Agenda

► Block Diagram
► ISP/ICP
► System Memory Map
► Power Management
► Reset Source
► Clock Control
► Interrupt
► CMSIS
M051 Series Functional Block Diagram

Cortex-M0
50 MHz CLK_ctl

AHB arbiter

SRAM 4KB

AHB arbiter

APB-Bridge

GPIO P0~P4

CLK_CTL

PLL

10K OSC

22M OSC

EXT. 12M XTAL

LDO
2.5 ~ 5.5V

ADC
8ch/12bit SARADC
600K SPS

SPI 0/1

UART 0/1

PWM 0~7

I2C

PAD Control

POR

Brown-Out

LVR

Flash Control

64KB(M0516)
32KB(M058)
16KB(M054)
8KB(M052)

CONFIG

DATAFLASH

ISP 4KB

EBI

AD[15:0]
nCS
nRD
nWR
mclk
ALE
ARM Cortex-M0 Functional Block Diagram

- Cortex-M0 processor
  - Nested Vectored Interrupt Controller (NVIC)
  - Bus matrix
- Debug
  - Breakpoint and Watchpoint unit
  - Debugger interface
- Cortex-M0 components
  - Interrupts
  - Wakeup Interrupt Controller (WIC)
  - AHBLite interface
- Serial Wire or JTAG debug port
  - Debug Access Port (DAP)

4 Break Point
2 Watchpoint
The NUC 100 series IC embed Cortex-M0 core
- running up to 50 MHz
- 32K/64K/128K bytes embedded flash by part No
  - Configurable data flash address and size for 128kB system, fixed 4kB data flash for the 32kB and 64kB system
- 4kB flash for ISP loader
- 4K/8K/16K embedded SRAM

The M051 series IC embed Cortex-M0 core
- 8KB/16KB/32KB/64KB Flash memory for program memory (APROM)
- 4KB Flash memory for data memory (DataFlash)
- 4KB Flash memory for loader (LDROM)
- 4KB SRAM for internal scratch-pad RAM (SRAM)

Support **ISP** (In System Programming) by USB or UART
Support 2 wire **ICP** (In Circuit Programming) update from ICE interface
Support fast parallel programming mode by external writer
2 wire SWD ICE interface
Wide operating voltage ranges from **2.5V to 5.5V**
ICP (In Circuit Programming)

NuMicro ICP AP

USB

Nu-Link

SWD I/F

NuMicro Family

NuMicro Family
ISP (In System Programming)

- NuMicro ISPAP
- RS232 Cable
- USB Cable
- UART I/F
- USB I/F
- M0-CPU
- LDROM
- APROM
- Nuvoton ISP LDROM firmware code
- Boot From LDROM
- USB interface only supports NUC120/NUC140 series
## NuMicro ISP vs ICP Difference

<table>
<thead>
<tr>
<th>Item</th>
<th>ISP</th>
<th>ICP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC AP Software</td>
<td>NuMicro ISP Programmer</td>
<td>NuMicro ICP Programming Tools.exe</td>
</tr>
<tr>
<td>Extra Hardware</td>
<td>No</td>
<td>NuLink</td>
</tr>
<tr>
<td>CPU Run Code</td>
<td>CPU runs on the LDROM</td>
<td>No</td>
</tr>
<tr>
<td>What Region Can Be Updated</td>
<td>• APROM</td>
<td>• APROM</td>
</tr>
<tr>
<td></td>
<td>• DataFlash</td>
<td>• LDROM</td>
</tr>
<tr>
<td></td>
<td>• Config</td>
<td>• DataFlash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Config</td>
</tr>
<tr>
<td>Interface</td>
<td>USB or UART</td>
<td>SWD</td>
</tr>
<tr>
<td>Support Off-Line Programming</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Address Range</td>
<td>Description</td>
<td>Color</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0xE000_0000</td>
<td>Cortex M0 System Register</td>
<td>Yellow</td>
</tr>
<tr>
<td>0xE000_0000</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x501F_0000</td>
<td>AHB peripherals</td>
<td>Light Blue</td>
</tr>
<tr>
<td>0x5000_0000</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x401F_0000</td>
<td>APB peripherals</td>
<td>Light Blue</td>
</tr>
<tr>
<td>0x4000_0000</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>RAM</td>
<td>Pink</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0010_0000</td>
<td>ISP Loader Program Memory (LDROM)</td>
<td>Yellow</td>
</tr>
<tr>
<td>0x0010_0000</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0001_0000</td>
<td>Application Program Memory (APROM)</td>
<td>Yellow</td>
</tr>
<tr>
<td>0x0000_0000</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
Power Management

- **Normal Run Mode**
  - Flexible system clock source selection
  - All peripherals clock can be turned off individually.

- **Sleep Mode (IDLE Mode)**
  - CPU halt, peripheral is probably under running which depends on your application

- **Deep Sleep Mode (Power Down Mode)**
  - CPU & peripheral are all halt
Power Management – Cont.

► Sleep mode

UNLOCKREG();
SCB->SCR = 4;
SYSCLK->PWRCON.WINT_EN = 0;
SYSCLK->PWRCON.PD_WAIT_CPU = 1;
SYSCLK->PWRCON.PWR_DOWN = 0;
LOCKREG();
__WFI();

► Deep Sleep mode

UNLOCKREG();
SCB->SCR = 4;
SYSCLK->PWRCON.WINT_EN = 0;
SYSCLK->PWRCON.PD_WAIT_CPU = 1;
SYSCLK->PWRCON.PWR_DOWN = 1;
LOCKREG();
__WFI();
System Reset

► Hardware Reset
  – The Power-On Reset
  – /RESET pin Reset
  – Watchdog Time Out Reset
  – Low Voltage Detected Reset (generates a reset when Vcc < Vref.)
  – Brown-Out-Detected Reset (an intentional or unintentional drop in voltage)

► Software Reset
  – CPU_RST
    • To write 1 to the CPU_RST(IPRSTC1[1],"IP Reset Source Register") register
    • Just only reset CPU & flash controller
  – CHIP_RST
    • To write 1 to CHIP_RST(IPRSTC1[0], IP Reset Source Register) register
    • To reset the whole chip like “Power-on reset”
  – MCU_RST
    • Write 1 to the SYSRESETREQ (AIRC[2], Application Interrupt & Reset Control Register”)
    • To reset the whole chip

“RSTSRC” register identify chip’s reset source from last operation
RSTRC (System Reset Source Register)

<table>
<thead>
<tr>
<th>Register</th>
<th>Address</th>
<th>R/W</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSTSRC</td>
<td>0x5000_0004</td>
<td>R/W</td>
<td>System Reset Source Register</td>
<td>0x0000_00xx</td>
</tr>
</tbody>
</table>

These bits are cleared by writing 1 to itself.
## System Reset Resource Table

<table>
<thead>
<tr>
<th>Reset Status</th>
<th>Reset Source</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RSTS_CPU</td>
<td>Reserved</td>
<td>RSTS_MCU</td>
<td>RSTS_BOD</td>
<td>RSTS_LVR</td>
<td>RSTS_WDG</td>
<td>RSTS_PAD</td>
<td>RSTS POR</td>
</tr>
<tr>
<td>Power-On Reset</td>
<td></td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Reset Pin Reset</td>
<td></td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Watchdog Time Out Reset</td>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LVR Reset</td>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brown-Out Reset</td>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cortex-M0 MCU Reset</td>
<td></td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CHIP_RST</td>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CPU_RST</td>
<td></td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Peripheral IP Reset

- Every peripheral has corresponded reset register
- "IPRSTC1" & "IPRSTC2" register had defined the corresponded peripheral asynchronous reset signal
NUC100 Clock Generator

- **External Crystal**
  - X32I
  - X32O
  - XT_IN
  - XT_OUT

- **Internal RC**
  - OSC10K
  - OSC22M

- **Clock Generators**
  - XTL32K
    - XTL32K_EN(PWRCON[1])
    - 32K
  - XTL12M
    - XTL12M_EN(PWRCON[0])
    - 12M
  - OSC22M
    - OSC22M_EN(PWRCON[6])
    - 22M
  - OSC10K
    - OSC10K_EN(PWRCON[3])
    - 10K
  - PLL
    - PLL_SRC(PLLCON[19])
    - PLL FOUT
    - 32M

- **Control Signals**
  - PWRCON[0]
  - PWRCON[6]
  - PWRCON[1]
M051 Clock Generator

- **External Crystal**
- **Internal RC**

Diagram showing the clock generation process:

- **XTL12M**
  - EN: PWRCON[0]
  - XT_IN
  - XT_OUT

- **XTL_IN**
  - 12M

- **PLL**
  - PLL_SRC: PLLCON[19]
  - PLL FOUT

- **OSC22M**
  - EN: PWRCON[2]
  - 22M

- **OSC10K**
  - EN: PWRCON[3]
  - 10K

- **External Crystal**
- **Internal RC**
SysTick Clock Source

For the other peripherals IP, they also have multi-clock source to set.
NuMicro NVIC

- NVIC (Nested Vectored Interrupt Controller)
  - An integrated part of the Cortex-M0 processor
  - It supports 32 peripheral interrupts input
  - It supports NMI (Nonmaskable Interrupt) input
  - It supports “Tail Chaining” & “Late Arrival”

- Interrupt handler follows the CMSIS coding rule
C ISR Function Call Name

In “startup_NUC1xx.s” vector table address save peripherals ISR function address
ISR Handler

```c
#include "NUC100.h"
extern uint32_t SystemFrequency;

void TIMER1_IRQHandler(void)
{
    g_timer1Ticks++;
    TIMER1->TSR.TIF = 1;
}

void TIMER2_IRQHandler(void)
{
    g_timer2Ticks++;
    TIMER2->TSR.TIF = 1;
}

void TIMER3_IRQHandler(void)
{
    g_timer3Ticks++;
    TIMER3->TSR.TIF = 1;
}
```
<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Interrupt Number (Bit in Interrupt Registers)</th>
<th>Interrupt Name</th>
<th>Source IP</th>
<th>Interrupt description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ~ 15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>System exceptions</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>BOD_OUT</td>
<td>Brown-Out</td>
<td>Brownout low voltage detected interrupt</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>WDT_INT</td>
<td>WDT</td>
<td>Watch Dog Timer interrupt</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>EINT0</td>
<td>GPIO</td>
<td>External signal interrupt from PB.14 pin</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>EINT1</td>
<td>GPIO</td>
<td>External signal interrupt from PB.15 pin</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>GPAB_INT</td>
<td>GPIO</td>
<td>External signal interrupt from PA[15:0] / PB[13:0]</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>GPCDE_INT</td>
<td>GPIO</td>
<td>External interrupt from PC[15:0]/PD[15:0]/PE[15:0]</td>
</tr>
<tr>
<td>22</td>
<td>6</td>
<td>PWMA_INT</td>
<td>PWM0~3</td>
<td>PWM0, PWM1, PWM2 and PWM3 interrupt</td>
</tr>
<tr>
<td>23</td>
<td>7</td>
<td>PWMB_INT</td>
<td>PWM4~7</td>
<td>PWM4, PWM5, PWM6 and PWM7 interrupt</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>TMR0_INT</td>
<td>TMR0</td>
<td>Timer 0 interrupt</td>
</tr>
<tr>
<td>25</td>
<td>9</td>
<td>TMR1_INT</td>
<td>TMR1</td>
<td>Timer 1 interrupt</td>
</tr>
<tr>
<td>26</td>
<td>10</td>
<td>TMR2_INT</td>
<td>TMR2</td>
<td>Timer 2 interrupt</td>
</tr>
<tr>
<td>27</td>
<td>11</td>
<td>TMR3_INT</td>
<td>TMR3</td>
<td>Timer 3 interrupt</td>
</tr>
<tr>
<td>28</td>
<td>12</td>
<td>UART0_INT</td>
<td>UART0</td>
<td>UART0 interrupt</td>
</tr>
<tr>
<td>29</td>
<td>13</td>
<td>UART1_INT</td>
<td>UART1</td>
<td>UART1 interrupt</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
<td>SPI0_INT</td>
<td>SPI0</td>
<td>SPI0 interrupt</td>
</tr>
<tr>
<td>31</td>
<td>15</td>
<td>SPI1_INT</td>
<td>SPI1</td>
<td>SPI1 interrupt</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>SPI2_INT</td>
<td>SPI2</td>
<td>SPI2 interrupt</td>
</tr>
</tbody>
</table>
## System Peripheral Interrupt Map (33~47)

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Interrupt Number (Bit in Interrupt Registers)</th>
<th>Interrupt Name</th>
<th>Source IP</th>
<th>Interrupt description</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>17</td>
<td>SPI3_INT</td>
<td>SPI3</td>
<td>SPI3 interrupt</td>
</tr>
<tr>
<td>34</td>
<td>18</td>
<td>I2C0_INT</td>
<td>I2C0</td>
<td>I2C0 interrupt</td>
</tr>
<tr>
<td>35</td>
<td>19</td>
<td>I2C1_INT</td>
<td>I2C1</td>
<td>I2C1 interrupt</td>
</tr>
<tr>
<td>36</td>
<td>20</td>
<td>CAN0_INT</td>
<td>CAN0</td>
<td>CAN0 interrupt</td>
</tr>
<tr>
<td>37</td>
<td>21</td>
<td>Reserved_INT</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>38</td>
<td>22</td>
<td>Reserved_INT</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>39</td>
<td>23</td>
<td>USB_INT</td>
<td>USBD</td>
<td>USB FS Device interrupt</td>
</tr>
<tr>
<td>40</td>
<td>24</td>
<td>PS2_INT</td>
<td>PS2</td>
<td>PS2 interrupt</td>
</tr>
<tr>
<td>41</td>
<td>25</td>
<td>ACMP_INT</td>
<td>ACMP</td>
<td>Analog Comparator-0 or Comparator-1 interrupt</td>
</tr>
<tr>
<td>42</td>
<td>26</td>
<td>PDMA_INT</td>
<td>PDMA</td>
<td>PDMA interrupt</td>
</tr>
<tr>
<td>43</td>
<td>27</td>
<td>I2S_INT</td>
<td>I2S</td>
<td>I2S interrupt</td>
</tr>
<tr>
<td>44</td>
<td>28</td>
<td>PWRWU_INT</td>
<td>CLKC</td>
<td>Clock controller interrupt for chip wake up from power-down state</td>
</tr>
<tr>
<td>45</td>
<td>29</td>
<td>ADC_INT</td>
<td>ADC0/1</td>
<td>ADC interrupt</td>
</tr>
<tr>
<td>46</td>
<td>30</td>
<td>Reserved_INT</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>47</td>
<td>31</td>
<td>RTC_INT</td>
<td>RTC</td>
<td>Real time clock interrupt</td>
</tr>
</tbody>
</table>
Hardware Abstraction Layers

- The application is written using the RTOS API.
- The Real-Time Operating System (RTOS) is used.
- Software Drivers ease the use of the subsystem cores.
- Hardware Abstraction Layers are used without modification with Cortex-M0, Cortex-M1.
Hardware Abstraction Layers

HAL translates the read and write requests relevant to hardware platform. Reuse code even when the hardware platform changes.
CMSIS

- Cortex Microcontroller Software Interface Standard
- ARM& Tool vendor (Keil, IAR..)
- Structure
  - Core Peripheral Access Layer (CPAL)
  - Middleware Access Layer (MWAL) (ARM is current in development)
  - Device Peripheral Access Layer (DPAL)
    - Provide definitions for all device peripherals
CMSIS directory structure

CMSIS_V1P30

CM0

CoreSupport

DeviceSupport

ARM

ARMCM0

startup

arm

gcc

 iar

Example

arm

ARCMCM0

arm

startuup_ARCMCM0.s

main_ARCMCM0.c

startuup_ARCMCM0.s

system_ARCMCM0.c

CM3

Sourcery G++Lite

iar

Documentation

CMSIS changes.htm

CMSIS debug support.htm

License.doc

core_cm0.c

core_cm0.h

ARCMCM0.h

system_ARCMCM0.c

system_ARCMCM0.h

CMSIS_Core.htm
CMSIS coding rule

- ANSI standard types defined in the ANSI C header file `<stdint.h>` are used
- For each exception/interrupt
  - An exception/interrupt handler rule
    - exception: the postfix `_Handler`
    - interrupts: the postfix `_IRQHandler`
  - A `#` define of the interrupt number with postfix `_IRQn`
  - A default exception/interrupt handler (**weak definition**) that contains an endless loop
What CMSIS Files Do You Need For NUC100 Series?

- **NUC1xx.h**(device.h)
  - Interrupt Number Definition
  - Device Peripheral Access Layer
    - Provides definitions for all device peripherals. It contains all data structures and the address mapping for the device specific peripherals
  - To define the peripherals for the actual device. It can use several other include files to define the peripherals of the actual devices

- **startup_NUC1xx.s**(startup_device.s)
  - Cortex-M0 startup code and the complete Interrupt Vector Table

- **system_NUC1xx.c**(system_device.c)

- core_cm0.h
- core_cm0.c

**ARM offer 2 files**

**IC Vendor Offer 3 files**
typedef enum IRQn
{
    /***** Cortex-M0 Processor Exceptions Numbers ***************************************************/
    NonMaskableInt_IRQn = -14, /*!< 2 Non Maskable Interrupt */
    HardFault_IRQn = -13, /*!< 3 Cortex-M0 Hard Fault Interrupt */
    SVCall_IRQn = -5, /*!< 11 Cortex-M0 SV Call Interrupt */
    PendSV_IRQn = -2, /*!< 14 Cortex-M0 Pend SV Interrupt */
    SysTick_IRQn = -1, /*!< 15 Cortex-M0 System Tick Interrupt */

    /***** ARMIKMCU Swift specific Interrupt Numbers **********************************************/
    BOD_IRQn = 0,
    WDT_IRQn = 1,
    EINT0_IRQn = 2,
    EINT1_IRQn = 3,
    GPAB_IRQn = 4,
    GPCDE_IRQn = 5,
    PWM0_IRQn = 6,
    PWM1_IRQn = 7,
    TMR0_IRQn = 8,
    TMR1_IRQn = 9,
    TMR2_IRQn = 10,
    TMR3_IRQn = 11,
    UART0_IRQn = 12,
    UART1_IRQn = 13,
    SPI0_IRQn = 14,
    SPI1_IRQn = 15,
    SPI2_IRQn = 16,
    SPI3_IRQn = 17,
    I2C0_IRQn = 18,
    I2C1_IRQn = 19,
    CAN0_IRQn = 20,
    CAN1_IRQn = 21,
    SD_IRQn = 22,
    USBD_IRQn = 23,
    PS2_IRQn = 24,
    ACMP_IRQn = 25,
    PDMA_IRQn = 26,
    I2S_IRQn = 27,
    PWRWU_IRQn = 28,
    ADC_IRQn = 29,
    DAC_IRQn = 30,
    RTC_IRQn = 31
} IRQn_Type;

The interrupt number is defined in the “NUC1xx.h” file
These exception names are fixed and define the start of the vector table for a Cortex-M0.
; External Interrupts
DCD  BOD_IRQHandler
DCD  WDT_IRQHandler
DCD  EINT0_IRQHandler
DCD  EINT1_IRQHandler
DCD  GPAB_IRQHandler
DCD  GPCDE_IRQHandler
DCD  PWMA_IRQHandler
DCD  PWMB_IRQHandler
DCD  TMR0_IRQHandler
DCD  TMR1_IRQHandler
DCD  TMR2_IRQHandler
DCD  TMR3_IRQHandler
DCD  UART02_IRQHandler
DCD  UART1_IRQHandler
DCD  SPI0_IRQHandler
DCD  SPI1_IRQHandler
DCD  SPI2_IRQHandler
DCD  SPI3_IRQHandler
DCD  I2C0_IRQHandler
DCD  I2C1_IRQHandler
DCD  CAN0_IRQHandler
DCD  Default_Handler
DCD  Default_Handler
DCD  USBD_IRQHandler
DCD  PS2_IRQHandler
DCD  ACMP_IRQHandler
DCD  PDMA_IRQHandler
DCD  I2S_IRQHandler
DCD  PWRWU_IRQHandler
DCD  ADC_IRQHandler
DCD  Default_Handler
DCD  Default_Handler
DCD  RTC_IRQHandler

These interrupt names are fixed and define the start of the vector table for a Cortex-M0
typedef struct
{
    __IO uint32_t  XTL12M_EN:1;
    __IO uint32_t  XTL32K_EN:1;
    __IO uint32_t  OSC22M_EN:1;
    __IO uint32_t  OSC10K_EN:1;
    __IO uint32_t  WU_DLY:1;
    __IO uint32_t  WINT_EN:1;
    __IO uint32_t  INTSTS:1;
    __IO uint32_t  PWR_DOWN:1;
    __I  uint32_t  RESERVE:23;
} SYSCLK_PWRCON_T;
Group Registers for fixed CLK_BA BaseAddr (SYSCLK_T Data Type)

typedef struct
{
    SYSCLK_PWRCON_T    PWRCON;
    SYSCLK_AHBCLK_T    AHBCLK;
    SYSCLK_APBCLK_T    APBCLK;
    uint32_t            RESERVED;
    SYSCLK_CLKSEL0_T    CLKSEL0;
    SYSCLK_CLKSEL1_T    CLKSEL1;
    SYSCLK_CLKDIV_T     CLKDIV;
    uint32_t            RESERVED2;
    SYSCLK_PLLCON_T     PLLCON;
    uint32_t            RESERVED3[3];
    SYSCLK_TREG_T      TREG;
} SYSCLK_T;
To access register

```c
#define AHB_BASE 0x50000000
#define SYSCLK_BASE (AHB_BASE + 0x00200)
#define SYSCLK ((SYSCLK_T *) SYSCLK_BASE)
```

<table>
<thead>
<tr>
<th>AHB Modules Space (0x5000_0000 – 0x501F_FFFF)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5000_0000 – 0x5000_01FF</td>
<td>GCR_BA</td>
<td>System Global Control Registers</td>
</tr>
<tr>
<td>0x5000_0200 – 0x5000_02FF</td>
<td>CLK_BA</td>
<td>Clock Control Registers</td>
</tr>
<tr>
<td>0x5000_0300 – 0x5000_03FF</td>
<td>INT_BA</td>
<td>Interrupt Multiplexer Control Registers</td>
</tr>
<tr>
<td>0x5000_4000 – 0x5000_7FFF</td>
<td>GPIO_BA</td>
<td>GPIO Control Registers</td>
</tr>
<tr>
<td>0x5000_8000 – 0x5000_BFFF</td>
<td>PDMA_BA</td>
<td>SRAM_APB DMA Control Registers</td>
</tr>
<tr>
<td>0x5000_C000 – 0x5000_FFFF</td>
<td>FMC_BA</td>
<td>Flash Memory Control Registers</td>
</tr>
</tbody>
</table>

SYSCLK->PWRCON.XTL12M_EN = 1;
# Cortex-M0 Core Registers Access (core_cm0.h)

<table>
<thead>
<tr>
<th>Function Definition</th>
<th>Core Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void __enable_irq (void)</td>
<td>PRIMASK = 0</td>
<td>Global Interrupt enable (using the instruction <strong>CPSIE i</strong>)</td>
</tr>
<tr>
<td>void __disable_irq (void)</td>
<td>PRIMASK = 1</td>
<td>Global Interrupt disable (using the instruction <strong>CPSID i</strong>)</td>
</tr>
<tr>
<td>void __set_PRIMASK (uint32_t value)</td>
<td>PRIMASK = value</td>
<td>Assign value to Priority Mask Register (using the instruction <strong>MSR</strong>)</td>
</tr>
<tr>
<td>uint32_t __get_PRIMASK (void)</td>
<td>return PRIMASK</td>
<td>Return Priority Mask Register (using the instruction <strong>MRS</strong>)</td>
</tr>
<tr>
<td>void __set_CONTROL (uint32_t value)</td>
<td>CONTROL = value</td>
<td>Set CONTROL register value (using the instruction <strong>MSR</strong>)</td>
</tr>
<tr>
<td>uint32_t __get_CONTROL (void)</td>
<td>return CONTROL</td>
<td>Return Control Register Value (using the instruction <strong>MRS</strong>)</td>
</tr>
<tr>
<td>void __set_PSP (uint32_t TopOfProcStack)</td>
<td>PSP = TopOfProcStack</td>
<td>Set Process Stack Pointer value (using the instruction <strong>MSR</strong>)</td>
</tr>
<tr>
<td>uint32_t __get_PSP (void)</td>
<td>return PSP</td>
<td>Return Process Stack Pointer (using the instruction <strong>MRS</strong>)</td>
</tr>
<tr>
<td>void __set_MSP (uint32_t TopOfMainStack)</td>
<td>MSP = TopOfMainStack</td>
<td>Set Main Stack Pointer (using the instruction <strong>MSR</strong>)</td>
</tr>
<tr>
<td>uint32_t __get_MSP (void)</td>
<td>return MSP</td>
<td>Return Main Stack Pointer (using the instruction <strong>MRS</strong>)</td>
</tr>
</tbody>
</table>
# Cortex-M0 Instruction Access

<table>
<thead>
<tr>
<th>Function Name</th>
<th>CPU Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void __WFI (void)</td>
<td>WFI</td>
<td>Wait for Interrupt</td>
</tr>
<tr>
<td>void __WFE (void)</td>
<td>WFE</td>
<td>Wait for Event</td>
</tr>
<tr>
<td>void __SEV (void)</td>
<td>SEV</td>
<td>Set Event</td>
</tr>
<tr>
<td>void __ISB (void)</td>
<td>ISB</td>
<td>Instruction Synchronization Barrier</td>
</tr>
<tr>
<td>void __DSB (void)</td>
<td>DSB</td>
<td>Data Synchronization Barrier</td>
</tr>
<tr>
<td>void __DMB (void)</td>
<td>DMB</td>
<td>Data Memory Barrier</td>
</tr>
<tr>
<td>uint32_t __REV (uint32_t value)</td>
<td>REV</td>
<td>Reverse byte order in integer value.</td>
</tr>
<tr>
<td>uint32_t __REV16 (uint16_t value)</td>
<td>REV16</td>
<td>Reverse byte order in unsigned short value.</td>
</tr>
<tr>
<td>sint32_t __REVSH (sint16_t value)</td>
<td>REVSH</td>
<td>Reverse byte order in signed short value with sign extension to integer.</td>
</tr>
</tbody>
</table>
## NVIC Setup Function Call

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void NVIC_SetPriorityGrouping (uint32_t PriorityGroup)</td>
<td>Priority Grouping Value</td>
<td>Set the Priority Grouping (Groups . Subgroups)</td>
</tr>
<tr>
<td>void NVIC_EnableIRQ (IRQn_Type IRQn)</td>
<td>IRQ Number</td>
<td>Enable IRQn</td>
</tr>
<tr>
<td>void NVIC_DisableIRQ (IRQn_Type IRQn)</td>
<td>IRQ Number</td>
<td>Disable IRQn</td>
</tr>
<tr>
<td>uint32_t NVIC_GetPendingIRQ (IRQn_Type IRQn)</td>
<td>IRQ Number</td>
<td>Return 1 if IRQn is pending else 0</td>
</tr>
<tr>
<td>void NVIC_SetPendingIRQ (IRQn_Type IRQn)</td>
<td>IRQ Number</td>
<td>Set IRQn Pending</td>
</tr>
<tr>
<td>void NVIC_ClearPendingIRQ (IRQn_Type IRQn)</td>
<td>IRQ Number</td>
<td>Clear IRQn Pending Status</td>
</tr>
<tr>
<td>void NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)</td>
<td>IRQ Number, Priority</td>
<td>Set Priority for IRQn</td>
</tr>
<tr>
<td>uint32_t NVIC_GetPriority (IRQn_Type IRQn)</td>
<td>IRQ Number</td>
<td>Get Priority for IRQn</td>
</tr>
<tr>
<td>uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority, uint32_t SubPriority)</td>
<td>IRQ Number, Priority Group, Preemptive Priority, Sub Priority</td>
<td>Encode priority for given group, preemptive and sub priority</td>
</tr>
<tr>
<td>NVIC_DecodePriority (uint32_t Priority, uint32_t PreemptPriority, uint32_t* pPreemptPriority, uint32_t* pSubPriority)</td>
<td>IRQ Number, Priority, pointer to Priority Group, pointer to Preemptive Priority, pointer to Sub Priority</td>
<td>Decode given priority to group, preemptive and sub priority</td>
</tr>
<tr>
<td>void NVIC_SystemReset (void)</td>
<td>(void)</td>
<td>Resets the System</td>
</tr>
</tbody>
</table>
The Lecture is strictly used for educational purpose.

MAKES NO GUARANTEE OF VALIDITY

The lecture cannot guarantee the validity of the information found here. The lecture may recently have been changed, vandalized or altered by someone whose opinion does not correspond with the state of knowledge in the relevant fields. Note that most other encyclopedias and reference works also have similar disclaimers.

No formal peer review

The lecture is not uniformly peer reviewed; while readers may correct errors or engage in casual peer review, they have no legal duty to do so and thus all information read here is without any implied warranty of fitness for any purpose or use whatsoever. Even articles that have been vetted by informal peer review or featured article processes may later have been edited inappropriately, just before you view them.

No contract; limited license

Please make sure that you understand that the information provided here is being provided freely, and that no kind of agreement or contract is created between you and the owners or users of this site, the owners of the servers upon which it is housed, the individual Wikipedia contributors, any project administrators, sysops or anyone else who is in any way connected with this project or sister projects subject to your claims against them directly. You are being granted a limited license to copy anything from this site; it does not create or imply any contractual or extracontractual liability on the part of Wikipedia or any of its agents, members, organizers or other users.

There is no agreement or understanding between you and the content provider regarding your use or modification of this information beyond the Creative Commons Attribution-Sharealike 3.0 Unported License (CC-BY-SA) and the GNU Free Documentation License (GFDL);
General Disclaimer

Trademarks

Any of the trademarks, service marks, collective marks, design rights or similar rights that are mentioned, used or cited in the lectures are the property of their respective owners. Their use here does not imply that you may use them for any purpose other than for the same or a similar informational use as contemplated by the original authors under the CC-BY-SA and GFDL licensing schemes. Unless otherwise stated, we are neither endorsed by nor affiliated with any of the holders of any such rights and as such we cannot grant any rights to use any otherwise protected materials. Your use of any such or similar incorporeal property is at your own risk.

Personality rights

The lecture may portray an identifiable person who is alive or deceased recently. The use of images of living or recently deceased individuals is, in some jurisdictions, restricted by laws pertaining to personality rights, independent from their copyright status. Before using these types of content, please ensure that you have the right to use it under the laws which apply in the circumstances of your intended use. You are solely responsible for ensuring that you do not infringe someone else's personality rights.