การเขียนโค้ดภาษา C/C++ สำหรับไมโครคอนโทรลเลอร์ ATSAM3X8E บนบอร์ด Arduino DUE (Rev.3)

- เรียนรู้ตัวอย่างการเขียนโค้ด C/C++ (Bare-Metal Programming) สำหรับบอร์ด Arduino DUE (SAM3X8E) โดยใช้ซอฟต์แวร์ Arduino IDE
- เรียนรู้ขั้นตอนการสร้างโปรเจกต์ใน Atmel AVR Studio IDE และเขียนโค้ดเพื่อนำมาทดลองกับบอร์ด Arduino DUE
- ลองใช้คำสั่งหรือฟังก์ชันของ Atmel's Advanced Software Framework (ASF) ในเบื้องต้นสำหรับบอร์ด Arduino DUE
- เปรียบเทียบการใช้งานระหว่าง Arduino IDE และ AVR Studio

IoT Engineering Education @KMUTNB

2019-12-26

Bare-Metal Programming in C for MCUs

❖ Direct Peripheral Register Access / Use of C Pointers for Registers:

- ใช้วิธีการเข้าถึงรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของวงจรต่าง ๆ (Hardware-Mapped Registers) ภายในไมโครคอนโทรลเลอร์ (เช่น UART, SPI, ADC, ...)
- โดยทั่วไป ก็ใช้วิธีประกาศพอยนต์เตอร์ ให้ชี้ไปยังแอดเดรสของรีจิสเตอร์ใน Memory Map
- ถ้าตัวประมวลผลมีขนาด 8 บิต ขนาดของรีจิสเตอร์ก็เท่ากับ 8 บิต แต่ถ้าเป็น 32 บิต รีจิสเตอร์จะมีขนาด 32 บิต เช่นกัน

❖ Indirect Peripheral Register / Bit-Field Access:

- ใช้ struct, bit fields และ typedef เพื่อประกาศชนิดข้อมูลสำหรับรีจิสเตอร์และเข้าถึงระดับบิต
- มีการประกาศใช้ struct ที่เป็นกลุ่มของรีจิสเตอร์ซึ่งเกี่ยวข้องกับวงจรชนิดเดียวกัน

❖ Use of Instance Header Files:

- โดยปรกติแล้ว ได้มีการประกาศใช้แมโคร (Macros) ไว้ในไฟล์ C Headers เพื่อความสะดวกในการอ้างอิงและเข้าถึงรีจิสเตอร์แต่ละตัว
- ชื่อที่ใช้นั้น ก็จะสอดคล้องกับชื่อของรีจิสเตอร์ในเอกสาร Datasheet ของผู้ผลิต

Reference: https://microchipdeveloper.com/32arm:sam-bare-metal-c-programming

Bare-Metal Programming in C for MCUs

```
unsigned int *PORT0_DIR_ptr; // declare a pointer variable

// point to the PORT0_DIR register located in the memory map
PORT0_DIR_ptr = (unsigned int *)(0x41004400);

// read a value from the PORT0_DIR register
unsigned int value = *PORT0_DIR_ptr;

// write Bit 23, Bit 13 and Bit 4 as 1 in the PORT0_DIR register
*PORT0_DIR_ptr = (1 << 23) | (1 << 13) | (1 <<4);
```

```
#include <samd21.h> // include the C header file for SAMD21 (ARM Cortex-M0+)

...
// Using GPIO PA28 pin as output
REG_PORT_DIR0 |=  (1<<28);  // PA28 output direction
REG_PORT_OUT0 |=  (1<<28);  // output high to PA28 pin
REG_PORT_OUT0 &= ~(1<<28);  // output low  to PA28 pin
```

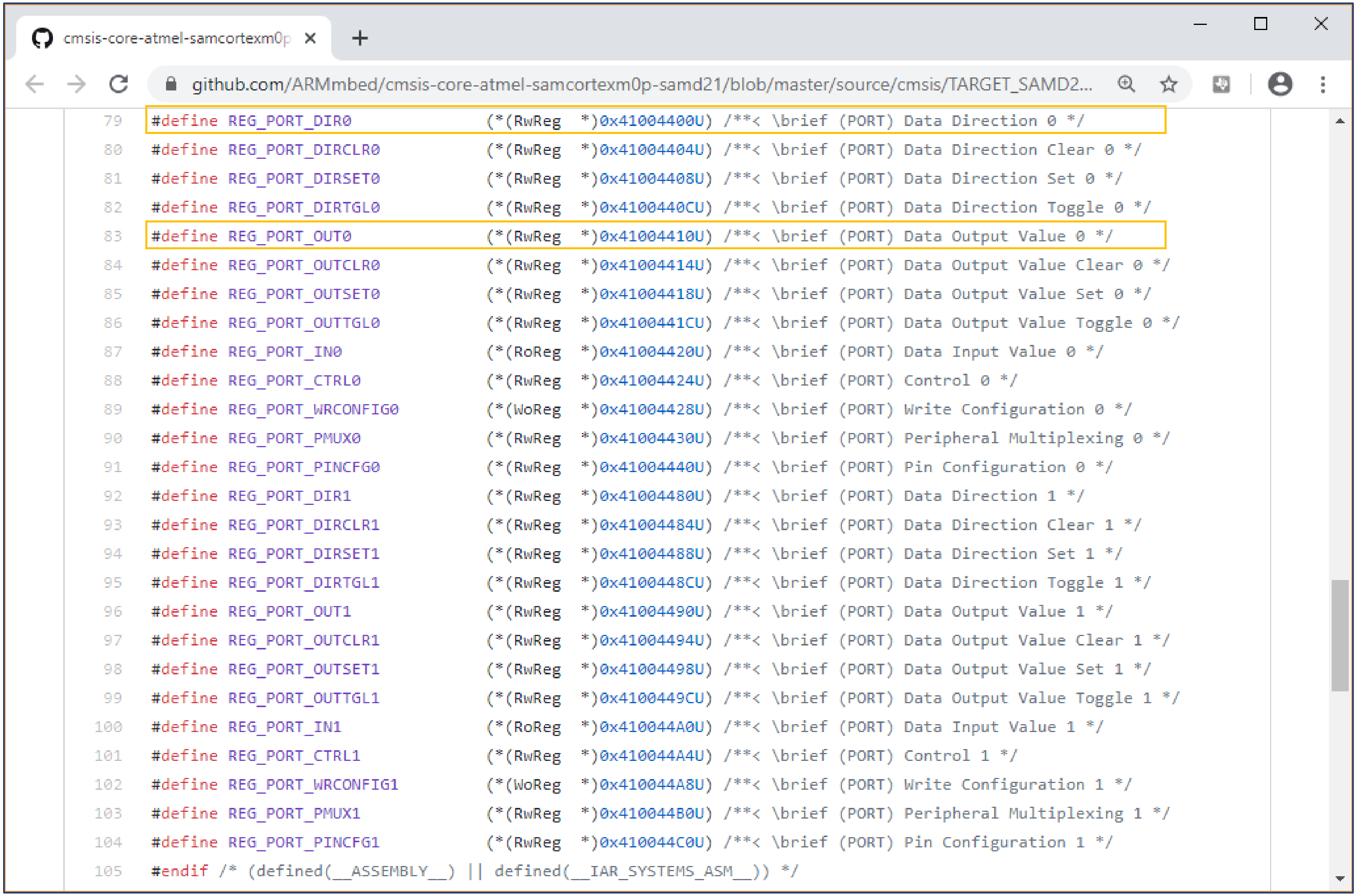
```
#define REG_PORT_DIR0    (*(RwReg *)0x41004400U) /* PORT Data Direction 0   */
```

```
#define REG_PORT_OUT0    (*(RwReg *)0x41004410U) /* PORT Data Output Value 0 */
```

```
typedef volatile  uint32_t RoReg;
typedef volatile  uint32_t WoReg;
typedef volatile  uint32_t RwReg;
```

RoReg = Read-Only Register
WoReg = Write-Only Register
RwReg = Read-Write Register

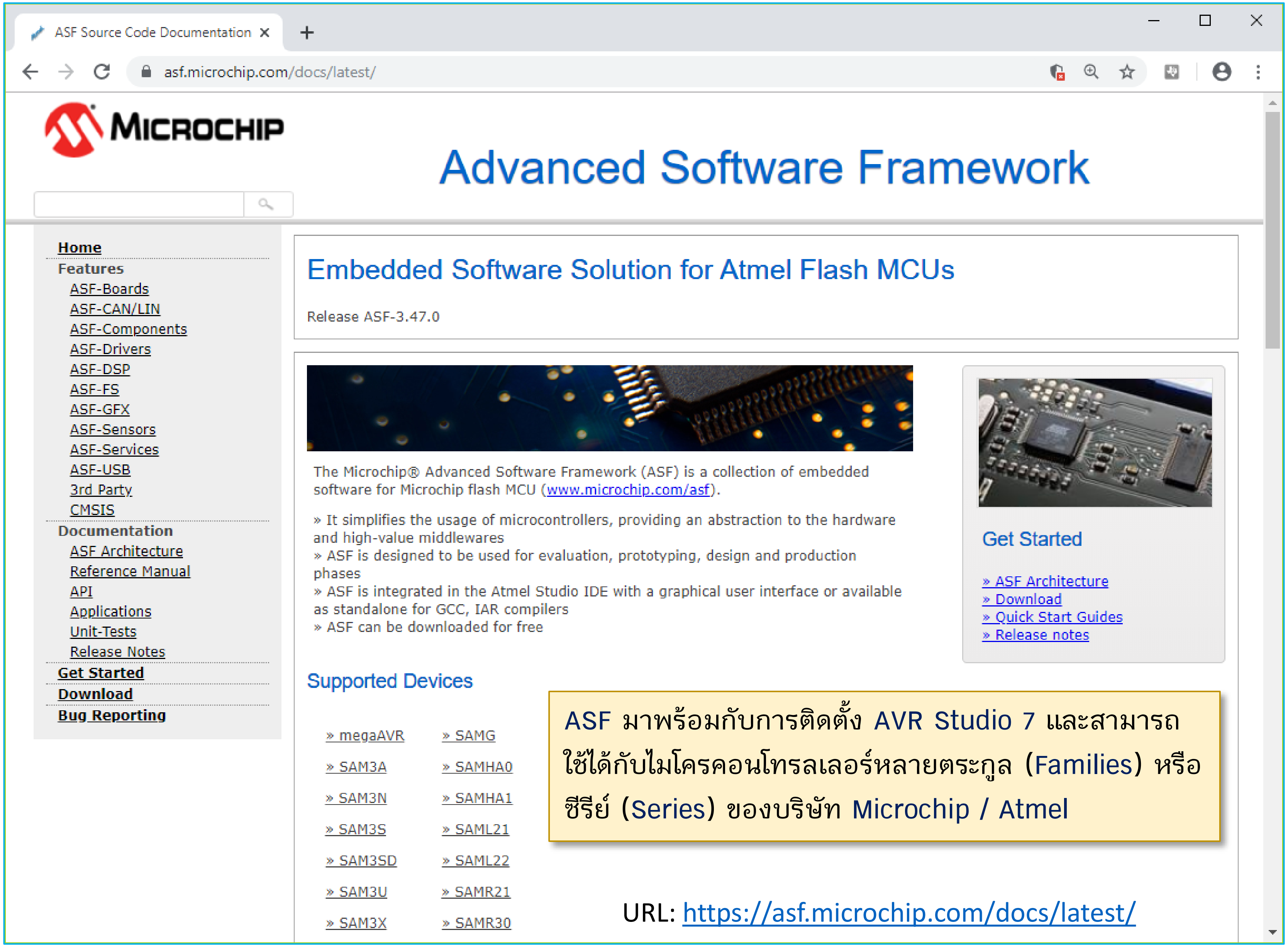
Bare-Metal Programming in C for MCUs


cmsis-core-atmel-samcortexm0p
github.com/ARMmbed/cmsis-core-atmel-samcortexm0p-samd21/blob/master/source/cmsis/TARGET_SAMD2...

```c
#define REG_PORT_DIR0              (*(RwReg  *)0x41004400U) /**< \brief (PORT) Data Direction 0 */
#define REG_PORT_DIRCLR0           (*(RwReg  *)0x41004404U) /**< \brief (PORT) Data Direction Clear 0 */
#define REG_PORT_DIRSET0           (*(RwReg  *)0x41004408U) /**< \brief (PORT) Data Direction Set 0 */
#define REG_PORT_DIRTGL0           (*(RwReg  *)0x4100440CU) /**< \brief (PORT) Data Direction Toggle 0 */
#define REG_PORT_OUT0              (*(RwReg  *)0x41004410U) /**< \brief (PORT) Data Output Value 0 */
#define REG_PORT_OUTCLR0           (*(RwReg  *)0x41004414U) /**< \brief (PORT) Data Output Value Clear 0 */
#define REG_PORT_OUTSET0           (*(RwReg  *)0x41004418U) /**< \brief (PORT) Data Output Value Set 0 */
#define REG_PORT_OUTTGL0           (*(RwReg  *)0x4100441CU) /**< \brief (PORT) Data Output Value Toggle 0 */
#define REG_PORT_IN0               (*(RoReg  *)0x41004420U) /**< \brief (PORT) Data Input Value 0 */
#define REG_PORT_CTRL0             (*(RwReg  *)0x41004424U) /**< \brief (PORT) Control 0 */
#define REG_PORT_WRCONFIG0         (*(WoReg  *)0x41004428U) /**< \brief (PORT) Write Configuration 0 */
#define REG_PORT_PMUX0             (*(RwReg  *)0x41004430U) /**< \brief (PORT) Peripheral Multiplexing 0 */
#define REG_PORT_PINCFG0           (*(RwReg  *)0x41004440U) /**< \brief (PORT) Pin Configuration 0 */
#define REG_PORT_DIR1              (*(RwReg  *)0x41004480U) /**< \brief (PORT) Data Direction 1 */
#define REG_PORT_DIRCLR1           (*(RwReg  *)0x41004484U) /**< \brief (PORT) Data Direction Clear 1 */
#define REG_PORT_DIRSET1           (*(RwReg  *)0x41004488U) /**< \brief (PORT) Data Direction Set 1 */
#define REG_PORT_DIRTGL1           (*(RwReg  *)0x4100448CU) /**< \brief (PORT) Data Direction Toggle 1 */
#define REG_PORT_OUT1              (*(RwReg  *)0x41004490U) /**< \brief (PORT) Data Output Value 1 */
#define REG_PORT_OUTCLR1           (*(RwReg  *)0x41004494U) /**< \brief (PORT) Data Output Value Clear 1 */
#define REG_PORT_OUTSET1           (*(RwReg  *)0x41004498U) /**< \brief (PORT) Data Output Value Set 1 */
#define REG_PORT_OUTTGL1           (*(RwReg  *)0x4100449CU) /**< \brief (PORT) Data Output Value Toggle 1 */
#define REG_PORT_IN1               (*(RoReg  *)0x410044A0U) /**< \brief (PORT) Data Input Value 1 */
#define REG_PORT_CTRL1             (*(RwReg  *)0x410044A4U) /**< \brief (PORT) Control 1 */
#define REG_PORT_WRCONFIG1         (*(WoReg  *)0x410044A8U) /**< \brief (PORT) Write Configuration 1 */
#define REG_PORT_PMUX1             (*(RwReg  *)0x410044B0U) /**< \brief (PORT) Peripheral Multiplexing 1 */
#define REG_PORT_PINCFG1           (*(RwReg  *)0x410044C0U) /**< \brief (PORT) Pin Configuration 1 */
#endif /* (defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_ASM__)) */
```

Ref.: https://github.com/ARMmbed/cmsis-core-atmel-samcortexm0p-samd21/blob/master/source/cmsis/TARGET_SAMD21/include/instance/ins_port.h

Atmel Advanced Software Framework (ASF)



Advanced Software Framework

Home

Features

- ASF-Boards
- ASF-CAN/LIN
- ASF-Components
- ASF-Drivers
- ASF-DSP
- ASF-FS
- ASF-GFX
- ASF-Sensors
- ASF-Services
- ASF-USB
- 3rd Party
- CMSIS

Documentation

- ASF Architecture
- Reference Manual
- API
- Applications
- Unit-Tests
- Release Notes

Get Started

Download

Bug Reporting

Embedded Software Solution for Atmel Flash MCUs

Release ASF-3.47.0

The Microchip® Advanced Software Framework (ASF) is a collection of embedded software for Microchip flash MCU (www.microchip.com/asf).

» It simplifies the usage of microcontrollers, providing an abstraction to the hardware and high-value middlewares
» ASF is designed to be used for evaluation, prototyping, design and production phases
» ASF is integrated in the Atmel Studio IDE with a graphical user interface or available as standalone for GCC, IAR compilers
» ASF can be downloaded for free

Get Started

» ASF Architecture
» Download
» Quick Start Guides
» Release notes

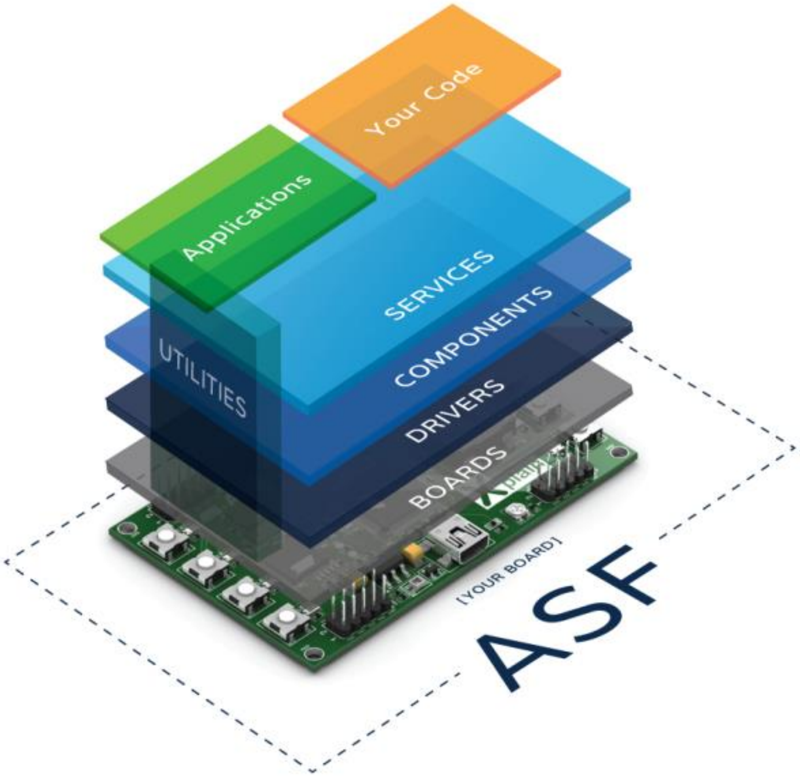
Supported Devices

» megaAVR	» SAMG
» SAM3A	» SAMHA0
» SAM3N	» SAMHA1
» SAM3S	» SAML21
» SAM3SD	» SAML22
» SAM3U	» SAMR21
» SAM3X	» SAMR30

ASF มาพร้อมกับการติดตั้ง AVR Studio 7 และสามารถใช้ได้กับไมโครคอนโทรลเลอร์หลายตระกูล (Families) หรือซีรีย์ (Series) ของบริษัท Microchip / Atmel

URL: https://asf.microchip.com/docs/latest/

Atmel Advanced Software Framework (ASF)

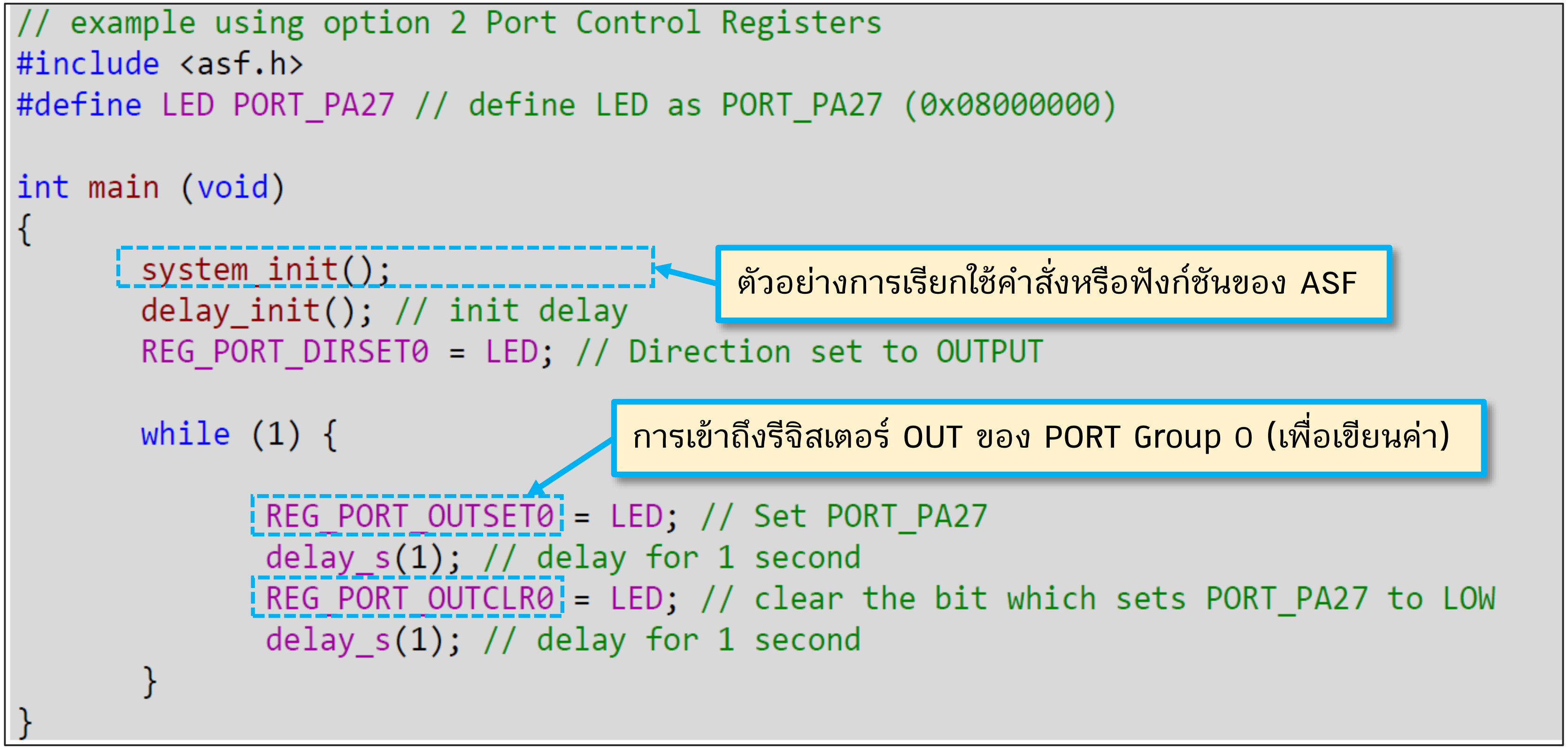


- **Drivers** that provide low level register interface functions to access a peripheral or device specific feature. The services and components will interface the drivers.
- **Services** is a module type which provides more application oriented software such as a USB classes, FAT file system, architecture optimized DSP library, graphical library, etc.
- **Components** is a module type which provides software drivers to access external hardware components such as memory, displays, sensors, wireless, etc.
- **Boards** contains mapping of all digital and analog peripheral to each I/O pin of Atmel's development kits.

Source: https://asf.microchip.com/docs/latest/architecture.html

นักพัฒนาสามารถเขียนโค้ด C/C++ สำหรับไมโครคอนโทรลเลอร์ของ Atmel / Microchip ตั้งแต่ระดับล่าง เช่น รูปแบบที่เรียกว่า Bare-Metal โดยการเข้าถึงรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของวงจรภายใน หรือ ระดับที่สูงขึ้นมา (Hardware Abstraction Layers) โดยเรียกใช้ฟังก์ชันจากไลบรารี หรือ API ที่มีการจัดทำไว้แล้ว

SAMD21 Programming with ASF



```
// example using option 2 Port Control Registers
#include <asf.h>
#define LED PORT_PA27 // define LED as PORT_PA27 (0x08000000)

int main (void)
{
       system_init();
       delay_init(); // init delay
       REG_PORT_DIRSET0 = LED; // Direction set to OUTPUT

       while (1) {

              REG_PORT_OUTSET0 = LED; // Set PORT_PA27
              delay_s(1); // delay for 1 second
              REG_PORT_OUTCLR0 = LED; // clear the bit which sets PORT_PA27 to LOW
              delay_s(1); // delay for 1 second
       }
}
```

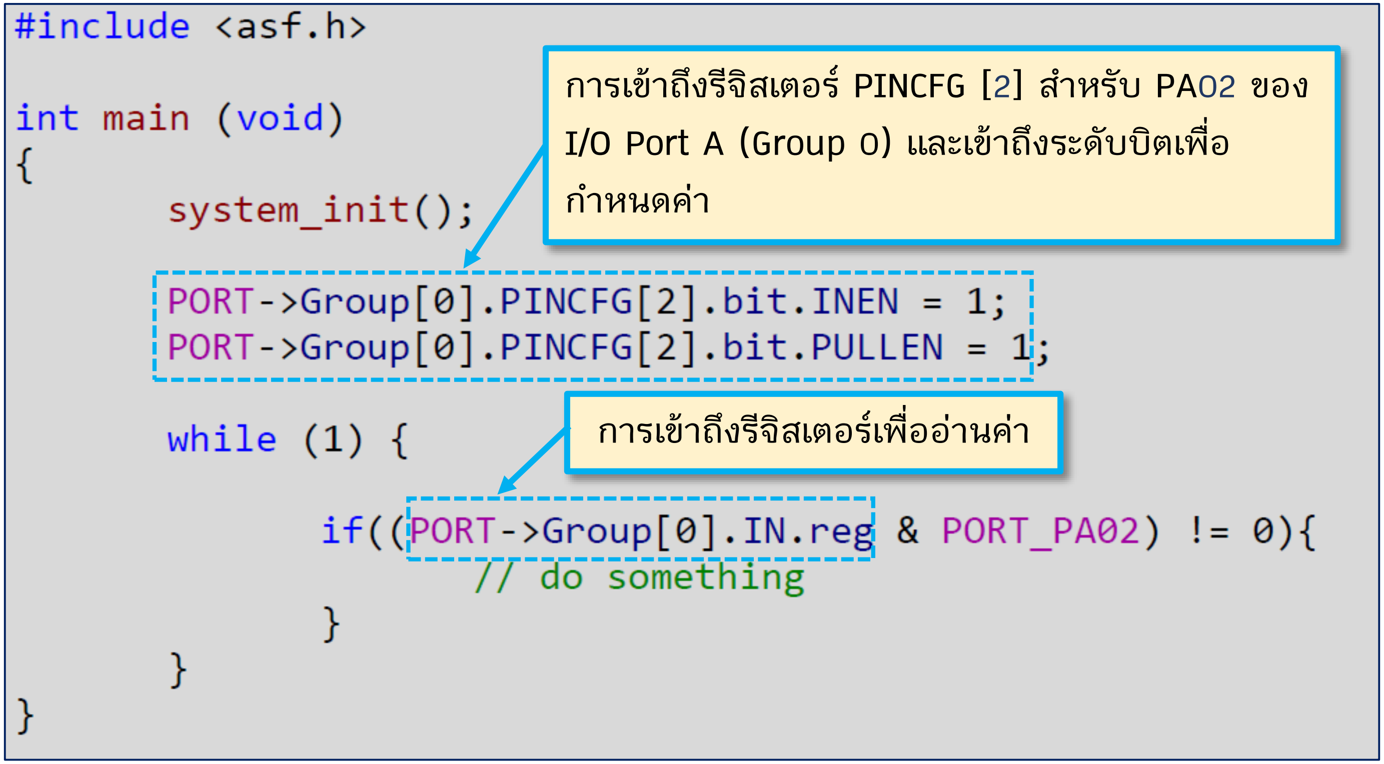
ตัวอย่างการเรียกใช้คำสั่งหรือฟังก์ชันของ ASF

การเข้าถึงรีจิสเตอร์ OUT ของ PORT Group 0 (เพื่อเขียนค่า)

Source: https://community.atmel.com/sites/default/files/forum_attachments/PIN-IO-SAMD21-SAMR21_RevA_0.pdf

ตัวอย่างโค้ดนี้สาธิตการทำให้ LED ที่ตรงกับขา PA27 เป็นเอาต์พุต (ควบคุมโดย Port I/O Controller Group: 0=A, 1=B) และมีการเขียนค่าลงในรีจิสเตอร์เพื่อกำหนดสถานะลอจิก ที่ขาเอาต์พุตดังกล่าว

SAMD21 Programming with ASF



```
#include <asf.h>

int main (void)
{
        system_init();

        PORT->Group[0].PINCFG[2].bit.INEN = 1;
        PORT->Group[0].PINCFG[2].bit.PULLEN = 1;

        while (1) {

                if((PORT->Group[0].IN.reg & PORT_PA02) != 0){
                        // do something
                }
        }
}
```

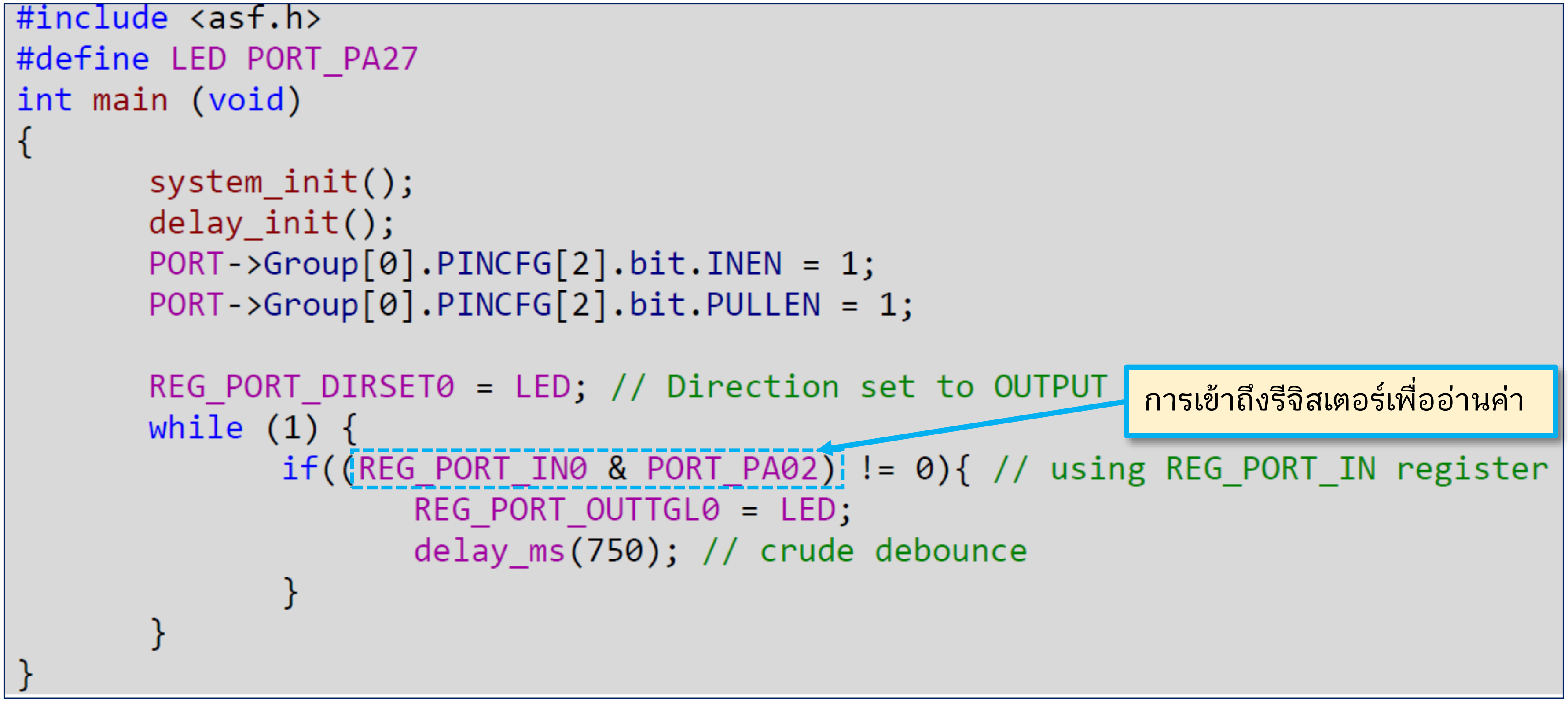
การเข้าถึงรีจิสเตอร์ PINCFG [2] สำหรับ PA02 ของ I/O Port A (Group 0) และเข้าถึงระดับบิตเพื่อกำหนดค่า

การเข้าถึงรีจิสเตอร์เพื่ออ่านค่า

Source: https://community.atmel.com/sites/default/files/forum_attachments/PIN-IO-SAMD21-SAMR21_RevA_0.pdf

ตัวอย่างโค้ดนี้สาธิตการทำให้ขา PA02 เป็นอินพุต (เช่น การต่อกับวงจรปุ่มกด) และมีการเปิดใช้งาน Internal Pull-up Resister สำหรับอินพุตที่ขาดังกล่าว จากนั้นจึงอ่านสถานะลอจิกที่ขาอินพุต

SAMD21 Programming with ASF



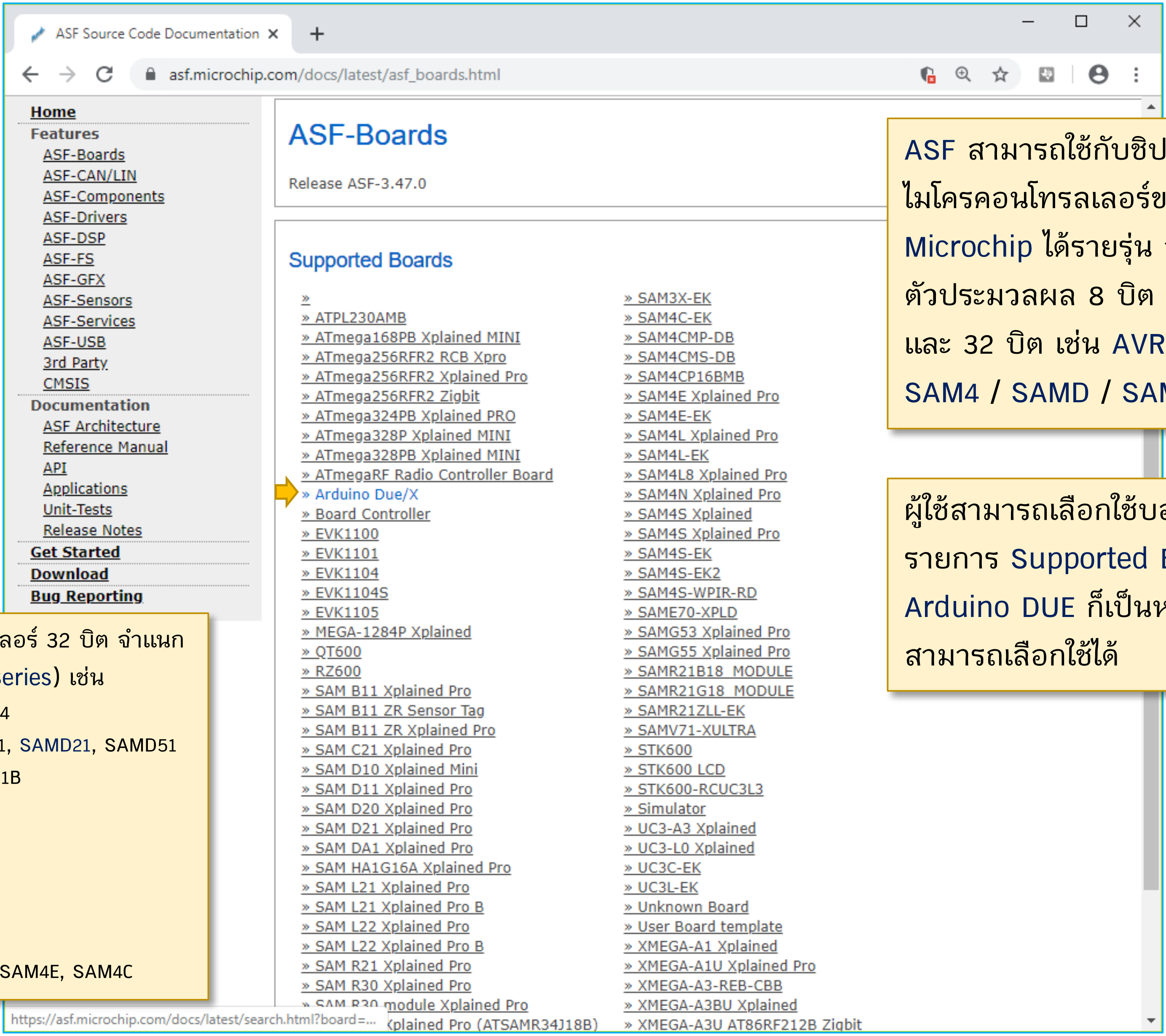
```
#include <asf.h>
#define LED PORT_PA27
int main (void)
{
        system_init();
        delay_init();
        PORT->Group[0].PINCFG[2].bit.INEN = 1;
        PORT->Group[0].PINCFG[2].bit.PULLEN = 1;

        REG_PORT_DIRSET0 = LED; // Direction set to OUTPUT
        while (1) {
                if((REG_PORT_IN0 & PORT_PA02) != 0){ // using REG_PORT_IN register
                        REG_PORT_OUTTGL0 = LED;
                        delay_ms(750); // crude debounce
                }
        }
}
```

การเข้าถึงรีจิสเตอร์เพื่ออ่านค่า

Source: https://community.atmel.com/sites/default/files/forum_attachments/PIN-IO-SAMD21-SAMR21_RevA_0.pdf

ตัวอย่างโค้ดนี้สาธิตการทำให้ขา PA02 เป็นอินพุต และเปิดใช้งาน Internal Pull-Up Resistor อ่านค่าอินพุตเพื่อนำมาใช้กำหนดสถานะเอาต์พุตที่ขา PA27 (สำหรับ LED) ให้สังเกตรูปแบบการเข้าถึงรีจิสเตอร์ในแต่ละกรณี

Atmel's MCU Boards Supported by ASF



ASF Source Code Documentation ×
asf.microchip.com/docs/latest/asf_boards.html

Home

Features

- ASF-Boards
- ASF-CAN/LIN
- ASF-Components
- ASF-Drivers
- ASF-DSP
- ASF-FS
- ASF-GFX
- ASF-Sensors
- ASF-Services
- ASF-USB
- 3rd Party
- CMSIS

Documentation

- ASF Architecture
- Reference Manual
- API
- Applications
- Unit-Tests
- Release Notes

Get Started

Download

Bug Reporting

ASF-Boards

Release ASF-3.47.0

Supported Boards

- »
- » ATPL230AMB
- » ATmega168PB Xplained MINI
- » ATmega256RFR2 RCB Xpro
- » ATmega256RFR2 Xplained Pro
- » ATmega256RFR2 Zigbit
- » ATmega324PB Xplained PRO
- » ATmega328P Xplained MINI
- » ATmega328PB Xplained MINI
- » ATmegaRF Radio Controller Board
- » Arduino Due/X
- » Board Controller
- » EVK1100
- » EVK1101
- » EVK1104
- » EVK1104S
- » EVK1105
- » MEGA-1284P Xplained
- » QT600
- » RZ600
- » SAM B11 Xplained Pro
- » SAM B11 ZR Sensor Tag
- » SAM B11 ZR Xplained Pro
- » SAM C21 Xplained Pro
- » SAM D10 Xplained Mini
- » SAM D11 Xplained Pro
- » SAM D20 Xplained Pro
- » SAM D21 Xplained Pro
- » SAM DA1 Xplained Pro
- » SAM HA1G16A Xplained Pro
- » SAM L21 Xplained Pro
- » SAM L21 Xplained Pro B
- » SAM L22 Xplained Pro
- » SAM L22 Xplained Pro B
- » SAM R21 Xplained Pro
- » SAM R30 Xplained Pro
- » SAM R30 module Xplained Pro
- Xplained Pro (ATSAMR34J18B)
- » SAM3X-EK
- » SAM4C-EK
- » SAM4CMP-DB
- » SAM4CMS-DB
- » SAM4CP16BMB
- » SAM4E Xplained Pro
- » SAM4E-EK
- » SAM4L Xplained Pro
- » SAM4L-EK
- » SAM4L8 Xplained Pro
- » SAM4N Xplained Pro
- » SAM4S Xplained
- » SAM4S Xplained Pro
- » SAM4S-EK
- » SAM4S-EK2
- » SAM4S-WPIR-RD
- » SAME70-XPLD
- » SAMG53 Xplained Pro
- » SAMG55 Xplained Pro
- » SAMR21B18_MODULE
- » SAMR21G18_MODULE
- » SAMR21ZLL-EK
- » SAMV71-XULTRA
- » STK600
- » STK600 LCD
- » STK600-RCUC3L3
- » Simulator
- » UC3-A3 Xplained
- » UC3-L0 Xplained
- » UC3C-EK
- » UC3L-EK
- » Unknown Board
- » User Board template
- » XMEGA-A1 Xplained
- » XMEGA-A1U Xplained Pro
- » XMEGA-A3-REB-CBB
- » XMEGA-A3BU Xplained
- » XMEGA-A3U AT86RF212B Zigbit

https://asf.microchip.com/docs/latest/search.html?board=...

ASF สามารถใช้กับชิปและบอร์ดไมโครคอนโทรลเลอร์ของ Atmel / Microchip ได้รายรุ่น จำแนกเป็นตัวประมวลผล 8 บิต เช่น megaAVR และ 32 บิต เช่น AVR32, SAM3 / SAM4 / SAMD / SAML เป็นต้น

ผู้ใช้สามารถเลือกใช้บอร์ด MCU จากรายการ Supported Boards และ Arduino DUE ก็เป็นหนึ่งในบอร์ดที่สามารถเลือกใช้ได้

ตัวอย่างไมโครคอนโทรลเลอร์ 32 บิต จำแนกตามซีรีย์ (เฉพาะ SAM Series) เช่น

- SAMG: SAMG55, SAMG54
- SAMD: SAMD10, SAMD11, SAMD21, SAMD51
- SAML: SAML21A, SAML21B
- SAMC: SAMC21
- SAME: SAME70
- SAMS: SAMS70
- SAM3: SAM3U, SAM3XA
- SAMR: SAMR21
- SAM4: SAM4N, SAM4S, SAM4E, SAM4C

Arduino DUE Board (REV.3)

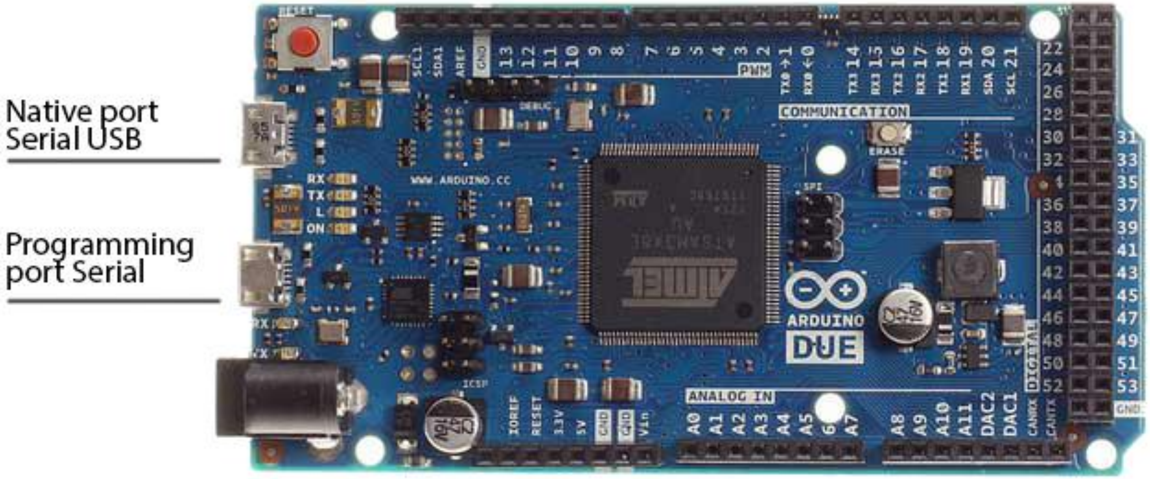


Photo source: https://www.arduino.cc/en/Guide/ArduinoDue

DUE pin mapping: https://www.arduino.cc/en/Hacking/PinMappingSAM3X

- **The Arduino Due is the first Arduino board based on a 32-bit microcontroller: Atmel SAM3X8E chip (144-lead LQFP) with ARM Cortex-M3 CPU, up to 84MHz.**
- **The Arduino Due has the same footprint as the Mega 2560.**
- **There are two USB ports available: the Programming Port (Serial) and the Native USB Port (SerialUSB).**
- **The Programming port is connected to an ATmega16U2 which acts as a USB-to-Serial converter and is used for uploading sketches and communicating with the Arduino.**
- **Note: Operating voltage: 3.3V (not 5V tolerant)**

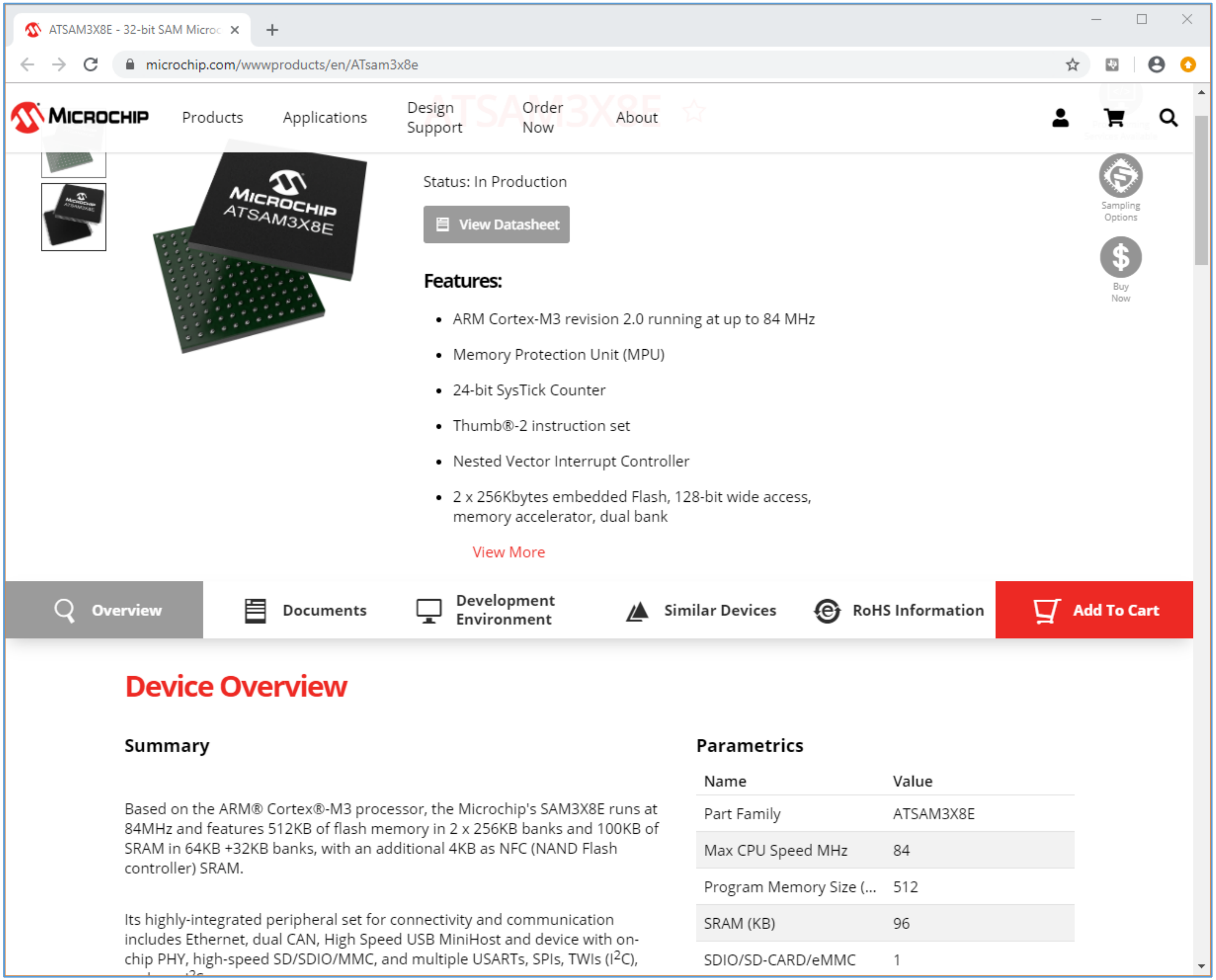
Schematic File (PDF) : https://www.arduino.cc/en/uploads/Main/arduino-Due-schematic.pdf

ATSAM3X8E Features

ทำไมบอร์ด Arduino DUE REV.3 (ATSAM3X8E MCU) จึงน่าสนใจ สำหรับนำมาใช้เป็นสื่อการเรียนรู้ด้าน Embedded System Programming / Software Development ?

- ใช้ตัวประมวลผลขนาด 32 บิต (ARM Cortex-M3), 3.3V ความเร็วสูงสุด 84 MHz
- ภายในมีหน่วยความจำ Flash สำหรับ Program Memory ขนาด 512 KB (2 x 256) และ SRAM สำหรับ Data Memory ขนาด 96 KB ซึ่งถือว่า ค่อนข้างมาก
- มีวงจรในส่วนที่เรียกว่า Memory Protection Unit (MPU) เหมาะสำหรับการทำงานที่ใช้ระบบปฏิบัติการเวลาจริง (RTOS) และสามารถรองรับการใช้งาน FreeRTOS (open source)
- มีขา I/O จำนวนมาก (มากกว่า 100) บอร์ดมีขนาดเท่ากับ Arduino Mega 2560
- มีวงจรภายใน (On-chip Peripherals) ต่าง ๆ หลายชนิดที่มักพบเห็นได้ในไมโครคอนโทรลเลอร์ประเภท High-Performance 32-bit Microcontrollers เช่น รองรับการทำงานในรูปแบบที่เรียกว่า DMA (Direct Memory Access) สำหรับ USART, USB และ Ethernet MAC เป็นต้น
- สามารถเลือกใช้บอร์ด Arduino DUE Clone จากจีน มีราคาไม่แพง (ต่ำกว่า 500 บาท)
- อัปโหลดโปรแกรมได้โดยใช้ JTAG/SWD หรือผ่าน USB / Serial (SAM-BA bootloader / BOSSA)
- สามารถเขียนโปรแกรมได้ในภาษา C/C++ โดยใช้ Arduino IDE (open source) หรือ AVR Studio (free) หรือซอฟต์แวร์อื่น ๆ (ใช้ร่วมกับ GCC-ARM Toolchain)

Online Resources for ATSAM3X8E



Status: In Production

View Datasheet

Features:

- ARM Cortex-M3 revision 2.0 running at up to 84 MHz
- Memory Protection Unit (MPU)
- 24-bit SysTick Counter
- Thumb®-2 instruction set
- Nested Vector Interrupt Controller
- 2 x 256Kbytes embedded Flash, 128-bit wide access, memory accelerator, dual bank

View More

Device Overview

Summary

Based on the ARM® Cortex®-M3 processor, the Microchip's SAM3X8E runs at 84MHz and features 512KB of flash memory in 2 x 256KB banks and 100KB of SRAM in 64KB +32KB banks, with an additional 4KB as NFC (NAND Flash controller) SRAM.

Its highly-integrated peripheral set for connectivity and communication includes Ethernet, dual CAN, High Speed USB MiniHost and device with on-chip PHY, high-speed SD/SDIO/MMC, and multiple USARTs, SPIs, TWIs (I^{2}C),

Parametrics

Name	Value
Part Family	ATSAM3X8E
Max CPU Speed MHz	84
Program Memory Size (...	512
SRAM (KB)	96
SDIO/SD-CARD/eMMC	1

URL: https://www.microchip.com/wwwproducts/en/ATsam3x8e

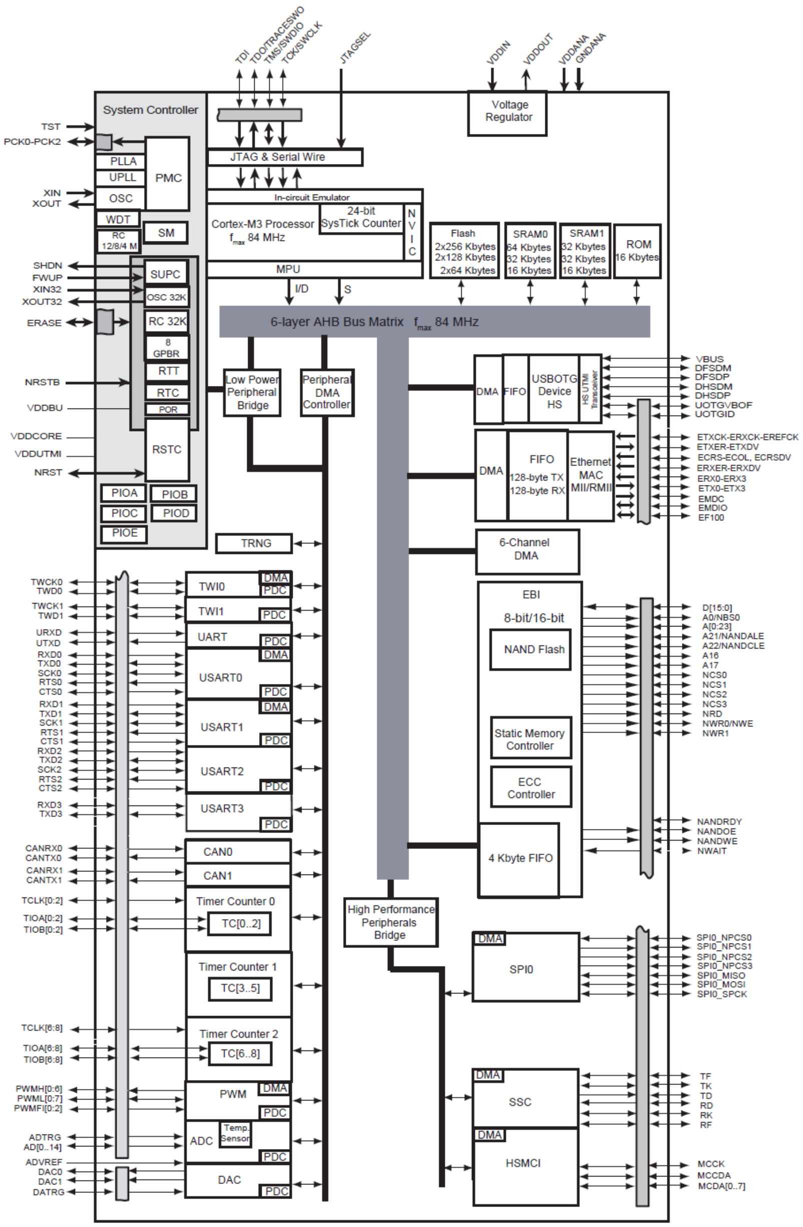
ATSAM3X8E Features

- Core
  - ARM Cortex-M3 revision 2.0 running at up to 84 MHz
  - Memory Protection Unit (MPU)
  - Thumb®-2 instruction set
  - 24-bit SysTick Counter
  - Nested Vector Interrupt Controller
- Memories
  - 256 to 512 Kbytes embedded Flash, 128-bit wide access, memory accelerator, dual bank
  - 32 to 100 Kbytes embedded SRAM with dual banks
  - 16 Kbytes ROM with embedded bootloader routines (UART, USB) and IAP routines
  - Static Memory Controller (SMC): SRAM, NOR, NAND support. NFC with 4 Kbyte RAM buffer and ECC
- System
  - Embedded voltage regulator for single supply operation
  - Power-on-Reset (POR), Brown-out Detector (BOD) and Watchdog for safe reset
  - Quartz or ceramic resonator oscillators: 3 to 20 MHz main and optional low power 32.768 kHz for RTC or device clock
  - High precision 8/12 MHz factory trimmed internal RC oscillator with 4 MHz default frequency for fast device startup
  - Slow Clock Internal RC oscillator as permanent clock for device clock in low-power mode
  - One PLL for device clock and one dedicated PLL for USB 2.0 High Speed Mini Host/Device
  - Temperature Sensor
  - Up to 17 peripheral DMA (PDC) channels and 6-channel central DMA plus dedicated DMA for High-Speed USB Mini Host/Device and Ethernet MAC

ATSAM3X8E Features

- Low-power Modes
  - Sleep, Wait and Backup modes, down to 2.5 µA in Backup mode with RTC, RTT, and GPBR
- Peripherals
  - USB 2.0 Device/Mini Host: 480 Mbps, 4 Kbyte FIFO, up to 10 bidirectional Endpoints, dedicated DMA
  - Up to 4 USARTs (ISO7816, IrDA®, Flow Control, SPI, Manchester and LIN support) and one UART
  - 2 TWI (I2C compatible), up to 6 SPIs, 1 SSC (I2S), 1 HSMCI (SDIO/SD/MMC) with up to 2 slots
  - 9-channel 32-bit Timer Counter (TC) for capture, compare and PWM mode, Quadrature Decoder Logic and 2-bit Gray Up/Down Counter for Stepper Motor
  - Up to 8-channel 16-bit PWM (PWMC) with Complementary Output, Fault Input, 12-bit Dead Time Generator Counter for Motor Control
  - 32-bit low-power Real-time Timer (RTT) and low-power Real-time Clock (RTC) with calendar and alarm features
  - 256-bit General Purpose Backup Registers (GPBR)
  - 16-channel 12-bit 1 msps ADC with differential input mode and programmable gain stage
  - 2-channel 12-bit 1 msps DAC
  - Ethernet MAC 10/100 (EMAC) with dedicated DMA
  - 2 CAN Controllers with 8 Mailboxes
  - True Random Number Generator (TRNG)
  - Register Write Protection
- I/O
  - Up to 103 I/O lines with external interrupt capability (edge or level sensitivity), debouncing, glitch filtering and on-die Series Resistor Termination
  - Up to six 32-bit Parallel Input/Outputs (PIO)

Block Diagram of ATSAM3X8E



ภายใน ATSAM3X8E นอกจาก CPU Core แล้ว มีการแบ่งออกเป็นวงจรส่วนต่าง ๆ (Peripherals) ตามฟังก์ชันการใช้งาน เชื่อมต่อเข้าด้วยกันโดยใช้ระบบบัส แบ่งได้เป็น 2 ระดับ ตามความเร็ว ได้แก่

- AHB (AMBA High-Speed Bus) สำหรับวงจรที่ทำงานและมีอัตราการรับส่งข้อมูลสูง
- APB (Advanced Peripheral Bus) สำหรับวงจรที่ทำงานหรือมีอัตราการรับส่งข้อมูลที่ช้ากว่ากลุ่มแรก

- **Power Supplies (VDDCORE, VDDIO, VDDIN, VDDOUT, VDDANA, …)**
- **Clocks, Oscillators and PLLs**
- **Shutdown, Wakeup Logic**
- **ICE and JTAG**
- **Flash Memory and NVM Configuration Bits**
- **Reset/Test**
- **Universal Asynchronous Receiver Transceiver – UART**
- **PIO Controller**
- **External Memory Bus**
- **Static Memory Controller – SMC**
- **NAND Flash Controller – NFC**
- **SDRAM Controller – SDRAMC**
- **High Speed Multimedia Card Interface – HSMCI**
- **Universal Synchronous Asynchronous Receiver Transmitter – USARTx**
- **Ethernet MAC 10/100 – EMAC**
- **CAN Controller – CANx**
- **Synchronous Serial Controller – SSC**
- **Timer/Counter – TC**
- **Pulse Width Modulation Controller – PWMC**
- **Serial Peripheral Interface – SPIx**
- **Two-Wire Interface – TWIx**
- **Digital-to-Analog Converter Controller – DACC**
- **Fast Flash Programming Interface – FFPI**
- **USB High Speed Device**

Atmel SAM3X8E: On-Chip Peripherals

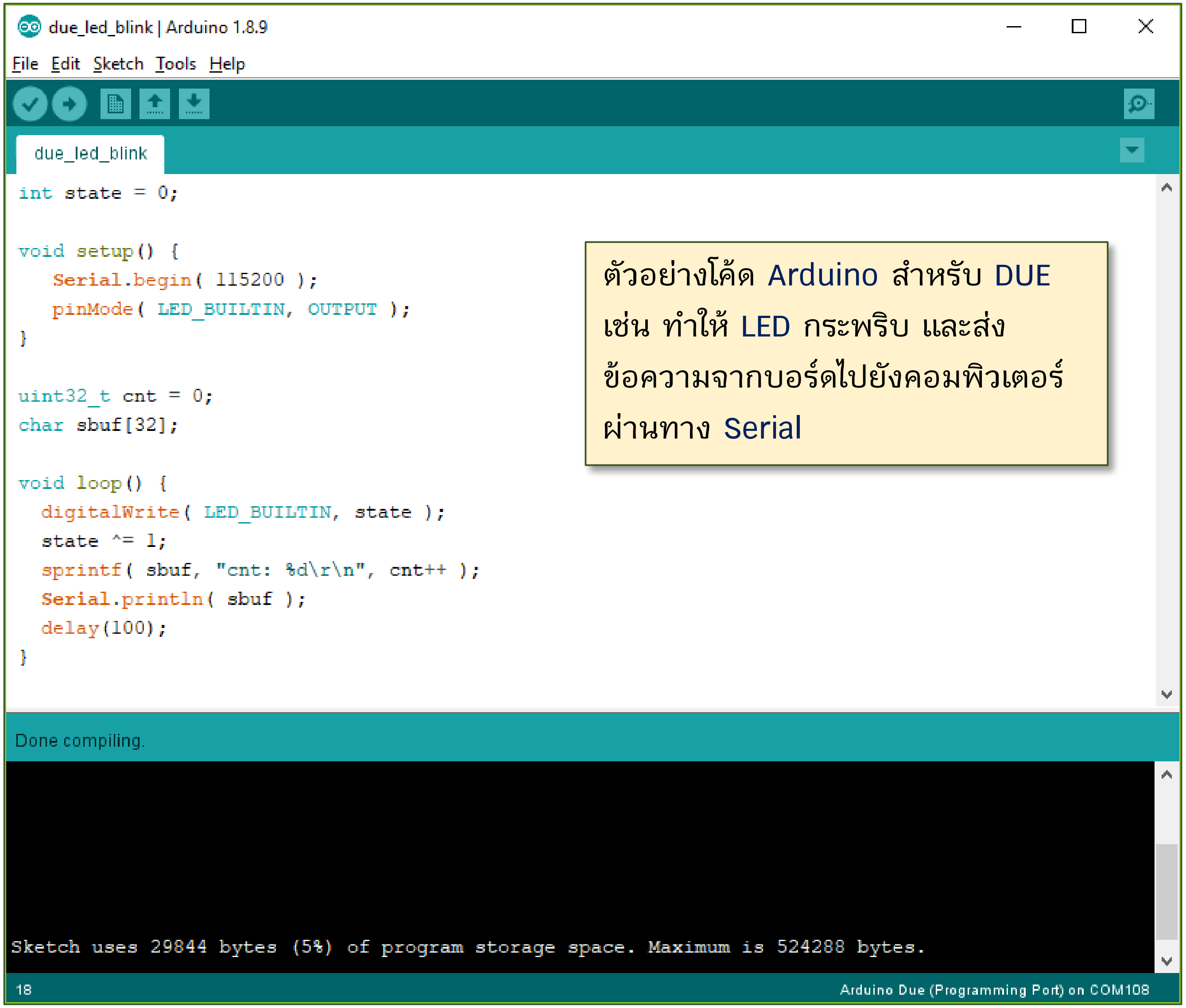
Instance Name

SUPC	Supply Controller
RSTC	Reset Controller
RTC	Real-time Clock
RTT	Real-time Timer
WDG	Watchdog Timer
PMC	Power Management Controller
EEFC0	Enhanced Embedded Flash Controller 0
EEFC1	Enhanced Embedded Flash Controller 1
UART	Universal Asynchronous Receiver Transceiver
SMC	Static Memory Controller
SDRAMC	Synchronous Dynamic RAM Controller
PIOA	Parallel I/O Controller A
PIOB	Parallel I/O Controller B
PIOC	Parallel I/O Controller C
PIOD	Parallel I/O Controller D
PIOE	Parallel I/O Controller E
PIOF	Parallel I/O Controller F
USART0	Universal Synchronous Async. Receiver Transmitter 0
USART1	Universal Synchronous Async. Receiver Transmitter 1
USART2	Universal Synchronous Async. Receiver Transmitter 2
USART3	Universal Synchronous Async. Receiver Transmitter 3
HSMCI	High Speed Multimedia Card Interface

TWI0	Two-Wire Interface 0
TWI1	Two-Wire Interface 1
SPI0	Serial Peripheral Interface 0
SPI1	Serial Peripheral Interface 1
SSC	Synchronous Serial Controller
TC0	Timer Counter Channel 0
TC1	Timer Counter Channel 1
TC2	Timer Counter Channel 2
TC3	Timer Counter Channel 3
TC4	Timer Counter Channel 4
TC5	Timer Counter Channel 5
TC6	Timer Counter Channel 6
TC7	Timer Counter Channel 7
TC8	Timer Counter Channel 8
PWM	Pulse Width Modulation Controller
ADC	ADC Controller
DACC	DAC Controller
DMAC	DMA Controller
UOTGHS	USB OTG High Speed
TRNG	True Random Number Generator

Peripheral Component (or ID): PIO, Peripheral Instance Names: PIOA, PIOB, …

Programming Arduino DUE with Arduino IDE



due_led_blink | Arduino 1.8.9
File Edit Sketch Tools Help
due_led_blink

```
int state = 0;

void setup() {
   Serial.begin( 115200 );
   pinMode( LED_BUILTIN, OUTPUT );
}

uint32_t cnt = 0;
char sbuf[32];

void loop() {
  digitalWrite( LED_BUILTIN, state );
  state ^= 1;
  sprintf( sbuf, "cnt: %d\r\n", cnt++ );
  Serial.println( sbuf );
  delay(100);
}
```

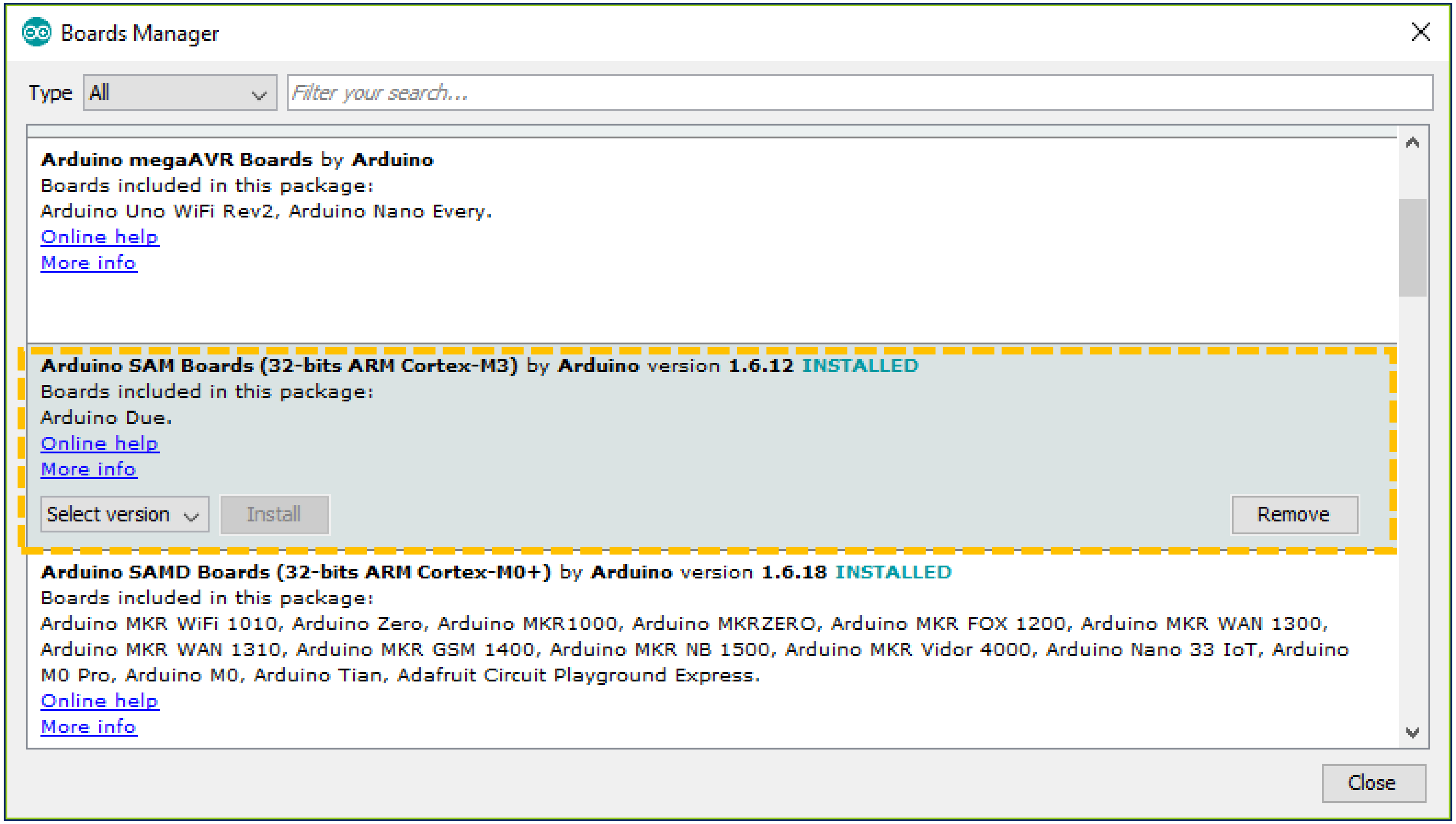
Done compiling.

```
Sketch uses 29844 bytes (5%) of program storage space. Maximum is 524288 bytes.
```

18
Arduino Due (Programming Port) on COM108

ตัวอย่างโค้ด Arduino สำหรับ DUE เช่น ทำให้ LED กระพริบ และส่งข้อความจากบอร์ดไปยังคอมพิวเตอร์ผ่านทาง Serial

Programming Arduino DUE with Arduino IDE



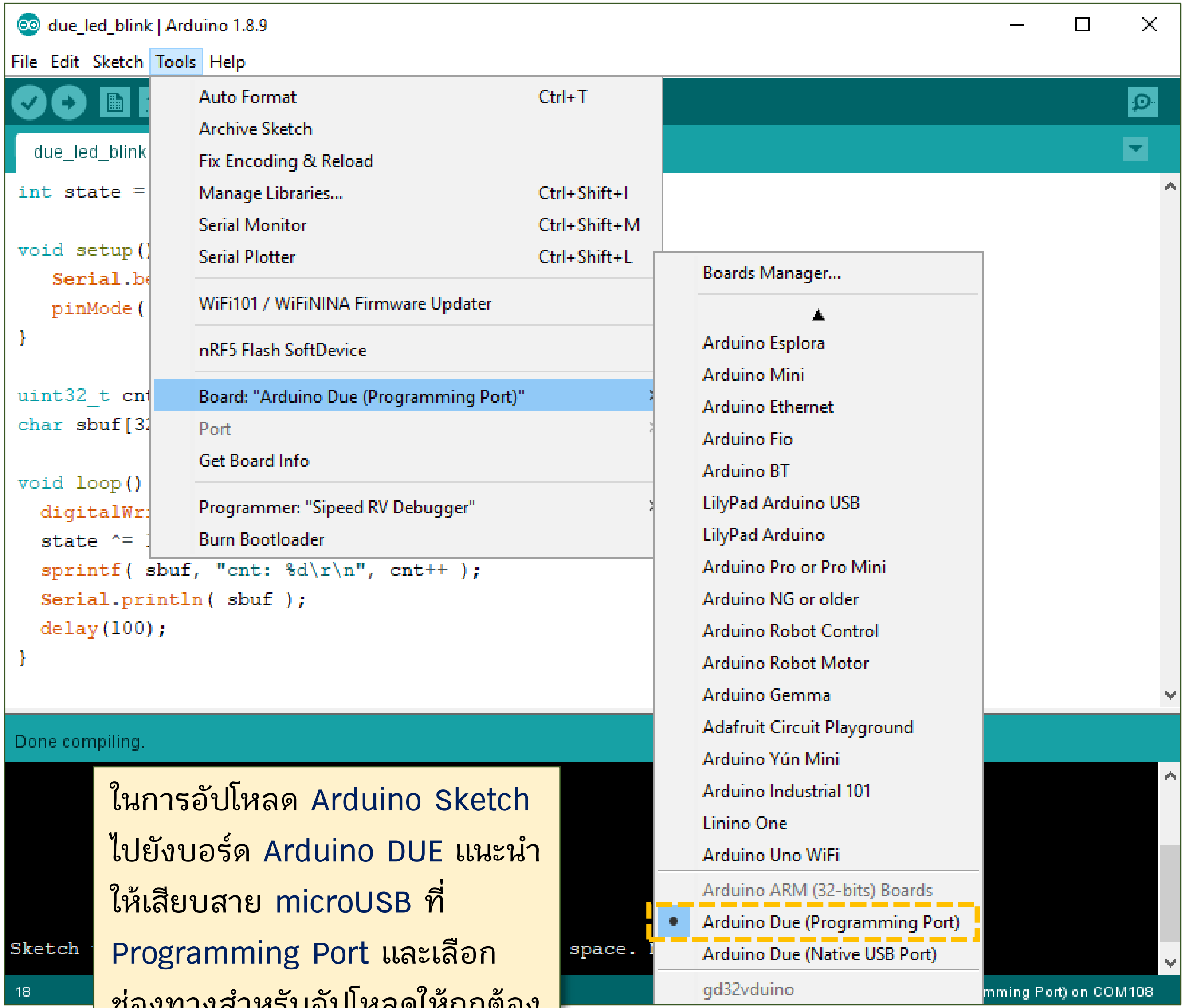
Arduino megaAVR Boards by **Arduino**
Boards included in this package:
Arduino Uno WiFi Rev2, Arduino Nano Every.
Online help
More info

Arduino SAM Boards (32-bits ARM Cortex-M3) by **Arduino** version **1.6.12** **INSTALLED**
Boards included in this package:
Arduino Due.
Online help
More info

Arduino SAMD Boards (32-bits ARM Cortex-M0+) by **Arduino** version **1.6.18** **INSTALLED**
Boards included in this package:
Arduino MKR WiFi 1010, Arduino Zero, Arduino MKR1000, Arduino MKRZERO, Arduino MKR FOX 1200, Arduino MKR WAN 1300, Arduino MKR WAN 1310, Arduino MKR GSM 1400, Arduino MKR NB 1500, Arduino MKR Vidor 4000, Arduino Nano 33 IoT, Arduino M0 Pro, Arduino M0, Arduino Tian, Adafruit Circuit Playground Express.
Online help
More info

ถ้าจะเขียนโค้ด Arduino Sketch สำหรับบอร์ด Arduino DUE โดยใช้ Arduino IDE
จะต้องติดตั้ง Arduino Core for SAM (ARM Cortex-M3) สำหรับ Boards Manager

Programming Arduino DUE with Arduino IDE



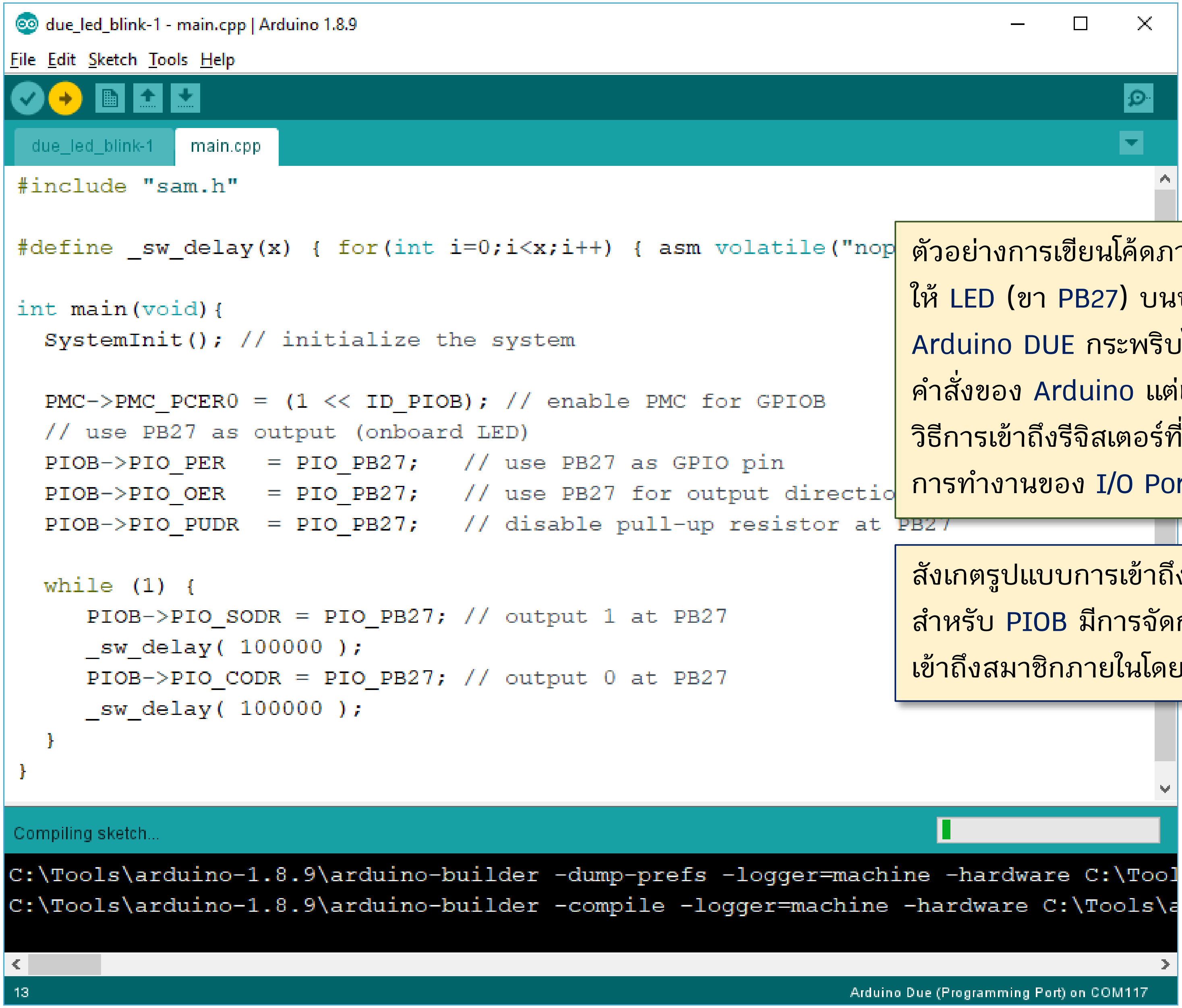
```
int state =

void setup()
   Serial.be
   pinMode(
}

uint32_t cnt
char sbuf[32

void loop()
  digitalWr
  state ^= 1
  sprintf( sbuf, "cnt: %d\r\n", cnt++ );
  Serial.println( sbuf );
  delay(100);
}
```

Done compiling.

```
Sketch                                         space. 
```

ในการอัปโหลด Arduino Sketch ไปยังบอร์ด Arduino DUE แนะนำให้เสียบสาย microUSB ที่ Programming Port และเลือกช่องทางสำหรับอัปโหลดให้ถูกต้อง

Programming Arduino DUE with Arduino IDE


due_led_blink-1 - main.cpp | Arduino 1.8.9
File Edit Sketch Tools Help
due_led_blink-1
main.cpp
Compiling sketch...
13
Arduino Due (Programming Port) on COM117

```
#include "sam.h"

#define _sw_delay(x) { for(int i=0;i<x;i++) { asm volatile("nop

int main(void){
  SystemInit(); // initialize the system

  PMC->PMC_PCER0 = (1 << ID_PIOB); // enable PMC for GPIOB
  // use PB27 as output (onboard LED)
  PIOB->PIO_PER   = PIO_PB27;   // use PB27 as GPIO pin
  PIOB->PIO_OER   = PIO_PB27;   // use PB27 for output directio
  PIOB->PIO_PUDR  = PIO_PB27;   // disable pull-up resistor at PB27

  while (1) {
     PIOB->PIO_SODR = PIO_PB27; // output 1 at PB27
     _sw_delay( 100000 );
     PIOB->PIO_CODR = PIO_PB27; // output 0 at PB27
     _sw_delay( 100000 );
  }
}
```

```
C:\Tools\arduino-1.8.9\arduino-builder -dump-prefs -logger=machine -hardware C:\Tool
C:\Tools\arduino-1.8.9\arduino-builder -compile -logger=machine -hardware C:\Tools\a
```

ตัวอย่างการเขียนโค้ดภาษา C เพื่อทำให้ LED (ขา PB27) บนบอร์ด Arduino DUE กระพริบได้ โดย<u>ไม่ใช้</u>คำสั่งของ Arduino แต่เปลี่ยนมาใช้วิธีการเข้าถึงรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของ I/O Port B

สังเกตรูปแบบการเข้าถึงรีจิสเตอร์สำหรับ PIOB มีการจัดกลุ่มและเข้าถึงสมาชิกภายในโดยใช้พอยน์เตอร์

Parallel I/O Controllers: PIOA, PIOB, ...

ในการเรียนรู้หลักการทำงานของวงจรต่าง ๆ ภายใน MCU สามารถศึกษาได้จากเอกสาร Datasheet ของผู้ผลิต
https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf

ตัวอย่างข้อความ (ภาษาอังกฤษ) จากเอกสาร Datasheet สำหรับ ATSAM3X8E ในส่วนที่เกี่ยวข้องกับ PIO Controller

- Each of the Parallel I/O Controllers (PIO) manages up to 32 fully programmable I/O lines per I/O Port.
  - Each I/O line is associated with a bit number in all of the 32-bit registers (fully programmable through set/clear registers)
  - Either used as a general-purpose I/O or be assigned to a function of an embedded peripheral: Multiplexing of four peripheral functions per I/O Line
- For each I/O Line (whether used as a peripheral pin or general-purpose I/O pin)
  - Input change, rising edge, falling edge, low level and level interrupt
  - Debouncing or Input-Glitch filtering option
  - Multi-drive option enables driving in open-drain output mode
  - Programmable pull-up on each I/O line
  - Pin data status register, supplies visibility of the level on the pin at any time
- Synchronous output, provides Set / Clear of several I/O lines in a single write
- Each PIO controller is controlled by the Power Management Controller (PMC).
  - The configuration of the I/O lines (e.g. for output) does not require the PIO Controller clock to be enabled.
  - However, when the clock is disabled, not all of the features of the PIO Controller are available, including input-glitch filtering for input and PIO interrupts.

Parallel I/O Controllers: PIOA, PIOB, ...

- ไมโครคอนโทรลเลอร์ SAM3X8E เป็นตัวประมวลผล 32 บิต และรีจิสเตอร์มีขนาด 32 บิต
- การใช้งานขา I/O ต่าง ๆ จะถูกควบคุมโดยส่วนที่เรียกว่า Parallel I/O (PIO) Controller ซึ่งแบ่งกลุ่มตามพอร์ต เช่น PIOA, PIOB, ..., PIOF แต่ละพอร์ตมีจำนวนขาที่เกี่ยวข้องสูงสุด 32 ขาสัญญาณ
- โดยทั่วไปแล้ว ขา I/O ของแต่ละพอร์ต จะทำหน้าที่เป็น General-Purpose I/O (GPIO) และถูกควบคุมการทำงานโดย PIO Controller หรืออาจถูกเลือกใช้เป็นขาสำหรับวงจรภายในได้ เรียกว่า Peripheral Pins และมี 2 โหมดให้เลือกคือ Peripheral A และ Peripheral B
  - ขาที่จะใช้สำหรับวงจรภายใน เช่น USART, PWM, TWI, SPI หรือ CAN เป็นต้น
  - นอกจากนั้นยังมีกลุ่ม Extra Function สำหรับ ADC และ DAC หรือเป็น Wakeup Pins
- ขา I/O สามารถเป็นแหล่งกำเนิดหรือสร้างสัญญาณอินเทอร์รัพท์ได้ (Interrupt Sources) โดยการตรวจสอบเหตุการณ์การเปลี่ยนแปลงระดับสัญญาณที่ขา I/O (มีหลายโหมดให้เลือก) เช่น ขอบสัญญาณขาขึ้น (Rising Edge) หรือ ขาลง (Falling Edge) หรือ ใช้ระดับสัญญาณ (Low or High Level) เป็นต้น

Parallel I/O Controllers: PIOA, PIOB, …

- การใช้งานขา I/O แต่ละขา สามารถเปิด-ปิดการใช้งานวงจร Pull-Up Resistor ได้ หลังจากการรีเซต Pull-Up Register จะถูกเปิดให้ใช้งานเป็นสถานะเริ่มต้น
- การทำงานของวงจร PIO Controller จะถูกควบคุมด้วยวงจรอีกส่วนหนึ่งที่เรียกว่า (Power Management Controller: PMC) สามารถเปิด-ปิดการทำงานของ Clock ให้กับ I/O Port ได้ เช่น PIOA, PIOB, … (การปิดการทำงานในส่วนนี้ ก็ช่วยในการประหยัดการใช้พลังงานของไมโครคอนโทรลเลอร์)
- รีจิสเตอร์ที่เกี่ยวข้องกับ PIO Controller สามารถเปิดปิดโหมดการป้องกันการเขียนได้ (Write Protect Mode)
- ข้อสังเกต: วงจร PIO Controller สำหรับแต่ละ I/O Port (A,B,C,…) มีจำนวนรีจิสเตอร์ที่เกี่ยวข้องค่อนข้างมาก (เมื่อเปรียบเทียบกับกรณีของ megaAVR)

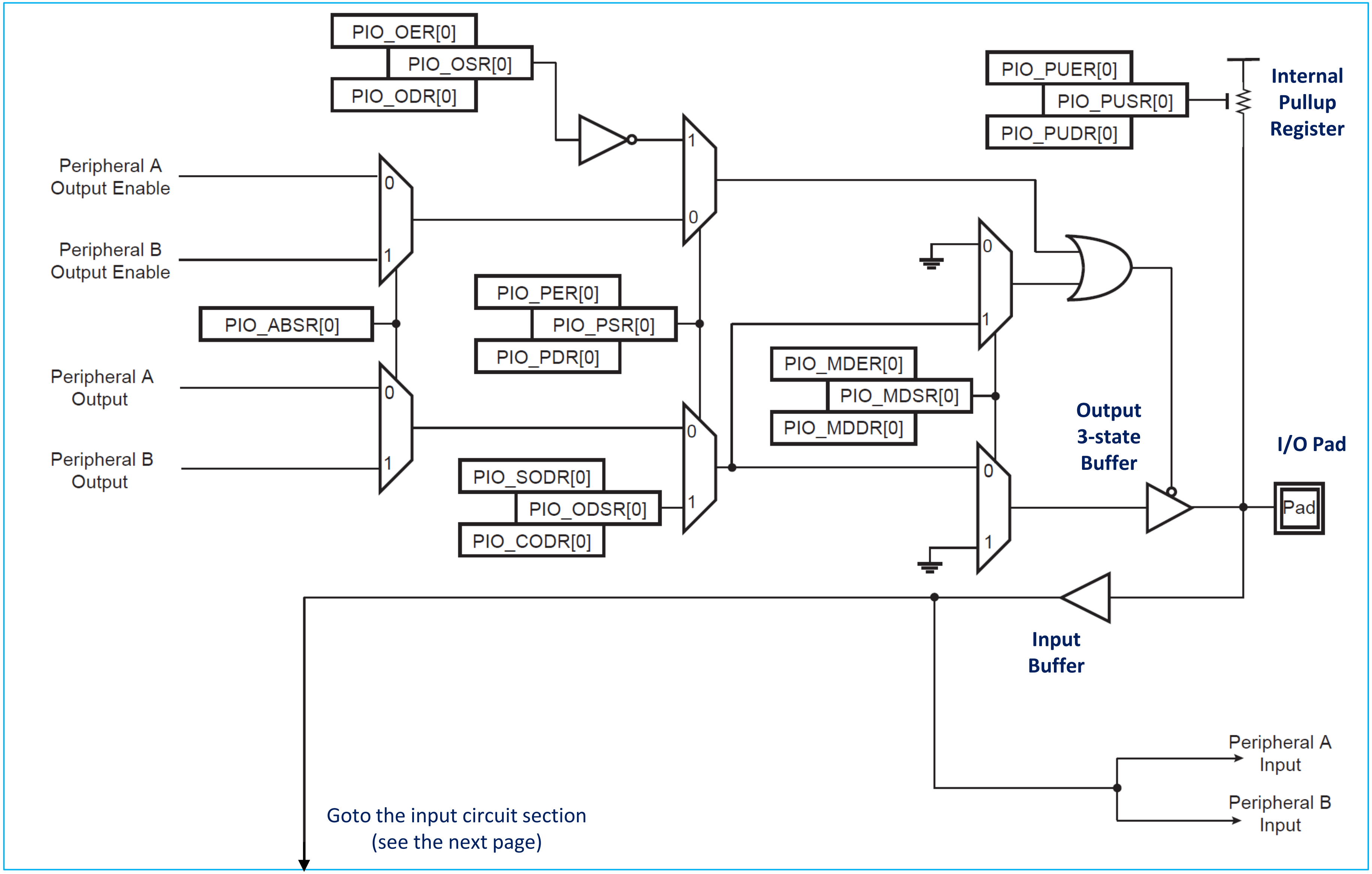
PIO_PER	PIO Enable Register
PIO_PDR	PIO Disable Register
PIO_OER	PIO Output Enable Register
PIO_ODR	PIO Output Disable Register
PIO_IFER	PIO Controller Input Filter Enable Register
PIO_IFDR	PIO Controller Input Filter Disable Register
PIO_SODR	PIO Set Output Data Register
PIO_CODR	PIO Clear Output Data Register
PIO_IER	PIO Interrupt Enable Register
PIO_IDR	PIO Interrupt Disable Register

PIO_MDER	PIO Multi-driver Enable Register
PIO_MDDR	PIO Multi-driver Disable Register
PIO_PUDR	PIO Pull-up Disable Resistor
PIO_PUER	PIO Pull-up Enable Register
PIO_ABSR	PIO Peripheral AB Select Register
PIO_OWER	PIO Output Write Enable Register
PIO_OWDR	PIO Output Write Disable Register
PIO_WPMR	PIO Write Protect Mode Register
PIO_WPSR	PIO Write Protect Status Register
….	….

Parallel I/O Controllers: PIOA, PIOB, ...

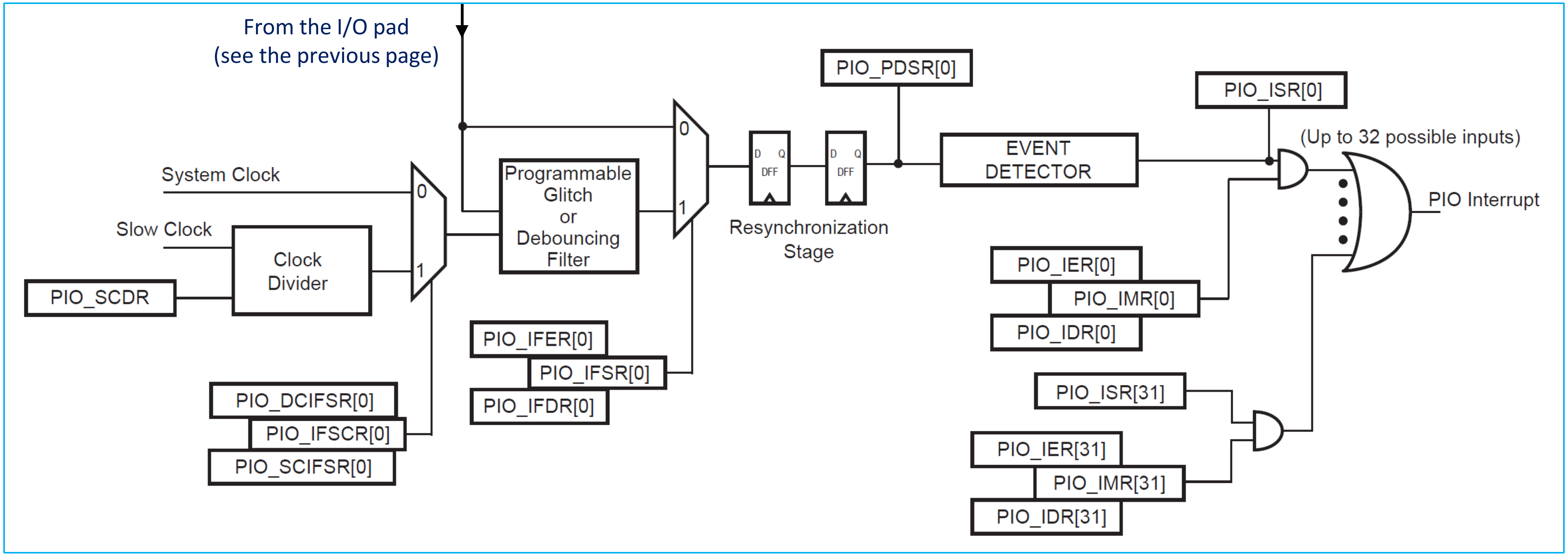
- รีจิสเตอร์ที่เกี่ยวข้องกับ PIO Controller สามารถแบ่งออกเป็นกลุ่มย่อย เช่น การกำหนดทิศทางของขา I/O แต่ละขาให้เป็นเอาต์พุตหรืออินพุต
  - PIO_OER = PIO Output Enable Register (Write-Only)
  - PIO_ODR = PIO Output Disable Register (Write-Only)
  - PIO_OSR = PIO Output Status Register (Read-Only)
- ในกรณีนี้ จะเห็นได้ว่า มีรีจิสเตอร์สำหรับ Set Bit หรือ Clear Bit แยกกัน (สำหรับการเขียนค่าไปยังรีจิสเตอร์เท่านั้น) และมีรีจิสเตอร์ไว้สำหรับระบุสถานะ (สำหรับการอ่านจากรีจิสเตอร์เท่านั้น)
- รีจิสเตอร์ที่เกี่ยวข้องกับ PIO โดยทั่วไปจะเป็นแบบเขียนหรืออ่านได้ แบบใดแบบหนึ่ง (Write-Only / Read-Only) แต่อาจมีบางตัวที่เขียนและอ่านได้ (Read-Write)

I/O Line: Block Diagram

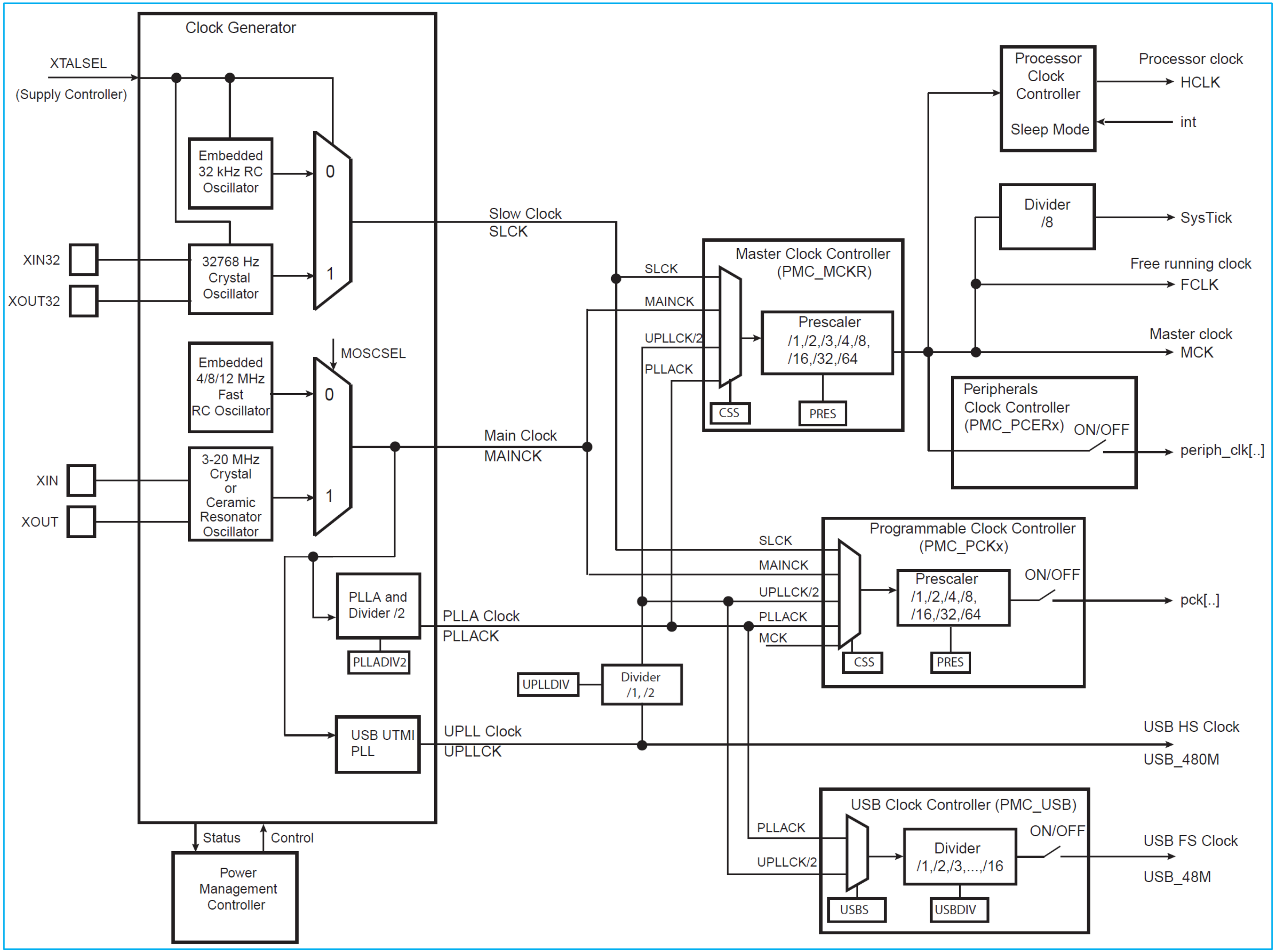


Goto the input circuit section
(see the next page)

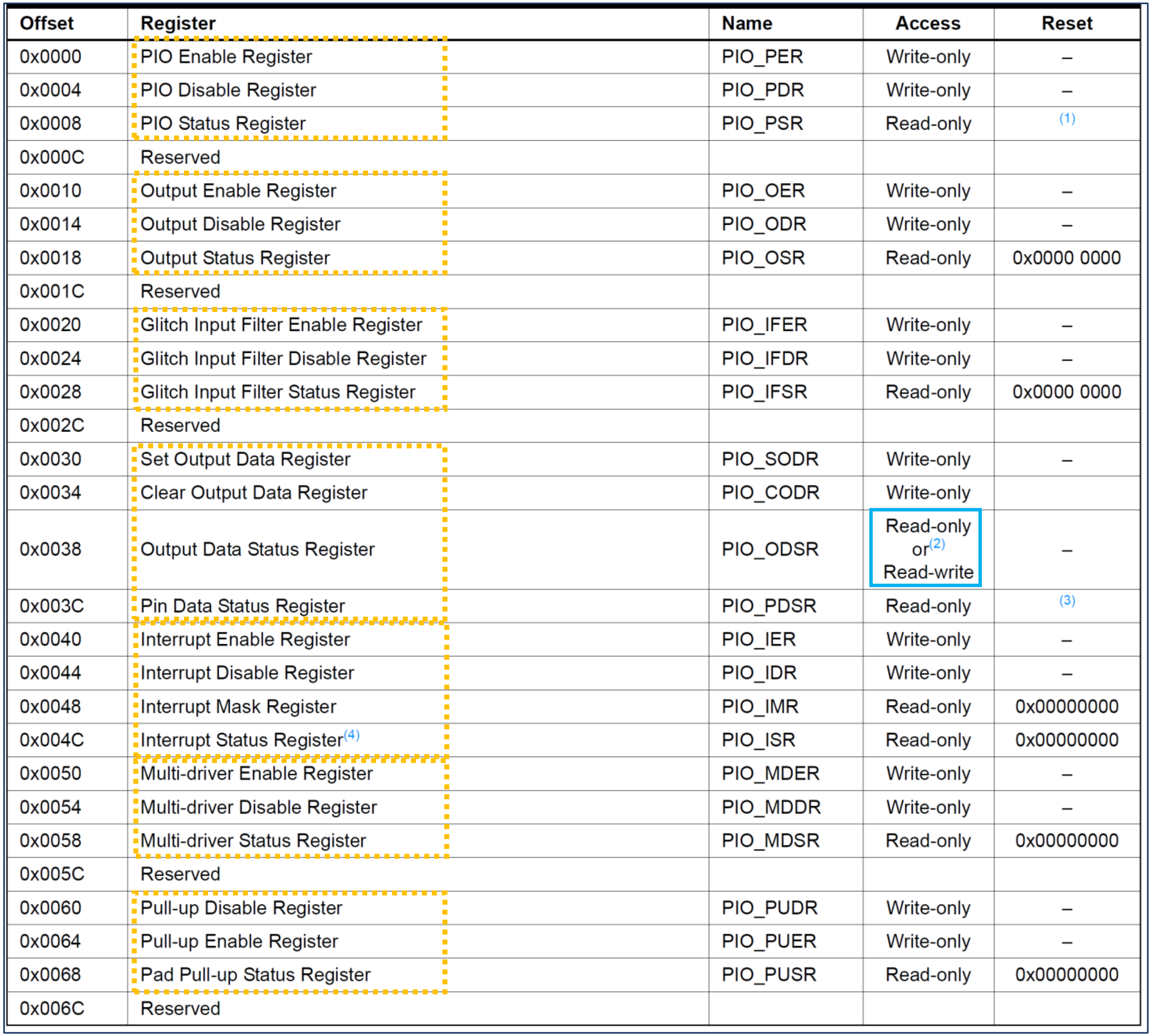
I/O Line: Block Diagram



General Clock: Block Diagram

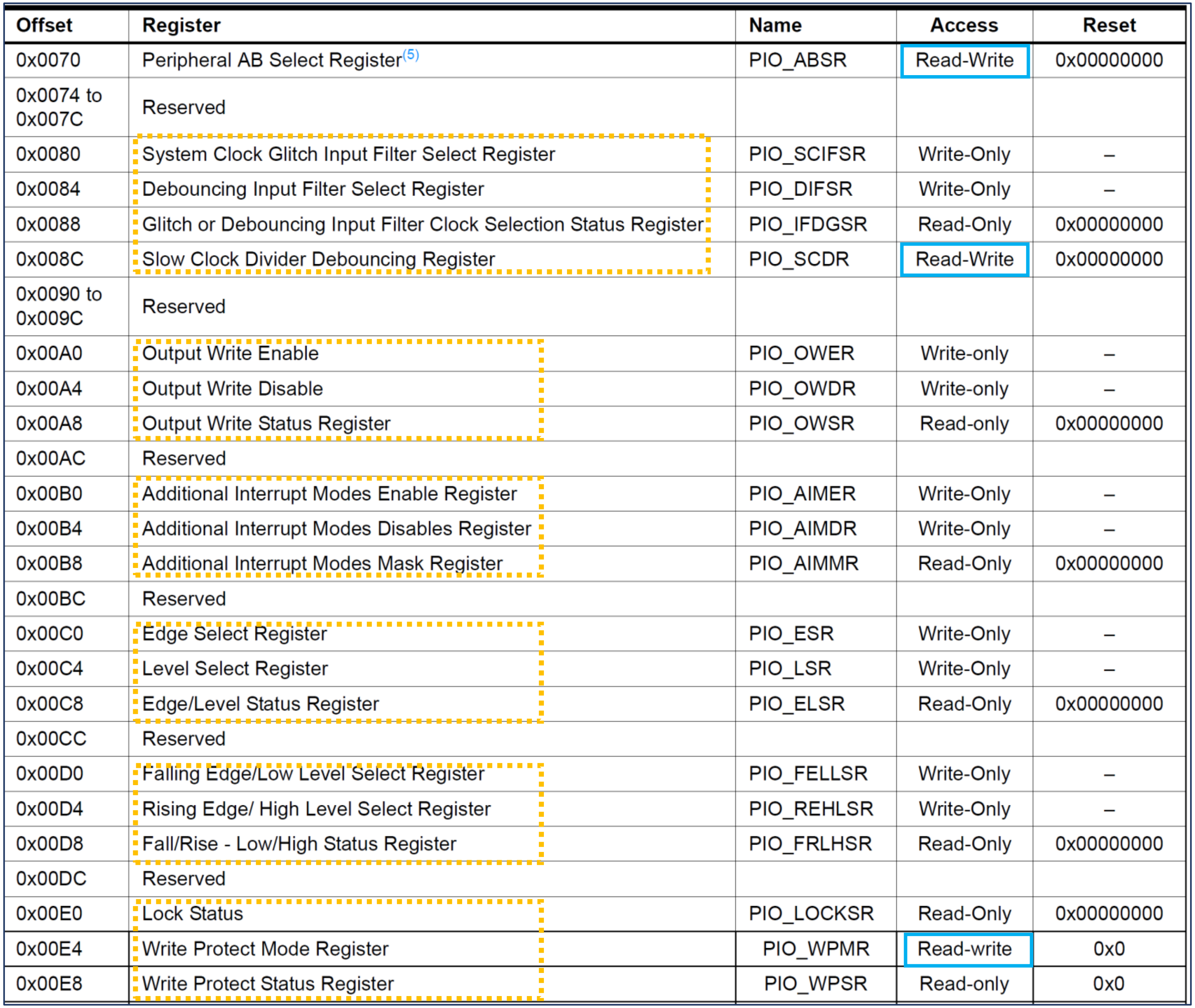


Parallel I/O Controllers: Register Mapping (1)



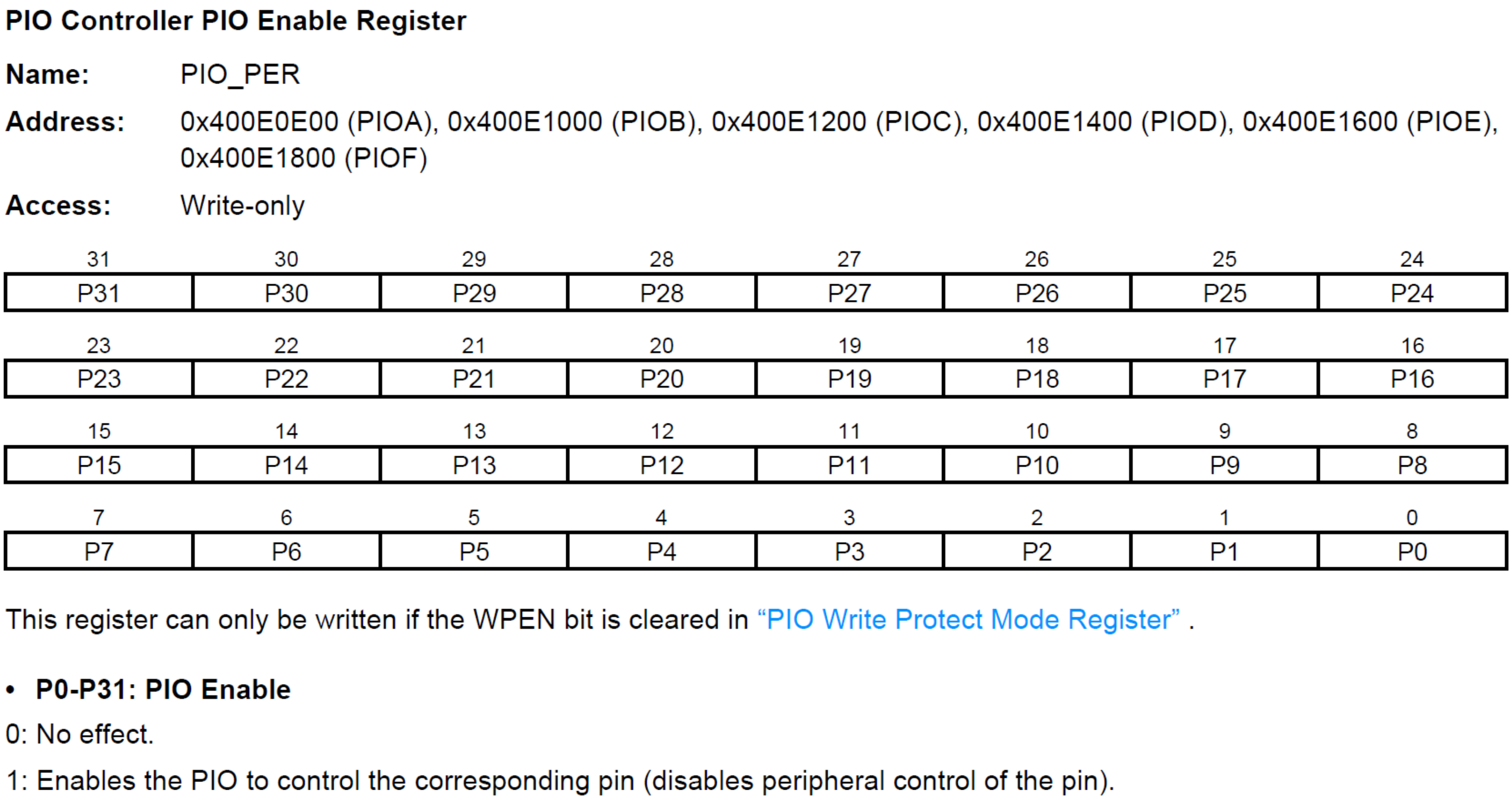
Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	[1]
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000
0x001C	Reserved			
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x0000 0000
0x002C	Reserved			
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or[2] Read-write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	[3]
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register[4]	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved			
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	0x00000000
0x006C	Reserved			

Parallel I/O Controllers: Register Mapping (2)



Offset	Register	Name	Access	Reset
0x0070	Peripheral AB Select Register[5]	PIO_ABSR	Read-Write	0x00000000
0x0074 to 0x007C	Reserved			
0x0080	System Clock Glitch Input Filter Select Register	PIO_SCIFSR	Write-Only	–
0x0084	Debouncing Input Filter Select Register	PIO_DIFSR	Write-Only	–
0x0088	Glitch or Debouncing Input Filter Clock Selection Status Register	PIO_IFDGSR	Read-Only	0x00000000
0x008C	Slow Clock Divider Debouncing Register	PIO_SCDR	Read-Write	0x00000000
0x0090 to 0x009C	Reserved			
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved			
0x00B0	Additional Interrupt Modes Enable Register	PIO_AIMER	Write-Only	–
0x00B4	Additional Interrupt Modes Disables Register	PIO_AIMDR	Write-Only	–
0x00B8	Additional Interrupt Modes Mask Register	PIO_AIMMR	Read-Only	0x00000000
0x00BC	Reserved			
0x00C0	Edge Select Register	PIO_ESR	Write-Only	–
0x00C4	Level Select Register	PIO_LSR	Write-Only	–
0x00C8	Edge/Level Status Register	PIO_ELSR	Read-Only	0x00000000
0x00CC	Reserved			
0x00D0	Falling Edge/Low Level Select Register	PIO_FELLSR	Write-Only	–
0x00D4	Rising Edge/ High Level Select Register	PIO_REHLSR	Write-Only	–
0x00D8	Fall/Rise - Low/High Status Register	PIO_FRLHSR	Read-Only	0x00000000
0x00DC	Reserved			
0x00E0	Lock Status	PIO_LOCKSR	Read-Only	0x00000000
0x00E4	Write Protect Mode Register	PIO_WPMR	Read-write	0x0
0x00E8	Write Protect Status Register	PIO_WPSR	Read-only	0x0

PIO Controllers: PIO_PER Register



PIO Controller PIO Enable Register

Name: PIO_PER

Address: 0x400E0E00 (PIOA), 0x400E1000 (PIOB), 0x400E1200 (PIOC), 0x400E1400 (PIOD), 0x400E1600 (PIOE), 0x400E1800 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

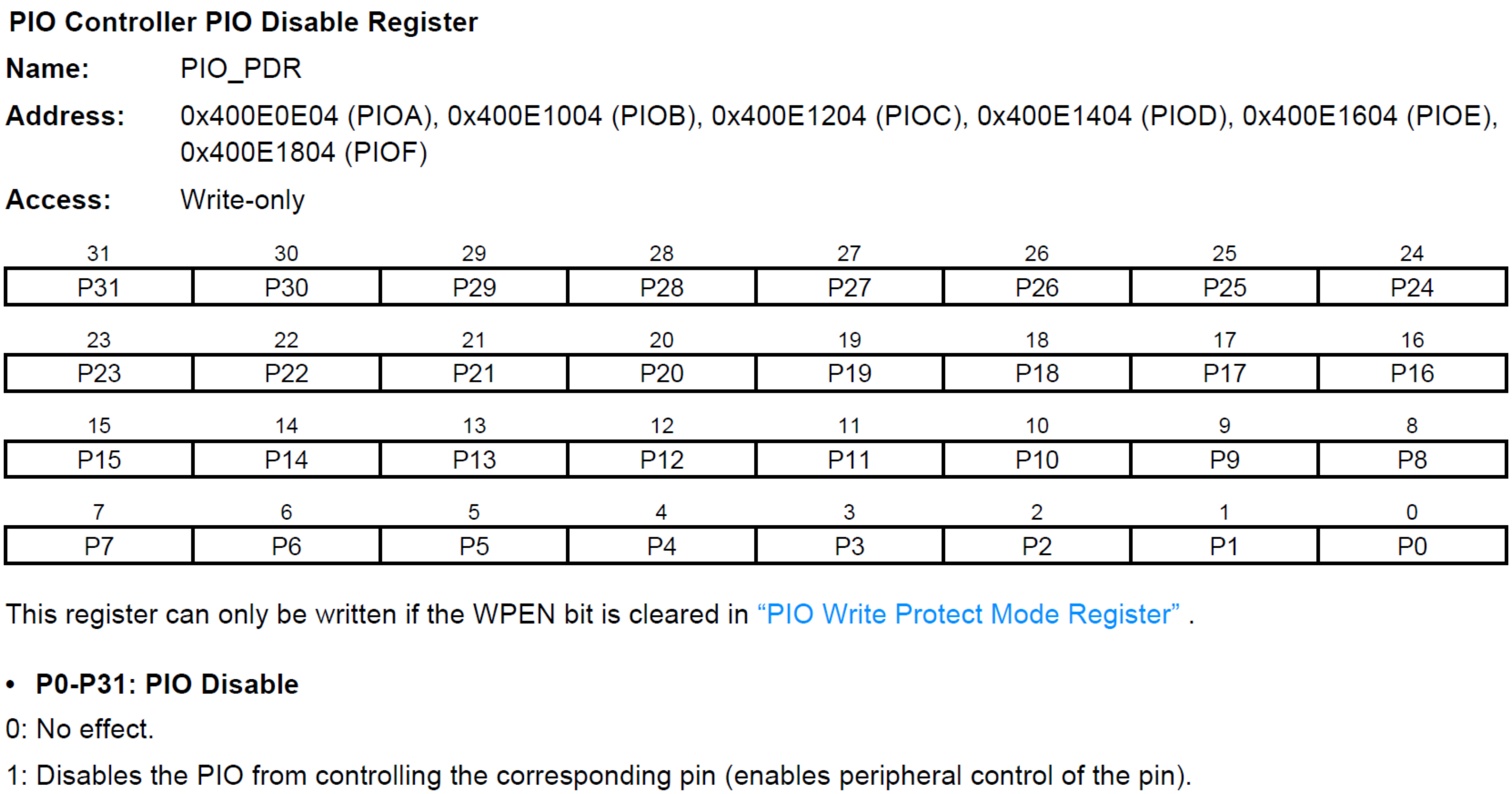
- **P0-P31: PIO Enable**

0: No effect.

1: Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

เขียน 1 หมายถึง เปิดใช้งานขา GPIO Pin (และปิดการใช้งาน Peripheral Pin)
แต่ถ้าเขียน 0 ไม่มีส่งผลต่อการเปลี่ยนแปลง

PIO Controllers: PIO_PDR Register



PIO Controller PIO Disable Register

Name: PIO_PDR

Address: 0x400E0E04 (PIOA), 0x400E1004 (PIOB), 0x400E1204 (PIOC), 0x400E1404 (PIOD), 0x400E1604 (PIOE), 0x400E1804 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

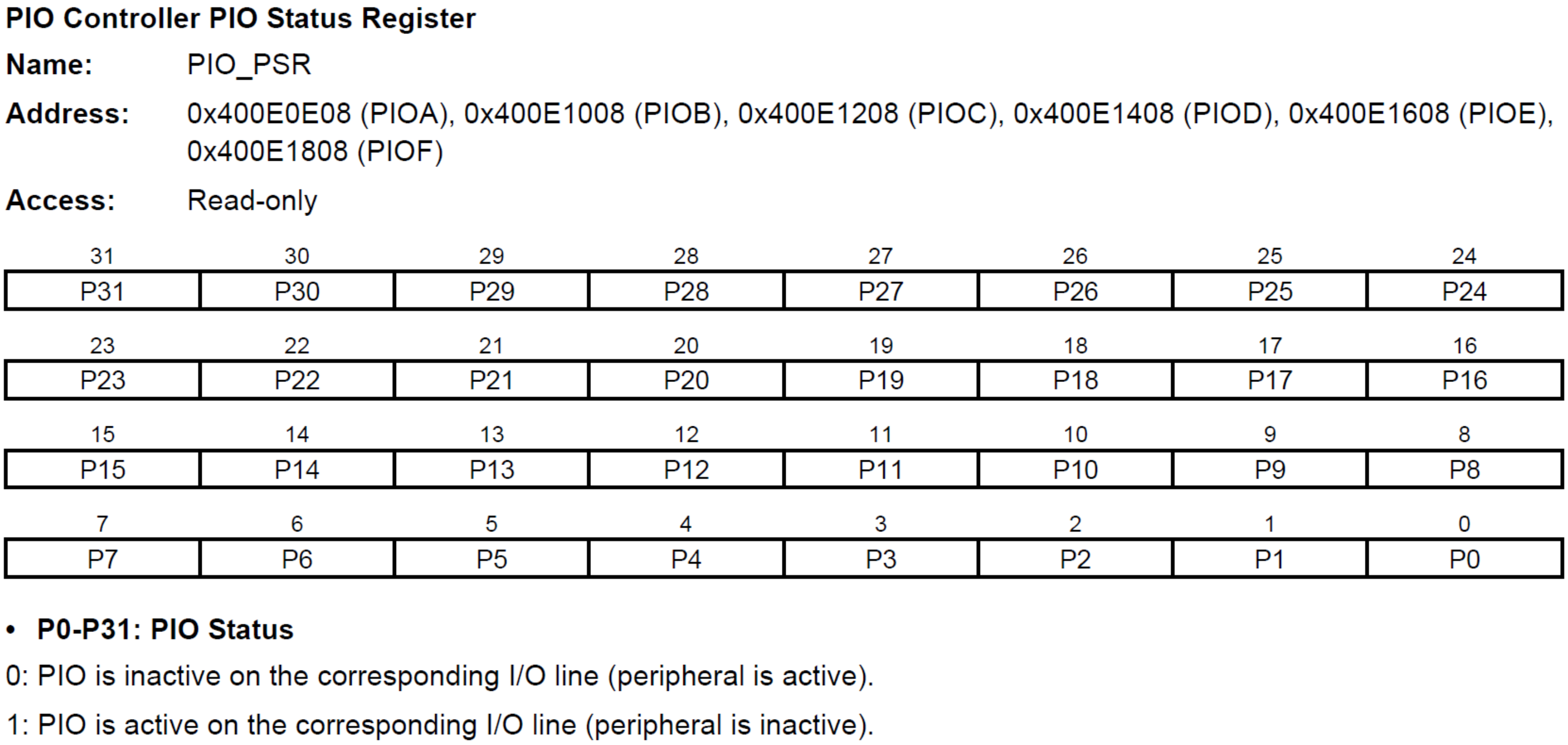
- **P0-P31: PIO Disable**

0: No effect.

1: Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

เขียน 1 หมายถึง เปิดใช้งานขา Peripheral Pin (และปิดการใช้งาน GPIO Pin) แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_PSR Register



PIO Controller PIO Status Register

Name: PIO_PSR

Address: 0x400E0E08 (PIOA), 0x400E1008 (PIOB), 0x400E1208 (PIOC), 0x400E1408 (PIOD), 0x400E1608 (PIOE), 0x400E1808 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

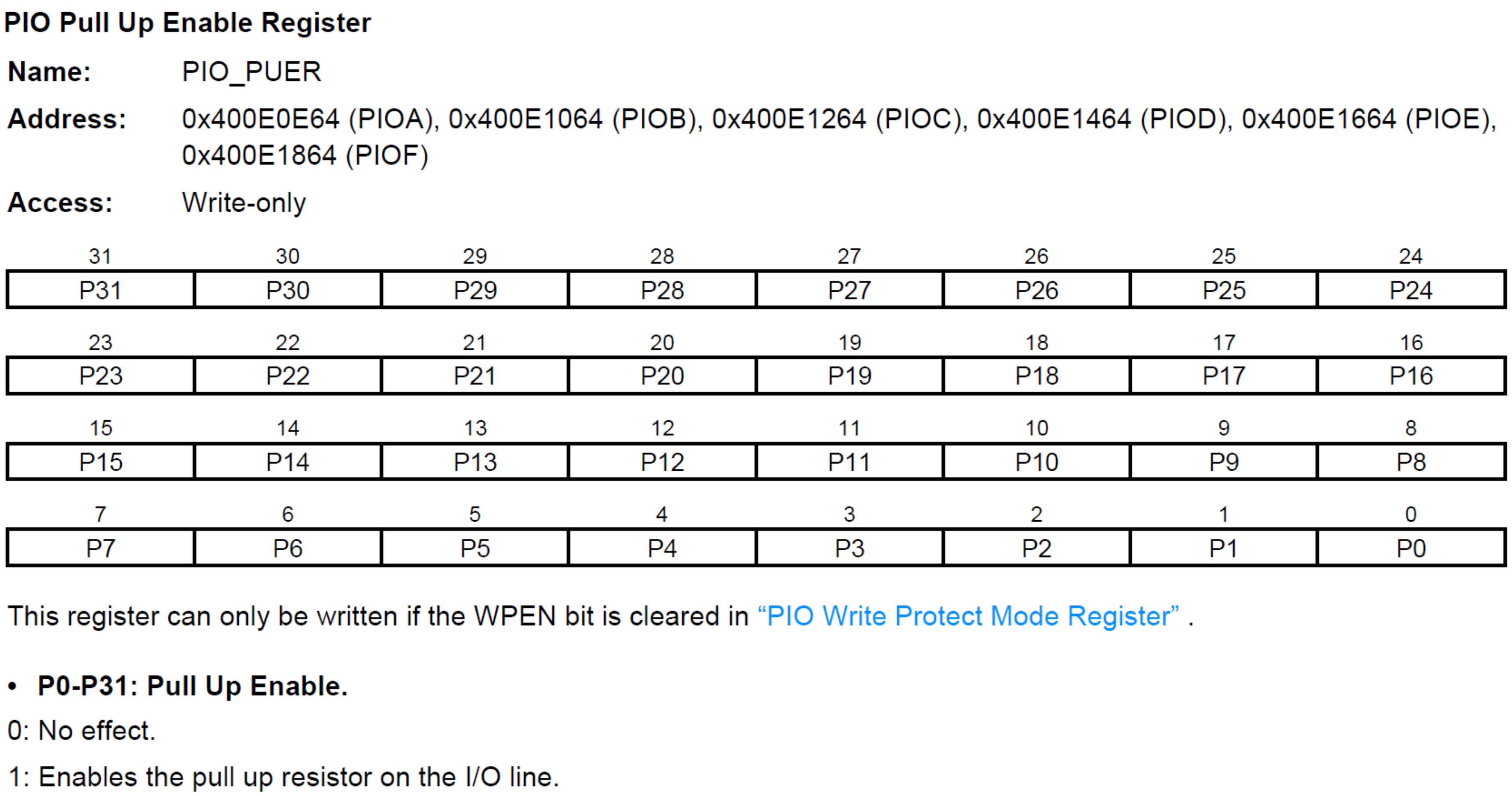
- **P0-P31: PIO Status**

0: PIO is inactive on the corresponding I/O line (peripheral is active).

1: PIO is active on the corresponding I/O line (peripheral is inactive).

รีจิสเตอร์มีไว้สำหรับอ่านค่าเท่านั้น เพื่อตรวจสอบดูว่า I/O Pin ในตำแหน่งบิตใด ทำงานในโหมด GPIO Pin (1) หรือ Peripheral Pin (0)

PIO Controllers: PIO_PUER Register



PIO Pull Up Enable Register

Name: PIO_PUER

Address: 0x400E0E64 (PIOA), 0x400E1064 (PIOB), 0x400E1264 (PIOC), 0x400E1464 (PIOD), 0x400E1664 (PIOE), 0x400E1864 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

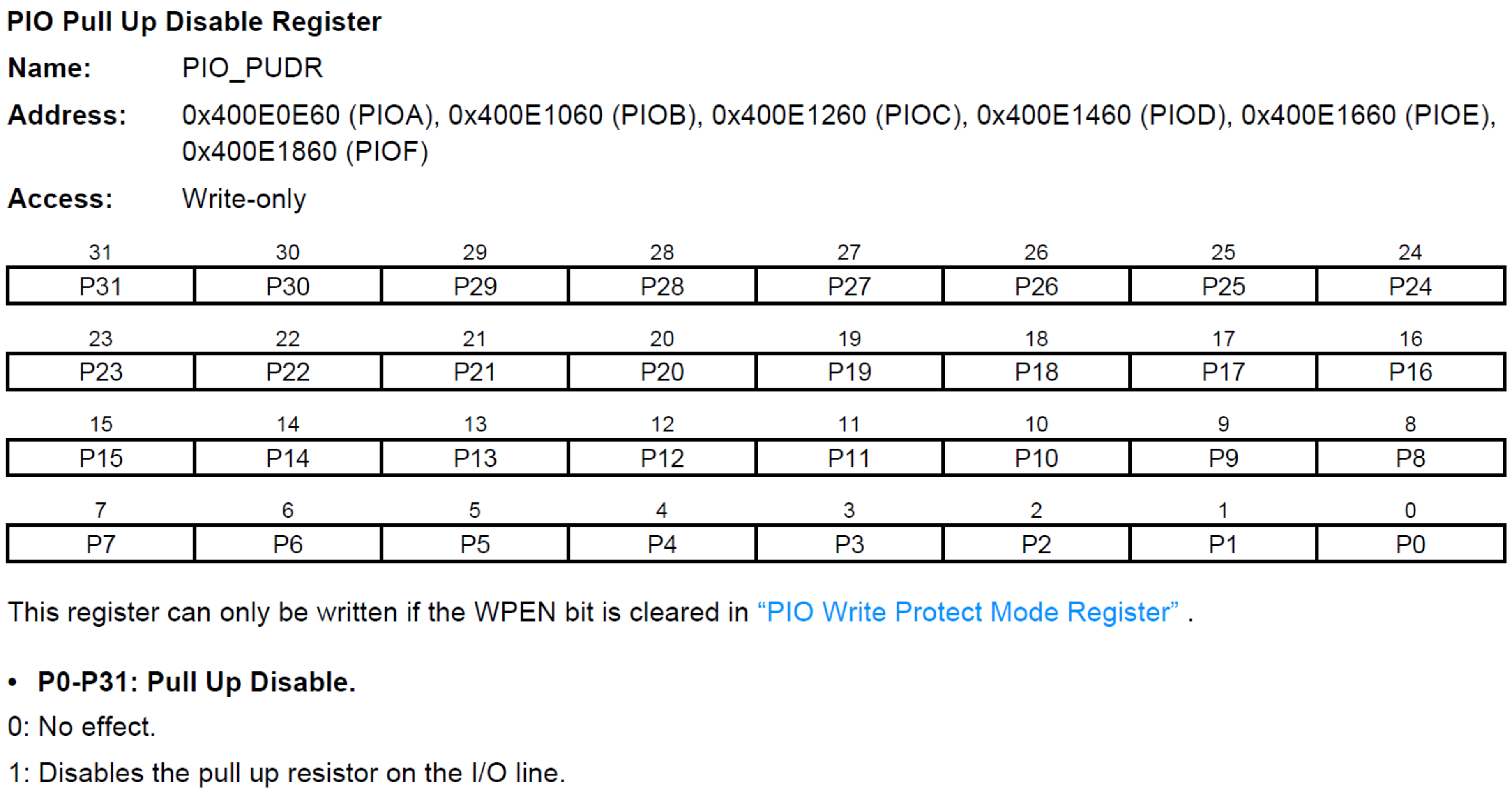
- **P0-P31: Pull Up Enable.**

0: No effect.

1: Enables the pull up resistor on the I/O line.

เขียน 1 หมายถึง เปิดใช้งาน Enable Pull-Up Register ที่ขา I/O แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_PUDR Register



PIO Pull Up Disable Register

Name: PIO_PUDR

Address: 0x400E0E60 (PIOA), 0x400E1060 (PIOB), 0x400E1260 (PIOC), 0x400E1460 (PIOD), 0x400E1660 (PIOE), 0x400E1860 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

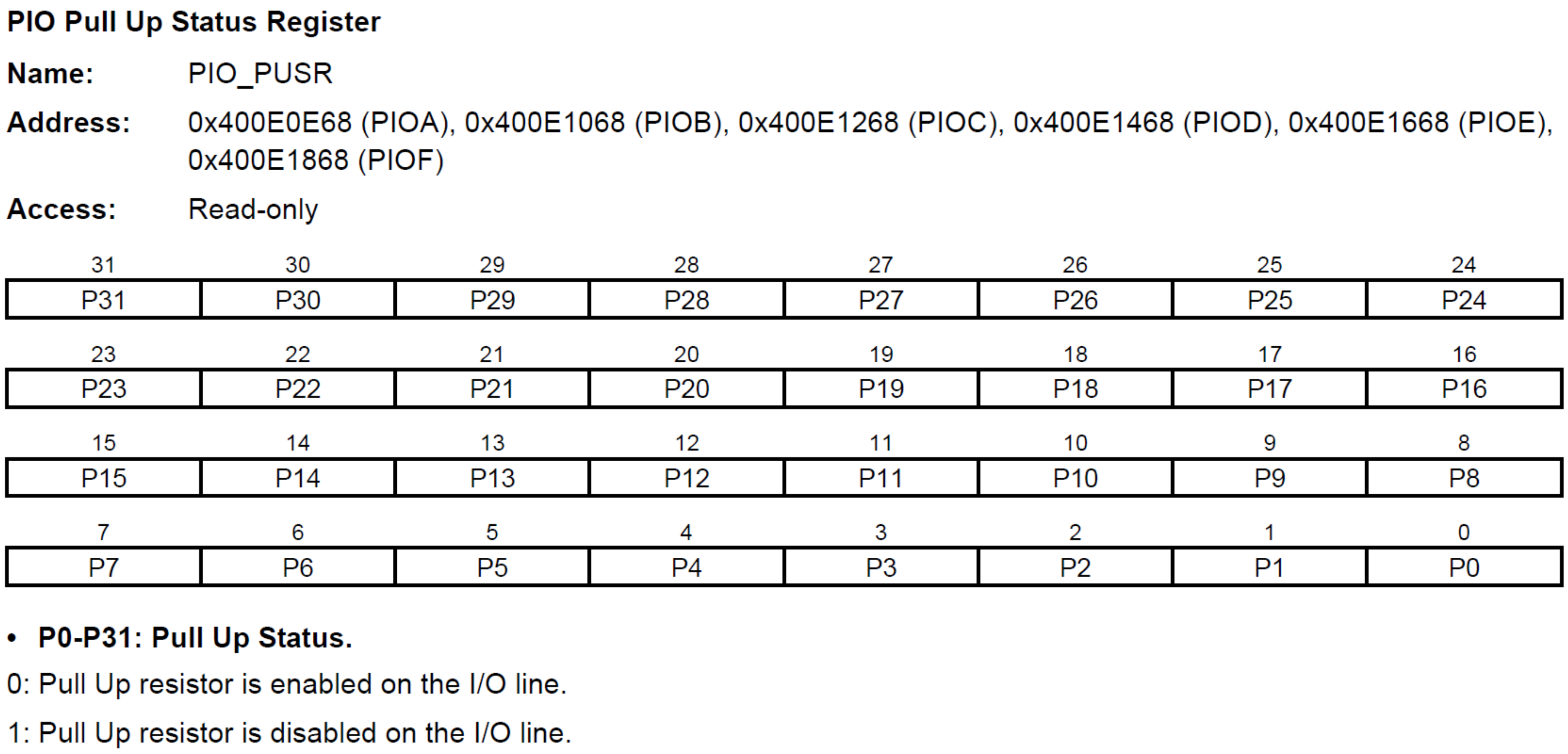
- **P0-P31: Pull Up Disable.**

0: No effect.

1: Disables the pull up resistor on the I/O line.

เขียน 1 หมายถึง ปิดใช้งาน Disable Pull-Up Register ที่ขา I/O แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_PUSR Register



PIO Pull Up Status Register

Name: PIO_PUSR

Address: 0x400E0E68 (PIOA), 0x400E1068 (PIOB), 0x400E1268 (PIOC), 0x400E1468 (PIOD), 0x400E1668 (PIOE), 0x400E1868 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

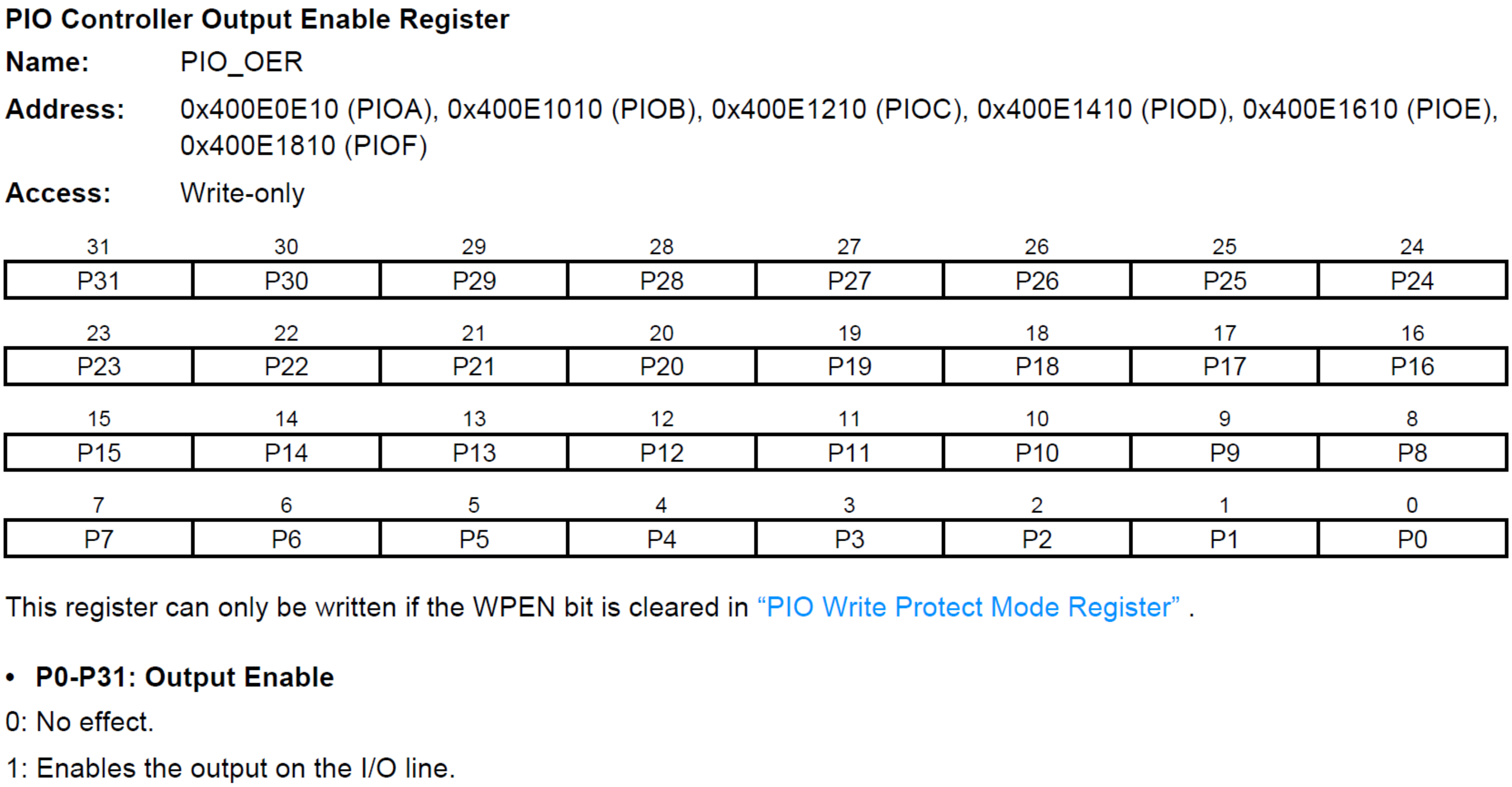
- **P0-P31: Pull Up Status.**

0: Pull Up resistor is enabled on the I/O line.

1: Pull Up resistor is disabled on the I/O line.

รีจิสเตอร์มีไว้สำหรับอ่านค่าเท่านั้น เพื่อตรวจสอบดูว่า I/O Pin ในตำแหน่งบิตใด มีการเปิดใช้งาน Pull-Up Register (0=Enabled, 1=Disabled)

PIO Controllers: PIO_OER Register



PIO Controller Output Enable Register

Name: PIO_OER

Address: 0x400E0E10 (PIOA), 0x400E1010 (PIOB), 0x400E1210 (PIOC), 0x400E1410 (PIOD), 0x400E1610 (PIOE), 0x400E1810 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

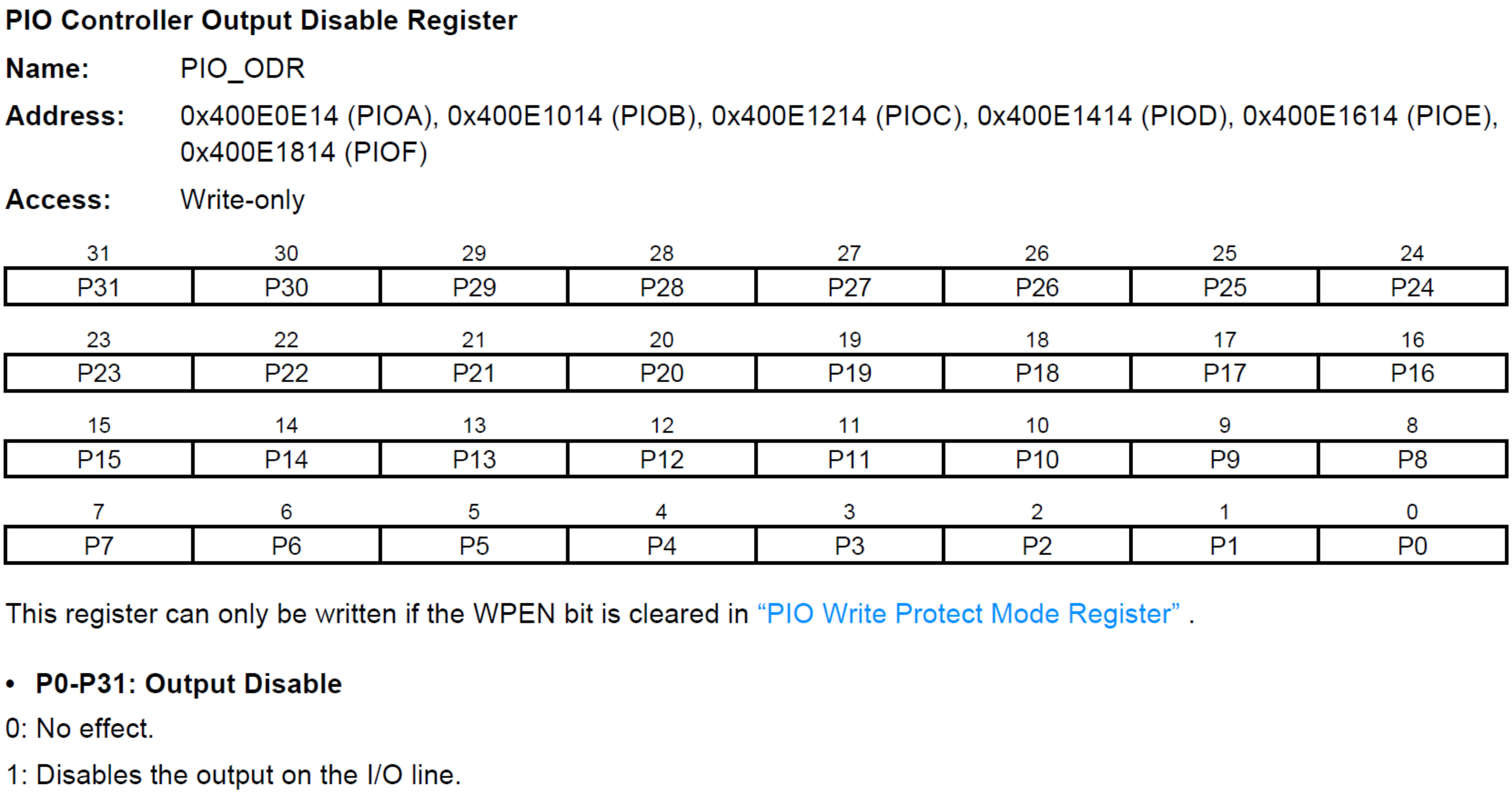
- **P0-P31: Output Enable**

0: No effect.

1: Enables the output on the I/O line.

เขียน 1 หมายถึง ใช้งานขา GPIO Pin ให้เป็นเอาต์พุต (Output) แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_ODR Register



PIO Controller Output Disable Register

Name: PIO_ODR

Address: 0x400E0E14 (PIOA), 0x400E1014 (PIOB), 0x400E1214 (PIOC), 0x400E1414 (PIOD), 0x400E1614 (PIOE), 0x400E1814 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

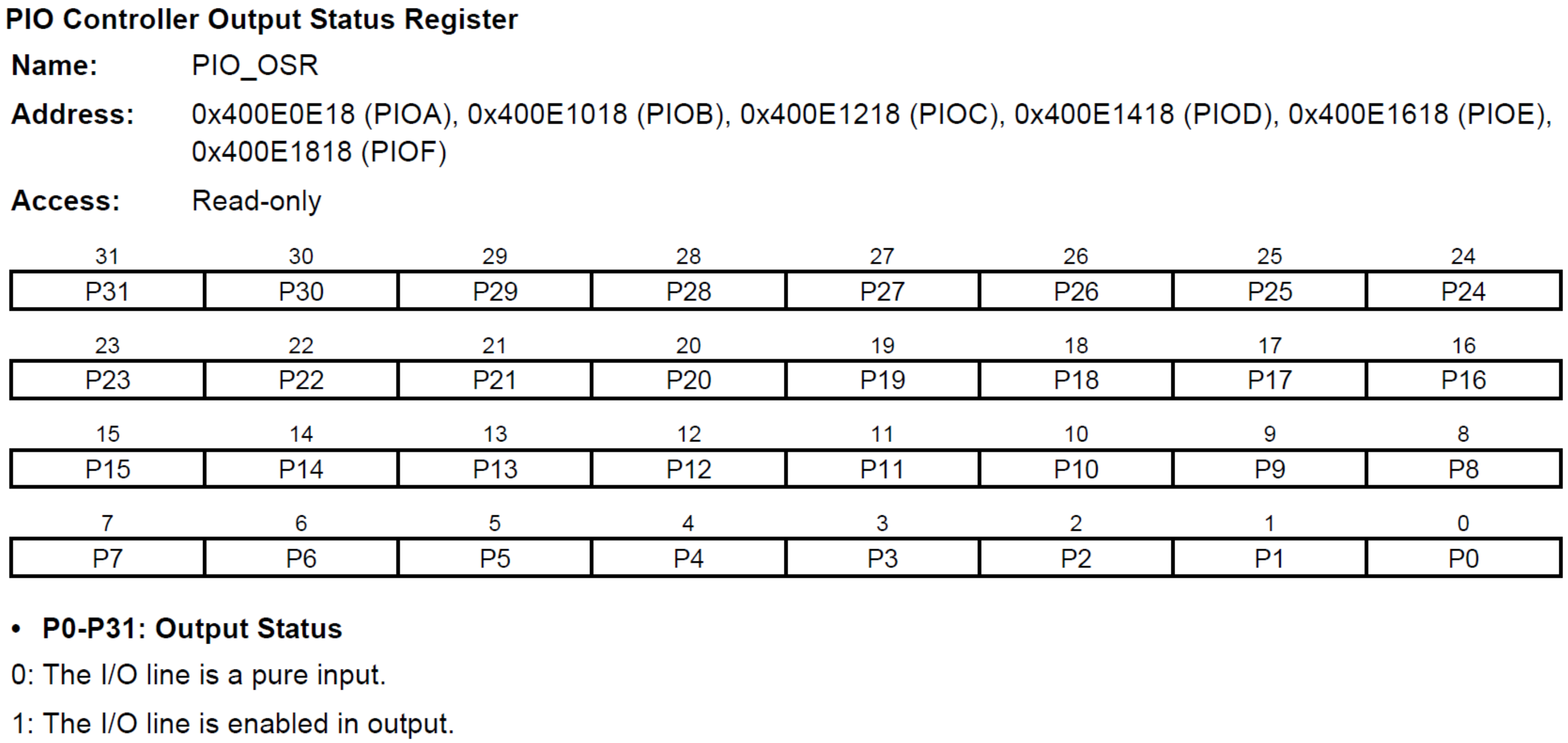
- **P0-P31: Output Disable**

0: No effect.

1: Disables the output on the I/O line.

เขียน 1 หมายถึง ใช้งานขา GPIO Pin ให้เป็นอินพุต (Input) แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_OSR Register



PIO Controller Output Status Register

Name: PIO_OSR

Address: 0x400E0E18 (PIOA), 0x400E1018 (PIOB), 0x400E1218 (PIOC), 0x400E1418 (PIOD), 0x400E1618 (PIOE), 0x400E1818 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

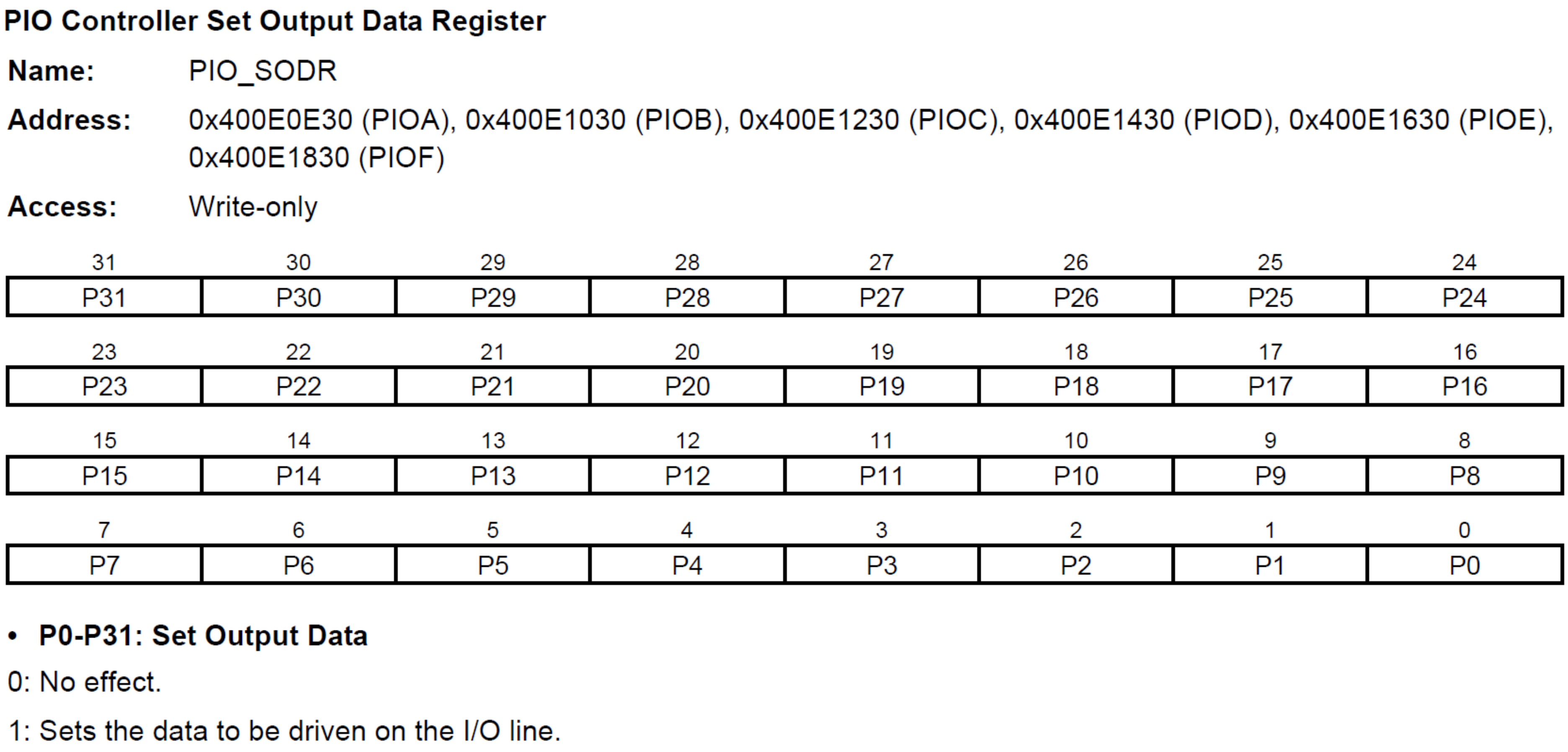
- **P0-P31: Output Status**

0: The I/O line is a pure input.

1: The I/O line is enabled in output.

รีจิสเตอร์มีไว้สำหรับอ่านค่าเท่านั้น เพื่อตรวจสอบดูว่า I/O Pin ในตำแหน่งบิตใด มีการเปิดใช้งานเป็นอินพุต (0=Input Direction) หรือเป็นเอาต์พุต (1=Output Direction)

PIO Controllers: PIO_SODR Register



PIO Controller Set Output Data Register

Name: PIO_SODR

Address: 0x400E0E30 (PIOA), 0x400E1030 (PIOB), 0x400E1230 (PIOC), 0x400E1430 (PIOD), 0x400E1630 (PIOE), 0x400E1830 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

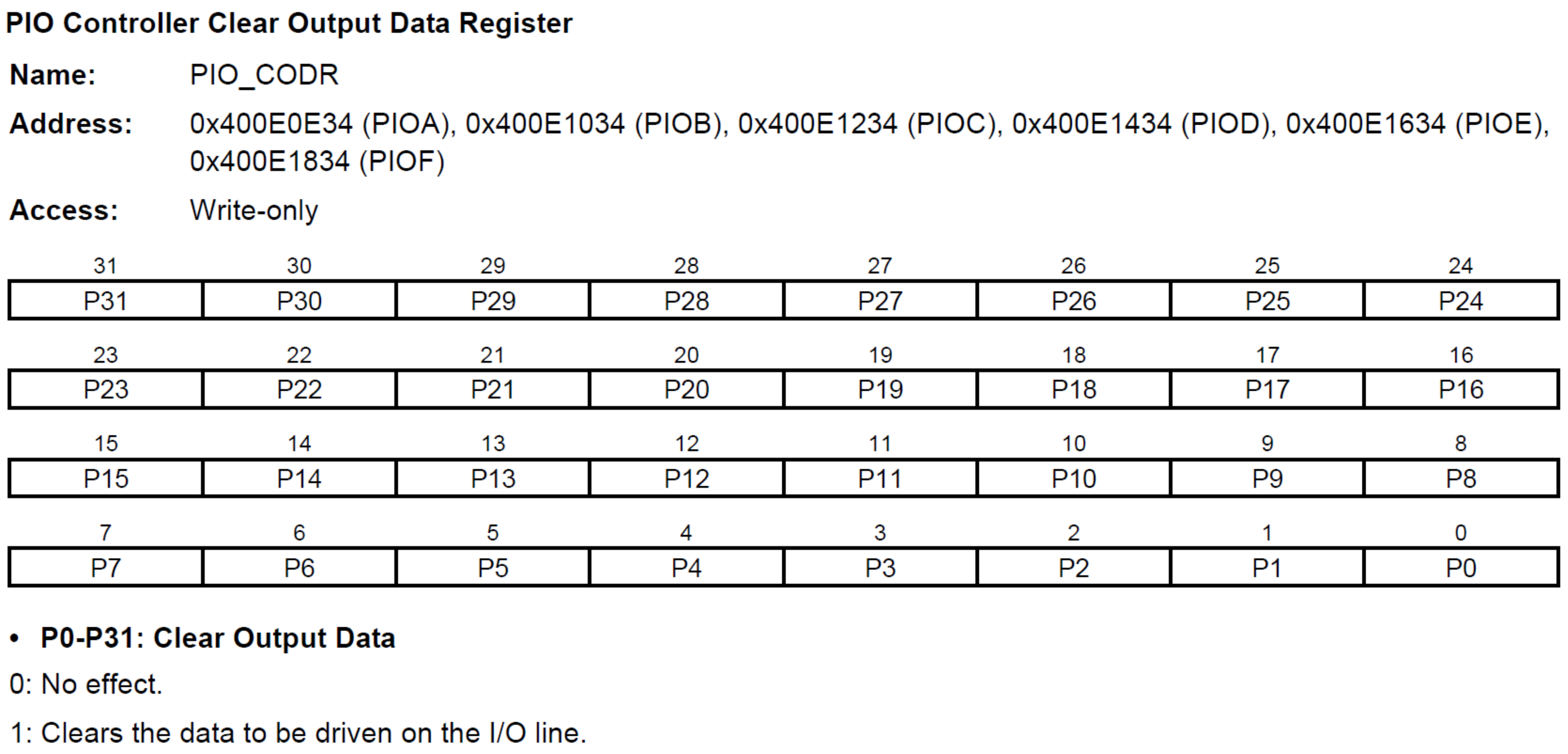
- **P0-P31: Set Output Data**

0: No effect.

1: Sets the data to be driven on the I/O line.

เขียน 1 หมายถึง ให้เอาต์พุตเป็น High แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_CODR Register



PIO Controller Clear Output Data Register

Name: PIO_CODR

Address: 0x400E0E34 (PIOA), 0x400E1034 (PIOB), 0x400E1234 (PIOC), 0x400E1434 (PIOD), 0x400E1634 (PIOE), 0x400E1834 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

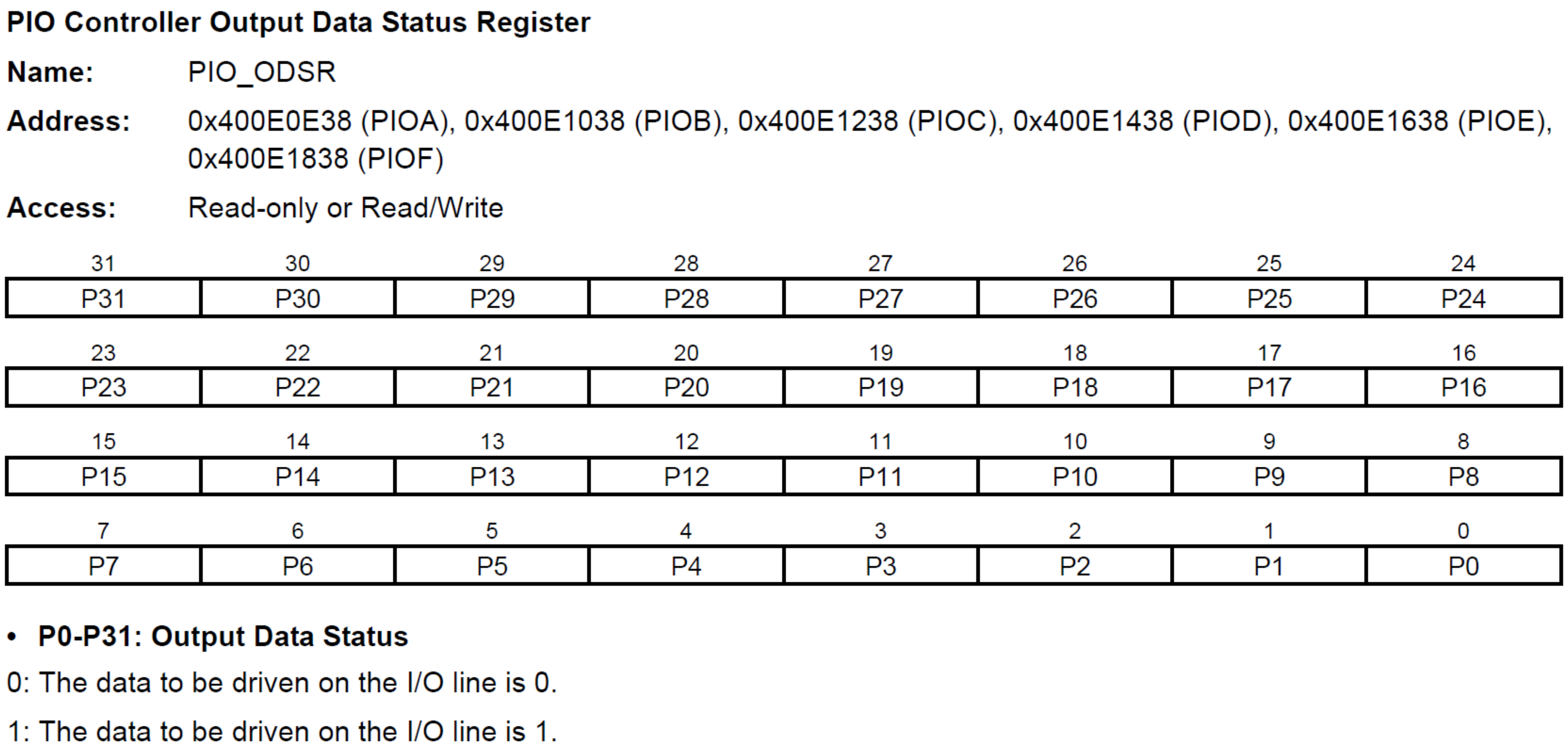
- **P0-P31: Clear Output Data**

0: No effect.

1: Clears the data to be driven on the I/O line.

เขียน 1 หมายถึง เคลียร์เอาต์พุตให้เป็น Low แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_ODSR Register



PIO Controller Output Data Status Register

Name: PIO_ODSR

Address: 0x400E0E38 (PIOA), 0x400E1038 (PIOB), 0x400E1238 (PIOC), 0x400E1438 (PIOD), 0x400E1638 (PIOE), 0x400E1838 (PIOF)

Access: Read-only or Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

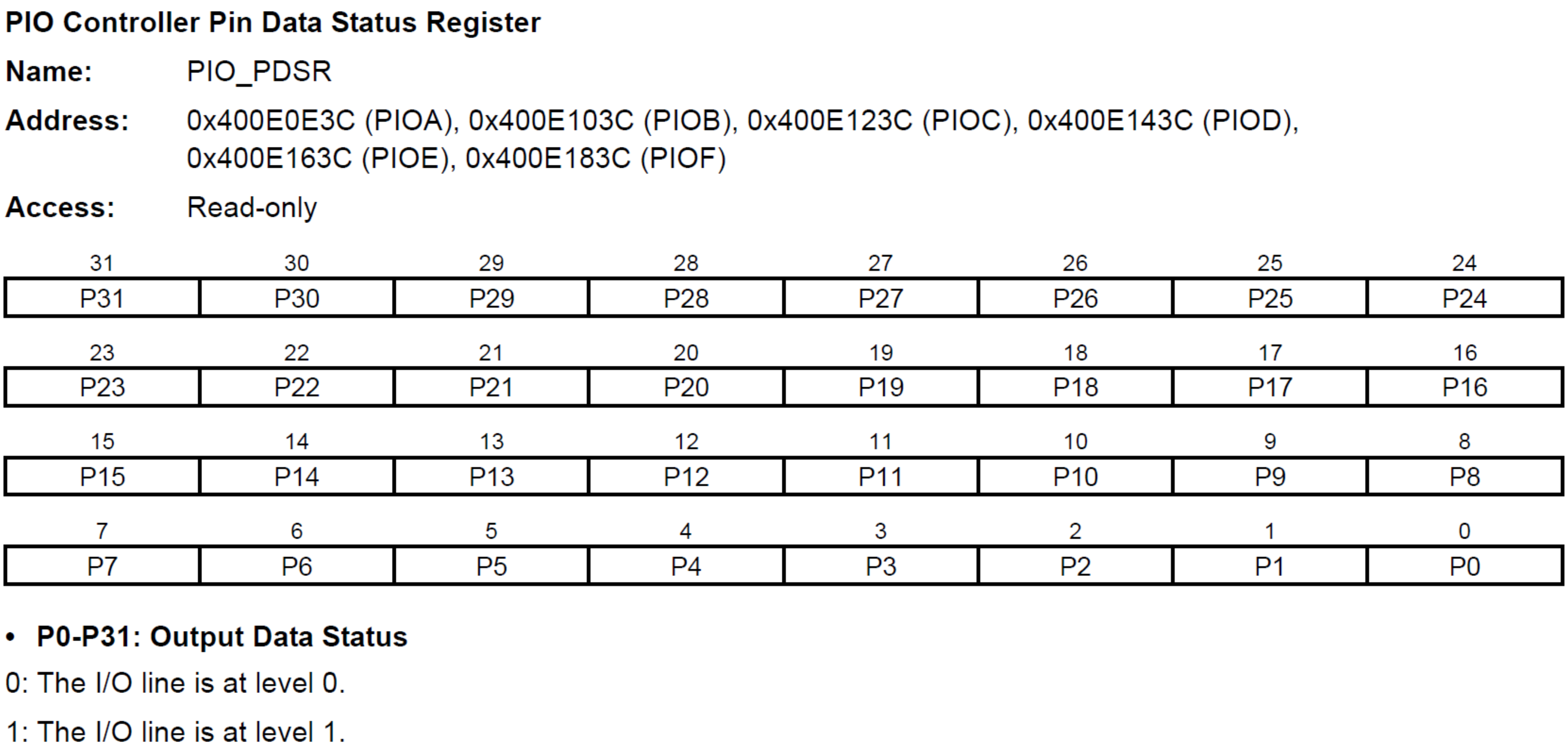
- **P0-P31: Output Data Status**

0: The data to be driven on the I/O line is 0.

1: The data to be driven on the I/O line is 1.

รีจิสเตอร์มีไว้สำหรับอ่าน (หรือเขียนก็ได้ด้วย) ถ้าอ่าน ก็เพื่อตรวจสอบดูว่า I/O Pin ในตำแหน่งบิตใด มีสถานะของเอาต์พุต เป็น 0 (Low) หรือ 1 (High)

PIO Controllers: PIO_PDSR Register



PIO Controller Pin Data Status Register

Name: PIO_PDSR

Address: 0x400E0E3C (PIOA), 0x400E103C (PIOB), 0x400E123C (PIOC), 0x400E143C (PIOD),
0x400E163C (PIOE), 0x400E183C (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

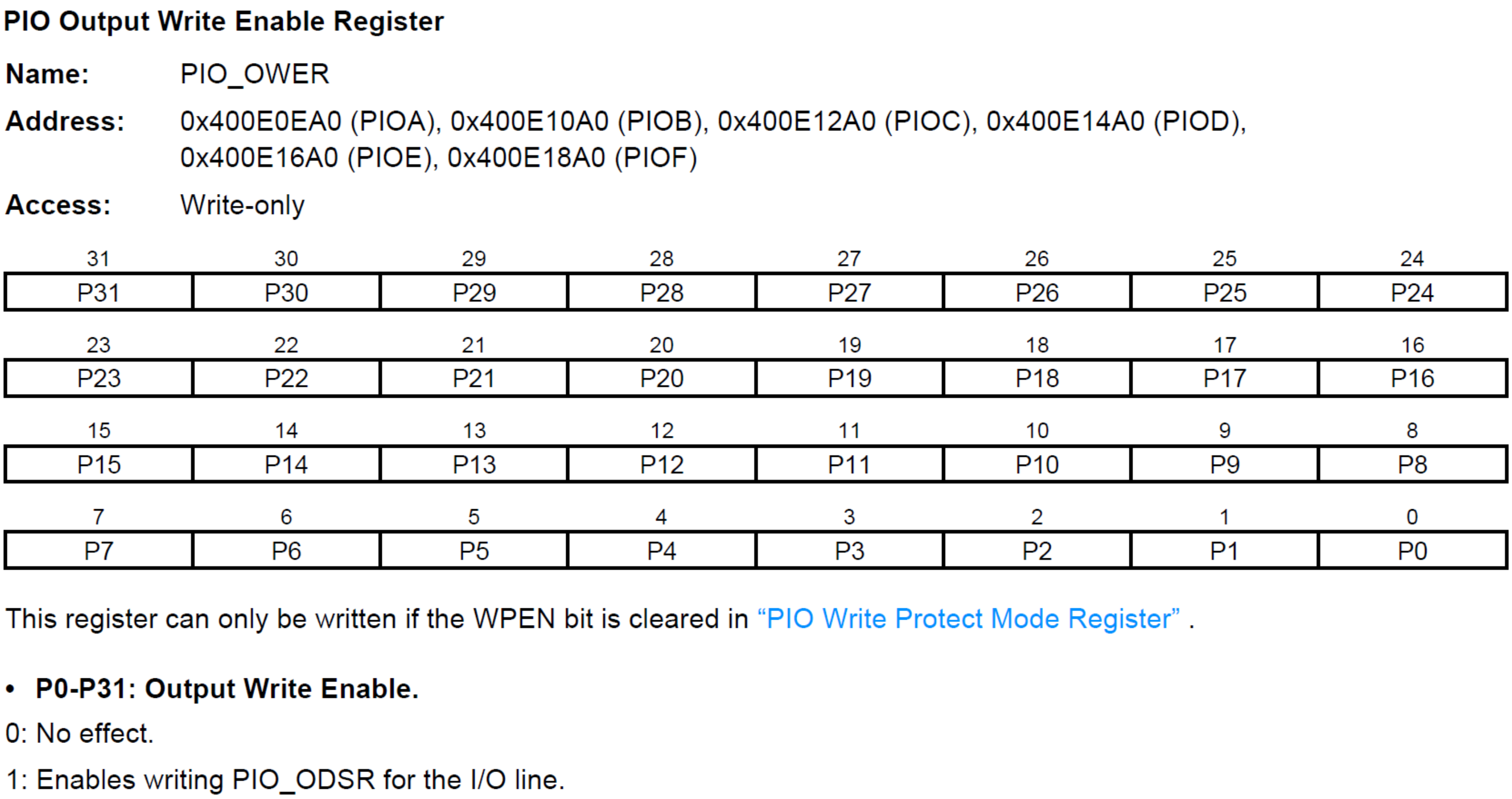
- **P0-P31: Output Data Status**

0: The I/O line is at level 0.

1: The I/O line is at level 1.

รีจิสเตอร์มีไว้สำหรับอ่านค่าเท่านั้น เพื่อตรวจสอบดูว่า I/O Pin ในตำแหน่งบิตใด มีสถานะลอจิกเป็น 0 (Low) หรือ 1 (High)

PIO Controllers: PIO_OWER Register



PIO Output Write Enable Register

Name: PIO_OWER

Address: 0x400E0EA0 (PIOA), 0x400E10A0 (PIOB), 0x400E12A0 (PIOC), 0x400E14A0 (PIOD),
0x400E16A0 (PIOE), 0x400E18A0 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

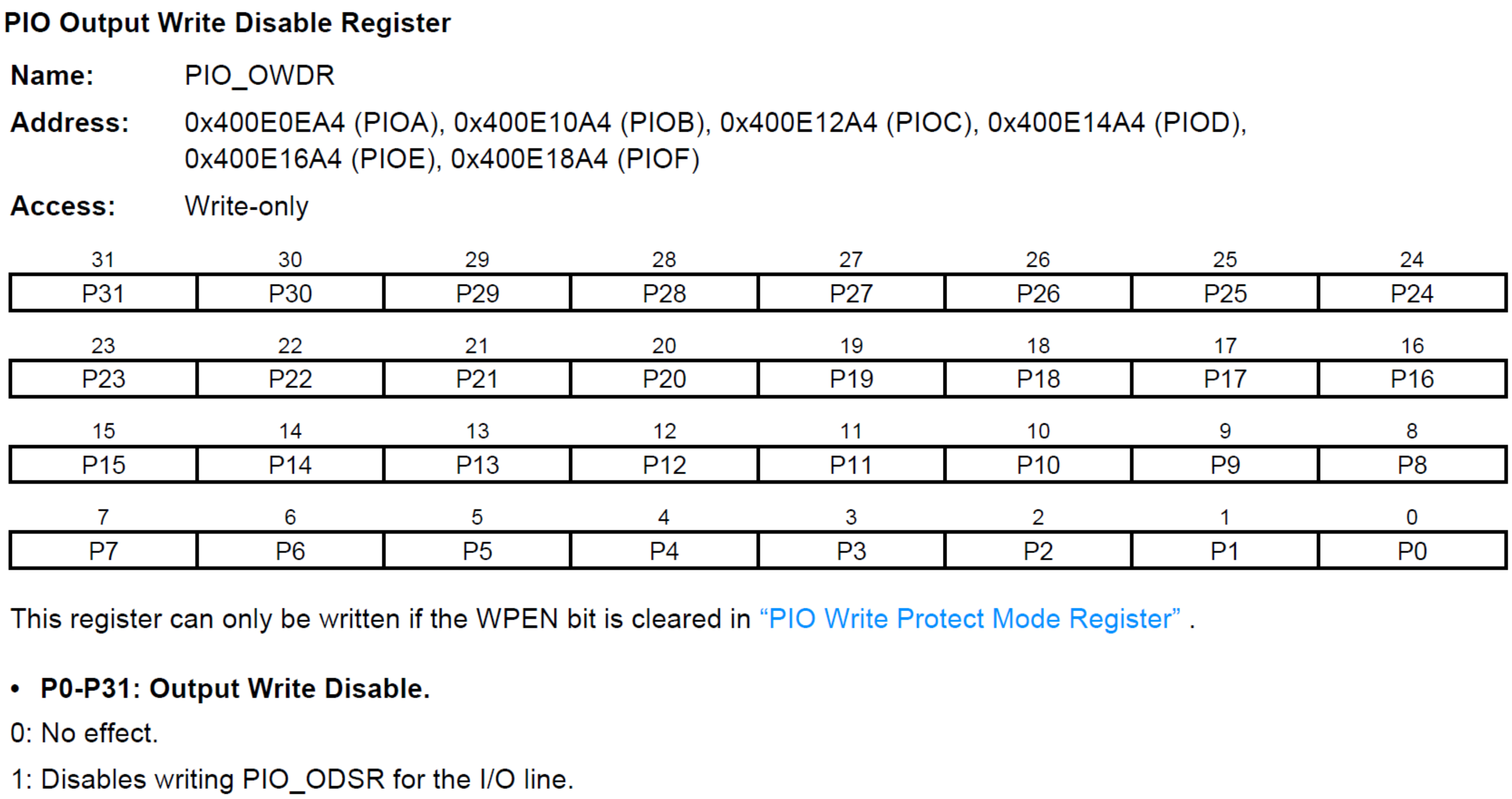
- **P0-P31: Output Write Enable.**

0: No effect.

1: Enables writing PIO_ODSR for the I/O line.

เขียน 1 หมายถึง อนุญาตให้เขียนค่าไปยังรีจิสเตอร์ PIO_ODSR และมีผลต่อเอาต์พุต (Output Write Enable) แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_OWDR Register



PIO Output Write Disable Register

Name: PIO_OWDR

Address: 0x400E0EA4 (PIOA), 0x400E10A4 (PIOB), 0x400E12A4 (PIOC), 0x400E14A4 (PIOD), 0x400E16A4 (PIOE), 0x400E18A4 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

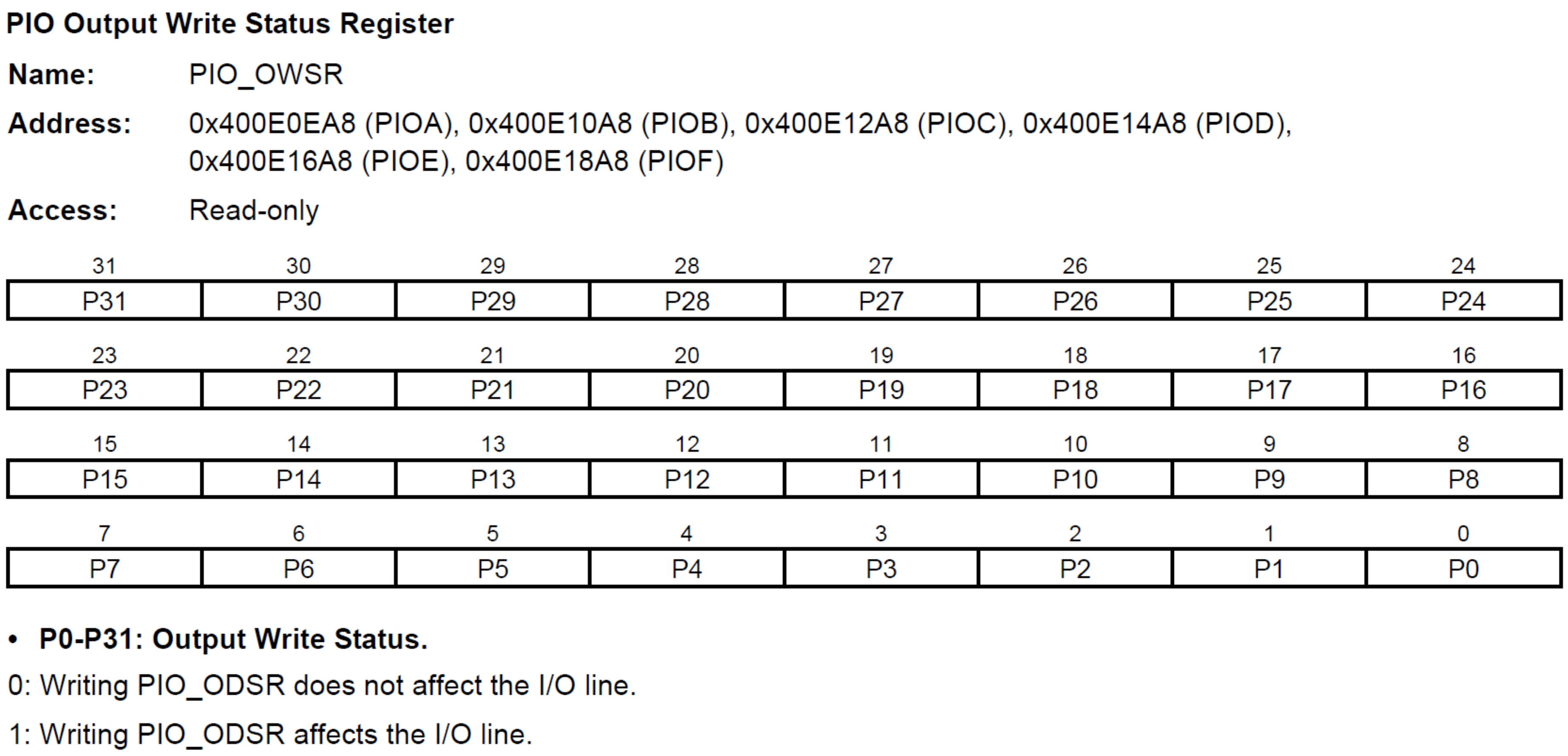
- **P0-P31: Output Write Disable.**

0: No effect.

1: Disables writing PIO_ODSR for the I/O line.

เขียน 1 หมายถึง ปิดการเขียนค่าไปยังรีจิสเตอร์ PIO_ODSR และไม่ส่งผลต่อเอาต์พุต (Output Write Disable) แต่ถ้าเขียน 0 ไม่มีการเปลี่ยนแปลง

PIO Controllers: PIO_OWSR Register



PIO Output Write Status Register

Name: PIO_OWSR

Address: 0x400E0EA8 (PIOA), 0x400E10A8 (PIOB), 0x400E12A8 (PIOC), 0x400E14A8 (PIOD), 0x400E16A8 (PIOE), 0x400E18A8 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Write Status.**

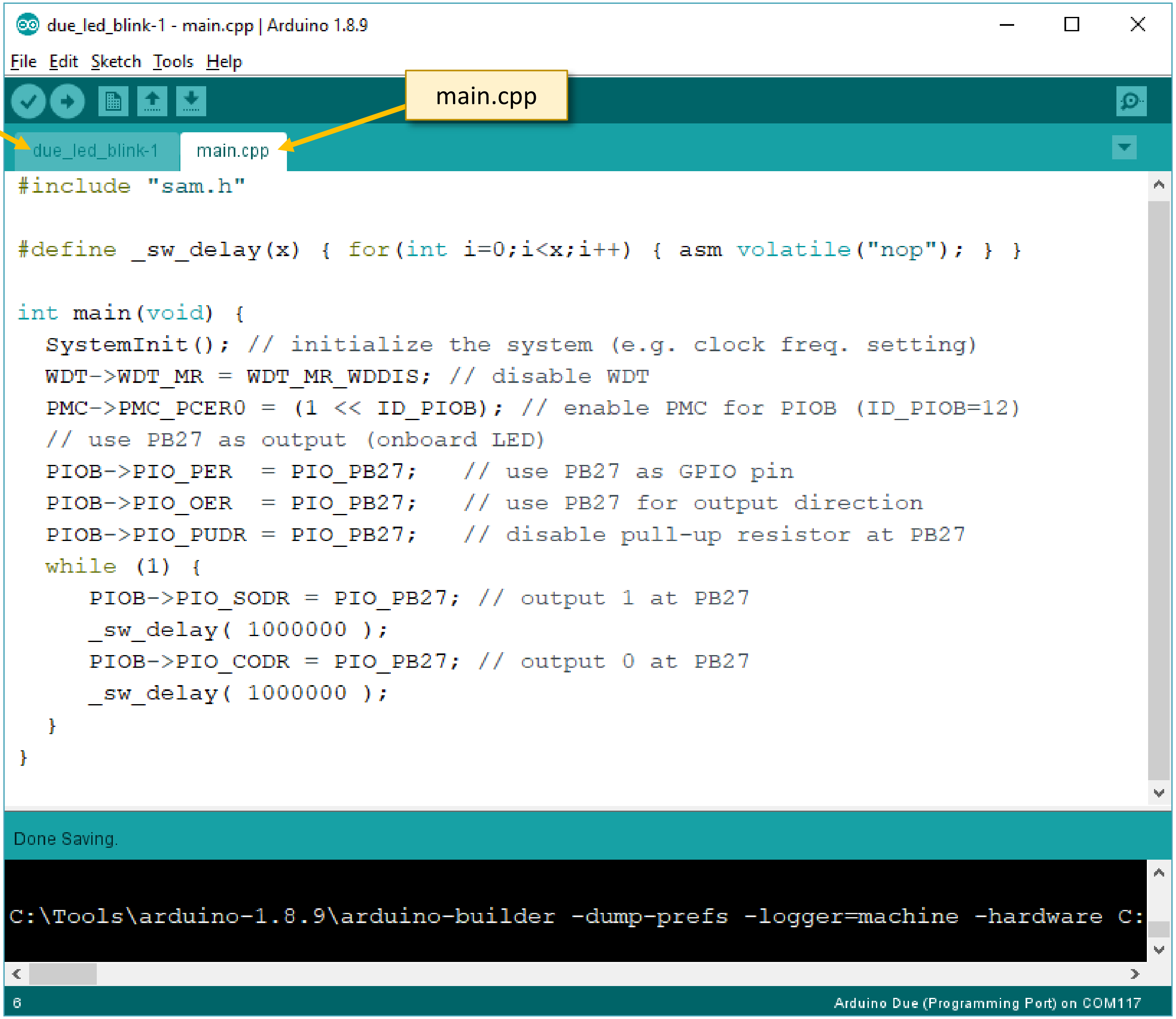
0: Writing PIO_ODSR does not affect the I/O line.

1: Writing PIO_ODSR affects the I/O line.

รีจิสเตอร์มีไว้สำหรับอ่านค่าเท่านั้น เพื่อตรวจสอบดูว่า I/O Pin ในตำแหน่งบิตใด ที่เมื่อมีการเปลี่ยนแปลงค่าของ PIO_ODSR แล้วมีผลต่อเอาต์พุต

Programming Arduino DUE with Arduino IDE: LED Blink

Empty Arduino Sketch file (.ino)


due_led_blink-1 - main.cpp | Arduino 1.8.9
File Edit Sketch Tools Help
due_led_blink-1
main.cpp

main.cpp

```
#include "sam.h"

#define _sw_delay(x) { for(int i=0;i<x;i++) { asm volatile("nop"); } }

int main(void) {
  SystemInit(); // initialize the system (e.g. clock freq. setting)
  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  PMC->PMC_PCER0 = (1 << ID_PIOB); // enable PMC for PIOB (ID_PIOB=12)
  // use PB27 as output (onboard LED)
  PIOB->PIO_PER  = PIO_PB27;   // use PB27 as GPIO pin
  PIOB->PIO_OER  = PIO_PB27;   // use PB27 for output direction
  PIOB->PIO_PUDR = PIO_PB27;   // disable pull-up resistor at PB27
  while (1) {
     PIOB->PIO_SODR = PIO_PB27; // output 1 at PB27
     _sw_delay( 1000000 );
     PIOB->PIO_CODR = PIO_PB27; // output 0 at PB27
     _sw_delay( 1000000 );
  }
}
```

Done Saving.

```
C:\Tools\arduino-1.8.9\arduino-builder -dump-prefs -logger=machine -hardware C:
```

Programming Arduino DUE with Arduino IDE: LED Blink

https://github.com/arduino/ArduinoModule-CMSIS-Atmel/blob/master/CMSIS-Atmel/CMSIS/Device/ATMEL/sam.h

```
#include "sam.h"

#define _sw_delay(x) { for(int i=0;i<x;i++) { asm volatile("nop"); } }

int main(void) {
  SystemInit(); // initialize the system (e.g. clock freq. setting)

  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  PMC->PMC_PCER0 = (1 << ID_PIOB); // enable PMC for PIOB (ID_PIOB=12)

  // use PB27 as output (onboard LED)
  PIOB->PIO_PER  = PIO_PB27;   // use PB27 as GPIO pin
  PIOB->PIO_OER  = PIO_PB27;   // use PB27 for output direction
  PIOB->PIO_PUDR = PIO_PB27;   // disable pull-up resistor at PB27

  while (1) {
     PIOB->PIO_SODR = PIO_PB27; // output 1 at PB27
     _sw_delay( 1000000 );
     PIOB->PIO_CODR = PIO_PB27; // output 0 at PB27
     _sw_delay( 1000000 );
  }
}
```

SystemInit() is implemented in `system_sam3xa.c`

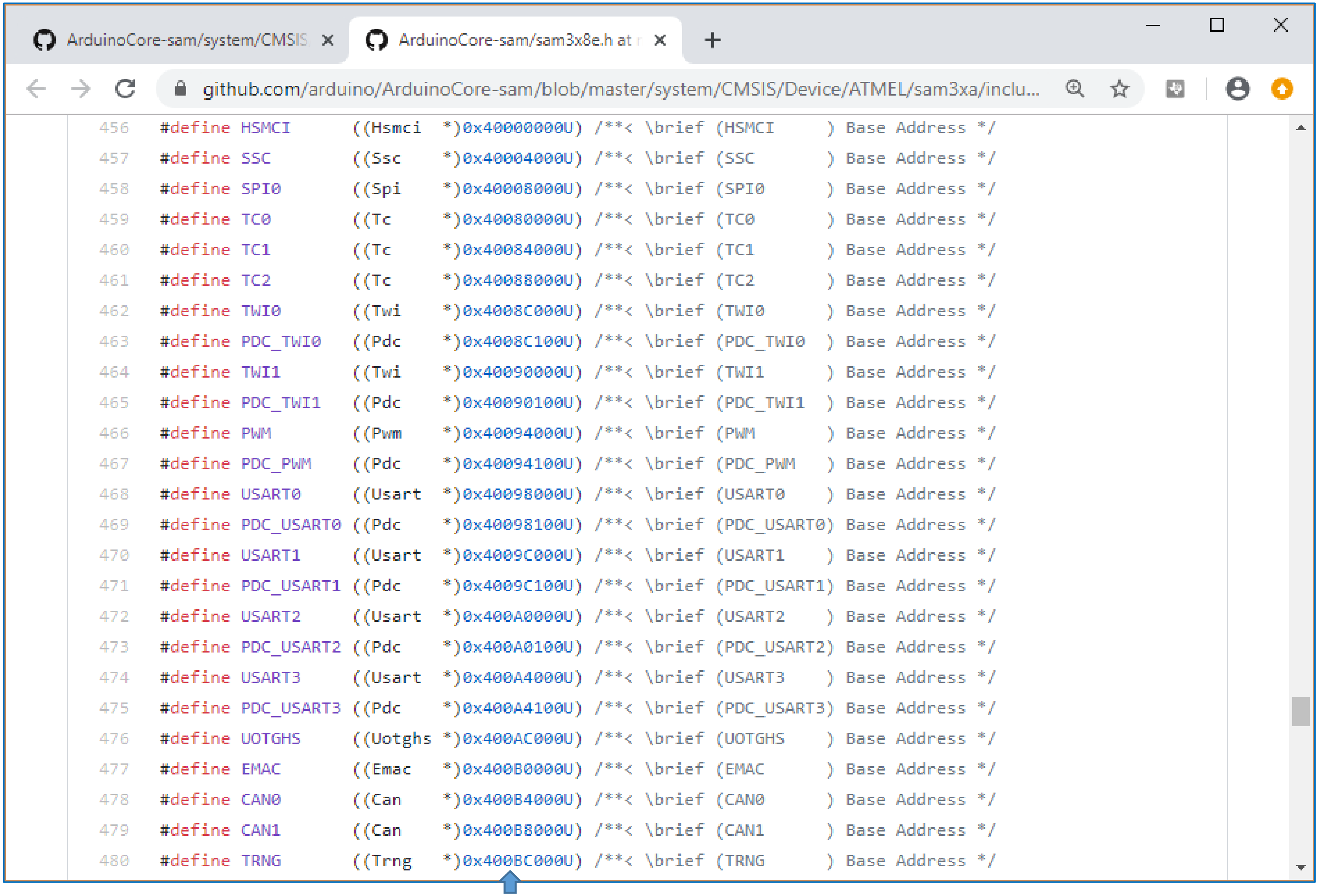
See: https://github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/source/system_sam3xa.c

Programming Arduino DUE with Arduino IDE: LED Blink

```
#include "sam.h"

#define _sw_delay(x) { for(int i=0;i<x;i++) { asm volatile("nop"); } }

int main(void) {
  SystemInit(); // initialize the system (e.g. clock freq. setting)

  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT

  PMC->PMC_PCER0 = (1 << ID_PIOB); // enable PMC for PIOB

  // use PB27 as output (onboard LED)
  PIOB->PIO_PER   = PIO_PB27;   // use PB27 as GPIO pin
  PIOB->PIO_OER   = PIO_PB27;   // use PB27 for output direction
  PIOB->PIO_PUDR  = PIO_PB27;   // disable pull-up resistor at PB27
  PIOB->PIO_OWER  = PIO_PB27;   // enable write to PIOB_ODSR for output

  while (1) {
     // toggle output at PB27
     REG_PIOB_ODSR = REG_PIOB_ODSR ^ PIO_PB27; // read-modify-write
     _sw_delay( 1000000 );
  }
}
```

SAM3X8E: Base Addresses for Registers

https://github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/sam3x8e.h

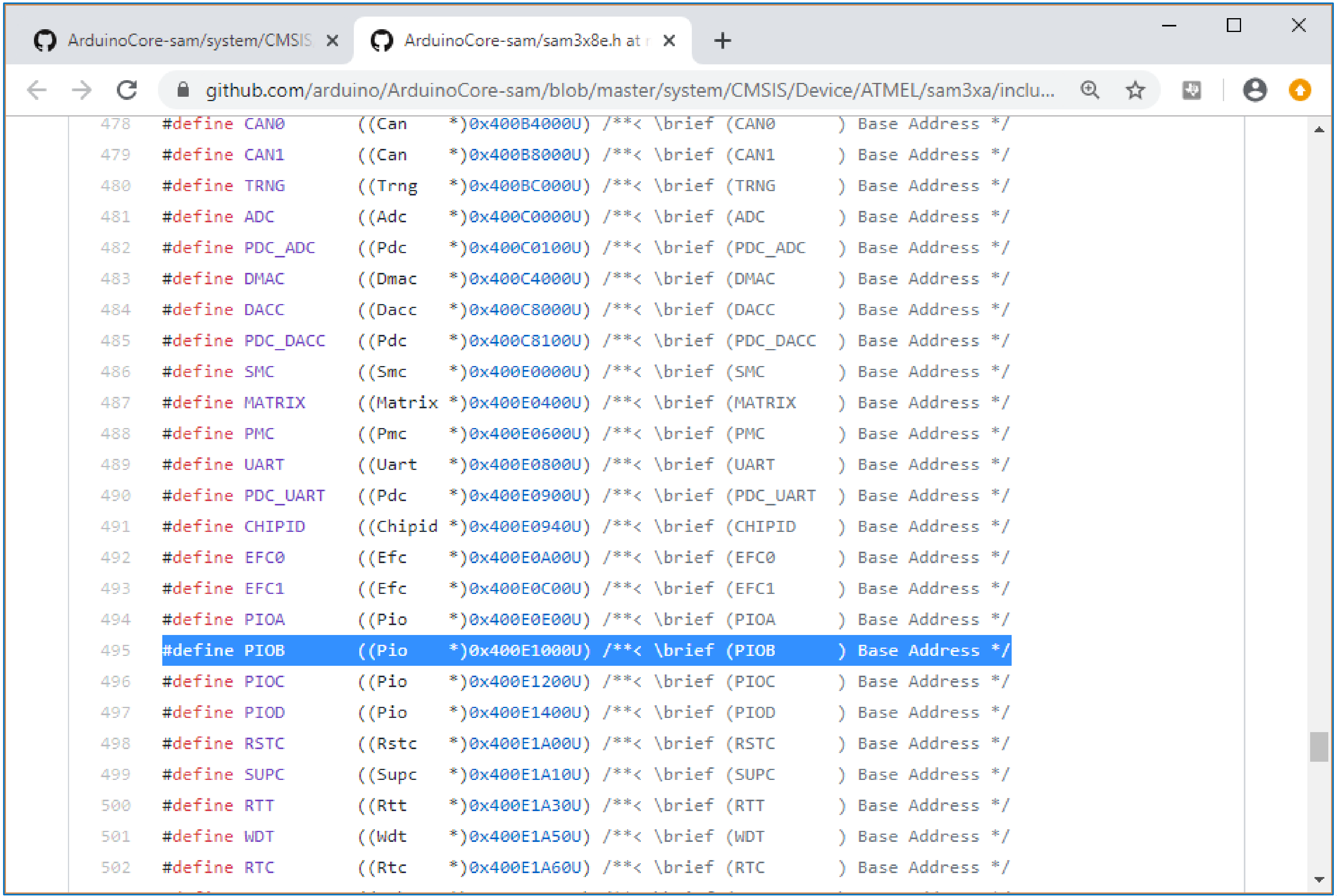

ArduinoCore-sam/system/CMSIS
ArduinoCore-sam/sam3x8e.h at r
github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/inclu...

```
#define HSMCI      ((Hsmci  *)0x40000000U) /**< \brief (HSMCI     ) Base Address */
#define SSC        ((Ssc    *)0x40004000U) /**< \brief (SSC       ) Base Address */
#define SPI0       ((Spi    *)0x40008000U) /**< \brief (SPI0      ) Base Address */
#define TC0        ((Tc     *)0x40080000U) /**< \brief (TC0       ) Base Address */
#define TC1        ((Tc     *)0x40084000U) /**< \brief (TC1       ) Base Address */
#define TC2        ((Tc     *)0x40088000U) /**< \brief (TC2       ) Base Address */
#define TWI0       ((Twi    *)0x4008C000U) /**< \brief (TWI0      ) Base Address */
#define PDC_TWI0   ((Pdc    *)0x4008C100U) /**< \brief (PDC_TWI0  ) Base Address */
#define TWI1       ((Twi    *)0x40090000U) /**< \brief (TWI1      ) Base Address */
#define PDC_TWI1   ((Pdc    *)0x40090100U) /**< \brief (PDC_TWI1  ) Base Address */
#define PWM        ((Pwm    *)0x40094000U) /**< \brief (PWM       ) Base Address */
#define PDC_PWM    ((Pdc    *)0x40094100U) /**< \brief (PDC_PWM   ) Base Address */
#define USART0     ((Usart  *)0x40098000U) /**< \brief (USART0    ) Base Address */
#define PDC_USART0 ((Pdc    *)0x40098100U) /**< \brief (PDC_USART0) Base Address */
#define USART1     ((Usart  *)0x4009C000U) /**< \brief (USART1    ) Base Address */
#define PDC_USART1 ((Pdc    *)0x4009C100U) /**< \brief (PDC_USART1) Base Address */
#define USART2     ((Usart  *)0x400A0000U) /**< \brief (USART2    ) Base Address */
#define PDC_USART2 ((Pdc    *)0x400A0100U) /**< \brief (PDC_USART2) Base Address */
#define USART3     ((Usart  *)0x400A4000U) /**< \brief (USART3    ) Base Address */
#define PDC_USART3 ((Pdc    *)0x400A4100U) /**< \brief (PDC_USART3) Base Address */
#define UOTGHS     ((Uotghs *)0x400AC000U) /**< \brief (UOTGHS    ) Base Address */
#define EMAC       ((Emac   *)0x400B0000U) /**< \brief (EMAC      ) Base Address */
#define CAN0       ((Can    *)0x400B4000U) /**< \brief (CAN0      ) Base Address */
#define CAN1       ((Can    *)0x400B8000U) /**< \brief (CAN1      ) Base Address */
#define TRNG       ((Trng   *)0x400BC000U) /**< \brief (TRNG      ) Base Address */
```

Base addresses for peripheral instances

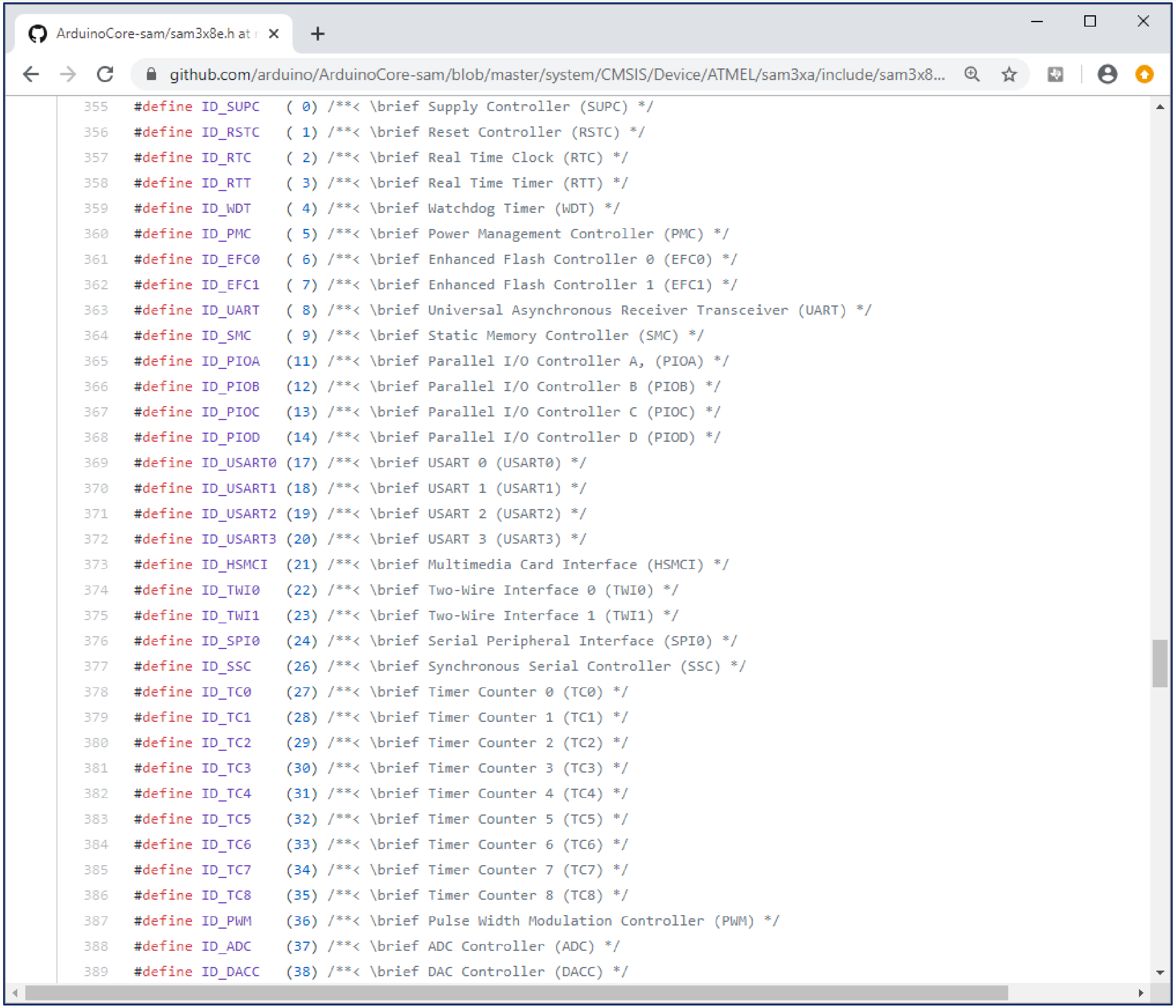
SAM3X8E: Base Addresses for Registers

https://github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/sam3x8e.h


ArduinoCore-sam/system/CMSIS
ArduinoCore-sam/sam3x8e.h at r
github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/inclu...

```
#define CAN0       ((Can    *)0x400B4000U) /**< \brief (CAN0      ) Base Address */
#define CAN1       ((Can    *)0x400B8000U) /**< \brief (CAN1      ) Base Address */
#define TRNG       ((Trng   *)0x400BC000U) /**< \brief (TRNG      ) Base Address */
#define ADC        ((Adc    *)0x400C0000U) /**< \brief (ADC       ) Base Address */
#define PDC_ADC    ((Pdc    *)0x400C0100U) /**< \brief (PDC_ADC   ) Base Address */
#define DMAC       ((Dmac   *)0x400C4000U) /**< \brief (DMAC      ) Base Address */
#define DACC       ((Dacc   *)0x400C8000U) /**< \brief (DACC      ) Base Address */
#define PDC_DACC   ((Pdc    *)0x400C8100U) /**< \brief (PDC_DACC  ) Base Address */
#define SMC        ((Smc    *)0x400E0000U) /**< \brief (SMC       ) Base Address */
#define MATRIX     ((Matrix *)0x400E0400U) /**< \brief (MATRIX    ) Base Address */
#define PMC        ((Pmc    *)0x400E0600U) /**< \brief (PMC       ) Base Address */
#define UART       ((Uart   *)0x400E0800U) /**< \brief (UART      ) Base Address */
#define PDC_UART   ((Pdc    *)0x400E0900U) /**< \brief (PDC_UART  ) Base Address */
#define CHIPID     ((Chipid *)0x400E0940U) /**< \brief (CHIPID    ) Base Address */
#define EFC0       ((Efc    *)0x400E0A00U) /**< \brief (EFC0      ) Base Address */
#define EFC1       ((Efc    *)0x400E0C00U) /**< \brief (EFC1      ) Base Address */
#define PIOA       ((Pio    *)0x400E0E00U) /**< \brief (PIOA      ) Base Address */
#define PIOB       ((Pio    *)0x400E1000U) /**< \brief (PIOB      ) Base Address */
#define PIOC       ((Pio    *)0x400E1200U) /**< \brief (PIOC      ) Base Address */
#define PIOD       ((Pio    *)0x400E1400U) /**< \brief (PIOD      ) Base Address */
#define RSTC       ((Rstc   *)0x400E1A00U) /**< \brief (RSTC      ) Base Address */
#define SUPC       ((Supc   *)0x400E1A10U) /**< \brief (SUPC      ) Base Address */
#define RTT        ((Rtt    *)0x400E1A30U) /**< \brief (RTT       ) Base Address */
#define WDT        ((Wdt    *)0x400E1A50U) /**< \brief (WDT       ) Base Address */
#define RTC        ((Rtc    *)0x400E1A60U) /**< \brief (RTC       ) Base Address */
```

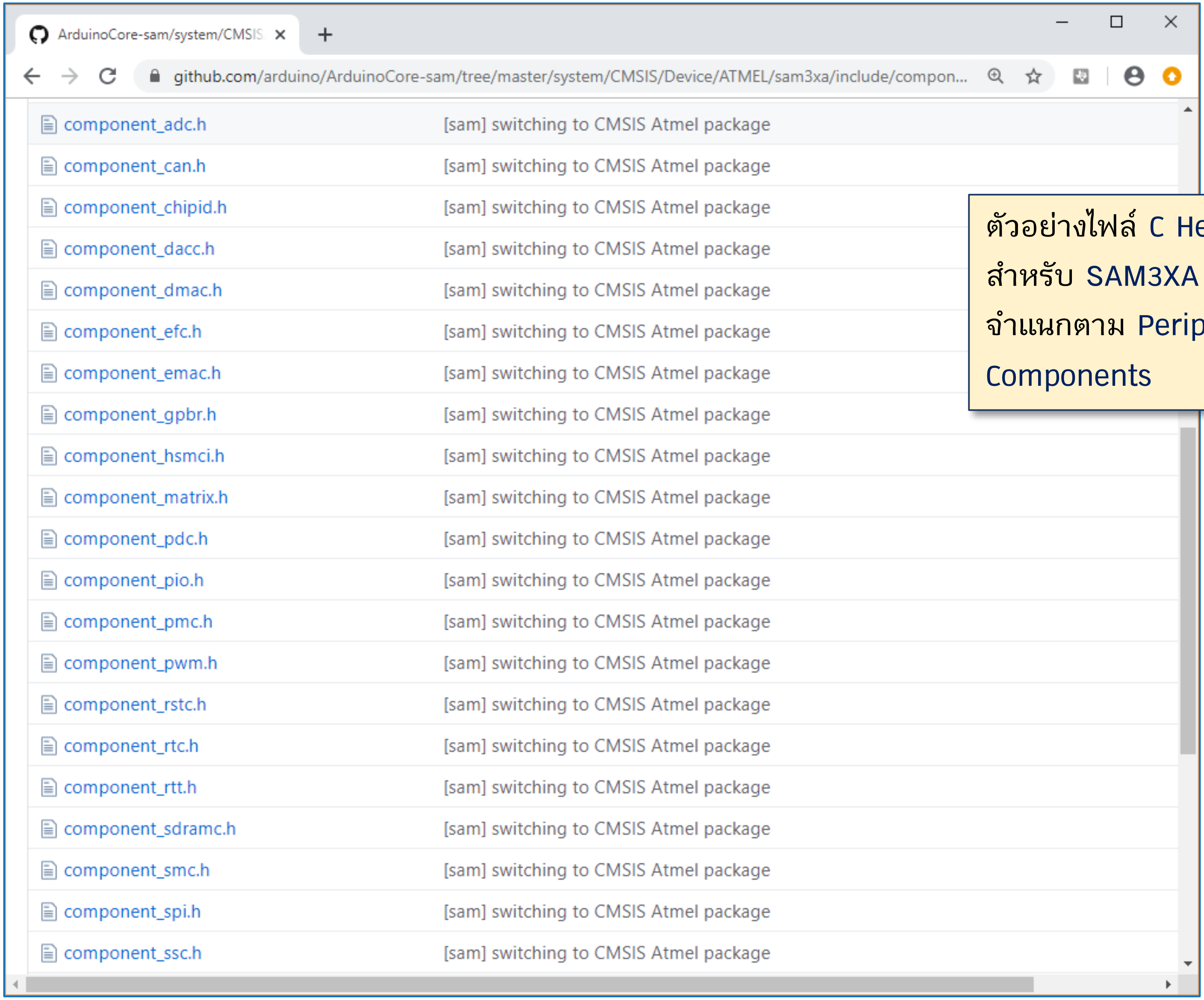
SAM3X8E: Peripheral ID Definitions


ArduinoCore-sam/sam3x8e.h at r
github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/sam3x8...

```
#define ID_SUPC   ( 0) /**< \brief Supply Controller (SUPC) */
#define ID_RSTC   ( 1) /**< \brief Reset Controller (RSTC) */
#define ID_RTC    ( 2) /**< \brief Real Time Clock (RTC) */
#define ID_RTT    ( 3) /**< \brief Real Time Timer (RTT) */
#define ID_WDT    ( 4) /**< \brief Watchdog Timer (WDT) */
#define ID_PMC    ( 5) /**< \brief Power Management Controller (PMC) */
#define ID_EFC0   ( 6) /**< \brief Enhanced Flash Controller 0 (EFC0) */
#define ID_EFC1   ( 7) /**< \brief Enhanced Flash Controller 1 (EFC1) */
#define ID_UART   ( 8) /**< \brief Universal Asynchronous Receiver Transceiver (UART) */
#define ID_SMC    ( 9) /**< \brief Static Memory Controller (SMC) */
#define ID_PIOA   (11) /**< \brief Parallel I/O Controller A, (PIOA) */
#define ID_PIOB   (12) /**< \brief Parallel I/O Controller B (PIOB) */
#define ID_PIOC   (13) /**< \brief Parallel I/O Controller C (PIOC) */
#define ID_PIOD   (14) /**< \brief Parallel I/O Controller D (PIOD) */
#define ID_USART0 (17) /**< \brief USART 0 (USART0) */
#define ID_USART1 (18) /**< \brief USART 1 (USART1) */
#define ID_USART2 (19) /**< \brief USART 2 (USART2) */
#define ID_USART3 (20) /**< \brief USART 3 (USART3) */
#define ID_HSMCI  (21) /**< \brief Multimedia Card Interface (HSMCI) */
#define ID_TWI0   (22) /**< \brief Two-Wire Interface 0 (TWI0) */
#define ID_TWI1   (23) /**< \brief Two-Wire Interface 1 (TWI1) */
#define ID_SPI0   (24) /**< \brief Serial Peripheral Interface (SPI0) */
#define ID_SSC    (26) /**< \brief Synchronous Serial Controller (SSC) */
#define ID_TC0    (27) /**< \brief Timer Counter 0 (TC0) */
#define ID_TC1    (28) /**< \brief Timer Counter 1 (TC1) */
#define ID_TC2    (29) /**< \brief Timer Counter 2 (TC2) */
#define ID_TC3    (30) /**< \brief Timer Counter 3 (TC3) */
#define ID_TC4    (31) /**< \brief Timer Counter 4 (TC4) */
#define ID_TC5    (32) /**< \brief Timer Counter 5 (TC5) */
#define ID_TC6    (33) /**< \brief Timer Counter 6 (TC6) */
#define ID_TC7    (34) /**< \brief Timer Counter 7 (TC7) */
#define ID_TC8    (35) /**< \brief Timer Counter 8 (TC8) */
#define ID_PWM    (36) /**< \brief Pulse Width Modulation Controller (PWM) */
#define ID_ADC    (37) /**< \brief ADC Controller (ADC) */
#define ID_DACC   (38) /**< \brief DAC Controller (DACC) */
```

SAM3X8E: C Header Files

https://github.com/arduino/ArduinoCore-sam/tree/master/system/CMSIS/Device/ATMEL/sam3xa/include/component

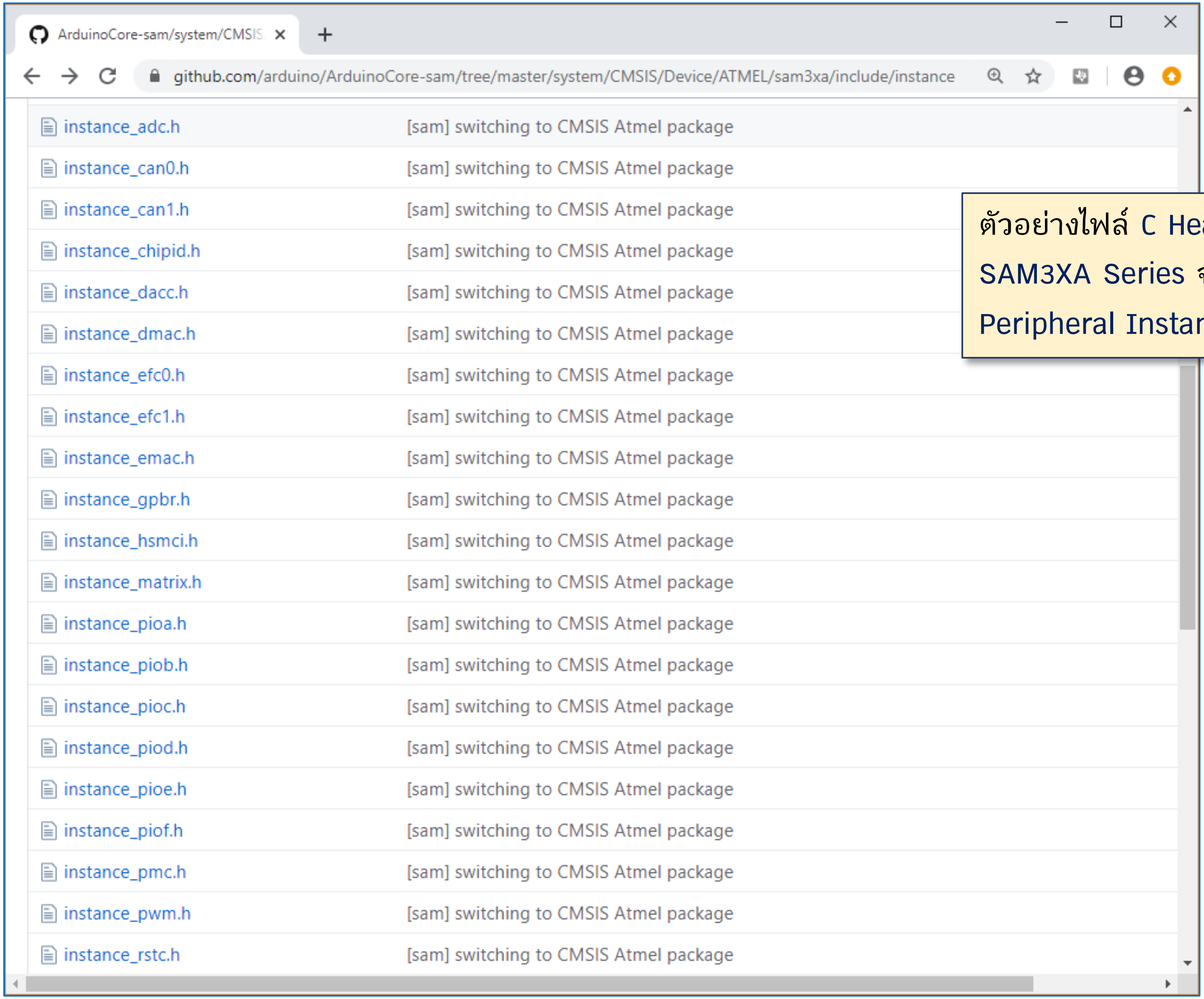

ArduinoCore-sam/system/CMSIS
github.com/arduino/ArduinoCore-sam/tree/master/system/CMSIS/Device/ATMEL/sam3xa/include/compon...

component_adc.h	[sam] switching to CMSIS Atmel package
component_can.h	[sam] switching to CMSIS Atmel package
component_chipid.h	[sam] switching to CMSIS Atmel package
component_dacc.h	[sam] switching to CMSIS Atmel package
component_dmac.h	[sam] switching to CMSIS Atmel package
component_efc.h	[sam] switching to CMSIS Atmel package
component_emac.h	[sam] switching to CMSIS Atmel package
component_gpbr.h	[sam] switching to CMSIS Atmel package
component_hsmci.h	[sam] switching to CMSIS Atmel package
component_matrix.h	[sam] switching to CMSIS Atmel package
component_pdc.h	[sam] switching to CMSIS Atmel package
component_pio.h	[sam] switching to CMSIS Atmel package
component_pmc.h	[sam] switching to CMSIS Atmel package
component_pwm.h	[sam] switching to CMSIS Atmel package
component_rstc.h	[sam] switching to CMSIS Atmel package
component_rtc.h	[sam] switching to CMSIS Atmel package
component_rtt.h	[sam] switching to CMSIS Atmel package
component_sdramc.h	[sam] switching to CMSIS Atmel package
component_smc.h	[sam] switching to CMSIS Atmel package
component_spi.h	[sam] switching to CMSIS Atmel package
component_ssc.h	[sam] switching to CMSIS Atmel package

ตัวอย่างไฟล์ C Headers สำหรับ SAM3XA Series จำแนกตาม Peripheral Components

SAM3X8E: C Header Files

https://github.com/arduino/ArduinoCore-sam/tree/master/system/CMSIS/Device/ATMEL/sam3xa/include/instance



ArduinoCore-sam/system/CMSIS

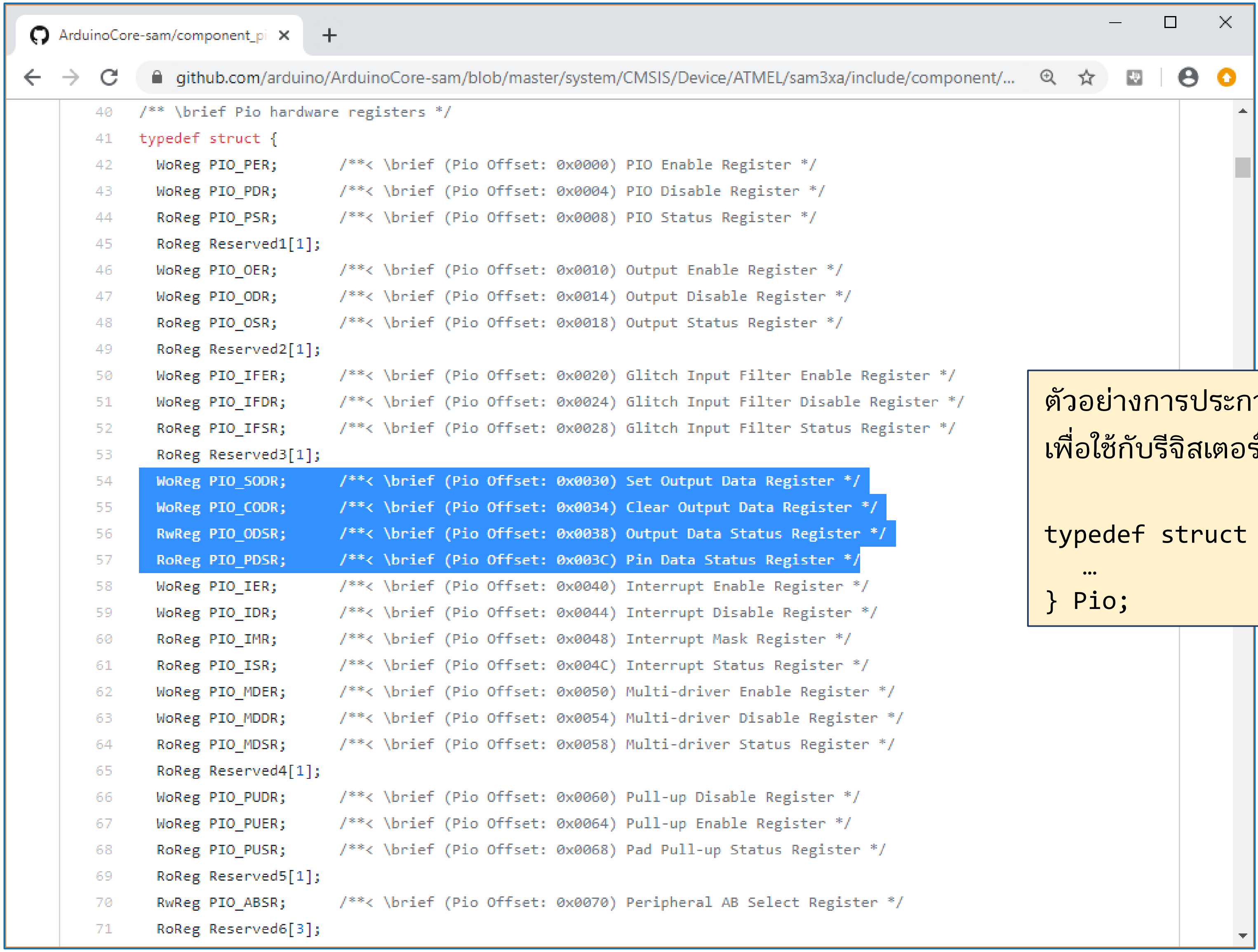
github.com/arduino/ArduinoCore-sam/tree/master/system/CMSIS/Device/ATMEL/sam3xa/include/instance

instance_adc.h	[sam] switching to CMSIS Atmel package
instance_can0.h	[sam] switching to CMSIS Atmel package
instance_can1.h	[sam] switching to CMSIS Atmel package
instance_chipid.h	[sam] switching to CMSIS Atmel package
instance_dacc.h	[sam] switching to CMSIS Atmel package
instance_dmac.h	[sam] switching to CMSIS Atmel package
instance_efc0.h	[sam] switching to CMSIS Atmel package
instance_efc1.h	[sam] switching to CMSIS Atmel package
instance_emac.h	[sam] switching to CMSIS Atmel package
instance_gpbr.h	[sam] switching to CMSIS Atmel package
instance_hsmci.h	[sam] switching to CMSIS Atmel package
instance_matrix.h	[sam] switching to CMSIS Atmel package
instance_pioa.h	[sam] switching to CMSIS Atmel package
instance_piob.h	[sam] switching to CMSIS Atmel package
instance_pioc.h	[sam] switching to CMSIS Atmel package
instance_piod.h	[sam] switching to CMSIS Atmel package
instance_pioe.h	[sam] switching to CMSIS Atmel package
instance_piof.h	[sam] switching to CMSIS Atmel package
instance_pmc.h	[sam] switching to CMSIS Atmel package
instance_pwm.h	[sam] switching to CMSIS Atmel package
instance_rstc.h	[sam] switching to CMSIS Atmel package

ตัวอย่างไฟล์ C Headers สำหรับ SAM3XA Series จำแนกตาม Peripheral Instances

SAM3X8E: C Header Files (Macro Definitions)

https://github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/component/component_pio.h


ArduinoCore-sam/component_pi
github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/component/...

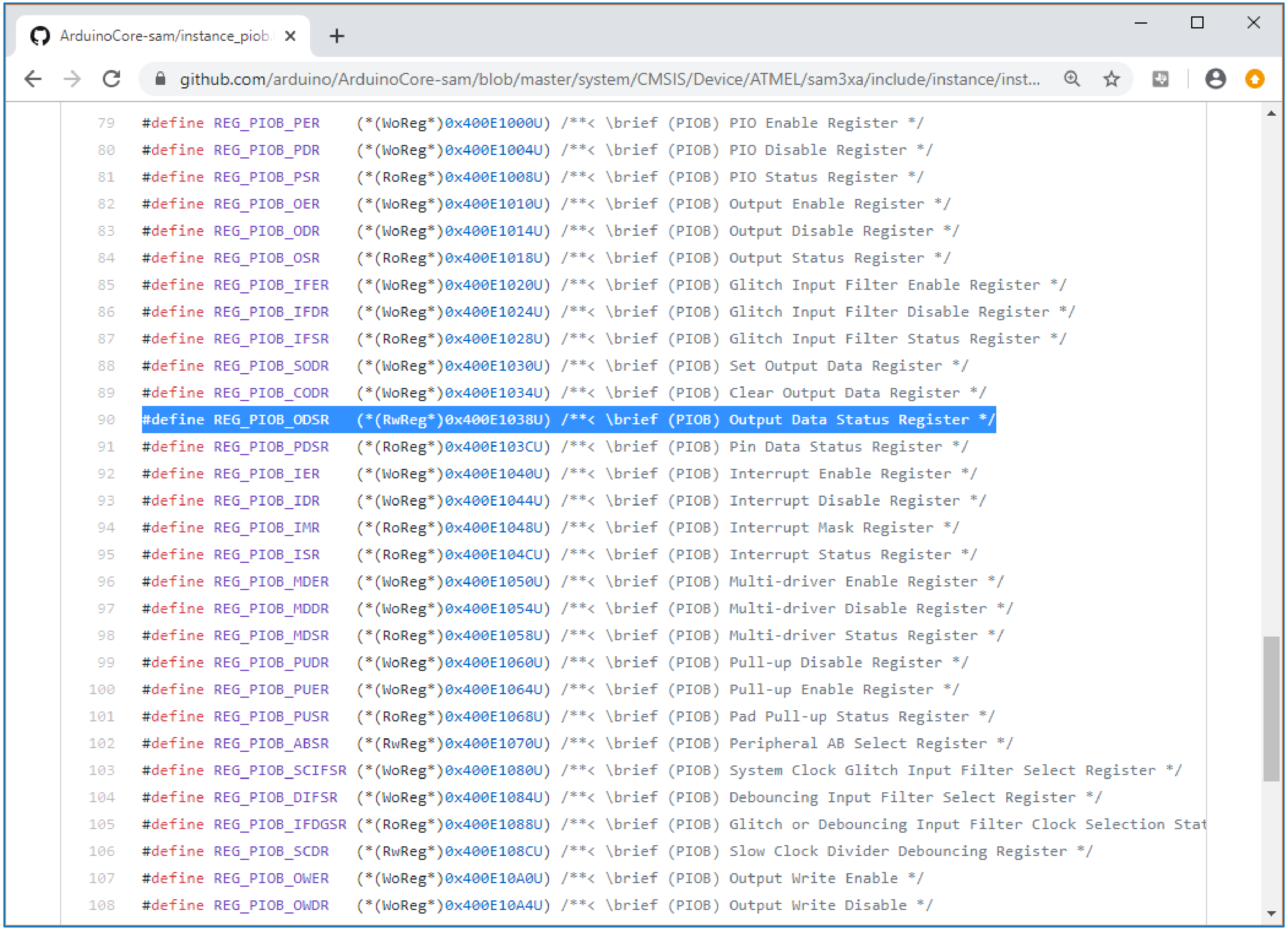
```
/** \brief Pio hardware registers */
typedef struct {
  WoReg PIO_PER;       /**< \brief (Pio Offset: 0x0000) PIO Enable Register */
  WoReg PIO_PDR;       /**< \brief (Pio Offset: 0x0004) PIO Disable Register */
  RoReg PIO_PSR;       /**< \brief (Pio Offset: 0x0008) PIO Status Register */
  RoReg Reserved1[1];
  WoReg PIO_OER;       /**< \brief (Pio Offset: 0x0010) Output Enable Register */
  WoReg PIO_ODR;       /**< \brief (Pio Offset: 0x0014) Output Disable Register */
  RoReg PIO_OSR;       /**< \brief (Pio Offset: 0x0018) Output Status Register */
  RoReg Reserved2[1];
  WoReg PIO_IFER;      /**< \brief (Pio Offset: 0x0020) Glitch Input Filter Enable Register */
  WoReg PIO_IFDR;      /**< \brief (Pio Offset: 0x0024) Glitch Input Filter Disable Register */
  RoReg PIO_IFSR;      /**< \brief (Pio Offset: 0x0028) Glitch Input Filter Status Register */
  RoReg Reserved3[1];
  WoReg PIO_SODR;      /**< \brief (Pio Offset: 0x0030) Set Output Data Register */
  WoReg PIO_CODR;      /**< \brief (Pio Offset: 0x0034) Clear Output Data Register */
  RwReg PIO_ODSR;      /**< \brief (Pio Offset: 0x0038) Output Data Status Register */
  RoReg PIO_PDSR;      /**< \brief (Pio Offset: 0x003C) Pin Data Status Register */
  WoReg PIO_IER;       /**< \brief (Pio Offset: 0x0040) Interrupt Enable Register */
  WoReg PIO_IDR;       /**< \brief (Pio Offset: 0x0044) Interrupt Disable Register */
  RoReg PIO_IMR;       /**< \brief (Pio Offset: 0x0048) Interrupt Mask Register */
  RoReg PIO_ISR;       /**< \brief (Pio Offset: 0x004C) Interrupt Status Register */
  WoReg PIO_MDER;      /**< \brief (Pio Offset: 0x0050) Multi-driver Enable Register */
  WoReg PIO_MDDR;      /**< \brief (Pio Offset: 0x0054) Multi-driver Disable Register */
  RoReg PIO_MDSR;      /**< \brief (Pio Offset: 0x0058) Multi-driver Status Register */
  RoReg Reserved4[1];
  WoReg PIO_PUDR;      /**< \brief (Pio Offset: 0x0060) Pull-up Disable Register */
  WoReg PIO_PUER;      /**< \brief (Pio Offset: 0x0064) Pull-up Enable Register */
  RoReg PIO_PUSR;      /**< \brief (Pio Offset: 0x0068) Pad Pull-up Status Register */
  RoReg Reserved5[1];
  RwReg PIO_ABSR;      /**< \brief (Pio Offset: 0x0070) Peripheral AB Select Register */
  RoReg Reserved6[3];
```

ตัวอย่างการประกาศ struct
เพื่อใช้กับรีจิสเตอร์ของ PIO

```
typedef struct {
   …
} Pio;
```

SAM3X8E: C Header Files (Macro Definitions)

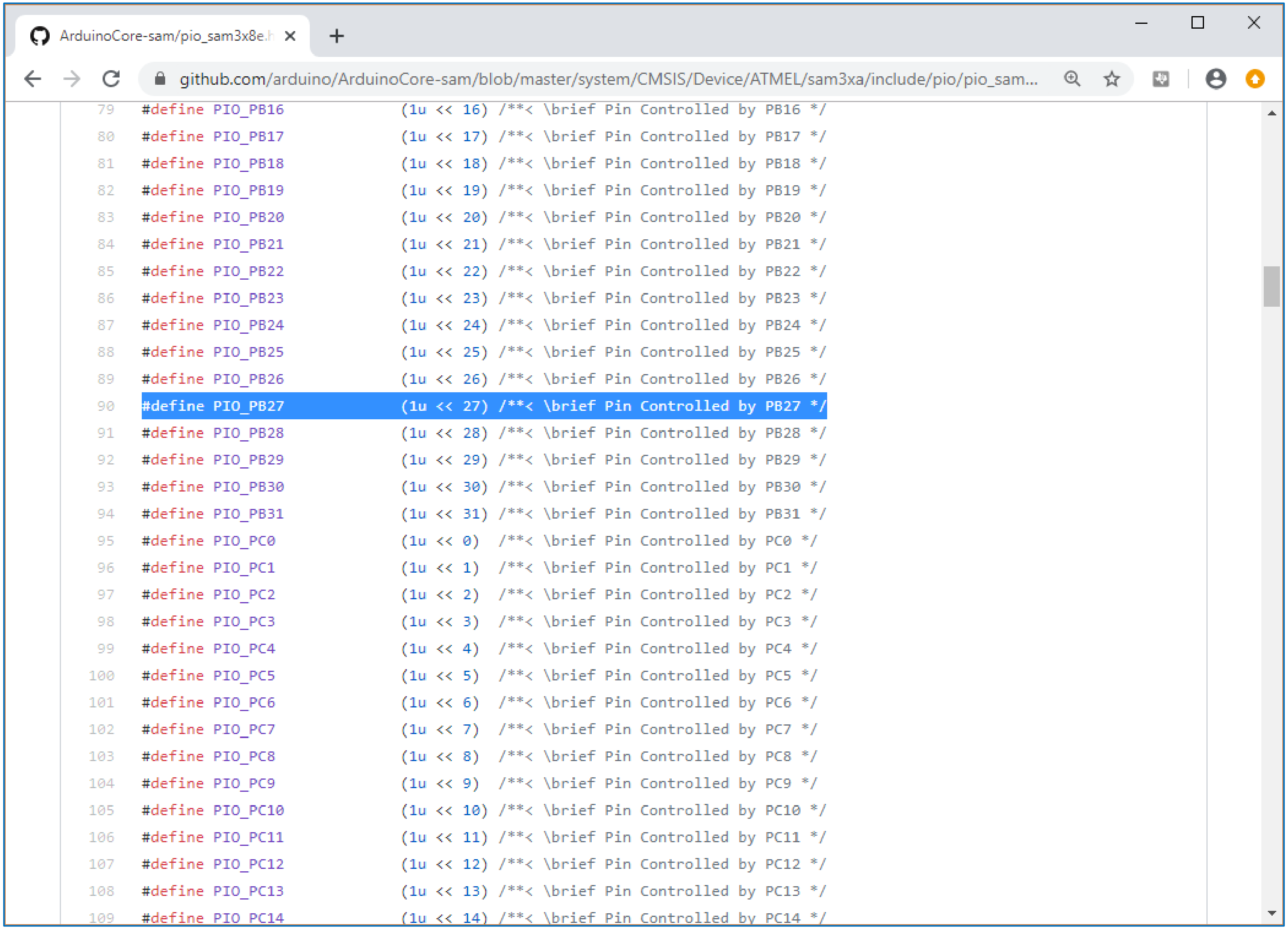
https://github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/instance/instance_piob.h


ArduinoCore-sam/instance_piob.
github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/instance/inst...

```
#define REG_PIOB_PER    (*(WoReg*)0x400E1000U) /**< \brief (PIOB) PIO Enable Register */
#define REG_PIOB_PDR    (*(WoReg*)0x400E1004U) /**< \brief (PIOB) PIO Disable Register */
#define REG_PIOB_PSR    (*(RoReg*)0x400E1008U) /**< \brief (PIOB) PIO Status Register */
#define REG_PIOB_OER    (*(WoReg*)0x400E1010U) /**< \brief (PIOB) Output Enable Register */
#define REG_PIOB_ODR    (*(WoReg*)0x400E1014U) /**< \brief (PIOB) Output Disable Register */
#define REG_PIOB_OSR    (*(RoReg*)0x400E1018U) /**< \brief (PIOB) Output Status Register */
#define REG_PIOB_IFER   (*(WoReg*)0x400E1020U) /**< \brief (PIOB) Glitch Input Filter Enable Register */
#define REG_PIOB_IFDR   (*(WoReg*)0x400E1024U) /**< \brief (PIOB) Glitch Input Filter Disable Register */
#define REG_PIOB_IFSR   (*(RoReg*)0x400E1028U) /**< \brief (PIOB) Glitch Input Filter Status Register */
#define REG_PIOB_SODR   (*(WoReg*)0x400E1030U) /**< \brief (PIOB) Set Output Data Register */
#define REG_PIOB_CODR   (*(WoReg*)0x400E1034U) /**< \brief (PIOB) Clear Output Data Register */
#define REG_PIOB_ODSR   (*(RwReg*)0x400E1038U) /**< \brief (PIOB) Output Data Status Register */
#define REG_PIOB_PDSR   (*(RoReg*)0x400E103CU) /**< \brief (PIOB) Pin Data Status Register */
#define REG_PIOB_IER    (*(WoReg*)0x400E1040U) /**< \brief (PIOB) Interrupt Enable Register */
#define REG_PIOB_IDR    (*(WoReg*)0x400E1044U) /**< \brief (PIOB) Interrupt Disable Register */
#define REG_PIOB_IMR    (*(RoReg*)0x400E1048U) /**< \brief (PIOB) Interrupt Mask Register */
#define REG_PIOB_ISR    (*(RoReg*)0x400E104CU) /**< \brief (PIOB) Interrupt Status Register */
#define REG_PIOB_MDER   (*(WoReg*)0x400E1050U) /**< \brief (PIOB) Multi-driver Enable Register */
#define REG_PIOB_MDDR   (*(WoReg*)0x400E1054U) /**< \brief (PIOB) Multi-driver Disable Register */
#define REG_PIOB_MDSR   (*(RoReg*)0x400E1058U) /**< \brief (PIOB) Multi-driver Status Register */
#define REG_PIOB_PUDR   (*(WoReg*)0x400E1060U) /**< \brief (PIOB) Pull-up Disable Register */
#define REG_PIOB_PUER   (*(WoReg*)0x400E1064U) /**< \brief (PIOB) Pull-up Enable Register */
#define REG_PIOB_PUSR   (*(RoReg*)0x400E1068U) /**< \brief (PIOB) Pad Pull-up Status Register */
#define REG_PIOB_ABSR   (*(RwReg*)0x400E1070U) /**< \brief (PIOB) Peripheral AB Select Register */
#define REG_PIOB_SCIFSR (*(WoReg*)0x400E1080U) /**< \brief (PIOB) System Clock Glitch Input Filter Select Register */
#define REG_PIOB_DIFSR  (*(WoReg*)0x400E1084U) /**< \brief (PIOB) Debouncing Input Filter Select Register */
#define REG_PIOB_IFDGSR (*(RoReg*)0x400E1088U) /**< \brief (PIOB) Glitch or Debouncing Input Filter Clock Selection Stat
#define REG_PIOB_SCDR   (*(RwReg*)0x400E108CU) /**< \brief (PIOB) Slow Clock Divider Debouncing Register */
#define REG_PIOB_OWER   (*(WoReg*)0x400E10A0U) /**< \brief (PIOB) Output Write Enable */
#define REG_PIOB_OWDR   (*(WoReg*)0x400E10A4U) /**< \brief (PIOB) Output Write Disable */
```

SAM3X8E: C Header Files (Macro Definitions)

https://github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/pio/pio_sam3x8e.h


ArduinoCore-sam/pio_sam3x8e.h
github.com/arduino/ArduinoCore-sam/blob/master/system/CMSIS/Device/ATMEL/sam3xa/include/pio/pio_sam...

```
#define PIO_PB16             (1u << 16) /**< \brief Pin Controlled by PB16 */
#define PIO_PB17             (1u << 17) /**< \brief Pin Controlled by PB17 */
#define PIO_PB18             (1u << 18) /**< \brief Pin Controlled by PB18 */
#define PIO_PB19             (1u << 19) /**< \brief Pin Controlled by PB19 */
#define PIO_PB20             (1u << 20) /**< \brief Pin Controlled by PB20 */
#define PIO_PB21             (1u << 21) /**< \brief Pin Controlled by PB21 */
#define PIO_PB22             (1u << 22) /**< \brief Pin Controlled by PB22 */
#define PIO_PB23             (1u << 23) /**< \brief Pin Controlled by PB23 */
#define PIO_PB24             (1u << 24) /**< \brief Pin Controlled by PB24 */
#define PIO_PB25             (1u << 25) /**< \brief Pin Controlled by PB25 */
#define PIO_PB26             (1u << 26) /**< \brief Pin Controlled by PB26 */
#define PIO_PB27             (1u << 27) /**< \brief Pin Controlled by PB27 */
#define PIO_PB28             (1u << 28) /**< \brief Pin Controlled by PB28 */
#define PIO_PB29             (1u << 29) /**< \brief Pin Controlled by PB29 */
#define PIO_PB30             (1u << 30) /**< \brief Pin Controlled by PB30 */
#define PIO_PB31             (1u << 31) /**< \brief Pin Controlled by PB31 */
#define PIO_PC0              (1u << 0)  /**< \brief Pin Controlled by PC0 */
#define PIO_PC1              (1u << 1)  /**< \brief Pin Controlled by PC1 */
#define PIO_PC2              (1u << 2)  /**< \brief Pin Controlled by PC2 */
#define PIO_PC3              (1u << 3)  /**< \brief Pin Controlled by PC3 */
#define PIO_PC4              (1u << 4)  /**< \brief Pin Controlled by PC4 */
#define PIO_PC5              (1u << 5)  /**< \brief Pin Controlled by PC5 */
#define PIO_PC6              (1u << 6)  /**< \brief Pin Controlled by PC6 */
#define PIO_PC7              (1u << 7)  /**< \brief Pin Controlled by PC7 */
#define PIO_PC8              (1u << 8)  /**< \brief Pin Controlled by PC8 */
#define PIO_PC9              (1u << 9)  /**< \brief Pin Controlled by PC9 */
#define PIO_PC10             (1u << 10) /**< \brief Pin Controlled by PC10 */
#define PIO_PC11             (1u << 11) /**< \brief Pin Controlled by PC11 */
#define PIO_PC12             (1u << 12) /**< \brief Pin Controlled by PC12 */
#define PIO_PC13             (1u << 13) /**< \brief Pin Controlled by PC13 */
#define PIO_PC14             (1u << 14) /**< \brief Pin Controlled by PC14 */
```

Programming Arduino DUE: Two LEDs Toggle

```
#include "sam.h"

#define _sw_delay(x) { for(int i=0;i<x;i++) { asm volatile("nop");} }

void pio_set_pin_output( Pio *pio, uint32_t pin_mask ) {
  pio->PIO_PER  = pin_mask;   // use as GPIO pin
  pio->PIO_OER  = pin_mask;   // output direction
  pio->PIO_PUDR = pin_mask;   // disable pull-up
}
void pio_set_pin_level( Pio *pio, uint32_t pin_mask, int level ) {
  if (level) {
    pio->PIO_SODR = pin_mask; // output high
  } else {
    pio->PIO_CODR = pin_mask; // output low
  }
}
int main(void) {
  SystemInit(); // initialize the system (e.g. clock freq. setting)
  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  PMC->PMC_PCER0 = (1<<ID_PIOA)|(1<<ID_PIOC); // enable CLK for PIOA & PIOC
  pio_set_pin_output( PIOA, PIO_PA21 ); // use onboard LED_TX
  pio_set_pin_output( PIOC, PIO_PC30 ); // use onboard LED_RX
  int state = 0;
  while (1) {
     pio_set_pin_level( PIOA, PIO_PA21,  state ); // update output at PA21
     pio_set_pin_level( PIOC, PIO_PC30, !state ); // update output at PC30
     state = !state; // toggle state
     _sw_delay( 10000000 );
  }
}
```

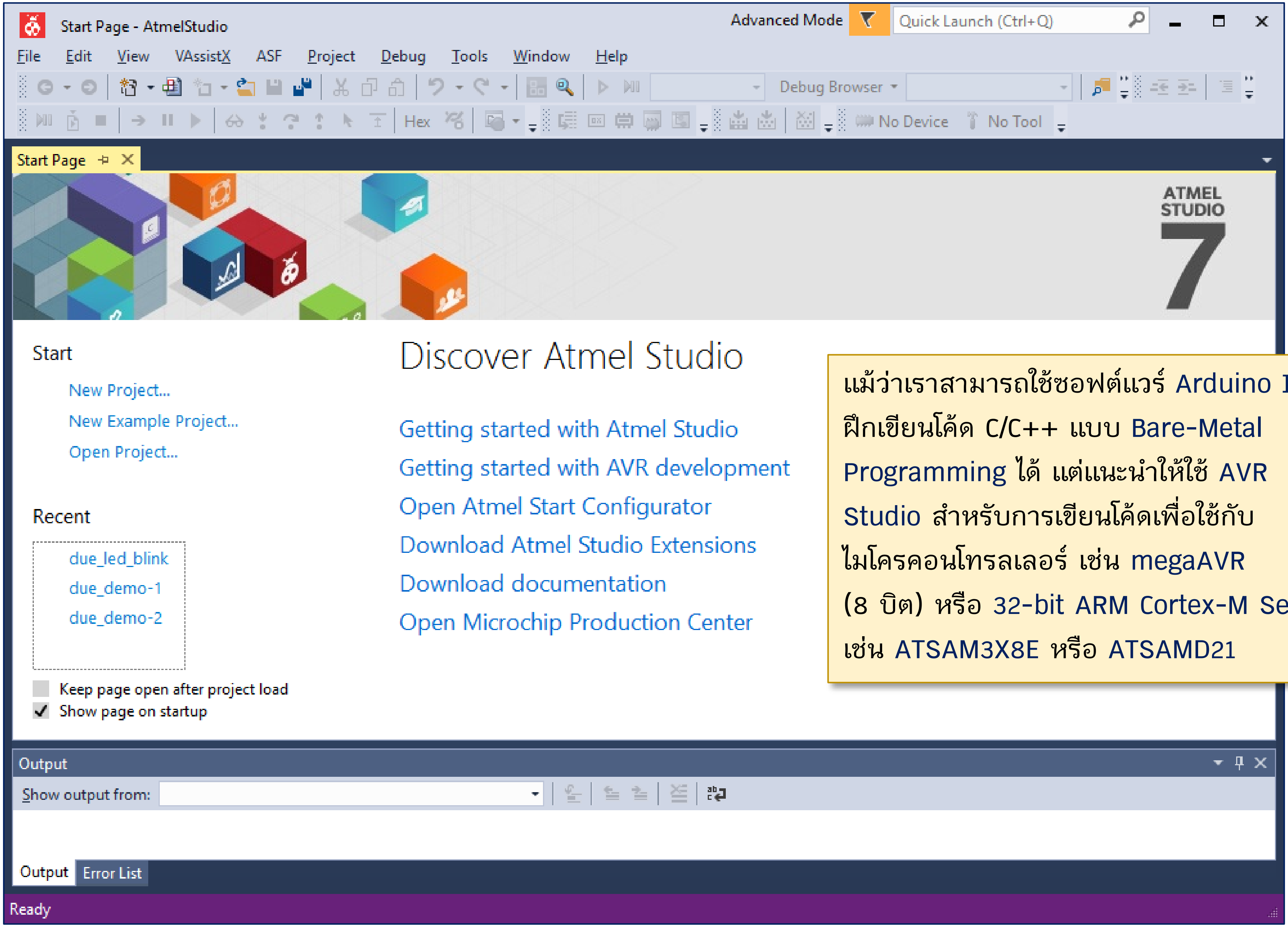
ตัวอย่างโค้ดนี้สาธิตการเขียนและสลับค่าเอาต์พุตที่ขา PA21 และ PC30 ซึ่งตรงกับ LEDs ที่อยู่บนบอร์ด Arduino DUE: LED_TX และ LED_RX ตามลำดับ

Programming Arduino DUE: LED + Push Button

```
#include "sam.h"

void pio_set_pin_level( Pio *pio, uint32_t pin_mask, int level ) {
  if (level) {
    pio->PIO_SODR = pin_mask; // output high
  } else {
    pio->PIO_CODR = pin_mask; // output low
  }
}
void init_PIO() {
  PMC->PMC_PCER0 = (1<<ID_PIOB); // enable PMC CLK for PIOB
  PIOB->PIO_PER  = (PIO_PB27 | PIO_PB25); // use as GPIO pin
  PIOB->PIO_OER  = PIO_PB27;  // output direction for PB27
  PIOB->PIO_PUDR = PIO_PB27;  // disable pull-up for PB27
  PIOB->PIO_PUER = PIO_PB25;  // enable  pull-up for PB25
}
int main(void) {
  int state;
  SystemInit(); // initialize the system (e.g. clock freq. setting)
  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  init_PIO();
  while (1) {
     state = !( PIOB->PIO_PDSR & PIO_PB25 ); // check input button
     pio_set_pin_level( PIOB, PIO_PB27, state ); // update LED output
  }
}
```

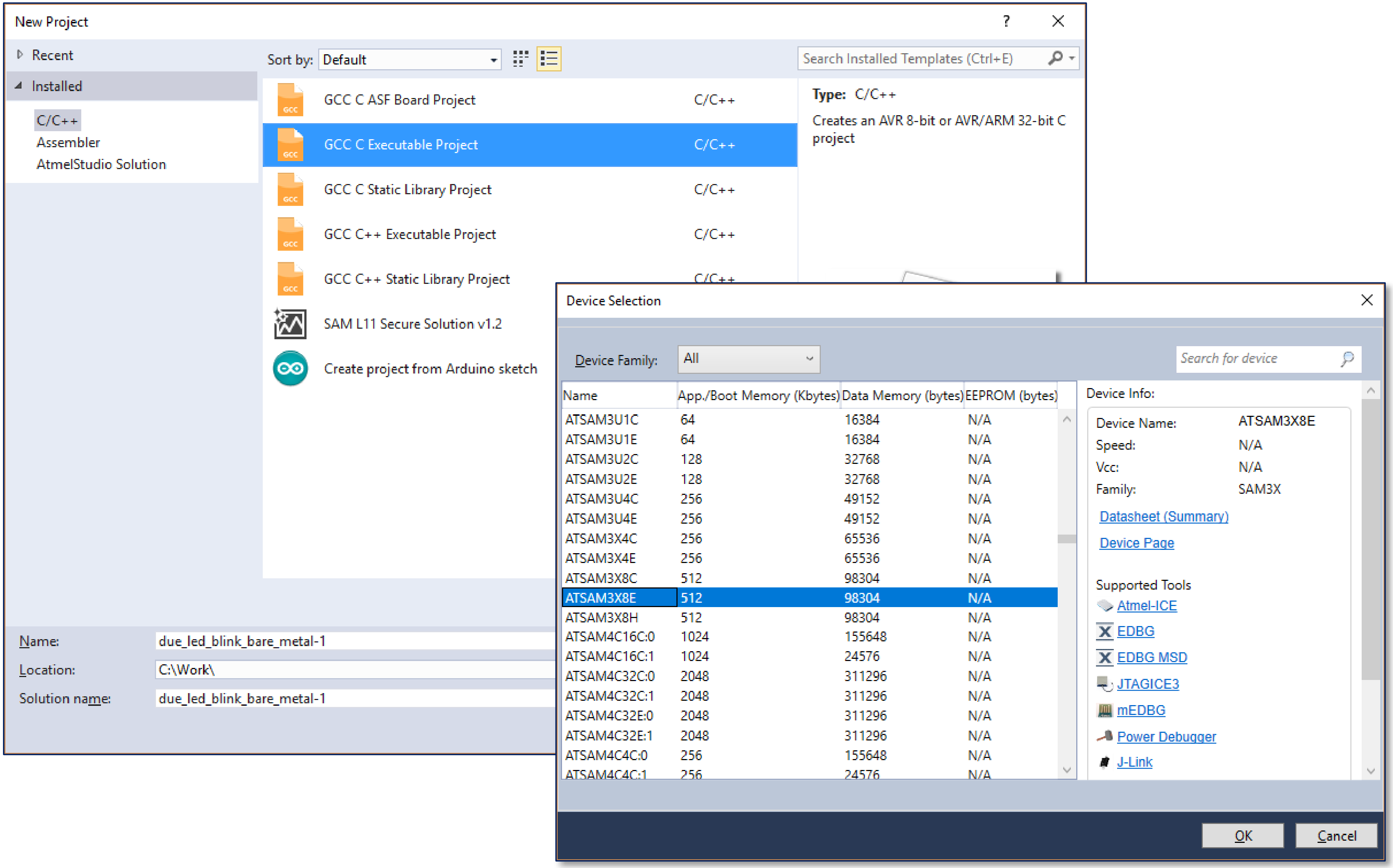
ตัวอย่างโค้ดนี้สาธิตการอ่านค่าอินพุตจากวงจรปุ่มกด (Push Button) ที่ได้นำมาต่อเพิ่มที่ขา D2 (PB25) ให้ทำงานแบบ Active-Low (ใช้ internal pull-up resistor) และนำค่าลอจิกของ I/O นี้ มาใช้กำหนดสถานะลอจิกของเอาต์พุตสำหรับ onboard LED ที่ขา D13 (PB27)

Arduino DUE Programming with AVR Studio 7



แม้ว่าเราสามารถใช้ซอฟต์แวร์ Arduino IDE ฝึกเขียนโค้ด C/C++ แบบ Bare-Metal Programming ได้ แต่แนะนำให้ใช้ AVR Studio สำหรับการเขียนโค้ดเพื่อใช้กับไมโครคอนโทรลเลอร์ เช่น megaAVR (8 บิต) หรือ 32-bit ARM Cortex-M Series เช่น ATSAM3X8E หรือ ATSAMD21

Arduino DUE: Bare-Metal Programming (no ASF)



New Project

GCC C ASF Board Project	C/C++
GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
SAM L11 Secure Solution v1.2	
Create project from Arduino sketch	

Type: C/C++

Creates an AVR 8-bit or AVR/ARM 32-bit C project

Name: due_led_blink_bare_metal-1

Location: C:\Work\

Solution name: due_led_blink_bare_metal-1

Device Selection

Device Family: All

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATSAM3U1C	64	16384	N/A
ATSAM3U1E	64	16384	N/A
ATSAM3U2C	128	32768	N/A
ATSAM3U2E	128	32768	N/A
ATSAM3U4C	256	49152	N/A
ATSAM3U4E	256	49152	N/A
ATSAM3X4C	256	65536	N/A
ATSAM3X4E	256	65536	N/A
ATSAM3X8C	512	98304	N/A
ATSAM3X8E	512	98304	N/A
ATSAM3X8H	512	98304	N/A
ATSAM4C16C:0	1024	155648	N/A
ATSAM4C16C:1	1024	24576	N/A
ATSAM4C32C:0	2048	311296	N/A
ATSAM4C32C:1	2048	311296	N/A
ATSAM4C32E:0	2048	311296	N/A
ATSAM4C32E:1	2048	311296	N/A
ATSAM4C4C:0	256	155648	N/A
ATSAM4C4C:1	256	24576	N/A

Device Info:

Device Name:	ATSAM3X8E
Speed:	N/A
Vcc:	N/A
Family:	SAM3X

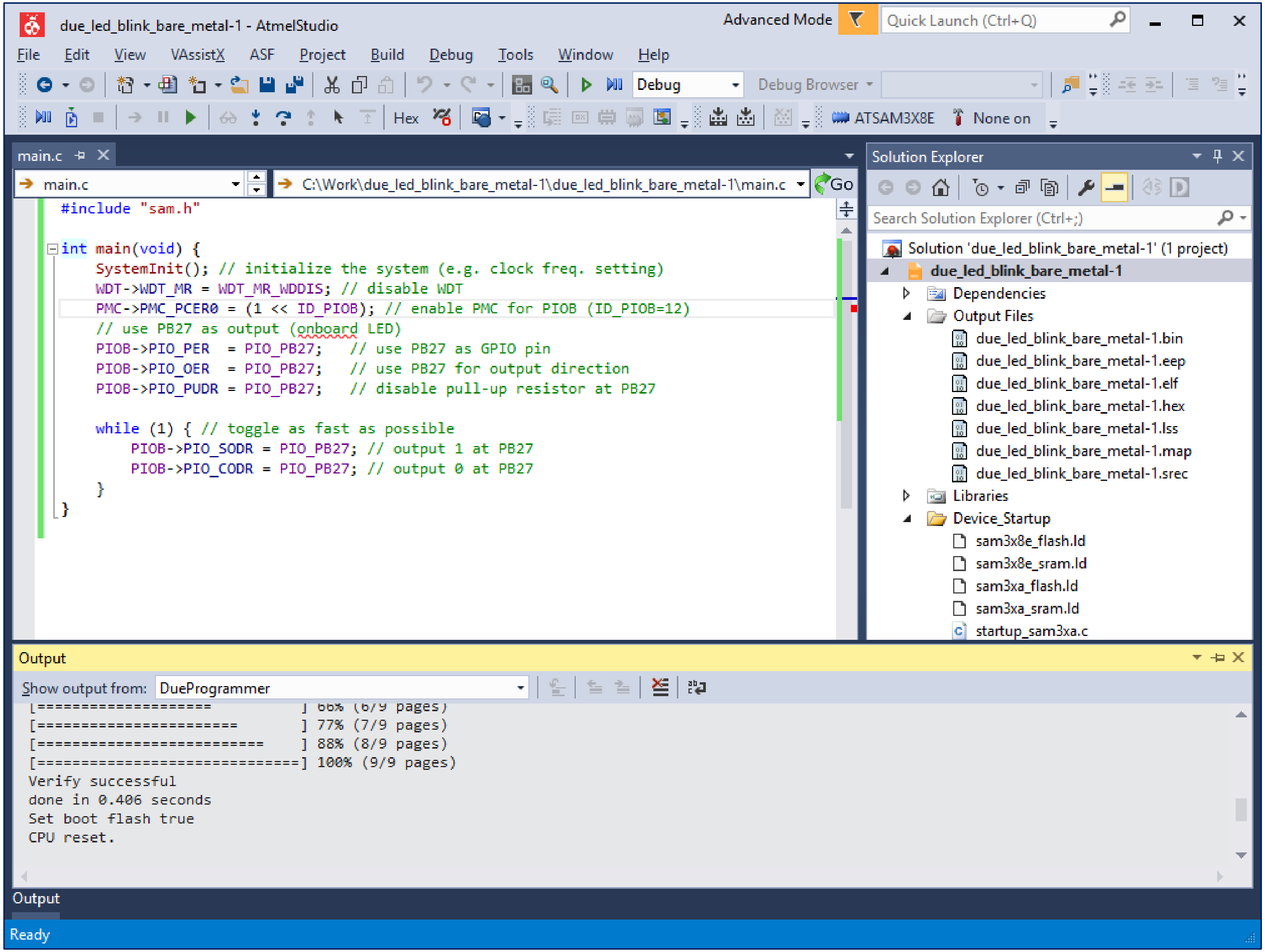
Datasheet (Summary)

Device Page

Supported Tools

- Atmel-ICE
- EDBG
- EDBG MSD
- JTAGICE3
- mEDBG
- Power Debugger
- J-Link

Arduino DUE: Bare-Metal Programming (no ASF)


due_led_blink_bare_metal-1 - AtmelStudio
Advanced Mode
Quick Launch (Ctrl+Q)
File
Edit
View
VAssistX
ASF
Project
Build
Debug
Tools
Window
Help
Debug
Debug Browser
Hex
ATSAM3X8E
None on
main.c
main.c
C:\Work\due_led_blink_bare_metal-1\due_led_blink_bare_metal-1\main.c
Go
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'due_led_blink_bare_metal-1' (1 project)
due_led_blink_bare_metal-1
Dependencies
Output Files
due_led_blink_bare_metal-1.bin
due_led_blink_bare_metal-1.eep
due_led_blink_bare_metal-1.elf
due_led_blink_bare_metal-1.hex
due_led_blink_bare_metal-1.lss
due_led_blink_bare_metal-1.map
due_led_blink_bare_metal-1.srec
Libraries
Device_Startup
sam3x8e_flash.ld
sam3x8e_sram.ld
sam3xa_flash.ld
sam3xa_sram.ld
startup_sam3xa.c
Output
Show output from:
DueProgrammer
Output
Ready

```c
#include "sam.h"

int main(void) {
    SystemInit(); // initialize the system (e.g. clock freq. setting)
    WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
    PMC->PMC_PCER0 = (1 << ID_PIOB); // enable PMC for PIOB (ID_PIOB=12)
    // use PB27 as output (onboard LED)
    PIOB->PIO_PER  = PIO_PB27;    // use PB27 as GPIO pin
    PIOB->PIO_OER  = PIO_PB27;    // use PB27 for output direction
    PIOB->PIO_PUDR = PIO_PB27;    // disable pull-up resistor at PB27

    while (1) { // toggle as fast as possible
        PIOB->PIO_SODR = PIO_PB27; // output 1 at PB27
        PIOB->PIO_CODR = PIO_PB27; // output 0 at PB27
    }
}
```

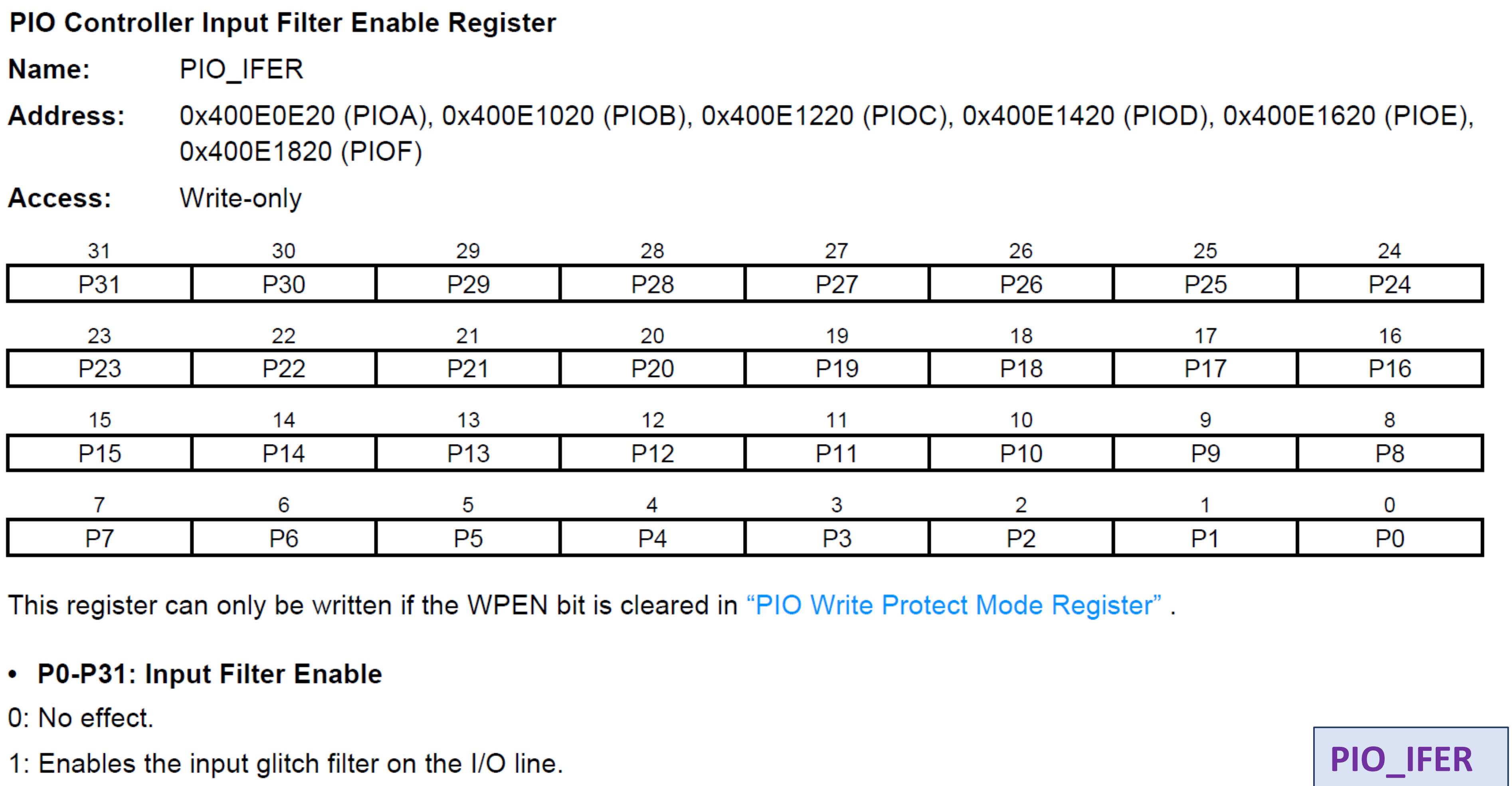
```
[====================         ] 66% (6/9 pages)
[=======================      ] 77% (7/9 pages)
[==========================   ] 88% (8/9 pages)
[=============================] 100% (9/9 pages)
Verify successful
done in 0.406 seconds
Set boot flash true
CPU reset.
```

Arduino DUE Programming: I/O with Interrupt

```
#include "sam.h"

volatile int state = 0;

void init_pio() {
  PMC->PMC_PCER0 = (1<<ID_PIOB); // enable PMC for PIOB
  PIOB->PIO_PER  = (PIO_PB27 | PIO_PB25);
  PIOB->PIO_OER  = PIO_PB27;
  PIOB->PIO_ODR  = PIO_PB25;
  PIOB->PIO_PUDR = PIO_PB27;
  PIOC->PIO_PUER = PIO_PB25;
  // select debouncing filter
  PIOB->PIO_SCDR  = 1023; // set DIV (14-bit) for slclk
  PIOB->PIO_DIFSR = PIO_PB25; // debouncing filter
  PIOB->PIO_IFER  = PIO_PB25; // enable input filtering
  // enable interrupt for PIOB
  NVIC_EnableIRQ( PIOB_IRQn );
  PIOB->PIO_AIMER = PIO_PB25; // use additional mode
  PIOB->PIO_ESR = PIO_PB25;   // select edge mode
  PIOB->PIO_FELLSR = PIO_PB25; // select falling edge
  PIOB->PIO_IER = PIO_PB25;  // enable interrupt on PB25
}
```

```
void update_led( int _state ) {
  if (_state) {
    PIOB->PIO_SODR = PIO_PB27;
  } else {
    PIOB->PIO_CODR = PIO_PB27;
  }
}

void PIOB_Handler(void) {
  if ( (PIOB->PIO_ISR & PIO_PB25)==PIO_PB25 ) {
    state = !state;
    update_led( state );
  }
}

int main(void){
  SystemInit(); // initialize the system
  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  init_pio();
  update_led( 0 );
  while (1) {}
}
```

SAM3x8E PIO Register: PIO_IFER



PIO Controller Input Filter Enable Register

Name: PIO_IFER

Address: 0x400E0E20 (PIOA), 0x400E1020 (PIOB), 0x400E1220 (PIOC), 0x400E1420 (PIOD), 0x400E1620 (PIOE), 0x400E1820 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

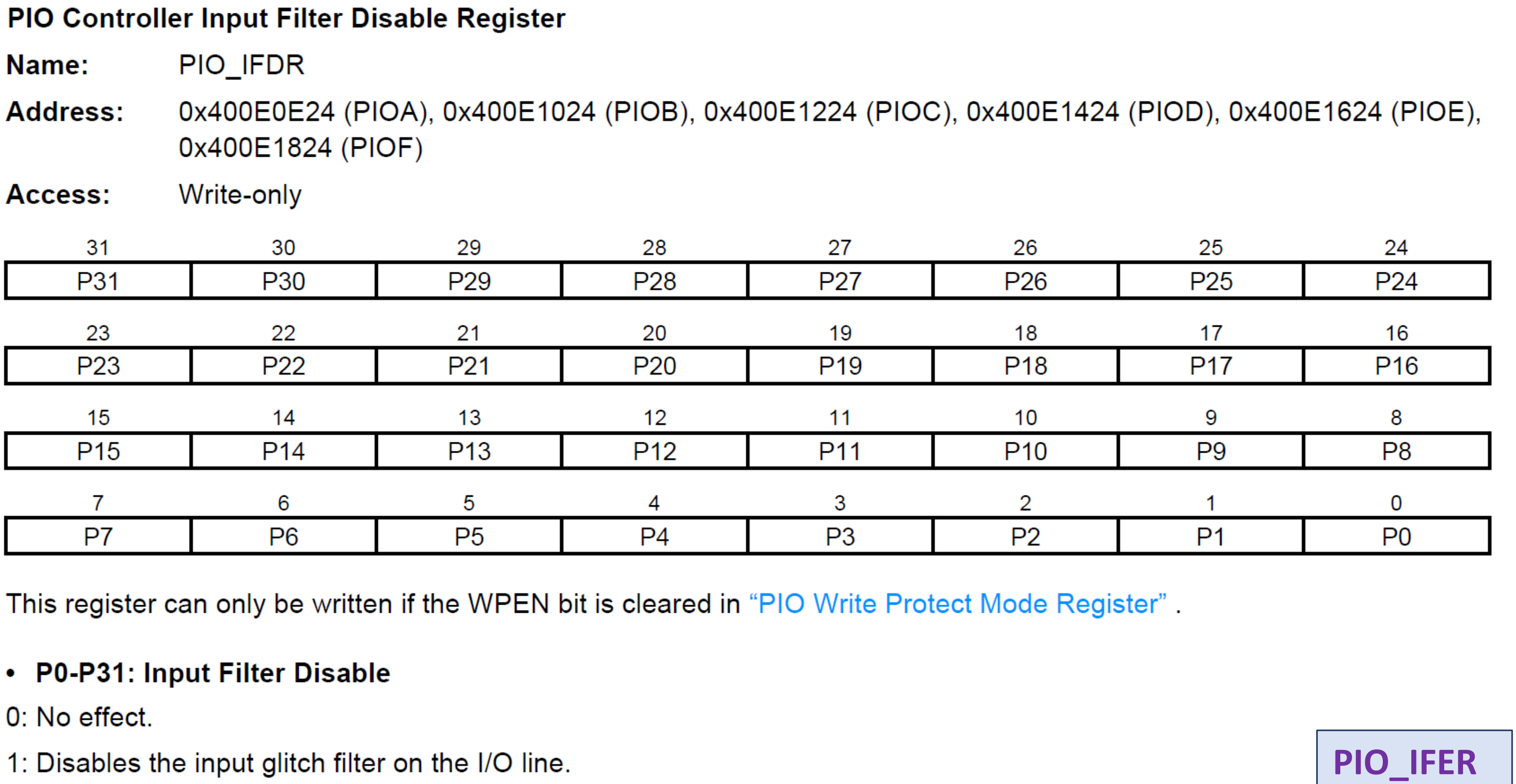
- **P0-P31: Input Filter Enable**

0: No effect.

1: Enables the input glitch filter on the I/O line.

PIO_IFER
PIO_IFDR
PIO_IFSR

SAM3x8E PIO Register: PIO_IFDR



PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Address: 0x400E0E24 (PIOA), 0x400E1024 (PIOB), 0x400E1224 (PIOC), 0x400E1424 (PIOD), 0x400E1624 (PIOE), 0x400E1824 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

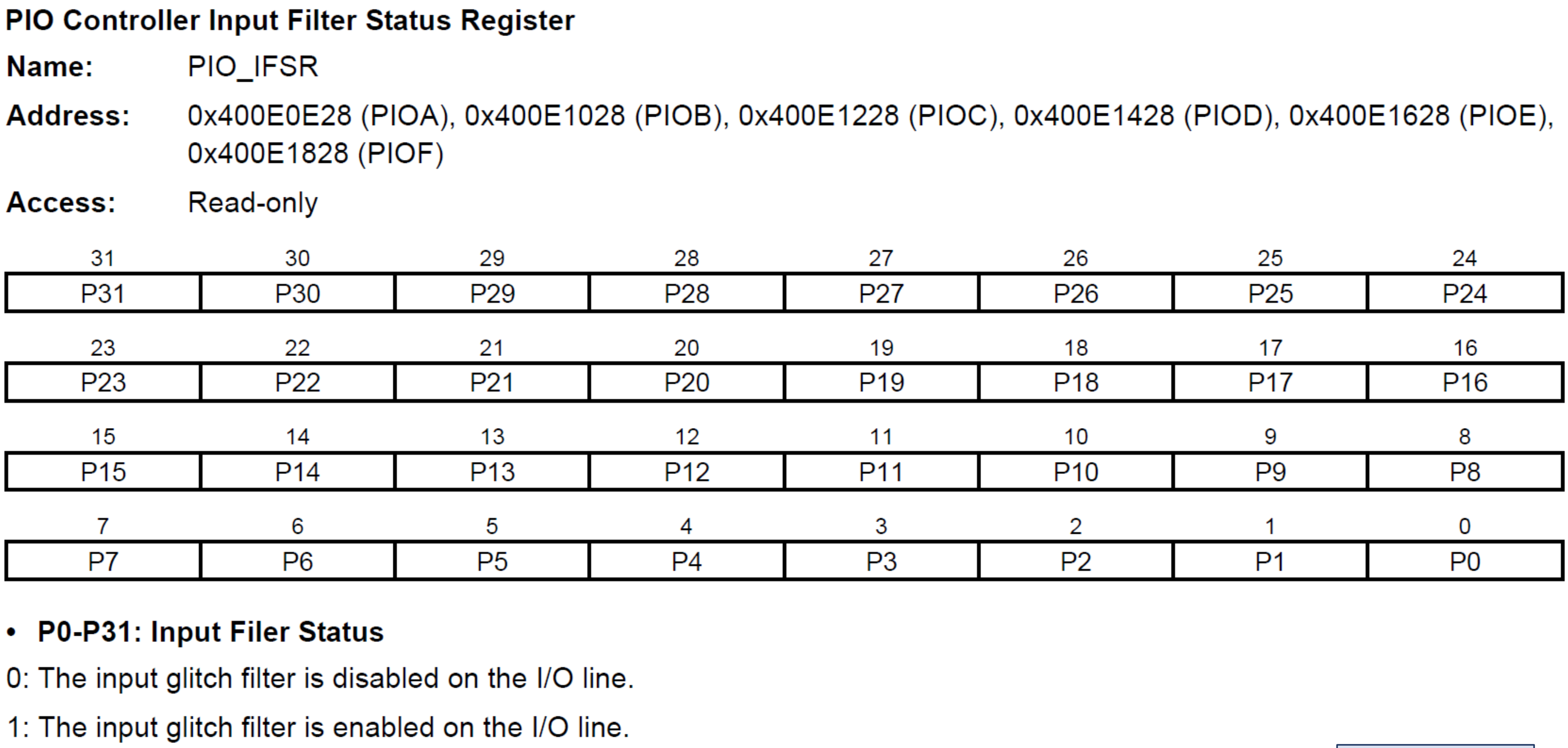
- **P0-P31: Input Filter Disable**

0: No effect.

1: Disables the input glitch filter on the I/O line.

PIO_IFER
PIO_IFDR
PIO_IFSR

SAM3x8E PIO Register: PIO_IFSR



PIO Controller Input Filter Status Register

Name: PIO_IFSR

Address: 0x400E0E28 (PIOA), 0x400E1028 (PIOB), 0x400E1228 (PIOC), 0x400E1428 (PIOD), 0x400E1628 (PIOE), 0x400E1828 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

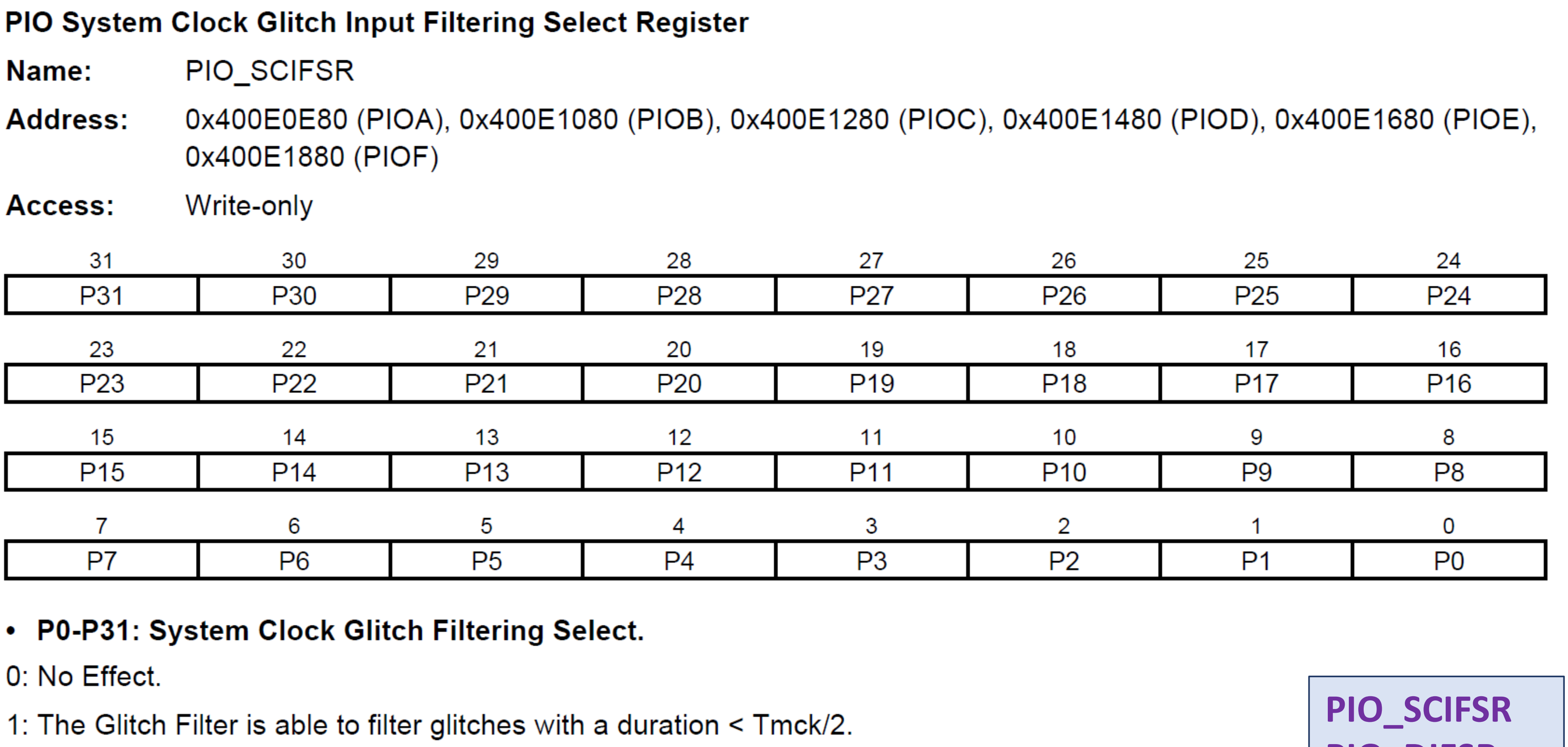
- **P0-P31: Input Filer Status**

0: The input glitch filter is disabled on the I/O line.

1: The input glitch filter is enabled on the I/O line.

PIO_IFER
PIO_IFDR
PIO_IFSR

SAM3x8E PIO Register: PIO_SCIFSR



PIO System Clock Glitch Input Filtering Select Register

Name: PIO_SCIFSR

Address: 0x400E0E80 (PIOA), 0x400E1080 (PIOB), 0x400E1280 (PIOC), 0x400E1480 (PIOD), 0x400E1680 (PIOE), 0x400E1880 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

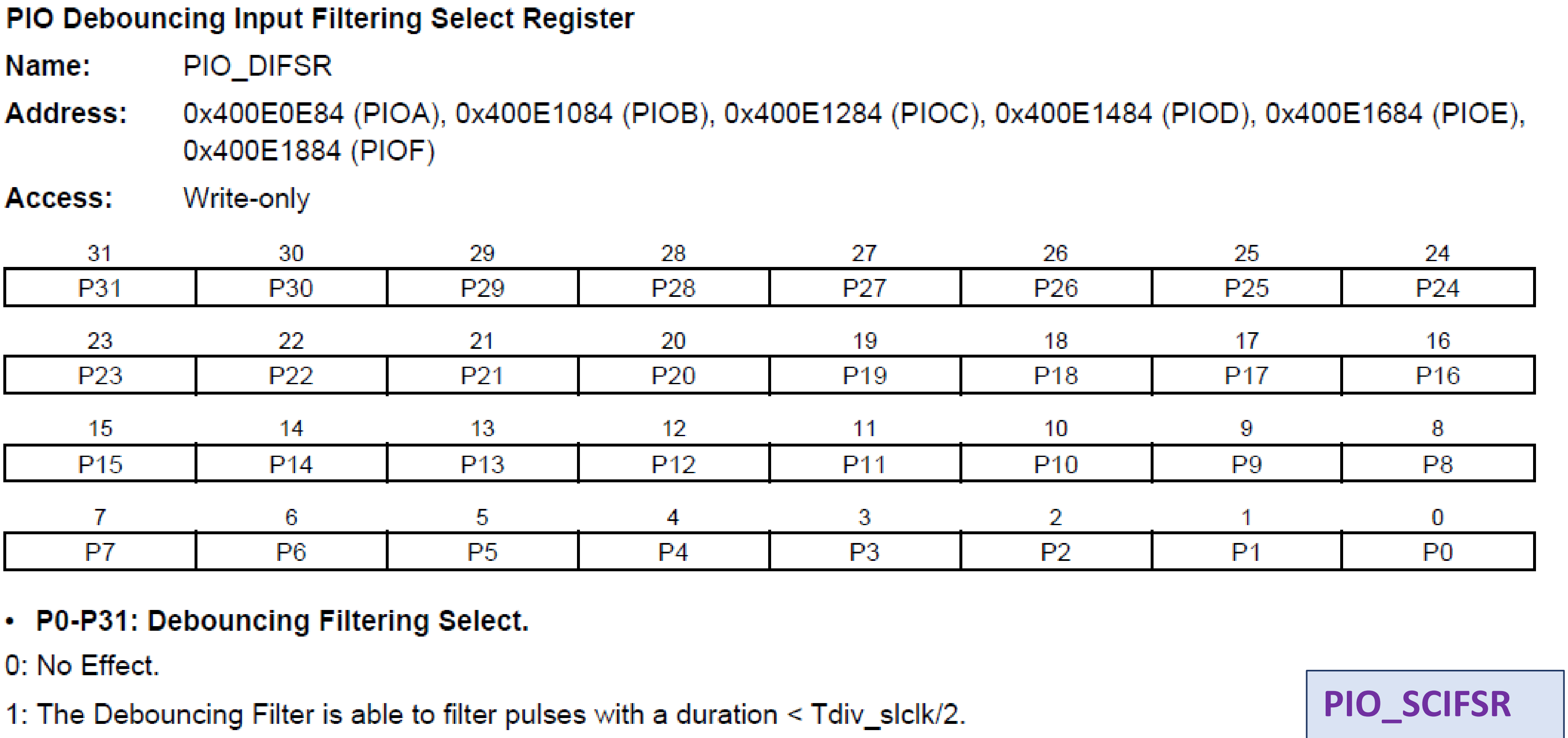
- **P0-P31: System Clock Glitch Filtering Select.**

0: No Effect.

1: The Glitch Filter is able to filter glitches with a duration < Tmck/2.

PIO_SCIFSR
PIO_DIFSR
PIO_IFDGSR
PIO_SCDR

SAM3x8E PIO Register: PIO_DIFSR



PIO Debouncing Input Filtering Select Register

Name: PIO_DIFSR

Address: 0x400E0E84 (PIOA), 0x400E1084 (PIOB), 0x400E1284 (PIOC), 0x400E1484 (PIOD), 0x400E1684 (PIOE), 0x400E1884 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

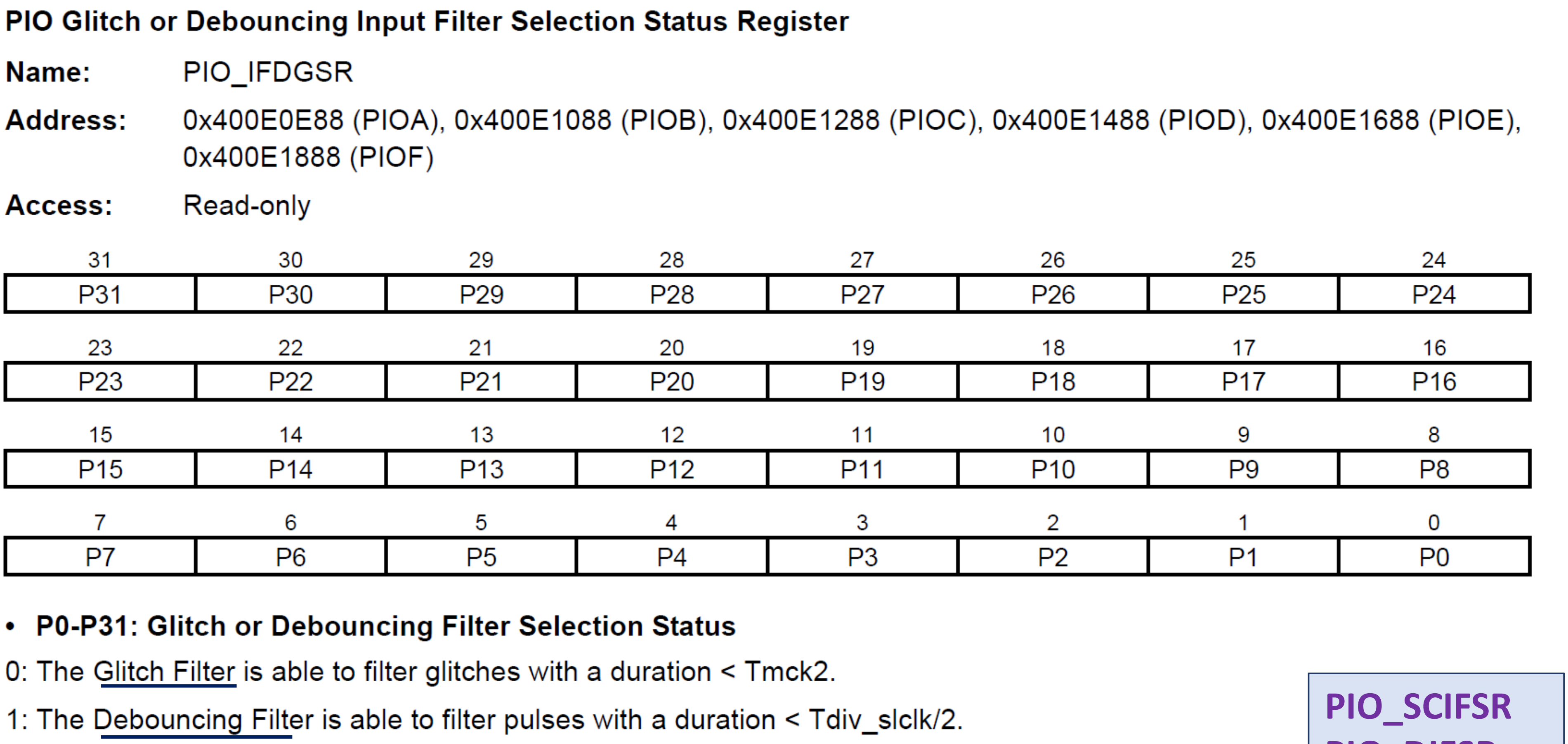
- **P0-P31: Debouncing Filtering Select.**

0: No Effect.

1: The Debouncing Filter is able to filter pulses with a duration < Tdiv_slclk/2.

PIO_SCIFSR
PIO_DIFSR
PIO_IFDGSR
PIO_SCDR

SAM3x8E PIO Register: PIO_IFDGSR



PIO Glitch or Debouncing Input Filter Selection Status Register

Name: PIO_IFDGSR

Address: 0x400E0E88 (PIOA), 0x400E1088 (PIOB), 0x400E1288 (PIOC), 0x400E1488 (PIOD), 0x400E1688 (PIOE), 0x400E1888 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

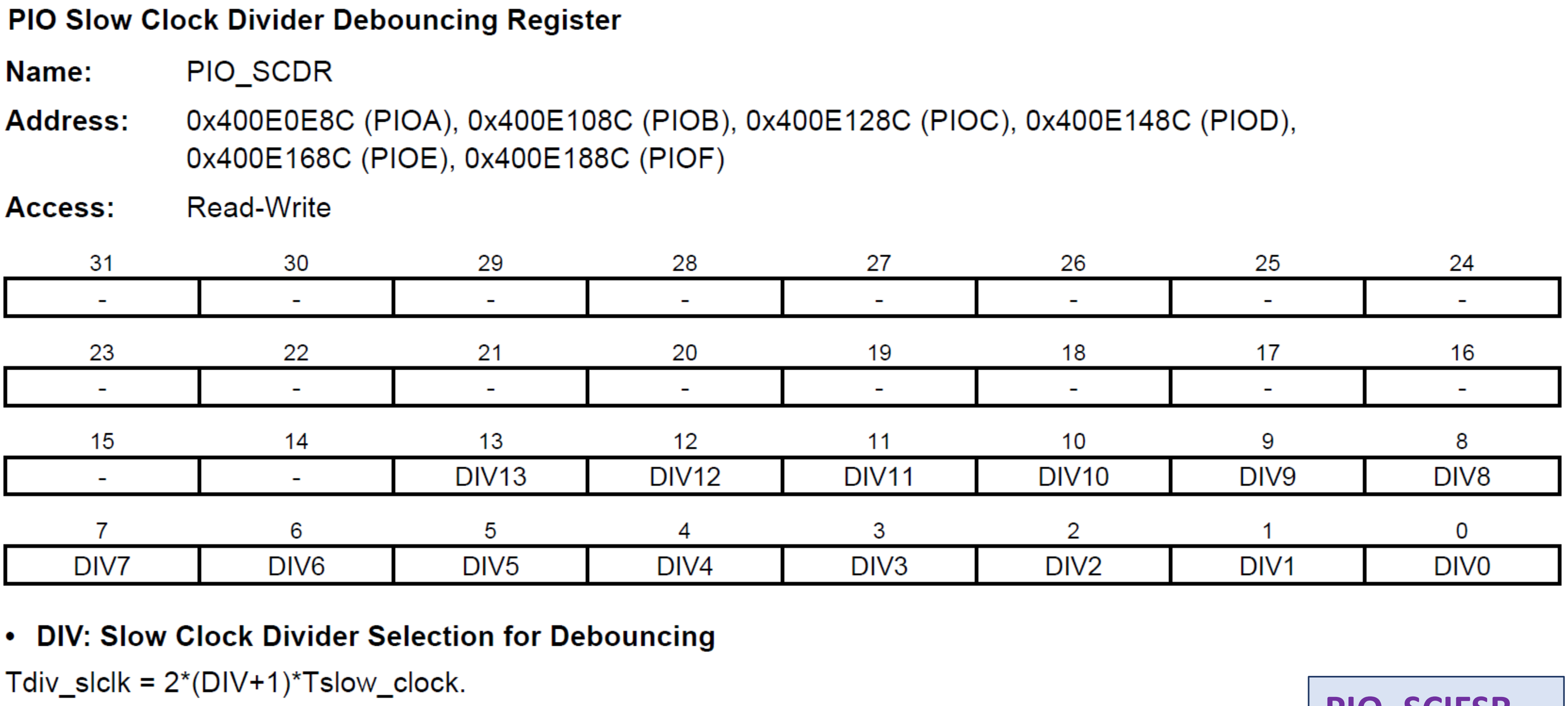
- **P0-P31: Glitch or Debouncing Filter Selection Status**

0: The Glitch Filter is able to filter glitches with a duration < Tmck2.

1: The Debouncing Filter is able to filter pulses with a duration < Tdiv_slclk/2.

PIO_SCIFSR
PIO_DIFSR
PIO_IFDGSR
PIO_SCDR

SAM3x8E PIO Register: PIO_SCDR



PIO Slow Clock Divider Debouncing Register

Name: PIO_SCDR

Address: 0x400E0E8C (PIOA), 0x400E108C (PIOB), 0x400E128C (PIOC), 0x400E148C (PIOD),
0x400E168C (PIOE), 0x400E188C (PIOF)

Access: Read-Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	DIV13	DIV12	DIV11	DIV10	DIV9	DIV8

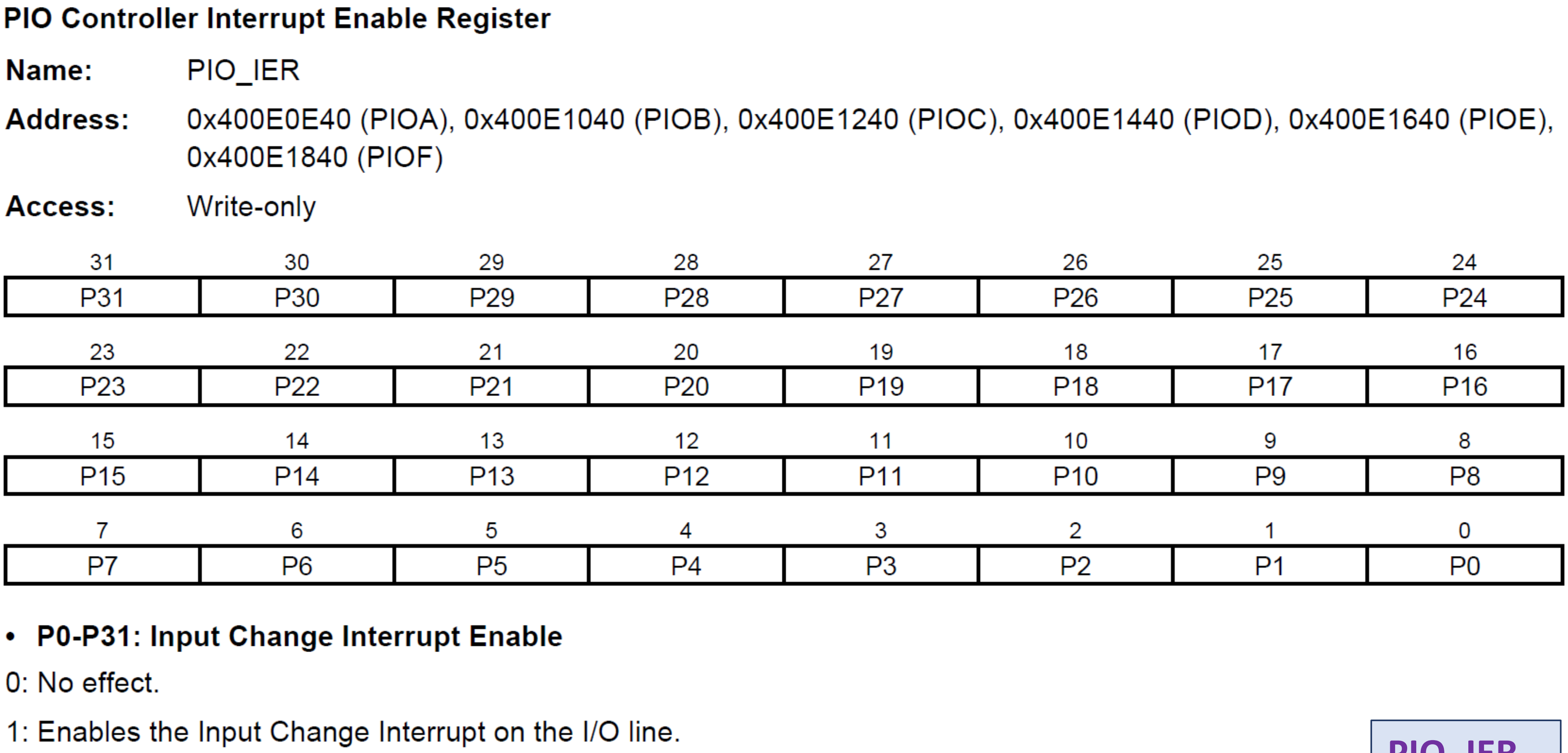
7	6	5	4	3	2	1	0
DIV7	DIV6	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0

- **DIV: Slow Clock Divider Selection for Debouncing**

Tdiv_slclk = 2*(DIV+1)*Tslow_clock.

PIO_SCIFSR
PIO_DIFSR
PIO_IFDGSR
PIO_SCDR

SAM3x8E PIO Register: PIO_IER



PIO Controller Interrupt Enable Register

Name: PIO_IER

Address: 0x400E0E40 (PIOA), 0x400E1040 (PIOB), 0x400E1240 (PIOC), 0x400E1440 (PIOD), 0x400E1640 (PIOE), 0x400E1840 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

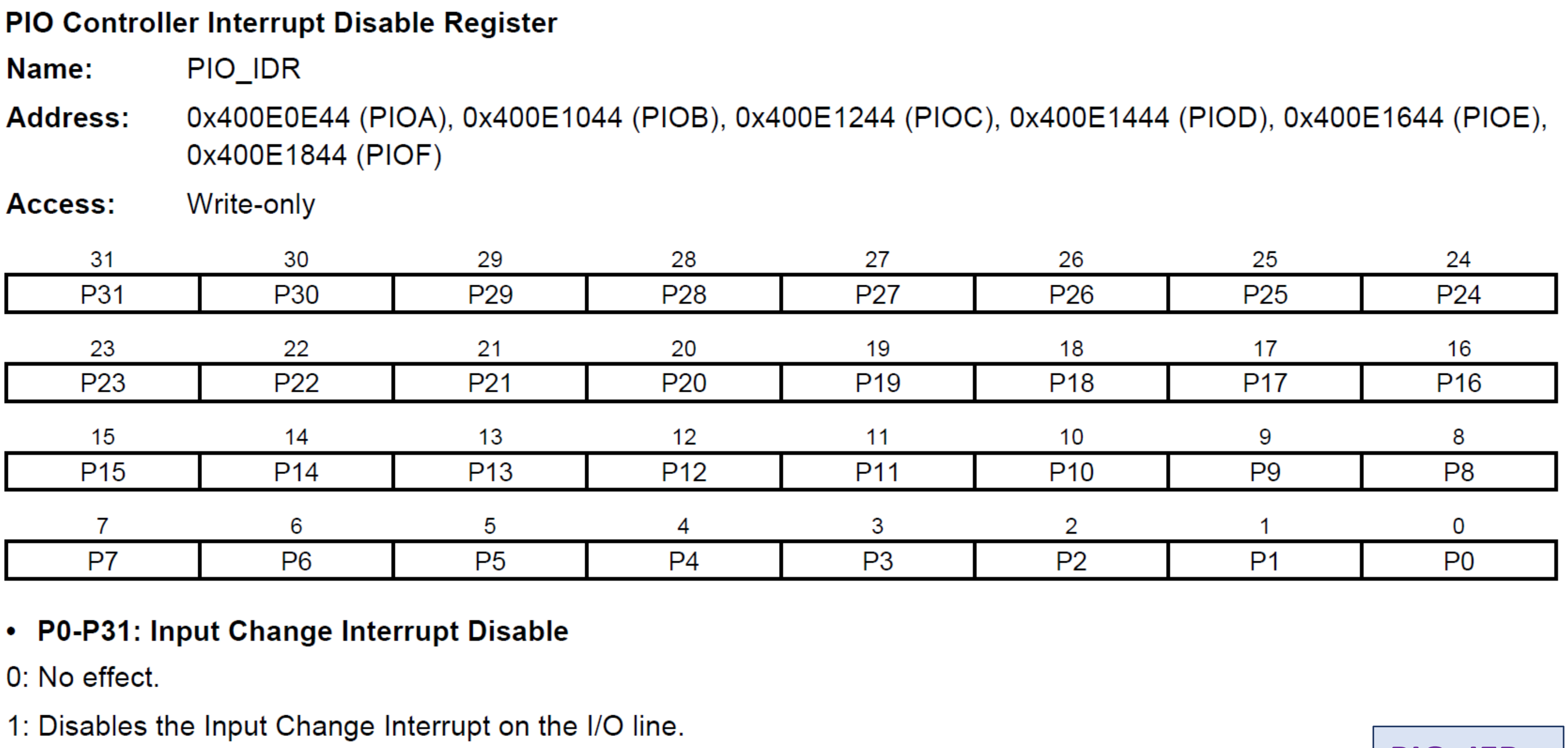
- **P0-P31: Input Change Interrupt Enable**

0: No effect.

1: Enables the Input Change Interrupt on the I/O line.

PIO_IER
PIO_IDR
PIO_IMR
PIO_ISR

SAM3x8E PIO Register: PIO_IDR



PIO Controller Interrupt Disable Register

Name: PIO_IDR

Address: 0x400E0E44 (PIOA), 0x400E1044 (PIOB), 0x400E1244 (PIOC), 0x400E1444 (PIOD), 0x400E1644 (PIOE), 0x400E1844 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

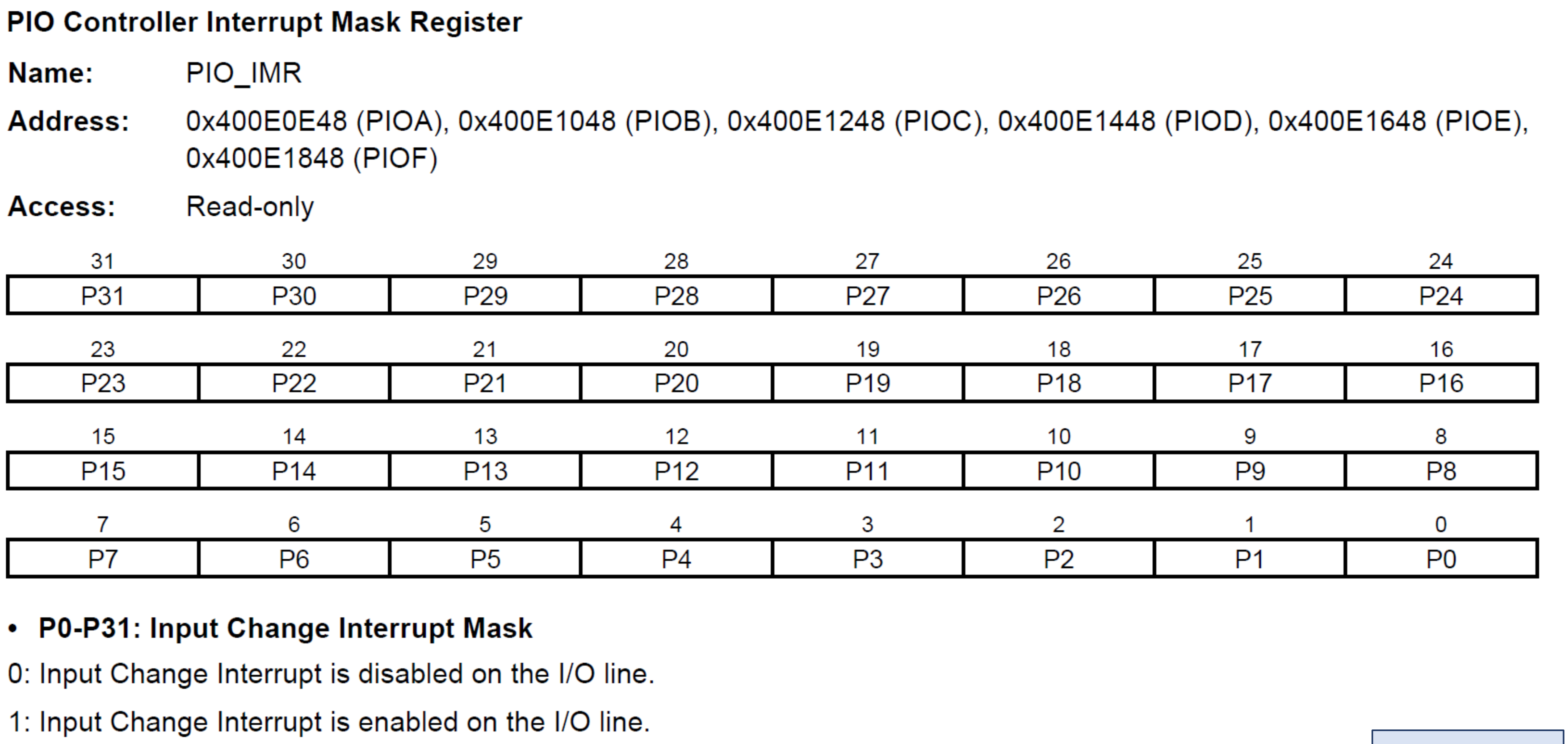
- **P0-P31: Input Change Interrupt Disable**

0: No effect.

1: Disables the Input Change Interrupt on the I/O line.

PIO_IER
PIO_IDR
PIO_IMR
PIO_ISR

SAM3x8E PIO Register: PIO_IMR



PIO Controller Interrupt Mask Register

Name: PIO_IMR

Address: 0x400E0E48 (PIOA), 0x400E1048 (PIOB), 0x400E1248 (PIOC), 0x400E1448 (PIOD), 0x400E1648 (PIOE), 0x400E1848 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

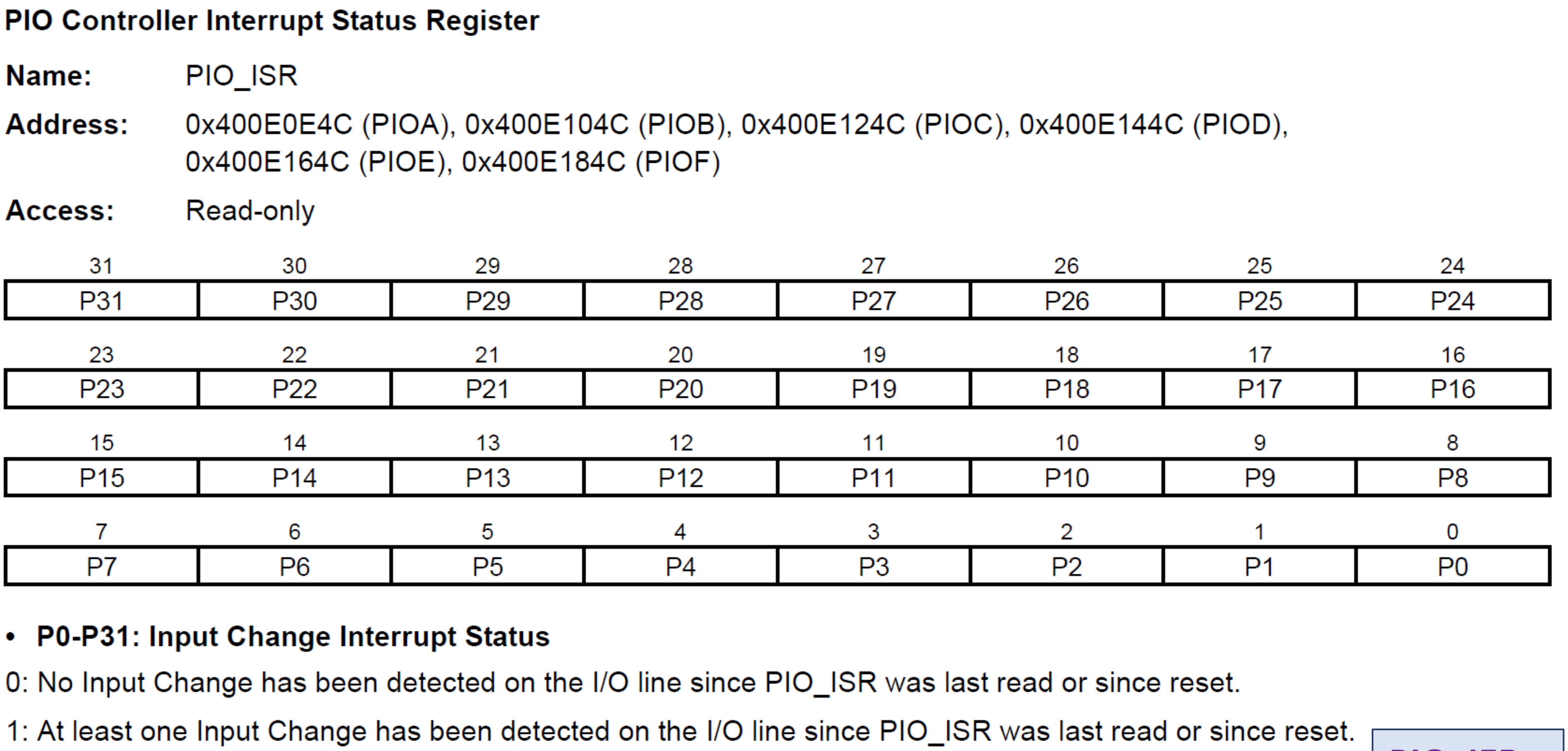
- **P0-P31: Input Change Interrupt Mask**

0: Input Change Interrupt is disabled on the I/O line.

1: Input Change Interrupt is enabled on the I/O line.

PIO_IER
PIO_IDR
PIO_IMR
PIO_ISR

SAM3x8E PIO Register: PIO_ISR



PIO Controller Interrupt Status Register

Name: PIO_ISR

Address: 0x400E0E4C (PIOA), 0x400E104C (PIOB), 0x400E124C (PIOC), 0x400E144C (PIOD),
0x400E164C (PIOE), 0x400E184C (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

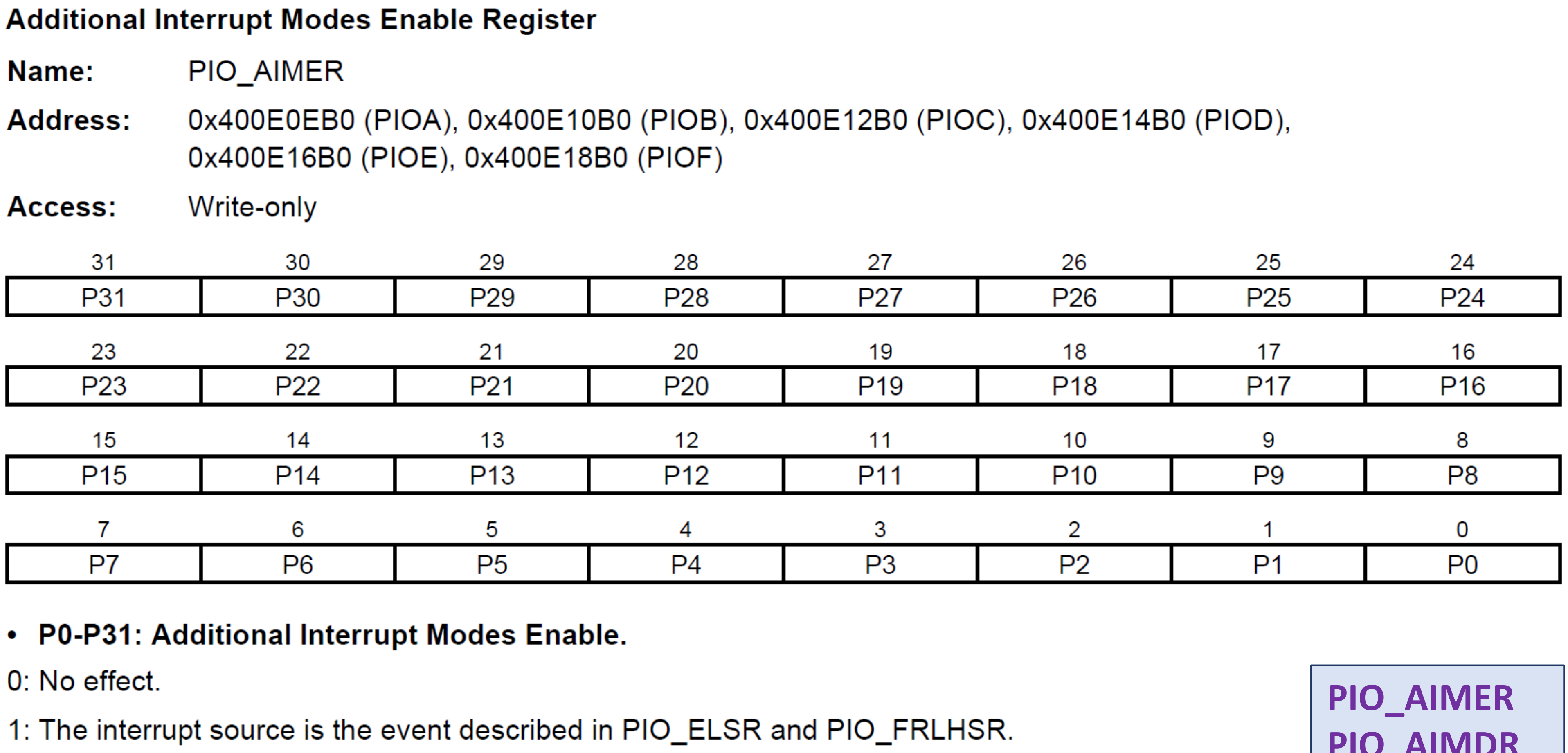
- **P0-P31: Input Change Interrupt Status**

0: No Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

1: At least one Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

PIO_IER
PIO_IDR
PIO_IMR
PIO_ISR

SAM3x8E PIO Register: PIO_AIMER



Additional Interrupt Modes Enable Register

Name: PIO_AIMER

Address: 0x400E0EB0 (PIOA), 0x400E10B0 (PIOB), 0x400E12B0 (PIOC), 0x400E14B0 (PIOD),
0x400E16B0 (PIOE), 0x400E18B0 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

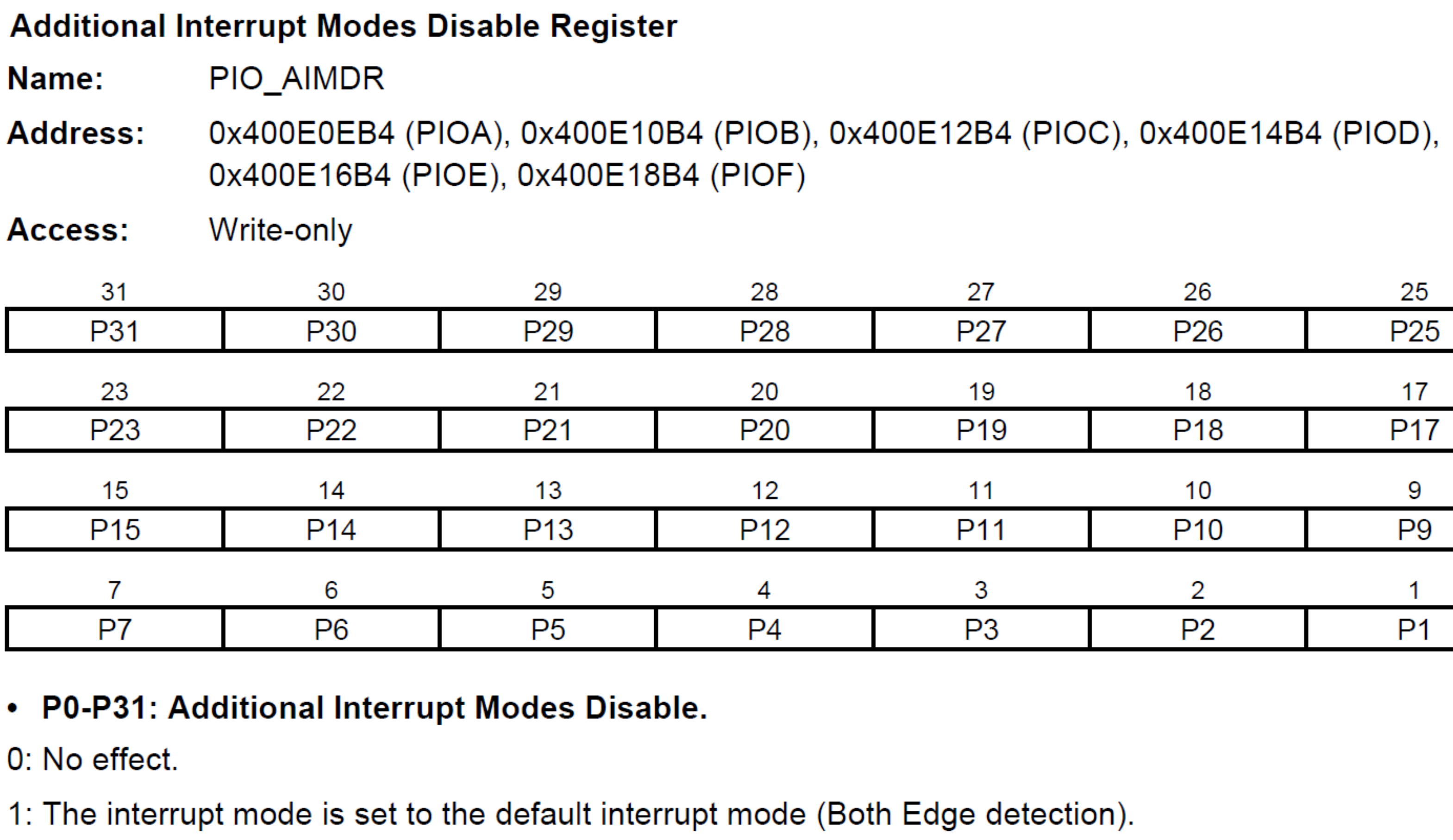
- **P0-P31: Additional Interrupt Modes Enable.**

0: No effect.

1: The interrupt source is the event described in PIO_ELSR and PIO_FRLHSR.

PIO_AIMER
PIO_AIMDR
PIO_AIMMR

SAM3x8E PIO Register: PIO_AIMDR



Additional Interrupt Modes Disable Register

Name: PIO_AIMDR

Address: 0x400E0EB4 (PIOA), 0x400E10B4 (PIOB), 0x400E12B4 (PIOC), 0x400E14B4 (PIOD),
0x400E16B4 (PIOE), 0x400E18B4 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

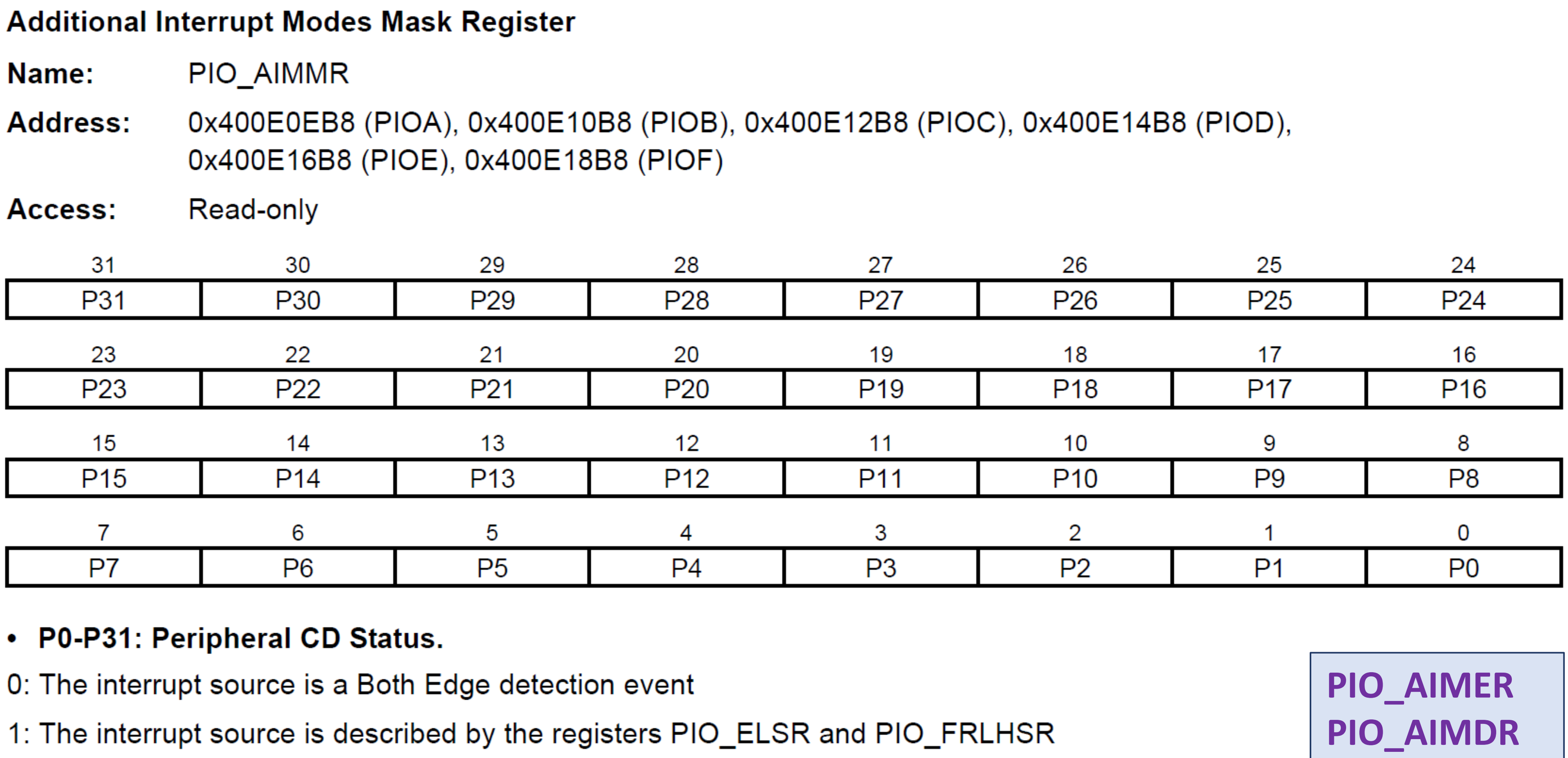
- **P0-P31: Additional Interrupt Modes Disable.**

0: No effect.

1: The interrupt mode is set to the default interrupt mode (Both Edge detection).

PIO_AIMER
PIO_AIMDR
PIO_AIMMR

SAM3x8E PIO Register: PIO_AIMMR



Additional Interrupt Modes Mask Register

Name: PIO_AIMMR

Address: 0x400E0EB8 (PIOA), 0x400E10B8 (PIOB), 0x400E12B8 (PIOC), 0x400E14B8 (PIOD), 0x400E16B8 (PIOE), 0x400E18B8 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

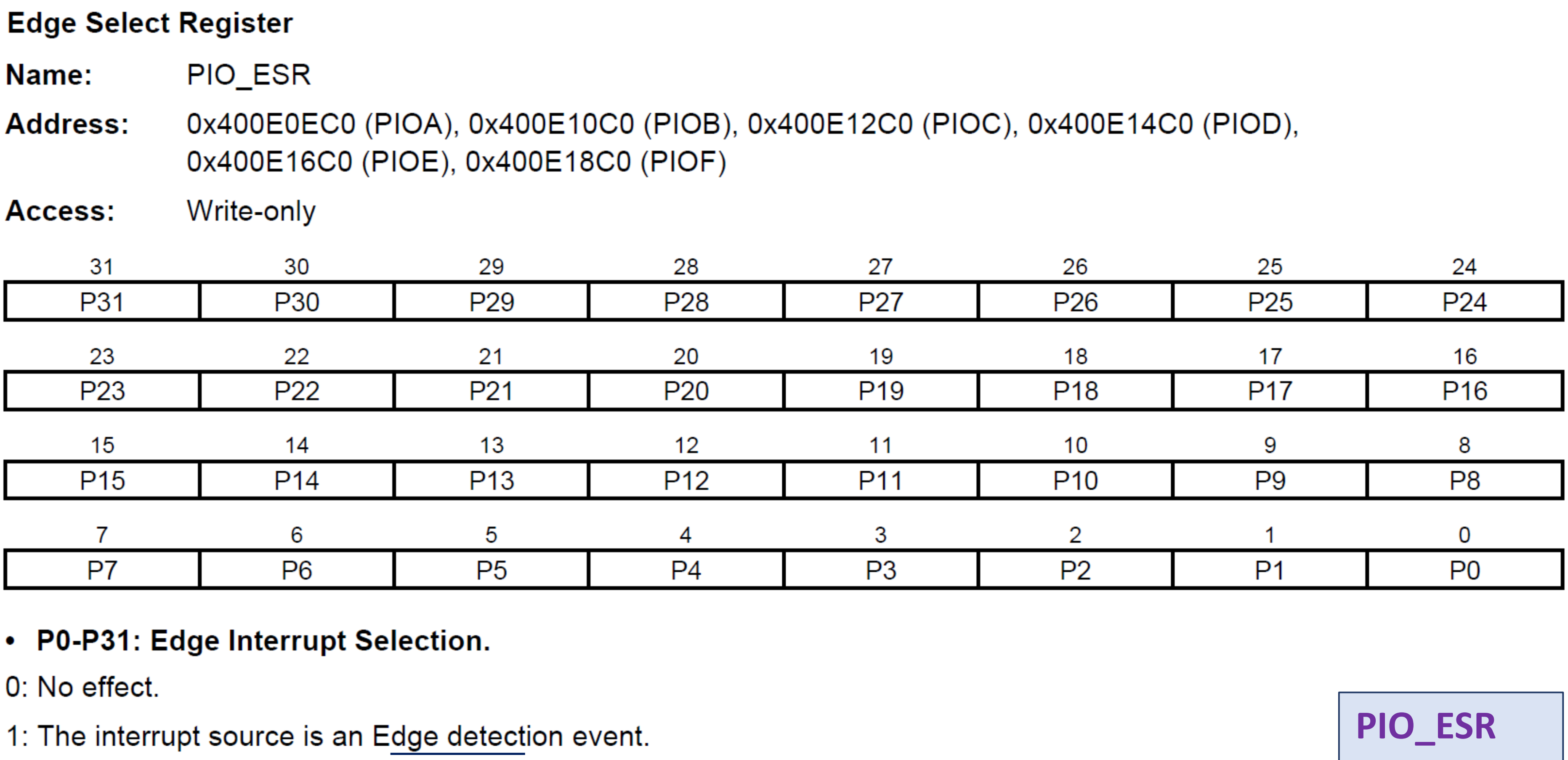
- **P0-P31: Peripheral CD Status.**

0: The interrupt source is a Both Edge detection event

1: The interrupt source is described by the registers PIO_ELSR and PIO_FRLHSR

PIO_AIMER
PIO_AIMDR
PIO_AIMMR

SAM3x8E PIO Register: PIO_ESR



Edge Select Register

Name: PIO_ESR

Address: 0x400E0EC0 (PIOA), 0x400E10C0 (PIOB), 0x400E12C0 (PIOC), 0x400E14C0 (PIOD),
0x400E16C0 (PIOE), 0x400E18C0 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

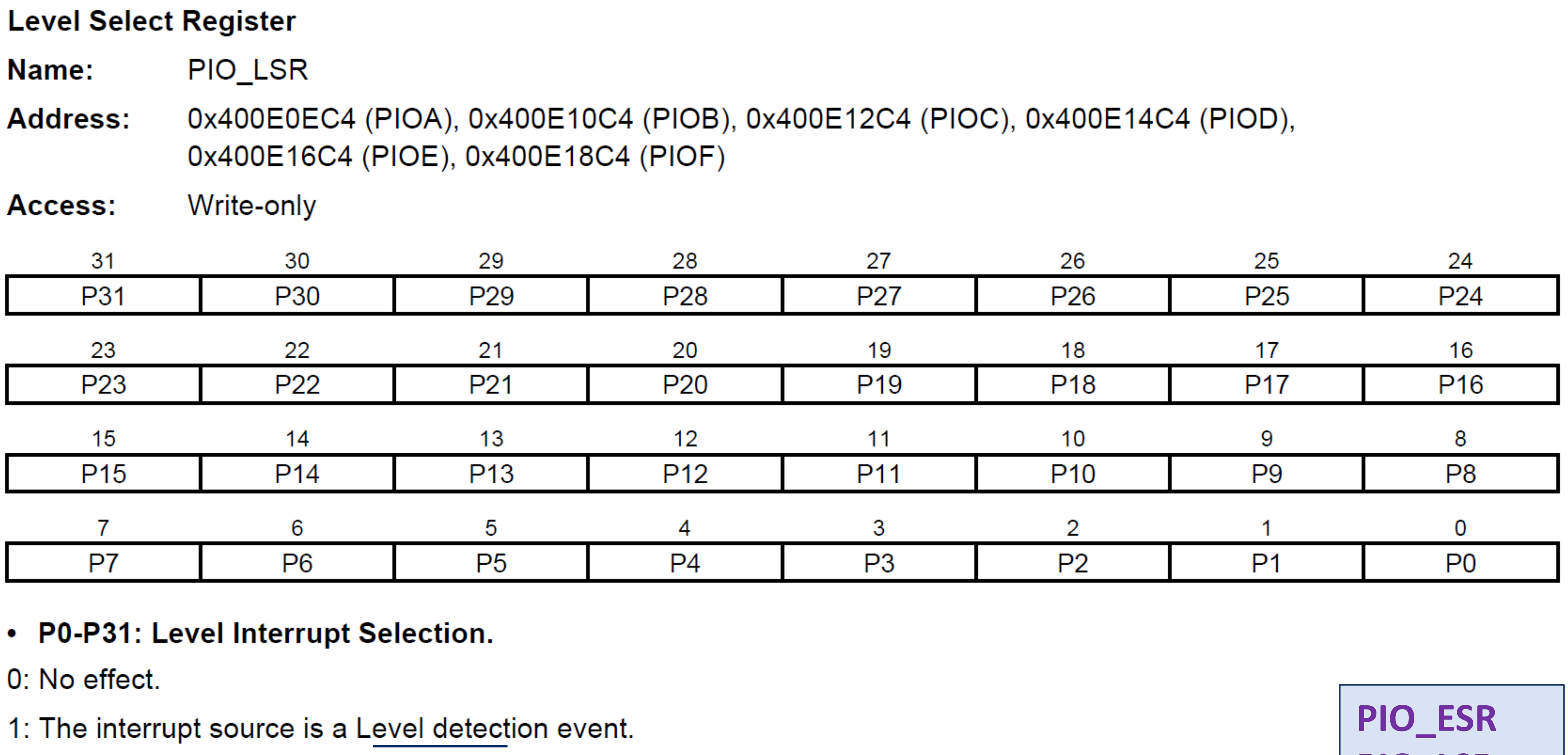
- **P0-P31: Edge Interrupt Selection.**

0: No effect.

1: The interrupt source is an Edge detection event.

PIO_ESR
PIO_LSR
PIO_ELSR

SAM3x8E PIO Register: PIO_LSR



Level Select Register

Name: PIO_LSR

Address: 0x400E0EC4 (PIOA), 0x400E10C4 (PIOB), 0x400E12C4 (PIOC), 0x400E14C4 (PIOD), 0x400E16C4 (PIOE), 0x400E18C4 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

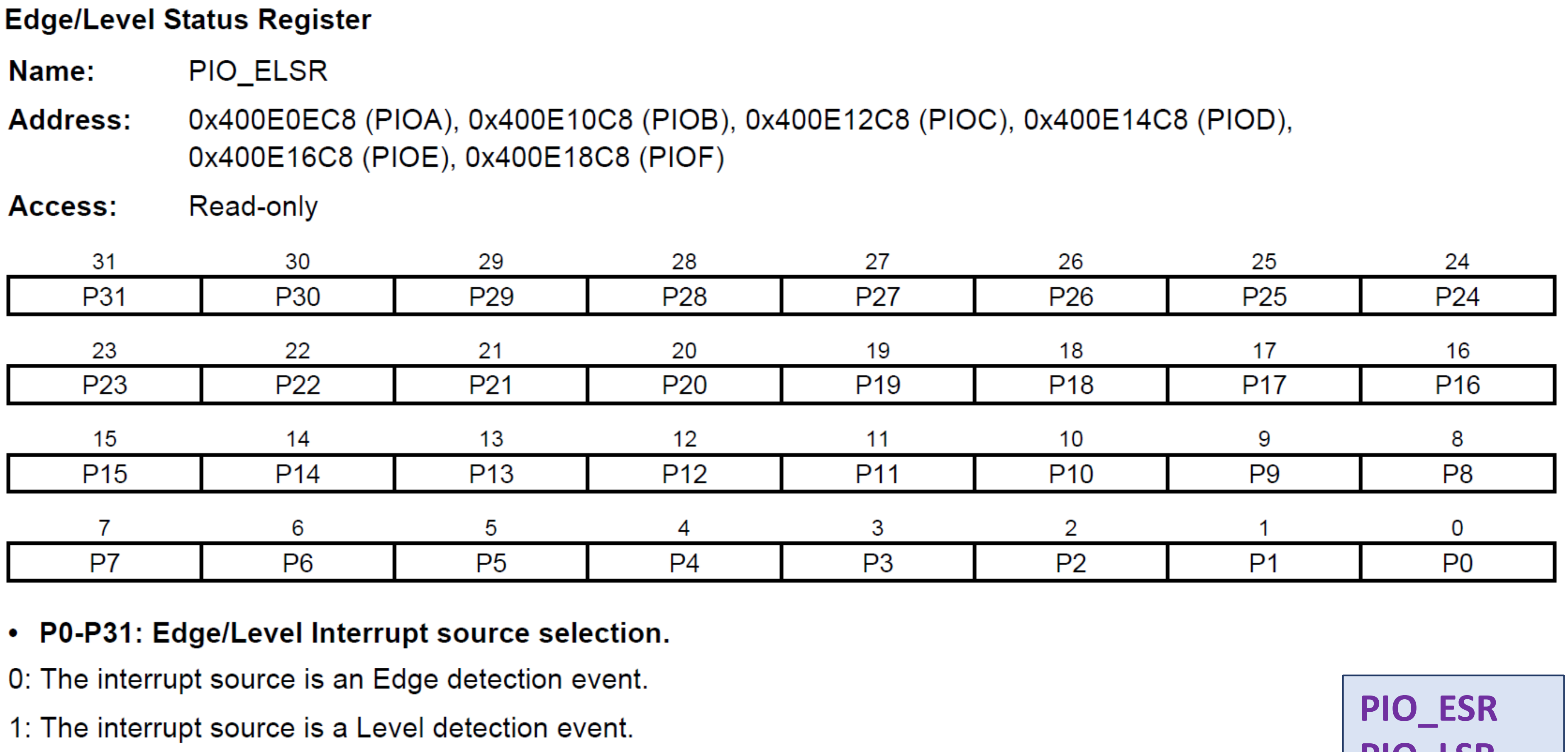
- **P0-P31: Level Interrupt Selection.**

0: No effect.

1: The interrupt source is a Level detection event.

PIO_ESR
PIO_LSR
PIO_ELSR

SAM3x8E PIO Register: PIO_ELSR



Edge/Level Status Register

Name: PIO_ELSR

Address: 0x400E0EC8 (PIOA), 0x400E10C8 (PIOB), 0x400E12C8 (PIOC), 0x400E14C8 (PIOD),
0x400E16C8 (PIOE), 0x400E18C8 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

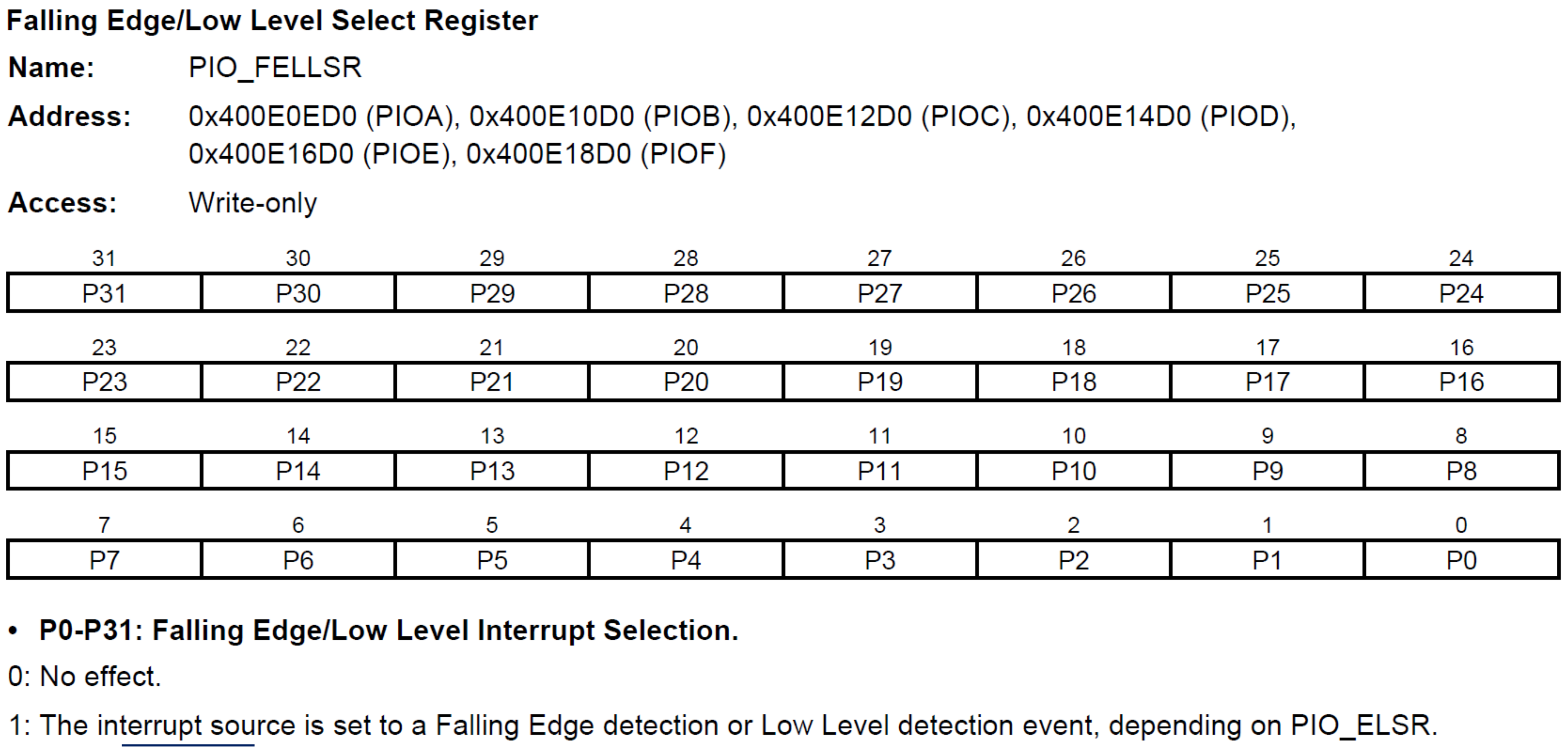
- **P0-P31: Edge/Level Interrupt source selection.**

0: The interrupt source is an Edge detection event.

1: The interrupt source is a Level detection event.

PIO_ESR
PIO_LSR
PIO_ELSR

SAM3x8E PIO Register: PIO_FELLSR



Falling Edge/Low Level Select Register

Name: PIO_FELLSR

Address: 0x400E0ED0 (PIOA), 0x400E10D0 (PIOB), 0x400E12D0 (PIOC), 0x400E14D0 (PIOD),
0x400E16D0 (PIOE), 0x400E18D0 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

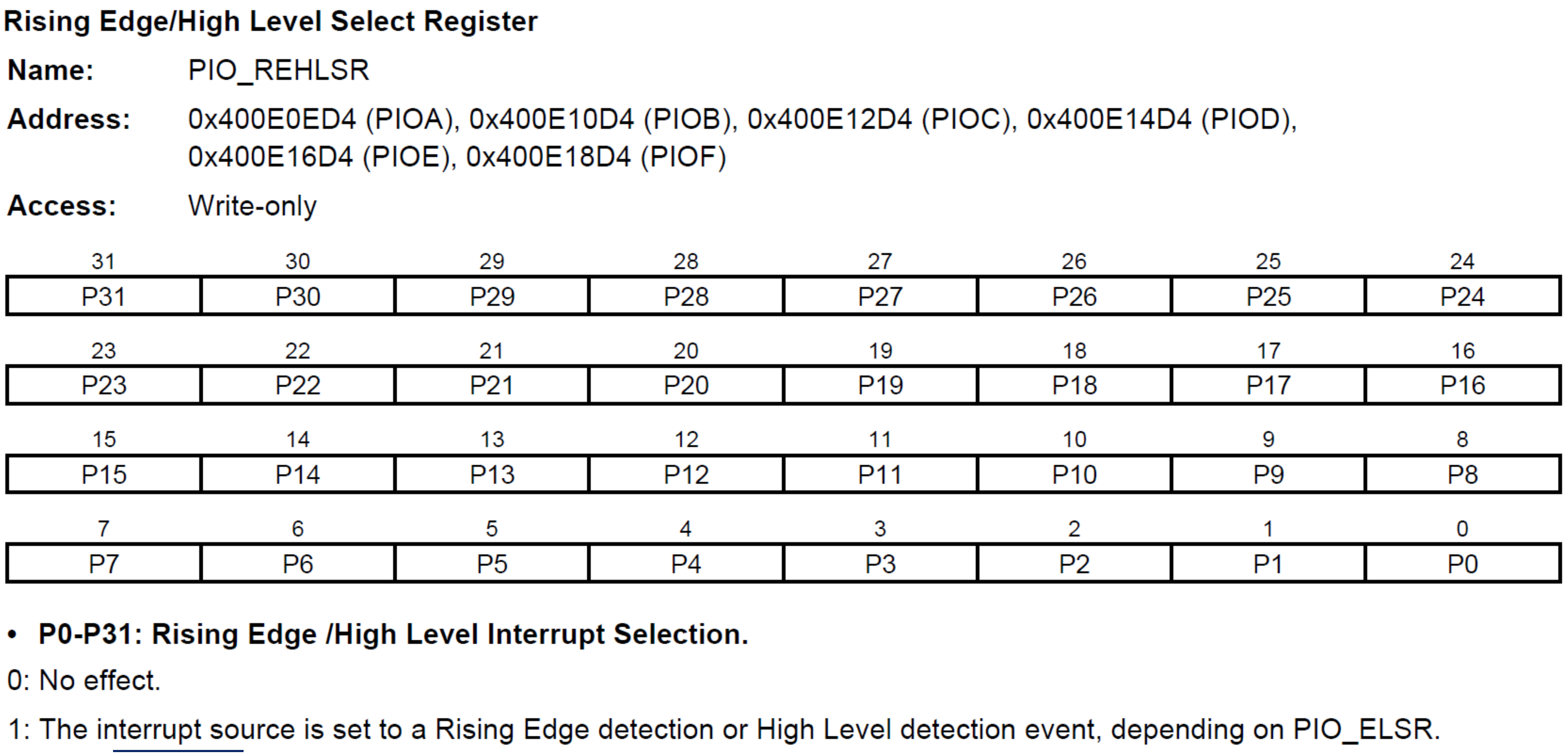
- **P0-P31: Falling Edge/Low Level Interrupt Selection.**

0: No effect.

1: The interrupt source is set to a Falling Edge detection or Low Level detection event, depending on PIO_ELSR.

PIO_FELLSR
PIO_REHLSR
PIO_FRLHSR

SAM3x8E PIO Register: PIO_REHLSR



Rising Edge/High Level Select Register

Name: PIO_REHLSR

Address: 0x400E0ED4 (PIOA), 0x400E10D4 (PIOB), 0x400E12D4 (PIOC), 0x400E14D4 (PIOD),
0x400E16D4 (PIOE), 0x400E18D4 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

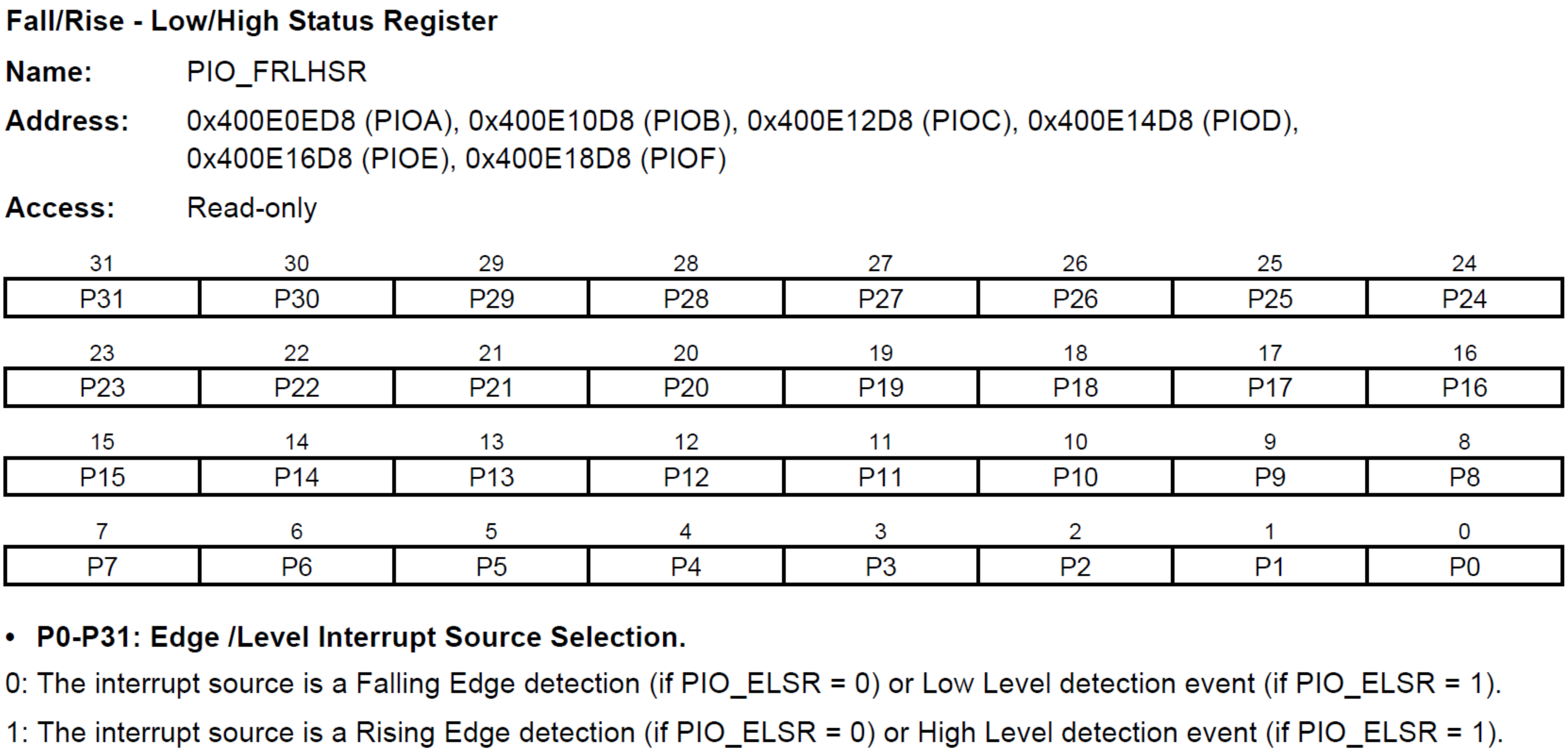
- **P0-P31: Rising Edge /High Level Interrupt Selection.**

0: No effect.

1: The interrupt source is set to a Rising Edge detection or High Level detection event, depending on PIO_ELSR.

PIO_FELLSR
PIO_REHLSR
PIO_FRLHSR

SAM3x8E PIO Register: PIO_FRLHSR



Fall/Rise - Low/High Status Register

Name: PIO_FRLHSR

Address: 0x400E0ED8 (PIOA), 0x400E10D8 (PIOB), 0x400E12D8 (PIOC), 0x400E14D8 (PIOD),
0x400E16D8 (PIOE), 0x400E18D8 (PIOF)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

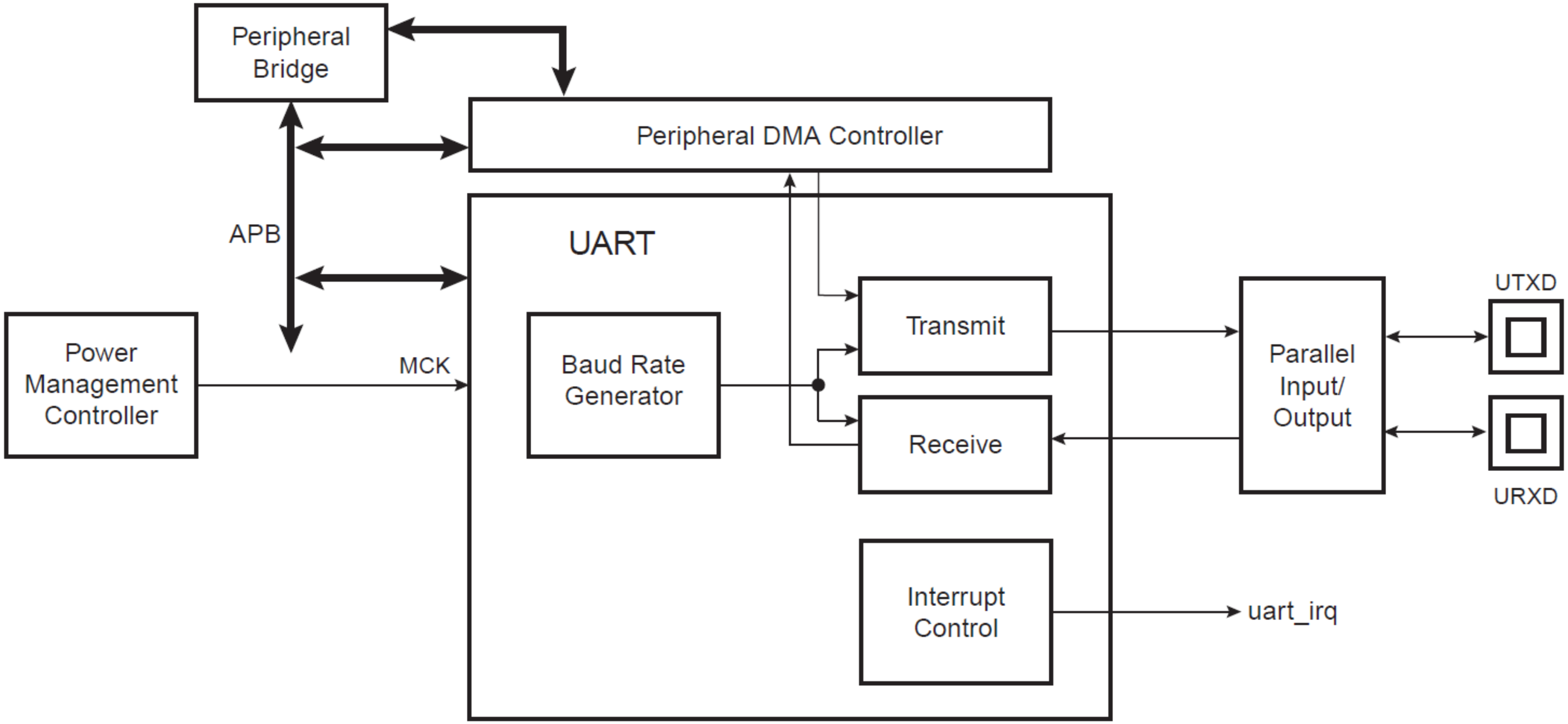
- **P0-P31: Edge /Level Interrupt Source Selection.**

0: The interrupt source is a Falling Edge detection (if PIO_ELSR = 0) or Low Level detection event (if PIO_ELSR = 1).

1: The interrupt source is a Rising Edge detection (if PIO_ELSR = 0) or High Level detection event (if PIO_ELSR = 1).

PIO_FELLSR
PIO_REHLSR
PIO_FRLHSR

SAM3X8E: UART



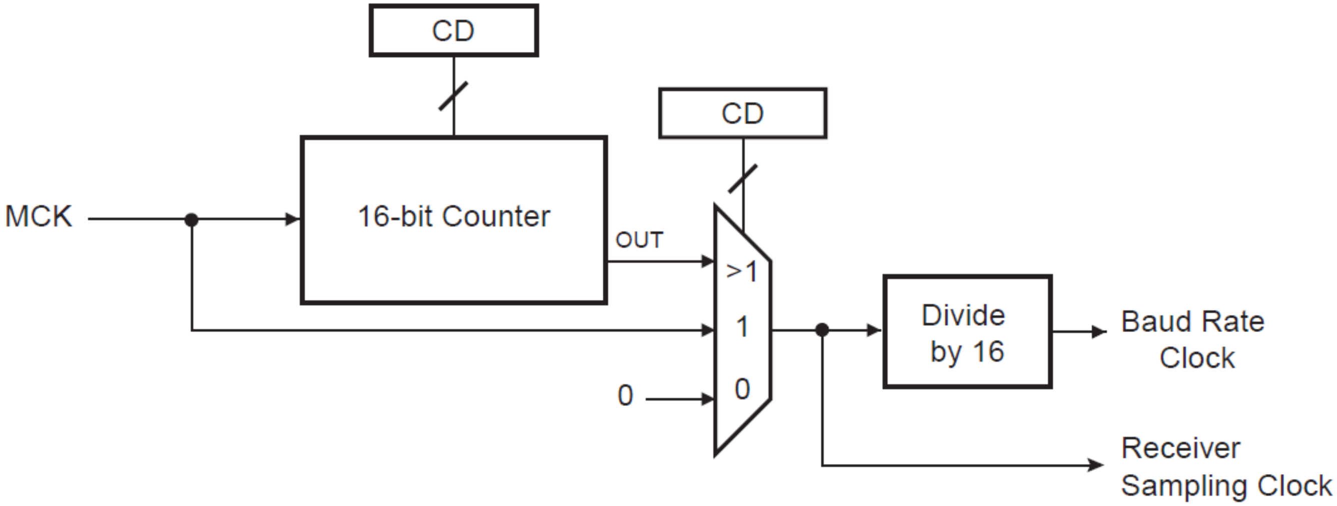
Instance	Signal	I/O Line	Peripheral
UART	URXD	PA8	A
UART	UTXD	PA9	A

The UART pins (Peripheral A pins) are multiplexed with PIO lines.

SAM3X8E: UART Baud Rate Generator

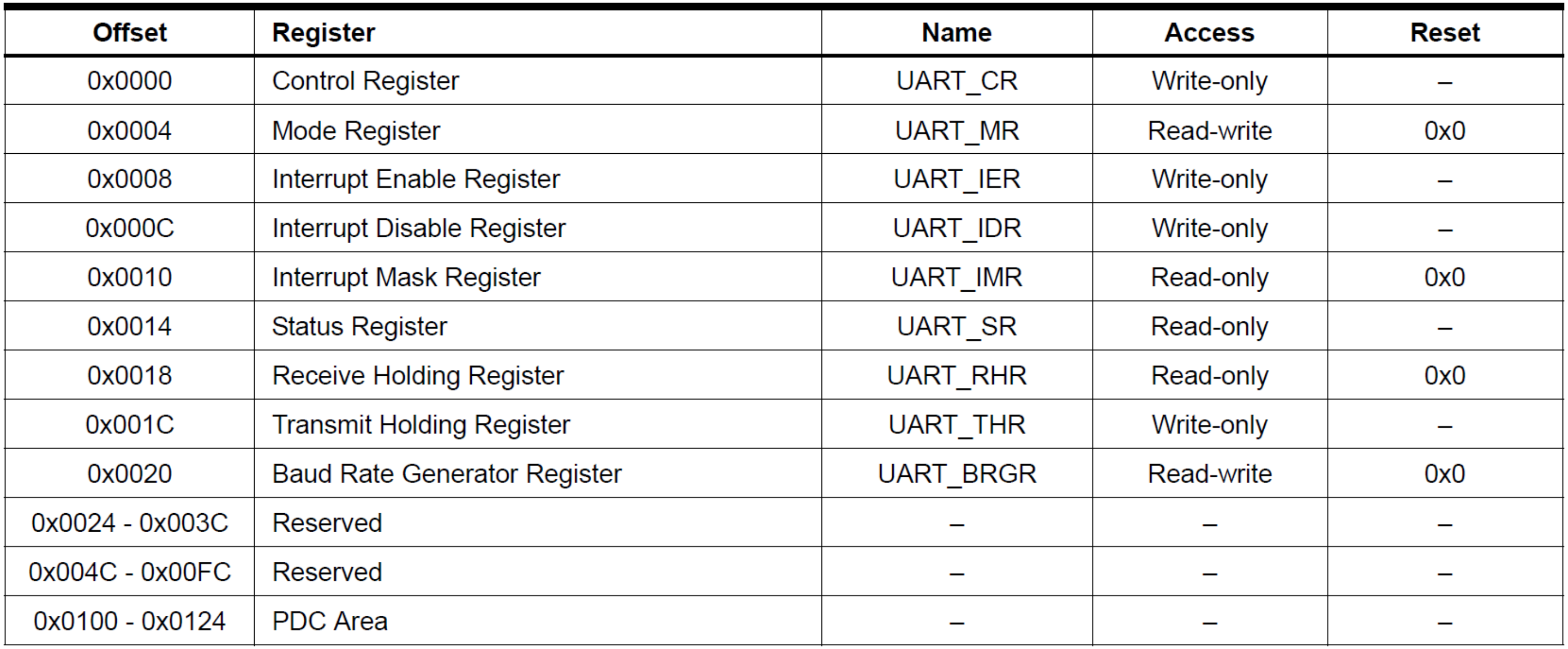
The baud rate clock is the master clock divided by 16 times the value (CD) written in UART_BRGR (Baud Rate Generator Register).

$$\text{Baud Rate} = \frac{\text{MCK}}{16 \times \text{CD}}$$



CD = MCK/(16 * Baud Rate) = 84000000/(16*115200) = **45**.57

SAM3X8E: UART Registers



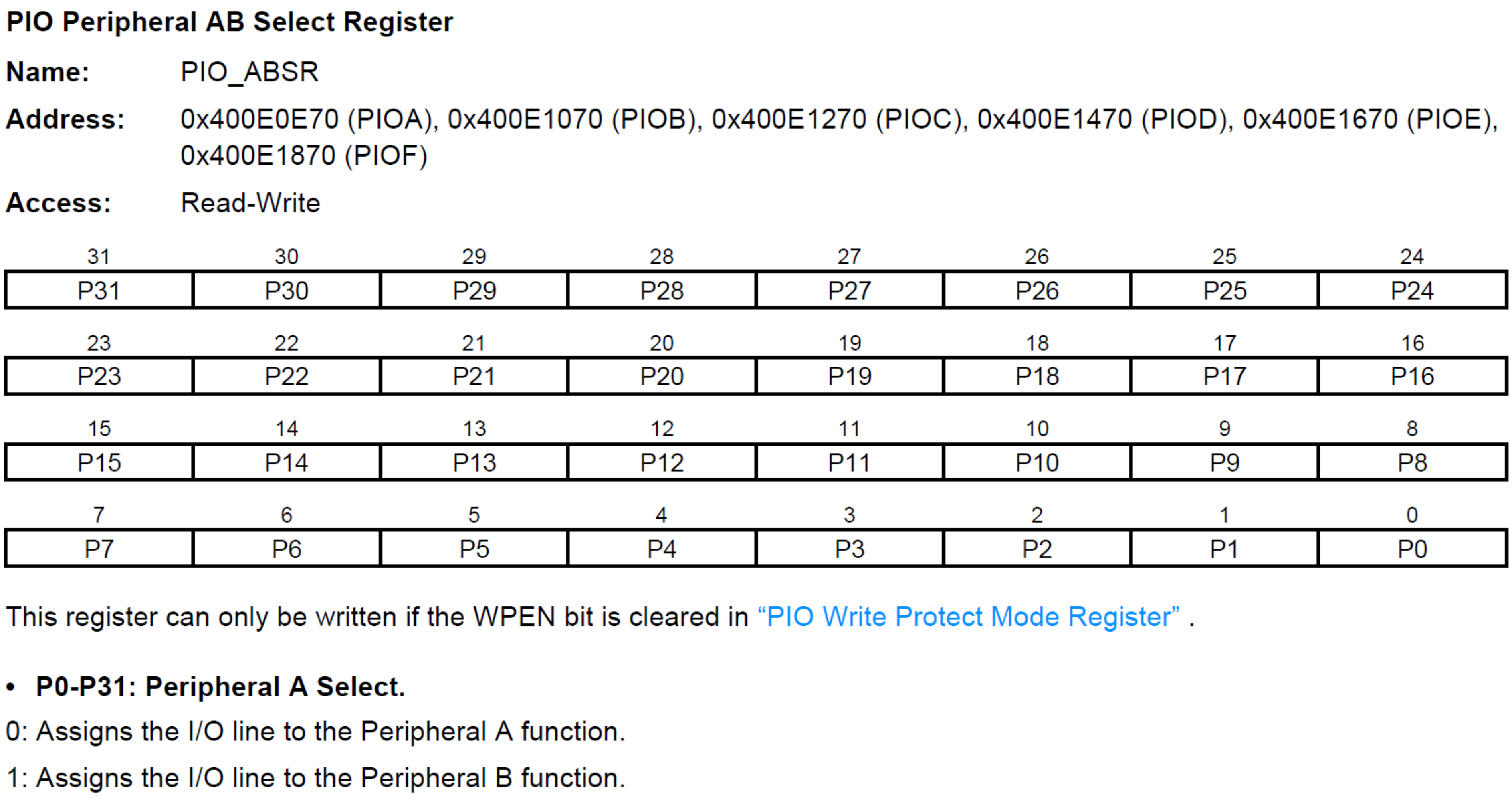
Offset	Register	Name	Access	Reset
0x0000	Control Register	UART_CR	Write-only	–
0x0004	Mode Register	UART_MR	Read-write	0x0
0x0008	Interrupt Enable Register	UART_IER	Write-only	–
0x000C	Interrupt Disable Register	UART_IDR	Write-only	–
0x0010	Interrupt Mask Register	UART_IMR	Read-only	0x0
0x0014	Status Register	UART_SR	Read-only	–
0x0018	Receive Holding Register	UART_RHR	Read-only	0x0
0x001C	Transmit Holding Register	UART_THR	Write-only	–
0x0020	Baud Rate Generator Register	UART_BRGR	Read-write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x004C - 0x00FC	Reserved	–	–	–
0x0100 - 0x0124	PDC Area	–	–	–

Arduino DUE Programming: UART Loopback

```
#include "sam.h"

#define BUAD_RATE  (115200)
#define UART_CD    (SystemCoreClock / (16*BUAD_RATE))

// PA8 = RX0 and PA9 = TX0
void init_UART() {
  // enable PMC CLK for the UART
  PMC->PMC_PCER0 = (1 << ID_UART);
  // enable pull-up resistors on the RX0/TX0 pins
  PIOA->PIO_PUER = PIO_PA8A_URXD | PIO_PA9A_UTXD;
  // enable peripheral pins for TX0/RX0 pins
  PIOA->PIO_PDR = PIO_PA8A_URXD | PIO_PA9A_UTXD;
  // set pins to use peripheral A
  PIOA->PIO_ABSR &= ~(PIO_PA8A_URXD | PIO_PA9A_UTXD);

  // reset & disable both RX and TX operation
  UART->UART_CR = UART_CR_RSTRX | UART_CR_RSTTX
                | UART_CR_RXDIS | UART_CR_TXDIS;
  // set the baud rate to 115200
  UART->UART_BRGR = UART_CD;
  UART->UART_MR = UART_MR_PAR_NO; // no parity
  // enable both receiver and transmitter
  UART->UART_CR = UART_CR_RXEN | UART_CR_TXEN;
}
```

```
uint8_t get_char() {
  // wait until Rx is ready (a complete char is received)
  while( !(UART->UART_SR & UART_SR_RXRDY) ){ }
  return UART->UART_RHR; // read from UART RX Holding Register
}

void put_char( uint8_t ch ) {
  // wait until Tx is ready
  while( !(UART->UART_SR & UART_SR_TXRDY)) { }
  UART->UART_THR = ch; // write to UART TX Holding Register
}

void put_str( const char *str ) {
  while (*str) {
     put_char( *str++ );
  }
  while( !((UART->UART_SR) & UART_SR_TXEMPTY)) { }
}

int main(void) {
  SystemInit(); // initialize the system
  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  init_UART();
  while (1) { // UART loopback test
    uint8_t ch = get_char(); // read next the incoming byte
    put_char( ch );          // send the received byte back
  }
}
```

PIO Controllers: PIO_ABSR Register



PIO Peripheral AB Select Register

Name: PIO_ABSR

Address: 0x400E0E70 (PIOA), 0x400E1070 (PIOB), 0x400E1270 (PIOC), 0x400E1470 (PIOD), 0x400E1670 (PIOE), 0x400E1870 (PIOF)

Access: Read-Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in "PIO Write Protect Mode Register" .

- **P0-P31: Peripheral A Select.**

0: Assigns the I/O line to the Peripheral A function.

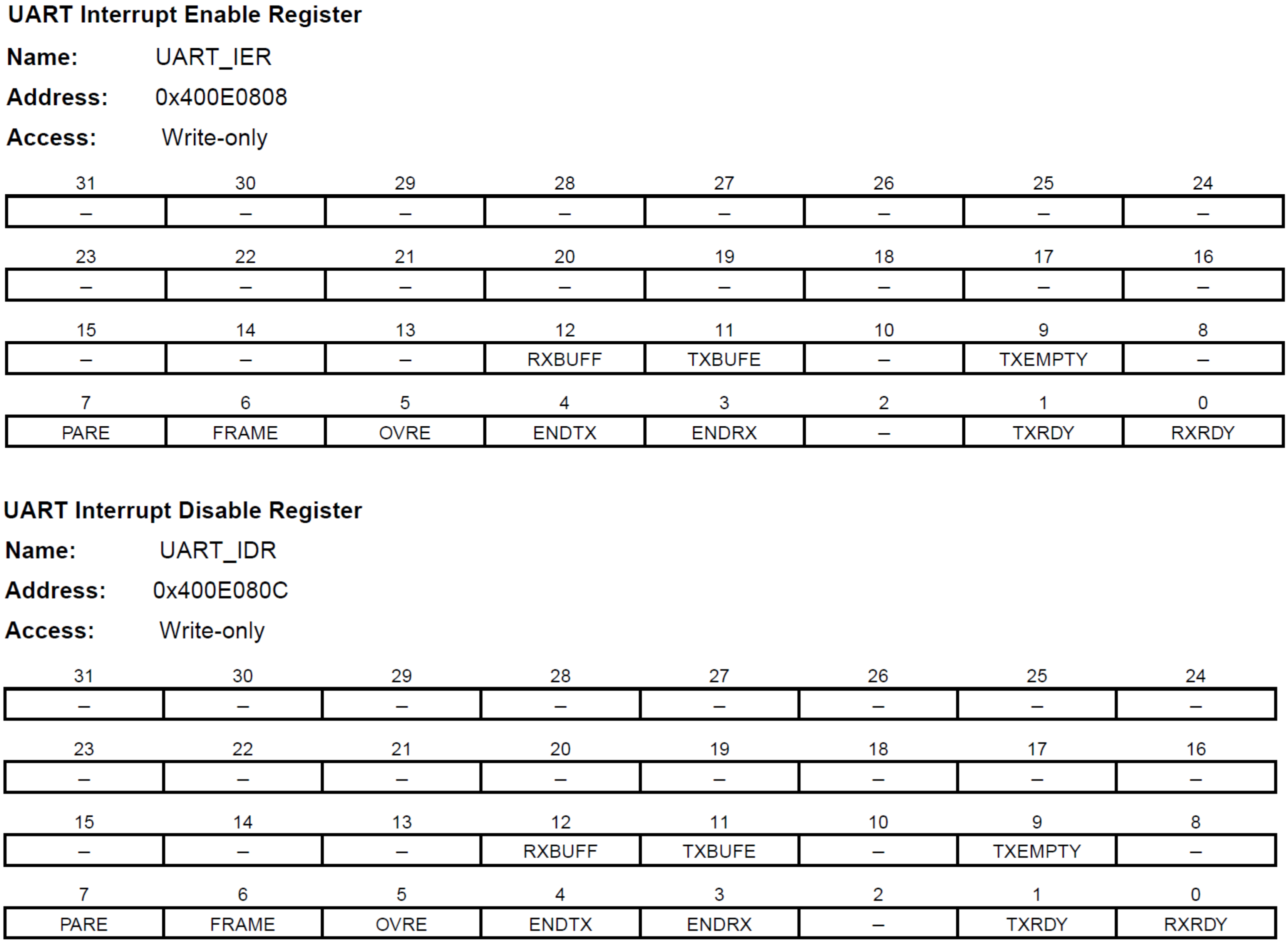
1: Assigns the I/O line to the Peripheral B function.

Arduino DUE Programming: UART Loopback with Interrupt

```
#include "sam.h"
#include <stdio.h>

#define BUAD_RATE  (115200)
#define UART_CD    (SystemCoreClock/(16*BUAD_RATE))

// PA8 = RX0 and PA9 = TX0
void init_UART() {
  // enable PMC CLK for the UART
  PMC->PMC_PCER0 = (1 << ID_UART);
  // enable pull-up resistors on the RX0/TX0 pins
  PIOA->PIO_PUER = PIO_PA8A_URXD | PIO_PA9A_UTXD;
  // enable peripheral pins for TX0/RX0 pins
  PIOA->PIO_PDR  = PIO_PA8A_URXD | PIO_PA9A_UTXD;
  // set pins to use peripheral A
  PIOA->PIO_ABSR &= ~(PIO_PA8A_URXD | PIO_PA9A_UTXD);

  // reset & disable both RX and TX operation
  UART->UART_CR = UART_CR_RSTRX | UART_CR_RSTTX
                | UART_CR_RXDIS | UART_CR_TXDIS;
  // set the baud rate to 115200
  UART->UART_BRGR = UART_CD;
  UART->UART_MR = UART_MR_PAR_NO; // no parity
  // enable RXRDY interrupt
  NVIC_EnableIRQ( (IRQn_Type) ID_UART );
  UART->UART_IER = UART_IER_RXRDY;
  // enable both receiver and transmitter
  UART->UART_CR = UART_CR_RXEN | UART_CR_TXEN;
}
```

```
volatile uint8_t data;

void UART_Handler(void) {
  if ( UART->UART_SR & UART_SR_RXRDY ) {
     data = UART->UART_RHR;
     // wait until Tx is ready
     while( !(UART->UART_SR & UART_SR_TXRDY)) {}
     UART->UART_THR = data;
   }
}

int main(void) {
  SystemInit(); // initialize the system
  WDT->WDT_MR = WDT_MR_WDDIS; // disable WDT
  init_UART();
  while (1) {}
}
```

SAM3x8E PIO Register: UART_IER & UART_IDR



UART Interrupt Enable Register

Name: UART_IER

Address: 0x400E0808

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

UART Interrupt Disable Register

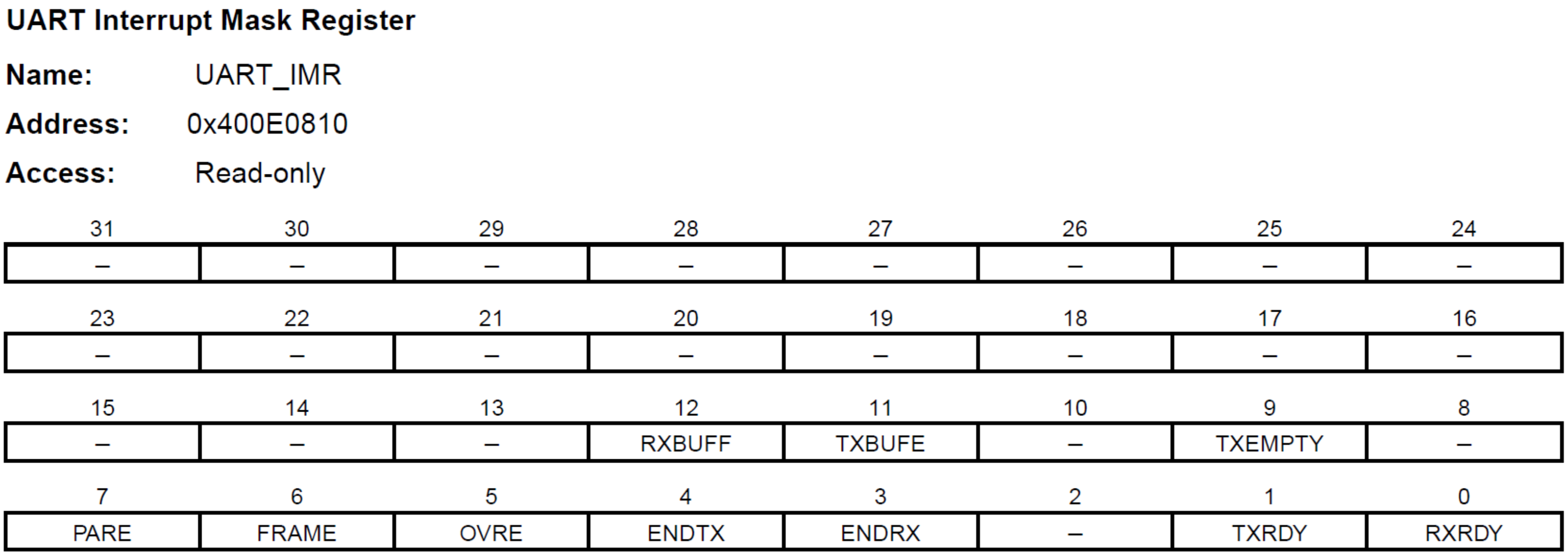
Name: UART_IDR

Address: 0x400E080C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

SAM3x8E PIO Register: UART_IMR



UART Interrupt Mask Register

Name: UART_IMR

Address: 0x400E0810

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

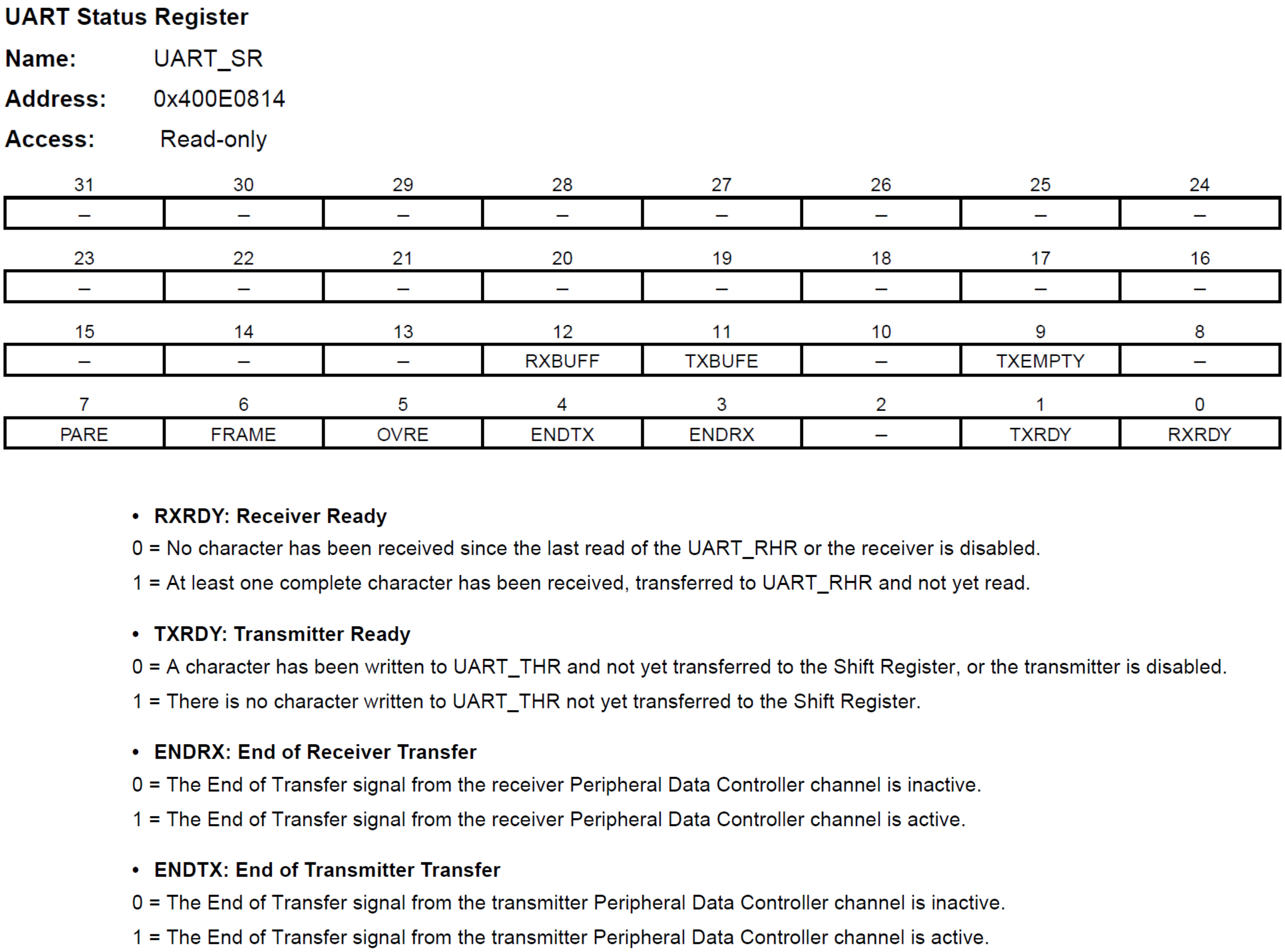
- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**

- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

SAM3x8E PIO Register: UART_SR



UART Status Register

Name: UART_SR

Address: 0x400E0814

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the UART_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to UART_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to UART_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to UART_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

0 = The End of Transfer signal from the receiver Peripheral Data Controller channel is inactive.

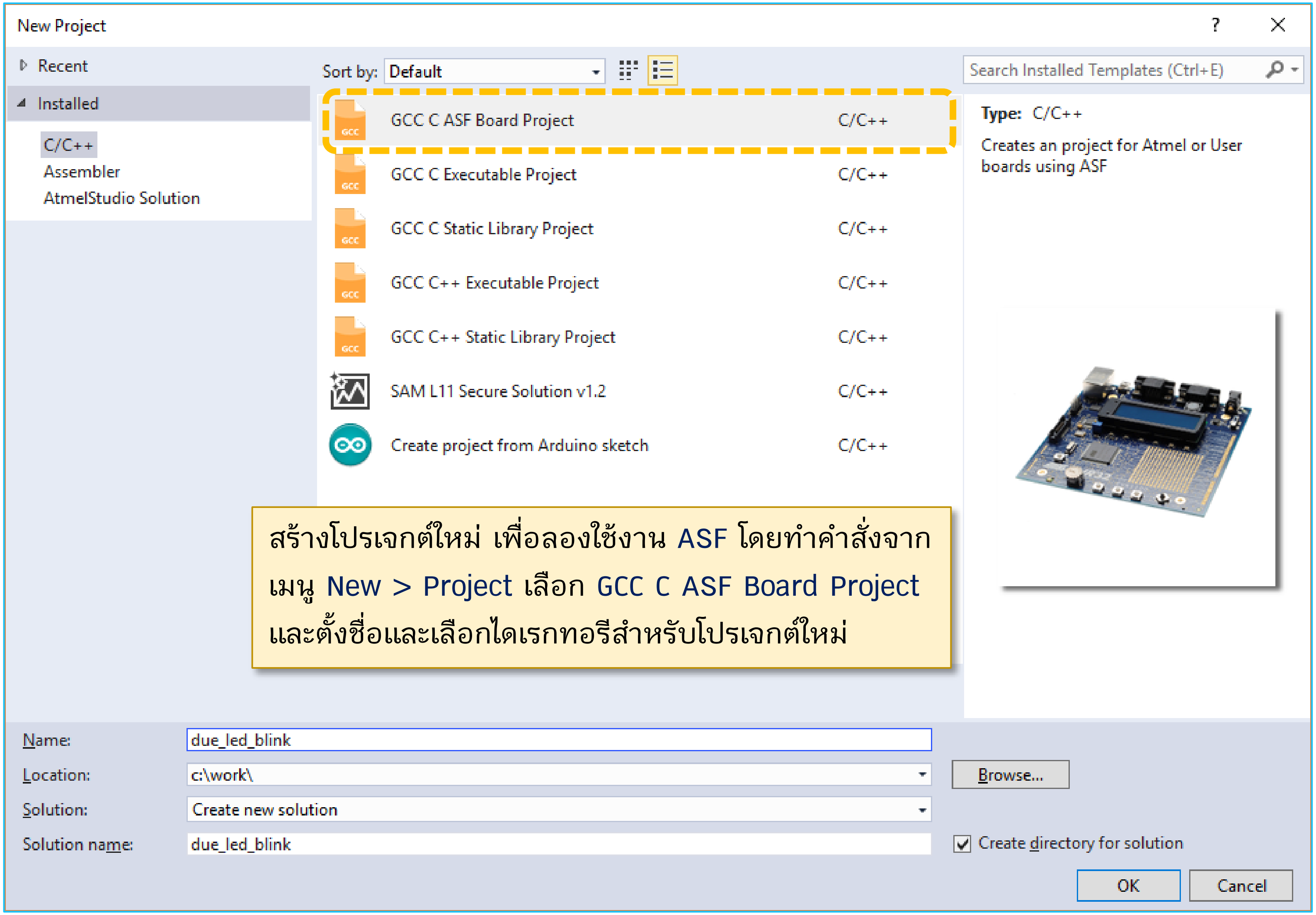
1 = The End of Transfer signal from the receiver Peripheral Data Controller channel is active.

- **ENDTX: End of Transmitter Transfer**

0 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is active.

AVR Studio 7 + Arduino DUE: GCC C ASF Project



GCC C ASF Board Project	C/C++
GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
SAM L11 Secure Solution v1.2	C/C++
Create project from Arduino sketch	C/C++

Type: C/C++

Creates an project for Atmel or User boards using ASF

Name: due_led_blink

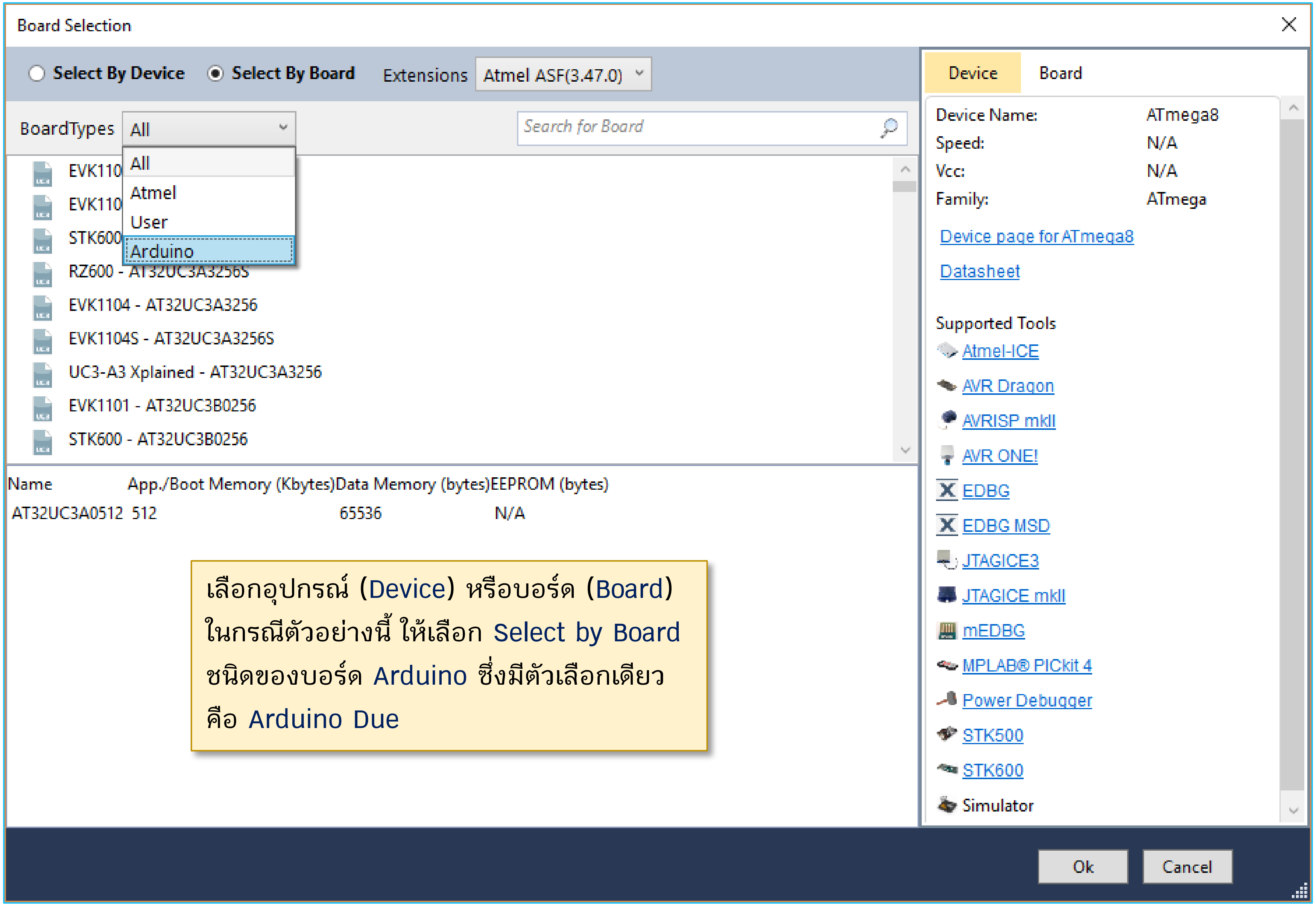
Location: c:\work\

Solution: Create new solution

Solution name: due_led_blink

สร้างโปรเจกต์ใหม่ เพื่อลองใช้งาน ASF โดยทำคำสั่งจาก เมนู New > Project เลือก GCC C ASF Board Project และตั้งชื่อและเลือกไดเรกทอรีสำหรับโปรเจกต์ใหม่

AVR Studio 7 + Arduino DUE: Device / Board Selection



Board Selection

○ Select By Device ● Select By Board Extensions Atmel ASF(3.47.0)

BoardTypes All

Search for Board

- All
- Atmel
- User
- Arduino

- EVK110
- EVK110
- STK600
- RZ600 - AT32UC3A3256S
- EVK1104 - AT32UC3A3256
- EVK1104S - AT32UC3A3256S
- UC3-A3 Xplained - AT32UC3A3256
- EVK1101 - AT32UC3B0256
- STK600 - AT32UC3B0256

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
AT32UC3A0512	512	65536	N/A

Device Board

Device Name:	ATmega8
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Device page for ATmega8

Datasheet

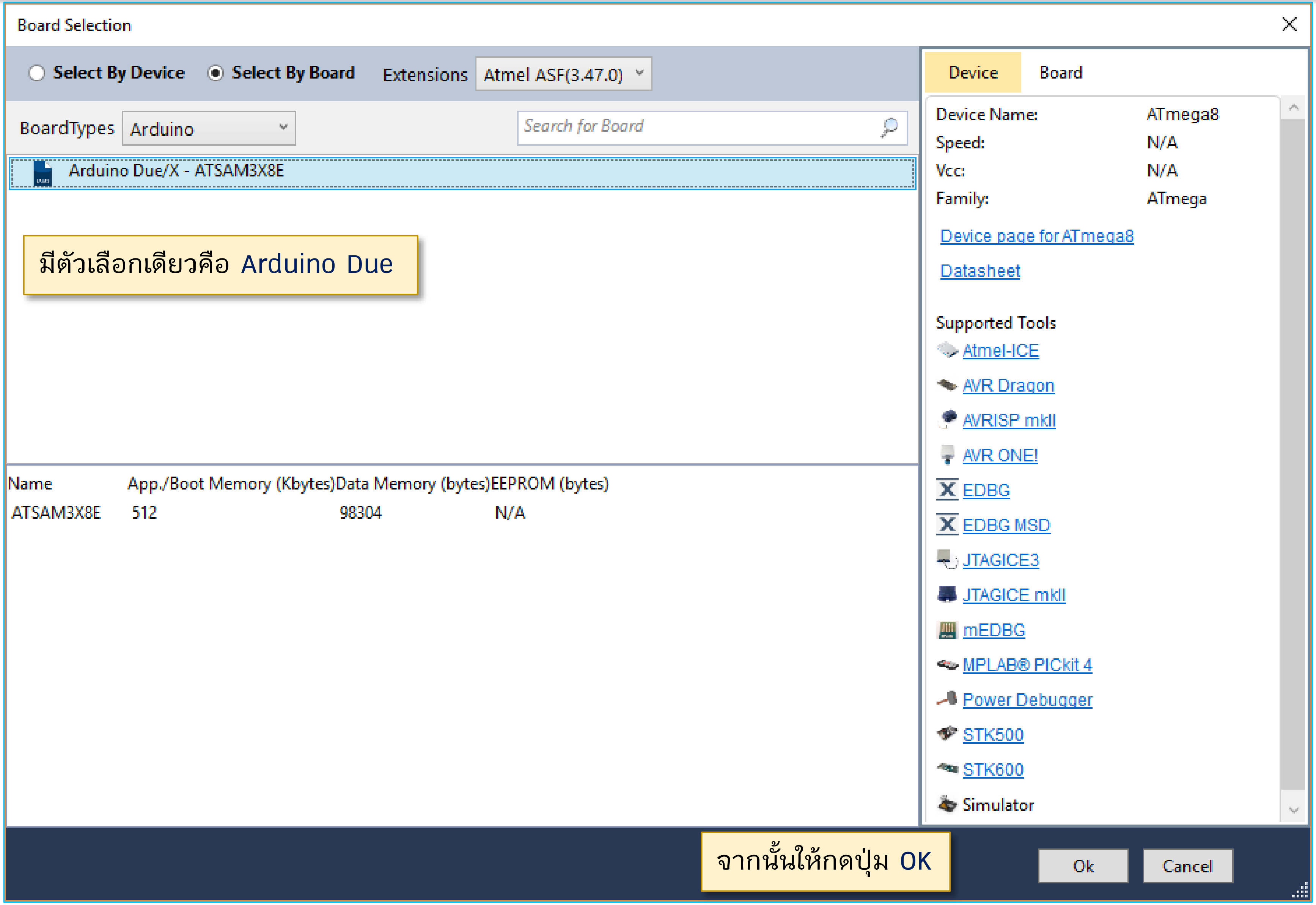
Supported Tools

- Atmel-ICE
- AVR Dragon
- AVRISP mkII
- AVR ONE!
- EDBG
- EDBG MSD
- JTAGICE3
- JTAGICE mkII
- mEDBG
- MPLAB® PICkit 4
- Power Debugger
- STK500
- STK600
- Simulator

Ok Cancel

เลือกอุปกรณ์ (Device) หรือบอร์ด (Board) ในกรณีตัวอย่างนี้ ให้เลือก Select by Board ชนิดของบอร์ด Arduino ซึ่งมีตัวเลือกเดียว คือ Arduino Due

AVR Studio 7 + Arduino DUE: Device / Board Selection



Board Selection

○ Select By Device ● Select By Board Extensions Atmel ASF(3.47.0)

BoardTypes Arduino Search for Board

Arduino Due/X - ATSAM3X8E

มีตัวเลือกเดียวคือ Arduino Due

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATSAM3X8E	512	98304	N/A

Device Board

Device Name:	ATmega8
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Device page for ATmega8

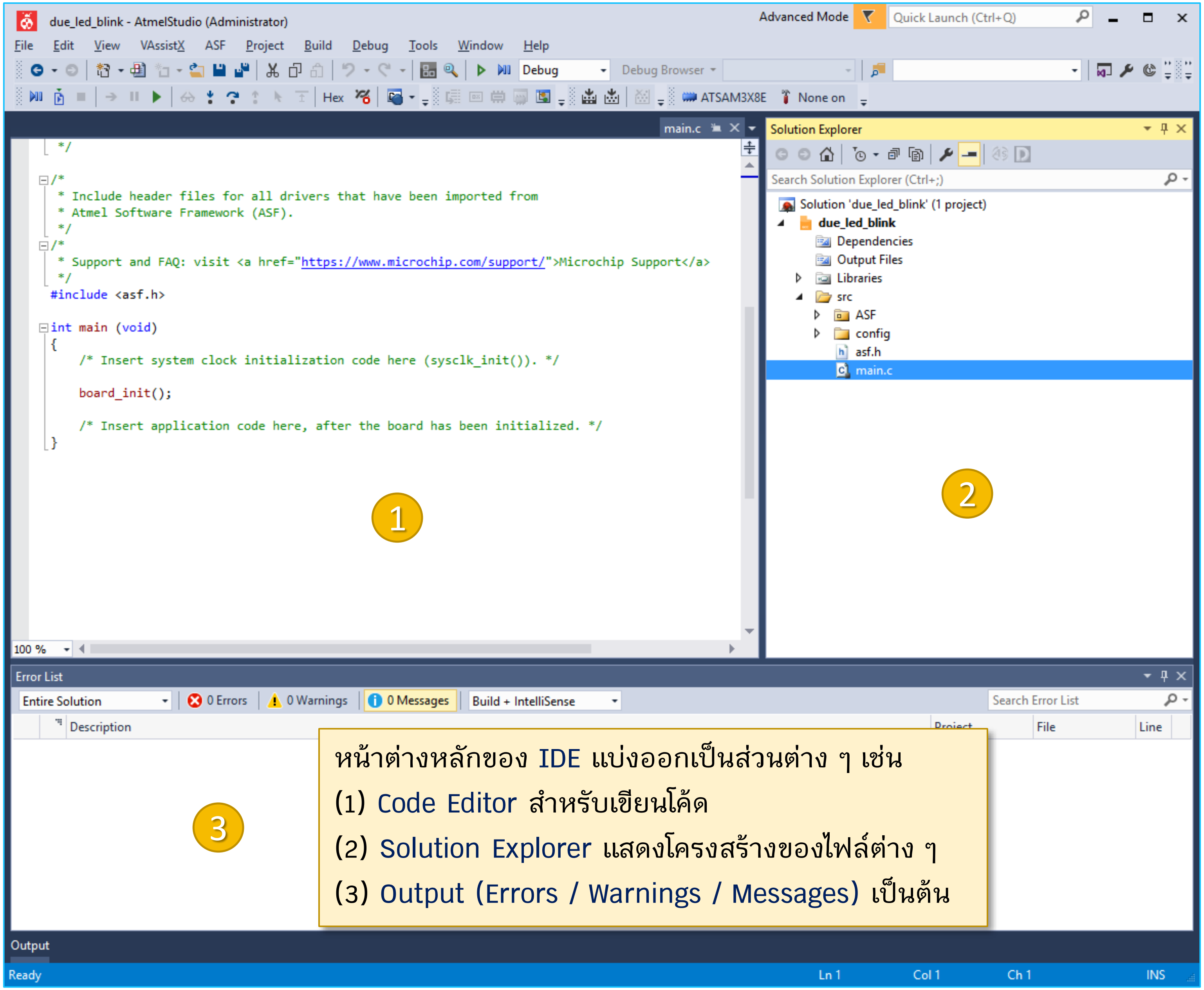
Datasheet

Supported Tools

- Atmel-ICE
- AVR Dragon
- AVRISP mkII
- AVR ONE!
- EDBG
- EDBG MSD
- JTAGICE3
- JTAGICE mkII
- mEDBG
- MPLAB® PICkit 4
- Power Debugger
- STK500
- STK600
- Simulator

จากนั้นให้กดปุ่ม OK

AVR Studio 7 + Arduino DUE: C Source Code Editor



```
 */

/*
 * Include header files for all drivers that have been imported from
 * Atmel Software Framework (ASF).
 */
/*
 * Support and FAQ: visit <a href="https://www.microchip.com/support/">Microchip Support</a>
 */
#include <asf.h>

int main (void)
{
    /* Insert system clock initialization code here (sysclk_init()). */

    board_init();

    /* Insert application code here, after the board has been initialized. */
}
```

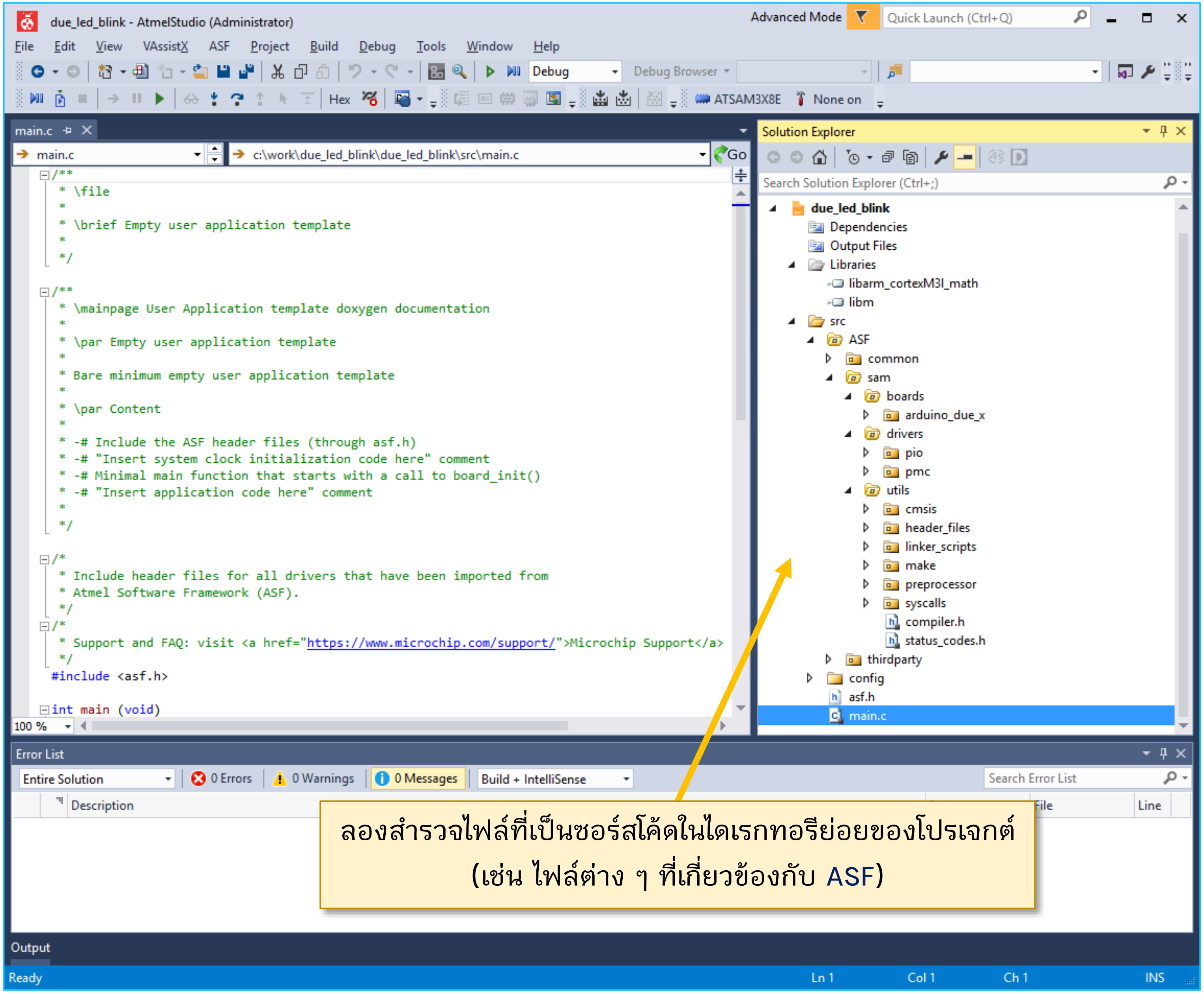
หน้าต่างหลักของ IDE แบ่งออกเป็นส่วนต่าง ๆ เช่น

(1) Code Editor สำหรับเขียนโค้ด

(2) Solution Explorer แสดงโครงสร้างของไฟล์ต่าง ๆ

(3) Output (Errors / Warnings / Messages) เป็นต้น

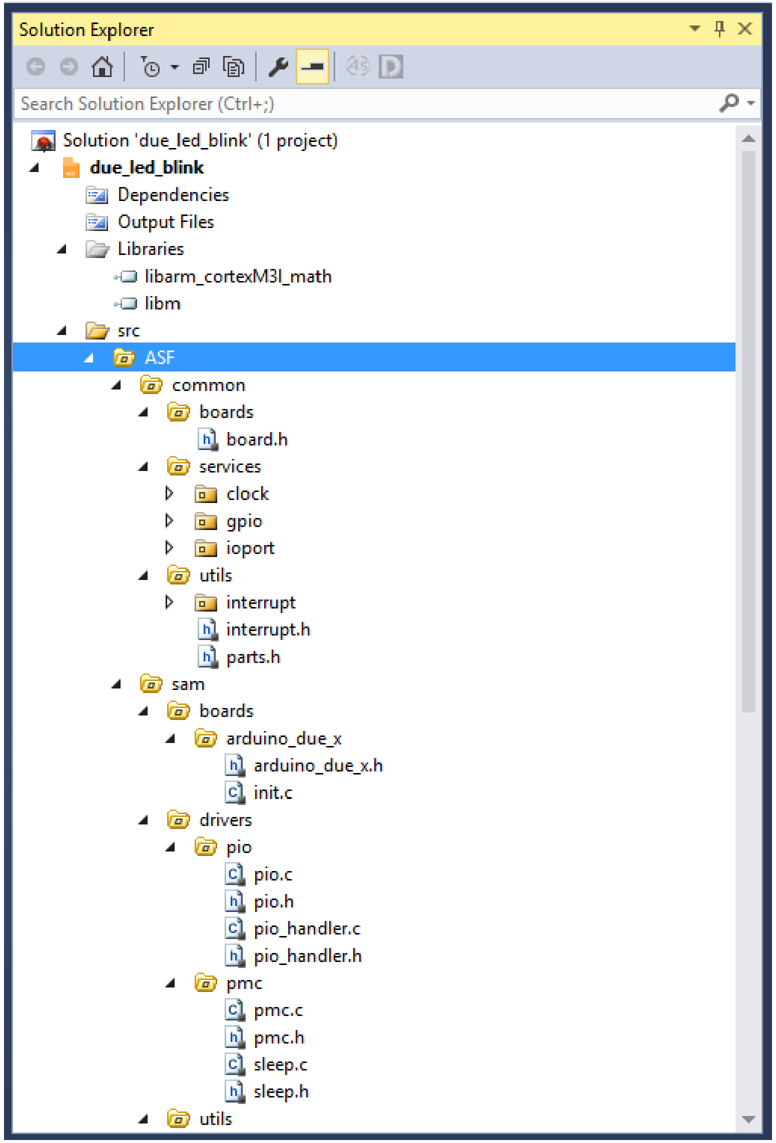
AVR Studio 7 + Arduino DUE



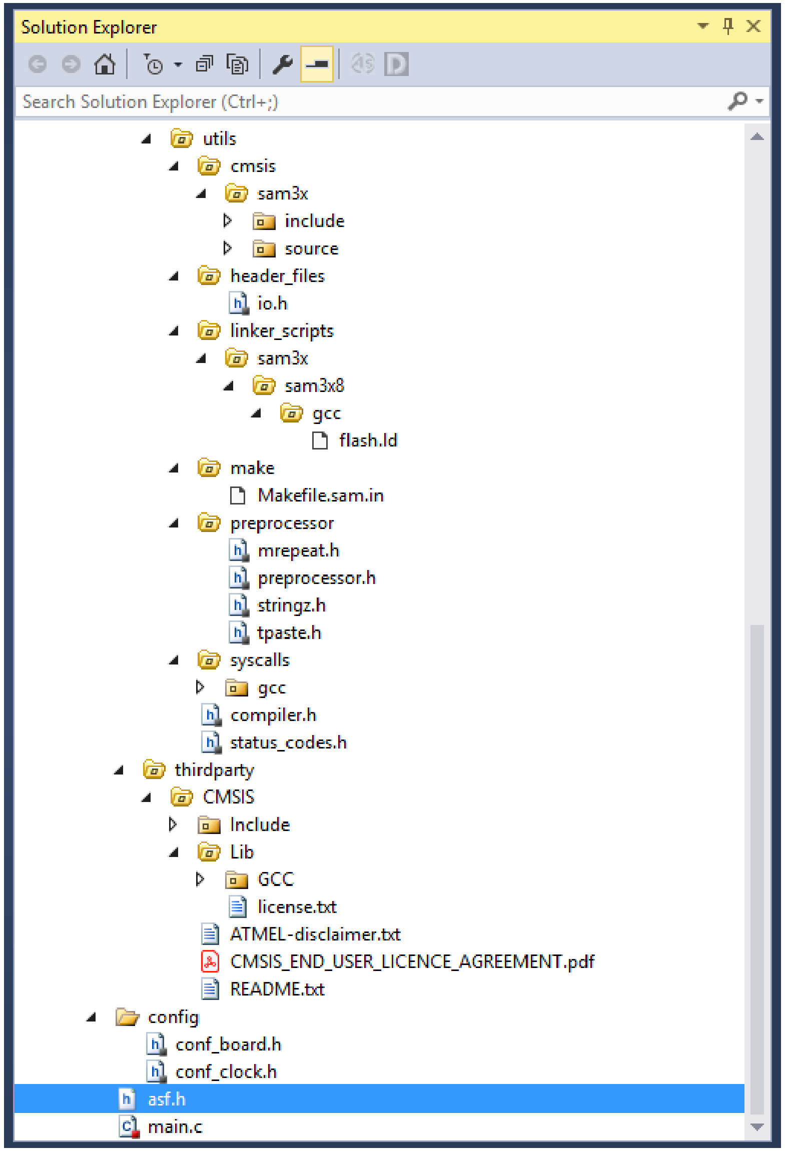
```
/**
 * \file
 *
 * \brief Empty user application template
 *
 */

/**
 * \mainpage User Application template doxygen documentation
 *
 * \par Empty user application template
 *
 * Bare minimum empty user application template
 *
 * \par Content
 *
 * -# Include the ASF header files (through asf.h)
 * -# "Insert system clock initialization code here" comment
 * -# Minimal main function that starts with a call to board_init()
 * -# "Insert application code here" comment
 *
 */

/*
 * Include header files for all drivers that have been imported from
 * Atmel Software Framework (ASF).
 */
/*
 * Support and FAQ: visit <a href="https://www.microchip.com/support/">Microchip Support</a>
 */
#include <asf.h>

int main (void)
```

ลองสำรวจไฟล์ที่เป็นซอร์สโค้ดในไดเรกทอรีย่อยของโปรเจกต์
(เช่น ไฟล์ต่าง ๆ ที่เกี่ยวข้องกับ ASF)

AVR Studio 7 + Arduino DUE



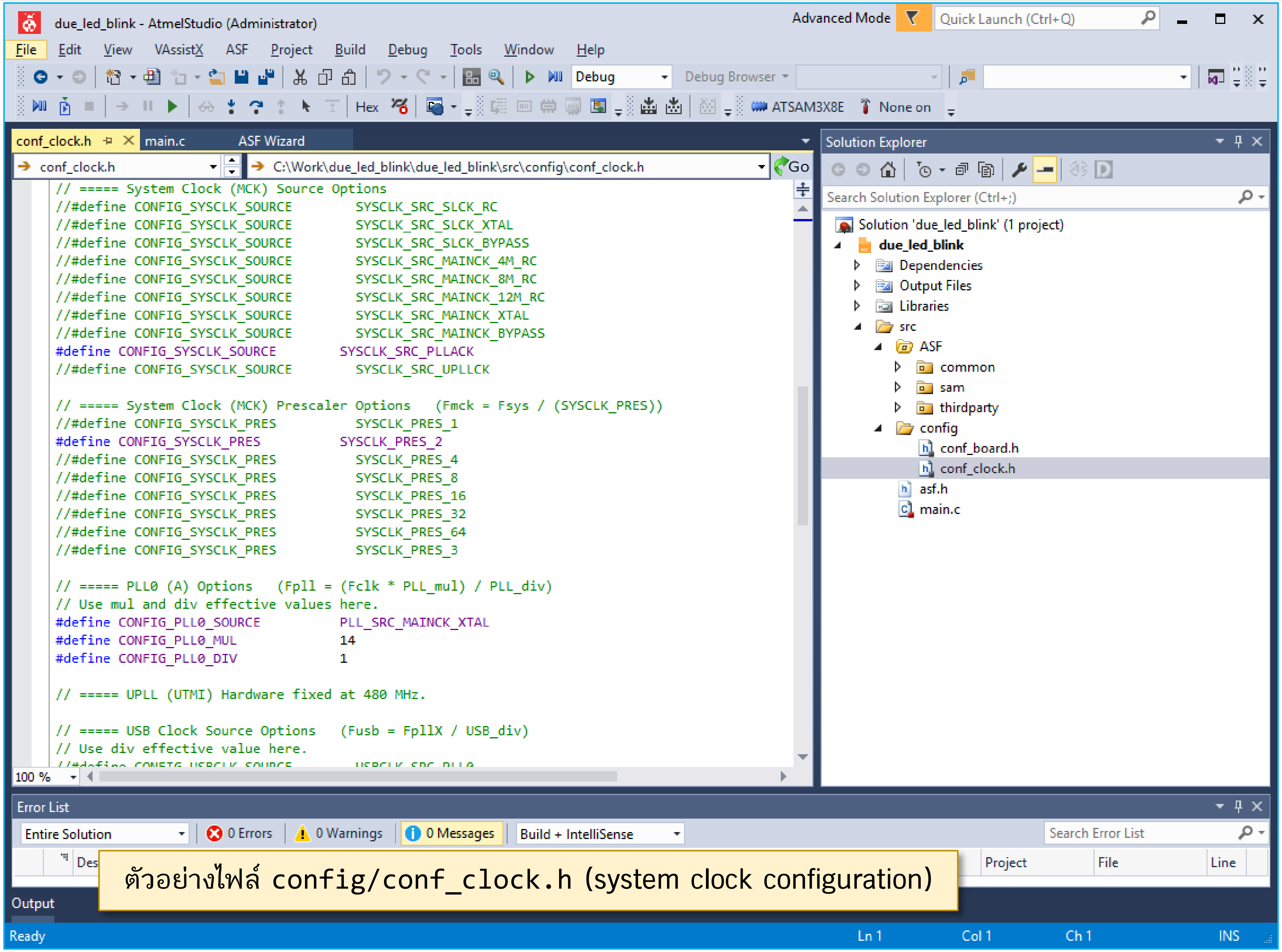
```
Solution 'due_led_blink' (1 project)
◢ due_led_blink
    Dependencies
    Output Files
  ◢ Libraries
      libarm_cortexM3l_math
      libm
  ◢ src
    ◢ ASF
      ◢ common
        ◢ boards
            board.h
        ◢ services
          ▷ clock
          ▷ gpio
          ▷ ioport
        ◢ utils
          ▷ interrupt
            interrupt.h
            parts.h
      ◢ sam
        ◢ boards
          ◢ arduino_due_x
              arduino_due_x.h
              init.c
        ◢ drivers
          ◢ pio
              pio.c
              pio.h
              pio_handler.c
              pio_handler.h
          ◢ pmc
              pmc.c
              pmc.h
              sleep.c
              sleep.h
        ◢ utils
```



```
        ◢ utils
          ◢ cmsis
            ◢ sam3x
              ▷ include
              ▷ source
          ◢ header_files
              io.h
          ◢ linker_scripts
            ◢ sam3x
              ◢ sam3x8
                ◢ gcc
                    flash.ld
          ◢ make
              Makefile.sam.in
          ◢ preprocessor
              mrepeat.h
              preprocessor.h
              stringz.h
              tpaste.h
          ◢ syscalls
            ▷ gcc
            compiler.h
            status_codes.h
      ◢ thirdparty
        ◢ CMSIS
          ▷ Include
          ◢ Lib
            ▷ GCC
              license.txt
            ATMEL-disclaimer.txt
            CMSIS_END_USER_LICENCE_AGREEMENT.pdf
            README.txt
    ◢ config
        conf_board.h
        conf_clock.h
      asf.h
      main.c
```

ให้ลองสำรวจไฟล์ต่าง ๆ ที่เป็นซอร์สโค้ด ในไดเรกทอรีย่อยของโปรเจกต์

AVR Studio 7 + Arduino DUE



```
// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480 MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
```

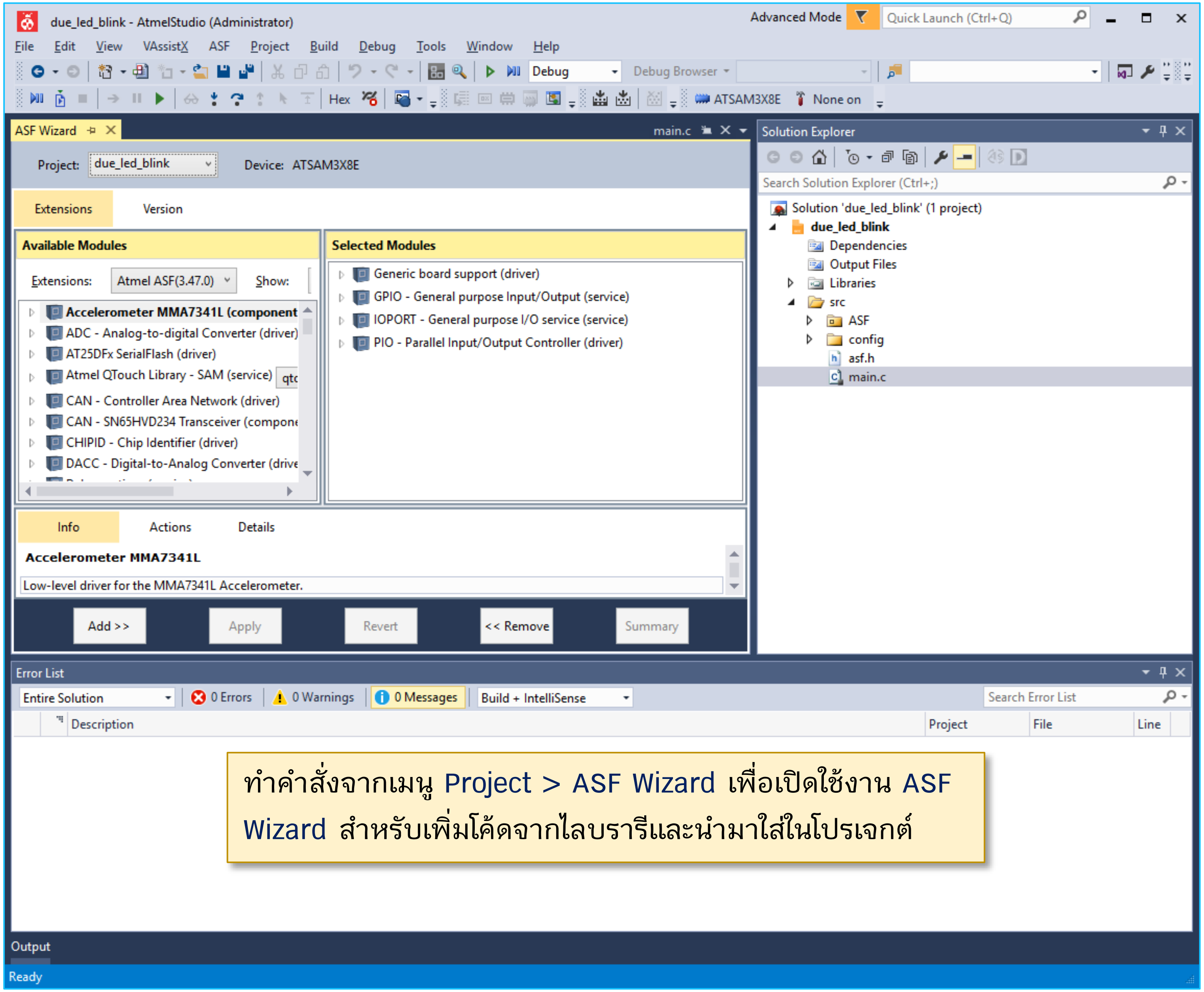
ตัวอย่างไฟล์ config/conf_clock.h (system clock configuration)

AVR Studio 7 + Arduino DUE

```
// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz

// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz
```

คำอธิบาย: การกำหนดค่าสำหรับ Clock Settings ในไฟล์ config_clock.h

AVR Studio 7 + Arduino DUE



Project: due_led_blink Device: ATSAM3X8E

Available Modules

Extensions: Atmel ASF(3.47.0) Show:

- Accelerometer MMA7341L (component
- ADC - Analog-to-digital Converter (driver)
- AT25DFx SerialFlash (driver)
- Atmel QTouch Library - SAM (service) qtc
- CAN - Controller Area Network (driver)
- CAN - SN65HVD234 Transceiver (compone
- CHIPID - Chip Identifier (driver)
- DACC - Digital-to-Analog Converter (drive

Selected Modules

- Generic board support (driver)
- GPIO - General purpose Input/Output (service)
- IOPORT - General purpose I/O service (service)
- PIO - Parallel Input/Output Controller (driver)

Accelerometer MMA7341L

Low-level driver for the MMA7341L Accelerometer.

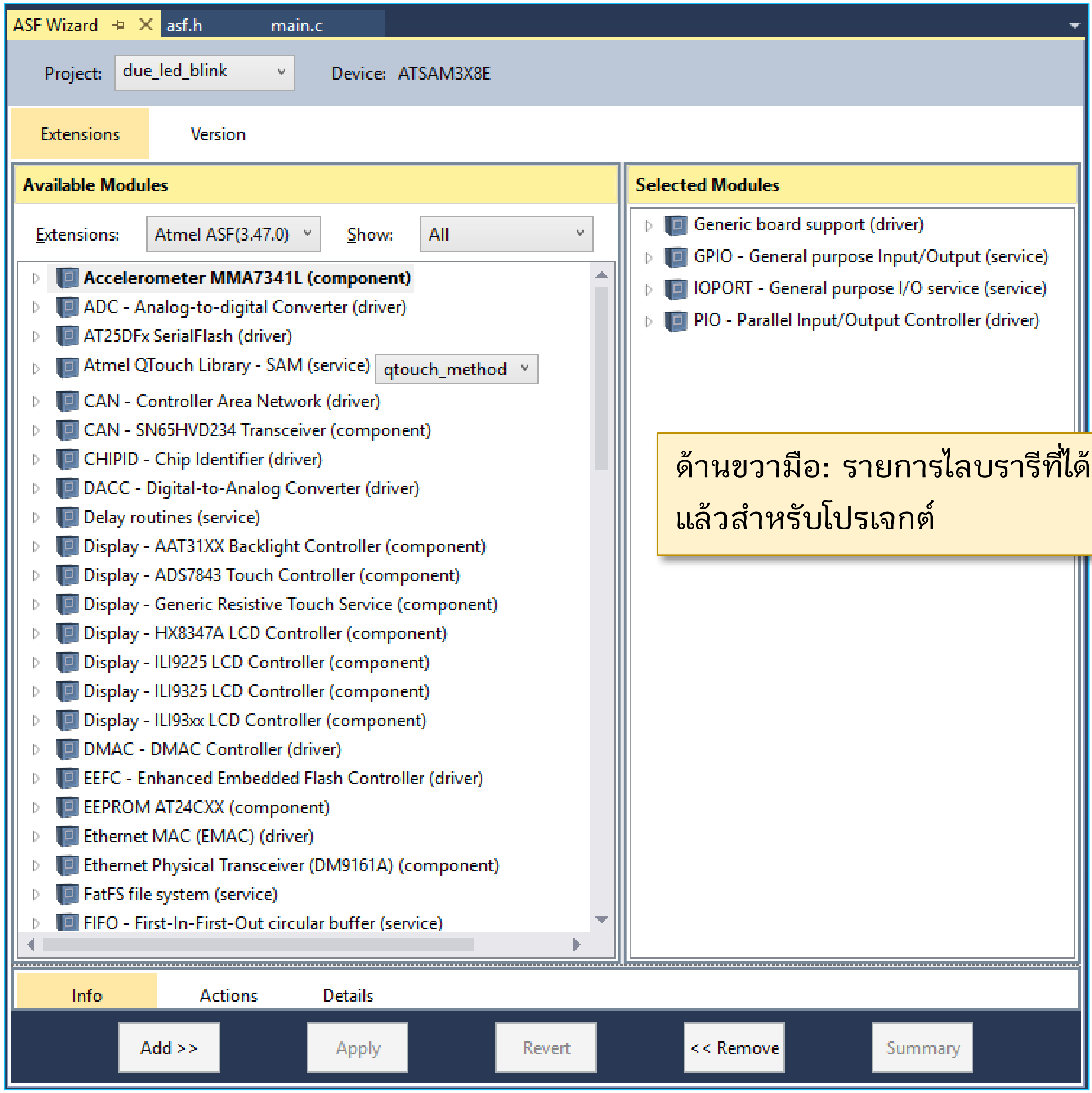
Solution 'due_led_blink' (1 project)
- due_led_blink
  - Dependencies
  - Output Files
  - Libraries
  - src
    - ASF
    - config
    - asf.h
    - main.c

Entire Solution | 0 Errors | 0 Warnings | 0 Messages | Build + IntelliSense

	Description	Project	File	Line

ทำคำสั่งจากเมนู Project > ASF Wizard เพื่อเปิดใช้งาน ASF Wizard สำหรับเพิ่มโค้ดจากไลบรารีและนำมาใส่ในโปรเจกต์

AVR Studio 7 + Arduino DUE



ASF Wizard | asf.h | main.c

Project: due_led_blink

Device: ATSAM3X8E

Extensions: Atmel ASF(3.47.0)

Show: All

Available Modules:

- Accelerometer MMA7341L (component)
- ADC - Analog-to-digital Converter (driver)
- AT25DFx SerialFlash (driver)
- Atmel QTouch Library - SAM (service) qtouch_method
- CAN - Controller Area Network (driver)
- CAN - SN65HVD234 Transceiver (component)
- CHIPID - Chip Identifier (driver)
- DACC - Digital-to-Analog Converter (driver)
- Delay routines (service)
- Display - AAT31XX Backlight Controller (component)
- Display - ADS7843 Touch Controller (component)
- Display - Generic Resistive Touch Service (component)
- Display - HX8347A LCD Controller (component)
- Display - ILI9225 LCD Controller (component)
- Display - ILI9325 LCD Controller (component)
- Display - ILI93xx LCD Controller (component)
- DMAC - DMAC Controller (driver)
- EEFC - Enhanced Embedded Flash Controller (driver)
- EEPROM AT24CXX (component)
- Ethernet MAC (EMAC) (driver)
- Ethernet Physical Transceiver (DM9161A) (component)
- FatFS file system (service)
- FIFO - First-In-First-Out circular buffer (service)

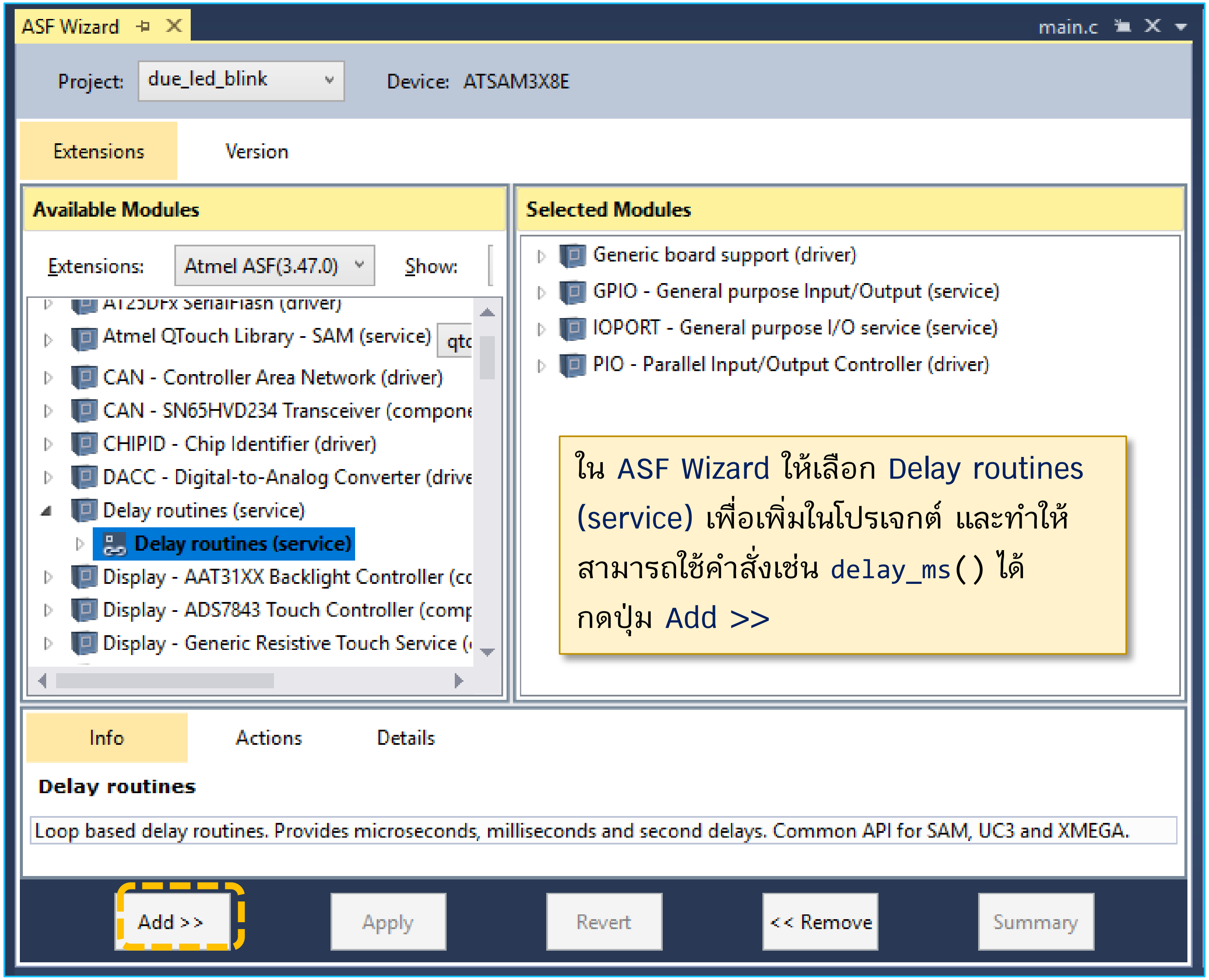
Selected Modules:

- Generic board support (driver)
- GPIO - General purpose Input/Output (service)
- IOPORT - General purpose I/O service (service)
- PIO - Parallel Input/Output Controller (driver)

ด้านซ้ายมือ: รายการไลบรารีต่าง ๆ ที่สามารถเลือกมาใช้ได้กับ ATSAM3X8E

ด้านขวามือ: รายการไลบรารีที่ได้เลือกมาใส่ไว้แล้วสำหรับโปรเจกต์

AVR Studio 7 + Arduino DUE


ASF Wizard
main.c
Extensions
Version
Available Modules
Extensions:
Show:
AT25DFx SerialFlash (driver)
Atmel QTouch Library - SAM (service)
qtc
CAN - Controller Area Network (driver)
CAN - SN65HVD234 Transceiver (compone
CHIPID - Chip Identifier (driver)
DACC - Digital-to-Analog Converter (drive
Delay routines (service)
Delay routines (service)
Display - AAT31XX Backlight Controller (cc
Display - ADS7843 Touch Controller (comp
Display - Generic Resistive Touch Service (
Selected Modules
Generic board support (driver)
GPIO - General purpose Input/Output (service)
IOPORT - General purpose I/O service (service)
PIO - Parallel Input/Output Controller (driver)
Info
Actions
Details
Add >>
Apply
Revert
<< Remove
Summary

Project: due_led_blink

Device: ATSAM3X8E

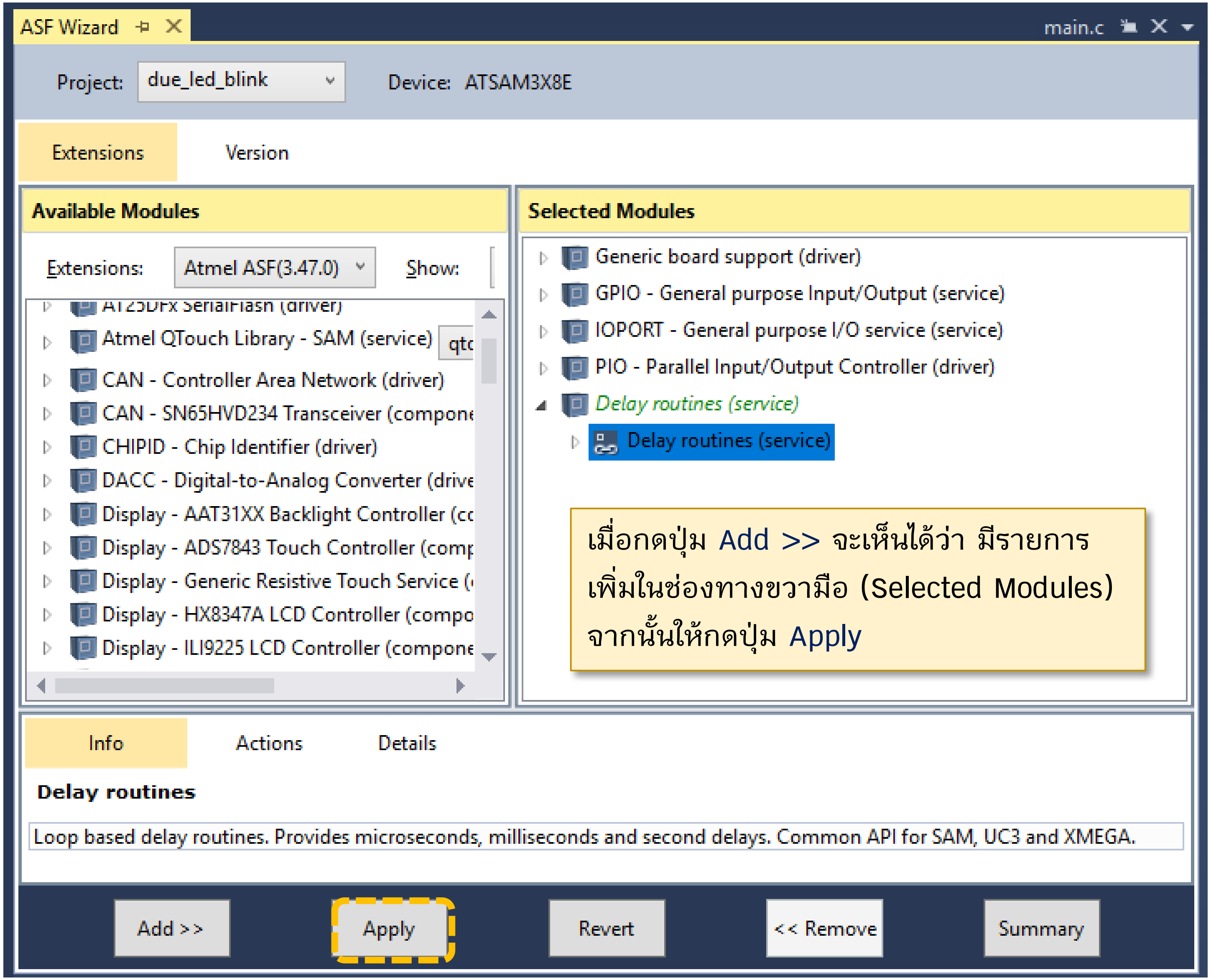
Extensions: Atmel ASF(3.47.0)

ใน ASF Wizard ให้เลือก Delay routines (service) เพื่อเพิ่มในโปรเจกต์ และทำให้สามารถใช้คำสั่งเช่น `delay_ms()` ได้ กดปุ่ม Add >>

Delay routines

Loop based delay routines. Provides microseconds, milliseconds and second delays. Common API for SAM, UC3 and XMEGA.

AVR Studio 7 + Arduino DUE



Project: due_led_blink

Device: ATSAM3X8E

Extensions: Atmel ASF(3.47.0)

Show:

Available Modules:

- AT25DFx SerialFlash (driver)
- Atmel QTouch Library - SAM (service) qtc
- CAN - Controller Area Network (driver)
- CAN - SN65HVD234 Transceiver (compone
- CHIPID - Chip Identifier (driver)
- DACC - Digital-to-Analog Converter (drive
- Display - AAT31XX Backlight Controller (cc
- Display - ADS7843 Touch Controller (comp
- Display - Generic Resistive Touch Service (
- Display - HX8347A LCD Controller (compo
- Display - ILI9225 LCD Controller (compone

Selected Modules:

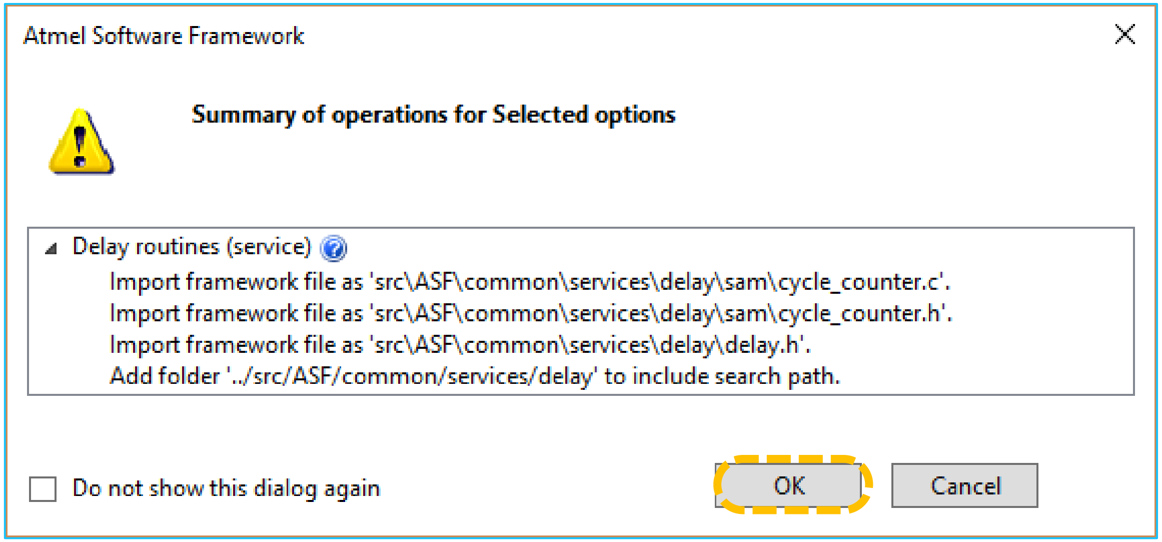
- Generic board support (driver)
- GPIO - General purpose Input/Output (service)
- IOPORT - General purpose I/O service (service)
- PIO - Parallel Input/Output Controller (driver)
- *Delay routines (service)*
  - Delay routines (service)

เมื่อกดปุ่ม Add >> จะเห็นได้ว่า มีรายการ
เพิ่มในช่องทางขวามือ (Selected Modules)
จากนั้นให้กดปุ่ม Apply

Delay routines

Loop based delay routines. Provides microseconds, milliseconds and second delays. Common API for SAM, UC3 and XMEGA.

AVR Studio 7 + Arduino DUE



Summary of operations for Selected options

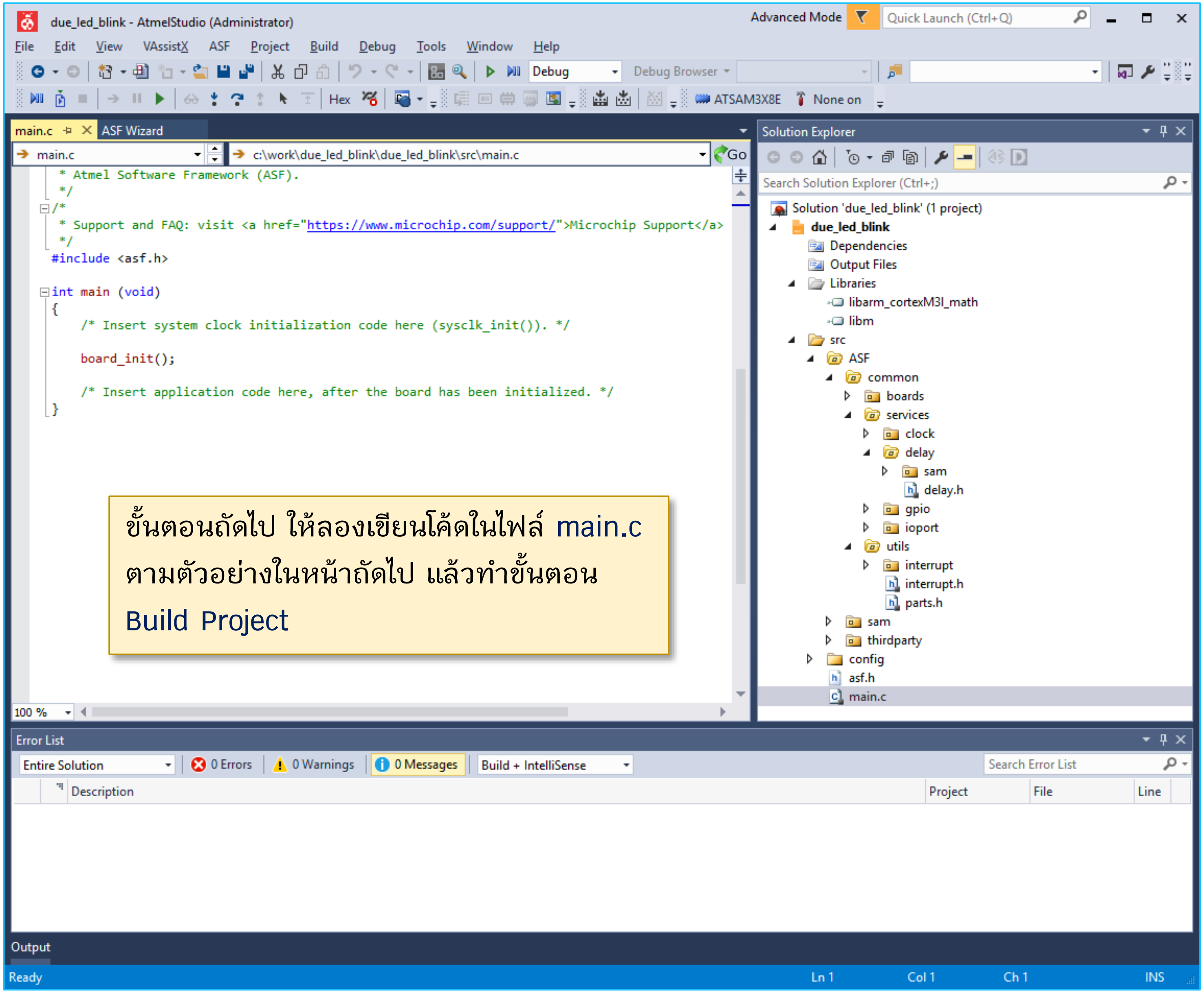
Delay routines (service)

```
Import framework file as 'src\ASF\common\services\delay\sam\cycle_counter.c'.
Import framework file as 'src\ASF\common\services\delay\sam\cycle_counter.h'.
Import framework file as 'src\ASF\common\services\delay\delay.h'.
Add folder '../src/ASF/common/services/delay' to include search path.
```

□ Do not show this dialog again

สังเกตรายชื่อไฟล์ (.c และ .h) สำหรับ delay routines ที่จะถูกนำมาใส่เพิ่มในโปรเจกต์ จากนั้นให้กดปุ่ม OK เพื่อดำเนินการต่อ

AVR Studio 7 + Arduino DUE



```
 * Atmel Software Framework (ASF).
 */
/*
 * Support and FAQ: visit <a href="https://www.microchip.com/support/">Microchip Support</a>
 */
#include <asf.h>

int main (void)
{
	/* Insert system clock initialization code here (sysclk_init()). */

	board_init();

	/* Insert application code here, after the board has been initialized. */
}
```

ขั้นตอนถัดไป ให้ลองเขียนโค้ดในไฟล์ main.c ตามตัวอย่างในหน้าถัดไป แล้วทำขั้นตอน Build Project

Sample Code:Onboard LED Blink (version 1)

```
#include <asf.h>

#define LED  IOPORT_CREATE_PIN(PIOB, 27) // PWM13 is on PB27, pin 68

int main(void) {
  sysclk_init(); // initialize the system clock (=> 84 MHz)
  board_init();  // initialize board (e.g. GPIOs and other peripherals)

  ioport_set_pin_dir( LED, IOPORT_DIR_OUTPUT );

  while(1) {
     ioport_set_pin_level( LED, IOPORT_PIN_LEVEL_HIGH  );
     delay_ms(500);
     ioport_set_pin_level( LED, IOPORT_PIN_LEVEL_LOW );
     delay_ms(500);
  }
}
```

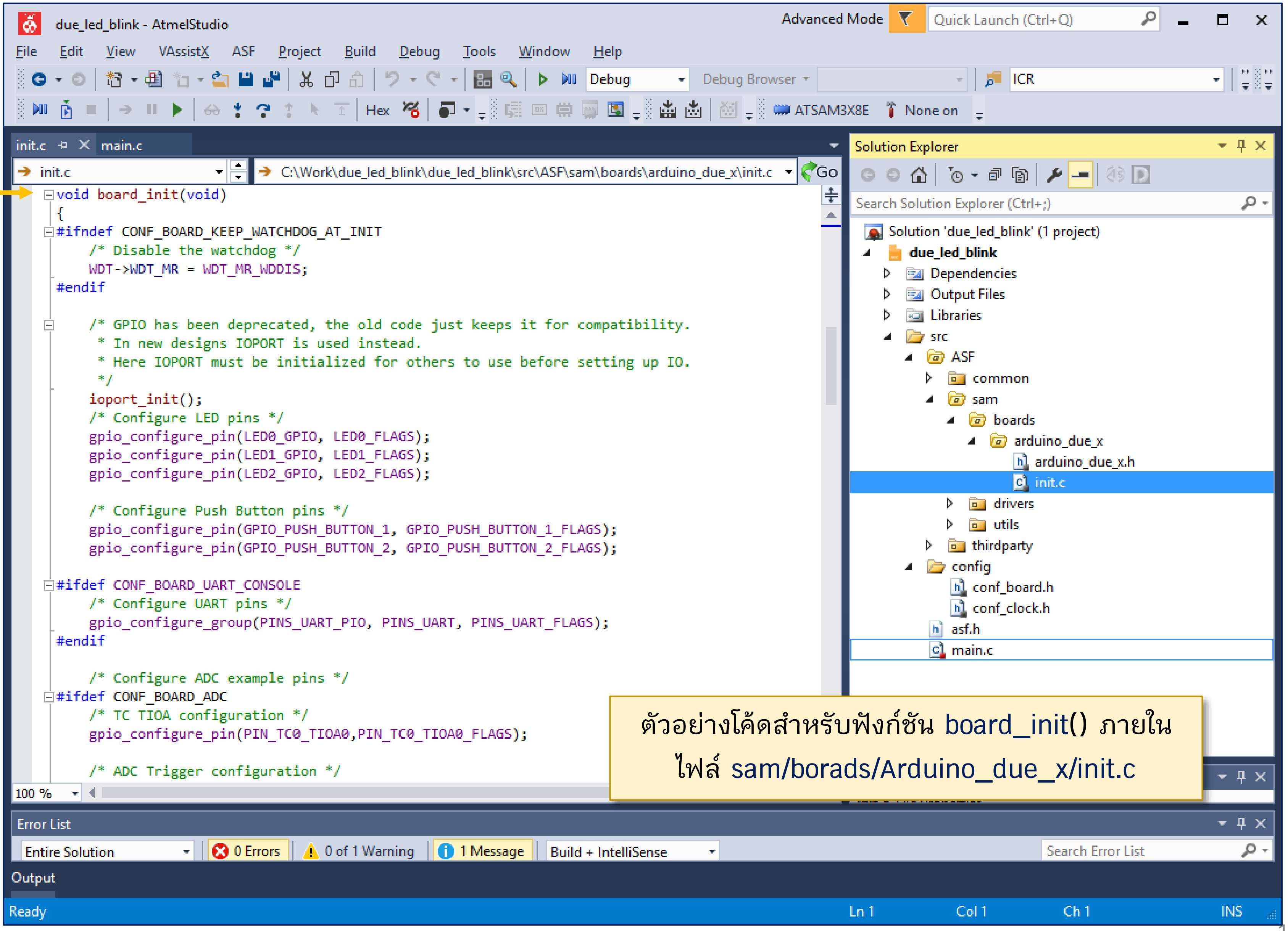
ตัวอย่างโค้ดสำหรับ main.c สาธิตการทำงานโดยทำให้ LED กระพริบได้ ซึ่งมีอยู่บนบอร์ด Arduino DUE และเป็นเอาต์พุตที่ขา PB27

Sample Code:Onboard LED Blink (version 2)

```
#include <asf.h>

#define LED  IOPORT_CREATE_PIN(PIOB, 27) // PWM13 is on PB27, pin 68

int main(void) {
  sysclk_init(); // initialize the system clock (=> 84 MHz)
  board_init();  // initialize board (e.g. GPIOs and selected peripherals)

  ioport_set_pin_dir( LED, IOPORT_DIR_OUTPUT );

  while(1) {
    ioport_toggle_pin_level( LED ); // toggle the LED
    delay_ms(500);
  }
}}
```

ตัวอย่างโค้ดสำหรับ main.c สาธิตการทำงานโดยทำให้ LED กระพริบได้ ซึ่งมีอยู่บนบอร์ด Arduino DUE และเป็นเอาต์พุตที่ขา PB27

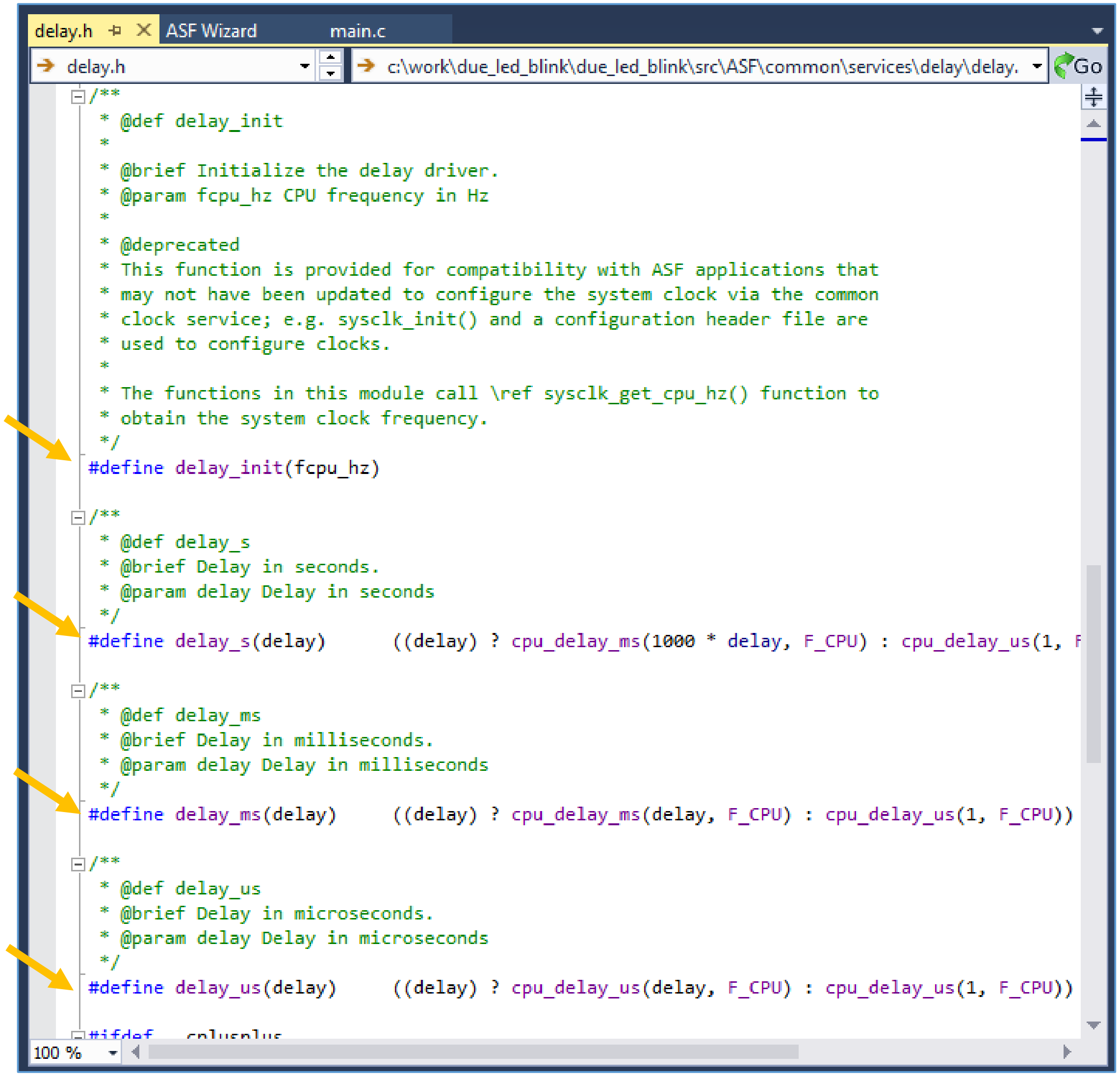
AVR Studio 7 + Arduino DUE



```c
void board_init(void)
{
#ifndef CONF_BOARD_KEEP_WATCHDOG_AT_INIT
    /* Disable the watchdog */
    WDT->WDT_MR = WDT_MR_WDDIS;
#endif

    /* GPIO has been deprecated, the old code just keeps it for compatibility.
     * In new designs IOPORT is used instead.
     * Here IOPORT must be initialized for others to use before setting up IO.
     */
    ioport_init();
    /* Configure LED pins */
    gpio_configure_pin(LED0_GPIO, LED0_FLAGS);
    gpio_configure_pin(LED1_GPIO, LED1_FLAGS);
    gpio_configure_pin(LED2_GPIO, LED2_FLAGS);

    /* Configure Push Button pins */
    gpio_configure_pin(GPIO_PUSH_BUTTON_1, GPIO_PUSH_BUTTON_1_FLAGS);
    gpio_configure_pin(GPIO_PUSH_BUTTON_2, GPIO_PUSH_BUTTON_2_FLAGS);

#ifdef CONF_BOARD_UART_CONSOLE
    /* Configure UART pins */
    gpio_configure_group(PINS_UART_PIO, PINS_UART, PINS_UART_FLAGS);
#endif

    /* Configure ADC example pins */
#ifdef CONF_BOARD_ADC
    /* TC TIOA configuration */
    gpio_configure_pin(PIN_TC0_TIOA0,PIN_TC0_TIOA0_FLAGS);

    /* ADC Trigger configuration */
```

ตัวอย่างโค้ดสำหรับฟังก์ชัน board_init() ภายใน
ไฟล์ sam/borads/Arduino_due_x/init.c

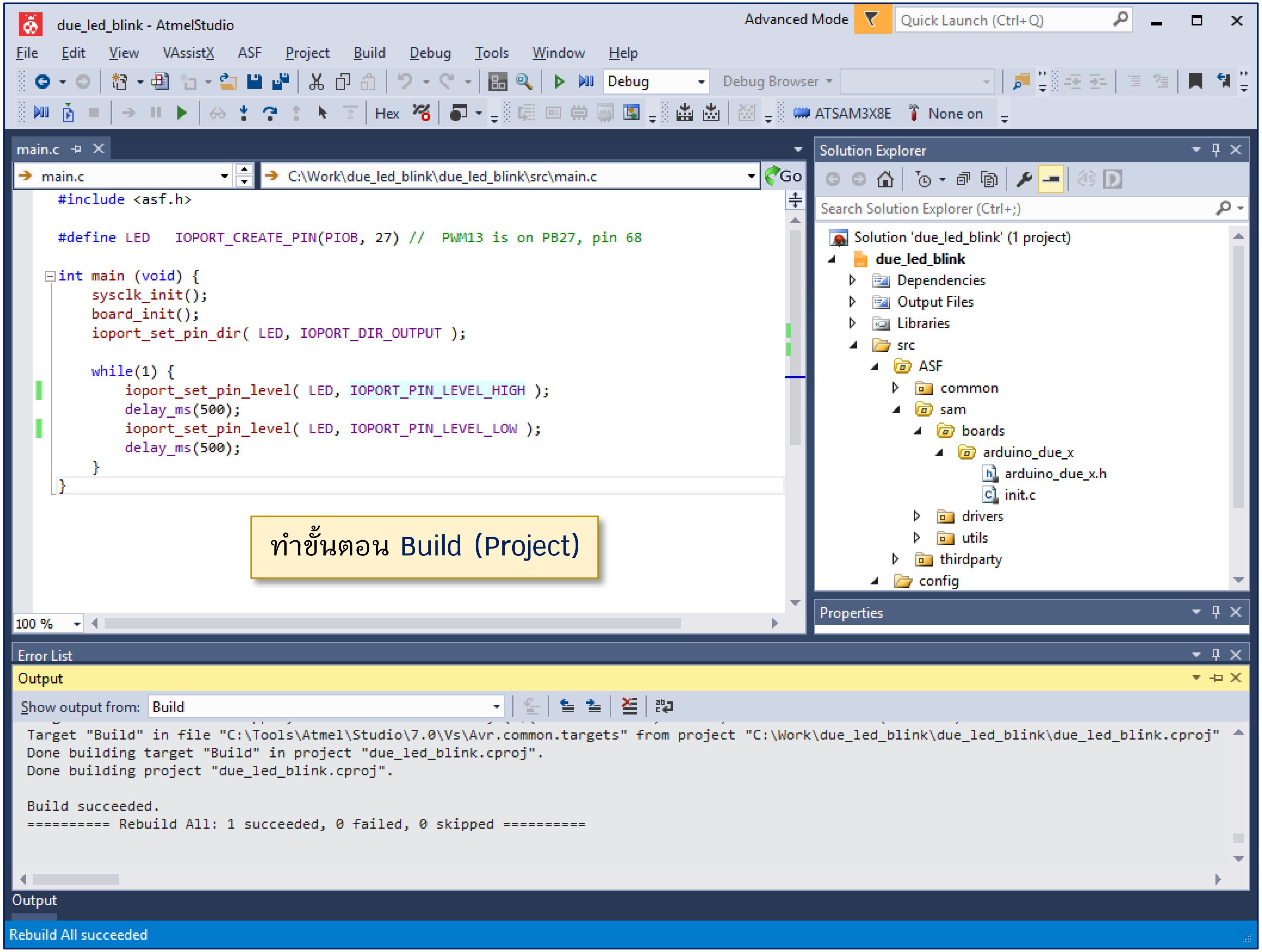
AVR Studio 7 + Arduino DUE


delay.h
ASF Wizard
main.c
delay.h
c:\work\due_led_blink\due_led_blink\src\ASF\common\services\delay\delay.
Go
100 %

```
/**
 * @def delay_init
 *
 * @brief Initialize the delay driver.
 * @param fcpu_hz CPU frequency in Hz
 *
 * @deprecated
 * This function is provided for compatibility with ASF applications that
 * may not have been updated to configure the system clock via the common
 * clock service; e.g. sysclk_init() and a configuration header file are
 * used to configure clocks.
 *
 * The functions in this module call \ref sysclk_get_cpu_hz() function to
 * obtain the system clock frequency.
 */
#define delay_init(fcpu_hz)

/**
 * @def delay_s
 * @brief Delay in seconds.
 * @param delay Delay in seconds
 */
#define delay_s(delay)      ((delay) ? cpu_delay_ms(1000 * delay, F_CPU) : cpu_delay_us(1, F

/**
 * @def delay_ms
 * @brief Delay in milliseconds.
 * @param delay Delay in milliseconds
 */
#define delay_ms(delay)     ((delay) ? cpu_delay_ms(delay, F_CPU) : cpu_delay_us(1, F_CPU))

/**
 * @def delay_us
 * @brief Delay in microseconds.
 * @param delay Delay in microseconds
 */
#define delay_us(delay)     ((delay) ? cpu_delay_us(delay, F_CPU) : cpu_delay_us(1, F_CPU))

#ifdef __cplusplus
```

ตัวอย่างโค้ดภายในไฟล์ delay.h

AVR Studio 7 + Arduino DUE



```c
#include <asf.h>

#define LED   IOPORT_CREATE_PIN(PIOB, 27) //  PWM13 is on PB27, pin 68

int main (void) {
    sysclk_init();
    board_init();
    ioport_set_pin_dir( LED, IOPORT_DIR_OUTPUT );

    while(1) {
        ioport_set_pin_level( LED, IOPORT_PIN_LEVEL_HIGH );
        delay_ms(500);
        ioport_set_pin_level( LED, IOPORT_PIN_LEVEL_LOW );
        delay_ms(500);
    }
}
```

ทำขั้นตอน Build (Project)

Show output from: Build

```
Target "Build" in file "C:\Tools\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Work\due_led_blink\due_led_blink\due_led_blink.cproj"
Done building target "Build" in project "due_led_blink.cproj".
Done building project "due_led_blink.cproj".

Build succeeded.
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Rebuild All succeeded

Setting Commands for Arduino DUE Programming

สร้างไฟล์ใหม่ดังนี้

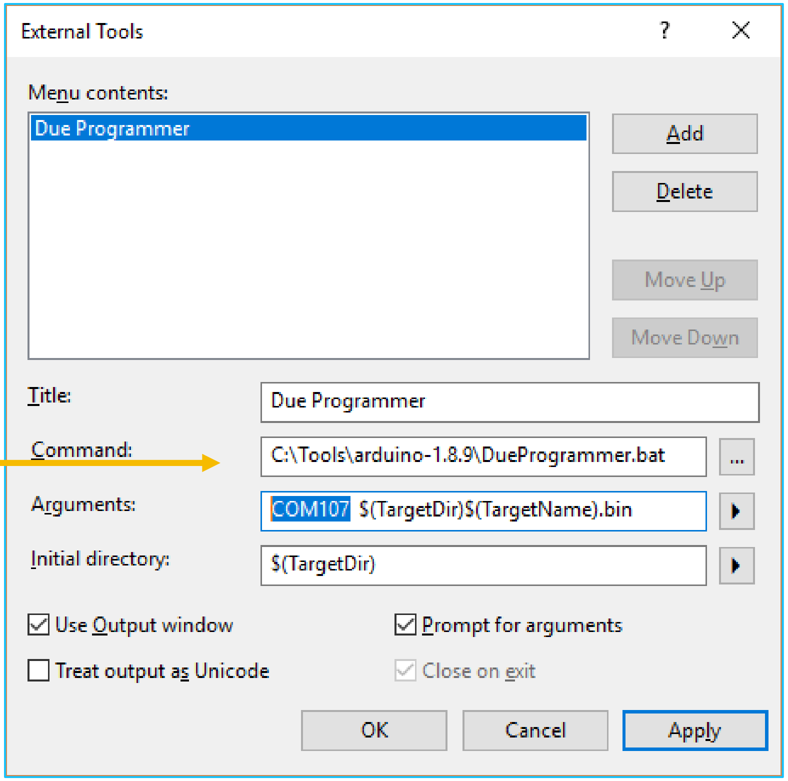
File: DueProgrammer.bat

```
@echo OFF
mode %1:1200,n,8,1,p > nul
set ARDUINO_PATH="C:\Users\%USERNAME%\AppData\Local\Arduino15"
%ARDUINO_PATH%\packages\arduino\tools\bossac\1.7.0\bossac.exe ^
  --port=%1 -U false -e -w -v -b %2 -R
```

สร้างไฟล์ DueProgrammer.bat ตามตัวอย่าง เพื่อเรียกใช้จาก AVR Studio สำหรับอัปโหลดไฟล์ .bin ที่ได้จากการคอมไพล์ ไปยังบอร์ด Arduino DUE เครื่องคอมพิวเตอร์จะต้องมีการติดตั้งโปรแกรม Arduino IDE แล้ว เนื่องจากจะต้องเรียกใช้ คำสั่ง bossac.exe ซึ่งเป็น Software Tool ของ Arduino

หมายเลข COM port อาจเปลี่ยนแปลงได้ขึ้นอยู่กับบอร์ด Arduino DUE ที่เชื่อมต่อกับคอมพิวเตอร์ขณะใช้งาน ดังนั้นจะต้องกำหนดหมายเลขพอร์ตให้ถูกต้อง (ให้เชื่อมต่อกับ Programming Port ของ Arduino DUE ด้วยสาย microUSB)

ทำคำสั่งจากเมนู Tools > External Tools เพื่อเพิ่มรายการคำสั่ง Due Programmer



External Tools

Menu contents:

Due Programmer

Title: Due Programmer

Command: C:\Tools\arduino-1.8.9\DueProgrammer.bat

Arguments: COM107 $(TargetDir)$(TargetName).bin

Initial directory: $(TargetDir)

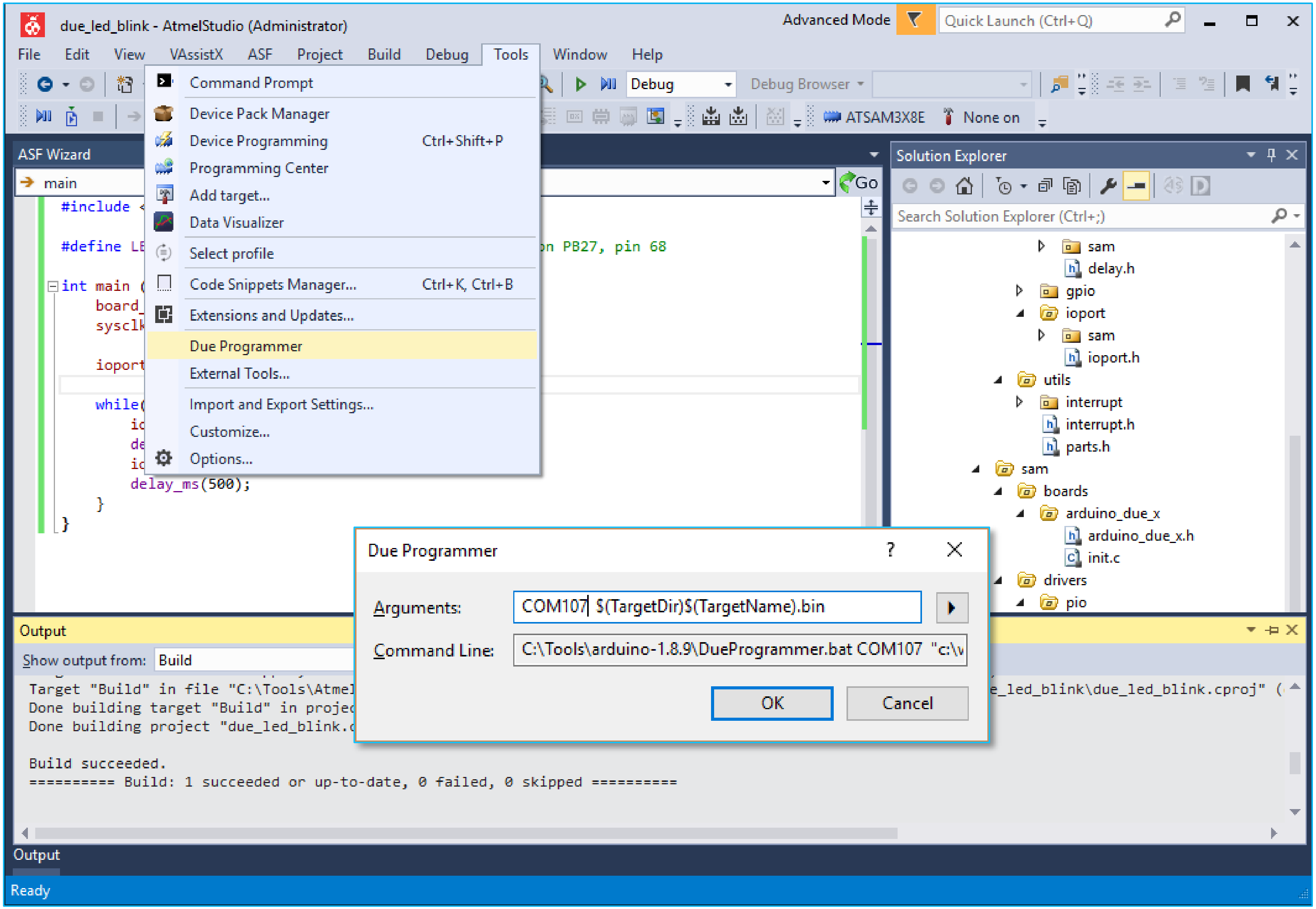
☑ Use Output window

☑ Prompt for arguments

☐ Treat output as Unicode

☑ Close on exit

Arduino DUE Programming



```
#include <
#define LE                                   on PB27, pin 68
int main (
    board_
    sysclk
    ioport
    while(
        io
        de
        io
        delay_ms(500);
    }
}
```

Due Programmer

Arguments: COM107 $(TargetDir)$(TargetName).bin

Command Line: C:\Tools\arduino-1.8.9\DueProgrammer.bat COM107 "c:\v

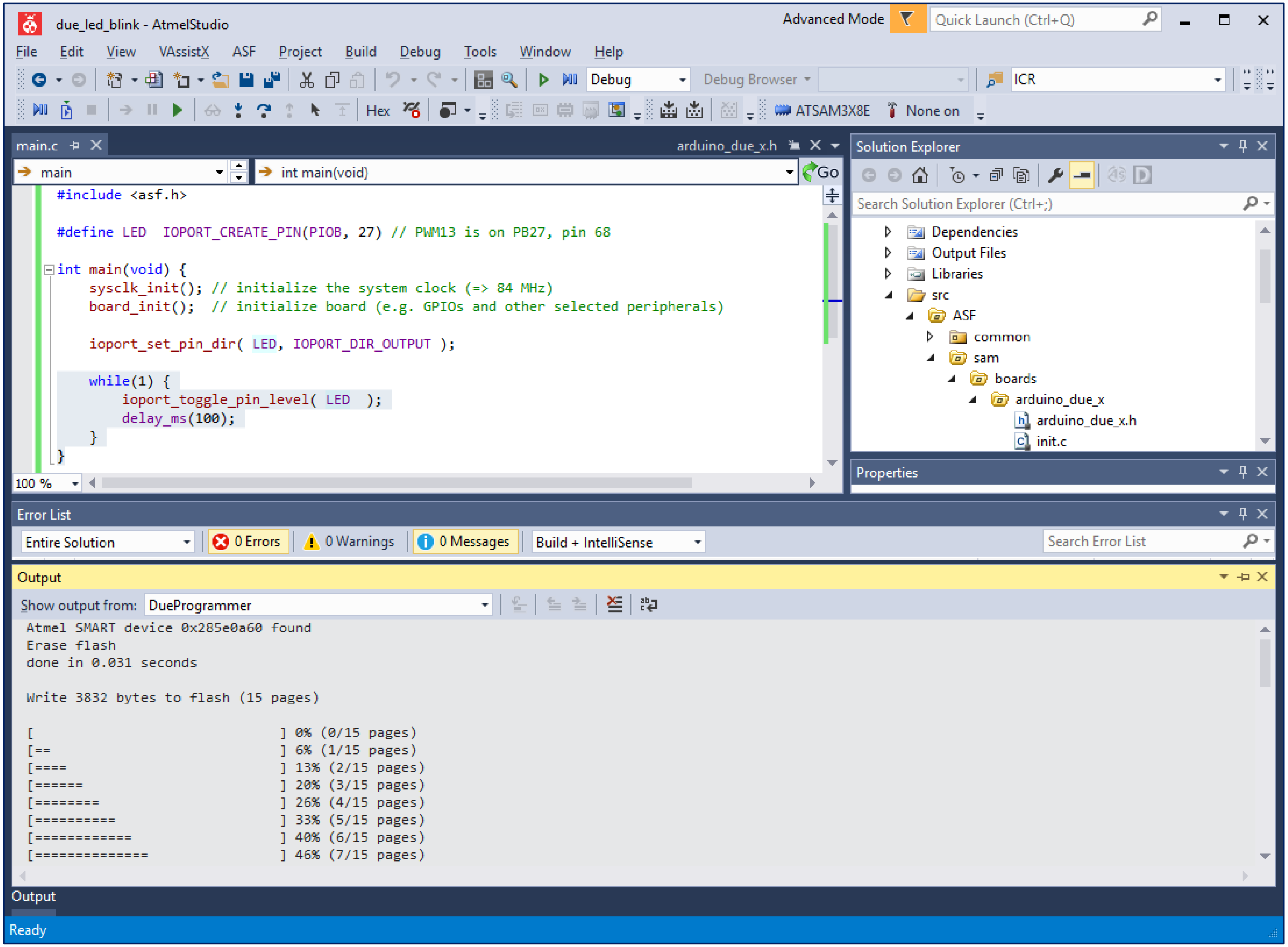
OK Cancel

Output

Show output from: Build

```
Target "Build" in file "C:\Tools\Atmel                                e_led_blink\due_led_blink.cproj" (
Done building target "Build" in projec
Done building project "due_led_blink.c

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Arduino DUE Programming



```
#include <asf.h>

#define LED  IOPORT_CREATE_PIN(PIOB, 27) // PWM13 is on PB27, pin 68

int main(void) {
    sysclk_init(); // initialize the system clock (=> 84 MHz)
    board_init();  // initialize board (e.g. GPIOs and other selected peripherals)

    ioport_set_pin_dir( LED, IOPORT_DIR_OUTPUT );

    while(1) {
        ioport_toggle_pin_level( LED  );
        delay_ms(100);
    }
}
```

```
Atmel SMART device 0x285e0a60 found
Erase flash
done in 0.031 seconds

Write 3832 bytes to flash (15 pages)

[                              ] 0% (0/15 pages)
[==                            ] 6% (1/15 pages)
[====                          ] 13% (2/15 pages)
[======                        ] 20% (3/15 pages)
[========                      ] 26% (4/15 pages)
[==========                    ] 33% (5/15 pages)
[============                  ] 40% (6/15 pages)
[==============                ] 46% (7/15 pages)
```