Appendix

November 7, 2010

ROSA User API

Contents

1 Kernel control

1.1 Brief API list

```
void ROSA_init(void);
void ROSA_start(void);
void ROSA_yield(void);
tcb * ROSA_tcbCreate(tcb * TCB, char *id,
                     void *taskFunction,
                     int *stack, int stackSize);
void ROSA_tcbInstall(tcb * TCB);
//Memory
void * malloc(size_t size)
void free(void *);
```

1.2 Detailed API list

1.2.1 ROSA_init

Prototype:	void ROSA_init(void)
Description:	Initialize the ROSA kernel.
Parameters:	None.
Return value:	Nothing.

1.2.2 ROSA_start

Prototype:	void ROSA_start(void)
Description:	Start execution of the installed TCB's.
Parameters:	None.
Return value:	Nothing.

1.2.3 ROSA_yield

Prototype:	void ROSA_yield(void)
Description:	Yield the current task execution and switch context. Save current task context. Write a new TCB into the global EXECTASK variable and continue execution in the task given by EXECTASK.
Parameters:	Nothing.
Return value:	Nothing.

1.2.4 ROSA_tcbCreate

Prototype:	void ROSA_tcbCreate(tcb *TCB, char *id, void *taskFunc, int *stack, int stackSize)
Description:	Create a TCB entry according to the given parameters.
Parameters:	

- tcb *TCB - A pointer to the TCB block to be created.
- char *id - A identification for the TCB block of length NAMESIZE (default NAMESIZE = 4)
- void *taskFunc - A pointer to the function which are to be executed by the task.
- int *stack - A pointer to the task stack area.
- int stackSize - The maximum allowed stack for this task.

Return value:	Nothing.

1.2.5 ROSA_tcbInstall

Prototype:	void ROSA_tcbInstall(tcb *TCB)
Description:	Install a TCB entry into the TCBLIST of the ROSA kernel.
Parameters:	tcb *TCB - A pointer to the TCB to install into the kernel.
Return value:	Nothing.

2 I/O Driver API

2.1 Brief API list

```
//Button
int isButton(int button_nr);

//Joystick
int isJoystickUp(void);
int isJoystickDown(void);
int isJoystickLeft(void);
int isJoystickRight(void);
int isJoystickPressed(void);

//GPIO
void gpioClear(int pinnr);
int gpioGet(int pinnr);
void gpioSet(int pinnr);
void gpioToggle(int pinnr);

//USART
void usartGetLine(volatile avr32_usart_t *, char *);
char usartGetChar(volatile avr32_usart_t *);
void usartWriteLine(volatile avr32_usart_t *, char *);
void usartWriteChar(volatile avr32_usart_t *,
     char);
void usartWriteTcb(volatile avr32_usart_t * usart,
     tcb * dbgtcb);

//Potentiometer
int potGetValue(void);
```

2.2 Detailed API list

2.2.1 isButton

Prototype:	int isButton(int button_nr)
Description:	Check if the button is pressed.
Parameters:	int button_nr - The button number, legal values are: • PUSH_BUTTON_0 • PUSH_BUTTON_1 • PUSH_BUTTON_2
Return value:	TRUE or FALSE depending on the state of the push button.

2.2.2 isJoystickUp

Prototype:	int isJoystickUp(void)
Description:	Check if the joystick is pressed up.
Parameters:	None.
Return value:	TRUE or FALSE depending on the state of the joystick.

2.2.3 isJoystickDown

Prototype:	int isJoystickDown(void)
Description:	Check if the joystick is pressed down.
Parameters:	None.
Return value:	TRUE or FALSE depending on the state of the joystick.

2.2.4 isJoystickLeft

Prototype:	int isJoystickLeft(void)
Description:	Check if the joystick is pressed left.
Parameters:	None.
Return value:	TRUE or FALSE depending on the state of the joystick.

2.2.5 isJoystickRight

Prototype:	int isJoystickRight(void)
Description:	Check if the joystick is pressed right.
Parameters:	None.
Return value:	TRUE or FALSE depending on the state of the joystick.

2.2.6 isJoystickPressed

Prototype:	int isJoystickPressed(void)
Description:	Check if the joystick is pressed/clicked down its center.
Parameters:	None.
Return value:	TRUE or FALSE depending on the state of the joystick.

2.2.7 gpioClear

Prototype:	void gpioClear(int pinnr)
Description:	Set the GPIO *'pinnr'* to 0.
Parameters:	int pinnr - The GPIO pin number.
Return value:	Nothing.

2.2.8 gpioGet

Prototype:	int gpioGet(int pinnr)
Description:	Read the value of the GPIO pin *'pinnr'*.
Parameters:	int pinnr - The GPIO pin number.
Return value:	The current value of the GPIO *'pinnr'*.

2.2.9 gpioSet

Prototype:	void gpioSet(int pinnr)
Description:	Set the GPIO *'pinnr'* to 1.
Parameters:	int pinnr - The GPIO pin number.
Return value:	Nothing.

2.2.10 gpioToggle

Prototype:	void gpioToggle(int pinnr)
Description:	Toggle the GPIO *'pinnr'*. Example: If it previously was 1, it will become 0, and vice verse.
Parameters:	int pinnr - The GPIO pin number.
Return value:	Nothing.

2.2.11 potGetValue

Prototype:	int potGetValue(void)
Description:	Get the current value of the potentiometer of the EVK1100.
Parameters:	None.
Return value:	The current value of the potentiometer.

2.2.12 usartGetLine

Prototype:	void usartGetLine(volatile avr32_usart_t * usart, char * buf)
Description:	Get a line, until a return is received, from the USART.
Parameters:	• avr32_usart_t * usart - A pointer to the USART controller. • char * buf - A pointer to the buffer to hold the input line.
Return value:	Nothing.

2.2.13 usartGetChar

Prototype:	char usartGetChar(volatile avr32_usart_t * usart)
Description:	Get a single character from the USART controller.
Parameters:	avr32_usart_t * usart - A pointer to the USART controller.
Return value:	A char from the USART controller.

2.2.14 usartWriteChar

Prototype:	void usartWriteChar(volatile avr32_usart_t * usart, char ch)
Description:	Write a single char*´ch´* to the USART controller.
Parameters:	• avr32_usart_t * usart - A pointer to the USART controller. • char ch - The character to write to the USART controller.
Return value:	Nothing.

2.2.15 usartWriteLine

Prototype:	void usartWriteLine(volatile avr32_usart_t * usart, char * string)
Description:	Write a string of characters to the USART controller.
Parameters:	• avr32_usart_t * usart - A pointer to the USART controller. • char * string - A pointer to the string to write to the USART controller.
Return value:	Nothing.

2.2.16 usartWriteTcb

Prototype:	void usartWriteTcb(volatile avr32_usart_t * usart, tcb * dbgtcb)
Description:	Write TCB debugging information to the USART controller.
Parameters:	• avr32_usart_t * usart - A pointer to the USART controller. • tcb *TCB - A pointer to the TCB to write to the USART controller.
Return value:	Nothing.

2.2.17 malloc

Prototype:	void * malloc(size_t size)
Description:	Allocate*'size'* bytes of memory from the heap.
Parameters:	size_t size - The number of bytes to allocate.
Return value:	A pointer to the allocated memory.

2.2.18 free

Prototype:	void * free(void *mem)
Description:	Free the allocated memory at the location pointed to by*´mem´*.
Parameters:	void * mem - A pointer to the allocated memory to set free.
Return value:	Nothing.