PRU Firmware Development

Building Blocks for PRU Development: Module 2

Agenda

- TI PRU Code Generation Tools
- PRU Register Header Files
- Development & Debug Options

TI PRU Code Generation Tools

Building Blocks for PRU Development: PRU Firmware

PRU Code Generation Tools (CGT)

- PRU CGT is available through CCS v6 App Center.
- Full support of C/C++
- Adds PRU-specific functionality:
  - Takes advantage of PRU architectural features automatically
  - Contains several intrinsics; A full list can be found in the Compiler documentation.
- Full instruction-set Assembler for hand-tuned routines

For more information, visit http://www.ti.com/lit/ug/spruhv7/spruhv7.pdf.

TI PRU CGT Assembly versus C

- Advantages of coding in Assembly over C:
  - Code can be tweaked to save every last cycle and byte of RAM.
  - There is no need to rely on the compiler to make code deterministic.

- Advantages of coding in C over Assembly:
  - More code reusability
  - Can directly leverage kernel headers for interaction with kernel drivers
  - Optimizer is extremely intelligent at optimizing routines:
    - "Accelerating" math via MAC unit
    - Implementing LOOP instruction
    - Etc.
  - Not mutually exclusive; inline Assembly can be easily added to a C project.

PRU Register Header Files

Building Blocks for PRU Development: PRU Firmware

PRU Register Header Files

- Created to make accessing a register easier
- Register names match those in documentation
- Code Completion feature in CCS automatically lists all members
- Developed to allow a user to program at the register-level or at a bit-field level.

  NOTE: Bit-field accesses could potentially cause some issues with other C compilers (e.g., gcc), but register-level should not.
- PRU cregister mechanism used to leverage PRU Constant Table when possible.
- Currently provides definitions for the following registers:
  - **`PRU INTC`**
  - **`PRU CFG`**
  - **`PRU IEP`**
  - **`PRU Control`**
  - **`PRU ECAP`**
  - **`PRU UART`**

PRU Register Headers Example

```
/* PRU_CFG register set */
typedef struct{

    /* PRU_CFG_SYSCFG register bit field */
    union {
        volatile uint32_t SYSCFG;

        volatile struct{
            unsigned IDLE_MODE : 2;
            unsigned STANDBY_MODE : 2;
            unsigned STANDBY_INIT : 1;
            unsigned SUB_MWAIT : 1;
            unsigned rsvd6 : 26;
        } SYSCFG_bit;
    } ; // 0x4

} pruCfg;

volatile __far pruCfg CT_CFG __attribute__((cregister("PRU_CFG", near), peripheral));
```

Excerpt from pru_cfg.h

- Access register directly (CT_CFG.SYSCFG)
- Or access specific bitfields (CT_CFG.SYSCFG_bit.STANDBY_INIT)
- Map the constant table entry to register structures

Example of how to use in C file:

- #include the specific header
- Access registers or fields

```
#include <stdint.h>
#include <pru_cfg.h>

/* Clear SYSCFG[STANDBY_INIT] to enable OCP master port */
CT_CFG.SYSCFG_bit.STANDBY_INIT = 0;
```

Development and Debug Options

Building Blocks for PRU Development: PRU Firmware

PRU Firmware Development

	Inside CCS	Outside of CCS
Upside		
Downside		

PRU Firmware Development

	Inside CCS	Outside of CCS
Upside	• Download and install PRU CGT package via the CCS App Center. • Open or create new PRU projects just like with any other device. • Code Completion helps make register accesses easier.	
Downside		

PRU Firmware Development

	Inside CCS	Outside of CCS
Upside	• Download and install PRU CGT package via the CCS App Center. • Open or create new PRU projects just like with any other device. • Code Completion helps make register accesses easier.	
Downside	• It can be difficult to debug while Linux kernel & user application are running concurrently.	

PRU Firmware Development

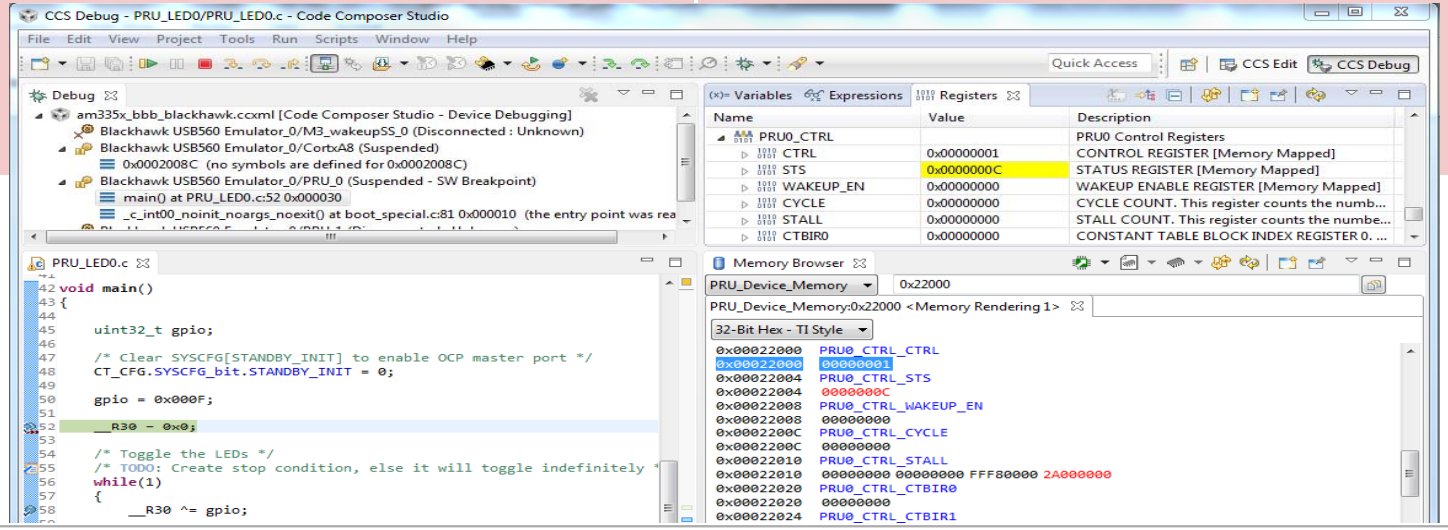
	Inside CCS	Outside of CCS
Upside	• Download and install PRU CGT package via the CCS App Center. • Open or create new PRU projects just like with any other device. • Code Completion helps make register accesses easier.	• Code in your favorite text editor, build via command line. *Linux & Windows CGT packages are available.* • It may be easier to script/automate different processes (build or otherwise).
Downside	• It can be difficult to debug while Linux kernel & user application are running concurrently.	

PRU Firmware Development

	Inside CCS	Outside of CCS
Upside	• Download and install PRU CGT package via the CCS App Center. • Open or create new PRU projects just like with any other device. • Code Completion helps make register accesses easier.	• Code in your favorite text editor, build via command line. *Linux & Windows CGT packages are available.* • It may be easier to script/automate different processes (build or otherwise).
Downside	• It can be difficult to debug while Linux kernel & user application are running concurrently.	• It can be difficult to debug PRU code. • Lacks Code Completion

PRU Debugging

Inside CCS	Outside of CCS
• Easy to view register and variable contents • Access to breakpoints and simple stepping mechanism	



PRU Debugging

Inside CCS	Outside of CCS
• Easy to view register and variable contents • Access to breakpoints and simple stepping mechanism	• Minimal debug control, but some debugfs control provided through remoteproc • Start, halt, single-stepping is all console-based; Clunky when done by hand, but can potentially be scripted.

For More Information

- Visit the PRU-ICSS Wiki: http://processors.wiki.ti.com/index.php/PRU-ICSS
- Download the PRU tools:
  - PRU Software Package — http://www.ti.com/tool/pru-swpkg
  - PRU CGT (Code Gen Tools) — *available through CCSv6 app center*
  - Linux drivers for interfacing with PRU — *available in Processor SDK*
- Order the PRU Cape: http://www.ti.com/tool/PRUCAPE
- For questions regarding topics covered in this training, visit the support forums at the TI E2E Community website: http://e2e.ti.com