Мікропроцесорна техніка

(лекція 7)
Благітко Б.Я.
2019 р.

PSoC Creator 4.2
Designing with PSoC 3/5

PSoC@3/5
Fan Controller

PSoC Creator 4.2
Designing with PSoC 3/5

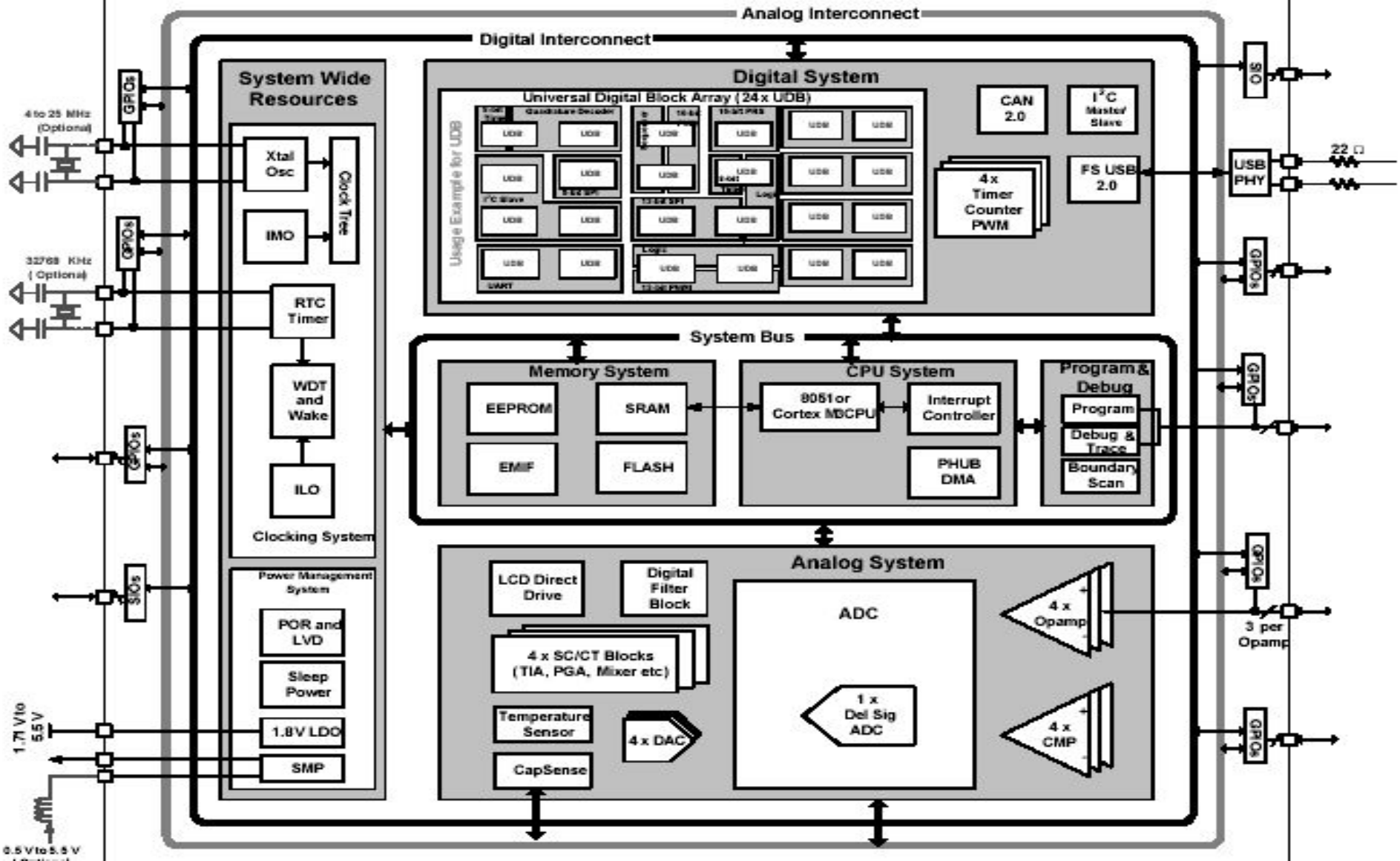
PSoC@3/5

Управління швидкістю обертання ротора двигунів постійного струму

PSoC Creator 4.2
Designing with PSoC 3/5

Цифрові модулі

Figure 1-1. Simplified Block Diagram



Цифрові модулі

Figure 7-3. Component Catalog

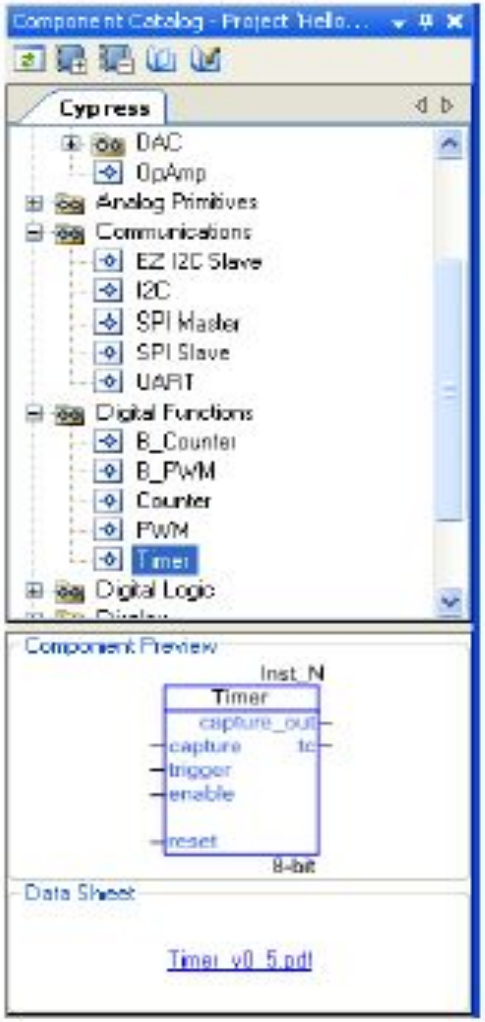
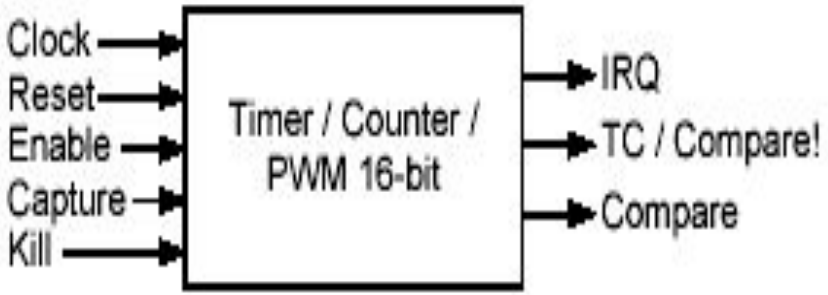
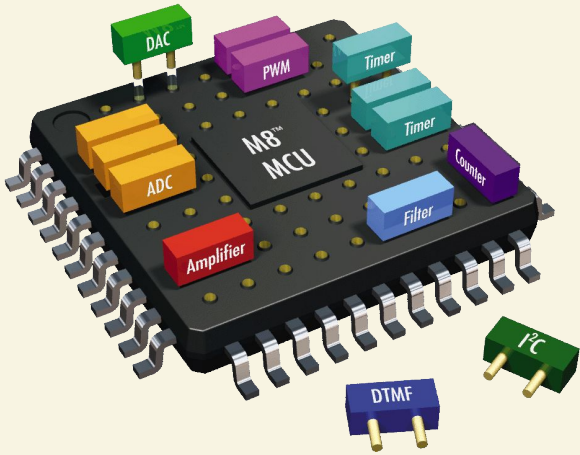


Figure 7-21. Timer/Counter/PWM





PWMs and Counters

PWMs and **Counter**s share many capabilities but each provides specific capabilities.

When to Use a PWM

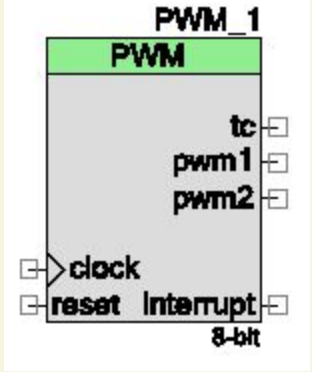
The most common use of the **PWM** is to generate periodic waveforms with adjustable duty cycles. The PWM also provides optimized features for power control, motor control, switching regulators and lighting control. The PWM can also be used as a clock divider by driving a clock into the clock input and using the terminal count or a PWM output as the divided clock output.

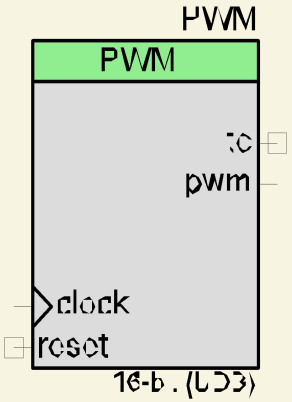
When to Use a Counter

A **Counter** component is better used in situations that require the counting of a number of events but also provides rising edge capture input as well as a compare output.

PWM

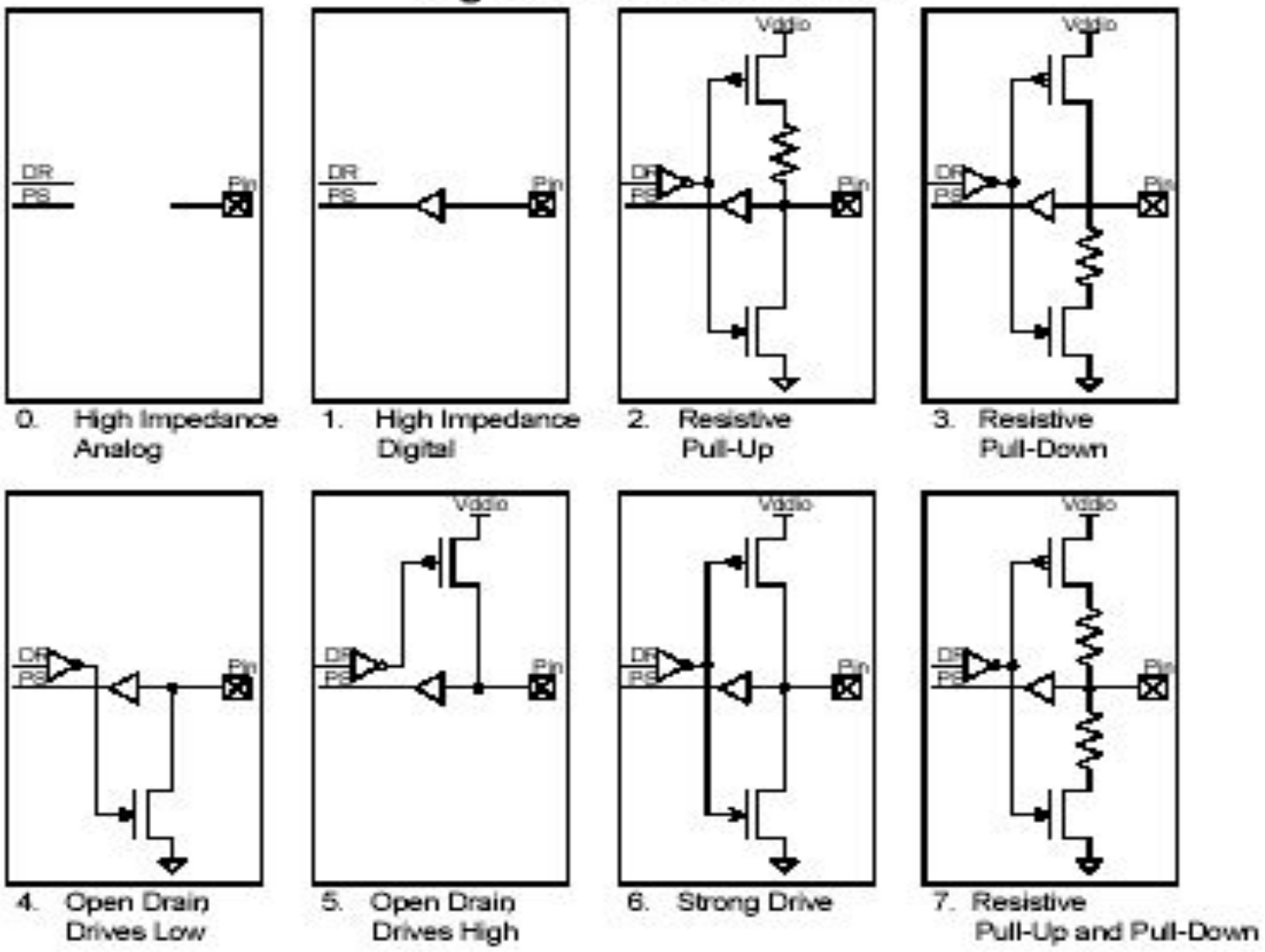
Output	May Be Hidden	Description
tc	N	The terminal count output is '1' when the period counter is equal to zero. In normal operation this output will be '1' for a single cycle where the counter is reloaded with period. If the PWM is stopped with the period counter equal to zero then this signal will remain high until the period counter is no longer zero. This output is synchronized to the block clock input of the component.
interrupt	Y	The interrupt output is the logical OR of the group of possible interrupt sources. This signal will go high while any of the enabled interrupt sources are true. The interrupt output shall remain asserted until the Status Register is read out by the software. In order to receive subsequent interrupts, the interrupt shall be cleared by reading the Status Register using the PWM_ReadStatusRegister() API. The interrupt output is not visible if the Use Interrupt parameter is not set. This allows the status register to be removed for resource optimization as necessary.
pwm/pwm1	Y	The pwm or pwm1 output is the first or only pulse width modulated output. This signal is defined by PWM Mode, compare modes(s), and compare value(s) as indicated in waveforms in the Configure dialog. When the instance is configured in one output, Dual Edged, Hardware Select, Center Aligned, or Dither PWM Modes, then the output "pwm" is visible. Otherwise the output "pwm1" is visible with "pwm2" the other pulse width signal. This output is synchronized to the block clock input of the component.





Port's Pins

Figure 6-11. Drive Mode



How They Work

A DC motor uses a commutator to supply continuous direct current to a coil placed between two or more magnets.

A stepper motor receives pulses of electricity from a pulse generator and sends them through a similar coil-magnet setup.

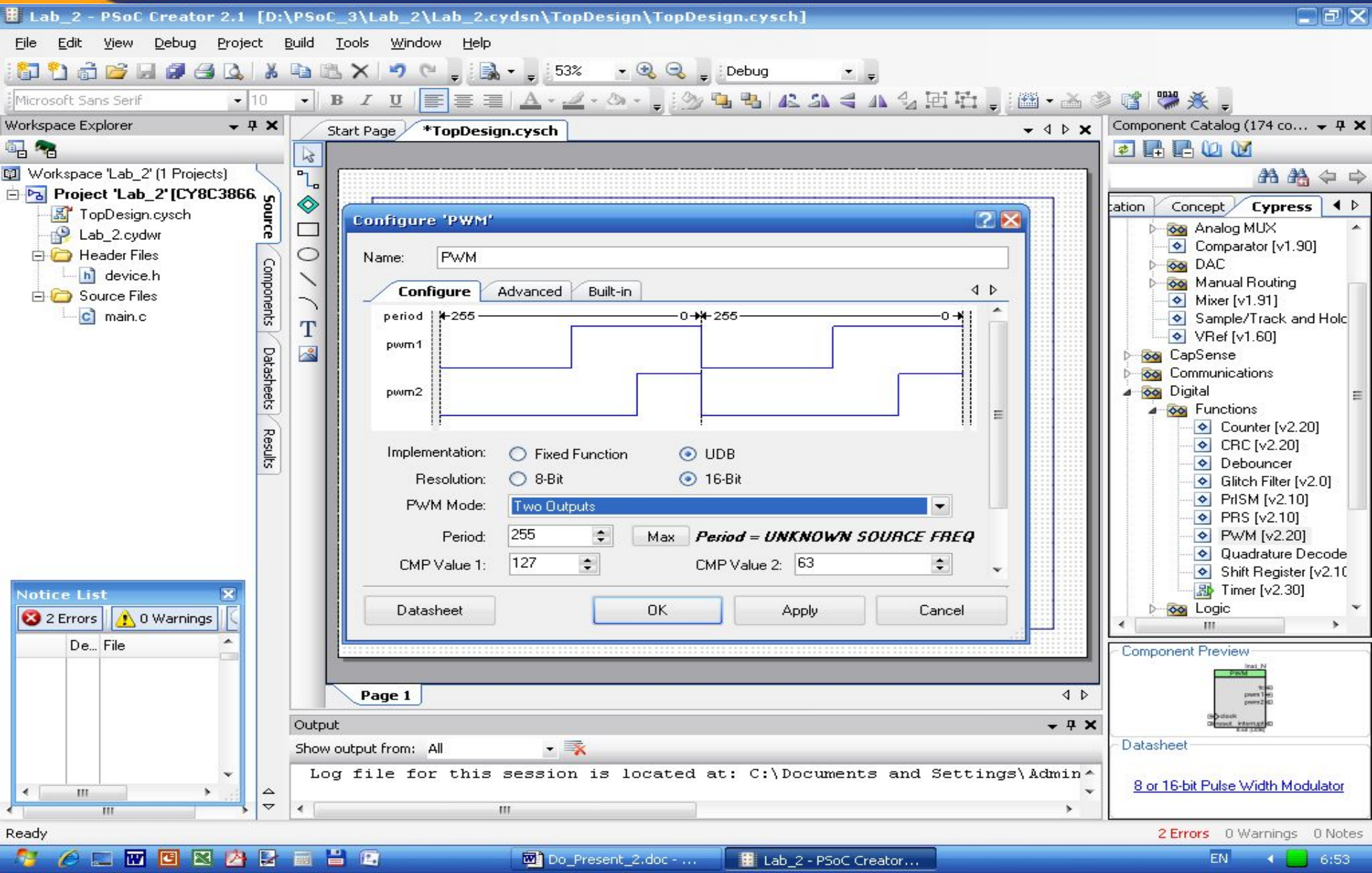
In both cases, the Oersted effect causes the coils to turn the motor's shaft.

Read more: DC Motor vs. Stepper Motor | eHow.com http://www.ehow.com/facts_6777501_dc-motor-vs_-stepper-motor.html#ixzz2QKmmwrgy

When to Use a Fan Controller

Use the Fan Controller component when you want to provide your application with device.

Lab_2 PWM


Lab_2 - PSoC Creator 2.1 [D:\PSoC_3\Lab_2\Lab_2.cydsn\TopDesign\TopDesign.cysch]
File Edit View Debug Project Build Tools Window Help
Workspace Explorer
Workspace 'Lab_2' (1 Projects)
Project 'Lab_2' [CY8C3866
TopDesign.cysch
Lab_2.cydwr
Header Files
device.h
Source Files
main.c
Source
Components
Datasheets
Results
Start Page
*TopDesign.cysch
Configure 'PWM'
Name: PWM
Configure
Advanced
Built-in
period
255
0
pwm1
pwm2
Implementation: Fixed Function UDB
Resolution: 8-Bit 16-Bit
PWM Mode: Two Outputs
Period: 255
Max
Period = UNKNOWN SOURCE FREQ
CMP Value 1: 127
CMP Value 2: 63
Datasheet
OK
Apply
Cancel
Page 1
Component Catalog (174 co...
Concept
Cypress
Analog MUX
Comparator [v1.90]
DAC
Manual Routing
Mixer [v1.91]
Sample/Track and Holc
VRef [v1.60]
CapSense
Communications
Digital
Functions
Counter [v2.20]
CRC [v2.20]
Debouncer
Glitch Filter [v2.0]
PrISM [v2.10]
PRS [v2.10]
PWM [v2.20]
Quadrature Decode
Shift Register [v2.1
Timer [v2.30]
Logic
Component Preview
Datasheet
8 or 16-bit Pulse Width Modulator
Notice List
2 Errors
0 Warnings
De...
File
Output
Show output from: All
Log file for this session is located at: C:\Documents and Settings\Admin
Ready
2 Errors 0 Warnings 0 Notes
Do_Present_2.doc - ...
Lab_2 - PSoC Creator...
EN
6:53

Fan Controller

FanController

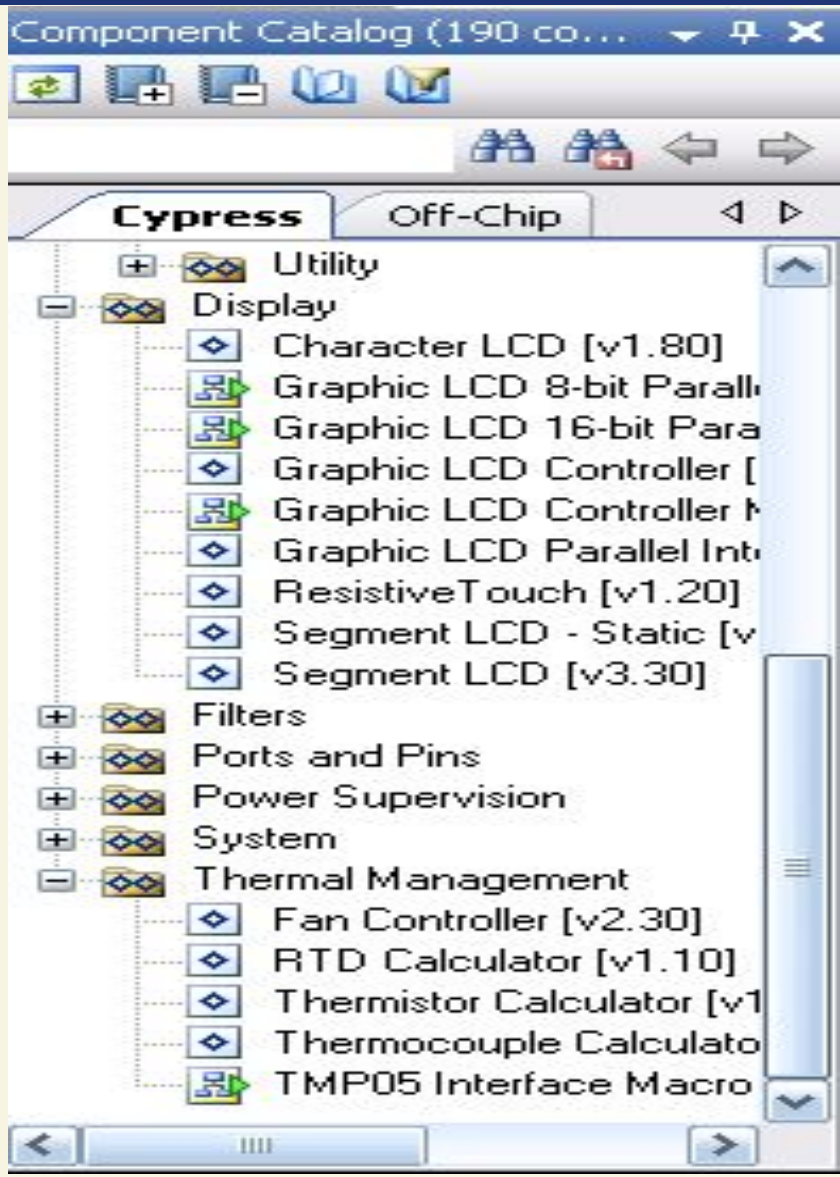
Компонента надає низькорівневий драйвер для управління швидкістю обертання ротора двигунів постійного струму . Крім того, ця компонента забезпечує для користувача настроювач, який полегшує налаштування дескрипторів

Fan Controller

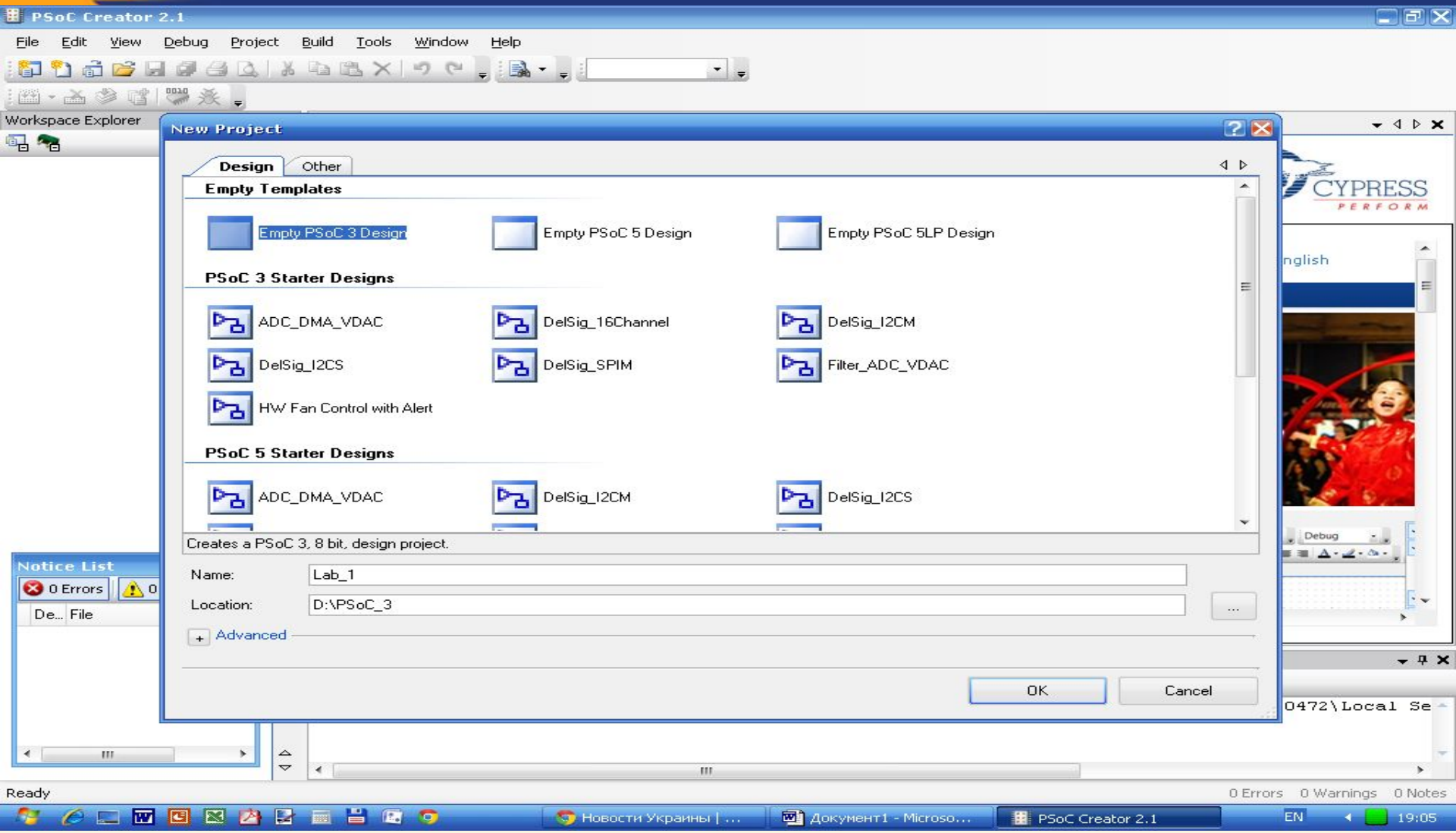
Компонента
в PSoC Creator 4.2,
яка працює з Fan

- Fan Controller

Ця компонента знаходиться в компонент каталозі, в категорії “Thermal Management”.

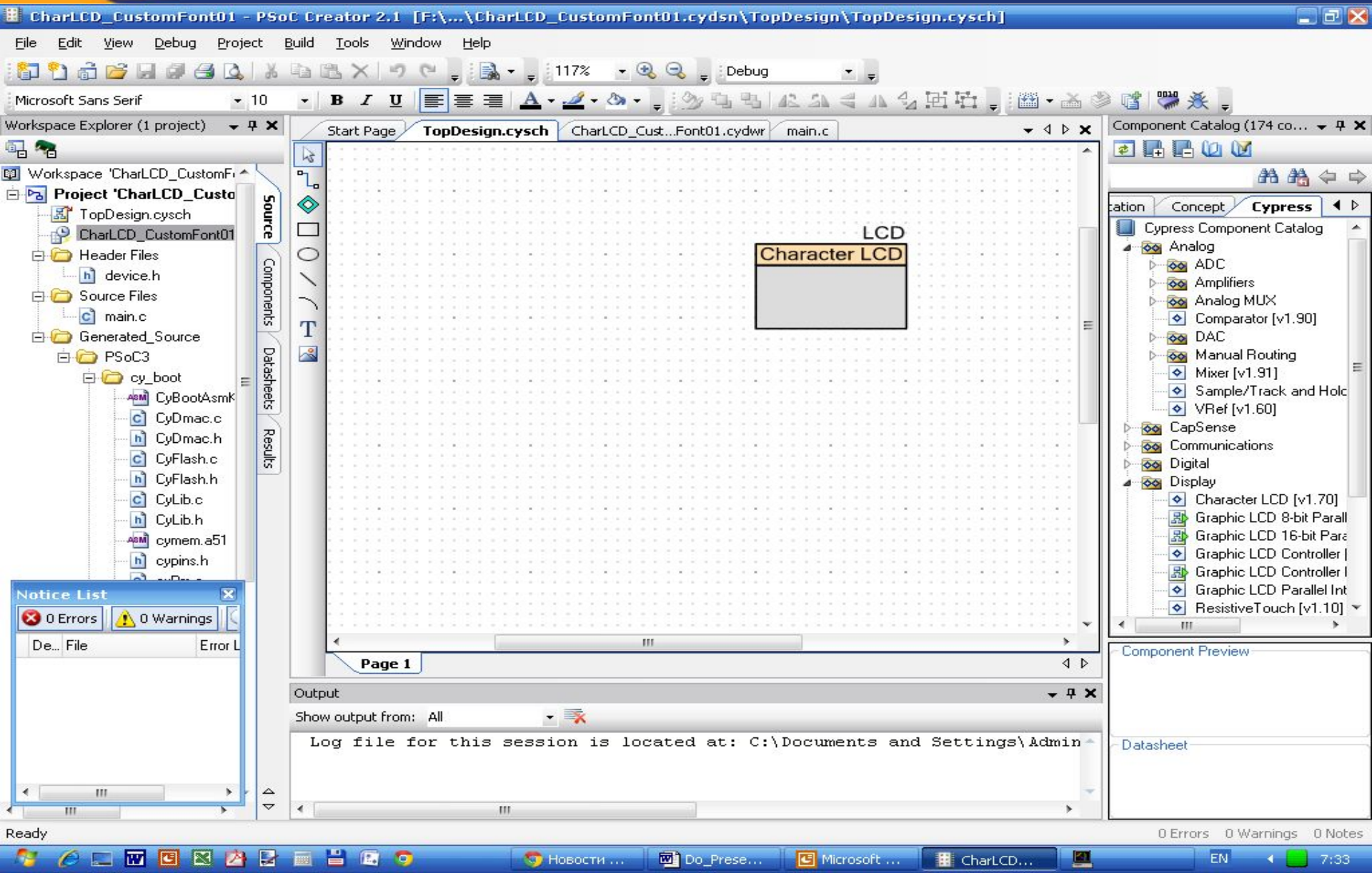


Empty PSoC 3/5 Design

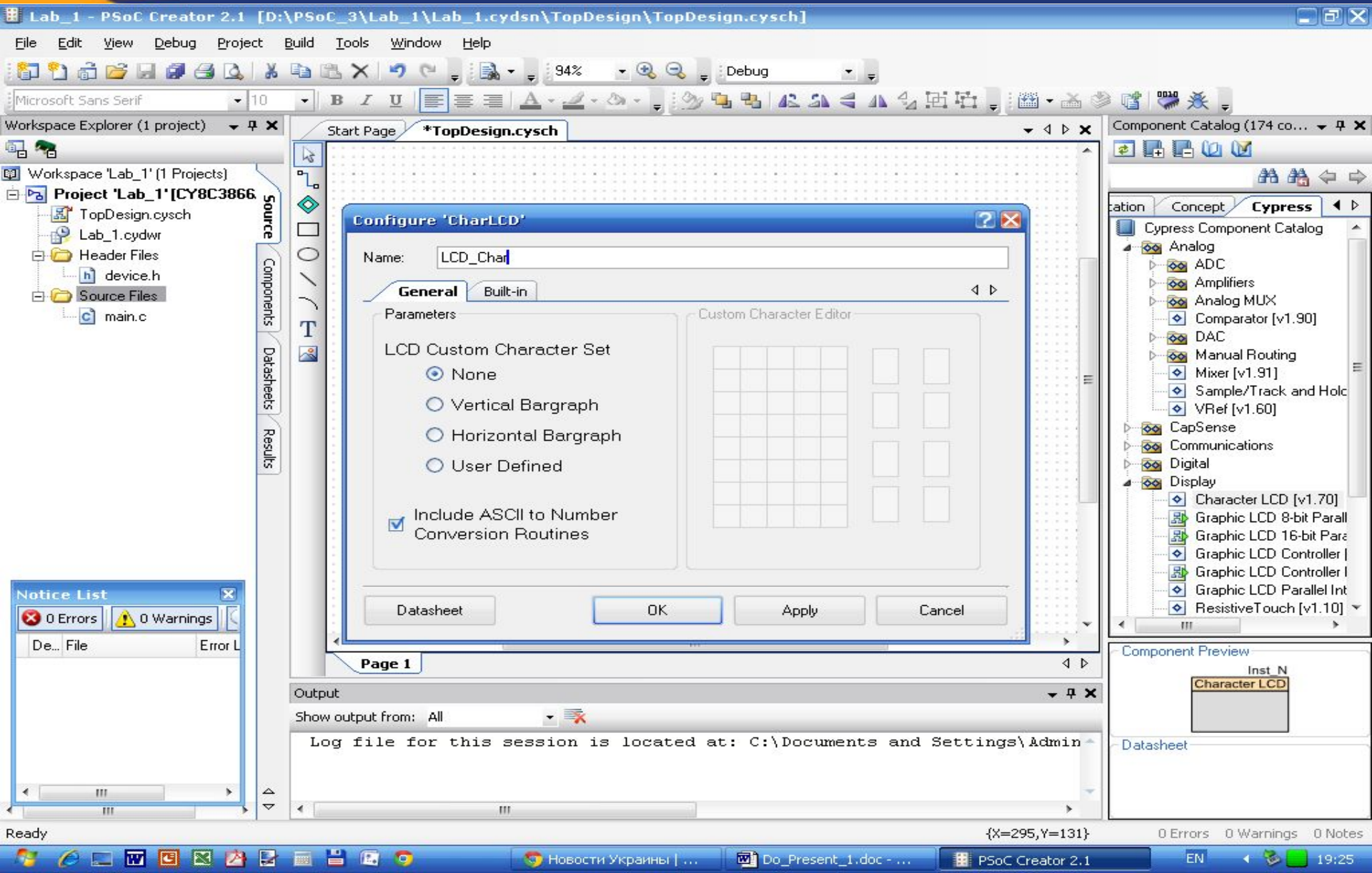


CHAPTER #LOW

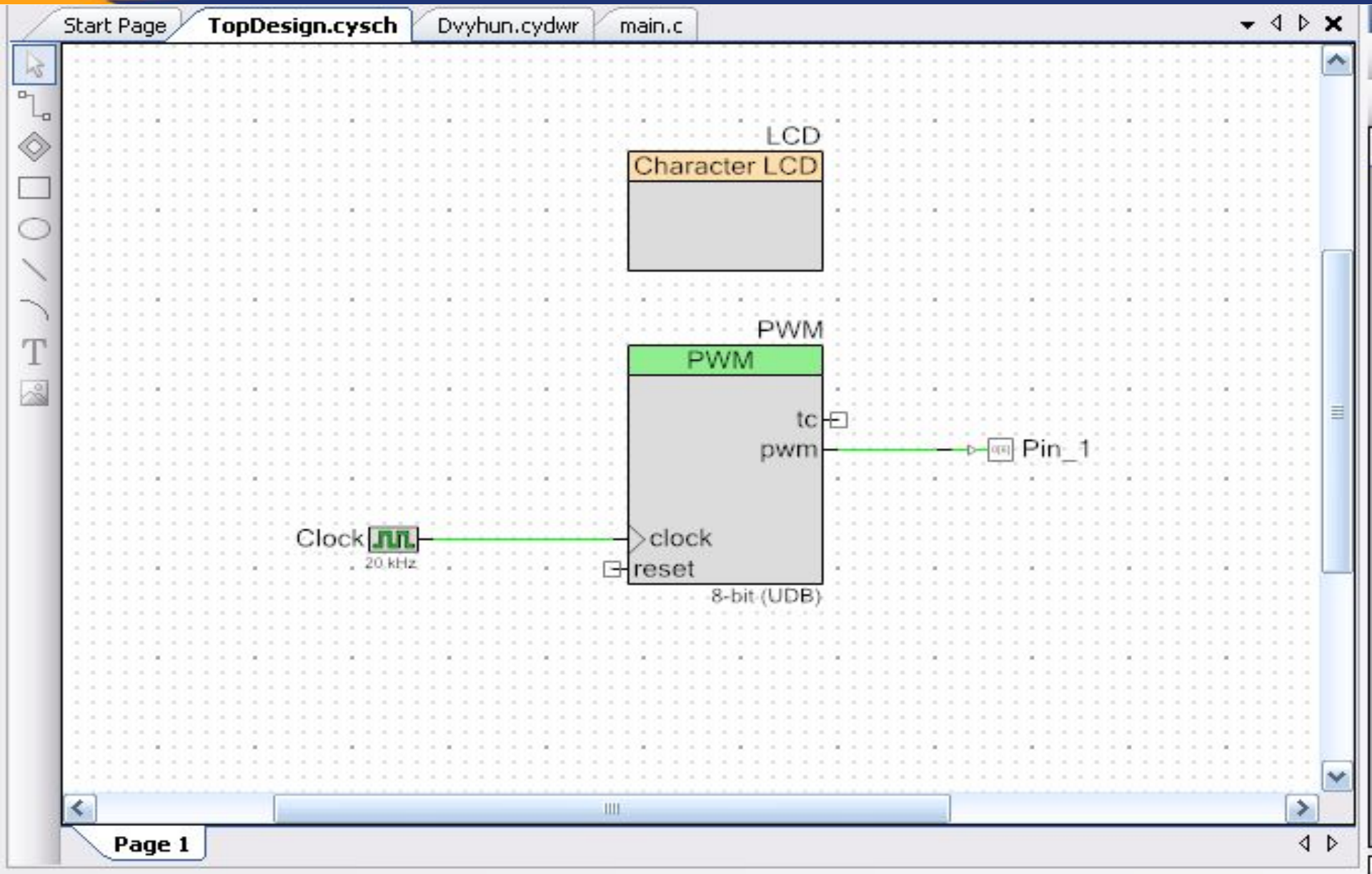
Lab_7 Fan


CharLCD_CustomFont01 - PSoC Creator 2.1 [F:\...\CharLCD_CustomFont01.cydsn\TopDesign\TopDesign.cysch]
File Edit View Debug Project Build Tools Window Help
Workspace Explorer (1 project)
Workspace 'CharLCD_CustomF
Project 'CharLCD_Custo
TopDesign.cysch
CharLCD_CustomFont01
Header Files
device.h
Source Files
main.c
Generated_Source
PSoC3
cy_boot
CyBootAsmK
CyDmac.c
CyDmac.h
CyFlash.c
CyFlash.h
CyLib.c
CyLib.h
cymem.a51
cypins.h
Notice List
0 Errors
0 Warnings
De... File Error L
Start Page
TopDesign.cysch
CharLCD_Cust...Font01.cydwr
main.c
LCD
Character LCD
Page 1
Output
Show output from: All
Log file for this session is located at: C:\Documents and Settings\Admin
Component Catalog (174 co...
Concept
Cypress
Cypress Component Catalog
Analog
ADC
Amplifiers
Analog MUX
Comparator [v1.90]
DAC
Manual Routing
Mixer [v1.91]
Sample/Track and Hold
VRef [v1.60]
CapSense
Communications
Digital
Display
Character LCD [v1.70]
Graphic LCD 8-bit Parall
Graphic LCD 16-bit Para
Graphic LCD Controller
Graphic LCD Controller
Graphic LCD Parallel Int
ResistiveTouch [v1.10]
Component Preview
Datasheet
Ready
0 Errors 0 Warnings 0 Notes
Новости ...
Do_Prese...
Microsoft ...
CharLCD...
EN
7:33

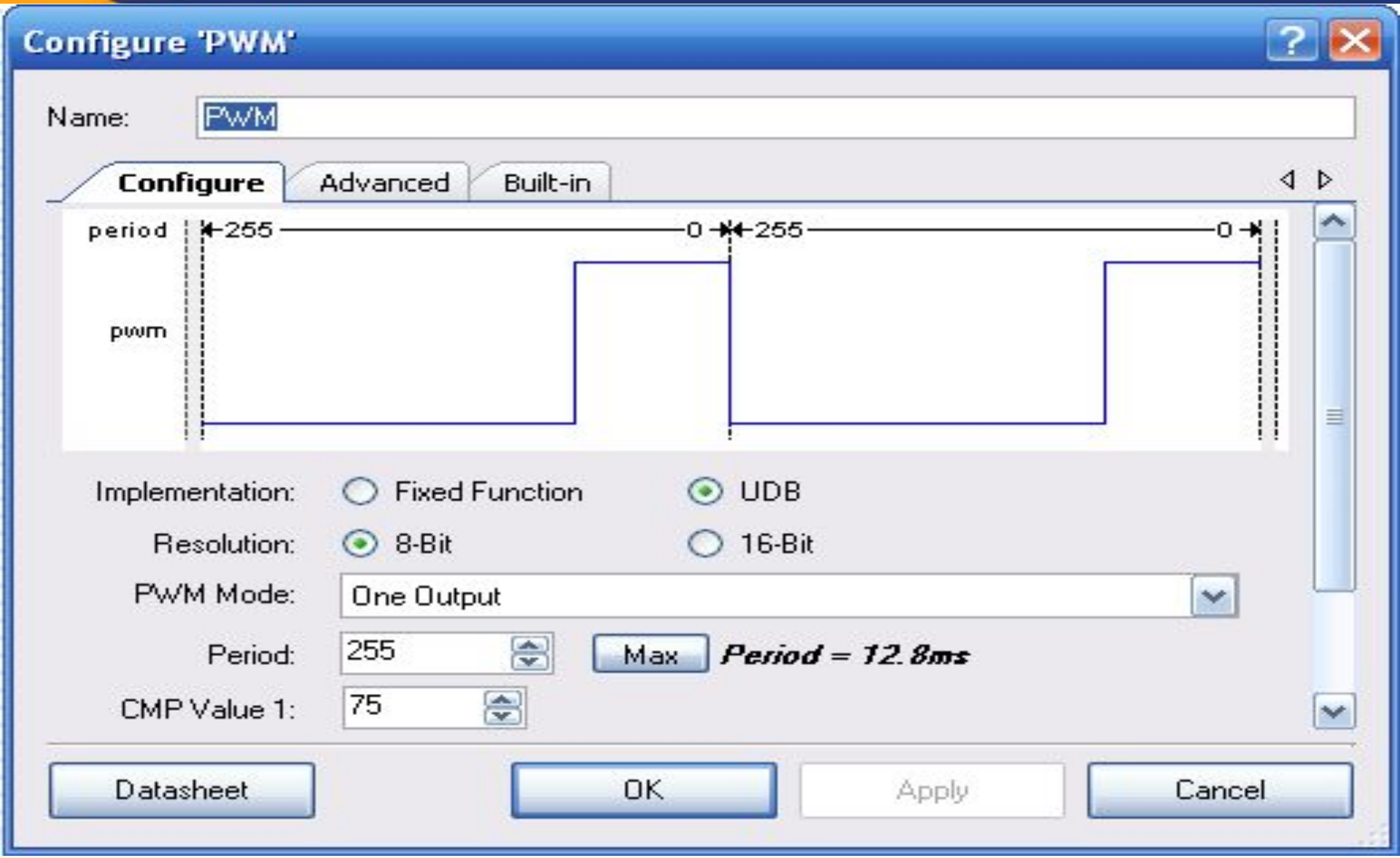
Configure LCD



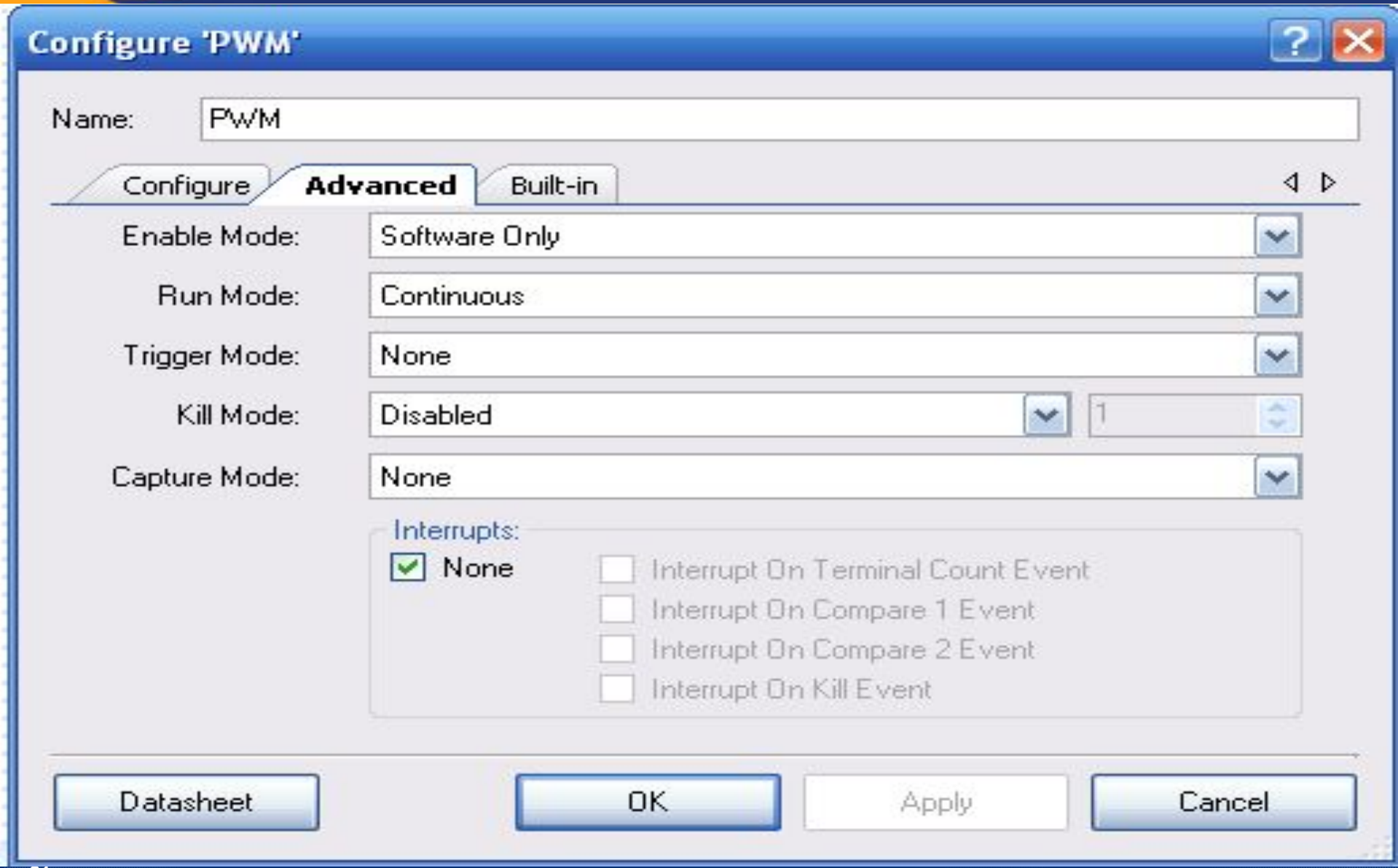
Lab_7 Fan


Start Page
TopDesign.cysch
Dvyhun.cydwr
main.c
LCD
Character LCD
PWM
PWM
tc
pwm
Pin_1
Clock
20 kHz
clock
reset
8-bit (UDB)
Page 1

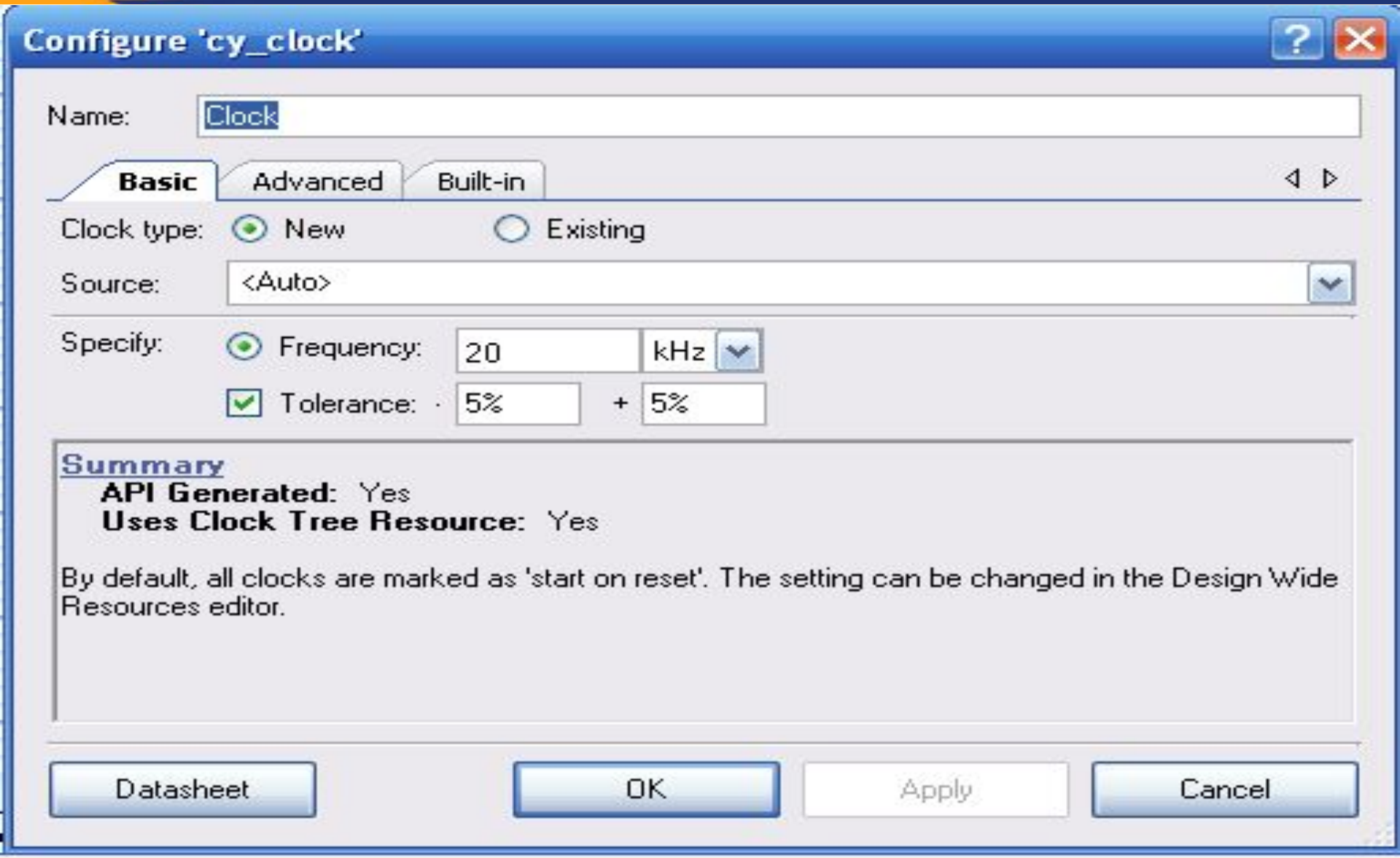
Lab_7 Fan


Configure 'PWM'
Name:
PWM
Configure
Advanced
Built-in
period
255
0
255
0
pwm
Implementation:
Fixed Function
UDB
Resolution:
8-Bit
16-Bit
PWM Mode:
One Output
Period:
255
Max
Period = 12.8ms
CMP Value 1:
75
Datasheet
OK
Apply
Cancel

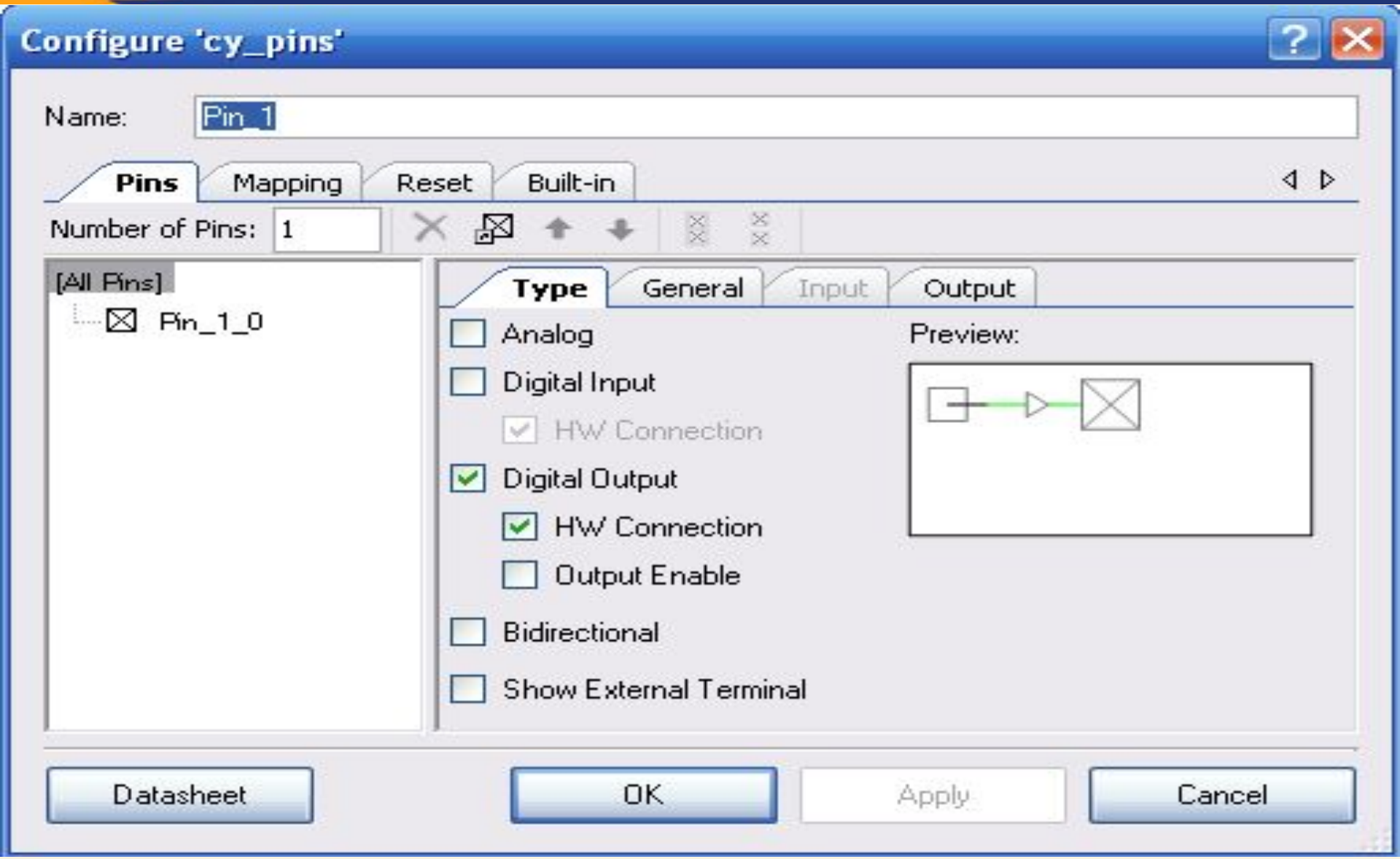
Lab_7 Fan


Configure 'PWM'
Name:
PWM
Configure
Advanced
Built-in
Enable Mode:
Software Only
Run Mode:
Continuous
Trigger Mode:
None
Kill Mode:
Disabled
1
Capture Mode:
None
Interrupts:
None
Interrupt On Terminal Count Event
Interrupt On Compare 1 Event
Interrupt On Compare 2 Event
Interrupt On Kill Event
Datasheet
OK
Apply
Cancel

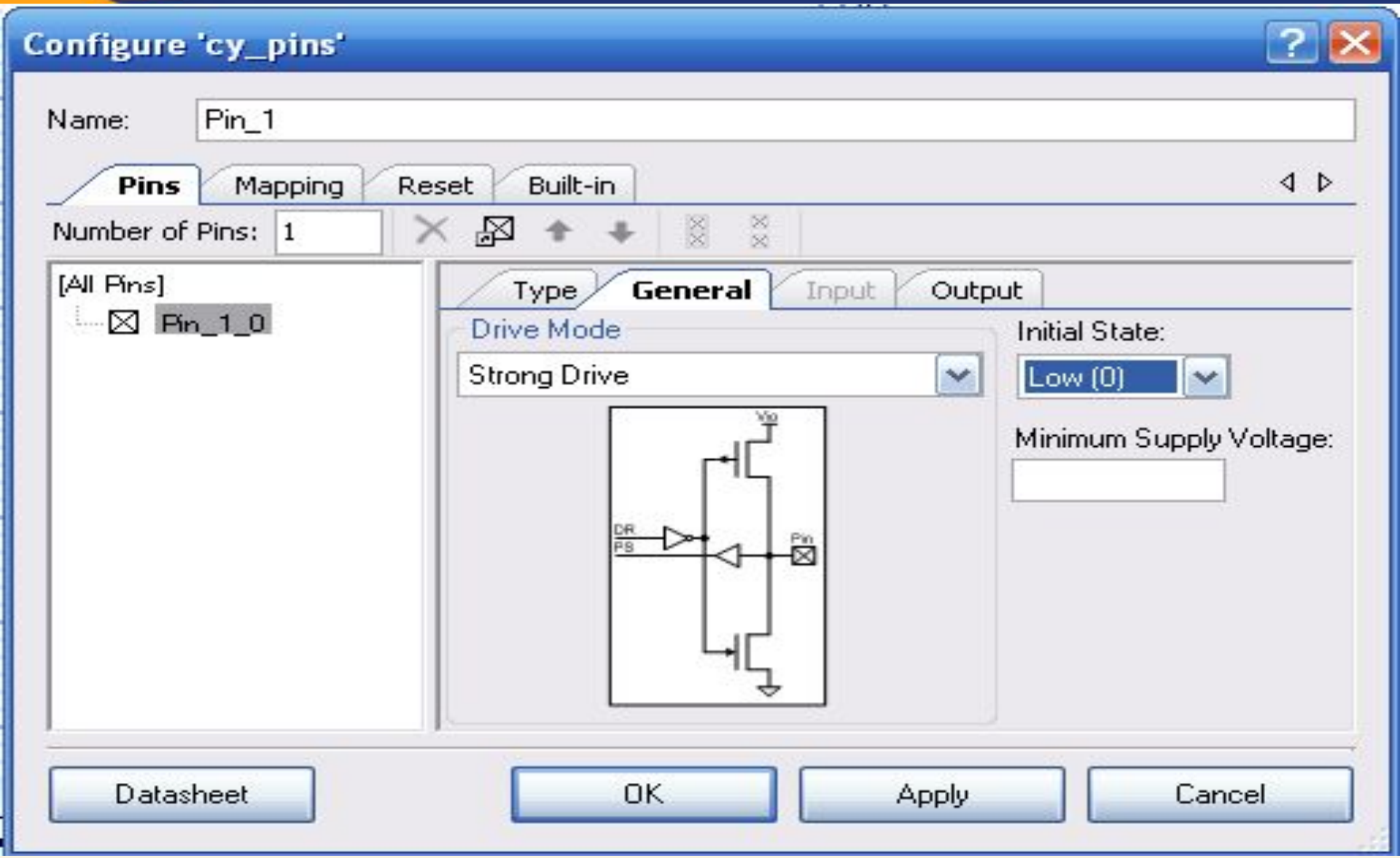
Lab_7 Fan


Configure 'cy_clock'
Name:
Clock
Basic
Advanced
Built-in
Clock type:
New
Existing
Source:
<Auto>
Specify:
Frequency:
20
kHz
Tolerance:
5%
+
5%
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

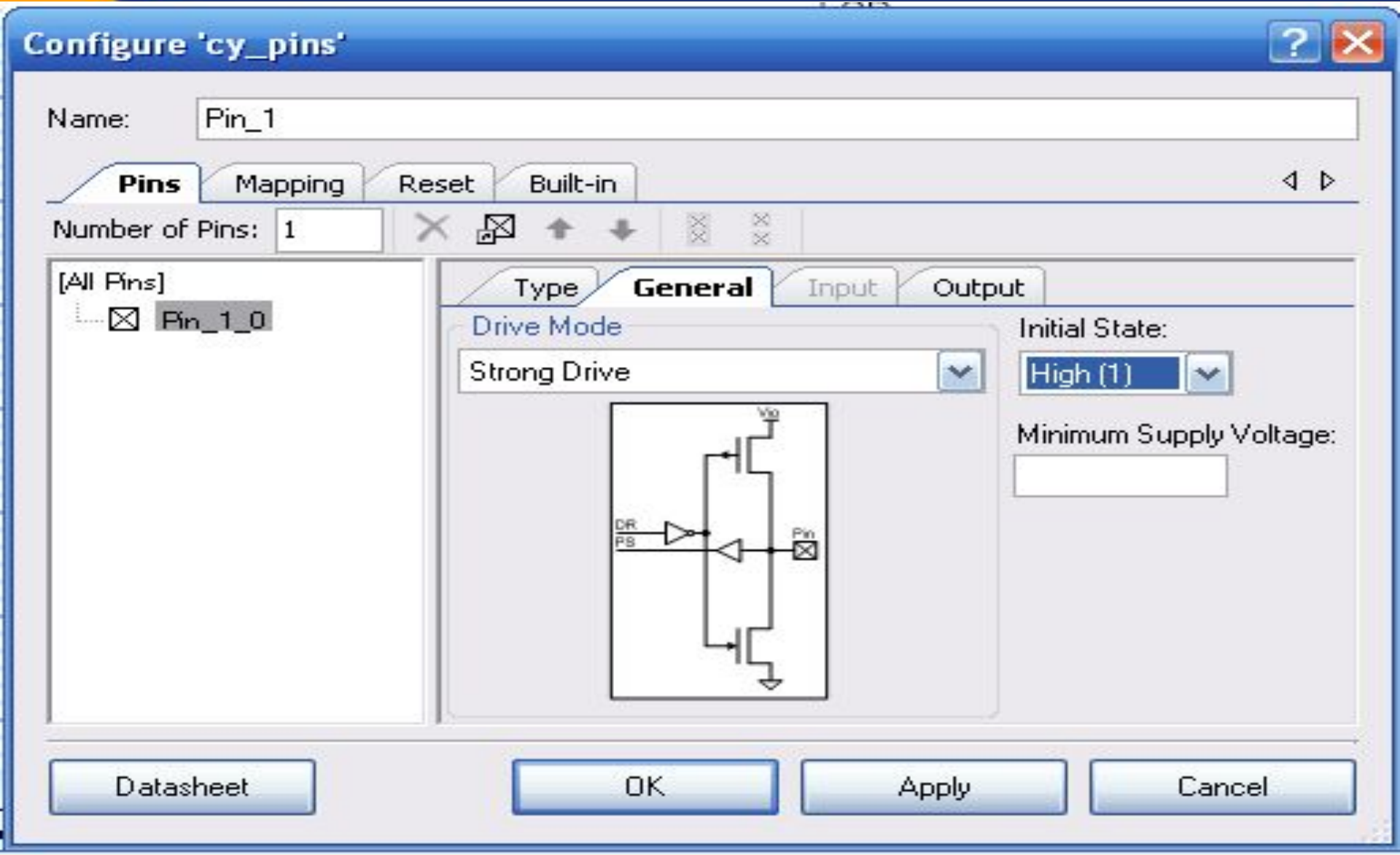
Lab_7 Fan


Configure 'cy_pins'
Name:
Pin_1
Pins
Mapping
Reset
Built-in
Number of Pins:
1
[All Pins]
Pin_1_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

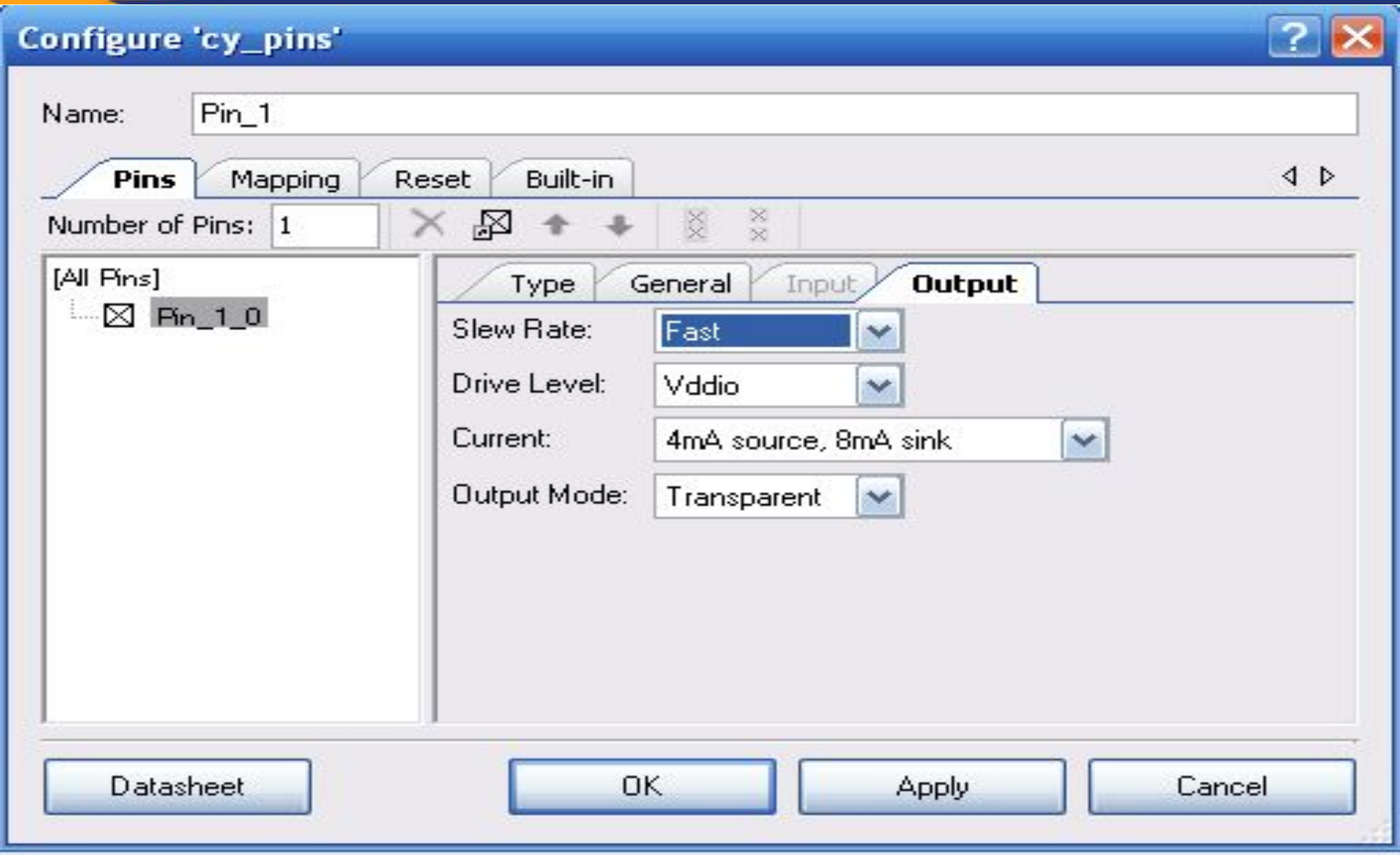
Lab_7 Fan


Configure 'cy_pins'
Name:
Pin_1
Pins
Mapping
Reset
Built-in
Number of Pins:
1
[All Pins]
Pin_1_0
Type
General
Input
Output
Drive Mode
Strong Drive
Initial State:
Low (0)
Minimum Supply Voltage:
DR
PS
Pin
Datasheet
OK
Apply
Cancel

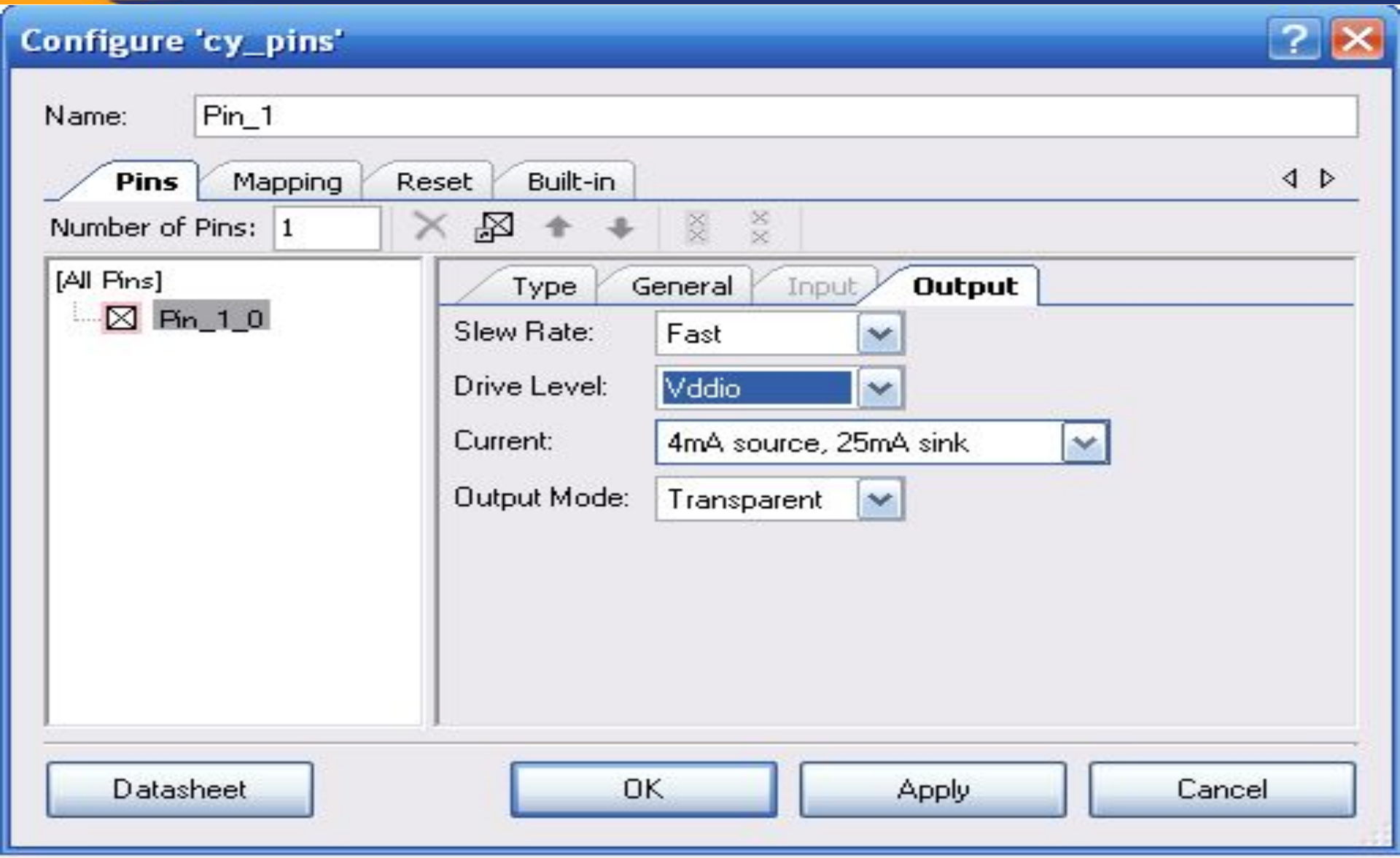
Lab_7 Fan


Configure 'cy_pins'
Name:
Pin_1
Pins
Mapping
Reset
Built-in
Number of Pins:
1
[All Pins]
Pin_1_0
Type
General
Input
Output
Drive Mode
Strong Drive
Initial State:
High (1)
Minimum Supply Voltage:
Vdd
DR
PS
Pin
Datasheet
OK
Apply
Cancel

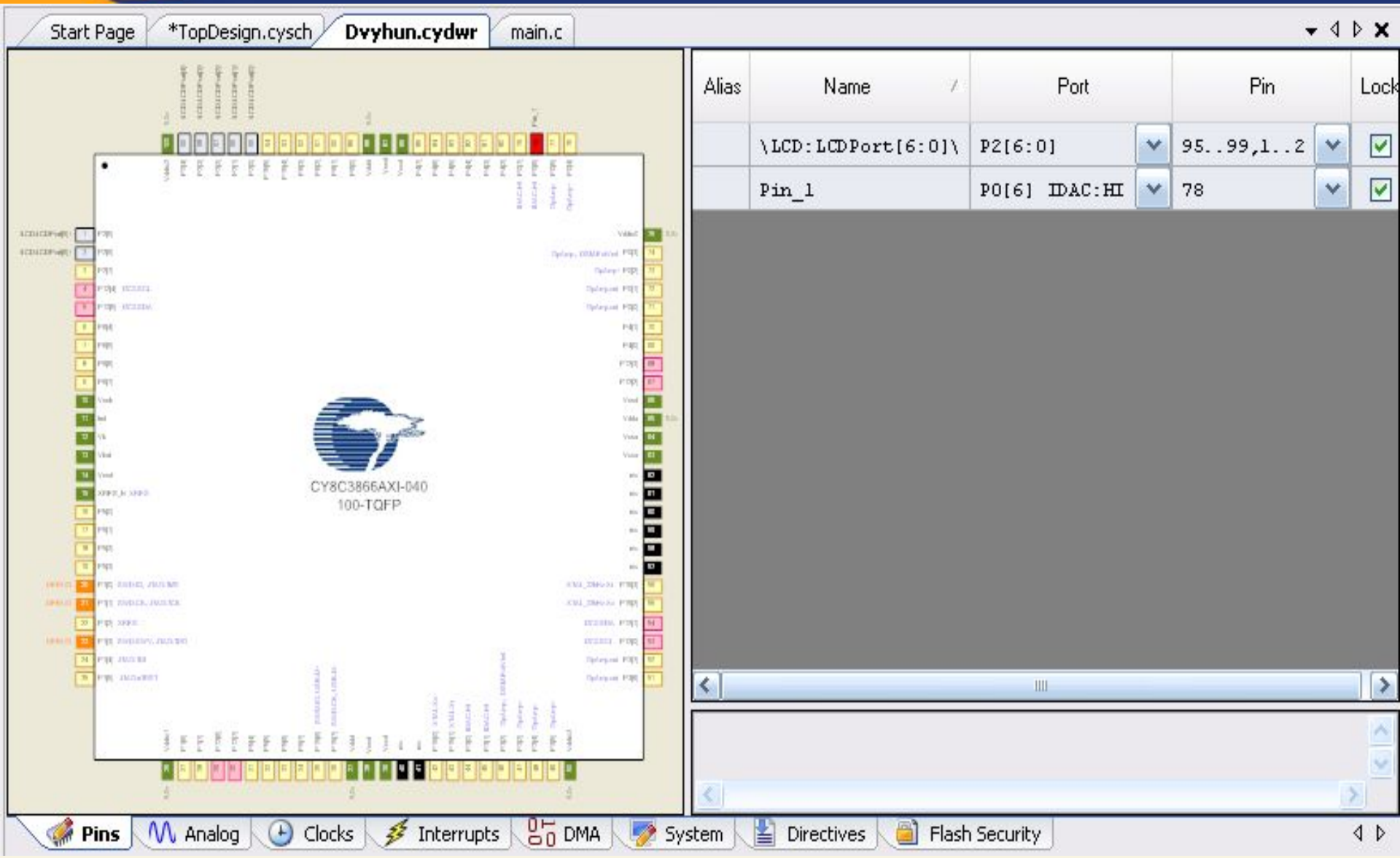
Lab_7 Fan


Configure 'cy_pins'
Name:
Pin_1
Pins
Mapping
Reset
Built-in
Number of Pins:
1
[All Pins]
Pin_1_0
Type
General
Input
Output
Slew Rate:
Fast
Drive Level:
Vddio
Current:
4mA source, 8mA sink
Output Mode:
Transparent
Datasheet
OK
Apply
Cancel

Lab_7 Fan


Configure 'cy_pins'
Name:
Pin_1
Pins
Mapping
Reset
Built-in
Number of Pins:
1
[All Pins]
Pin_1_0
Type
General
Input
Output
Slew Rate:
Fast
Drive Level:
Vddio
Current:
4mA source, 25mA sink
Output Mode:
Transparent
Datasheet
OK
Apply
Cancel

Lab_7 Fan


Start Page
*TopDesign.cysch
Dvyhun.cydwr
main.c
CY8C3866AXI-040
100-TQFP
Alias
Name
Port
Pin
Lock
\LCD:LCDPort[6:0]\
P2[6:0]
95..99,1..2
Pin_1
P0[6] IDAC:HI
78
Pins
Analog
Clocks
Interrupts
DMA
System
Directives
Flash Security

Lab_7 Fan

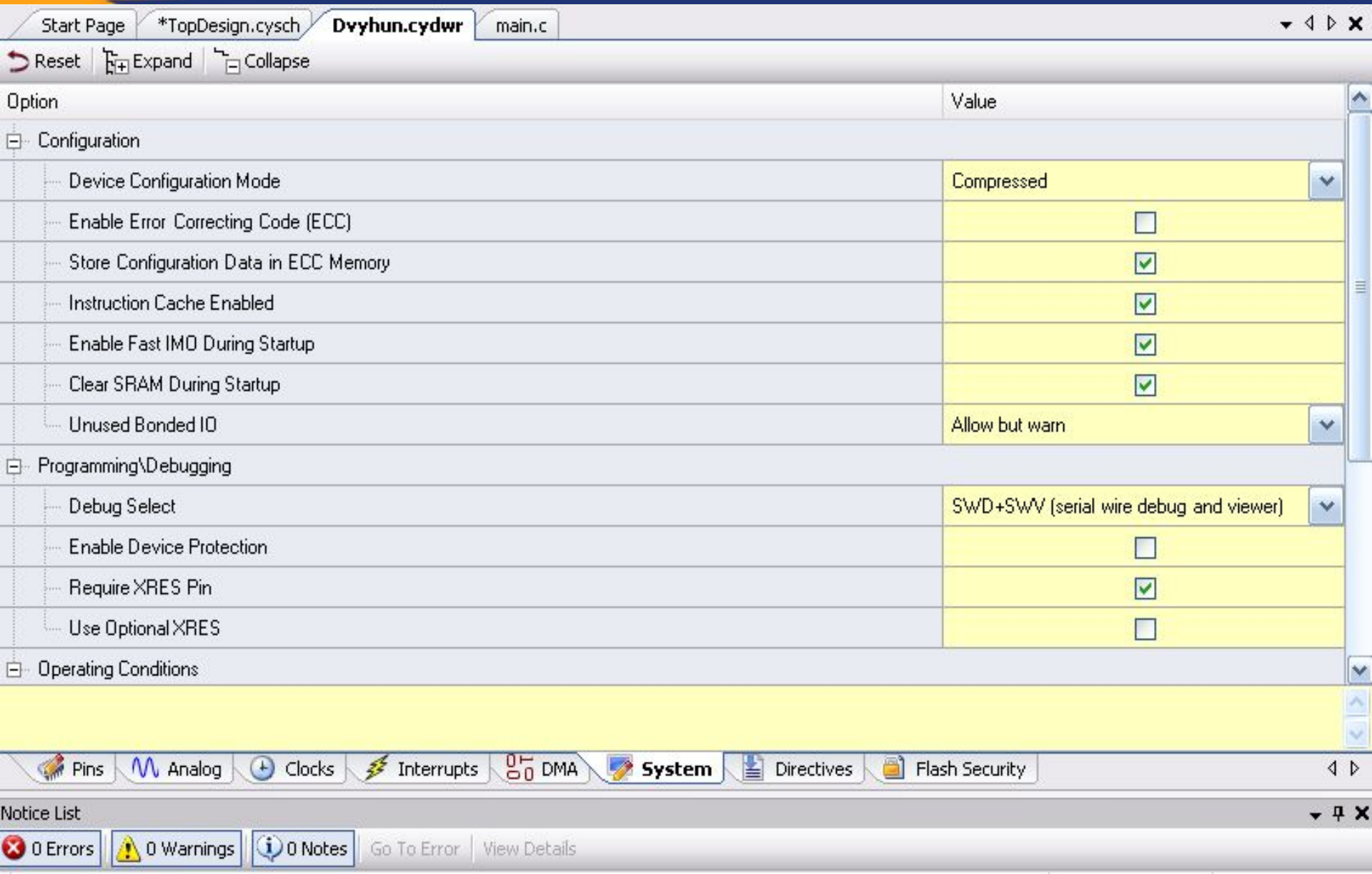
Start Page | *TopDesign.cysch | **Dvyhun.cydwr** | main.c

Add Design-Wide Clock... | Delete Design-Wide Clock | Edit Clock...

Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	Source Clock
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1	☐	IMOx2
System	Digital Signal	DIGITAL	? MHz	? MHz	±0	-	0	☐	
System	XTAL 32kHz	DIGITAL	32.768 kHz	? MHz	±0	-	0	☐	
System	XTAL	DIGITAL	25.000 MHz	? MHz	±0	-	0	☐	
System	ILO	DIGITAL	? MHz	1.000 kHz	-50, +100	-	0	☑	
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±1	-	0	☑	
System	BUS_CLK (CPU)	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	PLL_OUT
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±1	-	0	☑	IMO
Local	Clock	DIGITAL	20.000 kHz	20.000 kHz	±1	±5	150	☑	Auto: IMO

Pins | Analog | **Clocks** | Interrupts | DMA | System | Directives | Flash Security

Lab_7 Fan


Start Page
*TopDesign.cysch
Dvyhun.cydwr
main.c
Reset
Expand
Collapse
Option
Value
Configuration
Device Configuration Mode
Compressed
Enable Error Correcting Code (ECC)
Store Configuration Data in ECC Memory
Instruction Cache Enabled
Enable Fast IMO During Startup
Clear SRAM During Startup
Unused Bonded IO
Allow but warn
Programming\Debugging
Debug Select
SWD+SWV (serial wire debug and viewer)
Enable Device Protection
Require XRES Pin
Use Optional XRES
Operating Conditions
Pins
Analog
Clocks
Interrupts
DMA
System
Directives
Flash Security
Notice List
0 Errors
0 Warnings
0 Notes
Go To Error
View Details

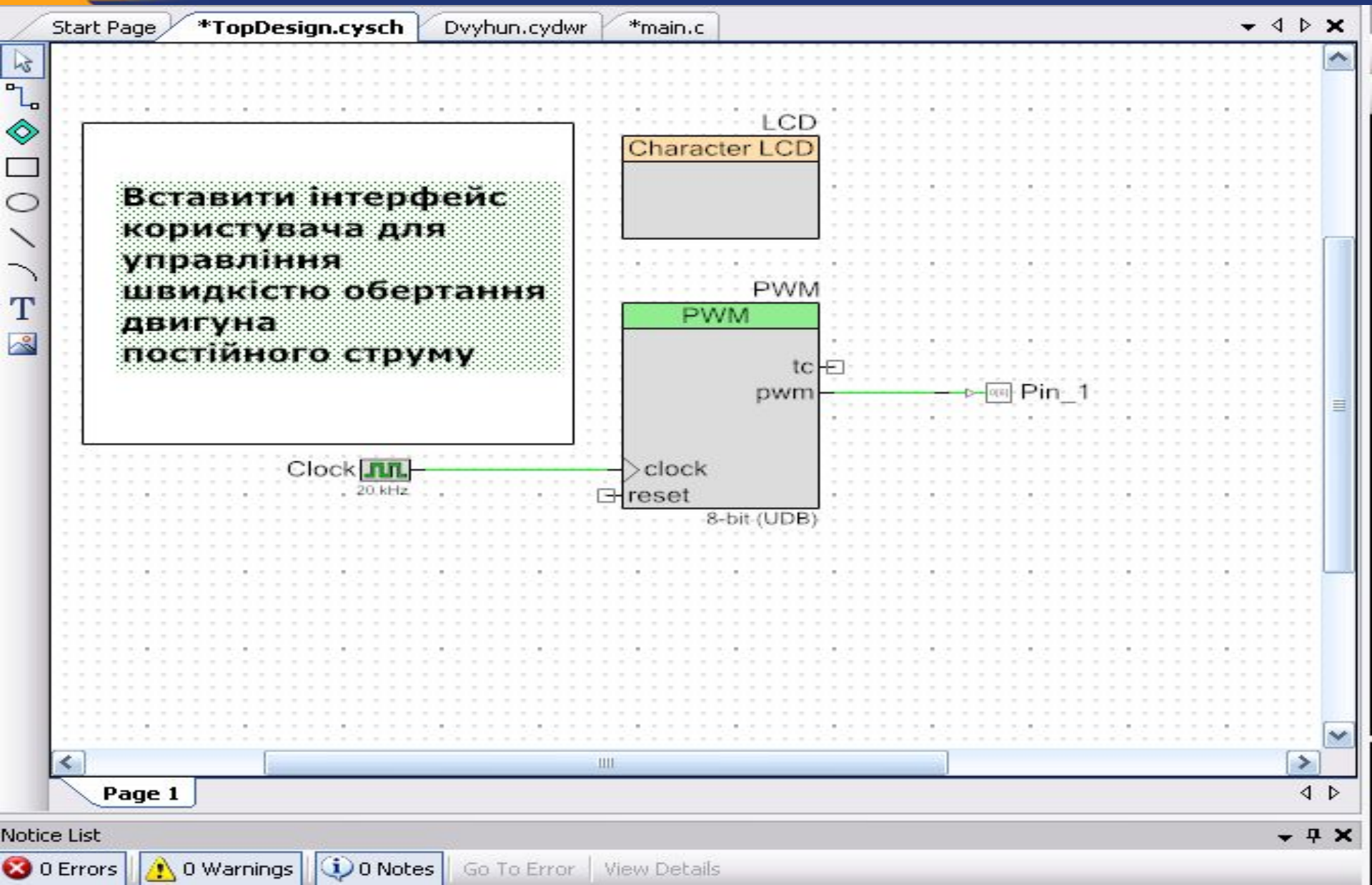
Start Page | *TopDesign.cysch | Dvyhun.cydwr | *main.c

```
/* ========================================
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 * ========================================
*/
#include <device.h>

void main()
{
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
/* Enable the global interrupts */
    CYGlobalIntEnable;
    LCD_Start();
    LCD_Position(0,9);
LCD_PrintString("Dvyhun");
    /* CyGlobalIntEnable; */ /* Uncomment this line to enable global interrupts. */
   PWM_Start();
    for(;;)
    {
        /* Place your application code here. */
    }
}

/* [] END OF FILE */
```

Notice List

0 Errors | 0 Warnings | 0 Notes | Go To Error | View Details

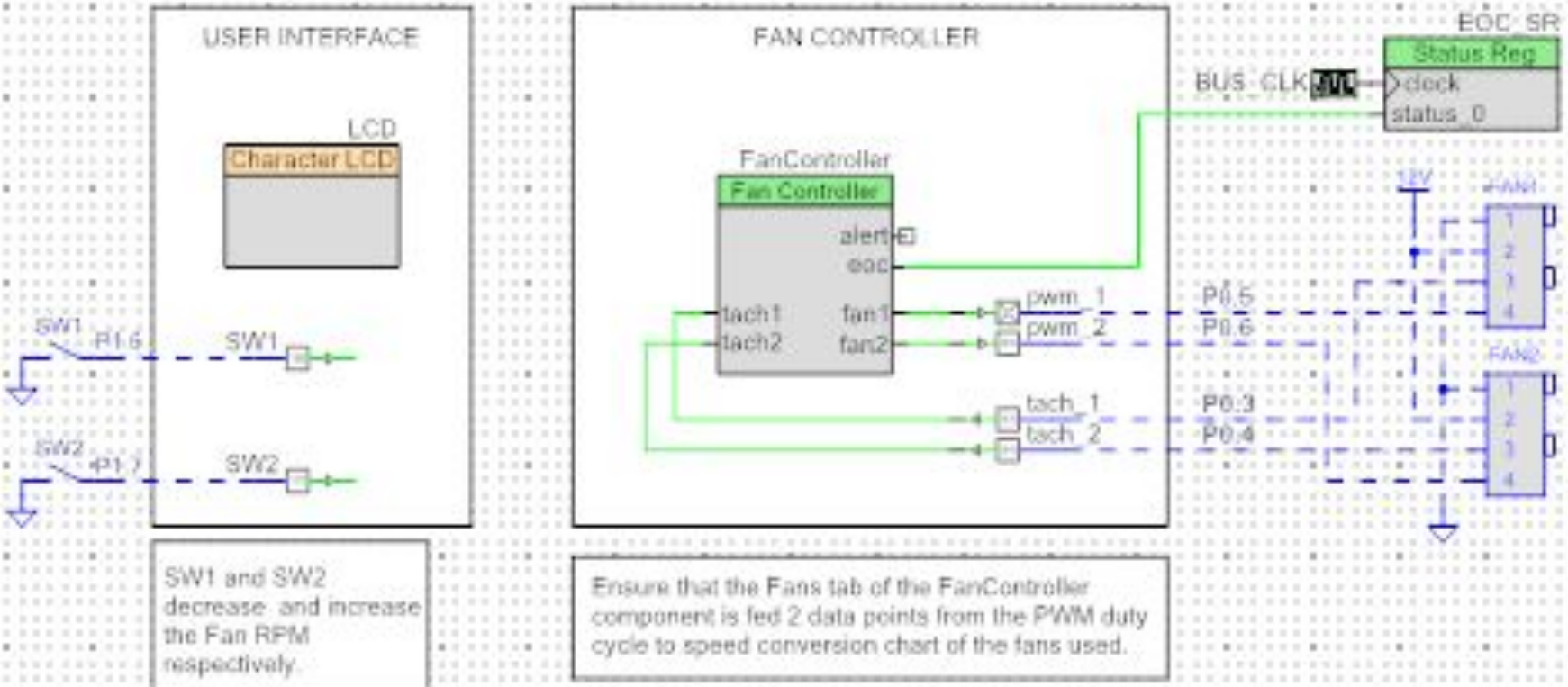
Lab_7 Fan


Start Page
*TopDesign.cysch
Dvyhun.cydwr
*main.c
Вставити інтерфейс користувача для управління швидкістю обертання двигуна постійного струму
LCD
Character LCD
PWM
PWM
tc
pwm
Pin_1
Clock
20 kHz
clock
reset
8-bit (UDB)
Page 1
Notice List
0 Errors
0 Warnings
0 Notes
Go To Error
View Details

CHAPTER #HI

FanController_Example Project - Firmware Fan Control
Fan Controller is configured into firmware (CPU) fan control mode
Speed control algorithm is implemented in firmware
Firmware synchornizes to hardware using end-of-cycle (eoc) pulse
2 Fans are supported (individual PWMs - no banks)

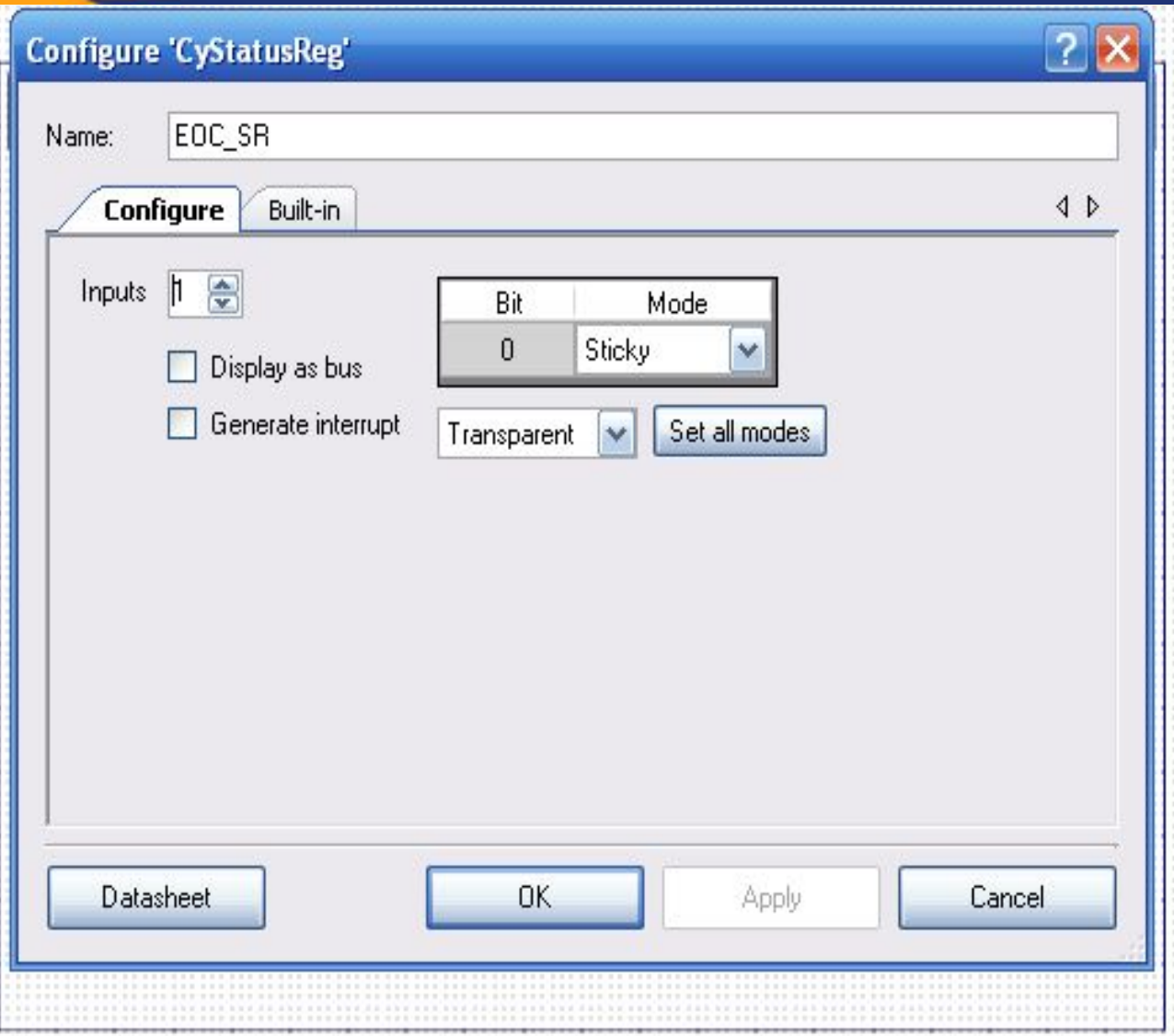
Sticky status register indicates when tachometer finishes reading speeds of all fans

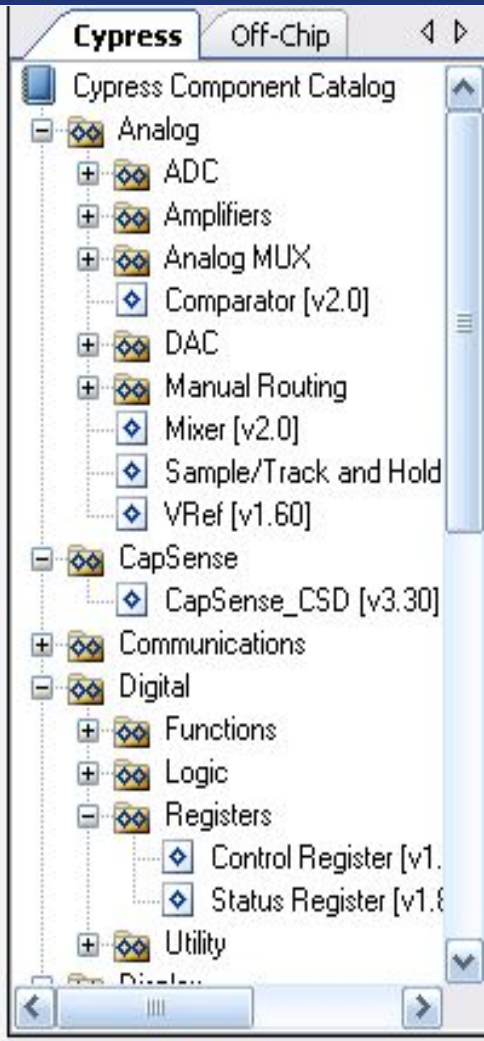


SW1 and SW2 decrease and increase the Fan RPM respectively.

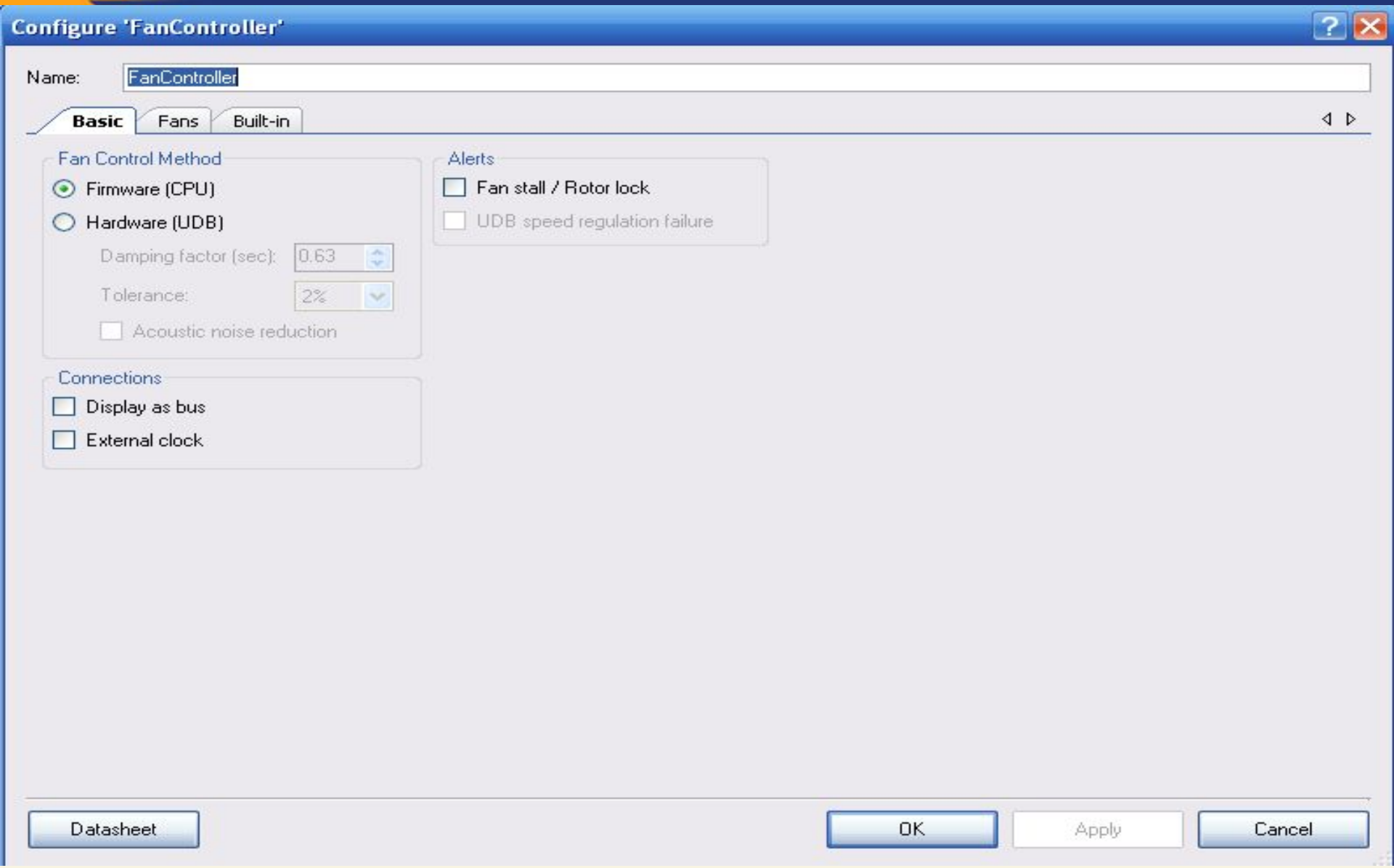
Ensure that the Fans tab of the FanController component is fed 2 data points from the PWM duty cycle to speed conversion chart of the fans used.

Lab_7

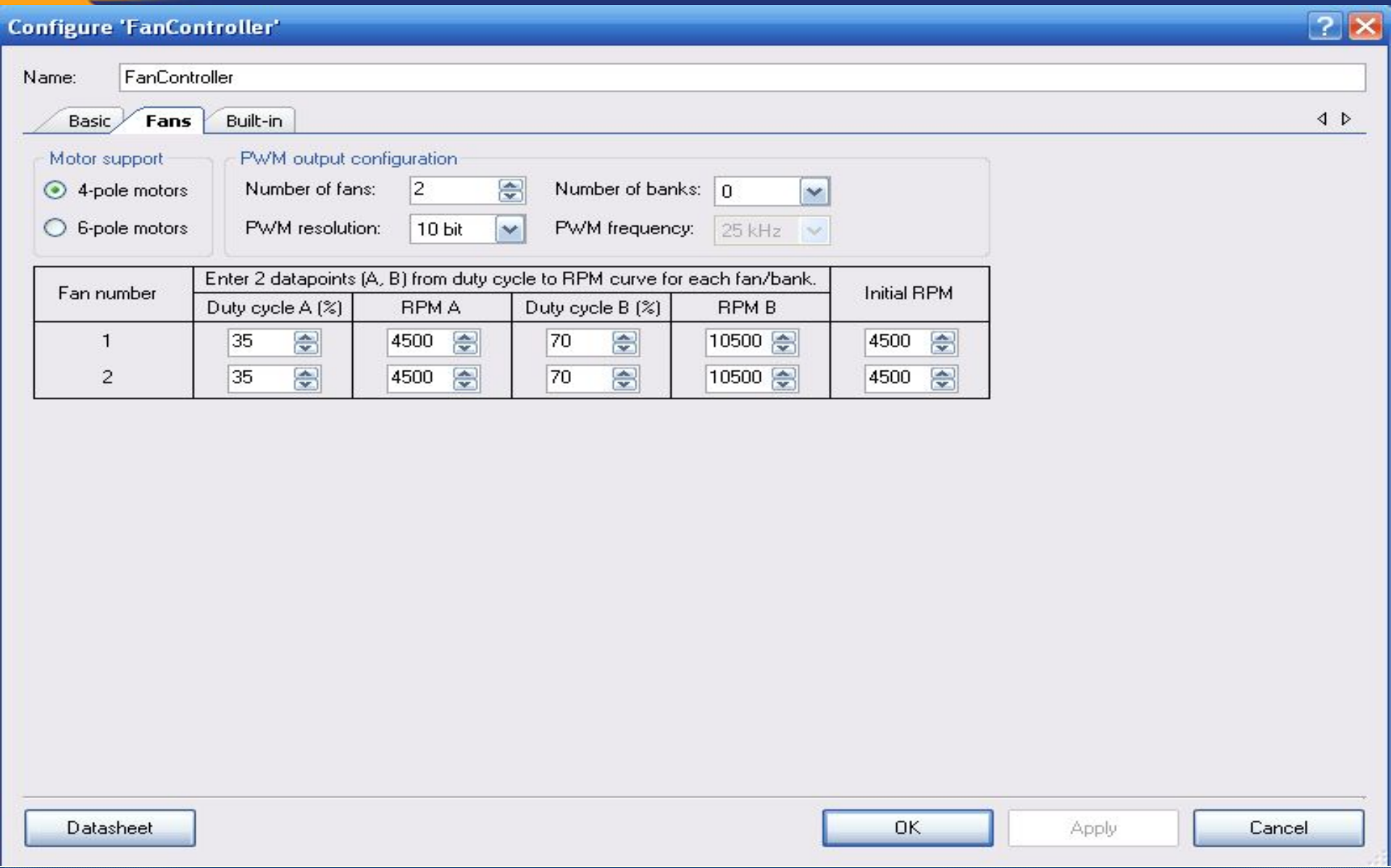

Configure 'CyStatusReg'
Name:
EOC_SR
Configure
Built-in
Inputs
1
Bit
Mode
0
Sticky
Display as bus
Generate interrupt
Transparent
Set all modes
Datasheet
OK
Apply
Cancel


Cypress
Off-Chip
Cypress Component Catalog
Analog
ADC
Amplifiers
Analog MUX
Comparator [v2.0]
DAC
Manual Routing
Mixer [v2.0]
Sample/Track and Hold
VRef [v1.60]
CapSense
CapSense_CSD [v3.30]
Communications
Digital
Functions
Logic
Registers
Control Register [v1.
Status Register [v1.
Utility
Component Preview

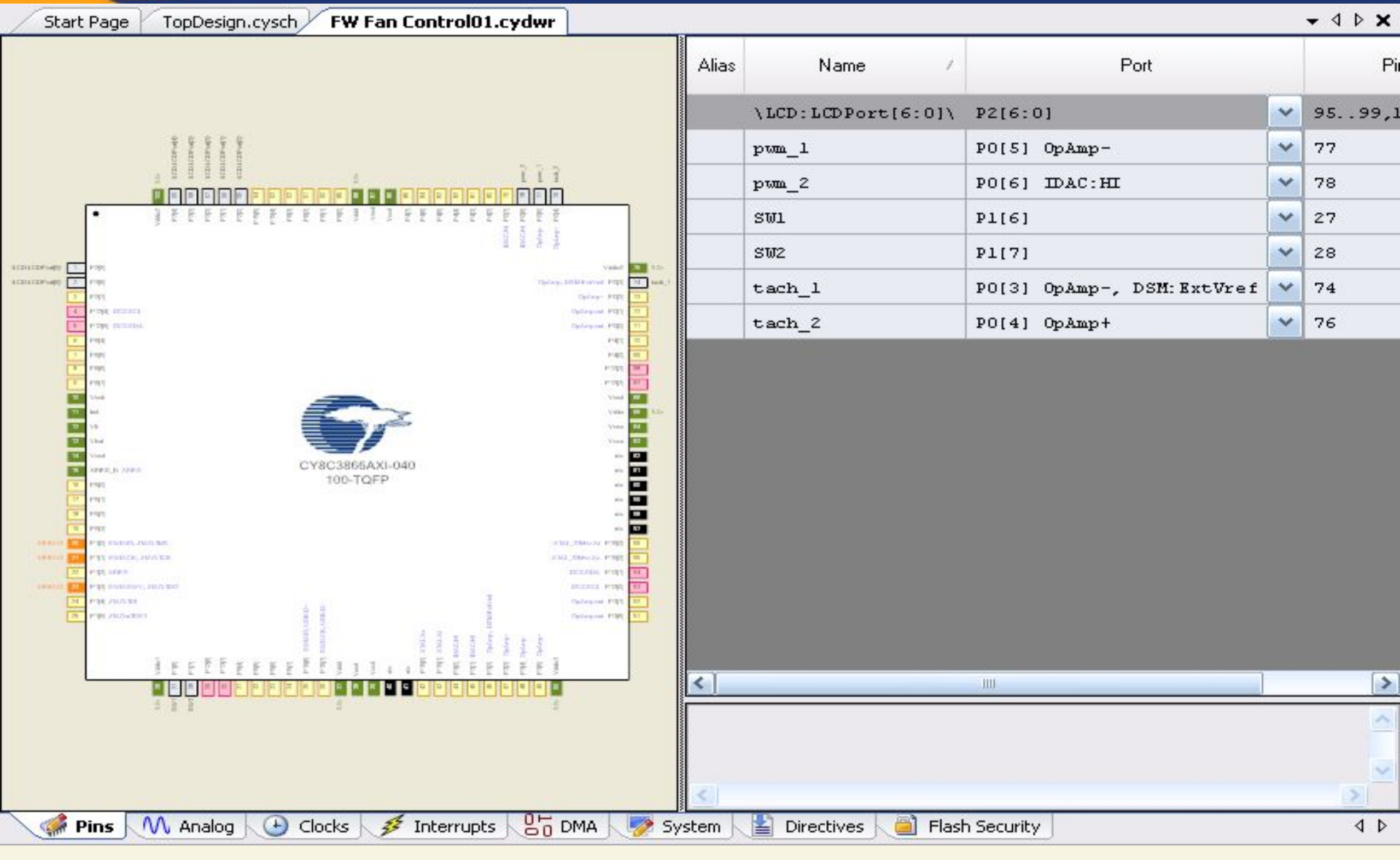
Configuration Fan Controller



Configuration Fan Controller



Lab_7 .cywr


Start Page
TopDesign.cysch
FW Fan Control01.cydwr
CY8C3866AXI-040
100-TQFP
Alias
Name
Port
\LCD:LCDPort[6:0]\
P2[6:0]
95..99,1
pwm_1
P0[5] OpAmp-
77
pwm_2
P0[6] IDAC:HI
78
SW1
P1[6]
27
SW2
P1[7]
28
tach_1
P0[3] OpAmp-, DSM:ExtVref
74
tach_2
P0[4] OpAmp+
76
Pins
Analog
Clocks
Interrupts
DMA
System
Directives
Flash Security

Lab_7

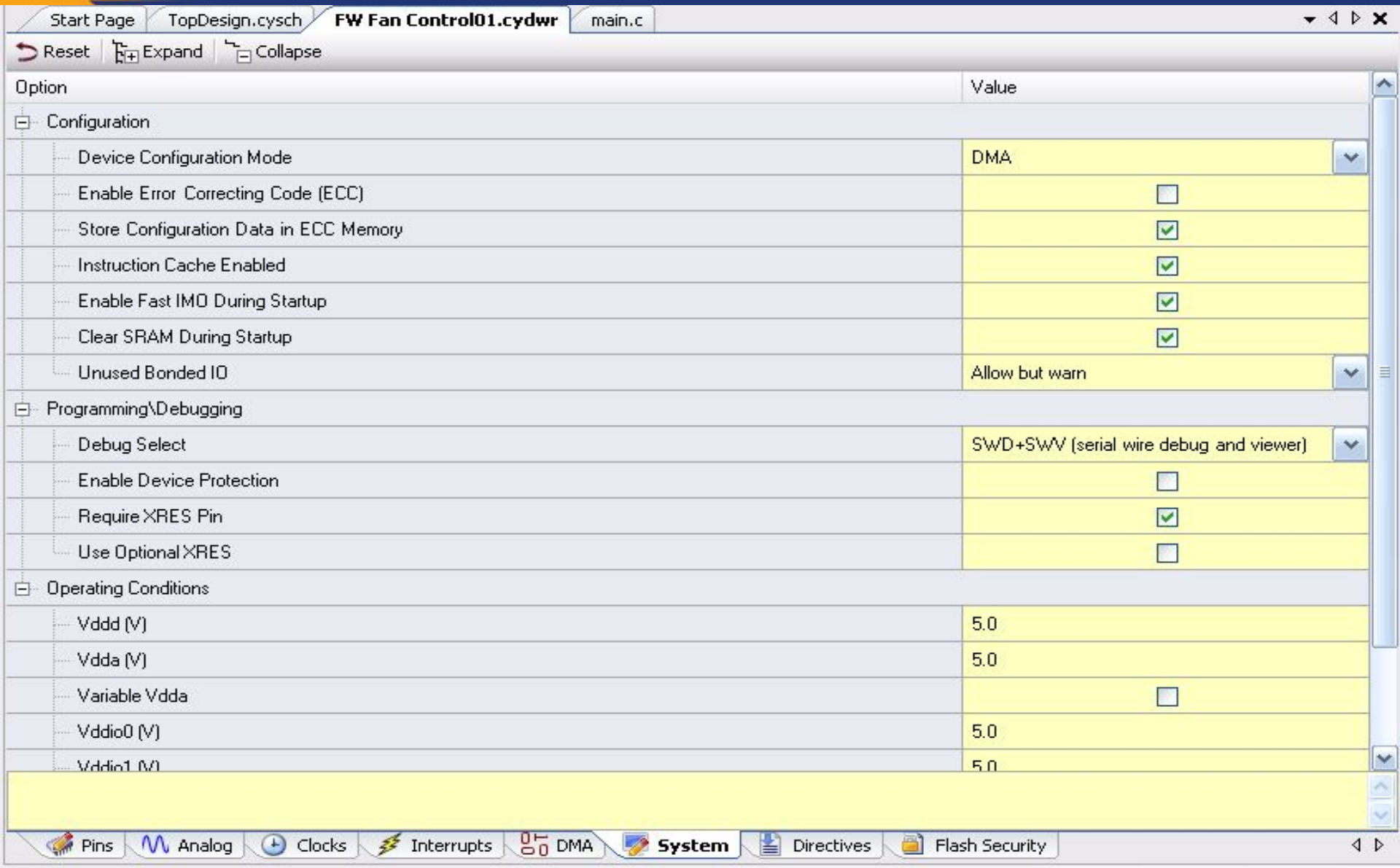
Start Page | TopDesign.cysch | **FW Fan Control01.cydwr** | main.c

Add Design-Wide Clock... | Delete Design-Wide Clock | Edit Clock...

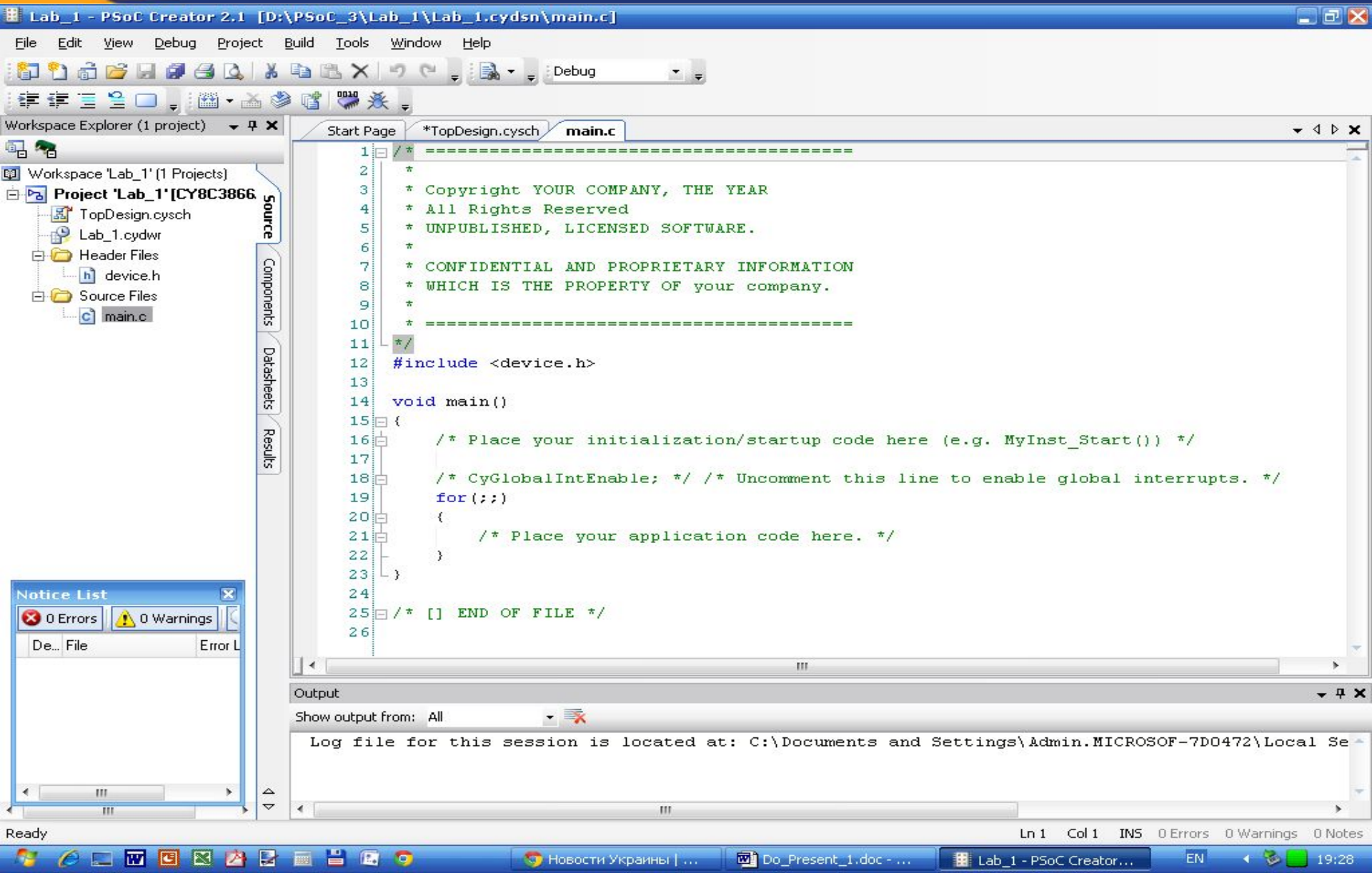
Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	Source Clock
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1	☐	IMOx2
System	Digital Signal	DIGITAL	? MHz	? MHz	±0	-	0	☐	
System	XTAL 32kHz	DIGITAL	32.768 kHz	? MHz	±0	-	0	☐	
System	XTAL	DIGITAL	25.000 MHz	? MHz	±0	-	0	☐	
System	ILO	DIGITAL	? MHz	1.000 kHz	-50, +100	-	0	☑	
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±1	-	0	☑	
System	BUS_CLK (CPU)	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	PLL_OUT
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±1	-	0	☑	IMO
Local	FanController_TACH_CLK_500K	DIGITAL	500.000 kHz	500.000 kHz	±1	±5	6	☑	Auto: IMO
Local	FanController_PWM_CLOCK_25k_10b	DIGITAL	24.000 MHz	24.000 MHz	±1	-	1	☑	MASTER_CLK
Local	FanController_BUS_CLK	DIGITAL	? MHz	24.000 MHz	±1	-	0	☑	BUS_CLK
Local	Clock_1	DIGITAL	? MHz	24.000 MHz	±1	-	0	☑	BUS_CLK

Pins | Analog | **Clocks** | Interrupts | DMA | System | Directives | Flash Security

Lab_7


Start Page
TopDesign.cysch
FW Fan Control01.cydwr
main.c
Reset
Expand
Collapse
Option
Value
Configuration
Device Configuration Mode
DMA
Enable Error Correcting Code (ECC)
Store Configuration Data in ECC Memory
Instruction Cache Enabled
Enable Fast IMO During Startup
Clear SRAM During Startup
Unused Bonded IO
Allow but warn
Programming\Debugging
Debug Select
SWD+SWV (serial wire debug and viewer)
Enable Device Protection
Require XRES Pin
Use Optional XRES
Operating Conditions
Vddd (V)
5.0
Vdda (V)
5.0
Variable Vdda
Vddio0 (V)
5.0
Vddio1 (V)
5.0
Pins
Analog
Clocks
Interrupts
DMA
System
Directives
Flash Security

Main.c


Lab_1 - PSoC Creator 2.1 [D:\PSoC_3\Lab_1\Lab_1.cydsn\main.c]
File Edit View Debug Project Build Tools Window Help
Debug
Workspace Explorer (1 project)
Workspace 'Lab_1' (1 Projects)
Project 'Lab_1' [CY8C3866
TopDesign.cysch
Lab_1.cydwr
Header Files
device.h
Source Files
main.c
Source
Components
Datasheets
Results
Start Page
*TopDesign.cysch
main.c
1 /* ==
2 *
3 * Copyright YOUR COMPANY, THE YEAR
4 * All Rights Reserved
5 * UNPUBLISHED, LICENSED SOFTWARE.
6 *
7 * CONFIDENTIAL AND PROPRIETARY INFORMATION
8 * WHICH IS THE PROPERTY OF your company.
9 *
10 * ==
11 */
12 #include <device.h>
13
14 void main()
15 {
16 /* Place your initialization/startup code here (e.g. MyInst_Start()) */
17
18 /* CyGlobalIntEnable; */ /* Uncomment this line to enable global interrupts. */
19 for(;;)
20 {
21 /* Place your application code here. */
22 }
23 }
24
25 /* [] END OF FILE */
26
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
Ln 1 Col 1 INS 0 Errors 0 Warnings 0 Notes
Новости Украины | ...
Do_Present_1.doc - ...
Lab_1 - PSoC Creator...
EN
19:28

Lab_7

```
/*******************************************************************************
* Project Name:       Firmware Fan Control
* Version:            1.10
*
* Device Used:        PSoC 3 CY8C3866AXI-040 or PSoC 5 CY8C5588AXI-060ES1
* Software Used:      PSoC Creator v2.1
* Compiler Used:      Keil(C51), ARM GNU CC
* Related Hardware:   CY8CKIT-001 PSoC DVK
********************************************************************************
* Theory of Operation:
*
*  Fan Controller is configured into  firmware (CPU) fan control mode
*  Speed control algorithm is implemented in firmware
*  Firmware synchornizes to hardware using end-of-cycle (eoc) pulse
*  2 Fans are supported (individual PWMs - no banks)
*
*  1st line of LCD displays Desired Speed and Actual Speed and PWM Duty Cycle of Fan 1
*  2nd line of LCD displays Actual Speed and PWM Duty Cycle of Fan 2
*  SW1 decreases desired speed in RPM
*  SW2 increases desired speed in RPM
*
********************************************************************************
* Copyright 2012, Cypress Semiconductor Corporation. All rights reserved.
* This software is owned by Cypress Semiconductor Corporation and is protected
* by and subject to worldwide patent and copyright laws and treaties.
* Therefore, you may use this software only as provided in the license agreement
* accompanying the software package from which you obtained this software.
* CYPRESS AND ITS SUPPLIERS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
* WITH REGARD TO THIS SOFTWARE, INCLUDING, BUT NOT LIMITED TO, NONINFRINGEMENT,
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
*******************************************************************************/
```

Lab_7

```
31  ********************************************************************************/
32
33  #include <device.h>
34  #include <stdio.h>
35
36  /* PWM duty cycle controls - units are hundredths of a percent */
37  #define MIN_DUTY              (50u)
38  #define MAX_DUTY              (9950u)
39  #define DUTY_STEP_COARSE      (100u)
40  #define DUTY_STEP_FINE        (2u)
41
42  /* Speed controls - units are RPM */
43  #define MIN_RPM               (2500u)
44  #define MAX_RPM               (9500u)
45  #define INIT_RPM              (4500u)
46  #define RPM_STEP              (500u)
47  #define RPM_DELTA_LARGE       (500u)
48  #define RPM_TOLERANCE         (100u)
49
50  void main()
51  {
52      uint16  desiredSpeed = INIT_RPM;
53      uint16  dutyCycle[2];
54      uint8   fanNumber;
55      char    displayString[6];
56
57      FanController_Start();
58      FanController_SetDesiredSpeed(1u, desiredSpeed);
59      FanController_SetDesiredSpeed(2u, desiredSpeed);
60      dutyCycle[0] = FanController_GetDutyCycle(1u);
61      dutyCycle[1] = FanController_GetDutyCycle(2u);
```

Lab_7

```
61      dutyCycle[1] = FanController_GetDutyCycle(2u);
62      LCD_Start();
63      LCD_Position(0u, 0u);
64      LCD_PrintDecUint16(desiredSpeed);
65      LCD_Position(1u, 0u);
66      LCD_PrintString("F/W");
67      while(1u)
68      {
69          /* Synchronize firmware to end-of-cycle pulse from FanController */
70          if(EOC_SR_Read())
71          {
72              for(fanNumber = 1u; fanNumber <= 2u; fanNumber++)
73              {
74                  /* Display Fan Actual Speeds */
75                  LCD_Position(fanNumber-1,5u);
76                  LCD_PrintDecUint16(FanController_GetActualSpeed(fanNumber));
77                  LCD_PrintString("    ");
78
79                  /* Firmware Speed Regulation */
80                  LCD_Position(fanNumber-1,9u);
81
82                  /* Fan Below Desired Speed */
83                  if(FanController_GetActualSpeed(fanNumber) < desiredSpeed)
84                  {
85                      if((desiredSpeed - FanController_GetActualSpeed(fanNumber)) > RPM_DELTA_Li
86                      {
87                          dutyCycle[fanNumber-1] += DUTY_STEP_COARSE;
88                      }
89                      else
90                      {
91                          dutyCycle[fanNumber-1] += DUTY_STEP_FINE;
92                      }
```

Lab_7

```
89                  else
90                  {
91                      dutyCycle[fanNumber-1] += DUTY_STEP_FINE;
92                  }
93                  if(dutyCycle[fanNumber-1] > MAX_DUTY)
94                  {
95                      dutyCycle[fanNumber-1] = MAX_DUTY;
96                  }
97              }
98              /* Fan Above Desired Speed */
99              else if(FanController_GetActualSpeed(fanNumber) > desiredSpeed)
100             {
101                 if((FanController_GetActualSpeed(fanNumber) - desiredSpeed) > RPM_DELTA_LA
102                 {
103                     if(dutyCycle[fanNumber-1] > (MIN_DUTY+DUTY_STEP_COARSE))
104                     {
105                         dutyCycle[fanNumber-1] -= DUTY_STEP_COARSE;
106                     }
107                 }
108                 else if((FanController_GetActualSpeed(fanNumber) - desiredSpeed) > RPM_TOL
109                 {
110                     if(dutyCycle[fanNumber-1] > MIN_DUTY)
111                     {
112                         dutyCycle[fanNumber-1] -= DUTY_STEP_FINE;
113                     }
114                 }
115             }
116             FanController_SetDutyCycle(fanNumber, dutyCycle[fanNumber-1]);
117
118             /* Display Current Duty Cycle Settings (in 100ths of a percent) */
119             LCD_Position(fanNumber-1,10u);
```

Lab_7

```
120             sprintf(displayString, "%5.2f", (((float)dutyCycle[fanNumber-1])/100));
121             LCD_PrintString(displayString);
122             LCD_PrintString("%     ");
123         }
124         CyDelay(250u);
125
126         /* Check for Button Press to Change Speed */
127         if((!SW1_Read()) || (!SW2_Read()))
128         {
129             /* Decrease Speed */
130             if(!SW1_Read())
131             {
132                 if(desiredSpeed > MIN_RPM)
133                 {
134                     desiredSpeed -= RPM_STEP;
135                 }
136             }
137
138             /* Increase Speed */
139             else
140             {
141                 desiredSpeed += RPM_STEP;
142                 if(desiredSpeed > MAX_RPM)
143                 {
144                     desiredSpeed = MAX_RPM;
145                 }
146             }
147
148             /* Display Updated Desired Speed */
149             LCD_Position(0u, 0u);
150             LCD_PrintDecUint16(desiredSpeed);
```

Lab_7

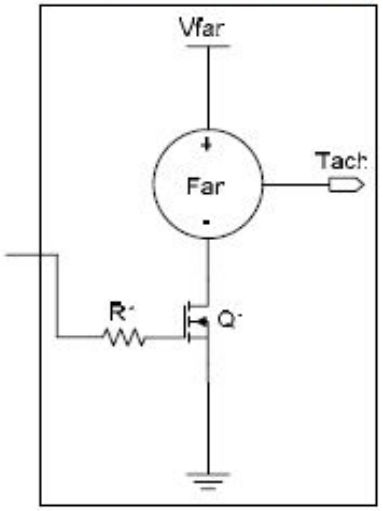
```
135                     }
136                 }
137
138                 /* Increase Speed */
139                 else
140                 {
141                     desiredSpeed += RPM_STEP;
142                     if(desiredSpeed > MAX_RPM)
143                     {
144                         desiredSpeed = MAX_RPM;
145                     }
146                 }
147
148                 /* Display Updated Desired Speed */
149                 LCD_Position(0u, 0u);
150                 LCD_PrintDecUint16(desiredSpeed);
151                 FanController_SetDesiredSpeed(1u, desiredSpeed);
152                 FanController_SetDesiredSpeed(2u, desiredSpeed);
153                 dutyCycle[0] = FanController_GetDutyCycle(1u);
154                 dutyCycle[1] = FanController_GetDutyCycle(2u);
155
156                 /* Switch Debounce */
157                 CyDelay(250u);
158             }
159         }
160     }
161 }
162
163
164 /* [] END OF FILE */
165
```

Fan Module

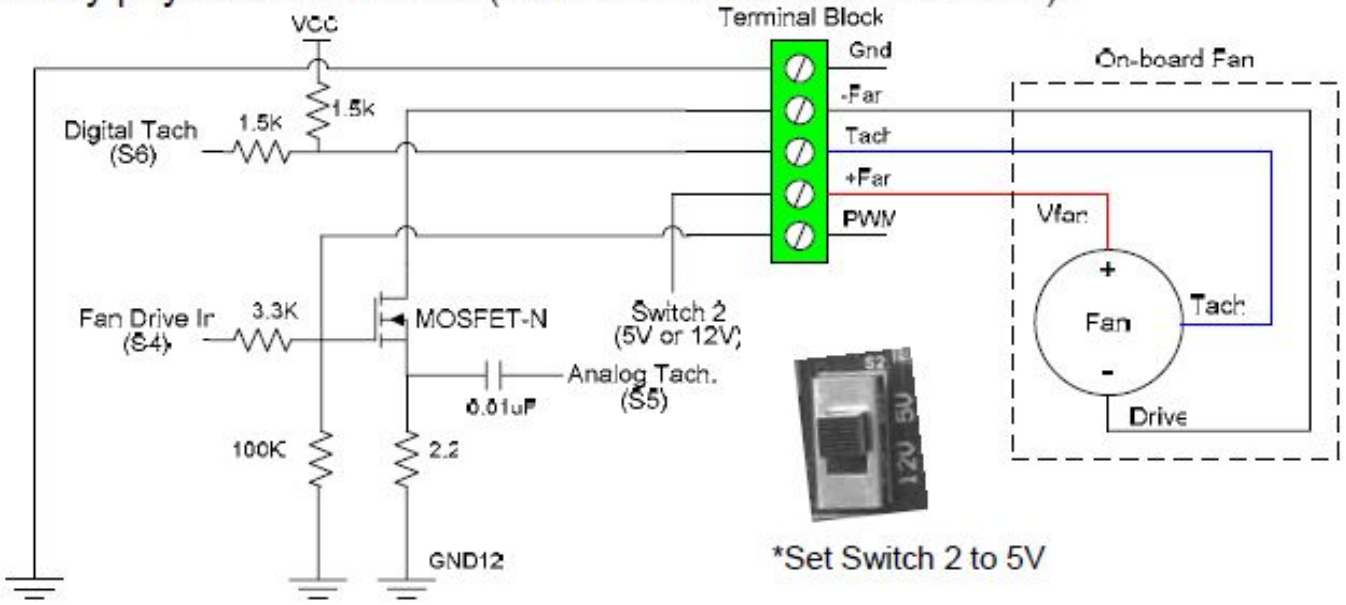
CY3210-ExpressDK and CY3210-FanMod

Using the on-board 3-wire, speed controllable, 5V external FET driven fan

1) Select appropriate PSoC Express output device:
   Fan >> Brushless DC >> Speed Control >> External Drive FET >> 5V 2-/3-Wire.



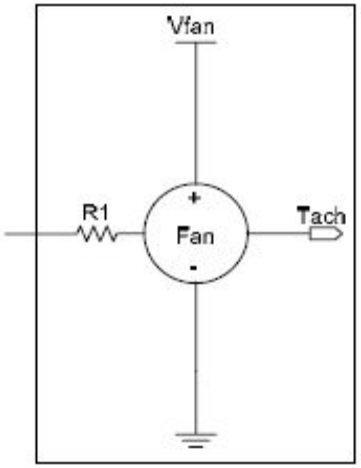
2) Verify physical connection (on-board fan to terminal block).



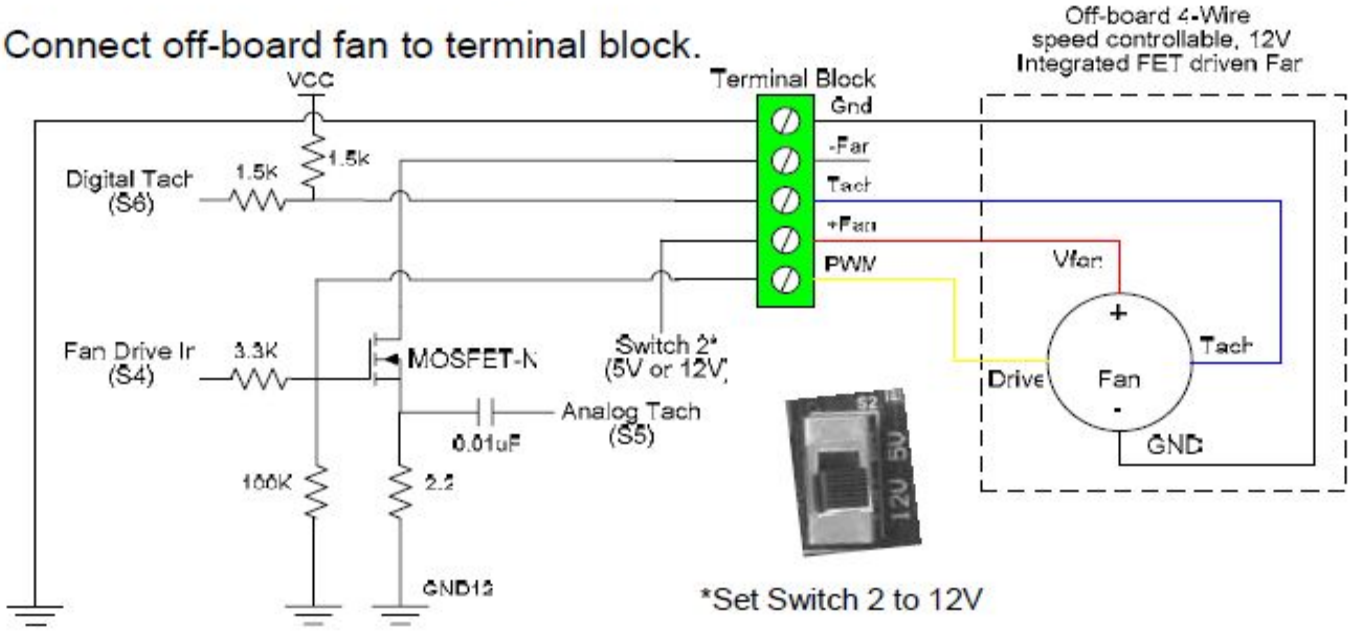
Lab_7

Attaching an off-board 4-wire, speed controllable, 12V integrated FET driven fan

1) Select appropriate PSoC Express output device::
Fan >> Brushless DC >> Speed Control >> Integrated Drive >> 12V 4-Wire.



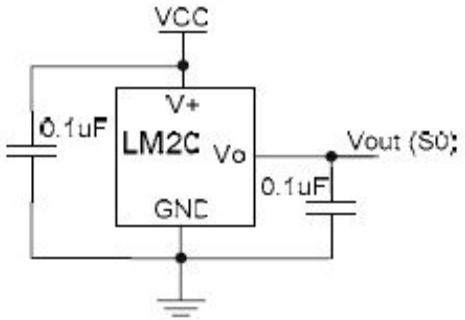
2) Connect off-board fan to terminal block.



Using the LM20

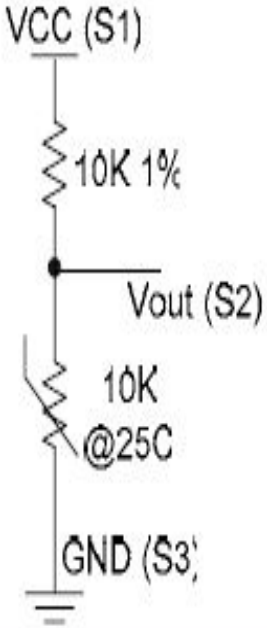
1) Select appropriate PSoC Express input device:
Temperature >> Integrated Circuits >> LM20 -55 to +130C.

2) Set Switch 1 to LM20. (LM20 Vout available on S0).



Lab_7

Using the Thermistor



1) Select appropriate PSoC Express input device:
   Temperature >> Thermistor Temp Sensor.

2) Set Switch 1 to LM20
   - Vcc available on S1
   - VTherm available on S2
   - GND available on S3

Diode Sensor Topologies

The Diode Sensor will be supported in future releases of PSoC Express.

Lab_7


PSoC Express Fan Board 1
Cypress Microsystems
S0:LM20 Vout
S1:Therm Rref
S2:Therm Vout
S3:Therm,Rtherm
S0:D1 Anode
S1:D1 Cathode
S2:D1 Cathode, Hi I
S3:D1 Cathode, Lo I
S0:D2 Anode, Lo I
S1:D2 Anode, Hi I
S2:D2 Anode
S3:D2 Cathode
S4:Fan Drive In
S5:Analog Tach
S6:Digital Tach

Fan Controller PSoC@3

Рекомендована література :

***Основна*:**

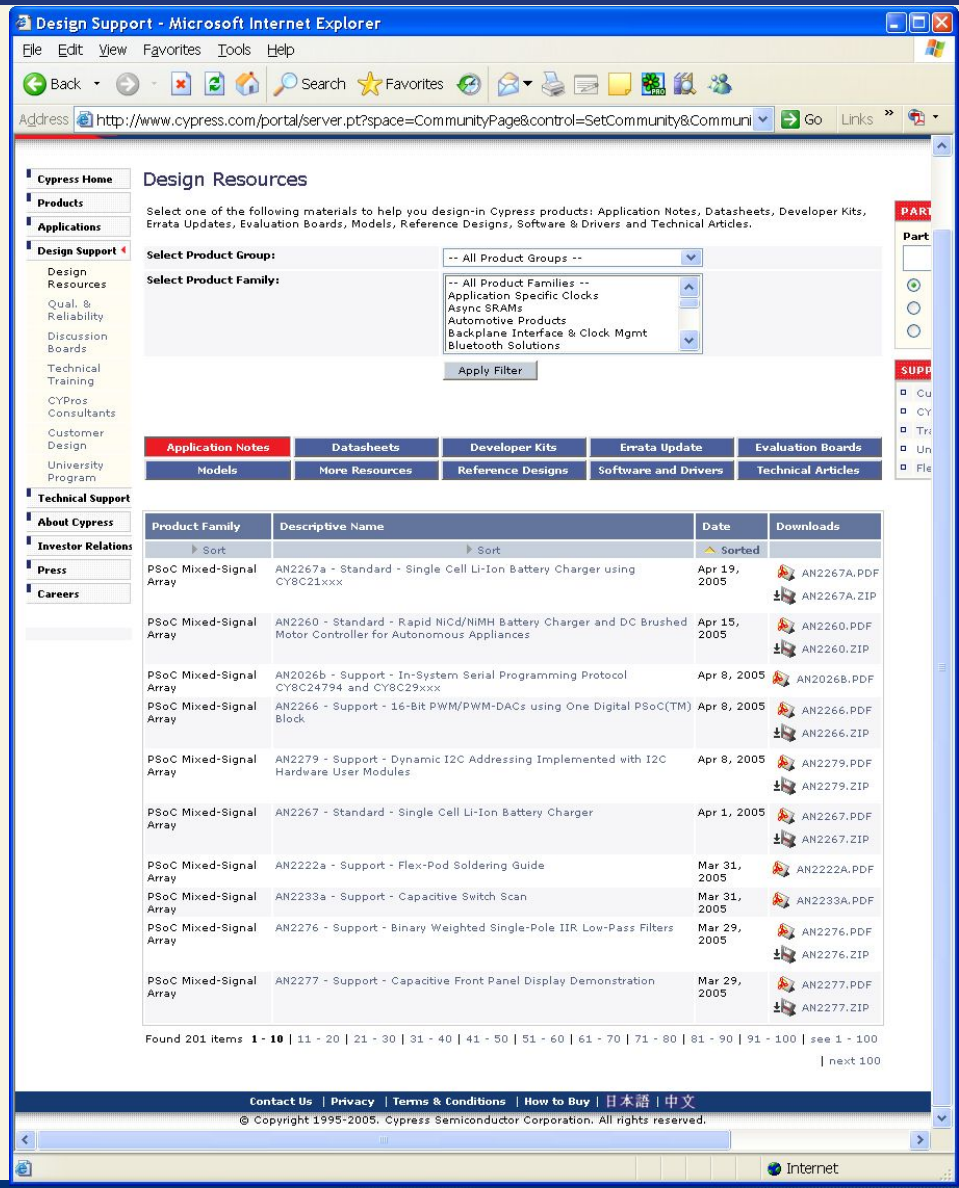
1. AN66627 PSoC® 3 and PSoC 5LP Intelligent Fan Controller . 2011 . - 22 с.
2. PSoC® 3 Architecture TRM (Technical Reference Manual) – 379 с.

***Додаткова*:**

1. PSoC® Creator™ Component Datasheet. Fan Controller – 6 с.

Приклади застосування МК PSOC

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



Мікропроцесорна техніка

2019 р.

(лекція 7, кінець)

Благітко Б.Я.