i.MX6SX M4 MPU Settings For RPMSG

Biyong SUN
10, AUG 2018
Issue Description

RPMSG demo doesn’t work on Customized board using 256M DDR.
Root cause

The ARMv7-M default address map also need to be configured to fit the customized DDR memory size.
# ARMv7-M default address map

<table>
<thead>
<tr>
<th>Name</th>
<th>Device Type</th>
<th>XN</th>
<th>Cache</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000-</td>
<td>Code</td>
<td>Normal</td>
<td>-</td>
<td>WT</td>
</tr>
<tr>
<td>0x01FFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20000000-</td>
<td>SRAM</td>
<td>Normal</td>
<td>-</td>
<td>WBWA</td>
</tr>
<tr>
<td>0x3FFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40000000-</td>
<td>Peripheral</td>
<td>Device</td>
<td>XN</td>
<td>-</td>
</tr>
<tr>
<td>0x5FFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x60000000-</td>
<td>RAM</td>
<td>Normal</td>
<td>-</td>
<td>WBWA</td>
</tr>
<tr>
<td>0x7FFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x80000000-</td>
<td>RAM</td>
<td>Normal</td>
<td>-</td>
<td>WT</td>
</tr>
<tr>
<td>0x9FFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA0000000-</td>
<td>Device</td>
<td>Device</td>
<td>shared</td>
<td>XN</td>
</tr>
<tr>
<td>0xBFFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC0000000-</td>
<td>Device</td>
<td>Device</td>
<td>XN</td>
<td>-</td>
</tr>
<tr>
<td>0xDFFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xE0000000-</td>
<td>System</td>
<td>-</td>
<td>XN</td>
<td>-</td>
</tr>
<tr>
<td>0xFFFFFFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000000-</td>
<td>PPB</td>
<td>SO,</td>
<td>(shared)</td>
<td>XN</td>
</tr>
<tr>
<td>0x00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+01000000</td>
<td>Vendor_SYS</td>
<td>Device</td>
<td>XN</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All of the 256M is inside the default 0x80000000-0x9FFFFFFF with write-thru cache attribute. **NOT shared device space.**
MPU(Memory Protect Unit)

The Protected memory System Architecture for ARMv7-M (PMSAv7) is an optional system level feature described in Protected Memory System Architecture (PMSAv7) on page B3-35. An implementation of PMSAv7 is known as a Memory Protection Unit (MPU).

Typically, ARMv7-M using MPU and ARMv7-Cortex using MMU.
FreeRTOS Key Code Changes to remap the address

platform/devices/MCIMX6X/startup/system_MCIMX6X_M4.c

/* M4 core clock root configuration. */
/* Initialize MPU */
/* Make sure outstanding transfers are done. */
__DMS();
/* Disable the MPU. */
KPU->CTRL = 0;

/* Select Region 0 to configure share memory with A9 (1MB). */
KPU->RNR  = 0;
KPU->RBAR = 0x8FF00000;
KPU->RASR = 0x03040027;

1M

/* Select Region 1 to configure ocram. */
KPU->RNR  = 1;
KPU->RBAR = 0x20900000;
KPU->RASR = 0x03080021;

/* Select Region 2 to configure DDR memory(1MB). */
KPU->RNR  = 2;
KPU->RBAR = 0x85F00000;
KPU->RASR = 0x03080027;

1M

/* Select Region 3 to configure QSPI memory. */
KPU->RNR  = 3;
KPU->RBAR = 0x60000000;
KPU->RASR = 0x03080029;

1M

/* Disable unused regions. */
MPU->RNR = 0;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 4;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 5;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 6;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 7;
MPU->RBAR = 0;
MPU->RASR = 0;

/* Enable Privileged default memory map and the MPU. */
MPU->CTRL = MPU_CTRL_ENABLE_MSK | MPU_CTRL_PRIVDEFENA_MSK;

/* Memory barriers to ensure subsequence data & instruction * transfers using updated MPU settings. */
__DSE();
__ISE();

/* Initialize Cache */
FreeRTOS MPU Address Map Example 1

/* M4 core clock root configuration. */
/* Initialize MPU */
/* Make sure outstanding transfers are done. */
_DMB();
/* Disable the MPU. */
MPU->CTRL = 0;

/* Select Region 0 to configure share memory with A9 (1MB). */
MPU->RNR = 0;
MPU->RBAR = 0x8F000000;
MPU->RASR = 0x03040027;

/* Select Region 1 to configure ocram. */
MPU->RNR = 1;
MPU->RBAR = 0x20900000;
MPU->RASR = 0x030B0021;

/* Select Region 2 to configure DDR memory(1MB). */
MPU->RNR = 2;
MPU->RBAR = 0x87F00000;
MPU->RASR = 0x030B0027;

/* Select Region 3 to configure QSPI memory. */
MPU->RNR = 3;
MPU->RBAR = 0x60000000;
MPU->RASR = 0x030B0029;

/* Disable unused regions. */
MPU->RNR = 3;

_DSB();
_ISB();

/* Initialize Cache */
__DMB();
/* Disable the MPU. */
MPU->CTRL = 0;

/* Select Region 0 to configure share memory with A9 (1MB). */
MPU->RNR = 0;
MPU->RBAR = 0x8FF00000;
MPU->RASR = 0x03040027;

/* Select Region 1 to configure ocram. */
MPU->RNR = 1;
MPU->RBAR = 0x20900000;
MPU->RASR = 0x030B0021;

/* Select Region 2 to configure DDR memory (1MB). */
MPU->RNR = 2;
MPU->RBAR = 0x8FE00000;
MPU->RASR = 0x030B0027;

/* Select Region 3 to configure QSPI memory. */
MPU->RNR = 3;

/* Enable Privileged default memory map and */
/* Memory barriers to ensure subsequence data */
/* transfers using updated MPU settings. */
MPU Region Attribute and Size Register (MPU_RASR)

The Region Attribute and Size Register (MPU_RASR) defines the size and access behavior of the associated memory region. The register fields are described in Table B3-42.

<table>
<thead>
<tr>
<th>Bits</th>
<th>R/W</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>R/W</td>
<td>ATTRS</td>
<td>The MPU Region Attribute Register as defined in Region attribute control on page B3-43</td>
</tr>
<tr>
<td>[15:8]</td>
<td>R/W</td>
<td>SRD</td>
<td>For regions of 256 bytes or larger, the Sub-Region Disable bits are used to disable 1/8 of the region per bit. For any bit that is set, the relevant 1/8 of the region’s address range is disabled with respect to the attribute settings.</td>
</tr>
<tr>
<td>[7:6]</td>
<td>-</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>[5:1]</td>
<td>R/W</td>
<td>SIZE</td>
<td>The region size as defined in Table B3-43.</td>
</tr>
<tr>
<td>[0]</td>
<td>R/W</td>
<td>ENABLE</td>
<td>When set, the associated region is enabled within the MPU. The global MPU enable bit must also be set for it to take effect.</td>
</tr>
</tbody>
</table>
Region size control

The size of a region is encoded in the MPU Region Size Register as shown in Table B3-43.

<table>
<thead>
<tr>
<th>Size Encoding</th>
<th>Region size in bytes</th>
<th>Size Encoding</th>
<th>Region size in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>-</td>
<td>10000</td>
<td>128KB</td>
</tr>
<tr>
<td>00001</td>
<td>-</td>
<td>10001</td>
<td>256KB</td>
</tr>
<tr>
<td>00010</td>
<td>-</td>
<td>10010</td>
<td>512KB</td>
</tr>
<tr>
<td>00011</td>
<td>-</td>
<td>10011</td>
<td>1MB</td>
</tr>
<tr>
<td>00100</td>
<td>32</td>
<td>10100</td>
<td>2MB</td>
</tr>
<tr>
<td>00101</td>
<td>64</td>
<td>10101</td>
<td>4MB</td>
</tr>
<tr>
<td>00110</td>
<td>128</td>
<td>10110</td>
<td>8MB</td>
</tr>
<tr>
<td>00111</td>
<td>256</td>
<td>10111</td>
<td>16MB</td>
</tr>
<tr>
<td>01000</td>
<td>512</td>
<td>11000</td>
<td>32MB</td>
</tr>
<tr>
<td>01001</td>
<td>1KB</td>
<td>11001</td>
<td>64MB</td>
</tr>
<tr>
<td>01010</td>
<td>2KB</td>
<td>11010</td>
<td>128MB</td>
</tr>
<tr>
<td>01011</td>
<td>4KB</td>
<td>11011</td>
<td>256MB</td>
</tr>
<tr>
<td>01100</td>
<td>8KB</td>
<td>11100</td>
<td>512MB</td>
</tr>
<tr>
<td>01101</td>
<td>16KB</td>
<td>11101</td>
<td>1GB</td>
</tr>
<tr>
<td>01110</td>
<td>32KB</td>
<td>11110</td>
<td>2GB</td>
</tr>
<tr>
<td>01111</td>
<td>64KB</td>
<td>11111</td>
<td>4GB</td>
</tr>
</tbody>
</table>
Region attribute control

The MPU region attribute fields define the memory type, cache, and access policies for a given memory region.

Table B3-44 Region attribute fields

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reserved</td>
<td>XN</td>
<td>Reserved</td>
<td>AP</td>
<td>Reserved</td>
<td>TEX</td>
<td>S</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table B3-45 TEX/CBS/S Encoding

<table>
<thead>
<tr>
<th>TEX</th>
<th>C</th>
<th>B</th>
<th>Description</th>
<th>Memory Type</th>
<th>Region Shareable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>Strongly ordered</td>
<td>Strongly ordered</td>
<td>Shareable</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
<td>1</td>
<td>Shared device</td>
<td>Device</td>
<td>Shareable</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>0</td>
<td>Outer and inner write through, no write allocate</td>
<td>Normal</td>
<td>5</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>1</td>
<td>Outer and inner write back, no write allocate</td>
<td>Normal</td>
<td>5</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>Outer and inner Non-cacheable</td>
<td>Normal</td>
<td>5</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
<td>IMPLEMENTATION DEFINED</td>
<td>IMPLEMENTATION DEFINED</td>
<td>IMPLEMENTATION DEFINED</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
<td>Outer and inner write back; write and read allocate</td>
<td>Normal</td>
<td>5</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>0</td>
<td>Non-shared device</td>
<td>Device</td>
<td>Not shareable</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>1</td>
<td>RESERVED</td>
<td>RESERVED</td>
<td>RESERVED</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>X</td>
<td>RESERVED</td>
<td>RESERVED</td>
<td>RESERVED</td>
</tr>
<tr>
<td>111</td>
<td>A</td>
<td>A</td>
<td>Cached memory.</td>
<td>Normal</td>
<td>5</td>
</tr>
</tbody>
</table>

Table B3-46 Cache policy encoding

<table>
<thead>
<tr>
<th>Memory Attribute Encoding (AA and BB)</th>
<th>Cache Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Non-cacheable</td>
</tr>
<tr>
<td>01</td>
<td>Write-back, write and read allocate</td>
</tr>
<tr>
<td>10</td>
<td>Write through, no write allocate</td>
</tr>
<tr>
<td>11</td>
<td>Write-back, no write allocate</td>
</tr>
</tbody>
</table>

The AP bits, AP[2:0], are used for access permissions. These are shown in Table B3-47.

Table B3-47 AP encoding

<table>
<thead>
<tr>
<th>AP[2:0]</th>
<th>Privileged Permissions</th>
<th>User Permissions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>No Access</td>
<td>No Access</td>
<td>All accesses generate a permission fault</td>
</tr>
<tr>
<td>001</td>
<td>Read/Write</td>
<td>No Access</td>
<td>Privileged access only</td>
</tr>
<tr>
<td>010</td>
<td>Read/Write</td>
<td>Read Only</td>
<td>Unprivileged (User) written generate permission faults</td>
</tr>
<tr>
<td>011</td>
<td>Read/Write</td>
<td>Read Only</td>
<td>Full access</td>
</tr>
<tr>
<td>100</td>
<td>UNPREDICTABLE</td>
<td>UNPREDICTABLE</td>
<td>RESERVED</td>
</tr>
<tr>
<td>101</td>
<td>Read Only</td>
<td>No Access</td>
<td>Privileged read only</td>
</tr>
<tr>
<td>110</td>
<td>Read Only</td>
<td>Read Only</td>
<td>Privileged User read only</td>
</tr>
<tr>
<td>111</td>
<td>Read Only</td>
<td>Read Only</td>
<td>Privileged User read only</td>
</tr>
</tbody>
</table>

The XN bit provides an Execute Never capability. Instructions must have read access as defined by the AP bits and XN clear for correct execution, otherwise a MemManage fault is generated when the instruction is issued.

Table B3-48 XN encoding

<table>
<thead>
<tr>
<th>XN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Instruction fetches allowed</td>
</tr>
<tr>
<td>1</td>
<td>Instruction fetches not allowed</td>
</tr>
</tbody>
</table>
Related changes(FreeRTOS)

platform/devices/MCIMX6X/linker/gcc/MCIMX6X_M4_ddr.Id (Suggest to create new one)

53 /* Specify the memory areas */
54 MEMORY
55 {
56   m_interrupts (RX) : ORIGIN = 0x87f00000, LENGTH = 0x00000240
57   m_text (RX) : ORIGIN = 0x87f00240, LENGTH = 0x00007DC0
58   m_data (RW) : ORIGIN = 0x20000000, LENGTH = 0x00008000
59 }
60
middleware/multicore/open-amp/porting/imx6sx_m4/platform_info.c
55  * Linux has a different alignment requirement, and its have 512 buffers instead of 32 buffers for the 2 ring
56  */
57  //#define VRING0_BASE                       0xBFFF0000
58  //#define VRING1_BASE                       0xBFFF8000
59  #define VRING0_BASE                       0x8FFF0000
60  #define VRING1_BASE                       0x8FFF8000
Confirm Link changes(FreeRTOS)

Please use hex editor to open the m4 binary image to confirm the link is correct.

```
<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>0A</th>
<th>0B</th>
<th>0C</th>
<th>0D</th>
<th>0E</th>
<th>0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000:</td>
<td>00</td>
<td>80</td>
<td>00</td>
<td>20</td>
<td>11</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>0010:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0020:</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>95</td>
<td>03</td>
</tr>
<tr>
<td>0030:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>41</td>
<td>04</td>
<td>F0</td>
<td>87</td>
<td>99</td>
<td>04</td>
</tr>
<tr>
<td>0040:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>0050:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>0060:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>0070:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>0080:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>0090:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>00A0:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>00B0:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>00C0:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
<tr>
<td>00D0:</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
<td>87</td>
<td>3D</td>
<td>03</td>
<td>F0</td>
</tr>
</tbody>
</table>
```

Related changes (Linux)

arch/arm/boot/dts/imx6sx-sdb-m4.dts (Suggest to create new one)

```c
11/{
12  reserved-memory {
13    m4_reserved{
14      reg = <0x87F00000 0x100000>;  
15      no-map;
16    };
17
18    rpmsg_reserved{
19      reg = <0x8FF00000 0x100000>;  
20      no-map;
21    };
22
23 };
24
25 }
26
27}
```
Confirm Linux Device Tree Changes

cat /proc/iomem

```
0x000000-0003ffff : /soc/aips-plane000000/pwm@0x000000
0x000300-0003ffff : /soc/aips-plane000000/pwm@0x300000
0x000600-00060000 : /soc/aips-plane000000/pwm@0x600000
0x000600-00060000 : /soc/aips-plane000000/pwm@0x600000
80000000-8007ffff : System RAM
80080000-80a842fb : Kernel code
80af0000-80bcb29b : Kernel data
88000000-88000000 : System RAM
90000000-bfffff : System RAM
root@mx6s_all:~#  
```

Diagram:

```
0x87F00000 (127M)
0x88000000 (128M)
0x8FF00000 (255M)
0x90000000 (256M)
```

```
口 m4_reserved
口 rpmmsg_reserved
```
Related changes (Linux)

arch/arm/mach-imx/imx_rpmsg.c

280 static int imx_rpmsg_probe(struct platform_device *pdev)
281 {
282  ..........  
283  ..........  
284   /* hardcodes here now. */
285   // rpdev->vring[0] = 0xBFFF0000;
286   // rpdev->vring[1] = 0xBFFF8000;
287   rpdev->vring[0] = 0x8FFF0000;
288   rpdev->vring[1] = 0x8FFF8000;
292   
293   } else {
294    break;
298   }  
299 
300  }  27  
28 };
Related changes(Linux)

CMA may need to be changed to fit the Physical Memory Size
arch/arm/boot/dts/imx6sx-sdb-m4.dts

    11 /{
    12       reserved-memory {
    13           m4_reserved{
    14               reg = <0x87F00000 0x100000>;  
    15               no-map;
    16            
    17           rpmsg_reserved{
    18               reg = <0x8FF00000 0x100000>;  
    19               no-map;
    20            
    21           linux,cma {
    22               size = <0x20000000>;
    23            
    24       
    25       
    26       
    27       
    28       
    29       
    30       
    31    };
CMA Arrangement Consideration

By default, CMA will be allocated from top of the whole memory. For example, 512M from base 0x80000000 is 0xA0000000. And grows down to the lower address. Please consider to arrange CMA first by no touching the area from top down to CMA size.
CMA Arrangement Consideration (Cont.)

Check the boot log to find the CMA Base Address and size

Starting kernel ...

Booting Linux on physical CPU 0x0

...........

OF: fdt:Machine model: Freescale i.MX6 SoloX SDB RevB Board
Reserved memory: created CMA memory pool at 0x9C000000, size 64 MiB
OF: reserved mem: initialized node linux,cma, compatible id shared-dma-pool
Memory policy: Data cache writealloc
Enable CMA debugfs to get more details.

- Enable the CMA debugfs

```c
CONFIG_CMA_DEBUGFS:
```

| Turns on the DebugFS interface for CMA. |
| Symbol: CMA_DEBUGFS [=y] |
| Type: boolean |
| Prompt: CMA debugfs interface |
| Location: |
| - Kernel Features |
| - Contiguous Memory Allocator (CMA [=y]) |
| Defined at mm/Kconfig:520 |
| Depends on: CMA [=y] && DEBUG_FS [=y] |

- Get the CMA Base Address

```
cat /sys/kernel/debug/cma/cma-0/base_pfn
```

638976

638976 = 0x9C000

0x9C000000 is the CMA Base Address.
Related changes (UBOOT Command)

bootaux 0x87F00000

setenv rpmsgimg rpmsg_pingpong_freertos_example.bin

setenv m4runaddr 0x87F00000

setenv bootm4frvddr 'fatload mmc ${mmcdev}:${mmcpart} ${m4runaddr} ${rpmsgimg};dcache flush;bootaux ${m4runaddr}'

After create bootm4frvddr, you can use “run bootm4frvddr” to run the m4 image or add to the bootcmd for auto boot.
PMU Configuration Code

/* M4 core clock root configuration. */
/* Initialize MPU */
/* Make sure outstanding transfers are done. */
__DMB();
/* Disable the MPU. */
MPU->CTRL = 0;

/* Select Region 0 to configure share memory with A9(1MB). */
MPU->RNR = 0;
MPU->RBAR = 0x8ff00000;
MPU->RASR = 0x03040027;

/* Select Region 1 to configure ocram. */
MPU->RNR = 1;
MPU->RBAR = 0x20900000;
MPU->RASR = 0x030B0021;

/* Select Region 2 to configure DDR memory(1MB). */
MPU->RNR = 2;
MPU->RBAR = 0x80000000;
MPU->RASR = 0x030B0027;

/* Select Region 3 to configure QSPI memory. */
MPU->RNR = 3;
MPU->RBAR = 0x60000000;
MPU->RASR = 0x030B0029;

/* Disable unused regions. */
MPU->RNR = 4;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 5;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 6;
MPU->RBAR = 0;
MPU->RASR = 0;
MPU->RNR = 7;
MPU->RBAR = 0;
MPU->RASR = 0;

/* Enable Privileged default memory map and the MPU. */
MPU->CTRL = MPU_CTRL_ENABLE_Msk | MPU_CTRL_PRIVDEFENA_Msk;

/* Memory barriers to ensure subsequence data & instruction transfers using updated MPU settings. */
__DSB();
__ISB();

/* Initialize Cache */
SECURE CONNECTIONS FOR A SMARter WORLD