BLµE RTOS Function Guide

General Initialization

Button_Init

Function Prototype:	void Button_Init(void)
Parameters:	None
Return:	None
Purpose:	Sets up the two push buttons on the TM4C123G Board. This is a quality of life function as it is only a single line and I bypasses the problem with the TivaWare Driver Library.
Example:	Button_Init();

Led_Init

Function Prototype:	void Led_Init(void)
Parameters:	None
Return:	None
Purpose:	Used to setup the R, B, and G Led on the TM4C123G Board. This is a quality of life function as it simplifies the LED initialization process.
Example:	Button_Init();

Gen_Init

Function Prototype:	void Gen_Init(void)
Parameters:	None
Return:	None
Purpose:	Used to call Both Led_Init and Button_init. This is a quality of life function.
Example:	Gen_Init();

Systick_Init

Function Prototype:	void Systick_Init(unsigned int TicksPerSec)
Parameters:	TicksPerSec – How many System ticks there are each second. Recommended 1000.
Return:	None
Purpose:	Setups the systick subsystem for use in timing for the RTOS. TicksPerSec is used to set the reload value for the systick subsystem. **This is a required function**.
Example:	Systick_Init(1000);

InitConsole

Function Prototype:	void InitConsole(unsigned int Baud)
Parameters:	Baud – Sets the Baud Rate for the UART.
Return:	None
Purpose:	Setups the USB UART subsystem for sending data over the USB to Tera Term or Putty. This is a quality of life function.
Example:	InitConsole(9600);

Tasks

AddFunc

Function Prototype:	void AddFunc(void(*Func)(),int Prio)
Parameters:	Func – is the address of the function to be ran for the task
	Prio – is the Priority of the tasks to be ran. 0 is the highest priority
Return:	None
Purpose:	Creates/adds Tasks to the RTOS. As a note, Tasks should always loop and never terminate. Forgetting to do this will cause problems.
Example:	AddFunc(Task1, 5);

StartRTOS

Function Prototype:	void StartRTOS(void)
Parameters:	None
Return:	None
Purpose:	Starts the RTOS and gets it running. It should never be returning from here.
Example:	StartRTOS();

TaskDelay

Function Prototype:	void TaskDelay(unsigned int Delay)
Parameters:	Delay – How long the function will sleep. With a default clock of 1000, units are in ms.
Return:	None
Purpose:	Sleep a function for a desired amount of time.
Example:	TaskDelay(100);

EnterCS

Function Prototype:	void EnterCS(void)
Parameters:	None
Return:	None
Purpose:	Used to Enter Critical Sections. Not really needed for user use but can be. **MUST BE COUPLED WITH EXITCS.**
Example:	EnterCS()

ExitCS

Function Prototype:	void ExitCS(void)
Parameters:	None
Return:	None
Purpose:	Used to Exit a Critical Section. Not really need for user use but it's still there. **MUST BE COUPLED WITH ENTERCS.**
Example:	ExitCS();

Semaphores

CreateSema

Function Prototype:	void CreateSema(Sema * Semaphore, unsigned int count)
Parameters:	Semaphore – Pointer to the semaphore being used.
	Count – Initial value for the Semaphore, should be positive.
Return:	None
Purpose:	Used to initialize Semaphores. Sets the counter for the semaphore to the count value. Count value should be greater than 0.
Example:	CreateSema(&MySema, 1);

PendSema

Function Prototype:	void PendSema(Sema * Semaphore)
Parameters:	Semaphore – Pointer to the semaphore being used.
Return:	None
Purpose:	Checks if the Semaphore is available and decrements it if it is. Else the tasks blocks.
Example:	PendSema(&MySema);

AcceptSema

Function Prototype:	unsigned int AcceptSema(Sema * Semaphore);
Parameters:	Semaphore – Pointer to the semaphore being used.
Return:	Error code: 0 - No Error. 1 – Semaphore in use.
Purpose:	Tries to acquire the semaphore. Returns 1 if I can't and returns 0 if I can.
Example:	Error = AcceptSema(&MySema);

PostSema

Function Prototype:	void PostSema(Sema * Semaphore)
Parameters:	Semaphore – Pointer to the semaphore being used.
Return:	None
Purpose:	Release the semaphore be incrementing the counter. Also unblocks all Tasks that are blocking on this Semaphore.
Example:	PostSema(&MySema);

Flags

CreateFlag

Function Prototype:	void CreateFlag(Flag * Flags, unsigned int Mask)
Parameters:	Flags – Pointer to the flag being used.
	Mask – Flags bit that are to be enabled.
Return:	None
Purpose:	Initialize the Flag with the Mask. Only bits that are enabled with the Mask can be used.
Example:	CreateFlag(&MyFlag, 0x1|0x2);

PendFlag

Function Prototype:	unsigned int PendFlag(Flag * Flags unsigned int Mask, unsigned short Consume)
Parameters:	Flags – Pointer to the flag being used.
	Mask – Flags bit that are to be enabled.
	Consume – Bool for clearing Flags
Return:	Error code: 0 - No Error. 404 – Mask Flag bits are not enabled
Purpose:	Check if Mask Bits are set on the Flag. If the Flags bit of the mask are not enable will return with error 404. If not set, then take will Block. If Consume is 1 then the flags will be cleared on a successful read.
Example:	Error = PendFlag(&MyFlag, 0x2, 0x1);

AcceptFlag

Function Prototype:	unsigned int AcceptFlag(Flag * Flags unsigned int Mask, unsigned short Consume)
Parameters:	Flags – Pointer to the flag being used.
	Mask – Flags bit that are to be enabled.
	Consume - Bool for clearing Flags
Return:	Error code: 0 - No Error. 1 – Flag in use. 404 – Mask Flag bits are not enabled
Purpose:	Check if Mask Bits are set on the Flag. If the Flags bit of the mask are not enable will return with 404 error. If not set, then will return with error 1. If Consume is 1 then the flags will be cleared on a successful read.
Example:	Error = AcceptFlag(&MyFlag, 0x1, 0x0);

PostFlag

Function Prototype:	unsigned int PostFlag(Flag * Flags, unsigned int Mask)
Parameters:	Flags – Pointer to the flag being used.
	Mask – Flags bit that are to be enabled.
Return:	Error code: 0 - No Error. 404 – Mask Flag bits are not enabled
Purpose:	Sets Mask Bits of the Flag. If the Flags bit of the mask are not enable will return with 404 error. On success will unblock all tasks blocked by the flag and will return error 0.
Example:	Error = PostFlag(&MyFlag, 0x3);

Mailboxes

CreateMailbox

Function Prototype:	void CreateMailbox(Mailbox * MBox)
Parameters:	MBox – Pointer to the Mailbox being used.
Return:	None
Purpose:	Initialize the Mailbox being used.
Example:	CreateMailbox(&MyBox);

PendMailbox

Function Prototype:	void * PendMailbox(Mailbox * MBox)
Parameters:	MBox – Pointer to the Mailbox being used.
Return:	Mail that is sent as type void *.
Purpose:	Will return next message in the mailbox as a void *. If there is no mail in the box, then task will block.
Example:	Data = (int) PendMailbox(&MyBox);

AcceptMailbox

Function Prototype:	void * AcceptMailbox(Mailbox * MBox, unsigned short * error)
Parameters:	MBox – Pointer to the Mailbox being used.
	Error – Pointer to Error Variable.
Return:	Mail that is sent as type void *.
Purpose:	Will return next message in the mailbox as a void *. If there is no mail in the box, then will Error will be 404. On success Error will be 0.
Example:	Data = (char) AcceptMailbox(&MyBox, &Error);

PostMailbox

Function Prototype:	unsigned short PostMailbox(Mailbox * MBox, void * mail)
Parameters:	MBox – Pointer to the Mailbox being used.
	Mail – value being passed into the mail box. Should be cased as a void *.
Return:	Error code: 0 – no error. 404 – Mailbox is full.
Purpose:	Put mail into the mailbox. Will return error 404 if the mailbox is full. On success will return 0.
Example:	Error = PostMailbox(&MyBox, (void *)Data);