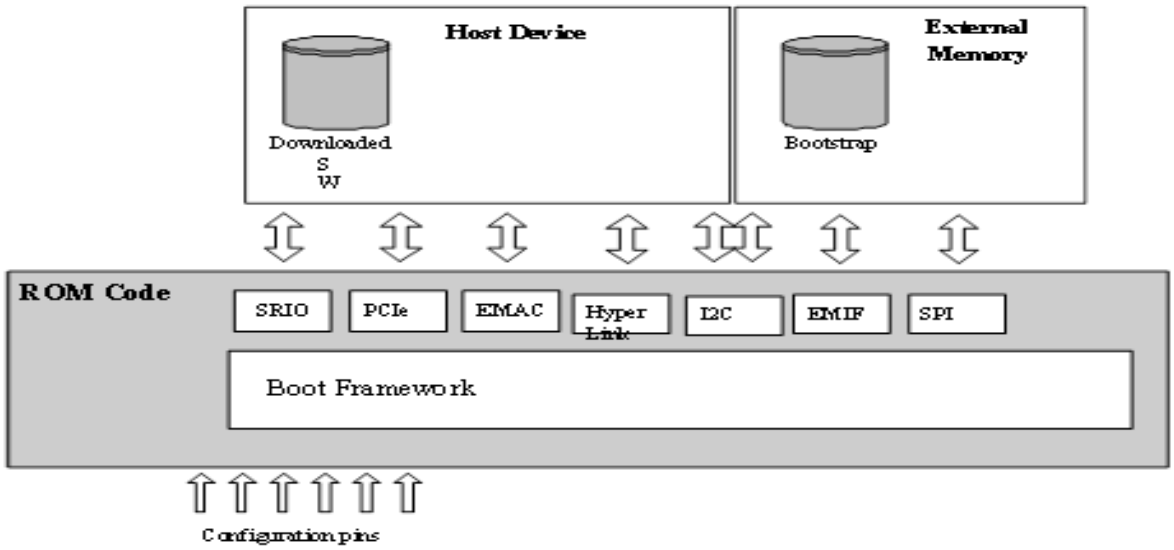
# Keystone Bootloader

# Keystone ROM Boot Loader

- Code to transfer application code from memory or host to high speed internal memory
- Boot loader code is burned in the DSP ROM (Non-modifiable)
- Base address for the Boot Code is 0x20B00000
- Boot Loader is broadly divided into two types
  - Memory boot  where application is stored in a slow external memory
  - Host Boot where the boot is driven by a host device connected through fast transport.
- Seven different types of boot modes are supported

# ROM Boot Modes

- **Supported Boot Modes**
  - **I2C Boot**
    - **Master Boot (from I2C EEPROM)**
    - **Master-Broadcast Boot(Master Boot followed by broadcast to slave cores)**
    - **Passive Boot (external I2C host)**
  - **SPI Boot (from SPI flash)**
  - **SRIO Boot(from external host connected through SRIO)**
  - **Ethernet Boot (boot from external host connected through Ethernet)**
  - **PCIe Boot (boot from external host connected through PCIe )**
  - **HyperLink Boot (boot from external host connected through HyperLink)**
  - **EMIF16 NOR Boot(boot from NOR Flash)**
    - **Device Manual will detail supported types.**



# Boot Mode Configuration Pins

- Boot mode and configurations are chosen using bootstrap pins on the device.
  - Pins are latched and stored in13 bits of the DEVSTAT register during POR.
- The configuration format for these 13 bits are shown in the table:

| Boot Mode Pins | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PLL Mult<br>I2C/SPI Ext Dev Cfg | | | Device Configuration | | | | | | | Boot Device | | |

- Boot Device [2:0] is dedicated for selecting the boot mode
- Device Configuration [9:3] is used to specify the boot mode specific configurations.
- PLL Multi [12:10] are used for PLL selection. In case of I2C/SPI boot mode, it is used for extended device configuration. (PLL is bypassed for these two boot modes)

# Device Startup from Power on Reset (POR)

- Boot Startup procedure executed only once during:
  - Power On
  - Hard Reset
  - Soft Reset
- Bootstrap pins are only latched during Power On Reset (POR)
- Default boot parameter table is chosen based on the selected boot mode
- Boot strap pin configuration parameters are updated in the boot parameter table
- At completion, the ROM code branches to the main boot function, using the table to configure boot operation
  - The boot table can be modified and Boot can be re-executed after startup is complete by branching to the boot run function. (Typical Case: Secondary Boot Loader – I2C loads custom parameter table)

# Device Startup from Hard/Soft reset

- For hard and soft resets the Boot code must determine the hibernation state.
  - Hibernation is the process of shutting down unused CorePacs and IP blocks to save power consumption of the overall system.
- Saving all relevant configurations and register values is the application's responsibility based on the selected hibernation mode.
  - Hibernation1 – Values stored in MSMC SRAM.
  - Hibernation2 – Values stored in DDR3.
- The Application is also responsible for setting the appropriate hibernation mode in the PWRSTATECTL register.
- The Application will also set the branch address in the PWRSTATECTL register.

# Hibernation explained

- Hibernation 1
  - The application needs to ensure that the chip control register is set correctly to avoid MSMC reset.
- Hibernation 2
  - MSMC is reinitialized to default values.
- For both modes, the Application is responsible for shutdown of all desired IP blocks
- A hard or soft reset can be configured to bring  a hibernating device out of hibernation
  - After the reset, the boot loader code checks the PWRSTATECTL register to identify the hibernation mode and branch address.
  - Subsequent Actions
    - Peripherals and Corepacs are powered
    - The awakened device branches to the application code which utilizes the values stored in MSMC or DDR3 prior to hibernation

# PLL Configuration

- The boot code sets the PLL multiplier based on the core frequency set in the EFUSE register.

| Boot PLL Select [2:0] | Input Clock Freq (MHz) | core = 800 MHz | | core = 1000 MHz | | core = 1200 MHz | | core = 1400 MHz | | Core = 1250 MHz | | Core = 1500 MHz | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Clkr | Clkf | Clkr | Clkr | Clkr | Clkf | Clkr | Clkf | Clkr | Clkf | Clkr | Clkf |
| 0 | 50.00 | 0 | 31 | 0 | 0 | 0 | 47 | 0 | 55 | 0 | 49 | 0 | 59 |
| 1 | 66.67 | 0 | 23 | 0 | 0 | 0 | 35 | 0 | 41 | 1 | 74 | 0 | 44 |
| 2 | 80.00 | 0 | 19 | 0 | 0 | 0 | 29 | 0 | 34 | 3 | 124 | 1 | 74 |
| 3 | 100.00 | 0 | 15 | 0 | 0 | 0 | 23 | 0 | 27 | 0 | 24 | 0 | 29 |
| 4 | 156.25 | 24 | 255 | 4 | 24 | 24 | 383 | 24 | 447 | 0 | 15 | 4 | 95 |
| 5 | 250.00 | 4 | 31 | 0 | 4 | 4 | 47 | 4 | 55 | 0 | 9 | 0 | 11 |
| 6 | 312.50 | 24 | 127 | 4 | 24 | 24 | 191 | 24 | 223 | 0 | 7 | 4 | 47 |
| 7 | 122.88 | 47 | 624 | 28 | 13 | 13 | 624 | 13 | 318 | 2 | 60 | 4 | 121 |

# Boot Device

- Boot Device Selection Values

| Boot Mode Pins: Boot Device Values | |
|---|---|
| Value | Boot Device |
| 0 | Sleep / EMIF16[1] |
| 1 | Serial Rapid I/O |
| 2 | Ethernet (SGMII) (PA driven from core clk) |
| 3 | Ethernet (SGMII) (PA driver from PA clk) |
| 4 | PCIe |
| 5 | I2C |
| 6 | SPI |
| 7 | HyperLink |

1. See the device-specific data manual for information.

- For interfaces supporting more than one mode of operation, the configuration bits are used to establish the necessary settings

# Boot Configuration – EMIF16 Mode

- EMIF16 mode is used to boot from the NOR flash.
- The boot loader configures the EMIF16 and then sets the boot complete bit corresponding to corePac0 in the boot complete register and then branches to EMIF16 CS2 data memory at 0x70000000.
- No Memory is reserved by the boot loader.

| Sleep / EMIF16 Configuration Bit Fields | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Reserved | | Wait Enable | Sub-Mode | | SR Index | |

| Sleep / EMIF16 Configuration Bit Field Description | | |
|---|---|---|
| Bit Field | Value | Description |
| Sub-Mode | 0b00 | Sleep Boot |
| | 0b01 | EMIF16 boot |
| | 0b10-0b11 | Reserved |
| Wait Enable | 0b0 | Wait enable disabled (EMIF16 sub mode) |
| | 0b1 | Wait enable enabled (EMIF16 sub mode) |

# Boot Configuration – Ethernet

- Ethernet(SGMII) boot configuration sets SERDES clock and device ID.

| Ethernet (SGMII) Device Configuration Bit fields | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| SERDES Clock Mult | | Ext connection | | Dev ID | Dev ID (SR ID) | |

| Ethernet (SGMII) Configuration Bit fields description | | |
|---|---|---|
| Bit field | Value | Description |
| Ext connection | 0 | Mac to Mac connection, master with auto negotiation |
| | 1 | Mac to Mac connection, slave, and Mac to Phy |
| | 2 | Mac to Mac, forced link |
| | 3 | Mac to fiber connection |
| Device ID | 0-7 | This value is used in the device ID field of the Ethernet ready frame. Bits 1:0 are use for the SR ID. |
| SERDES Clock Mult The output frequency of the PLL must be 1.25 GBs. | 0 | x8 for input clock of 156.25 MHz |
| | 1 | x5 for input clock of 250 MHz |
| | 2 | x4 for input clock of 312.5 MHz |
| | 3 | Reserved |

# Boot Configuration – Serial RapidIO

- SRIO boot configuration sets the Clock, Lane configuration, and mode

| Rapid I/O Device Configuration Bit Fields | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Lane Setup | Data Rate | | Ref Clock | | SR ID | |

| SRIO Configuration Bit Field Descriptions | | |
|---|---|---|
| Bit Field | Value | Description |
| SR ID | 0-3 | Smart Reflex ID |
| Ref Clock | 0 | Reference Clock = 156.25 MHz |
| | 1 | Reference Clock = 250 MHz |
| | 2 | Reference Clock = 312.5 MHz |
| Data Rate | 0 | Data Rate = 1.25 GBs |
| | 1 | Data Rate = 2.5 GBs |
| | 2 | Data Rate = 3.125 GBs |
| | 3 | Data Rate = 5.0 GBs |
| Lane Setup | 0 | Port Configured as 4 ports each 1 lane wide (4 -1x ports) |
| | 1 | Port Configured as 2 ports 2 lanes wide (2 – 2x ports) |

# Boot Configuration
## I2C Master Mode

- In master mode the I2C Device Configuration uses 7 bits of device configuration instead of 5 bits used in passive mode.
- In this mode device will make the initial read of the I2C EEPROM while the PLL is in bypass.
- The initial boot parameter table will contain the desired clock multiplier which will be setup prior to any subsequent reads.

| I2C Master Mode Device Configuration Bit Fields | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Rsvd | Speed | Address | Rsvd | Mode (0) | Parameter Index . | | | | |

| I2C Master Mode Device Configuration Field Descriptions | | |
|---|---|---|
| Bit Field | Value | Description |
| Mode | 0 | Master Mode |
| | 1 | Passive Mode |
| Address | 0 | Boot From I2C EEPROM at I2C bus address 0x50 |
| | 1 | Boot From I2C EEPROM at I2C bus address 0x51 |
| Speed | 0 | I2C data rate set to approximately 20 kHz |
| | 1 | I2C fast mode. Data rate set to approximately 400 kHz (will not exceed) |
| Parameter Index | 0-31 | Identifies the index of the configuration table initially read from the I2C EEPROM |

# Boot Configuration – I2C Passive Mode

- In passive mode the I2C Device Configuration uses 5 bits of device configuration instead of 7 used in master mode.
- In passive mode the device does not drive the clock, but simply acks data received on the specified address.

| I2C Passive Mode Device Configuration Bit Fields | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Rsvd (Must be 1) | Mode (1) | Receive I2C Address | | | Rsvd | |

| I2C Passive Mode Device Configuration Field Descriptions | | |
|---|---|---|
| Bit Field | Value | Description |
| Mode | 0 | Master Mode |
| | 1 | Passive Mode |
| Address | 0-7 | The I2C Bus address the device will listen to for data |

# Boot Configuration – SPI Mode

Similar to I2C, the bootloader reads either a boot parameter table or boot config table that is at the address specified by the first boot parameter table and executes it directly.

| SPI Device Configuration Bit Fields | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Mode (clk Pol/Phase) | | 4,5pin | Addr Width | Chip select | | Parameter Table | | | |

| SPI Device Configuration Field Descriptions | | |
|---|---|---|
| Bit Field | Value | Description |
| Mode | 0 | Data is output on the rising edge of SPICLK. Input data is latched on the falling edge. |
| | 1 | Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK. |
| | 2 | Data is output on the falling edge of SPICLK. Input data is latched on the rising edge. |
| | 3 | Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK. |
| 4,5 pin | 0 | 4 pin mode used |
| | 1 | 5 pin mode used |
| Addr Width | 0 | 16 bit address values are used |
| | 1 | 24 bit address values are used |
| Chip Select | 0-3 | The chip select field value |
| Parameter Table Index | 0-3 | Specifies which parameter table is loaded |
| SR Index | 0-3 | Smart Reflex Index |

# Boot Configuration – PCI Express

• In PCIe mode, the host configures memory and loads all the sections directly to the memory.

| PCI Device Configuration Bit Fields | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Rsvd | BAR Config | | | | SR ID | |

| PCI Device Configuration Bit Fields | | |
|---|---|---|
| Bit Field | Value | Description |
| SR ID | 0-3 | Smart Reflex ID |
| Bar Config | 0-0xf | See Next Slide |

# Boot Configuration – PCI Express

| BAR Config / PCIe Window Sizes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 32 bit Address Translation | | | | | 64 bit Address Translation | |
| BAR cfg | BAR0 | BAR1 | BAR2 | BAR3 | BAR4 | BAR5 | BAR1/2 | BAR3/4 |
| 0b0000 | PCIe MMRs | 32 | 32 | 32 | 32 | Clone of BAR4 | | |
| 0b0001 | | 16 | 16 | 32 | 64 | | | |
| 0b0010 | | 16 | 32 | 32 | 64 | | | |
| 0b0011 | | 32 | 32 | 32 | 64 | | | |
| 0b0100 | | 16 | 16 | 64 | 64 | | | |
| 0b0101 | | 16 | 32 | 64 | 64 | | | |
| 0b0110 | | 32 | 32 | 64 | 64 | | | |
| 0b0111 | | 32 | 32 | 64 | 128 | | | |
| 0b1000 | | 64 | 64 | 128 | 256 | | | |
| 0b1001 | | 4 | 128 | 128 | 128 | | | |
| 0b1010 | | 4 | 128 | 128 | 256 | | | |
| 0b1011 | | 4 | 128 | 256 | 256 | | | |
| 0b1100 | | | | | | | 256 | 256 |
| 0b1101 | | | | | | | 512 | 512 |
| 0b1110 | | | | | | | 1024 | 1024 |
| 0b1111 | | | | | | | 2048 | 2048 |

# Boot Configuration – HyperLink Mode

- HyperLink boot mode boots the DSP through the ultra short range HyperLink.
- The host loads the boot image directly through the link and then generates the interrupt to wake the DSP.

| MCM Boot Device Configuration | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 |
| Reserved | Data Rate | | Ref Clock | | SR Index | |

| MCM Boot Device Configuration Field Descriptions | | |
|---|---|---|
| Bit Field | Value | Description |
| SR Index | 0-3 | Smart Reflex Index |
| Ref Clock | 0 | 156.25 MHz |
| | 1 | 250 MHz |
| | 2 | 312.5 MHz |
| Data Rate | 0 | 1.25 GBs |
| | 1 | 3.125 GBs |
| | 2 | 6.25 GBs |
| | 3 | 12.5 GBs |

# Booting multiple cores

- During the boot process the boot loader code is loaded into the L2 of corePac0 from the ROM.
- The high 0xD23F bytes of this L2 is reserved for the boot code. User should not overwrite this area.
- All the other corePacs are executing IDLE.
- The user should load the image into the L2 of corePacs they want to boot up.
- Before setting the boot complete register, the user should also set the start address of the code in the respective BOOT MAGIC ADDRESS of the corePac's L2.
- Finally the user image should also write the IPC interrupt register to bring the required corePacs out of IDLE.

# Secondary Bootload Option

# Second Stage Boot Load Process

Q: What if more boot parameters are needed than can be specified in the boot pins?

A:Other parameter values can be updated through I2B boot mode

• In this case, the I2C boot will start with a I2C boot parameter table which will in turn load a custom updated parameter table for a specific boot mode.

• Once the default parameter table is updated, the boot code executes using the updated boot parameter structure, using the same process as the primary boot mode.

# Second Stage Boot Load Specifics

- The EEPROM image loaded will have two boot parameter tables
- The First one will be an I2C boot parameter table, setting the core clock and also the address of the next block.
- The next block will have the desired boot mode specific boot parameter table with the user desired values.
- After loading this image into the EEPROM, the boot mode in the boot strap is set for I2C master boot.
- After POR, the I2C boot code is executed as a first stage boot load, which will update the default boot parameter table and re-enter the boot code, executing the boot code utilizing the user specific parameters.

# EVM Specifics and IBL

# Additional Details

# Boot Runtime – Ethernet

- The SERDES, SGMII and switch are not configured if the options field of the Ethernet boot parameter table indicate initialization is bypassed. This will be the default case when the boot is initiated by hard or soft reset, reset isolation has been enabled, and the devices are powered up and enabled.
- SERDES: The boot ROM programs the SGMII_SERDES_CFGPLL register, the SGMII_SERDES_CFGRX registers (both lanes) and SGMII_SERDES_CFGTX register (both lanes).
- SGMII: The SGMII is enabled in full duplex mode, gigabit rate. Broadcast packet reception is enabled based on the boot parameter table.
- QMSS: The QMSS is configured to manage descriptors using a single memory region. Each descriptor is sized to 48 bytes with extended packet information block present.
- PASS: A custom firmware load is used for PASS. This load simple directs all received packets to the CPDMA, using flow configuration 0.
- Interrupt System: Polling is used to detect packet arrival in queue 896, so interrupts are not configured
- After initialization, the device will broadcast a ready frame containing its device ID and MAC address.
- The Ethernet Host is responsible to receive the ready frame and follow up with the boot packets to the DSP using the device's MAC address or ID. If Broadcast Rx is configured, then the host can broadcast a common image to all DSPs.
- The image is converted into a boot table and sent in packets from host to the DSP where the boot code reconstructs the image.

# Boot Runtime – Serial RapidIO

- SRIO boot behaves as supported on previous devices for DirectIO mode. For Nyquist-Shannon, boot using Messaging Mode is supported as well, provided the host sends Ethernet IP messages.
- The boot ROM will not configure the SERDES or SRIO if the boot options in the SRIO boot parameter table show that configuration bypass is enabled. This will be the case in hard or soft reset when reset isolation is enabled and the SRIO and SERDES have already been enabled.
- SERDES and SRIO register configurations are based on templates. Basic values are taken from the template and modified to match the provided input frequency, number of lanes, and output frequency.
- SERDES: The SERDES is configured before the SRIO. The boot ROM programs the SRIO_SERDES_CFGPLL register, the SRIO_SERDES_CFGRX registers, and the SRIO_SERDES_CFGTX registers. The values programmed into these registers are based on the configurable parameters in the SRIO boot parameter tables.
- In addition, for message booting over SRIO, the RIO_RXU_MAP00, RIO_RXU_MAP00_H and RIO_RXU_MAP00_QID registers are programmed. They are programmed to route all messages (promiscuous) to queue 896 using flow ID 0.
- QMSS: The QMSS is configured to manage descriptors using a single memory region. Each descriptor is sized to 48 bytes with extended packet information block present (EInfo). The receive configuration is identical to that of Ethernet and a single function is used for both configurations.
- For DirectIO mode, the DSP will poll the boot magic address. Once this address is populated, the DSP branches to the address specified in the boot magic address. For Messaging Mode, the operation is equivalent to Ethernet Boot

# Boot Runtime – SPI

- The SPI is configured as directed by the boot parameter table. The SPI is operated through direct register reads and write.
- The boot ROM initializes the peripheral and begins reading blocks of data starting at the address specified in the boot parameter table (see Table 34). The ROM reads the data in blocks. Each block consists of  two 16 bit word headers.
- The data in the blocks must contain boot table data. The data might also start with new boot parameter table. The new boot parameter block would specify a higher clock speed and a new boot address for the actual boot data.

# Boot Configuration – PCI Express

- The bootloader configures the base address registers, the number of windows, and their size.
- The PCIe power-up is configured through the external pin.
- If the PCIe boot is the primary boot, the BAR size configuration is driven by the BAR config fields as below.
- Once the BAR configurations are done, the host can access to the memory map of the DSP and image is loaded into the memory.
- At the end the boot magic address is also set to the entry point of the image.
- The DSP is brought out of IDLE by either a MSI interrupt or a legacy interrupt. (See PCIe user guide for further details.)

# Device Startup Summary

| ROM Boot Code Action | | | |
|---|---|---|---|
| Reset Type | Hibernation | Core | Action |
| Power ON | NA | 0 | System Initialization, Full Boot |
| | | Not 0 | Set boot complete, idle, wake then branch |
| Hard, Soft | H-1 | 0 | Restore MSMC, DDR, L2 MMRs, clear PWRSTATECTL, Branch to application defined location |
| | None, H-2, Standby | 0 | Clear PWRSTATECTL, Branch to application defined location |
| | All | Not 0 | Wait for PWRSTATECTL = 0, Branch to application defined location |
| Local | | All | Branch to application defined location |
| ROM Re-entry | | All | Trap |

# Boot Runtime
# I2C Master Mode

- Master Mode: In master mode the boot ROM reads blocks from the I2C EEPROM. The first 4 bytes are the header with the size of the block and the checksum.
  - The first block will be a **Boot Parameter Table**. They will set the proper PLL configuration for the core clock settings. Each block will be 128 bytes in size and the boot parameter table will specify the next block to read. The next block can be another boot parameter table or a boot table or a boot config table.
  - In case of **Boot Table Mode,** blocks are read from the I2C, the 4 byte header is stripped, and the data is passed to the boot table processing function. The boot code will parse the boot table and determines the start address. The code then populates the BOOT_ADDRESS register. Once the boot table is completely parsed the boot code branches to the address in the BOOT ADDRESS register and brings the DSP corePAC out of idle and execute the code.
  - In **Config Table Mode**, the data read from the I2C contain configuration tables. Each element in the table consists of three 32 bit fields.
    - This mode is typically used to poke registers needed before boot can be run, or to execute functions from a previously loaded boot.
    - Each entry in the table falls into one of three types.
      - Standard Entry for read-modify-write of an address
      - Branch entry for a function call to the specified address
      - Table Terminate to end and re-run boot

# Boot Runtime
# I2C Master-Broadcast Mode

- Master-Broadcast: If enabled, the DSP will re-broadcast the boot image loaded from I2C to all passive devices. This is used in case we have multiple DSPs in a same system and one DSP acts as a master driving the other DSPs.

# Boot Runtime
# I2C Passive Mode

- Passive Mode:  the boot ROM operates the I2C device in receiver mode.
  - The header format for passive mode is 19 xx xx yy yy zz zz . Where 19 is the I2C slave address, xxxx is the length, yyyy is Checksum and zzzz is the boot option.
  - The boot option can be again boot parameter table, boot table or a boot config table.