Harmonic Software Systems
1 Willow Place, Rosier Business Park
Coneyhurst Road, Billingshurst
West Sussex RH14 9DE
+44(0)1403 784500, +44 (0) 774 386 3080
sales@harmonicss.co.uk

Exploring the Environment

Setting the Build Environment

Environment variables must be configured so that you can build from the workstation command line.

WIND_HOME is installation dir (*installDir*)

WIND_BASE is vxworks-7 (or vxworks-6) dir

From WIND_HOME:

Linux:

```
$ ./wrenv.sh -p vxworks-7 (or vxworks-6)
```

OR—to preserve existing bash shell:

```
$ eval `./wrenv.sh -p vxworks-7 -o print_env -f sh
```

Windows:

```
> wrenv.exe -p vxworks-7 (or vxworks-6)
```

Setting the Workspace

Linux:

```
$ export WIND_WRTOOL_WORKSPACE=installDir/workspace
```

Windows:

```
> set WIND_WRTOOL_WORKSPACE=installDir\workspace
```

Exploring VxW Source Projects (VSB)

List VSB layers:

```
> wrtool prj vsb list vsbName
```

List All CPU Architectures Supported in this VxWorks Installation:

```
> wrtool vsb listcpus
```

Get VSB Configuration:

```
> wrtool prj vsb value get vsbParam vsbName
```

Is SMP enabled?	Is RTP enabled?	Debugs enabled?	A 64-bit VSB?
vsbParam = SMP	*vsbParam* = RTP	*vsbParam* = DEBUG_FLAG	*vsbParam*=LP64

Exploring VxW Image Projects (VIP)

Which BSP + VSB is the VIP based on?

```
> wrtool prj vip info vipName
```

Which components are included in this VIP?

```
> wrtool prj vip component list vipName
```

Where are this component's source files?

```
> wrtool prj vip component info vipName componentName
```

What is the VIP's default bootline?

```
> wrtool prj vip parameter value vipName DEFAULT_BOOT_LINE
```

Creating New Projects

Create a VSB

wrtool vxprj vsb create –S –force –bsp *bspName* –lp64 –cpu *cpuName* –smp myVSB

e.g. bspName = itl_generic, cpuName = NEHALEM

Add "–debug" to enable debug symbols

Remove "-lp64" for 32-bit VSB

Add a VSB Layer

wrtool vxprj vsb add *vsbName layerName*

Build a VSB

cd myVSB; make –j 32

Create a VIP

wrtool vxprj create –smp *bspName vipName* –profile *profileName* –vsb vsbName

e.g. bspName = itl_generic, profileName = PROFILE_INTEL_GENERIC

Add "-debug" to enable debug symbols

Add a VIP Component

wrtool vxprj component add *vipName componentName*

Add a VIP Bundle

wrtool vxprj bundle add *vipName bName*

e.g. bName = BUNDLE_STANDALONE_SHELL

Which bundle is added to a VIP?

wrtool vxprj bundle list *vipName*

Create a downloadable kernel module (DKM)

prj dkm create –vsb *vsbName dkmName*

Create a real-time process (RTP)

prj rtp create –vsb *vsbName rtpName*

Add a File to a Project

prj file add *fileName prjName*

Add "-link" to link to, not copy the file

VxWorks API Calls

VxWorks Show Routines

INCLUDE_DOSFS_SHOW:

STATUS dosFsShow (void * pDevName, u_int level)

INCLUDE_POOL_SHOW:

void poolShow (POOL_ID poolId, int level)

INCLUDE_SHOW_ROUTINES:

STATUS objShowAll (OBJ_ID objId, int showType)

INCLUDE_WATCHDOGS_SHOW:

STATUS wdShow (WDOG_ID wdId)

INCLUDE_TASK_SHOW:

STATUS taskShow (TASK_ID tid, int level)

INCLUDE_SYM_TBL_SHOW:

STATUS symShow (SYMTAB_ID pSymTbl, char * substr)

INCLUDE_STDIO_SHOW:

STATUS stdioShow (FAST_FILE * fp, int level)

INCLUDE_MODULE_MANAGER:

STATUS moduleShow (char * modNameorId, int options)

INCLUDE_MEM_SHOW:

STATUS memShow (int type)

STATUS memPartShow (PART_ID partId, int type)

INCLUDE_TASK_SHOW:

void envShow (TASK_ID taskId)

INCLUDE_MSG_Q_SHOW:

STATUS msgQShow (MSG_Q_ID msgQId, int level)

INCLUDE_POSIX_TIMER_SHOW:

Int timer_show (timer_t timerId, int verbose)

INCLUDE_SEM_SHOW:

STATUS semShow (SEM_ID semid, int level)

INCLUDE_HW_FP_SHOW:

void fppCtxShow (FP_CONTEXT * pFpContext)

INCLUDE_EDR_SHOW:

STATUS memEdrPartShow (PART_ID partId)

INCLUDE_VM_SHOW:

STATUS vmContextShow (VM_CONTEXT_ID context)

INCLUDE_VXBUS_SHOW:

void vxbDevShow (VXB_DEV_ID pRoot, int toggle)

Debugging Drivers

Exploring Exceptions

INCLUDE_EDR_ERRLOG:

STATUS edrShow (int start, int count, int facility, int severity)

From the kernel shell:

-> edrShow()

Exploring Fatal Exceptions

1—Set a breakpoint in sysToMonitor().

2—When target breaks, examine the CPU link register to get the calling address.

3—Set a new breakpoint there and re-run.

4—Keep working back till you find the problem.

Deferring Driver Startup (VxBus 2 / SR0660 and later)

INCLUDE_VXBUS_DRIVER_DEFER

List the driver names to be deferred separated by a colon (:) in VIP parameter **VXBUS_DRIVER_DEFER_LIST_STR**.

Manually Deferring and Initializing a Network Driver (VxBus 2)

Convert network driver from static to dynamic driver registration. For example:

1—Locate VXB_DRV_DEF (*VXB_DRV_structure_name*) and comment line out.

2—Create a new function in the driver:

void addMyEnetDrv() {

vxbDrvAdd(&*VXB_DRV_structure_name*);

}

3—Reboot target and from the kernel shell:

-> addMyEnetDrv()

-> ipcom_drv_eth_init("*drv_name*", 0, 0)

-> ifconfig ("*drv_name*0 *ip_address* netmask *net-mask* up")

VxWorks API Calls

VxWorks Message Queue Library

INCLUDE_MSG_Q:

MSG_Q_ID msgQOpen (const char * name, size_t maxMsgs, size_t maxMsgLength, int options, int mode, void * context)

STATUS msgQClose (MSG_Q_ID msgQId)

MSG_Q_ID msgQCreate (size_t maxMsgs, size_t maxMsgLength, int options)

STATUS msgQDelete (MSG_Q_ID msgQId)

ssize_t msgQNumMsgs (MSG_Q_ID msgQId)

STATUS msgQSend (MSG_Q_ID msgQId, char * buffer, size_t nBytes, _Vx_ticks_t timeout, int priority)

ssize_t msgQReceive (MSG_Q_ID msgQId, char * buffer, size_t maxNBytes, _Vx_ticks_t timeout)

STATUS msgQInfoGet (MSG_Q_ID msgQId, MSG_Q_INFO * pInfo)

VxWorks Pipe I/O Driver

INCLUDE_PIPES:

STATUS pipeDevCreate (const char * name, size_t nMessages, size_t nBytes)

STATUS pipeDevDelete (const char * name, BOOL force)

STATUS pipeAnonCreate (size_t nMessages, size_t nBytes, unsigned flags, int * pFd)

VxWorks I/O Library

int open (const char * name, int flags, ...)

int creat (const char * name, mode_t mode)

int remove (const char * name)

ssize_t write (int fd, const void * buffer, size_t nbytes)

int close (int fd)

ssize_t read (int fd, void * buffer, size_t max-bytes)

int ioctl (int fd, int function, …)

VxWorks API Calls

void vxbDrvShow (void)

INCLUDE_PCI_SHOW:

void vxbPciCtrlShow (void)

STATUS vxbPciDeviceShow (VXB_DEV_ID busCtrlID, UINT8 busNo)

void vxbPciTopoShow (VXB_DEV_ID busCtrlID)

STATUS vxbPciFuncShow (VXB_DEV_ID busCtrlID, UINT8 bus, UINT8 device, UINT8 function)

VxWorks Logging and kprintf Library

INCLUDE_LOGGING:

int logmsg (char * fmt, _Vx_usr_arg_t arg1, _Vx_usr_arg_t arg2, _Vx_usr_arg_t arg3, _Vx_usr_arg_t arg4, _Vx_usr_arg_t arg5, _Vx_usr_arg_t arg6)

INCLUDE_DEBUG_KPRINTF:

int kprintf (const char * fmt, ...)

ssize_t kputs (char * buffer)

VxWorks Events Library

INCLUDE_VXEVENTS:

STATUS eventClear (void)

STATUS eventReceiveEx (UINT32 events, UINT32 options, _Vx_ticks_t timeout, UINT32 * pEventsReceived)

STATUS eventReceive (UINT32 events, UINT8 options, _Vx_ticks_t timeout, UINT32 * pEventsReceived)

STATUS eventSend (TASK_ID taskId, UINT32 events)

VxWorks Semaphore Library

INCLUDE_SEM_LIB:

SEM_ID semBCreate (int options, SEM_B_STATE ini-tialState)

SEM_ID semCCreate (int options, int initialCount)

SEM_ID semMCreate (int options)

SEM_ID semOpen (const char * name, SEM_TYPE type, int initState, int options, int mode, void * con-text)

STATUS semClose (SEM_ID semId)

STATUS semDelete (SEM_ID semId)

STATUS semFlush (SEM_ID semId)

STATUS semGive (SEM_ID semId)

STATUS semTake (SEM_ID semId, _Vx_ticks_t timeout)