Document information

<table>
<thead>
<tr>
<th>Information</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>i.MX 8M device family, Arm Cortex-A53 processor (Armv8-A architecture), RTOS, Linux, hardware partitioning, Jailhouse hypervisor, NXP Linux Yocto, Zephyr RTOS, MCUpresso SDK</td>
</tr>
<tr>
<td>Abstract</td>
<td>This document presents the Harpoon release (EAR) 2.1 for i.MX 8M device family, using the Arm Cortex-A53 processor (Armv8-A architecture).</td>
</tr>
</tbody>
</table>
1 Overview

This document presents the Harpoon release (EAR) 2.1 for i.MX 8M device family, using the Arm Cortex-A53 processor (Armv8-A architecture).

Harpoon provides an environment for developing real-time demanding applications on an RTOS running on one (or several) Cortex-A core(s) in parallel of a Linux distribution, leveraging the 64-bit Arm architecture for higher performance.

The system starts on Linux and the Jailhouse hypervisor partitions the hardware to run both Linux and the guest RTOS in parallel.

The hardware partitioning is configurable and depends on the use case. This release includes an audio application, an industrial application and a real-time latency measurement application, all available both for FreeRTOS as well as Zephyr (some application feature limitations exist depending on the selected platform and RTOS).

This release supports the following software and hardware:

- **NXP Linux Yocto**
  - i.MX LF 5.15.32_2.0.0: For more information, see [i.MX Yocto Project User's Guide](#).
  - Real-time Edge Rev. 2.3: For more information, see [Real-time Edge Yocto Project User Guide](#).
- **i.MX 8M Series**
  - i.MX 8M Mini LPDDR4 EVKB
  - i.MX 8M Nano LPDDR4 EVK
  - i.MX 8M Plus LPDDR4 EVK
- **Jailhouse hypervisor**
- **FreeRTOS V10.4.3 kernel**
  - Cortex-A53 port, uniprocessor
  - Guest OS running on Jailhouse cell
- **Zephyr RTOS 3.0.0**
  - Cortex-A53 port, SMP
  - Guest OS running on Jailhouse cell
- **MCUXpresso SDK 2.11**
  - GIC, Timer and MMU Cortex-A53 drivers
  - CAN, ENET, ENET_QOS, GPT, I2C, SAI, and UART SoC drivers
  - Audio Codec drivers
  - Phy drivers
- **RTOS Applications**
  - Audio reference application
  - Industrial reference application
  - Real-time latency measurement application

1.1 Architecture

The following figure shows the architecture of the Harpoon solution.
The i.MX 8M box shows the hardware partitioning between Jailhouse cells. The boxes in dark orange (group 1) show the main hardware blocks allocated to the Linux OS. The boxes in blue (group 3) show the main hardware blocks allocated to the RTOS. The boxes in light orange (group 2) show the main hardware blocks shared between Linux and the RTOS. Harpoon-apps is the real-time application running on Jailhouse's inmate cell. It is built on top of the FreeRTOS and MCUXpresso drivers.

1.2 Hardware resource partitioning

Jailhouse hypervisor is used to run an RTOS in parallel with Linux: FreeRTOS and Zephyr are supported in this release.

Jailhouse is a simple hypervisor that assigns hardware resources to a guest OS instead of virtualising them. For instance, a CPU core is statically assigned to a specific guest and is not shared with other guests.
In Jailhouse terms, the RTOS (inmate) runs in a cell. A configuration file describes which hardware resources are assigned to this cell. This configuration file contains descriptions of the following:

- CPU cores assigned to the cell
- Interrupt lines assigned to the cell
- Memory regions assigned to the cell
- Virtual PCI devices used for communication between cells

There is also a root cell configuration that describes the hardware prior to the hardware partitioning.

The source files of the cell configurations are embedded through patches in the Jailhouse recipe of the Harpoon meta-layer, at the following locations:

- configs/arm64/imx8m*-freertos.c for the FreeRTOS rt_latency use case’s cell configuration
- configs/arm64/imx8m*-zephyr.c for the Zephyr rt_latency use case’s cell configuration
- configs/arm64/imx8m*-freertos-audio.c for the FreeRTOS audio use case’s cell configuration
- configs/arm64/imx8m*-zephyr-audio.c for the Zephyr audio use case’s cell configuration
- configs/arm64/imx8m*-freertos-industrial.c for the FreeRTOS industrial use case’s cell configuration
- configs/arm64/imx8m*-zephyr-industrial.c for the Zephyr industrial use case’s cell configuration
- configs/arm64/imx8m*.c for the root cell configuration

The CPU core allocated to the RTOS forms a bitmap in the cpu structure. Here, CPU core 3 is assigned to the cell:

```
.cpus = { 
  0b1000,
},
```

For a multicore (SMP) cell, two cores can be used, for instance:

```
.cpus = { 
  0b1100,
},
```

Memory regions assigned to the inmate cell are listed in the mem_regions structure. Memory regions can be reserved for the inmate cell or shared with the Linux root cell.

Memory regions can be DDR chunks for the inmate cell use as well as device memory mapped regions such as UART or SAI.

Interrupts are mapped to the cell with the irqchips structure.

Virtual PCI devices are defined with the pci_devices structure. These virtual devices are used by Jailhouse to implement IVSHMEM v2 communication channels.
2 Building Harpoon Yocto images

As mentioned in the overview section, Harpoon is compatible with both i.MX Yocto and Real-time Edge Yocto. Each distribution is addressed in a separate section below.

2.1 i.MX Yocto

To build this release, fetch its Yocto manifest and get the meta-layers:

```
$ mkdir yocto
$ cd yocto
$ repo init -u git://source.codeaurora.org/external/imx/imx-manifest.git -b imx-linux-kirkstone -m imx-5.15.32-2.0.0_harpoon-v2.xml
$ repo sync
```

Then, prepare the environment with the following command:

```
$ DISTRO=fsl-imx-xwayland MACHINE=<machine> source imx-harpoon-setup-release.sh -b build.<machine>
```

Where, `<machine>` is one of the following:

- `imx8mm-lpddr4-evk` for i.MX 8M Mini EVKB board
- `imx8mn-lpddr4-evk` for i.MX 8M Nano EVKB board
- `imx8mp-lpddr4-evk` for i.MX 8M Plus EVK board

The end user license agreement must be accepted to continue.

Then build the image with the following command:

```
$ bitbake imx-image-core
```

The image is then available in subdirectory `tmp/deploy/images/<machine>/`.

Copy the disk image to a micro-SD card. For example, assuming the card is recognized as `/dev/mmcblk0` by your host machine:

```
$ bzip2 -d -c imx-image-core-<machine>.wic.bz2 | sudo dd of=/dev/mmcblk0 bs=1M
```

The micro-SD card now contains the release.

2.2 Real-time Edge Yocto

Please refer to the [Real-time Edge Yocto Project User Guide](#) to build Harpoon and prepare an SD card for supported boards.

3 Hardware Setup

3.1 i.MX Reference Boards

This Harpoon release supports the following development boards.
Note: For more information to order the board, see [https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/evaluation-kit-for-the-i-mx-8m-mini-applications-processor:8MINILPD4-EVK](https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/evaluation-kit-for-the-i-mx-8m-mini-applications-processor:8MINILPD4-EVK)

Note: For more information to order the board, see [https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/evaluation-kit-for-the-i-mx-8m-nano-applications-processor:8MNANOD4-EVK](https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/evaluation-kit-for-the-i-mx-8m-nano-applications-processor:8MNANOD4-EVK)
3.2 Audio use case hardware

Harpoon's audio application uses the I2S HiFiBerry audio card *DAC+ ADC Pro*.

![HiFiBerry DAC+ ADC Pro](image)

**Note:** For more information to order the board, see [https://www.hifiberry.com/shop/boards/hifiberry-dac-adc-pro/](https://www.hifiberry.com/shop/boards/hifiberry-dac-adc-pro/).

The HiFiBerry DAC+ ADC Pro is an audio card designed for the Raspberry Pi, but it can be connected to EVK boards using the 40-pin connector, provided a few adaptations are made.

The following pins on the EVK's 40-pin connector must be connected to the following HiFiBerry's pins.

<table>
<thead>
<tr>
<th>EVK</th>
<th>HiFiBerry</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>5V</td>
</tr>
</tbody>
</table>

**Table 1. EVK - HiFiBerry transposition**
Table 1. EVK - HiFiBerry transposition...continued

<table>
<thead>
<tr>
<th>EVK</th>
<th>HiFiBerry</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>I2C SDA</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>I2C SCK</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>35</td>
<td>40</td>
<td>I2S TX</td>
</tr>
<tr>
<td>36</td>
<td>12</td>
<td>I2S clock</td>
</tr>
<tr>
<td>37</td>
<td>35</td>
<td>I2S word select for RX and TX</td>
</tr>
<tr>
<td>38</td>
<td>38</td>
<td>I2S RX</td>
</tr>
</tbody>
</table>

Figure 6. Handmade transposer

A complete setup, with a handmade transposer to respect above pinout, is shown as follows.
The audio card has both an ADC (PCM1863) to record audio and a DAC (PCM5122) for audio playback.

Record is done through the audio jack (connector highlighted in 1 in the following figure) and playback is done through the RCA connectors (highlighted in 2).

---

**Note:** For more information to order the board, see [https://www.hifiberry.com/shop/boards/hifiberry-dac-adc-pro/](https://www.hifiberry.com/shop/boards/hifiberry-dac-adc-pro/).

Control of the PCM1863 is done through I2C3, at address 0x4a. Control of the PCM5122 is done through I2C3, at address 0x4d.
Both the PCM1863 and PCM5122 use i.MX I2S5. The I2S5 is the I2S clock master. Two oscillators (one for sampling frequencies multiple of 44,100 Hz, one for sampling frequencies multiple of 48,000 Hz) are present on the HiFiBerry card, and controlled by PCM5122 GPIOs.

The following diagram shows the HiFiBerry architecture.

![HiFiBerry architecture diagram](image)

The PCM1863 and the PCM5122 use the same signal for I2S word select by using SAI synchronous mode.

### 3.3 Industrial use case hardware

Harpoon's industrial application may use the following hardware depending on the use case.
Figure 10. LS1028A AVB/TSN network bridge

Note: For more information to order the board, see https://www.nxp.com/design/qorIQ-developer-resources/layerscape-ls1028a-reference-design-board/LS1028ARDB.

The LS1028A RDB is used as a TSN bridge/switch in a TSN network to demonstrate the TSN ethernet use case running from the inmate cell.

Figure 11. RT1170 TSN endpoint
4 Running Harpoon Reference Applications

4.1 Basic setup

The EVK boards expose serial ports through their USB debug interface. One of these serial ports is used by Linux for its console, and another one is used by the guest RTOS.

To run the reference applications, open both serial ports with terminal emulators, insert the micro-SD card on which the Yocto image has been flashed in the EVK and power up the board.

4.2 Starting Linux kernel

Linux kernel must be started with a (Harpoon specific) Jailhouse compatible device tree. To do this, when U-Boot is executing, stop at U-Boot prompt with a terminal emulator connected to the serial port and execute the following command (based on the board and the application):

- For i.MX 8M Mini (audio or rt_latency):
  
  ```
  u-boot => setenv jh_root_dtb imx8mm-evk-harpoon.dtb
  u-boot => run jh_mmcboot
  ```

- For i.MX 8M Mini (industrial or rt_latency):
  
  ```
  u-boot => setenv jh_root_dtb imx8mm-evk-harpoon-industrial.dtb
  u-boot => run jh_mmcboot
  ```

- For i.MX 8M Nano (audio or rt_latency):
  
  ```
  u-boot => setenv jh_root_dtb imx8mn-evk-harpoon.dtb
  u-boot => run jh_mmcboot
  ```

- For i.MX 8M Nano (industrial or rt_latency):
  
  ```
  u-boot => setenv jh_root_dtb imx8mn-evk-harpoon-industrial.dtb
  u-boot => run jh_mmcboot
  ```

- For i.MX 8M Plus (audio or rt_latency):
  
  ```
  u-boot => setenv jh_root_dtb imx8mp-evk-harpoon.dtb
  u-boot => run jh_mmcboot
  ```

- For i.MX 8M Plus (industrial or rt_latency):
  
  ```
  u-boot => setenv jh_root_dtb imx8mp-evk-harpoon-industrial.dtb
  u-boot => run jh_mmcboot
  ```

**Note:** This configuration is not persistent after a reboot.
To make changes permanent, execute the following commands once (after `setenv` above):

```
    u-boot => setenv bootcmd 'run jh_mmcboot'
    u-boot => saveenv
```

Now, at each reboot, the system starts with the Jailhouse compatible configuration and no user interaction is required.

4.3 Audio application

4.3.1 Features of the audio application

The audio application is available in the harpoon share directory of the target's root file system:

```
    /usr/share/harpoon/inmates/freertos/audio.bin  # FreeRTOS binary
    /usr/share/harpoon/inmates/zephyr/audio.bin    # Zephyr binary
```

This application contains several running modes that can be started and stopped via a user space application running on Linux called `harpoon_ctrl`.

The different modes are:

- DTMF playback: plays a DTMF sequence.
- Sine wave playback: plays a generated sine wave.
- Loopback: record sound from HiFiBerry's input and play it live through HiFiBerry's output.
- Full Audio pipeline: implements a flexible 3-stage pipeline with different sources (DTMF, sine waves, SAI input) that can be routed to different sinks (SAI outputs).

All the modes support:

- Basic pipeline framework for audio processing
- 44100, 48000, 88200, 176400, 96000, and 192000 Hz sample frequency
- Audio processing period with 2, 4, 8, 16, or 32 frames
- Audio processing in 64bit float format
- Audio playback to both SAI3 (on board codec/sound jack) and SAI5 (HiFiBerry)
- Audio capture from SAI5 (HiFiBerry)

**Note:** Playback on SAI3: The i.MX 8M Plus EVK on board CODEC (WM8960) supports sample rates up to 48 kHz only. 88.2 kHz and above frequency settings will fail for this CODEC.

4.3.2 Starting the audio application with Jailhouse

The harpoon service uses the `/etc/harpoon/harpoon.conf` configuration file that contains the RTOS and the application to run. By default, the configuration file points to the FreeRTOS audio application. To use the Zephyr audio application, the following command can be run to generate an appropriate configuration file:

```
    # harpoon_set_configuration.sh zephyr audio
```
Note: Avoid changing the configuration while the Harpoon service is running (silent failure when restarting the service).

To use the audio application, Jailhouse must be started first. To start Jailhouse and the audio application, run the harpoon service with systemd:

```
# systemctl start harpoon
```

Once the harpoon service has been started, harpoon_ctrl is used to start or stop the audio modes with optional parameters. The different options for the audio application are:

<table>
<thead>
<tr>
<th>Audio options:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-f &lt;frequency&gt;</td>
<td>audio clock frequency (in Hz)</td>
</tr>
<tr>
<td></td>
<td>Supporting 44100, 48000, 88200, 176400, 96000, 192000 Hz</td>
</tr>
<tr>
<td>-p &lt;frames&gt;</td>
<td>audio processing period (in frames)</td>
</tr>
<tr>
<td></td>
<td>Supporting 2, 4, 8, 16, 32 frames</td>
</tr>
<tr>
<td>-r &lt;id&gt;</td>
<td>run audio mode id:</td>
</tr>
<tr>
<td></td>
<td>0 - dtmf playback</td>
</tr>
<tr>
<td></td>
<td>1 - sine wave playback</td>
</tr>
<tr>
<td></td>
<td>2 - playback &amp; recording (loopback)</td>
</tr>
<tr>
<td></td>
<td>3 - audio pipeline</td>
</tr>
<tr>
<td>-s</td>
<td>stop running audio mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Audio pipeline options:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-a &lt;pipeline_id&gt;</td>
<td>audio pipeline id (default 0)</td>
</tr>
<tr>
<td>-d</td>
<td>audio pipeline dump</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Audio element options:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-a &lt;pipeline_id&gt;</td>
<td>audio pipeline id (default 0)</td>
</tr>
<tr>
<td>-d</td>
<td>audio element dump</td>
</tr>
<tr>
<td>-e &lt;element_id&gt;</td>
<td>audio element id (default 0)</td>
</tr>
<tr>
<td>-t &lt;element_type&gt;</td>
<td>audio element type (default 0):</td>
</tr>
<tr>
<td>0 - dtmf source</td>
<td></td>
</tr>
<tr>
<td>1 - routing</td>
<td></td>
</tr>
<tr>
<td>2 - sai sink</td>
<td></td>
</tr>
<tr>
<td>3 - sai source</td>
<td></td>
</tr>
<tr>
<td>4 - sine source</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routing audio element options:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-a &lt;pipeline_id&gt;</td>
<td>audio pipeline id (default 0)</td>
</tr>
<tr>
<td>-c</td>
<td>connect routing output</td>
</tr>
<tr>
<td>-d</td>
<td>disconnect routing output</td>
</tr>
<tr>
<td>-e &lt;element_id&gt;</td>
<td>routing element id (default 0)</td>
</tr>
<tr>
<td>-i &lt;input_id&gt;</td>
<td>routing element input (default 0)</td>
</tr>
<tr>
<td>-o &lt;output_id&gt;</td>
<td>routing element output (default 0)</td>
</tr>
</tbody>
</table>

### 4.3.3 Audio latency in loopback mode

The loopback mode reads audio samples from HiFiBerry’s ADC in an audio buffer and sends this buffer to the HiFiBerry’s DAC when fully loaded.

The end to end latency, between the analog audio input and the analog audio output, has been measured and is dependent on the audio buffer size and the audio sampling rate. The RTOS and SoC combination does not alter the latency measurements.
4.3.4 Running audio application: examples

4.3.4.1 Playing DTMF

To start DTMF playback with default parameters (48000 Hz sampling rate):

```
# harpoon_ctrl audio -r 0
```

The DTMF is played both to the Hifiberry RCA outputs as well as the onboard jack.

To run another audio use case, the playback must be stopped with the following command:

```
# harpoon_ctrl audio -s
```

4.3.4.2 Playing in loopback mode

In loopback mode, the SAI input is copied to the SAI output.

To start loopback mode with default parameters (48000 Hz sampling rate, 8 frame period size):

```
# harpoon_ctrl audio -r 2
```

To run another audio use case, the playback must be stopped with the following command:

```
# harpoon_ctrl audio -s
```

4.3.4.3 Playing a full audio pipeline

The reference audio application is based on a basic pipeline framework for audio processing. Different audio processing elements can be assembled in a pipeline to process audio from source(s) to sink(s). The pipeline is processed in real time, cyclically with a fixed period.

In the audio pipeline mode there is a three stage pipeline composed of a routing element in stage 2 which can link source elements from stage 1 to sink elements from stage 3.

---

**Table 2. Audio application latency**

<table>
<thead>
<tr>
<th>Sampling rate (kHz)</th>
<th>Audio latency (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Audio buffer size (frames)</td>
</tr>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td>192</td>
<td>610</td>
</tr>
<tr>
<td>176.4</td>
<td>660</td>
</tr>
<tr>
<td>96</td>
<td>1210</td>
</tr>
<tr>
<td>88.2</td>
<td>1310</td>
</tr>
<tr>
<td>48</td>
<td>2380</td>
</tr>
<tr>
<td>44.1</td>
<td>2600</td>
</tr>
</tbody>
</table>
When running the audio pipeline, the routes can be configured dynamically with the `harpoon_ctrl` command. This command uses source and sink indices to connect elements.

Table 3. Indices of source elements

<table>
<thead>
<tr>
<th>Index</th>
<th>Source element</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DTMF, sequence 1</td>
<td>Software generated</td>
</tr>
<tr>
<td>1</td>
<td>DTMF, sequence 2</td>
<td>Software generated</td>
</tr>
<tr>
<td>2</td>
<td>Sine wave, 440 Hz</td>
<td>Software generated</td>
</tr>
<tr>
<td>3</td>
<td>Sine wave, 880 Hz</td>
<td>Software generated</td>
</tr>
<tr>
<td>4</td>
<td>SAI5, left channel</td>
<td>Hardware source</td>
</tr>
<tr>
<td>5</td>
<td>SAI5, right channel</td>
<td>Hardware source</td>
</tr>
<tr>
<td>6</td>
<td>SAI3, left channel</td>
<td>Hardware source</td>
</tr>
<tr>
<td>7</td>
<td>SAI3, right channel</td>
<td>Hardware source</td>
</tr>
</tbody>
</table>

Table 4. Indices of sink elements

<table>
<thead>
<tr>
<th>Index</th>
<th>Sink element</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SAI5, left channel</td>
<td>Hardware sink</td>
</tr>
<tr>
<td>1</td>
<td>SAI5, right channel</td>
<td>Hardware sink</td>
</tr>
<tr>
<td>2</td>
<td>SAI3, left channel</td>
<td>Hardware sink</td>
</tr>
<tr>
<td>3</td>
<td>SAI3, right channel</td>
<td>Hardware sink</td>
</tr>
</tbody>
</table>

This makes for a flexible pipeline. For instance, the following commands starts the pipeline and configures the routing element to have a loopback between SAI5 input and SAI3 output (i.e. sound recorded by the HiFiBerry card played by the EVK’s internal...
codec) while a DTMF sequence is played on the left channel of SAI5’s output and a 440 Hz sine wave on the right channel of SAI5’s output (i.e. HiFiBerry’s output):

```
# harpoon_ctrl audio -r 3             # start audio pipeline
# harpoon_ctrl routing -i 4 -o 2 -c  # SAI5’s input to SAI3's output (L)
# harpoon_ctrl routing -i 5 -o 3 -c  # SAI5’s input to SAI3's output (R)
# harpoon_ctrl routing -i 0 -o 0 -c  # DTMF to SAI5’s output (L)
# harpoon_ctrl routing -i 2 -o 1 -c  # sinewave 440Hz to SAI5's output (R)
```

## 4.4 Industrial application

### 4.4.1 Features of the industrial application

The industrial application is available in the harpoon share directory of the root file system:

```
/usr/share/harpoon/inmates/freertos/industrial.bin  # FreeRTOS binary
/usr/share/harpoon/inmates/zephyr/industrial.bin    # Zephyr binary (i.MX 8M Plus EVK only)
```

This application contains several use cases that can be started and stopped via the Harpoon Linux user space application, namely `harpoon_ctrl`.

**Note:** In the current release, a single industrial use case is supported under Zephyr: CAN on i.MX 8M Plus EVK.

The different use cases are:

- **CAN (i.MX 8M Plus EVK):**
  - Simple loopback example that uses the flexCAN interface to send and receive CAN messages through internal loopback interconnect (no cable required).
  - Ping-pong: Two boards are connected through their CAN1 connectors (J19) with a male-male DB9 CAN cable. The later can either be purchased or built following the CAN pinout standard. Endpoint A (board A) sends CAN FD messages to Endpoint B (board B). Endpoint B uses two receiving queues to receive messages in turns, and prints the message content (and the receiving queue number) to the terminal after any queue is full.

- **Ethernet (FreeRTOS only):**
  - Simple MCUXpresso SDK API based application to send and receive packets through the ENET interface (i.MX 8M Mini/Nano EVK)
  - Full TSN stack based application, running a gPTP stack and sending/receiving TSN packets on a TSN network:
    - Through the ENET_QOS interface, acting as a controller/IO device (i.MX 8M Plus EVK).
    - Through the ENET interface, acting as a controller/IO device (i.MX 8M Mini EVK).

**Note:** The ENET interface does not support 802.1Qbv. Packets are transmitted using basic, software based, strict priority scheduling.
4.4.2 Starting the industrial application

To use the industrial application, Jailhouse must be started first. To start Jailhouse and the industrial application, create the corresponding Harpoon configuration file and run the harpoon service using systemd; for instance:

```
# harpoon_set_configuration.sh freertos industrial
```

**Note:** Avoid changing the configuration while the Harpoon service is running (silent failure when restarting the service).

The configuration file is stored under `/etc/harpoon/harpoon.conf` and the harpoon systemd service uses it to start Jailhouse and the industrial application:

```
# systemctl start harpoon
```

Once the harpoon service has been started, `harpoon_ctrl` is used to start or stop the industrial features with optional parameters. The different options for the industrial application are:

<table>
<thead>
<tr>
<th>Industrial CAN options:</th>
<th>run CAN mode id:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-r &lt;id&gt;</code></td>
<td>0 - loopback</td>
</tr>
<tr>
<td></td>
<td>1 - interrupt</td>
</tr>
<tr>
<td></td>
<td>2 - pingpong</td>
</tr>
<tr>
<td><code>-n &lt;node_type&gt;</code></td>
<td>acting as node 'A' or 'B' (default 'A')</td>
</tr>
<tr>
<td></td>
<td>0 - node 'A'</td>
</tr>
<tr>
<td></td>
<td>1 - node 'B'</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>stop CAN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Industrial ethernet options:</th>
<th>set hardware MAC address (default 91:e0:f0:00:fe:70)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a &lt;mac_addr&gt;</code></td>
<td>set hardware MAC address</td>
</tr>
<tr>
<td><code>-r &lt;id&gt;</code></td>
<td>run ethernet mode id:</td>
</tr>
<tr>
<td><code>-i &lt;role&gt;</code></td>
<td>for genAVB/TSN: endpoint role (default 'controller', if not specified)</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>stop ethernet</td>
</tr>
</tbody>
</table>

4.4.3 Running the industrial application: examples

4.4.3.1 CAN use cases

**Loopback**

Type this command to start a CAN loopback transfer (CAN FD mode enabled by default):

```
# harpoon_ctrl can -r 0
```

To execute a new CAN use case, the previous run must be stopped with the following command:

```
# harpoon_ctrl can -s
```
Ping-pong

One board must be chosen as node A and the other board as node B. (Note: Node B should start first) Data is sent from the node A to the node B.

Type this command to start receiving CAN FD data on board B:

```
# harpoon_ctrl can -n 1 -r 2
```

Type this command on board A to start transferring data:

```
# harpoon_ctrl can -n 0 -r 2
```

4.4.3.2 Ethernet through MCUXpresso SDK API

A simple reference use case is given to exchange ethernet packets using the SDK API:

```
# harpoon_ctrl ethernet -r 1
```

One possibility to verify that the use case is functional is to plug an ethernet cable on the ethernet connector on one end, and to a Linux host computer on the other end.

The expected output on the inmate cell console is as follows:

```
ENET test start.
ENET: Wait for PHY link up...
ENET: PHY link speed 1000M full-duplex
INFO: ethernet_sdk_enet_stat: not implemented
INFO: cpu_load_stats : CPU load: 0.00%
ENET test_result:
  TX: total = 100; succ = 100; fail = 0
  RX: total = 100; succ = 0; fail = 0; empty = 100
```

To verify that data are successfully received on the host side, one may use the `tcpdump` tool (sudo permissions may be required):

```
$ tcpdump -i <INTERFACE> -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s2, link-type EN10MB (Ethernet), capture size 262144 bytes
11:48:40.402104 00:04:9f:06:96:36 (oui Freescale) > 01:80:c2:00:00:0e (oui Unknown), ethertype LLDP (0x88cc), length 269: LLDP, length 255: imx8mp-lpddr4-evk
11:48:46.648227 00:00:00:00:00:00 (oui Ethernet) > Broadcast, 802.3, length 986: LLC, dsap Null (0x00) Individual, ssap Null (0x00) Response, ctrl 0x0302: Information, send seq 1, rcv seq 1, Flags [Final], length 986
  0x00000: 0001 0203 0405 0607 0809 0a0b 0c0d 0e0f ................
  0x00100: 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f .................
  0x00200: 2021 2223 2425 2627 2829 2a2b 2c2d 2e2f .!*%&'()++-./
  0x00300: 3031 3233 3435 3637 3839 3a3b 3c3d 3e3f 0123456789:;<=>?
  0x00400: 4041 4243 4445 4647 4849 4a4b 4c4d 4e4f @ABCDEFGHIJKLMNOPQRSTUVWXYZ\]^_.
  0x00500: 6061 6263 6465 6667 6869 6a6b 6c6d 6e6f `abcdefghijklmno
  0x00600: 7071 7273 7475 7677 7879 7a7b 7c7d 7e7f pqrstuvwxyz{|}~.
  0x00700: 8081 8283 8485 8687 8889 8a8b 8c8d 8e8f ................
  0x00800: 9091 9293 9495 9697 9899 9a9b 9c9d 9e9f ................
  0x00900: a0a1 a2a3 a4a5 a6a7 a8a9 aaab acad aeaf .................
```

4.4.3.3 Ethernet with GenAVB/TSN stack

A more complex ethernet use case uses the GenAVB/TSN Stack, which provides advanced implementation for AVB as well as Time-Sensitive Networking (TSN) functionalities. Some functions for the latter do require special TSN hardware support, available in the i.MX 8M Plus SoC for instance.

The below sections gives some details on the hardware requirements, setup preparation and test execution.

As far as the Harpoon demonstration goes, the controller (i.MX 8MP Plus) runs in the Cortex-A53 FreeRTOS cell. The IO devices, which can be any TSN endpoint (i.MX 8M Plus, RT1170, etc...) and the TSN bridge complete the TSN network environment for this use case.

4.4.3.3.1 Requirements

- Two TSN endpoints (i.MX 8M Plus LPDDR4 EVK, or optionally an i.MX RT1170 EVK)
- One TSN bridge (LS1028ARDB)

**Note:** The second IO Device is optional.
4.4.3.3.2 Setup preparation

One of the TSN endpoint needs to be configured as “controller” and the other one as “IO device”. Both endpoints are connected to the TSN bridge.

4.4.3.3.2.1 i.MX RT1170 TSN Endpoint - IO Device (Optional)

If using an i.MX RT1170 as the IO device, first flash the latest GenAVB/TSN Endpoint image (https://mcuxpresso.nxp.com/download/685e45c32bb022c898e4d11e6d914010).

Once the RT1170 is flashed, press ‘insert’ and set the following parameters:

```
IO_DEVICE_0>>write tsn_app/role 1
IO_DEVICE_0>>write tsn_app/period_ns 100000
```

Press ‘insert’ to exit the configuration mode and reboot.

4.4.3.3.2.2 TSN Bridge

LS1028ARDB can be used as a generic time-aware bridge, connected to other time-aware end stations or bridges.

By default, LS1028ARDB does not forward packets if no bridge interface is configured under Linux. Enabling bridge interface is dependent on the board used.

TSN Bridge Configuration

Use the following commands to configure bridge on LS1028ARDB:

```
# ls /sys/bus/pci/devices/0000:00:00.5/net/
```

Get switch device interfaces for swp0, swp1, swp2 and swp3 as shown below:

```
ip link set dev eno2 up
ip link add name br0 type bridge
```
ip link set br0 up
ip link set master br0 swp0 up
ip link set master br0 swp1 up
ip link set master br0 swp2 up
ip link set master br0 swp3 up

Then start gPTP:

```bash
# tsn.sh start
```

TSN Bridge logging

Logs are stored in /var/log/tsn-br.

- Linux command:

```bash
# tail -f /var/log/tsn-br
```

- The bridge stack statistics are similar to the endpoint stack ones except that they are reported for each of the external ports of the switch (Port 0 to 3) and also for the internal port connected to the endpoint stack (Port 4) in case of Hybrid setup.

- $\texttt{Pdelay}$ (propagation delay), $\texttt{Link status}$, $\texttt{AS capability}$ and $\texttt{Port Role}$ are printed out for each port.

<table>
<thead>
<tr>
<th>Port(0): domain(0, 0): Role: Master</th>
<th>Link: Up</th>
<th>asCapable: Yes</th>
<th>neighborGptpCapable: Yes</th>
<th>delayMechanism: P2P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port(0): Propagation delay (ns): 334.29</td>
<td>min 329 avg 333 max 342</td>
<td>variance 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port(1): domain(0, 0): Role: Disabled</td>
<td>Link: Down</td>
<td>asCapable: No</td>
<td>neighborGptpCapable: Yes</td>
<td>delayMechanism: P2P</td>
</tr>
<tr>
<td>Port(2): domain(0, 0): Role: Master</td>
<td>Link: Up</td>
<td>asCapable: Yes</td>
<td>neighborGptpCapable: Yes</td>
<td>delayMechanism: P2P</td>
</tr>
<tr>
<td>Port(2): Propagation delay (ns): 386.54</td>
<td>min 380 avg 385 max 390</td>
<td>variance 9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port(3): domain(0, 0): Role: Disabled</td>
<td>Link: Down</td>
<td>asCapable: No</td>
<td>neighborGptpCapable: No</td>
<td>delayMechanism: P2P</td>
</tr>
<tr>
<td>Port(4): domain(0, 0): Role: Disabled</td>
<td>Link: Down</td>
<td>asCapable: No</td>
<td>neighborGptpCapable: No</td>
<td>delayMechanism: P2P</td>
</tr>
</tbody>
</table>

If a port is not connected, Link status takes the value $\texttt{Down}$.

If a port is not capable of communicating a synchronized time, AS$\_\text{Capable}$ status takes the value $\texttt{No}$.

4.4.3.3 Running the TSN use case

To start the Ethernet use case from the inmate cell (acting as a TSN Endpoint - Controller), type the following command:

```bash
# harpoon_ctrl ethernet -r 0
```

To start the Ethernet use case from the inmate cell (acting as a TSN Endpoint - IO Device), type the following command:

```bash
# harpoon_ctrl ethernet -r 0 -i 0
```
The expected output in the inmate console is:

```
INFO: main_task             : Industrial application started!
[5835/93378]
INFO: ethernet_avb_tsn_init : ethernet_avb_tsn_init
INFO 0 app gavb_stack init : talker_entity_id 0x0000000000000000
INIT 0.000000000 stack-freertos genavb_init
: NXP's GenAVB/TSN stack version dev-d71ce4fc
INIT 0.000000000 stack-freertos hw_avb_timer_init
: hw_timer_init done
INIT 0.000000000 stack-freertos hw_clock_init
: rate: 24000000, period: 100000000, mult(to ns): 699050667, shift(to ns): 24, mult(to cycles): 103079215, shift(to cycles): 32
INIT 0.000000000 stack-freertos hw_clock_register
: hw clock id: 1 registered
INIT 0.000000000 stack-freertos hw_timer register
: hw_timer(C0600080) of clock id: 1 registered
INIT 0.000000000 stack-freertos hw_timer register
: hw_timer(C06000C8) of clock id: 1 registered
INIT 0.000000000 stack-freertos hw_timer register
: hw_timer(C0600110) of clock id: 1 registered
INFO 0.000000000 stack-freertos
hw_avb_timer_register_device : dev(C06003D0), ref clock 24000000 Hz, min delay cycles 240
INFO 0.000000000 stack-freertos
gpt_hw_timer_set_period : gpt_dev (C06003A0) set period 125(us), 3000(cycles)
INIT 0.000000000 stack-freertos
gpt_init : gpt_init : registered AVB HW timer(C06003D0) channel: 0, prescale: 1
ERR 0.000000000 stack-freertos
gpt_init : failed to register GPT media clock recovery
INIT 0.000000000 stack-freertos __port_init
: port(0): C0604DF8
INIT 0.000000000 stack-freertos enet_qos_init
: port(0) enet(0) core clock: 125000000 Hz, ptp ref clock: 100000000 Hz, ptp/system clock: 80000000 Hz
INIT 0.000000000 stack-freertos enet_qos_init
: port(0) enet(0) num TX queue: 5, num RX queue: 4
INIT 0.000000000 stack-freertos hw_clock_init
: rate: 1000000000, period: 3b9ac9ffc4653600, mult(to ns): 1, shift(to ns): 0, mult(to cycles): 1, shift(to cycles): 0
INIT 0.000000000 stack-freertos hw_clock_register
: hw clock id: 2 registered
INIT 0.000000000 stack-freertos hw_timer register
: hw_timer(C0605930) of clock id: 2 registered
INIT 0.000000000 stack-freertos hw_timer register
: hw_timer(C0605970) of clock id: 2 registered
INIT 0.000000000 stack-freertos hw_timer register
: hw_timer(C06059B0) of clock id: 2 registered, pps support
INIT 0.000000000 stack-freertos os_clock_init
: clock ID: 0 success, flags: 0
```
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 1 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 2 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 3 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 4 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 5 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 6 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 7 has no hw clock
INIT 0.000000000 stack-freertos _os_clock_init
    : clock ID: 8 success, flags: 1
INIT 0.000000000 stack-freertos _os_clock_init
    : clock ID: 9 success, flags: 0
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 10 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 11 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 12 has no hw clock
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 13 has no hw clock
INIT 0.000000000 stack-freertos _os_clock_init
    : clock ID: 14 success, flags: 4
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 15 has no hw clock

[6599/94179]
ERR 0.000000000 stack-freertos _os_clock_init
    : clock ID: 16 has no hw clock
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : port(0) num tc: 5, num sr: 2, num hw queues: 5
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : num hw queues: 5, num cbs: 2
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : tc(0)->hw_queue_id: 0, flags: 2, hw queue prop: 1
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : tc(1)->hw_queue_id: 1, flags: 2, hw queue prop: 1
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : tc(2)->hw_queue_id: 2, flags: 2, hw queue prop: 1
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : tc(3)->hw_queue_id: 3, flags: 1, hw queue prop: 2
INFO 0.000000000 stack-freertos
net_qos_map_traffic_class_to_hw_ : tc(4)->hw_queue_id: 4, flags: 1, hw queue prop: 2
INFO 0.000000000 stack-freertos hw_timer_request
    : hw_timer(C0600110)
INFO 0.000000000 stack-freertos os_timer_create
    : os_timer(C0626B78), queue: 0
INIT 0.000000000 stack-freertos net_tx_task
 INIT 0.000000000 stack-freertos net_rx_task
 INIT 0.000000000 stack-freertos net_task_init
 INFO 0.000000000 stack-freertos timer_system_create
 INFO 0.000000000 stack-freertos os_timer_create
 INIT 0.000000000 stack-freertos hw_avb_timer_start
 INIT 0.000000000 stack-freertos management_task
 INIT 0.000000000 stack-freertos __net_tx_init
 INIT 0.000000000 stack-freertos ipc_tx_init
 INIT 0.000000000 stack-freertos __net_rx_init
 INIT 0.000000000 stack-freertos ipc_rx_init
 INIT 0.000000000 stack-freertos management_task
 INIT 0.000000000 stack-freertos phy_task
 INIT 0.000000000 stack-freertos management_task_init
 INIT 0.000000000 stack-freertos gptp_task
 INFO 0.000000000 gptp gptp_check_config
 INIT 0.000000000 gptp gptp_init
 INIT 0.000000000 stack-freertos ipc_tx_init
 INIT 0.000000000 stack-freertos ipc_tx_init
 INFO 0.000000000 stack-freertos __net_rx_init
 INFO 0.000000000 stack-freertos net_add_multi
 INFO 0.000000000 stack-freertos timer_system_create
 INFO 0.000000000 stack-freertos os_timer_create
 INFO 0.000000000 stack-freertos timer_system_create
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665598), queue: C0659FA0
INIT 0.0000000000 gptp gptp_cmlds_init
: CMLDS link port (0) initialized
INFO 0.0000000000 gptp dump_priority_vector
: domain(0, 0) system priority vector: root identity 00bbcfffedee12
INFO 0.0000000000 gptp dump_priority_vector
: system priority vector: priority1 255  priority2 248
INFO 0.0000000000 gptp dump_priority_vector
: system priority vector: class 255  accuracy 255
INFO 0.0000000000 gptp dump_priority_vector
: system priority vector: variance 17258
INFO 0.0000000000 gptp dump_priority_vector
: system priority vector: source port identity 00bbccfffedee12, port number 0
INFO 0.0000000000 gptp dump_priority_vector
: system priority vector: port number 0 steps removed 0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665600), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665600), queue: C0659FA0
INIT 0.0000000000 gptp gptp_instance_init
: Configuring Port(0) (C0663B30) domain(0, 0) delayMechanism(P2P)
INIT 0.0000000000 gptp gptp_port_init_timers
: Port(0)
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665668), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665668), queue: C0659FA0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665668), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665668), queue: C0659FA0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665668), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665738), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665738), queue: C0659FA0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665738), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665738), queue: C0659FA0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665738), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665808), queue: 0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665808), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665808), queue: C0659FA0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665808), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665808), queue: C0659FA0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665808), queue: 0
INFO 0.0000000000 stack-freertos os_timer_create
: os_timer(C0665940), queue: 0
INFO 0.0000000000 stack-freertos timer_system_create
: os_timer(C0665940), queue: C0659FA0
ERR            0.000000000 common timer_start
: timer(C0663FD8) 0ms period
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C06659A8), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C06659A8), queue: C0659FA0
INIT           0.000000000 gptp gptp_instance_init
: instance(C06635F0) domain(0, 0) Is enabled (gm capable 0)
INFO           0.000000000 gptp dump_priority_vector
: domain(1, -1) system priority vector: root identity
00bccfffeddee12
INFO           0.000000000 gptp dump_priority_vector
: system priority vector: priority1 255 priority2 248
INFO           0.000000000 gptp dump_priority_vector
: system priority vector: class 255 accuracy 255
INFO           0.000000000 gptp dump_priority_vector
: system priority vector: variance 17258
INFO           0.000000000 gptp dump_priority_vector
: system priority vector: source port identity
00bccfffeddee12, port number 0
INFO           0.000000000 gptp dump_priority_vector
: system priority vector: port number 0 steps removed 0
INIT           0.000000000 gptp gptp_instance_init
: Configuring Port(0) (C0664AB0) domain(1, -1)
    delayMechanism(COMMON_P2P)
INIT           0.000000000 gptp gptp_port_init_timers
: Port(0)
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665A10), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665A10), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665A78), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665A78), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665AE0), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665AE0), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665B48), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665B48), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665BB0), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665BB0), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665C18), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665C18), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665C80), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665C80), queue: C0659FA0
INFO           0.000000000 stack-freertos timer_system_create
: os_timer(C0665CE8), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
: os_timer(C0665CE8), queue: C0659FA0
ERR            0.000000000 common timer_start
    : timer(C0664F58) 0ms period
INFO           0.000000000 stack-freertos timer_system_create
    : os_timer(C0665D50), queue: 0
INFO           0.000000000 stack-freertos os_timer_create
    : os_timer(C0665D50), queue: C0659FA0
INIT           0.000000000 gptp gptp_instance_init
    : instance(C0664570) domain(1, -1) is disabled (gm capable 0)
INFO           0.000000000 gptp gptp_link_down
    : Port(0): link is DOWN
INIT           0.000000000 stack-freertos ipc_rx_init
    : ipc(C06039C8, C0666810) success
INIT           0.000000000 stack-freertos ipc_tx_init
    : ipc(C0603898, C06669B0) success
INIT           0.000000000 stack-freertos os_timer_rx_init
    : ipc(C0602A58, C0666DC0) success
INFO           0.000000000 gptp gptp_link_down
    : Port(0): link is DOWN
INIT           0.159357387 stack-freertos gptp_task_init
    : gptp main completed
INIT           0.159357387 stack-freertos srp_task
    : srp task started
INIT           0.159357387 stack-freertos ipc_rx_init
    : ipc(C06039C8, C066B5E0) success
INIT           0.159357387 stack-freertos ipc_tx_init
    : ipc(C0603898, C066B780) success
INFO           0.159357387 stack-freertos __net_rx_init
    : socket(C066B920)
INFO           0.159357387 stack-freertos __net_tx_init
    : socket(C066BB40) port_id(0)
INIT           0.159357387 stack-freertos ipc_rx_init
    : ipc(C06014F8, C066BF00) success
INIT           0.159357387 stack-freertos ipc_tx_init
    : ipc(C0601628, C066COD0) success
INIT           0.159357387 stack-freertos ipc_tx_init
    : ipc(C0601758, C066C240) success
INIT           0.159357387 srp msrp_map_init
    : done
INFO           0.159357387 stack-freertos timer_system_create
    : os_timer(C066B388), queue: 0
INFO           0.159357387 stack-freertos os_timer_create
    : os_timer(C066B388), queue: C0667BF0
INFO           0.159357387 stack-freertos timer_system_create
    : os_timer(C066B3F0), queue: 0
INFO           0.159357387 stack-freertos os_timer_create
    : os_timer(C066B3F0), queue: C0667BF0
INIT           0.159357387 srp mrp_init
    : mrp_app(C066AD40) done
INIT           0.159357387 srp mrp_port_init
    : port(0) done
INFO           0.159357387 srp msrp_create_domain
    : port(0) domain(6, 3, 2) created, num domains 1
INFO           0.159357387 stack-freertos timer_system_create
    : os_timer(C066B458), queue: 0
INFO           0.159357387 stack-freertos os_timer_create
    : os_timer(C066B458), queue: C0667BF0
INFO           0.159357387 srp mrp Alloc_attribute
    : mrp_app(C066AD40) port(0) attr(C066BD60, MSRP_ATTR_TYPE_DOMAIN)
INFO 0.159357387 srp mrp_mad_join_request
: mrp_app(C066AD40) port(0) attr(C066AD60, 
MSRP_ATTR_TYPE_DOMAIN) new(1)
INFO 0.159357387 srp msrp_create_domain
: port(0) domain(5, 2, 2) created, num domains 2
INFO 0.159357387 srp mrp_alloc_attribute
: mrp_app(C066AD40) port(0) attr(C066C3E0, 
MSRP_ATTR_TYPE_DOMAIN)
INFO 0.159357387 srp mrp_mad_join_request
: mrp_app(C066AD40) port(0) attr(C066C3E0, 
MSRP_ATTR_TYPE_DOMAIN) new(1)
INFO 0.159357387 srp
msrp_domain_update_boundary_port : port(0) class(0), srp
boundary 1
INFO 0.159357387 srp
msrp_domain_update_boundary_port : port(0) class(1), srp
boundary 1
INFO 0.159357387 stack-freertos net_add_multi
: port(0) 01:80:c2:00:00:0e
INFO 0.159357387 srp msrp_port_enable
: port(0) enabled
INFO 0.159357387 srp msrp_enable
: msrp(C066ABB8) enabled
INIT 0.159357387 srp msrp_init
: msrp(C066ABB8) done
INIT 0.159357387 stack-freertos ipc_rx_init
: ipc(C0601C18, C066C470) success
INIT 0.159357387 stack-freertos ipc_tx_init
: ipc(C0601D48, C066C610) success
INIT 0.159357387 stack-freertos ipc_tx_init
: ipc(C0601E78, C066C7B0) success
INIT 0.159357387 srp mvrp_map_init
: done
INFO 0.159357387 stack-freertos timer_system_create
: os_timer(C066B4C0), queue: 0
INFO 0.159357387 stack-freertos os_timer Create
: os_timer(C066B4C0), queue: C0667BF0
INFO 0.159357387 stack-freertos timer_system_create
: os_timer(C066B528), queue: 0
INFO 0.159357387 stack-freertos timer_system_create
: os_timer(C066B528), queue: C0667BF0
INFO 0.159357387 stack-freertos timer_system_create
: os_timer(C066B590), queue: 0
INFO 0.159357387 stack-freertos os_timer_create
: os_timer(C066B590), queue: C0667BF0
INIT 0.159357387 srp mvrp_init
: mrp_app(C066B078) done
INFO 0.159357387 stack-freertos net_add_multi
: port(0) 01:80:c2:00:00:21
INIT 0.159357387 srp mvrp_port_init
: port(0) done
INIT 0.159357387 srp mvrp_init
: mvrp(C066B010) done
INIT 0.159357387 srp mmrp_init
: mmrp(C066AB38) done
INIT 0.159357387 srp srp_init
: srp(C0667DA0) done
INIT 0.159357387 stack-freertos srp_task
: started
INFO 0.159357387 srp msrp_port_status
: msrp(C066ABB8) port(0) operational (0)
INFO 0.159357387 srp mvrp_port_status
: mvrp(C066B010) port(0) operational (0)
INIT 0.159357387 stack-freertos phy_task
: phy(1) initialized
INIT 0.159357387 stack-freertos srp_task_init
: srp main completed
INFO: ethernet_avb_tsn_run : tsn_app config
INFO: ethernet_avb_tsn_run : mode : NETWORK_ONLY
INFO: ethernet_avb_tsn_run : role : 0
INFO: ethernet_avb_tsn_run : num_io_devices : 1
INFO: ethernet_avb_tsn_run : motor_offset : 0
INFO: ethernet_avb_tsn_run : control_strategy : 0
INFO: ethernet_avb_tsn_run : app_period : 100000
INFO: ethernet_avb_tsn_run : BUILD_MOTOR disabled,
MOTOR_NETWORK and MOTOR_LOCAL modes cannot be used
INFO 0.159357387 stack-freertos hw_timer_request
: hw_timer(C06059B0) pps
INFO 0.159357387 stack-freertos os_timer_create
: os_timer(C06532B0), queue: 0
INFO 0 app gavb_pps_init : success,
clk_id: 1
INFO 0 app cyclic_task_init : cyclic task
type: 0, id: 0
INFO 0 app cyclic_task_init : task params
INFO 0 app cyclic_task_init : task_period_ns
: 100000
INFO 0 app cyclic_task_init :
task_period_offset_ns : 0
INFO 0 app cyclic_task_init :
transfer_time_ns : 50000
INFO 0 app cyclic_task_init :
sched_traffic_offset : 35000
INFO 0 app cyclic_task_init :
: 0
INFO 0 app cyclic_task_init :
use_fp
INFO 0 app cyclic_task_init :
use_st
: 1
INFO 0.159357387 stack-freertos __net_rx_init
: socket(C066EAD0)
INFO 0.159357387 stack-freertos net_add_multi
: port(0) 91:e0:f0:00:fe:71
INFO 0.159357387 stack-freertos __net_tx_init
: socket(C066ECF0) port_id(0)
INFO 0.159357387 stack-freertos hw_timer_request
: hw_timer(C0605970)
INFO 0.159357387 stack-freertos os_timer_create
: os_timer(C066FB30), queue: 0
INFO 0 app cyclic_task_init : success
INFO 0 app tsn_net_st_config_enable : scheduled
traffic config enabled
INFO 0.159357387 stack-freertos __net_rx_init
: socket(C06719B0)
INFO 0.159357387 stack-freertos net_add_multi
: port(0) 91:e0:f0:00:fe:a0
INFO 0 app alarm_task_monitor_init : success
INFO 0.159357387 srp msrp_vector_add_event
: port(0) domain(5, 2, 2) MSRP_ATTR_TYPE_DOMAIN MRP_ATTR_EVT_NEW
To stop the Ethernet use case (to eventually re-start it), the previous run must be stopped with the following command:

```
# harpoon_ctrl ethernet -s
```

### 4.5 rt_latency application

The rt_latency application is a simple benchmark application for real-time OS that measures the latency (time delta, in nanoseconds) between hardware IRQ events and software actions:

- **irq delay**: time to enter in the software IRQ handler after an hardware IRQ occurs (hardware + hypervisor + IRQ vector latency)
- **irq to sched**: time to enter in an RTOS task, scheduled by the IRQ handler (irq delay + RTOS scheduler)

All measurements are done using GPT timer and relative to the hardware IRQ event time, with sub-microsecond precision.

When running, the rt_latency application prints out regular statistics, based on the measurements taken, to help characterize the system real-time latency.

The rt_latency application is available in the harpoon share directory of the root file system:

```
/usr/share/harpoon/inmates/freertos/rt_latency.bin # FreeRTOS binary
/usr/share/harpoon/inmates/zephyr/rt_latency.bin    # Zephyr binary
```

To use the rt_latency application, Jailhouse must be started first. To start Jailhouse and the rt_latency application, create an appropriate Harpoon configuration file and run the harpoon service with systemd. For instance:

```
# harpoon_set_configuration.sh freertos latency
# systemctl start harpoon
```
The harpoon service uses the `/etc/harpoon/harpoon.conf` configuration file that contains the RTOS and the application to run. By default, the configuration file points to the FreeRTOS audio application. To run the `rt_latency` application, we have generated a corresponding configuration file. This step needs to be run only once.

Once the harpoon service has been started, the following `rt_latency` trace is shown in the terminal emulator connected to the other serial port:

```
Harpoon v2.1.0
main_task: running
```

After booting, the `rt_latency` application waits for commands to be received. A list of available commands is shown using the following command:

```
# harpoon_ctrl -h
```

The usage for the `rt_latency` application is shown:

<table>
<thead>
<tr>
<th>Latency options:</th>
<th>run latency test case id</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-r &lt;id&gt;</code></td>
<td><code>-s</code></td>
</tr>
<tr>
<td></td>
<td>stop running test case</td>
</tr>
</tbody>
</table>

Examples:

To stop the `rt_latency` application's current test case:

```
# harpoon_ctrl latency -s
```

To run a test case:

It is possible to engage some CPU load and/or IRQ load to measure their impact on the latency. To do so, different test cases (TC) can be executed, by specifying the test case id with the “-r” option:

```
# harpoon_ctrl latency -r <TC_ID>
```

**TC_ID:**

- 1: no extra load
- 2: extra CPU load (low priority task, executing busy loop and consuming all available CPU time)
- 3: extra IRQ load
- 4: extra CPU load + semaphore load
- 5: extra CPU load + Linux load (not provided by the test case)
- 6: extra CPU load + cache flush (instruction cache only for this release)

To execute test case 1:

```
# harpoon_ctrl latency -r 1
```

When running, latency statistics are printed out every 10 seconds:

```
---
Running test case 1:
benchmark_task: running
```
Both the irq delay and the irq to sched statistics are shown:

- min/mean/max: minimum, average and maximum latency value measured within the last period of time
- absmin/absmax: minimum and maximum latency value measured since the beginning of the test
- A histogram is also shown to give an idea of repartition of the measured latency values

Table 5. Real-time latencies measured on i.MX 8M Plus/FreeRTOS (in ns)

<table>
<thead>
<tr>
<th>Test description</th>
<th>i.MX 8M Plus IRQ Latency (ns)</th>
<th>i.MX 8M Plus Task Latency (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Average</td>
</tr>
<tr>
<td>No system load</td>
<td>708</td>
<td>708</td>
</tr>
<tr>
<td>Low priority task CPU load</td>
<td>708</td>
<td>711</td>
</tr>
<tr>
<td>Low priority IRQ load</td>
<td>11,000</td>
<td>11,042</td>
</tr>
<tr>
<td>Low priority task CPU load, mutex</td>
<td>708</td>
<td>728</td>
</tr>
<tr>
<td>Linux CPU + memory load</td>
<td>708</td>
<td>708</td>
</tr>
<tr>
<td>RTOS cold cache</td>
<td>708</td>
<td>912</td>
</tr>
</tbody>
</table>

Table 6. Real-time latencies measured on i.MX 8M Mini/FreeRTOS (in ns)

<table>
<thead>
<tr>
<th>Description</th>
<th>i.MX 8M Mini IRQ Latency (ns)</th>
<th>i.MX 8M Mini Task Latency (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Average</td>
</tr>
<tr>
<td>No system load</td>
<td>1,125</td>
<td>1,161</td>
</tr>
</tbody>
</table>
### Table 6. Real-time latencies measured on i.MX 8M Mini/FreeRTOS (in ns)

<table>
<thead>
<tr>
<th>Description</th>
<th>Min</th>
<th>Average</th>
<th>Max</th>
<th>Stddev</th>
<th>Min</th>
<th>Average</th>
<th>Max</th>
<th>Stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low priority task CPU load</td>
<td>1,125</td>
<td>1,166</td>
<td>1,583</td>
<td>350</td>
<td>3,000</td>
<td>3,006</td>
<td>5,125</td>
<td>9,937</td>
</tr>
<tr>
<td>Low priority IRQ load</td>
<td>11,500</td>
<td>12,097</td>
<td>12,250</td>
<td>27,249</td>
<td>13,375</td>
<td>14,221</td>
<td>14,416</td>
<td>31,240</td>
</tr>
<tr>
<td>Low priority task CPU load, mutex</td>
<td>1,125</td>
<td>1,163</td>
<td>1,250</td>
<td>172</td>
<td>2,958</td>
<td>3,004</td>
<td>3,916</td>
<td>5994</td>
</tr>
<tr>
<td>Linux CPU + memory load</td>
<td>1166</td>
<td>1167</td>
<td>1,625</td>
<td></td>
<td>2,958</td>
<td>3,004</td>
<td>4,125</td>
<td></td>
</tr>
<tr>
<td>RTOS cold cache</td>
<td>1,166</td>
<td>1,174</td>
<td>3,500</td>
<td>19,086</td>
<td>2,958</td>
<td>3,037</td>
<td>6,666</td>
<td>51,792</td>
</tr>
</tbody>
</table>

### Table 7. Real-time latencies measured on i.MX 8M Plus/Zephyr (in ns)

<table>
<thead>
<tr>
<th>Description</th>
<th>i.MX 8M Plus IRQ Latency (ns)</th>
<th>i.MX 8M Plus Task Latency (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Average</td>
</tr>
<tr>
<td>No system load</td>
<td>875</td>
<td>920</td>
</tr>
<tr>
<td>Low priority task CPU load</td>
<td>791</td>
<td>918</td>
</tr>
<tr>
<td>Low priority IRQ load</td>
<td>9,583</td>
<td>11,315</td>
</tr>
<tr>
<td>Low priority task CPU load, mutex</td>
<td>750</td>
<td>917</td>
</tr>
<tr>
<td>Linux CPU + memory load</td>
<td>750</td>
<td>929</td>
</tr>
<tr>
<td>RTOS cold cache</td>
<td>916</td>
<td>935</td>
</tr>
</tbody>
</table>
5 Known Issues

Table 8. Known issues

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Workarounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRPN-245</td>
<td>Linux cannot access eMMC.</td>
<td>Store root file system on SD card or NFS.</td>
</tr>
<tr>
<td>HRPN-447</td>
<td>Audio glitches on i.MX 8MN EVK</td>
<td>-</td>
</tr>
<tr>
<td>HRPN-448</td>
<td>RTOS crashes on Ethernet use case restart</td>
<td>Restart the Jailhouse cell.</td>
</tr>
<tr>
<td>HRPN-479</td>
<td>CAN communication does not work consistently when connecting two i.MX 8M Plus EVKs.</td>
<td>Investigation in progress.</td>
</tr>
</tbody>
</table>

6 Technical Details on Harpoon Applications

6.1 Description

Harpoon reference applications are embedded in a repository named harpoon-apps.

Several RTOS applications are embedded in this repository, which may run in Jailhouse cells, based on an RTOS (currently using FreeRTOS and Zephyr) and leveraging the MCUXpresso SDK. As a consequence, FreeRTOS-Kernel and mcux-sdk repositories are required to build FreeRTOS based applications and zephyr and hal_nxp repositories are required to build Zephyr based applications. Additionally, repository GenAVB_TSN is needed to build the industrial application. The west tool is used to fetch those repositories, along with harpoon-apps Git tree.

To manage Linux - RTOS communication, a control application running in the Linux root cell is used. This application is to be compiled with the Yocto toolchain.

The next section explains how to build binaries (RTOS application and Linux control application).

Related information
https://docs.zephyrproject.org/latest/guides/west/index.html

6.2 Manual build

6.2.1 Setting up the environment

You need to have both git and west installed to fetch the source code for Harpoon-apps, FreeRTOS, Zephyr and MCUXpresso SDK:

```bash
$ west init -m https://github.com/NXPmicro/harpoon-apps --mr harpoon_2.1.0 hww
$ cd hww
$ west update
```
6.2.2 Building the RTOS application for the RTOS cell

6.2.2.1 Building FreeRTOS based applications

FreeRTOS applications for Armv8-A must be compiled with a compatible toolchain.

The reference toolchain is the GNU Arm cross-toolchain for the A-profile cores GCC 10.3-2021.07.

To download the toolchain and install it:

```
$ wget https://developer.arm.com/-/media/Files/downloads/gnu-a/10.3-2021.07/binrel/gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf.tar.xz
$ tar -C /opt/ -xvf gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf.tar.xz
```

If starting from a fresh console, the cross-compiler variable must be set:

```
$ export ARMGCC_DIR=/opt/gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf/
```

Then build an RTOS application:

```
$ cd harpoon-apps/<RTOS_APP>/freertos/boards/<BOARD>/armgcc_aarch64
$ ./build_ddr_release.sh
```

Where:
- `RTOS_APP` is audio, industrial or rt_latency.
- `BOARD` is `evkmimx8mm` for i.MX 8M Mini, `evkmimx8mn` for i.MX 8M Nano, `evkmimx8mp` for i.MX 8M Plus.
- Build artefacts are available in the directory `ddr_release/`.
- The artefact to be used on target is the RTOS application binary: `<RTOS_APP>.bin`.

6.2.2.2 Building Zephyr based applications

Install cross-compile toolchain firstly, and then set the cross-compile environment:

```
$ export ARMGCC_DIR=/opt/gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf/
```

Then build an RTOS application:

```
$ cd harpoon-apps/<RTOS_APP>/zephyr/boards/<BOARD>/armgcc_aarch64
$ ./build_singlecore.sh
```

Where:
- `RTOS_APP` is audio, industrial, or rt_latency.
- `BOARD` is `evkmimx8mm` for i.MX 8M Mini, `evkmimx8mn` for i.MX 8M Nano, and `evkmimx8mp` for i.MX 8M Plus.
- Build artefacts are available in the directory `build_singlecore/zephyr/`.
- The artefact to be used on target is the RTOS application binary: `<RTOS_APP>.bin`. 
6.2.3 Building the Linux control application for the root cell

The Linux control application for Armv8-A must be compiled with a compatible toolchain. The reference toolchain is the Poky Arm cross-toolchain built with Yocto.

To generate this toolchain:

```
$ bitbake meta-toolchain
```

This generates a toolchain installer in directory `tmp/deploy/sdk`. The installer name depends on the `DISTRO` and `MACHINE` variables and on the image name of the current build. For instance, for an i.MX build, the installer name is `fsl-imx-xwayland-glibc-x86_64-meta-toolchain-cortexa53-crypto-imx8mm-lpddr4-evk-toolchain-5.10-hardknott.sh`.

When executed, the installer prompts for a directory where to put the toolchain. The default location for the i.MX toolchain is `/opt/fsl-imx-xwayland/5.10-hardknott`.

When the toolchain is installed, different cross-compile variables must be set. This is done by sourcing script `environment-setup-cortexa53-crypto-poky-linux`. For example with default installation path:

```
$ . /opt/fsl-imx-xwayland/5.10-hardknott/environment-setup-cortexa53-crypto-poky-linux
```

The Harpoon control application can then be built:

```
$ cd harpoon-apps/ctrl
$ ./build_ctrl.sh
```

The build artefact (`harpoon_ctrl`) is available in the same directory and can be used on target.

6.3 Starting an RTOS application with Jailhouse

6.4 Developing a Harpoon Application

Harpoon-apps is the basis to create a Harpoon application. It links with MCUXpresso drivers and a RTOS (FreeRTOS and Zephyr).

A Harpoon application has its own directory in the root folder of the harpoon-apps repository. Examples include `audio`, the audio test application, and `rt_latency`, the real-time benchmark test application.

6.4.1 Architecture of the audio application

The audio application, which serves as an example for this chapter, has the following architecture.
The DAC and ADC on the HiFiBerry card are controlled by the audio application. Control is done through I2C3 and data throughput through SAI5.

### 6.4.2 Source file creation

This chapter gives some information on how to develop an application for Harpoon by using the audio application as an example.

First, the application directory must be created in the root directory of repository harpoon-apps.

This directory contains the source code for the application, a CMake configuration file listing the files to be compiled. Source file can be common to all RTOS and platform, be RTOS dependent and / or platform dependent. Helper scripts are provided to build the application for each RTOS / platform combination.
The application starts in function `main()`, defined in file `main.c`. RTOS specific code goes to directory `audio/freertos` and `audio/zephyr`. Board specific code (clock configuration, hardware description, MMU configuration) goes to directory `audio/\langle rtos\rangle/boards/\langle boardid\rangle` and `audio/boards/\langle boardid\rangle`. OS-agnostic code goes to directory `audio/common`.

### 6.4.3 Board specific code

Board specific code and header files for the audio application include:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>app_board.h</code></td>
<td>Definition of SAI and I2C instances used for the demo. I2C addresses of HiFi Berry's DAC and ADC. SAI configuration. Audio samples format.</td>
</tr>
<tr>
<td><code>app_mmu.h</code></td>
<td>Device memory to map with MMU (includes SAI and I2C).</td>
</tr>
<tr>
<td><code>sai_clock_config.c</code></td>
<td>Configuration of Audio PLLs, Audiomix (for i.MX 8M Plus) and SAI clocks.</td>
</tr>
<tr>
<td><code>sai_config.c</code></td>
<td>Define configuration of each SAI instance.</td>
</tr>
<tr>
<td><code>codec_config.c</code></td>
<td>Helper functions to open, configure and close DAC and ADC drivers.</td>
</tr>
<tr>
<td><code>pin_mux.c</code></td>
<td>Functions to set IOMUX for the application use case.</td>
</tr>
<tr>
<td><code>CMakeLists.txt</code></td>
<td>CMake configuration file that includes all necessary MCUXpresso drivers.</td>
</tr>
<tr>
<td><code>flags.cmake</code></td>
<td>CFLAGS and LDFLAGS definitions for building the application.</td>
</tr>
</tbody>
</table>

### 6.4.4 Controlling application from Linux side

Linux side can control the Harpoon application by sending messages through the ivshmem communication channel provided by Jailhouse. The mailbox API is used for communication.

The audio application leverages this in function `main_task()`, defined in `audio/freertos/main.c`.

RTOS is prepared to work with the ivshmem memory:

```c
rc = ivshmem_init(0, &mem);
```
Then mailbox is initialized:

```c
mailbox_init(&m, mem.out, mem.out + mem.out_size * mem.id, false);
```

Finally, the application’s main thread periodically looks for incoming control messages:

```c
do {
    command_handler(&m, &ctx);
    [...]
} while (1);
```

The Linux user space application that sends control messages is located in the directory `ctrl` of the `harpoon-apps` repository.

## 7 Revision History

The following table provides the revision history for this document.

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Date</th>
<th>Substantive changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAR 2.1.0</td>
<td>28 July 2022</td>
<td>Minor changes to Section 4 and Section 5. Compatible with Real Time Edge Software Rev 2.3 release.</td>
</tr>
</tbody>
</table>
| EAR 2.1.0       | 30 June 2022| • New industrial application in harpoon-apps  
|                 |            | • Implementation of flexible audio pipeline in harpoon-apps  
|                 |            | • Support for i.MX 8M Nano EVK for i.MX Yocto  
|                 |            | • Support for EVK's internal audio codecs  
|                 |            | • Support for systemd  
|                 |            | • Support for Zephyr  
|                 |            | • Drivers for FlexCAN, ENET, ENET_QOS |
| EAR 2.0.1       | 29 March 2022| Full integration to NXP Real-Time Edge. |
| EAR 2.0.0       | 14 January 2022| Introduction of `harpoon-apps`. Support of FreeRTOS. Support of both i.MX BSP and Real-Time Edge SW. |
8.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or network changes) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer’s third party customer(s). Customer is responsible for doing all necessary testing for the customer’s applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer’s third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer’s general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors’ warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors’ specifications such use shall be solely at customer’s own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors’ standard warranty and NXP Semiconductors’ product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

8.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.
Contents

1 Overview .............................................................. 2
  1.1 Architecture .................................................... 2
  1.2 Hardware resource partitioning ........................... 3
2 Building Harpoon Yocto images ........................... 5
  2.1 i.MX Yocto ........................................................ 5
  2.2 Real-time Edge Yocto ......................................... 5
3 Hardware Setup ...................................................... 5
  3.1 i.MX Reference Boards ..................................... 5
  3.2 Audio use case hardware .................................. 7
  3.3 Industrial use case hardware ............................. 10
4 Running Harpoon Reference Applications ........ 12
  4.1 Basic setup .................................................... 12
  4.2 Starting Linux kernel ......................................... 12
  4.3 Audio application ............................................. 13
    4.3.1 Features of the audio application .................... 13
    4.3.2 Starting the audio application with
      Jailhouse ..................................................... 13
    4.3.3 Audio latency in loopback mode .................... 14
    4.3.4 Running audio application: examples .............. 15
      4.3.4.1 Playing DTMF ........................................ 15
      4.3.4.2 Playing in loopback mode ........................ 15
      4.3.4.3 Playing a full audio pipeline .................. 15
  4.4 Industrial application ..................................... 17
    4.4.1 Features of the industrial application ........... 17
    4.4.2 Starting the industrial application ............... 18
    4.4.3 Running the industrial application:
      examples ..................................................... 18
      4.4.3.1 CAN use cases ...................................... 18
      4.4.3.2 Ethernet through MCUXpresso SDK API .......... 19
      4.4.3.3 Ethernet with GenAVB/TSN stack ............... 20
  4.5 rt_latency application ..................................... 31
5 Known Issues ....................................................... 35
6 Technical Details on Harpoon Applications .... 35
  6.1 Description .................................................... 35
  6.2 Manual build .................................................. 35
    6.2.1 Setting up the environment ........................... 35
    6.2.2 Building the RTOS application for the
      RTOS cell .................................................... 36
      6.2.2.1 Building FreeRTOS based applications ........ 36
      6.2.2.2 Building Zephyr based applications .......... 36
      6.2.3 Building the Linux control application for the
        root cell .................................................. 37
      6.3 Starting an RTOS application with
        Jailhouse .................................................. 37
      6.4 Developing a Harpoon Application .................. 37
        6.4.1 Architecture of the audio application ........ 37
        6.4.2 Source file creation ................................ 38
        6.4.3 Board specific code ................................ 40
        6.4.4 Controlling application from Linux side ........ 40
  7 Revision History ............................................... 41
8 Legal information ............................................... 42

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2022 NXP B.V. All rights reserved.
For more information, please visit: http://www.nxp.com
Date of release: 28 July 2022
Document identifier: HRPNUG