[redacted]

[redacted]

Notes on device side clocks

The device side firmware uses a range of different clocks. Here is an attempt to document the clocks in use and for what they are used.

Table of Contents

Slow clock

^Top

~32kHz internal RC clock

Can be between 22 and 42 kHz

Main Oscillator / MAINCK

^Top

cf `PMC_MOR` register

16 MHz, based on external Xtal

PLL clock

^Top

cf `PMC_PLLR` register

96 MHz (MAINCK * 12 / 2)

Master Clock MCK, Processor Clock PCK, USB clock UDPCK

^Top

cf `common_arm/clocks.c`

cf `PMC_MCKR` and `PMC_SCER` registers

- MCK starts with RC slow clock (22 to 42 kHz).
- Then MCK configured from PLL: 48 MHz (PLL / 2)

cf `bootrom.c`:

PCK can be disabled to enter idle mode, but on Proxmark3 it's always on, so PCK is also 48 MHz.

USB need to be clocked at 48 MHz from the PLL, so PLL / 2 (cf `CKGR_PLLR`).

Peripheral clocks

^Top

cf `bootrom.c`:

Distribute MCK/PCK? clock to Parallel I/O controller, ADC, SPI, Synchronous Serial controller, PWM controller, USB.

cf `appmain.c`

Activate PCK0 pin as clock output, based on PLL / 4 = 24 MHz, for the FPGA.

1 kHz RTC: TickCount functions

^Top

cf `armsrc/ticks.c`

cf `PMC_MCFR` and `RTTC_RTMR` registers for configuration, `RTTC_RTVR` register for reading value.

Characteristics:

- 1 kHz, 32b (49 days), if used with 16b: 65s
- Configured at boot (or TIA) with `StartTickCount()`
- Time events with `GetTickCount()` / `GetTickCountDeltaDelta()`, see example
- Coarse, based on the ~32kHz RC slow clock with some adjustment factor computed by TIA
- Maybe 2.5% error, can increase if temperature conditions change and no TIA is recomputed
- If TimingIntervalAcquisition() is called later, StartTickCount() is called again and RTC is reset

Usage:

```
uint32_t ti = GetTickCount();
...do stuff...
uint32_t delta = GetTickCountDelta(ti);
```

Current usages:

- cheap random for nonces, e.g. `prng_successor(GetTickCount(), 32)`
- rough timing of some operations, only for informative purposes
- timeouts
- USB connection speed measure

Occasional PWM timer

^Top

- `void SpinDelayUs(int us)`
- `void SpinDelay(int ms)` based on SpinDelayUs
- `void SpinDelayUsPrecision(int us)`

Busy wait based on 46.875 kHz PWM Channel 0

- 21.3 us precision and maximum 1.39 s
- *Precision* variant: 0.7 us precision and maximum 43 ms

Occasional TC0+TC1 / CountUS functions

^Top

cf `armsrc/ticks.c`

About 1 us precision

- `void StartCountUS(void)`
- `uint32_t RAMFUNC GetCountUS(void)`

Use two chained timers TC0 and TC1. TC0 runs at 1.5 MHz and TC1 is

175 lines (113 sloc) | 4.85 KB

Raw

Blame

Can't be used at the same time as CountSspClk or Ticks functions.

Occasional TC0+TC1+TC2 SSP_CLK from FPGA / CountSspClk functions

^Top

cf `armsrc/ticks.c`

About 1 cycle of 13.56 MHz? precision

- `void StartCountSspClk(void)`
- `void ResetSspClk(void)`
- `uint32_t RAMFUNC GetCountSspClk(void)`
- `uint32_t RAMFUNC GetCountSspClkDelta(uint32_t start)` <=

TODO could be used more often

Use two chained timers TC0 and TC1. TC0 runs at SSP_CLK from FPGA (13.56 MHz?) and TC1 is clocked when TC0 loops.

Usage:

- for iso14443 commands to count field cycles
- Also usable with FPGA in LF mode ?? cf `armsrc/legicrfsim.c` SSP Clock is clocked by the FPGA at 212 kHz (sub-carrier frequency)

Can't be used at the same time as CountUS or Ticks functions.

Occasional TC0+TC1 / Ticks functions

^Top

cf `armsrc/ticks.c`

1.5 MHz

- `void StartTicks(void)`
- `uint32_t GetTicks(void)` <= **TODO** why no GetTicksDelta ?
- `void WaitTicks(uint32_t ticks)`
- `void WaitUS(uint32_t us)`
- `void WaitMS(uint32_t ms)`
- `void StopTicks(void)` <= **TODO** why a stop for this timer and not for CountUS / CountSspClk ?

Use two chained timers TC0 and TC1. TC0 runs at 1.5 MHz and TC1 is clocked when TC0 loops.

Maximal value: 0xffffffff = 2863 s (but don't use high value with WaitTicks else you'll trigger WDT)

Usage:

- Timer for bitbanging, or LF stuff when you need a very precise timer

Can't be used at the same time as CountUS or CountSspClk functions.