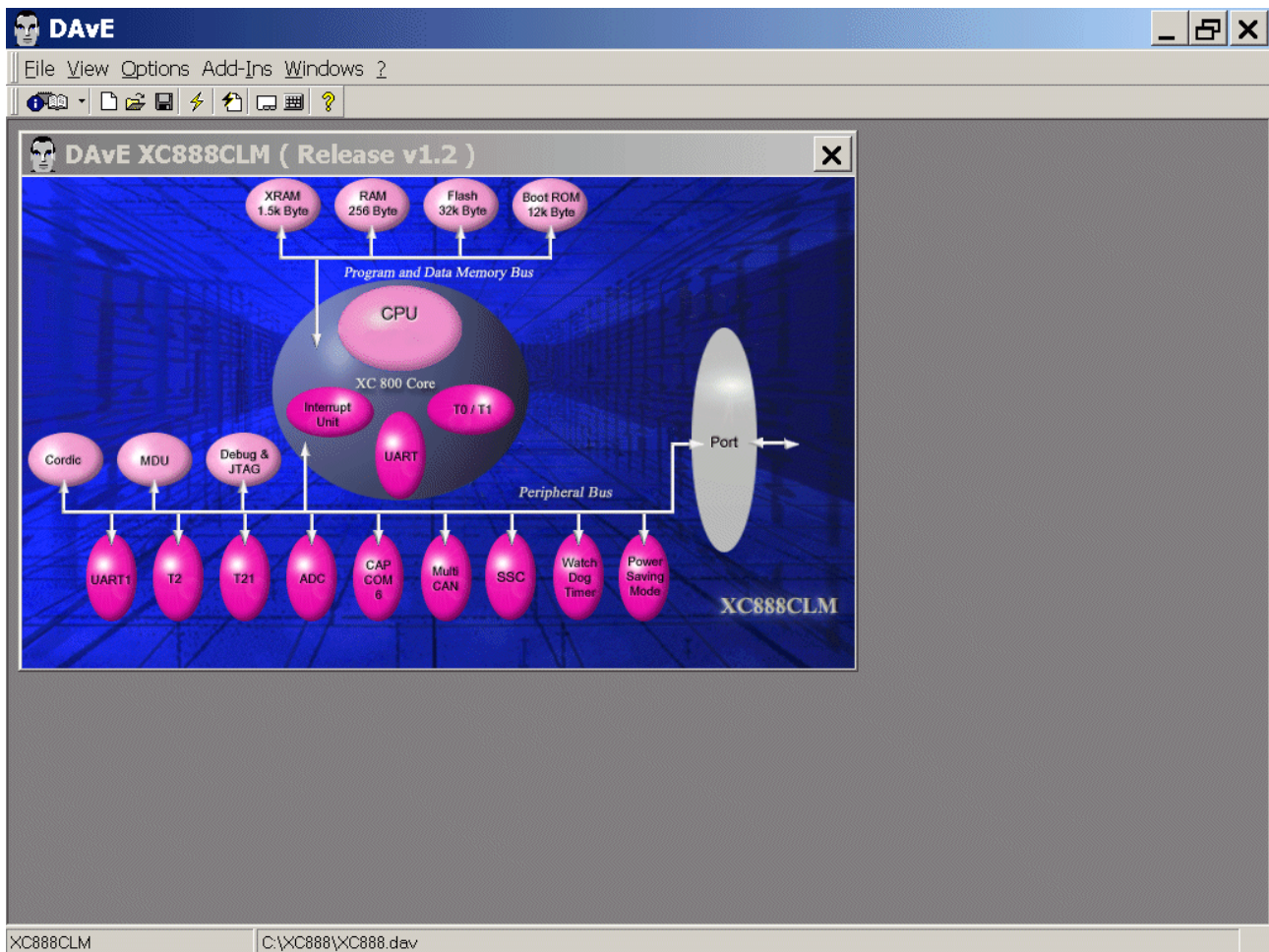
Open

Location: [C:\XC888](#)

Filename: [XC888.dav](#)

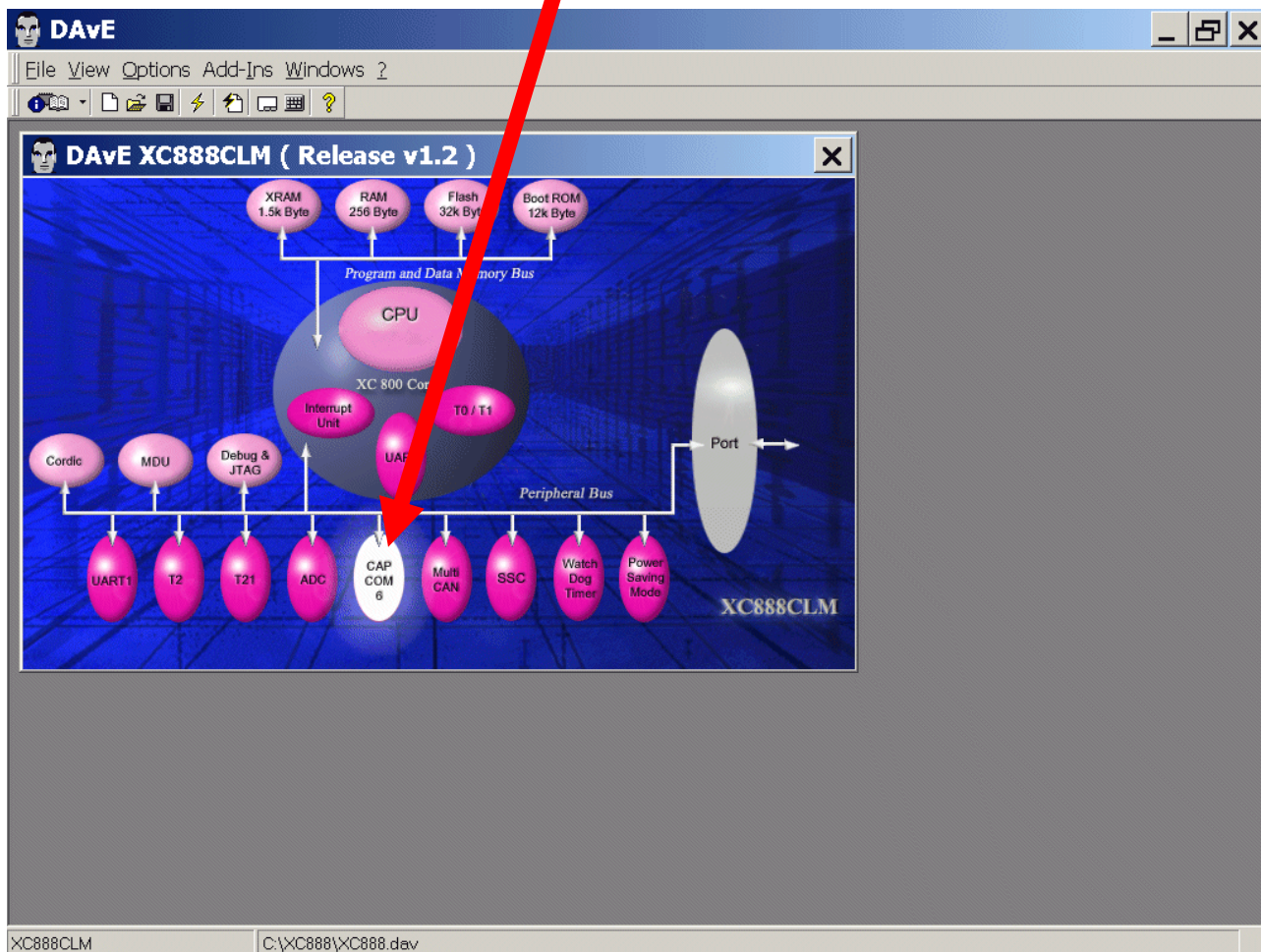


Click Open

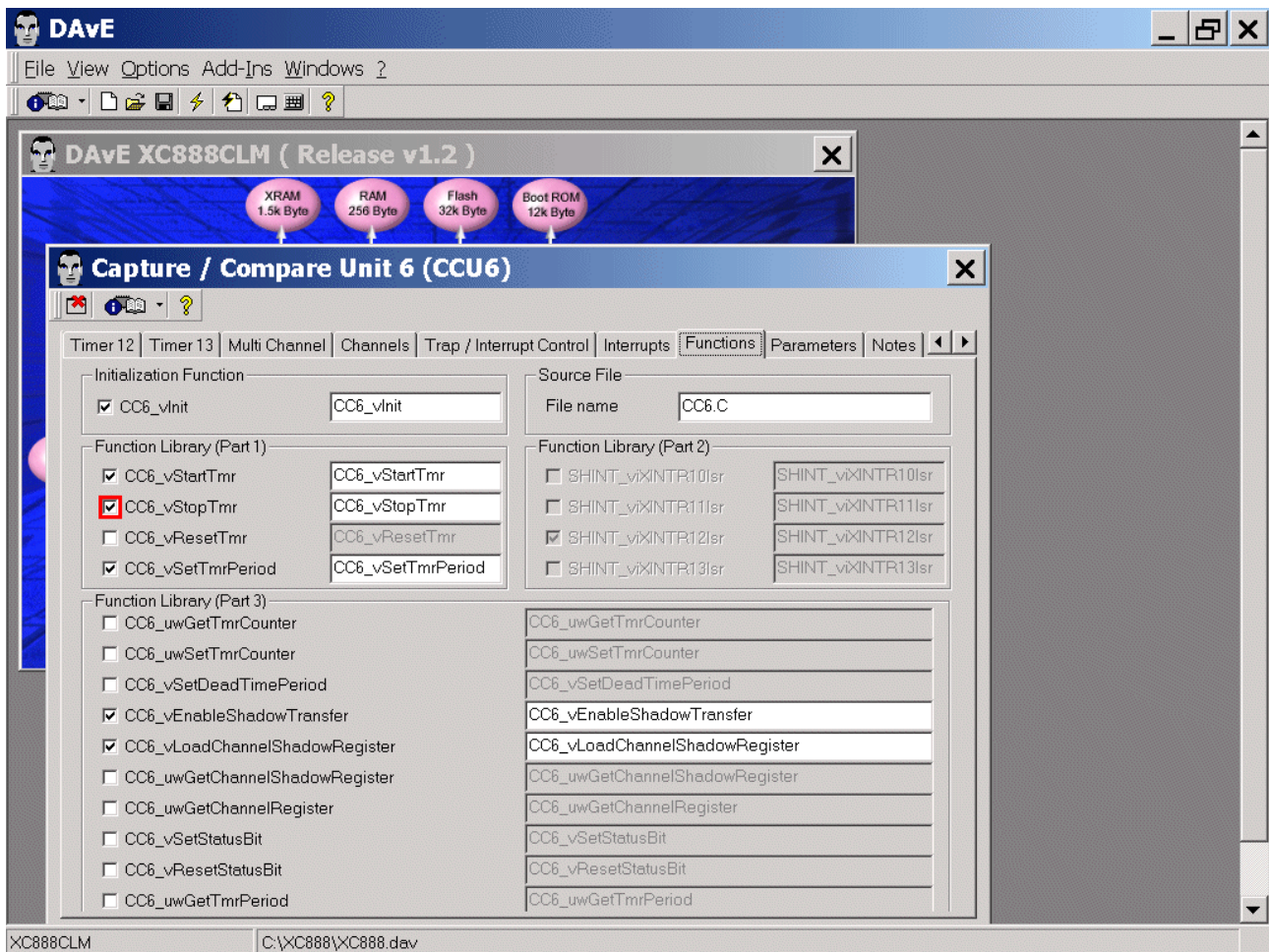


Reconfiguration of the CAPCOM 6 module:


The configuration dialog can be opened by clicking the specific block/module.



CCU6: Functions: Function Library (Part 1): click ☒ CCU6_vStopTmr

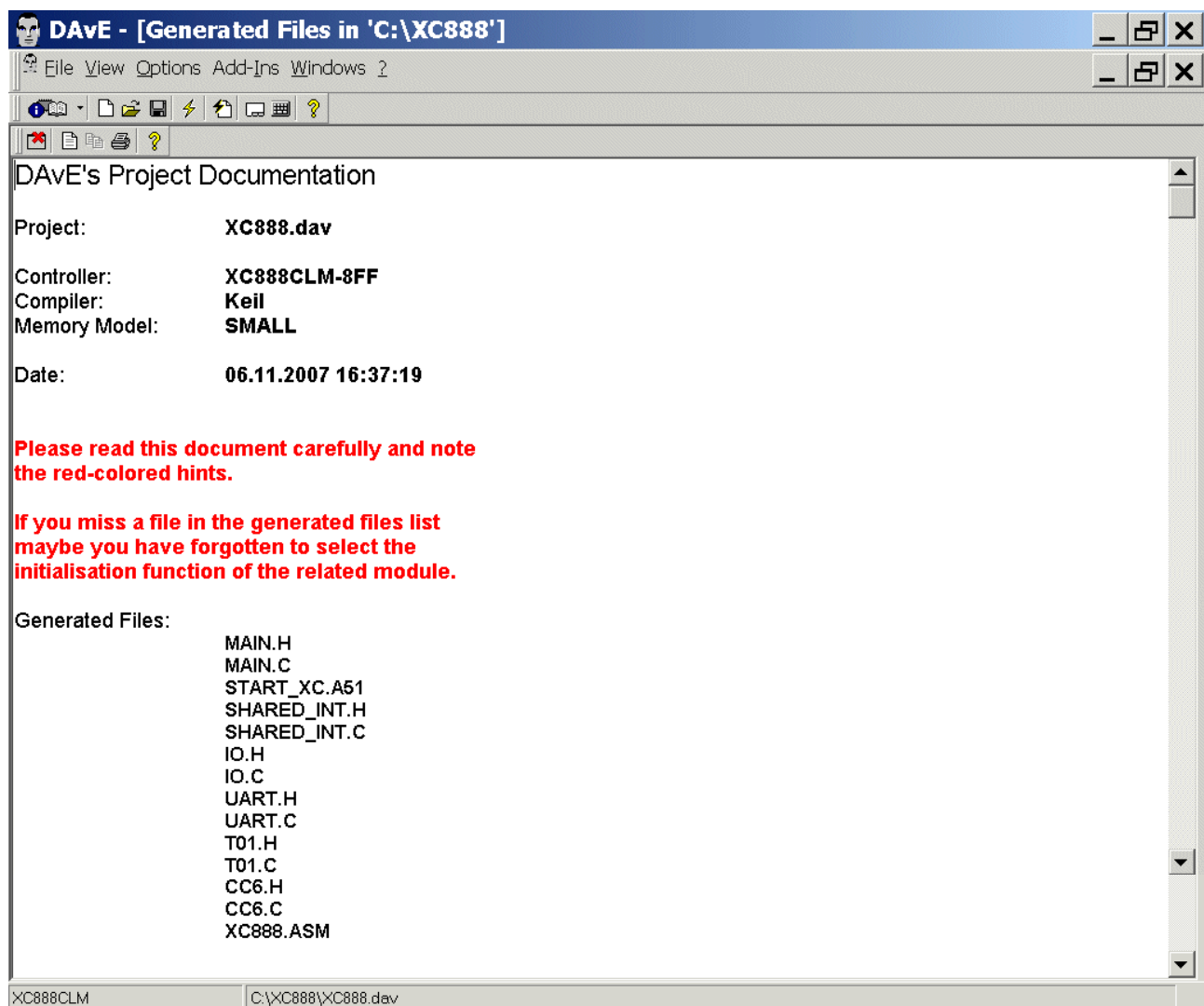


Generate Code:

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



DAvE will show you all the files he has generated
(Project Documentation opens automatically).



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



Start Keil μ Vision and open your Keil Project:

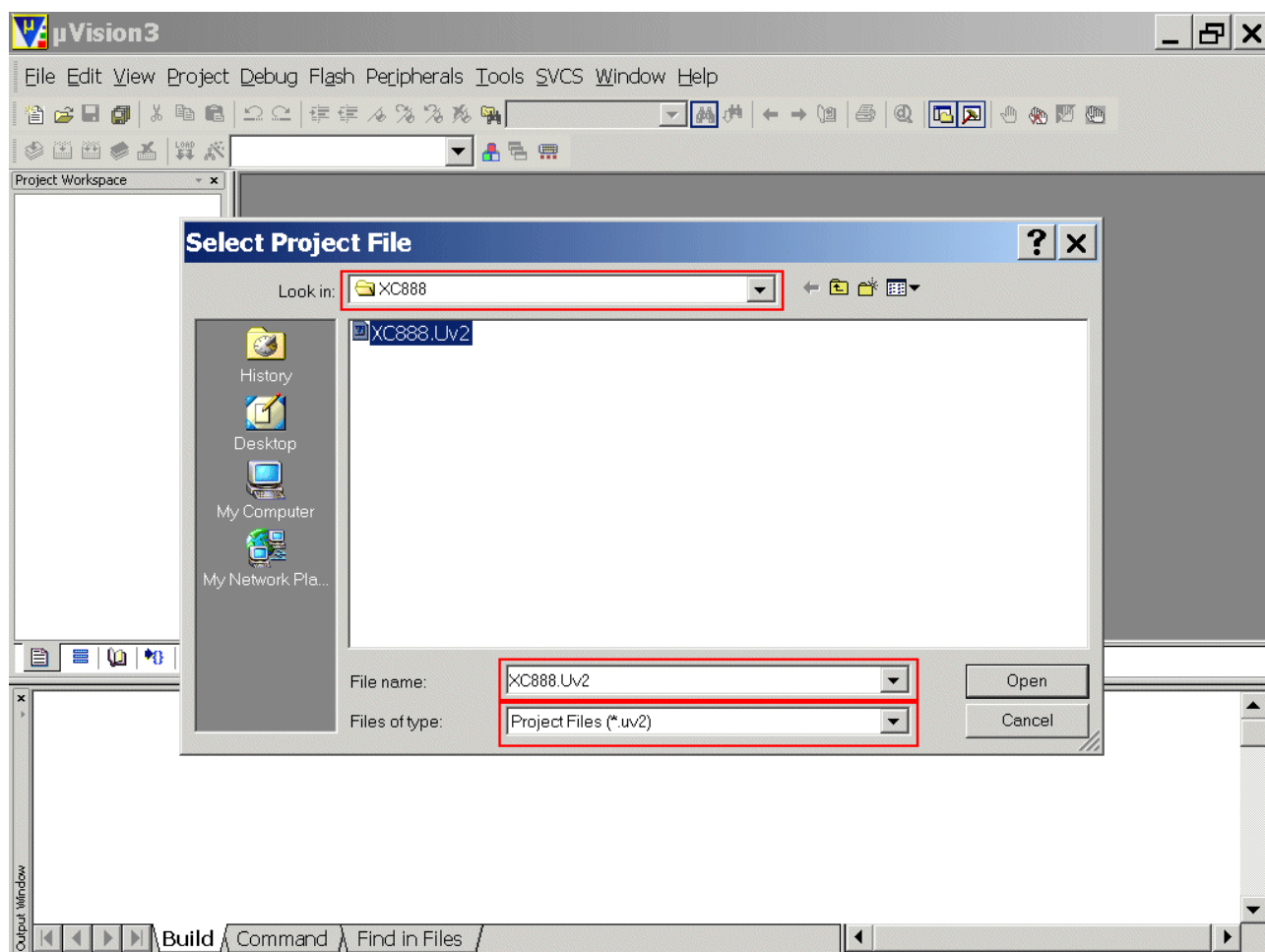
If you see an open project – close it: **Project - Close Project**

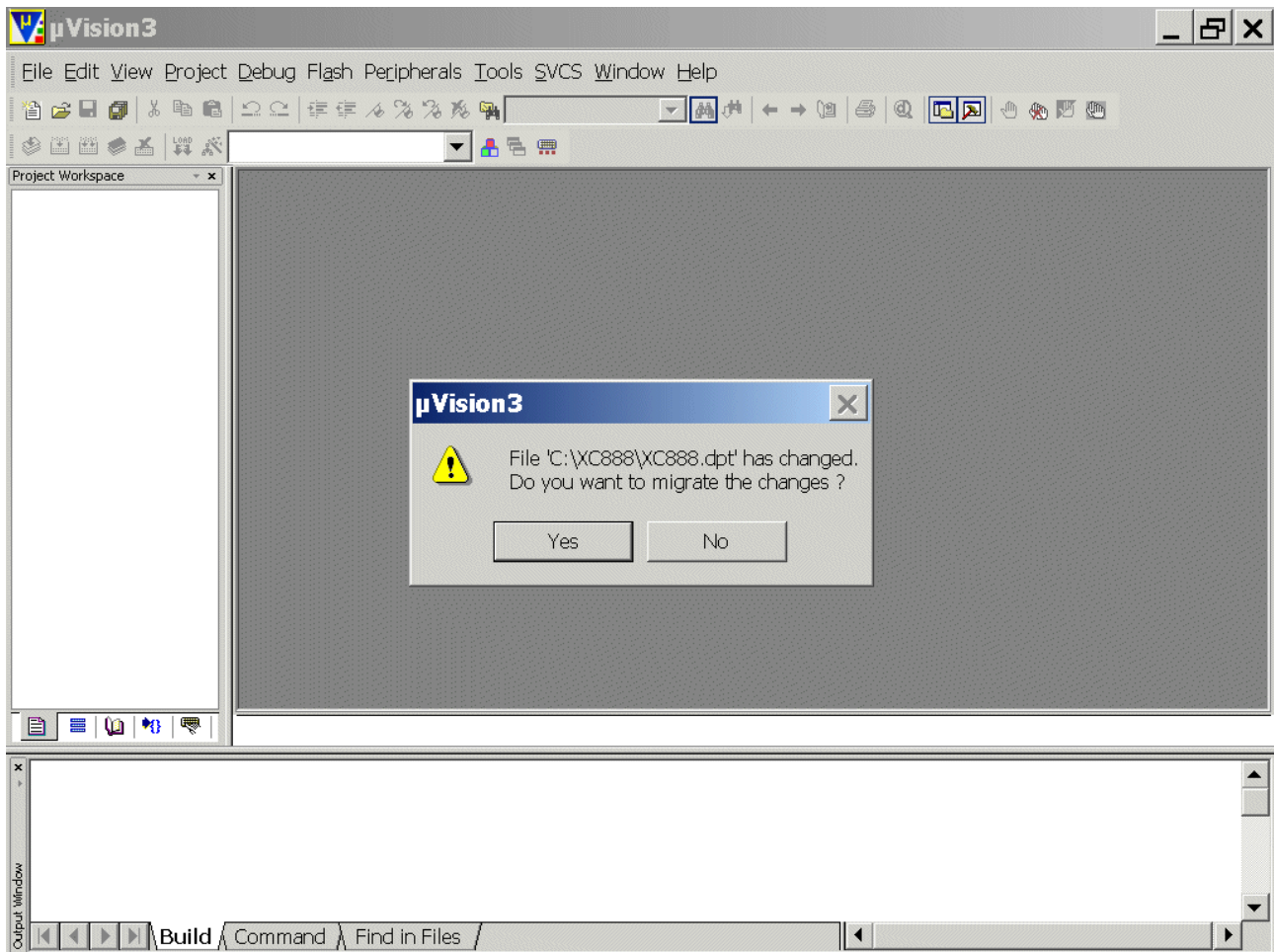
Project - Open Project

Select Project File: **Look in:** choose C:\XC888

Select Project File: **Files of type:** choose Project Files (*.uv2)
choose XC888.Uv2

Open





Click Yes

Double click **MAIN.C** and **change** code / comment / the syntax description for the music string

from:

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

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)

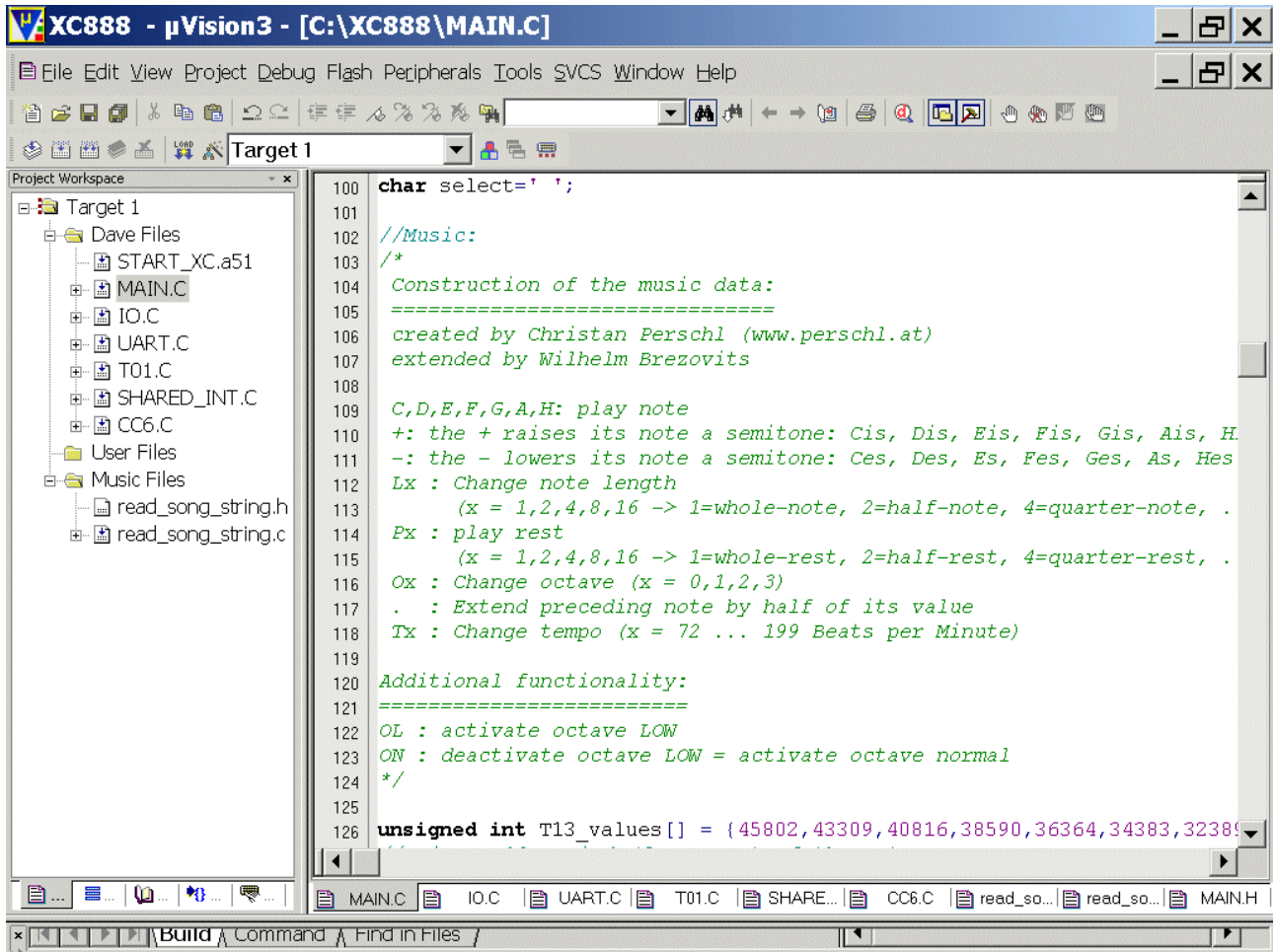
Note (restrictions/limits):
The chosen octaves supported by the algorithm used
[read_song_string()] fit a descant recorder perfectly.
Therefore, to keep the programming example simple, notes below
octave 0 (00) are not supported!
Also, be aware that not every song sounds good on a descant
recorder.
*/
```


to:

```
//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
*/
```



Double click `read_song_string.c` and insert code:

```
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

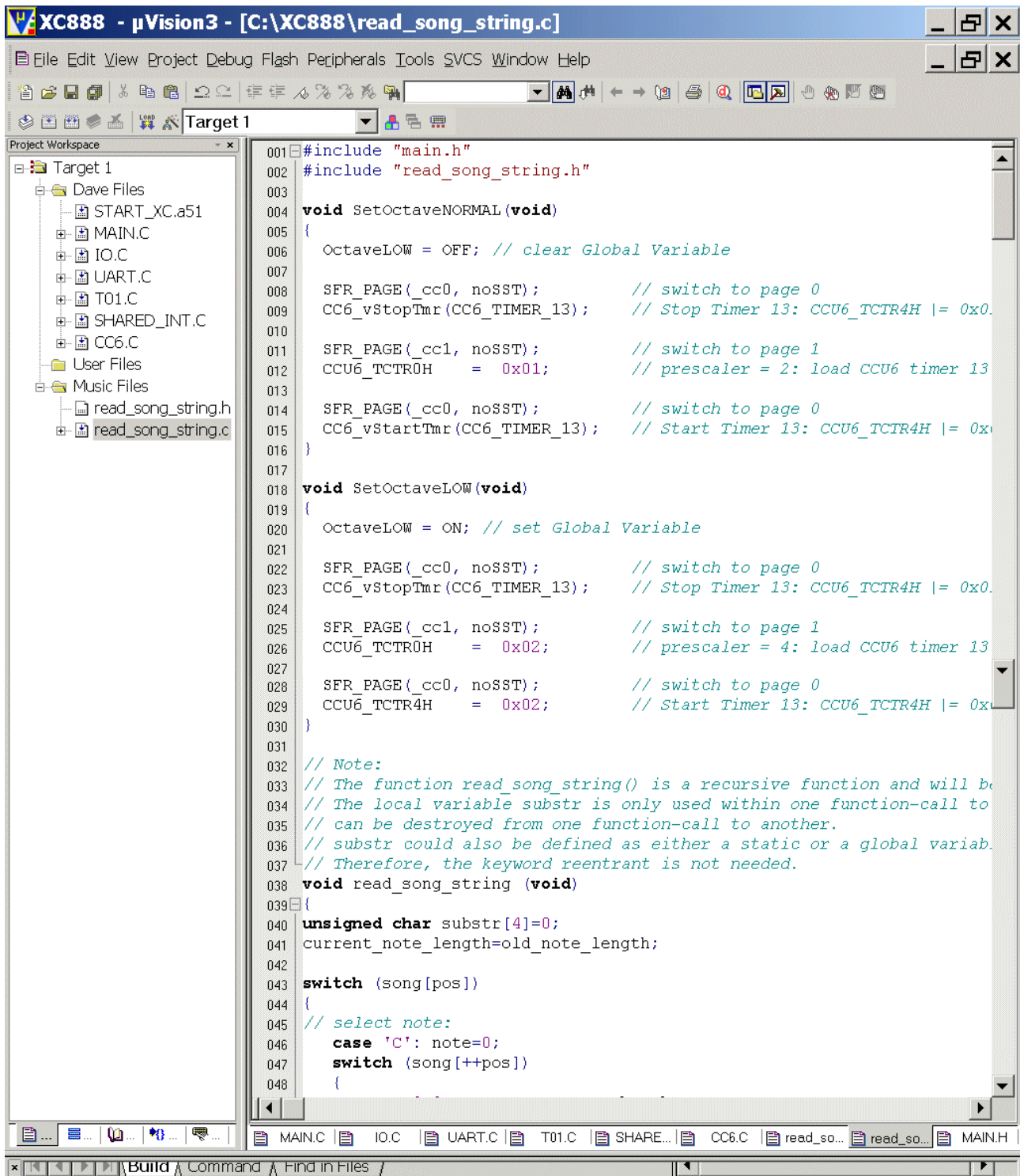
    SFR_PAGE(_cc0, noSST); // switch to page 0
    CC6_vStartTmr(CC6_TIMER_13); // Start Timer 13: CCU6_TCTR4H |= 0x02
}

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
}
```



```

001 #include "main.h"
002 #include "read_song_string.h"
003
004 void SetOctaveNORMAL(void)
005 {
006     OctaveLOW = OFF; // clear Global Variable
007
008     SFR_PAGE(_cc0, noSST); // switch to page 0
009     CC6_vStopTmr(CC6_TIMER_13); // Stop Timer 13: CCU6_TCTR4H |= 0x0
010
011     SFR_PAGE(_cc1, noSST); // switch to page 1
012     CCU6_TCTR0H = 0x01; // prescaler = 2: load CCU6 timer 13
013
014     SFR_PAGE(_cc0, noSST); // switch to page 0
015     CC6_vStartTmr(CC6_TIMER_13); // Start Timer 13: CCU6_TCTR4H |= 0x
016 }
017
018 void SetOctaveLOW(void)
019 {
020     OctaveLOW = ON; // set Global Variable
021
022     SFR_PAGE(_cc0, noSST); // switch to page 0
023     CC6_vStopTmr(CC6_TIMER_13); // Stop Timer 13: CCU6_TCTR4H |= 0x0
024
025     SFR_PAGE(_cc1, noSST); // switch to page 1
026     CCU6_TCTR0H = 0x02; // prescaler = 4: load CCU6 timer 13
027
028     SFR_PAGE(_cc0, noSST); // switch to page 0
029     CCU6_TCTR4H = 0x02; // Start Timer 13: CCU6_TCTR4H |= 0x
030 }
031
032 // Note:
033 // The function read_song_string() is a recursive function and will be
034 // The local variable substr is only used within one function-call to
035 // can be destroyed from one function-call to another.
036 // substr could also be defined as either a static or a global variab
037 // Therefore, the keyword reentrant is not needed.
038 void read_song_string (void)
039 {
040     unsigned char substr[4]=0;
041     current_note_length=old_note_length;
042
043     switch (song[pos])
044     {
045         // select note:
046         case 'C': note=0;
047             switch (song[++pos])
048             {

```




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 Reader - [XC88xCLM_UM_v1_1.pdf]

File Edit View Document Tools Window Help

Bookmarks

Pages

Page Attachments

Comments

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

14-60

V1.1, 2007-05

infineon

XC886/888CLM

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 $f_{T13} = f_{CCU} / 2^{<T13CLK>}$. 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 $f_{T13} = f_{CCU} / 128$

412 of 661

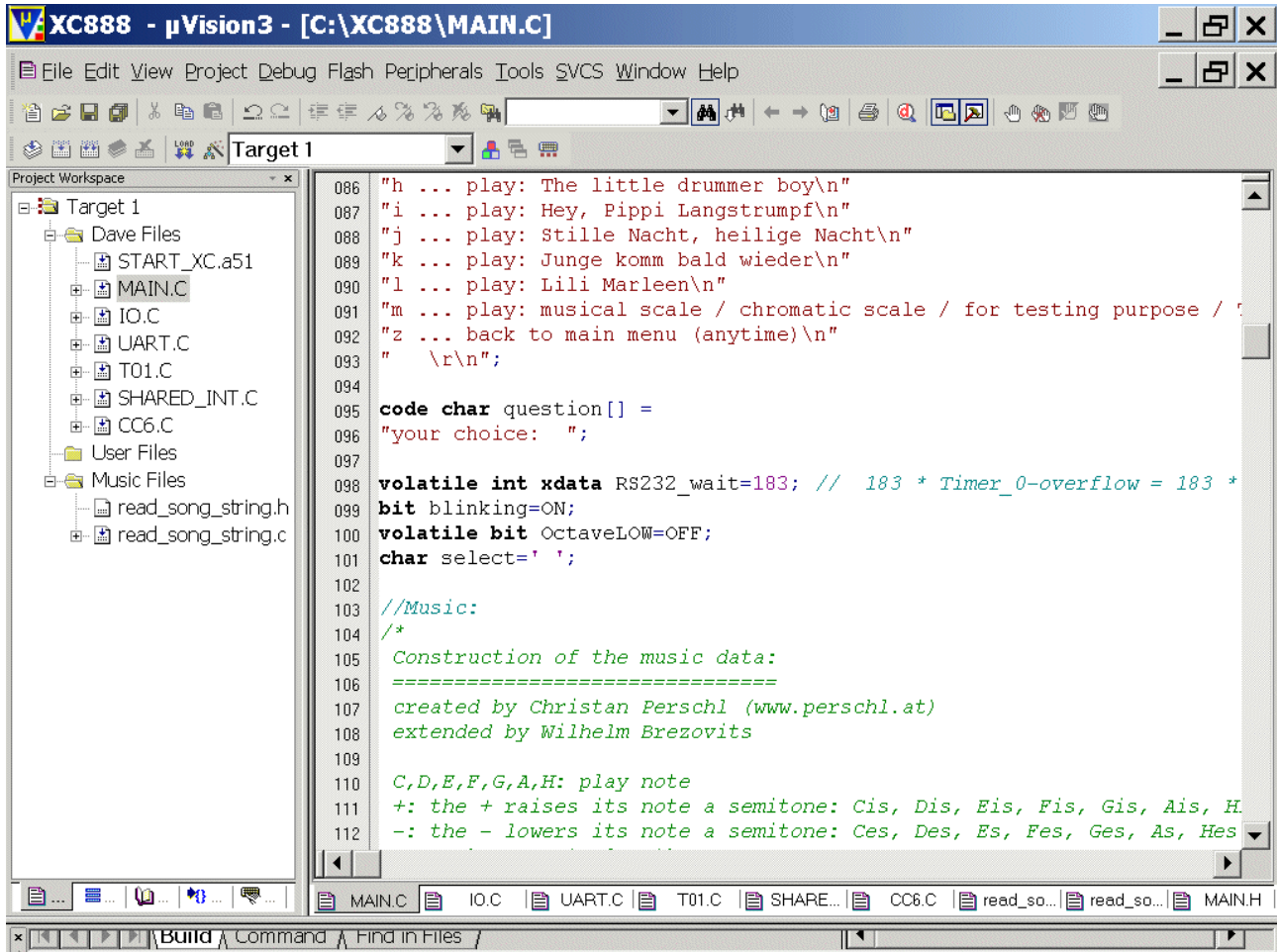
Application Note

269

V2.0, 2007-11

Double click **MAIN.C** and insert Global Variable **OctaveLOW**:

volatile bit OctaveLOW=OFF;



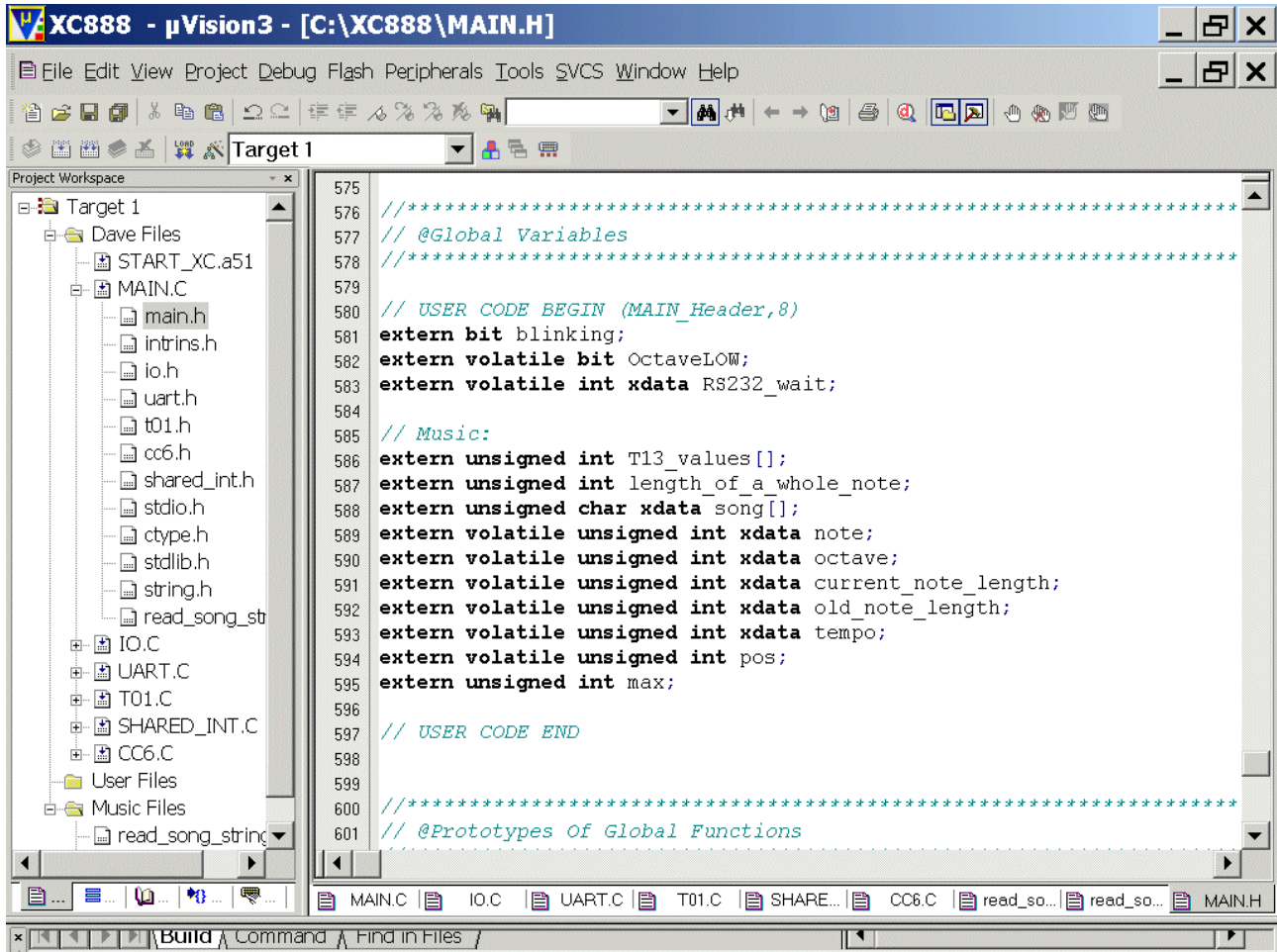
```

086 "h ... play: The little drummer boy\n"
087 "i ... play: Hey, Pippi Langstrumpf\n"
088 "j ... play: Stille Nacht, heilige Nacht\n"
089 "k ... play: Junge komm bald wieder\n"
090 "l ... play: Lili Marleen\n"
091 "m ... play: musical scale / chromatic scale / for testing purpose / "
092 "z ... back to main menu (anytime)\n"
093 "  \r\n";
094
095 code char question[] =
096 "your choice: ";
097
098 volatile int xdata RS232_wait=183; // 183 * Timer_0-overflow = 183 *
099 bit blinking=ON;
100 volatile bit OctaveLOW=OFF;
101 char select=' ';
102
103 //Music:
104 /*
105  Construction of the music data:
106  =====
107  created by Christian Perschl (www.perschl.at)
108  extended by Wilhelm Brezovits
109
110  C,D,E,F,G,A,H: play note
111  +: the + raises its note a semitone: Cis, Dis, Eis, Fis, Gis, Ais, H
112  -: the - lowers its note a semitone: Ces, Des, Es, Fes, Ges, As, Hes

```

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

extern volatile bit OctaveLOW;



Double click `read_song_string.c` and change code

from:

```
// 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 :    ; break;
}
pos++;
read_song_string(); break;
```

to:

```
// 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;
```

The screenshot shows the XC888 - µVision3 IDE. The Project Workspace on the left lists the files in the 'Target 1' project. The main code editor displays the following C code:

```

139     case '4':current_note_length=length_of_a_whole_note/4; break;
140     case '8':current_note_length=length_of_a_whole_note/8; break;
141     default :
142         ; break;
143     }
144     note=12;
145     pos++;
146     // adjust octave:
147     case 'O': switch (song[++pos])
148     {
149         case '0': octave=1; break;
150         case '1': octave=2; break;
151         case '2': octave=4; break;
152         case '3': octave=8; break;
153         default : if (song[pos]=='L') octave=1, SetOctaveLOW();
154                   if (song[pos]=='N') octave=1, SetOctaveNORMAL();
155                   break;
156     }
157     pos++;
158     read_song_string();
159     // tempo:
160     case 'T': pos++;
161               substr[3]=0; //string termination
162               if (song[pos]=='1')
163               {
164                   substr[0]=song[pos];
165               }

```

Double click **SHARED_INT.C** and **change** code

from:

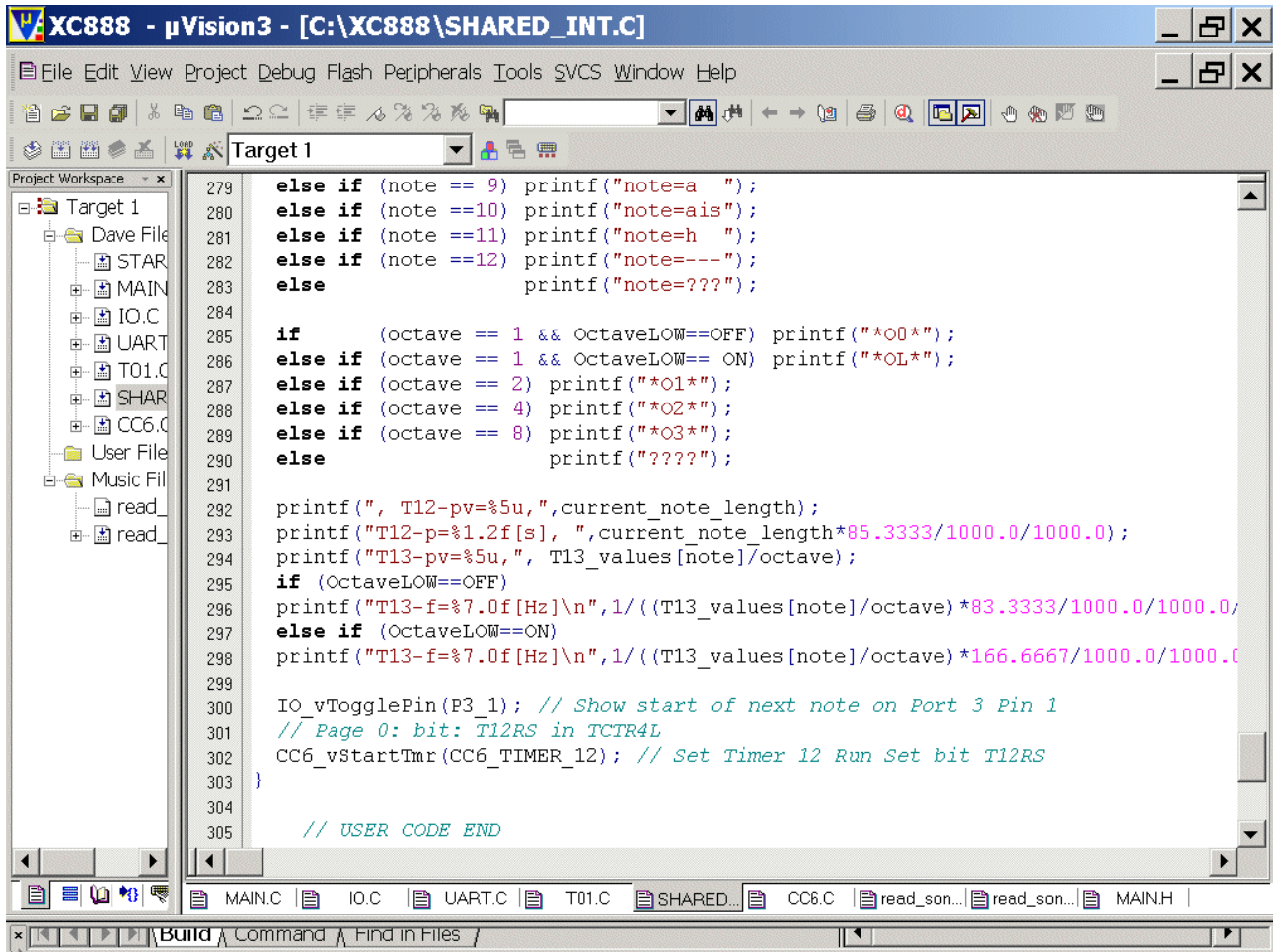
```
if      (octave == 1) printf("'  ");
else if (octave == 2) printf("' ' ");
else if (octave == 4) printf("' ' ' ");
else if (octave == 8) printf("' ' ' ' ");
else
    printf("????");

printf("  T12-pv=%5u," ,current_note_length);
printf("T12-p=%1.2f[s] , ",current_note_length*85.3333/1000.0/1000.0);
printf("T13-pv=%5u," , T13_values[note]/octave);
printf("T13-
f=%7.0f [Hz] \n",1/((T13_values[note]/octave)*83.3333/1000.0/1000.0/1000.0));
```

to:

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

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



```

279 else if (note == 9) printf("note=a ");
280 else if (note ==10) printf("note=ais");
281 else if (note ==11) printf("note=h ");
282 else if (note ==12) printf("note=---");
283 else
284     printf("note=???");
285
286 if      (octave == 1 && OctaveLOW==OFF) printf("O0*");
287 else if (octave == 1 && OctaveLOW== ON) printf("OL*");
288 else if (octave == 2) printf("O1*");
289 else if (octave == 4) printf("O2*");
290 else if (octave == 8) printf("O3*");
291 else
292     printf("???");
293
294 printf(", T12-pv=%5u,",current_note_length);
295 printf("T12-p=%1.2f[s], ",current_note_length*85.3333/1000.0/1000.0);
296 printf("T13-pv=%5u,", T13_values[note]/octave);
297 if (OctaveLOW==OFF)
298     printf("T13-f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*83.3333/1000.0/1000.0/
299     else if (OctaveLOW==ON)
300     printf("T13-f=%7.0f[Hz]\n",1/((T13_values[note]/octave)*166.6667/1000.0/1000.0/
301
302 IO_vTogglePin(P3_1); // Show start of next note on Port 3 Pin 1
303 // Page 0: bit: T12RS in TCTR4L
304 CC6_vStartTmr(CC6_TIMER_12); // Set Timer 12 Run Set bit T12RS
305 }
306
307 // USER CODE END
  
```


Double click **MAIN.C** and **change** Global Variable [songstring: **Hey, Pippi Langstrumpf (song i)**]:

from:

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

```
code unsigned char
```

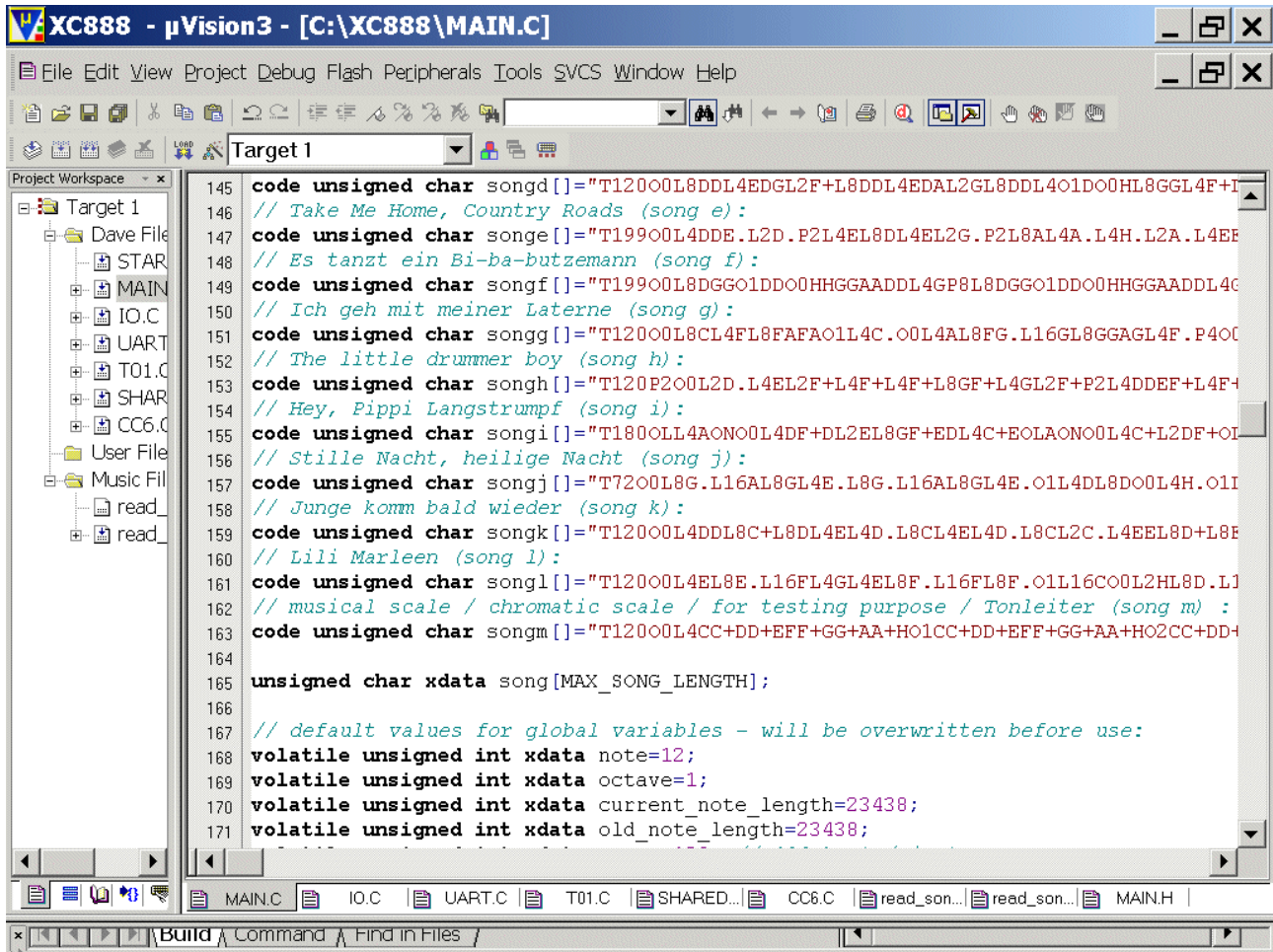
```
songi[]="T180O0L4ADF+DL2EL8GF+EDL4C+EAC+L2DF+L4ADF+DL2EL8GF+EDL4C+EAC  
+DP4P2L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+EE  
DC+DP4P2L2F+GAHO1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2GL4HAGF+EL2F+GL4  
AF+GAL2HO1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2GL4HAGF+EL2F+EDP2";
```

to:

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

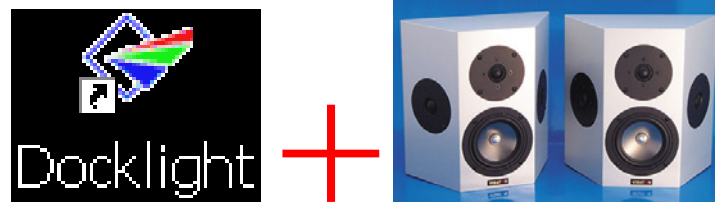
```
code unsigned char
```

```
songi[]="T180OLL4AONOO0L4DF+DL2EL8GF+EDL4C+EOLAONOO0L4C+L2DF+OLL4AONOO  
0L4DF+DL2EL8GF+EDL4C+EOLL4AONOO0L4C+DP4P2OLL4AONOO0L4DF+DL2EL8GF+ED  
L4C+EOLAONOO0L4C+L2DF+OLL4AONOO0L4DF+DL2EL8GF+EDL4C+EOLL4AONOO0L4C+  
DP4P2O0L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+  
EEDC+DP4L2F+GAH.O1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+GL  
4AF+GAL2H.O1L4DC+O0L4HAGL2A.O1L4C+O0L4HAGF+L2G.L4HAGF+EL2F+EDP2";
```



```

145 code unsigned char songd[]="T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL401DO0HL8GGL4F+I
146 // Take Me Home, Country Roads (song e):
147 code unsigned char songe[]="T19900L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EF
148 // Es tanzt ein Bi-ba-butzemann (song f):
149 code unsigned char songf[]="T19900L8DGG01DD00HHGGAADDL4GP8L8DGG01DD00HHGGAADDL4G
150 // Ich geh mit meiner Laterne (song g):
151 code unsigned char songg[]="T12000L8CL4FL8FAFA01L4C.00L4AL8FG.L16GL8GGAGL4F.P40C
152 // The little drummer boy (song h):
153 code unsigned char songh[]="T120P200L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+
154 // Hey, Pippi Langstrumpf (song i):
155 code unsigned char songi[]="T1800LL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+OI
156 // Stille Nacht, heilige Nacht (song j):
157 code unsigned char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8D00L4H.O1I
158 // Junge komm bald wieder (song k):
159 code unsigned char songk[]="T12000L4DDL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4EEL8D+L8E
160 // Lili Marleen (song l):
161 code unsigned char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16C00L2HL8D.LJ
162 // musical scale / chromatic scale / for testing purpose / Tonleiter (song m) :
163 code unsigned char songm[]="T12000L4CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+AA+HO2CC+DD+
164
165 unsigned char xdata song[MAX_SONG_LENGTH];
166
167 // default values for global variables - will be overwritten before use:
168 volatile unsigned int xdata note=12;
169 volatile unsigned int xdata octave=1;
170 volatile unsigned int xdata current_note_length=23438;
171 volatile unsigned int xdata old_note_length=23438;
  
```



See and hear:

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

```
your choice: i
playing: Hey, Pippi Langstrumpf
song-length = 384 Byte[s]
note=a *OL*, T12-pv= 5859,T12-p=0.50[s], T13-pv=27273,T13-f= 220 [Hz]
note=d *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=fis*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=32389,T13-f= 370 [Hz]
note=d *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=e *O0*, T12-pv=11719,T12-p=1.00[s], T13-pv=36364,T13-f= 330 [Hz]
note=g *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=30612,T13-f= 392 [Hz]
note=fis*O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=32389,T13-f= 370 [Hz]
note=e *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=36364,T13-f= 330 [Hz]
note=d *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=40816,T13-f= 294 [Hz]
note=cis*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=43309,T13-f= 277 [Hz]
note=e *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=a *OL*, T12-pv= 5859,T12-p=0.50[s], T13-pv=27273,T13-f= 220 [Hz]
note=cis*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=43309,T13-f= 277 [Hz]
note=d *O0*, T12-pv=11719,T12-p=1.00[s], T13-pv=40816,T13-f= 294 [Hz]
note=fis*O0*, T12-pv=11719,T12-p=1.00[s], T13-pv=32389,T13-f= 370 [Hz]
note=a *OL*, T12-pv= 5859,T12-p=0.50[s], T13-pv=27273,T13-f= 220 [Hz]
note=d *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=fis*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=32389,T13-f= 370 [Hz]
note=d *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=e *O0*, T12-pv=11719,T12-p=1.00[s], T13-pv=36364,T13-f= 330 [Hz]
note=g *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=30612,T13-f= 392 [Hz]
note=fis*O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=32389,T13-f= 370 [Hz]
note=e *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=36364,T13-f= 330 [Hz]
note=d *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=40816,T13-f= 294 [Hz]
note=cis*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=43309,T13-f= 277 [Hz]
note=e *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=a *OL*, T12-pv= 5859,T12-p=0.50[s], T13-pv=27273,T13-f= 220 [Hz]
note=cis*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=43309,T13-f= 277 [Hz]
note=d *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=---*O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv= 200,T13-f= 60000 [Hz]
note=---*O0*, T12-pv=11719,T12-p=1.00[s], T13-pv= 200,T13-f= 60000 [Hz]
note=a *OL*, T12-pv= 5859,T12-p=0.50[s], T13-pv=27273,T13-f= 220 [Hz]
```

[illegible]


```

note=a *00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=cis*01*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=21654,T13-f= 554 [Hz]
note=h *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=g *00*, T12-pv=17578,T12-p=1.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=h *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=fis*00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=g *00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=h *00*, T12-pv=17578,T12-p=1.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=d *01*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=20408,T13-f= 588 [Hz]
note=cis*01*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=21654,T13-f= 554 [Hz]
note=h *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=a *00*, T12-pv=17578,T12-p=1.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=cis*01*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=21654,T13-f= 554 [Hz]
note=h *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=g *00*, T12-pv=17578,T12-p=1.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=h *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=fis*00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=32389,T13-f= 370 [Hz]
note=e *00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=d *00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=40816,T13-f= 294 [Hz]
note=---*00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv= 200,T13-f= 60000 [Hz]
End of the song reached (pos= 385 of max 384).

```

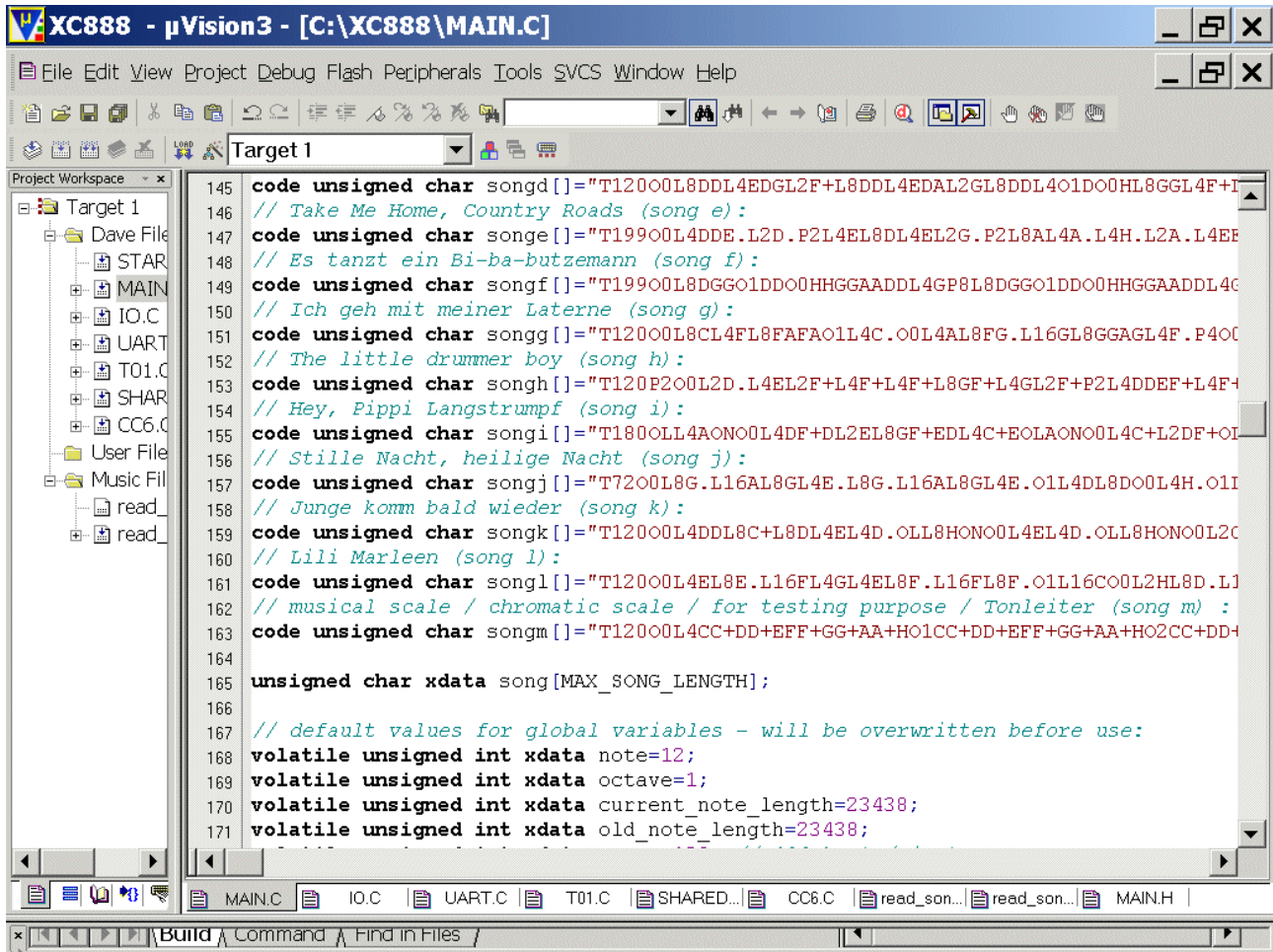
Double click **MAIN.C** and change Global Variable
[songstring: **Junge komm bald wieder (song k):**]:

from:

```
// Junge komm bald wieder (song k):
code unsigned char
songk[]="T120O0L4DDL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4EEL8D+L8EL4F+L4E.L8E
L4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+L8EL4EL2DL4EL4D.L8CL2C.L4D
DL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4EEL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8D
DDDDDDDDL4DP8L8DL8D+L8DDDDDL8D+L8DL4DP8L8DL8EEEEEL2GP8L8EL1DP8L8
DL8EEEL4E.P8L8GGGF+L8GL1A";
```

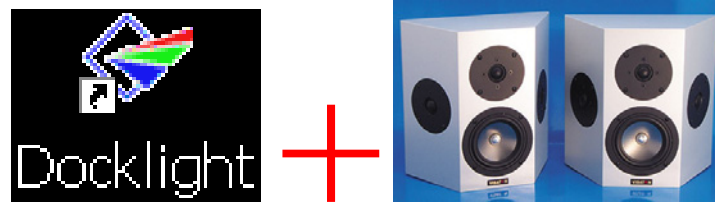
to:

```
// Junge komm bald wieder (song k):
code unsigned char
songk[]="T120O0L4DDL8C+L8DL4EL4D.OLL8HONOO0L4EL4D.OLL8HONOO0L2C.L4EEL8D+
L8EL4F+L4E.L8EL4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+.L8EL4EL2DL4E
L4D.L8COLL2H.ONOO0L4DDL8C+L8DL4EL4D.OLL8HONOO0L4EL4D.OLL8HONOO0L2C.L4E
EL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8DDDDDDDDDDL4DP8L8DL8D+L8DDDDDL8D
+L8DL4DP8L8DL8EEEEEL2GP8L8EL1DP8L8DL8EEEL4E.P8L8GGGF+L8GL1A.";
```



```

145 code unsigned char songd[]="T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL401DO0HL8GGL4F+I
146 // Take Me Home, Country Roads (song e):
147 code unsigned char songe[]="T19900L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EF
148 // Es tanzt ein Bi-ba-butzemann (song f):
149 code unsigned char songf[]="T19900L8DGG01DD00HHGGAADDL4GP8L8DGG01DD00HHGGAADDL4G
150 // Ich geh mit meiner Laterne (song g):
151 code unsigned char songg[]="T12000L8CL4FL8FAFA01L4C.00L4AL8FG.L16GL8GGAGL4F.P40C
152 // The little drummer boy (song h):
153 code unsigned char songh[]="T120P200L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+
154 // Hey, Pippi Langstrumpf (song i):
155 code unsigned char songi[]="T1800LL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+OI
156 // Stille Nacht, heilige Nacht (song j):
157 code unsigned char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8D00L4H.O1I
158 // Junge komm bald wieder (song k):
159 code unsigned char songk[]="T12000L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C
160 // Lili Marleen (song l):
161 code unsigned char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16C00L2HL8D.LJ
162 // musical scale / chromatic scale / for testing purpose / Tonleiter (song m) :
163 code unsigned char songm[]="T12000L4CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+AA+HO2CC+DD+
164
165 unsigned char xdata song[MAX_SONG_LENGTH];
166
167 // default values for global variables - will be overwritten before use:
168 volatile unsigned int xdata note=12;
169 volatile unsigned int xdata octave=1;
170 volatile unsigned int xdata current_note_length=23438;
171 volatile unsigned int xdata old_note_length=23438;
  
```



See and hear:

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

```
your choice: k
playing: Junge komm bald wieder
song-length = 324 Byte[s]
note=d *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=d *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=cis*00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=43309,T13-f= 277 [Hz]
note=d *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=40816,T13-f= 294 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=d *00*, T12-pv= 8788,T12-p=0.75[s], T13-pv=40816,T13-f= 294 [Hz]
note=h *0L*, T12-pv= 2929,T12-p=0.25[s], T13-pv=24291,T13-f= 247 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=d *00*, T12-pv= 8788,T12-p=0.75[s], T13-pv=40816,T13-f= 294 [Hz]
note=h *0L*, T12-pv= 2929,T12-p=0.25[s], T13-pv=24291,T13-f= 247 [Hz]
note=c *00*, T12-pv=17578,T12-p=1.50[s], T13-pv=45802,T13-f= 262 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=dis*00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=38590,T13-f= 311 [Hz]
note=e *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=36364,T13-f= 330 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=32389,T13-f= 370 [Hz]
note=e *00*, T12-pv= 8788,T12-p=0.75[s], T13-pv=36364,T13-f= 330 [Hz]
note=e *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=36364,T13-f= 330 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=32389,T13-f= 370 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=d *00*, T12-pv=17578,T12-p=1.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=30612,T13-f= 392 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=30612,T13-f= 392 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=30612,T13-f= 392 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=c *00*, T12-pv=11719,T12-p=1.00[s], T13-pv=45802,T13-f= 262 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=32389,T13-f= 370 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=36364,T13-f= 330 [Hz]
note=d *00*, T12-pv=17578,T12-p=1.50[s], T13-pv=40816,T13-f= 294 [Hz]
note=fis*00*, T12-pv= 5859,T12-p=0.50[s], T13-pv=32389,T13-f= 370 [Hz]
```


[illegible]

```
note=e *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=36364,T13-f= 330 [Hz]
note=e *00*, T12-pv= 8788,T12-p=0.75[s], T13-pv=36364,T13-f= 330 [Hz]
note=---*00*, T12-pv= 2929,T12-p=0.25[s], T13-pv= 200,T13-f= 60000 [Hz]
note=g *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=30612,T13-f= 392 [Hz]
note=g *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=30612,T13-f= 392 [Hz]
note=g *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=30612,T13-f= 392 [Hz]
note=fis*00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=32389,T13-f= 370 [Hz]
note=g *00*, T12-pv= 2929,T12-p=0.25[s], T13-pv=30612,T13-f= 392 [Hz]
note=a *00*, T12-pv=35157,T12-p=3.00[s], T13-pv=27273,T13-f= 440 [Hz]
End of the song reached (pos= 325 of max 324).
```



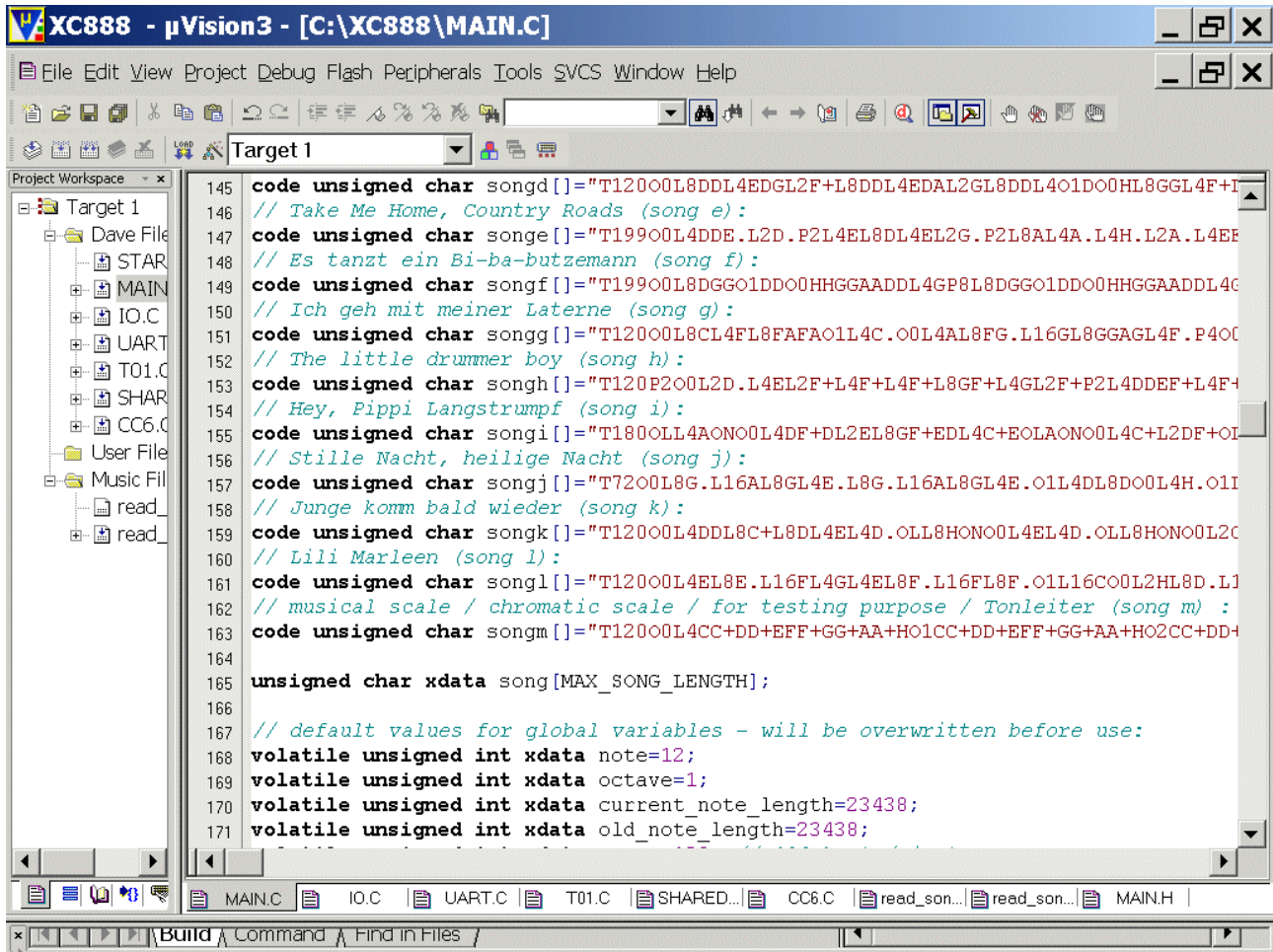
Double click **MAIN.C** and **change** Global Variable [songstring: **Lili Marleen (song 1)**]:

from:

```
// Lili Marleen (song 1):  
code unsigned char  
songl[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL  
8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL  
4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CO0L4EL4G.L8FL4FL4CL2C.";
```

to:

```
// Lili Marleen (song 1):  
code unsigned char  
songl[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL  
8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL  
4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HONOO0L2C.";
```



```

145 code unsigned char songd[]="T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL401DO0HL8GGL4F+I
146 // Take Me Home, Country Roads (song e):
147 code unsigned char songe[]="T19900L4DDE.L2D.P2L4EL8DL4EL2G.P2L8AL4A.L4H.L2A.L4EF
148 // Es tanzt ein Bi-ba-butzemann (song f):
149 code unsigned char songf[]="T19900L8DGG01DD00HHGGAADDL4GP8L8DGG01DD00HHGGAADDL4G
150 // Ich geh mit meiner Laterne (song g):
151 code unsigned char songg[]="T12000L8CL4FL8FAFA01L4C.00L4AL8FG.L16GL8GGAGL4F.P40C
152 // The little drummer boy (song h):
153 code unsigned char songh[]="T120P200L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+
154 // Hey, Pippi Langstrumpf (song i):
155 code unsigned char songi[]="T1800LL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+OI
156 // Stille Nacht, heilige Nacht (song j):
157 code unsigned char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8D00L4H.O1I
158 // Junge komm bald wieder (song k):
159 code unsigned char songk[]="T12000L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C
160 // Lili Marleen (song l):
161 code unsigned char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16C00L2HL8D.LJ
162 // musical scale / chromatic scale / for testing purpose / Tonleiter (song m) :
163 code unsigned char songm[]="T12000L4CC+DD+EFF+GG+AA+HO1CC+DD+EFF+GG+AA+HO2CC+DD+
164
165 unsigned char xdata song[MAX_SONG_LENGTH];
166
167 // default values for global variables - will be overwritten before use:
168 volatile unsigned int xdata note=12;
169 volatile unsigned int xdata octave=1;
170 volatile unsigned int xdata current_note_length=23438;
171 volatile unsigned int xdata old_note_length=23438;
  
```




See and hear:

```


your choice: 1
playing: Lili Marleen
song-length = 202 Byte[s]
note=e *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=e *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=f *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=f *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=f *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=f *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=c *01*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=22901,T13-f= 524 [Hz]
note=h *00*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=d *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=40816,T13-f= 294 [Hz]
note=d *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=40816,T13-f= 294 [Hz]
note=d *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=40816,T13-f= 294 [Hz]
note=e *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=f *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=f *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=g *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=h *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=f *00*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=e *00*, T12-pv= 8788,T12-p=0.75 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=c *00*, T12-pv= 2929,T12-p=0.25 [s] , T13-pv=45802,T13-f= 262 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=h *00*, T12-pv= 4393,T12-p=0.37 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=c *01*, T12-pv= 1464,T12-p=0.12 [s] , T13-pv=22901,T13-f= 524 [Hz]
note=h *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=a *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=h *00*, T12-pv= 8788,T12-p=0.75 [s] , T13-pv=24291,T13-f= 494 [Hz]
note=a *00*, T12-pv= 2929,T12-p=0.25 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=f *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=a *00*, T12-pv= 8788,T12-p=0.75 [s] , T13-pv=27273,T13-f= 440 [Hz]
note=g *00*, T12-pv= 2929,T12-p=0.25 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=f *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=g *00*, T12-pv= 8788,T12-p=0.75 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=e *00*, T12-pv= 2929,T12-p=0.25 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=g *00*, T12-pv= 8788,T12-p=0.75 [s] , T13-pv=30612,T13-f= 392 [Hz]
note=f *00*, T12-pv= 2929,T12-p=0.25 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=f *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=34383,T13-f= 349 [Hz]
note=d *01*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=20408,T13-f= 588 [Hz]
note=c *01*, T12-pv=11719,T12-p=1.00 [s] , T13-pv=22901,T13-f= 524 [Hz]
note=---*01*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv= 100,T13-f= 120000 [Hz]
note=e *00*, T12-pv= 5859,T12-p=0.50 [s] , T13-pv=36364,T13-f= 330 [Hz]
note=g *00*, T12-pv= 8788,T12-p=0.75 [s] , T13-pv=30612,T13-f= 392 [Hz]

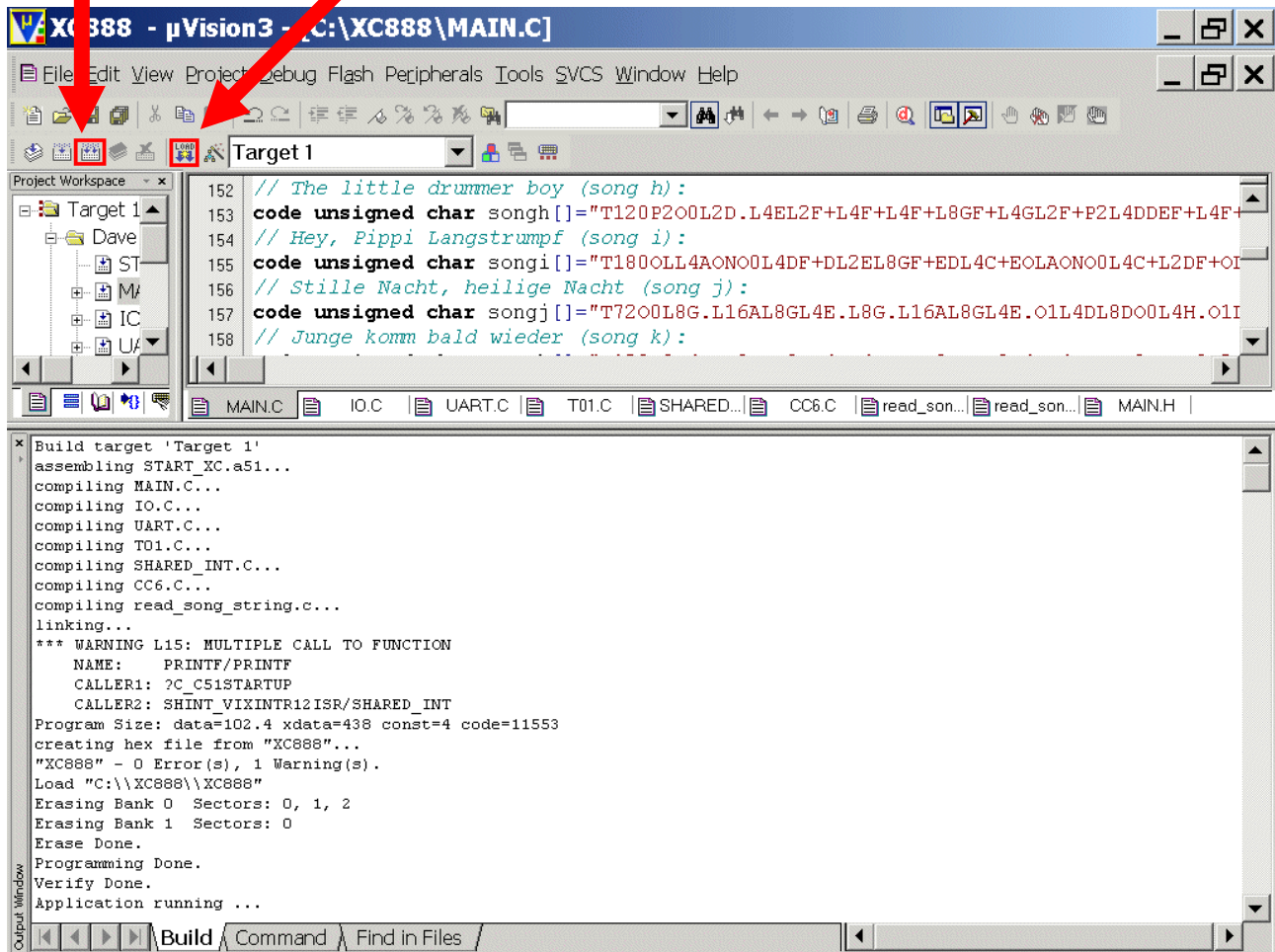
```

```
note=f  *O0*, T12-pv= 2929,T12-p=0.25[s], T13-pv=34383,T13-f= 349 [Hz]
note=f  *O0*, T12-pv= 5859,T12-p=0.50[s], T13-pv=34383,T13-f= 349 [Hz]
note=h  *OL*, T12-pv= 5859,T12-p=0.50[s], T13-pv=24291,T13-f= 247 [Hz]
note=c  *O0*, T12-pv=17578,T12-p=1.50[s], T13-pv=45802,T13-f= 262 [Hz]
End of the song reached (pos= 203 of max 202).
```

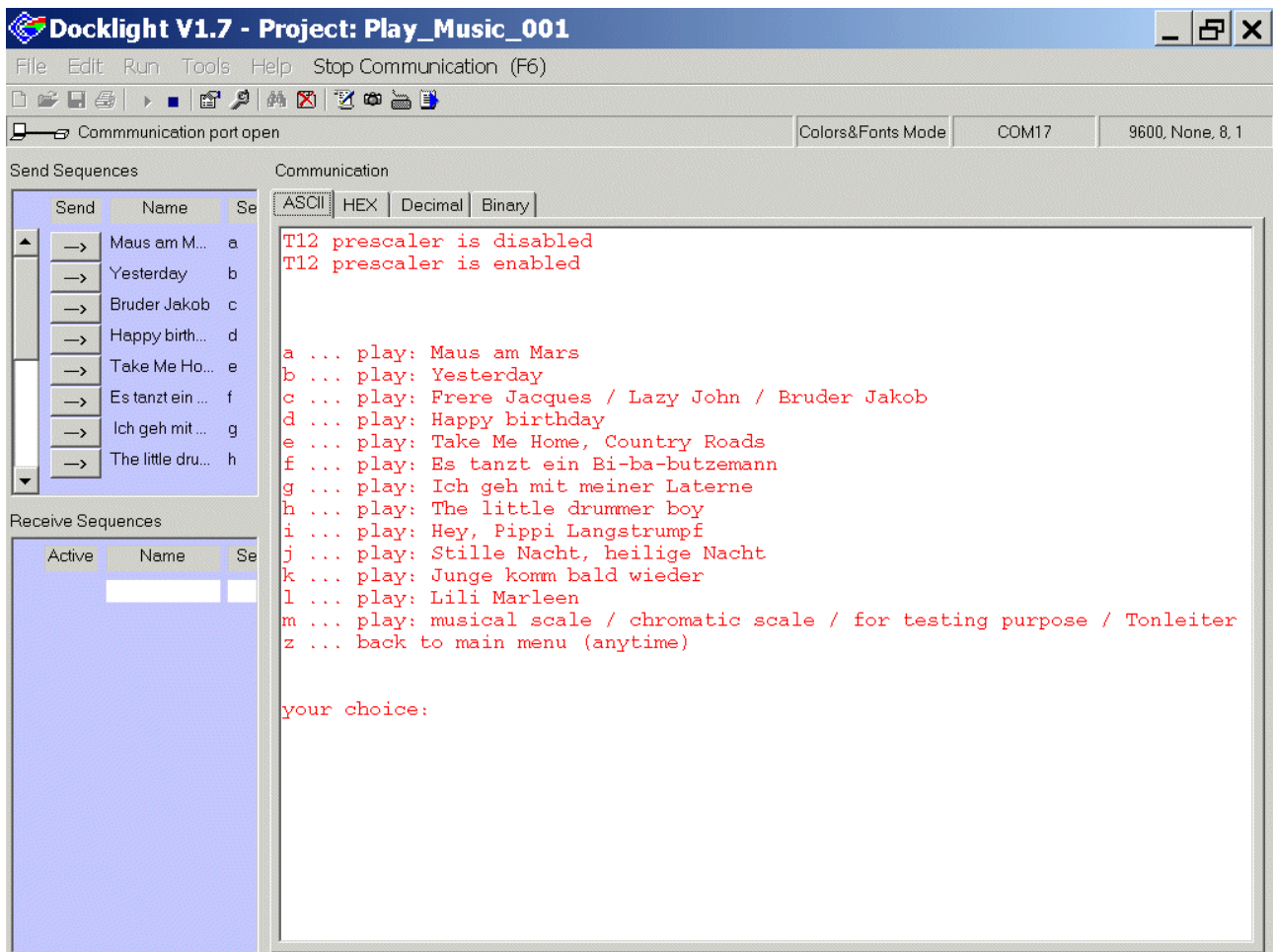


1.) click:  Rebuild all target files

2.) click:  Download to On-Chip-Flash Memory



Now you can **see** / **hear** / **enjoy** the results and there are no compromises (or wrong notes):



Have fun!

5.) 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
	$\frac{1}{2} \text{ (2 beats)} + (\frac{1}{2})/2 \text{ (1 beat)} = \frac{3}{4} \text{ (3 beats)}$

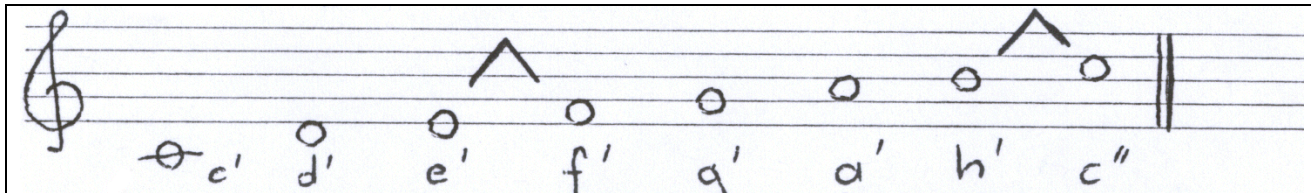
Note:

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:

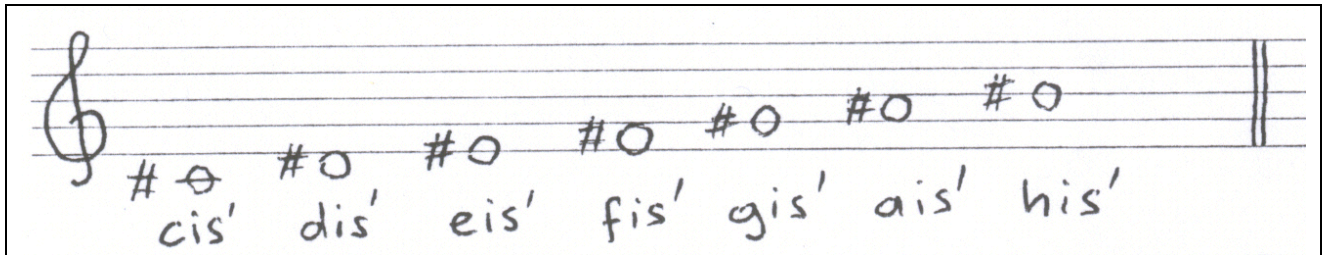
The notes C, D, E, F, G, A, H are named C, D, E, F, G, A, B in other countries.
In this document we stick to the German names.



Syntax used in our programming example:

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

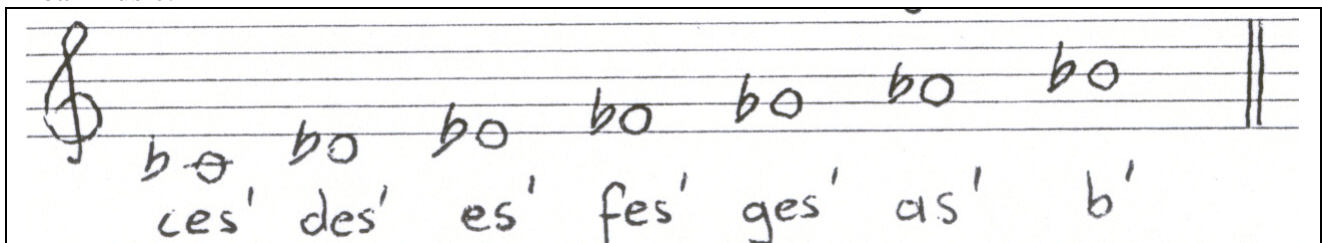
Real Music:



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:

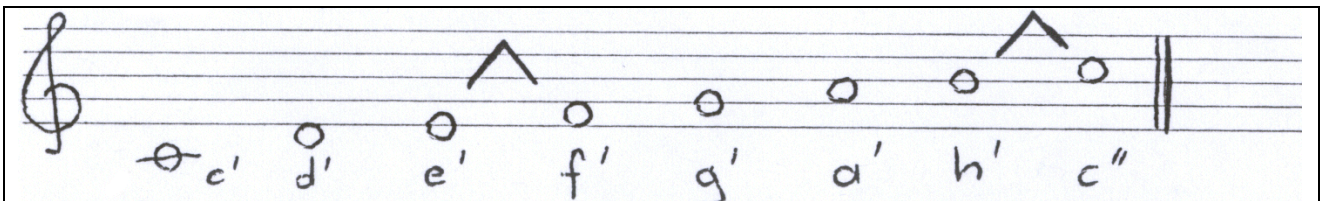
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

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 ...).

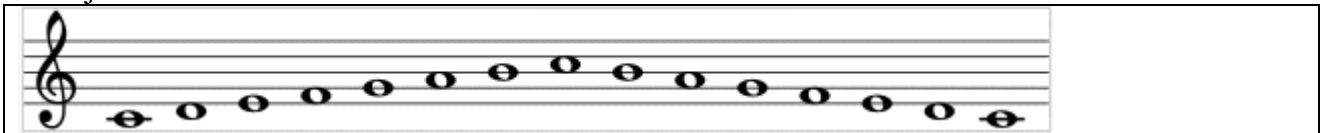
Syntax used in our programming example:

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

Real Music:

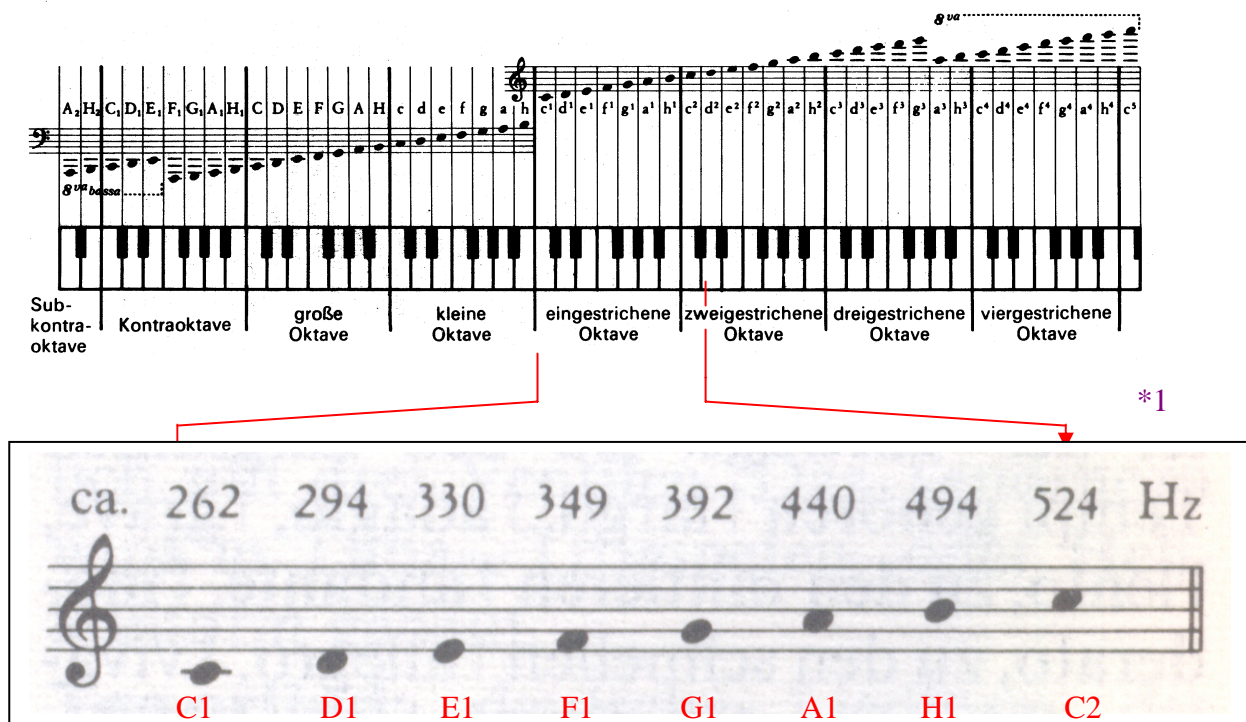


C major scale:



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

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



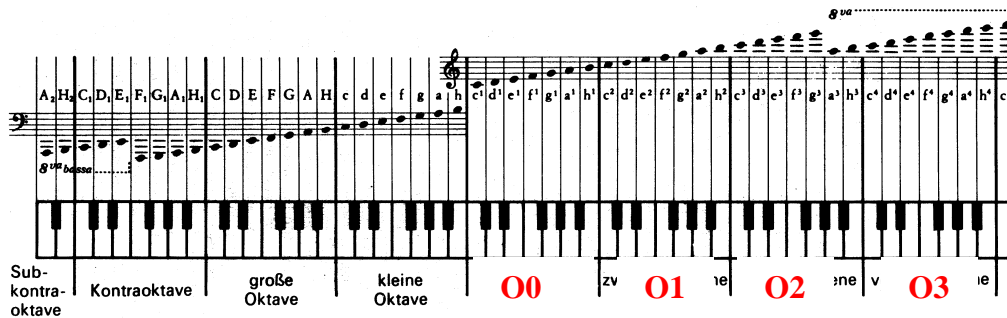
*2

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

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

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

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



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

```
unsigned int T13_values[] =
{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:

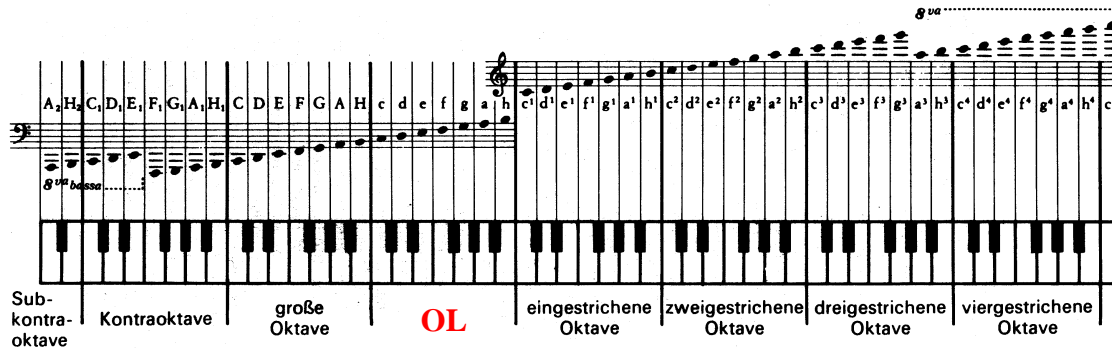
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

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:

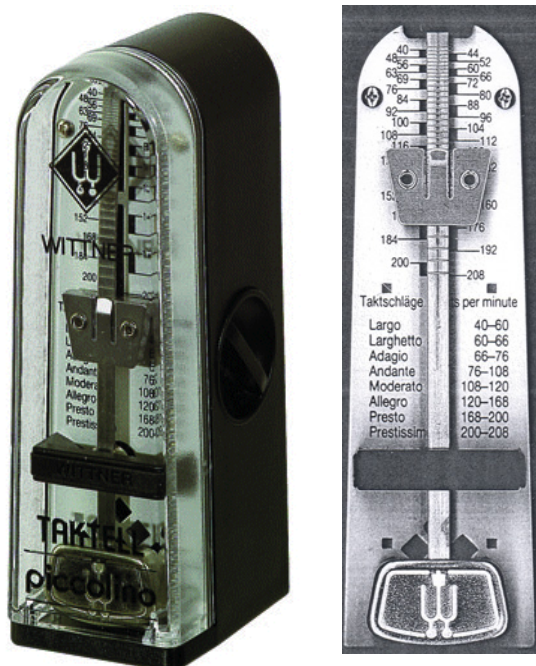
```
// T13, note-frequency:  
CCU6_T13PR=T13_values[note]/octave; // note-frequency  
CCU6_CC63SR=T13_values[note]/octave/2; // duty cycle = 50 %  
CCU6_vEnableShadowTransfer(CCU6_TIMER_13);
```

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:


```
CC6_vSetTmrPeriod(CC6_TIMER_12, (current_note_length/tempo*120));
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_0, (current_note_length/tempo*120)/100*95);
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_1, (current_note_length/tempo*120)/100*0);
CC6_vLoadChannelShadowRegister(CC6_CHANNEL_2, (current_note_length/tempo*120)/100*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:

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

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

```
// T12, note length:  
// period value note length  
CCU6_T12PR=current_note_length/tempo*120;  
// T12, duty cycle:  
// if CCU6_CC62SR == 0 -> 100% duty cycle for note length  
CCU6_CC62SR=(current_note_length/tempo*120);  
CCU6_vEnableShadowTransfer(CCU6_TIMER_12);
```

Implementing note length and note frequency on real hardware:

Using XC866 using CAPCOM6: 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 %

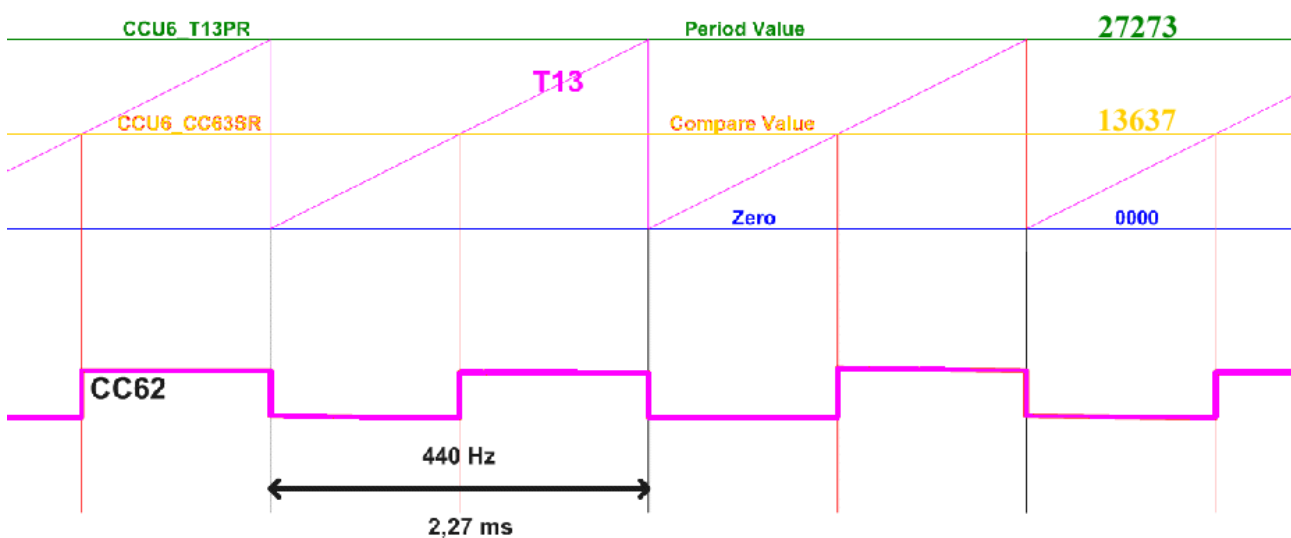
T13: note frequency (Octave = O0, O1, O2 and O3):

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 %

7.) Appendix: songs used

7.1.) 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.

7.2.) 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.

7.3.) Song c: Bruder Jakob:

Bruder Jakob

Französisches Kinderlied



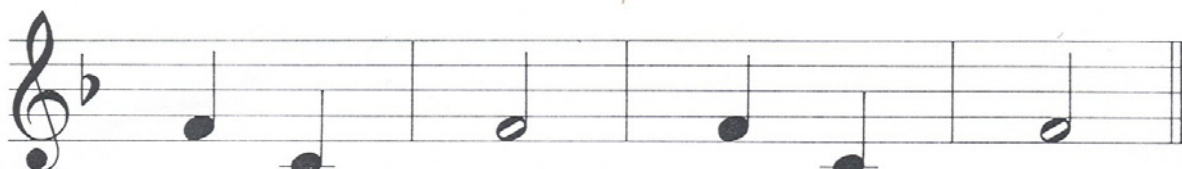
Bru- der Ja- kob, Bru- der Ja- kob,



schläfst du noch, schläfst du noch?



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



ding, dong, ding, ding dong, ding!

// Bruder Jakob (song c):

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

7.4.) 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!

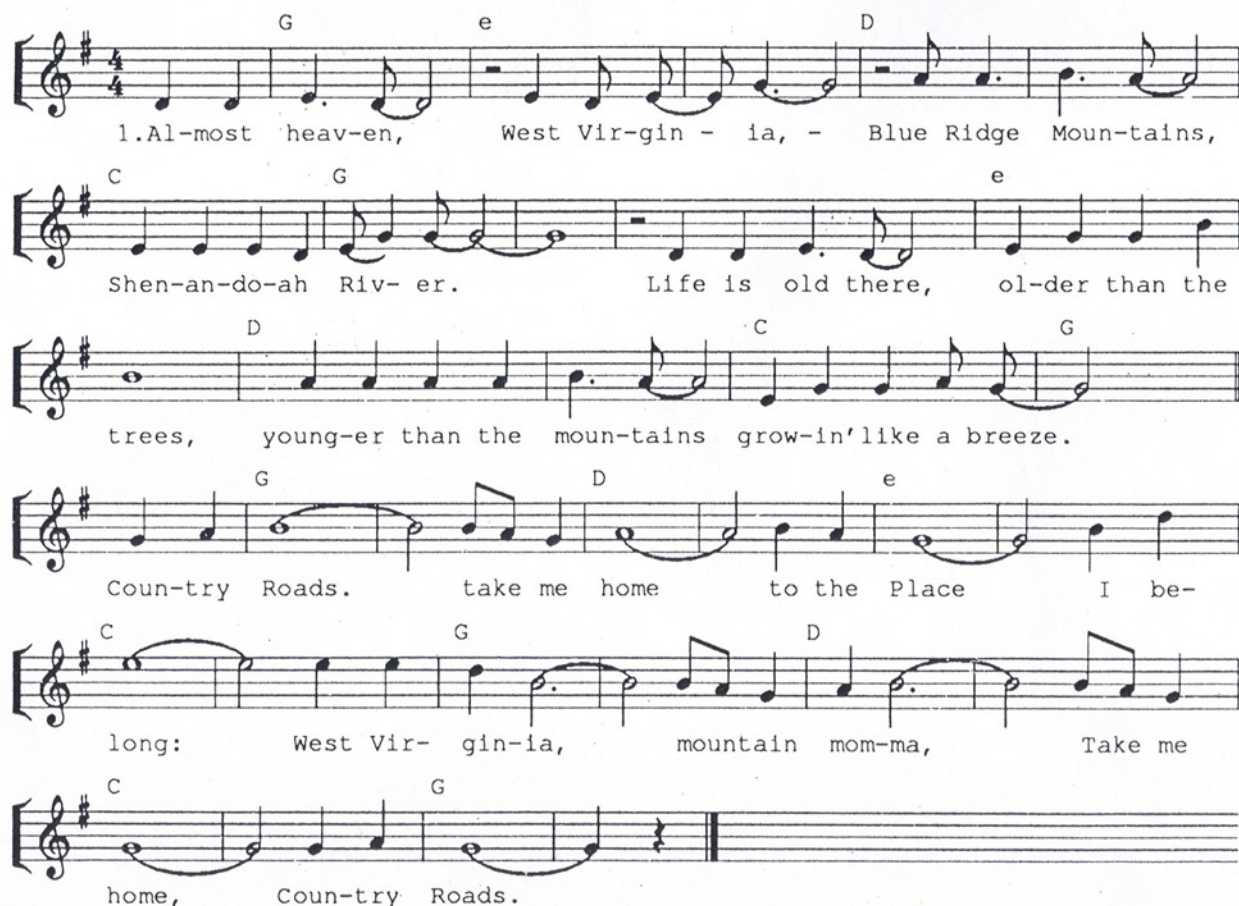
// Happy birthday (song d):

code unsigned char

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

7.5.) 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";
```


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

Es tanzt ein Bi-ba-butzemann

Volkswiese



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



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



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

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

code unsigned char

songf[]="T19900L8DGGO1DDO0HHGGAADDL4GP8L8DGGO1DDO0HHGGAADDL4GP8L8
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGGO1DDO0HHGGAADDL4G";

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

Ich geh mit meiner Laterne



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



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

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

code unsigned char

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

7.8.) Song h: The little drummer boy:

The little drummer boy



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

// 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";
```


7.9.) Song i: Hey, Pippi Langstrumpf:

Hey, Pippi Langstrumpf



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



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



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



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



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



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

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

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



Songstring used in chapter 3:

// Hey, Pippi Langstrumpf (song i):

code unsigned char

```
songi[]="T180O0L4ADF+DL2EL8GF+EDL4C+EAC+L2DF+L4ADF+DL2EL8GF+EDL4C+EAC
+DP4P2L2F+L4F+F+L2GL4GL8GF+L4EL8EEL4EL8EDL4C+DEP4L2F+L4F+F+L2GL4GF+EE
DC+DP4P2L2F+GAHO1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2GL4HAGF+EL2F+GL4
AF+GAL2HO1L4DC+O0L4HAGL2AO1L4C+O0L4HAGF+L2GL4HAGF+EL2F+EDP2";
```

Songstring used in chapter 4:

// 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";
```

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



Stille Nacht, heilige Nacht



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

code unsigned char

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

7.11.) 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, ...

Songstring used in chapter 3:

// Junge komm bald wieder (song k):

code unsigned char

```
songk[]="T120O0L4DDL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4EEL8D+L8EL4F+L4E.L8E  
L4GL4F+L4EL2D.L4GGGEL2CL4GF+L4EL2D.L4F+L4F+L8EL4EL2DL4EL4D.L8CL2C.L4D  
DL8C+L8DL4EL4D.L8CL4EL4D.L8CL2C.L4EEL8D+L8EL4F+L4E.L8EL4GF+L4AL2GP8L8D  
DDDDDDDDL4DP8L8DL8D+L8DDDDDL8D+L8DL4DP8L8DL8EEEEEEEL2GP8L8EL1DP8L8  
DL8EEEL4E.P8L8GGGF+L8GL1A";
```

Songstring used in chapter 4:

// 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+L4AL2GP8L8DDDDDDDDDDL4DP8L8DL8D+L8DDDDDL8D  
+L8DL4DP8L8DL8EEEEEEEL2GP8L8EL1DP8L8DL8EEEL4E.P8L8GGGF+L8GL1A.";
```


7.12.) Song 1: Lili Marleen:

Lili Marleen



Vor der Ka-ser-ne, vor dem gro-ßen Tor stand ei-ne La-ter-ne. Und steht sie noch da-vor, so woll'n wir uns da wie-der-sehn, bei der La-ter-ne woll'n wir stehn wie einst, Li-li Mar-leen, wie einst, Li-li Mar-leen.

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

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

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

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

Songstring used in chapter 3:

// Lili Marleen (song 1):

code unsigned char

```
song1[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CO0L4EL4G.L8FL4FL4CL2C.";
```

Songstring used in chapter 4:

// Lili Marleen (song 1):

code unsigned char

```
song1[]="T120O0L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL4FL8F.L16GL8H.L16AL8G.L16FL4E.L8CL4AL8H.O1L16CO0L4HL4AL4AL4GL4H.L8AL4GL4FL4A.L8GL4FEL4G.L8EL4G.L8FL4FO1L4DL2CP4O0L4EL4G.L8FL4FOLL4HON00L2C.";
```

7.13.) 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  
";
```

7.14.) 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 PWM Unit.

Have fun and enjoy working with microcontrollers with CCU6 modules!

Note:

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

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

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

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

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

8.) Thanks To



Maria, Christian, Hermann and Maureen for their support.



9.) Feedback (XC888 Playing Music):
Your opinion, suggestions and/or criticisms



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

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

email: mcdocu.comments@infineon.com

FAX: +43 (0) 4242 3020 5783



Your suggestions:

<http://www.infineon.com>

Published by Infineon Technologies AG