Autotalks SDK User Manual / 4.11.0

October 23, 2016

Build: rel

Contents

1 Setting up development environment on Linux

The Autotalks SDK currently supports two types of development host machine: Ubuntu/Xubuntu Linux 12.04 LTS 32-bit OS and Ubuntu/Xubuntu Linux 14.04 LTS 64-bit OS.

The host machine may be physical or virtual.

For virtual machine, VirtualBox can be obtained from: https://www.virtualbox.org/

For 64-bit OS, need to download and install additional packages.

Note that the below steps are not required when using an Autotalks-provided virtual machine.

1. Download LIBMPC2 package from: http://packages.ubuntu.com/precise/i386/libmpc2/download
2. Run the following package installation commands from command line:

```
sudo dpkg --add-architecture i386
sudo apt-get install libc6:i386 lib32z1
sudo dpkg -i libmpc2_0.9-4_i386.deb
```

3. The last command above outputs an error message. To correct it:

```
sudo apt-get install -f
```

1.1 Setting up Linux virtual machine

1.1.1 Creating the virtual machine

1. Install obtained Oracle VirtualBox.
2. Start the Oracle VirtualBox program.
3. Click “File > Import Appliance” or press *Ctrl-I*.
4. Click on the “Folder” icon on the right.
5. Browse to select the file `autotalks-vm-xubuntu-14.04-amd64.ova` and click “Next”.
6. Check the box “Reinitialize the MAC address of all network cards”.
7. Click “Import” and wait for the import process to complete. The process may take up to several minutes.

1.1.2 Configuring physical machine file sharing with virtual machine

1. Select the newly created virtual machine, right-click and select “Settings” or press *Ctrl-S*.
2. Select “Shared Folders”.
3. Press the *Insert* key or click the “+” icon on the right.
4. In the “Folder Path” box enter the name of the Windows drive that you want to access from the virtual machine. We’ll assume it’s `c:\`.
5. Check the “Auto-mount” box and click “OK”.
6. Click “OK” to save the virtual machine settings.

1.1.3 Configuring virtual machine networking

1. Select the newly created virtual machine, right-click and select “Settings” or press *Ctrl-S*.
2. Select “Network”.
3. Make sure that you are on the “Adapter 1” tab.
4. Make sure that “Enable Network Adapter” box is checked.
5. In the box labeled “Attached to”: select “Bridged Adapter”.
6. In the box labeled “Name”: select the name of your PC’s Ethernet interface card. This should be the interface that’s connected to the same Ethernet LAN as the PANGAEA4 unit.
7. Click “OK” to save the virtual machine settings.

1.1.4 Starting the virtual machine

1. Select your virtual machine and click “Start” (big green arrow icon).
2. Wait for the virtual machine desktop to appear. You may safely dismiss any pop-up windows regarding software updates.

1.1.5 Common problems

Common VirtualBox error messages and how to handle them.

1.1.5.1 Error "VT-x is disabled in the BIOS (VERR_VMX_MSR_VMXON_DISABLED)"

1. Select the newly created virtual machine, right-click and select "Settings" or press *Ctrl-S*.
2. Select "System".
3. Select the "Processor" tab.
4. Uncheck the "Enable PAE/NX" box.
5. Click "OK" to save the virtual machine settings.

1.1.5.2 Error "VT-x features locked or unavailable in MSR (VERR_VMX_MSR_LOCKED_OR_DISABLED)"

1. Select the newly created virtual machine, right-click and select "Settings" or press *Ctrl-S*.
2. Select "System".
3. Select the "Processor" tab.
4. Enter "1" in the "Processor(s)" field (or move the slider all the way to the left).
5. Click "OK" to save the virtual machine settings.

1.2 Setting up the SDK on a Linux machine

1.2.1 Installing host development tools

This step assumes that you're setting up the SDK on Ubuntu Linux or another Linux variant that supports the APT package manager. In particular, if you're using an Autotalks-provided virtual machine image then this step *should be* performed.

Open a Linux command terminal by pressing *Ctrl-Alt-T* and issue the commands:

```
sudo apt-get update
sudo apt-get install gcc make gdb
```

When prompted for a password enter `123` if you are using the Autotalks-provided virtual machine. Otherwise enter your user account password. Confirm when prompted whether you want to install the packages. Some packages may be already installed on your machine.

If you would like to use the SDK on a Linux variant that doesn't have the APT package manager, please contact Autotalks.

1.2.2 Installing GCC ARM cross-compiler

We'll assume that you're using a VirtualBox virtual machine on a Windows physical machine and you have placed the file `gcc-arm-none-eabi-4_8-2014q3-atk-1.0.0-linux.tar.xz` in `c:\autotalks`. To install the ARM cross-compiler, open a Linux command terminal and issue the commands:

```
cp /media/sf_C_DRIVE/autotalks/gcc-arm-none-eabi-4_8-2014q3-atk-1.0.0-linux.tar.xz .
tar -xf gcc-arm-none-eabi-4_8-2014q3-atk-1.0.0-linux.tar.xz
sudo mkdir -p /tools/gcc/arm
sudo mv gcc-arm-none-eabi-4_8-2014q3-atk-1.0.0 /tools/gcc/arm
rm gcc-arm-none-eabi-4_8-2014q3-atk-1.0.0-linux.tar.xz
```

If you're working on a machine where you don't have administrator privileges then you can install the cross-compiler under your home directory. In this case you will need to override the default value of the Make variable `CRATON_ARM_TOOLSET_PATH` (see `build/paths.mk` in the SDK directory).

1.2.3 Installing GCC ARC cross-compiler

Place the file `gcc-arc-elf32-4.8-2014.08-rc1-4.0.tar.xz` in `c:\autotalks`. Open a Linux command terminal and issue the commands:

```
cp /media/sf_C_DRIVE/autotalks/gcc-arc-elf32-4.8-2014.08-rc1-4.0.tar.xz .
tar -xf gcc-arc-elf32-4.8-2014.08-rc1-4.0.tar.xz
sudo mkdir -p /tools/gcc/synopsys
sudo mv gcc-arc-elf32-4.8-2014.08-rc1-4.0 /tools/gcc/synopsys
rm gcc-arc-elf32-4.8-2014.08-rc1-4.0.tar.xz
```

Note that the GCC ARC cross-compiler is only required when using the multi-core (MC) SDK.

1.2.4 Installing SDK

We'll assume that you're using a VirtualBox virtual machine on a Windows physical machine and you have placed the file `sdk-4.11.0.tar.xz` in `c:\autotalks`.

1. To unpack the SDK archive type in the Linux host terminal:

```
tar -xf /media/sf_C_DRIVE/autotalks/sdk-4.11.0.tar.xz
```

2. To develop CRATON remote applications you should build the CRATON client libraries for your host machine.

```
make -C sdk-4.11.0
```

To build CRATON remote applications for a machine other than your development machine, you need to install the relevant cross-compiler. Currently, only GNU cross-compilers are supported, targeting Linux or QNX. For example, if you'd like to use a Xilinx Zynq board as a host of remote CRATON applications and have `arm-xilinx-linux-gnueabi-gcc` in your `$PATH`, you should type:

```
make -C sdk-4.11.0 TOOLSET=arm-xilinx-linux-gnueabi-
```

(Note the trailing dash.)

Alternatively, you can also use the full path:

```
make -C sdk-4.11.0 TOOLSET=/full/path/to/bin/arm-xilinx-linux-gnueabi-
```

To build for QNX you should setup some environment variables first:

```
source /opt/qnx660/qnx660-env.sh
make -C sdk-4.11.0 TOOLSET=ntoarmv7-
```

1. Install the SDK:

```
sudo mkdir -p /tools/autotalks
sudo mv sdk-4.11.0 /tools/autotalks
```

The installation path above is a suggestion – you can choose any installation path. We'll use the installation path above for the remainder of this document.

2 Developing custom CRATON firmware

2.1 Building a CRATON firmware example

To get started with developing CRATON firmware, copy the `examples/craton-threadx` directory from the SDK to your development workspace.

Type the following commands in the Linux host terminal:

```
mkdir ~/project
cd ~/project
cp -r /tools/autotalks/sdk-4.11.0/examples/craton-threadx .
```

Open the file `craton-threadx/v2x/Makefile` with a text editor of your choice. Please note that your editor should support Unix line endings, otherwise you won't be able to edit comfortably. The editor Notepad++ is one of the Windows text editors that supports Unix line endings.

You can choose to edit the files from inside the Linux virtual machine as well, using the bundled LeafPad editor or any editor of your choice that's available for installation via the `apt-get install` command.

Inside `Makefile` find the line:

```
SDK_DIR := ../../..
```

And replace it with the line:

```
SDK_DIR := /tools/autotalks/sdk-4.11.0
```

This is the installation path of the Autotalks SDK in our example. Save the new version of `Makefile`.

In the Linux host terminal issue the commands:

```
cd ~/project/craton-threadx/v2x
make
```

This results in the following files created in the `v2x` directory:

1. `arm/obj/v2x-example.o` is the object file compiled from `v2x-example.c`.
2. `arm/obj/v2x-example.d` is a list of header files included by `v2x-example.c`. It is needed in order to decide when `obj/v2x-example.o` should be rebuilt.
3. `arm/bin/v2x-example` is the entire firmware as an ELF format executable.
4. `arm/img/v2x-example.img` is the firmware image in a format supported by the U-Boot bootloader.

To run the firmware image `v2x-example.img` on CRATON please follow the instructions given in section "Booting CRATON firmware image via TFTP".

After the firmware finishes its boot sequence you should see similar output in CRATON's serial console:

```
Example: Broadcast TX: "Example 0"
Example: Broadcast TX: "Example 1"
Example: Broadcast TX: "Example 2"
Example: Broadcast TX: "Example 3"
Example: Broadcast TX: "Example 4"
```

2.2 Building a CRATON MC firmware example

Using a MC SDK, please follow the steps in the previous section to build the firmware image.

Build output is similar to the output detailed in previous section with the following differences:

1. The folders `arc1` and `arc2` are created with output for ARC CPUs. These folders do not contain firmware image (`.img`) files.
2. The folder `arm/obj/` also contains ARM object files that contain ARC executable images as read-only data sections. Their names end in `.arc1.o` and `.arc2.o`.
3. The files whose names end in `.o.sym` are temporary files used by the build process.

2.3 Building a CRATON firmware example for a specific board

The SDK supports several types of boards. Supported board names are the ones appearing under the `board` folder.

Most SDK examples are configured in their `Makefile` to be built for atk22016 which is the PCB number of the PANGAEA4 EVK.

For example, to build a firmware image for atk22017a (PANGAEA4-S EVK), inside `Makefile` find the line:

```
BOARD := atk22016
```

And replace it with the line:

```
BOARD := atk22017a
```

2.4 Setting up TFTP server

2.4.1 Setting up TFTP server on Windows

If you prefer to run a TFTP server on a Windows machine you can use the free TFTPUtil program. It can be downloaded from http://sourceforge.net/projects/tftputil/.

Let's assume that your Windows TFTP server serves files from the directory `c:\tftpboot`. To copy `v2x-example.img` from the Autotalks Linux virtual machine to `c:\tftpboot\uImage` on the Windows host:

```
cd ~/project/craton-threadx/v2x
cp arm/img/v2x-example.img /media/sf_C_DRIVE/tftpboot/uImage
```

2.4.2 Setting up TFTP server on Linux

1. Install TFTP daemon:

   ```
   sudo apt-get install tftpd
   ```

2. Create TFTP boot image directory:

   ```
   sudo mkdir /tftpboot
   sudo chmod a+wrt /tftpboot
   ```

3. Create the file `/etc/xinetd.d/tftp` with this text inside:

```
service tftp
{
        protocol        = udp
        port            = 69
        socket_type     = dgram
        wait            = yes
        user            = nobody
        server          = /usr/sbin/in.tftpd
        server_args     = -s /tftpboot
        disable         = no
}
```

4. Apply the new configuration:

```
sudo reload xinetd
```

To create a file in `/etc` you need administrator privileges. A simple way to accomplish this is:

1. Start the `nano` editor as administrator:

```
sudo nano /etc/xinetd.d/tftp
```

2. Paste the text into the editor window.
3. Press *Ctrl-X*, press *y* at the prompt "Save modified buffer [. . .] ?", and then press *Enter* to confirm the file name.

   Note: For a TFTP server to work on a virtual machine the virtual network interface has to be in bridged mode (if you prefer to use NAT mode then tunneling for UDP port 69 must be configured).

2.5 Booting CRATON firmware image via TFTP

1. Open a serial connection to the unit (please refer to "PANGAEA4 — Autotalks V2X Communication Module User Manual", section "Connecting to Serial interface").
2. Make sure that the unit is connected to an Ethernet LAN.
3. Turn off the unit and turn it back on.
4. Enter U-Boot console by pressing any key during the 3 second countdown. You should see the U-Boot console prompt:

```
U-Boot>
```

5. We'll assume that the IP address of your TFTP server is 10.10.0.10. Make sure that the server machine is reachable by issuing the command:

```
ping 10.10.0.10
```

6. If the `ping` was successful, you should see output similar to:

```
Link: UP
Duplex: FULL
Speed 100BASE-X
Using  device
host 10.10.0.10 is alive
```

7. Configure the TFTP server's IP address:

```
setenv serverip 10.10.0.10
```

8. We'll assume that the image you'd like to boot is served by the TFTP server under the name `uImage`. Issue the following command to make the unit boot this image via TFTP every time it's powered on:

```
setenv bootcmd tftp uImage\; bootm
```

9. Save the unit's configuration and reset the unit via:

```
saveenv
reset
```

10. You should see output similar to:

```
RAM:  32 MiB
Flash: 8 MiB
In:    serial
Out:   serial
Err:   serial
Net:   Link: UP
Duplex: FULL
Speed 100BASE-X
Ethernet init done.

Hit any key to stop autoboot:  0
Link: UP
Duplex: FULL
Speed 100BASE-X
Using  device
TFTP from server 10.10.0.10; our IP address is 10.10.0.50
Filename 'uImage'.
Load address: 0x50000000
Loading: TftpRemotePort=69
#################################################################
         ##########################################
done
Bytes transferred = 552100 (86ca4 hex)
## Booting kernel from Legacy Image at 50000000 ...
   Image Name:   v2x-example
   Image Type:   ARM RTOS Kernel Image (uncompressed)
   Data Size:    552036 Bytes = 539.1 KiB
   Load Address: 50000000
   Entry Point:  50000000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK
Using sdk-4.11.0
Initializing syslog ... OK
Initializing syslog console ... OK
Initializing DMA ... OK
Initializing TCP/IP stack ... OK
Initializing syslog TCP ... OK
Initializing syslog UDP ... OK
Initializing CAN service ... OK
Initializing SHA ... OK
Initializing RNG ... OK
Initializing ECC device ... OK
Initializing ECC service ... OK
```

```
Initializing SPI ... OK
Initializing V2X stack ... OK
Initializing filesystem ... OK
Initializing SNMP agent ... OK
Initializing NAV service ... OK
Initializing VTP daemon ... OK
Initializing VCA services ... OK
Initializing CLI ... OK
ATE - Auto-talks test environment
ate>
```

Note: If the TFTP server's IP address changes then `serverip` variable must also be changed so that the unit can complete the boot sequence.

2.6 Understanding panic messages on the target machine

Upon detection of an internal fatal error, the panic handler prints the contents of the last running thread's stack to the console.

For example:

```
************ PANIC ARM ************
Fault type: Data abort
Fault reason: Permission
Faulted operation: write 00000000
Last thread: (508e7018) "init_thread"
PC : 500009fc   LR : 500007c9   SP : 508eb410   IP : 00000000
FP : 00000000   r10: 508e751c   r9 : 00000000   r8 : 00000000
r7 : 00000000   r6 : 00000000   r5 : 00000001   r4 : 50815ef8
r3 : 00000000   r2 : 00000000   r1 : 00000001   r0 : 50815efc
IRQ: on         Thumb: on
Last thread's stack dump follows ...
508e70d8 00000000 00000000 00000000 00000000 508e751c 00000000 5005722d
5082c400 508eb440 00000000 00000000 00000000 00000000 500795f0 00000000
50901204 00000000 00000000 500007c9 5080ca10 00008000 0000000f 0000000f
00000000 00000001 000004e8 00000000 00000000 50bfc828 02925692 0000c600
00009605 000008f8 00000005 5083c3d0 00000092 00000056 00000092 00000000
00000000 000000c6 00000000 00000000 00000000 00000005 00000000 00000000
50041da1 508e7018 00000000 50045ea9 00000000 00000000 00000000 50046c44
00000000 efefefef
```

To convert the stack hex-dump to a stack trace please do the following:

1. Copy-paste the hex-dump to a file (everything after "..."); let's call it `stack.txt`.
2. Assuming the panic was triggered when running `v2x-example`, issue the command:

```
cd ~/project/craton-threadx/v2x
/tools/gcc/arm/gcc-arm-none-eabi-4_8-2014q3-atk-1.0.0/bin/arm-none-eabi-addr2line \
-e arm/bin/v2x-example @stack.txt | grep -v :0
```

3. You should see several "file:line-number" lines that in most cases represent the call sites of each function that was called by the last thread. Note that when using this technique, a function pointer stored on the stack will create a false appearance of a called function.
4. You can also use the `-f` flag of `addr2line` to display function/data symbol names (use `-C` to de-mangle C++ function names if you have any).

Similarly, for converting a stack hex-dump on `arc1` or `arc2` to a stack trace (when using MC SDK), please use the following addr2line tool:

```
/tools/gcc/synopsys/gcc-arc-elf32-4.8-2014.08-rc1-4.0/bin/arc-elf32-addr2line
```

2.7 Using the system logger

Messages sent to the system logger can be directed to one or more of the following "sinks":

1. Console (i.e. UART).
2. TCP port 57.
3. UDP port 514.

The U-Boot environment variable `syslog_sink` controls which sinks are enabled. Its value is a decimal number which represents a bit-mask of:

1. Console: 0x2
2. TCP: 0x4
3. UDP: 0x8

For example, to enable syslog sinks UDP and Console:

1. Turn off the unit and turn it back on.
2. Enter U-Boot console by pressing any key during the 3 second countdown. You should see the U-Boot console prompt:

   ```
   U-Boot>
   ```

3. Issue the commands:

   ```
   setenv syslog_sink 10
   saveenv
   ```

The U-Boot environment variable `syslog_level` controls message level filtering. Its value is a decimal number identical to the values used in `syslog_level_t` (see `include/craton/syslog.h`).

For example, to log messages with level `LOG_WARNING` and above (i.e. to filter messages with level lower than `LOG_WARNING`):

1. Enter U-Boot console.
2. Issue the commands:

   ```
   setenv syslog_level 4
   saveenv
   ```

When using a UDP sink, the (optional) U-Boot environment variable `syslog_server_ip` can be used to set the server's IP.

For example, to direct syslog messages to IP address 10.10.0.10 over UDP:

1. Enter U-Boot console.
2. Issue the commands:

   ```
   setenv syslog_server_ip 10.10.0.10
   saveenv
   ```

2.8 Using ThreadX Execution Profile Kit (EPK)

ThreadX EPK is a handy tool to measure CPU load caused by each thread executing on a given processor. Please see API header `<tx_execution_profile.h>`.

In previous SDKs users were required to set the boot parameter `wd_enable` to 0 in order to use the EPK on ARC CPUs. This is no longer required.

2.9 Flashing CRATON firmware image

1. Open a serial connection to the unit (please refer to “PANGAEA4 — Autotalks V2X Communication Module User Manual”, section “Connecting to Serial interface”).
2. Make sure that the unit is connected to an Ethernet LAN.
3. Turn off the unit and turn it back on.
4. Enter U-Boot console by pressing any key during the 3 second countdown. You should see the U-Boot console prompt:

   ```
   U-Boot>
   ```

5. We’ll assume that the IP address of your TFTP server is 10.10.0.10. Make sure that the server machine is reachable by issuing the command:

   ```
   ping 10.10.0.10
   ```

6. If the `ping` was successful proceed to configure the TFTP server’s IP address:

   ```
   setenv serverip 10.10.0.10
   ```

7. We’ll assume that the image you’d like to boot is served by the TFTP server under the name `uImage`. Issue the following command to load the image via TFTP:

   ```
   tftp uImage
   ```

8. Erasing flash memory is done differently depending on flash size. When using a 8 MB flash device:

   ```
   protect off 80000 5fffff
   erase 80000 5fffff
   ```

9. When using a 32 MB flash device:

   ```
   mw.l 0x41016850 0x001f0000
   protect off 80000 1bffffff
   erase 80000 1bffffff
   ```

10. On board types atk22031b, atk22031c and atk22031d:

    ```
    mw.l 0x41016850 0x001f0000
    protect off 80000 dfffff
    erase 80000 dfffff
    ```

11. Replace existing firmware image with the new one:

    ```
    cp.b ${fileaddr} 80000 ${filesize}
    setenv bootcmd 'cp.b 80000 50000000 ${filesize}; bootm'
    saveenv
    ```

12. You should see output similar to:

```
U-Boot> tftp uImage
Link: UP
Duplex: FULL
Speed 100BASE-X
Using  device
TFTP from server 10.10.0.10; our IP address is 10.10.0.50
Filename 'uImage'.
Load address: 0x50000000
Loading: TftpRemotePort=69
#################################################################
         ########################################
done
Bytes transferred = 552100 (86ca4 hex)
U-Boot> protect off 80000 1bfffff
..................................................................... done
Un-Protected 120 sectors
U-Boot> erase 80000 1bfffff
..................................................................... done
Erased 120 sectors
U-Boot> cp.b ${fileaddr} 80000 ${filesize}
Copy to Flash... done
U-Boot> setenv bootcmd 'cp.b 80000 50000000 ${filesize}; bootm'
U-Boot> saveenv
Saving Environment to Flash...
.. done
Un-Protected 2 sectors
Erasing Flash...
.. done
Erased 2 sectors
Writing to Flash... done
.. done
Protected 2 sectors
Done
```

13. Reset the unit via the command:

```
reset
```

2.10 Using custom build automation scripts

If you wish to build CRATON firmware using build automation scripts that don't use the SDK's `build/craton.mk`, then it's highly recommended to use the relevant `compiler.flags` file when calling either the ARM or ARC compilers. In each SDK release these files contain the mandatory compiler flags needed to make sure that your compiled object files are binary-compatible with the object and library files provided by the SDK. They may change from one SDK release to the next.

To pass the contents of a file to GCC as a list of command-line arguments you can use the @file syntax. Please refer to GCC manual regarding this.

3 Developing remote CRATON applications

3.1 Building a remote CRATON application example

Copy the `examples/remote-posix` directory from the SDK to your development workspace.

Type the following commands in the Linux host terminal:

```
cd ~/project
cp -r /tools/autotalks/sdk-4.11.0/examples/remote-posix .
```

Open the file `remote-posix/v2x/Makefile` with a text editor of your choice. Please note that your editor should support Unix line endings, otherwise you won't be able to edit comfortably.

Inside `Makefile` find the line:

```
SDK_DIR := ../../..
```

And replace it with the line:

```
SDK_DIR := /tools/autotalks/sdk-4.11.0
```

This is the installation path of the Autotalks SDK in our example.

> Note: If you have installed the Autotalks SDK in your home directory please be aware that `~` is not expanded inside the Makefile like on the command-line. Use `$(HOME)` instead of `~`.

Save the new version of `Makefile`.

In the Linux host terminal issue the commands:

```
cd ~/project/remote-posix/v2x
make
```

This results in the following files created in the `v2x` directory, assuming that you are on a 64-bit x86 Linux machine:

1. `x86_64-linux-gnu/obj/v2x-example.o` is the object file compiled from `v2x-example.c`.
2. `x86_64-linux-gnu/bin/v2x-example` is the executable file.

To build the examples for a machine other than your development machine, you need to install the relevant cross-compiler. For example, if you'd like to use a Xilinx Zynq have `arm-xilinx-linux-gnueabi-gcc` in your `$PATH`, instead of `make` you should type:

```
make TOOLSET=arm-xilinx-linux-gnueabi-
```

(Note the trailing dash.)

This results in the following files created in the `v2x` directory:

1. `arm-xilinx-linux-gnueabi/obj/v2x-example.o` is the object file compiled from `v2x-example.c`.
2. `arm-xilinx-linux-gnueabi/bin/v2x-example` is the executable file.

3.2 Checking network connectivity to PANGAEA4

In order to control your PANGAEA4 units via TCP/IP you should set your units' network configuration (unit IP address, network mask and default gateway IP address). Please refer to "PANGAEA4 — Autotalks V2X Communication Module User Manual" for specific instructions.

For the sake of example we'll assume that the IP address of the unit you're working with is `10.10.0.50`.

To check that you have IP connectivity to the unit please go to the Linux host terminal and issue the command:

```
ping 10.10.0.50
```

You can terminate the command at any time by pressing *Ctrl-C*. If the `ping` command reports 100% packet loss or another type of error then please:

- Check that your PANGAEA4 unit's IP address, network mask and default gateway IP address are configured and match your network configuration.
- Check that your physical machine has IP network connectivity.
- Check that you can ping PANGAEA4 unit from your physical machine.
- Check that VirtualBox is not prevented from transmitting or receiving IP network traffic by a firewall that's installed on your physical machine.

3.3 Running programs on the host machine

We'll assume that the connectivity test in the previous section was successful. Again, for the sake of example we'll assume that the IP address of the PANGAEA4 unit is `10.10.0.50`.

To run the `v2x-example` application on the Linux host machine, type:

```
cd remote-posix/v2x
./host/bin/v2x-example 10.10.0.50
```

The application will connect to the unit using V2X Tunneling Protocol (VTP) and request it to transmit V2X frames.

We'll assume you have another PANGAEA4 unit with IP address `10.10.0.51` and that it's tuned to receive data from the first unit. The example application requires that DSRC Antenna A of the second unit can receive frames transmitted via DSRC Antenna A of the first unit.

Open another Linux terminal window or tab, make sure that the working directory is the same as in the 1st terminal, and type:

```
./host/bin/v2x-example 10.10.0.51
```

You should expect output similar to:

```
Example: Broadcast TX: "Example 0"
Example: RX from 92:56:92:01:00:be
* Message: "Example 55"
* RX power: -45.50 dBm
Example: Broadcast TX: "Example 1"
Example: RX from 92:56:92:01:00:be
* Message: "Example 56"
* RX power: -45.38 dBm
Example: RX from 92:56:92:01:00:be
* Message: "Example 57"
* RX power: -45.50 dBm
Example: Broadcast TX: "Example 2"
Example: Broadcast TX: "Example 3"
Example: RX from 92:56:92:01:00:be
* Message: "Example 58"
* RX power: -45.38 dBm
```

To terminate the example application press *Ctrl-C*.

3.4 Debugging programs on the host machine

Debugging programs on the host machine is possible using the GDB debugger.

To start debugging `v2x-example` on the host machine, type:

```
gdb --args host/bin/v2x-example 10.10.0.50
```

A short GDB command summary:

- Set breakpoint: `b`
- Resume program execution: `r`
- Execute only the next line of code: `n`
- Inspect the function call stack: `bt`
- Inspect the content of variables: `p`
- Get help on any command by typing `help` *command*.
- Pressing the *Enter* key at an empty GDB prompt will repeat the previous command.

The full manual for GDB can be found online at https://sourceware.org/gdb/current/onlinedocs/gdb/.

4 Frequently Asked Questions

4.1 How is time updated on PANGAEA4 units?

CRATON MAC HW layer includes a 64bit counter (per MAC) called the Timing Synchronization Function (TSF) counter. During system initialization this counter starts counting from 2004-01-01 midnight UTC; the counting is in the TAI timescale (i.e. UTC leap seconds are counted) in microseconds resolution.

TSF update and synchronization is done based on PPS signal received from an external GNSS receiver. Initial update is done when the GNSS receiver acquires “lock” status. From this point, MAC HW realigns the counter on each incoming PPS pulse, hence canceling any drift and producing the most accurate clock in the system.

4.2 How does `gettimeofday` receive its value?

When `gettimeofday` is called, the TSF counter is sampled and the value is adjusted to UTC (UTC leap seconds are not counted).

The behavior of `gettimeofday` is slightly different on `arc2` when running in a multi-core environment. Due to inaccessibility of MAC HW registers from `arc2` (this limitation is by design), `gettimeofday` uses a timer which is synced with the TSF counter periodically.

4.3 How to determine flash size?

1. Please enter U-Boot console and issue the command:

   ```
   flinfo
   ```

2. Flash size is reported. Example output for a 8 MB flash device:

   ```
   U-Boot> flinfo
   Bank # 1: CFI conformant flash (16 x 16)  Size: 8 MB in 135 Sectors
   ```

4.4 How to enable file system on flash?

As of sdk-4.3 file system on flash is supported on all flash sizes and is enabled by default.

Users migrating from sdk-4.2 might be required to erase their entire flash (or alternatively - disable file system on flash). Please see next questions on how to do so.

An inidication that this is required is seeing traces of the following form during system init:

```
E (safeFAT.c) mkdir_: f_mkdir error 1
W (rlog_writer.c) file_tree_walk: Failed readdir_r
E (rlog_writer.c) rlog_writer_open: Failed file_tree_walk (-1)
```

4.5 How to disable file system on flash?

1. Please enter U-Boot console and issue the command:

   ```
   setenv flash_fs_enable 0
   ```

2. Save the unit's configuration and reset the unit via:

   ```
   saveenv
   reset
   ```

4.6 How to erase entire flash (excluding U-Boot and U-Boot env)?

1. Please enter U-Boot console.
2. Erasing flash memory is done differently depending on flash size. When using a 8 MB flash device please issue the commands:

   ```
   protect off 80000 7fffff
   erase 80000 7fffff
   ```

3. When using a 32 MB flash device please issue the commands:

   ```
   mw.l 0x41016850 0x001f0000
   protect off 80000 1ffffff
   erase 80000 1ffffff
   ```

4.7 How to know which version of cross-toolchains to use?

The paths to ARM and ARC cross-toolchains are in `build/paths.mk` under `CRATON_ARM_TOOLSET_PATH` and `CRATON_ARC_TOOLSET_PATH` respectively.

5 Disclaimer

Autotalks reserves the right to make changes to information published in this document, at any time and without notice.

The information in this document is believed to be accurate and reliable. However, Autotalks does not give any representations or warranties as to the accuracy of information and shall have no liability for the consequences of use of such information.