Мікропроцесорна техніка

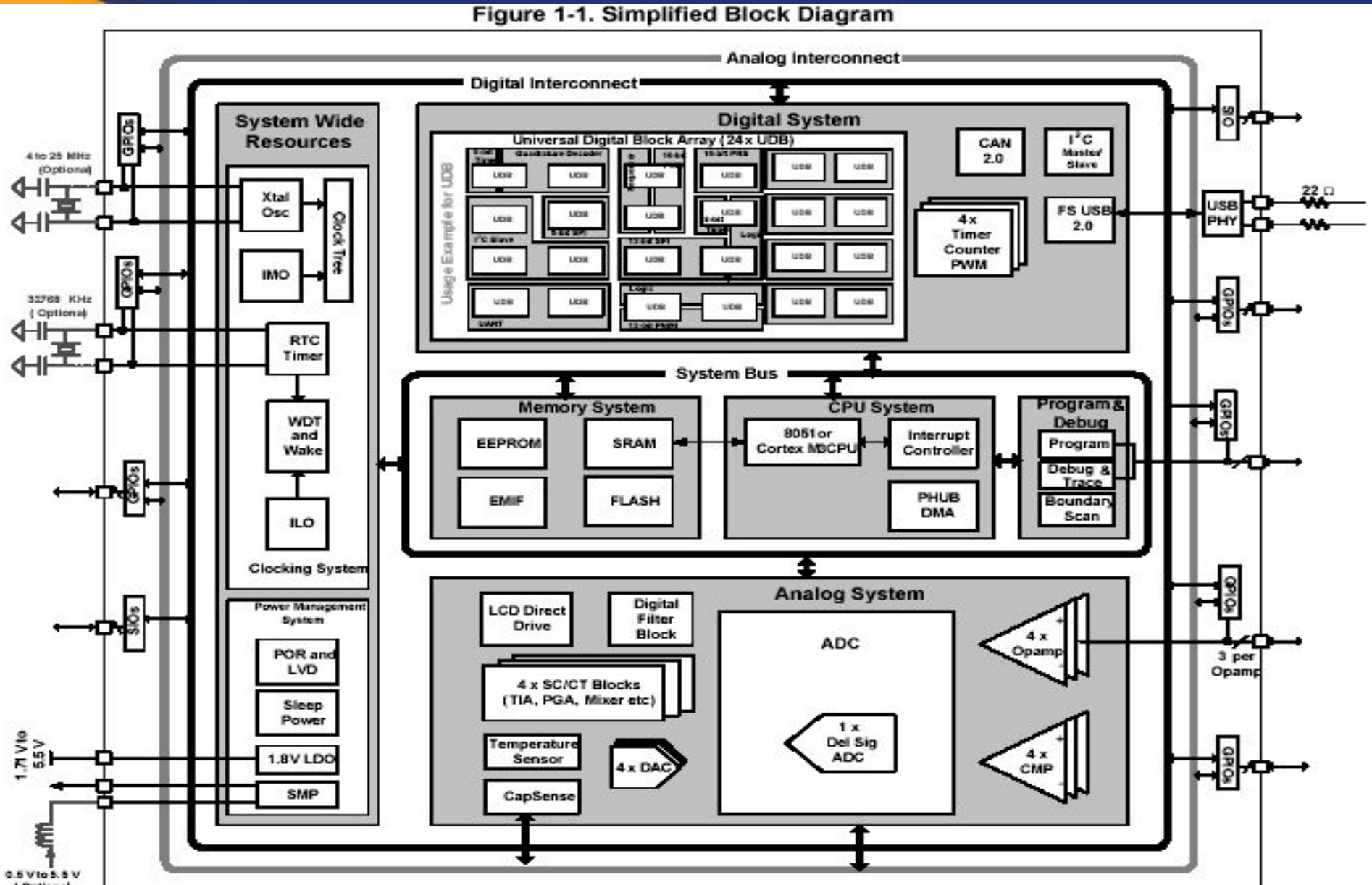
(лекція 10)
Благітко Б.Я.
2019 р

PSoC Creator 4.2
Designing with PSoC 3/5

PSoC@3/5
VDAC8+DMA

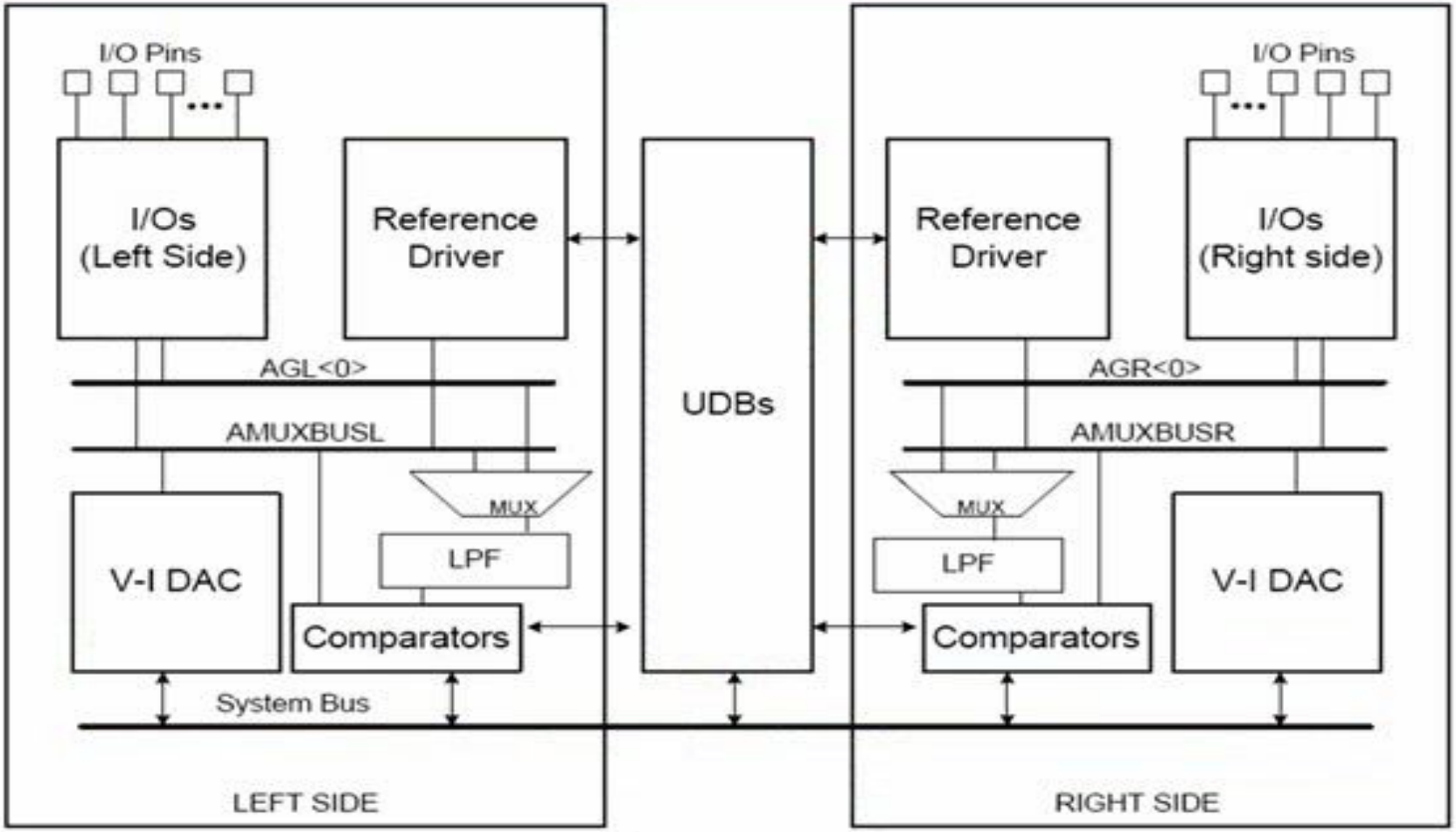
PSoC Creator 4.2
Designing with PSoC 3/5

Модулі PSoC@3/5

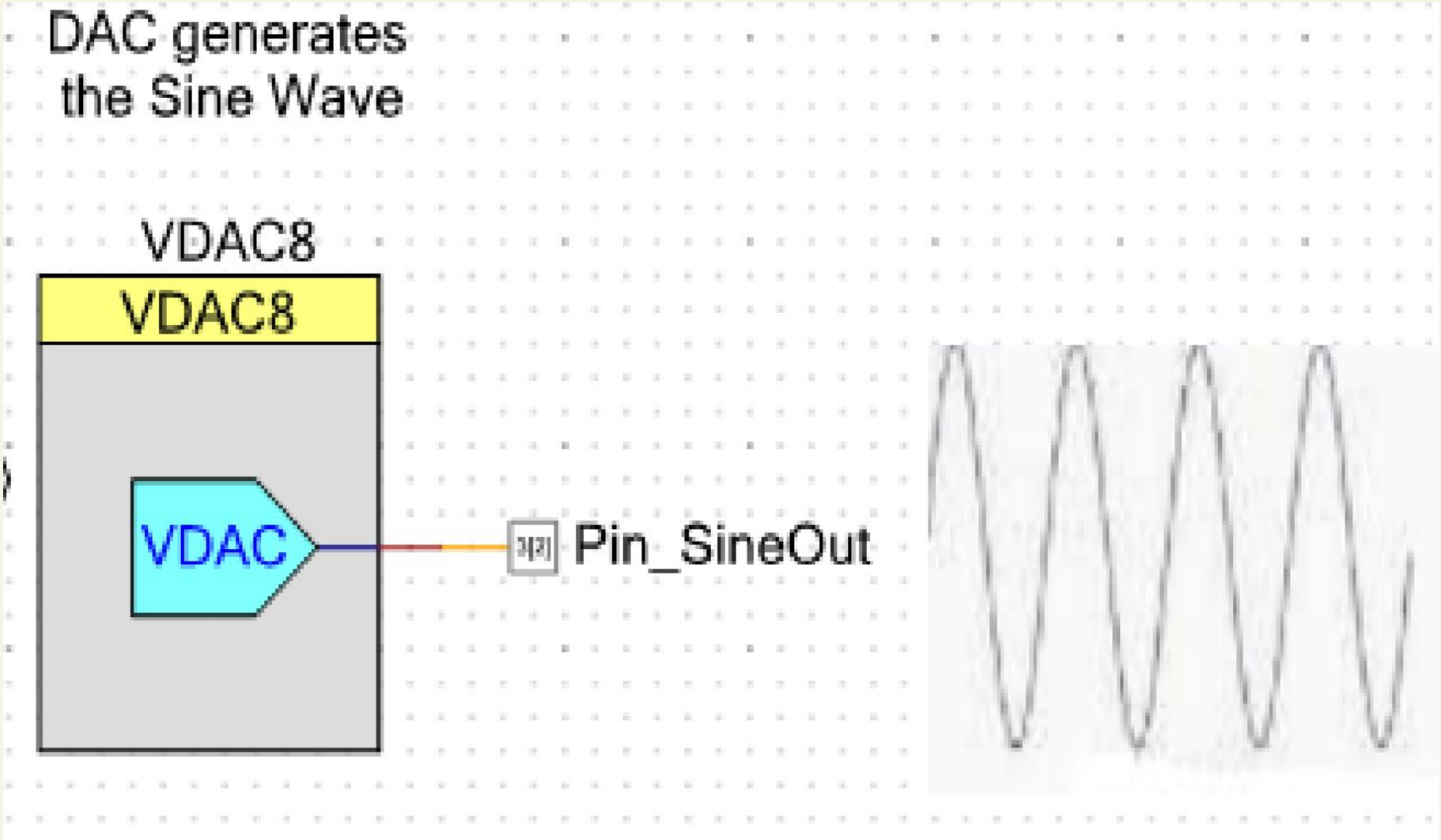


Модулі PSoC@3/5

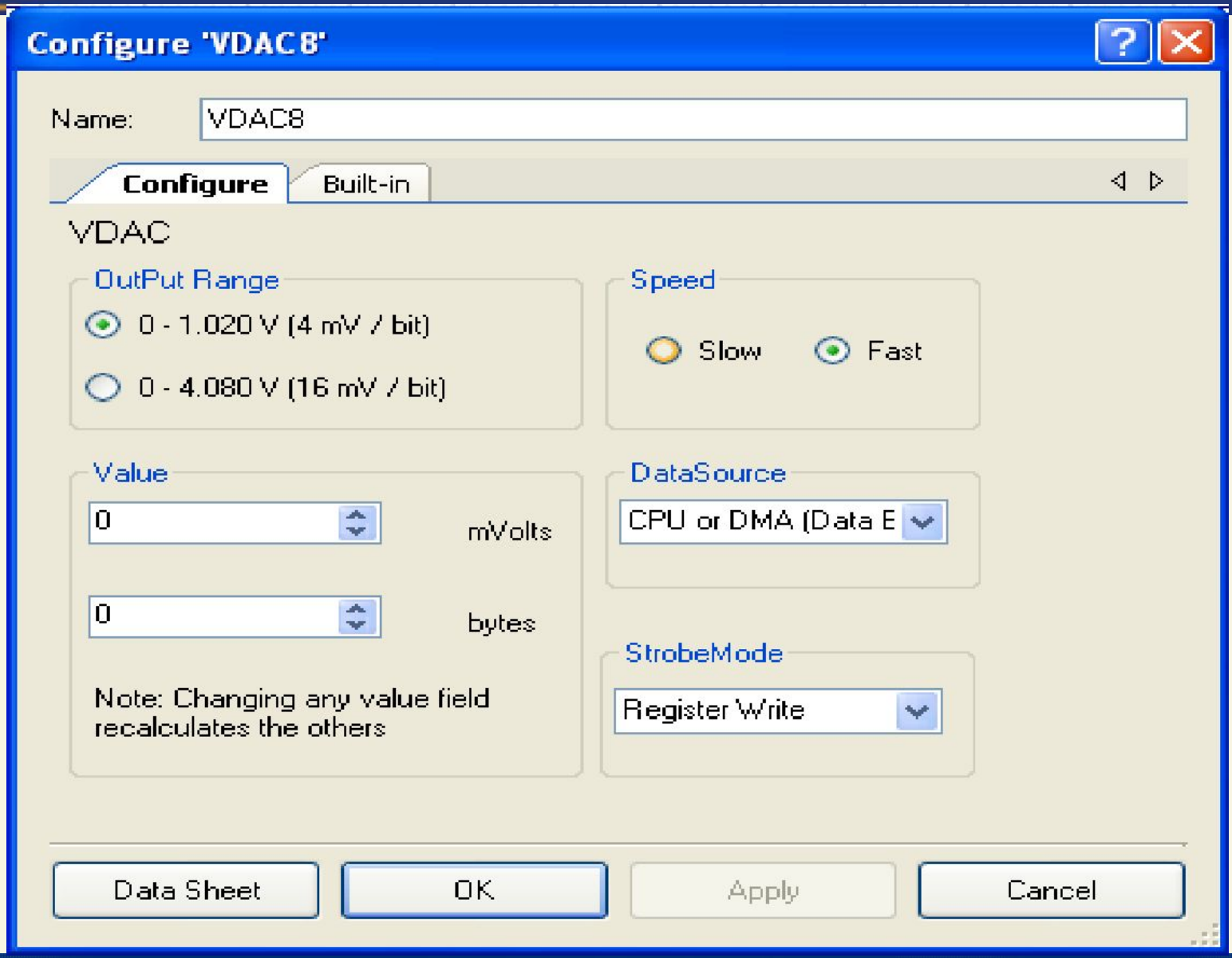
CapSense in PSoC 3 / PSoC 5



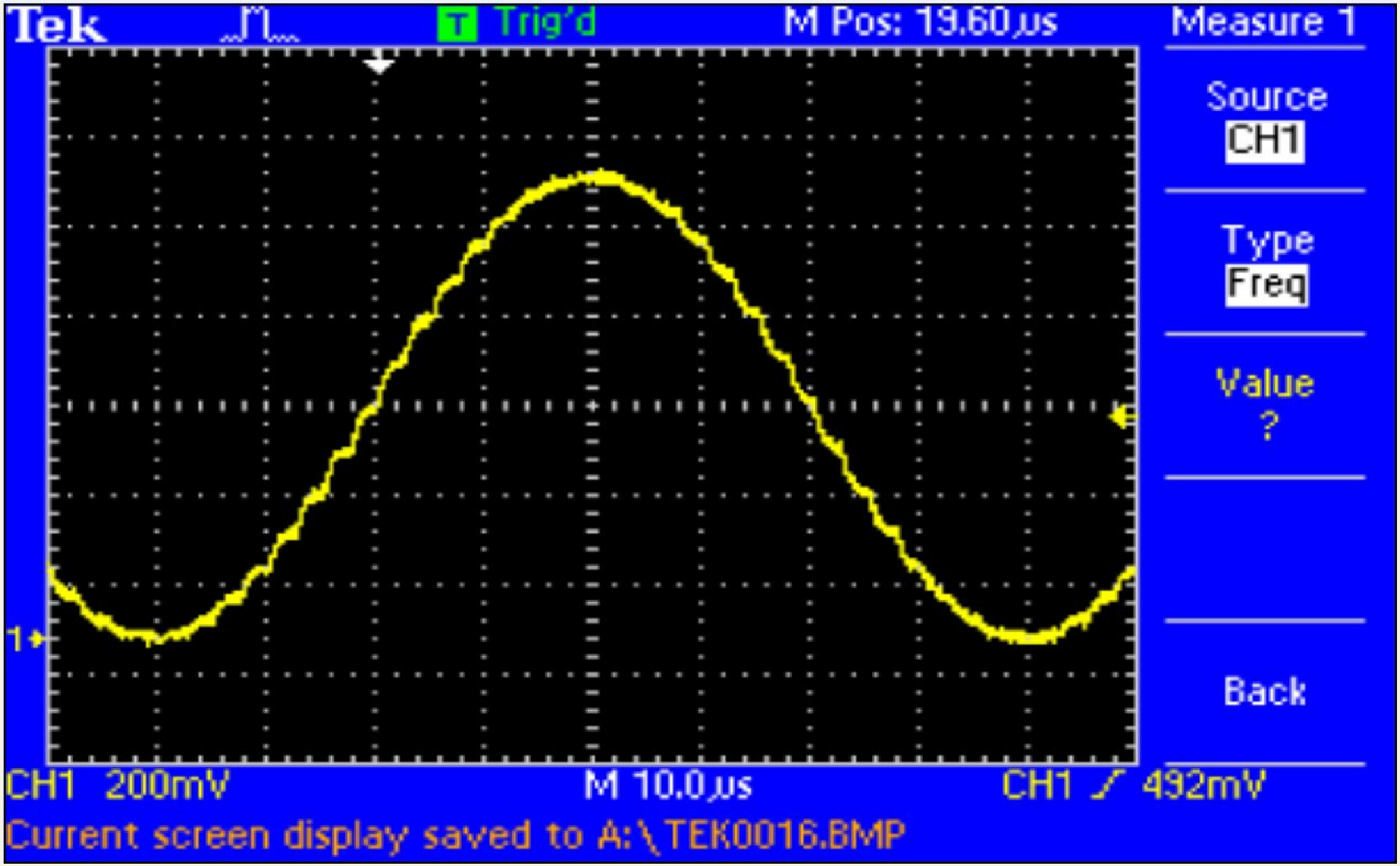
Модуль VDAC8



Модуль VDAC8



Модуль VDAC8



DMAC

The DMA controller (DMAC) in PSoC 3 and PSoC 5LP can transfer data from a source to a destination with no CPU intervention. This allows the CPU to handle other tasks while the DMA does data transfers, thereby achieving a 'multiprocessing' environment.

The PSoC DMA Controller (DMAC) is highly flexible – it can seamlessly transfer data between memory and on chip peripherals including ADCs, DACs, Filter, USB, UART, and SPI.

There are 24 independent DMA channels.

DMAC

There are 24 independent DMA channels. Each of the 24 DMA channels can independently transfer data.

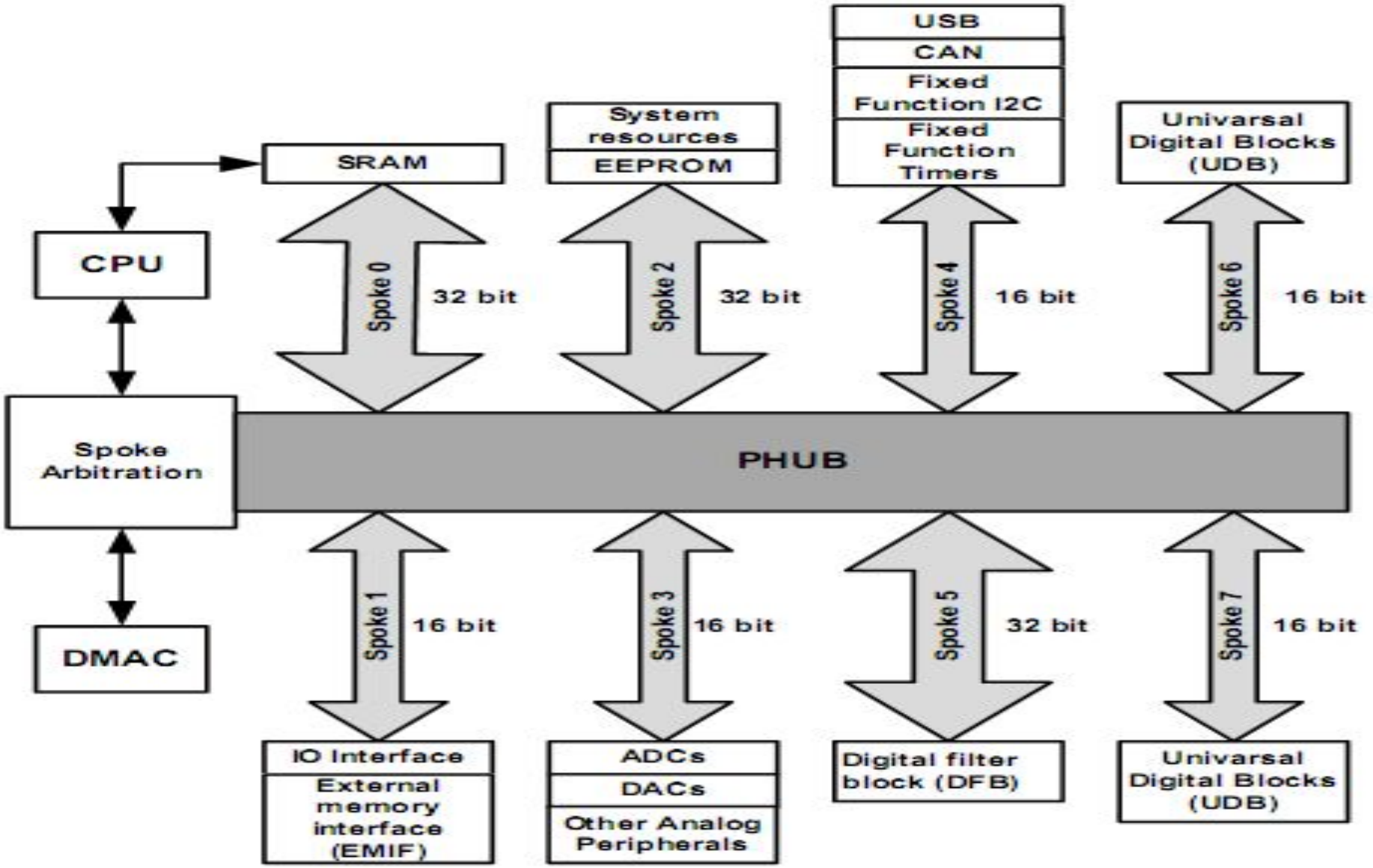
Each channel has a Transaction Descriptor (TD) chain.

The TD contains information such as source address, destination address, transfer count, and the next TD in the chain.

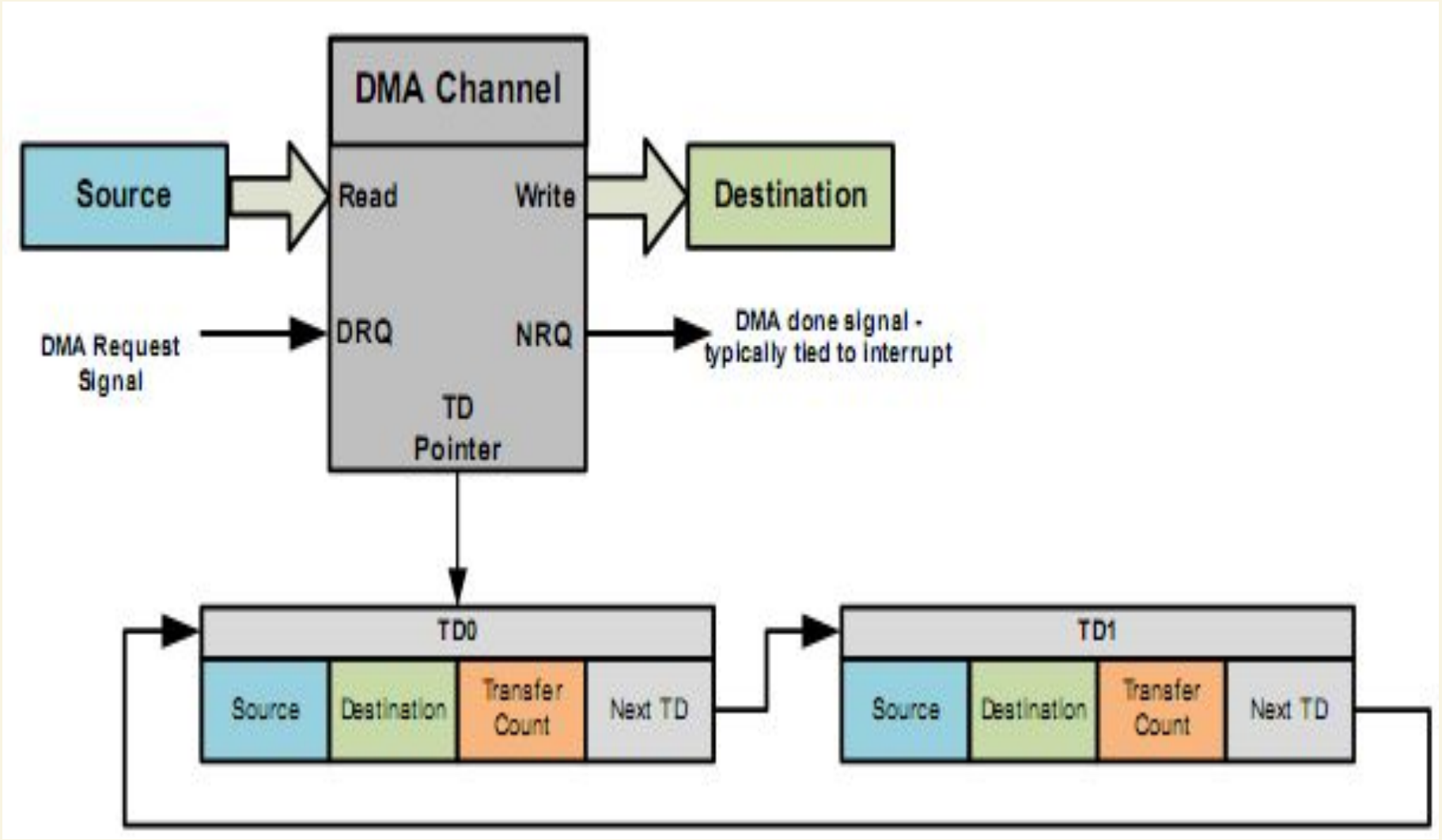
There can be as many as 128 TDs.

The combination of channel and TD describes the complete DMA transfer.

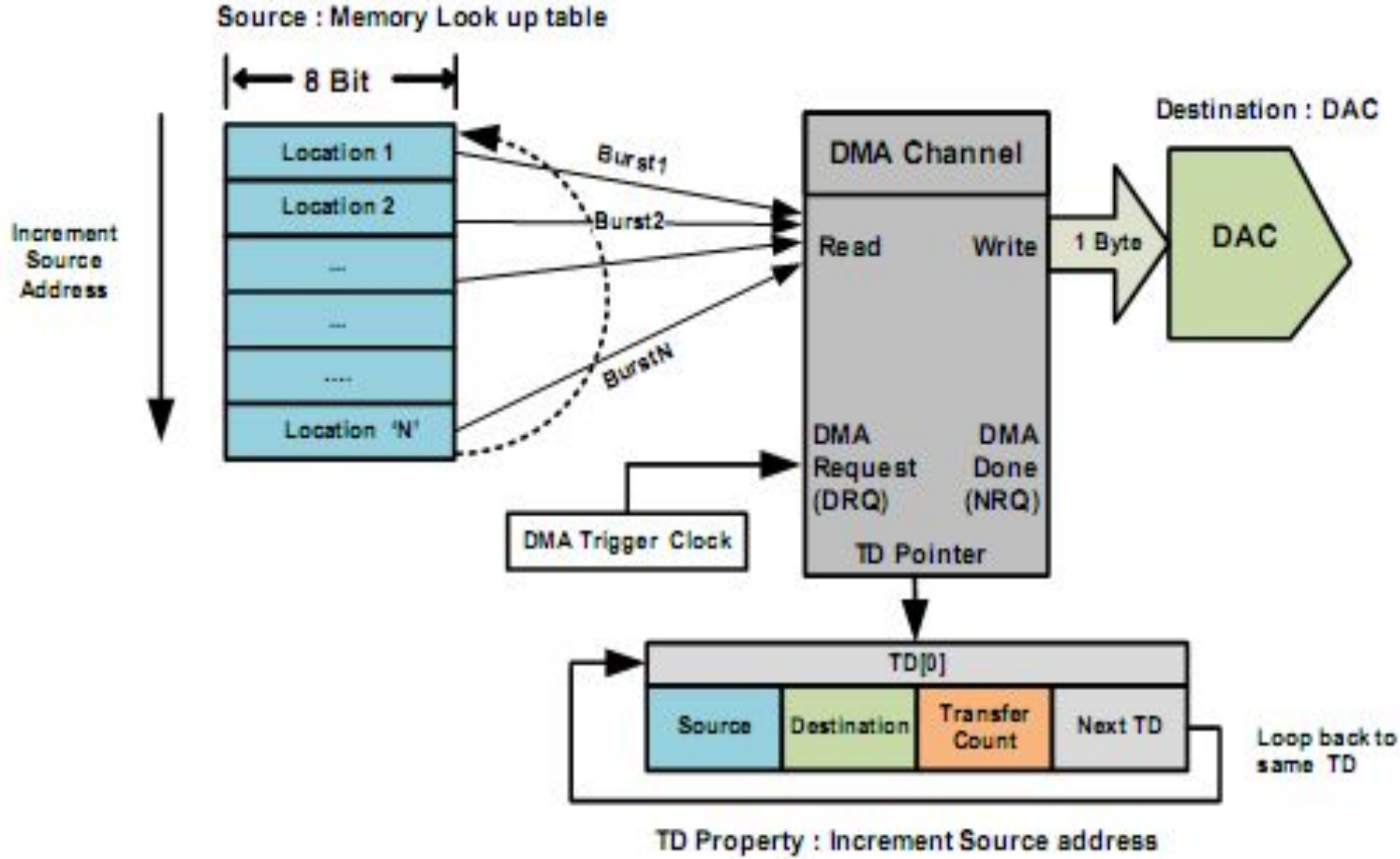
DMAC


USB
CAN
Fixed Function I2C
Fixed Function Timers
System resources
EEPROM
Univarsal Digital Blocks (UDB)
SRAM
CPU
Spoke 0
32 bit
Spoke 2
32 bit
Spoke 4
16 bit
Spoke 6
16 bit
Spoke Arbitration
PHUB
DMAC
Spoke 1
16 bit
Spoke 3
16 bit
Spoke 5
32 bit
Spoke 7
16 bit
IO Interface
External memory interface (EMIF)
ADCs
DACs
Other Analog Peripherals
Digital filter block (DFB)
Univarsal Digital Blocks (UDB)

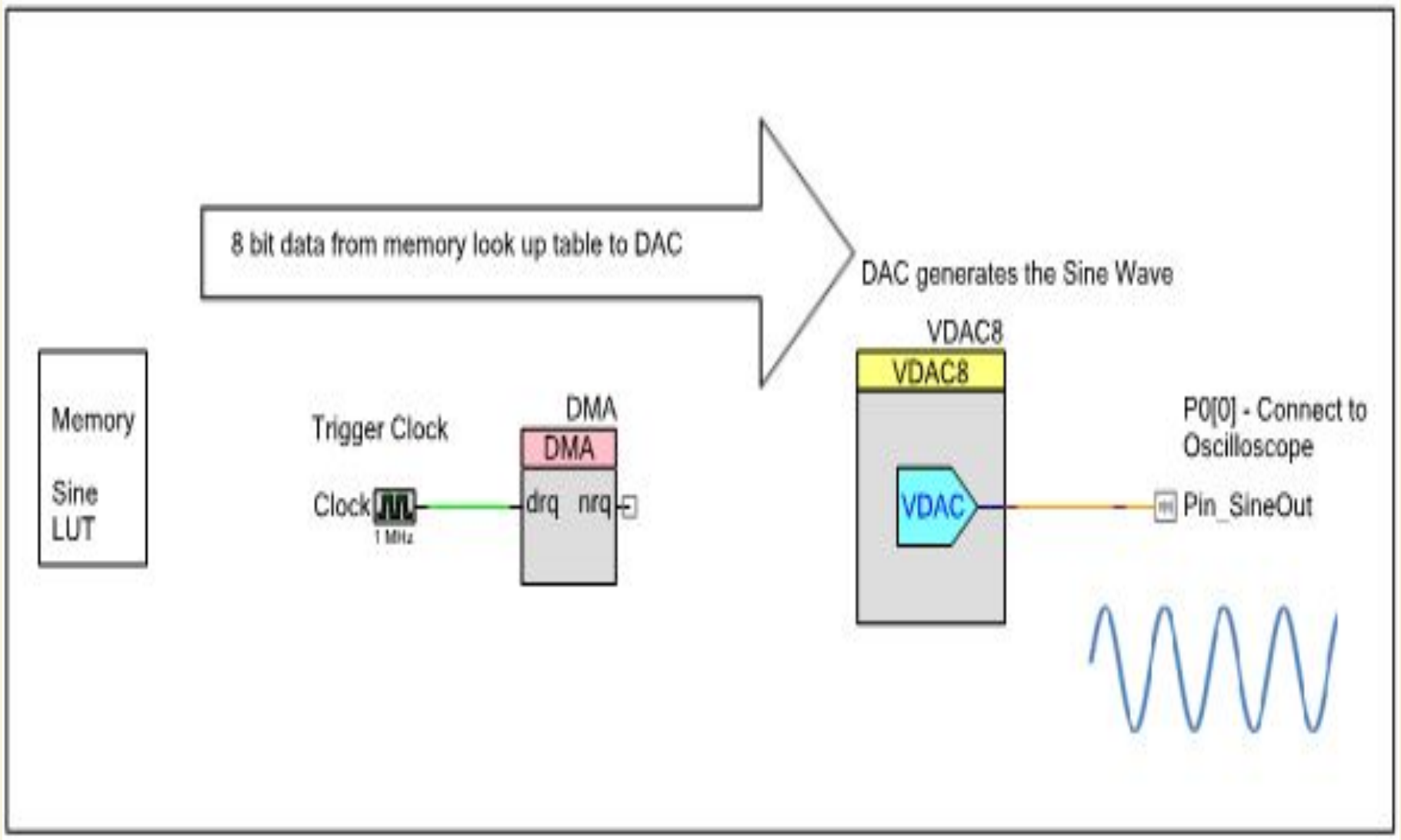
DMA Channel


DMA Channel
Source
Read
Write
Destination
DRQ
NRQ
DMA Request Signal
DMA done signal - typically tied to interrupt
TD Pointer
TD0
Source
Destination
Transfer Count
Next TD
TD1
Source
Destination
Transfer Count
Next TD

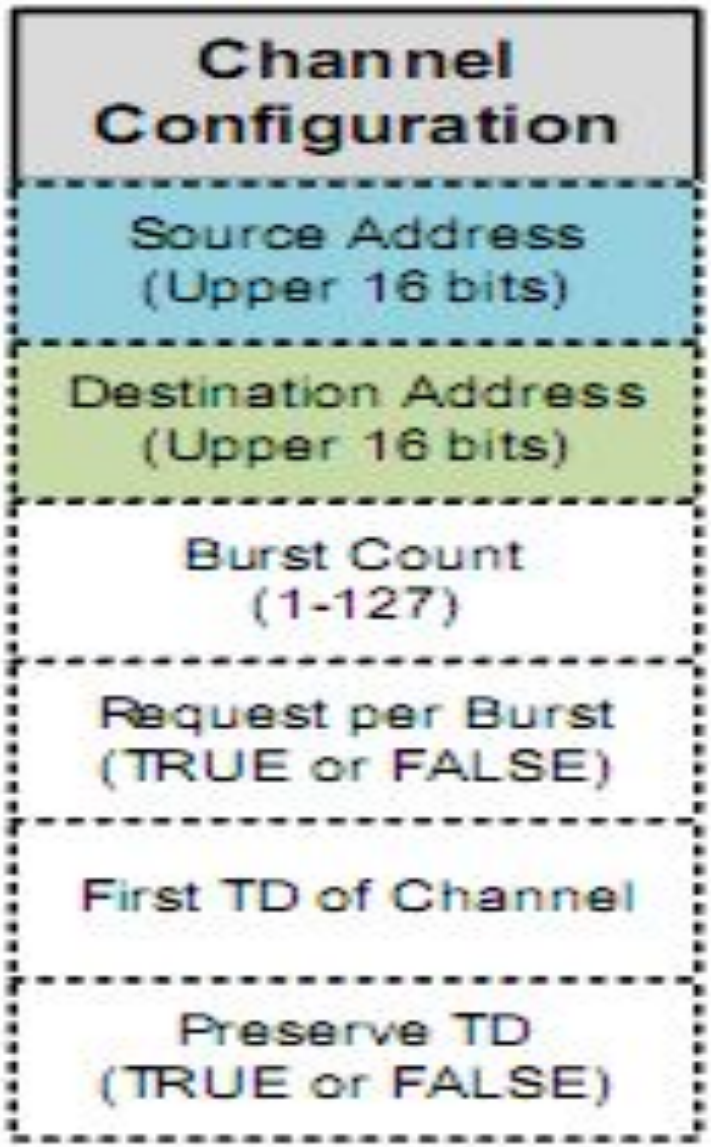
Memory-to-Peripheral Transfer



Memory-to-Peripheral Transfer

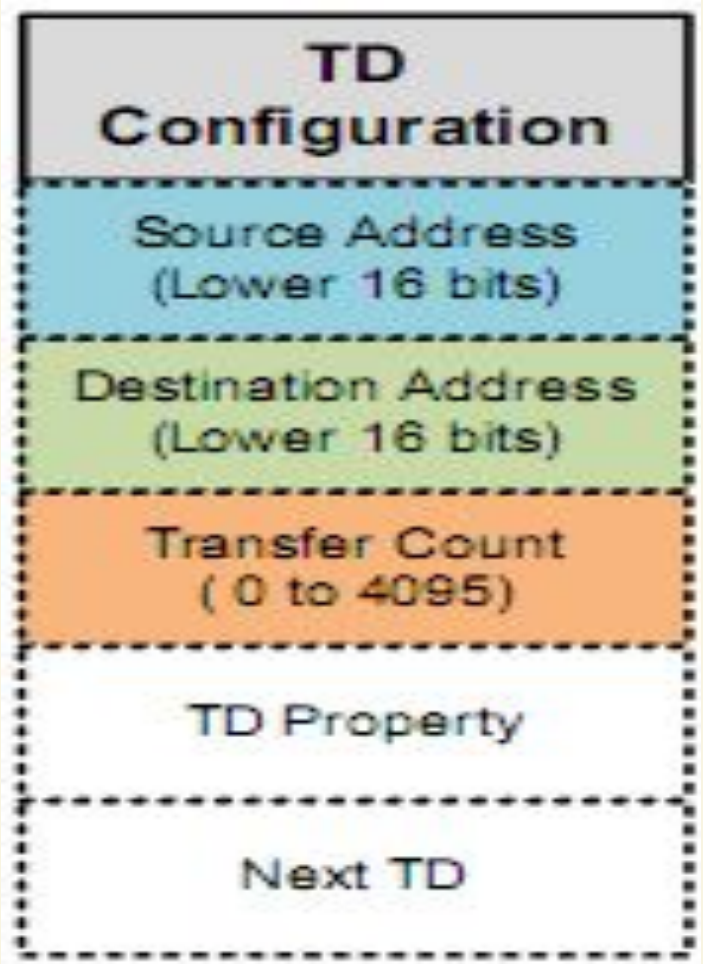


Channel Configuration



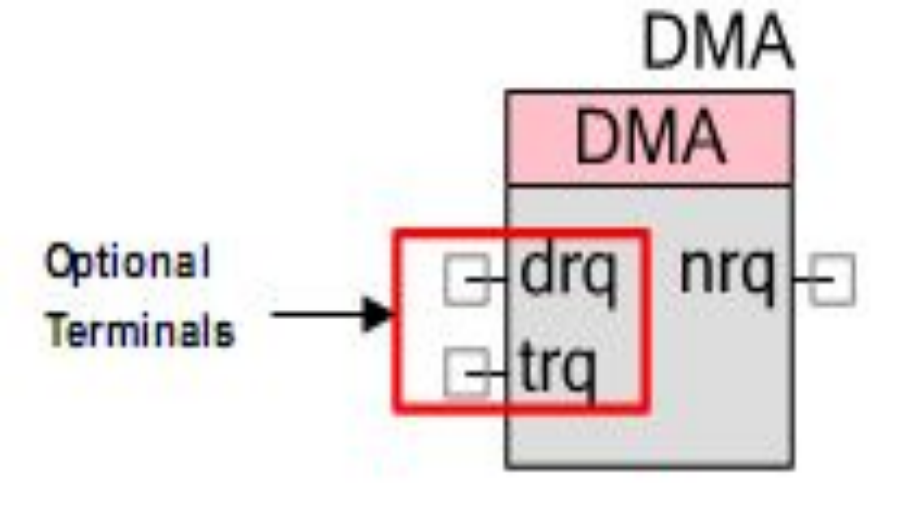
Parameter	Project Setting
Upper Source Address	HI16(CYDEV_FLS_BASE) for PSoC 3 HI16 (&sineTable) for PSoC 5LP
Upper Destination Address	HI16(CYDEV_PERIPH_BASE)
Burst Count	1 (One byte)
Request Per Burst	1 (True)
Initial TD	TD[0]
Preserve TD	1 (True)

TD[0] Configuration

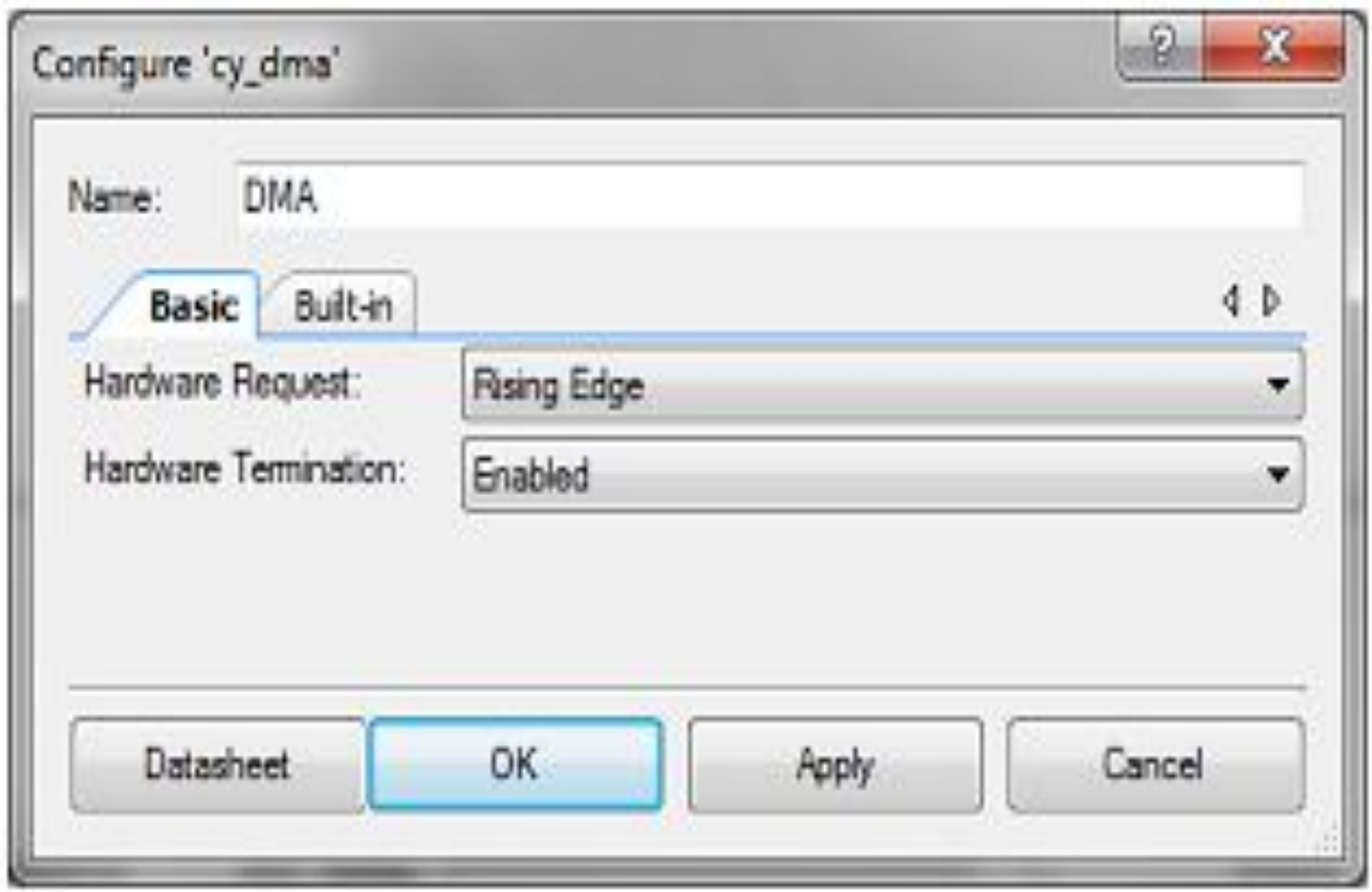


Parameter	Project Setting
Lower Source Address	LO16 (&sineTable)
Lower Destination Address	LO16(VDAC8_DATA_PTR)
Transfer Count	128 (No. of bytes in the sine look up table)
TD property	Increment source address (TD_INC_SRC_ADR)
Next TD	TD[0] - Loop back to the same TD again

DMA Channel Component



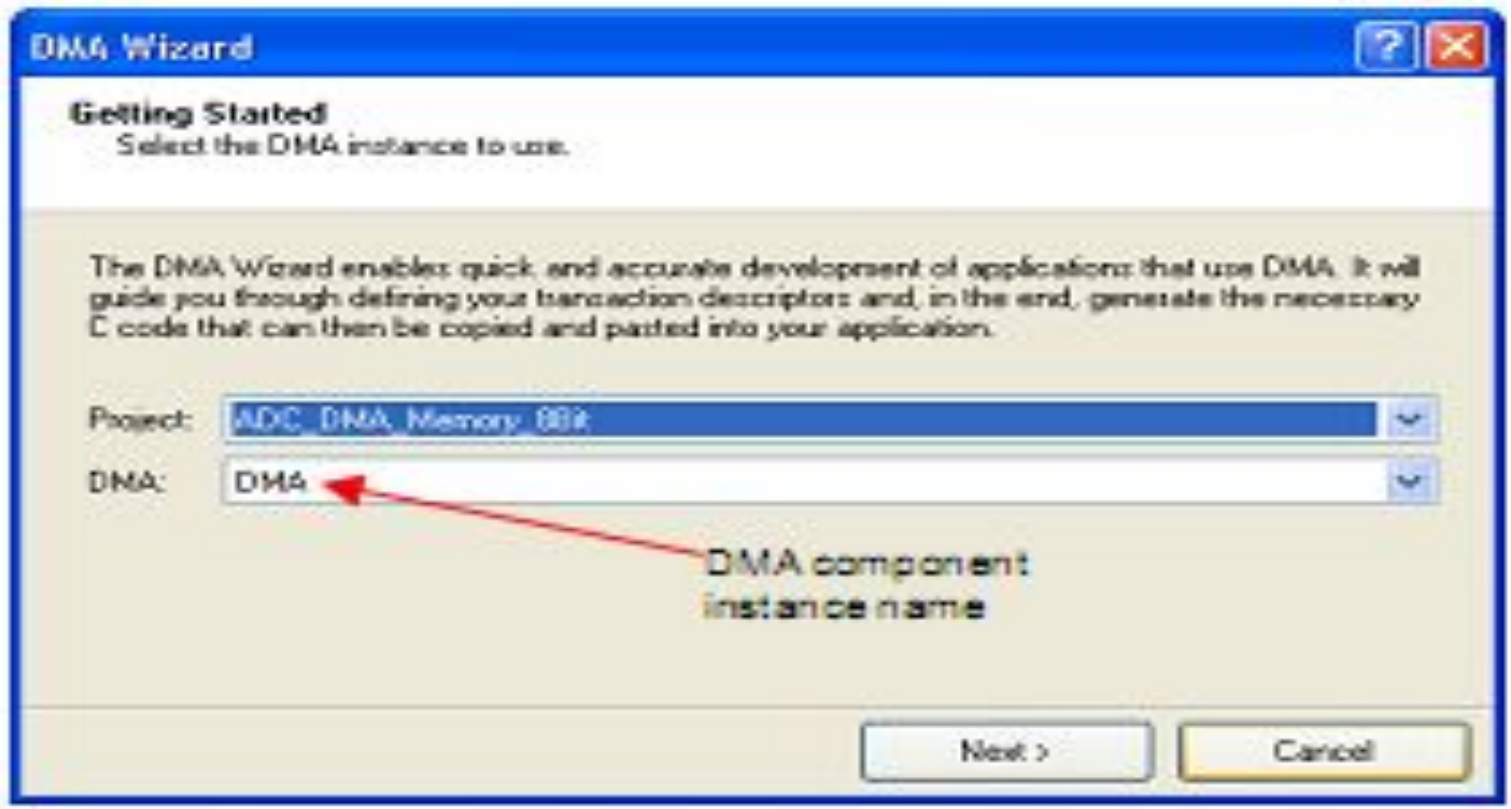
DMA component Configuration



DMA Wizard Configuration

To start the DMA wizard, go to PSoC Creator >Tools > DMA Wizard.

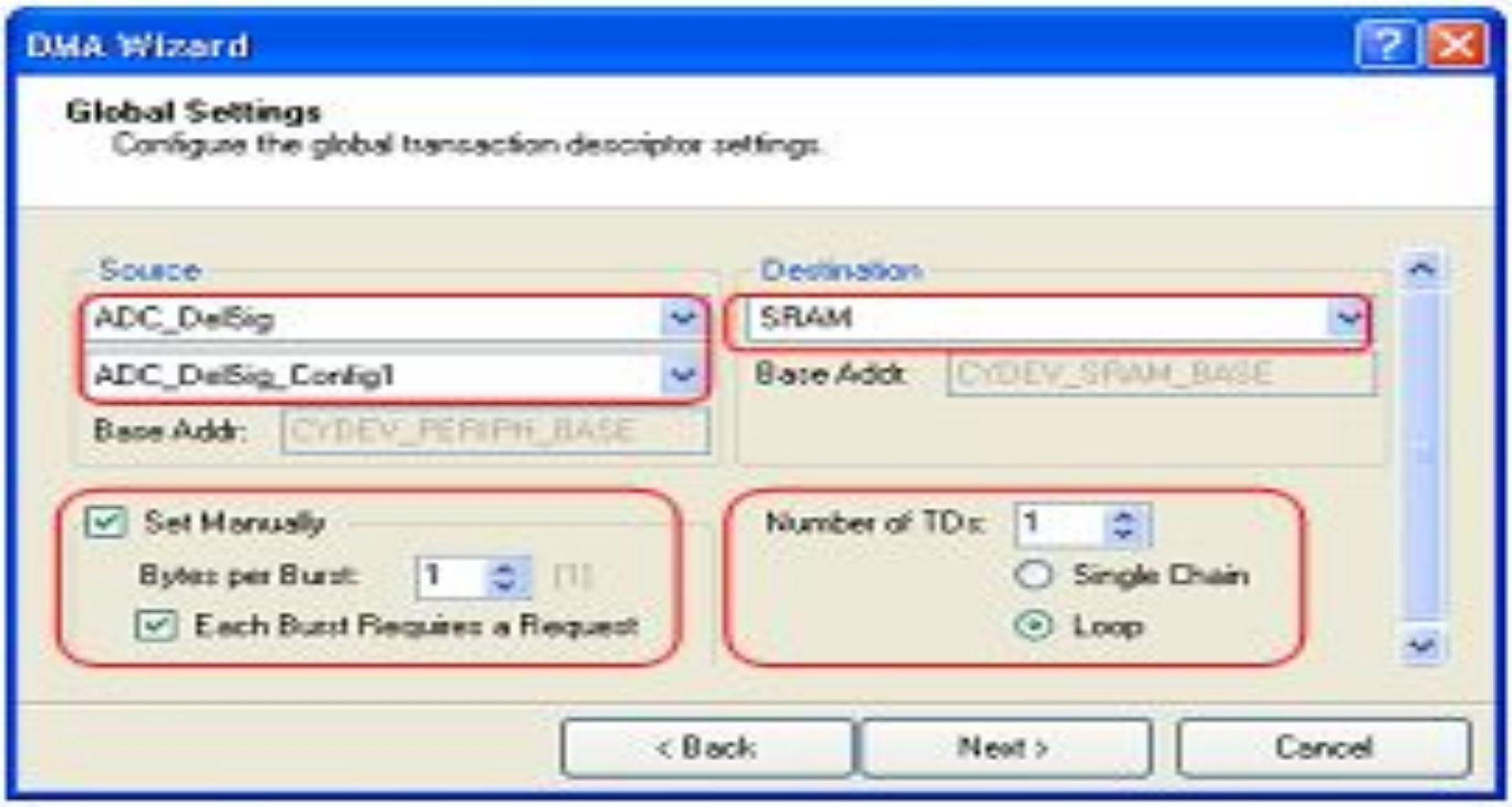
Step 1:
Select a DMA channel (DMA component instance)



DMA Wizard Configuration

To start the DMA wizard, go to PSoC Creator >Tools > DMA Wizard.

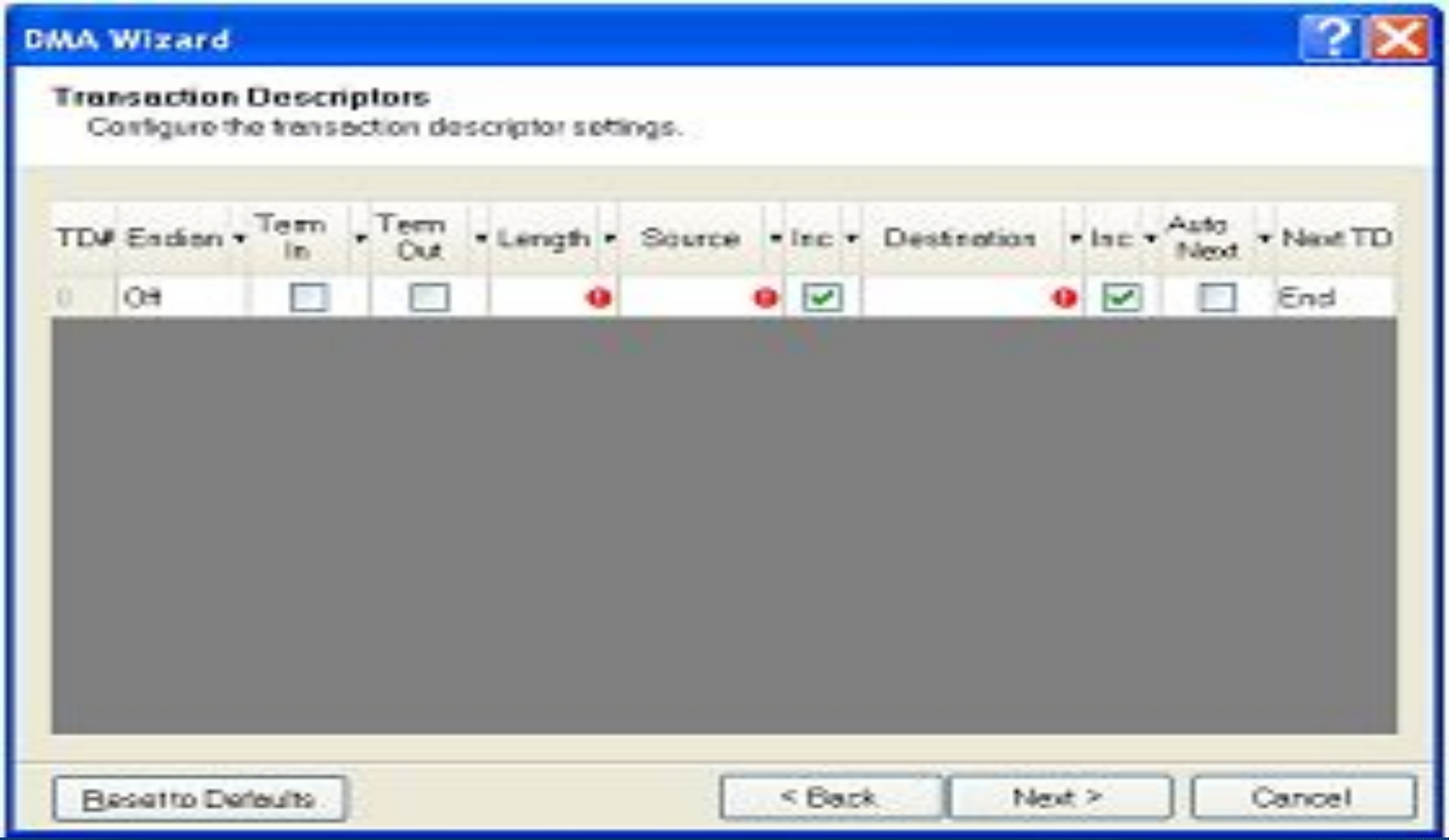
Step 2: Select global settings



DMA Wizard Configuration

Step 3:

Define the transaction descriptors for the channel



DMA Wizard Configuration

Step 3 (continue): TD Configuration Details

Field	Description
TD#	Displays the logical number for the Transaction Descriptor.
Endian	Enables 2- or 4-byte endian byte swapping. This enables swapping the byte while the data moves from source to destination. The **Bytes per Burst** setting must be set as a multiple of the endian selection. This is usually used for DMA transfers between PSoC 3 memory and peripherals because of the difference in endianess.
Term In	Enables ending the TD transaction on a rising edge of the TERMIN (trq) signal.
Term Out	Enables the creation of the TERMOUT (nrq) signal when the TD finishes.
Length	This specifies the transfer count for the TD in bytes (0 to 4095). This is the total number of bytes that the DMA should transfer to complete the transaction.
Source	The lower 16 bits of the source address for the DMA transfer. A drop-down list of addresses for the source is given by the DMA wizard if the source selected is a component (not memory). You can also edit or enter the source address manually.
Inc (Source)	Enables incrementing of the source address as the DMA does the transaction. If this is enabled, every time the DMA reads the data from source, the source address is incremented by the number of bytes that the DMA has read. The DMA increments the source address until the entire transaction (transfer count) is finished.
Destination	The lower 16 bits of the source address for the DMA transfer. A drop-down list of addresses for the destination is given by the DMA wizard if the destination selected is a component (not memory). You can also edit or enter the destination address manually.
Inc(Destination)	Enables incrementing of the destination address as the DMA does the transaction. The DMA increases the destination address until the entire transaction (transfer count) is finished.
Auto Next	Automatically execute the next TD without another DMA request.
Next TD	The next logical TD in the chain of TDs. Set to END if this TD chain is finished with this TD.

DMA Wizard Configuration

Step 4: Copy the code created by the DMA Wizard

After the DMA channels and TD configuration are finished, the wizard creates code for the DMA channel.

This code includes the configuration for the DMA channel and the TDs.

The code is generated in a window in the DMA Wizard dialog.

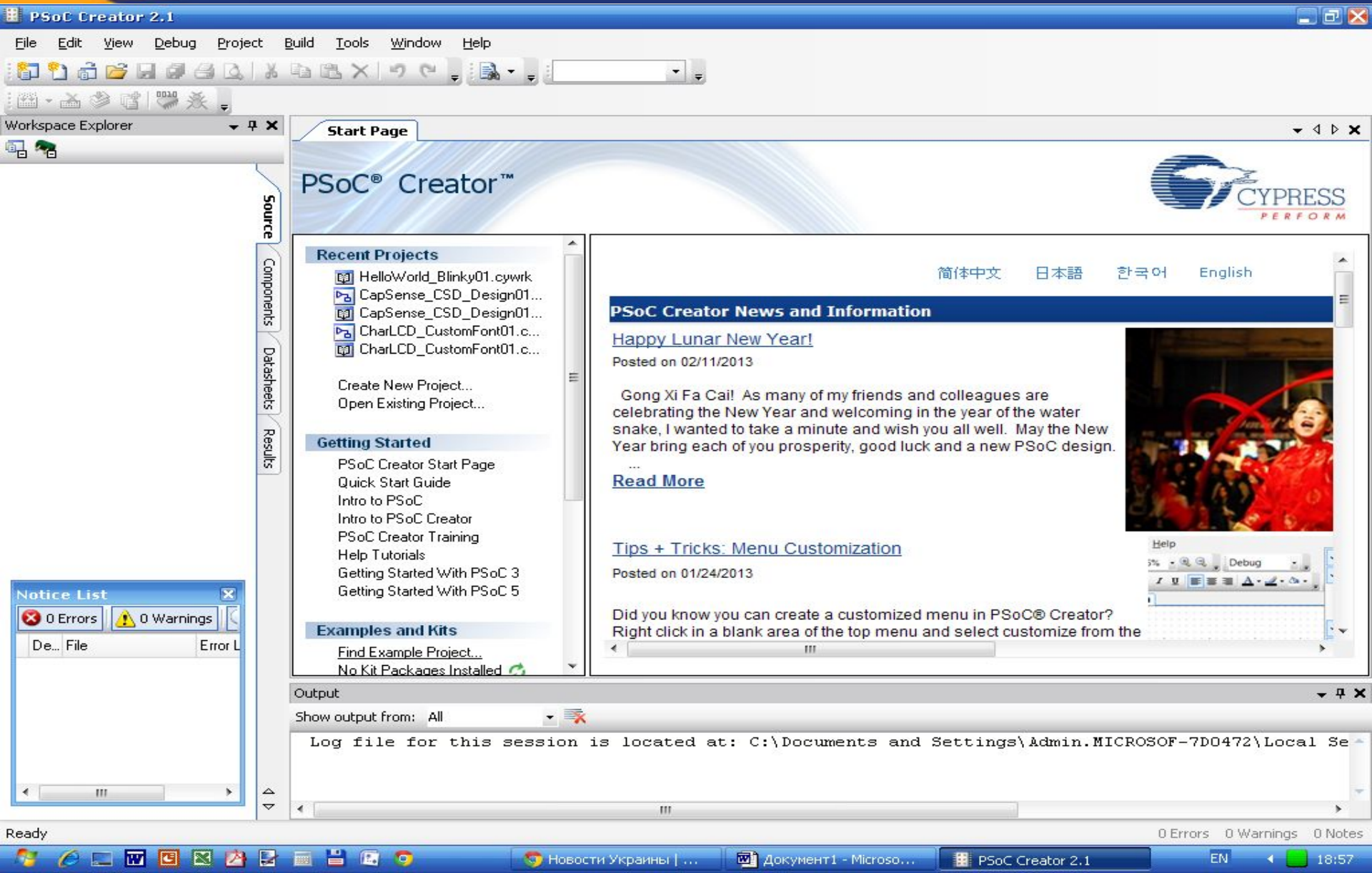
To use the code, select all in the window, copy it, and paste it in your main.c

VDAC8

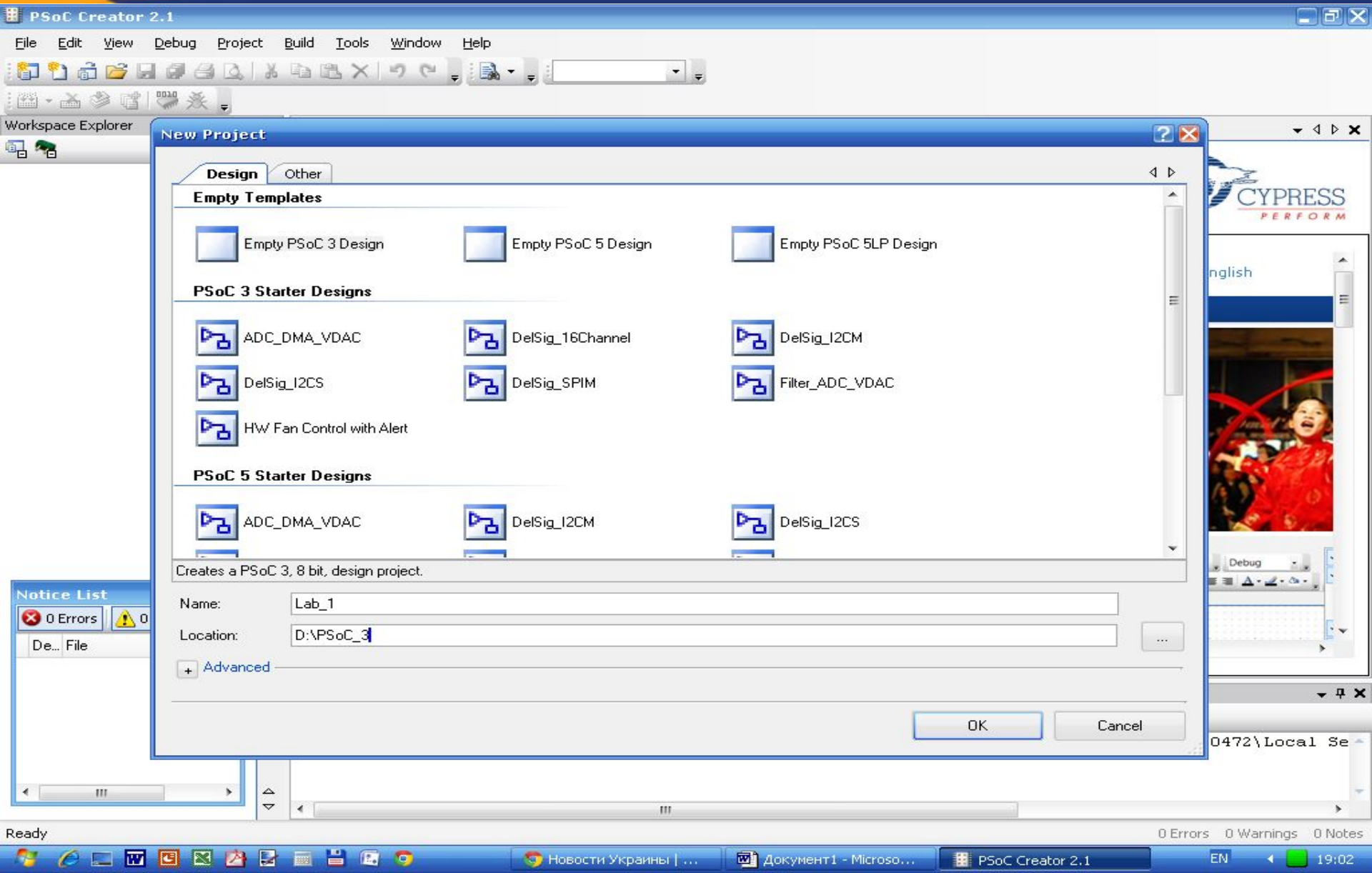
Follow the below steps to do this:

- **The Lab already has the LCD Character component installed and configured.**
- **Add aVDAC8 component from the component catalog.**
- **In the general tab, configure theVDAC8 component as in the image below**

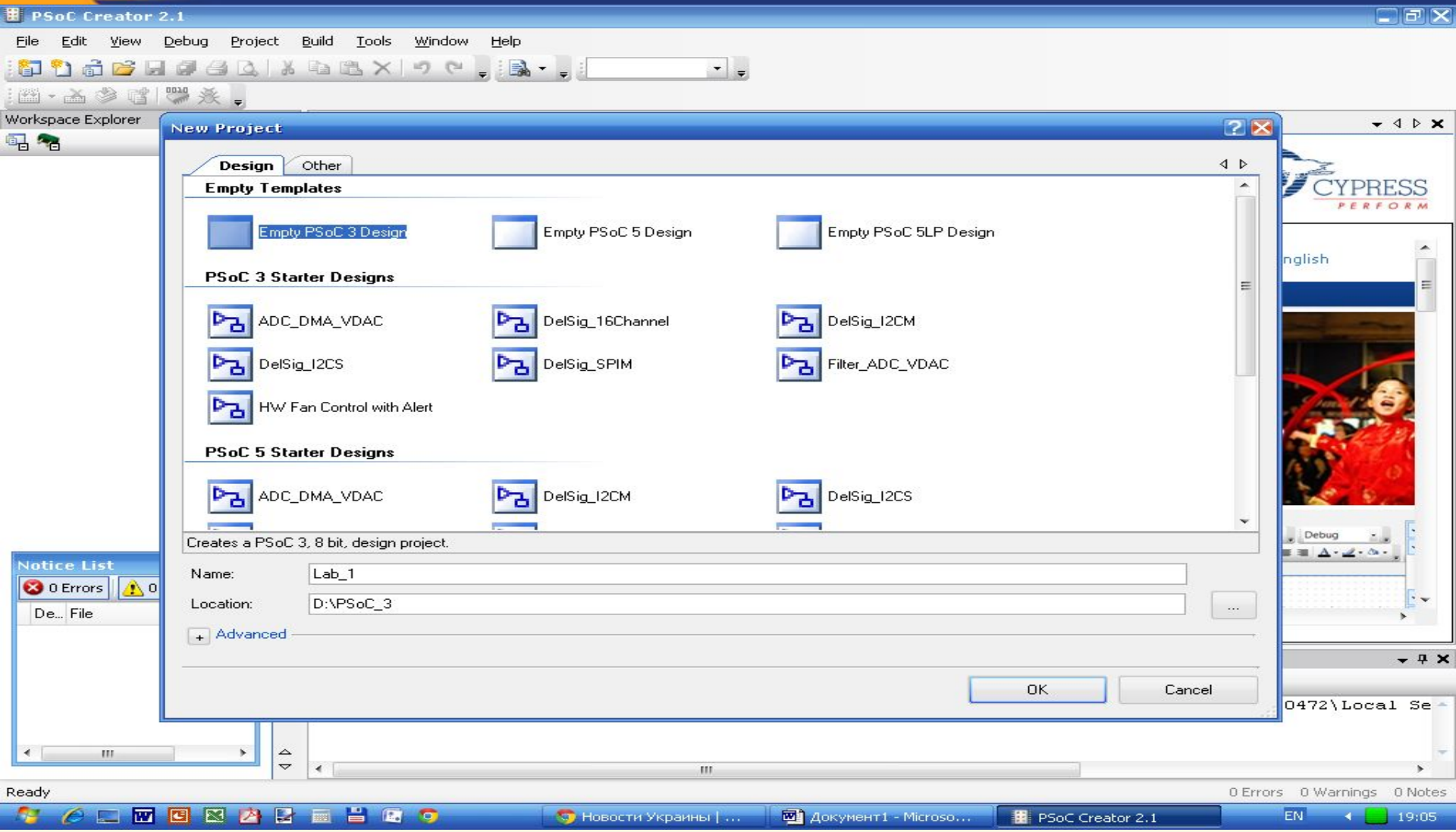
Creator


PSoC Creator 2.1
File Edit View Debug Project Build Tools Window Help
Workspace Explorer
Source
Components
Datasheets
Results
Start Page
PSoC® Creator™
CYPRESS PERFORM
Recent Projects
HelloWorld_Blinky01.cywrk
CapSense_CSD_Design01...
CapSense_CSD_Design01...
CharLCD_CustomFont01.c...
CharLCD_CustomFont01.c...
Create New Project...
Open Existing Project...
Getting Started
PSoC Creator Start Page
Quick Start Guide
Intro to PSoC
Intro to PSoC Creator
PSoC Creator Training
Help Tutorials
Getting Started With PSoC 3
Getting Started With PSoC 5
Examples and Kits
Find Example Project...
No Kit Packages Installed
简体中文 日本語 한국어 English
PSoC Creator News and Information
Happy Lunar New Year!
Posted on 02/11/2013
Gong Xi Fa Cai! As many of my friends and colleagues are celebrating the New Year and welcoming in the year of the water snake, I wanted to take a minute and wish you all well. May the New Year bring each of you prosperity, good luck and a new PSoC design.
...
Read More
Tips + Tricks: Menu Customization
Posted on 01/24/2013
Did you know you can create a customized menu in PSoC® Creator? Right click in a blank area of the top menu and select customize from the
Notice List
0 Errors
0 Warnings
De... File Error L
Output
Show output from: All
Log file for this session is located at: C:\Documents and Settings\Admin.MICROSOF-7D0472\Local Se
Ready
0 Errors 0 Warnings 0 Notes
Новости Украины | ...
Документ1 - Microso...
PSoC Creator 2.1
EN
18:57

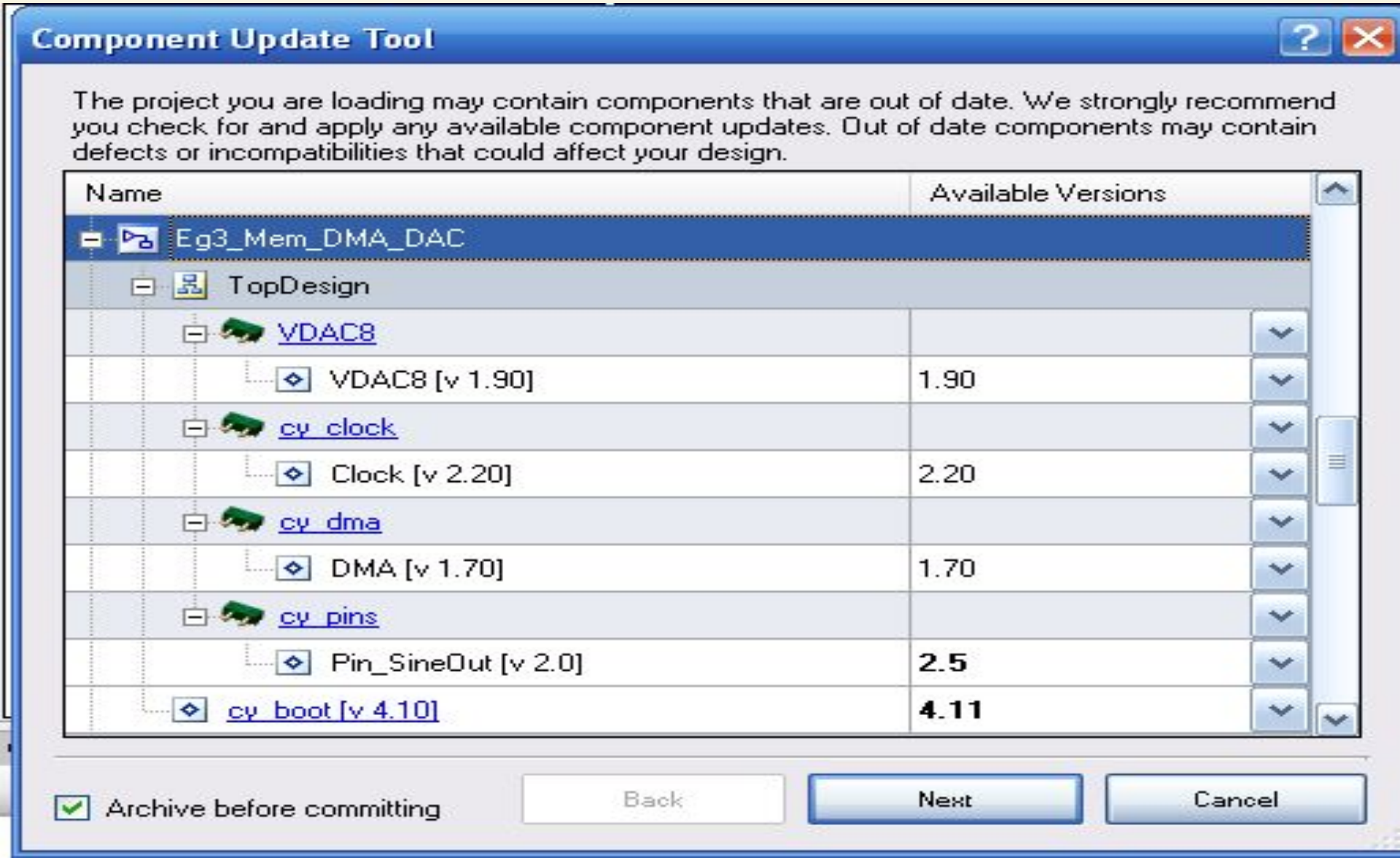
File – New - Projekt



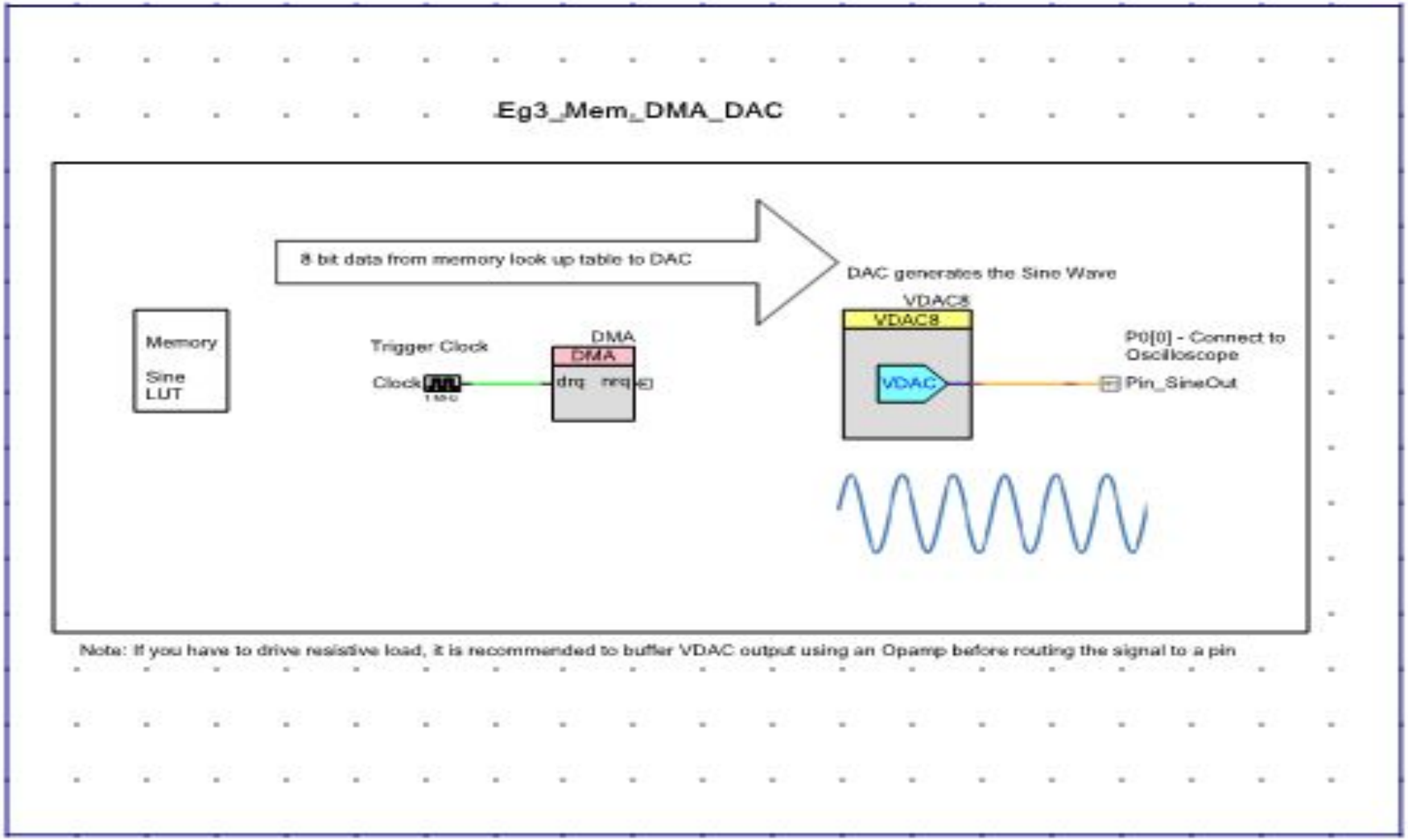
Empty PSoC 3.3 Design



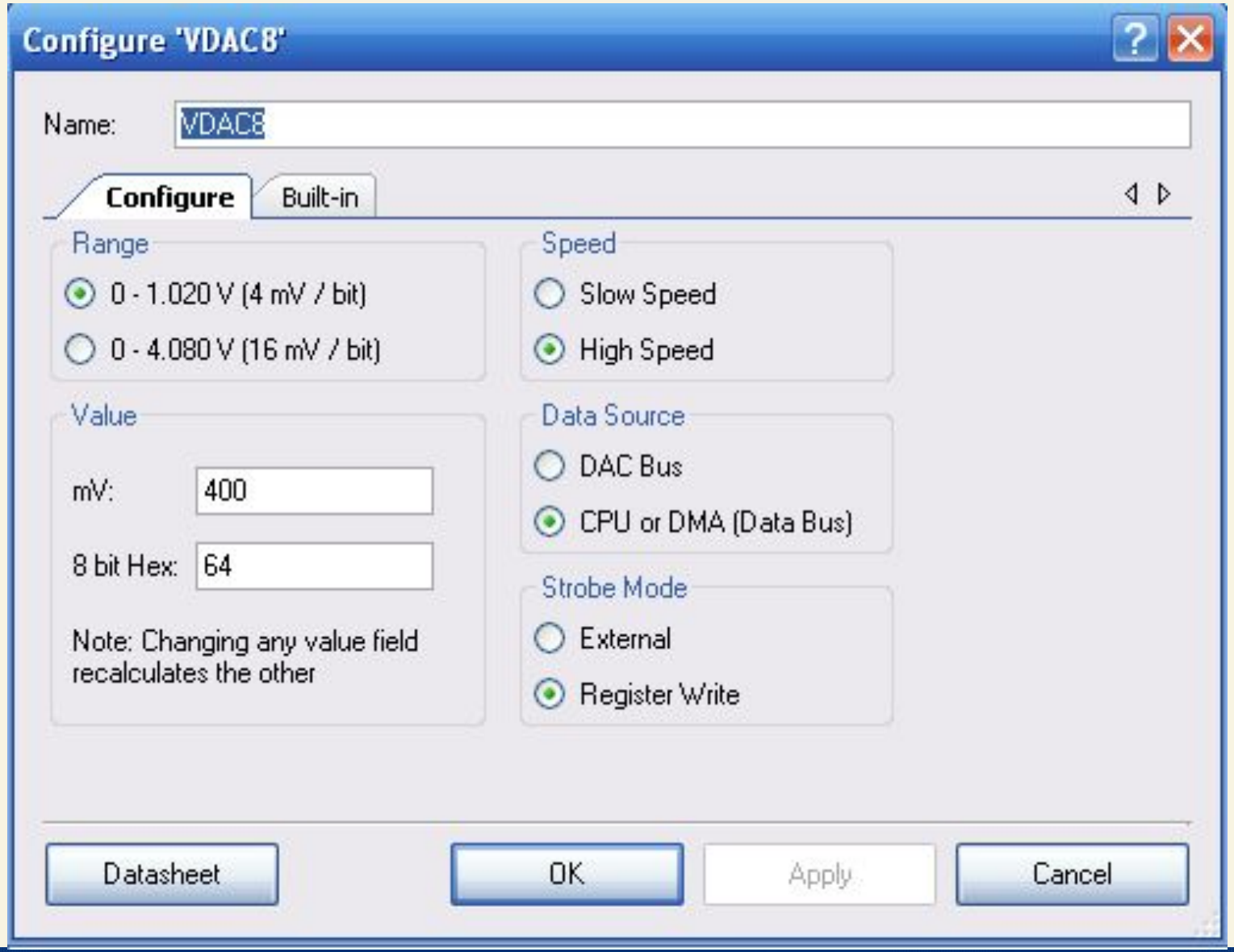
VDAC8+DMA


Component Update Tool
The project you are loading may contain components that are out of date. We strongly recommend you check for and apply any available component updates. Out of date components may contain defects or incompatibilities that could affect your design.
Name
Available Versions
Eg3_Mem_DMA_DAC
TopDesign
VDAC8
VDAC8 [v 1.90]
1.90
cy_clock
Clock [v 2.20]
2.20
cy_dma
DMA [v 1.70]
1.70
cy_pins
Pin_SineOut [v 2.0]
2.5
cy_boot [v 4.10]
4.11
Archive before committing
Back
Next
Cancel

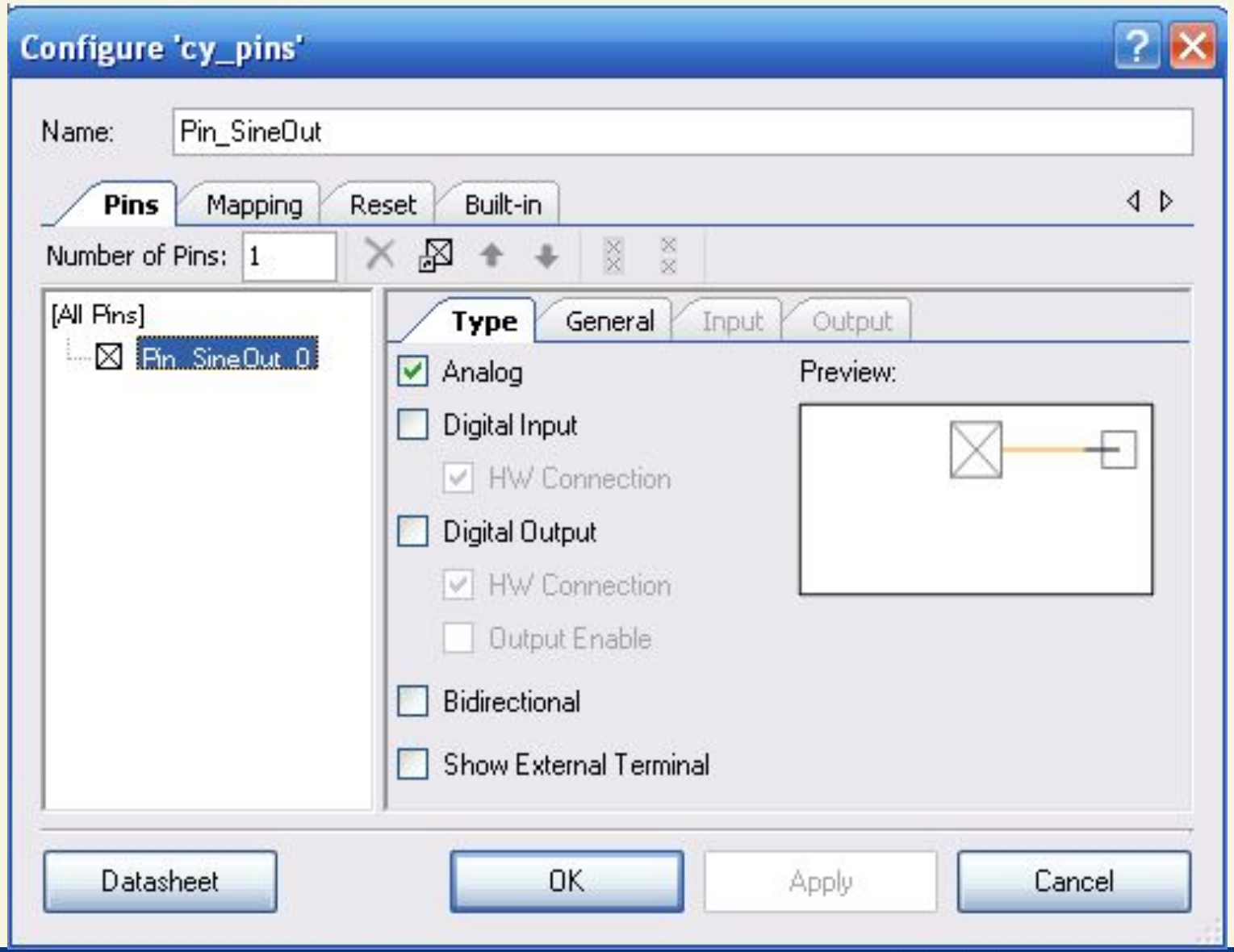
VDAC8+DMA


Eg3_Mem_DMA_DAC
8 bit data from memory look up table to DAC
DAC generates the Sine Wave
Memory
Sine
LUT
Trigger Clock
Clock
DMA
DMA
drq
nrq
VDAC8
VDAC8
VDAC
P0[0] - Connect to Oscilloscope
Pin_SineOut
Note: If you have to drive resistive load, it is recommended to buffer VDAC output using an Opamp before routing the signal to a pin

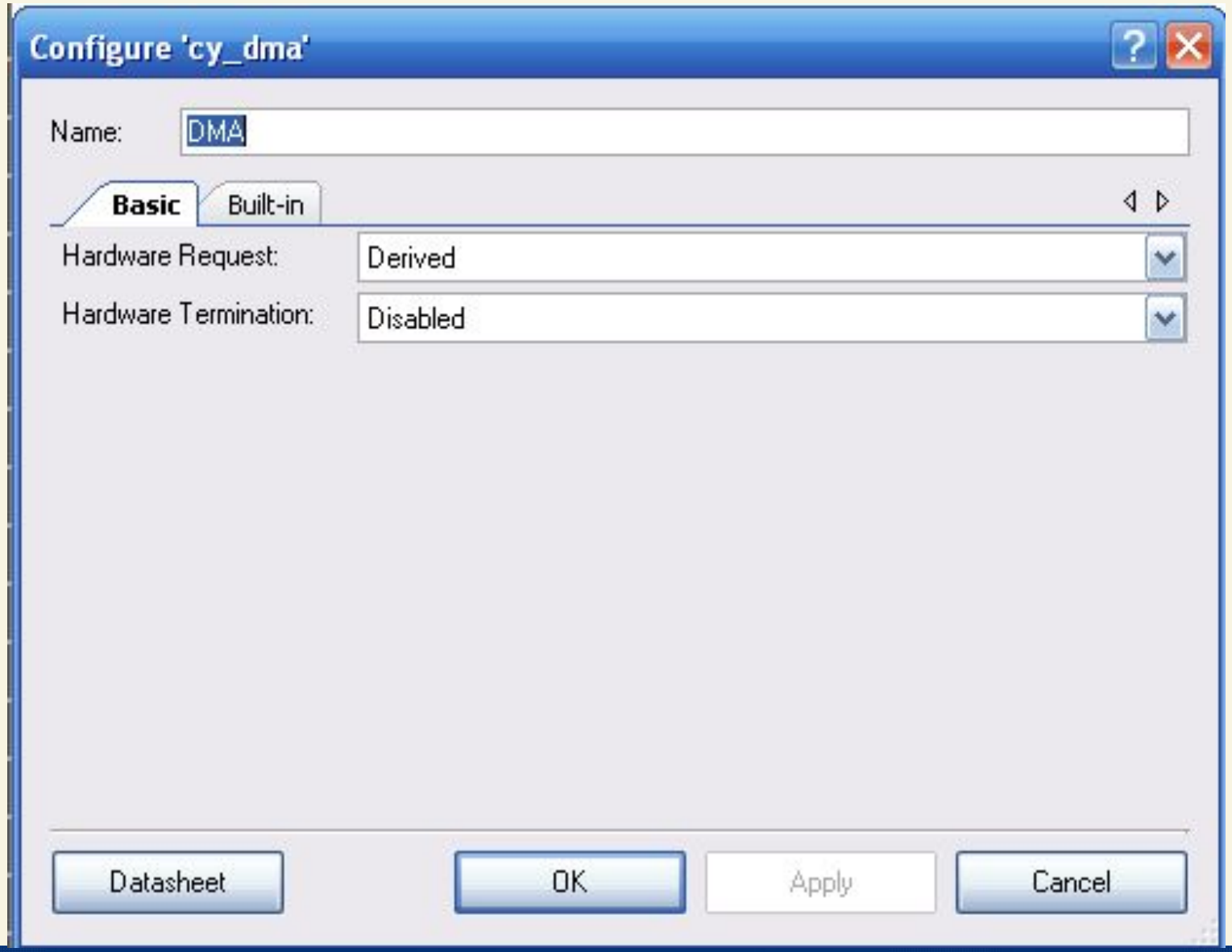
VDAC8+DMA


Configure 'VDAC8'
Name:
VDAC8
Configure
Built-in
Range
0 - 1.020 V (4 mV / bit)
0 - 4.080 V (16 mV / bit)
Value
mV:
400
8 bit Hex:
64
Note: Changing any value field recalculates the other
Speed
Slow Speed
High Speed
Data Source
DAC Bus
CPU or DMA (Data Bus)
Strobe Mode
External
Register Write
Datasheet
OK
Apply
Cancel

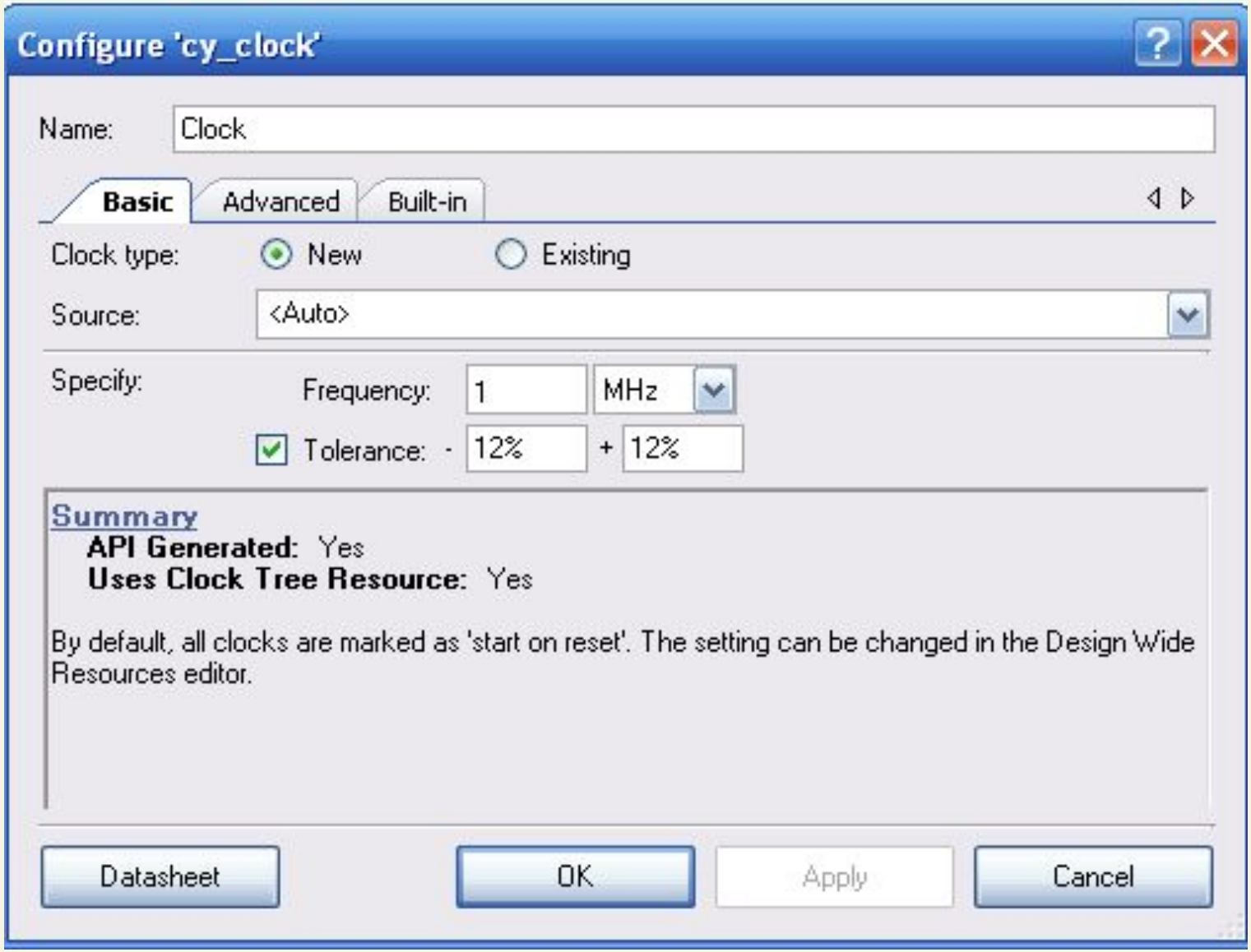
VDAC8


Configure 'cy_pins'
Name:
Pin_SineOut
Pins
Mapping
Reset
Built-in
Number of Pins:
1
[All Pins]
Pin_SineOut_0
Type
General
Input
Output
Analog
Digital Input
HW Connection
Digital Output
HW Connection
Output Enable
Bidirectional
Show External Terminal
Preview:
Datasheet
OK
Apply
Cancel

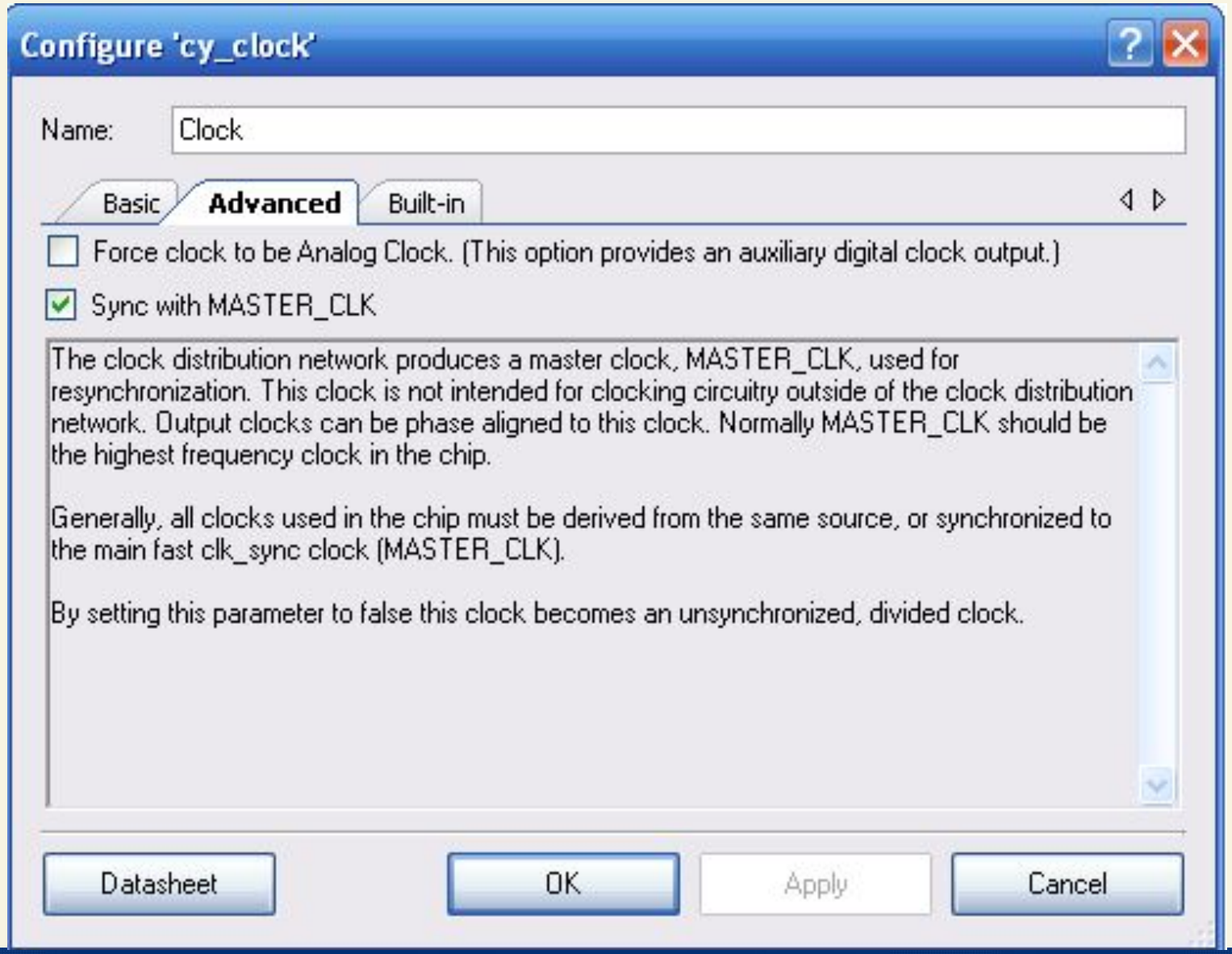
VDAC8


Configure 'cy_dma'
Name:
DMA
Basic
Built-in
Hardware Request:
Derived
Hardware Termination:
Disabled
Datasheet
OK
Apply
Cancel

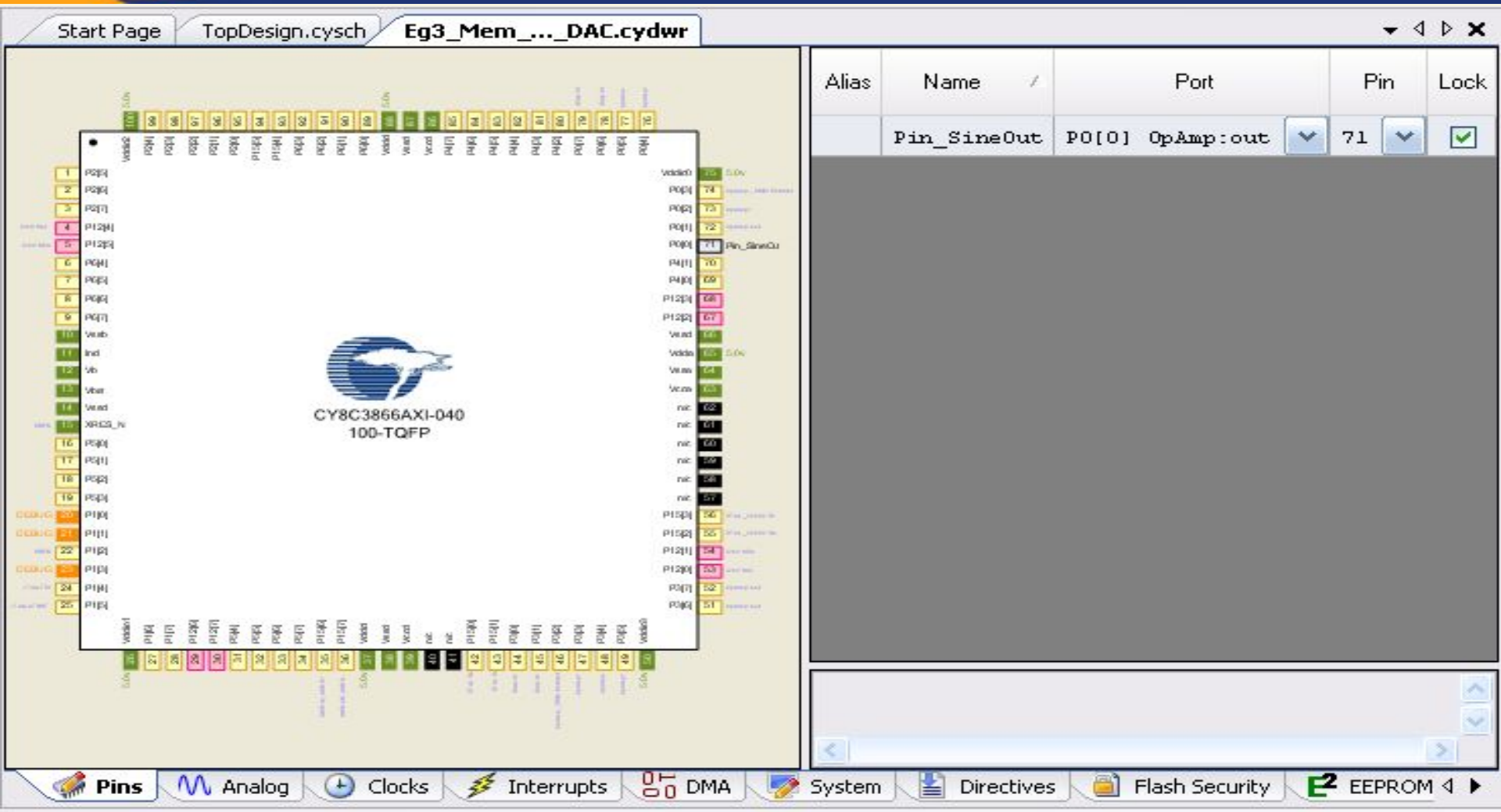
VDAC8


Configure 'cy_clock'
Name:
Clock
Basic
Advanced
Built-in
Clock type:
New
Existing
Source:
<Auto>
Specify:
Frequency:
1
MHz
Tolerance:
-
12%
+
12%
Summary
API Generated: Yes
Uses Clock Tree Resource: Yes
By default, all clocks are marked as 'start on reset'. The setting can be changed in the Design Wide Resources editor.
Datasheet
OK
Apply
Cancel

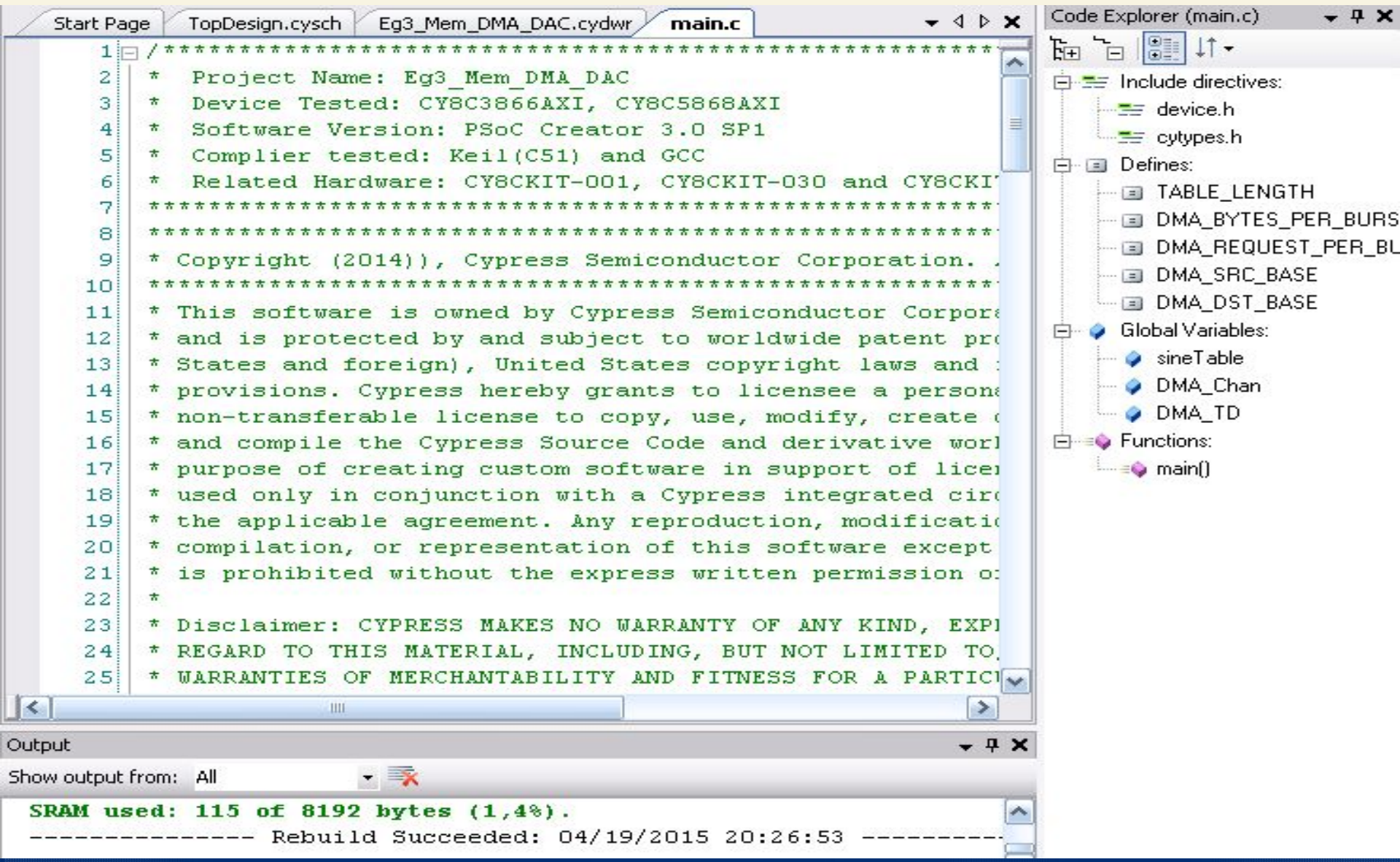
VDAC8


Configure 'cy_clock'
Name:
Clock
Basic
Advanced
Built-in
Force clock to be Analog Clock. (This option provides an auxiliary digital clock output.)
Sync with MASTER_CLK
The clock distribution network produces a master clock, MASTER_CLK, used for resynchronization. This clock is not intended for clocking circuitry outside of the clock distribution network. Output clocks can be phase aligned to this clock. Normally MASTER_CLK should be the highest frequency clock in the chip.
Generally, all clocks used in the chip must be derived from the same source, or synchronized to the main fast clk_sync clock (MASTER_CLK).
By setting this parameter to false this clock becomes an unsynchronized, divided clock.
Datasheet
OK
Apply
Cancel

VDAC8+DMA


Start Page
TopDesign.cysch
Eg3_Mem_..._DAC.cydwr
CY8C3866AXI-040
100-TQFP
Alias
Name
Port
Pin
Lock
Pin_SineOut
P0[0] OpAmp:out
71
Pins
Analog
Clocks
Interrupts
DMA
System
Directives
Flash Security
EEPROM
Output
Show output from:
All
----------Components are updated successfully----------

VDAC8+DMA


Start Page
TopDesign.cysch
Eg3_Mem_DMA_DAC.cydwr
main.c
/*****
* Project Name: Eg3_Mem_DMA_DAC
* Device Tested: CY8C3866AXI, CY8C5868AXI
* Software Version: PSoC Creator 3.0 SP1
* Complier tested: Keil(C51) and GCC
* Related Hardware: CY8CKIT-001, CY8CKIT-030 and CY8CKI
* Copyright (2014)), Cypress Semiconductor Corporation.
* This software is owned by Cypress Semiconductor Corpor
* and is protected by and subject to worldwide patent pr
* States and foreign), United States copyright laws and
* provisions. Cypress hereby grants to licensee a perso
* non-transferable license to copy, use, modify, create
* and compile the Cypress Source Code and derivative wor
* purpose of creating custom software in support of lice
* used only in conjunction with a Cypress integrated cir
* the applicable agreement. Any reproduction, modificati
* compilation, or representation of this software except
* is prohibited without the express written permission o
* Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXP
* REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTIC
Code Explorer (main.c)
Include directives:
device.h
cytypes.h
Defines:
TABLE_LENGTH
DMA_BYTES_PER_BURS
DMA_REQUEST_PER_BU
DMA_SRC_BASE
DMA_DST_BASE
Global Variables:
sineTable
DMA_Chan
DMA_TD
Functions:
main()
Output
Show output from: All
SRAM used: 115 of 8192 bytes (1,4%).
Rebuild Succeeded: 04/19/2015 20:26:53

VDAC8+DMA

Start Page | TopDesign.cysch | Eg3_Mem_DMA_DAC.cydwr | **main.c**

```
/*******************************************************************************
*  Project Name: Eg3_Mem_DMA_DAC
*  Device Tested: CY8C3866AXI, CY8C5868AXI
*  Software Version: PSoC Creator 3.0 SP1
*  Complier tested: Keil(C51) and GCC
*  Related Hardware: CY8CKIT-001, CY8CKIT-030 and CY8CKIT-050
********************************************************************************
********************************************************************************
* Copyright (2014)), Cypress Semiconductor Corporation. All Rights Reserved.
********************************************************************************
* This software is owned by Cypress Semiconductor Corporation (Cypress)
* and is protected by and subject to worldwide patent protection (United
* States and foreign), United States copyright laws and international treaty
* provisions. Cypress hereby grants to licensee a personal, non-exclusive,
* non-transferable license to copy, use, modify, create derivative works of,
* and compile the Cypress Source Code and derivative works for the sole
* purpose of creating custom software in support of licensee product to be
* used only in conjunction with a Cypress integrated circuit as specified in
* the applicable agreement. Any reproduction, modification, translation,
* compilation, or representation of this software except as specified above
* is prohibited without the express written permission of Cypress.
*
* Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH
* REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
```

VDAC8+DMA

Start Page | TopDesign.cysch | Eg3_Mem_DMA_DAC.cydwr | **main.c**

```
26  * Cypress reserves the right to make changes without further notice to the
27  * materials described herein. Cypress does not assume any liability arising out
28  * of the application or use of any product or circuit described herein. Cypress
29  * does not authorize its products for use as critical components in life-support
30  * systems where a malfunction or failure may reasonably be expected to result in
31  * significant injury to the user. The inclusion of Cypress' product in a life-
32  * support systems application implies that the manufacturer assumes all risk of
33  * such use and in doing so indemnifies Cypress against all charges.
34  *
35  * Use of this Software may be limited by and subject to the applicable Cypress
36  * software license agreement.
37  ******************************************************************************** */
38
39 /*******************************************************************************
40                                PROJECT DESCRIPTION
41  *******************************************************************************
42  * Sine Lookup of length 128 is created in a flash and these values are updated to
43  * at regular intervals, using DMA, in order to generate a sine wave.
44  * The update rate and the number of points in the sine look-up table determine th
45  * of the output sine wave.
46  * The values from the look-up table are updated into the DAC using a DMA.
47  * The DMA is set to update values on a hardware trigger.
48  * The hardware trigger is given by clock component.
49  * The output frequency of the sine wave equals update rate/number of points in th
50  * The example project generates a 7.8125 KHz sine wave with 128 points in the sin
```

VDAC8+DMA

Start Page | TopDesign.cysch | Eg3_Mem_DMA_DAC.cydwr | **main.c**

```
51  * update rate of  1MHz.
52  * Oscilloscope is connected to P0[0] to see the DAC output
53  ********************************************************************************
54
55  #include <device.h>
56  #include <cytypes.h>
57
58  #define TABLE_LENGTH 128
59  #define DMA_BYTES_PER_BURST 1
60  #define DMA_REQUEST_PER_BURST 1
61
62  /* This table stores the 128 points in Flash for smoother sine wave generation */
63  CYCODE const uint8 sineTable[TABLE_LENGTH] =
64  {
65      128, 134, 140, 147, 153, 159, 165, 171,
66      177, 183, 188, 194, 199, 204, 209, 214,
67      218, 223, 227, 231, 234, 238, 241, 244,
68      246, 248, 250, 252, 253, 254, 255, 255,
69      255, 255, 255, 254, 253, 252, 250, 248,
70      246, 244, 241, 238, 234, 231, 227, 223,
71      218, 214, 209, 204, 199, 194, 188, 183,
72      177, 171, 165, 159, 153, 147, 140, 134,
73      128, 122, 115, 109, 103,  97,  91,  85,
74       79,  73,  68,  62,  57,  52,  47,  42,
75       37,  33,  29,  25,  22,  18,  15,  12,
```

VDAC8+DMA

Start Page | TopDesign.cysch | Eg3_Mem_DMA_DAC.cydwr | **main.c**

```
73        128, 122, 115, 109, 103,  97,  91,  85,
74         79,  73,  68,  62,  57,  52,  47,  42,
75         37,  33,  29,  25,  22,  18,  15,  12,
76         10,   7,   6,   4,   2,   1,   1,   0,
77          0,   0,   1,   1,   2,   4,   6,   7,
78         10,  12,  15,  18,  22,  25,  29,  33,
79         37,  42,  47,  52,  57,  62,  68,  73,
80         79,  85,  91,  97, 103, 109, 115, 122
81    };
82
83    /* Variable declarations for DMA .
84     * These variables are defined as global variables to avoid "may be used before be
85     * issued by the PSoC 5 compilers  MDK/RVDS.In this case these variables are autor
86    uint8 DMA_Chan;                 /* The DMA Channel */
87    uint8 DMA_TD[1];                /* The DMA Transaction Descriptor (TD) */
88
89    int main()
90    {
91        /* Start VDAC  */
92        VDAC8_Start();
93
94        /* Defines for DMA configuration */
95        #if (defined(__C51__))  /* Source base address when PSoC3 is used */
96            #define DMA_SRC_BASE (CYDEV_FLS_BASE)
97        #else                   /* Source base address when PSoC5 is used */
```

VDAC8+DMA

Start Page | TopDesign.cysch | Eg3_Mem_DMA_DAC.cydwr | **main.c**

```
101  #define DMA_DST_BASE (CYDEV_PERIPH_BASE)  /* Destination base address */
102
103  /* Step1 : DmaInitialize - Initialize the DMA channel
104   * Bytes per burst = 1, (8 bit data transferred to VDAC one at a time)
105   * Request per burst = 1 (this will cause transfer of the bytes only with eve
106   * High byte of source address = Upper 16 bits of Flash Base address for PSoC
107                                  = HI16(&sineTable) for PSoC 5
108   * High byte of destination address =  Upper 16 bits of peripheral base addre
109  DMA_Chan = DMA_DmaInitialize(DMA_BYTES_PER_BURST, DMA_REQUEST_PER_BURST, HI16
110
111
112  /* Step2 :CyDmaTdAllocate - Allocate TD */
113  DMA_TD[0] = CyDmaTdAllocate();
114
115
116  /* Step3 :CyDmaTdSetConfiguration - Configures the TD:
117   * tdHandle = DMA_TD[0] - TD handle previously returned by CyDmaTdAlloc()
118   * Transfer count = table_length (number of bytes to transfer for a sine wave
119   * Next Td = DMA_TD[0] ; loop back to the same TD to generate a continous sig
120   * Configuration = The source address is incremented after every burst transf
121   */
122  CyDmaTdSetConfiguration(DMA_TD[0], TABLE_LENGTH, DMA_TD[0], TD_INC_SRC_ADR);
123
124
125  /* Step 4 : CyDmaTdSetAddress - Configure the lower 16 bit source and destina
```

VDAC8+DMA

Start Page | TopDesign.cysch | Eg3_Mem_DMA_DAC.cydwr | main.c

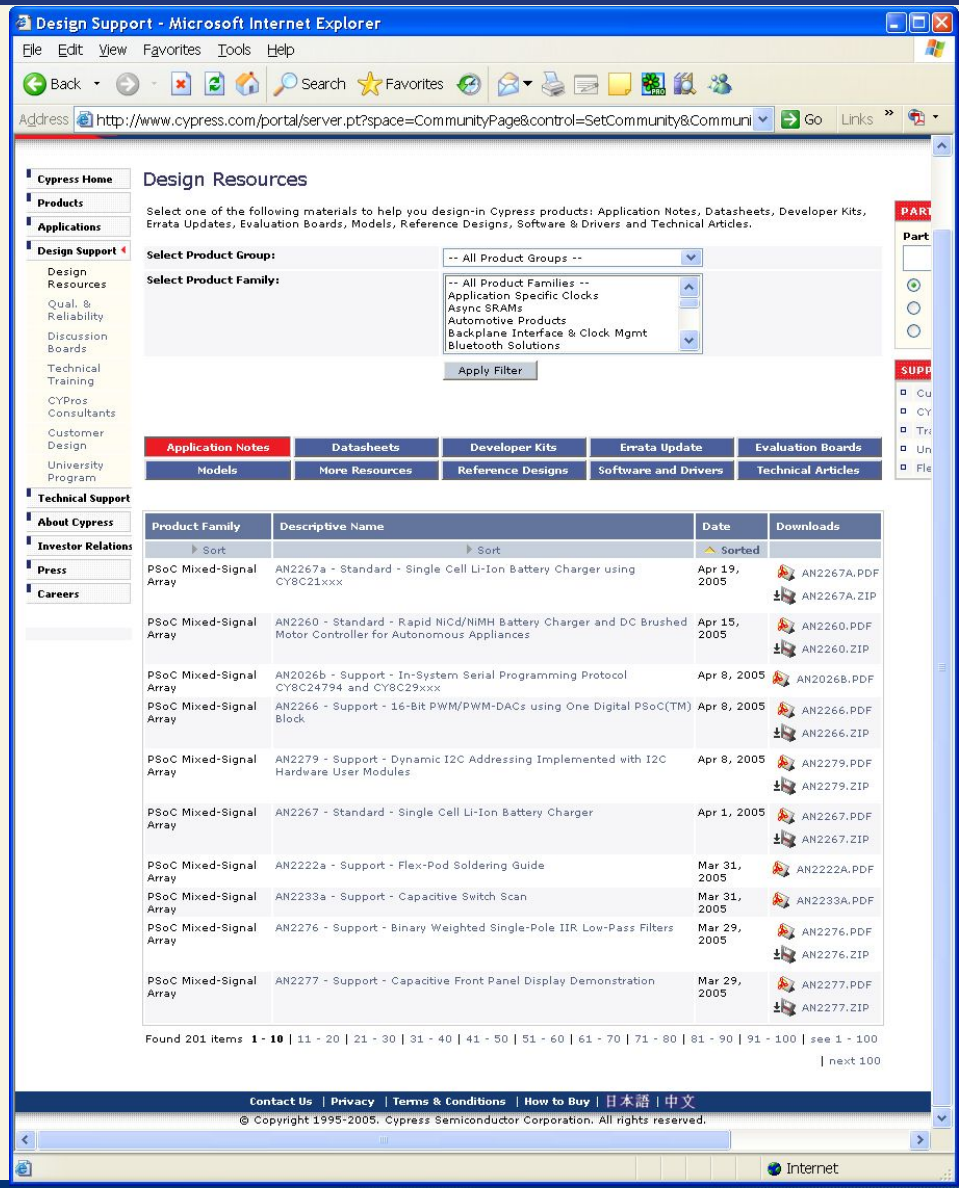
```
124
125     /* Step 4 : CyDmaTdSetAddress - Configure the lower 16 bit source and destina
126      * Source address = Lower 16 bits of sineTable array
127      * Destination address = Lower 16 bits of VDAC8_Data_PTR register */
128     CyDmaTdSetAddress(DMA_TD[0], LO16((uint32)sineTable), LO16((uint32)VDAC8_Data
129
130
131     /* Step 5: Map the TD to the DMA Channel */
132     CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);
133
134
135     /* Step 6: Enable the DMA channel */
136     CyDmaChEnable(DMA_Chan, 1);
137
138
139     for(;;)
140     {
141         /* Your code here. */
142         /* The sine look up table data is moved to the DAC by the DMA in the back
143          * without any CPU intervention */
144     }
145 }
146
147 /* [] END OF FILE */
148
```

Test VDAC8+DMA

Example : Memory-to-Peripheral Transfer
– Mem_DMA_DAC
The test setup is as follows:
1. Connect the oscilloscope probe to pin P0[0], the VDAC output.
2. Build the project and program the device.
3. Observe a sine wave of frequency 7.8 kHz on the oscilloscope.

Приклади застосування МК PSOC

На сайті фірми Cypress знаходиться більше 200 Application Notes і Reference Designs, які ілюструють області застосування мікроконтролерів PSoC.



Мікропроцесорна техніка

(лекція 10, кінець)

Благітко Б.Я.

2019 р.