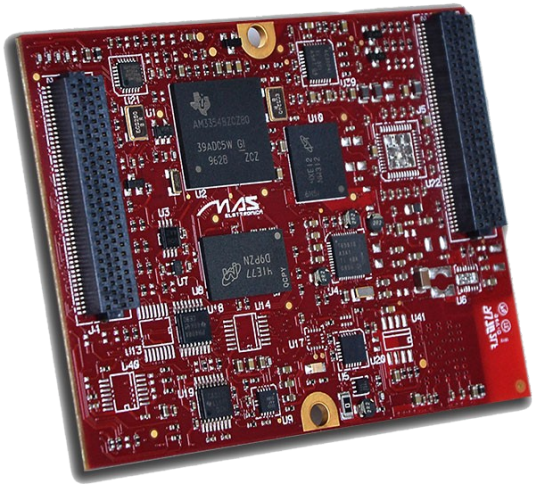


Fiamma AM335x
PRU Application Note

Mas Elettronica Sas
Via A. Rossi 1
35030 Rubano (PD)
Italy

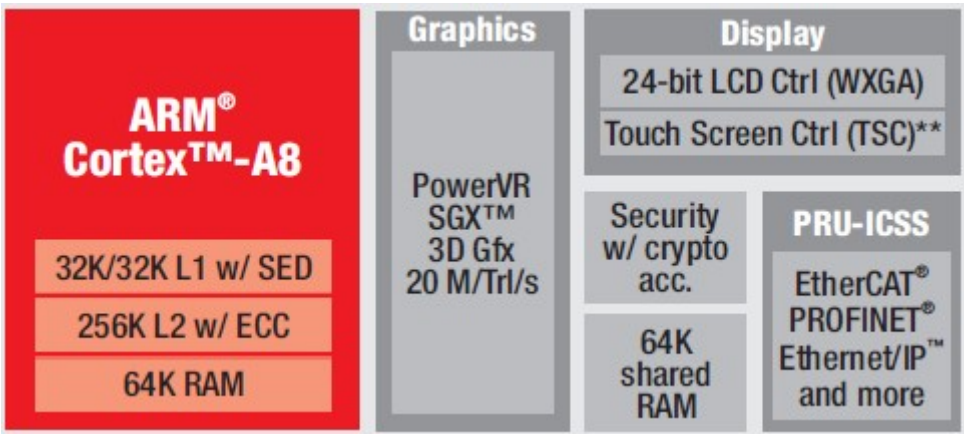
Revision History

Rev.	Document Code	Released	Written	Verified	Approved
1.0	FIAMMA AM335x PRU Application Note		Fabio Molon	Paolo Orsaria	Sandro Macetti

FIAMMA AM335x PRU Application Note

Introduction

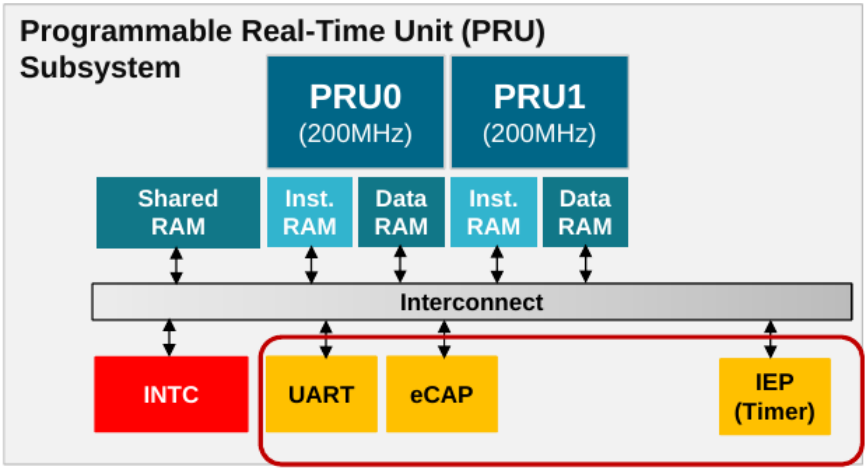
FIAMMA is designed to support industrial, domotic and communication applications with the best tradeoff among cost, reliability and performance. It's based on the proven Texas Instruments processor family Sitara™AM335x ARM Cortex™-A8 featuring clock speed up to 1GHz, 2D/3D graphics accelerator PowerVR SGX™, DDR3 memory, Wi-Fi and Bluetooth, double CAN port. Fiamma supports the AM3359 a particular version of the AM335x family that contains a Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem (PRU-ICSS).



The Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) consists of dual 32-bit RISC (200MHz) real time-cores, shared data and instruction memories, internal peripheral modules and an interrupt controller (INTC). The programmable nature of the PRUs, along with their access to pins and events, provide flexibility in implementing custom peripheral interfaces, fast real-time responses, power saving techniques, specialized data handling and DMA operations, and in offloading tasks from the other processor cores of the system-on-chip (SoC).

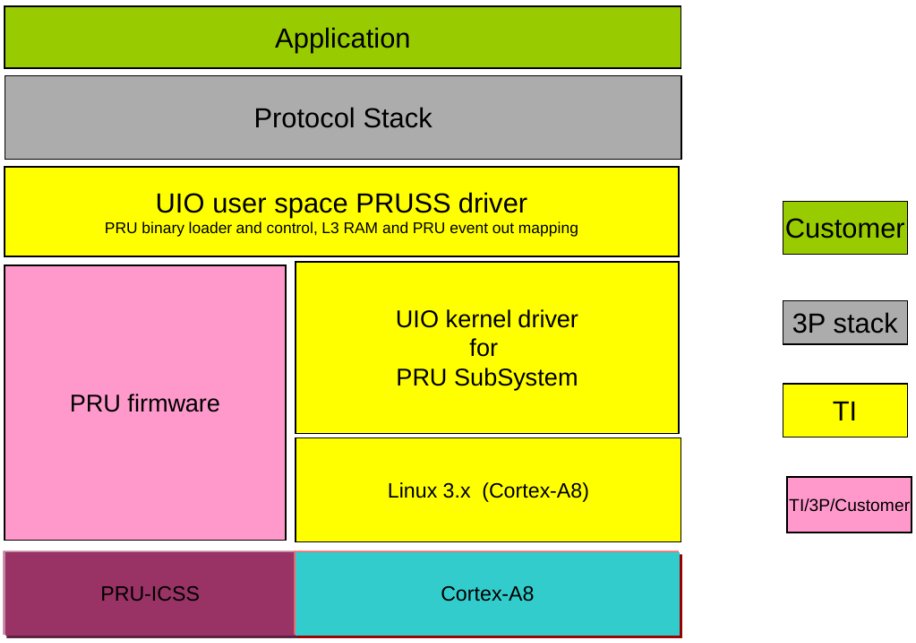
Main characteristics real-time (PRU) cores:

- 8KB of Instruction RAM with Single-Error Detection
- 8KB of Data RAM with Single-Error Detection
- 12KB of Shared RAM With Single-Error Detection
- Supports Protocols such as EtherCAT,PROFIBUS, PROFINET, EtherNet/IP, and more

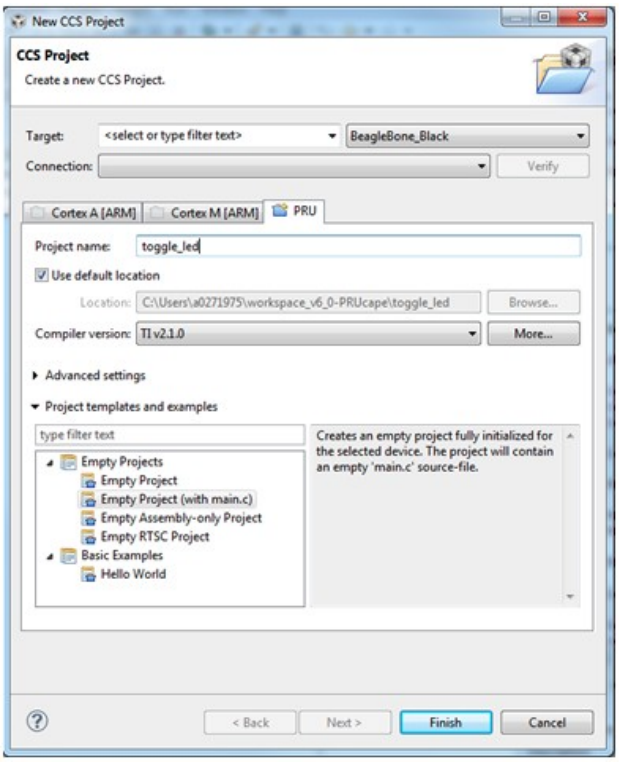


Development software for PRU

The PRU is a subsystem of the processor AM335x. It is an independent CPU with its own memory and instruction set and can run its own program, completely independent from Linux kernel on the main CPU. Before to use the PRU, we need to enable it by kernel. To enable PRU is necessary enable UIO support and Texas Instruments PRUSS driver on menuconfig of kernel. As with all Linux Device Drivers have a choice of building remoteproc and rpmsg into the kernel or building them in as modules.



In order to write programs that use the PRU, you'll need to install IDE Code Composer Studio 6 and PRU compiler (version TI v2.1.0). Open or create new PRU projects just like with any other device. The file Main.c opens automatically.Connect PRU to PC by XDS100v2 JTAG Debug (debug probes (emulators) for TI processors). The debugger XDS100v2 connects to the target board by 20-pin connector and to the host PC via USB2.0. It also requires a free license of Code Composer Studio running on the host PC. When debbuging is necessary implemented the GEL file for configure PRU. The GEL file is the expression language that is used by the CCS debugger. Everywhere you enter in a start address, variable name or condition on a breakpoint it is using GEL to evaluation that C-like expression. Target configurations in CCS often specify a startup script. These scripts are typically used to setup the memory map for the debugger, set any initial target state (by memory or register writes) that is necessary in order to connect the debugger. These scripts are usually written in GEL file. There is also a OnTargetConnect() function that is called when the target is connected. The startup scripts define these functions. Startup GEL files defined in the target configuration file will be automatically loaded when a debug session is started. The PRU application loader for Linux is a software tool which can be used to load a binary to PRU's memory area and to manage the code executed in the PRU from the user space. The software stack consists of two main sections: the user space driver and low level kernel driver. For the convenience of the user, the PRUSSDRV user space library talks to the low level kernel driver (uio_pru), which is responsible for the device setup and the primary interrupt handling. The PRUSSDRV library contains options to start and stop PRU, map PRU and external memories, and manage PRU-generated interrupts. The software architecture of the Linux-based loader is illustrated in follow figure.



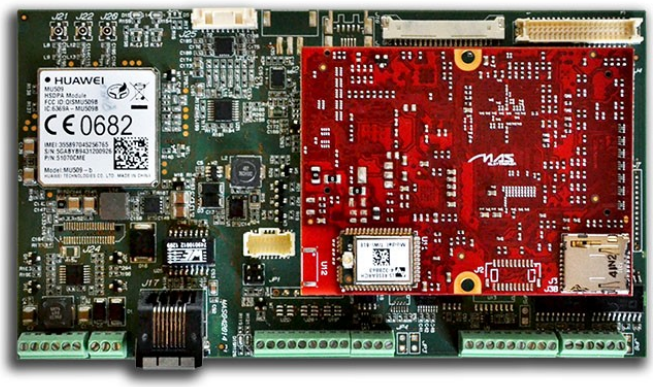
To load our program into the PRU, we'll use a program that runs in Linux user-space on the host CPU, and interfaces with a TI-supplied kernel module and library. A minimal outline of what the program does:

- initialize PRU
- set up the interrupt PRU
- load example.bin from the filesystem into PRU instruction memory and start the PRU
- wait until the PRU asserts the interrupt, telling us the program has completed
- clean up PRU

Most of those steps are reducible to one (or a few) calls into code that are part.

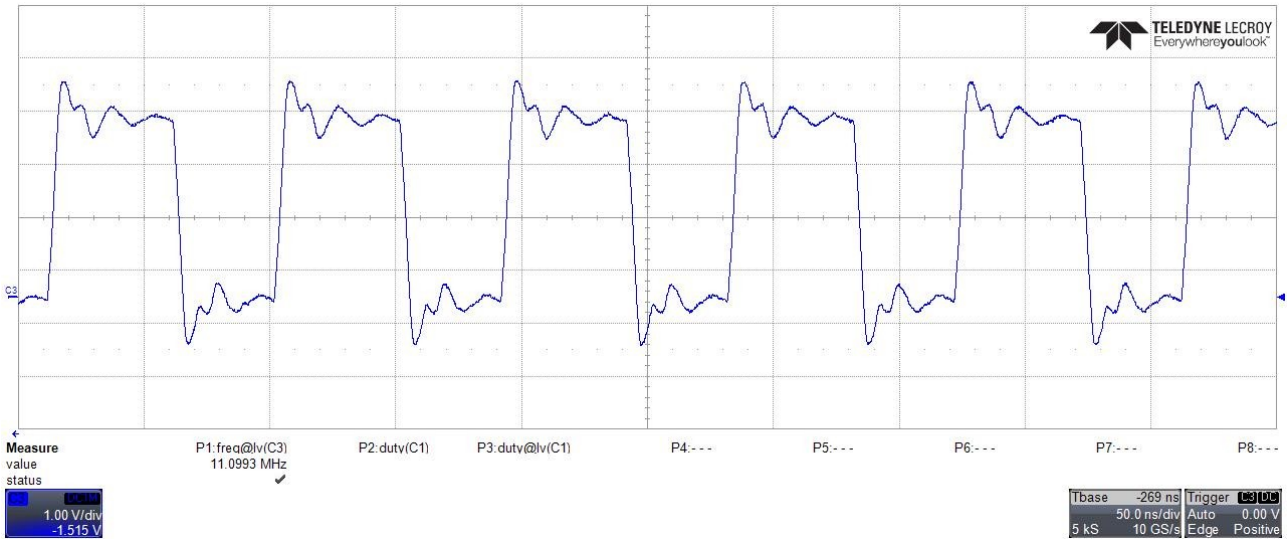
Example project PRU with FIAMMA- and CONTROLBOX

Example 1:



For this design we used our industrial carrier board called control box.

Example GPIO Toggle (frequency signal: 11MHz) with sytem Fiamma and Control Box



Example 2:

Example reading data from memory by SPI (clock frequency: 8 Mhz) the maximum clock for SPI is 48Mhz.