USING A PUBLIC TASK WITH VxWORKS EVENTS FOR TASK SYNCHRONIZATION

The typical use case has a public task (the "Receiver") in some memory space (RTP or the kernel) that pends on an eventReceive() call. Another task in some other memory space (the "Sender") issues an eventSend() call, thereby unblocking or synchronizing the Receiver.

TASK GUIDE

Step	Action	Example
1	Create a project for the Receiver and the Sender.	

2	For the Receiver, the first task to execute, write code that does the following:
	
	a) Creates the public task and captures its ID
	b) Loops, pending on the Event (Event No. 1 in this case) and printing a message when it's received
		The code will look something like this:
		`TASK_ID taskId;`
		`taskId=taskOpen("/publicTask",100,0, OM_CREATE, NULL,`
		`5000, 0, (FUNCPTR)eventReceiveTask,`
		`0,0,0,0,0,0,0,0,0,0);`
		`UINT32 *pEventsReceived = (UINT32 *)calloc (1,4);`
		`.`
		`.`
		`FOREVER{`
		`eventReceive(VXEV01, EVENTS_WAIT_ANY,`
		`WAIT_FOREVER, *pEventsReceived);`
		`printf ("Got the Event\n");`
		`}`

3	For the Sender, write code that does the following:
	
	a) Attaches to (gains access to) the Receiver task and captures the ID
	b) Loops, sending the Event every 5 seconds, for example
		The code will look something like this:
		`TASK_ID taskId;`
		`taskId=taskOpen("/publicTask", 0,0,0,0,0,0,0,0,0,0,`
		`0,0,0,0,0,0,0`
		`.`
		`.`
		`FOREVER{`
		`eventSend(taskId, VXEV01);`
		`taskDelay (sysClkRateGet() * 5);`
		`}`

Key Points

- The taskOpen () call is used either to create the public task initially or to gain access to it. The only difference is the OM_CREATE option.
- The basic dialog could be extended to have multiple Senders. That extension is not covered here.

education.windriver.com – training@windriver.com