Project Documentation DemoApplication

March 19, 2018

Contents

Part I

X2C Model

1 Version Information

1.1 X2C

- X2Cfull: Version 1405

1.2 Operating System

- OS: Windows 7 6.1

1.3 Scilab

- Scilab: Version 5.5.2.1427793548
- Java: Version 1.6.0_41

2 Model Structure

2.1 Xcos Model

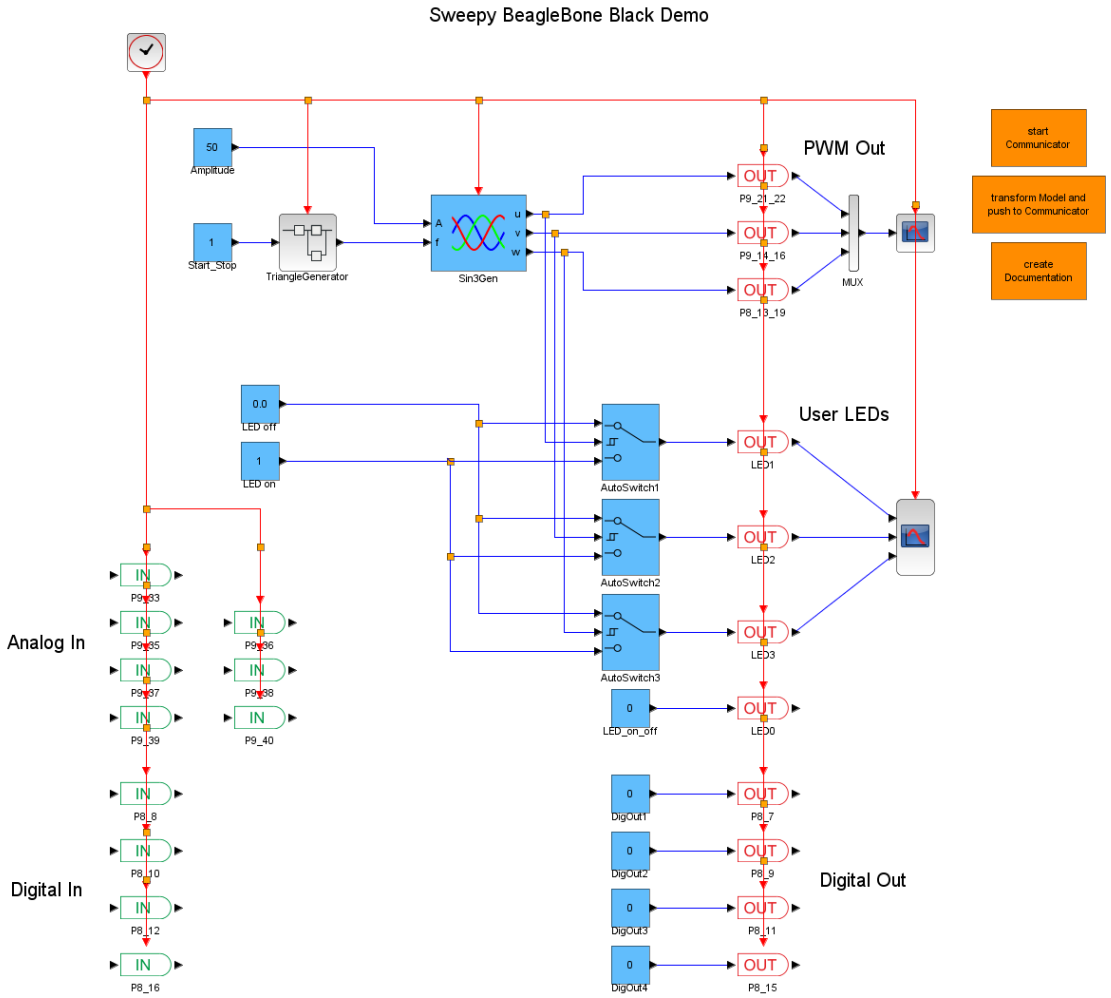


Figure 1: DemoApplication

2.2 Subsystems

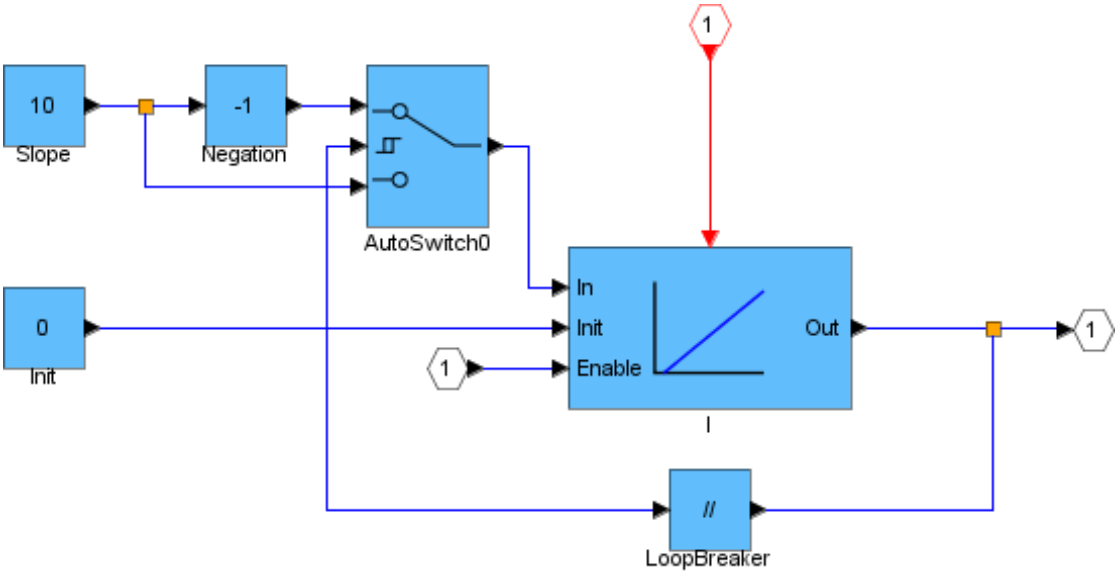


Figure 2: DemoApplication_TriangleGenerator

3 Model Parameter

3.1 Sample Time

Sample Time	
T_S	$100\mu s$

3.2 Scilab Parameter

```
// File with model parameters such as sample time, scaling factors, etc...
//
// Copyright (c) 2017, Linz Center of Mechatronics GmbH (LCM) http://www.lcm.at/
// All rights reserved.
//
// This file is licensed according to the BSD 3-clause license as follows:
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//     * Redistributions of source code must retain the above copyright
//       notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above copyright
//       notice, this list of conditions and the following disclaimer in the
//       documentation and/or other materials provided with the distribution.
//     * Neither the name of the "Linz Center of Mechatronics GmbH" and "LCM" nor
//       the names of its contributors may be used to endorse or promote products
//       derived from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
// ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
// WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
// IN NO EVENT SHALL "Linz Center of Mechatronics GmbH" BE LIABLE FOR ANY
// DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
// (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
// LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
// ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
// SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// $LastChangedRevision: 1293 $
// $LastChangedDate:: 2017-10-23 21:11:46 +0200#$
//
// This file is part of X2C. http://www.mechatronic-simulation.org/

// Sampling time
X2C_sampleTime = 100e-6;    // 10kHz sampling frequency

// Scaling factors

// Controller parameters
```

Listing 1: ModelParameter.sce

4 Mask Parameter

Constant: Amplitude	
Value	50.0
Used Implementation	Float32

AutoSwitch: AutoSwitch1	
Thresh_up	50.0
Thresh_down	50.0
Used Implementation	Float32

AutoSwitch: AutoSwitch2	
Thresh_up	50.0
Thresh_down	50.0
Used Implementation	Float32

AutoSwitch: AutoSwitch3	
Thresh_up	50.0
Thresh_down	50.0
Used Implementation	Float32

Constant: DigOut1	
Value	0.0
Used Implementation	Bool

Constant: DigOut2	
Value	0.0
Used Implementation	Bool

Constant: DigOut3	
Value	0.0
Used Implementation	Bool

Constant: DigOut4	
Value	0.0
Used Implementation	Bool

Constant: LED off	
Value	0.0
Used Implementation	Float32

Constant: LED on	
Value	1.0
Used Implementation	Float32

Constant: LED_on_off	
Value	0.0
Used Implementation	Float32

Sin3Gen: Sin3Gen	
fmax	1000.0
Offset	50.0
ts_fact	1.0
Used Implementation	Float32

Constant: Start_Stop	
Value	1.0
Used Implementation	Bool

AutoSwitch: AutoSwitch0	
Thresh_up	500.0
Thresh_down	0.0
Used Implementation	Float32

I: I	
Ki	1.0
ts_fact	1.0
Used Implementation	Float32

Constant: Init	
Value	0.0
Used Implementation	Float32

LoopBreaker: LoopBreaker	
Used Implementation	Float32

Negation: Negation	
Used Implementation	Float32

Constant: Slope	
Value	10.0
Used Implementation	Float32

Part II
Frame Program Documentation

5 File Index

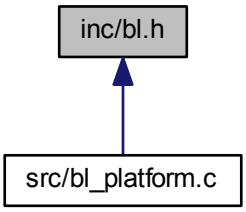
5.1 File List

Here is a list of all documented files with brief descriptions:

6 File Documentation

6.1 inc/bl.h File Reference

This file defines boot macros and objects.
This graph shows which files directly or indirectly include this file:

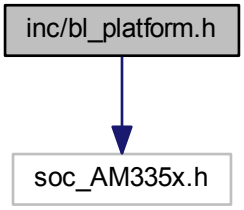


6.1.1 Detailed Description

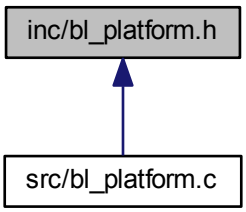
This file defines boot macros and objects.

6.2 inc/bl_platform.h File Reference

This file exports the APIs used for configuring devices required during boot.
`#include "soc_AM335x.h"`
Include dependency graph for bl_platform.h:



This graph shows which files directly or indirectly include this file:

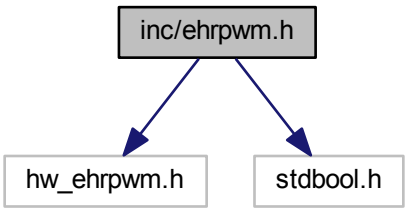


6.2.1 Detailed Description

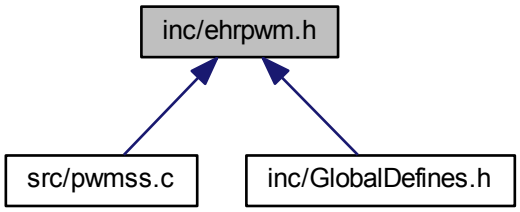
This file exports the APIs used for configuring devices required during boot.

6.3 inc/ehrpwm.h File Reference

This file contains the Macros and API prototypes for ehrpwm driver.
`#include "hw_ehrpwm.h"`
`#include <stdbool.h>`
Include dependency graph for ehrpwm.h:



This graph shows which files directly or indirectly include this file:



6.3.1 Detailed Description

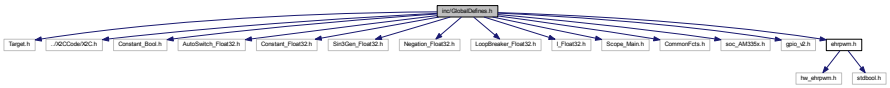
This file contains the Macros and API prototypes for ehrpwm driver.
==========

6.4 inc/GlobalDefines.h File Reference

Collection of globally needed defines.

```
#include "Target.h"
#include "../X2CCode/X2C.h"
#include "soc_AM335x.h"
#include "gpio_v2.h"
#include "ehrpwm.h"
```

Include dependency graph for GlobalDefines.h:



Macros

- #define SELECTED_SAMPLETIME SAMPLETIME_100US
- #define PWM_FREQUENCY PWM_20KHZ /∗ fPWM = 20kHz ∗/

X2C Outports

- #define **USER_LED0** (∗x2cModel.outports.bLED0) /∗ User LED 0 ∗/
- #define **USER_LED1** (∗x2cModel.outports.bLED1) /∗ User LED 1 ∗/
- #define **USER_LED2** (∗x2cModel.outports.bLED2) /∗ User LED 2 ∗/
- #define **USER_LED3** (∗x2cModel.outports.bLED2) /∗ User LED 2 ∗/

X2C Inports

- #define **AIN0** (x2cModel.inports.bP9_39) /∗ Analog input 0 ∗/
- #define **AIN1** (x2cModel.inports.bP9_40) /∗ Analog input 1 ∗/
- #define **AIN2** (x2cModel.inports.bP9_37) /∗ Analog input 2 ∗/
- #define **AIN3** (x2cModel.inports.bP9_38) /∗ Analog input 3 ∗/
- #define **AIN4** (x2cModel.inports.bP9_33) /∗ Analog input 4 ∗/

- #define **AIN5** (x2cModel.inports.bP9_36) /* Analog input 5 */
- #define **AIN6** (x2cModel.inports.bP9_35) /* Analog input 6 */

Port Pin Definitions

- #define **USER_LED0_ON** GPIOPinWrite(SOC_GPIO_1_REGS, 21, GPIO_PIN_HIGH)
- #define **USER_LED0_OFF** GPIOPinWrite(SOC_GPIO_1_REGS, 21, GPIO_PIN_LOW)
- #define **USER_LED1_ON** GPIOPinWrite(SOC_GPIO_1_REGS, 22, GPIO_PIN_HIGH)
- #define **USER_LED1_OFF** GPIOPinWrite(SOC_GPIO_1_REGS, 22, GPIO_PIN_LOW)
- #define **USER_LED2_ON** GPIOPinWrite(SOC_GPIO_1_REGS, 23, GPIO_PIN_HIGH)
- #define **USER_LED2_OFF** GPIOPinWrite(SOC_GPIO_1_REGS, 23, GPIO_PIN_LOW)
- #define **USER_LED3_ON** GPIOPinWrite(SOC_GPIO_1_REGS, 24, GPIO_PIN_HIGH)
- #define **USER_LED3_OFF** GPIOPinWrite(SOC_GPIO_1_REGS, 24, GPIO_PIN_LOW)
- #define **GPIO_1_13_ON** GPIOPinWrite(SOC_GPIO_1_REGS, 13, GPIO_PIN_HIGH)
- #define **GPIO_1_13_OFF** GPIOPinWrite(SOC_GPIO_1_REGS, 13, GPIO_PIN_LOW)
- #define **GPIO_1_15_ON** GPIOPinWrite(SOC_GPIO_1_REGS, 15, GPIO_PIN_HIGH)
- #define **GPIO_1_15_OFF** GPIOPinWrite(SOC_GPIO_1_REGS, 15, GPIO_PIN_LOW)
- #define **GPIO_2_2_ON** GPIOPinWrite(SOC_GPIO_2_REGS, 2, GPIO_PIN_HIGH)
- #define **GPIO_2_2_OFF** GPIOPinWrite(SOC_GPIO_2_REGS, 2, GPIO_PIN_LOW)
- #define **GPIO_2_5_ON** GPIOPinWrite(SOC_GPIO_2_REGS, 5, GPIO_PIN_HIGH)
- #define **GPIO_2_5_OFF** GPIOPinWrite(SOC_GPIO_2_REGS, 5, GPIO_PIN_LOW)
- #define **READ_GPIO_1_12** ((bool)(GPIOPinRead(SOC_GPIO_1_REGS, 12)))
- #define **READ_GPIO_1_14** ((bool)(GPIOPinRead(SOC_GPIO_1_REGS, 14)))
- #define **READ_GPIO_2_3** ((bool)(GPIOPinRead(SOC_GPIO_2_REGS, 3)))
- #define **READ_GPIO_2_4** ((bool)(GPIOPinRead(SOC_GPIO_2_REGS, 4)))

6.4.1 Detailed Description

Collection of globally needed defines.
Available Preprocessor Definitions:

- none

6.4.2 Macro Definition Documentation

6.4.2.1 #define PWM_FREQUENCY PWM_20KHZ /* fPWM = 20kHz */

PWM frequency

6.4.2.2 #define SELECTED_SAMPLETIME SAMPLETIME_100US

Sample time

6.5 inc/Hardware.h File Reference

Hardware initialization.

Functions

- void initHardware (void)

  Initialization of hardware.

6.5.1 Detailed Description

Hardware initialization.

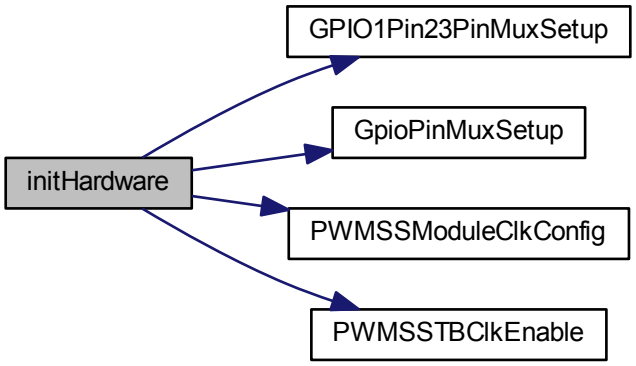
6.5.2 Function Documentation

6.5.2.1 void initHardware (void)

Initialization of hardware.

- Configuration of IO ports
- Configuration of PWM
  - Activation of modules 0, 1, 2
  - Frequency set to 20kHz
  - Center aligned mode
  - Active high complementary output mode
  - Dead band module activated, but delay is set to 0 by default
- Configuration of ADC
  - Activation of channels 0..6
  - 200kSamples/s
  - ADC is triggered by timer 4
- Configuration of timer 4
  - 24MHz timer clock
  - Generation of cyclic interrupt with selected sample time
  - Interrupt calls X2C main task

Here is the call graph for this function:



6.6 inc/InputControl.h File Reference

Handling of inputs.

Functions

- void readAnalogIn (void)

  Routine to read values from ADC.
- void readDigitalIn (void)

  Routine to read digital input pins.

6.6.1 Detailed Description

Handling of inputs.

- Reading of digital inputs
- Reading of analog inputs

6.6.2 Function Documentation

6.6.2.1 void readDigitalIn (void)

Routine to read digital input pins.

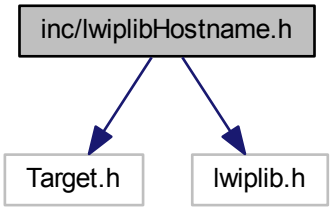
- read header P8 pin 8
- read header P8 pin 10
- read header P8 pin 12
- read header P8 pin 16

6.7 inc/lwiplibHostname.h File Reference

Extended lwIP configuration.

```
#include "Target.h"
#include "lwiplib.h"
```

Include dependency graph for lwiplibHostname.h:



Functions

- uint32 lwIPInitHostname (LWIP_IF *lwipIf, char *hostname)

  Initializes lwip TCP/IP stack with hostname.

6.7.1 Detailed Description

Extended lwIP configuration.
Modified lwiplib from TI StarterWare to support hostname configuration. Requires lwIP 1.4.0.

6.7.2 Function Documentation

6.7.2.1 uint32 lwIPInitHostname (LWIP_IF * *lwiplf,* char * *hostname*)

Initializes lwip TCP/IP stack with hostname.

Parameters

lwipIf	lwIP interface structure
hostname	Hostname

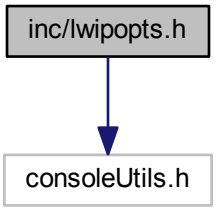
Returns

IP address

6.8 inc/lwipopts.h File Reference

```
#include "consoleUtils.h"
```

Include dependency graph for lwipopts.h:



Macros

- #define LWIP_NETIF_HOSTNAME 1

6.8.1 Detailed Description

- Configuration options for lwIP

Copyright (c) 2010 Texas Instruments Incorporated

6.8.2 Macro Definition Documentation

6.8.2.1 #define LWIP_NETIF_HOSTNAME 1

User specific macros.

6.9 inc/Main.h File Reference

Main function.

Functions

- void mainTask (void)

  Main control task.

6.9.1 Detailed Description

Main function.
X2C maintenance table, protocol & hardware initialization.

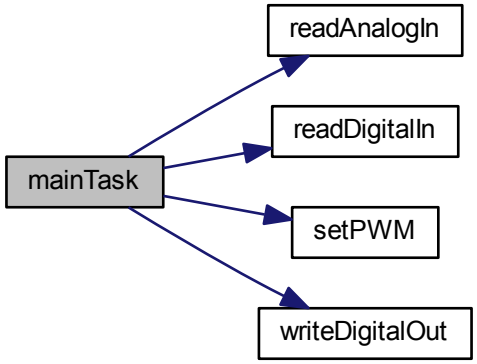
6.9.2 Function Documentation

6.9.2.1 void mainTask (void)

Main control task.
This task is/has to be called periodically. Calling rate = Sample time defined in GlobalDefines.h

- assign inports
- update X2C
- update outports

Here is the call graph for this function:



6.10 inc/MMUConfig.h File Reference

MMU configuration.

6.10.1 Detailed Description

MMU configuration.

6.11 inc/OutputControl.h File Reference

Handling of outputs.

Functions

- void setPWM (void)

  Routine to set PWM duty cycle.
- void writeDigitalOut (void)

  Routine to write to digital output pins.

6.11.1 Detailed Description

Handling of outputs.

- Setting duty cycle of PWM signals
- Setting of digital outputs

6.11.2 Function Documentation

6.11.2.1 void setPWM (void)

Routine to set PWM duty cycle.

- check range of duty cycle
- set duty cycle in PWM module

6.11.2.2 void writeDigitalOut (void)

Routine to write to digital output pins.

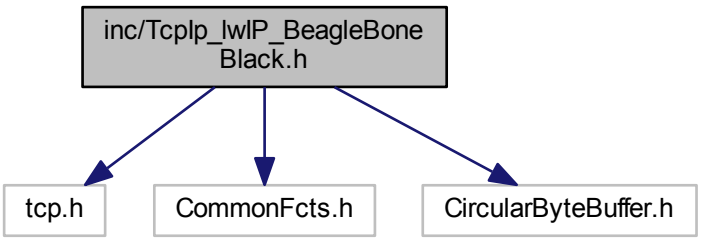
- LEDs
- General purpose outputs

6.12 inc/TcpIp_lwIP_BeagleBoneBlack.h File Reference

TCP/IP driver for BeagleBone Black.

```
#include "tcp.h"
#include "CommonFcts.h"
#include "CircularByteBuffer.h"
```

Include dependency graph for TcpIp_lwIP_BeagleBoneBlack.h:



Functions

- uint32 initTcpIp (tTcpIp *tcpip, uint32 ipAddress, uint16 port, char *hostname, uint8 *rxBuffer, uint16 rxSize, uint8 *txBuffer, uint16 txSize)
- void linkTcpIp (tProtocol *protocol, tTcpIp *tcpip)
- void closeConnection (tTcpIp *tcpip)
- uint16 getAvailableTcpIpBytes (tTcpIp *tcpip)

- uint16 getTcpIpData (tTcpIp *tcpip, uint8 *data, uint16 len)
- uint16 getFreeTcpIpBytes (tTcpIp *tcpip)
- uint16 putTcpIpData (tTcpIp *tcpip, uint8 *data, uint16 len)

6.12.1 Detailed Description

TCP/IP driver for BeagleBone Black.
TCP/IP driver implementation for BeagleBone Black board. Uses lwIP 1.4.0 stack for communication.

6.12.2 Function Documentation

6.12.2.1 void closeConnection (tTcpIp * *tcpip*)

Closes connection.
Parameters

tcpip	TCP/IP interface

6.12.2.2 uint16 getAvailableTcpIpBytes (tTcpIp * *tcpip*)

Returns available bytes in receive buffer.
Parameters

tcpip	TCIP/IP interface

Returns

The number of bytes being available to receive

6.12.2.3 uint16 getFreeTcpIpBytes (tTcpIp * *tcpip*)

Returns free bytes in transmit buffer.
Parameters

tcpip	TCP/IP interface

Returns

The number of free bytes in the transmit buffer

6.12.2.4 uint16 getTcpIpData (tTcpIp * *tcpip*, uint8 * *data*, uint16 *len*)

Reads data from communication buffers.
Parameters

tcpip	TCIP/IP interface

data	User data buffer
len	User data buffer len

Returns

The number of bytes being transferred

Note

The caller must guarantee correct buffer size.

6.12.2.5 uint32 initTcpIp (tTcpIp * *tcpip,* uint32 *ipAddress,* uint16 *port,* char * *hostname,* uint8 * *rxBuffer,* uint16 *rxSize,* uint8 * *txBuffer,* uint16 *txSize*)

Initializes TCP/IP interface.

Parameters

tcpip	TCP/IP interface
IP	address
port	TCP port
hostname	Hostname being used in DHCP (option #12)
rxBuffer	Receive buffer
rxSize	Receive buffer length
txBuffer	Transmit buffer
txSize	Transmit buffer length

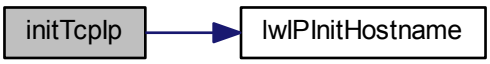
Returns

Assigned IP address

Note

Static IP address example: 192.168.0.1 => 0xC0A80001
If DHCP IP address is desired, the IP address has to be set to zero

Here is the call graph for this function:



6.12.2.6 void linkTcpIp (tProtocol * *protocol,* tTcpIp * *tcpip*)

Links protocol with TCP/IP interface.

Parameters

protocol	Protocol
tcpip	TCP/IP interface

6.12.2.7 uint16 putTcpIpData (tTcpIp * *tcpip*, uint8 * *data*, uint16 *len*)

Writes data to communication buffers.

Parameters

tcpip	TCIP/IP connection
data	User data buffer
len	User data buffer len

Returns

The number of bytes being transferred

Note

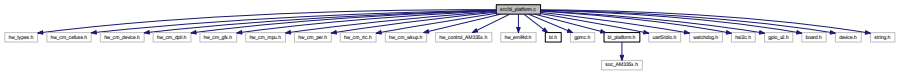
The caller must guarantee correct buffer size.

6.13 src/bl_platform.c File Reference

Initializes AM335x Device Peripherals.

```
#include "hw_types.h"
#include "hw_cm_cefuse.h"
#include "hw_cm_device.h"
#include "hw_cm_dpll.h"
#include "hw_cm_gfx.h"
#include "hw_cm_mpu.h"
#include "hw_cm_per.h"
#include "hw_cm_rtc.h"
#include "hw_cm_wkup.h"
#include "hw_control_AM335x.h"
#include "hw_emif4d.h"
#include "bl.h"
#include "gpmc.h"
#include "bl_platform.h"
#include "uartStdio.h"
#include "watchdog.h"
#include "hsi2c.h"
#include "gpio_v2.h"
#include "board.h"
#include "device.h"
#include "string.h"
```

Include dependency graph for bl_platform.c:



Functions

- void ConfigureVdd2 (unsigned int opVolMultiplier, unsigned maxLoadCurrent, unsigned int timeStep, unsigned int supplyState)
  - *Configure vdd2 for various parameters such as Multiplier, Maximum Load Current etc .*
- void SelectVdd2Source (unsigned int vddSource)

  Select the VDD2 value. VDD2_OP_REG or VDD2_SR_REG.
- void SetVdd2OpVoltage (unsigned int opVolSelector)

  set VDD2_OP voltage value.
- void SetVdd2SrVoltage (unsigned int opVolSelector)

  set VDD2_SR voltage value
- void SelectI2CInstance (unsigned int i2cInstance)

  Select I2C interface whether SR I2C or Control I2C.
- void ConfigureVdd1 (unsigned int opVolMultiplier, unsigned maxLoadCurrent, unsigned int timeStep, unsigned int supplyState)
  - *Configure vdd1 for various parameters such as Multiplier, Maximum Load Current etc .*
- void SelectVdd1Source (unsigned int vddSource)

  Select the VDD1 value. VDD1_OP_REG or VDD1_SR_REG.
- void SetVdd1OpVoltage (unsigned int opVolSelector)

  set VDD1_OP voltage value.

6.13.1 Detailed Description

Initializes AM335x Device Peripherals.

6.13.2 Function Documentation

6.13.2.1 void ConfigureVdd1 (unsigned int *opVolMultiplier,* unsigned *maxLoadCurrent,* unsigned int *timeStep,* unsigned int *supplyState*)

- Configure vdd1 for various parameters such as Multiplier, Maximum Load Current etc .

Parameters

opVolMultiplier	- Multiplier.
maxLoadCurrent	- Maximum Load Current.
timeStep	- Time step - voltage change per us(micro sec).
supplyState	- Supply state (on (high/low power mode), off)

Returns

: None.

6.13.2.2 void ConfigureVdd2 (unsigned int *opVolMultiplier,* unsigned *maxLoadCurrent,* unsigned int *timeStep,* unsigned int *supplyState*)

- Configure vdd2 for various parameters such as Multiplier, Maximum Load Current etc .

Parameters

opVolMultiplier	- Multiplier.
maxLoadCurrent	- Maximum Load Current.
timeStep	- Time step - voltage change per us(micro sec).
supplyState	- Supply state (on (high/low power mode), off)

Returns

: None.

6.13.2.3 void SelectI2CInstance (unsigned int *i2cInstance*)

Select I2C interface whether SR I2C or Control I2C.
Parameters

i2cInstance	- I2c instance to select.

Returns

None.

6.13.2.4 void SelectVdd1Source (unsigned int *vddSource*)

Select the VDD1 value. VDD1_OP_REG or VDD1_SR_REG.
Parameters

vddSource	- VDD2 value.

Returns

None.

6.13.2.5 void SelectVdd2Source (unsigned int *vddSource*)

Select the VDD2 value. VDD2_OP_REG or VDD2_SR_REG.
Parameters

vddSource	- VDD2 value.

Returns

None.

6.13.2.6 void SetVdd1OpVoltage (unsigned int *opVolSelector*)

set VDD1_OP voltage value.

Parameters

opVolSelector	- VDD2_OP voltage value.

Returns

None.

6.13.2.7 void SetVdd2OpVoltage (unsigned int *opVolSelector*)

set VDD2_OP voltage value.

Parameters

opVolSelector	- VDD2_OP voltage value.

Returns

None.

6.13.2.8 void SetVdd2SrVoltage (unsigned int *opVolSelector*)

set VDD2_SR voltage value

Parameters

opVolSelector	- VDD2_SR voltage value.

Returns

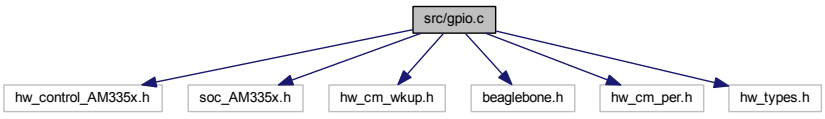
None.

6.14 src/gpio.c File Reference

This file contains functions which performs the platform specific configurations of GPIO.

```
#include "hw_control_AM335x.h"
#include "soc_AM335x.h"
#include "hw_cm_wkup.h"
#include "beaglebone.h"
#include "hw_cm_per.h"
#include "hw_types.h"
```

Include dependency graph for gpio.c:



Functions

- void GPIO1Pin23PinMuxSetup (void)

  This function does the Pin Multiplexing and selects GPIO pin GPIO1[23] for use. GPIO1[23] means 23rd pin of GPIO1 instance. This pin can be used to control the Audio Buzzer.
- void GpioPinMuxSetup (unsigned int offsetAddr, unsigned int padConfValue)

  This function does the pin multiplexing for any GPIO Pin.

6.14.1 Detailed Description

This file contains functions which performs the platform specific configurations of GPIO.

6.14.2 Function Documentation

6.14.2.1 void GPIO1Pin23PinMuxSetup (void)

This function does the Pin Multiplexing and selects GPIO pin GPIO1[23] for use. GPIO1[23] means 23rd pin of GPIO1 instance. This pin can be used to control the Audio Buzzer.

Parameters

None	

Returns

None

Note

Either of GPIO1[23] or GPIO1[30] pins could be used to control the Audio Buzzer.

6.14.2.2 void GpioPinMuxSetup (unsigned int *offsetAddr,* unsigned int *padConfValue*)

This function does the pin multiplexing for any GPIO Pin.

Parameters

offsetAddr	This is the offset address of the Pad Control Register corresponding to the GPIO pin. These addresses are offsets with respect to the base address of the Control Module.
padConfValue	This is the value to be written to the Pad Control register whose offset address is given by 'offsetAddr'.

The 'offsetAddr' and 'padConfValue' can be obtained from macros defined in the file 'include/armv7a/am335x/pin_mux.h'.

Returns

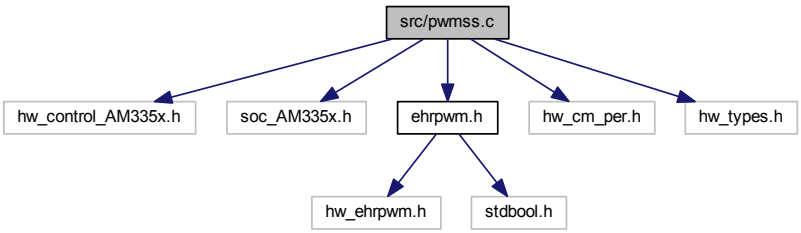
None.

6.15 src/pwmss.c File Reference

This file contains functions which does platform specific configurations for PWMSS.

```
#include "hw_control_AM335x.h"
#include "soc_AM335x.h"
#include "ehrpwm.h"
#include "hw_cm_per.h"
#include "hw_types.h"
```

Include dependency graph for pwmss.c:



Functions

- void PWMSSTBClkEnable (unsigned int instance)

  This function Enables TBCLK(Time Base Clock) for specific EPWM instance of pwmsubsystem.
- void PWMSSModuleClkConfig (unsigned int instanceNum)

  This function configures the L3 and L4_PER system clocks. It also configures the system clocks for the specified ePWMSS instance.

6.15.1 Detailed Description

This file contains functions which does platform specific configurations for PWMSS.

6.15.2 Function Documentation

6.15.2.1 void PWMSSModuleClkConfig (unsigned int *instanceNum*)

This function configures the L3 and L4_PER system clocks. It also configures the system clocks for the specified ePWMSS instance.

Parameters

instanceNum	The instance number of ePWMSS whose system clocks have to be configured.

'instanceNum' can take one of the following values: (0 <= instanceNum <= 2)

Returns

None.

6.15.2.2 void PWMSSTBClkEnable (unsigned int *instance*)

This function Enables TBCLK(Time Base Clock) for specific EPWM instance of pwmsubsystem.

Parameters

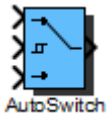
instance	It is the instance number of EPWM of pwmsubsystem.

Part III
Used X2C-Blocks

7 Project Specific Blocks

8 Internal Library Blocks

Block: AutoSwitch



Inports	
In1	Input #1
Switch	Input #2: Threshold signal
In3	Input #3

Outports	
Out	Either value of input #1 or input #3 dependent on value of input #2

Mask Parameters	
Thresh_up	Threshold level for rising switch signal
Thresh_down	Threshold level for falling switch signal

Description:

Switch between In1 and In3 dependent on Switch signal:
Switch signal rising: Switch >= Threshold up –> Out = In1
Switch signal falling: Switch < Threshold down –> Out = In3

Implementations:

FiP8 8 Bit Fixed Point Implementation
FiP16 16 Bit Fixed Point Implementation
FiP32 32 Bit Fixed Point Implementation
Float32 32 Bit Floating Point Implementation
Float64 64 Bit Floating Point Implementation

Implementation: FiP8

8 Bit Fixed Point Implementation

Inports Data Type	
In1	int8
Switch	int8
In3	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In1	int16
Switch	int16
In3	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In1	int32
Switch	int32
In3	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In1	float32
Switch	float32
In3	float32

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In1	float64
Switch	float64
In3	float64

Outports Data Type	
Out	float64

Block: Constant

Outports	
Out	Constant output

Mask Parameters	
Value	Constant factor

Description:

Constant value.

Implementations:

Bool	Boolean Implementation
FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: Bool

Boolean Implementation

Outports Data Type	
Out	bool

Implementation: FiP8

8 Bit Fixed Point Implementation

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

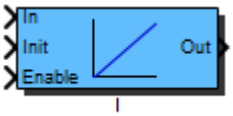
Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Outports Data Type	
Out	float64

Block: I



Inports	
In	Control error input
Init	Value which is loaded at initialization function call
Enable	Enable == 0: Deactivation of block; Out set to 0 Enable 0->1: Preload of integral part Enable == 1: Activation of block

Outports	
Out	Control value

Mask Parameters	
Ki	Integral Factor
ts_fact	Multiplication factor of base sampling time (in integer format)

Description:

I controller:
G(s) = Ki/s = 1/(Ti*s)

Each fixed point implementation uses the next higher integer datatype for the integrational value storage variable.
A rising flank at the *Enable* inport will preload the integrational part with the value present on the *Init* inport.

Transfer function (zero-order hold discretization method):

$$G(z) = K_{\mathrm{i}} T_{\mathrm{s}} \frac{1}{z-1}$$

Implementations:

FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP8

8 Bit Fixed Point Implementation

Inports Data Type	
In	int8
Init	int8
Enable	bool

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In	int16
Init	int16
Enable	bool

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32
Init	int32
Enable	bool

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32
Init	float32
Enable	bool

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64
Init	float64
Enable	bool

Outports Data Type	
Out	float64

Block: LoopBreaker

Inports	
In	Input In(k)

Outports	
Out	Output Out(k)=In(k-1)

Description:

Block to break algebraic loops.

Implementations:

Bool	Boolean Integration
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: Bool

Boolean Integration

Inports Data Type	
In	bool

Outports Data Type	
Out	bool

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32

Outports Data Type	
Out	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: Negation

Inports	
In	Input

Outports	
Out	Negated input value

Description:

Negation of input signal.

Calculation:

$$\mathrm{Out} = -\mathrm{In}$$

Implementations:

FiP8	8 Bit Fixed Point Implementation
FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP8

8 Bit Fixed Point Implementation

Inports Data Type	
In	int8

Outports Data Type	
Out	int8

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
In	int16

Outports Data Type	
Out	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
In	int32

Outports Data Type	
Out	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
In	float32

Outports Data Type	
Out	float32

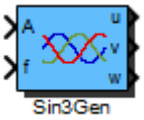
Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
In	float64

Outports Data Type	
Out	float64

Block: Sin3Gen



Inports	
A	Amplitude
f	Frequency

Outports	
u	Sine wave output phase u
v	Sine wave output phase v
w	Sine wave output phase w

Mask Parameters	
fmax	Maximum Frequency in Hz
Offset	Offset
ts_fact	Multiplication factor of base sampling time (in integer format)

Description:

Generation of a 3 sine waves with amplitude (A) and frequency (f).

Calculation fixed point implementation:

$$
\begin{aligned}
u_k &= A_k \sin\left(2 f_k f_{\max} k T_\mathrm{s}\right) + A_\mathrm{offset} \\
v_k &= A_k \sin\left(2 f_k f_{\max} k T_\mathrm{s} - \frac{2\pi}{3}\right) + A_\mathrm{offset} \\
w_k &= A_k \sin\left(2 f_k f_{\max} k T_\mathrm{s} + \frac{2\pi}{3}\right) + A_\mathrm{offset}
\end{aligned}
$$

For sine calculation a lookup table with 256 entries is used. This results in a short computation time but with the downside of reduced accuracy for the FiP32 implementation.

Calculation floating point implementation (parameter *f_max* is ignored):

$$
\begin{aligned}
u_k &= A_k \sin\left(2\pi f_k k T_\mathrm{s}\right) + A_\mathrm{offset} \\
v_k &= A_k \sin\left(2\pi f_k k T_\mathrm{s} - \frac{2\pi}{3}\right) + A_\mathrm{offset} \\
w_k &= A_k \sin\left(2\pi f_k k T_\mathrm{s} + \frac{2\pi}{3}\right) + A_\mathrm{offset}
\end{aligned}
$$

Implementations:

FiP16	16 Bit Fixed Point Implementation
FiP32	32 Bit Fixed Point Implementation
Float32	32 Bit Floating Point Implementation
Float64	64 Bit Floating Point Implementation

Implementation: FiP16

16 Bit Fixed Point Implementation

Inports Data Type	
A	int16
f	int16

Outports Data Type	
u	int16
v	int16
w	int16

Implementation: FiP32

32 Bit Fixed Point Implementation

Inports Data Type	
A	int32
f	int32

Outports Data Type	
u	int32
v	int32
w	int32

Implementation: Float32

32 Bit Floating Point Implementation

Inports Data Type	
A	float32
f	float32

Outports Data Type	
u	float32
v	float32
w	float32

Implementation: Float64

64 Bit Floating Point Implementation

Inports Data Type	
A	float64
f	float64

Outports Data Type	
u	float64
v	float64
w	float64