

Understanding Device Tree Files in Multicore Hypervisor/LWE Implementations

by *Networking and Multimedia Group*
Freescale Semiconductor, Inc.
Austin, TX

In some multicore usage scenarios, it is desirable to divide the available core processors and other system resources into individual partitions, each of which can function as an individual, self-contained system. Freescale's implementation of Hypervisor, also known as "Topaz," utilizes a series of device trees to specify how the overall system resources in a multi-core system are divided among various partitions, as well as how each of those individual partitions make use of the resources that are allocated to it.

This application note explains the contents of the master device tree in a multicore Hypervisor implementation used to allocate system resources to the individual partitions. Additionally, it describes the contents of the individual device trees that each partition uses for local allocation of those resources.

1 Basic System Configuration

1.1 Hypervisor

The multicore system that is partitioned and managed by the Freescale implementation of Hypervisor consists of multiple CPUs as well as shared resources such as IO and memory.

Contents

1. Basic System Configuration	1
2. Hypervisor Device Tree	3
3. Partition Device Tree	13
4. Executing Under the Simics® Simulator	16
5. Revision History	17
A. Example Device Tree File Set for an 8-Core, 8-Partition Implementation	18

The Hypervisor provides a thin layer between the actual system hardware and the individual partitions that share the system resources. The Hypervisor device tree contains information regarding how the available systems resources are divided up among the individual partitions. In addition to the master Hypervisor device tree, the implementation requires each of the individual partitions to have its own device tree that specifies how the resources allocated to it are used within the partition.

- Hypervisor device tree allocates resources to individual partitions
- Guest device trees define how individual partitions use resources

Figure 1 shows how the system resources are partitioned.

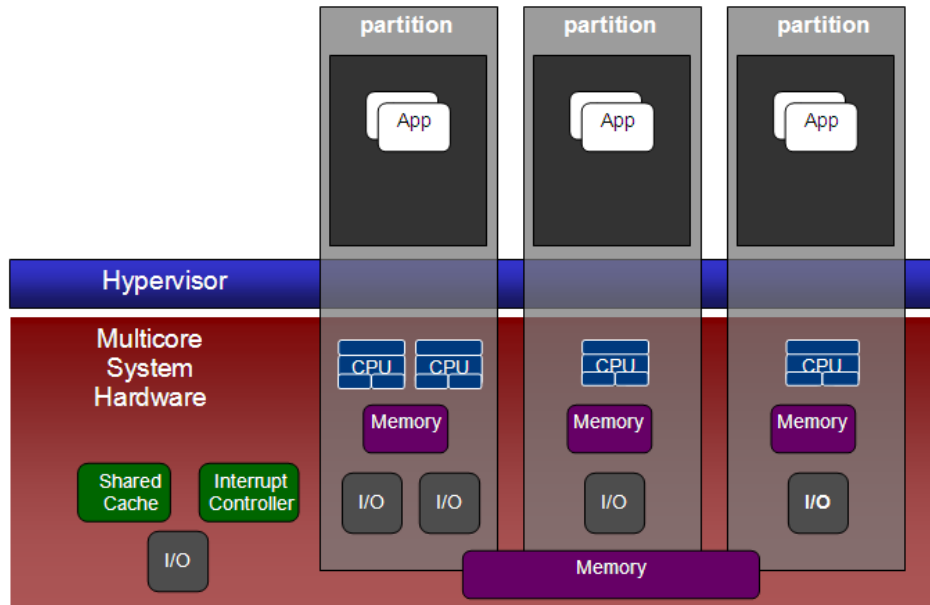


Figure 1. Partitioning of System Resources

1.2 Light Weight Executive

The Light Weight Executive (LWE) consists of a minimal set of library functions that allow for the execution of application code on a partition running under Hypervisor. This eliminates the overhead required to execute a full operating system on the individual partitions to have a framework under which to run application code. For example, for an eight-core system, a user might choose to set aside two CPUs under one partition to run an SMP Linux operating system, while the remaining six are each allocated to individual partitions that run application code under LWE.

1.3 Boot Process

Although the focus of this document is on the contents of the device tree files used to describe how system resources are allocated, it is helpful to know a bit about how the boot code, Hypervisor, and the individual partitions are initialized. The boot process can be described in the following phases:

1. U-Boot is executed to initialize the processor and to begin execution of the operating system code.

2. The operating system code that is executed is actually the Hypervisor. When Hypervisor is executed, it reads the master Hypervisor device tree file to obtain the definition of how the system is to be partitioned.
3. Hypervisor initializes the individual partitions using the individual partition device trees that are specified in the partitions node of the Hypervisor device tree.

2 Hypervisor Device Tree

The Hypervisor device tree is used to define how the available hardware resources will be allocated among partitions. Device trees are created from a text based source file, typically named with a “.dts” file extension. An application known as the “device tree compiler” creates a binary image, typically carrying a “.dtb” file extension.

A device tree file consists of several blocks, known as “nodes.” Each node can contain child nodes. Additionally, nodes contain items known as “properties” that define the attributes of the node. The Hypervisor device tree must contain the following nodes:

- **cpus** and **cpu**—Defines all physical CPUs in the system
- **memory** node—Specifies the private memory of the Hypervisor
- **chosen** node—Defines any global Hypervisor configuration options
- **partitions** node—Defines the guest partitions
- **csr/simple-bus** node—Defines the private I/O resources in CCSR space that belong to the Hypervisor
- **byte-channels** node—Defines all byte-channels. At least one byte-channel must be defined for the Topaz console

It is important to note that it is possible for boot code to modify the device tree binary prior to use. As explained in the discussion of the individual nodes below, there are a couple of properties that are listed as required by the Topaz documentation that do not appear in the device tree source file. This is because the U-boot code for this implementation modifies the device tree binary prior to use to add some of these required items.

The Hypervisor device tree conforms to a specification known as ePAPR. The ePAPR specification is an extension of the Power Architecture™ Platform Requirements specification that addresses boot services. Additional information regarding ePAPR can be found at: <http://www.power.org>.

2.1 Cpus and Cpu Nodes

The *cpus* node contains multiple *cpu* child nodes that must define all of the physical CPUs that exist in the system. One CPU, typically CPU 0, is designated as the master CPU. In addition to properties defining basic CPU related items such as cache line sizes and bus frequencies, the master *cpu* node contains a “status” property that is set to the value “okay.”

In contrast, each secondary *cpu* node must have the following properties:

- *status* must be set to “disabled.”
- *enable-method* must be set to “spin-table.”

- *cpu-release-addr* must be set as specified in the ePAPR.

Note that the *cpu-release-addr* property is one of those items that does not actually appear in the source file, but rather is added by U-Boot.

[Example 1](#) is an excerpt from a .dts file showing a *cpus* node containing child nodes to define 3 processors, one of which is the master, and the remaining two of which are secondary CPUs.

Example 1. cpus Node

```
cpus {
    #address-cells = <1>;
    #size-cells = <0>;

    cpu@0 {
        device_type = "cpu";
        compatible = "fsl,e500mc", "fsl,e500";
        reg = <0>;
        d-cache-line-size = <20>;
        i-cache-line-size = <20>;
        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
        status = "okay";
    };

    cpu@1 {
        device_type = "cpu";
        compatible = "fsl,e500mc", "fsl,e500";
        reg = <1>;
        d-cache-line-size = <20>;
        i-cache-line-size = <20>;
        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
        status = "disabled";
        enable-method = "spin-table";
    };
};
```

```
cpu@2 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <2>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;
    timebase-frequency = <0x83333333>;
    bus-frequency = <0x66666666>;
    clock-frequency = <0x66666666>;
    status = "disabled";
    enable-method = "spin-table";
};
};
```

2.2 Partitions and Partition Nodes

The *partitions* node is used to define and allocate resources to each individual partition. The *partitions* node contains a *partition* sub-node for each partition that is implemented. Each *partition* node must contain the following properties:

- A binary image of the guest device tree that is used by that partition
- The devices that make up the partition
- The physical to logical CPU id mapping for a partition's CPUs
- A property specifying which CPU should be the boot CPU for an SMP partition
- True physical address to guest physical address mapping

The following subsections provide a detailed explanation of the contents of the properties in the partition nodes.

2.2.1 fsl,hv-cpus

The *fsl,hv-cpus* property defines which physical CPUs are associated with a partition. The value of this property is expressed as an array of an arbitrary number of pairs that represent a starting CPU index followed by the number of sequential CPUs beginning at the specified index that is associated with the partition.

Example 2. fsl,hv-cpus—one CPU starting at physical CPU 0

```
fsl,hv-cpus = <0 1>
```

Example 3. fsl,hv-cpus—three CPUs starting at physical CPU 4

```
fsl,hv-cpus = <4 3>
```

Example 4. fsl,hv-cpus—one CPU starting at physical CPU 2 followed by one CPU starting at physical CPU 4

```
fsl,hv-cpus = <2 1 4 1>
```

Example 5. fsl,hv-cpus—one CPU starting at physical CPU1 followed by one CPU starting at physical CPU 3 followed by two CPUs starting at physical CPU 5

```
fsl,hv-cpus = <1 1 3 1 5 2>
```

2.2.2 fsl,hv-dtb

Each partition requires a device tree to define how the resources allocated it by the master device tree are utilized within the partition. The *fsl,hv-dtb* property specifies a pointer to a compiled device tree binary file. When the master Hypervisor device tree is compiled, the partition device tree files specified by this property are included as a part of the master Hypervisor device tree binary image. During initialization, this partition device tree binary is loaded at the address specified by the *fs,hv-dtb-window* property (described in [Section 2.2.3, “fsl,hv-dtb-window”](#)) for use by the partition being defined.

Example 6. fsl,hv-dtb

```
fsl,hv-dtb = mydevicetree.dtb
```

2.2.3 fsl,hv-dtb-window

The *fsl,hv-dtb-window* property defines the address within the physical memory of the guest partition where the device tree binary specified by the *fsl,hv-dtb* property will be loaded. This property is specified as a <prop-encoded-array> encoded as: < *dtb-addr-start dtb-size* >

dtb-addr is a <u64> value—the guest physical address specifying the start of a memory range where the guest device tree is to be loaded

dtb-size is a <u32> value specifying the size of the memory window for the DTB

Because *dtb-addr* is a 64-bit value, two 32-bit integer values are used in the <prop-encoded-array> to specify the full 64-bit value. In the following example the first two arguments “0 01000000” combine to form the full 64-bit address and the argument “10000” is a 32-bit value specifying the size.

Example 7. fsl,hv-dtb-window

```
fsl,hv-dtb-window = <0 01000000 10000>
```

2.2.4 fsl,hv-physaddr-map

The *fsl,hv-physaddr-map* property specifies the regions of the true physical memory that are made available to the partition as well as the guest physical address locations that they are mapped to within the guest partitions. The property is encoded as an arbitrary number of triplets in the form *<guest-physical-addr, true-physical-address, length>*.

guest-physical-addr is a *<u64>* value specifying the start of the guest physical address range.

true-physical-address is a *<u64>* value specifying the start of the true physical address range.

length is a *<u32>* value specifying the length of the address range.

Because both the guest physical address and the true physical address values are 64-bit values, two 32-bit integer values are used in the property array to contain each of the full 64-bit values. As a result, each triplet actually consists of five 32-bit integer values.

Consider the following example:

```
fsl,hv-physaddr-map = <0 0 0 1000000 17000000  
0 17000000 0 20000000 1000000>
```

This code example contains two sets of triplets consisting of five 32-bit values each. The first set of triplets specifies a memory range that is mapped to the guest partition starting at guest physical address 0x00 from the true physical address 0x1000000, and is 1700000 bytes in size, shown in [Figure 2](#).

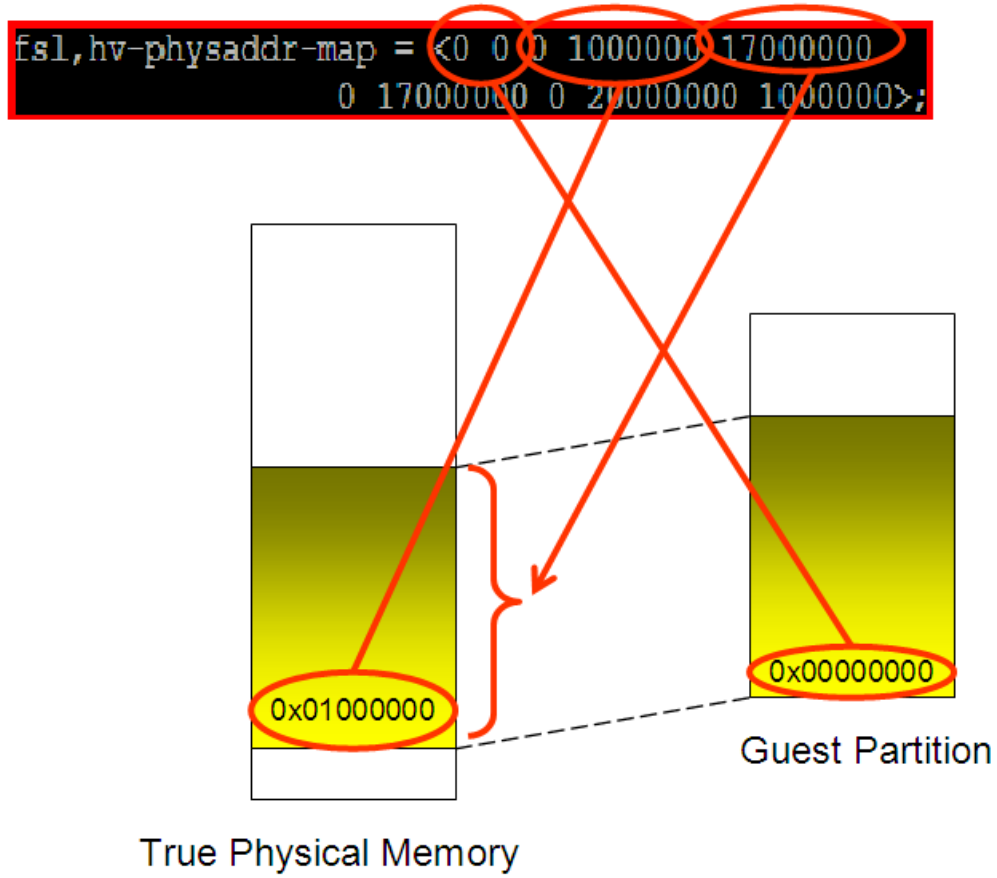


Figure 2. Physical Address Mapping (First Triplet from Example)

The second triplet indicates a memory range that is mapped to the guest partition starting at the guest physical address 0x17000000 from the true physical address 0x20000000, and is 1000000 in size, shown in Figure 3.

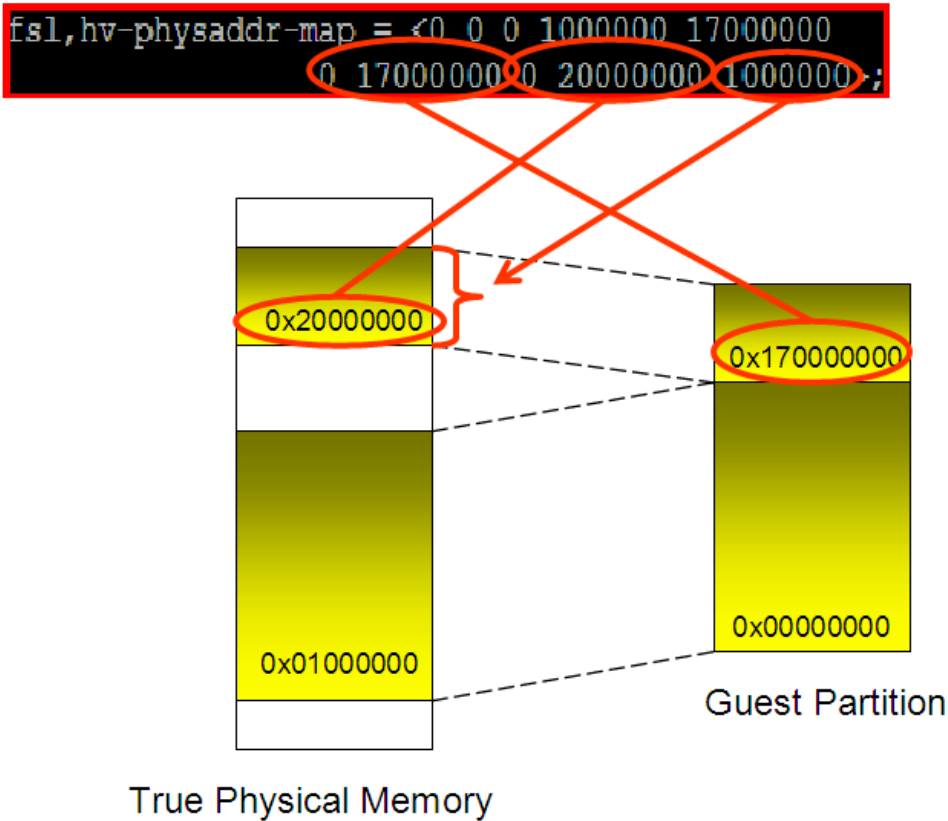


Figure 3. Physical Address Mapping (Second Triplet from Example)

From the perspective of the guest partition this creates a contiguous block of memory starting at guest physical address 0x00 and is 18000000 bytes in size, shown in Figure 4. However, that 18000000 block of memory is not contiguous in the true physical memory.

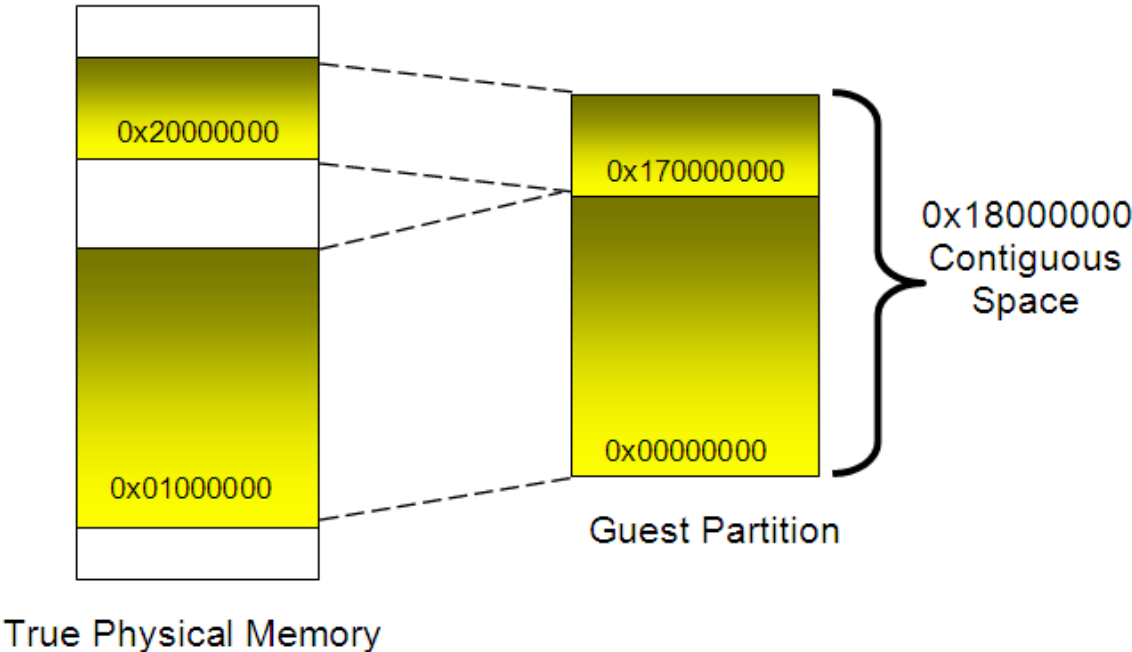


Figure 4. Memory Mapping Produced from Example

It is possible to map the same physical memory to multiple guest partitions to be used as an inter-partition communications mechanism, shown in [Figure 5](#).

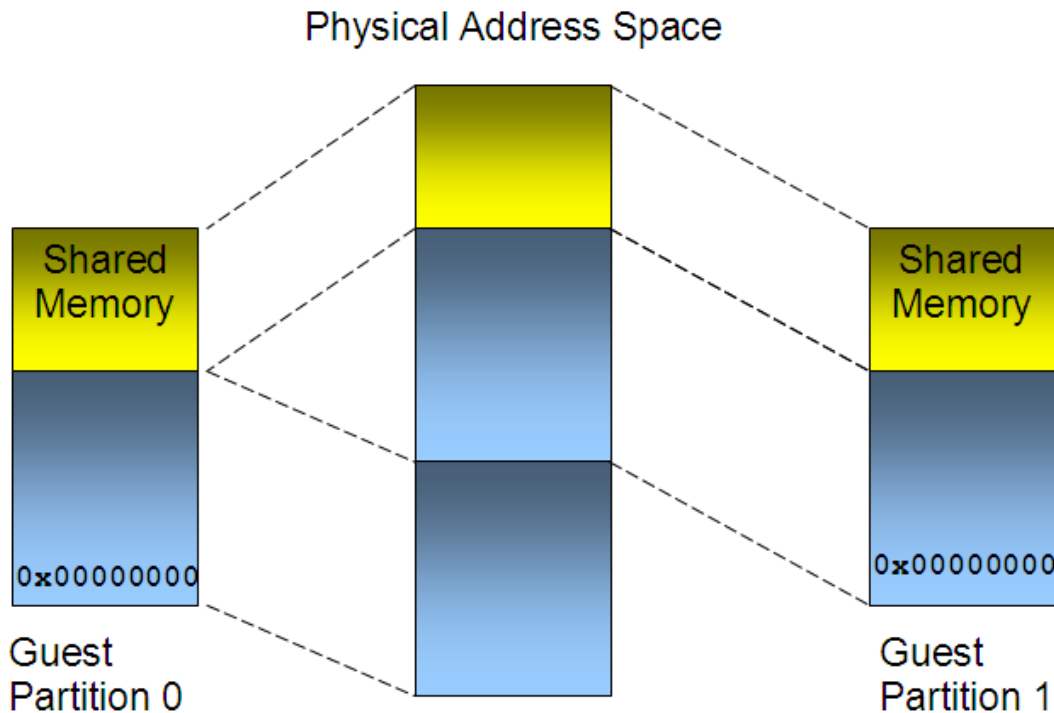


Figure 5. Mapped Shared Memory Areas

It is also important to understand that in [Figure 5](#), the guest physical addresses of the shared memory regions do not need to be the same across the guest partitions.

2.2.5 `fsl,hv-load-image-table`

The `fsl,hv-load-image` table property is used to specify one or more images to be loaded into the guest physical address space prior to starting the partition. The property is specified as an encoded array table with one or more rows. Each row consist of three cells: `<image-src-addr, image-gphys-addr, image-size>`

`image-src-addr` is a `<u64>` and is the source physical address of an image to be loaded into the partition. The image may be a binary image or may be in ePAPR ELF format.

`image-gphys-addr` is a `<u64>` and specifies the guest physical address within the partition's main memory segment. This must be a word aligned address.

`image-size` is a `<u32>` and specifies the size of the image in bytes. The image size can be zero if the image is an ELF image.

Because both the `image-src-addr` and `image-gphys-addr` items are 64-bit values two 32-bit integer values are required for each in order to express the full 64-bit value. In the following example the image located at true physical address `0xef200000` is loaded into the guest physical address location `0x00`. The size argument is zero, indicating that the image is an ELF image. One should also remember that the guest

physical location 0x00 is mapped from a location in the true physical address space by the *fsl,hv-physaddr-map* property described above.

Example 8. fsl,load-image-table

```
fsl,load-image-table = <0 ef200000 0 0 0>
```

2.3 Byte Channels

Byte channels are virtual 16 bit communications channels that can be used to communicate between partitions. Additionally, a byte channel can be associated with a serial console in order to provide an output mechanism for a partition. Byte channels are created in the master Hypervisor device tree by specifying a *byte-channels* node. The individual partitions make use of these byte channels by specifying a handle to the byte channel in the partition device tree.

Example 9. byte-channels Node

```
byte-channels {
    bc0: byte-channel0 {
        compatible = "fsl,hv-byte-channel";
        fsl,endpoint = <&uartmux>;
        fsl,mux-channel = <0>;
    };
}
```

2.4 Doorbell Interrupts

Another way to provide inter-partition communication is through the use of doorbell interrupts. As with byte channels a doorbell is specified in the master Hypervisor device tree. The individual partitions make use of these doorbells by specifying a send and/or a receive handle to the doorbell in the partition device tree.

Example 10. Doorbells Node in the Master Device Tree

```
doorbells {
    doorbell@0 {
        compatible = "fsl,hv-doorbell";
    };
};
```

3 Partition Device Tree

In addition to creating a master Hypervisor device tree that specifies the way that system resources are divided among the partitions, it is also necessary to create a device tree for each individual partition specifying how the available resources are utilized within that partition.

3.1 Root Node

The root node of the partition device tree is required and must have the following properties:

- *#address-cells*—Specifies the number of <u32> cells to represent the address in the reg property in children of root (must be 2).
- *#size-cells*—Specifies the number of <u32> cells to represent the size in the reg property in children of root (must be 1).
- *compatible*—Must be “fsl,hv-platform-p4080”, “fsl,hv-platform”
- *epapr-version*—Must be epapr-version = “ePAPR-1.0”

Example 11. Guest Device Tree Root Node

```
/ {
    compatible = "fsl,hv-platform-p4080",
                "fsl,hv-platform";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;
```

3.2 CPUs and CPU Nodes

Just as in the master Hypervisor device tree, the *cpus* node contains multiple *cpu* child nodes that define the properties of each of the CPUs available to the partition. These CPUs are made available to the partition in the master Hypervisor device tree partitions node.

From the perspective of the partition, these CPUs are contiguous. However, the actual physical CPUs being utilized may be non-contiguous. For example, the master Hypervisor tree may make physical CPUs 1, 3, and 5 available to a partition. However, from the perspective of the partition the available CPUs are viewed as CPUs 0, 1, and 2. Additionally, no CPUs other than those explicitly made available to a partition are visible from within the partition.

Example 12. CPU node

```
cpus {
    #address-cells = <1>;
    #size-cells = <0>;
```

Partition Device Tree

```

cpu@0 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <0>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;
    timebase-frequency = <0x83333333>;
    bus-frequency = <0x66666666>;
    clock-frequency = <0x66666666>;
};

};

```

3.3 Memory Node

The *memory* node specifies the physical address and size of memory belonging to the partition. The property is encoded as a pair of values consisting of a 64-bit address indicating the start of memory followed by a size argument. As with other unsigned 64-bit property values, two 32-bit integer values are used to comprise the complete 64-bit address. It should also be noted that the address is relative to the physical address within the partition and not to the true physical address from which the available partition memory is mapped.

Example 13. Memory Node

```

memory {
    reg = <0 0 0C000000>;
    device_type = "memory";
};

```

3.4 Byte Channels

The individual partitions make use of byte channels by specifying a handle within the *Hypervisor* node of the partition device tree. The *handles* node is a child node of the *Hypervisor* node. The *fsl,endpoint* property of the byte channel node ties the byte channel handle to the actual byte channel defined in the master device tree.

Example 14. Byte Channel Handle in the Partition Device Tree

```
hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel0";
            interrupt-parent = <&vmpic>;
        };
    };
};
```

3.5 Doorbell Interrupts

The individual partitions make use of doorbells by specifying a handle within the *Hypervisor* node of the partition device tree. The *handles* node is a child node of the *Hypervisor* node. The *fsl,endpoint* property of the doorbell node ties the doorbell handle to the actual doorbell defined in the master device tree.

Example 15. Doorbell Handle in the Partition Device Tree

```
hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;
        .
        .
        .
        doorbell@0 {
            compatible = "fsl,hv-doorbell-send-handle";
            fsl,endpoint = "/doorbells/doorbell@1";
        };
    };
};
```

```
};
```

4 Executing Under the Simics® Simulator

The Simics® simulator can be used to provide a model of the target multicore hardware so that development can proceed even before availability of actual hardware. The software and device tree files are identical to those that execute on the actual target hardware.

Simics uses a script file, typically with a `.simics` extension, to configure the simulation environment. There are several items that need to be placed into the `simics` script when executing Hypervisor/Lwe applications under the `simics` simulator.

4.1 Image files

The `simics` environment uses two arrays to specify image files and their locations in the simulated hardware memory, as follows:

- `guest_image[]`—This array specifies the binary image files on the host system containing the images to be loaded.
- `guest_addr[]`—This array specifies the physical addresses in the simulated hardware memory where the corresponding binaries specified in the `guest_image[]` array are loaded.

It is important to understand that the guest addresses specified should match the `fsl,hv-load-image-table` properties in the master Hypervisor device tree file.

Example 16. Image Arrays

```
$guest_image[0] = "myimage1.elf"  
$guest_addr[0]  = 0xe8700000  
$guest_image[1] = "myimage2.elf"  
$guest_addr[1]  = 0xe8800000
```

4.1 Device Tree File

The master Hypervisor device tree file must also be specified in the `.simics` script. This is done by setting the `dev_tree_blob` variable.

Example 17. device_tree_blob

```
$dev_tree_blob = "mydevicetree.dtb"
```

4.2 Hypervisor Kernel Image

In addition to having a pointer to the master device tree file, the simulator must know where the Hypervisor kernel is located. This is specified using the *kernel_image* variable in the .simics script.

Example 18. kernel_image

```
$kernel_image = "hv.uImage"
```

5 Revision History

Table 1 provides a revision history for this application note.

Table 1. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	10/2008	Initial public release.

Appendix A Example Device Tree File Set for an 8-Core, 8-Partition Implementation

A.1 Master Hypervisor Device Tree

```

/ {
    model = "fsl,MPC8578SIM";
    compatible = "fsl,MPC8578SIM";
    #address-cells = <2>;
    #size-cells = <1>;

    aliases {
        stdout = &uart0;
        stdin = &uart0;
        serial0 = &uart0;
        uartmux = &uartmux;
        byte-channel0 = &bc0;
        byte-channel1 = &bc1;
        byte-channel2 = &bc2;
        byte-channel3 = &bc3;
        byte-channel4 = &bc4;
        byte-channel5 = &bc5;
        byte-channel6 = &bc6;
    };

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
            d-cache-line-size = <20>;
            i-cache-line-size = <20>;
            d-cache-size = <8000>;
            i-cache-size = <8000>;
            timebase-frequency = <0x83333333>;
            bus-frequency = <0x66666666>;
            clock-frequency = <0x66666666>;
            status = "okay";
        };
        cpu@1 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <1>;
        };
    };
};

```

```

        d-cache-line-size = <20>;
        i-cache-line-size = <20>;
        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
        status = "disabled";
        enable-method = "spin-table";
};

cpu@2 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <2>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;
    timebase-frequency = <0x83333333>;
    bus-frequency = <0x66666666>;
    clock-frequency = <0x66666666>;
    status = "disabled";
    enable-method = "spin-table";
};

cpu@3 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <3>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;
    timebase-frequency = <0x83333333>;
    bus-frequency = <0x66666666>;
    clock-frequency = <0x66666666>;
    status = "disabled";
    enable-method = "spin-table";
};

cpu@4 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <4>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;

```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
        status = "disabled";
        enable-method = "spin-table";
    };

cpu@5 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <5>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;
    timebase-frequency = <0x83333333>;
    bus-frequency = <0x66666666>;
    clock-frequency = <0x66666666>;
    status = "disabled";
    enable-method = "spin-table";
};

cpu@6 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <6>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;
    timebase-frequency = <0x83333333>;
    bus-frequency = <0x66666666>;
    clock-frequency = <0x66666666>;
    status = "disabled";
    enable-method = "spin-table";
};

cpu@7 {
    device_type = "cpu";
    compatible = "fsl,e500mc", "fsl,e500";
    reg = <7>;
    d-cache-line-size = <20>;
    i-cache-line-size = <20>;
    d-cache-size = <8000>;
    i-cache-size = <8000>;

```

```

        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
        status = "disabled";
        enable-method = "spin-table";
    };
};

memory {
    reg = <0 0 100000>;
    device_type = "memory";
};

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "fsl,mpc8578-ccsr", "simple-bus";
    ranges = <0 0 fe000000 400000>;

    mpic: pic@40000 {
        clock-frequency = <0>;
        interrupt-controller;
        #address-cells = <0>;
        #interrupt-cells = <2>;
        reg = <40000 40000>;
        compatible = "fsl,mpc8578-mpic", "fsl,mpic",
            "chrp,open-pic";
        device_type = "open-pic";
        big-endian;
    };

    uart0: serial@11c500 {
        device_type = "serial";
        compatible = "fsl,mpc8578-uart", "fsl,ns16550", "ns16550";
        interrupts = <24 2>;
        interrupt-parent = <&mpic>;
        reg = <11c500 100>;
        clock-frequency = <0>;
    };

    uart1: serial@11c600 {
        device_type = "serial";
        compatible = "fsl,mpc8578-uart", "fsl,ns16550", "ns16550";
        interrupts = <24 2>;
        interrupt-parent = <&mpic>;
        reg = <11c600 100>;
        clock-frequency = <0>;
    };
};

```

```

};

global-utilities@e0000 { //global utilities block
    compatible = "fsl,mpc8572-guts";
    reg = <e0000 1000>;
    fsl,has-rstcr;
};

};

uartmux: uartmux {
    compatible = "fsl,hv-byte-channel-mux";
    fsl,phys-dev = <&uart1>;
};

byte-channels {
    bc0: byte-channel0 {
        compatible = "fsl,hv-byte-channel";
        fsl,endpoint = <&uartmux>;
        fsl,mux-channel = <0>;
    };

    bc1: byte-channell1 {
        compatible = "fsl,hv-byte-channel";
        fsl,endpoint = <&uartmux>;
        fsl,mux-channel = <1>;
    };

    bc2: byte-channel2 {
        compatible = "fsl,hv-byte-channel";
        fsl,endpoint = <&uartmux>;
        fsl,mux-channel = <2>;
    };

    bc3: byte-channel3 {
        compatible = "fsl,hv-byte-channel";
        fsl,endpoint = <&uartmux>;
        fsl,mux-channel = <3>;
    };

    bc4: byte-channel4 {
        compatible = "fsl,hv-byte-channel";
        fsl,endpoint = <&uartmux>;
        fsl,mux-channel = <4>;
    };

    bc5: byte-channel5 {

```

```

compatible = "fsl,hv-byte-channel";
fsl,endpoint = <&uartmux>;
fsl,mux-channel = <5>;
};

bc6: byte-channel6 {
compatible = "fsl,hv-byte-channel";
fsl,endpoint = <&uartmux>;
fsl,mux-channel = <6>;
};

bc7: byte-channel7 {
compatible = "fsl,hv-byte-channel";
fsl,endpoint = <&uartmux>;
fsl,mux-channel = <7>;
};
};

doorbells {
doorbell@0 {
compatible = "fsl,hv-doorbell";
};
doorbell@1 {
compatible = "fsl,hv-doorbell";
};
doorbell@2 {
compatible = "fsl,hv-doorbell";
};
doorbell@3 {
compatible = "fsl,hv-doorbell";
};
doorbell@4 {
compatible = "fsl,hv-doorbell";
};
doorbell@5 {
compatible = "fsl,hv-doorbell";
};
doorbell@6 {
compatible = "fsl,hv-doorbell";
};
doorbell@7 {
compatible = "fsl,hv-doorbell";
};
};

partitions {

```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

part1 {
    compatible = "fsl,hv-partition";
    fsl,hv-cpus = <0 1>;
    fsl,entry = <0>;
    fsl,privileged;
    fsl,hv-dtb = /incbin/("linux-p1.dtb");
    fsl,hv-physaddr-map = <0 0 0 1000000 17000000
                          0 17000000 0 20000000 1000000
                          0 fe040000 0 fe040000 20000
                          0 fe060000 0 fe060000 1000
                          0 fe061000 0 fe062000 1000
                          0 fe062000 ffffffff ffffffff 1e000>;
    fsl,hv-load-image-table = <0 ef100000 0 0 0
                              0 ef600000 0 01300000
00100000>;
    fsl,hv-dtb-window = <0 01000000 10000>;
};

part2 {
    compatible = "fsl,hv-partition";
    fsl,hv-cpus = <1 1>;
    fsl,entry = <0>;
    fsl,hv-dtb = /incbin/("lwe-p2.dtb");
    fsl,hv-physaddr-map = <0 0 0 21000000 04000000
                          0 04000000 0 64000000 04000000
                          0 08000000 0 6C000000 04000000>;
    fsl,hv-load-image-table = <0 ef700000 0 0 0>;
    fsl,hv-dtb-window = <0 01000000 10000>;
};

part3 {
    compatible = "fsl,hv-partition";
    fsl,hv-cpus = <2 1>;
    fsl,entry = <0>;
    fsl,hv-dtb = /incbin/("lwe-p3.dtb");
    fsl,hv-physaddr-map = <0 0 0 25000000 04000000
                          0 04000000 0 64000000 04000000
                          0 08000000 0 6C000000 04000000>;
    fsl,hv-load-image-table = <0 ef800000 0 0 0>;
    fsl,hv-dtb-window = <0 01000000 10000>;
};

part4 {
    compatible = "fsl,hv-partition";
    fsl,hv-cpus = <3 1>;
    fsl,entry = <0>;
    fsl,hv-dtb = /incbin/("lwe-p4.dtb");

```



```

fsl,hv-physaddr-map = <0 0 0 29000000 04000000
                        0 04000000 0 64000000 04000000
                        0 08000000 0 6C000000 04000000>;
fsl,hv-load-image-table = <0 ef900000 0 0 0>;
fsl,hv-dtb-window = <0 01000000 10000>;
};

part5 {
compatible = "fsl,hv-partition";
fsl,hv-cpus = <4 1>;
fsl,entry = <0>;
fsl,hv-dtb = /incbin/("lwe-p5.dtb");
fsl,hv-physaddr-map = <0 0 0 2d000000 04000000
                        0 04000000 0 64000000 04000000
                        0 08000000 0 6C000000 04000000>;
fsl,hv-load-image-table = <0 efa00000 0 0 0>;
fsl,hv-dtb-window = <0 01000000 10000>;
};

part6 {
compatible = "fsl,hv-partition";
fsl,hv-cpus = <5 1>;
fsl,entry = <0>;
fsl,hv-dtb = /incbin/("lwe-p6.dtb");
fsl,hv-physaddr-map = <0 0 0 41000000 04000000
                        0 04000000 0 64000000 04000000
                        0 08000000 0 6C000000 04000000>;
fsl,hv-load-image-table = <0 efb00000 0 0 0>;
fsl,hv-dtb-window = <0 01000000 10000>;
};

part7 {
compatible = "fsl,hv-partition";
fsl,hv-cpus = <6 1>;
fsl,entry = <0>;
fsl,hv-dtb = /incbin/("lwe-p7.dtb");
fsl,hv-physaddr-map = <0 0 0 45000000 04000000
                        0 04000000 0 64000000 04000000
                        0 08000000 0 6C000000 04000000>;
fsl,hv-load-image-table = <0 efc00000 0 0 0>;
fsl,hv-dtb-window = <0 01000000 10000>;
};

part8 {
compatible = "fsl,hv-partition";
fsl,hv-cpus = <7 1>;
fsl,entry = <0>;

```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

    fsl,hv-dtb = /incbin/("lwe-p8.dtb");
    fsl,hv-physaddr-map = <0 0 0 49000000 04000000
                          0 04000000 0 64000000 04000000
                          0 08000000 0 6C000000 04000000>;
    fsl,hv-load-image-table = <0 efd00000 0 0 0>;
    fsl,hv-dtb-window = <0 01000000 10000>;
};
};

chosen {
    fsl,hv-pic-coreint = <0>;
};
};

```

A.2 Partition 1 Device Tree

```

/ {
    compatible = "fsl,hv-platform-mpc8578",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-pl1";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    memory {
        reg = <0 0 18000000>;
        device_type = "memory";
    };

    ccsr@fe000000 {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "simple-bus";
        device_type = "soc";
        ranges = <0 0 fe000000 400000>;

        i2c0: i2c@118000 {
            device_type = "i2c";
            compatible = "fsl,mpc8578-i2c",
                        "fsl,i2c", "fsl-i2c";
            reg = <118000 100>;
            interrupts = <26 2>;
            interrupt-parent = <&mpic>;
            dfsrr;
        };
    };
};

```

```

serial@11d500 {          // UART2
    device_type = "serial";
    compatible = "fsl,mpc8578-uart",
                 "fsl,ns16550", "ns16550";
    reg = <11d500 100>;
    clock-frequency = <0>;
    interrupts = <25 2>;
    interrupt-parent = <&mpic>;
};

mpic: pic@40000 {
    clock-frequency = <0>;
    interrupt-controller;
    #address-cells = <0>;
    #interrupt-cells = <2>;
    reg = <40000 40000>;
    compatible = "fsl,mpc8578-mpic", "fsl,mpic",
                 "chrp,open-pic";
    device_type = "open-pic";
    big-endian;
    fsl,hv-interrupt-controller;
};

qman: qman@318000 {
    compatible = "fsl,mpc8578-qman";
    reg = <318000 1000>;
    interrupt-parent = <&mpic>;
    interrupts = <70 2>;
};

bman: bman@31a000 {
    compatible = "fsl,mpc8578-bman";
    reg = <31a000 1000>;
    interrupt-parent = <&mpic>;
    interrupts = <74 2>;
};

};

qman-portal@0 {
    compatible = "fsl,qman-portal";
    reg = <0 e4200000 4000 0 e4300000 1000>;
    /*
    interrupts = <28 2>;
    interrupt-parent = <&mpic>;
    stashing-dest = <&cpu0>;
    */
};

bman-portal@0 {

```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

        compatible = "fsl,bman-portal";
        reg = <0 e4000000 4000 0 e4100000 1000>;
        /*
        interrupts = <28 2>;
        interrupt-parent = <&mpic>;
        stashing-dest = <&cpu0>;
        */
    };

    vmpic:vmpic {
        compatible = "fsl,hv-vmpic";
        interrupt-controller;
        #interrupt-cells = <2>;
        fsl,hv-phandle-ref = <&vmpic>;
    };

    hypervisor {
        handles {
            #address-cells = <1>;
            #size-cells = <0>;
            compatible = "fsl,hv-handles";
            interrupt-parent = <&vmpic>;

            gpio {
                reg = <1>;
                compatible = "fsl,hv-gpio-handle";
                fsl,real-dev = "gpio";
                fsl,bitmask = <103ecc00>;
            };
        };
    };

    chosen {
        linux,stdout-path = "/ccsr@fe000000/serial@11d500";
        bootargs = "console=ttyS0,115200";
        linux,initrd-start = <01300000>;
        linux,initrd-end = <01700000>;
    };

    aliases {
        stdout = "/ccsr/serial@11d500";
    };
};

```

A.3 Partition 2 Device Tree

```

/ {
    compatible = "fsl,hv-platform-p4080",
                 "fsl,hv-platform",
                 "fsl,MPC8578SIM";
    model = "fsl,partman-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
            d-cache-line-size = <20>;
            i-cache-line-size = <20>;
            d-cache-size = <8000>;
            i-cache-size = <8000>;
            timebase-frequency = <0x83333333>;
            bus-frequency = <0x66666666>;
            clock-frequency = <0x66666666>;
        };
    };

    memory {
        reg = <0 0 0C000000>;
        device_type = "memory";
    };

    shmem_heap {
        reg = <0 60000000 4000000>;
        compatible = "fsl,share-mem-heap";
    };

    shmem_heap_guarded {
        reg = <0 68000000 4000000>;
        compatible = "fsl,share-mem-heap-guarded";
    };

    ccsr@fe000000 {
    
```

```

#address-cells = <1>;
#size-cells = <1>;
compatible = "simple-bus";
device_type = "soc";
ranges = <0 0 fe000000 400000>;

serial@11d600 {          // UART3
    device_type = "serial";
    compatible = "fsl,mpc8578-uart",
                 "fsl,ns16550", "ns16550";
    reg = <11d600 100>;
    clock-frequency = <0>;
    interrupts = <25 2>;
    interrupt-parent = <&mpic>;
};

gtbcr@410f0 {
    compatible = "fsl,mpc8578-gtbcr";
    reg = <410f0 300>;
    clock-frequency = <0>;
    interrupt-parent = <&mpic>;
};

mpic: pic@40000 {
    clock-frequency = <0>;
    interrupt-controller;
    #address-cells = <0>;
    #interrupt-cells = <2>;
    reg = <40000 40000>;
    compatible = "fsl,mpc8578-mpic", "fsl,mpic",
                 "chrp,open-pic";
    device_type = "open-pic";
    big-endian;
    fsl,hv-interrupt-controller;
};

};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;

```

```

#size-cells = <0>;
compatible = "fsl,hv-handles";
interrupt-parent = <&vmpic>;

byte-channel@0 {
    compatible = "fsl,hv-byte-channel-handle";
    fsl,endpoint = "byte-channel0";
    interrupt-parent = <&vmpic>;
};

gpio {
    reg = <1>;
    compatible = "fsl,hv-gpio-handle";
    fsl,real-dev = "gpio";
    fsl,bitmask = <020033ff>;
};

doorbell@0 {
    compatible = "fsl,hv-doorbell-send-handle";
    fsl,endpoint = "/doorbells/doorbell@1";
};
doorbell@1 {
    compatible = "fsl,hv-doorbell-receive-handle";
    fsl,endpoint = "/doorbells/doorbell@1";
};
doorbell@2 {
    compatible = "fsl,hv-doorbell-send-handle";
    fsl,endpoint = "/doorbells/doorbell@2";
};
doorbell@3 {
    compatible = "fsl,hv-doorbell-send-handle";
    fsl,endpoint = "/doorbells/doorbell@3";
};
doorbell@4 {
    compatible = "fsl,hv-doorbell-send-handle";
    fsl,endpoint = "/doorbells/doorbell@4";
};
doorbell@5 {
    compatible = "fsl,hv-doorbell-send-handle";
    fsl,endpoint = "/doorbells/doorbell@5";
};
doorbell@6 {
    compatible = "fsl,hv-doorbell-send-handle";
    fsl,endpoint = "/doorbells/doorbell@6";
};
doorbell@7 {
    compatible = "fsl,hv-doorbell-send-handle";

```

```

        fsl,endpoint = "/doorbells/doorbell@7";
    };

};

chosen {
    linux,stdout-path = "/ccsr@fe000000/serial@11d600";
    bootargs = "console=ttyS0,115200";
    linux,initrd-start = <01300000>;
    linux,initrd-end = <01400000>;
    fsl,partition = "partition2";
};

aliases {
    stdout = "/hypervisor/handles/byte-channel@0";
    uart3 = "/ccsr/serial@11d600";
};

bman-portal {
    compatible = "fsl,bman-portal";
    reg = <0 e4008000 4000 0 e4102000 1000>;
};

qman-portal {
    compatible = "fsl,qman-portal";
    reg = <0 e4208000 4000 0 e4302000 1000>;
};
};
};

```

A.4 Partition 3 Device Tree

```

/ {
    compatible = "fsl,hv-platform-p4080",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {

```



```

        device_type = "cpu";
        compatible = "fsl,e500mc", "fsl,e500";
        reg = <0>;
        d-cache-line-size = <20>;
        i-cache-line-size = <20>;
        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
    };

};

memory {
    reg = <0 0 0C000000>;
    device_type = "memory";
};

shmem_heap {
    reg = <0 60000000 4000000>;
    compatible = "fsl,share-mem-heap";
};

shmem_heap_guarded {
    reg = <0 68000000 4000000>;
    compatible = "fsl,share-mem-heap-guarded";
};

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    device_type = "soc";
    ranges = <0 0 fe000000 400000>;

    serial@11d600 {          // UART3
        device_type = "serial";
        compatible = "fsl,mpc8578-uart",
            "fsl,ns16550", "ns16550";
        reg = <11d600 100>;
        clock-frequency = <0>;
        interrupts = <25 2>;
        interrupt-parent = <&mpic>;
    };
};

```

```

gtbcr@410f0 {
    compatible = "fsl,mpc8578-gtbcr";
    reg = <410f0 300>;
    clock-frequency = <0>;
    interrupt-parent = <&mpic>;
};

mpic: pic@40000 {
    clock-frequency = <0>;
    interrupt-controller;
    #address-cells = <0>;
    #interrupt-cells = <2>;
    reg = <40000 40000>;
    compatible = "fsl,mpc8578-mpic", "fsl,mpic",
                "chrp,open-pic";
    device_type = "open-pic";
    big-endian;
    fsl,hv-interrupt-controller;
};

};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel1";
        };

        gpio {
            reg = <1>;
            compatible = "fsl,hv-gpio-handle";
            fsl,real-dev = "gpio";
            fsl,bitmask = <020033ff>;
        };
    };
};

```

```

door-bell@0 {
    compatible = "fsl,hv-doorbell-receive-handle";
    fsl,endpoint = "/doorbells/doorbell@2";
};

};

};

chosen {
    fsl,partition = "partition3";
};

aliases {
    stdout = "/hypervisor/handles/byte-channel@0";
};

bman-portal {
    compatible = "fsl,bman-portal";
    reg = <0 e400c000 4000 0 e4103000 1000>;
};

qman-portal {
    compatible = "fsl,qman-portal";
    reg = <0 e420c000 4000 0 e4303000 1000>;
};
};
    
```

A.5 Partition 4 Device Tree

```

/ {
    compatible = "fsl,hv-platform-mpc8578",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
        }
    }
}
    
```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

        d-cache-line-size = <20>;
        i-cache-line-size = <20>;
        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
    };

};

memory {
    reg = <0 0 0C000000>;
    device_type = "memory";
};

shmem_heap {
    reg = <0 60000000 4000000>;
    compatible = "fsl,share-mem-heap";
};

shmem_heap_guarded {
    reg = <0 68000000 4000000>;
    compatible = "fsl,share-mem-heap-guarded";
};

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    device_type = "soc";
    ranges = <0 0 fe000000 400000>;

    serial@11d600 {          // UART3
        device_type = "serial";
        compatible = "fsl,mpc8578-uart",
            "fsl,ns16550", "ns16550";
        reg = <11d600 100>;
        clock-frequency = <0>;
        interrupts = <25 2>;
        interrupt-parent = <&mpic>;
    };

    gtbcr@410f0 {
        compatible = "fsl,mpc8578-gtbcr";
        reg = <410f0 300>;
    };
};

```

```

        clock-frequency = <0>;
        interrupt-parent = <&mpic>;
    };

    mpic: pic@40000 {
        clock-frequency = <0>;
        interrupt-controller;
        #address-cells = <0>;
        #interrupt-cells = <2>;
        reg = <40000 40000>;
        compatible = "fsl,mpc8578-mpic", "fsl,mpic",
            "chrp,open-pic";
        device_type = "open-pic";
        big-endian;
        fsl,hv-interrupt-controller;
    };
};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel2";
        };

        gpio {
            reg = <1>;
            compatible = "fsl,hv-gpio-handle";
            fsl,real-dev = "gpio";
            fsl,bitmask = <020033ff>;
        };

        door-bell@0 {
            compatible = "fsl,hv-doorbell-receive-handle";
        };
    };
};

```

```

                fsl,endpoint = "/doorbells/doorbell@3";
            };

        };
    };

    chosen {
        fsl,partition = "partition4";
    };

    aliases {
        stdout = "/hypervisor/handles/byte-channel@0";
    };

    bman-portal {
        compatible = "fsl,bman-portal";
        reg = <0 e4010000 4000 0 e4104000 1000>;
    };

    qman-portal {
        compatible = "fsl,qman-portal";
        reg = <0 e4210000 4000 0 e4304000 1000>;
    };
};

```

A.6 Partition 5 Device Tree

```

/ {
    compatible = "fsl,hv-platform-p4080",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
            d-cache-line-size = <20>;
            i-cache-line-size = <20>;
        };
    };
};

```

```

        d-cache-size = <8000>;
        i-cache-size = <8000>;
        timebase-frequency = <0x83333333>;
        bus-frequency = <0x66666666>;
        clock-frequency = <0x66666666>;
    };

};

memory {
    reg = <0 0 0C000000>;
    device_type = "memory";
};

shmem_heap {
    reg = <0 60000000 4000000>;
    compatible = "fsl,share-mem-heap";
};

shmem_heap_guarded {
    reg = <0 68000000 4000000>;
    compatible = "fsl,share-mem-heap-guarded";
};

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    device_type = "soc";
    ranges = <0 0 fe000000 400000>;

    serial@11d600 {          // UART3
        device_type = "serial";
        compatible = "fsl,mpc8578-uart",
            "fsl,ns16550", "ns16550";
        reg = <11d600 100>;
        clock-frequency = <0>;
        interrupts = <25 2>;
        interrupt-parent = <&mpic>;
    };

    gtbcrcr@410f0 {
        compatible = "fsl,mpc8578-gtbcrcr";
        reg = <410f0 300>;
        clock-frequency = <0>;
        interrupt-parent = <&mpic>;
    };
};

```

```

};

mpic: pic@40000 {
    clock-frequency = <0>;
    interrupt-controller;
    #address-cells = <0>;
    #interrupt-cells = <2>;
    reg = <40000 40000>;
    compatible = "fsl,mpc8578-mpic", "fsl,mpic",
                "chrp,open-pic";
    device_type = "open-pic";
    big-endian;
    fsl,hv-interrupt-controller;
};

};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel3";
        };

        gpio {
            reg = <1>;
            compatible = "fsl,hv-gpio-handle";
            fsl,real-dev = "gpio";
            fsl,bitmask = <020033ff>;
        };

        door-bell@0 {
            compatible = "fsl,hv-doorbell-receive-handle";
            fsl,endpoint = "/doorbells/doorbell@4";
        };
    };
};

```



```

    };
};

chosen {
    fsl,partition = "partition5";
};

aliases {
    stdout = "/hypervisor/handles/byte-channel@0";
};

bman-portal {
    compatible = "fsl,bman-portal";
    reg = <0 e4014000 4000 0 e4105000 1000>;
};

qman-portal {
    compatible = "fsl,qman-portal";
    reg = <0 e4214000 4000 0 e4305000 1000>;
};
};

```

A.7 Partition 6 Device Tree

```

/ {
    compatible = "fsl,hv-platform-p4080",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
            d-cache-line-size = <20>;
            i-cache-line-size = <20>;
            d-cache-size = <8000>;
            i-cache-size = <8000>;
            timebase-frequency = <0x83333333>;
            bus-frequency = <0x66666666>;
        }
    }
}

```

```

        clock-frequency = <0x66666666>;
    };

};

memory {
    reg = <0 0 0C000000>;
    device_type = "memory";
};

shmem_heap {
    reg = <0 60000000 4000000>;
    compatible = "fsl,share-mem-heap";
};

shmem_heap_guarded {
    reg = <0 68000000 4000000>;
    compatible = "fsl,share-mem-heap-guarded";
};

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    device_type = "soc";
    ranges = <0 0 fe000000 400000>;

    serial@11d600 {          // UART3
        device_type = "serial";
        compatible = "fsl,mpc8578-uart",
            "fsl,ns16550", "ns16550";
        reg = <11d600 100>;
        clock-frequency = <0>;
        interrupts = <25 2>;
        interrupt-parent = <&mpic>;
    };

    gtbcr@410f0 {
        compatible = "fsl,mpc8578-gtbcr";
        reg = <410f0 300>;
        clock-frequency = <0>;
        interrupt-parent = <&mpic>;
    };

    mpic: pic@40000 {
        clock-frequency = <0>;
    };
};

```

```

        interrupt-controller;
        #address-cells = <0>;
        #interrupt-cells = <2>;
        reg = <40000 40000>;
        compatible = "fsl,mpc8578-mpic", "fsl,mpic",
                    "chrp,open-pic";
        device_type = "open-pic";
        big-endian;
        fsl,hv-interrupt-controller;
    };
};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel4";
        };

        gpio {
            reg = <1>;
            compatible = "fsl,hv-gpio-handle";
            fsl,real-dev = "gpio";
            fsl,bitmask = <020033ff>;
        };

        door-bell@0 {
            compatible = "fsl,hv-doorbell-receive-handle";
            fsl,endpoint = "/doorbells/doorbell@5";
        };

    };
};
};

```

```

chosen {
    fsl,partition = "partition6";
};

aliases {
    stdout = "/hypervisor/handles/byte-channel@0";
};

bman-portal {
    compatible = "fsl,bman-portal";
    reg = <0 e4018000 4000 0 e4106000 1000>;
};

qman-portal {
    compatible = "fsl,qman-portal";
    reg = <0 e4218000 4000 0 e4306000 1000>;
};
};

```

A.8 Partition 7 Device Tree

```

/ {
    compatible = "fsl,hv-platform-p4080",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
            d-cache-line-size = <20>;
            i-cache-line-size = <20>;
            d-cache-size = <8000>;
            i-cache-size = <8000>;
            timebase-frequency = <0x83333333>;
            bus-frequency = <0x66666666>;
            clock-frequency = <0x66666666>;
        };
    };
};

```

```

};

memory {
    reg = <0 0 0C000000>;
    device_type = "memory";
};

shmem_heap {
    reg = <0 60000000 4000000>;
    compatible = "fsl,share-mem-heap";
};

shmem_heap_guarded {
    reg = <0 68000000 4000000>;
    compatible = "fsl,share-mem-heap-guarded";
};

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    device_type = "soc";
    ranges = <0 0 fe000000 400000>;

    serial@11d600 {          // UART3
        device_type = "serial";
        compatible = "fsl,mpc8578-uart",
            "fsl,ns16550", "ns16550";
        reg = <11d600 100>;
        clock-frequency = <0>;
        interrupts = <25 2>;
        interrupt-parent = <&mpic>;
    };

    gtbcrc@410f0 {
        compatible = "fsl,mpc8578-gtbcrc";
        reg = <410f0 300>;
        clock-frequency = <0>;
        interrupt-parent = <&mpic>;
    };

    mpic: pic@40000 {
        clock-frequency = <0>;
        interrupt-controller;
        #address-cells = <0>;
        #interrupt-cells = <2>;
    };

```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

        reg = <40000 40000>;
        compatible = "fsl,mpc8578-mpic", "fsl,mpic",
                    "chrp,open-pic";
        device_type = "open-pic";
        big-endian;
        fsl,hv-interrupt-controller;
    };
};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel5";
        };

        gpio {
            reg = <1>;
            compatible = "fsl,hv-gpio-handle";
            fsl,real-dev = "gpio";
            fsl,bitmask = <020033ff>;
        };

        door-bell@0 {
            compatible = "fsl,hv-doorbell-receive-handle";
            fsl,endpoint = "/doorbells/doorbell@6";
        };
    };
};

chosen {
    fsl,partition = "partition7";
};

```

```

aliases {
    stdout = "/hypervisor/handles/byte-channel@0";
};

bman-portal {
    compatible = "fsl,bman-portal";
    reg = <0 e401c000 4000 0 e4107000 1000>;
};

qman-portal {
    compatible = "fsl,qman-portal";
    reg = <0 e421c000 4000 0 e4307000 1000>;
};
};

```

A.9 Partition 8 Device Tree

```

/ {
    compatible = "fsl,hv-platform-p4080",
                "fsl,hv-platform",
                "fsl,MPC8578SIM";
    model = "fsl,hv-test-linux-p2";
    epapr-version = "ePAPR-1.0";
    #address-cells = <2>;
    #size-cells = <1>;

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "fsl,e500mc", "fsl,e500";
            reg = <0>;
            d-cache-line-size = <20>;
            i-cache-line-size = <20>;
            d-cache-size = <8000>;
            i-cache-size = <8000>;
            timebase-frequency = <0x83333333>;
            bus-frequency = <0x66666666>;
            clock-frequency = <0x66666666>;
        };
    };
};

memory {
    reg = <0 0 04000000>;
};

```

```

        device_type = "memory";
    };

    shmem_heap {
        reg = <0 60000000 4000000>;
        compatible = "fsl,share-mem-heap";
    };

    shmem_heap_guarded {
        reg = <0 68000000 4000000>;
        compatible = "fsl,share-mem-heap-guarded";
    };

ccsr@fe000000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    device_type = "soc";
    ranges = <0 0 fe000000 400000>;

    serial@11d600 {          // UART3
        device_type = "serial";
        compatible = "fsl,mpc8578-uart",
            "fsl,ns16550", "ns16550";
        reg = <11d600 100>;
        clock-frequency = <0>;
        interrupts = <25 2>;
        interrupt-parent = <&mpic>;
    };

    gtbcr@410f0 {
        compatible = "fsl,mpc8578-gtbcr";
        reg = <410f0 300>;
        clock-frequency = <0>;
        interrupt-parent = <&mpic>;
    };

mpic: pic@40000 {
    clock-frequency = <0>;
    interrupt-controller;
    #address-cells = <0>;
    #interrupt-cells = <2>;
    reg = <40000 40000>;
    compatible = "fsl,mpc8578-mpic", "fsl,mpic",
        "chrp,open-pic";
    device_type = "open-pic";
}

```



```

        big-endian;
        fsl,hv-interrupt-controller;
    };
};

vmpic:vmpic {
    compatible = "fsl,hv-vmpic";
    interrupt-controller;
    #interrupt-cells = <2>;
    fsl,hv-phandle-ref = <&vmpic>;
};

hypervisor {
    handles {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,hv-handles";
        interrupt-parent = <&vmpic>;

        byte-channel@0 {
            compatible = "fsl,hv-byte-channel-handle";
            fsl,endpoint = "byte-channel6";
        };

        gpio {
            reg = <1>;
            compatible = "fsl,hv-gpio-handle";
            fsl,real-dev = "gpio";
            fsl,bitmask = <020033ff>;
        };

        door-bell@0 {
            compatible = "fsl,hv-doorbell-receive-handle";
            fsl,endpoint = "/doorbells/doorbell@7";
        };
    };
};

chosen {
    fsl,partition = "partition8";
};

aliases {
    stdout = "/hypervisor/handles/byte-channel@0";
};

```

Example Device Tree File Set for an 8-Core, 8-Partition Implementation

```

bman-portal {
    compatible = "fsl,bman-portal";
    reg = <0 e4020000 4000 0 e4108000 1000>;
};

qman-portal {
    compatible = "fsl,qman-portal";
    reg = <0 e4220000 4000 0 e4308000 1000>;
};
};

```

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
 Technical Information Center, EL516
 2100 East Elliot Road
 Tempe, Arizona 85284
 1-800-521-6274 or
 +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
 Technical Information Center
 Schatzbogen 7
 81829 Muenchen, Germany
 +44 1296 380 456 (English)
 +46 8 52200080 (English)
 +49 89 92103 559 (German)
 +33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
 Headquarters
 ARCO Tower 15F
 1-8-1, Shimo-Meguro, Meguro-ku
 Tokyo 153-0064
 Japan
 0120 191014 or
 +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
 Exchange Building 23F
 No. 118 Jianguo Road
 Chaoyang District
 Beijing 100022
 China
 +86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor
 Literature Distribution Center
 P.O. Box 5405
 Denver, Colorado 80217
 1-800 441-2447 or
 +1-303-675-2140
 Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale are trademarks or registered trademarks of Freescale Semiconductor, Inc. in the U.S. and other countries. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© Freescale Semiconductor, Inc., 2008. All rights reserved.

