De1-SoC Computer Driver API
ECSE 324, Fall 2017
McGill University

Slider Switches

Header File

slider_switches.h

Datatypes

None

Functions

1. **read_slider_switches_ASM**
   Arguments: None
   Return type: Integer
   Description: Returns the value of the last 10 bits in the Slider Switch *Data* register.

Red LEDs

Header File

LEDs.h

Datatypes

None

Functions

1. **read_LEDs_ASM**
   Arguments: None
   Return type: Integer
   Description: Returns the value of the last 10 bits in the red LED *Data* register.

2. **write_LEDs_ASM**
   Arguments:
   int LEDs
   Return type: None
   Description: Writes the last 10 bits in the argument LEDs to the last 10 bits in the red LED *Data* register.

Pushbuttons

Header File

pushbuttons.h

Datatypes

1. **PB_t**
   Type: Enum
   Members:

Name	Value
PB0	0x00000001
PB1	0x00000002
PB2	0x00000004
PB3	0x00000008

   Description: A datatype that defines a one-hot encoding scheme for the four pushbuttons

Functions

1. **read_PB_data_ASM**
   Arguments: None
   Return type: Integer
   Description: Returns the value of the last 4 bits in the pushbutton *Data* register.

2. **PB_data_is_pressed_ASM**
   Arguments:
   PB_t PB
   Return type: Integer
   Description: If any of the bits in the pushbutton *Data* register corresponding to the pushbutton(s) passed in the argument PB is set to 1, then the function returns 1, otherwise it returns 0.

3. **read_PB_edgecap_ASM**
   Arguments: None
   Return type: Integer
   Description: Returns the value of the last 4 bits in the pushbutton *Edgecapture* register.

4. **PB_edgecap_is_pressed_ASM**
   Arguments:
   PB_t PB
   Return type: Integer
   Description: If any of the bits in the pushbutton *Edgecapture* register corresponding to the pushbutton(s) passed in the argument PB is set to 1, then the function returns 1, otherwise it returns 0.

5. **PB_clear_edgecp_ASM**
   Arguments:
   PB_t PB
   Return type: None
   Description: Clears the bits in the pushbutton *Edgecapture* register corresponding to the pushbutton(s) passed in the argument PB.

6. **enable_PB_INT_ASM**
   Arguments:
   PB_t PB
   Return type: None
   Description: Sets to 1, the bits in the pushbutton *Interruptmask* register corresponding to the pushbutton(s) passed in the argument PB.

7. **disable_PB_INT_ASM**
   Arguments:
   PB_t PB
   Return type: None
   Description: Sets to 0, the bits in the pushbutton *Interruptmask* register corresponding to the pushbutton(s) passed in the argument PB.

HEX Displays

Header File

HEX_displays.h

Datatypes

1. **HEX_t**
   Type: Enum
   Members:

Name	Value
HEX0	0x00000001
HEX1	0x00000002
HEX2	0x00000004
HEX3	0x00000008
HEX4	0x00000010
HEX5	0x00000020

   Description: A datatype that defines a one-hot encoding scheme for the six HEX displays

Functions

1. **HEX_clear_ASM**
   Arguments:
   HEX_t hex
   Return type: None
   Description: Turns off all segments of the HEX display(s) passed in argument hex.

2. **HEX_flood_ASM**
   Arguments:
   HEX_t hex
   Return type: None
   Description: Turns on all segments of the HEX display(s) passed in argument hex.

3. **HEX_write_ASM**
   Arguments:
   HEX_t hex
   char val
   Return type: None
   Description: Writes the hexadecimal digit (0 - F) to the HEX display(s) passed in argument hex.

HPS Timers

Header File

HPS_TIM.h

Datatypes

1. **HPS_TIM_t**
   Type: Enum
   Members:

Name	Value
TIM0	0x00000001
TIM1	0x00000002
TIM2	0x00000004
TIM3	0x00000008

Description: A datatype that defines a one-hot encoding scheme for the four HPS timers

2. **HPS_TIM_config_t**
   Type: Struct
   Members:

Name	Datatype	Description
Tim	HPS_TIM_t	The current timer(s) being configured. Multiple TIM instances allowed
Timeout	int	The desired timeout value in microseconds
LD_en	int	Set to 1 to achieve the desired timeout, or 0 for maximum count of timer
INT_en	int	Set to 1 to enable interrupts, or 0 to disable interrupts
Enable	int	Set to 1 to activate the desired timers, or 0 to deactivate them
current_time	int	(Future use) The current timer count, not implemented in this version

   Description: A struct that is used to configure the different parameters of the HPS timers.

Functions

1. **HPS_TIM_config_ASM**
   Arguments:
   HPS_TIM_config_t *param
   Return type: None
   Description: Configures the timer instances according to the configuration struct stored at the address in argument param. Multiple timers can be configures via the same struct, and the driver handles the different hardware abstractions internally.

2. **HPS_TIM_clear_INT_ASM**
   Arguments:
   HPS_TIM_t tim
   Return type: None
   Description: Clears the *F* and *S* bits of the timer(s) passed in argument tim.

VGA

Header File

vga.h

Datatypes

None

Functions

1. **VGA_clear_charbuff_ASM**
   Arguments: None
   Return type: None
   Description: Sets all valid memory locations in the VGA character buffer to 0.

2. **VGA_clear_pixelbuff_ASM**
   Arguments: None
   Return type: None
   Description: Sets all valid memory locations in the VGA pixel buffer to 0.

3. **VGA_write_char_ASM**
   Arguments:
   int x
   int y
   char c
   Return type:None
   Description: The function first checks whether arguments x and y are valid offsets for the VGA character buffer (i.e. x is in [0,79] and y is in [0,59]), and if so, stores the value in argument c to the VGA character buffer address given by offsets x,y.

4. **VGA_write_byte_ASM**
   Arguments:
   int x
   int y
   char byte
   Return type:None
   Description: The function first checks whether arguments x and y are valid offsets for the VGA character buffer **when writing a byte**(i.e. x is in [0,78] and y is in [0,59]), and if so, stores two characters corresponding to the hexadecimal representation of the argument byte, starting at the character buffer address given by offsets x,y.

5. **VGA_draw_point_ASM**
   Arguments:
   int x
   int y
   short colour
   Return type:None
   Description: The function first checks whether arguments x and y are valid offsets for the VGA pixel buffer (i.e. x is in [0,319] and y is in [0,239]), and if so, stores the value in argument colour to the VGA pixel buffer address given by offsets x,y.

Audio

Header File

audio.h

Datatypes

None

Functions

1. **audio_read_data_ASM**
   Arguments:
   int *leftdata
   int *rightdata
   Return type: Integer
   Description: If there is valid data in **both** the left-channel and right-channel read FIFOs, the value in the *Leftdata* and *Rightdata* registers is stored at the address pointed to by the arguments char pointer leftdata and rightdata respectively, and the function returns a value of 1. If there is no valid data in either of the read FIFOs, the function simply returns 0.

2. **audio_read_leftdata_ASM**
   Arguments:
   int *data
   Return type: Integer
   Description: If there is valid data in the left-channel read FIFO, the value in the *Leftdata* register is stored at the address pointed to by the argument char pointer data, and the function returns a value of 1. If there is no valid data, the function simply returns 0.

3. **audio_read_rightdata_ASM**
   Arguments:
   int *data
   Return type: Integer
   Description: If there is valid data in the right-channel read FIFO, the value in the *Rightdata* register is stored at the address pointed to by the argument char pointer data, and the function returns a value of 1. If there is no valid data, the function simply returns 0.

4. **audio_write_data_ASM**
   Arguments:
   int leftdata
   int rightdata
   Return type: Integer
   Description: If there is space in **both** the left-channel and right-channel write FIFOs, then the value in the arguments leftdata and rightdata is written to the *Leftdata* and *Rightdata* registers respectively, and the function returns a value of 1. If there is no space in either one of the FIFOs, the functions simply returns 0.

5. **audio_write_leftdata_ASM**
   Arguments:
   int data
   Return type: Integer
   Description: If there is space in the left-channel write FIFO, then the value in the argument data is written to the *Leftdata* register and the function returns a value of 1. If there is no space, the functions simply returns 0.

6. **audio_write_rightdata_ASM**
   Arguments:
   int data
   Return type: Integer
   Description: If there is space in the right-channel write FIFO, then the value in the argument data is written to the *Rightdata* register and the function returns a value of 1. If there is no space, the functions simply returns 0.

7. **audio_enable_read_fifo_clear_ASM**
   Arguments: None
   Return type: None
   Description: Sets the *CR* field in the *Control* register to 1.

8. **audio_enable_write_fifo_clear_ASM**
   Arguments: None
   Return type: None
   Description: Sets the *CW* field in the *Control* register to 1.

9. **audio_disable_read_fifo_clear_ASM**
   Arguments: None
   Return type: None
   Description: Sets the *CR* field in the *Control* register to 0.

10. **audio_disable_write_fifo_clear_ASM**
    Arguments: None
    Return type: None
    Description: Sets the C*W* field in the *Control* register to 0.

11. **audio_enable_read_int_ASM**
    Arguments: None
    Return type: None
    Description: Sets the *RI* field in the *Control* register to 1.

12. **audio_enable_write_int_ASM**
    Arguments: None
    Return type: None
    Description: Sets the *WI* field in the *Control* register to 1.

13. **audio_disable_read_int_ASM**
    Arguments: None
    Return type: None
    Description: Sets the *RI* field in the *Control* register to 0.

14. **audio_disable_write_int_ASM**
    Arguments: None
    Return type: None
    Description: Sets the *WI* field in the *Control* register to 0.

15. **audio_read_wslc_ASM**
    Arguments None
    Return type: Integer
    Description: Returns the value in the *WSLC* field of the *Fifospace* register.

16. **audio_read_wsrc_ASM**
    Arguments: None
    Return type: Integer
    Description: Returns the value in the *WSRC* field of the *Fifospace* register.

17. **audio_read_ralc_ASM**
    Arguments: None
    Return type: Integer
    Description: Returns the value in the *RALC* field of the *Fifospace* register.

18. **audio_read_rarc_ASM**
    Arguments: None
    Return type: Integer
    Description: Returns the value in the *RARC* field of the *Fifospace* register.

PS/2 Interface

Header File

ps2_keyboard.h

Datatypes

None

Functions

1. **read_ps2_data_ASM**
   Arguments:
   char *data
   Return type: Integer
   Description: If there is valid data in the PS/2 FIFO, the value in the *data* field of the *PS2_Data* register is stored at the address pointed to by the argument char pointer data, and the function returns a value of 1. If there is no valid data in the PS/2 FIFO, the function simply returns 0.

2. **enable_ps2_int_ASM**
   Arguments: None
   Return type: None
   Description: Sets the *RE* bit in the *PS2_Control* register to 1.

3. **disable_ps2_int_ASM**
   Arguments: None
   Return type: None
   Description: Sets the *RE* bit in the *PS2_Control* register to 0.