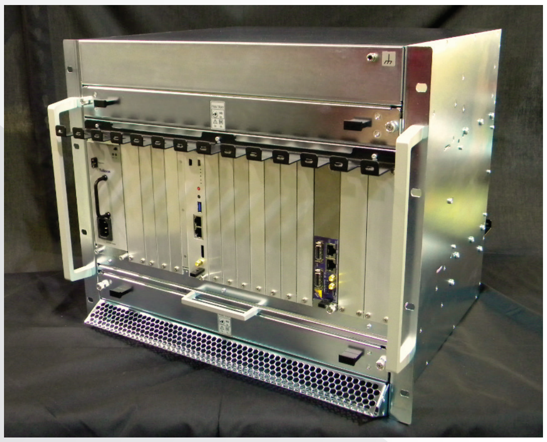
Solving MicroTCA Systems Integration Challenges

Written by: Todd Harrington, Principal Engineer

OVERVIEW

There are many challenges engineers face when integrating COTS third party hardware, especially incorporating the TI Multicore SOC into a MicroTCA carrier. The initialization process, testing Serial Rapid I/O (SRIO) and executing register data dumps in a MicroTCA environment can be complicated, if not time-consuming. This article focuses on some of these challenges and offers solutions that will allow engineers to start rapid development and accelerate deployment on this platform without spending vast resources.

MICROTCA INITIALIZATION: How it Works



The MicroTCA specification states that an Advanced Mezzanine Card (AMC) module must have a Module Management Controller (MMC). The MMC receives 3.3V management power after the Carrier Manager on the MicroTCA Carrier Hub (MCH) is initialized. The AMC MMCs and system Enhanced Module Management Controllers (EMMCs) can then be powered up and the MCH can communicate with it over the Intelligent Platform Management Interface (IPMI) bus. The Carrier Manager communicates with the MMC on each AMC to obtain required payload (12V) power as well as the topologies implemented on the fat and extended fat pipe regions (i.e. Ethernet, SRIO, PCI Express, etc). The Carrier Manager can then make a decision if it will apply payload power to the AMC as well as decide if it should enable the fat or extended fat pipe regions located on the MCH board.

CHALLENGE #1: MMC Functionality on EVMK2H is not Supported

Earlier generation Keystone-I boards use the MSP430 microcontroller as the MMC. The appropriate MMC code is implemented in these boards for proper communication with the MCH. The Evaluation Module (EVM) Keystone-II board uses a Baseboard Management Controller (BMC) to perform EVM initialization. The BMC on the Keystone-II EVM board does not currently implement MMC functionality. After reviewing the BMC code it appears as though the MMC functionality was intended to be implemented by TI but was not. Due to the fact that the TI EVMK2H does not have MMC firmware implemented, it cannot provide the Carrier Manager on the MCH the information it needs to apply payload power to the EVM board. Therefore, the EVM board will not power up when the Carrier is powered on.

The MCH queries each Field Replaceable Unit (FRU) during Carrier initialization to find out what its required payload power consumption (in Amps) is. The MCH will then make the decision, based on an internal algorithm, if it will apply payload power to the FRU. This decision to power up the EVM board cannot be made since that MMC code is not implemented. There are signals connected radially between the MCH and each AMC so the MCH knows the AMC is plugged in. From there, it continuously tries to communicate with the MMC on that AMC.

SOLUTION:

An easy fix for this dilemma can include overriding the MCH to apply payload power to the AMCs when it parses its mch.cfg file. The MCH can apply power to a FRU and allocate a specified number of Amps to that AMC. The parsing of this line will cause payload power to be applied to the AMC regardless of the presence of an MMC.

CHALLENGE #2: EVM Board BMC Bug

With the entry in the mch.cfg we can automatically apply payload power to an EVM installed in an AMC slot. The common problem faced during this stage is that the board may still not boot up due to a bug in the BMC code. The BMC code on the EVM executes once it receives 3.3V management power. It then polls a signal on the power sequencing controller that goes active when 12V payload power is actually applied. When this payload present signal goes active, the BMC code continues and the EVM board boots up. The bug causes the BMC code to check for the payload voltage present signal for a short period of time then exits with an error. If payload power is applied after this loop has exited, the EVM board will not boot.

Typically, the time required from when the MCH applies management power to the EVM and when the MCH parses the mch.cfg file to power on the EVM is greater than this loop time. Once the BMC code exits, the user will need to press the reset button each time on the EMVK2H for the BMC code to re-execute. Once reset, the payload power will be present and the EVM board will boot up.

SOLUTION:

The BMC code for the EVMK2H can be written so that it will poll for an extended duration, thus assuring that the EVM board will boot up without having to press the reset button each time.

CHALLENGE #3: SRIO Port Initialization

The MicroTCA carrier MCH includes an Ethernet base fabric switch with ports routed to each AMC site. It also has an optional extended fabric switch such as SRIO or PCI Express. Different AMCs may have different topologies on their fat and extended fat pipe regions (i.e. AMC ports 4 through 11). The Carrier Manager function of the MCH has the job of communicating with the MMC on each AMC to find out which extended fabric interface is implemented.

The process of matching the fabric interface on the MCH switch with that on the AMC is called E-Keying. If the interface on the AMCs FAT or extended FAT pipe region is compatible with the interface for the respective ports on the MCH switch, the Carrier Manager will enable the ports on its switch.

For example: The TI EVMK2H is used in an 11890-035 (9U) or 11850-020 (5U) chassis, the backplane routes the NAT-MCH SRIO switch signals to extended fat pipe ports 8 through 11 on each of the AMC sites. The TI EVMK2H connects the SRIO interface to AMC extended fat pipe ports 8 through 11, making it compatible with the SRIO fabric in the system. The 11890-035 and 11850-020 chassis route each of the extended fat pipe AMC ports to the MCH SRIO switch port where the MCH has a configurable Field-Programmable Gate Array (FPGA) that will connect SRIO to ports 8 through 11. Unfortunately, the Carrier Manager cannot communicate with the MMC on the TI EVMK2H to ask which fabric interfaces are used on the fat pipe or extended fat pipe region. Without the ability to communicate, the Carrier Manager cannot enable the respective MCH SRIO switch ports connected to each EVMK2H AMC.

SOLUTION:

NAT has resolved the issue by allowing an entry in the mch.cfg startup configuration file that will automatically enable the respective SRIO switch port (4 lanes) connected to the TI EVMK2H.

```
srio_port_init = switch_dev, port, speed

switch_dev:     switch device number 0,1
port:           physical switch port number
speed:          port speed
                0 - 1.25 Gbaud
                1 - 2.5 Gbaud
                2 - 3.125 Gbaud
                3 - 5.0 Gbaud
                4 - 6.25 Gbaud
```

Parameters for each entry change per each type of MicroTCA system. The 11850-020 (5U) chassis model will have different switch_dev parameter than the 11890-035 (9U) chassis model. The port parameter is obtained by selecting the correct port from the table below (Table 1 - NAT-MCH SRIO Switch Port to AMC Mapping). This table is included in the NAT MCH Hub Module Hardware Manual.

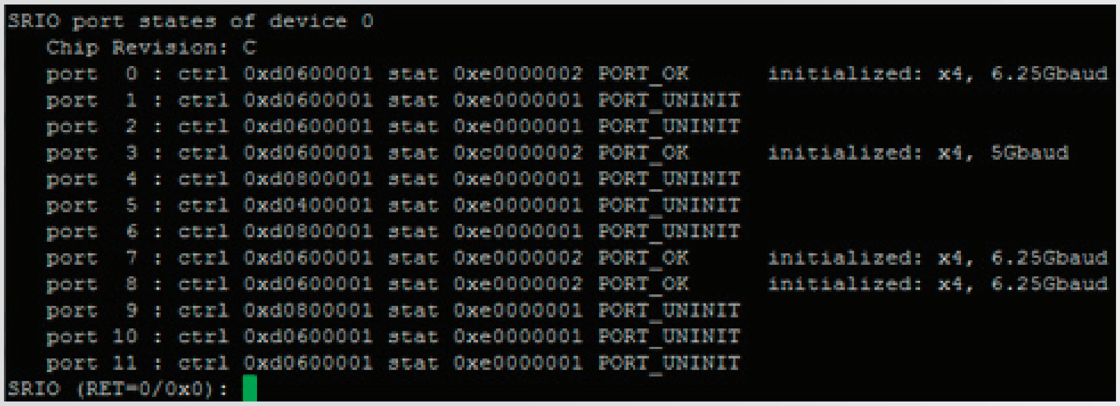
Port #	Assignment - Switch 1	Assignment - Switch 2
0	1st switch interconnect	AMC9
1	AMC2	AMC7
2	Uplink1	1st switch interconnect
3	AMC5	-
4	AMC3	-
5	MCH update	3rd switch interconnect
6	AMC6	AMC12
7	2nd switch interconnect	AMC10
8	3rd switch interconnect	AMC8
9	AMC1	-
10	Uplink2	2nd switch interconnect
11	AMC4	AMC11

Table 1 – NAT-MCH SRIO Switch Port to AMC Mapping

Speed is another important element that will need to be defined in the parameter. Thorough understanding of these systems and their required elements are the keys to successful SRIO initialization.

STATUS CHECK:

After the MicroTCA shelf is powered up, the mch.cfg will be parsed and the SRIO port (4 lanes) will be enabled. It is then the responsibility of the TI EVMK2H software to enable the SRIO port on the board. There are various software examples that use the TI SRIO driver to accomplish the task. Once the TI EVMK2H software has initialized the SRIO interface, the status of the SRIO link can be checked on the NAT-MCH switch (Figure 2 – NAT-MCH SRIO Port Status).



```
SRIO port states of device 0
   Chip Revision: C
   port  0 : ctrl 0xd0600001 stat 0xe0000002 PORT_OK     initialized: x4, 6.25Gbaud
   port  1 : ctrl 0xd0600001 stat 0xe0000001 PORT_UNINIT
   port  2 : ctrl 0xd0600001 stat 0xe0000001 PORT_UNINIT
   port  3 : ctrl 0xd0600001 stat 0xc0000002 PORT_OK     initialized: x4, 5Gbaud
   port  4 : ctrl 0xd0800001 stat 0xe0000001 PORT_UNINIT
   port  5 : ctrl 0xd0400001 stat 0xe0000001 PORT_UNINIT
   port  6 : ctrl 0xd0800001 stat 0xe0000001 PORT_UNINIT
   port  7 : ctrl 0xd0600001 stat 0xe0000002 PORT_OK     initialized: x4, 6.25Gbaud
   port  8 : ctrl 0xd0600001 stat 0xe0000002 PORT_OK     initialized: x4, 6.25Gbaud
   port  9 : ctrl 0xd0800001 stat 0xe0000001 PORT_UNINIT
   port 10 : ctrl 0xd0600001 stat 0xe0000001 PORT_UNINIT
   port 11 : ctrl 0xd0600001 stat 0xe0000001 PORT_UNINIT
SRIO (RET=0/0x0):
```

Figure 2 – NAT-MCH SRIO Port Status

The details of these registers are shown in Chapter 10 of the CPS-1848 manual. These are very helpful when troubleshooting SRIO negotiation problems.

The demo program used was the SRIO_TputBenchmarkingK2HC66TestProject. This sample project is included with MCSDK version 3.00.04.18.

TESTING: Details of SRIO Throughput Demo Program

The SRIO_TputBenchmarkingK2HC66TestProject demo program is a Producer/Consumer demo program that communicates between two tasks, each running on a DSP core. This test can be run in one of three modes (a # define in one of the .h header files):

- It can run in loopback mode on a single EVM board by putting the SERDES in loopback mode.
- It can be setup to use a TI SMA breakout board on each EVM board and cable between the two boards
- It can be setup to have each EVM board communicate SRIO through a NAT-MCH with SRIO switch option.

The demo itself performs 3 types of transfers:

- NWRITE
- NREAD
- TYPE 11

For each transfer type, packet sizes of 4 bytes up to 8192 bytes are transmitted by the Producer task running on one EVM board's DSP core and received by the Consumer task running on a DSP core on another EVM board's DSP core. All transfers are performed over SRIO from DSP core to DSP core.

The demo runs fine with minimal changes using the internal SERDES loopback mode. When operating the demo program through the NAT-MCH switch, there are a few changes required to get the software example to work in NREAD mode and Type 11 modes.

SRIO Routing

In order for the Consumer and Producer tasks running on separate AMC boards to communicate with one another, the SRIO routing table must be setup correctly on the SRIO switch. There are various ways routing table entries can be setup. In this example, the routing table is setup manually in the mch.cfg file. NAT provides commands that can bit bang the registers on the CPS-1848 SRIO switching chips. By using these commands in conjunction with the CPS-1848 manual, one instructs the switch which SRIO port to route each SRIO destination ID out to.

CHALLENGE #4: Register Dump Program

During SRIO software debugging, more often than not, the developer will need to query SRIO registers on the CPS-1848 switch chip. Useful registers are Transmit and Receive Counters for each SRIO switch port as well as Error Counter registers for each switch port. NAT provides a hidden menu (accessible by console or telnet) that allows the user to look at an individual CPS-1848 SRIO switch registers or ALL of the CPS-1848 registers. On occasion, it may be useful to look at a subset of registers that are pertinent to selected SRIO switch ports. Querying these selected registers can be problematic, causing delays in deployment time.

SOLUTION:

Axiomtek Systems has developed an effective method for Linux or Windows to query these selected registers. It only requires a one-time setup of the configuration file. From there, the appropriate verbose register dumps can be viewed in an easy to read manner. This dramatically reduces development time.

AXIOMTEK SYSTEMS AXuTCA-100 PLATFORM

Axiomtek Systems, in collaboration with Schroff, NAT, and TI, has developed the AXUTCA-100, a MicroTCA-based, application ready platform based on the TI EVMK2H Keystone-II Multicore DSP Board. This unique solution eliminates a lot of risks, time, and money associated with 3rd party hardware and software integration, allowing your project schedule to safely stay on track.

The AXuTCA-100 is a modular, scalable, cost effective small form factor subsystem, allowing for a low risk migration from development to deployment. In contrast to other high performance and bandwidth limited solutions, the AXuTCA-100 brings contemporary technologies to standards-based applications without the price tag. Suitable applications include: command and control, aerospace surveillance, land mobile communications and maritime networks, which must all collect and manage large amounts of data in real time.

At the heart of the AXuTCA-100 is the TI Keystone II Multicore SOC. The EVMK2H is a double wide AMC board that can be used on the bench with the provided 12V supply. It can also be used in a MicroTCA shelf where SRIO communicates through DSP cores – or with other devices connected to SRIO (i.e., FPGA). Serial Rapid I/O on this platform supports SRIO speeds up to 20 Gbps and is ideal for various applications including wireless base station infrastructure, video, server, imaging, military and industrial control.

This application-ready platform can be purchased from Axiomtek Sytems with reduced risk of start-up issues and incompatibilities. The tested platforms consists of a 5U or 9U MicroTCA Carrier, a NAT MCH with an integrated 12-port Ethernet Switch and 12-port Serial Rapid I/O switch, and two TI EVMK2H Keystone II Multicore DSP boards.

FOR FURTHER PROJECT ASSISTANCE

Based on years of experience assisting our customers integrating a variety of platforms for various applications, Axiomtek Systems has developed ways to overcome all of the above challenges listed in this article to help ease our customer's burden, enhance efficiency, and achieve successful and accelerated deployment. For example, Axiomtek Systems has instrumented and customized the BMC code for the EVMK2H to achieve extended poll duration so that the EVM board will boot up without having to be reset. Additional solutions designed to meet all of the TI EVMK2H AND MICROTCA integration challenges are available in further details. Contact us at solutions@axiomtek.com or call us at 978-285-0108 for assistance.

ABOUT AXIOMTEK CO., LTD AND AXIOMTEK SYSTEMS

Axiomtek Co. LTD is a leading designer and manufacturer of mission critical computer products. From its data acquisition and machine control roots, Axiomtek has leveraged the PC evolution by focusing on leading edge innovations for the embedded market. Established in 1990, the company has grown to employ over 800 experienced engineers and operational teams. With more than 60 distribution partners and offices and facilities in key locations around the world, Axiomtek's international presence provides global reach with regional expertise to their industry-focused client base.

Axiomtek offers a wide range of embedded X86 and RISC based embedded boards and systems. From small form factor fanless PCs, touch panel computers, digital signage, medical panel PCs, network appliances to environmentally hardened platforms and gaming products.

Axiomtek Systems is a USA-based business unit of Axiomtek specializing in high value-add systems integration. The business unit concentrates on COTS based solutions for customers deploying mission critical applications. Expertise in bus architectures and form factors include PCIe, MicroTCA, VME, ATX, PC/104, PICO ITX, 3.5", PICMG, Q7, SMARC, COM Express and more.

As an associate member of the Intel® Internet of Things Solutions Alliance, Axiomtek continuously develops and delivers cutting edge solutions based on the latest Intel® platforms.