

PROXIMITY SENSOR - CSD & CY8C21X34

Project Name:	Proximity_CSD_21x34
Associated Part Families:	CY8C21x34
Software Version:	PSD5.0 SP5
Programming Language:	C
Related Hardware:	CY3213, CY3240
Authors:	Pushek Madaan, M. Ganesh Raaja

PROJECT OBJECTIVE

Build a proximity sensor using CSD user module in CY8C21x34 device.

OVERVIEW

The project builds a proximity sensor using the CSD user module in the CY8C21434 device. A loop of wire is used as a sensor and when a hand approaches the sensor, a speaker is activated with a 1.4KHz signal. The raw counts, baseline and the difference counts are monitored using a CY3240 USB-I2C bridge.

USER MODULE LIST AND PLACEMENT

The following table lists user modules used in this project and the hardware resources occupied by each user module.

User Module	Placement
CSD	ACE00, ACE01, ASE11, DBB00, DBB01, DCB02
PWM	DCB03

GLOBAL RESOURCES

Important Global Resources		
Parameter	Value	Comments
CPU_Clk	SysClk/2	Set CPU Clock to 12MHz

Note:

All other parameters are left at the default value

USER MODULE PARAMETER SETTINGS

The following tables show the user module parameter settings for each of the user modules used in the project.

CSD		
Parameter	Value	Comments
Finger Threshold	40	
Noise Threshold	20	
Baseline Update Threshold	200	
Sensor Autoreset	Enabled	This prevents the sensor getting permanently enabled
Hysteresis	10	
Debounce	3	Only if the sensor is active for 3 successive scans, it is recognized
NegativeNoiseThreshold	20	
LowBaselineReset	50	
Scanning Speed	Slow	Slow scan speed gives a better noise performance
Resolution	16	Highest resolution provides the best sensitivity required for a proximity sensing application

Modulator Capacitor	P0[3]	
Feedback Resistor	P3[1]	
Reference	ASE11	
RefValue	1	
Shield Electrode	None	

Note:

The Finger threshold, noise threshold, baseline update rate are all left at their default value. These should be adjusted according to the performance requirements of the actual hardware.

PWM		
Parameter	Value	Comments
Clock	VC2	See Note
Enable	High	
Compare Out	Row_0_Output_1	
Terminal Count	None	
Period	255	
Pulsewidth	128	
Compare Type	Less Than	
Interrupt Type	Terminal Count	
ClockSync	SyncToSysClk	As the clock is derived from SysClk synchronize it to SysClk
InvertEnable	Normal	

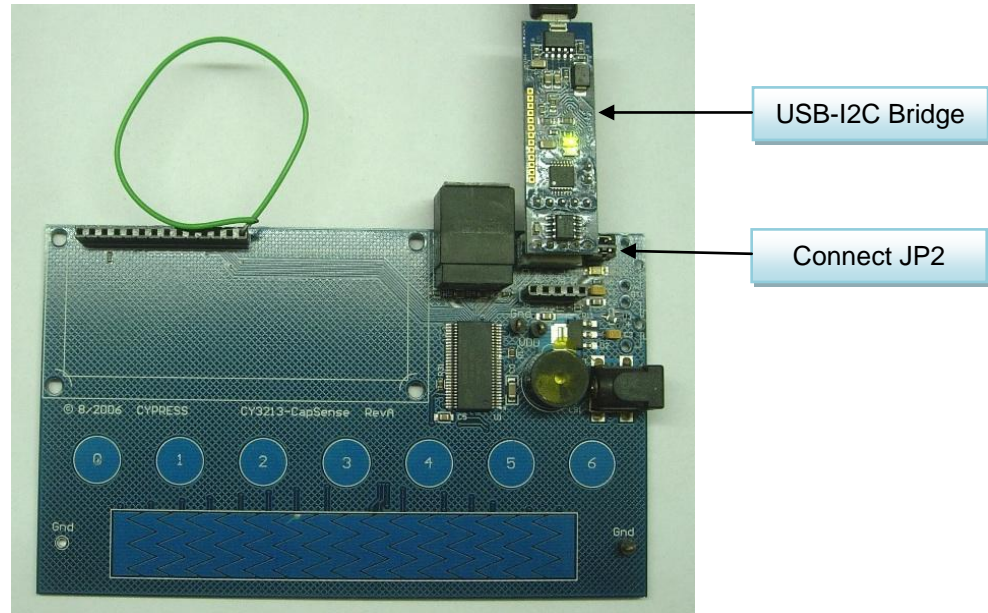
Note:

The CSD user module uses all the VC1, VC2 and VC3 clocks. The value of these dividers will depend on the scan speed and resolution settings. In this project, the scan speed is set to slow and resolution is set to 16. This results in a divider of 8 for VC1 (3MHz) and 8 for VC2 (375KHz). So, the input clock to PWM is 375KHz. The PWM divides this clock by 256 (Period + 1), resulting in an output frequency of about 1.46KHz.

HARDWARE

The project can be tested using the CY3213 evaluation board. A piece of wire is made into a loop and is fixed on Pin14 of J4, the LCD connector which connects to P2[3] of the 21434. A PWM is configured to generate around 1.4KHz signal which drives a speaker connected to P1[5]. Place a jumper on JP2 to connect P1[5] to the speaker. When a hand approaches the loop of wire the speaker sounds indicating the presence of the hand. As there is a 330 ohms resistance in series with the speaker, the volume is quite low.

The CY3240 I2C USB bridge may be connected to the ISSP header and sensor data may be observed using the bridge program. The use of CY3240 is optional and is only for debugging purposes. The project will work fine without the I2C USB bridge.



OPERATION

On power up, the code in boot.asm is executed. The device configuration is loaded inside boot.asm and all the hardware resources are configured as they are defined in the device editor. After the hardware resources are configured, main.c is executed. The following operations are performed inside main.c.

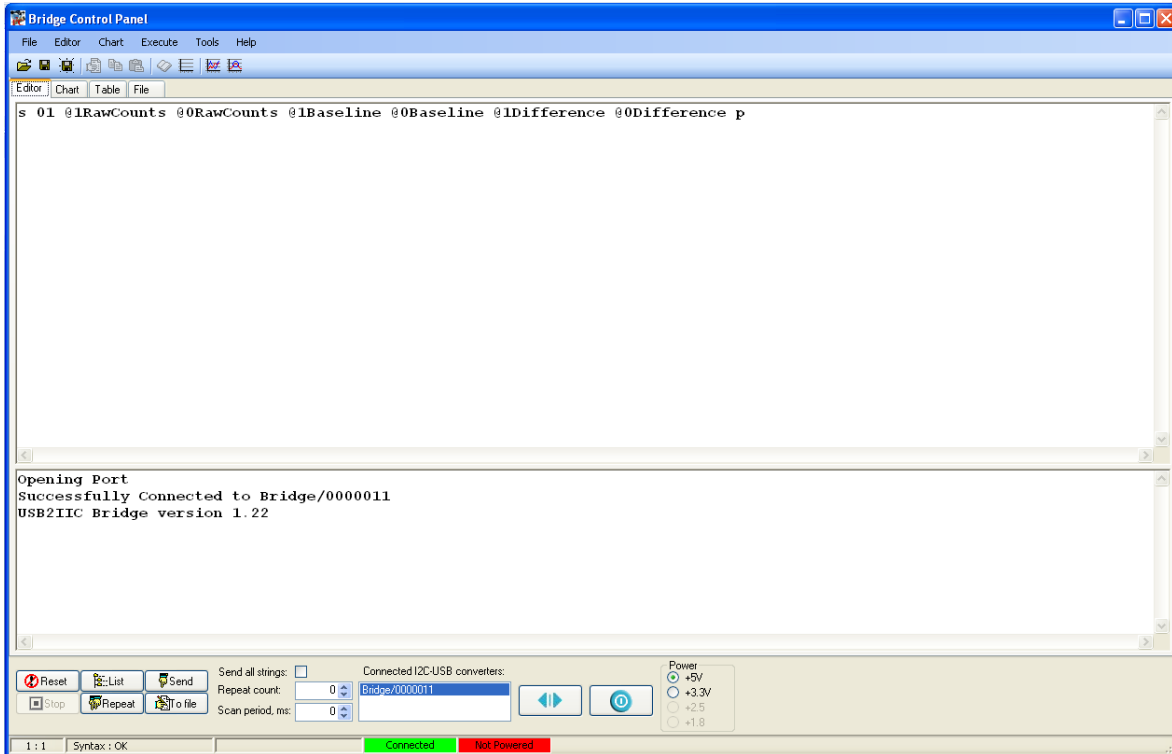
- Global interrupts are enabled.
- The EzI2C slave is setup and the RAM buffer is initialized to point to the CSD_Regs structure.
- The CSD is started.
- P1[5] is disconnected from the global bus and the PWM is started. P1[5] can now be connected or disconnected from the PWM by connecting or disconnecting from the Global bus using the PRT1GS register.
- Inside an infinite loop, the following operations are performed
 - The sensor is scanned and the baseline is updated
 - The CSD_Regs structure is loaded with the raw counts, baseline counts and difference counts from the sensor.
 - The CSD_blsSensorActive function is called to check if the sensor is activated. If yes, P1[5] is connected to the Global bus and the PWM output drives the buzzer. If the sensor is not active, then P1[5] is disconnected from global bus, thus disconnecting the PWM.

TESTING THE PROJECT

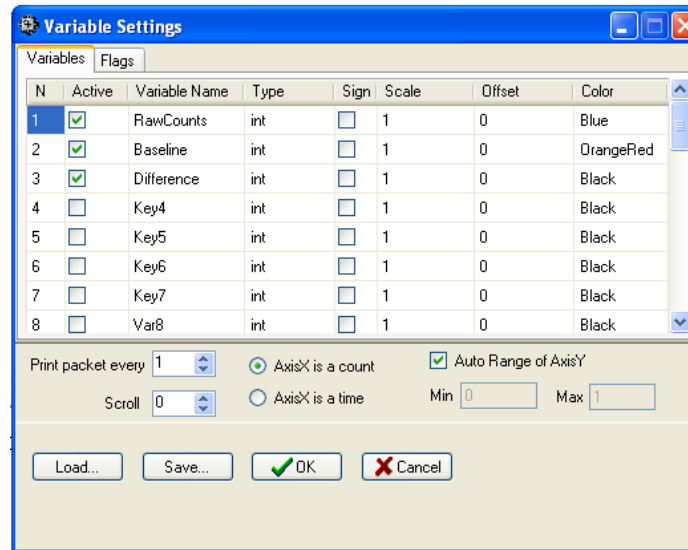
Connect a wire loop to pin 14 of LCD connector. Power up the board using either the power adaptor or by using a MiniProg. Bring the hand near the wire loop and observe the buzzer getting activated.

To debug the project using the CY3240 I2C-USB bridge:

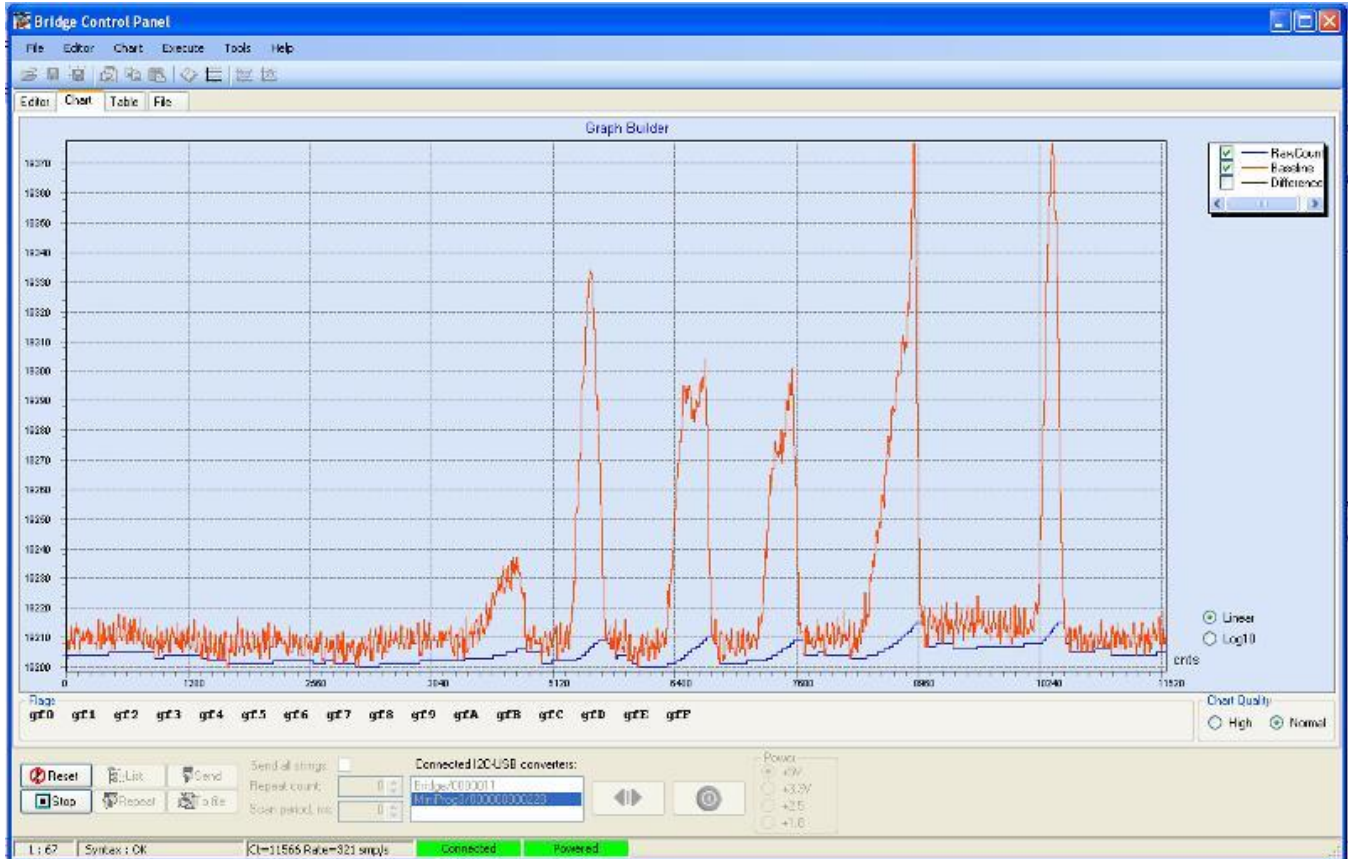
1. Connect the I2C-USB bridge to the ISSP header of the board.
2. Start the USB-I2C program in the PC and power the target board using the bridge.
3. Using File >> Open File command open the Proximity.iic file found in the Project folder. This loads the command that is to be sent to the bridge to read from the I2C slave.



- Open Chart >> Data Settings menu. This opens the Variable settings window. Click on the Load button and open the Proximity.ini file from the project folder. This configures the variables that are required to read the raw counts, baseline and difference counts from the board.



- Click on the Chart tab to open the chart view. Click on the repeat button. Now the bridge starts reading data from the CY3213 board and updates the chart. Disable the Difference data from the graph. Now bring the hand near the sensor and observe the raw counts increasing.



Copyright 2009, PlanetPSoc.com

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE AND THIS DOCUMENTATION ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.