

IPC Resource customTable

Contents

Introduction

Config

C Code

New TYPE_DEVMEM entry

Introduction

The IPC 3.x product introduced support for loading slave images which include a [Linux-defined resource table](http://lwn.net/Articles/489009/) (<http://lwn.net/Articles/489009/>). In IPC 3.00.01, IPC added the ability (<http://git.ti.com/cgiit/cgiit.cgi/ipc/ipcdev.git/commit/?h=ipc-next&id=44169db6cd6f02193acf7ee97c13f956f599431d>) for users to override the default resource table with their own. This article describes the mechanics involved to do that.

NOTE

Although the Linux community defined the resource table, the IPC port to QNX also uses the same resource table.

NOTE

This is an expert technique. Most users do not need to modify the default resource table.

Config

To indicate you want to provide your own resource table, you need to set the `Resource.customTable` config parameter true.

```
<syntaxhighlight lang='javascript'> /* Override the default resource table with my own */ var Resource = xdc.useModule('ti.ipc.remoteproc.Resource'); Resource.customTable = true;
</syntaxhighlight>
```

When `Resource.customtable` is set to true, IPC will no longer generate a default table, and the user will be able to supply their own table using a specially-named C struct, `ti_ipc_remoteproc_ResourceTable`.

C Code

The user-supplied resource table is a C structure named `ti_ipc_remoteproc_ResourceTable` that needs to be linked into the slave executable. The platform-specific default resource tables are provided in IPC's **packages/ti/ip/remoteproc/rsc_table_*.h**, and are the recommended place to start from.

After finding the default resource table for your platform, you can copy the entirety of it into your own C file, compile it, and link it into your executable.

NOTE

When configuring in a `TYPE_TRACE` element (a trace buffer) you must explicitly declare the symbol as an extern (as below). The trace buffer is generated during config, and unfortunately there is no header to `#include` that declares the name of this symbol. Also note the size of the trace buffer declared in the resource table must match the size you've specified in your `.cfg` script.

For reference, here is an Vayu compatible resource table derived from **packages/ti/ip/remoteproc/rsc_table_vayu_ipu.h** (Disclaimer: This is a reference design only, and it is subject to enhancements, improvements and other changes without notifications).

```
<syntaxhighlight lang='c'>
```

```
1. include "rsc_types.h"

/* IPU Memory Map */

1. define L4_DRA7XX_BASE 0x4A000000

/* L4_CFG & L4_WKUP */

1. define L4_PERIPHERAL_L4CFG (L4_DRA7XX_BASE)
2. define IPU_PERIPHERAL_L4CFG 0x6A000000

1. define L4_PERIPHERAL_L4PER1 0x48000000
2. define IPU_PERIPHERAL_L4PER1 0x68000000

1. define L4_PERIPHERAL_L4PER2 0x48400000
2. define IPU_PERIPHERAL_L4PER2 0x68400000

1. define L4_PERIPHERAL_L4PER3 0x48800000
2. define IPU_PERIPHERAL_L4PER3 0x68800000

1. define L4_PERIPHERAL_L4EMU 0x54000000
2. define IPU_PERIPHERAL_L4EMU 0x74000000

1. define L3_PERIPHERAL_PRUSS 0x4B200000
2. define IPU_PERIPHERAL_PRUSS 0x6B200000
```

```

1. define L3_PERIPHERAL_DMM 0x4E000000
2. define IPU_PERIPHERAL_DMM 0x6E000000

1. define L3_IVAHD_CONFIG 0x5A000000
2. define IPU_IVAHD_CONFIG 0x7A000000

1. define L3_IVAHD_SL2 0x5B000000
2. define IPU_IVAHD_SL2 0x7B000000

1. define L3_TILER_MODE_0_1 0x60000000
2. define IPU_TILER_MODE_0_1 0xA0000000

1. define L3_TILER_MODE_2 0x70000000
2. define IPU_TILER_MODE_2 0xB0000000

1. define L3_TILER_MODE_3 0x78000000
2. define IPU_TILER_MODE_3 0xB8000000

1. define L3_OCMC_RAM 0x40300000
2. define IPU_OCMC_RAM 0x60300000

1. define L3_EMIF_SDRAM 0xA0000000
2. define IPU_EMIF_SDRAM 0x10000000

1. define IPU_MEM_TEXT 0x0
2. define IPU_MEM_DATA 0x80000000

1. define IPU_MEM_IOBUFS 0x90000000

1. define IPU_MEM_IPC_DATA 0x9F000000
2. define IPU_MEM_IPC_VRING 0x60000000
3. define IPU_MEM_RPMSG_VRING0 0x60000000
4. define IPU_MEM_RPMSG_VRING1 0x60040000
5. define IPU_MEM_VRING_BUFS0 0x60040000
6. define IPU_MEM_VRING_BUFS1 0x60080000

1. define IPU_MEM_IPC_VRING_SIZE SZ_1M
2. define IPU_MEM_IPC_DATA_SIZE SZ_1M

1. if defined(VAYU_IPU_1)
2. define IPU_MEM_TEXT_SIZE (SZ_1M)
3. elif defined(VAYU_IPU_2)
4. define IPU_MEM_TEXT_SIZE (SZ_1M * 6)
5. endif

1. if defined(VAYU_IPU_1)
2. define IPU_MEM_DATA_SIZE (SZ_1M * 5)
3. elif defined(VAYU_IPU_2)
4. define IPU_MEM_DATA_SIZE (SZ_1M * 48)
5. endif

1. define IPU_MEM_IOBUFS_SIZE (SZ_1M * 90)

```

```
/*
```

```

* Assign fixed RAM addresses to facilitate a fixed MMU table.
* PHYS_MEM_IPC_VRING & PHYS_MEM_IPC_DATA MUST be together.
*/

```

```
/* See CMA BASE addresses in Linux side: arch/arm/mach-omap2/remoteproc.c */
```

```

1. if defined(VAYU_IPU_1)
2. define PHYS_MEM_IPC_VRING 0x9D000000
3. elif defined (VAYU_IPU_2)
4. define PHYS_MEM_IPC_VRING 0x95800000
5. endif

1. define PHYS_MEM_IOBUFS 0xBA300000

```

```
/*
```

```

* Sizes of the virtqueues (expressed in number of buffers supported,
* and must be power of 2)
*/

```

```

1. define IPU_RPMSG_VQ0_SIZE 256
2. define IPU_RPMSG_VQ1_SIZE 256

```

```
/* flip up bits whose indices represent features we support */
```

```
1. define RPMSG_IPU_C0_FEATURES 1
```

```
struct my_resource_table {
```

```
    struct resource_table base;
```

```
    UInt32 offset[21]; /* Should match 'num' in actual definition */
```

```
/* rpmsg vdev entry */
struct fw_rsc_vdev rpmsg_vdev;
struct fw_rsc_vdev_vring rpmsg_vring0;
struct fw_rsc_vdev_vring rpmsg_vring1;
```

```
/* text carveout entry */
struct fw_rsc_carveout text_cout;
```

```
/* data carveout entry */
struct fw_rsc_carveout data_cout;
```

```
/* ipcd data carveout entry */
struct fw_rsc_carveout ipcd_data_cout;
```

```
/* trace entry */
struct fw_rsc_trace trace;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem0;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem1;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem2;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem3;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem4;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem5;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem6;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem7;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem8;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem9;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem10;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem11;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem12;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem13;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem14;
```

```
/* devmem entry */
struct fw_rsc_devmem devmem15;
```

```
}; extern char ti_trace_SysMin_Module_State_0_outbuf__A;
```

```
1. define TRACEBUFADDR (UInt32)&ti_trace_SysMin_Module_State_0_outbuf__A
```

```
1. pragma DATA_SECTION(ti_ipc_remoteproc_ResourceTable, ".resource_table")
```

```
2. pragma DATA_ALIGN(ti_ipc_remoteproc_ResourceTable, 4096)
```

```
struct my_resource_table ti_ipc_remoteproc_ResourceTable = {
```

```
1, /* we're the first version that implements this */
21, /* number of entries in the table */
0, 0, /* reserved, must be zero */
/* offsets to entries */
```

```
{
    offsetof(struct my_resource_table, rpmsg_vdev),
    offsetof(struct my_resource_table, text_cout),
    offsetof(struct my_resource_table, data_cout),
    offsetof(struct my_resource_table, ipcddata_cout),
    offsetof(struct my_resource_table, trace),
    offsetof(struct my_resource_table, devmem0),
    offsetof(struct my_resource_table, devmem1),
    offsetof(struct my_resource_table, devmem2),
    offsetof(struct my_resource_table, devmem3),
    offsetof(struct my_resource_table, devmem4),
    offsetof(struct my_resource_table, devmem5),
    offsetof(struct my_resource_table, devmem6),
    offsetof(struct my_resource_table, devmem7),
    offsetof(struct my_resource_table, devmem8),
    offsetof(struct my_resource_table, devmem9),
    offsetof(struct my_resource_table, devmem10),
    offsetof(struct my_resource_table, devmem11),
    offsetof(struct my_resource_table, devmem12),
    offsetof(struct my_resource_table, devmem13),
    offsetof(struct my_resource_table, devmem14),
    offsetof(struct my_resource_table, devmem15),
},
```

```
/* rpmsg vdev entry */
{
    TYPE_VDEV, VIRTIO_ID_RPMSG, 0,
    RPMSG_IPU_C0_FEATURES, 0, 0, 0, 2, { 0, 0 },
    /* no config data */
},
/* the two vring */
{ IPU_MEM_RPMSG_VRING0, 4096, IPU_RPMSG_VQ0_SIZE, 1, 0 },
{ IPU_MEM_RPMSG_VRING1, 4096, IPU_RPMSG_VQ1_SIZE, 2, 0 },
```

```
{
    TYPE_CARVEOUT,
    IPU_MEM_TEXT, 0,
    IPU_MEM_TEXT_SIZE, 0, 0, "IPU_MEM_TEXT",
},
```

```
{
    TYPE_CARVEOUT,
    IPU_MEM_DATA, 0,
    IPU_MEM_DATA_SIZE, 0, 0, "IPU_MEM_DATA",
},
```

```
{
    TYPE_CARVEOUT,
    IPU_MEM_IPC_DATA, 0,
    IPU_MEM_IPC_DATA_SIZE, 0, 0, "IPU_MEM_IPC_DATA",
},
```

```
{
    TYPE_TRACE, TRACEBUFADDR, 0x8000, 0, "trace:sysm3",
},
```

```
{
    TYPE_DEVMEM,
    IPU_MEM_IPC_VRING, PHYS_MEM_IPC_VRING,
    IPU_MEM_IPC_VRING_SIZE, 0, 0, "IPU_MEM_IPC_VRING",
},
```

```
{
    TYPE_DEVMEM,
    IPU_MEM_IOBUFS, PHYS_MEM_IOBUFS,
    IPU_MEM_IOBUFS_SIZE, 0, 0, "IPU_MEM_IOBUFS",
},
```

```
{
    TYPE_DEVMEM,
    IPU_TILER_MODE_0_1, L3_TILER_MODE_0_1,
    SZ_256M, 0, 0, "IPU_TILER_MODE_0_1",
},
```

```
{
    TYPE_DEVMEM,
    IPU_TILER_MODE_2, L3_TILER_MODE_2,
    SZ_128M, 0, 0, "IPU_TILER_MODE_2",
},
```

```
{
    TYPE_DEVMEM,
    IPU_TILER_MODE_3, L3_TILER_MODE_3,
    SZ_128M, 0, 0, "IPU_TILER_MODE_3",
},
```

```
{
    TYPE_DEVMEM,
    IPU_PERIPHERAL_L4CFG, L4_PERIPHERAL_L4CFG,
    SZ_16M, 0, 0, "IPU_PERIPHERAL_L4CFG",
},
```

<pre>{ TYPE_DEVMEM, IPU_PERIPHERAL_L4PER1, L4_PERIPHERAL_L4PER1, SZ_2M, 0, 0, "IPU_PERIPHERAL_L4PER1", },</pre>
<pre>{ TYPE_DEVMEM, IPU_PERIPHERAL_L4PER2, L4_PERIPHERAL_L4PER2, SZ_4M, 0, 0, "IPU_PERIPHERAL_L4PER2", },</pre>
<pre>{ TYPE_DEVMEM, IPU_PERIPHERAL_L4PER3, L4_PERIPHERAL_L4PER3, SZ_8M, 0, 0, "IPU_PERIPHERAL_L4PER3", },</pre>
<pre>{ TYPE_DEVMEM, IPU_PERIPHERAL_L4EMU, L4_PERIPHERAL_L4EMU, SZ_16M, 0, 0, "IPU_PERIPHERAL_L4EMU", },</pre>
<pre>{ TYPE_DEVMEM, IPU_PERIPHERAL_PRUSS, L3_PERIPHERAL_PRUSS, SZ_1M, 0, 0, "IPU_PERIPHERAL_PRUSS", },</pre>
<pre>{ TYPE_DEVMEM, IPU_IVAHD_CONFIG, L3_IVAHD_CONFIG, SZ_16M, 0, 0, "IPU_IVAHD_CONFIG", },</pre>
<pre>{ TYPE_DEVMEM, IPU_IVAHD_SL2, L3_IVAHD_SL2, SZ_16M, 0, 0, "IPU_IVAHD_SL2", },</pre>
<pre>{ TYPE_DEVMEM, IPU_PERIPHERAL_DMM, L3_PERIPHERAL_DMM, SZ_1M, 0, 0, "IPU_PERIPHERAL_DMM", },</pre>
<pre>{ TYPE_DEVMEM, IPU_OCMC_RAM, L3_OCMC_RAM, SZ_4M, 0, 0, "IPU_OCMC_RAM", },</pre>
<pre>{ TYPE_DEVMEM, IPU_EMIF_SDRAM, L3_EMIF_SDRAM, SZ_256M, 0, 0, "IPU_EMIF_SDRAM", },</pre>

</syntaxhighlight> You can find 3 new TYPE_DEVMEM entries added in above resource table.

- L3_PERIPHERAL_PRUSS
- IPU_OCMC_RAM
- IPU_EMIF_SDRAM

All ELF section placements (as well as the VRINGS, which aren't in an ELF section) are placed in memory allocated from the remoteproc CMA area and are mapped to the virtual address as specified in the TYPE_CARVEOUT entries.

The virtual addresses #defined are: <syntaxhighlight lang='c'>

- ```
1. define IPU_MEM_TEXT 0x0
2. define IPU_MEM_DATA 0x80000000

1. define IPU_MEM_IOBUFS 0x90000000

1. define IPU_MEM_IPC_DATA 0x9F000000
2. define IPU_MEM_IPC_VRING 0x60000000
```

</syntaxhighlight>

The followings are fixed physical address to facilitate a fixed MMU table. <syntaxhighlight lang='c'>

- ```
1. if defined(VAYU_IPU_1)
2. define PHYS_MEM_IPC_VRING 0x9D000000
3. elif defined (VAYU_IPU_2)
4. define PHYS_MEM_IPC_VRING 0x95800000
5. endif
```

These PHYS_MEM values match exactly the physical address specified in the remoteproc CMA area in Linux DTS file:

```
ipu2_cma_pool: ipu2_cma@95800000 {
    compatible = "shared-dma-pool";
    reg = <0x95800000 0x3800000>;
    reusable;
    status = "okay";
};
```

```
ipu1_cma_pool: ipu1_cma@9d000000 {
    compatible = "shared-dma-pool";
    reg = <0x9d000000 0x2000000>;
    reusable;
    status = "okay";
};
```

The 1st entry in the resource table is:

```
/* rpmsg vdev entry */
{
    TYPE_VDEV, VIRTIO_ID_RPMSG, 0,
    RPMSG_IPU_C0_FEATURES, 0, 0, 0, 2, { 0, 0 },
    /* no config data */
},
/* the two vrings */
{ IPU_MEM_RPMSG_VRING0, 4096, IPU_RPMSG_VQ0_SIZE, 1, 0 },
{ IPU_MEM_RPMSG_VRING1, 4096, IPU_RPMSG_VQ1_SIZE, 2, 0 },
```

This tells remoteproc to allocate the vrings and vring buffers, and the remoteproc CMA area is used for this.

The 2nd-4th entries are:

```
{
    TYPE_CARVEOUT,
    IPU_MEM_TEXT, 0,
    IPU_MEM_TEXT_SIZE, 0, 0, "IPU_MEM_TEXT",
},
{
    TYPE_CARVEOUT,
    IPU_MEM_DATA, 0,
    IPU_MEM_DATA_SIZE, 0, 0, "IPU_MEM_DATA",
},
{
    TYPE_CARVEOUT,
    IPU_MEM_IPC_DATA, 0,
    IPU_MEM_IPC_DATA_SIZE, 0, 0, "IPU_MEM_IPC_DATA",
},
},
```

These carveouts tell remoteproc to allocate memory from its CMA area and map the allocated physical address to the virtual address specified in the carveout (the 1st field after TYPE_CARVEOUT), in the IPU's MMU ("iommu" in Linux kernel parlance). These are for the ELF sections that are placed (mapped) in those virtual address.

The TYPE_TRACE entry tells remoteproc where the remote executable's trace buffer is, using its C symbol.

The TYPE_DEVMEM entries are virtual <-> physical mappings. remoteproc just creates an IPU MMU mapping for the entry. The 1st TYPE_DEVMEM entry corresponds to the vrings and creates the IPU MMU mapping needed to access them from the IPU core:

```
{
    TYPE_DEVMEM,
    IPU_MEM_IPC_VRING, PHYS_MEM_IPC_VRING,
    IPU_MEM_IPC_VRING_SIZE, 0, 0, "IPU_MEM_IPC_VRING",
},
```

New TYPE_DEVMEM entry

To add a new TYPE_DEVMEM entry, for example, to access PRU-ICSS from IPU

- 1. specify the physical address of PRU-ICSS and its virtual address.

```
#define L3_PERIPHERAL_PRUSS 0x4B200000 #define IPU_PERIPHERAL_PRUSS 0x6B200000
```

- 2. increase size of **offset[X]** array in struct **my_resource_table**.
- 3. add new **struct fw_rsc_devmem devmemY** entry in struct **my_resource_table**.
- 4. increase **number** of entries in **ti_ipc_remoteproc_ResourceTable**.
- 5. add a **offsetof(struct my_resource_table, devmemY)** in **ti_ipc_remoteproc_ResourceTable**.
- 6. add actual entry in **ti_ipc_remoteproc_ResourceTable**.

```
{
    TYPE_DEVMEM,
    IPU_PERIPHERAL_PRUSS, L3_PERIPHERAL_PRUSS,
    SZ_1M, 0, 0, "IPU_PERIPHERAL_PRUSS",
},
```

1. define L3_PRU_ICSS 0x4B200000
2. define DSP PRU ICSS 0x4B200000

The MMU pagetable can be dumped through debugfs - "cat /sys/kernel/debug/omap_iommu/<iommu name>/pagetable"

	Keystone=								
{{ 1. switchcategory:MultiCore= - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article IPC Resource customTable here.	■ For technical support on MultiCore devices, please post your questions in the <u>C6000 MultiCore Forum</u> ■ For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum	C2000=For technical support on the C2000 please post your questions on <u>The C2000 Forum</u>. Please post only comments about the article IPC Resource customTable here.	DaVinci=For technical support on DaVincoplease post your questions on <u>The DaVinci Forum</u>. Please post only comments about the article IPC Resource customTable here.	MSP430=For technical support on MSP430 please post your questions on <u>The MSP430 Forum</u>. Please post only comments about the article IPC Resource customTable here.	OMAP35x=For technical support on OMAP please post your questions on <u>The OMAP Forum</u>. Please post only comments about the article IPC Resource customTable here.	OMAPL1=For technical support on OMAP please post your questions on <u>The OMAP Forum</u>. Please post only comments about the article IPC Resource customTable here.	MAVRK=For technical support on MAVRK please post your questions on <u>The MAVRK Toolbox Forum</u>. Please post only comments about the article IPC Resource customTable here.	For technic please pos questions i http://e2e.ti.com/forums/processors/114/maurk-toolbox Please pos comments article IPC customTa }}	

Amplifiers & Linear
Audio
Broadband RF/IF &
Clocks & Timers
Data Converters

DLP & MEMS
High-Reliability
Interface
Logic
Power Management

Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers

Temperature Sensors & Control ICs

Wireless Connectivity

This page was last edited on 9 November 2015, at 08:18.

Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.