Release Notes for the QNX Neutrino 6.4.0 BSP for Texas Instruments DRA446 EVM 1.0.0#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board version: ti dra446 evm
- ROM Monitor version UBoot v 1.1.3
- 64 MB AMD flash
- NAND flash

Host development system#

- QNX Momentics 6.4.0, one of the following host systems:
  - QNX Neutrino 6.4.0
  - Microsoft Windows Vista, XP SP2 or SP3, 2000 SP4
  - Linux Red Hat Enterprise Workstation 4 or 5, Red Hat Fedora Core 6 or 7, Ubuntu 6.0.6 LTS or 7, or SUSE 10
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- RS-232 serial port
- NULL-modem serial cable
- Ethernet link

Getting Started#

Step 1: Connect your hardware#

Connect the serial cable to the serial port of the jacinto board and to the first serial port on the host machine (e.g. ser1 on a Neutrino host).

Note:If you have a Neutrino host with a serial mouse, you may have to move the mouse to the second serial port on your host, because some terminal programs require the first serial port.

Step 2: Build the BSP#

You can build a BSP OS image from the source code or the binary components contained in a BSP package.

For instructions about building a BSP OS image, please refer to the chapter Working with a BSP in the Building Embedded Systems manual.

Step 3: Transfer the OS image to the target

Boot OS images by ROM monitor#

1. On your host machine, start your favorite terminal program with these settings:

- Baud: 115200
- Bits: 8
- Stop bits: 1
- Parity: none
- Flow control: none

2. Start your target. You should see output similar to the following:

U-Boot 1.1.3 (Jan 22 2007 - 16:13:39)

U-Boot code: 62080000 -> 620977D0 BSS: -> 6209C4C8
RAM Configuration:
Bank #0: 60000000 128 MB

********** DRI version 0.05 **********
qjwang@ti.com

ARM Clock :- 324MHz
SDRAM Clock :- 110MHz
SPANSION NOR Flash: 64 MB
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 3

Press any key to stop autobooting, and then you'll see the Jacinto prompt:

Jacinto #

On your target, type the following, filling in the appropriate IP addresses and ifs file:

Jacinto # setenv netmask 255.255.240.0
Jacinto # setenv ipaddr 10.42.101.242
Jacinto # setenv gatewayip 10.42.96.1
Jacinto # setenv serverip 10.42.98.211
Jacinto # setenv 'bootcmd tftpboot 40100000;go 40100000'
Jacinto # saveenv

Restart your target. You should see output similar to the following:

Using MAC Address 00:0E:99:02:61:53
TFTP from server 10.42.98.211; our IP address is 10.42.101.242
Filename 'ifs-jacinto.raw'.
Load address: 0x40100000
Loading: T T T T T T T ##
         ##
done
Bytes transferred = 627276 (9924c hex)
Starting application at 0x40100000 ...
Dcache: 512x32 WB
Icache: 512x32
arm926 rev 5 324MHz

System page at phys:40166000 user:fc404000 kern:fc404000
Starting next program at vfe0203a4
Welcome to QNX Neutrino 6.x on a TI Jacinto EVM
#

Boot OS images by IPL#

Note: Before burning the IPL, you might want to save the first 1 MB of flash in order to save the ROM monitor.

1. Run mkflashimage in the BSP's images directory to create a combined IPL/OS image to use when burning the flash.

 # ./mkflashimage

2. Download the ipl-ifs-jacinto.bin image to your target board, /dev/shmem/ipl-ifs-jacinto.bin.

3. Make sure the target can read the image you want to copy, and then execute the following commands in the Neutrino shell to burn the IPL-IFS images to flash:

```
# devf-generic -s0x20000000,64M
# flashctl -p/dev/fs0 -l6M -ve
# cp -V /dev/shmem/ipl-ifs-jacinto.bin /dev/fs0
```

4. Reboot the target; the IPL should come up. Either press **`s`** to transfer the OS image serially using sendnto, or if you generated ipl-ifs-jacinto.bin and copied it to /dev/fs0, press **`f`** to scan the flash for an os image.

Note: When booting serially, images larger than 6 MB won't load unless you first modify the IPL source.

Step 4: Start working with Neutrino OS#

You can test the OS simply by executing any shell builtin command or any command residing within the OS image (e.g. ls).

Once the initial image is running, you can update the OS image using the network and flash drivers. For sample command lines, please see the “Summary of driver commands” section.

Known issues for this BSP#

Wrong link definition for **/bin/sh** in build file. Currently the link definition is pointing to an invalid executable and will cause other application invoking "sh" to fail. In order to fix the problem you need to modify the build file to point to the proper "sh" application.

- Existing line : **`[type=link] /bin/sh=/proc/boot/sh`**
- Replace with : **`[type=link] /bin/sh=/proc/boot/ksh`**

Note: Please check the version of these release notes on the website for the most up-to-date information.

- If you use the SPI with DMA enabled, the ADE (and hence the MME) locks up. This conflict occurs because the prebuilt DSP/BIOS and ADE image from TI was compiled to use EDMA channel 1. (Ref# 54350)

Workaround: When you start spi-master, don't specify the **edma** option.

If you have acquired the ADE code from TI, you might be able to to resolve the conflict by modifying the EDMA channel numbers on both the DSP and ARM systems.

- Only one headphone output (p8 on the board) is currently supported.
- The graphics driver requires a TI DSP image with the TES eVRU v2.03 to achieve accelerated rendering. You should obtain this separately from this package.
- EMIF is not implemented in U-boot. This may cause the graphics to not work properly. You should use IPL booting for graphics.

- The spi-dm644x.so shared object doesn't support the SPIENA pin. The current driver supports only the first two chip selects.
- We haven't tested the ETFS driver because of a lack of flash devices.
- If you use gcc 2.95.3 to compile this BSP, you'll see this warning, which you can ignore:

/bsp-TI-Jacinto-1.0.0-20071115-binsrc/src/hardware/devg/jacinto/mode.c:62: warning: passing arg 1 of `disp_phys_addr' disca

- If you use gcc 3.3.5 or gcc 4.2.4, you'll see these warnings, which you can ignore:

/bsp-TI-Jacinto-1.0.0-20071115-binsrc/src/hardware/can/jacinto/driver.c:162: warning: comparison is always false due to limi

/bsp-TI-Jacinto-1.0.0-20071115-binsrc/src/hardware/devg/jacinto/mode.c:62: warning: passing arg 1 of `disp_phys_addr' disc

- SPI0 is inoperable because the dra446.h header file uses the incorrect base address for SPI0. (Ref# 52270)

Workaround: Edit dra446.h and change this line:

```
#define     DRA446_SPI0_BASE          0x01C41800
```

to this:

```
#define     DRA446_SPI0_BASE          0x01C4CC00
```

- In Microsoft Windows, certain programs (e.g. Norton Ghost) add directories inside double quotation marks (e.g. ...;"c:\Program Files\Norton Ghost\";...) to your PATH environment variable. This causes the Cygwin spawn() function to fail, which in turn causes cp to fail when called by ln-w. (Ref# 20046)

Workaround: Modify your PATH environment variable and remove any quotation marks.

- In those instances where the the ROM monitor's MAC address is different from the one you pass in when running io-pkt-v4, the host can cache the ROM monitor's address. This can result in a loss of connectivity.

Workaround: If you need to specify a MAC address to io-pkt-v4, we recommend that you use the same MAC address that the ROM monitor uses. This will ensure that if the host caches the ROM monitor's MAC address, you'll still be able to communicate with the target. Otherwise you might need to delete the target's arp entry on your host.

- The number of simultaneously connected USB peripherals is limited, as the host controller can only support four out standing transactions. For example, attaching a network dongle and a mass storage device will result in one or both devices not operating correctly. (Ref#61212)

- If you compile and run the DSPLink 1.40.05 v3 with this BSP in QNX6.4.0 enviornment, there will be some compile errors need to be worked around.

Workaround: Modify the following 4 files

1. Download the latest version of setenv.sh or Modify the existing setenv.sh (line 2)

Change from:
CPWD=`$QNX_HOST/usr/bin/pwd`
To:
CPWD=`$QNX_HOST/usr/bin/pwd.exe`

2. Modify the file lib/dsplink14005/src/api/nto/drv_api.c (From line 1432 to 1434)

Change from:

(SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params = save_SMAPOOL_Attrs;
(Uint32 *)((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->bufSizes = save_b
(Uint32 *)((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->numBuffers = sav
To:
((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params = save_SMAPOOL_Attrs;
((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->bufSizes = save_bufsizes;
((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->numBuffers = save_numBuf

3. Modify the file lib/dsplink14005/src/gen/coff.c (both line 930 and 1125)

Change from:
Char8 symbolFromStrTab [COFF_NAME_LEN + 1];
To:
Char8 symbolFromStrTab [COFF_NAME_LEN + 1] __attribute__((aligned(32)));

4. Modify the file services/Dsplink/resmgr.c (From line1209 to 1211)

Change from:
(SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params = smapoolattrs;
(Uint32 *)((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->bufSizes = bufsize
(Uint32 *)((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->numBuffers = num
To:
((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params = smapoolattrs;
((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->bufSizes = bufsizes;
((SMAPOOL_Attrs *)((POOL_OpenParams *)args->apiArgs.poolOpenArgs.params)->params)->numBuffers = numBufs;

Summary of driver commands#

The driver command lines below are specific to the TI Jacinto EVM board. See the online docs for each driver for additional command-line options and other details.

Note: Some of the following drivers are commented out in the default buildfile. To use the drivers in the target hardware, you'll need to uncomment them in your buildfile, rebuild the image, and load the image into the board.

Startup:#

Command:

```
startup-jacinto -L 0x67800000,0x800000 -vvvvv
```

Serial:#

Command:

```
devc-ser8250-jacinto -e -F -S -b 115200 -c24000000/16 0x01c20000^2,43
```

Required binaries:

- devc-ser8250-jacinto

Flash(NOR):#

Command:

```
devf-generic -s0x20000000,64M
```

Required binaries:

- devf-generic
- flashctl

SPI:#

Command:

```
spi-master -d dra446 base=0x01c24c00,irq=37,edmairq=0xc128,edmachannel=40,edma=1
```

Required binaries:

- spi-master

Required libraries:

- spi-dm644x.so

I2C:#

Command:

```
i2c-dm6446 -p0x01C21000 -i39 --u0
i2c-dm6446 -p0x01C21000 -i39 --u1
```

Required binaries:

- i2c-dm6446

USB:#

Command:

```
io-usb -d jacinto ioport=0x01c42000,irq=2
```

Required binaries:

- io-usb
- usb

Required libraries:

- devu-jacinto.so
- libusbdi.so

Ethernet:#

Command:

```
io-pkt-v4 -dsmc9000-jacinto ioport=0x2a000000,irq=106
```

Required binaries:

- io-pkt-v4
- ifconfig

Required libraries:

- devn-smc9000-jacinto.so
- libsocket.so

- devnp-shim.so

Block and SD:#

Command:

```
devb-eide-dm644x blk cache=12M
devb-sd-jacinto blk cache=12M
```

Required binaries:

- devb-eide-dm644x
- devb-sd-jacinto

Required libraries:

- devb-eide-dm644x
- libcam.so
- cam-disk.so
- cam-cdrom.so
- io-blk.so
- fs-qnx4.so
- fs-dos.so
- fs-cd.so

Audio:#

Command:

```
io-audio -d dra44x-jacinto
```

Note: This driver must be started after the I2C driver is active.

Required binaries:

- io-audio
- mix_ctl

Required libraries:

- deva-ctrl-dra44x-jacinto.so
- libasound.so

ETFS NAND flash#

Command when using 16-bit nand chips:

```
fs-etfs-jacinto -Dwidth=16 -m /fs/etfs
```

Command when using 8-bit nand chips:

```
fs-etfs-jacinto -Dwidth=8 -m /fs/etfs
```

You should then see /dev/etfs1 and /dev/etfs2 partitions. Required binaries:

- fs-etfs-jacinto
- etfsctl

To erase and format the NAND flash partition:

- etfsctl -d/dev/etfs2 -S -e
- etfsctl -d/dev/etfs2 -S -f -c

You should now see the mountpoint /fs/etfs/ which you can use to copy files to.

Note: You have to change the S1 config switch on the base board and start the driver with the correct command line options depending on the width of the chip.

Graphics:#

Command:

```
io-display -dvid=0,did=0
```

Required binaries:

- io-display

Required libraries:

- devg-jacinto.so
- libGLES_CL.so.1
- libffb.so

- libgf.so.1

Note:

1.The eVRU_Jacinto.out (version 2.03) DSP image must be running in order for the graphics driver to use hardware-accelerated

2. You must build the DSP image with the same configuration as DSPLink on the ARM side. Make sure you're using the correc

3. This BSP includes a sample jacinto.conf file. You may have to change the dspexe= option in the jacinto.conf file, depending

Required config files:

- display.conf
- jacinto.conf

Note: For more information about these commands, see the Neutrino Utilities Reference.

About graphics#

The graphics driver supports these display interfaces:

- VPSS (digital or analog)
- Internal Raster Controller

It's possible to do dual-display output using the VPSS analog output and the Internal Raster Controller. You can set this up in the jacinto.conf configuration file.

VPSS display interface#

The driver supports the following analog outputs:

- NTSC 720x480 interlaced output (composite, svideo, component)
- PAL 720x576 interlaced output (composite, svideo, component)

You can use jacinto.conf to configure digital output. We tested the driver with the LG Philips LB080WV3 LCD.

The VPSS controller supports the following layers or “OSD windows”:

- Video Window 0, supporting YUV422 or RGB888 formats
- Video Window 1, supporting YUV422 or RGB888 formats
- OSD Window 0, supporting RGB565 format
- OSD Window 1, supporting RGB565 format

You can control the following features for all layers:

- window start position and size
- horizontal & vertical zoom (2X, 4X)

The OSD windows can support the follow features:

- alpha blending (16 levels)
- transparency (when a pixel matches the transparent color, the pixel will be transparent and the underlying video pixels will be displayed)

You can enable or configure OSD Rectangular Cursor support using jacinto.conf.

Note: Caveats:

- The OSD window restrictions (from the TI documentation) include:
  - Video Window 1 and all OSD windows must be fully contained within Video Window 0.
  - Only one video layer can be set to RGB888 at one time.
  - Only one OSD window can be set to display 16-bit RGB data (but we've been able to run 2 RGB windows).
  - The OSD RGB565 window shouldn't overlap Video Window 1. ** Video Window 1's start X position must be offset a multiple of 16 pixels from the left edge of Video Window 0.
- Transparency of the OSD windows appears to work only with a pixel value of 0, due to an apparent HW limitation.

Raster controller#

The Raster Controller has been validated with the Sharp LQ050Q5DR01 TFT LCD. The Raster controller supports a single layer and runs only at the RGB565 pixel format.

Optional graphics driver rendering acceleration#

The graphics driver can run with SW rendering or can have accelerated 2D/3D rendering using a DSP image containing the TES eVRU graphics library.

Hardware-accelerated 2D rendering using the eVRU graphics library is supported for RGB565, RGB888, and ARGB8888 color formats. The 2D accelerated features are:

- filled rectangle
- filled rectangle with alpha map
- filled rectangle with global alpha
- blit (opaque)

- blit with alpha map
- blit with global alpha
- blit with alpha map and global value
- blit with per-pixel alpha (source surface must be ARGB8888 format)
- blit with per-pixel alpha and global value (source surface must ARGB8888 format)
- clip rectangle
- lines
- thick lines
- dashed line (backfilled only)
- anti-aliased lines
- filled polygon
- filled polygon (with anti-aliasing)
- scaled blit (opaque)
- scaled blit (with global alpha)

Note: The alpha map must be the same size as the fill/blit size.

Hardware-accelerated 3D rendering using the eVRU graphics library is limited, and is disabled by default to allow OpenGLES conformance. To enable the accelerated 3D rendering, set the sw3d option in jacinto.conf to 0 (sw3d=0). When enabled, the following features are accelerated for RGB565 and RGB888:

- flat shaded triangle
- flat shaded triangle strip
- flat shaded triangle fan
- flat shaded line
- flat shaded line strip
- flat shaded line loop
- flat shaded point
- color buffer clearing

Optional graphics memory management#

The Jacinto has two memory buses (EMIFA, EMIFB). We've found that for certain graphics operations, performance is improved if the source and destination surfaces are on separate EMIFs.

You can reserve memory from the system by using the -rBASE,SIZE option, where BASE is the physical base address of memory, and SIZE is the number of bytes to reserve.

To take advantage of this, you can specific two memory regions to devg-jacinto.so, using the following new options in jacinto.conf:

abase The physical base address of the reserved memory pool. This must match the address specified to startup with the **-r** option.

asize The size of the memory pool. This must match the size specified to startup with the **-r** option.

bbase The physical base address of reserved memory pool. This must match the address specified to startup with the **-r** option.

bsize The size of the memory pool. This must match the size specified to startup with the **-r** option.

The EMIFA base address is 0x40000000 and is 128 MB in size, and the EMIFB base address is 0x60000000 and is 128 MB in size.

In developing/testing this functionality, we used the following startup line:

startup-jacinto -r 0x60000000,0x200000 -r0x47E00000,0x200000 -L 0x67800000,0x800000 -vvvvv

The above line reserves 2 MB in EMIFA and 2 MB in EMIFB. The resulting entries in jacinto.conf are:

bbase=0x60000000,bsize=0x200000,abase=0x47e00000,asize=0x200000

The devg-jacinto.so driver creates two memory pools to allocate surfaces from while surfaces can be allocated.

By default, the driver attempts to allocate memory from POOL B (specified by **bbase** and **bsize**). If this allocation fails, the surface is allocated by the graphics framework from general system memory. It doesn't attempt to allocate the surface from the other pool.

You can select POOL A (specified by **abase** and **asize**) by passing in the GF_SURFACE_CREATE_CPU_FAST_ACCESS flag to gf_surface_create() or gf_surface_create_layer() as shown in the following examples:

```
if (gf_surface_create( &images[0], device, 100, 100, GF_FORMAT_BGR888,
      NULL, GF_SURFACE_CREATE_2D_ACCESSIBLE |
      GF_SURFACE_CREATE_CPU_FAST_ACCESS) != GF_ERR_OK) {
   return -1;
}

if (gf_surface_create_layer( &t_surface, &info->layer, 1, 0,
      info->width, info->height, info->layer_format,
      NULL, GF_SURFACE_CREATE_CPU_FAST_ACCESS) != GF_ERR_OK) {
  return 0;
}
```

If this allocation fails, the surface is allocated by the graphics framework from general system memory. It doesn't attempt to allocate the surface from the other pool.

If you don't specify memory regions to that driver, all surfaces are created by the graphics framework in general memory (whether or not memory was reserved at startup). AVME/devg coexistence

AVME/devg coexistence#

It's possible to have the TI AVME and QNX graphics coexist in the same system. To do this, the following must occur:

- The QNX graphics application must be started before AVME.
- In jacinto.conf, you must set the **avme** option to indicate which OSD windows of the VPBE will be managed by the devg driver, and which it will leave for AVME to manage.

For details on the **avme** options, see jacinto.conf.

Creating a flash partition#

1. Enter the following command to start the flash filesystem driver:

devf-generic -s0x20000000,64M

2. Erase the flash, except for the first 6 MB:

Note: Because the ROM monitor or IPL/OS image are in the first 6 megabyte of flash , you may not want to erase them. Use the **-l** (length) and **-o** (offset) options to avoid these areas.

flashctl -p/dev/fs0 -o6M -l58M -ve

3. Format the partition:

flashctl -p/dev/fs0p0 -o6M -l58M -vf

4. Slay and then restart the driver to mount the new partition:

slay devf-generic

devf-generic -s0x20000000,64M

You should now have a /fs0p0 and /fs0p1 directory automounted.