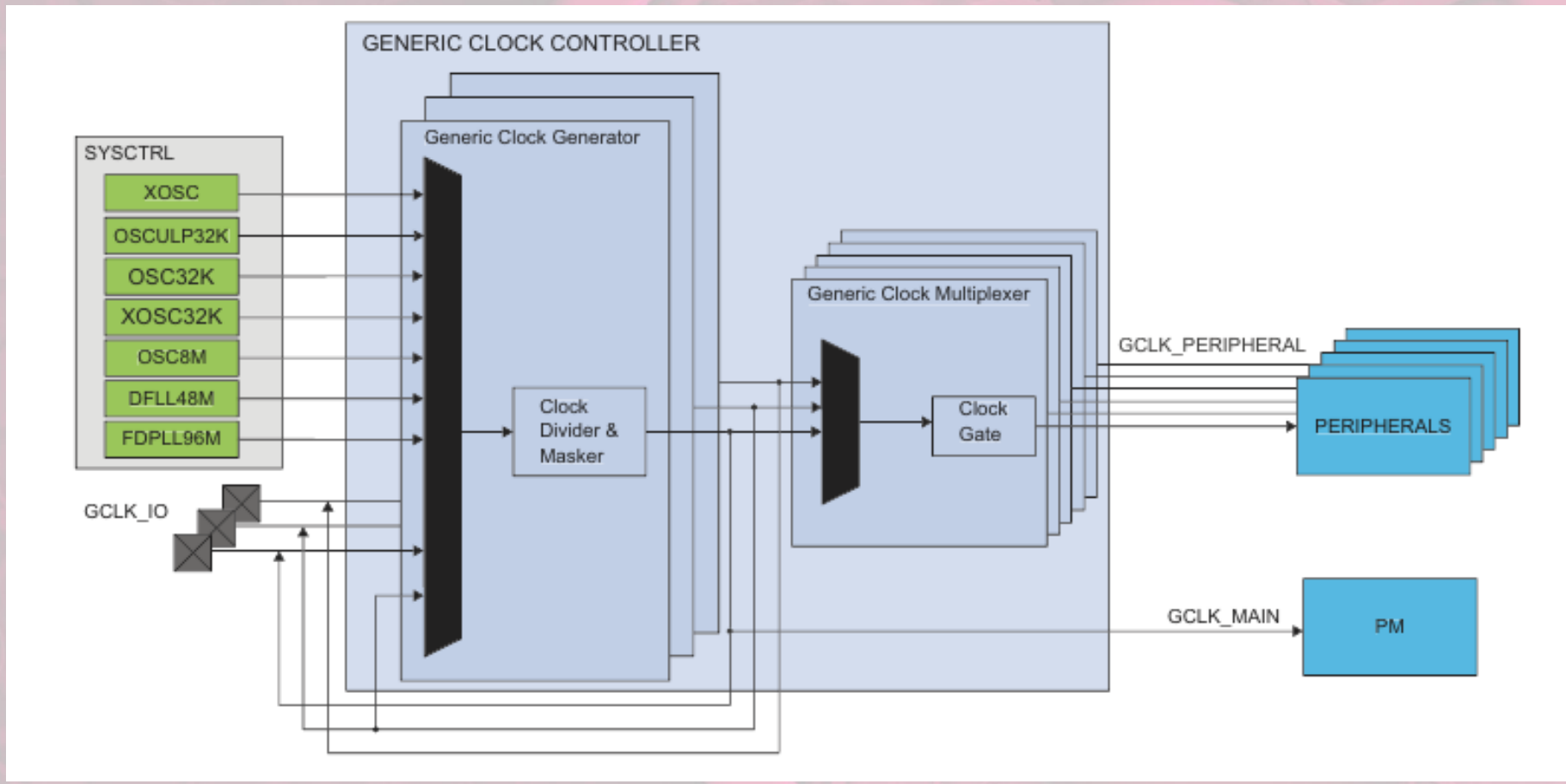
GCLK – Generic Clock Controller

Thomas Müller, muellet5@students.zhaw.ch

Content

- GCLK overview
- Clock sources
- Generic Clock Generator
- Generic Clock Multiplexer
- Synchronization
- Registers
- Programming API
- Demo

GCLK Overview

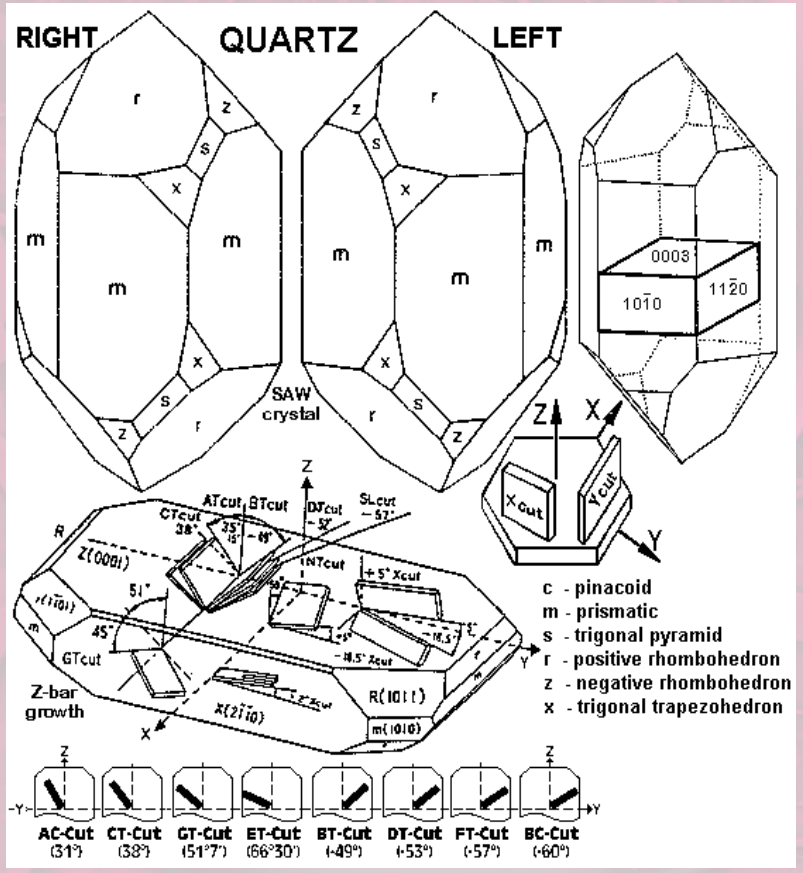


Clock Sources

- **XOSC**: 0.4 – 32MHz tunable crystal oscillator
- **OSCULP32K**: 32.768kHz ultra low power internal oscillator
- **OSC32K**: 32.768kHz high accuracy internal oscillator
- **XOSC32K**: 32.768 crystal oscillator
- **OSC8M**: 8MHz internal oscillator
- **DFLL48M**: 48MHz digital frequency locked loop
- **FDPLL96M**: 48 – 96MHz fractional digital phase locked loop

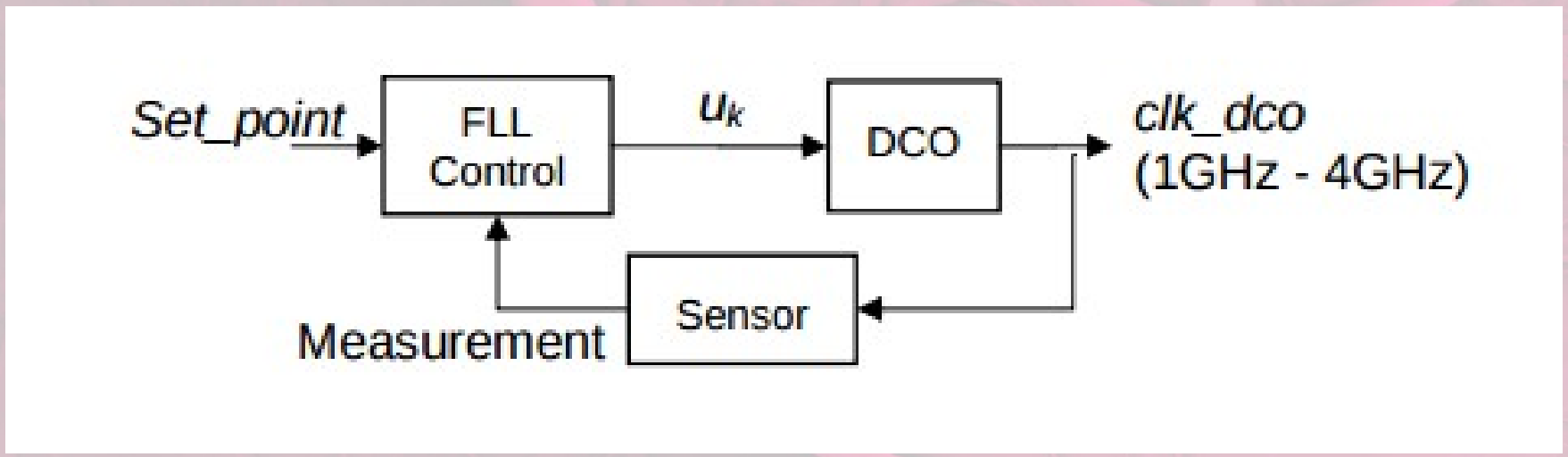
Clock Sources::Crystal Oscillator

- Vibrating crystal of piezoelectric material
- Frequency depending on crystal cut
- Most common application: 32.768kHz
  - Watches
  - Low frequency XY-cut



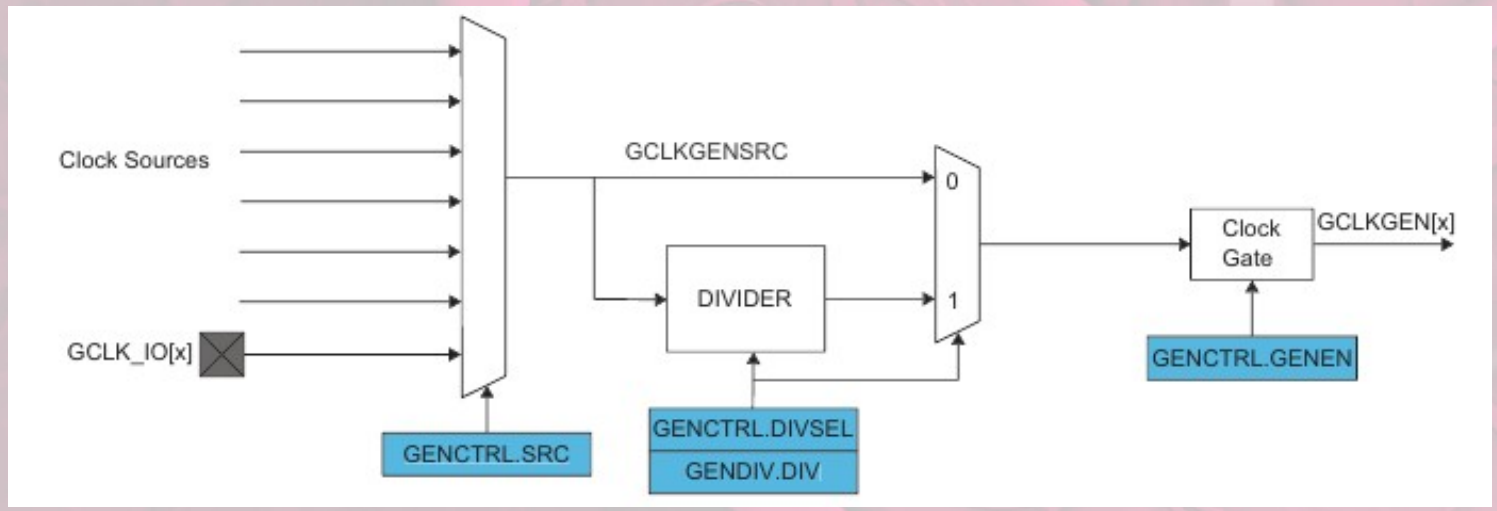
Clock Sources::Digital Frequency Locked Loop

- Electronic control system
- Input: Reference Clock
- Digitally-controlled oscillator (DCO) generates output frequency
- Feedback measurement for FLL control



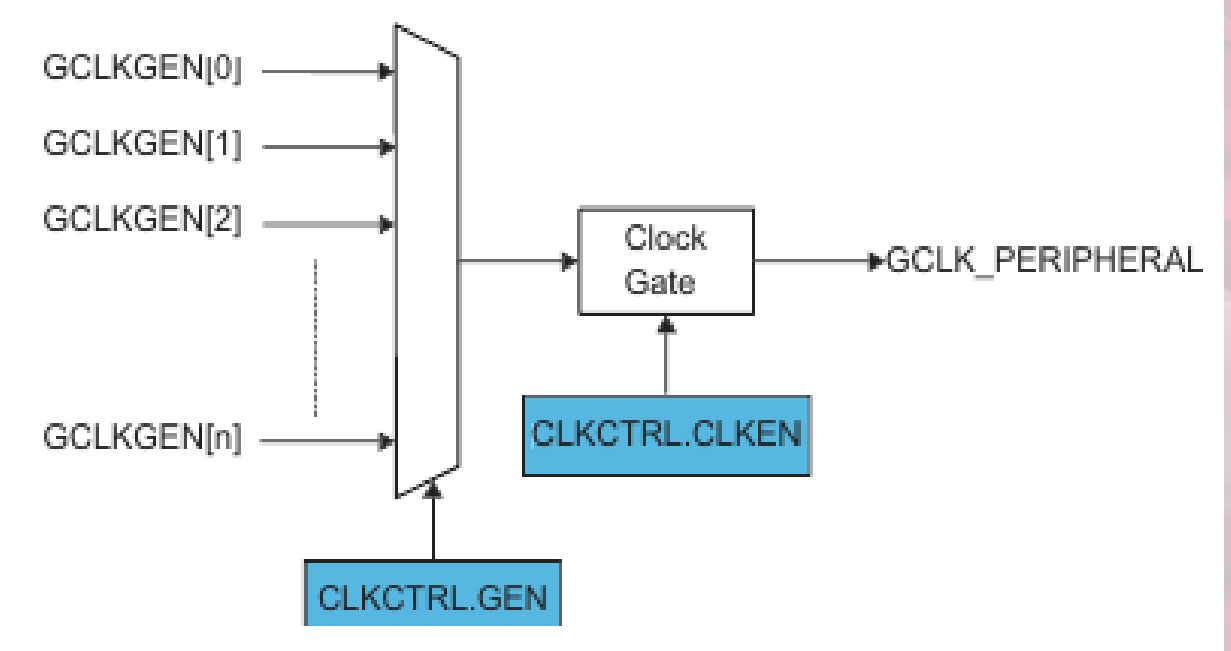
Generic Clock Generator

- Each generator can be set to run from one of eight clock sources
  - Except GCLKGEN[1]: act as source to other generic glock generators



Generic Clock Multiplexer

- Assign a clock source (generator) to peripheral
- Changing source for peripheral requires disabling the generic clock (over CLKCTRL.CLKEN)



Synchronization

- Registers can require synchronization on read / write access
- If synchronization required STATUS.SYNCBUSY is one → bus stalled
- CPU stalled as long as bus stalled

  → no interrupts can be handled
- Refer to processor handbook to find out which register access needs to be sync
- Be careful with slow clock sources (32kHz)

Registers

- **CTRL:** SWRST flag
- **STATUS:** SYNCBUSY flag
- **CLKCTRL**
- **GENCTRL**
- **GENDIV**

Registers::CLKCTRL

- Enable Clock
- Assign Generator to device

Bit	15	14	13	12	11	10	9	8
	WRTLOCK	CLKEN			GEN[3:0]			
Access	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			ID[5:0]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Registers::GENCTRL

- Configure generic clock generator
- Assign clock source to generator

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				SRC[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					ID[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Registers::GENDIV

- Assign division factor to specific clock generator

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIV[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					ID[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Programming API

- Assign clock source to generator:
  - sys::reg::GCLK.src2gen(SRC,GEN,flags);
  - SRC can be OSC8M, OSC32K etc.
  - GEN can be G0 to G8
- Assign clock generator to device:
  - sys::reg::GCLK.gen2dev(GEN,DEV);
  - GEN can be G0 to G8
  - DEV can be ADC, TC (Timer Counter) etc.
- Set division factor:
  - sys::reg::GCLK.gendiv(GEN,DIV);
  - GEN can be G0 to G8
  - DIV is 16Bit division factor

DEMO