

# AP16138

## XE164

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



Microcontrollers



Never stop thinking

**Edition 2008-07-16**

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

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

#### **LEGAL DISCLAIMER**

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

#### **Information**

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

#### **Warnings**

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

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

---

**AP08048**

**Revision History:** 2008-06 V2.0

Previous Version: none

Page	Subjects (major changes since last revision)

**We Listen to Your Comments**

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

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



**Note:** Table of Contents see [page 12](#) and page 13.

### Introduction:

This “Appnote” is a Hands On Training / Cookery Book / step-by-step book.  
It will help inexperienced users to get familiar with the CAPCOM 6 / CCU6 module.  
This step-by-step book is a follow-up to AP16137.

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

### **Note:**

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

Have fun and enjoy the CAPCOM 6 module!

### **Note:**

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

### **Note:**

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

### **Note:**

At the time this document was written there was no Keil simulation support for the XE164 microcontroller. If you want to learn how to setup the Keil software simulated logic analyzer to view the PWM signals on the Keil simulator we also suggest taking a look at AP08068.





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

Playing music





**Note:**


Port\_0 pins used by our PWM module CCU61:

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

Port\_2 pins used as GPIO:

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

Port pins used:

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

## Using the UConnect-CAN XE164





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

### On-board header X400

15	13	11	9	7	5	3	1
CANH	P0.4	P0.5	P0.3	P0.2	P5.9	P15.0	GND
CANL	P0.7	P0.6	P0.0	P0.1	P5.8	P5.0	+5V
16	14	12	10	8	6	4	2

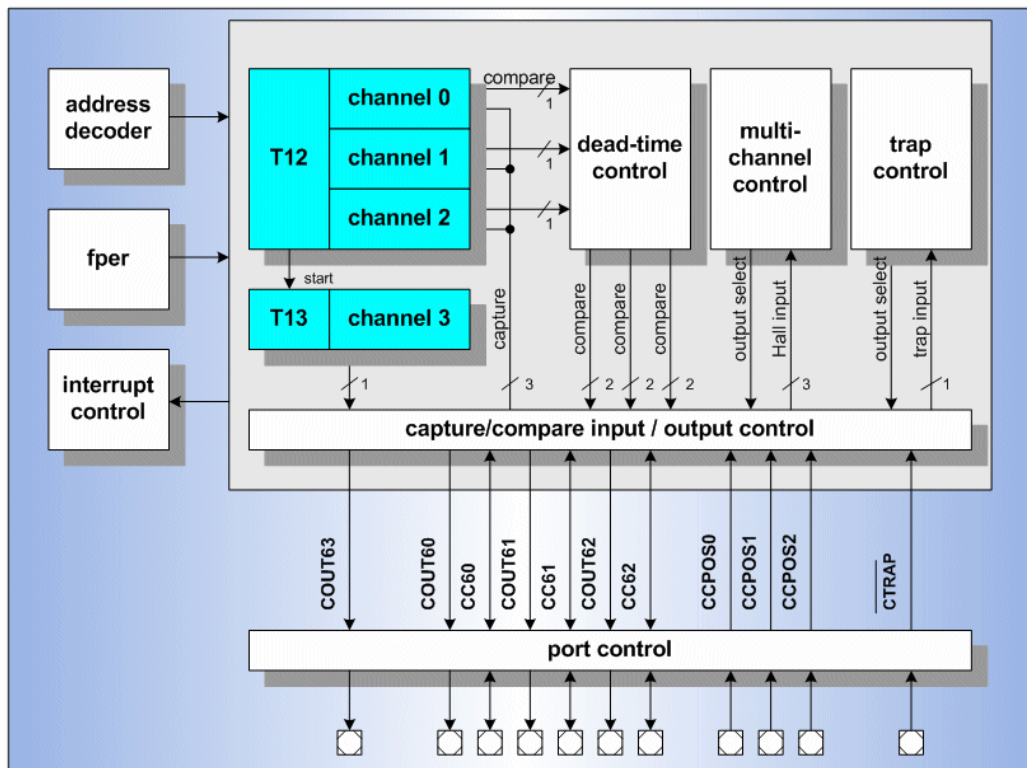
Pin number				
1	Ground			
2	+5V			
3	P15.0	ADC1_CH0		
4	P5.0	ADC0_CH0		
5	P5.9	ADC0_CH9	ADC1_CH9	CC2_T7IN CAPCOM2
6	P5.8	ADC0_CH8	ADC8_CH8	T12HRC / T13HRC CCU6x
7	P0.2	U1C0_SCK	CC62 CCU61	TXDC0 CAN0
8	P0.1	U1C0_DOUT	CC61 CCU61	TXDC0 CAN0
9	P0.3	U1C0_SELO	COUT60 CCU61	RXDC0B CAN0
10	P0.0	U1C0_DX0	CC60 CCU61	
11	P0.5	U1C1_SCK	COUT62 CCU61	
12	P0.6	U1C1_DOUT	COUT63 CCU61	TXDC1 CAN1
13	P0.4	U1C1_SELO	COUT61 CCU61	RXDC1B CAN1
14	P0.7	U1C1_DX0	U1C1_DX0	CTRAPB CCU61
15	CANH Signal from CAN transceiver			
16	CANL Signal from CAN transceiver			

#### Note:

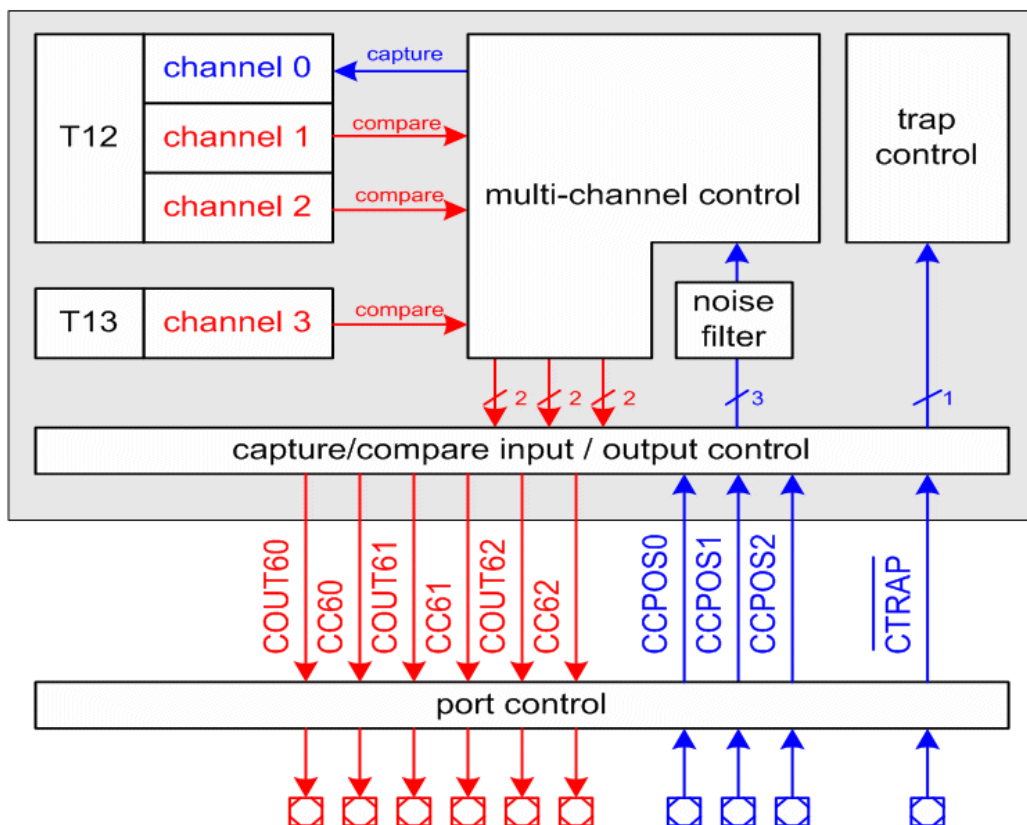
For further information, please refer to the [XE164 UConnect Manual, V.1.1](#).



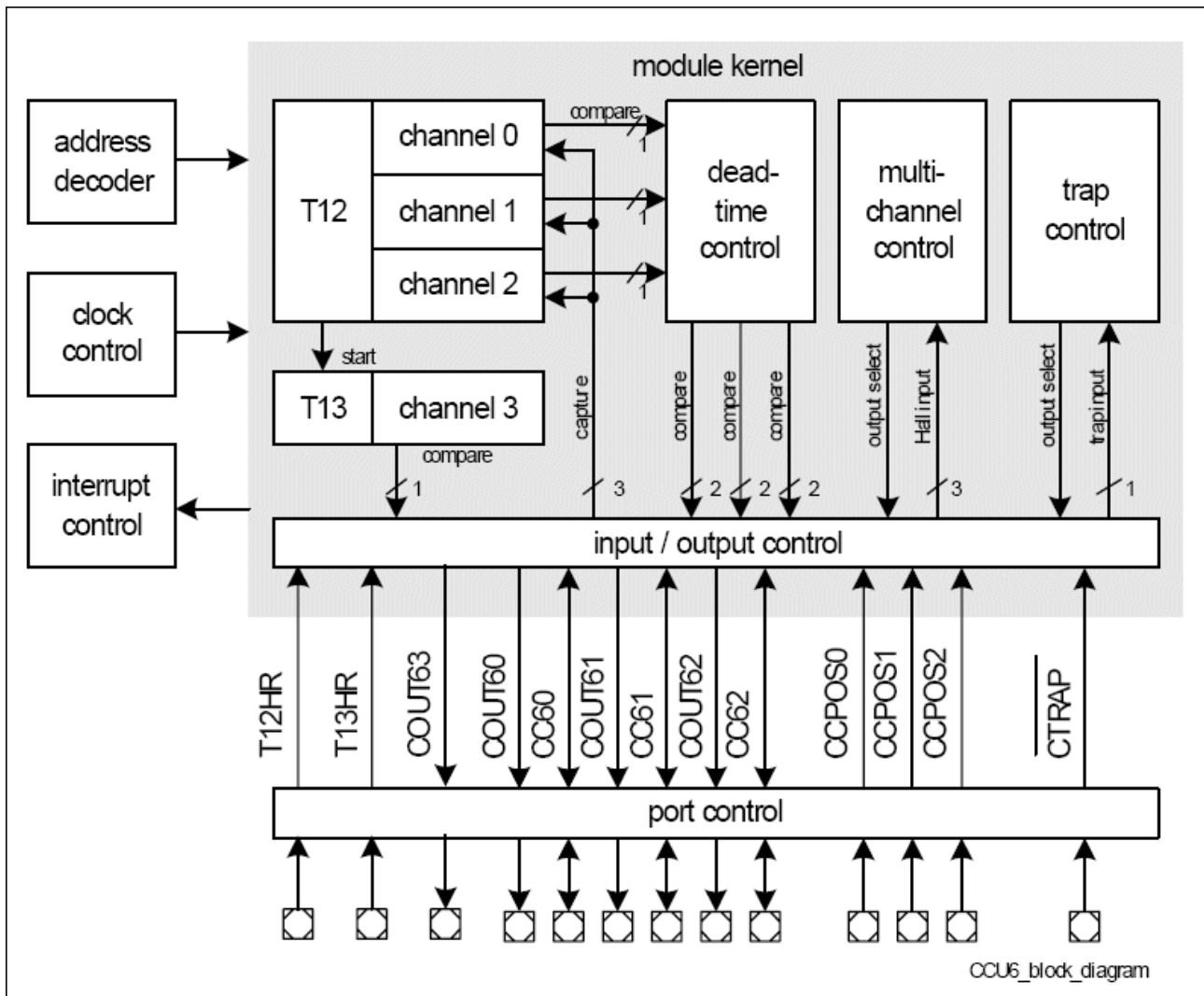
CCU6 Block Diagram – general use (Source: Product Marketing)



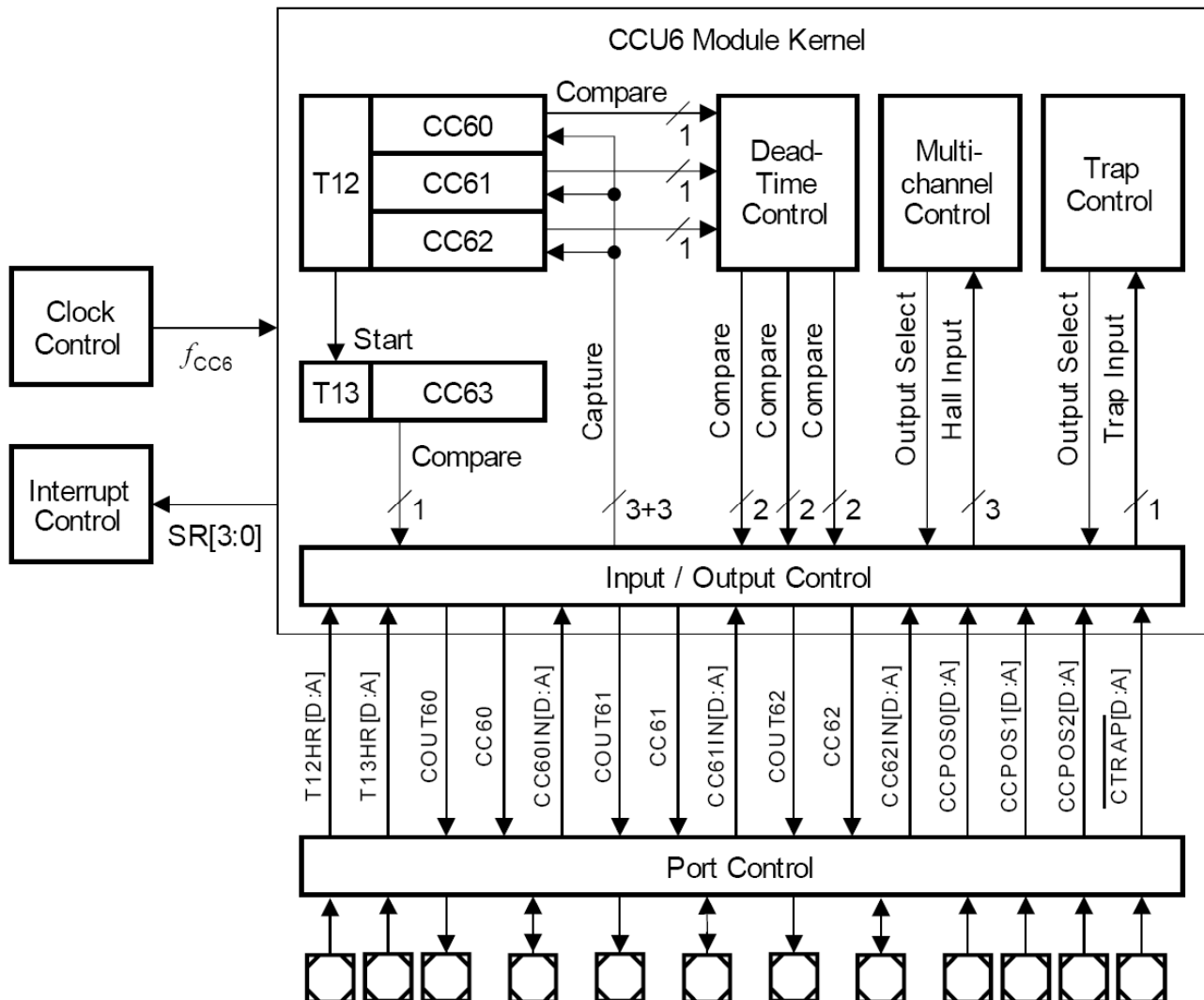
CCU6 Block Diagram – BLDC use (Source: Product Marketing)



CCU6 Block Diagram (Source: User's Manual)



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

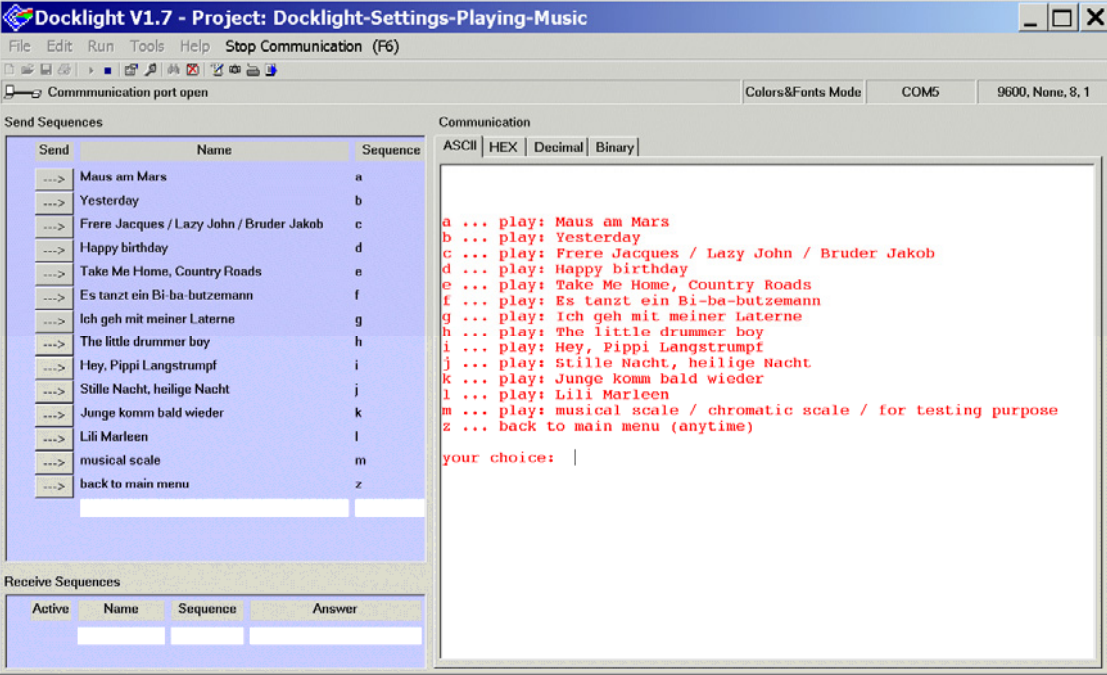


**Note:**

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

## “Cookery book“

For your first programming example for the CCU6:

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



**Appendix:**

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

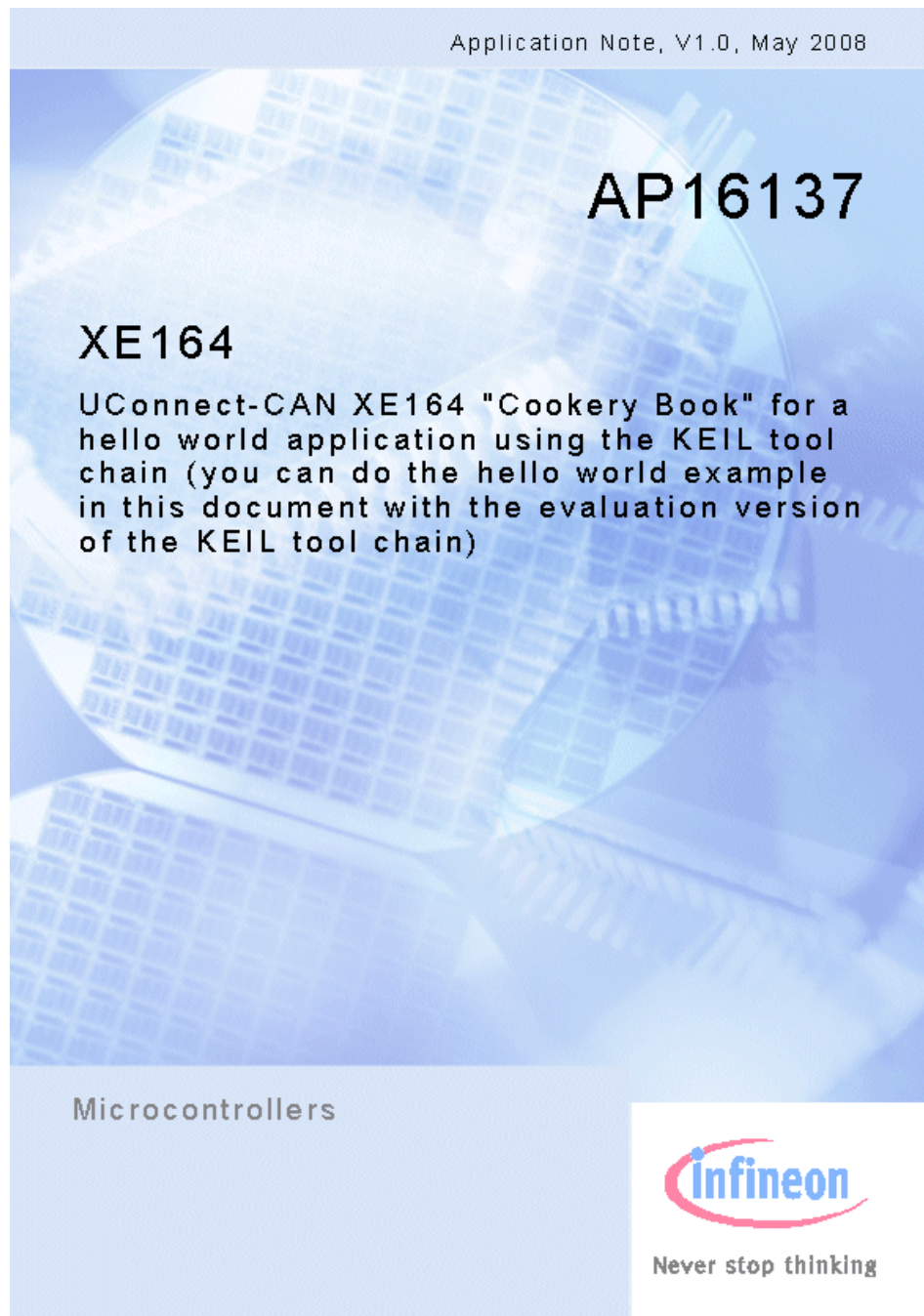
**Feedback:**

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

Do the UConnect-CAN XE164 Cookery Book:

**Note:**

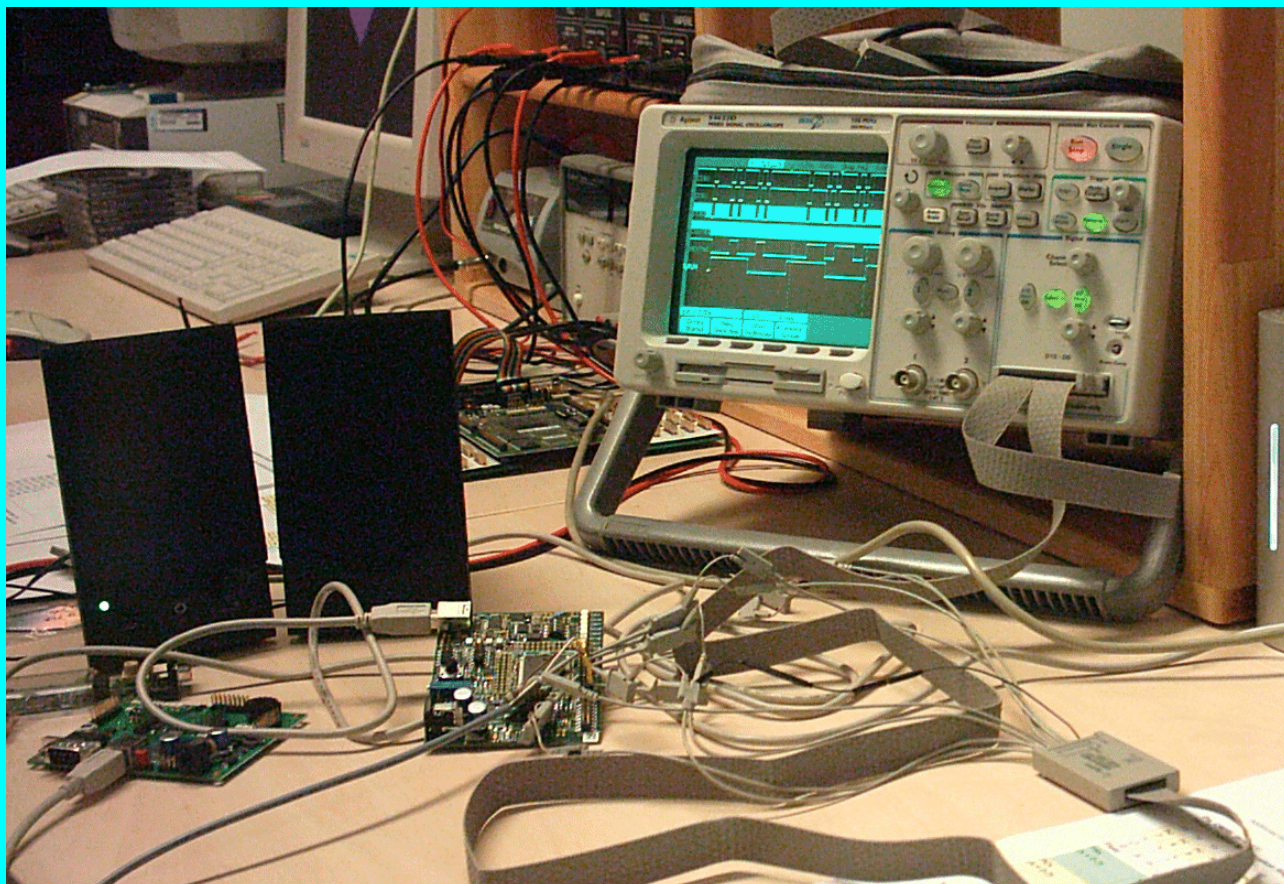
It is necessary to follow all instructions in the UConnect-CAN XE164 "Cookery Book" (AP16137) step by step, as this is the basis for all instructions which will follow later.



**Note:**

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

## 1.) Let's Get Started:



## Configuring and Reconfiguring the DAvE Project Settings:





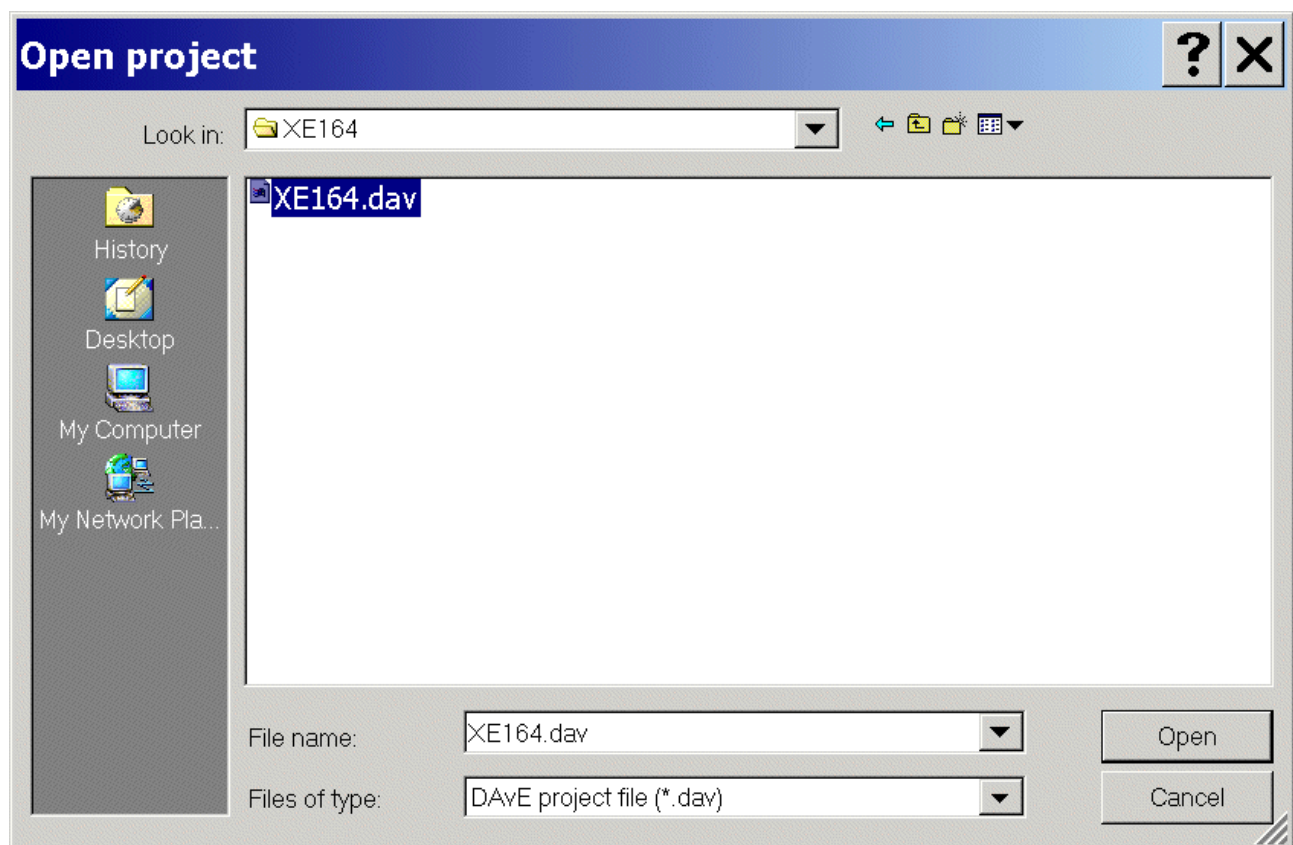
Start the program generator DAvE and open your [XE164.dav](#) DAvE project:

File

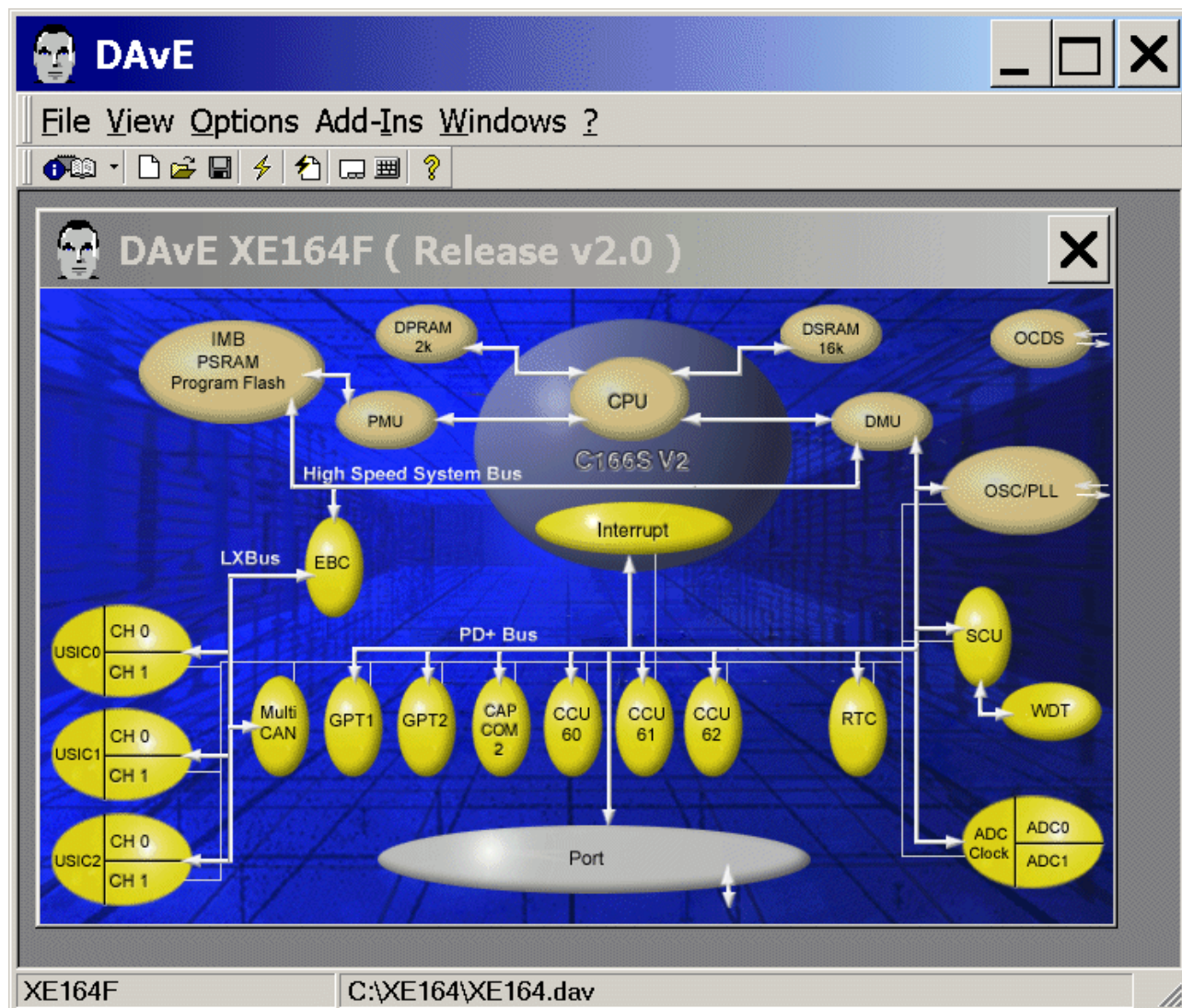
Open

Location: C:\XE164

Filename: [XE164.dav](#)



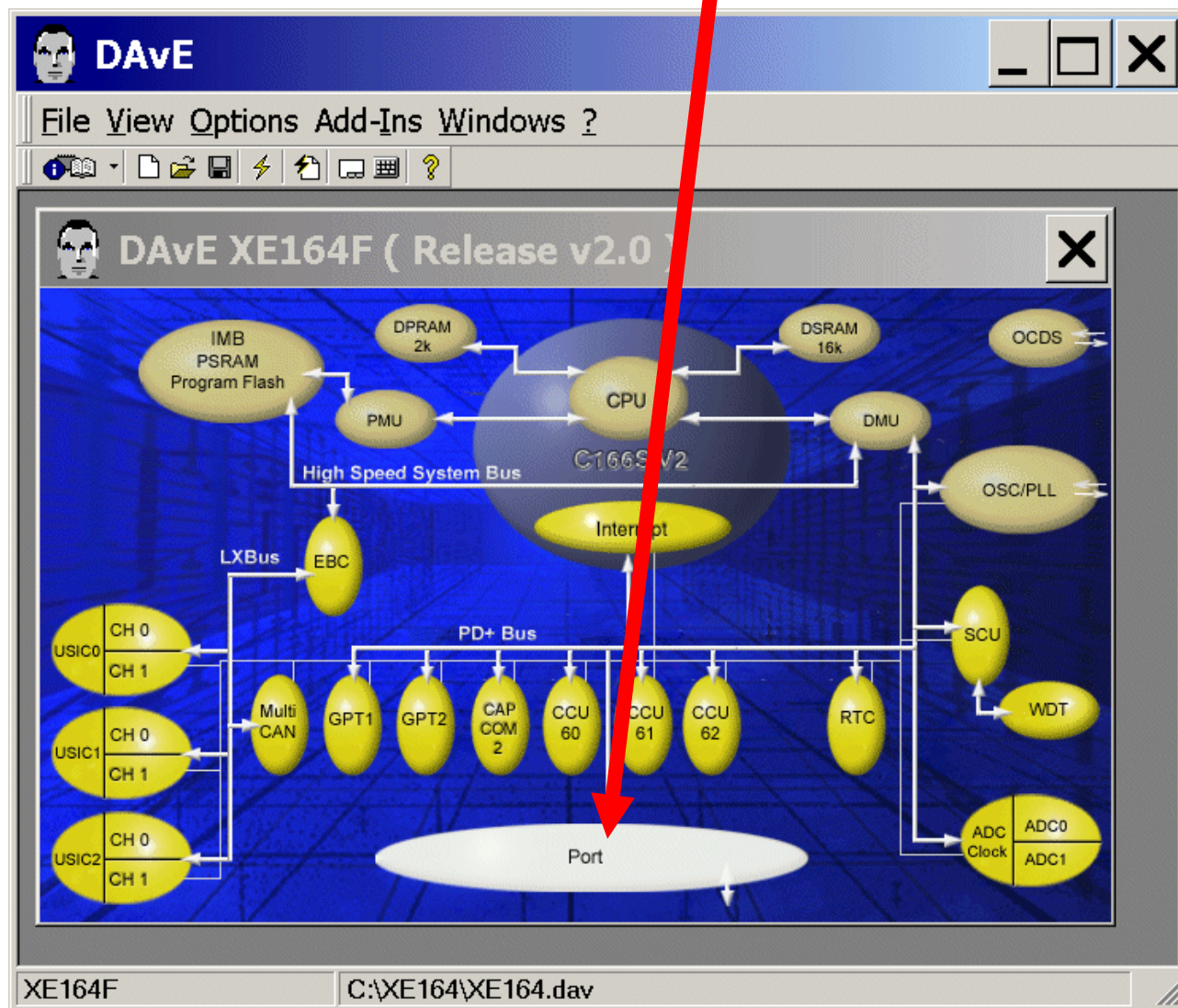
Click Open



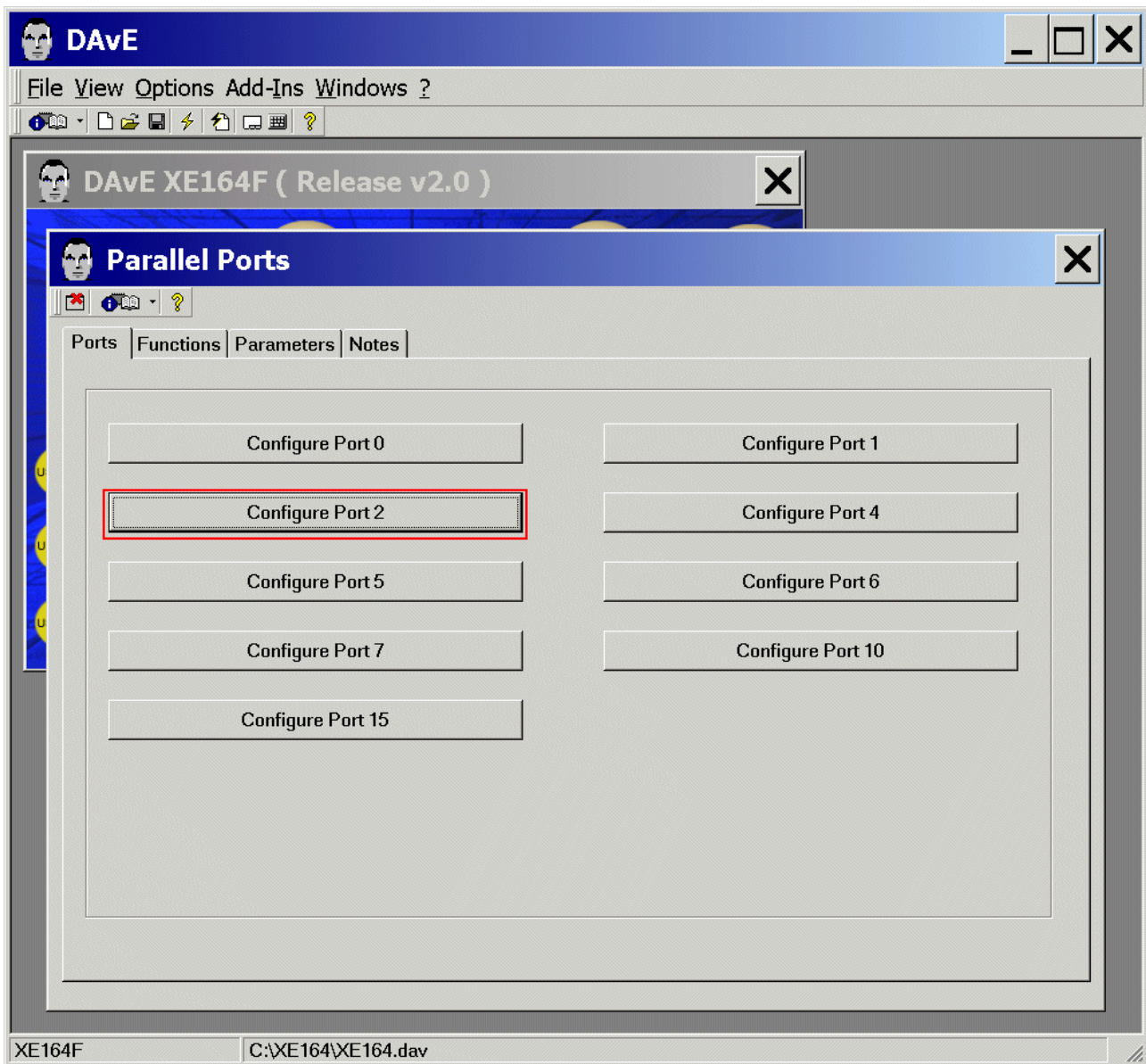


## Reconfiguration of Port 2:

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

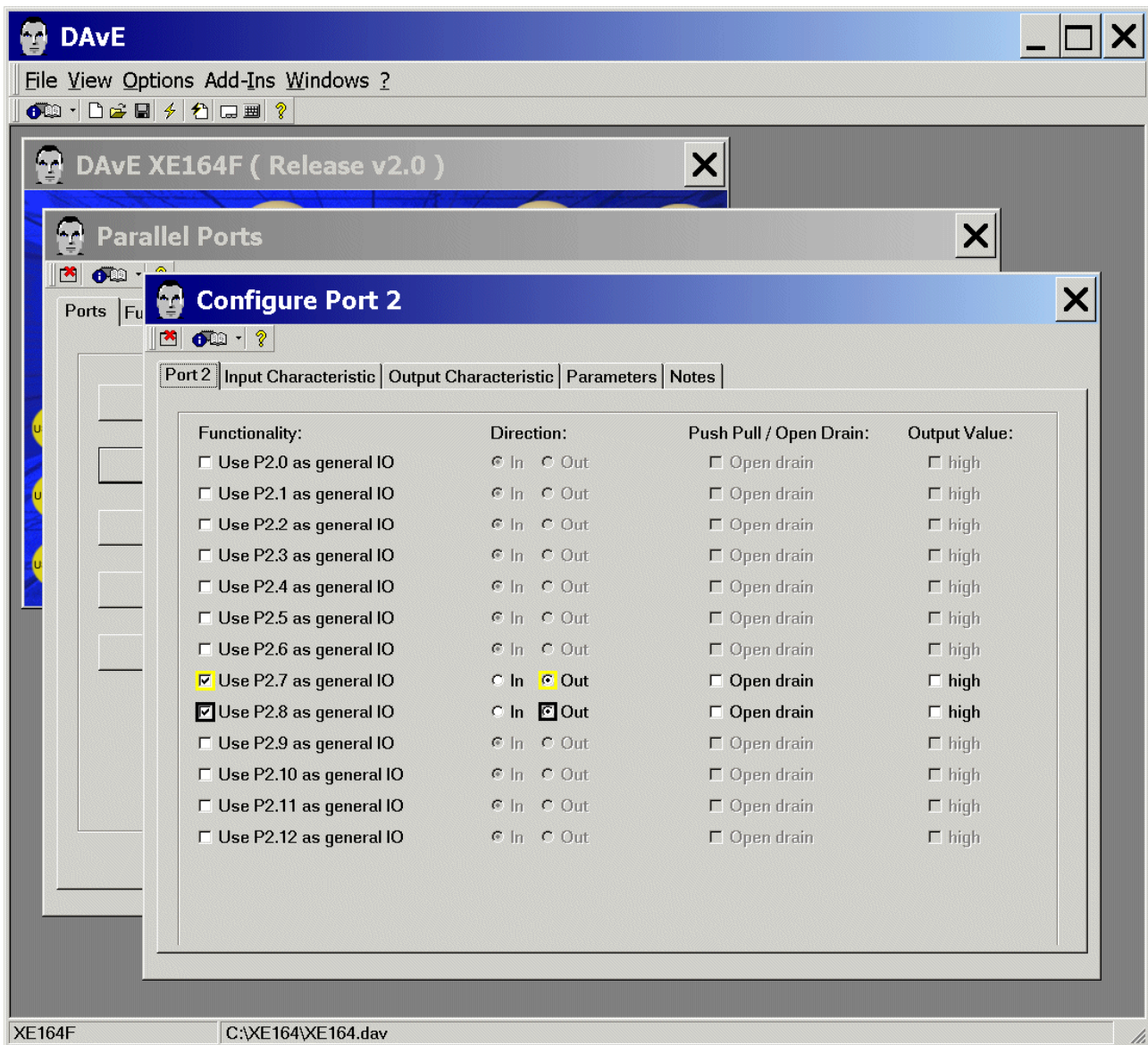


Ports: click “Configure Port 2”





Port 2: Functionality: **click** ☒ Use P2.8 as general IO - Direction: **click** ☒ Out



**Remember:**

Port pins used:

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




**Input Characteristic:** (do nothing)

**Output Characteristic:** (do nothing)

**Parameters:** (do nothing)


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

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

**Functions:** (do nothing)

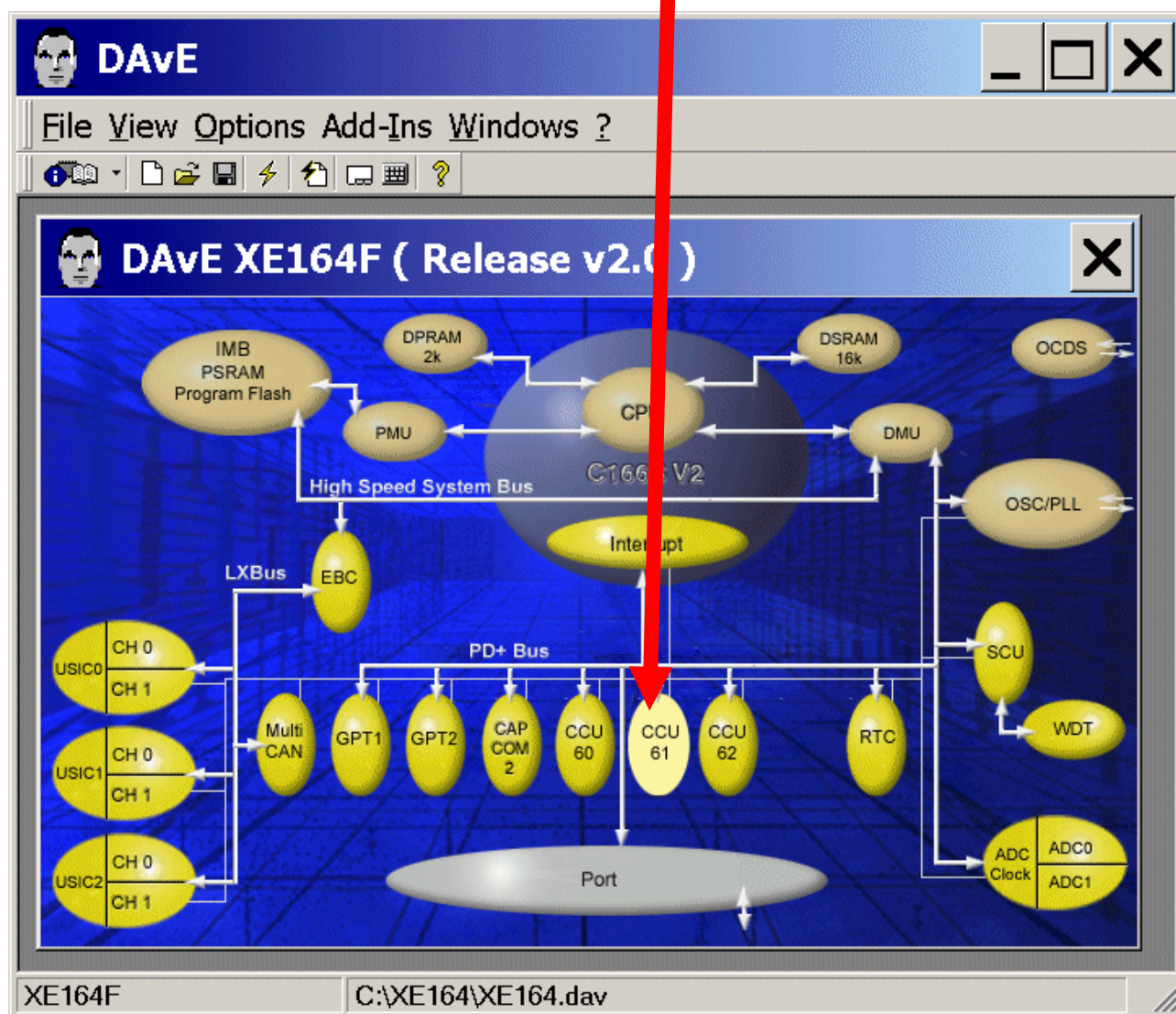
**Parameters:** (do nothing)

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

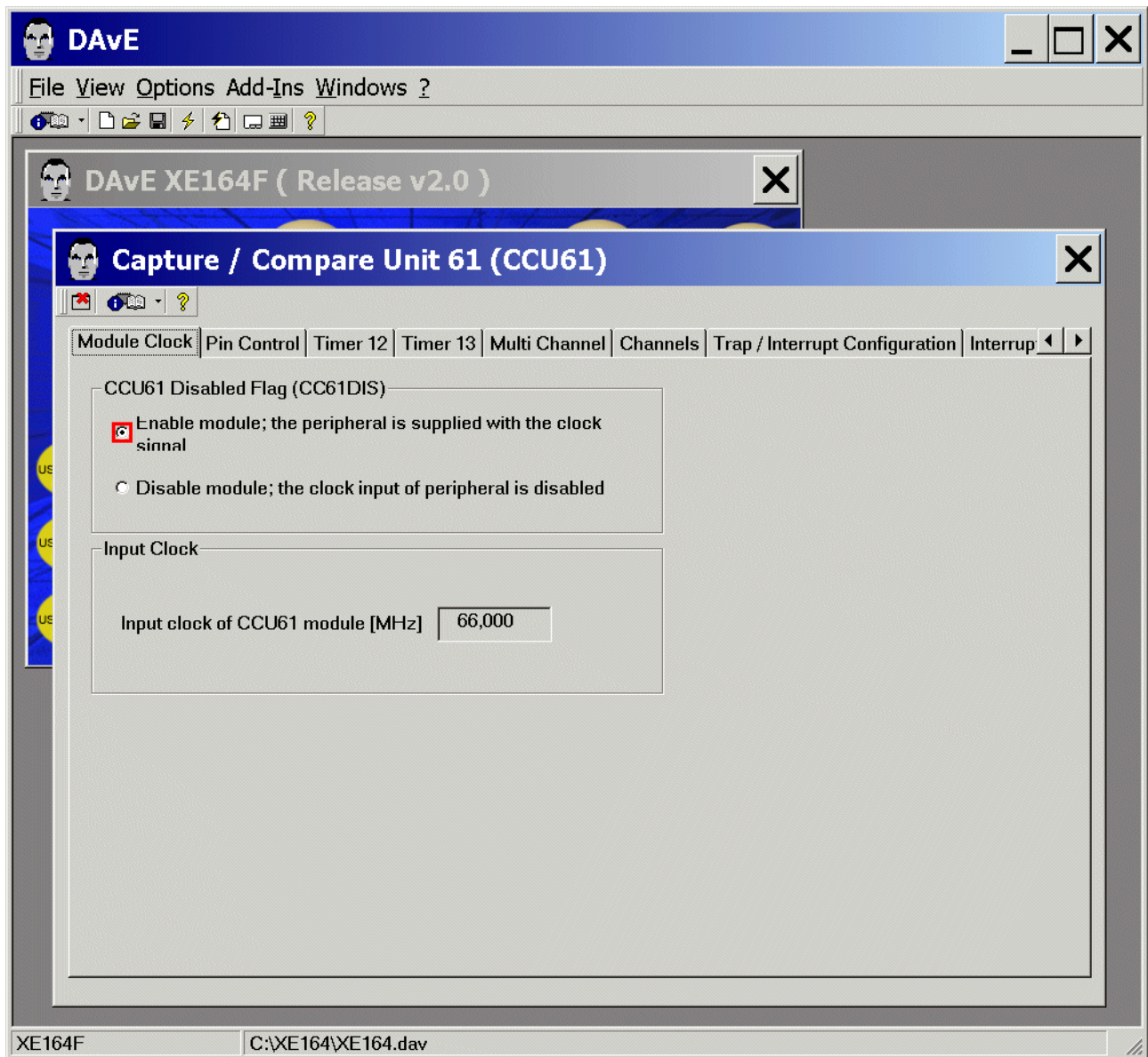
**Exit** and **Save** this dialog now by clicking  the close button.

Configuration of the CCU61 module:

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



CCU61: Module Clock: **click** ☒ Enable module





CCU61:

Pin Control: Control of Pins CC6x and CC6xIN: CC60: select Use pin CC60 as Output (P0.0)

CCU61:

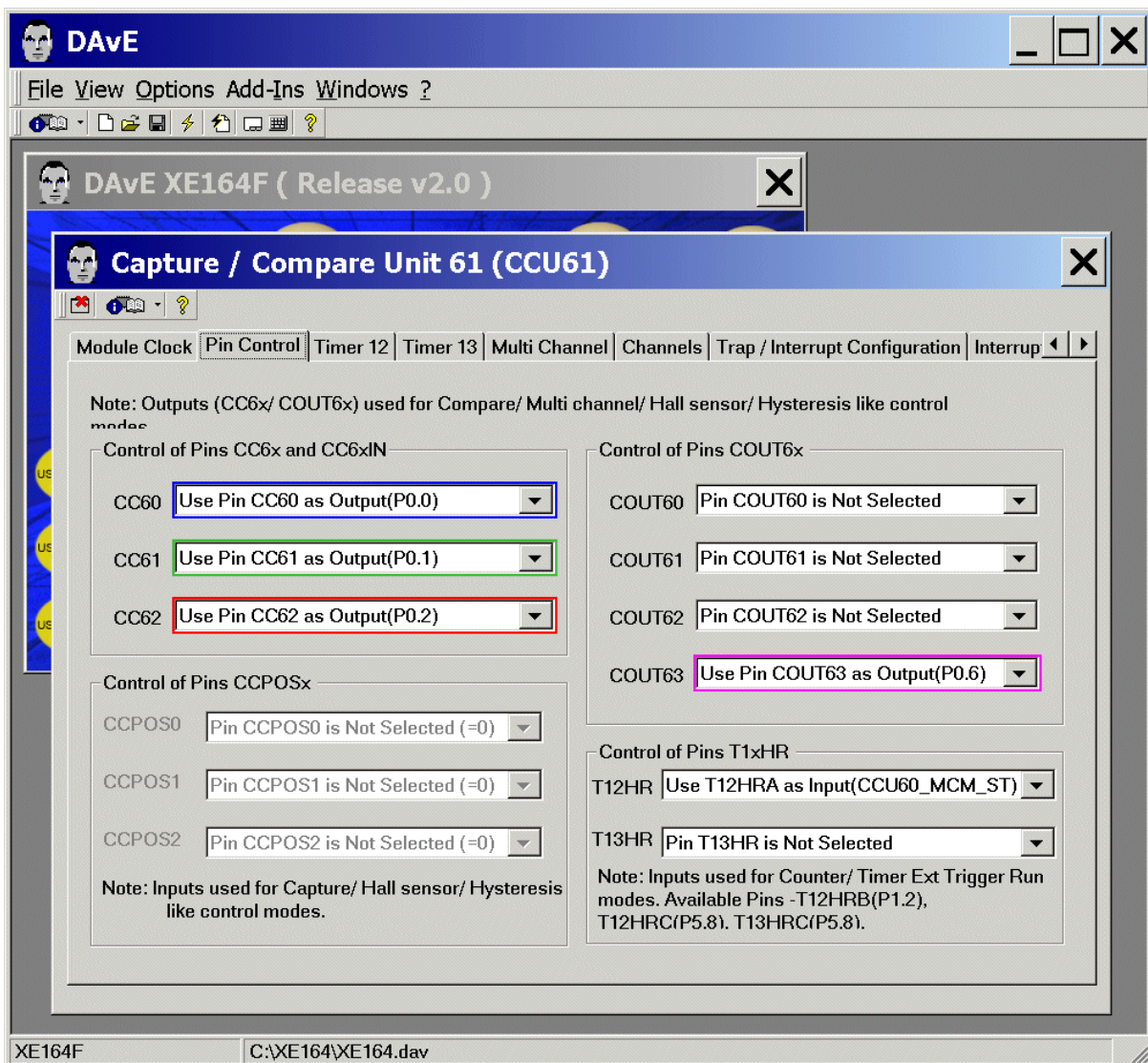
Pin Control: Control of Pins CC6x and CC6xIN: CC61: select Use pin CC61 as Output (P0.1)

CCU61:

Pin Control: Control of Pins CC6x and CC6xIN: CC62: select Use pin CC62 as Output (P0.2)

CCU61:

Pin Control: Control of Pins COUT6x: COUT63: select Use pin COUT63 as Output (P0.6)



**Remember:**

Port\_0 pins used by our PWM module CCU61:

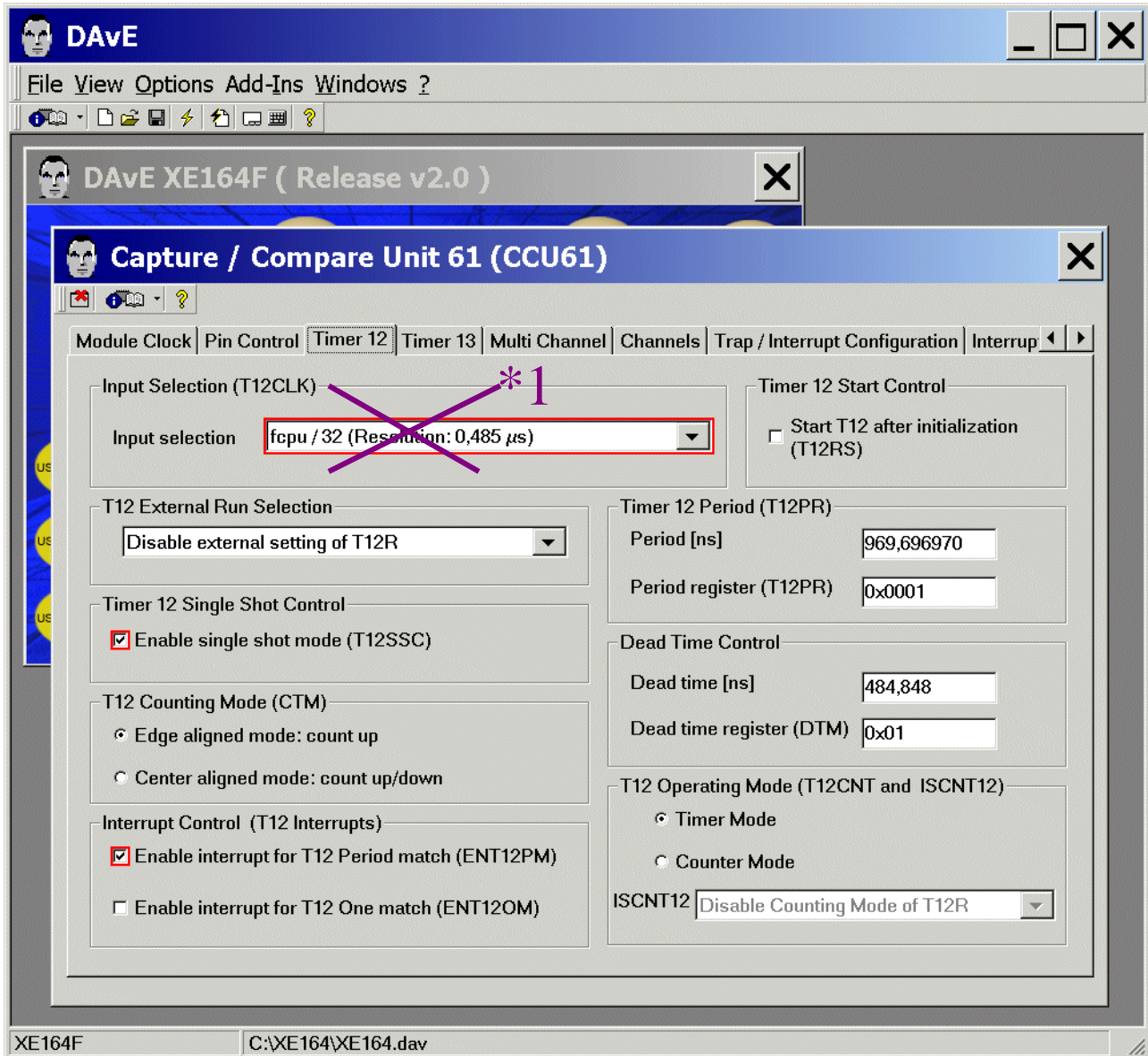
Port Lines	Signal	Duty Cycle [%] (purpose, modulated by)
P0.0	CCU61_CC60	100 (note length, Timer_12)
P0.1	CCU61_CC61	100 (note length, Timer_12)
P0.2	CCU61_CC62	100 (note length, Timer_12) + 50 (note frequency, Timer_13)
P0.6	CCU61_COUT63	50 (note frequency, Timer_13)

Timer 12: “note length”:

CCU61: Timer 12: Input Selection: Input selection: choose  $f_{CPU}/32 \rightarrow$  Resolution = 124,12  $\mu s$  \*1

CCU61: Timer 12: T12 Single Shot Control: click ☒ Enable single shot mode (T12SSC)

CCU61: Timer 12: Interrupt Control: click ☒ Enable interrupt for T12 Period match



\*1:

Timer 12 Resolution:

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



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



**Note:**

Unfortunately bit T12PRE is not available in the DAVe dialog.

Source: User's Manual:

The input clock for timer T12 can be from  $f_{CCU61}$  to a maximum of  $f_{CCU61}/128$  and is configured by bit field T12CLK. In order to support higher clock frequencies, an additional prescaler factor of  $1/256$  can be enabled for the prescaler of T12 if bit T12PRE = 1.

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

File Edit View Document Tools Window Help

125%

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

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

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

290 of 731

**Timer 12 Resolution:**

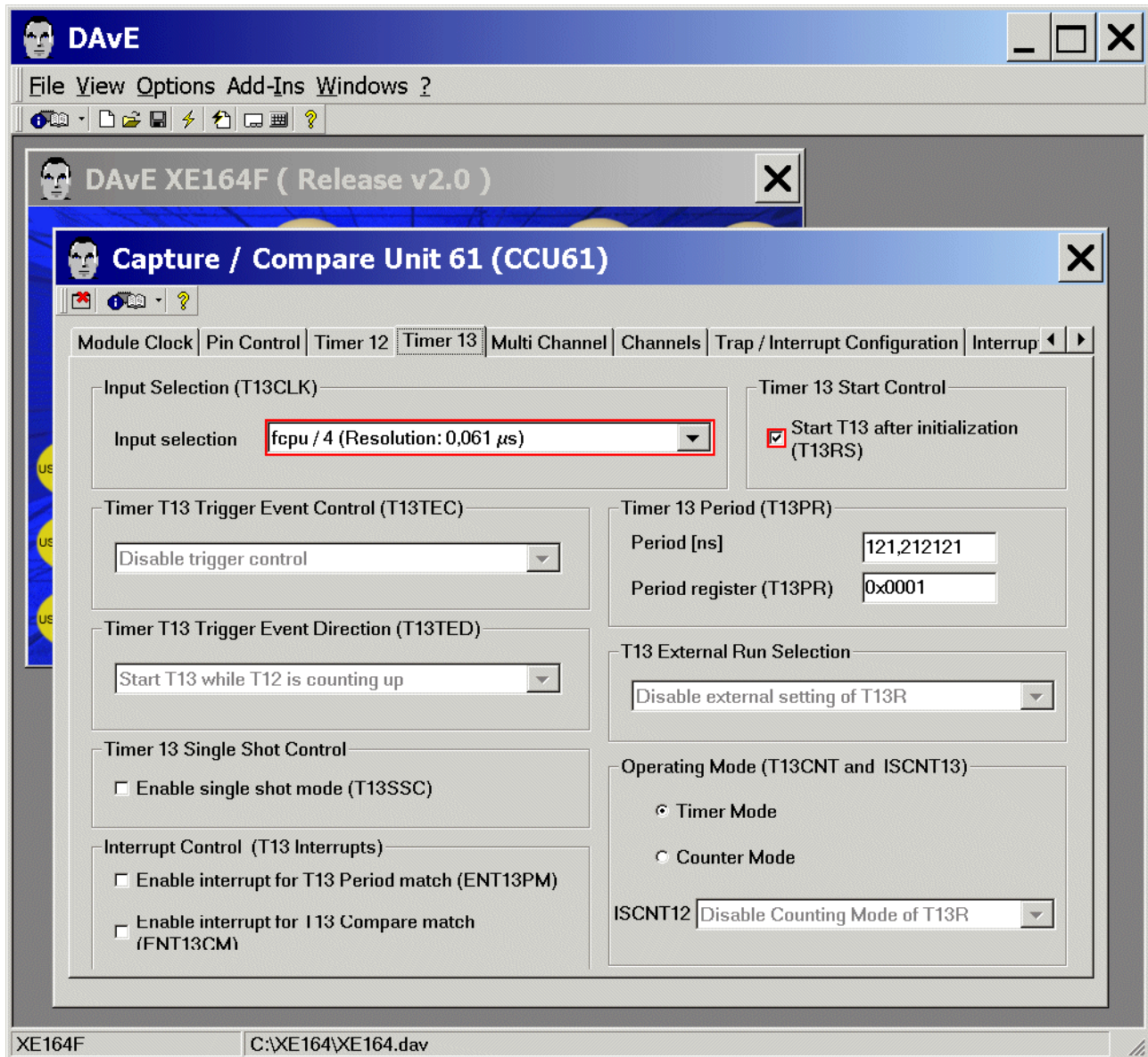
$66 \text{ MHz} / 256 \text{ (T12PRE=1, done by software)} / 32 = 8.056,64 \text{ Hz} \rightarrow \text{Resolution} = 124,12 \mu\text{s}$



Timer 13: "note frequency":

CCU61: Timer T13: Input Selection: Input selection select  $f_{CPU}/4$  (Resolution: 60,606 ns)

CCU61: Timer T13: Timer 13 Start Control: click ✓ Start T13 after initialization (T13RS)



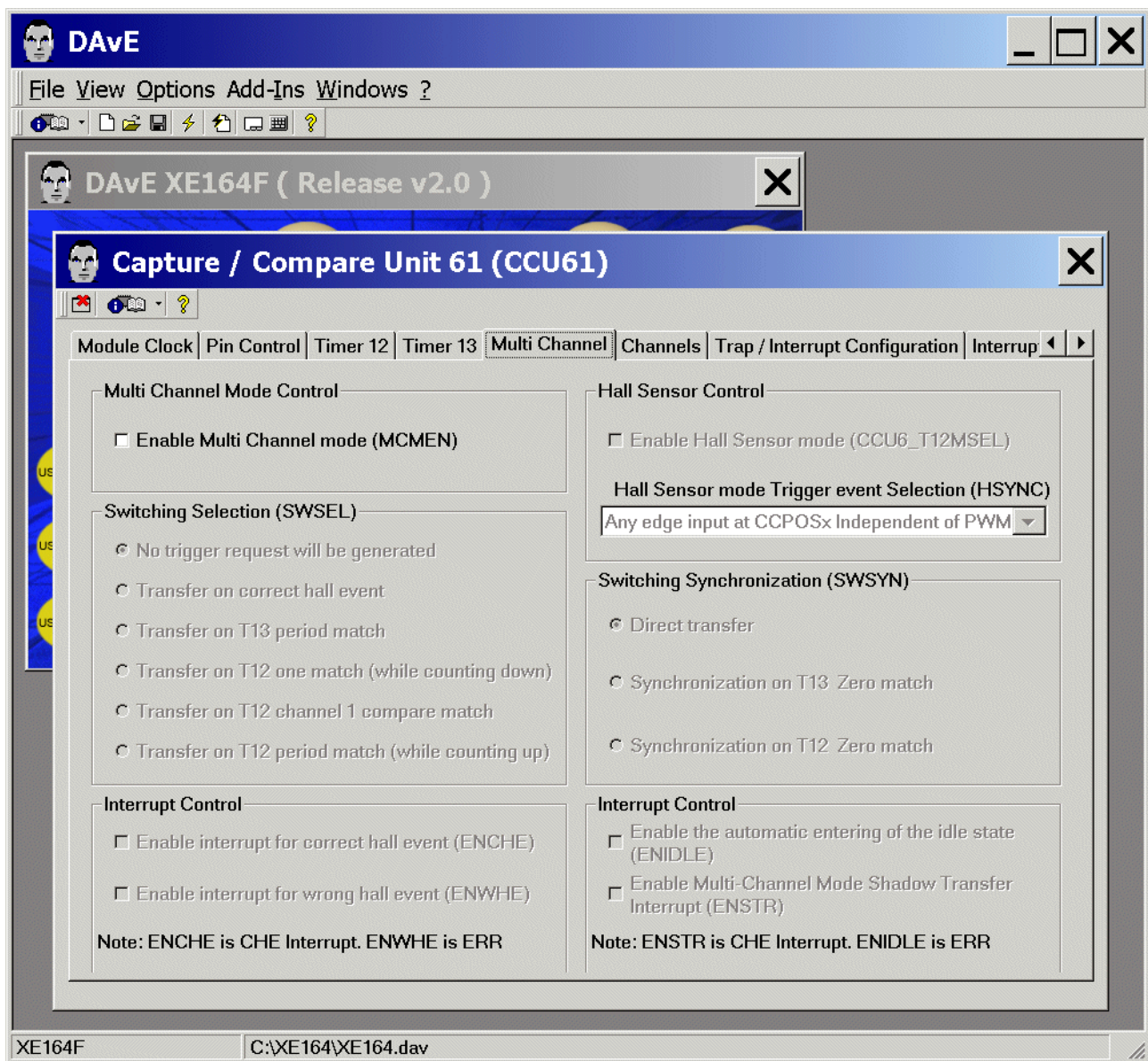
Note:

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



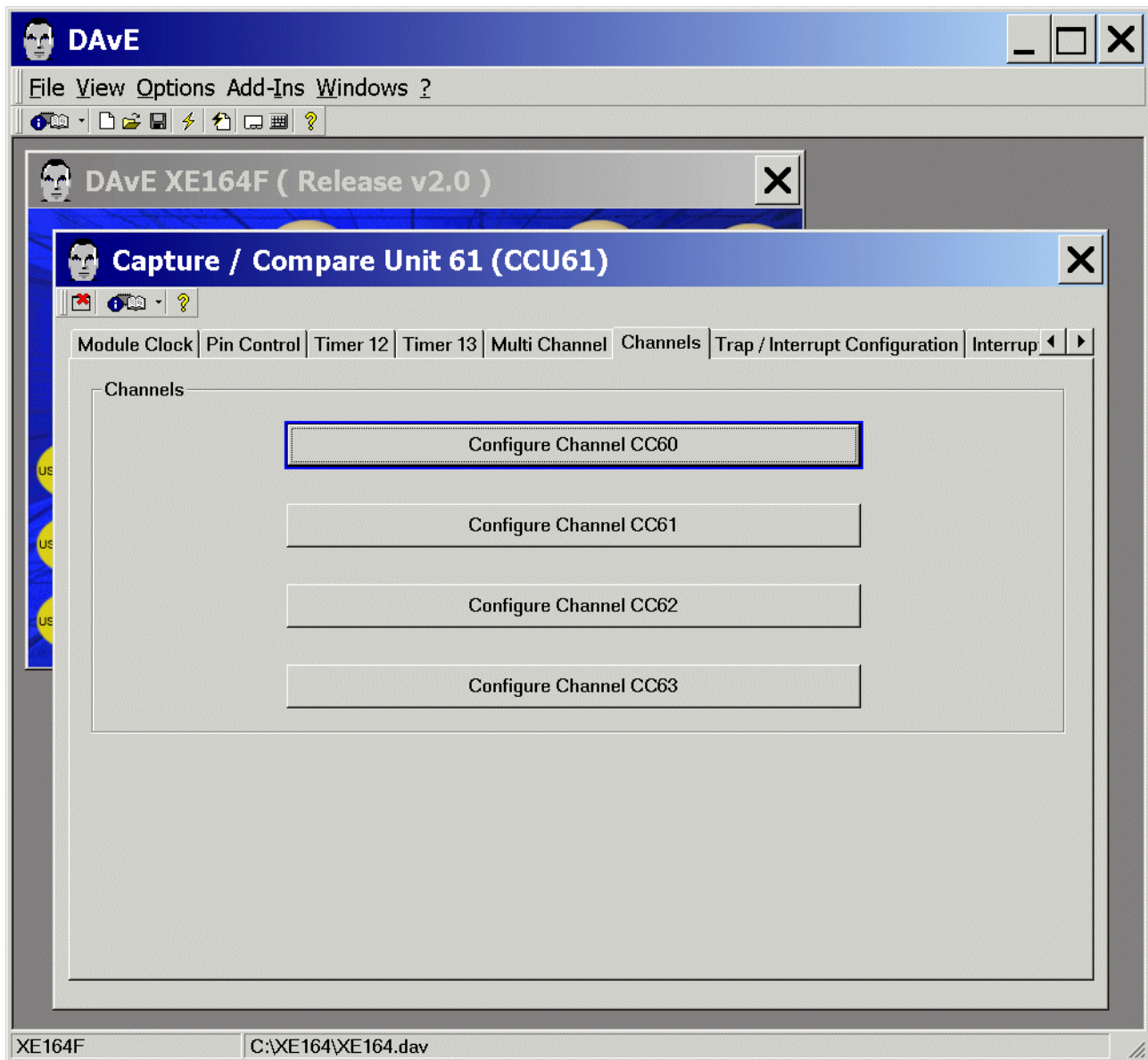
<<< !!! [click here to see more information about music](#) !!! >>>

CCU61: Multi Channel: (do nothing)



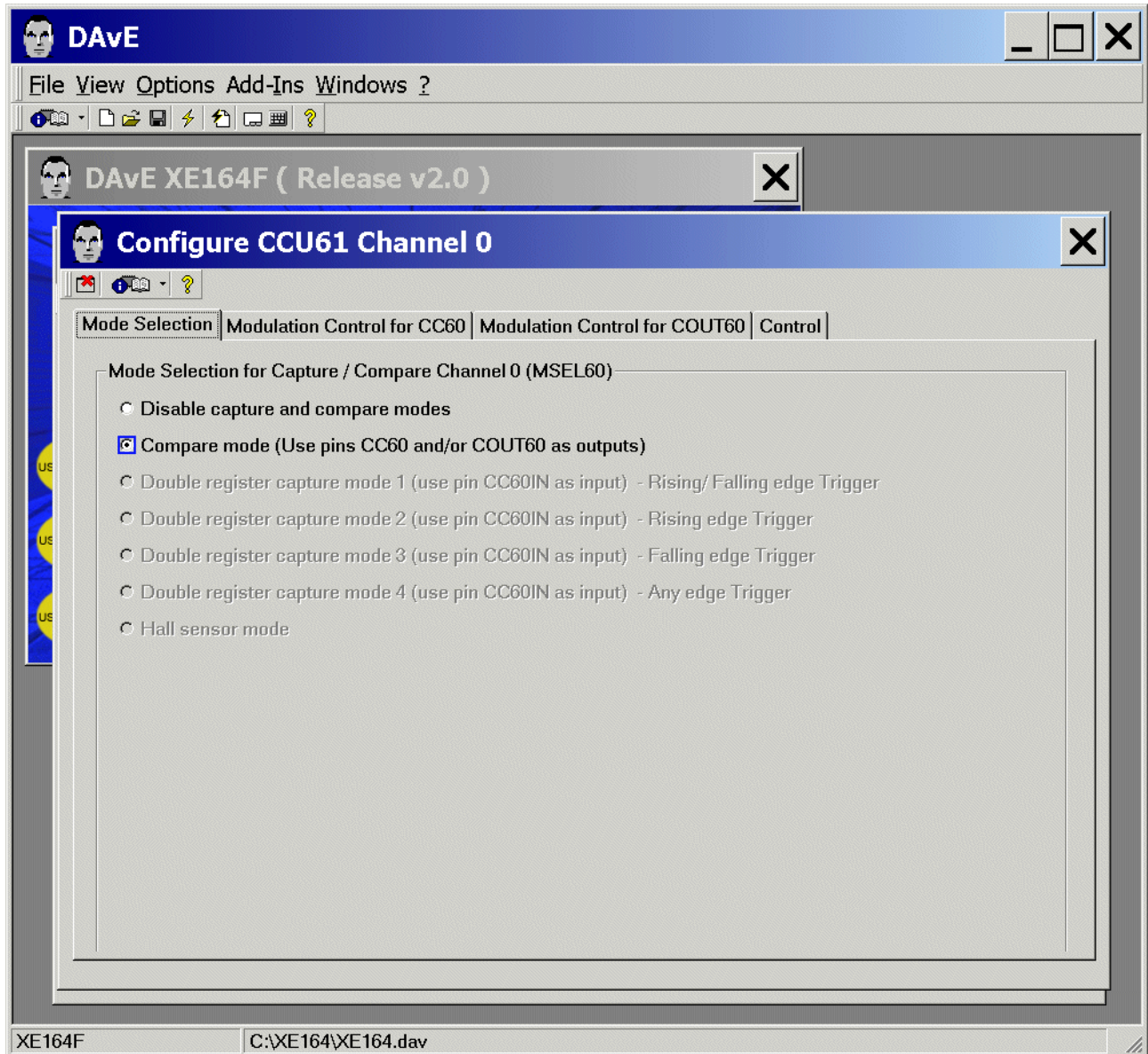


CCU61: Channels: **click** Configure Channel 0



CCU61: Channels: Configure Channel 0:

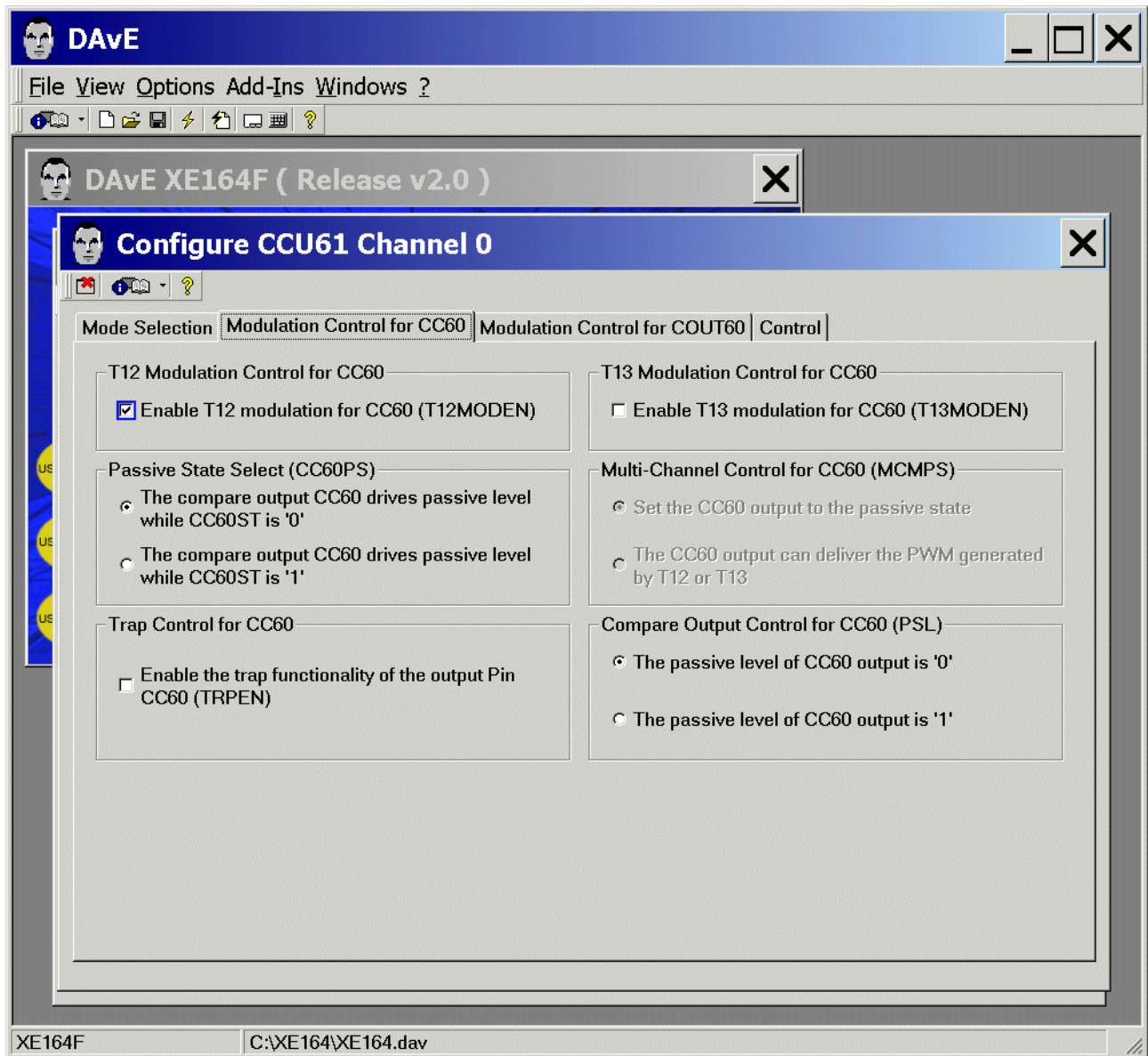
Mode Selection: Mode Selection for Capture / Compare Channel 0: click ☒ Compare mode



CCU61: Channels: Configure Channel 0:

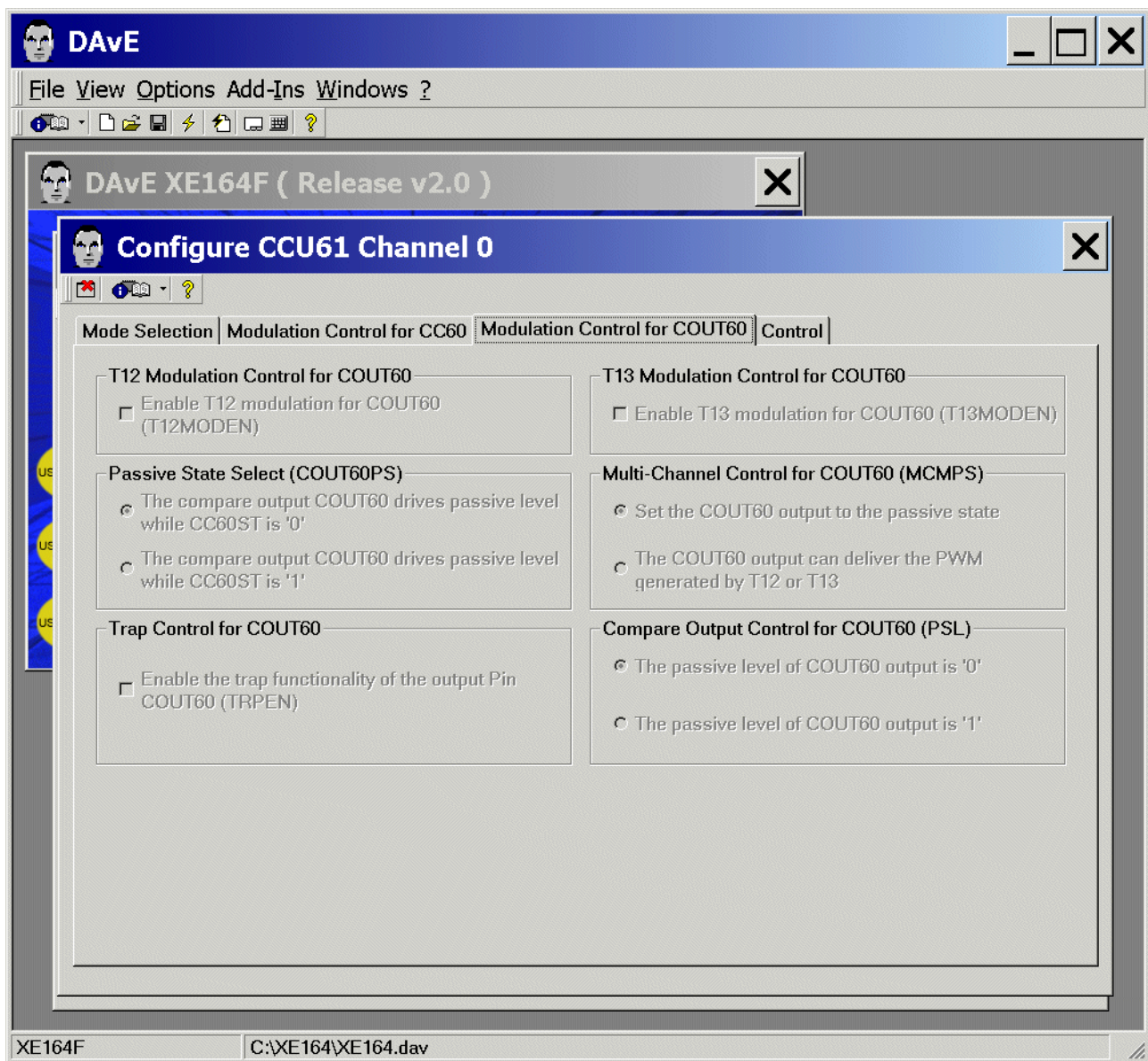
Modulation Control for CC60:

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

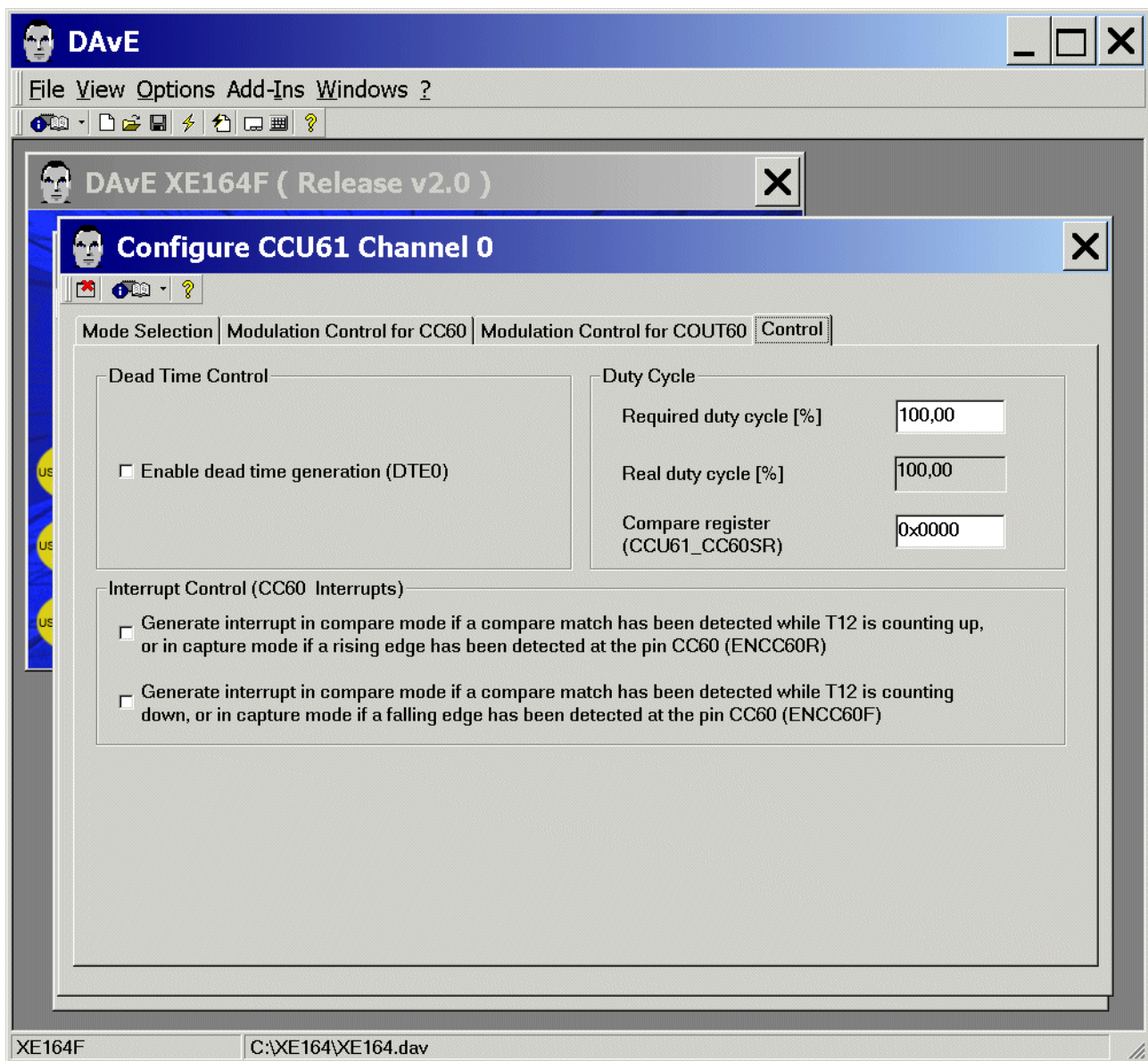





CCU61: Channels: Configure Channel 0: Modulation Control for COUT60: (do nothing)

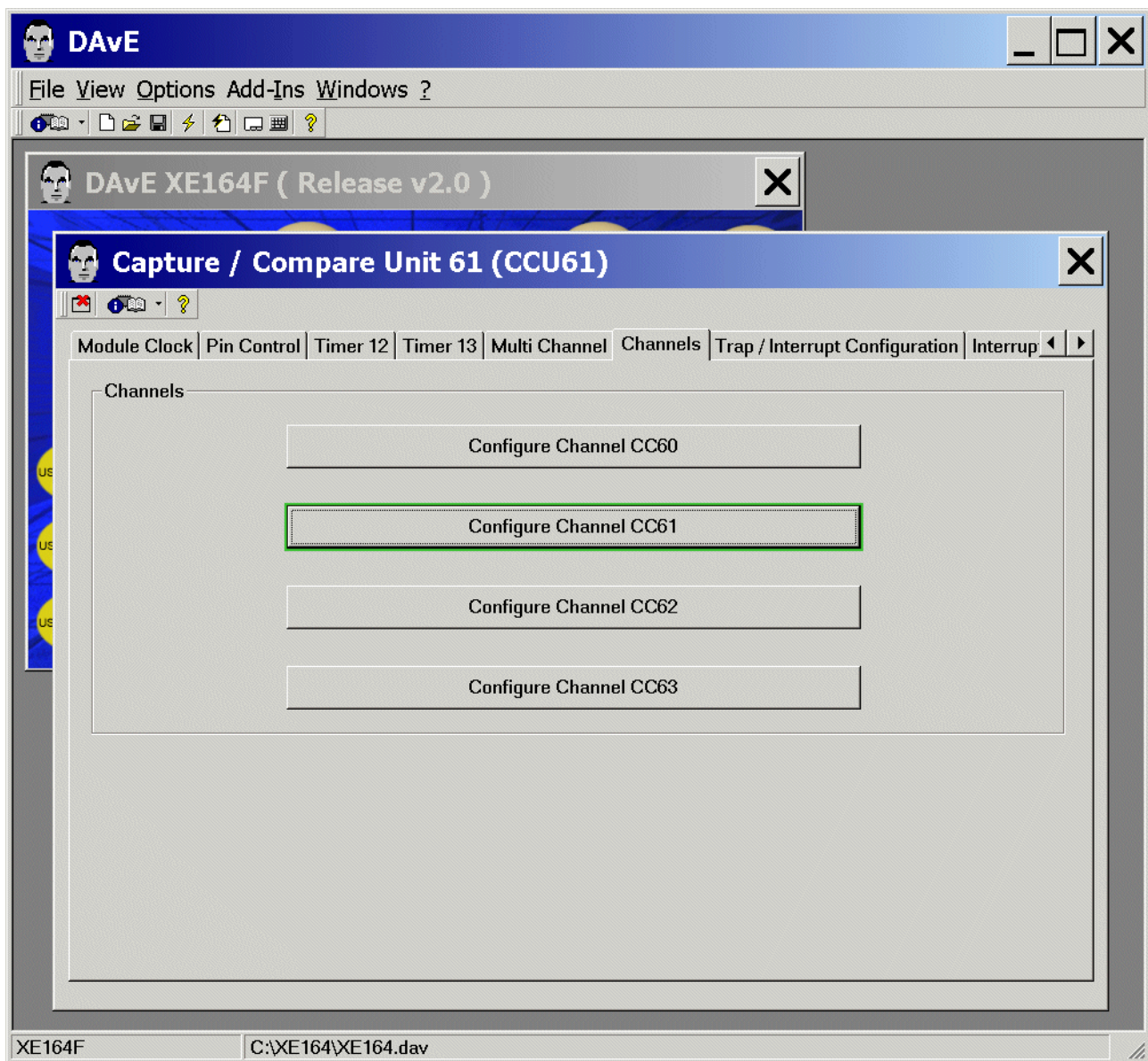


CCU61: Channels: Configure Channel 0: Control: (do nothing)



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

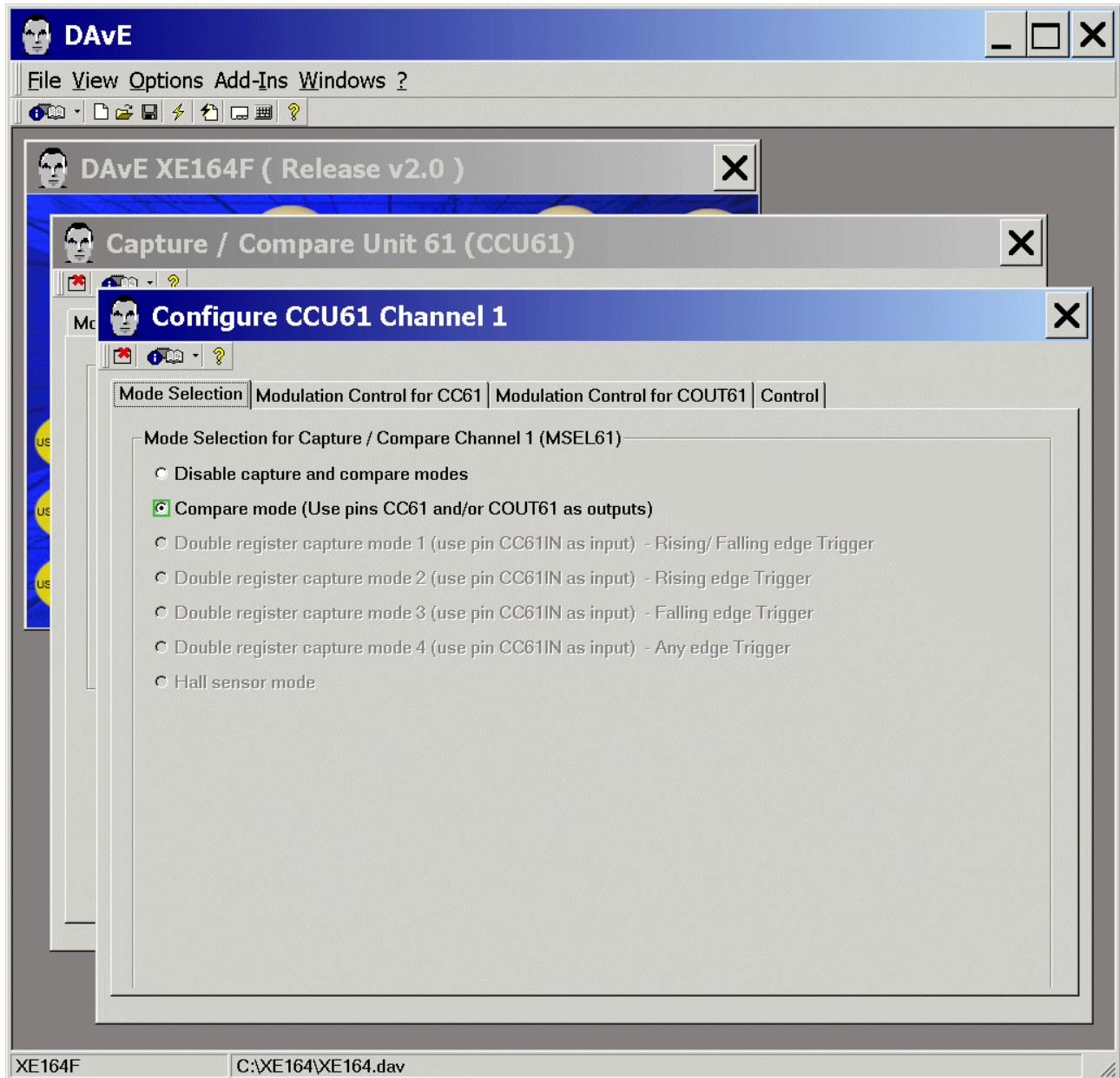
CCU61: Channels: **click** Configure Channel 1





CCU61: Channels: Configure Channel 1:

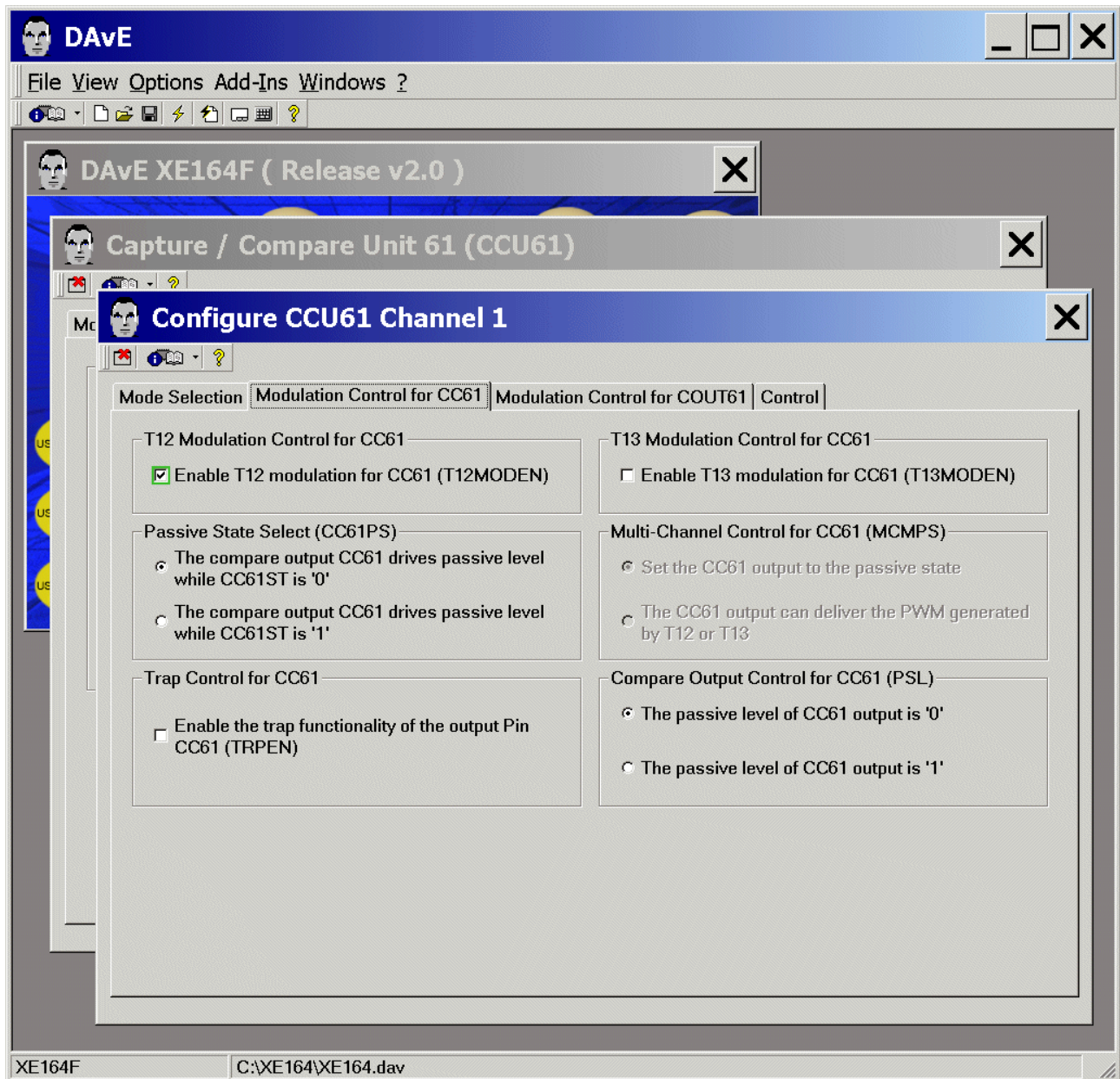
Mode Selection: Mode Selection for Capture / Compare Channel 1: click ☒ Compare mode



CCU61: Channels: Configure Channel 1:

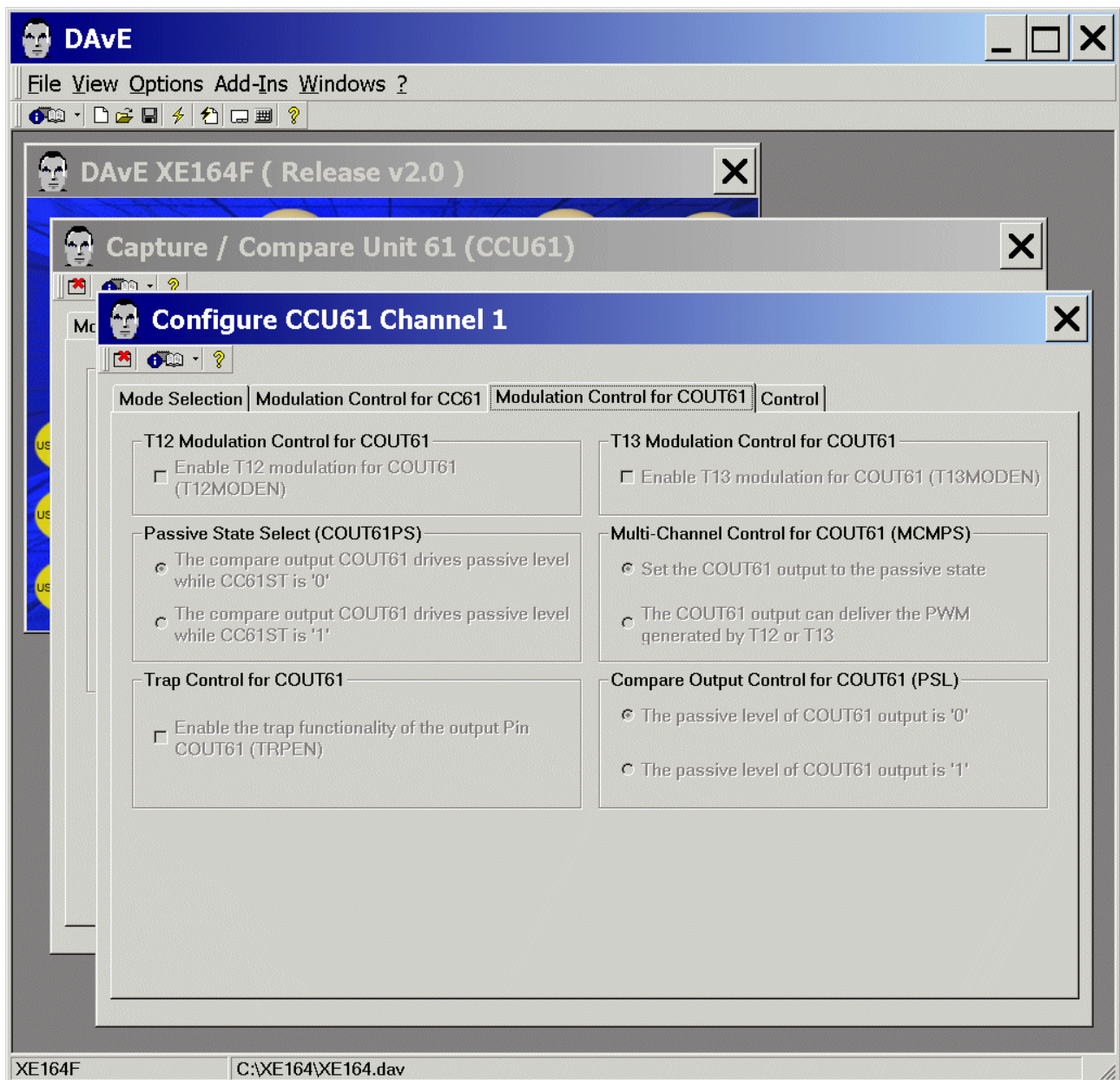
Modulation Control for CC61:

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

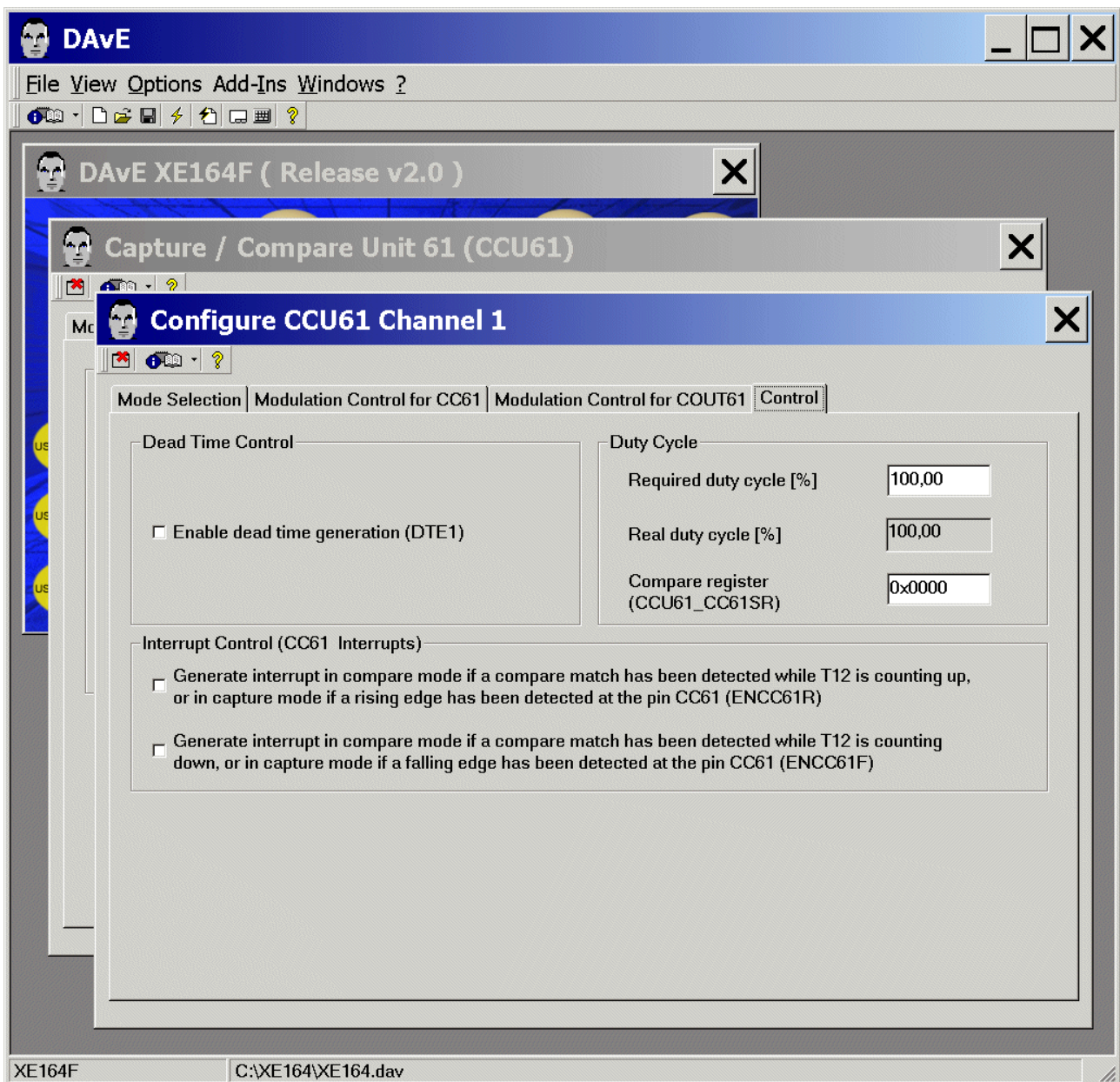





CCU61: Channels: Configure Channel 1: Modulation Control for COUT61: (do nothing)

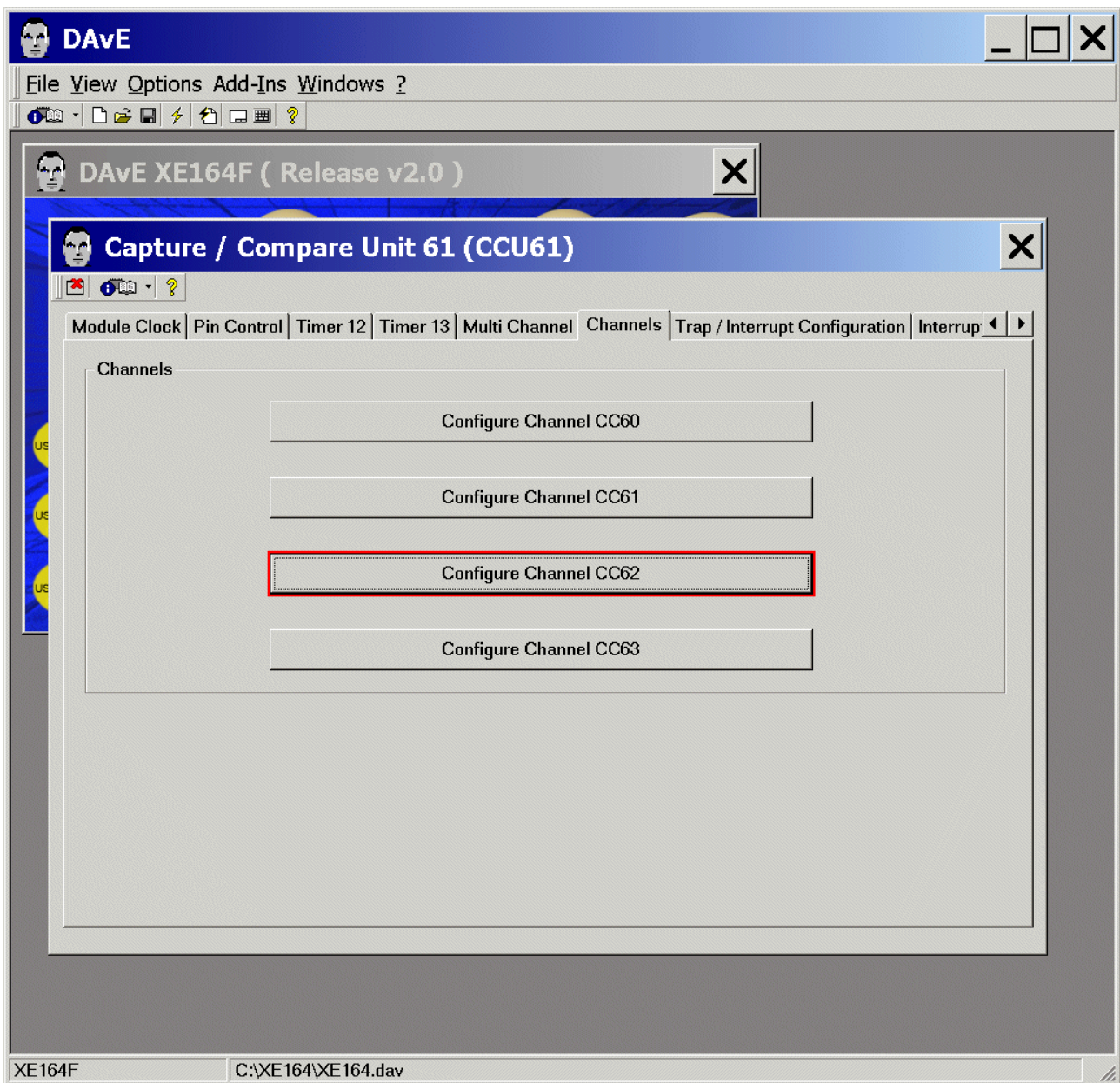


CCU61: Channels: Configure Channel 1: Control: (do nothing)



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

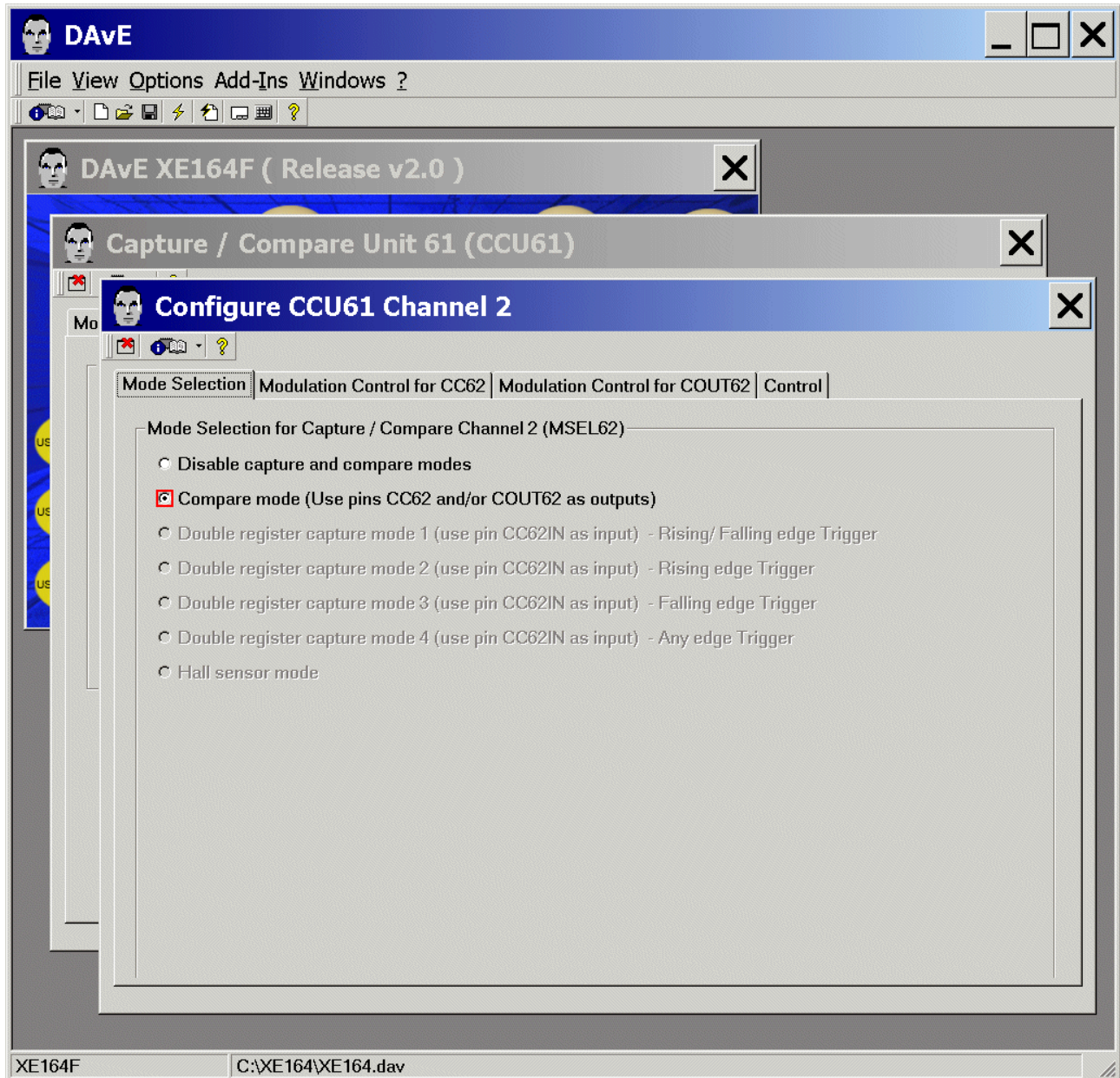
CCU61: Channels: **click** Configure Channel 2





CCU61: Channels: Configure Channel 2:

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



CCU61: Channels: Configure Channel 2:

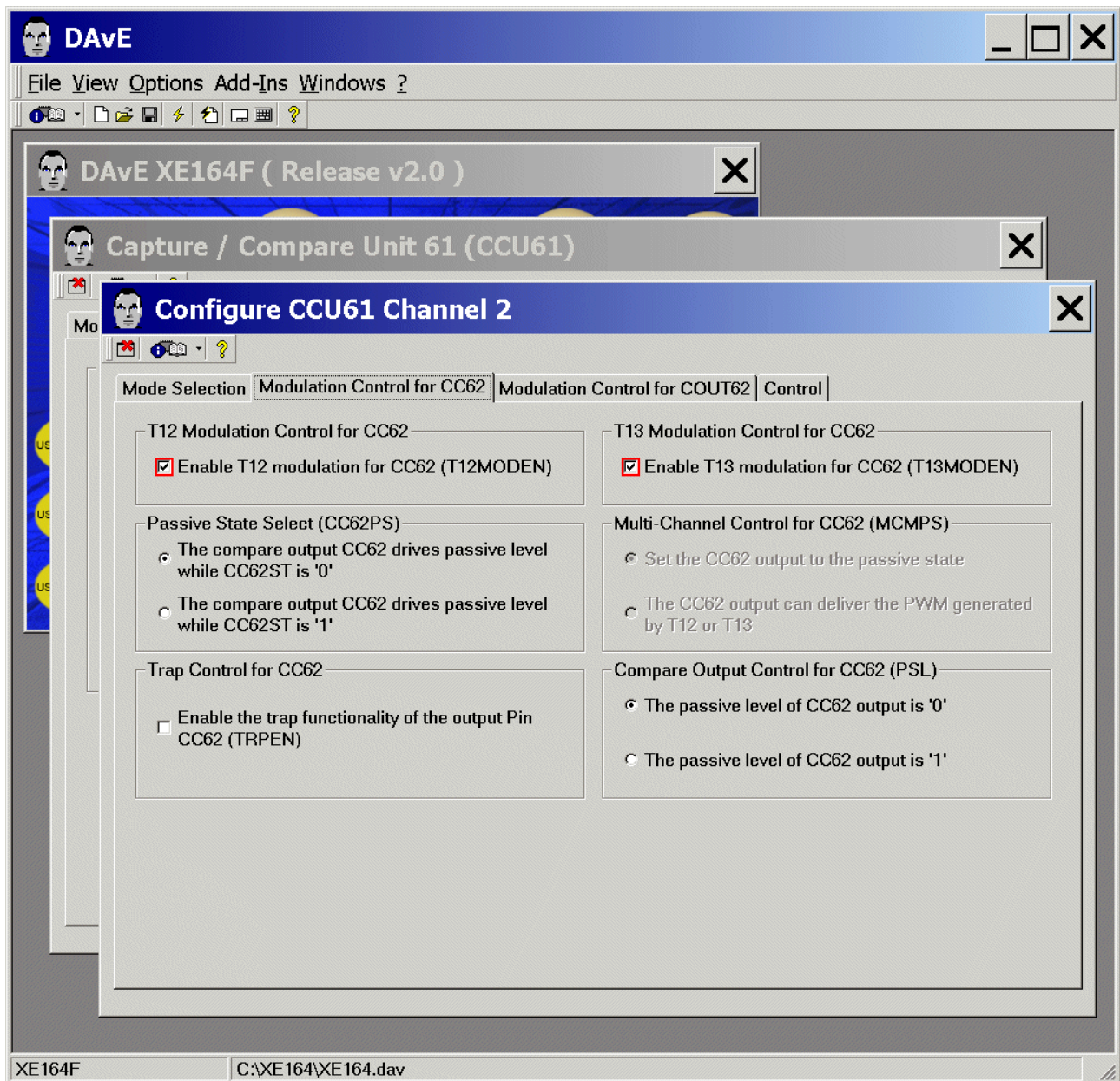
Modulation Control for CC62:

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

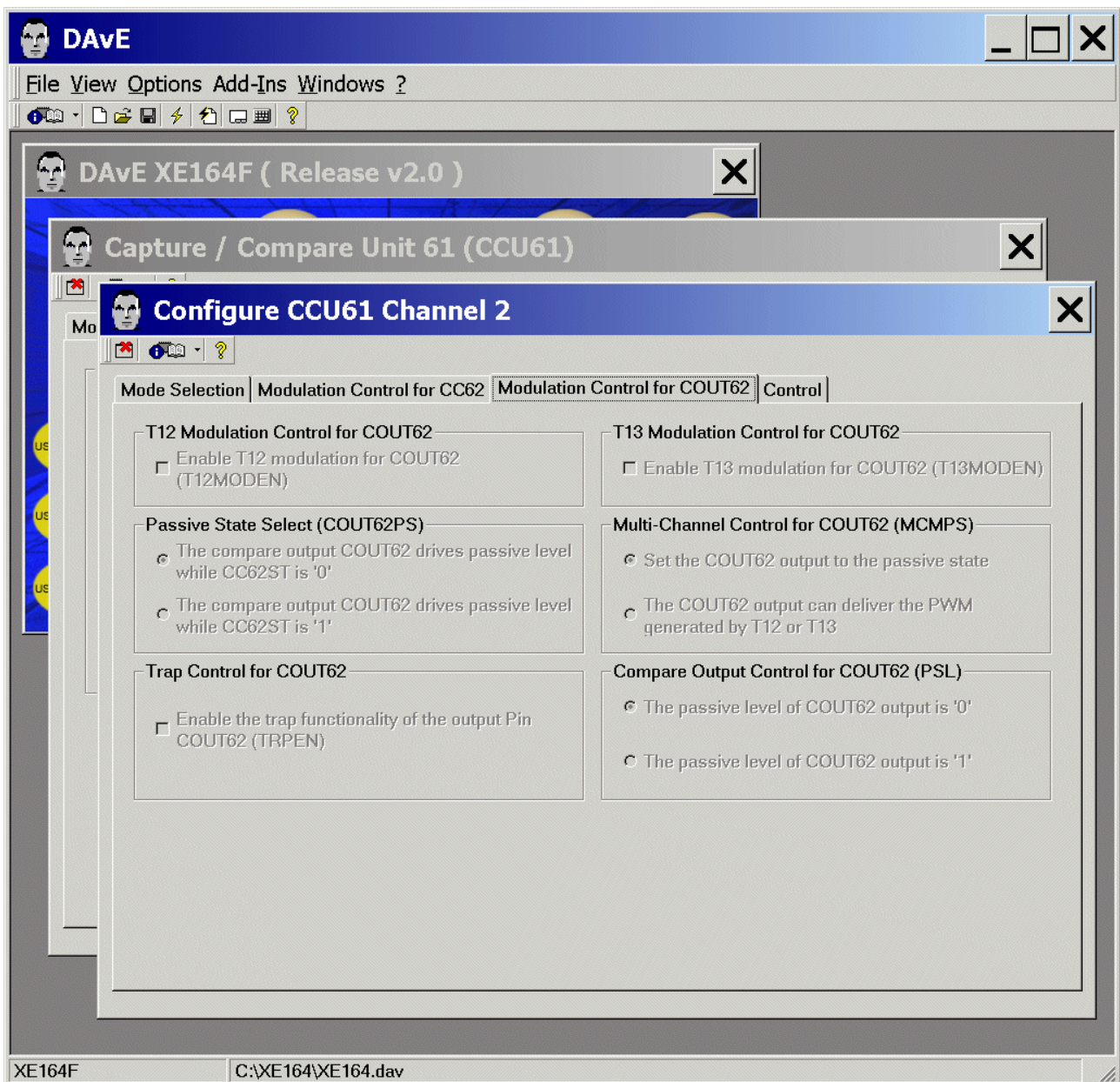
CCU61: Channels: Configure Channel 2:

Modulation Control for CC62:

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

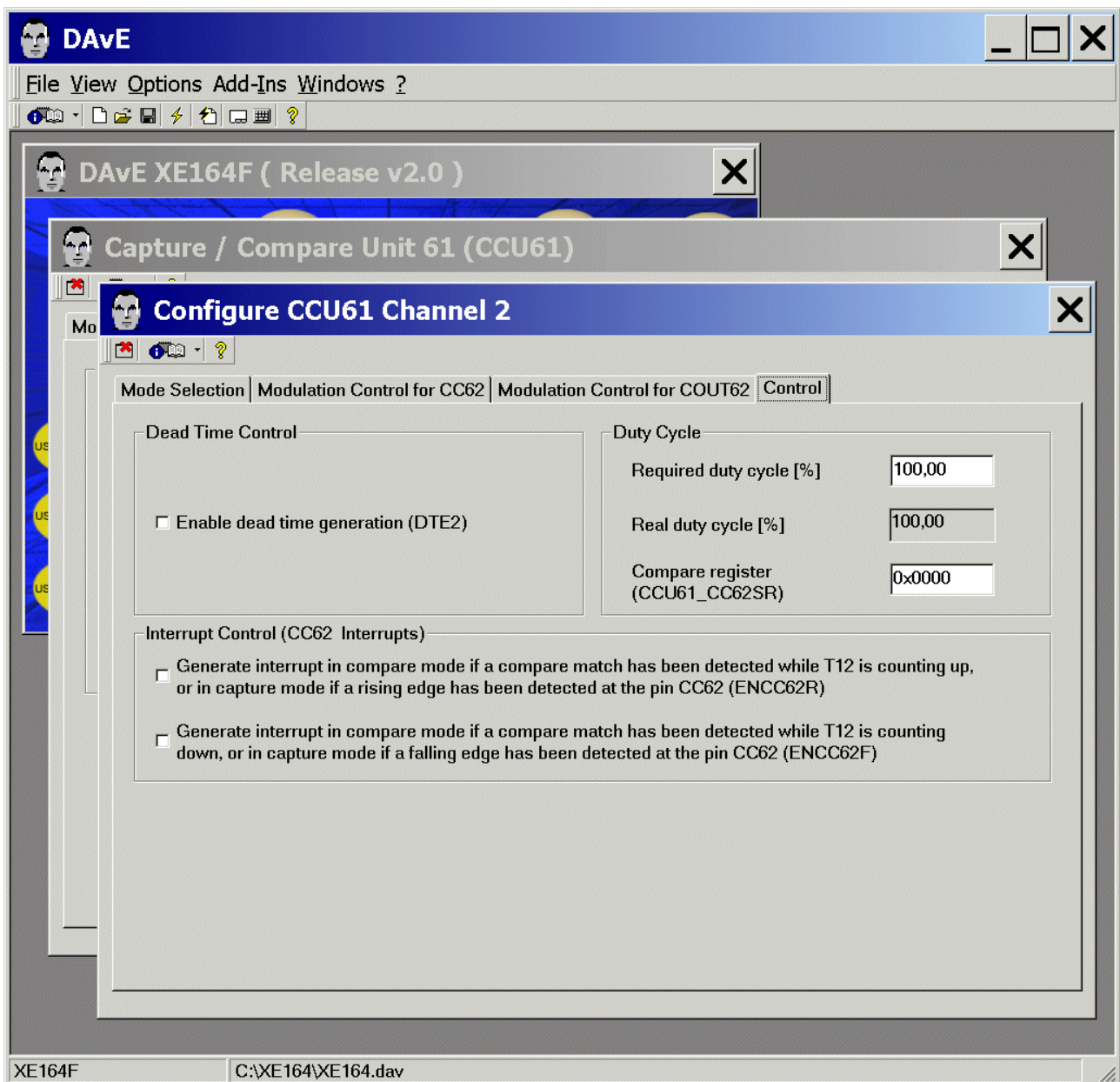



CCU61: Channels: Configure Channel 2: Modulation Control for COUT62: (do nothing)



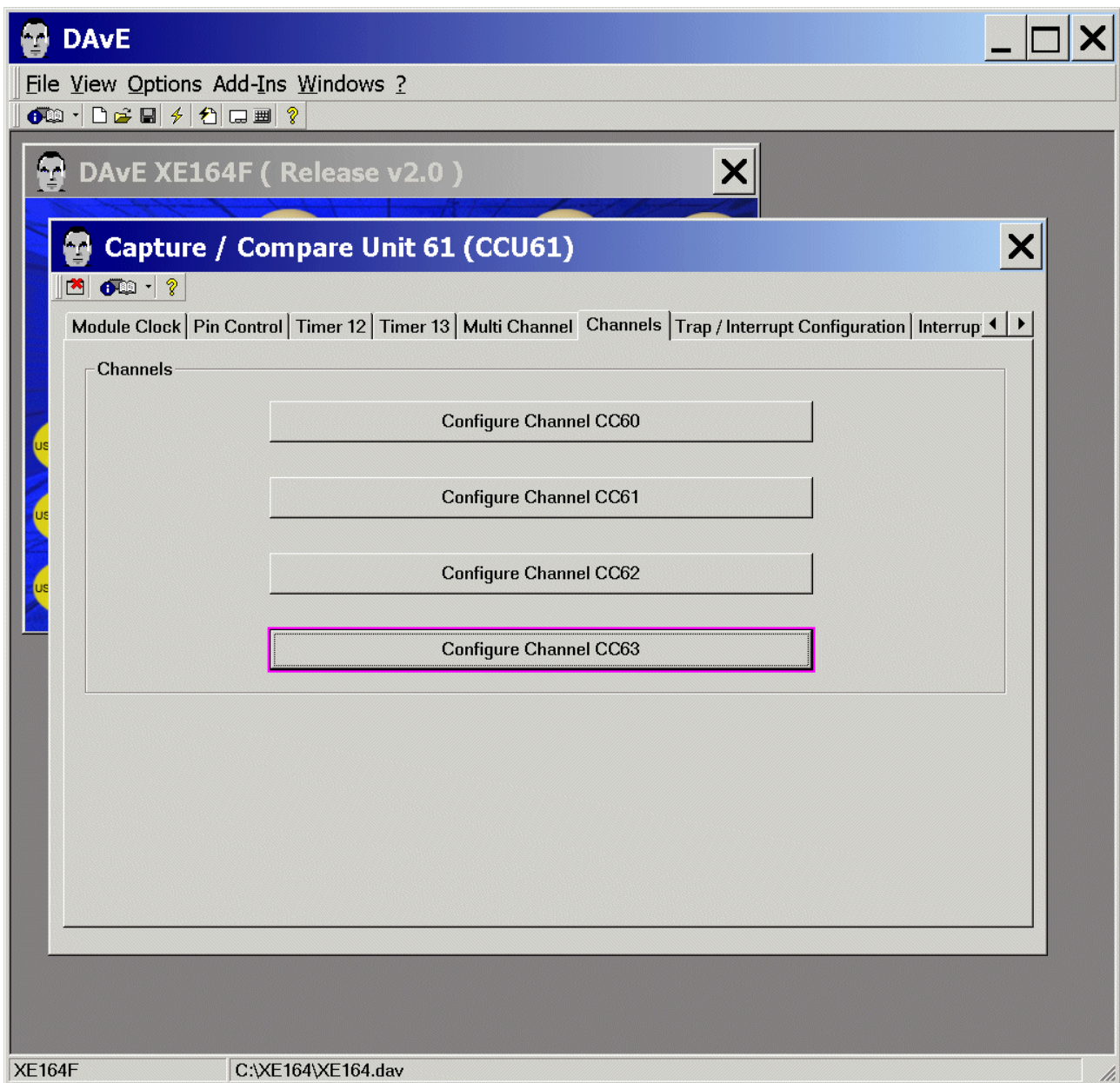


CCU61: Channels: Configure Channel 2: Control: (do nothing)



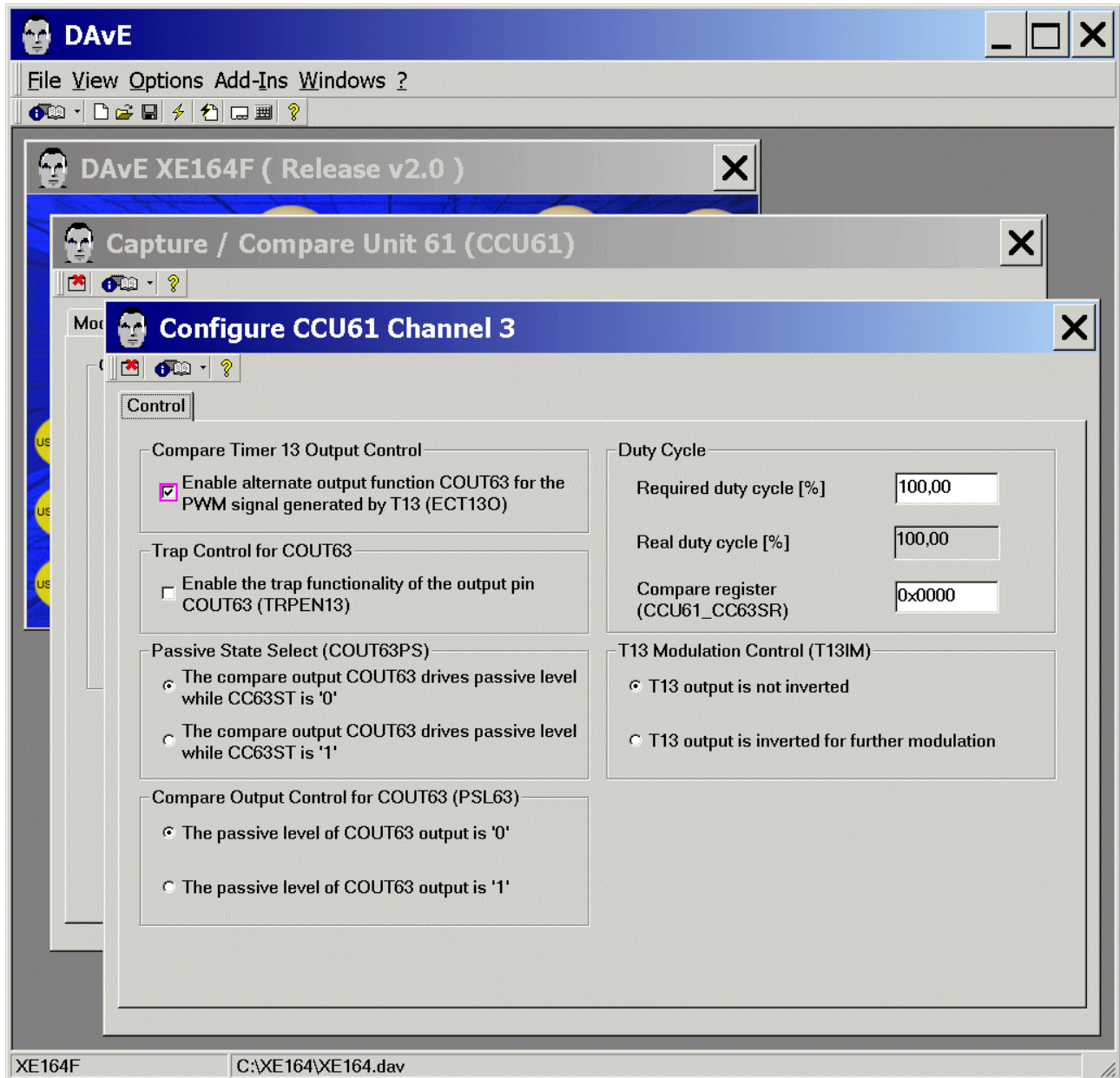
Exit and Save this dialog now by clicking  the close button.


CCU61: Channels: **click** Configure Channel 3



CCU61: Channels: Configure Channel 3:

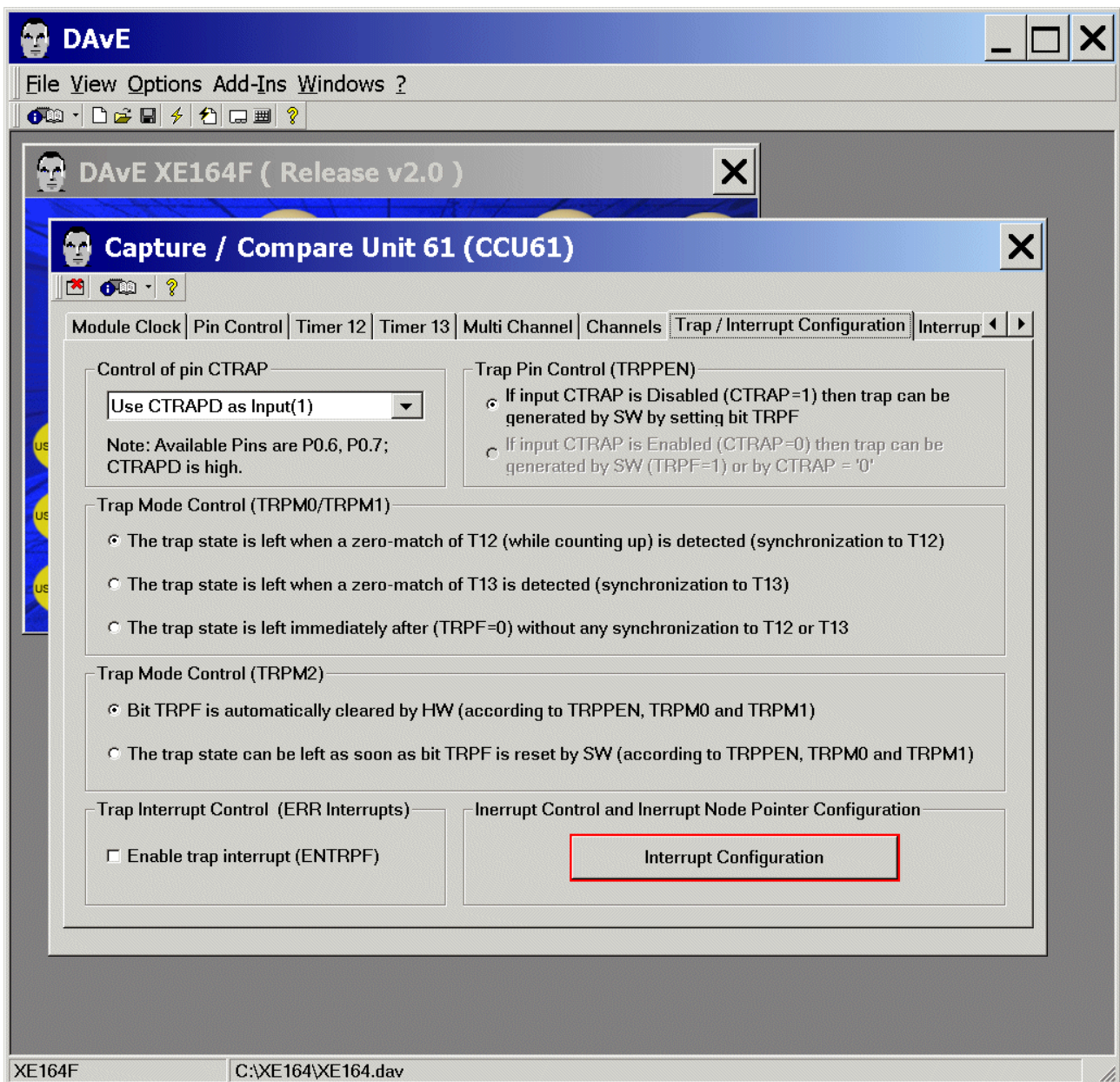
Control: Compare Timer 13 Output Control: click ☒ Enable alternate output function COUT63



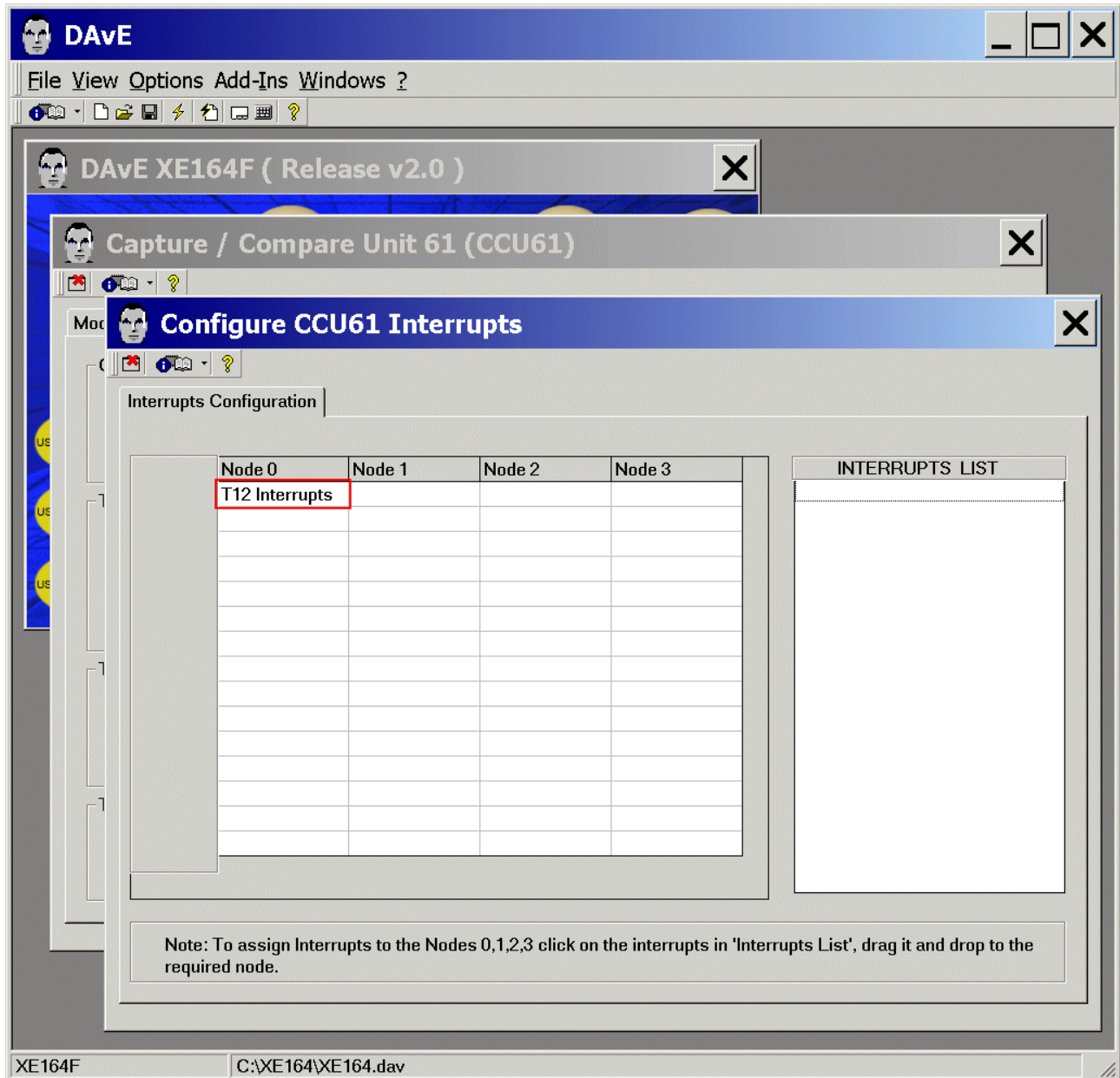
Exit and Save this dialog now by clicking  the close button.




CCU61: Trap / Interrupt Control: [click](#) Interrupt Configuration

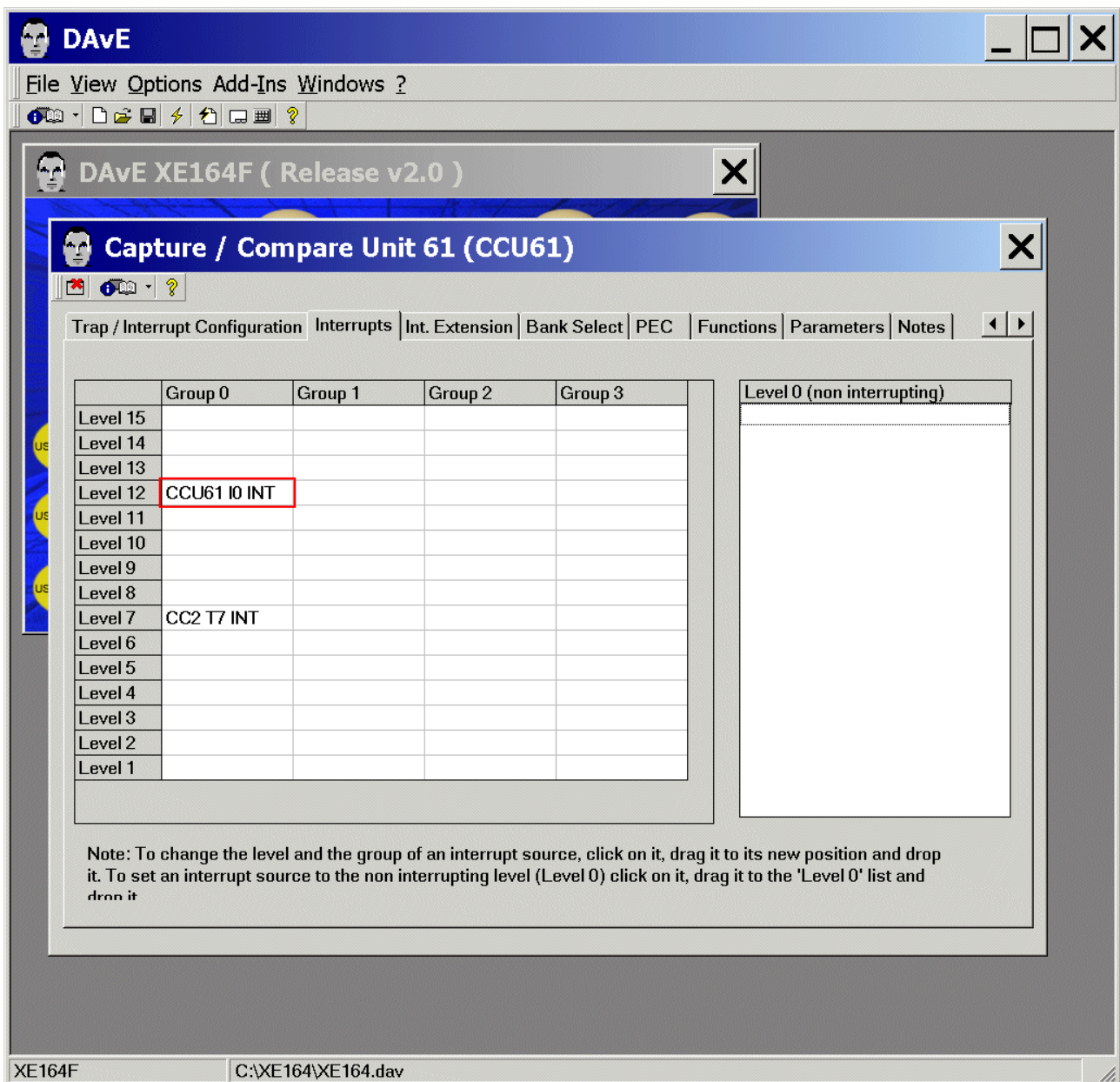


CCU61: Trap / Interrupt Control: Interrupt Configuration:  
Interrupts Configuration: drag and drop T12 Interrupts to Node 0



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

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



**DAve**

File View Options Add-Ins Windows ?

DAve XE164F ( Release v2.0 )

**Capture / Compare Unit 61 (CCU61)**

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

	Group 0	Group 1	Group 2	Group 3
Level 15				
Level 14				
Level 13				
Level 12	CCU61 I0 INT			
Level 11				
Level 10				
Level 9				
Level 8				
Level 7	CC2 T7 INT			
Level 6				
Level 5				
Level 4				
Level 3				
Level 2				
Level 1				

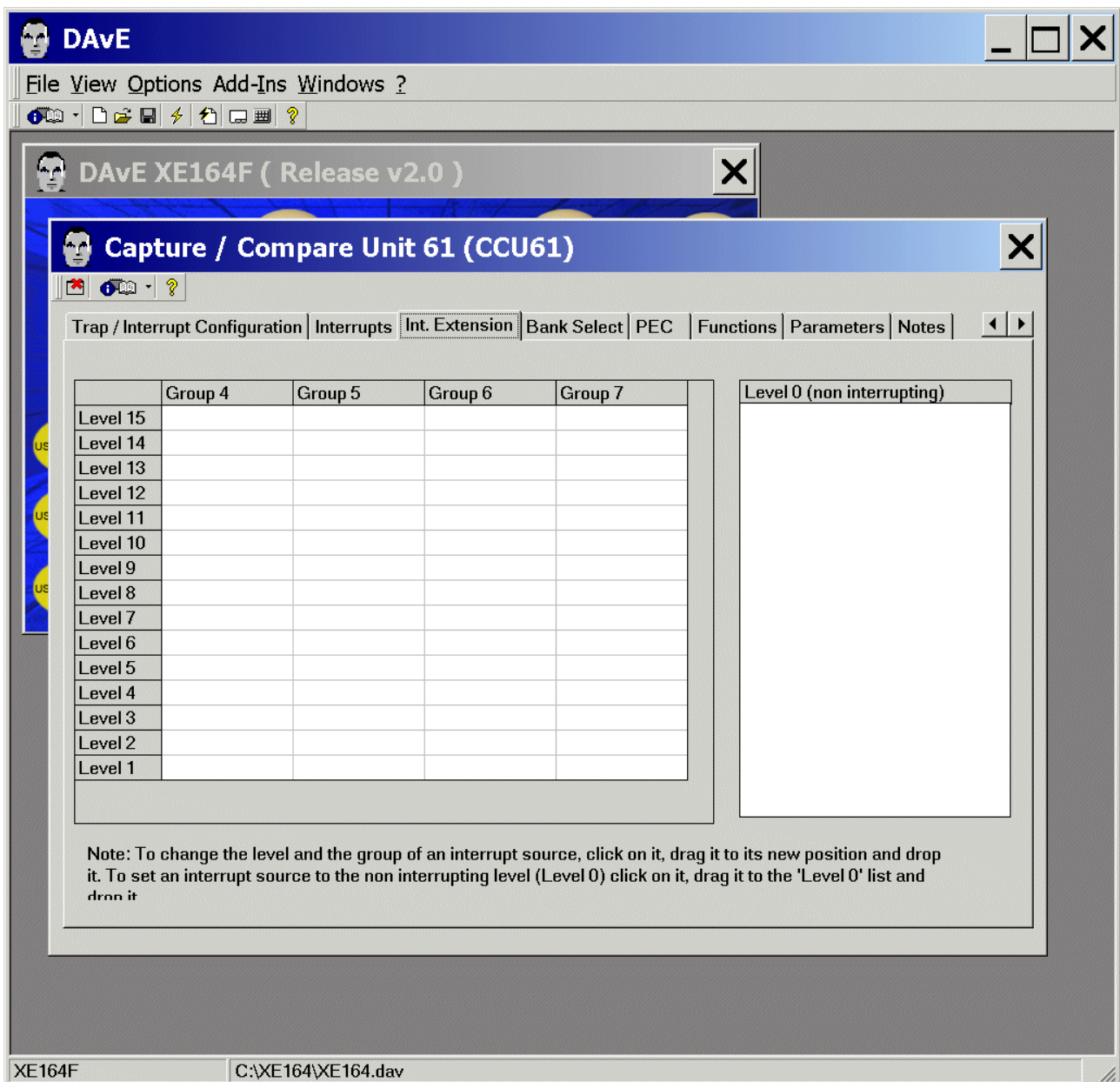
Level 0 (non interrupting)

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

XE164F C:\XE164\XE164.dav

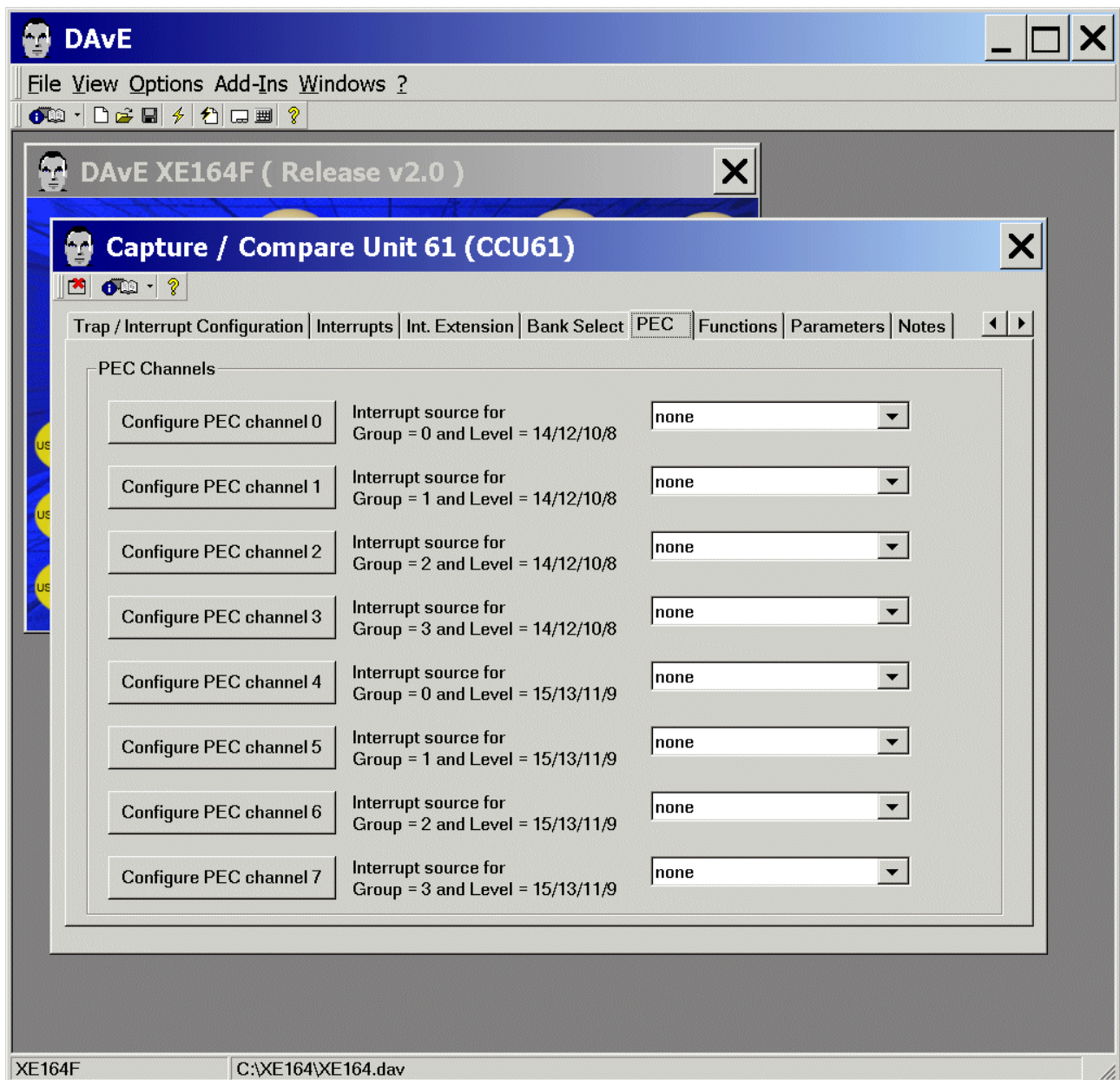


CCU61: Int. Extension: (do nothing)



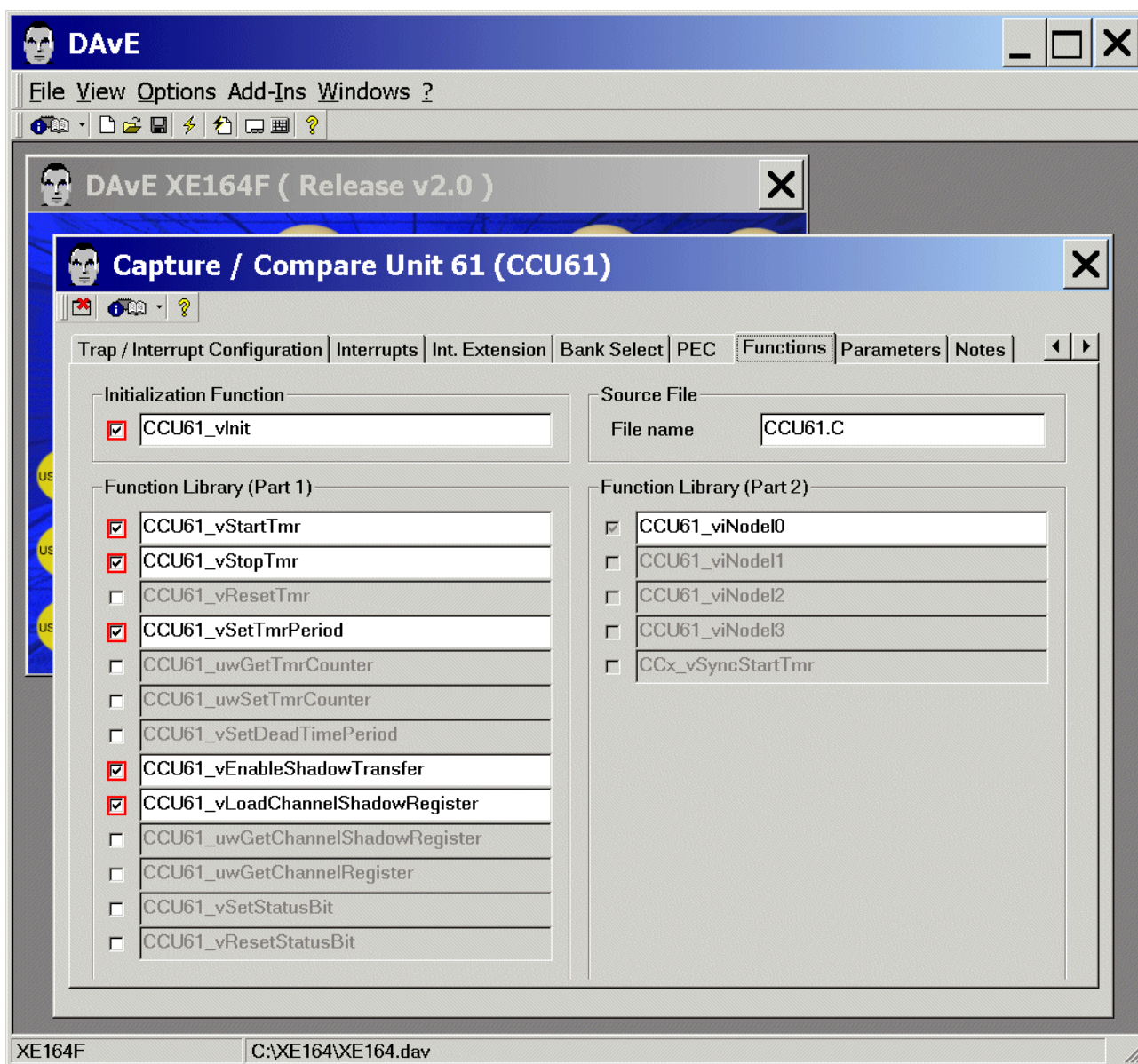
[illegible]

CCU61: PEC: (do nothing)

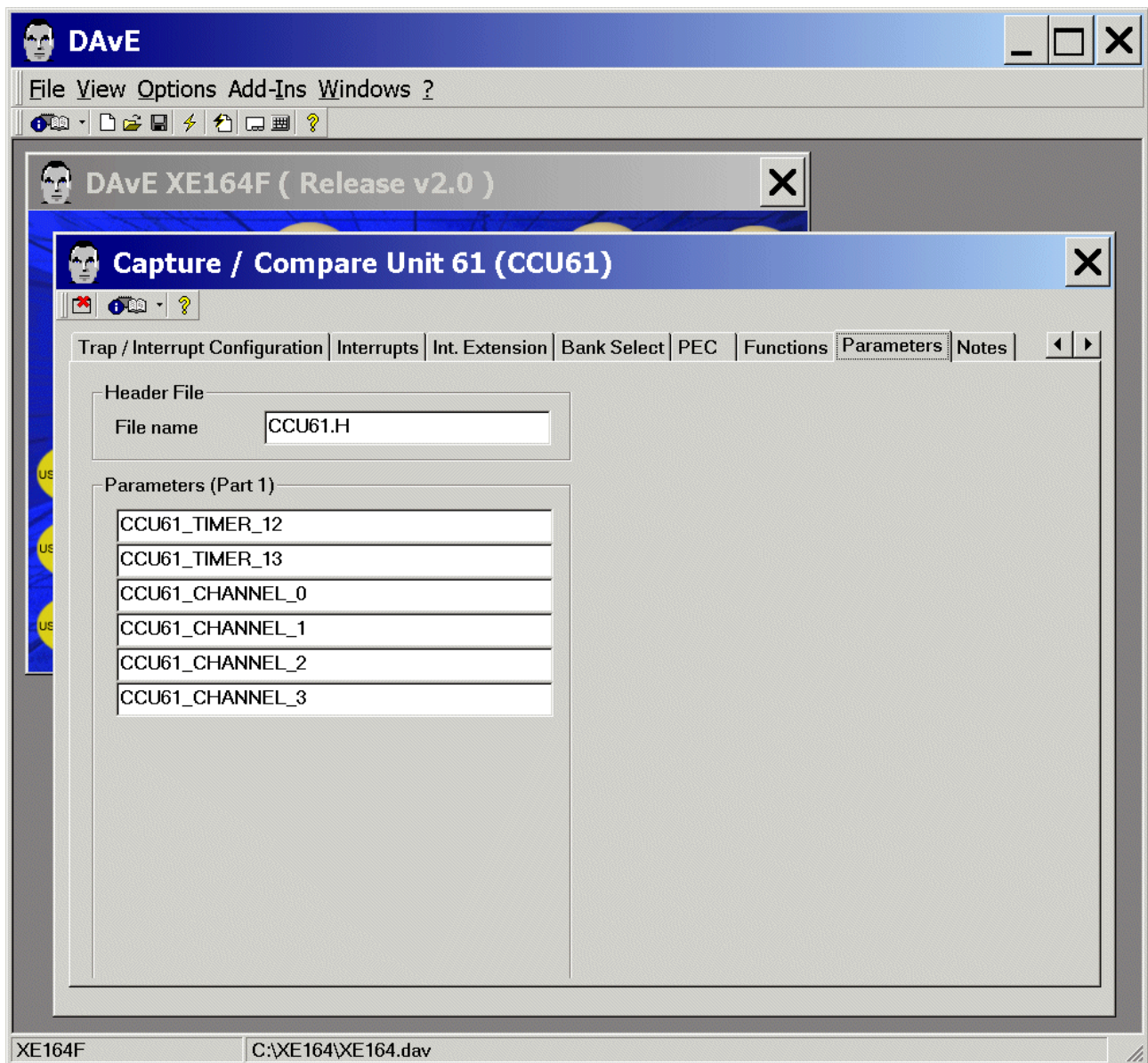





CCU61: Functions: Initialization Function: **click/check** ☒ CCU6\_vInit  
 CCU61: Functions: Function Library (Part 1): **click** ☒ CCU61\_vStartTmr  
 CCU61: Functions: Function Library (Part 1): **click** ☒ CCU61\_vStopTmr  
 CCU61: Functions: Function Library (Part 1): **click** ☒ CCU61\_vSetTmrPeriod  
 CCU61: Functions: Function Library (Part 1): **click** ☒ CCU61\_vEnableShadowTransfer  
 CCU61: Functions: Function Library (Part 1): **click** ☒ CCU61\_vLoadChannelShadowRegister




CCU61: Parameters: (do nothing)



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

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

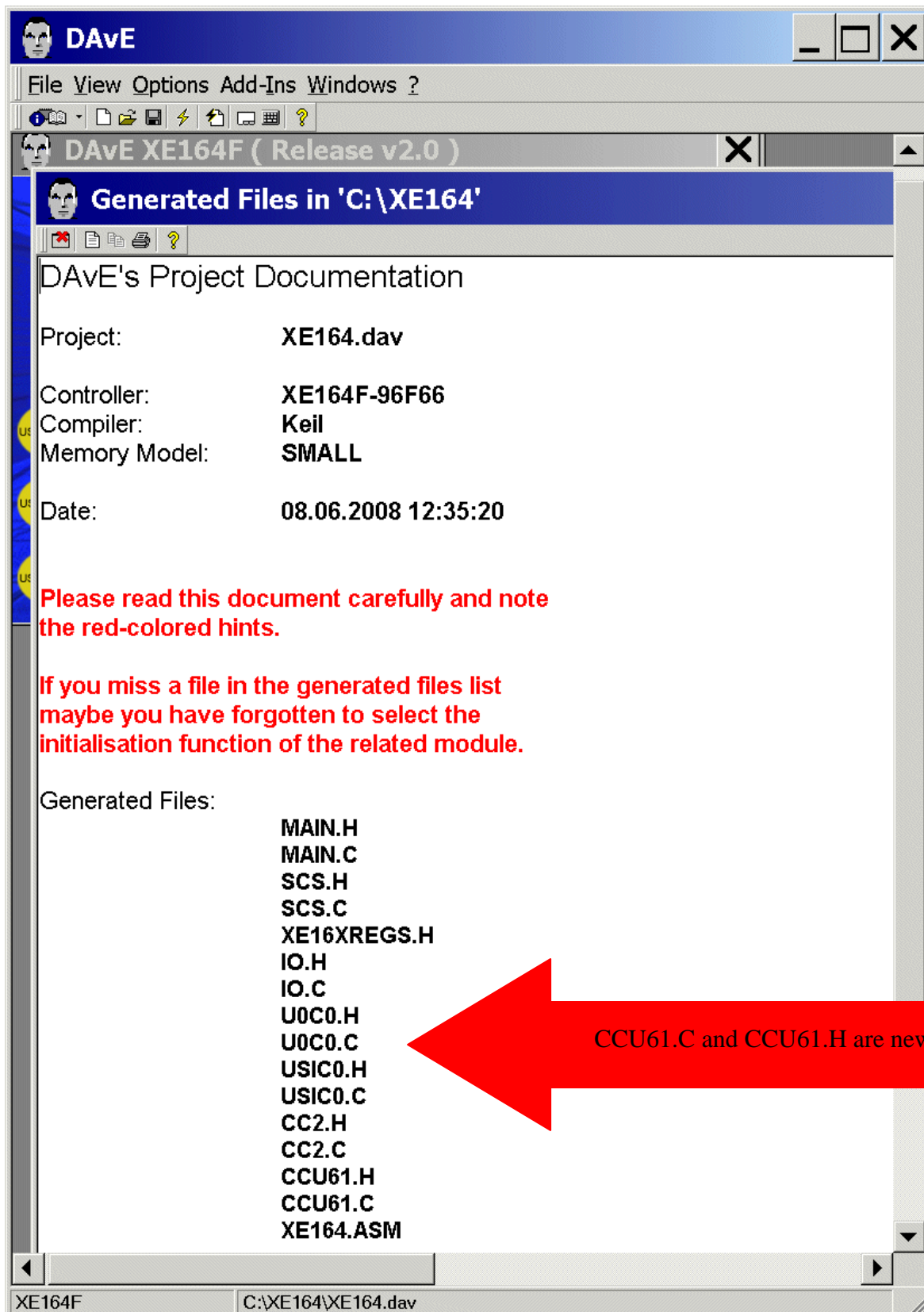
Generate Code:

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



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





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



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

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

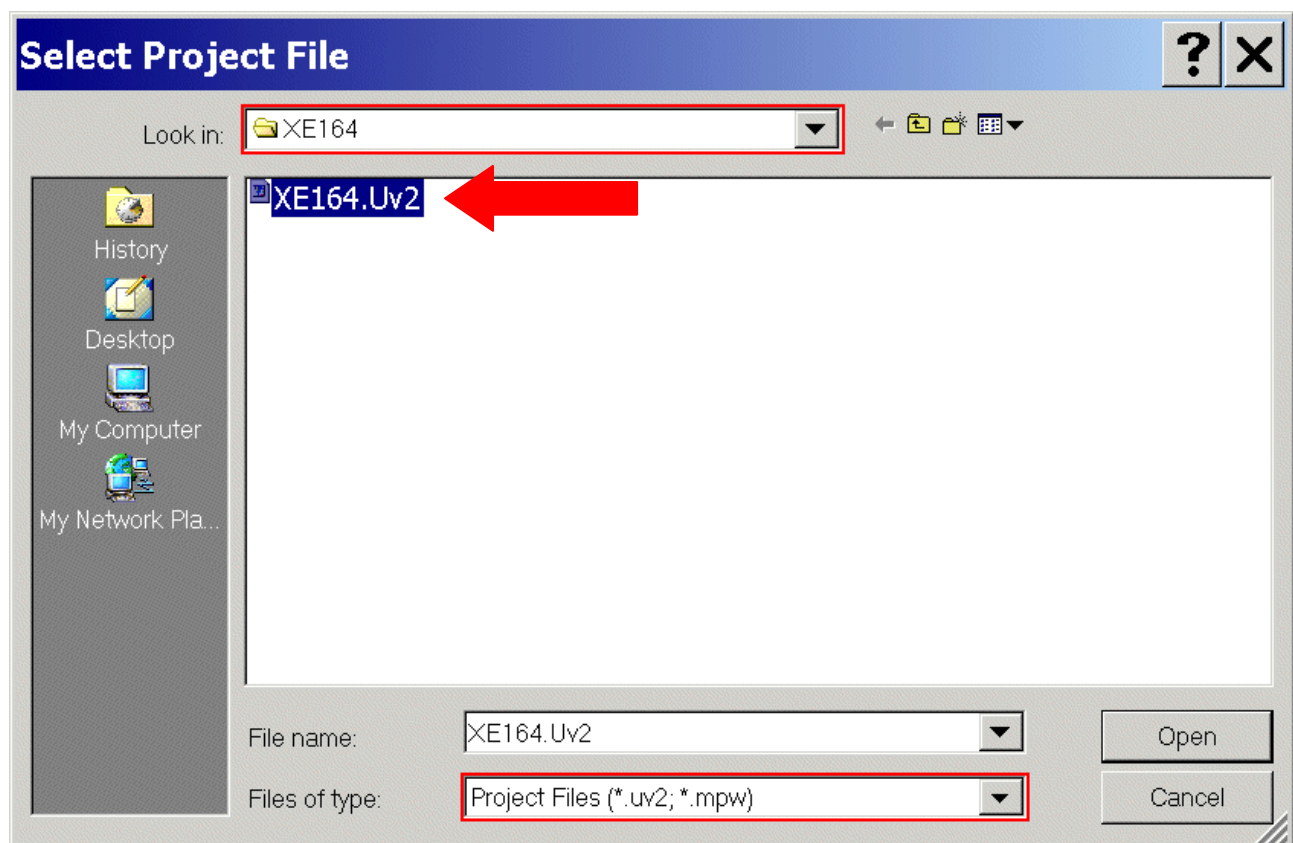
**Project - Open Project**

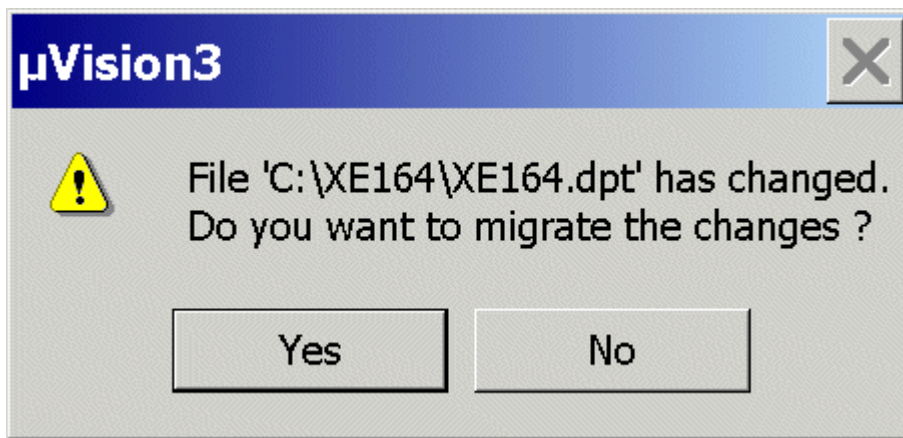
Select Project File: **Look in:** choose C:\XE164

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

**Click XE164.Uv2**

**Open**



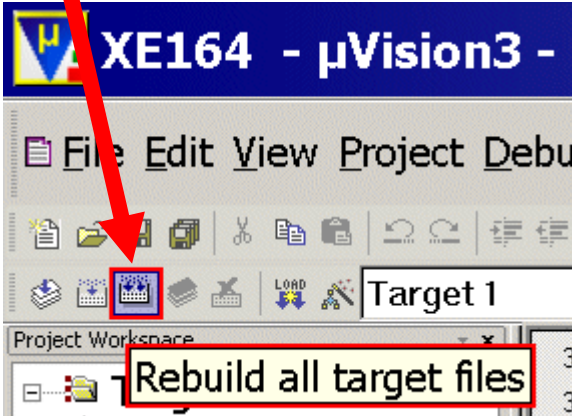


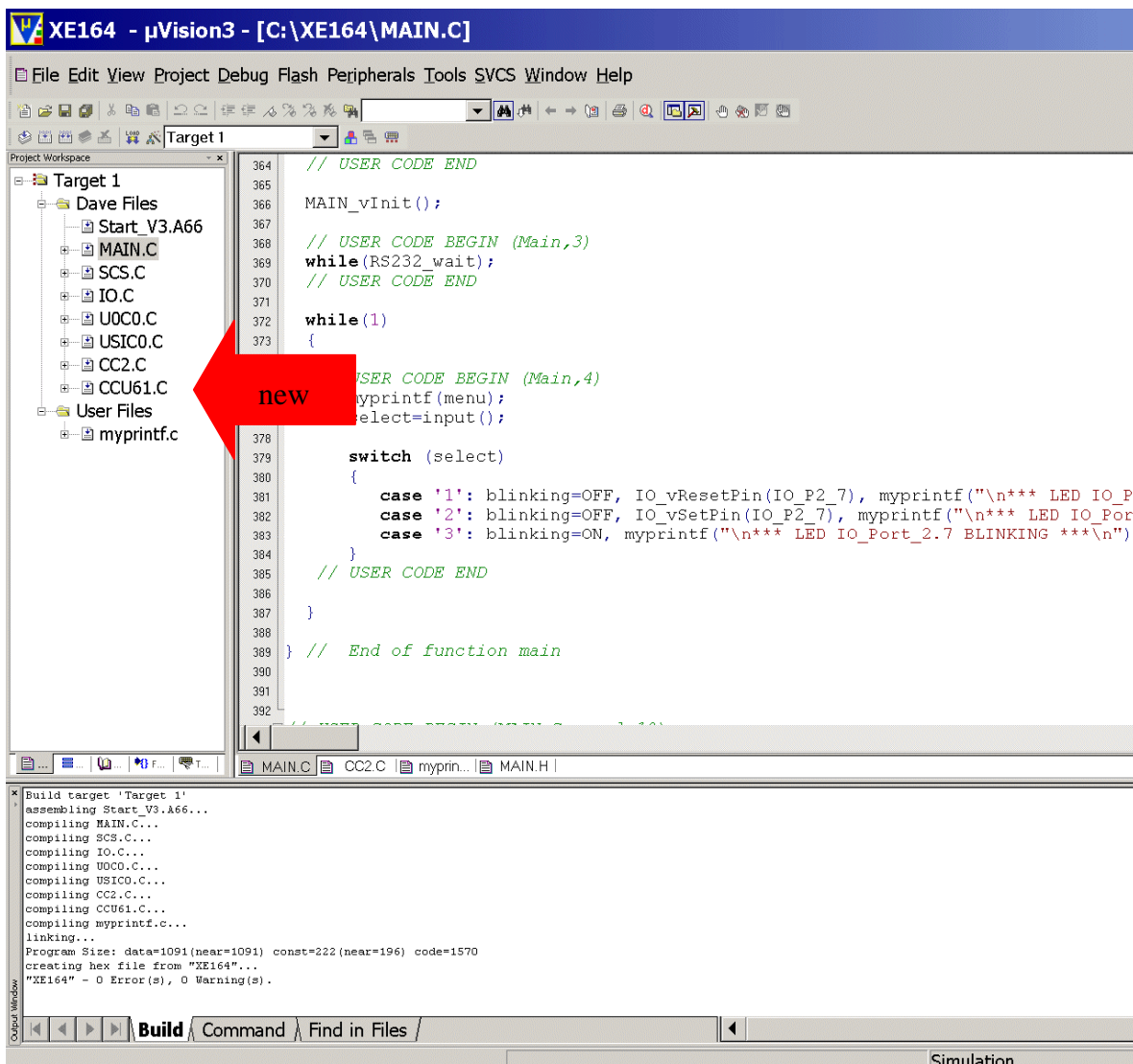
Click Yes



Project – Rebuild all target files

or click



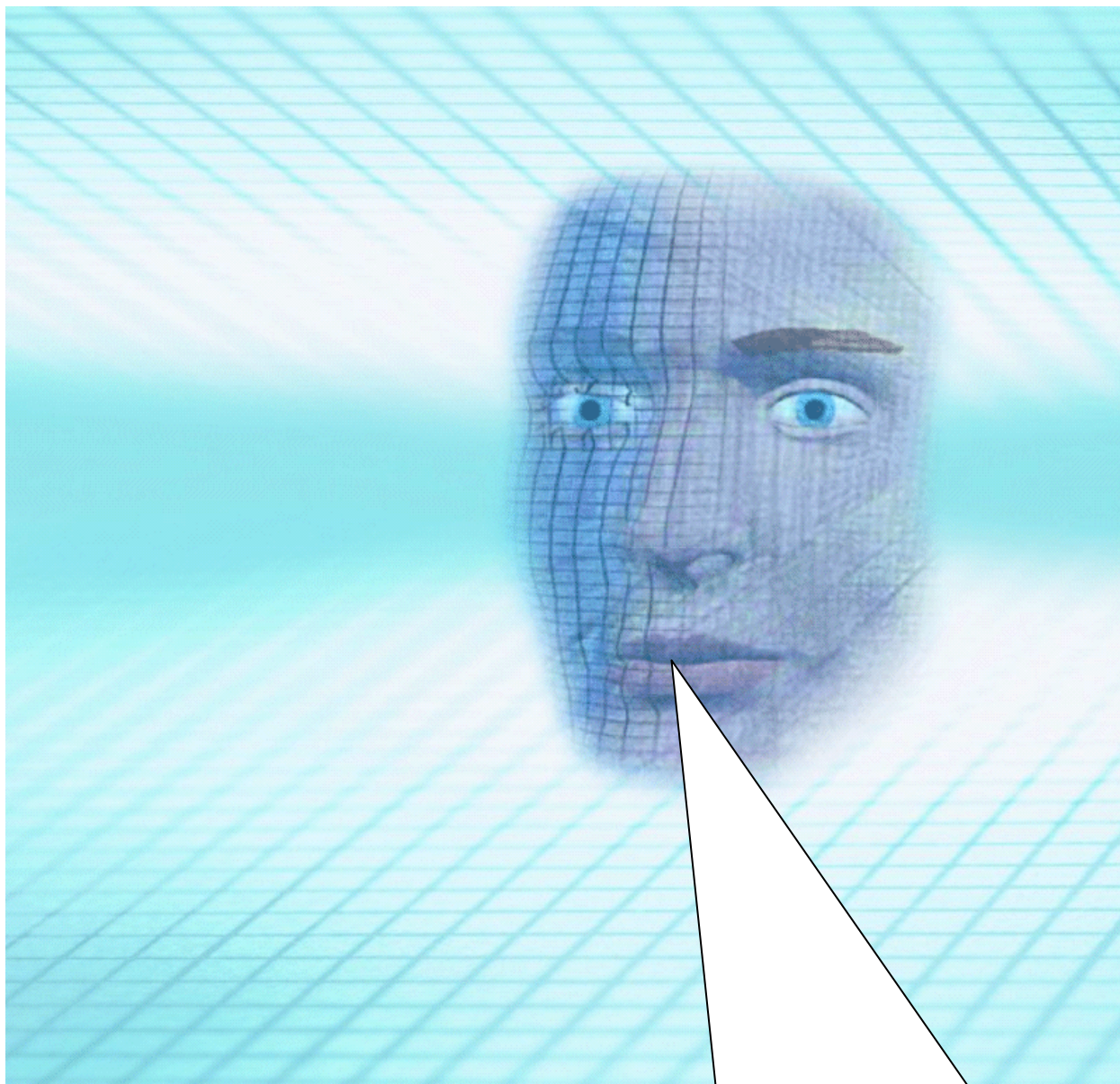


The screenshot shows the µVision3 IDE with the project 'XE164 - µVision3 - [C:\XE164\MAIN.C]'. The project workspace on the left shows a tree structure with 'Target 1' expanded, containing 'Dave Files' and 'User Files'. A red arrow points to the 'new' button in the project workspace tree. The main code editor shows the following code:

```

364 // USER CODE END
365
366 MAIN_vInit();
367
368 // USER CODE BEGIN (Main,3)
369 while(RS232 wait);
370 // USER CODE END
371
372 while(1)
373 {
374     // USER CODE BEGIN (Main,4)
375     myprintf(menu);
376     select=input();
377
378     switch (select)
379     {
380     case '1': blinking=OFF, IO_vResetPin(IO_P2_7), myprintf("\n*** LED IO_Pc
381     case '2': blinking=OFF, IO_vSetPin(IO_P2_7), myprintf("\n*** LED IO_Port
382     case '3': blinking=ON, myprintf("\n*** LED IO_Port_2.7 BLINKING ***\n");
383     }
384     // USER CODE END
385
386 }
387
388 } // End of function main
389
390
391
392
393 // USER CODE BEGIN (Main,5)
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2
```

Insert/Change your application specific program:



**Note:**

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

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

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

from:

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

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

to:

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

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

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



**XE164 - µVision3 - [C:\XE164\MAIN.C]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

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

```

MAIN.C | CC2.C | myprin... | MAIN.H

Build Command Find in Files

Simulation

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

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

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

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

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

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

**XE164 - µVision3 - [C:\XE164\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

093
094 volatile int RS232_wait=2;
095 char select=' ';
096
097 char mb1[500]; // message buffer for sprintf()
098
099 // *** Music ***:
100 /*
101  Construction of the music data:
102  =====
103  created by Christian Perschl (www.perschl.at)
104  extended by Wilhelm Brezovits
105
106  C,D,E,F,G,A,H: play note
107  +: the + raises its note a semitone: Cis, Dis, Eis, Fis, Gis, Ais, His
108  -: the - lowers its note a semitone: Ces, Des, Es, Fes, Ges, As, Hes
109  Lx : Change note length
110      (x = 1,2,4,8,16 -> 1=whole-note, 2=half-note, 4=quarter-note, ...)
111  Px : play rest
112      (x = 1,2,4,8,16 -> 1=whole-rest, 2=half-rest, 4=quarter-rest, ...)
113  Ox : Change octave (x = 0,1,2,3)
114      . : Extend preceding note by half of its value
115  Tx : Change tempo (x = 50 ... 199 beats per minute)
116
117  Additional functionality:
118  =====
119  OL : activate octave LOW
120  ON : deactivate octave LOW = activate octave normal (00,01,02,03)
121  */
122
123 unsigned int T13 values[] = {62977, 59550, 56122, 53061, 50000, 47278, 44685, 42092, 39796, 37500, 35450, 33401, 275};
124 // Timer-T13-periods(frequencies) of the notes
125 // [0]=c', [1]=cis', [2]=d', [3]=dis', [4]=e', [5]=f',
126 // [6]=fis', [7]=g', [8]=gis', [9]=a', [10]=ais', [11]=h',
127 // [12]=<Frequency for rest>
128
129 unsigned int length_of_a_whole_note = 16113;
130 // Default length of a whole-note with tempo 120
131
132 // USER CODE END
133

```

MAIN.C CC2.C myprn... MAIN.H

Build Command Find in Files

Simulation L:131 C:1



Double click MAIN.C and insert Global Variables:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

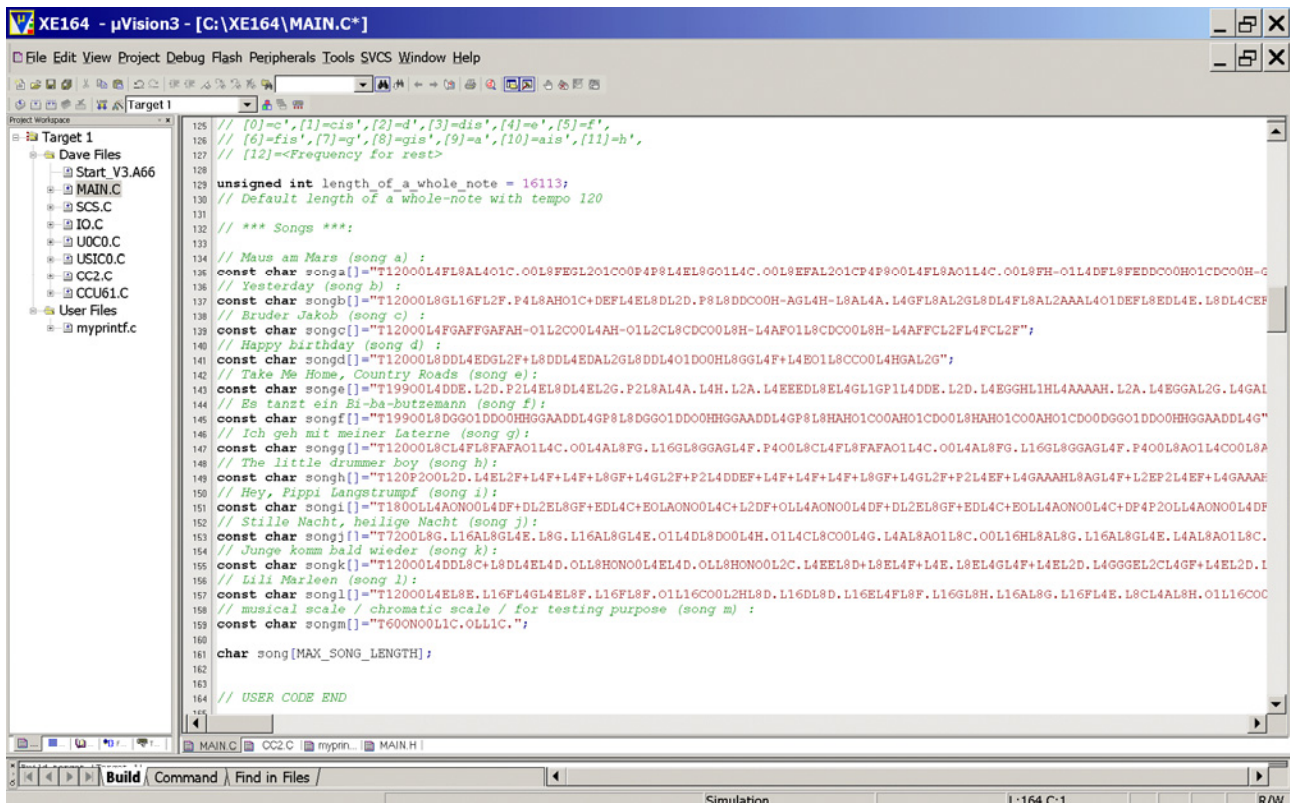
```
const char  
songh[]="T120P2O0L2D.L4EL2F+L4F+L4F+L8GF+L4GL2F+P2L4DDEF+L4F+L4F+L4F+L8G  
F+L4GL2F+P2L4EF+L4GAAHL8AGL4F+L2EP2L4EF+L4GAAHO1L8CO0L8HL4AL2GL8  
HAL4GL2F+L8AGL4F+L2EP1L2D.L4EL4F+F+F+F+L8GF+L4GL2F+P1L8EDL4EL2D";
```

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

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

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

char song[MAX_SONG_LENGTH];
```



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

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

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

```
// song counters:
```

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

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



**XE164 - µVision3 - [C:\XE164\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

150 // Hey, Pippi Langstrumpf (song i):
151 const char songi[]="T180OLL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+OLL4AON00L4DF
152 // Stille Nacht, heilige Nacht (song j):
153 const char songj[]="T7200L8G.L16AL8GL4E.L8G.L16AL8GL4E.O1L4DL8DO0L4H.O1L4CL8CO0L4G.
154 // Junge komm bald wieder (song k):
155 const char songk[]="T12000L4DDL8C+L8DL4EL4D.OLL8HON00L4EL4D.OLL8HON00L2C.L4EEL8D+L8
156 // Lili Marleen (song l):
157 const char songl[]="T12000L4EL8E.L16FL4GL4EL8F.L16FL8F.O1L16CO0L2HL8D.L16DL8D.L16EL
158 // musical scale / chromatic scale / for testing purpose (song m) :
159 const char songm[]="T600N00L1C.OLL1C.";
160
161 char song[MAX_SONG_LENGTH];
162
163 // default values for global variables - will be overwritten before use:
164 volatile unsigned int note=12;
165 volatile unsigned int octave=1;
166 volatile unsigned int current_note_length=16113;
167 volatile unsigned int old_note_length=16113;
168 volatile unsigned int tempo=120; // 120 beats/minute
169 volatile unsigned int pos=0; // current note
170 volatile unsigned int max=0; // song length
171
172 // song counters:
173 unsigned int next_song_a=0; // song counter song a
174 unsigned int next_song_b=0; // song counter song b
175 unsigned int next_song_c=0; // song counter song c
176 unsigned int next_song_d=0; // song counter song d
177 unsigned int next_song_e=0; // song counter song e
178 unsigned int next_song_f=0; // song counter song f
179 unsigned int next_song_g=0; // song counter song g
180 unsigned int next_song_h=0; // song counter song h
181 unsigned int next_song_i=0; // song counter song i
182 unsigned int next_song_j=0; // song counter song j
183 unsigned int next_song_k=0; // song counter song k
184 unsigned int next_song_l=0; // song counter song l
185 unsigned int next_song_m=0; // song counter song m
186
187 volatile bit OctaveLOW=OFF;
188
189 // USER CODE END
190

```

MAIN.C CC2.C myprin... MAIN.H

Build Command Find in Files

Simulation

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

from:

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

to:

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

**XE164 - µVision3 - [C:\XE164\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

448
449 // USER CODE BEGIN (Main,1)
450 char input (void)
451 {
452     char in=' ';
453     do
454     {
455         myprintf("your choice: ");
456         in = (char)U0C0_ASC_uwGetData();
457     }while ( !(in>='a'&&in<='m') );
458     return in;
459 }
460 // USER CODE END
461
462 void main(void)
463 {
464     // USER CODE BEGIN (Main,2)
465
466     // USER CODE END
467
468     MAIN_vInit();
469
470     // USER CODE BEGIN (Main,3)
471     while(RS232_wait);
472     // USER CODE END
473
474     while(1)
475     {
476
477         // USER CODE BEGIN (Main,4)
478         myprintf(menu);
479         select=input();
480
481         switch (select)
482         {
483             case '1': blinking=OFF, IO_vResetPin(IO_P2_7), myprintf("\n*** LED
484             case '2': blinking=OFF, IO_vSetPin(IO_P2_7), myprintf("\n*** LED I
485             case '3': blinking=ON, myprintf("\n*** LED IO_Port_2.7 BLINKING **
486         }
487         // USER CODE END
488     }
489 }

```

MAIN.C CC2.C myprin... MAIN.H

Build target 1 Build Command Find in Files

Simulation



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

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

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

```
pos=0;

if (max>0) // there is something to play
{
    // start CCU61 - Timer T12 - ISR the first time
    CCU61_ISS = CCU61_IS | 0x80; // set ST12PM -> Set-Timer-T12Period-Match-Flag

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

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



XE164 - µVision3 - [C:\XE164\MAIN.C\*]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

449 // USER CODE BEGIN (Main,1)
450 char input (void)
451 {
452     char in=' ';
453     do
454     {
455         myprintf("your choice: ");
456         in = (char)U0C0_ASC_uwGetData();
457     }while ( !(in>='a'&&in<='m') );
458     return in;
459 }
460
461 void play_song(void)
462 {
463     max=0;
464     if ( next_song_a && ((sizeof(songa)-1)< MAX_SONG_LENGTH) )
465         strcpy(song,songa), max=(sizeof(songa))-1, --next_song_a,
466         myprintf("\nplaying: Maus am Mars\n");
467     if ( next_song_b && ((sizeof(songb)-1)< MAX_SONG_LENGTH) )
468         strcpy(song,songb), max=(sizeof(songb))-1, --next_song_b,
469         myprintf("\nplaying: Yesterday\n");
470     if ( next_song_c && ((sizeof(songc)-1)< MAX_SONG_LENGTH) )
471         strcpy(song,songc), max=(sizeof(songc))-1, --next_song_c,
472         myprintf("\nplaying: Frere Jacques - Lazy John - Bruder Jakob\n");
473     if ( next_song_d && ((sizeof(songd)-1)< MAX_SONG_LENGTH) )
474         strcpy(song,songd), max=(sizeof(songd))-1, --next_song_d,
475         myprintf("\nplaying: Happy birthday\n");
476     if ( next_song_e && ((sizeof(songe)-1)< MAX_SONG_LENGTH) )
477         strcpy(song,songe), max=(sizeof(songe))-1, --next_song_e,
478         myprintf("\nplaying: Take Me Home, Country Roads\n");
479     if ( next_song_f && ((sizeof(songf)-1)< MAX_SONG_LENGTH) )
480         strcpy(song,songf), max=(sizeof(songf))-1, --next_song_f,
481         myprintf("\nplaying: Es tanzt ein Bi-ba-butzemann\n");
482     if ( next_song_g && ((sizeof(songg)-1)< MAX_SONG_LENGTH) )
483         strcpy(song,songg), max=(sizeof(songg))-1, --next_song_g,
484         myprintf("\nplaying: Ich geh mit meiner Laterne\n");
485     if ( next_song_h && ((sizeof(songh)-1)< MAX_SONG_LENGTH) )
486         strcpy(song,songh), max=(sizeof(songh))-1, --next_song_h,
487         myprintf("\nplaying: The little drummer boy\n");
488     if ( next_song_i && ((sizeof(songi)-1)< MAX_SONG_LENGTH) )
489         strcpy(song,songi), max=(sizeof(songi))-1, --next_song_i,
490         myprintf("\nplaying: Hey, Pippi Langstrumpf\n");
491     if ( next_song_j && ((sizeof(songj)-1)< MAX_SONG_LENGTH) )
492         strcpy(song,songj), max=(sizeof(songj))-1, --next_song_j,
493         myprintf("\nplaying: Stille Nacht, heilige Nacht\n");
494     if ( next_song_k && ((sizeof(songk)-1)< MAX_SONG_LENGTH) )
495         strcpy(song,songk), max=(sizeof(songk))-1, --next_song_k,
496         myprintf("\nplaying: Junge komm bald wieder\n");
497     if ( next_song_l && ((sizeof(songl)-1)< MAX_SONG_LENGTH) )
498         strcpy(song,songl), max=(sizeof(songl))-1, --next_song_l,
499         myprintf("\nplaying: Lili Marleen\n");
500     if ( next_song_m && ((sizeof(songm)-1)< MAX_SONG_LENGTH) )
501         strcpy(song,songm), max=(sizeof(songm))-1, --next_song_m,
502         myprintf("\nplaying: musical scale / chromatic scale / for testing purpose\n");
503
504     sprintf(mbl,"song-length = %5u Byte[s] \n",max);
505     myprintf(mbl);
506     pos=0;
507
508     if (max>0) // there is something to play
509     {
510         // start CCU61 - Timer T12 - ISR the first time
511         CCU61_ISS = CCU61_IS | 0x80; // set ST12PM -> Set-Timer-T12Period-Match-Flag
512
513         while (pos<=max); // wait until song end is reached or abort by user is done
514     }
515
516     if ( (U0C0_RBUF=='z') )
517     {
518         myprintf("Song aborted.\n");
519     }
520     else
521     {
522         sprintf(mbl,"End of the song reached (pos=%5u of max%5u).\n",pos,max);
523         myprintf(mbl);
524     }
525 }
526 // USER CODE END
527
528 void main(void)

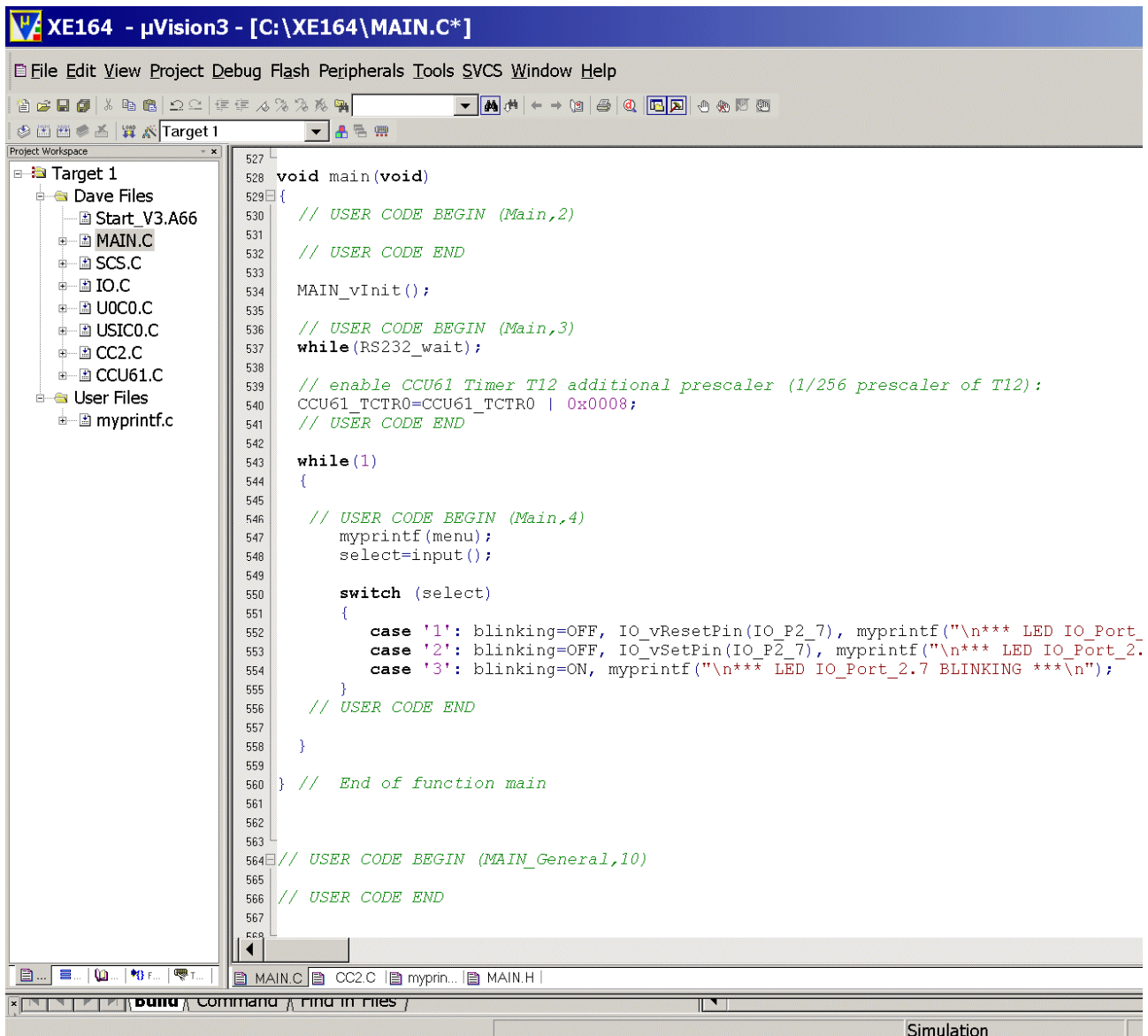
```

MAIN.C CC2.C myprin... MAIN.H

Simulation

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

```
// enable CCU61 Timer T12 additional prescaler (1/256 prescaler of T12):
CCU61_TCTR0=CCU61_TCTR0 | 0x0008;
```







**Remember:**

Unfortunately bit T12PRE is not available in the DAVe dialog.

Source: User's Manual:

The input clock for timer T12 can be from  $f_{CCU61}$  to a maximum of  $f_{CCU61}/128$  and is configured by bit field T12CLK. In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler of T12 if bit T12PRE = 1.

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

File Edit View Document Tools Window Help

125%

**TCTR0**  
Timer Control Register 0

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

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

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

**Timer 12 Resolution:**

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

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

from:

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

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

to:

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

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

**XE164 - µVision3 - [C:\XE164\MAIN.C\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

527 void main(void)
528 {
529     // USER CODE BEGIN (Main,2)
530
531     // USER CODE END
532
533     MAIN_vInit();
534
535     // USER CODE BEGIN (Main,3)
536     while(RS232_wait);
537
538     // enable CCU61 Timer T12 additional prescaler (1/256 prescaler of T12):
539     CCU61_TCTR0=CCU61_TCTR0 | 0x0008;
540     // USER CODE END
541
542     while(1)
543     {
544         // USER CODE BEGIN (Main,4)
545         myprintf(menu);
546         select=input();
547
548         switch (select)
549         {
550             case 'a': ++next_song_a, play_song(); break;
551             case 'b': ++next_song_b, play_song(); break;
552             case 'c': ++next_song_c, play_song(); break;
553             case 'd': ++next_song_d, play_song(); break;
554             case 'e': ++next_song_e, play_song(); break;
555             case 'f': ++next_song_f, play_song(); break;
556             case 'g': ++next_song_g, play_song(); break;
557             case 'h': ++next_song_h, play_song(); break;
558             case 'i': ++next_song_i, play_song(); break;
559             case 'j': ++next_song_j, play_song(); break;
560             case 'k': ++next_song_k, play_song(); break;
561             case 'l': ++next_song_l, play_song(); break;
562             case 'm': ++next_song_m, play_song(); break;
563
564         }
565         // USER CODE END
566     }
567 }

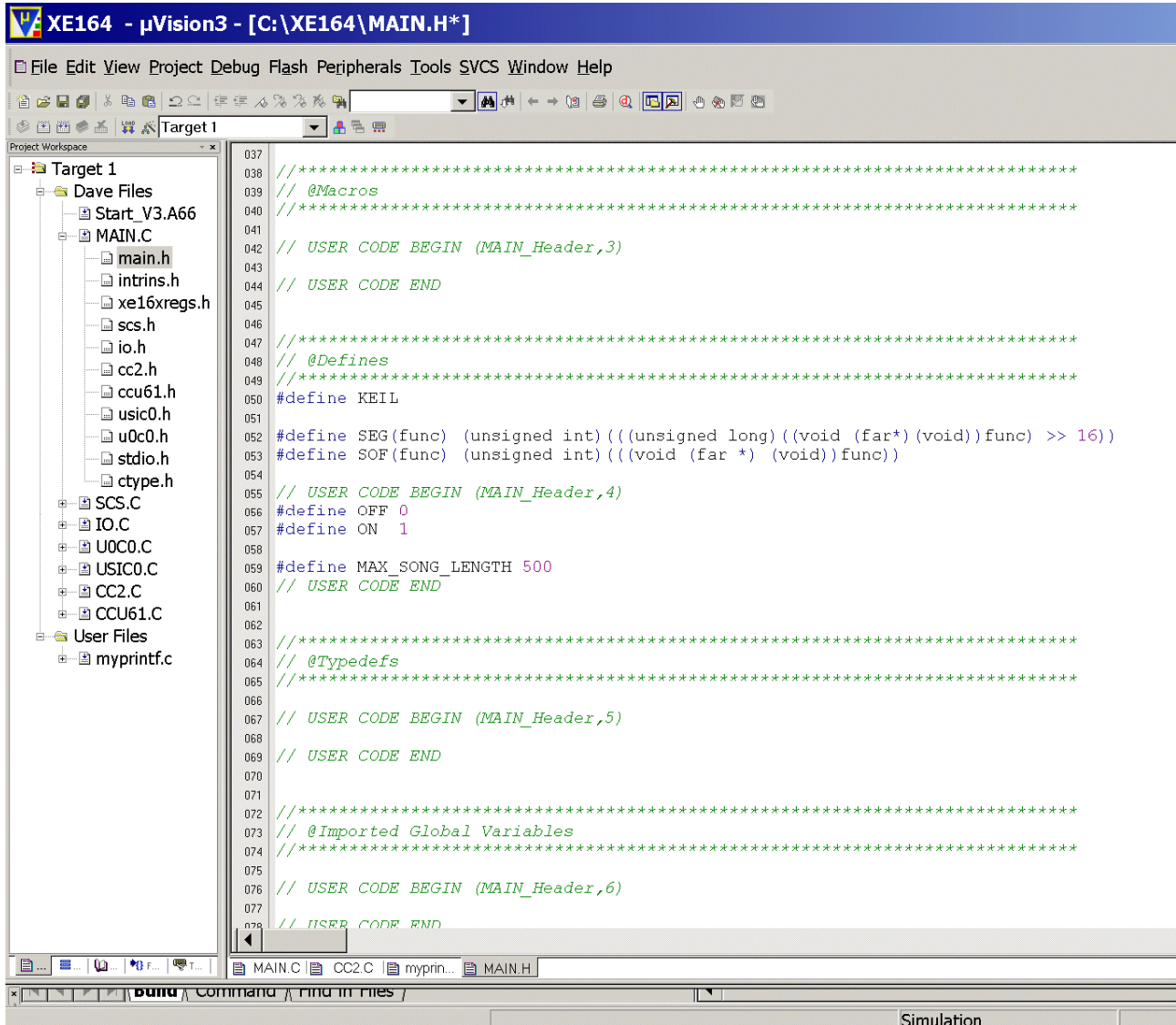
```

MAIN.C CC2.C myprin... MAIN.H

Simulation

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

```
#define MAX_SONG_LENGTH 500
```





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

from:

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

to:

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

**XE164 - µVision3 - [C:\XE164\MAIN.H\*]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
      - main.h
      - intrins.h
      - xe16xregs.h
      - scs.h
      - io.h
      - cc2.h
      - ccu61.h
      - usio0.h
      - u0c0.h
      - stdio.h
      - ctype.h
    - SCS.C
    - IO.C
    - U0C0.C
    - USIO0.C
    - CC2.C
    - CCU61.C
  - User Files
    - myprintf.c

```

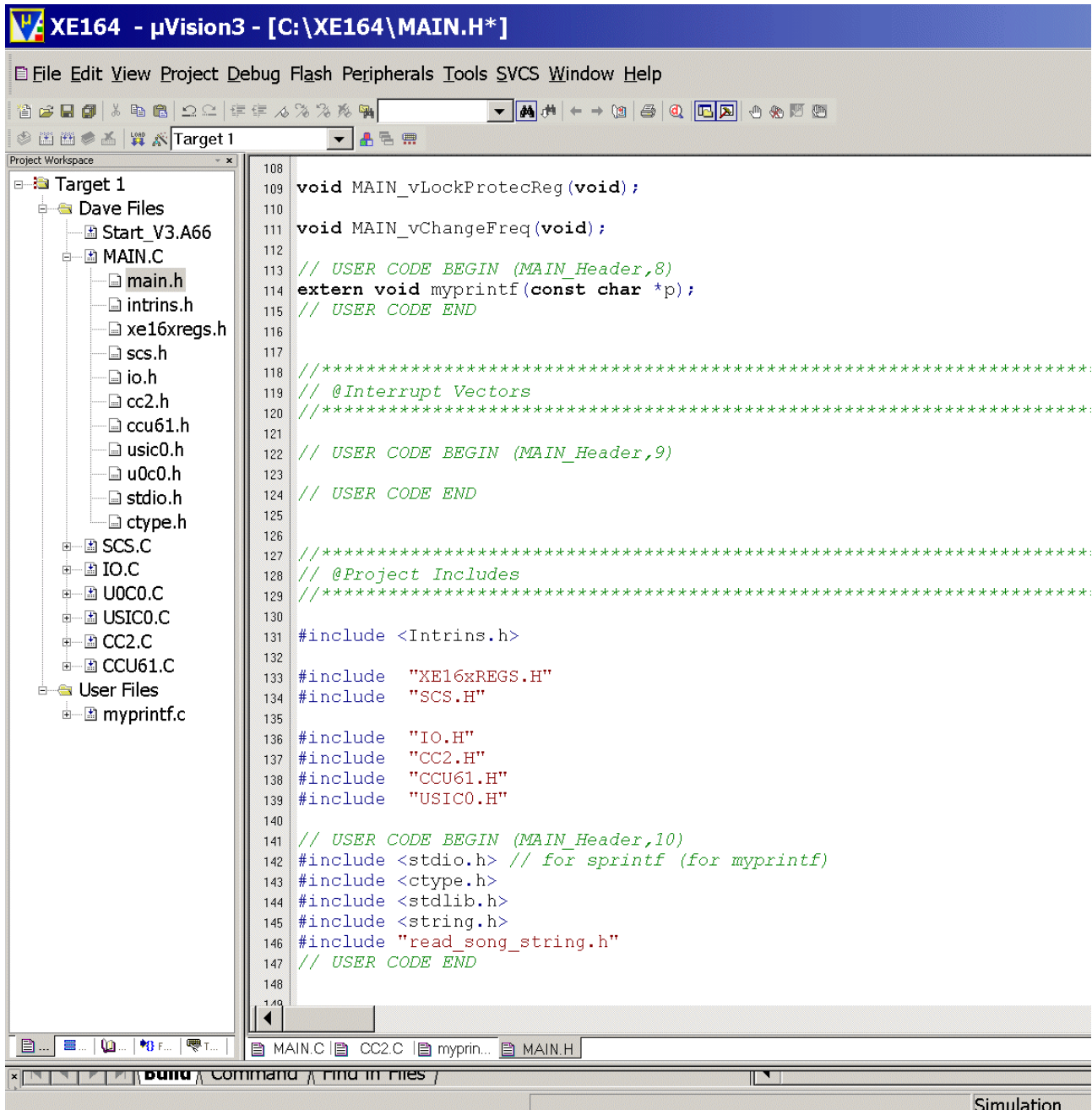
075 // USER CODE BEGIN (MAIN_Header,6)
076
077 // USER CODE END
078
079
080
081 //*****
082 // @Global Variables
083 //*****
084
085 // USER CODE BEGIN (MAIN_Header,7)
086 extern volatile int RS232_wait;
087
088 extern unsigned int T13_values[];
089 extern unsigned int length_of_a_whole_note;
090 extern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

MAIN.C | CC2.C | myprin... | MAIN.H

Simulation

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

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



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

from:

```
if(RS232_wait)
    RS232_wait--;

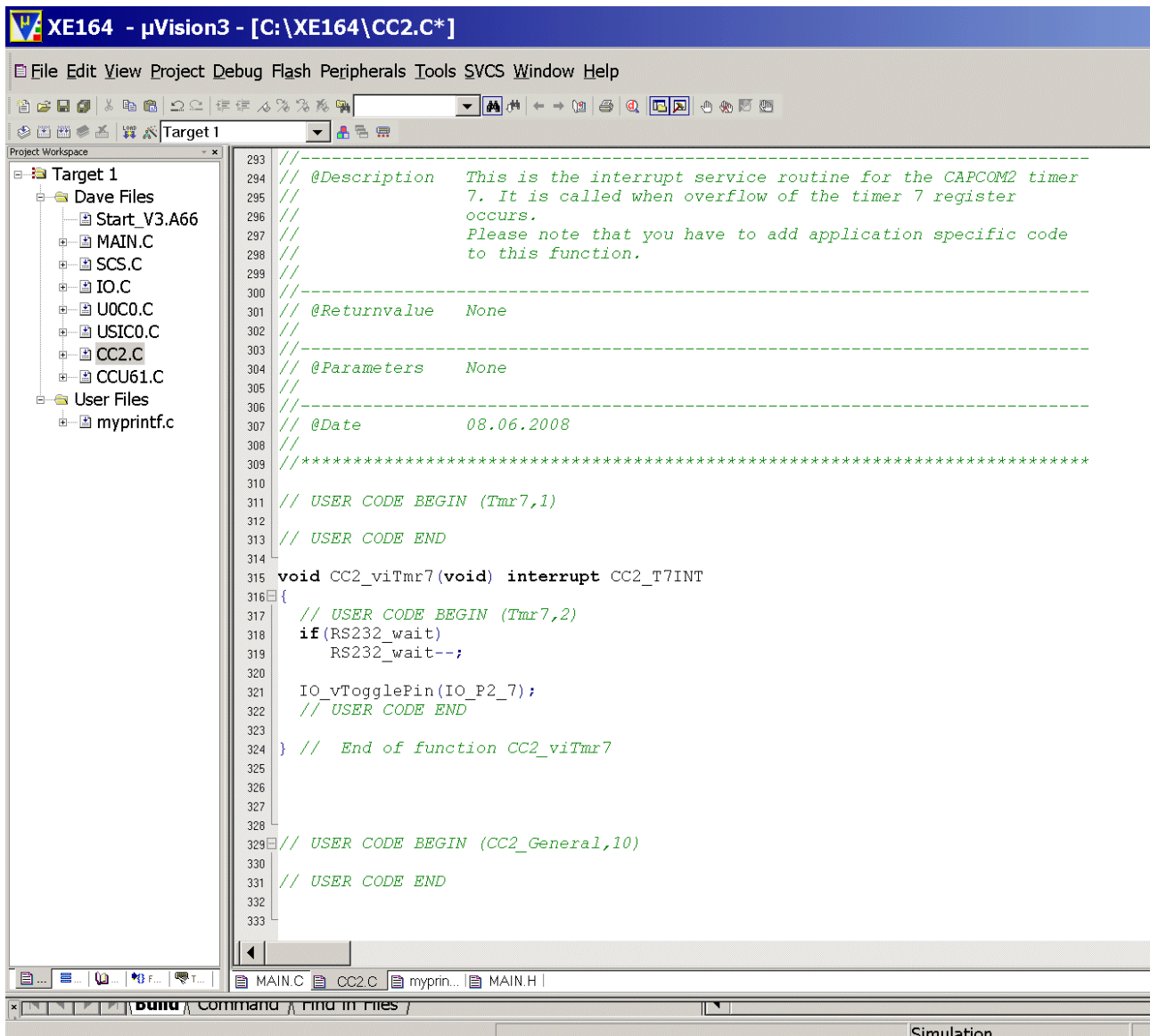
if (blinking)
{
    IO_vTogglePin(IO_P2_7);
}
```

to:

```
if(RS232_wait)
    RS232_wait--;

IO_vTogglePin(IO_P2_7);
```





```

293 //-----
294 // @Description  This is the interrupt service routine for the CAPCOM2 timer
295 //               7. It is called when overflow of the timer 7 register
296 //               occurs.
297 //               Please note that you have to add application specific code
298 //               to this function.
299 //-----
300 // @Returnvalue  None
301 //
302 //-----
303 // @Parameters  None
304 //
305 //-----
306 // @Date        08.06.2008
307 //
308 //*****
309 //
310 // USER CODE BEGIN (Tmr7,1)
311 // USER CODE END
312
313 void CC2_viTmr7(void) interrupt CC2_T7INT
314 {
315     // USER CODE BEGIN (Tmr7,2)
316     if(RS232_wait)
317         RS232_wait--;
318
319     IO_vTogglePin(IO_P2_7);
320     // USER CODE END
321 } // End of function CC2_viTmr7
322
323 // USER CODE BEGIN (CC2_General,10)
324 // USER CODE END
325
326
327
328
329
330
331
332
333

```

Remember:

Port\_2 pins used as GPIO:



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



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

```

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

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

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

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

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

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

    sprintf(mb2," T12-pv=%5u (%5u)",current_note_length,CCU61_T12PR);
    myprintf(mb2);
}

```

```

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

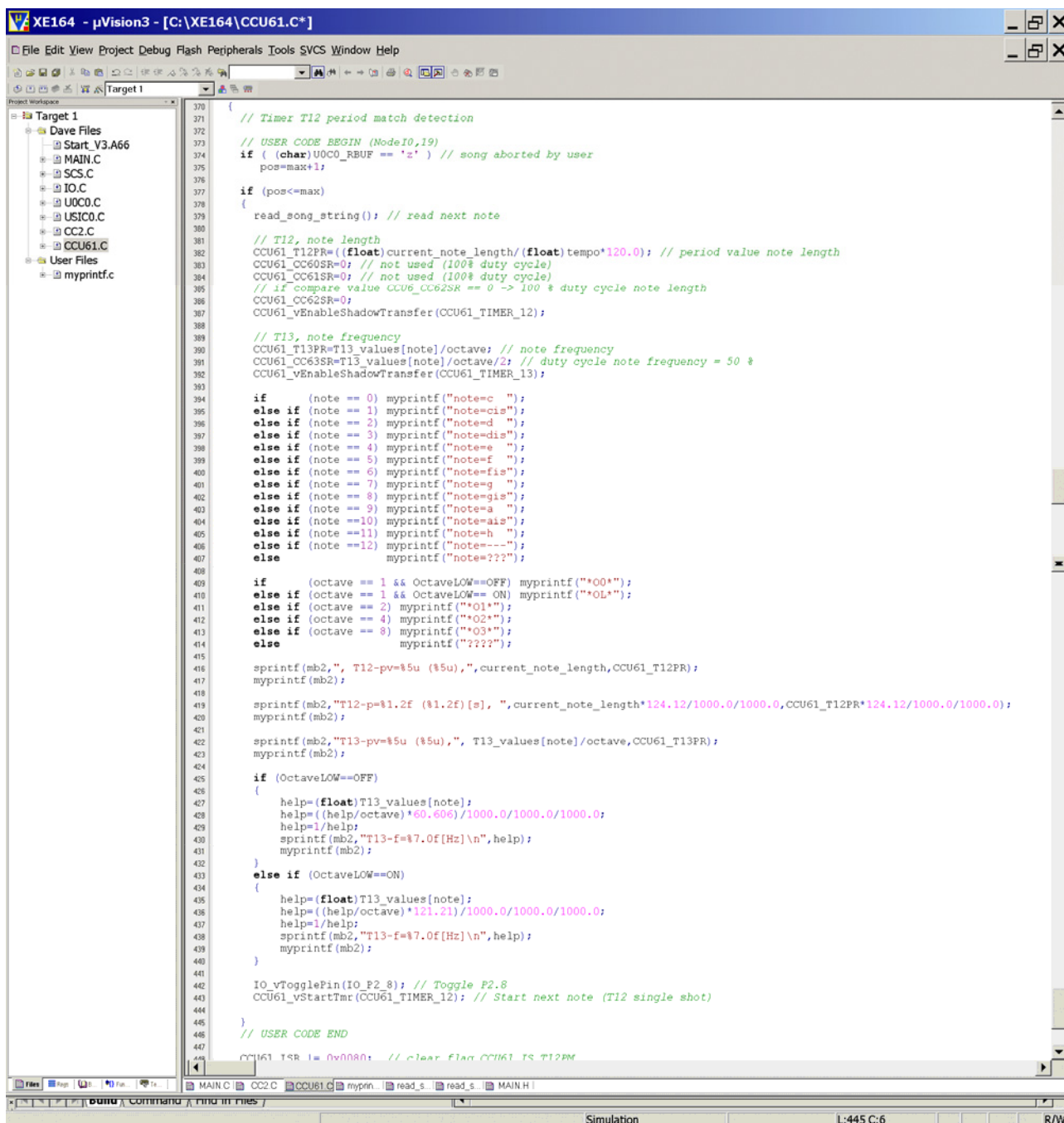
        sprintf(mb2, "T13-pv=%5u (%5u)", T13_values[note]/octave,CCU61_T13PR);
        myprintf(mb2);

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

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

    }

```



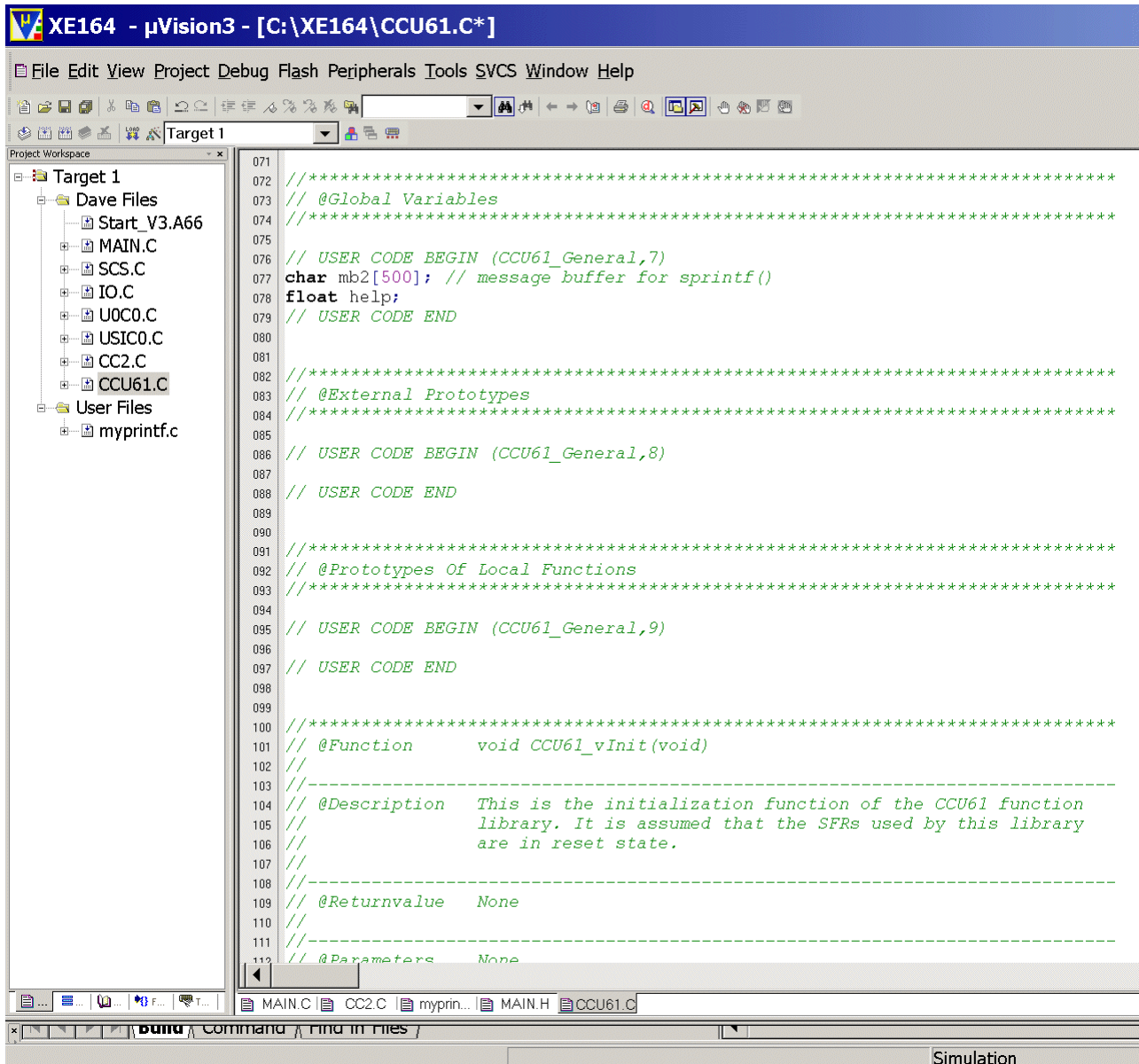
```

370 // Timer T12 period match detection
371
372 // USER CODE BEGIN (NodeI0,19)
373 if ( (char)U0C0_RBUF == 'z' ) // song aborted by user
374     pos=max+1;
375
376 if (pos<=max)
377 {
378     read_song_string(); // read next note
379
380     // T12, note length
381     CCU61_T12PR=((float)current_note_length/(float)tempo*120.0); // period value note length
382     CCU61_CC60SR=0; // not used (100% duty cycle)
383     CCU61_CC61SR=0; // not used (100% duty cycle)
384     // if compare value CCU6_CC62SR == 0 -> 100 % duty cycle note length
385     CCU61_CC62SR=0;
386     CCU61_vEnableShadowTransfer(CCU61_TIMER_12);
387
388     // T13, note frequency
389     CCU61_T13PR=T13_values[note]/octave; // note frequency
390     CCU61_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %
391     CCU61_vEnableShadowTransfer(CCU61_TIMER_13);
392
393     if (note == 0) myprintf("note=c ");
394     else if (note == 1) myprintf("note=cis");
395     else if (note == 2) myprintf("note=d ");
396     else if (note == 3) myprintf("note=dis");
397     else if (note == 4) myprintf("note=e ");
398     else if (note == 5) myprintf("note=f ");
399     else if (note == 6) myprintf("note=fis");
400     else if (note == 7) myprintf("note=g ");
401     else if (note == 8) myprintf("note=gis");
402     else if (note == 9) myprintf("note=a ");
403     else if (note ==10) myprintf("note=ais");
404     else if (note ==11) myprintf("note=h ");
405     else if (note ==12) myprintf("note=---");
406     else myprintf("note=???");
407
408     if (octave == 1 && OctaveLOW==OFF) myprintf("O0");
409     else if (octave == 1 && OctaveLOW==ON) myprintf("O1");
410     else if (octave == 2) myprintf("O2");
411     else if (octave == 4) myprintf("O4");
412     else if (octave == 8) myprintf("O8");
413     else myprintf("O?");
414
415     sprintf(mb2, "T12-pv=%5u (%5u)", current_note_length, CCU61_T12PR);
416     myprintf(mb2);
417
418     sprintf(mb2, "T12-p=%1.2f (%1.2f) [s]", current_note_length*124.12/1000.0/1000.0, CCU61_T12PR*124.12/1000.0/1000.0);
419     myprintf(mb2);
420
421     sprintf(mb2, "T13-pv=%5u (%5u)", T13_values[note]/octave, CCU61_T13PR);
422     myprintf(mb2);
423
424     if (OctaveLOW==OFF)
425     {
426         help=(float)T13_values[note];
427         help=(help/octave)*60.606/1000.0/1000.0/1000.0;
428         help=1/help;
429         sprintf(mb2, "T13-f=%7.0f[Hz]\n", help);
430         myprintf(mb2);
431     }
432     else if (OctaveLOW==ON)
433     {
434         help=(float)T13_values[note];
435         help=(help/octave)*121.21/1000.0/1000.0/1000.0;
436         help=1/help;
437         sprintf(mb2, "T13-f=%7.0f[Hz]\n", help);
438         myprintf(mb2);
439     }
440
441     IO_vTogglePin(IO_P2_8); // Toggle P2.8
442     CCU61_vStartTmr(CCU61_TIMER_12); // Start next note (T12 single shot)
443 }
444 // USER CODE END
445
446 CCU61_ISR |= 0x0080; // clear flag CCU61 IS T12PM
  
```



Double click CCU61.C insert Global Variables:

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



The screenshot shows the Infineon µVision3 IDE interface. The title bar reads "XE164 - µVision3 - [C:\XE164\CCU61.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 structure with "Target 1" containing "Dave Files" (Start\_V3.A66, MAIN.C, SCS.C, IO.C, U0C0.C, USIC0.C, CC2.C, CCU61.C) and "User Files" (myprintf.c). The main editor window displays the code for CCU61.C, with line numbers 071 to 112 visible. The code includes global variable declarations and function prototypes. The inserted code is highlighted in green.

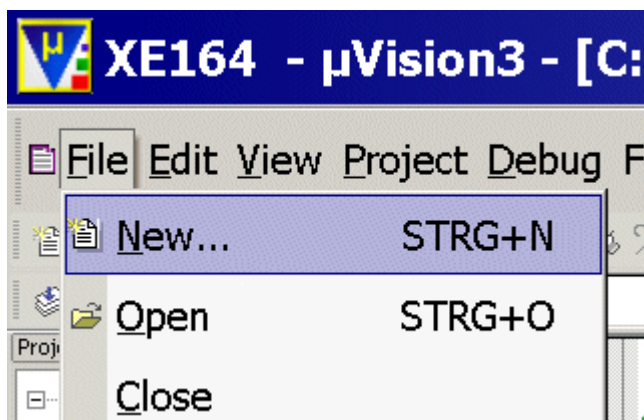
```

071 //*****
072 // @Global Variables
073 //*****
074 // USER CODE BEGIN (CCU61_General,7)
075 char mb2[500]; // message buffer for sprintf()
076 float help;
077 // USER CODE END
078
079 //*****
080 // @External Prototypes
081 //*****
082 // USER CODE BEGIN (CCU61_General,8)
083 // USER CODE END
084
085 //*****
086 // @Prototypes Of Local Functions
087 //*****
088 // USER CODE BEGIN (CCU61_General,9)
089 // USER CODE END
090
091 //*****
092 // @Function      void CCU61_vInit(void)
093 //
094 //-----
095 // @Description   This is the initialization function of the CCU61 function
096 //                library. It is assumed that the SFRs used by this library
097 //                are in reset state.
098 //
099 //-----
100 // @Returnvalue   None
101 //
102 //-----
103 // @Parameters    None
104

```

The status bar at the bottom shows "Simulation".

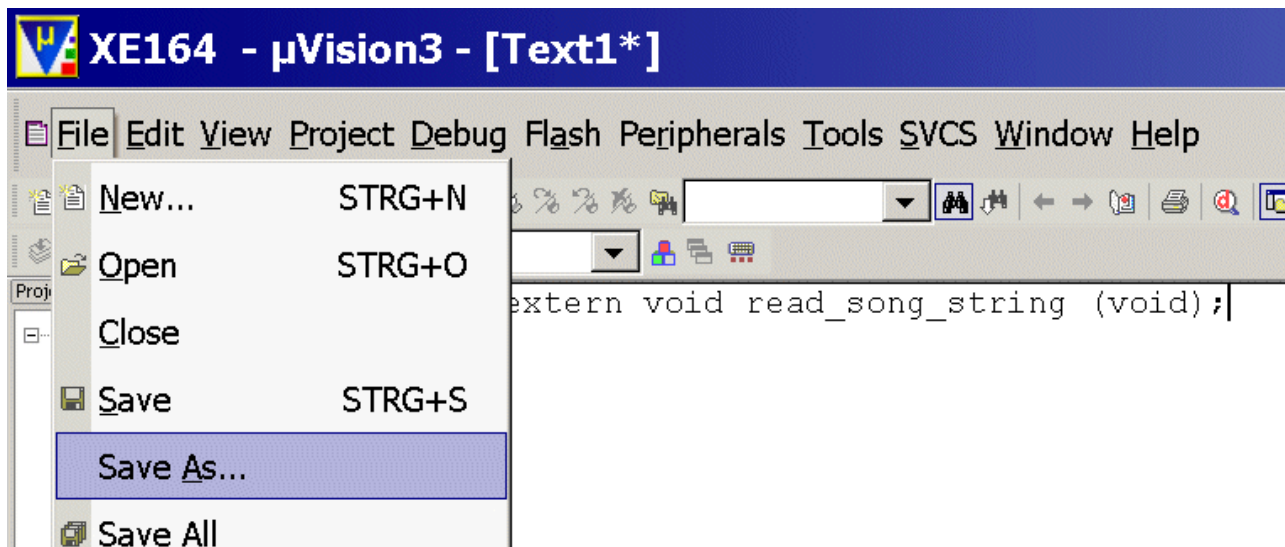
File – New



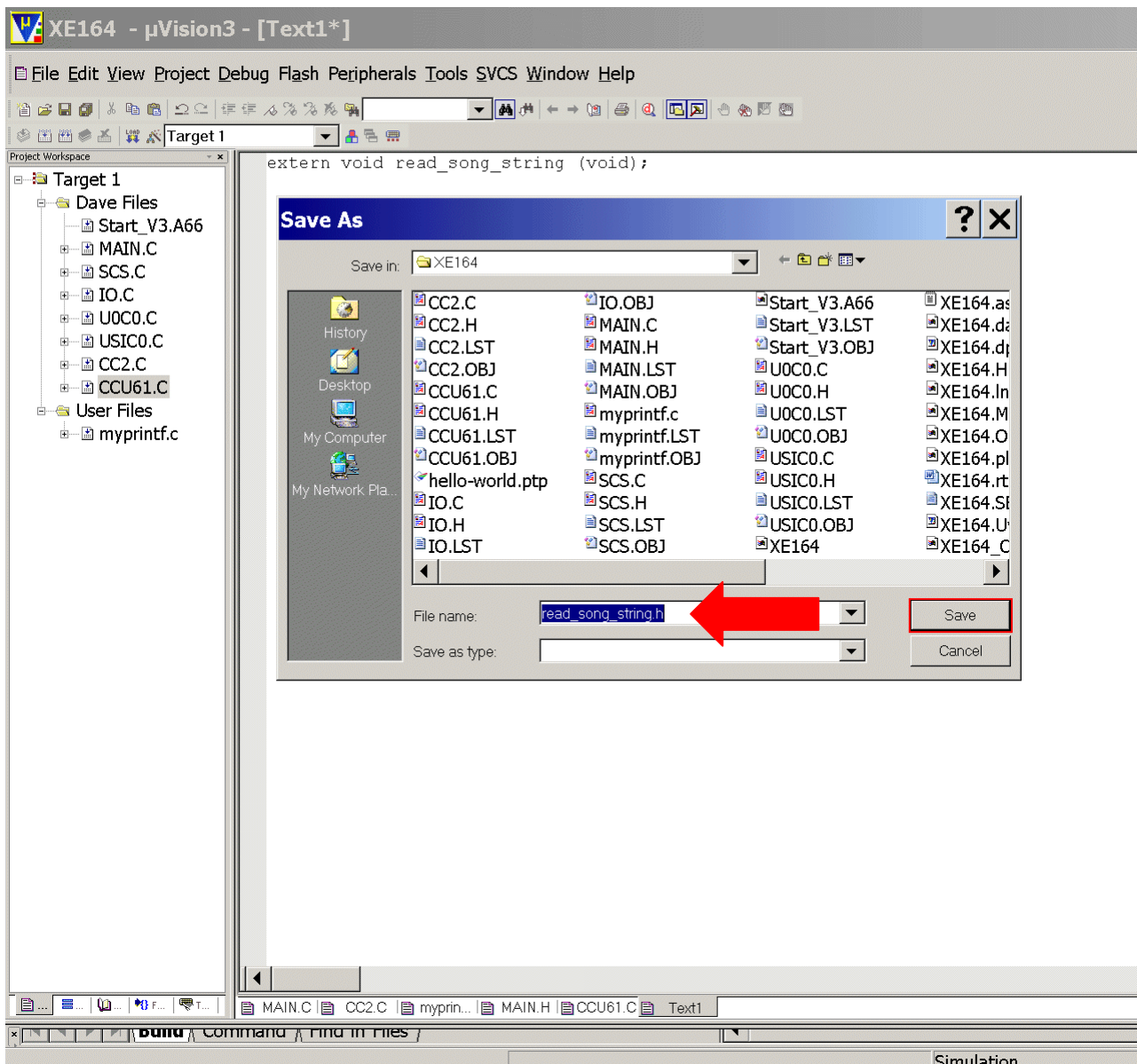
Insert:

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

File – Save As...

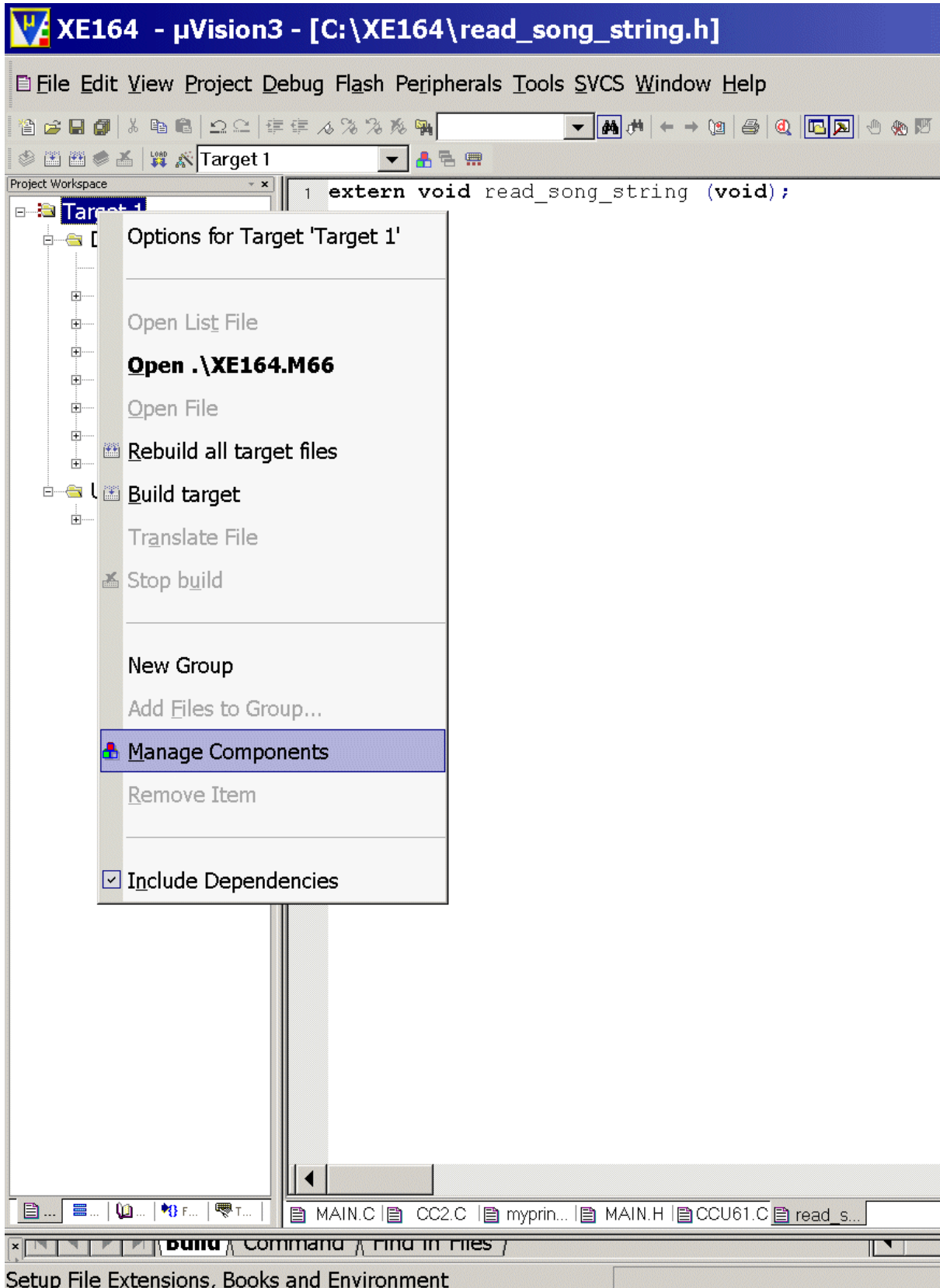


Insert: read\_song\_string.h



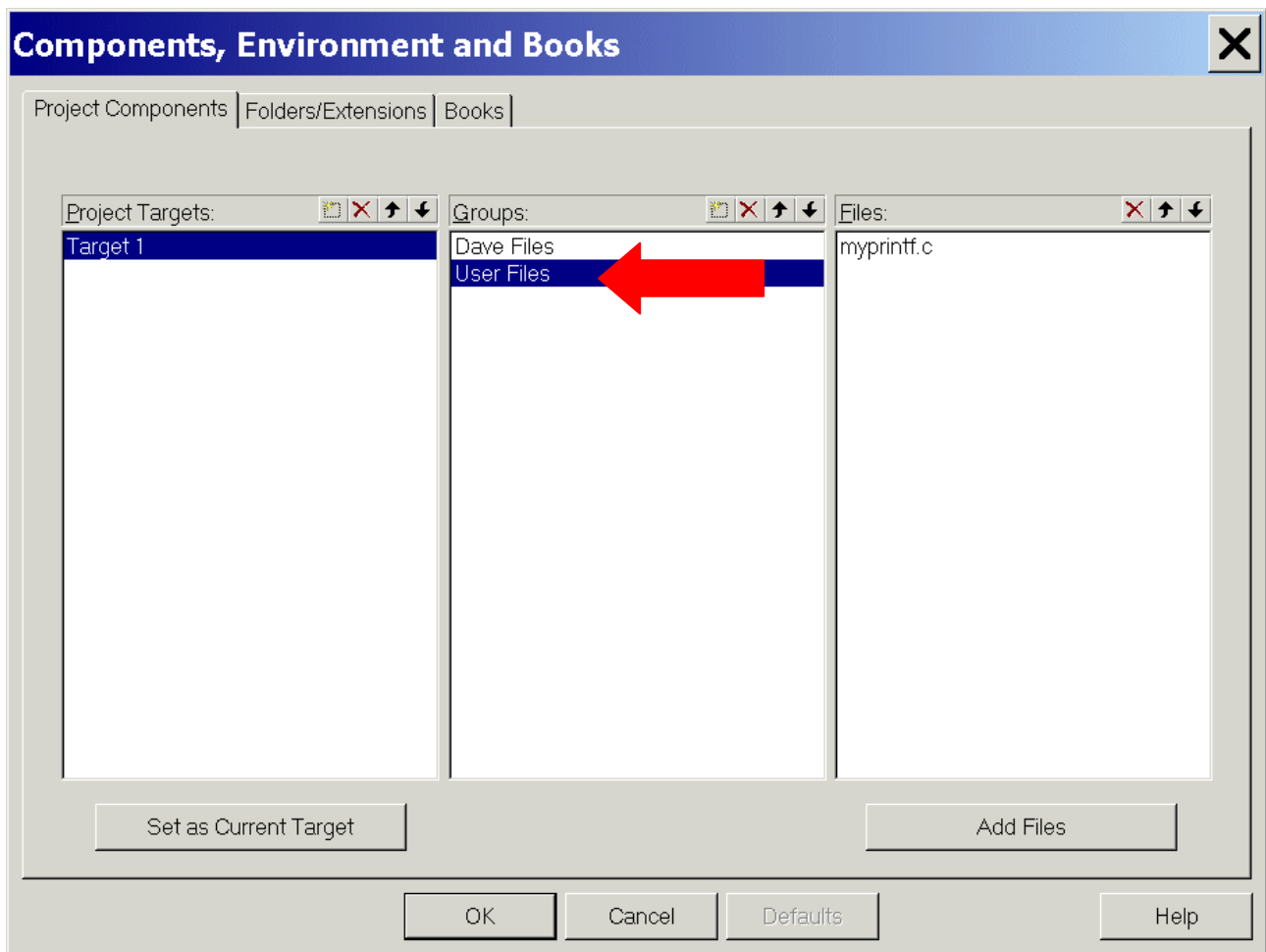
Click Save

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

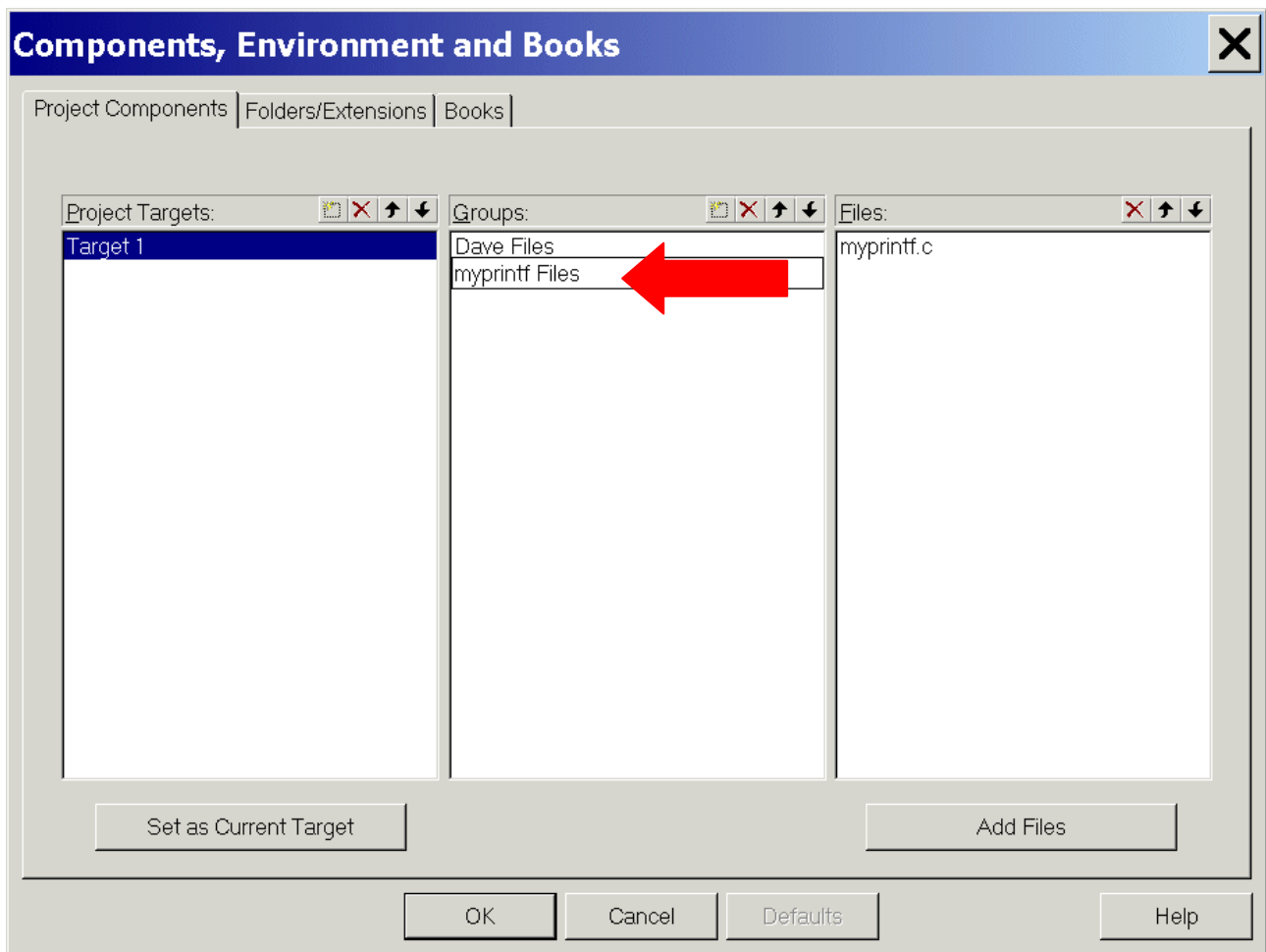




Double click User Files:

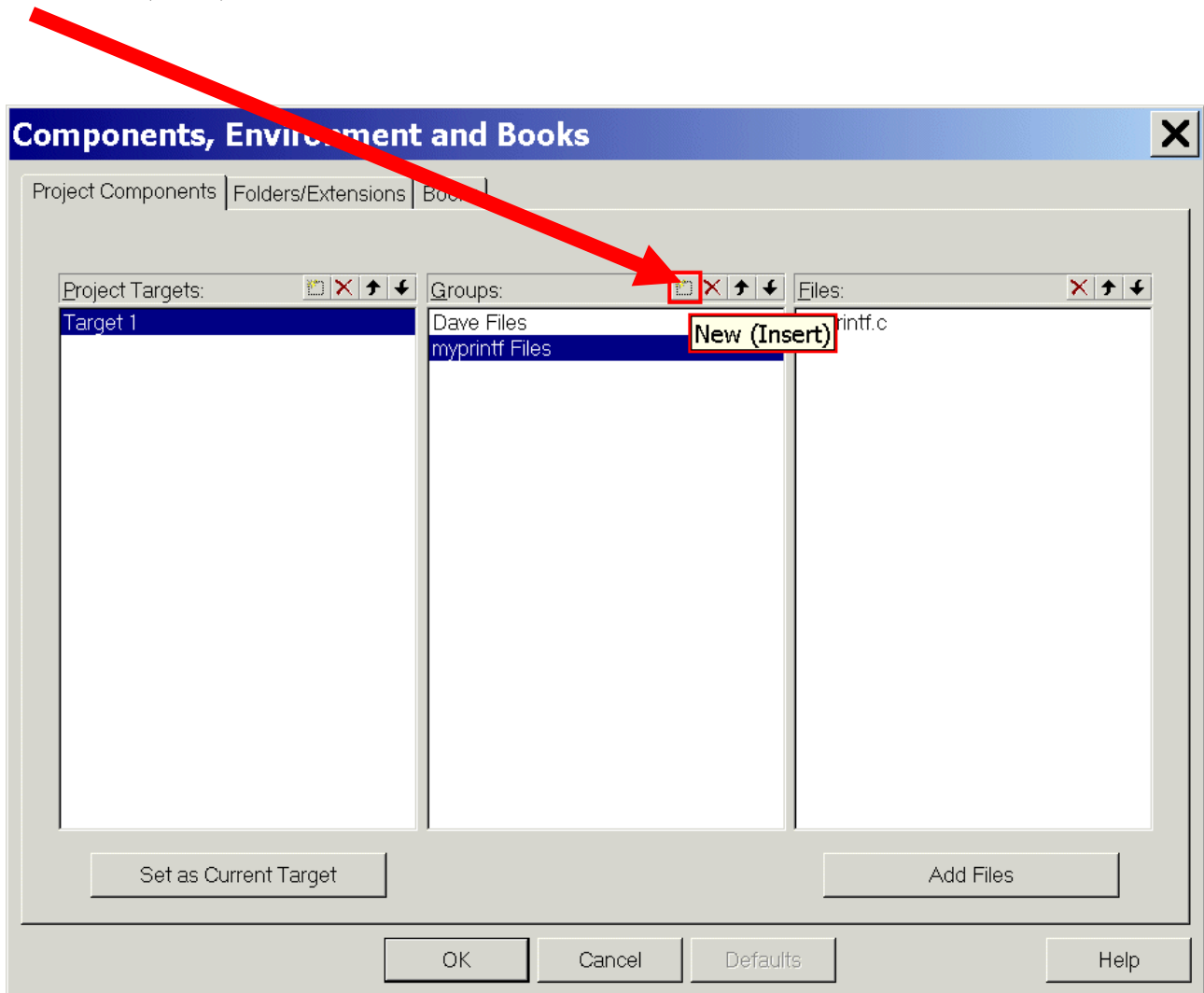


**Insert:** myprintf Files:

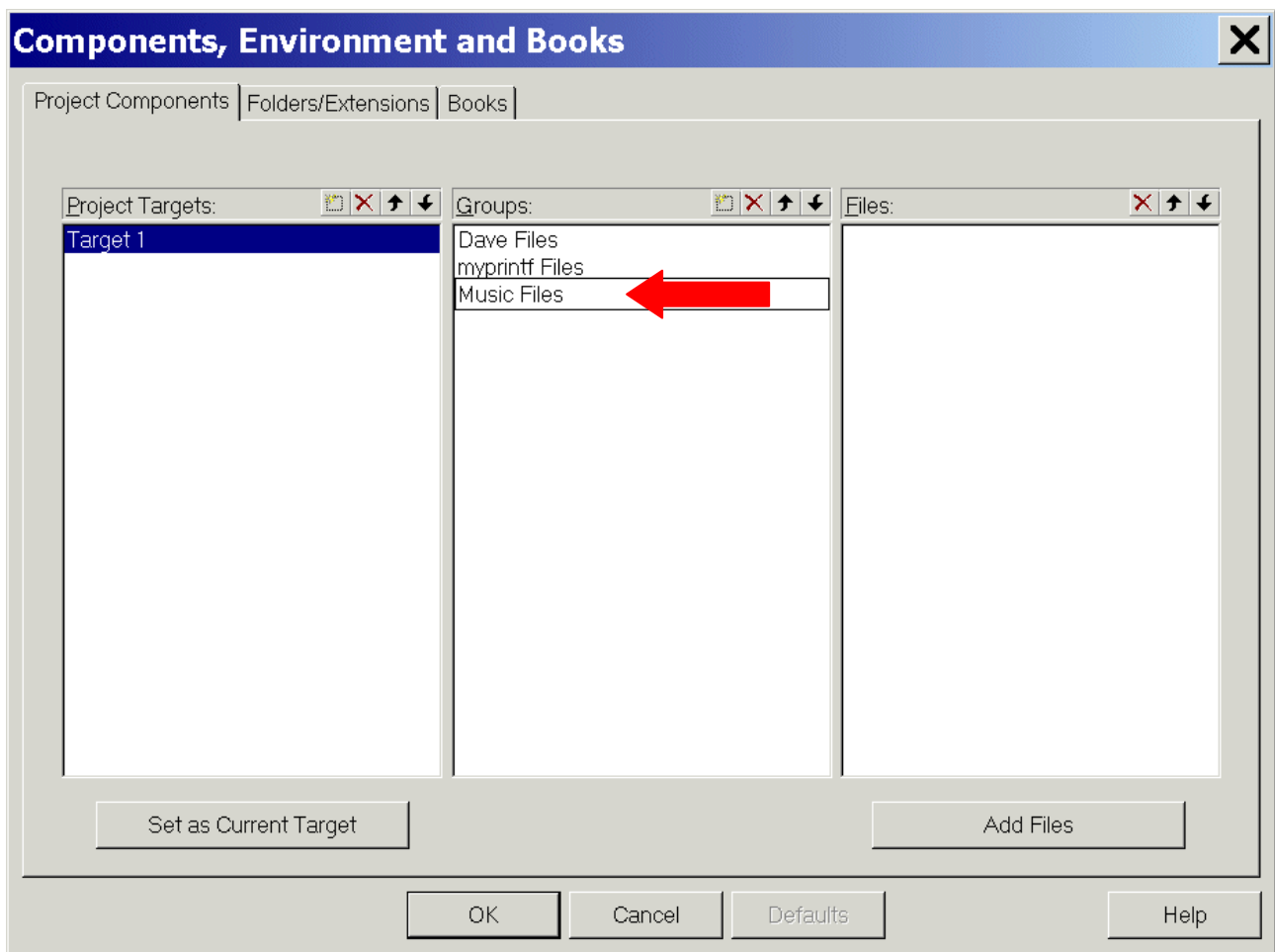


**<ENTER>**

Click New (Insert):



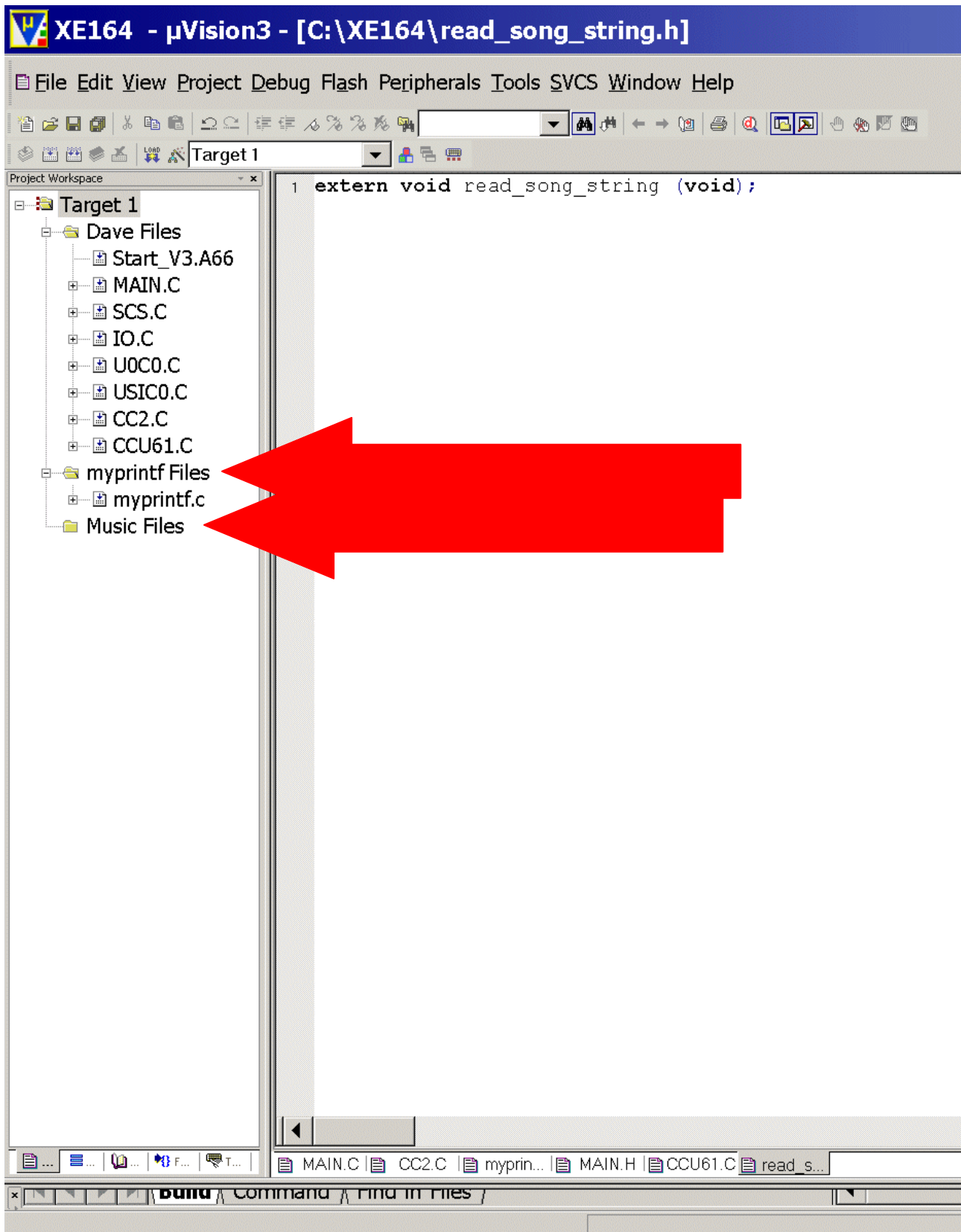
**Insert** Music Files:



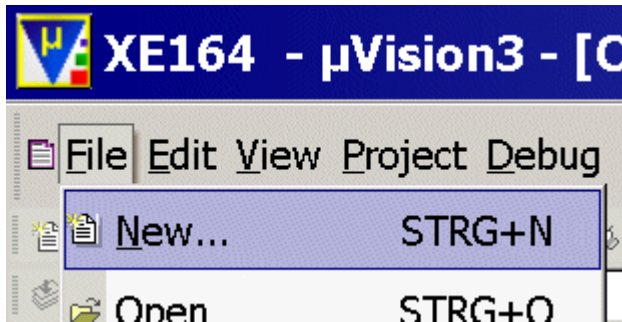
<ENTER>

OK





File – New



Insert:

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

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

    CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13

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

    CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
}

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

    CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13

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

    CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
}

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

    switch (song[pos])
```

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

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

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

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

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

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

```

    default : ;      break;
}                    break;

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

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

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

```



```

        }
        note=12;
        pos++;
        break;

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

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

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

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

if (pos==max) pos++;

```

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



**XE164 - µVision3 - [C:\XE164\read\_song\_string.c]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - myprintf Files
    - myprintf.c
  - Music Files
    - read\_song\_string.h
    - read\_song\_string.c

```

001 #include "main.h"
002 #include "read_song_string.h"
003
004 void SetOctaveNORMAL(void)
005 {
006     OctaveLOW = OFF; // clear Global Variable
007
008     CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
009
010     CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 4, clear T13CLK
011     CCU61_TCTR0 = CCU61_TCTR0 | 0x0200; // prescaler factor is 4, set T13CLK
012
013     CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
014 }
015
016 void SetOctaveLOW(void)
017 {
018     OctaveLOW = ON; // set Global Variable
019
020     CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
021
022     CCU61_TCTR0 = CCU61_TCTR0 & 0xF8FF; // prescaler factor is 8, clear T13CLK
023     CCU61_TCTR0 = CCU61_TCTR0 | 0x0300; // prescaler factor is 8, set T13CLK
024
025     CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
026 }
027

```

```

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

```



```

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

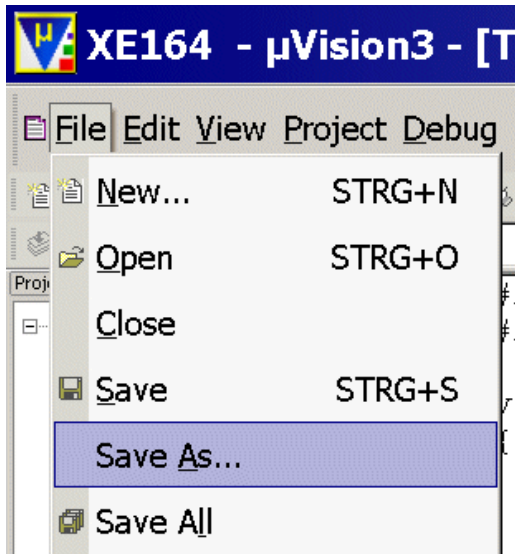
```

```

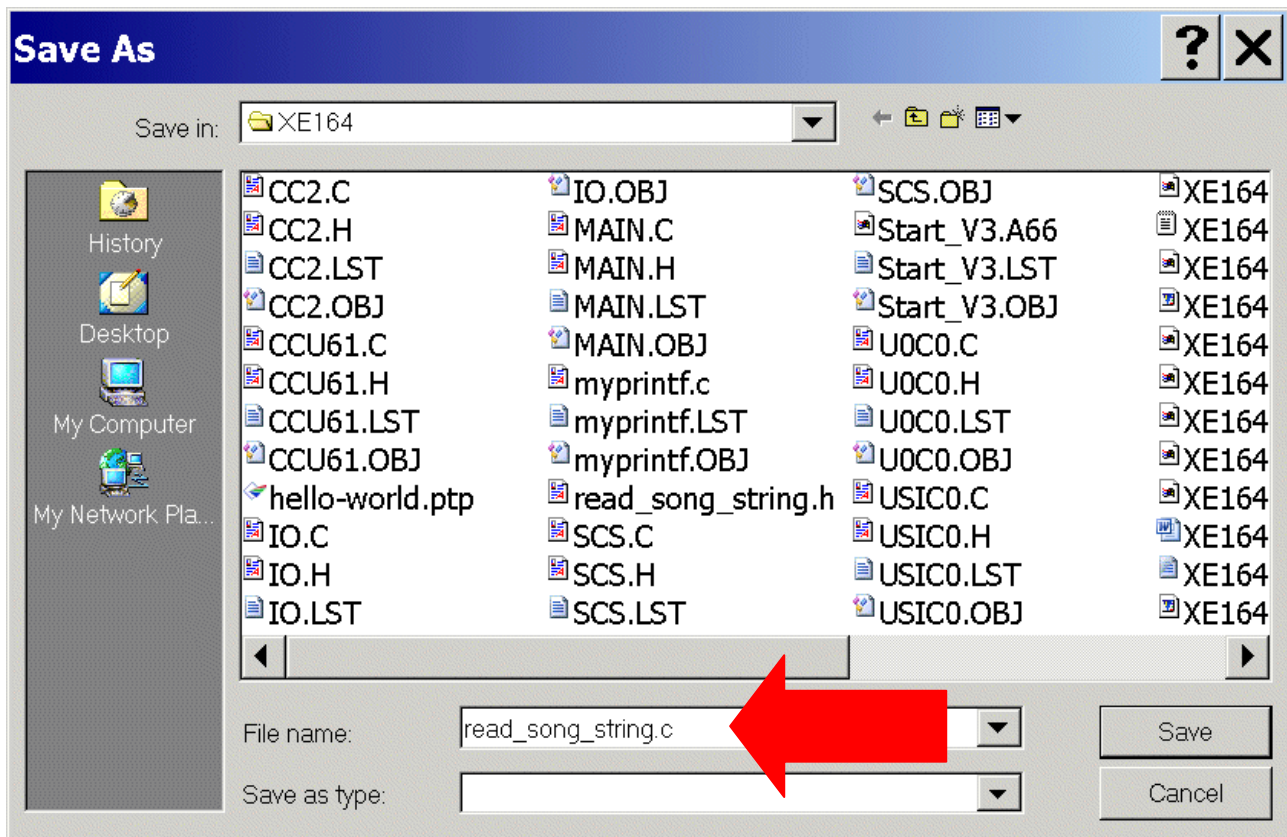
136 // adjust octave:
137 case 'O': switch (song[++pos])
138 {
139     case '0': octave=1; break;
140     case '1': octave=2; break;
141     case '2': octave=4; break;
142     case '3': octave=8; break;
143     default : if (song[pos]=='L') octave=1, SetOctaveLOW();
144               if (song[pos]=='N')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

```

File – Save As...

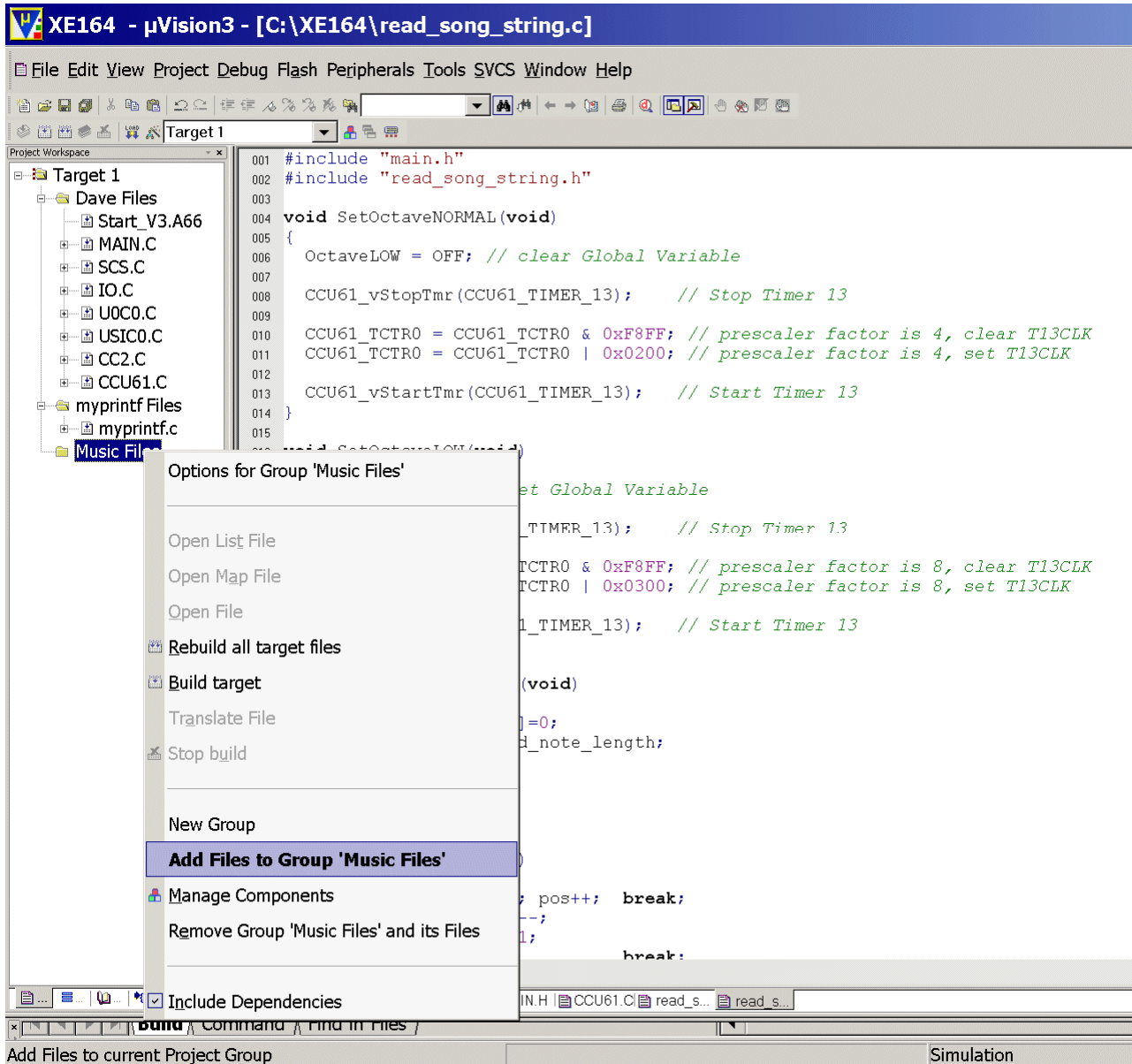


Insert: read\_song\_string.c



Click Save

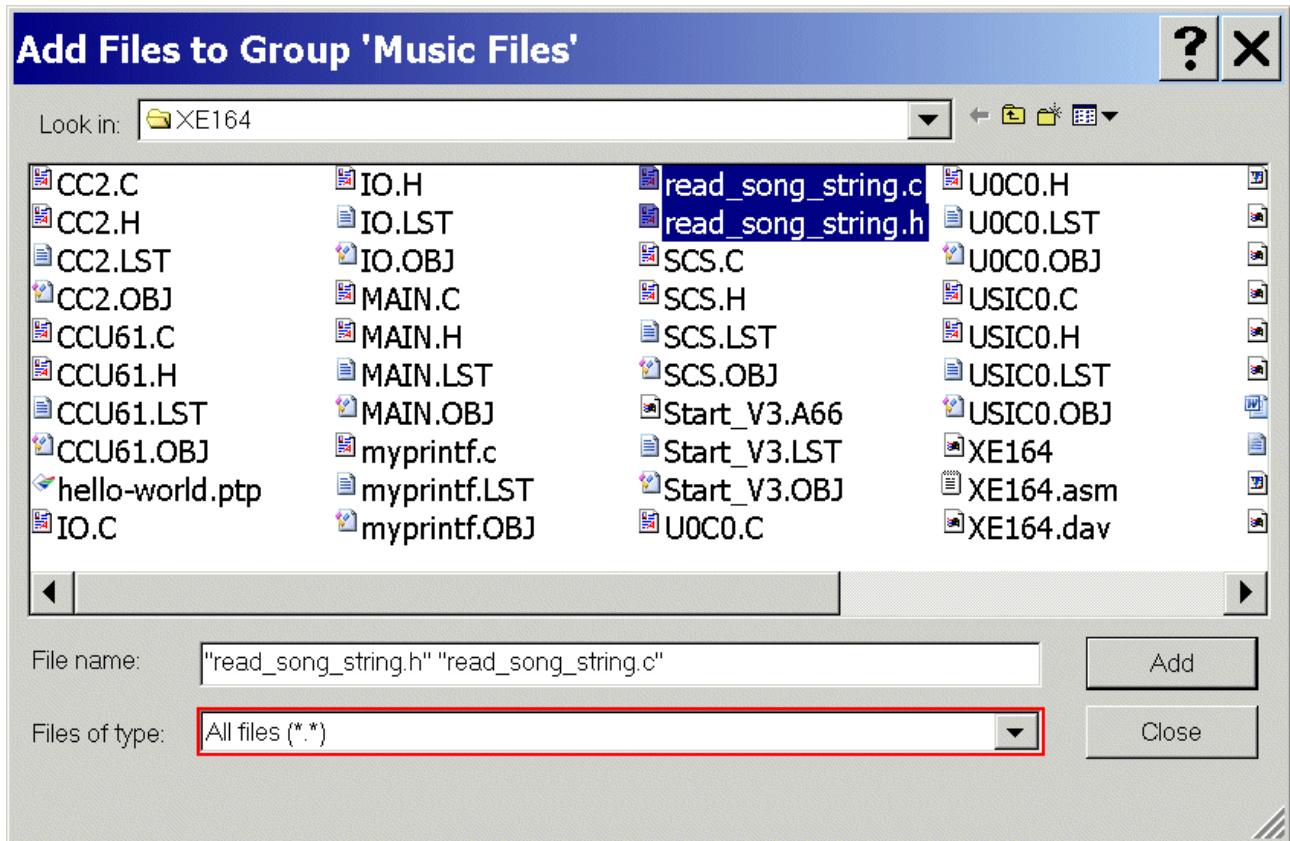
Mouse position: **Project Window**, Music Files: **click right mouse button**  
**Click** Add Files to Group 'Music Files'





Files of type: **select** All files

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



**Click** Add

**Click** Close

XE164 - µVision3 - [C:\XE164\read\_song\_string.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - myprintf Files
    - myprintf.c
  - Music Files
    - read\_song\_string.h
    - read\_song\_string.c

```

001 #include "main.h"
002 #include "read_song_string.h"
003
004 void SetOctaveNORMAL(void)
005 {
006     OctaveLOW = OFF; // clear Global Variable
007
008     CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
009
010     CCU61_TCTRO = CCU61_TCTRO & 0xF8FF; // prescaler factor is 4, clear T13CLK
011     CCU61_TCTRO = CCU61_TCTRO | 0x0200; // prescaler factor is 4, set T13CLK
012
013     CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
014 }
015
016 void SetOctaveLOW(void)
017 {
018     OctaveLOW = ON; // set Global Variable
019
020     CCU61_vStopTmr(CCU61_TIMER_13); // Stop Timer 13
021
022     CCU61_TCTRO = CCU61_TCTRO & 0xF8FF; // prescaler factor is 8, clear T13CLK
023     CCU61_TCTRO = CCU61_TCTRO | 0x0300; // prescaler factor is 8, set T13CLK
024
025     CCU61_vStartTmr(CCU61_TIMER_13); // Start Timer 13
026 }
027
028 void read_song_string (void)
029 {
030     unsigned char substr[4]=0;
031     current_note_length=old_note_length;
032
033     switch (song[pos])
034     {
035         // select note:
036         case 'C': note=0;
037             switch (song[++pos])
038             {
039                 case '+': note++; pos++; break;
040                 case '-': octave--;
041                     note=11;
042                     pos++; break;

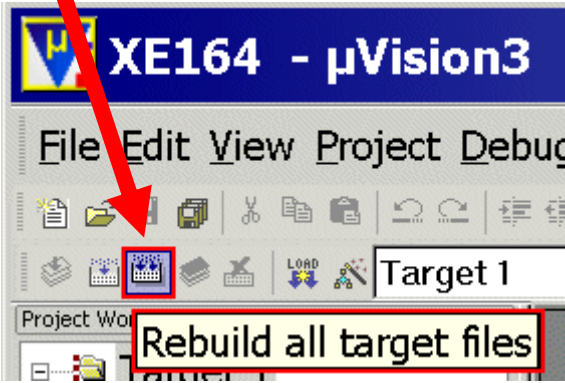
```

Files Regs B... Fun... Te...

MAIN.C CC2.C myprin... MAIN.H CCU61.C read\_s... read\_s...

Simulation

Generate your application program:

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

**XE164 - µVision3 - [C:\XE164\MAIN.C]**

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Target 1

Project Workspace

- Target 1
  - Dave Files
    - Start\_V3.A66
    - MAIN.C
    - SCS.C
    - IO.C
    - U0C0.C
    - USIC0.C
    - CC2.C
    - CCU61.C
  - myprintf Files
    - myprintf.c
  - Music Files
    - read\_song\_string.h
    - read\_song\_string.c

```

541 // USER CODE END
542
543 while(1)
544 {
545
546 // USER CODE BEGIN (Main,4)
547 myprintf(menu);
548 select=input();
549
550 switch (select)
551 {
552     case 'a': ++next_song_a, play_song(); break;
553     case 'b': ++next_song_b, play_song(); break;
554     case 'c': ++next_song_c, play_song(); break;
555     case 'd': ++next_song_d, play_song(); break;
556     case 'e': ++next_song_e, play_song(); break;
557     case 'f': ++next_song_f, play_song(); break;
558     case 'g': ++next_song_g, play_song(); break;
559     case 'h': ++next_song_h, play_song(); break;
560     case 'i': ++next_song_i, play_song(); break;
561     case 'j': ++next_song_j, play_song(); break;
562     case 'k': ++next_song_k, play_song(); break;
563     case 'l': ++next_song_l, play_song(); break;
564     case 'm': ++next_song_m, play_song(); break;
565 }
566 // USER CODE END
567
568 }

```

Files Regs B... Fun... Te...

MAIN.C CC2.C myprin... MAIN.H CCU61.C read\_s... read\_s...

Output Window

```

x Build target 'Target 1'
assembling Start_V3.A66...
compiling MAIN.C...
compiling SCS.C...
compiling IO.C...
compiling U0C0.C...
compiling USIC0.C...
compiling CC2.C...
compiling CCU61.C...
compiling myprintf.c...
compiling read_song_string.c...
linking...
Program Size: data=2663 (near=2663) const=3929 (near=3743) code=6990
creating hex file from "XE164"...
"XE164" - 0 Error(s), 0 Warning(s).

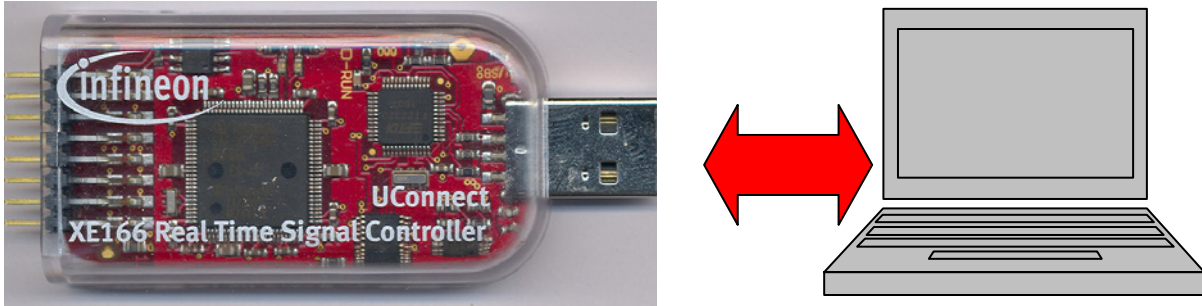
```

Build Command Find in Files

Si



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



#### USB Connection:

- .) used for: UART communication (the USIC0\_CH0/UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
- .) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
- .) the USB connection works also as the power supply.

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

Connect CCU61\_CC62 (P0.2) and GND (VSS) to your active loudspeaker(s):



On-board header X400

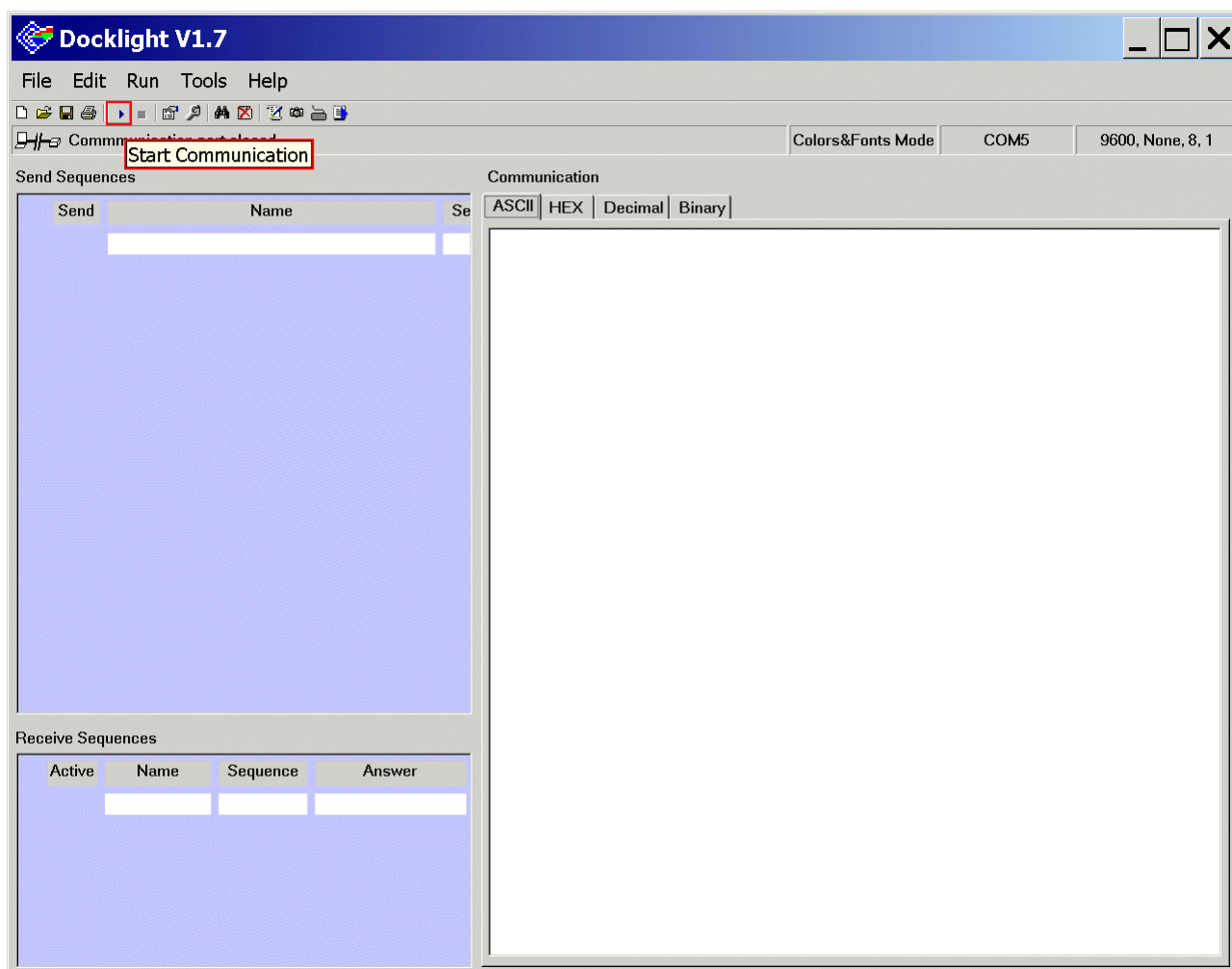


15	13	11	9	7	5	3	1
CANH	P0.4	P0.5	P0.3	P0.2	P5.9	P15.0	GND
CANL	P0.7	P0.6	P0.0	P0.1	P5.8	P5.0	+5V
16	14	12	10	8	6	4	2

Pin number				
1	Ground			
2	+5V			
3	P15.0	ADC1_CH0		
4	P5.0	ADC0_CH0		
5	P5.9	ADC0_CH9	ADC1_CH9	CC2_T7IN CAPCOM2
6	P5.8	ADC0_CH8	ADC8_CH8	T12HRC / T13HRC CCU6x
7	P0.2	U1C0_SCK	CC62 CCU61	TXDC0 CAN0
8	P0.1	U1C0_DOUT	CC61 CCU61	TXDC0 CAN0
9	P0.3	U1C0_SELO	COU60 CCU61	RXDC0B CAN0
10	P0.0	U1C0_DX0	CC60 CCU61	
11	P0.5	U1C1_SCK	COU62 CCU61	
12	P0.6	U1C1_DOUT	COU63 CCU61	TXDC1 CAN1
13	P0.4	U1C1_SELO	COU61 CCU61	RXDC1B CAN1
14	P0.7	U1C1_DX0	U1C1_DX0	CTRAPB CCU61
15	CANH Signal from CAN transceiver			
16	CANL Signal from CAN transceiver			



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

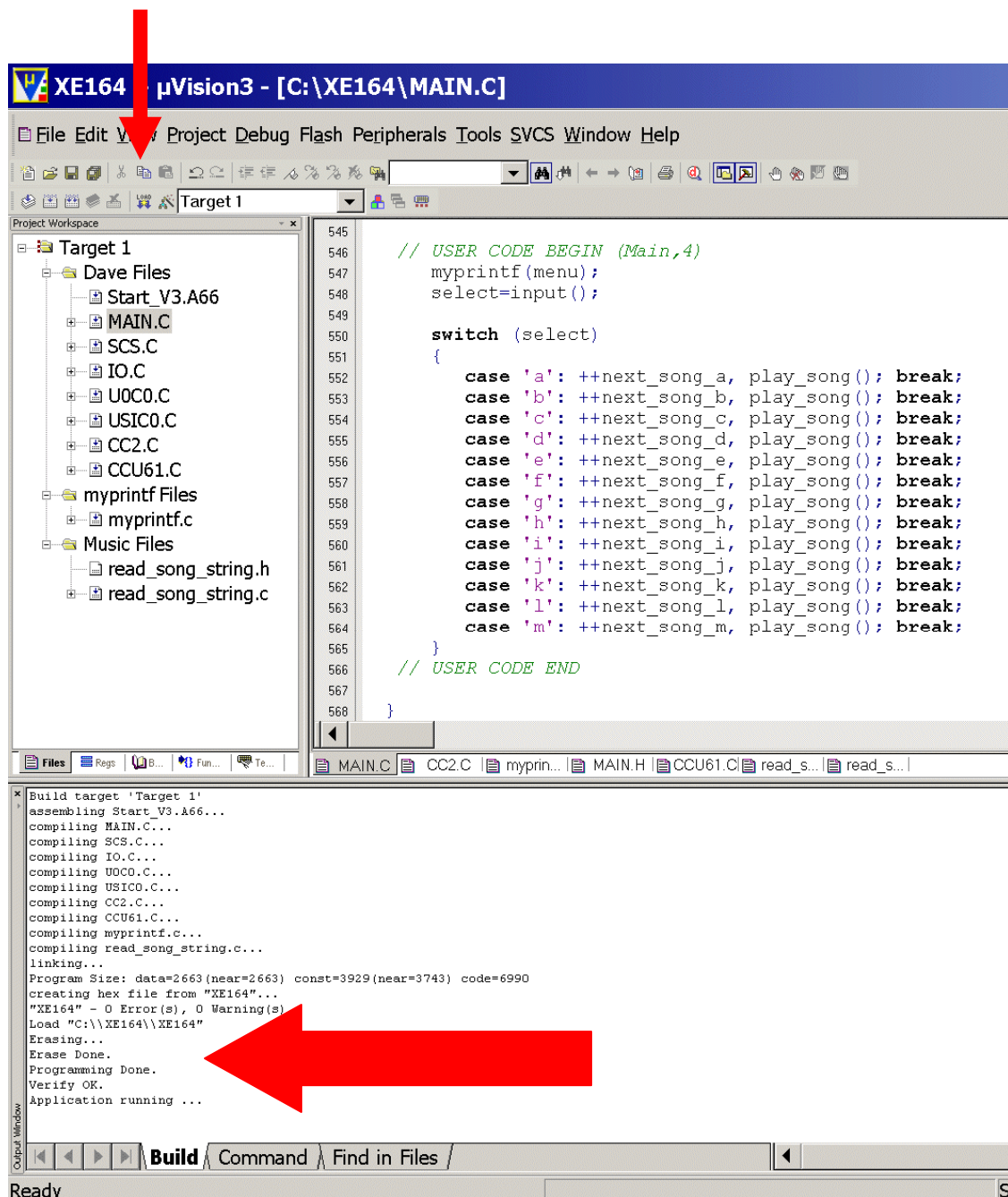




Go to  $\mu$ Vision, [Download/program](#) your application program into the On Chip Flash:



Click:



The screenshot shows the  $\mu$ Vision3 IDE interface. The top menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The Project Workspace on the left shows a tree structure for 'Target 1' with folders for Dave Files, myprintf Files, and Music Files, and individual source files like MAIN.C, SCS.C, IO.C, U0C0.C, USIC0.C, CC2.C, CCU61.C, myprintf.c, read\_song\_string.h, and read\_song\_string.c. The main editor window displays the source code for MAIN.C, which includes a switch statement for playing different songs. The Output Window at the bottom shows the build process, including compiling and linking files, and a red arrow points to the 'Build' button in the toolbar.

```

545 // USER CODE BEGIN (Main,4)
546 myprintf(menu);
547 select=input();
548
549 switch (select)
550 {
551     case 'a': ++next_song_a, play_song(); break;
552     case 'b': ++next_song_b, play_song(); break;
553     case 'c': ++next_song_c, play_song(); break;
554     case 'd': ++next_song_d, play_song(); break;
555     case 'e': ++next_song_e, play_song(); break;
556     case 'f': ++next_song_f, play_song(); break;
557     case 'g': ++next_song_g, play_song(); break;
558     case 'h': ++next_song_h, play_song(); break;
559     case 'i': ++next_song_i, play_song(); break;
560     case 'j': ++next_song_j, play_song(); break;
561     case 'k': ++next_song_k, play_song(); break;
562     case 'l': ++next_song_l, play_song(); break;
563     case 'm': ++next_song_m, play_song(); break;
564 }
565 // USER CODE END
566
567
568 }

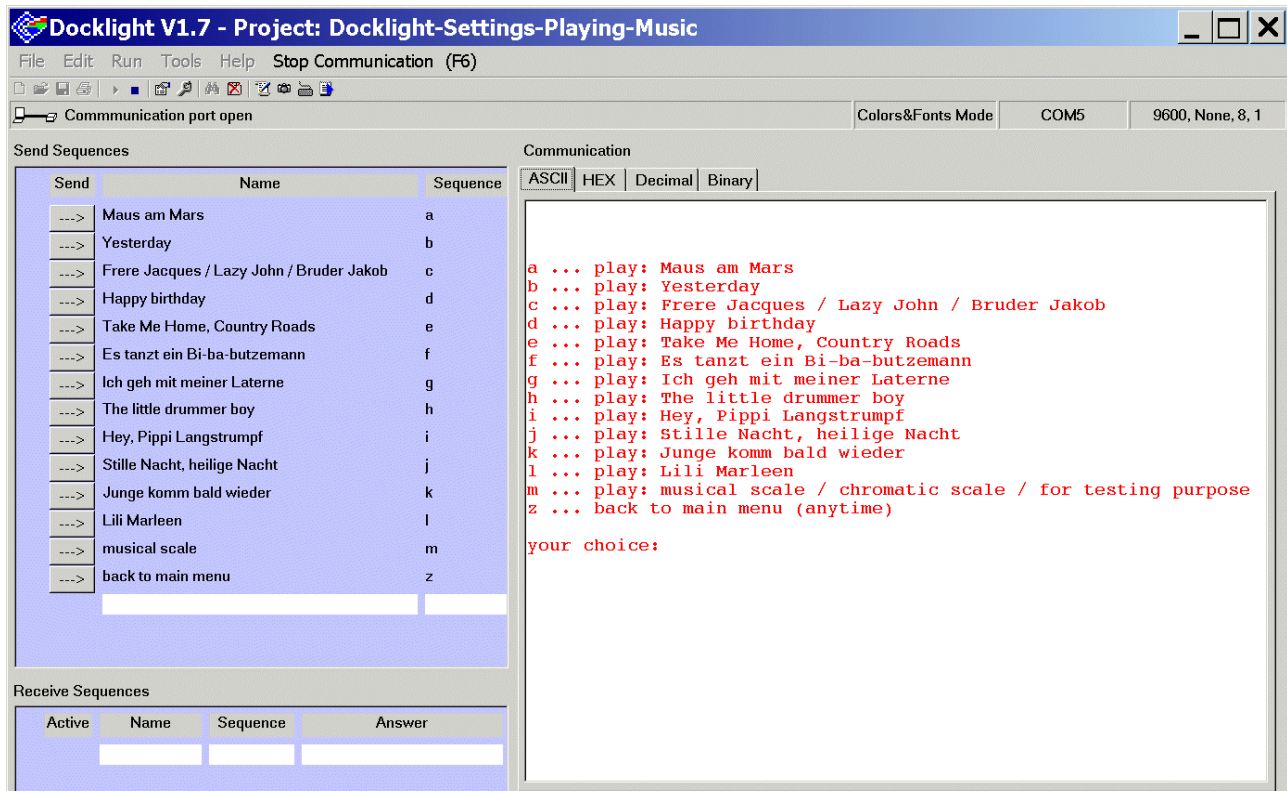
```

Build target 'Target 1'  
 assembling Start\_V3.A66...  
 compiling MAIN.C...  
 compiling SCS.C...  
 compiling IO.C...  
 compiling U0C0.C...  
 compiling USIC0.C...  
 compiling CC2.C...  
 compiling CCU61.C...  
 compiling myprintf.c...  
 compiling read\_song\_string.c...  
 linking...  
 Program Size: data=2663 (near=2663) const=3929 (near=3743) code=6990  
 creating hex file from "XE164"...  
 "XE164" - 0 Error(s), 0 Warning(s)  
 Load "C:\XE164\XE164"  
 Erasing...  
 Erase Done.  
 Programming Done.  
 Verify OK.  
 Application running ...

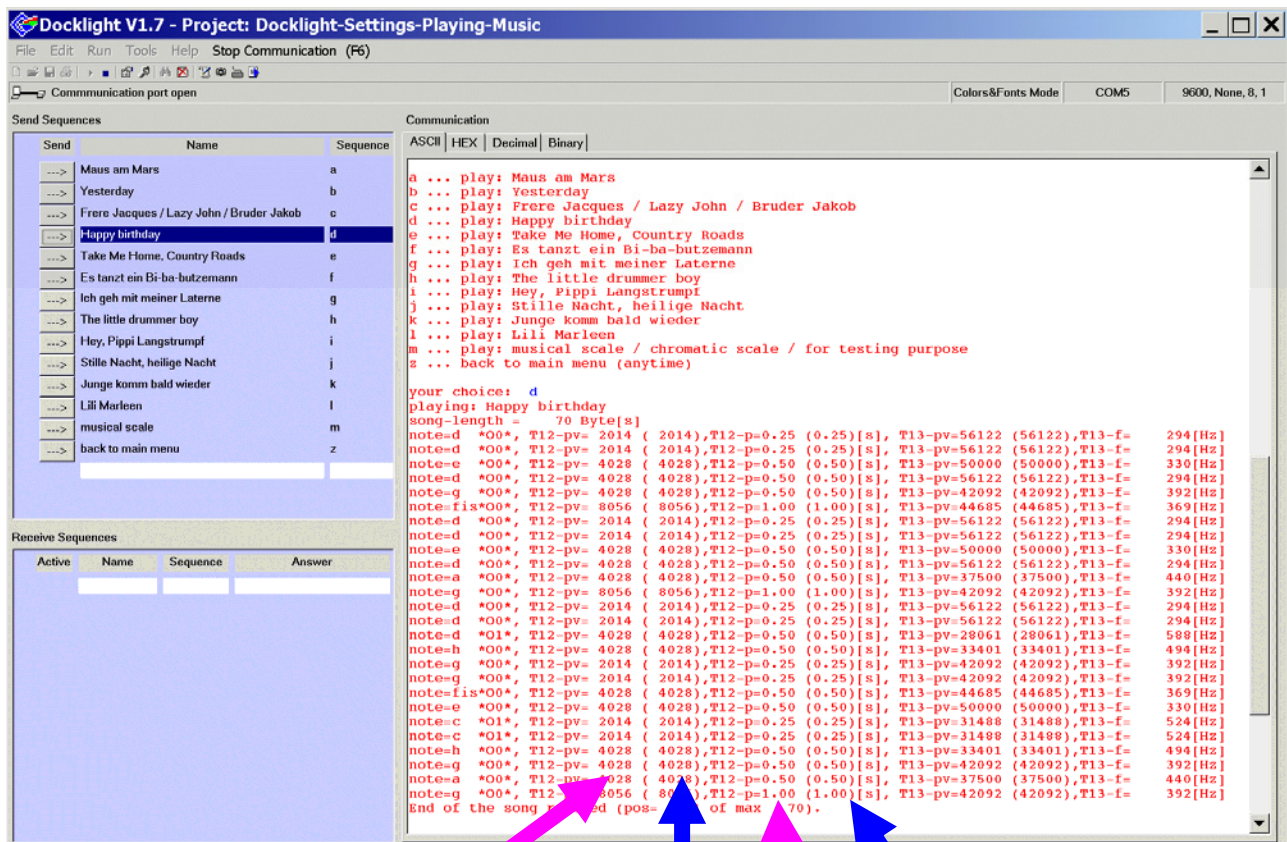




Go to Docklight and see / hear / enjoy the result:



Insert/select d

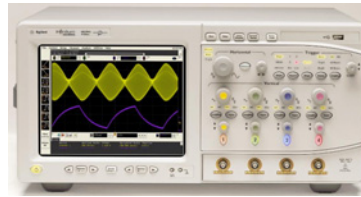


T12 period value  
@ tempo = 120  
beats/minute

T12 period value  
@ current tempo

note length  
@ tempo = 120  
beats/minute

Note length  
@ current tempo



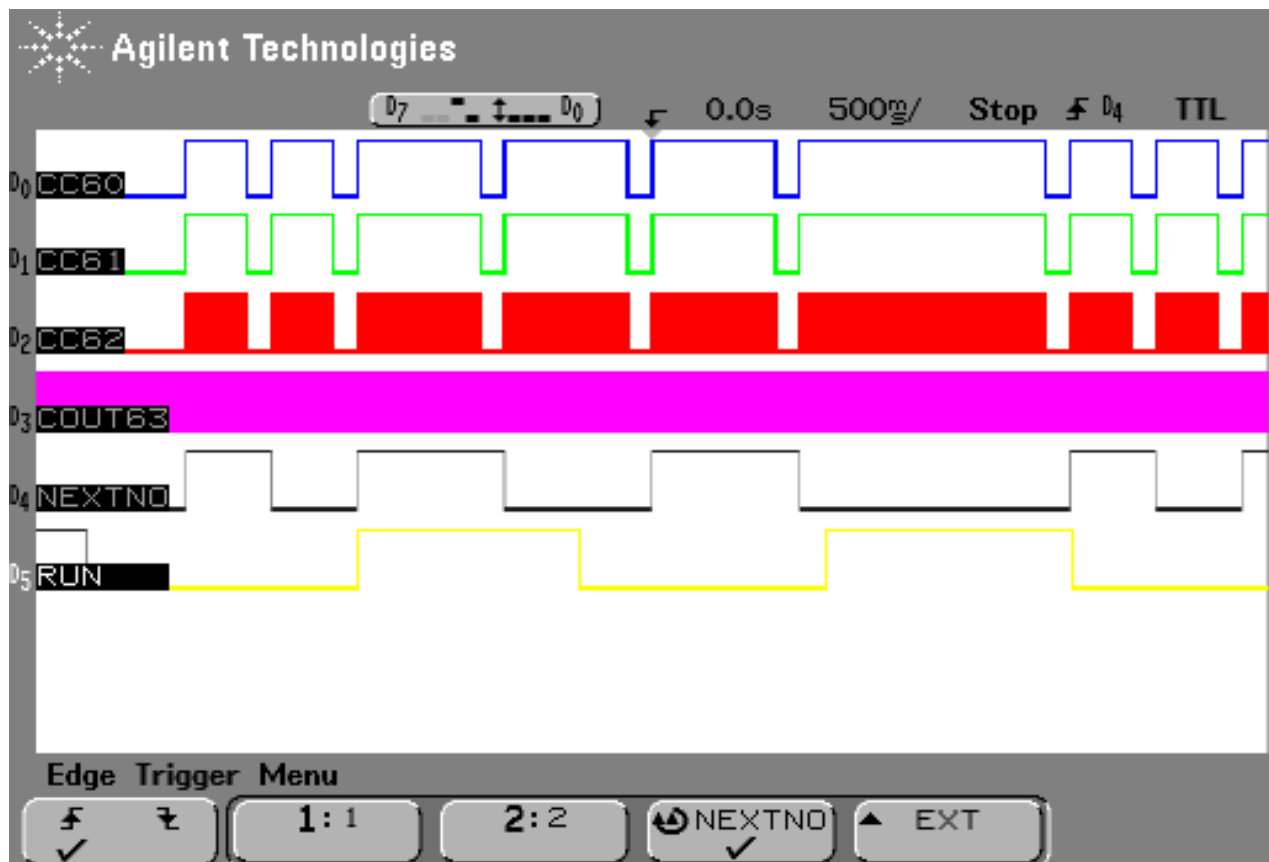
Use any Logic Analyser (LGA) / scope and see the result:

Note:

Song d: Happy birthday:

code unsigned char

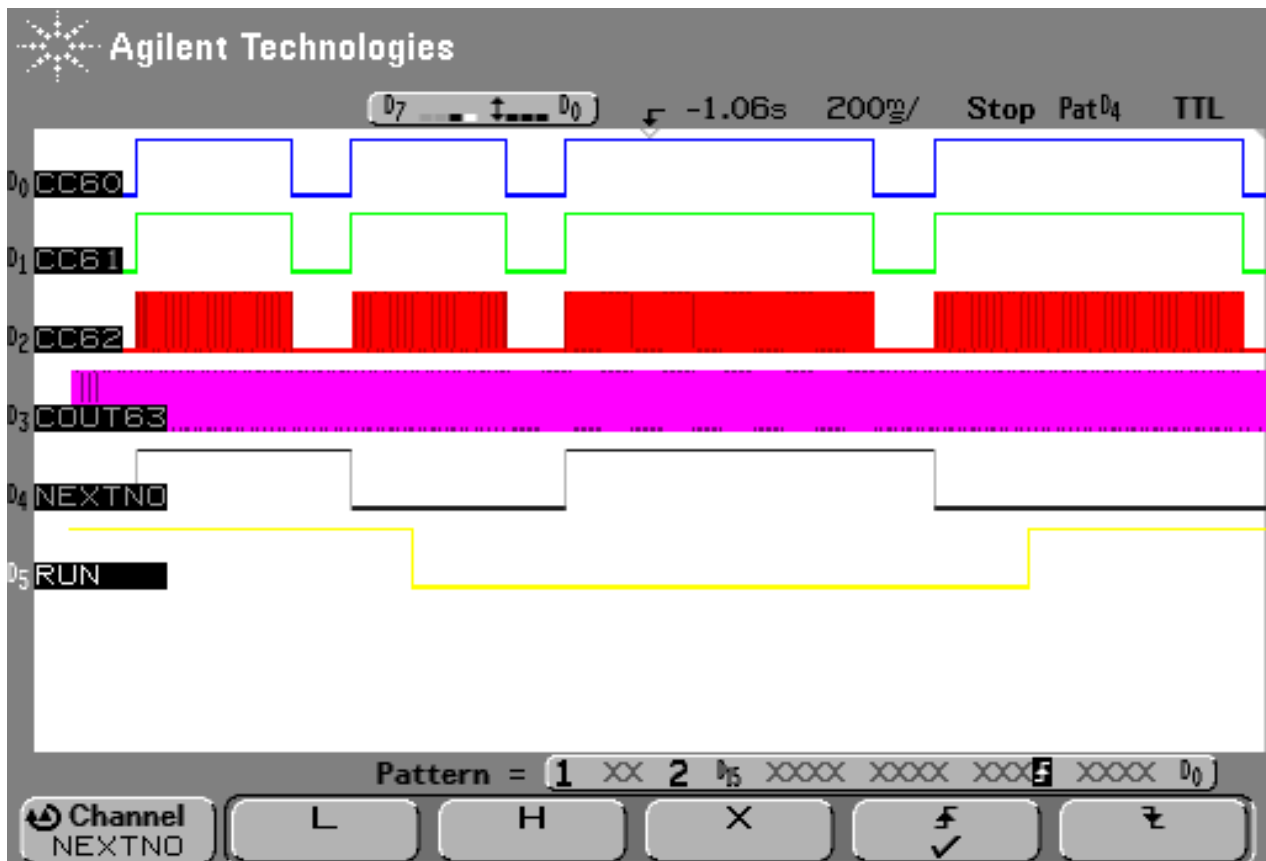
```
songd[]="T120O0L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F+L4EO1L8C  
CO0L4HGAL2G";
```





HAPPY BIRTHDAY =

T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F... ..

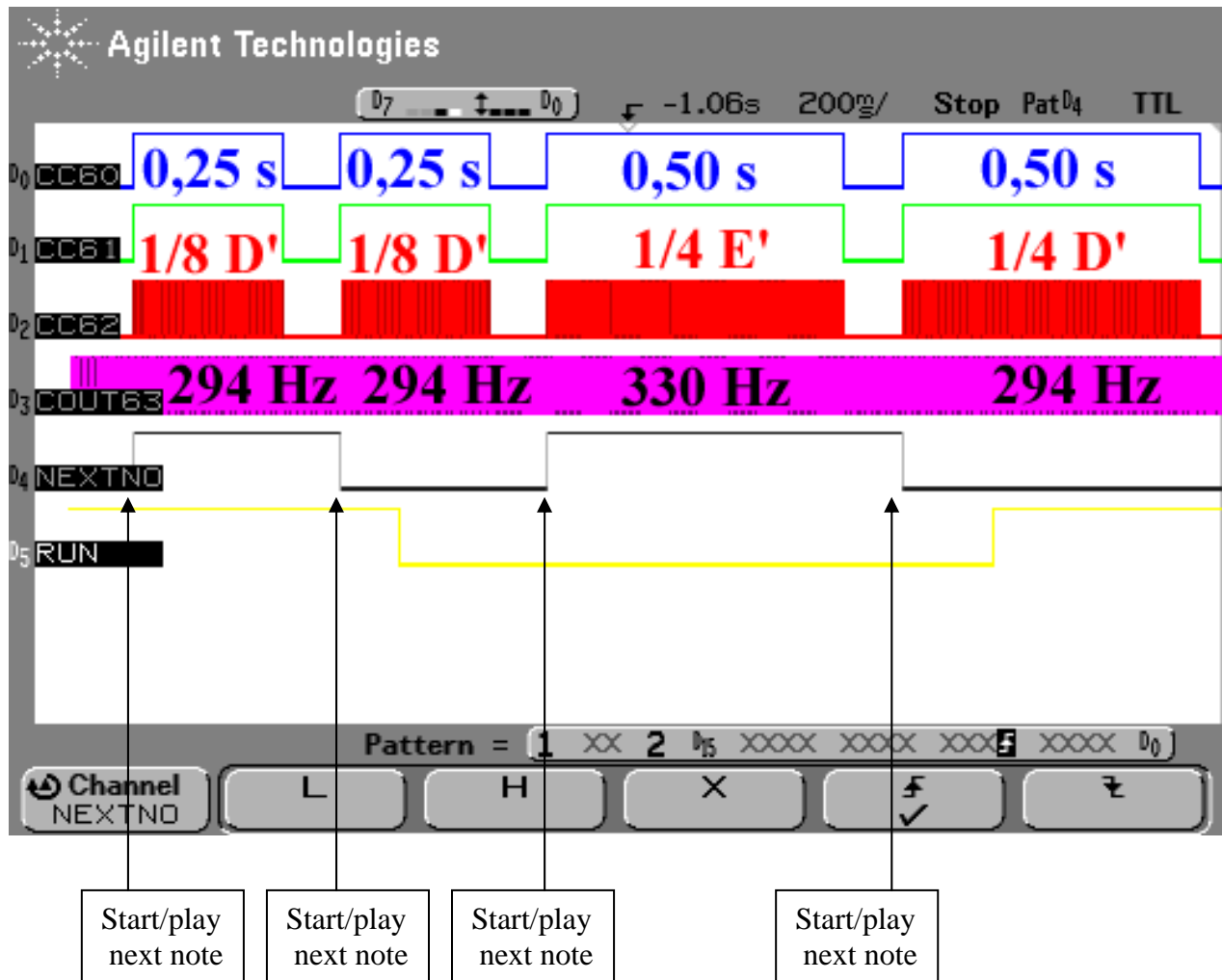






HAPPY BIRTHDAY =

T12000L8DDL4EDGL2F+L8DDL4EDAL2GL8DDL4O1DO0HL8GGL4F... ..

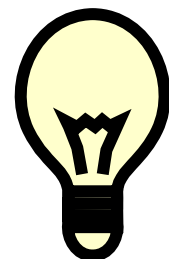


Note:

Start/play next note:

IO\_vTogglePin(IO\_P2\_8); // Show start of next note on Port 2.8

CCU61\_vStartTmr(CCU61\_TIMER\_12); // Start next note (T12 single shot)



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






Syntax used in our programming example:

Lx : Change note length

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

Real Music:




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

Syntax used in our programming example:

. : Extend preceding note by half of its value

Real Music:

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

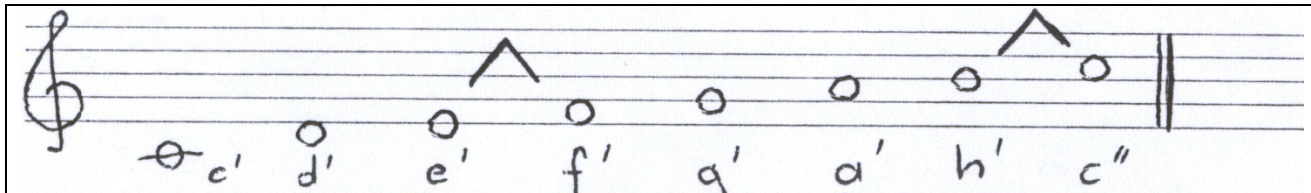
**Note:**

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

Syntax used in our programming example:

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

Real Music:



**Note:**

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

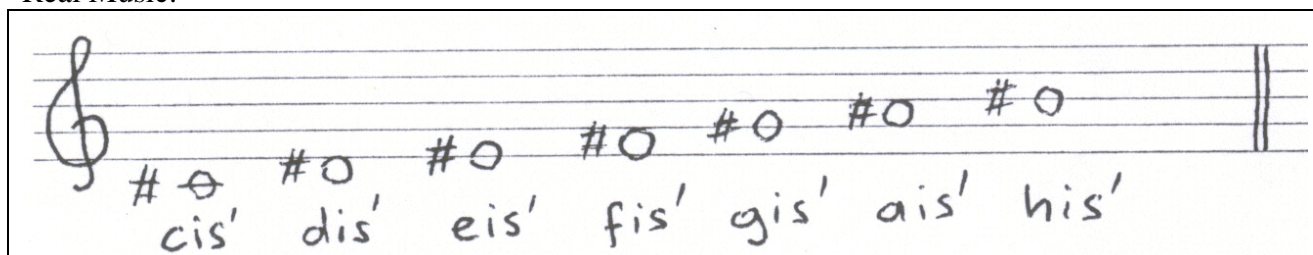




Syntax used in our programming example:

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

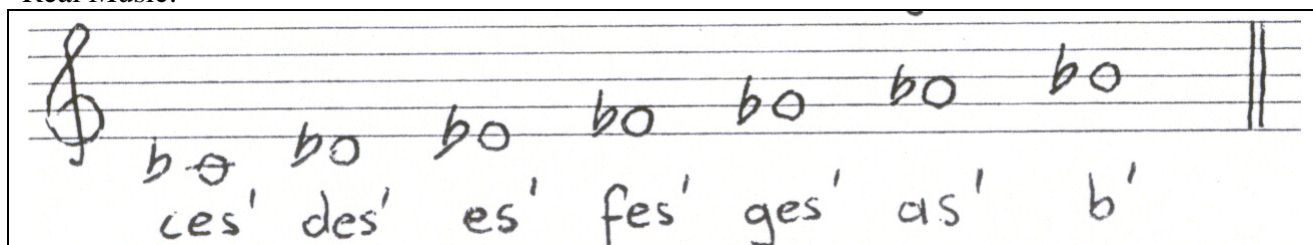
Real Music:



Syntax used in our programming example:

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

Real Music:













Syntax used in our programming example:

Px : play rest/pause/interval of silence

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

Real Music:

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

**Note:**

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

Therefore, playing a rest means playing a note.

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

Octave:

**Definition:**

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

**Note:**

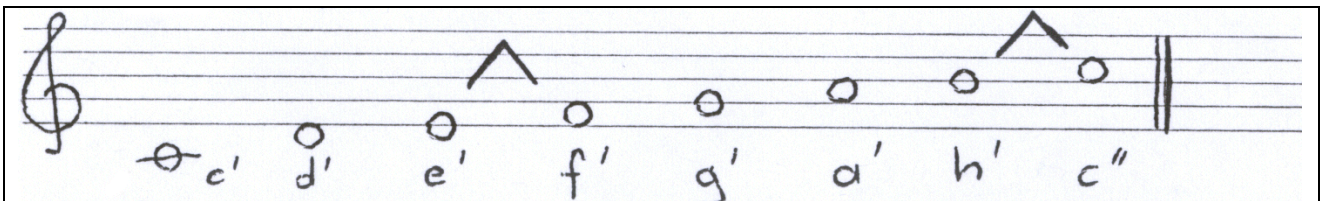
If one note has a frequency of 400 Hz, the note an octave above it is 800 Hz.

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

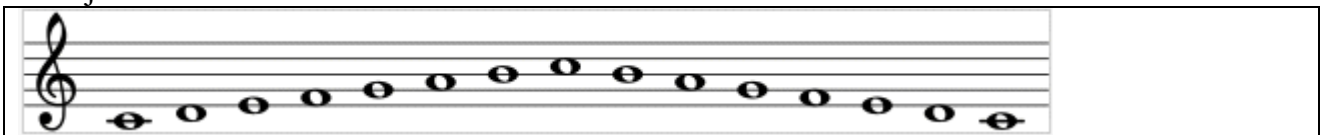
Syntax used in our programming example:

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

Real Music:

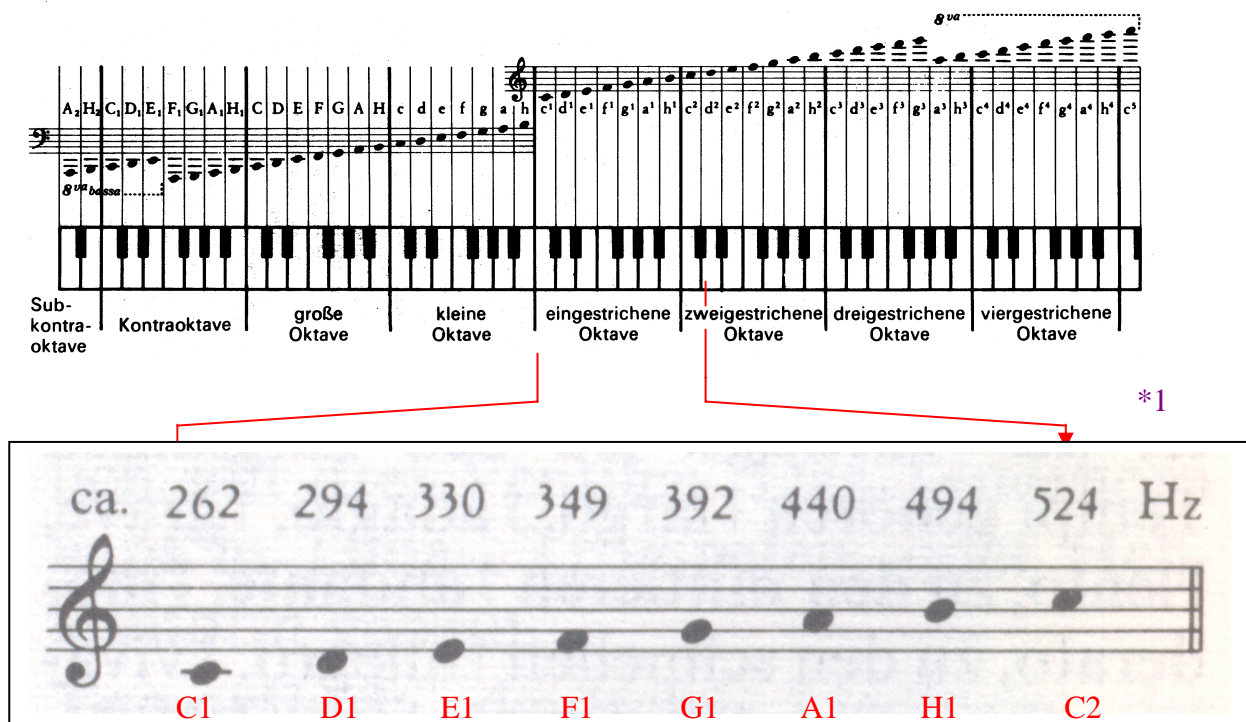


C major scale:



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

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



\*2

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

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

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

[illegible]

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

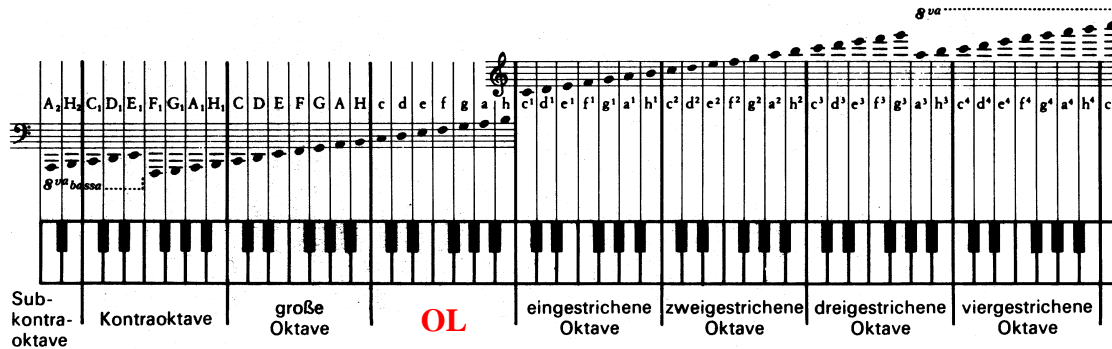
[**Note:** Timer 13 resolution =  $1/(f_{clk}/4) = 1/(66\text{MHz}/4) = 60,606 \text{ ns}$ ]:

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

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



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



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

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

So we get the following values shown in the table below

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

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

Therefore we use the following program sequence in our application:

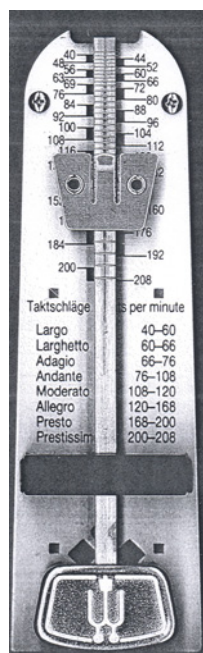
```
// T13, note frequency  
CCU61_T13PR=T13_values[note]/octave; // note frequency  
CCU61_CC63SR=T13_values[note]/octave/2; // duty cycle note frequency = 50 %  
CCU61_vEnableShadowTransfer(CCU61_TIMER_13);
```

note – length (Timer 12)

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



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



So we get the following table for speed:

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




**Note:**






Our software supports 50 to 199 Beats per minute:

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

And tempo is used in the following way:






```
CCU61_T12PR= ((float)current_note_length/ (float)tempo*120.0); // period value
note length
```


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

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

So we get the following values shown in the table below

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

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

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



In our programming example we use the following code sequences:

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

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

```
CCU61_T12PR=((float)current_note_length/(float)tempo*120.0); // period  
value note length
```

#### 4.) Appendix: songs used

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



// Maus am Mars (song a):

code unsigned char

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

#### Note:

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

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

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



// Yesterday (song b):

code unsigned char

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

#### Note:

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

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

4.3.) Song c: Bruder Jakob:

Bruder Jakob

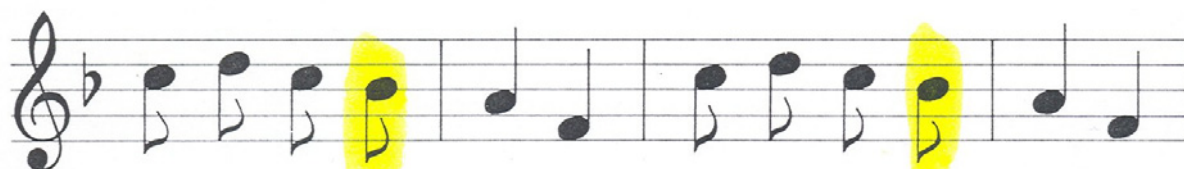
Französisches Kinderlied



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



schläfst du noch, schläfst du noch?



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



ding, dong, ding, ding dong, ding!

// Bruder Jakob (song c):

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

4.4.) Song d: Happy birthday:

Happy birthday

Englisches Kinderlied



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

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

hap- py birth- day to you!

// Happy birthday (song d):

code unsigned char

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



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

**TAKE ME HOME,  
COUNTRY ROADS**



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

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

code unsigned char

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

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

### Es tanzt ein Bi-ba-butzemann

Volkswiese



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



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



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

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

code unsigned char

```
songf[]="T19900L8DGG01DD00HHGGAADDL4GP8L8DGG01DD00HHGGAADDL4GP8L8  
HAHO1CO0AHO1CDO0L8HAHO1CO0AHO1CDO0DGG01DD00HHGGAADDL4G";
```

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

Ich geh mit meiner Laterne



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



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

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

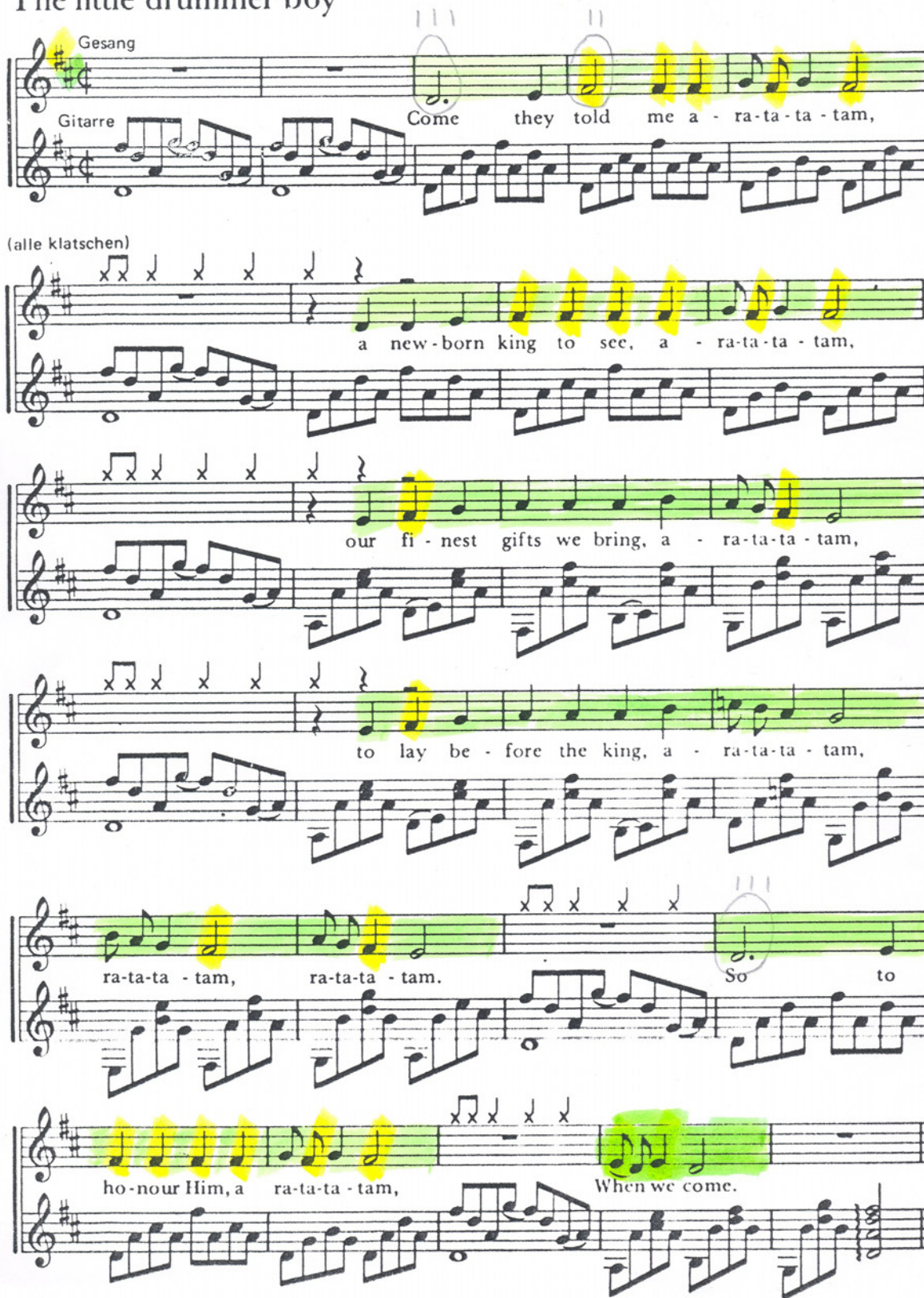
code unsigned char

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



4.8.) Song h: The little drummer boy:

The little drummer boy



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

// The little drummer boy (song h):

code unsigned char

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



4.9.) Song i: Hey, Pippi Langstrumpf:

# Hey, Pippi Langstrumpf



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



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



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



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



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



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

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

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



// Hey, Pippi Langstrumpf (song i):

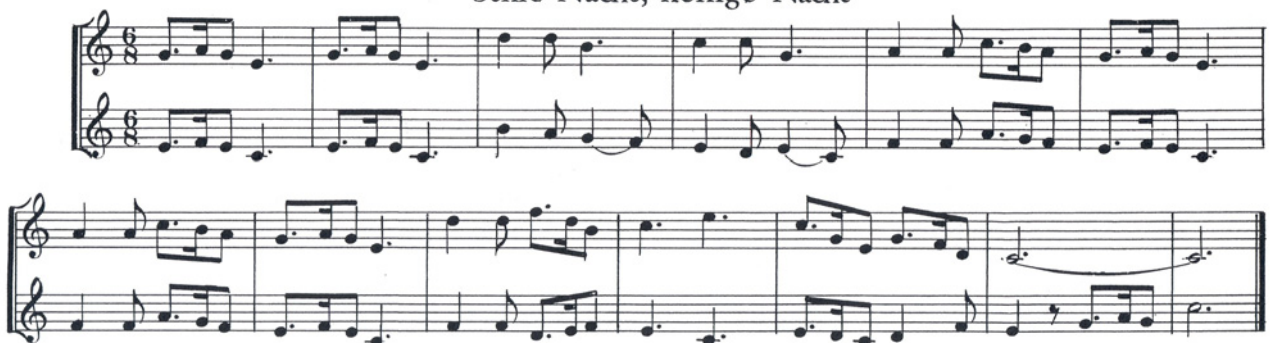
code unsigned char

```
songi[]="T180OLL4AON00L4DF+DL2EL8GF+EDL4C+EOLAON00L4C+L2DF+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";
```

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



Stille Nacht, heilige Nacht



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

code unsigned char

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

## 4.11.) Song k: Junge komm bald wieder:

### Junge komm' bald wieder



Jun - ge, komm' bald wie - der, bald wie - der nach Haus. Jun - ge, fahr' nie wie - der, nie

wie - der hin - aus. Ich mach' mir Sor - gen, Sor - gen um dich. Denk' auch an mor - gen,

denk' auch an mich. Jun - ge, komm' bald wie - der, bald wie - der nach Haus. Jun - ge, fahr' nie

wie - der, nie wie - der hin - aus. Ich weiß noch wie die er - ste Fahrt ver - lief, ich

schlich mich heim - lich fort, als Mut - ter schlief. Als sie er - wach - te, war ich auf dem

Meer. Im er - sten Brief stand: „Komm doch bald wie - der her!“

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

// Junge komm bald wieder (song k):

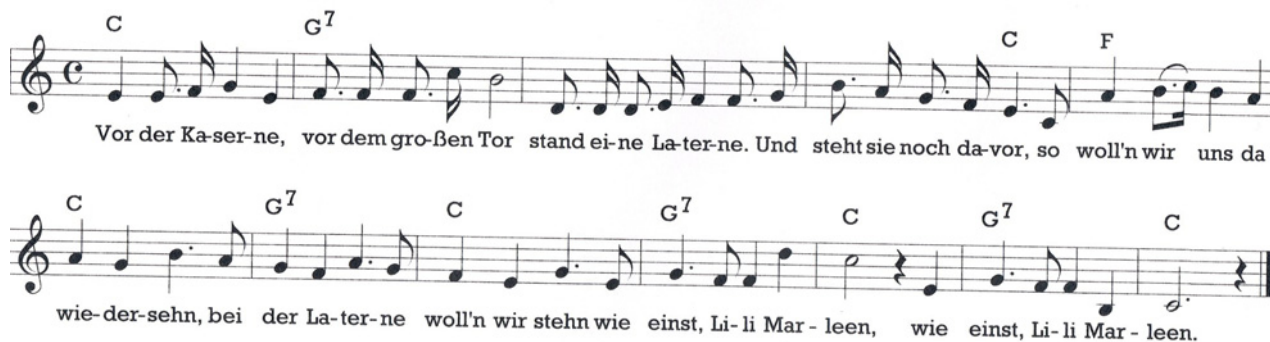
code unsigned char

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



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

### Lili Marleen



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

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

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

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

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

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

// Lili Marleen (song 1):

code unsigned char

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



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



// Lady Bird:

```
"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, XC16x, XE16x and XC2xxx) and 32 bit microcontrollers (e.g. TC1796 and TC1130).

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

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

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

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

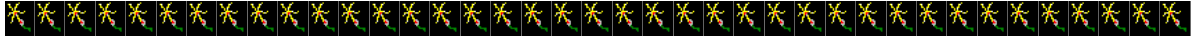
## 5.) Thanks To



Maria, Christian, Hermann and Maureen for their support.



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



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

---

---

---

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

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

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



**Your suggestions:**

---

---

---

---

---

---

---

---

---

<http://www.infineon.com>

Published by Infineon Technologies AG