Софтуерно дефинирани периферни шини с PRU

Bit banging на съвсем друго ниво.

Димитър Димитров <dimitar@dinux.eu>

10.05.2025

За автора

- Димитър Димитров
- **Innie:** 15 години интегрира AOSP и Embedded Linux в MM Solutions EAD.
  - Bootloaders, Linux kernel drivers, build systems, security, SoC secure boot, ...
- **Outie:** Хоби да сглобява Embedded у-ва и поддържа PRU backend за GCC.

Защо да пиша периферна шина?

- Занимаваш се с Embedded Linux?
- Ще закачаш нестандартна периферия?
  - Имам нужда от 15 `SPI` host контролера.
  - Ще си правя JTAG/SWD адаптер.
  - G-code интерпретатор и управление на 5 стъпкови мотора.
- Обаче не искаш `FPGA`?
- И не ти се занимава с външен $_{\mu}$C?

Решение: bit banging

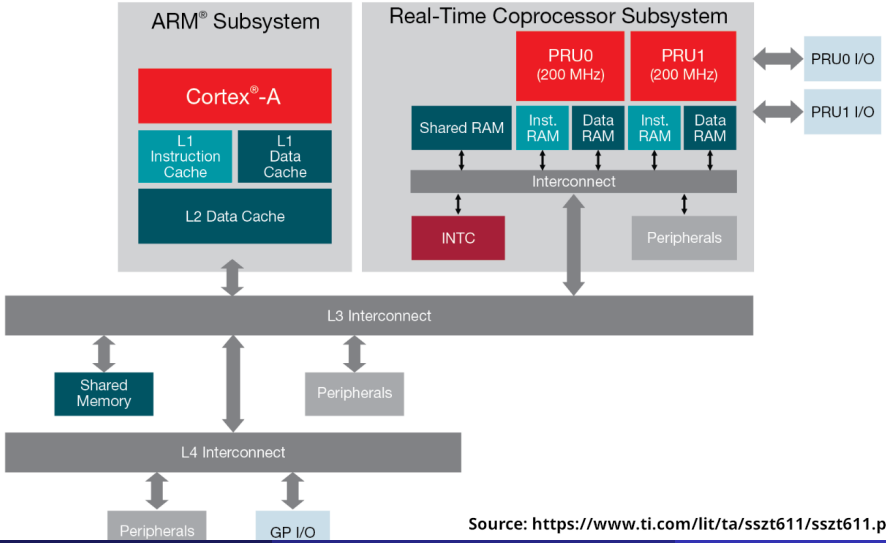
Пример за генериране на 500Hz тактов сигнал с произволно GPIO.

```
void loop()
{
  digitalWrite(8, HIGH);
  delay(1);
  digitalWrite(8, LOW);
  delay(1);
}
```

Какво е PRU?

- 32-bit RISC процесорна архитектура - Programmable Realtime Unit.
- Специализиран за входно/изходни операции.
  - Bit banging, но не само.
- 200MHz тактова честота.
- Включен в някои SoC от Texas Instruments.
  - Beaglebone OSHW едноплаткови компютри.
- 2-4 ядра работят паралелно (I/O, трансфер на данни, обработка).

PRU в SoC



Source: https://www.ti.com/lit/ta/sszt611/sszt611.pdf

За какво мога да го ползвам?

- Софтуерно реализирани периферни шини - bit bang I/O.
  - Повечето инструкции се изпълняват за точно 1 цикъл (5ns).
  - PRU GPIO - достъп с една инструкция (5ns).
  - PRU GPIO - достъп чрез специални CPU регистри.
  - Липсват прекъсвания - за да има детерминизъм при I/O.
  - Без поточни линии (no pipeline).
  - PRU firmware е обикновено комбинация от C и асемблер.

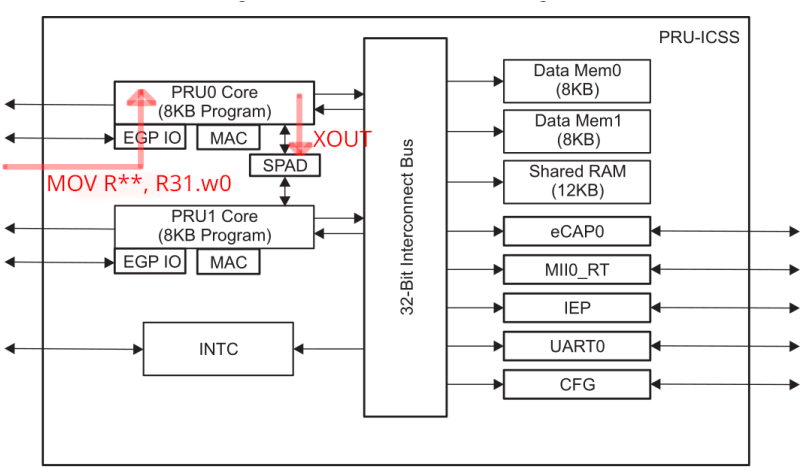
За какво мога да го ползвам?

- Софтуерно реализирано DMA.
  - Достъп до системната шина на SoC.
  - ... включително DDR-SDRAM паметта.
  - Предимство пред решение с външен μC.

PRU0: Пример за четене на GPIO

```
MOV     R21.w0, R31.w0
NOP
 ...
MOV     R28.w2, R31.w0
XOUT    10, &R21, 36
MOV     R21.w0, R31.w0
 LDI    R31, PRU1_PRU0_INTERRUPT + 16
MOV     R21.w2, R31.w0
JMP     $sample100m16$2
```

PRU0: Пример за четене на GPIO

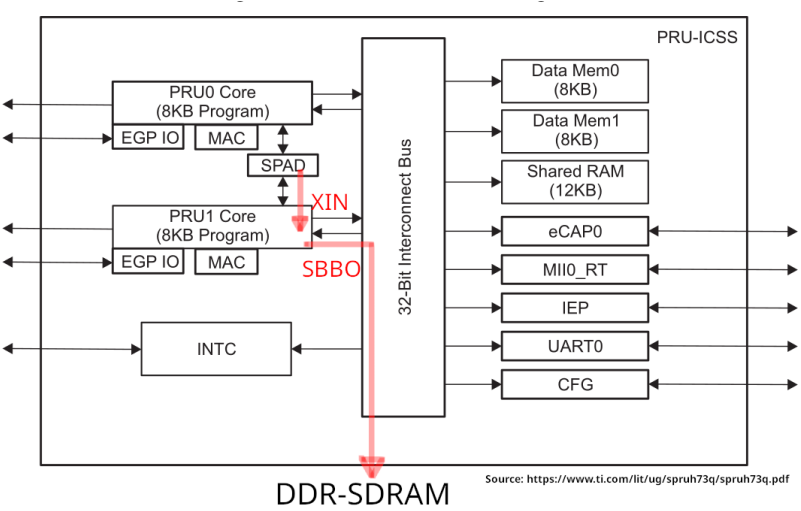


Source: https://www.ti.com/lit/ug/spruh73q/spruh73q.pdf

PRU1: Пример за запис на данни

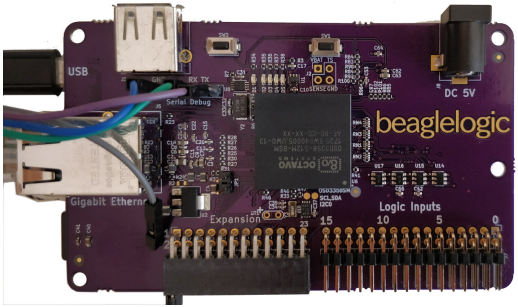
```
# Read 36 bytes from SPAD into R21-R29
XIN     10, &R21, 36
# Store 32 bytes into
# buffer pointed by R18
SBBO    &R21, R18, 0, 32
```

PRU1: Пример за четене на GPIO



Отворени проекти с PRU

- BeagleLogic - 14ch, 100Msps логически анализатор.
  - 100% OSHW, 99% Open Source Software
  - Компилира се с TI Proprietary PRU C Compiler



Gnu for PRU

През 2019г добавих поддръжка за PRU към GNU toolchain.

- Binutils
- GCC
- Newlib
- gnuprumcu

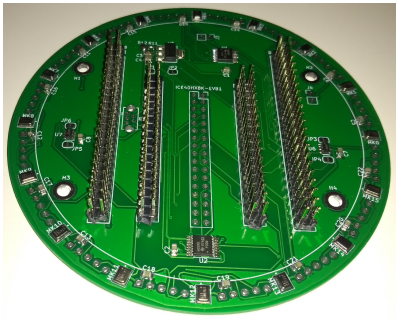
Свобода на 100%! Mainline на 100%!

Позволява да изградите система единствено от свободен софтуер!

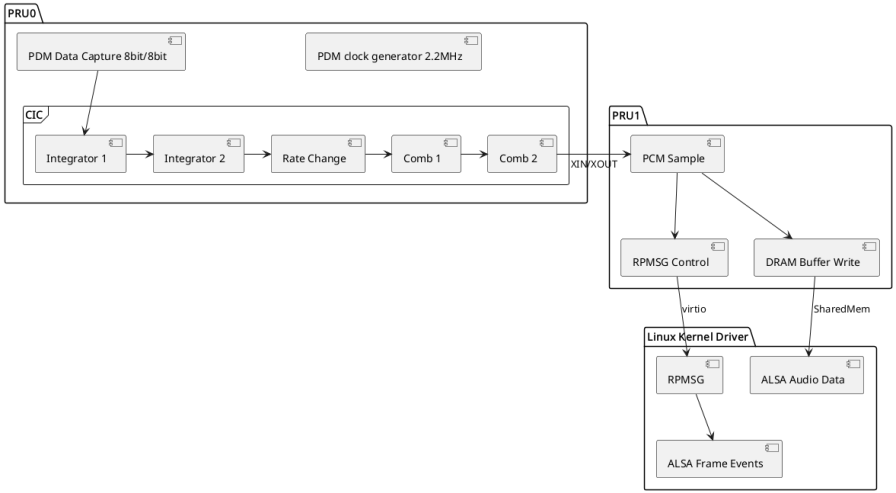
- ☑ SoC Linux BSP - good support in mainline.
- ☑ Remoteproc Linux kernel driver - in mainline.
- ☑ RPMSG Linux kernel driver - in mainline.
- ☑ Binutils + GCC + Newlib toolchain - in mainline.

Моят проект с PRU

- BeagleMic - USB аудио карта с 16 PDM микрофона.
  - 100% OSHW, 100% Open Source Software
  - Компилира се с GCC



Отворени проекти с PRU



Недостатъци

- Един единствен доставчик - Texas Instruments.
- Сложно е (Remoteproc, RPMSG, resource tables)
  - Но пък има множество примери, с които да се започне.
- Примитивен паралелизъм, далеч от възможностите на FPGA.
- Твори се на `C` и `Assembler`.

Полезни връзки

- https://bbb.io/pru
- https://github.com/dinuxbg/gnurpru
- https://github.com/dinuxbg/gnurpru/wiki
- https://github.com/dinuxbg/pru-gcc-examples
- https://github.com/dinuxbg/beaglemic